2015-01-01

Computation Offloading Decisions For Reducing Completion Time

Salvador Melendez
University of Texas at El Paso, smelendez5@miners.utep.edu

COMPUTATION OFFLOADING DECISIONS FOR REDUCING COMPLETION TIME

SALVADOR MELENDEZ

Department of Electrical and Computer Engineering

APPROVED:

Michael P. McGarry, Ph.D., Chair

Patricia J. Teller, Ph.D.

John A. Moya, Ph.D

Virgilio E. Gonzalez, Ph.D

Charles H. Ambler, Ph.D.
Dean of the Graduate School

A mis papás que siempre me han apoyado con mucho esfuerzo y dedicación en todos mis proyectos a lo largo de mi vida.

COMPUTATION OFFLOADING DECISIONS FOR REDUCING COMPLETION TIME

By

SALVADOR MELENDEZ, M.S.Cp.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

December 2015

Acknowledgments

I would first like to thank my advisors Dr. Michael McGarry and Dr. Patricia Teller for their invaluable guidance that help me to grow as an individual. Dr. McGarry and Dr. Teller have been more than just academic advisors, they have also served as mentors and guide in personal matters, always willing to provide advice and share their experiences with their students.

Dr. McGarry and Dr. Teller have also provided a research assistantship which allowed me pursue my degree for which I will always be really grateful.

I would also like to extend my gratitude to Dr. John Moya and Dr. Virgilio Gonzalez for kindly agreeing to serve in my dissertation committee.

Finally, I would like to thank the UTEP Graduate School, the Hispanic Scholarship Fund (HSF) and Google for their support in funding my education.

Abstract

Mobile devices are being widely used in many applications such as image processing, computer vision (e.g. face detection and recognition), wearable computing, language translation, and battlefield operations. However, mobile devices are constrained in terms of their battery life, processor performance, storage capacity, and network bandwidth. To overcome these issues, there is an approach called *Computation Offloading*, also known as *cyber-foraging* and *surrogate computing*. Computation offloading consists of migrating computational jobs from a mobile device to more powerful remote computing resources. Upon completion of the job, the results are sent back to the mobile device. However, a decision must be made; whether to execute a job locally or remotely, and consider the execution time of the job and the time to move the input and output data over a network (e.g. Internet).

This dissertation presents a computation offloading decision analysis using our system model to identify the conditions under which computation offloading reduces job completion time. We show the results of our numerical experiments using our system model to identify when computation offloading reduces job completion time. We conduct the same analysis using a real workload trace. Lastly, we present a network delay estimate decision analysis to find the effect of the network delay estimate error on the job completion time, and determine how accurate a network delay estimator has to be. This work is backed up with Monte Carlo simulations and behavioral simulations using the discrete event network simulator *NS-3*.

We have found that the bottleneck transmission rate has to be at least the same order of magnitude as the file size to benefit from computation offloading. We have observed that under poor network connections and using powerful hand-held devices, computation offloading of some applications to remote servers becomes less beneficial; however, cloudlets become more attractive. We have also identified real world applications that benefit from computation offloading from a real workload trace. Finally, we have identified the inflection point that impacts the average decision difference when adding error in the estimate of the communication cost.

Table of Contents

Page

List of Tables

List of Figures

List of Symbols

AOP	Aspect-Oriented Programming
API	Application Program Interface
AWS	Amazon Web Services
CC	Cloud Client
CCR	Computing to Communication Ratio
CLC	Cloud Controller
COCA	Computation Offload to Clouds using AOP
CSP	Cloud Service Provider
CWC	Computing While Charging
D2C	Data Center to Client
D2D	Data Center to Data Center
DNS	Domain Name System
EBS	Elastic Block Store
EC2	Elastic Compute Cloud
EMR	Elastic Map Reduce
GUI	Graphical User Interface
HCI	Human Computer Interaction
HSM	Hardware Security Module
IaaS	Infrastructure as a Service
IAM	Identify and Access Management
ICA	Independent Computing Architecture
IPC	Inter-Process Communication
IPS	Instructions Per Second
KMS	Key Management Service
LAN	Local Area Network

MANET	Mobile Ad Hoc Network
MECT	Minimum Expected Completion Time
MPI	Message Passing Interface
NC	Node Controller
NFS	Network File System
OTP	over-the-top
PaaS	Platform as a Service
PoP	Point of Presence
QoE	Quality of Experience
RDP	Remote Display Protocol
RDS	Relational Database Service
RLR	Remote to Local Ratio
RPC	Remote Procedure Call
S3	Simple Storage Service
SaaS	Software as a Service
SC	Storage Controller
SES	Simple Email Service
SLA	Service Level Agreement
SNS	Simple Notification Service
SOD	Stack On-Demand
SOS	Scalable Object Storage
SQ	Shortest Queue
SQS	Simple Queue Service
SWF	Simple Workflow Service
TACC	Texas Advanced Computing Center
ToR	Top-of-the-Rack

UI	User Interface
VM	Virtual Machine
VNC	Virtual Network Computing
VPC	Virtual Private Cloud

Chapter 1: Introduction

1.1 Motivation

Mobile devices are widely used in many applications that require intensive computation and storage. These applications include: a) mobile augmented reality (including computer vision), b) wearable computing, c) image processing, d) language translation, and e) battlefield operations.

Mobile augmented reality applications perform image based object recognition [2] by using the camera, other sensors, image processing and computer vision to identify the user's environment and overlay of contextual information (e.g. street names, description of objects, directions to drive) of the surroundings after the processing. These applications can help people to easily identify places of interest such as restaurants, ATMs, streets, museums, buildings, etc. However, all of these augmented reality applications require a lot of processing that may not be computed in a time frame required for a specified quality of experience (QoE) if executed by the mobile device.

Wearable computing is another technology under development that facilitates the human-computer interaction. Sensors collect body information that can be useful in biomedical applications. Medical doctors and physicians can access these data to keep track of the patients' health; in this way, they can make recommendations and plan effective rehabilitation strategies. Potential applications of wearable computing include the monitoring of: 1) blood pressure, 2) chronic pulmonary disease, 3) hemiplegic patients, and 4) Parkinson's disease [3] [4]. The monitoring of patients can be performed over long periods of time (e.g. days, weeks, or months). Remote execution is required since the collected information is analyzed by specialized medical software and because the handheld device may not be equipped with enough storage capacity to keep all the monitoring records. Wearable computing sensors regularly upload the data to databases on remote locations through a network (e.g. Internet). After collecting and processing the statistics from the data sets, the system detects and predicts the patient's clinical situation.

Image processing applications perform several filtering operations to sets of images that are

captured by a mobile device. In some cases, such operations may require a lot of processing depending on the applied filter. However, some of these applications may need to be executed in a shorter time frame than possible on the mobile device. To overcome this issue, image processing can benefit from computation offloading by creating a replication pattern on the cloud and then processing the images in parallel; with this approach, mobile devices can offer a better QoE to users [5].

Language translation applications are useful for people that need to communicate in another language while traveling to other countries. Such applications require the user to record the voice and get it translated. Users with visual disabilities need these applications to read them the translations on a specified time frame; however a mobile device may not meet such time requirements due to its limited storage capacity, processing capacity, or battery life [6].

In military battlefield operations in remote locations, there is limited or no infrastructure at all. In such situations, soldiers have missions where they collect and process data from the environment. These data includes information about possible enemy threats, detection of biological hazards, or voice captured from people speaking in different languages. However, soldiers are equipped with resource constrained mobile devices, that are unable to perform such computations.

Mobile devices are resource constrained in terms of their: 1) battery life, 2) processor performance, 3) storage capacity, and 4) network bandwidth. There are situations in which a computation job exceeds the capability of a computing device. For example, the mobile user may need to execute an application in a shorter time frame than possible on the mobile device. Or running a computation-intensive application unacceptably drains the battery of the mobile device. To overcome these limitations, there is a solution called ***Computation Offloading***, also known as *Cyber-Foraging* and *Surrogate Computing* [7] [8] [9]. Computation offloading migrates computation jobs from a mobile device to more powerful remote computing resources. These jobs are executed remotely and upon completion of the job, the results (i.e., output data) are sent back to the mobile device. Fig. 1.1 depicts the process of computation offloading: step 1)

the cloud client initiates offloading of a job to a cloud resource by sending the input data along with the program code, step 2) the cloud resource executes the job, and step 3) the cloud sends the output data through a communication network to the mobile device.

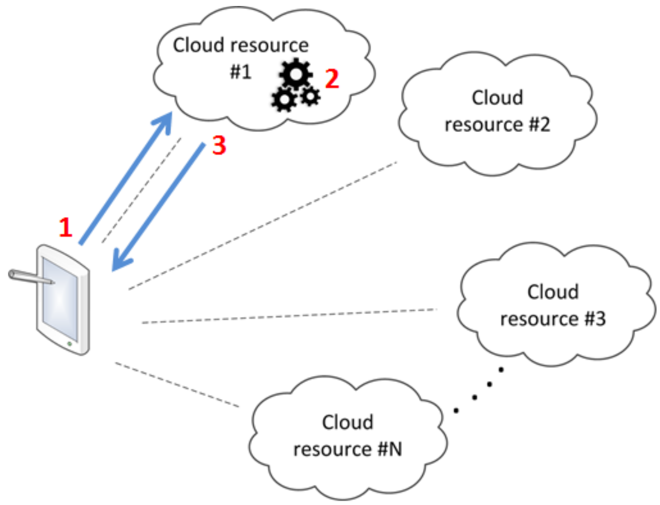


Figure 1.1: Computation Offloading Process

1.2 Problem Definition

Computation offloading will not reduce job completion time under all conditions. For this reason, we must determine under which conditions computation offloading is beneficial to reduce job completion time for mobile applications. Further, it is important to identify those specific applications that benefit from computation offloading. Mobile users need to make a decision whether to offload a computational job and a decision of where to offload it. However, it is difficult to have an instantaneous status of the computation offloading system since network conditions change frequently (e.g. queuing delays) and computation delays vary due to changing workloads. The uncertainty of network conditions is exacerbated for wireless networks where nodes are constantly moving and have frequent disconnections caused by topology changes. This dissertation work was sponsored by the United States Army Research Laboratory (ARL) [10] through Stanford University. As such, we consider attributes of computing on the battlefield. Soldiers in battlefield operations have unique requirements that make computation offloading

even more challenging. The battlefield requires communication over unreliable mobile ad-hoc networks that lead to more disconnections and route changes compared to non-mobile networks. Computation Offloading decisions are made with estimates of the following decision criteria: 1) job parameters, 2) computing resources parameters, and 3) communication network parameters. The job parameters include: a) the *job size* which is the application's size (e.g. bytes) that will get executed either on the mobile device or a cloud resource, b) the *input data* that include the information that will be processed by the application, and c) the *output data* which are the results produced from the computation. The computing resource parameters are the metrics that determine the status of the computing entities: a) CPU speed, b) memory capacity, c) storage capacity, and d) current workload. The communication network parameters are the ones that show the status of the networking conditions, they include: a) available bandwidth, b) congestion, c) packet loss rate, d) link quality, and e) number of hops to reach a resource. However, since these parameters are constantly changing over time, mobile devices cannot have an instantaneous and accurate measure of these metrics. Instead, estimated information is used to make offloading decisions. That is why, we have found the opportunity to study the impact that errors in the estimates of these parameters have on the quality of the computation offloading decisions. We want to understand how accurate an estimation has to be in order to make good decisions at a low risk. The questions that we want to answer in this research project are: 1) *When does offloading reduce completion time?*, 2) *Which applications benefit from offloading?*, and 3) *How does network delay estimation error affect computation offloading decisions when there is more than one offload target?*.

1.3 Dissertation Structure

The outline of this dissertation is as follows: Chapter 2 introduces *Cloud Computing*, including types of clouds, and cloud service models. An in-depth discussion of cloud services from Amazon Web Services, as well as a discussion of the communication patterns associated with cloud computing is included. Chapter 3 describes *Computation Offloading* in detail, including its use

cases, enabling architectures, computation offloading frameworks, and computation offloading criteria. Chapter 4 presents our *computation offloading decision analysis* we conducted, including our system model, our numerical experiments, and our real computation workload trace data analysis. Chapter 5 presents analysis of the effect of estimation error on our decision making using the Monte Carlo technique with our system model as well as simulation experiments with a NS-3 based simulator we created. Finally, Chapter 6 discusses our conclusions and future work.

Chapter 2: Cloud Computing

2.1 Definition

Cloud Computing is a more convenient way of providing resources over the Internet at a lower cost. In this new model, there is no need to buy anything (e.g. software or licenses), because it is a utility service in which you only pay for what you use. Cloud Computing accommodates elastic resource demand by: 1) allocating more capacity (e.g. computing, storage, memory) when the cloud user requires more resources to accomplish a task with certain constraints (e.g. time, throughput, quality of experience), and 2) deallocating resources whenever the cloud user reduces the workload or is no longer using them. A user may start with few resources and add more if needed. *Computing* refers to the processing, storage, and provisioning of resources located in remote data centers. *Cloud* is the hardware (e.g. networking devices and data centers) where this provisioning takes place. The cloud includes all of the communication network switching equipment (e.g., layer 2 switches, routers) that interconnects cloud clients with a data center as well as the computing equipment in a data center (e.g., compute servers, storage arrays). From the user's perspective, the cloud is like a "black box" where everything is stored, processed and loaded. Therefore, *Cloud Computing* means to provide computing resources from remote servers (e.g. data centers) over a network (e.g. Internet).

Looking at cloud computing from a slightly different perspective, cloud computing consists of several elements that include [11]: 1) *computers* that supply and manage the databases, file systems, virtualization software, applications, and middleware within the cloud, 2) *storage devices* that contain the input data to be processed and the produced output data, 3) *applications in the cloud* that run on Virtual Machines (VMs) to provide specific services (e.g. access to databases, load balancing, map reduce, transcoding, etc.), 4) *client devices* (e.g. smartphones, laptops, computers, tablets) that can have access to the cloud from many places and in charge of: a) establishing the connection to the network, b) communicating data to and from the cloud, c) taking user input, and d) displaying user output, and 5) *Cloud Manager* which is responsible

for configuring cloud resources and monitoring performance. Figure 2.1 illustrates the cloud computing elements.

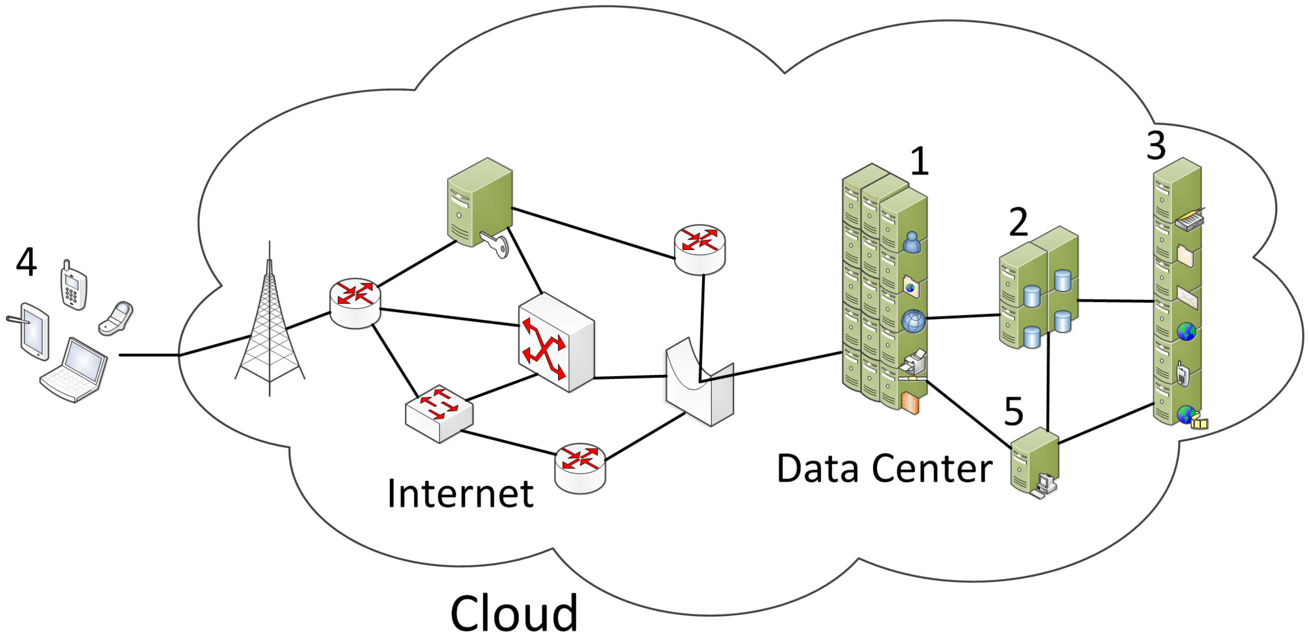


Figure 2.1: Cloud Computing Elements (1. computers, 2. storage devices, 3. applications, 4. client devices, and 5. cloud manager)

There are several compute-intensive applications that require multiple resources to process large amounts of data that can benefit from cloud computing services. These services offer different products that have benefits such as: 1) add or remove computing and storage capacity as needed, 2) reduction of operational costs, 3) pay for computing resources only when used, and 4) try new cloud services (e.g. media transcoding, databases, or dedicated network connections) [12] [13].

2.2 Types of Clouds

Clouds (i.e., data centers) are grouped in three main categories [12–16]: 1) *Public Clouds*, 2) *Private Clouds*, and 3) *Hybrid Clouds*. This classification is based on the location of the cloud computing resources and the audience to which the service is targeted. Users can be altruist organizations, medical groups, governments, universities, or the general public. Customers have

different needs and requirements; some of them have high security concerns, while others just need open-source software. Figure 2.2 shows the three types of clouds and the locations where they may be found. A brief description of each of these cloud types is presented in the next subsections.

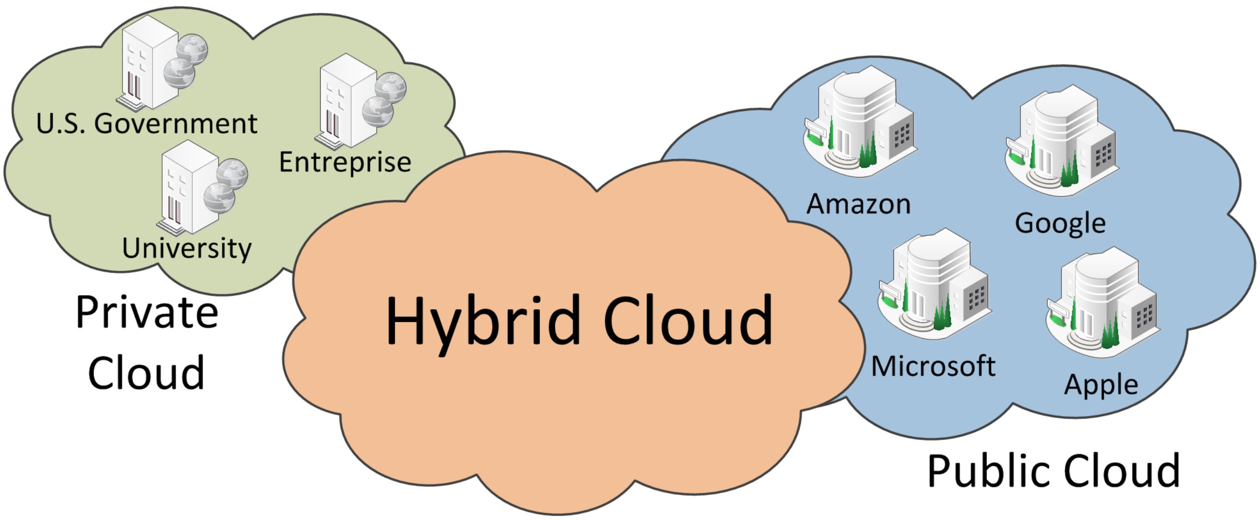


Figure 2.2: Types of Clouds

2.2.1 Public Clouds

Public Clouds are available resources (e.g. applications, computing processors, storage drives) offered to the general public over the Internet, as depicted in Figure 2.3. Users pay for what they use without the need to buy software, licenses or equipment. Customers don't need to pay for maintenance or training of personnel to keep their applications running. This type of cloud is flexible, since users can scale up when they require more resources or scale down when they reduce their workloads. There are several companies like Amazon, Google, Microsoft and Apple, that offer this type of cloud. By the principle of *economies of scale*, these companies offer their services at a lower cost due to the increasing number of users using the cloud. This type of cloud is suitable for users that store non-sensitive content and require operational efficiency.

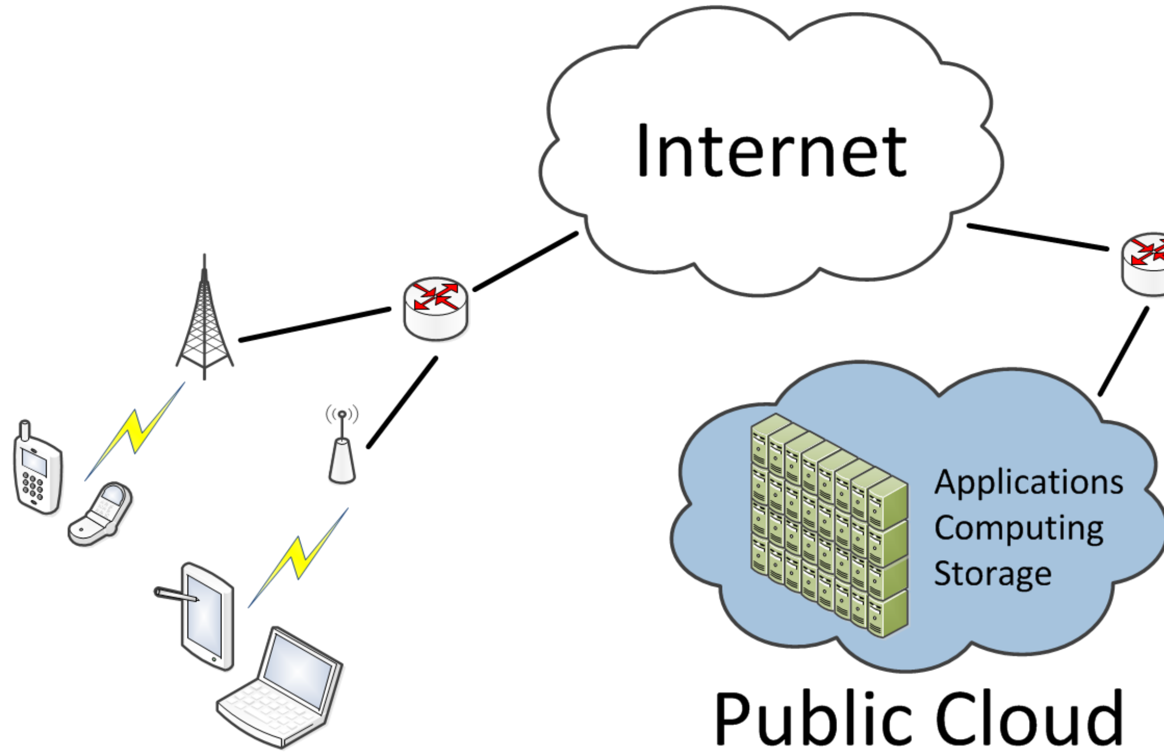


Figure 2.3: Public Cloud

2.2.2 Private Clouds

Private Clouds are restricted to a group of users, as illustrated in Figure 2.4. Hardware and software (e.g. servers, networks, applications) are provided internally to users within the organization. This type of cloud enforces security, since the data are kept within the company's facilities and administrators have full control over the resources. This type of cloud is suitable for organizations that require to perform processing and storage of sensitive data. However, private clouds are not as flexible as the public ones. Companies with a private cloud have some expenses like: maintenance, personnel to keep the servers running, hardware, software, and licenses. Also, these companies may be under provisioning resources to their users, that is, not supplying the minimum required resources to keep the organization running efficiently. On the other hand, companies may be over provisioning resources at some point, that is, wasting money by delivering more resources than necessary.

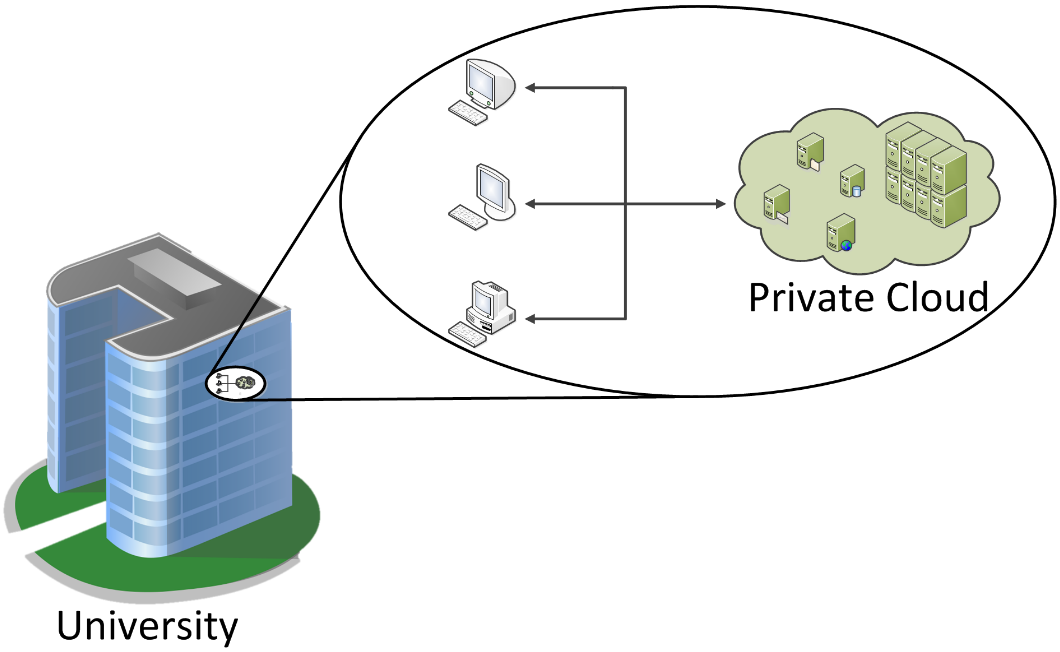


Figure 2.4: Private Cloud

2.2.3 Hybrid Clouds

Hybrid Clouds are a combination of two or more clouds (e.g. public or private) [15] as shown in Figure 2.5. This type of cloud is suitable for users that share content with other organizations [12]. In order to have an efficient hybrid cloud, designers need to determine the public cloud components and the private cloud components [16]. Enterprises may use their private cloud to control core activities and the public cloud to deliver a service to end customers. Companies that own a private cloud may require more capacity to accomplish complex tasks. Usually the public cloud is used whenever the private cloud requires more computation power to achieve better performance and prevent under provisioning of resources [13]. By having a hybrid cloud, companies still pay for maintenance, hardware, software, and licenses; however these costs are reduced whenever they lease public cloud services. Organizations may have fluctuating workloads that only use the public cloud services during peak time periods. To prevent the over provisioning of resources, the public cloud is released whenever the workloads get reduced, and only the private cloud is used.

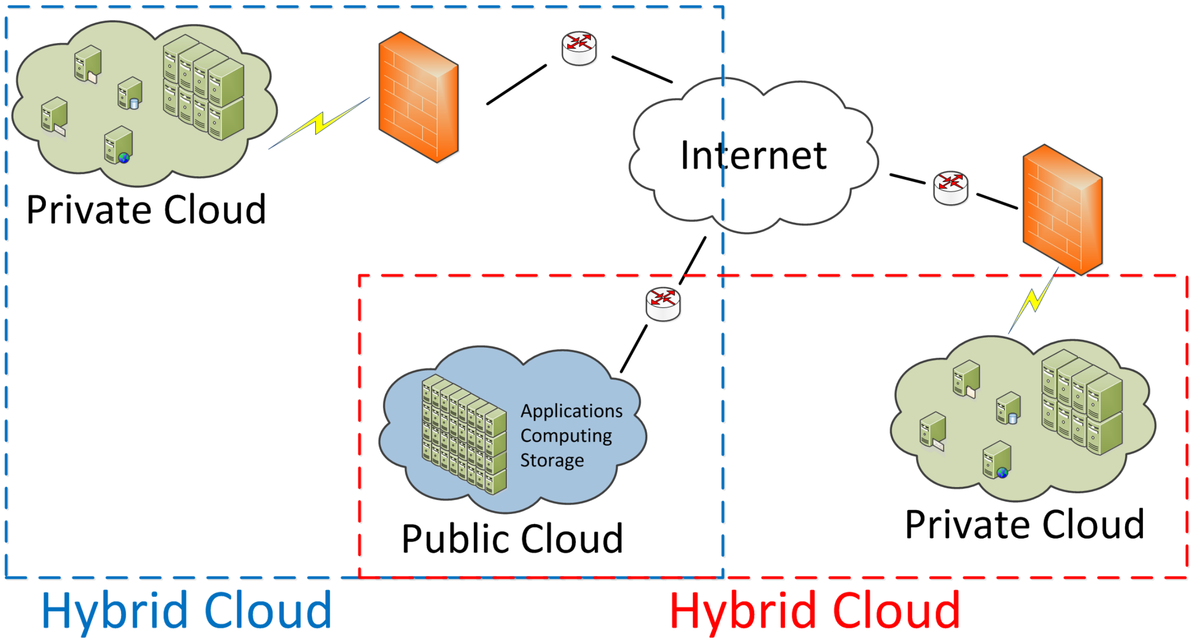


Figure 2.5: Hybrid Cloud

2.3 Cloud Service Models

Cloud Computing providers offer different types of products depending on the users' needs. These products are classified in three main cloud service models: 1) Infrastructure as a Service (IaaS), 2) Platform as a Service (PaaS), and 3) Software as a Service (SaaS). These cloud service models are layered according to the type of elements they provision to cloud users [16], as show in Figure 2.6.

2.3.1 Infrastructure as a Service (IaaS)

IaaS includes the *hardware layer* and the *infrastructure layer* as shown in Figure 2.6. The *hardware layer* refers to all the physical resources found in a data center: CPU, memory, storage, network infrastructure (e.g. switches, routers), cooling systems, and power electricity. The *infrastructure layer* is also known as the *virtualization layer* [16], where physical components are partitioned to service different users and increase utilization of the resources.

Nimbus [17] is an open-source platform that implements IaaS, and is used mainly by the scientific community. It allows users to have customized resources according to their needs.

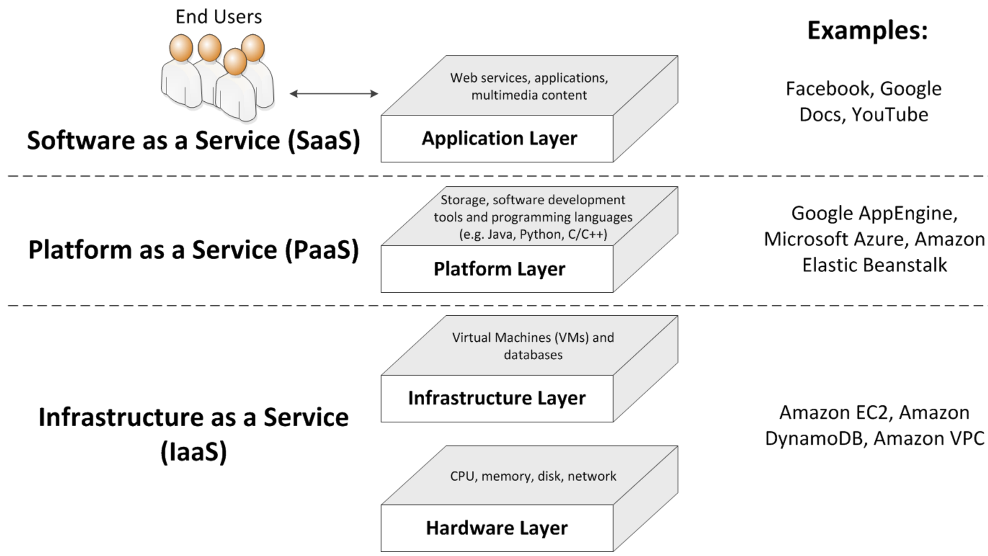


Figure 2.6: Cloud Service Models

Nimbus has three main tools: 1) *cloudinit.d*, 2) *Phantom*, and 3) *Context Broker*. *cloudinit.d* is the component in charge of launching, controlling, and monitoring the applications in the cloud. It deploys sets of VMs to multiple clouds from different vendors (e.g. Eucalyptus, OpenStack, Amazon, Nimbus). *Phantom* is the tool in charge of auto-scaling the resources. It monitors the status of the resources and automatically provisions and configures more if needed. The *Context Broker* is the cloud element that coordinates clusters of VMs. It allows users to customize VMs (e.g. configurations, access policies) and provides security to the cloud.

Eucalyptus [18] [19] is another platform that implements IaaS. It is composed of three levels: 1) Cloud level, 2) Cluster level, and 3) Node level. Within the *cloud level*, there are two elements: 1) *Cloud Controller (CLC)* and 2) *Scalable Object Storage (SOS)*. The CLC handles all the requests from the user and handles the authentication, accounting, and reporting. The SOS is the element that provides basic storage to the users. At the *cluster level*, there are two components: 1) *Cluster Controller (CC)* and 2) *Storage Controller (SC)*. The CC is the front end of Eucalyptus, it is in charge of managing the execution of the cloud instances (i.e. VMs) and carrying out the Service Level Agreements (SLAs). The SC is the element that has direct communication with the CC and NC to manage the block storage volumes that are used by the VMs. Finally, at the *Node*

Level, there is one component, the *Node Controller (NC)*. The NC is in charge of hosting and managing VMs. It also creates and caches images from the SOS.

2.3.2 Platform as a Service (PaaS)

PaaS encloses the *platform layer* as shown in Figure 2.6. This layer sits on top of the *infrastructure layer* and the *hardware layer*. This service offers the flexibility of developing and running applications without any hardware or software maintenance cost. It includes all the software development tools and programming languages (e.g. Java, Python, C/C++, PHP) used by software developers. It also provides revision control systems that allow programmers to collaborate on projects; developers can create, test, deploy, monitor, grow, and maintain applications quickly and easily [12] [20]. Google AppEngine, Microsoft Azure, and Amazon Elastic Beanstalk are examples of this kind of service.

2.3.3 Software as a Service (SaaS)

SaaS constitutes the *application layer* that sits on top of the *platform layer* and the rest of the layers as shown in Figure 2.6. This layer includes all the applications and software that end users access directly. Applications like Facebook, Google Docs, YouTube and Netflix are examples of this type of cloud service. They provide web services and content delivery networks to users around the world. These applications can be accessed over the Internet by many users through their computing devices (e.g. smartphone, tablet, laptop, PDA) by employing thin clients, terminals, emulators, web browsers, or mobile applications.

2.4 Amazon Web Services

Amazon Web Services (AWS) are a cloud computing infrastructure created and supported by Amazon [20], the most mature cloud service provider at present time. Amazon offers step by step tutorials and a friendly Graphical User Interface (GUI) that makes life easier to its customers. Amazon is an excellent opportunity to get a better insight and understanding on how to use the cloud and to show some examples of commercially available cloud products. Amazon's

products offer different types of services depending on their customers' needs. These products provide storage, computing, databases and tools to manage cloud resources. Amazon has many data centers across the world for redundancy and availability (e.g. Points of Presence (PoP)). Users are able to create, deploy and maintain many applications (e.g. games, social networks, bank apps). These applications can be built for different operating systems by using tools and programming languages that are already known to developers. Customers are allowed to process and store big data by adding and removing capacity dynamically. Six benefits are offered by AWS to their customers: 1) just pay for the resources used, 2) deliver services at low operational cost due to economies of scale, 3) prevent under provisioning or over provisioning by adding and removing resources based on the current load, 4) get resources quickly to test and deploy new applications, 5) reduce maintenance of customers' data centers, and 6) deploy applications with low latency to end customers. AWS offers products in two of the three cloud service models: 1) Infrastructure as a Service (IaaS) and 2) Platform as a Service (PaaS). For each of these cloud service models, AWS classify their products in different categories as show in Figure 2.7.

2.4.1 Infrastructure as a Service (IaaS)

Amazon offers 23 different products that are part of the IaaS service model. These products are classified under 5 categories: 1) compute, 2) storage & content delivery, 3) databases, 4) networking, and 5) administration & security. A brief description of each of these products is presented in the next subsections.

COMPUTE

Amazon EC2

Amazon Elastic Compute Cloud (EC2) is a service used to provide computing resources to users. It offers Virtual Machines (VMs) called *instances* that allow users to scale up or down depending on their current needs. These instances can be: a) small, b) medium, c) large, and d) extra large. Their instance sizes differ in their capacity of memory, compute units, storage, architecture, and I/O performance. Instances can contain any desired software and run different operating systems

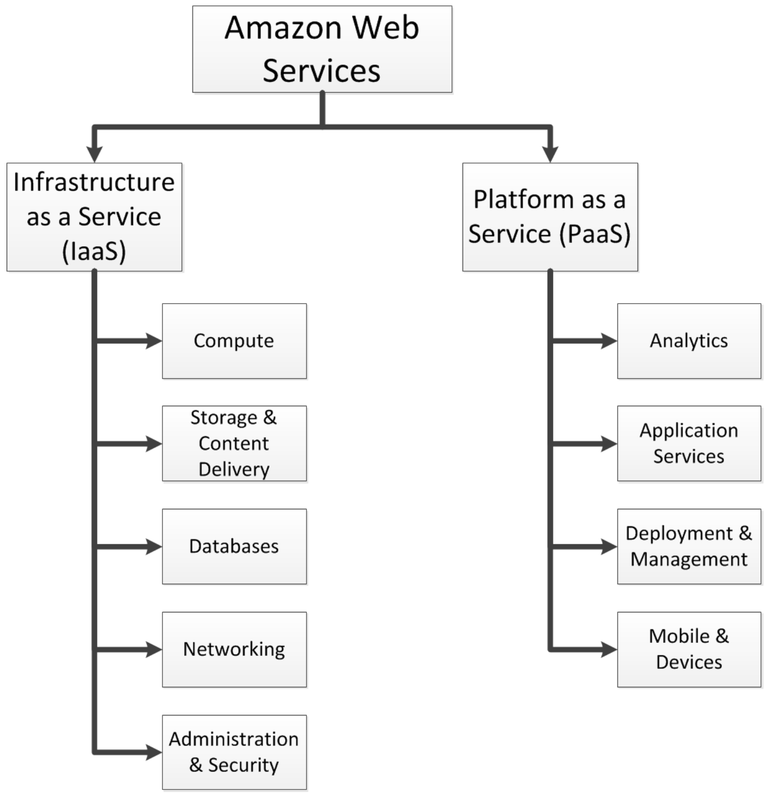


Figure 2.7: AWS Cloud Service Models

(e.g. Microsoft Windows, Linux). Users can create, launch, and terminate instances as needed. A user may create instances with different processor architectures (e.g. 32-bit, 64-bit) and have from 1 to 32 virtual CPUs. The memory capacity of an instance ranges from less than 1GB up to 244 GB, and the storage capacity can be up to 48 TB. An instance can have up to a 10-Gigabit network connection. To create an instance, a user accesses AWS through a web browser, configures the properties and launches the desired instance, as illustrated in Figure 2.8. There is a management section that lists all the instances, their details, and status, as show in Figure 2.9. After initializing and checking that the instance is ready to operate, a user may connect to it through ssh, as depicted in Figure 2.10. Some of the companies that use this service are: Netflix, AutoDesk, Animoto, and Bankinter.

Amazon EC2 Container Service

Amazon EC2 Container Service is a service that allows users to run distributed applications on

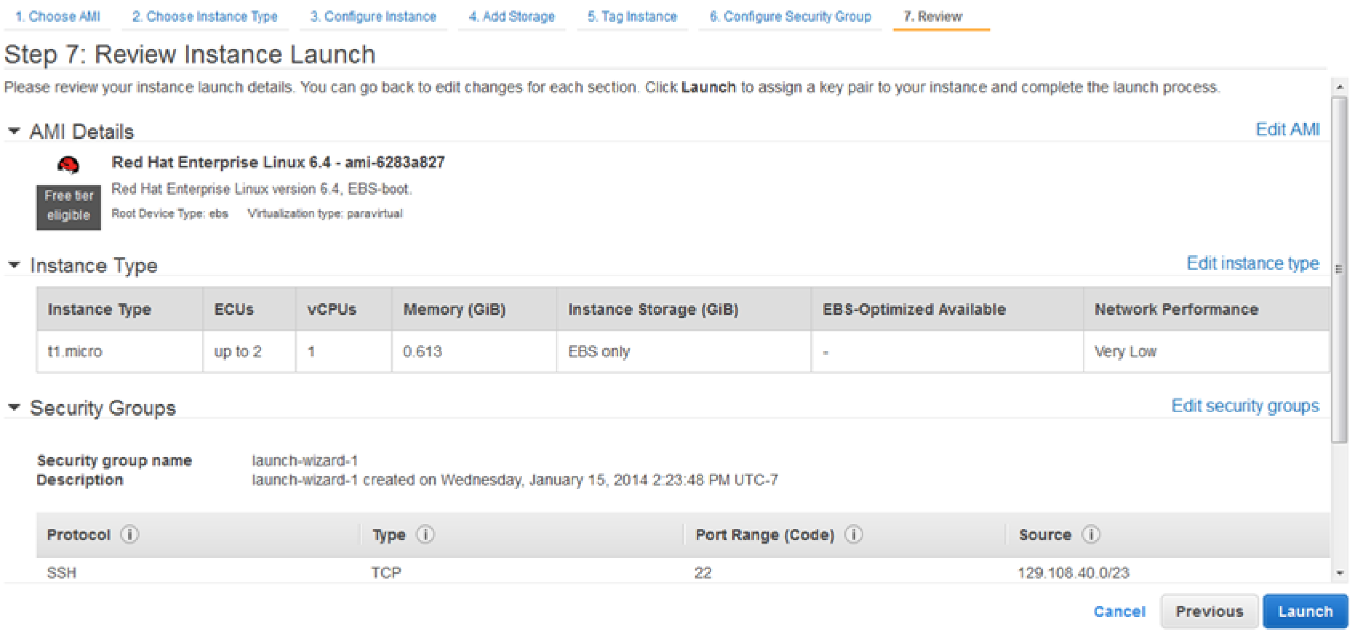


Figure 2.8: Creation of an EC2 instance

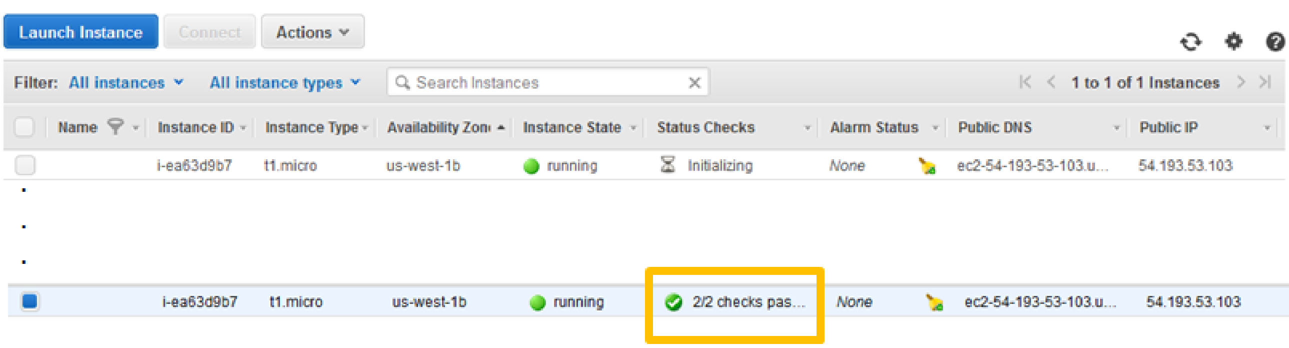


Figure 2.9: EC2 manager

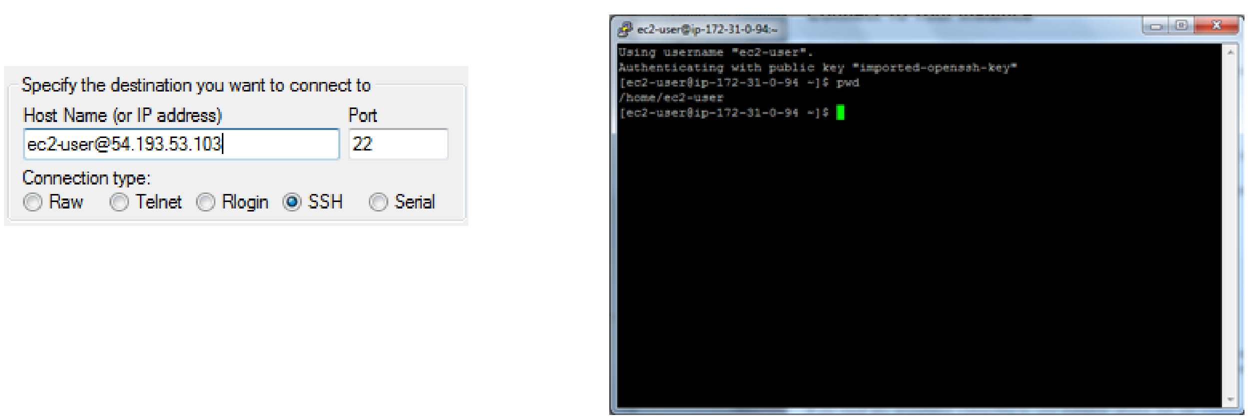


Figure 2.10: EC2 access through SSH

a cluster of Amazon EC2 instances. It allows users to place their instances across the cluster according to their resource needs, isolation policies, and requirements. This service is included when the user leases EC2 instances or the Elastic Block Store (EBS).

AWS Lambda

Amazon Lambda is a service that allows users to run their code in response to events (e.g. an image upload, website click, in-app activity). It manages the compute resources in response to new information generated, and the user only pays for the number of requests and the time required to run the code.

Auto Scaling

Auto Scaling allows users to scale up or down the EC2 instance capacity according to the user's requirements. It automatically increases the number of EC2 instances during peak demands and decreases them during idle periods. This type of service is ideal for applications with stable demand patterns (e.g. hourly, daily, weekly), because this service knows before hand the periods of time in which the capacity needs to be adjusted (e.g. every day at 12PM the capacity needs to be increased and two hours later this capacity needs to be decreased).

Elastic Load Balancing

Elastic Load Balancing automatically distributes the application traffic across multiple EC2 instances. It also allows users to achieve great levels of fault tolerance.

STORAGE & CONTENT DELIVERY

Amazon S3

Amazon Simple Storage Service (S3) is a service that allows users to store and retrieve any amount of data from everywhere. Data are stored in units called *buckets*. The bucket's size ranges from 1 byte to 5 terabytes. Users can select the region where to save their information (e.g. US East, US West, Asia, Europe) since Amazon has many infrastructure regions and edge points of presence in several countries across the world. This service can be used alone or with other AWS services (e.g. EC2, EBS, and Glacier). S3 is the service that holds all the information

needed by EC2 and EMR. Allocation of storage for an application and data are needed whenever a workflow is to be executed [21]. Input data are transferred from S3 buckets to EC2 instances. After processing the data, output files are sent back to S3. Some of the companies that use this service are: Nasdaq, Netflix, and Pinterest.

Amazon EBS

Amazon Elastic Block Store (EBS) is a service that provides block level storage volumes that are used with EC2 instances. It automatically replicates within its availability zone to: 1) protect users from failures, 2) offer high availability, and 3) ensure durability. It is characterized to be a low-latency performance service. Some of the companies that use this service are: Super Cell, Drop Cam, Unilever, GameVil, Hess, CopperEgg, and Adobe.

Amazon Glacier

Amazon Glacier is a service used as long-term data storage. It stores data redundantly across multiple devices to protect users' data. It allows users to move data from S3 buckets or any other third party storage device into Amazon Glacier. The cost of the service is around $0.01 per gigabyte per month. Some of the companies that use this service are: Illumina, nearmap.com, Scribd, Sound Cloud, and Backupify.

Amazon CloudFront

Amazon CloudFront is a content delivery web service used by developers and companies to deliver web content and data to their customers. This content delivery network has multiple copies of the data on different servers around the world, such that a client's request may be handled by the closest server. This product is ideal for applications such as: video streaming, audio streaming, and Internet television. It is a low latency service with high data transfer speeds. Some of the companies that use this service are: Sega, PBS, encoding.com, Classle, and JoomlArt.

DATABASES

Amazon RDS

Amazon Relational Database Service (RDS) is a service that allows users to launch a database

instance in minutes and have automatic backups of their information. RDS allows its users to monitor their metrics and gives access to well-known databases like MySQL, Oracle, or SQL Server; this makes RDS compatible with existing code, applications, and tools. This service is also re-sizable in capacity; users may request more space as needed or reduce it. Some of the companies that use this service are: Sega, Gumi, Flipboard, Unilever, and AirBNB.

Amazon DynamoDB

Amazon DynamoDB is a NoSQL database service; it is fully managed by Amazon. If a user requests more throughput, DynamoDB distributes the data and traffic over a number of servers using solid-state drives; this allows a more predictable performance. Some of the companies that use this service are: AdRolL, has offers, Scribble Live, and Drop Cam.

Amazon ElastiCache

Amazon ElastiCache is a service that improves the performance of a web application by allowing its users to retrieve information from high-speed caches instead of using slower disk-based databases. This service makes it easy to deploy, operate, and scale an in-memory cache in the cloud. In case of a failure, Amazon ElastiCache automatically replaces the failed nodes. Some of the companies that use this service are: Peel, Health Guru, Tadaa, PlaceIQ, PBS, and TicketLeap.com.

NETWORKING

Amazon VPC

Amazon Virtual Private Cloud (VPC) is a service that gives full control over a virtual network environment to its users. Administrators can: 1) select IP address ranges, 2) create subnets, and 3) configure routing tables and gateways. The service allows its users to create VPN connections between their databases and Amazon VPC.

AWS Direct Connect

AWS Direct Connect is a networking service that establishes dedicated network connections between AWS and its users' facilities (e.g. data center, office). It provides three benefits: 1)

reduces users' networking costs, 2) increases bandwidth throughput, and 3) provides a more consistent network experience.

Amazon Route 53

Amazon Route 53 is a cloud Domain Name System (DNS) web service. It translates web page names into numeric IP addresses. It handles users requests to AWS services (e.g. EC2, Elastic Load Balancing, S3). It also manages the users' traffic using different routing types (e.g. Latency Based Routing, Geo DNS, and Weighted Round Robin).

ADMINISTRATION & SECURITY

AWS Directory Service

AWS Directory Service is a service that allows users to connect to AWS resources with their existing directory or set up a new one in the cloud. When the connection is established, all users can access AWS resources and applications with their corporate credentials.

AWS IAM

AWS Identity & Access Management (IAM) is an administration service that allows its users to create and manage users and groups. Administrators can allow or deny access to AWS resources. This service is useful for organizations that have multiple users and systems that use Amazon services (e.g. EC2, RDS, Manage Console).

AWS CloudTrail

AWS CloudTrail is a web service that records all API calls of the user's account. It provides a log with the following information: 1) identity of the API caller, 2) time of the API call, 3) source IP of the API caller, 4) request parameters, and 5) response elements returned by the service. CloudTrail is useful for administrators to perform security analysis, resource tracking, and auditing. Some of the companies that use this service are: Intuit, Siemens, HTC, and Finra.

AWS Config

AWS Config is a service that provides: 1) resource inventory, 2) configuration history, and 3)

configuration changes. This services helps to discover available AWS resources and export them with all configuration details. This service is suitable for security analysis, resource tracking, auditing, and troubleshooting.

AWS CloudHSM

AWS Cloud Hardware Security Module (HSM) is a service that helps to control encryption keys and cryptography performed by Amazon. It also guaranties the migration of users' applications to the cloud by using Amazon VPC. This service works close to its users' EC2 instances to reduce network latency and improve application performance.

AWS KMS

AWS Key Management Service (KMS) is the service in charge of creating and controlling the encryption keys used to protect its users' data. It collaborates with other AWS products (e.g. EBS, S3, RDS, Redshift, Elastic Transcoder, CloudTrail) to provide protection.

Amazon CloudWatch

Amazon CloudWatch is a service in charge of monitoring the AWS cloud resources and applications that are running. it collects and tracks metrics, log files and sets alarms. It collaborates with other AWS products (e.g. EC2, DynamoDB, RDS) to allow users to check the resource utilization, application performance, and operational health.

AWS Service Catalog

AWS Service Catalog is a service that allows administrators to create and manage the list of resources that the users can access. With this service, administrators can give access to specific applications or cloud resources.

2.4.2 Platform as a Service (PaaS)

Amazon offers 19 different products that are part of the PaaS service model. These products are classified under 4 categories: 1) analytics, 2) application services, 3) deployment & management, and 4) mobile & devices. A brief description of each of these products is presented in the next

subsections.

ANALYTICS

Amazon EMR

Amazon Elastic MapReduce (EMR) is a web service that allows its users to process big data (e.g. large amounts of data). EMR collaborates in the processing of data with S3. S3 provides storage units (e.g. storage nodes) and EMR allows users to process large amounts of data by distributing the workload among EC2 instances (e.g. computing nodes). This collaboration is performed by using Hadoop to distribute and process the data. Users upload data through a web browser and it becomes available to the EMR system. Most users submit jobs to a system through a web browser. These jobs are placed in a queue and scheduled. They also may be partitioned for migration to an external cloud. Some portions are processed locally and some others remotely. Once a job is completed, the result or partial results are aggregated and retrieved by the user. The overall processing time includes: 1) uploading the input data, 2) partitioning the job into tasks, 3) executing the tasks, 4) aggregating the results from tasks, 5) downloading the results to the user, and 6) any other job movements between queues [22]. EMR is useful for applications such as: log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulations, and bioinformatics. Some of the companies that use this service are: Yelp, Expedia, Nokia, Euclid, and Snow Plow.

Amazon Kinesis

Amazon Kinesis is a cloud service used for real-time processing of large distributed data streams. This service constantly captures and stores a lot of data (e.g. terabytes) every hour from many resources (e.g. website clicks, bank transactions, social networks, logs, GPS tracking events). This service is useful for companies that need dynamic pricing (e.g. stock market) or advertising.

Amazon RedShift

Amazon RedShift is a cloud service that is fast and powerful to analyze all its users' data. It uses existing tools from its users to get the statistics and information needed. This service costs

$0.25 per hour or $1,000 per terabyte per year. Some of the companies that use this service are: AirBNB, Nokia, Schumacher Group, Euclid, and Pinterest.

AWS Data Pipeline

AWS Data Pipeline is a web service that allows users to process and move data between different AWS compute and storage products (e.g. S3, RDS, Dynamo, EMR). It can help users to create complex processing workloads (e.g. fault tolerant, repeatable, and highly available). Some of the companies that use this service are: Nokia, Coursera, and Swipely.

APPLICATION SERVICES

Amazon SQS

Amazon Simple Queue Service (SQS) is a managed message queuing service. It allows users to transmit any amount of data without losing messages or requiring other services. SQS is fast, reliable, and scalable.

Amazon SWF

Amazon Simple Workflow Service (SWF) is a cloud service that allows programmers to develop, run, and scale applications that have sequential or parallel steps. This service keeps track of the processing state, and recovers or retries a failed task.

Amazon AppStream

Amazon AppStream is a cloud service that allows its users to deploy their application to any mobile device without modifying the code. This is possible, because AWS infrastructure runs and renders the applications, and streams the output data to any mobile device (e.g. PC, tables, smartphones); this service follows the *thin client model* where the mobile device just displays what is being executed on the cloud.

Amazon Elastic Transcoder

Amazon Elastic Transcoder is a cloud service that allows its users to convert digital media into many audio and video formats (e.g. codecs). It also provides the S3 containers required by the

user's playback devices (e.g. smartphone, tablet, web browser, TV).

Amazon SES

Amazon Simple Email Service (SES) is an email sending service that keeps track of the outbound email only. This service is useful for companies dedicated to marketing or customer service departments that may want to quantify: 1) sending statistics, 2) notifications, 3) complaints, and/or 4) deliveries. Some of the companies that use this service are: FlipTop, D-Link, Siemens, Vodafone, Ticket Master, and Stitcher Smart Audio.

Amazon CloudSearch

Amazon CloudSearch is service that allows users to search data (e.g. web pages, documents, posts, product information). This product is used to index and search structured data or plain text. It supports full text search, search with fields, prefix searches, and some others. Some of the companies that use this service are: Bizo, Smug Mug, Zip List, Coursera, and reddit.

DEPLOYMENT & MANAGEMENT

AWS Elastic Beanstalk

AWS Elastic Beanstalk is a cloud service designed for software developers. It allows them to deploy, monitor, and grow their applications quickly and easily. This service supports well-known applications and software (e.g. Ruby, PHP, Python, Apache HTTP Server, .NET, Java). Some of the companies that use this service are: Adobe, Peel, and swiitch.

AWS OpsWorks

AWS OpsWorks is a cloud service that provides the automation tools to model and control the life cycle of an application. It also manages the infrastructure on which the application runs. It gives its users the standard package configurations, and allows them to customize, install, and create other elements. Some of the companies that use this service are: Wooga, Beach Mint, Art Sy, Fresh Desk, and Toronto Star.

AWS CloudFormation

AWS CloudFormation is a cloud service that allows its users to deploy the resources and applications on AWS. Users may use existing templates or create their own to start deploying applications on the cloud. Some of the companies that use this service are: DowJones, Adobe, Sega, Expedia, and Singapore post.

AWS CodeDeploy

AWS CodeDeploy is a cloud service that helps developers to automate the code deployments on EC2 instances. It handles the complexity of updating applications.

AWS CodePipeline

AWS CodePipeline is another cloud service that helps software developers to easily deploy their code on the cloud. This service is useful for users that require help with the building, testing and releasing of code to production.

AWS CodeCommit

AWS CodeCommit is a cloud service that hosts Git repositories. Developers can have a revision control system to collaborate with other developers. This service allows them to browse, edit, and share projects.

MOBILE & DEVICES

Amazon Cognito

Amazon Cognito is a cloud service that saves user data in the cloud and synchronizes that data across multiple mobile devices (e.g. tablet, smartphone, laptop). This allows users to use their application from other devices even when the main device is offline since the data are stored in the cloud. This service can be used with: Amazon, Facebook, Twitter, Digits, or Google.

Amazon Mobile Analytics

Amazon Mobile Analytics is a cloud service that collects and analyzes the usage of data and provides charts. It allows users to set up automatic export of their data to Amazon and use

other analytic tools. This service by default processes and exports user's data every hour. Some of the companies that use this service are: Peak, PlayerFM, and Forza Football.

Amazon SNS

Amazon Simple Notification Service (SNS) is a push messaging service that delivers notifications to connected devices (e.g. smart phone, tablet, laptop). Users can communicate through this service by delivering emails or text messages. Some of the companies that use this service are: Yelp, Wunderlist, Hike Messenger, Path, Yo, and Easy Taxi.

2.5 Communication Patterns

Transactions in a cloud environment must have the following properties [23]: 1) *atomicity*, that refers to successfully completing all the operations in the cloud, 2) *consistency* of the database state, 3) *isolation* between transactions to prevent access to the same data items concurrently, 4) *durability* of the committed transactions even when a failure occurs, and 5) *membership* of the transactions to be executed and completed successfully. Communication transactions can be classified in two categories [24]: 1) *short-lived transactions* which are common Internet transfers (e.g. web requests, real-time applications, interactive applications), and 2) *long-lived transactions* that are used to transfer large data blocks (e.g. file transfer protocols (FTP), video and audio streaming).

Communication for cloud computing is framed by the following entities: 1) Cloud Clients (CCs), and 2) Cloud Service Providers (CSPs). There are three segments where communication occurs [25] [26] as depicted in Fig. 2.11: 1) Communication between client and CSP, 2) Communication within CSP facilities (i.e., within data centers), and 3) Communication between CSP facilities.

2.5.1 Client-CSP

Cloud Service Providers (CSPs) offer three types of services [12] [24]: 1) *Infrastructure as a Service (IaaS)* to provide basic storage and computational capabilities over the network, 2)

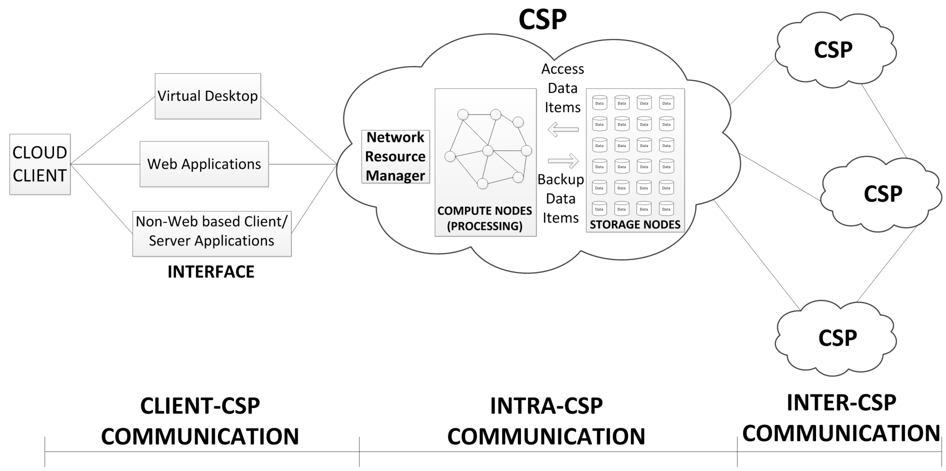


Figure 2.11: Cloud Computing Communication

Platform as a Service (PaaS) that grants resources to develop, deploy, test, host, and maintain applications, and 3) *Software as a Service (SaaS)* which offers complete software applications to users. The communication between the CC and CSP can be classified as either *non-interactive* or *interactive* as shown in Figure 2.12.

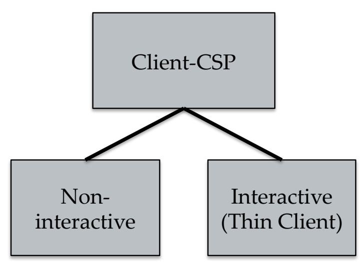


Figure 2.12: A taxonomy of client-CSP cloud computing communication. This class consists of two sub-classes: Non-interactive whereby computations are offloaded to a CSP with no user interaction while executing and Interactive (or thin-client) in which a user interacts with the computation executing in the CSP facility.

With the SaaS paradigm, CSPs host software on their servers for use by CCs. Most often this service is provided through a web interface running in a web browser. Google Docs, Amazon App Stream, and Microsoft Office 365 are examples of SaaS. Within the SaaS service paradigm the communication between the CC device and CSP device(s) is ***interactive*** and follow the ***thin-client model*** [27] [28]. Remote login via SSH or VNC are classic examples of the thin-client model. With the PaaS paradigm, CSPs host computing platforms on their servers for

use by CCs. Computing platforms include application runtime environments, database servers, and web servers. Google App Engine and Microsoft Azure are examples of PaaS. Finally, with the IaaS paradigm, CSPs host computing resources (physical or virtual) in their data centers for use by CCs. Computing resources include virtual machines and storage capacity. Amazon EC2/S3 and Microsoft Azure are examples of IaaS. Within the PaaS and IaaS service paradigms the communication between the CC device and CSP device(s) is very often ***non-interactive***. This non-interactive communication is defined by the movement of a computational workload or **P**rogram, **I**nput Data, and **O**utput Data between a CC device and CSP device(s). Figure 2.13 classifies the *non-interactive* communication according to where the program, input data, and output data are located; either at the CC device or the CSP device. In scenarios where the program and both input and output data are in and stay in the CSP device(s), PaaS and IaaS communication is somewhat similar to the thin-client model. The exception is that the level of user interaction will be merely to launch a computational job.

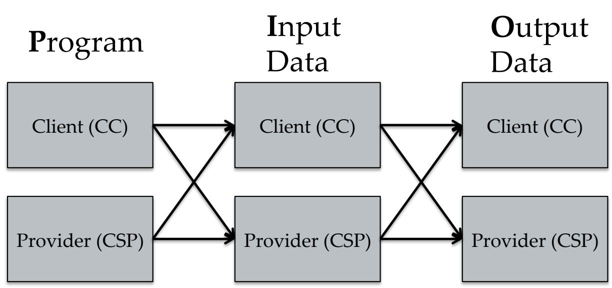


Figure 2.13: A taxonomy of **non-interactive** cloud computing communication between CC and CSP devices. These non-interactive communications are characterized by the location of the **P**rogram, **I**nput Data, and **O**utput Data; at the CC device or at the CSP device(s).

Non-interactive

In the non-interactive class of client-CSP communication, a client offloads a computational job to the cloud. This computational job could be: (*i*) a procedure call, (*ii*) an object, (*iii*) a process, or (*iv*) a virtual machine (VM) [29]. The nature of the communication is defined by the location of the executable set of instructions or **P**rogram, the **I**nput data, and the **O**utput data. The location of the program is not referred to where the program is executed but rather where

the executable file is located; it is assumed that the program is executed on a CSP computing resource. As shown in Figure 2.13, the location of each of the **P**, **I**, or **O** is either the client or the CSP; resulting in a total of eight possible combinations.

When the **P**, **I**, or **O** are already in the CSP facility, then the client sends a short message with a program or data identifier. When at the client, the **P**, **I**, or **O** must be transferred to the CSP facility. This could potentially involve the transfer of a large file. In some scenarios, the location of the input data or output data are not mutually exclusive. As an example, some of the input data might be sourced from the client, while some may be at the CSP facility. When the input data, **I**, is in the client a significant amount of input data are in the client. This example and its interpretation can be extended to the output data as well.

With a wide variety of programs and data there will not be any clear patterns. Specifically, it is not expected that file sizes will fit any probability distribution. The next subsections briefly discuss possible use cases for each of the combinations of **P**, **I**, and **O**.

P:Client;I:Client;O:Client

The client contains the program to be executed and a significant amount of input data. Once the program and input data are received by the CSP, the computational job begins execution. When the computational job completes a significant amount of data returns to the client. An example of this scenario is one in which the client has a custom application that must run in the cloud. However, the CSP would likely cache this program so that future invocations of that program would not require it to be re-transmitted into the CSP facilities. With this in mind, all of the scenarios in which the client contains the program are going to be much less common than those in which the program is already in the cloud.

P:Client;I:Client;O:CSP

The client contains the program to be executed and a significant amount of input data. Once the program and input data are received by the CSP, the computational job begins execution. However, when the computational job completes, the output data remains in the cloud. An ex-

ample might be a custom client program, that is not already in the CSP facility, that contributes data to a cloud database.

P:Client;I:CSP;O:Client

The client contains the program to be executed and the client identifies some input data located in the CSP facilities. Once the program is received by the CSP, the computational job begins execution. When the computational job completes a significant amount of data returns to the client. An example might be a custom client program, that is not already in the CSP facility, that performs a data retrieval operation on a cloud database.

P:Client;I:CSP;O:CSP

The client contains the program to be executed and the client identifies some input data located in the CSP facilities. Once the program is received by the CSP, the computational job begins execution. However, when the computational job completes, the output data remains in the cloud. An example would be a custom client program, that is not already in the CSP facility, that will perform some operation on a cloud database without retrieving any data.

P:CSP;I:Client;O:Client

The cloud contains the program and significant input data comes from the client to start the computational job. When the computational job completes a significant amount of data returns to the client. Search applications in which the search query generates significant data, such as voice/audio (e.g. Siri, Shazam) or image search (e.g., Google Goggles), fit this scenario.

P:CSP;I:Client;O:CSP

The cloud contains the program and significant input data comes from the client to start the computational job. However, when the computational job completes, the output data remains in the cloud. Crowdsourcing applications like GasBuddy to record gasoline prices fit this scenario.

P:CSP;I:CSP;O:Client

The cloud contains the program and the client identifies some input data (e.g., a street address) to start the computational job. When the computational job completes a significant amount of

data returns to the client. Mapping applications (e.g., Google Maps) or text search applications in which a very small amount of data (e.g., an address or a text search query) from the client is sent to the cloud and a large data set is processed to return a significant amount of data back to the client (e.g., map route, search results) fit this scenario.

P:CSP;I:CSP;O:CSP

The cloud contains the program, input data and output data. In this case, the client merely initiates the job and only short messages are exchanged between client and CSP. An example of this client-CSP non-interactive scenario could be from an over-the-top (OTP) IPTV provider. An OTP IPTV provider receives movies in a single master digital format from Hollywood and must encode various compressed versions of that video. The OTP IPTV provider may utilize Amazon's Elastic Transcoder service to transcode a master video into several other formats. The video then stays in Amazon's facilities because Amazon can also provide Content Distribution services for that OTP IPTV provider with its Cloudfront product.

Summary: support large file transfers

This class of client-CSP communication consists of file transfers that can be potentially large. To support this type of traffic, mechanisms that support large file transfers are necessary. Orchestra [30] is an application layer protocol that coordinates large file transfers for improved performance. Below the application layer, hybrid packet/circuit switched networks [31–37] could provide a circuit service for large file transfers. The circuit rate will not only provide a deterministic transfer time, but would serve as a traffic shaper. This traffic shaping protects other traffic from the congestion that could potentially be caused by large file transfers.

Interactive: thin client model

In the interactive (thin-client) model of client-CSP communication, the cloud client exchanges messages containing user interface data with the CSP (e.g., remote login, VNC, Amazon Workspaces, Amazon AppStream). The user interface data consists of input device data transmitted from the cloud client to the CSP and display data transmitted from the CSP to the cloud client; see

Figure 2.14.

The attributes of the communication arising from the thin-client model that are of interest to the communication network research community are: latency requirements, network throughput requirements, and packet sizes, each of which is discussed below.

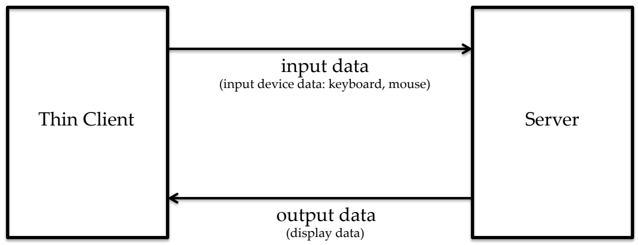


Figure 2.14: The thin client model of client-CSP communications consists of the exchange of user interface data.

Latency requirements

Human Computer Interaction (HCI) [38] research has resulted in the identification of response times of interest to user interactivity. Response times less than 150 milliseconds (msec) are considered imperceptible to the user. Response times between 150 msec and 400 msec are perceptible to the user but generally acceptable. Finally, response times above 400 msec are generally considered unacceptable. For thin-client computing this means that the time between a client's input and the corresponding update to the client's display must be less than the time budget associated with the desired user experience (e.g., less than 150 msec). That selected total time budget will be decomposed into the following five components:

i) sample the user input data

ii) transmit user input data over the network to the server

iii) process the user input data to affect the display at the server

iv) transmit display updates over the network to the client

v) render the display updates at the client

In a November 2000 comparison study [39] of thin-client systems operating over a 100 Mbps local area network, AT&T's Virtual Network Computing (VNC) was able to complete the user interface (UI) action of scrolling text in under 100 msec, while Citrix's Independent Computing Architecture (ICA) required just over 200 msec and Microsoft's Remote Display Protocol (RDP) required around 300 msec. The more intensive bitmap load UI action required just under 600 msec using VNC, just under 300 msec using ICA, and around 325 msec using RDP.

In a June 2002 comparison study [40] [41] of thin-client systems operating over both a low latency 100 Mbps local area network (LAN) and a high speed, 100 Mbps, wide area network (WAN) with an average latency of 64 msec; VNC was able to complete the text scroll action in under 100 msec over the LAN and around 150 msec over the WAN. ICA required around 175 msec over the LAN and around 225 msec over the WAN for the same text scroll action. RDP required just under 150 msec over the LAN and around 200 msec over the WAN. An image load action required just over 300 msec using VNC over the LAN and around 400 msec over the WAN. That same image load action required under 200 msec using ICA over the LAN and around 500 msec over the WAN; that action required just under 150 msec using RDP over the LAN and around 450 msec over the WAN. A web page load required 300 msec using VNC over the LAN and around 400 msec over the WAN. A web page load required just under 550 msec using ICA over the LAN and around 600 msec over the WAN; the web page load required around 600 msec over the LAN and just over 800 msec over the WAN.

A March 2006 study observed VNC response times [42] for various UI actions for image editing, presentation editing, and word processing applications across different network bandwidths and latencies. This study revealed that UI action response times were much more dependent on network latency than network bandwidth. As an example, when using the image editing application over a 100 Mbps network, an average network latency of 1 msec resulted in 99% of the UI action response times under 150 msec. When the average network latency was increased to 100 msec only 70% of the UI action response times were under 150 msec. When the same application was run over a 10 Mbps network, an average latency of 1 msec resulted in 98% of

the UI action response times under 150 msec; 64% for an average latency of 100 msec.

Network throughput (bandwidth) requirements

The network throughput requirements for thin-client systems depend on the: (*a*) frequency of UI actions (e.g., typing, scrolling, image loading), (*b*) volume of data transmitted for input device information and/or a display update in response to a UI action, and (*c*) desired latency.

The frequency of UI actions is determined by the application run on the thin client as well as user behavior. Web browsing by a user carefully reading web pages is likely to have low frequency (e.g., a screen full of text and images every few minutes), while video viewing by a user wanting high temporal resolution will have high frequency (e.g., 30 frames/sec). The volume of data transmitted containing input device information can be contained in a single minimum sized packet [39] [43]. The volume of data transmitted for a display update is dependent on the nature of the display update information and the pixel dimensions of the updated portion of the screen. Lastly, the desired latency is determined by the desired quality of experience (QoE) and the corresponding HCI user interactivity response time budget.

Those HCI user interactivity response time budgets set the total delay budget for the desired QoE; e.g., imperceptible response time has a maximum delay budget of 150 msec. That total delay budget must be partitioned among the five components described above; component (*ii*) determines the required thin-client to server bandwidth, while component (*iv*) determines the required server to thin-client bandwidth. Let's identify the network throughput required for a single UI action. Let m be the minimum packet size, U be the number of bits for an arbitrary display update, $d_{(ii)}$ be the portion of the delay budget allocated to component (*ii*), and $d_{(iv)}$ be the portion of the delay budget allocated to component (*iv*). Then the required network throughput from thin-client to server, T_{cs}, is:

$$T_{cs} = \frac{m}{d_{(ii)}}, \tag{2.1}$$

and the required network throughput from server to thin-client, T_{sc}, is:

$$T_{sc} = \frac{U}{d_{(iv)}}. \tag{2.2}$$

The expressions above characterize the network throughput dependence on the volume of data transmitted, bi-directionally, for a UI action and the desired latency. The sustained network throughput over time periods beyond that of a single UI action depends on the frequency of these actions over time. If the frequency is less than the desired latency, then the sustained network throughput will never exceed the throughput values characterized above. However, if the frequency of these actions is large enough such that the time between individual UI actions is less than the desired latency, then the sustained network throughput is an aggregate of the network throughputs of more than one individual UI action. As an example, for video viewing at 30 frames per second and a desired latency of under 150 msec, the UI action frequency of 30 updates per second requires an update every 33.33 msec. This time is smaller than the desired latency of 150 msec. Therefore the required network throughput of several UI actions will be aggregated; in this case the required network throughput of 4.5 UI actions.

According to the November 2000 study [39] referenced in Section 2.5.1, VNC sent 3.2KB from the server to the thin-client as a display update in response to the text scroll UI action. Whereas, RDP sent 1.9KB and ICA sent 1.1KB. The bitmap load UI action resulted in 84.3KB sent by VNC, 62.7KB sent by RDP, and 58.1KB sent by ICA. If the desired QoE is imperceptible response time, the delay budget is up to 150 msec. If 60% of that delay budget (90 msec) is allocated to component (iv), then for the bitmap load VNC requires 7.5Mbps network throughput from the server to the thin client, RDP requires 5.6Mbps, and ICA requires 5.16Mbps.

According to the June 2002 study [40] referenced in Section 2.5.1, VNC sent 60KB from the server to the thin client as a display update in response to a web page load UI action. Whereas, RDP sent 45KB and ICA sent 30KB. Looking at the latency data reported from this study presented in Section 2.5.1, it is observed that the overall latency for the web page load response was lower for VNC, compared to RDP and ICA. This lower latency is in spite of sending

more data, it may be due to the reduction in other components of the delay budget. It is very likely that the rendering of the display update at the client is consuming less time with VNC, compared to RDP and ICA, because VNC may have sent pixel data as opposed to high-level graphics commands. VNC has options to send raw pixel data or higher level graphics commands.

Packet sizes

In general, messages exchanged among nodes in a communication network are either smaller than a minimum size packet or larger than a maximum size packet. The packet sizes among thin-clients and servers follow this general trend, whereby input device data sent from thin-clients to servers are typically smaller than a minimum size packet and display updates sent from the server to thin-clients are typically larger than a maximum size packet.

The packet sizes sent from thin-clients to servers are almost always minimum size packets [39] [43] (e.g., 64 byte Ethernet frame). In contrast, display updates are often large enough to result in a packet train of maximum size packets (e.g., 1518 byte Ethernet frame) followed by a packet containing the remainder [39].

Summary: asymmetric traffic pattern and reducing latency

The input data being transferred from the thin client to the CSP is generally much smaller than the the display updates transferred from the CSP to the thin client. As a result, the network throughput from the CSP to the client will generally need to be larger than the client to CSP network throughput.

Circuit switching service can provide a particular guaranteed network throughput to keep the transfer of input data or display update data within the delay budget. However, the magnitude of the data transfers may vary dramatically over time and therefore the required network throughput would also vary dramatically. As a result, circuit switching would be an inefficient choice. Alternatively, packet scheduling mechanisms that provide probabilistic throughput guarantees would be much more effective. Differentiating the thin client traffic and using appropriate link-sharing packet schedulers would be best. There have been a variety of link sharing [44] packet schedulers

that have been proposed in the literature (e.g., Packet GPS [45], Deficit Round Robin [46]).

2.5.2 Communication within Cloud Service Provider Facilities

Four data center traffic studies [47–50] comprise what is understood as intra-Data Center traffic.

In [47] the authors analyze traffic data collected from a cloud data center that performs data analysis following a MapReduce style parallel data processing framework. In [48] the authors analyze traffic collected from several university, enterprise, and commercial cloud data centers. The fine grained packet level data the authors were able to utilize were from the university and enterprise data centers only. In this study, much of the traffic data collected is from network file systems and web applications. The web applications are not intra-data center traffic, instead the following discussions focus on the network file system traffic. In [49] the authors conduct a broad study of data center traffic volume using data from 30,000 servers in 50 data centers over a two-year time period. The traffic volume data are at the level of daily averages. In [50] the authors study the traffic in a single enterprise data center serving over one million users. The applications run in this data center do not perform parallel data processing. Instead, they include email, intranet web applications, file sharing, collaboration tools and a few specialized applications. Although this data center is for private use, it still acts as an internal cloud computing facility for the particularly company that owns it.

Data center architectures

Data centers consist of racks of servers interconnected via Ethernet switches. Each server rack (most commonly containing 48 rack units) consists of dozens of rack mounted servers and a Top-of-the-Rack (ToR) Ethernet switch. The ToR Ethernet switch contains at least one Ethernet port for each server in the rack along with several higher-speed ports to connect with a high-speed aggregation Ethernet switch. Several server racks are interconnected via high-speed aggregation Ethernet switches. As an example, if the ToR Ethernet switch connects via 10 Gbit/sec Ethernet to each server, the aggregation ports will be a higher-speed Ethernet technology such as 40 Gbit/sec Ethernet. The aggregation Ethernet switches then interconnect with data center routers

to interconnect to cloud clients via the Internet. The resulting network topology consists of three layers: rack, aggregation, and core. Figure 2.15 illustrates the network architecture of a typical data center.

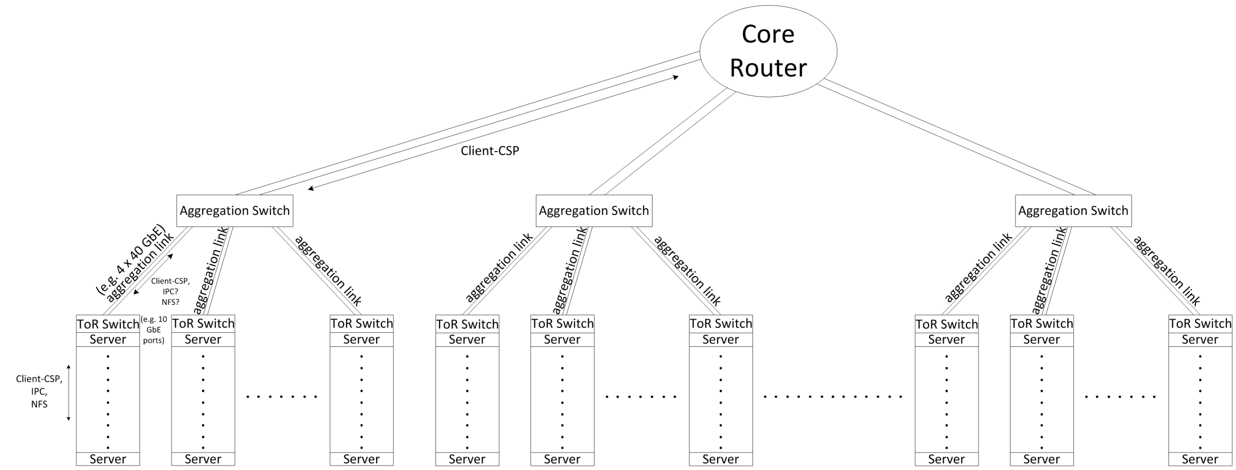


Figure 2.15: A typical data center network architecture. The network is organized into a three-level tree. The bottom level inter-connects several servers and storage devices inside a single rack. The middle level, called the aggregation level, inter-connects several racks of servers/storage devices; typically with a higher bandwidth network than that found at the bottom level. The top level, called the core level, inter-connects several aggregation switches to a router that inter-connects the entire data center with the Internet.

Macroscopic view of communication within a data center

The macroscopic communication patterns inside of data centers are fundamentally shaped by the mix of applications executing in the data center. These applications can be partitioned into three categories [47] [48]: *i*) Internet-facing applications (e.g., email, web search), *ii*) data mining applications (e.g., Map/Reduce), and *iii*) scientific applications (e.g., climate simulation).

Internet-facing applications primarily result in Client-CSP communication patterns that were discussed in the previous section. However, these application will also result in communication within the data center when accessing data via a centralized network file system. Data mining applications primarily result in intra-Data Center communication patterns. The data mining applications often consist of many threads processing shared data that are stored in a distributed network file system. These threads will synchronize their operations through inter-process com-

munication (IPC). The IPC can be via shared variables if all threads are allocated to the same shared memory computer or via messages transmitted over the network when all threads cannot be allocated to the same shared memory computer. Scientific applications have similar properties to the data mining applications except that they may consist of many more threads that will span multiple racks.

The Client-CSP communication leaving the data center is sometimes referred to as *north-south traffic*, while the communication that stays within the data center is referred to as *east-west traffic*. The specific application mix run in a data center determines the magnitude of north-south traffic in relation to east-west traffic. North-south traffic leaves server racks in the data center and travels through the aggregation and core layers onto the Internet. As a result of a work-seeks-bandwidth traffic pattern that has been observed inside data centers [47], east-west traffic primarily stays within a single rack. Observed data center traffic matrices indicate that most communication occurs among adjacent servers. The magnitude of traffic between servers decreases significantly as the distance between servers increases. Multi-threaded application programmers are aware of the cost of having cooperating processes/threads assigned to different racks and, therefore, design their software so that cooperating processes/threads are assigned to either the same shared-memory computer or to servers in the same rack, and thus are connected by a single ToR switch. As a result, east-west traffic is primarily constrained to a single ToR switch and rarely appears on an aggregation switch or on the core routers.

This macroscopic view of intra-Data Center traffic leads us to identify the following east-west traffic types: (*i*) inter-process communication for distributed memory parallel computing, (*ii*) centralized network file system data transfer, and (*iii*) distributed network file system data transfer. Figure 2.16 illustrates a taxonomy of the types of east-west traffic that stays inside a data center.

From the server's perspective, data presented in [49] indicates that servers inside data centers are primarily receivers of data or primarily senders of data. However, there is typically symmetry in the number of packets sent and received [49]. For example, if a server is primarily a receiver

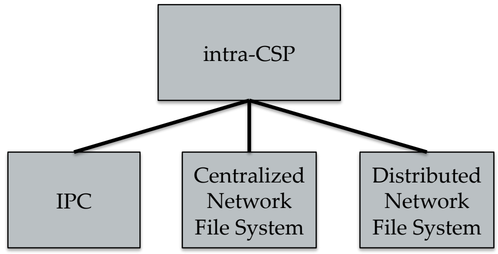


Figure 2.16: A taxonomy of cloud computing communication within a CSP facility. This class consists of three sub-classes: inter-process communication (IPC), centralized network file system (NFS), and distributed NFS.

of data, then it will send an almost equal number of packets but with much smaller packet sizes (e.g., Acknowledgment messages). The data presented in [49] also indicates that the average traffic rate of each server measured over a day is on the order of a few Mbps; 5.7 Mbps was specifically reported.

Microscopic view of communication within a data center

As discussed in Section 2.5.2, the microscopic communication patterns within a data center are shaped by IPC activity and network file system transfers, each of which are discussed below.

Inter-process communication (IPC) Inter-process communication (IPC) occurs between processes or threads cooperating in the execution of an application; it synchronizes their activities. This synchronization is used to either avoid race conditions and/or to coordinate computation. When cooperating processes co-exist on shared memory systems (e.g., multi-core systems), IPC will almost certainly be handled through shared variables. However, when processes cooperate across distributed memory systems (e.g., computing clusters), IPC must take place via message passing over the network; most commonly using the Message Passing Interface (MPI) library.

IPC traffic was not isolated in any of the intra-data center traffic studies, thus, future work could focus on this type of traffic in order to understand its magnitude inside the data center as well as its nature at the packet, burst and flow level.

Network file systems (NFS) The servers inside a data center will share data through a network file system, either centralized or distributed. In [48] it was shown that in a university data center flow sizes attributed to network file system traffic were roughly between 100 bytes and 10 Kbytes. At the packet level, packet sizes followed the typical IMIX or Broadcom packet size distribution [51] that has been observed in a variety of other contexts. The packet arrivals follow an ON/OFF model with the OFF period closely fitting a lognormal distribution and the ON period closely fitting a Weibull distribution. The packet interarrival times within the ON period closely fit a Weibull distribution.

Summary: more studies need to be conducted

With the limited number of intra-Data Center traffic studies there are quite a few opportunities for future work:

- measure the magnitude of north-south traffic in relation to east-west traffic
- measure the amount of east-west traffic that appears at the aggregation and core network layers in the data center
- distinguish between centralized and distributed network file system traffic in flow, burst, and packet level analysis
- flow, burst, and packet level analysis of IPC traffic

2.5.3 Communication among Cloud Service Provider Facilities

This class of communication refers to the communication among geographically diverse data centers operated by the same institution or among data centers operated by different institutions. From the literature, there is only one traffic study of the traffic between data centers [52]. Specifically, the authors of this study collected traffic data from edge routers of a network that inter-connects several Yahoo! data centers spread across several locations in the US, one location in the UK, and one location in Hong Kong. The traffic traversing these routers are both data

center to client (D2C) and data center to data center (i.e., inter-CSP or D2D). The authors developed a method to distinguish the D2C traffic from the D2D traffic.

The authors of [52] find that for certain Yahoo! data centers they studied the number of flows of the D2D traffic was strongly correlated to the number of flows of D2C traffic. They refer to this correlation as client triggered D2D traffic. On the other hand, there were other data centers that had weak or essentially no correlation among the two traffic types. They refer to that traffic as background D2D traffic. The authors infer that Yahoo! must have a hierarchical structure to their data centers whereby those data centers lower in the hierarchy have mostly D2C triggered D2D traffic and those at the upper levels consist of primarily background D2D traffic.

Unfortunately, the study presented in [52] does not provide a fine-grained view of what information is being communicated between the data centers. Therefore, there is a significant research opportunity represented by performing an inter-CSP communication study that is able to distinguish the type of traffic at least at the level that has been done by the study presented in [48].

Inter-CSP communication will be triggered by data migration for the purpose of: 1) migration of computational workloads or 2) redundancy. Future traffic studies of inter-CSP communication should seek to validate or invalidate this conjecture as well as provide insight into the properties of that class of communication.

Chapter 3: Computation Offloading

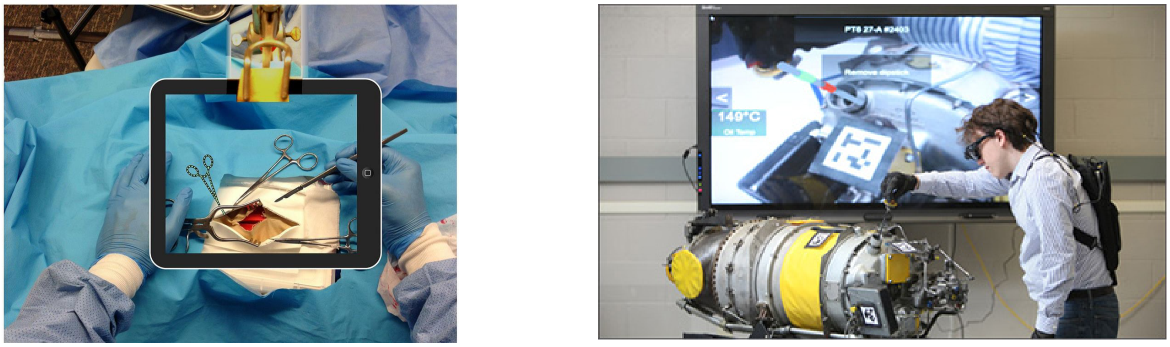
3.1 Definition and Use Cases

Cloud Computing is a complement for mobile devices. There are two ways in which a mobile device can benefit from cloud computing [53]: 1) *computation offloading* and 2) *capability extending*. **Computation offloading** is the idea of sending a fraction of a resource-intensive task to a more resourceful entity, executing that task in the cloud, and getting back the resulting output. Its main goals are: a) to extend the battery life of a mobile device and b) to make the application run faster. **Capability extending** is a way in which cloud computing extends the mobile device's capabilities (e.g. storage, networking, computation). Mobile devices can store and synchronize data with the cloud (e.g. images, videos, contacts, etc.), process data with a specialized software tools in the cloud (e.g. Amazon EMR), or interconnect different users to share information (e.g. video conferencing). This project focuses on computation offloading.

Some of the use cases of computation offloading include: 1) augmented reality and cyber-physical systems, 2) battlefield operations, and 3) wearable computing. There are several benefits to *Computation Offloading*. Some of these benefits include [9] [53] [54]: 1) improved performance for applications with time constraints, 2) energy savings on the mobile device, and 3) re-allocation of mobile device resources to other tasks.

Augmented reality is a new technology that integrates digital information with the real environment to enhance the user's experience. In the medical field, augmented reality can help doctors to train their practitioners. As shown in Figure 3.1a, an application is capable of rendering visualization datasets on a real patient to identify organs, and have precise information on where to perform an incision during a surgery [55]. In manufacturing, augmented reality is helpful to technicians that repair or assemble components to build final products. Augmented reality applications show step-by-step instructions as 3D drawings on the actual equipment [55] as illustrated in Figure 3.1b.

Mobile augmented reality applications, like the one depicted in Figure 3.2, perform image



(a) Medical application

(b) Manufacturing application

Figure 3.1: Augmented reality applications [(a) https://engineering.purdue.edu/isat/author/isat/ - (b) http://www.advice-manufacturing.com/Virtual-and-Augmented-Reality.html]

based object recognition [2] by using the camera, other sensors, image processing and computer vision to identify the user's environment and overlay of contextual information (e.g. street names, description of objects, directions to drive) of the surroundings after the processing. These applications can help people to easily identify places of interest such as restaurants, ATMs, streets, museums, buildings, etc. An application presented in [56], uses augmented reality to track feature points on videos; it renders 3D objects on matching sets of 2D images. However, all of these augmented reality applications require a lot of processing that may not be computed in the specified time frame by the mobile device; in such situations computation offloading comes into play. Augmented reality falls under the three benefits of computation offloading, since this technology can significantly improve its performance by using external resources that provide more computing capability. At the same time, it saves energy on the mobile device when performing heavy intensive computations, and allows the equipment to work on other tasks.

Other applications that can benefit from computation offloading are the military battlefield operations on remote places, as shown in Figure 3.3, where there is limited or no infrastructure at all. In these situations, soldiers have missions where they collect and process data from the environment. These data includes information about possible enemy threats, detection of biological hazards, or voice captured from people speaking in different languages. However, soldiers are equipped with resource constrained mobile devices, that are unable to perform such

Figure 3.2: Augmented reality example [http://www.mobimanage.com/5-plug-n-play-marketing-tools-to-make-you-stand-out/]

computations. In such cases, computation offloading is required to alleviate the shortcomings. Soldiers have three options: 1) offload jobs to other soldiers' mobile devices, 2) offload jobs to nearby surrogates located in tanks, helicopters, or tents, and 3) offload jobs to remote data centers. Battlefield operations fall under the three benefits of computation offloading, because mobile applications experience improved performance when soldiers send data to other resourceful entities (e.g. clouds or cloudlets), and the energy used for computation tasks on the mobile device gets reduced. By saving time and energy, soldiers may use their portable devices for other tasks.

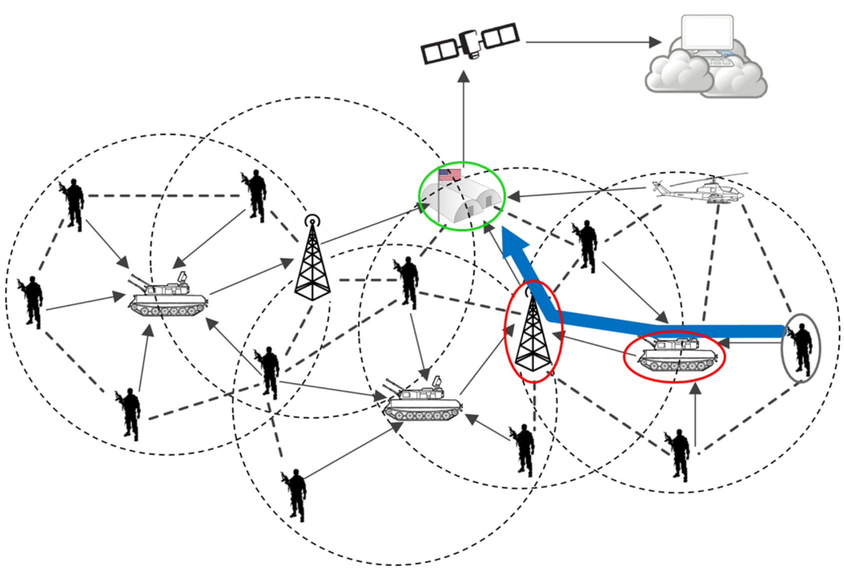


Figure 3.3: Battlefield operations

Wearable computing is another technology under development that facilitates the human-

computer interaction. Sensors collect body information that can be useful in biomedical applications. Medical doctors and physicians can access these data to keep track of the patients' health; in this way, they can make recommendations and plan effective rehabilitation strategies. Potential applications of wearable computing include the monitoring of: 1) blood pressure, 2) chronic pulmonary disease, 3) hemiplegic patients, and 4) Parkinson's disease [3] [4]. The monitoring of patients can be performed over long periods of time (e.g. days, weeks, or months). Remote execution is required since the collected information is analyzed by specialized medical software. Wearable computing sensors regularly upload the data to databases on remote locations through a network (e.g. Internet). Wearable computing may use different technologies to communicate with cloud resources [57], as illustrated in Figure 3.4. After collecting and processing the statistics from the data sets, the system detects and predicts the patient's clinical situation. Wearable computing falls in two out of the three computation offloading benefits. Wearable computing gains improved performance, because the sensing devices send the collected data to other entities that process the statistics. Remote processing of the data also saves energy on local devices.

Figure 3.4: Wearable computing communication technologies [1]

3.1.1 Enabling Architectures

There are several cloud architectures that enable computation offloading. Portable devices (e.g. smartphones, tablets, laptops, etc.) benefit from resourceful entities (e.g. cloudlets and/or clouds) by using the client-server model to offload applications. There are three types of architectures [53]: 1) *Centralized Cloud*, 2) *Cloudlet*, and 3) *Ad Hoc Mobile Cloud*, as shown in Figure 3.5.

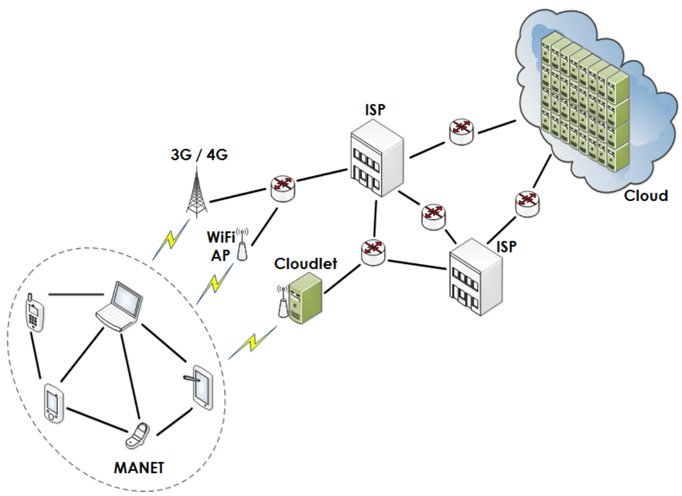


Figure 3.5: Mobile Cloud Computing Architecture

Centralized Cloud

The *Centralized Cloud* is referred to a big data center located far away from the client (i.e. mobile device). It is usually accessed through a cellular network (e.g. 3G, 4G, WiMax) or wireless access points (e.g. WiFi). This infrastructure is suitable for tasks that do not require much interaction between the client and the server; it is mainly used to store and process data for non real-time applications. Also, it can be used whenever there are computational jobs that cannot be fully completed neither by the mobile device nor a nearby surrogate. Examples of centralized clouds include: Amazon Web Services, Microsoft Azure, and Google AppEngine.

Cloudlet

A *Cloudlet* is a resource-rich entity (e.g. a small data center) located one-hop away from a mobile device, and connected to the Internet and other powerful resources through a high-speed connection. This architecture is very useful for real-time applications that are highly interactive and delay-sensitive (e.g. online games, language translation, image recognition). In the near future, commercial cloudlets may be deployed in public places such as coffee shops, shopping centers, bookstores, and airports. For military operations, cloudlets may be located in tanks, helicopters, towers, and shelters.

Ad Hoc Mobile Cloud

An *Ad Hoc Mobile Cloud* is the group of several mobile devices that form a network to share resources. This type of architecture is also known as a *Mobile Ad Hoc Network* (MANET). It is useful whenever clouds and cloudlets are not available or there is poor connectivity to remote servers. In this approach, a mobile device collaborates with other mobile devices by sharing its resources and processing part of a task. MANETs can be found in places such as sport events (e.g. golf, horse races, etc.), rescue missions (e.g. firefighters' emergencies), or social events (e.g. conferences, workshops, politician meetings, etc.), where there is no communication infrastructure.

3.2 Computation Offloading Frameworks

There are several research projects that have built computation offloading systems that demonstrate its feasibility. Some of these projects are: 1) Spectra [58], 2) SlingShot [59], 3) Chroma [60], 4) MAUI [61], 5) CloneCloud [62], 6) ThinkAir [63] [64], 7) eXCloud [65], 8) COCA [66], and 9) Cuckoo [67].

3.2.1 Spectra

Spectra [58] is a framework that determines how and where to execute an application by monitoring the availability and usage of resources of such application. The system considers three

factors: 1) performance, 2) energy, and 3) quality. Spectra uses history-based prediction, since it measures the application's usage and generates a model that helps to predict future demand. This framework is useful for applications with longer execution times (e.g. a second or more), due to the overhead added by the system when estimating the correct decision. Spectra works with a file system called *Coda*. Coda is a good solution when working under low bandwidth, since it buffers the file modifications and reintegrates such changes to the file servers in the background. Since Spectra predicts which files will be used, it verifies if some of them have buffered modifications, and triggers the reintegration of the changes to the servers before starting the job. Spectra identifies code components that may benefit from computation offloading and generates three execution plans: 1) local, 2) remote, and 3) hybrid. Spectra is considered a M-ary framework, since it is able to choose the remote server to use, measures the local and remote resources, and concentrates the server status information (e.g. availability, CPU load, cache status, etc.). Spectra has six types of monitors: 1) CPU, 2) network, 3) battery, 4) file cache state, 5) remote CPU, and 6) remote file cache state. These monitors help Spectra to provide a more accurate status of the local and remote resources that are available. The CPU monitor measures the cycles recently used by processes. The network monitor predicts the bandwidth and latency, and records the number of bytes sent and received; a short transfer approximates the round trip time, and a long transfer approximates the throughput. The battery monitor measures the energy consumption and reports the client's remaining energy. The file cache state monitor observes the Coda files accesses that are used to predict future file accesses. The remote proxy monitor continuously polls the available servers and gets the total resource usage. In order to select the best option, Spectra considers: 1) execution time, 2) energy usage, and 3) fidelity. The execution time includes the summation of the local and remote execution time, network delay, time to serve cache misses, and time to ensure data consistency. To estimate the computation delay, Spectra divides the predicted CPU cycles needed by the cycles per second of a particular machine. To estimate the communication delay, Spectra divides the predicted number of bytes to be transmitted by the available bandwidth. To estimate the latency, Spectra

multiplies the estimated number of Remote Procedure Calls (RPCs) by the round trip time. Spectra has been tested in three applications: 1) a speech-to-text translator called *Janus*, 2) a system to prepare Latex documents, and 3) a language translator called *Pangloss-Lite*. Spectra evaluates the trade-offs between execution time, energy, and fidelity. When energy is a concern, Spectra chooses the remote execution over the local execution. When the execution time starts to increase, the system may decide to decrease the fidelity. Spectra also prefers energy savings over faster execution time. Spectra's effectiveness is within 2% of the best option, and the utility within 7%. The overhead induced depends on the number of servers that are candidates for computation offloading. For 5 servers, the overhead is 74 ms., which is good for tasks that last for a second or more.

3.2.2 SlingShot

SlingShot [59] is a framework that combines the use of cloud resources and nearby surrogates (i.e. close computing devices). Slingshot does not perform migration; instead it takes advantage of replication. Migration means to suspend an application, save its current state, transmit the state to another entity, and resume the execution on the new computing device. Replication means to instantiate multiple clones on different computing devices. The advantage of replication over migration is that with replication there is always at least one entity serving the user, while migration needs to halt the application when shifting the execution to another device. The main goal of SlingShot is to have a trustworthy environment prone to failures, by replicating and executing applications on different entities. Slingshot replicates applications on two types of computing devices: 1) a remote server (i.e. home server) and 2) surrogates located at hotspots (e.g. cloudlets). The server's replication is known as *first-class replica* and the surrogates' replications are known as *second-class replicas*. Replication is useful in two ways: 1) from all the replicas, the system returns the fastest response to the user, and 2) in case of a failure, other replicas can service users' requests. The advantage of replicating on nearby surrogates is that the network latency is reduced, since these computing devices are located one-hop away from the

user and reachable through a high-bandwidth connection (e.g. 54Mbps). These surrogates may be deployed in public places such as coffee shops, shopping centers, bookstores, airports, etc. SlingShot partitions the application is two parts: 1) local component and 2) remote component. The local component is referred to the methods that are executed on the mobile device, usually graphical user interfaces. The remote component is the resource-intensive methods that can benefit from replication. After the partitioning, SlingShot creates a first-class replica (i.e. home server) and starts the execution of the application. At runtime, SlingShot starts instantiating one or more second-class replicas (i.e. surrogates). The *first-class replica* services user requests from a database contained in the remote server. *Second-class replicas* need to verify first if they can service user's requests from their cache or if they need to get a copy from the remote server. When a user connects to a surrogate, it sends the IP address of its home server and the service ID of the remote service it wants to instantiate. The remote server creates a *first-class replica* and starts executing the application; it also creates a checkpoint of such application to indicate the current execution state of the replica. Mobile devices log all the events and store the checkpoints in order to easily create replicas on surrogates. When a surrogate creates a *second-class replica*, it tries to serve users' requests from its cache, however, if the cache does not contain the needed piece, then the surrogate asks the remote server for the chunk. The surrogate then receives the checkpoint generated by the remote server and resumes the execution of the application. SlingShot has been tested on a speech recognition application and on a VNC session. Results from experiments show that Slingshot is 2.6 times faster than remote execution (i.e. remote migration). The drawbacks of remote execution are high latency and limited bandwidth. For the speech recognition application, remote execution took 229ms to transfer 44KB, and 77ms using SlingShot. For the VNC session, remote execution required 15.6 seconds and SlingShot just 3.2 seconds. If the system does not use the cached checkpoints, the speech recognition application requires the surrogate to contact the remote server, and takes around 28:06 minutes to transfer the state from the remote server to the surrogate. When using the cache option, it only takes 3:35 minutes to create a second-class replica.

3.2.3 Chroma

Chroma [60] is a framework designed to make partitioning decisions. The partitioning process of an application is reported in small descriptions called *tactics*. A tactic consists on the remote calls that can be used by an application and the sequence in which these calls need to be executed. After the tactics are defined, Chroma has to decide the tactic plan by predicting the resource usage of each tactic. This approach is performed by using history based prediction. Whenever a tactic plan is executed, the resource usage is logged for future demand prediction. Chroma continuously monitors the current resource availability (e.g. memory, CPU, bandwidth, energy, etc.) and evaluates the tradeoffs between accuracy and latency (e.g. accurate results or quick response times?). Chroma has three objectives: 1) seamless to the user, 2) effectiveness, and 3) minimize programmer's responsibility. The first objective is achieved by making Chroma automatic from the user's perspective. At runtime, Chroma is able to decide how and where to execute the applications. To reach the second objective, Chroma has to test every partitioning strategy and then pick the best option. The third goal is attained by using a generic API that is compatible with Chroma. Chroma is also able to exploit idle resources in order to improve the quality of experience in three ways. The first way is to call several remote calls of the same operation to different servers and use the fastest result. The second way is to split the job among different servers. And the third way is to perform the same operation on different servers and with different fidelities; the result with the highest fidelity and that meets the application's constraints is the one chosen by Chroma. This framework has been tested in three different applications: 1) a language translator called *Pangloss-Lite*, 2) a speech-to-text translator called *Janus*, and 3) a face recognition program called *Face*. For the *Pangloss-Lite* application, most of the decisions made by Chroma are the best option; there are few cases where Chroma made a different decision of up to 30%. When testing the system under the *Janus* application, Chroma was able to choose the best option most of the time; when the best option was not selected, Chroma had differences that ranged between 2% and 6%. With the *Face* application, Chroma

was able to select the best option in all of the cases.

3.2.4 MAUI

MAUI is a computation offloading system presented in [61]. The MAUI system consists of several entities. The first is a *programming environment and runtime tool* to offload code at the function or method level. In the programming environment, software developers can tag methods as offload candidates. Typically, any method that does not render the user interface or interact with a local device can be offloaded. The second entity is a *profiler* that dynamically collects data such as mobile device energy consumption, method call energy consumption, and communication network bandwidth. The third and final entity is a *binary decision algorithm* that utilizes the data collected by the profiler to determine whether a particular method should be offloaded to the single offload target. This binary decision algorithm is the solution of a binary integer mathematical program with an objective function to minimize energy consumption subject to a particular completion time constraint. The decision algorithm executes on the offload target to avoid burdening the mobile device.

MAUI is very similar to CloneCloud. Their main difference is that MAUI requires the programmer to specify the portions or methods of a program that need to be migrated into the cloud. To do so, programmers mark as "remotable" the methods that are candidates for offloading, based on some constraints. Methods with the following characteristics are not good candidates for offloading : a) methods that require user interface, b) methods that require I/O devices, and c) methods that interact with external components. The MAUI server gets a copy of the program in two ways: 1) directly from the mobile device, or 2) from a link to the location of the executable (provided by the mobile device). MAUI has been tested on applications such as: i) face recognition, ii) video games, and iii) chess.

3.2.5 CloneCloud

CloneCloud [62] is a project where threads are migrated from mobile devices to clones in the cloud. These threads are executed remotely and reintegrated back to the mobile device. However, there are some constraints on the methods that are not candidates for offloading: a) methods that require access to local resources (e.g. camera, GPS, microphone, sensor), b) methods with dependencies, and c) methods with nested migration. CloneCloud has three main components: 1) Static Analyzer, 2) Dynamic Profiler, and 3) Optimization Solver. The first component is the one that identifies the methods that are candidates for offloading based on the constraints. The second component evaluates the cost on each of the resources (e.g. mobile device and cloud). And the third component combines the results from the other two components to make the decision of what portions to offload.

3.2.6 ThinkAir

ThinkAir [63] [64] is a framework that works at the method-level computation offloading, and allows users to migrate their applications to the cloud. ThinkAir has two main features: 1) it exploits parallelism by creating, suspending, resuming, and destroying Virtual Machines (VMs), and 2) provides an efficient resource allocation mechanism. ThinkAir consists of three components as shown in Figure 3.6: 1) execution environment, 2) application server, and 3) profilers. The execution environment has four parts: 1) programmer API, 2) compiler, 3) execution controller, and 4) execution flow. The *programmer API* is the tool used by programmers to annotate any method that may be considered for offloading. In this way the execution controller detects the methods that are candidates for offloading, performs the profiling, makes a decision, and communicates with the server. The *compiler* is in charge of translating the code that will be run on the cloud. The *execution controller* is responsible for making offloading decisions at the client side. It considers the current environment conditions and analyzes past executions. It evaluates the network quality and decides where to execute the workload; ThinkAir considers

offloading only if the connection is good, otherwise it prefers local execution. The *execution flow* refers to the decisions made by the execution controller; if the methods are offloaded, then the mobile device sends the application to the server and waits for the outcome, otherwise it continues the execution locally. The application server is the component that handles the code on the server side. It has three parts: 1) client handler, 2) cloud infrastructure, and 3) automatic parallelization. The *client handler* is in charge of the connection between the client and the server; it receives and executes the code, and then it returns the results. In case of an exception (e.g. out of memory), the client handler dynamically allocates a more powerful VM. Parallelism helps to reduce execution time and energy consumption. The client handler resumes the needed VMs, distributes the workload among them, collects and merges their output, and finally sends the results to the client along with some profiling information that may be useful for future offloading decisions made by the *execution controller*. The *cloud infrastructure* manages the two types of servers in the cloud: 1) primary server, and 2) secondary servers. The primary server is always online and waiting for client's requests, while the secondary servers are resources allocated to users on-demand and that have three states: 1) powered-off, 2) paused, or 3) running. If a secondary server is powered-off, it does not have any resources allocated. If the secondary server is paused, it has some resources allocated but it is not consuming any CPU cycles. If the secondary server is running, it has resources allocated and is using the CPU. Allocating and deallocating resources (e.g. secondary servers) introduce delays; a VM takes around 300 milliseconds to resume from a paused state and between 6 to 7 seconds when multiple VMs resume at the same time. To start a VM from the powered-off state, it takes around 32 seconds on average. The *automatic parallelization* component is in charge or exploiting parallelism in two ways: 1) using multiple processors, or 2) partitioning and distributing the workloads among several VMs. Profilers are very important, since they collect and provide useful information when making computation offloading decisions. ThinkAir has three kinds of profilers: 1) hardware profiler, 2) software profiler, and 3) network profiler. The *hardware profiler* monitors parameters such as: CPU utilization, display brightness, and the radio's (e.g. WiFi, 3G) power state. The

software profiler measures the execution time of the methods, threads' CPU time, instructions executed, and methods called. The *network profiler* records the amount of data that is being sent/received between the client and the cloud, and estimates the bandwidth. All these profilers provide useful information to the *energy estimation model* which dynamically estimates the energy consumption of each of the methods. ThinkAir has been evaluated on different applications: 1) a puzzle called *N-queens*, 2) a face detection application, and 3) a virus scanning algorithm. After testing ThinkAir with the *N-queens* puzzle, it is observed that as N increases (e.g. N > 5), the execution time and energy consumption is less than that on the mobile device; the time and energy spent to send and process the data in the cloud (e.g. 2 seconds and 70 joules) is less than the time and energy spent by the mobile device (e.g. 80 seconds and 8000 joules) with this heavy computation. The face detection application showed a similar outcome; as the number of photos increases, the cloud provides better results. The virus scanning algorithm also had a poor performance when executed locally; it took one hour to process 3,500 files, and only three minutes when offloaded.

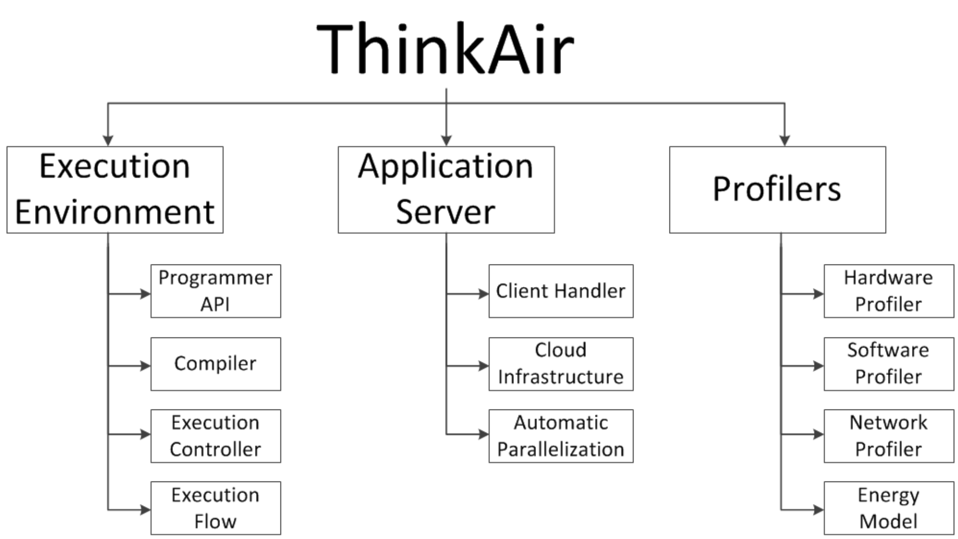


Figure 3.6: ThinkAir Structure

3.2.7 eXCloud

eXCloud [65] is a framework that considers two types of nodes: 1) cloud nodes and 2) mobile nodes. *Cloud nodes* are also known as computing nodes and are located in remote data centers; they provide high performance computing over data that requires heavy computation. *Mobile nodes* refer to the mobile devices that are equipped with limited amount of resources (e.g. memory capacity, CPU capacity, storage capacity, and energy); they are not designed to provide high performance computing. When combining cloud nodes and mobile nodes to form a mobile cloud computing environment, the thin-client model is adopted. Cloud nodes receive the input data from mobile nodes through a wireless network and perform all the heavy computation. On the other hand, mobile nodes just render the graphical user interface with the resulting output received from the cloud nodes. *eXCloud* works with a computation migration technique called *stack on-demand (SOD)* that distributes tasks across different nodes. SOD only transfers the required data to the cloud (i.e. part of the stack), and allows three types of migration: 1) among cloud nodes, 2) from mobile nodes to cloud nodes, and 3) from cloud nodes to mobile nodes. The migration occurs in two ways: 1) actively and 2) proactively. *Actively* means that the migration will be triggered when certain conditions of the system are met (e.g. threshold). *Proactive* migration occurs when there are certain exceptions (e.g. out of memory, library not found, etc.). *eXCloud* has six modules: 1) class preprocessor, 2) migration manager, 3) object preprocessor, 4) worker manager, 5) communication manager, and 6) resource manager. The class preprocessor module is in charge of translating the Java application bytecode. The migration manager module collaborates with the communication manager and other migration managers to serve the migration requests. The object preprocessor module synchronizes the required objects among the nodes. The worker manager module creates the process that will handle the migration request. The communication manager module coordinates the communication among the cloud nodes. The resource manager module handles proactive migrations by requesting missing resources in the current node. This framework has been tested in four different programs:

1) Fibonacci number calculation, 2) *n-queens* problem, 3) *traveling salesman* problem, and 4) Fast Fourier Transform computation. The experiments were performed to compare different migration mechanisms such as: 1) SOD, 2) JESSICA, and 3) G-JavaMPI. The metrics that were measured are: 1) capture time, 2) transfer time, 3) restore time, and 4) migration latency. From the results, SOD outperformed the other migration mechanisms in terms of the migration latency, followed by JESSICA2 and G-JavaMPI. SOD migration latencies ranged between 250ms and 400ms, and were between 3 and 56 times faster than the other approaches.

3.2.8 COCA

Computation Offload to Clouds using AOP (COCA) [66] is a framework designed for Android to work at the source level. COCA is useful whenever a job on a smartphone requires heavy computation and there is a connection to a network with remote servers. Aspect-Oriented Programming (AOP) refers to particular features that an application has (e.g. security, logging, checking, etc.) and that are written as modules. COCA uses AspectJ which is an Eclipse's plug-in written in Java to define aspects. COCA's design consists of three stages: 1) profile, 2) build, and 3) register. The profile stage is where COCA identifies all the *pure functions*. These functions are the ones without dependencies, where variables and objects are just used within the function. Also within this stage, COCA estimates the processing time of these functions and the needed resources if offloaded to a remote server. Once COCA profiles the functions, it generates a report with the results that will help in the decision making process. This framework considers the actual execution time and the available bandwidth for offloading. The build stage consists of scripts that partition the original source code into two parts: 1) functions that are candidates for offloading and 2) functions that may not be offloaded. The scripts generate two files: 1) a *jar* file for the cloud server and 2) an *apk* file for the mobile device. Within the register stage, COCA sends the *jar* file to the cloud server that keeps a database with all the functions that are candidates for offloading. The *jar* file indicates what functions should be loaded to serve user's computation offloading requests. COCA has been tested on a chess game that requires a lot of

computation to calculate the next moves in the game. From experiments, the build stage has an overhead that ranges between 195 nanoseconds and 290 nanoseconds per function call, which is considered negligible. The communication cost of this framework varies significantly depending on the communication technology used. COCA performs better over a WiFi connection, since the communication cost is approximately 77 milliseconds when transmitting 30 Kilobytes, compared to a communication cost of 1,622 milliseconds over a 3G network. In terms of energy savings, COCA has decreased its energy consumption up to 56%. From an experiment, it was observed that when COCA was not used, the smartphone consumed 443.88 micro AH, but when COCA was enabled, the mobile device consumed 247.91 micro AH. Around 42.8% of the total energy consumption is attributed to the processor when COCA is not used. However, if COCA offloads jobs to the cloud, then the processor uses less than 1% of energy because it is idle most of the time.

3.2.9 Cuckoo

Cuckoo [67] is a project that consists on deciding if an application should be run locally or remotely. The main goal is to maximize the computation speed and minimize the energy consumption. Cuckoo was develop in Java and for Android applications only. The decision making algorithm always selects remote execution if there are available resources. For the resource discovery, it uses QR codes that contain the address of the server. Clients request the server to execute a method, but if the service is not available on the server, the client must provide the installation file. Some applications that use Cuckoo are: 1) *eyeDentify* [68] which is an object recognition application and 2) *PhotoShoot* which is a real-time game that uses the camera as a gun to shoot other players.

3.2.10 Other Approaches

There is another computation offloading method called **Bayesian Approach** [69] which is a technique that evaluates the cost of executing locally and executing remotely; remote execution

includes sending the job into the cloud, processing the job, and getting back the results. This approach makes binary decisions; it doesn't evaluate the offloading on several cloud resources. Offloading becomes beneficial when local execution time is greater than remote execution time. In this approach, the remote computation cost is expressed in terms of the local computation cost to determine the *Critical Network Bandwidth*. This concept refers to the threshold that determines the minimum required bandwidth that makes offloading a beneficial choice. This technique keeps a history of previously observed bandwidths to predict the future bandwidth. Also, it estimates the penalty cost of making a local or remote decision in order to minimize the execution cost.

There are more approaches used on different projects. One of them is called *Shortest Queue (SQ)* [10] where mobile devices choose the surrogate with the shortest waiting queue. Another approach is *Minimum Expected Completion Time (MECT)* [10] where mobile devices choose the surrogate with the highest computation speed. *Computing While Charging (CWC)* [10] is an approach where mobile devices are replenishing their batteries and willing to cooperate with other nodes to complete a task. *Group Collaboration* [10] is a technique where mobile devices partitions a job and each one executes part of it, later they merge and share the results. Partitioning a program can be either *Static* or *Dynamic* [8]. *Static* is when the program gets partitioned before the execution of the application; the mobile device has knowledge of what functions are offloading candidates in advance. *Dynamic* is when the program starts its execution and as it needs more resources, some of its functions are delegated to surrogates; partitioning occurs while the application is running. Another technique is the migration of *Virtual Machines* [6]. A virtual machine saves its state, gets suspended, and migrated to another surrogate, to later get resumed and continue its execution. Some other approaches avoid the transferring of data as much as possible by providing links to the surrogates where they can get access to the job and/or the input data. Other approaches [9] use *probabilistic prediction*, *history-based prediction*, or *network bandwidth prediction* to estimate future offloading decisions.

3.3 Computation Offloading Decision Criteria

Computation Offloading has two main classes of decision: 1) *what* computation to offload and 2) *where* to offload. The *what* computation to offload decision is achieved by using the partitioning approach, where the application that requires heavy computation is divided in two parts: 1) the one that may be computed locally, and 2) the one that is candidate for offloading. The *where* computation to offload decision determines the location in which the computation will take place. Most of the frameworks and research studies focus on the binary decision problem, where they just consider two options: 1) to offload and 2) not to offload; in this approach, a client views the cloud as a single entity. On the other hand, there is the *M-ary* decision problem, where the client not only considers offloading, but also recognizes more than one cloud entity (e.g several cloud resources). This approach becomes more complex, since the client has more estimation to perform in order to make a good decision; a client needs to evaluate all and every possible available offloading location. A computation can take place on different types of entities (e.g. mobile device, cloudlet, cloud) as mentioned in the previous section. To make a computation offloading decision it is important to evaluate the trade-offs between communication and computation. The client needs to estimate the time for local computation and the time for remote computation that includes the time to exchange the input and output data between the mobile device and the surrogate (i.e. cloud resource). Computation offloading becomes beneficial whenever a program requires heavy computation and the computing resources are significantly faster than the local computing resources. Also, when there is high bandwidth available and/or only a small amount of data needs to be transferred between the client and the server. The objectives of computation offloading include: 1) minimize the computation completion time, 2) shift the energy consumption to a more resource-rich facility, and 3) optimize the cloud computing resource allocation.

Computation Offloading is complex since several elements need to be considered [8] [70]. Whether the offloading is static or dynamic, deciding where to offload (e.g. cloud, cloudlet,

mobile device), choosing a single surrogate or multiple surrogates, identifying the location of the job and the input data to be processed (e.i already in the cloud or provided by the user). Also, it is important to consider the surrogate characteristics (e.g. memory capacity, CPU speed, storage capacity, responsiveness, etc.), and if the mobile device will migrate the whole application or just part of it. A *Computation Offloading Taxonomy* is shown in Figure 3.7.

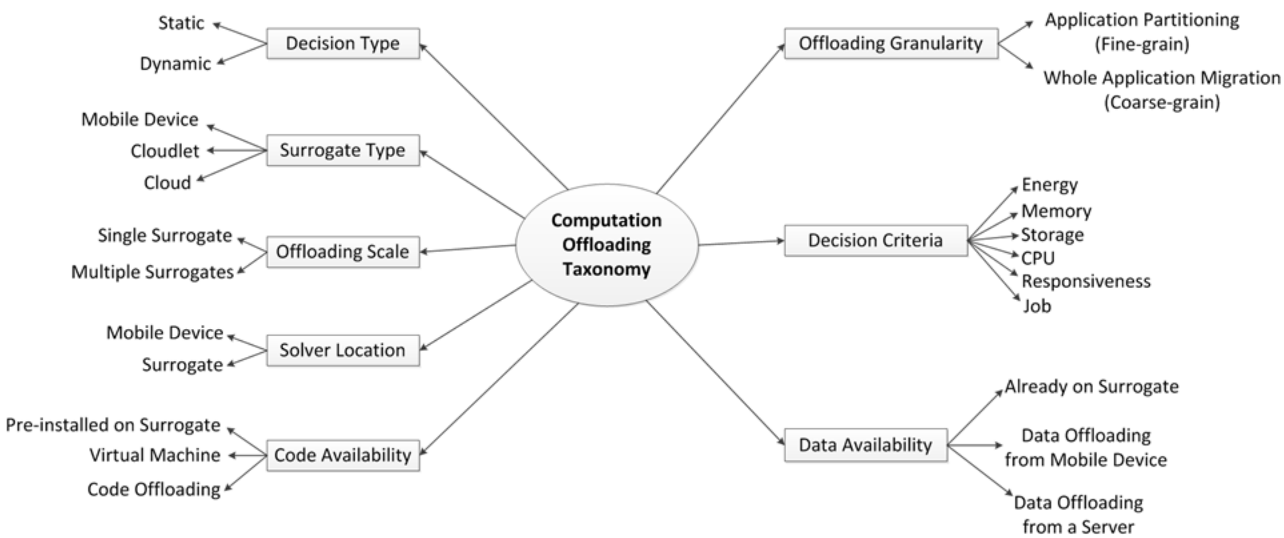


Figure 3.7: Computation Offloading Taxonomy

There are several criteria that influence the computation offloading decision and are organized in several sets [8] [71]. The first set of criteria are related to the *Cloud Service Provider (CSP)* considerations that evaluate the operation costs, the available resources, the revenue, and the location. The second set of criteria are related to *User Preferences*, where the user considers the cost of using a remote resource, the security of the cloud, the reliability of the servers, and the responsiveness of the service. The third set of criteria are related to *communication network conditions* such as available bandwidth, link quality, congestion, and number of hops to reach a cloud resource. The fourth set of criteria are related to the *Surrogate Characteristics* such as CPU speed, memory capacity, storage capacity, current system load, and available resources on the servers. Finally, the fifth set of criteria are related to the *Computation Job Characteristics* such as the size of the computation, the size of the input data, and the size of the output data.

A taxonomy of the *Decision Criteria* is shown in Figure 3.8.

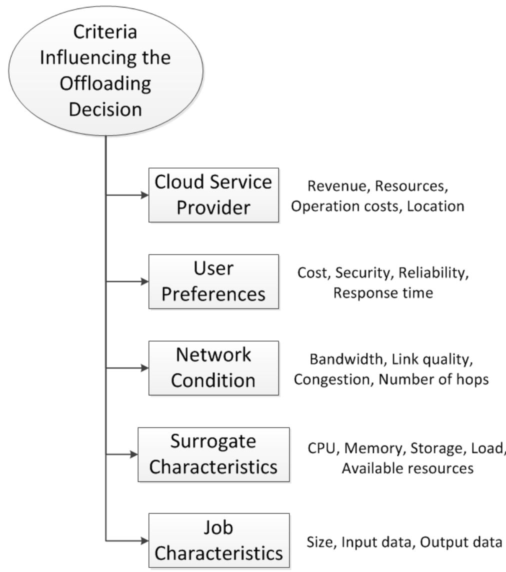


Figure 3.8: Decision Criteria Taxonomy

Chapter 4: Computation Offloading Decision Analysis

4.1 Introduction

Computation offloading has two main goals: 1) reduce the energy consumption on mobile devices, and 2) reduce execution time of applications [9]. The focus of this dissertation is on the reduction of the execution time of offloaded jobs. Whenever mobile device users experience a situation in which they are unable to execute an application within certain time constraints, they might consider computation offloading. However, a decision analysis must be performed before taking any action. A trade off between the communication and computation delays needs to be considered, as illustrated in Figure 4.1. In general, computation offloading is beneficial when: 1) a job contains a large amount of computation, 2) only a small amount of data needs to be exchanged between the mobile device and the cloud resource, and 3) when high network bandwidth is available [9] [72]. Whenever there is a large amount of data to transmit and the computation is small, computation offloading becomes less attractive and even more expensive than using local resources. In other cases, the computation offloading decision depends on the available bandwidth to exchange data between entities; with high bandwidth, computation offloading may be a good option. For this reason, it is important to consider the characteristics of: 1) the job, 2) the available resources, and 3) the network. These factors influence the ratio between computation time and communication time and determine if computation offloading is beneficial. Some researchers have focused on determining if computation offloading is beneficial by analyzing the relationship between three factors: 1) network bandwidth, 2) amount of computation to be performed, and 3) amount of data to be transmitted over the network [72]. Other researchers have also included in their analysis some servers' characteristics such as: 1) execution speeds, 2) available memory, and 3) load [9]. Usually the client needs to send the input data to the cloud for the first time, but subsequent computations may save energy and time by pointing to the data that is already in the cloud.

This chapter presents an analysis that considers several variables that impact computation

offloading decisions. The objectives of the analysis are: 1) identify the conditions in which computation offloading is beneficial for reducing job completion time, and 2) evaluate the impact of estimation error on the decision among M offload options. We consider the first objective in this chapter and the second objective in the next chapter. Most of the research projects, like the ones presented in Chapter 3 (e.g. MAUI, Cuckoo, CloneCloud, ThinkAir, etc.), study the binary decision problem, where the user contemplates only two options: 1) to offload, and 2) not to offload. In this kind of decisions, the user observes the cloud as a single entity. However, recent projects have addressed the *M-ary* decision problem, where users not only make a decision on whether to offload or not, but also decide where to offload their computational jobs.

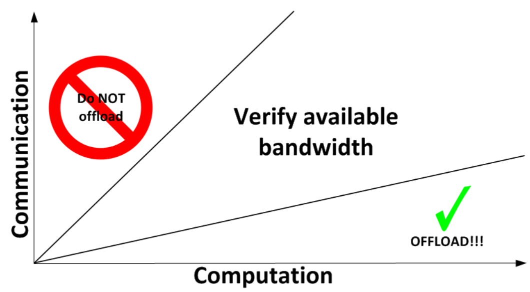


Figure 4.1: Trade off between computation and communication

A military research project presented in [73] considers the *M-ary* decision problem. Researchers analyze different parameters such as: 1) number of clients, 2) number of cloud resources, 3) communication bandwidth, and 4) job size. They also account for the computation capabilities and the current utilization of offload targets. A job is divided into n tasks and distributed among i cloud resources. A client obtains the utilization of the cloud resources by broadcasting a request to nearby targets, and these entities reply with their computation capacity (i.e., instructions per second) and their current utilization (i.e., length of the queue in seconds). This *utilization-aware* algorithm has been evaluated against three different alternatives: 1) local, 2) equal, and 3) architecture-aware. The *local* approach considers local execution of the entire job. The *equal* approach divides the job into equal size tasks and distributes them among multiple targets. The *architecture-aware* approach divides the job into different tasks, but distributes them

according to the computation capacities of the cloud resources. Results from experiments show that the *utilization-aware* algorithm increases its job completion time at a rate of 1.1 sec/byte, and the *equal* approach at a rate of 3.6 sec/byte, which means that the *utilization-aware* approach performs better as the size of the job increases. When the number of cloud resources increases, the *architecture-aware* algorithm performs similar to the *utilization-aware* algorithm, because of the small data transfers to different entities.

Another research project presented in [74], evaluates the performance of computation offloading decisions under different scenarios with multiple entities. The architecture of this system consists of: 1) a user, 2) a mobile service provider, 3) a data provider, and 4) a cloud resource. The framework also consists of six different modules that collaborate in the decision making process: 1) request/response handler, 2) context manager, 3) profiler, 4) execution planner, 5) service execution engine, and 6) offloading decision maker. The *request/response handler* is the module in charge of identifying the requests from clients and mapping them to the appropriate method. The *profiler* analyzes the operations (e.g. CPU cycles, memory size), identifies dependencies, and considers possible interactions with local resources, in order to evaluate the feasibility of computation offloading. The *context manager* is in charge of collecting current information about the user, like: 1) location, 2) mobile device's specifications, 3) current utilization of the resources, and 4) preferences. It also keeps track of all the available cloud resources (e.g. clouds, cloudlets, nearby mobile devices), and the available network connections (e.g. WiFi, Bluetooth). The *execution planner* generates several plans of execution based on the available information from the *context manager*. This module starts considering only local execution, but if the conditions are appropriate for offloading, then it also considers remote execution. The framework has a feature called *response forwarding*, in which the cloud resource can directly send the resulting output to the user. The *execution manager* is the module that executes the computation job on the mobile device, the cloud resource, or both; it depends on the selection made by the *offloading decision maker* module. The *offloading decision maker* module selects the best resource provider and determines which execution plan should be used. The system may choose between different

cloud resources (e.g. public clouds, cloudlets, or other mobile devices); it considers the current network conditions and the status of the resources. From Figure 4.2, researchers have identified six possible execution plans: ***Plan 1*** - executing the job locally with local input data, ***Plan 2*** - sending the input data from the client to the cloud through the mobile service provider, executing the job on the cloud, and getting the output data through the mobile service provider, ***Plan 3*** - sending the input data from the client to the cloud through the mobile service provider, executing the job on the cloud, and getting the output data directly from the cloud resource, ***Plan 4*** - executing the job locally, but fetching the input data from a data provider through the mobile service provider, ***Plan 5*** - executing the job on the cloud, but fetching the input data from a data provider, and getting the output data through the mobile service provider to the client, and ***Plan 6*** - executing the job on the cloud, but fetching the input data from a data provider, and getting the output data directly from the cloud resource to the client.

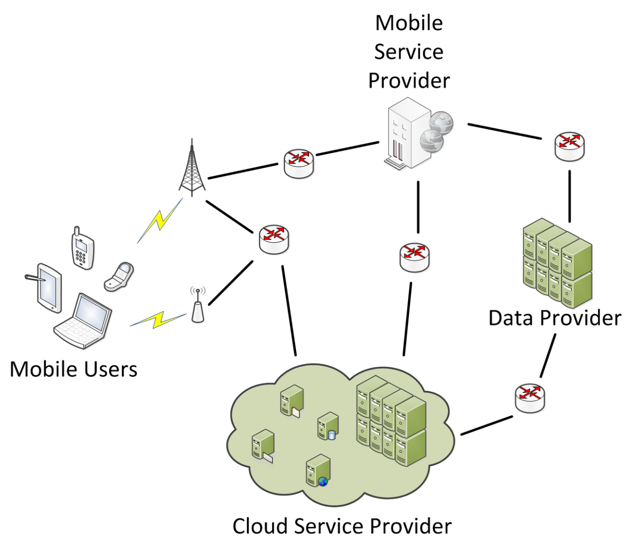


Figure 4.2: Cloud-assisted architecture

A scheme called *Follow-Me-Provider* has been proposed in [74], in which a user sends a request to a cloudlet, then the cloudlet performs the computation, and the output data is uploaded to the cloud. In case the user has moved, the mobile device can download the results from another cloudlet that may be close to it. Subsequent requests may also be serviced by other entities

and the resulting output would be available through other cloudlets. In order to make good offloading decisions, the framework takes into account components such as: memory space, amount of energy needed to execute a task, computational speed of the cloud resources, the available bandwidth, the input data size and output data size that are exchanged between the entities, and the energy needed for transmission. Since the data may travel through different links, the decision making analysis studies the location of the input data before the computation and the path through which the output data may be transmitted. To have a plan with the smallest response time, the framework also considers the resource consumption, the available local and remote resources, the feasible offloading plans, and the user preferences. From several image processing experiments, under different network connections (e.g. fast WiFi, slow WiFi, 3G), results show that *Plan 3* performs better when large amounts of data are transferred over slow communication links. However, *Plan 6* can achieve a performance of up to 6.1 times faster than executing locally over a high speed WiFi connection. *Plan 2* is suitable for applications where a lot of data transfers are needed and the energy consumption is a concern.

With our analysis we augment the existing literature by considering other network parameters that affect the communication time such as: 1) network congestion, and 2) number of hops between a cloud client and a cloud resource. With the analysis, we show: 1) the relationship between the computation time with respect to the communication time, and 2) the relationship between the remote execution speed with respect to the local execution speed. With these ratios we can determine when to offload a computational job.

In the next subsection, the system model we use in our analysis is explained in detail. It considers the computation delay and communication delay that affect the completion time of a computation job. The analysis evaluates the conditions in which computation offloading can reduce execution time, and therefore provides insight into the application types and infrastructure (computing and network) scenarios where offloading is beneficial.

4.2 System model

The system to be studied, consists of a client device that generates computational jobs that can be executed locally or on one of $(M-1)$ cloud resources. These cloud resources are distributed spatially in a network at varying numbers of hops from the client. The estimated time to complete the execution of a computational job serves as the criteria used to decide where to offload each computational job generated by the client. The choice that provides the shortest completion time is the one to be considered. The time to complete a job, if it is offloaded, is the time to transmit the input and output data files through the network plus the time to execute all of the instructions of the job on the selected cloud resource. The model includes the computation delay and communication delay that form the state variables for the computation offloading system. Using these models, another model formulates the computation offloading decision difference; the difference in completion time for the choice made with actual system state information and the completion time for the choice made with estimated system state information.

4.2.1 Computation delay

Let ξ be the time to execute a computational job on a certain computer, C be the size of the computational job (instructions), and E be the execution speed of a computer (instructions/sec). The computation delay for the computational job executed on that computer is $\xi = \frac{C}{E}$.

4.2.2 Communication delay

The per-hop model of communication delay incorporates the generally accepted categories of delay incurred at each hop [75]: (i) processing delay, (ii) queueing delay, (iii) transmission delay, and (iv) propagation delay. Let ψ be the time to transmit a packet across a hop in the network, S be the size of the packet (bits), α be the processing delay at a hop in a network (sec), β be the queueing delay incurred at that hop (sec), γ be the rate of the transmission channel at that hop (bits/sec), l be the length of that hop (meters), and c be the speed of light (meters/sec). The per-hop communication delay for that packet is $\psi = \alpha + \beta + \frac{S}{\gamma} + \frac{l}{\frac{2}{3}c}$.

Let h be the number of hops along the path that a packet traverses from its source to its destination. Converting the variables above to functions of the hop number then transform the equation above, ignoring the negligible processing and propagation delays to characterize the end-to-end communication delay as $\psi = \sum_{j=1}^{h} \left(\beta(j) + \frac{S}{\gamma(j)} \right)$.

Consider the transmission of a file that is larger than the maximum allowable packet size. In this case the file is transformed into a packet train; several maximum sized packets followed by a packet that can be up to the maximum size allowed. As individual packets in the packet train are transmitted through several hops in the network, the packet transmissions can occur in parallel. To model the communication delay considering the effect of the parallel packet transmissions, it is important to consider: (1) the time to transmit the entire file through the bottleneck transmission channel, and (2) the time to transmit the last packet in the packet train through each hop. Let F be the size of the file (bits), and M be the size of the last packet in the packet train. The communication delay for the entire file becomes,

$$\psi = \frac{F}{\min\{\gamma(j), \forall_j\}} + \sum_{j=1}^{h} \left(\beta(j) + \frac{M}{\gamma(j)} \right). \tag{4.1}$$

4.2.3 System state

The total system state can be partitioned into the computing state and the communication state. For the computing state, let C be the computational job size (instructions), I be the **p**rogram/**i**nput data size (bits), O be the **o**utput data size (bits), e be the execution speed for local device (instructions/sec), and $E(i)$ be the execution speed of each cloud resource, i, (instructions/sec). For the communication state, let $h(i)$ be the number of hops between the client and each cloud resource, $\beta(i,j)$ be the queueing delay at hop j toward cloud resource i, and $\gamma(i,j)$ be the transmission rate at hop j toward cloud resource i.

4.2.4 Computation offloading decision

The computation offloading decision algorithm we consider selects the choice with the smallest cost (i.e., job completion time). Let $x(i)$ be the cost of choice i, where $i = 0$ to $(M-1)$. We use $i = 0$ to refer to local execution and $1 \leq i \leq (M-1)$ to refer to execution on cloud resource i.

If $i = 0$, the cost is $x(0) = \frac{C}{e}$ and if $i \geq 1$, the cost is

$$x(i) = \frac{C}{E(i)} + \frac{(I+O)}{\min\{\gamma(i,j), \forall_j\}} + \sum_{j=1}^{h(i)} \left(\beta(i,j) + \frac{N}{\gamma(i,j)} \right). \tag{4.2}$$

Then, using the cost associated with each cloud resource, the optimal decision is $D = \arg\min_i \{x(i)\}$.

4.2.5 Useful ratios

The following two ratios are useful for our computation offloading analysis.

The computing-to-communication ratio (CCR) is the ratio of the computation time to the communication time. Using the symbols above,

$$CCR = \frac{\xi}{\psi}. \tag{4.3}$$

The remote-to-local ratio (RLR) is the ratio of the cloud resource execution speed to the local execution speed. Again, using the symbols above,

$$RLR = \frac{E}{e}. \tag{4.4}$$

4.3 When to offload computation?

To favor remote execution (or computation offloading) for reducing completion time, the following inequality must hold true:

$$\frac{C}{e} > \left[\frac{C}{E} + \frac{(I+O)}{\min\{\gamma(j), \forall_j\}} + \sum_{j=1}^{h}\left(\beta(j) + \frac{N}{\gamma(j)}\right)\right]. \tag{4.5}$$

Now the inequality is manipulated to derive useful insight into computation offloading system design. To ease manipulation of this inequality, let H represent the hop-by-hop network delay that is agnostic to the job size or the data size, F be all of the data to be transferred over the network (input and output), and Γ represent the transmission rate of the bottleneck link in the network. Now the RLR is isolated on the left hand side of the inequality.

$$\frac{C}{e} > \left[\frac{C}{E} + \left(\frac{F}{\Gamma} + H\right)\right] \tag{4.6}$$

$$C\left(\frac{1}{e} - \frac{1}{E}\right) > \left(\frac{F}{\Gamma} + H\right) \tag{4.7}$$

$$C\left(\frac{E}{e} - \frac{E}{E}\right) > E\left(\frac{F}{\Gamma} + H\right) \tag{4.8}$$

$$C\left(\frac{E}{e} - 1\right) > E\left(\frac{F}{\Gamma} + H\right) \tag{4.9}$$

$$\frac{E}{e} - 1 > \frac{E}{C}\left(\frac{F}{\Gamma} + H\right) \tag{4.10}$$

$$\frac{E}{e} - 1 > \frac{\left(\frac{F}{\Gamma} + H\right)}{\frac{C}{E}} \tag{4.11}$$

$$\frac{E}{e} > \frac{\left(\frac{F}{\Gamma} + H\right)}{\frac{C}{E}} + 1 \tag{4.12}$$

$$\frac{E}{e} > \frac{\psi}{\xi} + 1 \tag{4.13}$$

Table 4.1: RLR values required for offloading to be favorable for several CCR values.

CCR	**RLR**
1e-6	$\approx$ 1e6
1e-3	1001
0.01	101
0.1	11
1	2
1e3	1.001
1e6	1.000001

$$RLR > \frac{1}{CCR} + 1 \tag{4.14}$$

After isolating the RLR on the left hand side, the inequality is as shown in Eq. 4.14. That inequality shows there is a nearly inverse relationship between RLR and CCR. To visualize the implications of this inequality, the required RLR value has to be tabulated to make offloading favorable for various values of the CCR; see Table 4.1. A CCR of $1x10^{-3}$ requires cloud resources to be more than a thousand times faster than the local computing device. A CCR of $1x10^{-6}$ requires the cloud resources to be a million times faster! To obtain a sense of practical RLR values, several instructions per second (IPS) ratings of two embedded processors have been compiled. They represent the low end (Texas Instrument's MSP430) and the high end (Apple's A9) of the spectrum of the embedded processors deployed in handheld devices. To complete the RLR computation, the IPS ratings of a laptop-class processor (Intel's Celeron), desktop-class processor (Intel's Core i3), and a server-class processor (Intel's Xeon) were also compiled. See Table 4.2 for these IPS ratings.

Using the IPS ratings shown in Table 4.2, the RLR value for each pair of handheld and laptop/desktop/server class processor is computed. These RLR values are shown in Table 4.3. The RLR values range from 1.79 for an A9 processor offloading to a Celeron processor (CCR must be greater than 1.27 to offload) up to 8512.5 for an MSP 430 processor offloading to a Xeon

Table 4.2: Instructions per second ratings for several processors.

Processor	IPS
MSP430	$16x10^6$ [76]
A9	$3.6x10^9$ [77]
Celeron	$6.43x10^9$ [78]
Core i3	$36.8x10^9$ [79]
Xeon	$136.20x10^9$ [78]

Table 4.3: CPU RLR values

Local \ Remote	**Celeron**	**i3**	**Xeon**
MSP430	401.875	2300	8512.5
A9	1.78611	10.222	37.833

processor (CCR must be greater than $1.1x10^{-4}$ to offload).

4.4 Numerical experiments to find CCR

From the analysis above, it is observed that offloading is favorable when the inequality in Eq. 4.14 holds true. Now it is important to uncover the broad conditions (network and computing device parameters) in which offloading will reduce job completion time. For that, it may be useful to conduct numerical experiments to compute the CCR values for a wide range of parameters in the system model and check if the inequality in Eq. 4.14 holds true. Using the system model and the CCR in Eq. 4.3, this is the equation for CCR,

$$CCR = \frac{C\Gamma}{E(F + H\Gamma)}; \tag{4.15}$$

$$H = h\left(\beta + \frac{N}{\Gamma}\right); \tag{4.16}$$

$$N = 1518 * 8. \tag{4.17}$$

The following parameter values were used in the numerical experiments: $\Gamma \in \{ 1x10^3, 1x10^6, 1x10^9\}$, $F \in \{ 1x10^3, 1x10^6, 1x10^9, 1x10^{12}\}$, $C \in \{ 1x10^3, 100x10^3, 1x10^6, 1x10^9\}$, $E \in \{ 10x10^9, 100x10^9, 1x10^{12}, 10x10^{12}\}$, $h \in \{ 1, 2, 3, 4, 5\}$, and $\beta \in \{ 0, 0.001, 0.0001\}$.

We have chosen these values for h since we are studying cloudlet scenarios, where computation offloading occurs just few hops away from the mobile device. For the Γ parameter, we are considering these values since we are trying to visualize a wider spectrum of the different bottleneck transmission rates that may exist in cloudlet environments. For the β parameter, we consider the cases with no congestion, "light" congestion, and "moderate" congestion; in practice, congestion is very subjective since it depends on the quality of experience and the application requirements; in our case we are just considering congestion in the order of microseconds and milliseconds.

The CCR values computed were tabulated by evaluating Eq. 4.15 for all the possible combinations of the parameters above. On each table, there are sixteen CCR values that correspond to the four values of C and the four values of E, for a particular h, β, Γ, and F. Resulting in a total of 2,880 CCR values. The tables presented in this subsection, contain the cases where computation offloading can be beneficial. The complete set of tables with all the cases can be found in Appendix A.

In order to get a better insight of the conditions under which computation offloading becomes beneficial to mobile devices, we have proposed a CCR/RLR color scale shown in Figure 4.3. This color scale illustrates a wide spectrum of the processors' speeds. The *MSP430* represents the lower end of the spectrum (e.g. slowest processor) and the *A9* the higher end (e.g. fastest processor). RLR values increase from right to left and CCR values increase from left to right.

If the *MSP430* is used in a mobile device, RLR values to the left of that processor (i.e., values in red) will not benefit from computation offloading, however RLR values to the right of that processor (i.e., values in blue, yellow, dark green, or light green) will benefit from computation

offloading. If the *A9* is used in a mobile device, RLR values to the left of that processor (i.e., values in red, blue, yellow, or dark green) will not benefit from computation offloading, however RLR values to the right of that processor (i.e., light green) will benefit from computation offloading.

Red entries in the following tables indicate that neither the *MSP430* nor the *A9* benefit from computation offloading. Light green entries indicate that both *MSP430* and *A9* benefit from computation offloading. Values in blue, yellow or dark green indicate that only the *MSP430* benefit from computation offloading.

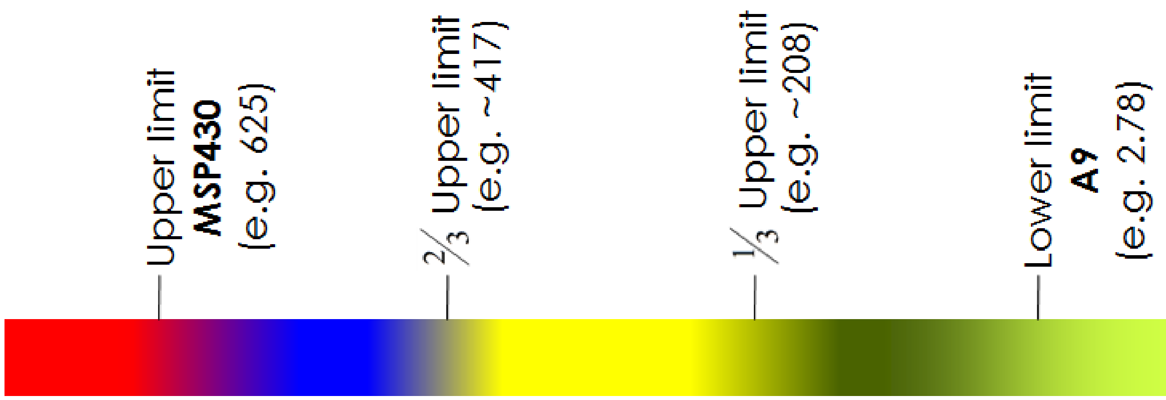


Figure 4.3: CCR/RLR color scale

Table 4.4: CCR values for: h = 1, β = 0, Γ = $1\text{x}10^3$, F = $1\text{x}10^3$

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1\text{x}10^3$	$4.96\text{x}10^{-09}$	$4.96\text{x}10^{-10}$	$4.96\text{x}10^{-11}$	$4.96\text{x}10^{-12}$
$100\text{x}10^3$	$4.96\text{x}10^{-07}$	$4.96\text{x}10^{-08}$	$4.96\text{x}10^{-09}$	$4.96\text{x}10^{-10}$
$1\text{x}10^6$	$4.96\text{x}10^{-06}$	$4.96\text{x}10^{-07}$	$4.96\text{x}10^{-08}$	$4.96\text{x}10^{-09}$
$1\text{x}10^9$	$4.96\text{x}10^{-03}$	$4.96\text{x}10^{-04}$	$4.96\text{x}10^{-05}$	$4.96\text{x}10^{-06}$

Table 4.5: CCR values for: h = 2, β = 0, Γ = $1\text{x}10^3$, F = $1\text{x}10^3$

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1\text{x}10^3$	$3.10\text{x}10^{-09}$	$3.10\text{x}10^{-10}$	$3.10\text{x}10^{-11}$	$3.10\text{x}10^{-12}$
$100\text{x}10^3$	$3.10\text{x}10^{-07}$	$3.10\text{x}10^{-08}$	$3.10\text{x}10^{-09}$	$3.10\text{x}10^{-10}$
$1\text{x}10^6$	$3.10\text{x}10^{-06}$	$3.10\text{x}10^{-07}$	$3.10\text{x}10^{-08}$	$3.10\text{x}10^{-09}$
$1\text{x}10^9$	$3.10\text{x}10^{-03}$	$3.10\text{x}10^{-04}$	$3.10\text{x}10^{-05}$	$3.10\text{x}10^{-06}$

Table 4.6: CCR values for: h = 3, β = 0, Γ = $1\text{x}10^3$, F = $1\text{x}10^3$

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1\text{x}10^3$	$2.25\text{x}10^{-09}$	$2.25\text{x}10^{-10}$	$2.25\text{x}10^{-11}$	$2.25\text{x}10^{-12}$
$100\text{x}10^3$	$2.25\text{x}10^{-07}$	$2.25\text{x}10^{-08}$	$2.25\text{x}10^{-09}$	$2.25\text{x}10^{-10}$
$1\text{x}10^6$	$2.25\text{x}10^{-06}$	$2.25\text{x}10^{-07}$	$2.25\text{x}10^{-08}$	$2.25\text{x}10^{-09}$
$1\text{x}10^9$	$2.25\text{x}10^{-03}$	$2.25\text{x}10^{-04}$	$2.25\text{x}10^{-05}$	$2.25\text{x}10^{-06}$

Table 4.7: CCR values for: h = 4, β = 0, Γ = $1\text{x}10^3$, F = $1\text{x}10^3$

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1\text{x}10^3$	$1.77\text{x}10^{-09}$	$1.77\text{x}10^{-10}$	$1.77\text{x}10^{-11}$	$1.77\text{x}10^{-12}$
$100\text{x}10^3$	$1.77\text{x}10^{-07}$	$1.77\text{x}10^{-08}$	$1.77\text{x}10^{-09}$	$1.77\text{x}10^{-10}$
$1\text{x}10^6$	$1.77\text{x}10^{-06}$	$1.77\text{x}10^{-07}$	$1.77\text{x}10^{-08}$	$1.77\text{x}10^{-09}$
$1\text{x}10^9$	$1.77\text{x}10^{-03}$	$1.77\text{x}10^{-04}$	$1.77\text{x}10^{-05}$	$1.77\text{x}10^{-06}$

Table 4.8: CCR values for: h = 1, β = 0.0001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.96x10^{-09}	4.96x10^{-10}	4.96x10^{-11}	4.96x10^{-12}
100x10^3	4.96x10^{-07}	4.96x10^{-08}	4.96x10^{-09}	4.96x10^{-10}
1x10^6	4.96x10^{-06}	4.96x10^{-07}	4.96x10^{-08}	4.96x10^{-09}
1x10^9	4.96x10^{-03}	4.96x10^{-04}	4.96x10^{-05}	4.96x10^{-06}

Table 4.9: CCR values for: h = 2, β = 0.0001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	3.10x10^{-09}	3.10x10^{-10}	3.10x10^{-11}	3.10x10^{-12}
100x10^3	3.10x10^{-07}	3.10x10^{-08}	3.10x10^{-09}	3.10x10^{-10}
1x10^6	3.10x10^{-06}	3.10x10^{-07}	3.10x10^{-08}	3.10x10^{-09}
1x10^9	3.10x10^{-03}	3.10x10^{-04}	3.10x10^{-05}	3.10x10^{-06}

Table 4.10: CCR values for: h = 3, β = 0.0001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.25x10^{-09}	2.25x10^{-10}	2.25x10^{-11}	2.25x10^{-12}
100x10^3	2.25x10^{-07}	2.25x10^{-08}	2.25x10^{-09}	2.25x10^{-10}
1x10^6	2.25x10^{-06}	2.25x10^{-07}	2.25x10^{-08}	2.25x10^{-09}
1x10^9	2.25x10^{-03}	2.25x10^{-04}	2.25x10^{-05}	2.25x10^{-06}

Table 4.11: CCR values for: h = 4, β = 0.0001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.77x10^{-09}	1.77x10^{-10}	1.77x10^{-11}	1.77x10^{-12}
100x10^3	1.77x10^{-07}	1.77x10^{-08}	1.77x10^{-09}	1.77x10^{-10}
1x10^6	1.77x10^{-06}	1.77x10^{-07}	1.77x10^{-08}	1.77x10^{-09}
1x10^9	1.77x10^{-03}	1.77x10^{-04}	1.77x10^{-05}	1.77x10^{-06}

Table 4.12: CCR values for: h = 1, β = 0.001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.96x10^{-09}	4.96x10^{-10}	4.96x10^{-11}	4.96x10^{-12}
100x10^3	4.96x10^{-07}	4.96x10^{-08}	4.96x10^{-09}	4.96x10^{-10}
1x10^6	4.96x10^{-06}	4.96x10^{-07}	4.96x10^{-08}	4.96x10^{-09}
1x10^9	4.96x10^{-03}	4.96x10^{-04}	4.96x10^{-05}	4.96x10^{-06}

Table 4.13: CCR values for: h = 2, β = 0.001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	3.10x10^{-09}	3.10x10^{-10}	3.10x10^{-11}	3.10x10^{-12}
100x10^3	3.10x10^{-07}	3.10x10^{-08}	3.10x10^{-09}	3.10x10^{-10}
1x10^6	3.10x10^{-06}	3.10x10^{-07}	3.10x10^{-08}	3.10x10^{-09}
1x10^9	3.10x10^{-03}	3.10x10^{-04}	3.10x10^{-05}	3.10x10^{-06}

Table 4.14: CCR values for: h = 3, β = 0.001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.25x10^{-09}	2.25x10^{-10}	2.25x10^{-11}	2.25x10^{-12}
100x10^3	2.25x10^{-07}	2.25x10^{-08}	2.25x10^{-09}	2.25x10^{-10}
1x10^6	2.25x10^{-06}	2.25x10^{-07}	2.25x10^{-08}	2.25x10^{-09}
1x10^9	2.25x10^{-03}	2.25x10^{-04}	2.25x10^{-05}	2.25x10^{-06}

Table 4.15: CCR values for: h = 4, β = 0.001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.77x10^{-09}	1.77x10^{-10}	1.77x10^{-11}	1.77x10^{-12}
100x10^3	1.77x10^{-07}	1.77x10^{-08}	1.77x10^{-09}	1.77x10^{-10}
1x10^6	1.77x10^{-06}	1.77x10^{-07}	1.77x10^{-08}	1.77x10^{-09}
1x10^9	1.77x10^{-03}	1.77x10^{-04}	1.77x10^{-05}	1.77x10^{-06}

Table 4.16: CCR values for: h = 1, β = 0, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9^** (~laptop-class core) RLR = (2.78, 625)	**100x10^9^** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12^** (~server-class core) RLR = (277.78, 62,500)	**10x10^12^** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.96x10^{-06}	4.96x10^{-07}	4.96x10^{-08}	4.96x10^{-09}
100x10^3	4.96x10^{-04}	4.96x10^{-05}	4.96x10^{-06}	4.96x10^{-07}
1x10^6	4.96x10^{-03}	4.96x10^{-04}	4.96x10^{-05}	4.96x10^{-06}
1x10^9	4.96x10^{+00}	4.96x10^{-01}	4.96x10^{-02}	4.96x10^{-03}

Table 4.17: CCR values for: h = 2, β = 0, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9^** (~laptop-class core) RLR = (2.78, 625)	**100x10^9^** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12^** (~server-class core) RLR = (277.78, 62,500)	**10x10^12^** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	3.10x10^{-06}	3.10x10^{-07}	3.10x10^{-08}	3.10x10^{-09}
100x10^3	3.10x10^{-04}	3.10x10^{-05}	3.10x10^{-06}	3.10x10^{-07}
1x10^6	3.10x10^{-03}	3.10x10^{-04}	3.10x10^{-05}	3.10x10^{-06}
1x10^9	3.10x10^{+00}	3.10x10^{-01}	3.10x10^{-02}	3.10x10^{-03}

Table 4.18: CCR values for: h = 3, β = 0, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9^** (~laptop-class core) RLR = (2.78, 625)	**100x10^9^** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12^** (~server-class core) RLR = (277.78, 62,500)	**10x10^12^** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.25x10^{-06}	2.25x10^{-07}	2.25x10^{-08}	2.25x10^{-09}
100x10^3	2.25x10^{-04}	2.25x10^{-05}	2.25x10^{-06}	2.25x10^{-07}
1x10^6	2.25x10^{-03}	2.25x10^{-04}	2.25x10^{-05}	2.25x10^{-06}
1x10^9	2.25x10^{+00}	2.25x10^{-01}	2.25x10^{-02}	2.25x10^{-03}

Table 4.19: CCR values for: h = 4, β = 0, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9^** (~laptop-class core) RLR = (2.78, 625)	**100x10^9^** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12^** (~server-class core) RLR = (277.78, 62,500)	**10x10^12^** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.77x10^{-06}	1.77x10^{-07}	1.77x10^{-08}	1.77x10^{-09}
100x10^3	1.77x10^{-04}	1.77x10^{-05}	1.77x10^{-06}	1.77x10^{-07}
1x10^6	1.77x10^{-03}	1.77x10^{-04}	1.77x10^{-05}	1.77x10^{-06}
1x10^9	1.77x10^{+00}	1.77x10^{-01}	1.77x10^{-02}	1.77x10^{-03}

Table 4.20: CCR values for: h = 5, β = 0, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.46x10^{-06}	1.46x10^{-07}	1.46x10^{-08}	1.46x10^{-09}
100x10^3	1.46x10^{-04}	1.46x10^{-05}	1.46x10^{-06}	1.46x10^{-07}
1x10^6	1.46x10^{-03}	1.46x10^{-04}	1.46x10^{-05}	1.46x10^{-06}
1x10^9	1.46x10^{+00}	1.46x10^{-01}	1.46x10^{-02}	1.46x10^{-03}

Table 4.21: CCR values for: h = 1, β = 0.0001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.94x10^{-06}	4.94x10^{-07}	4.94x10^{-08}	4.94x10^{-09}
100x10^3	4.94x10^{-04}	4.94x10^{-05}	4.94x10^{-06}	4.94x10^{-07}
1x10^6	4.94x10^{-03}	4.94x10^{-04}	4.94x10^{-05}	4.94x10^{-06}
1x10^9	4.94x10^{+00}	4.94x10^{-01}	4.94x10^{-02}	4.94x10^{-03}

Table 4.22: CCR values for: h = 2, β = 0.0001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	3.08x10^{-06}	3.08x10^{-07}	3.08x10^{-08}	3.08x10^{-09}
100x10^3	3.08x10^{-04}	3.08x10^{-05}	3.08x10^{-06}	3.08x10^{-07}
1x10^6	3.08x10^{-03}	3.08x10^{-04}	3.08x10^{-05}	3.08x10^{-06}
1x10^9	3.08x10^{+00}	3.08x10^{-01}	3.08x10^{-02}	3.08x10^{-03}

Table 4.23: CCR values for: h = 3, β = 0.0001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.24x10^{-06}	2.24x10^{-07}	2.24x10^{-08}	2.24x10^{-09}
100x10^3	2.24x10^{-04}	2.24x10^{-05}	2.24x10^{-06}	2.24x10^{-07}
1x10^6	2.24x10^{-03}	2.24x10^{-04}	2.24x10^{-05}	2.24x10^{-06}
1x10^9	2.24x10^{+00}	2.24x10^{-01}	2.24x10^{-02}	2.24x10^{-03}

Table 4.24: CCR values for: h = 4, β = 0.0001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.76x10^{-06}	1.76x10^{-07}	1.76x10^{-08}	1.76x10^{-09}
100x10^3	1.76x10^{-04}	1.76x10^{-05}	1.76x10^{-06}	1.76x10^{-07}
1x10^6	1.76x10^{-03}	1.76x10^{-04}	1.76x10^{-05}	1.76x10^{-06}
1x10^9	1.76x10^{+00}	1.76x10^{-01}	1.76x10^{-02}	1.76x10^{-03}

Table 4.25: CCR values for: h = 5, β = 0.0001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.44x10^{-06}	1.44x10^{-07}	1.44x10^{-08}	1.44x10^{-09}
100x10^3	1.44x10^{-04}	1.44x10^{-05}	1.44x10^{-06}	1.44x10^{-07}
1x10^6	1.44x10^{-03}	1.44x10^{-04}	1.44x10^{-05}	1.44x10^{-06}
1x10^9	1.44x10^{+00}	1.44x10^{-01}	1.44x10^{-02}	1.44x10^{-03}

Table 4.26: CCR values for: h = 1, β = 0.001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.73x10^{-06}	4.73x10^{-07}	4.73x10^{-08}	4.73x10^{-09}
100x10^3	4.73x10^{-04}	4.73x10^{-05}	4.73x10^{-06}	4.73x10^{-07}
1x10^6	4.73x10^{-03}	4.73x10^{-04}	4.73x10^{-05}	4.73x10^{-06}
1x10^9	4.73x10^{+00}	4.73x10^{-01}	4.73x10^{-02}	4.73x10^{-03}

Table 4.27: CCR values for: h = 2, β = 0.001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.92x10^{-06}	2.92x10^{-07}	2.92x10^{-08}	2.92x10^{-09}
100x10^3	2.92x10^{-04}	2.92x10^{-05}	2.92x10^{-06}	2.92x10^{-07}
1x10^6	2.92x10^{-03}	2.92x10^{-04}	2.92x10^{-05}	2.92x10^{-06}
1x10^9	2.92x10^{+00}	2.92x10^{-01}	2.92x10^{-02}	2.92x10^{-03}

Table 4.28: CCR values for: h = 3, β = 0.001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.11x10^{-06}	2.11x10^{-07}	2.11x10^{-08}	2.11x10^{-09}
100x10^3	2.11x10^{-04}	2.11x10^{-05}	2.11x10^{-06}	2.11x10^{-07}
1x10^6	2.11x10^{-03}	2.11x10^{-04}	2.11x10^{-05}	2.11x10^{-06}
1x10^9	2.11x10^{+00}	2.11x10^{-01}	2.11x10^{-02}	2.11x10^{-03}

Table 4.29: CCR values for: h = 4, β = 0.001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.65x10^{-06}	1.65x10^{-07}	1.65x10^{-08}	1.65x10^{-09}
100x10^3	1.65x10^{-04}	1.65x10^{-05}	1.65x10^{-06}	1.65x10^{-07}
1x10^6	1.65x10^{-03}	1.65x10^{-04}	1.65x10^{-05}	1.65x10^{-06}
1x10^9	1.65x10^{+00}	1.65x10^{-01}	1.65x10^{-02}	1.65x10^{-03}

Table 4.30: CCR values for: h = 5, β = 0.001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.36x10^{-06}	1.36x10^{-07}	1.36x10^{-08}	1.36x10^{-09}
100x10^3	1.36x10^{-04}	1.36x10^{-05}	1.36x10^{-06}	1.36x10^{-07}
1x10^6	1.36x10^{-03}	1.36x10^{-04}	1.36x10^{-05}	1.36x10^{-06}
1x10^9	1.36x10^{+00}	1.36x10^{-01}	1.36x10^{-02}	1.36x10^{-03}

Table 4.31: CCR values for: h = 1, β = 0, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.96x10^{-03}	4.96x10^{-04}	4.96x10^{-05}	4.96x10^{-06}
100x10^3	4.96x10^{-01}	4.96x10^{-02}	4.96x10^{-03}	4.96x10^{-04}
1x10^6	4.96x10^{+00}	4.96x10^{-01}	4.96x10^{-02}	4.96x10^{-03}
1x10^9	4.96x10^{+03}	4.96x10^{+02}	4.96x10^{+01}	4.96x10^{+00}

Table 4.32: CCR values for: h = 2, β = 0, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10³	3.10x10^{-03}	3.10x10^{-04}	3.10x10^{-05}	3.10x10^{-06}
100x10³	3.10x10^{-01}	3.10x10^{-02}	3.10x10^{-03}	3.10x10^{-04}
1x10⁶	3.10x10^{+00}	3.10x10^{-01}	3.10x10^{-02}	3.10x10^{-03}
1x10⁹	3.10x10^{+03}	3.10x10^{+02}	3.10x10^{+01}	3.10x10^{+00}

Table 4.33: CCR values for: h = 3, β = 0, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10³	2.25x10^{-03}	2.25x10^{-04}	2.25x10^{-05}	2.25x10^{-06}
100x10³	2.25x10^{-01}	2.25x10^{-02}	2.25x10^{-03}	2.25x10^{-04}
1x10⁶	2.25x10^{+00}	2.25x10^{-01}	2.25x10^{-02}	2.25x10^{-03}
1x10⁹	2.25x10^{+03}	2.25x10^{+02}	2.25x10^{+01}	2.25x10^{+00}

Table 4.34: CCR values for: h = 4, β = 0, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10³	1.77x10^{-03}	1.77x10^{-04}	1.77x10^{-05}	1.77x10^{-06}
100x10³	1.77x10^{-01}	1.77x10^{-02}	1.77x10^{-03}	1.77x10^{-04}
1x10⁶	1.77x10^{+00}	1.77x10^{-01}	1.77x10^{-02}	1.77x10^{-03}
1x10⁹	1.77x10^{+03}	1.77x10^{+02}	1.77x10^{+01}	1.77x10^{+00}

Table 4.35: CCR values for: h = 5, β = 0, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10³	1.46x10^{-03}	1.46x10^{-04}	1.46x10^{-05}	1.46x10^{-06}
100x10³	1.46x10^{-01}	1.46x10^{-02}	1.46x10^{-03}	1.46x10^{-04}
1x10⁶	1.46x10^{+00}	1.46x10^{-01}	1.46x10^{-02}	1.46x10^{-03}
1x10⁹	1.46x10^{+03}	1.46x10^{+02}	1.46x10^{+01}	1.46x10^{+00}

Table 4.36: CCR values for: h = 1, β = 0.0001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	8.32x10^{-04}	8.32x10^{-05}	8.32x10^{-06}	8.32x10^{-07}
100x10^3	8.32x10^{-02}	8.32x10^{-03}	8.32x10^{-04}	8.32x10^{-05}
1x10^6	8.32x10^{-01}	8.32x10^{-02}	8.32x10^{-03}	8.32x10^{-04}
1x10^9	8.32x10^{+02}	8.32x10^{+01}	8.32x10^{+00}	8.32x10^{-01}

Table 4.37: CCR values for: h = 2, β = 0.0001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.31x10^{-04}	4.31x10^{-05}	4.31x10^{-06}	4.31x10^{-07}
100x10^3	4.31x10^{-02}	4.31x10^{-03}	4.31x10^{-04}	4.31x10^{-05}
1x10^6	4.31x10^{-01}	4.31x10^{-02}	4.31x10^{-03}	4.31x10^{-04}
1x10^9	4.31x10^{+02}	4.31x10^{+01}	4.31x10^{+00}	4.31x10^{-01}

Table 4.38: CCR values for: h = 3, β = 0.0001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.90x10^{-04}	2.90x10^{-05}	2.90x10^{-06}	2.90x10^{-07}
100x10^3	2.90x10^{-02}	2.90x10^{-03}	2.90x10^{-04}	2.90x10^{-05}
1x10^6	2.90x10^{-01}	2.90x10^{-02}	2.90x10^{-03}	2.90x10^{-04}
1x10^9	2.90x10^{+02}	2.90x10^{+01}	2.90x10^{+00}	2.90x10^{-01}

Table 4.39: CCR values for: h = 4, β = 0.0001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.19x10^{-04}	2.19x10^{-05}	2.19x10^{-06}	2.19x10^{-07}
100x10^3	2.19x10^{-02}	2.19x10^{-03}	2.19x10^{-04}	2.19x10^{-05}
1x10^6	2.19x10^{-01}	2.19x10^{-02}	2.19x10^{-03}	2.19x10^{-04}
1x10^9	2.19x10^{+02}	2.19x10^{+01}	2.19x10^{+00}	2.19x10^{-01}

Table 4.40: CCR values for: h = 5, β = 0.0001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.76x10^-04	1.76x10^-05	1.76x10^-06	1.76x10^-07
100x10^3	1.76x10^-02	1.76x10^-03	1.76x10^-04	1.76x10^-05
1x10^6	1.76x10^-01	1.76x10^-02	1.76x10^-03	1.76x10^-04
1x10^9	1.76x10^+02	1.76x10^+01	1.76x10^+00	1.76x10^-01

Table 4.41: CCR values for: h = 1, β = 0.001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	9.80x10^-05	9.80x10^-06	9.80x10^-07	9.80x10^-08
100x10^3	9.80x10^-03	9.80x10^-04	9.80x10^-05	9.80x10^-06
1x10^6	9.80x10^-02	9.80x10^-03	9.80x10^-04	9.80x10^-05
1x10^9	9.80x10^+01	9.80x10^+00	9.80x10^-01	9.80x10^-02

Table 4.42: CCR values for: h = 2, β = 0.001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.92x10^-05	4.92x10^-06	4.92x10^-07	4.92x10^-08
100x10^3	4.92x10^-03	4.92x10^-04	4.92x10^-05	4.92x10^-06
1x10^6	4.92x10^-02	4.92x10^-03	4.92x10^-04	4.92x10^-05
1x10^9	4.92x10^+01	4.92x10^+00	4.92x10^-01	4.92x10^-02

Table 4.43: CCR values for: h = 3, β = 0.001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	3.28x10^-05	3.28x10^-06	3.28x10^-07	3.28x10^-08
100x10^3	3.28x10^-03	3.28x10^-04	3.28x10^-05	3.28x10^-06
1x10^6	3.28x10^-02	3.28x10^-03	3.28x10^-04	3.28x10^-05
1x10^9	3.28x10^+01	3.28x10^+00	3.28x10^-01	3.28x10^-02

Table 4.44: CCR values for: h = 4, β = 0.001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.47x10^{-05}	2.47x10^{-06}	2.47x10^{-07}	2.47x10^{-08}
100x10^3	2.47x10^{-03}	2.47x10^{-04}	2.47x10^{-05}	2.47x10^{-06}
1x10^6	2.47x10^{-02}	2.47x10^{-03}	2.47x10^{-04}	2.47x10^{-05}
1x10^9	2.47x10^{+01}	2.47x10^{+00}	2.47x10^{-01}	2.47x10^{-02}

Table 4.45: CCR values for: h = 5, β = 0.001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.97x10^{-05}	1.97x10^{-06}	1.97x10^{-07}	1.97x10^{-08}
100x10^3	1.97x10^{-03}	1.97x10^{-04}	1.97x10^{-05}	1.97x10^{-06}
1x10^6	1.97x10^{-02}	1.97x10^{-03}	1.97x10^{-04}	1.97x10^{-05}
1x10^9	1.97x10^{+01}	1.97x10^{+00}	1.97x10^{-01}	1.97x10^{-02}

Table 4.46: CCR values for: h = 1, β = 0, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.47: CCR values for: h = 2, β = 0, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.48: CCR values for: h = 3, β = 0, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9^** (~laptop-class core) RLR = (2.78, 625)	**100x10^9^** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12^** (~server-class core) RLR = (277.78, 62,500)	**10x10^12^** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table 4.49: CCR values for: h = 4, β = 0, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table 4.50: CCR values for: h = 5, β = 0, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table 4.51: CCR values for: h = 1, β = 0.0001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.52: CCR values for: h = 2, β = 0.0001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.53: CCR values for: h = 3, β = 0.0001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table 4.54: CCR values for: h = 4, β = 0.0001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table 4.55: CCR values for: h = 5, β = 0.0001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table 4.56: CCR values for: h = 1, β = 0.001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.57: CCR values for: h = 2, β = 0.001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.58: CCR values for: h = 3, β = 0.001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table 4.59: CCR values for: h = 4, β = 0.001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table 4.60: CCR values for: h = 5, β = 0.001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table 4.61: CCR values for: h = 1, β = 0, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
100x10^3	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}	1.25x10^{-06}
1x10^6	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}
1x10^9	1.25x10^{+01}	1.25x10^{+00}	1.25x10^{-01}	1.25x10^{-02}

Table 4.62: CCR values for: h = 2, β = 0, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
100x10^3	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}	1.25x10^{-06}
1x10^6	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}
1x10^9	1.25x10^{+01}	1.25x10^{+00}	1.25x10^{-01}	1.25x10^{-02}

Table 4.63: CCR values for: h = 3, β = 0, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
100x10^3	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}	1.24x10^{-06}
1x10^6	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}
1x10^9	1.24x10^{+01}	1.24x10^{+00}	1.24x10^{-01}	1.24x10^{-02}

Table 4.64: CCR values for: h = 4, β = 0, Γ = $1x10^9$, F = $1x10^6$

C \ E	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.24x10^{-05}$	$1.24x10^{-06}$	$1.24x10^{-07}$	$1.24x10^{-08}$
$100x10^3$	$1.24x10^{-03}$	$1.24x10^{-04}$	$1.24x10^{-05}$	$1.24x10^{-06}$
$1x10^6$	$1.24x10^{-02}$	$1.24x10^{-03}$	$1.24x10^{-04}$	$1.24x10^{-05}$
$1x10^9$	$1.24x10^{+01}$	$1.24x10^{+00}$	$1.24x10^{-01}$	$1.24x10^{-02}$

Table 4.65: CCR values for: h = 5, β = 0, Γ = $1x10^9$, F = $1x10^6$

C \ E	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.24x10^{-05}$	$1.24x10^{-06}$	$1.24x10^{-07}$	$1.24x10^{-08}$
$100x10^3$	$1.24x10^{-03}$	$1.24x10^{-04}$	$1.24x10^{-05}$	$1.24x10^{-06}$
$1x10^6$	$1.24x10^{-02}$	$1.24x10^{-03}$	$1.24x10^{-04}$	$1.24x10^{-05}$
$1x10^9$	$1.24x10^{+01}$	$1.24x10^{+00}$	$1.24x10^{-01}$	$1.24x10^{-02}$

Table 4.66: CCR values for: h = 1, β = 0.0001, Γ = $1x10^9$, F = $1x10^6$

C \ E	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.23x10^{-05}$	$1.23x10^{-06}$	$1.23x10^{-07}$	$1.23x10^{-08}$
$100x10^3$	$1.23x10^{-03}$	$1.23x10^{-04}$	$1.23x10^{-05}$	$1.23x10^{-06}$
$1x10^6$	$1.23x10^{-02}$	$1.23x10^{-03}$	$1.23x10^{-04}$	$1.23x10^{-05}$
$1x10^9$	$1.23x10^{+01}$	$1.23x10^{+00}$	$1.23x10^{-01}$	$1.23x10^{-02}$

Table 4.67: CCR values for: h = 2, β = 0.0001, Γ = $1x10^9$, F = $1x10^6$

C \ E	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.22x10^{-05}$	$1.22x10^{-06}$	$1.22x10^{-07}$	$1.22x10^{-08}$
$100x10^3$	$1.22x10^{-03}$	$1.22x10^{-04}$	$1.22x10^{-05}$	$1.22x10^{-06}$
$1x10^6$	$1.22x10^{-02}$	$1.22x10^{-03}$	$1.22x10^{-04}$	$1.22x10^{-05}$
$1x10^9$	$1.22x10^{+01}$	$1.22x10^{+00}$	$1.22x10^{-01}$	$1.22x10^{-02}$

Table 4.68: CCR values for: h = 3, β = 0.0001, Γ = 1x10^{9}, F = 1x10^{6}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.20x10^{-05}	1.20x10^{-06}	1.20x10^{-07}	1.20x10^{-08}
100x10^{3}	1.20x10^{-03}	1.20x10^{-04}	1.20x10^{-05}	1.20x10^{-06}
1x10^{6}	1.20x10^{-02}	1.20x10^{-03}	1.20x10^{-04}	1.20x10^{-05}
1x10^{9}	1.20x10^{+01}	1.20x10^{+00}	1.20x10^{-01}	1.20x10^{-02}

Table 4.69: CCR values for: h = 4, β = 0.0001, Γ = 1x10^{9}, F = 1x10^{6}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.18x10^{-05}	1.18x10^{-06}	1.18x10^{-07}	1.18x10^{-08}
100x10^{3}	1.18x10^{-03}	1.18x10^{-04}	1.18x10^{-05}	1.18x10^{-06}
1x10^{6}	1.18x10^{-02}	1.18x10^{-03}	1.18x10^{-04}	1.18x10^{-05}
1x10^{9}	1.18x10^{+01}	1.18x10^{+00}	1.18x10^{-01}	1.18x10^{-02}

Table 4.70: CCR values for: h = 5, β = 0.0001, Γ = 1x10^{9}, F = 1x10^{6}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.17x10^{-05}	1.17x10^{-06}	1.17x10^{-07}	1.17x10^{-08}
100x10^{3}	1.17x10^{-03}	1.17x10^{-04}	1.17x10^{-05}	1.17x10^{-06}
1x10^{6}	1.17x10^{-02}	1.17x10^{-03}	1.17x10^{-04}	1.17x10^{-05}
1x10^{9}	1.17x10^{+01}	1.17x10^{+00}	1.17x10^{-01}	1.17x10^{-02}

Table 4.71: CCR values for: h = 1, β = 0.001, Γ = 1x10^{9}, F = 1x10^{6}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.11x10^{-05}	1.11x10^{-06}	1.11x10^{-07}	1.11x10^{-08}
100x10^{3}	1.11x10^{-03}	1.11x10^{-04}	1.11x10^{-05}	1.11x10^{-06}
1x10^{6}	1.11x10^{-02}	1.11x10^{-03}	1.11x10^{-04}	1.11x10^{-05}
1x10^{9}	1.11x10^{+01}	1.11x10^{+00}	1.11x10^{-01}	1.11x10^{-02}

Table 4.72: CCR values for: h = 2, β = 0.001, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	9.98x10^{-06}	9.98x10^{-07}	9.98x10^{-08}	9.98x10^{-09}
100x10^3	9.98x10^{-04}	9.98x10^{-05}	9.98x10^{-06}	9.98x10^{-07}
1x10^6	9.98x10^{-03}	9.98x10^{-04}	9.98x10^{-05}	9.98x10^{-06}
1x10^9	9.98x10^{+00}	9.98x10^{-01}	9.98x10^{-02}	9.98x10^{-03}

Table 4.73: CCR values for: h = 3, β = 0.001, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	9.06x10^{-06}	9.06x10^{-07}	9.06x10^{-08}	9.06x10^{-09}
100x10^3	9.06x10^{-04}	9.06x10^{-05}	9.06x10^{-06}	9.06x10^{-07}
1x10^6	9.06x10^{-03}	9.06x10^{-04}	9.06x10^{-05}	9.06x10^{-06}
1x10^9	9.06x10^{+00}	9.06x10^{-01}	9.06x10^{-02}	9.06x10^{-03}

Table 4.74: CCR values for: h = 4, β = 0.001, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	8.30x10^{-06}	8.30x10^{-07}	8.30x10^{-08}	8.30x10^{-09}
100x10^3	8.30x10^{-04}	8.30x10^{-05}	8.30x10^{-06}	8.30x10^{-07}
1x10^6	8.30x10^{-03}	8.30x10^{-04}	8.30x10^{-05}	8.30x10^{-06}
1x10^9	8.30x10^{+00}	8.30x10^{-01}	8.30x10^{-02}	8.30x10^{-03}

Table 4.75: CCR values for: h = 5, β = 0.001, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	7.66x10^{-06}	7.66x10^{-07}	7.66x10^{-08}	7.66x10^{-09}
100x10^3	7.66x10^{-04}	7.66x10^{-05}	7.66x10^{-06}	7.66x10^{-07}
1x10^6	7.66x10^{-03}	7.66x10^{-04}	7.66x10^{-05}	7.66x10^{-06}
1x10^9	7.66x10^{+00}	7.66x10^{-01}	7.66x10^{-02}	7.66x10^{-03}

Table 4.76: CCR values for: h = 1, β = 0, Γ = $1x10^9$, F = $1x10^9$

E / C	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
$100x10^3$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$	$1.25x10^{-09}$
$1x10^6$	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$
$1x10^9$	$1.25x10^{-02}$	$1.25x10^{-03}$	$1.25x10^{-04}$	$1.25x10^{-05}$

Table 4.77: CCR values for: h = 2, β = 0, Γ = $1x10^9$, F = $1x10^9$

E / C	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
$100x10^3$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$	$1.25x10^{-09}$
$1x10^6$	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$
$1x10^9$	$1.25x10^{-02}$	$1.25x10^{-03}$	$1.25x10^{-04}$	$1.25x10^{-05}$

Table 4.78: CCR values for: h = 3, β = 0, Γ = $1x10^9$, F = $1x10^9$

E / C	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
$100x10^3$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$	$1.25x10^{-09}$
$1x10^6$	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$
$1x10^9$	$1.25x10^{-02}$	$1.25x10^{-03}$	$1.25x10^{-04}$	$1.25x10^{-05}$

Table 4.79: CCR values for: h = 4, β = 0, Γ = $1x10^9$, F = $1x10^9$

E / C	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
$100x10^3$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$	$1.25x10^{-09}$
$1x10^6$	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$
$1x10^9$	$1.25x10^{-02}$	$1.25x10^{-03}$	$1.25x10^{-04}$	$1.25x10^{-05}$

Table 4.80: CCR values for: h = 5, β = 0, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.81: CCR values for: h = 1, β = 0.0001, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.82: CCR values for: h = 2, β = 0.0001, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.83: CCR values for: h = 3, β = 0.0001, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.84: CCR values for: h = 4, β = 0.0001, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.85: CCR values for: h = 5, β = 0.0001, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.86: CCR values for: h = 1, β = 0.001, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.87: CCR values for: h = 2, β = 0.001, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table 4.88: CCR values for: h = 3, β = 0.001, Γ = 1x10^9, F = 1x10^9

E / C	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10³	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
100x10³	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$	$1.25x10^{-09}$
1x10⁶	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$
1x10⁹	$1.25x10^{-02}$	$1.25x10^{-03}$	$1.25x10^{-04}$	$1.25x10^{-05}$

Table 4.89: CCR values for: h = 4, β = 0.001, Γ = 1x10^9, F = 1x10^9

E / C	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10³	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
100x10³	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$	$1.25x10^{-09}$
1x10⁶	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$
1x10⁹	$1.25x10^{-02}$	$1.25x10^{-03}$	$1.25x10^{-04}$	$1.25x10^{-05}$

Table 4.90: CCR values for: h = 5, β = 0.001, Γ = 1x10^9, F = 1x10^9

E / C	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10³	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
100x10³	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$	$1.25x10^{-09}$
1x10⁶	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$
1x10⁹	$1.25x10^{-02}$	$1.25x10^{-03}$	$1.25x10^{-04}$	$1.25x10^{-05}$

4.4.1 Observations

The following observations are made from the CCR values and under which conditions the inequality in Eq. 4.14 holds true.

1. When the transmission rate (Γ) is the **same** order of magnitude as the file size (F), and the job size (C) is equal or greater than 1x10^9 instructions, computation offloading is beneficial. This holds true for:

   (a) Γ = 1x10^3 (1kbps), F = 1x10^3 (1KB), $\beta = \{0, 100x10^{-6}, 1x10^{-3}\}$, and h = $\{1, 2, 3, 4\}$

   (b) $\Gamma = \{1x10^6(1Mbps), 1x10^9(1Gbps)\}$, F = $\{1x10^6(1MB), 1x10^9(1GB)\}$, $\beta = \{0, 100x10^{-6}, 1x10^{-3}\}$, and h = $\{1, 2, 3, 4, 5\}$

2. When the transmission rate (Γ) is **one** order of magnitude greater than the file size (F), computation offloading is beneficial. This holds true for:

   (a) C = $\{1x10^6, 1x10^9\}$, Γ = 1x10^6 (1Mbps), F = 1x10^3 (1KB), $\beta = \{0, 100x10^{-6}, 1x10^{-3}\}$, and h = $\{1, 2, 3, 4\}$

   (b) C = $\{1x10^6, 1x10^9\}$, Γ = 1x10^9 (1Gbps), F = 1x10^6 (1MB), $\beta = \{0, 100x10^{-6}, 1x10^{-3}\}$, and h = $\{1, 2, 3, 4, 5\}$

   (c) C = $\{1x10^9\}$, Γ = 1x10^6 (1Mbps), F = 1x10^3 (1KB), $\beta = \{0, 100x10^{-6}, 1x10^{-3}\}$, and h = $\{5\}$

3. When the transmission rate (Γ) is **two** orders of magnitude greater than the file size (F), computation offloading is beneficial. This holds true for:

   (a) C = $\{1x10^3, 100x10^3, 1x10^6, 1x10^9\}$, Γ = 1x10^9 (1Gbps), F = 1x10^3 (1KB), $\beta = \{0\}$, and h = $\{1, 2, 3, 4\}$

   (b) C = $\{100x10^3, 1x10^6, 1x10^9\}$, Γ = 1x10^9 (1Gbps), F = 1x10^3 (1KB), $\beta = \{0\}$, and h = $\{5\}$

(c) C = $\{100x10^3, 1x10^6, 1x10^9\}$, Γ = 1x10^9 (1Gbps), F = 1x10^3 (1KB), $\beta = \{100x10^{-6}, 1x10^{-3}\}$, and h = $\{1, 2, 3, 4, 5\}$

4.5 TACC workload analysis to find CCR

Real computational workload data were obtained from the Texas Advanced Computing Center (TACC) in Austin, TX. The data was provided by the Director of High Performance Computing, Dr. Bill Barth. The data contains records of 78,176 computation jobs of 1,159 different applications. For each job, the data contains: 1) job ID, 2) application name, 3) job size, 4) bytes written, 5) bytes read, and 6) execution time.

Table 4.91 shows the average CCR values for the TACC data when there is no congestion, Table 4.92 shows the average CCR values for the TACC data when there is "light" congestion, and Table 4.93 shows the average CCR values for the TACC data when there is "moderate" congestion.

Table 4.91: CCR values for TACC trace data when there is no network congestion(i.e., $\beta = 0$ sec).

Γ \ h	**1**	**2**	**3**	**4**	**5**
1x10^3	5.57x10^{-02}	2.81x10^{-02}	1.89x10^{-02}	1.43x10^{-02}	1.16x10^{-02}
1x10^6	5.57x10^{+01}	2.81x10^{+01}	1.89x10^{+01}	1.43x10^{+01}	1.16x10^{+01}
1x10^9	5.57x10^{+04}	2.81x10^{+04}	1.89x10^{+04}	1.43x10^{+04}	1.16x10^{+04}

Table 4.92: CCR values for TACC trace data when there is "light" network congestion(i.e., $\beta =$ 100 μsec).

Γ \ h	**1**	**2**	**3**	**4**	**5**
1x10^3	5.57x10^{-02}	2.81x10^{-02}	1.89x10^{-02}	1.43x10^{-02}	1.16x10^{-02}
1x10^6	5.52x10^{+01}	2.79x10^{+01}	1.88x10^{+01}	1.42x10^{+01}	1.15x10^{+01}
1x10^9	6.53x10^{+03}	3.55x10^{+03}	2.55x10^{+03}	2.06x10^{+03}	1.76x10^{+03}

Table 4.93: CCR values for TACC trace data when there is "moderate" network congestion(i.e., $\beta = 1$ msec).

Γ \ h	**1**	**2**	**3**	**4**	**5**
1x10^3	5.57x10^{-02}	2.81x10^{-02}	1.89x10^{-02}	1.43x10^{-02}	1.16x10^{-02}
1x10^6	5.15x10^{+01}	2.60x10^{+01}	1.75x10^{+01}	1.33x10^{+01}	1.08x10^{+01}
1x10^9	1.22x10^{+03}	8.89x10^{+02}	7.76x10^{+02}	7.18x10^{+02}	6.83x10^{+02}

If the CCR values are translated into the RLR values required for offloading to reduce completion time, we get the RLR values in Tables 4.94, 4.95, and 4.96.

Table 4.94: RLR values for TACC workload data when there is no network congestion(i.e., $\beta = 0$ sec).

Γ \ h	**1**	**2**	**3**	**4**	**5**
1x10^3	18.96139	36.560243	53.807652	70.714239	87.290185
1x10^6	1.017961	1.03556	1.052808	1.069714	1.08629
1x10^9	1.000018	1.000036	1.000053	1.00007	1.000086

Table 4.95: RLR values for TACC workload data when there is "light" network congestion(i.e., $\beta = 100$ μsec).

Γ \ h	**1**	**2**	**3**	**4**	**5**
1x10^3	18.961536	36.56053	53.808074	70.71479	87.290861
1x10^6	1.018108	1.035847	1.053229	1.070265	1.086966
1x10^9	1.000153	1.000282	1.000391	1.000486	1.000569

If we consider a *MSP430* processor as the local device, in Tables 4.97, 4.98, and 4.99, we color an RLR value blue if the remote device must have at least a *Celeron* processor, green if the remote device must have at least an *i3* processor, and black if the remote device must have at least a *Xeon* processor.

In these tables, all the RLR values are blue, which means that a *Celeron* processor is enough

Table 4.96: RLR values for TACC workload data when there is "moderate" network congestion(i.e., $\beta = 1$ msec).

Γ \ h	**1**	**2**	**3**	**4**	**5**
1x10^3	18.962854	36.563112	53.81187	70.719752	87.296943
1x10^6	1.019424	1.038425	1.057015	1.07521	1.093021
1x10^9	1.000817	1.001125	1.001289	1.001393	1.001465

to satisfy the inequality that makes computation offloading beneficial.

Table 4.97: RLR values for TACC workload data when there is no network congestion(i.e., $\beta =$ 0 sec).

Γ \ h	**1**	**2**	**3**	**4**	**5**
1x10^3	18.96139	36.560243	53.807652	70.714239	87.290185
1x10^6	1.017961	1.03556	1.052808	1.069714	1.08629
1x10^9	1.000018	1.000036	1.000053	1.00007	1.000086

Table 4.98: RLR values for TACC workload data when there is "light" network congestion(i.e., $\beta = 100$ μsec).

Γ \ h	**1**	**2**	**3**	**4**	**5**
1x10^3	18.961536	36.56053	53.808074	70.71479	87.290861
1x10^6	1.018108	1.035847	1.053229	1.070265	1.086966
1x10^9	1.000153	1.000282	1.000391	1.000486	1.000569

If we consider an *A9* processor as the local device, in Tables 4.100, 4.101, and 4.102, we color an RLR value blue if the remote device must have at least a *Celeron* processor, green if the remote device must have at least an *i3* processor, and black if the remote device must have at least a *Xeon* processor. In these tables, we observe that for transmissions channels of 1Mbps and 1Gbps a *Celeron* processor is enough in the remote device in order for computation offloading to be beneficial. However, for a transmission channel of 1kbps, computation offloading is only

Table 4.99: RLR values for TACC workload data when there is "moderate" network congestion(i.e., β = 1 msec).

Γ \ h	1	2	3	4	5
1x10^3	18.962854	36.563112	53.81187	70.719752	87.296943
1x10^6	1.019424	1.038425	1.057015	1.07521	1.093021
1x10^9	1.000817	1.001125	1.001289	1.001393	1.001465

beneficial when the remote device is one or two hops away from the mobile device, and the remote device has at least a *Xeon* processor.

Table 4.100: RLR values for TACC workload data when there is no network congestion(i.e., β = 0 sec).

Γ \ h	1	2	3	4	5
1x10^3	18.96139	36.560243	53.807652	70.714239	87.290185
1x10^6	1.017961	1.03556	1.052808	1.069714	1.08629
1x10^9	1.000018	1.000036	1.000053	1.00007	1.000086

Table 4.101: RLR values for TACC workload data when there is "light" network congestion(i.e., β = 100 μsec).

Γ \ h	1	2	3	4	5
1x10^3	18.961536	36.56053	53.808074	70.71479	87.290861
1x10^6	1.018108	1.035847	1.053229	1.070265	1.086966
1x10^9	1.000153	1.000282	1.000391	1.000486	1.000569

4.5.1 Observation

From the preceding tables, we observe that the bottleneck transmission rate (Γ) and the number of hops between a cloud client and a cloud resource (h), are important factors that determine when computation offloading is beneficial. This can be noticed from Tables 4.100, 4.101, and

Table 4.102: RLR values for TACC workload data when there is "moderate" network congestion(i.e., $\beta = 1$ msec).

Γ \ h	**1**	**2**	**3**	**4**	**5**
1x10^3	18.962854	36.563112	53.81187	70.719752	87.296943
1x10^6	1.019424	1.038425	1.057015	1.07521	1.093021
1x10^9	1.000817	1.001125	1.001289	1.001393	1.001465

4.102, where for a transmission rate of 1kbps, the number of hops impacts the computation offloading decision. If the mobile device has an *A9* processor, it will only benefit from computation offloading when the cloud resource (e.g. cloudlet) is located no more than two hops away.

For applications like battlefield operations with:1) poor network connectivity or low bandwidth, and 2) mobile devices with more computational power (e.g. iPhone 6s), computation offloading to remote servers becomes less beneficial; however cloudlets become more attractive.

4.5.2 TACC Applications

The TACC data trace contains records of 78,176 computation jobs of 1,159 different applications. For each of these applications we have calculated the minimum, maximum, and average of the: 1) execution time, 2) input data size, and 3) output data size. The list of all TACC applications can be found in Appendix B.

Figure 4.4 shows a representation of the CCR for all of the applications contained in the TACC data trace. The *x-axis* represents computation time (i.e., execution time) and the *y-axis* represents the communication time (i.e., bytes read and bytes written over a 1Mbps connection). Each circle represents each of these applications, and the red line is the threshold for a CCR value of 0.01. Applications below the red line are the ones that benefit from computation offloading, since the computation time is greater than the communication time. Applications above the red line are the ones that do not benefit from computation offloading, since the communication time is greater than the computation time.

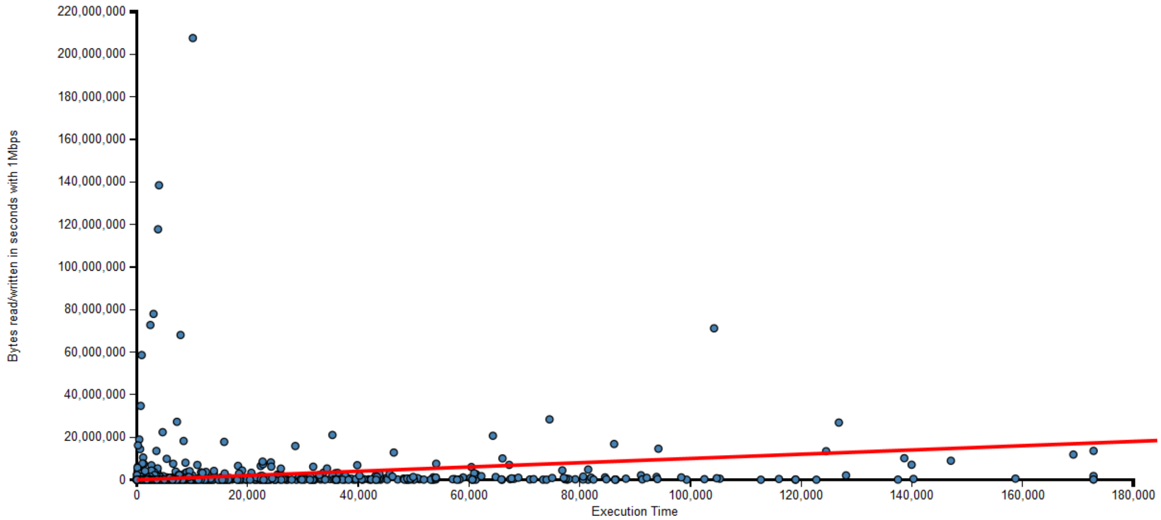


Figure 4.4: TACC applications

Descriptions of recognizable TACC application

vasp: +program for VAMP/VASP is a package for performing ab-initio quantum-mechanical molecular dynamics (MD) using pseudopotentials and a plane wave basis set. vasp: MPI parallel, charge density and wavefunction complex

namd2: NAMD is a parallel, object-oriented molecular dynamics code designed for high-performance simulation of large biomolecular systems. Simulation preparation and analysis is integrated into the visualization package VMD. Visit the NAMD website for complete information and documentation. Usually on small number of cores, non-smp gives better performance. However, when scales above 1000 cores, smp is usually faster. When you run smp, cpuaffinity is required. On Beagle the following flags are recommended when calling smp-based namd2 add this namd2 +ppn 11 +setcpuaffinity +pemap 1-11,13-23 +commap 0,12 to your aprun command and they are expected to give the best performance.

lmp_stampede: LAMMPS is a classical molecular dynamics simulator designed for parallel machines. It can model atomic, polymeric, biological, metallic, or mesoscale systems using a variety of force fields and boundary conditions and is easy to modify or extend. LAMMPS is a classical molecular dynamics code, and an acronym for Large-scale Atomic/Molecular Massively

Parallel Simulator. LAMMPS has potentials for solid-state materials (metals, semiconductors) and soft matter (biomolecules, polymers) and coarse-grained or mesoscopic systems. It can be used to model atoms or, more generically, as a parallel particle simulator at the atomic, meso, or continuum scale. LAMMPS runs on single processors or in parallel using message-passing techniques and a spatial-decomposition of the simulation domain. The code is designed to be easy to modify or extend with new functionality.

vasp_nlc: NCL is an interpreted language designed specifically for scientific data analysis and visualization. NLC is portable, robust, and freely available as binaries

cosmomc: Markov-Chain Monte-Carlo (MCMC) engine for exploring cosmological parameter space. Uses the likelihoods and data made available by the experimental collaborations to find the model that better represents the data.

charmm: CHARMM (Chemistry at HARvard Macromolecular Mechanics): is a versatile and widely used molecular simulation program with broad application to many-particle systems

nwchem: NWChem is an ab initio computational chemistry software package which also includes quantum chemical and molecular dynamics functionality.[1][2][3] It was designed to run on high-performance parallel supercomputers as well as conventional workstation clusters. It aims to be scalable both in its ability to treat large problems efficiently, and in its usage of available parallel computing resources.

fvcom: FVCOM is a prognostic, unstructured-grid, finite-volume, free-surface, 3-D primitive equation coastal ocean circulation model developed by UMASSD-WHOI joint efforts. FVCOM is written with Fortran 90 with MPI parallelization, and runs efficiently on single and multi-processor machines.

mdrun_mpi: The mdrun program is the main computational chemistry engine within GROMACS. Obviously, it performs molecular dynamics simulations, but it can also perform Brownian Dynamics, and Langevin Dynamics as well as Conjugate Gradient, L-BFGS, or Steepest Descents energy minimization. Normal Mode Analysis is another option.

siesta: SIESTA is both a method and its computer program implementation, to perform effi-

cient electronic structure calculations and ab initio molecular dynamics simulations of molecules and solids. SIESTA's efficiency stems from the use of strictly localized basis sets and from the implementation of linear-scaling algorithms which can be applied to suitable systems. A very important feature of the code is that its accuracy and cost can be tuned in a wide range, from quick exploratory calculations to highly accurate simulations matching the quality of other approaches, such as plane-wave and all-electron methods.

The TACC data trace visualizations and application descriptions presented in this dissertation were possible thanks to Herandy Vazquez, a team member of the *UTEP Army High Performance Computing Research Center Laboratory* (AHPCRC).

4.6 Numerical experiments to find F/C

From Eq. 4.14, we have identified the relationship between the RLR and CCR ratios. However, this inequality has a shortcoming. From Eq. 4.18, we can see that increasing E will not open more applications to benefit from computation offloading, since the E parameter is on both sides of the inequality. That is why we need to find another ratio where we can isolate the job characteristics from the infrastructure characteristics.

$$\frac{E}{e} > \frac{\left(\frac{F}{\Gamma} + H\right)}{\frac{C}{E}} + 1 \tag{4.18}$$

Considering an uncongested network so that H is negligible, from Eq. 4.7 we get:

$$C\left(\frac{1}{e} - \frac{1}{E}\right) > \frac{F}{\Gamma} \tag{4.19}$$

$$\left(\frac{1}{e} - \frac{1}{E}\right) > \frac{F}{\Gamma C} \tag{4.20}$$

$$\Gamma\left(\frac{1}{e} - \frac{1}{E}\right) > \frac{F}{C} \tag{4.21}$$

If we consider the following processor speeds, we can calculate the $\left(\frac{1}{e} - \frac{1}{E}\right)$ term as shown in Table 4.103:

Texas Instruments MSP430: 16x10^6 instructions / second

Apple A9: 3.6x10^9 instructions / second

Intel Celeron: 6.43x10^9 instructions / second

Intel i3: 36.8x10^9 instructions / second

Intel Xeon Phi: 136.2x10^9 instructions / second

Table 4.103: (1/e)-(1/E)

Remote / Local	**Celeron**	**i3**	**Xeon**
MSP430	6.2344x10^{-8}	6.2473x10^{-8}	6.2493x10^{-8}
A9	1.2226x10^{-10}	2.5060x10^{-10}	2.7044x10^{-10}

Assuming the TACC applications were executed on a single server-class core with an execution speed of 1x10^{12} (instr./sec), we derive the $\frac{F}{C}$ term for the ten most popular and recognizable TACC applications, as shown in Table 4.104. Assuming the TACC applications were executed on an 8-core server-class multiprocessor with an execution speed of 10x10^{12} (instr./sec), we derive the $\frac{F}{C}$ term for the ten most popular and recognizable TACC applications, as shown in Table 4.105.

Assuming a transmission channel of 1kbps and a MSP430 as a local device, we can identify some applications that may benefit from computation offloading as shown in Tables 4.106 and 4.107 for a single-core and an 8-core respectively. Values in green color represent the applications that make Eq. 4.21 true; values in red color are the applications that do not benefit from computation offloading. We can see that most of the applications benefit from computation offloading under this conditions.

Assuming a transmission channel of 1Mbps and a MSP430 as a local device, we can identify

Table 4.104: F/C - single server-class core

App. Name	min (bits/instr.)	avg (bits/instr.)	max (bits/instr.)
vasp	$0.00 \times 10^{+00}$	8.03×10^{-06}	6.85×10^{-04}
namd2	1.61×10^{-07}	4.45×10^{-05}	1.01×10^{-03}
lmp_stampede	1.36×10^{-07}	6.60×10^{-06}	9.53×10^{-04}
vasp_ncl	1.43×10^{-07}	6.56×10^{-06}	2.86×10^{-04}
cosmomc	2.34×10^{-07}	5.61×10^{-06}	4.84×10^{-04}
charmm	1.42×10^{-07}	9.32×10^{-06}	7.34×10^{-05}
nwchem	1.01×10^{-06}	7.06×10^{-05}	1.80×10^{-04}
fvcom	3.36×10^{-06}	2.17×10^{-04}	2.27×10^{-03}
mdrun_mpi	1.66×10^{-07}	8.18×10^{-06}	1.08×10^{-04}
siesta	1.47×10^{-07}	9.81×10^{-06}	5.29×10^{-05}

Table 4.106: F/C Ratio for: single server-class core, Γ = 1kbps, Local device = MSP430

App. Name	min (bits/instr.)	avg (bits/instr.)	max (bits/instr.)
vasp	$0.00 \times 10^{+00}$	8.03×10^{-06}	6.85×10^{-04}
namd2	1.61×10^{-07}	4.45×10^{-05}	1.01×10^{-03}
lmp_stampede	1.36×10^{-07}	6.60×10^{-06}	9.53×10^{-04}
vasp_ncl	1.43×10^{-07}	6.56×10^{-06}	2.86×10^{-04}
cosmomc	2.34×10^{-07}	5.61×10^{-06}	4.84×10^{-04}
charmm	1.42×10^{-07}	9.32×10^{-06}	7.34×10^{-05}
nwchem	1.01×10^{-06}	7.06×10^{-05}	1.80×10^{-04}
fvcom	3.36×10^{-06}	2.17×10^{-04}	2.27×10^{-03}
mdrun_mpi	1.66×10^{-07}	8.18×10^{-06}	1.08×10^{-04}
siesta	1.47×10^{-07}	9.81×10^{-06}	5.29×10^{-05}

Table 4.105: F/C - 8-core server-class

App. Name	min (bits/instr.)	avg (bits/instr.)	max (bits/instr.)
vasp	$0.00 \times 10^{+00}$	8.03×10^{-07}	6.85×10^{-05}
namd2	1.61×10^{-08}	4.45×10^{-06}	1.01×10^{-04}
lmp_stampede	1.36×10^{-08}	6.60×10^{-07}	9.53×10^{-05}
vasp_ncl	1.43×10^{-08}	6.56×10^{-07}	2.86×10^{-05}
cosmomc	2.34×10^{-08}	5.61×10^{-07}	4.84×10^{-05}
charmm	1.42×10^{-08}	9.32×10^{-07}	7.34×10^{-06}
nwchem	1.01×10^{-07}	7.06×10^{-06}	1.80×10^{-05}
fvcom	3.36×10^{-07}	2.17×10^{-05}	2.27×10^{-04}
mdrun_mpi	1.66×10^{-08}	8.18×10^{-07}	1.08×10^{-05}
siesta	1.47×10^{-08}	9.81×10^{-07}	5.29×10^{-06}

Table 4.107: F/C Ratio for: 8-core server-class, Γ = 1kbps, Local device = MSP430

App. Name	min (bits/instr.)	avg (bits/instr.)	max (bits/instr.)
vasp	$0.00 \times 10^{+00}$	8.03×10^{-07}	6.85×10^{-05}
namd2	1.61×10^{-08}	4.45×10^{-06}	1.01×10^{-04}
lmp_stampede	1.36×10^{-08}	6.60×10^{-07}	9.53×10^{-05}
vasp_ncl	1.43×10^{-08}	6.56×10^{-07}	2.86×10^{-05}
cosmomc	2.34×10^{-08}	5.61×10^{-07}	4.84×10^{-05}
charmm	1.42×10^{-08}	9.32×10^{-07}	7.34×10^{-06}
nwchem	1.01×10^{-07}	7.06×10^{-06}	1.80×10^{-05}
fvcom	3.36×10^{-07}	2.17×10^{-05}	2.27×10^{-04}
mdrun_mpi	1.66×10^{-08}	8.18×10^{-07}	1.08×10^{-05}
siesta	1.47×10^{-08}	9.81×10^{-07}	5.29×10^{-06}

some applications that may benefit from computation offloading as shown in Tables 4.108 and 4.109 for a single-core and an 8-core respectively. We can see that all of the applications benefit from computation offloading under this conditions.

Assuming a transmission channel of 1kbps and an A9 as a local device, we can identify some applications that may benefit from computation offloading as shown in Tables 4.110 and 4.111 for a single-core and an 8-core respectively. We can see that just few of the applications benefit from computation offloading under this conditions.

Assuming a transmission channel of 1Mbps and an A9 as a local device, we can identify some applications that may benefit from computation offloading as shown in Tables 4.112 and 4.113

Table 4.108: F/C Ratio for: single server-class core, Γ = 1Mbps, Local device = MSP430

App. Name	min (bits/instr.)	avg (bits/instr.)	max (bits/instr.)
vasp	0.00x10^{+00}	8.03x10^{-06}	6.85x10^{-04}
namd2	1.61x10^{-07}	4.45x10^{-05}	1.01x10^{-03}
lmp_stampede	1.36x10^{-07}	6.60x10^{-06}	9.53x10^{-04}
vasp_ncl	1.43x10^{-07}	6.56x10^{-06}	2.86x10^{-04}
cosmomc	2.34x10^{-07}	5.61x10^{-06}	4.84x10^{-04}
charmm	1.42x10^{-07}	9.32x10^{-06}	7.34x10^{-05}
nwchem	1.01x10^{-06}	7.06x10^{-05}	1.80x10^{-04}
fvcom	3.36x10^{-06}	2.17x10^{-04}	2.27x10^{-03}
mdrun_mpi	1.66x10^{-07}	8.18x10^{-06}	1.08x10^{-04}
siesta	1.47x10^{-07}	9.81x10^{-06}	5.29x10^{-05}

Table 4.109: F/C Ratio for: 8-core server-class, Γ = 1Mbps, Local device = MSP430

App. Name	min (bits/instr.)	avg (bits/instr.)	max (bits/instr.)
vasp	0.00x10^{+00}	8.03x10^{-07}	6.85x10^{-05}
namd2	1.61x10^{-08}	4.45x10^{-06}	1.01x10^{-04}
lmp_stampede	1.36x10^{-08}	6.60x10^{-07}	9.53x10^{-05}
vasp_ncl	1.43x10^{-08}	6.56x10^{-07}	2.86x10^{-05}
cosmomc	2.34x10^{-08}	5.61x10^{-07}	4.84x10^{-05}
charmm	1.42x10^{-08}	9.32x10^{-07}	7.34x10^{-06}
nwchem	1.01x10^{-07}	7.06x10^{-06}	1.80x10^{-05}
fvcom	3.36x10^{-07}	2.17x10^{-05}	2.27x10^{-04}
mdrun_mpi	1.66x10^{-08}	8.18x10^{-07}	1.08x10^{-05}
siesta	1.47x10^{-08}	9.81x10^{-07}	5.29x10^{-06}

Table 4.110: F/C Ratio for: single server-class core, Γ = 1kbps, Local device = A9

App. Name	min (bits/instr.)	avg (bits/instr.)	max (bits/instr.)
vasp	0.00x10^{+00}	8.03x10^{-06}	6.85x10^{-04}
namd2	1.61x10^{-07}	4.45x10^{-05}	1.01x10^{-03}
lmp_stampede	1.36x10^{-07}	6.60x10^{-06}	9.53x10^{-04}
vasp_ncl	1.43x10^{-07}	6.56x10^{-06}	2.86x10^{-04}
cosmomc	2.34x10^{-07}	5.61x10^{-06}	4.84x10^{-04}
charmm	1.42x10^{-07}	9.32x10^{-06}	7.34x10^{-05}
nwchem	1.01x10^{-06}	7.06x10^{-05}	1.80x10^{-04}
fvcom	3.36x10^{-06}	2.17x10^{-04}	2.27x10^{-03}
mdrun_mpi	1.66x10^{-07}	8.18x10^{-06}	1.08x10^{-04}
siesta	1.47x10^{-07}	9.81x10^{-06}	5.29x10^{-05}

Table 4.111: F/C Ratio for: 8-core server-class, Γ = 1kbps, Local device = A9

App. Name	min (bits/instr.)	avg (bits/instr.)	max (bits/instr.)
vasp	0.00x10^{+00}	8.03x10^{-07}	6.85x10^{-05}
namd2	1.61x10^{-08}	4.45x10^{-06}	1.01x10^{-04}
lmp_stampede	1.36x10^{-08}	6.60x10^{-07}	9.53x10^{-05}
vasp_ncl	1.43x10^{-08}	6.56x10^{-07}	2.86x10^{-05}
cosmomc	2.34x10^{-08}	5.61x10^{-07}	4.84x10^{-05}
charmm	1.42x10^{-08}	9.32x10^{-07}	7.34x10^{-06}
nwchem	1.01x10^{-07}	7.06x10^{-06}	1.80x10^{-05}
fvcom	3.36x10^{-07}	2.17x10^{-05}	2.27x10^{-04}
mdrun_mpi	1.66x10^{-08}	8.18x10^{-07}	1.08x10^{-05}
siesta	1.47x10^{-08}	9.81x10^{-07}	5.29x10^{-06}

for a single-core and an 8-core respectively. We can see that most of the applications benefit from computation offloading under this conditions.

Assuming a transmission channel of 1Gbps and an A9 as a local device, we can identify some applications that may benefit from computation offloading as shown in Tables 4.114 and 4.115 for a single-core and an 8-core respectively. We can see that all of the applications benefit from computation offloading under this conditions.

Table 4.112: F/C Ratio for: single server-class core, Γ = 1Mbps, Local device = A9

App. Name	min (bits/instr.)	avg (bits/instr.)	max (bits/instr.)
vasp	$0.00\text{x}10^{+00}$	$8.03\text{x}10^{-06}$	$6.85\text{x}10^{-04}$
namd2	$1.61\text{x}10^{-07}$	$4.45\text{x}10^{-05}$	$1.01\text{x}10^{-03}$
lmp_stampede	$1.36\text{x}10^{-07}$	$6.60\text{x}10^{-06}$	$9.53\text{x}10^{-04}$
vasp_ncl	$1.43\text{x}10^{-07}$	$6.56\text{x}10^{-06}$	$2.86\text{x}10^{-04}$
cosmomc	$2.34\text{x}10^{-07}$	$5.61\text{x}10^{-06}$	$4.84\text{x}10^{-04}$
charmm	$1.42\text{x}10^{-07}$	$9.32\text{x}10^{-06}$	$7.34\text{x}10^{-05}$
nwchem	$1.01\text{x}10^{-06}$	$7.06\text{x}10^{-05}$	$1.80\text{x}10^{-04}$
fvcom	$3.36\text{x}10^{-06}$	$2.17\text{x}10^{-04}$	$2.27\text{x}10^{-03}$
mdrun_mpi	$1.66\text{x}10^{-07}$	$8.18\text{x}10^{-06}$	$1.08\text{x}10^{-04}$
siesta	$1.47\text{x}10^{-07}$	$9.81\text{x}10^{-06}$	$5.29\text{x}10^{-05}$

Table 4.113: F/C Ratio for: 8-core server-class, Γ = 1Mbps, Local device = A9

App. Name	min (bits/instr.)	avg (bits/instr.)	max (bits/instr.)
vasp	$0.00\text{x}10^{+00}$	$8.03\text{x}10^{-07}$	$6.85\text{x}10^{-05}$
namd2	$1.61\text{x}10^{-08}$	$4.45\text{x}10^{-06}$	$1.01\text{x}10^{-04}$
lmp_stampede	$1.36\text{x}10^{-08}$	$6.60\text{x}10^{-07}$	$9.53\text{x}10^{-05}$
vasp_ncl	$1.43\text{x}10^{-08}$	$6.56\text{x}10^{-07}$	$2.86\text{x}10^{-05}$
cosmomc	$2.34\text{x}10^{-08}$	$5.61\text{x}10^{-07}$	$4.84\text{x}10^{-05}$
charmm	$1.42\text{x}10^{-08}$	$9.32\text{x}10^{-07}$	$7.34\text{x}10^{-06}$
nwchem	$1.01\text{x}10^{-07}$	$7.06\text{x}10^{-06}$	$1.80\text{x}10^{-05}$
fvcom	$3.36\text{x}10^{-07}$	$2.17\text{x}10^{-05}$	$2.27\text{x}10^{-04}$
mdrun_mpi	$1.66\text{x}10^{-08}$	$8.18\text{x}10^{-07}$	$1.08\text{x}10^{-05}$
siesta	$1.47\text{x}10^{-08}$	$9.81\text{x}10^{-07}$	$5.29\text{x}10^{-06}$

Table 4.114: F/C Ratio for: single server-class core, Γ = 1Gbps, Local device = A9

App. Name	min (bits/instr.)	avg (bits/instr.)	max (bits/instr.)
vasp	$0.00\text{x}10^{+00}$	$8.03\text{x}10^{-06}$	$6.85\text{x}10^{-04}$
namd2	$1.61\text{x}10^{-07}$	$4.45\text{x}10^{-05}$	$1.01\text{x}10^{-03}$
lmp_stampede	$1.36\text{x}10^{-07}$	$6.60\text{x}10^{-06}$	$9.53\text{x}10^{-04}$
vasp_ncl	$1.43\text{x}10^{-07}$	$6.56\text{x}10^{-06}$	$2.86\text{x}10^{-04}$
cosmomc	$2.34\text{x}10^{-07}$	$5.61\text{x}10^{-06}$	$4.84\text{x}10^{-04}$
charmm	$1.42\text{x}10^{-07}$	$9.32\text{x}10^{-06}$	$7.34\text{x}10^{-05}$
nwchem	$1.01\text{x}10^{-06}$	$7.06\text{x}10^{-05}$	$1.80\text{x}10^{-04}$
fvcom	$3.36\text{x}10^{-06}$	$2.17\text{x}10^{-04}$	$2.27\text{x}10^{-03}$
mdrun_mpi	$1.66\text{x}10^{-07}$	$8.18\text{x}10^{-06}$	$1.08\text{x}10^{-04}$
siesta	$1.47\text{x}10^{-07}$	$9.81\text{x}10^{-06}$	$5.29\text{x}10^{-05}$

Table 4.115: F/C Ratio for: 8-core server-class, Γ = 1Gbps, Local device = A9

App. Name	min (bits/instr.)	avg (bits/instr.)	max (bits/instr.)
vasp	$0.00\text{x}10^{+00}$	$8.03\text{x}10^{-07}$	$6.85\text{x}10^{-05}$
namd2	$1.61\text{x}10^{-08}$	$4.45\text{x}10^{-06}$	$1.01\text{x}10^{-04}$
lmp_stampede	$1.36\text{x}10^{-08}$	$6.60\text{x}10^{-07}$	$9.53\text{x}10^{-05}$
vasp_ncl	$1.43\text{x}10^{-08}$	$6.56\text{x}10^{-07}$	$2.86\text{x}10^{-05}$
cosmomc	$2.34\text{x}10^{-08}$	$5.61\text{x}10^{-07}$	$4.84\text{x}10^{-05}$
charmm	$1.42\text{x}10^{-08}$	$9.32\text{x}10^{-07}$	$7.34\text{x}10^{-06}$
nwchem	$1.01\text{x}10^{-07}$	$7.06\text{x}10^{-06}$	$1.80\text{x}10^{-05}$
fvcom	$3.36\text{x}10^{-07}$	$2.17\text{x}10^{-05}$	$2.27\text{x}10^{-04}$
mdrun_mpi	$1.66\text{x}10^{-08}$	$8.18\text{x}10^{-07}$	$1.08\text{x}10^{-05}$
siesta	$1.47\text{x}10^{-08}$	$9.81\text{x}10^{-07}$	$5.29\text{x}10^{-06}$

4.7 Summary

Computation offloading is beneficial when the local completion time is greater than the remote completion time. From our computation offloading decision analysis we have found the relationship between two important ratios: 1) Computing-to-Communication Ratio (CCR) and 2) Remote-to-Local Ratio (RLR). In order to benefit from computation offloading, the RLR value has to be greater than the inverse of the CCR value plus one. Also, we were able to derive the F/C inequality, which isolates the job characteristics from the infrastructure characteristics, and gives a much better insight of the conditions under which computation offloading becomes

beneficial for certain real world applications. By conducting numerical experiments with our model, we have derived a significant amount of CCR values, that provide insight into the scenarios under which applications may or may not benefit from computation offloading. The F/C inequality only contains the characteristics of the job (i.e., file size and job size), which helps to easily identify the conditions under which computation offloading is beneficial for a particular application.

Chapter 5: Effect of Estimation Error on Computation Offloading Decisions

5.1 Introduction

The analysis presented in the previous chapter considers the *binary decision*: 1) to offload and 2) not to offload. That decision considers the cloud a single resource to offload computation. However, there may exist several cloud resources that a mobile device can choose from. Therefore, we have an *M-ary decision problem.*

The computation offloading decision algorithm we consider selects the choice with the smallest cost. Let $x(i)$ be the cost of choice i; whereby $i = 0$ means that local computation is selected and $i \geq 1$ means that cloud resource i has been chosen, and let D be the decision.

If $i = 0$, the cost is $x(0) = \frac{C}{e}$ and if $i \geq 1$, the cost is

$$x(i) = \frac{C}{E(i)} + \frac{(I+O)}{\min\{\gamma(i,j), \forall_{i,j}\}} + \sum_{j=1}^{h(i)} \left(\beta(i,j) + \frac{M}{\gamma(i,j)} \right). \tag{5.1}$$

Using the cost of each choice, the optimal decision is $D = \arg\min_i \{x(i)\}$.

Since networking conditions vary over time (e.g. congestion, topology changes, available bandwidth), the computation offloading decision model must consider an error when estimating the communication time. Now, the optimal decision from the client's perspective will be the minimum completion time with an error in the estimation. Such error leads to a decision difference in the decision making process.

Let $\epsilon(i)$ be the error in the estimate of the communication cost for choice i, and $\hat{x}(i)$ be the estimated cost for choice i with the error incorporated. The estimated costs are $\hat{x}(i) = \xi(i) + \psi(i) + \epsilon(i)$ and the optimal decision with estimated communication costs is $\hat{D} = \arg\min_i \{\hat{x}(i)\}$.

Therefore the difference between the cost of the decision with and without error is

$$\Delta = x(\hat{D}) - x(D). \tag{5.2}$$

By studying the *M-ary* decision problem we will answer the following questions: 1) *How does*

the network delay estimate error affect job completion time? and 2) *How good does the network delay estimate have to be to make good computation offloading decisions?*. To help answer these questions, we employed two different methods: 1) Monte Carlo simulations to solve for the expected value of the decision difference, and 2) discrete event simulations (i.e., NS-3 simulator) to observe the decision difference.

5.2 Discrete-event Simulations

NS-3 [80] is an open source discrete-event network simulator used for research, development, and educational purposes. This project was funded by the National Science Foundation (NSF), the *Planete group* at INRIA Sophia Antipolis, the Georgia Institute of Technology, the University of Washington, and the U.S. Army Research Laboratory.

The system to be evaluated in this computation offloading project is shown in Figure 5.1. It consists on twelve different entities: 1) one cloud client, 2) four cloud resources, and 3) seven routing nodes.

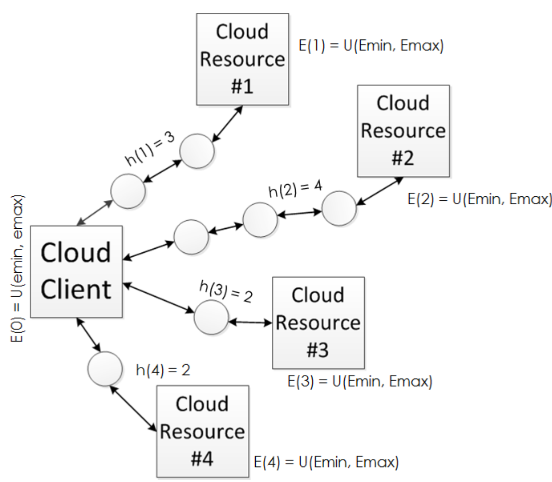


Figure 5.1: NS-3 Simulation Model

5.2.1 Simulation Model

Each entity has different behaviors depending on the tasks they perform. These behaviors determine the actions to be taken by each element of the system. The cloud client communicates

with the cloud resources through the routing nodes that exist in between. This communication follows a client to server communication pattern, where the client sends requests to the server and the server responds to such requests. For the case of computation offloading, the cloud client sends a job to a cloud resource, and the cloud resource responds with the output that results from the execution of that particular job. In the next subsection, the behavior(s) of each entity are described in detail.

The computation offloading decision system consists on two models: 1) the cloud client model and 2) the cloud resource model. Each model describes the behaviors of the cloud clients and the cloud resources that make up the system. In NS-3, they can be found as *"cloud-client"* and *"cloud-resource"*. The library *"comp-offloading-config"* contains all the shared data structures needed by the components of the system. It includes: a) job's structure, b) node's structure, c) computing scenarios, and d) queues.

Cloud Client

The cloud client has five behaviors. In NS-3, they can be found in the *cloud-client* model. The first behavior consists on generating computational jobs. This behavior is modeled by the method *GenerateJob()*. Each job has three properties: a) *job size* (C) (i.e. number of instructions needed to execute the job), b) *input data size* (I) (i.e. number of data bits to be processed), and c) *output data size* (O) (i.e. number of data bits produced after the job is completed). When a job is generated, the function *CraftJob()* is called to fill out the job's structure containing the properties mentioned above. Once a job is generated, the cloud client is responsible for deciding where to process the job (e.g. locally or remotely). In order to make a good decision, the cloud client starts by estimating the local computation cost and the remote computation costs. Since the model considers a M-ary decision (e.g. multiple cloud resources), the cloud client has to estimate a remote computation cost for each of the cloud resources. This estimation is carried out by the *Estimate()* function. The remote computation cost is divided in three main components: 1) time to send the input data from the cloud client to the cloud resource, 2) time

to process the job in the cloud resource, and 3) time to send the output data from the cloud resource to the cloud client. To estimate the remote completion time, the cloud client considers the three job's properties (i.e. C, I, O), the numbers of hops (i.e. segments between the cloud client and the cloud resource), the transmission rate, and the computation speed of the cloud resource. After estimating the local completion time and all the remote completion times, the cloud client computes the minimum completion time to determine if the job should be executed locally or remotely. If the minimum completion time equals the local decision cost, then the job is placed on the local queue, otherwise it is placed on the sending queue. In NS-3, this function is called *PlaceJob()*.

The second behavior of the cloud client is to verify if the processor is idle to execute a job locally. If the processor is idle and the local queue is not empty, then the cloud client removes a job from the head of the local queue and starts executing the job. This behavior is modeled in the *StartJobClient()* method.

The third behavior of the cloud client is to check if the sending queue has waiting jobs. If the sending queue is not empty and there is no other ongoing transmission (i.e. the transceiver is idle), then the cloud client removes the job from the head of the sending queue and starts the transmission of the input data to the cloud resource. This behavior is modeled in the *SendJobClient()* method. Figure 5.2 illustrates the first three cloud client behaviors that produce the outgoing traffic.

The fourth behavior of the cloud client is to receive completed jobs (e.g. output data) from the cloud resources. The cloud client receives packets that contain the data bits produced by the cloud resource after executing a job. A job is considered complete until all the output data packets are successfully received by the cloud client. This behavior is modeled in the *receivedPacketClient()* method.

The fifth cloud client's behavior is to report the completion time of a job. This occurs once the job has been completed locally (e.g. local resource) or the incoming output data of a particular job is fully received from a remote cloud resource. By recording the completion time, the cloud

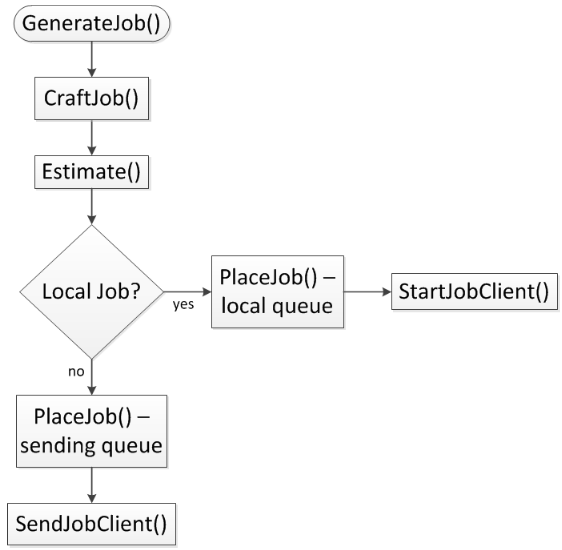


Figure 5.2: Cloud Client Model Behaviors

client can determine if the best choice was chosen or if there was a penalty for making such decision. This behavior is modeled in the *FinishJobClient()* method when the job is completed locally or in the *receivedPacketClient()* method when the job is executed remotely. Figure 5.3 illustrates the last two cloud client behaviors that model the incoming traffic.

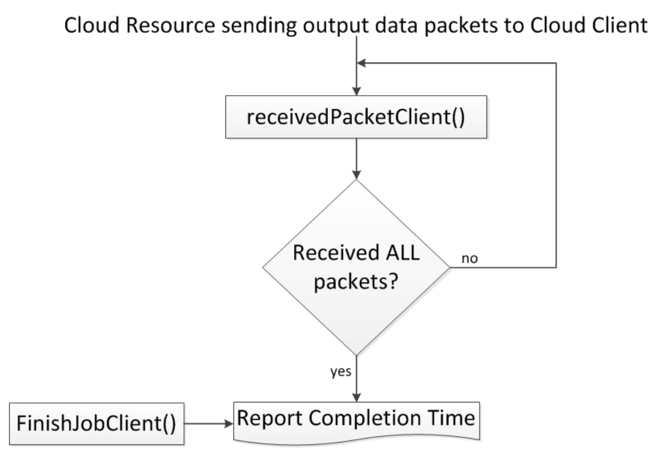


Figure 5.3: Cloud Client Model Behaviors

Cloud Resource

Each cloud resource has four behaviors. In NS-3, they can be found in the *cloud-resource* model. The first behavior consists on receiving input data from the cloud client. The input data is sent

from the cloud client to the cloud resource in packets. Once the job (i.e. input data) is fully received, then it is placed on the resource's processing queue. This behavior is modeled in the *receivedPacketResource()* method.

The second behavior of the cloud resource is to examine its processing queue. If the processing queue is not empty and there are available computing nodes, then the cloud resource removes the job from the head of its processing queue and starts executing the job. This behavior is modeled in the *StartJobResource()* method.

Once the job is completed, the third behavior is to place the output file (e.g. results) on the sending queue. These files are placed on the sending queue since there may be other ongoing transmissions. This behavior is modeled in the *FinishJobResource()* method.

The fourth behavior of the cloud resource is to check its sending queue. If the sending queue is not empty and the transceiver is available for transmission, then the cloud resource removes the job from the head of the queue and sends the output file to the cloud client. This behavior is modeled in the *SendJobResource()* method. Figure 5.4 illustrates the cloud resource behaviors.

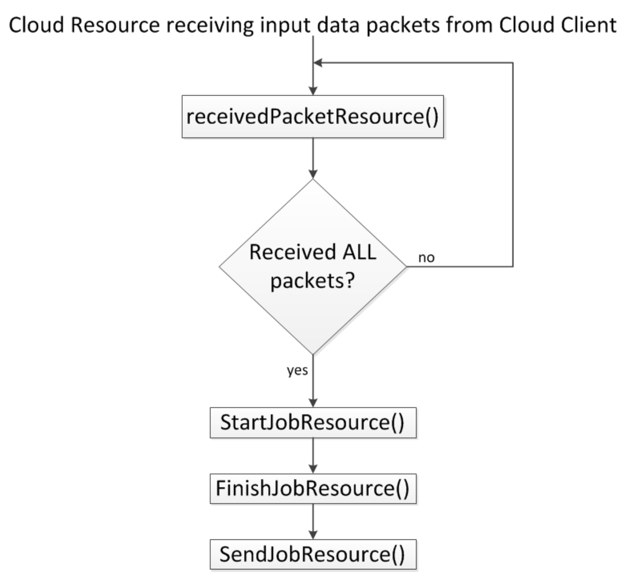


Figure 5.4: Cloud Resource Model Behaviors

Routing Nodes

The routing nodes that sit in between the cloud client and the cloud resources only have one behavior. They just forward incoming packets from one neighbor to the next one.

The complete flowchart of the offloading process is shown in Figure 5.5.

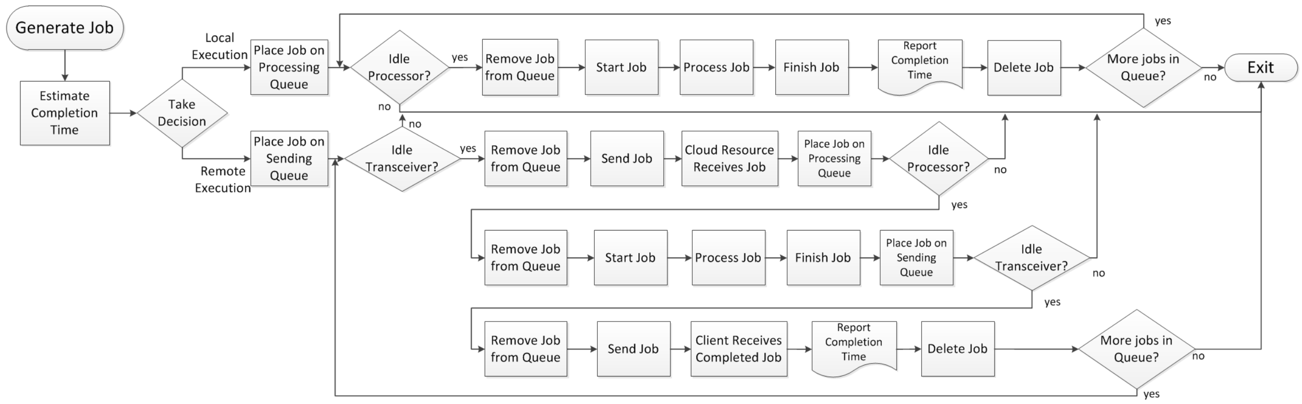
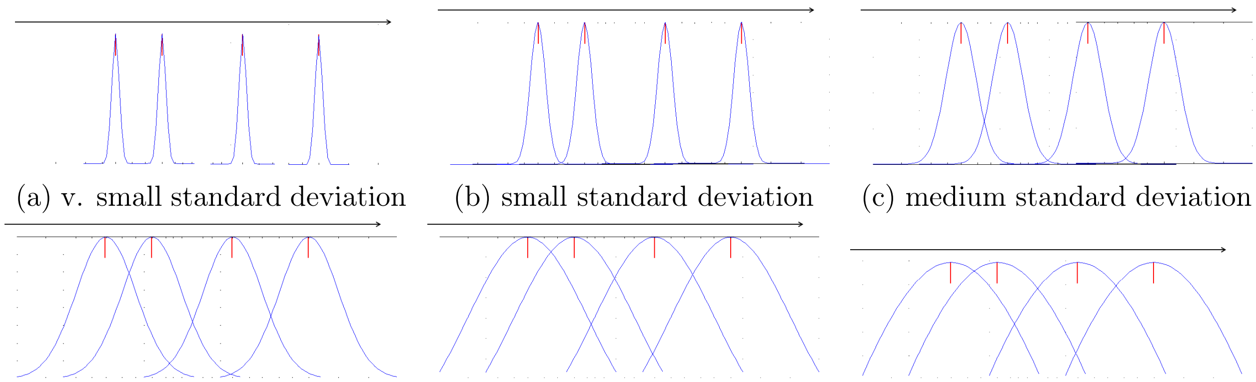


Figure 5.5: NS-3 Computation Offloading Process

5.3 Results

The network delay estimation error can be modeled from different probability distributions like: a) Uniform, b) Normal, c) Exponential, and d) Pareto. When considering an error from the Normal distribution with zero mean and a varying standard deviation, Figures 5.6a through 5.6f show how the decision difference gets affected as the standard deviation increases. In these figures, the red lines represent the completion time costs of the resources (local and remote) and the arrow line represents time. As the standard deviation of the error in the estimate of the communication cost increases, the curves start to overlap, which means an increase in the probability of confusing the different choices and thus making wrong decisions.

Figures 5.7a through 5.7f show the case where the difference between the completion time costs of the resources (local and remote) is very small, and how the decision difference starts getting affected with a small increase in the standard deviation. In this case, the network delay estimator has to be more accurate in order to distinguish the best choice if the computational job is subject to a critical completion time constraint. However, if the estimator is not completely

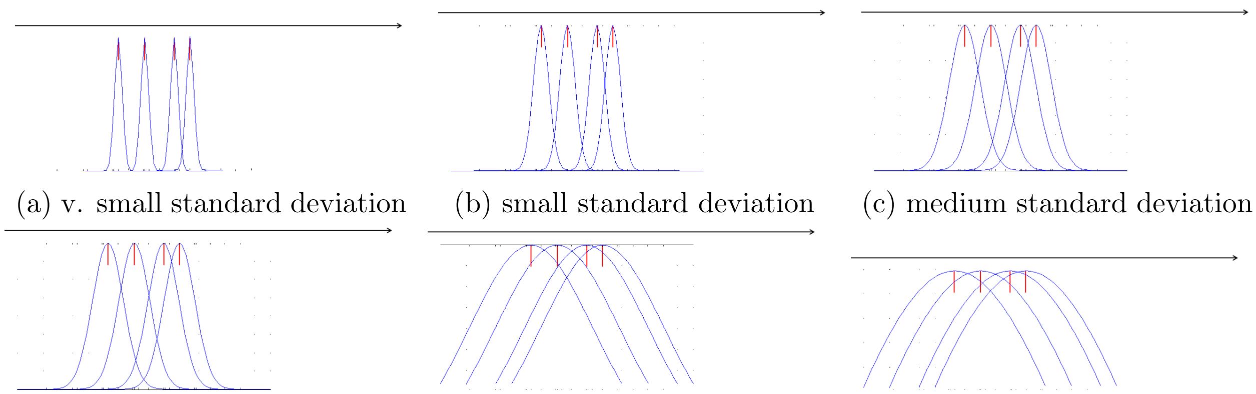


(a) v. small standard deviation (b) small standard deviation (c) medium standard deviation

(d) large standard deviation (e) v. large standard deviation (f) X large standard deviation

Figure 5.6: **Case 1:** When the completion time costs of the resources (local and remote) have a considerable difference between them. Fig. (a) through Fig. (f) show the effect of increasing the standard deviation of the error to the communication cost.

accurate, the decision difference is very small compared to the case where the choices have more difference in the completion time cost.



(a) v. small standard deviation (b) small standard deviation (c) medium standard deviation

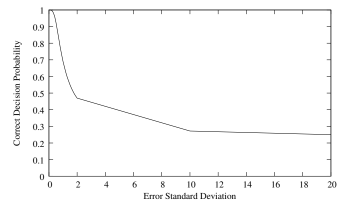
(d) large standard deviation (e) v. large standard deviation (f) X large standard deviation

Figure 5.7: **Case 2:** When the completion time costs of the resources (local and remote) have a tight difference between them. Fig. (a) through Fig. (f) show the effect of increasing the standard deviation of the error to the communication cost.

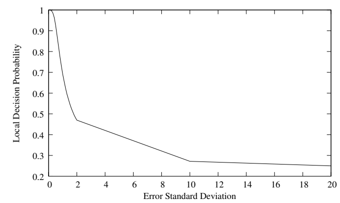
5.3.1 NS-3 Simulation

We conducted an extensive set of NS-3 simulation experiments to derive the average of the decision difference using our system model. We ran simulations using the Normal (or Gaussian) probability distribution with zero mean and different standard deviations to model the error of

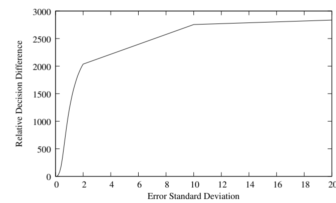
the estimate in the communication cost. We used CCR values of 1e^{-6}, 1e^{-3}, 1, and 1e^{3}. Figures 5.8 through 5.23 show the results for different CCR and RLR values, including: a) correct decision rate, b) local decision rate, c) relative decision difference, and d) decision difference, along with some zoomed versions of the decision difference.



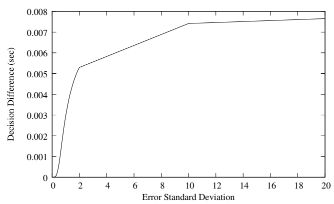
(a) Correct Decision Rate



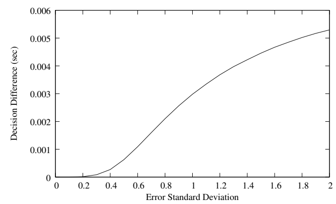
(b) Local Decision Rate



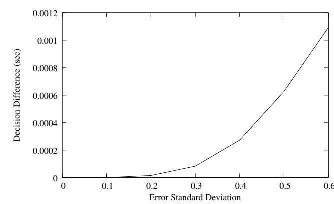
(c) Rel. Decision Diff.



(d) Decision Difference

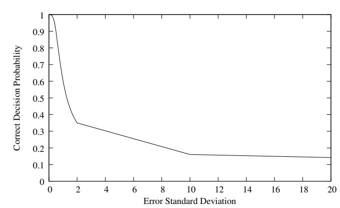


(e) Decision Diff. Zoomed

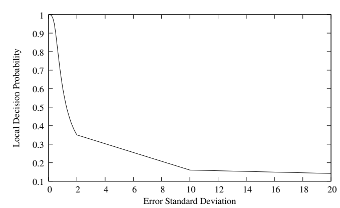


(f) Decision Diff. Zoomed

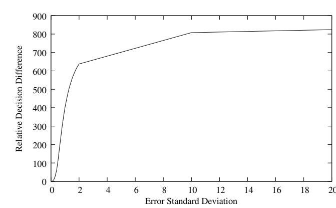
Figure 5.8: Plots for RLR = 2 and CCR = 1x10^{-3}



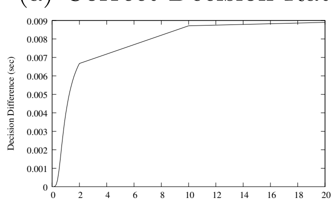
(a) Correct Decision Rate



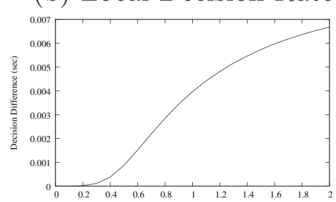
(b) Local Decision Rate



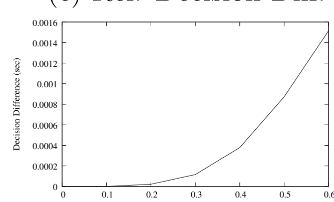
(c) Rel. Decision Diff.



(d) Decision Difference

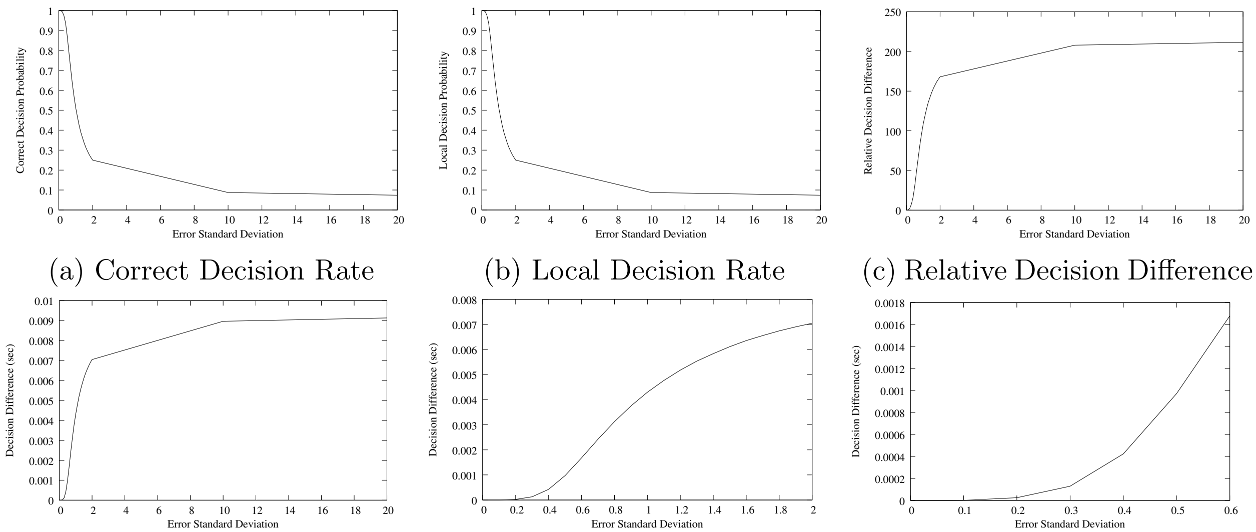


(e) Decision Diff. Zoomed



(f) Decision Diff. Zoomed

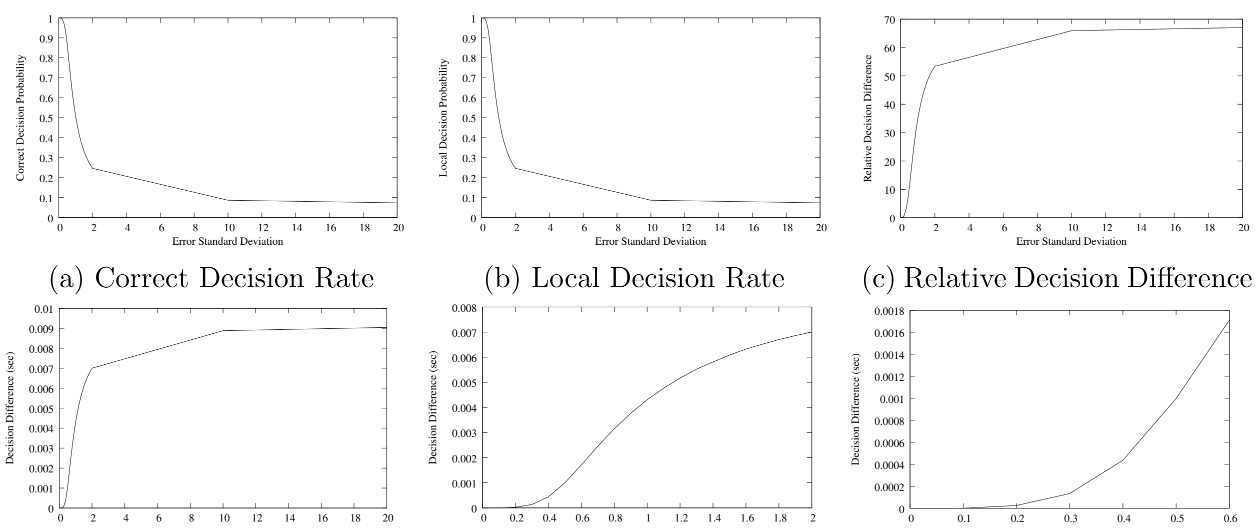
Figure 5.9: Plots for RLR = 8 and CCR = 1x10^{-3}



(a) Correct Decision Rate (b) Local Decision Rate (c) Relative Decision Difference

(d) Decision Difference (e) Decision Difference Zoomed (f) Decision Difference Zoomed

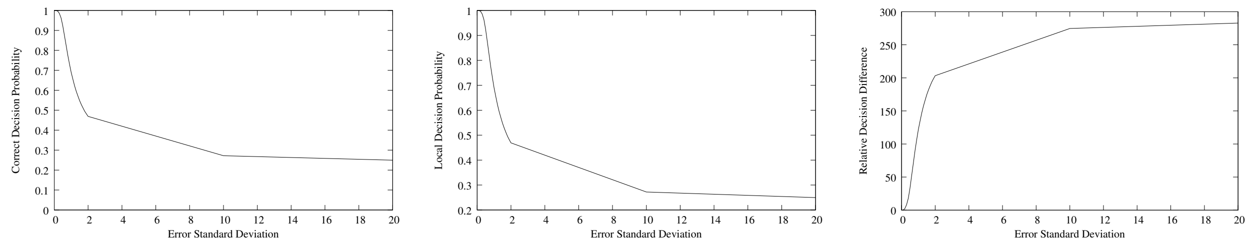
Figure 5.10: Plots for RLR = 32 and CCR = 1x10^{-3}



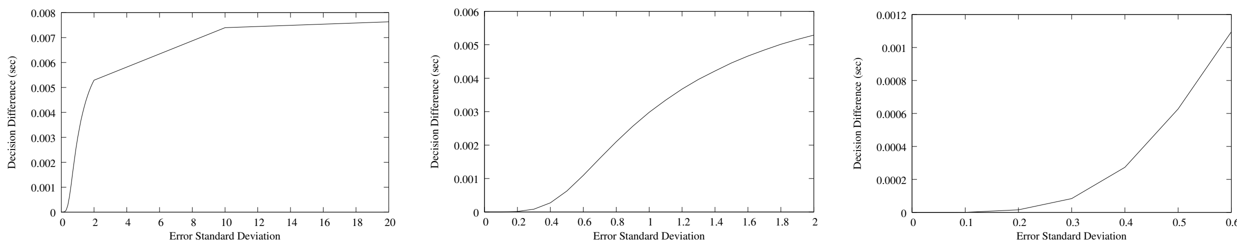
(a) Correct Decision Rate (b) Local Decision Rate (c) Relative Decision Difference

(d) Decision Difference (e) Decision Difference Zoomed (f) Decision Difference Zoomed

Figure 5.11: Plots for RLR = 100 and CCR = 1x10^{-3}

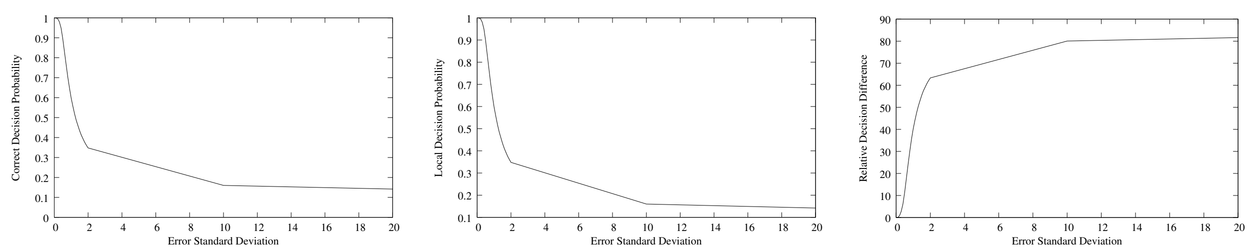


(a) Correct Decision Rate (b) Local Decision Rate (c) Relative Decision Difference

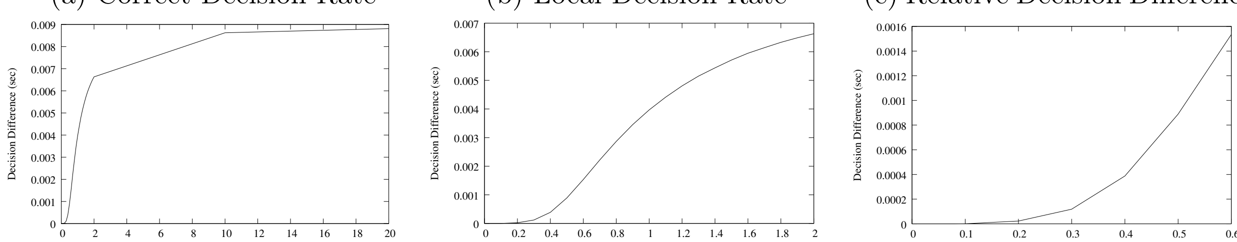


(d) Decision Difference (e) Decision Difference Zoomed (f) Decision Difference Zoomed

Figure 5.12: Plots for RLR = 2 and CCR = 1x10^{-2}

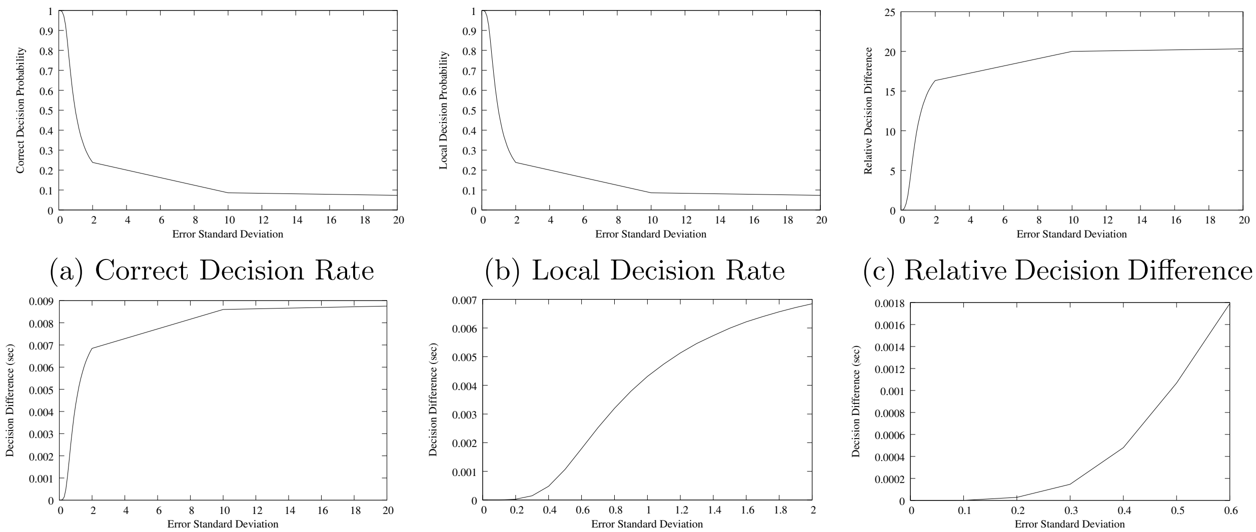


(a) Correct Decision Rate (b) Local Decision Rate (c) Relative Decision Difference



(d) Decision Difference (e) Decision Difference Zoomed (f) Decision Difference Zoomed

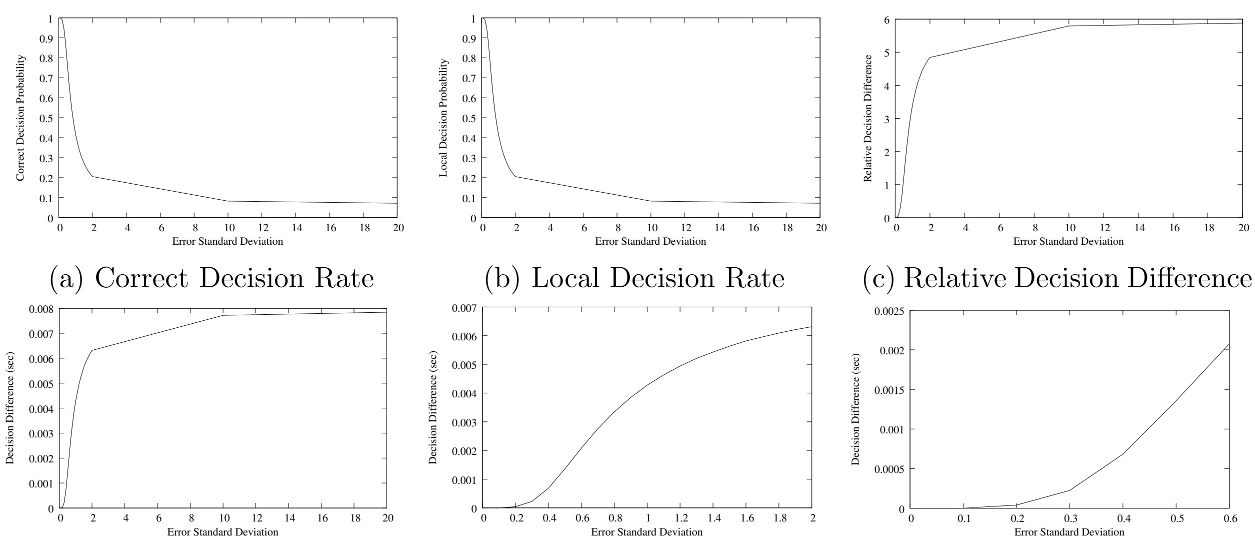
Figure 5.13: Plots for RLR = 8 and CCR = 1x10^{-2}



(a) Correct Decision Rate (b) Local Decision Rate (c) Relative Decision Difference

(d) Decision Difference (e) Decision Difference Zoomed (f) Decision Difference Zoomed

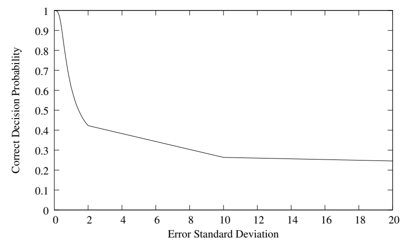
Figure 5.14: Plots for RLR = 32 and CCR = 1x10^{-2}



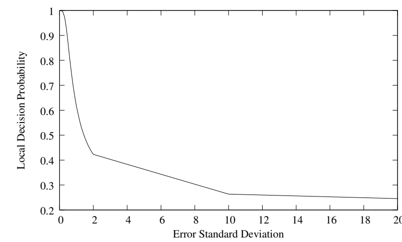
(a) Correct Decision Rate (b) Local Decision Rate (c) Relative Decision Difference

(d) Decision Difference (e) Decision Difference Zoomed (f) Decision Difference Zoomed

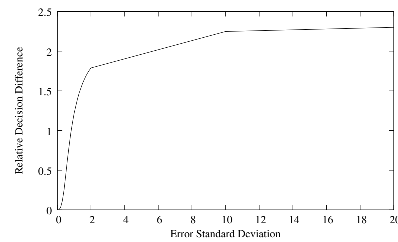
Figure 5.15: Plots for RLR = 100 and CCR = 1x10^{-2}



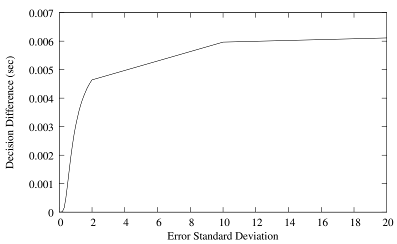
(a) Correct Decision Rate



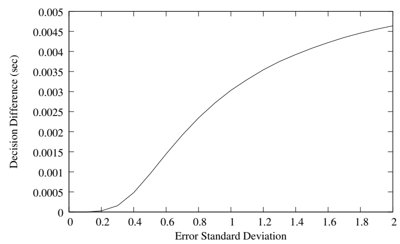
(b) Local Decision Rate



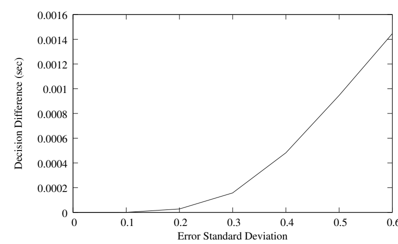
(c) Relative Decision Difference



(d) Decision Difference

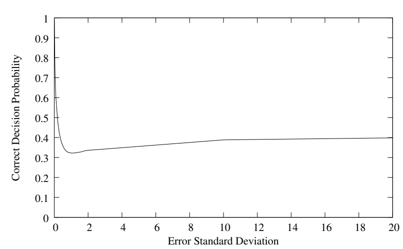


(e) Decision Difference Zoomed

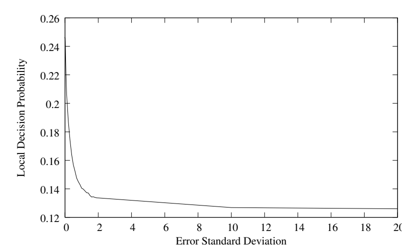


(f) Decision Difference Zoomed

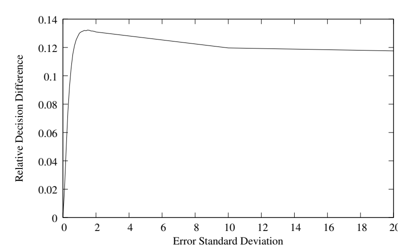
Figure 5.16: Plots for RLR = 2 and CCR = 1



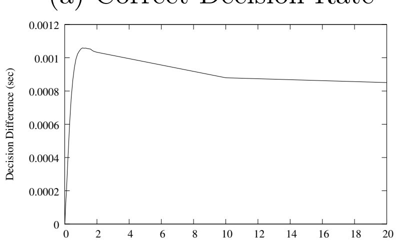
(a) Correct Decision Rate



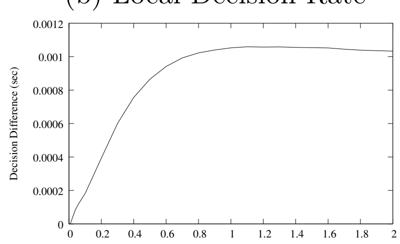
(b) Local Decision Rate



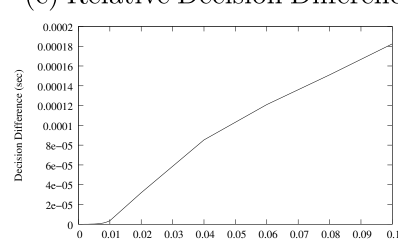
(c) Relative Decision Difference



(d) Decision Difference

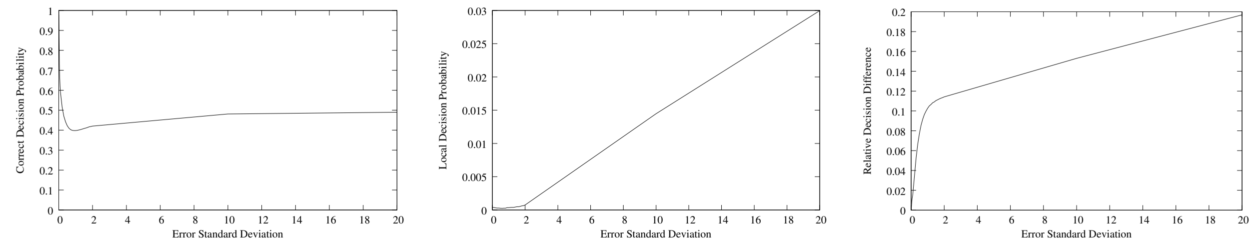


(e) Decision Difference Zoomed

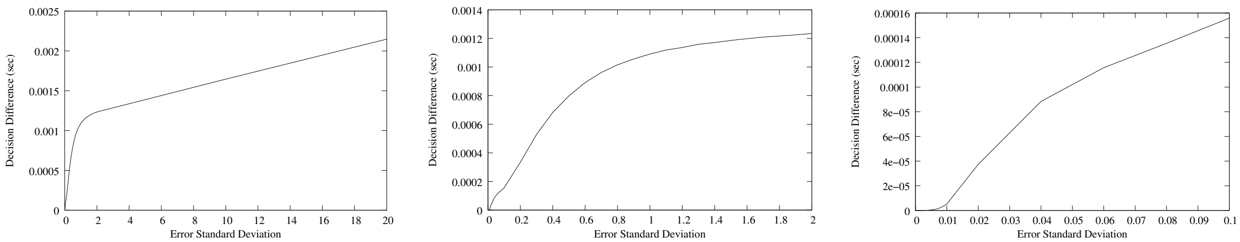


(f) Decision Difference Zoomed

Figure 5.17: Plots for RLR = 8 and CCR = 1

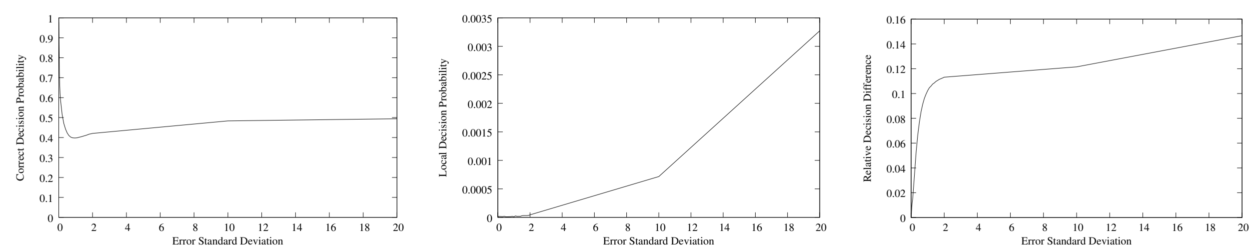


(a) Correct Decision Rate (b) Local Decision Rate (c) Relative Decision Difference

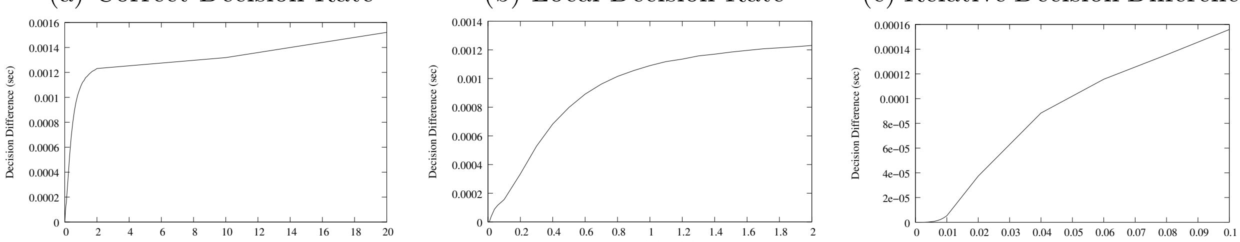


(d) Decision Difference (e) Decision Difference Zoomed (f) Decision Difference Zoomed

Figure 5.18: Plots for RLR = 32 and CCR = 1

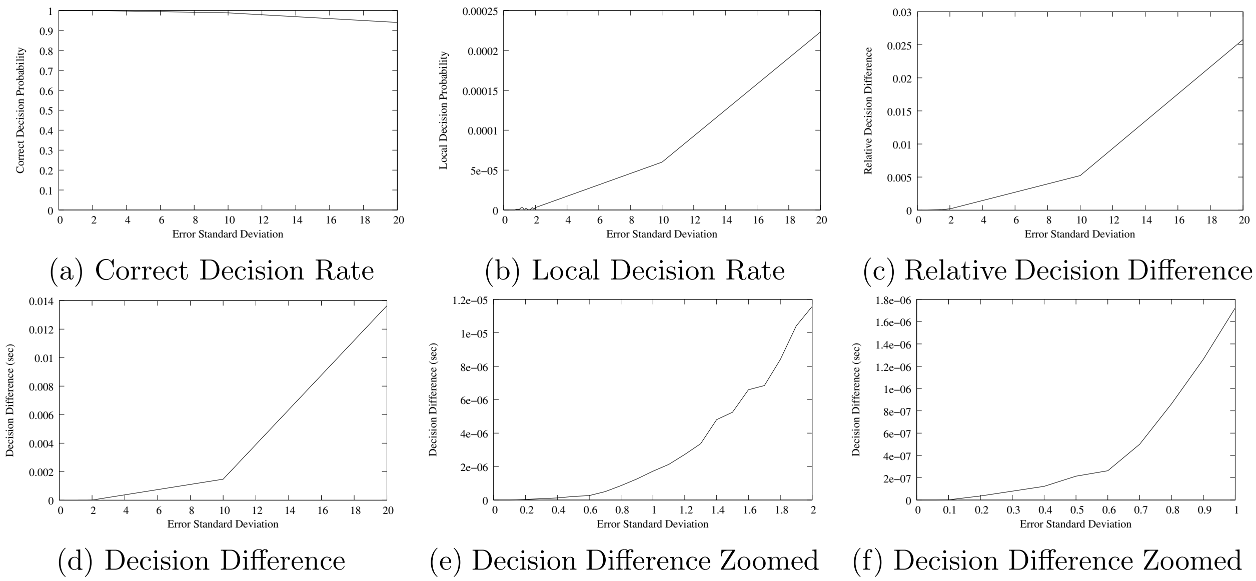


(a) Correct Decision Rate (b) Local Decision Rate (c) Relative Decision Difference



(d) Decision Difference (e) Decision Difference Zoomed (f) Decision Difference Zoomed

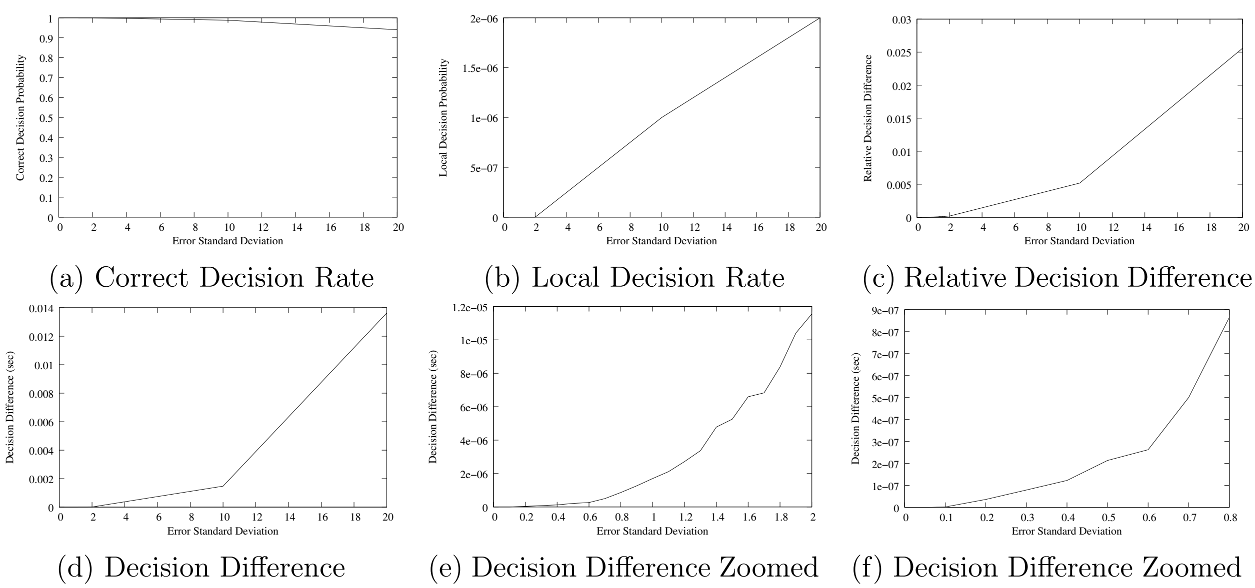
Figure 5.19: Plots for RLR = 100 and CCR = 1



(a) Correct Decision Rate (b) Local Decision Rate (c) Relative Decision Difference

(d) Decision Difference (e) Decision Difference Zoomed (f) Decision Difference Zoomed

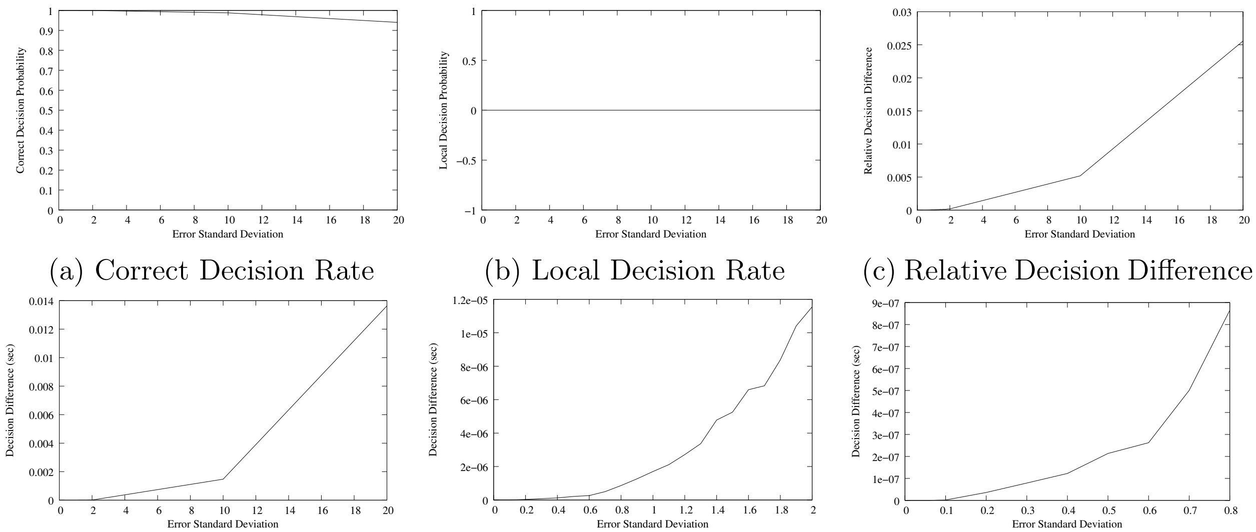
Figure 5.20: Plots for RLR = 2 and CCR = $1x10^3$



(a) Correct Decision Rate (b) Local Decision Rate (c) Relative Decision Difference

(d) Decision Difference (e) Decision Difference Zoomed (f) Decision Difference Zoomed

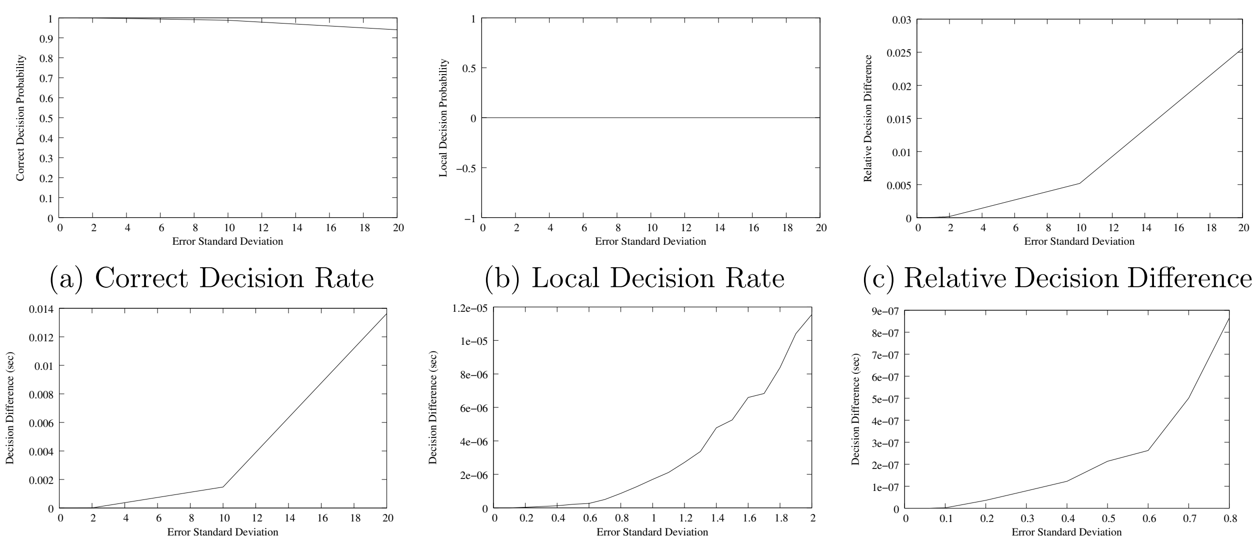
Figure 5.21: Plots for RLR = 8 and CCR = $1x10^3$



(a) Correct Decision Rate (b) Local Decision Rate (c) Relative Decision Difference

(d) Decision Difference (e) Decision Difference Zoomed (f) Decision Difference Zoomed

Figure 5.22: Plots for RLR = 32 and CCR = $1x10^3$



(a) Correct Decision Rate (b) Local Decision Rate (c) Relative Decision Difference

(d) Decision Difference (e) Decision Difference Zoomed (f) Decision Difference Zoomed

Figure 5.23: Plots for RLR = 100 and CCR = $1x10^3$

5.3.2 Monte Carlo Method

The *Monte Carlo* method [81] [82] is a probability analysis used to understand the impact of risk in quantitative analysis and decision making. It provides a wide range of possible outcomes and their probabilities, and allows to make better decisions under uncertainty. This method is used on different applications such as: finance, forecasting, project management, gambling, telecommunications, artificial intelligence, etc.

The Monte Carlo method requires a mathematical model of the decision under consideration. It uses randomness by pulling random values from a probability distribution for the factors with uncertainty. Decision makers are able to determine the range of possible outcomes and the probability of occurrence; their decision is based on the risk they are willing to take to get the needed or wanted outcome.

In this project, we employed the Monte Carlo technique to calculate the expected value for the average decision difference. This tool allowed us to test different parameter values in of our model. We were able to try different: 1) network topologies (i.e., number of hops), 2) bottleneck transmission rates, 3) numbers of clients, 4) numbers of cloud resources, 5) input data sizes, 6) output data sizes, and 7) execution speeds.

The Monte Carlo simulator and the simulation results presented in this dissertation were possible thanks to Elliott Gurrola, a team member of the *UTEP Communication Networks Laboratory* (NetLab).

5.3.3 NS-3 simulation vs. Monte Carlo method

We conducted an extensive set of Monte Carlo simulation experiments to derive the average of the decision difference using our system model. We did not make any assumptions about the distributions of the computing state variables. As such, we ran Monte Carlo simulations for a basket of computing state probability distributions: Uniform, Normal (or Gaussian), Exponential, and Pareto. We set the averages of these distributions to provide a range of CCRs.

Specifically, we used $1e^{-6}$, $1e^{-3}$, 1, and $1e^{3}$. We felt comfortable making some assumptions about the communication state probability distributions. We utilized a uniform distribution for the number of hops as we felt it was equally likely for the number of hops to lie in a certain range. We constrained ourselves to a network with no more than 5 hops between the client and each cloud resource; to best model a cloudlet environment. We utilized the Pareto heavy tail distribution for the queueing delay as the average queueing delay is quite small but there is significant likelihood of large queueing delays. Lastly, we utilized the Normal (or Gaussian) distribution for the transmission channel rates as those rates are relatively stable but may have minor variations due to a wide number of factors.

In our Monte Carlo simulations we compute actual communication delay and purposefully add a zero mean Gaussian random error, ϵ, to that actual communication delay to produce an estimate of that delay. Let $\epsilon = \mathcal{N}(0, \sigma^2)$ where σ is set to some fraction of the total actual communication cost.

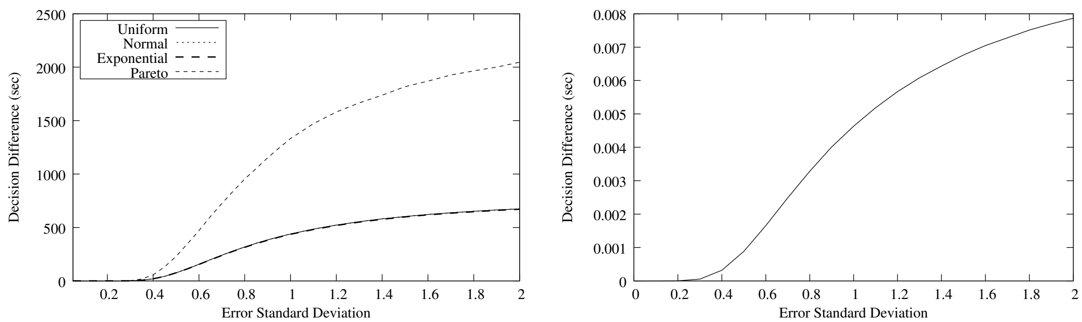
We observe both the decision difference, Δ from Eq.5.2, and the relative difference, $\frac{\Delta}{x(D)} = \frac{x(\hat{D}) - x(D)}{x(D)}$.

Our primary observation is that, initially increasing the standard deviation of the error term has no impact on the decision difference. At some point there is an inflection point after which the decision difference increases sharply. This inflection point is related to the difference in the communication delay values of the choices. If the standard deviation of the error term is set such that it is very unlikely that the error terms change the ordering of the choices, then correct decisions are still made and there is no effect on the decision difference. However, once the standard deviation on the error term becomes large enough relative to the differences in the communication delay of the choices, the ordering can change and incorrect decisions made.

For CCR values of $1e^{-6}$ and $1e^{-3}$, the communication time is the dominating factor and therefore the estimation error must be kept less than 40% to not impact the average decision difference. On the other hand, with a CCR value of $1e^{3}$, the computation time is the dominating factor and therefore the estimation error on the network delay can be very large before we see a major

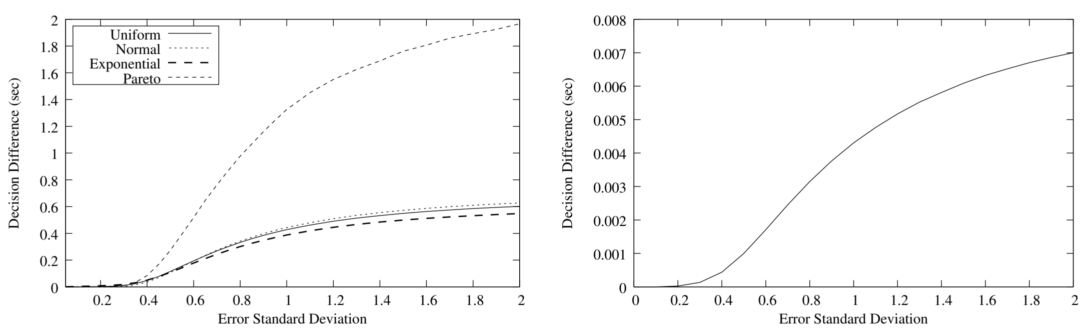
inflection point in the decision difference curve.

Figures 5.24 through 5.27 show the decision difference curves for the Monte Carlo and NS-3 simulation experiments. As we can see the trend is very similar in both simulations. However, the decision difference is not the same because of the different configurations used on each of the simulators (e.g. topology, transmission rates, input and output data sizes, etc.).



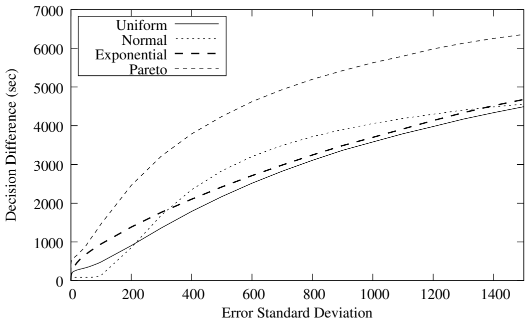
(a) CCR = 1x10^{-6}, Monte Carlo (b) CCR = 1x10^{-6}, NS-3

Figure 5.24: Average decision difference between the decision made with actual costs and the decision made with estimated communication time. Monte Carlo simulations on left hand side and NS-3 simulations on right hand side.

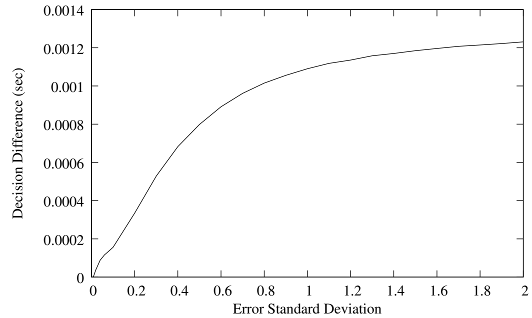


(a) CCR = 1x10^{-3}, Monte Carlo (b) CCR = 1x10^{-3}, NS-3

Figure 5.25: Average decision difference between the decision made with actual costs and the decision made with estimated communication time. Monte Carlo simulations on left hand side and NS-3 simulations on right hand side.

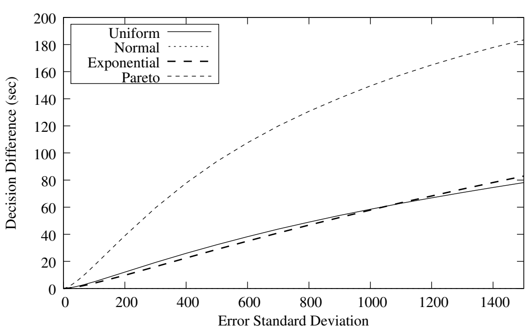


(a) CCR = 1, Monte Carlo

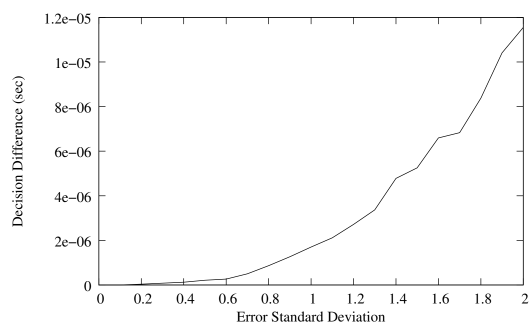


(b) CCR = 1, NS-3

Figure 5.26: Average decision difference between the decision made with actual costs and the decision made with estimated communication time. Monte Carlo simulations on left hand side and NS-3 simulations on right hand side.



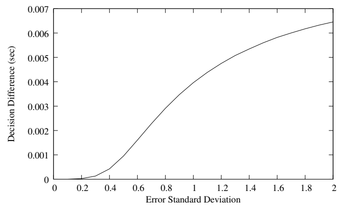
(a) CCR = $1 \text{x} 10^3$, Monte Carlo



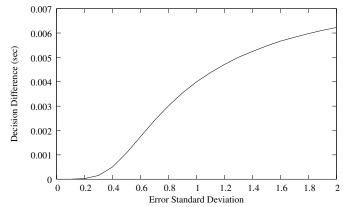
(b) CCR = $1 \text{x} 10^3$, NS-3

Figure 5.27: Average decision difference between the decision made with actual costs and the decision made with estimated communication time. Monte Carlo simulations on left hand side and NS-3 simulations on right hand side.

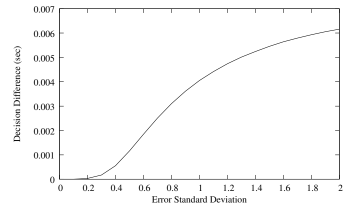
5.4 RLR and Decision Difference Inflection Point



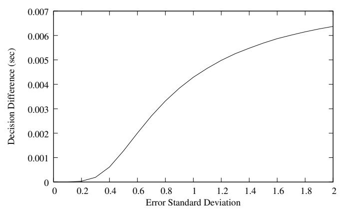
(a) CCR = 0.1, RLR = 4



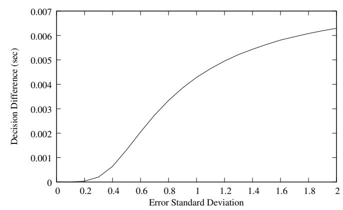
(b) CCR = 0.1, RLR = 8



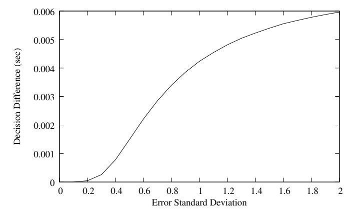
(c) CCR = 0.1, RLR = 10



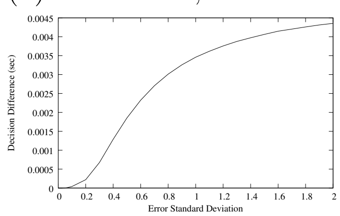
(d) CCR = 0.1, RLR = 11



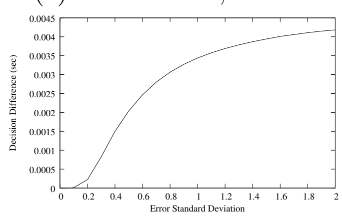
(e) CCR = 0.1, RLR = 12



(f) CCR = 0.1, RLR = 16

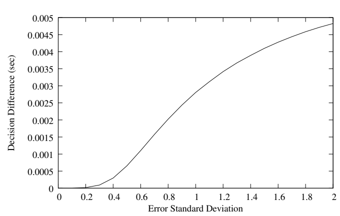


(g) CCR = 0.1, RLR = 32

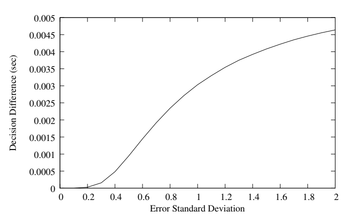


(h) CCR = 0.1, RLR = 64

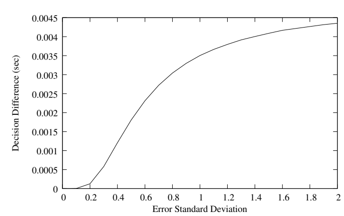
Figure 5.28: Average decision difference with estimated communication time. NS-3 simulations for different RLR values.



(a) CCR = 1, RLR = 1



(b) CCR = 1, RLR = 2



(c) CCR = 1, RLR = 4

Figure 5.29: Average decision difference with estimated communication time. NS-3 simulations for different RLR values.

Observation: From Figures 5.28 and 5.29, we can see that for smaller RLR values the "S" shape is flat, but as the RLR increases, the "S" shape starts getting sharper, thus having impact on the inflection point.

5.5 Summary

By conducting Monte Carlo simulations and NS-3 discrete-event network simulations, we conclude that for CCR values less than 1 (e.g. $1e^{-6}$, $1e^{-3}$), we get an inflection point with a small estimation error. For smaller RLR values the inflection point appears with a larger estimation error, because most of time the local device is preferred over a remote device. However, as the RLR value increases, the inflection point starts to appear with a smaller estimation error. For CCR values greater than 1 (e.g. $1e^{3}$) the estimation error can be very large before we see a major inflection point in the decision difference curve.

Chapter 6: Conclusions and Future Work

This dissertation contains an in-depth discussion of cloud computing that includes the types of clouds, cloud service models, Amazon Web Services, and cloud computing communication patterns. Computation offloading is described in detail, including its use cases, enabling architectures, computation offloading frameworks, and computation offloading criteria. To determine under which conditions computation offloading is beneficial, a computation offloading decision analysis is presented using our system model, our CCR analysis, and our TACC trace data analysis. Such analyses evaluate the conditions in which computation offloading can reduce execution time, and therefore provide insight into the application types and infrastructure (computing and network) scenarios where offloading is beneficial.

Computation offloading is beneficial when the local completion time is greater than the remote completion time. From our computation offloading decision analysis we have found the relationship between two important ratios: 1) Computing-to-Communication Ratio (CCR) and 2) Remote-to-Local Ratio (RLR). In order to benefit from computation offloading, the RLR value has to be greater than the inverse of the CCR value plus one. Also, we were able to derive the F/C inequality that gives a much better insight of the conditions under which computation offloading becomes beneficial for certain real world applications. By conducting numerical experiments with our model, we have derived a significant amount of CCR values, that provide insight into the scenarios under which mobile devices may or may not benefit from computation offloading. We find:

1. When the transmission rate(Γ) is the **same** order of magnitude as the file size (F), and the job size (C) is equal or greater than 1x10^9 instructions, computation offloading is beneficial. This holds true for:

    (a) Γ = 1x10^3 (1kbps), F = 1x10^3 (1KB), $\beta = \{0, 100x10^{-6}, 1x10^{-3}\}$, and h = $\{1, 2, 3, 4\}$

    (b) Γ = $\{1x10^6(1Mbps), 1x10^9(1Gbps)\}$, F = $\{1x10^6(1MB), 1x10^9(1GB)\}$,

$\beta = \{0, 100x10^{-6}, 1x10^{-3}\}$, and h = $\{1, 2, 3, 4, 5\}$

2. When the transmission rate(Γ) is **one** order of magnitude greater than the file size (F), computation offloading is beneficial. This holds true for:

   (a) C = $\{1x10^6, 1x10^9\}$, Γ = 1x10^6 (1Mbps), F = 1x10^3 (1KB), $\beta = \{0, 100x10^{-6}, 1x10^{-3}\}$, and h = $\{1, 2, 3, 4\}$

   (b) C = $\{1x10^6, 1x10^9\}$, Γ = 1x10^9 (1Gbps), F = 1x10^6 (1MB), $\beta = \{0, 100x10^{-6}, 1x10^{-3}\}$, and h = $\{1, 2, 3, 4, 5\}$

   (c) C = $\{1x10^9\}$, Γ = 1x10^6 (1Mbps), F = 1x10^3 (1KB), $\beta = \{0, 100x10^{-6}, 1x10^{-3}\}$, and h = $\{5\}$

3. When the transmission rate(Γ) is **two** orders of magnitude greater than the file size (F), computation offloading is beneficial. This holds true for:

   (a) C = $\{1x10^3, 100x10^3, 1x10^6, 1x10^9\}$, Γ = 1x10^9 (1Gbps), F = 1x10^3 (1KB), $\beta = \{0\}$, and h = $\{1, 2, 3, 4\}$

   (b) C = $\{100x10^3, 1x10^6, 1x10^9\}$, Γ = 1x10^9 (1Gbps), F = 1x10^3 (1KB), $\beta = \{0\}$, and h = $\{5\}$

   (c) C = $\{100x10^3, 1x10^6, 1x10^9\}$, Γ = 1x10^9 (1Gbps), F = 1x10^3 (1KB), $\beta = \{100x10^{-6}, 1x10^{-3}\}$, and h = $\{1, 2, 3, 4, 5\}$

We derived the CCR/RLR inequality from our system model. We also derived the F/C inequality from our system model. The F/C inequality only contains the characteristics of the job (i.e., file size and job size), which helps to easily identify the conditions under which computation offloading is beneficial for a particular application. We have evaluated some real world applications from the TACC trace to identify specific programs that benefit from computation offloading, and we found that:

1. Most of the applications benefit from computation offloading when the mobile device has a slower processor (e.g. MSP430) and the remote device has at least a single server-class core.

2. As the bottleneck transmission rate increases (e.g. from 1kbps to 1Mbps), more applications benefit from computation offloading. From the programs that we used in our analysis, we can see that all of the applications benefit when using the MSP430 processor with a transmission channel of 1Mbps.

3. When using a powerful computing device (e.g. A9) and a slow transmission channel, just few of the applications benefit from computation offloading. However, as the transmission rate increases (e.g. from 1Mbps to 1Gbps), then all of the applications benefit from computation offloading.

In conclusion, we can say that a mobile device benefits from computation offloading when there is high bandwidth available and only a small amount of data needs to be transferred.

By mining real world computation workload data from the Texas Advanced Computing Center (TACC), we have observed that under poor network connections and using powerful handheld devices, computation offloading of some applications to remote servers becomes less beneficial; however, cloudlets become more attractive. We identified several applications that benefit from computation offloading by drawing different CCR thresholds and visualize the trade off between computation and communication.

To understand how accurate an estimation has to be in order to make good decisions at a low risk, we used Monte Carlo simulations and NS-3 network simulations. We concluded that for CCR values less than 1 (e.g. $1e^{-6}$, $1e^{-3}$), we get an inflection point with a small estimation error. For smaller RLR values the inflection point appears with a larger estimation error, because most of time the local device is preferred over a remote device. However, as the RLR value increases, the inflection point starts to appear with a smaller estimation error. For CCR values greater

than 1 (e.g. $1e^3$) the estimation error can be very large before we see a major inflection point in the decision difference curve.

We present our experimental design used for our Monte Carlo simulations and our discrete event simulations using NS-3. We employed the Monte Carlo technique to calculate the expected value for the average decision difference. This tool allowed us to test different parameter values in our model. We were able to try different: 1) network topologies (i.e., number of hops), 2) bottleneck transmission rates, 3) numbers of clients, 4) numbers of cloud resources, 5) input data sizes, 6) output data sizes, and 7) execution speeds. We conducted an extensive set of NS-3 simulation experiments to derive the average decision difference using our system model, and validate the Monte Carlo results. Finally, the results of our simulation experiments are presented along with the conclusions from our experimental data analysis.

6.1 Future Work

There are several paths for future work in our research. The first path would continue to identify specific applications that may benefit from computation offloading. We are interested on finding popular applications that may be potentially used on mobile devices, that are computation-intensive, and that can benefit from computation offloading. We may get more data traces from different data centers (e.g. TACC, IBM, ARL).

The second path would extend the system analysis to consider energy consumption. Our system model can be modified accordingly to understand the impact of the network delay estimation error with respect to energy consumption.

The third path would be to extend the NS-3 simulation experiments to cover more scenarios. These scenarios may have more variation in the: 1) number of hops, 2) bottleneck transmission rates, 3) execution speeds, 4) number of clients, and 5) number of cloud resources.

The final path would perform experiments on physical testbeds to validate our findings with the Monte Carlo method and behavioral model (i.e., NS-3 simulator).

Bibliography

[1] EuroTech, "EuroTech," http://www.eurotech.com/it/prodotti/dispositivi+indossabili/wrist+worn+computers, 2015, [Online; accessed 10-October-2015].

[2] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 23–32, Mar 2013.

[3] P. Bonato, "Wearable sensors/systems and their impact on biomedical engineering," *IEEE Engineering in Medicine and Biology Magazine*, vol. 22, no. 3, pp. 18–20, May 2003.

[4] D. Raskovic, T. Martin, and E. Jovanov, "Medical monitoring applications for wearable computing," *The Computer Journal*, vol. 47, no. 4, pp. 495–504, Jul 2004.

[5] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," *Mob. Netw. Appl.*, vol. 16, no. 3, pp. 270–284, Jun 2011.

[6] N. Fernando, S. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, Jan 2013.

[7] N. Enzai and M. Tang, "A taxonomy of computation offloading in mobile cloud computing," in *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, Apr 2014, pp. 19–28.

[8] M. Sharifi, S. Kafaie, and O. Kashefi, "A survey and taxonomy of cyber foraging of mobile devices," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1232–1243, Oct 2012.

[9] K. Jumar, J. Liu, Y. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, Feb 2013.

[10] T. Sookoor, D. Bruno, and D. Shires, "Allocating tactical high-performance computer (hpc) resources to offloaded computation in battlefield scenarios," Army Research Lab Aberdeeb Proving Ground MD Computational and Information Sciences Directorate, Tech. Rep., Dec 2013.

[11] V. Bronstein and M. OâĂŹConnor, "A taxonomy of data communication for cloud computing," in *Empowering The Optical Edge*, MRV, Ed. MRV Communications, 2013, pp. 1–9.

[12] M. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: Distributed internet computing for it and scientific research," *IEEE Internet Computing*, vol. 13, no. 5, pp. 10–13, Sep 2009.

[13] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, Apr 2010.

[14] B. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Fifth International Joint Conference on INC, IMS and IDC, 2009. NCM'09.*, Aug 2009, pp. 44–51.

[15] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Apr 2010, pp. 27–33.

[16] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications.*

[17] "Nimbus platform," http://www.nimbusproject.org/.

[18] "Eucalyptus," https://www.eucalyptus.com/.

[19] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009. CCGRID '09.*, May 2009, pp. 124–131.

[20] "Amazon web services," aws.amazon.com/.

[21] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. Berman, and P. Maechling, "Data sharing options for scientific workflows on amazon ec2," in *2010 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2010, pp. 1–9.

[22] S. Kailasam, N. Gnanasambandam, J. Dharanipragada, and N. Sharma, "Optimizing service level agreements for autonomic cloud bursting schedulers," in *2010 39th International Conference on Parallel Processing Workshops (ICPPW)*, Sep 2010, pp. 285–294.

[23] Z. Wei, G. Pierre, and C. Chi-Hung, "Cloudtps: Scalable transactions for web applications in the cloud," *IEEE Transactions on Services Computing*, vol. 5, no. 4, pp. 525–539, Nov 2012.

[24] D. Marinescu, *Cloud computing: Theory and practice.* Newnes, 2013.

[25] B. Ahlgren, P. Aranda, P. Chemouil, S. Oueslati, L. Correia, H. Karl, M. SoÌĹllner, and A. Welin, "Content, connectivity, and cloud: ingredients for the network of the future," *IEEE Communications Magazine*, vol. 49, no. 7, pp. 62–70, Jul 2011.

[26] N. Bitar, S. Gringeri, and T. Xia, "Technologies and protocols for data center and cloud networking," *IEEE Communications Magazine*, vol. 51, no. 9, pp. 24–31, Sep 2013.

[27] B. Howard, "Thin is back," *PC Magazine*, vol. 19, no. 7, pp. 985–997, Apr 2000.

[28] S. Yang and J. Nieh, "Thin is in," *PC Magazine*, vol. 19, no. 13, p. 68, Jul 2000.

[29] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[30] M. Chowdhury, M. Zaharia, J. Ma, M. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," *ACM SIGCOMM Computer Communications Review*, vol. 41, no. 4, pp. 98–109, Aug 2011.

[31] M. Fischer and T. Harris, "A model for evaluating the performance of an integrated circuit- and packet-switched multiplex structure," *IEEE Transactions on Communications*, vol. 24, no. 2, pp. 195–202, Feb 1976.

[32] B. Maglaris and M. Schwartz, "Optimal fixed frame multiplexing in integrated line-and packet-switched communication networks," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 263–273, Mar 1982.

[33] M. Beckner, E. G. J.A. Davis, T. Hiller, P. Olson, and G. VanDine, "Integrated packet switching and circuit switching system," US Patent 4 592 048A, May 27, 1986.

[34] C. Gauger, P. Kuhn, E. Breusegem, M. Pickavet, and P. Demeester, "Hybrid optical network architectures: bringing packets and circuits together," *IEEE Communications Magazine*, vol. 44, no. 8, pp. 36–42, Aug 2006.

[35] C. Vadrevu, M. Tornatore, C. Guok, I. Monga, and B. Mukherjee, "Vertical and horizontal circuit/packet integration techniques for the future optical internet," *IEEE Network*, vol. 27, no. 1, pp. 52–58, Jan 2013.

[36] X. Wei, F. Aurzada, M. McGarry, and M. Reisslein, "DyCaPPON: Dynamic circuit and packet passive optical network," *Elsevier Optical Switching and Networking*, vol. 13, pp. 135–147, Jul 2014.

[37] H. Harai, H. Furukawa, K. Fujikawa, T. Miyazawa, and N. Wada, "Optical packet and circuit integrated networks and software defined networking extension," *IEEE/OSA Journal of Lightwave Technology*, vol. 32, no. 16, pp. 2751–2759, Aug 2014.

[38] B. Shneiderman and C. Plaisant, *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Prentice Hall, 2009.

[39] J. Nieh, S. J. Yang, and N. Novik, "A comparison of thin-client computing architectures," Department of Computer Science, Columbia University, Tech. Rep., Nov 2000.

[40] A. Lai and J. Nieh, "Limits of wide-area thin-client computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1, pp. 228–239, Jun 2002.

[41] A. Lai and J. Nieh, "On the performance of wide-area thin-client computing," *ACM Transactions on Computer Systems*, vol. 24, no. 2, pp. 175–209, May 2006.

[42] N. Tolia, D. Andersen, and M. Satyanarayanan, "Quantifying interactive user experience on thin clients," *IEEE Computer*, vol. 39, no. 3, pp. 46–52, Mar 2006.

[43] B. Emmert, A. Binzenhöfer, D. Schlosser, and M. Weiß, "Source traffic characterization for thin client based office applications," in *Open European Summer School and IFIP TC6.6 Workshop on Dependable and Adaptable Networks and Systems (EUNICE)*, vol. 4606, Jul 2007, pp. 86–94.

[44] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 365–386, Aug 1995.

[45] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, Jun 1993.

[46] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 375–385, Jun 1996.

[47] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of datacenter traffic: Measurements and analysis," in *ACM Internet Measurement Conference (IMC)*, Nov 2009, pp. 202–208.

[48] T. Benson, A. Akella, and D. Maltz, "Network traffic characteristics of data centers in the wild," in *ACM Internet Measurement Conference (IMC)*, Nov 2010, pp. 267–280.

[49] R. Birke, L. Chen, and C. Minkenberg, "A datacenter network tale from a server's perspective," in *IEEE/ACM International Symposium on Quality of Service (IWQoS)*, Jun 2013, pp. 1–10.

[50] A. Aghdai, F. Zhang, N. Dasanayake, K. Xi, and H. Chao, "Traffic measurement and analysis in an organic enterprise data center," in *IEEE International Conference on High Performance Switching and Routing (HPSR)*, Jul 2013, pp. 49–55.

[51] K. Claffy, G. Miller, and K. Thompson, "The nature of the beast: recent traffic measurements from an internet backbone," in *Internet Society Conference*, 1998.

[52] Y. Chen, S. Jain, V. Adhikari, Z. Zhang, and K. Xu, "A first look at inter-data center traffic characteristics via Yahoo! datasets," in *IEEE INFOCOM*, Apr 2011, pp. 1620–1628.

[53] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 14–22, Jul 2013.

[54] E. Lagerspetz and S. Tarkoma, "Mobile search and the cloud: The benefits of offloading," in *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, Mar 2011, pp. 117–122.

[55] R. Azuma, "A survey of augmented reality," *Presence*, vol. 6, no. 4, pp. 355–385, Aug 1997.

[56] T. Verbelen, P. Simoens, F. Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proceedings of the third ACM workshop on Mobile cloud computing and services*, Jun 2012, pp. 29–36.

[57] S. Armstrong, "Wireless connectivity for health and sports monitoring: a review," *British journal of sports medicine*, vol. 41, no. 5, pp. 285–289, Jan 2007.

[58] J. Flinn, S. Park, and M. Satyanarayanan, "Balancing performance, energy, and quality in pervasive computing," in *Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCSâĂŹ02)*, Jul 2002, pp. 217–226.

[59] Y. Su and J. Flinn, "Slingshot: deploying stateful services in wireless hotspots," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, Jun 2005, pp. 79–92.

[60] R. Balan, D. Gergle, M. Satyanarayana, and J. Herbsleb, "Simplifying cyber foraging for mobile devices," in *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, Jun 2007, pp. 272–285.

[61] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, Jun 2010, pp. 49–62.

[62] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*, Apr 2011, pp. 301–314.

[63] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proceedings IEEE INFOCOM*, Mar 2012, pp. 945–953.

[64] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Unleashing the power of mobile

cloud computing using thinkair," *Computing Research Repository (CoRR)*, pp. 1–17, May 2011.

[65] R. Ma, K. Lam, and C. Wang, "excloud: Transparent runtime support for scaling mobile applications in cloud," in *2011 International Conference on Cloud and Service Computing (CSC)*, Dec 2011, pp. 103–110.

[66] H. Chen, Y. Lin, and C. Cheng, "Coca: Computation offload to clouds using aop," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2012, pp. 466–473.

[67] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," in *Mobile Computing, Applications, and Services*, Oct 2012, pp. 59–79.

[68] R. Kemp, N. Palmer, T. Kielmann, F. Seinstra, N. Drost, J. Maasen, and H. Bal, "eyedentify: Multimedia cyber foraging from a smartphone," in *11th IEEE International Symposium on Multimedia, 2009. ISM'09*, Dec 2009, pp. 392–399.

[69] R. Wolski, S. Gurun, C. Krintz, and D. Nurmi, "Using bandwidth data to make computation offloading decisions," in *IEEE International Symposium on Parallel and Distributed Processing*, Apr 2008, pp. 1–8.

[70] M. Sookhak, H. Talebian, E. Ahmed, A. Gani, and M. K. Khan, "A review on remote data auditing in single cloud server: Taxonomy and open issues," *Journal of Network and Computer Applications*, vol. 43, no. 0, pp. 121–141, May 2014.

[71] A. Gani, G. M. Nayeem, M. Shiraz, M. Sookhak, M. Whaiduzzaman, and S. Khan, "A review on interworking and mobility techniques for seamless connectivity in mobile cloud computing," *Journal of Network and Computer Applications*, vol. 43, no. 0, pp. 84–102, May 2014.

[72] K. Kumar and Y. Lu, “Cloud computing for mobile users: Can offloading computation save energy?” *Computer*, no. 4, pp. 51–56, Apr 2010.

[73] T. Sookoor, D. Doria, D. Bruno, D. Shires, B. Swenson, and L. Pollock, “Offload destination selection to enable distributed computation on battlefields,” in *2014 IEEE Military Communications Conference (MILCOM)*, Oct 2014, pp. 841–848.

[74] K. Elgazzar, P. Martin, and H. Hassanein, “Cloud-assisted computation offloading to support mobile services,” *IEEE Transactions on Cloud Computing*, pp. 1–17, Aug 2014.

[75] J. Kurose and K. Ross, *Computer Networking: A Top Down Approach, 4e*, 2012, vol. 1.

[76] T. Instruments, “Texas Instruments MSP430,” http://www.ti.com/lit/ds/symlink/msp430g2553.pdf, 2013.

[77] ARM, “Cortex-A8 Processor,” http://www.arm.com/products/processors/cortex-a/cortex-a8.php, 2015.

[78] U. of California, “CPU performance,” https://setiathome.berkeley.edu/cpu_list.php, 2015.

[79] Intel, “Intel Core i3-2300 Mobile Processor Series,” http://download.intel.com/support/processors/corei3/sb/core_i3-2300_m.pdf, 2013.

[80] N.-. Project, “NS-3 Network Simulator,” https://www.nsnam.org/, 2015.

[81] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method, 2e*, 2008, vol. 1.

[82] G. C. Christian Robert, *Monte Carlo Statistical Methods*, 1999, vol. 1.

Appendix A: CCR Analysis Tables

The tables shown in this appendix, contain all the CCR values tabulated by varying: 1) computational job size (C), 2) computation speed (E), 3) file size (F), 4) bottleneck transmission rate (Γ), 5) queueing delay (β), and 6) number of hops (h). The following vectors show the values used to calculate the CCR values: C = [$1x10^3$, $100x10^3$, $1x10^6$, $1x10^9$] (instructions), E = [$10x10^9$, $100x10^9$, $1x10^{12}$, $10x10^{12}$] (instructions / second), F = [$1x10^3$, $1x10^6$, $1x10^9$, $1x10^{12}$] (bytes), Γ = [$1x10^3$, $1x10^6$, $1x10^9$] (bits / second), β = [0, $100x10^{-6}$, $1x10^{-3}$] (seconds), h = [1, 2, 3, 4, 5] (hops). Entries in light green color are the CCR values less than the RLR lower limit, entries in dark green color are the CCR values using 1/3 of the RLR upper limit, entries in yellow color are the CCR values using 2/3 of the RLR upper limit, entries in blue color are the CCR values using the RLR upper limit, and entries in red color are the CCR values greater than the RLR upper limit.

Table A.1: CCR values for: h = 1, β = 0, Γ = $1x10^3$, F = $1x10^3$

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$4.96x10^{-09}$	$4.96x10^{-10}$	$4.96x10^{-11}$	$4.96x10^{-12}$
$100x10^3$	$4.96x10^{-07}$	$4.96x10^{-08}$	$4.96x10^{-09}$	$4.96x10^{-10}$
$1x10^6$	$4.96x10^{-06}$	$4.96x10^{-07}$	$4.96x10^{-08}$	$4.96x10^{-09}$
$1x10^9$	$4.96x10^{-03}$	$4.96x10^{-04}$	$4.96x10^{-05}$	$4.96x10^{-06}$

Table A.2: CCR values for: h = 2, β = 0, Γ = $1x10^3$, F = $1x10^3$

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$3.10x10^{-09}$	$3.10x10^{-10}$	$3.10x10^{-11}$	$3.10x10^{-12}$
$100x10^3$	$3.10x10^{-07}$	$3.10x10^{-08}$	$3.10x10^{-09}$	$3.10x10^{-10}$
$1x10^6$	$3.10x10^{-06}$	$3.10x10^{-07}$	$3.10x10^{-08}$	$3.10x10^{-09}$
$1x10^9$	$3.10x10^{-03}$	$3.10x10^{-04}$	$3.10x10^{-05}$	$3.10x10^{-06}$

Table A.3: CCR values for: h = 3, β = 0, Γ = $1x10^3$, F = $1x10^3$

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$2.25x10^{-09}$	$2.25x10^{-10}$	$2.25x10^{-11}$	$2.25x10^{-12}$
$100x10^3$	$2.25x10^{-07}$	$2.25x10^{-08}$	$2.25x10^{-09}$	$2.25x10^{-10}$
$1x10^6$	$2.25x10^{-06}$	$2.25x10^{-07}$	$2.25x10^{-08}$	$2.25x10^{-09}$
$1x10^9$	$2.25x10^{-03}$	$2.25x10^{-04}$	$2.25x10^{-05}$	$2.25x10^{-06}$

Table A.4: CCR values for: h = 4, β = 0, Γ = $1x10^3$, F = $1x10^3$

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.77x10^{-09}$	$1.77x10^{-10}$	$1.77x10^{-11}$	$1.77x10^{-12}$
$100x10^3$	$1.77x10^{-07}$	$1.77x10^{-08}$	$1.77x10^{-09}$	$1.77x10^{-10}$
$1x10^6$	$1.77x10^{-06}$	$1.77x10^{-07}$	$1.77x10^{-08}$	$1.77x10^{-09}$
$1x10^9$	$1.77x10^{-03}$	$1.77x10^{-04}$	$1.77x10^{-05}$	$1.77x10^{-06}$

Table A.5: CCR values for: h = 5, β = 0, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.46x10^{-09}	1.46x10^{-10}	1.46x10^{-11}	1.46x10^{-12}
100x10^3	1.46x10^{-07}	1.46x10^{-08}	1.46x10^{-09}	1.46x10^{-10}
1x10^6	1.46x10^{-06}	1.46x10^{-07}	1.46x10^{-08}	1.46x10^{-09}
1x10^9	1.46x10^{-03}	1.46x10^{-04}	1.46x10^{-05}	1.46x10^{-06}

Table A.6: CCR values for: h = 1, β = 0.0001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.96x10^{-09}	4.96x10^{-10}	4.96x10^{-11}	4.96x10^{-12}
100x10^3	4.96x10^{-07}	4.96x10^{-08}	4.96x10^{-09}	4.96x10^{-10}
1x10^6	4.96x10^{-06}	4.96x10^{-07}	4.96x10^{-08}	4.96x10^{-09}
1x10^9	4.96x10^{-03}	4.96x10^{-04}	4.96x10^{-05}	4.96x10^{-06}

Table A.7: CCR values for: h = 2, β = 0.0001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	3.10x10^{-09}	3.10x10^{-10}	3.10x10^{-11}	3.10x10^{-12}
100x10^3	3.10x10^{-07}	3.10x10^{-08}	3.10x10^{-09}	3.10x10^{-10}
1x10^6	3.10x10^{-06}	3.10x10^{-07}	3.10x10^{-08}	3.10x10^{-09}
1x10^9	3.10x10^{-03}	3.10x10^{-04}	3.10x10^{-05}	3.10x10^{-06}

Table A.8: CCR values for: h = 3, β = 0.0001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.25x10^{-09}	2.25x10^{-10}	2.25x10^{-11}	2.25x10^{-12}
100x10^3	2.25x10^{-07}	2.25x10^{-08}	2.25x10^{-09}	2.25x10^{-10}
1x10^6	2.25x10^{-06}	2.25x10^{-07}	2.25x10^{-08}	2.25x10^{-09}
1x10^9	2.25x10^{-03}	2.25x10^{-04}	2.25x10^{-05}	2.25x10^{-06}

Table A.9: CCR values for: h = 4, β = 0.0001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.77x10^{-09}	1.77x10^{-10}	1.77x10^{-11}	1.77x10^{-12}
100x10^3	1.77x10^{-07}	1.77x10^{-08}	1.77x10^{-09}	1.77x10^{-10}
1x10^6	1.77x10^{-06}	1.77x10^{-07}	1.77x10^{-08}	1.77x10^{-09}
1x10^9	1.77x10^{-03}	1.77x10^{-04}	1.77x10^{-05}	1.77x10^{-06}

Table A.10: CCR values for: h = 5, β = 0.0001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.46x10^{-09}	1.46x10^{-10}	1.46x10^{-11}	1.46x10^{-12}
100x10^3	1.46x10^{-07}	1.46x10^{-08}	1.46x10^{-09}	1.46x10^{-10}
1x10^6	1.46x10^{-06}	1.46x10^{-07}	1.46x10^{-08}	1.46x10^{-09}
1x10^9	1.46x10^{-03}	1.46x10^{-04}	1.46x10^{-05}	1.46x10^{-06}

Table A.11: CCR values for: h = 1, β = 0.001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.96x10^{-09}	4.96x10^{-10}	4.96x10^{-11}	4.96x10^{-12}
100x10^3	4.96x10^{-07}	4.96x10^{-08}	4.96x10^{-09}	4.96x10^{-10}
1x10^6	4.96x10^{-06}	4.96x10^{-07}	4.96x10^{-08}	4.96x10^{-09}
1x10^9	4.96x10^{-03}	4.96x10^{-04}	4.96x10^{-05}	4.96x10^{-06}

Table A.12: CCR values for: h = 2, β = 0.001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	3.10x10^{-09}	3.10x10^{-10}	3.10x10^{-11}	3.10x10^{-12}
100x10^3	3.10x10^{-07}	3.10x10^{-08}	3.10x10^{-09}	3.10x10^{-10}
1x10^6	3.10x10^{-06}	3.10x10^{-07}	3.10x10^{-08}	3.10x10^{-09}
1x10^9	3.10x10^{-03}	3.10x10^{-04}	3.10x10^{-05}	3.10x10^{-06}

Table A.13: CCR values for: h = 3, β = 0.001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.25x10^{-09}	2.25x10^{-10}	2.25x10^{-11}	2.25x10^{-12}
100x10^3	2.25x10^{-07}	2.25x10^{-08}	2.25x10^{-09}	2.25x10^{-10}
1x10^6	2.25x10^{-06}	2.25x10^{-07}	2.25x10^{-08}	2.25x10^{-09}
1x10^9	2.25x10^{-03}	2.25x10^{-04}	2.25x10^{-05}	2.25x10^{-06}

Table A.14: CCR values for: h = 4, β = 0.001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.77x10^{-09}	1.77x10^{-10}	1.77x10^{-11}	1.77x10^{-12}
100x10^3	1.77x10^{-07}	1.77x10^{-08}	1.77x10^{-09}	1.77x10^{-10}
1x10^6	1.77x10^{-06}	1.77x10^{-07}	1.77x10^{-08}	1.77x10^{-09}
1x10^9	1.77x10^{-03}	1.77x10^{-04}	1.77x10^{-05}	1.77x10^{-06}

Table A.15: CCR values for: h = 5, β = 0.001, Γ = 1x10^3, F = 1x10^3

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.46x10^{-09}	1.46x10^{-10}	1.46x10^{-11}	1.46x10^{-12}
100x10^3	1.46x10^{-07}	1.46x10^{-08}	1.46x10^{-09}	1.46x10^{-10}
1x10^6	1.46x10^{-06}	1.46x10^{-07}	1.46x10^{-08}	1.46x10^{-09}
1x10^9	1.46x10^{-03}	1.46x10^{-04}	1.46x10^{-05}	1.46x10^{-06}

Table A.16: CCR values for: h = 1, β = 0, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.96x10^{-06}	4.96x10^{-07}	4.96x10^{-08}	4.96x10^{-09}
100x10^3	4.96x10^{-04}	4.96x10^{-05}	4.96x10^{-06}	4.96x10^{-07}
1x10^6	4.96x10^{-03}	4.96x10^{-04}	4.96x10^{-05}	4.96x10^{-06}
1x10^9	4.96x10^{+00}	4.96x10^{-01}	4.96x10^{-02}	4.96x10^{-03}

Table A.17: CCR values for: h = 2, β = 0, Γ = 1x10^{6}, F = 1x10^{3}

E / C	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	3.10x10^{-06}	3.10x10^{-07}	3.10x10^{-08}	3.10x10^{-09}
100x10^{3}	3.10x10^{-04}	3.10x10^{-05}	3.10x10^{-06}	3.10x10^{-07}
1x10^{6}	3.10x10^{-03}	3.10x10^{-04}	3.10x10^{-05}	3.10x10^{-06}
1x10^{9}	3.10x10^{+00}	3.10x10^{-01}	3.10x10^{-02}	3.10x10^{-03}

Table A.18: CCR values for: h = 3, β = 0, Γ = 1x10^{6}, F = 1x10^{3}

E / C	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	2.25x10^{-06}	2.25x10^{-07}	2.25x10^{-08}	2.25x10^{-09}
100x10^{3}	2.25x10^{-04}	2.25x10^{-05}	2.25x10^{-06}	2.25x10^{-07}
1x10^{6}	2.25x10^{-03}	2.25x10^{-04}	2.25x10^{-05}	2.25x10^{-06}
1x10^{9}	2.25x10^{+00}	2.25x10^{-01}	2.25x10^{-02}	2.25x10^{-03}

Table A.19: CCR values for: h = 4, β = 0, Γ = 1x10^{6}, F = 1x10^{3}

E / C	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.77x10^{-06}	1.77x10^{-07}	1.77x10^{-08}	1.77x10^{-09}
100x10^{3}	1.77x10^{-04}	1.77x10^{-05}	1.77x10^{-06}	1.77x10^{-07}
1x10^{6}	1.77x10^{-03}	1.77x10^{-04}	1.77x10^{-05}	1.77x10^{-06}
1x10^{9}	1.77x10^{+00}	1.77x10^{-01}	1.77x10^{-02}	1.77x10^{-03}

Table A.20: CCR values for: h = 5, β = 0, Γ = 1x10^{6}, F = 1x10^{3}

E / C	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.46x10^{-06}	1.46x10^{-07}	1.46x10^{-08}	1.46x10^{-09}
100x10^{3}	1.46x10^{-04}	1.46x10^{-05}	1.46x10^{-06}	1.46x10^{-07}
1x10^{6}	1.46x10^{-03}	1.46x10^{-04}	1.46x10^{-05}	1.46x10^{-06}
1x10^{9}	1.46x10^{+00}	1.46x10^{-01}	1.46x10^{-02}	1.46x10^{-03}

Table A.21: CCR values for: h = 1, β = 0.0001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.94×10^{-06}	4.94×10^{-07}	4.94×10^{-08}	4.94×10^{-09}
100x10^3	4.94×10^{-04}	4.94×10^{-05}	4.94×10^{-06}	4.94×10^{-07}
1x10^6	4.94×10^{-03}	4.94×10^{-04}	4.94×10^{-05}	4.94×10^{-06}
1x10^9	$4.94 \times 10^{+00}$	4.94×10^{-01}	4.94×10^{-02}	4.94×10^{-03}

Table A.22: CCR values for: h = 2, β = 0.0001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	3.08×10^{-06}	3.08×10^{-07}	3.08×10^{-08}	3.08×10^{-09}
100x10^3	3.08×10^{-04}	3.08×10^{-05}	3.08×10^{-06}	3.08×10^{-07}
1x10^6	3.08×10^{-03}	3.08×10^{-04}	3.08×10^{-05}	3.08×10^{-06}
1x10^9	$3.08 \times 10^{+00}$	3.08×10^{-01}	3.08×10^{-02}	3.08×10^{-03}

Table A.23: CCR values for: h = 3, β = 0.0001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.24×10^{-06}	2.24×10^{-07}	2.24×10^{-08}	2.24×10^{-09}
100x10^3	2.24×10^{-04}	2.24×10^{-05}	2.24×10^{-06}	2.24×10^{-07}
1x10^6	2.24×10^{-03}	2.24×10^{-04}	2.24×10^{-05}	2.24×10^{-06}
1x10^9	$2.24 \times 10^{+00}$	2.24×10^{-01}	2.24×10^{-02}	2.24×10^{-03}

Table A.24: CCR values for: h = 4, β = 0.0001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.76×10^{-06}	1.76×10^{-07}	1.76×10^{-08}	1.76×10^{-09}
100x10^3	1.76×10^{-04}	1.76×10^{-05}	1.76×10^{-06}	1.76×10^{-07}
1x10^6	1.76×10^{-03}	1.76×10^{-04}	1.76×10^{-05}	1.76×10^{-06}
1x10^9	$1.76 \times 10^{+00}$	1.76×10^{-01}	1.76×10^{-02}	1.76×10^{-03}

Table A.25: CCR values for: h = 5, β = 0.0001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.44x10^{-06}	1.44x10^{-07}	1.44x10^{-08}	1.44x10^{-09}
100x10^3	1.44x10^{-04}	1.44x10^{-05}	1.44x10^{-06}	1.44x10^{-07}
1x10^6	1.44x10^{-03}	1.44x10^{-04}	1.44x10^{-05}	1.44x10^{-06}
1x10^9	1.44x10^{+00}	1.44x10^{-01}	1.44x10^{-02}	1.44x10^{-03}

Table A.26: CCR values for: h = 1, β = 0.001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.73x10^{-06}	4.73x10^{-07}	4.73x10^{-08}	4.73x10^{-09}
100x10^3	4.73x10^{-04}	4.73x10^{-05}	4.73x10^{-06}	4.73x10^{-07}
1x10^6	4.73x10^{-03}	4.73x10^{-04}	4.73x10^{-05}	4.73x10^{-06}
1x10^9	4.73x10^{+00}	4.73x10^{-01}	4.73x10^{-02}	4.73x10^{-03}

Table A.27: CCR values for: h = 2, β = 0.001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.92x10^{-06}	2.92x10^{-07}	2.92x10^{-08}	2.92x10^{-09}
100x10^3	2.92x10^{-04}	2.92x10^{-05}	2.92x10^{-06}	2.92x10^{-07}
1x10^6	2.92x10^{-03}	2.92x10^{-04}	2.92x10^{-05}	2.92x10^{-06}
1x10^9	2.92x10^{+00}	2.92x10^{-01}	2.92x10^{-02}	2.92x10^{-03}

Table A.28: CCR values for: h = 3, β = 0.001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.11x10^{-06}	2.11x10^{-07}	2.11x10^{-08}	2.11x10^{-09}
100x10^3	2.11x10^{-04}	2.11x10^{-05}	2.11x10^{-06}	2.11x10^{-07}
1x10^6	2.11x10^{-03}	2.11x10^{-04}	2.11x10^{-05}	2.11x10^{-06}
1x10^9	2.11x10^{+00}	2.11x10^{-01}	2.11x10^{-02}	2.11x10^{-03}

Table A.29: CCR values for: h = 4, β = 0.001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.65x10^{-06}	1.65x10^{-07}	1.65x10^{-08}	1.65x10^{-09}
100x10^3	1.65x10^{-04}	1.65x10^{-05}	1.65x10^{-06}	1.65x10^{-07}
1x10^6	1.65x10^{-03}	1.65x10^{-04}	1.65x10^{-05}	1.65x10^{-06}
1x10^9	1.65x10^{+00}	1.65x10^{-01}	1.65x10^{-02}	1.65x10^{-03}

Table A.30: CCR values for: h = 5, β = 0.001, Γ = 1x10^6, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.36x10^{-06}	1.36x10^{-07}	1.36x10^{-08}	1.36x10^{-09}
100x10^3	1.36x10^{-04}	1.36x10^{-05}	1.36x10^{-06}	1.36x10^{-07}
1x10^6	1.36x10^{-03}	1.36x10^{-04}	1.36x10^{-05}	1.36x10^{-06}
1x10^9	1.36x10^{+00}	1.36x10^{-01}	1.36x10^{-02}	1.36x10^{-03}

Table A.31: CCR values for: h = 1, β = 0, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.96x10^{-03}	4.96x10^{-04}	4.96x10^{-05}	4.96x10^{-06}
100x10^3	4.96x10^{-01}	4.96x10^{-02}	4.96x10^{-03}	4.96x10^{-04}
1x10^6	4.96x10^{+00}	4.96x10^{-01}	4.96x10^{-02}	4.96x10^{-03}
1x10^9	4.96x10^{+03}	4.96x10^{+02}	4.96x10^{+01}	4.96x10^{+00}

Table A.32: CCR values for: h = 2, β = 0, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	3.10x10^{-03}	3.10x10^{-04}	3.10x10^{-05}	3.10x10^{-06}
100x10^3	3.10x10^{-01}	3.10x10^{-02}	3.10x10^{-03}	3.10x10^{-04}
1x10^6	3.10x10^{+00}	3.10x10^{-01}	3.10x10^{-02}	3.10x10^{-03}
1x10^9	3.10x10^{+03}	3.10x10^{+02}	3.10x10^{+01}	3.10x10^{+00}

Table A.33: CCR values for: h = 3, β = 0, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.25x10^-03	2.25x10^-04	2.25x10^-05	2.25x10^-06
100x10^3	2.25x10^-01	2.25x10^-02	2.25x10^-03	2.25x10^-04
1x10^6	2.25x10^+00	2.25x10^-01	2.25x10^-02	2.25x10^-03
1x10^9	2.25x10^+03	2.25x10^+02	2.25x10^+01	2.25x10^+00

Table A.34: CCR values for: h = 4, β = 0, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.77x10^-03	1.77x10^-04	1.77x10^-05	1.77x10^-06
100x10^3	1.77x10^-01	1.77x10^-02	1.77x10^-03	1.77x10^-04
1x10^6	1.77x10^+00	1.77x10^-01	1.77x10^-02	1.77x10^-03
1x10^9	1.77x10^+03	1.77x10^+02	1.77x10^+01	1.77x10^+00

Table A.35: CCR values for: h = 5, β = 0, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.46x10^-03	1.46x10^-04	1.46x10^-05	1.46x10^-06
100x10^3	1.46x10^-01	1.46x10^-02	1.46x10^-03	1.46x10^-04
1x10^6	1.46x10^+00	1.46x10^-01	1.46x10^-02	1.46x10^-03
1x10^9	1.46x10^+03	1.46x10^+02	1.46x10^+01	1.46x10^+00

Table A.36: CCR values for: h = 1, β = 0.0001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	8.32x10^-04	8.32x10^-05	8.32x10^-06	8.32x10^-07
100x10^3	8.32x10^-02	8.32x10^-03	8.32x10^-04	8.32x10^-05
1x10^6	8.32x10^-01	8.32x10^-02	8.32x10^-03	8.32x10^-04
1x10^9	8.32x10^+02	8.32x10^+01	8.32x10^+00	8.32x10^-01

Table A.37: CCR values for: h = 2, β = 0.0001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.31x10^{-04}	4.31x10^{-05}	4.31x10^{-06}	4.31x10^{-07}
100x10^3	4.31x10^{-02}	4.31x10^{-03}	4.31x10^{-04}	4.31x10^{-05}
1x10^6	4.31x10^{-01}	4.31x10^{-02}	4.31x10^{-03}	4.31x10^{-04}
1x10^9	4.31x10^{+02}	4.31x10^{+01}	4.31x10^{+00}	4.31x10^{-01}

Table A.38: CCR values for: h = 3, β = 0.0001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.90x10^{-04}	2.90x10^{-05}	2.90x10^{-06}	2.90x10^{-07}
100x10^3	2.90x10^{-02}	2.90x10^{-03}	2.90x10^{-04}	2.90x10^{-05}
1x10^6	2.90x10^{-01}	2.90x10^{-02}	2.90x10^{-03}	2.90x10^{-04}
1x10^9	2.90x10^{+02}	2.90x10^{+01}	2.90x10^{+00}	2.90x10^{-01}

Table A.39: CCR values for: h = 4, β = 0.0001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.19x10^{-04}	2.19x10^{-05}	2.19x10^{-06}	2.19x10^{-07}
100x10^3	2.19x10^{-02}	2.19x10^{-03}	2.19x10^{-04}	2.19x10^{-05}
1x10^6	2.19x10^{-01}	2.19x10^{-02}	2.19x10^{-03}	2.19x10^{-04}
1x10^9	2.19x10^{+02}	2.19x10^{+01}	2.19x10^{+00}	2.19x10^{-01}

Table A.40: CCR values for: h = 5, β = 0.0001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.76x10^{-04}	1.76x10^{-05}	1.76x10^{-06}	1.76x10^{-07}
100x10^3	1.76x10^{-02}	1.76x10^{-03}	1.76x10^{-04}	1.76x10^{-05}
1x10^6	1.76x10^{-01}	1.76x10^{-02}	1.76x10^{-03}	1.76x10^{-04}
1x10^9	1.76x10^{+02}	1.76x10^{+01}	1.76x10^{+00}	1.76x10^{-01}

Table A.41: CCR values for: h = 1, β = 0.001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	9.80x10^{-05}	9.80x10^{-06}	9.80x10^{-07}	9.80x10^{-08}
100x10^3	9.80x10^{-03}	9.80x10^{-04}	9.80x10^{-05}	9.80x10^{-06}
1x10^6	9.80x10^{-02}	9.80x10^{-03}	9.80x10^{-04}	9.80x10^{-05}
1x10^9	9.80x10^{+01}	9.80x10^{+00}	9.80x10^{-01}	9.80x10^{-02}

Table A.42: CCR values for: h = 2, β = 0.001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	4.92x10^{-05}	4.92x10^{-06}	4.92x10^{-07}	4.92x10^{-08}
100x10^3	4.92x10^{-03}	4.92x10^{-04}	4.92x10^{-05}	4.92x10^{-06}
1x10^6	4.92x10^{-02}	4.92x10^{-03}	4.92x10^{-04}	4.92x10^{-05}
1x10^9	4.92x10^{+01}	4.92x10^{+00}	4.92x10^{-01}	4.92x10^{-02}

Table A.43: CCR values for: h = 3, β = 0.001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	3.28x10^{-05}	3.28x10^{-06}	3.28x10^{-07}	3.28x10^{-08}
100x10^3	3.28x10^{-03}	3.28x10^{-04}	3.28x10^{-05}	3.28x10^{-06}
1x10^6	3.28x10^{-02}	3.28x10^{-03}	3.28x10^{-04}	3.28x10^{-05}
1x10^9	3.28x10^{+01}	3.28x10^{+00}	3.28x10^{-01}	3.28x10^{-02}

Table A.44: CCR values for: h = 4, β = 0.001, Γ = 1x10^9, F = 1x10^3

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	2.47x10^{-05}	2.47x10^{-06}	2.47x10^{-07}	2.47x10^{-08}
100x10^3	2.47x10^{-03}	2.47x10^{-04}	2.47x10^{-05}	2.47x10^{-06}
1x10^6	2.47x10^{-02}	2.47x10^{-03}	2.47x10^{-04}	2.47x10^{-05}
1x10^9	2.47x10^{+01}	2.47x10^{+00}	2.47x10^{-01}	2.47x10^{-02}

Table A.45: CCR values for: h = 5, β = 0.001, Γ = $1x10^9$, F = $1x10^3$

C \ E	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.97x10^{-05}$	$1.97x10^{-06}$	$1.97x10^{-07}$	$1.97x10^{-08}$
$100x10^3$	$1.97x10^{-03}$	$1.97x10^{-04}$	$1.97x10^{-05}$	$1.97x10^{-06}$
$1x10^6$	$1.97x10^{-02}$	$1.97x10^{-03}$	$1.97x10^{-04}$	$1.97x10^{-05}$
$1x10^9$	$1.97x10^{+01}$	$1.97x10^{+00}$	$1.97x10^{-01}$	$1.97x10^{-02}$

Table A.46: CCR values for: h = 1, β = 0, Γ = $1x10^3$, F = $1x10^6$

C \ E	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$
$100x10^3$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$	$1.25x10^{-12}$
$1x10^6$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
$1x10^9$	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$

Table A.47: CCR values for: h = 2, β = 0, Γ = $1x10^3$, F = $1x10^6$

C \ E	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$
$100x10^3$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$	$1.25x10^{-12}$
$1x10^6$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
$1x10^9$	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$

Table A.48: CCR values for: h = 3, β = 0, Γ = $1x10^3$, F = $1x10^6$

C \ E	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.24x10^{-11}$	$1.24x10^{-12}$	$1.24x10^{-13}$	$1.24x10^{-14}$
$100x10^3$	$1.24x10^{-09}$	$1.24x10^{-10}$	$1.24x10^{-11}$	$1.24x10^{-12}$
$1x10^6$	$1.24x10^{-08}$	$1.24x10^{-09}$	$1.24x10^{-10}$	$1.24x10^{-11}$
$1x10^9$	$1.24x10^{-05}$	$1.24x10^{-06}$	$1.24x10^{-07}$	$1.24x10^{-08}$

Table A.49: CCR values for: h = 4, β = 0, Γ = 1x10^3, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-11}	1.24x10^{-12}	1.24x10^{-13}	1.24x10^{-14}
100x10^3	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}	1.24x10^{-12}
1x10^6	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
1x10^9	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}

Table A.50: CCR values for: h = 5, β = 0, Γ = 1x10^3, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-11}	1.24x10^{-12}	1.24x10^{-13}	1.24x10^{-14}
100x10^3	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}	1.24x10^{-12}
1x10^6	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
1x10^9	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}

Table A.51: CCR values for: h = 1, β = 0.0001, Γ = 1x10^3, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.52: CCR values for: h = 2, β = 0.0001, Γ = 1x10^3, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.53: CCR values for: h = 3, β = 0.0001, Γ = 1x10^3, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-11}	1.24x10^{-12}	1.24x10^{-13}	1.24x10^{-14}
100x10^3	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}	1.24x10^{-12}
1x10^6	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
1x10^9	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}

Table A.54: CCR values for: h = 4, β = 0.0001, Γ = 1x10^3, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-11}	1.24x10^{-12}	1.24x10^{-13}	1.24x10^{-14}
100x10^3	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}	1.24x10^{-12}
1x10^6	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
1x10^9	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}

Table A.55: CCR values for: h = 5, β = 0.0001, Γ = 1x10^3, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-11}	1.24x10^{-12}	1.24x10^{-13}	1.24x10^{-14}
100x10^3	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}	1.24x10^{-12}
1x10^6	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
1x10^9	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}

Table A.56: CCR values for: h = 1, β = 0.001, Γ = 1x10^3, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.57: CCR values for: h = 2, β = 0.001, Γ = 1x10^3, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.58: CCR values for: h = 3, β = 0.001, Γ = 1x10^3, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-11}	1.24x10^{-12}	1.24x10^{-13}	1.24x10^{-14}
100x10^3	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}	1.24x10^{-12}
1x10^6	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
1x10^9	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}

Table A.59: CCR values for: h = 4, β = 0.001, Γ = 1x10^3, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-11}	1.24x10^{-12}	1.24x10^{-13}	1.24x10^{-14}
100x10^3	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}	1.24x10^{-12}
1x10^6	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
1x10^9	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}

Table A.60: CCR values for: h = 5, β = 0.001, Γ = 1x10^3, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-11}	1.24x10^{-12}	1.24x10^{-13}	1.24x10^{-14}
100x10^3	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}	1.24x10^{-12}
1x10^6	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
1x10^9	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}

Table A.61: CCR values for: h = 1, β = 0, Γ = 1x10^6, F = 1x10^6

E / C	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.62: CCR values for: h = 2, β = 0, Γ = 1x10^6, F = 1x10^6

E / C	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.63: CCR values for: h = 3, β = 0, Γ = 1x10^6, F = 1x10^6

E / C	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table A.64: CCR values for: h = 4, β = 0, Γ = 1x10^6, F = 1x10^6

E / C	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table A.65: CCR values for: h = 5, β = 0, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table A.66: CCR values for: h = 1, β = 0.0001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.67: CCR values for: h = 2, β = 0.0001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.68: CCR values for: h = 3, β = 0.0001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table A.69: CCR values for: h = 4, β = 0.0001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table A.70: CCR values for: h = 5, β = 0.0001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table A.71: CCR values for: h = 1, β = 0.001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.72: CCR values for: h = 2, β = 0.001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.73: CCR values for: h = 3, β = 0.001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table A.74: CCR values for: h = 4, β = 0.001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table A.75: CCR values for: h = 5, β = 0.001, Γ = 1x10^6, F = 1x10^6

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-08}	1.24x10^{-09}	1.24x10^{-10}	1.24x10^{-11}
100x10^3	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}	1.24x10^{-09}
1x10^6	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
1x10^9	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}

Table A.76: CCR values for: h = 1, β = 0, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
100x10^3	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}	1.25x10^{-06}
1x10^6	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}
1x10^9	1.25x10^{+01}	1.25x10^{+00}	1.25x10^{-01}	1.25x10^{-02}

Table A.77: CCR values for: h = 2, β = 0, Γ = 1x10^9, F = 1x10^6

E / C	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
100x10^3	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}	1.25x10^{-06}
1x10^6	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}
1x10^9	1.25x10^{+01}	1.25x10^{+00}	1.25x10^{-01}	1.25x10^{-02}

Table A.78: CCR values for: h = 3, β = 0, Γ = 1x10^9, F = 1x10^6

E / C	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
100x10^3	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}	1.24x10^{-06}
1x10^6	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}
1x10^9	1.24x10^{+01}	1.24x10^{+00}	1.24x10^{-01}	1.24x10^{-02}

Table A.79: CCR values for: h = 4, β = 0, Γ = 1x10^9, F = 1x10^6

E / C	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
100x10^3	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}	1.24x10^{-06}
1x10^6	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}
1x10^9	1.24x10^{+01}	1.24x10^{+00}	1.24x10^{-01}	1.24x10^{-02}

Table A.80: CCR values for: h = 5, β = 0, Γ = 1x10^9, F = 1x10^6

E / C	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.24x10^{-05}	1.24x10^{-06}	1.24x10^{-07}	1.24x10^{-08}
100x10^3	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}	1.24x10^{-06}
1x10^6	1.24x10^{-02}	1.24x10^{-03}	1.24x10^{-04}	1.24x10^{-05}
1x10^9	1.24x10^{+01}	1.24x10^{+00}	1.24x10^{-01}	1.24x10^{-02}

Table A.81: CCR values for: h = 1, β = 0.0001, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.23x10^{-05}	1.23x10^{-06}	1.23x10^{-07}	1.23x10^{-08}
100x10^3	1.23x10^{-03}	1.23x10^{-04}	1.23x10^{-05}	1.23x10^{-06}
1x10^6	1.23x10^{-02}	1.23x10^{-03}	1.23x10^{-04}	1.23x10^{-05}
1x10^9	1.23x10^{+01}	1.23x10^{+00}	1.23x10^{-01}	1.23x10^{-02}

Table A.82: CCR values for: h = 2, β = 0.0001, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.22x10^{-05}	1.22x10^{-06}	1.22x10^{-07}	1.22x10^{-08}
100x10^3	1.22x10^{-03}	1.22x10^{-04}	1.22x10^{-05}	1.22x10^{-06}
1x10^6	1.22x10^{-02}	1.22x10^{-03}	1.22x10^{-04}	1.22x10^{-05}
1x10^9	1.22x10^{+01}	1.22x10^{+00}	1.22x10^{-01}	1.22x10^{-02}

Table A.83: CCR values for: h = 3, β = 0.0001, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.20x10^{-05}	1.20x10^{-06}	1.20x10^{-07}	1.20x10^{-08}
100x10^3	1.20x10^{-03}	1.20x10^{-04}	1.20x10^{-05}	1.20x10^{-06}
1x10^6	1.20x10^{-02}	1.20x10^{-03}	1.20x10^{-04}	1.20x10^{-05}
1x10^9	1.20x10^{+01}	1.20x10^{+00}	1.20x10^{-01}	1.20x10^{-02}

Table A.84: CCR values for: h = 4, β = 0.0001, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.18x10^{-05}	1.18x10^{-06}	1.18x10^{-07}	1.18x10^{-08}
100x10^3	1.18x10^{-03}	1.18x10^{-04}	1.18x10^{-05}	1.18x10^{-06}
1x10^6	1.18x10^{-02}	1.18x10^{-03}	1.18x10^{-04}	1.18x10^{-05}
1x10^9	1.18x10^{+01}	1.18x10^{+00}	1.18x10^{-01}	1.18x10^{-02}

Table A.85: CCR values for: h = 5, β = 0.0001, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.17x10^{-05}	1.17x10^{-06}	1.17x10^{-07}	1.17x10^{-08}
100x10^3	1.17x10^{-03}	1.17x10^{-04}	1.17x10^{-05}	1.17x10^{-06}
1x10^6	1.17x10^{-02}	1.17x10^{-03}	1.17x10^{-04}	1.17x10^{-05}
1x10^9	1.17x10^{+01}	1.17x10^{+00}	1.17x10^{-01}	1.17x10^{-02}

Table A.86: CCR values for: h = 1, β = 0.001, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.11x10^{-05}	1.11x10^{-06}	1.11x10^{-07}	1.11x10^{-08}
100x10^3	1.11x10^{-03}	1.11x10^{-04}	1.11x10^{-05}	1.11x10^{-06}
1x10^6	1.11x10^{-02}	1.11x10^{-03}	1.11x10^{-04}	1.11x10^{-05}
1x10^9	1.11x10^{+01}	1.11x10^{+00}	1.11x10^{-01}	1.11x10^{-02}

Table A.87: CCR values for: h = 2, β = 0.001, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	9.98x10^{-06}	9.98x10^{-07}	9.98x10^{-08}	9.98x10^{-09}
100x10^3	9.98x10^{-04}	9.98x10^{-05}	9.98x10^{-06}	9.98x10^{-07}
1x10^6	9.98x10^{-03}	9.98x10^{-04}	9.98x10^{-05}	9.98x10^{-06}
1x10^9	9.98x10^{+00}	9.98x10^{-01}	9.98x10^{-02}	9.98x10^{-03}

Table A.88: CCR values for: h = 3, β = 0.001, Γ = 1x10^9, F = 1x10^6

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	9.06x10^{-06}	9.06x10^{-07}	9.06x10^{-08}	9.06x10^{-09}
100x10^3	9.06x10^{-04}	9.06x10^{-05}	9.06x10^{-06}	9.06x10^{-07}
1x10^6	9.06x10^{-03}	9.06x10^{-04}	9.06x10^{-05}	9.06x10^{-06}
1x10^9	9.06x10^{+00}	9.06x10^{-01}	9.06x10^{-02}	9.06x10^{-03}

Table A.89: CCR values for: h = 4, β = 0.001, Γ = 1x10^{9}, F = 1x10^{6}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	8.30x10^{-06}	8.30x10^{-07}	8.30x10^{-08}	8.30x10^{-09}
100x10^{3}	8.30x10^{-04}	8.30x10^{-05}	8.30x10^{-06}	8.30x10^{-07}
1x10^{6}	8.30x10^{-03}	8.30x10^{-04}	8.30x10^{-05}	8.30x10^{-06}
1x10^{9}	8.30x10^{+00}	8.30x10^{-01}	8.30x10^{-02}	8.30x10^{-03}

Table A.90: CCR values for: h = 5, β = 0.001, Γ = 1x10^{9}, F = 1x10^{6}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	7.66x10^{-06}	7.66x10^{-07}	7.66x10^{-08}	7.66x10^{-09}
100x10^{3}	7.66x10^{-04}	7.66x10^{-05}	7.66x10^{-06}	7.66x10^{-07}
1x10^{6}	7.66x10^{-03}	7.66x10^{-04}	7.66x10^{-05}	7.66x10^{-06}
1x10^{9}	7.66x10^{+00}	7.66x10^{-01}	7.66x10^{-02}	7.66x10^{-03}

Table A.91: CCR values for: h = 1, β = 0, Γ = 1x10^{3}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^{3}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^{6}	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^{9}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.92: CCR values for: h = 2, β = 0, Γ = 1x10^{3}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^{3}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^{6}	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^{9}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.93: CCR values for: h = 3, β = 0, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.94: CCR values for: h = 4, β = 0, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.95: CCR values for: h = 5, β = 0, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.96: CCR values for: h = 1, β = 0.0001, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.97: CCR values for: h = 2, β = 0.0001, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.98: CCR values for: h = 3, β = 0.0001, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.99: CCR values for: h = 4, β = 0.0001, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.100: CCR values for: h = 5, β = 0.0001, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.101: CCR values for: h = 1, β = 0.001, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.102: CCR values for: h = 2, β = 0.001, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.103: CCR values for: h = 3, β = 0.001, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.104: CCR values for: h = 4, β = 0.001, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.105: CCR values for: h = 5, β = 0.001, Γ = 1x10^3, F = 1x10^9

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.106: CCR values for: h = 1, β = 0, Γ = 1x10^6, F = 1x10^9

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.107: CCR values for: h = 2, β = 0, Γ = 1x10^6, F = 1x10^9

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.108: CCR values for: h = 3, β = 0, Γ = 1x10^6, F = 1x10^9

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.109: CCR values for: h = 4, β = 0, Γ = 1x10^6, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^-11	1.25x10^-12	1.25x10^-13	1.25x10^-14
100x10^3	1.25x10^-09	1.25x10^-10	1.25x10^-11	1.25x10^-12
1x10^6	1.25x10^-08	1.25x10^-09	1.25x10^-10	1.25x10^-11
1x10^9	1.25x10^-05	1.25x10^-06	1.25x10^-07	1.25x10^-08

Table A.110: CCR values for: h = 5, β = 0, Γ = 1x10^6, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^-11	1.25x10^-12	1.25x10^-13	1.25x10^-14
100x10^3	1.25x10^-09	1.25x10^-10	1.25x10^-11	1.25x10^-12
1x10^6	1.25x10^-08	1.25x10^-09	1.25x10^-10	1.25x10^-11
1x10^9	1.25x10^-05	1.25x10^-06	1.25x10^-07	1.25x10^-08

Table A.111: CCR values for: h = 1, β = 0.0001, Γ = 1x10^6, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^-11	1.25x10^-12	1.25x10^-13	1.25x10^-14
100x10^3	1.25x10^-09	1.25x10^-10	1.25x10^-11	1.25x10^-12
1x10^6	1.25x10^-08	1.25x10^-09	1.25x10^-10	1.25x10^-11
1x10^9	1.25x10^-05	1.25x10^-06	1.25x10^-07	1.25x10^-08

Table A.112: CCR values for: h = 2, β = 0.0001, Γ = 1x10^6, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^-11	1.25x10^-12	1.25x10^-13	1.25x10^-14
100x10^3	1.25x10^-09	1.25x10^-10	1.25x10^-11	1.25x10^-12
1x10^6	1.25x10^-08	1.25x10^-09	1.25x10^-10	1.25x10^-11
1x10^9	1.25x10^-05	1.25x10^-06	1.25x10^-07	1.25x10^-08

Table A.113: CCR values for: h = 3, β = 0.0001, Γ = 1x10^6, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.114: CCR values for: h = 4, β = 0.0001, Γ = 1x10^6, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.115: CCR values for: h = 5, β = 0.0001, Γ = 1x10^6, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.116: CCR values for: h = 1, β = 0.001, Γ = 1x10^6, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.117: CCR values for: h = 2, β = 0.001, Γ = 1x10^{6}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^{3}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^{6}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^{9}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.118: CCR values for: h = 3, β = 0.001, Γ = 1x10^{6}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^{3}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^{6}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^{9}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.119: CCR values for: h = 4, β = 0.001, Γ = 1x10^{6}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^{3}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^{6}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^{9}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.120: CCR values for: h = 5, β = 0.001, Γ = 1x10^{6}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^{3}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^{6}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^{9}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.121: CCR values for: h = 1, β = 0, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.122: CCR values for: h = 2, β = 0, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.123: CCR values for: h = 3, β = 0, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.124: CCR values for: h = 4, β = 0, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.125: CCR values for: h = 5, β = 0, Γ = 1x10^{9}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^{3}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^{6}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^{9}	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.126: CCR values for: h = 1, β = 0.0001, Γ = 1x10^{9}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^{3}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^{6}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^{9}	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.127: CCR values for: h = 2, β = 0.0001, Γ = 1x10^{9}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^{3}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^{6}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^{9}	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.128: CCR values for: h = 3, β = 0.0001, Γ = 1x10^{9}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^{3}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^{6}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^{9}	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.129: CCR values for: h = 4, β = 0.0001, Γ = 1x10^{9}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^{3}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^{6}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^{9}	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.130: CCR values for: h = 5, β = 0.0001, Γ = 1x10^{9}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^{3}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^{6}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^{9}	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.131: CCR values for: h = 1, β = 0.001, Γ = 1x10^{9}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^{3}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^{6}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^{9}	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.132: CCR values for: h = 2, β = 0.001, Γ = 1x10^{9}, F = 1x10^{9}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^{3}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^{6}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^{9}	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.133: CCR values for: h = 3, β = 0.001, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.134: CCR values for: h = 4, β = 0.001, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.135: CCR values for: h = 5, β = 0.001, Γ = 1x10^9, F = 1x10^9

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
100x10^3	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}	1.25x10^{-09}
1x10^6	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}
1x10^9	1.25x10^{-02}	1.25x10^{-03}	1.25x10^{-04}	1.25x10^{-05}

Table A.136: CCR values for: h = 1, β = 0, Γ = 1x10^3, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-17}	1.25x10^{-18}	1.25x10^{-19}	1.25x10^{-20}
100x10^3	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}	1.25x10^{-18}
1x10^6	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
1x10^9	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}

Table A.137: CCR values for: h = 2, β = 0, Γ = $1x10^3$, F = $1x10^{12}$

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-17}$	$1.25x10^{-18}$	$1.25x10^{-19}$	$1.25x10^{-20}$
$100x10^3$	$1.25x10^{-15}$	$1.25x10^{-16}$	$1.25x10^{-17}$	$1.25x10^{-18}$
$1x10^6$	$1.25x10^{-14}$	$1.25x10^{-15}$	$1.25x10^{-16}$	$1.25x10^{-17}$
$1x10^9$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$

Table A.138: CCR values for: h = 3, β = 0, Γ = $1x10^3$, F = $1x10^{12}$

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-17}$	$1.25x10^{-18}$	$1.25x10^{-19}$	$1.25x10^{-20}$
$100x10^3$	$1.25x10^{-15}$	$1.25x10^{-16}$	$1.25x10^{-17}$	$1.25x10^{-18}$
$1x10^6$	$1.25x10^{-14}$	$1.25x10^{-15}$	$1.25x10^{-16}$	$1.25x10^{-17}$
$1x10^9$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$

Table A.139: CCR values for: h = 4, β = 0, Γ = $1x10^3$, F = $1x10^{12}$

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-17}$	$1.25x10^{-18}$	$1.25x10^{-19}$	$1.25x10^{-20}$
$100x10^3$	$1.25x10^{-15}$	$1.25x10^{-16}$	$1.25x10^{-17}$	$1.25x10^{-18}$
$1x10^6$	$1.25x10^{-14}$	$1.25x10^{-15}$	$1.25x10^{-16}$	$1.25x10^{-17}$
$1x10^9$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$

Table A.140: CCR values for: h = 5, β = 0, Γ = $1x10^3$, F = $1x10^{12}$

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-17}$	$1.25x10^{-18}$	$1.25x10^{-19}$	$1.25x10^{-20}$
$100x10^3$	$1.25x10^{-15}$	$1.25x10^{-16}$	$1.25x10^{-17}$	$1.25x10^{-18}$
$1x10^6$	$1.25x10^{-14}$	$1.25x10^{-15}$	$1.25x10^{-16}$	$1.25x10^{-17}$
$1x10^9$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$

Table A.141: CCR values for: h = 1, β = 0.0001, Γ = 1x10^3, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-17}	1.25x10^{-18}	1.25x10^{-19}	1.25x10^{-20}
100x10^3	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}	1.25x10^{-18}
1x10^6	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
1x10^9	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}

Table A.142: CCR values for: h = 2, β = 0.0001, Γ = 1x10^3, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-17}	1.25x10^{-18}	1.25x10^{-19}	1.25x10^{-20}
100x10^3	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}	1.25x10^{-18}
1x10^6	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
1x10^9	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}

Table A.143: CCR values for: h = 3, β = 0.0001, Γ = 1x10^3, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-17}	1.25x10^{-18}	1.25x10^{-19}	1.25x10^{-20}
100x10^3	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}	1.25x10^{-18}
1x10^6	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
1x10^9	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}

Table A.144: CCR values for: h = 4, β = 0.0001, Γ = 1x10^3, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-17}	1.25x10^{-18}	1.25x10^{-19}	1.25x10^{-20}
100x10^3	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}	1.25x10^{-18}
1x10^6	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
1x10^9	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}

Table A.145: CCR values for: h = 5, β = 0.0001, Γ = 1x10^3, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-17}	1.25x10^{-18}	1.25x10^{-19}	1.25x10^{-20}
100x10^3	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}	1.25x10^{-18}
1x10^6	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
1x10^9	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}

Table A.146: CCR values for: h = 1, β = 0.001, Γ = 1x10^3, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-17}	1.25x10^{-18}	1.25x10^{-19}	1.25x10^{-20}
100x10^3	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}	1.25x10^{-18}
1x10^6	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
1x10^9	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}

Table A.147: CCR values for: h = 2, β = 0.001, Γ = 1x10^3, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-17}	1.25x10^{-18}	1.25x10^{-19}	1.25x10^{-20}
100x10^3	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}	1.25x10^{-18}
1x10^6	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
1x10^9	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}

Table A.148: CCR values for: h = 3, β = 0.001, Γ = 1x10^3, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-17}	1.25x10^{-18}	1.25x10^{-19}	1.25x10^{-20}
100x10^3	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}	1.25x10^{-18}
1x10^6	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
1x10^9	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}

Table A.149: CCR values for: h = 4, β = 0.001, Γ = 1x10^3, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^-17	1.25x10^-18	1.25x10^-19	1.25x10^-20
100x10^3	1.25x10^-15	1.25x10^-16	1.25x10^-17	1.25x10^-18
1x10^6	1.25x10^-14	1.25x10^-15	1.25x10^-16	1.25x10^-17
1x10^9	1.25x10^-11	1.25x10^-12	1.25x10^-13	1.25x10^-14

Table A.150: CCR values for: h = 5, β = 0.001, Γ = 1x10^3, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^-17	1.25x10^-18	1.25x10^-19	1.25x10^-20
100x10^3	1.25x10^-15	1.25x10^-16	1.25x10^-17	1.25x10^-18
1x10^6	1.25x10^-14	1.25x10^-15	1.25x10^-16	1.25x10^-17
1x10^9	1.25x10^-11	1.25x10^-12	1.25x10^-13	1.25x10^-14

Table A.151: CCR values for: h = 1, β = 0, Γ = 1x10^6, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^-14	1.25x10^-15	1.25x10^-16	1.25x10^-17
100x10^3	1.25x10^-12	1.25x10^-13	1.25x10^-14	1.25x10^-15
1x10^6	1.25x10^-11	1.25x10^-12	1.25x10^-13	1.25x10^-14
1x10^9	1.25x10^-08	1.25x10^-09	1.25x10^-10	1.25x10^-11

Table A.152: CCR values for: h = 2, β = 0, Γ = 1x10^6, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^12** (~server-class core) RLR = (277.78, 62,500)	**10x10^12** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^-14	1.25x10^-15	1.25x10^-16	1.25x10^-17
100x10^3	1.25x10^-12	1.25x10^-13	1.25x10^-14	1.25x10^-15
1x10^6	1.25x10^-11	1.25x10^-12	1.25x10^-13	1.25x10^-14
1x10^9	1.25x10^-08	1.25x10^-09	1.25x10^-10	1.25x10^-11

Table A.153: CCR values for: h = 3, $\beta = 0$, $\Gamma = 1x10^6$, F = $1x10^{12}$

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-14}$	$1.25x10^{-15}$	$1.25x10^{-16}$	$1.25x10^{-17}$
$100x10^3$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$	$1.25x10^{-15}$
$1x10^6$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$
$1x10^9$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$

Table A.154: CCR values for: h = 4, $\beta = 0$, $\Gamma = 1x10^6$, F = $1x10^{12}$

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-14}$	$1.25x10^{-15}$	$1.25x10^{-16}$	$1.25x10^{-17}$
$100x10^3$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$	$1.25x10^{-15}$
$1x10^6$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$
$1x10^9$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$

Table A.155: CCR values for: h = 5, $\beta = 0$, $\Gamma = 1x10^6$, F = $1x10^{12}$

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-14}$	$1.25x10^{-15}$	$1.25x10^{-16}$	$1.25x10^{-17}$
$100x10^3$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$	$1.25x10^{-15}$
$1x10^6$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$
$1x10^9$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$

Table A.156: CCR values for: h = 1, $\beta = 0.0001$, $\Gamma = 1x10^6$, F = $1x10^{12}$

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-14}$	$1.25x10^{-15}$	$1.25x10^{-16}$	$1.25x10^{-17}$
$100x10^3$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$	$1.25x10^{-15}$
$1x10^6$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$
$1x10^9$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$

Table A.157: CCR values for: h = 2, β = 0.0001, Γ = 1x10^6, F = 1x10^{12}

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.158: CCR values for: h = 3, β = 0.0001, Γ = 1x10^6, F = 1x10^{12}

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.159: CCR values for: h = 4, β = 0.0001, Γ = 1x10^6, F = 1x10^{12}

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.160: CCR values for: h = 5, β = 0.0001, Γ = 1x10^6, F = 1x10^{12}

C \ E	**10x10⁹** (~laptop-class core) RLR = (2.78, 625)	**100x10⁹** (~desktop-class core) RLR = (27.78, 6,250)	**1x10¹²** (~server-class core) RLR = (277.78, 62,500)	**10x10¹²** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.161: CCR values for: h = 1, β = 0.001, Γ = 1x10^6, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.162: CCR values for: h = 2, β = 0.001, Γ = 1x10^6, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.163: CCR values for: h = 3, β = 0.001, Γ = 1x10^6, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.164: CCR values for: h = 4, β = 0.001, Γ = 1x10^6, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^3	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^6	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^9	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.165: CCR values for: h = 5, β = 0.001, Γ = 1x10^{6}, F = 1x10^{12}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-14}	1.25x10^{-15}	1.25x10^{-16}	1.25x10^{-17}
100x10^{3}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}	1.25x10^{-15}
1x10^{6}	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
1x10^{9}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}

Table A.166: CCR values for: h = 1, β = 0, Γ = 1x10^{9}, F = 1x10^{12}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^{3}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^{6}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^{9}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.167: CCR values for: h = 2, β = 0, Γ = 1x10^{9}, F = 1x10^{12}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^{3}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^{6}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^{9}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.168: CCR values for: h = 3, β = 0, Γ = 1x10^{9}, F = 1x10^{12}

C \ E	**10x10^{9}** (~laptop-class core) RLR = (2.78, 625)	**100x10^{9}** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^{3}	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^{3}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^{6}	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^{9}	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.169: CCR values for: h = 4, β = 0, Γ = 1x10^9, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.170: CCR values for: h = 5, β = 0, Γ = 1x10^9, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.171: CCR values for: h = 1, β = 0.0001, Γ = 1x10^9, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.172: CCR values for: h = 2, β = 0.0001, Γ = 1x10^9, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.173: CCR values for: h = 3, β = 0.0001, Γ = 1x10^9, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.174: CCR values for: h = 4, β = 0.0001, Γ = 1x10^9, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.175: CCR values for: h = 5, β = 0.0001, Γ = 1x10^9, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.176: CCR values for: h = 1, β = 0.001, Γ = 1x10^9, F = 1x10^{12}

C \ E	**10x10^9** (~laptop-class core) RLR = (2.78, 625)	**100x10^9** (~desktop-class core) RLR = (27.78, 6,250)	**1x10^{12}** (~server-class core) RLR = (277.78, 62,500)	**10x10^{12}** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
1x10^3	1.25x10^{-11}	1.25x10^{-12}	1.25x10^{-13}	1.25x10^{-14}
100x10^3	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}	1.25x10^{-12}
1x10^6	1.25x10^{-08}	1.25x10^{-09}	1.25x10^{-10}	1.25x10^{-11}
1x10^9	1.25x10^{-05}	1.25x10^{-06}	1.25x10^{-07}	1.25x10^{-08}

Table A.177: CCR values for: h = 2, β = 0.001, Γ = $1x10^{9}$, F = $1x10^{12}$

C \ E	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$
$100x10^3$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$	$1.25x10^{-12}$
$1x10^6$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
$1x10^9$	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$

Table A.178: CCR values for: h = 3, β = 0.001, Γ = $1x10^{9}$, F = $1x10^{12}$

C \ E	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$
$100x10^3$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$	$1.25x10^{-12}$
$1x10^6$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
$1x10^9$	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$

Table A.179: CCR values for: h = 4, β = 0.001, Γ = $1x10^{9}$, F = $1x10^{12}$

C \ E	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$
$100x10^3$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$	$1.25x10^{-12}$
$1x10^6$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
$1x10^9$	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$

Table A.180: CCR values for: h = 5, β = 0.001, Γ = $1x10^{9}$, F = $1x10^{12}$

C \ E	**$10x10^9$** (~laptop-class core) RLR = (2.78, 625)	**$100x10^9$** (~desktop-class core) RLR = (27.78, 6,250)	**$1x10^{12}$** (~server-class core) RLR = (277.78, 62,500)	**$10x10^{12}$** (~8-core server-class multiprocessor) RLR = (2,777.78, 625,000)
$1x10^3$	$1.25x10^{-11}$	$1.25x10^{-12}$	$1.25x10^{-13}$	$1.25x10^{-14}$
$100x10^3$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$	$1.25x10^{-12}$
$1x10^6$	$1.25x10^{-08}$	$1.25x10^{-09}$	$1.25x10^{-10}$	$1.25x10^{-11}$
$1x10^9$	$1.25x10^{-05}$	$1.25x10^{-06}$	$1.25x10^{-07}$	$1.25x10^{-08}$

Appendix B: TACC Applications

Program	Execution Time			Input Data Size			Output Data Size		
	min	max	avg	min	max	avg	min	max	avg
arps_mpi	13	43173	868.4114	0	4508122288	712041713.2615	0	25987762520	851236847.9854
vasp	4	172816	3907.3432	0	63379684305	269110042.4178	0	56721422917	470521473.1160
vasp_std	13	172829	13810.9119	385966	47905785628	1121059333.6956	162168	49878442746	1879300416.4151
namd2	13	114410	5411.0200	711675	176958795206	1509981900.6826	159552	225607996570	9254536246.4557
pw.x	11	172822	7599.2143	0	5119136639090	11827307826.8310	0	8194169943290	166841872738.1632
vasp_std_v	9	172804	16636.1513	691329	46975891915	1050877626.2228	237568	112751878015	2050233645.1387
wrf.exe	8	127817	6365.8994	0	1797469690650	14492433152.0879	0	2859221971030	106132193448.3312
lmp_stampe	7	172829	41855.4833	823015	20880878332	1813657214.7667	126816	365097155328	24828674447.0167
vasp.535.s	9	172828	13026.8369	15683629	37837990425	1027102231.4170	688200	45727945272	1906749514.6873
S4	11	43642	1327.8301	3284873	716366481	25997737.7765	385640	827741004	26316398.6014
vasp_5.2.1	11	172812	28830.3886	14782699	3083235876	545880280.4251	249488	4069679359	791628502.5749
da_wrfvar.	14	35826	196.7519	49085151	79652673455	23275690843.0012	589024	968039036761	1890890941.6733
vasp_ncl	12	172828	21234.4830	1310375	18815566485	1048104144.0746	374688	46886807580	4796008396.4396
a.out	9	172827	5463.6538	0	4912205291040	118253905452.0687	0	1227457353760	19660995886.5522
vasp.5.3.3	17	25456	2954.0037	50487423	910451116	104819195.0902	755624	998710100	65626072.6961
arpsenkf_m	18	6414	346.1484	0	60366236180	14452464494.0703	0	53487098707	14220679973.8809
cosmomc	17	158040	24028.5099	1026718120	72927723402	1830351138.0395	1131200	47969072167	520876055.4289
charmm	7	172829	79198.3446	31155179	22681756938	6803984718.4863	256024	44223612805	21072002687.6089
vasp_ncl_v	26	172827	18340.9703	1576193	34404609750	1280755641.9840	729171	36505418812	1326892301.3959
mpipycitco	12	36027	3548.7365	9781243	2385035758000	350887656721.4141	4111235	948093302324	51577627591.6659
parallel	15	86425	27205.1531	953135	12187869934	2377100161.0981	406784	18236703280	3554830508.5263
us_mpi_ana	11	23459	896.7857	13356168	2634314202	95735350.1878	1471660	2960822883	71890858.3095
vasp_mpi	18	168170	13240.6704	36520503	9214056828	1088438261.4127	1452080	17355012017	2064238974.8227
orion2	9	172805	14234.6863	8224567	43570289942	7355606252.2241	242064	1184327617890	38802958114.2241
plascomcm	20	50421	2277.1345	26517113	3214900816	286643505.8041	3018500	7402559362	310947963.3889
vasp533_st	12	172829	73379.4507	67092982	26843454826	2654275431.0687	315632	40782527754	4019235672.8746
gsi.exe	39	115	78.8997	7356827520	27158667508	22406394212.6474	1958172330	5673414342	4423696372.9362
loopmodel.	27	33958	9930.3518	216848725	2355941698	1288271469.3941	2982917	1582551132	537724207.4430
vasp.5.3.3	9	47535	10191.8926	44153879	16477292451	1426035867.5034	330072	26735480620	2337441526.1980
lmp_stampe	17	126900	48788.2793	88766509	1139350735	494114357.1897	304844	1282328523	495322130.4862
flash4	12	172517	15681.9577	9015693	132652607806	13050162171.1373	941416	1599225796180	78045789589.5739
lmp_fixed_	17	86415	17332.8764	18092426	8660101968	908078714.4364	901712	109101847270	30460580556.4509
inverse_pr	75	7206	553.6468	15000929	27993071805	1544316981.4365	2898336	65084220279	8142089667.7103
vasp.5.3-v	15	36028	4944.2090	21658140	849033525	190770374.7623	639792	2586135580	339992300.7664
lchgall_NV	21	148952	115989.5732	24324805	18870368459	14784804448.0377	5522352	38787038679	30902961232.1297
nwchem	18	172215	12335.9156	100382194	20941986928	1050572358.8776	353608	31894864051	2561583106.9241
fvcom	13	76041	13902.4211	11823734	9213904459	1523109763.7544	580632	1777243152720	385569085780.1886
vasp533_st	10	172823	32104.1187	84230946	4067492811	773280245.7443	1085264	5975793824	1140013302.2740
main.cpu	13	7222	465.8269	0	5352653008	357285610.7452	0	3026998126	22053126.2163
helium.pro	10	3611	88.4478	5524893	34867255	6583639.7214	772694	34898063	2551899.4080
lbm	12	18025	5681.5253	8341497	21922687408	5723834598.8182	610712	5936659375	1247015195.1667
python	6	17480	342.0497	73648	155582109467	1446361140.9503	75552	137084555157	1373005803.8122
scf	11	4611	238.4778	7410308	12825884963	1061763386.6889	141176	10017210215	541185691.2167
vasp_533_i	16	172828	81501.3333	92953981	2932144586	1430395524.4138	1439736	8726496234	3040662372.4713
mdrun_mpi	10	172809	39601.8218	3490075	68789049041	3829654817.7069	759327	151644239465	9653478440.7241
grins	15	66791	2673.8333	43830250	19134548663	1761146018.2126	1277547	1104818517810	15962656807.0172
siesta	11	172829	45056.2428	2352628	7146867398	1033925670.2370	164824	1116837496200	92699935741.8613
cpmd.x	44	67068	51059.5767	28978543	3715115642	2707142246.3190	50145257	980221848001	25871386470.4908
cdp_if2	10	54021	14823.0613	8325365	35799486985	7213172595.1227	881435	206142860780	52482476212.7178
real.exe	11	48772	1611.2407	40312698	423994811963	29660284513.6358	1056315	423927200684	25337173512.2840
vasp.5.2.1	17	86430	45215.6605	94709877	4033248981	1291789840.5432	2018240	18634134404	8153653680.7593
diffusion	19	1828	250.3375	68892168	275441881882	4865320345.8750	14808568	28548800151	1449888250.0125
vasp.535.s	16	172806	19799.3205	18082337	7119848212	1106580565.9487	2401624	10607547193	1894742700.9103
enzo.exe	11	172828	22019.0641	8178587	1045158809170	14906759747.7372	254040	495043818874	51634779419.2244
UT-AIM1	17	43218	1453.9739	6184958	1847560814	87999392.2549	1278938	1852003214	77120526.6601
exos	710	85089	47123.7987	62839614	7526794707	3419882242.1611	397884422	26039484960	14396827655.1745
relion_ref	18	115444	15779.5772	24471534	22065337482400	2020443300958.8792	1156544	1706522248770	204484790842.1611
TTMMDMPI	11	83186	12680.3014	2641109	13568079704	983926651.8904	220656	130240678011	17291878406.5411
cDNS.x	16	10474	429.5278	14241333	5764433998	246135087.8750	1269352	135792485772	2903398000.7083
domain	11	35503	1779.3007	12192966	419081318	44716705.3357	423296	9364284409	614921473.1329
preps2d_p8	22	300	38.4685	5213666	21277060783	1033503409.8112	1150020	148193746	19929613.3427
CitcomSFul	9	188	15.1844	3166384	28887415	5006040.8936	254720	22096905552	171869473.6950
python2.7	10	172812	6986.3357	11428623	2082599194720	207585489159.3714	3686232	9856631379	1110821865.5000
cp2k.popt	23	172819	4979.0791	132438017	40332929937	1422001876.5396	9388178	507969751992	5823266980.0719
mfix.exe	12	172824	44043.9407	12281759	7170800425	1177113330.0519	2838314	39314918740	5867040667.1704
vasp_tensi	368	75314	24901.8947	85566033	2632042892	924560840.6241	18697643	3178877513	1066333877.9549

vasp533_st	21	171029	61211.9147	93333435	6441162387	1618611822.0543	1832743	10059812528	2452302592.6202
mdrun_sm	17	21096	16067.2500	37344920	5022320426	3350819195.5859	4545728	42920198940	34577702124.3984
vasp.5.3.2	139	143209	27011.2109	66110656	6423229963	1273967456.3281	3017809	8213368292	1604364714.3125
parsec.mpi	10	76347	7174.3440	6121159	2349231873	127478085.2720	180264	18562902072	1129787575.9200
hpcc	182	2419	268.0081	2791044	314516285	25309550.1371	1922067	349602839	20296102.7581
a.exe	19	86421	21585.7295	5681231	7488621671	1301816557.0164	628296	8694991151	1092512300.7459
1D_Shock.e	12	2166	279.7107	3339188	641342226	84029279.5124	189816	1090820139	136116540.7769
fppairs_p8	24	9017	1138.8729	26227447	19391504013900	1302846066161.7712	5820640	85555802608	4168062310.2797
mhd599	14	10817	2590.9310	12138168	1983596301	740047096.9828	1440288	323809341235	19276975817.5776
lapin	27	172829	81530.2544	130772943	226685674920	27017579943.7719	28082689	4427716997620	574984293560.3684
pisoFoam	14	21618	4432.9292	66143125	6277047196	713233115.8850	1820237	66591041314	6992491515.9204
ph.x	13	172826	46420.4821	2368817	6158696604040	60237983195.8482	1444700	9973684158730	1539789868126.0000
sus	24	172825	24210.0901	752213459	333831754116	31427128585.0991	40735550	17547668498900	990400051946.0181
cactus_gig	15	86424	30212.9545	160238374	176137761563	63352177910.4273	399216	247547109767	117983806415.8545
cgs_dem	10	85283	7248.3028	0	1268591141890	121935281106.0367	0	34091811941700	3282572793659.5415
Bdd-mtx-ds	39	18032	813.1743	666849045	66118135191	10615472205.5596	746745	1599530801	36911151.1193
mitgcmuv	22	172815	53547.8868	61479689	3678846872600	116570943363.3962	14106938	495945791859	26799072570.9057
md.xx	15	172829	122756.0566	4284008	5228659736	1153907040.3302	1444258	10465126611	2625271575.2642
DNS2d_p4.x	24	26174	5402.8095	7644810	1046732704520	507313887362.3333	16400027	3720383768580	723093921362.8762
nim	24	1678	240.9901	13117238	51920953029	1637116176.4356	2269392	2034872759	163853948.3465
Xvicar3dPA	14	81185	11127.9400	8201042	73983047788	22122006967.7100	1245906	305558860684	29035453254.9900
qcff_p	9	13	10.6100	1871737	2550687	1951524.1700	3331970	3578511	3397731.0800
cesm.exe	39	105698	2500.5500	70105492	178411284410	9244908241.9400	3340234	323406350946	9228176363.6000
vp	14	57829	6578.9500	49803451	4136572911	635621822.1500	890064	17942496430	2065234659.6900
test38	21	59472	15812.4040	2461246	12109757333	3190034231.8081	8805776	3302339468090	94264448550.4949
gryfx	13	23721	10352.0202	11916027	855724605	384952478.3535	873155	39358157599	20551942836.2121
NavStk	20	11723	1176.8571	114731943	23949959972	12722673520.5918	17959825	178359374346	13931477741.0000
top_IE	17	86425	19629.4694	6823578	7189276046	1643802228.0816	2396992	245949168186	28123623340.1837
graph500_m	419	1534	454.6020	0	106645608	12595444.3878	0	113810119	10066360.8265
chan2ph8x	11	84191	23468.0722	76113075	755558332	265231240.3608	203344	793252270	223854495.0309
mpsolve	16	86429	29731.8632	0	49104342602	19852960809.6947	0	248653051820	33770098853.4526
mdrun_465_	8211	86427	23310.8947	79236517	1466182476	290447014.5895	124530593	3708238685	623705143.3895
qwalk-rang	28	18771	3030.4737	47871259	10485548515	1327674456.2211	16506982	27726162211	3936409591.3789
diagnostic	15	49	24.9565	34969185	4664543635	2146316011.8696	13702041	783483137	485863391.3804
cactus_sim	28	171141	28624.4286	317350584	350905433897	54969513048.1099	16586741	21219499894200	1927603965152.3516
astrobear	17	86423	7097.0787	11756295	107680386106	5237427638.7978	1645936	924416763815	38842482267.6180
fleds	14	172816	15579.4831	5271455	5813481046	536381198.2135	1344943	11933376078	1095275613.7191
pelfe_STAM	15	39932	3484.9770	25590037	162247068081	25770649589.1264	5392853	188157489072	16395729816.6207
reconstruc	28	1812	202.5529	57576088	3773867422	1300932210.2824	2120776	2681497042	602373843.4235
docking_pr	18	4813	2973.9059	649714268	6063660997	5225273517.5647	53445112	13531712356	8409239780.2824
mst2.1	16	45024	1853.7059	24284581	5917873763	312853118.8588	2764640	7233599458	297335889.6000
output	18	14425	4249.6220	14405040	1346524878	371812465.3171	3104410	1460326028	406336455.9024
ifortddwnt	20009	86425	22439.9877	383346229	1470612766	432197888.1481	984153610	2191514339	1036664619.3210
bayesnet	20	1821	459.8395	12220853	7687079480	1325115856.3827	3913490	2467439331	491827240.6173
lmp_tacc	9	8381	820.5125	8333743	83787186	18187177.8125	237832	136955053	15261146.5500
P-Gadget3-	21	147974	31855.2785	208456168	184890975563	130800989129.9367	17655821	4726558374720	633612115947.3038
Preproc	19	5421	714.5443	16606674	27600488523	5721156559.5190	6072208	35999415673	21295550501.6456
vasp_shear	583	64349	12516.5641	98147927	2241446548	501910356.6282	27715430	2608521195	533632969.4487
modified-a	16	36029	5555.8267	53124787	2717595296	557783270.2800	1986837	2988401557	547493381.1867
fem_mhd	8	6292	924.6400	26340911	7432042153	1523480551.6267	126128	4227366193	446459136.2933
run	7	172810	24186.9333	3434820	61254471938	8439066873.3333	1151343	138992268432	12949577086.1733
helium.pro	12	1532	137.6622	5522990	18121938	6717007.9730	805004	200905269	5404616.9324
curvdns	19	7229	1650.4324	8191783	2118333673	248666459.5946	3147128	9057798279	432967006.3784
aims.07191	64	105164	12056.1486	39557161	3604374114	462712884.7297	6787172	4963750790	589711018.5676
mas	13	86419	6495.3151	33873562	101549209006	5932644512.3836	2100071	194398268689	14089469265.6301
lmp_Jun14	17	15519	2300.1250	40410291	166518651	58439527.9167	214401	191520088	26932273.2639
docking_pr	30	87	74.6944	1630724413	3194594950	3006042208.2500	119219703	721145013	618627688.4583
parody.exe	7	1501	323.7042	2478922	6231397538	1394641237.8592	334576	28407433501	6899916384.5352
stokes_tan	23577	172828	118977.7391	195143203	2803908360	1417218017.9275	227426205	3675464440	1778936084.9130
TMDMPI	10	172827	60198.0588	4221611	23686800858	5885295318.2353	361288	373093161435	91341995152.8676
P-Gadget3	18	147577	66050.3134	42489446	319081591034	138569230012.8507	1138736	3580986076760	1114924693061.5671
hello_affi	9	14	11.1212	390971	626002	479416.6061	196864	457030	287052.0909
hello_affi	6	12	8.6667	199647	423735	267955.7273	157816	417936	236795.7576
casino	11	86427	8840.2615	18322108	12541305214	1447201811.8154	1625652	15018235429	1638294122.3692
sunfluidh.	11	20158	460.8125	5072375	298077161	32576405.0156	172729	4954694876	1109320833.9062
nektar	13	84561	4858.3594	3102921	8883389276	1438220225.7656	185804	108394451653	8478290233.0312
a.out_sph2	18	1818	167.1111	9524033	2456671579	678643950.0635	3956851	1029914969	80073388.1429
preps2d.x	24	450	72.2381	11805491	21602900882	10389364509.7778	7350253	243215474	54325522.2857
SAURON	35	172705	60433.6190	494267652	56032325279	20166814876.5238	21252738	2182955026390	727956715995.9048
agk	13	122425	35879.3710	21402284	85468386528	18219611102.8226	1422608	979664389368	368123427250.5161
GIZMO	16	55064	1892.9508	14354218	11722344271	3128468266.4754	2139614	462243573646	19421190873.3934
DSMC-pro	17	172821	49055.8361	907878289	7503111555	3227188475.8361	2456984	123510169606	42810784475.4098

SPmodel.ou	17	3315	422.8197	0	1025906650	100307969.7541	0	32710536607	5408521270.1967
afnuclear	10	30244	4777.7705	2108117	517132767	135523135.3934	46536	592768886	149533093.0656
mdrun_44	16288	86424	40741.4754	272612577	1453999676	684774562.5738	560576683	2934710777	1379297982.9672
sph_mhd	10	1207	154.5593	17304683	341249421	223356484.2712	836419	552919549	80425083.5763
gvksx	16	108022	35475.1356	23903510	16854911624	5390957934.2203	2157144	158211511664	48609172596.3390
pelfe_STAM	15	36025	10034.2712	15218600	100377855486	36236619668.8814	1129328	743846338505	226322920145.6102
fsi.exe	11	172826	31998.2069	2579869	2839333814	306216769.7414	260728	6610146201	687627078.8276
docking_pr	13	238	47.8246	197553082	284848650	266271920.7544	1293624	48234728	8437701.1053
mdrun	8	172812	36349.4737	11646111	23534624080	2446678207.7544	2281544	72111976399	5662952883.8070
buoyantBou	23	104673	9953.9464	179363771	8228135346	1396028253.0536	15622240	22919589349	3268793597.4643
helium.pro	11	86424	17104.5273	6333715	110274853959	21988093702.7273	788016	135081910668	26361262791.4000
cactus_mva	1949	83339	9904.3636	373726043	24873406624	2687020707.0727	395550037	121879668299	5481969386.2727
lat3d	21	88467	48177.1273	10138925	6878193349	4468142826.1455	4269120	110605159668	65803818945.2545
metgrid.ex	10	44470	3612.0370	18626157	330418808863	32864019087.8333	572648	1471700980860	194069332048.0926
restart2te	249	39262	6884.9444	1252030932	112198704267	25221529904.6481	282742446	41753263806	9544574303.2593
mhd	17	108009	9944.8113	12030286	7867686661	1797157142.2453	214571253	792928216168	54287400383.6981
mpmc	88	172824	10548.3269	3965110	74356706136	2189037069.9038	3214254	89250387559	2558667312.7308
UT-GMRES-F	14	26502	3447.7115	22789449	19004928332	3081702998.1346	1198256	2765245879	684597986.2308
semi_lagra	21	1115	161.7308	5858638	1810746971	235846661.5385	797048	9062731092	712030503.0000
classifypa	25	430	186.7308	390719094	19892347854	8051724389.4038	24647760	134207184	68295551.9038
optexec	37	18000	1354.6471	41244839	6857414239	1739280548.5098	14176689	12058439776	1369756112.5294
UT-ETAIM1	13	30064	3533.9608	6252359	37372369265400	829050499508.5686	1527866	25518893246100	864279005206.2745
gen_be_ens	27	55	35.6275	9514608175	9515574131	9515149905.9020	409214743	410263691	409806961.0588
drtm_mpi	12	808	97.4510	251028704	61448351539	17517317456.7059	955840	2456913648	684949039.5490
BATSRUS.ex	22	33064	9405.5000	200040375	96415014674	35846549106.2600	42719912	731745698139	149467487360.3000
gridart_in	10	352	41.5200	76512899	153436576	89231717.4000	2345444	505487633	239081430.1400
overlap_in	34	25013	2167.7600	69849021	606339653259	176875594069.8000	19913040	172688979757	25486309373.1400
polaron.ex	35493	86429	82067.5306	331506888	1041368038	902846067.0204	22911207606	175200573271	121999634907.4694
launcher	10	12415	1246.5714	1673344	8810131444	765128290.8571	649698	1995296231600	109136481652.9388
cactus_thc	25	84635	33819.7551	376294515	11778370499	5503705441.5510	82510761	51700319021	21943158916.0612
fdtdmpi	33	7246	835.2653	7710927	8327105813	1092345351.1633	3854113	10323116646	1219631991.7347
main	16	10763	2900.2083	1570239	388189401786	260093463908.5833	1379078	6105902822	1837569084.8333
run1-5th	11	86429	71002.7021	2319566	816856735	618419763.7234	16687319	50780814511	17636658349.8511
mpi-waveto	8	83	21.8936	757710	1121078294	86200075.4043	292616	730703645	56308794.1277
zdock	20	988	176.9149	1582141	540527234	60168437.5957	645175	129123791	14027909.1489
gotoh.x	13	18	15.1277	2261607	7019105	3648012.2340	166520	451480	281189.9149
gdm.exe	15	86428	84580.7234	75855694	1041468006	959194646.4468	63262703	175121565948	139441154167.4043
icoFoam	15	2299	486.7609	64663578	4012963497	1640511375.0870	2047503	117071019578	6475116945.1522
lmp_tacc11	18	31569	7447.5217	8487108	1043102126	266925103.2174	922472	1313456451	593167685.8043
swimobj	51	172829	105307.4565	62255315	16103275841	2923525825.7609	31561302	127883975145	59545386438.6087
flash3	20	7230	4354.0889	37129686	15316046475	3922210495.9333	12669919	76130141928	7424248066.0889
reflect	28	172411	67259.2000	126056329	115249705865	25835820640.0000	21240201	3543408486700	849268791598.7333
fd3d130725	2121	2429	2320.0222	72474257	88101285	80643768.4667	123317463	149161504	137961658.8889
gene_stamp	23	42417	5548.9556	62667923	9085755503	983525817.5778	8032127	52263853875	6696956957.0000
lmp_28Jun1	18	72108	3744.0682	10758293	2346803509	446581307.5682	1272208	20711567610	1556565932.0000
HH.cpu	14	134	39.8636	832621	17309205	6981614.6591	194864	14048157	5302306.1136
DCMIP_NP4_	52	10810	2620.5227	50851516	33207740921	6276923749.8182	16730521	1322175485890	845081518480.7046
5.3.5	16	11150	973.1136	27306084	1696007364	264101786.5682	412888	1698118197	206450363.5000
wqm	14	19246	8952.9318	31942100	16343901171	7680209129.1818	1250805	5180442141	2462645126.2500
gemdm	10	8535	1697.9773	20945259	38187804095	11508653077.1136	1000904	41746082205	7793690859.9773
mdrun_mpi_	14	86417	74017.5116	4538961	2856544867	2439349991.3721	851434	5450191291	4392957135.6744
padcirc	15	65297	3900.8837	56667875	134515224269	7146640378.8837	2231536	425078451802	19983403033.2558
julia	15	628	83.6667	41393337	214819719	96389706.1429	2310258	18474169	5366168.5714
xsbig_hyb	8	1101	194.0714	1178463	179650726	17551507.4286	138912	1376019715	215746823.9286
docking_pr	42	1239	473.5952	475948609	3809643957	2715668443.7619	15030261	2692208652	819203820.6905
cactus_rep	18	12347	1056.7143	228938448	181346074471	11925948361.8333	64713954	1685053172	296855500.2619
senddiag	11	4048	801.7619	994897	366231112	79434340.5238	755717	363134496	67444569.1667
spcal2d.x	14	314	51.3571	2218632	867619312	184561807.3333	881211	702684139	79066618.5238
grab_machi	17	86405	30943.0238	522908	264094122461	21924744520.0000	368240	397903164149	24742974124.1667
lmp_foo	182	28131	8142.4524	52313861	3644050778	880096982.4524	6282955	19501562798	5063451540.1667
lmp_30sep1	17	30068	3992.5854	6675571	1473665277	159998001.9756	388960	6626180638	1023882340.2683
FEDVR_TDCC	14	8152	4821.9756	14194450	3892372308	2349368853.0000	9732464	24602551358	17525307629.7561
md.exe	16	167148	25132.8000	6103288	5055403467	931711824.3750	475992	47373468403	6418596062.0500
lmp_intel	12	172829	65957.6923	7442205	1415405209	545159173.3077	382560	1714161516	680672198.8462
helium.pro	12	197	49.7436	5300795	8047691	6095097.4615	833489	2791601	1363880.5128
vasp_std_s	146	86429	42202.2051	62520579	3008538976	992823798.6667	18321398	7697044149	1934604319.5897
c38b1_xxl_	14	53763	10501.2051	34683050	974915890	143559314.0000	257504	1842280875	336529935.8718
mdrun_mpi_	10	1229	292.6842	55270792	172598625	109529761.0000	792903	355798087	85231331.8947
pmemd.cuda	19	86412	22382.9211	32066818	34094354338	2631054636.3158	1020716	16837606881200	807476325361.5790
mhdfnc599	20	144487	50716.5789	11827516	10243631381	4062635547.6053	963928	172740502495	91702412628.1053
ramses3d	22	86429	36301.9730	97582813	158194643372	32985888476.2703	16867048	2528467213350	370310948560.7568
vasp.5.2.2	156	17227	7337.5946	28981919	822933742	193832924.0811	5461538	870916785	325517937.3514

rmc	15	134411	50059.9730	5163811	3767195101	1476949144.4324	385528	4706608299	1837136659.0270
hello_worl	8	15	9.6757	168930	1026435	249316.6486	95616	429654	167207.9459
Xvicar3dPA	22	91223	29317.7222	210067992	48908571570	9392296284.7500	13918320	1123924740950	166015061067.1111
Pcrystal	23	36992	5066.9722	85862888	47641007589	3770156431.6667	36111429	543270458719	111588372160.3611
dynmat.x	10	47	21.7500	1381638	289746131	16301321.2500	586832	334956920	17471185.0833
xlmg1	16	324	84.3611	3502487	122003453	39729669.7778	971016	59071415	14077910.2500
util-par.e	19	1753	889.2286	1201293703	1005358706000	545076942236.3428	15718720	127322730944	19799064148.7714
generate_m	17	200	29.3143	6050548	207268479	40373684.2571	4132122	26305820453	4098483563.2000
nls.x	16680	172819	102523.9714	283854946	34563435898	12769346670.5143	397505442	60634773169	21067828595.0571
vfmfe-opt	13	214	66.6286	19111510	199259614	86166365.6857	18047240	964055039	412520147.6571
out	7	10040	5270.0294	826670	521956256	262826720.4412	747972	571847384	295693079.8529
LES_case	16	85529	34922.4412	37824861	2935599971	1201102638.4118	591452	93912659057	34211961214.7353
mpnrlmol_s	10	1840	410.3235	4710672	2145699815	175392121.9706	565246	1393500340	199559765.5000
precipveri	9	68	48.0588	3547648	684384810	509833955.2059	147128	95806798	32252269.5588
pp.x	25	85	32.7059	49731673	519550205	359982794.2647	17811013	459790677	312487318.2353
antibody_H	4366	40186	16197.8182	3413952940	8086049028	4952012361.4242	1681387659	8066914611	3874622714.3939
pimpleFoam	23	18029	9188.6364	150480390	2995821794	1256333791.7879	4857704	123222950347	22804400695.8182
ioChannLes	15	172827	21930.1212	6304558	12046935483	2548614912.3333	395384	2223283385040	143254743870.0303
P-Gadget3_	21	17702	1914.3030	42481555	48727147689	24712586469.0303	3826240	28484621692	8129740962.1515
vlpl3d.e	413	43205	19138.9394	229071977	42771595637	8354707440.8788	170727556	301080831660	121856613463.6667
cfd.exe	24	7219	2530.2424	53465619	1717054992	1060273454.2121	17287809	9729615831	1467911219.0303
UT-GREEN1	39	7221	1034.6970	3372547	61138352	11144135.1212	1406321	69542542	11422043.6364
acdds_scan	18	72018	22820.4545	6344331	530117139165	91003264252.9697	829880	243515435669	96221934696.3939
bonesolve	34	132144	22455.1515	6609519	7984468790	1586284302.0000	179143833	318610737839	45943200013.6667
oceanM	21	172808	35327.8438	84169697	5263995323620	846039579647.4375	15283112	11227069181900	1785316175768.0625
python-mpi	13	7209	389.9062	1578475	2133760874610	81874108290.4375	806847	8074309201	407783001.5625
mhdf	120	44542	8078.8125	542037837	4335158264	1298581967.6562	211378158	407326327551	101064340303.5000
vasp-5.2-i	96	72006	9720.4688	20421706	1359606689	330201114.8125	86223226	1622198134	593658407.1562
eppic.x	19	36019	10323.5312	16329947	6713181405	705072175.5312	817405	52359808819	7228214221.4375
vasp.535.s	32	169641	39166.2812	91896905	11546386292	2223065937.1562	1960510	13318810337	2323503642.1250
mpi	14	3617	452.0323	4914570	4359338836	158749864.1935	1124642	4152771086	150315809.1935
cactus_bbh	28	172828	49452.1935	160933641	23576855141	6830439287.4194	8815377	38616752825	12018186018.0968
pmemd.MPI	200	72361	53324.4194	19124137	647139212	481456024.2258	9209380	20433754471	15908094321.9677
pelfe_paul	24	809	448.1613	24943130	137101219533	21611658359.3548	116279644	3690127464	770055331.9032
vasp_vtst_	16	172825	77763.8710	66809954	7061510303	2770979892.9032	824104	8119124830	3164887188.0645
sander.bul	14	5429	893.2258	18281103	204853814	41005070.7419	3994960	212429415	29573721.1290
cactus_bbh	16	171047	23363.0000	140550610	58623481073	4157841519.5000	271424	605430946654	28627476356.6333
openmc	62	1546	447.4000	1370577610	426671486013	50943765098.2333	4756092	1057979547	132142530.9000
IceNine	19	70848	13362.1667	84698975	37582932198	7542446787.1000	21139808	45861867182	8547288957.0000
wilson.cpu	16	24	19.0000	2119736	17181865	8800774.7333	1563316	13985090	7039709.5667
padcswan	18	41875	4075.0667	38027312	14167172079	1255924292.0000	5428128	307436236102	16415278340.4667
fd3d140203	58	63264	19035.0690	5901535	2076114142	625753995.1034	6796325	3005789470	1116580034.2414
iPic3D	15	928	375.1724	2006628	1262593736	454580858.4138	260768	255702596872	57282321249.8621
flmf	22	2430	354.2857	30648184	67494110	37581230.1071	8584824	159378106	38552383.5357
cam	48	43229	17135.5357	353158372	6260143370	3087711574.7857	29170839	50795995964	19939327024.4643
x	19	86428	43008.1429	73402278	20143389848	5388468799.0357	3931280	88278862269	11420291578.7857
xMGP5	250	108019	38204.1071	42763698	14844706544	2973113094.8571	642743927	356174374581	75117628647.1429
2dbinner	13	65	45.1786	4077372	32921358307	21879638679.0714	1139473	25744220	17862184.2143
varOmegar8	99	619	158.6667	10771533	41486176	20317944.3333	3682470645	5432737777	4898696198.5185
qcprog.exe	47	86405	18278.6667	314970344	595406902615	84043889845.1852	62937263	5032311946240	730514598219.8889
ymir_slabs	39	9803	918.7037	43684518	804481244	217710387.5926	3966987	89468536808	7800493684.9259
cactus_whi	29	86410	76855.1111	521206429	114680722890	81685826391.5185	18674400	669562879602	470445834046.0000
cm1.exe	21	63007	7069.7778	23787746	49079637652	10429691074.2222	7484618	584074712855	86443531546.1481
global_gsi	61	121	113.6667	3794387517	5178076078	5088661979.0370	205388210	1154878517	1108747171.0370
arg-sample	13	43808	6509.9630	5233325	1498462065	173016137.9630	404998	3108691520	348001211.5556
Floating_J	35	146011	15301.8889	2610513586	54081635002	15153992811.1481	27104000	596643848646	127609385435.4074
pegasus-mp	14	172531	74545.5926	6541071	9751290999780	3128949155983.4443	1233061	2908366253470	418447516454.6296
vasp-004	5539	172824	112723.7037	110590613	1539051711	1007086820.7407	58932101	1825761308	1166654224.2593
qb	20	13233	707.1923	1022998617	19408550960	4229524322.6538	14119240	16812265011	1936874753.4231
cactus_zel	34	43228	33698.9615	1749347106	121684921036	99930690830.8846	111040368	511240139288	316713811347.2692
enzo-intel	51	2847	561.6154	96566269850	175650872390	124252972720.1154	463377292	19691628763	2318669843.2692
rosetta_sc	25	2616	652.7308	1204655742	3169967479	2005882733.8077	116584552	1830196992	796590576.6538
coawstM	31	86129	24281.0000	200474852	735897867949	196872567757.3462	30883256	2208928816840	573106317545.7307
CitcomSReg	10	623	121.3077	3021446	157675678	10472182.8846	138624	2428567582	125773070.6538
RunTrialsM	10	1691	246.3462	1850605	40327521	7222103.9615	795960	50797100	7342048.4615
global_enk	405	482	420.6538	29315441208	30038116026	29854662311.6154	9123499233	9139308263	9134446389.4615
Snac	16	172802	21504.6154	19119359	16939384286	2196895307.5769	4493592	20608963660	6344714173.6538
bssn_xmhd	58	146021	128112.0000	125574589902	156635675183	142823884662.2308	72482975	141650849722	113211669741.9231
vasp.mpi	1496	28810	12526.5600	35098660	9655317984	1209380176.1600	42715047	3421598098	524510701.2800
lmp_tacc4	22	92294	37488.0400	21976819	4510673955	1841883870.6000	5270331	6306911668	2489072094.8800
vasp.g	19	172745	8466.4800	29429291	48919116522	2174860329.6400	1060968	57025035996	2647639260.5600
sf3	11	36008	4929.2000	2809360	114321772582	62960097990.8800	196816	153075137887	48517544187.7200

vasp.4.6	21	71817	31502.8400	31243574	2357266693	1005455337.4400	1777632	2721996509	1141348202.8000
bench_gtc_	26	926	499.7600	2918883	1418829339	219592602.5200	895637	1431987593	177563167.6800
vfmfe-3d-u	176	172829	82499.0000	105021144	11346198725	2713541200.2000	22846078	19617555969	10354846408.4000
Charging	20	1639	223.5200	7570410	45808686	11391313.2800	3053837	136856905	23209759.9600
athena	17	119095	61304.4583	67192164	221229483508	39667406560.5000	15395656	702204335063	280896142123.7917
Mod3DMT_MP	19	163830	99322.1250	5092166	5290414091	2885626178.4167	233008	7138037996	3950705048.7083
cactus_bow	34	86410	43959.0417	306175896	49866140401	13549347342.0417	896511746	369278156032	139852761145.7917
CAMx.andy.	37	235	126.9583	13428506	465464698	376794969.9167	4488376	264777459	87176999.0417
solver-imp	23	1187	292.4167	116975946	1083177841	350500594.7917	29837808	2979889514	420568107.7083
MHDAM3d.Li	48	86400	12865.0000	67158505	47914353910	8635678119.7500	1474965	94963216750	19748011755.7083
bash	56	3627	1321.2917	41647355	4179774536	2006314698.0417	47315413	4818800645	2226100645.2083
sf2	403	2902	1484.0833	2537842890	109100078974	52903507735.5833	1705010254	33719928078	20367693439.6250
ring.exe	18	3812	2273.1667	4939577	137812730	59308649.0417	977822	6311287770	5766901605.4583
exec-paral	8185	86429	80593.4583	428639060	4892094398	3753206677.5833	1480224804	18492552031	13328179523.4583
dg1d	16	2904	901.3333	1573876	56517792	22750618.3333	402792	2357240962	823763695.7500
lmp_linux	13	76502	25840.2917	14524245	850712411	267338805.2083	192960	171417554989	15562563158.5000
diag_to_as	325	429	374.0417	4963046278	8993544826	6978306611.6667	1348387930	1351618169	1349956715.0000
ideal.exe	14	63	31.5000	25963730	1859206869	603413888.8333	643816	3171080490	1441034767.7917
antares	80	172826	48316.9583	361550593	10028658309	3231805527.3333	6000582	24986856926	6663880965.0417
flameletSe	23	34735	9625.9130	421537816	19145568356	13504503663.6087	4142350	7742065148	2421963884.2609
epoch2d	9	76272	19926.1739	5475364	87978450963	11032026021.7826	1008749	596153285309	83745621843.3044
namd2-cuda	67	86407	40983.6087	88348255	4439038524	2157367437.9130	9312654	80743973542	37133321571.6957
binner	32	123	82.8261	887471303	70695006283	40343534527.0870	27794622	69775154	50781503.6957
ramsesASU3	17	7207	749.6522	33458255	34921696130	5760566885.5652	7441432	236833819236	41221416315.9130
geogrid.ex	8	17	13.0870	3450876	839468268	403798484.1304	811911	211527645	90917621.3043
ssca	19	1006	248.8696	9180529250	109014375759	21214680724.0000	7245176	9724579563	2870429639.1304
vasp-5.2.1	19	86424	31628.3478	18869157	988987011	306270931.2174	580695	1598473402	392757598.6957
mpipi	15	1221	118.0870	1909801	14666416	3186629.7826	1640526	19667345	3293190.6522
vasp.533.s	17	96134	6604.2273	89185620	3443242079	289369341.6364	1701601	6997308237	431475048.0909
decomposeP	57	1626	290.5455	376495472	1688701399	1203775788.0455	372021421	1819606834	1339448618.1818
DNS2d_p4.x	111	7223	1248.6364	810430138381	889501113263	838538255422.0000	2257174909	958296142902	96142408995.5909
cactus_whi	164572	169860	169173.0000	218579988589	220984496710	220289509353.5454	916543745779	1301208773700	1261177754908.5908
P-SPH_Sep1	18	172800	37657.7273	36896456	56835173654	13653060359.6364	2345063	846336852745	202345574572.5454
Inspector_	31	1067	434.3333	33045804	38190393102300	2289994598231.5239	7205440	358014343316	74930762526.7143
restart2vt	8	172813	34353.9048	971480	1797490149820	660425884428.0476	71120	4645136399	1623882709.4286
bgkcode	15	36024	11920.0952	2646234	660300896	186381469.7619	2291245	76186131987	43493068048.9048
tristan-mp	30	66908	11600.9524	126758613	82717020074	12161172809.2857	54954760	1840907365480	429734488103.5714
Floating_J	30	123831	12570.7143	1672564199	46692479360	12318893675.2857	168893148	1057081762220	106235835625.5238
oceanM_a4g	14	172820	22792.2857	13390743	269418675659	90910830193.6667	1086208	6739566370050	823731073570.1904
cdp_lsrbm3	34	29221	3635.8571	111705873	53147676207	9995033743.7143	10313913	255249575560	31660159753.5238
zeusmpi	4391	21613	19028.9048	24565530676	33389979883	32018436826.3810	2642396372	92300495559	65293695473.0476
mdrun_mpi_	31	77588	31782.9524	4034658	2537684361	897457472.9048	1368528	22146252696	6703005235.4286
main.Linux	15	172822	77310.0476	32072100	22175895211	9130264666.0952	3857872	84981318929	22437984660.9524
interFoam	21	172810	43749.5000	118490791	93928944015	17406683088.2500	5860448	112306929016	19710926562.4000
pasr	17	3759	382.4500	12912813	42953979	20047533.1500	539760	35997700	8892253.2500
rsmzz	9	86413	43371.7000	452562	5232527488	3829671699.0000	147544	5217337641	3583510075.4000
mhdf599	78	11186	5382.2000	540023410	1664536806	1084559832.9000	209065157	162549395358	87084794240.6000
simulation	61	49058	11096.3000	291785377	116329306926	29038794674.4500	879868948	104033298655	27177064179.8500
rasWaveDyM	502	172767	67452.0000	2959879183	17534321975	8372294164.1500	474943352	80243825923	25112207891.2000
SU4TorusMP	11	77756	8540.6500	2411363	10042015449	976092054.0000	210424	11666284422	1127741557.7500
siesta_lda	13	28827	4943.4000	11053738	3992080356	675279064.9500	908886	212869237704	35648167454.5500
numactl	13	172824	45666.4500	3295349	740347318448	46335542292.3500	178824	4515327550590	289395936584.4000
ses3d.exe	44	25695	4365.8000	790138856	41301123102	28502134093.1500	78115173	20558339244	3331578175.5000
Morris.cpu	17	24	20.0500	2306517	17400990	9067730.6500	1642473	14151505	7128921.6500
2LPTic	17	74107	7597.6500	3816352	919197896	147970032.7000	463146712	105229218522	11869034150.3500
Postproc	29	685	211.5263	4446181913	26622989498	18573743839.3684	41864700	791131344	577943219.6842
fppairs.x	25	1756	370.8421	85274961	258718982968	46793817550.2105	16852816	702101227	224472216.8947
mhdfnc2	22	69589	22298.4211	0	5314620961	2378289595.1053	0	324733595142	210575041541.2632
vasp5212_g	26	134751	15619.1053	56268607	24145636662	3487329550.3684	1357425	46747386213	11899467957.1053
pimpleDyMF	35	7223	1798.6316	400079233	2070521057	740574505.4211	54860433	1856193617	592457345.6842
test.cpu	9	36	14.6842	21597828	27422357	23476719.0000	809759	2518043	1402270.8947
vecwrite	87	491	166.7368	890191732	54472695491	31597781875.9474	246748493	108274987087	56809053824.6842
polaris_ch	58	73246	22621.2632	453119771	124459512558	51206013302.6842	20598042	77822167896	21876889060.5789
wrf.exe.cp	11	306	45.6111	49597909	1445007931	318149158.2778	1063911	739472626	129536986.1111
fluid3d_mp	78	86421	58939.6667	552660852	96415051393	52830317633.2222	3586295811	115467030720	82939163105.5556
pnfam_poun	15	27407	1927.0556	20149546	5821135853	510381170.3889	472432	6438025449	436749621.2778
vorpal	14	2795	440.7222	165152808	1924206019	1626076934.7778	569805	59935948	9262282.5000
parBC	13	752	119.1111	2024562	24381645	10054176.7778	137904	23522106	6994935.8889
mdrun_s	19	171102	88345.5000	101380901	42338591937	21708827598.1667	10291528	80478406491	40010526470.6667
cluster	1231	34239	14958.9444	296625595	2990901274	1587903718.1667	856035961	18682175440	8958868878.0556
olympus.sa	20	8510	566.1667	248153288	8526987626	2967163809.4444	869936	17644953486	2154513401.7778
sdsmC	62	380	123.6667	0	62651182	15853308.3333	0	58003857	12982768.8333

mat-vector	11	14	12.2222	4126120	4404832	4278090.2222	759972	1084045	936081.2222
laminarSMO	22	7222	3989.0588	93967358	1876092111	862583063.8824	6612518	1921440095	859727086.5294
analyze3dt	21	3600	1148.4706	6441105	237622136907	61510653828.9412	2216572	2009461106	498959924.2941
test.x	210	1849	618.8235	2101995374	15795940946	7334996107.1176	31276620	518438867	134841858.9412
lmp_ben_st	11	86425	26262.3529	150457059	4682758253	1432199366.9412	67286848	819645346571	234650260485.2941
aspect	16	315	40.1176	109111198	830276011	332496275.4118	1162278	51003056	11698572.3529
sander.bul	12	5412	586.5882	10557465	329489766	115961192.8235	198904	59302204	9266636.4706
dell_affin	14	1810	550.4118	10364293	3067059496	757790017.7059	1354472	2985335410	987396743.0588
mdrun_my	21	171028	140293.6471	37503891	11317988563	9185078183.2353	3132832	49688314622	39753740850.1176
xspecfem3D	477	50420	8783.4375	1570720042	33673959189	13956823636.1250	5667262863	2533487855830	987801147172.3125
vicar3dhye	9	50550	20681.5000	5838742	12659463427	4513032925.5000	323088	905527656704	407383962185.7500
line.exe	8	1399	450.1875	4928141	17151627142	8224672582.8125	164344	49338496	25484738.1250
siesta.vdw	22	12954	1813.4375	13893574	462690380	87673885.2500	751664	78034843158	9912532813.9375
iomi3d.Lin	21	1822	658.7500	40413822	6997355994	3844289278.5625	10525616	199366732379	105840655258.0625
flameletJu	26	28805	5457.2500	134499259	4356511936	3230597844.2500	10400375	1463708143	351696575.3125
Floating_J	26	172829	91069.7500	165545182	61752017296	35867051050.1250	15816016	721646829555	226876189170.8125
relax.mpi.	6076	12451	8548.0000	2494794463	4733310858	4035560426.0667	1023729102	2120887301	1514404024.2000
ens_driver	15	86412	68117.6667	27180292	35877759512	15962925516.9333	222888	80794186037	37153211183.6000
rfcst_t62k	23	96	30.3333	60749318	414065956	259558906.7333	15765503	43022831	24253972.8667
pre_proces	15	125	46.5333	2811004	56156998	22098766.2000	1678541	566213776	173113863.8000
parflow	30	172803	22181.8667	194282447	31338604772	4671298194.5333	15307288	110078950978	15373656399.6667
Charging6	12	1887	1001.2667	3492717	41244449	19506114.6000	1038263	119303108	58090371.1333
a_cool_mhd	22	31938	12492.2667	15380586	305450877888	143281812774.8000	3478904	660787505242	318293271926.9333
vh1-yy	24	4039	497.4667	90323556	2758807302	500145785.5333	13833304	9280906551	3667991101.3333
parC3	15	190	70.6000	2088022	6700447	3372760.7333	139712	3267719	887791.1333
nitsip	34	86423	22300.6667	73935341	11695511470	3252591935.6667	2909051	13110658483	3392770695.9333
izh.cpu	11	21	16.2667	665285	4679623	2014005.1333	122448	2717336	897134.1333
relion_par	88	12009	3713.4667	3084132935	1241434597760	454239118780.4667	12516472	9686452962	3243678129.7333
parfed	13	3495	1506.3333	1052898	385822367948	130908397635.8000	503107	1958771009	581051877.3333
namd2-2014	42	101585	33204.9286	21375491	13765963766	3363816085.2857	1128041	54208531422	12010743204.0714
rewrite2d.	23	251	78.2857	6862180	826297803458	177225591239.7857	2003754	551421994606	118271434594.4286
mpi.cpu	13	19	15.2143	1693572	34975283	7490164.9286	1521672	11917470	4813138.4286
post8	26	1285	587.0714	5581612241	4064762596330	1797609399946.5000	27469999	5779655097	3996151702.8571
vasp533_st	88	11377	4125.4286	179998334	13408902698	3835209619.5000	17695671	10720728360	3181309062.1429
run.x	16	21712	3889.0714	27963304	3400804746	1155744124.4286	1682096	147773220029	20464767312.2143
MHDAM3d.Li	24	3616	1464.9286	60311650	3561700670	743422166.9286	234056	1509833174	315089584.7857
sf3d	15	7564	4450.5714	3767622	108848378590	80721524730.8571	841844	71235399052	51509747950.5000
gmx_mpi	14	71917	27967.7143	31729241	1403746865	635996045.3571	1512642	8409165987	3384332738.8571
control2d.	19	361	134.7857	52748159	543710469	238367681.0000	3583464	36862687	15273309.0000
meep-mpi	124	6287	2242.7857	1939568	45612764813	10572249539.8571	1714069	173409200436	40031791599.4286
xsbig	27	211	69.4286	916876	2762036	1466780.0000	5837054	88377451	48359178.7143
omp_sunflu	10	80830	15912.8571	4869242	665183698	134687068.7857	128736	947650123	244256770.8571
jdftx	423	86419	53574.4615	20631636	1414756370	879695126.3846	8052214	1632375559	1009457140.4615
cannon	16	960	326.0000	2330249	33085146	12878742.4615	184248	35902717	12760998.6923
pelfe_STAM	786	7221	3741.7692	11526352495	48928987622	20117945727.7692	796594739	38613641741	11292119653.7692
wannier90.	9	595	197.0769	5567549	8609064143	5033227541.9231	778061	365877037	104874527.3846
vasp533_st	3395	86415	37132.3846	335124473	5704484167	2706963578.1538	283161156	6648708823	3134714053.6923
FSSVM	27	942	609.6923	110027582	1032640255	736041666.3846	106591694	1094933389	775860050.9231
mhd0	32	9815	2058.2308	537375162	1471411029	746566075.1538	414386796	127451901368	31652526893.4615
initmpi.li	10	1823	412.0000	5066527	218393398	39026000.9231	1064665	3335217691	832726834.2308
snappyHexM	11	7020	2404.5000	42160785	934555065	540526110.2500	963488	1904728623	787048308.8333
flameletSe	53	30028	5551.8333	1642975001	18391739112	7863358532.2500	2948682	6894760225	2168772099.9167
boat.exe	25	122892	16639.2500	43439192	16431636886	4609347627.0000	12852811	364289846515	50002650059.0833
gpaw-pytho	10	156898	13154.2500	1627301	4445460234	411678909.6667	432224	42574491433	3568002006.8333
pelfe_STAM	113	1427	402.6667	1904095705	120865216150	22196874176.0833	225709351	3545799833	2794624980.7500
cntor-pets	101	39484	16597.4167	126642265	14995629783	4133416727.8333	21166465	17337438282	4679674527.1667
cactus_abd	45	82892	64318.7500	32222942074	452615324770	306155442030.6667	225156160	3886663418270	2276765250968.5000
enzo-intel	154	3487	1779.6667	96895813942	179155072706	129285905003.6667	622647848	1003890114810	86363844056.6667
shear0z	3662	74798	53091.0000	29924575	3343168484	2711598841.9167	34620797	3442959799	2810540287.6667
messages.e	7	919	181.5833	162125	7641507	1633723.5000	87648	8796298	1800249.9167
gofr.x	27	11080	4119.2500	89630155800	458771241458	178606553247.5833	116788176	7987356044	2916406847.0000
DADDI_filt	3594	172822	158713.9167	38189744	1550537586	1363002689.5000	1988969190	99222635902	65061229557.2500
fim	39	173	85.0000	577892934	1101940923	766311766.0000	596482442	1512465262	932093989.0833
fortran_EV	250	3627	2579.8333	137118016	244479236	179562041.1667	118627483	309765828	238750945.0833
Floating_J	410	93007	17709.9091	7225950654	37343441905	13982344847.2727	2378587937	191851095129	32924637100.2727
mrg_dat.ex	21	1141	864.6364	15166710	240122511338	93745918627.7273	7797472	142565406990	102099400494.8182
FoldFS	50	36024	5947.0000	31971676	46992094751	12309087259.4545	29562538	51196174244	13507251900.2727
MM3D	69	36027	13335.1818	351171670	10493810530	3477217161.7273	24939144	11057050160	3434892335.9091
new.out	9	15	11.3636	837258	1588282	964793.4545	134824	382536	205792.0000
openmc.2.5	463	2507	1189.3636	13372572511	427721896469	114174914408.1818	65523416	2143581752	511982176.4545
SAM_ADV_MP	2496	146333	86220.9091	76363612749	177953720290	140500578733.7273	25529936067	3391847243610	1961550472065.7273
IZH.cpu	17	23	20.1818	1306599	17414328	12530325.9091	438048	14174690	9836087.0909

ccd_kernel	16	18	17.0000	3819829	5100629	4085261.9091	854799	1156256	1062701.5455
rockstar	835	2713	1446.5455	81728332879	363733865147	238009027065.7273	119749608	240679877446	93182205925.9091
PSE	1181	6656	4743.4545	201643632	870255114	620011288.0909	173781314	34176376948	6958350594.3636
testeroo	114	515	248.1818	2230616	5389269	3412658.3636	2016249	5928192	3386335.3636
charmm.den	20	444	174.3636	40600264	58589808	46944419.3636	3809674	17109176	9726212.7273
hw2-4.exe	9	928	750.0909	207196	7841254	6310008.4545	126728	9023721	7240436.3636
MHD-bsnsq_	10	777	273.2727	5711042	70306051	29621452.0000	1088614	5219957161	2285241398.1818
Bdd-bin-ds	17	36010	5999.6364	7134252	43529395229	26854388996.2727	241280	579053480	114087007.7273
Select_Par	15	621	190.8182	6437382	544451561810	144954490156.1818	321224	313623971	85042128.3636
papi	9	147	104.2727	140413	1401851	1047091.9091	126856	1566058	1148427.6364
reread2d.x	23	26	25.0000	36293347	340409276	309694722.7000	2050756	336153369	36193168.4000
turbc1.x	3615	54026	38789.9000	4305835343	12706963291	10115694690.3000	4386448711	70755012251	50172337137.0000
runRSQSim	25	573	109.7000	28144732	150705395	81271735.0000	16545569	104908922	33727756.4000
polaris_pr	19	80645	30105.1000	158350841	123401128593	68451321037.2000	15763807	85513992742	31898839268.3000
polaris_pe	18	86414	33306.1000	157246279	123010125462	66869280071.8000	15854160	84293522072	32270286968.0000
mm	12	121	28.1000	1432425	6424611	3139852.3000	281808	23901207	14484734.1000
koopman	27	665	197.0000	34036723439	2051167638890	765225914599.7000	69961312	27294048483	4909889670.8000
sf	9	2882	958.6000	3595318	109000065640	31442231980.7000	1007492	33384604105	13742769094.4000
p1s5	30	105	70.1000	43596693	238426893	212050873.1000	7573571	63397389	39121149.0000
polaris_bu	22	81115	30028.8000	160403955	123439534334	68050836130.1000	14712630	85651996407	31899379910.3000
mhdn	17	89262	35347.8000	13471075	6843877702	3496290674.4000	405335960	407302680573	269451621608.5000
prog2Procs	10	18	13.3000	205265	953891	434913.5000	121032	694352	324240.4000
cactus_rep	1211	30832	5684.7000	3790647469	200152970409	49072129198.3000	115106666	2805810446	581847516.3000
Floating_J	296	61057	25372.5000	9327009630	27293156082	16674965633.6000	2608938191	937276710900	285538945597.6000
free.exe	20	172815	80633.6000	66776862	63137695895	32107441098.4000	9927032	349925936185	155226602045.1000
polaris_et	18	80838	30255.2000	155423407	123718882394	69519249881.6000	13448895	85664466957	31966976590.0000
helloworld	11	263	41.0000	3029652	23596181	7569486.4000	2583208	10108002	5909782.1000
dns_shear_	7267	8288	7492.5000	111364254495	113255871392	112004756355.7000	216548943678	219473156439	217428179718.0000
hydratica-	12	76060	22953.6000	5473566	635671465	198991507.8000	302928	4252334397	1465308748.5000
model	17	56408	39562.9000	6666618	1556375690	1114583560.6000	1035136	104013929720	43956321056.9000
gh3d2m	327	67523	39814.6000	299312238268	749685485860	525572531207.7000	489916016	801812336177	329654509351.2000
RD_dbl.exe	24	93	59.5000	9934397	504221346	129838343.5000	1427597	161306585	39762720.4000
vat_3d_111	36	3091	1074.7000	612379490	636102224	620903730.8000	1301348	29557648	11391676.8000
vasp533_st	16	172810	49035.2000	66620688	3078685209	917906950.9000	355024	9195456161	2497619722.0000
analyze	16	4813	580.5000	17699225	649557886	113149213.8000	12733176	748325310	100824623.5000
mhdfn599	16	46645	25692.5000	12211021	4644313231	2686806349.0000	1359472	407295933251	244746523686.0000
sgNMF	55	459	338.2222	118382430	4988915820	3468359302.7778	117741911	11303036575	7823629005.6667
omp_mm	7	12	10.4444	200138	462265	334725.1111	88960	364359	222284.8889
ompmpiTKTc	21	151210	71676.0000	238638176	32875766034	16313193390.0000	5212088	37628116108	18439971434.4444
Stratified	6694	86424	43822.3333	5520691930	31122968620	16457718959.4444	51646545864	661776953724	329911511700.1111
openmc.2.5	209	2113	1475.2222	41294147	428163297360	94099707572.4444	16357102	2885659805	1002956619.7778
tod	92	91167	32963.6667	858522919	241922285415	104017150578.0000	243924830	428087701473	177862591293.8889
project	22	3764	1571.3333	241238534	21767186952	13347989857.6667	9005084	39895219647	24201716773.5556
pw2wannier	39	18003	9205.7778	95742037	224780807974	101647750454.7778	10882192	14743534635	7553297175.3333
boltz_	10	12537	2485.1111	2406284	51296275197	6271142224.5556	237216	100984163171	11610739428.5556
vasp.5.3.3	15	1688	316.2222	58362422	2682839227	913519289.1111	1366056	1667779511	193904755.3333
cactus_SBN	19	13020	1973.6667	162759787	5166434286	919461546.5556	504312	37038862811	5489604792.8889
UTAIM_post	468	529	487.3333	5010300611	5017548789	5013867125.5556	125796078	160992344	147138903.5556
sander.MPI	18	86419	12166.0000	17019043	2816599134	409954802.3333	282808	3251604098	451122345.3333
MATKTLAP	22	19236	2391.5556	8455208	7472725707	928473663.0000	2693563	8629958140	1054015934.7778
ermhd_w3	64016	172829	139958.6667	45449191896	230619579781	112014588644.3333	431826231610	1650623070200	771029227820.7778
IMB-MPI1	230	341	258.2222	827868573	1061772608	897792255.1111	522174046	794557921	603660456.1111
SWMF.exe	22	11507	2129.4444	56049722	25179932155	7843197337.6667	13063768	9824480759	2326189153.4444
diff-par.e	22	705	204.6667	42759757	1816909890880	583362046121.2222	11314204	21479141500	7238874786.3333
3dcode_y	6350	172697	91303.2222	593291377	12910629853	9159694828.3333	97859136	26555077926	15180865395.6667
nektarSNon	23	7214	1112.8889	27289307	567164778	112956603.2222	2463008	784182031	118547729.2222
uhp.exe	11	14	12.0000	1245085	1750905	1454661.0000	131520	424240	242399.1111
hmc	396	3599	1517.5000	859544716	3001723922	2045855895.7500	966286201	3259485077	2159318860.1250
g2	20	84379	22719.5000	23332504	28488897462	7517814892.7500	9064885	4207521522000	1062046646500.1250
scalarTran	32	63	41.5000	28206082987	30115622381	29233546832.6250	1766521550	2337498753	1983294198.2500
v14mtst5.x	54008	54026	54017.8750	1016008491	1128079061	1060681650.8750	19815880893	22528555403	21365678229.8750
WPCRsymInp	10	43227	6848.6250	4770121	354771996	60252513.3750	817970	433227425	68973610.6250
smpm_incom	21	3606	1574.0000	4928664	207761394	148525060.1250	1874072	213385117	55942710.2500
start.x	8	19	15.8750	7930554	43327462	28891140.6250	513824	720947680	380185093.2500
xaces3	25	35565	7896.2500	99095721	34310113303900	6443213831443.0000	53780979	10634884934900	2062771761871.2500
ilaunch	802	1132	1039.2500	71951178339	95409549370	92476830773.5000	55338780	67354955	58746303.1250
matrixmul.	21	47	24.6250	972837	2886539	1646621.7500	288576	1028325	763800.3750
pelfe_STAM	33	77518	21523.3750	153127535	127952908786	45377377862.1250	12671231	338737229460	91015072217.8750
lmp_stampe	26767	68727	60613.0000	475552613	1154235324	1023514373.0000	635175417	1635002079	1440524571.2500
enzo_wise.	19	520	312.5000	425321946	2001279290	1522938678.2500	12167274	174327679862	72149737267.6250
Wilson.cpu	16	24	18.2500	2357203	15767715	4226836.3750	1462778	12219473	3042291.7500
5.3.5_impi	562	14064	7308.5000	34794152	149170441	89488798.6250	11646107	357240670	129943464.0000

p3dn	47	324	203.6250	2474986	31967236	11857452.5000	1714126	30354826	10072162.6250
cactus_bow	35	85553	46172.6250	239477680	43811237634	14960230174.5000	6468736	389197909241	173582623849.5000
flameletSe	44	811	353.2500	1830788668	14530331042	5013105038.7500	3525027	1138356215	161730301.6250
arps3dvar_	23	189	163.0000	252969644	810945284	728035586.0000	11486720	660935065	577114547.3750
parsec-mpi	2909	13743	9722.3750	31661936	144663590	97807989.0000	237184117	14719368026	4907905553.7500
a.cpu	11	80	27.6250	1428737	4346104	2748620.5000	1484372	4017259	2592814.0000
fl2-avni-r	909	914	911.1250	14089407	14348642	14199899.8750	11378819	11635303	11488352.1250
projwfc.x	26	12483	3644.0000	36121259	301902368549	112672843899.8750	2252934	234060486817	111314570689.3750
pelfe_STAM	60	39415	20325.2857	20655517742	93705578448	59283344864.1429	371203725	153553390994	78583741298.8571
pimpleFoam	94	1697	866.0000	519467860	1346963729	692596095.2857	171949631	3091180298	928765281.4286
turbc2.x	18005	54028	48868.0000	6980743384	12738026873	11816781621.7143	23310873918	70714596485	61748832895.1429
openmc.2.5	1826	4425	3997.4286	13445173267	430104903560	182047625918.5714	169265081	5509000128	1915324598.2857
polaris_pe	12	21086	3029.8571	21890054	3826619075	762512304.5714	8938093	6972318482	1021683692.0000
CitcomS_GD	17	125603	78026.1429	6487172	4459584498	2800826884.2857	1025216	86703711827	43909536842.5714
pelfe_STAM	243	13739	6559.8571	9847802556	851345827746	365912935199.8571	185778552	1285870067430	574651758618.5714
MCPnektar	7219	172829	49428.1429	1234307513	68865580181	21820716093.5714	1128846239	112636977258	34606020967.8571
cactus_bow	30	85556	49926.5714	231659171	18821989202	8603604932.5714	6914153	368953798382	156543625997.2857
mdrun_6	784	171025	84806.4286	121748935	13639901875	6267405455.2857	222746558	47177860766	23140008996.1429
parC	16	168	93.7143	4056779	53410905	21192299.7143	346008	49315282	17302463.2857
xmeshfem3D	25	9304	6703.4286	1254127373	2147604510	1955223237.7143	100050320	80933138547	58627431079.8571
dftb+	10	625	214.0000	39034393	45119464	41322374.8571	349872	208956821	67880817.2857
mdrun2	19262	172820	104748.7143	18615075648	68184314090	46192472356.8571	8320052253	71863846226	43551945781.5714
bigstick-m	41	23318	7483.4286	292582279	29372338677	9450904037.7143	57685912	33157778078	10422906993.4286
spcam	22	6645	1327.7143	110162564	1042909558	735210455.5714	4599366	2979945036	994058433.7143
pelfe_STAM	22	70919	31350.2857	28849665	23101954983	13865595847.7143	6519383	424385560571	175137066745.4286
flash_benc	20	247	130.8571	97492045	209591961	161347133.1429	11325384	38727045343	30958088772.4286
c.exe	1567	34644	10421.4286	60623783	1125241222	355622201.2857	63563225	1306342413	406888783.5714
vasp5	440	2440	1047.5714	104858518	156274407	121578818.5714	429759103	2094918472	952886838.7143
tristan-mp	22341	172818	138629.1429	28537901990	106984046870	86793438506.0000	182029850345	1554390936050	1181058689664.4285
enkf.mpi	18	7692	6208.7143	30665860	33447447751	28552576973.4286	5277999	18007283058	15338955930.7143
Floating_J	421	72471	26705.1429	7233928446	30868736898	16937560007.1429	4228859938	76761765127	32506133360.8571
CCTM_D502a	8	215	119.7143	11898881	118596756722	28527119722.7143	983480	7460456133	2911919239.4286
mr_cov_red	182	184	182.8333	405690615	409100008	407357233.8333	2814453057	2814786152	2814635591.0000
procVortex	48	2873	1945.1667	810825723	60593095592	37703966771.6667	2729021	90808048798	56294070030.8333
qwalk-stam	228	37094	14818.0000	20685985	8042658804	3181431868.8333	24079442	10414969698	4046611389.8333
ompmpibulk	43	6812	3130.1667	41239560	481734968	243695585.3333	4499857	515863123	237865677.8333
mr_substri	56	69	61.3333	82699444	83737540	83042950.5000	394874406	449462691	404108368.8333
stormscale	10	1610	631.5000	3684029	10810387168	5405836958.5000	161912	13724707346	6845846575.3333
mr_dist_se	1579	2087	1679.8333	234662290538	235097700533	234924749024.5000	23020578804	23087072721	23040494723.5000
sander.bul	121	2718	784.8333	21327095	52189404	38514581.8333	11519897	47470812	29366339.1667
vasp.intel	14367	43144	23518.1667	994212729	5721641494	3544739304.6667	1094026897	6550548449	5180533814.8333
RD_lrgmem.	52	309	163.0000	9188389	13894999	11643372.8333	5385443	69657829	32701573.8333
countout	195	4443	1871.6667	29981572289	521296211260	258706863855.0000	16711732	284997196	138748652.8333
d16903d4.x	16	36029	30012.6667	19242318	4166150512	3408454350.5000	2849256	59676273572	46802475167.1667
parC2	14	183	88.0000	2076804	52045642	15552357.1667	130896	45814923	12114210.6667
singleFlam	22	593	261.1667	787212373	1143904252	1009846716.5000	36561576	692432720	158401851.8333
Compute_Th	3688	4565	4000.6667	17292322927800	17295790184500	17294031832066.6660	8228748346	11682914963	9601518530.6667
ipars	304	1641	1367.5000	36476941	46531651	44477475.1667	21010501	48985437	43178824.0000
WPSOCFTBul	11	3163	757.5000	4289316	29990611	11345547.5000	989230	30498885	8050979.8333
update-cos	1398	96493	26486.5000	325999235	33096425624	14920638787.5000	2190374766	17585352502	10046063825.8333
a_hydro	2256	16823	8450.0000	78533630483	3678407274600	1485950043101.8333	177987118468	1799616135910	792796478825.3334
WPFMBI	29	52875	23672.0000	10338119	17544971579	3374927836.1667	4805736	20358129039	3897455818.8333
preqx	194	663	442.8333	232453577	744777611	490570897.1667	166112255	595165647	389961150.5000
analyze200	24	2554	1331.3333	6260833	233984834055	141550256516.3333	2006566	2763312140	1202102211.1667
mr_substri	3448	4043	3761.1667	606335040389	607005818314	606723138644.5000	59848818110	59940818880	59902177303.3333
face_detec	927	935	930.0000	751549439	756009409	753784499.1667	283871405	284500078	284295184.1667
mr_bstore_	55	59	56.8333	82171039	83956559	83237553.5000	394949749	397055319	395660750.0000
procStrain	514	717	607.6667	16150106664	24215254523	18839089795.5000	20177755078	30258178965	23538355214.5000
mr_cov_map	958	993	982.6667	129204090058	140087518188	137998808921.0000	13698031418	13716038368	13708539333.5000
mr_dist_se	1565	1607	1587.8333	26240072756	28618973078	26753304498.3333	26341490532	26343929515	26342441391.6667
Sept1Test1	25	3532	610.5000	3332015	63163685	14049984.1667	743408	118675390	20534619.0000
pnfam_para	30	516	195.6667	5895316	43686820	29720810.1667	2575179	7997705	4643241.3333
arts	14	22856	3823.6667	52134929	3592459252	709941981.6667	951687	14446797641	2408969483.5000
montecarlo	18	1141	206.1667	5570788	12232255	6750251.8333	5343975	13665179	6828441.8333
vat_3d_110	14	2040	576.0000	2590980	625492244	413057530.0000	852802	17129175	5876107.1667
vat_2d_109	32206	172813	75028.6667	4277224580	22518943783	12770230330.0000	31200483055	213808362195	92669566073.6667
np9km_roms	19	56789	19023.6667	40246435	55894521626	21072959335.5000	15001249	1558542710860	523928621336.5000
Floating_J	190	6003	1656.0000	7150541908	11029500128	9008819356.5000	2064088046	146972185173	29543856262.0000
dam	22	129609	64820.1667	31805456	137310883910	69435693050.6667	2678089	182123206730	91121132134.8333
xspin1	308	7371	4757.1667	13162599478	120427290566	49740986188.0000	3431171205	404729108794	160772575468.1667
mr_f2_red	50	58	54.8333	81909912	83466725	82437344.8333	394708505	395416942	395023807.6667
msa-dta	228	28817	16560.8333	0	38698327902	14435378885.5000	0	37900732307	7231330024.6667

mr_bstore_	2210	2807	2418.6667	292384139525	299415686782	298133051432.8333	28452800326	28547197226	28498026579.5000
vasp_gamma	8437	76153	40621.5000	11590115056	18320456688	14788455778.3333	994536697	19245380572	13464924950.5000
mr_f2_map	2209	2872	2425.5000	300332473140	301118168804	300712235734.6667	28597131601	28660817602	28614141148.5000
ffinal	56	1617	845.0000	16448474354	245034899346	168563365930.3333	8942634	131154939	88276126.3333
Charles	201	7226	3398.1667	8330216	93994406	47201012.3333	21056643	473596622	214943142.0000
P-FoF-BG.s	17	1966	352.8333	17392162	4346089048	792043132.6667	2707472	278412593	51239473.3333
parBC2	16	83	48.8000	3147832	14271427	6059668.8000	902424	9775141	3128569.4000
addheader	966	2804	1578.2000	692950607	42947451263	13890336272.4000	728456586	2277771942	1431813502.8000
vasp_std_v	23	14007	4431.2000	64403848	299413140	140394774.0000	1004006	422284874	152941679.6000
windPlantS	34	12682	4171.8000	19994194117	31318748769	26792516952.0000	77082376	608500339052	146479376996.0000
fvcom_snap	97	31315	16811.8000	2429666445	13509585128	8280021759.0000	162636309	156616360759	94549789575.2000
GeneSeqerM	118	122	120.0000	10262600	10463078	10339279.6000	265239083	265478360	265328142.8000
adda_mpi	44	643	394.0000	11499953	53796399	20554663.4000	231165181	458797739	411159944.2000
Charging5	398	1734	1403.0000	17085009	61641177	48233464.2000	29404795	175689955	133987859.0000
varOmega.l	49	97265	19718.0000	12527748	4835549182	1969536828.6000	98940554	112044401596	24356265282.6000
vasp_ncl_s	27	622	340.0000	26809992	5969850882	1268442948.0000	18787539	550470845	220513972.0000
Sept1Test2	180	4933	2842.6000	7597248	86681118	51694673.2000	9127063	170851700	100028981.4000
nek5000	23	38795	8089.4000	53103898	10834762333	2759719178.6000	8554296	180274216091	39851733807.0000
Ray	20	5178	3069.6000	42723329	791530852210	461177969986.0000	1127824	1586345336	874500378.2000
HECTOR	439	86407	22608.4000	47460347	3183903203	991898624.2000	8234394526	630664614305	234941582991.6000
sfmpihello	10	16	11.4000	1371285	6955760	2698406.8000	381408	2656224	883041.6000
ABLSolver	458	70318	16250.2000	1977278161	7007345244	3181929577.0000	1341562673	37691998431	10531172668.0000
aaaaaaaaaa	18	41	25.2000	10735565	12604092	12022996.0000	6739235	9151770	7485700.2000
nrniv	29	32	30.2000	14525894	26856393	18025346.8000	1486920	4363355	2719000.4000
ungrib.exe	94	1444	580.2000	4434395988	33794472286	20331027327.4000	1633746298	722102029598	212791577133.0000
post_proce	16	205	73.0000	17562460	317610012333	63842886631.2000	8771360	37734922309	8255906082.2000
pelfe_STAM	25	31557	6393.8000	78943622	62119605641	17949865437.0000	67409943	158763873368	31939609950.8000
charmm-c38	24	77883	58281.2000	200327022	7760694405	5858719265.4000	4485308	13344900959	10024219160.0000
mh1w0ray1	34	144744	60938.2000	434051991	19546695819	9656229276.6000	186011986	930119871782	374112877177.0000
Map	23	2108	467.4000	37852975	4049937587	1535284546.0000	35224708	905026975	230962113.0000
post_main.	1974	7225	3550.0000	32607489020	410013275904	237454202965.4000	965743543	1141904870	1032045373.4000
vasp.gamma	1612	7154	4073.2000	70875093	128773041	95656112.6000	1207865753	2650117697	1797482408.0000
cntor	52	15645	10258.4000	24688583	2711173608	1781489638.2000	1568888	2948926010	1908532352.4000
UT-HYB-SV-	680	3607	1951.8000	26655349	126491298	60787351.2000	18309445	94773279	43912952.6000
amps	63	36005	10112.6000	1255705205	6959494816	2446476084.2000	876203630	157845714708	39195012872.6000
waveDyMFoa	119	172763	77507.8000	1499818020	14561038174	7473283185.2000	1706385003	146893975379	66181784527.2000
pelfe_STAM	40	71384	43091.2000	161160259	22578738877	16424927384.4000	77020109	366513667114	206010619527.0000
Gadget2	11	12936	2653.8000	2728251	55591480377	11335599605.4000	242592	2655995689780	531200119689.8000
lmp_12Apr1	18	5113	1088.4000	15028591	185007433	52004162.6000	770200	276699800	61087720.6000
matdyn.x	16	38	25.4000	1830759	7250756	4308192.8000	1154653	5729610	4118969.4000
DADDI_DP_S	3631	172820	60628.4000	38646077	1478949402	523009676.4000	2061940452	80388391516	28197196520.0000
pismr	14	17	15.0000	14313997	14710349	14530026.0000	272432	730872	522762.0000
ARWpost.ex	1353	12432	7014.7500	85901177590	602629835794	374816572155.0000	13417136993	198289368507	107243024233.5000
drivempi.l	14	25	21.7500	14821835	36000784	30631544.0000	3507900	4418395	4088598.2500
sapo.real.	30	16769	8402.2500	138399962	18006749045	9111578332.0000	16966773	35947971649	17982586940.2500
lmp_gpu	10	22	16.7500	14738148	15051713	14894695.2500	1072765	1091223	1082992.5000
polaris_pe	444	86405	22430.0000	218255821	51495086951	29531555931.0000	169187761	94796256795	23900833443.5000
aa_planeEx	13	457	250.2500	4044994	12784708414	7991796547.2500	1102502	106823490	66900946.0000
v14mtst.x	36011	36018	36012.7500	677641154	701876146	691645862.0000	13023044036	16304527716	14834017660.2500
sigma.real	150	1836	653.0000	66845348	20683100107	10364380758.2500	38268724	331046309	132641984.5000
histoParal	37	5400	1381.2500	753446876	91346649215	23413074447.7500	427069464	15956815672	4321429426.0000
RD_quad.ex	102	176	126.0000	21094623	1847841687	551528067.7500	25477501	485830814	163755373.7500
main.x	95	5413	2327.7500	179776980	1201257195	613005545.7500	526775619	38722027320	16548767940.2500
q2r.x	17	18	17.7500	4177830	7441889	5274503.7500	3523749	6877939	4552587.0000
ridge_regr	1014	1028	1020.5000	1202813084	1209177361	1207340871.5000	292150314	292718847	292334107.0000
sample	11	9817	4915.5000	35386460	16743607888	7184304103.2500	650480	28611239764	13490470084.2500
Sandia_100	24562	86426	57418.2500	6566295075	18902351759	13144843324.0000	9255442419	33504631977	22530852367.2500
CAMx.andy.	106	422	279.2500	769191367	793531946	781473918.7500	22208374	255199419	149708665.2500
sfmpiadjbo	116	8785	4659.0000	2699394354	2626961303610	1187082911838.0000	42075090421	3755419392350	1607562384485.5000
lmp_Jan14	415	419	417.0000	20129041	20387697	20283689.2500	5089172	5393340	5273323.7500
L-Gadget2	18	896	241.5000	40588033	39898685880	19886478798.0000	4842568	247426783796	61866803023.7500
Case6a	284	17432	6646.5000	7402229	402595570	131982842.5000	23510477	1313603595	462997068.7500
mdrun_mpid	37943	38631	38247.5000	316343321	325834988	319661160.5000	1160912440	1168479039	1164302460.0000
pelfe_STAM	306	72021	18684.7500	13533489034	60862172171	25605369691.7500	366180619	97009684147	24834434600.7500
Case6	1481	1666	1614.2500	22892431	37597459	27145521.0000	70837616	86759061	76515474.2500
enzo_2.4.e	52	5883	3091.0000	1405324617	2478819833	1849198037.7500	18467018	182540219429	54963926126.0000
vtkreader_	42	1029	418.7500	22156001	12900276303	3313997939.5000	14971666	297723377	95127809.5000
xhpcg	50	960	288.5000	5446481	35480519	14236205.0000	1917787	36552401	11139598.7500
ompmpiTKTc	24	29004	15861.0000	241393014	7222040766	3701249343.2500	8872834	8031535422	4059948940.0000
cml-proxy	50	7228	2074.0000	3887739444	9143988360	7039516000.0000	602876385	6909595381	2407845500.5000
ex4	216	493	318.5000	106816344	604069567	377490379.2500	88471280	563427741	342538079.0000
pelfe_STAM	138	4264	2008.5000	13577347646	14509372481	14055891605.5000	236820792	14487483782	5770956072.2500

maketrie	18	911	614.2500	2563105	3908977035	2928842203.5000	2099961	35980880241	19119301937.0000
xlmg1_v8_2	42	292	164.7500	59663241	217997953	130617230.5000	15525513	174067862	88408488.5000
athenasphe	19	4619	1170.7500	24693614	118294053871	29592790566.7500	12380283	265843573081	66471061316.0000
makesimobs	29	33	30.5000	3594747	4070194	3827302.5000	111163394	111333708	111249361.0000
bam	12	14294	4125.0000	3399638	639381189	169009361.0000	14974936	5973953596	1506580114.0000
ideal_mpi.	51	2633	1147.5000	738046091	2975710185	1736713087.5000	1025467232	105116291829	47278685408.0000
analyze225	809	1542	1125.7500	8687344529	13891869550	9990266665.7500	216391485	239059144	231649070.5000
pelfe_STAM	55	308	120.0000	13467018042	13930245120	13588777221.0000	232147235	593703601	323602180.5000
spart	14	34833	26016.7500	3932208	1614988278900	654917304770.5000	948808	8354823433	2461675539.2500
acdtapernm	25	26	25.5000	15604166	15798062	15733404.5000	167109824	167340392	167263808.0000
vat_3d_113	1398	2132	1884.2500	623054968	627772910	626491756.0000	14258775	106445858	40068178.0000
icoFoam_io	18	3605	2706.5000	127368244	154924028	147988433.5000	1444064	35958246	27287355.5000
boat1.exe	19	7220	2685.2500	1106180597	3215546379	2397586424.0000	14565893	20126699628	7428433933.0000
CalcSA2	761	6645	3125.7500	14993672	74007465	39081828.0000	47144387	264149762	125108341.2500
runSimsExe	44	172802	126816.0000	1373782331	41720670069	29172025290.5000	418793195	11379477519900	3329482252248.7500
postProcNe	353	3247	2088.2500	6396598725	48733146123	31928337234.7500	111069042	31281764604	16512437511.2500
NVT_180	16535	20622	18322.5000	552563666	694433872	614452567.2500	3193639565	3349489229	3267264692.5000
octopus_mp	21	172818	124517.7500	24889727	6436477539	2706609191.5000	660008	2887556000700	1674511613912.0000
mypisoFoam	3631	7228	6318.5000	2489071503	2685228330	2631181001.5000	1462664415	13606155014	4723320530.0000
polaris_pe	28	63306	16169.0000	215287713	42687714772	15132267759.5000	20045280	75733194186	19113734739.7500
compressib	22	87	38.7500	246272896	261952194	254507481.2500	12024869	16346415	13648901.2500
rcb	2683	3417	3047.0000	108755563194	108799329101	108779377156.7500	1722895578	13171087061	7454875734.0000
h6001b	19	128082	67696.0000	33362817	28407321578	17534742088.2500	3128656	119621060027	63646599661.2500
polaris_pe	32	57516	14584.0000	115625176	21718257588	8078488586.0000	16837626	38390419450	9716204507.5000
vasp533_nc	150	666	396.2500	120495686	135777747	128226212.5000	261067907	389328865	322897070.2500
d16903d5.x	54004	54029	54016.5000	5771485384	6729369053	6247435209.7500	78360357252	137255571093	112048055254.7500
olympus.pe	49	15074	7559.5000	7410425613	25524075973	12883804512.5000	24299949	589043724296	268273361670.2500
a_mhd	8019	9626	8717.0000	118655477938	118841907571	118721763208.3333	266611488490	266897822005	266714078972.3333
nrlmol_exe	704	77413	51842.6667	129500305	7023773463	4679621069.0000	919358438	8125444182	5723303896.6667
quadtabu	727	1450	980.0000	10088435	15994487	12189829.3333	7592385	14465605	10038565.6667
hostname	161	1807	727.0000	9244533	20640130	13833482.6667	7265234	21227901	12224697.0000
NewAlgorit	5533	10822	7892.3333	170293479	458563171	294048798.3333	668376316	2272599585	1314405543.6667
cdp_fdm	19	26	21.3333	300839877	304545789	303196162.3333	9596568	10921256	10464488.0000
athenasphe	4315	18233	9040.3333	51342566625	245114286714	138199777945.6667	117274800705	532289187483	305135505125.3333
dstatictes	19	36	25.6667	11447542	12475254	11839680.3333	6911585	8097754	7353905.0000
gemntr	48	54	51.0000	875775325	1006105821	947671039.6667	1255914491	1284280823	1271241215.0000
turb_stat_	19	576	207.3333	14182502	37215487230	13298949183.0000	5977443	1513417018	514583554.3333
rfcst_t62k	144	216	172.3333	145167422	154756983	148470043.0000	84740395	126756438	100807375.6667
parsec-mpi	247	786	454.0000	11528424	16237513	13279980.3333	4498613	58041264	38023769.3333
iblank_par	17	39	24.6667	8050427	461546503	169465060.0000	1359457	50597697	20505842.0000
Sept1Test2	6010	36001	18543.3333	121113400	635523379	344607047.0000	248580266	1036082523	636557411.3333
NewAlgorit	20	7200	4033.6667	5068382	131020686	80725950.0000	1697512	524650026	327763433.3333
NewAlgorit	2337	3522	3009.0000	31411934	45167363	39817394.6667	145703303	196353380	171903141.6667
ompmpibulk	45439	155769	118992.3333	2993955743	10187476903	7789636516.3333	3482097601	11851906411	9061970141.0000
turbsl12.x	54013	54018	54016.0000	11989936630	12045643914	12020740931.3333	115638456540	119324581366	117185814586.6667
hw2Q4.exe	125	144	133.6667	1204935	1315955	1243315.6667	1303298	1433444	1348274.3333
Case4a	1662	1875	1740.3333	21216637	24520997	22328297.6667	71933429	74483982	72793820.3333
INCITE_cut	271	1735	1110.0000	1051007886	1532606856	1365711668.6667	4903998	1493810359	827653305.3333
Compute_Co	2133	4792	3827.0000	9538560074870	17293731640800	14708624483490.0000	4802963854	9568265009	7883081063.0000
tolling	777	779	778.0000	425570654	426803145	426135001.6667	238848330	239312851	239138093.3333
P_merge_LD	21	2578	873.3333	44718293	12598840326200	4202562015179.6665	394741096	9367915064140	3125306121124.0000
CAMx.andy.	122	131	125.0000	152713804	389096317	300386812.3333	9256098	75998071	45186933.0000
echo	9	2035	804.0000	131321	3061100102	1021684288.6667	118040	1284640623	429670731.0000
lmp_stampe	40	4338	1768.6667	188118766	588706291	364867196.0000	12805048	1386665638	612876913.0000
quad	684	1247	873.6667	9820845	14289239	11379604.0000	7178987	12471133	9024729.6667
vasp5_iSte	1287	11188	5998.0000	583077547	2551369528	1555376061.6667	126259298	1110581726	647915597.0000
ArcOn	1805	172811	60609.3333	179332988	14196969017	5198973564.0000	44595559	15823744049	5657826337.6667
ompmpiTKTc	6555	50109	35591.0000	1675661108	12308376240	8764137862.6667	1739670046	13776004551	9763893049.3333
Case6b	10	7208	4808.3333	3523021	93606677	61897458.3333	1066907	424556531	282173567.6667
xtest	34	10818	4414.6667	5651880	354502052	147315167.0000	1566416	407743872	166501578.0000
vat_2d_109	387	7205	2877.3333	18370376	79702451	40804218.0000	9031906	142181192	57440225.0000
vat_2d_109	18	1955	1113.0000	14676183	32024805	24534805.3333	1107321	39218760	23307806.6667
vasp_5211_	145	702	511.3333	56283503	271937146	190513735.3333	7380452	196787054	131414605.6667
a_cool_hyd	2536	4089	3201.3333	78320531828	98961987599	85243473926.3333	177641180156	222107866713	192511302965.6667
mpi-waveto	86	102	93.0000	301346836	370652768	333741234.0000	201551136	239232151	217052907.6667
polaris_pe	94	49096	16549.3333	49550048	7742277624	3328569927.0000	138856977	13959677113	4747634541.3333
vh1-mpi	20	59	33.6667	6175870126	6309921283	6221595214.3333	17793888	12955971843	4331196131.6667
mini_ghost	72	779	442.0000	34330392	46859536	41634308.3333	3789427	10316188	7103295.6667
windPlantP	313	331	324.0000	296149371	686415082	555522168.3333	923741347	943773317	937027248.0000
citcom.mpi	13	22	16.3333	1916559	2915028	2279573.3333	1110660	29238547	11657533.3333
vasp533_ga	66855	172814	137494.3333	1471799747	3258262391	2648168827.0000	5329797064	9478156324	8072755270.0000
openmc.2.5	526	2796	1288.3333	13355273160	13422835163	13377853774.6667	40746086	123830301	68643864.6667

pH_protoco	270	2836	1205.0000	279575611	306831609	291685985.0000	4317097	35668287	17958150.3333
Rscript	744	768	754.6667	483576386	483811741	483663921.0000	446296069	446574391	446401354.0000
helium	1439	3250	2593.0000	58736487	101812562	87215596.6667	109997549	1905022335	1077069159.3333
PFFEM3DSol	613	18961	7582.0000	377025806	2793239980	1306234808.6667	116328864	2910721141	1184673636.6667
Stratified	335	141156	94215.6667	5039654708	112798066459	76878595875.3333	2679187608	2618866031060	1746803749909.3333
vasp5.3.3-	21	62	47.6667	106406737	143855316	120332507.6667	13522295	101980614	45002261.0000
vat_3d_110	12	1439	884.3333	1830926	622847106	415463418.6667	123016	14193745	9023705.3333
P-SPH_Jan9	20	26	23.6667	15643454	780174789	525259786.0000	1657952	765537288	510823472.0000
bin	18	150	103.0000	3712924	4423094400	2949870994.6667	760310	30852535	20713372.6667
real-org.e	30	260	181.0000	253398644	5226755493	3568936928.3333	1298753	13340131835	8887252463.6667
vat_2d_109	490	7209	4967.6667	19149610	265454528	121588674.6667	11227644	378594698	180684023.3333
OpenSees	9	200	74.0000	22697230	24364070	23560822.6667	206728	11590582	4094060.0000
analyzetur	1658	3725	2394.3333	175485638479	175517356917	175499265894.0000	1970928895	2702851889	2451918411.6667
h2mol	4228	30027	13327.0000	11460097572	40439322155	22029283408.3333	5871847650	49777717042	21134055582.0000
vasp533_nc	17	9973	3977.6667	69405719	289383676	164079661.3333	1702724	6858716915	2350194985.6667
relion_sta	15	20	16.6667	45813200	190934875	95282393.6667	390048	2072937	1003512.3333
rankhist_r	4771	4816	4789.3333	23765833942	25588734161	24980895180.0000	127025241	131532795	129036706.6667
gridart_pe	87	549	265.0000	155392129	162721106	157873932.6667	502176210	510803213	505309036.0000
vtkreader	1030	1324	1177.0000	12899758849	12901146028	12900452438.5000	14433440	16046277	15239858.5000
polaris_bu	3649	48541	26095.0000	423649864	2776233753	1599941808.5000	50083670	1847134840	948609255.0000
Sept1Test2	295	36011	18153.0000	11515375	614466691	312991033.0000	25099709	883120495	454110102.0000
Sept1Test2	4891	36008	20449.5000	125824748	642332007	384078377.5000	427337053	1074275394	750806223.5000
sextet.x	1694	2348	2021.0000	2360060336	3173906140	2766983238.0000	2544775257	3584438498	3064606877.5000
parsec-mpi	781	3621	2201.0000	16084763	39077001	27580882.0000	45972157	57843112	51907634.5000
pelfe_STAM	324	326	325.0000	13638709479	13671984922	13655347200.5000	230315943	236829731	233572837.0000
vat_2d_109	238	172821	86529.5000	16649307	1617862801	817256054.0000	5954989	21615434421	10810694705.0000
pagerank-d	20	93	56.5000	203664790	2860981302	1532323046.0000	2877520	46566343	24721931.5000
curviexy2d	56	56	56.0000	124637240951	124642325779	124639783365.0000	265651711	271318443	268485077.0000
pelfe_STAM	40	325	182.5000	293253941	572065114	432659527.5000	18508156	434376189	226442172.5000
pelfe_STAM	318	9032	4675.0000	13694926365	15086575480	14390750922.5000	232844663	41731097752	20981971207.5000
DLPOLY.X	1224	3620	2422.0000	15584085	35056004	25320044.5000	12868364	37821398	25344881.0000
Compute_Ev	6160	14071	10115.5000	25935907047600	25939623003800	25937765025700.0000	12688769755	17131932861	14910351308.0000
Sept1Test2	136	138	137.0000	7341871	7395319	7368595.0000	6254429	6313144	6283786.5000
parsec-mpi	781	3614	2197.5000	16204310	38942087	27573198.5000	45642102	58024377	51833239.5000
analyze_sc	166	256	211.0000	18953403910	31416266894	25184835402.0000	48590214	80805783	64697998.5000
Sept1Test2	3729	22719	13224.0000	72347387	372958481	222652934.0000	127910405	438785246	283347825.5000
bb_planeEx	246	429	337.5000	12783965790	12784787444	12784376617.0000	105245248	106895295	106070271.5000
a.881	98	101	99.5000	44845660	48275163	46560411.5000	32228208	32812540	32520374.0000
parsec-mpi	216	789	502.5000	11212065	16279095	13745580.0000	51102464	58095588	54599026.0000
h6002a	93882	93882	93882.0000	24562040192	24562040192	24562040192.0000	142537922582	142537922582	142537922582.0000
score_jd2.	11	15	13.0000	185773572	186333488	186053530.0000	1182556	1799117	1490836.5000
core.exe	18	10453	5235.5000	5559523	392297849	198928686.0000	629056	442336436	221482746.0000
Select_Par	2045	2845	2445.0000	8500169611020	9673757207860	9086963409440.0000	6231354404	6309920112	6270637258.0000
Sept1Test2	864	2522	1693.0000	16405332	26932747	21669039.5000	47622094	50656039	49139066.5000
parsec-mpi	780	3611	2195.5000	16262262	38908804	27585533.0000	45560417	58090205	51825311.0000
re32.exe	18	9713	4865.5000	5550939	344519042	175034990.5000	620928	393436653	197028790.5000
HHhost.cpu	18	21	19.5000	16098189	16139813	16119001.0000	12961483	12993224	12977353.5000
Sept1Test2	21	5666	2843.5000	7900038	127363904	67631971.0000	1543619	363694334	182618976.5000
parsec-mpi	598	792	695.0000	15365982	16141568	15753775.0000	55597633	58064314	56830973.5000
pelfe_STAM	18456	18456	18456.0000	166861051001	166861051001	166861051001.0000	195126189217	195126189217	195126189217.0000
dist_init_	14	56	35.0000	575854	1127103	851478.5000	292448	590242492	295267470.0000
h1201a	92111	92111	92111.0000	21037565527	21037565527	21037565527.0000	93600454521	93600454521	93600454521.0000
Charging2	160	3127	1643.5000	9533423	75915096	42724259.5000	13093757	201492378	107293067.5000
parsec-mpi	786	799	792.5000	16415322	16431720	16423521.0000	58240644	58355676	58298160.0000
helmtest.x	7	7	7.0000	11074353	11080259	11077306.0000	333240	340000	336620.0000
parsec-mpi	788	793	790.5000	16193082	16448959	16321020.5000	58070605	58278440	58174522.5000
parsec-mpi	792	799	795.5000	16417550	16503605	16460577.5000	58326786	58381572	58354179.0000
pelfe_STAM	316	3618	1967.0000	13902963063	18604072865	16253517964.0000	560297517	13878058883	7219178200.0000
NewAlgorit	1055	3188	2121.5000	16697204	43715728	30206466.0000	50785215	183046658	116915936.5000
ompmpibulk	94043	94043	94043.0000	19553266398	19553266398	19553266398.0000	20794233128	20794233128	20794233128.0000
dos.x	16	24	20.0000	5276874	24348779	14812826.5000	1295709	1724672	1510190.5000
parsec-mpi	789	801	795.0000	16400803	16494412	16447607.5000	58307227	58338547	58322887.0000
Sept1Test7	34018	34018	34018.0000	4269118037	4269118037	4269118037.0000	5988501997	5988501997	5988501997.0000
Sept1Test8	736	2410	1573.0000	16985240	41390760	29188000.0000	21836782	80091648	50964215.0000
arps_org_m	100	21033	10566.5000	2469847998	6043034525	4256441261.5000	407407063	16077586671	8242496867.0000
date	377	1519	948.0000	16000019	133442171	74721095.0000	15292357	24699351	19995854.0000
parsec-mpi	792	3618	2205.0000	16242151	39023243	27632697.0000	45806915	58065854	51936384.5000
hellohost.	11	12	11.5000	923790	963350	943570.0000	746002	794205	770103.5000
pelfe_STAM	120	145	132.5000	15542390105	41444431302	28493410703.5000	299585961	878048138	588817049.5000
WAVEAXI	65851	65851	65851.0000	639232164	639232164	639232164.0000	20849815915	20849815915	20849815915.0000
fd3d140203	9500	9585	9542.5000	158733685	160004704	159369194.5000	1493293794	1496823928	1495058861.0000
h0502b	98345	98345	98345.0000	23436345414	23436345414	23436345414.0000	114383522131	114383522131	114383522131.0000

calcAvgFlo	21	74	47.5000	45557869	638972436773	319508997321.0000	13278588	2602706884	1307992736.0000
vasp.5.2.2	4874	40904	22889.0000	320128701	372489732	346309216.5000	296030748	887855916	591943332.0000
c.lines.x	20	21	20.5000	32666551	1464893818	748780184.5000	1321530	2524045	1922787.5000
parsec-mpi	248	3617	1932.5000	11611461	38596130	25103795.5000	35699236	51516059	43607647.5000
b.exe	9502	10021	9761.5000	328712370	623550444	476131407.0000	377362924	715472055	546417489.5000
precipveri	65	66	65.5000	662894717	663090376	662992546.5000	35961133	36777610	36369371.5000
MWOB_8M12L	147046	147046	147046.0000	102568592105	102568592105	102568592105.0000	1014723634840	1014723634840	1014723634840.0000
Stratified	31	124	77.5000	1622994756	1624245226	1623619991.0000	56232564	1529520941	792876752.5000
Floating_J	3809	27923	15866.0000	10388673445	17806919268	14097796356.5000	14166657743	697490730805	355828694274.0000
ompmpiTKTc	57056	57056	57056.0000	15155870065	15155870065	15155870065.0000	16882296251	16882296251	16882296251.0000
parsec-mpi	785	3601	2193.0000	16252538	38486831	27369684.5000	34419423	58113567	46266495.0000
mas_0.2.1.	153	895	524.0000	245492124	917313718	581402921.0000	91465111	56226702388	28159083749.5000
hw2p2.exe	8	9	8.5000	194894	196102	195498.0000	112624	113448	113036.0000
pelfe_STAM	322	3608	1965.0000	13906547538	19687702112	16797124825.0000	559567658	17816339405	9187953531.5000
parsec-mpi	781	3624	2202.5000	16066030	39078198	27572114.0000	45987236	57895450	51941343.0000
mpi_hello_	9	15	12.0000	992746	3435647	2214196.5000	954908	3154659	2054783.5000
Sept1Test1	3040	3387	3213.5000	71003545	71484849	71244197.0000	157071109	188220007	172645558.0000
Sept1Test1	10802	20804	15803.0000	192509835	372996714	282753274.5000	395674896	758563131	577119013.5000
plascomcm.	1710	3557	2633.5000	30990953	52835686	41913319.5000	54970097	3613692829	1834331463.0000
ssh	226	5489	2857.5000	4031792	74298733	39165262.5000	4237742	75056468	39647105.0000
pot3d	30248	30255	30251.5000	3020972503	3032804490	3026888496.5000	3706199383	3718918423	3712558903.0000
Case5a	11	2345	1178.0000	5234710	32088071	18661390.5000	816980	146106083	73461531.5000
5.3.4_impi	1983	8337	5160.0000	42324487	96419685	69372086.0000	20004356	235786658	127895507.0000
Sept1Test2	26531	36017	31274.0000	463835260	613879407	538857333.5000	712923897	890528145	801726021.0000
enzo_freez	169	325	247.0000	832537041	1477689876	1155113458.5000	1225952592	172641218642	86933585617.0000
add_pertur	229	2003	1116.0000	813114006	138397200989	69605157497.5000	753164257	29241314569	14997239413.0000
xlmg1_v8_1	17	23	20.0000	13441608	23459501	18450554.5000	1253048	2762032	2007540.0000
mdrun_gpu	29	32	30.5000	259271505	260584014	259927759.5000	33035469	33250694	33143081.5000
vasp_gamma	22	98	60.0000	41652503	67845515	54749009.0000	19896480	50445520	35171000.0000
sfmpiinvbo	250	1804	1027.0000	427985825238	459448514117	443717169677.5000	512855923165	512866786434	512861354799.5000
simple	17	19	18.0000	2046757	2138497	2092627.0000	1857103	1944708	1900905.5000
cfs_ncep_p	39	39	39.0000	68721845	68978601	68850223.0000	143043148	143222454	143132801.0000
IOR	61	348	204.5000	175700688516	1599644185010	887672436763.0000	412681798467	1856208794360	1134445296413.5000
hw2-2.exe	9	11	10.0000	192896	382254	287575.0000	115368	341400	228384.0000
cdoitf	94	615	354.5000	24857448	103338495	64097971.5000	1788513	6889248	4338880.5000
DNS2d_p8.x	22	26	24.0000	7841360	345010296	176425828.0000	2130954	803506398	402818676.0000
mh1w0ray0	33	45	39.0000	898187578	901434610	899811094.0000	55561653	60453301	58007477.0000
mhd699	1388	1466	1427.0000	1254590012	1264305000	1259447506.0000	2227492349	2238784344	2233138346.5000
enzo-m.exe	20	21	20.5000	89558182	90277740	89917961.0000	5072104	6046000	5559052.0000
parsec-mpi	789	802	795.5000	16261964	16404955	16333459.5000	58020492	58310527	58165509.5000
namd2new	12	927	469.5000	15533400	156435378	85984389.0000	1056071	170176214	85616142.5000
ompmpiTKTc	27	59120	29573.5000	240424033	16644141975	8442283004.0000	7723464	18306130204	9156926834.0000
Sept1Test2	139	284	211.5000	7421786	8233619	7827702.5000	7175872	11611800	9393836.0000
p2p_isend	103	665	384.0000	10636970	19721179	15179074.5000	4721377	15219549	9970463.0000
binary_CH	6829	7967	7398.0000	204879976	320060819	262470397.5000	3116519909	3335041106	3225780507.5000
procDiff3D	416	596	506.0000	10774521272	16152282935	13463402103.5000	14800943416	22193409448	18497176432.0000
switch_map	154	1806	980.0000	3998891	27733934	15866412.5000	4728622	26732626	15730624.0000
postScalar	1296	1510	1403.0000	11748784338	11749225882	11749005110.0000	183428592	184295167	183861879.5000
Floating_J	1120	4849	2984.5000	9558557792	10678377692	10118467742.0000	2307879741	5807738522	4057809131.5000
SAM_ADV_MP	39734	40706	40220.0000	3209999212	3272835934	3241417573.0000	240145390290	240209064552	240177227421.0000
ccsm.exe	1468	1569	1518.5000	32197218856	105725420252	68961319554.0000	3103223376	145957410718	74530317047.0000
1690m2.x	36004	48671	42337.5000	732703333	926525165	829614249.0000	20506630822	21216594913	20861612867.5000
Stratified	238	317	277.5000	2418632127	2440808841	2429720484.0000	1954691661	4271400988	3113046324.5000
case3	12	7221	3616.5000	5372380	95091793	50232086.5000	977517	472497333	236737425.0000
DevelEP	314	318	316.0000	4855244	6959985	5907614.5000	4175889	10900635	7538262.0000
P_DNS	172810	172811	172810.5000	5618254600	5703938276	5661096438.0000	13434347884	23479804761	18457076322.5000
FPHC	35918	35918	35918.0000	2487362851	2487362851	2487362851.0000	16300981960	16300981960	16300981960.0000
parsec-mpi	550	3614	2082.0000	14262820	38926122	26594471.0000	45745536	55237682	50491609.0000
minbiasPar	41	45	43.0000	669248382	777833605	723540993.5000	327875576	455975336	391925456.0000
pelfe_STAM	600	21608	11104.0000	16452205068	55741858985	36097032026.5000	1275250764	76905601892	39090426328.0000
hw2Q2.exe	72	86	79.0000	784712	995746	890229.0000	818852	1062473	940662.5000
pelfe_STAM	23	49	36.0000	38298368	13467782745	6753040556.5000	16029173	216326214	116177693.5000
Vorticity_	31	32	31.5000	12839700257	13507040014	13173370135.5000	98071472	106189472	102130472.0000
pelfe_STAM	323	86407	43365.0000	14065726807	90787523727	52426625267.0000	737775745	245987944121	123362859933.0000
xlmg1_v8	17	157	87.0000	15519949	23598636	19559292.5000	1269632	16484888	8877260.0000
DischargeW	82	7207	3644.5000	7329069	141591874	74460471.5000	12350659	284337005	148343832.0000
wrf.exe.te	81	116	98.5000	55260365	1421126005	738193185.0000	2737437	787273066	395005251.5000
parsec-mpi	787	801	794.0000	16410708	16494031	16452369.5000	58308818	58314192	58311505.0000
varOmegaLr	172801	172824	172812.5000	35452524177	41977108366	38714816271.5000	145398341831	213753151264	179575746547.5000
lbs3d-mpi.	40	89	64.5000	11579710	42236552	26908131.0000	4916582	23305776	14111179.0000
pFP2d	33	38	35.5000	21827327	393852245	207839786.0000	1300104	64046648	32673376.0000
coawstGv3.	1594	1826	1710.0000	2031989553	2315397627	2173693590.0000	2312086094	2641908072	2476997083.0000

xget_pertu	21	22	21.5000	9422369081	9425152019	9423760550.0000	10454115040	10455819350	10454967195.0000
bands.x	31	96	63.5000	25645739941	220215148606	122930444273.5000	33318219	147443869	90381044.0000
mpcugles	33	45	39.0000	31016495358	35003626018	33010060688.0000	39597740458	41086562856	40342151657.0000
parsec-mpi	789	804	796.5000	16286077	16398523	16342300.0000	58054593	58317925	58186259.0000
01OrgSeqDi	927	3305	2116.0000	9355380	28180782	18768081.0000	9933617	32927838	21430727.5000
perm_test	8	8	8.0000	1119071	1119071	1119071.0000	1081729	1081729	1081729.0000
fox	82	82	82.0000	6511991	6511991	6511991.0000	2975166	2975166	2975166.0000
sleep	1808	1808	1808.0000	16291051	16291051	16291051.0000	18145074	18145074	18145074.0000
vat_2d_109	840	840	840.0000	22163179	22163179	22163179.0000	17705905	17705905	17705905.0000
vasp.5.3.3	13688	13688	13688.0000	431464909	431464909	431464909.0000	7938926902	7938926902	7938926902.0000
parsec-mpi	3620	3620	3620.0000	39063354	39063354	39063354.0000	45792933	45792933	45792933.0000
init	66	66	66.0000	37229738	37229738	37229738.0000	5918652341	5918652341	5918652341.0000
parsec-mpi	3629	3629	3629.0000	39014635	39014635	39014635.0000	44383881	44383881	44383881.0000
parsec-mpi	3629	3629	3629.0000	39255223	39255223	39255223.0000	45976308	45976308	45976308.0000
foo	1590	1590	1590.0000	18636627	18636627	18636627.0000	18277572	18277572	18277572.0000
parsec-mpi	805	805	805.0000	16360195	16360195	16360195.0000	58254469	58254469	58254469.0000
astrobear2	2871	2871	2871.0000	611403446	611403446	611403446.0000	4660208021	4660208021	4660208021.0000
analysis_t	172810	172810	172810.0000	8614721733	8614721733	8614721733.0000	9771318879	9771318879	9771318879.0000
Sept1Test1	10824	10824	10824.0000	534910382	534910382	534910382.0000	1225383793	1225383793	1225383793.0000
StudsCase1	15143	15143	15143.0000	372437565	372437565	372437565.0000	1357439592	1357439592	1357439592.0000
vasp_533_w	1203	1203	1203.0000	85533081	85533081	85533081.0000	53102462	53102462	53102462.0000
parsec-mpi	791	791	791.0000	16223587	16223587	16223587.0000	58155917	58155917	58155917.0000
asp	370	370	370.0000	43890754	43890754	43890754.0000	15878948	15878948	15878948.0000
parsec-mpi	798	798	798.0000	16193566	16193566	16193566.0000	58160669	58160669	58160669.0000
postScalar	1257	1257	1257.0000	11747664039	11747664039	11747664039.0000	182226219	182226219	182226219.0000
parsec-mpi	797	797	797.0000	16406164	16406164	16406164.0000	58321569	58321569	58321569.0000
parsec-mpi	788	788	788.0000	16176689	16176689	16176689.0000	58015292	58015292	58015292.0000
gcrmio	568	568	568.0000	392817638	392817638	392817638.0000	369478383623	369478383623	369478383623.0000
parsec-mpi	805	805	805.0000	16316156	16316156	16316156.0000	58243377	58243377	58243377.0000
astrobear_	13355	13355	13355.0000	7982725437	7982725437	7982725437.0000	51230172980	51230172980	51230172980.0000
parsec-mpi	792	792	792.0000	16209495	16209495	16209495.0000	58045891	58045891	58045891.0000
parsec-mpi	798	798	798.0000	16195614	16195614	16195614.0000	58067000	58067000	58067000.0000
parsec-mpi	729	729	729.0000	15647969	15647969	15647969.0000	57346202	57346202	57346202.0000
pelfe_STAM	302	302	302.0000	13568837259	13568837259	13568837259.0000	614504947	614504947	614504947.0000
OpenSeesSP	154	154	154.0000	27464977	27464977	27464977.0000	1919693	1919693	1919693.0000
parsec-mpi	3602	3602	3602.0000	38912845	38912845	38912845.0000	45761769	45761769	45761769.0000
parsec-mpi	789	789	789.0000	16206252	16206252	16206252.0000	58068018	58068018	58068018.0000
pw2bgw.x	97	97	97.0000	53570241	53570241	53570241.0000	3410437	3410437	3410437.0000
runJKI	8	8	8.0000	181391	181391	181391.0000	153032	153032	153032.0000
messagesFa	9	9	9.0000	434352	434352	434352.0000	112832	112832	112832.0000
vat_2d_109	344	344	344.0000	17556664	17556664	17556664.0000	7723982	7723982	7723982.0000
parsec-mpi	950	950	950.0000	17630135	17630135	17630135.0000	59976226	59976226	59976226.0000
cql_rw_si	466	466	466.0000	829739984	829739984	829739984.0000	193241268	193241268	193241268.0000
word_count	2468	2468	2468.0000	45116382	45116382	45116382.0000	50632493	50632493	50632493.0000
parsec-mpi	802	802	802.0000	16359708	16359708	16359708.0000	58345322	58345322	58345322.0000
dag-try-1	5210	5210	5210.0000	43186358	43186358	43186358.0000	49998137	49998137	49998137.0000
Sept1Test2	36009	36009	36009.0000	608664264	608664264	608664264.0000	836525428	836525428	836525428.0000
parsec-mpi	3628	3628	3628.0000	39310966	39310966	39310966.0000	46119310	46119310	46119310.0000
vat_2d_109	473	473	473.0000	18828219	18828219	18828219.0000	10552738	10552738	10552738.0000
parsec-mpi	3606	3606	3606.0000	38439750	38439750	38439750.0000	36535112	36535112	36535112.0000
parsec-mpi	3605	3605	3605.0000	38981606	38981606	38981606.0000	45717321	45717321	45717321.0000
parsec-mpi	800	800	800.0000	16335902	16335902	16335902.0000	58255586	58255586	58255586.0000
enzo-jw.ex	172823	172823	172823.0000	134171406650	134171406650	134171406650.0000	1562872193320	1562872193320	1562872193320.0000
idsolve	14955	14955	14955.0000	3353159398	3353159398	3353159398.0000	86254897220	86254897220	86254897220.0000
heat3D_syn	14	14	14.0000	974357	974357	974357.0000	805906	805906	805906.0000
hello_2.0b	203	203	203.0000	6407854	6407854	6407854.0000	3142110	3142110	3142110.0000
runIJK	9	9	9.0000	179132	179132	179132.0000	151338	151338	151338.0000
NewAlgorit	1507	1507	1507.0000	21641171	21641171	21641171.0000	80364591	80364591	80364591.0000
hw2.2.exe	9	9	9.0000	398844	398844	398844.0000	354136	354136	354136.0000
cubep3m_dm	1803	1803	1803.0000	25902018399	25902018399	25902018399.0000	197697266	197697266	197697266.0000
parsec-mpi	244	244	244.0000	11203825	11203825	11203825.0000	51301182	51301182	51301182.0000
wrf-org.ex	39	39	39.0000	776169606	776169606	776169606.0000	971441744	971441744	971441744.0000
parsec-mpi	788	788	788.0000	16375537	16375537	16375537.0000	58261429	58261429	58261429.0000
Floating_J	22	22	22.0000	55558878	55558878	55558878.0000	9209128	9209128	9209128.0000
parsec-mpi	3621	3621	3621.0000	38542917	38542917	38542917.0000	34433979	34433979	34433979.0000
Sept1Test6	7216	7216	7216.0000	181257615	181257615	181257615.0000	408693551	408693551	408693551.0000
p12s03	27	27	27.0000	447412303	447412303	447412303.0000	235822568	235822568	235822568.0000
enzo-gnu.e	1952	1952	1952.0000	112335700183	112335700183	112335700183.0000	1647811120	1647811120	1647811120.0000
MWOB8_MWOB	30	30	30.0000	5559007297	5559007297	5559007297.0000	8611792	8611792	8611792.0000
ff	104273	104273	104273.0000	161354431859	161354431859	161354431859.0000	8736850018410	8736850018410	8736850018410.0000
parsec-mpi	789	789	789.0000	16224987	16224987	16224987.0000	58046070	58046070	58046070.0000
Sept1Test2	36016	36016	36016.0000	613499471	613499471	613499471.0000	889711787	889711787	889711787.0000

vhone	6270	6270	6270.0000	46923351250	46923351250	46923351250.0000	68247427892	68247427892	68247427892.0000
Sept1Test2	1706	1706	1706.0000	31626972	31626972	31626972.0000	57701678	57701678	57701678.0000
NewAlgorit	7211	7211	7211.0000	105942951	105942951	105942951.0000	501136495	501136495	501136495.0000
NewAlgorit	10827	10827	10827.0000	230527079	230527079	230527079.0000	561470086	561470086	561470086.0000
051514_sus	36	36	36.0000	2157027658	2157027658	2157027658.0000	185662288	185662288	185662288.0000
residence_	142	142	142.0000	3147592	3147592	3147592.0000	2411958	2411958	2411958.0000
Sept1Test2	141	141	141.0000	7472883	7472883	7472883.0000	7220751	7220751	7220751.0000
harris2dbl	9540	9540	9540.0000	127320714072	127320714072	127320714072.0000	376797817916	376797817916	376797817916.0000
parsec-mpi	796	796	796.0000	16170204	16170204	16170204.0000	58119935	58119935	58119935.0000
postScalar	1296	1296	1296.0000	11747031018	11747031018	11747031018.0000	182186592	182186592	182186592.0000
cd-svm	424	424	424.0000	76240500	76240500	76240500.0000	5492315	5492315	5492315.0000
permtest	7	7	7.0000	849047	849047	849047.0000	766522	766522	766522.0000
parsec-mpi	3627	3627	3627.0000	38800524	38800524	38800524.0000	40304580	40304580	40304580.0000
Charging3	10800	10800	10800.0000	263132464	263132464	263132464.0000	692021461	692021461	692021461.0000
parsec-mpi	3604	3604	3604.0000	39123580	39123580	39123580.0000	45941395	45941395	45941395.0000
harris2dbl	10930	10930	10930.0000	3202069554	3202069554	3202069554.0000	866769264344	866769264344	866769264344.0000
parsec-mpi	3601	3601	3601.0000	38861955	38861955	38861955.0000	45320734	45320734	45320734.0000
parsec-mpi	828	828	828.0000	16531672	16531672	16531672.0000	58432569	58432569	58432569.0000
biomol	17	17	17.0000	13299470	13299470	13299470.0000	1253832	1253832	1253832.0000
diagsav2mp	129	129	129.0000	470433186981	470433186981	470433186981.0000	885490517	885490517	885490517.0000
Sept1Test2	1944	1944	1944.0000	33270903	33270903	33270903.0000	63053023	63053023	63053023.0000
vasp_gamma	16	16	16.0000	36867500	36867500	36867500.0000	4152510	4152510	4152510.0000
Sept1Test2	36009	36009	36009.0000	590413968	590413968	590413968.0000	697812398	697812398	697812398.0000
parsec-mpi	3600	3600	3600.0000	38784974	38784974	38784974.0000	45465324	45465324	45465324.0000
parsec-mpi	789	789	789.0000	16403962	16403962	16403962.0000	58276698	58276698	58276698.0000
runIKJ	9	9	9.0000	181644	181644	181644.0000	152715	152715	152715.0000
genfel	36	36	36.0000	2081311	2081311	2081311.0000	383632	383632	383632.0000
vasp_std_v	86403	86403	86403.0000	6044782063	6044782063	6044782063.0000	10146505735	10146505735	10146505735.0000
h1202a	77114	77114	77114.0000	18668517011	18668517011	18668517011.0000	91104840804	91104840804	91104840804.0000
profileCha	2729	2729	2729.0000	12653969565	12653969565	12653969565.0000	28320387	28320387	28320387.0000
h0501b	68920	68920	68920.0000	18162828985	18162828985	18162828985.0000	102612181965	102612181965	102612181965.0000
ablDyMFoam	42	42	42.0000	535061528	535061528	535061528.0000	44192656	44192656	44192656.0000
Case5	150	150	150.0000	10632759	10632759	10632759.0000	14638111	14638111	14638111.0000
fd3d131014	3630	3630	3630.0000	129747473	129747473	129747473.0000	1175745704	1175745704	1175745704.0000
Stratified	315	315	315.0000	6480853454	6480853454	6480853454.0000	130359855	130359855	130359855.0000
Sept1Test5	337	337	337.0000	10371671	10371671	10371671.0000	12342547	12342547	12342547.0000
Sept1Test1	731	731	731.0000	16353659	16353659	16353659.0000	26886292	26886292	26886292.0000
Sept1Test3	3488	3488	3488.0000	61290811	61290811	61290811.0000	117530835	117530835	117530835.0000
Sept1Test2	29823	29823	29823.0000	3831786923	3831786923	3831786923.0000	5381786876	5381786876	5381786876.0000
Select_Par	3017	3017	3017.0000	9735997054140	9735997054140	9735997054140.0000	8344896120	8344896120	8344896120.0000
Sept1Test9	735	735	735.0000	25218619	25218619	25218619.0000	36502890	36502890	36502890.0000
epsilon.re	176	176	176.0000	16652203116	16652203116	16652203116.0000	3848050895	3848050895	3848050895.0000
parsec-mpi	786	786	786.0000	16207967	16207967	16207967.0000	58049757	58049757	58049757.0000
parsec-mpi	788	788	788.0000	16231499	16231499	16231499.0000	58092131	58092131	58092131.0000
openmc.sta	542	542	542.0000	13355872569	13355872569	13355872569.0000	41710314	41710314	41710314.0000
word_count	1152	1152	1152.0000	59565193198	59565193198	59565193198.0000	41433665	41433665	41433665.0000
openmc.ori	4491	4491	4491.0000	3362870462	3362870462	3362870462.0000	37556489	37556489	37556489.0000
StudsCase2	14761	14761	14761.0000	373349174	373349174	373349174.0000	1381683626	1381683626	1381683626.0000
Main.out	140	140	140.0000	5402522	5402522	5402522.0000	2770218	2770218	2770218.0000
parsec-mpi	3610	3610	3610.0000	39039653	39039653	39039653.0000	45980097	45980097	45980097.0000
openmc.2x_	5401	5401	5401.0000	3382756843	3382756843	3382756843.0000	56338340	56338340	56338340.0000
Sept1Test2	36011	36011	36011.0000	594000257	594000257	594000257.0000	727916824	727916824	727916824.0000
mhdf02	9124	9124	9124.0000	1514998488	1514998488	1514998488.0000	162338772635	162338772635	162338772635.0000
READ_WRFNE	7676	7676	7676.0000	277152490156	277152490156	277152490156.0000	247137362	247137362	247137362.0000
parsec-mpi	3615	3615	3615.0000	38654193	38654193	38654193.0000	37917252	37917252	37917252.0000
parsec-mpi	787	787	787.0000	16223016	16223016	16223016.0000	58074066	58074066	58074066.0000
vasp_5.2.1	43226	43226	43226.0000	365829196	365829196	365829196.0000	407465426	407465426	407465426.0000
parsec-mpi	795	795	795.0000	16404628	16404628	16404628.0000	58308660	58308660	58308660.0000
2D_needle_	13926	13926	13926.0000	5136084475	5136084475	5136084475.0000	516361696962	516361696962	516361696962.0000
parsec-mpi	797	797	797.0000	16282892	16282892	16282892.0000	58230737	58230737	58230737.0000
runSimsExe	3524	3524	3524.0000	80213579	80213579	80213579.0000	966482928	966482928	966482928.0000
Sept1Test2	1117	1117	1117.0000	31264880	31264880	31264880.0000	70340679	70340679	70340679.0000
parsec-mpi	3603	3603	3603.0000	38770683	38770683	38770683.0000	45080224	45080224	45080224.0000
d16903dl.x	19	19	19.0000	20404087	20404087	20404087.0000	3007840	3007840	3007840.0000
m	3603	3603	3603.0000	66109835	66109835	66109835.0000	35726768	35726768	35726768.0000
parsec-mpi	3612	3612	3612.0000	38587640	38587640	38587640.0000	36510573	36510573	36510573.0000
Compute_Ra	118	118	118.0000	700161446348	700161446348	700161446348.0000	334789967	334789967	334789967.0000
Fpi_hybrid	112	112	112.0000	2864601	2864601	2864601.0000	2392467	2392467	2392467.0000
camb	2808	2808	2808.0000	27620046	27620046	27620046.0000	28359193	28359193	28359193.0000
parsec-mpi	800	800	800.0000	16327668	16327668	16327668.0000	58223778	58223778	58223778.0000
parsec-mpi	3628	3628	3628.0000	38481079	38481079	38481079.0000	34400493	34400493	34400493.0000
parsec-mpi	806	806	806.0000	16373124	16373124	16373124.0000	58311302	58311302	58311302.0000

lstopo	11	11	11.0000	5368725	5368725	5368725.0000	734006	734006	734006.0000
parsec-mpi	788	788	788.0000	16244630	16244630	16244630.0000	58057077	58057077	58057077.0000
nitfp	72	72	72.0000	98221608	98221608	98221608.0000	2544712	2544712	2544712.0000
parsec-mpi	790	790	790.0000	16222631	16222631	16222631.0000	58086324	58086324	58086324.0000
Sept1Test2	36024	36024	36024.0000	613469324	613469324	613469324.0000	889670543	889670543	889670543.0000
parsec-mpi	798	798	798.0000	16215842	16215842	16215842.0000	58134695	58134695	58134695.0000
Sept1Test6	6805	6805	6805.0000	174543798	174543798	174543798.0000	407071215	407071215	407071215.0000
Sept1Test1	8276	8276	8276.0000	1196822469	1196822469	1196822469.0000	3270681134	3270681134	3270681134.0000
Sept1Test1	16	16	16.0000	3619352	3619352	3619352.0000	1232048	1232048	1232048.0000
test_sine_	181	181	181.0000	200937066	200937066	200937066.0000	121096146	121096146	121096146.0000
Sept1Test1	93	93	93.0000	6036736	6036736	6036736.0000	3935197	3935197	3935197.0000
Sept1Test1	404	404	404.0000	12717977	12717977	12717977.0000	18997428	18997428	18997428.0000
parsec-mpi	3621	3621	3621.0000	39349661	39349661	39349661.0000	45905424	45905424	45905424.0000
Sept1Test2	304	304	304.0000	12792360	12792360	12792360.0000	24804714	24804714	24804714.0000
cql_ro_ran	1103	1103	1103.0000	2009459039	2009459039	2009459039.0000	466468532	466468532	466468532.0000
mhd1	5978	5978	5978.0000	1183075329	1183075329	1183075329.0000	107412346246	107412346246	107412346246.0000
vis_2d_tns	190	190	190.0000	2094386501	2094386501	2094386501.0000	4984631399	4984631399	4984631399.0000
parsec-mpi	785	785	785.0000	16211834	16211834	16211834.0000	58039890	58039890	58039890.0000
parsec-mpi	3624	3624	3624.0000	38853708	38853708	38853708.0000	45520837	45520837	45520837.0000
parsec-mpi	791	791	791.0000	16211342	16211342	16211342.0000	58047429	58047429	58047429.0000
embryos_mp	600	600	600.0000	344682031041	344682031041	344682031041.0000	1894162935	1894162935	1894162935.0000
parsec-mpi	799	799	799.0000	16380338	16380338	16380338.0000	58310511	58310511	58310511.0000
dsmc	11849	11849	11849.0000	295634716007	295634716007	295634716007.0000	108347403968	108347403968	108347403968.0000
NewAlgorit	10821	10821	10821.0000	631576957	631576957	631576957.0000	3588039046	3588039046	3588039046.0000
NVT_100	186	186	186.0000	25447847	25447847	25447847.0000	27598965	27598965	27598965.0000
parsec-mpi	3627	3627	3627.0000	39284682	39284682	39284682.0000	46090737	46090737	46090737.0000
Sept1Test2	1912	1912	1912.0000	36663797	36663797	36663797.0000	46522772	46522772	46522772.0000
Sept1Test2	24913	24913	24913.0000	436458683	436458683	436458683.0000	682965995	682965995	682965995.0000
parsec-mpi	3610	3610	3610.0000	38838565	38838565	38838565.0000	45607422	45607422	45607422.0000
Compute_Wa	695	695	695.0000	4336816556390	4336816556390	4336816556390.0000	2065052235	2065052235	2065052235.0000
sander.bul	5412	5412	5412.0000	201767146	201767146	201767146.0000	348088641	348088641	348088641.0000
parsec-mpi	3628	3628	3628.0000	38512889	38512889	38512889.0000	34656115	34656115	34656115.0000
lassospg	1804	1804	1804.0000	30358266	30358266	30358266.0000	18511331	18511331	18511331.0000
parsec-mpi	3606	3606	3606.0000	39024909	39024909	39024909.0000	44501141	44501141	44501141.0000
parsec-mpi	850	850	850.0000	16786187	16786187	16786187.0000	58843444	58843444	58843444.0000
messagesAl	73	73	73.0000	788346	788346	788346.0000	821320	821320	821320.0000
fault	1805	1805	1805.0000	16659926	16659926	16659926.0000	17936471	17936471	17936471.0000
parsec-mpi	3603	3603	3603.0000	38506370	38506370	38506370.0000	44124447	44124447	44124447.0000
runJIK	10	10	10.0000	328902	328902	328902.0000	321656	321656	321656.0000
CalcSA	36023	36023	36023.0000	614267524	614267524	614267524.0000	891092976	891092976	891092976.0000
mhdf01	9417	9417	9417.0000	1534574469	1534574469	1534574469.0000	162361297754	162361297754	162361297754.0000
parsec-mpi	3622	3622	3622.0000	39116536	39116536	39116536.0000	46686891	46686891	46686891.0000
parsec-mpi	939	939	939.0000	17408174	17408174	17408174.0000	57584079	57584079	57584079.0000
parsec-mpi	3626	3626	3626.0000	39117328	39117328	39117328.0000	45709015	45709015	45709015.0000
ls	1807	1807	1807.0000	20056285	20056285	20056285.0000	20599284	20599284	20599284.0000
parsec-mpi	794	794	794.0000	16086025	16086025	16086025.0000	57958823	57958823	57958823.0000
parsec-mpi	799	799	799.0000	16382742	16382742	16382742.0000	58292045	58292045	58292045.0000
mpi_trade.	10	10	10.0000	200969	200969	200969.0000	120872	120872	120872.0000
vat_2d_109	377	377	377.0000	17897933	17897933	17897933.0000	9076438	9076438	9076438.0000
prob2.exe	10	10	10.0000	197438	197438	197438.0000	116472	116472	116472.0000
rose_fft_0	5420	5420	5420.0000	45619029	45619029	45619029.0000	52518621	52518621	52518621.0000
parsec-mpi	3609	3609	3609.0000	38861730	38861730	38861730.0000	45242563	45242563	45242563.0000
runKIJ	9	9	9.0000	187629	187629	187629.0000	159687	159687	159687.0000
HelloWorld	232	232	232.0000	3746360444	3746360444	3746360444.0000	1269917246	1269917246	1269917246.0000
acd.x	13	13	13.0000	28297189	28297189	28297189.0000	13112728	13112728	13112728.0000
NVT_380	11019	11019	11019.0000	376187868	376187868	376187868.0000	2163889505	2163889505	2163889505.0000
prasterbla	1485	1485	1485.0000	510386899468	510386899468	510386899468.0000	3741935716	3741935716	3741935716.0000
rism3d.sng	57857	57857	57857.0000	1138303073	1138303073	1138303073.0000	887210829	887210829	887210829.0000
mpiiotest	32	32	32.0000	60329685	60329685	60329685.0000	40007269120	40007269120	40007269120.0000
parsec-mpi	219	219	219.0000	11317515	11317515	11317515.0000	51190676	51190676	51190676.0000
mandelbrot	87	87	87.0000	646311478	646311478	646311478.0000	15355135	15355135	15355135.0000
sem_model_	218	218	218.0000	1942682973	1942682973	1942682973.0000	23255003	23255003	23255003.0000
Sept1Test2	1570	1570	1570.0000	29655621	29655621	29655621.0000	52883396	52883396	52883396.0000
Sept1Test2	140	140	140.0000	7671455	7671455	7671455.0000	7493439	7493439	7493439.0000
Sept1Test2	2877	2877	2877.0000	49385861	49385861	49385861.0000	96537538	96537538	96537538.0000
homework2p	9	9	9.0000	195502	195502	195502.0000	116240	116240	116240.0000
Sept1Test2	3483	3483	3483.0000	61992192	61992192	61992192.0000	119102677	119102677	119102677.0000
Sept1Test2	2329	2329	2329.0000	42088840	42088840	42088840.0000	78039162	78039162	78039162.0000
Sept1Test2	138	138	138.0000	7595189	7595189	7595189.0000	6552867	6552867	6552867.0000
Sept1Test2	137	137	137.0000	5822703	5822703	5822703.0000	7141374	7141374	7141374.0000
xsbig_mpi	24	24	24.0000	11344260	11344260	11344260.0000	1761144	1761144	1761144.0000
pmemd.cuda	16	16	16.0000	9620975	9620975	9620975.0000	1110022	1110022	1110022.0000

mvapich.x	41	41	41.0000	179626251	179626251	179626251.0000	46707008	46707008	46707008.0000
parsec-mpi	3620	3620	3620.0000	38302232	38302232	38302232.0000	34185569	34185569	34185569.0000
parsec-mpi	3613	3613	3613.0000	38316822	38316822	38316822.0000	34506873	34506873	34506873.0000
peulfs3.4.	97	97	97.0000	44986540	44986540	44986540.0000	5367817	5367817	5367817.0000
vat_2d_109	7227	7227	7227.0000	80559245	80559245	80559245.0000	142569444	142569444	142569444.0000
NewAlgorit	7209	7209	7209.0000	88633551	88633551	88633551.0000	277858405	277858405	277858405.0000
parsec-mpi	797	797	797.0000	16402705	16402705	16402705.0000	58321729	58321729	58321729.0000
parsec-mpi	852	852	852.0000	16807844	16807844	16807844.0000	58878804	58878804	58878804.0000
fmm_cheb	1168	1168	1168.0000	92723267	92723267	92723267.0000	94420780	94420780	94420780.0000
parsec-mpi	790	790	790.0000	16434924	16434924	16434924.0000	58305811	58305811	58305811.0000
parsec-mpi	799	799	799.0000	16387020	16387020	16387020.0000	58287873	58287873	58287873.0000
ex59	1807	1807	1807.0000	57747805	57747805	57747805.0000	36195420	36195420	36195420.0000
omp-als	41	41	41.0000	19003115	19003115	19003115.0000	1025458	1025458	1025458.0000
parsec-mpi	689	689	689.0000	15135335	15135335	15135335.0000	60253825	60253825	60253825.0000
parsec-mpi	796	796	796.0000	16412682	16412682	16412682.0000	58262007	58262007	58262007.0000
qi.exe	12264	12264	12264.0000	428198060	428198060	428198060.0000	489831805	489831805	489831805.0000
Driver.p	387	387	387.0000	79376142	79376142	79376142.0000	129009427	129009427	129009427.0000
BADDI1	54077	54077	54077.0000	598603507654	598603507654	598603507654.0000	343517511507	343517511507	343517511507.0000
a.818	246	246	246.0000	50176062	50176062	50176062.0000	38339552	38339552	38339552.0000
runKJI	9	9	9.0000	174456	174456	174456.0000	147653	147653	147653.0000
parsec-mpi	3620	3620	3620.0000	39070090	39070090	39070090.0000	45837099	45837099	45837099.0000
parsec-mpi	796	796	796.0000	16413914	16413914	16413914.0000	58281975	58281975	58281975.0000
parsec-mpi	802	802	802.0000	16403017	16403017	16403017.0000	58319385	58319385	58319385.0000
parsec-mpi	789	789	789.0000	16219247	16219247	16219247.0000	58055305	58055305	58055305.0000
procPFiles	209	209	209.0000	801433424	801433424	801433424.0000	5931853225	5931853225	5931853225.0000
parsec-mpi	799	799	799.0000	16278758	16278758	16278758.0000	58191676	58191676	58191676.0000
impi.x	45	45	45.0000	178568149	178568149	178568149.0000	47740424	47740424	47740424.0000
parsec-mpi	786	786	786.0000	16204238	16204238	16204238.0000	58056788	58056788	58056788.0000
interpolat	24	24	24.0000	8315925	8315925	8315925.0000	1142287558	1142287558	1142287558.0000
ex1	7207	7207	7207.0000	60116941	60116941	60116941.0000	69982517	69982517	69982517.0000
parsec-mpi	797	797	797.0000	16370744	16370744	16370744.0000	58296885	58296885	58296885.0000
N-GenIC	617	617	617.0000	281343627	281343627	281343627.0000	188200607961	188200607961	188200607961.0000
overlap	152	152	152.0000	2721033	2721033	2721033.0000	2712275	2712275	2712275.0000
parsec-mpi	3620	3620	3620.0000	39128138	39128138	39128138.0000	45743238	45743238	45743238.0000
Sept1Test2	1693	1693	1693.0000	29835435	29835435	29835435.0000	57429877	57429877	57429877.0000
Sept1Test1	139	139	139.0000	6017375	6017375	6017375.0000	7361437	7361437	7361437.0000
Sept1Test1	4966	4966	4966.0000	85239159	85239159	85239159.0000	169899877	169899877	169899877.0000
vat_3d_112	3608	3608	3608.0000	641363979	641363979	641363979.0000	35544103	35544103	35544103.0000
Case4	141	141	141.0000	8812876	8812876	8812876.0000	13891062	13891062	13891062.0000
vfmfe-unil	38	38	38.0000	184996286	184996286	184996286.0000	29013023	29013023	29013023.0000
parsec-mpi	280	280	280.0000	11752782	11752782	11752782.0000	51880976	51880976	51880976.0000
wget	15439	15439	15439.0000	153153353	153153353	153153353.0000	3856381634	3856381634	3856381634.0000
pelfe_STAM	63	63	63.0000	9375247860	9375247860	9375247860.0000	132427240	132427240	132427240.0000
parsec-mpi	3624	3624	3624.0000	39299140	39299140	39299140.0000	45871387	45871387	45871387.0000
vasp533_st	86402	86402	86402.0000	1472164351	1472164351	1472164351.0000	1654724706	1654724706	1654724706.0000
Sept1Test2	4972	4972	4972.0000	86347670	86347670	86347670.0000	170410294	170410294	170410294.0000
orion2_run	62234	62234	62234.0000	21472499948	21472499948	21472499948.0000	188627320225	188627320225	188627320225.0000
parsec-mpi	3600	3600	3600.0000	38715427	38715427	38715427.0000	45337994	45337994	45337994.0000
parsec-mpi	3602	3602	3602.0000	39039564	39039564	39039564.0000	45829365	45829365	45829365.0000
parsec-mpi	3628	3628	3628.0000	39260353	39260353	39260353.0000	46272422	46272422	46272422.0000
mhd2	5975	5975	5975.0000	1183388291	1183388291	1183388291.0000	107815252901	107815252901	107815252901.0000
parsec-mpi	788	788	788.0000	16207729	16207729	16207729.0000	58036572	58036572	58036572.0000
pagerank	100	100	100.0000	714340597	714340597	714340597.0000	7307939	7307939	7307939.0000
vasp.nonco	220	220	220.0000	53134157	53134157	53134157.0000	5599372	5599372	5599372.0000
PDF.CHUNK.	35	35	35.0000	6578888247	6578888247	6578888247.0000	4164059	4164059	4164059.0000
tacc_io	34	34	34.0000	144550850	144550850	144550850.0000	275035364950	275035364950	275035364950.0000
parsec-mpi	3610	3610	3610.0000	38515365	38515365	38515365.0000	34452331	34452331	34452331.0000
Sept1Test2	11542	11542	11542.0000	240064618	240064618	240064618.0000	689595892	689595892	689595892.0000
Sept1Test2	6446	6446	6446.0000	137730241	137730241	137730241.0000	377562235	377562235	377562235.0000
Sept1Test2	11136	11136	11136.0000	238612886	238612886	238612886.0000	684993399	684993399	684993399.0000
CAMx.andy.	306	306	306.0000	405085380	405085380	405085380.0000	259088853	259088853	259088853.0000
jump_3d	25	25	25.0000	3284707	3284707	3284707.0000	653031121	653031121	653031121.0000
parsec-mpi	800	800	800.0000	16404182	16404182	16404182.0000	58348724	58348724	58348724.0000
parsec-mpi	789	789	789.0000	16384152	16384152	16384152.0000	58271608	58271608	58271608.0000
fvcom_snap	24471	24471	24471.0000	3333610751	3333610751	3333610751.0000	79716252061	79716252061	79716252061.0000
Sept1Test1	27	27	27.0000	3314999	3314999	3314999.0000	967584	967584	967584.0000
Sept1Test1	2054	2054	2054.0000	35940465	35940465	35940465.0000	69342846	69342846	69342846.0000
parsec-mpi	3618	3618	3618.0000	39122401	39122401	39122401.0000	49791286	49791286	49791286.0000
parsec-mpi	3611	3611	3611.0000	39055014	39055014	39055014.0000	45584532	45584532	45584532.0000
HansCode	1399	1399	1399.0000	35433337	35433337	35433337.0000	83159126	83159126	83159126.0000
hw2	1818	1818	1818.0000	35003691	35003691	35003691.0000	36060280	36060280	36060280.0000
wrf.exe.1.	317	317	317.0000	679055071	679055071	679055071.0000	742236255	742236255	742236255.0000

parsec-mpi	3600	3600	3600.0000	38269557	38269557	38269557.0000	34213952	34213952	34213952.0000
parsec-mpi	3621	3621	3621.0000	39067637	39067637	39067637.0000	45788699	45788699	45788699.0000
test	1807	1807	1807.0000	15811382	15811382	15811382.0000	18098663	18098663	18098663.0000
parsec-mpi	605	605	605.0000	14646788	14646788	14646788.0000	55835111	55835111	55835111.0000
interpolat	196	196	196.0000	14191864	14191864	14191864.0000	7527921	7527921	7527921.0000
AbinitioRe	18	18	18.0000	181572410	181572410	181572410.0000	5117342	5117342	5117342.0000
parsec-mpi	3608	3608	3608.0000	38493035	38493035	38493035.0000	45925835	45925835	45925835.0000

Curriculum Vitae

Salvador Melendez was born in El Paso, Texas. The first son of Manuel Raul Melendez Molina and Teresa Lujan Escobar. He graduated from the Tecnologico de Monterrey, Chihuahua, Mexico, in the Fall of 2006. During the next four years, he joined a metal mechanic company in Chihuahua, where he served as a design engineer and software developer. In Spring of 2011, he joined the University of Texas at El Paso (UTEP) to pursue a Master's of Science in Computer Engineering. During his second year, Salvador started working as a research assistant in the *Networks Communication Laboratory* (NetLab) under the supervision of Dr. Michael McGarry. Salvador graduated from his master's degree in Fall 2012. After graduation, he was accepted into the Electrical and Computer Engineering Doctoral program at UTEP. Salvador successfully completed the PhD qualifying exams with the best score of his generation. In Spring of 2013, Salvador was selected as a HSF/Google scholar recipient and participated in the very first Google Scholar's Retreat. He also was awarded with the *Graduate Student Fellowship Incentive Award* from Graduate School at UTEP, and also he was recognized as the outstanding research assistant for the *Networks Communication Laboratory* from 2013 through 2015. In Fall 2013, Salvador accepted the position of teaching assistant for the course of Digital Design in the Electrical and Computer Engineering department. In Spring 2014, Salvador joined the *Army High Performance Computing Research Center* (AHPCRC) Laboratory as a research assistant under the supervision of Dr. Patricia Teller. He participated in two poster presentations at the ARL boot camp (San Jose, CA) and at the UTEP Graduate Research Expo (El Paso, TX). Salvador also contributed to the publications: 1) *The Effect of Network Delay Estimation Error on Computation Offloading Decisions* and 2) *Communication Patterns of Cloud Computing.*

Permament Address: Salvador Melendez
6501 McNutt Rd
Anthony, NM. 88021
USA