2013-01-01

Improved Efficiency of RNA Secondary Structure Prediction using Distributed Computing

Gerardo A. Cardenas
University of Texas at El Paso, gacardenas@utep.edu

IMPROVED EFFICIENCY OF RNA SECONDARY STRUCTURE PREDICTION USING DISTRIBUTED COMPUTING

GERARDO ALBERTO CARDENAS JAMES

Computational Science Program

APPROVED:

Ming-Ying Leung, Ph.D., Chair

Kyle L. Johnson, Ph.D.

Rodrigo Romero, Ph.D.

Benjamin C. Flores, Ph.D.
Dean of the Graduate School

Dedication

I would like to dedicate this thesis proposal to my beloved parents Ofelia James Martinez and Jose Luis Cardenas Arciniega, my wife Edith Castrellon Quezada, and my siblings Ofelia Cardenas James, Jose Luis Cardenas James, and Carlos Cardenas James for all their continued support and counsel during my life and career.

IMPROVED EFFICIENCY OF RNA SECONDARY STRUCTURE PREDICTION USING DISTRIBUTED COMPUTING

by

GERARDO ALBERTO CARDENAS JAMES, M. Sc.

THESIS

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

MASTER OF SCIENCE

Computational Science Program

THE UNIVERSITY OF TEXAS AT EL PASO

Spring 2013

Acknowledgements

I would like to express my sincere gratitude to Dr. Ming-Ying Leung for giving me all her guidance and support which has contributed towards my research work. To my committee, Dr. Kyle L. Johnson and Dr. Rodrigo Romero, I am extremely grateful for their valuable assistance and support throughout this project.

Abstract

The rapidly growing amounts of available biomolecular sequence data, which may represent information from small gene fragments to large complete genomes, have led to the a great need for powerful computational resources for data analysis and storage. With the decoding of the human and other genomes, RNA secondary structure prediction has become an important area of interest in biology and medicine because they help in understanding the mechanisms of many biological processes such as gene regulation and viral replication, and in designing RNA-based therapies to treat various diseases. Due to the complexity of their algorithms, many existing and upcoming computational tools for the prediction of RNA secondary structures, require large amounts of memory and processing time, and therefore can only handle RNA sequences of limited length. For example, the pknotsRG program, which can predict RNA secondary structures with pseudoknots, has a limitation of handling no more than 800 nucleotide at one time. However, many RNA, such as the RNA viral genomes, contains thousands of nucleotides, making secondary structure prediction impractical if not impossible. I will present an alternative approach, in which a cutting method to generate chunks of RNA sequences is first applied, then the pknotsRG program is used for prediction, and finally a high-throughput distributed batch computing system called HTCondor is used to reduce the waiting time for the RNA secondary structure prediction.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Biological significance of RNA

Ribonucleic acid (RNA) is a macromolecule that plays important roles in many biological functions. A major role of the RNA is to participate in protein synthesis. In the cell, the RNA acts as carriers of genetic information, catalysts in cellular processes and as a mediator in determining the expression level of genes by carrying the information from the DNA to the protein-building system. There are three main different types of RNA involved in the protein synthesis: messenger RNA (mRNA, transfer RNA (tRNA), and ribosomal RNA (rRNA). The mRNA makes a copy of the DNA information for protein synthesis, the tRNA translates mRNA sequence into amino acid sequence, and rRNA makes up to 55% of ribosomes and the rest of protein. Understanding the roles for the different types of RNA in the biological cellular processes can lead to develop new methods of diagnosis and understanding and treating disease.

RNA is also the genomic molecules in many viruses, some of which cause fatal diseases. Understanding the RNA in viruses is important for developing treatments and epidemiological control strategies to counteract the development and propagation of these diseases.

1.2 Secondary structure analysis

The secondary structure of RNA is the collection of hydrogen-bonded base pairs. The RNA constructs such sequence by folding back to itself forming base-pared segments that can yield structures. The secondary structures are made of two categories of structural elements: stem-loops and pseudoknots. The stem loops are structures where two regions of the sequence base pair each other forming a stem with some unpaired bases in the center forming a loop. The pseudoknots are complex structures that occurs when stretches of the same sequence interacts near each other. The stability of such structures resides, among other factors, in how the nucleotides are organized within the sequence forming base-stacking interactions, the base pairs interactions of A-U, C-G, and U-G, and the minimum free energy of the regions in the stem-loop structures

.

The function of an RNA molecule is determined by its structure. Hence, the significance of the secondary structure analysis lies in that it can help in determining the functionality of the RNA molecule as well as of many regulatory, catalytic, and structural processes of the cell. Besides from informing about the functionality of the RNA molecule, the secondary structure of RNA gives information about the domain structure of the molecule and location of important sites within the structure. The RNA structure prediction has two important roles. First, it helps in the interpretation of experiments related to the RNA function. Second, it helps to establish new experiments to probe function.

One of the problems with the RNA secondary structure prediction is the heavy handling and processing of long RNA sequences due to the limitation of computer hardware and software. Most of the existing RNA secondary structure prediction with pseudoknots programs process RNA sequences sequentially causing a decrease in the speed up of the RNA molecule prediction because each long chain of RNA sequence is processed one at a time. Furthermore, the prediction of long RNA sequences such as the RNA viruses, whose genomes are composed of RNA encoding a number of proteins, is usually impossible due to the limitation in sequence length that can be processed by the existing RNA prediction programs. In order to minimize the execution time, the RNA sequence can be broken up into several chunks, predicted individually, and assembled back to its original position to form the final predicted structure [24]. The HTCondor cluster can provide resources to manage the prediction of each RNA sequence chunk in parallel by distributing chunks to idle machines available in the cluster, allowing processing long length RNA sequences and improving the executing time.

1.3 HTCondor: High-throughput computing software

The HTCondor [18] software project was born at the University of Wisconsin-Madison (UW-Madison) with the sole purpose of having a new system for distributed computing capable of running computationally intensive jobs. It is classified as a high-throughput distributed batch computing system meaning that HTCondor has shared utilization of autonomous resources toward a common goal by exploiting all available computing resources efficiently. Given its batch computing system classification, the HTCondor system inherits all the batch system features such as job management, scheduling policy, priority scheme, resource monitoring, and resource management. One of the problems that most of the

distributed systems face is the lack of scalability. The features of HTCondor include flexibility, allowing scalability, easy to install [18].

The HTCondor system manages efficiently a queue of jobs when a single computer is available; however, HTCondor surpasses its efficiency when hundreds of computers take place in the pool. HTCondor is a job-scheduling system that allows distributing several large jobs based upon a policy that requires no user interaction on a grid-style computing in a parallel fashion with the purpose of running as many tasks as the grid is able to handle at the same time by optimizing the execution time and maximizing computational throughput. HTCondor creates a computer cluster from idle computer resources and utilizes the cluster efficiently to distribute and execute large jobs in either serial or parallel fashion. HTCondor decides when and where to send the jobs based on a job-scheduling process.

A job is submitted by the user to the HTCondor pool that determines which resource is free to execute the job based on pre-established security constraints and rules. The job submitted for execution in the HTCondor pool is described in a submit file. Such file contains location of the input, executable, and output files, the platform type required to run the program, how many times to run a job as well as the data in form of arguments (parameters) that will be processed during the execution. HTCondor sends the job to a remote idle machine along with a copy of the data that will act as parameters. The data integrity and security is maintained intact since the original data is kept on the machine where the job was submitted. HTCondor sends back the output results to the machine where HTCondor is running as a master and removes the executable and data files copied on the remote machine.

1.4 Statement of the Problems

The objective of this thesis is to demonstrate that the execution time for processing secondary structure prediction of large RNA molecules can be substantially reduced using the HTCondor grid computing technology. We attempt to address the following problems:

Problem 1. Given an RNA sequence along with the resulting chunks obtained from the segmentation process, how can we use our available computing resources to reduce the completion time for its secondary structure prediction?

The bioinformatics computer network at the University of Texas at El Paso includes over 21 computers with a total of over 64 processors distributed in two computer labs in different buildings with the purpose of providing shared computing resources for research and instructional activities. The HTCondor software has been installed on all computers available in the bioinformatics network, forming the "Bioinformatics Grid" to take advantage of the idle machines for job processing in parallel to reduce the completion time for the secondary structure prediction.

The HTCondor software has been installed on all computers available in the Bioinformatics Computing Lab (BCL) to take advantage of the idle machines for job processing in parallel to reduce the completion time for the secondary structure prediction.

Problem 2. HTCondor distributes jobs to idle processors based on its internal policy. As the number of processors in the HTCondor grid increases, at what rate does the waiting time to complete a secondary structure prediction job decreases?

The waiting time for the secondary structure prediction of RNA sequences using the pknotsRG program [5] increases when running sequentially as the number and length of the sequences becomes larger. For small datasets, the waiting time is not a significant problem since it can be done in minutes or even hours; however, for larger datasets sizes, the waiting time increases rapidly due to the extensive manipulation that needs to be done for their prediction. The objective of using a HTCondor pool is to reduce the waiting time for completion of the secondary structure prediction of a dataset of RNA sequences. Measurements are taken to determine whether how much the waiting time can be reduced with the addition of nodes to the HTCondor pool.

Chapter 2: Literature Review

2.1 Computational prediction algorithms for RNA secondary structures

The waiting time for the secondary structure prediction of RNA sequences using the pknotsRG program increases when running sequentially as the number and length of the sequences becomes larger. For small datasets, the waiting time is not a significant problem since it can be done in minutes or even hours; however, for larger datasets sizes, the waiting time increases rapidly due to the extensive manipulation that needs to be done for their prediction. The objective of using a HTCondor pool is to reduce the waiting time for completion of the secondary structure prediction of a dataset of RNA sequences. Measurements are taken to determine whether how much the waiting time can be reduced with the addition of nodes to the HTCondor pool.

Due to the great amount of RNA sequences available and their variability in size of sequences, computational algorithms for the prediction of RNA secondary structures have been developed since years ago to try to optimize their processing by minimizing the execution time. One well-known algorithm developed by Michael Zuker [25] used stacking and destabilizing energies to construct the secondary structure prediction of RNA based on the minimum free energy. The applied technique allowed improving the execution time by N^3 of pre-existing prediction algorithms such as 2^N for the RNA prediction using the thermodynamics by Pipas and McMahon [25] and N^5 for the program of Studnicka [25] where N is the number of nucleotides in the sequence that usually allowed folding sequences up to 800 nucleotides long. One particular characteristic of these algorithms is that they do not take into account pseudoknots structures which make the predication more complex and time consuming [25].

Most recently developed computer algorithms show a great improvement in comparison to rustic algorithms. The computer algorithm to predict common RNA secondary structure motifs including pseudoknots in unaligned sequences developed by Yongmei Ji [8] allows predicting mRNA structure motifs that play an important role in regulating gene expression. Depending on the technique used, its worst complexity falls into $O(m^n)$, where m is the maximum number of stems studied in one sequence and n the total number of sequences whereas the optimal complexity, by using greedy algorithms, falls

in O(m), where m is the total number of stem blocks in the graph. This complexity is faster but it only gives one structure and is not guaranteed to be the optimal [8].

Another dynamic programming algorithm developed by Rivas and Eddy to predict the optimal RNA secondary structure including pseudoknots, which have relevant function in the RNA processes, has a worst case complexity of $O(n^6)$, time and $O(n^4)$ in storage. This approach is based on thermodynamic so it generates the optimal minimum free energy structure considering pseudoknots.[15] However, the design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics presented by Jens Reeder take in consideration the pseudoknots as well; claiming to have an improved execution time in comparison to the Rivas and Eddy's approach. This recursive pseudoknots algorithm requires $O(n^4)$, time and $O(n^2)$ space to predict the energetically RNA optimal structure usually containing pseudoknots [13].

A recent approach is a dynamic programming algorithm for finding the optimal segmentation of RNA sequence in secondary structure predictions. It splits the RNA sequence into chunks, the final predicted structure. Its running time oscillates in $O(n^2)$, which makes this algorithm a faster method in the prediction on RNA structures [10].

2.2 Other Bioinformatics applications using HTCondor

Due to the exponential growth of datasets in the Bioinformatics field area, some studies have relied on HTCondor functionalities to process the humongous amount of information available to analyze and to acquire performance benefits. HTCondor has been applied in different science areas such the Bioinformatics area, which deals with a great amount of biological data available to address biological questions. For instance, a project called BioCompute has employed the HTCondor techniques for executing batch bioinformatics jobs. The use of HTCondor was to distribute input queries across the HTCondor pool with the solely objective of conducting large BLAST searches by improving the runtimes. In this project, a daily load of 2310 sequences were processed by the HTCondor cluster of 32 nodes in 20 hours in comparison to a single node that would have taken more than three weeks to process the same amount of sequences [3].

Proteomics is another field where HTCondor has been utilized for distributing the execution of the heuristic BLASTP algorithm together with a sliding window approach. A protein segment of n amino acids is used with the BLASTP algorithm to find its similarities. Once this segment has been processed by BLASTP, a window fraction is move one amino acid further down the protein sequence. The process is repeated until all possible regions of the protein sequence have been covered. This process ends up with an increasing amount of execution time because it is done sequentially. Thus, the execution time can be reduced by distributing different window sizes to different idle processors by using the HTCondor grid computing [2].

In the same area, the automatic annotation of Glycosylphosphatidylinositol (GPI) Structures is another study that has applied the HTCondor techniques on distributing chunks of information to idle machines to reduce its execution time. The automatic annotation tool is a library-search algorithm that helps in identifying GPIs by matching fragment peaks in the spectra with molecular masses obtained from theoretical GPI structures. This algorithm was tested using the protozoan parasite Trypanosoma cruzi, agent associated with the Chagas disease. The algorithm for the prediction of the T.cruzi dataset was executed sequentially using a Dell PowerEdge T300 with four 64-bit processors, 8GB RAM memory and CenOS Linux, taking up to 10 days of continuous work for its prediction. The same T.cruzi dataset along with the prediction algorithm have been submitted to the HTCondor pool, consisting of 31 processors in 64-bit machines running CentOS Linux with a speed ranging from 1 to 3.3 GHz and from 1 to 8 GB RAM, having as a result an improvement of 4 days in total to complete its prediction [1].

Chapter 3: Methodology

In order to improve the efficiency for predicting secondary structures of long RNA sequences, a sequence segmentation approach has been recently proposed [21-24]. The idea is to cut the long RNA sequence into shorter segments called chunks, carry out the prediction for the chunks individually, possibly in parallel, and then assemble the predicted chunk structures to form an overall predicted structure of the original sequence. In this thesis, I propose to do the structure prediction for the chunks

using distributed computing by first establishing an HTCondor pool encompassing all the Linux-based computers of the BCL.

3.1 Configuration of the Bioinformatics HTCondor pool

HTCondor does not require having any special hardware or software installed prior to its configuration, however, it is recommended to have sufficient memory and two CPU/cores primarily for the collector and negotiator daemons on the central manager. Moreover, it needs to have the HTCondor binary distribution installed on each computer participating in the HTCondor cluster. Currently, the BCL has the HTCondor software installed on a central manager called Biolinux20 with 4 CPU/cores and 8 GB of RAM running the collector and negotiator daemons solely and 20 computers called Biolinux01-Biolinux19 and Biolinux21, totaling 64 CPU/cores, using between 2 and 4 CPU/cores and between 2 and 8 GB of RAM (Appendix A).

The HTCondor installation includes a configuration file called condor_config which includes the specification of the name of the central manager, information about the machine configured, name of the pool, read-write privileges for each machine in the pool, daemons to run, and the ports where the communication takes place. It needs to be configured for both, the central manager (Figure 3.1) and the nodes participating in the pool (Figure 3.2).

```
## What machine is your central manager?
CONDOR_HOST = biolinux20.bioinformatics.utep.edu

## Where have you installed the bin, sbin and lib condor directories?
RELEASE_DIR = /usr/local/condor-7.9.1

## Where is the local condor directory for each host?
## This is where the local config file(s), logs and spool/execute directories are located
LOCAL_DIR = /usr/local/condor-7.9.1/local.$(HOSTNAME)

## When something goes wrong with condor at your site, who should get the email?
CONDOR_ADMIN = root1@biolinux01.bioinformatics.utep.edu

## Full path to a mail delivery program that understands that "-s"
## means you want to specify a subject:
MAIL = /bin/mailx

## Network domain parameters:
## Internet domain of machines sharing a common UID space.
UID_DOMAIN = bioinformatics.utep.edu

## Internet domain of machines sharing a common file system.
UID_DOMAIN = bioinformatics.utep.edu

## Internet domain of machines sharing a common file system.
FILESYSTEM_DOMAIN = bioinformatics.utep.edu

## The user/group ID <uid>.<gid> of the "Condor" user.
CONDOR_IDS = 1308.501

## Condor needs to create a few lock files to synchronize access to
## various log files.
LOCK = /tmp/condor-lock.$(HOSTNAME)0.70971278807065

## When is this machine willing to start a job?
START = TRUE

## When to suspend a job?
SUSPEND = FALSE

## When to nicely stop a job?
PREEMPT = FALSE

## When to instantaneously kill a preempting job
KILL = FALSE

DAEMON_LIST = MASTER, COLLECTOR, NEGOTIATOR

## Java parameters:
JAVA = /usr/bin/java
JAVA_MAXHEAP_ARGUMENT = -Xmx1024m
```

Figure 3.1: Example of Condor Manager Configuration File.

```
## What machine is your central manager?
CONDOR_HOST = biolinux20.bioinformatics.utep.edu

## Where have you installed the bin, sbin and lib condor directories?
RELEASE_DIR = /usr/local/condor-7.9.1

## Where is the local condor directory for each host?
## This is where the local config file(s), logs and spool/execute directories are located
LOCAL_DIR = /usr/local/condor-7.9.1/local.$(HOSTNAME)

## When something goes wrong with condor at your site, who should get the email?
CONDOR_ADMIN = root1@biolinux01.bioinformatics.utep.edu

## Full path to a mail delivery program that understands that "-s"
## means you want to specify a subject:
MAIL = /bin/mailx

## Network domain parameters:
## Internet domain of machines sharing a common UID space.
UID_DOMAIN = bioinformatics.utep.edu

## Internet domain of machines sharing a common file system.
UID_DOMAIN = bioinformatics.utep.edu

## Internet domain of machines sharing a common file system.
FILESYSTEM_DOMAIN = bioinformatics.utep.edu

## The user/group ID <uid>.<gid> of the "Condor" user.
CONDOR_IDS = 1308.501

## Condor needs to create a few lock files to synchronize access to
## various log files.
LOCK = /tmp/condor-lock.$(HOSTNAME)0.70971278807065

## When is this machine willing to start a job?
START = TRUE

## When to suspend a job?
SUSPEND = FALSE

## When to nicely stop a job?
PREEMPT = FALSE

## When to instantaneously kill a preempting job
KILL = FALSE

DAEMON_LIST = MASTER, SCHEDD, STARTD

## Java parameters:
JAVA = /usr/bin/java
JAVA_MAXHEAP_ARGUMENT = -Xmx1024m
```

Figure 3.2: Example of Condor Node Configuration File.

Once HTCondor configuration file has been set up, the condor_master command has to be executed to start the condor master deamon on both the central manger to activate the HTCondor pool by starting up the collector and negotiator, and each client node to enable each machine to participate in the HTCondor pool by running the scheduler and startd deamon.

3.2 The segmentation approach to RNA secondary structure prediction

The segmentation algorithm consists of dividing each sequence into one or more chunks based on the cutting methods described by Zhang et al. [24]. After all chunks of RNA sequences have been generated for each sequence, a Perl script is executed to generate the HTCondor submit description files which in turn distribute each chunk to idle processors participating in the HTCondor pool for its processing. Once all distributed chunks of RNA sequences have been processed and an output file has been obtained for each distributed chunk, another Perl script is used to concatenate each output file with respect to their original sequence in one single file to have the final predicted secondary structure for each sequence

3.2.1 Segmentation of RNA Sequences into chunks

A long RNA sequence can be segmented into short chunks by one of the three cutting methods, namely the regular, centered, and optimized methods. The regular method is a naïve method, just taking chunks of a fixed size starting from the 5' end of the RNA until the sequence is exhausted. The centered and optimized methods try to minimize the likelihood of losing significant structural information by strategically avoiding cutting the sequence in the middle of an inversion. For the performance testing in this thesis, I have used the naïve regular method just because of its convenience, but the same analysis can be done on chunks of sequences produced by any segmentation method. The chunks generated will be inputted to the pknotsRG program for secondary structure prediction using the computers in the HTCondor pool.

3.2.2 Generate the HTCondor submit file

HTCondor requires a formatted submit description file containing information about the job to be executed such as the physical path location of the executable program, the runtime environment, the

specification of the platform type required to run the program, the e-mail notification to be sent when a job is completed, and the arguments that will be used to process the job (Figure 3.3).

A Perl script called GenerateChunkSubmitFile_1.pl (Appendix A) has been developed to generate the submit description file. Once the Perl script is executed, two files are created. The first file called directories.txt is created containing a list of directory names where each one represents a set of chunks of a specific sequence. Furthermore, a second file called filenames.txt is created containing a list of file names where each one represents a chunk of sequence. Finally, the Perl script creates a submit description file containing the location of the pknotsRG executable program, the specification of the vanilla universe, the Linux platform type, the e-mail notification deactivated, the chunk sequence name with the ".out" extension as the output filename, and all the RNA sequence filename of each chunk as argument. The output will be named and placed on the location specified in the Perl script.

3.2.3 Prediction of RNA secondary structure using pknotsRG

The sequence segmentation results in a number of RNA chunks that need to be processed by the pknotsRG software to predict their secondary structure. To reduce the waiting time that would be needed to process the chunks sequentially, HTCondor is used. The submit file called RNAChunkPrediction.submit (Figure 3.2), generated as described in 3.1.3, is sent to the HTCondor pool using the condor_submit command for its processing. Once the condor_submit is processed, each job included executes the PredictChunkRNAStructure_2.pl script (Appendix B) to predict the RNA secondary structure using pknotsRG.

Table 3.1: RNA Secondary Structure: HTCondor Submit File

```
# Fasta Sequence File
executable = /export/home/phd/scripts/PredictChunkRNAStructure_2.pl
universe   = vanilla
requirements = ( Arch=="X86_64") && ( OpSys=="LINUX" )
notification = never

arguments  = 001[1,100]BBV.RNA1_L3G1M0C100Centered.fasta
queue

arguments  = 001[1,100]BBV.RNA1_L4G2M0C100Centered.fasta
```

```
queue
.
.
.
arguments  = 001[983,1082]BBV.RNA1_L5G0M0C100Centered.fasta

queue
```

3.2.4 Assembly of the overall predicted RNA structure

A Perl script called assembly_3.pl (Appendix C) has been developed to assemble each output generated by HTCondor. The assembly for each predicted chunk to build the final secondary structure prediction of each sequence on the different datasets is done purely by concatenating consecutively one chunk with another based on the sequence position numbers specified in the FASTA sequence name. If the sequence position numbers are not consecutive, gaps in the form of dashes ("-") are added to fill out the missing positions.

3.3 Measurement of waiting time reduction

The sequential execution time measurements of the RNA secondary structure prediction is performed using a Perl script called PredictChunkRNAStructureSEQ_2.pl (Appendix D) for a sample of sequence chunks of 800 nucleotides each from the nodamura virus genomic RNA2. The nodamura RNA2 sequence is used repeatedly to generate two datasets, respectively with 64 and 100 chunks. The measure of the execution time is done by using a Crontab job that monitors the condor queue and condor status per minute using the condor_q and condor_status commands, respectively. Thus, the recorded time might be up to 1 minute less than the actual time used. For the former, the measure is obtained by using and internal Perl function called localtime which is placed just before the code where the prediction of each RNA sequence chunk using pknotsRG program starts and just after the last chunk of RNA sequence has been predicted. For the latter, the measure begins just after the data fragmentation has been performed and finishes after the final predicted chunk structure has been generated.

Chapter 4: Results, discussion and future work

4.1 Results

The Bioinformatics HTCondor pool has been adapted to run on 2, 4, 8, 16, 32, and 64 nodes. On each run, a total of 64 chunks of RNA sequences with 800 nucleotides each were used to measure the waiting time of the RNA secondary structure prediction using the HTCondor pool. The waiting time has been measured when using both the traditional sequential execution of pknotsRG and the distributed computing design. Furthermore, the speed-up has been calculated based on the serial and parallel execution time to determine how fast the RNA secondary structure is predicted by distributing chunks versus predicting it sequentially. Speed-up is calculated by dividing the elapsed time needed by one processor (sequential execution) divided by the time needed on *p* processors [9]. Finally, the efficiency has been calculated to determine how effectively the CPU's are being used versus how much time is spent on synchronization and communication [9]. Efficiency is calculated by dividing the speed-up achieved by the number of processors used.

Sequentially, the prediction of 64 chunks of RNA took approximately 178 minutes whereas the HTCondor pool with 64 processors took around 4 minutes (Table 4.1).

Table 4.1: Waiting time, speed-up, and efficiency measurements using 64 chunks.

Nodes	Time (min)	Speed-up	Efficiency
1	178	1.00	1.00
2	94	1.89	0.95
4	59	3.02	0.75
8	25	7.12	0.89
16	12	14.83	0.93
32	6	29.67	0.93
64	4	44.50	0.70

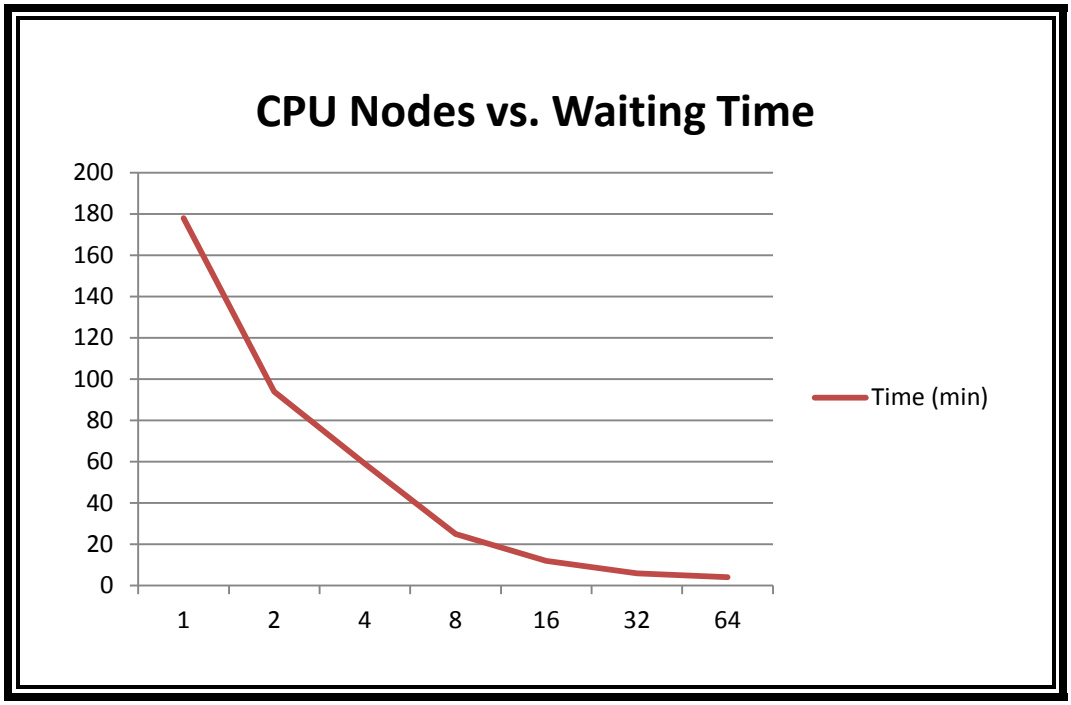

Figure 4.1: Graph of CPU nodes vs. waiting time using 64 chunks.

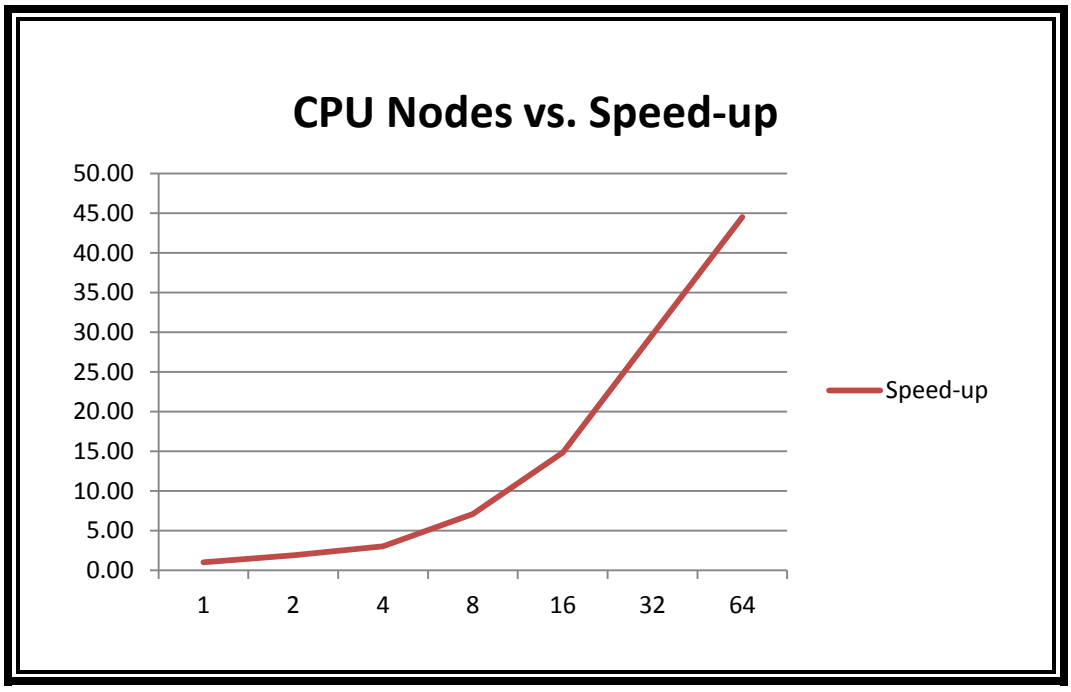

Figure 4.2: Graph of CPU nodes vs. speed-up using 64 chunks.

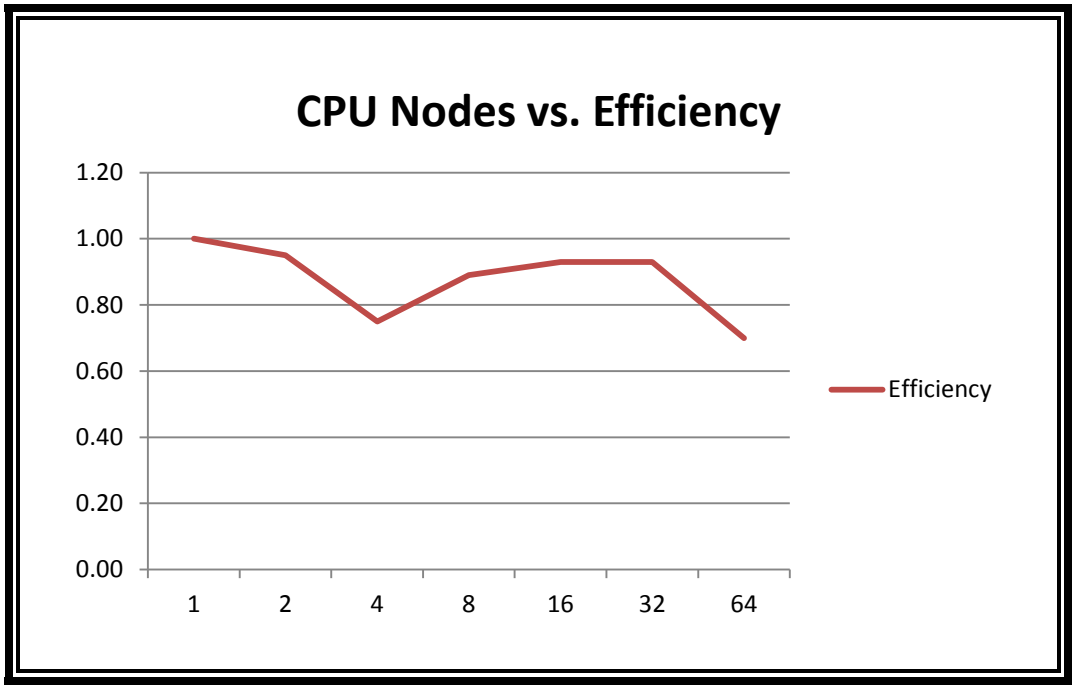

Figure 4.3: Graph of CPU nodes vs. efficiency using 64 chunks.

A second test has been performed with a total of 100 chunks of RNA sequences with 800 nucleotides each. The sequential execution took approximately 277 minutes whereas HTCondor with 64 processors took around 21 minutes (Table 4.2).

Table 4.2: Waiting time, speed-up, and efficiency measurements using 100 chunks.

Nodes	Time (min)	Speed-up	Efficiency
1	277	1.00	1.00
2	256	1.08	0.54
4	175	1.58	0.40
8	73	3.79	0.47
16	36	7.69	0.48
32	27	10.26	0.32
64	21	13.19	0.21

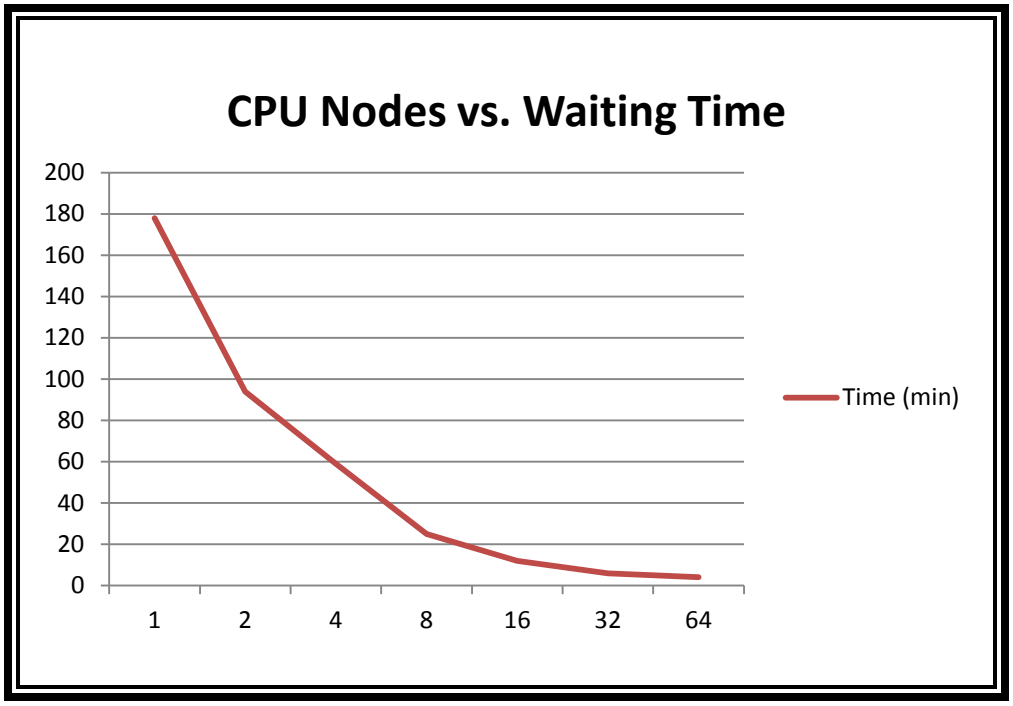

Figure 4.4: Graph of CPU nodes vs. waiting time using 100 chunks.

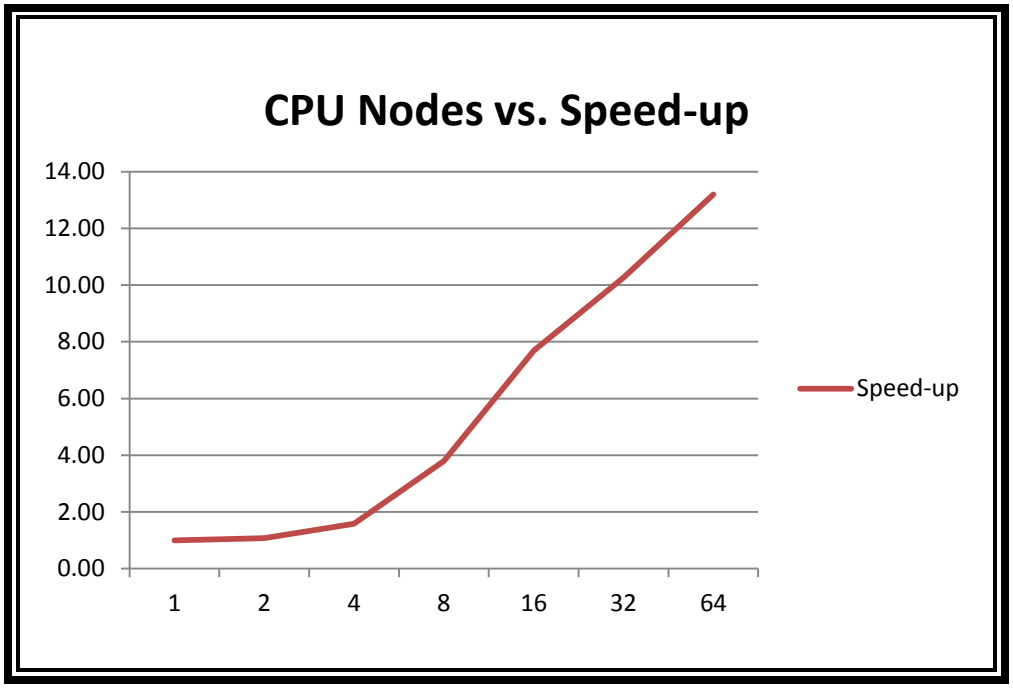

Figure 4.5: Graph of CPU nodes vs. speed-up time using 100 chunks.

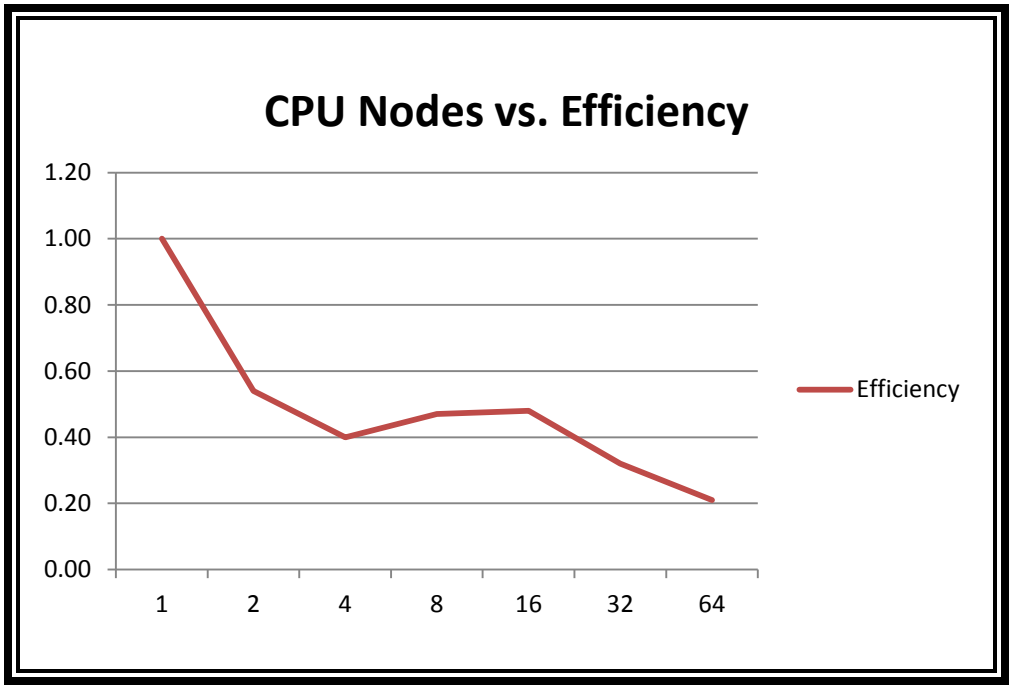

Figure 4.6: Graph of CPU nodes vs. efficiency time using 100 chunks.

Clearly, the waiting time has been reduced tremendously as the number of nodes was increased.

4.2 Discussion

One of the problems with the RNA Secondary Structure prediction is the heavy handling and processing of long RNA sequences due to the limitation of computer hardware and software. The pknotsRG program was able to process long RNA sequences by cutting them into chunks of no more than 800. However, the basic HTCondor design does not provide any mechanism to measure the actual CPU execution time, but only the real waiting time. This is because the central manager is not able to monitor every node activity to avoid a large overhead in communication. As the number of nodes increases, the Bioinformatics HTCondor pool was able to reduce the waiting time, in comparison to the sequential execution, from 180 to 5 minutes and 277 to 25 minutes when 16 and 25 sequence datasets were processed, respectively. The waiting time using HTCondor distributed computing was not measured with higher accuracy because HTCondor does not offer the capability of monitoring the status queue in a smaller time scale, easily. Hence, a crontab job was used to check the condor queue every minute which also caused loss of accuracy in the waiting time measurement.

Due to the overhead caused by the HTCondor communication, the frequent monitoring of the HTCondor status per minute, and the number of chunks submitted to HTCondor, the waiting time increased at some degree. However, it could be controlled by modifying the maximum chunk-length in the cutting method and the maximum jobs (shadows) running on each node, the scheduler queue size, and how and when to run jobs in the HTCondor configuration file.

4.3 Recommendations and future work

4.3.1 HTCondor Improvements.

For future work, several grid configurations will be done on the BCL HTCondor pool to determine which way would to improve the speed-up and efficiency since it is expected to have thousands of chunks to be predicted. A first configuration would be to set in place a powerful server as a central manager to have an efficient management in both the collection of information about every node participating in the pool and the negotiation (match-making) between the nodes participating in the pool and the number of jobs, based on a policy, to be processed. A second configuration would be to set in place a powerful as a node acting as a submitter and executer machine to have an efficient management in both the submission of jobs accumulated in a local queue to the central manager and the execution of queued jobs sent back from central manager for processing. Furthermore, the configuration files for the submit machines will be adjusted allowing a better control in the local queue of nodes and the maximum jobs that can be running simultaneously.

4.3.2 Datasets

For further performance testing with real RNA data using distributed computing on the established HTCondor pool, we will use two datasets. The first is a collection of Rfam sequences with known secondary structures containing pseudoknots, listed in Table 4.3. Rfam is a secondary structure database containing information about non-coding RNA families obtained from the use of sequence alignments and covariance models. Each family contained in the Rfam database represents a group of RNA sequences that function at the RNA level such as tRNA, microRNAs, and spliceosomal RNAs [6].

The second dataset is the 14 RNA genome sequences (see Table 4.4) from the family of nodaviruses. Nodaviruses are non-enveloped, icosahedral, RNA viruses with diameters in the range of 25-34 nm considered the causative agents of viral nervous necrosis of marine fish. The diseases produced by the Nodaviruses involve the central nervous system and the retina where they usually produce vacoulation and cell necrosis [21, 22].

Table 4.3: Names and lengths of 16 Rfam sequences with pseudoknots

Sequence Name	Length
RF00010_A.bpseq	312
RF00024_A.bpseq	451
RF00061_B.bpseq	323
RF00094_A.bpseq	89
RF00140_B.bpseq	112
RF00165_A.bpseq	62
RF00216_A.bpseq	302
RF00233_B.bpseq	84
RF00259_A.bpseq	169
RF00261_B.bpseq	221
RF00381_B.bpseq	59
RF00390_A.bpseq	23
RF00458_A.bpseq	202
RF00499_A.bpseq	27
RF00505_A.bpseq	64
RF00507_B.bpseq	79

Table 4.4: Names and lengths of 14 nodavirus sequences with pseudoknots

Sequence Name	Length
BBV.RNA1_C100Regular	3106
BBV.RNA2_C100Regular	1399
BoV.RNA1_C100Regular	3097
BoV.RNA2_C100Regular	1322
ETNN.RNA1_C100Regular	3103
ETNN.RNA2_C100Regular	1433
FHV.RNA1_C100Regular	3107
FHV.RNA2_C100Regular	1400
NoV.RNA1_C100Regular	3204
NoV.RNA2_C100Regular	1336
PAV.RNA1_C100Regular	3011
PAV.RNA2_C100Regular	1311
SJV.RNA1_C100Regular	3107

SJV.RNA2_C100Regular	1421

4.3.2 Improving the prediction of RNA secondary structures.

Currently, the segmentation approach for RNA secondary structure prediction only considers secondary structures formed within a single chunk after a long RNA sequence has been cut into chunks. It will a natural extension to allow formation of possible structures spanning multiple chunks. This process will be even more computationally intensive. If we allow up r chunks to interact, the number of predictions required increases combinatorially to

$$\sum_{i=1}^{r} \binom{n}{i}$$

the length of sequence in each prediction will also proportionally increase.

Furthermore, when interactions among multiple chunks are allowed in the prediction process, we would expect to see overlapping predicted structures that would not be in agreement. An efficient assembly algorithm to construct the final predicted structure based on the best consensus among all the predictions will be needed

4.3.3 Assembly algorithm considering overlapping structures.

Due to the consideration of overlapping structures when predicting, I will createan efficient assembly algorithm to construct the final predicted structure from possibly overlapping structures predicted in groups of chunks.

References

[1] Aguilar-Bonavides, Clemente, Gerardo A. Cardenas, E. S. Nakayasu, F. G. Lopes, Igor C. Almeida, Ming-Ying Leung, “Automatic Annotation of GPI Structures Using Grid Computing”. 5th International Conference on Bioinformatics and Computational Biology (BICoB).

[2] Andrade, Jorge, "Grid and High-Performance Computing for Applied Bioinformatics", Royal Institute of Technology, 2007.

[3] Braga-Henebry, Patrick, "BioCompute: Harnessing Distributed Systems for Bioinformatics Applications", May 2009.

[4] Cuevas, Jose M., Pilar Domingo-Calap, Marianoel Pereira-Gomez and Rafael Sanjuan*, "Experimental Evolution and Population Genetics of RNA Viruses", The Open Evolution Journal, 2009, p. 9-16.

[5] Dirks, Robert M., Justin M. Bois, Joseph M. Shaeffer, Erik Winfree, Niles A. Pierce, "Thermodynamic Analysis of Interacting Nucleic Acid Strand". SIAM Review. Society for Industrial and Applied Mathematics, 2007, p.65-88.

[6] Gardner, Paul P., Jennifer Daub., John Tate, Benjamin L. Moore, Isabelle H. Osuch, Sam Griffiths-Jones, Robert D. Finn, Eric P. Nawrocki, Diana L. Kolbe, Sean R. Eddy, Alex Bateman., “Rfam: Wikipedia, clans and the decimal release”. Nucleic Acids Research. Vol. 39. D141-D145.

[7] Gjerde, Douglas T, Lee Hoang, David Hornby. "RNA Purification and Analysis", Wiley-VCH, 2009, p.1-16.

[8] Ji, Yongmei, Xu, Xing, and Stormo, Gary D. "A graph theoretical approach to predict common RNA secondary structure motifs including pseudoknots in unaligned sequences", Oxford University Press, February 2004.

[9] Karp, Alan H., Horace P. Flatt. "Measuring Parallel Processor Performance", Communications of the ACM, 1990. Vol. 33. p.539-543.

[10] Licon, A., M. Taufer, M.-Y. Leung, and K. Johnson, “A dynamic programming algorithm for finding the optimal segmentation of an RNA secondary structure prediction,” in Proc. of the International Conference on Bioinformatics and Computational Biology, March 2010.

[11] Mubnday, B.L. and T. Nakai. “Special topic review: Nodaviruses as pathogens in larval and juvenile marine finfish.” World Journal of Microbiology & Biotechnology, 1997. Vol 13. p. 375-381.

[12] Nishizawa, T., M. Furuhasi, T. Nagai, T. Nakai, K. Muroga. “Genomic Classification of Fish Nodaviruses by Molecular Phylogenetic Analysis of the Coat Protein Gene.” Appl. Environ. Microbiol, 1997. p. 1633-1636.

[13] Reeder, J. and Giegerich, R. “Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics.” BMC Bioinformatics, 2004. p. 1-12.

[14] Reeder, Jens, Peter Steffen and Robert Giegerich. pknotsRG: RNA pseudoknot folding including near-optimal structures and sliding windows” Nucleic Acids Res., 35, W320-W324.

[15] Rivas E., Eddy S.R., "A Dynamic Programming Algorithm for RNA Structure Prediction Including Pseudoknots", Academic Press, 1999.

[16] Seetin, Matthew G., David H. Mathews, "RNA Structure Prediction: An Overview of Methods", Methods in Molecular Biology, Volume 905, 2012, p. 99-122.

[17] Taufer, Michela., Abel Licon, Roberto Araiza, David Mireles, F. H. D. van Batenburg, Alexander P. Gultyaev, Ming-Ying Leung, "PseudoBase++: an extension of PseudoBase for easy searching, formatting and visualization of pseudoknots", Nucleic Acids Research, 2009. D127-D135.

[18] Thain, Douglas, Todd Tannenbaum, Miron Livny. "Distributed Computing in Practice: The Condor Experience", Vol. 17, No. 2-4, 323-356.

[19] van Batenburg, F. H. D., A. P. Gultyaev, C. W. A. Pleij, J. Ng, J. Oliehoek. "PseudoBase: a database with RNA pseudoknots". Nucleic Acids Research, 2000. p. 201–204.

[20] Watson, James D., Tania A. Baker, Stephen P. Bell, Alexander Gann, Michael Levine, Richard Losick, "Molecular Biology of the Gene", Pearson Education, 2003, p.1-33.

[21] Yehdego, D. T., Zhang, B., Kodimala, V. K., Vegesna, R., Johnson, K. L., Taufer, M., Leung, M.-Y., (to appear 2013). Secondary structure predictions for long RNA sequences based on inversion excursions and MapReduce. Proceedings of the 12th IEEE International Workshop on High Performance Computational Biology (HiCOMB), Boston, May 20 2013.

[22] Yehdego, D. T., Kodimala, V. K., Viswakula, S., Zhang, B., Vegesna, R., Johnson, K.L., Taufer, M., Leung, M.-Y. (2012). Secondary Structure Predictions for Long RNA Sequences. Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine. pp. 545-547. New York, NY.

[23] Zhang, B., Yehdego, D. T., Johnson, K. L., Leung, M.-Y., Taufer, M., (2012). A Modularized MapReduce Framework to Support RNA Secondary Structure. (pp. 1-8). IEEE: 2012 IEEE International Conference on Bioinformatics and Biomedicine (BIBM).

[24] Zhang, B., Yehdego, D. T., Johnson, K. L., Leung, M.-Y., and Taufer, M., (to appear 2013) Enhancement of Accuracy and Efficiency for RNA Secondary Structure Prediction by Sequence Segmentation and MapReduce, BMC Structural Biology.

[25] Zuker M., Stiegler P., "Optimal Computer folding of large RNA sequences using thermodynamics and auxiliary information", Nucleic Acids Research 9, 1981, p.133-148.

Appendices

APPENDIX A. BCL HETEROGENEOUS COMPUTING ENVIRONMENT

Machine	OS	CPU	RAM
Biolinux01	CentOS 5.3	Intel® Core™2 Duo CPU 2.33GHz	5 GB
Biolinux02	CentOS 5.6	Intel® Core™2 Duo CPU 2.33GHz	5 GB
Biolinux03	CentOS 5.6	Intel® Core™2 Duo CPU 2.33GHz	5 GB
Biolinux04	CentOS 5.5	Intel® Core™2 Duo CPU 2.33GHz	5 GB
Biolinux05	CentOS 5.5	Intel® Core™2 Duo CPU 2.93GHz	8 GB
Biolinux06	CentOS 5.3	Dual-Core AMD Opteron™ CPU 1000MHz	2 GB
Biolinux07	CentOS 5.5	Intel® Core™2 Duo CPU 2.93GHz	8 GB
Biolinux08	CentOS 5.5	Intel® Core™2 Duo CPU 2.33GHz	6 GB
Biolinux09	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB
Biolinux10	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB
Biolinux11	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB
Biolinux12	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB
Biolinux13	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB
Biolinux14	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB
Biolinux15	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB
Biolinux16	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB
Biolinux17	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB
Biolinux18	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB
Biolinux19	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB
Biolinux20	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB
Biolinux21	CentOS 6.3	Intel® Core™ i3-2120 CPU 3.30GHz	8 GB

APPENDIX B. GENERATECHUNKSUBMITFILE_1.PL PERL SCRIPT

```
##############################################################################
# Author: Gerardo Cardenas <gacardenas@utep.edu>
# Pre-Execution: GenerateSingleFastaSeq_0.pl
# Mid-Execution: None
# Post-Execution: PredictChunkRNAStructure_2.pl through Condor Submit File
# Description: This script generates the submit file for HTCondor to use.
#              It creates a submit file containing filenames of chunks of
#              sequences as arguments.
##############################################################################
#----------------
# PROGRAM STARTS
#----------------

#!/usr/bin/perl
use strict;

# Generate file containing the directory names for each set of sequences.

  if ( !-d "Datasets")
  {
    print "\nDatasets directory does not exists!\n\n";
    exit;
  }

  system("ls -l Datasets | egrep '^d' | awk '{ print \$9 }' > directories.txt");

# Load the directory's file into memory
  my($directoryName) = "directories.txt";

  open (FILE,$directoryName) or die $!;
    my @fastaDirectoriesList = <FILE>;
  close(FILE);

# Iterate for each Fasta directory
  #for(my $i=0;$i<=0;$i++)
  for(my $i=0;$i<=scalar(@fastaDirectoriesList)-1;$i++)
  {
    my ($fastaDirectoryName) = $fastaDirectoriesList[$i];

    chomp($fastaDirectoryName);

    # Create Output directory
    my ($OutputPrediction) =
"Datasets/".$fastaDirectoryName."/".$fastaDirectoryName."_Output";
```

```
    if ( !-d $OutputPrediction)
    {
      mkdir($OutputPrediction);
    }

    # Generate file containing the filenames for every chunk.
      my ($file) = "filenames.txt";
      my ($fileLocation) =
"Datasets/$fastaDirectoryName/$fastaDirectoryName"."_SingleFastaSeq";

      if ( !-d $fileLocation)
      {
        print "\nERROR: The Datasets directory does not contain sequences in Fasta format.";
        print "\nERROR: Please make sure the complete sequences (NO CHUNKS) are in the
Datasets";
        print "\nERROR: directory in FASTA format in the format
Datasets\<MethodDir>\<MethodDir>_Fasta\n\n";
        exit;
      }

      my $dir = qx{ls $fileLocation};

    # Load filename's content into memory
      open(my $OUT, ">", 'filenames.txt') or die $!;
        print $OUT $dir;
      close $OUT;

      open(FILE, $file) or die $!;
        my (@filenames) = <FILE>;
      close(FILE);

    # Create submit file structure
      my ($submitfile) = "RNAChunkPrediction.submit";
      my ($description) = "Fasta Sequence File";
      my ($executable) =
"/export/home/gacardenas/Desktop/RNAProjectv3/PredictChunkRNAStructure_2.pl";

      open (SUBMITFILE, "> $submitfile");
        print SUBMITFILE "# ".$description."\n";
        print SUBMITFILE "executable = ".$executable."\n";
        print SUBMITFILE "universe   = vanilla"."\n";
        print SUBMITFILE "requirements = ( Arch==\"X86_64\") && ( OpSys==\"LINUX\"
)\n";
        print SUBMITFILE "when_to_transfer_output  = ON_EXIT\n";
        print SUBMITFILE "notification = never\n\n";
        for(my $i=0;$i<=scalar(@filenames)-1;$i++)
        {
          my ($filename) = $filenames[$i];
```

```
            chomp($filename);

            # Match a dot, followed by any number of non-dots until the end of the line.
              my ($ext) = $filename =~ /(\.[^.]+)$/;

            if (($ext eq ".fasta") || ($ext eq ".fas"))
            {
              print SUBMITFILE "arguments  = ".$filename."\n";
              print SUBMITFILE "queue\n\n";
            }
          }
        close (SUBMITFILE);
    }
#----------------
# PROGRAM ENDS
#----------------

########################
# FUNCTION DEFINITIONS
########################

# Remove extension from a filename
sub remove_extension {
    my $filename = shift @_;

    $filename =~ s/
                (.)          # matches any character
                \.           # the literal dot starting an extension
                [^.]+          # one or more NON-dots
                $            # end of the string
        /$1/x;

    return $filename;
}

# Remove Fasta name and get just the Fasta sequence
sub get_fasta{
  my $filename = $_[0];

  open(FILE, $filename) or die("Cannot open FASTA file.\n");
  my %seqs;
  my $header;
  my $first = 0;
  my @lines= <FILE>;
  foreach my $line(@lines){
  chomp($line);

  if ($line =~ /^>/){
```

```
    $header = $line;
    #print "HEADER1: $header\n\n";
    $header =~ s/^>//;
    #print "HEADER2: $header\n\n";
    $header =~ s/\s.*//;
    #print "HEADER3: $header\n\n";
    if ($first == 0){
      $first = 1;
    }
    next;
  }
  if ($first == 0){ die("Not FASTA file.\n"); }
    $seqs{$header} = $seqs{$header}.$line;
    #print "HASH: $seqs{$header}\n\n";
  }
  close(FILE);
  return \%seqs;
}
#########################
```

APPENDIX C. PREDICTCHUNKRNASTRUCTURE_2.PL PERL SCRIPT

```
#!/usr/bin/perl
use strict;

###############################################################################
# Author: Gerardo Cardenas <gacardenas@utep.edu>
# Pre-Execution: condor_submit RNAChunkPrediction.submit
# Mid-Execution: None
# Post-Execution: Assembly_3.pl
# Description: This script uses the submit file to submit it to the Condor
#                    Pool, and store the prediction of chunks in the output directory.
#                    specified in the code below.
###############################################################################
#----------------
# PROGRAM STARTS
#----------------

# Generate file containing the directory names for each set of sequences.

  if ( !-d "Datasets")
  {
    print "\nDatasets directory does not exists!\n\n";
    exit;
  }
```

```
  system("ls -l Datasets | egrep '^d' | awk '{ print \$9 }' > directories.txt");

# Load the directory's file into memory
  my($directoryName) = "directories.txt";

  open (FILE,$directoryName) or die $!;
    my @fastaDirectoriesList = <FILE>;
  close(FILE);

# Process each RNA Chunk using pknotsRG
  #for(my $i=0;$i<=0;$i++)
  for(my $i=0;$i<=scalar(@fastaDirectoriesList)-1;$i++)
  {
    my ($fastaDirectoryName) = $fastaDirectoriesList[$i];

    chomp($fastaDirectoryName);

    # Check if Single Fasta file directory Output exists
    my ($SingleSeqFastaDir) =
"Datasets/".$fastaDirectoryName."/".$fastaDirectoryName."_SingleFastaSeq";
```

APPENDIX D. ASSEMBLY_3.PL PERL SCRIPT

```
#!/usr/bin/perl
use strict;

##############################################################################
# Author: Gerardo Cardenas <gacardenas@utep.edu>
# Pre-Execution: condor_submit RNAChunkPrediction.submit
# Mid-Execution: None
# Post-Execution: Assembly_3.pl
# Description: This script uses the submit file to submit it to the Condor
#                    Pool, and store the prediction of chunks in the output directory.
#                    specified in the code below.
##############################################################################
#----------------
# PROGRAM STARTS
#----------------

# Generate file containing the directory names for each set of sequences.

  if ( !-d "Datasets")
  {
    print "\nDatasets directory does not exists!\n\n";
    exit;
```

```
  }

  system("ls -l Datasets | egrep '^d' | awk '{ print \$9 }' > directories.txt");

# Load the directory's file into memory
  my($directoryName) = "directories.txt";

  open (FILE,$directoryName) or die $!;
    my @fastaDirectoriesList = <FILE>;
  close(FILE);

# Process each RNA Chunk using pknotsRG
  #for(my $i=0;$i<=0;$i++)
  for(my $i=0;$i<=scalar(@fastaDirectoriesList)-1;$i++)
  {
    my ($fastaDirectoryName) = $fastaDirectoriesList[$i];

    chomp($fastaDirectoryName);

    # Check if Single Fasta file directory Output exists
    my ($SingleSeqFastaDir) =
"Datasets/".$fastaDirectoryName."/".$fastaDirectoryName."_SingleFastaSeq";
```

APPENDIX E. PREDICTCHUNKRNASTRUCTURESEQ_2.PL PERL SCRIPT

```
#!/usr/bin/perl
use strict;

###############################################################################
# Author: Gerardo Cardenas <gacardenas@utep.edu>
# Pre-Execution: condor_submit RNAChunkPrediction.submit
# Mid-Execution: None
# Post-Execution: Assembly_3.pl
# Description: This script uses the submit file to submit it to the Condor
#                    Pool, and store the prediction of chunks in the output directory.
#                    specified in the code below.
###############################################################################
#----------------
# PROGRAM STARTS
#----------------

# Generate file containing the directory names for each set of sequences.

  if ( !-d "Datasets")
  {
```

```
    print "\nDatasets directory does not exists!\n\n";
    exit;
  }

  system("ls -l Datasets | egrep '^d' | awk '{ print \$9 }' > directories.txt");

# Load the directory's file into memory
  my($directoryName) = "directories.txt";

  open (FILE,$directoryName) or die $!;
    my @fastaDirectoriesList = <FILE>;
  close(FILE);

# Process each RNA Chunk using pknotsRG
  #for(my $i=0;$i<=0;$i++)
  for(my $i=0;$i<=scalar(@fastaDirectoriesList)-1;$i++)
  {
    my ($fastaDirectoryName) = $fastaDirectoriesList[$i];

    chomp($fastaDirectoryName);

    # Check if Single Fasta file directory Output exists
    my ($SingleSeqFastaDir) =
"Datasets/".$fastaDirectoryName."/".$fastaDirectoryName."_SingleFastaSeq";
```

Vita

Gerardo Cardenas received his bachelor degree in Computer Science from the University of Texas at El Paso (UTEP) in El Paso, Texas. After he graduated, he worked in the Information Technology area in different enterprises including media, manufacture and healthcare. Nowadays, he is currently employed in a government I.T. area at the City of El Paso. In 2009, he completed his M. Sc. Degree in Bioinformatics from the University of Texas at El Paso (UTEP), El Paso, Texas. During his Masters, his research focus was on developing a tool to standardize the semantic descriptions of data and transformations commonly found in the domain of phylogenetic analysis. In fall 2010, he joined the Computational Science PhD program at UTEP. He is conducting his thesis under the supervision of Dr. Ming-Ying Leung. The focus on his research is to improve computational efficiency of RNA secondary structure prediction using distributed computing. After defending his thesis, he wishes to continue working in the same program towards his doctoral degree.

Permanent address: 2520 Azure Sky Pl.
El Paso, TX, 79938

This thesis was typed by the author.