10-2016

Preliminary Investigation of Mobile System Features Potentially Relevant to HPC

David Pruitt
The University of Texas at El Paso, ddpruitt@miners.utep.edu

Eric Freudenthal
The University of Texas at El Paso, efreudenthal@utep.edu

Comments:

Technical Report: UTEP-CS-16-69

UTEP Technical Report UTEP-CS-16-69

Preliminary Investigation of Mobile System Features Potentially Relevant to HPC

David Pruitt
University of Texas at El Paso
500 W. University Avenue
El Paso, Texas, USA
(915) 244-9759
ddpruitt@miners.utep.edu

Eric Freudenthal
University of Texas at El Paso
500 W. University Avenue
El Paso, Texas, USA
(915) 317-6246
efreudenthal@utep.edu

Energy consumption's increasing importance in scientific computing has driven an interest in developing energy efficient high performance systems. Energy constraints of mobile computing has motivated the design and evolution of low-power computing systems capable of supporting a variety of compute-intensive user interfaces and applications. Others have observed the evolution of mobile devices to also provide high performance [14]. Their work has primarily examined the performance and efficiency of compute-intensive scientific programs executed either on mobile systems or hybrids of mobile CPUs grafted into non-mobile (sometimes HPC) systems [6, 12, 14].

This report describes an investigation of performance and energy consumption of a single scientific code on five high performance and mobile systems with the objective of identifying the performance and energy efficiency implications of a variety of architectural features. The results of this pilot study suggest that ISA is less significant than other specific aspects of system architecture in achieving high performance at high efficiency. The strategy employed in this study may be extended to other scientific applications with a variety of memory access, computation, and communication properties.

1. INTRODUCTION

This report describes an analysis of relationship of performance and energy consumption by the CoMD proxy app executed on a variety of mobile and small HPC systems that implement multiple ISAs. This report begins with a description of the application and architectures examined in this study. They are followed by a description of the experimental methodology, results, and conclusions.

1.1 CoMD's EAMForce Kernel

CoMD is a Molecular Dynamics (MD) proxy application [3]. MD codes are an important class of application that has a high cache hit rate and low floating point throughput on superscalar processors [10]. CoMD implements two force computation methods. Our experiments utilize the EAMForce force computation method, which consumes over 95% of CoMD execution time in the configurations we selected.

CoMD simulates the interactions of atoms within a rectangular 3-D region. This region is decomposed into a 3-D spatial mesh of rectangular prisms assigned to each thread. These prisms are further divided into cubic "link cells." An execution consists of repeatedly executing a two-phase bulk synchronous "EAMForce" computation that updates the, energy, location, and velocity of each atom. Data is transferred among neighbors via halo cells at end of each phase.

EAMForce consists of several nested loops that iterate over nearby atom pairs within each cell and its eight neighbors. These loops utilize table lookups and interpolation to determine embedding energies and forces.

Since atoms can move, both the set of atoms within any particular cell, and each atom's set of neighbors may change between update cycles. The storage for atoms within each link cell is statically allocated and membership sets are maintained as lists of pointers. Different components of atom state are stored in distinct arrays. As a result, accesses to each atom's set of neighbors have poor spatial and temporal locality. Fortunately, accesses to nearby atoms have a high L1 cache hit rate.

CoMD provides few opportunities for vectorization other than computations that compute 3-dimensional force, position, and velocity vectors. In these cases, exposed ILP is limited to three double precision operations.

Multiple inner loops of EAMForce contain multiple control hazards. These hazards are dependent on distance cut-off comparison that are unlikely to be predictable and therefore will inhibit effective prefetching.

As described in Section 5 Results, executions of EAMForce on all of the architectures examined underutilize floating point throughput capacity.

Our intention was to initially examine an application likely to perform well on mobile systems. We incorrectly assumed that mobile systems would provide far lower floating point and memory system throughput than HPC. We selected CoMD because CoMD issues floating point and memory at low rates, even when executed on an HPC Intel.

2. Systems Examined

This study examines the interaction of architectural features with energy efficiency and performance. Practical considerations limited our examination to a set of five systems characterized in Table 1 Architecture Characteristics and described below. This section begins with a description of features associated with the systems' ISAs that are independent of architectural implementation. That description is followed with a summary of architectural implementation characteristics.

2.1 Instruction Set Architecture

The systems we examine implement the Intel x64 or ARM instruction sets. Power-constrained embedded and mobile systems are primary market for ARM based systems, and systems implementing this ISA are frugal consumers of energy. Intel's x64 ISA is a descendant of the instruction set implemented by Intel's 8086 families of CPU dating from the 1980's and is implemented within a variety CPUs developed for HPC, workstation, and mobile systems. As described below, the variation in performance or energy efficiency observed in our study may not be due to differences in ISA.

Our investigation examines performance of CoMD on five systems representing both HPC and mobile implementations of both instructions set families,

2.1.1 Instruction set

The systems we examine employ ARM and Intel ISAs. ARMv7 and ARMv8 are RISC instruction sets, and the two Intels implement a CISC instruction set. In the past, systems implementing RISC ISAs would directly implement each instruction. In contrast, systems implementing CISC ISAs would generally translate each instruction into a sequence of "micro ops." Modern implementations of RISC architectures also translate ISA instructions into micro-operations. Superscalar implementations of both families include logic to proactively and concurrently schedule micro-ops onto multiple functional units. As observed by [1] commonalities in the set of micro-ops result in implementations that are closer in design than their ISAs would suggest.

Table 1 Architecture Characteristics

CPU		Cortex A15	Cortex A57	I7-3840qm	Xeon E5-1650	Thunder X
ISA	Platform Type	Mobile			HPC	
	Instruction set	Arm		Intel		Arm
	Cores (Threads)	4 (4)		4 (8)	6 (12)	96 (96)
	Max Core Freq (GHz)	2.32	1.734	2.8	3.7	2.0
	Word Size	32 bit	64 bit			
	Float 64 Ops	Y				
	Vector Size	128		256		*
FP	DP Mul Latency	5 Cyc		6 Cyc		*
	GFlop/s Peak	20.255	30.79	115.8	245.745	143.71
	GFlop/s 1 core @ 1GHz	0.89	1.21	5.61	3.56	0.40
Memory	L1 Cache Latency	3-4 Clock Cycles				
	RAM Channel	12.5-17 GiB/s throughput				
		Low voltage			Standard Voltage	
	# RAM Channels	1	2		4	
	RAM Access Latency	Random: 30 - 45 ns				

2.1.2 Number of Cores

All the examined systems have multiple independent execution cores. All of the mobile systems contain 4 cores. The HPC Intel system contains 6 cores and the HPC Arm system contains 2 ThunderX 48 core CPUs, providing a total of 96 cores. Both Intel systems are capable of "hyperthreading" (a.k.a. simultaneous multithreading), of two threads upon each core.

2.1.3 Word size

Most of the devices we examine have a 64-bit word size. The exception is the mobile ARM 32 system, which is the only system with a 32-bit word size. The word size refers to the size of the general purpose registers that are used for integer and memory operations and are typically independent of those used for floating point operations.

2.1.4 Floating Point Operations

All three of the ISAs (ARM 32, ARM 64, and Intel x64) include 32 and 64 bit IEEE 754 floating point operations. Since only one of the systems has a 32-bit word size, clearly word size and floating point capability are independent of one another.

All three of the ISAs also support floating point operations on short vectors of floating point operands. Both Intel systems support 32-byte vectors (eight single precision or four double precision values), the others support 16-byte vectors. Although detailed specifications for the HPC ARM are not readily available it is listed as conforming to the Arms AArch64 standard for 64 bit devices. ARM's AArch64 standard requires floating point and vector support; therefore, we assume that the HPC ARM system provides short vector support.

2.2 Implementation Characteristics

The systems examined have several implementation attributes relevant to our investigation of the energy and execution characteristics of CoMD.

2.2.1 Platform types

Our target platforms consist of two main types of platforms; those targeted for mobile contexts such as smartphones and laptops, and those target for HPC environments. Mobile platforms are generally powered by batteries with limited energy capacity. To help cope with limited energy availability mobile devices` components implement aggressive energy saving features. For example, these systems utilize low voltage DRAM signaling and network devices that support low power standby states. HPC systems are typically designed for high throughput and frequently implement standard voltage memory signaling and high throughput I/O interfaces that may not incorporate aggressive power saving features such as low-power standby modes that can be enabled during idle periods.

2.2.2 Floating Point

Floating point throughput varies considerably across the systems examined. Table 1 Architecture Characteristics indicates both per-core floating point throughputs at maximum system core clock rates and also for core clocks rate normalized to 1 GHz. The HPC ThunderX has the lowest floating point throughput at 0.4 GFlop/s, with the Mobile Intel having the highest at 5.6 GFlop/s.

LMBench indicates that the latency of floating point operation is similar (5-6 cycles) across all system except for the ThunderX, for which it only reported nonsense values (e.g. 0.12 cycles).

2.2.3 Memory

DRAM: The HPC ARM incorporates DDR4 DRAM, which can transfer two 64-byte blocks to two cache lines (of the same size) in a single memory-channel clock cycle. All the other systems utilize DDR3 memory, which can transfer one 64-byte blocks in a single clock cycle.

The mobile systems implement either one or two memory signaling channels. The HPC systems implement four channels.

Although our devices encompass two generations of DDR memory, the throughput of a single channel is within a narrow range of 12 – 15 GiB/s. Accesses to a random DRAM pages for all five systems examined in this study (and most others since the mid 1970's) falls into the narrow range of 30 to 45 ns.

In order to minimize energy use, mobile systems generally utilize DRAM with low voltage signaling. CPUs intended for mobile generally can also signal at standard voltage, and many low cost ARM boards (such as the QuadMo 747-X incorporated within the Tibidabo system implemented by the Barcelona Supercomputing Center [12]) utilize less-energy-frugal standard voltage DRAMS. The mobile systems we examined all use DRAMs with low voltage signaling. The HPC systems we examine use DRAM with standard voltage signaling.

2.2.4 Caches

All of the systems implement 32kB L1 caches with 64 byte lines Table 1 Architecture Characteristics indicates an access to L1 has a latency of 4 cycles on all systems except the HPC-ARM, for which an L1 access requires 3 cycles.

Both Intel devices implement a hierarchical 3-level cache, where the L2 caches are clocked with the cores and a single L3 cache is clocked with uncore [4].

The ARMs all implement two-level caches where the L2 shares a common clock with the computational cores.

2.3 Characteristics correlating with Platform Type

The mobile and HPC systems examined in this study have some common characteristics.

One frequently overlooked characteristic is the maximum CPU core frequency. Although both types of systems must limit system temperature to reasonable levels, mobile systems are used in environments with limited cooling and where high temperatures are undesirable. The maximum CPU frequency allowed is dependent on the temperature of the CPU, and thus varies based on the amount of energy dissipated by each core.

The frequencies listed are the maximum available under when all cores are heavily loaded. The mobile devices maximum frequency varies between 1.7 GHz and 2.8 GHz. The HPC Intel has a maximum operating frequency of 3.7 GHz, significantly higher than the mobile devices. The ThunderX has a lower maximum frequency of 2 GHz, which may be due to the high core count.

The HPC platforms implement four memory signaling channels and can provide commensurately higher peak memory throughput. As a result of a larger number of cores and higher core frequency, the total floating point throughput of the HPC systems is significantly higher than that of the Mobile systems.

2.4 Methodology

Performance characteristics of all systems was determined using standard benchmarks. LMBench was employed to determine memory and floating point latency [7]. Where possible the results from LMBench were confirmed using published specifications. Core and system memory and floating point throughput were measured using the CS Roofline Toolkit was used [13]. Measurements were taken for all cores and a single core across all systems to allow for comparison across the various architecture implementations.

To measure power consumption for the mobile systems, we constructed a filtered current and voltage measurement and logging system that enables continuous measurement and logging while CoMD is being executed. Output from this device was used both to determine power consumption and to detectover various execution phases.

Data collected using this device was used to measure power and detect program phase changes. We instrumented voltage and current between the external AC-DC power supply and board level power supply because, unlike the highly efficient on-board DC-DC converters, the power conversion efficiency varies based on power supply design and system load.

The power measurement on the HPC ARM system was obtained from instrumentation within its DC power supply. and logs power over intervals of one minute. To ensure consistent results for this system we run our experiments in continuous loops and collect at least 100 samples.

We attempted to use performance counters to measure and characterize micro-architectural characteristics such as memory stalls and branch hazards. Measuring various attributes through the performance counters resulted in inconsistent and contradictory results. Examples of this include the ratio of floating point instructions to memory request varying wildly across the same ISAs for repeated executions of the same code. In contrast, counts derived from simulations conducted using ValGrind [9] yielded consistent and reasonable values. Our experience is consistent with the observations of others who indicates performance counters can become inaccurate with more complex sequences of code such as those found within CoMD [15, 16].

Execution parameters were identical across all systems to allow for direct comparison of one system to another under different system configurations. Parameters were selected to allow for long execution times to prevent overhead and system noise from skewing results. After the data was collected, the relationships among attributes was graphed. This allowed us to easily compare and correlate the relationships between efficiency and performance.

3. Results

Throughput of CoMD executions at various levels of concurrency are illustrated in Figure 1. The vertical axis indicates execution throughput (runs/minute) and the horizontal axis indicates core clock frequency (MHz). Interfaces to specify CPU core clock frequency are available for all of the mobile platforms. Our experiments included sweeps of these frequency options.

HPC v. Mobile Intel: At its highest clock rate, an eight core execution on the mobile Intel is 30% faster than the HPC Intel. There is little difference between the throughput of four executions on mobile ARM and HPC Intel at the mobile ARM's highest clock rate.

HPC v. Mobile ARM: The HPC ARM provides lower per-core throughput than either of the mobile ARMs.

High core counts provide high throughput: The HPC systems with high core count provide the highest throughput (5.5 runs/min for 96 cores on the HPC ARM, 2.4 runs/min for 12 cores on the HPC Intel).

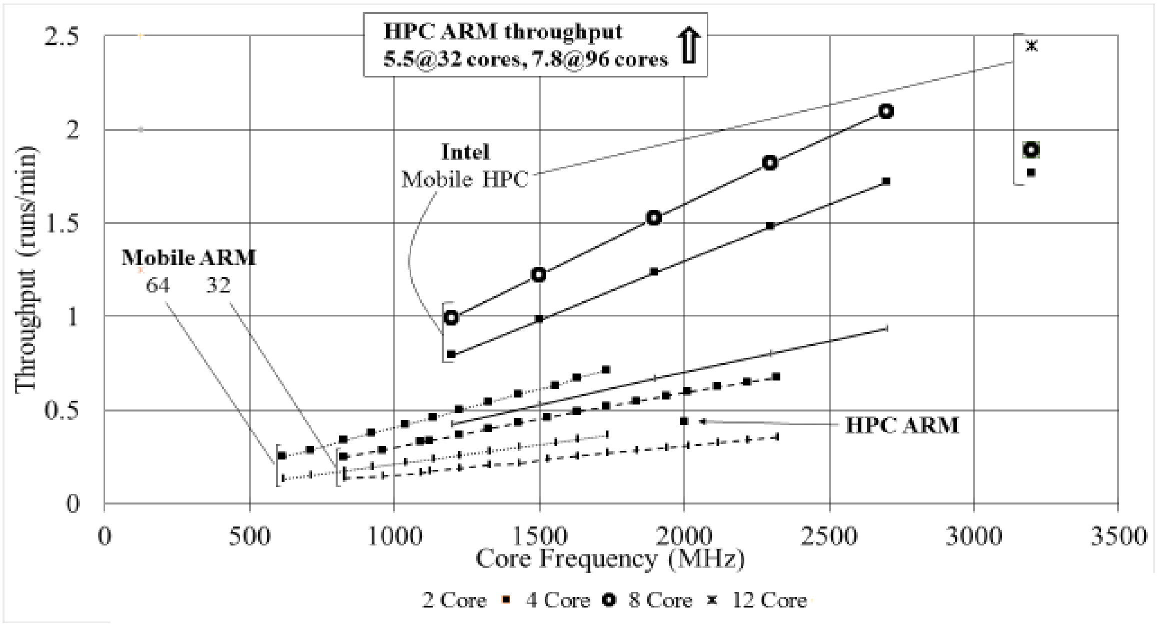


Figure 1 CoMD Floating Point Throughput

Throughput scales proportionally with clock frequency. Functional units (such as FPUs) and L1 caches are closely coupled to cores and are clocked together. As a result, all of their timing characteristics scale proportionally with clock frequency. The proportional scaling of throughput with clock frequency indicates that there are an insignificant number of core execution stalls due to interactions with "uncore" subsystems whose timing doesn't scale with core clocking such as DRAM. This proportional speedup is consistent with our measurements of L1 cache hit rates above 99.5% on all systems.

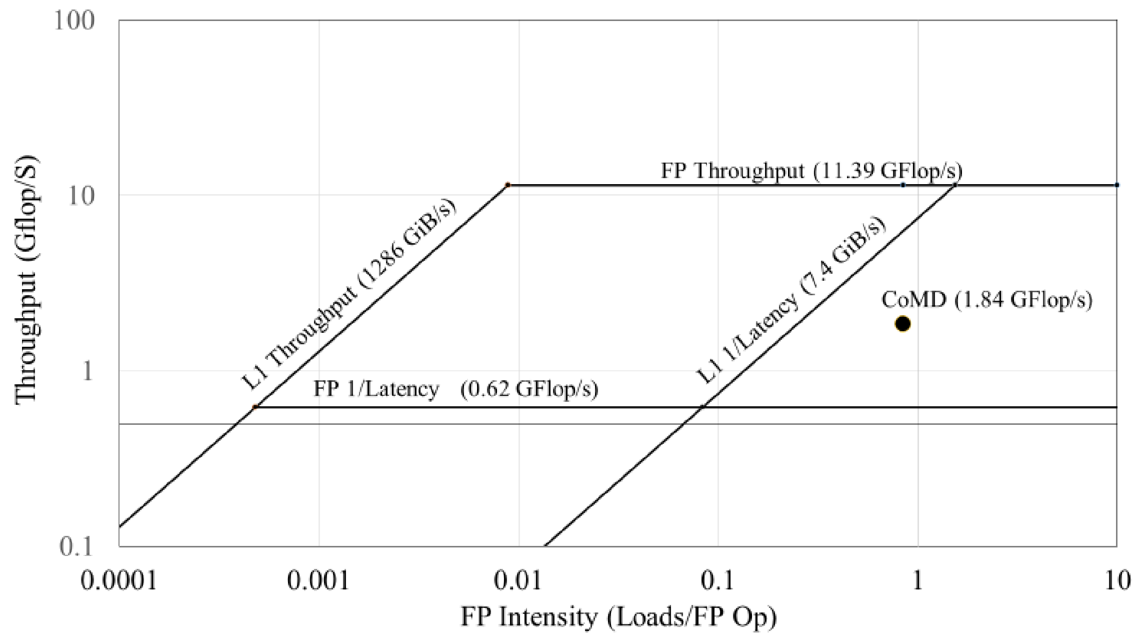


Figure 2 Roofline for CoMD on HPC Xeon

Comparison of Throughput on ARM and Intel: While the ARM systems examined all execute CoMD more slowly on a single core, this performance difference may not be inherent to the differences of ISA. An observation that all three of the ARM CPUs have lower peak floating point throughput than either of the Intel CPUs motivates an examination of whether the differences in CoMD throughput may be substantially an artifact of system floating point throughput. If this conjecture is correct, then the architectural peak floating point throughput will linearly correlate with CoMD throughput.

Figure 2 and Figure 3 indicate that the average floating point and memory intensity of these applications are far below architectural FP throughput capabilities and above inverse-latency serialization.

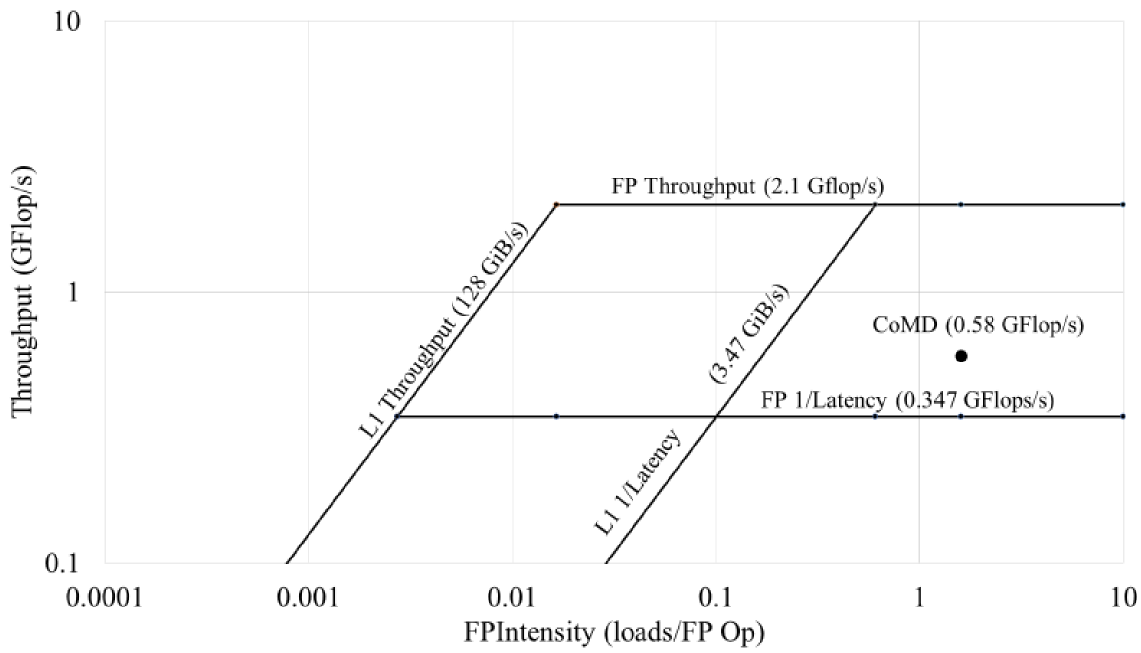
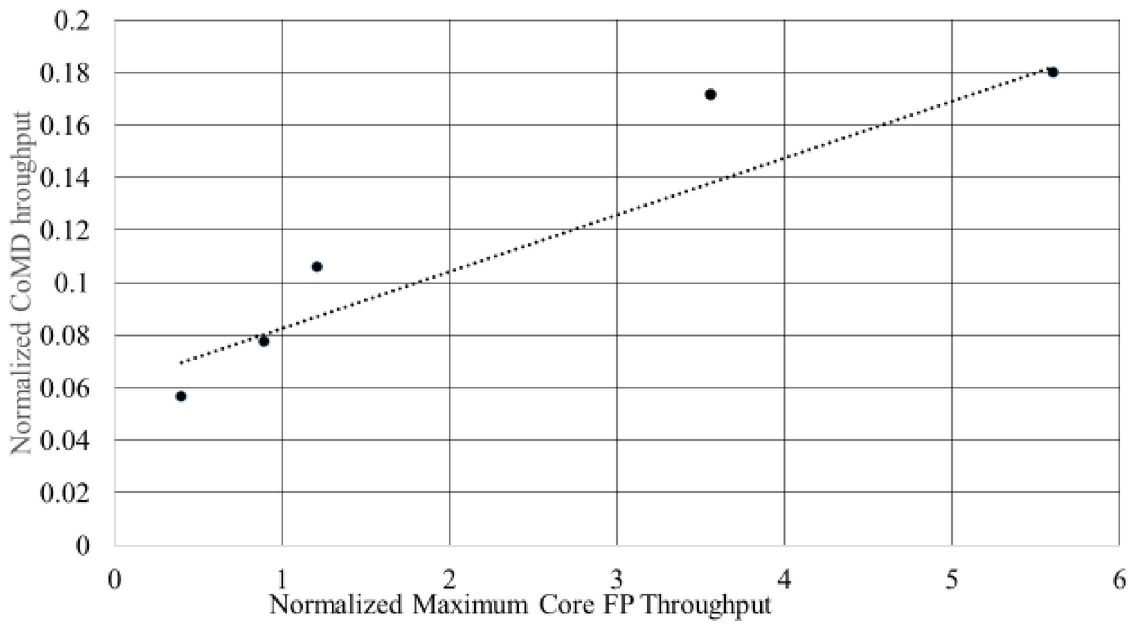


Figure 3 Roofline for CoMD on Mobile Arm 32

While average floating point and memory intensity is far below architectural limits, CoMD may at times reach them. Fioe 4 illustrates the linear correlation of clock-frequency normalized architectural peak floating point throughput and CoMD throughput. The Pearson product-moment correlation coefficient for this set of samples is 0.968, which leads us to conjecture that the variation in CoMD throughput for this set of observations may be principally due to the systems' variation of peak floating point throughput along the critical path of execution.



Fioe 4 Correlation of Floating Point Throughput and CoMD Throughput

Parallel Speedup: Table 2 CoMD Parallel Efficiencyindicates the fraction of perfect speedup measured for various levels of thread-level concurrency. Italicized entries are for executions where hyperthreading is employed to enable more than one thread to execute on at least one core. The mobile ARMs have high speedups at all levels of concurrency, which indicates minimal congestion for resources shared among cores.

Table 2 CoMD Parallel Efficiency

Cores	Cortex A15	Cortex A57	i7-3840qm	Xeon E5-1650	Thunder X
1	1	100%	100%	100%	1
2	98%	99%	96%	97%	98%
4	94%	97%	89%	80%	96%
8			*54%*	*43%*	94%
12				*37%*	
16					88%
32					82%
64					76%
80					75%
96					72%

Speedup of Intel systems is near perfect when hyperthreading is not utilized. Due to the perfect frequency scaling described earlier, the lower speedup when hyperthreading is utilized is due to contention among devices clocked with the cores. In this case, the likely culprits are functional units shared among co-resident hyperthreads, such as those that implement floating point operations.

While the HPC ARM is not hyperthreaded, each core has a small 32kB L1 cache. The low speedups for 32 and 96 core runs is likely due to contention for the single 16MB L2 cache that processes L1 misses from all 96 cores.

Energy Efficiency: Power consumption was instrumented on all the systems examined; These measurements were used to compute energy efficiency in units of runs/kJ (higher is better). This efficiency metric is the reciprocal of the frequently cited power-delay product (PD) [5].

Energy efficiency is plotted on the vertical axis of

Figure 5 CoMD Energy Efficiencyand CPU clock frequency is plotted on the horizontal axis.

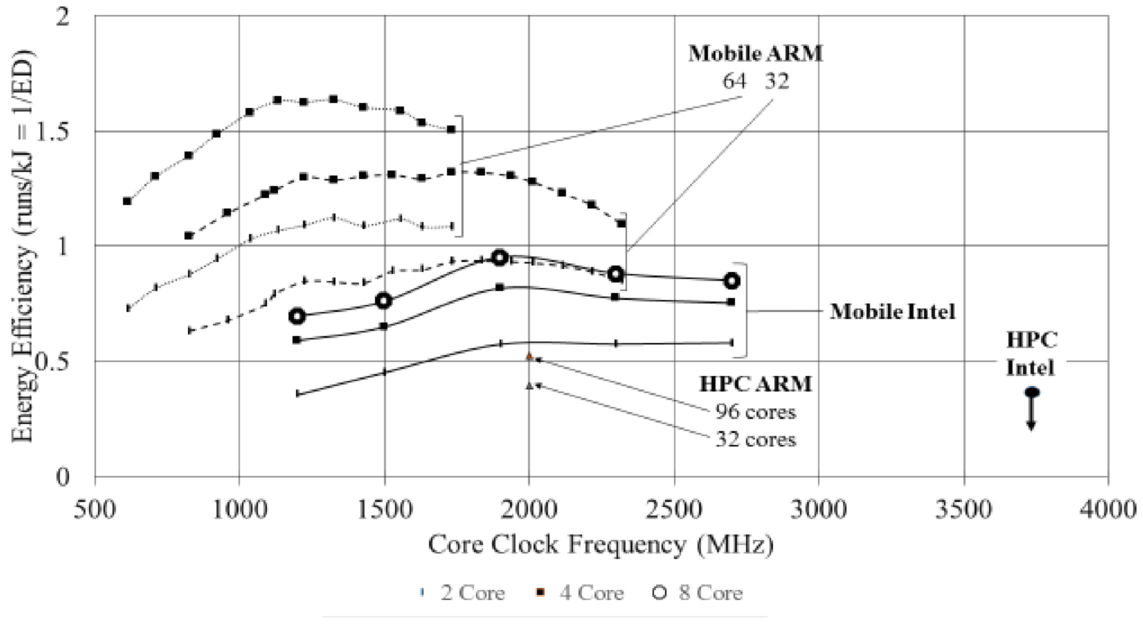


Figure 5 CoMD Energy Efficiency

The Mobile ARM 64 system is the most energy-efficient of those examined in this study. It is only incrementally more efficient that the Mobile ARM 32. The ARM 32 has larger feature sizes than the ARM 64. It is unclear whether the difference in energy consumption may be due to evolutionary improvements to devices manufacturing processes or changes to the architecture that enable higher throughput at lower clock rates.

The Mobile Intel has substantially lower per-core energy efficiency than either of the mobile ARM systems.

Both of the HPC systems have lower efficiency any of the mobile systems examined.

The HPC system load power consumption is approximately one and a half times that of idle power consumption, while mobile systems typical have a load power consumption that is four or five times higher that idle power consumption.

We suspect that this is due to their incorporation of subsystems and interfaces that are not optimized for energy efficiency such as standard-voltage DDRn memory channels and communication interfaces that do not enter low-power idle modes.

The only power consumption data available for the Intel HPC was for the system when idle (~90W). This idle power consumption is surely lower than would be consumed when executing CoMD, and sufficient to determine that this system is the least efficient the systems examined in this study. With the additional power consumed when under load it is likely that Intel HPC system would be far less efficient than any of the other systems examined.

DVFS: The nuanced relationship between frequency, voltage, and energy efficiency is illustrated in Figures Figure 6 and Figure 7.

As is common for DVFS scaling [11] CPU core efficiency drops with increased frequency. This drop in efficiency is a side effect of the increase of core voltage required for higher clock rates. This increase in core voltage also increases transistor leakage current and decreases device efficiency [8].

These figures detail energy consumption for 1 core and 8 core executions of CoMD on the mobile Intel system. The upper plots indicate efficiency (unit labels on the left axis) and the lower plot indicates core voltage (unit labels on the right axis). The plots labelled “System Energy” and “Idle Energy” are derived from execution timing and power supply measurements during execution and when the system is idle. “Active” energy corresponds to the excess energy consumed by an active computation and is computed as the difference between “System” and “Idle” energy.

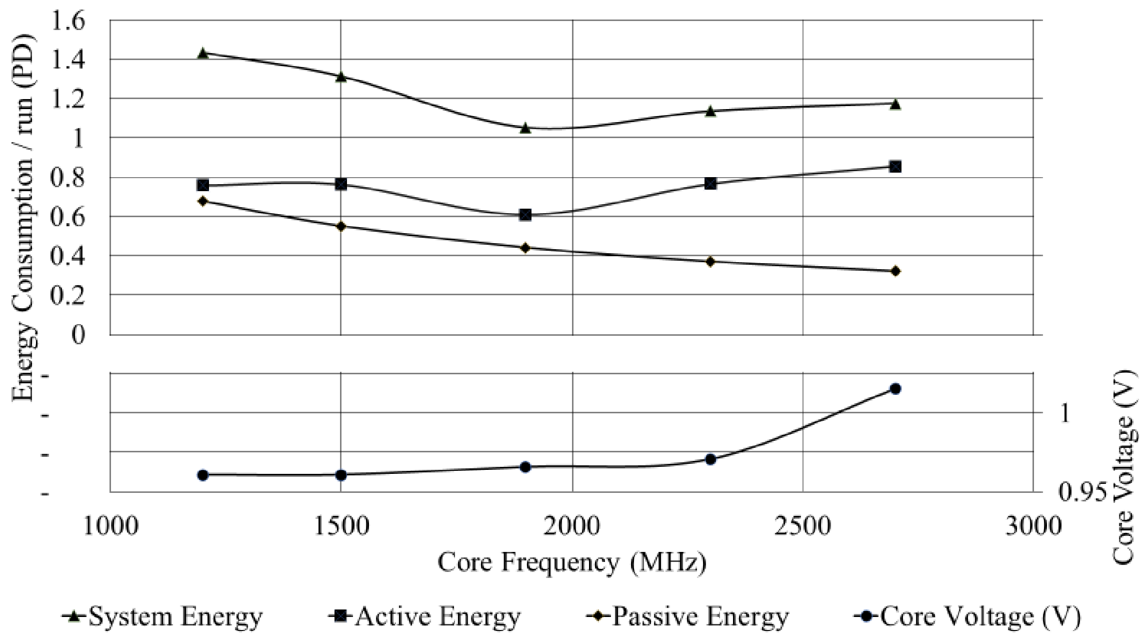


Figure 6 Mobile Intel 8 Core Energy Detail

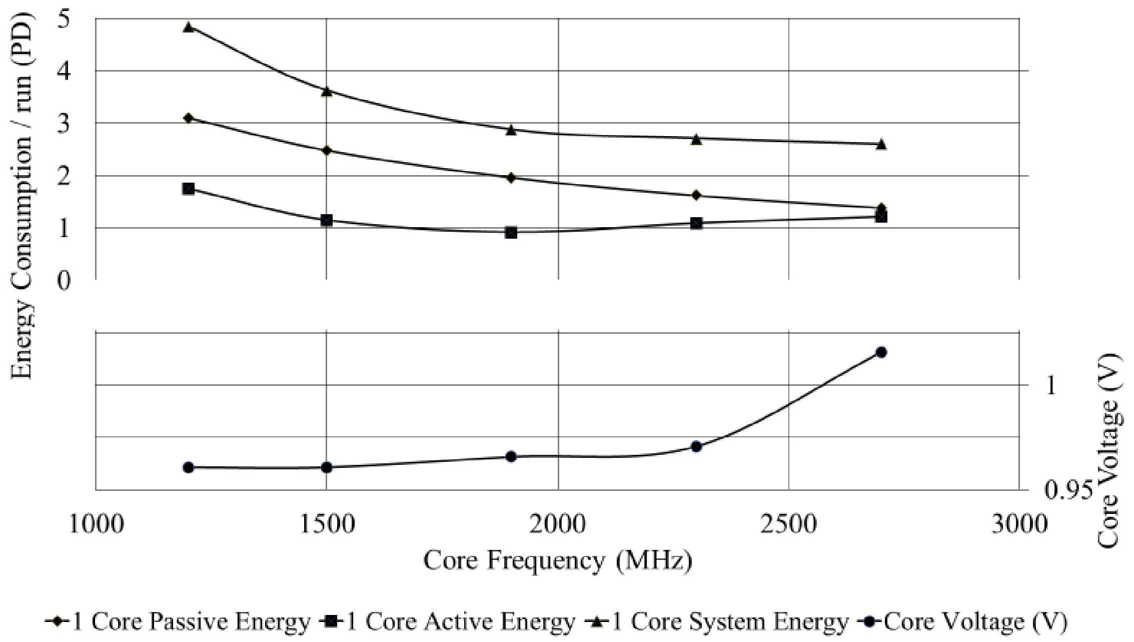


Figure 7 Mobile Intel 1 Core Energy Detail

For 1-core executions, active energy is lowest at 1900 MHz. However, idle energy dominates active energy at all speeds and results in highest system efficiency (lowest energy consumption) at the highest clock rate.

In contrast, for eight core executions, active energy consumption dominates idle energy consumption. In this case, both the active and total system energy is minimized at 1900 MHz.

When both graphs are compared significant differences the relationships between active and idle energy are observed. For 8-core execution idle and active energy consumption are approximately equal at the lowest frequency and diverge as core frequency increases. For all frequencies active energy is greater than idle energy. For single core execution idle energy consumption dominates at the lowest frequency, with idle and active energy consumption converging at higher frequencies.

For single core execution energy efficiency increases with frequency whereas 8 core execution loses efficiency at higher frequencies.

Energy Delay Product: Energy delay product (EDP) is a commonly used metric to compare two HPC systems. Energy delay product is a useful metric as it allows one to easily compare systems with similar energy consumption while favoring those that have higher throughput [5].

Energy delay product does not take into account practical considerations such as the cost of additional cores, required throughput, or limits on available energy.

Since energy delay product has asymptotic behavior it is difficult to compare systems where energy consumption varies greatly. It will favor systems with high concurrency, such as those with high core counts, when executing embarrassingly parallel applications even in cases where the more efficient system’s throughput is sufficient for the problem at hand.

As seen in Figure 8 the HPC ARM systems has an EDP of 14 compared to the next lowest EDP of 33. This is largely due to the HPC ARM far greater number cores than any of the other systems examined. This trend continues with the 8 and 4 core systems having progressively higher EDP values.

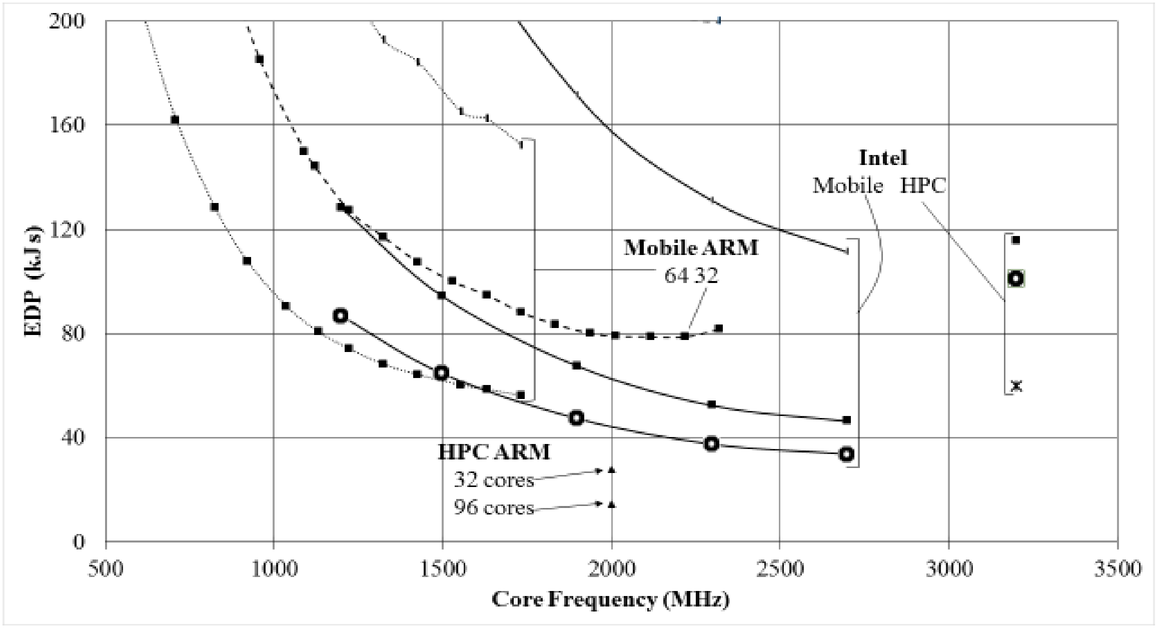


Figure 8 CoMD Energy Delay Product

This is the opposite behavior exhibited when observing energy efficiency, the 4 core mobile system is most efficient followed by the 8 core system. The HPC ARM system with the lowest EDP consumes 1.9 kJ to complete the computation, while the next most efficient system consumes 1.4 kJ.

EDP can be helpful for selecting strategies for efficiently improving execution throughput. However, if methods for achieving sufficient execution throughput are available, it is generally preferable to optimize using more conventional metrics such as energy consumption or manufacturing cost.

4. Conclusions

Relevance of ISA and implementation features: As suggested by [1, 2], observed differences in performance and energy efficiency are primarily due to implementation characteristics of floating point subsystems, memory hierarchies, clock frequencies, and pipeline depth rather than ISA. In the case of CoMD, differences in performance appear to be substantially attributable to differences in floating point throughput.

Concern about the HPC ARM system: The HPC ARM's shallow cache hierarchy was sufficiently congested by cache misses from CoMD to substantially reduce throughput at high degrees of parallelism. CoMD has a high cache hit rate. Applications with lower L1 cache hit rates are likely to generate even more congestion at the L2 cache and therefore be limited to even lower parallel efficiency.

Characteristics of applications likely to have high performance and on current mobile platforms: We are surprised to learn that DRAM communication channels on all systems that we examined have similar performance characteristics despite the mobile systems' incorporation of low voltage signaling. The mobile systems incorporated either one or two of these channels, in contrast to the HPC systems' incorporation of four. Since CoMD's offered memory and floating point load is far below the throughput of all of the mobile platforms, it is likely that apps that generate substantially higher memory and floating point throughput will execute with only incremental slowdown on the mobile platforms as compared to their throughput on an HPC Intel.

Apps with that generate floating point traffic at a lower rate than CoMD may execute with similar throughput on the mobile platforms to Intel HPC.

Characteristics of mobile systems of relevance to HPC: Idle power is the principal difference between HPC and mobile systems examined in this study. We observe that the mobile systems incorporate network, I/O, and memory interfaces with energy-saving features such as low voltage signaling and various forms of low-power standby modes. These energy savings features are not generally present on HPC systems. This design decision adversely affects energy efficiency in two ways.

An I/O interface that draws a significant fraction of its active power when idle will increase the overall system idle power requirements. As described in the Results section, race-to-idle strategy may yield highest efficiency with the side effect of reducing overall system energy efficiency.

Furthermore, if the system is operated under a power cap, the allocation of power to underutilized devices will reduce the amount of energy available to support useful computation.

Acknowledgements

This work was partially by the Air Force Office of Scientific Research under Award No. FA9550-12-1-0476. We acknowledge use of the Bolt and ThunderX hosts maintained by the Gordon Group at the University of Iowa. We gratefully appreciate Phillip Tennen's assistance in the typesetting of graphs included in this paper.

References

[1] Blem, E. et al. 2015. ISA Wars. *ACM Transactions on Computer Systems*. 33, 1 (2015), 1–34.

[2] Blem, E. et al. 2013. Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures. *Proceedings - International Symposium on High-Performance Computer Architecture*. Hpca (2013), 1–12.

[3] ExMatEx | Proxy Applications | CoMD: *http://www.exmatex.org/comd.html*. Accessed: 2016-07-23.

[4] Hammarlund, P. 2013. 4th generation Intel^{TM} Core processor, codenamed Haswell. *2013 IEEE Hot Chips 25 Symposium (HCS)* (Aug. 2013), 1–35.

[5] Hsu, C.H. et al. 2005. Towards efficient supercomputing: A quest for the right metric. *Proceedings - 19th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2005*. 2005, April (2005).

[6] Jundt, A. et al. 2015. Compute bottlenecks on the new 64-bit ARM. *Proceedings of the 3rd International Workshop on Energy Efficient Supercomputing - E2SC '15* (New York, New York, USA, 2015), 1–7.

[7] Mcvoy, L. and Staelin, C. lmbench: Portable tools for performance analysis.

[8] Nam Sung Kim et al. 2003. Leakage current: Moore's law meets static power. *Computer*. 36, 12 (Dec. 2003), 68–75.

[9] Nikolskiy, V. and Stegailov, V. 2016. Floating-point performance of ARM cores and their efficiency in classical molecular dynamics. *Journal of Physics: Conference Series*. 681, (Feb. 2016), 012049.

[10] Pallipadi, A. and Starikovskiy, A. 2006. The ondemand governor: past, present and future. *Proceedings of Linux Symposium Volume Two*. (2006), 215–230.

[11] Rajovic, N. et al. 2014. Tibidabo1: Making the case for an ARM-based {HPC} system. *Future Generation Computer Systems*. 36, (2014), 322–334.

[12] Spear, W. and Norris, B. A Roofline Visualization Framework.

[13] Tiwari, A. et al. 2015. Performance and energy efficiency analysis of 64-bit ARM using GAMESS. *Proceedings of the 2nd International Workshop on Hardware-Software Co-Design for High Performance Computing - Co-HPC '15* (New York, New York, USA, 2015), 1–10.

[14] Weaver, V.M. et al. 2013. Non-determinism and overcount on modern hardware performance counter implementations. *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (Apr. 2013), 215–224.

[15] Zaparanuks, D. et al. 2009. Accuracy of performance counter measurements. *ISPASS 2009 - International Symposium on Performance Analysis of Systems and Software*. (2009), 23–32.