

Power-Performance Efficiency of Asymmetric Multiprocessors for Multi-threaded Scientific Applications

Ryan E. Grant

Ahmad Afsahi

*Department of Electrical and Computer Engineering
Queen's University*

Kingston, ON, Canada K7L 3N6

ryan.grant@ece.queensu.ca ahmad.afsahi@queensu.ca

Abstract

Recently, under a fixed power budget, asymmetric multiprocessors (AMP) have been proposed to improve the performance of multi-threaded applications compared to symmetric multiprocessors. An AMP is a multiprocessor system in which its processors are not operating at the same frequency.

Power consumption has become an important design constraint in servers and high-performance server clusters. This paper explores the power-performance efficiency of Hyper-Threaded (HT) AMP servers, and proposes a new scheduling algorithm that can be used to reduce the overall power consumption of a server while maintaining a high level of performance. Prototyping AMPs on a commercial 4-way SMP server, we show that on average 15.6% energy savings and 6.1% slowdown for the HT-disabled case, and 7.1% energy savings and 4.8% slowdown for the HT-enabled case can be achieved across NAS and SPEC OpenMP applications.

1. Introduction

Power has become a critical design constraint for contemporary microprocessors [22]. Researchers have studied various power-related optimization techniques including voltage and frequency scaling of the processor core [11, 15], dynamically tuning processor resources with adaptive processing [1], and multi-core architectures [16, 17].

While there has been a large body of work in saving energy in the areas of mobile computing and embedded systems to extend battery life including [6, 30], energy conservation has recently become important in servers and clusters [7, 8, 10, 2] with their non-interactive high-performance computing workload in *message passing interface* (MPI) [21] and OpenMP [23]. In such systems, the focus is to improve the reliability and to reduce the cost of powering and cooling.

Researchers have proposed multi-ISA multi-core architectures [25] as well as single-ISA multi-core architectures [17]. It has been shown that single-ISA multi-core architectures can essentially reduce power consumption, increase throughput, and mitigate Amdahl's law [16]. Recently, researchers from Intel [2, 3] have illustrated how an *asymmetric multiprocessor* (AMP) can be built from a 4-way commercial *symmetric multiprocessor* (SMP). AMP, a form of single-ISA heterogeneous architecture, is a system made up of multiple processors that are not operating at the same speed. This is accomplished by clock throttling, where the duty cycle of the processor is reduced. This corresponds to a heat and energy savings proportional to the change in duty cycle. The Authors in [2] consider AMP configurations that can achieve an average of 38 percent wall clock speedup over the SMP on a wide range of multi-threaded applications under a given power budget.

While the work in [2] assumes a fixed power budget, this paper, for the first time, evaluates the power-performance efficiency of *simultaneous multithreading* (SMT) [29] AMPs for the sake of energy saving with a minimal impact on the performance of multi-threaded applications in OpenMP. SMT is a technique that allows multiple independent threads to execute different instructions each cycle. Intel *Hyper-Threading* (HT) technology is an implementation of SMT where two threads can be executed in parallel on a single physical processor. These threads have their own independent run-queues and each physical processor appears to the operating system as two logical processors. These logical processors share the resources of the physical CPU including cache, execution units, translation look aside buffers, branch prediction unit and load and store buffers. Previous work [9] has found that enabling HT on all processors in an SMP may not be beneficial in terms of performance. Contrary to the work in [2] that focuses on HT-disabled AMPs, we present performance and

energy saving results on both HT-enabled and HT-disabled AMPs.

A modification to the Linux scheduler is proposed as a method of potentially reducing the power consumption of a system, while producing less of a performance impact on the system than the original scheduling algorithm. Previous research has shown that system noise including operating system (OS) interference with the application has a dramatic effect on high-performance computing [24, 14, 28]. For this, using static clock throttling and processor affinity, we bind all OS activities to logical process zero (or physical process zero) that runs at a lower frequency than the rest of (logical) processors in the AMP. In order to sustain the performance for the parallel OpenMP threads, all other (logical) processors run at their maximum frequency. Overall, this new scheduler does a good job at reducing the energy consumption of the AMPs running NAS OpenMP [13] and SPEComp [27] applications while having a minimal impact on the performance of the system. Our performance results indicate 15.6% energy savings and 6.1% slowdown for the HT-disabled case, and 7.1% energy savings and 4.8% slowdown for the HT-enabled case across all applications studied in this paper.

The rest of this paper is organized as follows. In Section 2, we describe the related work and motivation behind this study. Section 3 defines the metrics used in this work. Experimental framework including the AMP setup is discussed in Section 4. We present and analyze the results in Section 5. Section 6 concludes the paper.

2. Related Work and Motivation

There have been quite a number of microarchitectural techniques on the subject of energy reduction of modern day processors. These include dynamically tuning processor resources with adaptive processing [1], comparison of SMT and chip multiprocessing (CMP) [18, 26], and heterogeneous multi-core architectures [16, 17]. Most of such research has been done with single-threaded applications and through simulations, or analytical methods. Our work in this paper concerns with multi-threaded workloads on real systems.

Dynamic voltage and frequency scaling (DVFS) is known as one of the most effective methods to reduce CPU power consumption, unfortunately at the expense of performance degradation. In fact, the semiconductor industry has recently introduced many energy saving technologies into their chips. The most successful of these have been *SpeedStep* technology from Intel as

well as *PowerNow!* and *Cool'n'Quiet* from AMD. Many works have been reported in utilizing such features to reduce power and energy consumption, from devising a compiler algorithm for optimizing single-threaded programs for energy usage on laptops [11], power and energy management techniques for servers and data centers [15, 4], to high-performance computing workloads in SMP and AMP servers [2, 3] and in high-performance clusters [7, 8, 10].

The authors in [2, 3] describe the process of creating an AMP node from a commercial Intel SMP server. They analyze the energy per instruction (EPI) gains that can be obtained from using CPUs operating at different frequencies [2]. In fact, they determined that by utilizing a setup that consists of one fast processor to run sequential code, assisted by three slower CPUs to run parallel code, one could reduce the overall EPI of a system while maintaining a higher speed than a normal SMP using the fixed power budget of one 2.0GHz Intel Xeon processor. They used the SPEComp benchmarks as well as several other applications in their study; however, their work did not address HT-enabled systems [2].

In [3], researchers investigated the impact of performance asymmetry of different AMPs on commercial applications as well as SPEComp applications. For commercial applications, they observed significant performance instability. They were able to eliminate this for some applications by devising a new kernel scheduler ensuring faster cores never go idle before slower ones. For SPEComp scientific applications with tight coupling among different threads, they found stability, but with poor scalability as the slowest core forces faster ones to idle. To eliminate performance asymmetry, they changed the static OpenMP loop scheduling used in the codes to dynamic scheduling. However, it resulted in degraded performance. Although, the work in [3] is only focused on performance asymmetry, they indicate AMP systems can be effective for power/performance efficiency. This work did not address HT-enabled systems, either.

This paper builds upon the work in [2, 3]. However, our motivation is completely different. We seek to reduce the overall energy consumption of a high-performance server while sustaining performance when running multi-threaded scientific applications. We address both HT-enabled and HT-disabled AMPs. To sustain the SMP performance we propose a new kernel scheduler for our AMP.

Nikolopoulos and his associates [19] use hardware performance counters to identify the best mix of threads to run across the processors and within each

processor of SMT-based SMPs. For this, they keep track of memory bandwidth utilization of the threads, bus transactions per thread, stall cycles, and cache miss rate per thread. They could realize an improvement of over 28.7% for a limited number of the NAS OpenMP [13] benchmarks. One has to bear in mind that although such schedulers can improve performance, they may also increase the overall system energy consumption.

Previous research has shown that system noise may have a dramatic effect on high-performance computing systems [24, 14, 28]. In [24], Petrini and his colleagues noticed that their application has a better performance when using three processors per node instead of the full four. Using a number of methodologies, they discovered that this is due to neither the MPI implementation nor the network, but the system noise including OS daemons, kernel threads, OS real-time interrupts, among other things. The authors in [14, 28] also verified the effects of system noise on the performance of applications.

While there have been some effective techniques proposed in [24, 14] to reduce the impact of system noise, such as removing (or moving to another processor) unnecessary OS daemons and kernel threads, lowering tick rate, and co-scheduling, leaving one processor for OS tasks is still a simple, viable option to effectively separate system noise from the computation [28]. Meanwhile, past work on real-time processing with Linux schedulers [5] has found that reserving a CPU specifically to respond to real-time priority threads significantly decreases the latency for real-time threads as well as the interrupt response time.

By offloading system noise onto a single (logical) processor, we can speculate that by avoiding swapping the threads in and out of the CPU we can increase the time available to the user threads. This reduced noise will correspond to an increased performance of the user threads such that the impact of reserving the CPU for system tasks is minimized. In the event of a system load that does not corresponds to a full load for a single (logical) processor, there exists an opportunity to reduce the frequency of the reserved CPU such that its load is as close to 100% as possible. This frequency scaling of the reserved CPU has a power savings effect. Therefore, the new scheduler should provide better performance over the original scheduler when the frequency of one of the processors is scaled down.

3. Metrics

We use the following metrics in studying the power-performance characteristics of systems.

Performance. High-performance computing has always been concerned with performance. Performance of an application running on a system is given by wall-clock execution time, D .

Power. Theory [22] tells us that the power consumed by a CMOS processor, in watt, is equal to the activity factor of the system (percentage of gates that switch for each cycle, on average 50%) multiplied by the capacitance of the CPU times the voltage squared times the frequency. This is shown in Equation 1.

$$P = \alpha CV^2 f \quad (1)$$

Note we have ignored the power expended due to short-circuit current, and the power loss from leakage current, as the dynamic power consumption, $\alpha CV^2 f$, dominates in CMOS circuits.

Frequency is directly proportional to the supply voltage. Therefore, power is proportional to the cube of a changing frequency. However, historical data [2] suggests that power on modern processors is proportional to the square of the duty cycle. Therefore, for this paper we will use the square relation, to better model the real power consumption of the system.

Energy. Power is the consumption at a discrete point in time. Energy is the cost during the execution time, D , and is shown as:

$$E = \int_0^D P dt = P_{avg} \times D \quad (2)$$

Power-performance efficiency. This metric allows choosing the operating point at which maximum energy saving can be achieved with acceptable performance degradation. We use the energy-delay product to quantify the power-performance efficiency, as shown in Equation (3):

$$E \cdot D = E \times D \quad (3)$$

4. Experimental Framework

The experiments were conducted on a Dell PowerEdge 6650 server. The PowerEdge 6650 has four 1.4GHz Intel Xeon MP processors with 12KB shared execution trace cache, 8KB L1 shared data cache with 4-way associativity, 256KB shared and unified L2 cache, 512KB shared and unified L3 cache (both L2 and L3 with 8-way associativity), and 2GB of DDR-SDRAM on a 400 MHz Front Side Bus. The operating system is based upon the RedHat Linux 9 distribution, but with the Vanilla kernel version 2.6.9.

The kernel supports Hyper-Threading. The task scheduler has undergone a complete rewrite in the 2.6.9 kernel. It is known as the O(1) scheduler since it can make a scheduling decision in constant time and independent of the number of processors or the number of tasks. Using the LMBench [20] we have measured the L1, L2, and main memory latencies of the processor as 1.42ns, 13.29ns, and 187.94ns, respectively. The main memory read and write bandwidths are 627 MB/s and 743 MB/s, respectively.

4.1 Application Benchmarks

4.1.1 NAS OpenMP

The NAS OpenMP parallel benchmarks (version 3.2) have been widely used to characterize high-performance computers [13]. The suite consists of five kernels, (CG, MG, FT, IS, EP), and three simulated CFD applications (BT, SP, LU). We experimented with Class B of CG, MG, BT, SP, and LU benchmarks. Class B is large enough to provide realistic results, ensuring their working set fits in memory.

4.1.2 SPEC OMP

The SPEC OMPM2001 (version 3.0) suite of applications [27] is from the SPEC High-Performance Group. The suite consists of a set of OpenMP-based scientific applications. These programs were originally part of the SPEC CPU2000 suite and were parallelized by inserting OpenMP directives. We worked with seven of the nine SPEC applications, specifically *apsi*, *art*, *quake*, *fma3d*, *mgrid*, *swim*, and *wupwise*. For more information about each application, please refer to [27]. The Intel Fortran and C/C++ compilers (version 8.1) were used to build the benchmark applications.

4.2 AMP Setup

To evaluate the power-performance efficiency of AMP over SMP systems, we created static AMP configurations on our platform through clock throttling and affinity control. In clock throttling, one can set the duty cycle to one of the seven available levels. Clock throttling does have a similar impact on performance as reducing the frequency [2].

The Linux 2.6.9 kernel supports clock throttling through a *sysfs* interface with appropriate drivers. The system was configured to enable CPU frequency scaling. The *p4-clockmod* driver was built into the kernel and the standard *sysfs* interface was used. The

frequency governor was set to user-space control, creating a static frequency scaling system. By static setup, we mean the duty cycle is set only once before the application run.

We have implemented a new Linux scheduler, to be called *power-saving scheduler* (PS-Scheduler), by modifying the Linux scheduler to reserve a single (logical) CPU that runs only kernel threads, leaving the rest of the CPUs in the system to execute all user threads at maximum frequency. This is accomplished by using the processor affinity properties available in the Linux 2.6.9 that allow processes to be bound to a specific set of (logical) processors, or an individual (logical) processor.

The available operating points on our platform are 1.4GHz, 1.225GHz, 1.05GHz, 875MHz, 700MHz, 525MHz, and 350MHz. The CPU frequency of the first physical processor was adjusted throughout the available frequency operating points for the execution of system activities, while the rest of processors in the system remained at 1.4GHz to run application threads only. However, our experimentation with the application benchmarks revealed the performance of the AMPs with the first processor at 875MHz or less abruptly decreases to less than half that of the system with the first processor at 1.05GHz. Therefore, we will not report those results in section 5.

It should be mentioned that the duty cycle could be set on a per physical processor basis on Intel multiprocessors. Therefore, in the case of an HT-enabled system, this creates an asymmetrical imbalance among the logical processors executing the user threads (the benchmarks). This can have a negative effect on the system performance.

4.2.1 AMP/SMP Base Power Consumption

Investigating the results in [2] one can conclude that their physical power measurements of a 4-way Intel Xeon 2.0GHz SMP server at consuming 220W when active are reliable. They provide a method based on historical data, where for a changing frequency, the power consumption is proportional to the square of the duty cycle. Therefore, if a 4-way 2.0GHz Intel Xeon SMP processor system consumes 220W when highly active, a 4-way 1.4GHz Intel Xeon SMP system would be expected to consume 107.8W ($220W \times (1.4/2.0)^2$) when highly active. This corresponds to an energy consumption of 26.95W per processor. When CPUs are in an idle state the power consumption of the 4-way 2.0GHz system was 48W, corresponding to 12W per processor. Applying the same scaling as used for the active case, we determine that the idle energy

consumption for a single 1.4GHz processor is 5.88W. Using the same methodology, one can easily find the highly active power consumption of an AMP with one CPU operating at 1.225GHz and the other three CPUs operating at 1.4GHz to be 101.5W. Knowing the approximate energy consumption of our AMP systems, when highly active or idle, allows us estimating the power consumption while executing the NAS and SPEC OpenMP benchmarks studied.

5. Experimental Results and Analysis

In this section, we describe the results and analysis of our experiments. We first compare the baseline performance of the *PS-Scheduler* with the default Linux scheduler both operating at full speed. We will then present the slowdown and energy savings of our AMP configurations with the new scheduler at different operating points over SMP at full speed. Finally, we analyze the power-performance efficiency of the proposed AMPs.

Both HT-enabled and HT-disabled results will be presented in this section. The notation HT-X refers to an SMP system with HT enabled and X user threads. The notation SMP-X refers to a standard SMP system (HT disabled) with X user threads. The notation AMP-X refers to an asymmetric multiprocessor system with X user threads.

5.1 PS-Scheduler vs. Linux Scheduler

The main motivation behind the new scheduler is to sustain the SMP performance with its original O(1) scheduler, and to provide room for energy savings. Our intention in this section is to see if the new scheduler performs in par with the SMP and HT configurations.

Figure 1 compares the performance of the *PS-Scheduler* running with three threads with two different SMP configurations running with the default scheduler on the NAS benchmarks. Note all four physical processors are operating at 1.4GHz. In the case of *PS-Scheduler*, the scheduler runs on processor zero, while the three application threads run on the other three processors. We included the results for SMP-3 with the default scheduler, as it might seem it would provide a fair comparison with the new scheduler in terms of the number of threads. However, SMP-3 on four physical processors produces an imbalanced load. SMP-4 is included because we want to see the trade-off between the AMP with the new scheduler and a full-load SMP running original scheduler.

As shown in Figure 1, the performance of the applications with the new scheduler compares favourably with the default Linux scheduler under the SMP-3 case. For the SMP-4 case, the new scheduler performs almost in par across the applications.

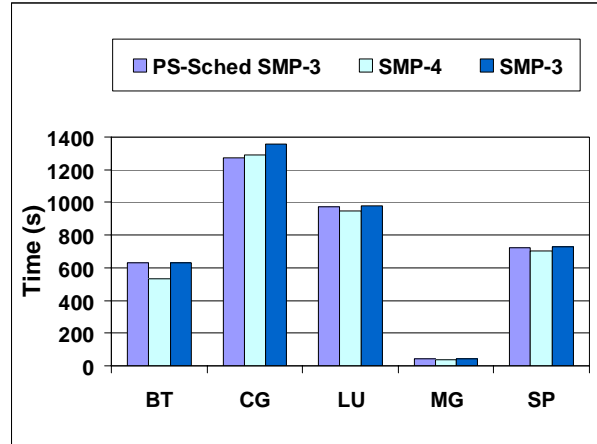


Figure 1: PS-Scheduler vs. O(1) scheduler for NAS benchmarks with HT-disabled processors at 1.4GHz

Figure 2 illustrates the performance of the NAS applications under the two schedulers with HT-enabled processors. Note all eight logical processors are operating at their maximum speed. In the case of new scheduler, the scheduler runs on logical processor zero, while the application threads run on the other seven logical processors. For all applications except CG, the new scheduler performs in par with its counterparts. For CG, the performance is almost sustained.

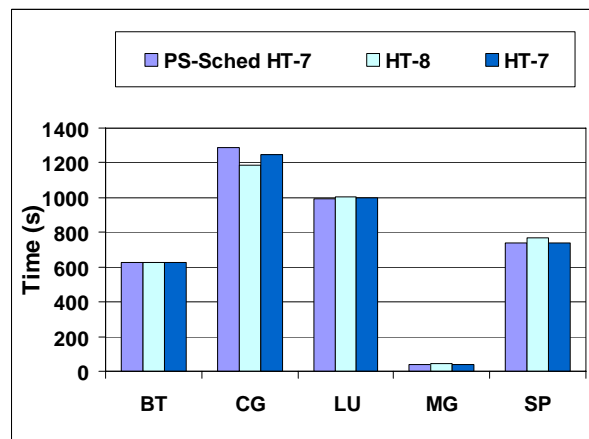


Figure 2: PS-Scheduler vs. O(1) scheduler for NAS benchmarks with HT-enabled processors at 1.4GHz

Space does not permit to include the figures for SPECComp. Our performance results indicate that the new scheduler performs well at full speed across the

SPECComp applications. The performance gain of the *PS-Scheduler* over the original scheduler for the case of SMP-4 is -9% to +11% for HT-enabled and between -22% and +7% for HT-disabled, with the average performance gain for HT-enabled being +0.1% and -10.5% for HT-disabled. Interestingly, the performance of the *PS-scheduler* sustains as the frequency of the first processor of the AMP system is reduced from 1.4GHz to 1.05GHz. This is shown in the next section.

5.2 Slowdown and Energy Savings

The energy savings and slowdown due to frequency scaling of the system while executing the NAS benchmarks are presented in Figure 3 and Figure 4. Note the AMP frequency in the figures corresponds to the CPU frequency of the first physical processor in the system. The remaining physical processors are running at maximum frequency. In the case of the HT-enabled processors, it should be noted that the first two logical processors are scaled in frequency. Therefore, one logical CPU that is executing the user threads has a reduced frequency in addition to the reserved CPU.

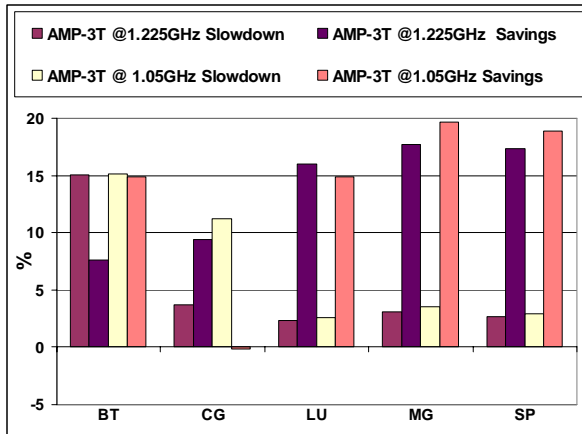


Figure 3: AMP slowdown and energy savings over SMP-4 for NAS benchmarks with HT-disabled processors

As shown in Figure 3, LU, MG, and SP benefit 15% to 19.7% energy savings with up to 3.5% performance loss. While BT enjoys a 15% energy savings at 1.05GHz, it does incur 15% slowdown. Only CG shows there is limited energy savings at the slowest gear. This is consistent with the CG results with HT-enabled processors shown in Figure 4, where there is a larger energy savings at 1.225GHz with little performance loss.

From Figure 4, only LU suffers in AMP with the new scheduler. We are currently investigating the reasons

behind this. Interestingly, MG and SP have both speedup as well as significant energy savings. The average energy savings of AMP at 1.05GHz with the *PS-scheduler* for NAS benchmarks is -12.3% to +13.5% for HT-enabled and between -0.2% to +19.8% for HT-disabled, with the average savings for HT-enabled being +5.8% and -13.6% for HT-disabled. Across the applications and for both HT-enabled and HT-disabled, the average savings is 9.7% while the average slowdown is 5.8%.

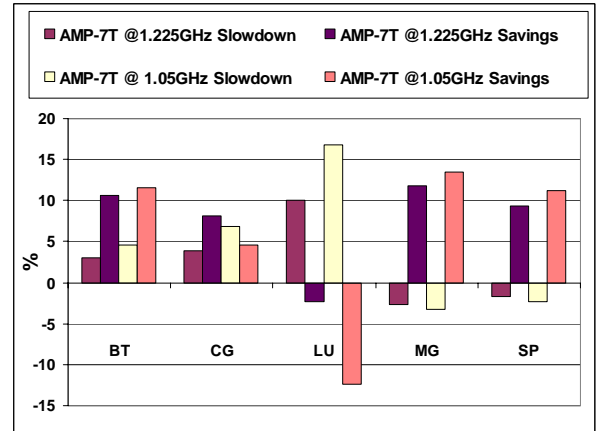


Figure 4: AMP slowdown and energy savings over HT-8 for NAS benchmarks with HT-enabled processors

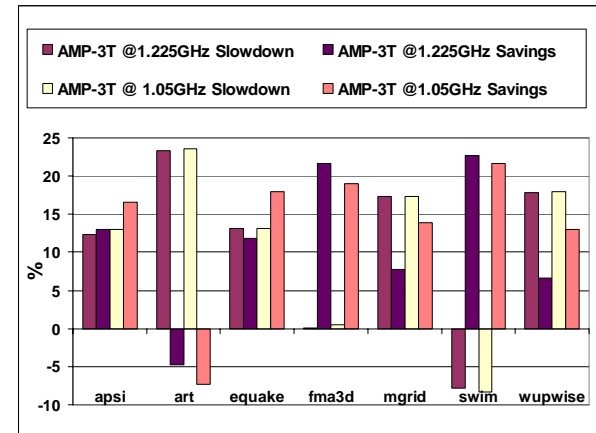


Figure 5: AMP slowdown and energy savings over SMP-4 for SPEC benchmarks with HT-disabled processors

The slowdown and energy savings of our AMP systems while executing the SPEC OpenMP benchmarks at different frequencies is presented in Figure 5 and Figure 6. In Figure 5, swim, and in Figure 6, fma3D and apsi enjoy speedup as well as significant energy savings. The swim benchmark is a memory intensive benchmark [9], therefore more opportunities for energy savings. The results for swim and fma3D show our superior AMP configuration over the AMP

proposed in [2], where it actually made fma3D and swim to perform worse than SMP. We are currently investigating the reasons behind the performance and thus energy savings of some of the applications using Intel VTune Performance Analyzer [12].

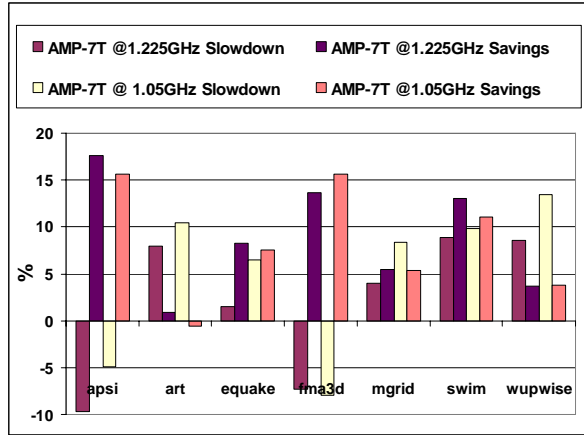


Figure 6: AMP slowdown and energy savings over HT-8 for SPEC benchmarks with HT-enabled processors

The average energy savings of AMP at 1.05GHz with the *PS-scheduler* for SPEC applications is -0.6% to +15.7% for HT-enabled and between -8.3% to +23.6% for HT-disabled, with the average savings for HT-enabled being +8.36% and -13.5% for HT-disabled. Across the applications and for both HT-enabled and HT-disabled, the average savings is 10.9% while the average slowdown is 5.1%. In summary, the performance of the new scheduler with the SPEC benchmarks is encouraging as it better models real-world applications that servers, such as the one used in this study, would be likely to run in a scientific environment.

5.3 Energy-Delay Analysis

Figure 7 through Figure 10 present the energy-delay of the AMP with the *PS-Scheduler* normalized to the original scheduler at 1.4GHz for both HT-8 and SMP-4 cases. An energy-delay of less than one shows that the savings in energy consumption outpace the corresponding increase in execution speed. Values greater than one represents the case in which the energy savings are not sufficiently offset by the increase in execution time. A downward trend in the graph from the left to the right indicates that the rate of decrease of energy consumption and the rate of delay related to said energy savings is diverging beneficially. With the exception of the LU in the HT-enabled case and the BT

in the HT-disabled case, the overall trends with the new scheduler are good.

The energy conservation of the new scheduler is observable as the majority of the benchmark applications are below the baseline energy-delay of one. With the exception of some of the SPEC benchmarks, namely art, mgrid and wupwise (equake and apsi are close to one) for the SMP-4 case, and the wupwise for the HT-8 case, the new scheduler manages to conserve an amount of energy that offsets the rise in total execution time. The full speed operating point is also very good in terms of power savings and overall speed. Note this operating point does not reflect an AMP but an SMP with the new scheduler, instead.

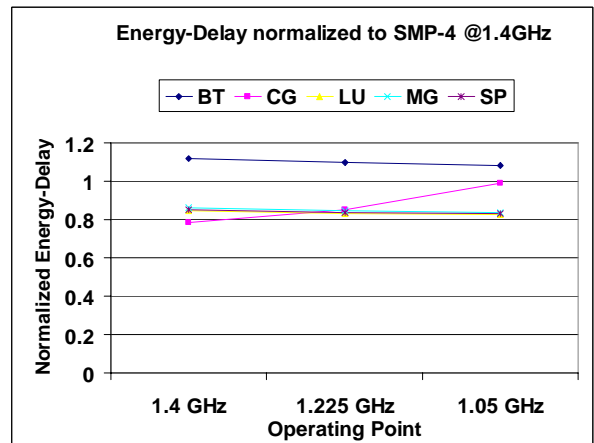


Figure 7: Normalized Energy-Delay for NAS benchmarks with HT-disabled processors

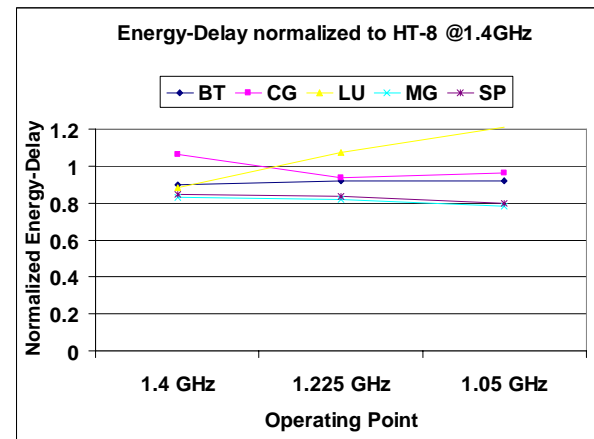


Figure 8: Normalized Energy-Delay for NAS benchmarks with HT-enabled processors

6. Conclusions and Future Work

The work in [2] was done on HT-disabled CPUs, but it provided us with the basis for the power estimates

that we make in this paper. This work builds upon the AMP results shown and extends the findings to HT-enabled processors. This paper demonstrates an alternative scheduling algorithm that decreases the performance impact of frequency scaling on commercial SMP systems. This scheduler modification is the early stage of a more substantial change in the Linux scheduler, demonstrating that further work in optimizing this scheduling method could provide further energy consumption benefits for real systems. This paper shows that power savings are possible in a high performance environment, and that with some changes to task scheduling, significant power savings can be realized while maintaining high performance.

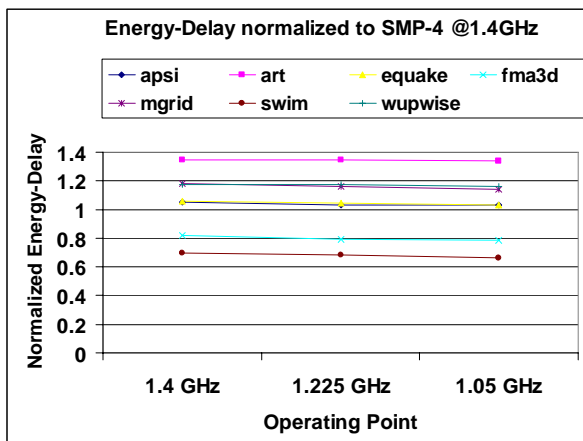


Figure 9: Normalized Energy-Delay for SPEC benchmarks with HT-disabled processors

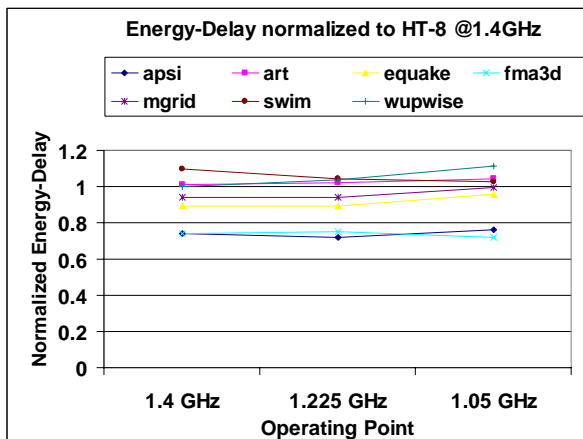


Figure 10: Normalized Energy-Delay for SPEC benchmarks with HT-enabled processors

The scheduler modifications presented here can be enhanced to increase their overall effectiveness using information from the available hardware performance counters. The further improvements to be made to the

scheduler will hopefully further enhance its performance and be able to provide performance equal to that of the original scheduler at full frequency, while operating the new scheduler at a lower frequency. We are currently in the process of enabling our infrastructure to physically measure the power at real-time.

Acknowledgments

This work was supported by grants from Natural Sciences and Engineering Research Council of Canada (NSERC), Queen’s University, Canada Foundation for Innovation (CFI), and Ontario Innovation Trust (OIT).

References

- [1] D.H. Albonesei et al., “Dynamically Tuning Resources with Adaptive Processing”, *IEEE Computer*, Vol. 36, No. 12, December 2003, pp. 49-58.
- [2] M. Annavaram, E. Grochowski, and J. Shen, “Mitigating Amdahl’s Law Through EPI Throttling”, In *Proceedings of the 32nd Annual International Symposium on Computer Architecture, ISCA’05, 2005*, pp. 298-309.
- [3] S. Balakrishnan, R. Rajwar, M. Upton, and K. Lai, “The Impact of Performance Asymmetry in Emerging Multicore Architectures”, In *Proceedings of the 32nd Annual International Symposium on Computer Architecture, ISCA’05, 2005*, pp. 506-517.
- [4] R. Bianchini and R. Rajamony, “Power and Energy Management for Server Systems”, *IEEE Computer*, Vol. 37, No. 11, November 2004, pp. 68-74.
- [5] S. Brosky, “Shielded CPUs: Real-Time Performance in Standard Linux”, *Linux Journal*, <http://www.linuxjournal.com/>.
- [6] K. Flautner, S. Reinhardt, and T. Mudge, “Automatic Performance-setting for Dynamic Voltage Scaling”, in *Proceedings of the 7th International Conference on Mobile Computing and Networking, MobiCom 01, 2001*, pp. 260-271.
- [7] V.W. Freeh, F. Pan, N. Kappiah, and D.K. Lowenthal, “Using Multiple Energy Gears in MPI Programs on a Power-Scalable Cluster”, In *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming, PPOPP’05, 2005*, pp. 164-173.
- [8] R. Ge, X. Feng, and K.W. Cameron, “Improvement of Power-Performance Efficiency for High-End Computing”, In *Proceedings of Workshop on High-Performance, Power-Aware Computing, HP-PAC 2005, April, 2005*.
- [9] R.E. Grant and A. Afsahi, “Characterization of Multi-threaded Scientific Workloads on Simultaneous Multithreading Intel Processors”, In *Proceedings of the Workshop on Interaction between Operating System and Computer Architecture, IOSCA 2005, October, 2005*.

- [10] C.-H. Hsu and W.-C. Feng, "A Power-Aware Run-Time System for High-Performance Computing", In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing, SC'05*, November, 2005.
- [11] C.-H. Hsu and U. Kremer, "The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction", In *Proceedings of the ACM SIGPLAN Conference on Programming Languages, Design, and Implementation, PLDI'03*, 2003. pp. 38-48.
- [12] Intel Inc., Intel VTune Performance Analyzer, <http://www.intel.com/software/products/vtune>, 2005.
- [13] H. Jin, M. Frumkin, and J. Yan, "The OpenMP Implementation of NAS Parallel Benchmarks and its Performance, Report NAS-99-011", *NASA Ames Research Center*, October, 1999.
- [14] T. Jones et al., "Improving the Scalability of Parallel Jobs by Adding Parallel Awareness to the Operating System", In *Proceedings of the ACM/IEEE Supercomputing Conference, SC'03*, November, 2003.
- [15] R. Kotla, S. Ghiasi, T. Keller, and F. Rawson, "Scheduling Processor Voltage and Frequency in Servers and Cluster Systems", In *Proceedings of the 19th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2005)*, April, 2005.
- [16] R. Kumar, D.M. Tullsen, N.P. Joupi, and P. Ranganathan, "Heterogeneous Chip Multiprocessors", *IEEE Computer*, Vol. 38, No. 11, November 2005, pp. 32-38.
- [17] R. Kumar, K. Farkas, N.P. Jouppi, P. Ranganathan, and D.M. Tullsen, "Single-ISA Heterogeneous Multi-core Architectures: The Potential for Processor Power Reduction", In *Proceedings of the 36th International Symposium on Microarchitecture, MICRO-36*, 2003, pp. 81-92.
- [18] Y. Li, D. Brooks, Z. Hu, and K. Skadron, "Performance, Energy, and Thermal Considerations for SMT and CMP Architectures", In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture, HPCA-11*, 2005, pp. 71-82.
- [19] R.L. McGregor, C.D. Antonopoulos, and D.S. Nikolopoulos, "Scheduling Algorithms for Effective Thread Pairing on Hybrid Multiprocessors", In *Proceedings of 19th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2005)*, April, 2005.
- [20] L.W. McVoy and C. Staelin, "lmbench: Portable Tools for Performance Analysis", In *USENIX Annual Technical Conference*, 1996, pp. 279-294.
- [21] Message Passing Interface Forum: MPI, A Message Passing Interface standard, Version 1.2, 1997.
- [22] T. Mudge, "Power: A First Class Design Constraint", *IEEE Computer*, Vol. 34, No. 4, April 2001, pp. 52-57.
- [23] OpenMP Architecture Review Board, OpenMP Specification Version 2.5, May, 2005.
- [24] F. Petrini, D.J. Kerbyson, and S. Pakin, "The Case of Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q", In *Proceedings of the ACM/IEEE Supercomputing Conference, SC'03*, November, 2003.
- [25] D. Pham et al., "The Design and Implementation of a First-Generation Cell Processor", In *Proceedings of International Symposium on Solid-State Circuits and Systems, ISSCC 2005*, 2005, pp. 184-186.
- [26] R. Sasanka, S. Adve, Y. Chen, and E. Debes, "The Energy Efficiency of CMP vs. SMT for Multimedia Workloads", *18th Annual International Conference on Supercomputing, ICS 2004*, pp. 196-206.
- [27] SPEC OMP Benchmark Suite, <http://www.spec.org/omp/>.
- [28] D. Tsafirir, Y. Etsion, D.G. Feitelson, and S. Kirkpatrick, "System Noise, OS Clock Ticks, and Fine-Grained Parallel Applications", In *Proceedings of the 19th Annual International Conference on Supercomputing, ICS'05, 2005*, pp. 303-312.
- [29] D. Tullsen, S. Eggers, and H. Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism", in *Proceedings of 22nd Annual International Symposium on Computer Architecture (ISCA'95)*, 1995, pp. 392-403.
- [30] H. Zeng, X. Fan, C. Ellis, A. Lebeck, and A. Vahdat, "ECOSystem: Managing Energy as a First Class Operating System Resource", In *Proceedings of the 10th International Conference on Architectural Support of Programming Languages and Operating Systems, ASPLOS-X, 2002*, October, 2002, pp. 123-132.