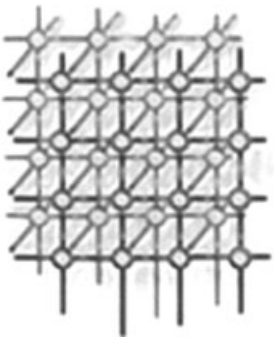
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2009; **21**:2355–2376
Published online 8 July 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1454



Improving energy efficiency of asymmetric chip multithreaded multiprocessors through reduced OS noise scheduling

Ryan E. Grant and Ahmad Afsahi*, †

Department of Electrical and Computer Engineering, Queen's University, Kingston, ON, Canada K7L 3N6

SUMMARY

The performance of the emerging chip multithreaded symmetric multiprocessors (SMPs) is of great importance to the high performance computing community. However, the growing power consumption of such systems is of increasing concern, and techniques that can be used to increase the overall system power efficiency while sustaining the performance are very desirable. Operating system (OS) noise can have a dramatic effect on the system performance. Effectively handling the smaller OS tasks while simultaneously preserving application thread synchronicity leads to gains in the overall system efficiency. Recently, under a fixed power budget, asymmetric multiprocessors (AMP) have been proposed to improve the performance of multithreaded applications. An AMP in this context is a multiprocessor system in which its processors are not operating at the same frequency. This paper proposes two simple scheduling methods that reduce the impact of OS noise, while simultaneously taking advantage of an opportunity to increase the overall machine energy efficiency on AMP servers. Prototyping AMPs on a commercial 2-way dual-core Hyper-Threaded (HT) Intel Xeon SMP server, using real power measurements across six SPEC OpenMP applications, indicates that the first proposed scheduler performs better on average for HT-enabled systems, whereas the second scheduler is superior on average for HT-disabled systems.

Received 18 November 2008; Revised 24 February 2009; Accepted 29 March 2009

KEY WORDS: asymmetric multiprocessors; OS noise; scheduling; energy efficiency; OpenMP

*Correspondence to: Ahmad Afsahi, Department of Electrical and Computer Engineering, Queen's University, Kingston, ON, Canada K7L 3N6.
†E-mail: ahmad.afsahi@queensu.ca

Contract/grant sponsor: Natural Sciences and Engineering Research Council of Canada; contract/grant number: RGPIN/238964-2005
Contract/grant sponsor: Canada Foundation for Innovation's; contract/grant number: 7154
Contract/grant sponsor: Ontario Innovation Trust's; contract/grant number: 7154

1. INTRODUCTION

Industry has recently adopted aggressive development of *chip multithreading* (CMT) [1] processors for general-purpose applications. The Intel Xeon [2], IBM Power6 [3], Intel Itanium 2 [4], and Sun UltraSPARC T2 [5] are examples of such microprocessors. CMT processors combine *chip multiprocessing* (CMP) [6] and *simultaneous multithreading* (SMT) [7] to provide better support for thread-level parallelism. CMPs contain multiple cores on a single chip allowing more than one thread to be executed at a time. Each core has its own resources as well as shared resources, such as the memory bus. The extent of shared resources varies between different implementations.

SMT is a technique that allows multiple independent threads to execute different instructions each cycle. Intel *Hyper-Threaded* (HT) processors [8] are implementations of SMT. An HT-enabled core appears as two logical processors to the operating system (OS), where each processor maintains a separate run queue. These logical processors share many hardware resources such as cache, execution units, translation look-aside buffers, branch prediction unit, and load and store buffers. It is shown that numerous complex interactions among the shared resources may affect the performance of multithreaded applications running on SMTs [9–13].

The emerging CMT-based *symmetric multiprocessor* (SMP) servers present new challenges as well as new opportunities to maximize the performance provided that the resources available could be shared efficiently. The performance of such servers running multithreaded applications in OpenMP [14] is of great importance to the high-performance computing (HPC) community. However, their growing power consumption is of increasing concern, and techniques that can be used to increase the overall system power efficiency while sustaining the performance are needed.

Researchers from Intel [15,16] have illustrated how an *asymmetric multiprocessor* (AMP) can be built from a commercial SMP. AMP, a form of single-ISA heterogeneous architecture [17], 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 [15] consider AMP configurations that can achieve an average of 38% wall clock speedup over the SMP on a wide range of multi-threaded applications under a given power budget. While the work in [15] assumes a fixed power budget, this paper evaluates the power-performance efficiency of SMT-capable AMPs for the sake of energy saving with a minimal impact on the performance of OpenMP multithreaded applications. Contrary to the work in [15] that focuses on HT-disabled AMPs, we present the performance and energy-saving results of both HT-enabled and HT-disabled AMPs.

Previous research has shown that system noise including OS interference with the application has a dramatic effect on HPC [18–20]. However, the competition for resources between highly CPU intensive tasks and the OS yields opportunities to increase the overall system efficiency. For this, we propose two different schedulers to reduce the impact of OS noise and to potentially reduce the power consumption of a server, while producing less of a performance impact on the system than the original scheduling algorithm. Using static clock throttling and processor affinity, the first method of thread management, *PS-Sched-1* (power-saving scheduler 1), originally proposed in [21], masks off a single logical/physical core for OS tasks only and scales its frequency in order to save power, while running user threads on the remaining cores at maximum frequency. In the second proposed method [22], *PS-Sched-2*, the system has one core running at its full clock speed performing OS and

DOI: 10.1002/cpe

user tasks while the remaining cores run user threads at lower operating frequencies. The difference in operating frequency between the OS/user core and the other cores helps to ensure that the OS core can maintain synchronicity while being interrupted to perform OS tasks.

Overall, the proposed schedulers do a good job at reducing the energy consumption of the AMPs running SPEC OMP [23] applications while having a minimal impact on the performance of the system. Our performance results on a two-way Intel Xeon CMT-based SMP server show that *PS-Sched-2* performs better on average for the HT-disabled architectures and at some operating points for HT-enabled architectures, whereas *PS-Sched-1* is superior in terms of overall average slowdown and energy savings for the HT-enabled architectures.

The remainder of this paper is organized as follows. Section 2 discusses the related work and motivation behind this work. Section 3 defines the metrics used in this work. The experimental framework including system configuration, terminology, AMP setup, and power measurement is discussed in Section 4. In Section 5, we present the impact of OS noise on cache and system performance. Section 6 proposes the power-saving schedulers. In Section 7, we present and analyze the power/performance results including application slowdown and energy savings. Finally, Section 8 concludes the paper.

2. RELATED WORK AND MOTIVATION

Power has become a critical design constraint for modern microprocessors [24], and there have been quite a number of studies on the subject of energy reduction of modern day processors. These include dynamically tuning processor resources with adaptive processing [25], comparison of SMT and chip multiprocessing (CMP) [26,27], heterogeneous multi-core architectures [17,28,29], and the effect of scheduling on power and performance [30,31]. Most of such research has been done with single-threaded applications and through simulations, or analytical methods. Our work in this paper concerns multithreaded parallel workloads on real CMT-based SMP systems.

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 [32,33], energy conservation has recently become important in servers and clusters [15,34–37] with their non-interactive HPC workload in message passing interface (MPI) [38] and OpenMP. In such systems, the focus is to improve reliability and to reduce the cost of powering and cooling.

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!*, *Cool'n'Quiet*, and *CoolCore* technology from AMD. Several researchers have proposed methods to utilize such features to reduce power and energy consumption, such as devising a compiler algorithm for optimizing single-threaded programs for energy usage on laptops [39], and power and energy management techniques for servers and data centers [40,41]. In addition, work has been done on HPC workloads in SMP and AMP servers [15,16,34] and in high-performance clusters [35–37].

DOI: 10.1002/cpe

The authors in [15,16] describe the process of creating an AMP node from a commercial Intel SMP server. In [15], Annavaram *et al.* analyze the energy per instruction (EPI) gains that can be obtained from using CPUs operating at different frequencies. 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 2.0 GHz Intel Xeon processor. They used the SPEC OMP benchmarks as well as several other applications in their study; however, their work did not address HT-enabled systems [15].

In [16], Balakrishnan *et al.* investigated the impact of performance asymmetry of different AMPs on commercial applications as well as SPEC OMP 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 that faster cores never go idle before slower ones. For SPEC OMP scientific applications with tight coupling among different threads, they found stability, but with poor scalability as the slowest core forces the 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 a degraded performance. This work did not address HT-enabled systems, either. Although, the work in [16] is only focused on performance asymmetry, they indicate that AMP systems can be effective for power/performance efficiency.

This paper builds upon the work in [15,16]. 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 two new kernel schedulers for AMP systems.

Previous research has shown that OS noise may have a dramatic effect on HPC systems [18–20]. In [18], Petrini and his colleagues noticed that their MPI 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 system noise including such things as OS daemons, kernel threads, and OS real-time interrupts. The authors in [19,20] also verified the effects of system noise on the performance of applications.

While there have been some effective techniques proposed in [18,19] to reduce the impact of system noise, such as removing unnecessary OS daemons and kernel threads (or moving to another processor), lowering tick rate, and co-scheduling, leaving one core for OS tasks is a simple, viable option to effectively separate system noise from the computation. Meanwhile, past work on real-time processing with Linux schedulers [42] 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) core, we can speculate that by avoiding swapping the parallel threads in and out of the other cores 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 correspond to a full load for a single (logical) processor, there exists an opportunity to reduce the frequency of the reserved CPU (as in the proposed PS-Sched-1) such that its load is as close to 100% as possible. This frequency scaling of the reserved CPU has a power savings effect.

DOI: 10.1002/cpe

While our work is focused on minimizing the impact of OS noise on parallel applications with the intention to improve their energy efficiency, Mogul and his associates [43] advocate the use of simple cores optimized for energy-efficient execution of OS code in asymmetric multi-core systems. They dynamically switch between cores in OS kernel codes. Their work is based on cycle-level simulation, and does not consider real-world parallel applications.

3. METRICS

Metrics such as performance, power, energy, and energy-delay are traditionally used in studying the power-performance characteristics of applications running on a system.

3.1. Performance

HPC has always been concerned with performance. The performance of an application running on a system is given by wall clock execution time, D.

3.2. Power

CPU is the major power-consuming component in a system. Theory [24] tells us that the power consumed by a CMOS processor, in watts, 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 as follows:

$$P = \alpha C V^2 f \tag{1}$$

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

3.3. Energy

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

$$E = \int_0^D P \, \mathrm{d}t = P_{avg} \times D \tag{2}$$

3.4. Power-performance efficiency

The power-performance 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 follows:

$$E \cdot D = E \times D \tag{3}$$

DOI: 10.1002/cpe

4. EXPERIMENTAL FRAMEWORK

The experiments were conducted on a Dell PowerEdge 2850 SMP server. The PowerEdge 2850 has two dual-core 2.8 GHz Intel Xeon EM64T processors (Paxville core), with a 12 KB shared execution trace cache, and 16 KB L1 shared data cache on each core. A private 2 MB L2 cache is available for each core on the chip. There are 4 GB of DDR-2 SDRAM on an 800 MHz Front Side Bus.

Using LMbench [44] we have measured the L1, L2, and main memory latencies of the processor at 1.43 ns, 10.61 ns, and 136.85 ns, respectively. The main memory read and write bandwidths are 3.57 and 1.77 GB/s, respectively. The system was tested to determine if any memory bandwidth differences existed between the operation of threads on a single physical chip and the operation of those same threads spread out to both physical chips. It was discovered that the main memory read and write bandwidths when using two physical chips are 4.43 and 1.61 GB/s, respectively.

The OS used for the testing was a Fedora Core 4 distribution with Kernel 2.6.11. This kernel provided the option to limit the number of processors that can be activated by the OS. The number of active processors was limited using the maxcpus $= X$ boot option of the kernel. This option causes the kernel to only initialize and use X logical processors. This method of disabling additional processors is preferable since it better emulates a smaller system. To test the system in a variety of configurations some of the tests were run while masking off some of the available processors in the system, enabling us to test the performance of the system using different thread distributions between the existing resources. The default Linux scheduler was used for assigning the individual threads among the specified processors.

4.1. Terminology

The test results detailed in this paper show the results for the testing on a variety of different configurations. Figure 1 presents the labeling used for the hardware contexts in the HT-enabled and HT-disabled systems [11]. Such labeling will help understand the hardware contexts available for use in each configuration.

Given that hybrid chip multithreaded SMPs can be configured in multiple configurations [11], a naming scheme must be created to describe each configuration uniquely. Table I shows the different configurations used in our study under three different schedulers: the (Linux) default scheduler, and our proposed schedulers, PS-Sched-1 and PS-Sched-2 (to be discussed in Section 6). The basic terminology used to describe these configurations is comprised of three parts. The first part is either HT_{off} or HT_{on}, which describes the state of Hyper-Threading in the system. The second term indicates the total number of application threads that was used. The third term represents the number of physical processor chips used in the tests (i.e. the number of dual-core processors used, either 1 or 2). Note that for AMP systems using the proposed schedulers, the aforementioned terminology will be prefixed by AMP, such as AMP-HT_{on}-3-1.

4.2. SPEC OMP application benchmarks

The SPEC OMPM2001 (version 3.0) suite of applications [23] is from the SPEC High-Performance Group. The suite consists of a set of OpenMP-based scientific applications. These programs were

DOI: 10.1002/cpe

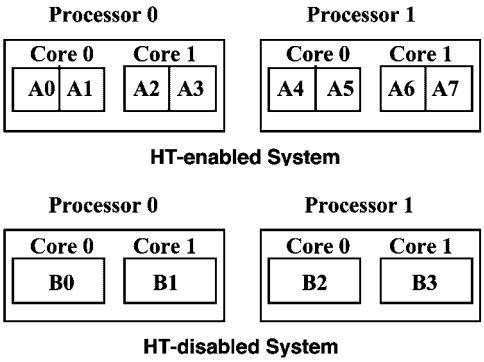


Figure 1. System configuration labeling.

Table I. Configuration information.

Terminology			
Default scheduler, PS-Sched-2	PS-Sched-1	Hardware contexts	Corresponding architecture
HT_{on}-2-1	HT_{on}-1-1	A0, A1	SMT
HT_{off}-2-1	HT_{off} -1-1	B0, B1	CMP
HT_{on}-4-1	HT_{on}-3-1	A0, A1, A2, A3	CMT
HT_{off}-2-2	HT_{off}-1-2	B0, B2	SMP
HT_{on}-4-2	HT_{on}-3-2	A0, A1, A4, A5	SMT-based SMP
HT_{off}-4-2	HT_{off}-3-2	B0, B1, B2, B3	CMP-based SMP
HT_{on}-8-2	HT_{on}-7-2	A0, A1, A2, A3, A4, A5, A6, A7	CMT-based SMP

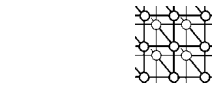
originally part of the SPEC CPU2000 suite and were parallelized by inserting OpenMP directives. We worked with six of the nine SPEC applications, specifically apsi, art, fma3d, mgrid, swim, and wupwise. More information about each application can be found in [23]. The Intel Fortran and C/C++ compilers (version 8.1) were used to build the benchmark applications.

Application characteristics such as the total number of L2 cache access and L2 cache hit ratios were gathered at run time using the Oprofile 0.9.3 suite [45], with data processing using the oprofile-report tool. The collection of data was done with parameters that have been determined to produce $\sim 2\%$ overhead.

4.3. AMP setup

To evaluate the power-performance efficiency of AMP systems with the proposed schedulers, we created static AMP configurations on our 2-way dual-core platform through clock throttling and affinity control. In clock throttling, one can set the duty cycle to one of the seven available levels.

DOI: 10.1002/cpe



Clock throttling has a similar impact on the performance as reducing the frequency [15]. We have observed that the clock throttling of a processor results in an expected loss of performance for the scaled processor. No performance drop was observed in the remaining non-throttled CPUs when individual CPUs are clock throttled.

The Linux 2.6.11 kernel supports clock throttling through a sysfs interface with appropriate drivers. We built the p4-clockmod driver into the kernel, and the standard sysfs interface was used to enable CPU frequency scaling using clock throttling. The frequency governor was set to user-space control, creating a static operating point in which the duty cycle is set only once before the application run.

The available operating points on our platform are 2.8 GHz, 2.4 GHz, 2.1 GHz, 1.8 GHz, 1.5 GHz, 1.2 GHz, 900 MHz, and 600 MHz. However, for legibility and due to limited space, we only present the results for 2.8, 2.4, 2.1, and 1.8 GHz.

4.4. Power measurement

In this paper, we present the real power consumption of the applications running on our system. Our measurement infrastructure consists of a Keithley 2701/7710 Digital Multimeter (DMM), a shunt resistor, and a profiling PC, as shown in Figure 2. Owing to the nature of the rack-mounted system, power measurements on the output of the power supply inside the machine were impractical. Therefore, the power consumption numbers presented are affected by the power supply losses and take into account all the components of the system fed by the main power supply.

We measure the power/energy consumption of our system by measuring the voltage of a shunt resistor placed between the wall power outlet and the node. Knowing the value of the resistor, we first calculate the current and then the power and the energy consumption of our platform. We read 60 AC-voltage samples per second. In the DMM, the signal first goes through an internal analog RMS-converter, where 1000/60 DC samples are read out and averaged for each AC sample.

The testing set for the real-world validation was changed to only the SPEC OMP benchmark suite in order to achieve longer run times and better overall power measurements. The power measurement equipment was validated through the use of a Wattsup EPS Pro power meter, and found to be within the acceptable error range of the two devices, with the Keithley meter having a $\pm 1\%$ error range.

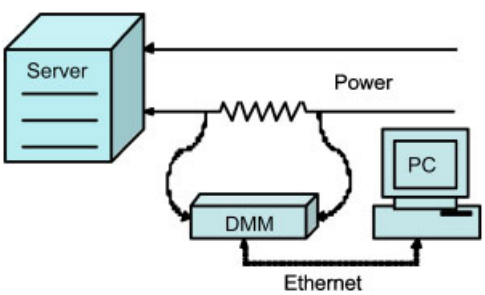


Figure 2. Power measurement methodology.

DOI: 10.1002/cpe

5. OPERATING SYSTEM NOISE EFFECTS

OS noise is a term describing the overhead caused by running the requisite daemons and services required by the OS. For our purposes, we define OS noise as was defined by Petrini and his associates [18], as any process not related to the current active computational task that is running on the system. In our system, these tasks can be simple system services such as the cron daemon or event handler, or even an ssh daemon. We do not define OS noise to be any calls from the program under test to the OS as in [43], as these are assumed to be a part of the program under test's execution, and therefore not external OS noise.

Operating systems have been shown to have an effect on the performance of intensive workloads in large-scale clusters [18,19]. OS noise is also of significant concern to real-time applications as interruptions during processing can lead to failure to meet real-time deadlines [42]. The load caused due to OS services is typically marginal, but the context switches and computational time devoted to these services can cause the main computational workload to lose synchronicity which degrades performance, as the workload must wait at barriers in the code. As the number of simultaneously executing threads increases so does the penalty that is incurred at barriers, as a larger number of threads must wait for the small number of threads lagging behind the group in the workload. These effects have not been previously investigated for CMT/SMP hybrid architectures running multithreaded parallel applications. In order to study the effect of OS noise, the OS noise first needed to be isolated. This was accomplished by masking off a single processor (logical or physical depending on the configuration), and assigning OS tasks to that processor.

Figure 3 depicts the ratio of the overall number of L2 cache references and L2 hit rates, as well as the speedup, of a non-OS noise configuration (using a dedicated OS processor) versus the OS noise configuration (default Linux configuration), averaged across all SPEC OMP applications studied under different architectural configurations. The resulting ratios show a beneficial effect when they are below 1 for the L2 cache access ratio, meaning that the cache is accessed less without OS noise, but show beneficial behavior for values above 1 for the L2 hit rate ratio and speedup, meaning that the system has a better cache performance and faster execution time.

Examining the results for the HT-enabled architectures, we can see that the configurations that see a drop in the total number of L2 cache accesses (the HT_{on}-4-1 and HT_{on}-4-2 configurations)

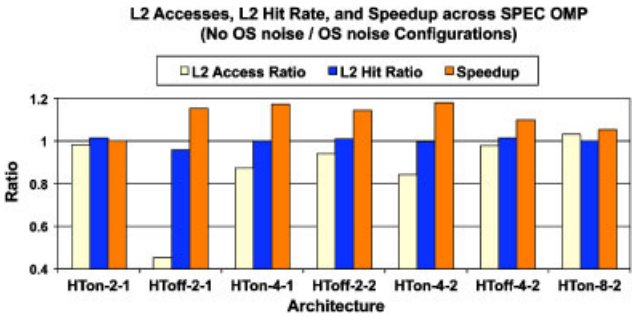


Figure 3. Improvement of overall L2 cache accesses, L2 cache hit rates and execution times without OS noise across SPEC OMP applications.

DOI: 10.1002/cpe

also see the greatest amount of speedup. For the HT_{on}-8-2 configuration, the mediocre speedup for the non-OS noise configuration is balanced by minimally increased cache accesses which are in turn offset by a higher cache hit rate. The HT_{on}-2-1 configuration sees a small increase in the number of cache accesses for the non-OS noise configuration for some benchmarks (not shown) and an increase (0.4%) in its L2 cache hit rate. It should be noted that the results for the HT_{on}-2-1 configuration are affected by the superb performance of the swim benchmark which sees a decrease in cache accesses of 29% and a subsequent speedup of 13.4%. This offsets the poorer performance seen over the other benchmarks, which results in a favorable overall average for the HT_{on}-2-1 configuration while the reality is that the reduction of OS noise is significant for certain applications, the majority of the applications do not see a major benefit for this architecture. The other architectures see an increase in speed corresponding to the reduction in the overall cache activity, with the HT_{on}-4-1 configuration seeing a 12.7% reduction in cache activity with a speedup of 17.3%. The HT_{on}-4-2 configuration sees a 15.8% reduction in cache activity and a speedup of 18.0%. The HT_{on}-8-2 configuration sees a 3.1% increase in cache activity but manages a speedup of 5.3%. This illustrates that although the cache activity is a significant factor in increasing the performance when OS noise is eliminated, it is not the only factor that affects the performance.

Examining the HT-disabled architectures, we can see the same trend as was observed for the HT-enabled system configurations. Although the cache-hit rate is not significantly affected by the reduction in OS noise, the overall number of cache accesses declines in all of the HT-disabled cases. The speedup that corresponds to this reduced number of memory accesses is proportional to the reduction in total memory accesses. The HT_{off}-2-1 configuration shows the limit to the speedup that can be achieved using this method as it shows an excellent reduction in memory accesses (54.7%) when OS noise is isolated, but only shows a speedup of 15.4%. In contrast, the HT_{off}-2-2 configuration has a reduction of 6% in its overall cache accesses but shows a speedup of 14.5%. Finally, the HT_{off}-4-2 configuration sees a 2.2% reduction in cache activity with a speedup of 9.9%. The HT_{off}-4-2 configuration illustrates a similar behavior to the HT_{on}-8-2 configuration in that it shows relatively small reduction in cache activity while providing a good speedup. This demonstrates that the cache activity is the major, but not the sole source of speedup when eliminating OS noise.

The results of the experiments conducted in this section indicate that a potential performance increase could be realized in the emerging chip multithreaded multiprocessors by taking advantage of OS noise effects. In essence, isolating the OS noise on a single core was shown to be effective in improving runtimes for the majority of cases.

6. THE PROPOSED SCHEDULERS

In this section, we propose two schedulers that bind OS noise activities to a single (logical) core with the intention to increase energy efficiency while improving/sustaining performance.

6.1. Power-saving scheduler, PS-Sched-1

As shown in Section 5, isolating the OS noise on a single (logical) core is effective in improving the performance of applications in most cases. However, we can further exploit methods of isolating

DOI: 10.1002/cpe

OS noise, by recognizing that the load caused by OS noise is not significant enough to require a single (logical) core operating at full speed. This implies that energy savings can be realized by DVFS or clock throttling on the OS bound core. For this, the default Linux scheduler is modified to automatically allocate OS functions to a single core, which is operating at lower frequencies, while other cores executing the threads of the parallel application at maximum frequency. The difference in frequency for cores resembles an AMP system. As stated earlier, OS threads are identified by being threads not related to the running user tasks on the system; such threads are assigned a processor mask to ensure that they only run on the OS core. This is accomplished by using the processor affinity properties available in the Linux 2.6.11. This method, which utilizes only a single frequency scaled core and reserves the said core for only OS threads, is referred to as *Power-Saving Scheduler*, *Method 1* (*PS-Sched-1*).

6.2. Power-saving scheduler, PS-Sched-2

With the emergence of many-core processors in the near future, one of the likely limitations of the *PS-Sched-1* scheduler is in its power savings potential, as only one core is operating at a lower frequency. An alternative method of isolating OS noise is therefore proposed here, namely the *Power-Saving Scheduler*, *Method 2* (*PS-Sched-2*). In this method, one core is still designated as the core to handle OS tasks, however, that core is also permitted to run application threads. The motivation behind this method is to determine the potential performance and the energy consumption benefits that could be achieved if a system was assembled using a single core that operates at a higher frequency than the rest of the cores in the system, but also handles all of the OS tasks. Therefore, the operating points that are reported for this alternative method represent the speed of all of the cores in the system excluding the OS/application core, which runs at maximum frequency. This approach attempts to balance the overall thread progression among the cores such that the faster OS/application core is synchronized with the other slower application cores even after being interrupted to perform OS tasks.

7. EXPERIMENTAL RESULTS AND ANALYSIS

The main motivation behind the proposed schedulers is to sustain the performance with the original scheduler, while providing energy savings. We would like to first understand whether the proposed schedulers perform on par with the default scheduler in both HT-enabled and HT-disabled configurations, when operating at maximum frequency. Second, we compare their baseline power consumptions. We will then present the slowdowns and energy savings that can be gained with the new schedulers when the system is operating as an AMP. Finally, we will analyze the power-performance efficiency of the schedulers for the AMP systems under consideration. Note that for legibility and space considerations, we will only present and analyze the slowdowns, energy savings, and energy-delay of the proposed schedulers in comparison with the largest available architectural configurations in the system using the default scheduler, that is HT_{on}-4-1 (CMT), HT_{on}-8-2 (CMT-based SMP), and HT_{off}-4-2 (CMP-based SMP), and for 2.8 GHz, 2.4 GHz, 2.1 GHz, and 1.8 GHz. Similar results have been observed for other configurations.

DOI: 10.1002/cpe

7.1. Baseline performance

Figure 4 compares the baseline performance of the proposed schedulers with the default scheduler for the SPEC OMP benchmarks for both HT-disabled and HT-enabled systems. For the HT-disabled case (HT_{off}-3-2 (PS-Sched-1), and HT_{off}-4-2 (PS-Sched-2 and default)), all four cores are operating at maximum frequency (2.8 GHz). For the PS-Sched-1 scheduler, the scheduler runs on core zero of the first processor, while the three application threads run on the other three cores. For the PS-Sched-2 scheduler, the scheduler again runs on core zero of the first processor, while the four application threads run on the available cores. Similar analogy can be made about the HT-enabled cases under study.

The performance of the PS-Sched-1 compared with the default scheduler is promising, with almost all configurations and benchmarks seeing an improvement in the performance. The HT_{on}-3-1 configuration shows the best speedup of the PS-Sched-1 configurations at an average of 17.2%. The HT_{off}-3-2 configuration shows a speedup of 8.6%, while HT_{on}-7-2 configuration shows a speedup of 1.4%. Overall, the performance gain of the PS-Sched-1 for the configurations compared with the original scheduler ranges from 1.4 to 17.2%, with an average performance gain of 7.2%.

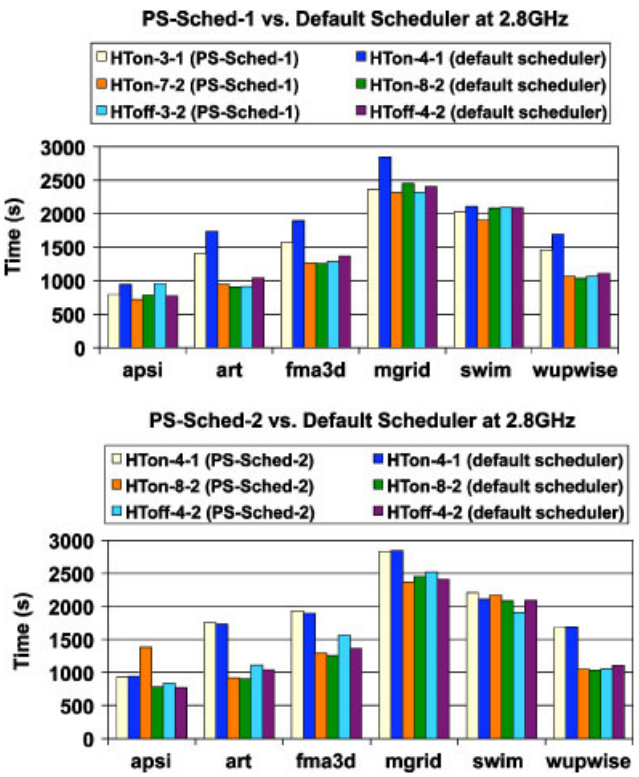


Figure 4. Runtimes of PS-Sched-1 and PS-Sched-2 vs the default scheduler for SPEC OMP benchmarks.

DOI: 10.1002/cpe

PS-Sched-2 provides benefits in other frequency gears, but does not provide a better performance at the 2.8 GHz frequency. The HT_{on}-4-1 configuration sees a 0.9% slowdown, whereas the HT_{on}-8-2 configuration sees a 8.4% slowdown and the HT_{off}-4-2 configuration sees a 2.7% slowdown compared with the default scheduler.

7.2. Baseline power consumption

The average instantaneous power measurements for each of the configurations at maximum operating frequency are presented for the proposed and the default schedulers in Figure 5. The initial results of the power consumption of the system with the new scheduler are surprising. In many cases, the new scheduler actually leads to an increase in power consumption. The highest average instantaneous power consumption of any scheduler at 2.8 GHz occurs with PS-Sched-1 with a high of 323.9 W on average for the HT_{on}-7-2 configuration. This compares to the power consumption for the default scheduler of 308.2 W for the HT_{on}-8-2 configuration. The average power for

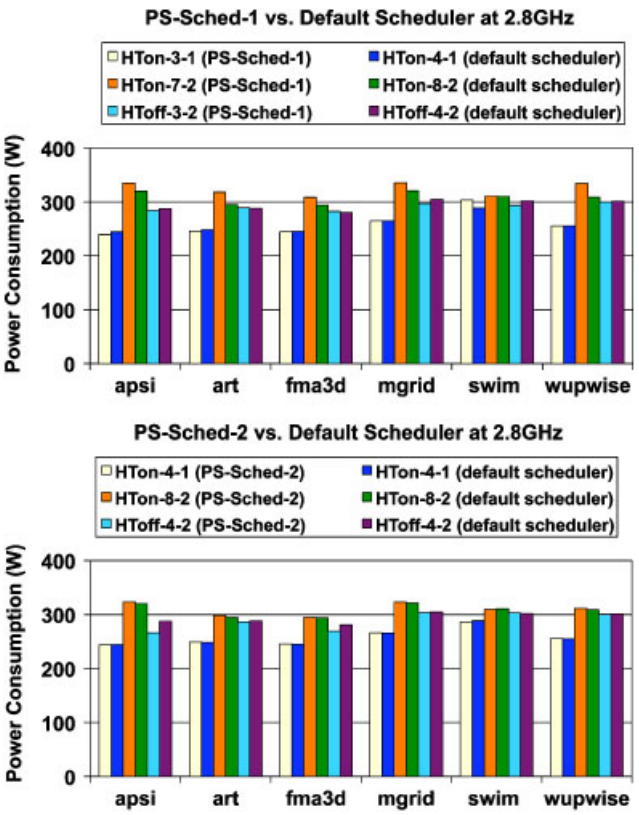


Figure 5. Average power consumption of PS-Sched-1 and PS-Sched-2 vs the default scheduler for SPEC OMP benchmarks.

DOI: 10.1002/cpe

PS-Sched-2 is 310.0 W for the HT_{on}-8-2 configuration. The average power consumption of the default scheduler for all configurations was 286.72 W, with the averages being 291.4 and 285.2 W for the PS-Sched-1 and PS-Sched-2 approaches, respectively.

7.3. Slowdown and energy savings

This section presents the actual slowdown in wall clock time that occurs when using the PS-Schedulers and the corresponding energy savings that occur. Please note that the AMP frequency in the following figures for PS-Sched-1 corresponds to the CPU frequency of the first physical processor in the system. The remaining physical processors are running at maximum frequency. For the PS-Sched-2, the first physical processor operates at maximum frequency, while the AMP frequency shown corresponds to the frequency of the other CPUs in the system.

It should be mentioned that the duty cycle could be set on a per physical core basis on Intel multiprocessors. Therefore, in the case of an HT-enabled AMP system and for PS-Sched-1, 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. Note that naturally there will be an asymmetrical imbalance among the logical processors for the AMP systems with PS-Sched-2.

The AMP-HT_{on}-3-1 results in Figure 6 show that by using PS-Sched-1, speedup occurs for both the 2.4 GHz and 2.1 GHz operating points, providing energy savings of between 18 and 32% while simultaneously reducing execution time for all applications. The average speedup for the 2.4 and 2.1 GHz operating points is 13.7% across all of the applications.

The PS-Sched-2 AMP-HT_{on}-4-1 configuration has its best operating point at 2.4 GHz, with speedup ranging from -1.5 to 1.2% and energy savings ranging from -6.7 to 8.2%. The average speedup is 0.5% with an average energy savings of 0.4%. The swim benchmark provides a speedup across all of the operating points, with speedup of 9.7 and 9.4% for the 2.4 and 2.1 GHz operating points, beating the speedup of the PS-Sched-1, by 5.7 and 3.3% in relation to the default scheduler at 2.8 GHz. However, at these operating points, the PS-Sched-2 method uses 6% more energy in the 2.4 GHz operating point than the default scheduler and 4.5% more at 2.1 GHz. This still beats the PS-Sched-1 performance for swim that uses 13.9 and 14.8% more energy than the default scheduler at 2.8 GHz for the 2.4 and 2.1 GHz operating points.

In general, the PS-Sched-2 shows a poorer performance and energy savings versus PS-Sched-1 on average. Although it provides some good performance/power savings opportunities for swim benchmark, the overall performance is inferior to PS-Sched-1, leading to the conclusion that for this architecture, PS-Sched-1 is the clear winner.

The AMP-HT_{on}-7-2 configuration for PS-Sched-1 shown in Figure 7 has good performance for the apsi, mgrid and swim, but the other benchmarks show some slowdown and consequently, poor energy savings. The mgrid and swim benchmarks have higher power usage than the reference configuration and therefore have poor energy savings. Overall PS-Sched-1 has a slowdown of 0.7% and an energy savings of 1.6% across all applications and operating points.

The PS-Sched-2 approach shows an average slowdown of 0.8% (neglecting apsi at 2.4 GHz) and an average energy savings of 3.8% across all applications. Owing to the larger energy-savings percentage versus the appreciable slowdown, this represents a positive gain for all application lengths, of 3% power efficiency, regardless of the length of computation. When apsi at 2.4 GHz

DOI: 10.1002/cpe

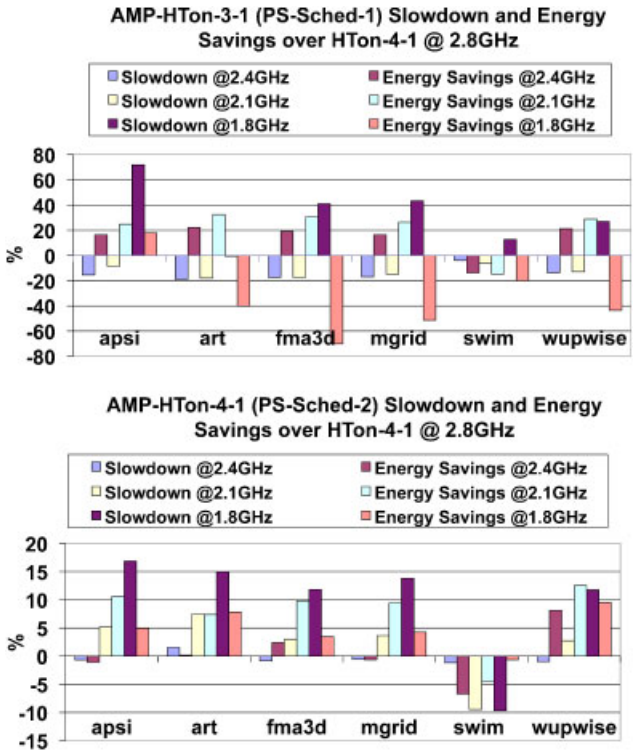


Figure 6. Slowdown and energy savings for AMP-HTon-3-1 and AMP-HTon-4-1 over HTon-4-1.

is included, the average energy savings become negative ($<-1\%$) and the slowdown seen is 5.2%. Comparing the results for PS-Sched-2, AMP-HT_{on}-8-2 configuration with those of the PS-Sched-1, AMP-HT_{on}-7-2 configuration in Figure 7, we can see that the second method provides good improvement in the performance and energy savings of the system for many of the benchmarks compared with the first proposed method. Of particular interest is the apsi benchmark results which show up to $\sim 50\%$ energy savings with very little performance impact at 1.8 GHz.

The PS-Sched-2's performance with apsi shows that at higher frequencies (2.4 GHz), the system is unable to keep up effectively, and shows an 80.1% slowdown and -83.1% energy savings. This is most likely due to poor thread synchronization, as this problem can be corrected with apsi by reducing the frequency of the user thread only CPUs. This also accounts for the marked improvement in the PS-Sched-1 case, as the alleviation of OS noise allows for better synchronization and therefore better results.

The final architecture that was examined for PS-Sched-1, AMP-HT_{off}-3-2, has its slowdown and energy savings illustrated in Figure 8. The results are varied, with apsi and mgrid showing excellent results, while art shows terrible slowdown and consequently poor energy-savings numbers. The fma3d and wupwise applications show that some energy savings are possible, but the resulting slowdown is slightly greater than the potential energy savings. The swim benchmark shows some speedup and some potential for energy savings as well.

DOI: 10.1002/cpe

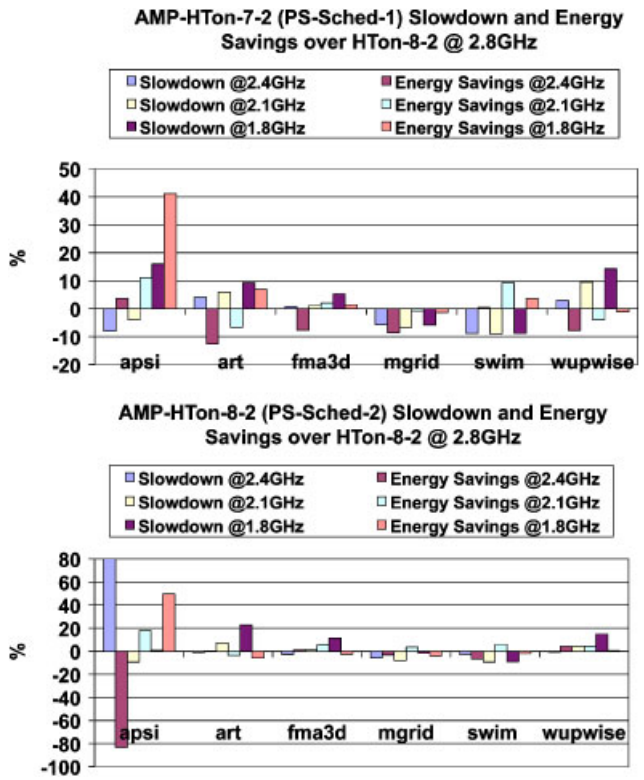


Figure 7. Slowdown and energy savings for AMP-HTon-7-2 and AMP-HTon-8-2 over HTon-8-2.

The PS-Sched-2 approach in Figure 8 shows that it performs well at 2.4 GHz for all applications, and has poorer performance at lower frequencies. The best performing benchmark at 2.4 GHz is the apsi benchmark with an 8.9% speedup and an 18.1% energy savings over the default scheduler at 2.8 GHz. At the 2.4 GHz operating point, PS-Sched-2 is 14.5% faster and has 11.4% energy savings more than the PS-Sched-1 approach compared with the default scheduler. However, the PS-Sched-1 method is superior at the 2.1 and 1.8 GHz operating points.

7.4. Energy-delay analysis

To compare the schedulers fairly, one must also take into account the speed of each scheduler in addition to the energy savings that can be obtained. To this end, the energy-delay analysis is presented in Figures 9–11 for the proposed schedulers normalized to the default scheduler. Energy-delay is a metric used to determine whether a trade-off between energy usage and the delay that is caused is beneficial or not. Energy-delay products of less than one indicate a beneficial trade-off, essentially showing that the savings in energy consumption outpace the corresponding increase in execution speed. Values greater than one represent the case in which the energy savings are not

DOI: 10.1002/cpe

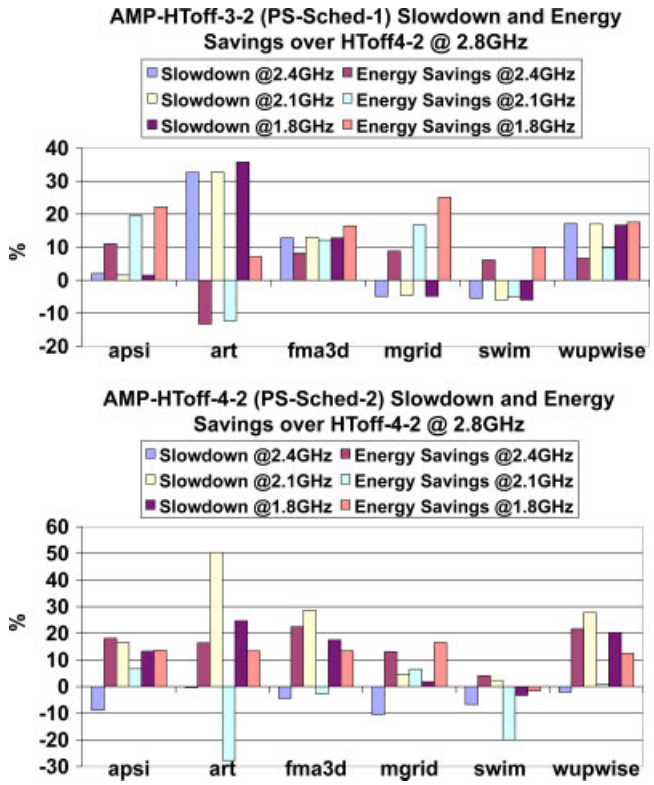


Figure 8. Slowdown and energy savings for AMP-HToff-3-2 and AMP-HToff-4-2 over HToff-4-2.

sufficiently offset by the increase in execution time. A downward trend in the graph from left to the right indicates that the rate of decrease of energy consumption and the rate of delay related to the said energy savings is diverging beneficially.

The energy-delay figures for AMP-HT_{on}-3-1 and the AMP-HT_{on}-4-1 configuration are shown in Figure 9. The AMP-HT_{on}-3-1 configuration has a majority of the applications with energy-delays of less than one for both the 2.4 and 2.1 GHz operating points, but sharply trends up at the 1.8 GHz operating point. Overall, its average energy-delay for the 2.4 GHz operating point is 0.75, while its energy-delay for the 1.8 GHz operating point is significantly above 1. The energy-delay of the AMP-HT_{on}-4-1 (PS-Sched-2) configuration in Figure 9 shows that all of the applications can benefit from using the method at the 2.1 GHz operating point, while approximately half can benefit at the 2.4 GHz operating point. When the results of the PS-Sched-1 method are compared with the PS-Sched-2 configuration, we can see that the first method is superior in terms of energy-delay, in that it has lower energy delays for the majority of benchmarks.

For the AMP-HT_{on}-7-2 (PS-Sched-1) configuration presented in Figure 10, the majority of applications do not see an energy-delay of less than one until the operating point is lowered to 2.1 GHz or lower. Overall, the trend of the graph is mostly flat with only a minor downtrend for the 2.1 GHz

DOI: 10.1002/cpe

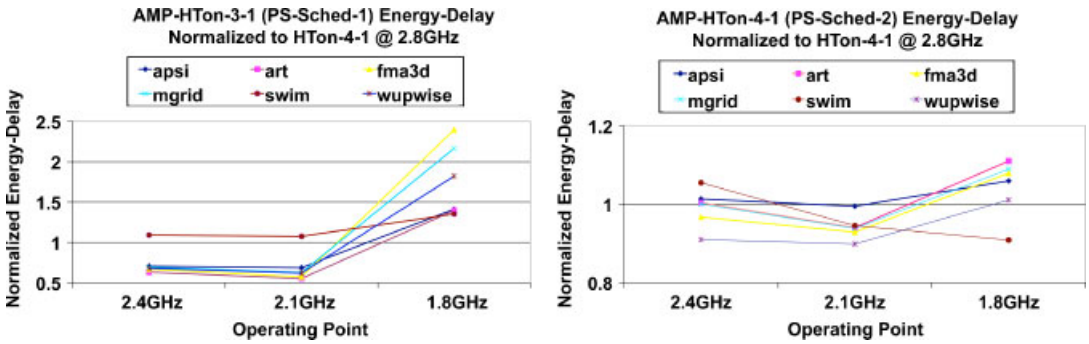


Figure 9. Normalized energy-delay for AMP-HTon-3-1 and AMP-HTon-4-1 over HTon-4-1.

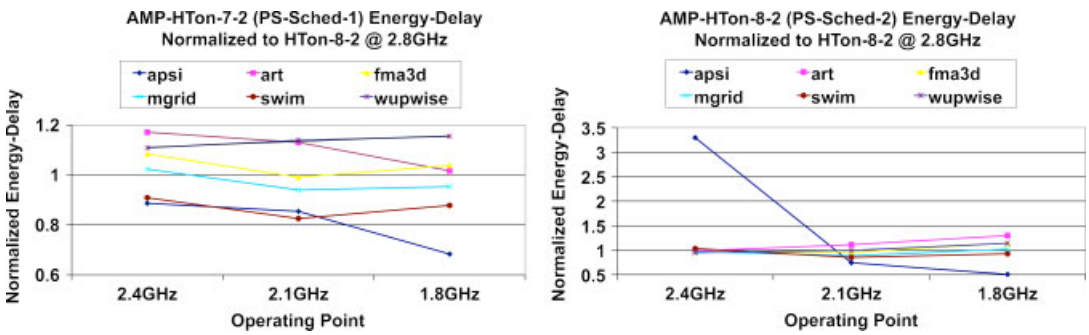


Figure 10. Normalized energy-delay for AMP-HTon-7-2 and AMP-HTon-8-2 over HTon-8-2.

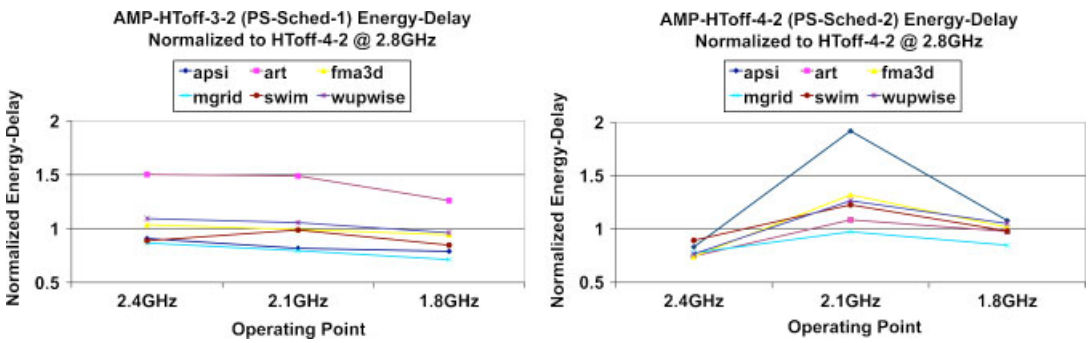


Figure 11. Normalized energy-delay for AMP-HToff-3-2 and AMP-HToff-4-2 over HToff-4-2.

operating point. Half of the applications see a benefit to using the PS-Sched-1. This leads us to conclude that the AMP-HT_{on}-7-2 configuration with the PS-Sched-1 method can be beneficial, but it is dependant on the type of applications that are being run.

DOI: 10.1002/cpe

The PS-Sched-2 approach shown in Figure 10 can be seen to be sufficient at the highest operating point for having marginal improvements to the performance and energy savings; that trend higher for lower operating points except for mgrid and swim, which improve slightly at the 2.1 GHz operating point before losing ground at 1.8 GHz. The apsi benchmark at the 2.4 GHz operating point is abnormally high for PS-Sched-2; this is most likely due to poor synchronization, resulting in the system spending a large proportion of time idle at this operating point. The synchronization of the threads obviously improves at the 2.1 and 1.8 GHz operating points as the performance is improved and the energy-delay products have decreased accordingly.

The energy-delay product for the AMP-HT_{on}-8-2 configuration with PS-Sched-2 in Figure 10 can be compared with the AMP-HT_{on}-7-2 configuration. At the higher operating points, the PS-Sched-1 is superior, but it should be noted that some of the applications that do not perform well using the PS-Sched-1 react well to the reduction in the operating frequency when using the PS-Sched-2, particularly the apsi benchmark.

We can see that the energy-delay of the PS-Sched-1 is mediocre for the AMP-HT_{off}-3-2 case in Figure 11, with no significant overall savings. Although art, mgrid, and swim benefit from using the scheduler, the majority of the applications do not. In fact, the system sees a general improvement of wall clock times but this is offset by higher power consumption. This leads to poor energy-delay products as the increase in speed happens at a slower rate than the increase in energy consumption.

When comparing the energy-delay results for the AMP-HT_{off}-4-2 configuration with PS-Sched-2 with the AMP-HT_{off}-3-2 results in Figure 11, we can observe that at the 2.4 GHz operating point, the PS-Sched-2 offers energy-delays less than 1 for all of the benchmarks. This clearly outperforms the AMP-HT_{off}-3-2 configuration with PS-Sched-1. Although the energy-delay products for the AMP-HT_{off}-4-2 configuration rise faster than PS-Sched-1 configuration for operating points after 2.1 GHz, the best energy-delay results for either configuration are for the PS-Sched-2 at 2.4 GHz, having an average energy-delay of 0.79.

8. CONCLUSIONS AND FUTURE RESEARCH

Chip multithreaded multi-core systems have become the mainstream in processor design. Such processors present new challenges as well as new opportunities for maximizing the performance, especially when a number of them are configured in an SMP fashion. Recently, growing power consumption has become an important design constraint in emerging hybrid CMT/SMP systems. This work explored the power-performance efficiency of HT asymmetric multiprocessors, and proposed two new scheduling algorithms to reduce overall energy consumption while having a minimal impact on the performance of multi-threaded OpenMP applications. The work presented considered scientific applications from the SPEC OMP benchmark suite on a range of system configurations on a real 2-way dual-core HT Intel Xeon SMP server, addressing both HT-enabled and HT-disabled AMPs.

This paper has shown that OS noise has an adverse effect on system performance. The main performance bottleneck for the hybrid CMT/SMP server studied was found to be the memory subsystem and it has been shown that the performance of the cache can be improved by isolating OS noise. Once OS noise is isolated, the runtime performance increase ranges from 1.4 to 17.2%, with an average performance gain of 7.2%.

DOI: 10.1002/cpe

The OS noise isolation also yielded the opportunity to increase overall energy efficiency by using an AMP. The first proposed power-saving scheduler uses a core for OS tasks only and scales its frequency in order to save power, while running user threads on the remaining cores at maximum frequency. In the second proposed method, the system has one core running at its full speed performing OS and user tasks while the remaining cores run user threads at lower operating frequencies.

Our results show that some system configurations benefit more from OS isolation than others. The system that sees the most benefit in terms of energy efficiency is the AMP-HT_{on}-3-1 configuration, with energy-delays between 0.6 and 0.8 on average for 2.4 and 2.1 GHz using the PS-Sched-1 approach. The second scheduling method was found to be useful for a subset of the benchmarking applications, and superior at higher operating frequencies for HT-disabled systems, having energy delays in the range of 0.7–0.8 for the 2.4 GHz operating point. The improvement that the second method provides over the first for HT-disabled systems could be a direct result of proper synchronization of execution threads with efficient handling of OS noise.

We showed that the PS-Sched-1 method can provide excellent performance for its best operating points, showing an average 14.4% speedup and 13.5% energy savings at 2.4 GHz, and an average speedup of 15.5% and energy savings of 21.4% at 2.1 GHz, for the AMP-HT_{on}-3-1 configuration running the SPEC benchmarks. The application benefiting the most from the PS-Sched-1 approach was the art benchmark at 2.1 GHz (AMP-HT_{on}-3-1) with a speedup of 18.2 and 32.5% energy savings over the default scheduler at 2.8 GHz. The performance of the PS-Sched-2 method for the 2.4 GHz operating point of the AMP-HT_{off}-4-2 configuration was shown to provide a 5.5% speedup and a 16.0% energy savings. Given these results we can conclude that the scheduling methods proposed can provide significant energy savings for HT-enabled and HT-disabled systems while providing increased performance. This increase in efficiency has also been seen to be effective for a large group of the studied applications.

This paper shows that energy savings are possible in a high performance server environment, and that with some changes to task scheduling, significant savings can be realized while maintaining the peak performance. Further energy savings could be realized by tuning the system for dynamic workload-based frequency and voltage scaling. This would involve tracking the overall workload and reporting back to the scheduler.

ACKNOWLEDGEMENTS

This work is supported by the Natural Sciences and Engineering Research Council of Canada through grant RGPIN/238964-2005, Canada Foundation for Innovation's grant #7154, and Ontario Innovation Trust's grant #7154.

REFERENCES

1. Spracklen L, Abraham SG. Chip multithreading: opportunities and challenges. *Proceedings of the 11th International Symposium on High-performance Computer Architecture* (*HPCA*), San Francisco, CA, 2005; 248–252. DOI: 10.1109/HPCA.2005.10.
2. Intel Corporation. *Intel Xeon Processor*, 2007.
3. Le HQ, Starke WJ, Fields JS, O'Connell FP, Nguyen DQ, Ronchetti BJ, Sauer WM, Schwarz EM, Vaden MT. IBM power6 microarchitecture. *IBM Journal of Research and Development* 2007; **51**(6):639–662.

4. McNairy C, Bhatia R. Montecito: A dual-core, dual-thread Itanium processor. *IEEE Micro* 2005; **25**(2):10–20. DOI: 10.1109/MM.2005.34.
5. Sun Microsystems. *Sun UltraSPARC T2*. Available at: http://www.sun.com/processors/UltraSPARC-T2 [2 September 2007].
6. Hammond L, Nayfeh BA, Olukotun KA. A single-chip multiprocessor. *IEEE Computer* 1997; **30**(9):79–85. DOI: 10.1109/2.612253.
7. Tullsen D, Eggers S, Levy H. Simultaneous multithreading: maximizing on-chip parallelism. *Proceedings of the 22nd Annual International Symposium on Computer Architecture* (*ISCA*), Santa Margherita Ligure, Italy, 1995; 392–403.
8. Marr DT, Binns F, Hill DL, Hinton G, Koufaty DA, Miller JA, Upton M. Hyper–Threading technology architecture and microarchitecture. *Intel Technology Journal* 2002; **6**(1):1–12.
9. McGregor RL, Antonopoulos CD, Nikolopoulos DS. Scheduling algorithms for effective thread pairing on hybrid multiprocessors. *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium* (*IPDPS*), Denver, CO, 2005. DOI: 10.1109/IPDPS.2005.390.
10. Liao C, Liu Z, Huang L, Chapman B. Evaluating OpenMP on chip multithreading platforms. *Proceedings of the 1st International Workshop on OpenMP* (*IWOMP*) (*Lecture Notes in Computer Science*, vol. 4315). Springer: Berlin, 2005; 178–190. DOI: 10.1007/978-3-540-68555-5_15.
11. Grant RE, Afsahi A. A comprehensive analysis of multithreaded OpenMP applications on dual-core Intel Xeon SMPs. *2007 Workshop on Multithreaded Architectures and Applications* (*MTAAP*), *Proceedings of the 21st International Parallel and Distributed Processing Symposium* (*IPDPS*), Long Beach, CA, 2007. DOI: 10.1109/IPDPS.2007.370682.
12. Grant RE, Afsahi A. Characterization of multithreaded scientific workloads on simultaneous multithreading Intel processors. *Proceedings of the 2005 Workshop on Interaction between Operating System and Computer Architecture* (*IOSCA*), Austin, TX, 2005.
13. Zhang Y, Burcea M, Cheng V, Ho R, Voss M. An adaptive OpenMP loop scheduler for Hyperthreaded SMPs. *Proceedings of the 16th ISCA International Conference on Parallel and Distributed Computing Systems* (*PDCS*), San Francisco, CA, 2004; 256–263.
14. OpenMP Architecture Review Board, *OpenMP Specification Version 3.0*, May 2008.
15. Annavaram M, Grochowski E, Shen J. Mitigating Amdahl's law through EPI throttling. *Proceedings of the 32nd Annual International Symposium on Computer Architecture* (*ISCA*), Madison, CI, 2005; 298–309. DOI: 10.1109/ISCA.2005.36.
16. Balakrishnan S, Rajwar R, Upton M, Lai K. The impact of performance asymmetry in emerging multicore architectures. *Proceedings of the 32nd Annual International Symposium on Computer Architecture* (*ISCA*), Madison, CI, 2005; 506–517. DOI: 10.1109/ISCA.2005.51.
17. Kumar R, Farkas K, Jouppi NP, Ranganathan P, Tullsen DM. Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction. *Proceedings of the 36th International Symposium on Microarchitecture* (*MICRO*), San Diego, CA, 2003; 81–92. DOI: 10.1109/MICRO.2003.1253185.
18. Petrini F, Kerbyson DJ, Pakin S. The case of missing supercomputer performance: achieving optimal performance on the 8192 processors of ASCI Q. *Proceedings of the 2003 ACM/IEEE Supercomputing Conference* (*SC*), Phoenix, AZ, 2003. DOI: 10.1109/SC.2003.10010.
19. Jones T, Dawson S, Neely R, Tuel W, Brenner L, Fier J, Blackmore R, Caffrey P, Maskell B, Tomlinson P, Roberts M. Improving the scalability of parallel jobs by adding parallel awareness to the operating system. *Proceedings of the 2003 ACM/IEEE Supercomputing Conference* (*SC*), Phoenix, AZ, 2003. DOI: 10.1109/SC.2003.10024.
20. Tsafrir D, Etsion Y, Feitelson DG, Kirkpatrick S. System noise, OS clock ticks, and fine-grained parallel applications. *Proceedings of the 19th Annual International Conference on Supercomputing* (*ICS*), Cambridge, MA, 2005; 303–312. DOI: 10.1145/1088149.1088190.
21. Grant RE, Afsahi A. Power-performance efficiency of asymmetric multiprocessors for multi-threaded scientific applications. *2006 Workshop on High-performance*, *Power-aware Computing* (*HP-PAC*), *Proceedings of the 20th International Parallel and Distributed Processing Symposium* (*IPDPS*), Rhodes Island, Greece, 2006. DOI: 10.1109/IPDPS.2006.1639601.
22. Grant RE, Afsahi A. Improving system efficiency through scheduling and power management. *2007 International Workshop on Green Computing* (*GreenCom*), invited paper, work-in-progress session. *Proceedings of the 9th IEEE International Conference on Cluster Computing* (*Cluster*), Austin, TX, 2007; 478–479. DOI: 10.1109/CLUSTR.2007.4629271.
23. SPEC OMP Benchmark Suite. Available at: http://www.spec.org/omp [10 June 2006].
24. Mudge T. Power: a first class design constraint. *IEEE Computer* 2001; **34**(4):52–57. DOI: 10.1109/2.917539.
25. Albonesi DH, Balasubramonian R, Dropsbo SG, Dwarkadas S, Friedman FG, Huang MC, Kursun V, Magklis G, Scott ML, Semeraro G, Bose P, Buyuktosunoglu A, Cook PW, Schuster SE. Dynamically tuning resources with adaptive processing. *IEEE Computer* 2003; **36**(12):49–58. DOI: 10.1109/MC.2003.1250883.
26. Li Y, Brooks D, Hu Z, Skadron K. Performance, energy, and thermal considerations for SMT and CMP architectures. *Proceedings of the 11th International Symposium on High-performance Computer Architecture* (*HPCA*), San Francisco, CA, 2005; 71–82. DOI: 10.1109/HPCA.2005.25.

27. Sasanka R, Adve S, Chen Y, Debes E. The energy efficiency of CMP vs. SMT for multimedia workloads. *Proceedings of the 18th Annual International Conference on Supercomputing* (*ICS*), Saint-Halo, France, 2004; 196–206. DOI: 10.1145/1006209.1006238.
28. Kumar R, Tullsen DM, Joupi NP, Ranganathan P. Heterogeneous chip multiprocessors. *IEEE Computer* 2005; **38**(11): 32–38. DOI: 10.1109/MC.2005.379.
29. Pham D, Asano S, Bolliger M, Day MN, Hofstee HP, Johns C, Kahle J, Kameyama A, Keaty J, Masubuchi Y, Riley M, Shippy D, Stasiak D, Suzuoki M, Wang M, Warnock J, Weitzel S, Wendel D, Yamazaki T, Yazawa K. The design and implementation of a first-generation cell processor. *Proceedings of the 2005 International Symposium on Solid-state Circuits and Systems* (*ISSCC*), San Francisco, CA, 2005; 184–186. DOI: 10.1109/ISSCC.2005.1493930.
30. De Vuyst M, Kumar R, Tullsen DM. Exploiting unbalanced thread scheduling for energy and performance on a CMP of SMT processors. *Proceedings of the 20th International Parallel and Distributed Processing Symposium* (*IPDPS*), Rhodes Island, Greece, 2006. DOI: 10.1109/IPDPS.2006.1639374.
31. El-Moursy A, Garg R, Albonesi DH, Dwarkadas S. Compatible phase co-scheduling on a CMP of multi-threaded processors. *Proceedings of the 20th International Parallel and Distributed Processing Symposium* (*IPDPS*), Rhodes Island, Greece, 2006. DOI: 10.1109/IPDPS.2006.1639376.
32. Flautner K, Reinhardt S, Mudge T. Automatic performance-setting for dynamic voltage scaling. *Journal of Wireless Networks* 2002; **8**(5):507–520. DOI: 10.1023/A:1016546330128.
33. Zeng H, Fan X, Ellis C, Lebeck A, Vahdat A. ECOSystem: managing energy as a first class operating system resource. *Proceedings of the 10th International Conference on Architectural Support of Programming Languages and Operating Systems* (*ASPLOS*), San Diego, CA, 2002; 123–132. DOI: 10.1145/605432.605411.
34. Curtis-Maury M, Blagojevic F, Antonopoulos CD, Nikolopoulos DS. Prediction-based power-performance adaptation of multithreaded scientific codes. *IEEE Transactions on Parallel and Distributed Systems* 2008; **19**(8):1396–1410. DOI: 10.1109/TPDS.2007.70804.
35. Freeh VW, Pan F, Lowenthal DK, Kappiah N, Springer R, Rountree BL, Femal ME. Analyzing the energy-time tradeoff in high-performance computing applications. *IEEE Transactions on Parallel and Distributed Systems* 2007; **18**(6):835–848. DOI: 0.1109/TPDS.2007.1026.
36. Ge R, Feng X, Cameron K. Improvement of power-performance efficiency for high-end computing. *2005 Workshop on High-performance*, *Power-aware Computing* (*HP-PAC*), *Proceedings of the 19th International Parallel and Distributed Processing Symposium* (*IPDPS*), Denver, CO, 2005. DOI: 0.1109/IPDPS.2005.251.
37. Hsu C-H, Feng W-C. A power-aware run-time system for high-performance computing. *Proceedings of the 2005 ACM/IEEE Supercomputing Conference*, (*SC*), Seattle, WA, 2005. DOI: 10.1109/SC.2005.3.
38. Message Passing Interface Forum: MPI, A Message Passing Interface standard. *Version 1.2*, 1997.
39. Hsu C-H, Kremer U. The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction. *Proceedings of ACM SIGPLAN 2003 Conference on Programming Languages*, *Design*, *and Implementation* (*PLDI*), San Diego, CA, 2003; 38–48. DOI: 10.1145/781131.781137.
40. Kotla R, Ghiasi S, Keller K, Rawson F. Scheduling processor voltage and frequency in servers and cluster systems. *2006 Workshop on High-performance*, *Power-aware Computing* (*HP-PAC*), *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium* (*IPDPS*), Denver, CO, 2005. DOI: 10.1109/IPDPS.2005.392.
41. Bianchini R, Rajamony R. Power and energy management for server systems. *IEEE Computer* 2007; **37**(11):68–74. DOI: 10.1109/MC.2004.217.
42. Brosky S. Shielded CPUs: real-time performance in standard Linux. *Linux Journal*. 2004; **2004**(121). Available at: http://www.linuxjournal.com/article/6900 [1 May 2004].
43. Mogul JC, Mudigonda J, Binkert N, Ranganathan R, Talwar V. Using asymmetric single-ISA CMPs to save energy on operating systems. *IEEE Micro* 2008; **28**(3):26–41. DOI: 10.1109/MM.2008.47.
44. McVoy LW, Staelin C. Lmbench: Portable tools for performance analysis. *Proceedings of USENIX 1996 Annual Technical Conference*, San Diego, CA, 1996; 279–294.
45. Oprofile. Available at: http://oprofile.sourceforge.net [17 July 2007].