Improving System Efficiency through Scheduling and Power Management

Ryan E. Grant[1], Ahmad Afsahi[2]

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

[1]ryan.grant@ece.queensu.ca
[2]ahmad.afsahi@queensu.ca

I. Introduction

The performance of the emerging commercial chip multi-threaded multiprocessors 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 could be effectively used to increase overall system power efficiency while sustaining performance are very desirable. In essence, researchers have recently proposed various mechanisms to achieve this goal for multi-threaded applications [2, 3, 4].

Previous research has shown that system noise may have a dramatic effect on memory hierarchy and consequently on performance [3, 5]. The effect will be more pronounced with the introduction of larger chip multiprocessors. The competition for resources between highly CPU intensive tasks and the operating system yield opportunities to increase overall system efficiency. Effectively handling the smaller operating system tasks while simultaneously preserving application thread synchronicity, can lead to gains in overall efficiency.

II. Proposed Schedulers

The first method of thread management, originally proposed in [4], masks off a single logical/physical core for operating system tasks only and scales its frequency in order to save power, while running user threads on the remaining cores at maximum frequency. The difference in frequency for cores resembles an *asymmetric multiprocessor* (AMP) [1]. In the second method proposed here, the system has one core in the system running at its full clock speed performing OS and 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.

III. Experimental Results

We present the results on a two-way dual-core SMT-capable Intel Xeon system. The results are differentiated by a naming scheme made up of three parts. The first part indicates if *Hyper-Threading* (SMT) is enabled or not, the second part indicates the number of threads that the system is running and the third part shows the number of physically independent chips that the threads are being executed on.

The slowdown and resultant energy savings for both methods for a select group of architectures is shown in Figure 1. The results for the average improvement for each architecture at their best operating point are detailed in Table 1, with energy savings based on real power measurements. Although method 2 performs better on average for the non-SMT architectures and at some operating points for SMTs, method 1 is superior in terms of overall average slowdown and energy savings for the SMT architectures.

TABLE I
IMPROVEMENTS OF METHOD 2 OVER METHOD 1

Architecture	Frequency	Speedup	Energy Savings
AMP-HT$_{on}$-4-1	1.8GHz	22.6%	39.5%
AMP-HT$_{on}$-8-2	2.1GHz	1.2%	6.4%
AMP-HT$_{off}$-4-2	2.4GHz	14.6%	11.4%

The AMP normalized energy-delay results for each of the configurations with each method are shown in Figure 2. We see the same trend in energy-delay products as was observed for Figure 1; that method 1 is better on average for SMT-enabled systems while method 2 is better on average for non-SMT systems.

The improvement that the second method provides over the first could be a direct result of proper synchronization of execution threads with efficient handling of operating system noise. Of course, the speed of the cores is playing an essential role in the performance difference between the two methods. This leads us to the conclusion that the methods studied are effective, but must be carefully managed depending on the system load, operating system noise and real power consumption of the system in any given phase.

References

[1] M. Annavaram, E. Grochowski, and J. Shen, "Mitigating Amdahl's Law through EPI throttling," 3*2nd Annual International Symposium on Computer Architecture (ISCA'05),* 2005, pp. 298-309.

[2] M. Curtis-Maury, J. Dzierwa, D. Antonopoulos and D. S. Nikolopoulos, "Online strategies for high-performance power-aware thread execution on emerging multiprocessors," *2nd Workshop on High-Performance, Power-Aware Computing (HP-PAC '06), 2006.*

[3] R. E. Grant and A. Afsahi, "Improving performance and power consumption of chip multi-threading symmetric multiprocessors," submitted.

[4] R. E. Grant and A. Afsahi, "Power-performance efficiency of asymmetric multiprocessors for multi-threaded scientific applications," *2nd Workshop on High-Performance, Power-Aware Computing (HP-PAC '06),* 2006.

[5] D. Tsafrir, Y. Etsion, D. G. Feitelson and S. Kirkpatrick, "System noise, OS clock ticks, and fine-grained parallel applications," *19th Annual International Conference on* Supercomputing (ICS'05), 2005, pp. 303-312.

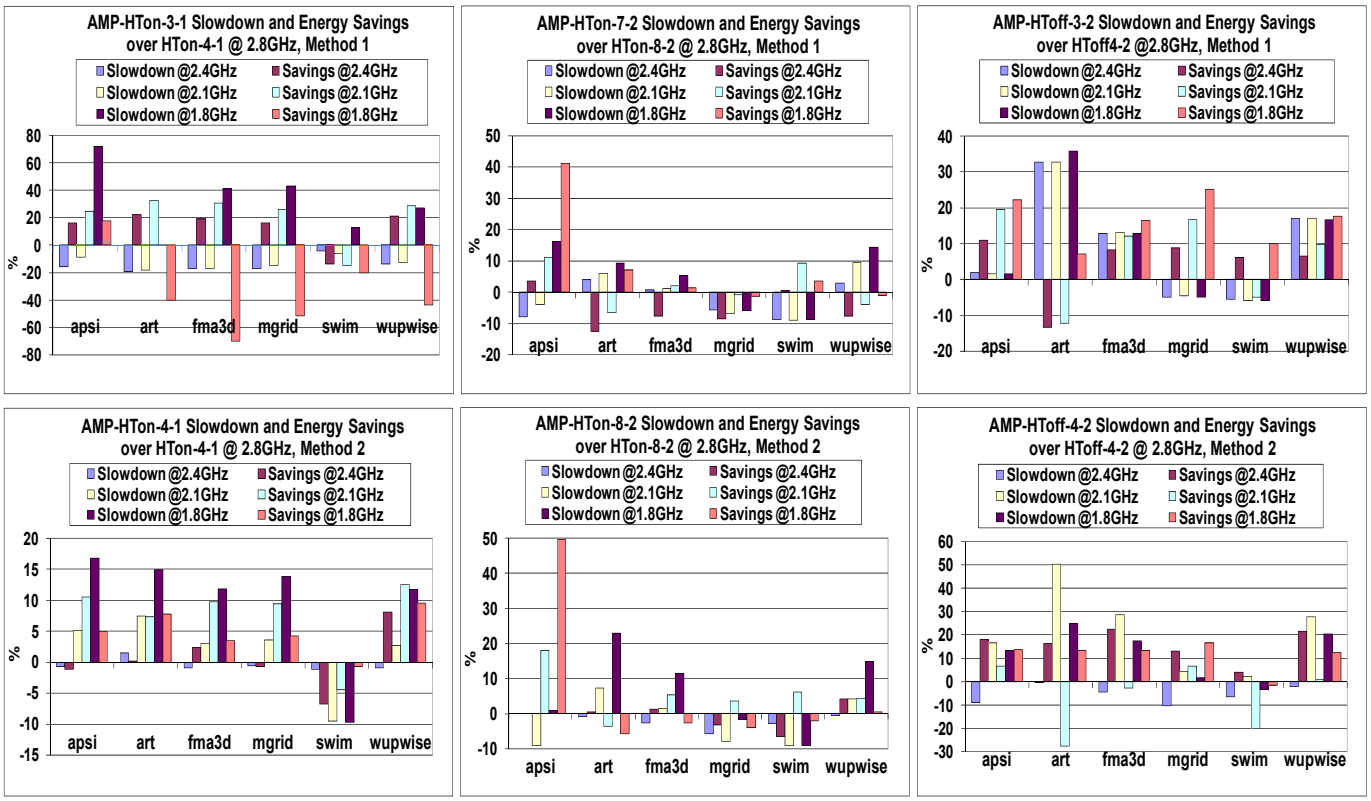

Figure 1. AMP slowdown and energy savings with the proposed schedulers for SPEC benchmarks

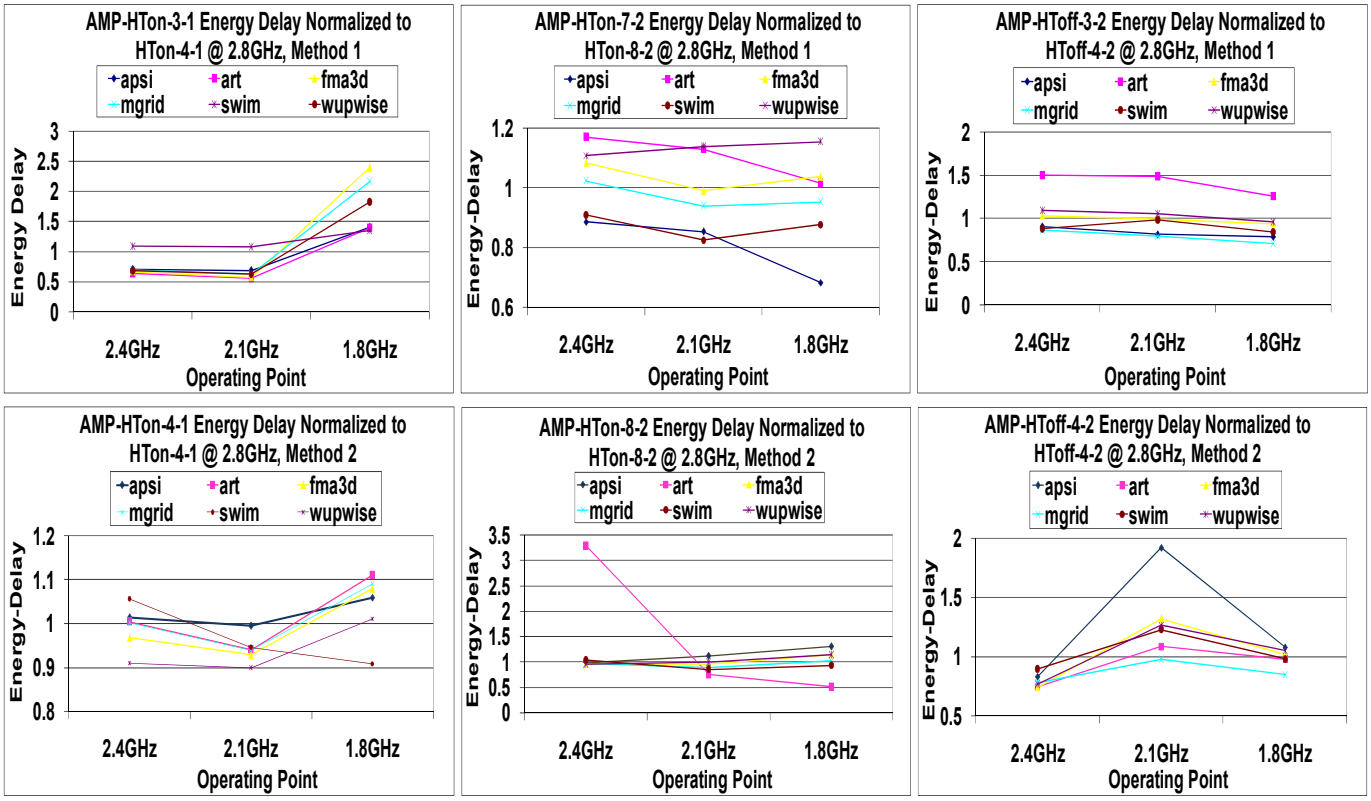

Figure 2. AMP normalized Energy-Delay with the proposed schedulers for SPEC benchmarks