Simultaneous Multithreading on x86_64 Systems: An Energy Efficiency Evaluation

Robert Schöne Daniel Hackenberg Daniel Molka
Center for Information Services and High Performance Computing (ZIH)
Technische Universität Dresden, 01062 Dresden, Germany
{robert.schoene, daniel.hackenberg, daniel.molka}@tu-dresden.de

ABSTRACT

In recent years, power consumption has become one of the most important design criteria for microprocessors. CPUs are therefore no longer developed with a narrow focus on raw compute performance. This means that well-established processor features that have proven to increase compute performance now need to be re-evaluated with a new focus on energy efficiency. This paper presents an energy efficiency evaluation of the symmetric multithreading (SMT) feature on state-of-the-art x86_64 processors. We use a mature power measurement methodology to analyze highly sophisticated low-level microbenchmarks as well as a diverse set of application benchmarks. Our results show that–depending on the workload–SMT can be at the same time advantageous in terms of performance and disadvantageous in terms of energy efficiency. Moreover, we demonstrate how the SMT efficiency has advanced between two processor generations.

1. INTRODUCTION

Hardware multithreading is an established technique to increase processor throughput by handling multiple threads in parallel. It can be distinguished into temporal multithreading (TMT) and simultaneous multithreading (SMT). While TMT processors can only issue instructions from a single thread in one cycle, SMT processors are able to issue instructions from multiple threads concurrently. The first SMT-capable system [2] was installed in 1988. Intel added simultaneous multithreading to the popular x86 family with the Netburst microarchitecture (e.g. Pentium 4) [6]. The brand name for this technology is Hyper-Threading (HT) [8]. It provides two threads (logical processors) per core and still is available in many Intel processors.

The primary objective of multithreading is to improve the throughput of pipelined and superscalar architectures. Working on multiple threads provides more independent instructions and thereby increases the utilization of the hardware as long as the threads do not compete for the same crucial resource (e.g., bandwidth). However, MT may also decrease the performance of a system due to contention for shared hardware resources like caches, TLBs, and re-order buffer entries. Harmful effects such as cache and TLB thrashing can be the result.

Many scientific studies have investigated the effectiveness of SMT in general and HT in particular. In this paper we re-evaluate this technique with an new focus: Does HT increase the power consumption of the processor, and if so, does this outweigh the performance advantage so that the overall energy requirement to carry out a given task increases? The question is even more difficult to answer when considering the fact that the answer strongly depends on the type of workload that is executed. This paper contributes an energy efficiency study of Hyper-Threading using a mature power measurement methodology and two state-of-the-art x86_64 Intel microarchitectures (Westmere-EP and Sandy Bridge). We analyze the impact of HT on the power consumption when running synthetic benchmarks as well as a rich set of application benchmarks from the SPEC CPU and OMP benchmark suites.

2. RELATED WORK

Ungerer et al. describe the different types of hardware multithreading that are implemented in processors along with their advantages and shortcom-

HotPower '11, October 23, 2011, Cascais, Portugal.

ings [13]. Tullsen et al. focus on simultaneous multithreading (SMT) and illustrate the substantially increased complexity of the processor design [12]. Hyper-Threading is the most common type of SMT and has been described in [8, 6]. A recent performance evaluation of Nehalem cluster has demonstrated that the performance improvement of HT is highly application dependent [11]. Li et al. [7] model the effects of SMT on power consumption for POWER4-like architectures. They conclude that the IBM implementation of SMT can significantly improve energy efficiency.

A set of synthetic low-level benchmarks that allows to determine the performance of data transfers within shared memory x86_64 systems has been presented in[9, 4]. They have been further extended to determine the energy consumption of accesses to different levels in the memory hierarchy and performing arithmetic operations on two-socket AMD and Intel systems [10]. This study has shown first signs that the use of Hyper-Threading introduces a significant power consumption overhead while not providing any performance advantage. In this paper we build upon the results of [10] in order to investigate how the more recent Sandy Bridge microarchitecture performs and how this effect translates to real applications as represented by the benchmarks SPEC OMP [1] and SPEC CPU2006 [5].

3. EXPERIMENTAL SETUP

3.1 Test System Hardware

We use two state-of-the-art x86_64 test systems to evaluate the energy efficiency of the multithreading implementation (HT) of two different generations of Intel processors. One test system is based on the most current dual socket Intel Xeon processors (Westmere-EP). This processor features six cores with SSE units, private L1 and L2 caches and a shared L3 cache. The second test system is based on the most recent Intel processor family (Sandy Bridge)[1]. This processor features four cores with AVX units and a similar memory subsystem. Table 1 lists the test system configuration in detail.

Both of the test systems are highly power optimized with only indispensable components installed. The power supplies of both systems are very efficient and the Dell system additionally features low-power DRAM modules. Due to the unavailability of dual socket Sandy Bridge processors, we compare a single socket Sandy Bridge system with a dual socket Westmere-EP machine. In order to ensure a fair comparison, we do not look at absolute power consumption numbers, but only relative changes, i.e. the percentage difference of the power and energy consumption when running a workload with SMT enabled or disabled. Moreover, we compare workloads that–while using all available cores of the system–perform almost no communication between the individual threads. Finally, we ensure that memory allocation of each thread occurs locally, so that no significant inter-socket communication occurs on the dual socket machine.

[1]This is in fact a desktop class CPU (Core i7). However, almost identical results were obtained on a preproduction Intel Xeon 1280 test system. The Xeon system showed some stability issues and we therefore present the Core i7 results.

Table 1: Hardware Configuration

System	Dell PowerEdge R510	Fujitsu ESPRIMO P700 E85+
Processor	2x Xeon X5670	1x Core i7 2600
Core clock	2.93 GHz (Turbo 3.33 GHz)	3.4 GHz (Turbo 3.8 GHz)
Uncore clock	2.66 GHz	3.4 GHz (Turbo 3.8)
TDP	95 W	95 W
Codename	Westmere-EP	Sandy Bridge
FPU	2x 128 Bit (SSE)	2x 256 Bit (AVX)
L1 cache	2x 32 KiB per core	2x 32 KiB per core
L2 cache	256 KiB per core	256 KiB per core
L3 cache	12 MiB per chip	8 MiB per chip
IMC channels	3x RDDR3	2x DDR3
Memory type	PC3L-10600R	PC3-10600
Memory size	12 GiB (6x 2 GiB)	8 GiB (2x 4 GiB)
Chipset	Intel 5520	Intel Q65
Power supply	Dell 480W PN H410J	Fujitsu DPS-250AB-62 A
OS	Linux 2.6.38	Linux 2.6.38

3.2 Software Workloads

The highly optimized microbenchmarks [9, 4] allow us to determine the data throughput of different instruction types that access well-defined locations within the memory hierarchy. Either n (on an n-core system with HT disabled) or 2n (HT enabled) threads perform a uniform workload consisting of independent operations that on disjoint memory regions. The innermost benchmark routines are written in highly optimized assembly code that achieves near peak bandwidth for every cache level as well as main memory. One thread per core is generally sufficient to fully utilize the data paths, leaving little to no potential for performance increases through the use of HT. With HT enabled, the execution units and caches are concurrently stressed by two threads per core that both achieve roughly half of the peak per-core throughput. The existing bench-

marks have been extended to use AVX instructions on the Sandy Bridge platform in order to fully utilize the 256 Bit wide execution units and data paths.

We also run a subset of the SPEC CPU2006 application benchmarks on our test systems. Officially submitted SPEC CPU2006 results from HT-capable systems are often generated with HT enabled to improve the overall benchmark performance. In our study we focus on the integer part of SPEC CPU2006 as we found these benchmarks to be more sensitive to the use of HT. We focus on the throughput mode of the benchmark (*rate* instead of *speed*) that runs multiple copies of the same serial workload on all available cores. To avoid unnecessary complexity, we use the *base* configuration that allows for just one set of compiler flags for all benchmarks instead of individually tuned configurations (*peak*). We omit benchmarks that do not run on our Westmere-EP test system due to an excessive per-core memory footprint (400, 401, 429). Our application study is complemented with measurements of the SPEC OMP benchmark suite that includes floating-point intense, OpenMP parallelized workloads from different fields of High Performance Computing. However, these benchmarks are not suitable for our single-socket four-core test system, as no official submission is available to serve as a performance reference. Our compiler setup for all SPEC benchmarks is listed in Table 2. We run three iterations of each benchmark and present the average of all three runs in Section 4.2. Similar to previously submitted SPEC results we enable the Turbo Boost for the SPEC CPU2006 benchmarks, but not for SPEC OMP.

Table 2: Compiler Configuration

	Westmere-EP	Sandy Bridge
Compiler Suite	Intel Composer XE 12.0	
Compiler Flags SPEC CPU	`-m32 -xSSE4.2 -ipo -O3 -no-prec-div -static -opt-prefetch`	`-m32 -xAVX -ipo -O3 -no-prec-div -static -opt-prefetch`
Compiler Flags SPEC OMP	`-xSSE4.2 -O3 -ipo1 -openmp`	

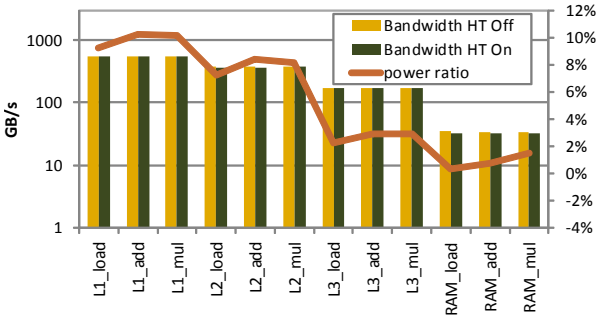
3.3 Power Measurement

For both the synthetic and the application benchmarks we measure the power consumption using a ZES Zimmer LMG450 power analyzer attached to the power supply of each test system. This device provides highly accurate power consumption data with a sampling frequency of 20 Hz. The power consumption data is collected by dedicated hardware and software components [10]. Using the benchmark runtime and the power consumption data (in Watts) we then calculate the overall energy consumption of the workload. The power and energy data is merged with the standard benchmark results in a post mortem step, thereby effectively eliminating any measurement overhead.

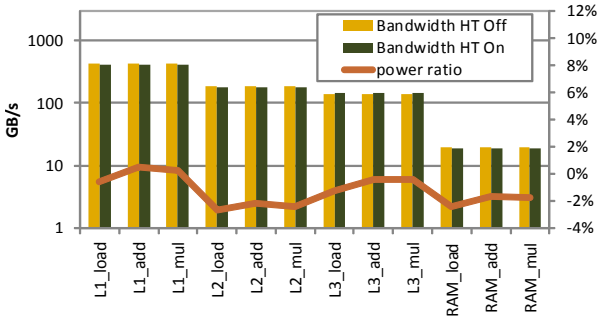
4. RESULTS

4.1 Synthetic Benchmarks

Figure 2 illustrates the power and performance impact of Hyper-Threading on low-level benchmark routines that execute either `load` or `add` or `mul` instructions to access different levels of the memory hierarchy. While the accumulated throughput of all benchmark threads is only impacted slightly,

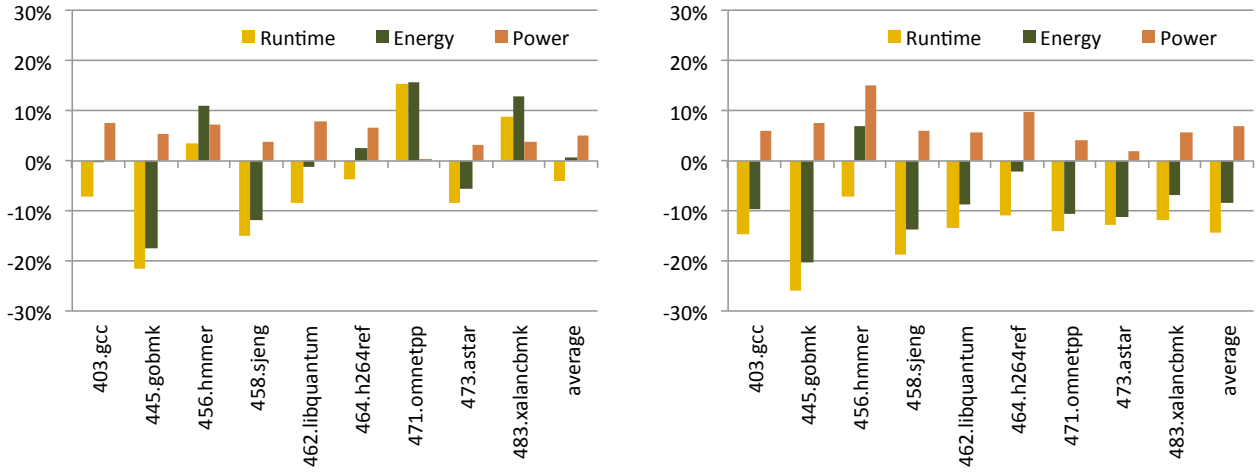


(a) Dual Socket Intel Westmere-EP



(b) Single Socket Intel Sandy Bridge

Figure 1: Absolute data throughput and relative power consumption of different SSE/AVX packed double assembly instructions with HT disabled (1 thread per core) and enabled (2 threads per core). While the accumulated data throughput is mostly independent of the number of threads per core, the power consumption can vary significantly. On the Westmere-EP system, the power consumption of a workload that stresses exclusively the L1 cache can increase by up to 10 % when using two threads per core without providing any performance advantage. This effect has been effectively eliminated in the subsequent Sandy Bridge design.



(a) Dual Socket Intel Westmere-EP

(b) Single Socket Intel Sandy Bridge

Figure 2: Runtime, power and energy consumption of selected SPECint_rate_base2006 benchmarks for the two test systems. Running two threads per core (HT enabled) consistently increases the power consumption. Three benchmarks run less energy efficient with HT enabled on Westmere-EP. For 462.h264ref the increased power consumption outweighs the runtime advantage, thus decreasing energy efficiency. While two benchmarks show increased runtimes with HT enabled on Westmere-EP, all benchmark runtimes decrease on the Sandy Bridge test system. 456.hmmer is the one exception that still shows decreased efficiency with HT enabled.

the additional power demand in the HT enabled case on the Westmere-EP system is evident, reaching up to 10%. In previous work on a different dual socket test system we have measured a 40-Watts premium for some SSE2 packed integer operations without any notable performance advantage. The extent to which the power consumption increases is highly proportional with the overall data throughput. In contrast to Westmere or Nehalem processors, the new Sandy Bridge architecture does not show this behavior at all.This demonstrates a considerable improvement of the power efficiency of Intel's SMT implementation. The interesting question now is whether and how this advancement translates from synthetic benchmarks to an improved energy efficiency of real applications.

4.2 Application Benchmarks

We use published SPEC results in order to find a competitive set of compiler flags.[2] Due to the missing *smartheap* library that is known to strongly influence C++ results, our C++ benchmark runs show noticeably lower performance. Most of our other CPU2006 benchmarks runtimes are in line with published results on both test platforms. The SPECint_rate_base2006 results depicted in Figure 2 show a consistent increase in power consumption with HT enabled. A performance penalty with HT enabled only occurs on the Westmere-EP platform. For minor runtime benefits the increased power consumption can worsen the overall energy efficiency in terms of Joule per workload. This is the case for 464.h264ref on Westmere-EP and for 456.hmmer on Sandy Bridge. A comparison of the runtime and energy bars between both platform clearly shows that the HT implementation in the Sandy Bridge microarchitecture has been strongly improved in terms of both performance and energy efficiency.

The SPEC OMP benchmark characteristics differ significantly from CPU2006, as the participating tasks actually synchronize with each other and scalability therefore is an issue. The scalability of 314.mgrid, 318.galgel, and 320.equake is known to be poor [3], which explains the HT performance penalty shown in Figure 3. Additionally to the runtime increase, HT causes a power penalty that makes the HT disadvantage even worse in terms of energy efficiency. Benchmarks that do not scale poorly and are less memory bound than 312.swim typically benefit from Hyper-Threading. However, 330.art as well as the overall average show again that a minor performance advantage can come with a bigger energy efficiency disadvantage.

[2] `http://www.spec.org/cpu2006/results/res2011q2/cpu2006-20110329-15360.html`
`http://www.spec.org/cpu2006/results/res2010q1/cpu2006-20100315-09799.html`

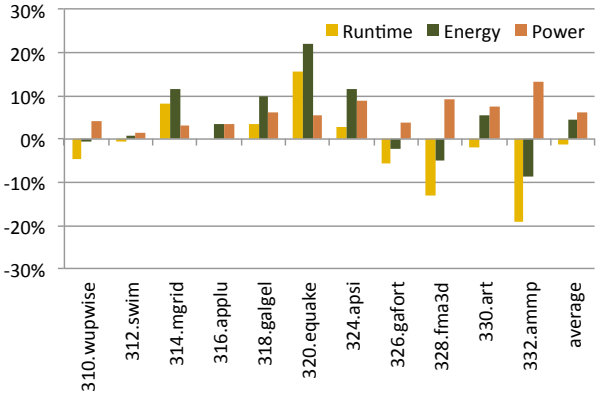


Figure 3: Runtime, power consumption and energy consumption of SPEC OMP2001M benchmarks for the Westmere-EP test system. Running two threads per core consistently increases the power consumption. The energy efficiency typically decreases.

5. CONCLUSION

This paper presents an in-depth study of SMT on current x86_64 Intel processors. Using a sophisticated power measurement methodology we extend the traditional performance analysis by including important energy efficiency aspects. A set of synthetic low-level microbenchmarks is used to demonstrate how the use of SMT increases the power consumption of a Westmere-EP compute node by up to 10% even though the data throughput remains unchanged. We also show that this deficiency of Intel's SMT implementation has been effectively removed in the latest Sandy Bridge microarchitecture.

Compared to earlier processors, the newer SMT implementation also provides more significant performance gains when running application benchmarks. Although the use of SMT still increases the power consumption consistently for all workloads, the performance gains typically outweigh the increased power consumption. For parallel programs that do not scale with the number of parallel tasks (e.g. some of the SPEC OMP workloads), using SMT may increase both runtime and power consumption at the same time. Although our study shows significant advancements of Intel's SMT implementation, the use of SMT still needs to be carefully considered–preferably not only with the runtime but also the power consumption in mind.

Acknowledgment This work has been funded by the Bundesministerium für Bildung und Forschung via the Spitzencluster CoolSilicon (BMBF 13N10186) and the eeClust research project (01IH08008C). We thank Intel Germany for providing us with an Intel Xeon E3-1280 evaluation system.

6. REFERENCES

[1] Vishal Aslot, Max J. Domeika, Rudolf Eigenmann, Greg Gaertner, Wesley B. Jones, and Bodo Parady. Specomp: A new benchmark suite for measuring parallel computer performance. In *Proceedings of the International Workshop on OpenMP Applications and Tools: OpenMP Shared Memory Parallel Programming*, WOMPAT '01, pages 1–10, London, UK, 2001. Springer-Verlag.

[2] Mikhail N. Dorozhevets and Peter Wolcott. The el'brus-3 and mars-m: Recent advances in russian high-performance computing. *The Journal of Supercomputing*, 6:5–48, 1992. 10.1007/BF00128641.

[3] Karl Fürlinger, Michael Gerndt, and Jack Dongarra. Scalability analysis of the SPEC OpenMP benchmarks on large-scale shared memory multiprocessors. In Yong Shi, Geert van Albada, Jack Dongarra, and Peter Sloot, editors, *Computational Science – ICCS 2007*, volume 4488 of *Lecture Notes in Computer Science*, pages 815–822. Springer Berlin / Heidelberg, 2007.

[4] Daniel Hackenberg, Daniel Molka, and Wolfgang E. Nagel. Comparing cache architectures and coherency protocols on x86-64 multicore smp systems. In *MICRO 42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 413–422, New York, NY, USA, 2009. ACM.

[5] John L. Henning. Spec cpu2006 benchmark descriptions. *SIGARCH Comput. Archit. News*, 34:1–17, September 2006.

[6] David Koufaty and Deborah T. Marr. Hyperthreading technology in the netburst microarchitecture. *IEEE Micro*, 23:56–65, March 2003.

[7] Yingmin Li, David Brooks, Zhigang Hu, Kevin Skadron, and Pradip Bose. Understanding the energy efficiency of simultaneous multithreading. In *Proceedings of the 2004 international symposium on Low power electronics and design*, ISLPED '04, pages 44–49, New York, NY, USA, 2004. ACM.

[8] Deborah T. Marr, Frank Binns, David L. Hill, Glenn Hinton, David A. Koufaty, Alan J. Miller, and Michael Upton. Hyper-Threading technology architecture and microarchitecture. *Intel Technology Journal*, 6(1), February 2002.

[9] Daniel Molka, Daniel Hackenberg, Robert Schöne, and Matthias S. Müller. Memory performance and cache coherency effects on an Intel Nehalem multiprocessor system. In *PACT '09: Proceedings of the 2009 18th International Conference on Parallel Architectures and Compilation Techniques*, pages 261–270, Washington, DC, USA, 2009. IEEE Computer Society.

[10] Daniel Molka, Daniel Hackenberg, Robert Schöne, and Matthias S. Müller. Characterizing the energy consumption of data transfers and arithmetic operations on x86-64 processors. In *Proceedings of the 1st International Green Computing Conference*, pages 123–133. IEEE, 2010.

[11] Subhash Saini, Andrey Naraikin, Rupak Biswas, David Barkai, and Timothy Sandstrom. Early performance evaluation of a "nehalem" cluster using scientific and engineering applications. *SC Conference*, 0:1–12, 2009.

[12] D.M. Tullsen, S.J. Eggers, and H.M. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. In *Computer Architecture, 1995. Proceedings. 22nd Annual International Symposium on*, pages 392 – 403, June 1995.

[13] Theo Ungerer, Borut Robič, and Jurij Šilc. A survey of processors with explicit multithreading. *ACM Comput. Surv.*, 35:29–63, March 2003.