Evaluation of energy consumption and performance of Intel's Nehalem architecture

Axel Busch, Technical Student

Julien Leduc, CERN openlab Fellow

October 29, 2009

Contents

1 Executive summary

The Intel Xeon® 5500 processor family (codenamed Nehalem) arrived recently on the market and provides several interesting new properties and functionalities. The performance per clock and the energy efficiency were increased. Simultaneous Multithreading (SMT) was reintroduced and a functionality called Intel Turbo® Boost Technology was implemented for the first time. We measured the performance gain using the popular benchmark suite SPEC CPU2006. The use of this suite also allowed the energy efficiency of the new architecture to be evaluated. An interesting observation was the impact of SMT and Turbo mode. More precisely the measurements were done for three flavours of Nehalem: The low power L5520, the mid-range E5540 and the most powerful of the Nehalem series, the X5570.

The gain with SMT with small benchmarks was quite impressive, thus it was decided to have a closer look at the overall performance it provides for larger physics applications. For these tests, a benchmark from the GEANT4 Suite and the ALICE framework were taken. A multithreaded benchmark, based on Intel's Threading Building Blocks (TBB), was also used. The benchmarks were controlled by a Linux kernel utility called CPUset, that allows threads to run on defined logical cores. This provides a detailed insight into the scalability of the scheduling constrained applications, as well as the amount of benefit in performance and power consumption one can be expect using SMT, and what the gain can be in practice.

With the help of CPUset it was also possible to reevaluate the older SMT implementation of the 2005 Xeon (introduced as Irwindale) in terms of performance.

2 Introduction

CERN, the European Organization for Nuclear Research, has finished the build and repair process of the LHC[1]. It is now a matter of weeks before the particles make again their way through it, and the experiments relying on it start to increase the pressure on the Computer Center again.

The CERN Computer Center, built in 1972, must sustain the increasing flows of data and associated higher workloads, while being constrained to a global power envelope of 2.9 MW for its computing facilities. To solve this equation of an ever increasing computing power under a fixed electrical power limitation, the only possibility resides in increasing the global server efficiency. CERN does this by regularly replacing the oldest, so the less efficient servers, with the most efficient ones available. Today, many of its latest servers are equipped with Intel's "Harpertown" series CPUs, based on the latest Intel Core2 microarchitecture.

Intel's current processor generation, the "Nehalem" CPU, sports a new architecture, and includes several new features. This paper discusses the performance of the Nehalem compared to the previous Intel architecture, the impact of the new features in terms of efficiency, and their possible usage in the Computer Center.

2.1 Intel Microarchitecture, codenamed Nehalem

Intel's current processor generation is nicknamed "Nehalem". In Intel's "tick-tock-model"[2] a tick represents a smaller and more efficient version of the microarchitecture, with an increased transistor density. A tock is a new version of Intel's microarchitecture. Nehalem describes a tock. In this architecture Intel reintroduced SMT[3], previously implemented in Intel's Pentium 4 as Hyper Threading Technology (HT Technology). The aim of SMT is to increase the throughput of the CPU when more than one thread is running. Another new feature of Nehalem is the "Intel Turbo Boost Technology" (Turbo mode). It allows automatic overclocking of the CPU cores when the entire processor is below the specified Thermal Design Power (TDP). This is often the case when only one core is loaded and all the other cores are idle. This said, Turbo mode is also possible if all cores are running under some load. Nehalem can overclock dynamically the cores by up to two "Speed Bins". One Speed Bin is correlated to the QPI's base frequency of 133 MHz.

Another new feature is the integrated memory controller. It allows the CPU to access the local memory with lower latency. As a result the CPU[4] needs less time to wait for uncached data and can work more efficiently. The Nehalem memory controller can access up to 3 DDR3 channels. Additionally, this new architecture implements Intel's Quick Path Interconnect (QPI), a Point-to-Point connection between the chipset and CPU. This increases scalability and throughput, and replaces Intel's Front-Side-Bus. QPI is also used for communication between the individual cores. The third improvement is the addition of an L3 cache with a capacity of 8 MB.[1]

[1] Large Hadron Collider
[2] http://www.intel.com/technology/tick-tock/index.htm
[3] Simultaneous Multithreading
[4] CPU = processor

Another new feature is the possibility to almost completely deactivate each core to save power. This is known as "C6 mode" and is used to decrease the consumption of single CPUs if they are unused. Depending on the workload it is possible to set up to 3 cores in C6 mode.

With all of these additions a new socket was deemed necessary. The current socket is the LGA1366, supporting CPUs with TDP ranging from 60 W (low power L5500 Series) up to 130 W (extreme performance X5500 Series).

3 Energy measurements

CERN IT has decided for its energy measurements, 20 % of the time a given server should be in an idle state and 80 % under load. We therefore use this mix as our reference and it is defined as the CERN IT standard measurement. Multiple energy measurements were executed for each server configuration. The following BIOS configurations were used for each hardware configuration:

- SMT mode off and Turbo mode off
- SMT mode on and Turbo mode off
- SMT mode off and Turbo mode on
- SMT mode on and Turbo mode on

3.1 Hardware configuration

Three Nehalem processor flavours, L5520@2.26 GHz, E5540@2.53 GHz and X5570 @2.93 GHz, were installed on a dual socket Intel S5520UR motherboard. This motherboard supports up to 12 DDR3 DIMMs of memory, running at 800, 1066 or 1333 MHz.[2] With each of the three flavours the board was equipped with 12 2 GB memory modules. It is also equipped with two 500 GB SATA drives. The memory ran at 1066 MHz.

A second test system was the Intel's SX58SO desktop board for the Intel Core i7 965@3.2 GHz. This motherboard was a single-socket system, equipped with one Intel Core i7 CPU, supporting a maximum of 8 GB RAM (1066, 1333 or 1600 MHz modules).[3] For the tests it was equipped with 6 GB of memory and ran in the 1333 MHz mode.

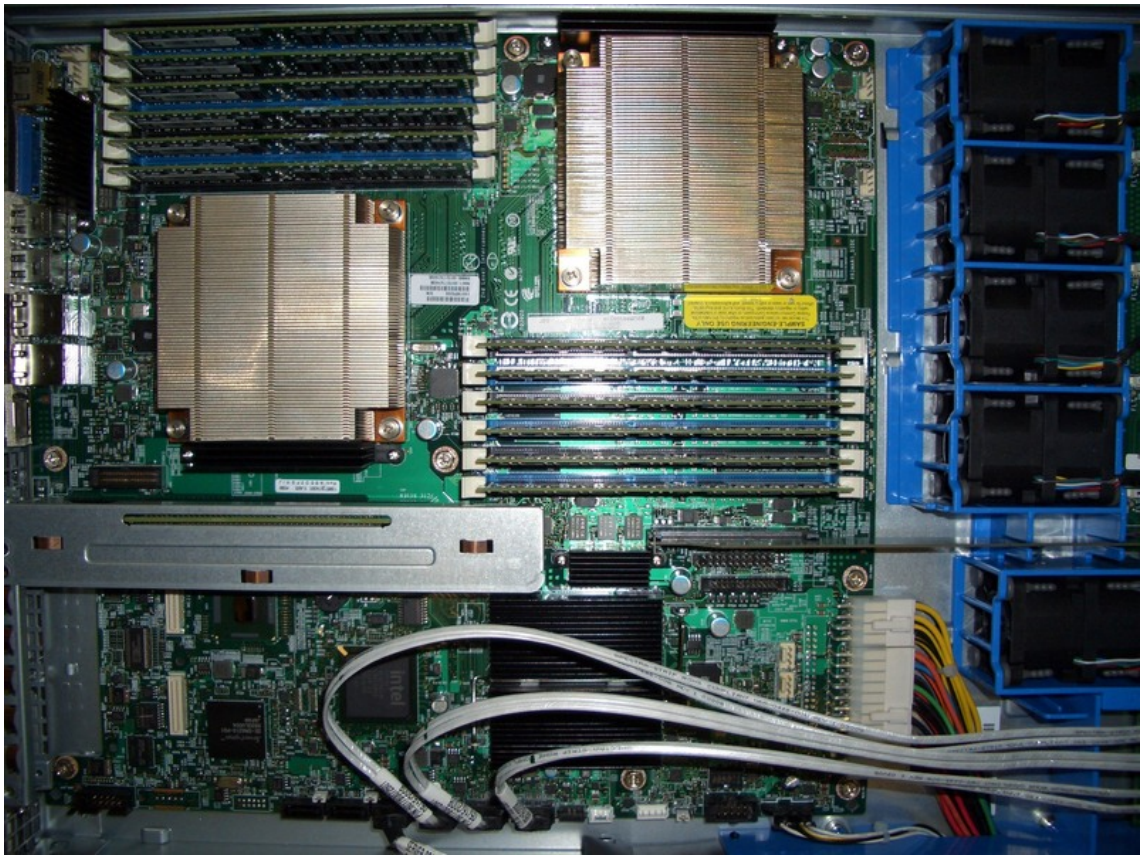


Figure 1: Xeon 5500 system with Intel S5520UR Motherboard

3.2 System software setup

Scientific Linux CERN 5.3 (SLC 5) was used on both machines for all tests. SLC 5 is based on Red Hat Enterprise Linux 5 (Server). The standard SLC5 kernel (version 2.6.18-128.1.1.el5) was used.

3.3 Measurement equipment

For the energy measurements, a high precision power analyzer LMG450 of ZES-Zimmer Electronic Systems was deployed. It was possible to connect the power analyzer via the RS232 interface to a laptop allowing the current energy consumption to be read. Each measurement was carried out two times. First, the energy consumption was measured in an idle state, and then, with a separate run, it was measured while the server was under load. The results were sent then to a specified email account by a script running on the laptop. Finally the average result was written to a spreadsheet.

3.4 ZES-Zimmer LMG450 power analyzer

The ZES-Zimmer LMG450 power analyzer allows the measurement common power electronics and as well as network analysis. It has an accuracy of 0.1 % and allows the measurement of four channels at the same time. For openlab only three values are of importance:

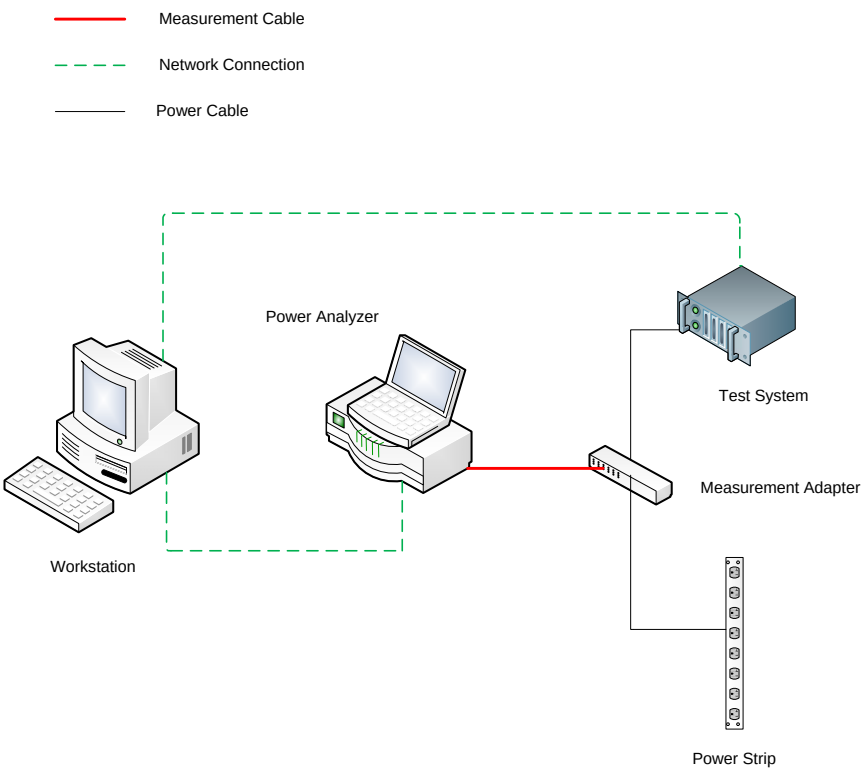


Figure 2: Power test setup

- *Active Power (P)* : The active power is also often called "real" power and is measured in Watts (W). If the active power is measured over time the kilowatt hours value is determined.

- *Apparent Power (S)* : Apparent power is the product of voltage (in volts) and current (in amperes) in the loop. This part describes the consumption from the electrical circuit. It is measured in VA.

- *Power Factor* : In our case, the power factor means the efficiency of the power supply. The closer the power factor is to one, the better is the efficiency. ($power\,factor = \frac{active\,power}{apparent\,power}$)[4]

The idle power consumption was measured while only the standard operating system was working with no other additional applications or components. To measure the servers under load, it was ensured that the machines were truly fully loaded by using CPUBurn to stress the CPU cores. Since more and more memory is installed in modern multicore servers, the energy consumption of the memory becomes increasingly more significant. For this reason, LAPACK was used to generate a memory load in parallel to CPUBurn.

3.5 CPUBurn

CPUBurn was originally made as a tool for overclockers, so that they can stress the overclocked CPUs, and check if they run stably or not. It can report if an error occurs while the benchmark is running. It runs Floating Point Unit (FPU) intensive operations to get the CPUs under full load, allowing the highest power consumption to be measured.

3.6 LAPACK

LAPACK (Linear Algebra PACKage) was written in Fortran90 and is used to load both the memory system and the CPU. It provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The memory consumption depends on the size of the generated matrices.

3.7 Results of the energy measurements

In figure 3 wee see the CERN IT energy measurements. These measurements do not convey much information about the throughput efficiency of the system. It was therefore necessary to perform the performance measurements with SPEC CPU2006, in order to create a correlation between achieved SPEC marks and energy consumption.

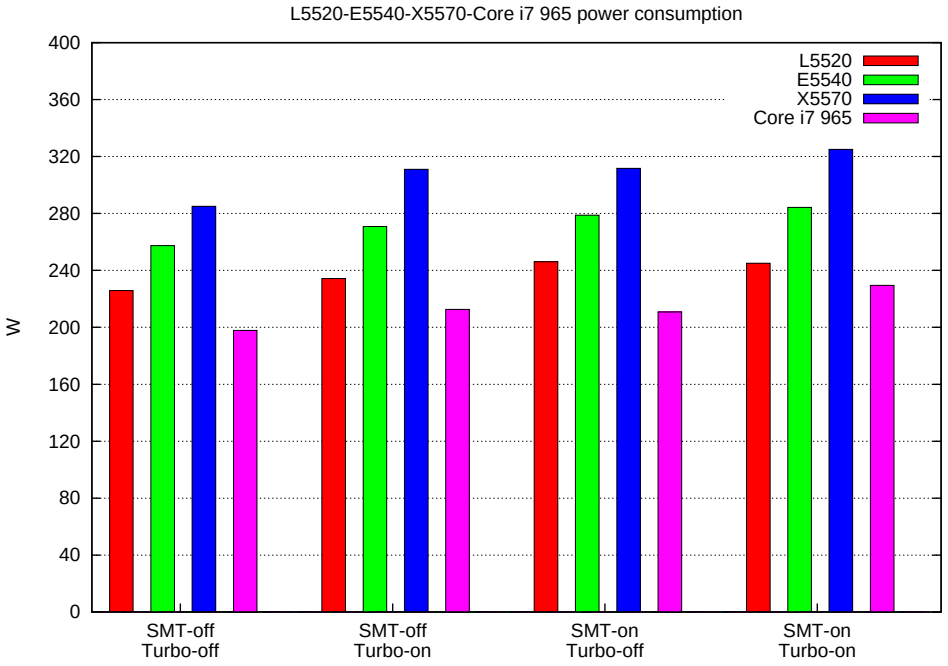


Figure 3: CERN IT power measurements: 20 % Idle and 80 % load mixture

4 Performance measurements

4.1 SPEC CPU2006

For executing the CPU performance measurements the SPEC CPU2006 benchmark from the SPEC[5] Corporation was used. SPEC CPU2006 is an industry standardized benchmark suite, made for stressing a system's processor, the caches and the memory subsystem. The benchmark suite contains real user applications and the source code is available. A HEP working group has demonstrated load correlation between the SPEC results and High Energy Physics applications. The C++ subset of the tests from the SPEC CPU2006 benchmark suite is used, to cover the requirements of CERN. It was compiled with GCC 4.1.2, GCC 4.3.3 and Intel's ICC v. 11.0; each in 32- and 64-bit mode. The results show the differences between all the compilers, and the respective compiler versions.

The performance measurements were also carried out with the same BIOS settings as for the power measurements, i.e. SMT mode on/off and Turbo mode on/off. In total, four test series, comprising of six single SPEC jobs had to be completed (32 and 64 bit for each of the three compilers).

4.2 test40

The benchmark test40, is a test based on the Geant4 simulation toolkit. Geant4 is typically used in High Energy Physics (HEP) for simulating the passing of particles through matter. As a benchmark candidate for the SPEC CPU2006 benchmark suite, test40 was used to test the new Nehalem CPU. In order to compare these new results with results that were recorded previously GCC 4.3.0 was used for compiling the code of test40. This ensures that the same conditions were established for the Nehalem system as for the previous system.

4.3 tbb benchmark

"tbb" is a benchmark based on the track fitter from High Level Trigger in ALICE. Openlab adapted it to run multithreaded on X86 using Intel's Threading Building Blocks (TBB), hence its name.

4.4 Results

4.4.1 Results - SPEC CPU2006

The results of the various performance measurements showed that the highest SPEC CPU2006 marks on the different flavours of the Nehalem were obtained enabling both SMT and the Turbo mode. According to 7 and 5, the absolute SPEC marks are about 140 for the L5520, 150 for the E5540 and 170 for the X5570.

The results show that Turbo mode can provide a benefit from about 1 % (with GCC 4.3.3, 64 bit, L5520) to 12 % (with GCC 4.1.2, 64 bit, E5540). SMT achieves a gain

[5] Standard Performance Evaluation Corporation (http://www.spec.org)

from 19 % (with GCC 4.1.2, 32 bit, E5540) to 30 % (with GCC 4.1.2, 64 bit, L5520). Intel's ICC compiler, provided a maximum performance gain of 15 % (with SMT enabled and Turbo mode disabled) over GCC 4.1.2 or GCC 4.3.3 in the 32 bit mode. In the 64 bit mode it offered 5 % (also with SMT enabled and Turbo mode disabled) as maximal additional performance.

In previous years, the energy costs for computing were rather insignificant but now, in times of increasing energy costs, it becomes an important topic. The efficiency unit used here is SPEC marks per Watt and is calculated for the different CPUs with the ratio of the reached SPEC mark using GCC 4.1.2 compiler in 32 Bit mode by the CERN IT standard measurement. The choice of the conditions for the SPEC measurements were done according to the standard default compiler included in CERN SLC 5.3 and the standard compilation mode.

In those conditions, the reference system using Harpertown reached 60.76 SPEC marks consuming 200 Watt.

According to figure 8, the L5520 CPU is the most efficient of the 3 Nehalem flavours and the Core i7 CPU. During the tests it reached 0.411 SPECs/Watt. With 0.393 for the E5540 and 0.389 SPECs/Watt for the X5570, both were slightly less efficient than the L5520. The Core i7 965 reached 0.295 and the reference system, equipped with the Harpertown 5410, reached 0.303 SPECs/Watt. Thus, in our measurements, the Nehalem reached about 36 % more efficiency (SPECs/Watt) compared to the Harpertown. In our study of the Nehalem CPUs, the Core i7 turned out to be the least efficient one. But the comparison is somewhat unfair, since it is a desktop CPU and the peripheral components are less optimized for power savings.

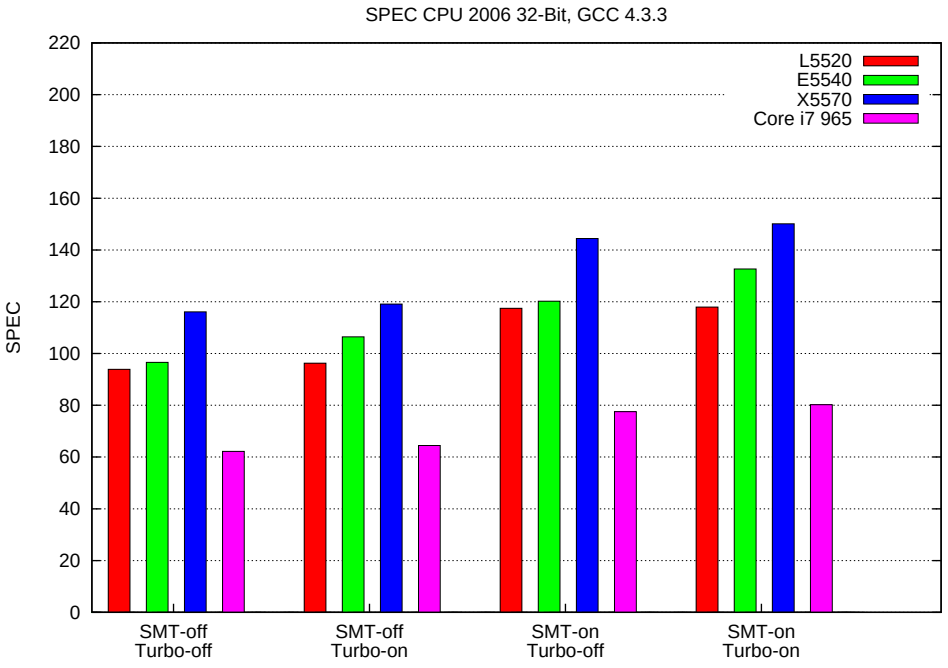


Figure 4: SPEC CPU2006 results of Xeon 5500 and Core i7 965 for GCC 4.3.3, 32 Bit

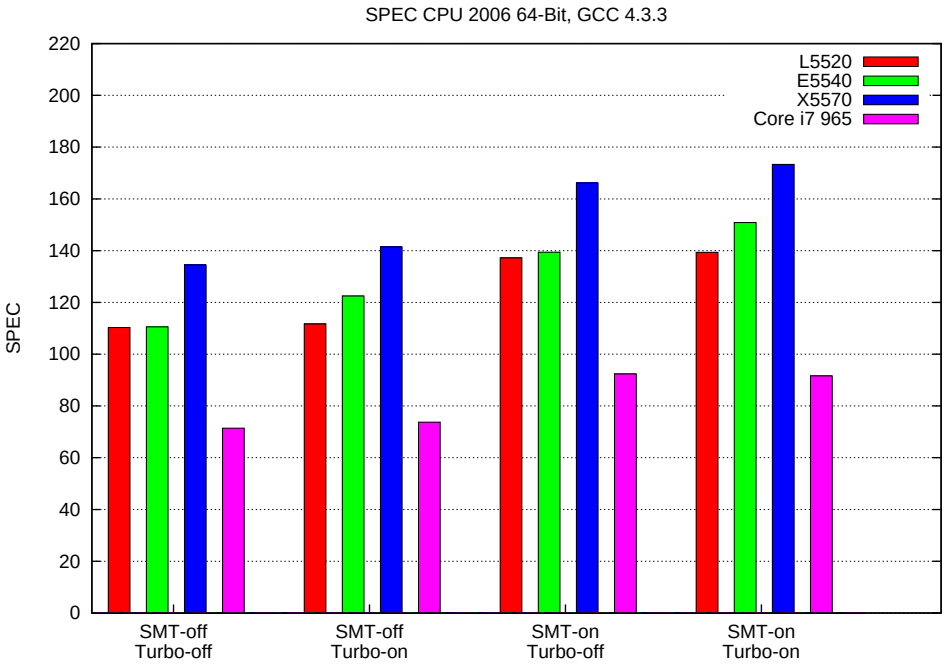


Figure 5: SPEC CPU2006 results of Xeon 5500 and Core i7 965 for GCC 4.3.3, 64 Bit

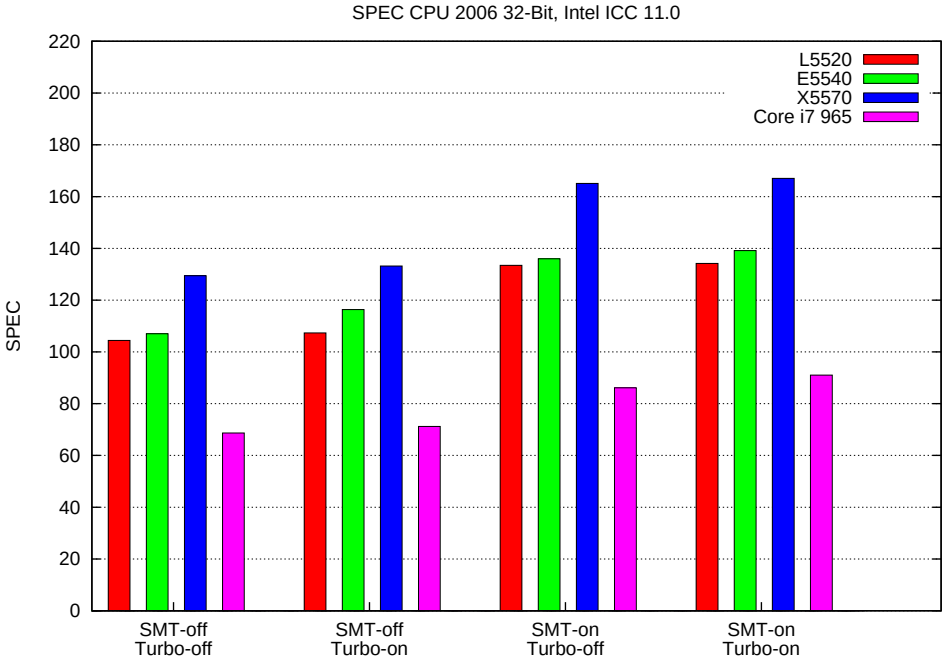


Figure 6: SPEC CPU2006 results of Xeon 5500 and Core i7 965 for Intel ICC 11.0, 32 Bit

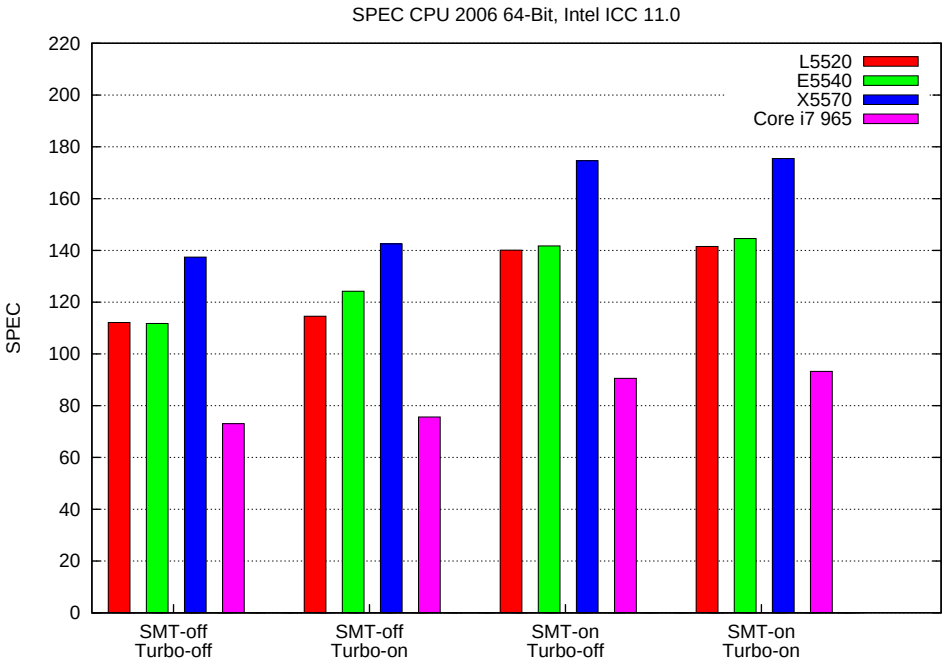


Figure 7: SPEC CPU2006 results of Xeon 5500 and Core i7 965 for Intel ICC 11.0, 64 Bit

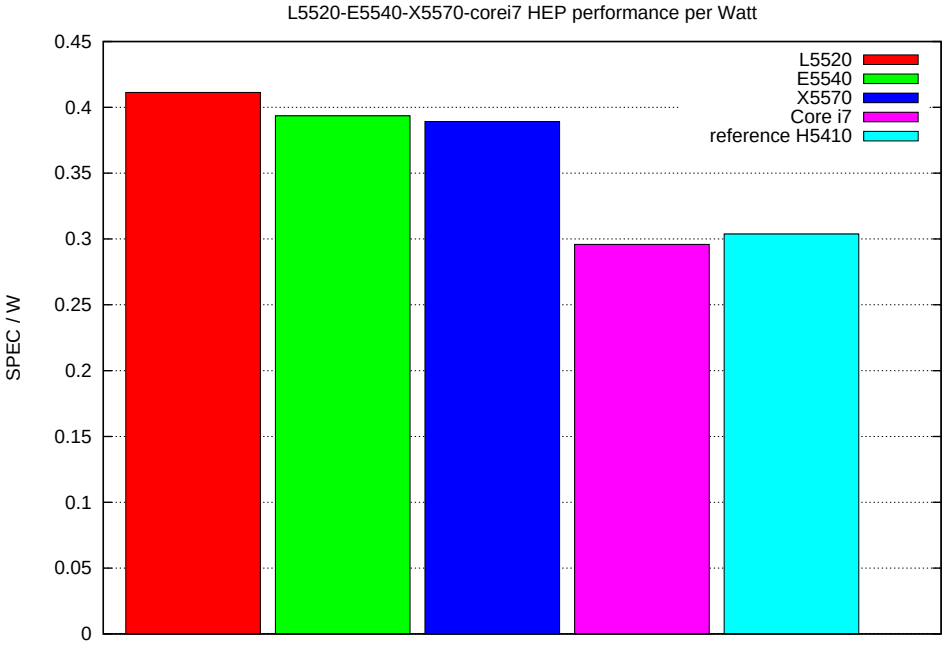


Figure 8: Efficiency of Xeon L5520, E5540, X5570, Core i7 965 (SMT-off, Turbo-on) and Harpertown H5410 (GCC 4.1.2, 32 Bit)

4.4.2 test40 & tbb

The benchmarks test40 and tbb were only tested with the E5540. The older Harpertown E5472 needed about 32 seconds(t) for a single test40 run. For the same benchmark the E5540 needed 34 seconds. Therefore throughput was only at 95 %. But the E5540 needed about 10.4 % fewer cycles to compute the same work (1 - $\frac{t_{E5540} \cdot f_{E5540}}{t_{E5472} \cdot f_{E5472}}$). It should not be forgotten that the E5472 has a clock frequency(f) of 3.0 GHz with only 4GB RAM and the E5540 runs at 2.53 GHz and has 24 GB RAM which also uses energy.

If the energy consumption is taken into account the E5540 has a throughput per Watt that was between 20 % to 40 % more than that of the E5472. Throughput per Watt was up to 780 % (SMT on, Turbo mode on and 16 threads) of performance per Watt compared to the E5472, with only one thread. If the E5472 uses 8 threads, the E5540 with 16 threads was about 18.75 % more efficient.

tbb provided similar results to those of test40. The real fit time per track needed for the E5540 about was 0.51 μs, whereas the E5472 took only 0.47 μs. Concerning the energy consumption, there was a gain of 3 to 10 % in throughput per Watt. This goes up to 542 % (SMT on, Turbo mode on and 16 threads) of performance per Watt, over that of the E5472, with only one thread. If the previous E5472 system computes 8 threads at the same time and the new E5540 calculates 16 threads, the E5540 was 1.84 % more efficient.

5 Evaluation of SMT

When SMT is enabled the operating system recognized more processors in the system than it actually had. One physical processor with one physical core was shown as two logical processors. The term "logical" was used, because two logical processors are not the same as a dual core processor. A dual core processor has for each core its own set of execution units. The two logical cores on one physical core share the same execution units.[5]

Since the tests conducted showed that SMT can provide an interesting gain in performance, a closer investigation of SMT was undertaken. The aim was to find out how SMT behaves with real applications and what kind of performance can be expected. The tests were again carried out with test40, the tbb benchmark and the framework of the ALICE experiment.

5.1 CPUset

To evaluate SMT, it was necessary to make sure that a set of processes run only on specific cores and that the Linux scheduler was forced to use only those; and fortunately CPUset provides this functionality. CPUsets are objects in the Linux kernel, which allows the partitioning of machines in terms of CPUs. It is a virtualization layer and allows the creation of exclusive areas in which processes can be attached and are allowed to run. This function has been implemented in kernel 2.6 as a kernel patch.

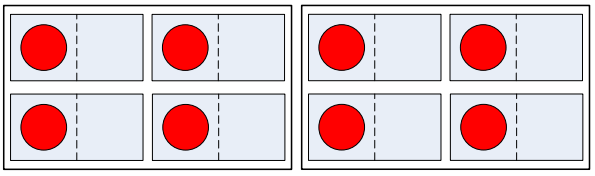
CPUsets can be created and manipulated by a pseudo filesystem. In a CPUset it is also possible to create sub CPUsets. Processes and memory (memory nodes) can then be allocated.

5.2 Test procedure

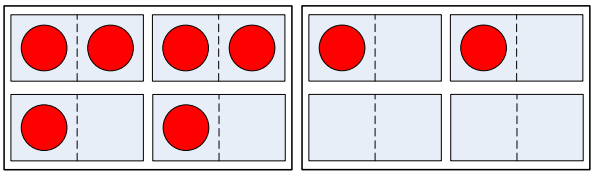
Altering BIOS settings is cumbersome, because CPUsets will restrict all processes to particular cores by themselves. Thus, the number of CPUset tests has a tending to increase quickly. Therefore SMT and turbo mode were switched on all the time.

The Nehalem dual-socket system provides 8 physical cores, and when SMT is enabled, each physical core hosts 2 logical cores, providing a total of 16 logical cores. Thus 1 to 16 cores can be tested. For example with 8 cores, a number of different CPUset configurations were possible. Here are several examples:

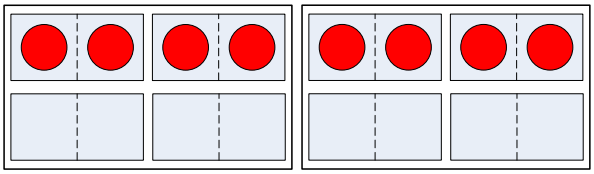
- Eight threads using eight logical cores on eight physical cores



- Eight threads using eight logical cores on six physical cores



- Eight threads using eight logical cores on four physical cores



If this was carried out for all possible configurations, there would be 44 tests to complete for test40, and again 44 tests for the tbb benchmark:

$2 \cdot \sum_{i=1}^{\frac{n}{2}} (\lfloor \frac{i}{2} \rfloor + 1) - \frac{\lfloor \frac{n}{2} \rfloor}{2}$, n = number of logical cores; two logical on one physical core

Due to process scheduling issues with odd numbers of CPU cores (for example 14 logical cores on 7 physical cores), it was not possible to measure all configurations.

These tests were carried out in order to understand in which area the gain of SMT shows up, and also how the hardware behaves in terms of energy consumption. Energy consumption was measured as with the comparison between Harpertown and Nehalem.

5.3 test40 & the tbb benchmark

test40 is a single threaded benchmark, so it was forked *n*-times and restricted to a CPUset with *n* cores attached. Therefore, runtime of a single test40 benchmark does not decrease using more cores. It can be seen that single test40 instances are atomic, and thus have no influence on each other. As a result, the gain with SMT can be observed.

Unlike test40, the tbb benchmark is a multithreaded benchmark, and thus it was not forked *n* times, but instead the tbb scheduler was restricted to use *n* threads for *n* cores. Thus, the runtime of the tbb benchmark should decrease if more cores are used. This benchmark also shows the gain with SMT and the scalability of Intel's TBB.

5.4 Results of test40 & the tbb benchmark

5.4.1 test40 results

The test40 scheduling graph in figure 9 shows the execution time, according to the number of simultaneously scheduled processes and to the scheduling policy. The green line represents the execution time according to the scheduling policy that tries to maximize the number of dedicated physical cores. Here, n is the number of processes:

- $1 <= n <= 8$: between 1 and 8 scheduled processes, each process was assigned to a dedicated physical core: n logical cores on n physical cores.

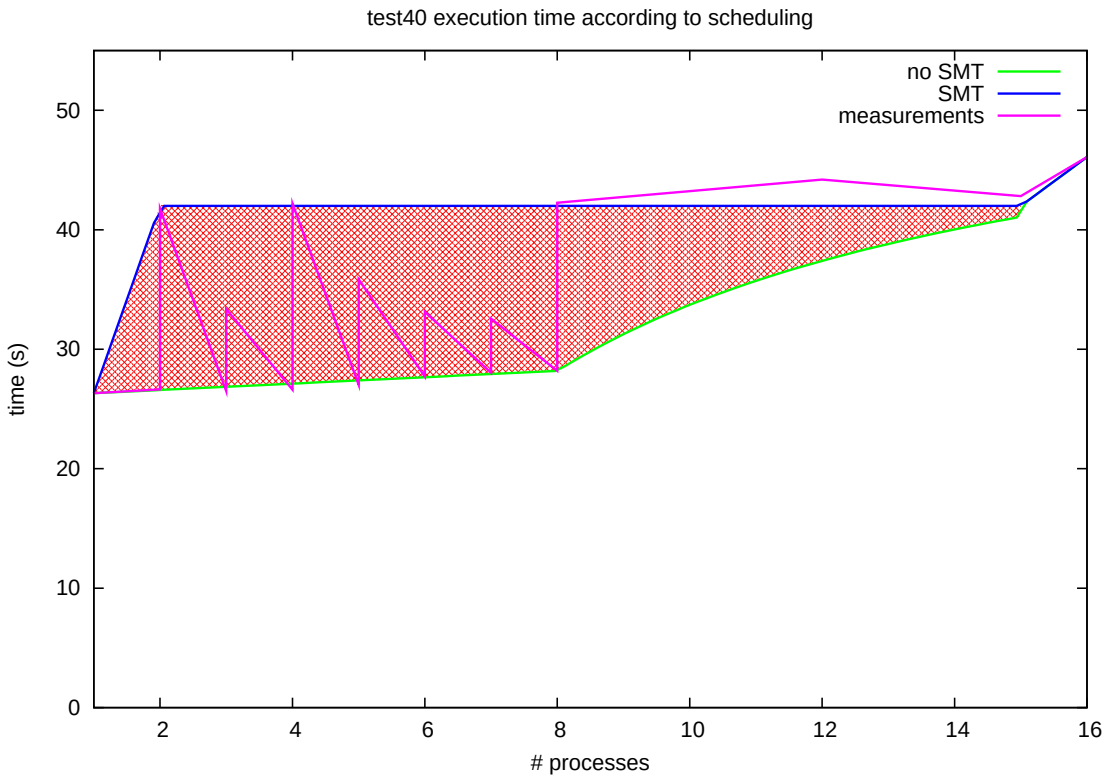


Figure 9: Execution time of test40 on Xeon X5570 using #processes

- $9 <= n <= 16$: between 9 and 16 scheduled processes, there were not enough physical cores available for all the processes anymore: n logical cores on 8 physical cores.

The blue line, being the opposite scheduling policy, tries to minimize the number of physical cores used on the server: n processes use n logical cores on $\lfloor (n+1)/2) \rfloor$ physical cores. Those two scheduling policies are two extremes, and all the other scheduling schemes will give an execution time within these two extreme boundaries. Thus, graphically the other scheduling policies are in the red area.

To deepen the analysis, the execution time was correlated with the power consumption during the runs, in order to deduce the overall amount of energy, consumed by each run. This correlation is shown in figure 10, and now highlights 3 interesting scheduling policies:

1. *"loose no SMT"* scheduling policy, that tries to maximize the use of dedicated physical cores.
2. *"SMT"* scheduling policy, that tries to minimize the number of dedicated physical cores.
3. *"strict no SMT"* scheduling policy, that uses only dedicated physical cores and requires an additional server to run further processes.

Figure 10 highlights a new area, shown in red, that corresponds to the energy consumed by any scheduling policy that tries to minimize the number of servers.

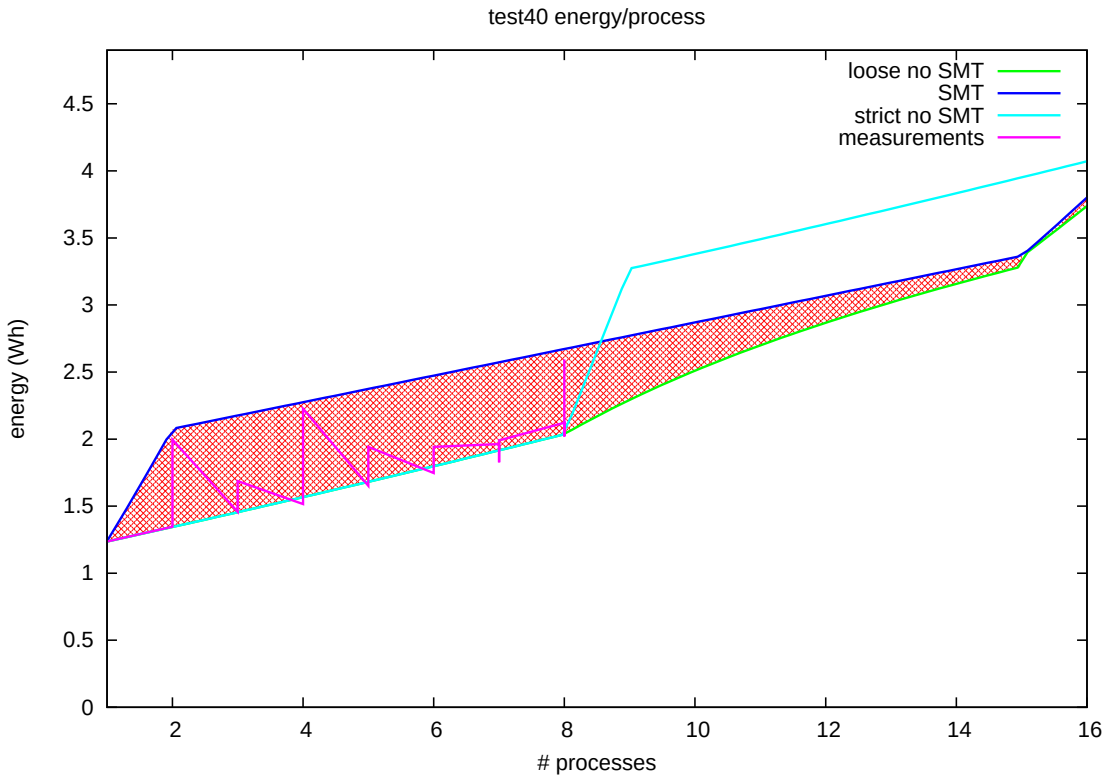


Figure 10: Energy per process of test40 on Xeon X5570 using #processes

These first two graphs, figures 9 and 10, allow taking another point of view and analyze the gains in terms of the throughput SMT offers, compared to a sole reliance on the dedicated physical cores. Indeed, to evaluate SMT, an even number of processes was required in order to schedule a process on the two logical cores of each physical core. When using a *strict no SMT* scheduling policy the exact same physical core only runs one process. The result was that a single core runs one process in 26 seconds, with the *strict no SMT* policy, or two processes in 42 seconds using the *SMT* policy.
Below is the resulting execution time taken to run two processes with both of the afore mentioned policies on a dedicated physical core:

- *strict no SMT policy*, a single core requires $2 * 26 = 52$ seconds
- *SMT policy*, the two processes run in 42 seconds

This clearly displays the advantage in using SMT scheduling in terms of throughput. Moreover, the gain in throughput was larger than the consumption penalty seen in figure 10. Thus, SMT not only increases overall compute efficiency, but also increases energy efficiency. Figure 11 depicts those gains in terms of throughput and power efficiency, thanks to SMT, depending on the number of dedicated physical cores. According to this, the gain in performance with SMT was between 20 to 25 %, increasing linearly when using more and more physical cores. At the same time the energy consumption was pretty stable, at just over 15 % when using SMT. The drop observed at 8 physical cores, is due to the fact that scheduling 16 processes with SMT implies using the last

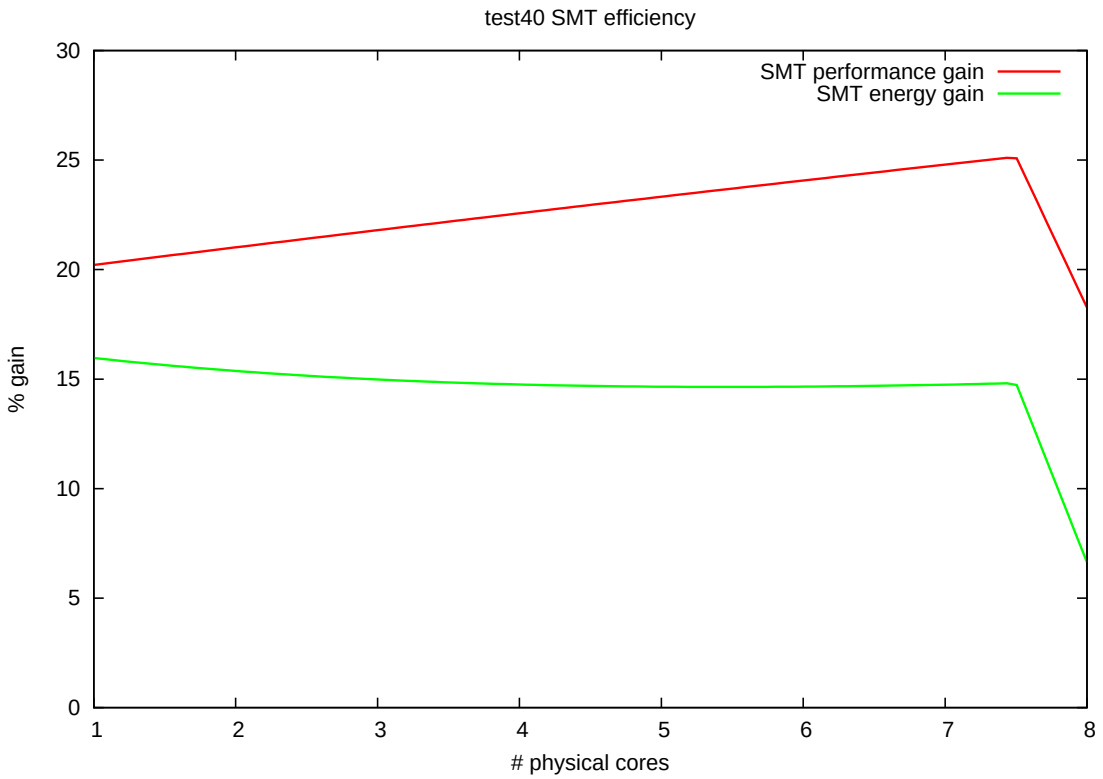


Figure 11: Efficiency per thread of test40 on Xeon X5570 using #processes

core, $core_0$, that is normally running more OS related processes. This leads to lower increases in both performance and energy.

5.4.2 tbb results

Since tbb is a multithreaded process, all the comparisons for tbb were observed using the number of threads within a single process. This allows a more intuitive, direct comparison, since running a tbb process with any number of threads always carries out a fixed amount of work.

Figure 12 shows the execution time according to the number of threads and the scheduling policy. The green line represents the execution time according to the scheduling policy that tries to maximize the number of dedicated physical cores. Let say that n was the number of threads:

- $1 <= n <= 8$: between 1 and 8 threads, each thread was assigned to a dedicated physical core: n logical cores on n physical cores.
- $9 <= n <= 16$: between 9 and 16 threads, there were not enough physical cores for all the threads available anymore: n logical cores on 8 physical cores.

The blue line, was the opposite scheduling policy, where as it tries to minimize the number of physical cores used on the server: n processes use n logical cores on $\lfloor n/2 \rfloor$ physical cores. All other scheduling policies land in the red area. This plot also shows

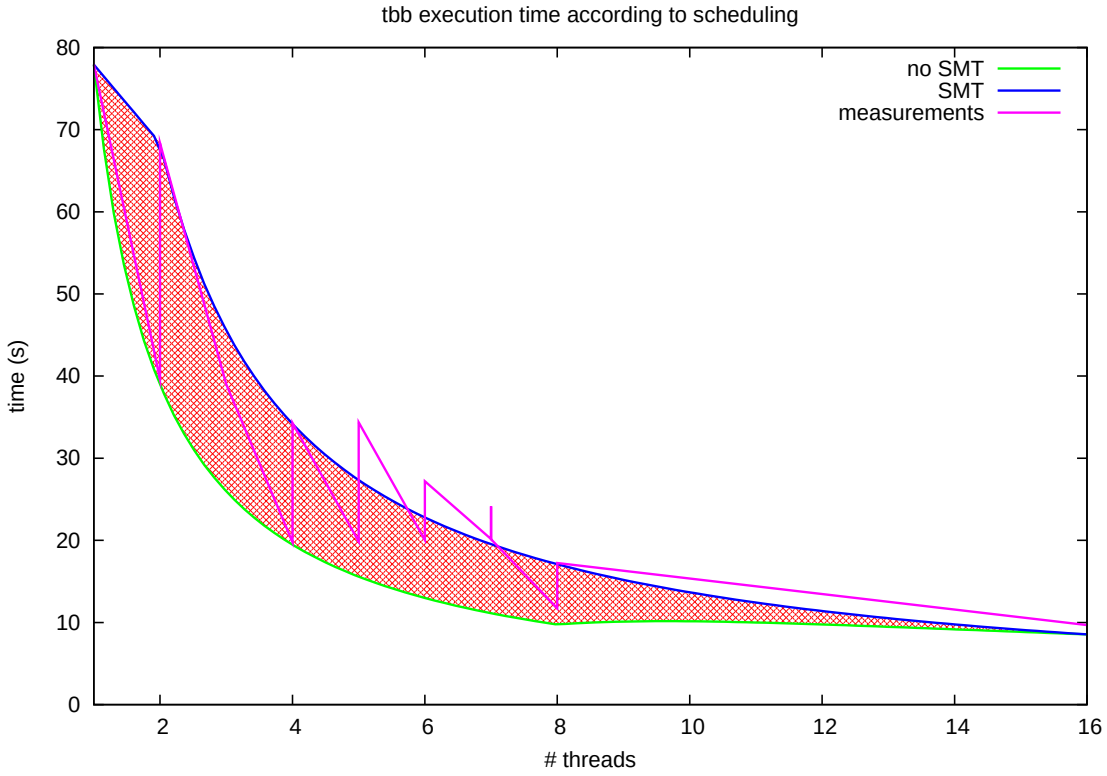


Figure 12: Execution time of tbb benchmark on Xeon X5570 using #threads

that the tbb benchmark scales well, because using twice the amount of threads almost halves the execution time.

As already done for test40, the execution time was correlated with power consumption during the runs according to the number of threads and the scheduling policies. This is shown in figure 13, with again the three different scheduling policies: *loose no SMT*, *SMT* and *strict no SMT*.
The first two graphs, figures 12 and 13, can also be analyzed in terms of computing throughput. Here, to compare different policies with a fixed number of dedicated cores used, the execution times can be directly compared using twice the number of threads for the *strict no SMT* scheduling policy.
To run a single tbb process fully using n cores requires:

- *strict no SMT policy*, n cores run n threads
- *SMT policy*, n cores run $2 * n$ threads

If we dedicate two physical cores to one tbb process:

- *strict no SMT policy*, two threads run on two logical cores on two physical cores, and require 38.9 seconds, and consume 2.033 Wh.
- *SMT policy*, four threads run on four logical cores in two physical cores, and require 34.2 and consume 1.917 Wh.

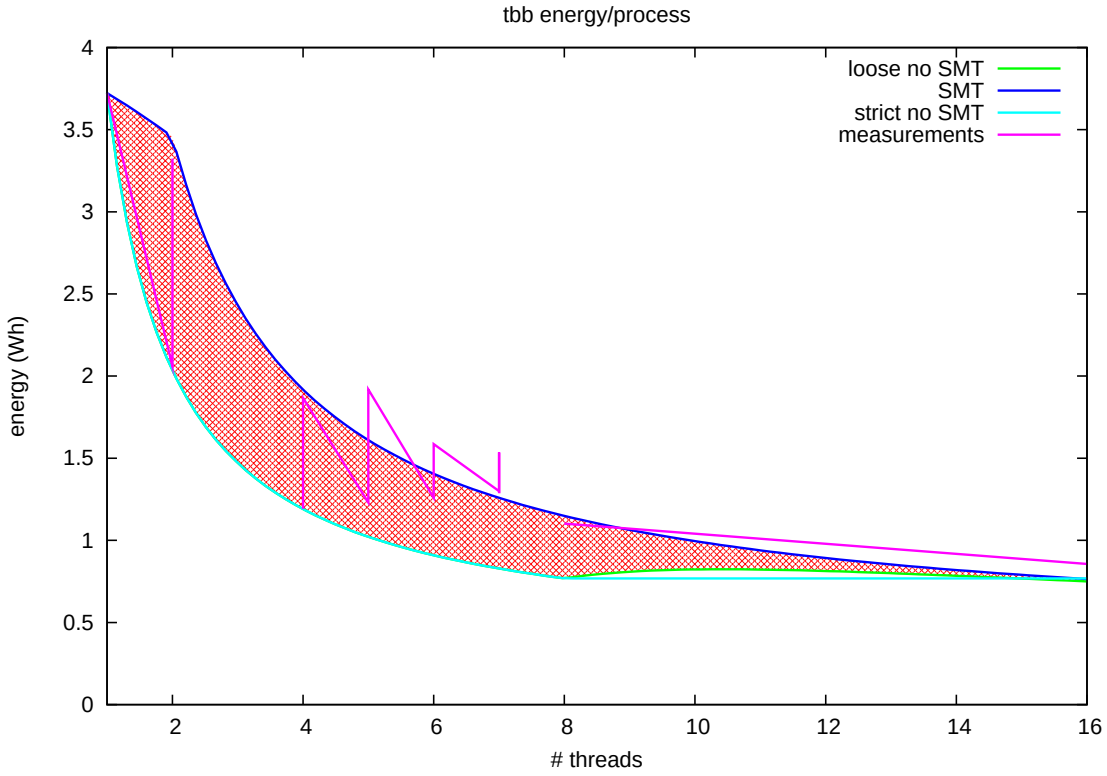


Figure 13: Energy per thread of the tbb benchmark on Xeon X5570 using #threads

This again gives an advantage to the SMT scheduling policy in terms of throughput and energy efficiency. Figure 14 provides a full picture of the gains in terms of both compute efficiency and energy efficiency using the SMT scheduling policy over the no SMT scheduling policy. According to this, the gain in performance was stable at just over 12 %, while the energy efficiency was decreasing from 7 to just under 1 %. SMT is therefore very interesting in terms of throughput, and never a penalizing factor on the energy consumption.

5.4.3 test40 & tbb tests conclusion

The two tests uncovered an interesting gain that SMT provides. To investigate this further it was decided to carry out a deeper analysis with a real physics application: the framework of the ALICE experiment.

5.5 ALICE framework

The ALICE framework contains the Geant3 toolkit and is comprised of two basic parts: the first part being the simulation part, and the second being the reconstruction. For our tests, 100 events were used.
The tests worked fine using eight threads or less, but if more were used the average CPU load decreased (as will explained later).

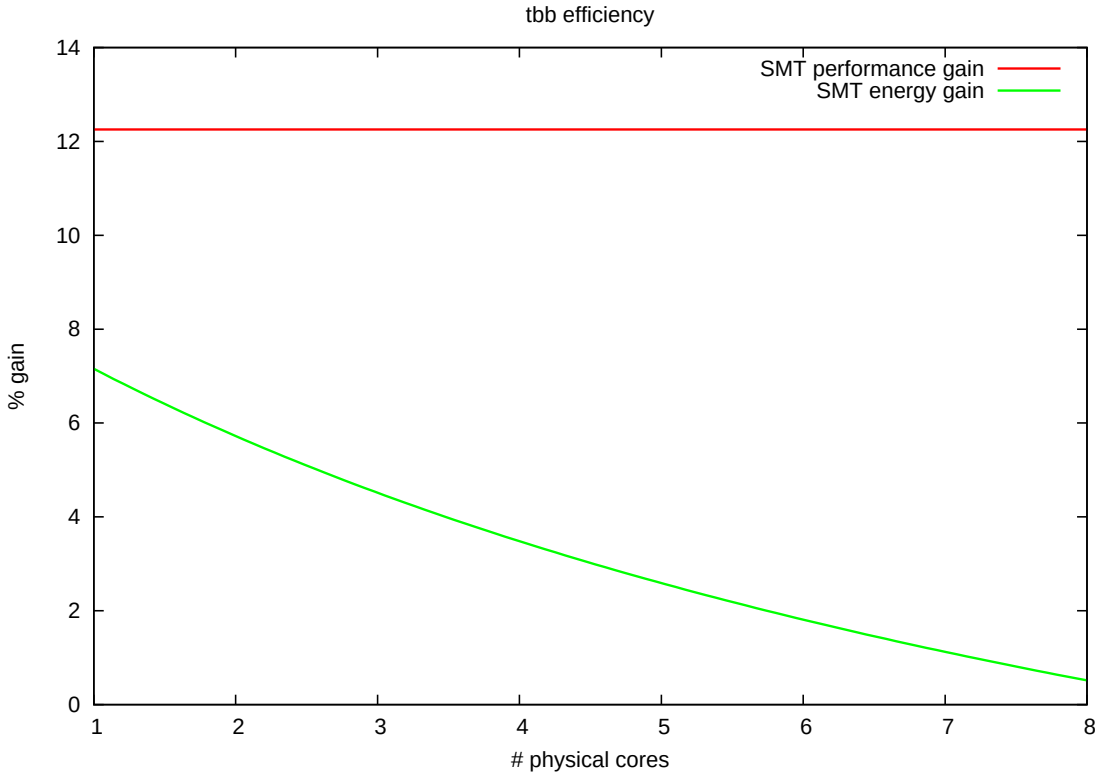


Figure 14: Efficiency per thread of tbb benchmark on Xeon X5570 using #threads

With the aid of sysstat[6] it was observed that in the case of 12 processes that all carry out some I/O disk operations. The result was a lot of I/O waits spent in the CPUs. The Nehalem server only contained one SATA disk drive, with 12 processes of the ALICE framework accessing it at the same time it was simply overloaded. Using two SATA disks to run the ALICE framework, however, showed much less I/O waits and also the CPU load was back to optimal.

	#proc total	#physical cores	#logical cores	Runtime AVG(m)	CPU-time AVG(m)	Throughput
Alice framework	1	1	1	200.270	200.270	100 %
	2	2	2	199.245	398.490	100.5 %
	2	1	2	330.300	330.300	117.5 %
	4	4	4	202.520	810.080	98.9 %
	4	2	4	315.660	631.320	121.2 %
	8	8	8	218.392	1747.136	91.0 %
	8	4	8	300.270	1201.080	125.0 %
	12	6	12	369.130	2214.780	107.8 %
2 disks	12	6	12	340.185	2041.110	115.1 %

SMT provides also a stable gain calculating the work done by the ALICE framework.

[6] "sysstat" is a powerful tool to measure activities of the server subsystems like CPU load, memory usage, memory swapping, network usage or disk load.

Thus, in conclusion, it increased the performance of a real application by at least 15 %.

5.6 Comparison between SMT on the Irwindale and on the Nehalem

Irwindale, a member of Intel's Xeon processor family was introduced in 2005. It was the first CPU that provided an implementation of SMT: Intel Hyper Threading Technology (HT Technology). Early tests showed that HT did not increase the performance and could even sometimes decrease it. To see if the hardware or the software implementation was the problem, the aforementioned tests were repeated on a dual-socket Irwindale server with 2x 3.6 GHz CPUs. Thus, this system had two physical or four logical cores available.

5.6.1 Results of the SMT evaluation on the Irwindale

First, the SPEC CPU2006 and the tbb benchmark were executed on the Irwindale system. The settings of the tbb benchmark were slightly changed, in order to take the much longer runtime for the benchmarks on the Irwindale into account.

At first all the instances of SPEC CPU2006 were only scheduled on two independent hardware threads. The other two hardware threads were left unused by the kernel. However, forcing the scheduler with CPUsets to use all the cores solved this issue. With the help of CPUsets it was shown that the previous SMT implementation on the Irwindale provides a benefit of about 17 % for SPEC CPU2006 (figure 15).

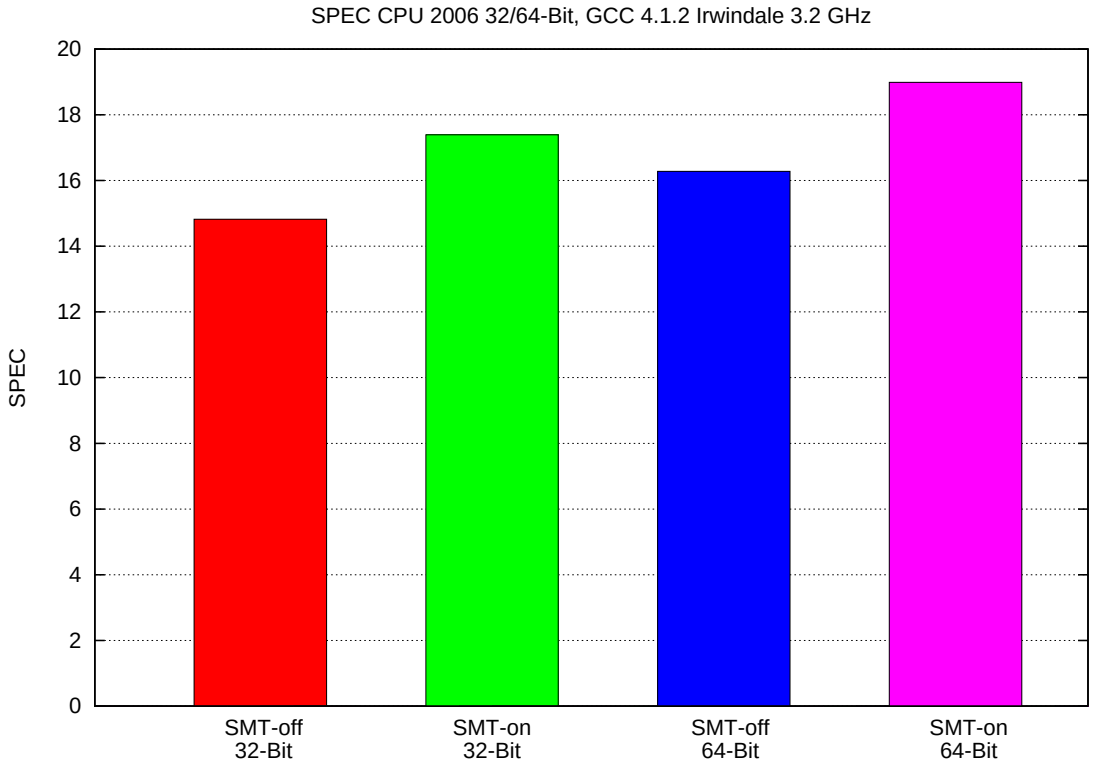


Figure 15: SPEC CPU2006 results of Irwindale 3.6 GHz for GCC 4.1.2

The tbb benchmark provided a gain of about 29 % (figure 16). With two logical cores on two physical cores the benchmark needed 545.737 seconds. Using two logical cores on one physical core the runtime increases to 769.933 seconds. To be able to compare these two values it can be calculated: $\frac{545.737 \cdot 2 - 769.933}{545.737 \cdot 2} = 0.294$
The first runtime has to be doubled, because the tbb benchmark ran on two physical cores and the second time only on one physical core.

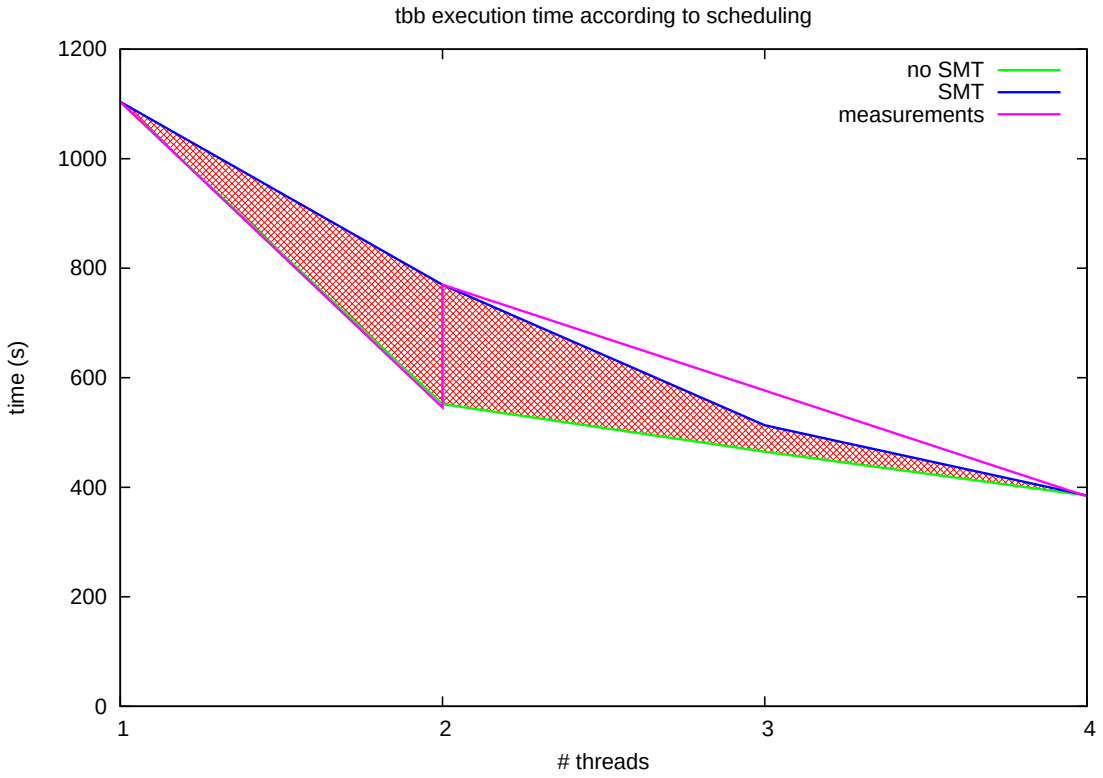
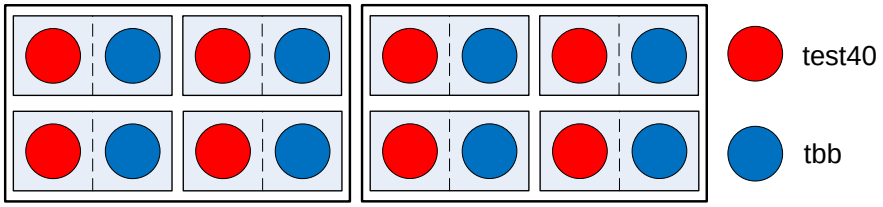


Figure 16: Execution time of tbb benchmark on Irwindale 3.6 GHz using #threads

The conclusion is that it was not the hardware implementation of Irwindale's SMT that caused the initial problems, but rather the way the old Linux scheduler schedules the processes on the individual cores.

5.7 Running test40 & tbb at the same time on Nehalem

A scenario was conceived where in a single unit of time, one instance of test40 would run on one logical core on one physical core, and one instance of tbb on the other logical core of the same physical core. This should show if the gain of SMT decreases if different benchmarks run on the same physical core and should be a more realistic scenario, since not only one kind of application will run on a system, but multiple.



Again the same tbb and test40 benchmarks were used to observe the impact of scheduling a mixture of both benchmarks running on each logical core. The measurements were always carried out twice. Since test40 and tbb do not have the same runtime, it had to be ensured that on the other logical core of the physical core (without the focus for the measurement), the benchmark runs long enough for the benchmark with the focus on it. After that the same procedure was carried out for the benchmark that did not have the focus before.

5.7.1 Results

While running test40 and the tbb benchmark at the same time, test40 provides a speedup of about 2.61 %. The tbb benchmark shows a speeddown of 1.28 % on average. But the result using 16 cores was maybe less representative, due to the fact that one of the jobs was scheduled on $core_0$. There was no significant influence in total. Thus, it was possible to run different benchmarks at the same time observing that they do not interfere with each other significantly.

	#threads total	#physical cores	#logical cores	Runtime AVG (s)	CPU-time AVG (s)	Throughput
test40 (only)	2	1	2	41.770	41.770	100.00 %
tbb+test40: test40	2	1	1	40.650	20.235	102.68 %
	4	2	2	40.685	40.685	102.60 %
	8	4	4	40.710	81.420	102.54 %
	16	8	8	45.928	183.712	(90.05 %)
tbb (only)	2	1	2	68.373	68.373	100.00 %
tbb+test40: tbb	2	1	1	132.660	66.330	102.99 %
	4	2	2	69.270	69.270	98.69 %
	8	4	4	36.070	72.140	94.49 %
	16	8	8	21.310	85.240	(75.33 %)

tbb is a multithreaded benchmark, thus as reference the CPU-time was taken. For the singlethreaded benchmark test40 the average runtime was taken for comparisons. CPU-time = $\frac{\#logical\_cores}{\#physical\_cores \cdot 2} \cdot runtime$

6 Final conclusion

When the performance between a L5520 Nehalem server and a H5410 Harpertown server was compared, the Nehalem server reached a 58.4 % higher performance with SPEC CPU 2006 than the system equiped with a CPU from the previous generation.

The power measurements correlated with the performance results of the SPEC CPU 2006 benchmark observed for the Nehalem L5520 show a gain of about 36 % in terms of energy efficiency, compared to the Harpertown 5410. Compared to the X5570 and E5540, the L5520 was the most energy efficient processor of the Nehalem family in our tests.

As already mentioned, several new performance-related technologies were introduced: Turbo mode was implemented for the first time in the Nehalem. It provides a gain of only percents, but the energy penalty grows linearly with the performance gain. Thus turbo mode is a good way to boost the performance with a relatively modest energy penalty.

The reintroduced SMT technology provided an impressive gain in terms of performance, achieving some 15 to 20 % increase. Not only in benchmarks, but also in real-world physics applications the gain was noticeable. In detail the ALICE framework benefitted from a 19 % increase in performance through the use of SMT technology. Running a mixture of several applications, such as test40 and the tbb benchmark on a single server did not effect the gain provided by SMT.

Enabling SMT increase the available cores in a Nehalem dual socket system, from 8 to 16. Running 16 processes at the same time requires the management of the available resources, made possible through the use of CPUset. Restricting the processes to CPUsets are ready for practical use and easy to manage. Using CPUsets can guarantee that jobs are executed with a defined amount of performance. The running processes have no influence on each other, as long as I/O performance or other subsystems are not the limiting factors. In our tests, the number of CPU cores increased permanently, but the other areas of the system, such as the hard disk performance were not scales. This can often lead to the different subsystems, other than the CPU, becoming the bottleneck of the system. All this should be taken into account if many processes are running simultaneously and generate a lot of I/O operations.

7 References

[1] http://www.tecchannel.de/pc_mobile/prozessoren/1752990/
intels_nehalem_enthuellt_stark_verfeinerte_core_mikroarchitektur/index.html
[2] http://www.intel.com/products/server/motherboards/s5520ur/s5520ur-overview.htm
[3] http://www.intel.com/products/desktop/motherboards/DX58SO/DX58SO-overview.htm
[4] Gyorgy Balazs, Sverre Jarp, Andrzej Nowak - "Is the Atom processor ready for High Energy Physics?", CERN 2008
[5] http://www.ddj.com/architect/215802084/

8 Appendix

8.1 Results of the energy measurements in detail

8.1.1 Results of the Nehalem CPUs

State	Idle		
Memory	Active Power [W]	Apparent Power [W]	Power Factor
L5520 SMT-off turbo-off			
24	119.029	133.191	0.894
L5520 SMT-off turbo-on			
24	120.732	134.408	0.898
L5520 SMT-on turbo-off			
24	129.721	143.291	0.905
L5520 SMT-on turbo-on			
24	130.916	144.482	0.906

E5540 SMT-off turbo-off			
24	117.232	131.366	0.892
E5540 SMT-off turbo-on			
24	117.221	131.29	0.893
E5540 SMT-on turbo-off			
24	129.266	142.823	0.905
E5540 SMT-on turbo-on			
24	129.422	142.96	0.905
E5540 SMT-off turbo-off SpeedStep=off			
24	130.809	144.296	0.907
E5540 SMT-on turbo-off SpeedStep=off			
24	148.379	160.966	0.922

X5570 SMT-off turbo-off			
24	122.413	136.423	0.897
X5570 SMT-off turbo-on			
24	118.701	132.742	0.894
X5570 SMT-on turbo-off			
24	133.288	146.609	0.909
X5570 SMT-on turbo-on			
24	133.136	146.288	0.910

State	Load		
Memory	Active Power [W]	Apparent Power [W]	Power Factor
L5520 SMT-off turbo-off			
24	252.591	262.944	0.961
L5520 SMT-off turbo-on			
24	262.687	272.719	0.963
L5520 SMT-on turbo-off			
24	275.113	285.066	0.965
L5520 SMT-on turbo-on			
24	273.54	283.524	0.965

E5540 SMT-off turbo-off			
24	292.483	302.031	0.968
E5540 SMT-off turbo-on			
24	309.231	318.48	0.971
E5540 SMT-on turbo-off			
24	316.216	325.459	0.972
E5540 SMT-on turbo-on			
24	322.952	332.142	0.972
E5540 SMT-off turbo-off SpeedStep=off			
24	317.111	326.371	0.972
E5540 SMT-on turbo-off SpeedStep=off			
24	319.316	328.556	0.972

X5570 SMT-off turbo-off			
24	325.632	334.837	0.972
X5570 SMT-off turbo-on			
24	359.091	367.979	0.976
X5570 SMT-on turbo-off			
24	356.214	365.092	0.976
X5570 SMT-on turbo-on			
24	372.953	381.667	0.977

8.1.2 Results of the Core i7 965

State	Idle		
Memory	Active Power [W]	Apparent Power [W]	Power Factor
Core i7 965 SMT-off turbo-off			
6	109.039	110.822	0.984
Core i7 965 SMT-off turbo-on			
6	109.366	111.231	0.983
Core i7 965 SMT-on turbo-off			
6	110.797	112.590	0.984
Core i7 965 SMT-on turbo-on			
6	111.295	113.089	0.984

State	Load		
Memory	Active Power [W]	Apparent Power [W]	Power Factor
Core i7 965 SMT-off turbo-off			
6	220.078	221.981	0.991
Core i7 965 SMT-off turbo-on			
6	238.422	240.228	0.992
Core i7 965 SMT-on turbo-off			
6	235.737	237.567	0.992
Core i7 965 SMT-on turbo-on			
6	259.089	260.795	0.993

8.2 Results of the performance measurements in detail

8.2.1 Results of the Nehalem CPUs [SPEC marks]

E4 12x2G SMT-off Turbo-off

Mode	Compiler	L5520	E5540	X5570
32 bit	**Gcc 4.1.2**	92.57	100.06	114.43
	Gcc 4.3.3	93.90	96.56	116.08
	Intel 11.0	104.44	107.03	129.44
64 bit	**Gcc 4.1.2**	106.86	108.90	136.90
	Gcc 4.3.3	110.28	110.57	134.43
	Intel 11.0	112.14	111.78	137.40

E4 12x2G SMT-off Turbo-on

Mode	Compiler	L5520	E5540	X5570
32 bit	**Gcc 4.1.2**	94.25	103.46	118.21
	Gcc 4.3.3	96.24	106.45	119.12
	Intel 11.0	107.36	116.38	133.18
64 bit	**Gcc 4.1.2**	112.5	122.43	138.33
	Gcc 4.3.3	111.67	122.50	141.43
	Intel 11.0	114.57	124.23	142.50

E4 12x2G SMT-on Turbo-off

Mode	Compiler	L5520	E5540	X5570
32 bit	**Gcc 4.1.2**	116.19	119.26	146.92
	Gcc 4.3.3	117.44	120.23	144.42
	Intel 11.0	133.44	135.99	165.09
64 bit	**Gcc 4.1.2**	139.04	138.70	173.86
	Gcc 4.3.3	137.19	139.39	166.10
	Intel 11.0	140.09	141.71	174.64

E4 12x2G SMT-on Turbo-on

Mode	Compiler	L5520	E5540	X5570
32 bit	**Gcc 4.1.2**	117.97	128.70	151.44
	Gcc 4.3.3	117.93	132.64	150.12
	Intel 11.0	134.17	139.13	167.06
64 bit	**Gcc 4.1.2**	143.57	152.10	172.18
	Gcc 4.3.3	139.35	150.86	173.32
	Intel 11.0	141.49	144.58	175.47

8.2.2 Results of the Core i7 965 [SPEC marks]

Mode	Compiler	3x2G SMT-off Turbo-off Core i7 965	3x2G SMT-off Turbo-on Core i7 965
32 bit	**Gcc 4.1.2**	61.52	63.37
	Gcc 4.3.3	62.17	64.44
	Intel 11.0	68.67	71.19
64 bit	**Gcc 4.1.2**	71.38	73.05
	Gcc 4.3.3	71.41	73.75
	Intel 11.0	73.05	75.65

Mode	Compiler	3x2G SMT-on Turbo-off Core i7 965	3x2G SMT-on Turbo-on Core i7 965
32 bit	**Gcc 4.1.2**	77.33	80.51
	Gcc 4.3.3	77.56	80.24
	Intel 11.0	86.14	91.05
64 bit	**Gcc 4.1.2**	90.90	92.59
	Gcc 4.3.3	92.37	91.64
	Intel 11.0	90.56	93.26