Evaluation of the Intel Sandy Bridge-EP server processor

Sverre Jarp, Alfio Lazzaro, Julien Leduc, Andrzej Nowak
CERN openlab, March 2012 – version 2.2

Executive Summary

In this paper we report on a set of benchmark results recently obtained by CERN openlab when comparing an 8-core "Sandy Bridge-EP" processor with Intel's previous microarchitecture, the "Westmere-EP". The Intel marketing names for these processors are "Xeon E5-2600 processor series" and "Xeon 5600 processor series", respectively. Both processors are produced in a 32nm process, and both platforms are dual-socket servers. Multiple benchmarks were used to get a good understanding of the performance of the new processor. We used both industry-standard benchmarks, such as SPEC2006, and specific High Energy Physics benchmarks, representing both simulation of physics detectors and data analysis of physics events.

Before summarizing the results we must stress the fact that benchmarking of modern processors is a very complex affair. One has to control (at least) the following features: processor frequency, overclocking via Turbo mode, the number of physical cores in use, the use of logical cores via Simultaneous Multi-Threading (SMT), the cache sizes available, the memory configuration installed, as well as the power configuration if throughput per watt is to be measured. Software must also be kept under control and we show that a change of operating system or compiler can lead to different results, as well. We have tried to do a good job of comparing like with like.

In summary, we obtained a performance improvement of 9 – 20% per core and 46 – 60% improvement across all cores available. We also found that Turbo mode has been improved. Vectorized applications get an additional performance boost given by the new AVX instructions. We were particularly impressed by the performance/Watt measurements where we obtained a 70% improvement (and even more when we modified the software). Intel has improved the thermal characteristics of the Sandy Bridge substantially and this was reflected in the measurements of idle power, but also in the measurements of a fully loaded system. Computer centers that are power constrained will, without doubt, appreciate the improvements in this important domain. By raising the bar to such a high level, Intel has set high expectations and we are keen to see whether the pace of improvement can be sustained for the 22 nm processors, namely the Ivy Bridge processor planned for 2013 and the Haswell for 2014.

Table of Contents

Introduction

Description of the processor

In this paper we compare the two Xeon EP generations produced in 32 nm silicon process. The first generation, "Westmere-EP" or "Xeon 5600" was produced about two years ago as a shrink of the 45 nm-based "Nehalem-EP" processor. The Westmere-EP contained up to 6 cores and 12 MB of L3 cache on the die. The Sandy Bridge-EP processor increases these features to 8 cores and 20 MB of L3 cache. Since both processors are produced in the same silicon process, the Sandy Bridge processor is necessarily quite a bit larger (435 mm^2) than the Westmere-EP processor (248 mm^2). The increase is about 75% which reflects well the L3 cache increase (67%), given that cache memory occupies by far the largest portion of the die. Nevertheless, the Thermal Design Power for the two generations is more or less unchanged, reflecting the considerable amount of effort that Intel is putting into optimizing the power consumption. The highest TDP for Westmere-EP was 130W and for Sandy Bridge-EP 135W. In the comparison we actually use several Westmere-EP processors, with multiple TDP values (and frequencies lower than the peak) but all our results are either calculated as performance/Watt or shown as frequency-scaled absolute performance, so, in our experience, this is a fair way of doing comparisons.

Both generations support Simultaneous Multithreading (SMT) that allows two threads or processes to be active on each core. Furthermore, both processors support Turbo-boost mode that allows the processor to increase its frequency when only a few of the cores are in use, while maintaining power within the designed envelope.

The major new architectural feature in the Sandy Bridge processor is undoubtedly the Advanced Vector eXtensions (AVX) that allow new Single Instruction Multiple Data (SIMD) operations to be performed on 256 bits of data. This is twice the width of Streaming SIMD Extensions (SSE) that we find in the Westmere-EP architecture and its predecessors. Consequently, this is a feature we have tested extensively in the Parallel Maximum Likelihood fitting benchmark, since it allows good usage of this SIMD feature.

Hardware configuration

The Sandy Bridge-EP processor evaluated in this paper is an E5-2680 running at 2.70GHz. The motherboard installed in our test system is an Intel S2600CP motherboard that supports up to 16 DDR3 memory DIMMs. This test system is equipped with 32 GB of memory (8 x 4 GB DIMMs 1333MHz Low Voltage), and a 1TB SATA hard drive.

Software configuration

The system is running 64-bit Scientific Linux CERN 6.2 (SLC6), based on Red Hat Enterprise Linux 6 (Server). The default SLC6 Linux kernel (version 2.6.32-220.el6) was used for all the measurements. Some SLC5 measurements where performed

and then, the system was running 64-bit Scientific Linux CERN 5.7 on the default kernel (version 2.6.18-274.3.1.el5).

Standard energy measurement procedure

Procedure and methodology

The standard energy measurement procedure is well-established in the CERN IT department for measuring the power consumption of any system that might be operated in the CERN Computing Center. Since this procedure was already thoroughly described in a previous openlab paper: "Evaluation of energy consumption and performance of Intel's Nehalem architecture", it is now included as an appendix.

Results

The system is equipped with one power supply (PSU). When conducting the tests without SMT the dual-socket system appears as having 16 cores in total, so that, according to the standard energy measurement procedure, the Load stress test consists of running 8 instances of *CPUBurn* along with 8 instances of *LAPACK*, (using 4 GB of memory each).

In a second phase, now with SMT enabled, the system was considered as a 32 core server, meaning that the Load stress test should be conducted by running 16 instances of *CPUBurn* along with 16 instances of *LAPACK*, (using 2 GB of memory each). The two tables below are comparing power measurements when the server is running two different versions of Scientific Linux CERN (SLC): SLC5.7 and a more recent SLC6.2:

Active Power		*Idle*	*Load*	*Standard measurement*
32 GB	SMT-off	140 W	390 W	340 W
	SMT-on	149 W	425 W	370 W

Table 1: Power consumption under SLC5

Active Power		*Idle*	*Load*	*Standard measurement*
32 GB	SMT-off	110 W	385 W	330 W
	SMT-on	113 W	414 W	354 W

Table 2: Power consumption under SLC6

A first look at these power consumption measurements shows an impressive difference of the overall system electrical consumption when comparing the "Idle"

and the “Load” state. Another interesting point is that with a more recent Linux distribution power management greatly improves, leading to 24% lower power consumption in idle mode SMT-on. This lower idle power consumption alone is able to diminish the standard measurement by 2% which is non negligible for a data center. Using the server internal power monitoring facilities, the two CPUs consume 27% more power running SLC5 than running SLC6.

Benchmarks

HEPSPEC2006

One of the important performance benchmarks in the IT industry is the SPEC CPU2006 benchmark from the SPEC Corporation[1]. This benchmark can be used to measure both individual CPU performance and the throughput rate of servers.

It is an industry standard benchmark suite, designed to stress a system’s processor, the caches and the memory subsystem. The benchmark suite is based on real user applications, and the source code is commercially available. A High Energy Physics (HEP) working group demonstrated a few years ago a good correlation between the SPEC results and High Energy Physics (HEP) applications when using the C++ subset of the tests from the SPEC CPU2006 benchmark suite [WLCG09]. As a result the HEP community has decided to use the C++ subset of SPEC2006, “HEPSPEC06” rather than internal benchmarks because SPEC2006 is readily available, and its results can be directly generated by computer manufacturers to evaluate the performance of a system aimed at running HEP applications.

In this set of benchmarks it was compiled with GCC 4.1.2 in 64-bit mode, the standard compiler available with SLC5 and the performance measurements were carried out with SMT enabled, and with Turbo mode off.

Since SPEC CPU2006 benchmark execution flow consists in serially launching several single threaded applications, several independent instances have to be launched simultaneously to evaluate the system scalability. This means that the HEPSPEC06 benchmark is indeed a rate measurement.

HEPSPEC06 results and X5670 Based “Westmere-EP” comparison

When comparing the two systems, frequency-scaled, the graph clearly shows an advantage to the most recent server generation. In the first phase, when the two systems count enough physical cores to accommodate the load, the E5-2680 core is offering at least 13% more HEPSPEC performance than the Westmere core. But when the CPU occupancy approaches its maximum: 12 cores for the Westmere-based system and 16 cores for the Sandy Bridge, this advantage reaches a peak at 20% additional HEPSPEC performance per core for the E5 core.

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

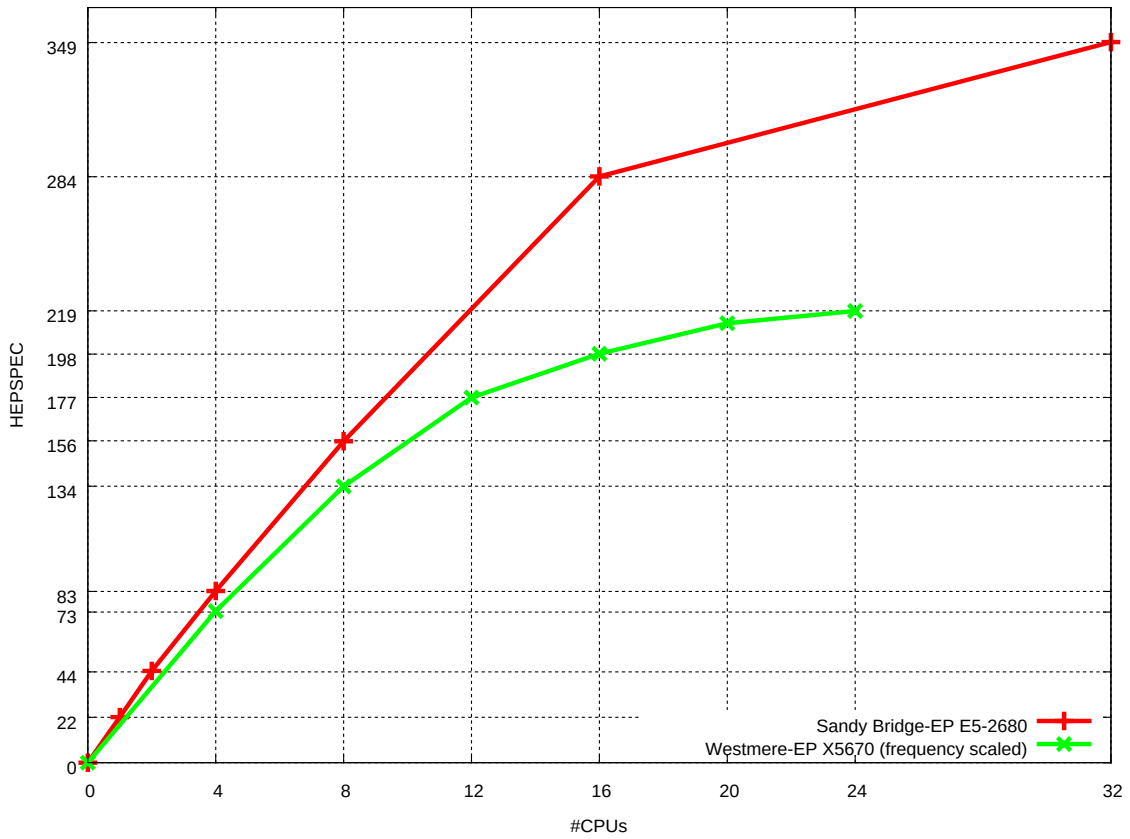

Figure 1: HEPSPEC06 performance comparison Turbo-off (higher is better)

This can be explained by the fact that the E5 scalability is better than the 5600 series CPU in this initial phase: according to our previous Nehalem/Westmere HEPSPEC comparison [OPL10], an inflexion was observed in Westmere HEPSPEC scalability between 8 and 12 cores, and this is not the case for the Sandy Bridge.

SMT advantage

The gain produced by SMT can be computed by comparing the HEPSPEC06 results for 16 and 32 cores for the Sandy Bridge, and for 12 and 24 cores for the Westmere. In the case of the Westmere, the gain is 23.7% and for the Sandy Bridge, the SMT gain is 22.8%. This shows that SMT is a well-established technology that steadily delivers around 23% of additional HEPSPEC performance on Intel Xeon CPUs.

Platform efficiency

Given that the platform power consumption and HEPSPEC06 measurements have been measured, a derived measurement can now be obtained. This measurement, which is very relevant for a computer center, is the power efficiency of the platform in terms of HEPSPEC per Watt.

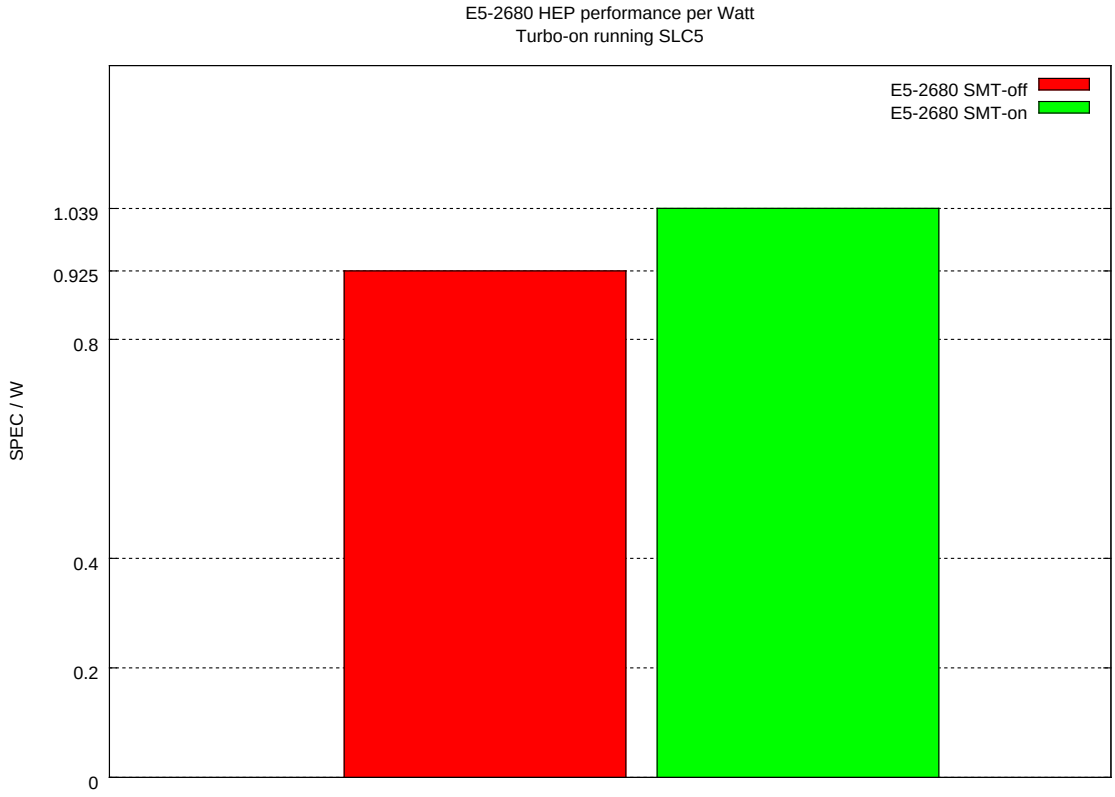

Figure 2: Efficiency of the E5-2680 Turbo-on under SLC5

This figure shows that SMT offers an 11% boost in terms of efficiency with Turbo-on running SLC5. SMT additional power consumption is around 10% when Turbo is on, and this additional power reduces some of the SMT gain when looking at power efficiency. We highlight the fact that the E5-2680 is able to cross the 1 HEPSPEC/W line which can be considered particularly impressive.

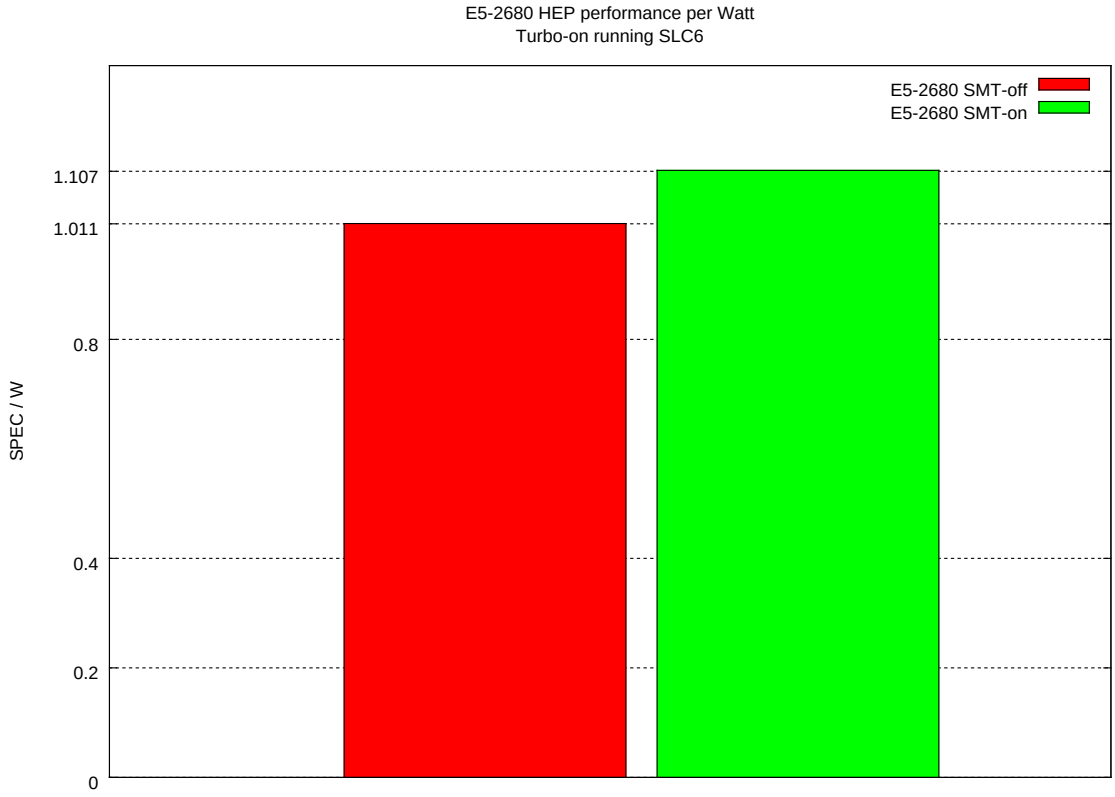

Figure 3: Efficiency of the E5-2680 Turbo-on under SLC6

When running SLC6, the efficiency gets an additional 6% boost with SMT on, reaching 1.107 HEPSPEC/W. Some of the performance is due to the compiler change: going from gcc 4.1.2 for SLC5 to gcc 4.4.6 for SLC6: this change provides 19% increase of HEPSPEC SMT-off and 8% SMT-on.

Comparing the results when using SLC5, we observe that the microarchitecture change from the X5600 series to the E5-2600 series allowed a massive improvement in terms of platform efficiency: going from 0.611 HEPSPEC/W to 1.039 HEPSPEC/W, which is a 70% increase for efficiency. This is, for instance, twice the efficiency improvement from the previous microarchitecture change that occurred between Harpertown and Nehalem.

Multi-threaded Geant4 prototype

Geant4 is one of the principal toolkits used in Large Hadron Collider (LHC) simulation. Its primary purpose is to simulate the passage of particles through matter. This type of simulation is a CPU-intensive part of a bigger overall framework used to process the events coming from the detectors. It is representative to an extent of real life workloads and can constitute a substantial portion of the CPU time of the Worldwide LHC Computing Grid. Since HEP has always been blessed with parallelism inherent in the processing model, it is natural to try to utilize modern multi-core systems by

converging towards multi-threaded event processing. The Geant4 prototype discussed here is one of the key steps in that direction.

Based around Geant4, this suite has been updated to support multi-threading by two Northeastern University researchers: Xin Dong and Gene Cooperman. The example used in this case is "ParFullCMSmt", a parallelized version of the "ParFullCMS" program, which represents a simulation close in properties to what the CERN CMS experiment is using in production. Thus, this is an example of a real-world application in use at CERN.

One of the key metrics of a parallel application and the underlying parallel platform is scalability. The tests described in this chapter focus on the scalability of the multi-threaded Geant4 prototype, which is defined as throughput. In principle, a single-threaded process has a specified average time it takes to process 100 events. Thus we measure the influence of the number of processes (and implicitly the number of processing units engaged) on the processing time of 100 events. In an ideal case, as more processes with more work are added, one would expect the throughput to grow proportionally to the added resources, and so the processing time of 100 events would remain unchanged (per thread). Another key metric considered in this case is "efficiency", which is defined as the scaling of the software relative to the single thread runtime of the parallelized code, confronted with ideal scaling determined by the product of the core count and the single thread throughput. In cases where multiple hardware threads are being used, perfect scaling is defined by the maximum core count of the system (16).

Technical test setup

The threads were pinned to the cores running them, and the throughput defining factor was the average time it takes to process one 300 GeV pi- event in a predefined geometry based on the real world Compact Muon Solenoid (CMS) detector. Although the Turbo feature remained switched off, the system was SMT-enabled – which means that the hardware threading feature was activated and used during the tests. Thus, if there were no more physical cores available, the jobs would be pinned to hardware threads, still maximizing the amount of physical cores used. The tested framework was based on Geant4 4.9.2p01, CLHEP 2.0.4.2 and XercesC 2.8.0, and was compiled using the GCC 4.3.3 compiler.

Scalability testing

The tests showed stable efficiency figures up to the full physical core count. When running with 16 software threads on 16 cores, the scaling factor was 15.8x, which means that the setup was 98% efficient. The figure matches good results observed on previous Intel processors, especially from the "Westmere" family. Detailed scalability data for intermediate points reveals good scaling, as expected, all the way between 1 and 16 processes:

- 2x for 2 processes (100% efficiency)
- 4x for 4 processes (100% efficiency)
- 8x for 8 processes (100% efficiency)
- 12x for 12 processes (100% efficiency)

There was no noticeable efficiency drop.

The graph below (Figure 4) shows the total simulation time curve in blue and the efficiency (scalability) curve in green. Towards its end, the efficiency curve surpasses 100%, since for thread counts higher than 16 expected scalability is fixed to 16x. Thus a final value of 125% indicates that the system loaded with 32 threads of the benchmark yields 25% more throughput than a perfectly scaled serial version on 16 physical cores.

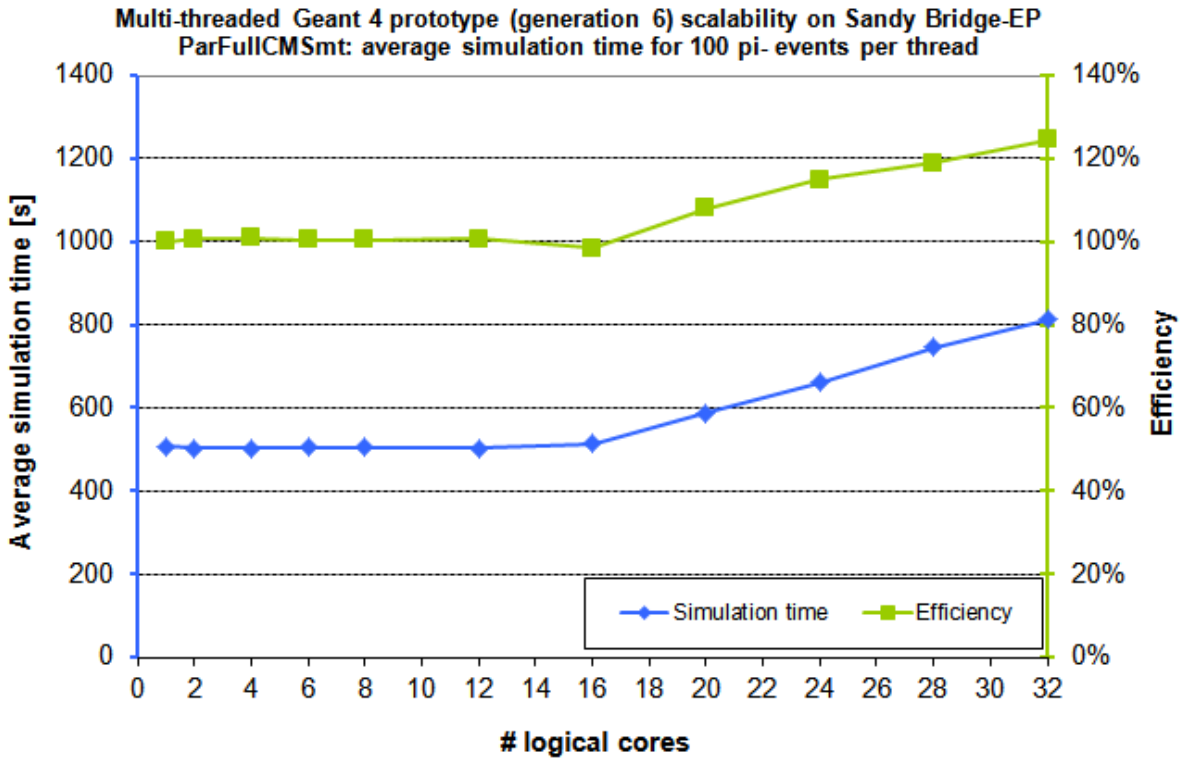

Figure 4: ParFullCMSmt scalability on Sandy Bridge-EP

Hyper-threading advantage

The advantage of running with SMT was already 8% for 20 hardware threads, suggesting that the run with 16 threads (full core count) might be slightly suboptimal. The SMT gains later were 15% with 24 hardware threads and finally 25% with all hardware threads engaged. One should note that this extra performance is traded in for a penalty in memory usage, as the number of software threads is twice the one in the case of 16 cores.

This advantage is comparable to previously tested dual-socket systems, even if slightly higher. It could testify to the slightly increased efficiency of SMT, but also to a potential minor sub optimality of the benchmark runs with all physical cores loaded (16 threads).

L5640 based "Westmere-EP" comparison

When compared to a Xeon L5640 based Westmere-EP platform tested earlier, the probed Sandy Bridge-EP system performs better in terms of frequency scaled performance, and also delivers a substantial 33% increase in core count. The overall advantage of the Sandy Bridge-EP solution over the comparable Westmere-EP platform was established to be 46% at full load, which is when using 32 threads for the Sandy Bridge, and 24 threads for the Westmere.

An 8-9% average frequency-scaled advantage was noticed when running with up to 12 jobs to match the physical core count of the Westmere-EP system. This is attributed to the on-die innovations in the Sandy Bridge microarchitecture.

The product of the core count increase (1.333) and the average microarchitectural advantage (1.09) yields a 45% expected improvement over the previous platform at full load, and the measured 46% improvement is within 0.5% of that goal (1.46 / 1.45 = 1.005). This means that the tested Sandy Bridge-EP solution continues the tradition of highly efficient dual socket systems based on the Intel microarchitecture, delivering scalable and predictable performance.

Parallel Maximum Likelihood fit

The HEP community makes large use of many complex data analysis techniques, like maximum likelihood fits, neural networks, and boosted decision trees [COW09, STAT01]. These techniques are employed for a better discrimination between interesting events with respect to the total events collected by the physics detectors, in order to discover possible new physics phenomena [PHYS08]. The increase of the sample sizes and use of complex data analysis models require high CPU performance for the execution of the data analysis software applications. The execution can be speeded up by having recourse to parallel implementations of the data analysis algorithms, i.e. strong scaling of the parallel applications. Traditionally all software for data analysis developed in HEP do not use parallelism. However, with the introduction of multi-core CPUs, an effort for parallelization has been started in recent years. The benchmark used here is based on an unbinned maximum likelihood data analysis application [COW08]. The code has been developed by CERN openlab, and it represents a prototype of the RooFit package (package inside the ROOT software framework developed at CERN), generally used in the HEP for maximum likelihood fits [ROF06]. The prototype makes use of an optimized algorithm with respect to the algorithm used in RooFit for the likelihood evaluation [JAR11a]. The parallelization of the implementation of the algorithm is based on OpenMP. Because of the design of the algorithm, the initial implementation was subject to a high rate of last-level cache (LLC) load misses, i.e. shared L3 cache, and a significant OpenMP overhead that limited the scalability. These bottlenecks have been mitigated by an improved version of the code, described in Ref. [JAR11b]. In this new case, the OpenMP overhead becomes negligible (<1% when running with high number of threads). Then the high rate of LLC misses is reduced by means of splitting the data in blocks, achieving a better overlap between computation and memory accesses.

Clearly the application will benefit from systems with a bigger L3 shared cache size. Furthermore, using a scattered affinity topology maximizes the cache memory available per thread, i.e. threads are bound to cores of CPUs on different sockets before filling the cores of a given CPU. For example, running with 4 threads on the dual-socket systems means 2 threads per CPU (instead of 4 threads on the same CPU). Note that the application takes in account NUMA effects. Considering these optimizations, the application is expected to scale close to the theoretical scalability predicted by the Amdahl's law.

The likelihood function definition is taken from the data analysis performed at the *BaBar* experiment [BBR09]. Thus, this is an example of a real-world application in use in the HEP community. The maximization of the likelihood function, or the equivalent minimization of the negative logarithm of the likelihood function (negative log-likelihood), is performed by using the MINUIT package [MIN72]. The main algorithm in this package, MIGRAD, searches for the minimum of a function using the gradient information [NUM07]. Several evaluations of the negative log-likelihood are needed before reaching the minimum of the function, so it becomes important to speed up the evaluations. The implementation guarantees that the number of evaluations and the results do not depend on the number of executed parallel threads, i.e. the workload is fixed between the parallel executions. All calculations are performed in double precision. Input data and results are organized in vectors, which are shared across the threads so that there is a small increase of the memory footprint with the number of threads. Input data is composed by 500,000 entries per 3 observables, for a total of about 12MB. Results are stored in 29 vectors of 500,000 values, i.e. about 110MB, which are combined to get the value of the negative log-likelihood function. Operations are performed on all elements by means of loops, so there are 500,000 iterations in total per loop. Loops are auto-vectorized by the Intel compiler and parallelized. The events are organized in blocks for the reason described before. A heuristic approach is followed to decide the dimensions of the blocks, which depend on the number of parallel threads. They can be put in relation with the cache size available per thread, so the dimensions decrease accordingly with the number of threads. In particular the block dimensions allow having the fastest execution for a given parallel execution. Table 3 reports the block sizes used in the tests.

# Threads	*Block size (# events)*
1 and 2	10,000
4	5,000
6	3,000
8, 10 and 12	2,000
16, 24 and 32	1,000

Table 3 Block dimensions. Note that 24 and 32 threads use SMT.

First of all we look at the speed-up given by the vectorization. In this case the speed-up is defined as the ratio between the execution times of the non-vectorized and vectorized code. Note that Sandy Bridge microarchitecture introduces a new set of SIMD instructions, namely Advanced Vector Extensions (AVX) instructions, which

enables 256-bit vector registers. These instructions extend the SSE ones, which use 128-bit vector registers. Secondly we look at the speed-up given by the Turbo mode, so, in this case, the speed-up is defined by the ratio between the execution time with Turbo mode off and Turbo mode on. The comparison is done when running the application in sequential (a single thread execution) and in parallel. The Turbo mode is expected to give the maximum contribution when the system is loaded with a low number of threads per CPU. However, a small overall benefit is expected also when running with fully loaded parallel threads, just because the Turbo mode speeds up the application during its sequential portion. Finally we discuss the performance of the sequential and parallel executions and we show the scalability results, including the SMT benefit. All results are compared with the reference Westmere-EP system, which is frequency-scaled (nominal frequency) to allow a direct comparison of the performances between the two systems.

Technical setup

The systems have Linux version: Scientific Linux CERN SLC release 6.2, based on Red Hat Enterprise Linux release 6.2. Code is compiled with ICC v12.1.0. The reference Westmere-EP system is a dual-socket system with Intel Xeon CPU X5650 at 2.67GHz (12 physical cores in total, 12MB L3 cache per CPU). The systems are SMT-enabled, which means that the hardware threading feature was activated and used during the tests. Thus, if there are no more physical cores available, the jobs are pinned to hardware threads by requiring 2 threads per core.

Timings are obtained from the average over three consecutive runs of the application (errors are <0.5%). The sequential execution time of the application is 487 seconds when running on the Sandy Bridge without vectorization and Turbo mode off (the slowest execution). It is worth to underline that the application is floating-point intensive; in particular the execution of the exponential function takes about 60% of the total execution time.

Vectorization results

We compile the code in 3 different configurations by using the ICC flags:

1. `-no-vec`
2. `-msse3`
3. `-xAVX`

We found that the speed-up results do not significantly depend on the number of threads and the Turbo mode (standard deviation is 0.03x). The average speed-up results are shown in Table 4. The Westmere-EP system has the same speed-up when using the `-msse3` configuration.

	-no-vec	-msse3
-msse3	1.78x	
-xAVX	1.99x	1.12x

Table 4 Average speed-up results for different vectorization configurations (given by the indicated compiler flags) on the Sandy Bridge-EP system. The results are obtained from the ratio between execution times for the configurations in the columns and the corresponding configurations in the rows.

A detailed analysis of speed-up result value between the `-msse3` and `-xAVX` configurations (1.12x) shows that the vectorized loops that contain exponential function operations are faster executed in the `-xAVX` configuration. For these loops the Intel compiler uses the functions `_svml_exp2.R` and `_svml_exp4.R` for SSE and AVX configurations, respectively. Although the total time exclusively spent in the executions of the AVX function is 1.61x faster than the SSE case, the divide and square root operations and the and loads/stores of data, which are executed inside the same loops with the exponential function, limit the speed-up of the loops to values between 1.14x and 1.17x (vectorized AVX versions of divide and square root operations take the same time of the corresponding SSE versions). Surprisingly, also the execution of loops with evaluation of pure polynomials inside does not go faster with AVX, as it would be expected, because of the loads/stores. All these effects explain the smaller value of the speed-up for `-msse3` versus `-xAVX` with respect to `-no-vec` versus `-msse3` configurations.

Turbo mode results

Speed-up results obtained from the comparison of the execution times with Turbo mode on and off are shown in Figure 5.

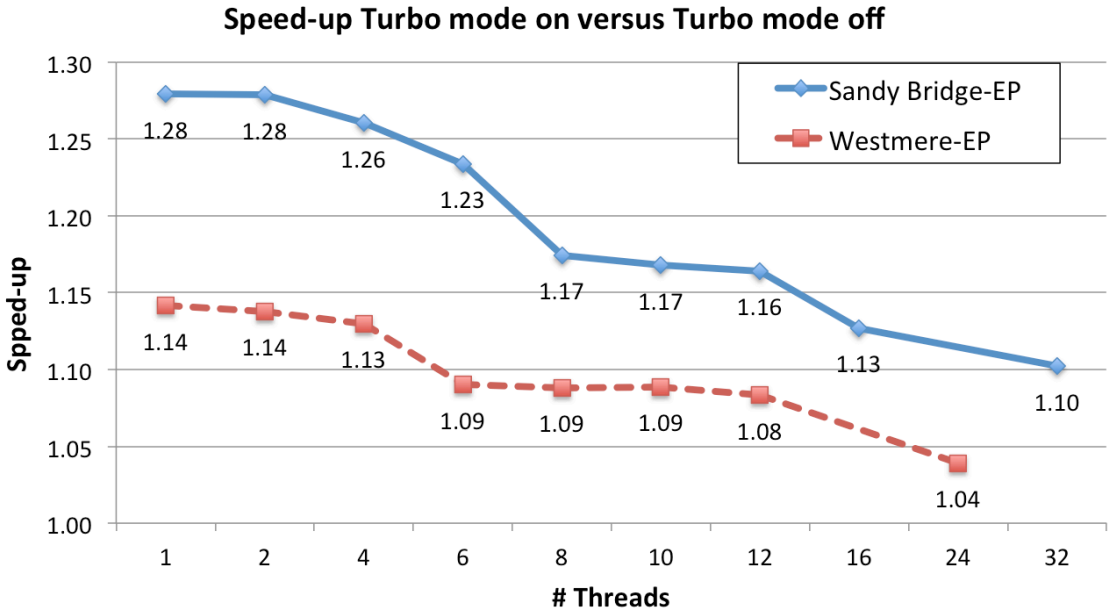

Figure 5 Speed-up ratios obtained from the comparison of the execution times with Turbo mode off and Turbo mode on.

We found that the effect of the Turbo mode does not depend on the vectorization mode. With two threads (one thread per CPU) the speed-up of the Turbo mode is compatible with the increase in frequency from the nominal value of 2.70GHz to 3.46GHz. For the Westmere-EP system the effect on Turbo mode is significant less, increasing from the nominal 2.67GHz to 3.04GHz. The benefit of the Turbo mode decreases when more parallel threads are executed. With 16 threads on the Sandy Bridge-EP the speed-up drops to 1.13x, with a variation of 0.15x with respect to the single thread case. For the corresponding case (12 threads) for the Westmere-EP the variation is 0.06x. When also SMT is used (fully loaded system), Turbo mode gives still a contribution for the Sandy Bridge-EP system (1.10x), which is higher than the corresponding case of the Westmere-EP system (1.04x). To conclude, the Turbo mode behaves much better on the Sandy Bridge-EP system with respect to the Westmere-EP, reaching higher speed-up for any number of loaded threads.

Performance comparison and scalability results

We compare the execution time of the application compiled with AVX vectorization on the Sandy Bridge-EP system with respect to the execution time when running with SSE vectorization on the Westmere-EP. Both systems have Turbo mode on. Comparing the corresponding runs executed with the same number of threads on both systems, i.e. between 1 and 12 threads, we obtain that the Sandy Bridge-EP is in average 1.49x faster (standard deviation is 0.03x). Removing the speed-up due to the AVX vectorization and Turbo mode on, we obtain that the Sandy Bridge-EP single core performs 1.19x faster than the Westemere-EP one. The same factor can be simply obtained from the comparison of the execution time of the runs with SSE vectorization and Turbo mode off on both systems. Analyzing this comparison we derive that the speed-up is mainly due to a faster execution of exponential functions (1.16x) and square root operations (1.54x). A small benefit comes also from the larger L3 cache available on the Sandy Bridge-EP system. Then we compare the fully loaded Westmere-EP and Sandy-Bridge systems with and without SMT, i.e. 12 versus 16 and 24 versus 32 threads, respectively. We obtain that the overall gain in performance between the two systems is 1.91x, without SMT, and 1.86x, with SMT. Note that the small decrement in the comparison of the overall performance is due to the higher number of threads running on the Sandy Bridge-EP system that limits the scalability. Also in this case we can compute the speed-up eliminating the effects of AVX vectorization and Turbo mode, which is about 1.60x. We obtain that this number is in agreement with the equivalent single core speed-up value (1.19x) scaled by the different core counts of the systems.

We perform scalability tests using the AVX vectorization configuration and Turbo mode off. In these tests the speed-up is defined as the ratio between the execution times spent by the application running with a single thread and with a certain number of threads in parallel. The fraction of the execution time spent in code that we can execute in parallel is 99.7%. We should underline that because Turbo mode contributes mainly when running with a low number of threads per CPU, it worsens the scalability for a high number of threads. This is the reason why we run the tests with Turbo mode off. We show the scalability results in Figure 6. We can clearly see

that the scalability is very close to the theoretical expectation. The application scales slightly worse on the Westmere-EP system. Analysis of the difference shows that this is a consequence of the smaller L3 cache size.

Finally we determine the contribution given by using SMT. This contribution is obtained from the ratio of the execution times of the application when running with maximum number of threads without and with SMT, i.e. 16 and 32 threads. The results do not depend on the vectorization, while Turbo mode gives a small impact: 1.29x and 1.26x for Turbo mode off and on, respectively. Corresponding values for Westmere-EP are slightly higher: 1.35x and 1.30x. This can be attributed to the higher numbers of threads involved in the Sandy Bridge-EP case.

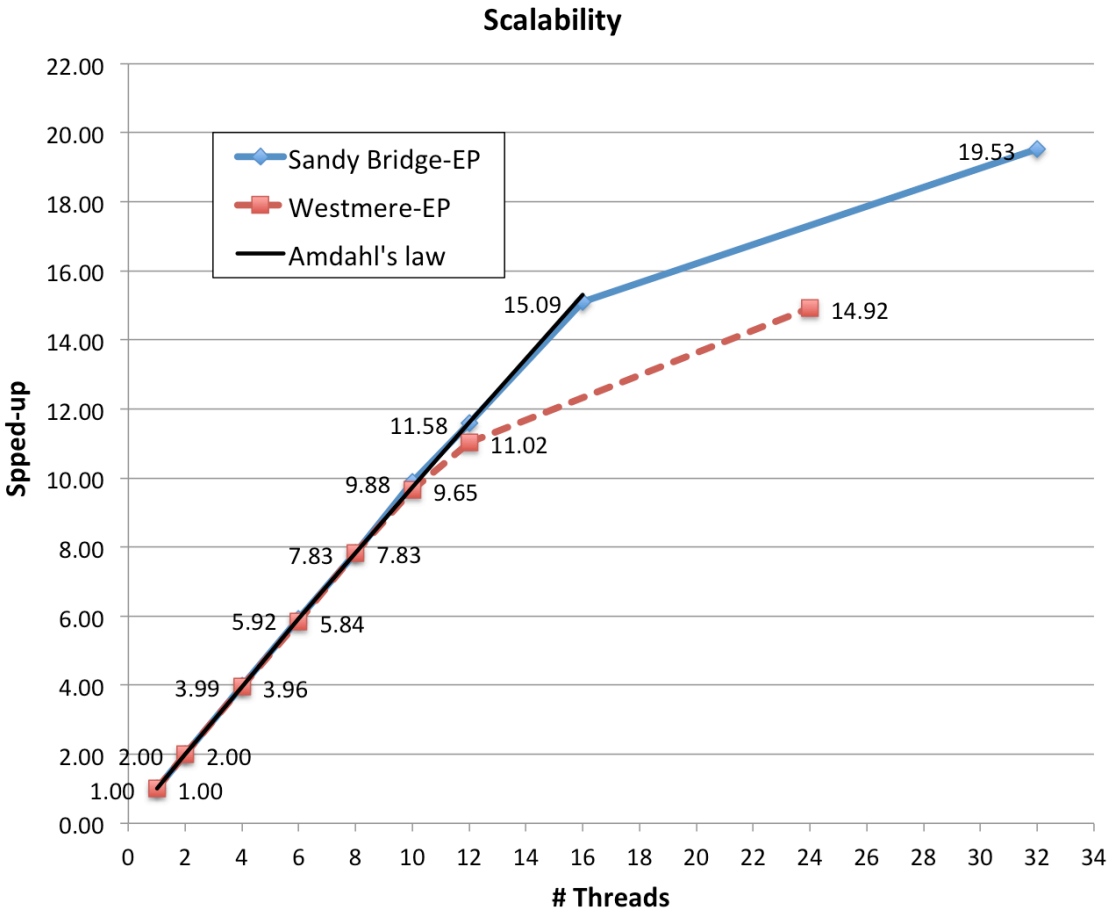

Figure 6 Scalability results. Data labels on the left (right) of the markers are for the Sandy Bridge-EP (Westmere-EP) system. The solid black line without markers is the theoretical speed-up obtained by Amdahl's law with a parallel fraction of 99.7%.

Conclusions and summary

As already discussed, the thermal performance of the mature 32 nm process has allowed Intel to include two additional cores per chip with respect to the previous Westmere-EP generation.

This is “only”a 33% increase, so, for all of benchmarks, it can be concluded that the improvements are always consistently better than what could be anticipated by looking simply at the core count increase. The reason for the additional performance can therefore be found in the combination of improved thermal management inside each core, a more performant microarchitecture, and the bigger L3 cache.

In the HEPSPEC tests we obtained a 20% performance increase per core when both servers were fully loaded. This represents a 60% increase across all available cores in the two systems. The gain of 23% with SMT is practically the same for both.

The calculated performance/Watt improvement confirmed that Intel has put a significant effort into optimizing the power consumption of the Sandy Bridge EP processor. We measured an improvement of 70% compared to Westmere-EP with SLC5 and gcc 4.1.2 and we noticed that, in particular, idle power is much lower in the newer generation. When we moved to SLC6 and gcc 4.4.6 we observed that the power consumption of the Sandy Bridge server was reduced by 5% and the throughput rate increased by 3%. It is therefore true to say that we have not seen such an impressive performance/Watt improvement since the days when core counts increased geometrically (1 → 2 → 4). The CERN Computing Center is power limited, just like most of the Tier-1 centers in W-LCG, so such improvements are always very welcome.

When we tested weak scaling using the Multithreaded Geant4 benchmark we found that performance increased by 46% when using all SMT cores on both servers. This is somewhat lower than the HEPSPEC number, possibly because Geant4 behaves well inside the L3 cache and did not see any benefit in the increased size per core. The SMT benefit was 25%.

When running the Parallel Maximum Likelihood fitting benchmark, which has a fixed problem size enforcing a strong scaling behavior, we obtained a 19% performance increase per core when using SSE on both servers. This corresponds to a 59% increased across all available cores – very similar to the improvement seen by the HEPSPEC results. The pleasant surprise with this benchmark was the fact that the Intel 12.1 compiler allows a further 12% gain when using autovectorisation with AVX. This is quite a bit below the theoretical gain of 100% but it has to be remembered that mathematical functions, such as divide and square root, have roughly the same throughput as on Westmere-EP.

In conclusion we confirm that the Sandy Bridge EP processor is a significant improvement compared to the previous generation. The raw performance improvement is in the range of 1.46 to 1.60 whereas the performance/Watt is 1.70 and even 1.83 when more recent software is used on Sandy Bridge. Especially the latter should be of great interest to power-constrained computing centers. We also found that Turbo mode has been improved. With this new generation Intel has allowed expectations to be set at a high level and we are keen to see whether the

pace of improvement can be sustained for the 22 nm processors, namely the Ivy Bridge processor planned for 2013 and the Haswell for 2014.

References

WLCG09	Multiple authors: *Transition to a new CPU benchmarking unit for the WLCG* (2009)
OLP10	S. Jarp *et al.*: *Evaluation of the Intel Westmere-EP server processor*, CERN openlab (2010). EPRINT: CERN-IT-Note-2011-004
COW98	G. Cowan: *Statistical Data Analysis*, Clarendon Press, Oxford (1998)
STAT01	J. Friedman, T. Hastie and R. Tibshirani: *The Elements of Statistical Learning*, Springer (2001)
PHYS08	G. Kane, A. Pierce: *Perspectives on LHC Physics*, World Scientific (2008)
ROF09	W. Verkerke and D. Kirkby: *The RooFit Toolkit for Data Modeling*, proceedings of PHYSTAT05, Imperial College Press (2006)
JAR11a	S. Jarp *et al.*: *Evaluation of Likelihood Functions for Data Analysis on Graphics Processing Units*, ipdpsw, pp. 1349--1358, 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum (2011). EPRINT: CERN-IT-2011-010
JAR11b	S. Jarp *et al.*: *Parallel Likelihood Function Evaluation on Heterogeneous Many-core Systems*, proceeding of International Conference on Parallel Computing, Ghent, Belgium (2011). EPRINT: CERN-IT-2011-012
BBR09	B. Aubert *et al.*: B *meson decays to charmless meson pairs containing η or η' mesons*, Phys. Rev. D80, 112002 (2009)
MIN72	F. James: *MINUIT - Function Minimization and Error Analysis*, CERN Program Library Long Writeup D506 (1972)
NUM07	W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery: *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press (2007)

Appendix A - standard energy measurement procedure

Measurement equipment

For the measurements a high precision power analyzer with several channels is required, since it must be able to measure the power consumption of any system from a simple UP system, with a single power supply unit (PSU) to a large server equipped with 4 PSUs.

To this extend a ZES-Zimmer LMG450 power analyzer is used. It allows the measurement of common power electronics. It has an accuracy of 0.1% and allows the measurement of four channels at the same time, and thanks to its convenient RS232 port, it can be linked to a PC to sample the values on the 4 channels, as shown on Figure 7.

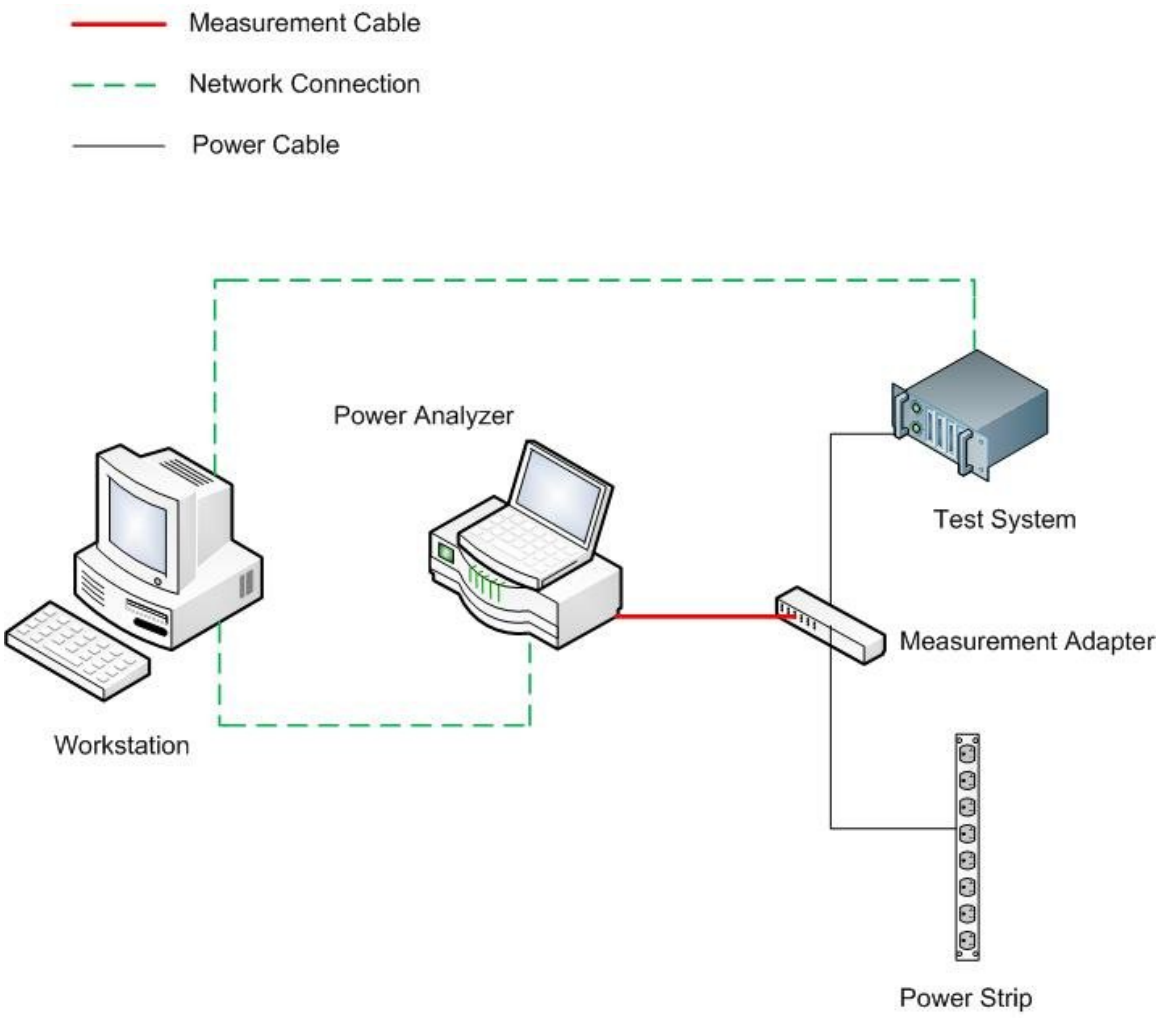

Figure 7: Power test setup

Three units are measured for each channel:

- *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 energy in kilowatt hours 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 = active power/apparent power

If the system includes several PSUs the Active Power must be summed on all the channels in use to compute the total Active Power of the system, for the two stress conditions.

LAPACK/CPUBurn

Those two tools are used to stress the evaluated systems, providing a reproducible load for any type of server:

1. *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 and is easy to adapt to fit the needs.
2. *CPUBurn* was originally written as a tool for overclockers, so that they can stress the overclocked CPUs, and check if they are stable. 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 from the CPU.

Standard energy measurement

The standard energy measurement is a combination of the Active power measured under two different stress conditions:

1. *Idle*: the system is booted with the Operating System and it does nothing.
2. *Load*: the system is running CPUBURN on half of the cores, and LAPACK on all the other cores, using all the installed memory.

An example to stress a system counting 8 cores and 12 GB of memory for the Load condition, would imply to run 4 instances of CPUBurn along with 4 instances of LAPACK each consuming 3 GB of memory.

According to that, the standard energy measurement is a mix of the active power under the Idle condition, accounting for 20%, and the active power under Load condition, accounting for 80%.