Platform™

Optimizing the performance of Intel's XEON 5500 architecture with Platform LSF®

TECHNOLOGY WHITEPAPER

AUTHORS:

Robert Stober
Platform Computing
rstober@Platform.com

Håkon Bugge
Platform Computing
hbugge@Platform.com

Date: May 8th, 2009

Contents

Abstract

The purpose of this paper is to show that innovations in Intel's processor architecture, and advances in Platform LSF in terms of processor binding capability, work together to deliver significant gains in application performance. Processor binding functionality was first built into Platform LSF with the 7.0 release, and has been enhanced with each new update of the product.

This paper begins with a brief discussion of the XEON 5500's apparent overcoming of the downward trend in processor speeds caused by memory bandwidth limitations. This is followed by a comparative study of Intel's two server implementations, Nehalem (XEON 5500) and Xeon 5400. Since XEON 5500 represents the first commercial implementation of these innovations, it is of great importance to understand the performance implications. A combination of micro-benchmarks and HPC application workload has been used in the evaluation. The findings indicate that XEON 5500 delivers a quantum leap in performance compared to its predecessor and that the new hardware multi-threading improves performance on HPC workloads.

The final part of the paper looks at benchmarks performed by Intel and third party organizations which show that the XEON 5500 processor is more than 50% faster than the previous generation. Performance increases of this magnitude were documented in applications across the board. This section of the paper also demonstrates that running applications on a Platform LSF cluster combined with the XEON 5500 delivers a statistically significant performance improvement. Furthermore, running jobs on XEON 5500 micro architecture in combination with Platform LSF processor binding provides an additional 12% gain in speed.

Part 1: Introduction – A Welcome Reversal

The XEON 5500 family of processors launched by Intel in the fall of 2008 represents a fundamentally different processor from its predecessors. XEON 5500 is more of a revolutionary step for Intel rather than an evolution of the technology used in the previous generation. XEON 5500 is a new micro-architecture that departs from Intel's long tradition of a shared memory controller. XEON 5500 implements the first instantiation of Intel's QuickPath Interconnect, and contains a new implementation of hardware multi-threading and an integrated memory controller. These innovations will likely be included in Intel CPUs for years to come.

Ever since integrated circuits were invented engineers have been finding ways to shrink them. Their never ending goal is to put more transistors on a chip and to pack them closer together resulting in higher frequencies and lower power consumption. In the past, smaller geometries and faster switching meant faster processors. But today, they imply more processing cores per CPU. Quad core chips are now the standard with six and eight core processors on the horizon. The memory sub-system has historically never been able to keep pace with the gains being made in processors, as memory bandwidth has only increased ~11% per year compared with the transistor budget of ~55% year.

With the XEON 5500, Intel has taken a major step towards addressing this lag, by creating a processor with a new micro-architecture which implements multi-threading and an integrated memory controller. The advent of the XEON 5500 has not only halted the downward trend (Figure 1) in Intel processor speeds, but has in fact reversed it (Figure 2). This bodes well not only for everyone that deploys the new processor, but especially those data centers that run compute- or data-intensive HPC applications.

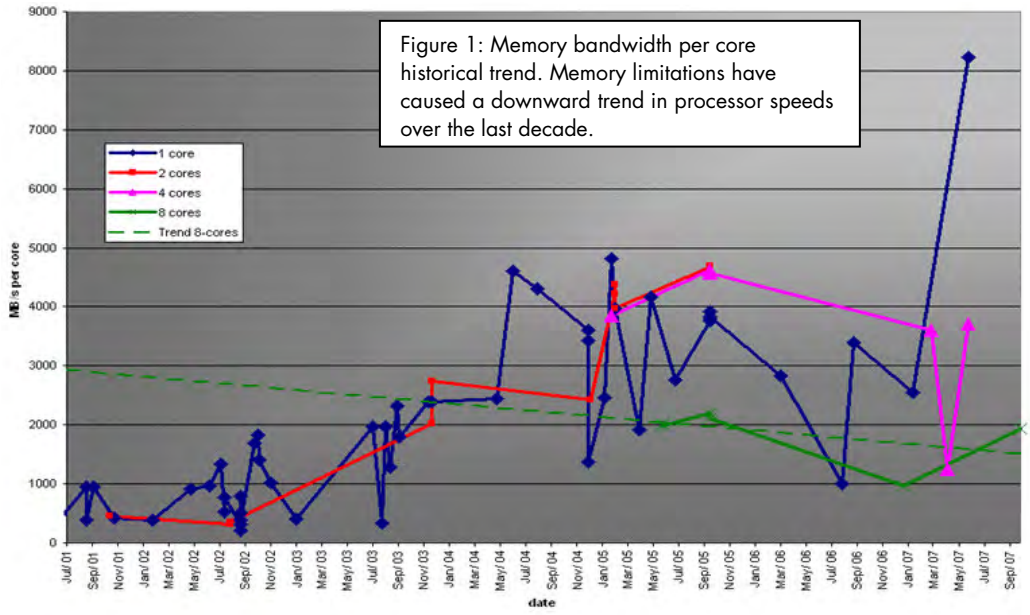
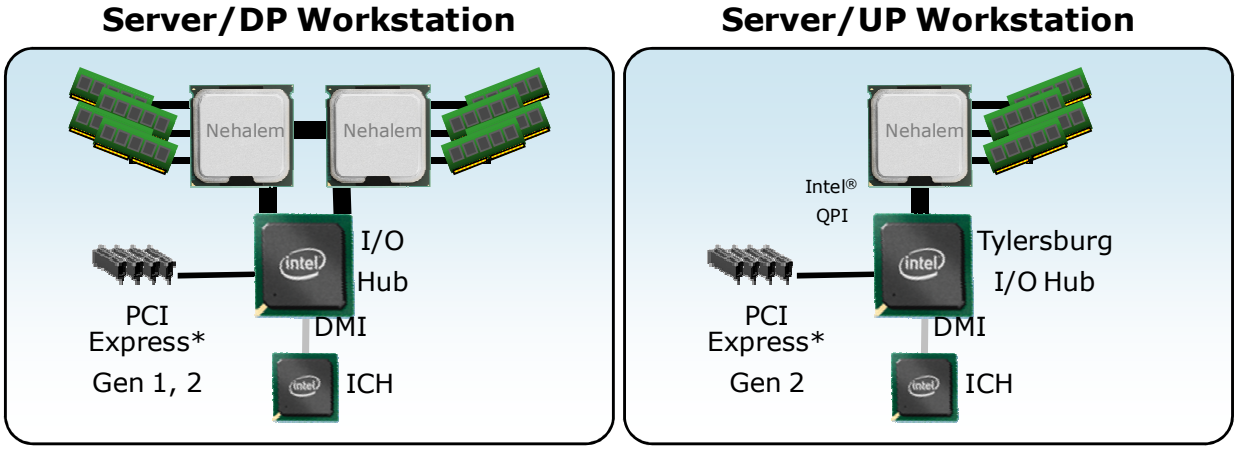


Figure 1: Memory bandwidth per core historical trend. Memory limitations have caused a downward trend in processor speeds over the last decade.

The XEON 5500 Micro-Architecture

With its emphasis on parallelism and bandwidth, XEON 5500 represents a major shift in strategy. The core micro architecture forms a set of building blocks that can be packaged and repackaged for different market segments from desktops, to servers, to notebooks. The diagram below shows a graphical representation of the set-ups of both a dual and a single processor.



Intel® QuickPath Interconnect

- New point to point interconnect
- 2 links per CPU socket
- Up to 25.6 Gb/sec total bandwidth/link

Integrated DDR3 Memory Controller

- 3 channels per processor
- Massive amounts of bandwidth
- Significant memory latency reduction

Huge Latency Decrease and Bandwidth Increase Over Prior Generation

The integrated memory controller is one of the key technology enhancements in the XEON 5500. It reduces latency and supports three memory channels at up to 1066MHz[1]. In most servers, the CPU talks to the north bridge, and the north bridge talks to the RAM. This causes one of the bigger problems in cutting-edge CPUs – clock cycles are sometimes wasted just "waiting" for data from the RAM. In the XEON 5500 the CPU skips the north bridge and talks directly to the RAM. The resulting speed increase is not trivial. Removing this memory bottleneck means that the CPU is spending more time working and less time waiting (high utilization). Another noticeable gain is achieved by transitioning away from exotic FB-DIMMs to conventional DDR3 RAM. The performance improvements this enables will be particularly dramatic in virtualization (VMware, XEN, Platform VMO) where mapping plays an important role.

Another improvement is simultaneous multi-threading (SMT). Essentially a refined version of hyper-threading, it means that each processor core can handle two consecutive threads. In other words a 4-core server is treated as having 8 virtual cores; an 8-core server is treated as having 16 virtual cores, etc. Although 8

[1] The memory controller in the test system operated at 1333MHz.

physical cores are still faster than 8 virtual cores, 8 virtual cores will offer a huge improvement over a conventional 4-core machine in multithreaded HPC applications since it can execute twice as many threads. To make this possible Intel has resized several key buffers and modified key buffer partitioning schemes to prevent bottlenecks. Intel has also increased the performance of thread synchronization instructions.

Another interesting new feature is turbo mode. A common criticism of multi-core processors is that if an application is not using every core, you're not actually getting the most out of your computer. Turbo mode is designed to fix this. If the processor is sufficiently cool, or if not all cores are being used, the server will automatically and temporarily over-clock the cores in use to provide a boost even to single-threaded applications. The gain, though modest, means that faster performance is achieved whether an application is multi-threaded or not.

Other changes in technology that make possible the incredible gains in speed achieved by the XEON 5500 include:

- Extended Page Table (EPT): An extension of the VT-x technology, which reduces hypervisor overhead by allowing virtual machines to directly access to virtualized page tables.
- Three levels of cache: 32KB L1 and 256KB L2 cache dedicated to each core, and an 8MB L3 cache, which is shared between the cores.
- QuickPath Interconnect (QPI): A high speed, point-to-point interconnect that connects XEON 5500 CPUs to each other and to the system devices via the x58 (Tylersburg) chipset. The QPI bandwidth on the model we tested was 25.6 GB/sec[2]
- SSE4.2: An extension of seven instructions to the command set that will accelerate string processing. This is the second part of the 54 instruction SSE4 command set extension that was introduced in Intel Penryn core. In January, 2008 the SSE 4.1 command set provided the first 47 instructions.

XEON 5500 Breaks the Downtrend

To test the performance of the XEON 5500, we ran the McCalpin STREAM benchmark on a dual socket, quad-core with 24 gB RAM with three DDR3/1333 channels. Under this scenario, the system had a theoretical maximum memory bandwidth of 63,984 MB/s[3].

We obtained results that averaged almost 4 GB/s/core (3,974 MB/s/core) – effectively doubling the previous generation. As you can see from Figure 2 below, memory bandwidth per core, which had been in a downtrend, is now in an uptrend. This is a direct result of XEON 5500's redesigned memory architecture.

[2] 2 bytes x 6.4GT/sec x 2 (bidirectional) = 25.6 GB/sec
[3] 24 bytes x 1,333MHz x 2 sockets = 63,984 MB/sec

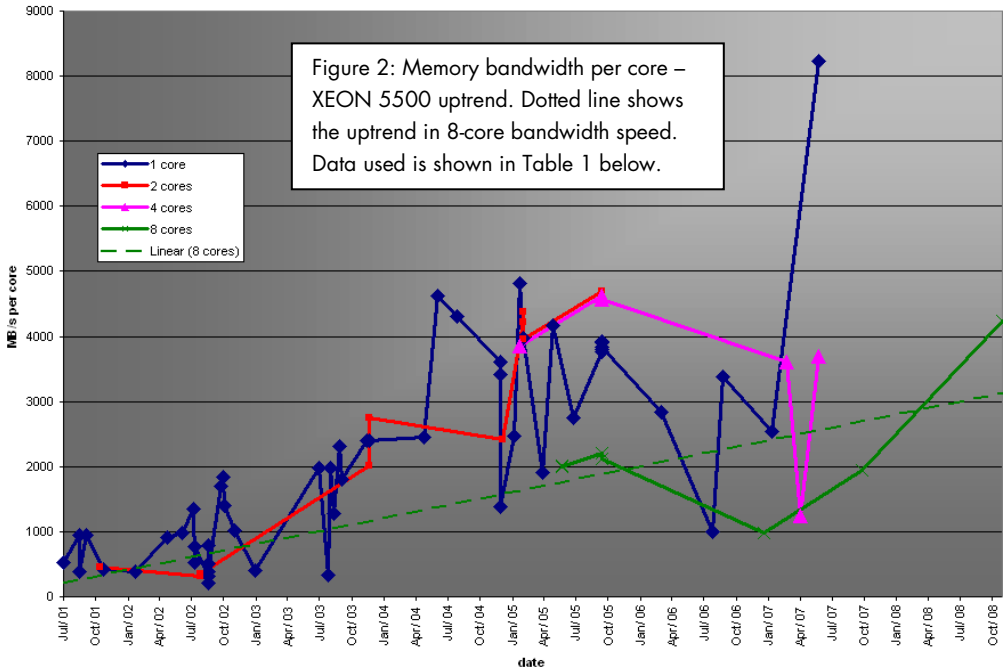


Figure 2: Memory bandwidth per core – XEON 5500 uptrend. Dotted line shows the uptrend in 8-core bandwidth speed. Data used is shown in Table 1 below.

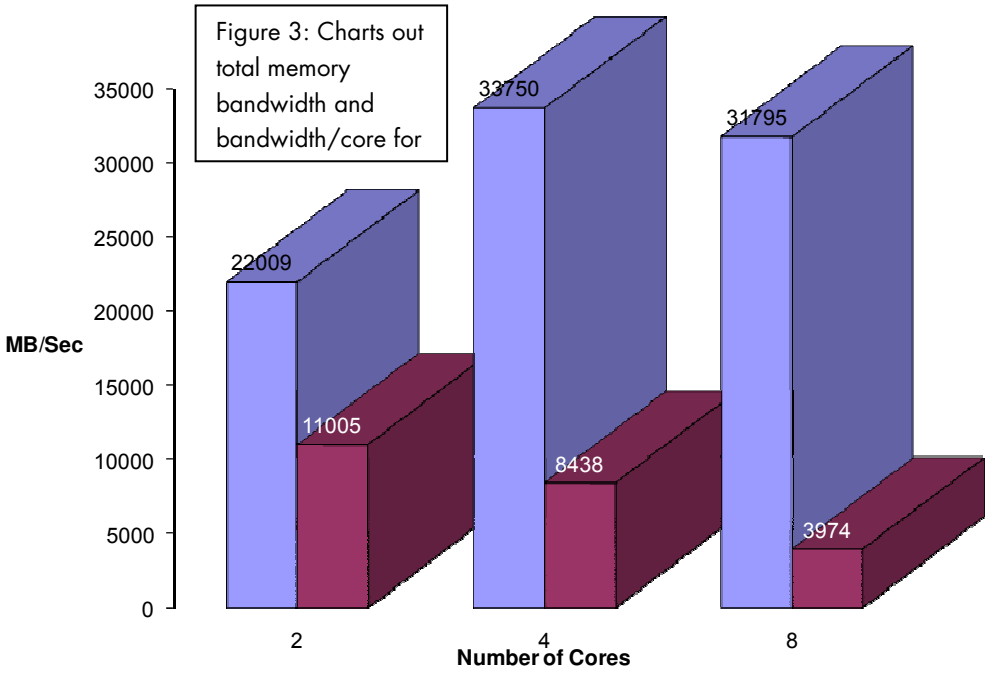


Figure 3: Charts out total memory bandwidth and bandwidth/core for

	2 Threads		4 Threads		8 Threads	
Run	Memory BW	BW/Core	Memory BW	BW/Core	Memory BW	BW/Core
1	24136	12068	36822	9206	32215	4027
2	23960	11980	33352	8338	36045	4506
3	22367	11183	36855	9214	28505	3563
4	20172	10086	36679	9170	30355	3794
5	23419	11709	28779	7195	27541	3443
6	19439	9719	35037	8759	36050	4506
7	19988	9994	36793	9198	30209	3776
8	19694	9847	30734	7684	29974	3747
9	22740	11370	36671	9168	33152	4144
10	24174	12087	25782	6446	33900	4237
	22009	11004	33750	8438	31795	3974

Table 1: Depicts results of McCalpin STREAM benchmark of XEON 5500 memory bandwidth (MB/sec) using Intel Software Tools

The results illustrated by Table 1 and Figure 3 were generated using the McCalpin STREAM benchmark compiled and optimized using the Intel Software Tools. The environment variable OMP_NUM_THREADS was set to the specified number of threads for each test. Each test was run 10 times and the results were averaged in the bottom line.

Part 2: A Comparative Evaluation of Intel's Core i7 Architecture

Introduction

The XEON 5500, a microprocessor based on a new micro-architecture, Core™ i7, (codename Nehalem) is Intel's first implementation of the x86_64 instruction set. This instruction set breaks Intel's tradition of using a shared memory controller for all processors in a shared memory processing system (SMP). A XEON 5500 processor has an integrated memory controller and one can assume therefore, better scalability of the memory bandwidth per core ratio, as more processors are added to the SMP.

In this section of the paper, a detailed comparison between Intel's former high-end server processor, Xeon 5400, and XEON 5500 will be given. Different micro-benchmarks, each exercising different parts of the systems, will be used to analyze the two implementations of the x86_64 architecture and compare them quantitatively. The data set for XEON 5500 used in this comparison was derived from the same tests already presented above.

Lastly, the Standard Performance Evaluation Corporation (SPEC) has a suite of parallel MPI applications constituting the SPEC MPI2007 benchmark. This application suite will be used to compare the two processors. This comparison would be relevant to end-users running High Performance Computing (HPC) workloads on the two processors. The applicability of the micro-benchmarks to predict real world application workloads will be discussed in this context.

The methodology, the description of the micro- and application level benchmarks, and the use of conventions are described in the Methodology section. A description of the two processors and the systems used are discussed in the Description of the Systems section. The final section of Part 1, Benchmark Characterization, contains two major subsections discussing the micro- and application benchmarks respectively.

Methodology

Bandwidth measurements are reported as Gigabytes per second (GB/s), where *Giga* is 10^9 and a *byte* is an Octet (8 bits). Memory sizes and footprints are denoted by kilo-, mega-, and giga-bytes (kB, mB, gB), using the normal conventions (*kilo* equals 2^{10}, *mega* is 2^{20}, and *giga* is 2^{30} bytes).

Sequential memory bandwidth

Sequential memory bandwidth is measured using a modified version of McCalpin's *stream* benchmark [5]. The stream benchmark measures the time it takes to perform a *copy* ($\forall$ i, $a_i = b_i$), a *scale* ($\forall$ i, $a_i = \beta \times b_i$), an *add* ($\forall$ i, $a_i = b_i + c_i$), and a *triad* ($\forall$ i, $a_i = b_i + \beta \times c_i$). Based on the size of the arrays and size of the data elements (8-byte double precision floating point in our case), the *net* memory bandwidth is calculated.

It is worth mentioning that in a cache-based system, the *net* memory bandwidth might be different from the *gross* bandwidth. This is because the caches on a write-miss will allocate the cache surrounding the datum

written. In the *copy* case, the cache will initiate a memory read of a cache-line on a miss when the processor references *b*. A write miss in the cache when *a* is written will normally also trigger a cache-line read from the memory. This is because most caches implement an allocation-on-write-miss policy. Over time, 50% of the cache entries will contain modified data (the *a* array). Hence, 50% of the cache-entry allocations will trigger a flush of a dirty cache-line back to the memory. Hence, the *gross* bandwidth will nominally be 50% higher than the *net* bandwidth in the *copy* and *scale* case. In the two latter constructs, *add* and *triad*, the *gross* bandwidth will be 33% higher than the *net* bandwidth.

Most modern processors, including the two studied in this paper, have special write instructions, which place data in a special write-buffer. When the write-buffer is full, its content is written to the memory bypassing the cache(s), thereby avoiding polluting the cache(s) in the case the data will not be referenced again whilst residing in the cache. Hence, these instructions are often referred to as nonpolluting, non-temporal, or streaming stores. This paper uses the term streaming stores.

On a system where the read and write bandwidth to the memory controller are fairly similar and the flops-to-bandwidth ratio is high (as on most modern microprocessors), one can tell by looking at the results of the *stream* benchmark if streaming stores have been used. In that case, the memory bandwidth of all four patterns would be fairly similar. If streaming stores are not used, *copy* and *scale* will demonstrate less memory bandwidth as compared to *add* and *triad*. In this paper, a re-factored version of the *stream* benchmark is used. The loops measuring the memory bandwidth have been separated out from the rest, in order to more easily apply special compiler optimizations to them. They were compiled using both normal stores and streaming stores. In addition, the harness was ported to the Message Passing Interface (MPI), to enable the benchmark to span all available cores on the systems. Unless otherwise noted, the streaming store version of the loops will be used.

Memory latencies

Sometimes, an execution unit will stall until a specific data element has been read from the memory system. In these situations, the time it takes to read the data element is of vital interest. The latencies of memory accesses, when the access pattern is highly random, will stress different parts of the system than those used with sequential memory accesses. For example, if the footprint of the random accesses is larger than the address space spanned by the virtual-to-physical address translation cache (ATC), then every access will trigger an ATC miss. With the *stream* benchmark on the other hand, one ATC miss will be triggered occasionally and since the accesses in *stream* are sequential, many memory requests from the processor will be served without incurring new ATC misses. Further, the processor can easily pre-fetch data in case of the *stream* benchmark, whereas a highly random access pattern cannot be predicted by hardware. Hence, a benchmark generating random memory accesses has been developed. It is also an MPI program, so latencies can be measured using all cores in the system simultaneously.

SPEC MPI2007

SPEC MPI2007 is a suite of 13 real-world applications, a toolset to run and verify the runs, and a set of rules for running and reporting results. The suite contains applications from 9 applications areas, as listed in the following table.

Scientific Area	SPEC MPI2007 Applications
Computational Fluid Dynamics	leslie3d, fds4, zeusmp, pop2
Quantum Chromodynamics	Milc
Weather Forecasting	Wrf
Parallel Ray Tracing	Tachyon
Molecular Dynamics	lamps
Heat transfer	geoFEM
Hydrodynamics	tera_tf
Linear Algebra	lu
Density Functional Theory	Socorro

Table 1: SPEC MPI2007 Scientific Areas and Applications

If all applications are compiled with the same compiler (per language) and use the same compiler switches, it is called a *base* run. It is also possible to apply selective optimization to the various applications and other special optimizations, in this case the run will be denoted as a *peak* run. In this paper, only *base* runs will be used. All runs were executed using Platform MPI 5.6.4.

Each application is executed a minimum of two times, and the median time is used to calculate a speed-up ratio relative to a reference system. The geometric mean of the 13 ratios is the SPECmpiM_base2007 ratio. The results reported here are, in accordance with SPEC's Run and Reporting Rules, an *Estimate*. This because the results have not been reviewed and published by SPEC.

Description of the Systems

Both systems evaluated in this paper have dual sockets, and each socket is equipped with a quad-core processor (or in the case of Xeon 5400, each processor is a Multi Chip Module (MCM) employing two dual-core dies).

Xeon 5400 system

The system has two E5472 Intel Xeon processors running at 3.00GHz. Each processor is connected to the memory controller hub (MCH) using a private, 64-bit wide front side bus (FSB) running at 1600MHz. The MCH connects to 16 gB of memory (8*2gB PC2-6400 CL5-5-5 FB-DIMMs). The theoretical maximum memory bandwidth is 12.8 GB/s for a processor and twice that for the system. Xeon 5400 has, as depicted in Table 2, two 32 kB L1 caches, one for instructions and one for data. On each die, the two cores share a 6 MB L2 unified L2 cache. Since each processor has two dies, a total of 12 MB L2 cache resides on the MCM.

XEON 5500 system

The XEON 5500 system consists of two X55700 Intel processors running at 2.93GHz. The processors are connected by 4 instantiations of Intel's new QuickPath Interconnect (QPI), formerly known as Common Systems Interconnect (CSI), each delivering 25.6 GB/s peak bandwidth.

Each processor has three 64-bit buses connecting to fully buffered DIMMs running at 1333MHz. On the system used, 20 gB of memory were installed. However, due to an error on the motherboard on the pre-production system made available for this research, 4 DIMMs (8 gB) were installed on the first processor, whereas 6 DIMMs (12 gB) were installed on the second processor. The asymmetrical placement of memory led to a reduction of available bandwidth of 17% for the first processor. The reduced performance is caused by lack of interleaving due to a lower number of DIMMs. The theoretical peak memory bandwidth per processor is 32 GB/s and 64 GB/s for the system used in this experiment.

The L2 caches of XEON 5500 are equal to those of Xeon 5400. XEON 5500's L2 cache is a private, unified 256 kB cache. Contrary to Xeon 5400, XEON 5500 has a unified and shared L3 of 8 MB. It is worth noting that XEON 5500's L3 cache is 33% smaller than Xeon 5400's two L2 caches.

Cache	XEON 5500	Xeon 5400
L1	L1 32KB I + 32KB D per core	32KB I + 32KB D per core
L2	L2 256K I+D per core	12MB I+D, 6MB shared / 2 cores
L3	L3 8MB i+D shared by all cores	None

Table 2: Description of the cache-system of the two processors

Benchmark characterization

Stream benchmark

In our first experiment, we used only a single core on each system. The *stream copy* function was used to evaluate the two processors, using both streaming and ordinary stores (the latter labeled *polluting* in Figure 1).

The polluting flavor of the benchmark exhibits the highest performance for footprints up to 4 MB in the Xeon 5400 case, whereas XEON 5500 continues this trend up to 8 MB. Interestingly, above 8 MB, XEON 5500 performs similarly for the two cases, with writes in the version using polluting being slightly better.

This is extraordinary, and two hypotheses might explain this behavior. First, it is possible that XEON 5500 is able to anticipate the behavior of the application and "translate" the ordinary stores into streaming stores, thereby not filling up the L3 cache with the *a* array at all. This would reduce the benchmark cache footprint by a factor of two, since only the *b* array would use the cache. Hence, if this hypothesis holds true, we should see a shift in performance at 16 MB in the polluting case. The shift is at 8 MB, so this hypothesis does not hold true. The second hypothesis is that it is the cache-line writes that restrict the performance and

that one or two cache-line reads does not matter. Further experiments are required in order to verify this hypothesis, and it is considered a topic for future research.

For the polluting stores case, we see three plateaus in the Xeon 5400 case. The first plateau, which this processor shares with XEON 5500, corresponds to the L1 cache size and ends at 32 kB. The next plateau goes from 32 kB up to 4 MB. One would expect it to end at 6 MB, but we consider this an artifact of a non power-of-two cache size. The last plateau levels out at 3.5 GB/s and is the copy memory bandwidth available to a single Xeon 5400 core.

XEON 5500 has 4 plateaus, simply corresponding to the cache sizes at the various levels. We see from Figure 1 that XEON 5500's L2 cache delivers 40 GB/s from 32-256 MB, whereas Xeon 5400 in this area delivers 28 GB/s. Beyond 256 kB and up to 4 MB, the two processors perform equally. XEON 5500's memory bandwidth of 11.5 GB/s is 3.2 times faster compared to Xeon 5400.

Looking at the streaming graphs in Figure 1, we see that the streaming version of the *copy* function is counterproductive until the data footprint exceeds the size of the last cache level. We also see that this function reduces the benchmark cache footprint effectively in half, XEON 5500's plateau changes at 64 kB. This is because 50% of the footprint bypasses the cache.

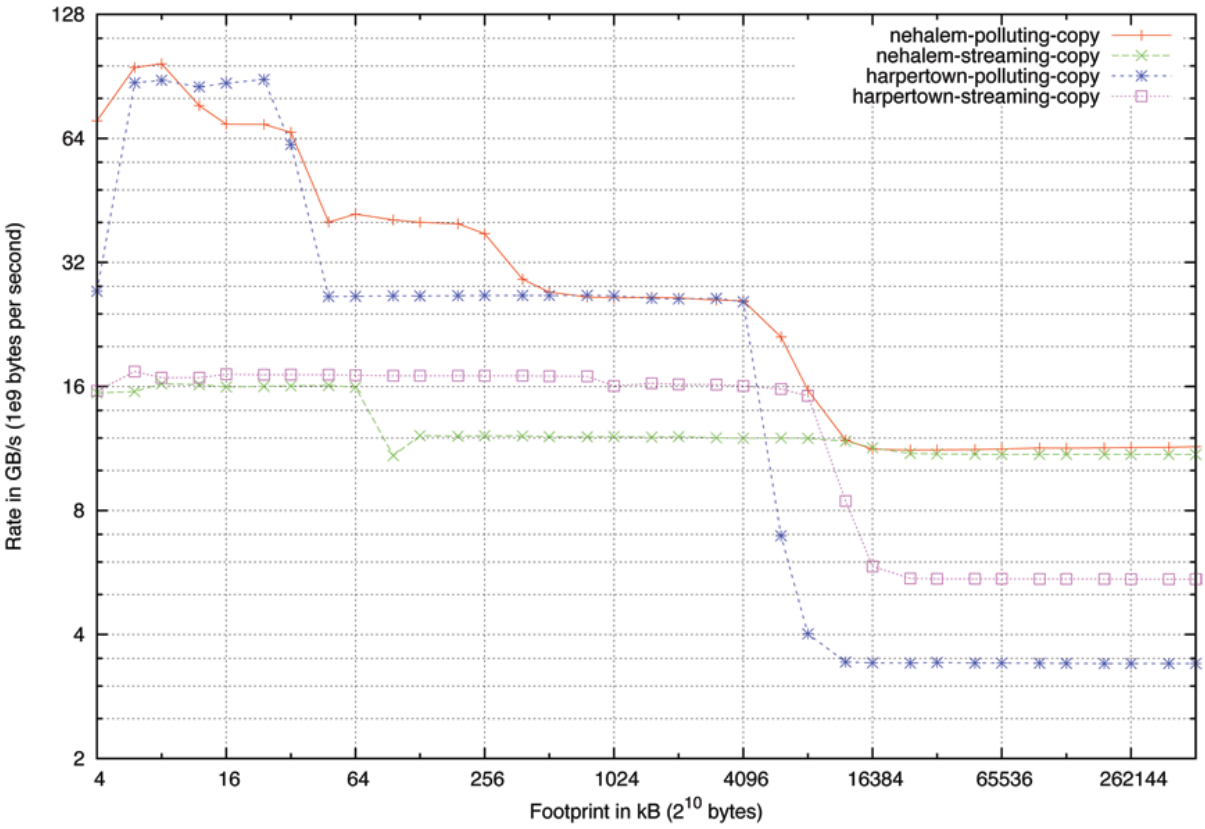


Figure 1: Stream copy performance using a single process on XEON 5500 and Xeon 5400

In our second experiment, we compared the two processors using the streaming version of the *copy* function, going from a single process up to 8 processes, and in the XEON 5500 case, we also performed an evaluation using 16 processes. In the latter case, each process ran on a separate execution thread inside a core, a so-called simultaneous multi-thread, SMT.

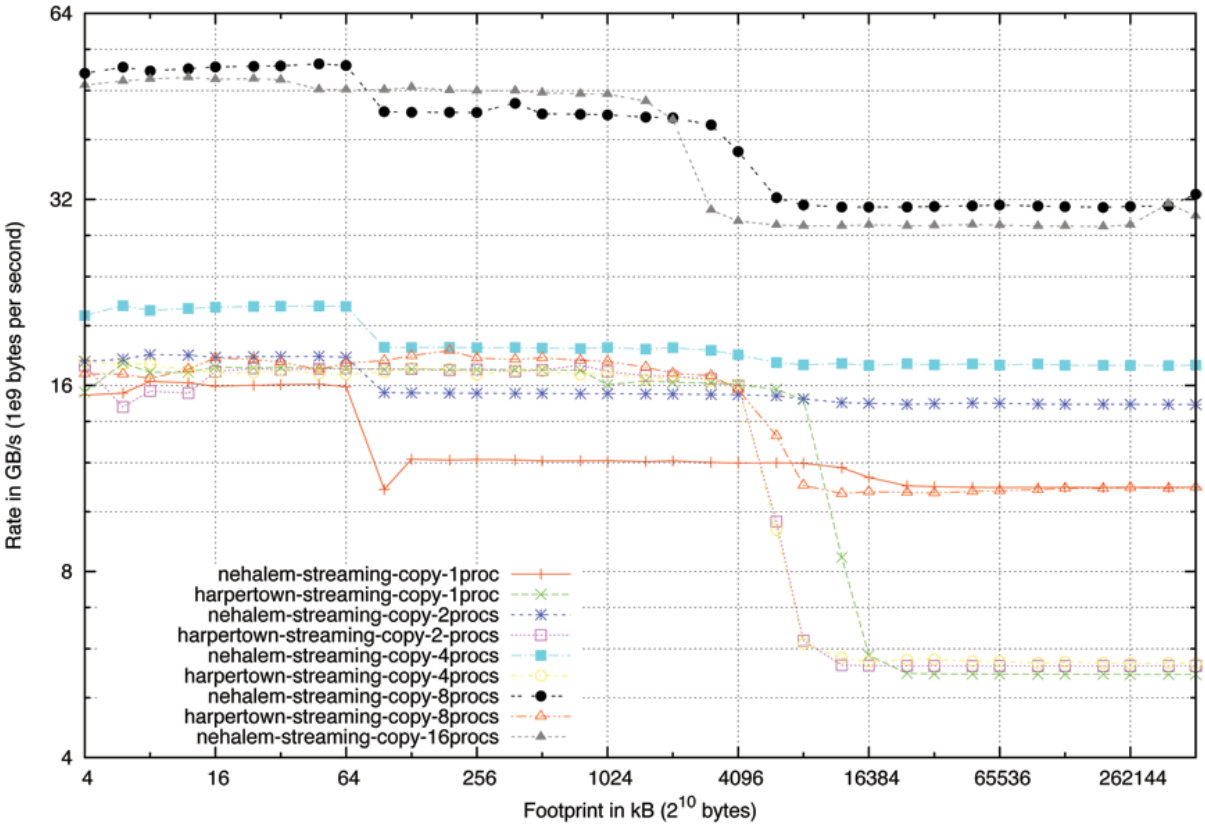


Figure 2: Stream copy performance on XEON 5500 vs. Xeon 5400

Since the experiment used all cores on one processor before using the other, it is interesting to see if the accumulated bandwidth increases with the number of cores used per processor. From Figure 2, we clearly see that 1, 2, and 4 processes on a Xeon 5400 processor all yield 5.5 GB/s, with microscopic lift when going from one to two processes. Going from 4 to 8 processes on Xeon 5400 yields a linear improvement in bandwidth, we observer 11 GB/s using all cores on both processors, which translates to a modest 1.4 GB/s per core.

Looking at XEON 5500, we see that each increase in number of processes also increases the accumulated bandwidth. Further, we see that a single XEON 5500 process achieves exactly the same as 8 Xeon 5400 processes. Put it another way, a memory bandwidth-constrained application would run equally fast on a single XEON 5500 core compared to two fully utilized Xeon 5400 processors. Utilizing more cores on XEON 5500 yields more accumulated bandwidth, but it is far from linear. We observe 11, 7.5, 4.3, and 4 GB/s per core using 1, 2, and 4 cores on a single processor.

With both processors fully utilized, 32.6 GB/s is observed, slightly less than a linear scaling. We construe this lack of linear scalability to the artifact of the motherboard, as explained in the description of the XEON 5500 system. Utilizing the SMT shows, as one would expect, almost the same accumulated bandwidth. That is, one can run 16 processes on the XEON 5500 system and achieve 1.9 GB/s per SMT, which is still more that the 1.4 GB/s per core achieved by the Xeon 5400 system. An interesting observation utilizing SMT is that it achieves better performance for footprint sizes matching the L2 cache on XEON 5500, as compared to only utilizing the cores.

Memory latencies

Moore's law has predicted the evolution of processor performance and frequencies. Memory latencies have not kept pace with the processor advances. Hence, the cost of waiting for memory in terms of the equivalent number of instructions which could have been executed if data resided in a cache close to the processor, increases almost proportionally to Moore's law. From this perspective and also as a pure evaluation of XEON 5500's integrated memory controller, it would be interesting to see how the processors compare with respect to memory latencies, and see the effect of adding more active processes or cores to the picture.

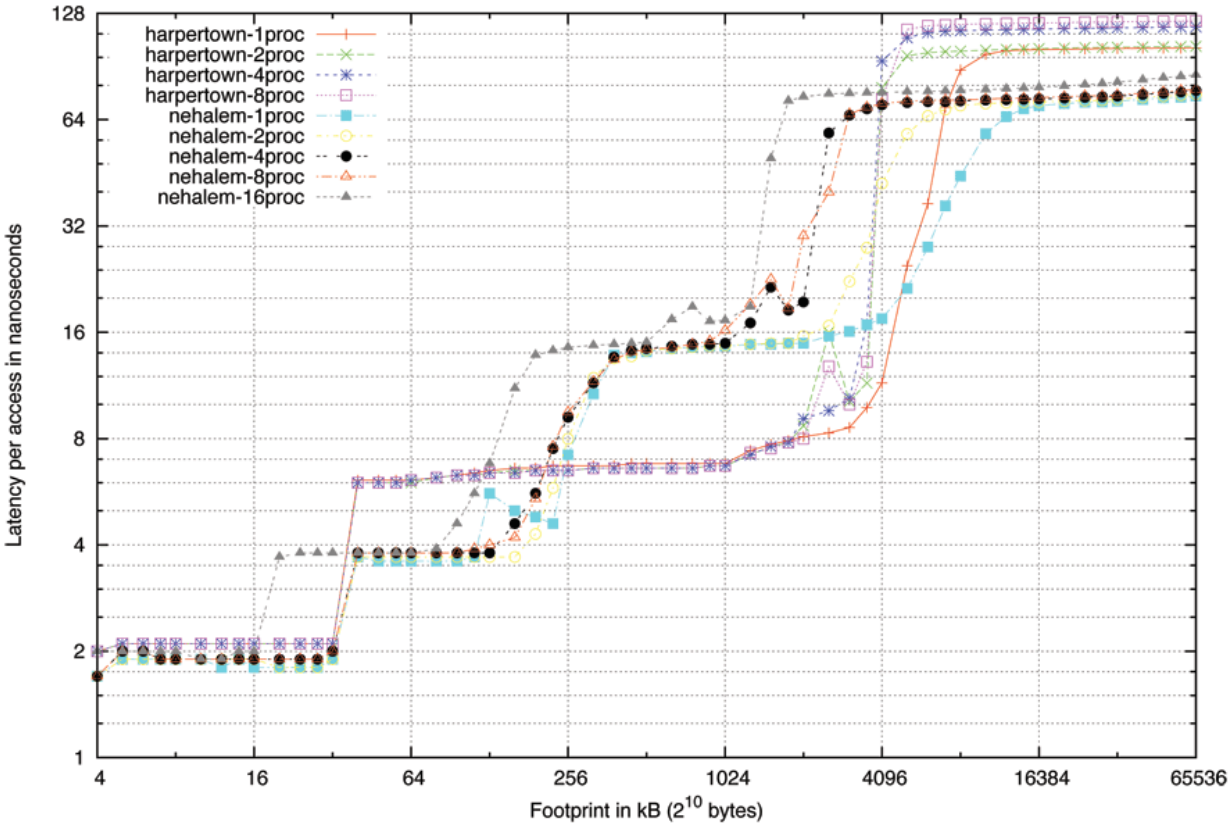


Figure 3: XEON 5500 and Xeon 5400 memory latencies as function of memory footprint

In Figure 3, the effective latency for reading data is given. The figure clearly depicts the different latency plateaus and also which of the caches are shared or private.

For workloads which exhibit a high cache-hit rate in the L1 cache, we see latencies close to 2 ns for both processors. In the 16-processor SMT case using XEON 5500, the L1 cache is shared between two SMTs residing on the core owning the L1 cache. Hence, for this case, we see the plateau ending at 8 kB, ½ the L1 cache size.

The benefit of XEON 5500's L2 cache is visible; for footprints up to 256 kB we see a latency of 4 ns, whereas Xeon 5400 achieves 6 ns. Again we see the SMT effectively sharing the core-private L2 cache, almost the same shape we see for Xeon 5400 using two processes (on the same die sharing the L2 cache). Since increasing the number of processes to 4 and 8 on Xeon 5400, we see these 3 lines have the same shape effectively for the whole range of footprints.

Going from 1, 2, 4, …, 16 processes on XEON 5500 will effectively shift the graph to the left and this trend is clearly visible. There is however one exception to this trend; going from 4 to 8 processes. These two graphs are almost identical. This is because we have just replicated the experiment on two processors instead of one. The memory latencies for the different cases are listed in Table 3, since they are hard to read out of Figure 3.

Processor	No of processes	Latency (ns)
Xeon 5400	1	102
	2	103
	4	117
	8	122
XEON 550	1	74
	2	76
	4	77
	8	78
	16 (SMT)	86

Table 3: Memory latencies

From Table 3 we see that XEON 5500 has significantly shorter memory latencies. Furthermore, Xeon 5400's latencies increase by 20% going from a single process to using all 8 processes/cores. The corresponding increase in XEON 5500 is only 4 ns, a modest 5%. Using the SMT feature of XEON 5500, an 8ns increase is observed. Although this is higher that the step from a single process to 8, 16 processes running on the XEON 5500 system still exhibit far shorter latencies than Xeon 5400 running only a single process.

The results from the latency measurements indicate the differences between the two architectures to be far smaller using this metric as compared to the memory bandwidth differences.

Application performance

SPEC MPI2007 was run on the two systems, as an indication of how a real-world HPC workload would experience them. On the Xeon 5400 system 8 processes were used, whereas on the XEON 5500 system, 8

and 16 processes were evaluated. This is in order to investigate if Intel's SMT technology is applicable for HPC workloads. Table 4 summarizes the results.

Application	H-8	XEON 5500-8	XEON 5500-8 faster than H-8	XEON 5500-16	XEON 5500-16 faster than N-8	XEON 5500-16 faster than H-8
104.milc	0.48	1.68	250%	2.00	19%	317%
130.socorro	0.55	2.20	300%	2.14	-3%	289%
137.lu	0.49	1.43	192%	1.46	2%	198%
128.GAPgeofem	0.67	1.92	187%	1.93	1%	188%
113.GemsFDTD	1.03	2.63	155%	2.77	5%	169%
107.leslie3d	0.62	1.50	142%	1.66	11%	168%
127.wrf2	1.35	3.38	150%	3.32	-2%	146%
115.fds4	0.60	1.46	143%	1.40	-4%	133%
132.zeusmp2	0.83	1.56	88%	1.64	5%	98%
126.lammps	0.84	1.26	50%	1.58	25%	88%
121.pop2	1.51	2.28	51%	2.40	5%	59%
122.tachyon	1.05	1.13	8%	1.52	35%	45%
129.tera_tf	1.25	1.45	16%	1.76	21%	41%
SPECmpiM _base2007(est)	**0.69**	**1.75**	**154%**	**1.90**	**9%**	**176%**

Table 4: SPEC MPI2007 ratios of the two systems, sorted on how much faster XEON 5500 using 16 processes are compared to the Xeon 5400 system

The two most important findings from table 4 are that a) XEON 5500 using SMT and 16 processes (N-16) is up to 300% faster than Xeon 5400 using 8 processes (H-8) and b) those applications which do not have a large speedup using 8 processes on XEON 5500 (N-8), seem to take a great advantage of SMT. Actually, the application with the lowest benefit from N-8, *tachyon*, get the highest benefit from the SMT technology, a 35% improved performance over N-8.

The average improvement of N-8 over H-8 is 154%. By utilizing SMT, N-16 performs 176% faster on the average of these 13 applications. The SMT technology alone improves the average by 9%.

The SMT technology is counterproductive in 3 of the cases, but only with a small decrease, namely 2, 3, and 4% respectively. The three applications that benefit the most from this technology, improve by 21, 25, and 25% respectively. In lieu of Intel's former implementation of hardware multi-threading, HyperThreading, the SMT implementation on XEON 5500 seem significantly better for HPC workloads.

Part 3: Optimizing HPC Performance on XEON 5500 via Platform LSF Processor Binding

High Performance Thread Scheduling

The Linux Kernel scheduler is a good, general purpose scheduler. It allocates tasks to logical processors in order to satisfy its sometimes conflicting goals, such as maximizing resource utilization while minimizing interactive response times. The kernel always tries to keep some computing resources idle so that when a user logs in, she will feel that her commands are being executed in real time.

In order to meet these goals in a multi-user, production system, the operating system will often move threads from one core to another and sometimes from a core with a hot cache to a core with a cold cache, reducing performance. Because it is general purpose scheduler, the kernel is not able to optimally assign processes to cores for high-performance technical computing applications.

The Kernel Scheduler

vs.

The Platform LSF Scheduler

The Platform LSF scheduler provides hard processor binding for sequential jobs and parallel jobs that run on a single host. Platform LSF is only concerned about enforcing the policies you define, it is able to optimally assign and bind processes to cores for HPC applications. Processor binding for Platform LSF job processes takes advantage of the power of multiple processors, cores and threads to provide hard processor binding functionality for sequential Platform LSF jobs. Binding jobs to logical processors can result in increased performance. When processor binding for Platform LSF job processes is enabled on supported hosts, the jobs processes are bound to a processor according to the binding policy in effect.

PCT (Processors, Cores, or Threads)

By default, the number of logical processors a host has is equal to the number of physical processors it has. For hosts with multiple processors, cores, and threads, logical processors can be defined by the cluster administrator to be:

- Processors
- Processors and cores
- Processors, cores, and threads

This is a cluster-wide definition, which is controlled by EGO_DEFINE_NCPUS parameter in the lsf.conf file.

Binding Policy

The binding policy works in concert with the PCT definition to set hard processor affinity for sequential or single-host parallel HPC applications. A default binding policy can be specified at the cluster level in the lsf.conf file, and the default policy can be over-ridden on an application-by-application basis in the lsb.applications file.

The BIND_JOB=BALANCE policy instructs Platform LSF to bind the job based on the load of the available cores. For example, the first job process is bound to the first core on the first physical processor, the second job process is bound to the first core on the second physical processor, the third job process is bound to the second core on the first physical processor, and so on.

The BIND_JOB=PACK policy directs Platform LSF to bind the job to a single processor where it makes sense, without oversubscribing the processors. The other processors are used when they are needed. For example, assume that three single-host parallel jobs are submitted. The first job is bound to the first and second cores of the first processor, the second job is bound to the third and fourth cores of the first processor. Binding the third job to the first processor would oversubscribe the cores in the first processor, so the third job is bound to the first and second cores of the second processor.

The binding policy can also be delegated to the user through the BIND_JOB=USER and BIND_JOB=USER_CPU_LIST policies. If the binding policy is USER, the user can specify the binding policy through the LSB_USER_BIND_JOB environment variable. The policy can be Y, N, NONE, BALANCE, PACK, or ANY. If the binding policy is USER_CPU_LIST LSF binds the job to the explicit list of processors specified in environment variable $LSB_USER_BIND_CPU_LIST.

Platform LSF Processor Binding Increases Memory Bandwidth

As impressive as the enhancements to the XEON 5500 memory system gains are, our tests showed that Platform LSF processor binding can increase total memory bandwidth by another 9-14%.

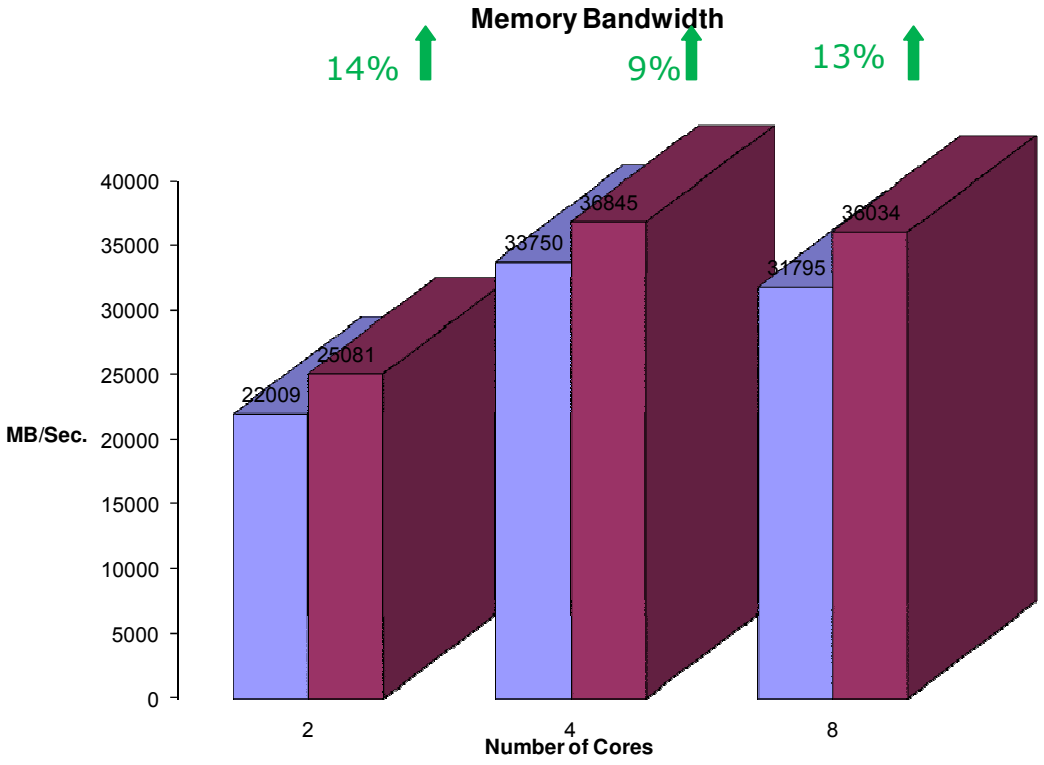


Figure 1: Charts out total memory bandwidth for Linux Kernel Scheduler compared to Platform LSF using 2, 4 and 8 cores.

	2 Threads		4 Threads		8 Threads	
Run	Kernel	Platform LSF	Kernel	Platform LSF	Kernel	Platform LSF
1	24,136	22,686	36,822	36,907	32,215	36,029
2	23,960	25,958	33,352	36,757	36,045	36,144
3	22,367	25,928	36,855	36,692	28,505	36,016
4	20,172	25,924	36,679	36,771	30,355	35,988
5	23,419	25,885	28,779	36,908	27,541	36,012
6	19,439	24,332	35,037	36,892	36,050	36,064
7	19,988	22,900	36,793	36,861	30,209	36,018
8	19,694	25,931	30,734	36,888	29,974	36,023
9	22,740	25,400	36,671	36,889	33,152	36,033
10	24,174	25,864	25,782	36,884	33,900	36,017
	22,009	25,081	33,750	36,845	31,795	36,034

Table 1: Compares results of McCalpin STREAM benchmark of XEON 5500 processor speed in test of Linux kernel and Platform LSF scheduler showing clear performance gains with Platform LSF.

Platform LSF Processor Binding Increases Memory Bandwidth / Core

As a result memory bandwidth per core also increased by 9-14%.

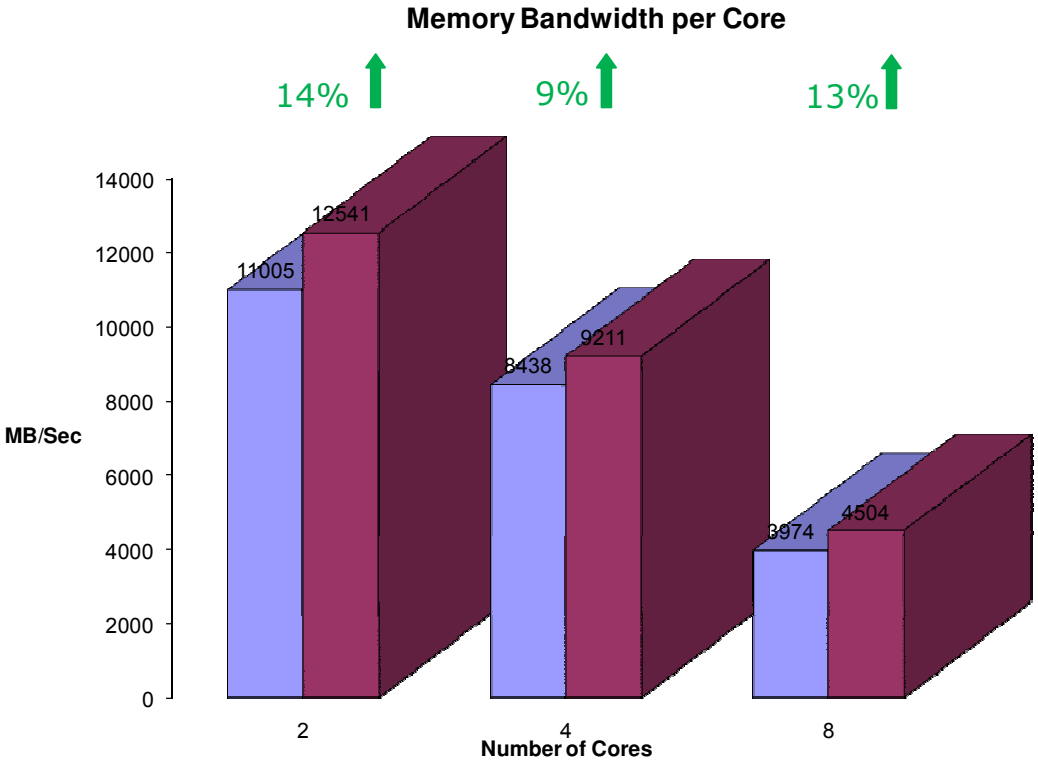


Figure 2: Charts out total memory bandwidth/core for Linux Kernel Scheduler compared to Platform LSF using 2, 4 and 8 cores

	2 Threads		4 Threads		8 Threads	
Run	Kernel	LSF	Kernel	LSF	Kernel	LSF
1	12068	11343	9206	9227	4027	4504
2	11980	12979	8338	9189	4506	4518
3	11183	12964	9214	9173	3563	4502
4	10086	12962	9170	9193	3794	4498
5	11709	12942	7195	9227	3443	4501
6	9719	12166	8759	9223	4506	4508
7	9994	11450	9198	9215	3776	4502
8	9847	12965	7684	9222	3747	4503
9	11370	12700	9168	9222	4144	4504
10	12087	12932	6446	9221	4237	4502
	11004	12540	8438	9211	3974	4504

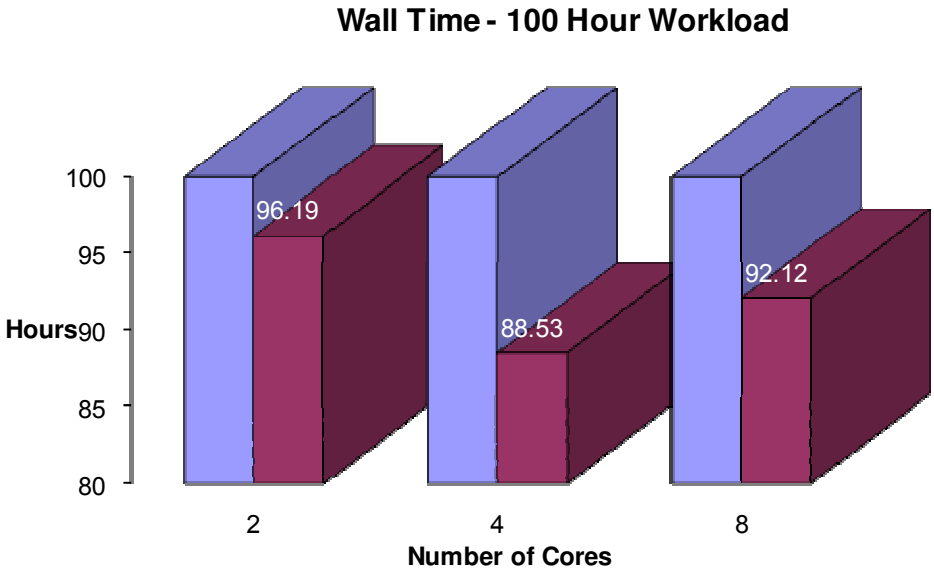
Memory Bandwidth (MB/sec)

Table 2: Compares results of McCalpin STREAM benchmark of XEON 5500 processor speed in test of Linux kernel and Platform LSF scheduler showing clear performance gains with Platform LSF.

Platform LSF Processor Binding = Shorter Run Time

Platform LSF processor binding directly and positively increases the productivity of your compute servers by reducing application run times. It's true that different types of applications will benefit from processor binding to differing degrees depending on how many threads they use, how memory intensive they are, and what binding policy is used among other factors.

Whether you're designing semi-conductors, researching new vaccines, or studying the composition of sub-atomic particles, reduced application run time means a shorter time to results, reduced costs, lower energy usage, more efficient use of expensive software licenses, and maximized use of computing resources.



Conclusions

In this paper, Intel's newest server processor, XEON 5500, has been evaluated and compared to the former flagship, the Xeon 5400 processor. A thorough evaluation by micro-benchmarks and applications has revealed that XEON 5500 is significant better than its predecessor in many areas. A single process on the XEON 5500 system has more memory bandwidth available than the accumulated bandwidth using 8 processes on the Xeon 5400 system. Using 16 processes by means of the SMT technology on the XEON 5500 system, each process has more available bandwidth than a process on the Xeon 5400 system when it is fully loaded.

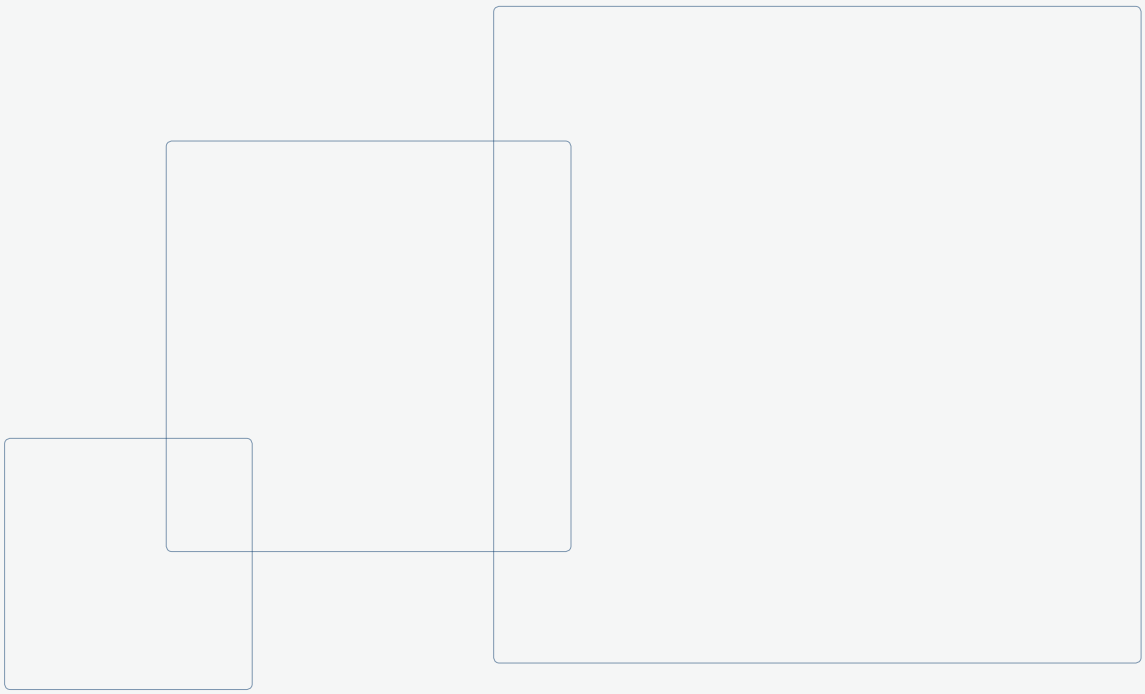
Applications run up to 4 times faster on XEON 5500, a significant achievement. Of the 13 applications evaluated, 9 of them ran more than 2 times faster (98 – 317% faster). Intel's new hardware multi-threading, SMT, improved performance by more that 10% for 5 of the applications, with a peak improvement of 35%.

Acknowledgements

A lot of people and organizations have contributed to this work. Intel Corporation has lent my employer, Platform Computer Corporation, a pre-production XEON 5500 system. Without access to that system, this work couldn't have been accomplished. Further, SGI's *orion* system was used to run the Xeon 5400 benchmarks and applications. SGI's HPC support has done a tremendous job. The SPEC MPI2007 committee has always been responsive and helpful whenever the author had questions related to the SPEC application and benchmark harness.

References

1. Intel Corporation, Intel® Xeon® Processor 5500 Series, Product Brief, 2009
2. Intel Corporation, Quad-Core Intel® Xeon® Processor 5400 Series, Product Brief, 2007
3. Message Passing Interface Forum, MPI: A message-passing interface standard. http://www.mpi-forum.org (2003) Matthias S. Müller and Kumaran Kalyanasundaram and Greg Gaertner and Wesley B. Jones and Rudolf Eigenmann and Ron Lieberman and G. Matthijs van Waveren and Brian Whitney, SPEC HPG Benchmarks for Large Systems, ISHPC, volume 2858, 189–201 (2003)
4. J. McCalpin, Memory bandwidth and machine balance in current high performance computers, in IEEE Comp. Soc. Tech. committee on Computer Architecture (TCCA) Newsletter, Dec. 1995, 19-25
5. Intel Corporation, Intel® QuickPath Architecture, White Paper, 2008



Platform™

Platform Computing is the leader in grid and cloud computing software that dynamically connects IT resources to workload demand according to business policies. Over 2,000 of the world's largest organizations rely on our solutions to improve IT productivity and reduce data center costs. Platform has strategic relationships with Dell™, HP, IBM®, Intel®, Microsoft®, Red Hat®, and SAS®. Building on 16 years of market leadership, Platform continues to help data centers be more efficient, responsive and dynamic. Visit www.platform.com.

World Headquarters
Platform Computing Inc.
3760 14th Avenue
Markham, Ontario
Canada L3R 3T7
Tel: +1 905 948 8448
Fax: +1 905 948 9975
Toll-free tel: 1 877 528 3676
info@platform.com

Sales - Headquarters
Toll-free tel: 1 877 710 4477
Tel: +1 905 948 8448

North America
New York: +1 646 290 5070
San Jose: +1 408 392 4900
Detroit: +1 248 359 7820

Europe
Basingstoke: +44 (0) 1256 883756
London: +44 (0) 20 7977 1480
Paris: +33 (0) 1 41 10 09 20
Düsseldorf: +49 2102 61039 0
Munich: +49 89 517397 52
Oslo: +44 1256 883756
info-europe@platform.com

Asia-Pacific
Beijing: +86 10 82276000
Xi'an: +86 029 87607400
asia@platform.com

Tokyo: +81(0)3-6302-2901
info-japan@platform.com

Singapore: +65 6307 6590
lliew@platform.com