Performance of Two-Way Opteron and Xeon Processor-Based Servers for Scientific and Technical Applications

Douglas M. Pase and James Stephens
IBM eServer xSeries Performance
Research Triangle Park, North Carolina, USA

Abstract

The performance[1] of Linux clusters used for High-Performance Computation (HPC) applications is affected by the performance of three important components of server architecture: the Arithmetic Logic Unit (ALU) or processor core, the memory, and the high-speed network used to interconnect the cluster servers or nodes. The behavior of these three subsystems is in turn affected by the choice of processor used in the server.

In this paper we compare the performance of two servers that are typical of those used to build Linux clusters. Both are two-way servers based on 64-bit versions of x86 processors. The servers are each packaged in a 1U (1.75 inch high) rack-mounted chassis. The first server we describe is the IBM® eServer™ 326, based on the AMD Opteron™ processor. The second is the IBM eServer xSeries™ 336, based on the Intel® Xeon processor with Extended Memory 64 Technology (EM64T) enabled. Both are powerful servers designed and optimized to be used as the building blocks of a Linux cluster that may be as small as a few nodes or as large as several thousand nodes.

We describe the architecture and performance of each server. We use results from the popular SPEC® CPU2000 and Linpack benchmarks to present different aspects of the performance of the processor core. We use results from the STREAM benchmark to present memory performance. Finally, we discuss how characteristics of the I/O slots affect the interconnect performance, whether the choice is Gigabit Ethernet, Myrinet, InfiniBand, or some other interconnect.

Introduction

The IBM eServer 326, or e326 [1], is a rack-optimized 1U system based on the AMD Opteron processor.[2] The e326 is optimized for scientific and technical workloads requiring efficient floating-point processing and high bandwidth to memory. The e326 is ideally suited for use in

---

[1] Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

[2] A "U" is a measurement of height, 1.75 inches, used to separate sets of bolts in a standard electronic equipment rack.

high-performance clusters. It supports one or two Opteron processors and up to 16GB of DDR400 memory. Disk subsystems may be either IDE or hot-swap SCSI. Each Opteron processor supports its own memory controller, which makes the e326 a Non-Uniform Memory Access (NUMA) design.

The IBM eServer xSeries 336, or x336 [2], is also a rack-optimized 1U system, but it is based on the Intel Xeon processor family. The x336 supports one or two Xeon processors and up to 16GB of DDR-II 400 memory. Disk subsystems are hot-swap SCSI. The processors support a higher clock frequency range than does Opteron, which gives the x336 an advantage for some types of calculations. Xeon requires a single external memory controller, called a north bridge, which makes the x336 a Symmetric Multi-Processor (SMP) design.

Scientific workloads may be characterized generally as heavily numeric in nature, often dominated by 64-bit floating-point operations. Many of the codes were originally developed on vector machines like those produced by Cray Research, Fujitsu and NEC. Vector machines have high memory bandwidth and provide large performance gains to applications that apply the same operation to large sections of memory using regular strides. This is very different from cache-based scalar microprocessors, which reward possibly varying operations applied to the same data over and over again. To make things more difficult for the system designer, many key vector applications have been converted to run well on cache-based systems because of their attractive pricing. This means that, for a system to do well on scientific workloads, it must have a fast processor, high bandwidth to memory, a large cache and an affordable price.

Unfortunately, there is often a significant gap between the hardware peak memory performance and the memory performance available to an application. The STREAM benchmark [3] was created to measure available memory bandwidth. STREAM is a simple benchmark of four loops. The first loop, COPY, copies data from one vector into another; the second, SCALE, multiplies a vector by a scalar value; the third loop, ADD, adds two vectors; and the fourth loop, TRIAD, combines a scale with an addition operation. This last loop is very similar to a DAXPY operation (Double-precision A times X Plus Y). The arrays used in these four loops are required to be much larger than the largest processor cache, so operands are always retrieved from memory. All operations use 64-bit operands and memory accesses are stride-one (i.e., a[i], a[i+1], a[i+2], ...).

In contrast to STREAM, the Linpack benchmark [4] focuses primarily on processor Arithmetic Logic Unit (ALU) floating-point performance rather than memory performance. Linpack itself is a collection of subroutines that analyze and solve linear equations and linear least-squares problems [5, 6]. Originally designed for supercomputers in the 1970s and early 1980s, Linpack solves linear systems whose matrices are general, banded, symmetric indefinite, symmetric positive definite, triangular, and tridiagonal square. Linpack is built upon the Basic Linear Algebra Subroutine package, or BLAS. The Linpack benchmark uses the Linpack library to solve a general dense system of linear equations using LU factorization [7]. The Linpack library has largely been replaced by Lapack, which extends and improves upon the routines [8]. The routines have been carefully rewritten and tuned to take advantage of processor cache, and relatively few references actually go to memory. For our tests, we use the double-precision High Performance Linpack (HPL) benchmark over a wide range of problem sizes. Our benchmark implementation uses the high-performance BLAS created by Kazushige Goto [9].

STREAM and Linpack are both kernel benchmarks; that is, they focus on measuring the performance of a small, relatively simple mathematical kernel. In so doing they give an accurate measure of the performance of a single subsystem of the computer. To avoid the bias inherent in kernel benchmarks, we also measure system performance using a more realistic benchmark, SPEC CPU2000 [10]. This benchmark is actually two suites of applications. One suite, SPEC CINT2000, consists of highly cacheable integer applications. The other, SPEC CFP2000, contains memory-intensive, floating-point applications. SPEC CPU2000 allows the suites to be run on a single processor to measure processor speed, or to be run on one or more processors to measure system throughput rates.

x86 64-Bit Processor Architecture

This new generation of processors offers all of the advantages of low cost and compatibility available to older generations of x86 processors. It natively supports the x86 instruction set, but it also supports 64-bit extensions, called the x86-64 extensions. This compatibility allows both Opteron and Xeon to execute all x86 applications at full speed. No translation or emulation layer is required to execute x86 programs. The x86-64 extensions provide 64-bit virtual addresses and an extended register set, so 64-bit programs can also be executed on the same machine. 64-bit Linux operating systems allow both 32-bit and 64-bit applications to be executed at the same time with no difficulty whatsoever. An application can be compiled for 64-bit mode, that is, to use the larger addresses and additional registers, with little more effort than to change a flag on the compile line. The larger address space allows an application to address more than 3GB of memory without resorting to overlays, or interfaces (e.g., PAE36) that are slow to execute and difficult to use. The larger register set allows a compiler to reduce the amount of memory traffic by saving more of the intermediate values into registers, when appropriate. The x86 and x86-64 registers are illustrated in Figure 1.
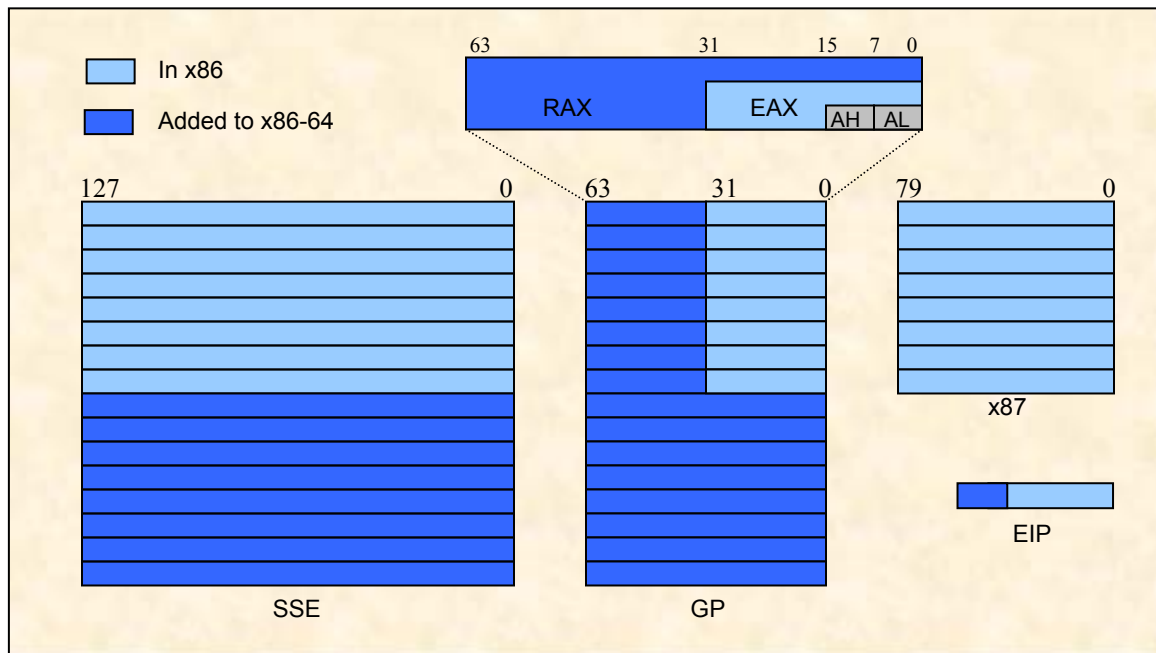

Figure 1. x86 and x86-64 Registers

Because "64-bit" is such a popular term, it is important to be clear on what it means. For purposes of this document, it means the architecture supports 64-bit virtual addresses. x86 and other architectures that are considered 32-bit architectures support 64-bit integers and 64-bit floating-point operands. Such processors may also support physical addresses larger than 32-bits. The size of the addresses used by a user application determines whether an architecture, or an application, is 32- or 64-bit.

Xeon and x336 Architecture

The x336, a 1U dual-processor Xeon system, is rack-optimized and designed to be used as a computational node in a scientific cluster. It supports up to 16GB of main physical memory in eight Dual Inline Memory Module (DIMM) slots. The system also supports two 100 MHz PCI-X slots, or one 133 MHz PCI-X slot, or one PCI-Express (PCI-E) x8 slot. Processor speeds

currently range from 2.8 GHz to 3.6 GHz, and the system supports 400 MHz DDR-II memory. A simplified block diagram of the x336 is shown in Figure 2.
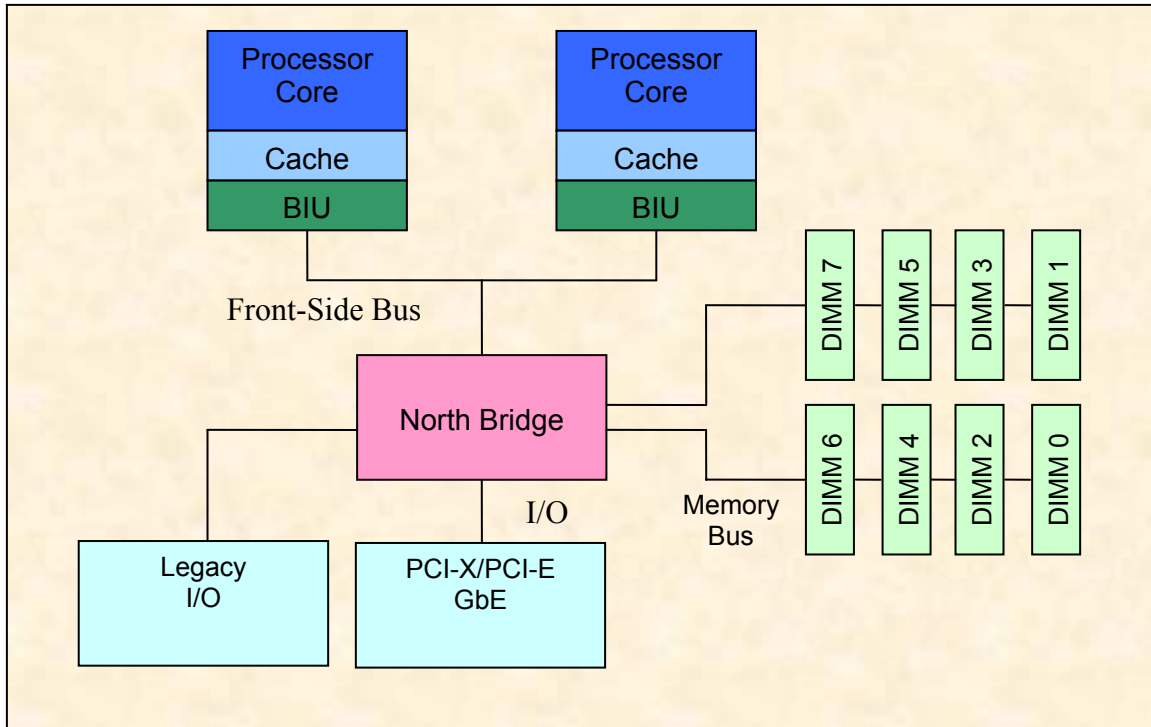

Figure 2. IBM eServer xSeries 336 Block Diagram

Systems based on the Intel Xeon processor follow an SMP architecture, with a shared Front-Side Bus (FSB) connecting the processors to a single controller, called a north bridge. Each processor communicates with the front-side bus through its Bus Interface Unit (BIU). The Arithmetic Logic Unit (ALU) contains two simple integer units and one Floating-point Multiply Add (FMA) unit, all deeply pipelined. Current processors have a 1MB or 2MB L2 write-back cache.

The north bridge is the heart of the system. It handles traffic between the processor and memory, control traffic between the processor and I/O devices, and data traffic between I/O devices and memory. In Figure 2 the central position of the north bridge can be seen, as can the shared front-side bus. The two channels to memory can also be seen. These components play the dominant role in determining memory performance.

A single processor is capable of saturating the front-side bus, so the second processor could compete for memory bandwidth. The processors also share memory bandwidth with I/O devices, including intercommunication devices.

Notice that the north bridge is a completely separate device from the processors, and is clocked separately from the processors. The north bridge clock is tied to the speed of the front-side bus, so even as processor clock rates increase, the latency to memory remains virtually the same.

The speed of the front-side bus places an upper bound on the rate at which a processor can send data to or receive data from memory. In fact, front-side bus bandwidth is often tuned to match the bandwidth of available memory technology at the time. It is expected that a single processor will **not** saturate the front-side bus over time because the processor has a cache where data most likely to be referenced are stored. Cache reduces the front-side bus pressure, so there is capacity to allow more than one processor to operate on the front-side bus. A two-processor Xeon system has a front-side bus that is 8 bytes wide and clocked at 800 MHz. Its memory controller has two

channels, each 8 bytes wide, to DDR-II 400 memory. This gives the front-side bus and the memory bus 6.4 GB/s of bandwidth each.

Nothing in this abstract architecture inherently limits the number of memory channels, or the size of cache, or the speed of the front-side bus. In the real world, the limit is its cost. For a bus to be faster it must be wider or it must be clocked at a faster rate. To make a cache larger requires more transistors (i.e., more silicon). All of this adds cost to the processor or system. Common practice today is to match the front-side bus bandwidth to the memory bus bandwidth. It is a cost-effective solution that, nevertheless, impacts performance on data-intensive applications.

The I/O subsystem supports two 100 MHz PCI-X slots, or one 133 MHz PCI-X slot, or one PCI-Express (PCI-E) x8 slot. A PCI-X bus is 64 bits wide, and it allows transmission in only one direction at a time. Thus a 100 MHz slot is capable of transmission rates no greater than 800 MB/s. A 133 MHz slot is limited to 1067 MB/s. PCI-Express is clocked at 2.5 GHz, and uses two dedicated buses that transmit in opposite directions. It uses a 10b/8b encoding scheme, so the hardware peak bandwidth of a PCI-E 8x slot is 2 GB/s in each direction. In practice, the transmission protocols of both PCI-X and PCI-E consume a portion of the bandwidth, and the peak realizable throughput is approximately 90% of the hardware peak.

Opteron and e326 Architecture

Opteron has a different type of architecture, NUMA rather than SMP. The memory controller is integrated into the processor. This can be an advantage for two reasons. First, the memory controller is clocked at the same rate as the processor. So, as the processor speed is increased, the memory controller speed is also increased, reducing the latency through the memory controller, allowing faster access to memory. Second, when a processor is added to a system, more paths to memory are also added. As the demand for memory bandwidth increases due to the additional processor, more bandwidth is available to satisfy that demand.

The architecture of the Opteron processor is shown in Figure 3. As in the previous diagram there is a processor core and cache. But in place of a bus interface unit and an external memory controller, there is an integrated memory controller (MCT), an interface to the processor core (SRQ), three Coherent HyperTransport (cHT) units and a cross bar switch to handle routing of data, commands and addresses between them. The memory controller supports up to two 8-byte channels to memory. The processor is able to support up to 400 MHz (DDR-I 400) registered Error Correcting Code (ECC) memory. At 333 MHz (DDR-I 333), the memory bandwidth is 5.3 GB/s across both channels; at 400 MHz it is 6.4 GB/s. The cHT units may be used to connect to I/O devices or to other processors. The protocol used for routing memory traffic is somewhat more elaborate than what is used for I/O, but the I/O protocol is a proper subset so cHT links may be used for either purpose.

Note once again that every device within the processor package is clocked using a synchronized clock. As the processor clock is increased from one generation or processor speed bin to the next, the memory controller clock is automatically increased at the same rate. This has the advantage of decreasing the latency of a memory request from the processor core to the memory, which speeds access.

The Opteron ALU is similar to the Xeon ALU with the following important exceptions. The Opteron ALU contains three simple integer units and one FMA unit instead of two and one, as Xeon has. All units are also deeply pipelined, but because the processor frequency is about two thirds that of Xeon, there are correspondingly fewer stages in Opteron's pipelines. Shallower pipelines offer somewhat improved efficiency because the penalty of a pipeline stall is not as great as with deeper pipelines. Finally, current processors have only a 1MB L2 write-back cache, whereas Xeon also offers a 2MB L2 cache.

Figure 3. Opteron Processor Block Diagram

The cHT links are point-to-point connections. Links are never shared. A processor can be directly connected to only as many processors as it has available links. In theory it can be indirectly connected to an arbitrary number of processors in a ring, or tree or other topology, but in practice the number of processors in a system is limited.

Each link consists of two unidirectional paths, each 2 bytes wide, with a small number of control lines for sending commands and status. Coherent HyperTransport links today operate at 800 MHz, with speeds increasing to 1 GHz and beyond in the future. Data transmissions are "double pumped"; that is, data are transmitted on both the rising edge and on the falling edge of the clock.

It is this combination of integrated memory controller and point-to-point links that is one of Opteron's strengths. Because of this, adding more processors to a system automatically adds memory bandwidth. Thus there is less risk of oversubscribing a shared resource like the front-side bus of the previous architecture.

The e326 is a 1U dual-processor Opteron system. It is rack-optimized and designed to be used as a computational node in a scientific cluster. It supports up to 16GB of main physical memory in eight DIMM slots, two IDE or two hot-swap SCSI drives, two Gigabit Ethernet (GbE) ports, and either two 100 MHz PCI-X slots or one 133 MHz PCI-X slot. It does **not** support a PCI-E slot. Processor speeds currently range from 2.0 GHz to 2.6 GHz, and the system supports 333 MHz and 400 MHz DDR-I memory. Figure 4 is a block diagram of the system.

The system must be loaded with one, two or four DIMMs on processor A, and may have any of zero, one, two or four DIMMs on processor B. All memory must be of the same speed, that is, DDR333 or DDR400. Pairs of DIMMs – DIMMs 0 and 1, 2 and 3, *etc.* – must also have the same size. The best performance is obtained by filling all eight DIMM slots with DDR400 memory.

Memory Performance (STREAM)

Memory performance is typically a complex topic because the memory subsystem is a complex system. The processor must generate a request to read or write data in the ALU and pass that request on to the various layers of cache. If the request cannot be satisfied in cache, it is forwarded on through various mechanisms to the memory controller (e.g., north bridge for Xeon

or MCT for Opteron). The memory controller may hold incoming read or write requests in a queue while it completes other requests currently in progress. As new requests come in, they are checked against existing requests in the queue. If the new requests are related to requests currently in the queue, they can be combined or reordered to save time.
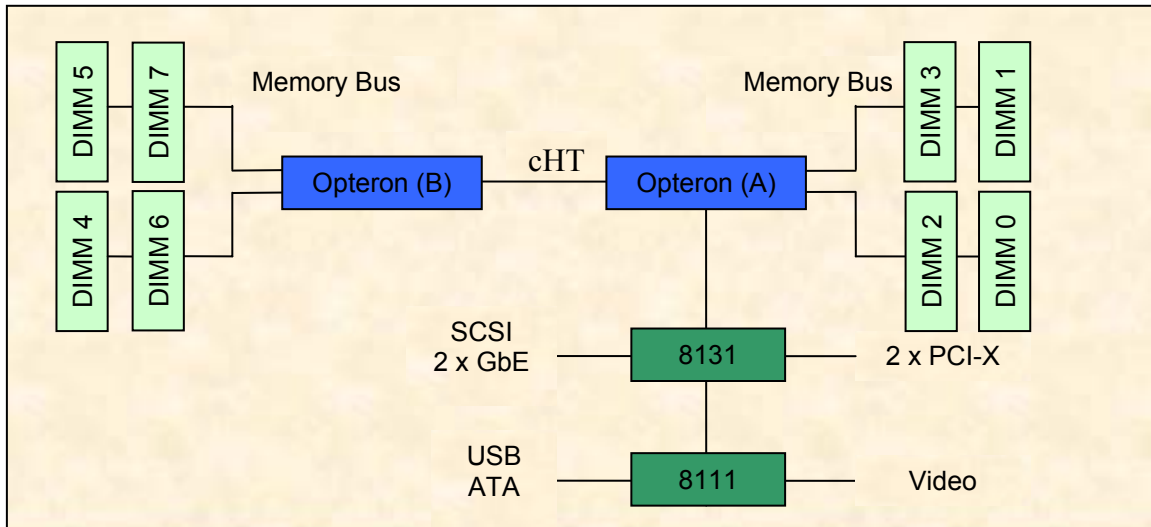

Figure 4. IBM eServer 326 Block Diagram

Once the memory controller is ready to issue a read or write request, the address lines are set and the command is issued to the DIMMs. The memory address lines are multiplexed to reduce hardware costs, and the addresses are divided into row addresses and column addresses. A row address is the upper half of the address (e.g., the upper 16 bits of a 32-bit address), whereas a column address is the lower half of the address. The column address must be set first; then the row address is set.

When two requests have different column addresses but use the same row address, they are said to occur in the same page. When multiple requests to the same page occur together, the memory controller can set the row address once, and then change the column address as needed for each reference until the page is no longer needed. The act of changing a column address is referred to as Column Address Set, or CAS. An important measure of memory speed is CAS Latency, or CL.

CAS latency measures the number of memory clocks that elapse between the time a memory controller sets the column address to request a line of data, and when the DIMM is able to respond with that data. Lower numbers indicate faster memory. CL values of 2.5 or 3.0 are typical of current technology. Numbers with fractions are possible because data may be clocked at a different rate than commands. With Double Data Rate (DDR) memory, data is clocked at double the speed of commands. For example, 400 MHz DDR memory has a data clock of 400 MHz, and a native clock (command and address) of 200 MHz. Thus CL2.5 memory has a CAS latency of 2.5 command clocks, which is equivalent to 5 data clocks.

The significance of this discussion is that memory performance is highly dependent upon not just whether the data is in cache or main memory, but also how the access patterns appear to the memory controller. The access pattern will strongly affect how the memory controller reorders or combines memory requests, whether successive requests hit the same page, *etc.* Programs dominated by unit stride accesses, such as the STREAM benchmark, may have better memory performance than programs that access memory randomly, but it is a complicated question to answer.

From the above presentation it should be clear that memory performance is affected by a number of complex factors. Those factors may include the choice of architecture, processor frequency, memory frequency, the number of DIMMs in the system, and whether the system is set up as a NUMA or an SMP system. We present our results considering each of those factors.

Our first set of results considers the nominal performance of each architecture. The x336 was configured with eight 1GB 400 MHz DIMMs of DDR-II memory. The e326 system was similarly configured with eight 1GB 400 MHz DIMMs of DDR-I memory. Both systems used optimal BIOS settings. The x336 was booted with an SMP operating system, whereas the e326 was booted with a NUMA-aware operating system. For each system we ran two tests. The first test used a single processor. The second test used two processors with a separate thread for each processor. The results are shown in Figure 5.
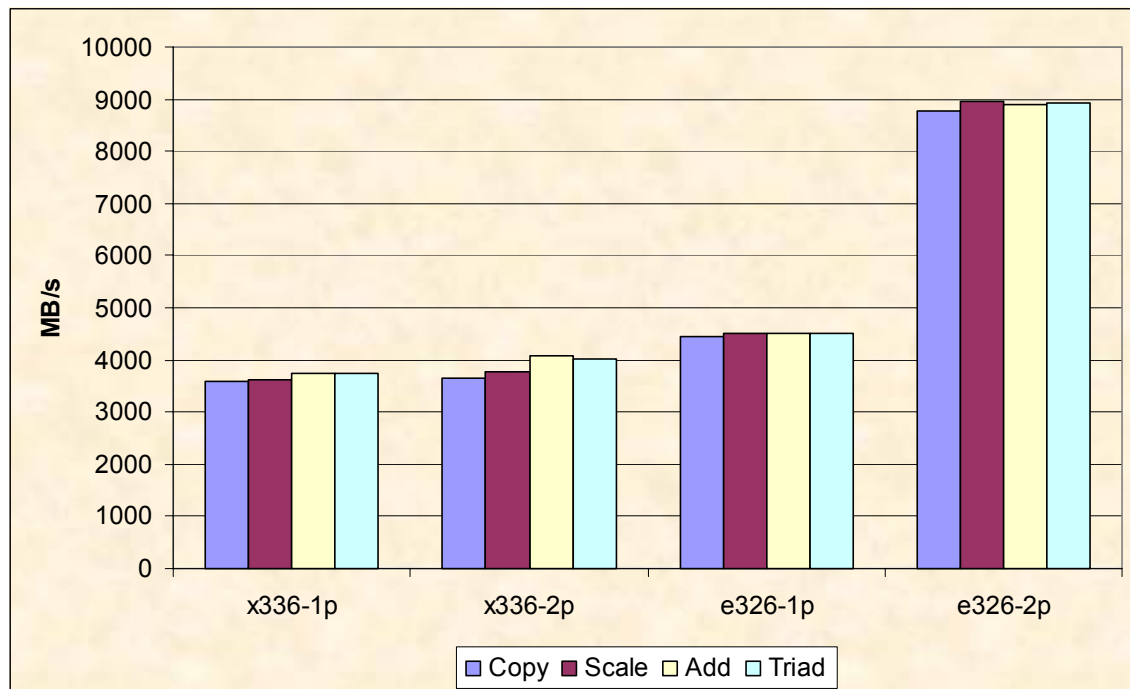
Figure 5. STREAM Results for the 3.6 GHz x336 and 2.4 GHz e326 Servers

We can see from Figure 5 that there is a substantial difference in memory throughput between the x336 and e326. The differences shown are primarily due to the architectural differences between the systems. For example, the largest difference occurs between the two-processor e326 results (e326-2p) and all other results (x336-1p, x336-2p and e326-1p). The reason is that the e326-2p test brings four memory channels into play, whereas the others make use of only two memory channels. The x336 has only two channels to use, and the e326-1p test makes use of only two of the available four channels. The NUMA-enabled operating system allocates memory for each thread in the physical memory that is attached to the processor where the thread is running, referred to as local memory. Because of that, the data is always accessible in the fastest possible way. A later test will show the effects of mixing local and remote references.

The second effect to notice is the difference between the x336 and the e326-1p tests. Both tests have the same number of memory channels, and the DIMM frequency is the same, 400 MHz. The differences are in the memory controllers and the type of memory used. It would be very difficult to objectively measure only the differences between the memory controllers alone because there is no way to separate the memory controllers from the rest of the system and place them in identical environments. These tests show roughly 16% to 18% difference between the x336 and e326, depending on the test, in favor of the e326. From careful examination of the

DDR-I and DDR-II memory architecture (not presented here), we believe there is an inherent performance advantage to DDR-I memory of about 5%. That leaves around 10+% difference due to the memory controller itself.

The third effect to notice is that there is a slight difference in performance between using the x336 with one processor and with two. This difference is due to the higher loading of the memory controller. The higher load allows more requests to be queued within the memory controller. This gives a little greater opportunity to take advantage of write-combining and prefetching, but only up to a point. As more threads attempt to fetch or store data to memory, the addresses appear more random to the memory controller, and that quickly interferes with its performance. Because the difference between one thread and two is so slight, we will not always report them separately, but results will be labeled as appropriate.

Our final observation from this data is that the four parts of the STREAM benchmark give very similar results in our tests. Even though the mix of reads and writes is different between the four tests, the results are similar enough that we only need to concern ourselves with one of the tests. For the remainder of this paper we will focus on the TRIAD test, though we could have chosen any of the other three and drawn the same conclusions.

Next we analyze memory performance by memory frequency. This is really an issue only for the e326, and not for the x336. The e326, as mentioned before, uses DDR-I memory. Specifically, it uses registered Error Correcting Code (ECC) memory designed for servers. DDR-I memory has been available for several years and is available in a variety of speeds, including DDR200, DDR266, DDR333 and DDR400. Unfortunately, DDR-I is at its end of life and will not be extended further. Non-registered DDR memory is available in faster speeds, but such memory cannot be used in servers like the e326. The e326 supports the use of 333 MHz and 400 MHz registered ECC memory. The effects of using different speeds of memory are shown in Figure 6.
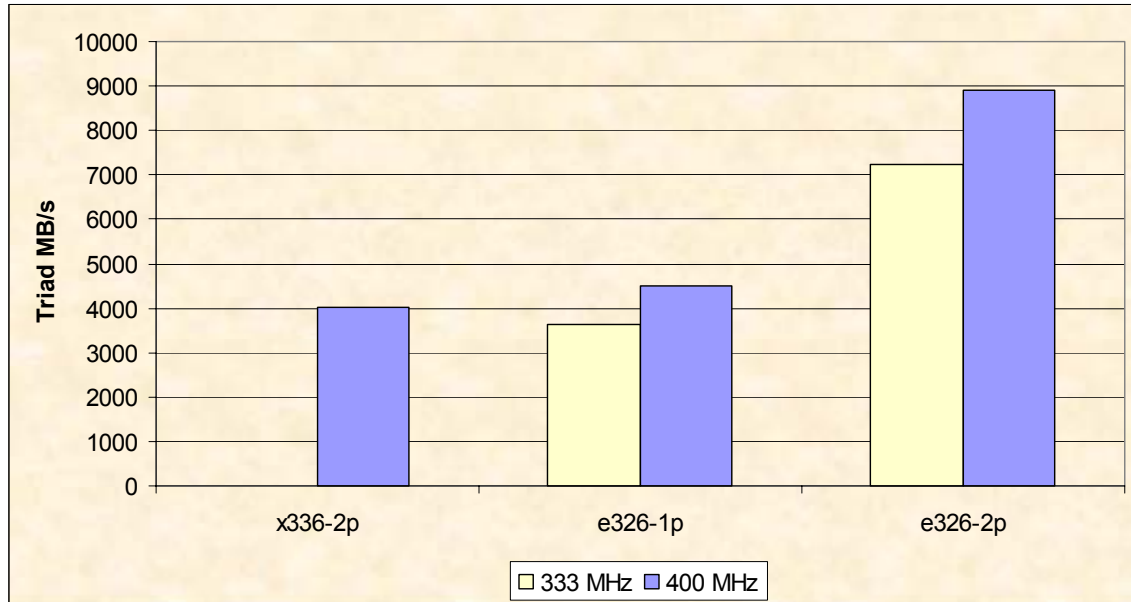
Figure 6. Comparing 333 MHz and 400 MHz Memory

The x336 also uses registered ECC memory, but it uses DDR-II memory rather than DDR-I. DDR-II is just beginning its life, and so there is only a single speed available, namely 400 MHz. Its performance is included in Figure 6 for reference.

The interesting feature of Figure 6 is not so much that there is a difference between faster and slower memory, but that the difference in benchmark performance is slightly **greater** than the

difference in memory speed. Memory of 400 MHz is, on the surface, exactly 20% faster than 333 MHz memory. But in both the 1p and 2p results, the benchmark runs about 23% faster using DDR400 memory than DDR333. The explanation is two-fold. First, the 400 MHz memory is CL3.0 memory, while the 333 MHz memory is CL2.5. But that would make the difference **less** than 20%, not **greater**. The second piece of the puzzle holds the key.

Memory must talk with memory controllers over a synchronous interface; that is, both must use the same clock. Xeon solves this problem by placing communication between the processor and memory controller on an independent bus, the front-side bus. Traffic between the memory controller and the DIMMs is over a separate bus. The memory controller is clocked relative to the two buses, and is generally independent of the processor. The processor bus interface unit must match the processor speed to the front-side bus speed, and it requires that the processor clock be an integer multiple of the front-side bus command clock (200 MHz). So the memory clock drives the north bridge clock requirements, which drives the front-side bus clock requirements. No matter whether the processor is clocked at 2.8 GHz or 3.6 GHz, the front-side bus remains constant and the memory controller is clocked at a constant rate, consistent with the memory speed.

Opteron must also use a synchronous interface, but additional problems arise because the memory controller is integrated into the processor chip, and driven by the processor clock. This has the advantage that faster processors also enjoy shorter memory latencies due to the shorter trip time through the memory controller. But the memory must still communicate with the memory controller over a synchronous interface. That requires the memory controller clock speed to be an integer multiple of the memory clock speed, just as it is for the Xeon bus interface unit. If the rated frequency of the processor is not an integer multiple of the rated memory frequency, the memory frequency must be slowed until the integer-multiple relationship holds.

At 2.4 GHz that means a 333 MHz DIMM must be slowed to 319 MHz on the data clock, or about 96% of its peak speed, in other words a little over 20% slower than 400 MHz DDR memory. The amount of performance loss due to this gearing effect varies with the processor speed, and does not exist at all when 400 MHz memory is used.

As mentioned above, the Opteron memory controller is integrated directly into the processor and driven at processor clock rates. This allows memory read and write requests to pass through the memory controller more quickly for faster processors. However, a major portion of the time is still spent within the memory DIMMs themselves, so it is important to know how much performance is improved, if any. The memory performance compared by processor clock frequency is shown in Figure 7.

Notice that as Opteron frequency increases, so does the memory bandwidth. In this case, the processor frequency increased by 20%, and the memory bandwidth increased by a little over 5%. The improvement becomes less pronounced at higher processor clock frequencies. It can be also seen from Figure 8 that the same performance improvement (a little over 5%) holds with processor clock speed when two processors are used.

It is also interesting to note that there is even a slight effect for Xeon processor-based systems (see Figure 7). This is due to the fact that the processor is able to generate addresses and place them on the bus more quickly. The effect is small, but noticeable.

Memory controllers also occasionally have opportunities for additional performance optimizations. The Intel memory controller uses an optimization technique based on the number of DIMMs available in each channel. Any number of DIMMs, up to four, may be used in each channel, for a total of eight to a system. (DIMMs must be matched across channels for size and speed, and dual-ranked DIMMs count as two each.) But the memory controller is able to operate more efficiently when there are exactly four DIMMs in each channel. The engineers who developed this technology do not discuss the details, but the results are shown in Figure 9.
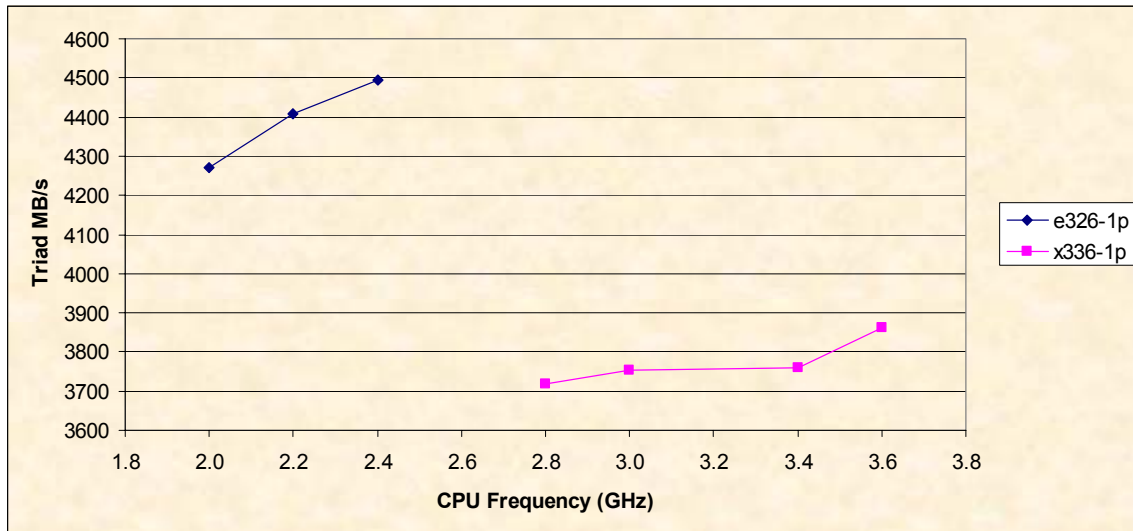
Figure 7. x336 and e326 1P DDR400 Memory Performance by Processor Frequency
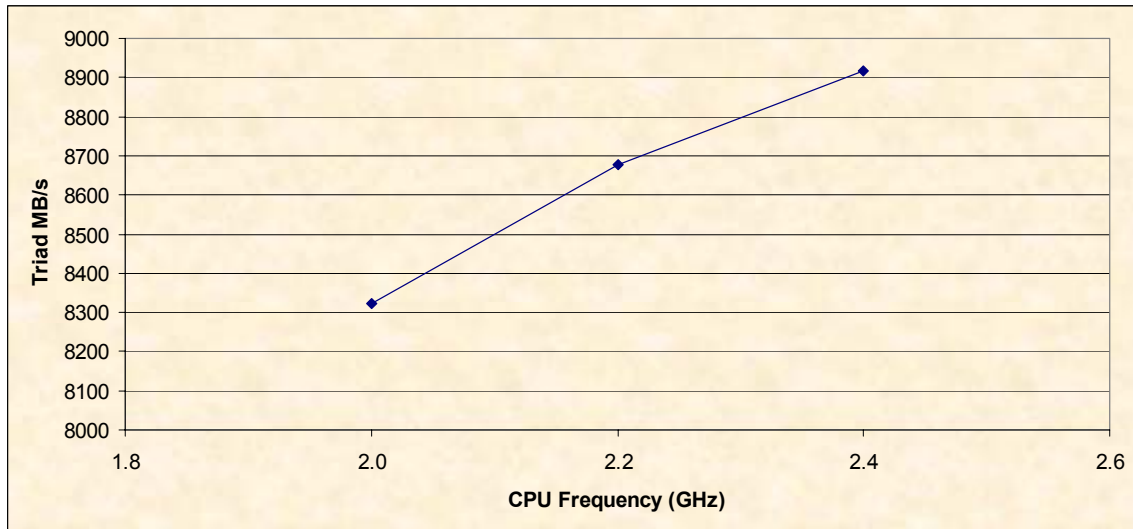

Figure 8. e326 2P Memory Performance by Processor Frequency

For the Intel processor-based system, the performance improves by approximately 20% when eight DIMMs (four per channel) are used, over any smaller number of DIMMs. The Opteron processor-based system does not have this optimization, and its performance improves much less, around 6%, when all DIMM slots are filled. Although it is not shown here, as one might expect, the Opteron processor shows the same improvement when two processors are used.

The last effect we will describe regarding memory concerns NUMA optimizations. Because the x336 is an SMP system, this discussion only focuses on the e326. Opteron supports a mode called *node interleave*. When node interleave is **disabled**, the memory controller maps the local memory of each processor to a single contiguous range of physical addresses. This allows the operating system to map user data to local memory, whenever possible, to allow programs to access data the most rapidly. When node interleave is **enabled**, physical addresses are partitioned into 4KB blocks, and alternated among the processors. The operating system is then unable to use NUMA optimizations, and the memory space is treated as if the system were an SMP system.
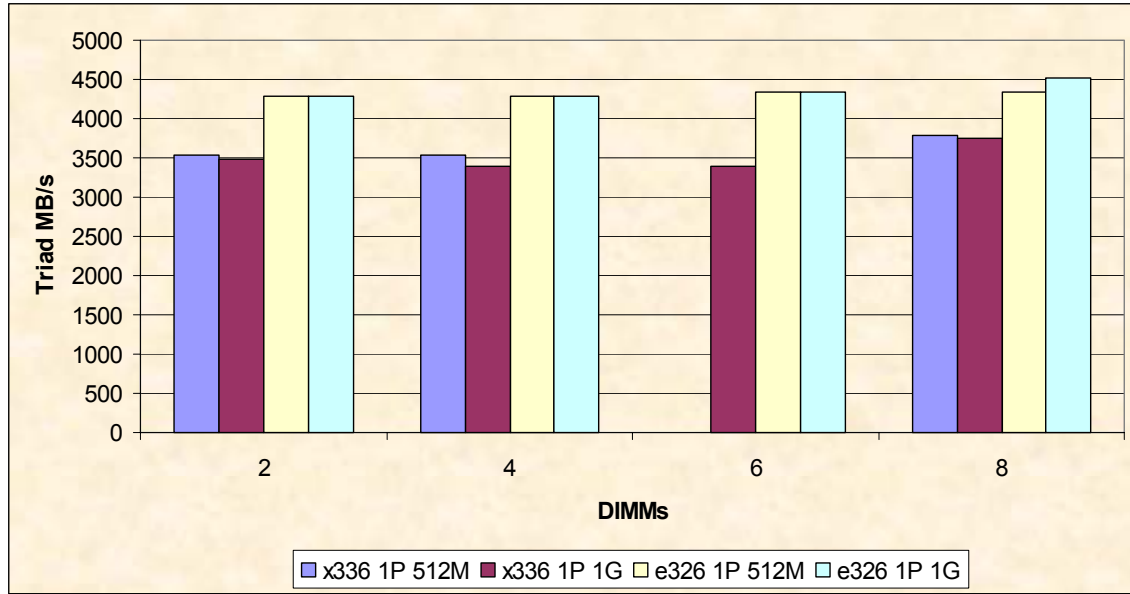
Figure 9. Memory Performance by Number of DIMMs

We did measure the performance of a system configured in this manner, and the difference is quite large. This only applies to multi-processor systems, but dual-processor results are significantly slower, when node interleave is enabled, than the same systems operating in NUMA mode. For example, the TRIAD results for two threads and two processors are approximately 5000 MB/s when node interleave is enabled, compared to 8900 MB/s when operating in NUMA mode.

This performance difference is the result of using the system in SMP mode rather than NUMA mode. Disabling node interleave and using the same kernel in SMP mode give the same maximum performance, but with significantly wider performance variation. With node interleave enabled, memory access alternates consistently between near and far memory pages. With node interleave disabled and an SMP operating system, memory access may still be to near and far memory pages, but there is no consistency and, therefore, the access time varies a lot.

Floating-Point Performance (Linpack and SPEC CFP2000)

In this section we discuss the performance of the processor core. We focus most of our attention on floating-point performance. Processor core performance is achieved when data used in calculations are mostly found in processor registers and local processor cache. In those cases system performance is limited by the speed of the processors. The speed of the processor is determined by the number of functional units, the clock rate at which they are driven, and how efficiently they are used. When data are close to the processor, efficiency can be above 80%. When many branches occur, or data must be fetched from main memory, efficiency can be considerably lower.

Xeon and Opteron processors both have a single Floating-point Multiply-Add (FMA) unit that is deeply pipelined. An FMA unit can produce one multiply and one addition result in alternating clocks. The FMA unit, called the SSE2, executes as a short vector unit. It is capable of issuing two 64-bit add operations and two 64-bit multiply operations on alternating clocks. When single precision (32-bit) results are needed, it can issue four add and four multiply operations on alternating clock cycles.

The version of Linpack we use is the High Performance Linpack benchmark. For each test in this set of experiments, we use two processes on one system. Our aspect ratio is always P=1 x Q=2

because that gives the best performance. The number of blocks distributed between the processes (NB) varies from test to test in order to find the best possible result. We were not exhaustive in our attempts to optimize NB, so it is possible to find better results.

Figure 10 shows the results of our experiments with the Linpack benchmark. Several things stand out in Figure 10. The most obvious is that while there is some variation, performance is clearly proportional to clock speed of the processor. Because of its superior clock speed, Xeon has a very strong advantage over Opteron, and thus is a very good choice when applications are able to cache their data and have a high proportion of floating-point operations.
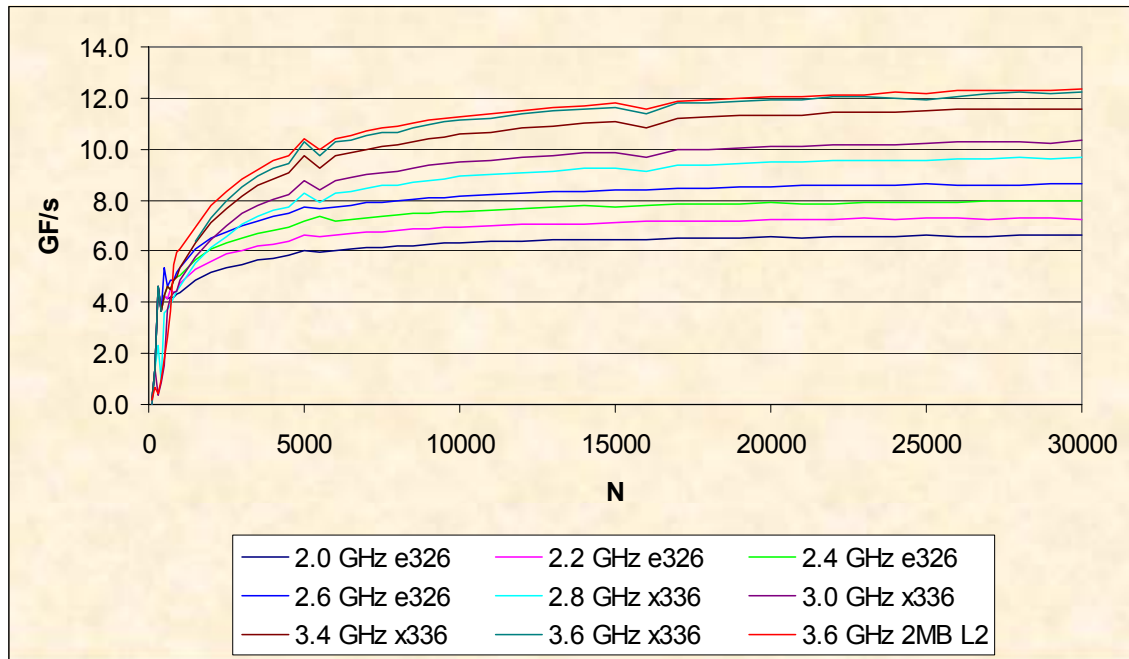
Figure 10. x336 and e326 Linpack Performance

The second observation is that there is considerable dependence on the size of the problem (N). The operation completes so quickly for small values of N that the results reflect the benchmark start-up costs and timer granularity, rather than the performance of the intended operations.

The third observation is that there is a noticeable dip in performance near N=1000. This comes about in the following way. The benchmark generates a pseudo-random matrix of the specified size, and measures the amount of time required to solve it. Matrices that are sufficiently small are loaded into processor cache when the matrix is generated and remain there for the duration of the benchmark. Matrices too large for cache are expelled from cache as they are generated, and must be reloaded when the matrices are solved. The cost of reloading data into cache is especially noticeable with small matrices because the number of floating-point operations is so low. As the problem size becomes larger, the number of operations performed each time a datum is loaded into cache increases. In short, the dip in performance occurs at the transition where the data no longer fits completely within processor cache.

Linpack represents a class of benchmark where operations are heavily floating-point-intensive and data are highly cacheable. While this is representative of some applications, many scientific applications are floating-point-intensive but have data that is not as cacheable. For example, applications (e.g., Computation Fluid Dynamics), that solve partial differential equations often use techniques that require sweeps across multiple spatial dimensions, and that precludes high cache hit rates for large data sets. For this reason we also present performance results from the SPEC CFP2000 benchmark suite in Figures 11 and 12.

The SPEC CFP2000 suite uses 14 separate applications representing a range of scientific work loads. Included are Computation Fluid Dynamics (CFD), ocean modeling, facial recognition, seismic codes, and others. Performance scores are classified as base or peak scores, with the difference being the number of compiler optimization flags that are used. Base results are limited to four optimization flags, whereas peak scores are unlimited. We are interested only in the peak scores.

SPEC CFP2000 scores are also divided into speed and rate scores. Speed scores are obtained by sequentially running several copies of each application to completion. Each application must be run an odd number of times (but at least three times), and the median score for that application is selected. The benchmark score is the geometric mean of the individual application scores, i.e., it is the 14th root of the product of the 14 application scores. Rate scores are obtained by concurrently running some number of copies to completion, and the benchmark score is the geometric mean of the application median scores.

The significance of this is that the speed runs are sequential in nature and do not load the memory subsystem as heavily as the rate scores. Such scores reflect applications where fast turn-around times are needed. Rate runs load all processors and the memory subsystem as fully as possible. Individual application turn-around times may suffer, but system throughput will be better. Speed runs are viewed as reflective of a desk-top or dedicated environment, while rate runs are considered reflective of a more batch-oriented environment.

Figure 11 shows the results of speed runs by processor frequency[3]. It is interesting to note that in floating-point-intensive benchmarks, such as Linpack, the Xeon has a significant advantage, but that advantage is not reflected here. The ALU operations are still predominantly floating-point in nature, but the data must be fetched from memory much more often than with Linpack. As a result, the performance is very similar between Opteron and Xeon, with Opteron, perhaps, having a slight edge. This should not be a surprise because the 1P STREAM results were similar between Xeon and Opteron with about the same difference in performance between them.

Figure 12 shows the rate results by processor frequency[4]. Again, it is clear that while processor performance is important, memory performance dominates here. The rate results reflect both processors being used, and while single-processor performance is similar, dual-processor performance is not. Opteron's second pair of memory channels is used to its advantage. Its 2x memory bandwidth gives about 30% better benchmark performance. One can also see from the 3.6 GHz results that increasing the L2 cache from 1MB to 2MB gives 10% performance boost.

Interconnect Performance Limitations

So far we have presented detailed information on processor and memory subsystem performance of individual systems. While that is important, the key characteristic of clusters is that the systems are connected together so that they can "talk" to each other. Servers talk by sending messages over some fabric, usually using a communication protocol such as Message Passing Interface (MPI).

---

[3] The IBM x336 (1 x Intel Nocona processor with 1MB L2 cache) running SPEC CFP2000: 2.8 GHz measures 1479, 3.0 GHz measures 1539, 3.4 GHz measures 1653, 3.6 GHz measures 1721, 3.6 GHz with 2MB L2 cache measures 1784. The IBM e326 (1 x AMD Opteron processor): 2.0 GHz measures 1436, 2.2 GHz measures 1676, 2.4 GHz measures 1771, 2.6 GHz measures 1939. The above results were obtained using SUSE LINUX SLES 9 and the PathScale EKO Compiler Suite.

[4] The IBM x336 (2 x Intel Nocona processors) running SPEC CFP2000 Rates: 2.8 GHz measures 26.1, 3.0 GHz measures 26.8, 3.4 GHz measures 27.8, 3.6 GHz measures 28.6, 3.6 GHz with 2MB L2 cache measures 31.5. The IBM e326 (2 x AMD Opteron processors): 2.0 GHz measures 32.9, 2.2 GHz measures 37.9, 2.4 GHz measures 39.9, 2.6 GHz measures 44.2. The above results were obtained using SUSE LINUX SLES 9 and the PathScale EKO Compiler Suite.
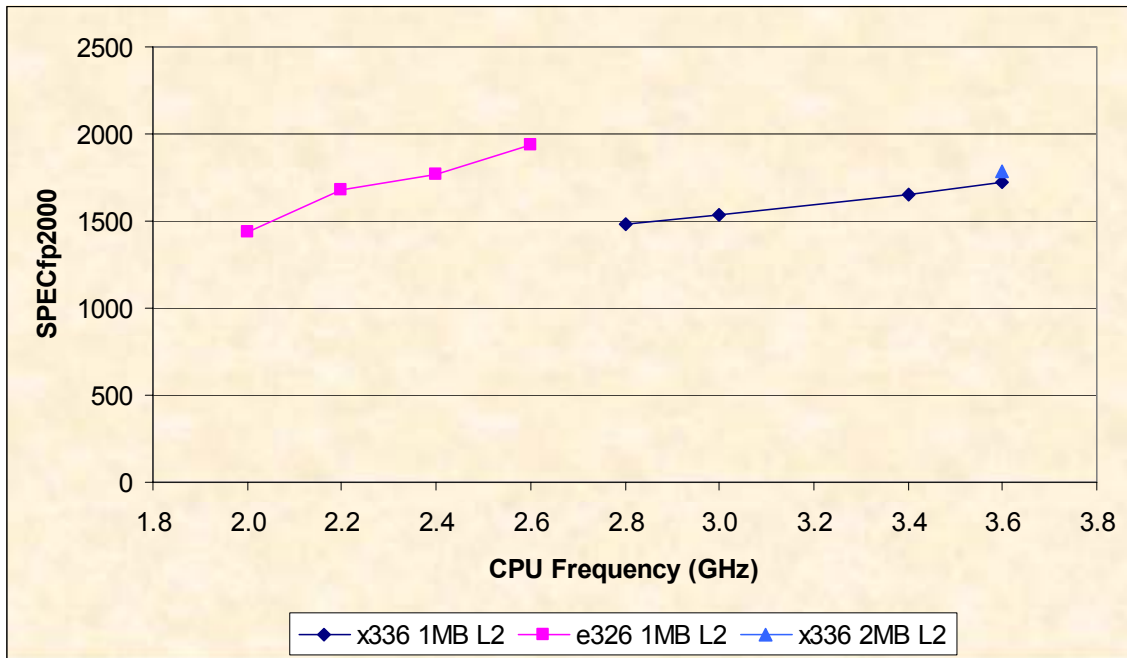
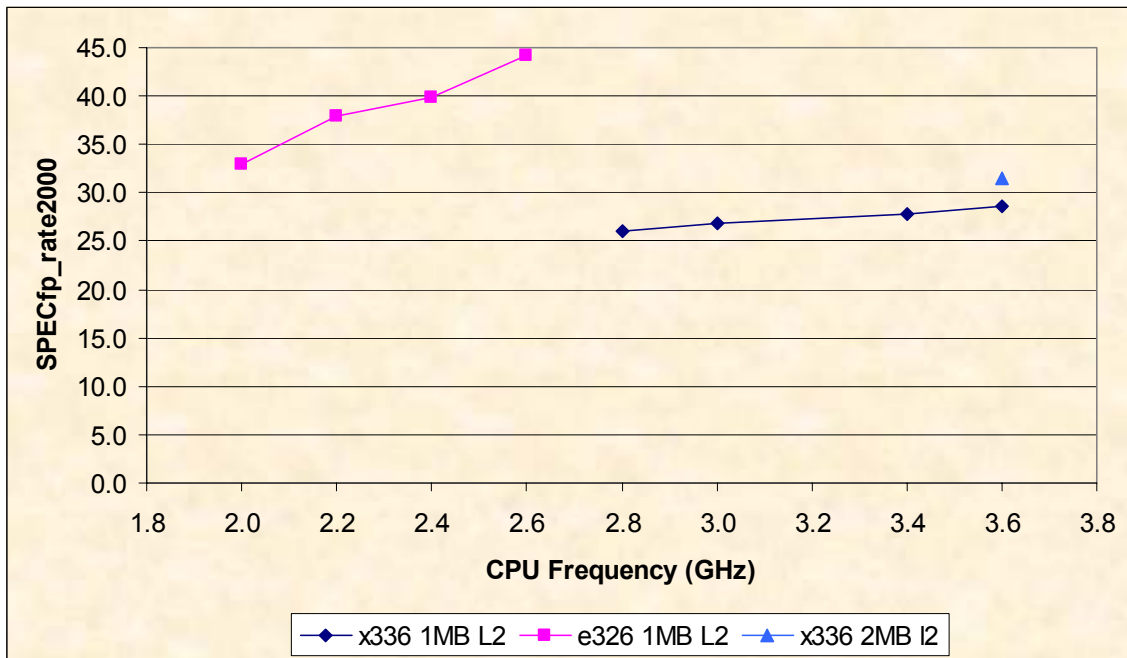*Figure 11. x336 and e326 SPEC CFP2000 Speed Performance*

Figure 12. x336 and e326 SPEC CFP2000 Rates Performance

The flow of data and control through the stack may vary depending on the network and software stack that is used. Figure 13 illustrates two such stacks, Ethernet using TCP/IP, and Myrinet using GM drivers. Other hardware and software stacks are available, but at this level of description they usually resemble one of these two.

Let's consider Ethernet first. When an application sends data to another system, it prepares the data and passes it to the message-passing library. In this illustration the library is MPICH, which

is a freely available implementation of MPI. MPICH further prepares the data and passes it to the operating system with a request to send it out over TCP/IP. The TCP/IP stack breaks up the message, if necessary, into packets no larger than the Maximum Transmission Unit (MTU). It supplies the appropriate headers, addresses and transmission-error-detection fields for the packets. The MTU is typically approximately 1500 bytes, but may be larger or smaller. The packets are then transferred from memory to the hardware adapter for transmission.

When a packet is received, the adapter may interrupt the operating system to request a buffer, or in some cases it may fill a pre-allocated buffer. Once the buffer is allocated and filled, the adapter must interrupt the operating system to indicate that a packet has been received. The operating system then passes the packet to the TCP/IP stack for processing. Once processing is complete the data must remain in a protected buffer until the application asks for it, after which the data can be discarded.
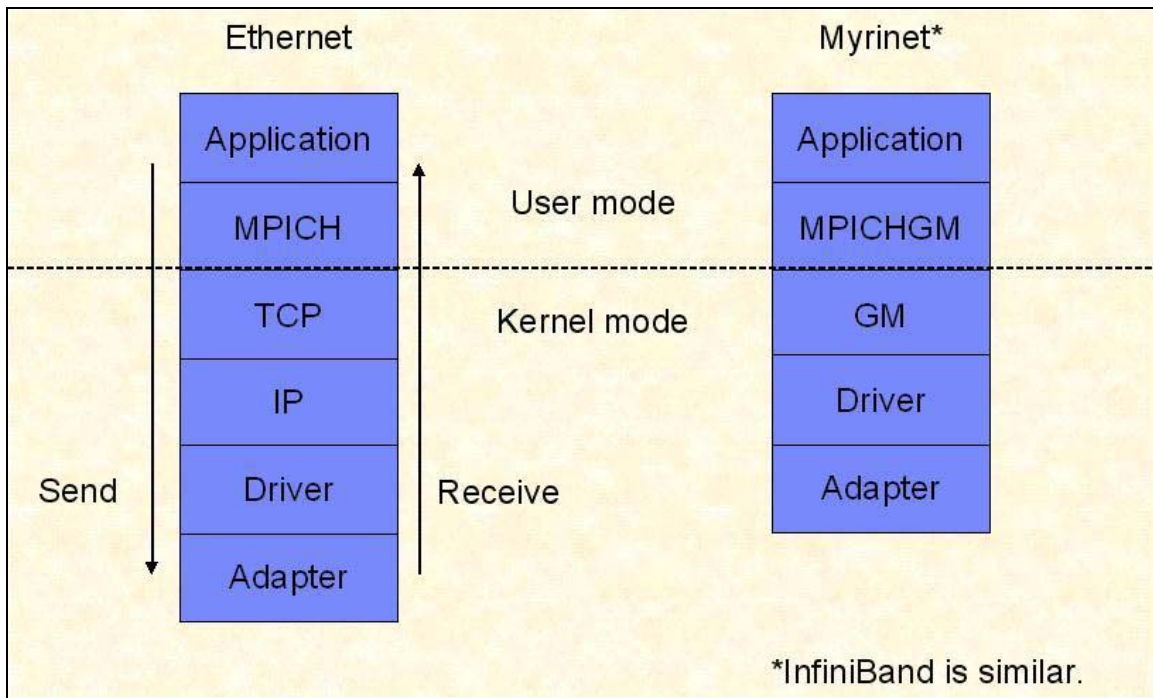
Figure 13. Data and Control Flow in Message Passing

From this high level, the flow for Myrinet looks very similar. Over Myrinet the application hands the message to MPICHGM, which calls the operating system to hand it to GM. GM processes the data further, breaking it into packets, if necessary, then hands it to the adapter driver. The driver passes the data over an I/O bus to the adapter, which transmits the data to its destination.

Some of the differences are that GM replaces the TCP/IP stack (the GM protocol is more efficient than TCP/IP), and the hardware and drivers are different. Similarities include the way in which interrupts are used to signal the operating system that data transfers have completed, and the fact that all data passes over an I/O bus from memory to the adapter, or from the adapter to memory.

It is this last detail—that all data must pass over an I/O bus—that is of the greatest interest here. Each network type has different speeds and functions, some faster, some slower. But all are limited by the speed of the I/O bus used to connect the adapters to memory. One may argue convincingly, for example, that Quadrics is faster than Myrinet, but if both are attached to a bus that is slower than either of them, the discussion is moot. Furthermore, the intent of this paper is

to discuss server performance rather than adapter performance, and the I/O bus is the main component of the network that is part of the server.

Network cards are available for both PCI-X 1.0 and PCI-Express I/O buses. PCI-X is a 64-bit version of the 32-bit PCI bus. PCI-X is a shared bus that uses a reversible, unidirectional protocol. In other words, more than one I/O card can share the bus. Also, transmission from adapter to server, or from server to adapter, uses the full bandwidth of the bus, but transmission can take place in only one direction at a time. For transmission to take place in the opposite direction, the bus must first become idle for a short period of time, and then the reverse transmission can begin. PCI-X 1.0 is available at speeds of 66 MHz, 100 MHz and 133 MHz, though 66 MHz is becoming less common. Two 100 MHz slots can share a single bus, but for electrical reasons, only one 133 MHz slot can be on a PCI-X bus. The bandwidth of a 66 MHz bus is 533 MB/s. A 100 MHz bus is capable of 800 MB/s, and a 133 MHz bus is capable of 1067 MB/s peak bandwidth. Efficiency varies with the adapter and application, but it can reach 90%.

PCI-Express (PCI-E) is a point-to-point bus; that is, it is not shared, and it is not reversible. There are two sets of connections, each traveling in opposite directions. Each connection is clocked at 2.5 GHz, and uses 10b/8b encoding. That means each connection is capable of transmitting at a rate of 2 Gb/s. A 1x (or one lane) PCI-E bus has one such pair of connections, and can transmit 2 Gb/s each direction for a total of 500 MB/s both directions. A four-lane PCI-E bus can do four times that, or 2 GB/s both directions.

Many network adapters are available for PCI-X and PCI-E. Adapters available for PCI-X include Gigabit Ethernet, Myrinet, InfiniBand and Quadrics. Gigabit Ethernet transmits, as the name says, up to 1 gigabit per second (Gb/s), or 125 MB/s, each direction. The total realizable performance is just below that, at 240 MB/s, driving both directions together. A dual-port Gigabit Ethernet card will not saturate a 100 MHz PCI-X slot, though a quad-port card may if driven to full capacity. Myrinet cards achieve about 500 MB/s in both directions on each port. Thus a one-port card does not exceed the available bandwidth of a 100 MHz PCI-X slot, but a two-port card is best placed in a 133 MHz PCI-X slot. Quadrics can achieve 950 MB/s or more with latencies below 2 microseconds on a 133 MHz PCI-X slot.

InfiniBand 1x is designed to have a peak bandwidth of 500 MB/s, both directions, or about the same as a single-port Myrinet card. These tend to be specialized cards for blade systems and are not generally available for PCI-X or PCI-E. InfiniBand 4x, however, is available for both. InfiniBand 4x enjoys a peak bandwidth of 2000 MB/s, both directions, which is far greater than any PCI-X 1.0 bus can manage. The PCI-X versions are necessarily bus limited. It does, however, match perfectly the performance of a PCI-E 4x slot.

In addition to the greater bandwidth, higher frequencies also permit shorter latencies. So applications that benefit from low latency connections benefit from higher frequency adapter slots, even when they do not saturate the available bandwidth. That is a significant reason for the popularity of high-end adapters like Quadrics. Generally speaking, Gigabit Ethernet can achieve 40-microsecond latencies on short messages. Myrinet and InfiniBand are both below 7 microseconds, and Quadrics can achieve latencies below 2 microseconds.

Now that we've outlined the significance of various I/O slots, we return to the question of server performance. The e326 has a single PCI-X bus that can support either two 100 MHz PCI-X slots or a single 133 MHz slot. It **does not** have a PCI-E slot. Opteron-based systems supporting PCI-E were not available at the time of this writing. The x336 also has a single PCI-X bus that supports either two 100 MHz or one 133 MHz PCI-X slots, but, in contrast, it **does** support an optional PCI-E 8x slot, capable of fully supporting a two-port InfiniBand 4x card. That makes the x336 the preferred choice when communication bandwidth is the critical concern, or if InfiniBand is preferred for other reasons. Either system would be appropriate for a low latency interconnect based on Quadrics.

Conclusions

Memory performance may be stated in terms of latency or bandwidth. For those applications that are sensitive to memory bandwidth, a single Opteron processor-based system, such as the e326, will have a small advantage over an equivalent Xeon processor-based system such as the x336. Our tests showed slightly more than a 10% difference. A dual Opteron processor-based system has a similarly small advantage when run in SMP mode, but a substantial advantage (greater than 2x) when run in NUMA mode. We believe the memory latencies to be similar between similarly configured Opteron and Xeon processor-based systems.

Both systems benefited from having all DIMM slots populated, but the Xeon processor-based system benefited more. The Intel memory optimization showed a 12% improvement when all DIMMs were used *vs*. any configuration that did not use all DIMM slots. The Opteron processor-based system also showed an improvement, but only about 6% over any other configuration.

Both processors have a single Floating Multiply-Add (FMA) unit in the processor core, but the core frequencies are different between them. Xeon supports a processor frequency that is 50% higher than that of Opteron. That gives Xeon a large advantage for applications that are core-limited in nature. Linpack is one example, as a popular benchmark, that benefits fully from the additional core performance. Linpack is almost exactly 50% faster on a 3.6 GHz Xeon than on a 2.4 GHz Opteron platform.

On the other hand, applications that have a mix of floating-point and memory operations, typified by the SPEC CFP2000 benchmark, tend to benefit from the additional memory performance from Opteron rather than the additional core performance from Xeon.

Last, if a fast interconnect is needed, there is some advantage to using a Xeon processor-based system such as the x336. Both systems support two Gigabit Ethernet ports and two 100 MHz or one 133 MHz PCI-X slots. This is appropriate for most interconnects such as the low-cost and reliable Myrinet, InfiniBand, or the ultra-low-latency Quadrics. But the x336 also supports a single 8x PCI-E slot, which can be used for high-bandwidth, low-latency InfiniBand interconnect fabrics. InfiniBand cards that are PCI-E-enabled are capable of bandwidths that are much higher than anything available over a PCI-X bus. At the time of this writing PCI-E is available only on Xeon processor-based servers, though that will change soon.

References

1. IBM eServer 326,
ftp://ftp.software.ibm.com/common/ssi/rep_sp/n/ESD00635USEN/ESD00635USEN.PDF.
2. IBM eServer xSeries 336,
ftp://ftp.software.ibm.com/common/ssi/rep_sp/n/XSD00361USEN/XSD00361USEN.PDF.
3. STREAM: Sustainable Memory Bandwidth in High Performance Computers,
www.cs.virginia.edu/stream/.
4. J. J. Dongarra, "The Linpack benchmark: An explanation," in A. J. van der Steen, editor,
Evaluating Supercomputers, pages 1-21. Chapman and Hall, London, 1990.
5. J.J Dongarra, Cleve B. Moler, G. W. Stewart, *Linpack User's Guide*, Society for Industrial &
Applied Mathematics, June 1979.
6. Linpack, www.netlib.org/linpack/index.html.
7. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, "LU Decomposition and Its
Applications," §2.3 in *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, 2nd ed.
Cambridge, England: Cambridge University Press, pp. 34-42, 1992.
8. E. Anderson, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenhaum, S.
Hammarling, A. McKenney, Sorensen D., Zhaojun Bai, Christian H. Bischof, *Lapack User's
Guide*, 3rd edition, Society for Industrial & Applied Mathematics, 2000.
9. Kazushige Goto, "High-Performance BLAS," www.cs.utexas.edu/users/flame/goto/.
10. SPEC CPU2000, www.spec.org/cpu2000/.