IBM e326m Performance for HPC

Douglas M. Pase
IBM xSeries Performance Development and Analysis
3039 Cornwallis Rd.
Research Triangle Park, NC 27709-2195
pase@us.ibm.com

Abstract

In this paper we examine the performance of single- and dual-core versions of the IBM® eServer™ 326m (e326m) server. Our analysis includes memory bandwidth and latency, floating-point vector performance, performance of the SPEC® CPU2000 speed and rate benchmarks, and both uni- and bidirectional traffic over the PCI-Express I/O slot using a high-speed InfiniBand™ adapter from Voltaire® with an MPI benchmark. In this paper we find that the processor-memory complex performs well. The PCI-Express slot also performs well under most loads, but high-bandwidth bidirectional traffic suffers somewhat with dual-core processors because of the single-threaded nature of device drivers and the lower clock speeds supported by dual-core processors.

1. Introduction

The IBM eServer 326m is the next generation of the two-processor Opteron system. It supports two processor sockets, two hot-swap SCSI or two IDE disk drives and up to 16GB of main memory in a rack-optimized 1U chassis.[1] The e326m supports improved electronics, including an improved I/O subsystem, an eight-lane PCI-Express slot and faster processors.

The e326 has an architecture that is similar to most two-socket system architectures. Two AMD Opteron processors are connected by a coherent HyperTransport™ link. Each processor has two channels to memory. A north bridge is connected to the master CPU, and a south bridge is connected to the north bridge. The south bridge provides access to low-performance I/O devices such as USB ports. High-performance I/O is connected through the north bridge. The e326m has two I/O buses, which support a combination of 133 MHz PCI-X and 8x PCI-Express. Figure 1 shows a high-level block diagram of the system.

The e326m supports both single- and dual-core Opteron processors. The single-core processors use higher processor clock frequencies than the dual-core processors [1]. Thus single-core processors are faster at single-threaded tasks such as executing device drivers, whereas dual-core processors have greater throughput, as can be seen with highly parallel numerical calculations.

1. A “U” is a unit of length equivalent to 1.75 inches.

The e326m supports two single-core processor speeds – 2.6 GHz and 2.8 GHz. Four speeds of dual-core processors are also supported, namely 1.8 GHz, 2.0 GHz, 2.2 GHz and 2.4 GHz.

With four channels to memory the e326m supports up to eight DIMMs, two per channel. Memory must be attached to the master CPU, but attaching memory to the secondary processor is optional. However, the best performance is obtained when all DIMM slots are populated with identical memory. The reason is that all channels are needed for best performance, and because memory addressing can be done more quickly when each processor has four ranks of memory. The e326m supports DDR400 memory in 512MB, 1GB and 2GB sizes.

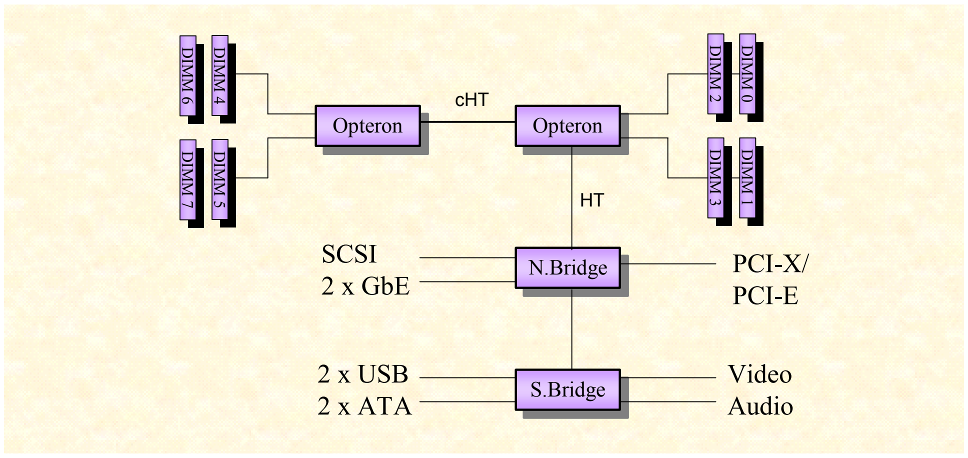

Figure 1. Block Diagram of the e326m

2. Memory Latency

2.1 Local and Remote Latency

The e326m, like all multiprocessor Opteron designs, is a shared-address, Non-Uniform Memory Access (NUMA) design. The address space spans all memory within the system, so any processor can store or retrieve data anywhere in memory, but memory that is directly attached to the processor can be retrieved more rapidly than memory attached to the other processor. When a reference misses both L1 and L2 cache, the memory subsystem must ask the other processor whether it has the data in its cache. This is called a *snoop request*. The processor must respond with the data, or with a message stating that it does not have the desired data. This is called a *snoop response*. A reference to local memory requires a snoop request and response before data gathered from local memory can be used. A reference to remote memory (memory attached to the other processor) also requires a snoop request and response, but suffers from additional latency when the data is transmitted over the HyperTransport link.

Measurements of local and remote latency are shown in Figure 2. The figure clearly shows memory latency when the program data being used (also known as the *working set*) fits into L1 cache, L2 cache, or main memory. On the 2.2 GHz dual-core Opteron processor, L1 cache latency is just under 1.4 nanoseconds (ns). L2 cache latency is about 6.7 ns. Latency to local memory is just slightly over 61 ns. In our experiment we used DDR400 CL3 memory,[1] and the time to fetch data directly from the DIMMs can be calculated to be 35 ns.[2] The remaining 26 ns is used to recognize that data is not in cache, translate physical addresses into memory requests, and to handle snoop requests and responses, which seems quite efficient.

The experiments in Figure 2 show several interesting things. First, local latencies are the same whether a single core is operating with all other cores completely idle, or one core is operating per socket. The same is true of remote references. It also shows cache latencies are the same whether the data cached is local or remote.

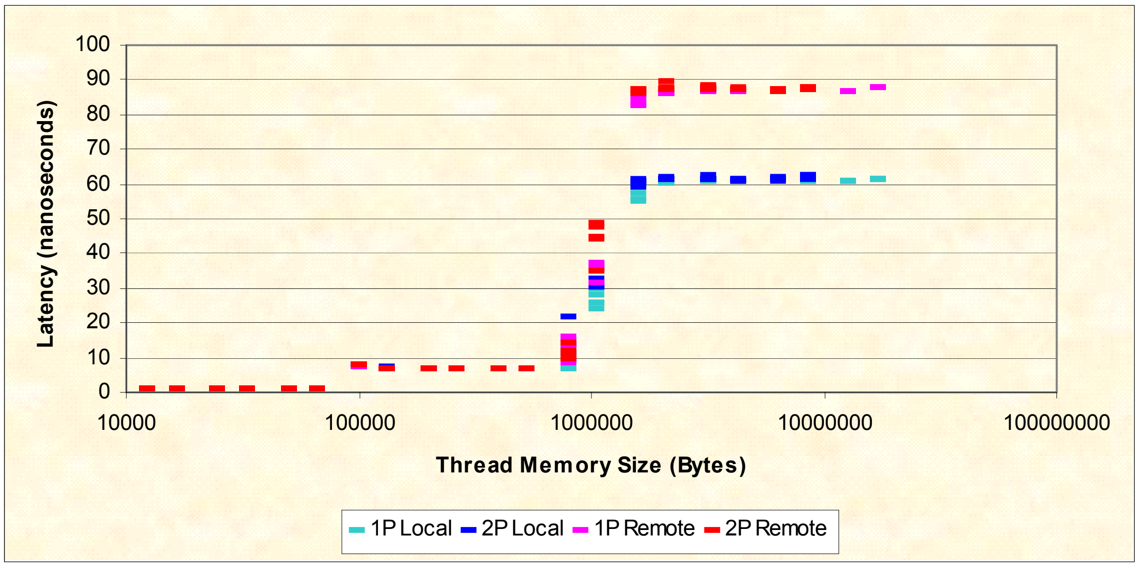

Figure 2. Local and Remote Latencies for One and Two 2.2 GHz Processors

The next chart, Figure 3, shows the memory latencies for one, two and four process threads. From this chart it is much easier to see that remote memory latency is a constant 26 ns longer than local memory latency. This is approximately the time required to transfer one cache line across an 800 MHz HyperTransport link. Furthermore, it can be seen that adding a second active thread to a dual-core processor impacts the memory latency only slightly, about 5 to 6 nanoseconds.

1. DDR400 memory accepts commands and addresses at 200 MHz. It sends and receives data at 400 MHz. CAS is an abbreviation for *Column Address Select*, and CAS latency (CL) is the minimum time, in command clocks, to receive the first data once the CAS command has been received. A CAS latency value of 3 is typical of DDR400 memory available today.
2. Time to fetch the first data from memory can be computed as the command cycle time (5 ns) times the CAS latency (3 command cycles), or 15 ns. A 64-byte cache line is fetched at 400 MHz in blocks of eight bytes, requiring 20 ns, hence 35 ns total to fetch data from the DIMM.

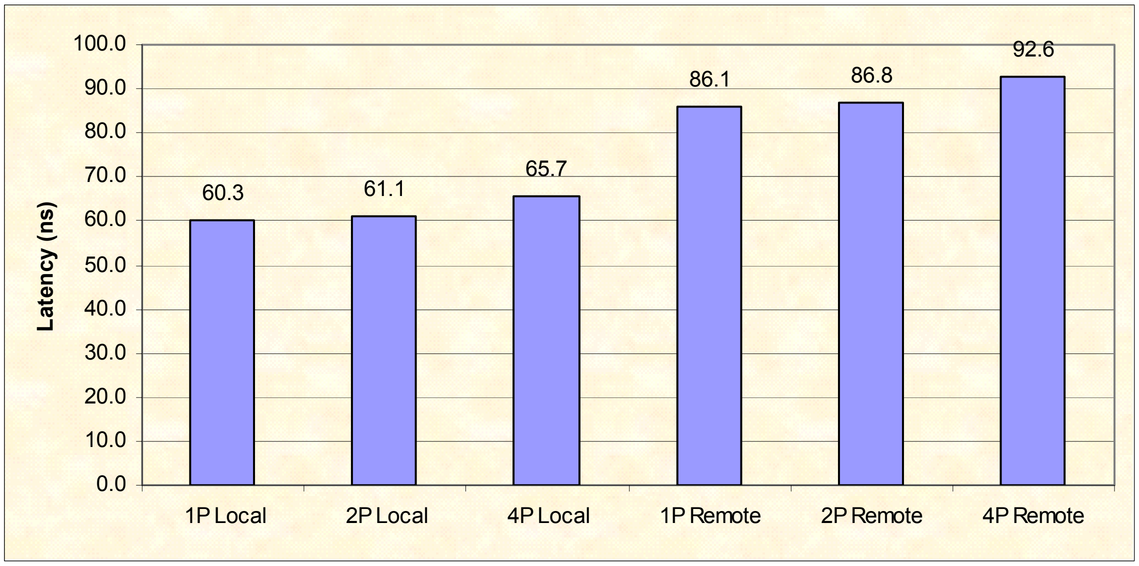

Figure 3. Local and Remote Memory Latencies for a 2.2 GHz Dual-Core System

2.2 Latency and Processor Frequency

As discussed in a previous section and elsewhere [2], Opteron processors use an integrated memory controller. This feature is the key to understanding memory performance of Opteron processor-based systems, and one principal effect of this feature is that memory performance is dependent on the clock frequency of the processor. The integrated memory controller is driven by the same clock as the processor core, so as the processor clock frequency increases, the memory controller clock frequency increases along with it. This effect can be seen clearly in Figure 4.

Two other effects can also be seen from this figure. The first effect is that dual-core latency is generally better than single-core latency, when compared by frequency. This is an interesting result. It says that a 2.4 GHz dual-core processor has better memory latency than a 2.4 GHz single-core processor. It suggests that AMD has enhanced the performance of the memory controller for dual-core processors in order to address, at least partially, the additional memory load the second core brings.

The second effect is that there is an exception to the first effect – for some unknown reason, the 2.2 GHz single-core processor has significantly better performance than the 2.2 GHz dual-core processor. The data has been checked repeatedly and it is measured and recorded correctly. The processor stepping levels were checked, as were the processors themselves. Memory, BIOS and software configurations were also checked and no differences were found. It may be an anomaly of the particular server the test was run on rather than a statement about 2.2 GHz processors in general. This question is currently unresolved.

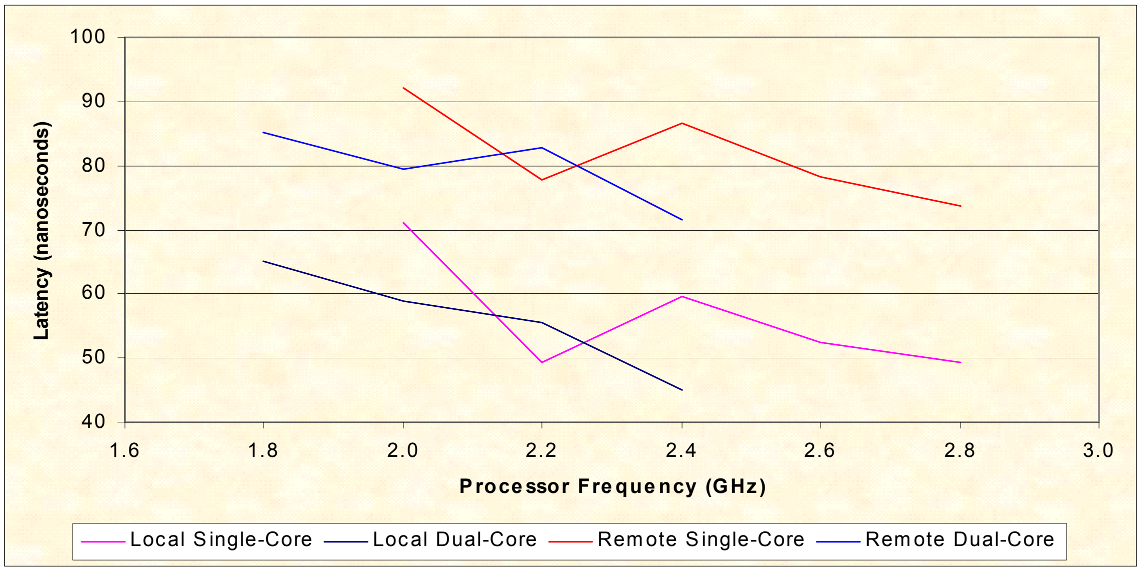

Figure 4. Local Memory Latency by Processor Frequency

2.3 Loaded and Unloaded Latency

Servers are designed to perform well under heavy workloads. Applications may have most of their data in cache or registers, in which case the application is considered to be *processor-core-bound*. In contrast, an application may have such a large working set that much of the data to be referenced does not fall within cache and must be fetched from main memory. The experiments presented so far consider only a single memory reference for each thread occurring at any given moment. When a single thread repeatedly generates a single reference to memory within a system, the latency of such a reference is often described as *unloaded latency*, because the memory subsystem is not being taxed under a typical full server load. Each memory reference is delayed only by the minimum latency of each subsystem, and not by other memory references that may be competing for those subsystems.

Under normal use there may be many threads operating, and each thread may be generating multiple concurrent memory read and write operations. Under these conditions a memory reference may be delayed, not only by the hardware, but also by competition from other memory references. Latency measurements under these circumstances is described as *loaded latency*, because the server is operating under a full load.

Before we examine the effect of memory load on processor latency, we first examine the conditions that can increase memory load. The key to understanding memory load is true processor concurrency. Consider a typical time-sharing operating system such as Linux®. A Linux server may run multiple applications at once. Each application executing within the system consists of one or more threads. When a processor running under Linux finds itself idle for any reason, it selects a thread that is ready to run, and executes its program for a period of time. For that period, the processor executes that thread exclusively. When the time period expires, or the thread gives up the processor voluntarily, the operating system selects another thread and the cycle repeats. In this way the operating system creates the illusion that it is running all available threads simultaneously.

The important point to recognize is that it is, in fact, an illusion. The operating system does not actually run all threads concurrently, but serially in very small increments. At most one thread can execute concurrently for each processor core within the system. Ten threads executing in a two-core system create no greater memory load than two of the same threads alone.

A second point to recognize is that a single thread can create multiple concurrent references to memory. Typically, references to scalar variables are loaded into cache and remain there, so they don't contribute significantly to the memory load. On the other hand, vector variables, such as arrays of floating-point values, may easily be accessed in ways that step outside of cache and must go to main memory. Memory loads are often high when many arrays are used together, or when values in many parts of an array must be used together in a calculation.

In our experiments we use a pointer-chasing benchmark. A chain of pointers is set up to fill any desired size of memory. Only one pointer is used per cache line, and the chain is followed repeatedly until the elapsed time is significantly longer than the clock resolution, guaranteeing that an accurate measurement has been made. One thread is executed per processor core. The load is indicated by the number of pointer chains being chased concurrently by each thread. The results are shown in Figure 5.

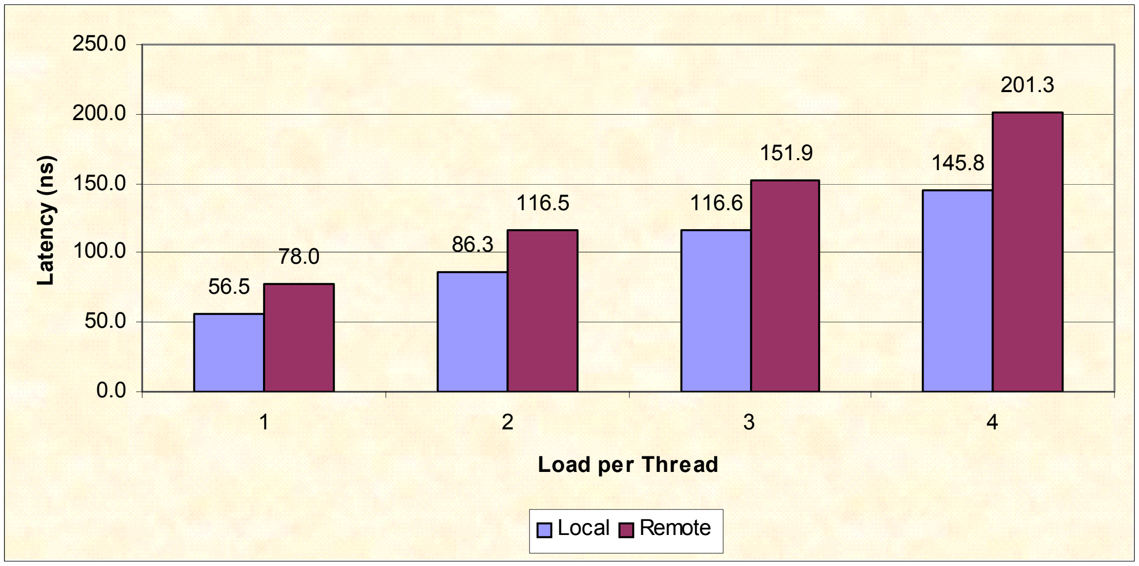

Figure 5. 2.2 GHz Dual-Processor Loaded Memory Latency

This figure reflects a two processor, dual-core system with one thread running on each of the four processor cores. This translates to four threads, each with one, two, three or four outstanding concurrent memory read operations. Stated differently, this reflects four, eight, twelve and sixteen total references. The chart shows that as more local references are added, the average latency grows by about 30 ns. In other words, as each processor adds two concurrent memory references, one per core (or thread), the average latency of all local references increases by 30 ns.

Remote references are more variable, but the behavior appears to be similar. As remote references are added, the average latency for all references grows by about 40 ns.

3. Memory Throughput

Detailed studies of e326 memory bandwidth have been reported in earlier papers [2][3], but changes do occur as processor, memory and system technologies mature. To verify memory throughput performance we ran the familiar Stream benchmark and recorded Triad throughput.

As one would expect, and as has been reported in other papers, bandwidth is best when all memory channels are put to use, meaning threads are run on each processor. In previous reports we described a noticeable difference in performance, about 10%, between single-core and dual-core processor systems. That difference has diminished greatly, although a small advantage still remains. This is shown in Figure 6.

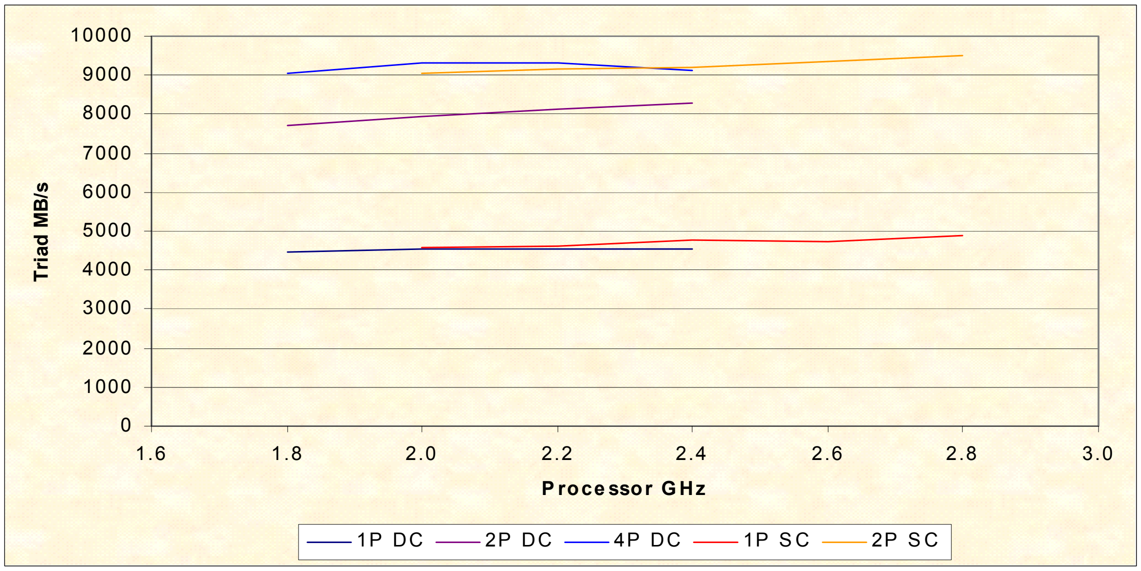

Figure 6. Stream TRIAD Results by Processor Frequency

Figure 6 also shows how performance increases with processor clock frequency, due to the integrated memory controller. In the previous system the top-speed bin dual-core processor outperformed the top-speed bin single-core processor, even though the latter had a higher clock frequency. It appears that is no longer the case. The best memory performance is achieved by the top-speed single-core processor. Furthermore, memory performance does **not** improve when only one core of a dual-core processor is used. Instead, performance drops by a little over 10%.

4. CPU Performance

Processor performance is another area where few changes beyond incremental improvements in technology are expected. The 2.8 GHz single-core and 2.4 GHz dual-core processors seem to perform with no surprises.

The Linpack benchmark measures 64-bit floating-point vector performance, with long, easily exploited vectors that fit within available cache [4]. This means memory performance is not a factor. It also means Linpack performance is scalable with the processor clock frequency and the number of vector units in the system. The 2.4 GHz dual-core processor offers the best performance at just under 16 GF/s per system, as Figure 7 shows.

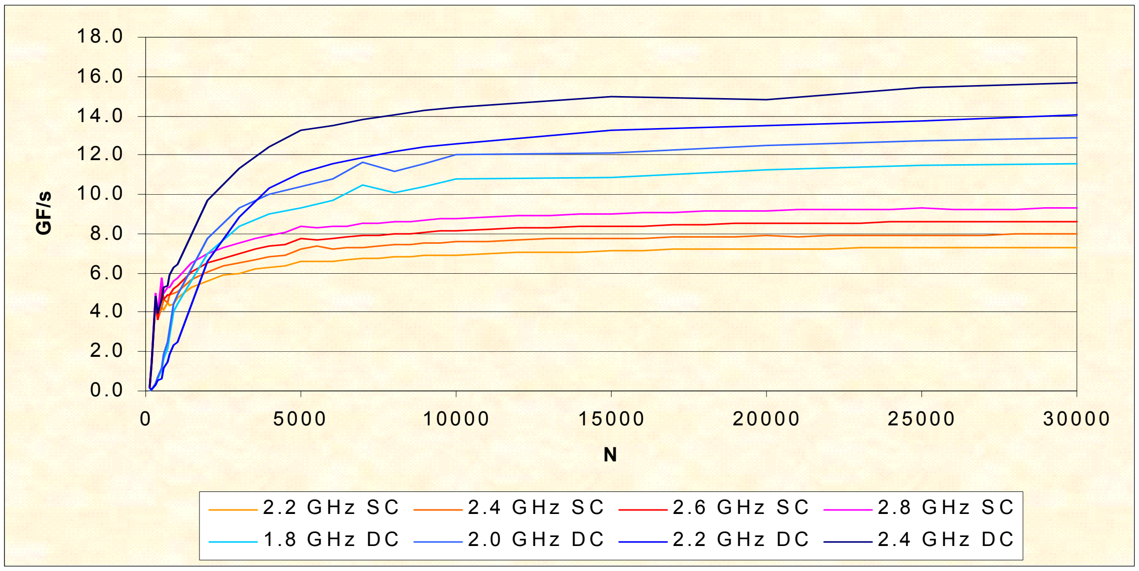

Figure 7. Linpack Performance

5. SPEC CPU2000 Performance

The SPEC CPU2000 benchmark is actually eight benchmarks packaged together [5]. It consists of three independent dimensions that are used together to identify each benchmark. Those dimensions are operation type (integer or floating-point), execution mode (speed or rate), and compilation mode (base or peak). Each run of the benchmark uses a selection of applications taken from the computer industry that reflect the intention of that benchmark. Each application is run three times and given a score based on the median run time of those three runs. All of the application scores are then combined to form a geometric mean, which becomes the benchmark score. Eight different benchmark scores are possible, namely integer base speed, integer base rate, integer peak speed, integer peak rate, floating-point base speed, and so on.

The integer benchmarks use 14 integer applications, and go by the name of CINT2000. Most of the applications are highly cacheable, and overall, memory performance has little effect on benchmark performance. The floating-point benchmarks use 12 floating-point applications that generally have strong dependence on memory performance and floating-point performance, although large caches can also have impact on the outcome. The floating-point benchmarks are called CFP2000.

SPEC CPU2000 can be executed as either a speed benchmark or a rate benchmark. The speed benchmarks execute a single copy of each application serially, using a single processor core, and report the results. These results approximate the speed with which a single task can be completed. Rate benchmarks execute multiple copies of each benchmark concurrently, usually one copy per processor core. This approximates the system throughput under full load. Since servers are designed to complete many tasks at once, sometimes sacrificing the speed of individual tasks, the rate benchmarks are generally a more relevant measure of server performance.

The compilation mode is often the most misunderstood aspect of the SPEC CPU2000 benchmarks. The base and peak benchmarks differ only in what compiler optimization flags may be used to compile the applications. Proper execution of the base benchmarks require that no more than four optimization flags be used, and that all applications use the same optimization flags. Peak results may use any number of flags and each application may be different. The base results approximate an environment such as code development, where a developer wants some combination of flags that works reasonably well, but is unwilling to fine-tune the compilation of the application. In my opinion, base results do not reflect the performance of the system very well. Rather, they reflect how well the compiler has packaged its optimization flags, and how well it has implemented its general purpose optimization flags such as -O3.

Peak results are unconstrained in their number and choice of optimization flags, so the compiler is free to take advantage of architectural features of the server. As a result, peak results more accurately reflect the performance of the system. In this report only peak results are used.

5.1 Speed Results

As mentioned before, the CINT2000 benchmarks are highly cacheable. As a result, the speed results are almost perfectly scalable with processor frequency. This is shown in Figure 8. This high degree of cacheability makes the load on the memory subsystem light. It also means that there is very little difference between the performance of a single-core processor and a single core of a dual-core processor of the same frequency. This benchmark is processor-core-performance-constrained; that is, other components of the system are not oversubscribed and therefore do not limit benchmark performance.

The CFP2000 speed results are also nearly perfectly scalable with processor frequency, although those applications are not cacheable nearly to the extent that CINT2000 is. They scale well, nonetheless, because the memory subsystem is not oversubscribed for the processor frequency range that is measured here. As the processor frequency climbs, the memory performance scales more slowly than the processor performance does. In our experience with other processors, when the memory subsystem becomes a bottleneck, the performance of this benchmark flattens out. No such flattening occurs, as Figure 9 shows, because the memory subsystem is adequate for the load being generated.

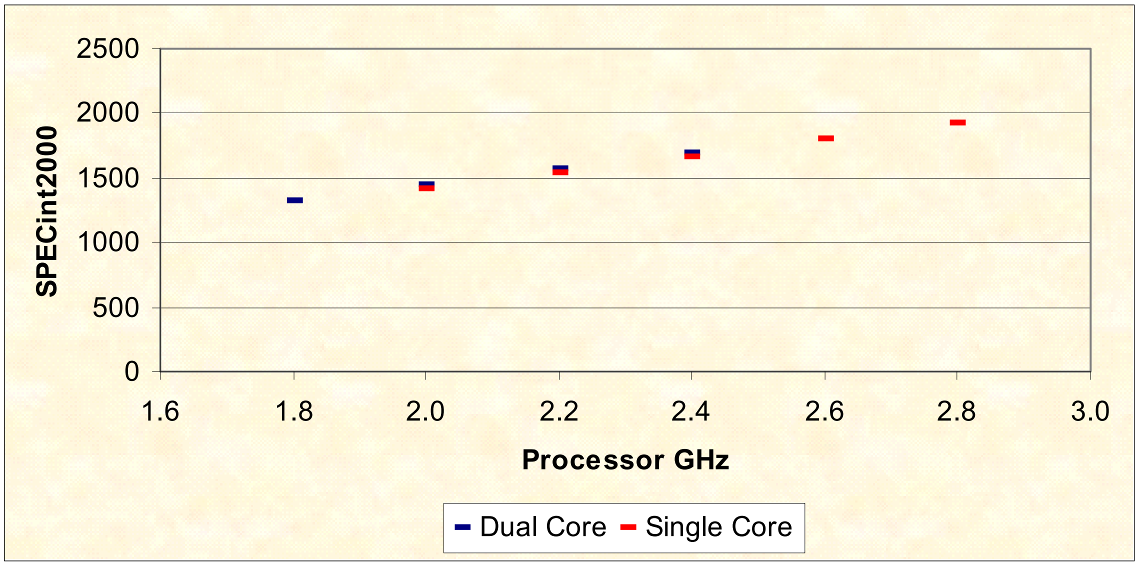

Figure 8. SPEC CINT2000 Performance by Processor Frequency

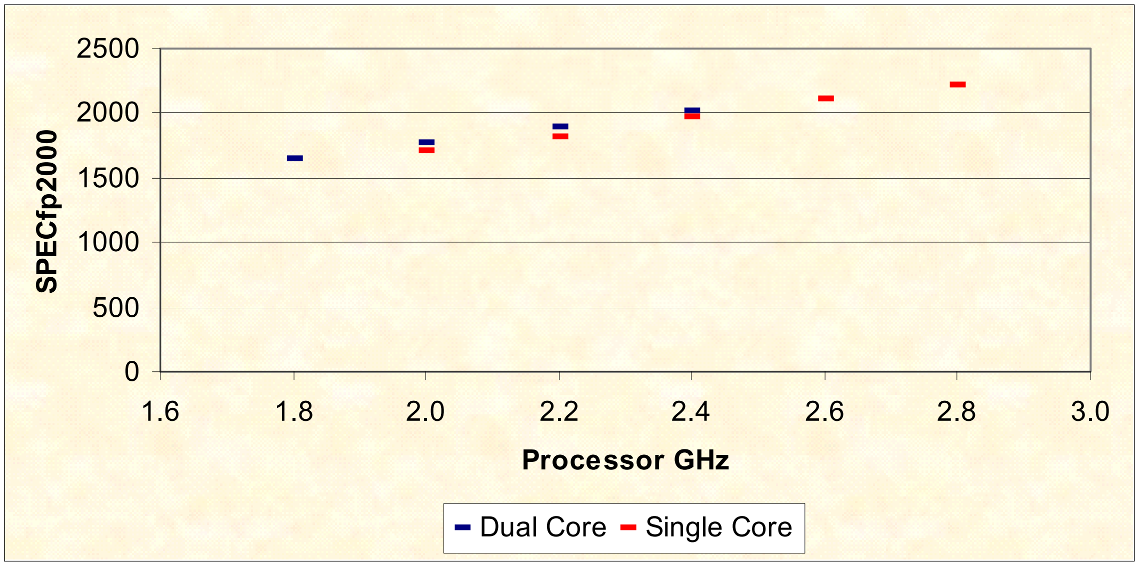

Figure 9. SPEC CFP2000 Performance by Processor Frequency

There is something interesting to be learned from comparing the single-core performance of single- and dual-core processors. Comparing processors by frequency (e.g., 2.2 GHz single-core against 2.2 GHz dual-core), we see that dual-core processors are slightly faster. This comparison is shown in Figure 10. It is interesting to note that speed results are only slightly faster, around 2% to 4%. Since memory throughput was about the same between processors, it seems unlikely that is the reason for the difference here. Memory latency seems a more likely cause, but we don't know for sure.

The more obvious comparison is by speed bin rather than by frequency. The fastest single-core processor clearly outperforms the fastest dual-core processor, by 12% in this case, because of its higher clock speed. Single-core processors should always excel at speed tests.

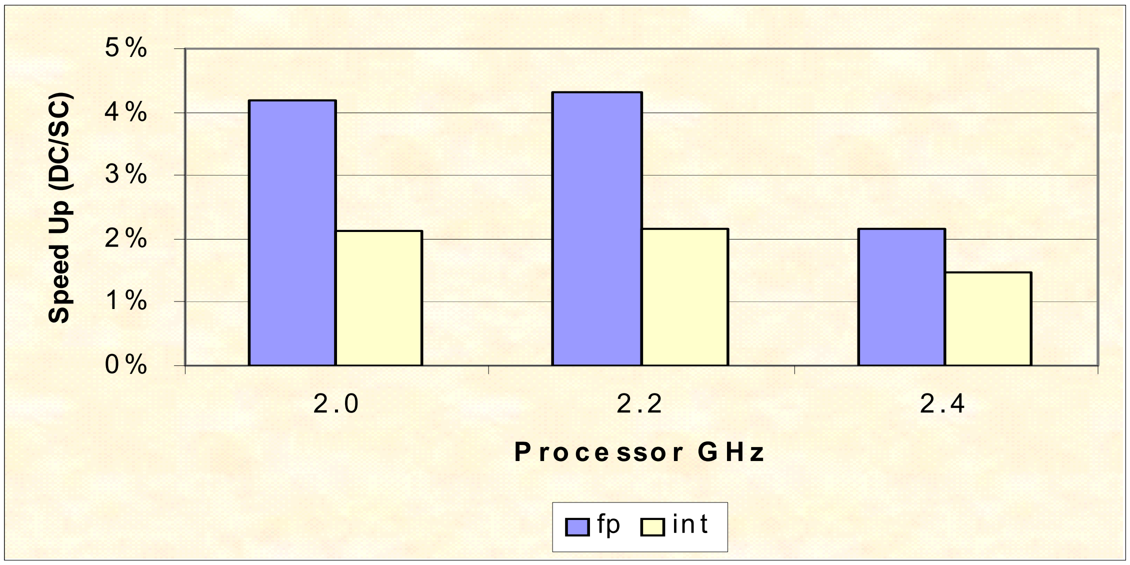

Figure 10. Speed Comparison of Single- and Dual-Core Processors by Frequency

5.2 Rate Results

SPEC CPU2000 rate results are generally more relevant measures of server performance, because this more closely approximates the environment in which servers are used. The rate benchmarks measure system throughput for integer and floating-point workloads. Since dual-core processors are designed for greater throughput, even at the expense of individual tasks, one would reasonably expect better performance from dual-core processors. If there are any surprises, it is how well dual-core processors do their job.

Figure 11 shows the CINT2000 rate benchmark. Scaling by processor frequency is almost perfectly linear because of its high degree of cacheability. CFP2000 also shows a high degree of scalability (Figure 12) even though it is much less cacheable, because the memory subsystem scales with the number of processors in the system.

Figure 13 shows a difference in performance when the systems are compared by processor frequency. It can be seen quite clearly from the chart that dual-core processors get nearly perfect scores. Out of a possible 100% speed-up, they achieve an outstanding 94% on the CINT2000 rate benchmark. Dual-core processors also do very well on CFP2000 rates, although the results are not as dramatic as CINT2000. On CFP2000 they achieve 60% speed-up, which is very respectable. This performance scaling appears to be limited by memory performance.

Figure 14 shows a comparison of the top-speed processors of each type. This comparison is probably more useful than a comparison by frequency. It allows people to see which processor better matches their requirements based on what is available today. Based on the processor speeds available today, the greatest speed-up one would expect is about 71%. The 2.4 GHz dual-core system achieved about 68% speed-up over the 2.8 GHz single-core system on CINT2000 rates, which shows the dual-core system is very good at this type of workload. Comparing the systems with CFP2000 rates showed a 42% speed-up, which is still good.

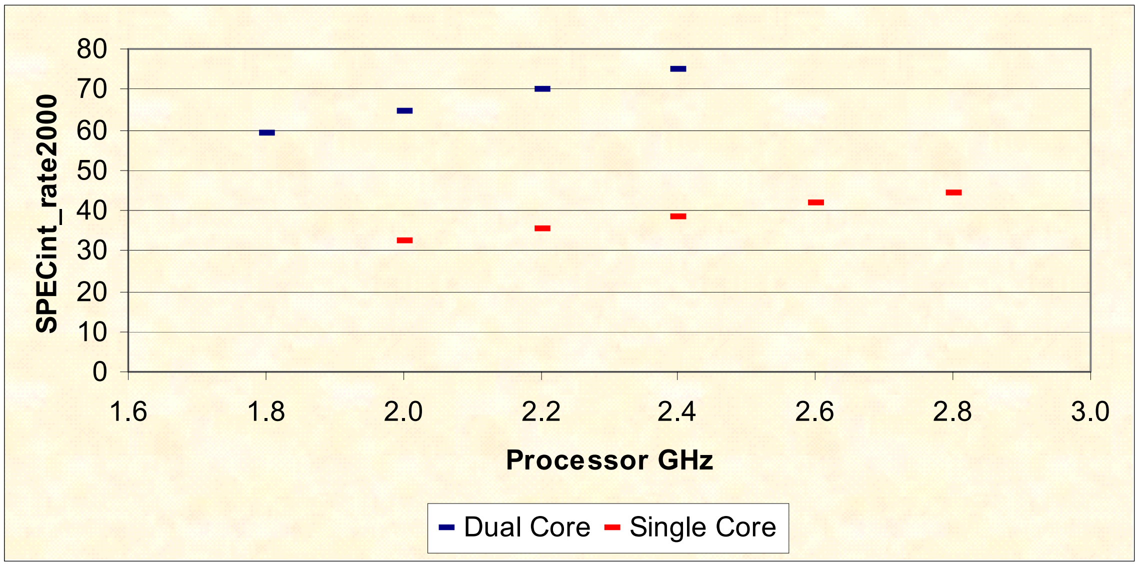

Figure 11. SPEC CINT2000 Rate Performance by Processor Frequency

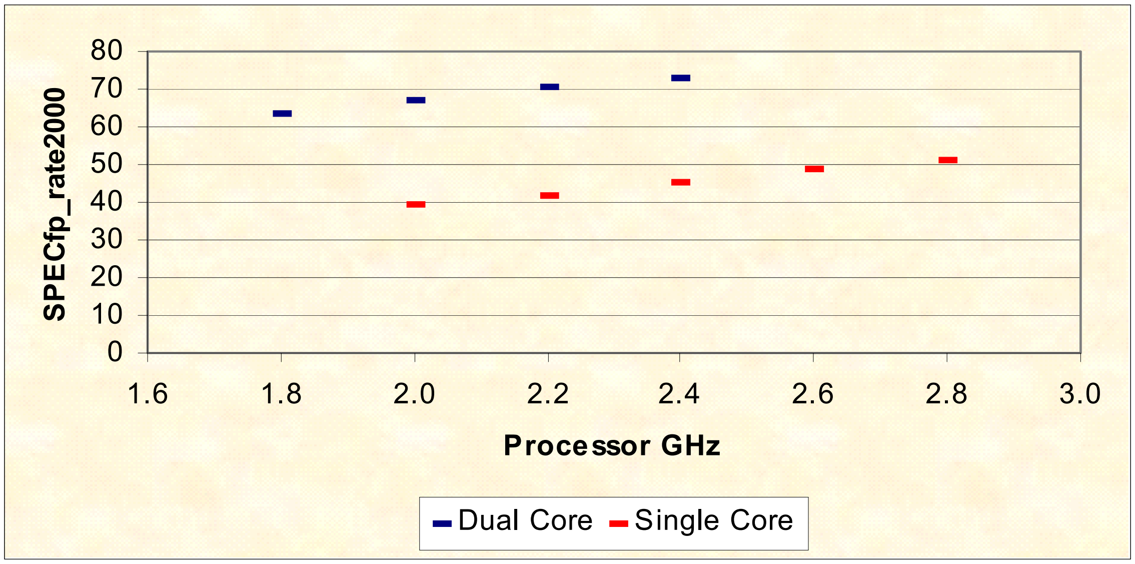

Figure 12. SPEC CFP2000 Rate Performance by Processor Frequency

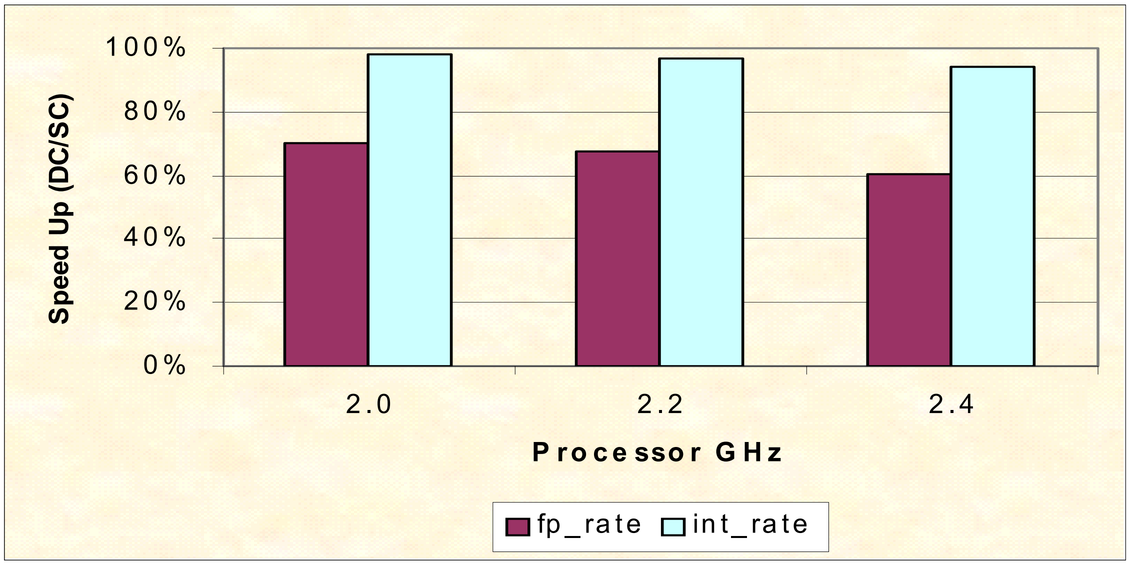

Figure 13. Throughput Comparison of Single- and Dual-Core Processors by Frequency

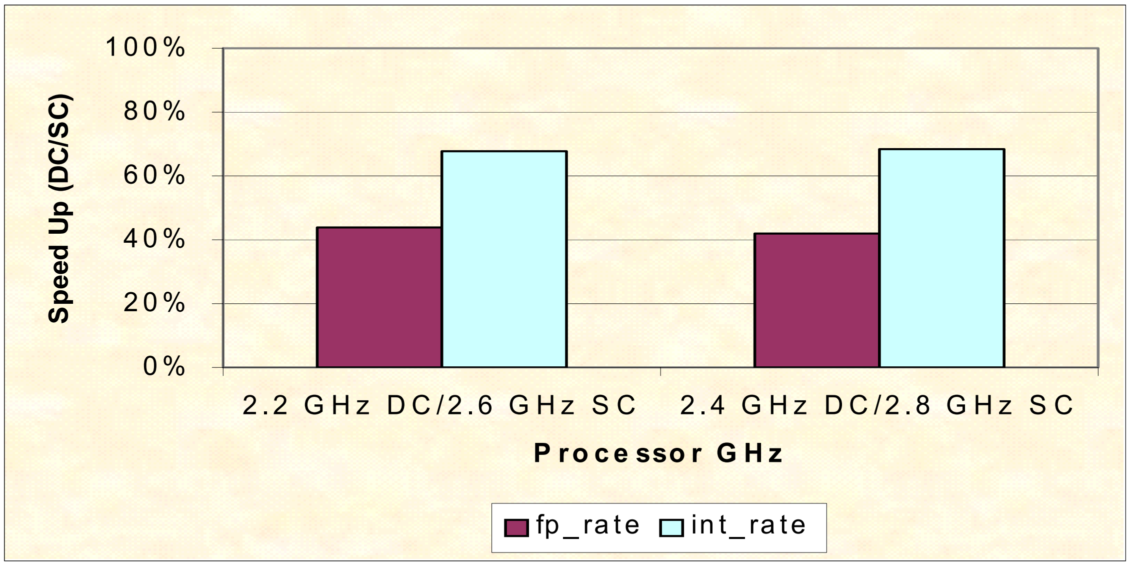

Figure 14. Throughput Comparison of Top-Speed Bin Single- and Dual-Core Processors

6. PCI-Express Performance

The e326m adds a new PCI-Express (PCI-E) slot to the system. This slot may be used for any type of I/O, but it is most likely to be used for high-performance communication adapters such as InfiniBand. Because we have no specialized hardware we can use to measure PCI-E performance directly, we chose, instead, to measure it indirectly through the performance of an adapter in the slot. We used Voltaire HCA 400Ex InfiniBand adapters. This adapter performs well, and we have measurements we can use for comparison using this adapter on a 3.6 GHz IBM eServer xSeries® 336 (x336) server. Tests on the e326m were performed using 2.2 GHz dual-core Opteron processors.

The two tests we chose were unidirectional send/receive throughput, and bidirectional send/receive throughput. In both cases we used the same adapters, operating system kernel, drivers, and MPI stack, on both the e326m and on the x336. The performance of the unidirectional tests are shown in Figure 15. The chart shows nearly identical performance between the two systems, indicating an adequacy for this type of load.

The bidirectional performance, however, is not as good. Figure 16 shows that divergence of the two systems becomes significant at around 8K bytes. Transmission of large messages ultimately shows as much as 25% lower performance on the e326m than on the x336. The e326m has a much slower clock speed, but its raw integer performance is lower by only 9%, so it isn't clear why the performance should be so much slower on the e326m.

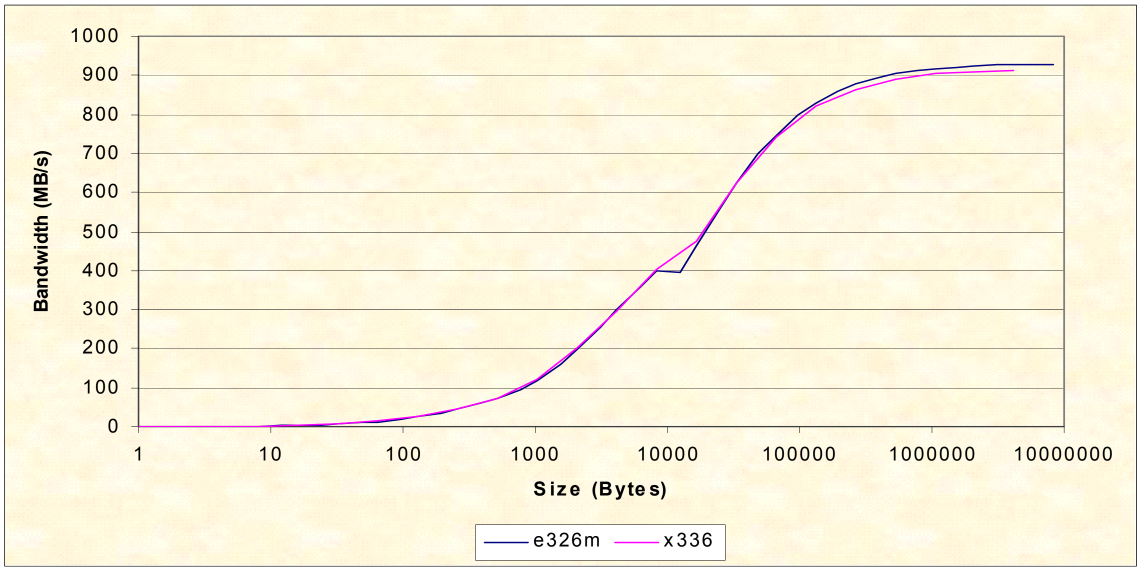

Figure 15. Unidirectional (PingPong) Performance of 4x InfiniBand over PCI-Express

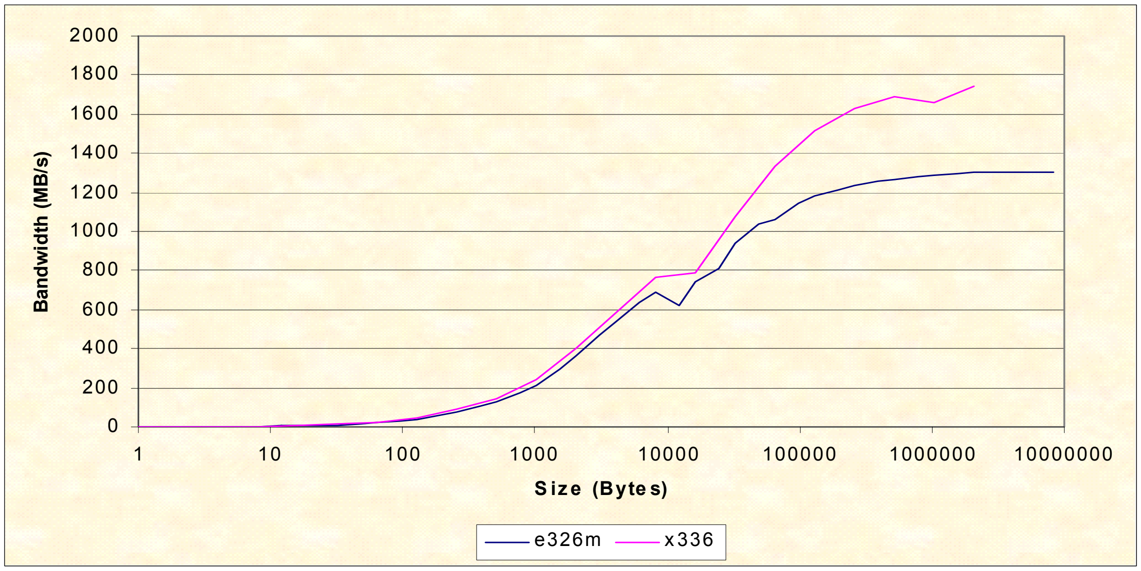

Figure 16. Bidirectional (Sendrecv) Performance of 4x InfiniBand over PCI-Express

7. Conclusions

The e326m is a high-performance, rack optimized, Opteron processor-based server that supports both PCI-X and PCI-E adapters. It also supports both single-core and dual-core processors in a NUMA architecture. Unloaded local memory latencies range from a little under 50 ns to a little below 75 ns, depending on the choice of processor, with dual-core processors having better latencies than single-core processors, and higher-frequency processors having better latencies than lower-frequency processors. Unloaded remote memory latencies add another 26 ns to the local latency, which is precisely the time required to transmit a cache line over an 800 MHz HyperTransport link. Loaded memory latency was tested on a dual-processor system with 2.2 GHz dual-core Opteron processors, running one thread on each processor core. Adding a memory load to each of the threads added 30 ns to the average local memory latency and 40 ns to the remote memory latency.

Memory throughput is essentially as we reported in earlier papers, with incremental improvement due to higher frequencies and maturing technology. Best performance is obtained by populating all DIMM slots with identical memory. Dual-core processors give slightly better throughput than single-core processors of the same frequency, but the fastest single-core processors give better throughput than the fastest dual-core processors.

Floating-point performance is faster with the faster processors in proportion to the faster clock speeds. The 2.4 GHz dual-core processors are now able to achieve almost 16 GF/s on the Linpack benchmark.

SPEC CPU2000 performance continues to scale well with CPU frequency. CINT2000 speed and rate benchmarks both scale well because the benchmark is highly cacheable and measures primarily processor performance. CFP2000 speed and rate benchmarks also scale well, but do so because the memory subsystem scales with the number of processors. The server has sufficient memory bandwidth to support the load imposed by both benchmarks. Both CINT2000 and CFP2000 speed benchmarks perform better on single-core processors, because of their higher clock frequencies. Both rate benchmarks perform better on dual-core processors. Dual-core processors obtain nearly perfect scaling with the number of cores on the CINT2000 rate benchmark, achieving 94% of the second core. Dual-core processors achieve 60% of the second core on the CFP2000 rate benchmark, which is still quite respectable.

PCI-Express performance was good relative to the x336 in most cases, but performance was down in some cases. Unidirectional performance was almost identical to the x336 for all message sizes, but bidirectional performance was up to 25% lower on very large messages. Small message sizes, messages below 8KB, had nearly identical performance.

8. References

[1] Douglas M. Pase and Matthew Eckl, "A Comparison of Single-Core and Dual-Core Opteron Processor Performance for HPC," ftp://ftp.software.ibm.com/eserver/benchmarks/wp_Dual_Core_072505.pdf, IBM, July 2005.
[2] Douglas M. Pase and James Stephens, “Performance of Two-Way Opteron and Xeon Processor-Based Servers for Scientific and Technical Applications,” ftp://ftp.software.ibm.com/eserver/benchmarks/wp_server_performance_030705.pdf, IBM, March 2005. Also published under 2005 LCI Conference, http://www.linuxclustersinstitute.org/Linux-HPC-Revolution/Archive/PDF05/14-Pase_D.pdf.
[3] Douglas M. Pase, "Memory Performance of the IBM eServer 326 for HPC Workloads," http://perform.raleigh.ibm.com:8090, IBM, May 2005.
[4] Douglas M. Pase, "Linpack HPL Performance on IBM eServer 326 and xSeries 336 Servers," ftp://ftp.software.ibm.com/eserver/benchmarks/wp_Linpack_072905.pdf, IBM, July 2005.
[5] SPEC CPU2000, http://www.spec.org/cpu2000/.