August 2006
Performance Analysis

Performance of the IBM System x™ 3755

Douglas M. Pase and Matthew A. Eckl
IBM Systems and Technology Group

Abstract

In this paper we examine the performance of the IBM System x™ 3755 server, or x3755. Our analysis includes memory bandwidth and latency, floating-point vector performance and performance of the SPEC® CPU2000 speed and rate benchmarks. In this paper we find the innovations of the x3755 to be quite effective. The processor pass-through module allows the server to be used as a three-processor system, with better performance, in some cases, than a four-processor configuration. Through innovative engineering the x3755 also supports 667 MHz memory even when all 32 DIMMs are used.

1. Introduction

The IBM System x™ 3755, or x3755, is a new rack-optimized server based on the AMD Opteron dual-core processor. It supports four processor sockets and 32 memory DIMM slots. Supported speeds include 2.0 GHz, 2.2 GHz, 2.6 GHz and 2.8 GHz processors. Memory is 667 MHz DDR2, in sizes ranging from 512MB to 4GB[1] per DIMM. This gives a total capacity of up to 128GB of main system memory. The x3755 supports a variety of I/O slots, including one HTX slot, one PCI-Express (PCI-E) x16 slot, two PCI-E x8 slots, one PCI-E x4 slot, and two PCI-X slots. The two PCI-X slots may be used as a pair of 100 MHz slots, or only one may be used as a 133 MHz slot. The system also supports four hot-swap SAS disk drives. The server is housed in a 4U chassis.[2]

The server generally follows a standard four-socket Opteron processor design,[3] but with improvements in key areas that will be mentioned throughout this paper. The four processors are connected together by HyperTransport™ links in a ring topology. Each processor may have up to eight DIMMs on two memory channels. One key improvement is that all 32 DIMMs operate at 667 MHz. AMD system development guidelines specify that 533 MHz memory be used for reasons of signal timing when connecting more than two DIMMs per channel. IBM developers have discovered a way to maintain timing and electrical integrity while using four DIMMs of the faster memory. This innovation allows the x3755 to use high-speed memory even in very large memory configurations (configurations above 64GB), and less expensive memory in configurations up to 64GB without significant loss of performance.

Another key innovation is the pass-through module. Figure 1(A) shows an x3755 configured with four processors. Each pair of processors in opposite corners of the system is separated by two hops; that is, in order to communicate, every message must traverse two coherent HyperTransport (cHT) links. If the processor in the upper-left position were removed, processors in the lower-left and upper-right positions would form a line rather than a ring. More importantly they would still be separated by two hops. As shown in Figure 1(B), IBM provides a pass-through module that can be inserted into the system in place of the missing processor to change the system topology from a

1. The x3755 will support the 8GB (2x4GB) DIMM option when it becomes available.
2. A “U” is a unit of length equivalent to the spacing of slots in a standard equipment rack, or 1.75 inches.
3. Throughout this paper a distinction will be made between processors and processor cores. A processor core is that portion of a processor capable of independently executing an instruction stream. A processor is a collection of one or more cores in a package that fits into a processor socket. A dual-core processor is a processor that contains two processor cores.

line back to a ring. It will be shown in a later section (Section 2.6) that this configuration has significant benefits to memory performance.

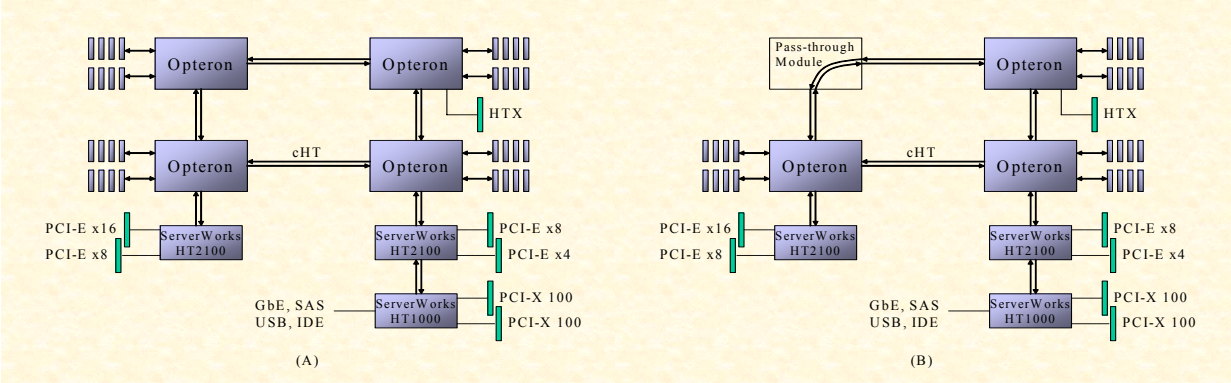

Figure 1. Block Diagram of the x3755

The I/O subsystem includes two ServerWorks HT2100 bridges. The first bridge connects the master processor to the majority of the I/O devices, including two of the PCI-E slots, the PCI-X slots, storage and legacy I/O. The second bridge connects a separate processor to two additional PCI-E slots. A third slot, the HTX slot, is connected to an additional processor without a bridge. Devices connected through the HTX slot can transfer data to and from main system memory without incurring any penalty normally associated with passing data through a bridge. This also gives a performance advantage for those devices that are able to use it.

2. Memory Performance

The x3755, 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 that any processor can store or retrieve data anywhere in memory, but data within memory that is directly attached to the processor can be retrieved more rapidly than data in memory attached to another processor. When a reference misses both L1 and L2 cache, the memory subsystem must ask all other processors whether it has a more current copy of the data in its cache. This is called a *snoop request*. Each processor must then 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 a response from every other processor 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 remote data is transmitted over the HyperTransport link.

Memory performance can be stated in terms of memory bandwidth or throughput and latency. The total bandwidth of a system is the product of the number of channels within the system and the speed of each channel. It is an indication of the upper bound of performance – a measure guaranteed never to be exceeded, but not necessarily something that can be achieved. The bandwidth of a fully configured x3755 is 42,667 MB/s. Memory throughput is a measure of what can actually be achieved. In our tests we used the Stream benchmark [1] for measuring memory

throughput. We were able to achieve 18,772 MB/s on a three-processor configuration and 17,173 MB/s on a four-processor configuration using the Stream Triad benchmark. We also measured memory latencies from just below 45 ns up to just below 100 ns depending on data locality and memory loading, using a custom benchmark called pChase.

The Stream benchmark was designed to test memory throughput. It uses four common operations from scientific and technical computing. The four operations are copy, scale, add and triad. Copy and scale request one 64-bit floating-point value for each 64-bit value that is stored. Add and triad each load two values for each value stored. All four operations request and store data sequentially over a range of memory that is much larger than the largest cache. In this way the Stream benchmark approximates some of the important operations that frequently occur in high-performance computing (HPC) applications.

The pChase benchmark is a pointer-chasing benchmark. A chain of pointers is set up to fill any desired size of memory. The pointers are set up specifically so that they do not reference memory sequentially. Only one pointer per cache line is used. The chain is followed repeatedly until the elapsed time is significantly longer than the system clock resolution, guaranteeing that an accurate measurement has been made. No more than one thread is executed per processor core. The benchmark can measure local and remote latency as well as loaded and unloaded latency. Both pChase and Stream are used to determine the effects of different conditions on memory performance.

Unless otherwise specified, all performance results in this paper were measured on an x3755 using four 2.8 GHz processors and 64GB of CL5, 667 MHz DDR2 memory in 32 DIMMs.

2.1 Node Interleave

We first measure the effect of page interleaving on memory performance. Memory is divided into multiple pages of 4096 bytes. The pages are then mapped to physical addresses by the memory controller using one of two page interleaving modes. One mode interleaves page addresses among all processors. In other words, the first page address is mapped to memory on the first processor, the second page address is mapped to memory on the second processor, and so on, in a round-robin fashion until all memory has been given an address. This is referred to as *node interleaving*. If this mode is desired, it can be set in the BIOS. It has the side effect of disabling NUMA processing in the operating system. When node interleave is enabled, the operating system must ignore any differences in the physical locality of memory and treat all memory as a flat space.

The second mode treats memory as monolithic blocks. All memory attached to a processor is given a single contiguous block of physical addresses. Low addresses are assigned to the master processor. The second processor is assigned a contiguous block of addresses beginning where the master processor's addresses end. This continues until all processors have been allocated physical addresses for their memory. This interleaving mode is desirable for NUMA processing because it allows the operating system to manage memory according to its physical locality. This mode is achieved by disabling node interleave in the BIOS.

Figure 2 shows the effect of node interleaving on memory performance. The biggest issue is that enabling node interleaving disables NUMA processing and prevents the operating system from

exploiting any locality that exists. It should not be a surprise that mixing local and remote references would be slower than using local references alone. We can see from the figure that, for these cases, enabling node interleaving increases latency by 10 to 15 nanoseconds, and throughput drops by almost half. Clearly node interleave should not be enabled for these work loads when performance is a prime consideration.

Although not shown here, node interleave can be useful when a small number of contiguous pages forms a hot spot. Node interleave can redistribute the hot spot over multiple DIMMs. Even so, node interleave should be useful only rarely because NUMA operating systems are becoming very effective at managing memory affinity.

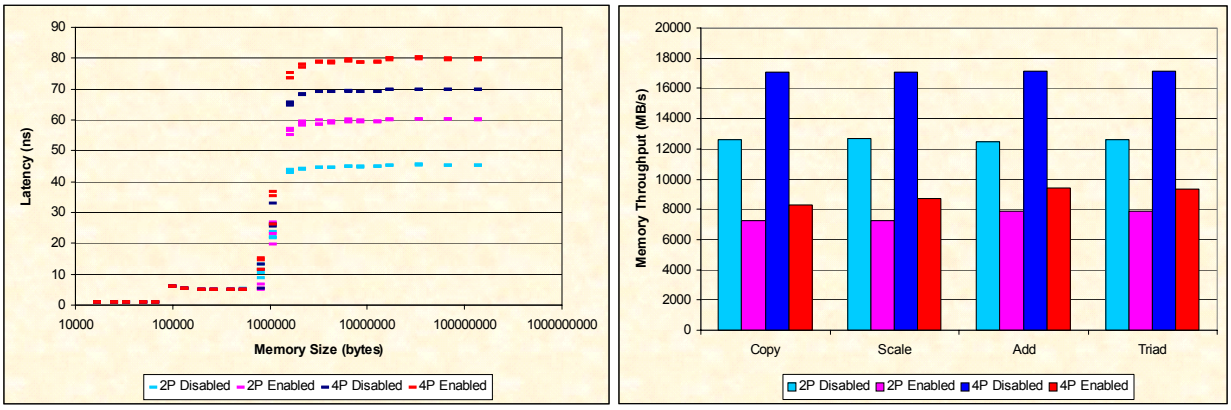

Figure 2. Effect of Node Interleave on Local Memory Performance

In addition to the latency of system memory, the latency to L1 and L2 cache can also be seen. The L1 latency is just below 1.1 ns. The L2 latency is measured at about 5.8 ns.

2.2 ChipKill

ChipKill is a feature that increases the reliability of system memory. Server memory has error correcting features called error correcting codes (or ECC) that attempt to correct when a bit within a cache line has become corrupted, and detect when two bits have changed so that the system can take proper action. ChipKill is a feature that increases the system's ability to detect errors that occur within memory.[1] An important question is whether the increased reliability afforded by ChipKill hurts performance in any way.

In Figure 3 we see that it does not hurt performance. Both latency and throughput are nearly identical, certainly to within experimental variation, between cases where ChipKill is enabled and where it is disabled. Given that this is the case, it is hard to see any motivation for turning ChipKill off.

1. For a good introductory paper on ChipKill see [2].

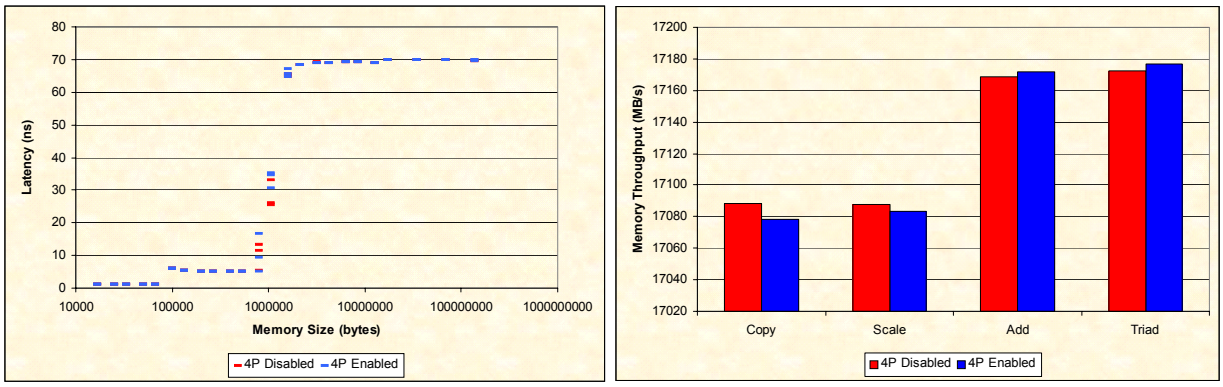

Figure 3. Effect of ChipKill on Local Memory Performance

2.3 Processor Scaling

Processor scaling is one of the more interesting aspects of performance analysis. Processor scaling shows how performance changes when processors are added to or removed from a system. Figure 4 shows the latency and throughput of local memory references for one, two, three and four processors. The three-processor configuration uses the pass-through module. Notice how memory throughput scales nearly perfectly with two and three processors, but there is a significant drop in throughput when the fourth processor is added. The scaling is near perfect for two processors because snoop traffic must only cross a single hop (HyperTransport link), and snooping takes less time than is required to retrieve the data from local memory. It is also near perfect for three processors because the pass-through module creates a fully connected interconnect where the snoop traffic again only needs to traverse a single hop. Performance drops when the fourth processor is added because the fourth processor requires some of the snoop traffic to traverse two hops. (The need for a second hop with four processors can be easily seen in Figure 1.)

A similar effect occurs for memory latency, and for similar reasons. The local memory latency is the same for one, two and three processors because when the system is fully connected, any snoop traffic can complete before the data can be retrieved from the local memory. When a fourth processor is added, a portion of the snoop traffic must traverse two hops within the system. This traffic is slower because it must travel farther (two hops instead of one) and because it must now compete with other snoop traffic for access to the HyperTransport link. The longer distance and the contention both cause the snoop traffic to take longer to complete, and it now takes longer to complete the snoop traffic than it does to retrieve data from local memory.

The effect of using (or not using) the pass-through module will be explored in a later section (Section 2.6), but it is worthwhile noting here that the high performance of the three-processor configuration is not seen when the pass-through module is not used.

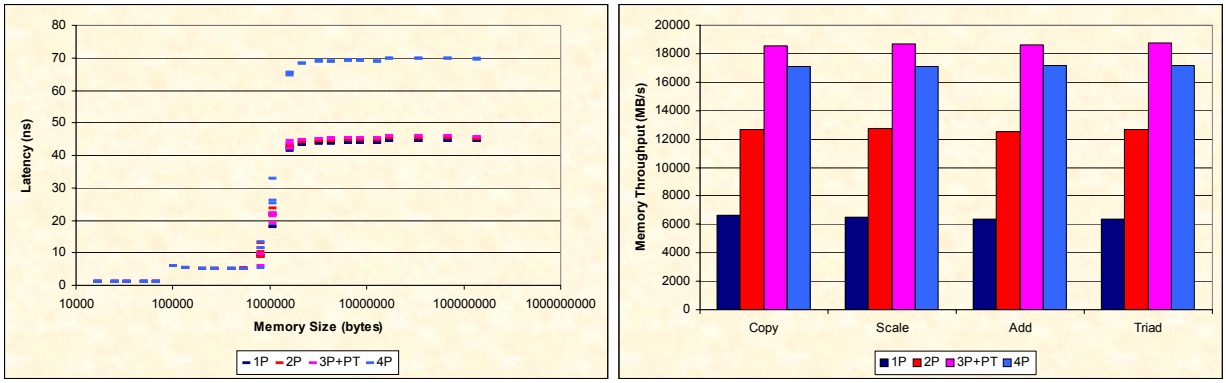

Figure 4. Effect of Processor Scaling on Local Memory Performance

2.4 Remote Latency

Every NUMA system has local memory and remote memory. In fact, the great benefit of NUMA is that while flat memory may be easier to work with, it may be possible to achieve better performance at less expense by placing memory closer to some processors than to others. While the emphasis is on exploiting locality, remote references still occur and must be addressed.

Figure 5 shows the latency of local and remote references for one, two, three and four processors. As can be seen in the left half of the figure, the local and remote latencies are consistent for one through three processors. On the right we can see the latencies are quite different for four processors. The four-processor local latency is only slightly less than the one-hop latency, but much greater than the local latency for one, two or three processors. Interestingly, the one-hop latency is the same regardless of the number of processors. The reason is that the latency is dominated by the time to retrieve and transfer the data from the remote processor. The two-hop latency is much longer (about 25 ns longer) than any other because of the extra distance that the data must travel.

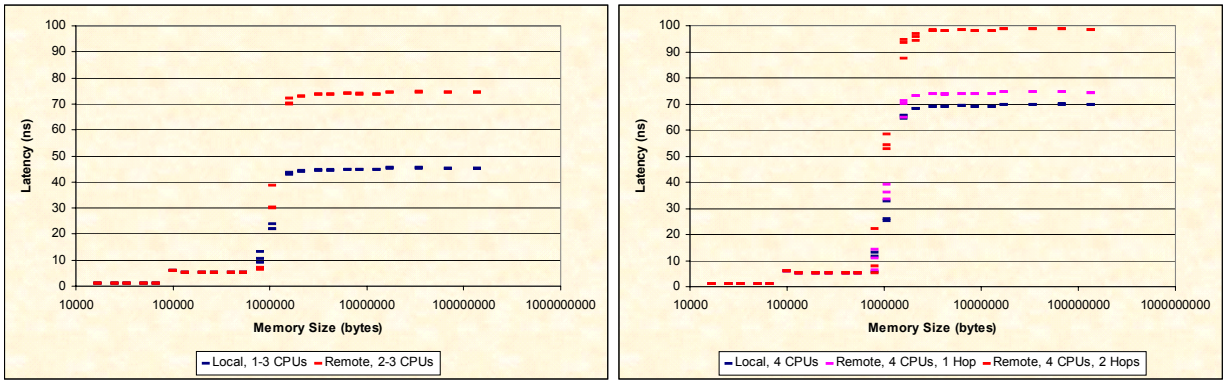

Figure 5. Effect of Remote References on Memory Latency

2.5 Memory Loading

Memory loading refers to the number of concurrent references outstanding at a given moment. Servers are designed to maximize system throughput, and that includes memory throughput. It is quite common to have multiple outstanding memory references being executed concurrently. Stream addresses this issue with the add and triad operations where each iteration requests two operands from memory for each store operation that takes place. The benchmark also uses loop unrolling to initiate the memory references from multiple iterations at once. In this way the Stream benchmark makes heavy use of memory loading.

The pChase benchmark also measures memory loading, but in a more controlled way. Unlike Stream, the number of outstanding concurrent references can be specified directly. When only a single reference is in flight, that is described as *unloaded latency.* When more than a single reference is executed concurrently, it is considered *loaded latency.* For the next set of experiments the loading varied between one and two concurrent references over multiple processors. The results are shown in Figure 6.

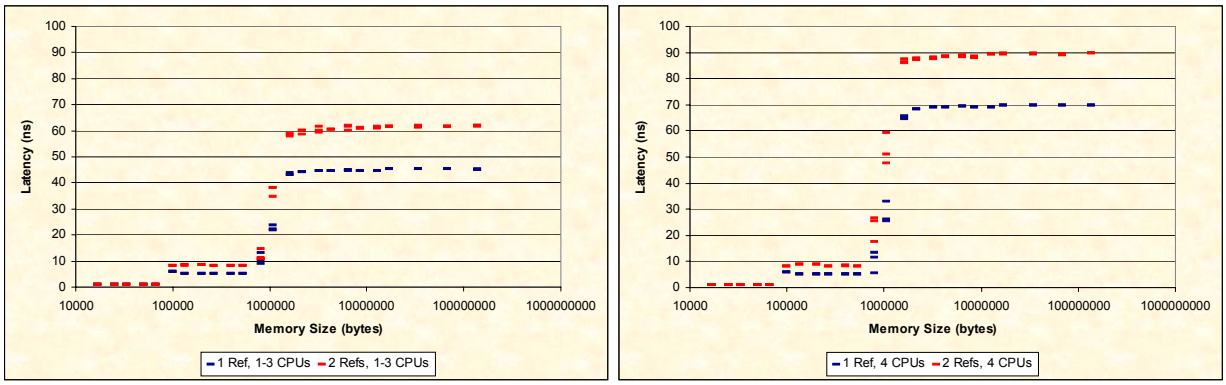

Figure 6. Effect of Loading on Local Memory Latency

As one would expect, loading clearly increases the latency of memory operations, and the effect is slightly more pronounced with four processors than when fewer processors are used. In the four processor case, the difference in latency between having one and two outstanding references is about 20 ns. In other words, the first reference costs 70 ns, the second costs an additional 20 ns. In the one to three processor cases, the difference is approximately 17 ns.

2.6 Pass-Through Modules

As mentioned before, a significant IBM innovation is the pass-through module. It turns what would otherwise be a linear processor topology into a fully connected topology, with a significant improvement in performance. Any four-socket Opteron processor-based server can have one of the processor sockets depopulated to obtain a three-way configuration. But in doing so it leaves the server configured as a line, with one processor on each end and one processor in the middle. This configuration requires that snoop requests and responses originating from one of the two end processors, and certain non-local references, travel over two hops. As we've seen in the four-

processor case, two-hop traffic is noticeably slower than one-hop memory traffic. This is seen very clearly in Figure 7.

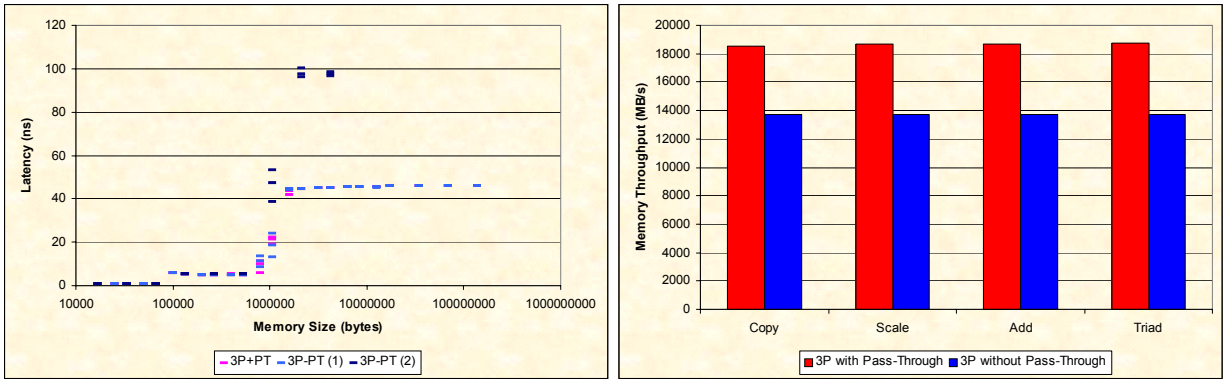

Figure 7. Effect of Pass-Through Modules on Three-Processor Memory Performance

In this figure "3P+PT" refers to a three-processor configuration that includes the pass-through module. The label "3P-PT (1)" refers to a three-processor configuration without the pass-through module where the data was fetched from a nearest neighbor. (Only one hop was required.) The "3P-PT (2)" case is also without the pass-through module, but data was fetched by a processor on one end from a processor on the opposite end of the system.

This figure shows that the single-hop latencies are the same whether the system employs the pass-through module or not, at least in these experiments. In part this is an artifact of the way in which a portion of the experiment was set up. The system latency using the pass-through module (45 ns) does accurately reflect the latency for that case. It is also the case that the one-hop latency for a system without the pass-through module will sometimes be 45 ns. In order for this to happen, the data must be requested by the processor in the middle of the system so that snoop requests are only one hop, which is what this experiment did. The latencies would be somewhat longer when one of the end processors requests data from the middle processor. Latencies are longer still when one end processor requests data of the other end processor, which is shown in this figure as the "3P-PT (2)" case.

Memory latency is much better when the pass-through module is used, and so is the throughput. Stream only measures local memory throughput, so the differences reflected here represent the effect of the longer snoop delays. Local memory throughput is over 18 GB/s using the pass-through module, but below 14 GB/s without it because the snoops take longer to complete.

2.7 Processor Frequency Scaling and Memory Gearing

Another curious effect occurs in the new Opteron processor systems, and that is memory gearing. Opteron processors use an integrated memory controller with a synchronous interface to memory. That means the processor frequency and the memory frequency must mesh; that is, the processor frequency must be an integer multiple of the memory frequency. When the native processor and memory frequencies don't mesh, the memory frequency is slowed to a point where they do. This

is called *memory gearing*, and it has the effect of slowing memory down in some cases. Figure 8 shows the extent to which memory gearing is needed in these systems. This figure shows the effect of memory gearing on 667 MHz memory such as that used in the x3755, but it also shows how memory gearing affects 533 MHz memory such as may be used in competing systems.

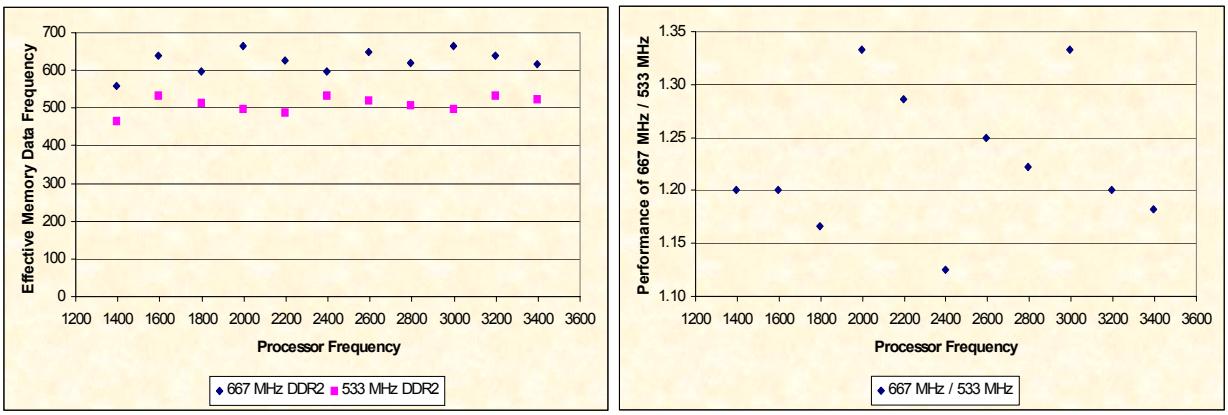

Figure 8. Theoretical Effect of Memory Gearing by Processor Frequency

The figure shows that memory performance is likely to be better for some frequencies than for others. For example, using 667 MHz memory, 2.0 GHz and 3.0 GHz processor frequencies would have better memory performance than, say, 1.8 GHz or 2.4 GHz processors. It also shows that the frequencies that are best for 667 MHz are different than those that are best for systems using 533 MHz memory. The relative performance chart (on the right) shows how much faster 667 MHz memory will be than 533 MHz memory. For example, at 2.0 GHz processor frequency the 667 MHz memory is clocked at its full rate whereas the 533 MHz memory must be clocked down to 500 MHz, making the 667 MHz memory a full 33% faster.[1]

3. 64-Bit Floating-Point Performance

Processor performance is an area where few changes beyond incremental improvements in technology are expected. The 2.8 GHz dual-core processors perform with no surprises. To measure this type of performance we use the High-Performance Linpack (HPL) benchmark.

The Linpack benchmark measures 64-bit floating-point vector performance, with long, easily exploited vectors that fit within available cache [3]. 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 (SSE2) units in the system. Performance is also a function of the problem size and the available system memory. HPL is a benchmark that employs multiple processes that communicate by sending messages over an interface called MPI[4]. As the problem size grows, the benchmark is increasingly dominated by floating-point operations and less influenced by its communication behavior, but the rate of improvement slows as the system approaches its

1. Of course this statement only applies to theoretical hardware performance. Application performance is affected by many factors, of which the effective memory clock frequency is only one.

asymptotic performance limit. In our tests using 64 GB of memory the system reached a little over 39 GF/s, which is 87.5% of the system peak performance. (With four dual-core processors at 2.8 GHz the hardware peak performance is 44.8 GF/s.) More memory would allow the performance to go even higher, but only by a very small amount. The performance profile of HPL on this system can be seen in Figure 9.

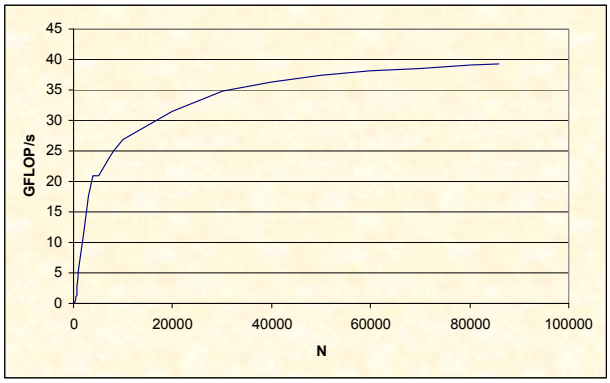

Figure 9. Linpack Performance Using Four 2.8 GHz Dual-Core Processors

4. 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[1], 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. Base and peak results are usually paired together, so the benchmarks are commonly referred to as SPEC CINT2000 for the integer speed benchmark, SPEC CINT2000 Rates for the integer rate benchmark, and similarly, CFP2000 and CFP2000 Rates for the floating-point speed and rate benchmarks.

The integer benchmarks use 14 integer applications. 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 both memory performance and floating-point performance, although large caches can also have impact on the outcome.

1. The geometric mean of a series of k numbers is the k^{th} root of the product of those numbers. For example, the geometric mean of x_1, x_2, x_3, x_4 and x_5 would be $(x_1\, x_2\, x_3\, x_4\, x_5)^{1/5}$.

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. 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 which compiler optimization flags may be used to compile the applications. Proper execution of the base benchmarks requires 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 our 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.

4.1 Integer Results

Figure 10 shows the results of the integer benchmarks by processor frequency. Because the benchmark is highly cache-friendly, both speed and rate results show near perfect scalability with processor frequency. And although these results are not shown here, it is a fairly safe extrapolation to say that this benchmark scales very closely with the number of processors, too.

4.2 Floating-Point Results

Figure 11 shows the results of the floating-point benchmarks by processor frequency. We can see from the charts that performance scales quite linearly with processor frequency. The reason is that there is sufficient memory throughput within the system so that the memory performance does not become the limiting factor.

As mentioned earlier in this section, the performance of this benchmark suite is strongly dependent on memory performance. This touches on two aspects of the Opteron processor design, both having to do with the integrated memory controller. The first is that memory performance is improved with increasing processor frequency. As the processor frequency goes up, requests to the memory controller become faster. This contributes significantly to the linearity of the results in Figure 11.

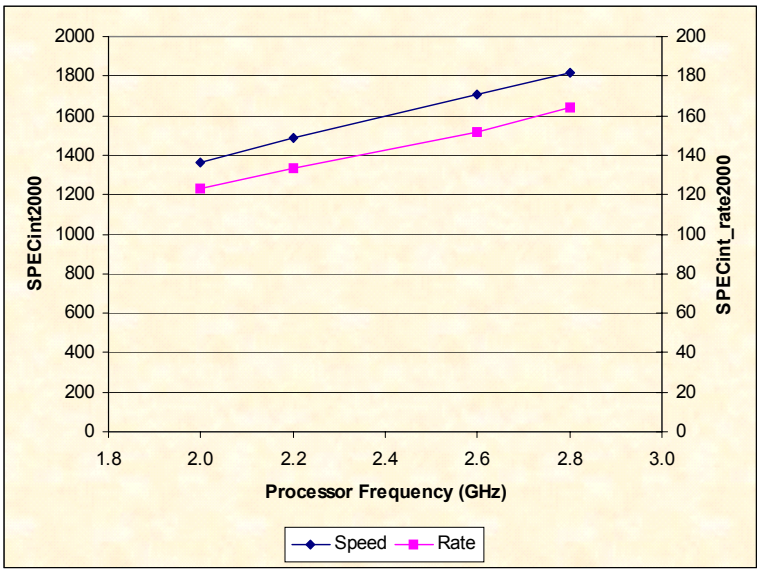

Figure 10. SPEC CINT2000 Speed and Rate Results by Processor Frequency

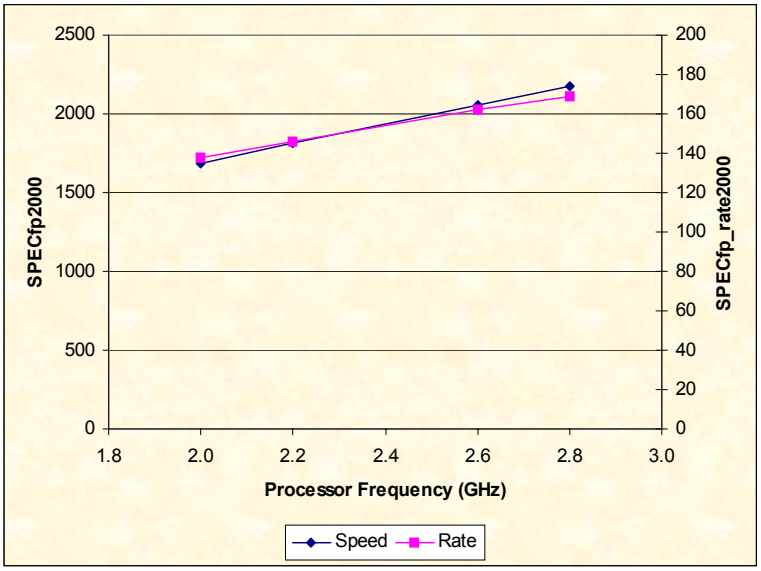

Figure 11. SPEC CFP2000 Speed and Rate Results by Processor Frequency

The second aspect of the integrated memory controller is that memory bandwidth increases as processors are added to the system. We do not show these results, but as processors are added to a system, it is reasonable to expect the performance increase of the rate benchmark to be very close to linear. We would expect the processor scaling to be nearly perfect from one to three processors. We also expect the scaling to be slightly lower than perfect linear scaling for four processors because of the drop in memory performance going from three processors to four, as demonstrated in previous sections.

5. Conclusions

The x3755 is a new server from IBM that supports up to four AMD Opteron processors. It has an innovative design that supports faster memory when using all 32 DIMM slots, and a pass-through module that gives superior performance for three-processor systems, in some cases surpassing the performance of four-processor systems. Based on the findings presented in this paper, we make the following conclusions:

1. Enabling the Node Interleave setting in the BIOS results in significant performance loss.
2. Enabling ChipKill in the BIOS increases the robustness of the memory system and does not decrease memory performance.
3. The x3755 shows nearly perfect linear processor scaling from one to three processors, and three processors provide better memory throughput by about 8% than the four-processor case.
4. The pass-through module adds significant performance to the system. It increases memory throughput by over 30% and decreases memory latency by more than 50% over a three-processor configuration without it.
5. Local (unloaded) memory latency for one to three processors is quite fast. We measure approximately 45 ns. Remote latency is somewhat slower at approximately 75 ns.
6. We measure local latency for four processors of 70 ns, though the one-hop latency is also 75 ns just as it is for the two- and three-processor case. The two-hop latency is just under 100 ns. The higher four-processor local memory latency is due to the additional time it takes to complete snoop requests.
7. Loading memory with two concurrent references adds from 20 ns to 30 ns to memory latency for random accesses.
8. The 64-bit floating-point performance is quite high, achieving over 39 gigaflops (87.5%) on High-Performance Linpack.
9. The SPEC CPU2000 performance is also in line with expectations.

6. References

[1] Memory Bandwidth: Stream Performance Results, http://www.cs.virginia.edu/stream/.
[2] Timothy J. Dell, “A White Paper on the Benefits of Chipkill - Correct ECC for PC Server Main Memory,” IBM Microelectronics Division, November 1997, http://www-03.ibm.com/servers/eserver/pseries/campaigns/chipkill.pdf .
[3] 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.
[4] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra, MPI—The Complete Reference, Vol. 1, 2nd Ed., MIT Press, 1996.
[5] SPEC CPU2000, http://www.spec.org/cpu2000/.

IBM Systems and Technology Group

Department MX5A

Research Triangle Park NC 27709

Produced in the USA.

08-06