A Comparison of Single-Core and Dual-Core Opteron Processor Performance for HPC

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

Abstract

Dual-core AMD Opteron™ processors represent the latest significant development in microprocessor technology. In this paper, using the IBM® eServer™ 326, we examine the performance of dual-core Opteron processors. We measure the core performance under ideal work loads using the Linpack HPL benchmark, and show it to be 60% faster than the fastest single-core Opteron. We measure unloaded memory latency and show that the dual-core processor's slower clock frequency makes the latency longer. However, we also show that its memory throughput is 10% greater than single-core processors. Finally, we show that dual-core Opteron processors offer a significant performance advantage even on realistic applications, such as those represented by the SPEC® CPU2000 benchmark suite.

1. Introduction

It is well-recognized that computer processors have increased in speed and decreased in cost at a tremendous rate for a very long time. This observation was first made popular by Gordon Moore in 1965, and is commonly referred to as Moore's Law. Specifically, Moore's Law states that the advancement of electronic manufacturing technology makes it possible to double the number of transistors per unit area about every 12 to 18 months. It is this advancement that has fueled the phenomenal growth in computer speed and accessibility over more than four decades. Smaller transistors have made it possible to increase the number of transistors that can be applied to processor functions and reduce the distance signals must travel, allowing processor clock frequencies to soar. This simultaneously increases system performance and reduces system cost. All of this is well-understood. But lately Moore's Law has begun to show signs of failing. It is not actually Moore's Law that is showing weakness, but the performance increases people expect and which occur as a side effect of Moore's Law.

One often associates performance with high processor clock frequencies. In the past, reducing the size of transistors has meant reducing the distances between the transistors and decreasing transistor switching times. Together, these two effects have contributed significantly to faster processor clock frequencies. Another reason processor clocks could increase is the number of transistors available to implement processor functions. Most processor functions, for example, integer addition, can be implemented in multiple ways. One method uses very few transistors, but the path from start to finish is very long. Another method shortens the longest path, but it uses many more transistors. Clock frequencies are limited by the time it takes a clock signal to cross the longest path within any stage. Longer paths require slower clocks. Having more transistors to work with allows more sophisticated implementations that can be clocked more rapidly.

But there is a down side. As processor frequencies climb, the amount of waste heat produced by the processor climbs with it. The ability to cool the processor inexpensively within the last few years has become a major factor limiting how fast a processor can go. This is offset, somewhat, by reducing the transistor size because smaller transistors can operate on lower voltages, which allows the chip to produce less heat. Unfortunately, transistors are now so small that the quantum behavior of electrons can affect their operation. According to quantum mechanics, very small particles such as electrons are able to spontaneously tunnel, at random, over short distances. The transistor base and emitter are now close enough together that a measurable number of electrons can tunnel from one to the other, causing a small amount of leakage current to pass between them, which causes a small short in the transistor. As transistors decrease in size, the leakage current increases. If the operating voltages are too low, the difference between a logic one and a logic zero becomes too close to the voltage due to quantum tunneling, and the processor will not operate. In the end, this complicated set of problems allows the number of transistors per unit area to increase, but the operating frequency must go down in order to be able to keep the processor cool.[1]

This issue of cooling the processor places processor designers in a dilemma. The market has high expectations that each new generation of processor will be faster than the previous generation; if not, why buy it? But quantum mechanics and thermal constraints may actually make successive generations *slower*. On the other hand, later generations will also have more transistors to work with and they will require less power. So, what is a designer to do? Manufacturing technology has now reached the point where there are enough transistors to place two processor cores – a dual-core processor – on a single chip. The trade-off that must now be made is that each processor core is slower than a single-core processor, but there are two cores, and together they may be able to provide greater throughput even though the individual cores are slower. Each following generation will likely increase the number of cores and decrease the clock frequency.

The slower clock speed has significant implications for processor performance, especially in the case of the AMD Opteron processor. The fastest dual-core Opteron processor will have higher throughput than the fastest single-core Opteron, at least for work loads that are processor-core limited[2], but each task may be completed more slowly. The application does not spend much time waiting for data to come from memory or from disk, but finds most of its data in registers or

1. The thermal envelope can be extended by changing from air to liquid cooling, but liquid cooling is much more expensive and even that has its limitations.
2. A core-limited work load is one in which the limiting factor to performance is the clock frequency of the processor. In order for this to occur, all, or nearly all, of the data in current use must be very close to the processor on which the application is running. The data, sometimes called the working set of the application, must be either in registers or cache.

cache. Since each core has its own cache, adding the second core doubles the available cache, making it easier for the working set to fit. For dual-core to be effective, the work load must also have parallelism that can use both cores. When an application is not multi-threaded, or it is limited by memory performance or by external devices such as disk drives, dual-core may not offer much benefit, or it may even deliver less performance.

Opteron processors use a memory controller that is integrated into the same chip and is clocked at the same frequency as the processor. Since dual-core processors use a slower clock, memory latency will be slower for dual-core Opteron processors than for single-core, because commands take longer to pass through the memory controller. Applications that perform a lot of random access read and write operations to memory, applications that are latency-bound, may see lower performance using dual-core. On the other hand, memory bandwidth increases in some cases. Two cores can provide more sequential requests to the memory controller than can a single core, which allows the controller to interleave commands to memory more efficiently.

Another factor that affects system performance is the operating system. The memory architecture is more complex, and an operating system not only has to be aware that the system is NUMA (that is, it has Non-Uniform Memory Access), but it must also be prepared to deal with the more complex memory arrangement. It must be dual-core-aware. The performance implications of operating systems that are dual-core-aware will not be explored here, but we state without further justification that operating systems without such awareness show considerable variability when used with dual-core processors. Operating systems that are dual-core-aware show better performance, though there is still room for improvement.

The remainder of this paper describes benchmarks and benchmark results that illustrate the principles just described. Section 2 describes the design of Opteron processors. Section 3 presents the performance on work loads that are core-limited and ideally suited for dual-core processors. Section 4 presents the memory performance of single- and dual-core processors. Section 5 presents the performance of the SPEC CPU2000 benchmark. SPEC CPU2000 is a collection of 26 real applications, and, as such, provides a realistic representation of how the systems are likely to be used. Finally we summarize our conclusions about single- and dual-core processors.

2. Processor Architecture

At a high level, the Opteron processor architecture consists of several parts: the processor core, two levels of cache, a memory controller (MCT), three coherent HyperTransport™ (cHT) links, and a non-blocking crossbar switch that connects the parts together. A single-core Opteron processor design is illustrated in Figure 1. The cHT links may be connected to another processor or to peripheral devices. The NUMA design is apparent from the diagram, as each processor in a system has its own local memory, memory to which it is closer than any other processor. Memory commands may come from the local core, or from another processor or a device over a cHT link. In the latter case the command comes from the cHT link to the crossbar, and from there to the MCT. The local processor core does not see or have to process outside memory commands, although some commands may cause data in cache to be invalidated or flushed from cache.

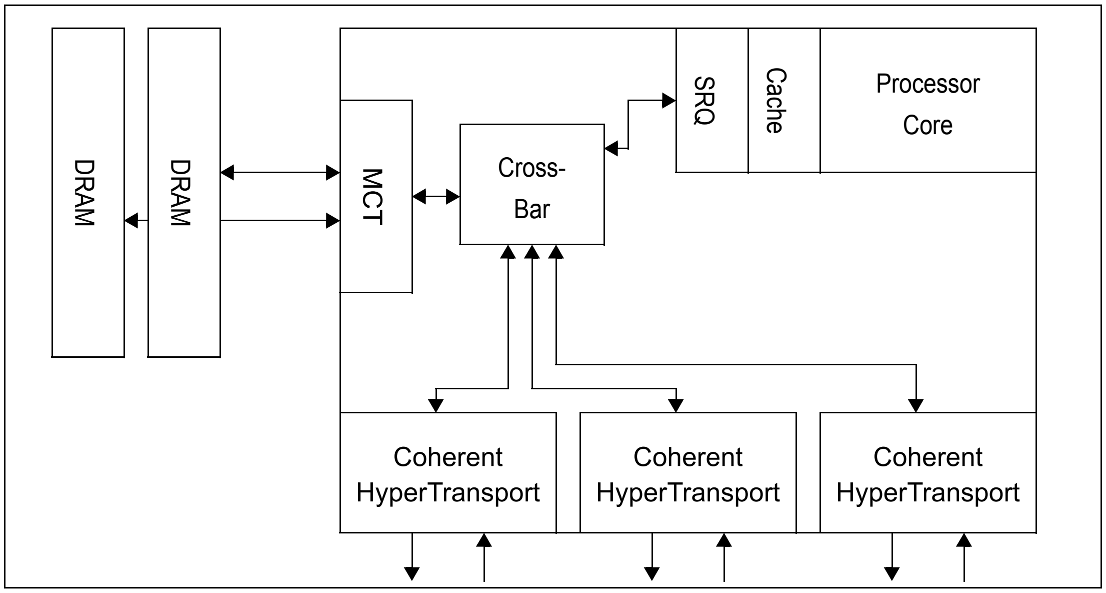

Figure 1. Single-Core Opteron Processor Block Diagram

The dual-core design is illustrated in Figure 2. What is most significant is how similar it is to the single-core design. It adds a second processor core, processor cache and System Request Queue (SRQ), connected directly to the crossbar switch. The second core does not share cache with the first. This is desirable as, from experience, cache thrashing can be a problem in shared-cache designs. The crossbar switch has been designed to support the full load of all components. But the memory performance hasn't been significantly increased, and with the increased compute capacity comes additional memory loading. In the end the memory controller and memory are more likely to be a bottleneck to system performance than is the case with single-core processors.

From the operating system perspective, code that manages the processor/memory/thread affinity must take into account that there are two processor cores that share a local memory pool. This is not a difficult transition to make, but failing to make the transition correctly means applications may have their physical memory allocated remotely more often than necessary. In a NUMA-aware operating system designed for single-core processors, each processor has its own unique pool of local memory to draw from. An operating system designed for dual-core processors must have local memory pools that are shared among cores, and it must map each processor core to the correct memory pool. If it fails to do so, physical memory will be allocated remotely at times when it is not necessary to do so, and performance will be lost.

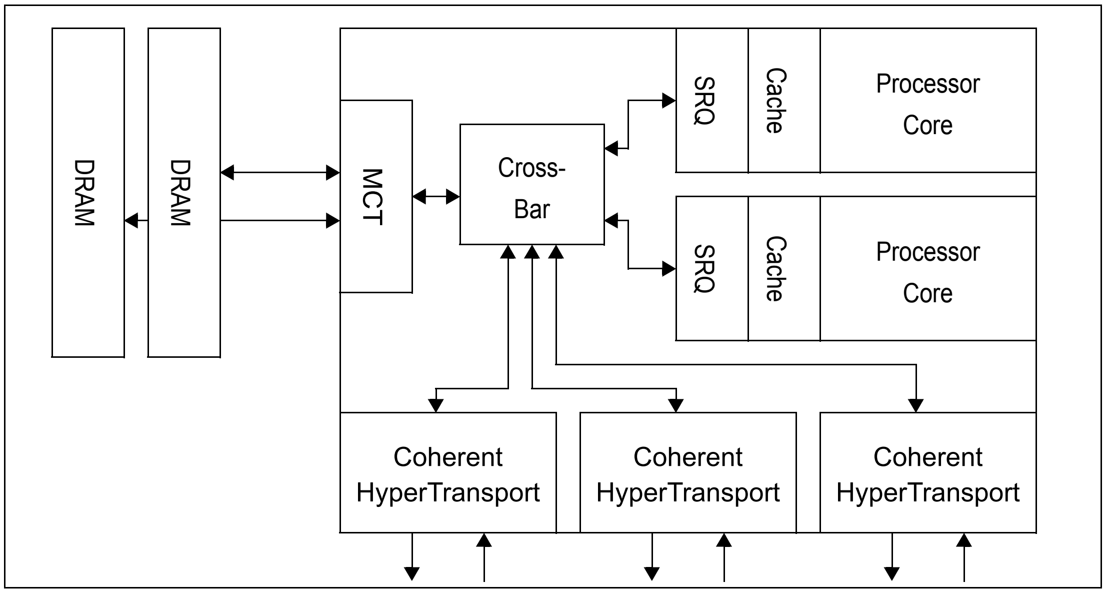

Figure 2. Dual-Core Opteron Processor Block Diagram

3. Processor Core Performance

It was mentioned in Section 1 that dual-core processors are well-suited for parallel, core-limited work loads. A popular benchmark that has those characteristics is the Linpack HPL benchmark. Linpack performs an operation from linear algebra called LU Factorization. (A more complete description of Linpack may be found in [1], [2] and [3]. The benchmark itself may be found in [4].) It is highly parallel and stores most of its working data set in processor cache. It makes relatively few references to memory for the amount of computation it performs. The processor operations it does perform are predominantly 64-bit floating-point vector operations. Our implementation uses a high-performance library of linear algebra routines from Kazushige Goto [5] to perform the core operations. The results are given in Figure 3.

The feature of greatest significance in this figure is that the dual-core processors live up to their potential almost perfectly. Each processor core has an SSE2 functional unit that is capable of generating two 64-bit floating-point results per clock. Thus the fastest dual-processor, single-core system is capable of up to 2.6 GHz x 2 results per clock x 2 cores = 10.4 GF/s. The fastest similar dual-core system is capable of up to 2.2 GHz x 2 results per clock x 4 cores = 17.6 GF/s, or potentially up to 70% faster. In actual measurements the single-core system achieved 8.63 GF/s, whereas the dual-core system achieved 13.76 GF/s. In this test the dual-core system was about 60% faster out of a potential 70%, which seems very good. Other processor speeds did similarly well.

It is important to remember that this is a work load that is ideally suited for dual-core systems, in that it is both highly parallel and core-limited. Both characteristics are needed in order for dual-core processors to achieve their potential. An application must be cacheable to be core-bound, and Linpack is highly cacheable.

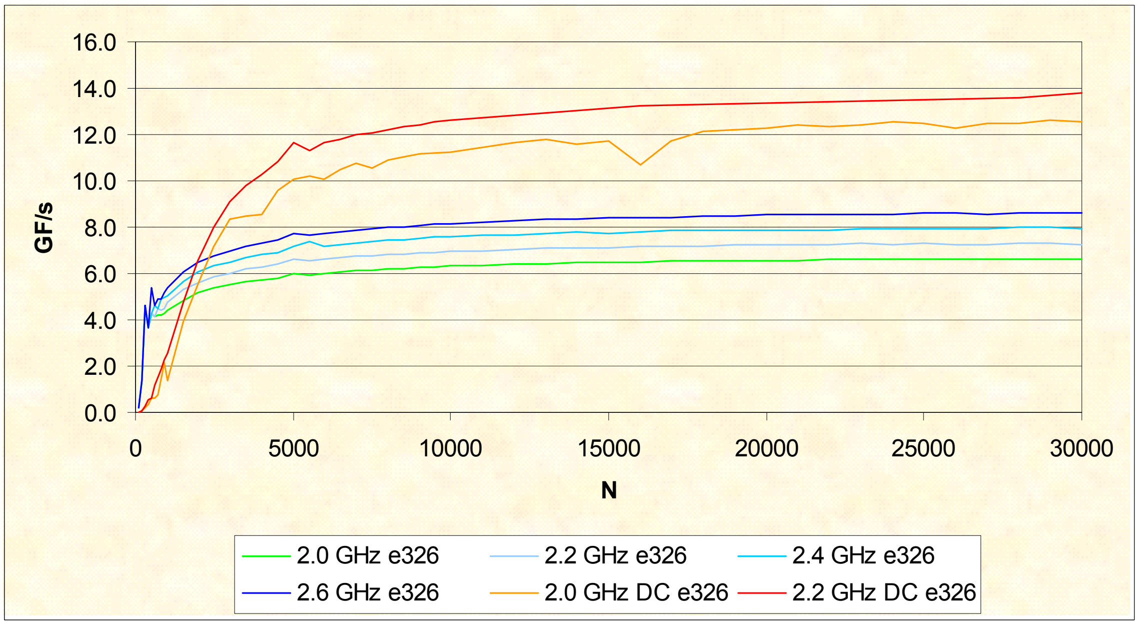

Figure 3. Dual Processor Single- and Dual-Core e326 Linpack Results

4. Memory Performance

In this section we describe the memory performance of the e326 with single- and dual-core processors. Memory performance is often viewed in two ways. The first is random access latency; the second is the throughput of sequential accesses. Each is relevant to a different work load, and each tells a different story.

Random access latency was determined by measuring how long it takes to chase a chain of pointers through memory. Only a single chain was followed at a time, so this is a measure of unloaded latency. Each chain stores only one pointer in a cache line, and each cache line is chosen at random from a pool of memory. The pool of memory simulates the working set of an application. When the memory pool is small enough to fit within cache, the benchmark measures the latency required to fetch data from cache. By adjusting the size of the memory pool one can measure the latency to any specific level of cache, or to main memory, by making the pool larger than all levels of cache.

We measured unloaded latency using a 2.6 GHz single-core Opteron processor and a 2.2 GHz dual-core Opteron processor. These processors were selected because they represent the fastest processors of each type available at this time. The results of these measurements are shown in Figure 4. Based on the processor frequencies alone, one would expect that the 2.6 GHz single-core processor would have lower latencies to cache and to memory, and from the figure we can see that this is the case. The 2.6 GHz processor has a latency to L1 cache of about 1.2 nanoseconds, whereas the L1 cache latency of the 2.2 GHz processor is about 1.45 nanoseconds. This corresponds exactly to the difference one would expect based on frequency differences alone. The L2 cache latencies are 5.6 and 6.6 nanoseconds, respectively, which also show perfect frequency scaling.

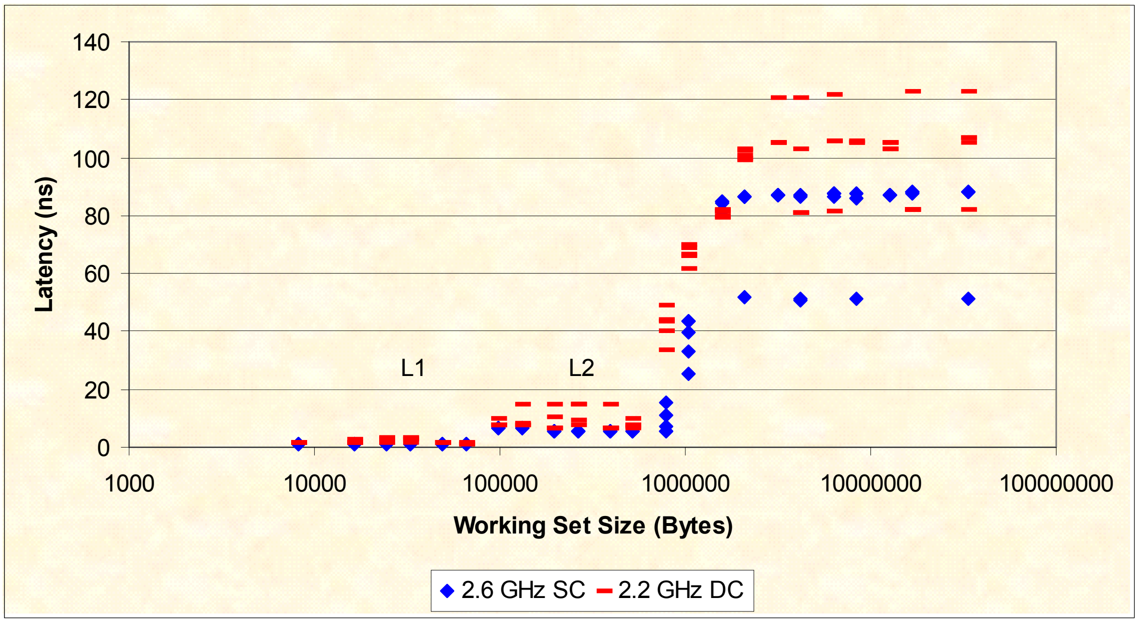

Figure 4. Single- and Dual-Core e326 Memory Latency

The surprise is that for each level in the memory hierarchy there are multiple latencies that may occur. Latencies are not spread randomly across a range, but they gravitate to specific discrete values. This suggests that each latency value reflects a different path through the system, but at this time the idea is only speculation. It definitely bears further investigation. One observation is that the benchmark left the memory affinitization implicitly to the operating system, so it is possible that the higher latencies are references to remote memory, while the lower latencies are to local memory. If true this would reflect an unexpected failure on the part of the benchmark or the operating system, and it would not explain why there are multiple latencies to cache. As mentioned before this will be investigated further, but what is important is to note that multiple latencies do happen.

The next few charts, Figure 5 and Figure 6, show the memory throughput for sequential memory accesses. These measurements were taken using the Triad loop of the Stream benchmark [1][6]. The surprise is that memory throughput is sometimes better for dual-core than for single-core. It has been speculated that dual-core would have lower memory throughput because of its lower clock frequency, and it was thought that the additional cores would cause additional memory contention. Figure 5 shows that the lower frequency clearly does have an effect when only a single processor or core is being used. But this should occur infrequently, seeing how these systems are used as heavily loaded servers rather than as lightly loaded workstations.

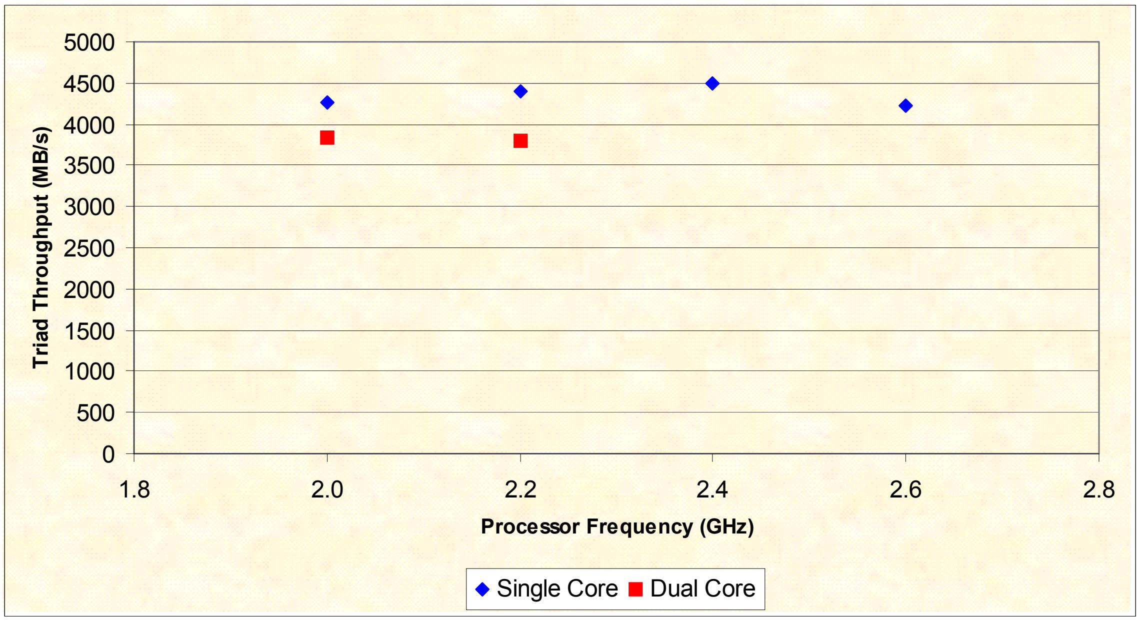

Figure 5. Single-Processor e326 Memory Throughput

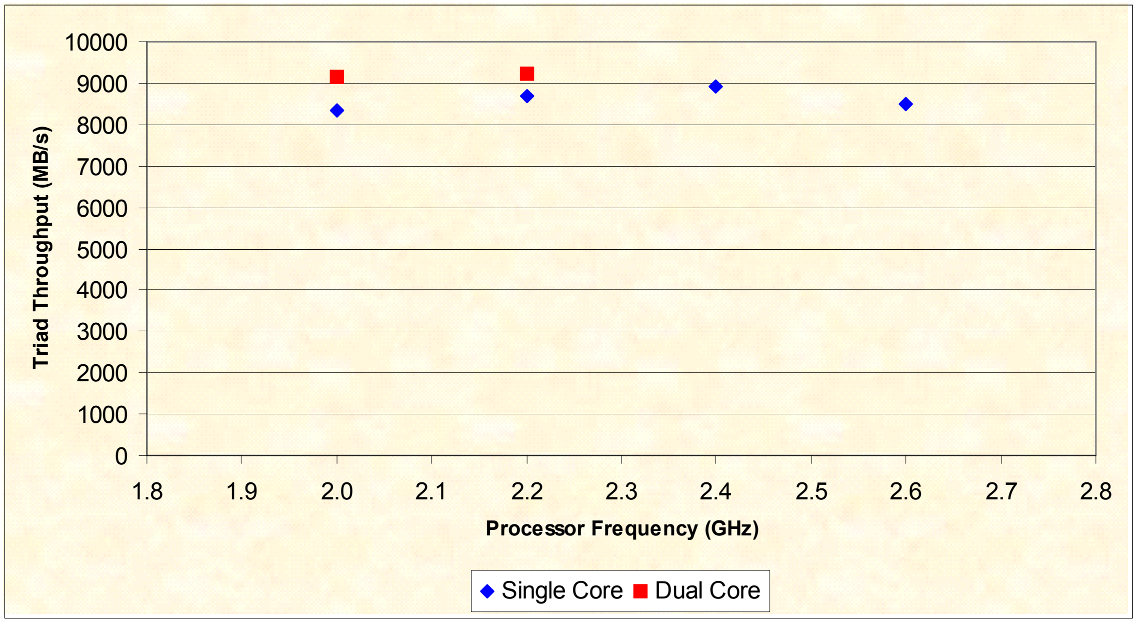

Figure 6. Dual-Processor e326 Memory Throughput

Memory contention, if it occurred, would be seen in Figure 6. In this figure each system is loaded with a single thread per core that generates memory references as fast as it is able, one write operation for every two read operations. The operating system allocates physical memory in such a way that references access local memory. If memory contention were occurring, the dual-core system would have lower throughput than the single-core system. But we can see from Figure 6 that the dual-core system has *higher* throughput, not lower, so we conclude something different is occurring. Instead of causing contention, the additional load allows the memory controller to operate more effectively, doing a better job of scheduling the commands to memory. So even though the memory controller is clocked more slowly and the memory it's connected to is the same, it is still able to achieve up to 10% higher bandwidth.

5. Application Performance

Linpack and Stream are valuable benchmarks, but they measure the performance of specific subsystems and don't represent some important ways in which systems are used. Applications often use much more than a single subsystem, so it is worthwhile to examine the effects of dual-core processors on more complex and realistic work loads. The SPEC CPU2000 benchmark suite is useful for doing that. It consists of two suites, CINT2000 and CFP2000. CINT2000 contains 14 integer applications, while CINT2000 contains 12 floating-point applications. Each application has been selected for inclusion in its suite as being representative of an activity or task commonly found on computer systems. Applications use the processor and memory subsystems almost exclusively; they are not dependent on disk or network performance. The suites are described in more detail on the SPEC Web site [7].

The suites may be used to measure single-core speed, as might be appropriate for a workstation environment, or throughput, as might be appropriate for a server environment. Speed measurements are accomplished by measuring how fast a single copy of the suite can be completed on a single core. Throughput measurements are accomplished by measuring how fast multiple copies can be completed, usually one copy for each processor core within the system. For our tests we only consider system throughput.

One characteristic of the CINT2000 suite, in addition to using primarily integer operations, is that it tends to be very cacheable. Each application fits most of its working set into a 1MB L2 cache. As such it tends to be very processor-core-intensive. This makes it ideally suited for dual-core processors, as the suite is highly parallel and processor-core-intensive. The results are presented in Figure 7. Once again, like the Linpack benchmark, they show that dual-core processors are very effective with this type of work load. The fastest dual-core Opteron processor, at 2.2 GHz, is 96% faster than the 2.2 GHz single-core processor, and 62% faster than the fastest single-core processor, at 2.6 GHz. So although the scaling isn't quite perfect, it is very close.

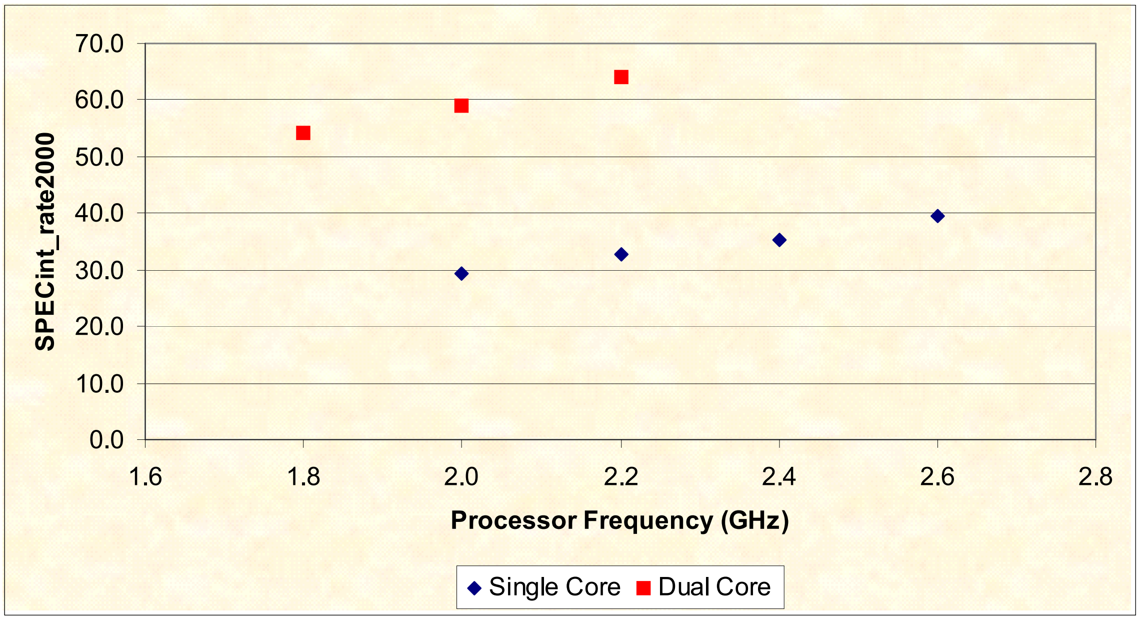

Figure 7. Dual-Processor SPEC CINT2000 Rate Performance

The CFP2000 benchmark suite is not as core-intensive as the CINT2000 suite. Processor and memory performance both have a significant effect on whether or not the benchmark performs well. As such it is a more thorough stress test of dual-core processor capabilities. The CFP2000 suite focuses primarily on floating-point operations, and is not as cacheable as CINT2000, but it is still highly parallel. The results are presented in Figure 8.

Interestingly, the dual-core processors significantly outperform the single-core processors on this benchmark. The 2.2 GHz dual-core processor achieves a score of 65.9 SPECfp_rate2000 marks, whereas the 2.2 GHz and 2.6 GHz single-core processors achieve 38.8 and 46.6, respectively. The dual-core processor is 70% faster than the 2.2 GHz single-core processor, and 35% faster than the 2.6 GHz single-core processor on this benchmark. So, even though the benchmark is significantly dependent on memory performance and the dual-core processors have only slightly better memory throughput, they still prove to be very capable on this benchmark suite.

There is an important lesson in this result. As we mentioned in Section 1 and later demonstrated in Section 3, dual-core processors are at their best when the work load is highly parallel and core-limited. But that does *not* mean they are ineffective elsewhere. On the CFP2000 suite dual-core processors significantly outperform single-core processors in spite of the memory component. The additional core performance adds significant value, even when the situation isn't ideal. Applications that require some, or even a lot of memory throughput will still see some benefit to dual-core, as long as the system has work it can do in parallel.

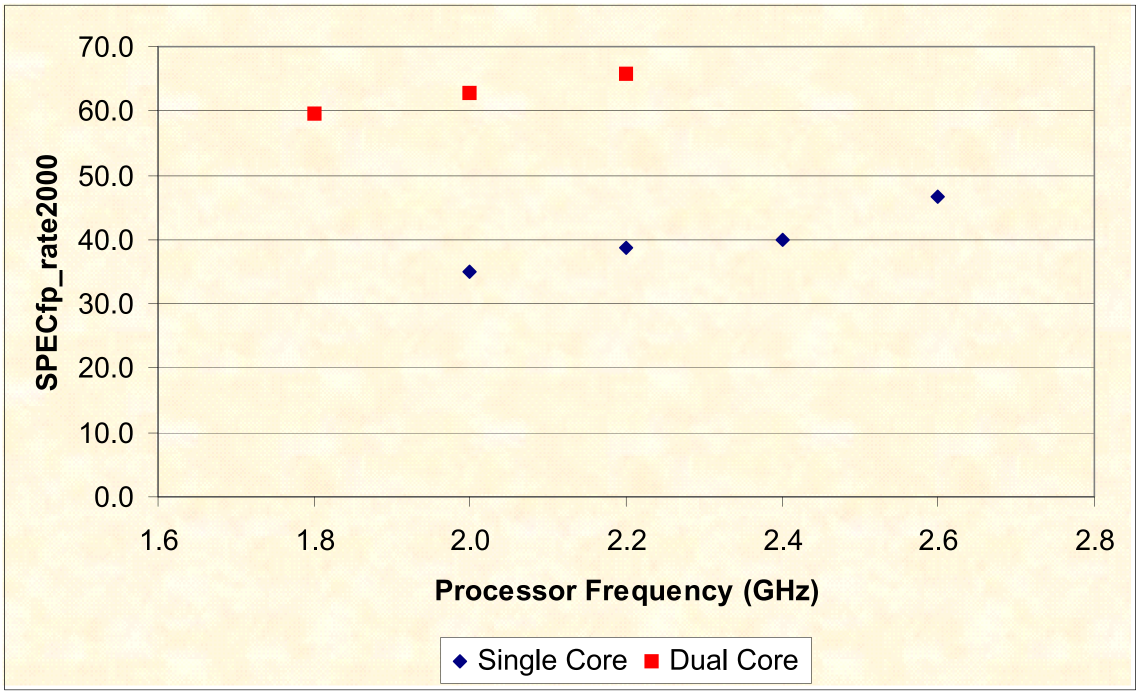

Figure 8. Dual-Processor SPEC CFP2000 Rate Performance

6. Conclusions

Following the lead of its single-core predecessor, dual-core Opteron processors have proven themselves to be very capable processors. Their slower clock speed makes them unsuitable for tasks that require fast sequential processing. But the second core can be used very effectively in applications where there is a high degree of parallelism that can be exploited. The fastest dual-core Opteron processor today operates at 2.2 GHz, whereas the fastest single-core Opteron operates at 2.6 GHz. This gives the fastest dual-core processor about 70% greater peak compute capacity than the fastest single-core processor. To its credit, on many applications it is approximately 60% faster than the fastest single-core Opteron. In other words, it achieves more than 85% of its potential performance.

Dual-core Opteron processors perform very well on applications that are highly parallel and whose performance is limited by the processor core, that is to say, by the frequency of the processor clock. Examples of this include the Linpack HPL and SPEC CINT2000 Rate benchmarks, where the dual-core systems were 60% faster. Applications that are parallel and limited by memory throughput may also see some performance improvement. Those that are highly throughput-limited may see up to 10% improvement, as we saw with the Stream Triad benchmark. Applications that are less dependent on memory throughput may see between 10% and 60% improvement, depending on the degree to which they are throughput-limited. For example, of this type of work load include the SPEC CFP2000 Rate benchmark, where we measured 35% better performance on dual-core. On the other hand, applications exhibiting little or no parallelism may be as much as 15% slower on dual-core processors.[1]

7. References

[1] 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.
[2] Jack J. Dongarra, Piotr Luszczek, and Antoint Petitet, “Linpack Benchmark: Past, Present, and Future,” http://www.cs.utk.edu/~luszczek/articles/hplpaper.pdf.
[3] 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.
[4] Linpack, www.netlib.org/linpack/index.html.
[5] Kazushige Goto, “High-Performance BLAS,” www.cs.utexas.edu/users/flame/goto/.
[6] STREAM: Sustainable Memory Bandwidth in High Performance Computers, www.cs.virginia.edu/stream/.
[7] SPEC CPU2000, http://www.spec.org/cpu2000/.

1. These comparisons are based on 2.2 GHz dual-core and 2.6 GHz single-core Opteron processors. Comparing processors with other frequencies may give different results.