Performance Analysis of the Alpha 21364-based HP GS1280 Multiprocessor

Zarka Cvetanovic
Hewlett-Packard Corporation
Zarka.Cvetanovic@hp.com

Abstract

This paper evaluates performance characteristics of the HP GS1280 shared memory multiprocessor system. The GS1280 system contains up to 64 Alpha 21364 CPUs connected together via a torus-based interconnect. We describe architectural features of the GS1280 system. We compare and contrast the GS1280 to the previous-generation Alpha systems: AlphaServer GS320 and ES45/SC45. We further quantitatively show the performance effects of these features using application results and profiling data based on the built-in performance counters. We find that the HP GS1280 often provides 2 to 3 times the performance of the AlphaServer GS320 at similar clock frequencies. We find the key reasons for such performance gains are advances in memory, inter-processor, and I/O subsystem designs.

1. Introduction

The HP AlphaServer 1280 is a shared memory multiprocessor containing up to 64 fourth-generation Alpha 21364 microprocessors [1]. Figure 1 compares the performance of GS1280 to the other systems using the SPECfp_rate2000, a multiprocessor throughput standard benchmark [8]. We show the published SPECfp_rate2000 results as of March 2003, with the exception of the 32P GS1280 for which the data was measured on an engineering prototype, but not published yet. We use floating-point rather than integer SPEC benchmarks for this comparison since several of the floating-point benchmarks stress memory bandwidth, while all integer benchmarks fit well in the MB-size caches and thus are not a good indicator of memory system performance. The results in Figure 1 indicate that GS1280 scales well in memory-bandwidth intensive workloads and has substantial performance advantage over the previous-generation Alpha platforms despite disadvantage in the processor clock frequency. We analyze key performance characteristics of the GS1280 in this paper to expose the key design features that allowed GS1280 to reach such performance levels.

The GS1280 system contains many architectural advances – both in the microprocessor and in the surrounding memory system - that contribute to its performance. The 21364 processor [1][16] uses the same core as the previous-generation 21264 processor [4]. However, 21364 includes three additional components: (1) an on-chip L2 cache, (2) two on-chip Direct Rambus (RDRAM) memory controllers and (3) a router. The combination of these components helped achieve improved access time to the L2 cache and local/remote memory.

These improvements enhanced single-CPU performance and contributed to excellent multiprocessor scaling. We describe and analyze these architectural advances and present key results and profiling data to clarify the benefits of these design features. We contrast GS1280 to two previous-generation Alpha systems, both based on a 21264 processor: GS320 – a 32-CPU SMP NUMA system with switch-based interconnect [2], and SC45 – a 4-CPU ES45 systems connected in a cluster configuration via a fast Quadrics switch. [4][5][6][7].

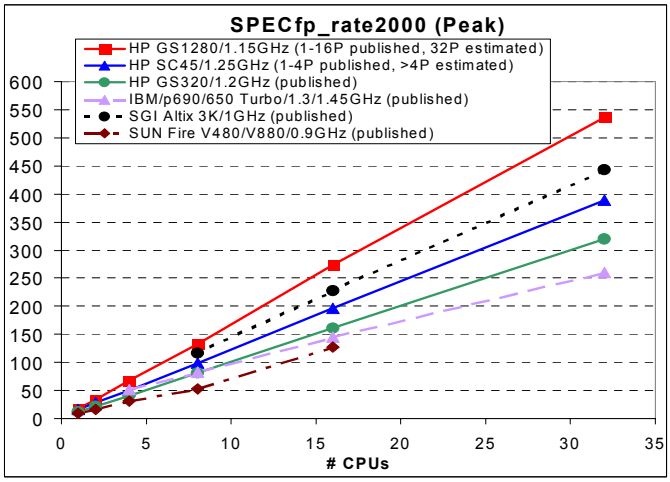

Figure 1. SPECfp_rate2000 comparison.

We include results from kernels that exercise memory subsystem [9][10]. We include profiling results for standard benchmarks (SPEC CPU2000 [8]). In addition, we analyze characteristics of representatives from 3 application classes that impose various levels of stress on memory subsystem and processor interconnect. We use profiles based on the built-in non-intrusive CPU hardware monitors [3]. These monitors are useful tools for analyzing system behavior with various workloads. In addition, we use tools based on the EV7-specific performance counters: Xmesh [11]. Xmesh is a graphical tool that displays run-time information on utilization of CPUs, memory controllers, inter-processor (IP) links, and I/O ports.

The remainder of this paper is organized as follows: Section 2 describes the architecture of the GS1280 system. Section 3 describes the memory system improvements in GS1280. Section 4 describes the inter-processor performance characteristics. Section 5 discusses application performance. Section 6 shows tradeoffs associated with memory striping. Section 7 summarizes comparisons. Section 8 concludes.

2. GS1280 System Overview

The Alpha 21364 (EV7) microprocessor [1] shown in Figure 2 integrates the following components on a single chip: (1) second-level (L2) cache, (2) a router, (3) two memory controllers (Zboxes), and (4) a 21264 (EV68) microprocessor core. The processor frequency is 1.15 GHz. The memory controllers and inter-processor links operate at 767 MHz (data rate). The L2 cache is 1.75 MB in size, 7-way set-associative. The load-to-use L2 cache latency is 12 cycles (10.4 ns). The data path to the cache is 16-bytes wide, resulting in peak bandwidth of 18.4 GB/s. There are 16 Victim buffers from L1 to L2 and from L2 to memory. The two integrated memory controllers connect processor directly to the RDRAM memory. The peak memory bandwidth is 12.3 GB/s (8 channels, 2 bytes each). There can be up to 2048 pages open simultaneously. The optional 5th channel is provided as a redundant channel. The four interprocessor links are capable of 6.2 GB/s each (2 unidirectional links with 3.1 GB/s each). The IO chip is connected to the EV7 via a full-duplex link capable of 3.1 GB/s.

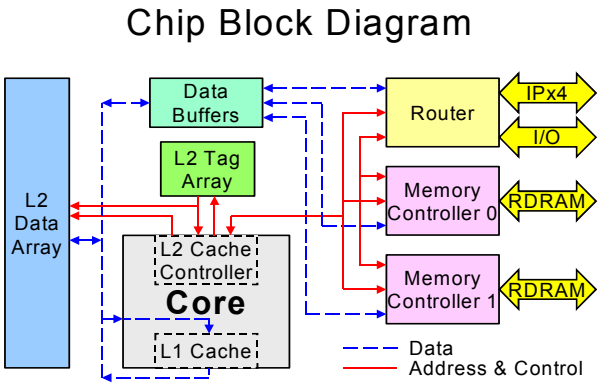

Figure 2. 21364 block diagram.

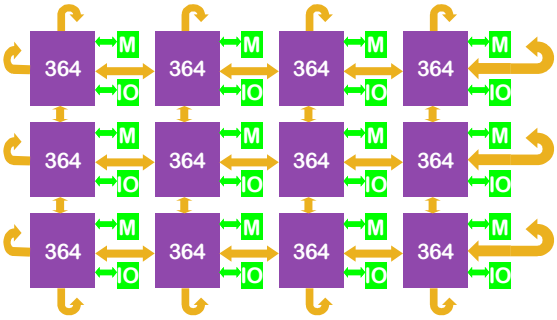

Figure 3. A 12-processor 21364-based multiprocessor.

The router [16] connects multiple 21364s in a two-dimensional, adaptive, torus network (Figure 3). The router connects to 4 links that connect to 4 neighbors in the torus: North, South, East, and West. Each router routes packets arriving from several input ports (L2 cache, ZBoxes, I/O, and other routers) to several output ports. (i.e., L2 cache, ZBoxes, I/O, and other routers). To avoid deadlocks in the coherence protocol and the network, the router multiplexes a physical link among several virtual channels. Each input port has two first-level arbiters, called the local arbiters, each of which selects a candidate packet among those waiting at the input port. Each output port has a second-level arbiter, called the global arbiter, which selects a packet from those nominated for it by the local arbiters.

The global directory protocol is a forwarding protocol [16]. There are 3 types of messages: Requests, Forwards, and Responses. A requesting processor sends a Request message to the directory. If the block is local, the directory is updated and a Response is sent back. If the block is in Exclusive state, the Forward message is sent to the owner of the block, who sends the Response to the requestor and directory. If the block is in Shared state (and the request is to modify the block), Forward/invalidates are sent to each of the shared copies, and a Response is sent to the requestor.

To optimize network buffer and link utilization, the 21364 routing protocol uses minimal adaptive routing algorithm. Only a path with minimum number of hops from source to destination is used. However, a message can choose the less congested minimal path (adaptive protocol). Both the coherence and adaptive routing protocols can introduce deadlocks in the 21364 network. The coherence protocol can introduce deadlocks due to cyclic dependence between different packet classes. For example, the Request packets can fill up the network and prevent the Response packets from ever reaching their destinations. The 21364 breaks this cyclic dependence by creating virtual channels for each class of coherence packets and prioritizing the dependence among these classes. By creating separate virtual channels for each class of packets, the router guarantees that each class of packets can be drained independent of other classes. Thus, a Response packet can never block behind a Request packet. A Request can generate a Block Response, but a Block Response cannot generate a Request.

Adaptive routing can generate two types of deadlocks: intra-dimensional (because the network is a torus, not a mesh) and inter-dimensional (arises in any square portion of the mesh). The intra-dimensional deadlock is solved with virtual channels: VC0 and VC1. The inter-dimensional deadlocks are solved by allowing message to route in one dimension (e.g. East-West) before routing in the next dimension (e.g. North-South) [12]. Additionally, to facilitate adaptive routing, the 21364 provides a separate virtual channel called the Adaptive channel for each class. Any message (other than I/O packets) can route through the Adaptive channel. However, if the Adaptive channels fill up, packets can enter the deadlock-free channels.

The previous-generation GS320 system uses a switch to connect four processors to the four memory modules in a single Quad Building Block (QBB) and then a hierarchical

switch to connect QBBs into the larger-scale multiprocessor (up to 32 CPUs) [2].

3. Memory Subsystem

In this section we characterize memory subsystem of GS1280 and compare to the previous-generation Alpha platforms. The section includes the analysis of local memory latency, memory bandwidth, single-CPU performance, and remote memory latency.

3.1 Local memory latency for dependent loads

The 21374 processor provides two RDRAM memory controllers with 12.3 GB/s peak memory bandwidth. Each processor can be configured with 0, 1, or 2 memory controllers. The L2 1.75MB on-chip cache is 7-way set associative. The L2 cache on ES45 and GS320 is 16MB, off-chip, direct-mapped.

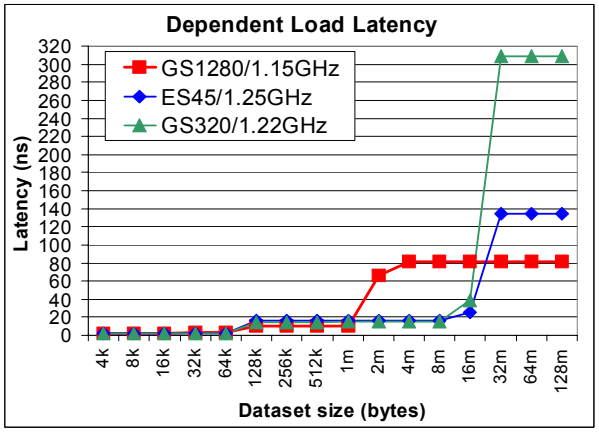

Figure 4. Dependent load latency comparison.

Figure 4 compares the dependent-load latency [9]. The dependent-load latency measures load-to-use latency where each load depends on the result from the previous load. The lower axis varies the referenced data size to fit in different levels of the memory system hierarchy. Data is accessed in a stride of 64 bytes (cache block). The results in Figure 4 show that GS1280 has 3.8 times lower "dependent-load" memory latency (32MB size) than the previous-generation GS320. This indicates that large-size applications that are not blocked to take advantage of 16MB cache will run substantially faster on GS1280 than on the 21264-based platforms. For data range between 1.75MB and 16MB, the latency is higher on GS1280 than on GS320 and ES45, since the block is fetched from memory on GS1280 vs. from the 16MB L2 cache on GS320/ES45. This indicates that the application sizes that fall in this range are likely to run slower on GS1280 than on the previous-generation platforms. For datasizes between 64KB and 1.75MB, latency is again much lower on GS1280 than GS320/ES45. That is because the L2 cache in GS1280 is on-chip, thus providing much lower access time than the off-chip caches in GS320/ES45.

Figure 5 shows dependent load latency on GS1280 as both dataset size and stride increase. This data indicates that the latency increases from ~80ns for open-page access to ~130ns for closed-page access (larger-stride access).

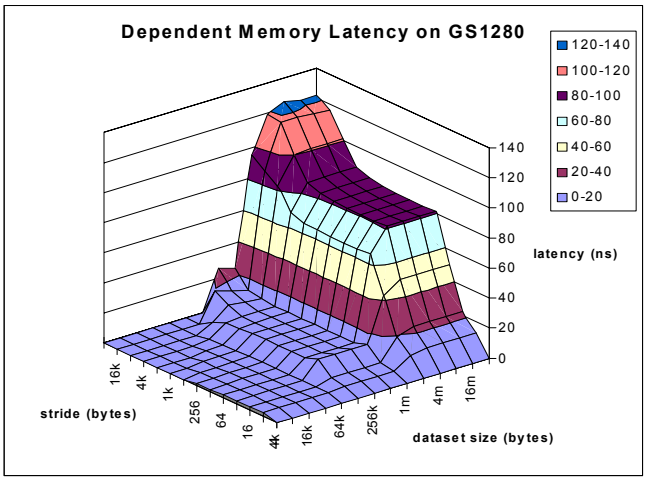

Figure 5. GS1280 dependent load latency for various strides.

3.2 Memory Bandwidth

The STREAM benchmark [10] measures sustainable memory bandwidth in megabytes per second (MB/s) across four vector kernels: Copy, Scale, Sum, and SAXPY.

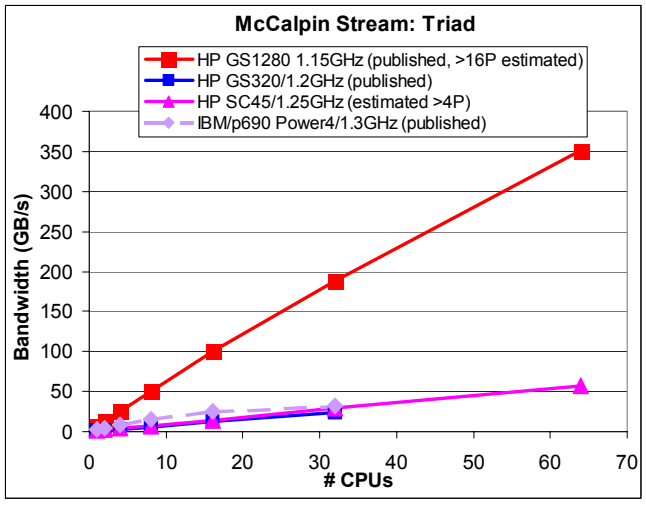

Figure 6. McCalpin STREAM bandwidth comparison.

We show only the results for the Triad kernel in Figure 6 (the other kernels have similar characteristics). This data indicates that the memory bandwidth on GS1280 is substantially higher than the previous-generation GS320 and all other systems shown.

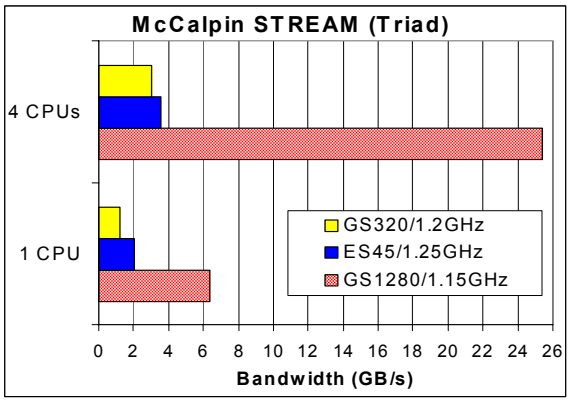

Figure 7. STREAM bandwidth for 1-4 CPUs.

Figure 7 indicates that GS1280 exhibits not only 1-CPU advantage in memory bandwidth (due to high-bandwidth memory-controller design provided by the 21364 processor), it also provides linear scaling in bandwidth as the number of CPUs increases. This is due to GS1280 memory design where each CPU has its own local memory, thus avoiding contention for memory between jobs that run simultaneously on several CPUs. This is not the case on ES45 and GS320, where four CPUs contend for the same memory. Therefore, bandwidth improvement from one to four CPUs on ES45/GS320 is less-than-linear (as indicated in Figure 7). The data in Figures 6 and 7 indicate that the memory-bandwidth intensive applications will run exceptionally well on GS1280. The advantage is likely to be even more pronounced as the number of CPUs increases. One such example is the SPEC throughput benchmarks shown in Figure 1.

3.3. Single-CPU performance: CPU2000

Figures 8 and 9 compare Instructions-per-Cycle (IPC) for floating-point (fp) and integer SPEC CPU2000 benchmarks on GS1280 vs. GS320 and ES45 [8].

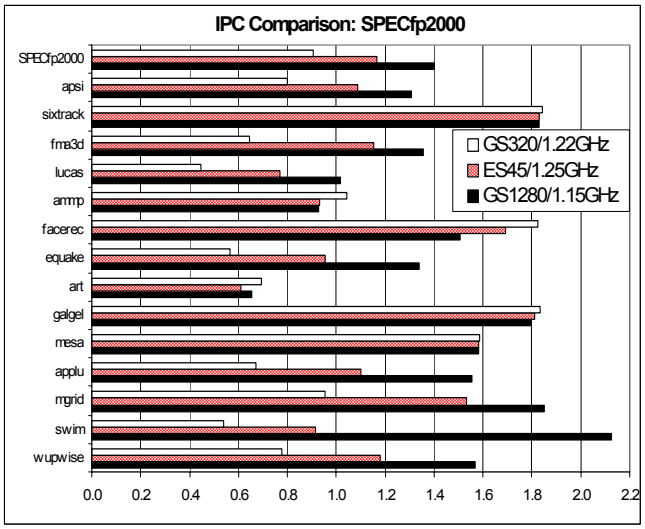

Figure 8. IPC for SPECfp2000.

On average, GS1280 shows advantage over both GS320 and ES45 in SPECfp2000, and comparable performance in SPECint2000. Note that some benchmarks demonstrate a substantial advantage on GS1280 over ES45/GS320. For example, swim shows 2.3 times advantage on GS1280 vs. ES45 and 4 times advantage vs. GS320. However, many other benchmarks show comparable performance (e.g. most integer benchmarks). Yet, there are cases where GS320 and ES45 outperform GS1280 (e.g. facerec and amp). In order to better understand the causes of such differences, we generated profiles that show memory controller utilization for all benchmarks (Figures 10 and 11).

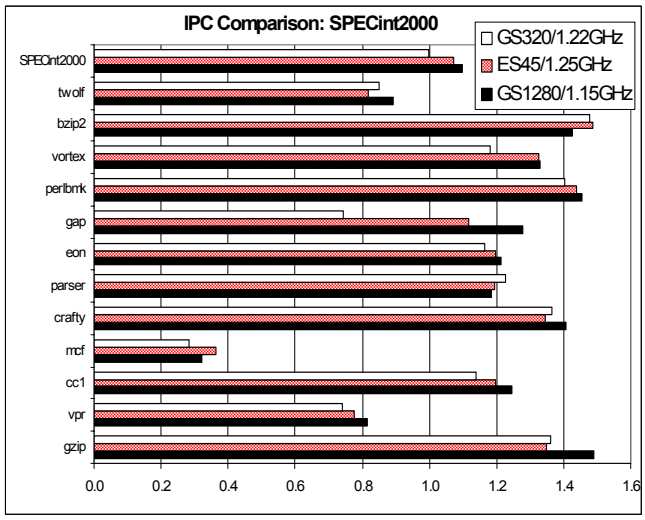

Figure 9. IPC for SPECint2000.

Figures 10 and 11 illustrate memory controller utilization profiling histograms for SPEC CPU2000 benchmarks on GS1280. The profiles are collected using the 21364 built-in performance counters and are shown as a function of the elapsed time for the entire benchmark run. This data indicates that the benchmarks with high memory utilization are the same benchmarks that show significant advantage on GS1280. Swim is the leader with 53% utilization, followed by applu, lucas, equake, and mgrid (20-30%), fma3d, art, wupwise, and galgel (10-20%). Interestingly, facerec has 8% utilization: still GS1280 has lower IPC than the other systems. That is due to the smaller cache size on GS1280 (1.75MB vs. 16MB on GS320/ES45). The simulation results show that facerec dataset fits in the 8MB cache, but not in the 1.75MB cache. Therefore, facerec accesses memory on GS1280, while it fetches data mostly from the 16MB cache on GS320 and ES45. Figure 4 illustrates that the cache access on GS320 is faster than the memory access on GS1280.

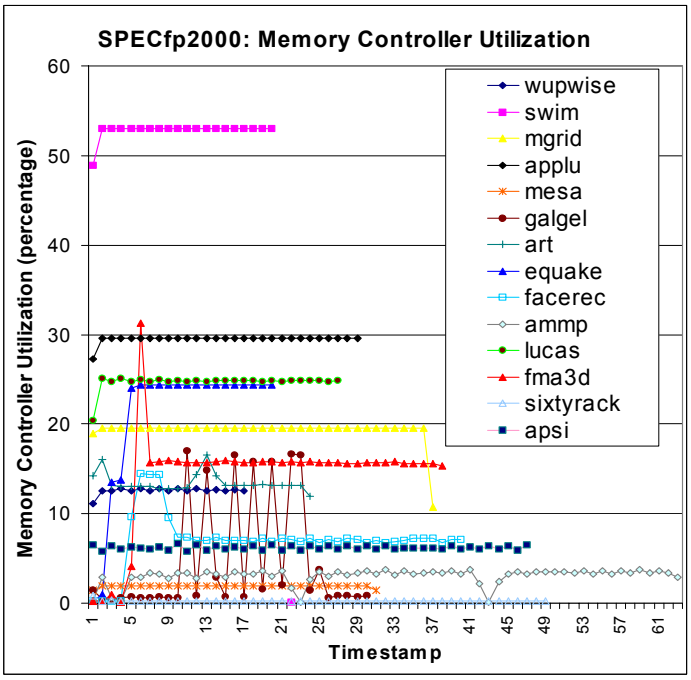

Figure 10. GS1280 memory controller utilization in SPECfp2000.

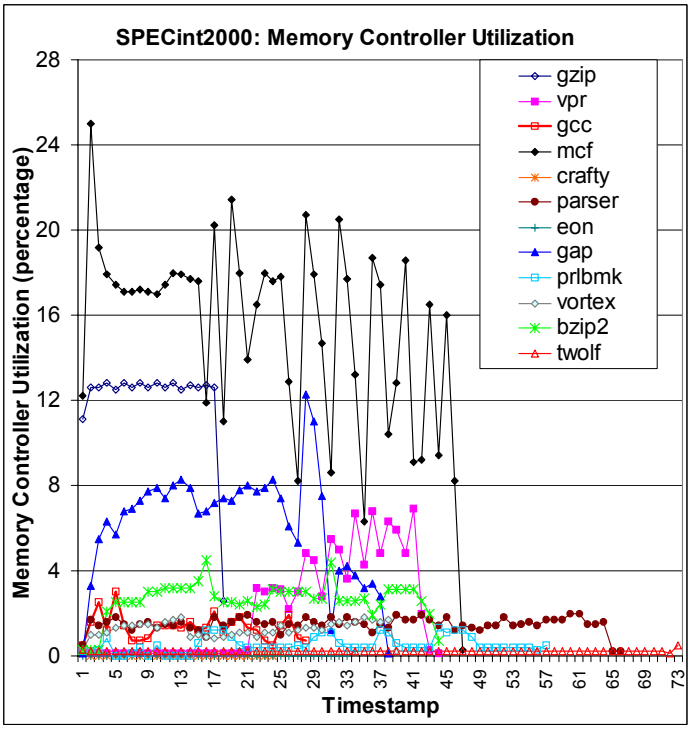

Figure 11. GS1280 memory controller utilization in SPECint2000.

3.4. Remote memory latency

In Sections 3.1 and 3.2 we contrasted local memory latency and bandwidth on GS1280 and previous-generation platforms. Local memory characteristics are important for the single-CPU workloads and multiprocessor workloads that fit well in processor's local memory. However, in order to characterize applications that do not fit well in local memory, we need to understand how local latency compares to the remote latency. Figure 12 compares local and remote memory latency on GS320 and GS1280. Latency is measured from CPU0 to all other CPUs in a 16-CPU system. Note that GS320 has two levels of latency: local (within a set of 4 CPUs called QBB) and remote (outside that QBB). The GS1280 system has many levels of remote latency, depending on how many hops need to be passed from source to destination.

Figure 12 indicates that GS1280 shows 4 times advantage in average memory latency on 16 CPUs. The advantage is even higher (6.6 times) when Read-Dirty instead of Read-Clean latencies are compared. Note that in the case of Read-Dirty, a cache block is read from another processor's cache rather than from memory.

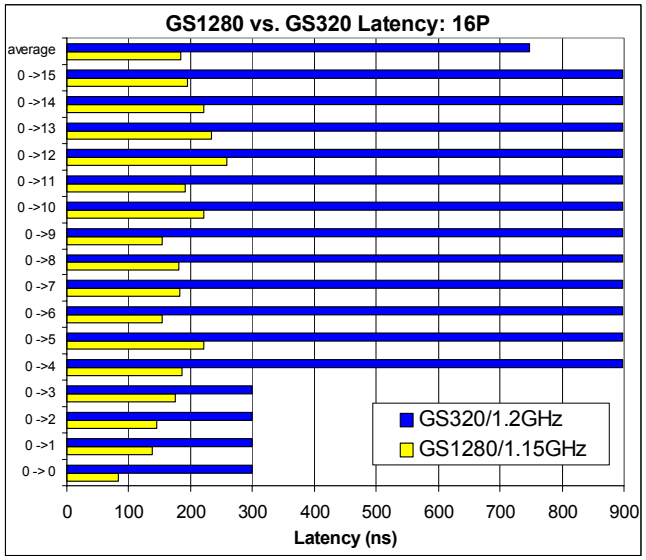

Figure 12. Local/remote latency on 16 CPUs.

83	145	186	154
139	175	221	182
181	221	259	222
154	191	235	195

Figure 13. Remote memory latencies (ns) on GS1280 (each square represents a CPU in a 16-CPU torus).

Figure 13 illustrates measured latency from node 0 to all other nodes in the 16-CPU GS1280 system (each square is a CPU within a 4x4 torus). The local memory latency of 83ns is increased to 139-154 ns for the 1-hop neighbors. Note that the 1-hop latency is the lowest for the neighbors on the same module (139 ns), and the highest for the neighbors that are connected via a cable (154 ns). The 2-hop latency is 175-195 ns (6 nodes are 2-hop away). The 4-hop latency (worst-case for 16 CPUs) is 259 ns (1 node is 4-hops away).

Figure 14 shows the average load-to-use latency as the number of CPUs increases. Figures 12 and 14 show that GS1280 has significant advantage over GS320 not only in local, but also in remote memory latency. This data indicates that applications that are not structured to fit well within a processor's local memory will run much more efficiently on GS1280 that on GS320. In addition, this advantage will be even more pronounced in applications that require high amount of data sharing (parallel workloads) due to efficient Read-Dirty implementation in GS1280.

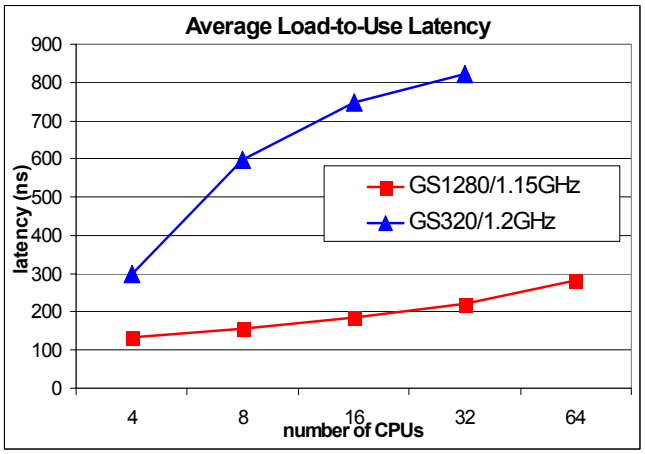

Figure 14. Remote memory latency for 4-64 CPUs.

4. Interprocessor Bandwidth

The memory latency in Figure 14 is measured on an idle system with only 2 CPUs exchanging messages. In this section, we evaluate interprocessor network response as the load increases. This is needed in order to characterize applications that require all CPUs to communicate simultaneously (more closely related to the real application environment).

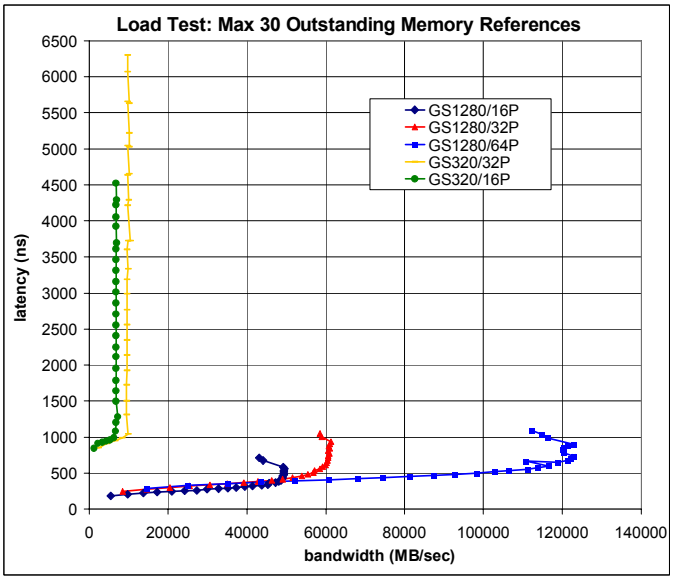

Figure 15. Load test comparison.

Figure 15 compares bandwidth under increasing load on GS1280 and GS320. Each CPU randomly selects another CPU to send a Read request to. The test is started with a single outstanding load (leftmost point). For each additional point, one outstanding load is added (up to 30 outstanding requests). In ideal case, bandwidth will increase (moving to the right), and latency will not change (line stays low and flat).

Figure 15 indicates that GS1280 shows increase in latency, but it is not nearly as high as in GS320. GS1280 is much more resilient to the load: bandwidth increases at much smaller latency increase. This is an important system feature for applications that require substantial inter-processor (IP) bandwidth. Figure 14 also indicates another interesting phenomenon: as the load is increased beyond saturation point, the delivered bandwidth starts to decrease. Although interesting from the theoretical point of view, this phenomenon has no implications on performance of real applications, as we have not observed any applications that operate even close to this point.

4.1 Shuffle Interconnect

We discovered that performance of an 8-CPU GS1280 configuration could be improved by a simple swap of the cables (which we call "shuffle")[12]. Figures 16 and 17 show how the connections are changed from the standard "torus" interconnect (Figure 16) to "shuffle" (Figure 17) in order to improve performance. The redundant North-South connections in an 8-CPU torus are used to connect to the furthest nodes to crate a shuffle interconnect.

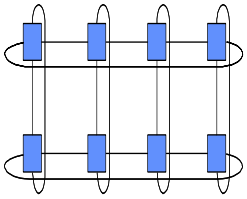

Figure 16. Torus.

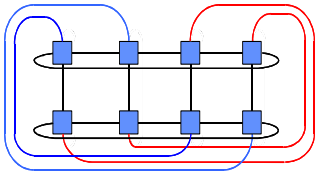

Figure 17. Shuffle.

Table 1 shows the performance improvement from shuffle vs. torus using a simple analytical model. Note that shuffle is more beneficial in rectangular rather than in square shaped interconnects (bisection width and worst-case latency). The benefits in average latency increase as the system size grows.

Table 1: Performance gains from shuffle.

	aver. latency	worst latency	bisection width
4x2	1.200	1.500	2.000
4x4	1.067	1.333	1.000
8x4	1.171	1.500	2.000
8x8	1.185	1.333	1.000
16x8	1.371	1.500	2.000
16x16	1.454	1.778	1.000

Figure 18 shows the performance gains from shuffle measured on an 8-CPU GS1280 prototype. We experimented with two shuffle routing approaches: (1) **shuffle with 1-hop:** shuffle links are used as the initial (and only) hop, (2) **shuffle with 2-hops:** shuffle links are used for 1 and 2 hops (e.g. we use shuffle links to alleviate load on horizontal links). The performance data indicates that 1-hop shuffle provides between 5% and 25% performance gain (depending on network load) vs. torus. The 2-hop shuffle provides lower additional (2-5%) gain.

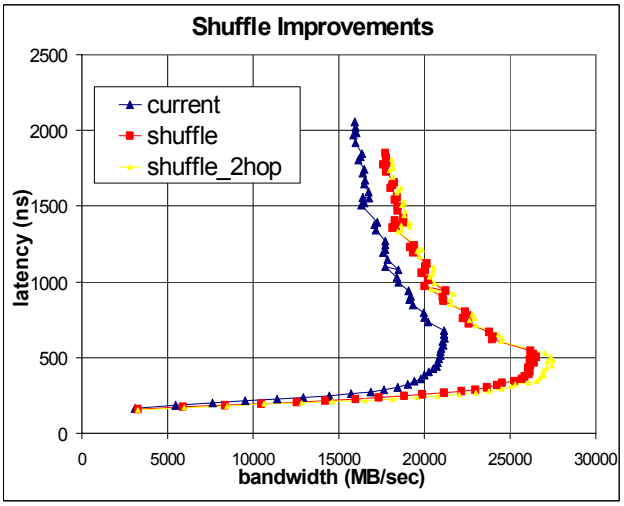

Figure 18. Performance Improvement from Shuffle.

5. Application performance

In this section, we compare GS1280 to the other systems using 3 types of applications: (1) CPU-intensive applications that do not stress either memory controller or inter-processor (IP) links, (2) memory-bandwidth intensive applications that stress memory bandwidth, but not the IP links (many MPI applications belong to this category), and (3) applications that stress both IP-links and memory bandwidth. An example of application that represents each class is included. The utilization of memory controllers and IP links is measured using the 21364 built-in performance counters [11].

5.1. CPU-intensive application: Fluent (CFD)

Fluent is the standard Computational Fluid Dynamics application [13]. For comparison, we selected a large case (l1) that models flow around a fighter aircraft (Figure 19). The results in Figure 19 are published as of March 2003. This data indicates that GS1280 shows comparable performance to ES45. Examining Figure 20 with measured utilization shows that the reason is that this application does not put significant stress on either memory controller or IP-links bandwidth. The large 16MB cache in ES45 often provides advantage in applications that can be blocked for cache re-use (such as Fluent).

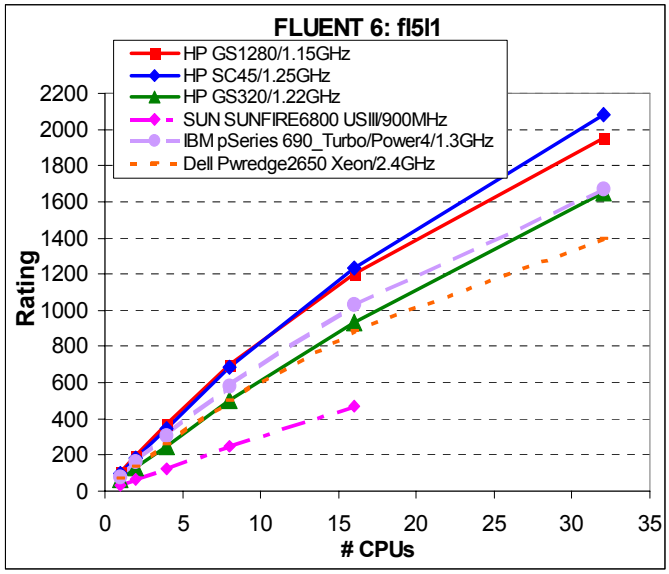

Figure 19. Fluent Performance.

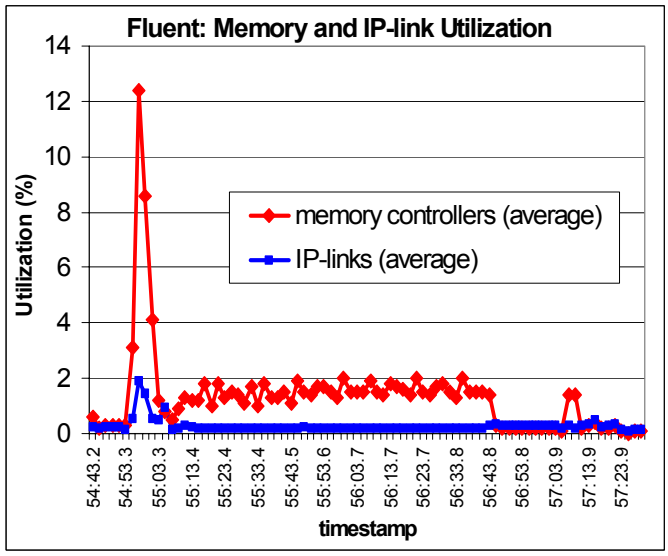

Figure 20. Memory and IP-links utilization in Fluent.

5.2. Memory-Bandwidth Intensive application: NAS Parallel

NAS Parallel benchmarks represent a collection of kernels that are important in many technical applications [14]. The kernels are decomposed using MPI and can run on either shared-memory or cluster systems. With the exception of EP (embarrassingly parallel), majority of these kernels (solvers, FFT, grid, integer sort) put significant stress on memory bandwidth (when size C is used). Note that although these are small kernels, they provide the same level of stress on memory subsystem as many large real applications.

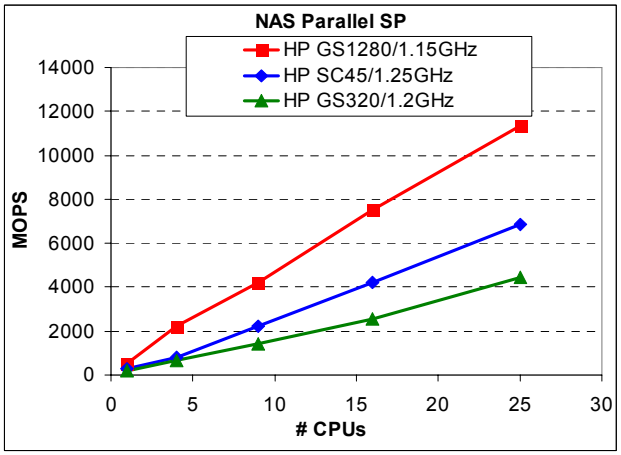

Figure 21. SP Performance comparison.

Figure 21 compares GS1280 performance to the previous-generation Alpha platforms in the SP solver. This data shows substantial advantage on GS1280 compared to the other systems.

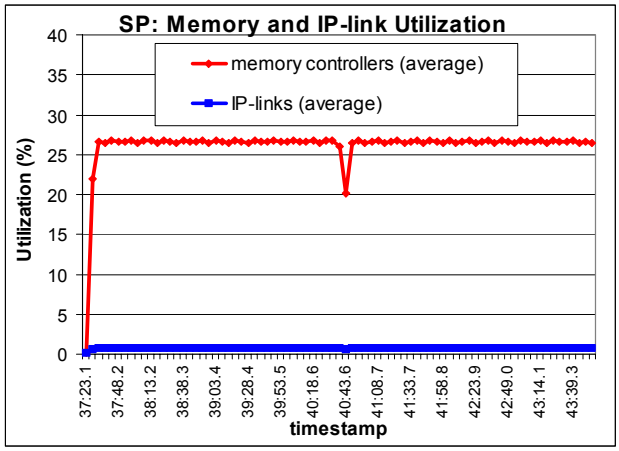

Figure 22. Memory Controller utilization in SP

In order to explain this advantage, we show the memory and IP-link utilization in Figure 22. Figure 22 shows that memory bandwidth utilization is high in SP (26%). GS1280 has substantially higher memory bandwidth than ES45 and GS320 (Figures 6 and 7), thus the advantage in SP. Since ES45 has higher memory bandwidth than GS320 (Figure 7), GS1280 shows even higher advantage vs. GS320 than EC45. The IP links utilization in these MPI kernels is low (Figure 22). We also observed that IP link utilization is low in many other MPI applications. The GS1280 provides very high IP-link bandwidth that in many cases exceeds the needs of MPI applications (many of which are designed for cluster interconnects with much lower bandwidth requirements).

5.3. IP bandwidth intensive application: GUPS

GUPS is a multithreaded (OpenMP) application where each thread updates an item randomly picked from the large table [15]. Since the table is so large that it spans the entire memory in the system, this application puts substantial stress on the IP-link bandwidth (Figure 24). In this application, GS1280 shows the most substantial advantage over the other systems, as shown in Figure 23. This is because this application exploits substantial IP-link bandwidth advantage on GS1280, as discussed in Section 4 (Figure 15). It is also interesting that the links show uneven utilization in Figure 24: East/West links show higher utilization than North/South links. This is because the link utilization is higher on horizontal than on vertical links in a 4x8 torus. This is also the reason for the bend in performance at 32 CPUs: the cross-sectional bandwidth is comparable in both 16P and 32P torus configurations.

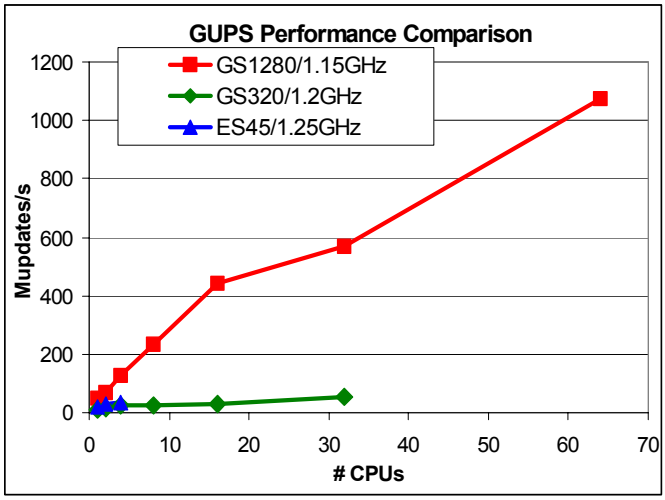

Figure 23. GUPS Performance comparison.

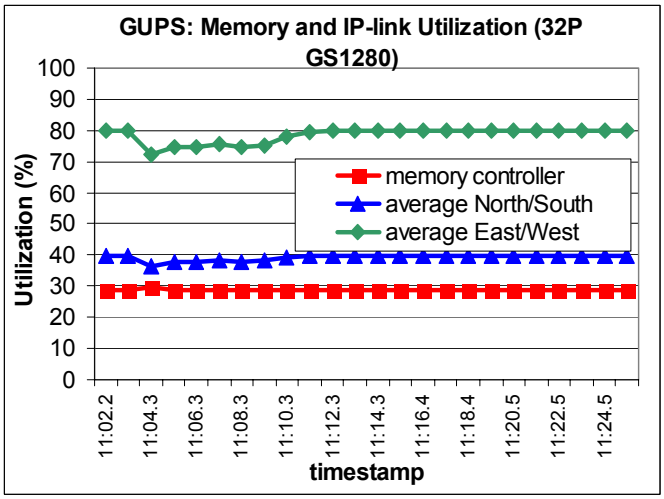

Figure 24. Memory and IP-link utilization in GUPS.

6. Memory Striping

Memory striping allows interleaving of 4 cache lines across two CPUs, starting with CPU0/controller0, then CPU0/controller1, and then CPU1/controller0, and finally CPU1/controller1. The CPUs chosen to participate in striping are the closest neighbors (CPUs on the same module). Striping provides performance benefit in alleviating hot spots, where a hot-spot traffic is spread across 2 CPUs (instead of one). The disadvantage of memory striping is that it puts additional burden on the IP links between pairs of CPUs.

The results of our evaluation of memory striping are presented in Figures 25 and 26. Figure 25 shows that striping degrades performance 10-30% in throughput applications due to increased inter-processor traffic. We observed degradation as high as 70% in some applications.

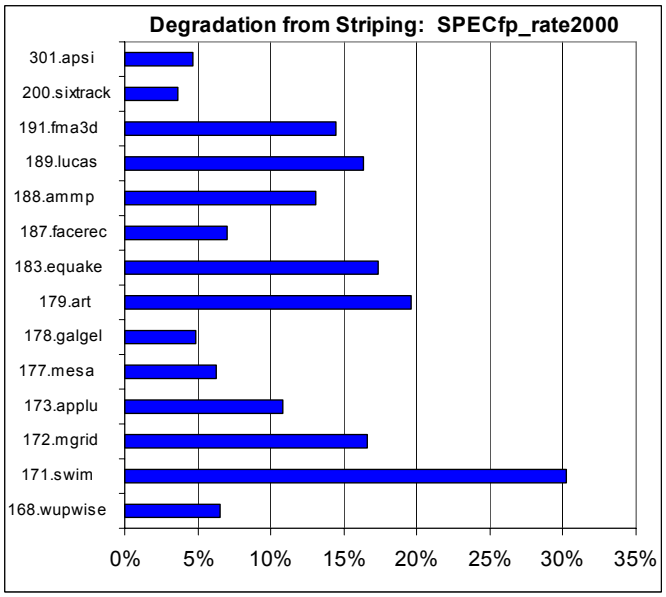

Figure 25. Degradation from striping.

Figure 26 shows that striping improves performance of a hot-spot traffic pattern (all CPUs read data from CPU0) up to 80%. We use the Xmesh tool based on built-in performance counters to recognize the hot-spot traffic (Figure 27). The tool indicates that the IP-link and memory traffic on the links to/from CPU0 (left corner) is higher than on any other CPU, and that Zbox utilization on that CPU is 53% (much higher than on any other CPU). We observed 30% improvement in real applications that generate hot-spot traffic.

A more extensive study over a variety of applications indicated that only a small portion of applications benefit from striping (while most others degrade performance).

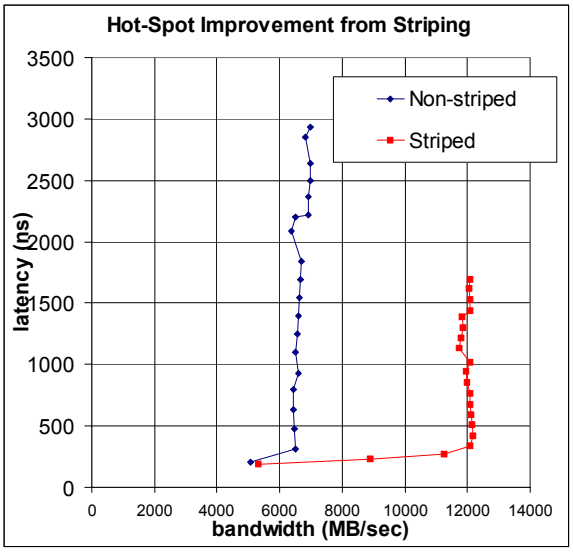

Figure 26. Improvement from striping.

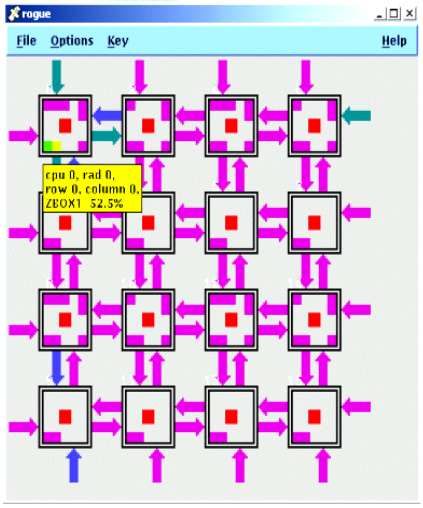

Figure 27. Xmesh with a hot-spot.

7. Summary Comparisons

In Section 5 we analyzed performance of representatives of three application classes. In this section, we compare GS1280 to GS320 across a wider range of applications (Figure 28). The data in Figure 28 is shown as the ratio of GS1280 improvement vs. GS320. The data is grouped in the following categories: system components (CPU, memory, Inter-Processor, I/O), integer and commercial benchmarks (SPECint_rate2000, SAP Transaction Processing, Decision Support [8][17]), HPTC standard benchmarks (SPECfp_rate2000, NAS Parallel, and SPEComp2001 [8][14]), HPTC applications (CFD, chemistry, weather prediction, structural modeling) [13][18][19][20][21], and two cases where GS1280 shows the highest improvement (GUPS and swim) [8][15].

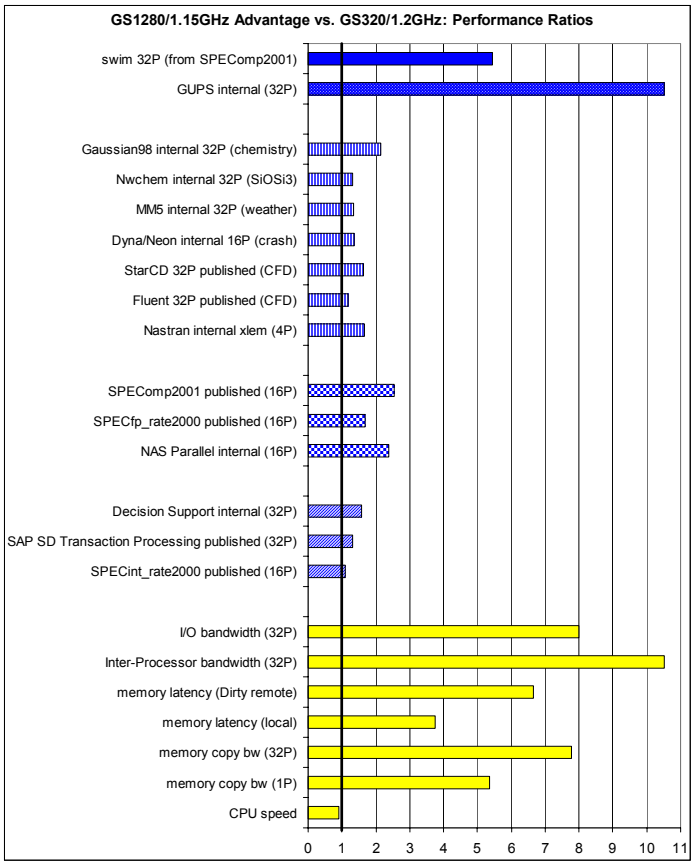

Figure 28. GS1280 vs. GS320 summary comparisons.

The data in Figure 28 indicates that GS1280 shows the most significant improvement in IP bandwidth (over 10 times), and I/O and memory bandwidth (8 times). The application comparisons show that although GS1280 and GS320 have comparable processor clock speeds, the majority of applications run faster on GS1280 than GS320. The exceptions are the small integer benchmarks (SPECint2000) that fit well in the on-chip caches. The commercial workloads show between 1.3-1.6 times advantage on GS1280 vs. GS320. The standard HPTC benchmarks (SPEC and NAS Parallel) show between 1.7-2.6 times gain mainly due to memory-bandwidth advantage on GS1280. The GS1280 advantage in ISV applications ranges from 1.2-2.1 times. The swim and GUPS applications benefit from memory and IP-link bandwidth advantage in GS1280.

This data indicates that the designs of memory interface, I/O subsystem, and interprocessor interconnect have profound effect on application performance. Often, the emphasis is placed on processor design, and other system components take lower priority. Our study indicates that the key factor for achieving high application performance is the balanced design that includes not only a high-performance processor, but also matching high-performance designs of memory, interconnect, and I/O subsystems.

8. Conclusions

We evaluated the architecture and performance characteristics of the HP AlphaServer GS1280, based on the Alpha 21364 processor. The Alpha 21364 shows substantial departure in processor design compared to the previous-generation Alpha processors. It incorporates (1) on-ship cache (smaller than the off-chip cache in the previous-generation 21264), (2) two memory controllers that provide exceptional memory bandwidth, and (3) a router that allows efficient glue-less large-scale multiprocessor design. The 21364 processor places on a single chip all components that previously required an entire CPU module.

The results from our analysis show that this is a superior design for building large-scale multiprocessors. The exceptional memory bandwidth that GS1280 provides is important for a number of applications that cannot be structured to allow for cache reuse. We observed 2-4 times advantage of GS1280 vs. the previous-generation AlphaServer GS320 in this type of applications (e.g. NAS Parallel). The low latency and exceptional bandwidth on IP links allow for very good scaling in applications that cannot be blocked to fit in the local memory of each processor. We observed even higher advantage of GS1280 vs. the previous-generation AlphaServer GS320 in this type of applications (e.g. over 10 times in GUPS). Since Alpha 21364 preserved the same core as the previous-generation Alpha 21264 (and the CPU clock speeds are comparable), the applications that are blocked to fit well in the on-chip caches perform comparably on GS1280 and GS320 (e.g. SPECint2000). Some applications take advantage of the large 16MB cache, and therefore run faster on GS320 than on GS1280 (e.g. facerec from SPECfp2000). However, most applications benefit from the GS1280 design, indicating that the architecture of memory interface, interprocessor interconnect, and I/O subsystem is as important as the processor design.

We proposed a simple change in routing (called shuffle) that provides substantial performance improvements on

an 8-CPU torus interconnect. We also determined that striping memory across two processors is beneficial only in applications that generate a hot-spot traffic, while it was detrimental for majority of applications due to increased nearest-neighbor bandwidth.

We have heavily relied on profiling analysis based on the built-in performance counters (Xmesh) throughout this study. Such tools are crucial for understanding system behavior. We have used profiles to explain why some workloads perform exceptionally well on GS1280, while others show comparable (or even worse) performance than GS320 and ES45. In addition, these tools are crucial for identifying areas for improving performance on GS1280: e.g. Xmesh can detect hot-spots, heavy traffic on the IP links (indicate poor memory locality), etc. Once such bottlenecks are recognized, various techniques can be used to improve performance.

The GS1280 system is the last-generation Alpha server. In our future work, we plan to extend our analysis to non-Alpha based large-scale multiprocessor platforms. We will also place more emphasis on characterizing real I/O intensive applications.

Acknowledgments

The author would like to thank to Jason Campoli, Peter Gilbert, and Andrew Feld for profiling data collection. Special thanks to Darrel Donaldson for his guidance and help with the GS1280 system and to Steve Jenkins, Sas Durvasula, and Jack Zemcik for supporting this work.

References

[1] Peter Bannon, "EV7", Microprocessor Forum, Oct. 2001.

[2] K. Gharachorloo, M. Sharma, S.Steely, S. Van Doren, "Architecture and Design and AlphaServer GS320", ASPLOS 2000.

[3] DCPI and ProfileMe External Release page: http://www.research.digital.com/SRC/dcpi/release.html

[4] Z. Cvetanovic and R. Kessler, "Performance Analysis of the Alpha 21264-based Compaq ES40 System", *The 27th Annual International Symposium on Computer Architecture,* June 10-14, 2000, pp. 60-70.

[5] Z. Cvetanovic and D. Bhandarkar, "Characterization of Alpha AXP Performance Using TP and SPEC Workloads", *The 21st Annual International Symposium on Computer Architecture*, April 1994, pp. 60 - 70.

[6] Z. Cvetanovic and D. Bhandarkar, "Performance Characterization of the Alpha 21164 Microprocessor Using TP and SPEC Workloads," *The Second International Symposium on High-Performance Computer Architecture* (February 1996), pp. 270–280.

[7] Z. Cvetanovic and D. D. Donaldson, "AlphaServer 4100 Performance Characterization", *Digital Technical Journal, Vol 8 No. 4, 1996:* pp. 3-20.

[8] SPEC CPU2000 Benchmarks available at http://www.spec.org/osg/cpu2000/results
SPEC® and SPEC CPU2000® are registered trademarks of the Standard Performance Evaluation Corporation.

[9] Information about the lmbench available at http://www.bitmover.com/lmbench/

[10] The STREAM benchmark information available at http://www.cs.viginia.edu/stream

[11] Z. Cvetanovic, P. Gilbert, J. Campoli, "Xmesh: a graphical performance monitoring tool for large-scale multiprocessors", HP internal report, 2002.

[12] J. Duato, S Yalamanchili, L. Ni, "*Interconnection Networks, an Engineering Approach*", IEEE Computer Society, Los Alamitos, California

[13] Fluent data available at http://www.fluent.com/software/fluent/fl5bench/fullres.htm

[14] NAS Parallel data available at http://www.nas.nasa.gov/Software/NPB/

[15] GUPS information available at http://iram.cs.berkeley.edu/~brg/dis/gups/

[16] "EV7 System Design Specification", Internal HP document, August 2002.

[17] SAP Benchmark results available at http://www.sap.com/benchmark/index.asp?content=http://www.sap.com/benchmark/sd2tier.asp

[18] StarCD data available at http://www.cd-adapco.com/support/bench/315/aclass.htm

[19] LS-Dyna information available at http://www.arup.com/dyna/applications/crash/crash.htm

[20] NWchem data available at http://www.emsl.pnl.gov:2080/docs/nwchem/nwchem.html

[21] MM5 data available at http://www.mmm.ucar.edu/mm5/mm5-home.html