PERFORMANCE ANALYSIS OF ALPHASERVER GS1280
White Paper

Zarka Cvetanovic
Contributions by: Characterization, Marvel, EV7, Tru64, Performance, and HPTC teams
AlphaServer Product Development, Hewlett-Packard Company
January 2003

Abstract: *This paper evaluates performance characteristics of the HP AlphaServer GS1280 shared-memory multiprocessor system. The GS1280 system contains up to 64 Alpha 21364 (EV7) 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), as well as other-vendor systems (IBM, SUN, SGI). We characterize GS1280 performance using synthetic workloads that stress various system components (processor, memory, Interprocessor (IP) links, and I/O. We then expand the analysis to the real workloads, including standard benchmarks, ISV/HPTC and commercial applications, and customer benchmarks. We characterize these applications using built-in performance counter tools (Xmesh). Based on the data, we provide guidelines (software and hardware/configuration) for improving GS1280 performance.*

1. **INTRODUCTION**
The HP AlphaServer GS1280 (codename Marvel) is a shared memory multiprocessor system containing up to 64 fourth-generation Alpha 21364 (EV7) microprocessors [1].

The GS1280 system contains many architectural advances – both in the microprocessor and in the surrounding memory system - that contribute to its performance. The EV7 processor [1][16] uses the same core as the previous-generation 21264 (EV6) processor. However, EV7 includes three additional components: (1) an on-chip L2 cache, (2) two on-chip Direct Rambus (RDRAM) memory controllers and (3) an router. The combination of these components helped achieve improved access time to the L2 cache, low memory latency (local and remote), exceptional memory bandwidth, and very high interconnect bandwidth. These improvements enhance single-CPU performance and contribute to excellent SMP 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 the 21264 (EV6) processor: GS320 – a 32-CPU SMP NUMA system with switch-based interconnect [2], and SC45 – 4-CPU ES45 systems connected in a cluster configuration via a fast Quadrics switch. [4][5][6][7].

We include results from kernels that exercise the memory subsystem [9][10]. We compare GS1280 to the previous-generation Alpha and other-vendor platforms in standard benchmarks (SPEC2000, SPEComp2001, Linpack NXN, STREAMS, Transaction-Processing, Java Business application, etc. [8][10]). We further expand the analysis to other large ISV/HPTC applications (Structural Modeling, CFD, Crash simulation, Material/Life Sciences) [13][18]-[27]. We identify applications that show significant advantage on GS1280 vs. other systems ("killer" applications). We demonstrate how to use tools based on the EV7 and IO7 specific performance counters (Xmesh, Profileme [11][3]) to characterize applications and predict their performance on GS1280. We describe Xmesh, a graphical tool that provides a run-time display of utilization of CPUs, memory controllers, inter-processor (IP) links, and I/O ports. We provide guidelines on how to optimize application performance using software techniques. Finally, we discuss how to configure GS1280 for best performance. Note that very little GS1280 performance data is published at this time. We will keep this information updated as published data becomes available.

The remainder of this paper is organized as follows: Section 2 describes the architecture of the GS1280 system. Section 3 characterizes the memory, interconnect, and I/O system improvements. Section 4 compares application performance of GS1280 to the other systems (for standard benchmarks, ISV/HPTC applications, and customer benchmarks). In this section we also identify applications that perform exceptionally well on GS1280 ("killer apps"). Section 5 describes tools for analyzing GS1280 performance (Xmesh). Section 6 discusses methods for improving GS1280 performance. Section 7 concludes.

2. GS1280 SYSTEM OVERVIEW

The Alpha 21364 (EV7) microprocessor [1] shown in Figures 1, 1a, and 1b integrates the following components on a single chip: (1) second-level (L2) cache, (2) router, and (3) two memory controllers (Zboxes), and (4) a 21264 (EV68) microprocessor core. The current GS1280 product goal for 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 (Zboxes) connect the processor directly to the RDRAM memory (Figure 1a). 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 IO7 is connected to the EV7 via a full-duplex link capable of 3.1 GB/s. Each IO7 supports one AGP, one PCI, and two PCI-X ports. The dual CPU building block consists of 2 EV7 processors, memory cards, power regulators and a CPU management card (Figure 1b).

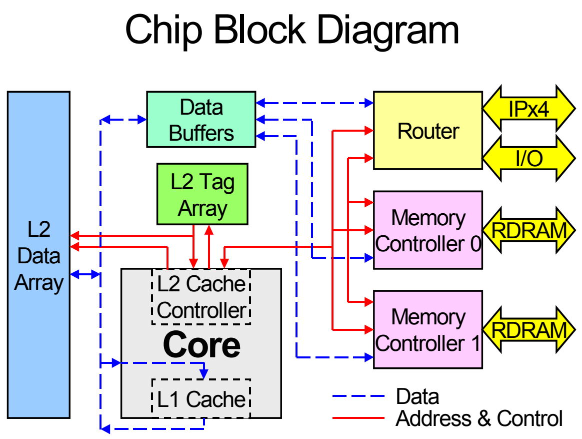


Figure 1. EV7 (21364) block diagram.

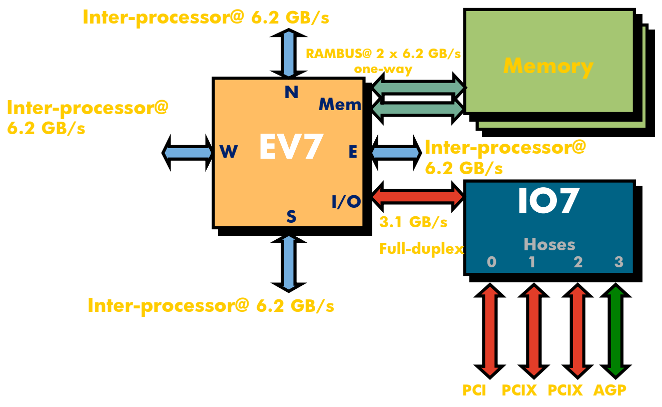

Inter-processor@ 6.2 GB/s
RAMBUS@ 2 x 6.2 GB/s one-way
Memory
Inter-processor@ 6.2 GB/s
N
Mem
W
EV7
E
I/O
S
Inter-processor@ 6.2 GB/s
3.1 GB/s Full-duplex
IO7
Hoses
0 1 2 3
Inter-processor@ 6.2 GB/s
PCI PCIX PCIX AGP

Figure 1a. EV7 interface to memory and I/O.

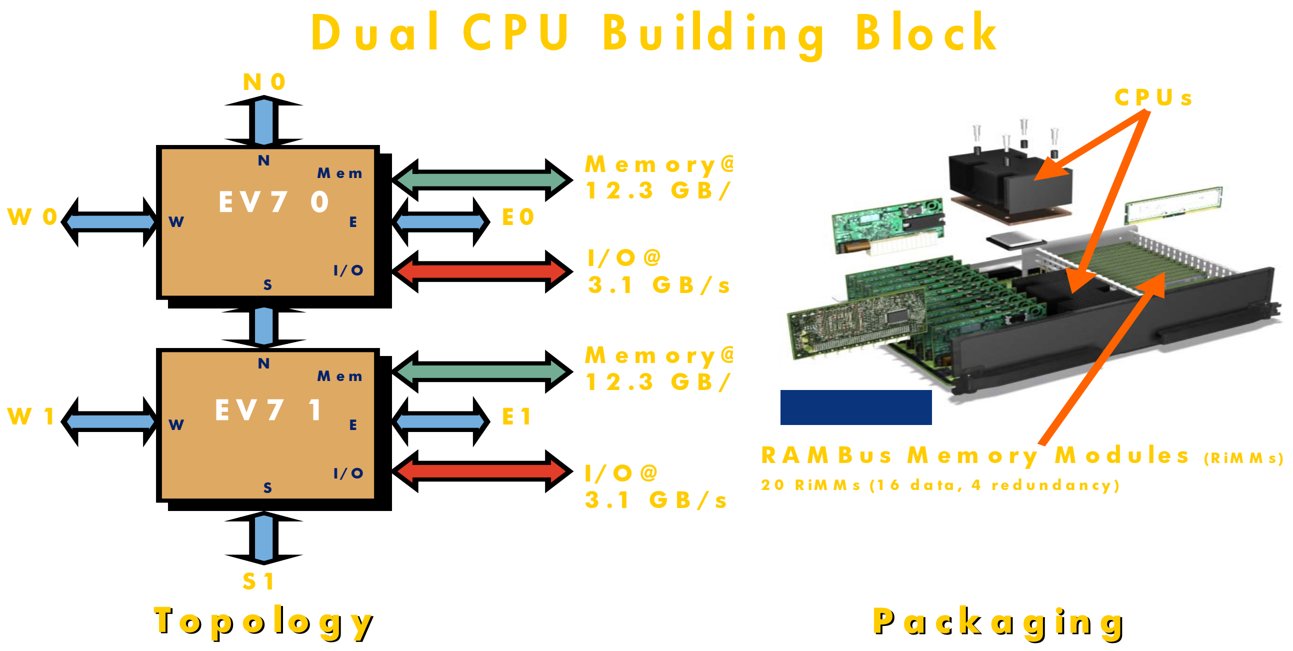


Figure 1b. Dual CPU building block.

The router [16] connects multiple 21364s in a two-dimensional, adaptive, torus network. Examples of the interconnect for 16 and 32 CPU GS1280 are shown in Figures 2a and 2b. The actual cable connections are shown on the left hand side of each figure.

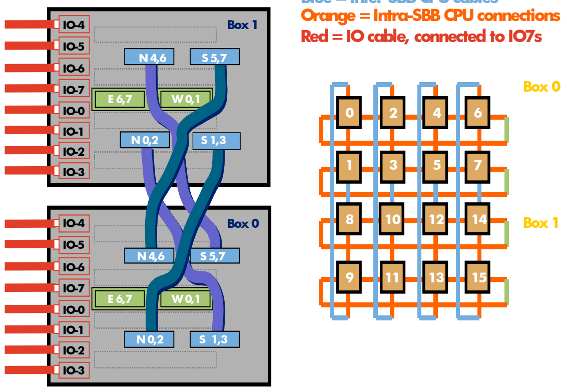

Green = Loopback Plug-in Module
Blue = Inter-SBB CPU cables
Orange = Intra-SBB CPU connections
Red = IO cable, connected to IO7s
Box 1
Box 0
IO-4
IO-5
IO-6
IO-7
IO-0
IO-1
IO-2
IO-3
N 4,6
S 5,7
E 6,7
W 0,1
N 0,2
S 1,3
0
2
4
6
1
3
5
7
8
10
12
14
9
11
13
15

Figure 2a. 16-CPU Interconnect.

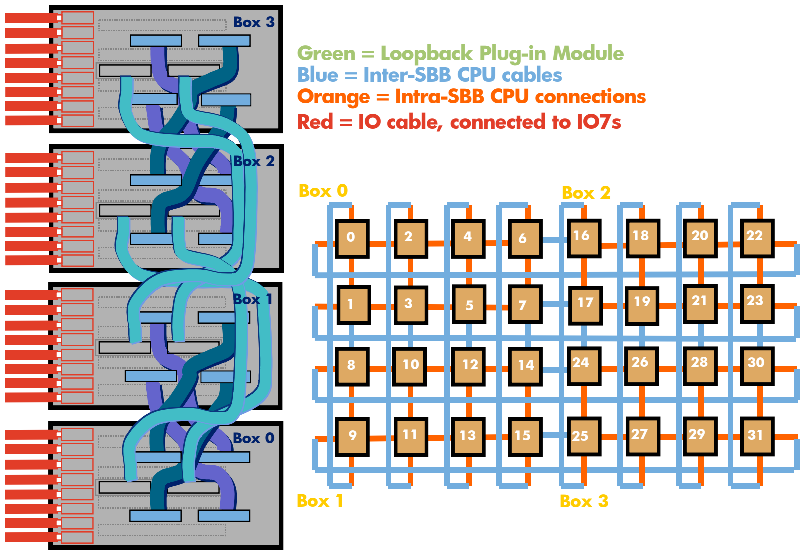

Box 3
Box 2
Box 1
Box 0
Green = Loopback Plug-in Module
Blue = Inter-SBB CPU cables
Orange = Intra-SBB CPU connections
Red = IO cable, connected to IO7s
Box 0
Box 2
Box 1
Box 3
0
2
4
6
16
18
20
22
1
3
5
7
17
19
21
23
8
10
12
14
24
26
28
30
9
11
13
15
25
27
29
31

Figure 2b. 32-CPU Interconnect.

The router connects to 4 links that connect to 4 neighbors in the torus: North, South, East, and West (2 uni-directional links each). 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 the 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 scheme. 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. The dependence ordering is as follows: Read I/O, Write I/O, Requests, Forwards, Invalidation Broadcast, Acknowledgments, and Block Responses. Thus, 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 resolved by allowing messages 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 AlphaServer 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 system (up to 32 CPUs) [2]. The design and characterization of the ES45 system are described in [4].

3. Memory, Interconnect, and I/O Subsystem

In this section we characterize memory, interprocessor, and I/O subsystems of GS1280 and compare them to the previous-generation Alpha platforms.

3.1 Memory Latency for dependent loads

The 21364 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 the previous-generation AlphaServers (ES45 and GS320) is 16MB, off-chip, direct-mapped.

Figure 3 shows “dependent-load” latency. The “dependent-load latency” [11] measures load-to-use latency where each load depends on the result from the previous load. The lower axis in Figure 3 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 3 show that GS1280 has 3.8 times lower “dependent-load” memory latency (32M size) vs. the previous-generation GS320.This data indicates that large-size applications which are not blocked to take advantage of large 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 application sizes that fall in this range are likely to run slower on GS1280 than on the previous-generation platforms. For data range between 64KB and 1.75MB, latency is again much lower on GS1280 than GS320/ES45. That’s because the L2 cache in GS1280 is on-chip, thus providing much lower access time than the off-chip caches in GS320/ES45. Figure 3 shows that GS1280 provides the lowest memory latency of all systems compared (more than 3 times lower than IBM/Power4). Figure 4 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).

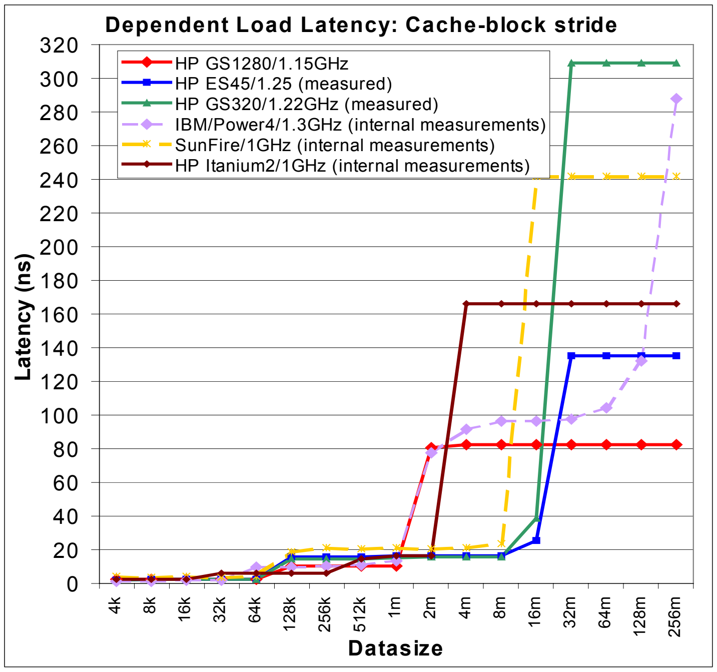


Figure 3. Dependent load latency comparison.

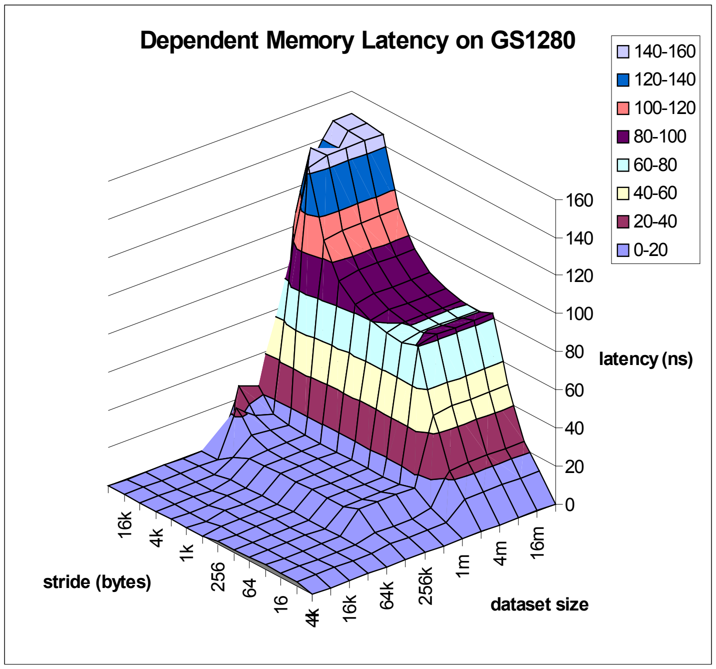


Figure 4. Dependent load latency on GS1280 for various strides.

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 (Triad) [10]. We show only the results for the Triad kernel (the other kernels have similar results). Figure 5 compares memory bandwidth of the STREAM Triad loop across Alpha servers as well as other leading-vendor systems [10]. This data shows that the memory bandwidth on GS1280 is substantially (7 times) higher than the previous-generation GS320. In addition, GS1280 shows substantial advantage in memory bandwidth over all other systems shown (5 times advantage over IBM/Power4).

Figure 6 shows that GS1280 exhibits not only 1-CPU advantage in memory bandwidth (due to high-bandwidth RDRAM memory provided by the EV7 processor), it also provides linear scaling in bandwidth as the number of CPUs increases. The reason is because each CPU in GS1280 system 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, where 4 CPUs contend for the same memory, and GS320 where all 4 CPUs within the same QBB share the same memory. Therefore, bandwidth improvement from 1 to 4 CPUs on ES45/GS320 is less-than-linear (as indicated in Figure 6). Note that bandwidth scaling on IBM/Power4 is less-than-linear as well (Figure 5).

The data in Figures 5 and 6 indicate that the applications that stress memory bandwidth will run exceptionally well on GS1280. The advantage is likely to increase as the number of CPUs grows.

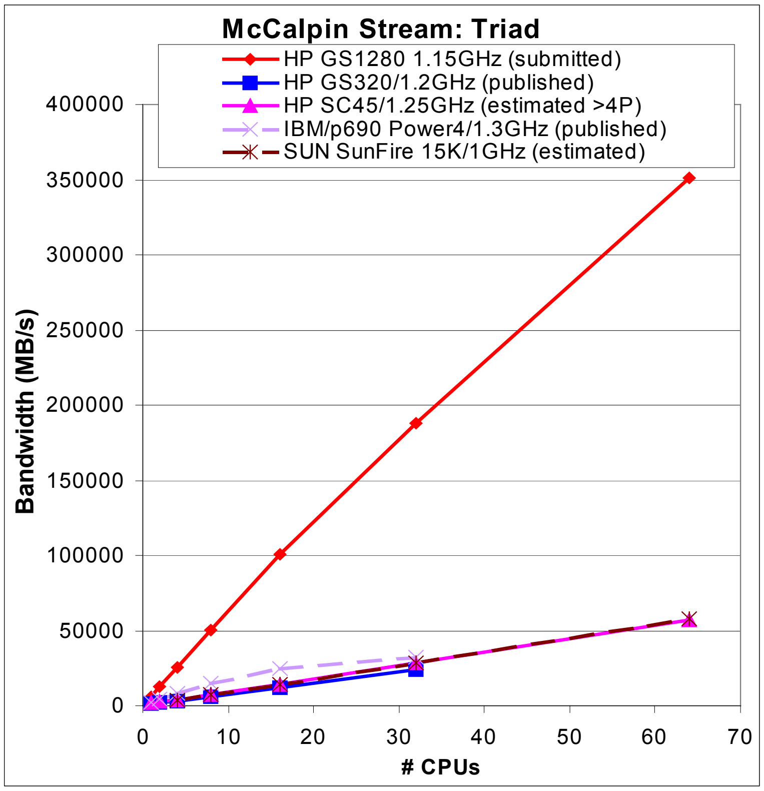


Figure 5. McCalpin STREAM bandwidth comparison.

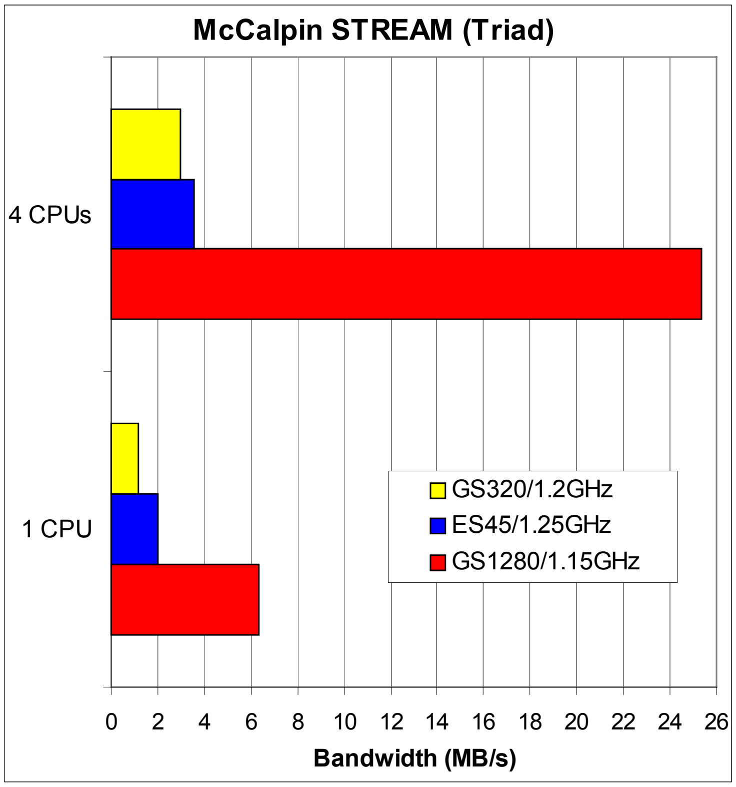


Figure 6. STREAM bandwidth for 1-4 CPUs.

3.3 1-CPU performance: SPEC2000

Figures 7 and 8 compare Instructions-per-Cycle (IPC) for floating-point (fp) and integer CPU2000 benchmarks on GS1280 vs. GS320 and ES45. Note that IPC is proportional to performance, except that clock differences are eliminated. IPC-based comparisons are done in most architectural studies to isolate the performance effects of architecture design.

On average, GS1280 shows advantage over both GS320 and ES45 in SPECfp2000, and comparable performance in SPECint2000. This is because integer benchmarks fit well in the on-chip caches, while several fp benchmarks stress memory bandwidth (Figures 7 and 8). GS1280 has significant advantage in memory bandwidth (as illustrated in Figure 6). The IPC in integer benchmarks is comparable on all 3 systems (primarily determined by the clock speed).

Note that some benchmarks demonstrate a substantial advantage on GS1280 over ES45/GS320 (e.g. swim: 2.3 times vs. ES45 and 4 times vs. GS320). However, many others show comparable performance (most integer benchmarks). Yet, there are cases where GS320 and ES45 outperform GS1280 (e.g. facerec). In order to better understand what causes such differences, we generated profiles that show memory controller utilization for all benchmarks (Figures 9 and 10).

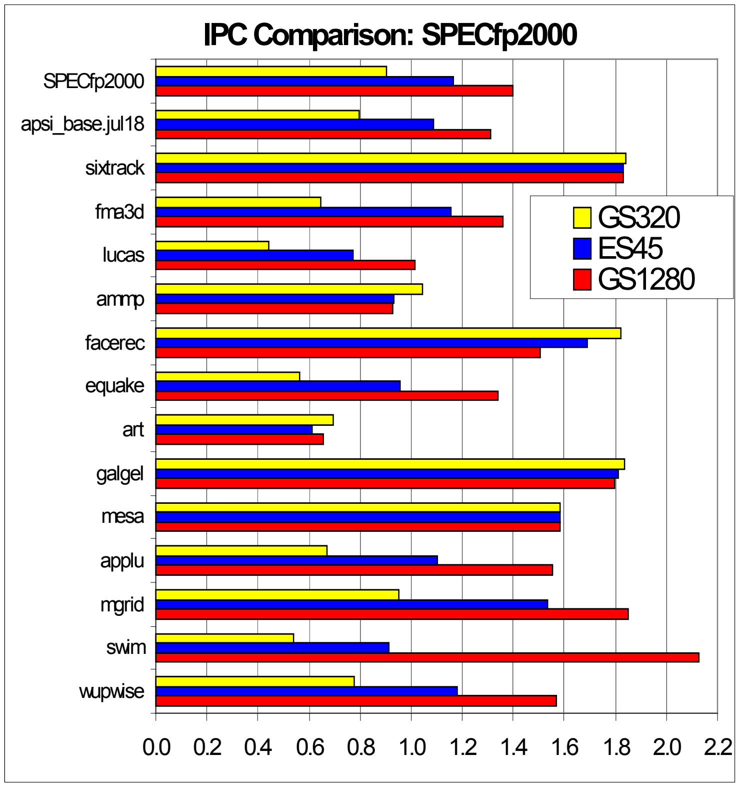


Figure 7. IPC for SPECfp2000.

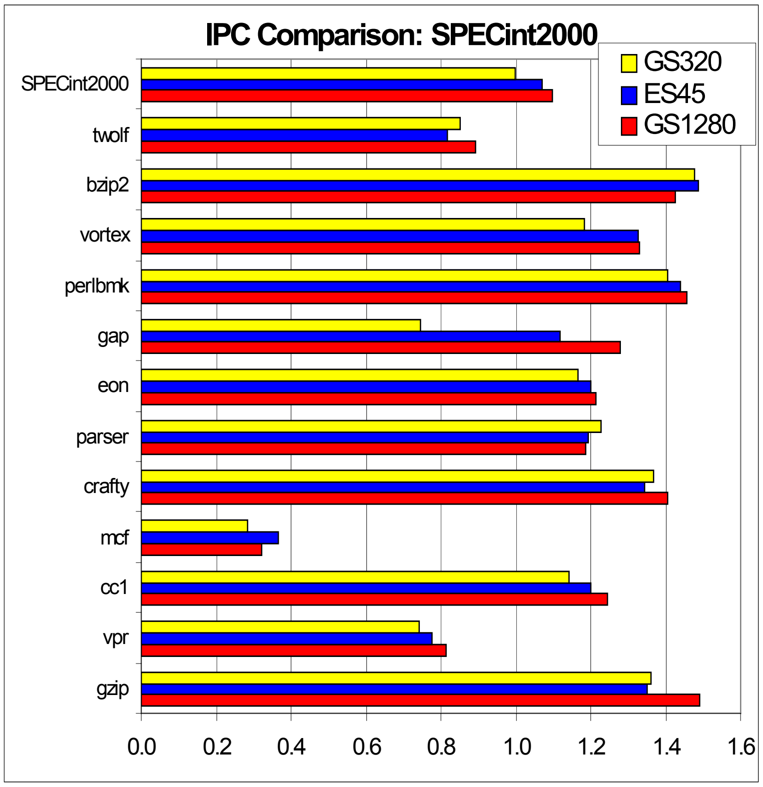


Figure 8. IPC for SPECint2000.

Figures 9 and 10 illustrate memory controller utilization profiling histograms for SPEC2000 benchmarks on GS1280. The profiles are collected using the Xmesh tool that is based on the EV7 built-in performance counters (detailed description of Xmesh provided in Section 5). The histograms are shown as a function of 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 55% 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, it has to access memory on GS1280, whereas it fetches data mostly from the 16MB cache on GS320 and ES45. Figure 3 illustrates that the cache access on GS320 is faster than the memory access on GS1280. Similarly, mcf has high memory utilization, however it does not show advantage on GS1280. That's because cache misses on mcf are significantly lower with the 16MB cache than with the 1.75MB cache.

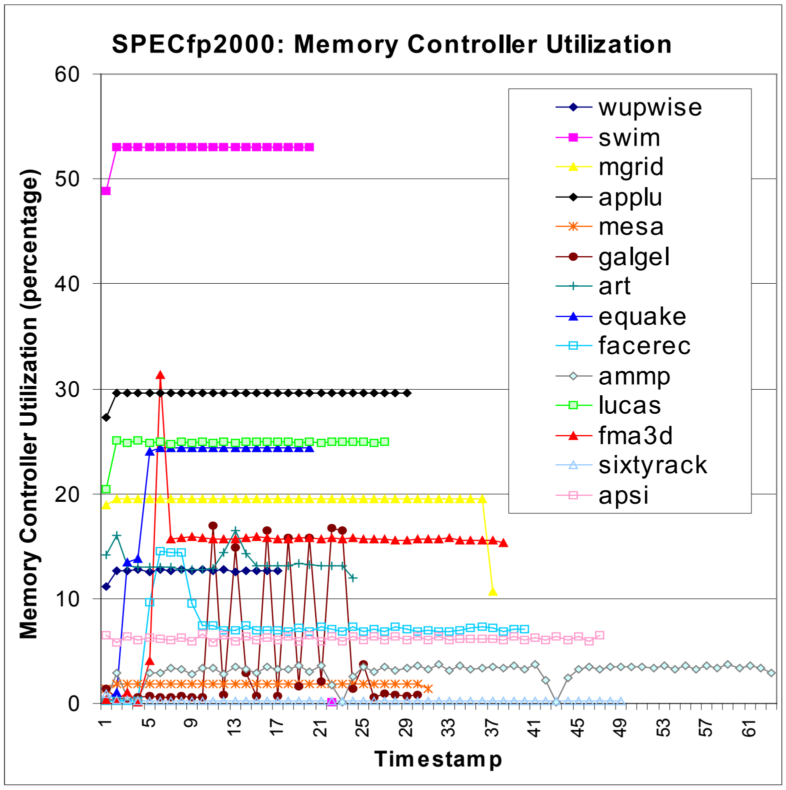


Figure 9. GS1280 memory controller utilization in SPECfp2000.

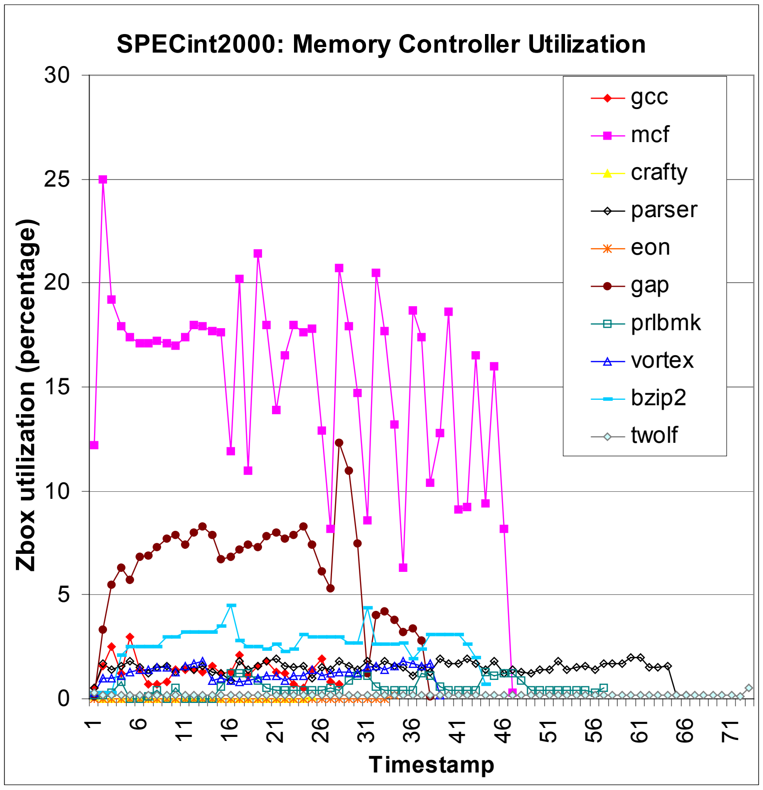


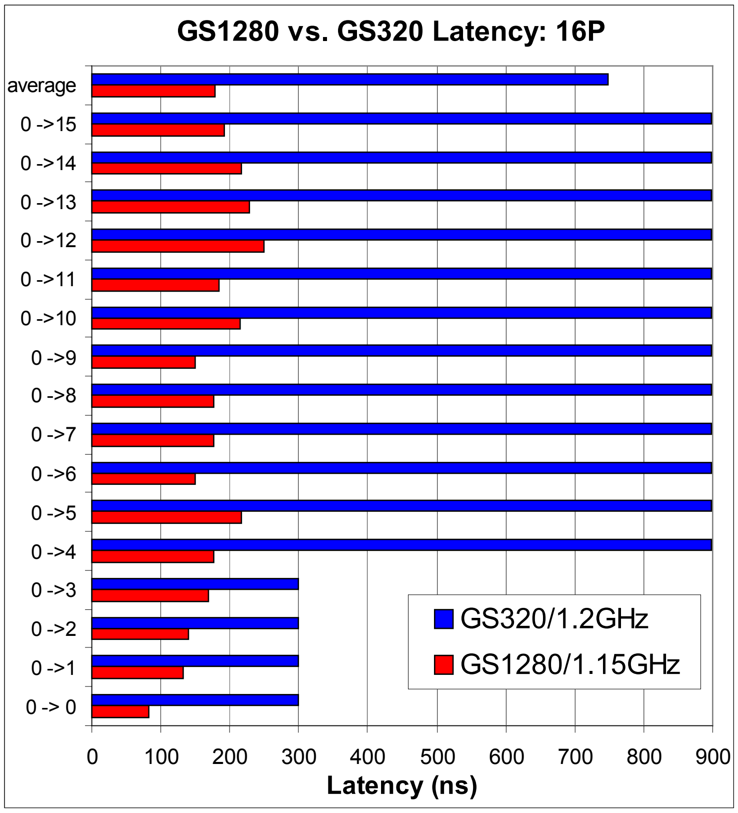
Figure 10. GS1280 memory controller utilization in SPECint2000.

3.4. Remote memory latency

In Sections 3.1 and 3.2 we contrasted local memory latencies and bandwidths on GS1280 with previous-generation Alpha platforms. Local memory characteristics are important for the single-CPU workloads and multiprocessor workloads that fit well in 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 11 compares local and remote memory latency on GS320 and GS1280 for 16 CPUs. Latency is measured from CPU0 to all other CPUs in a 16-CPU system.

Figure 11 indicates that GS320 has 2 levels of latency: local (within a set of 4 CPUs called QBB) and remote (outside the QBB). The GS1280 system has many levels of remote latency, depending on how many hops need to be traveled from source to destination. Note that GS1280 shows substantial improvement in not only local, but also remote memory latency. The average remote latency is on GS1280 is 4 times lower than on GS320 (with 16 CPUs). The worst-case memory latency on 16-CPU GS1280 (4-hop latency from node 0 to 12) is lower that the local memory latency on GS320.

Figure 12 shows local and remote Read Dirty latency. In this case, a cache block is read from another processor's cache (instead of memory), since it was written by another processor (thus Dirty). The latency data indicates that GS1280 shows even higher advantage over the other platforms in dirty (Figure 12) than clean (Figure 11) latency. The advantage is increased from 4 times (Figure 11) to 6.6 times (Figure 12). The average 16-CPU dirty latency on GS1280 is lower than the average 4-CPU dirty latency on ES45 (Figure 12). In fact, GS1280 is the first Alpha system where dirty latency is comparable (or lower) than the clean latency. This feature is very important in applications that require a lot of data sharing (many commercial and parallel/multithreaded applications belong to this class).



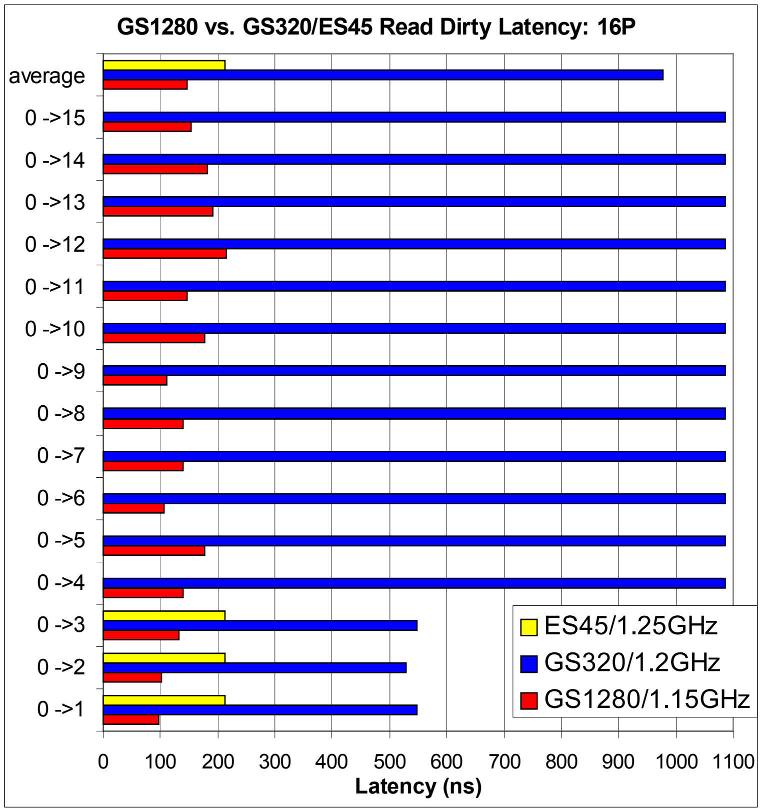


Figure 11. Local/remote clean memory latency on 16 CPUs. Figure 12. Local/remote dirty memory latency on 16 CPUs.

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 82ns is increased to 133-150ns for the 1-hop neighbors (4 of them). Note that the 1-hop latency is the lowest for the neighbors on the same module (133ns), and the highest for the neighbors that are connected via a cable (150ns). The 2-hop latency is between 169 and 192ns (6 nodes are 2-hop away). The 3-hop latency is between 215ns and 228ns (4 nodes). The 4-hop latency (worst-case for 16 CPUs) is 250 ns (1 node is 4-hops away).

The average latencies measured for different numbers of CPUs are compared in Figure 14. This figure illustrates that GS1280 has a significant advantage over GS320 not only in local, but also in average remote memory latency. The advantage is 2 times on a 4-CPU system and closer to 4 times on a 32-CPU system, as indicated in Figure 14. Even the 64-CPU GS1280 system has close to 3 times lower average remote latency than the 32-CPU GS320. The 4-CPU GS1280 shows comparable average memory latency to ES45.

The GS1280 advantage in remote memory latency indicates that applications which are not structured to fit well within a processor's local memory will run much more efficiently on GS1280 than on GS320.

82	141	177	149
133	169	216	176
176	215	250	216
150	184	228	192

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

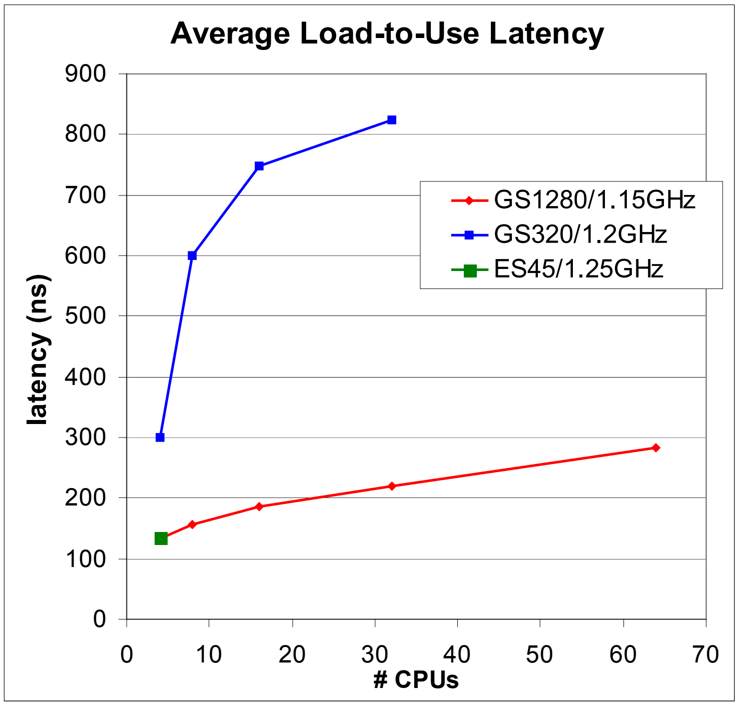


Figure 14. Average load-to-use latency comparison.

3.5 Interprocessor Bandwidth

The memory latency in Figure 14 is measured on an idle system with only 2 CPUs active (the reader CPU sends a Read Request to another CPU that provides a block by sending a Block Response). In this section, we evaluate performance of the loaded system. This is needed in order to characterize applications that require all CPUs to exchange messages simultaneously (more closely related to the real application environment).

Figure 15 compares bandwidth under load on GS1280 to ES45/GS320. Each CPU randomly selects another CPU to send a Read request to. The test is started with a single outstanding load (leftmost point on the graph). For each additional point, one outstanding request is added (up to 20 outstanding requests). In ideal case, as the number of outstanding requests increases, the bandwidth will increase too (moving to the right), and latency will not change (line stays low and flat). Figure 14 shows that GS1280 has substantial performance advantage over the other systems as the system load is increased. Although there is an increase in latency on GS1280, it is not nearly as high as in previous-generation platforms (ES45 and GS320). This indicates that 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 (e.g. simultaneous processes that access memory randomly). The GS1280 is much better suited for applications that are not structured well for NUMA architectures (poor data locality).

Figure 16 compares GS1280 to SC45 and GS320 (16 CPUs) using the Pallas Exchange test. The Pallas set of benchmarks [22] is used to evaluate MPI performance (which is important in many MPI applications). The set of tests includes Single-transfer tests (PingPong), Parallel-transfer tests (Exchange), and Collective tests (Reduce, Alltoall). The comparison in Figure 16 indicates substantial (3-4 times) bandwidth advantage of GS1280 vs. GS320 and ES45 in the Exchange test (for larger buffer sizes). Note that the advantage vs. SC45 comes mainly from the faster inter-processor interconnect on GS1280 vs. the Quadrics switch on SC45. This indicates that up-to 64 CPUs, GS1280 takes advantage of the faster interconnect and will outperform 64-CPU SC45. However, for larger-than 64-CPU configurations, SC1280 will start to be affected by the lower bandwidth on Quadrics interconnect.

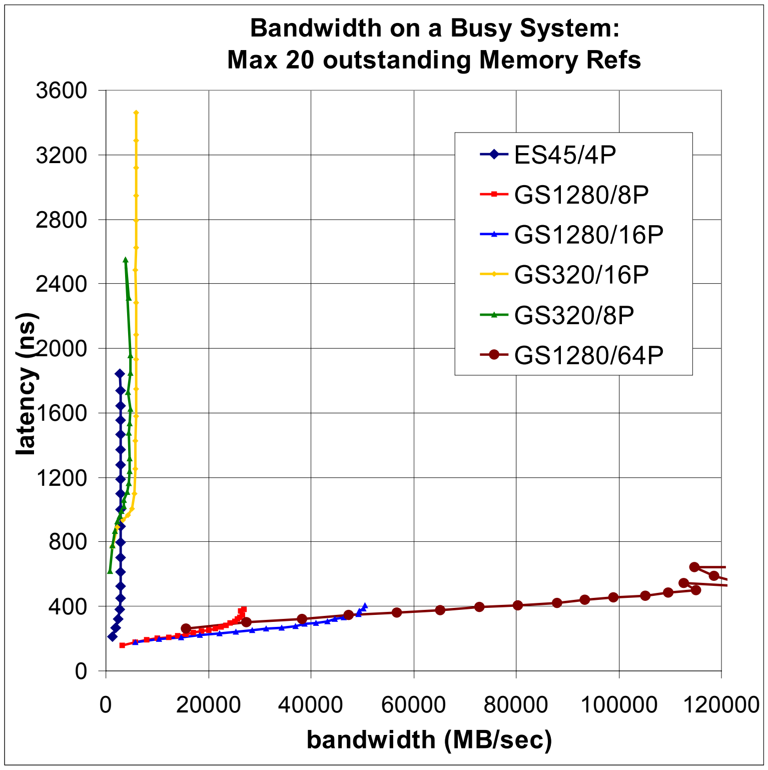


Figure 15. Loaded-system performance comparison.

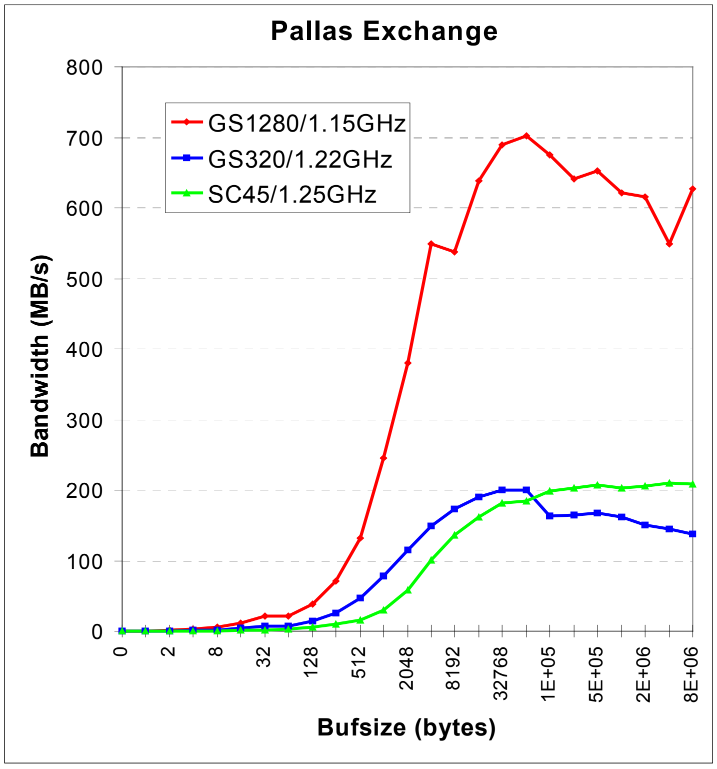


Figure 16. Pallas Exchange comparison.

3.6. GS1280 IO performance and scaling

The IO7 chip connects to the EV7 processor via two unidirectional data paths delivering aggregate bandwidth of 3.1 GB/s. In addition to Standard configuration (Figure 17), the high-performance (Figure 18) configuration is provided. The standard configuration provides ports to 3 PCI/PCI-X busses and one AGP bus.

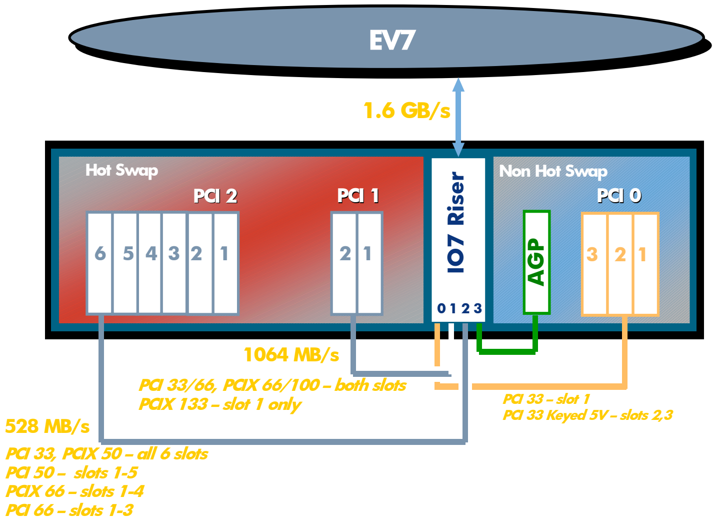


Figure 17. Standard I/O drawer configuration.

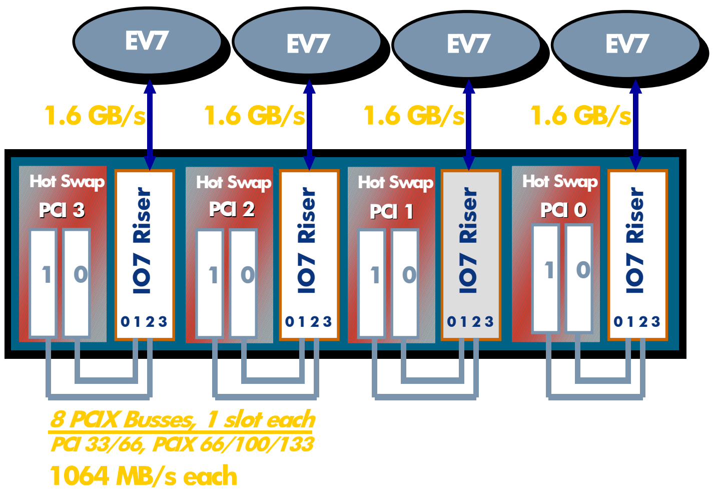


Figure 18. X-Shelf I/O drawer.

Each EV7 within the GS1280 system can be configured with its own IO7 (for highest bandwidth). The measured I/O bandwidth for this configuration with up to 8 CPUs is shown in Figure 19. The I/O bandwidth is measured as “raw device” (no file system): note that real applications that use a file system will experience lower I/O bandwidth. Large (128 KB) transfers are used. Each I/O is configured with enough disks and controllers such that it reaches a saturation point (around 1 GB/s). This data indicates that GS1280 achieves linear scaling in I/O bandwidth, delivering 8 GB/s I/O Read bandwidth across 8 CPUs and 8 IOs. This is a substantially higher I/O bandwidth than the previous-generation Alpha platforms (2.7 times vs. ES45 and 8 times v. GS320). Note that GS1280 advantage comes for the ability to configure each CPU with I/O (while in ES45 and GS320, an I/O is shared by 4 CPUs). This data can be used to extrapolate GS1280 I/O bandwidth for other I/O configurations (e.g. one IO per two or four CPUs).

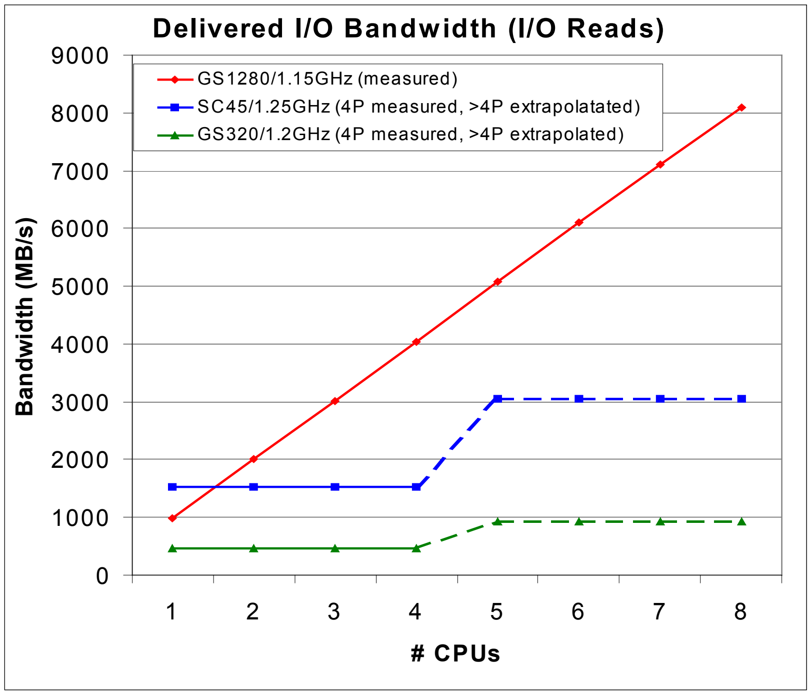


Figure 19. Delivered I/O bandwidth (raw device).

4. Application performance

In the previous section, we discussed GS1280 architectural improvements and performance of various system components (processor, memory, I/O). In this section, we analyze how GS1280 compares to the previous-generation Alpha servers, as well as other-vendor systems over a variety of technical/commercial standard benchmarks (Section 4.1) and HPTC/ISV applications (Section 4.2). In several cases, we include results from our profiling analysis to explain why some applications show different behavior on GS1280. Section 4.2 includes several applications that perform exceptionally well on GS1280 ("killer" applications). We again use profiling analysis to highlight application characteristics that result in outstanding performance on GS1280.

4.1 Standard benchmarks

The standard benchmarks discussed here include the following classes of benchmarks:

- Benchmarks important in HPTC: SPEC2000 (1 CPU and multiprocessor rates), SPEComp2001 (parallel/decomposed SPEC benchmarks), Linpack NxN (solver). Note that STREAM results (also important in HPTC) are discussed in Section 3.2.
- Commercial benchmarks: Transaction Processing, SPECjbb2000, and Decision Support.

4.1.1 CPU2000

Figure 20 shows GS1280 competitive comparison in CPU2000 as of January 10, 2003 [8]. The CPU2000 results on GS1280 will be published in late January 2003. Note that these are 1-CPU workloads and are mainly indicative of processor and memory subsystem performance, as well as compiler optimizations. Only "peak" results are included (where all compiler optimizations are allowed). This data indicates that GS1280 (and EV7 processor) show advantage in SPECfp2000, which is mainly due to EV7's advantage in memory bandwidth (see STREAM results in Section 3.2). The EV7 processor does not show advantage in SPECint2000, since these benchmarks do not stress memory bandwidth and are highly dependent on CPU clock speed. That is the main reason for Pentium4 advantage in SPECint2000, since it is running at close to 2x the frequency of all other CPUs (2.8GHz). On the other hand, EV7 is 1.7 times faster than Pentium4 in SPECfp2000, since these benchmarks put more stress on memory bandwidth (Figure 9). Note that Itanium2 shows excellent performance, indicating that Alpha customers migrating to IPF can expect very good performance in the follow-on future systems.

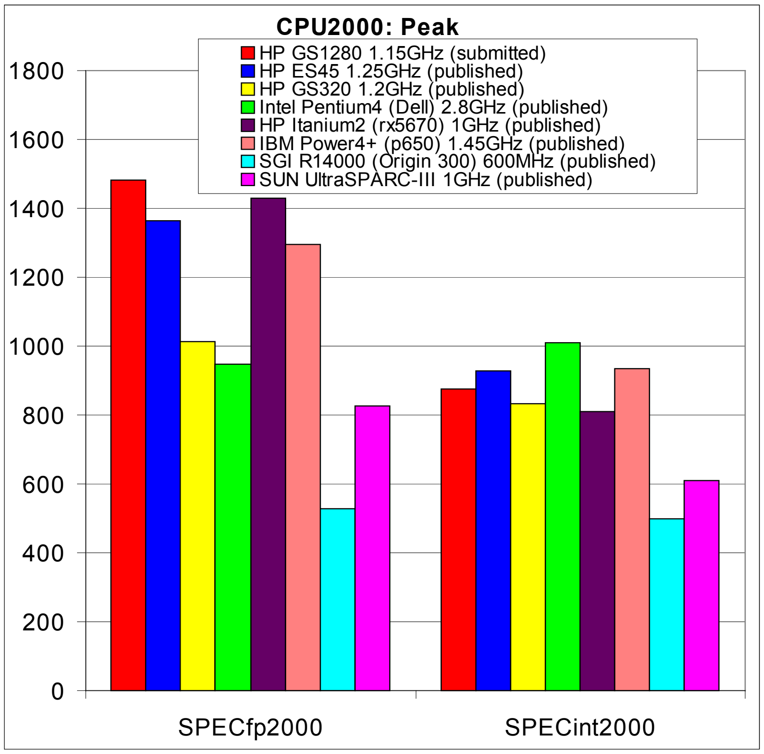


Figure 20. CPU2000 comparison.

4.1.2 CPU2000 Rates

The SPEC2000 rates are a measure of throughput (independent jobs running simultaneously) [8]. Several SPECfp2000 benchmarks stress memory bandwidth, and thus do not scale linearly on all platforms. However, such benchmarks show linear scaling on GS1280 (Figure 21). The reason is in the GS1280 NUMA architecture design, where each processor has its own local memory, thus avoiding contention for memory among simultaneous processes.

The GS1280 server shows substantial competitive advantage in SPECfp2000 rates: 2 times advantage vs. IBM/Power4 (32 CPUs). Note that GS1280 shows much lower advantage (1.2x) on 1 CPU. This indicates that memory-bandwidth intensive throughput applications will scale much better on GS1280 than on IBM/Power4. The GS1280 server also shows substantial (1.7 times) improvement vs. the previous-generation AlphaServer GS320 in SPECfp_rate2000. The GS1280 architecture is a perfect fit for this type of application: memory-bandwidth intensive jobs running simultaneously. Note that GS1280 shows much lower advantage in SPECint2000 rates (Figure 22). This is because these workloads do not stress memory bandwidth, and therefore show close-to-linear scaling on all platforms.

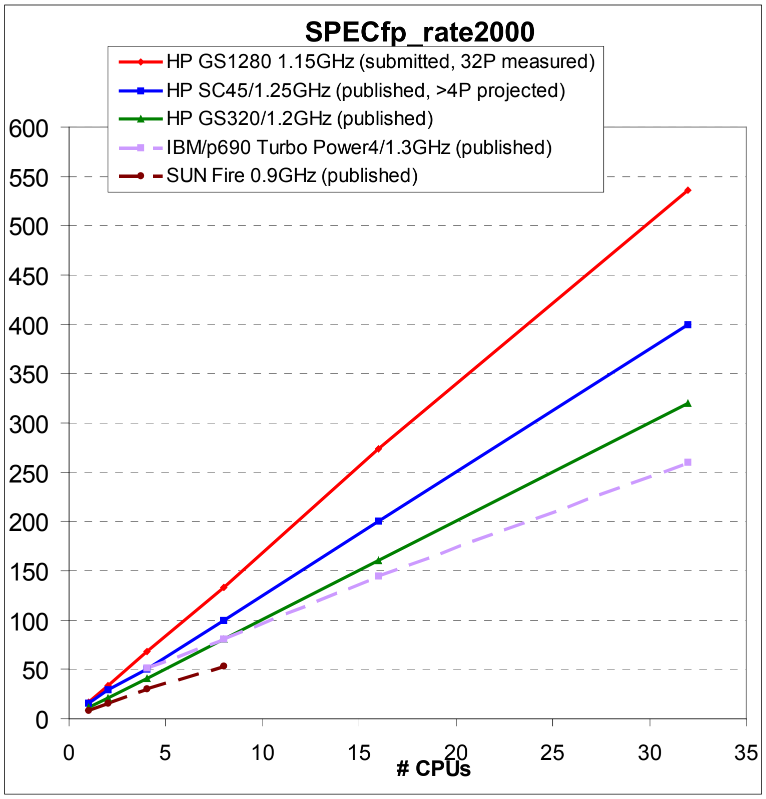


Figure 21. SPECfp_rate2000 comparison.

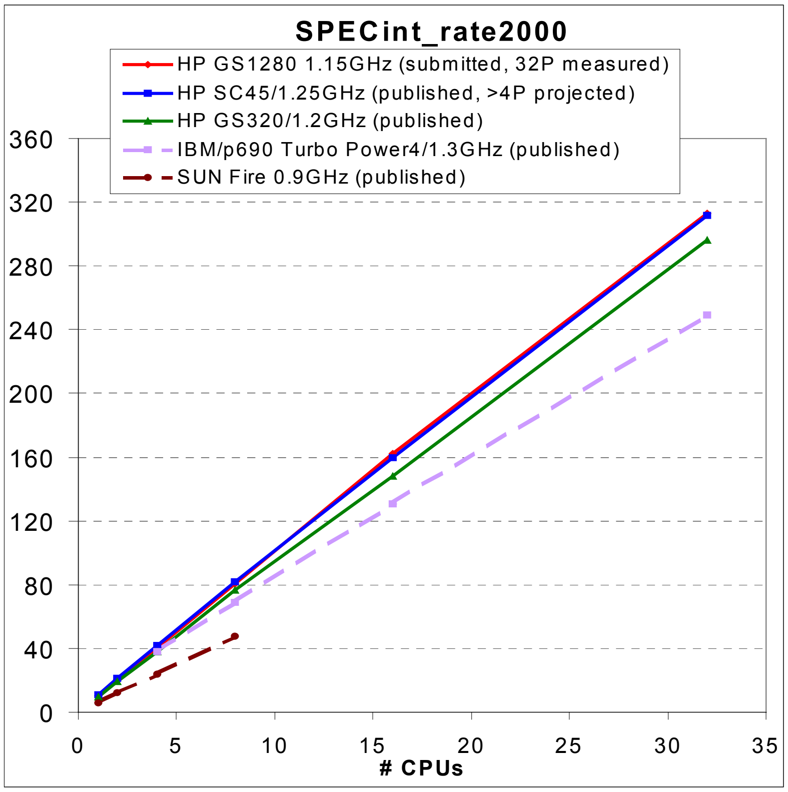


Figure 22. SPECint_rate2000 comparison.

4.1.3 SPEComp2001

SPEComp2001 are OpenMP multi-threaded workloads, decomposed to run on a shared-memory multiprocessor [8]. As opposed to the SPEC2000 rates (Section 4.1.2), these workloads stress not only a single-CPU, but put heavy demand on the inter-processor interconnect (since multiple threads access data through shared memory). This class of workloads is representative of scientific and engineering applications (important in HPTC). Eleven different application benchmarks - covering everything from computational chemistry to finite-element crash simulation to shallow water modeling - are included in the benchmark suite.

The results for OMPM2001 are compared in Figure 23. This data shows that GS1280 has significant performance advantage over other comparable systems. GS1280 shows 20% advantage vs. IBM/Power4+/1.45GHz. In addition, GS1280 shows very good scaling, with a 16-CPU result matching the 64-CPU SGI result, and outperforming SUN 16-CPU result by almost a factor of 3 times. GS1280 also shows substantial advantage vs. the previous-generation Alpha platforms such as GS320 and ES45.

The superior Marvel performance in these workloads is due to excellent hardware (as indicated in Figure 15) and software support for NUMA.

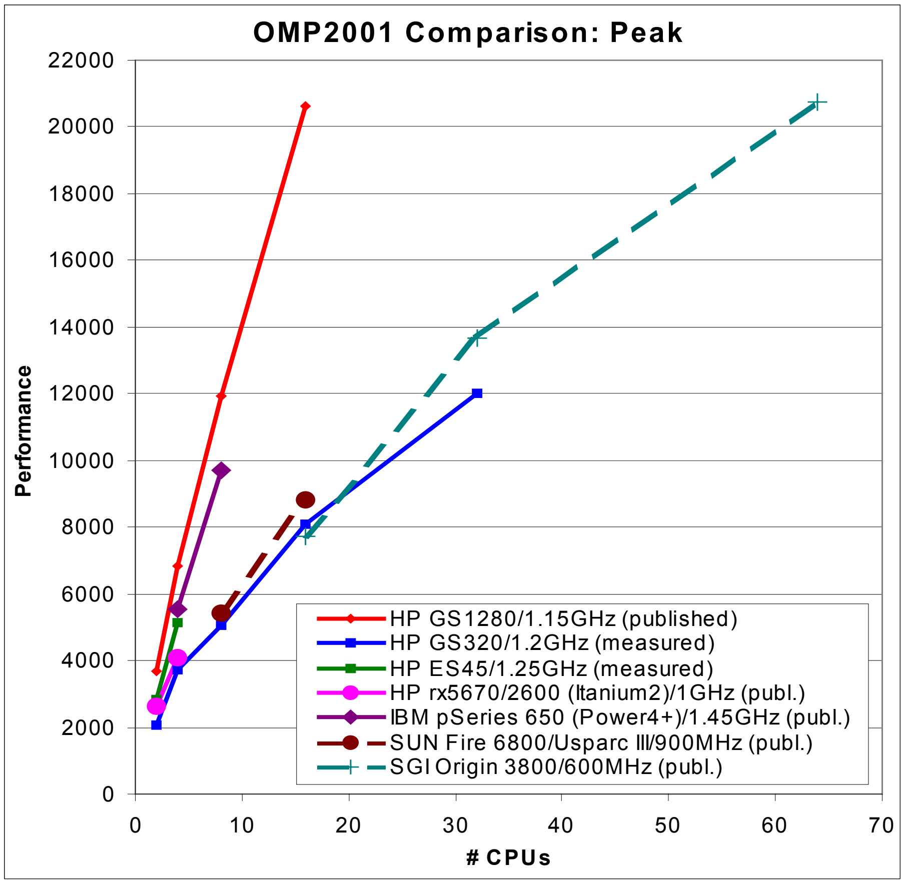


Figure 23. SPEComp2001 (OMPM2001) comparison.

4.1.4 Linpack NxN

Linpack NxN is a solver of a dense system of linear equations [28]. The benchmark is fairly popular in HPTC; it is used as a metric for Top500 rating of supercomputer systems. The NxN version allows users to scale the size of the problem and to optimize the software to achieve the best performance on a given system. The benchmark is usually blocked to fit well in the on-chip caches, thus not stressing memory bandwidth. The performance of this benchmark is determined mainly by the CPU clock speed, effectiveness of the math library functions (BLAS), and the number of floating-point units.

All Alpha platforms show disadvantage in the benchmark (Figure 24). The reason is in the lower number of floating-point units in Alpha vs. most other processors (one Multiply and one Add instruction in Alpha vs. 2 MUL/ADD instructions in other processors). Therefore, this Alpha disadvantage will show in most applications that run close to peak FLOPS performance. In our experience and from profiling data based on real customer benchmarks, we find that most real applications rarely run at the peak FLOPS performance. In most cases, large applications cannot be easily blocked to fit in the on-chip caches, and their performance is limited by the memory subsystem.

Although GS1280 shows performance disadvantage in this benchmark, the results show very good (linear) scaling to large number of CPUs. In addition, GS1280 shows higher percentage of the peak performance than most other platforms (85% of the peak vs. 57% in IBM/Power4).

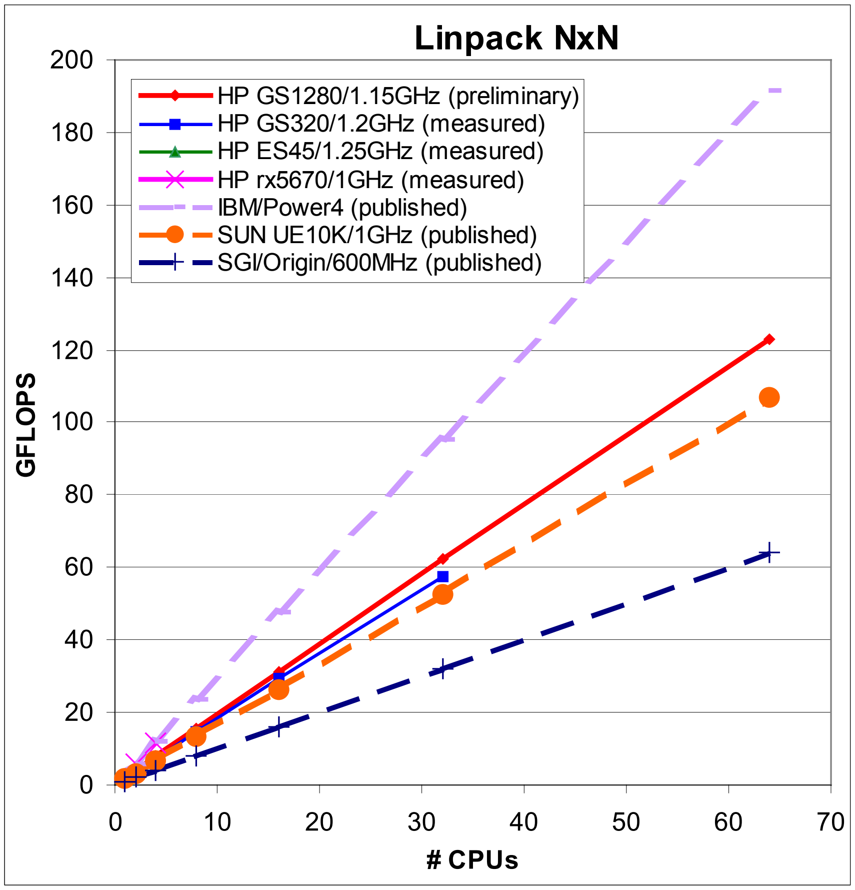


Figure 24. Linpack NxN comparison.

4.2 ISV/HPTC Applications

The following ISV/HPTC applications are discussed in this section:

- Structural modeling: NASTRAN, Abaqus, Marc
- Computational Fluid Dynamics (CFD): Fluent, StarCD, PowerFLOW
- Crash analysis: LS-Dyna
- Material Sciences: Gaussian98, NWchem, Amber, MM5 (Weather prediction)
- Life Sciences: Blast, Fasta, other

4.2.1 NASTRAN

NASTRAN is a popular Structural-modeling application [18]. It consists of 6 kernels used for car-body or engine modeling.

The GS1280 results are shows in Figures 29-32. This data indicates that GS1280 shows advantage vs. IBM/Power4 in xlrs, lgqd and tdf. GS1280 shows much better scaling than IBM. GS1280 also shows significant advantage vs. the previous-generation Alpha platform GS320. In most cases, GS1280 performance is comparable to ES45/1.25GHz (up to 4 CPUs).

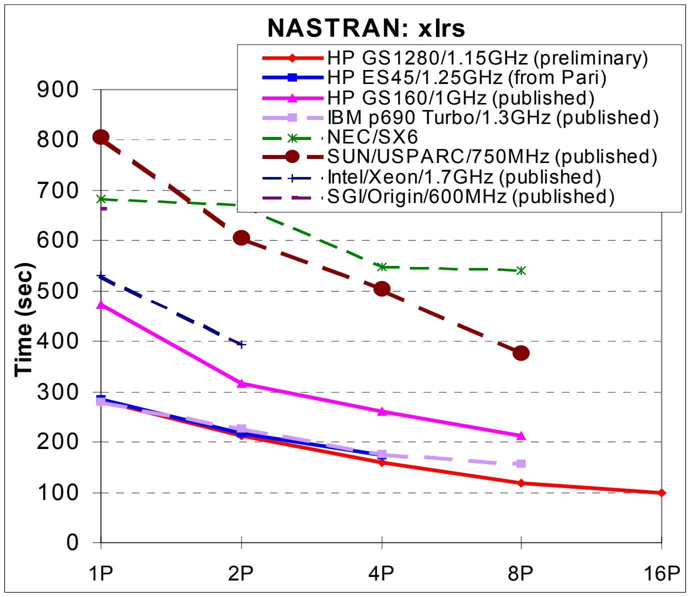


Figure 29. Nastran xlrc comparison.

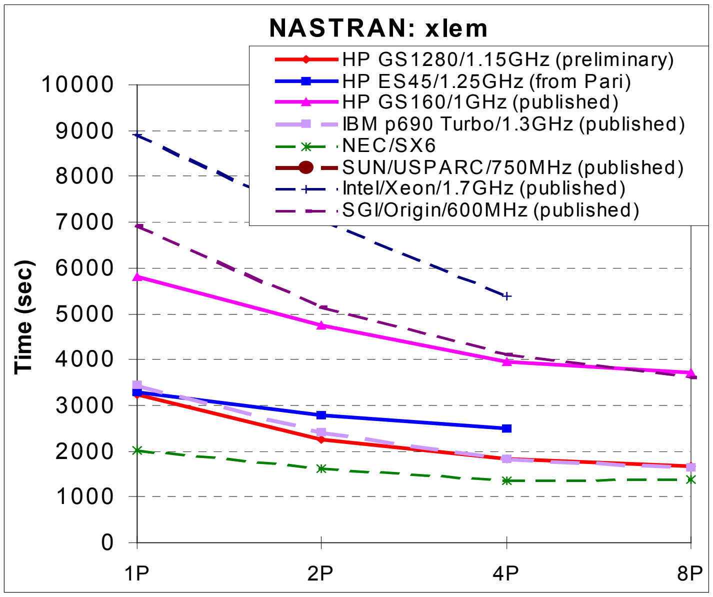
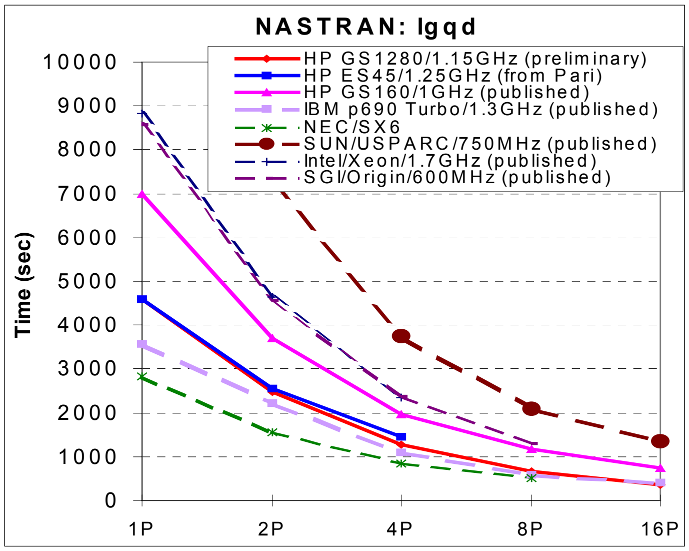


Figure 30. Nastran xlem comparison.



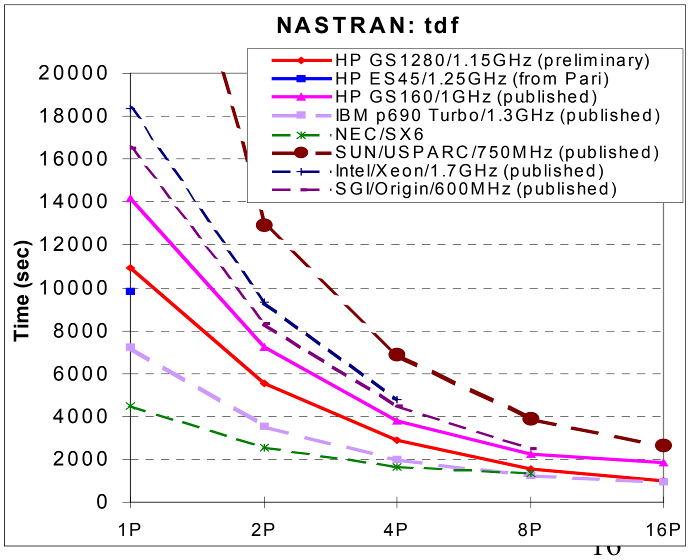


Figure 31. Nastran lgqd comparison. Figure 32. Nastran tdf comparison

4.2.2 Abaqus

Abaqus is a Structural-analysis application [19]. Abaqus/Standard provides traditional finite element analysis (static, dynamics, thermal). Abaqus/Explicit is focused on transient dynamics and quasi-static analyses (drop test, crushing, and many manufacturing processes).

As indicated in Figure 33, GS1280 shows 10-30% advantage vs. IBM/Power4 in Abaqus/Explicit. Note that version 6.2 is used for comparison. Since 6.2 submissions are no longer accepted, version 6.3 is likely to be published. The reason we used 6.2 here is that no published 6.3 competitive results were available at the time this paper is written.

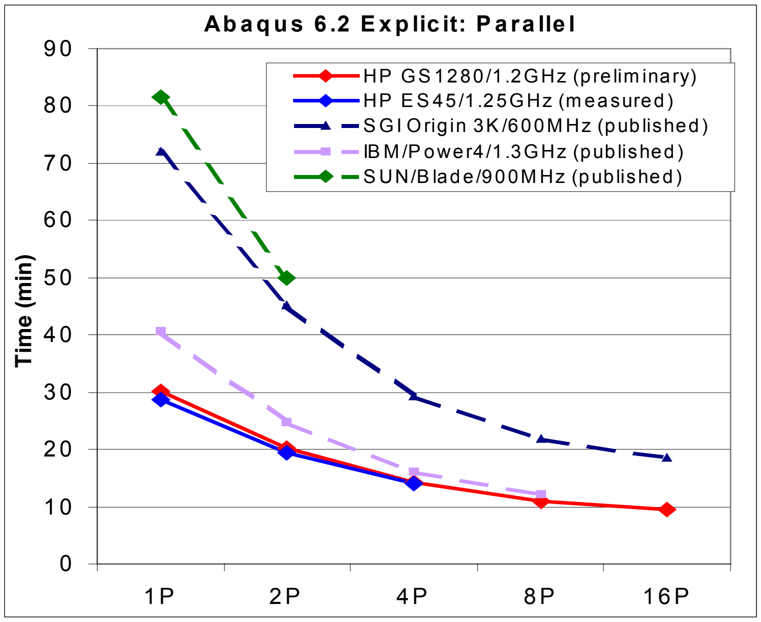
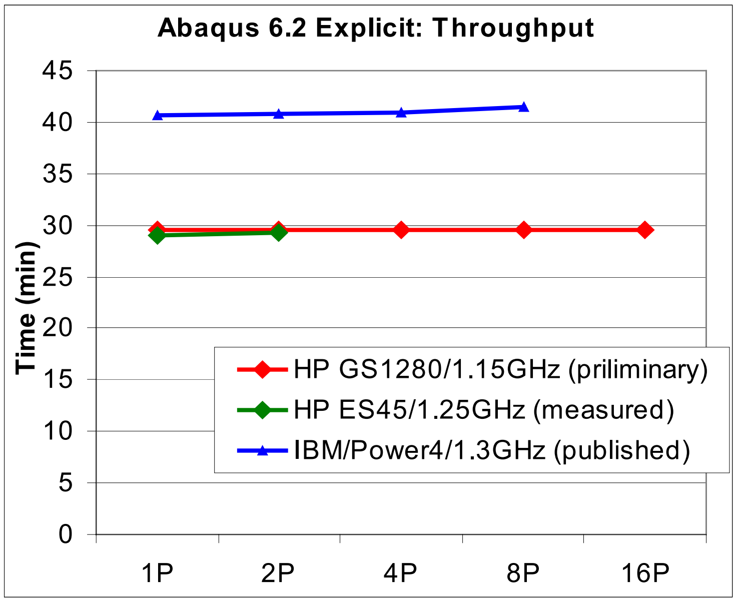


Figure 33. Abaqus Explicit comparison.

Figures 34 and 35 show linear scaling on GS1280 (no increase in execution time) in Abaqus/Throughput (multiple Abaqus jobs executed simultaneously). The GS1280 shows performance advantage in Explicit vs. IBM/Power4, but no advantage in Standard (1-CPU). Note that IBM published only 1-CPU Standard performance. The GS1280 shows linear scaling in Standard/Simultaneous, while the scaling on IBM is unknown.



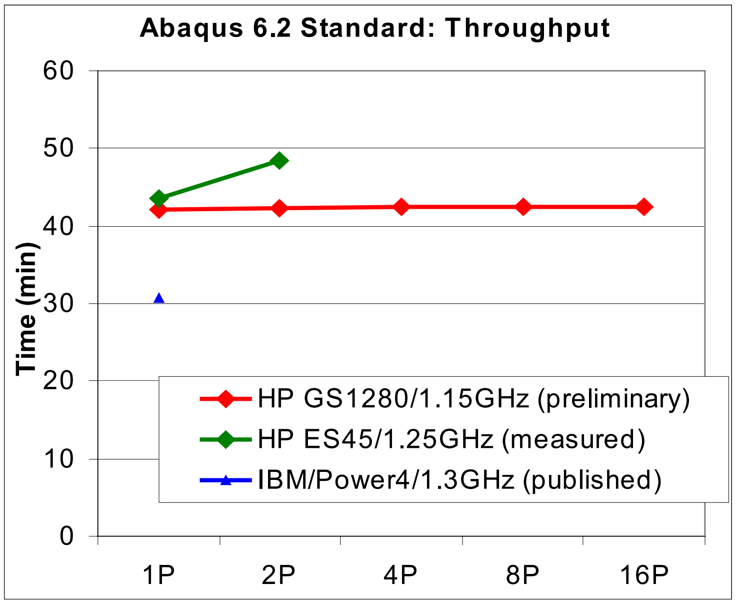


Figure 34. Abaqus Explicit/Simultaneous comparison. Figure 35. Abaqus Standard/Simultaneous comparison.

4.2.3 Marc

Marc is a non-linear finite element solver that includes Engine, Thread Rolling, and Turbine Blade benchmarks [20].

The data shown in Figure 36 are for the Thread Rolling model (note: other results are comparable). The GS1280 shows advantage vs. the other Alpha platforms. It shows good scaling (note: application does not scale well beyond 8 CPUs). Note that there are no recent competitive results published for Marc (IBM in particular).

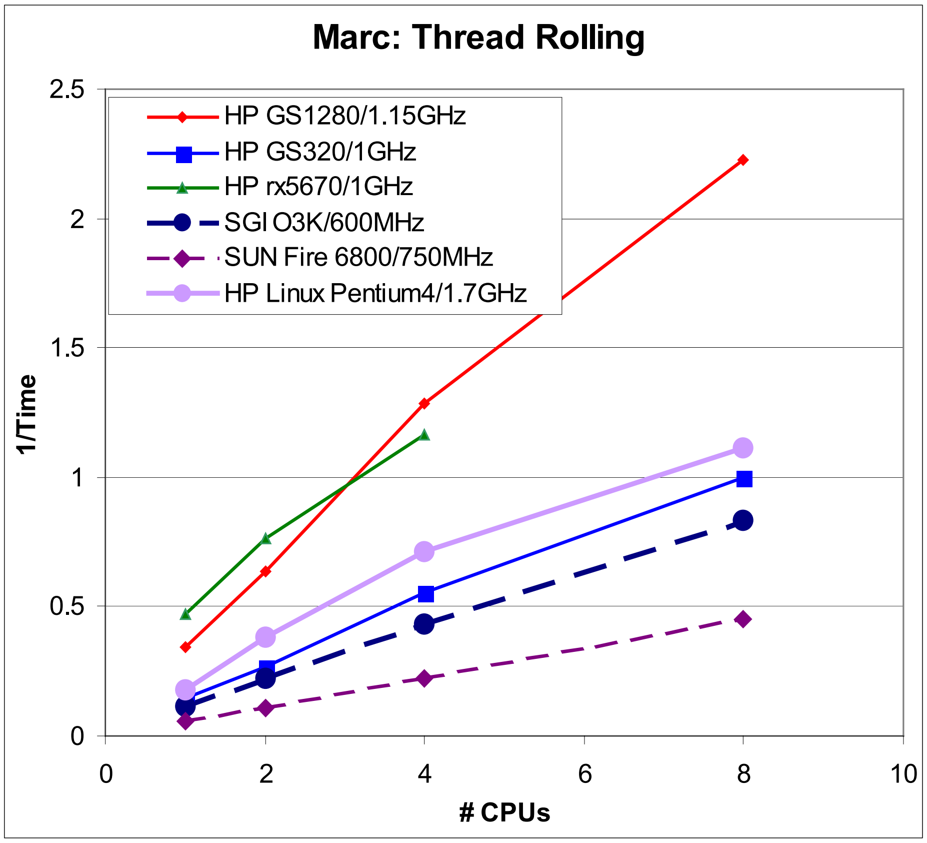


Figure 36. Marc Thread Rolling comparison.

4.2.4 Fluent (CFD)

Fluent is a commercial Computational Fluid Dynamics application [13]. For comparison, we selected a large case (l1) that models flow around a fighter aircraft.

Figure 37 compares GS1280 performance to the previous-generation Alpha platforms and other-vendor published results. This data indicates that GS1280 shows comparable performance to the SC45. Examining Figures 38 and 39 with measured utilization shows that the reason is that this benchmark does not put significant stress on either memory controller or IP-links bandwidth (thus explaining comparable performance to SC45). GS1280 shows 20% advantage vs. IBM/Power4 and substantial (2-3 times) advantage vs. SUN. GS1280 also shows substantial (1.5 times) advantage vs. previous-generation Alpha platform (GS320).

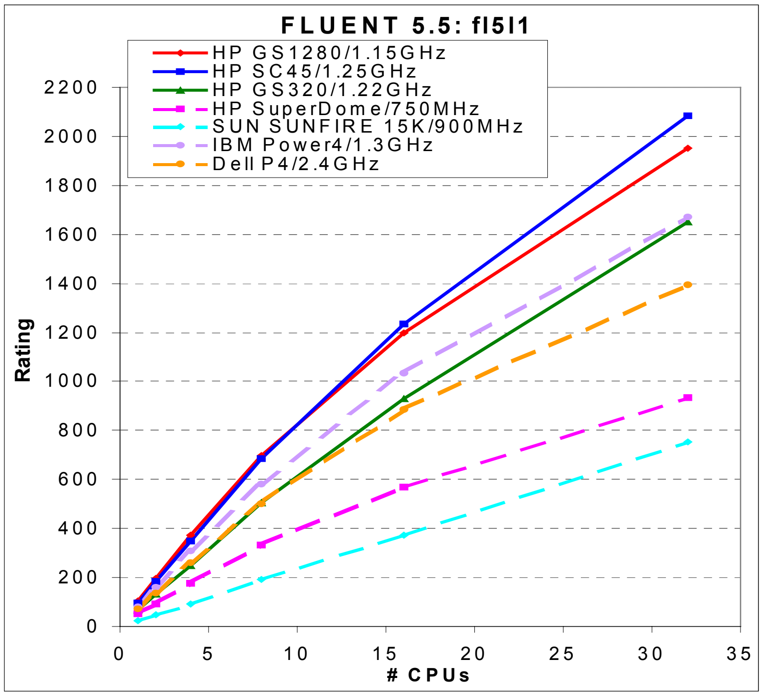
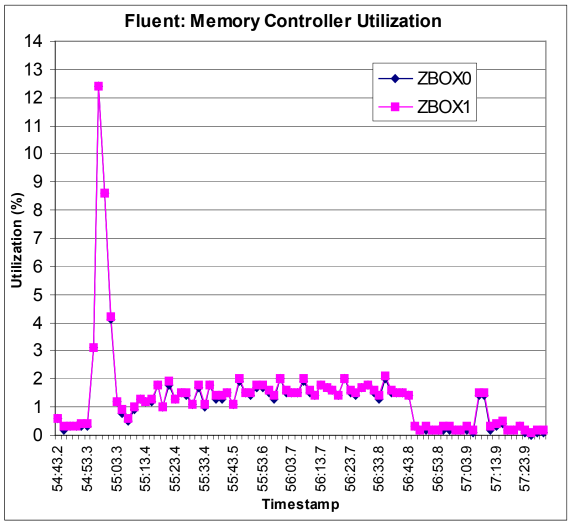


Figure 37. Fluent fl5l1 performance comparison.



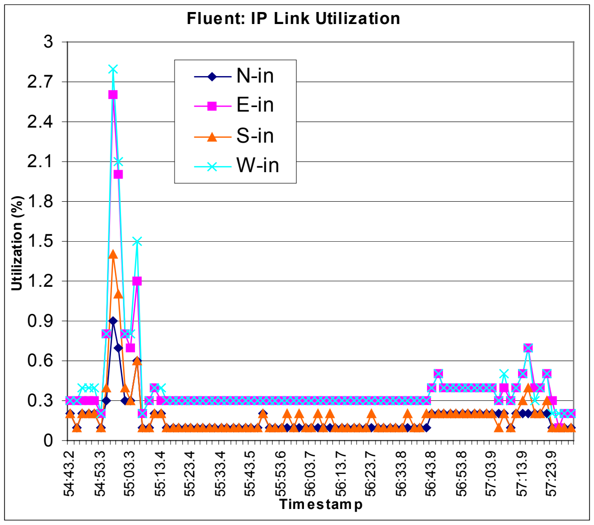


Figure 38. Memory Controller utilization in Fluent.

Figure 39. IP links utilization in Fluent.

4.2.5 PowerFLOW

PowerFLOW is another CFD application [22]. The benchmark shown here is the external airflow around a car.

GS1280 shows significant performance advantage and much better scaling than IBM (20% with 16 CPUs and 50% with 32 CPUs). GS1280 shows comparable performance as ES45/1.25GHz (Figure 40).

4.2.6 StarCD

StarCD is another CFD application, modeling turbulent flow around the A-class car [23].

GS1280 shows significant performance advantage vs. the other Alpha platforms (Figure 41). This is mainly due to exceptional memory bandwidth on GS1280 (Figure 5). Our profiling data indicates that this application shows high memory bandwidth utilization (15-18% on average), and well as high IP-link utilization (~10%). This explains substantial advantage of GS1280 vs. GS320. Using more aggressive prefetching, we were able to improve performance of this workload by 20-30% on 16-32 CPU GS1280.

GS1280 also shows advantage vs. IBM/Power4 (25% with 32 CPUs). The scaling is significantly better on GS1280 than on IBM. Itanium2 shows excellent performance in this application (Figure 41).

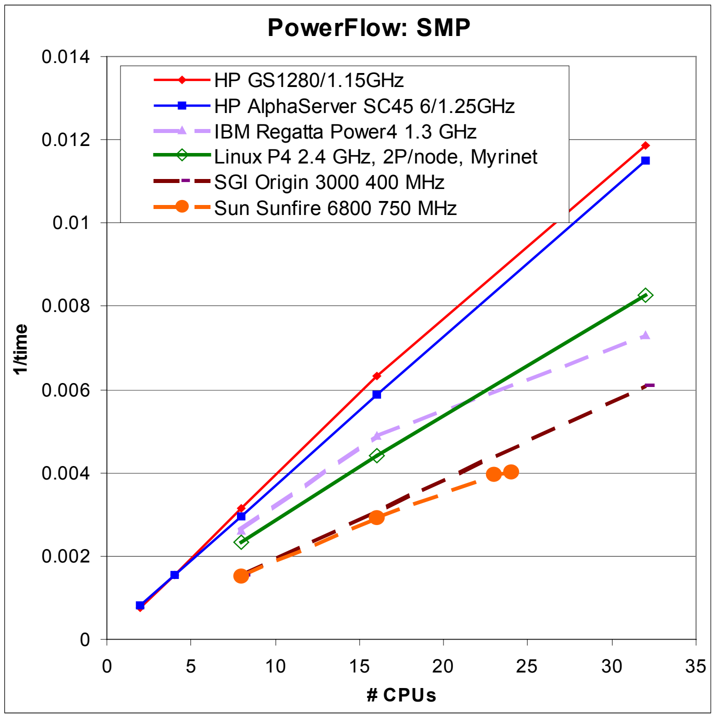


Figure 40. PowerFLOW comparison.

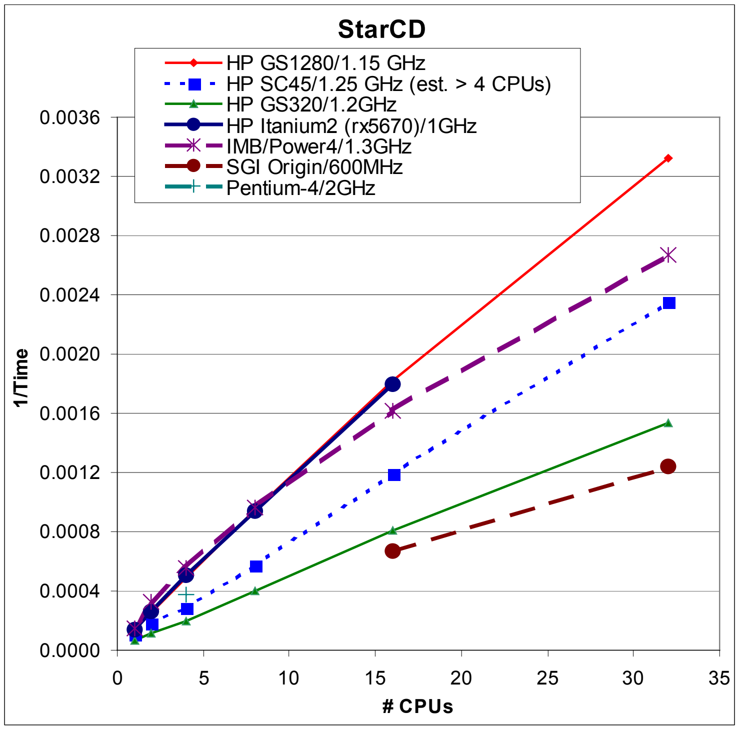


Figure 41. StarCD comparison.

4.2.8 Crash modeling: LS-Dyna

LS-Dyna is a crash simulation application [24]. The results for the model of a Neon dataset with 535K elements are shown in Figures 42a (for SMP version) and 42b (for MPI version). The results from Figure 42 show that GS1280 performs comparably to IBM (IBM results up to 2 CPUs based on internal HP measurements). Itanium2 shows very good performance in an SMP version. The MPI version shows GS1280 advantage over other systems and linear scaling in performance as the number of CPUs is increased.

The LS-Dyna Caravan results from Figure 43 show substantial performance advantage on GS1280 vs. GS320. The advantage is particularly pronounced for 8 CPUs. The profiling data indicate that lower remote latency (clean and dirty) is the key factor for GS1280 advantage.

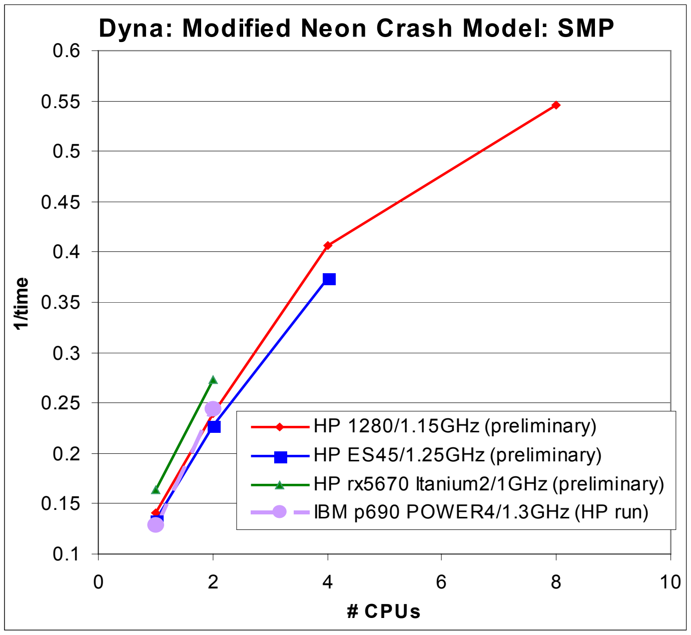


Figure 42a. Dyna Neon comparison (SMP).

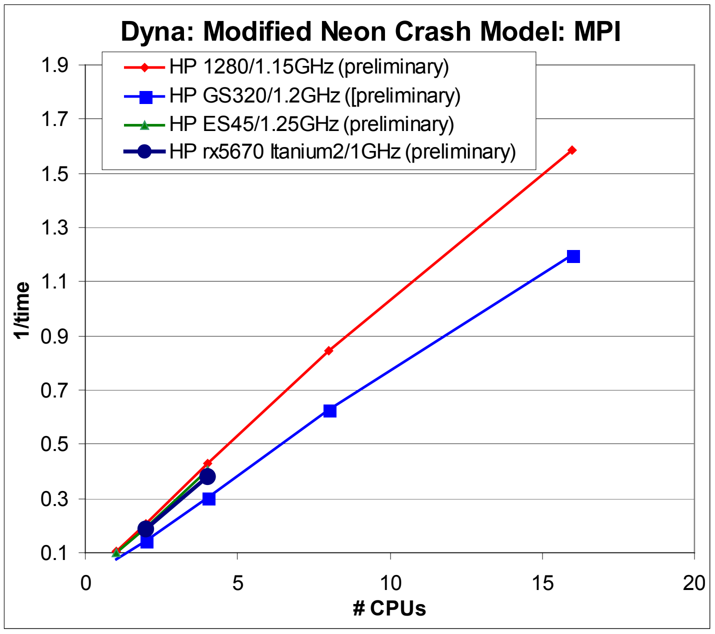


Figure 42b. Dyna Neon comparison (MPI).

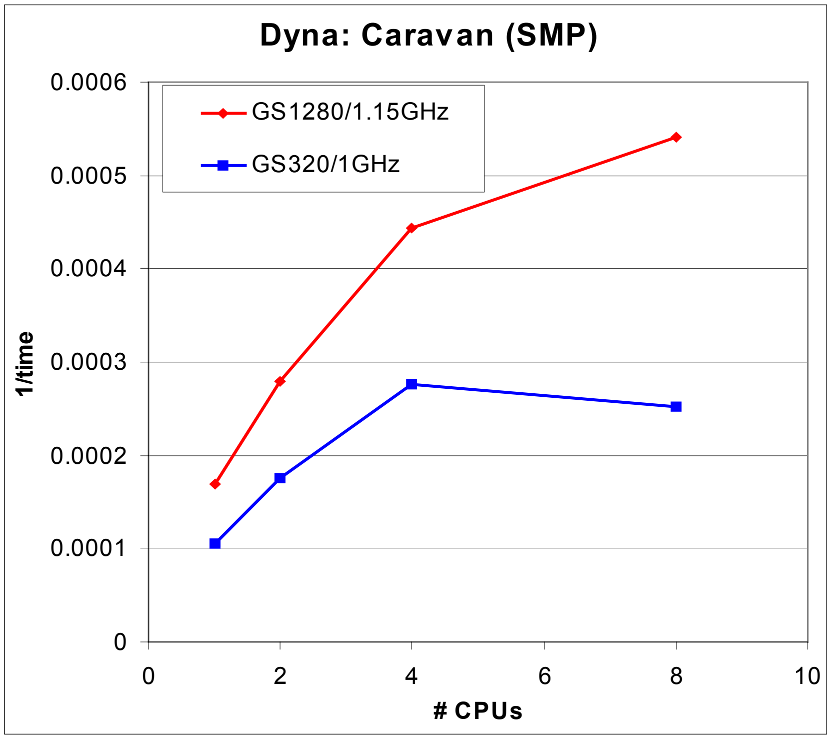


Figure 43. Dyna Caravan comparison (SMP).

4.2.9 NWchem

NWchem is a molecular dynamics modeling application [25]. Data from Figures 44 and 45 show GS1280 advantage for more than 16 CPUs (thus GS1280 has better scaling than other platforms). GS1280 shows 20-30% advantage vs. the previous-generation GS320 and 10-20% advantage vs. IBM/Power4.

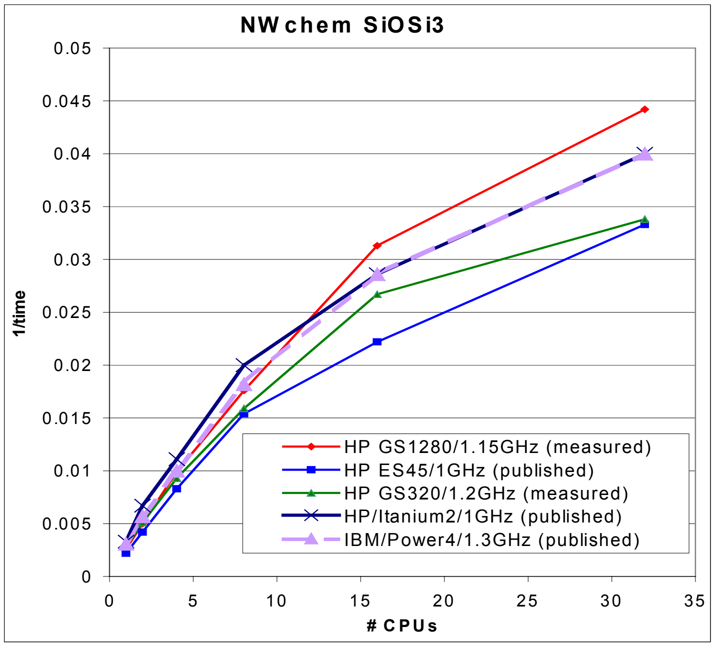


Figure 44. NWchem SiOSi3 comparison.

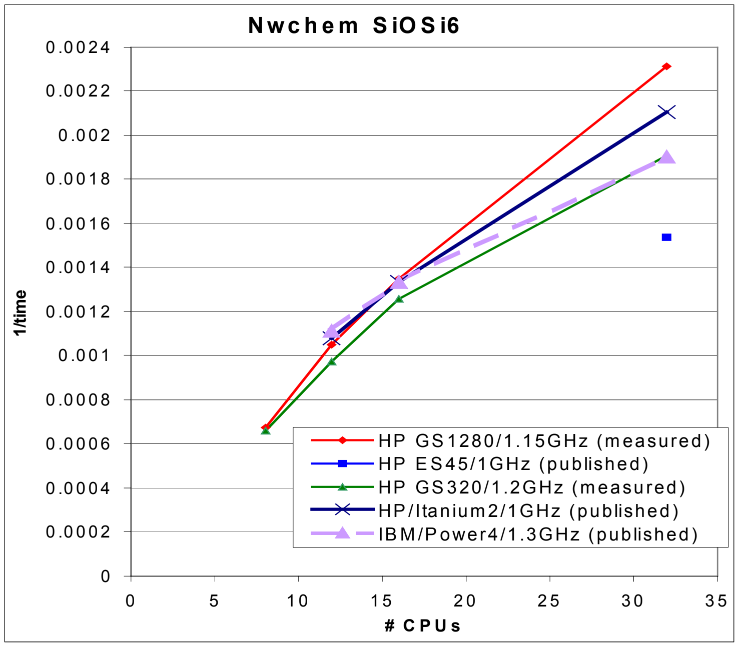


Figure 45. NWchem SiOSi6 comparison.

4.2.10 MM5: weather prediction

MM5 is the Penn State NCAR Mesoscale Model5 weather prediction [26]. GS1280 shows much better scaling than IBM/Power4: 20% advantage with 16 CPUs and 40% advantage with 32 CPUs (Figure 46).

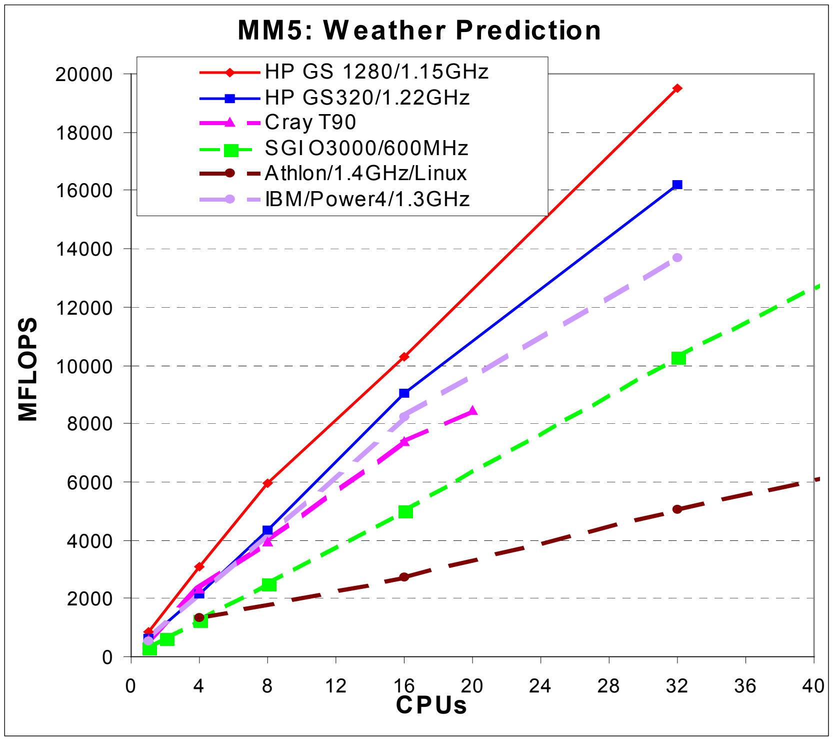


Figure 46. MM5 weather prediction comparison.

4.2.11 Gaussian98

Gaussian98 is a chemistry application. The Alpha Pinene input files are used for the comparison below (Figure 47). Data shows that GS1280 has substantially better scaling than GS320 (16 and 32 CPUs). GS1280 also shows substantial performance advantage and much better scaling advantage than IBM/Power4. The ES45 and Itanium2 outperform GS1280 for 4 CPUs.

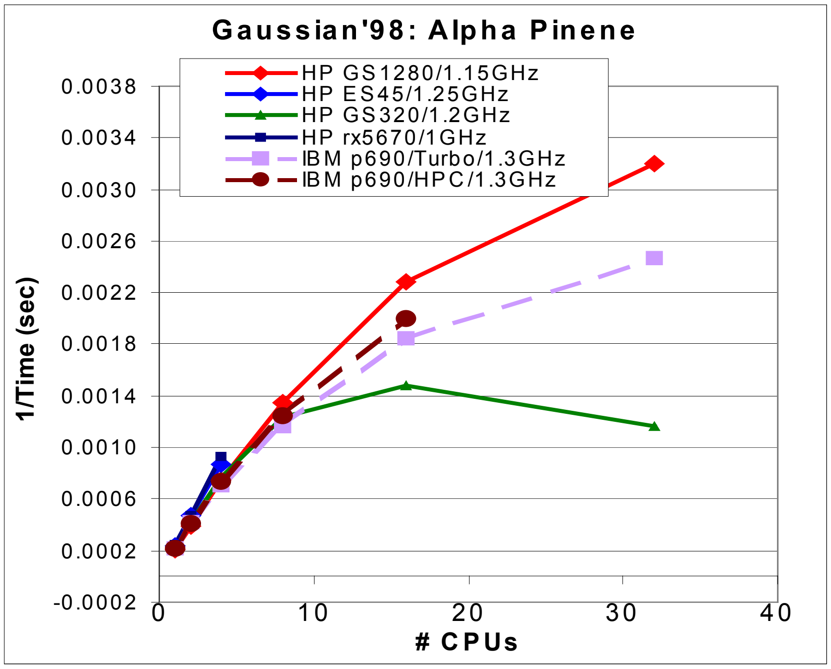


Figure 47. Gaussian98 comparison.

4.2.12 Amber

Amber is another chemistry application [27]. GS1280 shows good scaling and comparable performance as ES45 up to 4 CPUs (Figures 48 and 49). GS1280 scales much better than GS320 (substantially better performance on 32 CPUs).

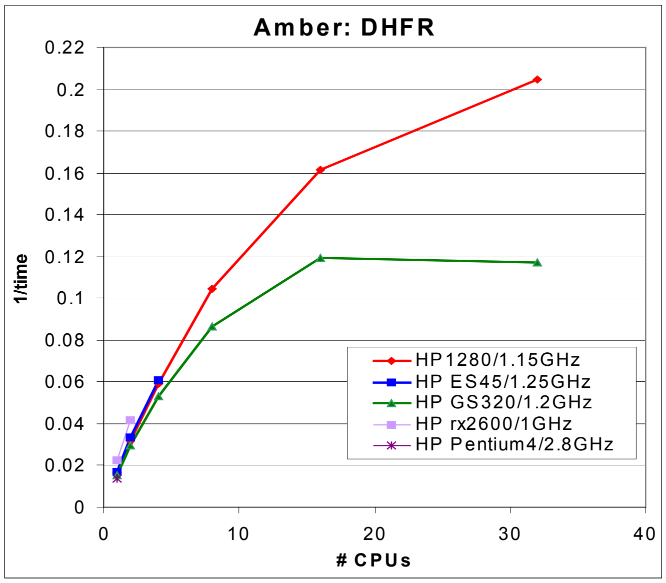


Figure 48. Amber DHFR comparison.

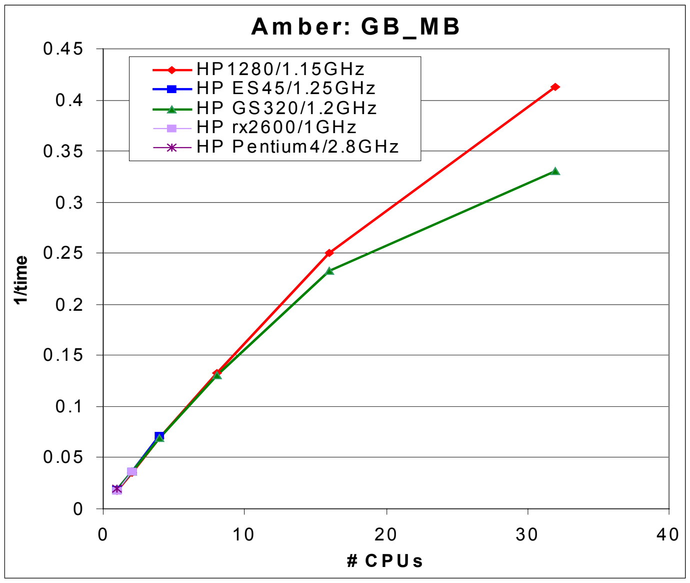
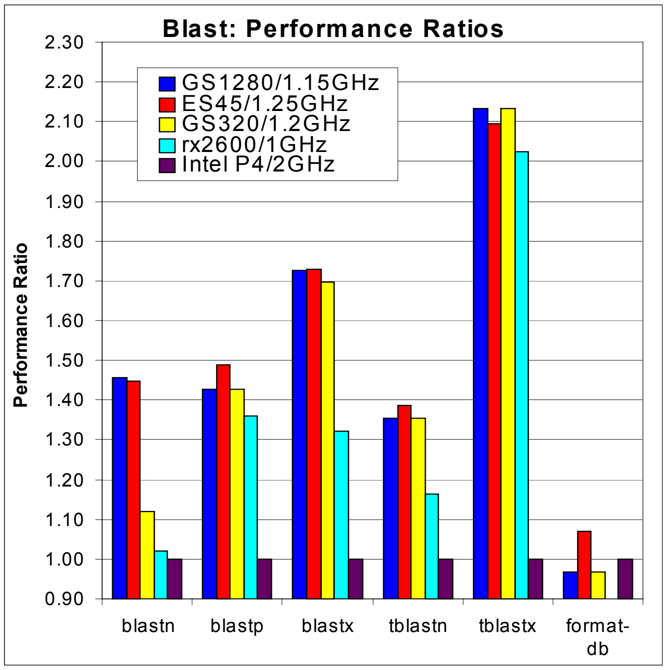


Figure 49. Amber GB_MB comparison.

4.2.13 Life Sciences

This set of benchmarks includes BLAST and FASTA (performance ratios relative to Intel/Pentium4/2GHz are shown in Figures 50 and 51). Note that these only 1-CPU data is available for these workloads. In Blast (most important set of benchmarks) GS1280 shows comparable performance to ES45 and GS320. GS1280 shows substantial advantage (2 times) vs. Pentium4/2GHz. Note that these benchmarks use integer operations and do not stress memory bandwidth. GS1280 shows lower advantage in Fasta.



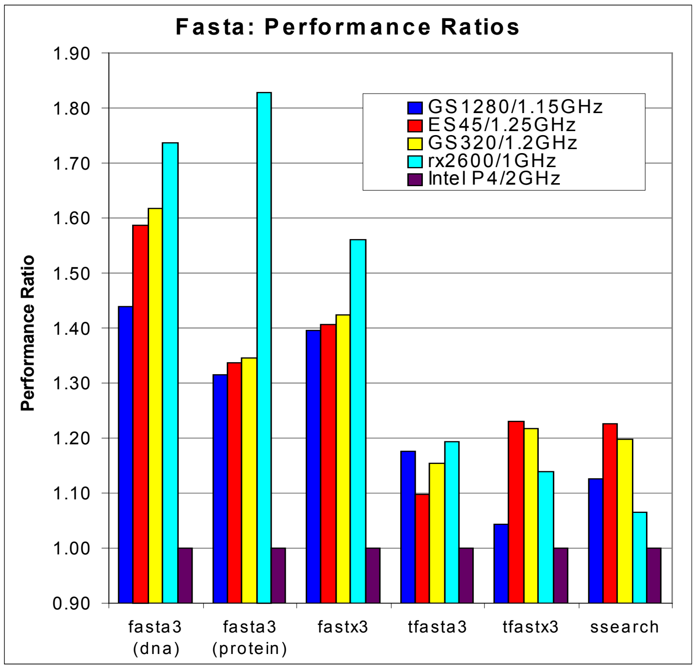
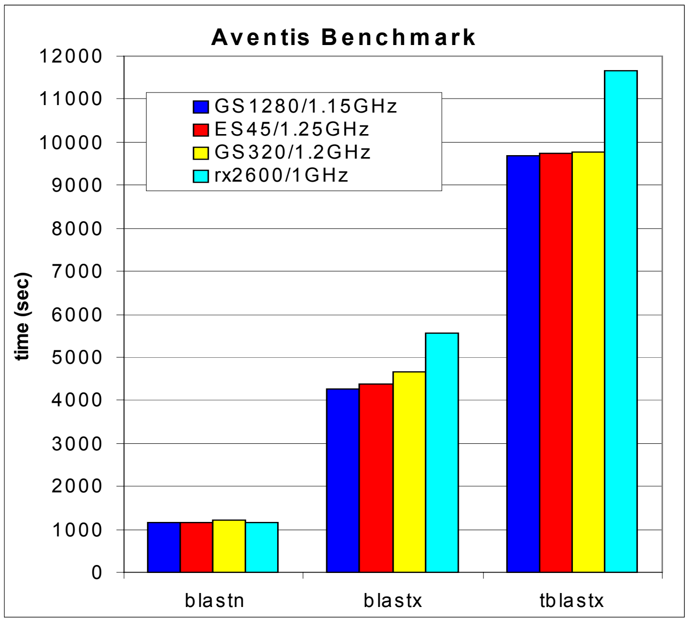


Figure 50. Blast performance ratios. Figure 51. Fasta performance ratios.

In Aventis and Couragen (Figure 52), GS1280 is 2-10% faster than ES45 and 1-30% faster than Itanium2. In other benchmarks GS1280 shows a wide range of performance: between 20% slower and 20% faster vs. ES45, 30% slower and 3% faster vs. Itanium2, and 40% slower and 8 times faster vs, Pentium4.



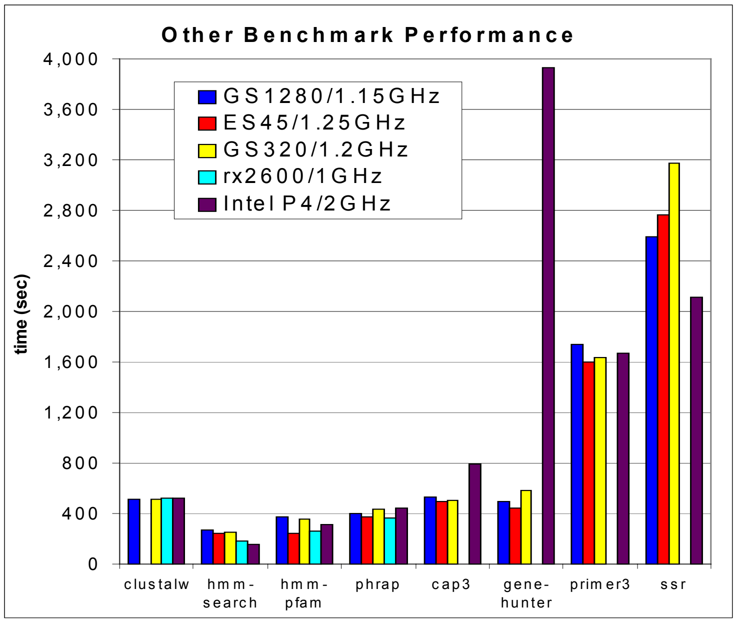


Figure 52. Aventis and performance comparison. Figure 53. Other benchmarks performance comparison.

4.3 “Killer” applications on GS1280

The previous sections indicate that in many standard HPTC/ISV applications, GS1280 shows comparable performance as ES45. Compared to IBM, GS1280 shows 20-30% advantage, and much better scaling for 16-32 CPUs.

In this section, we identify and characterize applications where GS1280 shows exceptional performance (“killer applications”). These applications provide much more stress on memory subsystem and IP-link bandwidth. Our profiling data indicates that none of the ISV applications provides substantial stress on any of these system components.

In this section, we include several examples of “killer applications” on GS1280:

1. Applications that stress memory bandwidth:
   - Swim: from SPEComp2001
   - NAS Parallel (except EP)
2. Applications that stress both memory and IP links bandwidth:
   - GUPS (random updates to a large memory)
3. Many customer benchmarks

4.3.1 Swim: memory bandwidth intensive application

Swim is one of the SPEComp2001 parallel benchmarks [8]. Swim is the weather prediction application that performs finite-element model of the shallow-water equations. This workload provides heavy stress on the memory subsystem (30% memory bandwidth utilization measured). It provides low stress on the IP links (Figure 9).

As indicated in Figure 54, GS1280 is substantially faster in this benchmark: 3-4.5 times vs. ES45, 4-5.4 times vs. GS320, 4 times vs. IBM/Power4, 7-10 times vs. SUN.

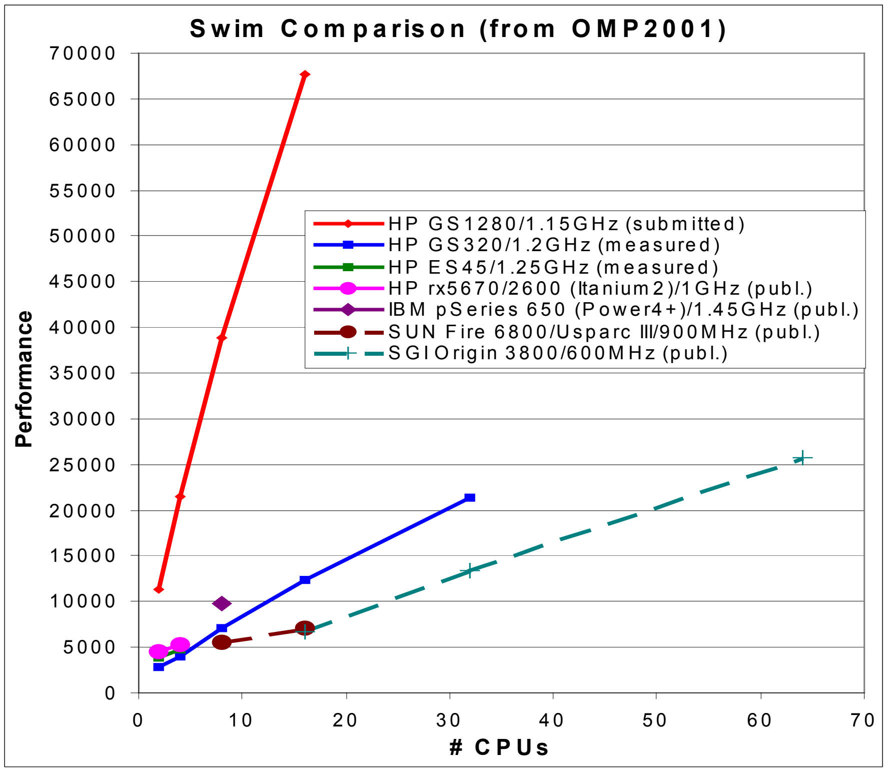
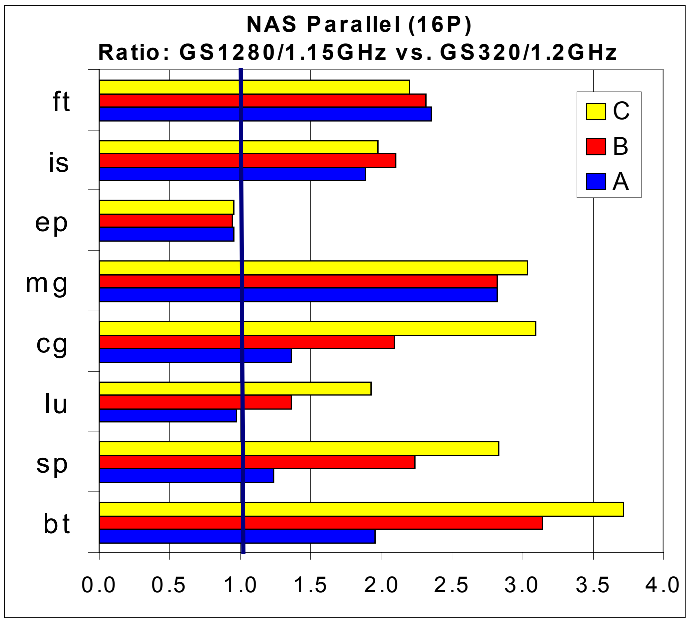


Figure 54. Swim comparison.

4.3.2 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), the majority of these kernels (solvers, FFT, grid, integer sort) put significant stress on memory bandwidth (when size C is used). Figures 55 and 56 show substantial advantage on GS1280 compared to the other systems: 1.5 times advantage vs. ES45 (Figure 56) and 3.5 times advantage vs. GS320 (Figure 55). The reason that this advantage is higher for GS320 than ES45 is because ES45 has higher memory bandwidth than GS320 (as indicated in Figure 5).



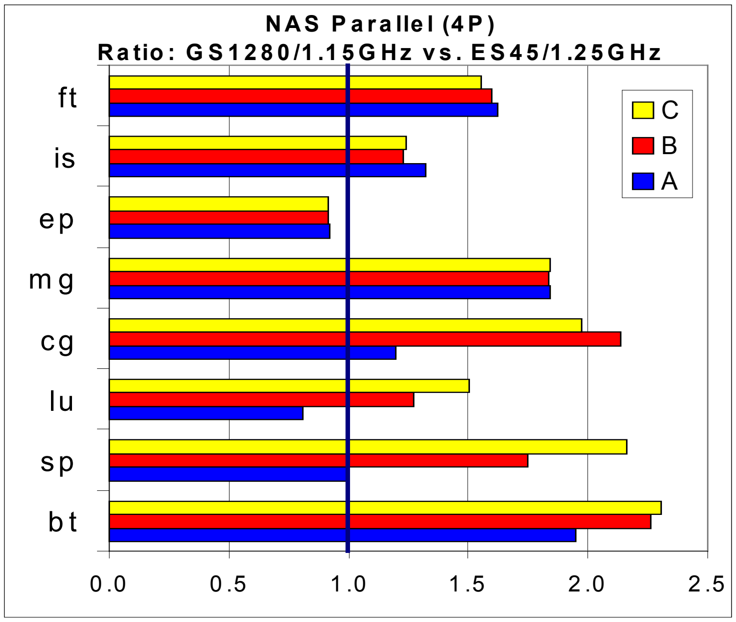


Figure 55. GS1280 vs. GS320 in NAS Parallel.

Figure 56. GS1280 vs. ES45 in NAS Parallel.

The scaling of one of the NAS Parallel solver benchmarks (SP) is shown in Figure 57. GS1280 shows excellent scaling and substantial (1.8 times) advantage vs. IBM/Power4 in this benchmark.

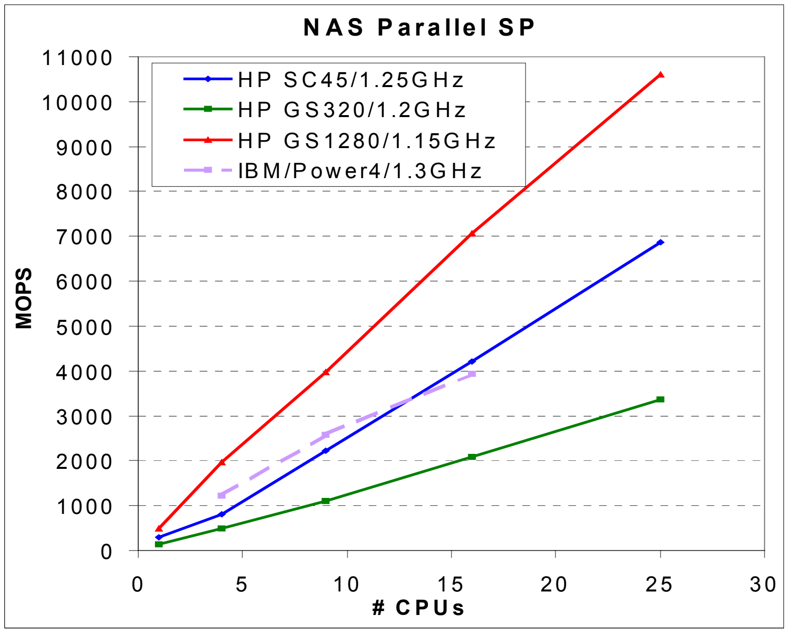


Figure 57. SP comparison.

SP profiles

Figure 58 shows that memory bandwidth utilization is high in SP (26%), which explains the advantage on GS1280. The IP links utilization in these benchmarks is low (Figure 59). We observed that IP link utilization is low in many MPI applications. We believe that's because such applications are message-passing written with much higher (cluster-type) latencies in mind.

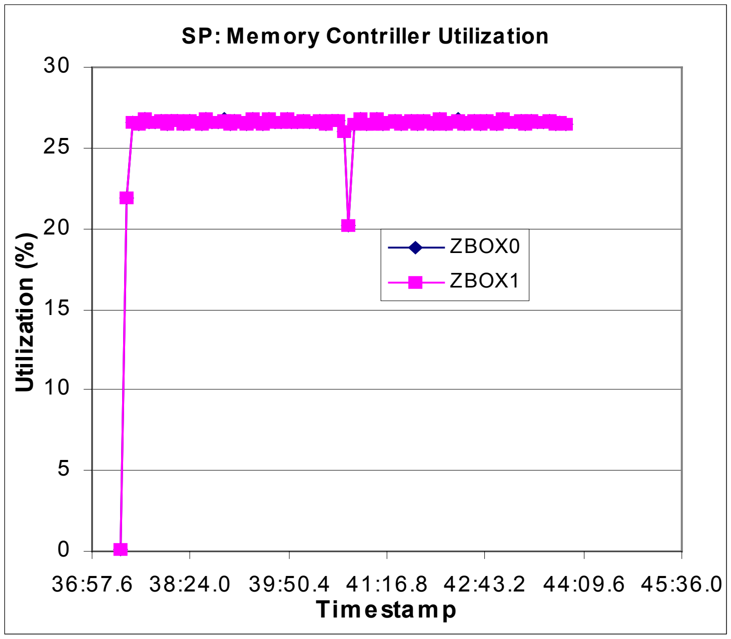


Figure 58. Memory Controller utilization in SP.

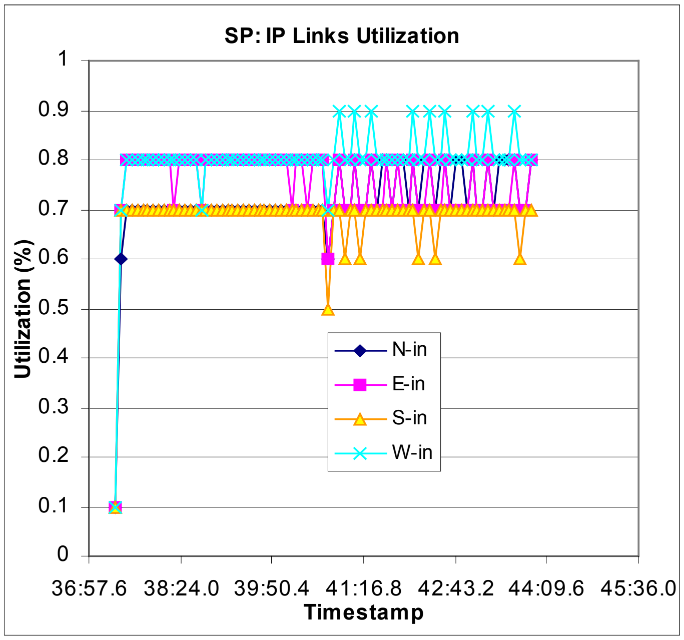


Figure 59. IP links utilization in SP.

4.3.3 IP-links and memory bandwidth intensive application: GUPS

The 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 stresses memory bandwidth, as well as IP links (as illustrated by profiling data in Figures 61 and 62). In this application, GS1280 shows the most substantial advantage over the other systems (Figure 60). The reason is a substantial IP-links and memory bandwidth advantage on GS1280, as discussed in Section 3.5 (Figure 15). It is also interesting that the links show uneven utilization in Figure 62: East/West links show higher utilization than North/South links. This is because these profiles are taken on a 32-CPU GS1280 system, and link utilization is higher on horizontal than vertical links on a 4x8 torus.

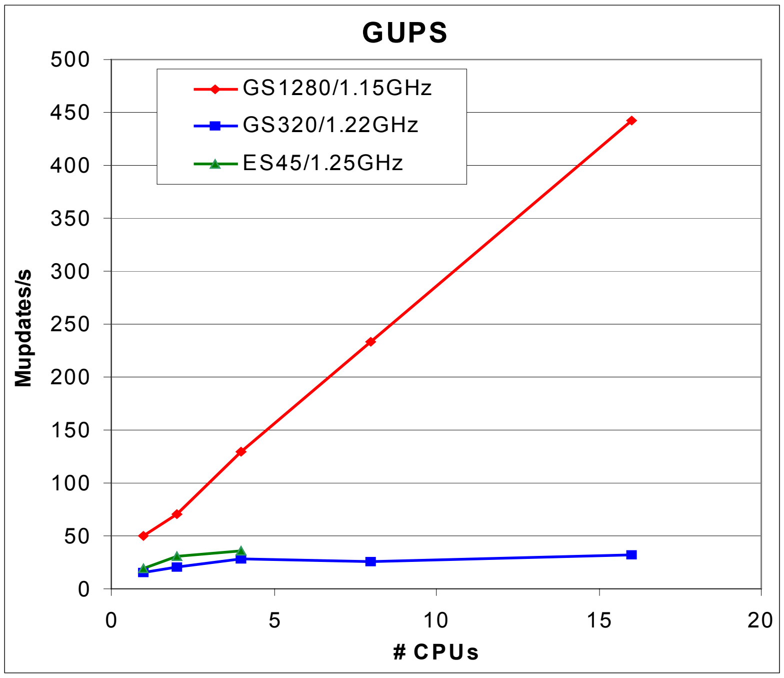


Figure 60. GUPS Performance.

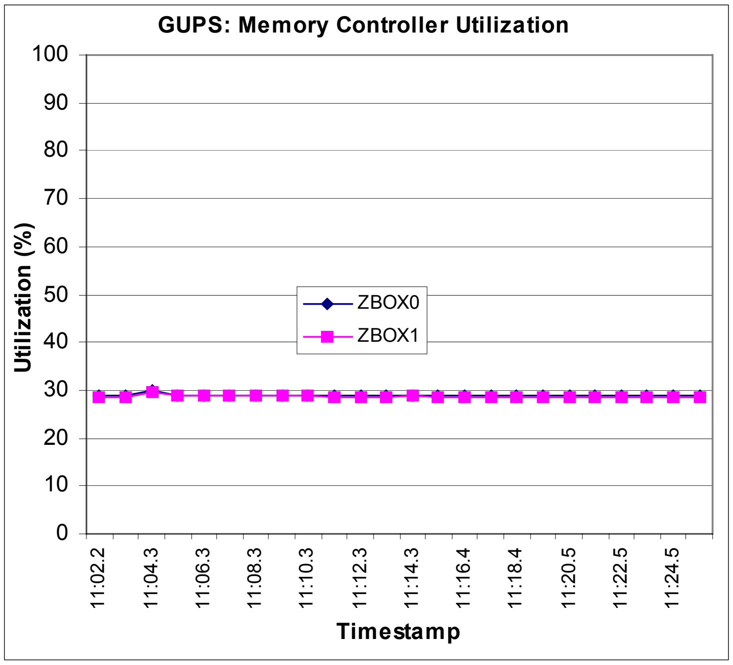


Figure 61. Memory Controller utilization in GUPS.

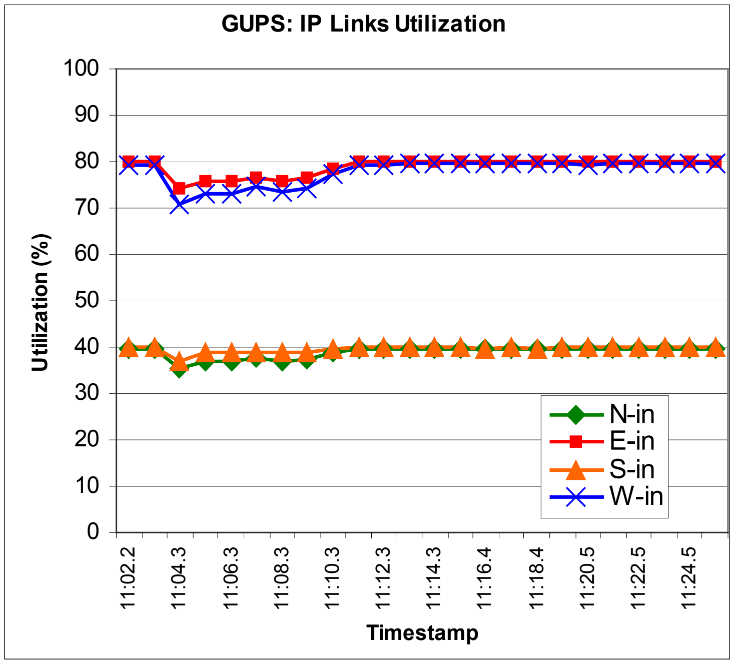


Figure 62. IP links utilization in GUPS.

4.4 Customer applications

Many customer applications are not tuned to fit well in the on-chip caches and thus show much significant stress on memory and IP-link bandwidth.

The examples below include several samples of customer applications: Structural Mechanics, Molecular Dynamics, and FFT. The data in Figures 63-65 shows 2-3 times GS1280 advantage vs. IBM/Power4, 2-5 times advantage vs. GS320, and 2-3 times advantage vs. ES45.

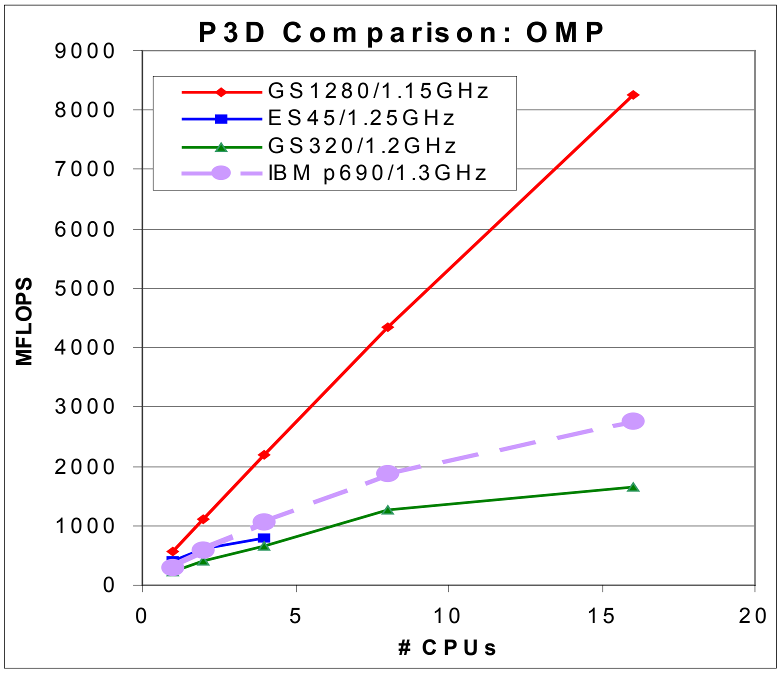
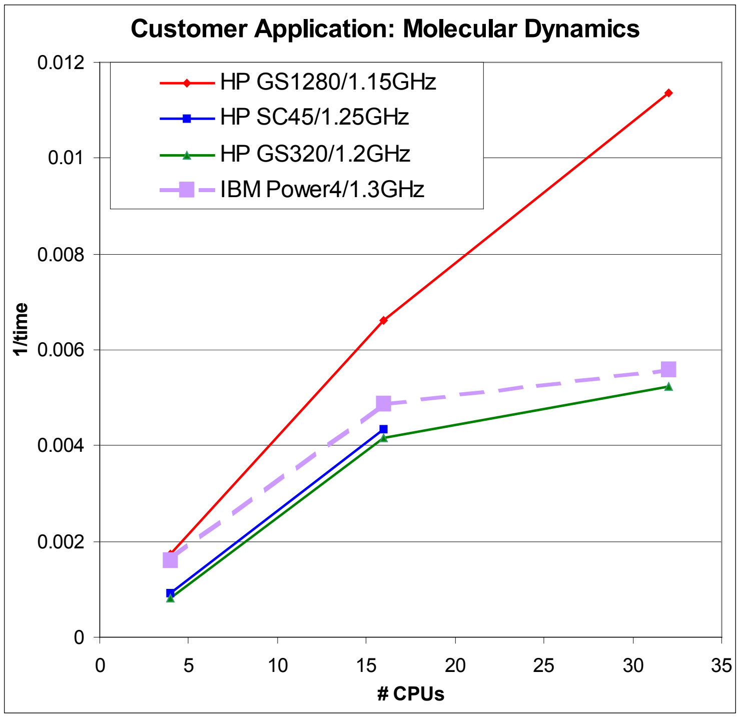


Figure 63. Structural Mechanics comparison.



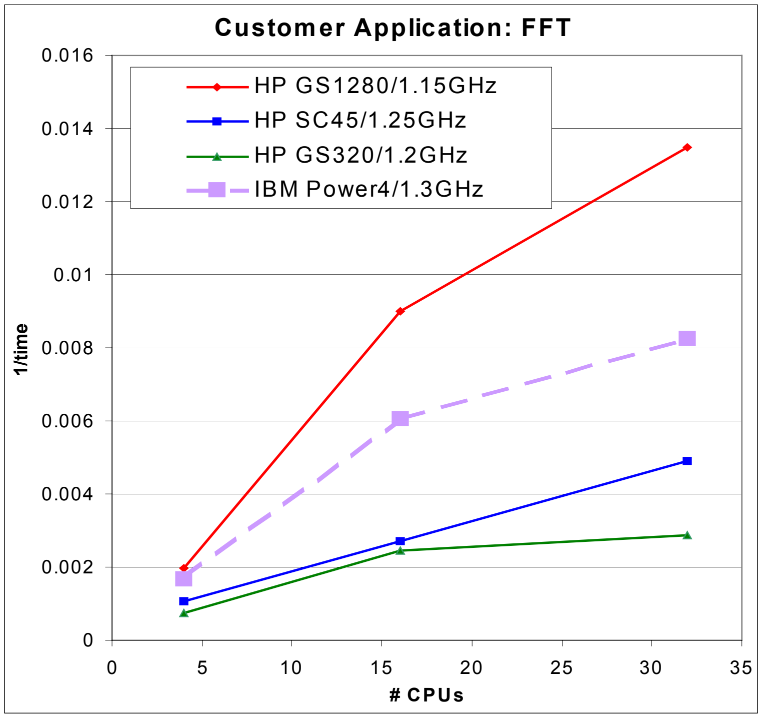


Figure 64. Molecular dynamics comparison.

Figure 65. FFT comparison.

5. Tools for analyzing performance of multiple-processor systems: Xmesh

Most of today's mainframe and supercomputer systems consist of multiple processors. In many cases, such systems include a large number of processors: 64 CPUs in HP/GS1280, 128 CPUs in HP Superdome, 32 CPUs in IBM/p690, 128 CPUs in SUN Fire. Performance analysis of such systems with growing numbers of CPUs is becoming more and more challenging as we face new design problems never encountered in smaller-scale systems. Frequently, system-wide performance is compromised by bottlenecks that develop in as few as one processor. These bottlenecks, called hot spots, can be very difficult to track down. They may be CPU, memory, interprocessor-link, or I/O-based bottlenecks, and sometimes even a combination. It is very important when studying these bottlenecks to not only detect and locate them, but also to distinguish between each of these causes. Being able to identify such bottlenecks is the first step in improving performance in large-scale multiprocessor systems.

Consider one example. Some systems have central interleaved memory, while others are CC-NUMA (Cache Coherent Non-Uniform Memory Access). In some CC-NUMA systems (such as GS1280), multithreaded applications use a First-Touch memory allocation policy. It is possible that memory is allocated to a single processor, and all other processors need to access memory from that processor, thus generating a memory hot spot. This greatly reduces performance of a multiprocessor, compared to the case where memory is uniformly distributed. Tools that can recognize such bottlenecks are crucial for identifying such limitations and improving performance in large-scale multiprocessors. Once such bottlenecks are recognized, various software/hardware techniques can be used to alleviate them. In addition, lessons learned in the current design can be applied toward designing more efficient large-scale multiprocessors in the future.

We have developed a tool called Xmesh that is used for monitoring and analyzing performance of large-scale multiprocessor systems. Xmesh is based on the built-in non-intrusive performance counters provided by the Alpha processor 21364 (EV7). Xmesh continuously reads counter information and processes it in a format that is uniform for all system hardware components -- percent of utilization. It then displays this information in real-time as a graphical representation of the torus network, which is the interconnection of CPUs on Alpha GS1280 systems (Figure 66). Within each of the numbered CPUs it reports the percent of utilization for the hardware components associated with it. The Xmesh reports utilization for the following hardware components: the CPU, the Zbox (two memory controllers), the RBox (router), the IO ports (ports 0, 1, 2, and 3), the I/O Down Hose, the I/O Up Hose, and the North, South, East and West Interprocessor ports on the EV7 processor. For improved readability, the status line shows a component's name and associated CPU number as the mouse is dragged over it. Figure 66 shows an example of Xmesh on a 16-CPU GS1280 system. Each CPU is represented by a box and interconnect links reflect the links and cables from a real GS1280 system (as read from the configuration registers). Within each CPU, different boxes represent utilization of the following components: CPU (center), two memory controllers (leftmost lower corner), four I/O ports (upper row), and two I/O hoses (sides). Since Xmesh is based on the built-in performance counters, the overhead of running it is very low (we observed less that 0.3% overhead on 16 CPUs).

A color spectrum indicates levels of utilization for each hardware component. There are 10 colors in the spectrum, ranging from purple, which indicates low utilization, to red, which indicates high utilization. The spectrum of colors can be adjusted to accommodate cases where utilization is low and falls within a single category. For example, the utilization of less-than 10% that falls in the purple category may be distinguished further by shifting the 10 spectrum color changes from 1 to 10% rather than the default 1 to 100%. In addition to colors, the tool allows utilization values to be displayed as well.

The Xmesh can be used to monitor hardware characteristics dynamically, while the system is running a particular application. However, it is not always feasible to watch the system interactively for extended periods of time (e.g. overnight, or over the weekend). In such cases, we are providing an extension of Xmesh, called Bmesh that allows storing Xmesh information in a file. The tool bplot is used to graphically view information from the file generated by the Bmesh (as illustrated in numerous figures in this paper).

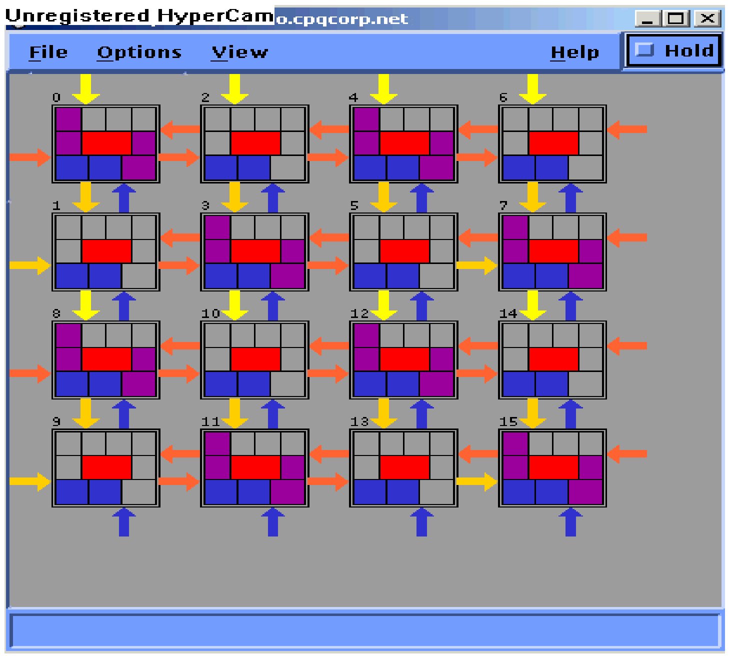


Figure 66. Xmesh for a 16-CPU GS1280.

We have used Xmesh extensively to characterize user applications that are memory, CPU, or I/O intensive. The intensity of stress on various system components (memory controllers, Interprocessor links, or I/O) can be quantified using Xmesh. For example, Figure 9 shows memory controller utilization for all SPECfp2000 benchmarks. This data helped us understand why some workloads show significant advantage on GS1280 vs. the other systems, while the others do not. For example, the workloads with high memory controller utilization as identified by Xmesh on Figure 9 (swim, applu, lucas, equake, mgrid) run substantially faster on GS1280 than other systems due to GS1280's memory bandwidth advantage.

In addition, the information obtained from the Xmesh tool can be used to monitor data flow between CPUs, determine resource bottlenecks, and watch for non-optimal performance. If Xmesh reveals that utilization factors are reaching maximum capacity and there are performance bottlenecks, a user can chose among several alternatives. For instance, if there is too much I/O targeted to one CPU's disks, a user can modify the I/O subsystem to spread I/O usage among several I/O subsystems. If Xmesh reveals contention for access to memory on a particular CPU, or perhaps a CPU is reaching its maximum of memory utilization, a user can apply Xmesh information to modify an application to run on a particular CPU, allocate memory to a particular CPU, increase memory capacity, or bind processes to a particular RAD (Resource Affinity Domain). This is often necessary in cases where processes migrate too often, causing a slow, detrimental memory migration, or also when a process is using a larger than ideal memory allocation, causing it to visibly (in Xmesh) "spill" into one or more adjacent CPUs. With Xmesh, a user can easily detect hot spots, where all traffic is targeted towards a single CPU (or a small number of CPUs). With first-touch memory allocation, one thread can touch memory that is needed by other threads. The first thread gets all data in its memory, and all other CPUs need to access memory from a single-CPU memory. The Xmesh tool has been extensively used to recognize uneven memory usage and optimize algorithms and user interfaces to improve memory locality (maximize the ratio of local to remote memory accesses), and thereby help ensure optimal performance on CC-NUMA hardware (as a part of Tru64 Version 5.1 enhancements).

The Xmesh tool is also helpful in qualification efforts where engineers can determine whether a particular workload drives each of the system components to saturation. The debug engineers have been using the tool to recognize performance problems. For example, an un-even utilization on the link due to non-optimized routing table was recognized using Xmesh and later corrected.

6. GS1280 Tuning Guidelines

6.1. Software tuning

In this section we provide a few tips for getting the best performance on GS1280.
Tru64 provided many important enhancements to support GS1280 architecture:

- VM managed big-page support (for text, data, shared memory, and anonymous memory)
- GS1280 mesh-aware VM allocation (memory allocations kept as near to the home RAD as possible)
- GS1280 mesh-aware multi-path I/O (I/O sent on the shortest path to the executing CPU)
- GS1280 mesh-aware scheduling (thread/task creations are kept close to the home RAD)

The following are a few tips that are worthwhile experimenting for improving performance on GS1280:

- Experiment with compiler switches. Tune for the target architecture with "-arch host" and "-tune host". This allows more aggressive prefetching that is better tuned for the 21364 processor (due to exceptional memory bandwidth). Try using "-fast", which includes these switches. If some tight inner loops are a bottleneck, try using "-unroll n". We have observed 4-5% improvement from compiler tuning on many workloads (20% in more tight loops).
- Bind processes to CPUs by either using “runon –r RAD” or a script that uses a dmpirun “process-file” for binding MPI processes. We’ve observed that in many applications binding provides a few percent improvement (higher improvement observed in some cases).
- When running OMP code on fewer than all processors, the O/S does an excellent job starting the threads on a group of nearby CPUs. However, it is sometimes beneficial to create a processor-set ("man processor_sets") to better specify the CPUs to use. For example, it is possible to gain performance by limiting CPU usage to the CPUs that form a square (thus minimizing the distance between cooperating CPUs). It may be simplest to create a processor-set containing all the CPUs you *don't* want to use, then run on the default pset (pset 0).
- Use sched_distance tunable to limit the number of hops a thread/task can be scheduled away from home.
- The SCS threads sometimes provide better performance than PSC threads in multithreaded applications.
- “Spiking” an image usually provides additional performance gains (particularly beneficial for workloads where no sources are available). We have observed over 20% gain in StarCD by using pixie/spike.
- Experimenting with shared memory allocation (striped or First-Touch) by modifying shm_allocate_striped can improve performance in some cases.
- Using Fortran prefetch directives to manually increase levels of prefetching is another area of experimentation (e.g. “use dfor$prefetch”)
- Experiment with memory refaulting. Sometimes a threaded application may know that memory lives in the wrong RAD, i.e., future accesses will predominantly come from a RAD other than the home RAD. In these cases it may be useful to explicitly migrate the memory to the desired RAD, or, in general, disassociate the pages from their current home RAD such that "first-touch" will move them to the desired RAD. An example of when this would be useful is when a threaded application uses a section of virtual memory with one access pattern and then subsequently will use it with a different access pattern. There are several routines in the libots3 library, which can be used to affect these changes. The most generally useful of these routines is _OtsMigrateNextTouch(start_address, end_address). After a successful call to this routine, the virtual memory in the specified address range loses its binding to any physical memory. A subsequent access of that memory will move the relevant page to RAD from which the memory is accessed.
- Use the latest versions of threads library.

- Experiment with big pages vs. small pages. This has the most significant performance impact of all tuning tips. We strongly suggest trying an application with both big and small page settings ("vm_bigpg_*" sysconfig settings allow dynamic reconfiguring of big/small pages). Applications that are likely to benefit from big pages usually make random (or large-stride) accesses to a large memory. Applications that are likely to degrade from big pages are mostly multithreaded applications.
  Big pages are beneficial in many applications, since they reduce the number of TB misses. We have observed 10-30% gain from big pages in many applications (up to 3.5 times gain in extreme cases). However, big pages can also be detrimental in other applications. The reason is that in many multithreaded applications data within one big page is shared by several parallel threads (while this is not the case with small pages). Using Xmesh tools, we have observed that in such cases, a single-CPU bottleneck (hot spot) is generated: a set of big pages is accessed by a single thread (First-Touch policy), and all other threads access data from that single CPU. Figure 67 indicates that within the same set of benchmarks (SPEComp2001), we have some that gain from big pages, and others that gain from small pages. The GUPS benchmark shows significant improvement from big pages for memory sizes between 2MB and 512MB (Figure 68).

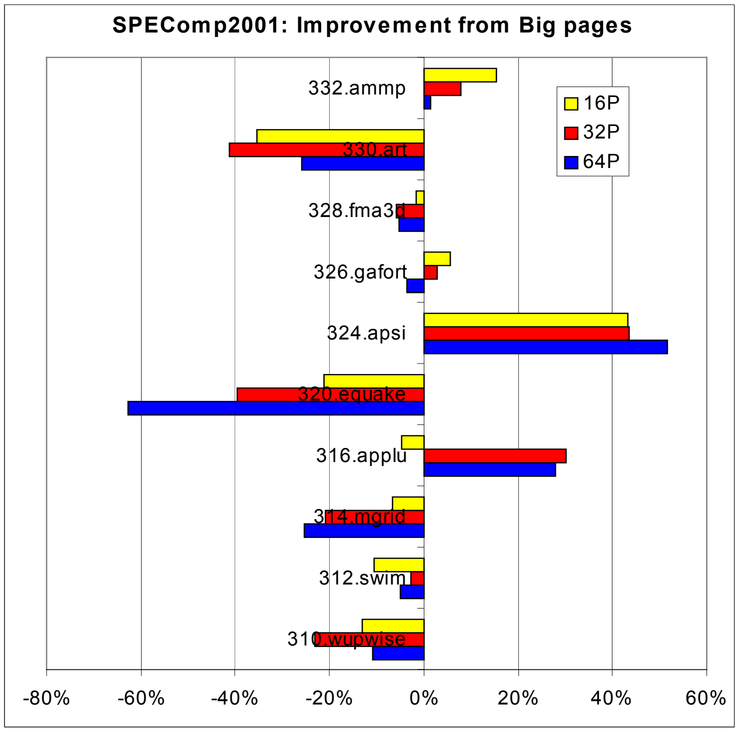


Figure 67. Performance effects from big pages in SPEComp2001.

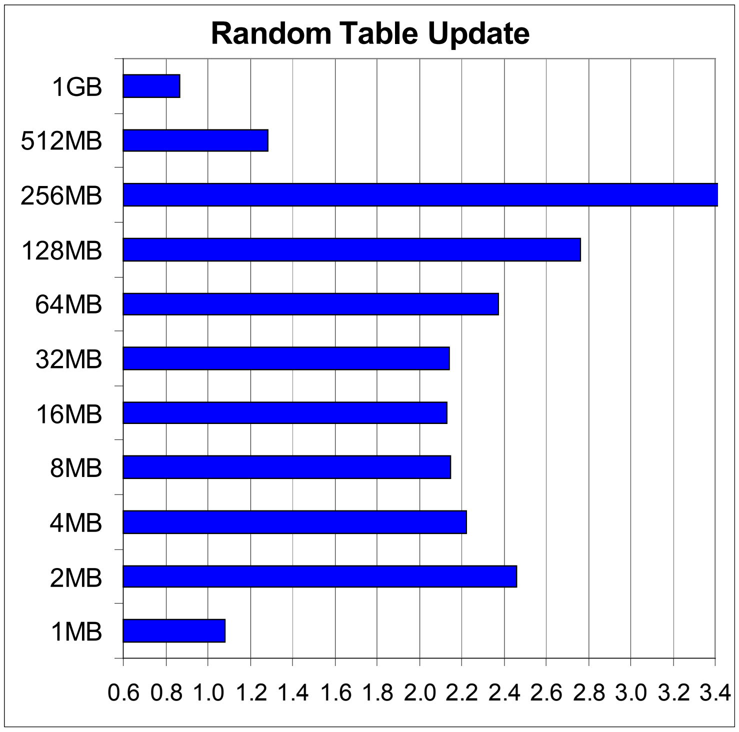


Figure68. Performance ratio (big/small) in GUPS.

6.2 Configuring GS1280 for best performance

In this section we describe how to configure GS1280 for best performance.
We include data on following:

- Performance differences between ES80 (ring) and GS1280 Model8 (GS1280).
- Performance effects from memory striping.
- Performance degradation from populating only one memory controller vs. two.
- Performance effects from 4GB vs. 1GB memory.
- Configuring I/O for best performance.

6.2.1 Performance differences between ES80 (ring) and GS1280 Model8 (GS1280)

The interconnect configuration in ES80 is the "ring" where all processors are connected in a ring configuration. The 8-CPU drawer in GS1280 uses a torus interconnect (Figure 69).

The performance degradation on ES80 vs. GS1280 is between 15-20% on a system where interconnect is not heavily used (Figure 70) and 20-50% on a system with heavily loaded interconnect (Figure 71). Note that the applications that do not stress IP links will not experience any performance difference between ring and torus.

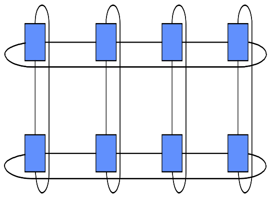


Figure 69. Torus interconnect.

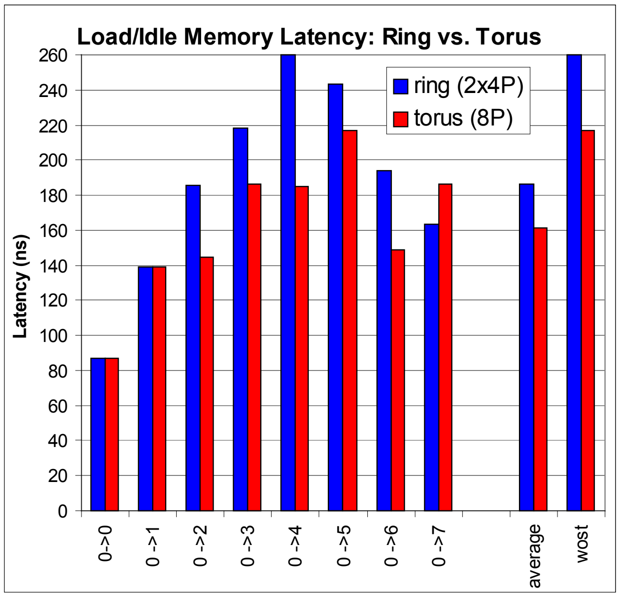


Figure 70. Latency comparison.

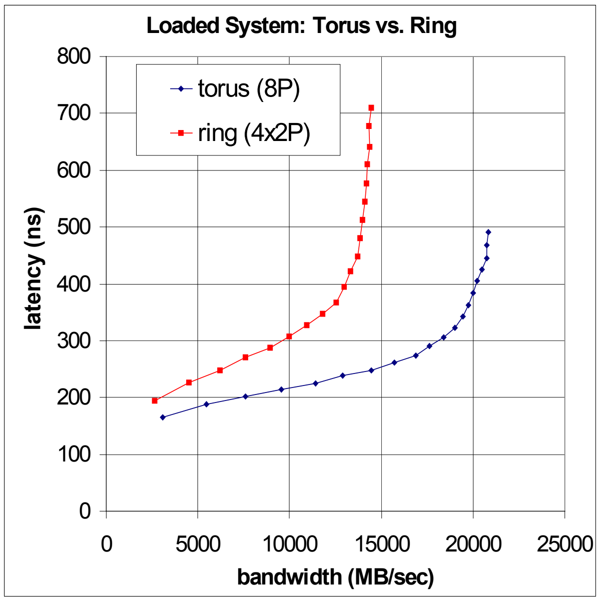


Figure 71. Bandwidh/latency comparison.

6.2.2. Memory Striping

Memory striping allows interleaving of 4 cache lines across 2 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 72 and 73. Figure 72 shows that striping degrades performance 10-30% in throughput (as well as MPI) applications due to increased inter-processor traffic (we observed degradation as high as 70%). Figure 73 shows that striping improves performance of a hot-spot traffic pattern (all CPUs read data from CPU0) up to 80%. The hot-spot traffic is recognized using the Xmesh tool (Figure 74). The tool indicates that the IP and memory traffic on the links to/from CPU0 (left corner) is higher than on any other CPU. We observed 30% improvement from memory striping 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). Therefore, the default setting for GS1280 is with striping turned off. We recommend that most users keep striping at the default value. We recommend turning memory striping on only in exceptional cases where Xmesh shows hot-spot traffic that cannot be eliminated using software techniques.

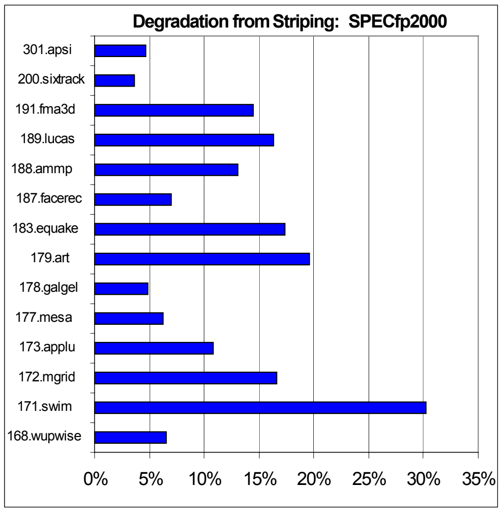


Figure 72. Degradation from striping.

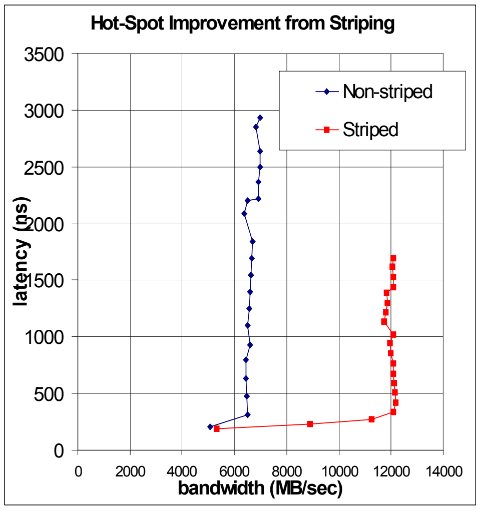
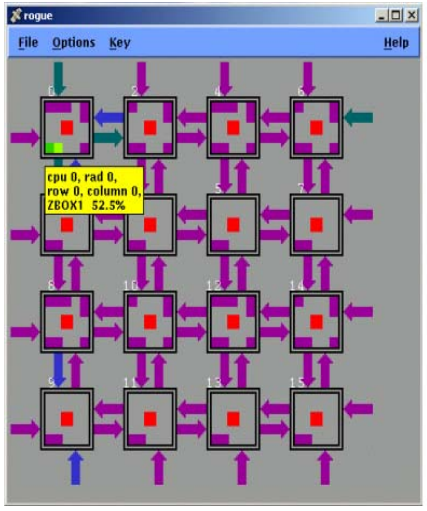


Figure 73. Improvement from striping.

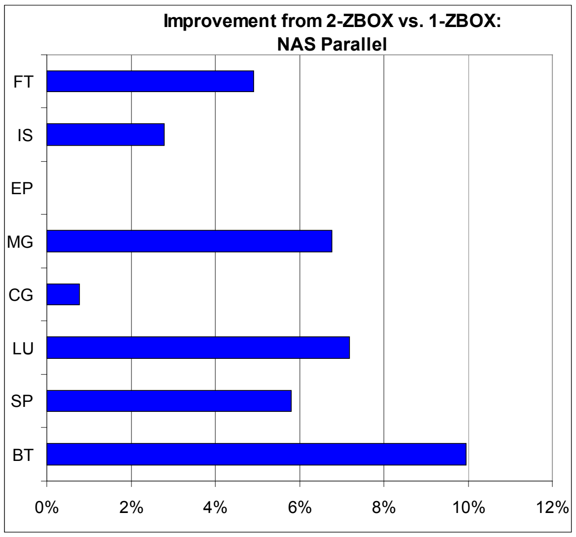


cpu 0, rad 0,
row 0, column 0,
ZBOX1 52.5%

Figure74. Xmesh with a hot-spot.

6.2.3 Configuring Memory for Performance

The EV7 processor has two memory controllers (ZBoxes). The GS1280 memory can be configured as follows: (1) partially populated memory with only one memory controller populated (5 RIMMs: 4 required and 1 optional for redundancy), (2) fully populated memory with both memory controllers populated (10 RIMMs: 8 required and 2 optional for redundancy). The 5th (optional) channel provides additional redundancy in case of failure on any of the channels. Our experiments show that the 5th channel does not introduce any performance penalty. Figure 75 shows performance improvement of fully vs. partially populated memory in NAS Parallel and SPECfp2000 benchmarks. Most NAS Parallel benchmarks are memory-bandwidth intensive (except for EP, Section 4.3.2). Several SPECfp2000 benchmarks stress memory bandwidth (Figure 9). The data in Figure 75 indicates that memory-bandwidth intensive workloads gain between 5% and 15% from fully vs. partially populated memory. In synthetic tests (as McCalpin STREAM benchmark), the improvement is much higher (50%).



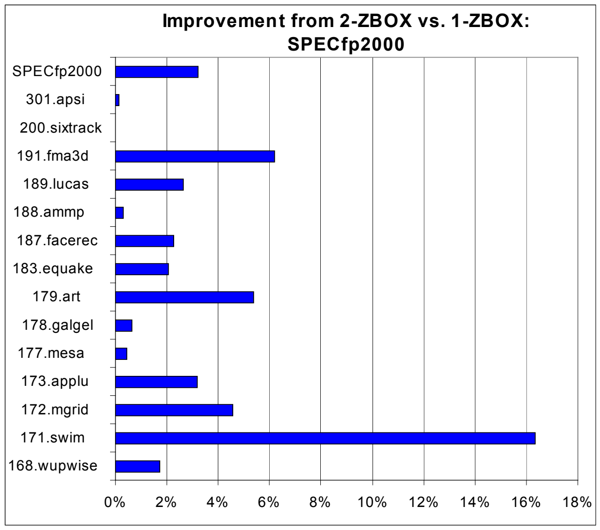


Figure 75. Improvement from 2 memory controllers in NAS Parallel and SPECfp2000.

The supported RIMM sizes in GS1280 vary from 128MB (1GB memory per CPU) to 512MB at FRS (4 GB memory per CPU). Note that larger memory will be supported after FRS. The 512MB RIMMs provide not only larger size, but also higher number of devices that helps reduce bank conflicts (16 devices with 512MB vs. 4 devices with 128MB RIMMs). Figure 76 shows that 4GB memory per CPU provides 2-4% improvement vs. 1GB memory. The gain is mainly due to higher number of devices (not memory size). The improvement is higher (12%) in STREAM.

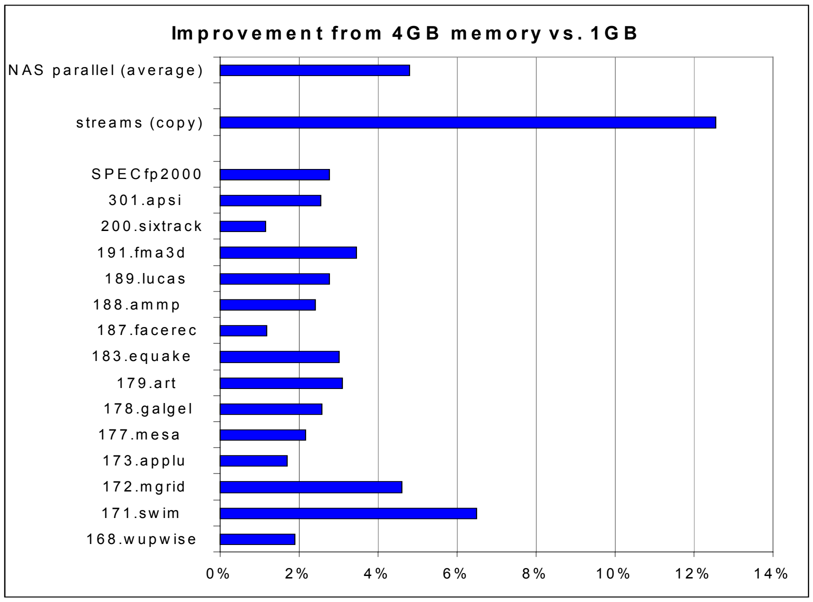


Figure 76. Improvement from 4 GB vs. 1GB memory.

6.2.4 Configuring I/O for Performance

The GS1280 system can be configured such that each processor has its own I/O drawer. This configuration provides the highest I/O bandwidth (measured 1 GB/s per IO7, Section 3.6). However, for applications that do not demand such high bandwidth, GS1280 can be configured with lower number of IO drawers (minimum one I/O drawer on the entire system). In the case where fewer I/O drawers than CPUs are used, it is important that I/O is configured such that the overall latency is minimized (e.g. checkerboard pattern cases where there are twice as many CPUs that IOs). An example of a 32-CPU GS1280 with 8 I/O drawers is shown in Figure 77.

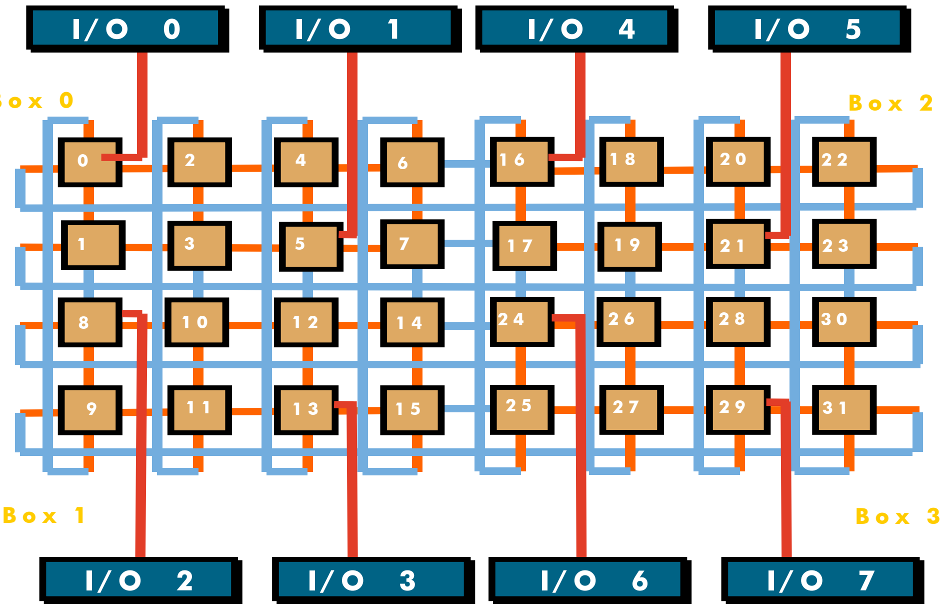


Figure 77. An example of 32-CPU GS1280 with 8 I/O drawers.

The high-performance I/O (Xshelf) can be used to further maximize delivered I/O bandwidth (as illustrated in the Figure 78). Additional details on configuring I/O for performance can be found in [29].

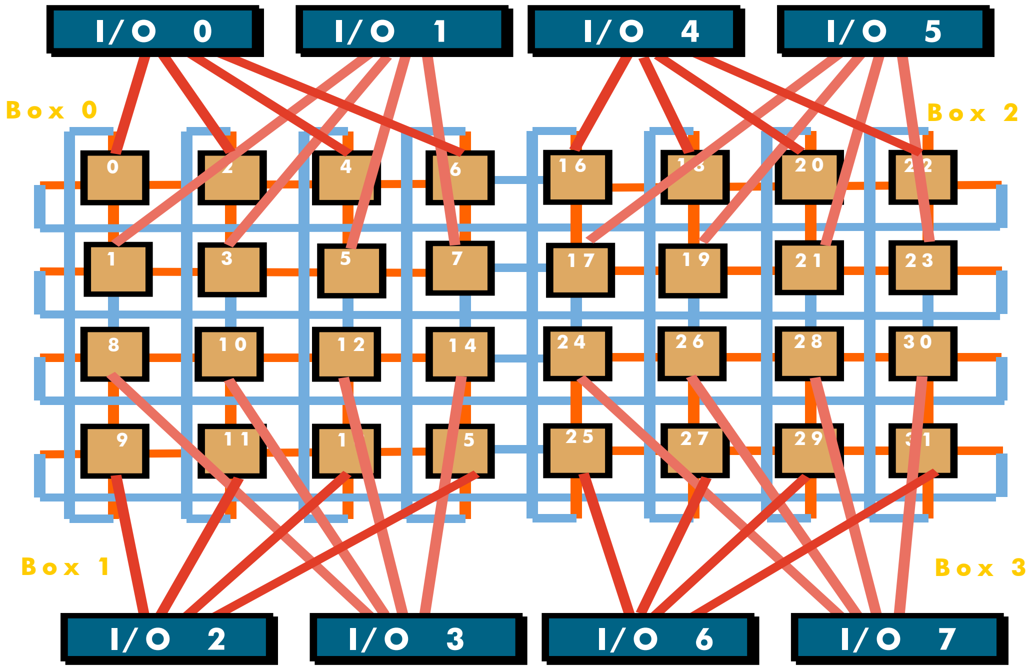


Figure 78. High-performance I/O in 32-CPU GS1280.

7. Summary and 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) an on-chip L2 cache (smaller than the off-chip L2 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. It places on a single chip all components that previously required an entire CPU module. Our data shows that this is a superior design for building large-scale multiprocessor systems. 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 a 2-4 times performance advantage of GS1280 vs. the previous-generation AlphaServer GS320 in these types of applications (e.g. NAS Parallel, StarCD). Efficient local-memory high-bandwidth memory design makes GS1280 an excellent match for throughput applications (e.g. superior performance and scaling in SPECfp_rate2000 and Abaqus/Throughput). 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. GS1280 provides and excellent NUMA support in both hardware and software for parallel and commercial applications (e.g. SPEComp2001, Dyna/SMP, Transaction Processing and Java applications). We observed the highest performance advantage of GS1280 vs. the previous-generation AlphaServer GS320 is in applications that stress memory bandwidth and IP links the most (e.g. 11 times in GUPS). Since Alpha 21364 preserved the same core as the previous-generation Alpha 21264 (and the CPU clock speed is comparable), the applications that are blocked to fit well in the on-chip caches perform comparably on GS1280 and GS320. In some cases, applications can take advantage of the larger 16MB cache, and therefore run faster on GS320 than on GS1280.

The GS1280 system shows a 20-30% advantage vs. IBM/Power4 Regatta system in majority of applications evaluated here. In almost all applications, GS1280 shows better scaling than IBM due to lower contention for memory (e.g. SPECfp_rate2001: 40% on 32 CPUs than vs. 10% advantage on 1 CPU). In floating-point workloads that run close to the peak speed (e.g. Linpack NxN), Alpha platforms show a disadvantage compared to the most other-vendor platforms.

We have heavily relied on profiling analysis based on built-in performance counters (Xmesh) throughout this study. 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.

We determined that some applications benefit from big pages (applications that randomly access large memory benefit the most), while others degrade (many parallel applications on 16 and 32 CPUs). Since it is difficult in all cases to determine which applications will benefit from big pages and which will not, we recommend running an application both ways (system should be booted with big pages, and small pages should be set dynamically using sysconfig). Also, binding processes to CPUs helps improve performance in some cases. Re-compiling with EV7 tuning switches can also be beneficial.

We determined that an IP ring topology in ES80 provides performance degradation (20-30%) compared to a torus only in applications with poor memory locality. Similarly, using 16 CPUs on a 32-CPU system will result in lower performance than on a dense 16-CPU system (due to a loss of wrap-around links). We determined that memory striping is beneficial only for applications that generate a hot spot. However, memory striping degrades performance of most applications. Therefore, our recommendation is to keep striping set to OFF (default value). We also determined that fully populated memory provides 5-15% performance gain in memory-bandwidth intensive workloads compared to a half-populated memory. The 4GB memory per CPU also provides additional gain compared to 1 GB memory.

The GS1280 provides flexibility in configuring I/O depending on application requirements (from a maximum of one I/O per CPU to a minimum one I/O on entire system). The measured raw-device (no files system) I/O Read bandwidth is 1 GB/s per I/O drawer. We observed that I/O bandwidth scales linearly as additional I/O drawers are added. The GS1280 provides substantial improvement in I/O bandwidth compared to ES45 (3 times) and GS320 (8 times). When using fewer I/O drawers, it is important to configure them in such a way that overall latency in the mesh is minimized (e.g. checkerboard pattern).

Acknowledgments

Many teams contributed to the results shown in this paper. This was a true cross-organizational team effort.

- **Characterization Team**: Jason Campoli, Peter Gilbert, Andrew Feld, Brian Rerecich, and Andrew Whitlingum. I would like to thank my team members for outstanding job in producing performance data shown here and for the Xmesh tools that allowed us to gain more insight into the system behavior.
- **Marvel team**: thanks to Darrel Donaldson, Glenn Herdeg, San Duncan, Dave Hartwell, and Ted Gent. Special thanks to Darrel Donaldson for his advice, guidance, and support throughout this project.
- **EV7 team**: thanks to Pete Bannon, Brian Lilly, Maurice Steinman, and Dave Webb for providing insights into the EV7 architecture.
- **Performance team**: thanks to John Shakshober for TPC-C data, Brush Bradley for SPEC2000, Chris Beer for SPECjbb, and Sean Reilly for the I/O configuration presentation.
- **HPTC**:
- **Dave Field's ISV team**: thanks to Chuck Newman for PowerFLOW and Abaqus data, help with Fluent tuning, and thorough review of the paper, Pari Rajaram for Nastran and help with Marc, Susan Devere for Amber, Richard Readings for Gaussian98, and Dave Field, Sharon Shaw, Olivier Schreiber, and Yih-Yih Lin for Itanium2 data.
- **John Kreatsoulas' HPTC benchmarking team**: thanks to Greg Gaertner for SPEComp2001, Hans Leichtl for Dyna, Ray Hookway for Life-Science workloads, Dick Foster for GUPS, Enda O'Brien for StarCD, and John Eck for NAS Parallel.
- **Mark Gantly's team**: Miles Deegan for NWchem, and Eamonn O'Toole for Gaussian IBM data.
- **Customer support**: thanks to Jean-Marie Verdun, Patrick Grogniet, Frederick Ciesielski, Ghislain De Jacquelot, and Dominique Gillot for their feedback on customer applications.
- **Tru64:** Jim Woodward, Dave Bernardo, and Ross Knepper for providing insights into the Tru64 tuning.
- **Management:** thanks to Steve Jenkins, Subhash Dandage, Sas Durvasula, and Jack Zemcik for supporting this work.
- **Marketing:** thanks to John Bennett and Ed Turkel for their feedback.

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, R. Kessler, “Performance Analysis of the Alpha 21264-based Compaq ES40 System”, ISCA 2000
[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] SPEC2000 Benchmarks (Manassas, information available at http://www.specbench.org
[9] Information about the lmbench available at http://www.bitmover.com/lmbench/
[10] The STREAM benchmark information available from the University of Virginia, at http://www.cs.viginia.edu/stream
[11] Z. Cvetanovic, P. Gilbert, J. Campoli, “Xmesh: a graphical performance monitoring tool for large-scale multiprocessors”, submitted to the HP Technical Conference 2003
[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] Richard Kessler, “EV7 System Design Specification”, Internal HP document, August 2002.
[17] TPC-C data available at http://www.tpc.org/information/results_spreadsheet.asp
[18] NASTRAN data available at http://www.mscsoftware.com/support/prod_support/nastran/performance/v01_dmp.cfm
[19] Abaqus data available at http://www.abaqus.com/products/products_performance.html
[20] Marc data available at http://www.marc.com/
[21] PowerFLOW data (password protected) available at http://www.exa.com/exasource/
[22] Pallas information available at http://www.pallas.com/e/products/pmb/
[23] StarCD data available at http://www.cd-adapco.com/support/bench/aclass.htm
[24] LS-Dyna information available at http://www.arup.com/dyna/applications/crash/crash.htm
[25] NWchem data available at http://www.emsl.pnl.gov:2080/docs/nwchem/nwchem.html
[26] MM5 data available at http://www.mmm.ucar.edu/mm5/mm5-home.html
[27] Amber data available at http://www.amber.ucsf.edu/amber/amber7.bench4.html
[28] Linpack data available at http://www.netlib.org/performance/html/PDSreports.html
[29] Sean Reilly “Configuring the Next Generation HP AlphaServer Systems and Their I/O for Performance and Availability, HPETS October 2001http://habanero.mro.cpqcorp.net/aspg/pres.html