Performance of Variant Memory Configurations for Cray XT Systems

Wayne Joubert, *Oak Ridge National Laboratory*

ABSTRACT: *In late 2009 NICS will upgrade its 8352 socket Cray XT5 from Barcelona (4 cores/socket) processors to Istanbul (6 cores/socket) processors, taking it from a 615 TF machine to nearly 1 PF. To balance the machine and keep 1 GB of memory per core, NICS is interested in reconfiguring the XT5 from its current mix of 8- and 16-GB nodes to a uniform 12 GB/node. This talk examines alternative memory configurations for attaining this, such as balancing the DIMM count between sockets of a node vs. unbalanced configurations. Results of experiments with these configurations are presented, and conclusions are discussed.*

KEYWORDS: Cray XT5, memory subsystem performance, DIMM configurations, memory bandwidth, latency, benchmarking

1. Introduction

Recent trends in microprocessor design have led to non-power-of-two core counts for processor dies, due to layout and power consumption issues. Examples include AMD Istanbul (6 cores) and Intel Dunnington (6 cores).

The use of irregular core counts for multicore processors brings complications to implementing memory configurations for these chips. DIMM memory bank configurations may have restrictions on the size and number of DIMMs allowed in each bank—restrictions which do not fit well with processors having irregular core counts.

In 2009 the National Institute for Computational Sciences intends to upgrade its Kraken Cray XT5 system from AMD Barcelona processors (4 cores/socket) to AMD Istanbul processors (6 cores/socket).

The purpose of this study is to evaluate memory configurations for the new system which will maintain at least 1 GB of memory per processor core, a requirement for targeted applications, and also will provide good memory subsystem performance, within the allowable hardware requirements of memory unit configurations.

2. The Cray XT5 Compute Node

Each Cray compute blade consists of four compute nodes. In turn each compute node is composed of two processor sockets, each with its own memory. In particular, each socket is associated with four DDR-2 DIMM memory slots, arranged into two banks of two DIMM slots each. Each processor's Northbridge memory controller accesses the socket's DIMMs directly. Furthermore, each processor can access the other socket's DIMMs in a NUMA fashion through a HyperTransport link. See Figure 1.

Restrictions exist on how each CPU socket's four memory slots may be populated. In particular, the two slots of each bank must either both be empty or both be populated with DIMMS of the same memory capacity. Other configurations are not officially supported.

Processor sockets are populated with AMD Opteron processors. For this study, the primary processor is the AMD Barcelona quadcore CPU with 2.3 GHz clock speed, 4x64 KB L1 data cache, 4x512 KB L2 cache and 2 MB shared L3 cache. Memory modules considered include DDR2-800 4 GB DIMMs, DDR2-667 2 GB DIMMs and DDR2-533 8 GB DIMMs.

Experiments are performed on the Oak Ridge National Laboratory "Chester" platform, a Cray XT5 test and development system with 448 compute cores, with the same hardware and software configuration as the ORNL NCCS JaguarPF Cray XT5 platform.

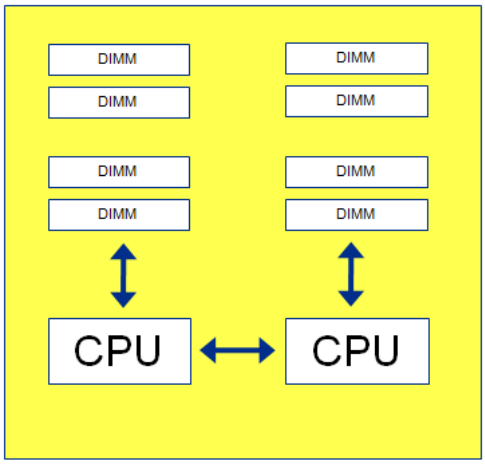

Figure 1. Cray XT5 Node Memory Architecture.

3. Design of Experiments

For these experiments, nodes of the Chester platform are reconfigured with the desired memory layout, and benchmarks are run on the reconfigured nodes. Table 1 shows the set of memory configurations considered. The notation "A-B-C-D, E-F-G-H" is used to represent the memory capacities, in GB, of the eight DIMM modules on the node, where "A-B-C-D" are the slots for the first socket, "A-B" are the slots for the first bank of the first socket, and so forth.

Configuration	Description
4-4-0-0, 4-4-0-0	2 GB/core, balanced
4-4-0-0, 2-2-0-0	1.5 GB/core, unbalanced
2-2-2-2, 2-2-0-0	1.5 GB/core, unbalanced
2-2-2-0, 2-2-2-0	1.5 GB/core, balanced
4-4-2-2, 4-4-2-2	3 GB/core, balanced
8-8-0-0, 8-8-0-0	4 GB/core, balanced

Table 1. Experimental Memory Configurations

This table indicates the average memory per core for the quadcore Barcelona processor. For the six-core Istanbul upgrade, the memory per core will be reduced correspondingly, leaving at least 1 GB per core in each case.

For each configuration, the following questions are to be addressed:

- What is the effective memory bandwidth and latency?
- Particularly for unbalanced configurations, how much performance penalty is there for accessing off-socket memory?

The latter question is of particular significance for large-memory applications that are memory bandwidth limited. A typical use case is for the application to allocate equal amounts of memory for each processor core. For large-memory cases under this regime on unbalanced configurations, this necessitates overflowing on-socket memory and accessing off-socket memory. It is desirable to understand what performance penalty may result from this.

To test memory performance, the following codes are considered:

- The STREAM benchmark. In particular, the triad operation z = y + a x from this benchmark is used.
- A DAXPY kernel y = y + a x.
- The LMBENCH benchmark, to measure memory latency.
- The S3D application code. This is a petascale combustion application based on 3-D structured grids which typically has memory-bound performance.

4. Experiments: Memory Spillage Effects

This battery of experiments uses a Chester node with a standard 4-4-0-0, 4-4-0-0 or 8-8-0-0, 8-8-0-0 balanced memory configuration to understand the impact of executing a code on one compute socket and letting memory allocations spill to the other socket's memory. This is done by gradually increasing the problem size until all node memory is used. See Figure 2.

Figure 3 shows results for the STREAM triad benchmark, executed on 1 to 8 compute cores, using the 4-4-0-0, 4-4-0-0 memory configuration. Here the peak node memory bandwidth is 25.6 GB/sec theoretical, 21.2 GB/sec actual. One can clearly see a drop in memory bandwidth performance for very long vectors that span across all of node memory, a drop of up to 15%.

It should be noted that for this system, memory pages are assigned based on a Linux first-touch policy, and the structure of STREAM array initialization puts array elements z(i), x(i) and y(i) all on the same page (and thus

the same socket) for a given value of i. Thus at any given time, a core is accessing either all on-socket or all off-socket memory.

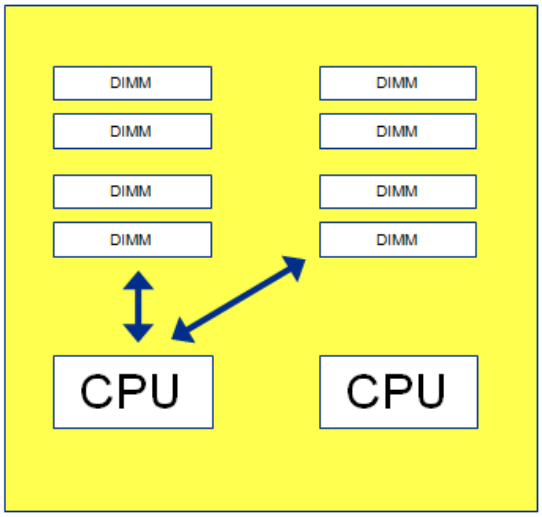

Figure 2. Off-Socket Memory Reference.

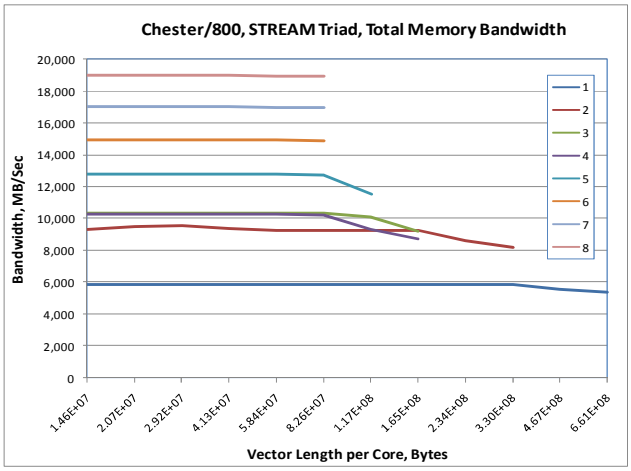

Figure 3. STREAM Triad Performance.

Figure 4 gives results for the same experiment except that array initialization is changed from interleaved to sequential, so that all of array z is touched, then all of y, then all of x. Thus for given i, each of z(i), x(i) and y(i) are on different memory pages and, potentially, on different sockets' memory. The results show that for off-socket memory references, in some cases there is an uptick in memory bandwidth performance. This is due to one processor concurrently accessing both on-socket memory through its memory controller and off-socket memory through HyperTransport. In some situations this technique could be used to boost memory performance for an application. However, it is not helpful for the common use case in which all cores of the node are accessing memory at the same time.

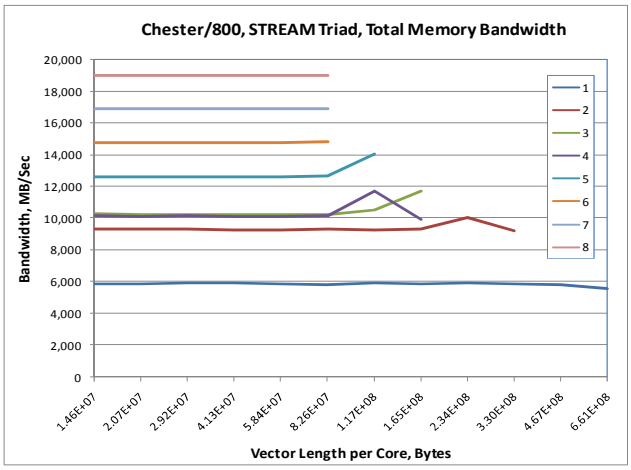

Figure 4. STREAM Triad Performance.

Figure 5 shows similar results for a DAXPY kernel with sequential array initialization. The performance shows a similar uptick in performance as on- and off-socket memory is accessed concurrently by a single processor.

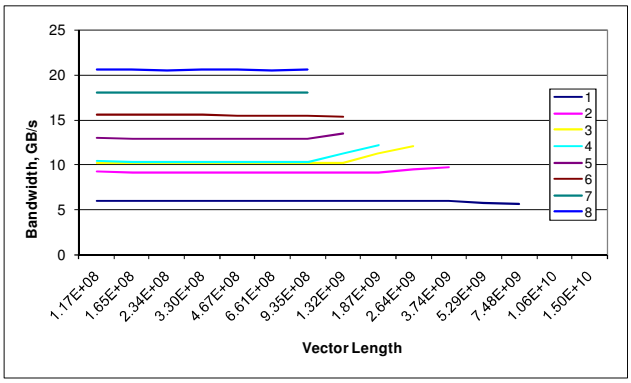

Figure 5. DAXPY Performance.

Figure 6 shows performance of the S3D application for the 4-4-0-0, 4-4-0-0 (DDR2-800) and the 8-8-0-0, 8-8-0-0 (DDR2-667) configurations. S3D is executed either on 1 socket (4 MPI tasks) or 2 sockets (8 MPI tasks), and the gridcell count per core is gradually increased. The wallclock time per gridcell per core is graphed. The circled area indicates run cases for which off-socket memory is used. One can see a slight increase in runtime for these cases, though the amount of increase is small. It should be noted that for these cases, S3D spends about half of its time in memory references, thus it is significantly memory intensive.

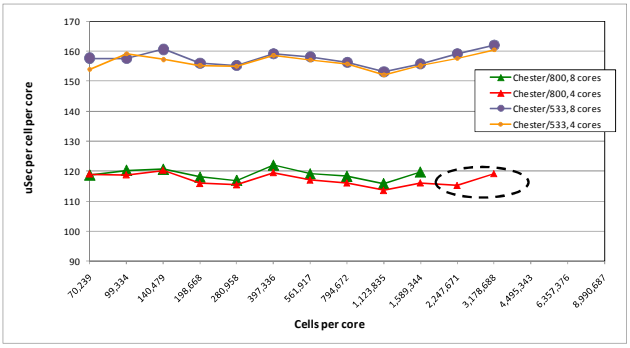

Figure 6. S3D Performance.

5. Memory Configuration Experiments: S3D

In this section we consider the impact of different memory configurations on S3D performance, for runs using all 8 cores of a compute node. For the baseline case, we take the 4-4-0-0, 4-4-0-0 balanced configuration. The time per gridcell for this case is expected to be flat—see Figure 7.

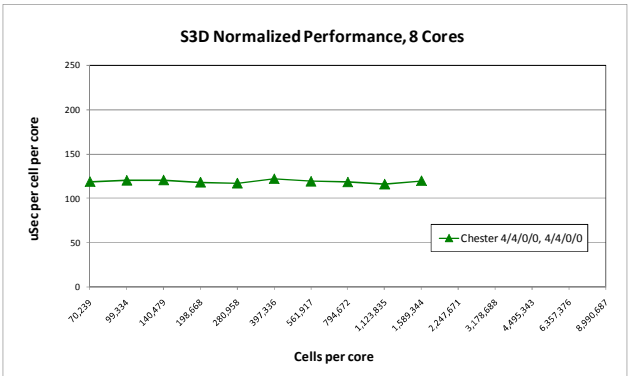

Figure 7. Baseline S3D Performance.

Figure 8 shows S3D performance for the unbalanced 4-4-0-0, 2-2-0-0 configuration. The performance is slightly worse overall than the baseline case. However, the run case for which off-socket memory is used is only slightly worse than the case of using all on-socket memory. Thus the impact of unbalanced memory on performance is small.

Figure 9 shows S3D performance for the unbalanced 2-2-2-2, 2-2-0-0 configuration. The performance is 6% slower than the baseline case overall and 17% slower when off-socket memory is used. It is unclear whether the overall slightly slower performance is primarily due to differences in the DIMMs or due to the processor's memory controller behavior.

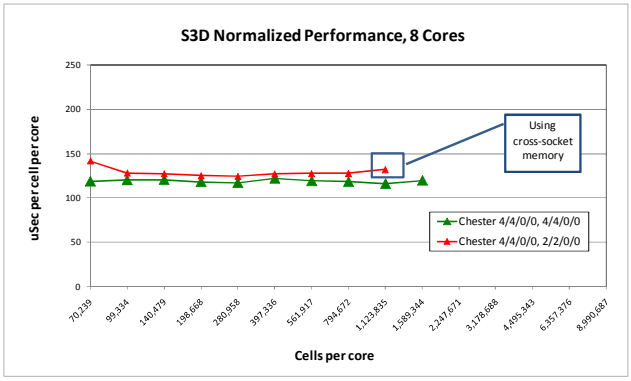

Figure 8. S3D Performance, 4-4-0-0, 2-2-0-0.

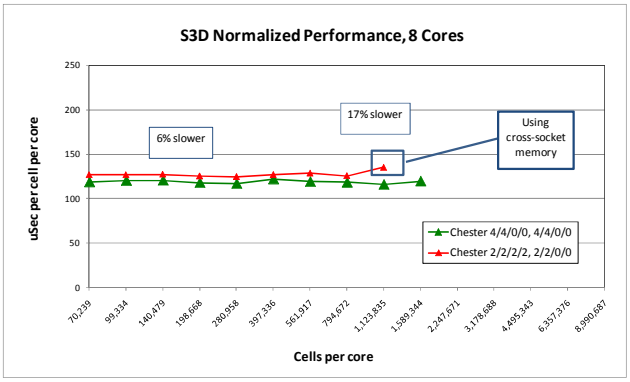

Figure 9. S3D Performance, 2-2-2-2, 2-2-0-0.

Figure 10 shows S3D performance for the balanced 2-2-2-0, 2-2-2-0 configuration. It should be noted that this configuration is unsupported, since some DIMM banks are only half-filled. Performance for this case is considerably worse, up to 39% slower. The likely explanation of this behavior is that the memory controller performs striping of allocated memory across the two DIMMs of the memory bank; thus, performance of the bank is cut in half for a half-populated bank.

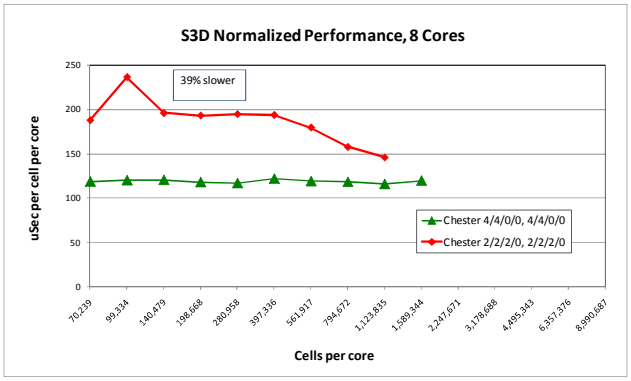

Figure 10. S3D Performance, 2-2-2-0, 2-2-2-0.

Figure 11 shows S3D performance for the balanced 4-4-2-2, 4-4-2-2 configuration. Performance is very slightly worse than but almost identical to the baseline case.

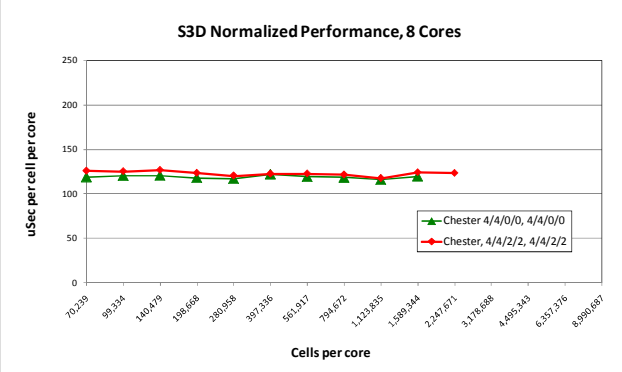

Figure 11. S3D Performance, 4-4-2-2, 4-4-2-2.

6. Memory Configuration Experiments: LMBENCH

The LMBENCH benchmark measures array load latency as a function of array length. For these experiments, the benchmark is run on a single core, and the array size is increased to span all of node memory.

Figure 12 shows performance for the baseline case of 4-4-0-0, 4-4-0-0. One can clearly see the effects of L1/2/3 cache as well as the increase in latency as the array accesses spill off-socket.

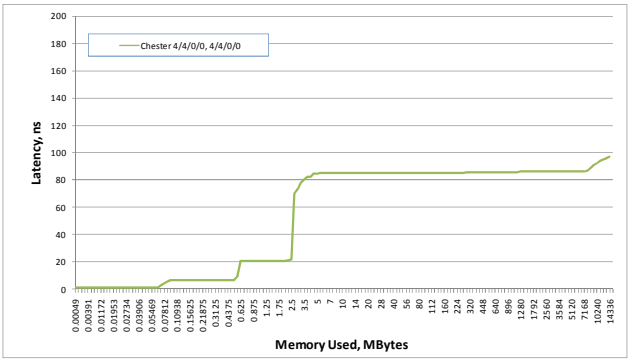

Figure 12. LMBENCH Performance, Baseline Case.

Figure 13 shows LMBENCH performance for several memory configurations. For the unbalanced case, the experiment is run alternatively from each CPU socket on the node, to show differences in performance between the two sockets due to the memory asymmetry. Notably, the performance for all cases is nearly identical. Differences for very long vector lengths are due to the fact that for the different cases, the sockets have different amounts of memory, so memory spillage begins to occur at different vector lengths. For detail of this behavior, see Figure 14, where the points of spillage for each case are shown. The on-socket latency for all cases is 85 ns, and the extrapolated off-socket latency for all cases is in the range of 102-108 ns.

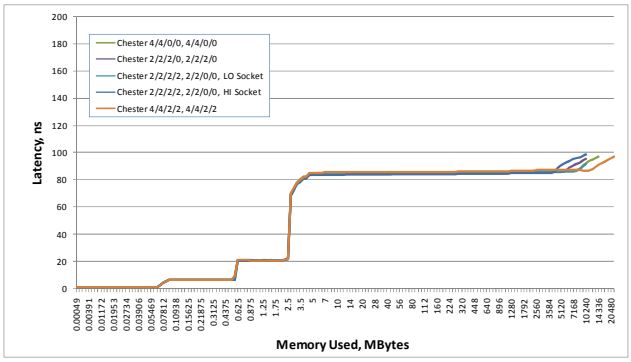

Figure 13. LMBENCH Performance.

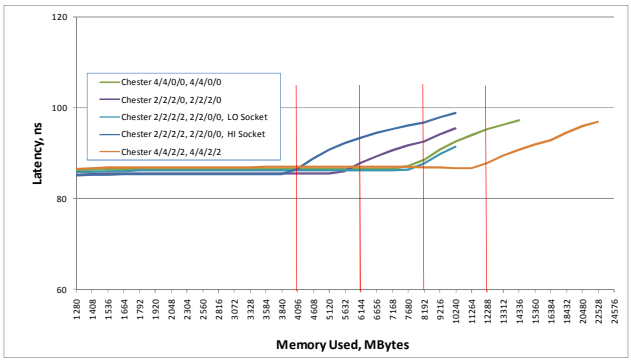

Figure 14. LMBENCH Performance, Detail.

7. Conclusions

The conclusions from these experiments are as follows:

- The impact of referencing off-socket memory for unbalanced memory configurations is less than expected, less than 20% for the example application used.
- The unsupported memory configuration performs very poorly due to incompatibility with the memory controller's striping policy.

- Different memory configurations have very little effect on memory latency.
- In some rare cases an application might benefit from using on- and off-socket memory concurrently, though for the typical application use case this is of no benefit.

These results argue in favor of the viability of unbalanced memory configurations. For applications for which the balance of memory is important, the impact of the memory not being balanced is relatively small. Furthermore, for applications that do not push the limit of node memory for their use cases, or for applications that are not memory-intensive, use of off-socket memory does not affect application performance to an appreciable degree.

Acknowledgments

The author would like to thank Ben Bales, Jeff Larkin, Don Maxell, Brian Waldecker and Terry Watts for comments and assistance.

About the Author

Wayne Joubert is Senior Computational Scientist at the National Center for Computational Sciences at Oak Ridge National Laboratory. He specializes in performance optimization, numerical methods, numerical linear algebra, linear solver algorithms and software, high performance computing and performance analysis. He can be reached at Oak Ridge National Laboratory, 1 Bethel Valley Road, MS-6008, Oak Ridge, TN 37831 USA, joubert @ ornl.gov.