Performance Effects of Node Mappings on the IBM BlueGene/L Machine

Brian E. Smith[1] and Brett Bode[2]

[1] IBM Rochester, 3605 Highway 52 North, Rochester MN 55901
smithbr@us.ibm.com
[2] Ames Laboratory, 329 Wilhelm Hall, Ames IA 50011
brett@scl.ameslab.gov

Abstract. The IBM BlueGene/L supercomputer consists of up to 65536 compute nodes connected by several networks including a three-dimensional torus. The BlueGene/L control system allows a user to re-map MPI ranks to different physical torus coordinates at run-time. Effects of node mapping on application performance are investigated for Gray-code mappings with differing aspect ratios, permutations of the X, Y, and Z coordinates, random mappings and four new mapping types. Results are presented for three NAS parallel benchmarks - BT, CG, and MG - on 128-way partitions in co-processor mode and virtual node mode on the prototype BlueGene/L hardware.

1 Introduction

Scientific application developers are constantly looking for ways to get better performance from their code to allow them to solve larger problems. One relatively easy method of increasing performance is to ensure that the system running the application is the best system available. For example, if the application code assumes a certain network topology, running on a different network topology would not provide the best performance benefits.

The IBM BlueGene/L supercomputer allows a user assign physical torus coordinates to logical MPI ranks. While not changing the physical topology, this logical re-arranging allows a user to make more assumptions in the code or port code from other platforms with less work. And, performance of existing codes might be improved by providing a more optimal node map for a given application. This paper looks at the performance benefits of several new mapping strategies and several existing mapping strategies on three of the NAS parallel benchmarks.

2 BlueGene/L Overview

2.1 The IBM BlueGene/L Hardware

The BlueGene/L supercomputer is a new parallel system from IBM [1],[2] . The system consists of up to 65,536 relatively modest compute nodes. Each node

J.C. Cunha and P.D. Medeiros (Eds.): Euro-Par 2005, LNCS 3648, pp. 1005–1013, 2005.
© Springer-Verlag Berlin Heidelberg 2005

consists of two PowerPC 440 cores - each with two 64-bit floating point units, five network controllers. Each node has a total of 512 MB of memory. The three user-level networks are a three dimensional torus for high-performance point-to-point message passing, a global combining/broadcast tree for high-speed collective operations, and a global interrupt network for extremely fast barriers. The other two networks for low-level machine control and access to external file servers.

2.2 IBM BlueGene/L Software

The compute nodes run a custom, lightweight kernel. The kernel provides a single address space to one running application. Therefore, there is no context switching overhead and applications can use the majority of the 512 MB of memory. The kernel function-ships system calls to the I/O nodes and handles application generated signals. For more information on the BG/L software, see [3].

The standard application programming interface (API) for BG/L application developers is MPI. Specifically, BG/L uses an optimized MPICH [4] implementation where many collective operations are optimized to make use of the underlying BG/L hardware.[5] Code is compiled with the IBM XL compilers and the system is rather "Linux-like" in supported system calls.

2.3 IBM BlueGene/L Control System

The BG/L control system is responsible for booting the system and running jobs. There are two main operating modes. The standard mode is called co-processor mode. In this mode, one core is responsible for most computations while the second core is used for message passing. The other operating mode is called "virtual node mode" (VNM). In this mode, all of the physical resources of a node are split in half between the two cores. Since there are half as many packet send/receive queues per core, there can be performance degradation on communications-intensive applications. VNM can be very beneficial to applications that are not memory intensive, nor communications-bound.

3 Mappings

The BG/L control system pieces work together to provide compute processes their MPI rank based on their physical coordinates in the torus. For example, a node at physical coordinates $(0, 0, 0)$ would become MPI rank 0. A node at $(1, 0, 0)$ would then become MPI rank 1. In virtual node mode, a fourth coordinate is required (the processor or core ID, usually referred to as 'T'). A user may specify an arbitrary mapping file (as long as all nodes in a partition are uniquely accounted for in the file, even if they are not used). This allows for some interesting possibilities for improving application performance. For example, many applications might assume a two-dimensional mesh of processors. Given a proper

mapping file, the three-dimensional BG/L torus can be made to look like a two-dimensional mesh with no link contention. This paper looks at the performance effects of the standard permutations of X, Y, and Z coordinates (XYZ, XZY, YXZ, YZX, ZXY, ZXY. In virtual node mode, the core ID can come first or last), along with Gray-code mesh mappings, and two other unique mapping strategies with several variations. A random mapping was also used to compare results.

3.1 Gray-Code Mappings

When hyper-cubic machine topologies were more common in the early 1990s, work was done on embedding different order hyper-cubic graphs in whatever dimension hypercube was available on the physical hardware of a given machine. See [6] or [7]. This was done using Gray codes [8]. Gray code sequences are usually constructed using a binary reflected Gray code algorithm. For example, the sequence for 8 values (3 bits) would be: 000, 001, 011, 010, 110, 111, 101, 100. For a simple discussion on embedding a lower-order graph in a higher order topology, see [6]. The general strategy is to realize that a k-way partition can be written as $k = 2^n$ for some n. Each node in the partition then has an n-bit "address". If each node exists in a three-dimensional torus, then the n-bit address can be broken down into three sub-addresses (x-bits, y-bits, z-bits) for the physical coordinates. To embed a two dimensional mesh in the three-dimensional torus, the three sub-addresses are split into two sub-addresses with m-bits and n-bits for location information. This is shown with an example. The 128-way partitions on the BG/L development machine are 8x4x4 nodes. Each node can then be represented as $n = x_1x_2x_3y_1y_2z_1z_2$. To embed a 16x8 mesh we convert each node "address" n to look like $n = a_1a_2a_3a_4b_1b_2b_3$. For example, a Z-first Gray-code mesh would have $n = z_1z_2x_1x_2x_3y_1y_2$ to then convert to a mesh "address". The T Gray-code meshes have the core ID first.

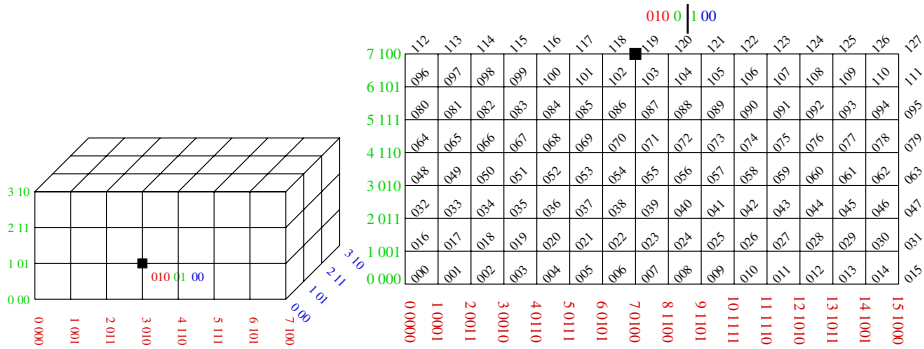

Fig. 1. Embedding 16 by 8 mesh in 128-Way BG/L Partition

The utility developed to generate all of the maps also allows the user to specify the dimensions of the mesh to control the aspect ratio of the 2D mesh.

3.2 Other Mapping Strategies

Because the Gray-code mesh mappings move some nodes away from each other that would normally be neighbors, another map type called "unfolding" was developed to try to keep more neighbor nodes next to each other while creating a mesh-like mapping. Ranks increase along the orientation axis, then increase in a perpendicular axis and back-track along the orientation axis. This continues until all nodes are mapped.

In virtual node mode, three different ways of counting the core ID - corefirst (CF), corelast (CL), and coreplanes (CP) were investigated. In corefirst unfolds, MPI rank 0 corresponds to physical coordinates $(0, 0, 0, 0)$ while MPI rank 1 corresponds to physical coordinates $(0, 0, 0, 1)$.

In corelast unfolds, the ranks progress along the orientation axis, then move to the second core and progress back along the orientation axis before increasing the next orthogonal coordinate.

In coreplane unfolds, the ranks increment along the orientation axis, then increase on the orthogonal axis, and continue increasing as the physical coordinate decreases. When the physical coordinate is back at 0, the processor core is increased and the process repeats. These three mappings are shown in 2 for two rows from an 8x8x8 torus.

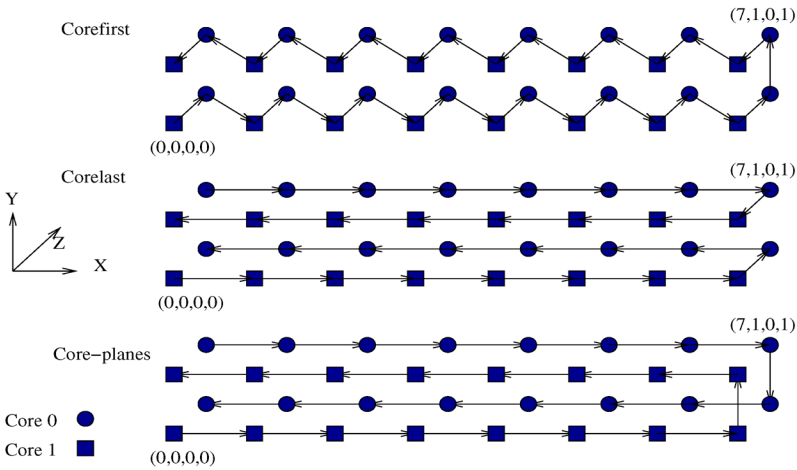

Fig. 2. VNM Unfolding Types

4 Procedure

In Sect. 5, results from three NAS benchmarks are presented. The three benchmarks included show the most variability between mappings. They are BT (block tri-diagonal), CG (conjugate gradient), and MG (multi-grid). For more information on the NAS benchmarks see [9], [10], and [11]. For the results of other NAS benchmarks (and other benchmarks in general), see [12].

The NAS benchmarks come in multiple problem sizes called classes. Results are shown for classes A, B, and C. Class A is the smallest class available for parallel machines and tends to be communications-bound on larger partitions such as 128-way. Class C was the largest class that still showed variation between mappings on 128-way partitions (in co-processor or virtual node mode), though there is a class D version of most of the NAS benchmarks. Results are presented for 128-way BG/L partitions in co-processor mode and virtual node mode (therefore utilizing 128 or 256 processors).

The result reported by the NAS benchmarks is "MOp/s" (millions of operations per second). The graphs in Sect. 5 show the MOp/s divided by number of processors used. Larger values of MOP/s indicate better performance. The NAS BT benchmark requires a square number of processors so only 121 processors are used for the co-processor mode results. However, the results get reported as if 128 processors were actively used. Therefore the MOp/s per processor values are recalculated for only 121 nodes.

The benchmarks were all run on first generation prototype hardware and were compiled with the significantly less efficient (compared to IBM XL) GNU compilers. Because of this, the numbers are not meant to represent absolute BG/L performance. Instead, they are meant to show the relative effects of mappings on performance. Re-running the benchmarks with the XL compilers should lessen computational time without affecting communications so the results should show differences between mappings more clearly.

5 Results

Typically, a smaller problem class performs fewer MOP/s per processor so it is possible to plot all three classes on the same graph to conserve space. However, a few smaller problem classes out-performed or performed similarly to larger class problems for a given mapping. This obscures the larger problem results. This is especially true in the CG benchmark.

Figure 3 shows the results for the NAS BT benchmark in co-processor mode for classes A, B, and C.

The most variation between mappings is seen in the class A runs. This is probably because less time is required for computation in the small problem sizes. Because the 128-way partition is physically arranged as 8 by 4 by 4 nodes, the mappings that hide the latency from the fact that there are an average of twice as many hops in the X direction as Y or Z show reasonably good performance increases. For example, the YZX and ZYX stock mappings show approximately a fifteen percent improvement over the default XYZ mapping. Similarly the X-first Gray-codes have the X coordinates as the most significant bit so the physical X location changes slowly through the MPI ranks. The unfold Y map counts along the Y axis, then Z before incrementing X. Finally, the Blocks - X mapping also increases the X coordinate most slowly. Figure 4 shows the results of running the CG benchmark with the mappings and Fig. 5 shows the results from MG with 128 processors in co-processor mode. Both results are somewhat

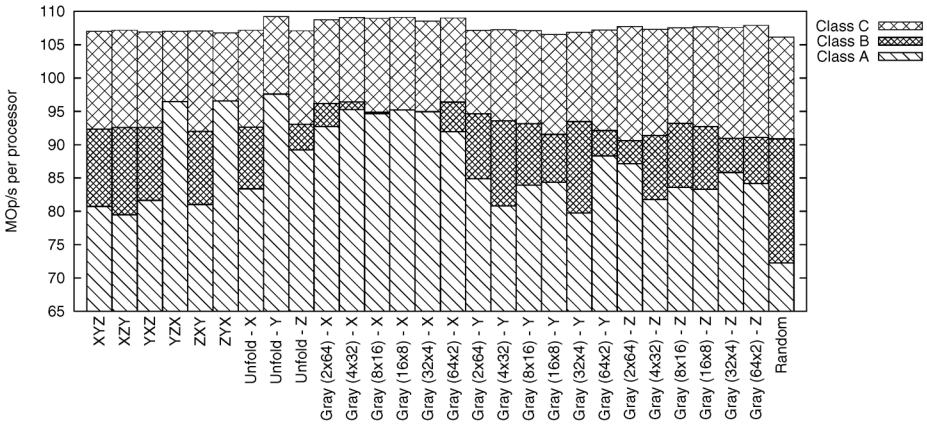

Fig. 3. BT 128 Processors

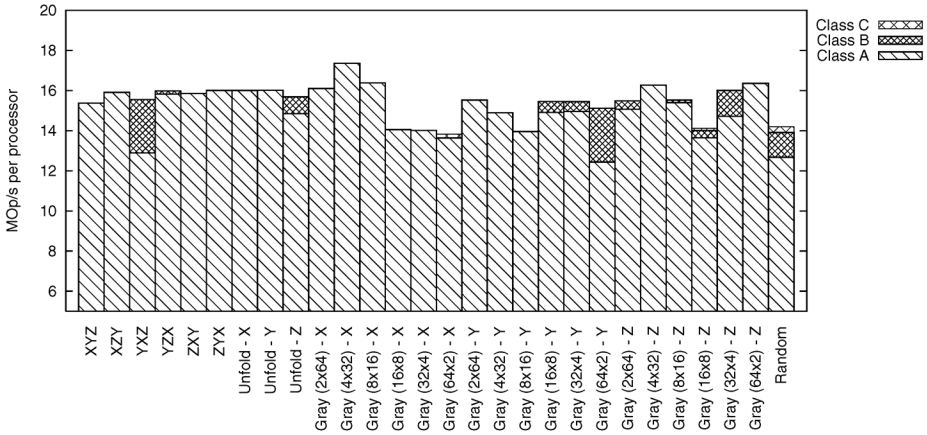

Fig. 4. CG 128 Processors

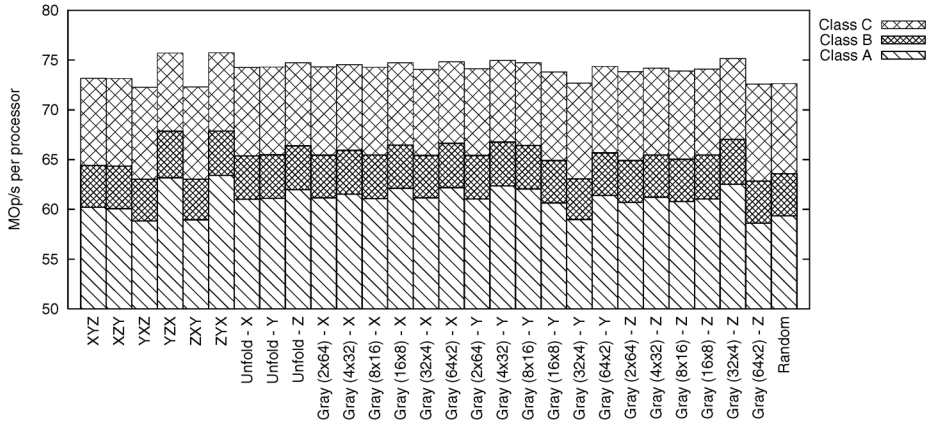

Fig. 5. MG 128 Processors

similar in that there was not much variation between the mappings, especially MG. The MG results are similar to the BT results - the mappings that have the X coordinate varying the slowest had the best performance.

Figure 6 shows the BT results with a 128-way partition booted in virtual node mode. Because BT assumes a square mesh of processes, it makes sense that the square meshes are the best Gray-code performers. As in the co-processor case, the maps that hide the latency of twice as many hops in X showed the best performance in general. Figure 8 shows the MG results. As in the 128-way case, there is less variation than with BT. There was very little variation between the Gray-code mappings as a function of aspect ratio. The virtual node mode unfold maps oriented along Y performed well since those mappings cause the physical X coordinate to vary slowly. Figure 7 shows the results for CG. The results were somewhat unexpected. Gray-code meshes that were more rectangular tended to perform better than the square meshes. However, the unfold and stock results were similar to the BT and MG results. The Gray-code mapping results will require more investigation of the CG code.

6 Conclusions

Changing the MPI rank mapping on BlueGene/L can show very good performance increases (as much as fifteen percent over the default XYZ mapping). This is especially true if the mapping hides the latency from extra network hops in asymmetric partition sizes. The majority of BlueGene/L partition sizes are not perfect cubes so there is potential for performance increases with minimal effort for most configurations.

Changing the logical node mapping is also an easy thing for an application developer to try. At the very minimum, the BG/L control system provides the stock permutations to the user, and developing mapping files is not difficult.

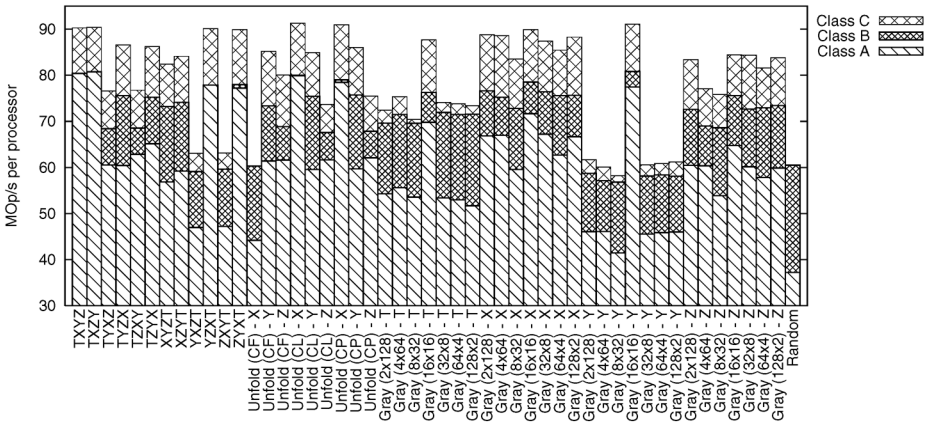

Fig. 6. BT 256 Processors

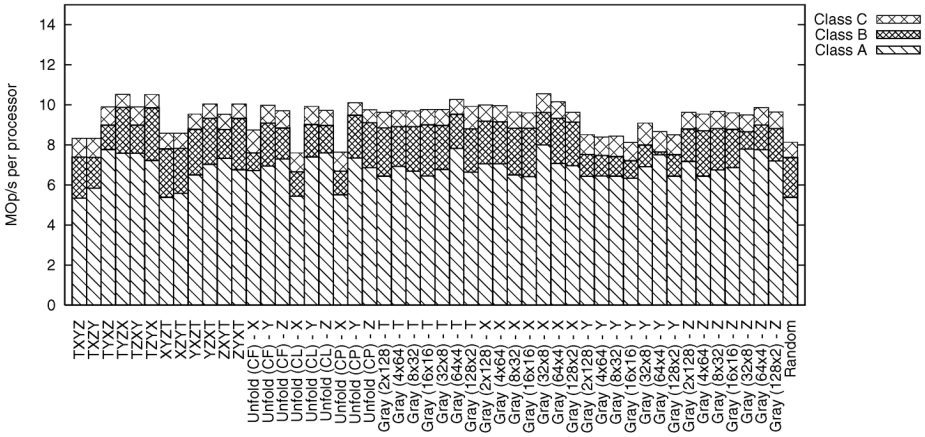

Fig. 7. CG 256 Processors

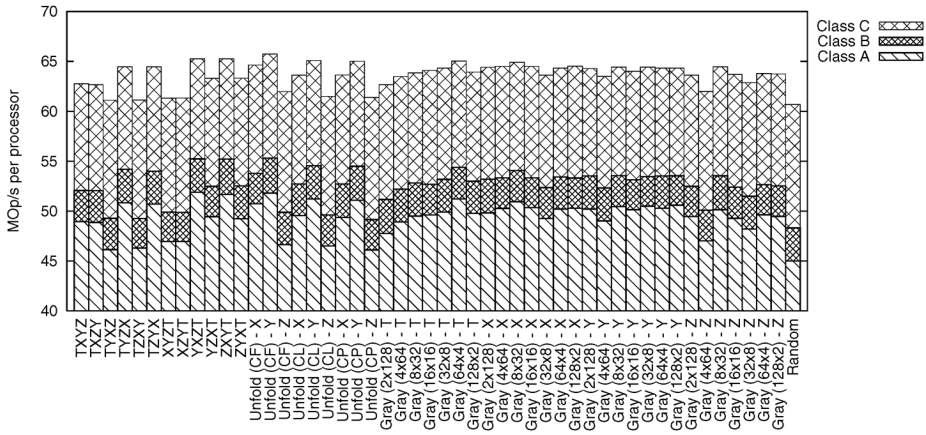

Fig. 8. MG 256 Processors

Generally, mappings that can reduce or hide latency from extra physical distance from node to node show the best improvement. The performance benefits should be even better on larger configurations (unfortunately, many of the NAS benchmarks do not run on large partition sizes to test this) For example, the full 4096-way prototype system at IBM Rochester is thirty-two by eight by sixteen nodes. On that system, mappings that decrease the X coordinate slowest should perform very well.

Even in extremely powerful supercomputers, it is important to get as much performance as possible from the given hardware and software. Node mappings is one area where very little effort is required for possibly large gains in application performance.

Acknowledgments

This work was supported in part by U.S. Department of Energy. This manuscript has been authored by Iowa State University of Science and Technology under Contract No. W-7405-ENG-82 with the U.S. Department of Energy. The authors would also like to thank IBM for access to BlueGene/L hardware, especially Sam Ellis, Charles Archer, and Dr. José Moreira.

References

1. The IBM BlueGene/L Team. An overview of the BlueGene/L supercomputer. In *Supercomputing 2002*, 2002.
2. F. Allen et al. Blue gene: A vision for protein science using a petaflop supercomputer. *IBM Systems Journal*, 40(2), 2001.
3. Gheorghe Almási et al. An overview of the BlueGene/L system software organization. *Parallel Processing Letters*, 13(4):561–574, 2003.
4. William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, 1996.
5. Gheorghe Almási et al. Implementing MPI on the BlueGene/L supercomputer. In *Proceedings of Euro-par 2004*, 2004.
6. Youcef Saad and Martin H. Schultz. Topological properties of hypercubes. *IEEE Transactions on Computers*, 37(7):867–872, 1988.
7. M. Y. Chan and F. Y. L. Chin. On embedding rectangular grids in hypercubes. *IEEE Transactions on Computers*, 37(10):1285–1288, 1988.
8. James R. Bitner, Gideon Ehrlich, and Edward M. Reingold. Efficient generation of the binary reflected Gray code and its applications. *Communications of the ACM*, 19(9):517–521, 1976.
9. Rob F. Van der Wijingaart. NAS parallel benchmarks version 2.4. Technical report, NASA Advanced Supercomputing Division, Ames Research Center, Moffett Field, CA 94035-1000, 2002. NAS-02-007.
10. D. Bailey et al. The NAS parallel benchmarks 2.0. Technical report, NASA Advanced Supercomputing Division, Ames Research Center, Moffett Field, CA 94035-1000, 1995. NAS-95-020.
11. D. Bailey et al. The NAS parallel benchmarks. Technical report, NASA Advanced Supercomputing Division, Ames Research Center, Moffett Field, CA 94035-1000, 1994. NAS-94-007.
12. Brian E Smith. Performance effects of node mapping on the IBM BlueGene/L machine, June 2005.