Enabling Contiguous Node Scheduling on the Cray XT3

Chad Vizino
vizino@psc.edu
Pittsburgh Supercomputing Center

ABSTRACT: The Pittsburgh Supercomputing Center has enabled contiguous node scheduling to help applications achieve better performance. The implementation and application performance benefits will be discussed.

KEYWORDS: scheduling, pbs, cpa, xt3, job placement, optimization, graphical monitor.

1.0 Introduction

The Pittsburgh Supercomputing Center (PSC) has implemented a flexible job scheduling environment to enable jobs to be placed onto its Cray XT3 system in ways that can help them achieve better performance.

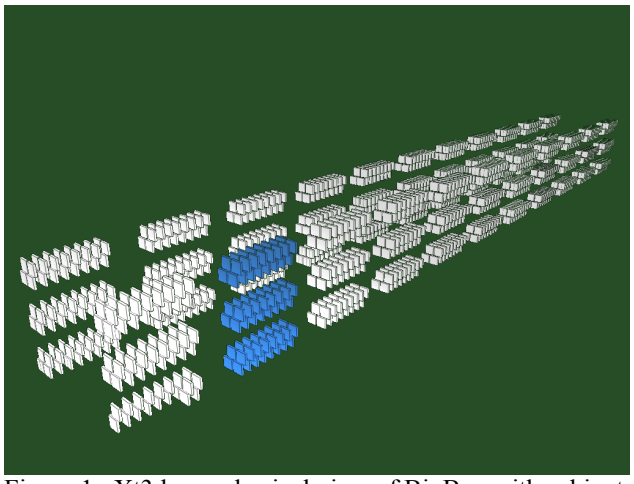

Figure 1. Xt3dmon physical view of BigBen with cabinet nodes highlighted.

An overview of past work will be provided, discussing changes made to the batch system, work on a graphical machine monitor to help visualize jobs placed on the system, and node assignment changes to help minimize communication overhead to all jobs.

Finally, a summary of work done to accommodate applications that are topology-aware (know in advance how they should be placed on the system) and the system changes that have been made to enable them will be discussed.

2.0 Review of Past Work

PSC's Cray XT3 system, called BigBen, contains 2090 dual core nodes of which 2068 are compute nodes and 22 are service nodes. The system is physically configured in two rows of 11 cabinets each. The interconnect topology is a torus in three dimensions: 11x12x16.

On BigBen, PSC runs Catamount on the compute nodes and uses a customized version of PBS Pro 5.3 software [1] along with a custom scheduler called Simon[2]. These changes will be briefly reviewed.

Other changes to the environment including a graphical monitor, and job placement observations will also be discussed in this section.

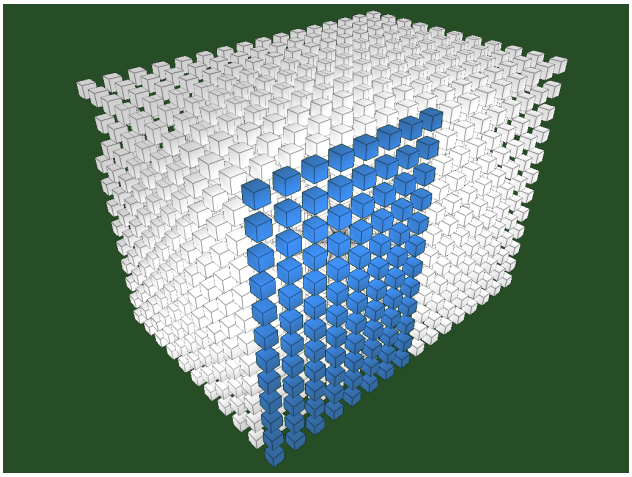

Figure 2. Xt3dmon wired view of BigBen with cabinet nodes highlighted.

2.1 Graphical Monitor

To help visualize XT3 nodes and various system characteristics, a graphical machine monitor called

Xt3dmon [3][4] has been developed that features several views of the machine including both physical and wired (logical) ones using node attribute information from the *processor* table of the *XTAdmin* database within the System Database (SDB). Figure 1 shows a physical view from Xt3dmon of the machine in two rows with nodes grouped by cabinet, cage and module.

Figure 2 shows a wired view of the machine which helps to visualize the interconnect topology.

In both the physical and wired views nodes may be color coded to display the location of jobs within the system. Using both views, this tool has helped to visualize the proximity of nodes assigned to jobs. Xt3dmon has been an important tool in helping to understand both the node connections and node selections from node allocation algorithms.

2.2 Batch System Changes

On BigBen, PSC has developed a custom scheduler called Simon[2] and has made changes to PBS Pro[1] that allow the selection of nodes on which a job will run. This provides complete control over node selection for job assignment and moves the node selection logic for a job from the Compute Processor Allocator (CPA) in the Cray software to Simon. Figure 3 shows an example job demonstrating Simon's controlled node selection as viewed with Xt3dmon.

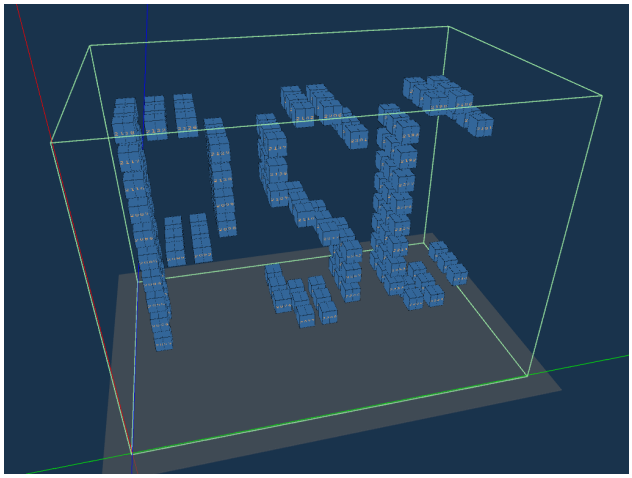

Figure 3. Nodes allocated by Simon in "PSC" shape.

XT3 Operational Enhancements[5] details these changes and provides further introduction to Xt3dmon.

2.3 Job Placement

In *Optimizing Job Placement on the Cray XT3*[6] it was determined that allocating job nodes cubically results in fewer communication hops among allocated nodes. This work showed that a speedup of up to 10% was possible by assigning nodes in a cubic geometry versus assignment using an ascending numeric ordering of node ids.

This work also showed that the key to cubic node selection was to select nodes in x-major order by selecting nodes from cabinets down one row then the other. On BigBen, node wiring in the x-direction moves down rows linking cabinets, while node wiring in the y-direction connects nodes within a cabinet and finally, node wiring in the z-direction connects nodes across rows. Figure 4 shows a subset of node connections highlighting x, y and z connections.

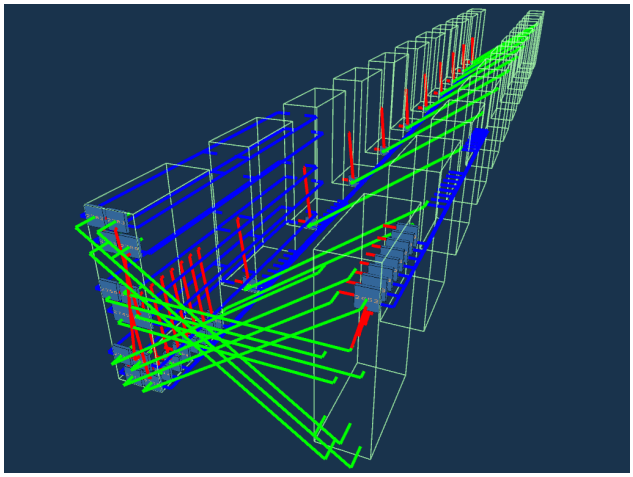

Figure 4. Wired connections of BigBen. Blue shows x-connections down rows. Red shows y-connections within cabinets. Green shows z-connections across rows.

3.0 System Changes to Benefit All Jobs

Using knowledge gained by cubic layout experiments and the ability to control job node layout, a change was made to the node selection algorithm in Simon. Node selection was changed from ascending node id order to an ordering using a mask. Several visual examples will be shown to illustrate this change.

3.1 Scheduler Changes

PSC had previously been selecting job nodes by sorting the list of free nodes by ascending node id and then selecting the first N for a job requiring N nodes. This algorithm was the same as the default one being used by CPA.

Based on job placement work discussed in [6] and using Xt3dmon to visualize job layout, it was discovered that nodes were being assigned in a more fragmented and planar layout. Figure 5 shows job layout with nodes

colored by job and illustrates the planar and fragmented nature of the default selection algorithm.

The new node selection algorithm was designed to select nodes in a cubic geometry by using a node ordering mask, a static, total ordering of all compute nodes, constructed by taking the shortest path through the machine from node to node. The mask was then used to order free nodes on each scheduling cycle, assigning the first N nodes from this list to a job requiring N nodes. The reader is encouraged to view an animation[7] illustrating the construction of the node ordering mask by comparing the physical and wired views of the machine as nodes are added to the mask.

Ordering the list of free nodes according to this mask is computationally no more expensive than sorting them numerically, so there is no additional overhead in using this new algorithm.

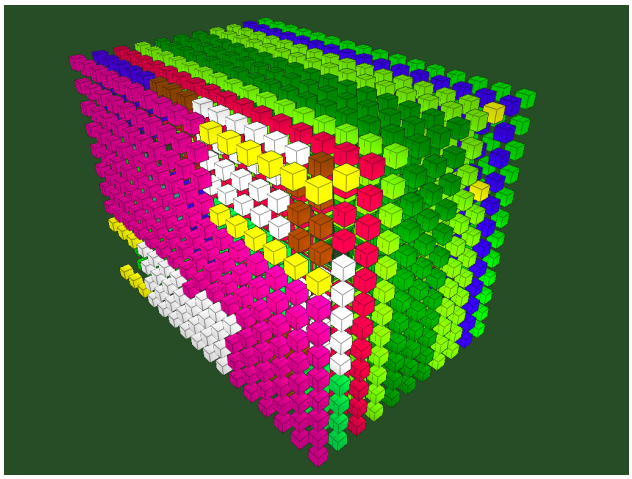

Figure 5. Xt3dmon wired view showing planar nature of default node selection algorithm leading to non contiguous node assignment within a job. Jobs are color coded. Service nodes are yellow.

To illustrate the node selection differences between the default and new algorithms on a set of real jobs, a time lapse animation[8] has been produced that shows a six hour window starting from an empty state on the machine. This animation contrasts the differences between the two algorithms on the same set of jobs and shows how larger jobs generally get contiguous nodes in a cubic geometry using the new algorithm while jobs using the old default node id ordering algorithm have a more planar and non-contiguous geometry. Figures 5 and 6 also help to illustrate these differences.

4.0 System Changes to Benefit Specific Jobs

The changes detailed in section 3 were made to help improve interconnect performance for all jobs. In this section system changes to accommodate applications that understand the machine topology and that can assign tasks to take advantage of node proximity will be reviewed.

For these topology-aware codes each must be given a specific geometry or shape. In addition the codes must know the coordinates of the nodes that have been assigned so that they may assign tasks appropriately.

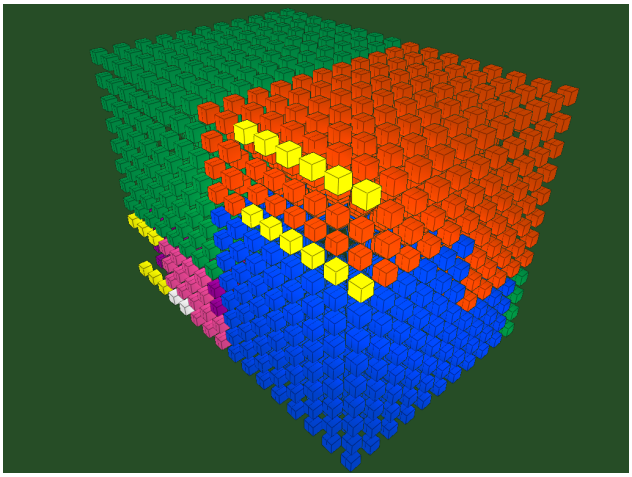

Figure 6. Xt3dmon wired view showing cubic nature of new node selection algorithm leading to contiguous node assignment within a job. Jobs are color coded. Service nodes are yellow.

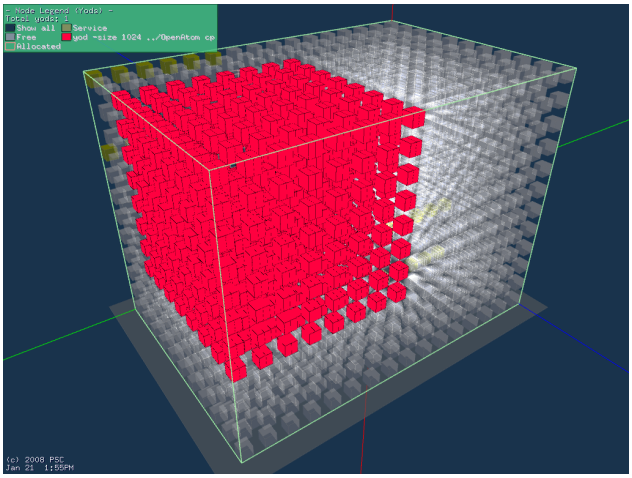

Figure 7. Xt3dmon wired view showing an 8x8x8 node job allocation in red.

4.1 OpenAtom

OpenAtom is a quantum chemistry code that is highly communications bound and its performance is highly influenced by placement on a torus topology machine[9]. The goal of the researchers working with this code on BigBen is to minimize the communication volume of

OpenAtom by minimizing the hop counts between the nodes allocated to the job in which the code is running. Nodes for the code must be allocated in specific geometries: 8x8x16, 8x8x8, 8x8x4 or 8x4x4. See Figures 7 and 8 for an 8x8x8 allocation example. Tasks are then assigned by the code based on the topology of the machine and the geometry of the node allocation.

PSC has helped to facilitate these runs by providing node coordinates from the SDB and also administrator created, shape-oriented reservations that allow jobs to be placed on specific nodes on the system. Initially, as work began on this project, separate reservations were created for each geometry required. Jobs were run in these geometries and results tabulated. This approach turned out to be both time consuming for the users and system administrators to coordinate geometry transitions between each reservation.

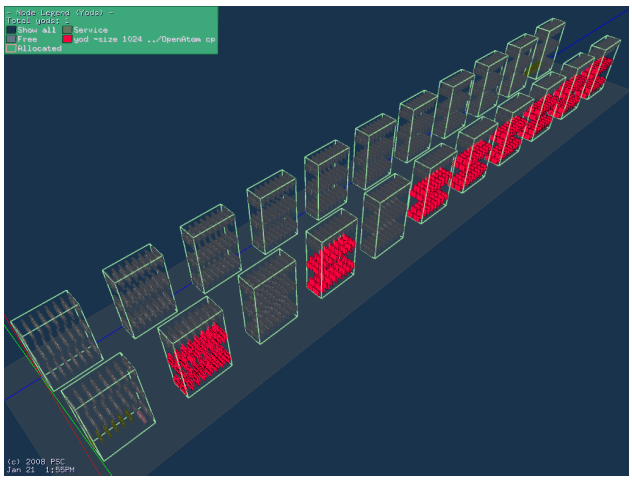

Figure 8. Xt3dmon physical view showing an 8x8x8 node job allocation in red.

After studying needs further, refinements were made to help accommodate the runs without system administrator intervention between geometry transitions. Using files with lists of node ids comprising the desired geometries (8x8x16, 8x8x8, etc.) and passing these files to the system application launcher command, *yod*, some of the geometry control was passed to the users so that they could control the shape at application launch time. Since all desired shapes were sub-shapes of the largest one allocated for the job, no further adjustments to the reservation was needed for the duration of the reservation.

To accommodate these runs in production in the future, users will need a way to request a geometry when submitting jobs to the system and this is being considered.

4.2 Coarse Grain/Fine Grain Decomposition

A second topology-aware application using BigBen that is still in development is a molecular dynamics code. The goal of this project is to speed up simulations by running them in parallel. Progress with the application is just beginning so details at the time of writing are not complete. This section discusses expected plans to accommodate this work.

The expectation is that the parallel decomposition of the code will break down work into fine-grained units of up to four nodes that will communicate frequently with each other. See Figure 9 for an expected fine-grained grouping of nodes. Therefore, it is important that these nodes be kept close together.

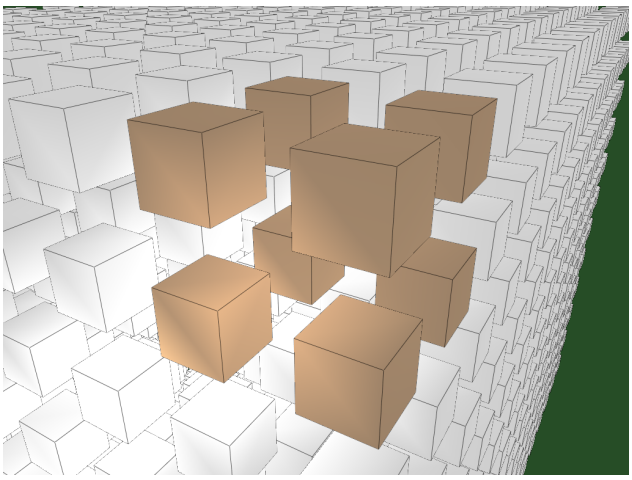
Figure 9. Wired view showing grouping of four contiguous nodes used for fine-grained communications. Intra-group communication is very frequent.

At a higher level, the fine-grained decomposed groups will communicate with each other less frequently. They will be placed close together, again using topology information from the machine. See Figure 10 illustrating the course-grained layout.

For these jobs, the scheduler will allocate a specific shape for the job through a reservation and then the application will layout tasks appropriately within the shape assigned. As with the OpenAtom project, node coordinate information from the SDB will be made available to this code.

5.0 Future Work

While previous work showed that up to a 10% speedup was possible using contiguous node scheduling, it is not clear how well this benefits all jobs. Further study of this is planned. Also, fragmentation is an issue with

placement scheduling and more work needs to be done concerning fragmentation mitigation. Plans to study ways to better schedule jobs so that special geometries can be requested while keeping fragmentation to a minimum are being considered.

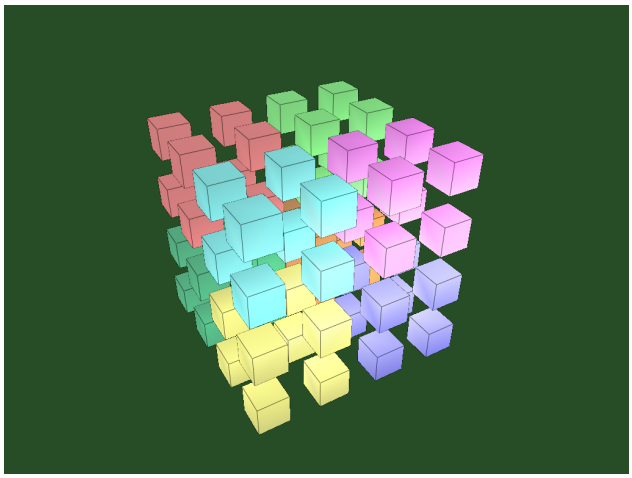

Figure 10. Wired view showing groupings of fine-grained communication blocks.

Also, users need to be provided with a means to specify a desired node geometry for reserved nodes and to be provided with accessible machine topology information. Changes to the batch reservation system are being considered to accommodate these needs.

Finally, all work discussed has been conducted on a Catamount system. Similar results using Compute Node Linux (CNL) are expected. Plans are being made to run tests under CNL to confirm equivalent operation and results.

6. Conclusions

It is clear that contiguous node placement can benefit communication intensive jobs. Changes have been made to the system to provide better communication performance to these jobs through the use of a new scheduling algorithm that helps to keep nodes assigned to jobs close together.

Changes have also been made to the system to accommodate jobs that require a specific node geometry. These changes help reduce intervention between the user and administrator when these jobs are run.

Finally, as a message to Cray, users who understand how their codes can take advantage of the topology of the machine have found it frustrating to get machine topology information from within their codes. Cray is urged to make this information more readily available.

Acknowledgments

This work is supported by the National Science Foundation under TeraGrid Resource Partners, SCI-0504078, and Cooperative Agreement No. SCI-0456541.

The author wishes to thank Jared Yanovich and Derek Simmel for their hard work and attention to detail in preparing the images used and the animations referenced in this paper.

About the Author

Chad Vizino is a member of the Scientific Computing Systems group currently working on scheduling systems, resource management and accounting. He can be reached at PSC, 300 S. Craig St., Pittsburgh, PA 15213. Phone: 412-268-4960. E-mail: vizino@psc.edu.

References

[1] http://www.altair.com/software/pbspro.htm

[2] C. Vizino, J. Kochmar, R. Scott: Custom Features of a Large Cluster Batch Scheduler. *PDPTA 2005: 3-9*.

[3] C. Vizino, J. Kochmar, N. Stone, R. Scott. Batch Scheduling on the Cray XT3. *CUG 2005*.

[4] Xt3dmon Man Page: http://staff.psc.edu/yanovich/xt3dmon/xt3dmon.pdf

[5] C. Vizino, N. Stone, R. Scott. XT3 Operational Enhancements. *CUG 2006.*

[6] D. Weisser, N. Nystrom, C. Vizino, S. Brown, J. Urbanic. Optimizing Job Placement on the Cray XT3. *CUG 2006*.

[7] Animation showing node id mask in physical and wired views: http://staff.psc.edu/vizino/cug2008/opt_nid_list_mask.mov

[8] Animation showing contiguous versus node id ordering node selection: http://staff.psc.edu/vizino/cug2008/cubic_vs_planar.mov

[9] A. Bhatele, E. Bohm, L. Kale. Improving parallel scaling performance using topology-aware task mapping on Cray XT3 and IBM Blue Gene/L. Submitted to *The 37th International Conference on Parallel Processing* (ICPP-08).