Task Mapping for Non-contiguous Allocations

David P. Bunde
Knox College
dbunde@knox.edu

Johnathan Ebbers
Knox College
jebbers@knox.edu

Stefan P. Feer
3M Health Information Systems, Inc.
sfeer@mmm.com

Vitus J. Leung
Sandia National Laboratories
vjleung@sandia.gov

Nickolas W. Price
Knox College
nprice@knox.edu

Zachary D. Rhodes
Allstate Corporation
rhodesz87@gmail.com

Matthew Swank
Knox College
mswank@knox.edu

ABSTRACT

We examine task mapping algorithms for systems that allocate jobs non-contiguously, such as the Cray X series. Several studies have shown that task placement affects job running time for both contiguously and non-contiguously allocated jobs. Our focus is on jobs with a stencil communication pattern. We use experiments on a Cray XE to evaluate novel task mapping algorithms as well as some adapted to this setting. This is done with the miniGhost miniApp which mimics the performance of CTH, a shock physics application. Our strategies improve average running time by as much as 28% over a baseline strategy.

1. INTRODUCTION

We focus on improving the performance of parallel jobs by optimizing the placement of their tasks. This problem is called task mapping because the tasks are being mapped to nodes. The goal is to map communicating tasks to nearby nodes. This can improve message latency, but also bandwidth since messages consume bandwidth on each link used. Thus, extra hops mean wasted bandwidth. This has always been true, but its importance is being fueled by two ongoing trends. First, processors are improving faster than networks, increasingly making bandwidth the performance limiting factor. Second, node counts are growing, increasing both the number of hops between nodes and the potential for hotspots. Several experiments have shown that improved mapping can significantly impact performance, including a speedup of 1.64 times achieved on a full application [2].

Broadly speaking, prior work on task mapping falls into two categories of approaches. Graph-based approaches represent the problem as embedding a communication graph into a machine graph. This formulation loses geometric information such as node coordinates. Thus, the problems are formally hard and cannot exploit structure appearing in some practical applications. Whole-machine approaches target systems such as Blue Gene that allocate each job a contiguous group of nodes. These approaches address structured communication patterns by folding and stretching one grid onto another. Such algorithms cannot be used on systems with non-contiguous allocation, such as the Cray XE.

In this paper, we study task mapping for jobs with structured communication patterns and non-contiguous allocations. Specifically, we look at mapping jobs that communicate in a regular 3D nearest neighbor pattern onto a 3D mesh. This is the simplest case, but non-trivial because nodes allocated to other jobs interfere with the mapping.

2. ALGORITHMS

Now we describe our algorithms. As preprocessing, they rotate the job to match the relative order of the job dimensions with those of the nodes' bounding box. For example, a job with longer x dimension will rotate if assigned nodes whose bounding box has a longer y dimension. All algorithms except Baseline and Grouping (prior work) do this.

We compare our algorithms against two from prior work:

- Baseline (provided by ALPS and Moab) numbers cores in allocation order, numbers tasks in row-major order, and maps corresponding elements.
- Grouping [3] follows Baseline after grouping tasks into 2 by 2 by 4 blocks, each mapped to a 16-core node.

We also adapted a couple of mapping algorithms that assume contiguous allocation [1]:

- Corner (called "Expand from corner" in [1]) numbers both the tasks and cores by their distance from $(0, 0)$ and maps corresponding elements.
- AllCorners ("Corners to center") rotates between the corners, selecting the 1st node by distance to the bottom left, the 2nd by distance to the top left, etc.

Finally, we implemented some new algorithms:

- ColMajor numbers tasks and nodes in column major order and maps corresponding elements.
- RowMajor does this with row major numbering.
- Ordered extends ColMajor and RowMajor by trying different linear orderings and taking whichever gives the lowest average hops.

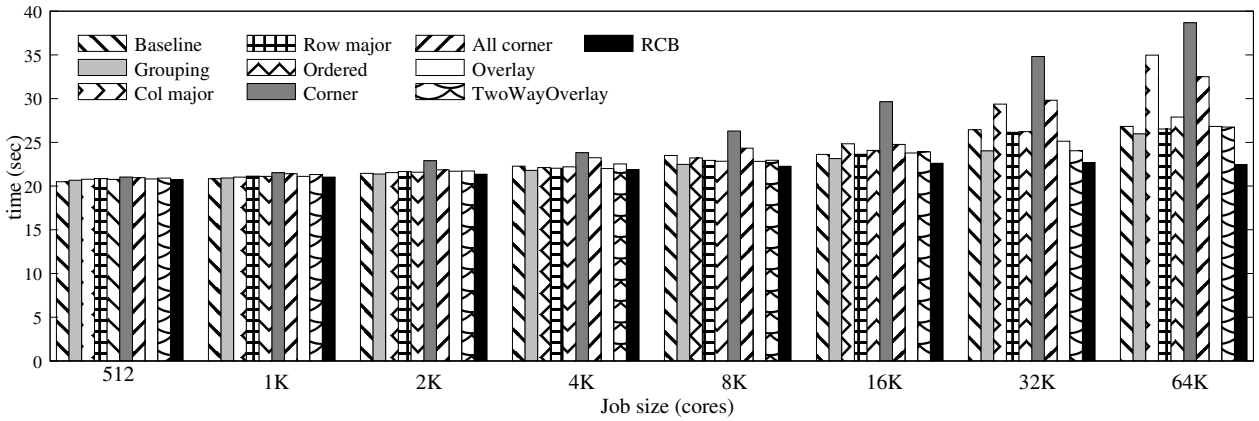

Figure 1: Running time as a function of job size

- Overlay identifies a desired place for each task. The lower left task's desired coordinates are the minimum x and y among allocated nodes. Other tasks get desired places relative to this. Each task is then mapped to the unmapped node closest to its desired place.
- TwoWayOverlay follows Overlay, but alternately maps tasks from the lower left and the top right.
- RCB (Recursive Coordinate Bisection) bisects the job based on its longest dimension (X, Y, or Z), bisects the nodes using the same dimension, and then recursively maps each half of the job onto the corresponding half of the nodes. This creates parallel decomposition trees.

[5] also has a bisection heuristic, but they take a graph-based approach. Since the job and machine are arbitrary graphs, bisection is done with graph partitioning, losing problem geometry. RCB outperforms their approach in our setting.

3. EXPERIMENTS

The experiments were run on Cielo (#22 on the June 2013 Top 500 list), a Cray XE6 with 143,104 compute cores in 8,944 dual socket compute nodes. Each socket has an oct-core processor. The interconnect is a Cray Gemini 3D torus in a 16x12x24 topology, with two nodes per Gemini.

The application used in the experiments was miniGhost. As part of the exascale research program, the DOE lab community is developing mini applications (miniApps) that are representative of the computational core of major advanced simulation and computing codes. MiniGhost is a miniApp modeled on the computational core of CTH, which uses stencil computations for multi-material, large deformation, strong shock wave, solid mechanics simulations. Its communication performance has been shown similar to CTH [4].

For a given job size (in cores), miniGhost can run with different numbers of cores per MPI rank (denoted cores/rank), which changes the number of ranks (but not nodes) to vary the MPI/OpenMP mix. A trend we observed is that the worst mappers tend to perform best at 16 cores/rank (fewer messages) while better mappers favor intermediate values.

Figure 1 shows average running time (over 5 runs) by job size using 4 cores/rank, a value used by others [3] to balance message number and size. RCB does consistently well and is the best algorithm for most job sizes. With 4 cores/rank, its outperformance of Baseline increases with job size, reaching just over 16% at 64K cores. It does even better with other values, reaching 24.1% with 2 cores/rank and 28.4% with 1. These gains are fairly consistent across runs, with standard deviations of 0.6, 0.4, and 0.2 seconds for 4, 2, and 1 cores/rank. The overall best running time for jobs with 8K+ cores was RCB with 2 cores/rank in 17 of the 20 trials (5 runs of 4 sizes). We also found that

- The rotation step improves total time in ∼60% of the runs, for 1–6% average improvement.
- Spearman rank correlation tests show average hops between communicating nodes correlates with running time. Also correlated (but not as highly) with time are maximum hops and variance.

Acknowledgments

We thank C. Vaughan and K. Hastings for helpful discussions. D.P. Bunde, J. Ebbers, S.P. Feer, N.W. Price, Z.D. Rhodes, and M. Swank were partially supported by contract 899808 from Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

4. REFERENCES

[1] A. Bhatelé, G.R. Gupta, L.V. Kalé, and I.-H. Chung. Automated mapping of regular communication graphs on mesh interconnects. In Proc. Intern. Conf. High Performance Computing (HiPC), 2010.

[2] F. Gygi et al. Large-scale electronic structure calculations of high-Z metals on the BlueGene/L platform. In Proc. 2006 ACM/IEEE Conf. Supercomputing (SC), 2006.

[3] R. Barrett et al. Navigating an evolutionary fast path to exascale. In Proc. 3rd Intern. Workshop Performance Modeling, Benchmarking and Simulation of High Performance Computing Systems (PMBS), 2012.

[4] R.F. Barrett et al. MiniGhost: A miniApp for exploring boundary exchange strategies using stencil computations in scientific parallel computing. Tech. Report SAND2011-5294832, Sandia National Laboratories, 2011.

[5] T. Hoefler and M. Snir. Generic topology mapping strategies for large-scale parallel architectures. In Proc. 25rd ACM Intern. Conf. Supercomputing (ICS), 2011.