Optimization of the Hop-Byte Metric for Effective Topology Aware Mapping

C. D. Sudheer
Department of Mathematics and Computer Science
Sri Sathya Sai Institute of Higher Learning, India
Email: cdsudheerkumar@sssihl.edu.in

A. Srinivasan
Department of Computer Science
Florida State University, Tallahassee, FL 32306, USA
Email: asriniva@cs.fsu.edu

***Abstract*—Suitable mapping of processes to the nodes of a massively parallel machine can substantially improve communication performance by reducing network congestion. The hop-byte metric has been used as a measure of the quality of such a mapping by several recent works. Optimizing this metric is NP hard, and thus heuristics are applied. However, the heuristics proposed so far do not directly try to optimize this metric. Rather, they use some intuitive methods for reducing congestion and use the metric just to evaluate the quality of the mapping. In fact, heuristics intending to optimize other metrics too don't directly optimize for them, but, rather, use the metric to evaluate the results of the heuristic. In contrast, we pose the mapping problem with the hop-byte metric as a quadratic assignment problem and use a heuristic to directly optimize for this metric. We evaluate our approach on realistic node allocations obtained on the Kraken system at NICS. Our approach yields values for the metric that are up to 75% lower than the default mapping and 66% lower than existing heuristics. However, the time taken to produce the mapping can be substantially more, which makes this suitable for somewhat static, though possibly irregular, communication patterns. We introduce new heuristics that reduce the time taken to be comparable to that of existing fast heuristics, while still producing mappings of higher quality than existing ones. We also use theoretical lower bounds to suggest that our mapping may be close to optimal, at least for medium sized problems. Consequently, our work can also provide insight into the tradeoff between mapping quality and time taken by other mapping heuristics.**

I. Introduction

Recent works [3], [4], [5], [6], [8], [9], [10], [11] have shown substantial communication performance improvement on large parallel machines by suitable assignment of processes or tasks to nodes of the machine. Earlier works on graph embedding are usually not suitable for modern machines because the earlier works used metrics suitable for a store-and-forward communication mechanism. On modern machines on the other hand, in the absence of network congestion, latency is quite independent of location; communication performance is limited by contention on specific links. Yet another significant difference is that the earlier works typically embedded graphs onto standard network topologies such as hypercubes and meshes. On massively parallel machines, jobs typically acquire only a fraction of the nodes available, and the nodes allocated do not correspond to any standard topology, even when the machine does. For example, we show below in fig. 1 an allocation for 1000 nodes on the 3D torus Jaguar machine at ORNL. We can see that the nodes allocated are several discontinuous pieces of the larger machine. Assignment of tasks to nodes that take this into account can reduce communication overhead. For example, we used a specific mapping to reduce communication time in a load balancing algorithm by over 30% on Jaguar, and this mapping also reduced MPI_allgather time by a similar amount.

Recent works use heuristics that can be intuitively expected to reduce network congestion and then evaluate it either empirically or using some metric. The hop-byte metric has attracted much attention recently. It is defined as the sum over all the messages of the product of the message size and number of hops the message has to traverse. On a store-and-forward network, this would correspond to the total communication volume. The intuition behind this metric is that if the total communication volume is high, then it is also likely to increase the contention for specific links, which would then become communication bottlenecks. Although this metric does not directly measure the communication bottleneck caused by contention, heuristics with low values of this metric tend to have smaller communication overheads. This serves as a justification for this metric. The advantage of this metric is that it requires only the machine topology, while computing contention would require routing information. We discuss recent developments in this topic in section 2.

In contrast to other approaches, we use the hop-byte metric for producing the mappings too, rather than just using it for evaluating the mapping. Optimizing for the hop-byte metric can easily be shown to be a specific case of the Quadratic Assignment Problem (QAP), which

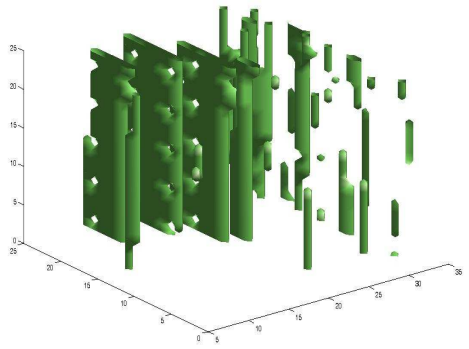

Fig. 1. Allocation of 1000 nodes on Jaguar. The axes correspond to indices on the 3D torus, and the green region corresponds to allocated nodes.

is NP hard. Exact solutions can be determined using branch and bound for small problem sizes. We use the existing GRASP heuristic for medium-sized problem. An advantage of the QAP formulation is that we can use theoretical lower bounds to judge the quality of our solution. The time taken to determine the mapping using an exact solver or GRASP increases rapidly with problem size. In that case, we consider a couple of alternate approaches. In the first case, we develop new heuristics that improve on some limitations of other heuristics for this problems. In the second case, we use graph partitioning to break up the problem into smaller pieces, and then apply GRASP to each partition. We discuss our approach further in section 3.

We evaluate our approach on six different communication patterns. We determine the values of the metric for different heuristics using node allocations obtained on the Kraken supercomputer at NICS. In contrast to other works that usually assume some standard topology, our results are based on actual allocations obtained. We see up to 75% reduction in hop-bytes over the default allocation, and up to 66% reduction over existing heuristics. Furthermore, our results are usually within a factor of two of a theoretical lower bound on the solution. For small problem sizes, that lower bound is usually just a little over half the exact solution. Consequently, it is likely that our solutions are close to optimal. We describe our results further in section 4.

The outline of the rest of the paper is as follows. We describe related work in section 2 followed by description of our approach in section 3. We present empirical results in section 4 and summarize our conclusions in section 5.

II. RELATED WORK

Mapping of processes to nodes based on the network topology attracted much attention in the earlier days of parallel computing. It lost its importance for some time with the advent of communication mechanisms such as worm-hole routing. However, for reasons explained in section I, it had once again attracted much attention recently.

Different topological strategies for mapping torus process topologies onto the the torus network of BlueGene/L were presented by Yu, Chung, and Moreira [9]. Bhatele and Kale [5] proposed topology-aware load-balancing strategies for CHARM++ based Molecular Dynamics applications. Their analysis maps mesh and torus process topologies to other mesh and torus network topologies. Several techniques for mapping regular communication graphs onto regular topologies were developed [7]. The mapping is a NP-hard problem [13]; hence heuristics are used to approximate the optimal solution. Heuristic techniques for mapping applications with irregular communication graphs to mesh and torus topologies were developed, and some of them even take advantage of the physical coordinate configuration of the allocated nodes [8]. The performance of these heuristics were evaluated based on the hop-byte metric. Hoefler and Snir [10] present mapping algorithms that are meant for more generic use and the algorithms are evaluated using the maximum congestion metric – the message volume on the link with maximum congestion. Their heuristics based on recursive bisection and graph similarity were used to map application communication patterns on realistic topologies. The metrics here again are used to evaluate the mappings rather than being used to determining the mapping. The algorithm Greedy Graph Embedding (GGE) proposed in [10] is used by us for comparison, because it performs best among the heuristics they have proposed. Furthermore, the algorithm can be used with arbitrary communication patterns and network topologies, even though the implementation in [10] was more restricted.

Krishna et al. developed topology aware collective algorithms for Infiniband networks. These networks are hierarchical with multiple levels of switches, and this knowledge was used in designing efficient MPI collective algorithms [11]. The reduced scalabilty of the latency and effective bandwidth due interconnect hot spot for fat-tree topologies is addressed with topology-aware MPI node ordering and routing-aware MPI collective operations [12]. Graph partitioning libraries such as SCOTCH and Metis provide support for mapping graphs to network topologies.

Massively parallel systems such as Cray XT and BlueGene/P systems are generally heavily loaded with

multiple jobs running and sharing the network resources simultaneously, this results in application performance being dependent on the node allocation for a particular job. Balaji et al. analyzed the impact of different process mappings on application performance on a massive Blue Gene/P system [14]. They show that the difference can be around 30% for some applications and can even be two fold for some. They have developed a scheme where by the user can describe the application communication pattern before running a job, and the runtime system then provides a mapping that potentially reduces contention.

III. MAPPING HEURISTICS

A. Problem formulation

We model the problem using two graphs. The node graph G is an undirected graph with vertices representing the nodes of the machine that have been allocated to the job submitted, and edge weights e_{ij} representing the number of hops between the vertices i and j linked by that edge. The hops can be determined from the machine topology information and knowledge of the nodes actually allocated. These can usually be obtained on most supercomputing systems. We assume that static routing is used, which is fairly common.

The task graph $G' = (V', E')$ represents the submitted job. Each vertex is represents a process running on a node and edge weight e'_{ij} represents the total sizes of messages, in both directions, between vertices i and j linked by that edge. The number of vertices in this graph must equal the number in the node graph. If there are more tasks than nodes, then a graph partitioning algorithm can be applied to aggregate tasks so that this condition is satisfied.

Minimizing the hop-bytes then is the following quadratic assignment problem, where x_{ij} = 1 implies that task j is assigned node i.

$$min \sum_{ij} \sum_{kl} e_{ik} e'_{jl} x_{ij} x_{kl}, (1)$$

subject to:

$$\sum_{i} x_{ij} = 1, for\, all\, j$$

$$\sum_{j} x_{ij} = 1, for\, all\, i$$

x_{ij} in {0,1}

This is a well known NP hard problem and considered hard to approximate, though there are reasonable approximation algorithms [2] for dense instances. The exact solution can be found for small instances using branch and bound (bounds are used to reduce the search space). A couple of popular lower bounds are the elimination bound and the Gilmore-Lawler bound. We use the exact solution for small instances, and also the bounds for medium size instances, in order to evaluate the effectiveness of our heuristics.

B. GRASP heuristic

Several heuristics have been proposed for QAP, based on meta-heuristics such as simulated annealing, tabu search, ant colony optimization, and greedy randomized adaptive search procedures (GRASP). GRASP is based on generating several random initial solutions, finding local optima close to each one, and choosing the best one. We use a GRASP heuristic for QAP from [1].

C. MAHD and exhaustive MAHD heuristics

We first describe our faster heuristic, Minimum Average Hop Distance (MAHD), in Algorithm 1 below. It improves on the following limitation of the GGE [10] heuristic. GGE replaces step 7 of algorithm 1 with a strategy that places the task on the node closest to its most recently mapped neighbor. We would ideally like this task to be close to all its neighbors. MHT [8] addresses this by placing the node closest to the centroid of all previously mapped neighbors. On the other hand MHT works only on meshes. Our algorithm works on a general graph, and places the task, in this step, on the node that has the minimum average hop distance to nodes on which all previously mapped neighbors of the task have been mapped. MHT also selects a random node on which to place the initial task. We intuitively expect a task with the maximum number of neighbors to be a "central" vertex in a graph, and so try to map it to a node which is "central" in its graph. We do this by placing it on the node with the minimum average hop distance to any other vertex. The first task selected may not actually be "central" (for instance, in the sense centrality measures such as betweenness centrality). We introduce an Exhaustive MAHD (EMAHD) heuristic to see if a better choice of initial vertex is likely to lead to significant improvement in mapping quality. In this heuristic, we try all possible nodes as starting vertices, and then choose the one that yields the best time.

Algorithm: MAHD heuristic. (The actual implementation deals with special cases such as the graphs not being connected.)

D. Hybrid heuristic with graph partitioning

If $|V'|$ is too large for GRASP to be feasible, then we use the following hybrid heuristic. We partition graphs G and G' into partitions of size p each. Any graph partitioning algorithm can be used. We use a multilevel heuristic available in parMetis. We create graphs H and

Algorithm 1 MAHD(G, G')

1. s = vertex in G with maximum number of neighbors
2. p = vertex in Gẃith minimum average hop distance to all other vertices
3. Assign task s to node p
4. Insert all neighbors of s into max-heap H, where the heap is organized by the number of neighbors
5. **while** H in not empty **do**
6. s = H.pop();
7. s is mapped to a node with minimum average hop distance to processes hosting mapped neighbors of s;
8. Insert neighbors of s into H if they are not in H and have not been mapped;
9. **end while**

H' corresponding to the partitions of G and G' respectively. In H, each vertex corresponds to a partition in G and in H', each vertex corresponds to a partition in G'. Each edge in H has weight corresponding to the average hops from nodes between the two partitions linked by that edge. Each edge in H' has weight corresponding to the total message sizes between the two partitions linked by that edge. A mapping of partitions in H' to partitions in H is performed using GRASP and mapping of tasks to nodes within each partitions is again performed using GRASP with corresponding subgraphs of G and G'.

IV. Evaluation of Heuritics

A. Experimental platform

The experimental platform is the Cray XT5 Kraken supercomputer at NICS. It contains 18,816 dual hex-core Opteron nodes running at 2.6 GHz with 8 GB memory per socket. The nodes are connected by SeaStar 2+ routers with a 3-D torus topology having dimensions 25 x 16 x 24. Compute Node Linux runs on the compute nodes. We used the native Cray compiler with optimization flag "$-O3$".

The QAP codes were obtained from QAPlib (http://www.opt.math.tu-graz.ac.at/qaplib/codes.html), which includes codes from a variety of sources. A branch and bound algorithm was used for the exact solution, the Gilmore-Lawler bound for a lower bound[1] and a dense GRASP heuristic for larger problem sizes.

Three standard collective communication patterns used in MPI implementations – recursive doubling, Bruck, and binomial tree – were studied. We also used the following three irregular communication patterns. A 3-D spectral element elastic wave modeling problem (3Dspectralwave) and a 2-D PDE (aug2dc) from the University of Florida sparse matrix collection, and a 2D unstructured mesh pattern from the ParFUM framework available in CHARM++ library.

OSU MPI micro benchmark suite was used for the empirical tests on the Kraken machine to observe the impact of the heuristic based mapping on MPI collective calls.

[1]The Gilmore-Lawler bound performed better than the elimination bound for large problem sizes.

B. Experimental Results

Figures 2-3 compare the default mapping, GRASP result, and the Gilmore-Lawler bound for small problem sizes. The quality (hop-byte metric value) is divided by that for the exact solution to yield a normalized quality.

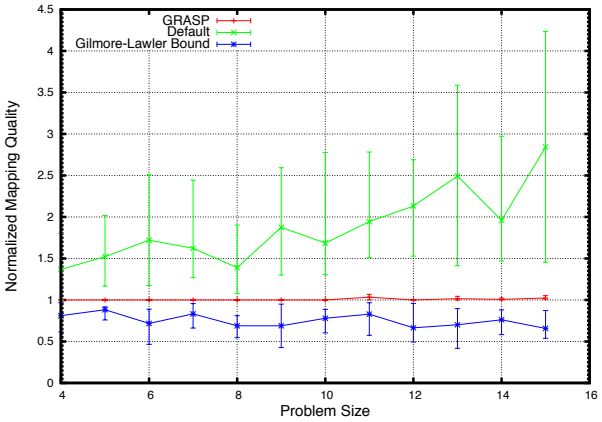

Fig. 2. Quality of solutions on the recursive doubling pattern for small problem sizes.

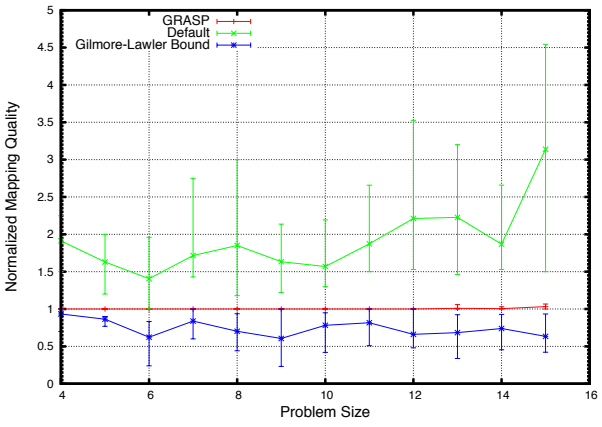

Fig. 3. Quality of solutions on the binomial tree pattern for small problem sizes.

We can see that the GRASP is close to the exact solution for these problem sizes. We also note that the lower bound is a little over the half of the exact solutions toward the higher end of this size range. Similar trend were observed for the other patterns, which are not shown here.

Figures 4-9 compare the heuristics for medium problem sizes. The quality is normalized against the default solution, because computing with the exact solution is not feasible. These figures, therefore, show improvement over the default mapping.

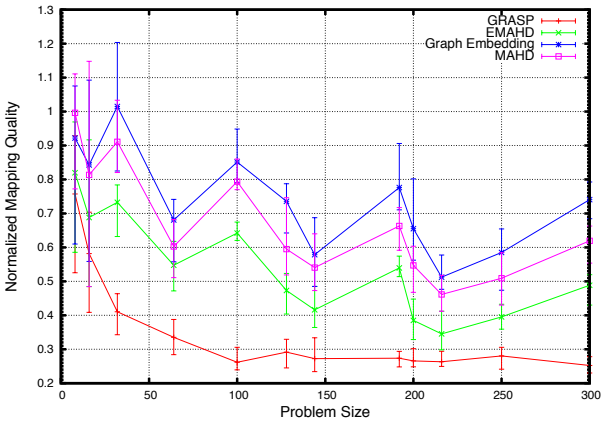

Fig. 4. Quality of solution on the recursive doubling pattern for medium problem sizes.

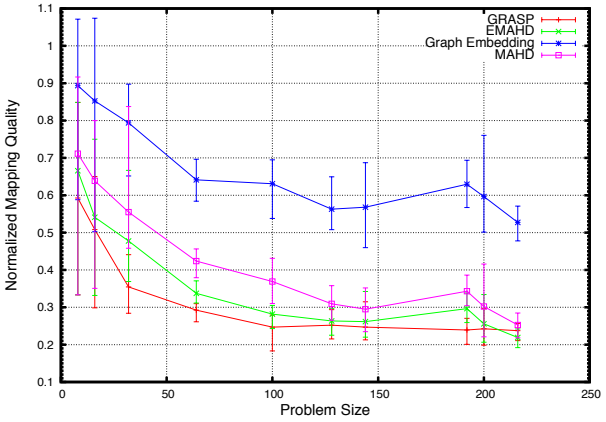

Fig. 5. Quality of solution on the binomial tree pattern for medium problem sizes.

We can see that the GRASP heuristic is consistently better than the default and the GGE heuristic, while MAHD and EMAHD are sometimes comparable to GRASP. EMAHD is often much better than MAHD, suggesting that a better choice of the initial vertex has potential to make significant improvement to MAHD.

However, the time taken by GRASP and EMAHD are significantly larger than that for GGE or MAHD, as shown in figure 10. Consequently, they are more suited to static communication patterns. Since EMAHD typically does not produce better quality than GRASP either, it does not appear very useful. On the other hand, its quality suggests that if a good starting vertex can be found for MAHD without much overhead, then

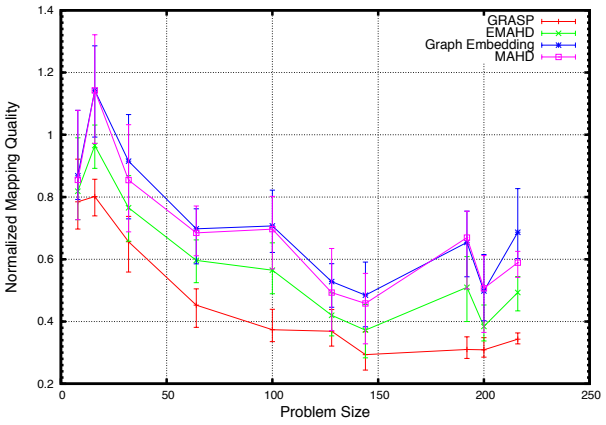

Fig. 6. Bruck Quality of solution on the Bruck pattern for medium problem sizes.

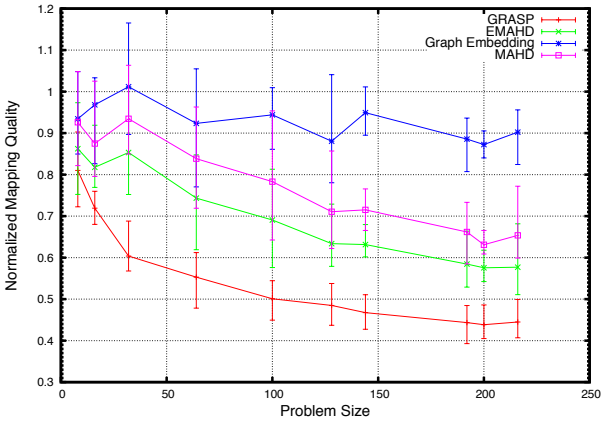

Fig. 7. Quality of solution on the 3D spectral pattern for medium problem sizes.

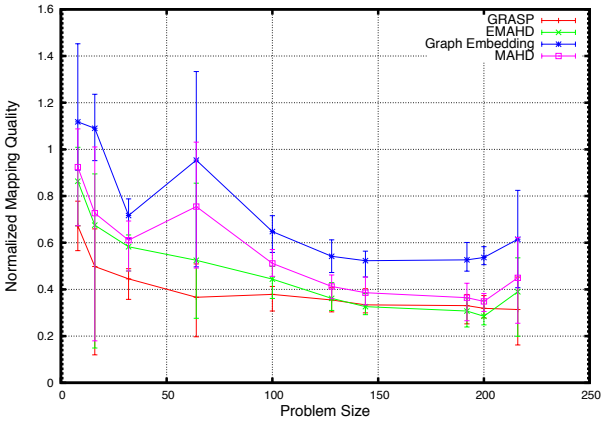

Fig. 8. Quality of solution on the Aug2D pattern for medium problem sizes.

MAHD's quality can be improved without increasing its run time. When the communication pattern changes dynamically, then MAHD is a better alternative to the above two schemes and also to GGE. It is as fast as

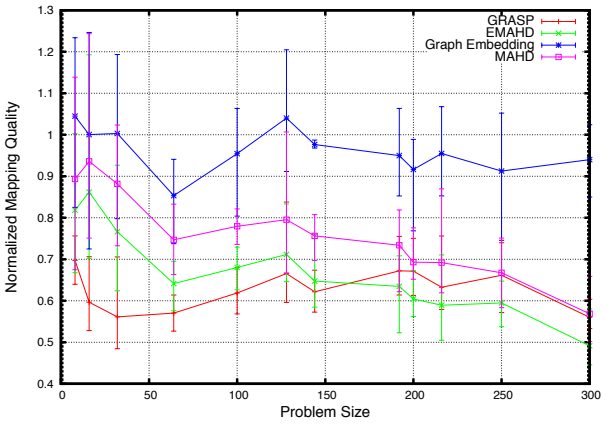

Fig. 9. Quality of solution on the mesh pattern for medium problem sizes.

GGE, while producing mappings of better quality. Its speed also makes it feasible to use it dynamically, while GRASP is too slow.

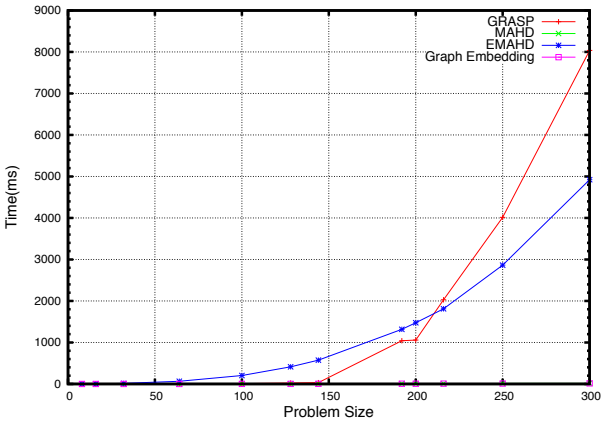

Fig. 10. Comparison of time taken by the heuristics.

An alternative to MAHD for large problem sizes with dynamic communication patterns is the hybrid algorithm. Preliminary results on 1000 nodes with partitions of size 125 are shown in figure 11. The hybrid algorithm performs better than the default and GGE with both communication patterns. It is better than MAHD and EMAHD for recursive doubling, but is worse with the binomial tree. The binomial tree has less communication volume than recursive doubling, and further experiments are necessary to check if the hybrid algorithm tends to perform better when the communication volume is larger. We note that even for medium sized problems, GRASP (which is the underlying heuristic behind the hybrid algorithm), was comparable with EMAHD and MAHD for the binomial tree, but much better for recursive doubling. The hybrid scheme produces a further reduction in quality, which makes it worse than MAHD and EMAHD for binomial tree, but since GRASP is much better for recursive doubling, the hybrid algorithm is better than MAHD and EMAHD for it, though by a smaller margin. As the number of partitions increases, the hybrid algorithms relative advantage decreases, as can be seen in figure 12[2], which is based on 16 partitions of size 125 each.

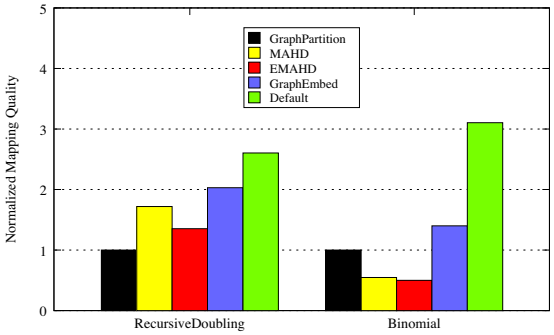

Fig. 11. Comparison of heuristics on 1000 nodes (12,000 cores).

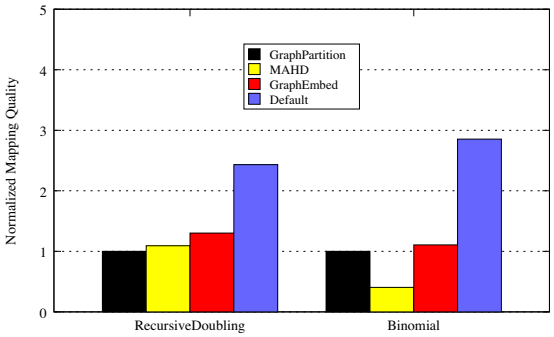

Fig. 12. Comparison of heuristics on 2000 nodes (24,000 cores).

We next evaluate how well GRASP compares with the lower bound for medium problem sizes. Figures 13-18 show that GRASP is around a factor of two from the Gilmore-Lawler bound. As shown earlier, the above bound was usually a little higher than half the exact solution for small problem sizes, and did not get tighter with increased sizes. These results, therefore, suggest that GRASP is close to the optimal solution.

Finally, we wish to verify if improving the hop-byte metric actually improves the communication performance. (Related works mentioned in section II, dealing with this metric, have provided further evidence in favor of this.) Preliminary studies with recursive doubling on the MPI_Allgather implementation with 1KB messages on problems with 128 nodes showed that GRASP and EMAHD are about 25% faster than the default and 20%

[2]It was not feasible to use EMAHD for 2000 nodes due to the time required by it.

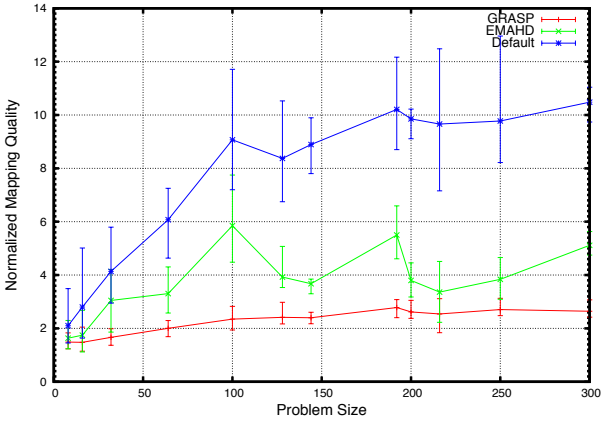

Fig. 13. Quality of solution on the recursive doubling pattern compared with a lower bound.

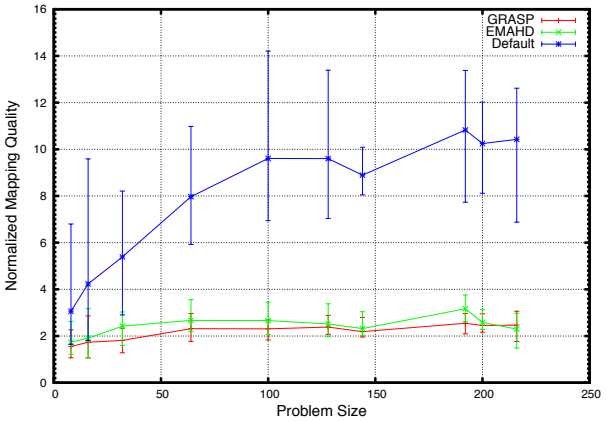

Fig. 14. Quality of solution on the binomial tree pattern compared with a lower bound.

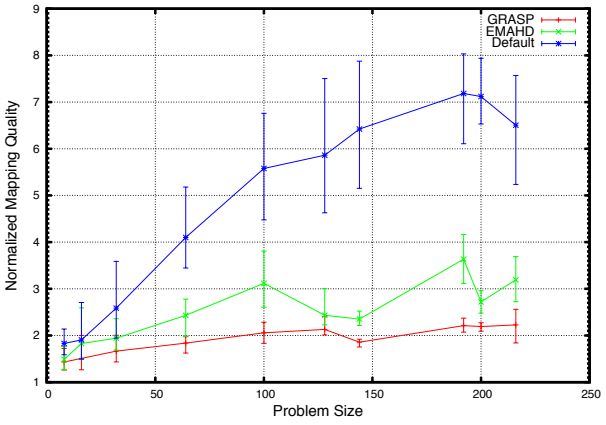

Fig. 15. Bruck Quality of solution on the Bruck pattern compared with a lower bound.

faster than GGE. The improvement over the default is significant, though not as large as that indicated by the hop-byte metric, because that metric is only an indirect indication of the quality of the mapping. However, it

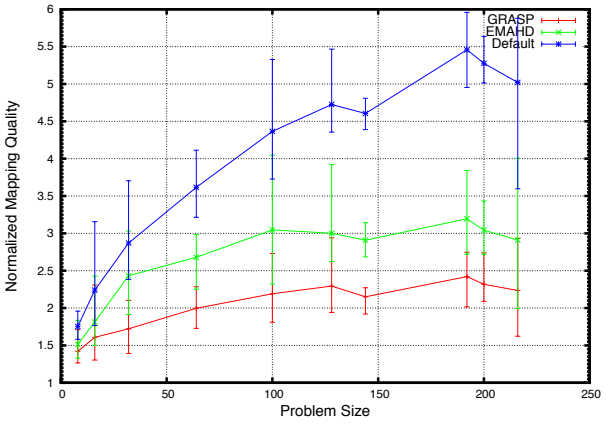

Fig. 16. Quality of solution on the 3D spectral pattern compared with a lower bound.

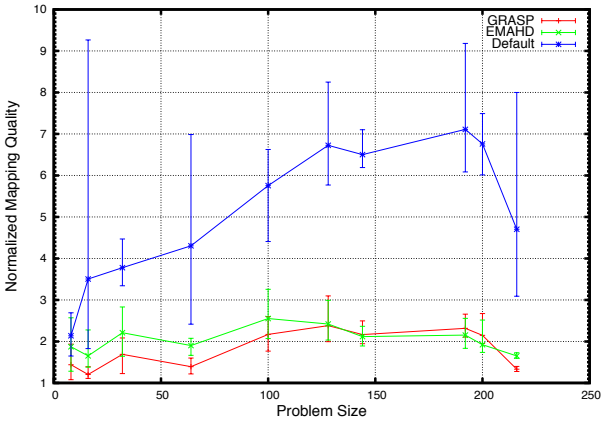

Fig. 17. Quality of solution on the Aug2D pattern compared with a lower bound.

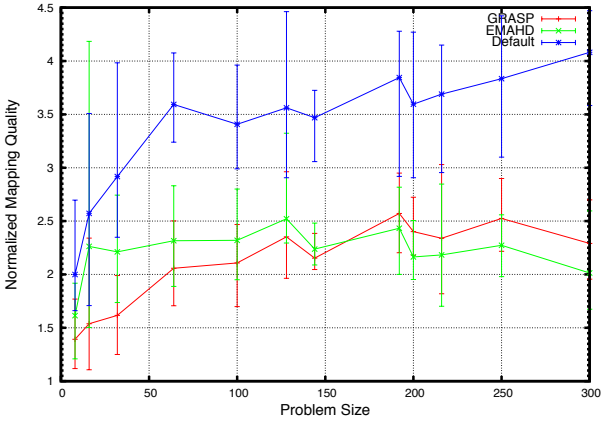

Fig. 18. Quality of solution on the mesh pattern compared with a lower bound.

does suggest that optimizing the hop-byte metric leads to improved performance.

V. CONCLUSION AND FUTURE WORK

We have shown that optimizing for the hop-bytes metric using the GRASP heuristic leads to a better mapping than existing methods, which typically use some metric just to evaluate the heuristic, rather than to guide the optimization. We have evaluated the heuristic on realistic node allocations, which typically consist of many disjoint connected components. GRASP performs better than GGE, which is among the best prior heuristics that can be applied to arbitrary graphs with arbitrary communication patterns, and performs much better than the default mapping. In fact, GRASP is optimal for small problem sizes. Comparison with the lower bound suggests that GRASP may be close to optimal for medium problem sizes too. However, it does not scale well with problem size and is infeasible for large graphs. We proposed two solutions for this. One is the MAHD algorithm and the other is a hybrid algorithm. The former is fast and reasonably good, while the latter is sometimes better, but much slower than MAHD. For static communication patterns on medium sized graphs, GRASP would be the best option. For large problems with static communication patterns, the hybrid approach would be a good alternative, especially when the communication volume is large. However, MAHD too can be effective. For dynamic communication patterns, MAHD is the best alternative. In fact, results with EMAHD suggest that if a good starting vertex can be found, then MAHD may be competitive even for many static patterns. MAHD takes roughly the same time as GGE, but consistently outperforms it. Preliminary experiments on MPI also suggest that optimizing for the hop-byte metric improves the actual MPI collective communication latency, though not to the extent predicted by this metric. This is reasonable, because the metric does not directly account for the congestion bottleneck.

One direction for future work is in reducing the time taken by the GRASP heuristic. GRASP is a general solution strategy, rather than a specific implementation. The particular implementation that we used is for a general QAP problem. We plan to develop an implementation specific to our mapping problem. For instance, solutions generated by the fast heuristics can be used as starting points in GRASP, thereby reducing the search space. Furthermore, we used a dense GRASP implementation because the node graph is complete. We can remove edges with heavy weights (corresponding to nodes that are far away) so that a sparse algorithm can be used. A different direction lies in optimizing for a different metric. The actual bottleneck is contention on specific links. We have posed the problem of minimizing the maximum contention as an integer programming problem, and will develop heuristics to solve it.

ACKNOWLEDGMENT

This work was partially supported by an ORAU/ORNL grant under the HPC program and by computer time allocation from XSEDE.

REFERENCES

[1] Y. Li, P.M. Pardalos, and M.G.C. Resende, A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 16 (1994) 237-261.

[2] S. Arora, A. Frieze, and H. Kaplan, A New Rounding Procedure for the Assignment Problem with Applications to Dense Graph Arrangement Problems. Mathematical Programming, Vol. 92 (2002), 1-36.

[3] G. Bhanot, A. Gara, P. Heidelberger, E. Lawless, J.C. Sexton, and R. Walkup, Optimizing Task Layout on the Blue Gene/L Supercomputer. IBM Journal of Research and Development, Vol. 49 (2005) 489-500.

[4] P. Balaji, R. Gupta, A. Vishnu, and P. Beckman, Mapping Communication Layouts to Network Hardware Characteristics on Massive-Scale Blue Gene Systems. Comput. Sci. Res. Dev., Vol. 26 (2011) 247-256.

[5] A. Bhatele, L.V. Kale, and S. Kumar, Dynamic Topology Aware Load Balancing Algorithms for Molecular Dynamics Applications. In proceedings of ICS, 2009.

[6] A. Bhatele and L.V. Kale, Application-Specific Topology-Aware Mapping for Three Dimensional Topologies. In proceedings of IPDPS, 2008.

[7] A. Bhatele, G. Gupta, L. V. Kale, and I.-H. Chung, Automated Mapping of Regular Communication Graphs on Mesh Interconnects, in Proceedings of International Conference on High Performance Computing (HiPC), 2010.

[8] A. Bhatele and L.V. Kale, Heuristic-based techniques for mapping irregular communication graphs to mesh topologies, Proceedings of Workshop on Extreme Scale Computing Application Enablement - Modeling and Tools, 2011.

[9] H. Yu, I.-H. Chung, and J. Moreira. Topology mapping for Blue Gene/L supercomputer. In SC06, page 116, New York, NY, USA, 2006. ACM.

[10] T. Hoefler and M.Snir, Generic Topology Mapping Strategies for Large-scale Parallel Architectures, In proceedings of the 2011 ACM International Conference on Supercomputing (ICS 2011).

[11] K. Kandalla, H. Subramoni, A. Vishnu, and D.K. Panda, "Designing Topology-Aware Collective Communication Algorithms for Large Scale Infiniband Clusters: Case Studies with Scatter and Gather". The 10th Workshop on Communication Architecture for Clusters (CAC 10), held in conjunction with Int'l Parallel and Distributed Processing Symposium (IPDPS 2010).

[12] E. Zahavi, "Fat-trees routing and node ordering providing contention free traffic for MPI global collectives". Journal of Parallel and Distributed Computing, February 2012, ISSN 0743-7315, 10.1016/j.jpdc.2012.01.018.

[13] Shahid H. Bokhari, On the Mapping Problem, IEEE Trans. Computers, vol. 30, no. 3, pp. 207214, 1981.

[14] Pavan Balaji, Rinku Gupta, Abhinav Vishnu, Pete Beckman, "Mapping communication layouts to network hardware characteristics on massive-scale blue gene systems", Comput Sci Res Dev (2011) 26: 247256 DOI 10.1007/s00450-011-0168-y.