OS/Runtime challenges for dynamic topology aware mapping

Abhinav Bhatele and Todd Gamblin
Center for Applied Scientific Computing, Lawrence Livermore National Laboratory
E-mail: bhatele@llnl.gov, tgamblin@llnl.gov

As processors have become faster over the years, the cost of a prototypical "computing" operation, such as a floating point addition, has diminished to a negligible quantity. On the other hand, the cost of communicating data has become proportionately higher. For example, even on a high-end supercomputer, it takes less than a quarter nanosecond (amortized, in a pipelined unit) for a floating point addition, 50 ns to access DRAM memory, and thousands of nanoseconds to receive data from another node. If one considers energy, the comparison is also stark: currently, a floating point operation costs 30-45 picojoules (pJ), an off-chip 64-bit memory access costs 128 pJ, and remote data access over the network costs between 128 and 576 pJ [1].

In addition to faster processors, the increasing number of cores per node is stressing the memory and interconnect further. It is common knowledge that the memory bandwidth on node has not kept pace with the increased processing power on the node. The comparison of network bandwidth with floating point operation (flop) counts is similar. Table 1 shows the number of bytes the network can accept for each flop on the node (B-to-F ratio, byte/flop) for different machines. Newer generations of IBM and Cray machines have more cores and more flop/s on each node, but the network bandwidth has not increased in proportion. As a result, for each flop on the node, the network is able to communicate fewer and fewer bytes. It is clear that the cost of communication (in terms of time, silicon area, and energy) will become the critical issue in coming years and, eventually, the primary determinant of overall performance. In order to optimize communication and overall application performance and reduce energy costs, it is imperative to optimize data movement, both on-node and off-node.

IBM		Cray	
Blue Gene/L	0.375	XT3	8.77
Blue Gene/P	0.375	XT4	1.36
Blue Gene/Q	0.117	XT5	0.23

Table 1: Byte-to-flop ratios

Typically, application developers attempt to minimize off-socket and off-node communication through graph partitioning of the application domain. However, given a partitioned domain, the "mapping" or assignment of sub-domains/communicating tasks in this graph to the underlying on-node and network topology is often neglected. *Topology aware task mapping* involves embedding the partitioned graph of communicating tasks of a parallel application within the node and network topology, to optimize for performance, power (i.e., data motion), or other objectives. In the last ten years, several applications [2, 3, 4, 5] have developed their own hand-tuned mapping algorithms and mechanisms to optimize communication. Several libraries have also been developed to create mappings for torus interconnects [6, 7, 8, 9]. Even so, an application developer has to spend significant time and effort to formulate efficient mappings for his code.

As we move to exascale, we face complex on-node and off-node interconnect topologies. Automatic intelligent mapping by the runtime with assistance from the OS and resource manager will be necessary to achieve good performance. Below, we describe the major challenges and requirements from the resource manager, operating system and runtime for dynamic topology aware mapping:

Topology aware job scheduling: Only 6% of the top500 machines (primarily the IBM Blue Gene series) provide contiguous cuboidal partitions for their jobs. This is the primary requirement for achieving any benefit from mapping of application tasks to the network topology. If the allocated nodes for a job are sprinkled all over the machine, factors such as network interference from other jobs and I/O are outside the control of each job. A careful consideration of the significant performance benefits from contiguous job partitions versus higher overall system utilization from random node allocations is necessary to understand the tradeoffs.

API standard for topology discovery: Mapping at runtime requires knowledge of the topology of the allocation job partition. This information is available through data structures such as `BGPPersonality` on Blue Gene/P, the `MPIX` interface on Blue Gene/Q, through PMI and RCA calls on Cray machines and using `ibnetdiscover` on Infiniband machines. Partial on-node topology and memory hierarchy is available through the `hwloc` library. All of these interfaces are different and only some of them can be used at runtime. An industry standard needs to be established and the vendors should ensure that the OS supports a commonly agreed upon topology discovery API that can be used by the runtime for mapping.

Scalable API for application communication graph: The application should be able to specify its communication graph and this information can be exploited by the runtime to map tasks to the underlying network topology (e.g. routines such as `MPI_Cart_create` and `MPI_Graph_create` in the MPI runtime). However, these routines are no-ops and the MPI runtime does not map its processes to the underlying network topology even if the application describes its communication using these. Future runtime(s) should provide topology aware implementations of such routines that map the application graph on the physical hardware efficiently. This requires research on scalable mapping algorithms (see below).

Scalable mapping algorithms: Even though several libraries have been developed to tackle the mapping problem, either the solutions are very generic (and hence not optimal for all problems) or too specific to a particular application. Significant research is required in tackling this NP-hard [10] problem to develop scalable mapping algorithms that cover most common cases. Research is also needed in developing metrics to predict performance of the application with different mappings and in understanding on-node memory contention and off-node network congestion. The mapping algorithms need to be deployed in the runtime to create the best possible mapping of application tasks to the on-node and off-node topology. Automated mapping by the runtime for optimizing memory accesses on NUMA multi-cores has not received much attention so far [11].

Migration of tasks at runtime: Currently, the mapping of MPI applications can only be changed when a job is launched (given prior knowledge of the application communication graph and the shape of the partition). Most parallel applications have several phases, with varied communication patterns and network behavior. An initial mapping decided at program start-up might not be optimal for the entire execution. At exascale, in the face of adaptivity, asynchrony and dynamic load imbalance, we will require dynamic task migration (that is also topology-aware). Only a few runtimes such as Charm++ [12] have the capability to migrate tasks at runtime. This will be a much desired feature in runtimes of the future.

If the HPC (OS and runtime) community can agree on these five facets/requirements of task mapping described above and we spend significant effort on this research, only then there is hope for near-optimal communication efficiency in exascale applications.

The cost of communication (in terms of time, silicon area, and energy) will become the critical issue in coming years and, eventually, the primary determinant of overall performance. In order to optimize communication and overall application performance and reduce energy costs, it is imperative to maximize data locality and minimize data movement, both on-node and off-node.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-MI-563334).

References

[1] Peter Kogge, Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Jon Hiller, Sherman Karp, Stephen Keckler, Dean Klein, Robert Lucas, Mark Richards, Al Scarpelli, Steven Scott, Allan Snavely, Thomas Sterling, R. Stanley Williams, and Katherine Yelick. Exascale computing study: Technology challenges in achieving exascale systems, 2008.

[2] F. Gygi, E. W. Draeger, M. Schulz, B. R. de Supinski, J. A. Gunnels, V. Austel, J. C. Sexton, F. Franchetti, S. Kral, C. W. Ueberhuber, and J. Lorenz. Large-scale electronic structure calculations of high-Z metals on the Blue Gene/L platform. *In Proceedings of Supercomputing 2006*, 2006. International Conference on High Performance Computing, Network, Sto rage, and Analysis. 2006 Gordon Bell Prize winner (Peak Performance).

[3] Blake G. Fitch, Aleksandr Rayshubskiy, Maria Eleftheriou, T. J. Christopher Ward, Mark Giampapa, and Michael C. Pitman. Blue matter: Approaching the limits of concurrency for classical molecular dynamics. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, New York, NY, USA, 2006. ACM Press.

[4] Abhinav Bhatelé, Laxmikant V. Kalé, and Sameer Kumar. Dynamic topology aware load balancing algorithms for molecular dynamics applications. In *23rd ACM International Conference on Supercomputing*, 2009.

[5] Abhinav Bhatele, Eric Bohm, and Laxmikant V. Kale. Optimizing communication for charm++ applications by reducing network contention. *Concurrency and Computation: Practice and Experience*, 23(2):211–222, 2011.

[6] 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*, 49(2/3):489–500, 2005.

[7] Hao Yu, I-Hsin Chung, and Jose Moreira. Topology mapping for Blue Gene/L supercomputer. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 116, New York, NY, USA, 2006. ACM.

[8] Abhinav Bhatele. *Automating Topology Aware Mapping for Supercomputers*. PhD thesis, Dept. of Computer Science, University of Illinois, August 2010. `http://hdl.handle.net/2142/16578`.

[9] A. Bhatele, T. Gamblin, S. H. Langer, P.-T. Bremer, E. W. Draeger, B. Hamann, K. E. Isaacs, A. G. Landge, J. A. Levine, V. Pascucci, M. Schulz, and C. H. Still. Mapping applications with collectives over sub-communicators on torus networks. In *Proceedings of 2012 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2012.

[10] Shahid H. Bokhari. On the Mapping Problem. *IEEE Trans. Computers*, 30(3):207–214, 1981.

[11] Laercio Pilla, Christiane Ribeiro, Daniel Cordeiro, Chao Mei, Abhinav Bhatele, Philippe Navaux, Francois Broquedis, Jean-Francois Mehaut, and Laxmikant Kale. A hierarchical approach for load balancing on parallel multi-core systems. In *2012 International Conference on Parallel Processing (ICPP)*, 2012.

[12] L.V. Kalé and S. Krishnan. CHARM++: A Portable Concurrent Object Oriented System Based on C++. In A. Paepcke, editor, *Proceedings of OOPSLA'93*, pages 91–108. ACM Press, September 1993.