Lawrence Livermore National Laboratory

LLNL-TR-678732
FY2015 LDRD Final Report

Task Mapping on Complex Computer Network Topologies for Improved Performance*

Lead PI: Abhinav Bhatele (13-ERD-055)

Abstract

The increase in flop/s capacity and memory bandwidth on-node at a higher rate than the inter-node link bandwidth is making many large-scale parallel applications communication-bound, meaning their performance is now limited by available network resources. The higher cost of moving data on the network necessitates developing techniques to optimize data movement on the network. Task mapping refers to the placement of communicating tasks on interconnected nodes with the goal of optimizing data movement between them. This project focuses on developing tools and techniques for mapping application tasks on complex supercomputer network topologies to improve the performance of high performance computing (HPC) simulations.

In order to develop algorithms for task mapping, we first need to gain a better understanding of congestion on supercomputer networks. We present tools we have developed to quantify network congestion empirically on popular networks. We use these tools to measure network congestion during application execution and then use the gathered network congestion data to build machine learning models that can provide insights into the root causes of network congestion. Next, we discuss RAHTM and Chizu, tools we have developed to map applications tasks to network topologies. We present results demonstrating significant performance improvements by using task mapping on production HPC applications.

In the course of this project, we identified additional objectives that could assist in completing the main goals of the project. The first is studying inter-job congestion on supercomputer systems in which the network is shared by multiple users and their jobs. The second is developing a network simulator to perform what-if analyses impossible to do on a production system and to study network congestion and task mapping on future supercomputer systems. We present Damselfly and TraceR, tools we have developed for simulating application execution on supercomputer networks.

1 Background and Research Objectives

As processors have become faster over the years, the cost of a prototypical "computing" operation, such as a floating point addition, has rapidly grown smaller. 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 [Kogge et al. (2008)]. In order to optimize communi-

*Lawrence Livermore National Laboratory is operated by Lawrence Livermore National Security, LLC, for the U.S. Department of Energy, National Nuclear Security Administration under Contract DE-AC52-07NA27344.

cation 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.

The main goal of this LDRD project is to minimize communication costs through interconnect topology aware *task mapping* of HPC applications. Task mapping involves understanding the communication behavior of the code and embedding its communication within the network topology, optimizing for performance, power (i.e., data motion), or other objectives. Several application codes from LLNL have demonstrated significant performance improvements by mapping tasks intelligently to the underlying physical network. The Gordon Bell award winning submissions for both Qbox [Gygi et al. (2006)] and ddcMD [Streitz et al. (2005)] used task mapping to improve performance by more than 60% and 35% respectively, on the Blue Gene/L platform. For two WCI codes, Kull and ALE3D, developers have demonstrated performance improvements of 62% and 27%, respectively, simply by changing the assignment of domains to MPI ranks, which changes the message routing on the physical network.

The mappings mentioned above were created for specific applications/packages targeting a specific architecture. Creating these mappings is time consuming and difficult, because the general problem is NP-hard [Shahid H. Bokhari (1981)]. Complex topologies such as the five-dimensional (5D) torus on the Blue Gene/Q Sequoia system at LLNL make this even harder – Qbox, ddcMD and other application developers have expressed the difficulty of mapping to five dimensions and a need for tools/frameworks that can do this. The work described here has focused on creating these mappings for application developers and end users automatically.

The research objectives of this work are: (a) develop tools to analyze the communication behavior of parallel applications and the resulting network congestion, (b) investigate techniques and develop models to understand network congestion on supercomputers, and (c) design, implement and evaluate algorithms for mapping tasks in a parallel application to the underlying network topology. The research objectives were met, as described in Sections 2.1, 2.2 and 2.3. We also identified additional objectives in the course of the project: (d) study inter-job congestion on supercomputer systems in which the network is shared by multiple users and their jobs, and (e) develop a network simulator to perform what-if analyses impossible to do on a production system and to study network congestion and task mapping on future supercomputer systems. The outcomes of these additional goals are described in Sections 2.4 and 2.5 respectively.

2 Scientific Approach and Accomplishments

In this section, we describe the experimental and theoretical methods developed to achieve the objectives of the project and the results that show the successful completion of these objectives.

2.1 Measuring Network Congestion

In order to measure congestion empirically on IBM and Cray supercomputer networks, we developed libraries that can record network hardware performance counters on these platforms. These counters can record information about the network traffic on the system. The libraries developed for the IBM Blue Gene/Q system and the Cray XC30 (Cascade) system are described below:

BGQNCL: The Blue Gene/Q Network Performance Counters Monitoring Library (BGQNCL) monitors and records network counters during application execution. BGQNCL accesses the Universal Performance Counter (UPC) hardware counters through the Blue Gene/Q Hardware Performance Monitoring (HPM) API. The UPC hardware programs and counts performance events from

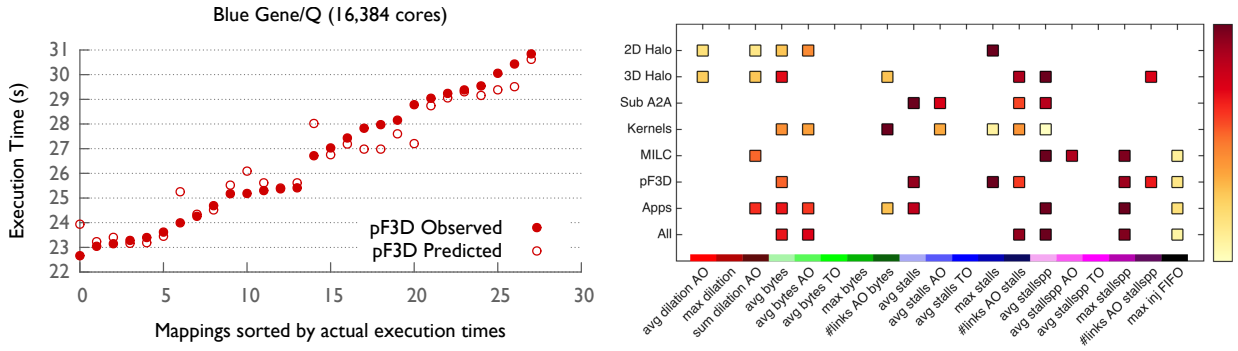

Figure 1: The left figure compares the predicted versus observed execution time of pF3D on 16,384 cores of Blue Gene/Q for different task mappings. The predictions are made using a machine learning model based on various input features derived from network hardware performance counters, and lead to a highly accurate ordering of mappings. The right figure compares the ranks or importances of different features (hardware counters or metrics) in the model for eight different datasets. The three most important features are: receive buffers on intermediate nodes, network links and injection FIFOs in decreasing order of importance. Note that the marker colors for each row are scaled independently (maroon/red is high and yellow is low).

multiple hardware units on a Blue Gene/Q node. BGQNCL is based on the PMPI interface and uses HPM to control the network unit of the UPC hardware.

AriesNCL: The Aries Network Performance Counters Monitoring Library (AriesNCL) monitors and records network router tile performance counters on the Aries router of the Cray XC30 (Cascade) platform. AriesNCL accesses the network router tile hardware counters through the Performance API (PAPI) [Mucci et al. (1999)].

Both BGQNCL and AriesNCL provide an easy-to-use interface to monitor and record network performance counters for an application run. Counters can be recorded for the entire execution or for specific regions or phases marked with `MPI_Pcontrol` calls. These libraries were used extensively in this research and are being released under the GNU Lesser General Public License (LGPL).

2.2 Identifying Culprits Behind Congestion

The ability to predict the performance of communication-heavy parallel applications without actual execution can be very useful. This requires understanding which network hardware components affect communication and in turn, performance on different interconnection architectures. A better understanding of the network behavior and congestion can help in performance tuning through the development of congestion-avoiding and congestion-minimizing algorithms.

Jain et al. (2013) developed a machine learning approach to understand network congestion on supercomputer networks. We used regression analysis on communication data and execution time to find correlations between the two and to learn models for predicting the execution time of new samples. Using our methodology, we obtained prediction scores close to 1.0 for individual applications. We were also able to reasonably predict the execution time on higher node counts using training data for smaller node counts. We also obtained reasonable ranking predictions of execution times for different task mappings of new applications using datasets based on communication kernels only. Figure 1 (left plot) shows the predicted and observed execution times of a laser-plasma interaction code, pF3D [Langer et al. (2011)], used in National Ignition Facility (NIF) experiments.

Using a quantile analysis technique to identify relevant feature subsets, Bhatele et al. (2015) were also able to extract the relative importance of different features and that of the corresponding

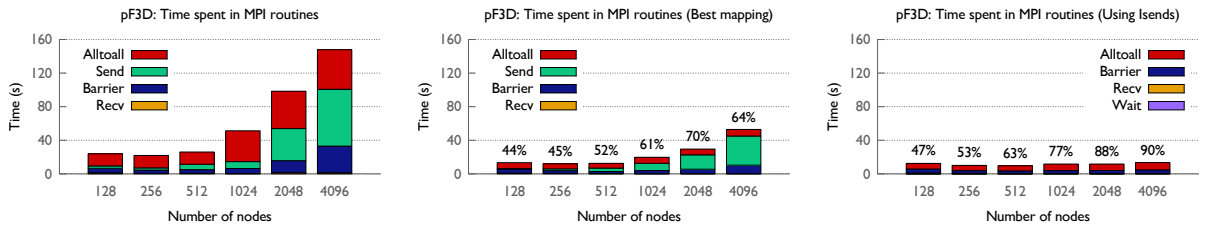

Figure 2: A scaling comparison of the time spent in different MPI routines in pF3D running on Blue Gene/Q using the default mapping (left), using the best mapping discovered (center) and using the best mapping with the Isend optimization (right). We observe performance improvements ranging from 44% to 90% depending on the node count.

hardware components in predicting execution time (Figure 1, right). This helped us to identify the primary root causes or culprits behind network congestion, which is a difficult challenge. We were able to identify the hardware components that are primarily responsible for impacting and hence, predicting the execution time. These are – receive buffers on intermediate nodes, network links and injection FIFOs in decreasing order of importance. We also observed that network hot-spots and dilation or the number of hops a message travels are lesser indicators of network congestion. This knowledge gives us a real insight into network congestion on torus interconnects and can be very useful to network designers and application developers.

2.3 Developing and Evaluating Task Mappings

Creating task mappings is time consuming and difficult. Hence, we have developed tools and frameworks described below that create these mappings for application developers automatically.

RAHTM: Abdel-Gawad et al. (2014) designed and implemented a routing algorithm aware, hierarchical task mapping (RAHTM) technique that scales well to 16K tasks. RAHTM uses linear programming and overcomes the computational complexity of routing-aware mapping by using a mix of near-optimal techniques and heuristics. RAHTM's bottom-up approach solves leaf-level problems using near-optimal linear programming techniques and a heuristic (greedy) combining technique to merge leaf-level mapping solutions into a larger whole. RAHTM is an offline mapping tool and its mapping can be used repeatedly in subsequent runs of the application. As such, the cost to derive the mapping is not on the critical execution path and does not add to run-time overheads. RAHTM achieves a 20% reduction in communication time which translates to a 9% reduction in overall execution time for a mix of three communication-heavy benchmarks.

Chizu: Chizu is a framework for mapping application processes or tasks to physical processors or nodes in order to optimize communication performance. It takes the communication graph of an HPC application and the interconnection topology of a supercomputer as input. The output is a new MPI rank to processor mapping, which can be used when launching the HPC application within a job partition. Chizu exploits graph partitioning software available in the public domain such as METIS [George Karypis and Vipin Kumar (1998)], Scotch [Chevalier et al. (2006)], PaToH and Zoltan [Devine et al. (2005)]. It uses a recursive k-way bi-partitioning algorithm that performs recursive partitioning and mapping. The aim is to optimize different communication metrics such as the total number of hops that messages travel, or bisection bandwidth or network interface card (NIC) congestion. Chizu is under review and being released under the GNU LGPL.

Chizu and a previously developed mapping tool, Rubik [Bhatele et al. (2012)] have been used to map several LLNL applications and proxy applications including ALE3D, pF3D, Qbox, Kripke and

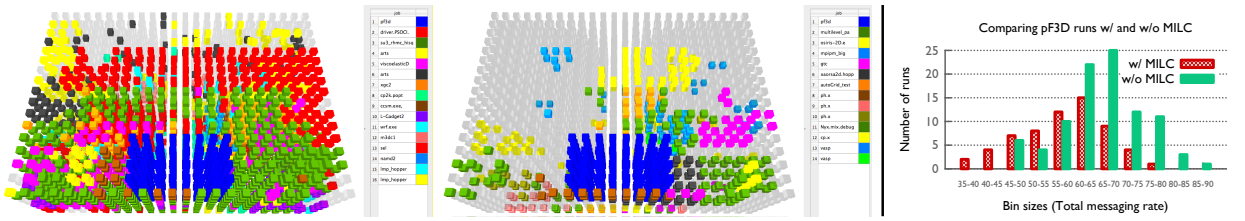

Figure 3: The two screenshots on the left depict the placement of pF3D (blue) and conflicting jobs on Hopper for two separate short runs. The April 11 job (left) yielded a messaging rate nearly 25% below that of the April 16 job (center). The two jobs had the same node placement of pF3D but the slower April 11 job was surrounded by nodes of several other jobs, including a large communication-heavy job, MILC (shown in green). The right plot shows the distributions of messaging rates observed for pF3D runs coinciding with a MILC job (red, patterned) and runs with no coinciding MILC job (green, solid).

AMG. Bhatele et al. (2014) used Rubik to map pF3D and MILC and demonstrated significant performance improvements using task mapping (Figure 2). In the process, we also discovered a long standing performance bug in pF3D which has improved the overall scaling and performance of the code by two to four times.

2.4 Studying Inter-job Congestion and Interference

Our goal in this work was to investigate the sources of performance variability in parallel applications running on HPC platforms in which the network is shared between multiple jobs running simultaneously on the system. We used pF3D, a highly scalable, communication-heavy, parallel application that is well-suited for such a study because of its inherent computational and communication load balance. We performed our experiments on three different parallel architectures: IBM Blue Gene/P (Dawn and Intrepid), IBM Blue Gene/Q (Mira), and Cray XE6 (Cielo and Hopper).

When comparing variability on the different architectures, we found that there is hardly any performance variability on the Blue Gene systems, and that there is a significant variability on the Cray systems. We discovered differences between the XE6 machines due to their node allocation policies and usage models. Since Hopper is designed to serve small to medium sized jobs, the nodes allocated to a job tend to be more fragmented than on Cielo, which mostly serves large jobs. This fragmentation on Hopper resulted in higher variability for pF3D, where the communication time varied from 36% faster to 69% slower when compared to the average. We focused our efforts in this paper on investigating the source of the variability on Hopper.

Bhatele et al. (2013) investigated the impact of OS noise, shape of the allocated partition, and interference from other jobs on Hopper and concluded that the primary reason for higher variability is contention for shared network resources from other jobs. From queue logs collected during the pF3D runs, we plotted the position of the concurrent jobs relative to pF3D and examined the performance. We found multiple cases where there was strong evidence that the differences in performance are due to communication activities of competing jobs, with message passing rates of pF3D up to 27.8% slower when surrounded by a communication-heavy application (Figure 3). This work was featured on hpcwire.com.

2.5 Network Simulators and Performance Prediction

Performance prediction tools are important for studying the behavior of HPC applications on large supercomputer systems that are not deployed yet or have access restrictions. They are needed to understand the messaging behavior of parallel applications in order to prepare them for efficient

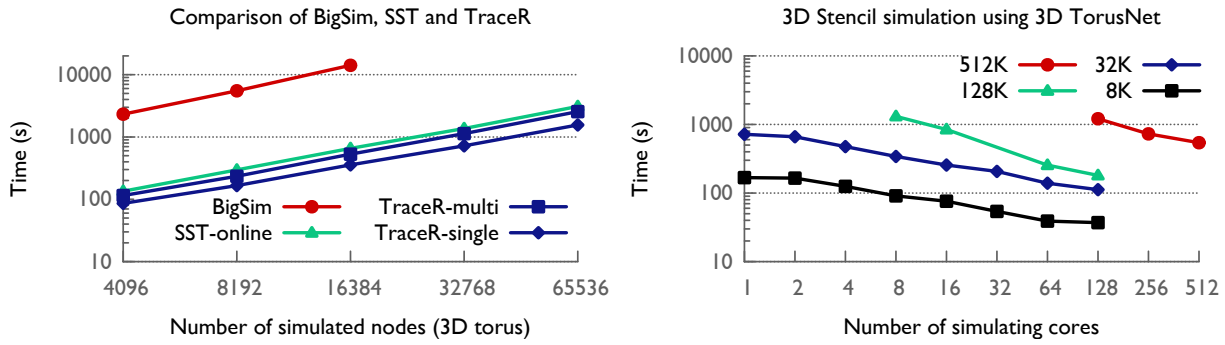

Figure 4: Comparison of the sequential execution time of TraceR with that of BigSim and SST for 3D tori of different sizes (left). Weak and strong scaling performance of TraceR simulating 3D tori of different sizes on different core counts (right).

execution on future systems. We have developed two methods for simulating the communication of HPC applications on supercomputer networks: functional modeling and trace-driven simulation.

Damselfly: Damselfly is a model-based network simulator for modeling traffic and congestion on the dragonfly network designed by Cray [Faanes et al. (2012)]. Given a router connectivity graph and an application communication pattern, the network model in Damselfly performs an iterative solve to redistribute traffic from congested to less loaded links. Jain et al. (2014) have used Damselfly to explore the effects of job placement, parallel workloads and network configurations on network health and congestion. Since Damselfly models steady state network traffic for aggregated communication graphs as opposed to communication traces with time stamps, it does not model and/or predict execution time. Damselfly and TraceR are being released under the GNU LGPL.

TraceR: Predicting execution time requires a detailed cycle-accurate or flit or packet-level simulation of the network. TraceR is designed as an application on top of the CODES simulation framework [Cope et al. (2011)]. It uses traces generated by BigSim's emulation framework [Zheng et al. (2005)] to simulate an application's communication behavior by leveraging the network API exposed by CODES. Under the hood, CODES uses the Rensselaer Optimistic Simulation System (ROSS) as the PDES engine to drive the simulation [Bauer Jr. et al. (2009)]. TraceR can output network traffic as well as predicted execution time.

To the best of our knowledge, TraceR is the first network simulator that enables simulations of multiple jobs with support for per job, user-defined job placement and task mapping. Acun et al. (2015) have shown that TraceR performs significantly better than other state-of-the-art simulators such as SST [Underwood et al. (2007)] and BigSim. Figure 4 shows the sequential and parallel performance of TraceR when simulating a 3D torus network. TraceR also supports other supercomputer network topologies such as dragonfly and fat-tree.

We have been using Damselfly and TraceR to perform various what-if studies and performance prediction. Figure 5 shows a representative result from our studies in which the effect of executing multiple jobs simultaneously on network traffic is compared with that of running a job alone.

3 Impact on Mission

Using tools developed for task mapping, we have been able to improve the performance of pF3D, MILC and Qbox significantly. Plots in Figure 2 show the time spent by pF3D in different MPI

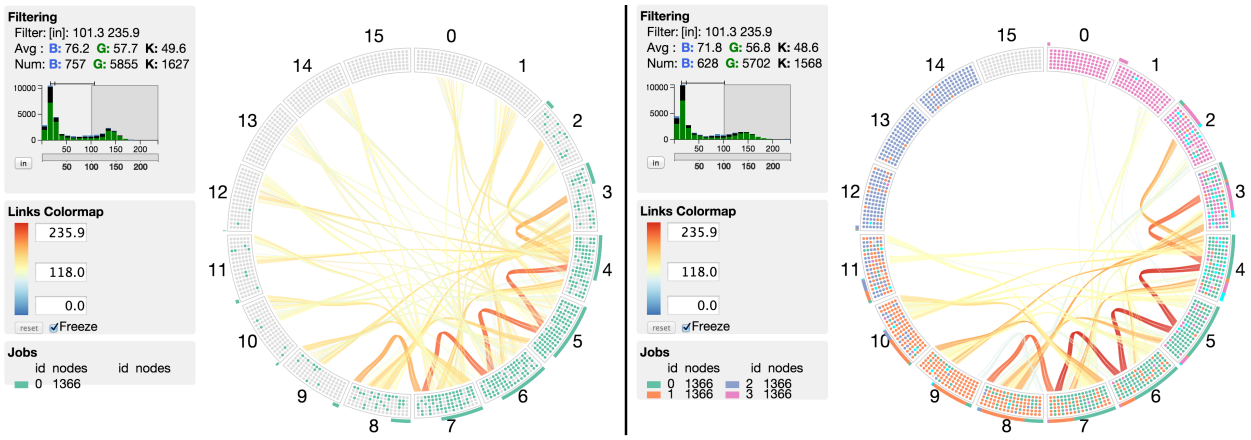

Figure 5: Visualizations of the network traffic on a dragonfly system for two different job workloads/scenarios (traffic output obtained using Damselfly). Comparing with a scenario in which Job 0 (4D Stencil) runs alone (left), when it is run in a workload along side other jobs (right), the number of links with traffic above a certain threshold decreases and overall maximum traffic on inter-group links increases (231 MB as opposed to 191 MB when run alone). In the parallel workload run, Job 0's traffic is confined to fewer inter-group links in order to share bandwidth with other jobs.

routines for the default mapping, the best mappings we found, and the best mappings combined with the Isend optimization. The labels in the center and right plot denote the percentage reduction in communication time compared to the default ABCDET mapping. Tiled mappings improve the communication performance of pF3D on Blue Gene/Q by 2.8× on 4096 nodes and the Isend modification improves it further by 3.9×. Task mappings developed for Qbox were used for scaling it on Sequoia and generating results for a Gordon Bell submission.

In general, the prediction techniques we have developed are widely applicable to a variety of scenarios, such as, (a) creating offline prediction models that can be used for low overhead tuning decisions to find the best configuration parameters, (b) predicting the execution time in new setups, e.g., on a different number of nodes, or different input datasets, or even for an unknown code, (c) identifying the root causes of network congestion on different architectures, and (d) generating task mappings for good performance.

Procurement, installation and operation of supercomputers at leadership computing facilities is expensive in terms of time and money. It is important to understand and evaluate various factors that can impact overall system utilization and performance. In this project, we focused on analyzing network congestion as communication is a common performance bottleneck. Such analyses coupled with the monetary costs of configuration changes can inform future supercomputer purchases and potential upgrades by the Department of Energy (DOE) and LLNL/Livermore Computing. Our performance prediction and visualization tools can be used by machine architects, system administrators and end users to understand application, network and/or overall system performance.

As part of this LDRD, we have released six software related to congestion analysis, task mapping and network simulation. We have made a successful postdoc offer to a graduate student (Nikhil Jain) who worked on this LDRD as part of the subcontract to the University of Illinois. We are also in negotiations with the University of Illinois/National Center for Supercomputing Applications to set up a CRADA via NSF's Petascale Application Improvement Discovery (PAID) program. The subcontract to LLNL has already been approved by NSF and is pending approval by the University of Illinois, LLNL and DOE.

4 Conclusion

In summary, research conducted in this project has led to: (a) a better understanding of network congestion on torus networks, (b) development of a methodology for highly accurate performance prediction using machine learning models, (c) design of tools for task mapping and performance improvements of production scientific applications, and (d) development of network simulators for performing what-if analyses.

Scalable network simulators such as TraceR are a powerful tool to study HPC applications and architectures. In the future, we plan to use TraceR to compare different network topologies and understand deployment costs versus performance tradeoffs. We will also use network simulations to identify performance bottlenecks in production applications on supercomputing systems to be deployed in the near future.

We plan to extend our investigations on inter-job congestion by using a light-weight monitoring framework to collect system-wide performance data and archive it for data mining experiments. We are setting up collaborations with researchers at the University of Illinois/National Center for Supercomputing Applications and the University of Arizona in this area.

References

Abdel-Gawad, A., Thottethodi, M., and Bhatele, A. (2014). RAHTM: Routing-algorithm aware hierarchical task mapping. In *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14. IEEE Computer Society. LLNL-CONF-653568.

Acun, B., Jain, N., Bhatele, A., Mubarak, M., Carothers, C. D., and Kale, L. V. (2015). Preliminary evaluation of a parallel trace replay tool for hpc network simulations. In *Proceedings of the 3rd Workshop on Parallel and Distributed Agent-Based Simulations*, PADABS '15. LLNL-CONF-667225.

Bauer Jr., D. W., Carothers, C. D., and Holder, A. (2009). Scalable time warp on blue gene supercomputers. In *Proceedings of the 2009 ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation*, PADS '09, Washington, DC, USA. IEEE Computer Society.

Bhatele, A., Gamblin, T., Langer, S. H., Bremer, P.-T., Draeger, E. W., Hamann, B., Isaacs, K. E., Landge, A. G., Levine, J. A., Pascucci, V., Schulz, M., and Still, C. H. (2012). Mapping applications with collectives over sub-communicators on torus networks. In *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '12. IEEE Computer Society. LLNL-CONF-556491.

Bhatele, A., Jain, N., Isaacs, K. E., Buch, R., Gamblin, T., Langer, S. H., and Kale, L. V. (2014). Improving application performance via task mapping on IBM Blue Gene/Q. In *Proceedings of IEEE International Conference on High Performance Computing*, HiPC '14. IEEE Computer Society. LLNL-CONF-655465.

Bhatele, A., Mohror, K., Langer, S. H., and Isaacs, K. E. (2013). There goes the neighborhood: performance degradation due to nearby jobs. In *ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '13. IEEE Computer Society. LLNL-CONF-635776.

Bhatele, A., Titus, A. R., Thiagarajan, J. J., Jain, N., Gamblin, T., Bremer, P.-T., Schulz, M., and Kale, L. V. (2015). Identifying the culprits behind network congestion. In *Proceedings of the IEEE International Parallel & Distributed Processing Symposium*, IPDPS '15. IEEE Computer Society. LLNL-CONF-663150.

Chevalier, C., Pellegrini, F., Futurs, I., and I, U. B. (2006). Improvement of the efficiency of genetic algorithms for scalable parallel graph partitioning in a multi-level framework. In *In Proceedings of Euro-Par 2006, LNCS 4128:243252*, pages 243–252.

Cope, J., Liu, N., Lang, S., Carns, P., Carothers, C., and Ross, R. (2011). Codes: Enabling co-design of multilayer exascale storage architectures. In *Proceedings of the Workshop on Emerging Supercomputing Technologies*.

Devine, K. D., Boman, E. G., Heaphy, R. T., Hendrickson, B. A., Teresco, J. D., Faik, J., Flaherty, J. E., and Gervasio, L. G. (2005). New challenges in dynamic load balancing. *Appl. Numer. Math.*, 52(2–3):133–152.

Faanes, G., Bataineh, A., Roweth, D., Court, T., Froese, E., Alverson, B., Johnson, T., Kopnick, J., Higgins, M., and Reinhard, J. (2012). Cray cascade: A scalable hpc system based on a dragonfly network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, Los Alamitos, CA, USA. IEEE Computer Society Press.

George Karypis and Vipin Kumar (1998). Multilevel k-way Partitioning Scheme for Irregular Graphs. *Journal of Parallel and Distributed Computing*, 48:96–129 .

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

Jain, N., Bhatele, A., Ni, X., Wright, N. J., and Kale, L. V. (2014). Maximizing throughput on a dragonfly network. In *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14. IEEE Computer Society. LLNL-CONF-653557.

Jain, N., Bhatele, A., Robson, M. P., Gamblin, T., and Kale, L. V. (2013). Predicting application performance using supervised learning on communication features. In *ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '13. IEEE Computer Society. LLNL-CONF-635857.

Kogge, P., Bergman, K., Borkar, S., Campbell, D., Carlson, W., Dally, W., Denneau, M., Franzon, P., Harrod, W., Hiller, J., Karp, S., Keckler, S., Klein, D., Lucas, R., Richards, M., Scarpelli, A., Scott, S., Snavely, A., Sterling, T., Williams, R. S., and Yelick, K. (2008). Exascale computing study: Technology challenges in achieving exascale systems.

Langer, S., Still, B., Bremer, T., Hinkel, D., Langdon, B., and Williams, E. A. (2011). Cielo full-system simulations of multi-beam laser-plasma interaction in nif experiments. *CUG 2011 proceedings*.

Mucci, P. J., Browne, S., Deane, C., and Ho, G. (1999). PAPI: A portable interface to hardware performance counters. In *Proc. Department of Defense HPCMP User Group Conference*.

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

Streitz, F. H., Glosli, J. N., Patel, M. V., Chan, B., Yates, R. K., de Supinski, B. R., Sexton, J., and Gunnels, J. A. (2005). 100+ TFlop Solidification Simulations on BlueGene/L. In *Proceedings of the International Conference in Supercomputing*. ACM Press.

Underwood, K., Levenhagen, M., and Rodrigues, A. (2007). Simulating red storm: Challenges and successes in building a system simulation. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS '07)*.

Zheng, G., Wilmarth, T., Jagadishprasad, P., and Kalé, L. V. (2005). Simulation-based performance prediction for large parallel machines. In *International Journal of Parallel Programming*, volume 33, pages 183–207.