Power and Performance Optimization of Multi-level Exascale Networks

Ehsan Totoni and Laxmikant V. Kale

Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
E-mail: {totoni2, kale}@illinois.edu

***Abstract*—Multi-level directly connected networks had been proposed for large-scale parallel computers as those machines become larger toward Exascale. One reason is that they have low diameter and hence applications will experience lower latency on those enormous number of processors. These networks have more resources (e.g. links), so they present many opportunities and challenges for power and performance optimizations. Here, we present how to design collective communication algorithms using simulation, and discuss a case study for Alltoall communication inside a network level. Furthermore, we use application benchmarks and real applications to demonstrate the merits of dynamic power management by on/off links. IBM PERCS, which is a two-level directly connected network (similar to Dragonfly topology), is used as our concrete network example to evaluate our methods. Our results for all-to-all communication show up to 5-fold improvement for large messages over the commonly used Pairwise-Exchange algorithm. We also present from 53% to 86% power saving of network links on NAMD, MILC, jacobi4d and some of NAS Parallel Benchmarks, which can result in up to 14% to 23% total machine power and energy saving.**

I. Introduction

Single-thread performance improvement had been very limited in the past several years, however, the demand for performance is increasing every day, especially in large-scale supercomputers. Therefore, large-scale parallel computers are becoming much bigger in terms of number of processors and larger interconnection networks are being designed and deployed for those machines. However, extremely fast and scalable networks are required to achieve multi-Petaflop/s and Exaflop/s performance. Since lower latency and higher bandwidth than existing popular networks (e.g. 3D Torus) are necessary, new network topologies such as multi-level directly connected ones are being proposed and used[1].

The new designs proposed by IBM (PERCS network [2]) and the DARPA sponsored Exascale study report[3] (Dragonfly topology[4]) are multilevel directly connected networks. Nodes are logically grouped together in each level and there are all-to-all connections inside a group. Hence, nodes of a group form a clique. At a higher level, each group of the lower level becomes a node of the higher level and these new nodes are also directly connected. In this way, there will be fewer hops between nodes and applications will experience lower communication latencies.

The extensive number of all-to-all connections between nodes presents opportunities and challenges for performance and power optimizations. For example, communication algorithms can use more links for better bandwidth and higher performance. Also, dynamic power management schemes can turn off some unused links or turn down the voltage of non-critical links for power savings. In this work, we use PERCS [2] network to study multilevel directly connected networks and address power and performance optimizations. PERCS is a two-level directly connected network (similar to Dragonfly topology) and will be used in many IBM machines in the future.

For performance tuning, our approach is to use detailed simulation and analysis to tune the applications and runtimes before the machine comes online. BigSim [5] simulation framework has been developed over the years for this purpose. Designing efficient communication and collective algorithms inside runtime systems is particularly important because performance of most parallel applications depends on them. MPI_Alltoall is one of the most demanding collectives, which is used in many applications and kernels such as FFT and Matrix Transpose. Here, as a case study, we show how we identified that the standard Pairwise-Exchange algorithm for MPI_Alltoall is inefficient on the PERCS architecture and how we designed a new algorithm. The simulation-based methodology used in this work can be used to design other communication algorithms for different networks as well. Overall, our algorithm is designed for All-to-all inside a "Supernode" of PERCS and shows five-fold improvement over the older algorithm (more comprehensive discussion in [6]).

For power optimization, we suggest that turning off unused links can result in significant power savings. Interconnection networks consume a significant portion of the machine's power budget (up to 40%[7]) and most of that power (up to 65%[7]) is consumed by the links. We show that many of the links of PERCS network are unused during execution of common parallel applications. The studied applications are NAMD, MILC, ISAM, Jacobi4d and some of NAS Parallel Benchmarks. We see from 53% to 100% of the links are not used for these applications, which translates to up to 26% machine power savings if the system allows the runtime system to turn off some of the links.

A. BigSim Simulation Framework

BigSim [5] simulation framework addresses performance tuning by its unique "emulation followed by simulation" approach. The emulation runs the user application at the target scale using a much smaller machine. For example, an

application can be run on an existing one hundred thousand machine pretending to have one million cores. This is made possible using processor virtualization feature of CHARM++ and AMPI runtime systems. Other than revealing the scaling bugs by running the application at scale, emulation produces the application traces needed for simulation.

These traces contain the dependencies of computations and messages, as well as salient features of computation blocks. The simulator of BigSim uses these traces to produce different performance and timing outputs. In this work, we plugin a packet-level network model of PERCS network (which has been validated extensively [6]) inside the simulator to model the machine accurately.

B. PERCS Architecture

PERCS (Productive, Easy-to-use, Reliable Computing System) is an architecture by IBM that uses a two-level directly-connected network [2]. In PERCS, the system is divided into *supernodes*, containing 32 nodes each. Each supernode is divided into four drawers (eight nodes per drawer). 24 GB/s LLocal (LL) links connect each node to the seven other nodes in its drawer and 5 GB/s LRemote (LR) links connect it to the other 24 nodes in its supernode. 10 GB/s D links connect all supernodes to one another.

Each node contains four POWER7 chips, with total of 32 cores, and a Hub chip. The POWER7 chips are connected with 192 GB/s of bandwidth to the Hub chip. This Hub chip interfaces the node with the network.

II. RELATED WORK

Performance of large-scale parallel applications is highly dependent on collective algorithms. `MPI_Alltoall` is an important collective operation, with extensive usage in applications, such as FFT and matrix transpose. The pairwise-exchange algorithm [8] for an *all-to-all* has been found to achieve better results on most machines. In each step of the algorithm, $P/2$ pairs of tasks perform a tightly coupled send-receive operation. The communication pattern has been found to have minimal congestion for topologies like torus and fat-trees, with a small number of independent paths between nodes. However, as we show, direct implementations of the algorithm will perform poorly on the PERCS network. Our new algorithm will have up to five-fold improvement for large messages.

Power management of interconnection networks using on/off links had been studied before[7]. However, their approach is implemented in hardware and will cause considerable delay for applications, because hardware does not have enough information of the application for effective power management of the links. Soteriou et al.[9] show severe possible performance impacts of hardware approaches and propose parallelizing compilers for power management of the links. However, parallelizing compilers are not widely used because of their limited effectiveness and most parallel applications are created using parallel programming models. Furthermore, compilers do not have enough information about input dependent message flow of the application and cannot manage the power effectively. Our approach of using the runtime system for network power management avoids latencies in most cases and results in greater power saving.

III. ALLTOALL OPTIMIZATION

`MPI_Alltoall` is an important collective operation, which is used in many parallel applications and kernels such as FFT and Matrix Transpose. We narrow our focus to All-to-all of large messages inside a supernode, because of its many practical interests. The Pairwise-Exchange algorithm is the dominant approach for all-to-all of large messages, which is based on tightly coupled send-receive operation of $P/2$ pairs of tasks in each step. Figure 1(a) is BigSim's output that shows link utilizations of three different links of a node at the same time for this algorithm. It can be seen that Pairwise-Exchange algorithm does not utilize all the links simultaneously for the whole all-to-all duration. Thus, in our new algorithm, we try to exploit all the links of a node by having each core send to a different node in each phase of the algorithm:

1) $t = n*c$ tasks are running on n nodes with c cores each.
2) Each task has to send $t-1$ messages. Any core can reach a particular set of c cores by using the direct link between the destination node and its home node.
3) In phase i ($0 \leq i \leq n-1$), core j ($0 \leq j \leq c-1$) on every node sends data to the set of cores residing in the $((j+i) \bmod n)$th node.

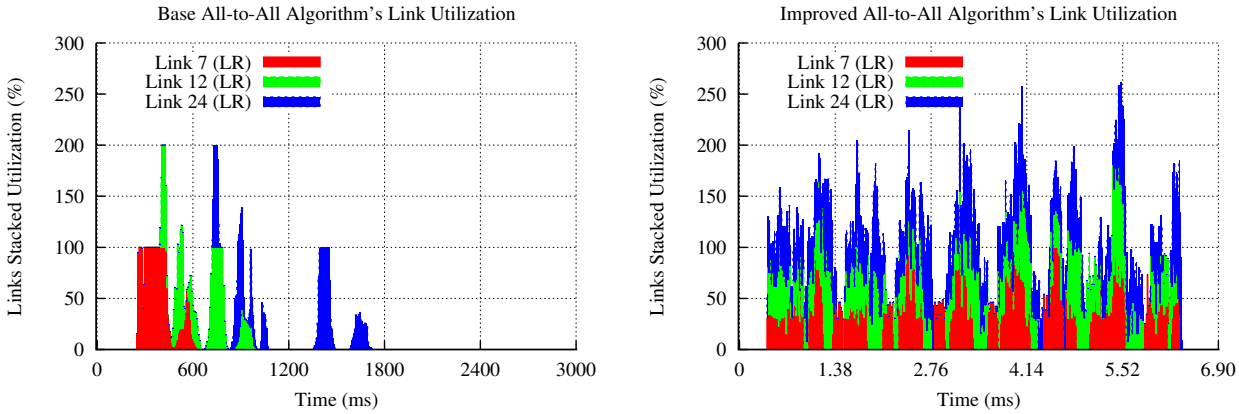
Figure 2 illustrates one phase of the algorithm by using a four node fully connected network (four cores per node). An arrow with the same color as a core shows the core's destination in that phase. Using this algorithm we obtain the link utilization graph of Figure 1(b), which shows that all the links are being used for the whole all-to-all duration.

Figure 3 compares the execution time of our algorithm with Pairwise-Exchange algorithm and theoretical peak of the links. Note that the theoretical calculation does not consider other limitations of the hardware, such as bandwidth of each node to the network and limited buffer sizes of the Hub chip. We see up to 80% reduction in execution time of all-to-all for large messages, which corresponds to a five-fold improvement.

IV. DYNAMIC POWER OPTIMIZATION

Dense networks such as multilevel directly connected ones try to be general and provide enough resources and links to support many possible communication patterns. These include very demanding communications such as all-to-all as well. However, each application has its own communication pattern, which might just use a small fraction of the network. In addition, having low latency and few hops between every node pair dictates having a lot of links, however, not every node pair of a machine communicate during execution of an application.

Interconnection networks, either in chip-multiprocessors or multi-chip machines, acquire a large fraction of the machines power budget (up to 40%[7]). In networks, the major consumers of the power are the connection links, which take up to 65% of network's power[7] (up to 26% of the whole machine). The power consumption of a link does not depend much on its usage and it consumes the same power when it is "on"



(a) Link utilization of pairwise-exchange

(b) Link utilization of new algorithm

Fig. 1. New all-to-all algorithm illustration

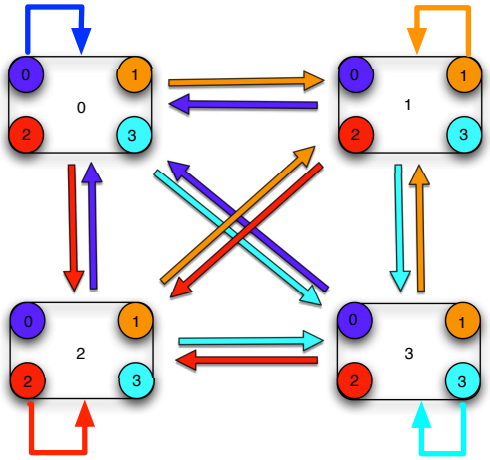


Fig. 2. Sends in first phase of the new all-to-all algorithm

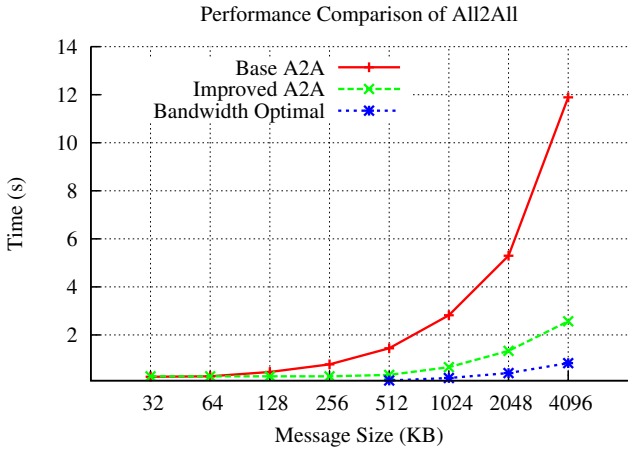


Fig. 3. Performance comparison of different algorithms for all-to-all

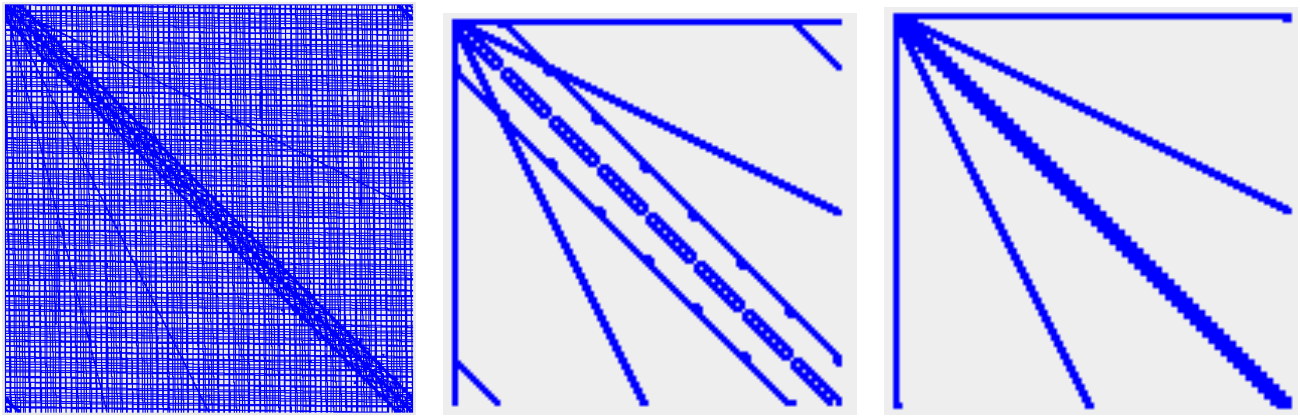
(with the same voltage). Thus, unused links represent a huge waste of power and energy in parallel computers. Knowledge of application's communication can help reduce this cost by turning off the unused links.

Figure 4 shows the communication pattern of different applications by showing the nodes that communicate together. Vertical axis represents the sender nodes while the horizontal axis represents the receiver nodes. A point is dotted if the node on vertical axis sends a message to the node on the horizontal axis at any time during the execution of an application. As can be seen, many of the node pairs never communicate during execution of an application. In addition, the number of pairs that communicate together are highly dependent on the application. For instance, in NAMD much more nodes send messages to each other than MILC. Applications that have more communicating pairs seem more likely to occupy more of the network but it highly depends on the network topology.

We use different networks such as PERCS for our study. In this work, we assume direct routing for PERCS network, which means sending each message directly to the destination supernode. Indirect routing, which uses a random intermediary supernode, is also proposed with the hope of more bandwidth and at the cost of latency and implementation overheads. Indirect routing might use more links of the network but the results should be similar. We also assume deterministic dimension order routing for torus networks.

Figure 5 shows link usage of different application, assuming fully connected network (a link between every pair of nodes) and PERCS (two-level connected). We consider a link as used if it was used at any time during execution of an application. As can be seen, link usage of each application is different and depends also on the topology. For example, BT can only use 7.7% of the links of a fully connected network, while it can use 14.2% of the PERCS links. For most applications, a larger fraction of PERCS links can be used than a fully-connected network, except CG that uses less fraction of PERCS. CG's communication is not localized (like nearest neighbor) and thus it cannot use all the links of inside a supernode. Since many destinations are inside other supernodes in this case, each supernode-to-supernode link (D-link) is on the path of possibly several messages. Nevertheless, for these applications, from 52.4% to 85.8% of the links are never used during the programs execution. This can be leveraged by the runtime system to save power.

Note also that the applications can use all the links of a



(a) NAMD 256K (b) MILC 4K (c) BT 4K

Fig. 4. Communication pattern of different applications

3D torus, which is one of the dominant topologies in the past and current supercomputers. CG is an exception, which leaves 3.1% of the links unused, but this is negligible. This happens even with deterministic routing, which can potentially exploit less links than adaptive routing. This shows that implementing on/off links for those machines is not significantly useful and probably, this is why it had not been implemented before. However, for multi-level directly connected networks such as PERCS, the benefits justifies the implementation cost of on/off links.

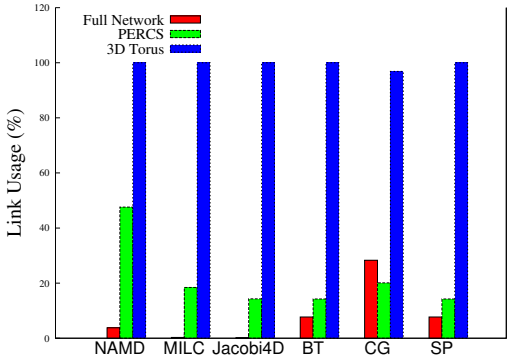


Fig. 5. Fraction of links used during execution

Other than these applications, there are cases where the network is virtually unused. Data parallel applications do not have much communications (except some startup and I/O in the beginning and at the end) and do not use the network during execution. For example, ISAM, which is a climate modeling application, only uses climate data to make progress in computation (in its standalone mode). Thus, almost 100% of the network power can be saved for these applications.

This mass of unused links presents opportunities for power optimization and savings. Although past studies suggest hardware and compiler techniques, we believe that this should be done by the runtime system. Hardware and compiler do not have enough information about the future of the application, so they would make conservative assumptions or cause unnecessary delays. For example, NAMD's communication depends on the input and previous iterations and the compiler cannot assume any unused links. It is also difficult at the application level since it would hurt portability and programmer productivity. On the other hand, the runtime system has enough information about both the application and the hardware to make wise decisions. Scientific and many other parallel applications are usually iterative in nature and near future iterations are predictable from past data, even in dynamic applications such as NAMD. The dynamic load balancing framework of Charm++, which is used in NAMD, is based on this principle of predictability. After a few iterations of the application, the runtime system has enough knowledge to start its power management.

Based on the related work, power management at the runtime system level is plausible. The hardware usually can turn links on and off with tolerable delay. If the delay is in the order of tens or about a hundred mili-seconds, even fine-grained dynamic applications such as NAMD can benefit from it. This might not be true for some optical links (usually between drawers) but must of the links can have short switching delays. Also, including instructions for link power management is feasible. Thus, the runtime system can easily perform its link power management.

In our non-data-parallel benchmark applications, 52.4% to 85.8% of the links' power can be saved, which translates to up to 13.6% to 22.3% of machines power. Since our assumptions are very conservative and only links that are never used (and are not likely to be used) are turned off, the application will not experience a significant performance penalty. Thus, this approach translates to up to 22.3% of energy savings for our benchmark applications. For data parallel applications, almost all the 26% link power budget can be saved.

V. Conclusion

With ever increasing communication demands of large-scale parallel systems toward Exascale, multilevel directly connected networks are becoming more appealing. Optimizing the power and performance of these innovative networks presents a new challenge for parallel systems. We presented the usage of detailed simulation to optimize the performance of collective algorithms and designed a new algorithm for all-to-all communication. Our algorithm is up to five-times faster than the commonly used Pairwise-Exchange algorithm

for large messages. We also showed that many parallel applications do not use a significant fraction of the network links, which suggests power optimization opportunities. Thus, the runtime system can optimize the power consumption of the links by turning off the unused ones. This approach can result in up to 86% saving of links' power for commonplace applications with nearest neighbor communication, which is about 22% of total machine's power.

As future work for performance optimization, other collective algorithm such as *all-reduce* can be designed using this simulation and measurement approach. For power optimization, less conservative approaches that turn off more links can be used, which may have some performance penalties. Furthermore, dynamic voltage scaling of network links can be exploited for the links that do not transfer the messages on the critical path. Overall, more power management techniques by the runtime system should be further explored.

REFERENCES

[1] A. Bhatele, N. Jain, W. D. Gropp, and L. V. Kale, "Avoiding hot-spots on two-level direct networks," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '11. New York, NY, USA: ACM, 2011, pp. 76:1–76:11.

[2] B. Arimilli, R. Arimilli, V. Chung, S. Clark, W. Denzel, B. Drerup, T. Hoefler, J. Joyner, J. Lewis, J. Li, N. Ni, and R. Rajamony, "The PERCS High-Performance Interconnect," in *2010 IEEE 18th Annual Symposium on High Performance Interconnects (HOTI)*, August 2010, pp. 75 –82.

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

[4] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," *SIGARCH Comput. Archit. News*, vol. 36, pp. 77–88, June 2008.

[5] G. Zheng, T. Wilmarth, P. Jagadishprasad, and L. V. Kalé, "Simulation-based performance prediction for large parallel machines," in *International Journal of Parallel Programming*, vol. 33, no. 2-3, 2005, pp. 183–207.

[6] E. Totoni, A. Bhatele, E. Bohm, N. Jain, C. Mendes, R. Mokos, G. Zheng, and L. Kale, "Simulation-based performance analysis and tuning for a two-level directly connected system," in *Proceedings of the 17th IEEE International Conference on Parallel and Distributed Systems*, December 2011.

[7] V. Soteriou and L.-S. Peh, "Exploring the design space of self-regulating power-aware on/off interconnection networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 18, no. 3, pp. 393 –408, march 2007.

[8] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in mpich," *International Journal of High Performance Computing Applications*, vol. 19, no. 1, pp. 49–66, Spring 2005.

[9] V. Soteriou, N. Eisley, and L.-S. Peh, "Software-directed power-aware interconnection networks," *ACM Trans. Archit. Code Optim.*, vol. 4, no. 1, Mar. 2007.