Toward Runtime Power Management of Exascale Networks by On/Off Control of Links

Ehsan Totoni, Nikhil Jain and Laxmikant V. Kale

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

***Abstract*—Higher radix networks, such as high-dimensional tori and multi-level directly connected networks, are being used for supercomputers as they become larger but need lower diameter. These networks have more resources (e.g. links) in order to provide good performance for a range of applications. We observe that a sizeable fraction of the links in the interconnect are never used or underutilized during execution of common parallel applications. Thus, in order to save power, we propose addition of hardware support for on/off control of links in software and their management using adaptive runtime systems.**

We study the effectiveness of our approach using real applications (NAMD, MILC), and application benchmarks (NAS Parallel Benchmarks, Jacobi). They are simulated on representative networks such as 6-D Torus and IBM PERCS (similar to Dragonfly). For common applications, our approach can save up to 16% of total machine's power and energy, without any performance penalty.

I. Introduction

Single-thread performance improvement has been very limited in the past several years. However, the demand for performance is increasing every day, especially in large-scale supercomputers. As a result, large-scale parallel computers are becoming much bigger in terms of the number of processors, and larger interconnection networks are being designed and deployed for those machines. Moreover, the many-core era with on-chip networks is rapidly approaching, which will add another level to the interconnection network of the system. The performance and power consumption of the system highly depends on these immense networks.

Power and energy consumption are major constraints of HPC systems and facilities [1]. Interconnection network is one of the major consumers of the system, along with the processor chips. For example, routers and links are expected to consume about 40% of some server blades' power, which is the same as their processor's budget [2, 3]. For current HPC systems, using an accurate measurement framework, Laros et.al. [4] report more than 25% total energy savings by shutting off some of the network resources of a Cray XT system. In the future systems, especially because of their low frequency many-cores and aggressive network designs, the network is expected to consume 30% of the system's total power [5]. From this power, up to 65% is allocated to the links and the resources associated with them [3]. In addition, up to 40% [3] of the many-core processor's power budget is expected to go to its on-chip network. Thus, saving network power is highly crucial for HPC machines.

In contrast to processors, the network's power consumption does not currently depend on its utilization [3] and it is near the peak whenever the system is "on". For this reason, while about 15% of the power and energy [6] is allocated to the network, it can go as high as 50% [7] when the processors are not highly utilized in non-HPC data centers. While the processors are not usually that underutilized in HPC data centers, they are not fully utilized all the time and energy proportionality of the network is still a problem. Therefore, it is essential to save the network's power and make it energy proportional [7], i.e. the power and energy consumed should be related to the usage of the network. As evidenced in this paper, we observe that most of the network's links are never used during execution of common parallel applications on higher radix networks and thus, they can be turned off for better energy proportionality.

Higher radix networks are emerging, as 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 for some applications, higher radix network topologies such as multi-level directly connected ones [8, 9] and high-dimensional tori [10][11] are being proposed and used. However, these networks are designed to provide enough bisection bandwidth for the worst case, such as all-to-all communication, but not all applications need that. Furthermore, the intention is to provide low latency for all applications, but communicating node pairs of different applications vary. Therefore, the network would provide small hop count and low diameter for any pairs of nodes, which would leave some part of the network unused in each application. This results in the fact that many application do not use a large fraction of higher radix networks. Thus, we propose addition of hardware support for exposure of on/off links (links that can be turned on and off) to the runtime system, so that it can save the wasted power and energy consumption. We also discuss how the runtime would accomplish that, considering its design and the applications' behavior.

The applications we use for our study are NAMD [12], MILC [13], ISAM [14], Jacobi benchmarks (representing nearest neighbor communication patterns) and some of NAS Parallel Benchmarks [15]. For commonly used nearest neigh-

bor applications such as MILC, using our basic approach, 81.5% of the links can be turned off for a two-level directly connected network (around 16% of total machine power) and 20% for 6D Torus (Sections III and IV). Adaptive runtimes have also been shown to be effective for load balancing and power management (using DVFS) [16] and our approach makes use of the same infrastructure. We discuss why the hardware and compiler cannot perform this task effectively and it should be done by the runtime system. We also show that different mappings can affect network's power consumption.

A. Related Work

Power consumption of interconnection networks in supercomputers, distributed systems and data centers has received special attention in recent time. Several techniques have been proposed for reduction of network power in non-HPC data centers [7, 6]. Intelligent power aware job allocation and traffic management form the basis of these approaches. Hoefler [17] provides an overview of the power problem, and the related aspects of interconnect power, with focus on supercomputers. Laros et.al. [4] present results on potential power saving using CPU and network scaling, by post processing the data collected from the monitoring system of Cray XT machines. Their work, using real systems (instead of simulations and projections) and real applications, shows the importance and potential of network power management for supercomputers.

Among hardware based approaches, power management of interconnection networks using on/off links has been studied [3, 18, 19]. On/off links, which refers to shutting down communication links that are not being used, has shown to be a useful method to save power. In addition, PowerHerd [20] uses routers as controllers to constrain network power consumption. However, dependence on hardware for power management may cause considerable delay for some applications. Additionally, hardware does not have enough global information about the application to manage network power effectively.

Soteriou et.al. [21] show severe possible performance penalty of hardware approaches, and propose the use of 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 explicit parallel programming models. Furthermore, compilers do not have information about input dependent message flow of an application, and cannot manage the power effectively for such applications.

As an alternative to hardware and compiler driven power management, we advocate network power management by the run-time system. Limited network power management by the run-time system, such as for collective algorithms, has been proposed in the past. Dong et.al. [22] present a metric to estimate network power consumption of various MPI collective algorithms at run-time, and recommend selection of the algorithm with least power consumption. Limited power management using on/off links in the run-time system has also been studied [23]. However, that approach is limited to management of network links only during collective operations in MPI. In this paper, we propose usage of an adaptive runtime system to manage the power of network links using on/off control, taking into account all of the communications performed by an application.

Network Power Management Support on Current Machines: Unfortunately, network power management support on current HPC machines is very limited. For example, It is possible to reduce link and node injection bandwidth on a Cray XT system (effectively turning off some portion of the links), but it requires a reboot of the whole machine [4]. Thus, using this feature is mostly impractical. However, techniques such as on/off links have been implemented before, and it seems feasible to include them for HPC machines as well. For instance, some commercial systems [1] can disable some of the board-to-board and box-to-box links to save power. Currently it takes 10,000 cycles to turn the links on/off, but it can be improved much further [3].

B. Exascale Networks

In this section, we briefly describe n-dimensional tori and two-tier direct networks, which seem to be the front runners on the road to Exascale.

n-dimensional tori have been used in many supercomputers such as IBM Blue Gene series, Cray XT/XE, and the K computer. An n-dimensional torus strike a good balance in terms of bisection bandwidth, latency, and the link cost, and have been shown to be scalable. In recent times, most vendors have increased the torus dimensionality from three (as it is in IBM BlueGene/P and Cray XT/XE) to five (IBM BG/Q) and six (the K computer). This shift is necessary in order to keep latency low, with possible increase in the bisection bandwidth. We present analysis and results for link utilization of an n-dimensional torus, with n varying from 3 to 10.

Two-tier direct networks have been proposed by IBM (PERCS network [8]), the DARPA sponsored Exascale study report [1] (Dragonfly topology [24]), and Cray (Aries network [25]). In all of these proposals, similar multi-level directly connected networks have been described. In these networks, nodes are logically grouped together at two levels, in each of which nodes (or the grouped entities from previous level) are connected in an all-to-all manner. Hence, in the first level a clique of nodes is formed, and in the second level, a clique of cliques (from the first level) is constructed. The resultant network, with its large number of links, boasts of a large bisection width. At the same time, the latency of the entire system is low (3 hop connectivity between any two nodes). The small number of hops make this topology very appealing, as the expensive conversion between electrical and optical signals (if different types of links for global and local connections are used) needs to be done very few times in comparison to networks with multiple hops. Currently, this network is used in some large-scale IBM Power 775 machines [2].

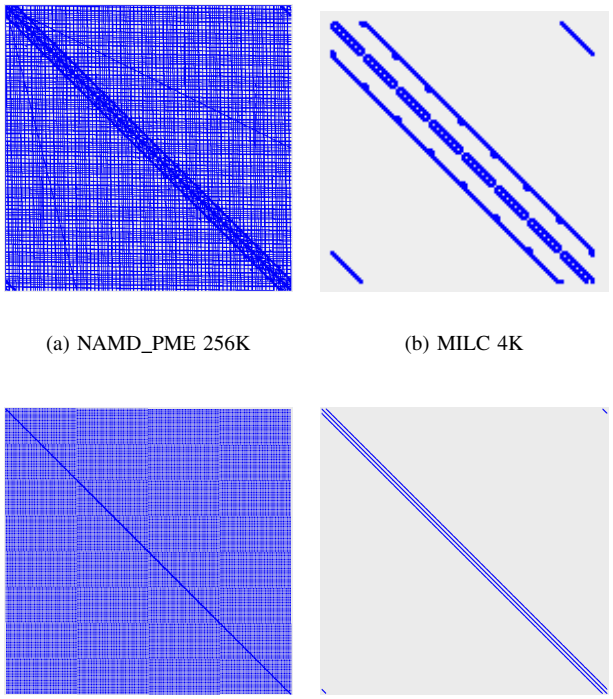
[1] Motorola MC92610 WarpLink 2.5 Gb/s Quad SERDES Transceiver, Motorola Inc., www.motorola.com

[2] www.top500.org

We observe that the two networks we present results on, two-tier networks and torus with high dimensionality, have a large number of links. The presence of these links provides an opportunity for high performance as well as a challenge for power and energy proportionality. In the later sections, we present results which attempt to address this challenge.

C. Application Communication Patterns

Interconnection networks are designed to serve a range of communication patterns, in terms of bandwidth and latency. However, each application has its own communication pattern, so many node pairs of a system may not communicate during execution of an application, leaving a large fraction of the network unused.



(a) NAMD_PME 256K (b) MILC 4K

(c) CG 64K (d) MG 64K

Fig. 1. Communication pattern of different applications

Figure 1 shows the communication pattern of some of the applications we use in this paper. Both the vertical and horizontal axes represent the nodes in the system. A point (x, y) is marked if the node y on the vertical axis sends a message to the node x on the horizontal axis, during the execution of an application. Each marked point has been enlarged for better illustration. It can be seen that many of the node pairs never communicate during execution of an application. Moreover, the number of pairs that communicate varies significantly with the applications. For instance, in NAMD_PME and CG, the number of node pairs that communicate is much larger than in MILC and MG.

Most of the communicating pairs in NAMD_PME are due to the FFT performed in the PME phase, which is done once every four iterations. Without the PME option, NAMD has a near neighbor communication pattern, which can be seen in the dense region around the diagonal of Figure 1(a). CG, on the other hand, has a more uniform and dense communication pattern. Applications like NAMD_PME and CG, that have large number of communicating pairs are more likely to use most of the network.

On the other hand, the number of communicating pairs in MILC (Figure 1(b)) and MG (Figure 1(d)) are few, and concentrated near the diagonal. As such, these applications are expected to make use of a small fraction of the available network links. These applications represent a large class of applications in science and engineering, such as the ones following the nearest neighbor pattern.

Note that all the illustrated cases have a dense region close to the diagonal of their communication graph. It suggests that near neighbor communication constitutes a major part of the applications' communication. This can be used as a clue in understanding a network link's usage. We use Jacobi, decomposed in two, three and four dimensions, to study network's link usage for near neighbor communication patterns. From this discussion, we also conclude that there is an extensive opportunity to save the power of the network links in higher-radix topologies in many usual cases, since they are designed for the worst cases with many communicating pairs.

The cases on the other extreme are embarrassingly parallel applications that essentially do not rely on the network during their computation. Thus, they do not use any of the links, and the link power can be saved easily.

II. Assumptions About the System and Network

In this paper, we refer to on/off links in discussions but the same principles can be applied to other power management schemes as well. For example, when a link is in the power saving state, it can be completely "off", or in some other intermediate idle state, or lower voltage state. Thus, it depends on the underlying hardware features for power management. However, software control to those hardware features is necessary, which we assume in our discussions.

For the simulated systems, we assume 32 cores per node, which is the core count for the upcoming Blue Waters system and for the initial design of PERCS. This is a reasonable number for the system sizes of up to 300K cores that we use in this paper. For future machines, the number of cores per node is expected to increase, but the results will be similar to the one presented in this paper. Moreover, we only consider the network across nodes in our results. The intra-node network, such as network-on-chip, can be considered as another level of the network with comparable results.

As mentioned earlier, we use different networks such as PERCS/Dragonfly and Torus for our study. We assume default mapping for processes to processors for all the networks. For our 32 cores per node case, it means that first 32 processes are mapped to the first node and so on. This is the default setting of most supercomputer sites. We assume direct routing for PERCS, which means that the message is sent directly to the

receiver, instead of going through an intermediate supernode (i.e. indirect routing). Effect of different mappings and indirect routing are discussed and evaluated in Subsection III-A. For tori, we assume minimal dimension order routing, which is used in many of the current supercomputers. Adaptive routing is also supported in some machines with tori such as BlueGene/P. Evaluating the gains for a network with adaptive routing is more complicated and we leave it for future work. Nevertheless, it makes the usage of the run-time system for link power management difficult, since it requires low level information to be exposed to the run-time system.

Throughout the paper, unless otherwise mentioned, we assume the same power cost for different links of the network for simplicity. This may not hold true in some cases; for example, the power cost of local links inside a supernode of PERCS, may be different from the global links across supernodes. Many times, the underlying technology is also different for various links (e.g. electrical for local and optical for global). Nevertheless, our estimate is a good conservative approximation, and we observe that it is pessimistic, since more costly links are used infrequently. We also assume that links consume their full power during their on/off transition and zero power in their off state (same as our related work [3]).

In this paper, we only consider the application's messages, and ignore the control messages of the run-time system for several reasons. The run-time system messages are usually not performance critical, and can be re-routed through other nodes if a link was turned off. Alternatively, if the transition delay is high, the runtime system can maintain a connected tree to ensure connectivity. Such a control tree, if used, has low power overhead as it only takes $n-1$ (n is the number of nodes) links in total. Moreover, many of these links can be among the links used by the application. We also performed experiments taking into account these control messages, and found that addition of control message adds a small percentage to the link usage of applications.

Finally, we consider a link as used if it transfers a message even once during the execution of the application. In our basic approach, we do not turn these used links off, even when they are used only for a small time period. We make this pessimistic assumption because the switching delay is technology dependent and not easy to determine.

III. Potentials of Basic Network Power Management

Interconnection networks, consume a large fraction of the machine's power budget, and the major consumers of that power are the connection links. The power consumption of a link does not usually depend much on its usage and it consumes the same power when it is "on" (with the same voltage). Thus, unused links represent a huge waste of power and energy in parallel computers. Knowledge of the application's communication can help in reducing this cost by turning off the unused links. In Subsection I-C, we showed that the number of communicating pairs for an application is not large, which indicates that a sizeable fraction of link may be unused. In this section, we discuss and evaluate our basic approach of link power management inside the adaptive runtime system.

Figure 2 and Figure 3 show link usage of different applications and benchmarks, assuming a fully connected network (a link between every pair of nodes), 3D Torus, 6D Torus and PERCS (two-level fully-connected). Note that the full network is an asymptotic case that is not usually reached by the large-scale networks. However, for example, small jobs (less than 1k cores for PERCS) running on a two-tier system will have a fully connected network. In this case, most of the links can be shut down according to our results, which saves a significant fraction of the system's power.

As can be seen, link usage of each application is different, and also depends on the topology of the system. For example, MILC only uses 3.93% of the links of a fully connected network, while it uses 80% of the 6D Torus links. For most applications, a larger fraction of 6D Torus links will be used than PERCS network, except CG that uses more fraction of PERCS. This shows that analysis of the link utilization of different networks is not trivial and depends on various aspects of the topology and the application.

For the applications of Figure 2, from 4.4% to 82.94% of the links are never used during the program's execution on PERCS. In the Jacobi benchmarks of Figure 3, Jacobi2D only uses 11.91% of the PERCS links, with similar numbers for other Jacobi dimensions. This demonstrates a great opportunity for the runtime system to save link power of two-level directly connected networks.

There are good saving potentials for 6D Torus as well. Even though NAMD uses all of the links, MG leaves 47.92% of the links unused. However, many applications can use most of the links of a 3D torus, which has been one of the dominant topologies in the past and in the current supercomputers. There are saving potentials for some cases (30.67% for Jacobi2D) but not as high as other networks. This happens even with deterministic routing, which uses fewer links than adaptive routing. This shows that implementing on/off links for those networks is not significantly useful, and probably is the reason that they have not been implemented before. However, for high dimensional tori and multi-level directly connected networks, the benefits justify the implementation cost of software controlled on/off links. If we take MILC to represent common HPC applications (which usually have near neighbor communication), 81.51% of PERCS links and 20% of 6D Torus links can be saved for power easily. Assuming that 65% of network power goes to links and the network consumes 30% of the total machines power, around 16% of total machines power is saved for PERCS systems and around 4% is saved for 6D Torus systems.

In NAMD_PME, the communication intensive PME calculation is usually performed every four iterations, which may take around 80ms assuming 20ms per iteration. In this case, many links can be turned off after PME, and turned back on right before that. Thus, for link power management to be effective without performance penalty, the switching delay should be much less than that. This low delay seems hard

to achieve for global links (which are typically optical), but easier for local links (which are typically electrical). Thus, even for communication-intensive applications, there can be savings with good hardware.

Note that even though Jacobi3D has a 3D communication pattern, when it is mapped to a system with 32 cores per node, the communication between nodes is not an exact 3D pattern anymore. Thus, some fraction of the links (12%) are not used.

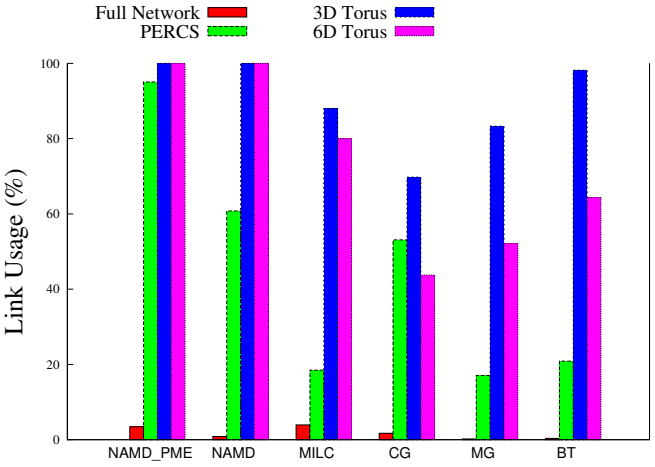


Fig. 2. Fraction of links used during execution

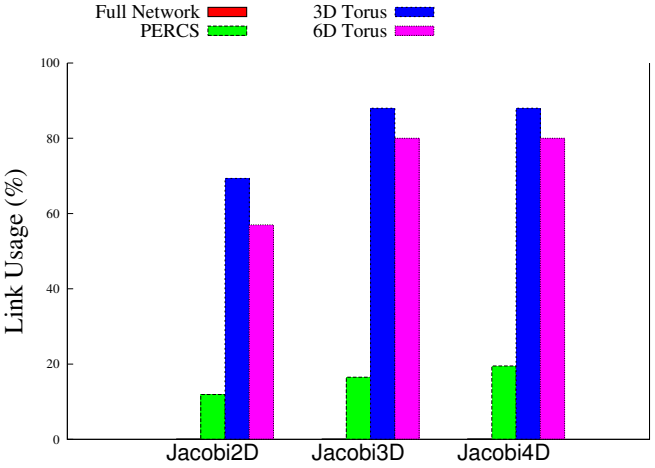


Fig. 3. Fraction of links used during execution

Figure 4 and Figure 5 show the usage of different types of PERCS links. As can be seen, D-links, which are power hungry global links, are usually less utilized than the local LL and LR links. This happens because most of the communication of the applications is local or near neighbor exchanges. Even though NAMD_PME has less savings in general, it can save 23.09% of D-links because of that. MILC shows high usage of D-links, because the results are for 4K processors (4 supernodes) and there are just 12 D-links. For larger configurations, it should be similar to Jacobi4D and have a very low D-link utilization. CG is again an exception and it uses more of the D-links. This is because its communication is not local but distributed as mentioned earlier. The Jacobi benchmarks only use around 1% of D-links and most of those links can thus be turned off safely. Thus, our simple model of same power cost for all links is pessimistic and the actual savings are much higher in many cases, as the power hungry D-links have less utilization.

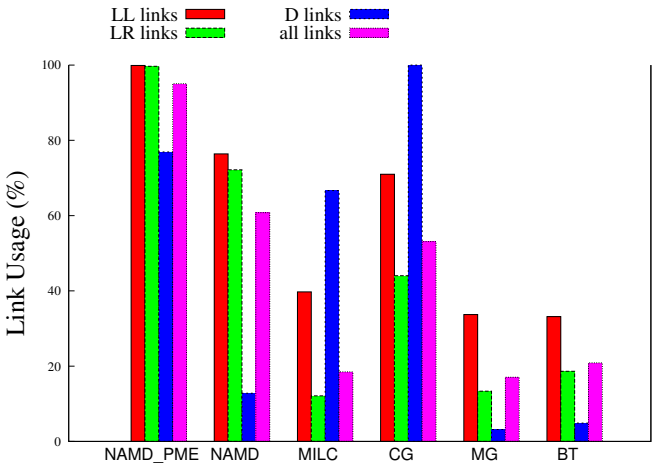


Fig. 4. Fraction of different links used during execution on PERCS

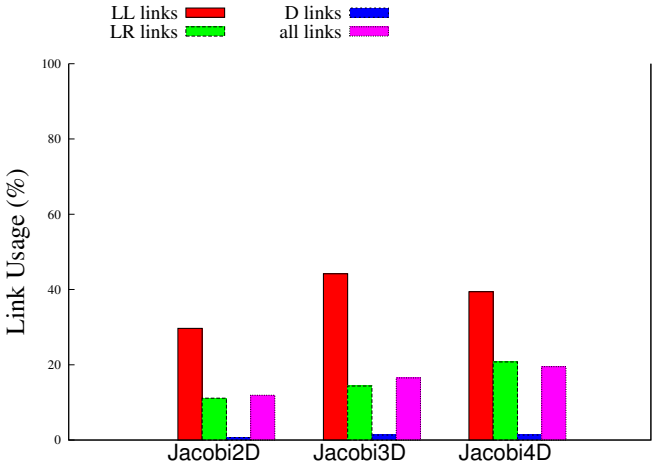


Fig. 5. Fraction of different links used during execution on PERCS

Figure 6 and Figure 7 show the link usage of the applications on tori with different dimensions, from 3 to 10. As the dimension goes higher, less fraction of the links are used, which is intuitively expected. For example, Jacobi4D uses only 53% of the links of a 10D torus network. Even NAMD that does not have any savings on low dimensional torus, shows potential on a torus with sufficiently high dimensions, starting from 7D. It uses 65% of the links of a 10D torus, which shows that even such applications have potential of link power saving on high dimensional tori.

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 [14], which is a climate modeling application, only uses stored climate data to do the

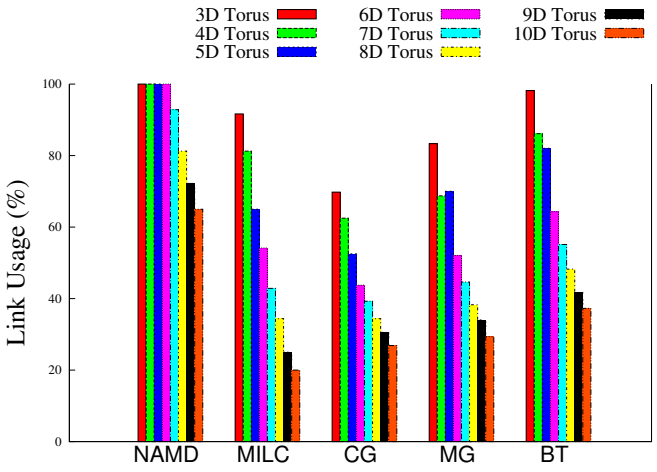


Fig. 6. Fraction of links used during execution

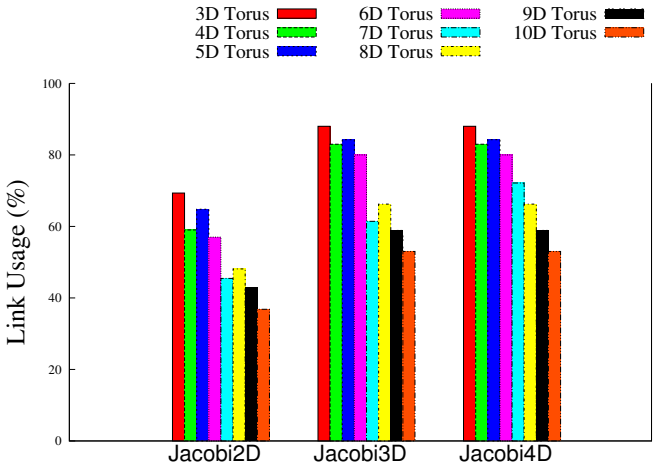


Fig. 7. Fraction of links used during execution

computation (in its standalone mode). Thus, almost 100% of the network power can be saved for these applications.

To summarize, in the common near neighbor applications like MILC, up to 16% of total machines power can be saved (assuming 30% network power budget and 65% of network power associated with links) using a basic power management approach. 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. For data parallel applications, almost all the 30% network power budget can be saved.

A. Different Mappings

In the results so far, we assumed 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 purpose of more bandwidth and at the cost of latency and implementation overheads. A previous study on PERCS network [9] suggests using random mapping or indirect routing for the PERCS network for better link utilization and bandwidth.

Figure 8 shows the link usage of these schemes compared with the default. Random mapping has higher link usage than default mapping, which is intuitive. It can use 33.18% of the links, which is higher than 16.51% of the default. However, the overall usage is still low and the possible savings are as high as around 67%. Note that this scheme uses many more D-links, which may increase the power consumption significantly. Thus, when choosing among different mappings for future machines, power consumption should also be taken into account and some (possibly small) performance gains might be power costly.

Indirect routing uses all of the links of the network, since every small packet is routed through a random supernode. Therefore, it is very expensive in terms of power and no saving is possible. However, random mapping is shown to have similar performance as indirect routing on PERCS networks [9]. Therefore, indirect routing should be avoided and random mapping should be used to have much less power consumption but the same performance. Thus, different aspects of hardware and software design can affect power consumption of the network significantly and power should be considered at every stage of the design.

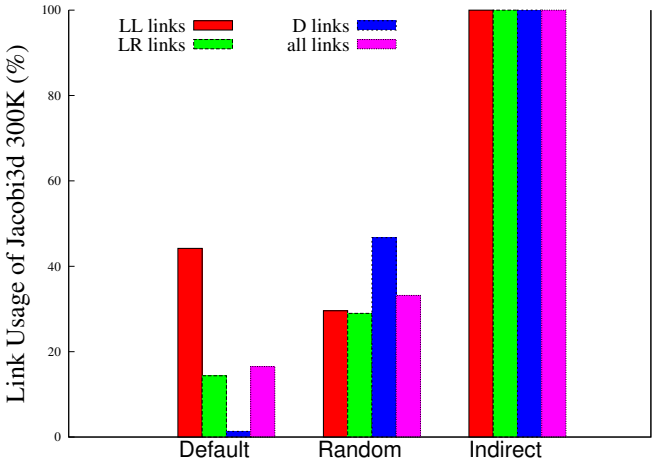


Fig. 8. Fraction of links with different mappings

IV. Implementation in Runtime System and Hardware

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 hence 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, a powerful runtime system, such as that of Charm++, has enough information about both the

application and the hardware to make wise decisions. The runtime system obtains this information about the application by monitoring the communication performed as the application executes. For our discussions, we focus on CHARM++ runtime system, which performs continuous introspection by instrumenting all the computation and communication. Addition of monitoring components to any runtime system including MPI is possible, and causes very small overheads. We expect other runtime systems to become more sophisticated like CHARM++, as we move towards Exascale.

A. Runtime System Support

CHARM++ runtime system mediates all the communications and computation, so it can instrument the application easily. It uses these informations for many purposes such as load balancing [26] and power management [16]. It also obtains the network's characteristics such as its topology [27]. Here, we only need a small subset of these data to save network power, which is the communication graph of the application and the topology of the network.

Using this information, our approach can turn off unnecessary links as follows. We assume that each node keeps track of the destinations of its messages. At the network power management stage, each node calculates the route for each of its destinations. It sends a message to each of the intermediate nodes to have them mark their used links. At the end, when it has received all its messages and marked its own links, it turns off all of its unused links. This sequence is summarized in Algorithm 1.

```
Algorithm: Network power management
// Each node runs this
Input: list of destinations of local messages
Result: Unused links are turned off
for each destination D do
    calculate route R for reaching D
    mark used local links
    for each intermediate node N do
        ask N to mark the required links
    end
end
wait for all messages to be received
turn unused local links off
```

Algorithm 1: Network power management by on/off links

This algorithm needs to be invoked at appropriate times, which is feasible in most case since scientific and many other parallel applications are usually iterative in nature. For the common case of static communication pattern, which encompasses all of our benchmarks except NAMD, even one iteration represents the application's communication pattern afterwards. Thus, one invocation (e.g. after the first iteration) is enough. Note that even in this simple case, the hardware cannot make wise decisions on its own, because it is not aware of the iteration time of the application and its window might be too small. In addition, hardware does not see the global picture of the application's message flow and it even usually works at the packet and flit levels.

For NAMD, the communication pattern between objects is static, even though the objects may migrate between processors and the actual communication pattern varies. In this case, the new communication pattern can be determined by the runtime system at (or after) load balancing steps. Thus, the network power management algorithm needs to be called at every load balancing step. Even in this case, the switching delay of links does not matter much, since the runtime will not make any mistake in switching the links' states. In addition, since load balancing is not performed very frequently (usually once in thousands of iterations), our method will not add significant overhead. In the same way may other dynamic (and phase based) applications, such as Adaptive Mesh Refinement (AMR) [28], can be handled as well.

B. Hardware support

For our approach, we only demand the network hardware to implement links that can be turned off and on (or any other power saving means such as DVS), along with a software interface to control them. We do not strictly require any change in routing and switching tables, because we can handle different possible (but rare) issues in software. One issue is the case of an exception message, which is a message that needs an off link in its route. This can happen when the application needs to send a message for a special condition in the physical simulation, for instance. The runtime needs to inspect every message that is being sent out of a node. If the destination was not in the list of "allowed" ones, it should handle the exception. It can either turn the link on or reroute the message manually. If the hardware provides DVS of links instead of turning them off completely, no action is required by the runtime systems at some performance cost. Choosing between these options depends on their overheads and presents a tradeoff.

As mentioned, the runtime keeps a tree for connectivity so it can be used for rerouting exception messages as well. If exception messages appear too frequently, they can degrade the performance significantly. This might be a result of frequent change in the communication pattern (as for the contagion application mentioned above) or not having enough information at the runtime system. We believe that these cases are rare, and we leave their evaluation for future work. Thus, software can solve connectivity and routing issues, which are the main barriers for the hardware implementation of on/off links.

V. CONCLUSIONS AND FUTURE WORK

With ever increasing communication demands of large-scale parallel systems toward Exascale, multilevel directly connected networks and high dimensional tori are becoming more appealing. Optimizing the power and performance of these innovative networks presents a new challenge for parallel systems. We showed that many parallel applications do not fully exploit a significant fraction of the network links,

which present opportunities for power optimization. Thus, the runtime system can optimize the power consumption of the links by turning off the unused ones, with minimal hardware support. This approach results in up to 16% saving of total system's power for common place applications with near neighbor communication.

For future work, less conservative approaches that turn off more links can be used, which may have some performance penalties. Furthermore, dynamic voltage scaling (or reducing the bandwidth) of the network links can be exploited for the links that do not transfer messages on the critical path. Overall, more adaptive power management techniques by the runtime system should be explored further.

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] Li Shang, Li-Shiuan Peh, and N.K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*, pages 91–102, feb. 2003.

[3] V. Soteriou and Li-Shiuan Peh. Exploring the design space of self-regulating power-aware on/off interconnection networks. *Parallel and Distributed Systems, IEEE Transactions on*, 18(3):393–408, march 2007.

[4] James Laros, Kevin Pedretti, S. Kelly, Wei Shu, and C. Vaughan. Energy based performance tuning for large scale high performance computing systems. In *Proceedings of 20th High Performance Computing Symposium*, HPC, 2012.

[5] PM Kogge. Architectural challenges at the exascale frontier (invited talk). *Simulating the Future: Using One Million Cores and Beyond*, 2008.

[6] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. Energy aware network operations. In *INFOCOM Workshops 2009, IEEE*, pages 1–6, april 2009.

[7] Dennis Abts, Michael R. Marty, Philip M. Wells, Peter Klausler, and Hong Liu. Energy proportional datacenter networks. In *Proceedings of the 37th annual international symposium on computer architecture*, ISCA '10, pages 338–347, New York, NY, USA, 2010. ACM.

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

[9] Abhinav Bhatele, Nikhil Jain, William D. Gropp, and Laxmikant V. Kale. Avoiding hot-spots on two-level direct networks. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 76:1–76:11, New York, NY, USA, 2011. ACM.

[10] M. Yokokawa, F. Shoji, A. Uno, M. Kurokawa, and T. Watanabe. The k computer: Japanese next-generation supercomputer development project. In *Low Power Electronics and Design (ISLPED) 2011 International Symposium on*, pages 371 –372, aug. 2011.

[11] Y. Ajima, Y. Takagi, T. Inoue, S. Hiramoto, and T. Shimizu. The Tofu interconnect. In *High Performance Interconnects (HOTI), 2011 IEEE 19th Annual Symposium on*, pages 87–94, aug. 2011.

[12] Laxmikant V. Kale, Abhinav Bhatele, Eric J. Bohm, and James C. Phillips. NAnoscale Molecular Dynamics (NAMD). In D. Padua, editor, *Encyclopedia of Parallel Computing (to appear)*. Springer Verlag, 2011.

[13] Claude Bernard, Tom Burch, Thomas A. DeGrand, Carleton DeTar, Steven Gottlieb, Urs M. Heller, James E. Hetrick, Kostas Orginos, Bob Sugar, and Doug Toussaint. Scaling tests of the improved Kogut-Susskind quark action. *Physical Review D*, (61), 2000.

[14] A.K. Jain and X. Yang. Modeling the effects of two different land cover change data sets on the carbon stocks of plants and soils in concert with CO2 and climate change. *Global Biogeochem. Cycles*, 19(2):1–20, 2005.

[15] D.H. Bailey, E. Barszcz, L. Dagum, and H.D. Simon. NAS parallel benchmark results. In *Proc. Supercomputing*, November 1992.

[16] Ehsan Totoni Osman Sarood, Phil Miller and L. V. Kale. 'Cool' Load Balancing for High Performance Computing Data Centers. In *IEEE Transactions on Computer - SI (Energy Efficient Computing)*, September 2012.

[17] T. Hoefler. Software and hardware techniques for power-efficient HPC networking. *Computing in Science Engineering*, 12(6):30–37, nov.-dec. 2010.

[18] Marina Alonso, Salvador Coll, Juan-Miguel Martínez, Vicente Santonja, Pedro López, and José Duato. Dynamic power saving in fat-tree interconnection networks using on/off links. In *Proceedings of the 20th international conference on Parallel and distributed processing*, IPDPS'06, pages 299–299, Washington, DC, USA, 2006. IEEE Computer Society.

[19] Jian Li, Wei Huang, C. Lefurgy, Lixin Zhang, W.E. Denzel, R.R. Treumann, and Kun Wang. Power shifting in thrifty interconnection network. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pages 156 –167, feb. 2011.

[20] Li Shang, Li-Shiuan Peh, and N.K. Jha. Powerherd: a distributed scheme for dynamically satisfying peak-power constraints in interconnection networks. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(1):92 – 110, jan. 2006.

[21] Vassos Soteriou, Noel Eisley, and Li-Shiuan Peh. Software-directed power-aware interconnection networks. *ACM Trans. Archit. Code Optim.*, 4(1), March 2007.

[22] Yong Dong and Juan Chen. Network energy optimization for MPI operations. In *Intelligent Computation Technology and Automation (ICICTA), 2012 Fifth International Conference on*, pages 221 –224, jan. 2012.

[23] S. Conner, S. Akioka, M.J. Irwin, and P. Raghavan. Link shutdown opportunities during collective communications in 3-D torus nets. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1 –8, march 2007.

[24] John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. Technology-driven, highly-scalable dragonfly topology. *SIGARCH Comput. Archit. News*, 36:77–88, June 2008.

[25] Keynotes - hoti 2012 [four abstracts]. In *High-Performance Interconnects (HOTI), 2012 IEEE 20th Annual Symposium on*, pages xv –xix, aug. 2012.

[26] Gengbin Zheng, Abhinav Bhatele, Esteban Meneses, and Laxmikant V. Kale. Periodic Hierarchical Load Balancing for Large Supercomputers. *International Journal of High Performance Computing Applications (IJHPCA)*, March 2011.

[27] 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.

[28] Akhil Langer, Jonathan Lifflander, Phil Miller, Kuo-Chuan Pan, Laxmikant V. Kale, and Paul Ricker. A Scalable Mesh Restructuring Algorithm for Distributed-Memory Adaptive Mesh Refinement. In *Proceedings of 24th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2012.