Are Nonblocking Networks Really Needed for High-End-Computing Workloads?

N. Desai,*[1] P. Balaji,*[2] P. Sadayappan,†[3], M. Islam†[4]

*Mathematics and Computer Science Division
Argonne National Laboratory, Argonne, IL 60439, USA
[1]desai@mcs.anl.gov
[2]balaji@mcs.anl.gov

†Department of Computer Science and Engineering
Ohio State University, Columbus, Oh 43210, USA
[3]saday@cse.ohio-state.edu
[4]islammo@cse.ohio-state.edu

***Abstract*—High-speed interconnects are frequently used to provide scalable communication on increasingly large high-end computing systems. Often, these networks are nonblocking, where there exist independent paths between all pairs of nodes in the system allowing for simultaneous communication with *zero* network contention. This performance, however, comes at a heavy cost as the number of components needed (and hence cost) increases superlinearly with the number of nodes in the system.**

In this paper, we study the behavior of real and synthetic supercomputer workloads to understand the impact of the network's nonblocking capability on overall performance. Starting from a fully nonblocking network, we begin by assessing the worse-case performance degradation caused by removing interstage communication links, resulting in overprovisioning and hence potentially blocking in the communication network. We also study the impact of several factors on this behavior, including system workloads, multicore processors, and switch crossbar sizes. Our observations show that a significant reduction in the number of interstage links can be tolerated on all of the workloads analyzed, causing less than 5% overall loss of performance.

I. Introduction

As high-end computing (HEC) systems continue to grow in size and scale, it is a generally accepted notion that the network that connects these systems also has to grow to allow for improved performance and scalability. High-speed interconnects, such as InfiniBand (IB), Myrinet, 10-Gigabit Ethernet and Quadrics, are popular choices for connecting machines on modern HEC systems. These networks are composed of small building blocks known as crossbars or switch building blocks (SBBs) that connect together to form the overall network. For IB and Myrinet, 16-port and 24-port crossbars are common today; switches with larger port counts are created by interconnecting these crossbars internally using an efficient network topology such as the Clos network.

To achieve the best communication performance, several HEC systems use completely nonblocking network topologies, where there exist independent paths between all pairs of nodes in the system, allowing for simultaneous communication with *zero* network contention. Unfortunately, to scale a network linearly with the system size, while retaining such nonblocking capabilities increases the number of network components (specifically the number of crossbars and cables) superlinearly. For example, in order to connect 16 nodes in a completely nonblocking manner, one single 16-port crossbar is required in a Clos network; for 32 nodes, four such crossbars would be required. The SUN Enterprise 3456 switch (formerly known as the Magnum switch) is another such example. SUN Magnum is a five-stage Clos network, with each stage containing 288 24-port crossbars, to only create a 3456-port IB switch. That is, the number of internal ports used for inter-crossbar connectivity is an order of magnitude higher than the number of external ports to which nodes can be connected. When we move forward to systems with tens of thousands of nodes, this problem will only get worse. Further, since the crossbars are the basic building blocks of a network fabric, the cost of the network fabric increases superlinearly as well.

In this paper, we quantify the performance benefits of such nonblocking topologies compared to less expensive blocking network topologies. Specifically, a topology that reduces the number of interstage links by a factor of two can halve the cost of the network components. However, every job that is scheduled on the system is not evenly slowed down by a factor of two. Jobs that are allocated such that all their processes are on nodes that are connected to a single crossbar do not experience any slowdown, since the crossbar itself is completely nonblocking. That is, on a 24-port crossbar, any job that uses less than 24 processes can potentially be fully scheduled on a single crossbar. With multicore or multiprocessor architectures, even larger jobs can be scheduled on the same crossbar (for dual-processor quad-core nodes connected to a 24-port crossbar, up to 192 process jobs can be scheduled on the same crossbar). Depending on the actual communication percentage of the job, the potential for contention further reduces rapidly.

Based on the above observations, it is not very clear whether, in practice, such network overprovisioning actually causes significant performance degradation in real workloads. Accordingly, we study the impact of various parameters discussed on network contention and overall job slow-down that occurs due to reductions in interstage network links.

Analysis is performed with both synthetic traces (that allow us to study the impact of different job characteristics) and real system workloads from different supercomputers. Our simulation results demonstrate that even in the worst case where all jobs perform 100% communication, and halving the numer of leaf switch uplinks, the overall system wide performance of jobs is not slowed down by more than 5% on an average.

II. Technical Approach

The goal of our work is to model the effects of network contention on the overall utility of real-world systems, in order to understand the quantitative benefit of nonblocking interconnection networks. Clearly, contention is a large issue in some environments, particularly those with workloads composed primarily of large jobs. Our basic approach is to determine per-crossbar utilization levels using a worst-case communication demand from jobs. Our workloads include the results of actual scheduling policy and hence are representative of real-world behavior on production systems. For this work, we assume that independent jobs do not communicate with one another.

Several hardware aspects of cluster systems are currently in flux. Multicore CPUs have become the norm for cluster systems. Likewise, the size of network crossbars, the building blocks of switches, are growing, which leads to flatter networks. We will analyze each of these factors in turn to determine how they alter the attractiveness of nonblocking networks.

A. Network Contention

Network performance is a key indicator of cluster utility. In parallel clusters, many applications are limited by their ability to communicate effectively with their peer processes. Frequently, network contention is believed to be a performance limiting factor.

Network performance is determined by two main factors in parallel systems. The first is a node's ability to communicate with its peers. This ability depends on the node and network hardware and on the node's software stack. The second is the ability of a network to quickly convey a packet from the sender to its intended recipeient. If the network is idle, this process will occur quickly without any errors. If transmission path includes any network links already carrying other traffic, then the packet will be slowed or even lost. When this contention occurs, performance is reduced, or retransmission may be required.

Cluster interconnection networks are frequently nonblocking. These networks are often described as having full bisection bandwidth. nonblocking networks are structured so that there exist enough internal links in the network fabric for the two parts of an arbitrary bisection of the network to communicate without network contention. This effect is achieved by ensuring that each nonblocking leaf crossbar can communicate with all other leaf crossbars at full bandwidth. As the number of leaf switches grows, additional layers of switches that provide high-bandwidth connectivity to other leaf switches must be added at certain points. When a new stage must be added into a network, the per-port cost of the network increases dramatically. For example, a fully populated three-stage network constructed out of 16 port crossbars will provide ports for 128 nodes. When the 129th node is added, the three stage network will need to be expanded to a five-stage network. These price discontinuities make nonblocking networks unappealing for many system sizes, depending on size of network crossbars used. Over the past several years, crossbar sizes have grown dramatically. Whereas 8-port crossbars used to be the norm, 16- and 24-port crossbars are now common and 32-port crossbars are on the horizon; and this trend is expected to continue well past this point. Increased crossbar sizes enable the construction of larger nonblocking networks with a fixed number of stages.

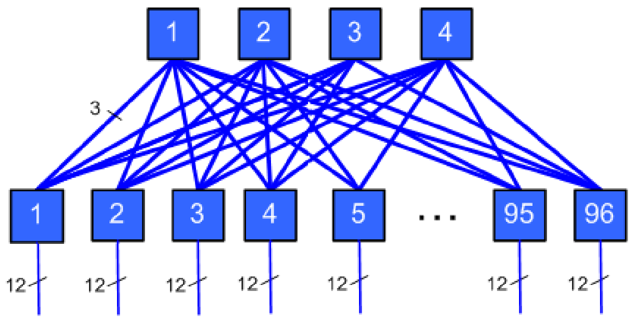

Fig. 1. Block diagram of the Atlas Infiniband Network

Figure 1, taken from the LLNL computing site [1], shows the Infiniband network used on the Atlas system at LLNL, which is one of the traces analyzed in this paper. This network shows a mixed view of crossbars and integrated switches. The boxes at the bottom of the diagram are leaf crossbars, with 12 ports connecting to nodes. The top layer of boxes are the network spine. This view is simplified, showing just the spine as four integrated 288 port switches. Internally, these switches are three-stage networks composed of 24-port crossbars.

The alternative to a nonblocking network is one in which bandwidth is internally constrained. Here, the optimal scenario possible in the nonblocking case cannot occur during heavy network utilization; interswitch links will frequently block, limiting end-to-end bandwidth for clients or causing packet loss. Blocking networks can be constructed in a variety of ways.

The contended networks analyzed in this paper are constructed by connecting nonblocking crossbars with reduced uplinks to a nonblocking network spine. Each leaf crossbar is connected to nodes on a half of its ports, while using some fraction of the remaining ports as uplinks. The advantage of this approach is that all contention occurs locally; jobs that share crossbars contend with each other for limited uplink capacity from that crossbar to the network spine. This architecture is shown in Figure 2.

In this 2:1 overcommitted network, S1 and S2 are 288-

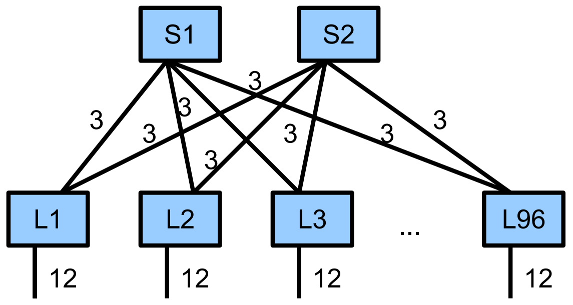

Fig. 2. Block diagram of an 2:1 overcommitted network

port Clos networks, built using 24-port crossbars. $L1 \dots L96$ are 24-port leaf crossbars. On each leaf switch, 12 ports are connected to nodes, while 3 ports are connected to each of S1 and S2. Compared with the Atlas network in Figure 1, the component savings in terms of spine switches and associated links is obvious.

B. Analysis Methodology

The goal of our analysis was to quantify the effects of network contention on realistic system workloads. We procured accounting logs for several systems: the Argonne Jazz system and two systems at Lawrence Livermore National Lab.

Jazz is a 350-node cluster acquired in late 2002. It is used by the Argonne community at large, and so the workload is unpredictable. The user base changes frequently, and the applications used vary widely in terms of scalability and network requirements. We have analyzed data from the entire operational life of Jazz, from 2003 until the present.

The other real-world data sources are from two LLNL clusters, Atlas and Thunder. Atlas is an 1152-node cluster, with 8 cores per node. Atlas is used primarily as a capability cluster. Thunder is a 1024-node cluster with four cores per node. This system is used primarily as a capacity cluster.

The analysis input consisted of a series of jobs; the pertinent information was start time, end time, and node allocations. We assummed a linear allocation of interconnect switch ports: in other words, if a crossbar is connected to 8 nodes, nodes 0–7 will be connected to the first leaf crossbar, and nodes 8–15 will be connected to the second, and so forth.

From this node-to-crossbar mapping, we were able to determine the number of jobs that are subject to uplink contention. Jobs that were run on nodes connected to a single crossbar will communicate internally only on a single, nonblocking crossbar; hence, these jobs cannot experience any uplink contention. All subsequent analysis was performed on jobs subject to uplink contention.

The next step was to calculate each job's worst-case uplink usage for communication traffic. This number differs for each crossbar used in a job, based on the way the job is spread across multiple crossbars. Uplink use for crossbar n for job j is defined as the following, where $c(n, j)$ is the number of nodes on crossbar n where job j is running.

$$U(n, j) = min(c(n, j), \sum_{\substack{m=0 \\ m \neq n}}^{xbars} c(m, j)) \qquad (1)$$

This equation determines a job's maximum ability to use crossbar uplinks, which is limited by both the number of job nodes on the local crossbar and the sum of job nodes on other crossbars. I/O activity, typically performed on the same interconnect network, is not factored in; it is outside the scope of this paper.

Using this information about crossbar uplink use, in conjunction with job start and end times, we can calculate the aggregate uplink use for each crossbar over time. In order to combine uplink use for two jobs, used crossbar uplink counts are added for overlapping time intervals, and left the same for nonoverlapping times. We can then generate the aggregate uplink utilization over time for all crossbars in the system, called U_{total}.

The final stage of analysis determines the impact of removing crossbar uplinks at leaf crossbars. An ideal nonblocking interconnect complex will not suffer from any oversubscription effects. As leaf node uplinks are removed, network performance begins to suffer if the aggregate uplink utilization is greater than the number of available uplinks. For a given reduction in uplink count, we can determine the number of affected jobs and the intervals in which contention occurs. A worst-case network slowdown can be determined by the ratio of original network bandwidth to available network bandwidth.

Given an uplink reduction level for leaf crossbars, the slowdown for a single job can be calculated for each interval **i** during its execution as follows. **R** is the number of remaining links, while $U_{total}(n, i)$ is the uplink use of crossbar **n** during interval **i**.

$$S(job, i) = \frac{max(U_{total}(0, i), \dots, U_{total}(xbars, i)))}{R} \qquad (2)$$

Jobs are assumed to be tightly coupled; hence each will be limited by the crossbar with highest demand in a given interval. The slowdown is the ratio of demand to available uplinks. The total slowdown, in wall time, for a single job consists of the uncontended time, combined with per-interval slowdown times that interval length.

This slowdown is indeed a worst-case scenario; it would be substantially reduced in the case of jobs that spend less than 100% of their time communicating. A more accurate approximation of job slowdown on a single crossbar can be determined probabilistically. Here, m represents usable crossbar uplinks, n is the remaining number of links (we assume that there are no idle links) and J is the fraction of time spent by processes communicating (communication probability). If $m <= n$ no contention will occur; contention will only occur if at least *(m + 1)* of the n data links are transmitting data at the same time.

The probability that exactly k of the links are transmitting data at the same time is given by:

$$\binom{n}{k} \times J^k \times (1 - J)^{n-k} \tag{3}$$

We are interested in the case where either exactly *(m + 1)* links are active, or exactly *(m + 2)* links are active, so on up to exactly n links being active. Thus, our overall probability of contention, *E(J)*, as a function of communication probability J can be given by:

$$E(J) = \sum_{k=m+1}^{n} [\binom{n}{k} \times J^k \times (1 - J)^{n-k}] \tag{4}$$

This equation represents a part of the newton binomial equation, whose sum would be *one*, if k ranged from *0* to n. For the partial sum given in equation 4, we expect this value to be much smaller than one, when the range is small, i.e., when *m+1* is close to n; in other words, we expect this probability to be close to zero, when the number of overprovisioned links is not very high.

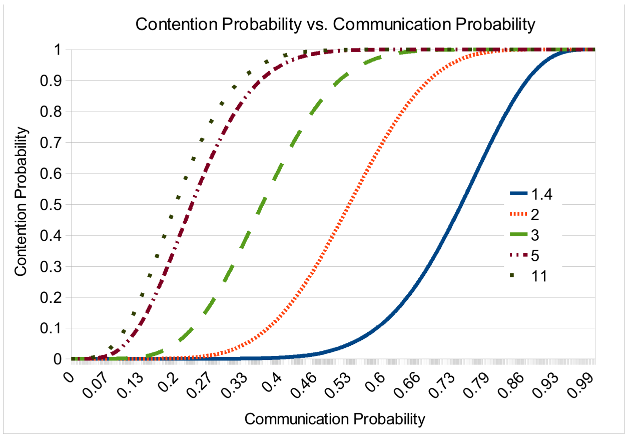

Fig. 3. Contention Probability as a function of Communication Probability

To further illustrate this, we plot the function *E(J)* against J in Figure 3, for a 24-port crossbar with different fractions of overprovisioning. The figure illustrates a steep increase in contention probability for high values of communication probability. Specifically, we notice that even for a communication probability of 20% and an overprovisioning factor of 3, there is a less than 5% probability of contention on a 24-port ASIC. Further, a 20% communication percentage is considered very high for most scientific applications. Note that the time spent within the MPI library, for example, is a combination of communication time and idle wait time waiting for communication.

In summary, while this paper deals with the worst case scenario of 100% communication probability for all applications, in reality, the potential amount of contention can be expected to be much lower. Further, this also gives us an indication of what kind of overprovisioning system integrators can utilize, assuming that they are aware of the approximate amount of communication their applications are expected to perform.

C. Multicore Effects on Network Contention

The introduction of multicore nodes into computational clusters clearly reduces overall network performance available to processes running on nodes. This reduction is due to the multiplication of computation resources without (in most cases) an addition of network capacity. Functionally, multicore or SMP fat nodes divide up available network bandwidth by the number of processes on the local node.

That said, the outgoing and incoming bandwidth available to a given node stays the same: it is just split more ways. Hence, the same basic model for network contention works for fat nodes.

The use of multicore systems does have one important effect on network contention. Jobs of a fixed size can run on a smaller number of nodes on these systems than on comparable single core nodes. This compression effectively produces a workload with a smaller per-job node counts. Smaller node counts translate to reduced potential for network contention; hence, we expect these multicore systems to be more tolerant of network contention.

To accurately simulate multicore effects, we have scaled the resources used by each job in our workloads according to an increased per-node core count. In most cases, this alteration will result in a job executing on a smaller number of fatter nodes. For example, a 32-node job from the Jazz workload (a system with uniprocessor nodes) is scaled to run on a smaller number of 8-core nodes. We preserve the fragmentation present in the original workload; if a job was run on a series of noncontiguous nodes, it will run on a disjoint set of cores spread across a larger number of multicore nodes. The best possible scenario is the case where a job ran on contiguous resources. In this case, the core locations will be compressed onto nodes that are effectively job-dedicated. In this case, the node count will be reduced by a factor of the ratio of new core count to old core count. In the worst-case scenario, fragmentation will spread the job allocation across the same number of nodes used in the original case.

Each workload comes from a system with a different number of cores per node. Jazz has one core per node, Atlas has eight, and Thunder has four. Each of these will be scaled up to larger core counts to demonstrate the impact of core count on network contention. The allocation fragmentation present in each workload will vary the improvements offered by multicore growth in each case.

III. Results

We describe three sets of results. First, we will show the initial analysis of potential job contention in the presence of blocking network behavior. Next, we show the effect of varying network crossbar size for a given workload. Finally, we show how multicore nodes affect network contention.

A. *Network Contention Analysis*

Our first round of analysis considered the historical workloads on networks built with 16 and 24 port crossbars. All three production workloads and the synthetic workload all showed similar trends. Figures 4, 5, and 6 show examples of these results. These figures include the worst-case scenarios for communication slowdown; the probabalistic slowdown will also be included in the final paper.

Each graph includes two types of data for each column. The first measures slowdown as an effect on the system overall. This is calculated by determining the aggregate wallclock time increase and comparing it to the overall wallclock time used by jobs in the trace. The second column measures slowdown as it impacts jobs that suffered contention. This metric shows the impact on users for individual jobs, when slowdown occurred.

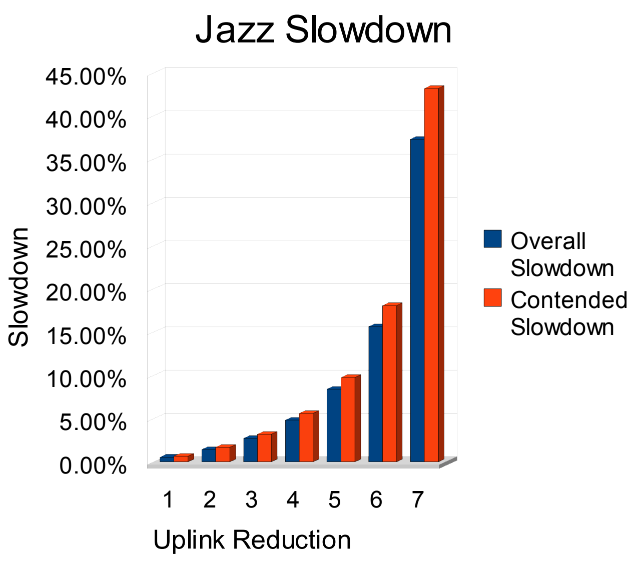

Fig. 4. Effect of uplinks removal on the Jazz workload

Each graph shows a similar trend. As expected, as uplinks were removed, more jobs suffered from contention, resulting in larger slowdowns. More surprising, the impact of removing a small number of uplinks is nearly negligible. Moving from a nonblocking interconnect network to one with a 1:2 oversubscription ratio results in only a 5% system-wide slowdown. Also, Jazz, a 350-node cluster, showed the highest sensitivity to network contention. Atlas and Thunder, both considerably larger systems, showed much less sensitivity. These results show another interesting pattern: the difference between system slowdowns and single job slowdowns varies greatly by workload. Both of these results were quite unexpected.

The slowdowns displayed in the graphs are the mean of calculated slowdowns. While the average case looks good even with many uplinks removed, the slowdowns experienced by jobs can be quite bad in the worse cases. Figure 7 shows the downside of this approach.

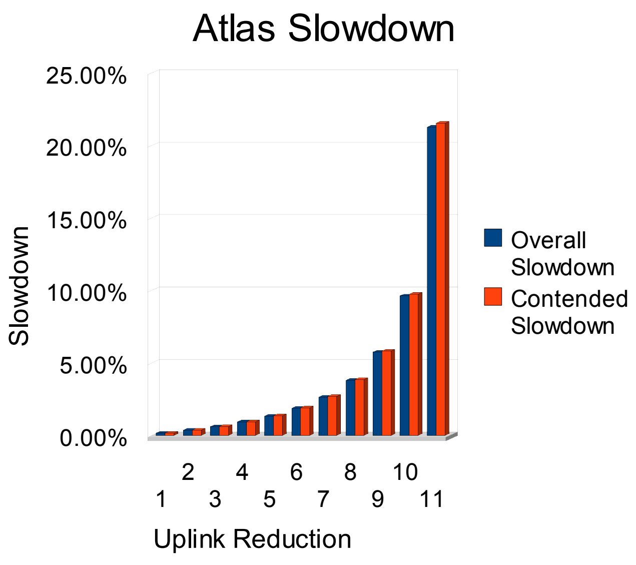

Fig. 5. Effect of uplink removal on the Atlas workload

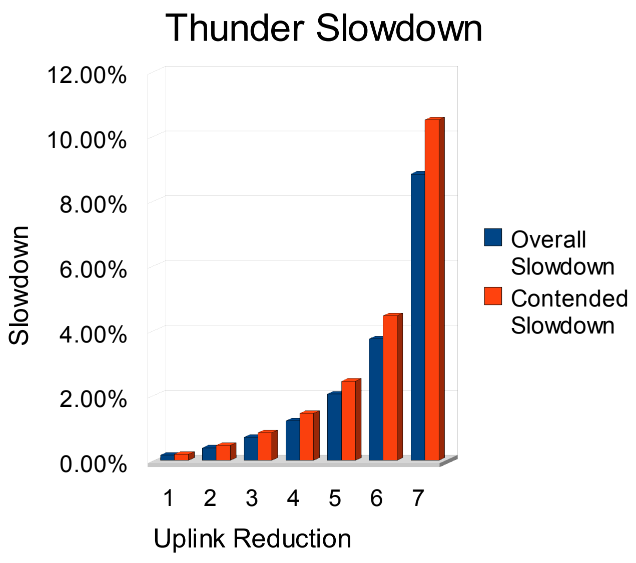

Fig. 6. Effect of uplink removal on the Thunder workload

In the Jazz and Thunder workload, slowdowns of 350% were encountered, while slowdowns of 550% occurred in the Atlas workload. These maximum slowdowns are related to the crossbar sizes. The Jazz and Thunder workloads were analyzed with 16-port crossbars, while the Atlas workload runs with a 24-port crossbar. While substantial slowdowns did occur in each workload, they occurred extremely infrequently. This result is shown by the average slowdown being less than 50%. While the overall slowdowns look good in most cases, particular jobs were badly impacted.

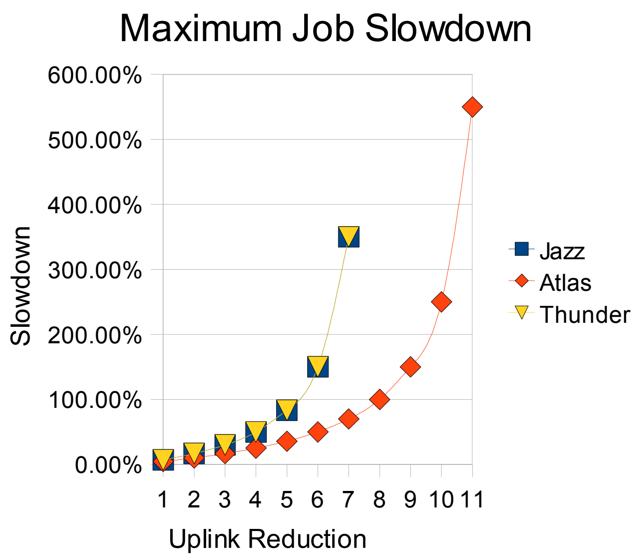

Fig. 7. Maximum Slowdown by Workload.

B. Crossbar Size Comparison

The next analysis varied crossbar sizes for a given workload and preexisting schedule. Again, all datasets showed a similar pattern, an example of which is shown in Figure 8. Two important patterns are shown in this data. First, slowdowns grow quite slowly until a majority of the uplinks are removed. Larger crossbar sizes incur decreased slowdown for the same ratio of removed uplinks. Also, for each crossbar size, the last case incurs additional slowdowns compared with the slowdowns shown by smaller crossbars.

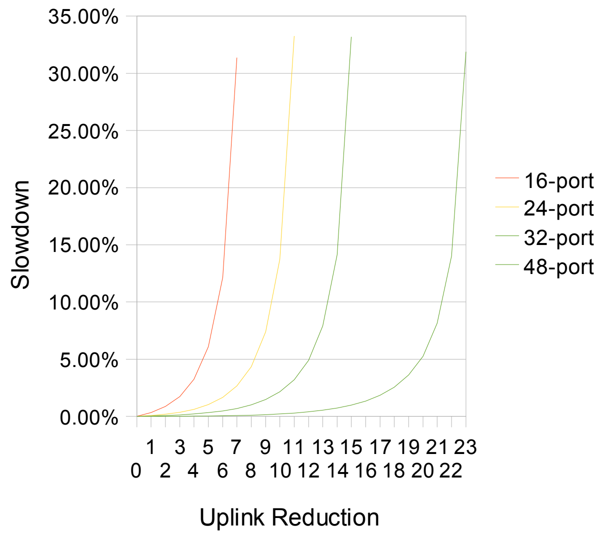

Fig. 8. Effect of crossbar growth on a synthetic workload

This data shows that networks built out of larger crossbars will be less sensitive to network contention for a given workload.

C. Multicore Comparison

The final analysis simulates the effect of increases in multicore systems on network contention. The scaling used to simulate fatter nodes is described in detail in Section II-C.

Two interesting trends emerge from the data. We simulated each workload on systems with increasing numbers of cores per node. In each case, the network contention decreased substantially with each expansion of node size, even in the presence of substantial allocation fragmentation. Each of the following graphs chart the slowdown for the increasing core counts, starting at their actual core counts and proceeding upward.

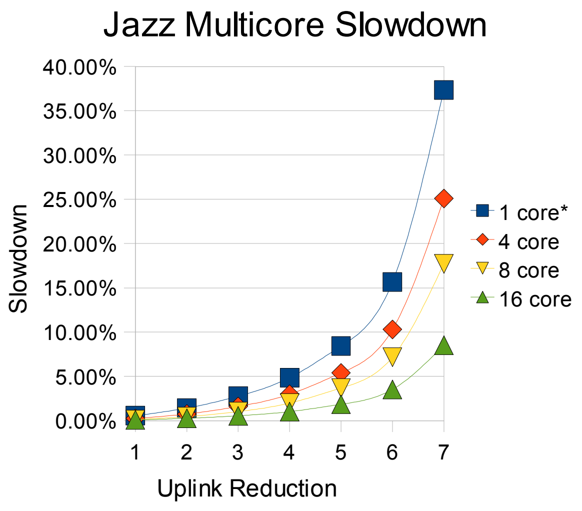

Fig. 9. Effect of multicore processors on the Jazz workload

As core counts increase, the amount of network contention for each workload decreases sharply. This situation bodes well for the use of overcommitted networks as core sizes grow. Even for the worst contention behavior we have seen, in the Jazz workload, growth to modern core counts shows a dramatic improvement. In the worse contention case, where only one uplink remains from each leaf switch, changing from uniprocessor nodes to eight-core nodes reduces the slowdown from 37.35% to 17.72%. The other traces show similar, though less striking, improvement.

Comparison of the three traces is also useful, as each system has a different number of cores per node. The Jazz workload has shown much more sensitivity to network contention than either of the other two workloads, despite the fact that Atlas and Thunder are both large systems. Moreover, Atlas is a capability system, and its average job size is considerably larger than that of either Thunder or Jazz. This analysis, showing the impact of increased core sizes, reveals the likely cause.

Comparison of the Atlas and Thunder workloads is also interesting. These are comparably sized systems, in terms of

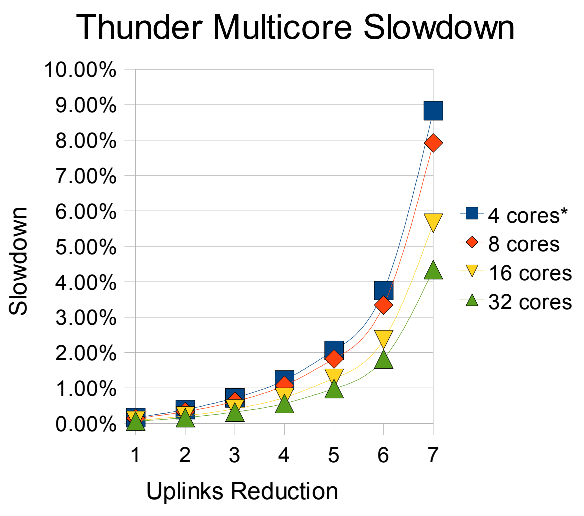

Fig. 10. Effect of multicore processors on the Thunder workload

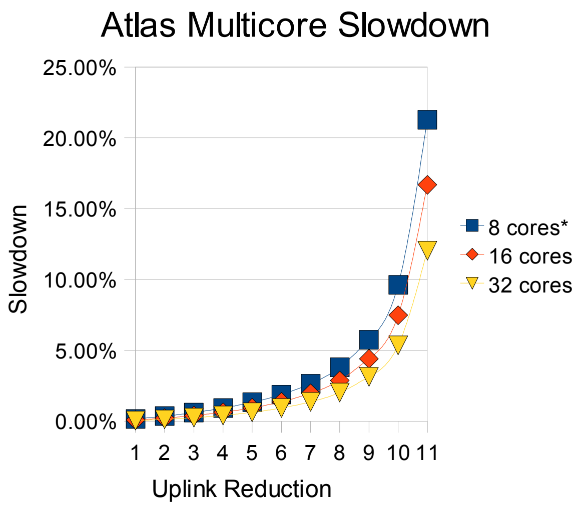

Fig. 11. Effect of multicore processors on the Atlas workload

node counts. The two main differences are the number of cores per node and the workload. The higher number of cores per node on the Atlas system partially balances out the difference in workload between the systems. Despite the fact that the average job size on Atlas is nearly four times larger than that of Thunder, the differences in slowdown are much smaller. This effect provides an interesting window into the future, where the number of cores in a single node and the size of network crossbars will continue to grow.

IV. Conclusion

In this paper, we have presented a model that describes network contention on space-shared clusters. We have used this model to calculate the aggregate impact of network contention on workloads for several systems. These workloads each expressed aspects of the variety present on many clusters. One workload was taken from a moderately sized production cluster at Argonne National Laboratory. The others were taken from capacity and capability clusters at Lawrence Livermore National Laboratory. In each case, analysis showed that the workload could tolerate the slowdowns caused by moderate amounts of leaf switch uplink reduction. This finding suggests that nonblocking networks may not be an absolute requirement in modern HEC systems.

Our initial analysis showed that scaling back the uplink bandwidth of the leaf crossbars in a cluster interconnect should have a minimal impact in aggregate, provided the majority of uplinks are not removed. While individual jobs can be greatly impacted, these jobs are not so common as to substantially slow systems overall. Also, as the size of network crossbars increases, the impact of contention is dramatically reduced for a fixed workload. Moreover, multicore systems have a substantial positive impact on the sensitivity to network contention for a given workload.

Each of these findings is independently encouraging, particularly with an eye toward the near future. System sizes have grown considerably over the past few years, and this trend is sure to continue. As system sizes grow, the cost of nonblocking interconnect networks increases dramatically. Because we expect core count and crossbar size growth to continue, system tolerances for network contention and oversubscription will expand. These factors will make oversubscribed networks compelling for the large systems of the next five years.

This strategy of reducing uplinks to the network spine is useful in several situations. Nonblocking switch complexes grow non-linearly; at certain points, the addition of small numbers of nodes requires the addition of a large amount of switching infrastructure. The strategy of removing leaf crossbar uplinks while retaining a nonblocking network spine would provide a limited slowdown for user jobs while substantially reducing the network costs.

While our analysis suggests that many systems can tolerate increased network contention, this is not a universal truth. Particular systems, including those featuring large numbers of jobs that frequently occupy a majority of that system, will be susceptible to network contention solely due to this workload.

Uplink removal is not without detrimental aspects. Frequently, network switch complexes are subject to hotspot behavior. This situation has been widely studied, and various improvements have been proposed, including multipathing [2]. Hotspots can occur for a variety of reasons, including hardware failures or load imbalances in multipathing. The removal of spine capacity will negatively impact networks in either of these cases.

Caveats aside, these results demonstrate that oversubscribed networks could fill a vital role in large scale clusters – that of price-competitive, though still high-performance, networks.

V. Future Work

While our model provides a basic mechanism to assess workload-related uplink contention, several steps are required before this process can be used as a design guide for new systems.

I/O is not factored into this model. Many applications spend large fractions of their wall time performing I/O. This activity typically uses the same interconnect network used for job communication and hence has the potential to interfere with other jobs in blocking networks. Our model will need to take this into account if it is to be useful for prediction purposes.

Another issue we have not examined here is the impact of reusing idle uplink ports for nodes. This approach can clearly result in even higher levels of contention than the worst numbers in our graphs, if leaf crossbars use only a single uplink. It is unclear whether more conservative use of this approach results in good price/performance compared with the approach described in this paper.

Also, this model does not factor in hotspots in the network spine. Hotspots have been shown to occur relatively frequently and hence have a bearing on this model.

Several studies have focused on effective node allocation and its impact of on application performance. Many of these efforts have considered mesh [3], [4], [5], [6], [7], [8], [9] and hypercube topologies [10], [11]. The Buddy strategy for node allocation has been widely studied [12], [13], [14], [15].

These studies have shown various ways of improving application performance via topology-aware mapping and scheduling. The thrust of our effort is in an orthogonal direction – development of upper-bounds on overall performance degradation of a workload scheduled on a shared cluster. Our analysis makes worst-case assumptions regarding demand on network communication links. In practice, with the use of effective mapping and scheduling algorithms developed in other studies, the actual degradation from removal of communication links from nonblocking networks would likely be considerably lower than the upper bounds that we developed. This would further strengthen our conclusions: nonblocking interconnection networks are even more an overkill than shown by our upper-bound analysis.

Finally, this work does not include an explicit pricing model for determining precise network costs. If this approach is to have impact in cluster-buying decisions, network costs must be factored in.

Software Availability

The software used to perform the analysis is open source and will be publicly available by publication time. It is able to natively process PBS and Slurm accounting logs and an extended version of SWF files that include job execution location data.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

Some of the analysis in this paper, particularly the multicore analysis portion, would not have been possible without the Parallel Workloads Archive [16] compiled and maintained by Dror Feitelson. In particular, the traces provided by Moe Jette of Lawrence Livermore National Laboratory for the Atlas and Thunder systems were crucial for this work.

References

[1] B. Barney. Linux clusters overview. Lawrence Livermore National Laboratory. [Online]. Available: https://computing.llnl.gov/tutorials/linux_clusters/#Hardware

[2] A. Vishnu, M. Koop, A. Moody, A. R. Mamidala, S. Narravula, and D. K. Panda, "Hot-spot avoidance with multi-pathing over infiniband: An mpi perspective," in *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 479–486.

[3] P. Chuang and N. Tseng, "An efficient submesh allocation strategy for mesh computer systems," in *Proceedings of the 11th International Conference on Distributed Computing Systems*, 1991, pp. 256–263.

[4] K. Li and K. H. Cheng, "A two-dimensional buddy system for dynamic resource allocation in a partitionable mesh connected system," *Journal of Parallel and Distributed Computing*, vol. 12, pp. 79–83, 1991.

[5] Y. Zhu, "Efficient processor allocation strategies for mesh-connected parallel computers," *Journal of Parallel and Distributed Computing*, vol. 16, pp. 328–337, 1992.

[6] V. Leung, E. Arkin, M. Bender, D. Bunde, J. Johnston, A. Lal, J. Mitchell, C. Phillips, and S. Seiden, "Processor Allocation on Cplant: Achieving General Processor Locality Using One-Dimensional Allocation Strategies," *Proceedings of the IEEE International Conference on Cluster Computing*, 2002.

[7] J. Mache, V. Lo, and S. Garg, "Job scheduling that minimizes network contention due to both communication and I/O," *Proceedings of the 14th International Parallel Processing Symposium (IPPS*, vol. 0, p. 457, 2000.

[8] S. Moore and L. Ni, "The effects of network contention on processor allocation strategies," *Proceedings of the 10th International Parallel Processing Symposium (IPPS*, pp. 268–273, 1996.

[9] M. A. Bender, D. P. Bunde, E. D. Demaine, S. P. Fekete, V. J. Leung, H. Meijer, and C. A. Phillips, "Communication-aware processor allocation for supercomputers: Finding point sets of small average distance," *Algorithmica*, vol. 50, no. 2, pp. 279–298, 2008.

[10] P. Krueger, T. Lai, and V. Dixit-Radiya, "Job scheduling is more important than processor allocation for hypercube computers," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 5, 1994, pp. 488–497.

[11] P. Mohapatra, C. Yu, C. R. Das, and J. Kim, "A lazy scheduling scheme for improving hypercube performance," in *Proceedings of the 1993 International Conference on Parallel Processing*, vol. I - Architecture. CRC Press, 1993, pp. I–110–I–117.

[12] M. Chen and K. G. Shin, "Processor allocation in an n-cube multiprocessor using gray codes," *IEEE Trans. on Computers*, vol. C, no. 36, pp. 1396–1407, 1987.

[13] S. Dutt and J. P. Hayes, "Subcube allocation in hypercube computers," *IEEE Trans. on Computers*, vol. 40, no. 3, pp. 341–352, 1991.

[14] J. Kim, C. R. Das, and W. Lin, "A top-down processor allocation scheme for hypercube computers," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, 1991, pp. 20–30.

[15] V. Subramani, R. Kettimuthu, S. Srinivasan, J. Johnston, and P. Sadayappan, "Selective buddy allocation for scheduling parallel jobs on clusters," *Cluster Computing, 2002. Proceedings. 2002 IEEE International Conference on*, pp. 107–116, 2002.

[16] D. Feitelson. Parallel workloads archive. Hebrew University. [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/