Interconnection Architectures for Petabyte-Scale High-Performance Storage Systems

Andy D. Hospodor
Senior Member, IEEE
andy.hospodor@ieee.org

Ethan L. Miller
Storage Systems Research Center
University of California, Santa Cruz
elm@cs.ucsc.edu

Abstract

As demand for storage bandwidth and capacity grows, designers have proposed the construction of petabyte-scale storage systems. Rather than relying upon a few very large storage arrays, these petabyte-scale systems have thousands of individual disks working together to provide aggregate storage system bandwidth exceeding 100 GB/s. However, providing this bandwidth to storage system clients becomes difficult due to limits in network technology. This paper discusses different interconnection topologies for large disk-based systems, drawing on previous experience from the parallel computing community. By choosing the right network, storage system designers can eliminate the need for expensive high-bandwidth communication links and provide a highly-redundant network resilient against single node failures. We analyze several different topology choices and explore the tradeoffs between cost and performance. Using simulations, we uncover potential pitfalls, such as the placement of connections between the storage system network and its clients, that may arise when designing such a large system.

1. Introduction

Modern high-performance computing systems require storage systems capable of storing petabytes[1] of data, and delivering that data to thousands of computing elements at aggregate speeds exceeding 100 GB/s. Just as high-performance computers have shifted from a few very powerful computation elements to networks of thousands of commodity-type computing elements, storage systems must make the transition from relatively few high-performance storage engines to thousands of networked commodity-type storage devices.

[1]A petabyte (PB) is 2^{50} bytes.

The first part of this shift is already occurring at the storage device level. Today, many storage subsystems utilize low-cost commodity storage in the form of 3.5” hard disk drives. The heads, media and electronics of these devices are often identical to storage used on desktop computers. The only remaining differentiator between desktop and server storage is the interface. At present, Fibre-Channel and SCSI remain the choice of large, high-end storage systems while the AT attachment (ATA) remains the choice of desktop storage. However, the introduction of the Serial ATA interface provides nearly equivalent performance and a greatly reduced cost to attach storage.

Most current designs for such petabyte-scale systems rely upon relatively large individual storage systems that must be connected by very high-speed networks in order to provide the required transfer bandwidths to each storage element. We have developed alternatives to this design technique using 1 Gb/s network speeds and small (4–12 port) switching elements to connect individual object-based storage devices, usually single disks. By including a small-scale switch on each drive, we develop a design that is more scalable and less expensive than using larger storage elements because we can use cheaper networks and switches. Moreover, our design is more resistant to failures—if a single switch or node fails, data can simply flow around it. Since failure of a single switch typically makes data from at least one storage element unavailable, maintaining less data per storage element makes the overall storage system more resilient.

In this paper, we present alternative interconnection network designs for a petabyte-scale storage system built from individual nodes consisting of a disk drive and 4–12 port gigabit network switch. We explore alternative network topologies, focusing on overall performance and resistance to individual switch failures. We are less concerned with disk failures—disks will fail regardless of interconnection network topology, and there is other research on redundancy schemes for massive-scale storage systems [16].

2. Background

There are many existing techniques that provide high-bandwidth file service, including RAID, storage area networks, and network-attached storage. However, these techniques cannot provide 100 GB/s on their own, and each has limitations that manifest in a petabyte-scale storage system. Rather, network topologies originally developed for massively parallel computers are better suited to construct massive storage systems.

2.1. Existing Storage Architectures

RAID (Redundant Array of Independent Disks) [2] protects a disk array against failure of an individual drive. However, the RAID system is limited to the aggregate performance of the underlying array. RAID arrays are typically limited to about 16 disks; larger arrays begin to suffer from reliability problems and issues of internal bandwidth. Systems such as Swift [9] have proposed the use of RAID on very large disk arrays by computing parity across subsections of disks, thus allowing the construction of larger arrays. However, such systems still suffer from a basic problem: connections between the disks in the array and to the outside world are limited by the speed of the interconnection network.

Storage area networks (SANs) aggregate many devices together at the block level. Storage systems are connected together via network, typically FibreChannel, through high-performance switches. This arrangement allows servers to share a pool of storage, and can enable the use of hundreds of disks in a single system. However, the primary use of SANs is to decouple servers from storage devices, not to provide high bandwidth. While SANs are appropriate for high I/O rate systems, they cannot provide high bandwidth without appropriate interconnection network topology.

Network-attached storage [6] (NAS) is similar to SAN-based storage in that both designs have pools of storage connected to servers via networks. In NAS, however, individual devices present storage at the file level rather than the block level. This means that individual devices are responsible for managing their own data layout; in SANs, data layout is managed by the servers. While most existing network-attached storage is implemented in the form of CIFS- or NFS-style file systems, object-based storage [15] is fast becoming a good choice for large-scale storage systems [16]. Current object-based file systems such as Lustre [13] use relatively large storage nodes, each implemented as a standard network file server with dozens of disks. As a result, they must use relatively high-speed interconnections to provide the necessary aggregate bandwidth. In contrast, tightly coupling switches and individual disks can provide the same high bandwidth with much less expensive, lower-speed networks and switches.

2.2. Parallel Processing

Interconnection networks for computing elements have long been the subject of parallel computing research. No clear winner has emerged; rather, there are many different interconnection topologies, each with its own advantages and disadvantages, as discussed in Section 3. Traditional multiprocessors such as the Intel Touchstone Delta [14] and the CM-5 [10] segregated I/O nodes from computing nodes, typically placing I/O nodes at the edge of the parallel computer. File systems developed for these configurations were capable of high performance [3, 12] using a relatively small number of RAID-based arrays, eliminating the need for more complex interconnection networks in the storage system.

There have been a few systems that suggested embedding storage in a multiprocessor network. In RAMA [11], every multiprocessor compute node had its own disk and switch. RAMA, however, did not consider storage systems on the scale that are necessary for today's systems, and did not consider the wide range of topologies discussed in this paper.

Fortunately, storage systems place different, and somewhat less stringent, demands on the interconnection network than parallel processors. Computing nodes typically communicate using small, low latency messages, but storage access involves large transfers and relatively high latency. As a result, parallel computers require custom network hardware, while storage interconnection networks, because of their tolerance for higher latency, can exploit commodity technologies such as gigabit Ethernet.

2.3. Issues with Large Storage Scaling

A petabyte-scale storage system must meet many demands: it must provide high bandwidth at reasonable latency, it must be both continuously available and reliable, it must not lose data, and its performance must scale as its capacity increases. Existing large-scale storage systems have some of these characteristics, but not all of them. For example, most existing storage systems are scaled by replacing the entire storage system in a "forklift upgrade." This approach is unacceptable in a system containing petabytes of data because the system is simply too large. While there are techniques for dynamically adding storage capacity to a existing system [7], the inability of such systems to scale in performance remains an issue.

One petabyte of storage capacity requires about 4096 (2^{12}) storage devices of 250 GB each; disks with this capacity are appearing in the consumer desktop market in

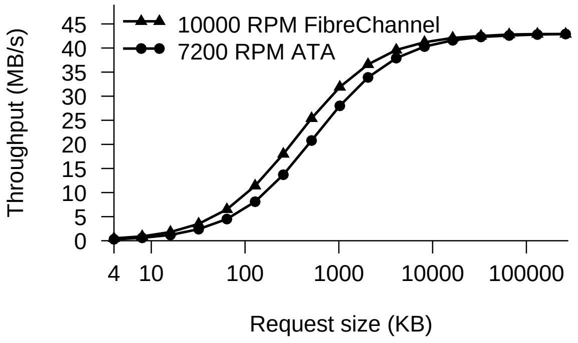


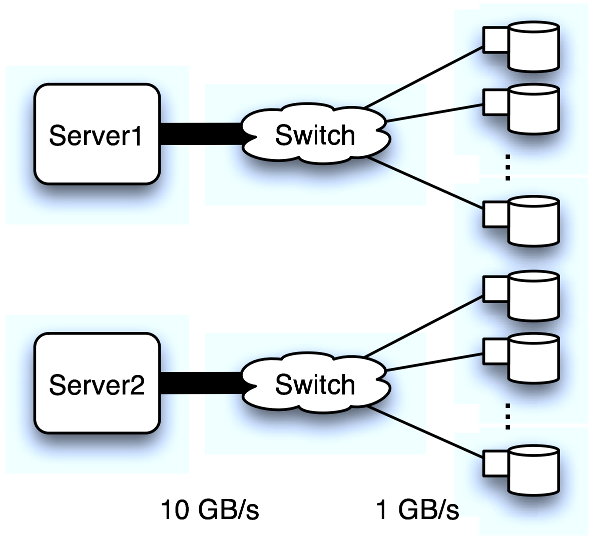
Figure 1. Disk throughput as a function of transfer size.

early 2004. A typical high-performance disk, the Seagate Cheetah 10K, has a 200 MB/s FibreChannel interface and spins at 10000 RPM with a 5.3 ms average seek time and sustained transfer rate of 44 MB/s. In a typical transaction processing environment, the Cheetah would service a 4 KB request in about 8.3 ms for a maximum of 120 I/Os per second, or 0.5 MB/s from each drive. The aggregate from all 4096 drives would be 4096×0.5 MB/s, or only 2 GB/s—far below the required bandwidth of 100 GB/s. By increasing the request size to 512 KB, disk throughput is increased to 25 MB/s per drive, for an aggregate bandwidth of 100 GB/s. Alternatively, Figure 1 shows that low-cost serial ATA drives, such as the 7200 RPM Seagate Barracuda 7200, could also meet the 100 GB/s requirement with a slightly larger request size of 1 MB.

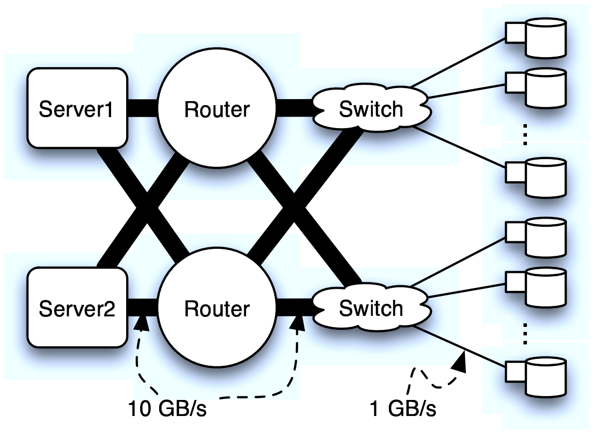
3. Interconnection Strategies

The interconnection network in a petabyte-scale storage system must be capable of handling an aggregate bandwidth of 100 GB/s as well as providing a connection to each *n*-storage node capable of transferring 25*n* MB/s of data. Thus, a 1 Gb/s network link can support nodes with at most 2–3 disks, and a 10 Gb/s network link can support up to 25 disks per node. As with any technology, however, faster is more expensive—1 Gb/s networks cost less than $20 per port, while 10 Gb/s can cost $5000 per port. In time, 10 Gb/s networks will drop in price, but it is likely that storage bandwidth demands will increase, making a tradeoff still necessary. This non-linear tradeoff in cost-performance compels us to consider complex architectures that can leverage 1 Gb/s interconnects.

The challenge facing storage system designers is an architecture that connects the storage to servers. Figure 2(a) shows a simple strategy that connects a server to 32 storage devices through a switch. Simple replication of this strategy 128 times yields a system capable of meeting the requirement. This strategy is remarkably similar to RAID



(a) Disks connected to a single server. This configuration is susceptible to loss of availability if a single switch or server fails.



(b) Disks connected to redundant servers. Switch failure is still an issue, but routers allow for more redundancy.

Figure 2. Simple interconnection strategies.

level 0, known as Just a Bunch of Disks (JBOD), and suffers from similar issues with reliability described later in the paper. Here, the port cost would be 4096 switch ports of 1 Gb/s and 128 ports of 10 Gb/s. However, the placement of data becomes crucial in order to keep all storage devices active. Since individual servers can only communicate with a small fraction of the total disk, clients must send their requests to the correct server. Unless there is a switching network comparable to that in the designs we discuss below interposed between the clients and the servers, this design is not viable. If there *is* a switching fabric between clients and servers, the designs below provide guidelines for how the network should be designed.

3.1. Fat Trees

Figure 2(b) shows a hierarchical strategy similar to a fat-tree that provides redundant connections between components. This strategy expands to have each server connect to two of eight routers that interconnect with the 128 switches that finally attach the 4096 storage devices. Each router has 32 ports attached to the servers and 128 ports attached to each of the switches and seven additional ports to the other routers. The port cost would be 4096 ports of 1 Gb/s, 2048 ports of 10 Gb/s that connect the 128 switches to the 8 routers (one port at either end), 112 ports of 10 Gb/s that interconnect the 8 routers, and 256 ports of 10 Gb/s that connect each servers to two routers. This configuration has the added drawback that the routers must be very large; it is typically not possible to build monolithic network devices with over 100 ports, so the routers would have to be constructed as multi-stage devices. While this device would allow any client to access any disk, the routers in this configuration would be very expensive. Furthermore, the 2418 ports of 10 Gb/s add nearly $10M to the overall cost, making this configuration a poor choice.

3.2. Butterfly Networks

Butterfly networks provide a structure similar to the hierarchical strategy at a more reasonable cost. Figure 3 shows a butterfly network interconnection strategy that connects disks to servers. The butterfly network can have relatively few links, but the links may need to be faster because each layer of the network must carry the entire traffic load on its links. In order to keep the individual links below 1 Gb/s, the butterfly network would need 1024 links per level for an aggregate throughput of 100 GB/s. Building a full butterfly network for 4096 disks using 1024 links and 128 switches per level would require three levels of 16-port switches and an additional level of 36-port "concentrators" to route data to and from 32 disks. Alternatively, the switching network could be built entirely from five levels of 8-port switches, using an additional level of 10-port switches to aggregate individual disks together. We use this second configuration in the remainder of the paper because, while 16 port switches are possible, we believe that the 8 port switches necessary for the second design are more reasonable.

While butterfly networks appear attractive in many ways, they do not have the fault-tolerance provided by cube-style networks such as meshes and torii. In fact, only a single path exists between any pair of server and storage devices connected by the butterfly network. In traditional parallel computers, network failures could be handled either by shutting down the affected nodes or by shutting down the entire system. Storage fabrics, on the other hand, must continue to run even in the face of network failures, making butterfly networks less attractive unless there is a method to route traffic around failed links. Cube-style networks have many routes between any two nodes in the fabric, making them more tolerant of link failures.

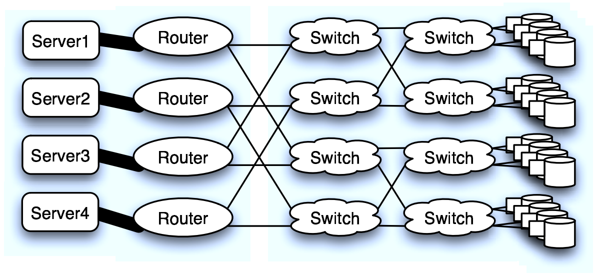


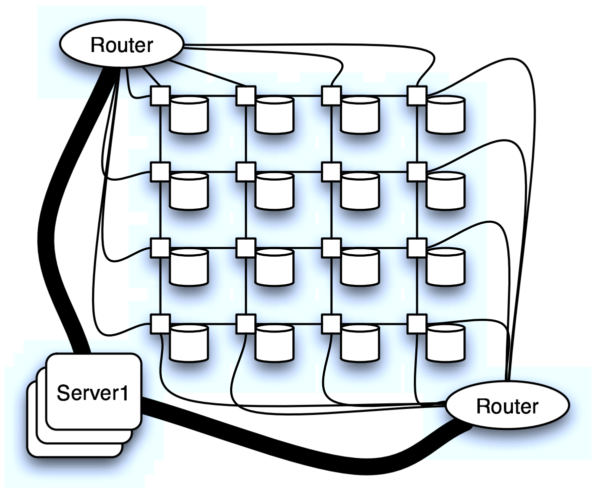
Figure 3. Disks connected in a butterfly network topology.

3.3. Meshes and Torii

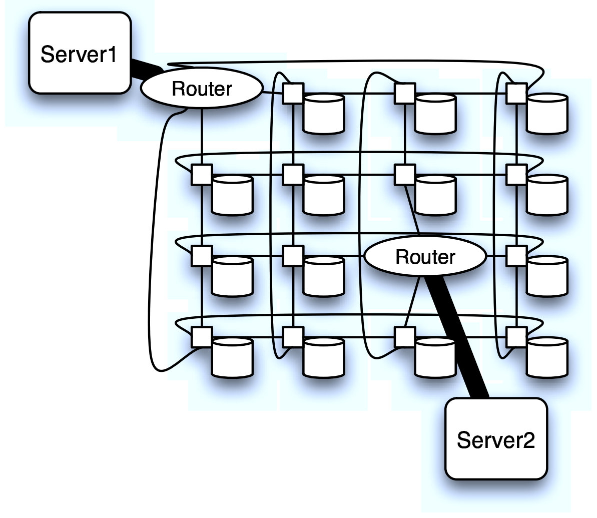
Figure 4(a) shows a mesh strategy that combines the storage device with a small switch that contains four 1 GB/s ports. The 4096 storage devices would be arranged as a 64×64 mesh with routers connecting the edge of the mesh to the servers. This configuration would require eight routers to connect to 128 servers to provide the necessary 100 GB/s bandwidth. This configuration would require 16384 1 Gb/s ports for the storage, 256 10 Gb/s ports that connect the 8 routers to the servers on two edges of the mesh, and the 256 10 Gb/s ports that connect the servers to the routers. Optionally, another 256 ports would connect all four sides of the mesh to the routers, although this much redundancy is not likely to be necessary. Router interconnects are no longer necessary because the mesh provides alternate paths in case of failure.

Torus topologies, shown in Figure 4(b), are similar to meshes, but with the addition of "wrap-around" connections between opposing edges. The inclusion of these additional connections does not greatly increase cost, but it cuts the average path length—the distance between servers and storage for a given request—by a factor of two, reducing required bandwidth and contention for network links. However, this design choice requires external connectivity into the storage fabric through routers placed at dedicated locations within the torus .

Mesh and torus topologies are likely a good fit for large scale storage systems built from "bricks," as proposed by IBM (IceCube [8]) and Hewlett Packard (Federated Array of Bricks [5]). Such topologies are naturally limited to three dimensions (six connections) per element, though they may resemble hypercubes if multiple highly connected disks are packed into a single "brick."



(a) Disks connected in a mesh topology.



(b) Disks connected in a torus topology.

Figure 4. Mesh and torus interconnection topologies.

3.4. Hypercubes

Figure 5 shows a hypercube strategy [1] of a two-dimensional hypercube of degree four that intersperses the routers and storage devices throughout the hypercube. In a 4096 node storage system, 3968 storage devices and 128 routers could be arranged in a hypercube of degree 12. Each node in this configuration has twelve 1 Gb/s ports and each router has two additional 10 Gb/s ports that connect to two servers. Bandwidth in the hypercube topology may scale better than in the mesh and torus topologies, but the cost is higher because the number of connections at each node increases as the system gets larger. Note also that hypercubes are a special case of torii; for example, a degree

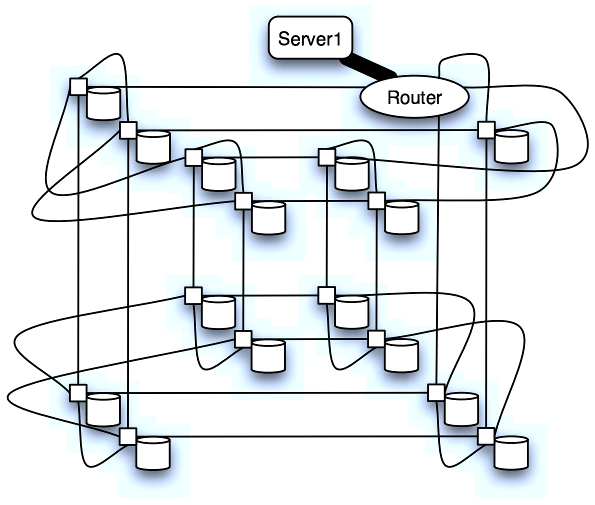


Figure 5. Disks connected in a hypercube topology.

12 hypercube can also be considered a 6-D symmetrical torus. Furthermore, the hypercube can be extended as a torus by adding nodes in one dimension; there is no need to add nodes in powers of two.

Hypercubes and high-dimensional torii need not be built from individual disks and routers. To make packaging less expensive, a small group of disks may be assembled into a unit, and the units connected together using a lower-dimensional torus. For units with eight disks in a degree 12 hypercube, this approach requires each unit to have 48 external connections—eight connections per cube face. This is not an unreasonable requirement if the system uses gigabit Ethernet or optical fiber network connections.

4. Analytic Results

All of the topologies listed in Section 3 appear capable of providing 100 GB/s bandwidth from a cluster of 4096 disks. Further inspection shows that, because of limitations in link capacities, this is not the case. Moreover, the topologies differ in several critical ways, including overall system cost, aggregate bandwidth, latency and resistance to component failures. We analyzed the basic characteristics of seven specific topologies, listed in Table 1.

4.1. System Cost

One benefit for the designs in which switches are embedded in the “storage fabric” is that they require far fewer high speed ports at the cost of additional low speed ports. The 2004 cost of a gigabit Ethernet port is less than $20, whereas the cost of a 10 gigabit Ethernet port is on the or-

Network	Dimensions	Ports
Fat Tree	32-8-32-1024	6,512
2D mesh	64×64	16,384
2D torus	64×64	16,384
3D torus	$16 \times 16 \times 16$	24,576
4D torus	$8 \times 8 \times 8 \times 8$	32,768
5D torus	$4 \times 8 \times 4 \times 8 \times 4$	40,960
6D hypercube	$4 \times 4 \times 4 \times 4 \times 4 \times 4$	49,152
Butterfly	256 4×4 switches/layer	14,336

Table 1. Switching fabric topologies to accommodate about 4000 disks.

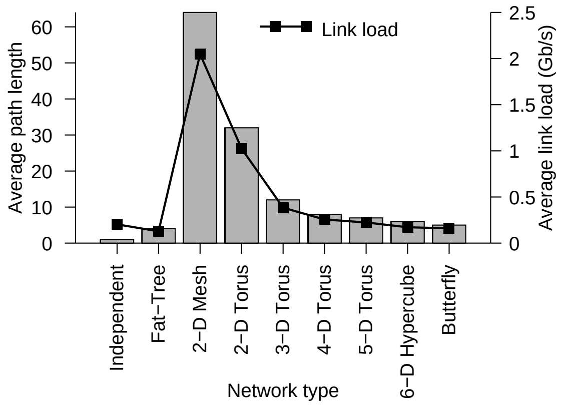


Figure 6. Average number of network hops and expected per-link bandwidth for each interconnection network topology. The "independent" topology is omitted because it relies upon the host computer for communication between storage nodes.

der of $5000. This non-linear tradeoff makes the fabric-type structures more appealing than the other structures because they simply cost less. The overall cost of a 4096 node system with different configurations is shown in Figure 7. The "independent" system is shown as a baseline; in such a system, each disk is connected to exactly one server, and servers are not connected to one another. While this is by far the least expensive option, it requires that the file system use the network of host servers to manage client access of data from the entire storage system, and limits the storage system's ability to perform internal communication for reliability and other functions. Among the other options, lower dimensionality "cubes" are the least expensive, with higher-dimension and cubes and torii being the most expensive and butterfly networks in between.

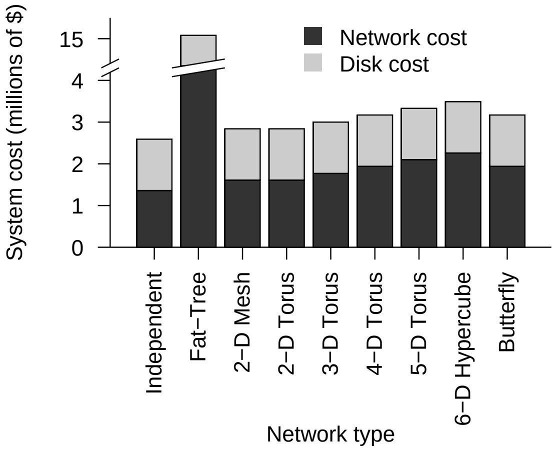


Figure 7. Total system cost for different interconnection network topologies. The "independent" topology relies upon the host computer for communication between storage nodes.

4.2. System Bandwidth

Another consideration for massive storage systems is the average number of network hops between a disk and a server, as shown in Figure 6. The number of hops is crucial because the maximum simultaneous bandwidth to all disks is $\frac{\text{link speed} \times \text{number of links}}{\text{average hops}}$. Systems with a small number of links but few average hops may perform better than systems with more links but longer average path distances between the disk and the server. For example, a $16 \times 16 \times 16$ torus might seem like a good topology because it is relatively low cost and is easy to assemble. However, this topology would have $4096 \times 6/2 = 12288$ links, and the average distance from an edge to a desired node would be $16/4 + 16/4 + 16/4 = 12$ hops. This would limit the theoretical bandwidth to $1 \times 12288/12 = 1024$ Gb/s, or 128 GB/s at most, which might be insufficient to meet the 100 GB/s demand. Figure 6 shows the expected number of hops for each network topology as well as the expected load on each network link.

Our models have assumed that links are half-duplex. If full-duplex links are used, individual links can double their theoretical maximum bandwidth. However, this doubling is only realized if the load on the link is the same in both directions. For mesh and torus topologies the load in both directions will depend on the location of the router nodes. However, for butterfly and fat-tree topologies, the ability to do full-duplex transmission is largely wasted because, for large reads, data flows in only direction through the network: from disks to routers. Large writes work similarly—data flows in one direction only, from routers to disks.

A prime consideration is the ability of individual connections into the storage fabric to supply the necessary bandwidth. For a 4096 node system supplying 100 GB/s, we have assumed that 128 external connections would be sufficient; certainly, 128 links at 10 Gb/s could indeed provide 100 GB/s of aggregate bandwidth. However, designs in which individual nodes have relatively few links cannot support such routers because the intra-fabric bandwidth available to a router is too low to support an external link of 10 Gb/s. For example, a two-dimensional fabric has four ports per node; at 1 Gb/s/port, the total bandwidth available to an external link is about 4 Gb/s, far less than the 10 Gb/s capacity of the external link and insufficient to allow 128 such nodes to provide 100 GB/s to the outside world.

Figure 6 shows that the link bandwidth required by the 4-D, 5-D and 6-D configurations could be served by 1 Gb/s interconnects, such as Gigabit Ethernet. Unfortunately, the bandwidth needs of the 2-D and 3-D configurations require the use of a faster interconnect, such as 10 Gigabit Ethernet, to meet the 100 GB/s requirement of the overall system. The cost of the necessary 10 Gb/s ports add $80M to the 2-D configuration, and a whopping $100M to the 3-D configuration.

Figures 6 and 7 make it clear that, although low-dimensionality torii are attractive because of the low number of links they require, they cannot meet the 100 GB/s requirement without resorting to more costly interconnects. On the other hand, high-dimensionality hypercubes require less bandwidth per link, yet have many links and require switches with many ports. The 4-D and 5-D torii appear to have the best combination of relatively low cost, acceptable bandwidth on individual links and reasonable path lengths. Compared to butterfly networks, the resiliency of the 4-D and 5-D torii offset the 30% to 40% added cost.

The 6-D hypercube has the highest cost and highest aggregate throughput performance. Dividing the cost by aggregate throughput, as shown in Figure 8, shows that the cost per GB/s of bandwidth is nearly identical for the butterfly and hypercube topologies. The 6-D hypercube becomes a cost effective choice for a large reliable system with quality of service constraints.

5. Simulation Results

While analytic results show theoretical performance, there are many real-world considerations that affect performance. We simulated usage of the networks whose performance was analyzed in Section 4, using a simple simulator that modeled network traffic loads along the network links. Each router between outside clients and disks made 0.5 MB requests of randomly-selected data that induced load sufficient to drive the overall system bandwidth to 100 GB/s. Requests were routed through mesh, torus, and hypercube interconnection networks using dimensional routing [4]; routing in butterfly networks is fixed because there exists only one route between requester and disk.

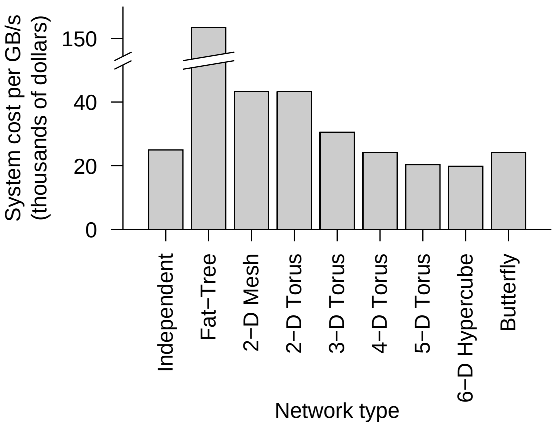
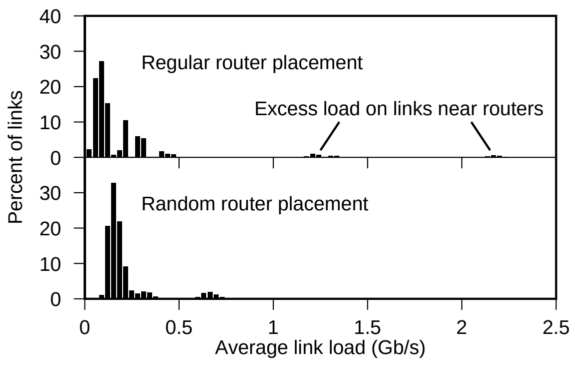


Figure 8. Cost per gigabyte per second for different interconnection network topologies.

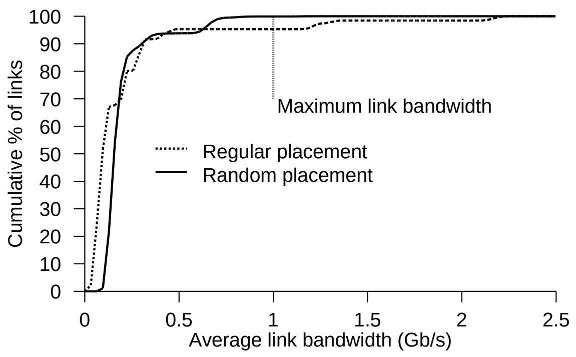
We generated a range of interconnection networks with 4096 nodes, including either 128 routers through which clients would connect to the storage fabric; the remainder of the nodes were storage nodes. The butterfly networks, on the other hand, had 4096 nodes connected through a switching fabric to 128 routers. The specific fabrics we tested are listed in Table 1.

In our first cube-style networks (meshes, torii, and hypercubes), we connected external clients through routers placed at regular locations within the network. This resulted in very poor performance due to congestion near the router nodes, as shown in Figure 9. A histogram of load distribution on individual links in a $4 \times 4 \times 4 \times 4 \times 4 \times 4$ hypercube is shown in Figure 9(a). When routers were placed in a regular arrangement, some links had bandwidths of 1.25–2.25 Gb/s. Individual disks require about 25% of a single 1 Gb/s link's bandwidth and are not affected greatly by requests that "pass through" the switch next to the disk. Routers, on the other hand, require nearly the full bandwidth of all the incoming connections. When routers are adjacent, the bandwidth is greatest nearest the routers and falls off further from the routers, resulting in overloaded links in part of the fabric and underloaded links elsewhere. Figure 9(a) shows that, in addition to overloading some links, regular placement *underloads* most of the remaining links—the histogram is shifted to the left relative to that for random node placement. The cumulative distribution of link load is shown in Figure 9(b); under regular placement, about 5% of all links experience overload.

We addressed the problem of crowding by placing routers randomly throughout the storage fabric. While this



(a) Histogram of link load.



(b) Cumulative distribution of link load.

Figure 9. Distribution of load on links in a $4 \times 4 \times 4 \times 4 \times 4 \times 4$ **hypercube. Randomly-placed router nodes improve the evenness of load in the fabric.**

did not decrease average path length, it dramatically reduced the congestion we noticed in our original network designs, as Figure 9 shows. As a result, bandwidth was spread more evenly throughout the storage fabric, reducing the maximum load on any given link. We believe that it might be possible to further balance load by devising an optimal placement; however, this placement is beyond the scope of this paper.

6. Future Work

This paper merely scratches the surface of issues in network design for petabyte-scale storage systems. Some of the unanswered questions about this design can be answered best by building an inexpensive proof of concept system using commodity drives, gigabit networking, and small-scale switches. This setup would allow us to verify our models against a small system, providing some confidence that the large systems we are modeling will perform as expected.

As with many other computer systems, changes in the ratios between disk bandwidth and network bandwidth will also affect storage system design. For example, when 10 Gb/s network connections become inexpensive, it is likely that designs with multiple disks per switch will become feasible. Given the aggregate bandwidth limitations using 1 Gb/s links, however, placing two or three disks per switch will overload the network for most topologies.

Of course, standard issues such as protocol choice, storage system network congestion, and reliability concerns are relevant to systems in which storage is embedded in a network fabric. However, other questions such as the placement of connections into the network (edge or core), the use of a few “shortcut” links to reduce dimensionality, and other problems specific to dense interconnection networks will be relevant to designs such as those presented in this paper.

Perhaps the most important question, though, is whether this design is applicable to commercial environments, in which bandwidth is less crucial, as well as scientific computing environments. If this design is a good fit to commercial systems, it is likely that the “bricks” used to build a storage fabric will become cheaper, allowing the construction of higher performance scientific computing storage systems as well as faster, more reliable commercial storage systems.

7. Conclusions

In this paper, we have introduced the concept of building a multiprocessor-style interconnection network solely for storage systems. While this idea has been alluded to in the past, our research shows the tradeoffs between different configurations and demonstrates that “storage fabrics” based on commodity components configured as torii and hypercubes improve reliability as well as performance. More specifically, the 4-D and 5-D torii appear to be reasonable design choices for a 4096 node storage system capable of delivering 100 GB/s from 1 PB. Furthermore, these designs become faster as the system grows, removing the need to replace the entire storage system as capacity and bandwidth demands increase. It is for these reasons that we believe that storage network topologies as described in this paper will become crucial to the construction of petabyte-scale storage systems.

Acknowledgments

Ethan Miller was supported in part by Lawrence Livermore National Laboratory, Los Alamos National Laboratory, and Sandia National Laboratory under contract B520714. The Storage Systems Research Center is supported in part by gifts from Hewlett Packard, IBM, Intel, LSI Logic, Microsoft, Overland Storage, and Veritas.

References

[1] W. C. Athas and C. L. Seitz. Multicomputers: message-passing concurrent computers. *IEEE Computer*, 21:9–24, Aug. 1988.

[2] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2), June 1994.

[3] P. F. Corbett and D. G. Feitelson. The Vesta parallel file system. *ACM Transactions on Computer Systems*, 14(3):225–264, 1996.

[4] D. Culler, J. P. Singh, and A. Gupta. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, 1999.

[5] S. Frølund, A. Merchant, Y. Saito, S. Spence, and A. Veitch. FAB: Enterprise storage systems on a shoestring. In *Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOS-IX)*, Kauai, HI, May 2003.

[6] G. A. Gibson and R. Van Meter. Network attached storage architecture. *Communications of the ACM*, 43(11):37–45, 2000.

[7] R. J. Honicky and E. L. Miller. Replication under scalable hashing: A family of algorithms for scalable decentralized data distribution. In *Proceedings of the 18th International Parallel & Distributed Processing Symposium (IPDPS 2004)*, Santa Fe, NM, Apr. 2004. IEEE.

[8] IBM Company. IceCube – a system architecture for storage and Internet servers. http://www.almaden.ibm.com/StorageSystems/autonomic_storage/CIB_Hardware/.

[9] D. D. E. Long, B. R. Montague, and L.-F. Cabrera. Swift/RAID: A distributed RAID system. *Computing Systems*, 7(3):333–359, 1994.

[10] S. J. LoVerso, M. Isman, A. Nanopoulos, W. Nesheim, E. D. Milne, and R. Wheeler. *sfs*: A parallel file system for the CM-5. In *Proceedings of the Summer 1993 USENIX Technical Conference*, pages 291–305, 1993.

[11] E. L. Miller and R. H. Katz. RAMA: An easy-to-use, high-performance parallel file system. *Parallel Computing*, 23(4):419–446, 1997.

[12] N. Nieuwejaar and D. Kotz. The Galley parallel file system. In *Proceedings of 10th ACM International Conference on Supercomputing*, pages 374–381, Philadelphia, PA, 1996. ACM Press.

[13] P. Schwan. Lustre: Building a file system for 1000-node clusters. In *Proceedings of the 2003 Linux Symposium*, July 2003.

[14] R. Stevens. Computational science experiences on the Intel Touchstone DELTA supercomputer. In *Proceedings of Compcon '92*, pages 295–299. IEEE, Feb. 1992.

[15] R. O. Weber. Information technology—SCSI object-based storage device commands (OSD). Technical Council Proposal Document T10/1355-D, Technical Committee T10, Aug. 2002.

[16] Q. Xin, E. L. Miller, D. D. E. Long, S. A. Brandt, T. Schwarz, and W. Litwin. Reliability mechanisms for very large storage systems. In *Proceedings of the 20th IEEE / 11th NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 146–156, Apr. 2003.