Impact of Failure on Interconnection Networks for Large Storage Systems

Qin Xin[†] qxin@cs.ucsc.edu
Ethan L. Miller[†] elm@cs.ucsc.edu
Thomas J. E. Schwarz, S. J.[‡] tjschwarz@scu.edu
Darrell D. E. Long[†] darrell@cs.ucsc.edu

[†]*Storage Systems Research Center, University of California, Santa Cruz*
[‡]*Computer Engineering Department, Santa Clara University*

Abstract

Recent advances in large-capacity, low-cost storage devices have led to active research in design of large-scale storage systems built from commodity devices for supercomputing applications. Such storage systems, composed of thousands of storage devices, are required to provide high system bandwidth and petabyte-scale data storage. A robust network interconnection is essential to achieve high bandwidth, low latency, and reliable delivery during data transfers. However, failures, such as temporary link outages and node crashes, are inevitable. We discuss the impact of potential failures on network interconnections in very large-scale storage systems and analyze the trade-offs among several storage network topologies by simulations. Our results suggest that a good interconnect topology be essential to fault-tolerance of a petabyte-scale storage system.

1. Introduction

System architects are building ever-larger data storage systems to keep up with the ever-increasing demands of bandwidth and capacity for supercomputing applications. While high parallelism is attractive in boosting system performance, component failures are no longer exceptions. In a petabyte-scale storage system with thousands of nodes and a complicated interconnect structure, strong robustness in network interconnections is highly desired but difficult to achieve.

Failures, which appear in various modes, may have several effects on a large-scale storage system. The first is connectivity loss: requests or data packets from a server may not be delivered to a specific storage device in the presence of link or switch failures. The result is disastrous: many I/O requests will be blocked. Fortunately, today's storage systems include various levels of redundancy to tolerate failures and ensure robust connectivity. The second effect is bandwidth congestion caused by I/O request detouring. Workload analysis of a large Linux cluster with more than 800 dual processor nodes at Lawrence Livermore National Laboratory [19] shows that I/O requests are very intensive in supercomputing environments. They arrive on average about every millisecond. The average size of a single I/O request can be as large as several megabytes. Suppose that such a large system suffers a failure on a link or delivery path on an I/O stream. In this case, the stream has to find a detour or come to a temporary standstill. The rerouting will bring I/O delays and bandwidth congestion and might even interrupt data transfer. The I/O patterns particular to supercomputing demand a network architecture that provides ultra-fast bandwidth and strong robustness simultaneously. The third effect is data loss caused by the failure of a storage device. As disk capacity increases faster than device bandwidth, the time to write and hence to restore a complete disk grows longer and longer. At the same time, the probability of single and multiple failure increases with the number of devices in the storage system.

Current petabyte-scale storage system designs, such as Lustre [3], rely on a high-speed storage area network to provide required bandwidth, but do not address fault tolerance. Inspired by parallel computing architecture, recent research [10] proposes using switch-attached storage devices to create scalable and robust network interconnection and explores several potential topologies from butterfly networks to hypercubes. We focus on the failure impact on network interconnection for systems built from various topologies. We investigate the failure-resilient capacity and compare the trade-offs among several network interconnection architectures for a petabyte-scale storage system. We consider various degraded modes in which a certain number of links and (or) nodes fail and examine the impact of these failures on such a system. By simulation, we observe that a well-chosen network topology such as mesh and hypercube can still guarantee good network interconnection under various failure modes. At the same time, neighbors of the failed nodes and links suffer much more from bandwidth congestion than average. Our results indicate that an adaptive network routing protocol is needed for such a large system in order to solve the hot-spot problems brought by various failures in the system.

2. Related Work

Designs for parallel file systems, such as Vesta [5] and RAMA [13], were aimed at high performance computing, but did not consider the large scale of today's systems and the impact of failures which comes along with such a scale.

Industry solutions, such as IBM TotalStorage Infrastructure [11], EMC SAN arrays using fibre channel switches [12], and Network Appliance Fibre Channel SAN storage solutions [16], can support up to tens of terabytes data capacity. Our work is aimed for the design of petabyte-scale storage systems that can provide non-stop data delivery in the presence of failures.

Hospodor and Miller [10] explored potential interconnection architectures for petabyte-scale storage systems. The main concern of their work is efficient network interconnection between storage nodes, routers, switches and servers; however, they do not consider the consequences of failures on the network.

One of the main approaches to achieve fault-tolerant interconnection networks is adding redundant nodes and links in arrays and meshes. Blake and Trivedi [2] analyzed the reliability of multi-stage interconnection networks and showed that adding intra-stage links is more beneficial than adding an extra stage. Zhang [21] designed fault-tolerant graphs with small degree for good scalability and small number of spare nodes for low cost.

Resilient routing is crucial for interconnection network reliability in the face of link outages. Vaidya *et al.* [18] studied fault-tolerant routing algorithms in a multiprocessor system. Their focus was Duato's routing algorithm—Double East-Last West Last (DELWL) [8] in a 2D mesh topology. The system they examined was on the order of thirty to forty nodes; while our study is for much larger scale storage networks—on the order of thousands of nodes. In addition, we have switches and routers which differ in functionality from storage nodes.

The MIT Resilient Overlay Networks (RON) architecture [14] provides reactive routing to path failure detection and recovery on large-scale, Internet-based distributed systems. It monitors the quality of paths by frequent probing between nodes, and stores probe results in a performance database. In principle, there is much less control on the overall architecture of an Internet network such as RON than our storage network. Also, we consider both node failure and link failure while RON only focuses on path failure.

3. Network Interconnection and Failure Impacts

Modern supercomputing systems require a high bandwidth storage network storing petabytes of data. Traditional storage architectures, such as RAID [4], Storage Area Network (SAN) [17], and Network Attached Storage (NAS) [9] cannot meet the needs for bandwidth and scale of such a large storage system. Network topologies for massively parallel computers are better suited to build large storage networks [10], as they are capable of delivering high performance and dealing with very large scale.

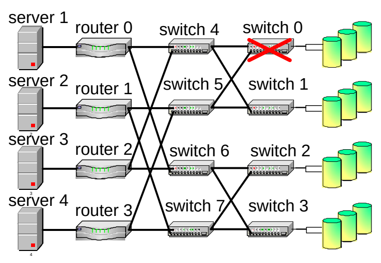
3.1. System Overview

Our research targets for a storage system with multi-petabytes of data. Such a system typically consists of over 4,000 storage devices, thousands of connection nodes (routers/switches/concentrators), and tens of thousands of network links. There is a high reliability demand on our system because data is difficult, perhaps even impossible to regenerate, and may not be reproducible. To achieve necessary failure tolerance, we store data redundantly. The choice of redundancy mechanisms is decided by the desired system cost, workload characteristics, and performance criteria. For instance, if the storage cost is one of the main concern and the system is primarily read-only, erasure correcting coding would be a good redundancy scheme for the system. When small writes appear frequently in the system and high storage redundancy is affordable, pure replication (mirroring) will fit the design. The data recovery process differs with redundancy mechanisms. If we simply mirror the data, then we just ask for the node(s) where the the replica is stored. If erasure coding is configured for the system, then the request is routed to a number of other storage servers that collectively can rebuild missing data when a storage device becomes inaccessible. We discuss tradeoffs among several data redundancy schemes and the use of data declustering for expediting the data repair process in our previous work [20].

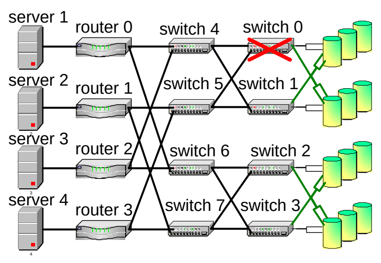
3.2. Network Interconnect Architectures

There are several potential strategies for interconnect architectures, such as a simple hierarchical structure, butterfly networks, meshes, and hypercubes [1]. The analysis of the total system cost and comparison of system bandwidth under failure-free status of each strategy were discussed in prior research [10]. Here we focus on the failure impact under these topologies.

The simple tree-based hierarchical structure requires large routers, which make the configuration very expensive [10]. Also, it cannot provide sufficient fault tolerance



(a) A failure on switch 0 disconnects several devices.



(b) Add spare links between switches and devices.

Figure 1. Butterfly Networks Under Failures.

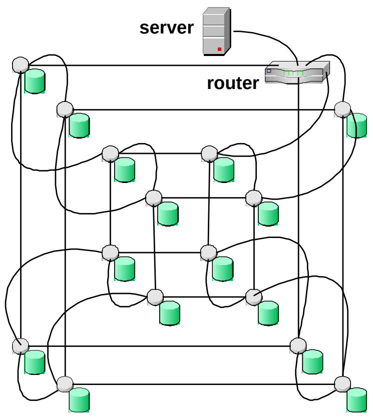


Figure 2. A Hypercube Structure.

as it suffers from the failures of higher-level nodes. Butterfly networks cannot offer high reliability either, because there is only a single path from a server to a storage device. For example, when a failure occurs at switch 0, as shown in Figure 1(a), a number of storage devices will lose their connections to the system. One way to solve this problem is to add spare links as shown in Figure 1(b). However, switch 1 then becomes over-loaded as all the requests to the storage devices attached with switch 0 will now go through it. It also requires more expensive switches with more ports for spare links. Furthermore, there may be additional link outages on the path from a server to a storage device, which will break the connection.

Cube-like architectures, such as meshes (Figure 3(a)), hypercubes (Figure 2), and torus are structured with multiple routes between servers and storage devices, and thus more resilient to failures. However, the system cost for these cube-like structures is higher than that for simple tree structure and butterfly networks.

3.3. Failure Scenarios

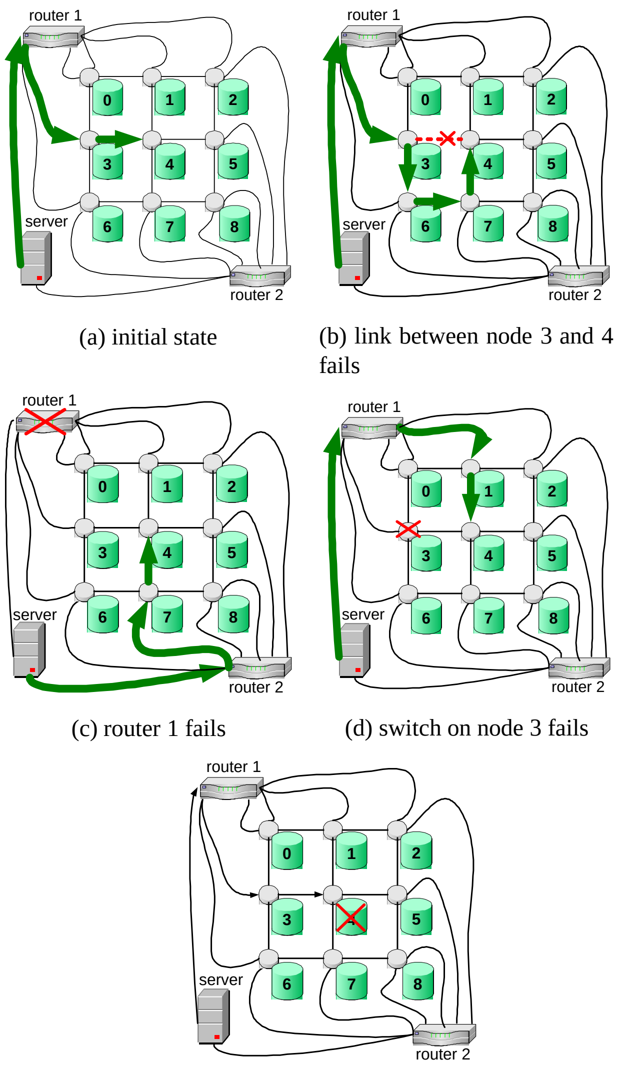
We consider three types of failure scenarios: link failure, connection node failure, and storage device failure.

I. *link failure*: The connection between any two components in a system can be lost. If there exists only one path between two components, a system is at risk when any link along this single path is broken. A robust network interconnection must be tolerant of link failures. Multiple paths between two components will decrease the vulnerability of a single-point of failure and effectively balance I/O workload.

II. *connection node failure*: Connection nodes include switches, routers, and concentrators that link servers to storage nodes. They are used for communications and do not store any data. Compared with link outage, failures on an entire switch or router are more harmful for network connection since a number of links that were attached on the switch or the router are simultaneously broken, but losing connection nodes will not directly lead to data loss.

III. *storage device failure*: When a storage device fails, it cannot carry any load. Further, additional traffic for data reconstruction will be generated. The increase in bandwidth utilization brought by data construction is of great concern when data is widely declustered in such a system. We modeled and analyzed several redundancy mechanisms for such very large storage systems in our previous work [20].

As a case study, we examine four kinds of possible failures in a 3×3 2D mesh storage system shown in Figure 3. In our example, there are nine storage devices (labeled from 0 to 8), two routers and one server. Assume a specified I/O stream will be sent from the server to a storage device, say node 4. We trace the path of this I/O stream in various scenarios. At the initial state without any failures, the I/O request can be simply transferred via router 1 and node 3 as indicated in Figure 3(a). However, a rerouting strategy has to be introduced if failure occurs. If the link between node 3 and node 4 is broken, as shown in Figure 3(b), the I/O request has to take an alternate route to get to its target. It can pick up a path (i) {router 1→node 3→ node 6→node 7→node 4} as shown in Figure 3(b), or (ii) {router 1→node 1→node 4}, or (iii) {router 2→node 7→node 4}, etc. The choice of the new



(a) initial state

(b) link between node 3 and 4 fails

(c) router 1 fails

(d) switch on node 3 fails

(e) node 4 fails

Figure 3. A Storage System Structured as a 3×3 2D mesh Under Degraded Modes

path is determined by the routing protocol and system status at that moment. Figure 3(c) and 3(d) show further examples of detouring due to a router and a switch failure respectively. The worst case in failure modes is that the target node fails, as shown in Figure 3(e). If either the switch fails or the disk drive crashes on the target node, the I/O request cannot be delivered.

4. Evaluation

It is expensive to build a real petabyte-scale storage system in order to evaluate the impact of failures on interconnection networks. Instead, we use simulations of large-scale systems to develop a better understanding of the impact of failures. We simulate several network interconnection topologies and inject varied failure scenarios to evaluate system behavior under degraded mode and estimate system reliability of a petabyte-scale storage system.

4.1. Assumptions

Generally, nodes are classified into two types: *storage nodes* that contain data, such as disk drives; and *connection nodes* that are used only for communication, such as routers and switches. We investigate node failure and link failure in our system. In reality, there are many other types of failure, such as power failure and software failure. Our failure model simply focuses on network interconnection, but does not consider the Byzantine failure model under which arbitrary or malicious failures would appear. We also assume all failures be detected in a timely manner.

We assume I/O requests to be very intensive and in large size. User data is spread out over the whole system evenly.q We use Dijkstra's algorithm [6] as our routing algorithm. This algorithm helps us understand the network status and trace the path of each I/O request, although it cannot scale to large networks due to its dependence on global information. We are considering failure-resilient routing techniques, such as wormhole routing [15], in our ongoing work. We do not consider the buffer/cache issues of routers and switches for simplification.

4.2. Simulation Methodology

We evaluate impact of failures on a petabyte-scale storage system under various interconnection configurations by event-driven simulation. The simulator, implemented in C++, can evaluate the failure impact on a system under various configurations. There are three main pieces in our simulator: topology, failures, and requests. The network interconnection architecture was implemented as a class object Graph with the functions for building the network topology, *i.e.* build_nodes and build_links. The function inject_failures sets up the degraded system mode under which one or more failures happen in an overlapped time period. Servers send out I/O requests under a synthetic workload based on our analysis of a Linux cluster for supercomputing applications [19].

We have simulated three kinds of topologies for our system: a multi-stage butterfly network, a 64×64 2D mesh shown in Figure 3(a) and a 6D hypercube shown in Figure 2. We expect to include several other topologies such as butterfly network, torus, and tower graph [21] in the full paper. Previous work [10] estimated the required number of nodes and ports for a petabyte-scale storage system using butterfly, mesh and hypercube topology. We list the parameters set up in our simulator in Table 1. The but-

Table 1. Parameters for butterfly, mesh and hypercube topology.

parameter	butterfly	mesh	hypercube
number of servers	128	128	128
number of disks	4096	4096	3968
number of routers	128	8	128
total number of links	7552	16,392	23,612
total number of nodes	4736	4232	4224

terfly network is a hierarchical structure with one level of routers, three levels of switches with 128 switches per level. In the 64×64 2D mesh, each router is connected to the edge nodes and the interior nodes are connected with four other nodes. In a hypercube topology, each storage device is attached to a 12-port switch and each router has two additional ports connected to servers.

4.3. Simulation Results

In our simulated system, servers send I/O requests in parallel to storage devices at an interarrival rate of 1 millisecond. We set up several degraded scenarios for three network topologies—butterfly network, 64×64 2D mesh, and hypercube topologies, including varied number of link failures and node failures. We trace the I/O requests and count the number of hops for each request and record the load on each link in the system. We calculate the ratio of the requests which cannot be delivered to the target device due to failures under various degraded modes, to measure how well the system is connected. We also show the average number of hops of I/O requests under varied degraded modes and compare the link load in the neighborhood of failures with average links.

4.3.1. I/O path connectivity

The first and the most important aspect of robustness of network interconnection is that an I/O request can be delivered to its target storage device. We refer to such an ability as *I/O path connectivity*. We borrow the metric of system availability [7] and measure the connectivity in units of "nines," which is defined as $-log_{10}(1-P)$, where P is the fraction between the number of I/O requests that can be successfully sent to the targets and the total number of I/O requests during a period of time. Three "nines" connectivity means that 99.9% of the I/O requests can be delivered to their targeted storage devices.

We trace all the I/O requests sent in 60 seconds under seven failure modes: with one, two routers failed, with one, two switches failed, and with four, eight, and sixteen links failed. Failures cannot be repaired during 60 seconds even if they can be detected. Our results are reported in Figure 4. We found that failures on switches have greater influence than those on routers and links. The butterfly network suffers greatly from broken switches, as we discussed in Section 3.2. As expected, the 6D hypercube and 2D mesh structure achieve a better connectivity than the butterfly network, although up to 0.05% of the requests did not arrive at their target devices when two switches failed. As for link failures, every I/O request found a healthy path in the presence of up to sixteen broken network links under 2D mesh and 6D hypercube topologies, but about 0.048% of the requests were not delivered successfully when 16 links were broken under the butterfly network structure. Within 60 seconds, on the order of 10^8 I/O requests were sent from the servers in our simulation. As a result, the accuracy of our reliability measurement is up to eight nines ($-log_{10}(1-\frac{10^8-1}{10^8})=8$).

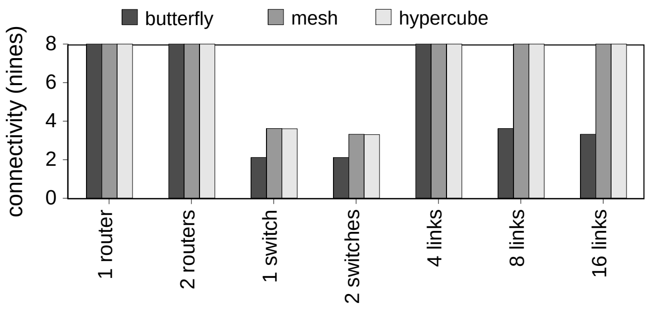
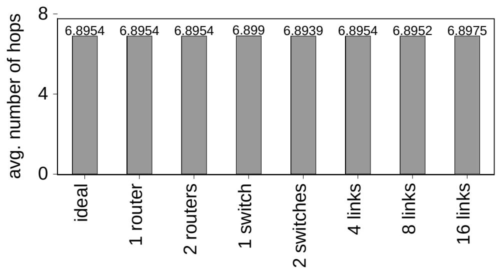


Figure 4. I/O path connectivity

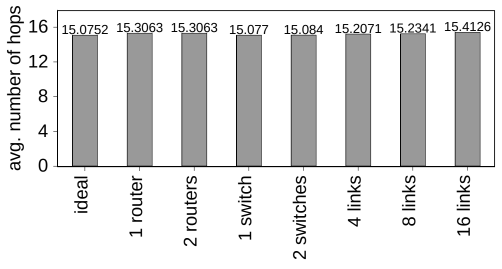
4.3.2. Number of hops for I/O requests

The number of hops is calculated as the number of links that an I/O request has to travel through the system to arrive at its targeted device. It is an important metric for both I/O latency and system bandwidth. We measure the minimum number of hops in the simulator; while in reality, an I/O request may go through more steps than the minimum for the considerations of load balance.

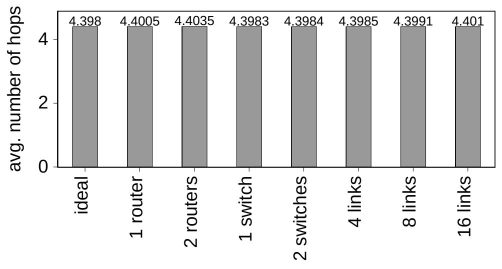
We compare an ideal fault-free case (labeled as "ideal") with seven degraded modes: with one, two routers failed, one, two switches failed, and with four, eight, and sixteen links failed (Figure 5). We do not count the case when there is no path for an I/O request in the calculation of the average number of hops. Compared with the ideal connection, the average number of hops is only slightly higher under all degraded modes in all three topologies. There are two underlying reasons for this: first, the proposed butterfly, mesh, and hypercube structures provide redundant paths and thus lead to good fault tolerance; second, the possibility that a failed component is on the path of many I/O requests is small due to the limited number of I/O requests during a short period time. As a result, the number of aver-



(a) Butterfly networks.

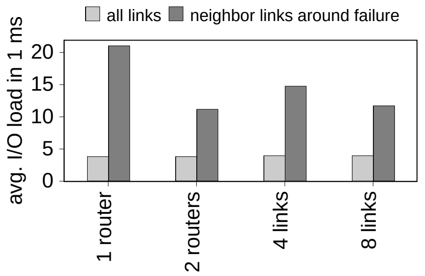


(b) 64 × 64 2D mesh.

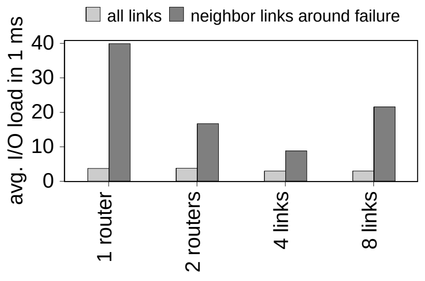


(c) 6D hypercube with 4,096 nodes.

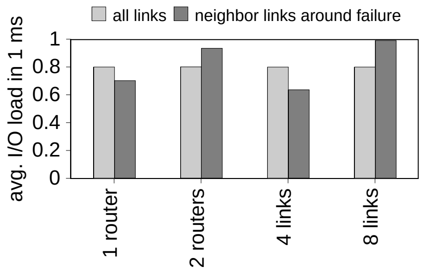
Figure 5. Average number of hop per I/O request



(a) Butterfly networks with 4,096 disk drives.



(b) 64 × 64 2D mesh.



(c) 6D hypercube with 4,096 nodes.

Figure 6. I/O load comparison of the neighborhood links around failures and all the links in a system.

age hops remains nearly at the same level under the examined degraded modes. For a well-chosen topology which does not suffer from a single point of failure, the system would be robust unless many components fail at once. This occurrence only happens under certain circumstances such as large-scale power outages, which can easily pull down any local-area networks.

4.3.3. Failure impact on network neighborhood

One of the important system behaviors after failures is request rerouting. The failure impact on its neighborhood links/nodes is not negligible. An abrupt increase in network load around the neighborhood of failures can overload a certain number of nodes and links, such that I/O requests may not be delivered successfully. In order to analyze the failure impact on the network neighborhood, we monitored the the network links around failures and compared their I/O load with the average load on all the links in the system. We observed a pronounced increase in the average I/O load on neighboring links around failures under four degraded modes: with one router, two routers, four links, and eight links failed (as Figure 6 shows.)

Comparatively, neighbors around a failed router carry more I/O load than those around a failed link in most cases. This phenomenon comes from the different functionalities of routers and links. We also note that neither the butterfly network nor 2D mesh structure balances the load around failures, but the hypercube topology handles it well. The link load around failures is four to thirteen times higher than average link load in the butterfly network and 2D mesh system, whereas it is not obviously higher than the average link load in the 6D hypercube structure. This is because there are fewer network links and much weaker path redundancy in the butterfly network and 2D mesh structure than those in the 6D hypercube topology. Our results indicate that for a petabyte-scale storage system, although butterfly network and mesh structure can provide decent I/O path connectivity without increasing the number of hops, they cannot deal with neighborhood load increase as gracefully as the 6D hypercube structure.

4.4. Result Summary and Discussion

In our simulated petabyte-scale storage system connected by a butterfly network, a mesh or a hypercube architecture, four to sixteen link failures do not result in an obvious increase in the number of hops for I/O requests. This shows good fault tolerance to link failures under these three network interconnection topologies. Switch failures are much more likely to cause I/O path disconnect. We found that the butterfly network can survive under router failures but is very sensitive to switch failures and sixteen link outages; while the average number of I/O hops in the 2D mesh structure is one to two times higher than that in the butterfly network and the hypercube structure. This indicates that different network topologies have their pros and cons under different system degraded modes. The impact of failures on neighborhood links is significant, especially for the multi-stage butterfly network and 2D mesh structure, which may lead to network congestion and the slowing down of data transfers. Based on our results, the hypercube structure pays off its higher system cost and outperforms the butterfly network and 64×64 2D mesh in the robustness of network interconnection.

5. Conclusions and Future Work

Robust network interconnects are essential to large-scale storage systems. We study various failure scenarios and their impacts on a petabyte-scale storage system. The fault-tolerance capacity of three potential network topologies, namely, multi-stage butterfly network, 2D mesh, and 6D hypercube structures, has been evaluated by simulation. We examined I/O path connectivity, the number of hops for I/O requests, and the I/O load on neighborhood links around failures. Our preliminary results have shown that a well-chosen network topology is capable of ensuring a high tolerance of failures. Router and switch failures have a larger impact on network robustness than link failures, and the neighborhood around failures suffers more greatly than average link load, especially for butterfly network and 2D mesh. Our simulator can flexibly investigate various network topologies with injection of any degraded modes, which enables us to estimate robustness of network interconnections and helps the system architects with their design decisions on building reliable petabyte-scale storage systems.

There are many other network topologies that we have not explored yet. For example, torus and tower graph are promising architectures for petabyte-scale storage systems. We plan to study the fault tolerance capacity of torus and tower graph structure in our future work. We are still investigating more complicated failure modes and system behavior under these degraded modes over longer intervals.

Acknowledgments

We thank the members and sponsors of Storage Systems Research Center, including Department of Energy, Engenio, Hewlett-Packard Laboratories, Hitachi Global Storage Technologies, IBM Research, Intel, Microsoft Research, Network Appliance, and Veritas, for their help and support. We also thank the anonymous reviewers and our shepherd Curtis Anderson for their comments that help us improve the paper. This work is funded in part by Lawrence Livermore National Laboratory, Los Alamos National Laboratory, and Sandia National Laboratory under contract B520714, and by IBM Research Grant 41102-COEN-RSCH-IG-IG09.

References

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

[2] J. T. Blake and K. S. Trivedi. Reliabilities of two fault-tolerant interconnection networks. In *Proceedings of the 18th International Symposium on Fault-Tolerant Computing (FTCS '88)*, pages 300–305, 1988.

[3] P. J. Braam. The Lustre storage architecture. http://www.lustre.org/documentation.html, Cluster File Systems, Inc., Aug. 2004.

[4] 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.

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

[6] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[7] J. R. Douceur and R. P. Wattenhofer. Large-scale simulation of replica placement algorithms for a serverless distributed file system. In *Proceedings of the 9th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '01)*, pages 311–319, Cincinnati, OH, Aug. 2001. IEEE.

[8] J. Duato. A theory of fault-tolerant routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 8(8):790–802, 1997.

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

[10] A. Hospodor and E. L. Miller. Interconnection architectures for petabyte-scale high-performance storage systems. In *Proceedings of the 21st IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 273–281, College Park, MD, Apr. 2004.

[11] IBM Corporation. Storage consolidation for large workgroups and departments: An IBM SAN business value solution, 2002.

[12] T. Joyce. NAS gateways allow IP access to SANs. Network World Fusion, "http://www.nwfusion.com", Apr. 2004.

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

[14] MIT RON (Resilient Overlay Networks) Project. http://nms.lcs.mit.edu/ron/.

[15] P. Mohapatra. Wormhole routing techniques for directly connected multicomputer systems. *ACM Computing Surveys*, 30(3):374–410, 1998.

[16] Network Appliance fibre channel SAN storage solutions. http://www.netapp.com/solutions/fcsan.html.

[17] W. C. Preston. *Using SANs and NAS*. O'REILLY, 2002.

[18] A. S. Vaidya, C. R. Das, and A. Sivasubramaniam. A testbed for evaluation of fault-tolerant routing in multiprocessor interconnection networks. *IEEE Transactions on Parallel and Distributed Systems*, 10(10):1052–1066, 1999.

[19] F. Wang, Q. Xin, B. Hong, S. A. Brandt, E. L. Miller, D. D. E. Long, and T. T. McLarty. File system workload analysis for large scale scientific computing applications. In *Proceedings of the 21st IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 139–152, College Park, MD, Apr. 2004.

[20] Q. Xin, E. L. Miller, T. J. Schwarz, D. D. E. Long, S. A. Brandt, 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.

[21] L. Zhang. Fault tolerant networks with small degree. In *Proceedings of the 12th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 65–69. ACM, 2000.