Hybrid Host/Network Topologies for Massive Storage Clusters

Asha Andrade
aandrade@cs.sunysb.edu

Ungzu Mun
umun@ic.sunysb.edu

Dong Hwan Chung
donchung@ic.sunysb.edu

Alexander Mohr
amohr@cs.sunysb.edu

Department of Computer Science, Stony Brook University

Abstract

The high demand for large scale storage capacity calls for the availability of massive storage solutions with high performance interconnects. Although cluster file systems are rapidly improving and have the potential to allow extremely large numbers of commodity storage nodes to be pooled into a single large file-system, the number of ports on individual switches has not been increasing as quickly -- the largest switches available today support fewer than 2,000 Gigabit Ethernet ports. Our goal, therefore, is to develop a new interconnect topology that can connect hundreds of thousands of nodes and achieve performance comparable to a single switch of equivalent size. At the same time, such a new topology should be readily buildable using inexpensive components. Our proposed architecture exploits the multiple Ethernet ports that are now standard on servers and combines host-based routing and forwarding with network-based switching to allow massively large storage clusters to be built. Simulation results have shown that our proposed design achieves 72% to 90% of the performance of a single switch capable of accommodating all storage nodes, but our approach scales to hundreds of thousands of nodes. Furthermore, we use common off-the-shelf layer-2 switches rather than more expensive models that support layer-3 routing. Finally, our approach is resilient to network faults because it maintains multiple paths between storage nodes.

1. Introduction

Today's high-performance computing places increasing demands on a storage system's capacity and the performance of its components [7]. Furthermore, the amount of data being created in scientific applications continues to increase. As an example, the ATLAS detector, which is just one of the experiments that is part of CERN's Large Hadron Collider (LHC), is expected to collect five to eight petabytes of data per year. The LHC project as a whole is expected to generate fifteen petabytes of data each year [1]. Assuming the availability of 1TB hard disks, ATLAS and LHC would require 5,000 to 15,000 such disks per year to store just one copy of the data set without replication.

One possibility for storing such a large amount of data would be to use a cluster-based file system on a large number of commodity storage nodes. Unfortunately, the current generation of cluster-based file systems is limited in the number of storage nodes it supports, although those limits are increasing with time. RedHat's GFS [6] supports up to 256 nodes. and the largest Google File System cluster in 2003 was composed of just over 1,000 storage nodes [8]. Other cluster file systems, such as Panasas's PanFS [5] and Cluster File Systems's Lustre [2] now support much larger limits: the largest production Lustre system has 15,000 nodes. Promising approaches that federate meta-data servers, such as Ceph [10] are likely push these limits to the hundreds of thousands.

Given such progress on the cluster-based file system front, we now turn to the interconnection topologies that tie these clusters together. The simplest approach is to use a single switch to connect the storage nodes in the cluster together, but this approach limits the cluster size to the largest number of ports available on a single switch. Unfortunately, the largest switches today support fewer than 2,000 Gigabit Ethernet ports, so any solution that scales to our target sizes will require multiple switches.

In order to overcome this limitation, we note that today's servers now come standard with multiple Ethernet ports, high-speed internal busses, and a large amount of processing power. We exploit that combination by proposing a hybrid architecture in which each port on a storage node is connected to a different layer-2 switch and that node acts as a software router to forward packets between those interfaces. The result is an interconnect that can connect hundreds of thousands of storage nodes with a design that achieves 72% to 90% of the performance of a single switch capable of accommodating those nodes, were such a switch to exist. By using common off-the-shelf layer-2 switches rather than more expensive models that support layer-3 routing, the cost of our design is minimized. Finally, because

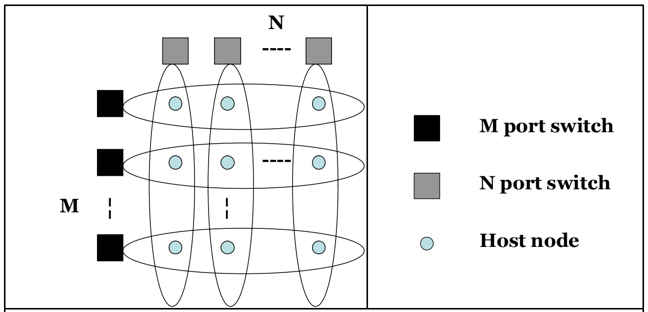

Figure 1. A 2 dimensional *M x N* hybrid design composed of *M N*-port switches and *N M*-port switches. Each storage node is required to have 2 ports.

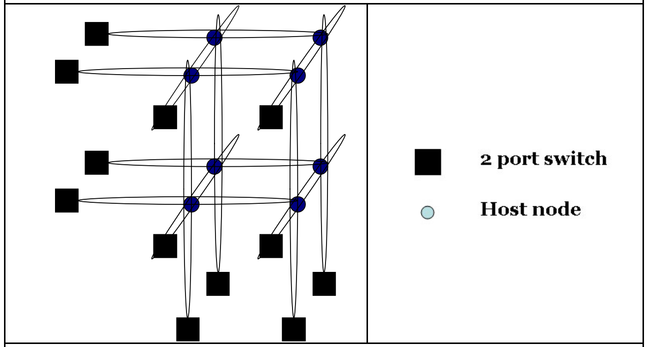

Figure 2. A *2 x 2 x 2* 3D hybrid storage cluster.

there are multiple paths between storage nodes, our design is resilient to large numbers of network faults.

Our paper is organized as follows: in Section 2 we detail our design approach, together with advice for practical implementation. Section 3 delves into the methodology used to determine the credibility of our approach. Here we provide the algorithm used for the purpose of simulation. Section 4 details our simulation results together with effects of switch failures. Finally, we summarize our conclusions in Section 5.

2. Storage Cluster Design

In this section, we turn to a hybrid design for interconnects that uses both nodes and switches to route packets. We describe the construction, explain how it could be built in practice, and discuss how clients would connect with the cluster. We note that numerous other researchers have looked at the general problem of topologies for storage clusters [7, 3, 9], including at prior instances of this conference [4, 11], and refer readers to those papers for details; in this section we focus on our design.

2.1. Our Hybrid Topology

Our goal is achieve performance comparable to a single massively-large switch, were such a switch to exist. To achieve that goal, we use common off-the-shelf layer-2 switches and exploit the presence of multiple Ethernet ports on storage nodes by enabling host-based routing and forwarding.

In our 2D hybrid, we arrange our storage nodes in an *M* by *N* grid, as shown in Figure 1. We augment that grid of storage nodes with an *M*-port switch at the top of each column and an *N*-port switch to the left of each row, for a total of *M* + *N* switches. Each storage node in this example has two Ethernet ports; one is connected to the switch at the top of its column and the other is connected to the switch along its row. This regular pattern of connections simplifies wiring compared to FNNs.

In order to move packets from a source at (*a, b*) to a destination at (*c, d*), the source can send a packet to either (*a, d*) or (*c, b*) via the switch that connects them. That intermediate node will then forward the packet to the destination, with which it also shares a switch. Thus, our 2D hybrid has two disjoint paths from source to destination, which could be used to benefit from multipath routing or improved resilience to switch failures.

Compared to using a single monolithic switch, we require two Ethernet ports per storage node versus the single switch's one. Also, the latency between source and destination increases due to the traversal of two switches rather than one in addition to the time required for forwarding at the intermediate node. However, compared to typical disk seek and rotation latencies, the additional latency caused by our approach is likely to be small. As we will see in Section 4, however, the simulation results show the bandwidth to be comparable.

To extend the 2D hybrid into a 3D hybrid, we would begin with *L* layers of *M*x*N* hybrids, add an additional switch layer containing *M*x*N* switches with *L* ports each, and connect each storage node to its corresponding switch in the switch layer, as shown in Figure 2 for a 2x2x2 network.

The 3D hybrid can greatly increase the number of storage nodes that can be connected together with switches of a given port count. Alternatively, for a fixed number of storage nodes, one can build a 3D hybrid interconnect using switches with far fewer ports per switch than the equivalent 2D hybrid. Another added benefit is that the 3D hybrid has much richer connection properties with many more paths between a source and destination. However, the additional dimension requires one more port on each storage node, more (but smaller) switches, and more cables; latency will also increase.

Should even larger clusters be required, the hybrid approach easily extends to 4D or higher in the obvious way. We thus achieve the goal of building an enormous cluster with impressive data forwarding capability.

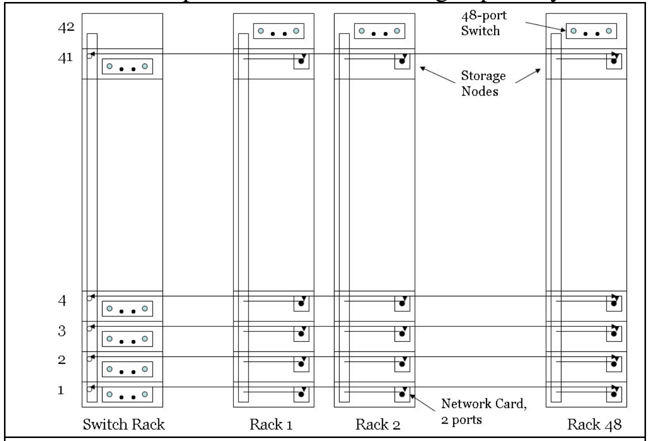

Figure 3. A two-dimensional hybrid model built using 49 standard 42U racks: the switch rack contains 41 switches with 48-ports each and racks 1 through 48 contain 1 switch and 41 nodes each. The end result is a cluster of 41 x 48 storage nodes.

2.2. Real World Constraints

When turning these abstract interconnect models into working storage clusters, one must account for real-world layout constraints. In this sub-section, we describe how one might go about building a storage cluster that uses our hybrid interconnect topology in the face of such constraints. We start with the trivial case of a 1D hybrid and then iteratively extend it to 2D and 3D.

Given a standard 42U rack, one would build the trivial one-dimensional hybrid model by placing 41 storage nodes at rack positions 1 to 41 with the 42nd position reserved for a 48-port switch. Each storage node is attached to a free port in the switch, thus obtaining a 41-node cluster.

A 2D version of the hybrid model is shown in Figure 3. It requires a rack of 41 switches with 48 ports each and a row of 48 racks of the type described in the previous paragraph. The first port (drawn as a filled circle) of each node is connected to the switch in the top of its rack, as before. The second port (drawn as an inverted triangle) of a storage node at height *i* is connected to the switch at height *i* in the switch rack. This results in a *41 x 48* cluster of 1968 storage nodes.

To build a 3D hybrid, we would start with 48 rows of the 2D hybrid described above and add an additional row of racks containing a *48 x 41* array of switches to which the third port of the storage nodes is attached. The resulting cluster is of size *41 x 48 x 48*, or 94464 nodes altogether. For larger cluster sizes, it would presumably make sense to use 96 port or larger switches rather than attempting to build a 4D hybrid.

Smaller clusters could also be built. A *41 x 24* 2D hybrid cluster of 984 storage nodes could be built using a switch rack containing 41 24-port switches. Alternatively, one could use 21 48-port switches in which nodes at heights 1 and 2 are connected to the same switch, and so on. The advantage of the latter approach is that if the cluster were later to double in size, one would only need to purchase 20 additional 48-port switches, install them in the switch rack, and wire the switches as in the *41 x 48* case. Thus, our topology can gracefully grow from a single rack cabinet to sizes necessary to support the data requirements of the LHC [1].

2.3. Connecting Clients

So far, we've discussed how storage nodes connect to each other, but neglected how clients connect to storage nodes. We envision a number of possible scenarios, depending on the number of clients and their request patterns.

If the number of clients is less than the number of storage nodes, perhaps the most straightforward approach would be to add an extra port to each storage node and directly connect the client to that port, assuming that the storage node also acts as a server for the cluster file system (and that appropriate firewall rules are in place to avoid unwanted attacks). If the number of clients is much larger than the number of servers, a similar approach could be employed, but using switches to fan out each connection to multiple clients.

3. Methodology

We programmed a software simulator in Python which implements the designs of our proposed storage cluster. The resulting storage cluster is run through a sequence of operations to determine its performance relative to a single switch and selected other designs. Prior to delving into details of the algorithm and the metrics used for comparison, we mention the underlying assumptions in the implementation of the simulator.

Because packet level tracing using ns-2 would have been too resource intensive for the cluster sizes we wanted to explore, our simulator works at the flow level. Our primary assumption is that when there is contention for the bandwidth of a link, whichever transport protocol is used will result in an allocation of bandwidth between flows that is approximately equal.

```
procedure compute_aggregate_rate_bw
compute shortest path between all nodes;
for each randomly generated  flow do
  for each link in flow do
    flow_cnt[link]++;
    bw[link] = capacity[link]/flow_cnt[link];
  done
done
for each link in order of non-increasing load do
  for each flow having link in its shortest path do
    bw[flow] = bw[edge];
    capacity[link] -= bw[flow];
  done
done
avg_bw = sum (bw[flow]) / flow cnt;
end
```

Figure 4. Algorithm to compute average bandwidth across a given network.

The sequence of steps shown in Figure 4 is used to determine the bandwidth assigned to each flow. We are interested in the performance metrics of aggregate bandwidth and average bandwidth per flow to compare our proposed designs with potential alternatives.

3.1. Simulation Parameters

We have run simulations for a single large switch, a mesh, a torus, FNN, and our 2D and 3D hybrids. In the case of the mesh, torus, and our proposed hybrid models, we need to explicitly specify the number of nodes along each dimension, i.e. for a mesh it would be something along the lines 48 x 48, but for a 3D hybrid it would be 48 x 48 x 48 and so on. We use combinations of commonly available switch sizes, such as 8, 16, 24, 48, 288, and 630 ports. For the FNN interconnects, we generated topologies from the public topology generator on their web page [3].

4. Simulation Results

We assess the performance of the various approaches by imposing three categories of loads: low, medium, and high. In the low load case, the number of flows is one-tenth the number of storage nodes in the system. For the medium load case, the number of flows is equal to the number of nodes, while for the high load case, the number of flows is 10 times the number of storage nodes. We assume that the links in the system are all gigabit Ethernet links.

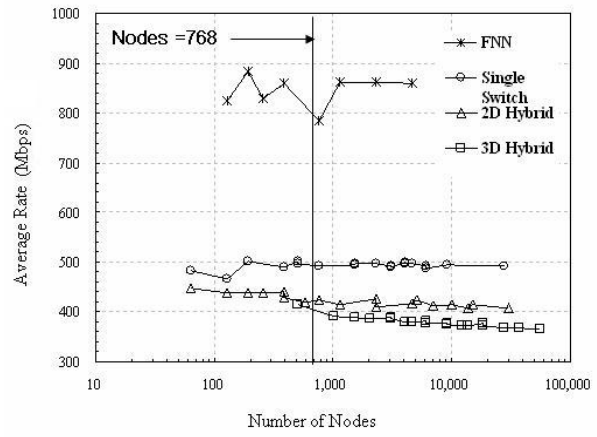

Figure 5. Average bandwidth vs. node count when the number of flows across the network equals the node count. The 2D and 3D Hybrid models are our designs.

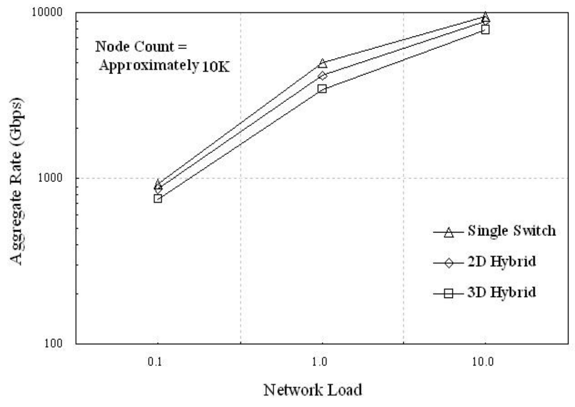

Figure 6. Aggregate bandwidth when the number of flows is 1/10th the node count, equal to node count and 10 times the node count when the node count is 10,000.

4.1. System Bandwidth

In Figure 6, we plot the average bandwidth versus number of nodes in the cluster for the medium load case in which the flow count equals the node count for all of our proposed designs. FNN performs extremely well up to 4608 nodes, but at that point was using storage nodes with 7 ports each and switches with 1152 ports. Given such results, it seems infeasible for FNNs to scale to the massive sizes we are interested in. Our proposed 2D hybrid achieves an average bandwidth value that is 81%

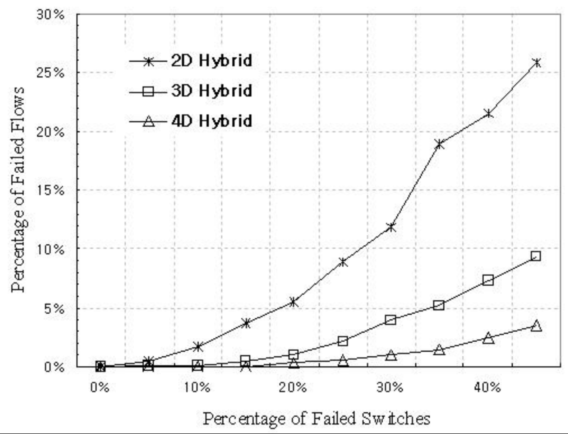

Figure 8. Percentage of failed flows for our proposed hybrid models. The node count used is 4096, with flow count being ten times the node count.

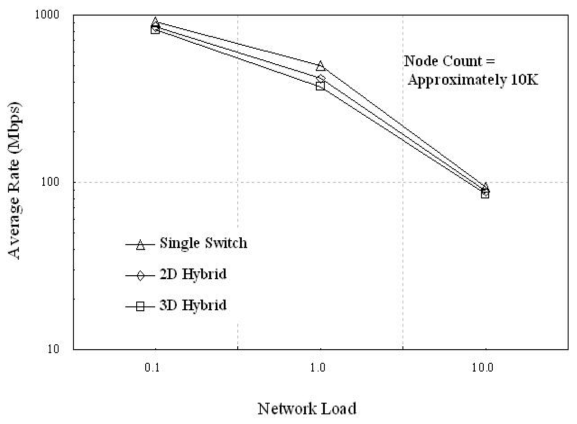

Figure 7. Average bandwidth per flow vs. network load factor.

to 90% of the bandwidth of a single switch; the 3D hybrid achieves 72% to 82% of a single switch.

We also look at the data of Figure 6 in more detail for the specific case of 768 nodes. While the single-switch model with a hypothetical 768-port switch results in an average value of approximately 490 Mbps, our 2D hybrid reports 415 Mbps utilizing a total of 64 switches of which 48 are 16-port and 16 are 48-port. The 3D model on the other hand reports 397 Mbps utilizing 320 switches. The FNN however achieves 990 Mbps utilizing 11 288-port switches. We note here that our models perform closely with the single-switch model, using inexpensive switches, while the FNN is a restrictive model using 11 highly priced 288-port switches.

The plot of average rate in Figure 6 also depicts the consistency in the average bandwidth utilization for our proposed designs, although there is a slight decrease with increasing number of nodes.

Figures 7 is a comparison of the average bandwidth per flow for a hybrid topology with 10,000 nodes for the three levels of load we examined. While there is a significant increase in the aggregate utilizations with an increased load, the average shows a decline. We see that our results closely track those for the single switch.

We speculate that the slight decrease in performance for 3D hybrid compared to 2D hybrid comes from increased contention in the network. A single switch must only deal with contention on the links connecting the source and destination, whereas the 2D hybrid must deal with it at those locations as well as the intermediate node. Because our bandwidth allocation assigns a rate to a flow based on its slowest bottleneck, flows that traverse more links are more likely to see a slower bottleneck due to statistical variation.

Another possibility is that because the simulator currently selects the first shortest-path link, flows are having more contention than they would if a random shortest path were selected. It is possible that we could mitigate this effect by adding random selection to our simulator, or by choosing the least loaded among the set of best paths.

4.2. When Switches Fail

Robustness of a storage system is crucial for an organization, so we also investigate the behavior of various topologies under conditions of switch failure. Simulation measurements for a network with 4096 hosts and 40,960 flows reveals that robustness depends largely on the type of topology.

In Figure 8 we plot the failed flow percentages against the percentage of switch failures. We are thus effectively determining the percentage of connections that cannot be established given a switch failure fraction. In the extreme case where 50% of the switches have failed, 24% of the flows in a 2D hybrid model cannot be established, whereas it is 9% in case of the 3D hybrid and a small 3% in the case of the 4D hybrid. In particular, a 2D hybrid cluster is less robust than the corresponding 3D and 4D models due to the absence of one and two levels of alternate routes respectively. With additional dimensions, the probability of having a significant portion of the network connected is much greater that it would be otherwise.

In Figure 9, we plot the average rate of the surviving flows as a function of the percentage of failed switches. We report a 33% drop in the average rate when 50% of the switches in the cluster fail, while it being less than 6% with 5% of switch failures.

5. Conclusion

In this paper we have presented a new approach to the design of cluster interconnects networks that can connect hundreds of thousands of nodes and achieve performance results comparable to a single switch of equivalent size. Our proposed architecture exploits the multiple Ethernet ports that now ship as standard on servers and combines host-based routing and forwarding with network-based switching to allow massive storage clusters to be built using inexpensive layer-2 switches. Simulation results show that our 2D hybrid achieves 81% to 90% of the performance of a single switch and the 3D hybrid achieves 72% to 82% of its performance. Furthermore, our design is practical to deploy in datacenters and is resilient to even significant numbers of failed switches. Our hybrid topology's potential to enable cluster file systems to meet storage demands, bundled with its high degree of scaling, resilience to both switch and node failures, and moderate infrastructure cost make it a viable solution for future massive storage cluster needs

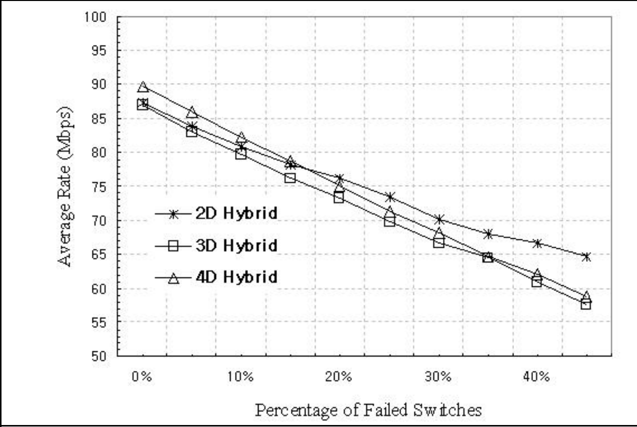

Figure 9. Average rate in the presence of switch failures.

References

[1] CERN,European Organization for Nuclear Research, http://lhc.web.cern.ch/lhc/

[2] Cluster File Systems, "Lustre: A Scalable, High-Performance File System."

[3] A. A. Douglas, B. Jonathan and E. Robert, "98¢/Mflops/s ultra-large-scale neural-network training on a pIII cluster", *Proceedings of Supercomputing 2000*.

[4] A. Hospodor and E. L. Miller, "Interconnection Architectures for Petabyte-scale High-performance Storage Systems", *Proceedings Mass Storage Systems and Technologies (MSST),* April 2004.

[5] Panasas, http://www.panasas.com

[6] Red Hat, "Red Hat Global File System."

[7] R. Renato John, "Server I/O networks past, present, and future", *Proceedings of the ACM SIGCOMM workshop on Network-I/O convergence: experience, lessons, implications*, Karlsruhe, Germany, 2003.

[8] G. Sanjay, G. Howard and L. Shun-Tak, "The Google file system", *Proceedings of the Symposium on Operating Systems Principles*, 2003.

[9] H. Thomas, I. M. Timothy, P. L. Raymond, G. D. Henry and P. G. Huang, "High-cost CFD on a low-cost cluster", *Proceedings of Supercomputing 2000.*

[10] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long and C. Maltzahn, "Ceph: A Scalable, High-Performance Distributed File System", *Proceedings of Operating Systems Design and Implementation (OSDI06)*, Seattle, WA, November 2006.

[11] Q. Xin, E. L. Miller, T. J. E. Schwarz and D. D. E. Long, "Impact of Failure on Interconnection Networks for Large Storage Systems", *Proceedings of Mass Storage Systems and Technologies (MSST 2005)*, Monterey, CA, April 2005.