Distributed Object Storage Rebuild Analysis via Simulation with GOBS

Justin M. Wozniak
MCS Division
Argonne National Laboratory
Argonne, IL, USA
wozniak@mcs.anl.gov

Seung Woo Son
MCS Division
Argonne National Laboratory
Argonne, IL, USA
sson@mcs.anl.gov

Robert Ross
MCS Division
Argonne National Laboratory
Argonne, IL, USA
rross@mcs.anl.gov

Abstract

Community acceptance of the object storage device model as represented by standards and use in existing HPC filesystems has enabled the development of more complex data storage systems. Object replicas may be placed in a variety of ways to obtain various properties, such as scalable lookup times, concurrent access to multiple objects, and efficient reorganization. The construction of a fully functional object-based parallel filesystem is an enormous effort, so evaluation of potential techniques and algorithms is typically performed by analysis or simulation. In this work, we present an extensible simulator designed to evaluate multiple object placement models under fault-induced rebuilds. We use results obtained by the simulator to weigh the benefits of simple object replica placement models.

1 Introduction

Object-based storage systems form the basis of many parallel and distributed filesystems in use and under development today. A growing variety of schemes has been proposed to place and locate objects on large storage systems of up to 1,000 object servers. Beyond the challenges inherent in the storage of ever-growing quantities of data, storage systems designed for use by massively parallel filesystems pose particular challenges for object storage placement algorithms. Object placement algorithms approach the problem of storage management as framed by multiple requirements. The design evaluation of an object placement scheme typically includes the lookup complexity, flexibility, and redundancy provided.

High-performance storage systems in the 2015-2018 timescale are expected to contain over 1,000 servers and tens of thousands of disks, serving an exabyte of data or more [8]. In practice, such a filesystem will nearly always be in a state of rebuild because of a partial failure such as the loss of a component hard disk. Consider an example 960 PB storage system consisting of 32,000 disk drives, each at 30 TB. Using a responsible estimate of 10% disk mortality per year [9], the system would unexpectedly retire an average of 8.76 disks per day, containing 263 TB of redundant data. Restoring the nominal level of redundancy would require a perpetual network commitment of 3.125 GB/s. Here, we use coarse-grained simulation to investigate the impact of this quantity of rebuild work on the overall filesystem.

Traditional rebuild methods wait for the administrator to insert a replacement disk, then restore the nominal redundancy level by moving data to the new disk. This delay in returning to the steady state expands the *window of vulnerability* [1] in which data loss may occur as a result of subsequent faults. The window can be shortened by improving rebuild times through the application of an aggressive fault response in which disk space is dynamically allocated for replacement object replicas and *objects are concurrently transferred* to multiple new locations. To investigate this behavior, we developed the General OBject Space (GOBS) simulator. As described below, this simulator allows the rapid evaluation of various combinations of algorithms and techniques used by parallel filesystems, including file/object generation, redundancy schemes, object placement, rebuild mechanisms, and workload impact.

The remainder of this document is organized as follows. In the next section, we note existing object replica placement schemes and filesystem simulators. In Section 3, we describe the aspects of the simulated system we intend to model and measure and describe how this is carried out with the simulator. In Section 4, we investigate basic behavior of replica placement schemes and measure the data loss characteristics. In Section 5 we summarize our contributions.

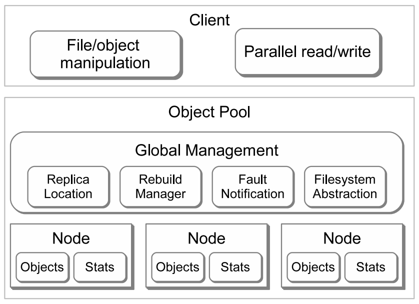

Figure 1. Simulated object storage cluster.

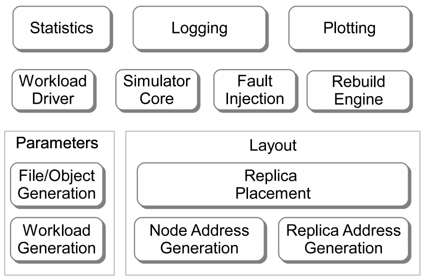

Figure 2. GOBS simulator components.

2 Related Work

Hashing has been widely used for data placement as it eliminates the cost of maintaining global maps for locating data items, including replicas. Distributed hash tables (DHTs) provide an interesting background for object placement algorithms. While most of the schemes considered here are literally DHTs, the term typically implies systems designed for Internet-based storage systems, with correspondingly large scale and low reliability of individual components. Notable file systems built on DHTs including PAST [10] and FARSITE [7].

Another commonly used replica placement scheme is *chain* data placement [3, 5, 10], which first chooses a primary node through any data placement scheme and then places replicas on a server adjacent to the primary, that is, an object O with identifier x is placed on the k servers with identifiers closest to x; k is commonly 2 or 3. The chain placement scheme is used not only in peer-to-peer systems, such as PAST and CFS, but also multiple distributed filesystems.

While chaining places replicas in a correlated manner, it can suffer from load imbalance and poor failure recovery time. Recently proposed distributed storage systems, such as Ceph [11] instead use a pseudo-random replica placement algorithm. Ceph places objects using the underlying Reliable, Autonomic Distributed Object Store (RADOS) [12], which replicates objects using the Replication Under Scalable Hashing (RUSH) [4] algorithm. Kinesis [6] achieves balanced utilization of storage and network resources using three design features: partition of servers into k disjoint segments; freedom of choice to allocate a server to store and retrieve data based on current system availability; and independent, pseudo-random spread of replicas in the system.

3 Object Placement Simulation

The GOBS simulator allows for the investigation of storage performance and reliability characteristics, as described in the Introduction. In particular, it models the behavior of the storage network at a high level and is generally concerned with the emergent characteristics of object placement and movement for large numbers of large objects in a cluster of on the order of 1,000 storage nodes.

The simulated system is represented in Figure 1. At the top are client operations such as the insertion and location of objects, as well as simulated read/write operations, since the object of our investigation is ultimately on user experience with the simulated system. Next, the global management infrastructure is represented, including the object placement mechanism, rebuild management, and metadata such as filesystem abstractions. The GOBS simulator does not model control operations or metadata management explicitly. At the base, the collection of storage nodes is modeled.

Other simulator components are represented in Figure 2. At the top level are the desired outputs, including functional statistics, logs, and graphical plot output. The central mechanisms of the simulator include the Workload Driver, which models user operations, the Rebuild Engine, which models fault-triggered operations, and the fault model based on mean-time-to-failure estimates (MTTF) as modeled by an exponential distribution. At the bottom are the extensible components, including the file/object generation mechanism, and the replica placement schemes. Use cases may be generated using simple techniques or by interpolating complex distributions such as those given by published studies.

This paper focuses on the effects of modifying the replica location algorithm on the performance of rebuild operations after a fault. Rebuilds are simulated in a coarse-grained manner: data tranfer times are approximated by dividing the total data size by the data rate achievable on the storage hardware in use; computation, metadata and control operations are not considered. Upon replica loss, the required

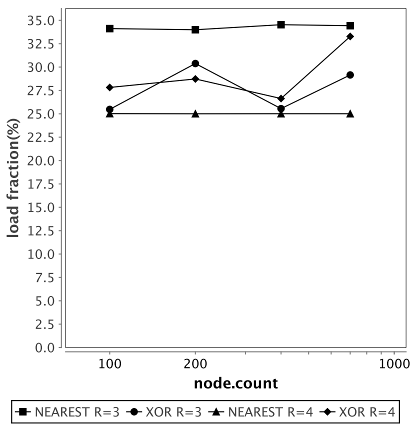

Figure 3. Rebuild load maximum when replica is pulled from primary node.

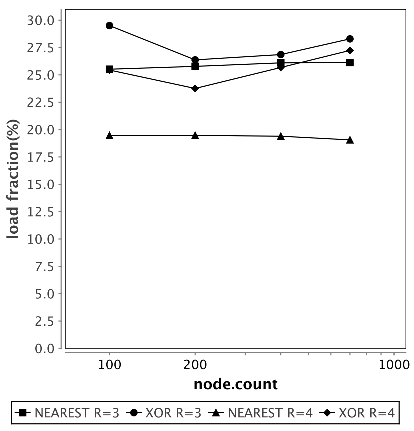

Figure 4. Rebuild load maximum when replica is pulled from random node.

object copy operations are tabulated and randomly scheduled. Copy operations are issued immediately as storage devices become available to perform them, and no device is involved in more than one transfer at a time.

4 Simulator Results

In this section, we demonstrate the utility of the simulator as applied to the replica chaining method. First, we use the simulator to perform rebuilds after a storage fault and identify the concurrency available under algorithmic variations. Next, we run the simulator over a long timescale and produce overall rebuild traffic patterns and the typical traffic pattern local to a rebuild, again under algorithmic variations. Then, we show that the simulator can estimate user data object loss rates, given a per-disk MTTF estimate and

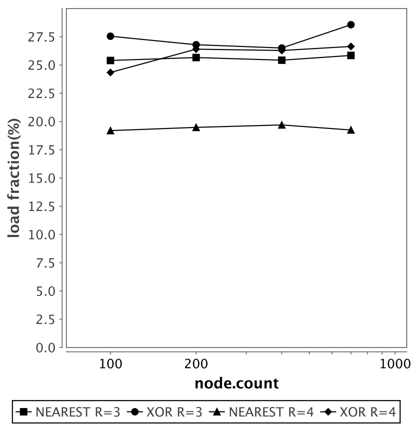

Figure 5. Rebuild load maximum when replica is pulled from last node in replica chain.

a placement algorithm.

4.1 System Model

The system modeled in this study is designed as follows. The basic storage element is the storage server node, which has an address in the object address space. Each node contains multiple local RAID arrays of 30 TB disks. The unit of failure is a whole RAID array; individual disk failures and the performance impact of the resulting local RAID rebuild is not modeled. RAID arrays fail in accordance with the basic reliability formula [2]. Disks read and write data at 400 MB/s, the device transfer rate is the disk rate multiplied by the performance boost offered by RAID. The whole system stores approximately 1 EB (exabyte) of user data objects.

The placement algorithm considered here is a chain placement algorithm. The `NEAREST` algorithm places R replicas of object x on the nodes s_r such that $|x - s_r|$ is minimized. The `XOR` algorithm places R replicas such that x `xor` s_r is minimized [7]. Upon the loss of a RAID array, each node applies the placement algorithm to determine which objects that it holds must be copied to restore the nominal redundancy level. Replicas are then copied in continuous time; a given RAID array is involved in only one copy at a time. The single closest server to x is the *primary* node for x, the other replicas are *secondaries*.

4.2 Rebuild Hot Spots

Figures 3, 4, and 5 diagram the bottleneck in rebuild concurrency under both algorithms at redundancy levels $R = 3$ and $R = 4$. In these cases, the simple $\sim$1 EB filesystem is deployed onto a simulated storage network containing `node.count` nodes, and the loss of a single RAID array is simulated. The rebuild response is initiated immediately upon the (instantaneous) global detection of the fault; the system does not wait for the insertion of a new disk.

The loss of a local RAID array causes the loss of the local replica of each object stored on that array. Thus, data movement is triggered on server nodes that hold the remaining replicas for these replicas. If an additional RAID array is available on the same node, object data will be copied from a neighbor node to the remaining array. If no arrays are operational, the objects are and copied to other available server nodes. For each node count on the x-axis, the fraction of the rebuild workload (reads and writes) performed by the maximally loaded server node is reported, averaged over 10 runs.

In Figure 3, the replica is always copied from the primary node, simulating a case in which the system prefers to control the consistency of the contents of each object at the primary. The case in which a replica may be copied from any secondary node is considered in Figure 4, and the case in which the last replica in the chain must be the replica source is covered in Figure 5. Note that in these cases the local storage layout and object count does not affect the result.

Each node participating in the rebuild performs some read and write operations to move data to the node that lost the replicas. For each node count on the x-axis, the fraction of the rebuild workload (reads and writes) performed by the maximally loaded server node is reported, averaged over 10 runs.

Although each case shows considerable variation because of the varying circumstances of the rebuild, available rebuild concurrency in the chaining algorithm is limited by the number of replicas. The objects involved are tightly clustered in the address space, constraining the rebuild process to a small number of nodes.

4.3 Rebuild Distributions

Figures 6, 7, and 8 diagram the same basic fault conditions as the previous series of figures but show the load performed by each node involved in the rebuild. Only the system with 600 server nodes is considered. The nodes are ordered by load level; the load fraction for node 0 is the average workload fraction performed by the most heavily loaded node, node 1 is the next heavily loaded node, etc., until no more work is distributed to server nodes.

This shows that both NEAREST and XOR produce the expected high workload concentration near the site of the data loss, and the workload trails off for nodes farther from the fault site. The slope of the workload curves indicates that some work may be distributed but that the central nodes (node 0, etc.) will act as a bottleneck and cause a “long tail” effect. Thus, the redundancy level will be restored piece-

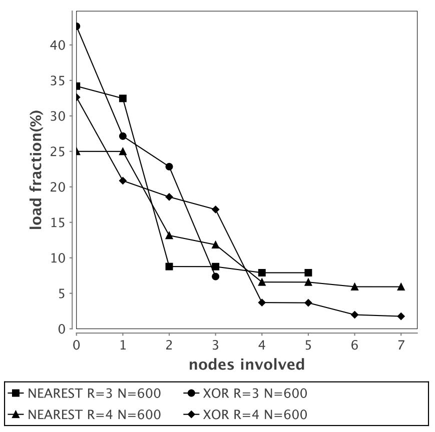

Figure 6. Rebuild load maximum when replica is pulled from primary node.

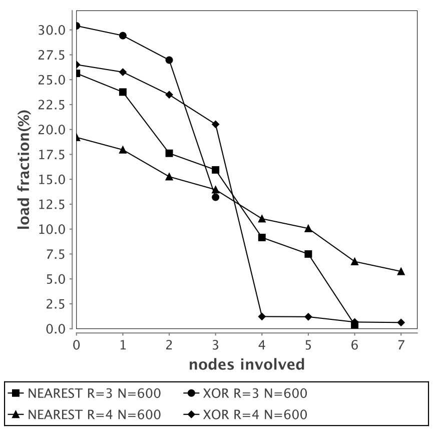

Figure 7. Rebuild load maximum when replica is pulled from random node.

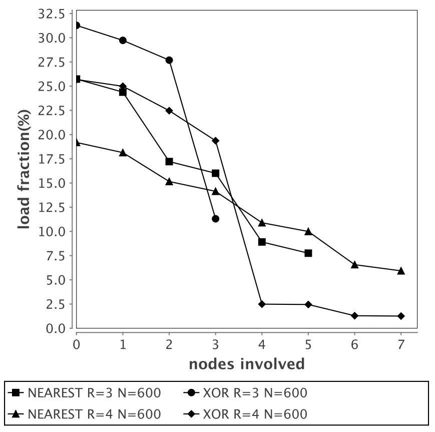

Figure 8. Rebuild load maximum when replica is pulled from last node in replica chain.

meal over time. The impact of this differentiated concurrency is considered in the following experiments. Additionally, as shown, the use of additional replicas increases the ability of the system to distribute work to more nodes.

4.4 Rebuild Traffic

We now investigate the impact of the non-trivial rebuild completion behavior on long timescale simulations in the presence of potentially overlapping storage faults. As shown in Figure 9, the system of interest to this investigation is regularly in a state of replica repair. This figure diagrams a 1 EB storage system serving data from nodes with 4 RAID arrays, each configured as RAID-5 (4+1). The individual disk MTTF was 10 years. The mean time to disk reinsertion was 1 day.

In Figure 10, we show average available rebuild concurrency in the hours following the fault for this system. The two configurations include a SAN-like system (“san”) with 8 RAID arrays per node configured as RAID-6 (8+2) with 2 replicas per object ($R = 2$) and a cluster-like system (“target”) with 4 RAID arrays per node configured as RAID-5 (4+1). We ran the system in “active” response mode (distributed sparing), in which the system immediately started making copies in response to a fault, and “latent” response mode, in which the system waited for the faulty disk to be replaced before scheduling replica copies.

As shown, the active mode makes many additional copies compared to the latent mode. This includes copies that are made in direct response to the fault in addition to copies to the new empty disk, as well as intermediate copies that may be made as a result of subsequent faults. The active mode is shown to be capable of quickly making use of high object transfer concurrency to reduce the window of vulnerability at the cost of greater network and disk load.

Additionally, there is a notable correspondence between the workload distribution and the ability of the system to sustain concurrent data transfers over time. In the chaining scheme used here, the nodes that are peripherally involved in the rebuild run out of work to do, so the concurrency level decreases over time. In a system with many consecutive, overlapping disk failures, this behavior could have a significant effect on data loss.

4.5 Potential for Data Loss

In this final case, we look at the ability of the simulator to estimate the number of user data objects lost per year given an algorithm and a per-disk MTTF. In this test, we inserted the user objects and applied the methods from the previous subsection, varying the per-disk MTTF over a range of unrealistically low MTTF values. The quantity of data loss was measured on a per-object basis for the one year runs.

As shown in Figure 11, data loss is unlikely unless the per-disk MTTF is set to an extremely low value of less than one year. Somewhat surprisingly, the “active” method loses more objects than the “latent” method. It is difficult to generalize about such rare cases even with the traces from the simulator but it may be the case that the extra work caused by the “active” method has the potential to overload the servers at certain critical times.

5 Summary

As object placement routines become more complex and storage systems add ever more component devices, the need to analyze the behavior of the overall system becomes more pronounced. To address this need, we have presented the design of a coarse-grained simulator to quickly evaluate the ability of an algorithm and its variations on a simulated

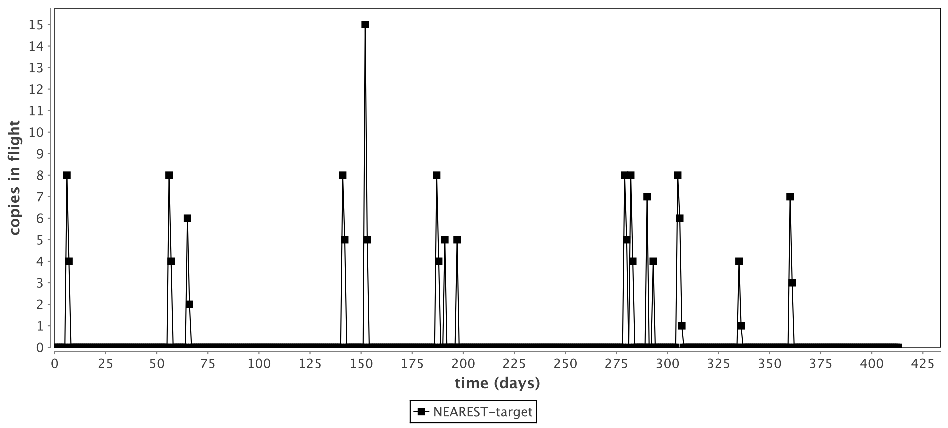

Figure 9. Long timescale rebuild traffic report.

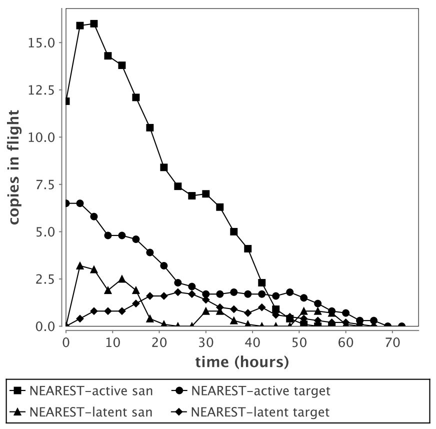

Figure 10. Average rebuild concurrency over time.

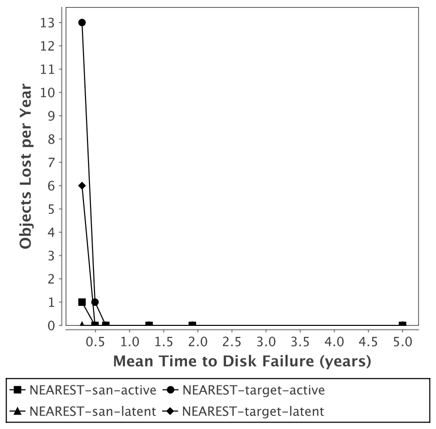

Figure 11. Data loss rate for varying (extremely low) per-disk MTTFs.

large scale filesystem. w demonstrate the simulator's ability to extract meaningful results including limits to concurrency during rebuilds (§4.2), work distribution during rebuilds (§4.3), rebuild-centered traffic patterns (§4.4), and the data loss rate (§4.5).

The utility of the simulator and its results stand to be improved. First, the network model could be improved by making use of a congestion model, preferably above the packet level. Second, it could be integrated with a more complex local storage model that provides useful, coarse-grained performance approximations for disks, RAID devices, and the local object storage service. Third, additional implementations of other well-known placement algorithms should be produced so that the community can run the various algorithms with modifications on simulated systems. The software will be released under an open source license.

6 Acknowledgments

This research is supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy under Contracts DE-AC02-06CH11357. Work is also supported by DOE with agreement number DE-FC02-06ER25777.

References

[1] M. Baker, M. Shah, D. S. H. Rosenthal, M. Roussopoulos, P. Maniatis, T. J. Giuli, and P. Bungale. A fresh look at the reliability of long term digital storage. In *EuroSys*, 2006.

[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), 1994.

[3] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proc. Symposium on Operating Systems Principles*, 2001.

[4] R. J. Honicky and E. L. Miller. A fast algorithm for online placement and reorganization of replicated data. In *Proc. International Parallel and Distributed Processing Symposium*, 2004.

[5] H.-I. Hsiao and D. J. DeWitt. Chained declustering: A new availability strategy for multiprocessor database machines. In *Proc. International Conference on Data Engineering*, 1990.

[6] J. MacCormick, N. Murphy, V. Ramasubramanian, U. Wieder, J. Yang, and L. Zhou. Kinesis: A new approach to replica placement in distributed storage systems. *ACM Transactions on Storage*, 4(4), 2009.

[7] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proc. Workshop on Peer-to-peer Systems*, 2002.

[8] Peter Kogge et. al. Exascale computing study: Technology challenges in achieving exascale systems. DARPA Information Processing Techniques Office, 2008.

[9] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In *Proc. USENIX Conference on File and Storage Technologies*, 2007.

[10] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. *SIGOPS Operating System Review*, 35(5), 2001.

[11] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn. Ceph: A scalable, high-performance distributed file system. In *Proc. Operating Systems Design and Implementation*, 2006.

[12] S. A. Weil, A. W. Leung, S. A. Brandt, and C. Maltzahn. RADOS: A scalable, reliable storage service for petabyte-scale storage clusters. In *Proc. Petascale Data Storage Workshop*, 2007.