RobuSTore: Robust Performance for Distributed Storage Systems

Huaxia Xia and Andrew A. Chien
University of California, San Diego
{hxia, achien}@ucsd.edu

Abstract

Emerging large-scale scientific applications involve massive, distributed, shared data collections in petabytes, and require robust, high performance for read-dominated workloads. Achieving robust performance, i.e. low variability, in storage systems is difficult. We propose RobuSTore, a novel storage technique, which combines erasure codes and speculative access to reduce performance variability and increase performance. RobuSTore uses erasure codes to add flexible redundancy then spreads the encoded data across a large number of disks. Speculative access to the redundant data from multiple disks enables application requests to be satisfied with only early-arriving blocks, reducing performance dependence on the behavior of individual disks.

We present the design and an evaluation of RobuSTore which shows improved robustness, reducing the standard deviation of access latencies by as much as 5-fold vs. traditional RAID. In addition, RobuSTore improves access bandwidth by as much as 15-fold. RobuSTore secures these benefits at the cost of a 2-3x storage capacity overhead and ~1.5x network and disk I/O overhead.

1. Introduction

Existing and emerging large-scale scientific applications and data-intensive applications have new requirements on storage systems. These applications involve read-dominated accesses to massive, distributed data collections in petabytes, and share the data among thousands of distributed users. The applications require not only high performance but also robust performance from distributed storage systems. Here, robustness means low-variability in access latency.

In such distributed shared environment, disks have highly heterogeneous and variable performance, which prevents conventional parallel accesses from achieving robust, high performance. Sources of performance variation include heterogeneous disk types, irregular on-disk data layout, and differences in disk load. The combination of the factors leads to 100s of times difference on individual disk performance. Traditional parallel filesystems use data striping and parallel access to aggregate disk performance. The access completion requires a copy of all the stripes, so the performance is limited by the slowest disk; even if data are replicated, late arriving disk responses can still delay completion of the larger request, as in Figure 1.

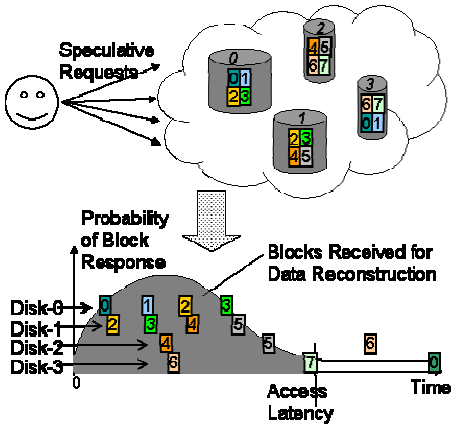

Figure 1.Aggregating Disks with Conventional Parallel Scheme: 8 blocks, 2 replicas. Disk performance is varied.

We present new storage architecture RobuSTore, which combines erasure coding and speculative accesses together. RobuSTore uses erasure codes to add continuous redundancy for striping; with such layouts, clients can use speculative parallel access and decoding of the fast-returning blocks to both increase performance, and reduce performance dependence on stragglers (lower variability). As a result, RobuSTore can efficiently aggregate large number of distributed storage devices to deliver robust, high access performance.

The RobuSTore approach is feasible with the following observations. First, the application workloads are read-dominated. Further, new technologies have been improving network bandwidth, CPU speed, and disk capacity rapidly. For example, the low-cost optical transmission and Dense Wavelength Division Multiplexing (DWDM) technique enables individual fibers to carry 100's of 10 Gbps "lambdas", providing wide-area networks with private 10Gbps or even 40Gbps connections [1, 2]. CPU speed doubles every 18 months and disk capacity doubles every 12 months [3].

The remainder of the paper is organized as follows. In Section 2, we present the RobuSTore approach and describe the RobuSTore design, and it is the simulation-based evaluation in Section 3. Section 4 surveys related work and Section 5 summarizes the work.

* Supported in part by the National Science Foundation -- Cooperative Agreement ANI-0225642(OptIPuter), CCR-0331645(VGrADS), ACI-0305390, and Research Infrastructure Grant EIA-0303622. Support from the UCSD Center for Networked Systems, BigBangwidth, and Fujitsu is also gratefully acknowledged.
The authors thank Justin Burke and Frank Uyeda for their comments.

[1] An extended version of this paper is available as UCSD Tech Report CS2006-0851.

2. Robust, High Performance Distributed Storage

We describe the RobuSTore approach which meets our three goals: robust access latency, high bandwidth, and small overhead on access large data segment.

2.1. RobuSTore Idea

The key idea in RobuSTore is the combination of erasure codes and speculative access. We use erasure codes to add systematic data redundancy and then exploit speculative access to statistically aggregate the disk access bandwidths, producing an earlier and lower-variance in access latency for large requests.

On write, the client splits a large file or a large data segment into N data blocks (N=100s~1000s), encodes them into M coded blocks (M>N) using erasure codes, and distributes the coded blocks to tens or hundreds of disks. Erasure coding allows the data reconstruction from almost any N(1+δ) coded blocks, where δ is zero or a small number decided by the choice of coding algorithms, and (1+δ) is called *reception overhead*.

A read client speculatively requests redundant coded blocks in parallel. As soon as the client receives enough number of blocks, it can complete the reconstruction of the original data, without waiting for later blocks. This improves the access bandwidth, as shown in Figure 2. In practice, the client may cancel requests for the unreceived blocks to save disk IO and network bandwidth.

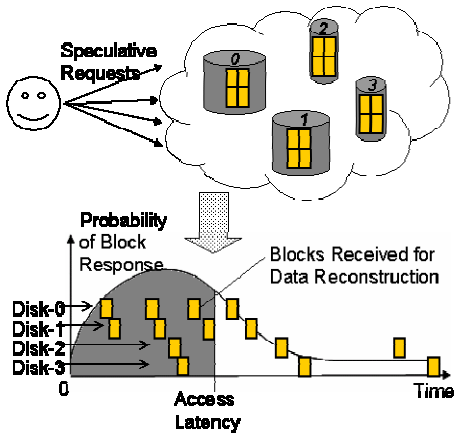

Figure 2.RobuSTore Data Access. Reconstruction with first arriving coded blocks. Disk performance is varied.

Furthermore, if any of these early-arrived blocks gets delayed or lost, the client can utilize the next arrived block to complete the access. So an individual disk or block has little impact on the overall access performance, which improves the performance robustness.

2.2. RobuSTore Design

To realize the RobuSTore idea, we design a flexible RobuSTore system framework, enabling broad exploration of the design space. We describe its components and possible design choices.

RobuSTore storage system architecture includes three key components: client, metadata server, and storage servers (see Figure 3).

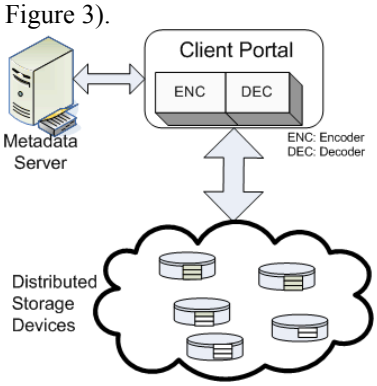

Figure 3.RobuSTore System Architecture

Client Portal performs many distributed filesystem functions: accessing metadata, planning layout, encoding/decoding data, and communicating with storage servers. The client includes erasure coding modules to implement the encoding and decoding. If desired, a separate machine/cluster can be used for coding to save client CPU or improve coding speed. Coding is transparent to the storage servers.

While there are many different erasure codes, RobuSTore uses modified Luby Transform (LT) codes [4, 5]. The LT codes are part of a general class of low-density parity check codes and use block-XOR operations based on sparse bipartite graphs. LT codes have a number of advantages. First, they are "rateless", allowing redundancy to be decoupled from other system design issues, such as the number of storage servers used. Second, LT codes use irregular bipartite graphs and block-XOR operations, enabling fast encoding and decoding throughputs. Third, LT codes allow decoding to be overlapped with receiving data from storage servers, masking decoding time. We adapted LT codes by generating coding graphs which guarantee decodability [5], and built a fast implementation which delivered decoding bandwidth up to 320 MByte/s on a Intel 2.4Ghz Xeon CPU. Our experiments show that the memory bandwidth is the bottleneck to coding performance. Our work suggests that multi-processors of the future or processors with higher memory bandwidth (such as AMD Opteron or AMD Athlon 64) will surpass a coding bandwidth of 1GByte/s.

Metadata Server maintains both file information (size, organization, and location) and storage server information (available space, performance, connectivity, current load, etc). A simple central metadata server minimizes update and synchronization costs at some penalty in scalability. Reliability and scalability can be addressed by well-known replication and failover techniques.

Storage Servers provide data storage at block level (erasure-coded block, presumed to be larger than disk blocks). Servers may be single disks or arrays, and each implements local admission and access control. Servers

typically have variable performance due to heterogeneity in hardware, data layout, or load.

Access Control: There are many good candidates for the distributed filesystem security, for example, credential-based access control is flexible in the environments with a large number of servers and users.

3. Evaluation

We use a detailed discrete-event simulation to evaluate the RobuSTore idea across the configuration space, varying access strategy, the number of storage servers, data size, block size, network latency, and degree of redundancy.

3.1. Methodology

We compare RobuSTore and conventional approaches with different strategies for data layout, redundancy, and access.

Data Layout and Redundancy: Possible data layout methods include: (1) split the data into blocks, and distribute them to many disks; (2) split the data and distribute the blocks with replication; (3) split the data, encode the blocks, and distribute these redundant coded blocks to many disks. They are depicted in Figure 4.

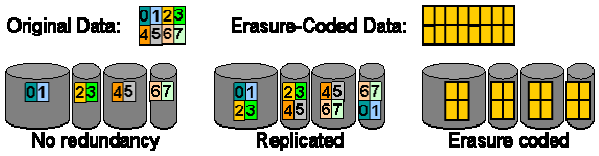

Figure 4.Data Layouts. 8 original blocks; 2x data redundancy in replicated and coded layouts

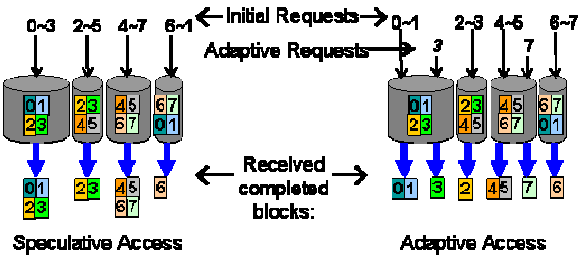

Figure 5.Access Strategies. Disk performance is varied.

Access Strategies: Possible data access strategies are: (1) speculative access, i.e., request redundant blocks at once in the beginning of the access and cancel the requests once enough blocks have been received; (2) adaptive access, in which the client dynamically requests the unreceived bytes. Examples are shown in Figure 5.

We evaluate the following four combined schemes:

- **RAID-0:** No data redundancy + speculative access
- **RRAID-S:** Replication + speculative access
- **RRAID-A:** Replication + adaptive access
- **RobuSTore:** Erasure coding + speculative access

3.1.1. Workload and Performance Metrics

Workload: Since our focus is on supporting the needs of applications with large-read dominated workloads, we study access performance for single 128MB, 256MB, 512MB, and 1GB reads. Data objects larger than 1GB are presumed to be accessed by multiple 1GB reads.

Performance Metrics: While robustness is the major goal of RobuSTore, we must also maintain high bandwidth with small overhead. So we formalize them into the following three metrics:

- *Standard Deviation of Access Latency:* the standard deviation over a set of one hundred accesses. Smaller standard deviations correspond to higher robustness.
- *Access Bandwidth*
- *Disk IO Size/Data Size:* This is the cost from speculative access.

3.1.2. Simulation Method

We simulate an environment with one client and 128 storage servers connected by wide-area networks.

Each storage server is simulated using one DiskSim [6] process. DiskSim processes are configured to have different disk-level block layouts such that individual disk performance varies from 0.52MBps to 53MBps. This represents the performance variability in shared distributed storage environments with many sources of variability, as discussed in Section 1.

The virtual client models all other overheads for metadata access, server connection, network latency, and block decoding. The metadata access and server connection are assumed to take constant time; network latency to each server is configured constant from 1ms to 100ms. For block decoding, since it can be pipelined with data receiving, extra latency is only incurred for decoding the last block; we model it as a constant 5 ms overhead. We assume sufficient network bandwidth and CPU power.

In the experiments, we study the four storage schemes (RAID-0, RRAID-S, RRAID-A, and RobuSTore) along five system parameters: number of disks, data size, block size, network latency, and degree of redundancy. In each experiment, we vary only one parameter, and compare to a fixed baseline. The baseline is a typical SAN configuration: to access 1GB data from 64 disks, 1ms network round-trip time (RTT), 1MB block size and 4x data redundancy, except for RAID-0 which always has 1x data redundancy.

3.2. Results

We simulate the four storage schemes over five configuration dimensions. Due to space limitation, we only show part of the results in the paper; more results are available in [7].

3.2.1. Robustness Improvement

Robustness vs. Number of Disks

When we vary the number of disks from 2 to 128, the standard deviation of latency changes as depicted in Figure 6. RAID-0 suffers because it exploits no redundancy, and the performance is thus subject to the slowest disk. RRAID-S explores the replicated data to hide the slow disks; however, it reads the blocks on same

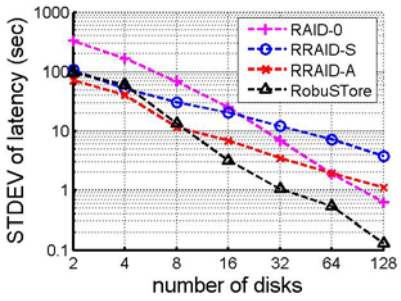

Figure 6.Robustness vs. Number of Disks.

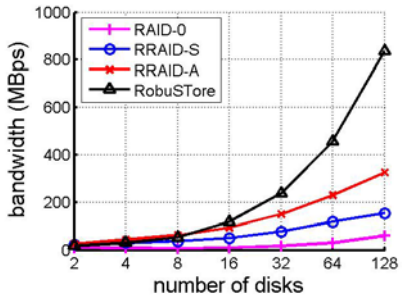

Figure 7.Bandwidth vs. Number of Disks.

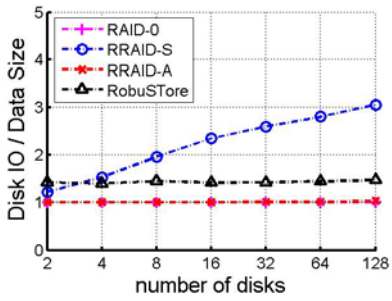

Figure 8.Overhead vs. Number of Disks.

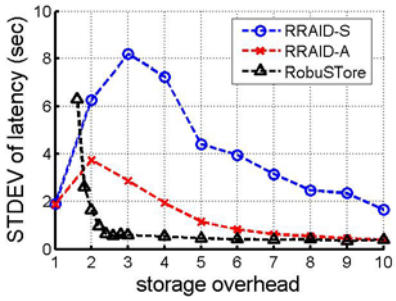

Figure 9.Robustness vs. Data Redundancy.

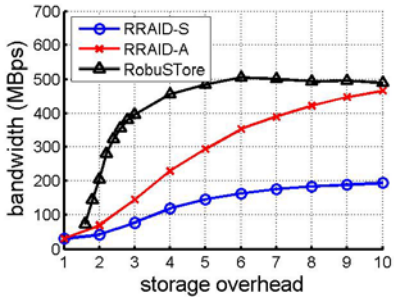

Figure 10.Bandwidth vs. Data Redundancy.

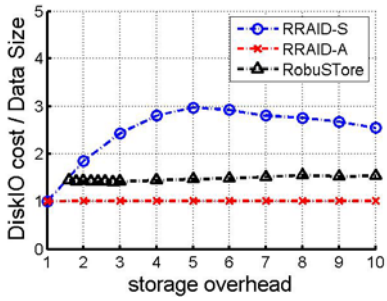

Figure 11.Overhead vs. Data Redundancy.

disk in fixed order, so its performance depends on both intra-disk block ordering and inter-disk block mapping. RRAID-S has the highest variability due to the combination of these factors. RRAID-A mitigates the dependency on intra-disk block ordering by accessing blocks selectively, but still depends on inter-disk block mapping. When using small number of disks (<8), RRAID-S and RRAID-A essentially perform whole-file replication, so they suffer a low-level of inter-disk dependence and have comparable robustness to RobuSTore.

Performance variability in RobuSTore is from the total bandwidth of all the disks and is not related to intra-disk ordering or inter-disk mapping at all. RobuSTore has the lowest performance variability for systems with more than a few disks (>8). The standard deviation of access latency on 64 disks for RAID-0, RRAID-S, RRAID-A and RobuSTore are 1.9, 7.3, 1.9, and 0.5 seconds respectively; and they are 0.63, 3.8, 1.1, and 0.13 seconds on 128 disks. RobuSTore improves robustness for up to 5x compared to RAID-0, and more than 15x compared to RRAID-S.

Robustness vs. Redundancy

We vary storage overhead from 1x to 10x (900% overhead) to quantify its impact (see Figure 9). RAID-0 is represented by the 1x point on the RRAID-S curve.

In RRAID-S and RRAID-A, the variability comes from disk speed, intra-disk block ordering (in RRAID-S), and inter-disk block mapping. Higher data redundancy reduces impact from inter-disk block mapping, while increases impact from intra-disk block ordering. RAID-0 only suffers variability from the slowest disk. Due to the combination of these factors, RRAID-S and RRAID-A have worse robustness than RAID-0 when redundancy is small, and gradually get better as redundancy increases.

RobuSTore achieves the best performance robustness, and needs only 2-3x data redundancy to obtain most of this benefit. When using more than 4x storage space, the standard deviation of latency is only ~ 0.5 seconds or about 25% of the average access latency.

3.2.2. Bandwidth Improvement

In most of our experiments, RobuSTore delivers significantly better performance than other schemes.

We vary the number of disks and present the bandwidth results in Figure 7. RobuSTore is slightly worse than RRAID-A for small numbers of disks (<8) due to the reception overhead of 1.4x; but it performs the best for large numbers of disks. RobuSTore achieves 15x the bandwidth of RAID-0 for 16-128 disks. The access bandwidth to 1GB on 64 disks is as follows: 31 MBps for RAID-0, 117 MBps for RRAID-S, 228 MBps for RRAID-A, and 459 MBps for RobuSTore.

Figure 10 shows that increasing data redundancy increases the bandwidth of RobuSTore rapidly. The bandwidth of RRAID-S and RRAID-A both benefit less than RobuSTore as data redundancy is increased. This is because their structured redundancy (replication) cannot adapt to reading more blocks from the faster disks as flexibly as in RobuSTore.

3.2.3. Access Overhead

The benefits of aggressive access to redundant copies can yield performance benefits, but it also increases network and disk I/O costs. Figure 8 and Figure 11 depict the disk overhead for different schemes. RAID-0 incurs

no additional costs, and achieves a ratio of 1. RRAID-A is just a little bit more than 1. RRAID-S generates a large number of speculative requests, reaching overhead ratios as high as 3x. RobuSTore has cost about 1.4 due to the requirement of extra blocks for decoding.

4. Related Work

There has been a wealth of work on distributed storage and performance aggregation of multiple disks.

Parallel file systems ([8-10], etc) aggregate multiple disks, addressing the performance and capacity limitation of single disks or servers. They assume uniform arrays of storage devices in a SAN or LAN environment. However, these systems do not tolerate dynamic performance variability in a fashion comparable to RobuSTore.

Some peer-to-peer file sharing systems ([11, 12], etc) improve access performance by speculatively fetching from massively replicated data copies. However, the massive replication is expensive in terms of storage overhead and access scheduling. Further, these systems focus on the shared internet where per-node network bandwidth is as low as 1-10 megabits/s.

Numerous storage systems exploited erasure codes for data reliability and availability ([13-16], etc.), but not for robustness or bandwidth.

Collins and Plank's work [17] studied the usage of Reed-Solomon Codes and LDPC Codes to improve bandwidth of wide-area storage systems. However, they only focused on access bandwidth, with no study on performance robustness. Furthermore, they assume slow shared networks, bandwidths < 10MByte/s, and small number of blocks ($N \leq 100$), and they concludes that Reed-Solomon Codes perform better than or equal to LDPC codes. In contrast, we focus on performance robustness as well as bandwidth; we design RobuSTore for high bandwidth wide-area networks (>10Gbps), and explore a much wider array of design choices in data coding parameters, redundancy, layout and access.

5. Summary and Future Work

We have proposed RobuSTore, a system combining erasure coding and speculative accesses to both improve performance robustness and absolute performance. We present a system framework which explains the major architecture and function required to make it workable.

Using discrete-event simulation, we compare the performance of RobuSTore with three traditional parallel schemes across a wide range of system configurations. These simulation results show that RobuSTore delivers the best robustness, up to 5x compared to a baseline RAID-0 scheme. At the same time, RobuSTore also provides highest access bandwidth of nearly 15x from RAID-0. RobuSTore incurs only moderate I/O costs of about 1.4x and storage overheads of 2-3x.

We have only taken initial steps in evaluating RobuSTore. Natural extensions include: 1) evaluation with a richer set of workloads, varying access sizes and including writes, 2) empirical studies with a RobuSTore prototype, 3) experiments with real applications and real testbeds, 4) study of different algorithms for encoding, and 5) evaluation for multi-user workloads and shared storage servers.

References

[1] L. L. Smarr, A. A. Chien, T. DeFanti, J. Leigh, and P. M. Papadopoulos, "The OptIPuter," *Communications of the ACM*, vol. 46, pp. 58-67, 2003.

[2] "iGrid2005," http://www.igrid2005.org, San Diego, CA, Sep 26-30, 2005.

[3] E. Grochowski and R. D. Halem, "Technological impact of magnetic hard disk drives on storage systems," *IBM Systems Journal*, vol. 42, pp. 338-346, 2003.

[4] M. Luby, "LT Codes," IEEE 43rd Symp. On Foundations of Computer Science 2002.

[5] F. Uyeda, H. Xia, and A. Chien, "Evaluation of a High Performance Erasure Code Implementation," UCSD, Technical Report CS2004-0798, 2004.

[6] J. S. Bucy and G. R. Ganger, "The DiskSim Simulation Environment Version 3.0 Reference Manual," Carnegie Mellon University CMU-CS-03-102, January 2003.

[7] H. Xia and A. A. Chien, "RobuSTore: Robust Performance for Distributed Storage Systems," UCSD, Technical Report CS2006-0851, 2006.

[8] P. H. Carns, W. B. L. III, R. B. Ross, and R. Thakur, "PVFS: A Parallel File System For Linux Clusters," 4th Annual Linux Showcase and Conference, Atlanta, GA, 2000.

[9] F. Schmuck and R. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," USENIX Conference on File and Storage Technologies (FAST), Monterey, CA, 2002.

[10] C. F. System, "Lustre: A Scalable, High-Performance File System," Lustre File System v1.0 Architecture White Paper from clusterfs.org, 2002.

[11] "Kazaa," http://www.kazaa.com.

[12] "BitTorrent," http://www.bittorrent.com.

[13] J. Kubiatowicz, D. Bindel, and e. al., "OceanStore: An Architecture for Global-Scale Persistent Storage," the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Cambridge, MA, 2000.

[14] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker, "Total recall: System support for automated availability management," the First ACM/Usenix Symposium on Networked Systems Design and Implementation (NSDI), San Francisco, CA, 2004.

[15] H. Weatherspoon and J. D. Kubiatowicz, "Erasure Coding vs. Replication: A Quantitative Comparison," the First International Workshop on Peer-to-Peer Systems (IPTPS), Cambridge, MA, 2002.

[16] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Fault-Scalable Byzantine Fault-Tolerant Services," Symposium on Operating Systems Principles, Brighton, UK, 2005.

[17] R. L. Collins and J. S. Plank, "Assessing the performance of Erasure Codes in the Wide Area," DSN-2005: The International Conference on Dependable Systems and Networks, Yokohama, Japan, 2005.