REMOTE DATA BACKUP SYSTEM FOR DISASTER RECOVERY

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAI'I IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

December 2004

By
Hua Lin

Thesis Committee:
Galen Sasaki, Chairperson
Nancy Reed
Yingfei Dong
ABSTRACT

This thesis presents two improvements in remote backup and disaster recovery systems. One is fast parity update, which is used to improve disk reliability. The method computes parity with less I/O requests when performing backup operations. The other is chained-declustering with distributed parity, which can recover from neighboring failures and improve system reliability. Finally, a remote backup and disaster recovery system is specifically designed to perform backup, retrieval, restore and failure recovery operations. The system is composed of clients, a manager server and a number of disks. They are connected through a Local Area Network (LAN) and a Storage Area Network (SAN). The manager server provides service to clients by storing the backup data from clients onto disk, returning retrieved or restored data to clients and recovering from failures. The LAN is used to transfer control information and the SAN is used to transfer data. Our C++ simulation results showed that the parity was updated correctly and missed data in neighboring failures were recovered correctly. The DiskSim2.0 experimental data proved that fast parity update reduces the number of disk accesses.
# TABLE OF CONTENTS

Abstract........................................................................................................ iii  
List of Tables.................................................................................................. v  
List of Figures............................................................................................... vi  

1. Introduction............................................................................................... 1  

2. Background............................................................................................... 2  
   1. Literature Review.................................................................................. 2  
   2. Data Protection Methods...................................................................... 15  
   3. Related Work......................................................................................... 24  

3. System Design and Implementation....................................................... 29  
   1. Fast Parity Update............................................................................... 30  
   2. Chained-Declustering with Distributed Parity................................. 33  
   3. System Components............................................................................ 41  
   4. Operations............................................................................................ 50  

4. Simulation Results.................................................................................. 52  
   1. Test Results and Discussion of Backup, Retrieval and Restore Operations.... 53  
   2. Test Results and Discussion of Missing Data Recovery and Reconstruction... 54  
   3. Simulation Data and Comparison..................................................... 55  
   4. Simulation Result Discussion............................................................. 58  

5. Future Work............................................................................................. 59  
   1. Traffic Director.................................................................................... 59  
   2. Stripe Subset....................................................................................... 61  
   3. Chained-Declustering with Double Distributed Parity..................... 61  

6. Conclusion................................................................................................ 63  

References.................................................................................................... 64
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Most Frequent Disasters</td>
<td>5</td>
</tr>
<tr>
<td>2.</td>
<td>Number of Reads and Writes vs. Number of Disks</td>
<td>56</td>
</tr>
<tr>
<td>3.</td>
<td>Time vs. I/O Requests</td>
<td>57</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Architecture of RAID</td>
<td>8</td>
</tr>
<tr>
<td>2.</td>
<td>Hierarchy of a Typical Storage Area Network with a RAID System</td>
<td>10</td>
</tr>
<tr>
<td>3.</td>
<td>Three Components of a Disaster Recovery System</td>
<td>13</td>
</tr>
<tr>
<td>4.</td>
<td>Centralized Administration and Management of a Backup System</td>
<td>14</td>
</tr>
<tr>
<td>5.</td>
<td>Striping with Distributed Parity</td>
<td>16</td>
</tr>
<tr>
<td>6.</td>
<td>Mirrored-Striping Strategy for Data Replication</td>
<td>17</td>
</tr>
<tr>
<td>7.</td>
<td>Chained-Declustering Strategy for Data Replication</td>
<td>18</td>
</tr>
<tr>
<td>8.</td>
<td>Chained-Declustering during a Failure</td>
<td>18</td>
</tr>
<tr>
<td>9.</td>
<td>Dynamic Balancing in Chained-Declustering</td>
<td>19</td>
</tr>
<tr>
<td>10.</td>
<td>Chained-Declustering during Multi-Failures</td>
<td>20</td>
</tr>
<tr>
<td>11.</td>
<td>Data Unavailability in Chained-Declustering</td>
<td>21</td>
</tr>
<tr>
<td>12.</td>
<td>Data and Redundancy Organization in RAID Levels 0 through 5</td>
<td>22</td>
</tr>
<tr>
<td>13.</td>
<td>Petal Prototype</td>
<td>25</td>
</tr>
<tr>
<td>14.</td>
<td>Log-Based Striping</td>
<td>26</td>
</tr>
<tr>
<td>15.</td>
<td>TSM Backup and Recovery</td>
<td>27</td>
</tr>
<tr>
<td>16.</td>
<td>VERITAS Disaster Recovery Solution</td>
<td>28</td>
</tr>
<tr>
<td>17.</td>
<td>Parity Update in Striping with Distributed Parity</td>
<td>30</td>
</tr>
<tr>
<td>18.</td>
<td>Fast Parity Computation Procedures</td>
<td>31</td>
</tr>
<tr>
<td>19.</td>
<td>Fast Parity Update</td>
<td>32</td>
</tr>
<tr>
<td>20.</td>
<td>Chained-Declustering with Distributed Parity</td>
<td>35</td>
</tr>
<tr>
<td>21.</td>
<td>Two Neighboring Servers Fail</td>
<td>37</td>
</tr>
<tr>
<td>22.</td>
<td>Recover from the Other Copies</td>
<td>38</td>
</tr>
<tr>
<td>23.</td>
<td>Recover from Parity Reconstruction</td>
<td>39</td>
</tr>
<tr>
<td>24.</td>
<td>Recover the Neighboring Failure</td>
<td>40</td>
</tr>
<tr>
<td>25.</td>
<td>Remote Backup System</td>
<td>41</td>
</tr>
<tr>
<td>26.</td>
<td>Backup Data Flow</td>
<td>44</td>
</tr>
<tr>
<td>27.</td>
<td>Storage Server Components</td>
<td>45</td>
</tr>
<tr>
<td>28.</td>
<td>Storage Area Network</td>
<td>46</td>
</tr>
<tr>
<td>29.</td>
<td>Network Attached Storage</td>
<td>47</td>
</tr>
<tr>
<td>30.</td>
<td>LAN-Free Backup</td>
<td>48</td>
</tr>
<tr>
<td>31.</td>
<td>Storage Area Network: The Network behind Servers</td>
<td>49</td>
</tr>
<tr>
<td>32.</td>
<td>Full Backup and Incremental Backup</td>
<td>51</td>
</tr>
<tr>
<td>33.</td>
<td>Number of Reads vs. Number of Disks</td>
<td>57</td>
</tr>
<tr>
<td>34.</td>
<td>Time vs. I/O Requests</td>
<td>58</td>
</tr>
<tr>
<td>35.</td>
<td>Traffic Directors</td>
<td>60</td>
</tr>
<tr>
<td>36.</td>
<td>RAID 5+1</td>
<td>62</td>
</tr>
<tr>
<td>37.</td>
<td>Chained-Declustering with Double Distributed Parity</td>
<td>63</td>
</tr>
</tbody>
</table>
Remote Data Backup and Disaster Recovery

1. Introduction

The purpose of this project is to design a remote data backup system. Due to natural disasters, accidents or human mistakes, data stored in local computers are very vulnerable. For example, in the terrorist attack on the World Trade Center in New York City on September 11, 2001, all computer systems were destroyed along with their valuable information. Thereby, many businesses were brought to a halt. The great Northeast blackout of August 14, 2003 stopped all airlines and financial institutions because computer data became inaccessible. Major earthquakes and other natural disasters could destroy computer equipment and their data, completely disrupting many businesses. Sometimes a malicious intruder (hacker) could destroy data or computer operators could accidentally delete something important. Thus, it is very important to have protection for data. Ideally, if data in local computers are lost, we can recover it immediately from remote backup servers, which are in a safe place.

We propose a system that is an improvement on the well-known data backup architectures: striping with distributed parity and chained-declustering. Section 2 discusses the background of data backup and disaster recovery system. Section 3 has detail discussion of our improvements. The system was simulated and results are given in Section 4. Future work was briefly discussed in Section 5. Finally, our conclusions are given in Section 6.
2. Background

This section will discuss the background of data backup and disaster recovery industry. Subsection 2.1 is a literature review, Subsection 2.2 is a review of data protection methods and Subsection 2.3 presents the current research systems and current commercial systems.

2.1. Literature Review

The U.S. computer backup industry was built by several small regional backup computer facilities located in the Midwest in the mid-1970's [1]. The industry founder SunGard Recovery Services was established in Philadelphia, Pennsylvania, in 1979. Ten years later, the industry grew to over 100 providers of backup services throughout the U.S. By 1989, the industry generated $240 million in annual subscription fees. The industry has been growing steadily since 1989. Today, the industry is comprised of 31 companies and generates more than $620 million in annual subscription fees. The market is now dominated by 3 companies: Comdisco, IBM, and SunGard. These companies control over 80 percent of the market.

The backup industry is primarily focused on financially-oriented subscribers since they are the largest data centers within the U.S [1]. About half of the backup service providers focus exclusively on financial firms. Approximately 45 percent of all revenues ($279 million) come from financially-oriented firms. More than 65 percent of all recoveries involved financially-oriented firms. Comdisco, IBM, and SunGard have supported over 67 percent of all disaster recoveries. VERITAS is a leader in network backup/restore and Storage Area Network (SAN) management. Most of the Fortune 500 use VERITAS products for data protection and storage management.
SunGard is a global leader in integrated software and processing solutions [2]. They primarily provide high availability and business continuity for financial services. SunGard helps information-dependent enterprises to ensure the continuity of their business. They provide service to more than 20,000 customers in more than 50 countries, including the world's 50 largest financial services companies. They have conducted more than 100,000 simulated continuity tests and have maintained a 100% success rate on over 1,500 actual recoveries.

Comdisco offers end-to-end enterprise continuity services that ensure data availability across all platforms including networks, distributed systems, etc [3]. They successfully supported 46 customers with a total of 91 declarations in response to the terrorist attack on September 11, the most declarations submitted in their history. Comdisco led the industry in moving from the old concept of "disaster recovery" to "business continuity". They have gained over 400 successful recoveries and over 35,000 tests. With more than 100 locations around the world, Comdisco serves more than 4,000 customers in North America, South America, Europe and the Asia/Pacific Rim. Comdisco is the winner of the 1999 and 2000 UK Business Continuity Supplier of the Year award and is the global leader in business continuity.

IBM is the world's largest information technology company with 80 years of leadership [4]. Their Tivoli Storage Manager (TSM) protects data from hardware failures and other errors by storing backup and archive copies of data on offline storage. IBM TSM protects hundreds of computers running a dozen OS ranging from laptops to mainframes and connected together via the network. Storage Manager's centralized web-based management, smart-data-move, store techniques and policy-based automation
altogether to minimize administration costs to both computers and networks. The IBM TotalStorage Open Software Family won the industry’s best management solution at the 10th Annual Network Storage Conference in Monterey, California on March 11, 2004. The award winning solution is comprised of IBM Tivoli Storage Manager, and two key elements of the recently announced IBM TotalStorage Productivity Center, IBM Tivoli SAN Manager and IBM Tivoli Storage Resource Manager.

VERITAS Software was founded in 1989 [5]. VERITAS is a supplier of storage software products and services. VERITAS ranks among the top 10 software companies in the world. It develops and markets high availability products focused on business efficiency and continuity. Its software products protect, archive and recover data, provide application availability and enable recovery from disasters.

The backup industry has successfully recovered 582 companies since 1982 with an average of 40 per year [1]. Almost 44 percent of the recoveries resulted from regional events where multiple subscribers were affected simultaneously. In these cases, no client was denied access to a recovery facility because of excessive demand. The industry has been growing at approximately 17 percent per year in revenue and approximately 30 percent per year in subscriptions. The disaster recovery industry will continue to grow. The recent disaster events increase the growth of this industry. The market for data backup and disaster recovery will continue to grow in the foreseeable future.

In recent years, data recovery has become important for business continuity [6]. Data in our computers may be lost because of a user mistake, software error, site disaster, virus, storage failure, etc. Table 1 shows the most frequent disasters that cause data loss. According to the risk values in the table, natural events and IT failure appear to be the
riskiest causes of data loss. Disruptive act is almost fifteen times riskier than IT move or upgrade. Power outages and fire are seven and two times as risky as IT move or upgrade respectively. The water leakage has the lowest level of risk of the seven major types of disasters.

Table 1. Most Frequent Disasters [6]

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Risk Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural Event</td>
<td>Earthquakes, hurricanes, or severe weather (for example, heavy rains or snowball).</td>
<td>79.1</td>
</tr>
<tr>
<td>IT Failure</td>
<td>Hardware, software, or network problems.</td>
<td>69.7</td>
</tr>
<tr>
<td>Disruptive Act</td>
<td>Worker strikes and other intentional human acts, such as bombs or civil unrest, designed to interrupt the normal processes of organizations.</td>
<td>32.9</td>
</tr>
<tr>
<td>Power Outage</td>
<td>Loss of Power.</td>
<td>14.2</td>
</tr>
<tr>
<td>Fire</td>
<td>Electrical or natural fires.</td>
<td>4.5</td>
</tr>
<tr>
<td>IT Move/Upgrade</td>
<td>Data center moves and CPU upgrades undertaken by the company that cause disruption.</td>
<td>2.1</td>
</tr>
<tr>
<td>Water Leakage</td>
<td>Unintended losses of contained water (for example, pipe leaks, main breaks).</td>
<td>0.22</td>
</tr>
</tbody>
</table>

The ability to quickly recover the missed data when disasters occurred is essential to computing resources [7]. According to a report published by Strategic Research Corporation, under a system outage, for a brokerage firm, the cost is US $6.5 million per hour; for credit card authorization system, the cost is US $2.6 million per hour; for Automated Teller Machine (ATM), the cost is US $14,500 per hour in ATM fee.

Disaster recovery refers to a set of business processes covering how a company will recover quickly from disasters, data loss, communications problems, or facilities damage [8]. All of the businesses should have a proper disaster recovery plan. Otherwise, when disaster strikes, more than 90% of businesses that lose data would go out of business within two years. Therefore, having a disaster recovery plan is fatally important for a business’ survival.
Failures are classified as: transient, media, site, operator, and malicious failures, ranging from least to most severe [9]. Transient failure refers to the loss of messages due to the network. In addition, malicious failures are the worst since they can destroy all information. They may even attempt to destroy all backups.

Media failure refers to data corruption in storage devices [9]. To solve this problem, data must be backed up periodically. The backup procedure may be manual or automatic. The backup may be on tapes, disks, or cartridges. After a media failure, the backup can be retrieved to recover the lost data.

Site failure can affect a cluster of workstations in a room, or all clusters in a building [9]. Data on storage or on backup cartridges may be irrecoverable. Site failure is the first type of failure that is classified as a disaster. It affects computers within a large region. Existing disaster recovery plans and facilities are designed to tolerate site failures. Geographical separation of redundant hardware and data facilitates recovery from this type of disaster.

Operator failure is caused by human mistakes [9]. It is difficult to confine the scope of operator failures and to distinguish good data from bad data. The recovery procedure is time-consuming. Operator failure can be reduced by limiting the privileges of inexperienced operators.

Extensive backup procedures have been developed to protect against data losses during disasters. A system must be able to provide normal services after a disaster strikes. Data replication is the basis of disaster recovery solutions [9]. Most recovery of data and systems relied on redundancy. Redundancy allows secondary data or system resources to provide service in a short time when primary resources fail or become unavailable.
Traditional backup strategies archive copies of data at a given time so that they can be restored later. Currently, data are periodically copied onto a secondary storage device located far away. Organizations may also replicate servers and other hardware at multiple locations to protect against failure. If the primary storage device has failed, data on the secondary storage device will be immediately activated. The basic system architecture consists of a primary site and a backup site. The backup stores enough information so that if the primary fails, the information stored at the backup may be used to recover data lost at the primary. A lot of sophisticated disaster recovery techniques have grown in popularity due to the events of terrorist attack on September 11, 2001.

Both mirroring and chained-declustering techniques achieve high availability by having two copies of all data and indexes [10]. Mirroring technique stores two copies of data across two identical disks. Chained-Declustering technique stores two copies of each data across one set of disks while keeping both copies on separate disks. Both mirroring and chained-declustering techniques pay the costs of doubling storage requirements and requiring updates to be applied to both copies for immediate recovery.

Mirroring offers greater simplicity and universality [10]. The mirroring architectures do not share disks globally and mirroring does not require data to be declustered (spread over multiple disks). Mirroring uses a mirrored disk updated in parallel with the primary disk. In case of data loss on the primary disk, the missed data can be restored from the mirrored disk.

Chained-Declustering offers significant improvements in recovery time, mean time to loss of both copies of data, throughput during normal operation, and response time during recovery [10]. Chained-Declustering enables data to be spread across disks
for improved load balancing. Both of mirroring and chained-declustering provide high availability because data are available immediately during failure. The cost is that the size of disk space is doubled and data are required to be written to both copies.

Protection against disk failure is the main goal of Redundant Array of Independent Disks (RAID) technology [11]. The basic idea of RAID is to combine multiple inexpensive disk drives into an array of disks to obtain performance, capacity and reliability that exceeds that of a single large drive. Figure 1 illustrates the RAID architecture. The array of disks is managed by a RAID controller, which performs parity check, management of recovery from disk failures, and striping data across disks [12]. Internal drives are attached using Small Computer System Interface (SCSI). The array of disks appears as a single logical disk to the host system.

**Figure 1. Architecture of RAID [12]**

A RAID is a group of disks and one associated parity disk or another identical group of disks [12]. The redundant disks store redundant information to recover the
original data if a disk fails. The redundant copies of data across a computer network require the same amount of space as the original data. Redundant data on multiple disks provides fault tolerance and allow RAID systems to have the desirable availability to survive disk crashes. The redundant copies increase availability of computer systems not only in the disk failures but also in disasters or site failures. Therefore, the concept of RAID can be extended to a disaster recovery system.

Network recovery is the future of the disaster recovery industry. Backing up and storing of data over network will dominate the market in the next several years [13]. To protect data against disasters, remote sites are located hundreds of miles away. The IP connectivity provides more ways to access data during an emergency. With the IP-based remote replication, data is replicated from one site to another over the network infrastructure. Data centers use Fiber Channel for their storage networks. During initial set up, a copy of data at a site will be duplicated and transferred to a remote site over the network. During data recovery, the data at the remote sites are allowed to access over the network by any server located anywhere.

A storage area network (SAN) is a high-speed special-purpose network that interconnects different kinds of data storage devices with associated data servers on behalf of a larger network of users [13]. Typically, a storage area network is part of the overall network of computing resources for an enterprise. A storage area network can use existing communication technology such as optical fiber. The SAN supports disk mirroring, backup and restore, archival and retrieval of data, data migration from one storage device to another storage device, and the sharing of data among different servers in a network.
In today’s Storage Area Network (SAN) environment, the storage systems are centralized and interconnected [13]. A SAN is a high-speed network that allows the establishment of direct connections between storage devices and servers, e.g., Fiber Channel or Gigabit Ethernet. The SAN can be viewed as an extension to the storage bus concept, which enables storage devices and servers to be interconnected using similar elements as in local area networks (LANs) and wide area networks (WANs): routers, hubs, switches, directors, and gateways. A SAN can be shared between servers and/or dedicated to one server. It can be local or can be extended over geographical distances. The SAN provides new methods of attaching storage to servers. These new methods can enable great improvements in both availability and performance. Figure 2 illustrates a hierarchy of a typical SAN which includes a RAID.

![Figure 2. Hierarchy of a Typical Storage Area Network with a RAID System [13]](image-url)
A SAN facilitates direct, high speed data transfers between servers and storage devices [13]. Data can be transferred in three ways between servers and storage: server to storage, server to server, storage to storage. Server to storage is the traditional model of interaction with storage devices. The advantage is that the storage device may be accessed serially or concurrently by multiple servers. Server to server allows high-speed, high-volume communications between servers over the SAN. Storage to storage enables data to be moved without server intervention. This data movement capability can free up server processor cycles for other activities like application processing. The SAN allows applications that move data to perform better by having the data sent directly from source to target device with minimal server intervention. The SAN also enables new network architectures where multiple hosts access multiple storage devices connected to the same network.

The traditional backup and restore solution is tape media [13]. Network Attached Storage (NAS) is replacing tape for data backup. Online backup and recovery will protect business-critical data from natural disasters, human error, virus attacks, equipment failures, etc. NAS is a term used to refer to storage elements that connect to a network and provide data access services to computer systems. NAS is basically a LAN attached storage device that serves data via the network. A NAS storage device may be attached to any type of network.

SAN-attached storage devices can be organized into a common disk pool within a disk subsystem or across multiple disk subsystems [14]. Storage can be dynamically added to the disk pool and assigned to any SAN-attached server. This provides efficient
access to shared disk resources since all storage devices are directly attached to all the servers.

The SAN provides many benefits [13]. The SAN improves application availability and increases application performance. Storage is independent of applications and accessible through multiple data paths for better reliability, availability, and serviceability. Storage processing is off-loaded from servers and moved onto a separate network. The SAN also simplifies and centralizes the storage management.

A disaster recovery system relies on three components: the server, client and storage [15]. Figure 3 illustrates the three components. Each computer and server to be protected will have client software installed on it for communications with the server and the storage. The client software on each individual computer or server will send the system’s data to the remote storage when responding to a backup task. When a restore is required, the client can quickly access the online data and restore a file or an entire system. Once this is done, an organization can use the server program to configure and schedule backup tasks on all of their protected systems.
Due to competitive pressures, companies require data protection, rapid recovery and simplified management to be cost effective, more controllable, reliable, secure and fast [16]. Organizations need a storage management and data protection solution that must meet a number of requirements. First, an ideal backup solution must provide up-to-the-moment data protection because disasters do not occur on a set schedule.

Second, an ideal backup solution should not noticeably degrade network performance [16]. Many companies perform backups during off hours to minimize the impact of the slowdown on employee productivity. A storage management solution that

Figure 3. Three Components of a Disaster Recovery System [15]
minimizes network traffic allows companies to perform backups as needed during the day to ensure a high level of data protection.

Third, an ideal backup and recovery solution should help keep administration costs under control by allowing centralized administration and control because network administration and management represent a major component of the cost of network ownership [16]. The administrator should be able to install, configure, and administer storage management client software from a central location, without traveling to each client site. Figure 4 illustrates centralized administration and management of a backup system.

![Centralized Administration and Management of a Backup System](image)

**Figure 4. Centralized Administration and Management of a Backup System [16]**

Fourth, an ideal storage management solution should maximize the storage efficiency of data backup and the efficiency of data transmission over the network [16,
The volume of data being stored on networks is growing dramatically. So does the volume of backup data [16]. Ideal storage management solution employs data compression to reduce space occupation and network traffic [17]. Traditional backup is performed on tapes. With the price of disk becoming cheaper and CPUs becoming faster, the use of disk-based network backup system to replace tape-based backup becomes realistic. Disk-based compression takes the advantage of both compression speed and compression ratio over tape-based compression. Thus, disk-based backup system can perform data compression more efficiently and quickly.

2.2. Data Protection Methods

This subsection discusses the technologies to protect our data. There are two major technologies to recover the missed data from failures. One is striping with distributed parity, which can reconstruct the missed data during a failure. The other is chained-declustering, which stores a copy of data in its neighbor. The concept of Redundant Array of Independent Disks (RAID) can also be extended to recover data from disasters and/or failures.

Striping with Distributed Parity: In striping with distributed parity, disks are divided into blocks [18]. A stripe is a collection of data blocks and an associated parity block across storage servers. A block in a stripe is called a stripe unit. A parity block contains parity bits used to recover data when there is a failure. Parity of each stripe is computed by XORing each data block in the stripe. Figure 5 shows that stripes span across storage servers 0 to 3. In a stripe, each block is stored on a different server, and the collection of servers that a stripe spans is called a stripe group. In Figure 5, the stripe group is the set
of all four servers. In the figure, each stripe has three data blocks and one parity block. For example, Stripe 1 is composed of data blocks \( D0, D1, D2 \) and parity block \( P0 \).

![Figure 5. Striping with Distributed Parity](image)

As mentioned earlier, the parity mechanism is used to recover the lost data in case of a server failure. The manager server computes parity for each stripe and writes the parity as the disk is written. Parity of a stripe is computed by XORing all data blocks in the stripe. The parity fragment allows the content of any fragment in a stripe to be reconstructed by XORing the contents of the rest of the fragments with the parity. If one of the servers fails, the manager server will fetch all data from the remaining servers and reconstruct the missing data by XORing the parity with them.

The disadvantage of striping with distributed parity is that parity reconstruction creates high load. To recover a failed disk, it has to fetch all the data from the remaining disks in the stripe and XOR them to reconstruct the missed data. Moreover, it has to fetch all data in the stripe to perform parity computation for each new data coming. Therefore, system performance could be degraded when backing up a large file.

**Chained-Declustering:** Another strategy to protect data from disk failure is chained-declustering. It relies on data replication, which is to store a copy of data in another
server. Therefore, when one server fails, we still have another copy available. The overhead is that the disk space required is doubled.

Figure 6 illustrates the mirrored-stripping data placement scheme. In the mirrored-stripping data placement scheme, two copies of each data block are stored on neighboring servers and stripped round-robin across a set of mirror servers [19]. Each pair of mirrored servers are identical to each other. However, with such a simple mirrored redundancy scheme, the failure of one server will result in a 100% load increase on the other with no opportunity for dynamic load balancing.

![Figure 6. Mirrored-Striping Strategy for Data Replication](image)

Chained-Declustering offers high availability and load balancing in the event of failures. Figure 7 illustrates the chained-declustering data placement scheme. Chained-Declustering is similar to mirrored-stripping. The difference from the mirrored-stripping data placement scheme is that two copies of each data block are stored by interleaving. Every pair of neighboring servers has 50% of common data blocks. In chained-declustering, the first copy is termed primary copy and the second copy is termed backup copy. In a system that contains $N$ disks, if the primary copy resides at stripe $i$ on server $j$, the backup copy resides at stripe $i + 1$ on disk $(j+1) \mod N$. 

17
In chained-declustering, each pair of copies are stored by interleaving and round-robin across the servers. Chaining the data placement in this way allows each server to offload its read load to both of its neighbors. Figure 8 illustrates how the two neighbors share a failed node’s load. When Server 1 fails, Servers 0 and 2 will automatically share Server 1’s workload. Server 1’s copy of $D0$ is on Server 0, Server 1’s copy of $D1$ is on Server 2, and so on. Therefore, if Server 1 fails, Server 0 and Server 2 share Server 1’s read load but Server 3 experiences no load increase. The workload on Server 0 and Server 2 will increase 50%. This is not perfect but better than having a single server bear the entire load of the failed server.

In this case, only the two neighbors bear the workload of the failed server. The rest of the working servers do nothing. A better way to reduce the workload on the two
neighbors is to get all the remaining servers to share the failed server’s load. That requires each server to distribute its backup copies to all other servers in the stripe group. Though it may reduce the maximum server load, it increases the probability of failure. When a server fails, some data will be unavailable if any of the other servers fails.

Dynamic balancing allows all remaining servers to share the workload and maintain availability [20]. Data blocks are chained across neighboring servers in chained-declustering. Therefore, the load of a server can be easily shifted to its neighbors. With dynamic balancing, all remaining servers bear the workload of the failed server rather than only its 2 neighbors such that the workload on each server only increases by \( \frac{1}{N-1} \), where \( N \) is the number of total disks. When a server fails, the workload can be dynamically reassigned and uniformly distributed to all remaining servers. In Figure 9, Server 1’s workload is evenly shifted to both of its neighbors Servers 0 and 2. Therefore, workload on Servers 0 and 3 would increase 50%. To balance the load increase, Servers 0 and 2 shift 1/3 of workload on \( D2, D3, D6 \) and \( D7 \) to their chained copies on Server 3 such that all remaining servers bear the same amount of workload. Thus, the workload on Servers 0, 2 and 3 only increases by 33.33%.

![Figure 9. Dynamic Balancing in Chained-Declustering](image-url)
Replicating data copies allow us to recover from failures by restoring missing data from the other copies. If only a single server fails, all its data can be easily recovered by restoring from both of its neighbors. Figure 10 shows how multi-failures occur in chained-declustering. If two or more servers fail, we still can recover the failed servers as long as they are not adjacent. In chained-declustering, the two copies of data are stored across two consecutive servers. If the number of servers is even, the set of odd-numbered servers contain the same data as the set of the even-numbered servers. Therefore, an additional advantage with chained-declustering is that we can recover from site failures by placing all the even-numbered servers at one site and all the odd-numbered servers at another site.

![Figure 10. Chained-Declustering during Multi-Failures](image)

A disadvantage of chained-declustering relative to mirrored-striping is that it is less reliable [21]. With mirrored-striping, if a server failed, only the failure of its mirrored server could cause data unavailability. With chained-declustering, if a server fails, a failure on either of its two neighboring servers will cause data unavailability. Thus, there is a higher likelihood of data unavailability. Figure 11 shows that two consecutive servers fail at the same time and cause data unavailability. In this case, although \( D0, D2, D4, \) and
$D_6$ can be easily restored from Servers 0 and 3, $D_1$ and $D_5$ will not be recovered because both copies of $D_1$ and $D_5$ are lost in Servers 1 and 2.

![Diagram of data unavailability in chained-declustering](image)

**Figure 11. Data Unavailability in Chained-Declustering**

**Redundant Arrays of Independent Disks:** Redundant Arrays of Independent Disks (RAID) is a storage system, in which many disks are organized into a large and high-performance logical disk [22]. It is designed to provide high data availability. RAID has 5 levels of basic architectures. Level 0 is non-redundant, level 1 is mirroring, level 2 is striping with Hamming Code ECC, level 3 is byte-striping with parity, level 4 is block-interleaved with parity and level 5 is block-interleaved with distributed parity.

Figure 12 shows the RAID from level 0 to level 5. “D” represents a block of data, “d” represents a bit or byte of data, “hx-y” represents a Hamming code computed over data bits/bytes x through y, “px-y” represents a parity bit/byte computed over data blocks x through y, and “Px-y” represents a parity block computed over data blocks x through y. The numbers on the left indicate the stripes. Shaded blocks represent redundant information, and non-shaded blocks represent data.
Figure 12. Data and Redundancy Organization in RAID Levels 0 through 5 [22]
Data backup and disaster recovery system and RAID use similar technologies (striping, parity and mirroring) to protect data. The concept of RAID can be applied to backup and disaster recovery system to protect data against failures. Both data backup and disaster recovery system and RAID are designed to provide secure data storage and high data availability. The difference between data backup and disaster recovery system and RAID is that, in RAID, disks are connected via Small Computer System Interface (SCSI). All data stored in RAID are local because all disks are placed nearby. RAID is mainly used to protect data against disk failure. When one disk crashes, we still have a copy of data on another disk or we can reconstruct the missed data if parity is used to protect data. RAID may be used as the Network Attached Storage in a data backup and disaster recovery system.

In a data backup and disaster recovery system, disks are attached via the LAN or SAN. In data backup and disaster recovery system, copies of data are allowed to be stored at remote sites to protect against site failure or disasters. For example, when an earthquake happens, all disks at a place may be damaged. In this case, RAID cannot provide any protection to data. All the data, including the backup copies, can be destroyed during the earthquake because all of the disks are placed at the same site. Data backup and disaster recovery system will protect our data against this natural disaster because its storage devices are placed at different sites. After the earthquake, we still can recover our data from a copy stored at a remote and secure place.

Data backup and disaster recovery system is designed to recover data from site failure or disaster. RAID is designed to recover data from media failure. Data backup and disaster recovery system requires geographical separation of redundant hardware and
data. RAID locally provides stable storage to data. A RAID system could be used as a storage device in a data backup and disaster recovery system.

2.3. Related Work

In this subsection, we will discuss several backup systems that provide protection to data and how they compare to our system. We will discuss Petal and Swarm systems, which are two research backup systems. We will also discuss IBM Tivoli Storage Manager (TSM) and VERITAS Global Cluster Manager, which are current commercial backup systems. After that, we will compare our system with these systems.

Petal: Petal is a block level storage system [21]. Petal consists of a collection of network-connected servers that manage a number of disks. From a Petal client's view, the collection of virtual disks is a highly available storage system.

Storage servers in Petal cooperatively manage a pool of physical disks. Each Petal server is as complex as a RAID controller. Petal currently supports chained-declustering. The data replication scheme is specified when the virtual disk is created. Figure 13 shows the prototype of Petal.
Swarm: Swarm is a network storage system [23]. It uses log-based data striping to provide high throughput by using data striping. Segments of the log are striped across storage servers, similar to RAID terminology. Unlike RAID, the striping is done across the network.

In order to achieve high reliability, Swarm clients compute and store parity for each stripe. If a single server crashes, the system will be able to continue operation by computing the missing data using the parity. This downside of this is that it increased storage space usage. Figure 14 illustrates log-based striping in Swarm. Files A, B and C are collected in a log and the log is striped across the servers. File A is large enough to fill an entire stripe, while files B and C together fill a stripe. Parity information is stored for each stripe.
IBM Tivoli Storage Manager: IBM TSM is a disk-based backup system [24]. Tivoli Storage Manager limits the bandwidth needed for backups and restores by performing progressive incremental backups. This reduces the total amount of storage space required by the storage management system.

Data in the primary storage pool can be duplicated in one or more Copy Storage Pools. Mirroring protection is used to recover disasters or hardware failures. Backup data can be migrated off to tape storage to free up the disk space. The Tivoli Storage Manager Server automatically tracks the location of the backup images and initiates the restore process. Restored data from the storage will travel through the Tivoli Storage Manager Server to a Tivoli Client over the LAN or SAN. Figure 15 shows the TSM backup and recovery processes.
VERITAS Global Cluster Manager: VERITAS disaster recovery system is consisting of VERITAS Global Cluster Manager, VERITAS Cluster Server and VERITAS Volume Replicator [25]. Figure 16 shows a VERITAS disaster recovery system. VERITAS Cluster Server handles local data availability, VERITAS Volume Replicator duplicates data to a remote site, and VERITAS Global Cluster Manager monitors and manages the whole system. VERITAS replication and remote mirroring technologies can provide quick recovery by making current data immediately available at an alternate location.
System Comparison: Basically, Petal system organized data copies in chained-declustering data placement scheme. Swarm organizes data into stripes and recovers missed data by using parity. Both of IBM TSM and VERITAS Global Cluster Manager systems use mirroring to protect data. Our system uses our new approach -- chained-declustering with distributed parity, which is discussed in Section 3, to protect data.

Petal cannot recover neighboring failure. Swarm cannot recover failures from multi-failures. Our system is able to recovery multi-failures with any two adjacent failures. The probability for our system to survive failures is higher. Therefore, our system is more reliable than Petal and Swarm.
Comparing with IBM TSM and VERITAS Global Cluster Manager, our system is more load-balancing during failures and our system requires less disk space to recover neighboring failure. Data of our system are stored in chained-declustering data placement scheme. Such that our system allows dynamic balance during failures. IBM TSM and VERITAS Global Cluster Manager are mirroring. To recovery neighboring failure, both of them require another multiple of disk space. Our system only needs one more server for the parity.

3. System Design and Implementation

To address the disadvantages of striping with distributed parity and chained-declustering, we propose two improvements: fast parity update and chained-declustering with distributed parity. Fast parity update computes parity in a more efficient way because it requires less I/O requests. Chained-Declustering with distributed parity can recover from neighboring failure. We independently discovered fast parity update. Later, we found that RAID 5 used a similar method to update parity. Chained-Declustering with distributed parity is a unique technique to protect data so far. We have not found any reference that used this approach since we discovered it. We will discuss details of our improvements on fast parity update and chained-declustering with distributed parity. We also discuss how to implement our own remote backup system, which allows clients to perform backup, retrieval and restore operations and allows the manager server to recover from disk failures. We will introduce fast parity update and chained-declustering with distributed parity in Subsections 3.1 and 3.2 respectively. In Subsection 3.3, we will describe the components of the system in more detail, and in Subsection 3.4, we will describe its operations.
3.1. Fast Parity Update

The idea of Fast Parity Update is to cancel out the original data term from the parity. Then other terms remain in the parity. When a new data is being backed up, we can simply XOR the new data with the remaining terms to compute the new parity. In other words, the old data term is replaced by the new data. Thus, we do not have to fetch data from all the servers to compute the new parity for each backup.

A parity is computed by XORing all data in the stripe. Each time the disk is written, the manager server has to compute the parity again. In Figure 17, when new data arrives, the manager server has to fetch data from all the fragments in a stripe to compute the new parity. Fetching data from all servers slows down the system’s performance.

Fast Parity Update will reduce the process time by fetching data only from two data blocks to compute the new parity, instead of from all data blocks. Figure 18
illustrates the procedures of performing fast parity computation. If new data is being backed up, we can cancel out the old data term from the parity by XORing the old data with the parity. After that, the other terms are still left in the parity. Then we can compute the new parity by XORing the new data with the remaining terms. Thus, we only need to fetch the parity and the original data instead of all data in the stripe to do the parity computation. Therefore, the process time is shortened.

**Figure 18. Fast Parity Computation Procedures**

Figure 19 shows a stripe that contains three data blocks $X$, $Y$, $Z$ and a parity block that equals to $X \oplus Y \oplus Z$. Say a new data $Z^*$ is to replace data block $Z$. After data $Z^*$ arrives, the manager server will fetch the parity and data $Z$. 
Then the manager server XORs data Z with the parity and cancels out the data Z term from the parity:

\[
\begin{align*}
\text{Old Parity} & \oplus Z \\
= (X \oplus Y \oplus Z) & \oplus Z \\
= (X \oplus Y) & \oplus (Z \oplus Z) \\
= (X \oplus Y) & \oplus 0 \\
= X & \oplus Y \\
\end{align*}
\]

Note that the data X and Y terms are still left in the parity. We can compute the new parity by XORing the new data with the remaining terms:

\[
\begin{align*}
\text{New parity} \\
= \text{Old Parity} \oplus Z \oplus Z^* \\
= (\text{Old Parity} \oplus Z) & \oplus Z^* \\
= (X \oplus Y) & \oplus Z^* \\
= X & \oplus Y \oplus Z^*
\end{align*}
\]

Thus, to compute the new parity, we only need to fetch the old data and parity rather than all the data.

For the remainder of this subsection, we will prove our proposed method is efficient and requires less I/O requests. Suppose a system has \( N \) disks, the number of
reading data from storage is $R$ and the number of writing data to storage is $W$. Then $N =$ number of disks, $R =$ number of reads and $W =$ number of writes. The old way involves all of the disks when backing up data. Thus, for each writing operation, the number of disk accesses required by parity computation in the old way is $R+ W$, where $R = N-2$ and $W = 2$. The number of disk accesses is linear to the number of disks. Fast parity update only requires two read requests to read the old data and parity, and two write requests to write the new data and parity. The number of disk accesses required to update the parity by using Fast Parity Update is $R = 2$ and $W = 2$.

This process is similar to RAID 5 system's Read-Modify-Write (RMW) parity update. RAID 5 systems use this approach to reduce disk accesses [26]. Distributed parity RAID 5 does not have a dedicated parity disk, but interleaves both data and parity on all disks [29]. In RAID 5, the access arms can move independently of one another. This enables multiple concurrent accesses to the array disks, thereby satisfies multiple concurrent I/O requests and providing higher transaction throughput.

We can reduce the arm movements by reducing the disk accesses. Therefore, we can reduce tear and wear or the mechanical damage of the array disk. The reliability of the storage system will increase. The SAN enables multiple hosts to access multiple storage devices connected to the same network and allows disks to be accessed by servers serially or concurrently [13]. For a disaster recovery system, if data are accessed in serial, fast parity update will be able to shorten the processing time.

3.2. Chained-Declustering with Distributed Parity

When a server fails, we can recover its data from the copy. If multi-failures occur, we still can recover them in a straightforward manner from their copies as long as any
pairs of them are not consecutive. One way to recover multi-failures is long-chained-declustering, which stores $N$ copies of data across different servers. Missing data can be recovered if no more than $N$-1 failures occur. Hence, the disk space required is multiplied by $N$ times.

We do not have to use $N$ times the disk space to recover multi-failures. Since the probability of two consecutive disks failing at the same time is low, we can bear the time consumption of parity reconstruction. Chained-Declustering with distributed parity recovers multi-failures by using parity in chained-declustering. We only need to add one more disk rather than multiply $N$ times the disk space. The procedures for chained-declustering with distributed parity to recover multi-failures are discussed in the rest of this section.

Reliability is the most important thing for a data backup system. In chained-declustering, when a server fails, we can easily recover it from the other copy in its neighbors. If one server fails and either of its neighbors also fails, missing data will not be recovered. We can solve this problem by using parity in the chained-declustering data placement. This can be done by organizing the data blocks across servers into stripes and using one of the data blocks in each stripe as a parity fragment in chained-declustering scheme. We will use the parity to reconstruct the missing data during neighboring failures. Figure 20 illustrates the chained-declustering with distributed parity.
In a chained-declustering data placement scheme, the even-numbered stripes contain primary copies and the odd-numbered stripes contain backup copies. We call a stripe that contains primary copies a *primary stripe* and a stripe containing backup copies a *backup stripe*. Hence, each server's first data block contains a primary copy and the backup copy is stored on the neighbor after it; meaning the second data block contains a backup copy and the primary copy is on the neighbor before it.

Next we will explain how data is recovered from two adjacent server failures. Without loss of generality, let the stripes be number 0, 1, 2, .... Recall that stripes are paired so that all even numbered stripes $i$ are primary and odd numbered stripes $i+1$ are backup to even numbered stripes $i$. A block in server $j$ of a primary stripe $i$ has a backup copy in server $(j+1) \mod N$ in backup stripe $i+1$, where $N$ is the number of disks.

Now suppose there are two failures in server $j$ and $(j+1) \mod N$ for some arbitrary $j$. Then there are two failures in a primary stripe $i$. However, the failure in server $(j+1) \mod N$ has a backup copy in server $(j+2) \mod N$ in backup stripe $i+1$. Therefore, stripe $i$ only has one failure, which is in server $j$. Since it has only one failure, the primary stripe can be recovered with the parity.
Also, there are two failures in the backup stripe $i+1$. However, the failure in server $j$ has another copy in server $(N+j-1) \mod N$ in primary stripe $i$. In addition, the failure in server $(j+1) \mod N$ has the primary copy in server $j$ in primary stripe $i$, which is recovered with the parity. Therefore, the backup stripe can be recovered. This shows that the system can recover from two adjacent server failures.

In chained-declustering, a data block cannot be recovered if both of its copies are lost in the two failed servers. When neighboring failures occur, we can recover the failed blocks that still have an available copy on their neighbors first. After that, each stripe only has one failed block because the two copies of each missing data are interleaving across the two failed servers. Then we can reconstruct the missing data by XORing the data from remaining servers. Finally, we can recover the two failed neighboring servers after reconstructing the missed data blocks in all stripes. Figure 21 illustrates this situation for the case of $N = 4$ servers.
When two neighboring servers fail at the same time, each stripe has two failed data blocks. We can recover some of the backup copies in the first server (Server 1) since their primary copies are stored on the server before it (Server 0). Data $P0$ and $D3$ on Server 1 are recovered from Server 0. Meanwhile, we also can recover the primary copies in the second server (Server 2) from their backup copies stored on the server after it (Server 3). Data $D1$ and $P1$ on Server 2 are recovered from Server 3. If both copies of a data are lost in the 2 failed servers, the data will not be recovered in chained-declustering, like $D0$ and $D4$ cannot be recovered in this example. Figure 22 illustrates data recovery from the other copies when neighboring failure occurs.
Parity reconstruction is illustrated in the example in Figure 23. In the figure, Servers 1 and 2 fail. Then each primary stripe can recover its data lost in Server 2 (blocks D1 and P1) because there are backups in Server 3 in the respective backup stripes. And each backup stripe can recover its data lost in Server 1 (blocks P0 and D4) because there are backups in Server 0 in the respective backup stripes. The data in Server 1 (blocks D0 and D4) whose backups are lost in Server 2 can be recovered using parity reconstruction.
After the missed are reconstructed, we will write them onto both of their primary and secondary locations. Figure 24 shows that missing data are recovered. The neighboring failure will be recovered after every pair of stripes are reconstructed.
We will quantify the improved reliability of chained-declustering with distributed parity over ordinary chained-declustering. Consider a chained-declustering system with \( N \) servers, and suppose one has failed. The system is still operable. Suppose there is a second failure, equally likely to occur at any of the remaining servers. The likelihood that it is adjacent to the first failed server is \( \frac{2}{N-1} \). Thus, there is a significant probability of system failure if there are two server failures, especially if \( N \) is small. On the other hand, chained-declustering with distributed parity will survive any two failures. We should note that this comparison does not take into account that chained-declustering with distributed parity stores less data. In particular, the parity occupies \( \frac{1}{N} \) of the total disk space.
3.3. System Components

The Remote Backup and Disaster Recovery system is shown as in Figure 25. It has a manager server. The manager server provides service to the clients by storing backup data from the clients into remote storage and returning retrieved data from the remote storages to the clients after receiving requests from the clients. The manager server protects the data stored in disks by performing data recovery and data reconstruction.

Figure 25. Remote Backup System
In order to have better management, we use the manager server to centralize the administration. It contains a database that keeps track of addresses of data in the disks, including primary copies and backup copies. It sends data to disks when receiving backup commands from clients. It also reads data from disks, then returns to the clients after receiving retrieve or restore commands. The manager server can automatically perform data recovery and data reconstruction when failures are detected.

We transfer data over a Storage Area Network (SAN) rather than LAN to reduce the traffic from the LAN. Control information from the clients flows over the LAN from the clients to the server. Data flow over a SAN, which could be a Fiber Channel or a Gigabit Ethernet from clients to the servers and then storage devices. A Host Bus Adapter is a PCI/Fiber interface which allows the clients to transmit data over a fiber channel.

**Clients**: Original data are stored in clients. Clients are able to backup, retrieval, and restore data. Clients initiate backup, retrieve, and restore operations by sending commands to the manager server. The clients do not have to know how and where their data is stored. The information is saved into the database in the manager server.

When backing up data, a client sends a BACKUP command and the backup data to the manager server. The manager server will send the backup data to storage and save the data information in its database. When retrieving or restoring data, a client sends a RETRIEVE command or RESTORE command along with the data information to the manager server, the manager server will use the data information to calculate the data’s addresses in the database. Then the manager server fetches the data from the addresses and returns to the client.
Manager Server: The manager server provides management services to clients. It receives control information and data information from clients, manages data flow and recovers from failures [27]. The client generates the backup data as a data stream using TCP/IP sockets and sends it across the network to the manager server. The manager server determines where to locate the data's primary copies and secondary copies so that they are in chained-declustering with distributed parity data placement. The manager server directs the received data stream to the appropriate attached storage device. While the backup operation is progressing, information about the data is stored in the database. When backing up to disk, the manager server writes data to the file path and records the path of the data in the database.

The Manager Server contains a database manager, a backup manager and a disk manager. The backup manager takes care of the communication with clients and performs backup operations. The database manager maintains a database, which records information about all backup and restore operations, and all of the data. The disk manager transfers data between clients and disks, and manages the storage. The disk manager handles the actual writing of data onto the disk. It receives backed up data from clients and writes them to disks, or reads data from disks and return them to clients. It contains a Lease Table that detects the status of disks. When failures are detected, the manager server will automatically recover the data on them. Figure 26 illustrates the backup data flow between clients, the manager server and remote storage.
Database stores the data information such as owner, type, size, time, address, etc. The database consists of the database space and the recovery log. The database is the heart of the manager server. It is important to the operation of the server because it contains data location information. When backing up, it saves information about the new data. When retrieving or restoring, the manager server will look at the database to find out the data’s addresses before reading data from disks.

One of these features is the recovery log. A recovery log is used to help maintain the integrity of the database. It keeps track of all changes made to the database. When a change to the database occurs, the recovery log is updated before the database is updated. A record of the newest changes to the system could be saved in the log before a system failure occurs. Then we can recover from the latest version after a crash.
Remote Storage: Storage servers perform actual data read/write operations onto disks and provide disk space to data [28]. The storage server component provides storage resources and services for the clients. Clients back up or retrieve their data onto server storage resources such as disk, tape, or optical devices.

As shown in Figure 27, a storage server contains a database, log and data pool. The two key components of the server are the data pool and database. The data pool is where the data are actually stored. All data are stored in this data pool. The database serves as an inventory or index to the physical locations of data within the data pools. The log is similar to the recovery log in the manager server. It keeps the latest changes of data and maintains integrity of the database.

Storage Area Network: The capacity of hard drive has been growing tremendously. A computer may contain a 40 GB, 80 GB, 160 GB or even bigger hard drive. The size of files is getting larger and larger too. For example, a movie file may be larger than 1GB. It may take more than ten hours to transfer such a movie file over the LAN. Using the LAN to backup data may cause very busy network traffic and disrupt other applications’ access to the network.
A Storage Area Network (SAN) is a network consisting of servers and storage devices (RAID, tap, switch, etc.) and attached via a high-speed network such as Gigabit Ethernet, ATM, Fiber Channel, etc, which range between 2Gbps and 10Gbps [29]. The storage devices on the SAN are called Network Attached Storage (NAS). Figure 28 illustrates the SAN and Figure 29 illustrates the NAS.

Figure 28. Storage Area Network [29]
The LAN-free data-transfer function is for the Storage Area Network to carry data protection traffic instead of the LAN [30]. The LAN connection is used to transfer control information such as operation commands and metadata between the server and the clients. Meanwhile, data requests are made directly to disk over the SAN. The data movement is carried over the SAN and data is written directly to the storage devices over the SAN. The SAN can offload the data movement of backup and recovery from the LAN. That will avoid disrupting applications on the LAN and allow the user to access the network when backing up data. Figure 30 illustrates the data flow travel through clients, a backup server and storages over the SAN.
SAN provides an alternative path for data movement between the clients and server. LAN-free data transfer expands the SAN path by allowing the clients to back up and restore data directly to and from SAN-attached storage, which is shared between the server and clients and managed by the server. Figure 31 shows that clients, servers and storages are connected over SAN.
Clients, servers and disk storage pools are SAN-connected so that data can move directly between client computers and the SAN-connected disks over the SAN [28]. The SAN not only offloads LAN traffic but also it restores data directly from the SAN-connected disks and returns the data to the client. When the SAN connection fails, the server can automatically switch to LAN to transfer data. LAN communications bandwidth will be used for the usual backup data traffic. In this case, data and control information are transferred over LANs.
3.4. Operations

For this system, clients are allowed to backup, retrieve and restore data onto remote storages through a manager server. Clients decide which files to backup or retrieve. Storage servers are commanded by the manager server to read or write data onto disks. The master server handles storage server management, data routing (send data to storage server), and failure detection and recovery. Next we describe the three operations of the system.

**Backup:** This system uses *incremental backup* to back up data. The incremental backup only backs up files changed since the last backup, which could be full or incremental backup [31]. For the first time, data are backed up by using *full backup*, which is to backup all the files stored in the clients to the remote storage devices. After that, only modified files since the previous backup need to be backed up to the storage devices.

The incremental backup reduces the traffic on network and speeds up the backup process. Optimized backup performance can be achieved by processing backup operations during off-peak hours. Figure 32 illustrates the full backup and incremental backup. All the files are backed up to storage by full backup on weekend. Then only the files that have been modified need to back up, such as file 2a and file 4a on Tuesday.
Figure 32. Full Backup and Incremental Backup [31]

For the first time, clients back up all the data. After that, clients only back up the new data or modified data. To initiate a backup operation, the client sends a BACKUP command to the manager server first. When the manager server receives a BACKUP command from clients, the manager server will check the database and decide where to store the primary copy and the backup copy of the data, then sends back ACK. After the client receives ACK, it begins to send data to the manager server. When data arrive at the manager server, the manager server will send them to the remote storages and write them into the addresses in disks.

Retrieve and Restore: Retrieve operation allows a client to fetch a specific file from the remote storage. The restore operation allows a client to fetch all its data stored in the remote storage. When the manager server receives a RETRIEVE command, it will look at its database and find the file's location. Then it fetches the data and returns the data to the client. When it receives a RESTORE command, the manager server will look at the
database and find out all the data belonging to the client. Then the manager server fetches all the data belonging to the client and returns the data to the client.

**Failure Detection and Recovery:** When a storage device is added to the system, it will send a membership request message to the manager server [31]. The manager server will grant it membership by issuing a lease and assigning a lease ID to it after receiving the request. Lease and disk information are saved in a Lease Table. All storage devices have to renew the lease every 30 seconds. If the manager server does not receive a lease before timeout, the manager server will treat the storage as a failed node and start to recover it.

4. Simulation Results

We discuss how to test this system and the simulation results in this section. We will show that backup, retrieval and restore operations are executed successfully. Data can be backed up, retrieved, restored and reconstructed correctly. Therefore, our proposed improvements, fast parity update and chained-declustering with distributed parity, work well. Subsection 4.1 discusses the tests and results of backup, retrieval and restore operations. Subsection 4.2 discusses the tests and results of missing data recovery and reconstruction. Subsection 4.3 discusses the simulation data and comparison of fast parity update and the parity computation in the old way. Finally, Subsection 4.4 discusses the simulation result.

This test uses a client, a manager server and five disks. Note that one client is sufficient to verify correctness. In addition, using one client will reduce the complexity of implementation. The client, the manager server and the five disks are connected via the SAN and LAN. Two switches forward data flows to the destinations. Data are transferred
over the SAN. Control information and communication between the client and the server are transferred over the LAN.

The system simulation is implemented in C++ under Visual Studio 6.0. The executable application can run on a PC. It allows a user to input data into the client and modify the data. Also, it allows the user to enter BACKUP, RETRIEVE, RESTORE and RECONSTRUCTION commands to test the corresponding operations.

I used DiskSim2.0 to measure and compare the performance of fast parity update and parity computation in the old way. DiskSim2.0 is an efficient and accurate disk system simulator written in C [32]. It was developed by Carnegie Mellon University to evaluate performance of RAID architectures and storage systems.

4.1. Test Results and Discussion of Backup, Retrieve and Restore Operations

To test the backup operation, we entered data to the client. Then the client will back them up to the manager server. The manager server will store the data into the primary and backup location on disks.

After the initial full backup is done, we enter a RESTORE command to the manager server. The manager server will check its database to find out where the data are stored. After all data belonging to the client are found, the manager server will fetch all the data from storage and return the data to the client. When the client has received the data, it will compare the restored data with its local data. If the restored data matches the local data in the client, the full backup operation and restore operation were completed successfully.

After the full backup operation is done, we can test the incremental backup. We modify the data in the client. Then the client will detect the change and back up the new
data. After the incremental backup operation is done, we use a RETRIEVE command to get back the data and then use a RESTORE command to test whether we can get back the latest data.

To show the operations work correctly, we use the restore and retrieve operations to get back the backup data and compare them with the local data after each backup operation. If the data fetched from the disks match the data in the client, it means the client can restore its data from the remote storage and backup operations are done correctly. For example, we initially enter a string “ThisIsNO.XComputer.”. After the full backup is done, we can get back this string by entering a RESTORE command to fetch the client’s data. Next, we change “X” to “0”. After the incremental operation is done, we can get back “0” instead of “X”. We also can get back the string “ThisIsNO.0Computer.”. All the data fetched from disks match the local data in clients exactly. Thus, all the data can be backed up, retrieved and restored correctly.

4.2. Test Results and Discussion of Missing Data, Recovery and Reconstruction

To simulate a disk failure, we erase all the stored data of the disk and set a failure flag after a RECONSTRUCTION command and the disk’s ID are received. For easier control of the program, the manager server checks the disk flag instead of timeout very 30 seconds. If the disk flag is set to failure, it knows the disk has failed. Then it will start the recovery procedures we mentioned above to recover it.

To test single failure, we disable a disk. The manager server will detect which disk is not working. Then the manager server will automatically launch the recovery process to recover it. Multi-failures are tested similarly to the single failure recovery. Multiple disks are disabled. If the manager server finds out the failures are not adjacent,
the manager server will recover them by copying the other copies to the failed disks. The manager server will recover nonadjacent failures first, then start the neighboring failure reconstruction process.

For each time failures are recovered, we use a RESTORE command to fetch the data from disks to compare with the local data in clients. If they match each other, it means the system does not suffer the problem of data unavailability and the reconstruction operation is done successfully.

To show the data are recovered and reconstructed correctly, we save the data stored in the disk before erasing them to simulate the disk failure. Then we compare them with the reconstructed data after neighboring failure is recovered. The data before failure match the reconstructed data perfectly. Missing data are reconstructed correctly. This not only proves that neighboring failure recovery works but also fast parity update works because parity has to be correctly updated.

4.3. Simulation Data and Comparison

Suppose a system has $N$ disks, the number of reading data from storage is $R$ and the number of writing data to storage is $W$. When update a new data in the old way, we have to fetch all data except the old parity and the old data. We XOR them with the new data to compute the new parity. Since the backup copy can be directly copied from the primary location, we only need to write the new data and the new parity to disks. Therefore, in the old way, the number of disk accesses required to store the new data and update the parity is $R = N-2$ and $W = 2$. The number of read requests is linear to the number of disks.
For fast parity update, we can compute the new parity by XORing the new data with the old parity and the old data. Then we only need to fetch the old parity and the old data. Therefore, the number of disk accesses required to store the new data and update the new parity by using Fast Parity Update is \( R = 2 \) and \( W = 2 \).

Fast parity update and the old way have the same number of writes \( W = 2 \). Fast parity update reduces the number of disk accesses by cutting off the read requests from \( N - 2 \) to 2. The number of disk accesses required by fast parity update is fixed since it is bounded by constants. The number of read requests required by the old way is a linear function of \( N \). The more disks used in the system, disks will be accessed more.

To show fast parity update is more efficient than the parity computation in the old way, we used DiskSim2.0 to simulate the processes. We chose all of I/O requests to be writing requests since we only need to test writing operation. DiskSim2.0 used HP-C3323A disks. Each block contains 1,000 512-Byte sectors. DiskSim2.0 generated 10,000 sequential writing requests. Blocking factor was set to 8. The maximum number of disks for DiskSim2.0 to simulate is 15. Therefore, the number of disks we can test is from 3 (the minimum disk number for a system with Parity) to 15. Table 2 shows the number of reads and writes taken when using different disk number.

**Table 2. Number of Reads and Writes vs. Number of Disks**

<table>
<thead>
<tr>
<th>Number of Disks</th>
<th>The Old Way</th>
<th>Fast Parity Update</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of Reads</td>
<td>Number of Writes</td>
</tr>
<tr>
<td>3</td>
<td>9999</td>
<td>19996</td>
</tr>
<tr>
<td>4</td>
<td>19999</td>
<td>19996</td>
</tr>
<tr>
<td>6</td>
<td>39999</td>
<td>19996</td>
</tr>
<tr>
<td>9</td>
<td>69999</td>
<td>19994</td>
</tr>
<tr>
<td>12</td>
<td>99999</td>
<td>19996</td>
</tr>
<tr>
<td>15</td>
<td>129999</td>
<td>19994</td>
</tr>
</tbody>
</table>
Figure 33 shows that fast parity update requires less reads than the old way. Thus, fast parity update is more efficient. The number of reads required by fast parity update is a horizontal line as we expected it to be. The number of reads required by the old way is linear function of the number of disks. It increases like its linear expression: $N-2$, where $N$ is the number of disks.

![Graph showing number of reads vs. number of disks]

**Figure 33. Number of Reads vs. Number of Disks**

Table 3 shows the time data varied with the I/O requests increased. In this case, we only test writing operations too. We used 15 disks. Then we changed the I/O requests from 1,000 to 10,000 and measured the time.

**Table 3. Time vs. I/O Requests**

<table>
<thead>
<tr>
<th>I/O Requests</th>
<th>Time Required by Old Way (ms)</th>
<th>Time Required by Fast Parity Update (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>23950.15</td>
<td>14094.6</td>
</tr>
<tr>
<td>2000</td>
<td>47650.02</td>
<td>27450.02</td>
</tr>
<tr>
<td>4000</td>
<td>95691.5</td>
<td>55391.5</td>
</tr>
<tr>
<td>6000</td>
<td>143692.2</td>
<td>83414.38</td>
</tr>
<tr>
<td>8000</td>
<td>191347</td>
<td>110724.8</td>
</tr>
<tr>
<td>10000</td>
<td>238701.9</td>
<td>138479.7</td>
</tr>
</tbody>
</table>
Figure 34 illustrates that fast parity update requires less time to complete the same amount of I/O requests than the old way. Fast parity update has better performance than the old way for all comparisons.

![Graph showing comparison between old way and fast parity update](image)

**Figure 34. Time vs. I/O Requests**

### 4.4. Simulation Result Discussion

Figure 33 shows that fast parity update requires less number of disk accesses. Reducing the number of disk accesses can be reduced tear and wear or the mechanical damage of the disks. Then reliability of the storage system will increase. For a disaster recovery system, the SAN enables clients to access multiple disks and allows disks to be accessed by servers serially or concurrently. If data are accessed in serial, the processing time can be shortened by a factor of \( N \), like the number of reads.

From figure 34, we can see the performance of fast parity update is better than the old way in load. The old way requires more disk accesses. For a system using the old way, the disks have more arm movements. Then it may take more time to move the arms
to seek the right data. That could reduce the performance even though data are accessed concurrently. Therefore, fast parity update has better performance than the old way.

5. Future Work

In this subsection, we discuss how to improve this system in the future and make it even more reliable. We can increase the reliability of this system by using traffic director, stripe subset and chained-declustering with double distributed parity.

5.1. Traffic Director

With the rapid growth of the Internet and e-commerce, data centers and IT infrastructures have to handle growing data demands and have to be highly available and reliable [33]. An additional manager server is added to handle growing requests for data. It replicates connections and provides additional protection to single node disjoint failures.

In order to provide a highly available traffic management solution, a Traffic Director runs as a two-node clustered system. All connections and session information are duplicated between these two nodes. Each Traffic Director node may act as a backup node for the other. If one of the Traffic Director nodes fails, operations will be transferred to the other Traffic Director node immediately without interrupting the clients. Also, both of the Traffic Director nodes can be deployed in an active-active configuration. In this case, both Traffic Director nodes can host one or more services and each node can act as a backup node for the other at the same time. Figure 35 illustrates how Traffic Directors ensure the data access service when a single node disjoint failure occurs.
Figure 35. Traffic Directors [33]
5.2. Stripe Subset

Although our method can recover two neighboring disk failures, it still takes time to perform the data reconstruction. That degrades the performance of the system. Multiple failures in a stripe could cause unrecoverable data loss. Using stripe groups containing a subset of the storage servers has several advantages over using stripe groups containing all storage servers [18].

First, fragment reconstruction involves fewer storage servers. In the event of a storage server failure, fewer storage servers may be affected. Therefore, the impact of failure on system performance is reduced.

Second, using stripe groups containing a subset of the storage servers makes it possible for the system to tolerate multiple server failures as long as two neighboring failures do not occur in the same stripe group. This is the most important advantage. If all stripes were striped across all servers, multiple server failures would result in lost data.

5.3. Chained-Declustering with Double Distributed Parity

RAID 5+1 is a combination of RAID 5 and RAID 1 [21]. It provides high availability and reliability. RAID 5+1 can recover from three adjacent failures. For example, servers 0, 1 and 2 fail, server 2 can be recovered from server 3 because they are mirrored. After that, the mirrored pair of servers 0 and 1 also can be recovered by using parity. Figure 36 illustrates the architecture of RAID 5+1.
We can turn our backup system into chained-declustering with double distributed parity by using an additional parity in each stripe. We divide the disks into two subsets. One consists of even-numbered disks. The other consists of odd-numbered disks. Then each stripe is also divided into two stripe subsets and each stripe subset has its own parity. Such that, three adjacent failures can be recovered like RAID 5+1.

When three neighboring servers fail at the same time, for example, servers 0, 1 and 2 fails at the same time, we can recover server 1 by using parity in subset 2. Then the problem of three adjacent failures becomes nonadjacent multi-failures. Therefore, we can recover servers 0 and 2 from their backup copies. Chained-Declustering with double distributed parity will provide a more load-balancing storage than RAID 5+1. Figure 37 illustrates chained-declustering with double distributed parity.
6. Conclusion

This project designs a reliable and secure data backup system. Two primary techniques provide improvements in the design. The first one is fast update parity. The second, to provide more protection to data, is chained-declustering with distributed parity data placement. We use an additional parity in chained-declustering to reconstruct the missing data when two adjacent storage devices fail at the same time. The simulation results show that the proposed backup method can back up, restore and retrieve data correctly, and can successfully recover missing data from neighboring failures. The DiskSim2.0 experimental data also shows that fast parity update reduces the number of disk accesses and has better performance than parity computation in the old way.
References


http://articles.findarticles.com/p/articles/mi_m0BRZ/is_12_23/ai_112800715


