A project Report on

Redundant Array of Independent Disks

Joshi Dnyanesh Madhav	MT2013061
Duggineni V Krishna Chaitanya	MT2013050
Pillalamarri Ramanana	MT2013107
Amrutha Muralidharan Nair	MT2013009

-Author: Duggineni V Krishna Chaitanya

INDEX

1. Problem Background

RAID is a storage technology that combines multiple disk drive components into a logical unit for the purposes of data redundancy and performance improvement. Data is distributed across the drives in one of several ways, referred to as RAID levels, depending on the specific level of redundancy and performance required.

The term "RAID" was first used by David Patterson, Garth A. Gibson, and Randy Katz at the University of California, Berkeley in 1987, standing for **redundant array of inexpensive disks**. Industry RAID manufacturers later tended to interpret the acronym as standing for **redundant array of independent disks**.

RAID is now used as an umbrella term for computer data storage schemes that can divide and replicate data among multiple physical drives: RAID is an example of storage virtualization and the array can be accessed by the operating system as one single drive. The different schemes or architectures are named by the word RAID followed by a number (e.g. RAID 0, RAID 1). Each scheme provides a different balance between the key goals: reliability and availability, performance and capacity. RAID levels greater than RAID 0 provide protection against unrecoverable (sector) read errors, as well as whole disk failure.

Raid Types :

RAID 0 :

RAID 0 comprises striping (but no parity or mirroring). This level provides no data redundancy nor fault tolerance, but improves performance through parallelism of read and write operations across multiple drives. RAID 0 has no error detection mechanism, so the failure of one disk causes the loss

of all data on the array.

RAID 1

RAID 1 comprises mirroring (without parity or striping). Data are written identically to two (or more) drives, thereby producing a "mirrored set". The read request is serviced by any of the drives containing the requested data. This can improve performance if data is read from the disk with the least seek latency and rotational latency. Conversely, write performance can be degraded because all drives must be updated. Thus the write performance is determined by the slowest drive. The array continues to operate as long as at least one drive is functioning.

Raid 10

Raid 10 comprises of 2 levels. Striping at level 1 and mirroring each of the striped file at level two. i.e Raid 0 at level 1 and Raid 1 on the resultant files at level 2.

Raid 01

Raid 01 also contains 2 levels, mirroring i.e Raid 1 at level 1 and Striping i.e Raid 0 on the resultant files at level 2.

2. Problem Statement

Improving reliability and performance of disk based data storage systems

3. Proposal Statement (Goals)

One solution to the problem is Redundant Array of Independent Disks (RAID), which is a category of disk drives that employs two or more physical disks in combination to improve reliability and/or performance. Improvent in reliability is achieved via redundancy and improvement in performance is achived via parallelism.

Reliability Improvement: In a traditional single-physical-disk system, if the physical disk fails, data stored on it may be lost. However, in a storage system if there are two physical disks which are copies of each other, then the probability that both of them will fail simultaneously is much lower than the probability that one of them will fail. Also, the probability that the good disk will fail while the failed disk is being repaired is very low. This is how reliability is improved in RAID via redundancy.

Performance Improvement: In a traditional single-physical-disk system, all data is written to and read from the only physical disk available. However, in a storage system if data is spilt (data striping) across multiple physical disks, then read/write performace is improved via parallelism thus achieved.

It is possible to achieve both reliability and performance improvement simultaneously in RAID by implementing various disk configurations.

RAID Implementation

This project aims to implement (simulate) RAID 0, RAID 1, RAID 0+1 and RAID 1+0 using USB flash drives as disks.

4. Brief Outline of Implementation plan

4.1 Modules

1) **Writing a File to a Raid drive :**

Functionality :
User creates a new file and edits the contents of it. When he tries to save this file, it gets mirrored or striped according to the Raid mechanism we have selected. For Raid 0 the File contents are split into two seperate files and these two are stored in two seperate location with two concurrent threads one for each location. Same goes for Raid 1 except for the fact that instead of Striping the file we actually mirror it and store these two copies in two seperate locations. For Raid 01, the file is mirrored first into two copies and each of these copies is split into two more files (striping) and the resultant 4 files are kept at four seperate locations. For Raid 10, the file is striped or split into two seperate files and each of these striped versions is mirrored resulting 4 seperate files. These 4 files are again kept at 4 different locations with the help of threads

Features
Usage of threads to store files in multiple locations helps reducing the time required to store these files.

2) **Reading a File from a Raid drive :**

Functionality
User enters the Raid type of the file which he wants to open. GUI will then show the disks that have that particular Raid mechanism. After selecting the disk, GUI will show the files on the disk. User can select one such file and try to open it. This opening mechanism is Raid type dependent. For Raid 0, the two striped files are read by two seperate threads and brought to main memory where they will be combined to form the original file. For Raid 1, either of the two files can be used to display the main file. For Raid 10, one of the mirrored copies from the Raid 1 mechanism at the bottom is selected and moved up. So, the two striped versions of the file are combined and level 1 to form the original file. For Raid either of the two mirrored copies of the files that are split at level 2 can be combined to form the original file.

Features
a) Due to the use of multiple threads to read from various locations the reading is much faster.
b) Because of the redundency of the disks, there will be high availability of the file.

3) Additional functions:

1) **Restor**e : After selecting the raid type and disk user can restore the files on this disk if possible. Once again, the ability to restore/recover a disk depends on the recovery mechanism that we use.
For Raid 0 if one of the striped files is lost, we won't be able to reconstruct the actual file.

So, Disk recovery for Raid 0 drives is not possible.
For Raid 1, if one of the mirrored versions is lost, the original file can still be created and the other copy can be once again created.
For Raid 10 if any of the mirrored versions of the same striped split of the actual file is lost, it can be recreated. But, if two files are lost in either of the striped versions, construction of the actual file is impossible.
For Raid 01, suppose file1, file2,file3 and file4 are the lower level raid 0 copies respectively. So file 1 and file 3 are mirrors and file 2 and file 4 are mirrors. So if wither one of the is lost, they can be reconstructed again. But if both file1 and file3 or both file2 and file4 are lost, reconstruction is impossible.

2) Save and Refresh :
After displaying the contents of the actual file, application allows user to modify the existing conents. The updated contents and be saved according to the corresponding raid mechanism by This save and refresh button. This Function also helps the user to repopulate any missing file if possible(According to raid mechanism).

3) Log Maintance :
The application consists of a log that is automatically maintained. This log stores details such as when a file was created and what at what drive. It also keeps track of modifications of a file and recovery of a disk.

4) **Graphical User Interfaces for User and Integration:**
This module contains the design of the following windows
a) **Raid Selection**
Allows the user to select a Raid type and then allows him to select one of the disks that is dedicated to that raid type. After this selection happens user can choose to open an existing file from the drop down list of files on that disk, he can choose to create a new file or he can choose to restore the disk that he has choosen.
b) **Simple raid Display**
Used for displaying Raid 0 and Raid 1 files. Associated with Save and refresh function.
c) **Complex Raid Display**
Used for displaying Raid 10 and Raid 01 files. Also associated with Save and refresh fucntion.
After this design integaration of all the existing modules into the GUI had taken place.

4.2 Salient features of the implementation

One of the good things about this application is that it was developed with high modularity. So, changes to a module doesn't change the working of the other module. Raid sub files are displayed in colors so that the user can understand how the data is striped or mirrored.

Multi threaded implementation of many functions proved to save much time. (For instance, run the main method inside the Raid0PerformanceTester class to see a demonstration.)

Several Exception classes are added in the code. They can help maintain the code in a much better way.

RAID Implementation

5. Testing Plan and Implementation of Testing

While integrating the modules into the GUI each of the modules is unit tested with temporary drivers and after integration of each module into the GUI, Integration testing with several test cases to avoid flaws.

6. Scope Of the Project

Current application simulates the way in which a file can be redundantly stored in various locations thereby increasing reliability and performance.
Functions inside this code can also be added by programmers to store their files in a more reliable fashion.

7. Limitations

One of the main disadvantages with redundant storage is that the amount of storage space that we use for 1 file doubles or triples (may be even more as the number of disks increases).

8) References

1) http://en.wikipedia.org/wiki/RAID
2) http://www.cs.cmu.edu/~garth/RAIDpaper/Patterson88.pdf
3) docs.oracle.com/**java**se/tutorial/ui**swing**/