DLSU Research Congress 2024
De La Salle University, Manila, Philippines
June 20 to 22, 2024

A Centralized Virtual Storage Management System for Distributed Heterogeneous Storage Devices

Patricia Ysabel Flores[1], Renji David Ong[1], Aaron Wy[1], and Fritz Kevin Flores[1]
[1] *De La Salle University - Advanced Research Institute for Informatics, Computing, and Networking Center for Networking and Information Security*
**Corresponding Author: fritz.flores@dlsu.edu.ph*

Abstract: Current existing storage solutions for aggregated file storage applications require the use of proprietary systems, as well as a homogenous file system across all storage devices. This study investigates the possibility to design a centralized virtual storage management system for distributed heterogeneous storage devices. The system was verified through functionality, performance, and stress testing, which include upload and download completion times, unreachable node scenarios, partially available files, configuring RAID on a file, load balancing, and user interface functionality. CVSMS consistently demonstrated the ability to carry out the described functions for both upload and download operations. The capability to configure file-based RAID on includes RAID 0 which is striping, RAID 1 which is mirroring and pRAID which is parity. In terms of load balancing, the system exhibits the capability to distribute files across nodes with larger storage capacities, effectively achieving an optimal resource allocation. For CVSMS, it is observed that there are trends on the completion time of both the upload and download functionalities, however the completion time will be affected by the following factors: the network speed, application processing speed, data reassembly or splitting, file size, and nature of the RAID. Hence, it is concluded that CVSMS shows some potential in relation to developing a centralized virtual management system.

Key Words: centralized virtual storage, distributed storage aggregation, RAID

1. INTRODUCTION

Open-source aggregated storage solutions such as the TrueNAS Project, have been developed to allow the capabilities of NAS to be more accessible to everyone, however TrueNAS is built as an operating system, hence would need users to have a dedicated computer to serve as the storage management device of their storage disks. This may introduce certain limitations for environments and users who are unable to dedicate machines to function as solely a NAS device.

These proprietary applications, despite being fully functional, have certain limitations such as the possible existence of vendor lock-out, decreasing the reliability and customizability of the platform, and some would even require that storage disks are homogeneous. These limitations may hinder typical home users as well as low-budget organizations from being able to maximize their existing storage devices unless they subscribe to cloud storage services or purchase additional expensive storage devices.

This presents an opportunity to develop a storage management solution that would not require a dedicated machine but would instead be implemented as an application for an existing operating system and would still be able to perform similar functions as that of a NAS device or even

provide improvements beyond the traditional capabilities of a NAS. Some of the functional improvements in the storage application may include the capability to aggregate storage from devices of different vendors for more flexible expandability, or to centralize the management of multiple nonidentical storage devices located on different networks. These opportunities for development would prove to be beneficial to certain use cases, as storage devices may be heterogeneous but would still have the capability to function as a single storage and storage device hosts can remain usable for other applications to function.

Current existing storage solutions with advanced capabilities and functionalities require the use of proprietary systems, a homogenous file system across all storage devices, as well as a dedicated machine running the storage management operating system. This presents an opportunity to design a storage management scheme as an accompanying storage management software to allow a flexible and customizable

storage management capability. The objective of this study is to design and implement a Centralized Virtual Storage Management System that is accessible via a network, with the functionalities of data aggregation, data redundancy, data archival and storage prioritization.

This study aims to design a virtual storage management scheme for managing heterogenous storage devices in a network; implement a virtual storage management system that is capable of supporting heterogeneous storage device in a network.; and evaluate the storage management system.

The design of the virtual storage management scheme uses a database or dictionary file to track devices, storage configurations, file hash values, metadata, and device contents. It may include functionalities like virtual RAID, data redundancy, versioning, and snapshots. The centralized storage design includes a centralized server for information on all devices and their contents. The study does not use a separate file system, this would mean that the entirety of the disk drive may not be used by the system, and only a portion of the disk may be allocated as part of the virtual storage used by the system.

The virtual storage management system will be implemented using Python as a cross-platform software application. It follows a star-topology topology, with a centralized server for managing files and storage devices. The system will have two types: a server application on a web server and a client application on devices with storage drives. The client application will be tested on Windows, Ubuntu, and Raspberry Pi OS, while the server application will be deployed on an Ubuntu Server machine. The study will use at least five machines with each storage device. The virtual storage management system will support scalability, error handling, and fault tolerance, addressing issues such as device disconnections, corrupted files, manual creation, modification, or deletion, and the addition of new storage devices. The system will automatically store and create archives of files, incorporating file management, balancing, archival, backup, and redundancy.

The evaluation of the proposed storage management system would be based on gathered performance metrics, functionality testing and interface functionality testing.

Tests would include a comprehensive examination of the capabilities of the system as well as their performance of the different functionalities present. A virtual environment would be used for testing and evaluation of the performance of the system with tests such as the stress test as well as functionality test.

2. METHODOLOGY

The development can be summarized into three phases: The first phase is gathering research topics regarding the usage of RAID, data aggregation, redundancy, backup, file systems, distributed file systems and security. Second is gathering data and surveys on different types of physical storage: HDD, SDD, USB Flash Drives, and SD Card. At this stage, the group will look for current market solutions and find ways to improve or add functionality to these solutions. Lastly, the software implementation and testing phase. This would be the culmination of the gathered data and research from the first two phases.

The first phase is understanding how data aggregation, redundancy, back up and RAID are being used in different types of scenario and systems. In addition to this, researching additional information such as file systems, distributed file systems and different types of security: hashing, encryption, salting, etc. will prove to be beneficial to the group, as these are the some of the functionalities that the group is trying to integrate at the proposed software.

The second phase will prioritize research on different physical mediums to have a better understanding of how data is physically stored. The group will also be required to research and understand how proprietary storage solutions in the market operate.

In the implementation phase, knowledge obtained from the first two phases will be integrated into a system that will be developed by the group. This phase would include additional research on knowledge gaps from the previous phases, development, consultation, testing, debugging, and documentation. The tests during this phase may include data reading, data encryption, storage balancing, and if data is accessible on different devices.

2.1 System Architecture

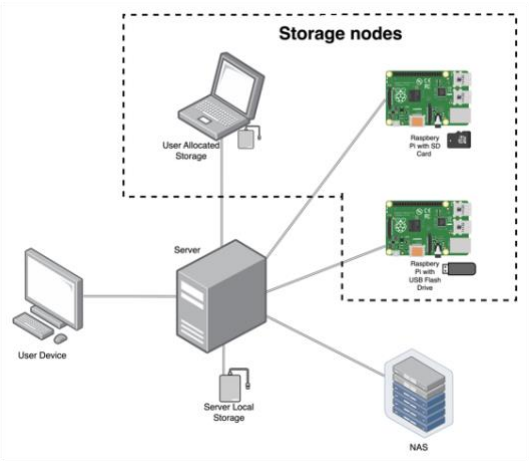

Fig. 1. System Architecture

The main entities of CVSMS are going to be a Server, User, Storage. The Server is the one responsible for doing the following functionalities: data aggregation, data redundancy, data archival and storage prioritization. Additionally, the Server is the one tasked to keep track of the changes made to the metadata on each Storage Nodes and hosts the Web Interface for the users to interact with the system. The next entity is the User which interacts with the system, where they may upload and delete files from the system. Further, the Storage Nodes functions as the decentralized storage location to be accessed by the Server, which may contain the different kinds of devices such as NAS, a PC or RPI with DAS (HDD, SSD, Flash Drive etc.). Lastly, the User-Storage is a User device that allocates a portion of their local storage to be part of the storage pool.

The system process starts when the user connects to the server's IP address through a web browser, after which an HTTP request to access the UI Module to login or register is performed. If the user decides to either login or register, the UI Module forwards the data to the Account Module to either register or authorize and authenticate the user if the inputs are valid. After the user has successfully logged in or registered, the user is redirected to the file modification page where the UI Module retrieves the list of files available for the User from the Storage Modification Module. After being redirected, the User is allowed to select a variety of actions which is then forwarded to the Storage Modification Module, where all the necessary queries and edits to the metadata database occurs. Over on the Server Connection Module, data is forwarded from the Storage Modification Module containing information on all the Storage Nodes from its previous query, involved in the file modification process. If a Storage Node is involved in a file modification process, the Server Connection Module will forward a command to its Storage Connection Module where it will process the incoming message and forward the commands to the File Modification Module. After execution, a status message depending on what occurred in the File Modification Module will then be forwarded back to the Storage Connection Module. If the Storage Node is idle, the Storage Connection Module will just constantly send a heartbeat message to the Server Connection Module. However, if a Storage Node has yet to create a connection to the server, it must await the connection command from the Storage Registration Module of the Storage Node. Once the File Modification Module receives the command from the Storage Connection Module, it will execute the command and send a status message back to the Storage Connection Module.

2.2 Module Design

The system overview was modified upon implementation of the system. A Network Attached Storage (NAS) was initially included in system overview, however due to the project being implemented in a virtual cloud environment a NAS was not used. In addition to that change, the architecture of the system was also modified. Upon the implementation of the system, it was observed that there were no clear advantages to having redundant databases on both the storage node and the server. As such, it was decided that it would be more advantageous to have a centralized database for the whole system.

Server Initialization Module - For the implementation of the Account Module, the architectural design remains the same. The module functions are to bind to the port and wait for

connection and log storage nodes that are disconnected.

User Interface - For the implementation of the Account Module, the architectural design remains the same. This is where the users can access the system and have the upload, delete, and download functionality available to every user. While the RAID, archive, viewing of the user, and storage nodes are an extra feature that the admin role has.

Account Module - For the implementation of the Account Module, the architectural design remains the same. The module will check if the login credentials is valid. Once valid it will redirect the user from the login page to the home page and depending on the role of the user different homepage will be seen by the user.

Storage Modification Module- For the implementation of this module, the architectural design was modified while still retaining some features from the initially proposed architecture.

Metadata Submodule - For the implementation of this submodule, the architectural design was modified. The submodule was proposed to have function that would synchronize its database to the storage nodes, however due to the change in the architecture mentioned in the system overview, this will no longer be necessary. The metadata of each storage node will instead be stored in a single database on the server which would be accessed and modified solely by this submodule.

RAID Submodule - For the implementation of this submodule, the architectural design remains the same. There were no issues that were encountered upon the submodule's implementation.

Local Storage Submodule - For the implementation of this submodule, the architectural design was modified. In the initially proposed architectural design, the 5 submodules will automatically archive files after a certain period, this function was removed and the submodule will no longer have any form of automatic archiving, and instead any file that will be archived will have to be manually initiated by an administrator.

Server Connection Module - For the implementation of the Account Module, the architectural design remains the same. The module function is to accept and maintain connections from the storage nodes. This module is also expected to receive and pass commands of upload and download to the storage nodes.

File Modification Module - For the implementation of this module, the architectural design was modified. Due to the changes mentioned in the system overview, modification of the database on storage nodes will no longer be necessary, as such the "delete" function on this module was removed. The module will still, however, send updates to the server regarding the status of the file transfer and file retrieval process.

Storage Registration Module - For the implementation of the Account Module, the architectural design remains the same. The module will load all of the registration details from the config file and send it to the server that would act as the registration and once the registration is completed the heartbeat would follow to signify the continuous connection with the server.

3. RESULTS AND DISCUSSION

For the following tests, Table 1 would be used as a reference for the partitions contents and would be used as a basis for the performance testing to better visualize how the file is being stored and allocated in storage nodes as well as to demonstrate the comparison between the different RAID types.

Table 1. File Partition Table

RAID	Partition 1	Partition 2	Partition 3	Total Size
None	100%	-	-	100%
RAID0	50%	50%	-	100%
RAID1	100%	100%	-	200%
pRAID	50%	50%	50%	100%

The Upload Tests focuses on the time it takes for the system to upload files of different file sizes. The results are presented in Table 2 and Table 3.

Table 2. Upload Test Results

Description	Trial 1 (sec)	Trial 2 (sec)	Trial 3 (sec)	Mean (sec)
Upload a 10 KB file	2.04	2.02	2.04	2.03
Upload a 10 MB file	3.03	3.03	3.03	3.03
Upload a 50 MB file	4.02	4.02	4.03	4.03
Upload a 100 MB file	7.06	6.12	7.05	6.75
Upload a 500 MB file	23.36	21.36	22.38	22.37

Table 3. File Size Scaling Upload Test Results

Description	Trial 1 (sec)	Trial 2 (sec)	Trial 3 (sec)	Mean (sec)
Upload a 100 MB file	6.06	6.04	6.05	6.05
Upload a 200 MB file	8.05	9.06	9.04	8.72
Upload a 300 MB file	15.23	15.07	14.06	14.79
Upload a 400 MB file	17.32	18.08	18.09	17.83
Upload a 500 MB file	21.08	23.09	22.37	22.18

As can be observed from Table 2, despite the smallest file size to be tested is 10KB, having significant size different from 50MB, the mean upload time yielded a difference of around 1 second, which may be attributed to the fact that once an upload command is issued, the internal processing of the system incurs a delay of around 2 seconds before an action would be performed. It is also evident in the results comparing the increasing file size upload, that there is a trend of around 25MB/s per upload, added 2 seconds for processing, creating a function to predict the general amount of time an upload file would take.

In Table 3, it can be observed that there is an increasing upload completion time in relation to the file size and that there is linear growth for files as they are incremented from 100 MB to 500 MB. The result also closely follows the previously mentioned upload function of 2s+25MB/s, which generally true for all tests, except for the 200MB test, which may be incidental which is caused by sudden decreases in the read and write processes of the storage devices as part of the general operating system functions.

The Download Tests focuses on the time it takes for the system to retrieve a file of different file sizes and prepare the file for downloading. The results are presented in Table 4 and Table 5.

Table 4. Download Test Results

Description	Trial 1 (sec)	Trial 2 (sec)	Trial 3 (sec)	Mean (sec)
Download a 10 KB file	2.03	2.03	2.02	2.03
Download a 10 MB file	3.02	3.03	3.03	3.03
Download a 50 MB file	4.03	4.03	4.03	4.03
Download a 100 MB file	5.07	5.04	5.04	5.05
Download a 500 MB file	23.41	22.35	23.39	23.05

Table 5. File Size Scaling Download Test Results

Description	Trial 1 (sec)	Trial 2 (sec)	Trial 3 (sec)	Mean (sec)
Download a 100 MB file	6.11	6.11	6.11	6.11
Download a 200 MB file	9.04	9.04	9.04	9.04
Download a 300 MB file	15.09	15.07	13.07	14.41
Download a 400 MB file	18.28	19.30	17.31	18.30
Download a 500 MB file	23.07	22.07	22.07	22.40

As can be observed from Table 4 and Table 5, there is an increasing download completion time in relation to the file size and that there is linear growth for files from 100 MB to 500 MB. Interestingly enough, the trend that is observed during the Upload Test in Table 2 and Table 3, also shows that the upload function delay of 2s+25MB/s also holds true with the Download Test. This may be attributed to the fact that how the system functions is that the server, first retrieves the file from the storage devices, before forwarding them to the requestors, effectively performing both download (storage devices to the server) and upload (server to the requestors) instructions, hence both tests yielding very similar results.

The next tests would be the Redundancy and RAID Tests which focuses on the time it takes for the system to complete a specific RAID configuration on files of different file sizes. The results are presented in Table 6 and Table 7.

Table 6. Redundancy and RAID Test Results

Description	Trial 1 (sec)	Trial 2 (sec)	Trial 3 (sec)	Mean (sec)	Mean (MB/s)
RAID 0 – 50MB file	7.17	7.16	7.15	7.16	6.98
RAID 0 – 100MB file	9.33	9.25	9.32	9.30	10.75
RAID 0 – 500 MB file	19.88	20.85	20.97	20.57	24.31
RAID 1 – 50MB file	9.10	9.11	9.120	9.11	5.49
RAID 1 – 100MB file	12.15	12.22	12.12	12.16	8.22
RAID 1 – 500 MB file	42.59	38.25	38.38	39.74	12.58
pRAID – 50MB file	11.32	11.31	11.36	11.33	4.41
pRAID – 100MB file	12.47	13.41	13.43	13.10	7.63
pRAID – 500 MB file	33.46	33.54	31.34	32.78	15.25

Table 6. Redundancy and RAID Test Results

Description	Trial 1 (sec)	Trial 2 (sec)	Trial 3 (sec)	Mean (sec)	Mean (MB/s)
RAID 0 – 50MB file	7.17	7.16	7.15	7.16	6.98
RAID 0 – 100MB file	9.33	9.25	9.32	9.30	10.75
RAID 0 – 500 MB file	19.88	20.85	20.97	20.57	24.31
RAID 1 – 50MB file	9.10	9.11	9.120	9.11	5.49
RAID 1 – 100MB file	12.15	12.22	12.12	12.16	8.22
RAID 1 – 500 MB file	42.59	38.25	38.38	39.74	12.58
pRAID – 50MB file	11.32	11.31	11.36	11.33	4.41
pRAID – 100MB file	12.47	13.41	13.43	13.10	7.63
pRAID – 500 MB file	33.46	33.54	31.34	32.78	15.25

As can be observed from Table 6 and Table 7, RAID 0 has the fastest process among the 3 RAID configurations, which is logical as the nature of how RAID 0 functions as presented in Table 1, is that RAID 0 would use the least total amount of storage, with 2 storage devices only containing 50% of the data, as opposed to the other RAID configurations. However comparing RAID 1 and the proposed pRAID configuration, it may be observed that RAID 1 performed slightly faster on the 50MB test, but worse on the 500MB. This would mean that as the file size would increase further than 100MB, the proposed pRAID would perform faster than RAID 1.

4. CONCLUSIONS

Efficient utilization of existing storage devices is becoming essential as people generate more and more data. These issues may be solved through the use of expensive proprietary storage. With the increasing capabilities of computers, excess resources may be used to facilitate the sharing of additional storage capacity. This study proposed and designed a system that would utilize excess storage capacity of devices in a network. An application capable of storing files, load balancing, placing files in a RAID configuration, and archiving, all of which can be accessed through a web browser.

Based on the results from the performance tests, it is observed that there is a linear and proportional trend in relation to the increasing file size concerning the completion time of both the upload and download functionalities, however the completion time will be affected by the following factors: the network speed, application processing speed, data reassembly or splitting, file size, and nature of the RAID. Hence, it is concluded that CVSMS shows some potential in relation to developing a centralized virtual management system.

It is recommended for future research endeavors to investigate and integrate queuing systems during the planning of the architecture. The implementation of the RAID function may be improved by using a configurable setting that would automatically place all the files being uploaded in a pre-determined RAID configuration. Lastly, the usage of Asynchronous JavaScript and XML (AJAX) to improve the responsiveness of the file, storage nodes and file status in RAID format.

6. REFERENCES

W. Li, C. Feng, K. Yu and D. Zhao, "MISS-D: A fast and scalable framework of medical image storage service based on distributed file system," Computer Methods and Programs in Biomedicine, vol. 186, p. 105189, 2020.

H. Huang, J. Lin, B. Zheng, Z. Zheng and J. Bian, "When Blockchain Meets Distributed File Systems: An Overview, Challenges, and Open Issues," 2020.

M. Nakagami, J. Fortes and S. Yamaguchi, "Performance Improvement of Hadoop ext4-based Disk I/O," 2020 Eighth International Symposium on Computing and Networking (CANDAR), pp. 181-187, 2020.

L. Sudha Rani, L. Sudhakar and S. Vinay Kumar, "Distributed File Systems A Survey," 2014.

J. Dongo, Y. Atik, C. Mahmoudi and F. Mourlin, "Distributed File System for NDN: an IoT Application," 2018.

H. Rahman, N. Ahmed and I. Hussain, "Comparison of Data Aggregation Techniques In Internet of Things (IoT)," IEEE WiSPNET 2016 conference, pp. 1296-1300, 2022.