



MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DISTRIBUTED SYSTEMS (CSE -3261) MINI PROJECT REPORT ON

Distributed System implementing File Sharding

SUBMITTED TO

Department of Computer Science & Engineering

by

Swathi Mohan - 200905004

Chitra A C - 200905016

Sanjana Ganesh Nayak - 200905022

Sakshi Sharma - 200905109

Dr. Venkatesh A Bhandage

Name & Signature of Evaluator 1

Ms. Anita Kini

Name & Signature of Evaluator 2

(Jan 2023 – May 2023)

Table of Contents			
			Page No
Chapter 1	INTRODUCTION		
	1.1	Distributed File System	2
	1.2	File Sharding	2
	1.3	Features of File sharding in distributed file system	4
Chapter 2	METHODOLOGY		5
	2.1	Methodology	
Chapter 3	RESULTS AND DISCUSSION		6
	3.1	Result	
Chapter 4	CONCLUSIONS AND FUTURE ENHANCEMENTS		10
	4.1	Conclusions	10
	4.2	Future enhancements	10
REFERENCES			

1. Introduction

1.1 Distributed File Systems

A distributed file system is a form of file system that enables several nodes, even those that are not physically close to one another, to share and access the same data and storage devices. In a distributed file system, files are kept on numerous servers or nodes, and information is sent from one to the other through a network.

Scalability, availability, and performance can all be enhanced with distributed file systems. They can support more users and handle bigger data quantities than conventional file systems since files are distributed across numerous servers. They can also offer redundancy and fault tolerance so that data is still accessible even if one or more servers malfunction. All of these systems share the underlying objective of offering scalable and dependable file storage and access in dispersed contexts, though each has its own distinct features and abilities.

Distributed file systems are utilized in a variety of applications, from cloud computing to scientific research, and are a vital tool for managing vast volumes of data in current computing environments.

1.2 File sharding

A technique used in database architecture to increase performance and scalability is data sharding. It involves shredding a big database into more manageable chunks known as shards, which can be kept across numerous servers or nodes. The goal of data sharding is to divide a database's burden among several servers so that each one only needs to manage a fraction of the total data. As a result, performance and scalability may improve and the processing and storage load on individual servers may be lessened.

File sharding can be done in a variety of ways, including vertical and horizontal sharding. Vertical sharding divides a database into segments based on a column or attribute, with each segment containing a subset of the original table's columns. On the other hand, horizontal sharding entails dividing a database into subsets of rows, with each shard containing a portion of the original table's rows.

Making sure that data is evenly distributed and duplicated among the shards to prevent data loss or inconsistencies is one of the problems of data sharding. Consistent hashing and distributed transactions are two methods that have been developed to overcome issues. Overall, data

sharding can be an effective strategy for increasing the speed and scalability of large databases, especially in contexts with high traffic or volume.

1.3 Features

- **Authentication and Security:** Authentication is a crucial part of distributed systems security since it guards against malicious attacks and helps prevent unauthorized access. To access the file system, the user must first log in. The implementation of MD5 encryption is used to encrypt the password.
- **File Sharding:** File sharding, also referred to as file partitioning, is a distributed systems approach that divides huge files into smaller chunks, or shards, and stores them on various nodes around the system. The performance, scalability, and fault tolerance of this method are all improved. Large files might be challenging to manage and move across the network in a distributed file system. By breaking up huge files into smaller, easier-to-manage chunks that may be kept on various nodes around the system, file sharding provides a solution to this issue. Multiple shards can be kept on a single node, and each shard normally has a defined size.
- **Scalability:** Scalability is a system's capacity to manage growing workloads. Since distributed systems are made to be scalable, an increase in the number of users, requests, or data can be handled without negatively affecting performance or dependability. In order to accommodate growing workloads, distributed systems can scale horizontally by adding more nodes to the system. This contrasts with conventional centralized systems, which may demand very pricey improvements in order to scale vertically.
- **Decentralization:** Since there is no centralized authority or single point of control in distributed systems, they are decentralized. As a result, they become more versatile and adaptable to changing requirements, as well as more resistant to attacks and failures.
- **Availability:** High availability is what distributed systems are built to offer. Even in the event of failures or network outages, they guarantee that consumers always have access to services and data. This is accomplished by replicating data and services across numerous nodes and distributing workloads using methods like load balancing.
- **Replication:** Replication is the process of copying data or services across multiple nodes. Even if some nodes fail, distributed systems can employ replication to make sure that data and services are always accessible.

2. METHODOLOGY

The methodology for file sharding in a distributed system involves several steps that must be followed to ensure the proper implementation and functioning of the sharding technique. These steps include:

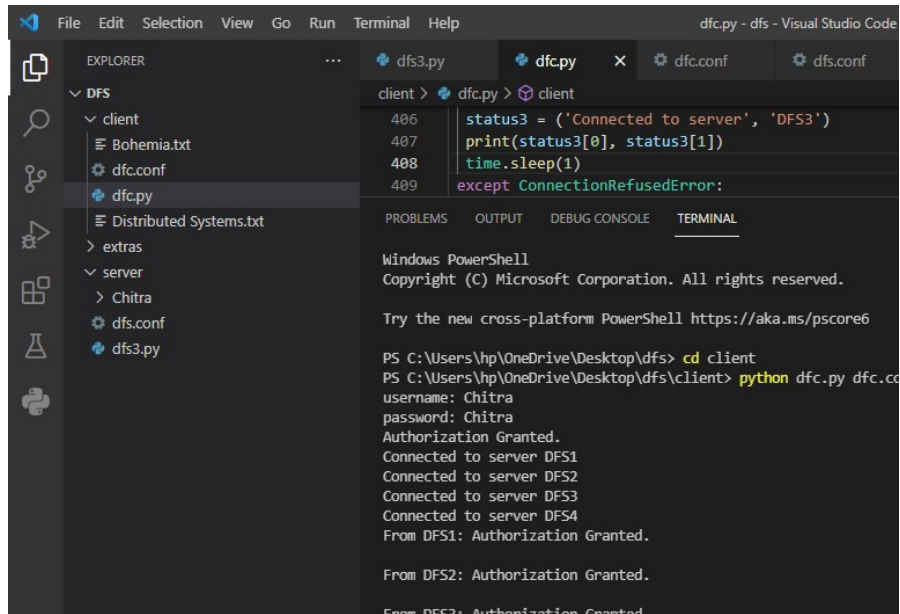
1. **Analyze the file to be sharded:** The first step is to analyze the file to be sharded and determine its size, type, and any other relevant characteristics. This analysis will help determine the appropriate size of the shards and the number of shards required to store the file. In this case, we should ensure that the file that has to be sharded is purely a text file, with no images, hyperlink, videos or gifs.
2. **Determine the sharding algorithm:** Once the file has been analyzed, the next step is to determine the sharding algorithm to be used. There are several algorithms available for sharding files, including round-robin, hash-based, and range-based algorithms. In this case we are using a *hash based sharding algorithm* which is called *MD5 (Message Digest File-128bit)*
3. **Partition the file:** After the sharding algorithm has been selected, the file can be partitioned into shards based on the algorithm. Each shard should be a fixed size to simplify management and replication.
4. **Replicate the shards:** To ensure fault tolerance, each shard should be replicated on multiple servers. In this case, each chunk is replicated across two servers.
5. **Assign shards to servers:** Once the file has been partitioned into shards, the next step is to assign the shards to servers within the distributed system. In our mini project, we are using 4 servers, and 1 client. A huge text file is broken down into multiple smaller parts, and each part of the text file is stored across a server.
6. **Ensure consistency:** To ensure that all shards are consistent with each other, consistency protocols should be used. These protocols ensure that updates to one shard are reflected in all replicas of that shard. The protocol that we used is *consistent hashing*- Consistent hashing is a technique used to distribute data evenly across multiple nodes in a distributed system. Consistent hashing algorithms ensure that each server is responsible for a consistent portion of the data, which allows for efficient scaling and fault tolerance.
7. **Access and retrieve the file:** Once the file has been sharded and stored on the distributed system, it can be accessed and retrieved back by the client. Our distributed system has a mechanism for the client to request the file, which will retrieve the shards from the appropriate servers and reassemble them into the original file.

In conclusion, file sharding in a distributed system requires careful planning and implementation to ensure proper functioning and fault tolerance. The methodology involves several steps, including analyzing the file, selecting a sharding algorithm, partitioning the file, assigning shards to nodes, replicating the shards, ensuring consistency, and accessing and retrieving the file. By following this methodology, a distributed system can efficiently store and retrieve large files while maintaining fault tolerance and scalability.

3. Results and discussion

- Running the file servers and client in the respective nodes. Authorization is granted by the servers

Client:



The screenshot shows the Visual Studio Code interface with the 'client' directory selected in the Explorer. The file explorer shows 'Bohemia.txt', 'dfc.conf', 'dfc.py', 'Distributed Systems.txt', 'extras', 'server', 'Chitra', 'dfs.conf', and 'dfs3.py'. The 'dfc.py' file is open in the editor, showing the following code:

```
406 status3 = ('Connected to server', 'DFS3')
407 print(status3[0], status3[1])
408 time.sleep(1)
409 except ConnectionRefusedError:
```

The terminal output shows the execution of the client code:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

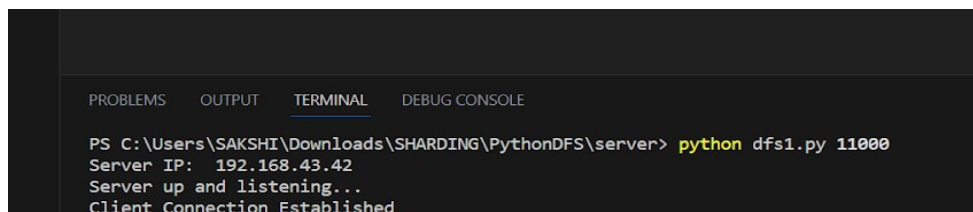
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\hp\OneDrive\Desktop\dfs> cd client
PS C:\Users\hp\OneDrive\Desktop\dfs\client> python dfc.py dfc.co
username: Chitra
password: Chitra
Authorization Granted.
Connected to server DFS1
Connected to server DFS2
Connected to server DFS3
Connected to server DFS4
From DFS1: Authorization Granted.

From DFS2: Authorization Granted.

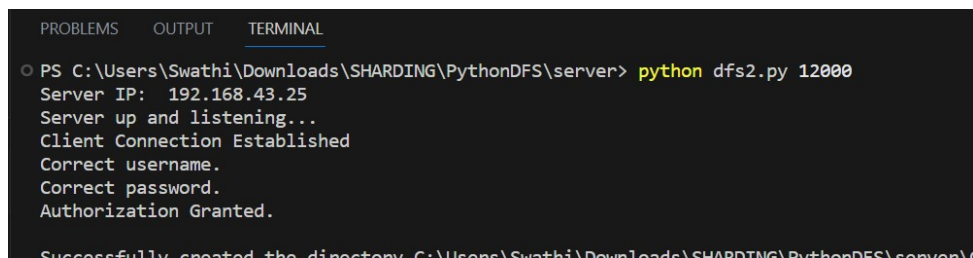
From DFS3: Authorization Granted.
```

Servers:



The screenshot shows a terminal window with the following output:

```
PS C:\Users\SAKSHI\Downloads\SHARDING\PythonDFS\server> python dfs1.py 11000
Server IP: 192.168.43.42
Server up and listening...
Client Connection Established
```



The screenshot shows a terminal window with the following output:

```
PS C:\Users\Swathi\Downloads\SHARDING\PythonDFS\server> python dfs2.py 12000
Server IP: 192.168.43.25
Server up and listening...
Client Connection Established
Correct username.
Correct password.
Authorization Granted.

Successfully created the directory C:\Users\Swathi\Downloads\SHARDING\PythonDFS\server\
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
server DFS2 hp\OneDrive\Desktop\dfs\server> python dfs3.py 13000
Server IP: 192.168.43.164
connected to Server up and listening...
server DFS3 Client Connection Established
Correct username.
connected to Correct password.
```

```
Asus@LAPTOP-L50C19C3 MINGW64 ~/miniproj/dfs/server
$ python dfs4.py 14000
Server IP: 192.168.43.166
Server up and listening...
Client Connection Established
Correct username.
Correct password.
Authorization Granted.
Successfully created the directory C:\Users\Asus\miniproj\dfs\server\Ch
```

- Uploading the file by the server

Client:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Please specify a command [download, list, upload]: upload
Current files:
-----
Bohemia
Distributed Systems

Please specify a file: Bohemia

Sending Bohemia_3.txt...

Sending Bohemia_4.txt...

Sending Bohemia_1.txt...

Sending Bohemia_2.txt...

DFS1 Chunk 1 transfer complete.
```

Servers:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Successfully created the directory C:\Users\SAKSHI\Downloads\SHARDING\PythonDFS\server\Chitra
The user requested to upload files.
The buffer size is: 104892
Receiving Bohemia_3.txt...

No symbols found in document 'dfs.conf'

Successfully created the folder C:\Users\SAKSHI\Downloads\SHARDING\PythonDFS\server\Chitra\Bohemia
Chunk 1 successfully transferred.

Receiving Bohemia 4.txt...
```

```
PROBLEMS  OUTPUT  TERMINAL

Successfully created the directory C:\Users\Swathi\Downloads\SHARDING\PythonDFS\
The user requested to upload files.
The buffer size is: 104892
Receiving Bohemia_4.txt...

Successfully created the folder C:\Users\Swathi\Downloads\SHARDING\PythonDFS\ser
Chunk 1 successfully transferred.

Receiving Bohemia_1.txt...

Chunk 2 successfully transferred.
```

```
TERMINAL

list, upload]: uplo  hemia_1.txt...

Successfully created the folder C:\Us
\Chitra\Bohemia
Chunk 1 successfully transferred.

Receiving Bohemia_2.txt...
```

```
Authorization Granted.

Successfully created the directory C:\Users\Asus\miniproj\dfs\server\Chitra
The user requested to upload files.
The buffer size is: 104892
Receiving Bohemia_2.txt...

Successfully created the folder C:\Users\Asus\miniproj\dfs\server\Chitra\Bohemia
Chunk 1 successfully transferred.

Receiving Bohemia_3.txt...

Chunk 2 successfully transferred.
```


- Listing and downloading the files by the client:

Client:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Please specify a command [download, list, upload]:

Current DFS1\Chitra files:
-----
Bohemia_3.txt
Bohemia_4.txt

Current DFS2\Chitra files:
-----
Bohemia_1.txt
Bohemia_4.txt

Current DFS3\Chitra files:
-----
Bohemia_1.txt
Bohemia_2.txt
```

Server:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Bohemia_4.txt

Current DFS3\Chitra files:
-----
Bohemia_1.txt
Bohemia_2.txt

Current DFS4\Chitra files:
-----
Bohemia_2.txt
Bohemia_3.txt

Would you like to download files, upload files, or exit?
[download, upload, exit]: download
Successfully created the directory C:\Users\hp\OneDrive\Desktop
Please specify a file: Bohemia
File chunks successfully transferred.
File chunks successfully transferred
```

4. Future enhancements:

- Populate .conf files dynamically for usernames and passwords
- Chunks stored and retrieved with every other line empty
- Adding Salt to hashing to improve password security
- Encrypting all the traffic messages
- Send other file types (images, videos, hyperlink etc.)

5. References

- <https://www.google.com/search?q=encoding%3D%27cp1252%27&oq=encoding%3D%27cp1252%27&ags=chrome..69i57.518j0j7&sourceid=chrome&ie=UTF-8>
- <https://www.simplilearn.com/tutorials/cyber-security-tutorial/md5-algorithm#:~:text=the%20MD5%20algorithm.-,What%20is%20the%20MD5%20Algorithm%3F,means%20for%20digital%20signature%20verification.>
- <https://www.geeksforgeeks.org/what-is-sharding/>