Innopolis University

Distributed Systems

Distributed File System

Nikita Ulianov, Andrey Polovinkin,
Svetlana Ostrovskaya, Oleg Surnin

November 30, 2017

1 Deployment

You can use our demo infrastructure to test the full functionality of distributed file system. The instructions how to use it are provided in the subsection 3.4.
If you want to deploy the whole system by yourself, use the following links:

- The source code is available at github. Last commit - f8b3a763d6757b4cb1c3d85e9fe79230ee77f097
- The API documentation is available at the link.
- The docker image of Nameserver is available at dockerhub.
- The docker image of Storage Server is available at dockerhub.

Also, it is needed to have an SQL database with structure mentioned in the subsection 3.1. The default one is PostgreSQL, but it is possible to use any from the SQL family.
These steps are required to deploy the whole system:

1. Create tables in the database
2. Add records of Storage Servers IP addresses to the database
3. Change database connection config in the Nameserver
4. Start Nameserver dockers:

```
# docker run -d --name ns_client -p 5000:5000 -it legiks/nameserver:master python3
    nameserver/nameserver_client.py
# docker run -d --name ns_storage -p 5010:5010 -it legiks/nameserver:master python3
    nameserver/nameserver_storage.py
```

5. Start Storage dockers:

```
# echo {} > /var/lib/docker/volumes/ds_fs_storage/_data/mapping.json
# mkdir /var/lib/docker/volumes/ds_fs_storage/_data/files/
# mkdir /var/lib/docker/volumes/ds_fs_storage/_data/files/tmp
# docker run --rm -v ds_fs_storage:/usr/src/app/storage -p 8010:8010 -p
    8020-8049:8020-8049 --name storage kekisokay/ds_fs_storage_server
```

6. Change the Nameserver IP address in the Client

2 Features

- File caching at the client:

  Client application provides a file caching. When the application launch, directory "/tmp/F-Stemp*" is created. When a client tries to open a file, system checks if needed file already exist in this directory. If so, needed file will be opened immediately. Alternatively, the file will be loaded from the server and saved in this temp directory. When the application closes, directory with all temp files is deleted.

- Concurrent access handling:

  Each write operation at Storage Server is accomplished through a single threaded blocking process for each user.

- Robustness:

  Each server keeps track of all writing operations that modifies Storage Server state that is the same on all Storage Servers in one Cluster. When the server is restoring after a crush, it asks the current Primary Server to invoke all operations that were missing during downtime.

3 Architecture

The distributed file system architecture is shown in Figure 1.

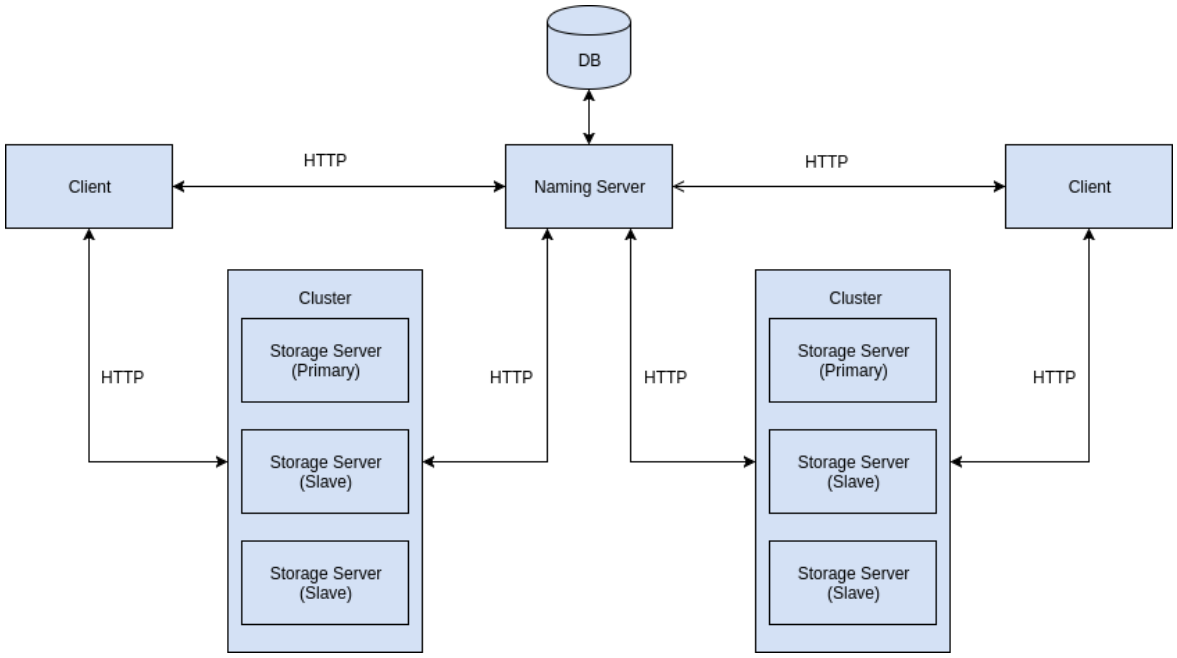

Figure 1: File system architecture

In our architecture we use only one Nameserver. To exclude the one point of failure, it is possible to run multiple name servers and add their IP addresses into the client's config. Our Storage Servers are distributed all over the world (Russia, Germany, USA). This architecture provides small latency for our customers from different countries.

3.1 Database

The PostgreSQL was chosen as the main database. In comparison with other SQL databases, it provides less hassle with licensing, custom data types, rules systems, and database events. The structure of the database is shown in Figure 2.

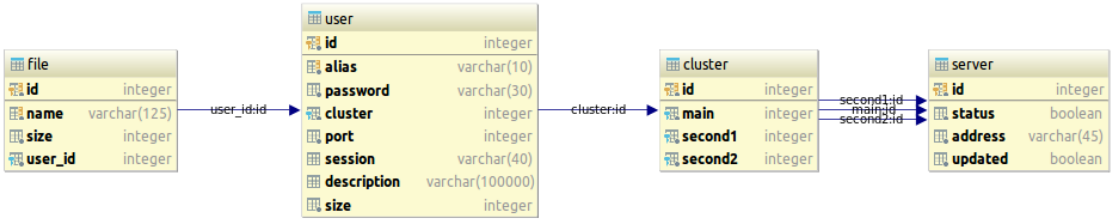

Figure 2: Database model

3.2 Nameserver

The Nameserver application consists of two parts. The first part is responsible for communication with Clients. The second one is responsible for communication with Storage Servers.

3.2.1 Nameserver client

The Client Nameserver supports the following requests:

- **/registration**. Users can register in the system.
- **/login**. Users must be logged in to perform any actions in the File System.
- **/init**. Initialization of the File System to the user.
- **/read**. Users can read their files and open directories. The response will be a list of IP addresses of the Storage Servers in JSON format. Servers will be provided in different order for users to spread their load. Only updated servers will be in the response.
- **/write**. Users can upload files to the File System. The response is the IP address of the Prime Server from the database.
- **/delete**. Users can delete files from the File System.
- **/mkdir**. Users can create directories in the File System.
- **/rmdir**. Users can delete directories from the File System.
- **/size**. Users can request the size of files and directories.

With the last five operations the Client will get a response from the Nameserver, which performed all communication with Storage Servers.

3.2.2 Nameserver storage

The Nameserver expects request on the 5010 port from the Storage Server. The Storage Nameserver has the following requests:

- **/alive**. The Nameserver expects a heartbeat request from the Storage Servers. If the Storage Servers do not send the messages during a time interval Nameserver will point out Storage as not working. The time interval is equal to 7 seconds. If the Prime Storage Server is not working Nameserver will switch to another one.

- **/submitted**. A Storage Server asks Nameserver is it allowed to do a write operation. Nameserver checks is request came from the Primary Server and that file fits into user's available space. In case when everything is fine and write operation is allowed, Nameserver returns list of Secondary Servers the new file should be replicated to.
- **/update_failed**. This request happens when the Storage Server cannot update files from the Primary Server after a crash.
- **/updated**. This request happens when the Storage Server updates files from the Primary after crash.

3.3 Storage Servers

Storage Servers are organized in Clusters. Our idea is that each user has equal limited amount of available space and each user is assigned to particular Storage Cluster inside which full replication of files is performed. This provides a fault tolerance. When one Cluster cannot fit users anymore, e.g. because of lack of the storage space, a new Cluster can be added to the system.

3.3.1 Storage Cluster structure

Each Cluster contains 3 servers that store the same data. If one server fails, there are still 2 copies available. One of the servers is Primary, on which all writes operations are performed. The rest of the servers are Secondaries that only allowed to provide files to read. When a write is performed on a Primary Server, it asks permissions from the Nameserver. It checks is this server still Primary and allowed to do the write. Also, server checks the user's storage space quota.

3.3.2 Storage Servers' Processes and Main Data objects

Storage Server has to perform several tasks: providing data to the Client and perform file writes, communicate with Nameserver and replicate data. Thus, there is a need of several concurrent processes.

Storage Writer - a process that provides API for Client, Nameserver and other Storage Servers to perform write operations. Heavy operation as writing a file is accomplished by a Client. Lightweight operations, such as creation/deletion of directories and deletion of files, are performed directly by Nameserver.

When Client invokes a file write operation on a Primary Server, the Primary Server asks permissions from the Nameserver and gets a list of Secondaries where it should replicate the file. Then, it replicates file to all Secondaries in the list.

Each user has its own Storage Writer that is associated with particular folder and port on the Storage Servers. When storage is initialized for a user, Storage Server creates Storage Writer, user specific folder, and opens user specific port.

Storage Writer is blocking single threaded process per user that can serve only one request at a time, thus the problem of concurrent writes is solved.

Write operation is represented on Figure 4.

API

- **/mkdir** - create directory
- **/rmdir** - remove directory

- **/write** - write file
- **/replicate** - replicate file
- **/delete** - delete file

Update Log - a data object that is managed by Storage Writers for each user separately and keeps track of all write operations. Write and read to Update log are performed with using of a lock that controls concurrent access. While a server restores after crash, it asks the current Prime Server use its update log to detect missing updates and invoke missing operations on the recently awaken server.

Storage Reader - a process that provides API for a Client to read data. When Storage is initialized for a user, the Storage Reader, that has a user specific folder and user specific port, is created. There are at most 10 Storage Readers on the Storage Server - one for each user.

API

- **/read** - read file
- **/lsdir** - list directory

Storage Pusher - a process that pushes new data to requesting Secondary Server. It is used when one of the replicas was inactive for some time. Storage Pusher takes the id of the last operation performed on the recently restored server and invoke all missing operations.

API

- **/push_updates** - invoke write operation on the requesting server if its version is old

Storage Updater - a process that asks the Primary Server to push missing updates. Storage Updater is blocked until new data is completely pushed and when everything is pushed, it sends a notification to the Nameserver through Storage Initializer.

Storage Initializer - a process that provides API for Nameserver to initialize user storage and other Storage Servers to retrieve missing updates. It is the parent process on Storage Server and creates other components when needed.

API

- **/init** - Nameserver use this method to create Storage Writer and Storage Reader for a new user.
- **/youneedupdate** - Nameserver signals to Storage Server that it needs an update and Storage Server instantiate Updaters
- **/create_pusher** - is used by other Storage Servers to create Storage Pusher that will update requesting server

User Mapping - a data object that is managed by Storage Initializer and contains information about what username is assigned to which Storage Writer and Storage Reader. While server restores after a crash, it asks the current Primary Server to provide its User Mapping to synchronize with it.

Storage Heartbeat - a standalone process that signals to Nameserver that Storage Server is alive.

3.3.3 Diagrams

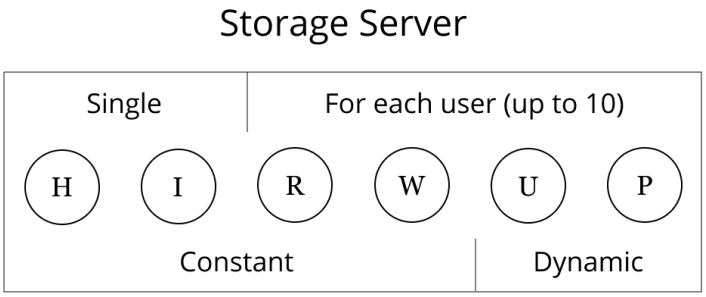

Figure 3: Storage Server processes

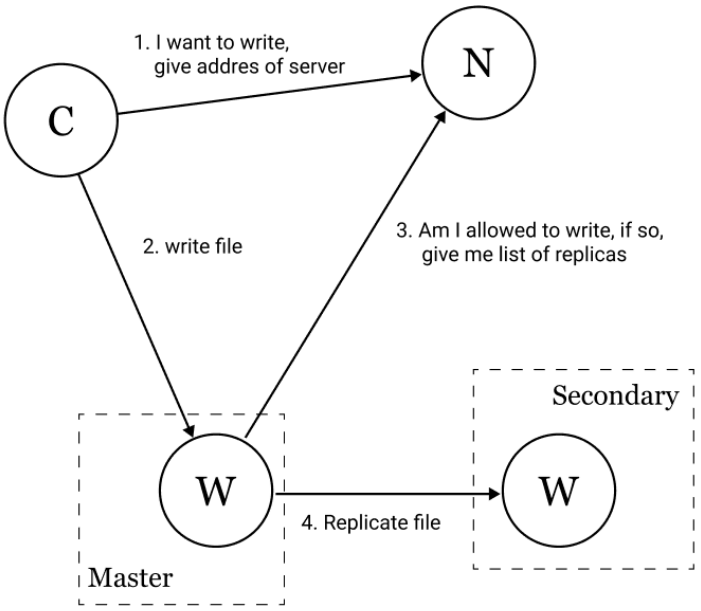

Figure 4: File write operation

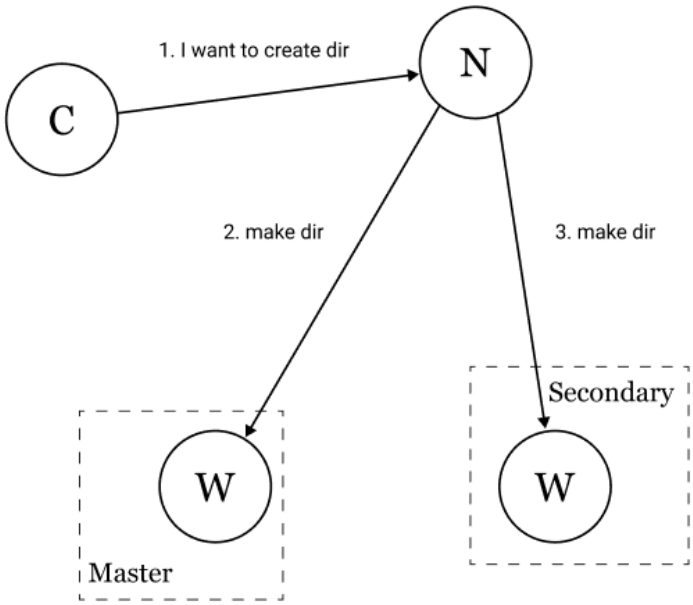

Figure 5: File deletion and directory creation/deletion operation

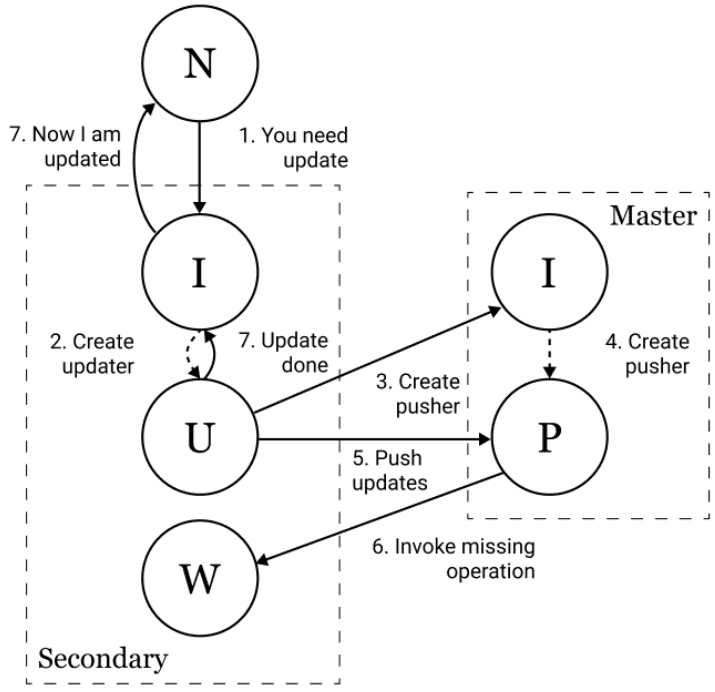

Figure 6: Updating of recently awaken Storage Server

3.4 Client

To start work with Client application script "client.sh" should be launched.

3.4.1 Registration and authorization

To work with File System user should be authorized. The authorization window provides both registration and authorization. The user has to enter his or her alias and password to get an access. In case if the user does not authorized, registration must be performed in the same way. After registration, authorization will be performed automatically and the main window will be opened.

3.4.2 General description

Main window provides several features:

- view existing files and directories
- go through directories
- upload new files
- delete files
- create new directories
- delete directories

When the main window is launched, the user can see his or her starting directory with all existing files and directories.

3.4.3 View files and directories

To open a file or directory user should double-click on the needed item in the displayed list. Also, there exist two special items: root and back. Root allows to jump to the starting directory. Back allows to return to the previous directory.

3.4.4 Upload new files and create directories

To upload a new file or create a directory user should click on the "C" icon. An opened dialog allows to enter a name of directory (note that "Directory" checkbox must be marked) or choose a file from the local directory. If entered data is correct, after pressing button "Create" dialog will be closed and new file or directory will be added.

3.4.5 Delete files or directories

To delete file or directory user should click on the "D" icon. Opened dialog will ask a confirmation. If deletion is confirmed, dialog will be closed and file or directory will be deleted.

3.4.6 Logging

When the application launch, file "FileSystem.log" is created. This file provides an information about all actions performed by the user, as well as information about system queries and responses.