Distributed Systems
Assignment #5
A Replicated Distributed File System

Mohamed Osama Azmy 51
Moustafa Mahmoud Mohamed 65

Contents

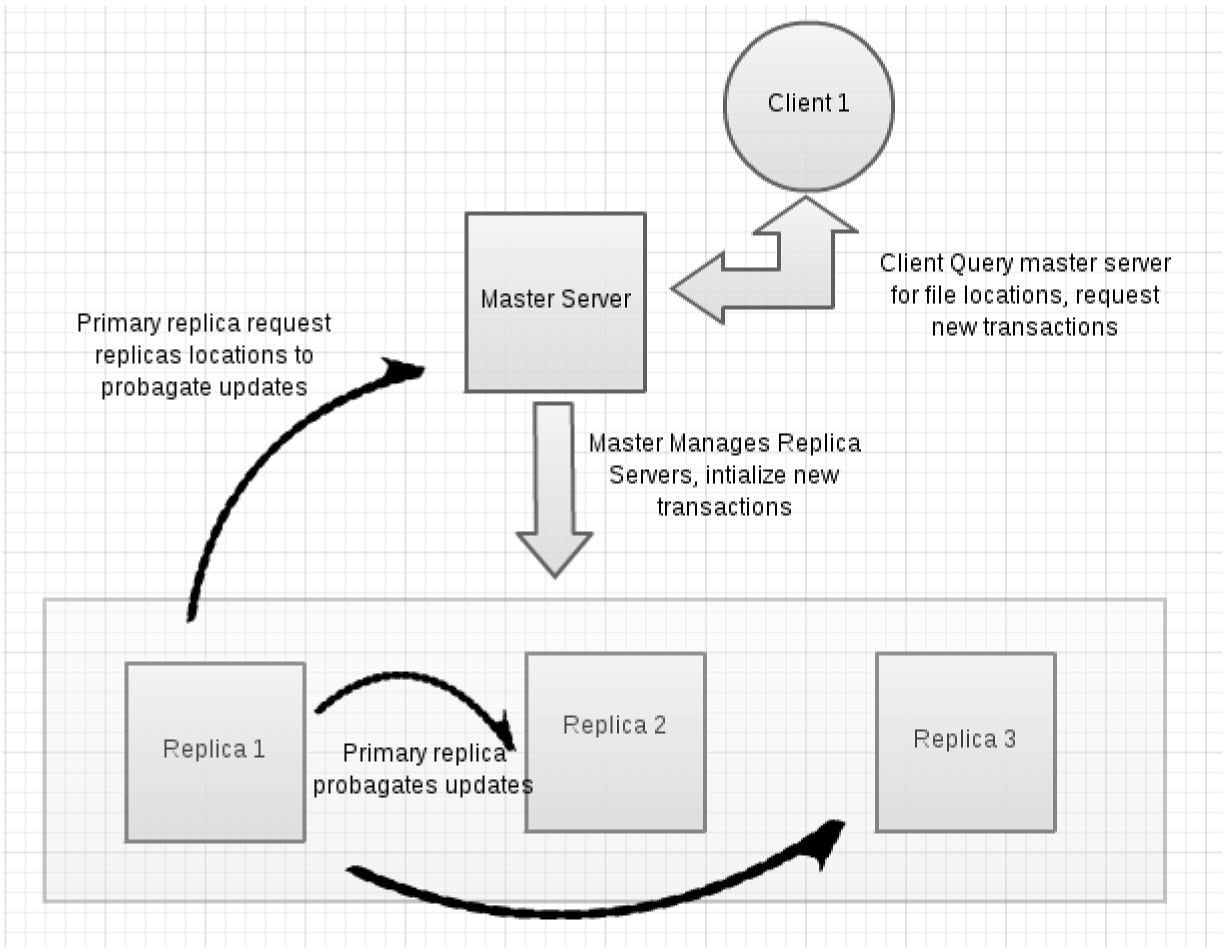
1. Introduction
2. Overview of proposed System
3. Basic Operations
   3.1. Read
   3.2. Write
   3.3 Abort
   3.4 Commit
   3.5 Propagation of commands among replicas
4. Sequential consistency and concurrency
5. Master server
   5.1. Meta data setup
   5.2. Replica server interface
   5.3 Client Interface
   5.4. Main methods
6. Replica server
   6.1. Ensuring Sequential consistency
   6.2. Ensuring Durability
   6.3. Main Methods
7. Use cases
   7.1. Read
   7.2. Write
   7.3. Commit
8. Handling Failures
   8.1. Client failure before committing transaction
   8.2. Client write message Loss

1. Introduction

It is required to implement a replicated file system. There will be one main server (master) and, data will be partitioned and replicated on multiple replica Servers. This file system allows its concurrent users to perform transactions, while guaranteeing ACID properties. This means that the following need to be ensured:

- The master server maintains meta data about the replicas and their locations.
- The user can submit multiple operations that are performed atomically on the shared
- objects stored in the file system.
- Files are not partitioned.
- Assumption: each transaction access only one file.
- Each file stored on the distributed file system has a primary replica.
- After performing a set of operations, it is guaranteed that the file system will remain in consistent state.
- A lock manager is maintained at each replica Server.
- Once any transaction is committed, its mutations to objects stored in the file system are required to be durable.

2 Overview of proposed system


Client 1
Client Query master server for file locations, request new transactions
Master Server
Primary replica request replicas locations to probagate updates
Master Manages Replica Servers, intialize new transactions
Replica 1
Primary replica probagates updates
Replica 2
Replica 3

3 . Basic Operations

The system handles the basic operations, which are read,write,commit and abort

3.1. Read

The whole file is sent to the client

3.2. Write

The client is permitted to perform multiple write requests in a given transaction

3.3. Commit

No changes are reflected to the system until committing

3.4. Abort

A client can abort a transaction before committing

3.5. Propagation of commands

Primary Replicas are responsible for Propagation of commands

4 Sequential consistency and concurrency

The system maintains a sequential consistency for all replicated files, such that the sequential order of the transactions updating a file is maintained and the order of operations within each transaction is maintained, example: if T1 and T2 are writing to file F1, and T1 time stamp t1< T2 time stamp t2, if T2 commits before T1, the updates are reflected on the file F1 in the order T1 then T2.

5 Master server

The Master server holds information about:

1. File locations (Replica servers)
2. each file primary replica

5.1. Meta data setup

Initially the Master server query each replica server for the files stored in the replica server directory, the addresses of replicas are fed into the master through a file named “Metadata.txt”

5.2. Replica Server Interface

The Master provides a means for a Primary replica to query for other replicas replicating a given file, so that it can propagate updates to other replicas

5.3. Client Interface

The Master provides the Client with the following functionality:

1. A means to initiate a new transaction
2. A means to query for replicas replicating a given file

5.4. Main methods

Client Interface:

```
Write(file name)
    Txid ← create unique transaction id
    If file doesn't exist
      select 3 replicas for creating new file
    end if
    initaiate new transaction with id: Txid on primary replica
    return primary replica address

Read(file name)
    if file doesn't exist
      infrom client
    else
      return all replicas address replicating the file
```

Replica Server Interface:

```
getReplicas(filename)
    return all replica addresses replicating the file
```

6 . Replica server

Replica servers are managed by the master server, and provide an interface for clients to perform basic operations on files, also they are responsible for propagating any updates on any given file

6.1. Ensuring Sequential consistency

The system maintains a sequential consistency for all replicated files, such that the sequential order of the transactions updating a file is maintained and the order of operations within each transaction is maintained, example: if T1 and T2 are writing to file F1, and T1 time stamp t1< T2 time stamp t2, if T2 commits before T1, the updates are reflected on the file F1 in the order T1 then T2.

6.2. Ensuring Durability

Any uncommitted updates are not visible to the file system, until a successful commit is performed

6.3. Main Methods

Master Server Interface:

```
RequireFiles()
        return all file names on the replica

InitTransaction(transaction id,filename)
        create new transaction object and initialize maps
```

Client Interface:

```
Read(transaction id)
        return whole file to client

Write(transaction id,data)
        Append data to write buffer

Commit(transaction id)
        check for lost messages and inform client for retransmission
        if there exist older transaction writing to the same file
           wait till it ends
        flush write buffer to disk and make file visible
        Free transaction resources
```

7 . Use cases

7.1 Read

1. client query master for replicas containing file
2. Master return list of addresses
3. client select on replica to connect with
4. client issues a read request to the replica
5. replica return whole file
6. client saves the file in its local directory

7.2 Write

1. client request new transaction from master given a file name
2. master requests the primary replica to initialize new transactions
3. master returns primary replica address to the client
4. client connect to primary replica
5. client request the replica to write some data
6. replica buffer the data and ask the master for address of replicas holding the same file
7. master respond with a list of address
8. primary replica propagate the write request to all other replicas provided by the master

7.3 Commit

1. Client request a commit for current transaction from replica server
2. replica ask the master for address of replicas holding the same file
3. master respond with a list of address
4. primary replica propagate the commit request to all other replicas provided by the master
5. replica make file visible and release transaction resources

8. Handling Failures

In a typical replica file system a number of failures can occur like:

- Client failure
- Master failure
- Replica failure
- Message failure

In this system we address only client failure and message failure.

8.1 Client failure

If a client crashed in a middle of a transaction before committing, the primary replica can detect this, as it maintains a timer for each transaction, eventually a timeout will happen and the primary replica would abort the transaction.

8.2 Message failure

A message sequence number is maintained between the client and the replica servers, to detect any message loss, upon detecting the replica server acknowledge the client, and the client retransmits the lost message .