CSE 490h – Project 4: Paxos Writeup

Wayne Gerard - wayger
Zachary Stein - steinz

March 7, 2011

1 Nodes

Our system has three kinds of nodes: clients, coordinators, and two phase commit (2PC) coordinators.

Both the number of coordinators in the system and the total number of nodes in the system are required to be fixed. The user must also specify the number of coordinators to assign to each filename in the system.

Coordinators are expected to take on the lowest addresses in the address space, followed by the 2PC coordinators, and any other clients.

The majority of the coordinator address space should be alive at any given time to ensure responsiveness. A majority of 2PC coordinators must be alive in order for the system to successfully abort or commit any transactions. We currently only support one 2PC coordinator. The system's availability is unaffected by other client's liveliness.

2 Commands

Node counts can be configured by passing any node the following commands:

```
coordinators <count>, perfile <count>, nodes <count>
```

Clients support the following commands:

```
FS Commands: create <filename>, delete <filename>, get <filename>,
             put <filename> <contents>, append <filename> <contents>
TX Commands: txstart <filenames>, txcommit, txabort
```

3 File System Semantics

Our file system maintains a consistent, distributed log of operations for each file in the system using Paxos. The logs are currently only stored in memory since they can be recovered through Paxos. Storing them persistently would be consistent and could speed up node reintegration after brief failure.

Operations can be appended to the log as long as a majority of coordinators assigned to each file is responsive. Since we use Paxos, appending operations to the log is somewhat high latency. However, reads from the log are serviced locally (and therefore very quickly) using the log entries learned so far. If a consistent read is desired, it should be wrapped in a transaction.

Commands executed on different files are handled asynchronously. Clients are required to verify the validity of commands they attempt to append to the log. Because of this, multiple operations on the same file are handled synchronously, since the validity of executing commands following the first command in the chain depends on the state of the log after the first command is successfully

appended. Transaction commands are also handled synchronously and consistently using two phase commit.

The Command Graph

4 Paxos

Clients are all proposers. We expect the load to be distributed roughly evenly across the different files in the system. Coordinators could easily detect contention on a specific file and then switch on lead-proposer-mode via a log entry to ensure Paxos liveliness. The coordinators assigned to each file act as the acceptors and learners for that file. Clients listen to logs by registering with any learner (coordinator).

Coordinators are assigned to files using a simple hash function. The filename's hash code and following integers are used as the addresses of the coordinators. In a multiple data center setting, addresses should be distributed round-robin amongst data centers for maximum reliability. Alternatively, decreased latency could be achieved by assigning addresses sequentially within a data center.

5 Two Phase Commit

Clients write TryTXStart (which include the list of filenames used in the transaction), TryTX-Commit, and TryTXAbort messages to the log. TryTXStart entries implicity lock the log and are required to be written in order to avoid deadlock.

The 2PC coordinator listens to all the logs and monitors the state of transactions. When it notices that all of the filenames in some transaction have TryTXCommit or TryTXAbort entries in their logs, it proposes TXCommit or TXAbort entries to those logs. Each file is unlocked as soon as it is committed or aborted.

If the 2PC coordinator goes down while performing a transaction, it can recover it's state by reading the logs.

It is also up to the 2PC coordinator to abort transactions that have been started, but not committed or aborted by their owners. The 2PC coordinator is free to try and inject TXAborts into the log whenever it pleases - the client will learn that their transaction has been aborted and can respond appropriately. However, this mechanism is intended primarily to cleanup after failed nodes - not interrupt them.

6 Comparison to Client Server Architecture

Compare guarantees, node usage

Compare Paxos round packet round-trip counts to client server round-trip counts

Code size and complexity

Don't need replication, don't even need persistent storage for files - everything is in the consistent, distributed log maintaned by Paxos.

7 Packet and Persistent Store Formats