15-440/640 Distributed Systems
Homework 3

Due: November 17, In Class

Name:
Andrew: ID

October 27, 2015

Distributed Replication and PAXOS

1. You have set up a fault-tolerant banking service. Based upon an examination of other systems, you've decided that the best way to do so is to use Paxos to replicate log entries across three servers, and let one of your employees handle the issue of recovering from a failure using the log.

   The state on the replicas consists of a list of all bank account mutation operations that have been made, each with a unique request ID to prevent re-transmitted requests, listed in the order they were committed.

   Assume that the replicas execute Paxos for every operation.[1] Each value that the servers agree on looks like "account action 555 transfers $1,000,000 from Mark Stehlik to David Andersen". When a server receives a request, it looks at its state to find the next unused action number, and uses Paxos to propose that value for the number to use.

   The three servers are S1, S2, and S3.

   At the same time:

   S1 receives a request to withdraw $500 from A. Carnegie.

   - S1 picks proposal number 101 (the n in Paxos)

   - S2 receives a request to transfer $500 from A. Carnegie to A. Mellon.
     - S2 picks proposal number 102

   Both servers look at their lists of applied account actions and decide that the next action number is 15. So both start executing Paxos as a leader for action 15.

   Each sequence below shows one initial sequence of messages of a possible execution of Paxos when both S1 and S2 are acting as the leader. Each message shown is received successfully. The messages are sent one by one in the indicated order. No other messages are sent until after the last message shown is received.

   Answer three questions about the final outcomes that could result from each of these sequences:

   (a) Is it possible for the servers to agree on the withdrawal as entry 15?

   - Is it possible for the servers to agree on the transfer as entry 15?
   - For each of these outcomes, explain how it either could occur or how Paxos prevents it.

   To be clear on the message terminology: The PREPARE message is the leader election message. RESPONSE is the response to the prepare message. ACCEPT says "you've agreed that I'm the leader, now take a value."

[1] In practice, most systems use Paxos to elect a primary and let it have a lease on the operations for a while, but that adds complexity to the homework problem.

Sequence 1:

```
S1 -> S1 PREPARE(831)
S1 -> S1 RESPONSE(nil, nil)

S1 -> S2 PREPARE(831)
S2 -> S1 RESPONSE(nil, nil)

S1 -> S3 PREPARE(831)
S3 -> S1 RESPONSE(nil, nil)

S2 -> S1 PREPARE(832)
S2 -> S2 PREPARE(832)
S2 -> S3 PREPARE(832)
... the rest of the Paxos messages.
```

Sequence 2:

```
S1 -> S1 PREPARE(831)
S1 -> S1 RESPONSE(nil, nil)

S1 -> S2 PREPARE(831)
S2 -> S1 RESPONSE(nil, nil)

S1 -> S3 PREPARE(831)
S3 -> S1 RESPONSE(nil, nil)

S1 -> S3 ACCEPT(831, ‘‘withdraw...’’)

S2 -> S1 PREPARE(832)
S2 -> S2 PREPARE(832)
S2 -> S3 PREPARE(832)
... the rest of the Paxos messages.
```

Sequence 3:

```
S1 -> S1 PREPARE(831)
S1 -> S1 RESPONSE(nil, nil)

S1 -> S2 PREPARE(831)
S2 -> S1 RESPONSE(nil, nil)

S1 -> S3 PREPARE(831)
S3 -> S1 RESPONSE(nil, nil)

S1 -> S3 ACCEPT(831, ‘‘withdraw...’’)
S1 -> S1 ACCEPT(831, ‘‘withdraw..’’)
S2 -> S1 PREPARE(832)
S2 -> S2 PREPARE(832)
S2 -> S3 PREPARE(832)
... the rest of the Paxos messages.
```

(b) Suppose one of the servers received an ACCEPT for a particular instance of Paxos (remember, each "instance" agrees on a value for a particular account event), but it never heard back about what the final outcome was. What steps should the server take to figure out whether agreement was reached and what the agreed-upon value was? Explain why your procedure is correct even if there are still active leaders executing this instance of Paxos.

Map-Reduce/HADOOP

2. (a) What requirements does MapReduce/Hadoop place on what can be done in a Map operation, and why?

   (b) In MapReduce/Hadoop, if a Mapper node dies, can the cluster recover? If so, how?

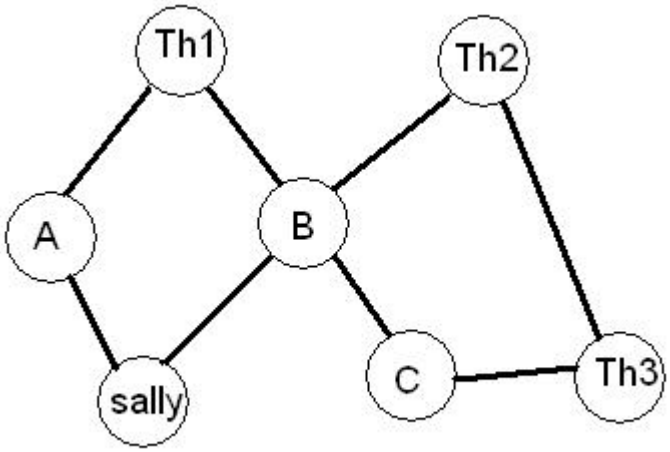
   (c) If a Reducer node dies, can the cluster recover? If so, how?

3. Sally has a very very very large map. She's really excited about finding all the movie theaters on the map, and how long it would take to get to each one (she collects movie tickets from various theaters, it's quite an impressive collection). Sally will need your help to find the shortest path from her house to every movie theater using MapReduce.

   Assume the following:

   - There are arbitrary points on the graph that represent intersections, known as A, B, C, etc.
   - All edges on the graph are undirected.

   Here is an example of small map:



   Clearly, on a graph of smaller size, Sally can use Dijkstra's to find the information she needs, but because this graph is so large, she wants to use MapReduce, where the MapReduce will effectively be another graph traversal algorithm.

   (a) Suppose that your input for the problem looks like this:

```
Sally A 3
B Th2 4
C B 1
...
```

   Set up a possible solution to this problem. Tell us how many MapReduce phases there would be, and what each Mapper and each Reducer would do. The same phase can be run more than once, so make sure you explain how and why. Give examples of input and output for each map and reduce of a phase.

   The general format can be:
   General idea:

   Phase 1: Map:
   Phase 1: Reduce:
   Run Phase 1 X times.

Distributed File Systems (HDFS/GFS)

4. (a) What are some design assumptions which apply to HDFS/GFS

   (b) GFS and HDFS, by default, triplicate each data block on three different nodes. How many node failures can it tolerate (at least)?

(c) Triplication results in 200% space capacity overhead. So your TA wants to reduce this overhead by using RAID and recalls that RAID is very efficient in saving space. Thus instead of storing block A and block B each on 3 different DataNodes by triplication, he modifies HDFS to store block A and B on 2 different data nodes respectively, and also store 2 new parity blocks calculated from A and B (e.g., $A \bigoplus B$ and $f(A, B)$—some function other than exclusive-or) on 2 more different DataNodes. What is the storage overhead now? How many node failures could this scheme ensure to tolerate?

(d) Your TA has a MapReduce job that has no output. Can you explain briefly the reason why your TA may observe this job runs slower on input data stored by the above scheme?

(e) (2 points) If we decrease the default HDFS block size, how could this change affect the NameNode?

Consistent Hashing

5. David is designing a distributed hash table with n nodes. The table will store values with m-bit keys; each node in the DHT has an ID obtained by hashing the nodes name (e.g., ID1 = h("node1")). He is considering two schemes for assigning key-value pairs to nodes responsible for storing them:

   - **Scheme 1**: Use consistent hashing. The node responsible for key k is the first node whose ID is equal to or follows k in the identifier space (modulo 2^m).
   - **Scheme 2**: Order the nodes numerically by their IDs. If a nodes position in this ranking is r (where $r \in [0, n)$), it is responsible for keys in the range $[\frac{r}{n} \cdot 2^m, \frac{r+1}{n} \cdot 2^m)$.

   (a) What is one advantage of scheme 1?

   (b) What is one advantage of scheme 2?

   (c) Consider a new node joining the DHT (for a new total of n+ 1). On average, for each scheme, what fraction of the key space will be assigned to a different node as a result? For simplicity, assume the following:

   - The new nodes ID is higher than any current nodes ID.
   - The IDs of the current nodes are evenly distributed: $ID_i = \frac{i}{n} \cdot 2^m$

DNS

6. In this problem, we will use dig tool available on Linux and Mac OS to explore DNS servers. You can read about it using man dig. Recall that a DNS server higher in the DNS hierarchy delegates a DNS query to a DNS server lower in the hierarchy, by sending back to the DNS client the name of that lower-level DNS server (assuming no recursion is specified). Hint: Be sure to use the +norecurse option to dig, and remember that you will need to specify different target DNS servers (@) each time. Your queries should look like $dig + norecurse$ @$< targetserver > RecordToResolve\ RecordType$.

   (a) Starting with a root DNS server (from one of the root servers [a-m].root-servers.net),initiate a sequence of queries using dig for the A-type record for www.cs.cmu.edu without using recursion. Be sure to show the list of the names of DNS servers in the entire delegation chain starting from the root in answering your query.

   (b) Repeat the same procedure as above for www.google.com

(c) Repeat the same procedure for 81.183.132.209.in-addr.arpa. This time, dig for the **PTR**-type record.

(d) Use **dig -x IP_addr** to perform reverse DNS lookup for the IP address **209.132.183.81**. What is the domain name associated with this IP address? What is the type of the DNS record in the answer section? Compare the answer section to part(c)'s answer