Distributed Systems

Homework #1

1. Despite that GFS scales well, it could be argued that the master is still a potential bottleneck. What would be a reasonable alternative to replace it?

- There are two solutions:
  1. Provide replication by using more than one master node.
  2. Considering that master uses a file name to look up a chunk server, we could also implement the master in the form of a DHT-based system and use a hash of the file name as the key to be looked up. In this way, one would obtain a fully decentralized master.

2. Give a simple extension to the NFS lookup operation that would allow iterative name lookup in combination with a server exporting directories that it mounted from another server.

- If a lookup operation always returns an identifier for the server from which a directory was mounted, transparent iterative name lookups across multiple servers would be easy. Whenever a server looks up a mount point on which it mounted a remote file system, it simply returns the server's ID for that file system. The client can then automatically mount the directory as well, and contact its associated server to continue name resolution.

3. Examples of distributed file systems are: AFS, NFS, and GFS.

a. List three of the design assumptions of the GFS.

1. File system is built from many inexpensive components – frequent failures.
2. Many large sequential writes that append data to files
3. Master maintains file system metadata:
   - File and chunk namespaces
   - Mapping from files to chunks
   - Location of each chunk's replicas

b. "An NFS file system appears at the same file path on all clients." State whether the previous sentence is true or false. Justify your answer.

False.

Each client can mount a shared directory.

c. "An AFS file system appears at the same file path on all clients." State whether the previous sentence is true or false. Justify your answer.

True.

AFS has a fixed mounting directory

d. List three of the requirements of a distributed file systems. Choose one of the three distributed file systems studied in class and show how these three requirements are satisfied in it.

1. Transparency.
2. Concurrent File Updates.
3. File Replication

In GFS:

1. **Transparency:** the master node private this requirement as the access to chunks is transparent to clients.
2. **Concurrent File Updates:** the primary replica controls concurrent file updates.
3. **File Replication:** each file is replicated on 3 chunks.

4. In a system with N processes that uses Lamport's logical clocks.

e. What does L_i represent?
Logical clock of process p_i.

f. When and how does it change?
L_i is incremented before event e is issued at p_i: $L_i := L_i+1$.

5. In a system with N processes that uses vector clocks.

a. What does VTi[j] represent?

- **VTi[i]**: Logical clock of process $\mathbf{p_i}$.
- **VTi[j] (i≠j)**: $\mathbf{p_i}$**'s** best guess of logical time at $\mathbf{p_j}$.

More specifically, the time of the occurrence of the last event in p_j which "happened before" the current event in p_i (based on messages received)

b. When and how does it change?

- Clock VTi[i] must be incremented between any two successive events in process Pi: $VT_i[i]:= VT_i[i]+1$.
- If event a is the event of sending a message m in process Pi, then message m is assigned a vector timestamp tm =VTi When that same message m is received by a different process Pj, VTj is updated as follows:
  For all p, VTj[p]:= max(VTj[p], tm [p])

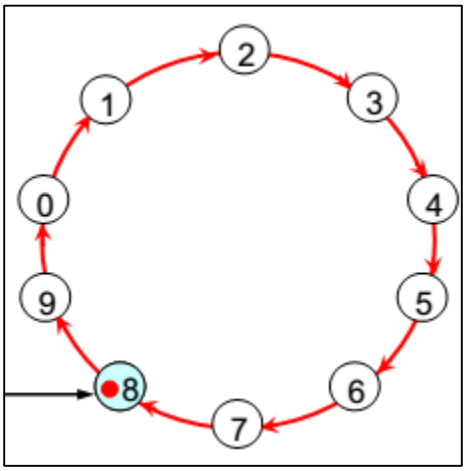
6. We studied in class a distributed algorithm for performing mutual exclusion that uses Lamport logical clocks. Explain why Lamport's logical clock cannot be replaced with a vector logical clock in this algorithm. Would it be suitable to use a physical clock? Justify your answer.

- The idea behind performing mutual exclusion is to do total ordering of all events in the system, Lamport Clock is used for achieving total ordering. Total order cannot be enforced by vector timestamp.
- Physical clocks can't be used, as the clocks need to be synchronized.

7. To achieve totally-ordered multicasting with Lamport timestamps, is it strictly necessary that each message is acknowledged?

No, it is sufficient to multicast any other type of message, as long as that message has a timestamp larger than the received message. The condition for delivering a message m to the application, is that another message has been received from each other process with a large timestamp. This guarantees that there are no more messages underway with a lower timestamp.

8. Give an example execution of the ring-based algorithm to show that processes are not necessarily granted entry to the critical section in happened-before order



Assume that now process 8 has the token and there is event (b) happened at it once it received the token, but there is event at 6 (a) happened when the token was with node 7. Therefor node 8 will be able to enter the critical region before node 6 although a happened before b.