Exam 2 Review

Question 1

Why did Dropbox add notification servers to their architecture?

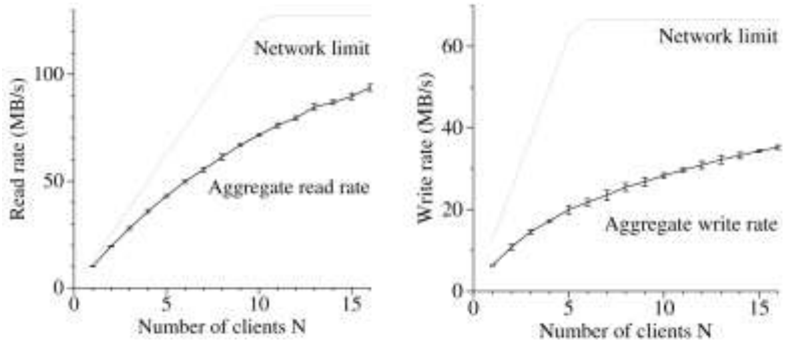
To avoid the overhead of clients polling the servers periodically to check if there are updates.

Polling is bad. Lots of clients hitting a server just to check if there are updates can create a lot of load.

But: Notification servers require a persistent connection ***from*** the client.

Question 2

As evidenced by the benchmarks from the Google GFS paper (shown), read data rates are much higher than write data rates in a GFS cluster. To what do you attribute this performance disparity? ?



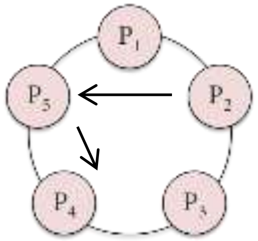
A write requires propagating data to replicas, so there is *N* times more data sent for *N* levels of replication.

A read can be made from any replica. With concurrent activity, reads can be load-balanced; writes cannot.

Question 3

A logical ring of five (5) processes is shown. Process P_1 is the leader but dies. Process P_2 detects the death of P_1 and starts a ring election in a counter-clockwise direction.

(a) Describe the message that Process P_4 receives



$\{ P_2, P_5 \}$

(b) What message does P4 forward and to whom?

Message = $\{ P_2, P_5, P_4 \}$

Forwarded to P_3

Question 4

Suppose a Paxos cluster of eight acceptors along with a bunch of clients gets partitioned in half. Some clients end up communicating with one set of acceptors while others communicate with the other set. Four acceptors agree to one value while four others agree to a different value. Consensus is not achieved. What is wrong with this scenario and how does Paxos avoid this problem of network partitioning?

Paxos requires a majority of acceptors to function. The algorithm will refuse to accept any proposal if a majority is not running.

Question 5

In MapReduce, the results of each *map* worker are stored in intermediate files at that worker. It would be more efficient to send a (*key, value*) pair to the *reduce* worker immediately, so the *reduce* worker can sort the keys as it receives each message and does not have to fetch the data from the *map* workers when they terminate.
What is a problem with this proposal?

This works great provided that neither map nor reduce workers fail!

If a *map* worker fails, it will need to be restarted but some of its data was already sent to *reduce* workers and is intermixed with results from other, non-failed, *map* workers.

If a *reduce* worker fails, a new one will be unable to get the input data unless it was stored in a well-defined, reliably-accessible place.

Note: the question does not state that the *reduce()* functions have started execution

Question 6

You have a database of records representing the inventory of a huge store (e.g., Amazon), each of which contains the fields {item, category, item-price, quantity-in-stock}.

Show how you can use MapReduce to compute the total value of the inventory of each category. Outline in pseudocode the map and reduce functions. Assume that each map function gets a single record as input. An emit(key, value) function will emit a {key, value} pair from the map function. A reduce function is called with a key and an array of items that match the key. The reduce function can call print to print values.

```
map(item, category, item-price, quantity-in-stock):
    emit(category, item-price * quantity-in-stock)


reduce(key, value[]):
    print(key, sum(value))
```

Question 7

Which statement best describes a client's interaction with Chubby?

a) A client contacts any Chubby server, which then forwards the request to the appropriate server in the Chubby cell.

b) A client contacts any Chubby server, which can process the request since all Chubby servers are replicated.

c) A client contacts the Chubby master, which then forwards the request to the appropriate server in the Chubby cell.

d) A client contacts the Chubby master, which handles all client requests.

Question 8

Dropbox differs from many other Internet-based services (e.g., Twitter, Facebook, Reddit) in that:

a) Clients send almost as many *write* requests as *read* requests.

b) Client requests are shorter than with most other services but very frequent.

c) Client requests for data are far larger but very infrequent.

d) Dropbox is almost exclusively a read-only service, making it easy to replicate servers for load balancing.

Question 9

Which file systems place file metadata on separate servers from file data?

a) Dropbox and GFS

b) Chubby, Dropbox, and GFS

c) AFS and GFS

d) GFS and Chubby

Metadata = data about the files: name, size, timestamps, access rights

Chubby is a single server that happens to keep replicas of itself

AFS may use multiple server but each server serves volumes; volume = complete files and directories

Question 10

A lease is:

a) A lock that has a timeout.

b) A lock that multiple processes can share.

c) A fine-grained lock.

d) A coarse-grained lock.

Question 11

Compared with Lamport's mutual exclusion algorithm, Ricart & Agrawala's requires approximately:

a) 2/3 the messages of Lamport's.

b) 3/2 (1.5x) the messages of Lamport's.

c) 1/2 the messages of Lamport's.

d) The same number of messages of Lamport's.

Lamport's:

N requests + N acks + N releases = 3N

R&A:

N requests + N acks (0 or more delayed) = 2N

Question 12

For Lamport's [mutual exclusion] algorithm to work correctly:

a) No two requests can have the same timestamp.

b) No two requests for the same resource should be sent at the same time.

c) A majority of processes must be running.

d) One process must be elected to run as a coordinator.

Lamport’s mutex algorithm requires each process to store a request queue sorted by timestamp order. Each process must have an identical view of the queue. If two messages have the same timestamp, there is no assurance that all processes will place them in the same order.

Question 13

Which one of these approaches works best for choosing a winner in the ring election algorithm?

a) Choose the first process in the list.

b) Choose the last process in the list.

c) Choose the process with the lowest process ID in the list.

d) Choose a random process from the list

We might have multiple concurrent elections.

Every process that completes an election must arrive at the same decision from the data in their list.

First & last processes will differ based on who started the election.

Question 14

The Chang & Roberts optimization to the ring algorithm:

a) Optimizes the ring to find a leader quicker.

b) Drops certain messages to try to stop concurrent elections.

c) Adds a coordinator to keep track of the progress of the election.

d) Avoids searching through a list at the end by having every process replace the process ID in an incoming message with its own.

(a) No. The election must circulate through all live machines

(b) Yes. C&R tries to kill off concurrent elections – but not *all* of them. If a process participated in an election & has a higher ID than an incoming election message, there is no chance the process ID in that election will be the winner, so drop the message.

(c) No.

(d) Close. A process replaces the process ID in the message with its own **only if** [Process ID] > [ID in the message].

Question 15

Which is the most accurate statement about active-passive replication?

a) Only one server processes client requests; it propagates updates to others that are on standby in case it fails.

b) Client requests are load-balanced among all servers; the server that handles a given request propagates updates to all replicas.

c) The system consists of only one server, but it backs up its state onto permanent (durable) storage.

d) The client sends the request to a master server, which then forwards the request to an available passive server.

Active-passive replication requires only one-way writes: from the master (active) system to all replicas.

Question 16

A state transfer takes place when:

a) A process joins a group.

b) A process leaves a group.

c) A view change takes place.

d) A replicated backup process takes over for a primary process.

A *state transfer* is when a process needs to bring itself up to date and acquire the latest state of the system (e.g., it was dead and did not get updates).

Note: the question does not ask about a "view change"

Question 17

A message is *unstable* when:

a) It is not certain whether it has been delivered to all group members.

b) It is associated with an expiration time.

c) It has errors in it due to a byzantine fault.

d) It has been received but not acknowledged.

In virtual synchrony, an unstable message is a message whose receipt has not been confirmed by all group members.

A view change cannot take place until all unstable messages are delivered.

Question 18

Which statement is *FALSE*? In virtual synchrony, *flush* messages:

a) Ensure that all unstable messages are delivered to their process group.

b) Implement a barrier.

c) Must be acknowledged before a view change completes.

d) Discard all undelivered messages.

(a) Yes. a *flush* forces delivery of any unstable messages & confirmation

(b) Yes

(c) Yes

(d) No. A *flush* delivers unstable messages, not discards them.

Question 19

By sending messages through Paxos, processes can ensure that concurrent messages become:

a) Global time ordered multicasts.

b) Totally ordered multicasts.

c) Causally ordered multicasts.

d) Unordered multicasts.

Paxos ensures that each message gets a unique sequence number. Any proposal for a lower sequence number than one that has already been used or promised will be rejected.

Question 20

With Paxos, message sequence numbers are assigned by the:

a) Acceptor

b) Client

c) Learner

d) Proposer

(a) No. The acceptors promise to disallow requests with lower numbers. All acceptors from a majority of live acceptors need to agree to the proposed number (it might be the case that only one has the latest number).

(b) No. Clients just send streams of events

(c) No. Learners propagate the results (responsible for multicasting results in total order).

(d) Yes. The proposer takes client requests, associates them with a number, and asks the acceptors to agree to that number. If they fail, try a higher number.

Question 21

A write-ahead log does *NOT* enable a process to:

a) Undo changes in case of an abort.

b) Record that a transaction has committed.

c) Record the response that was sent to a vote in a commit protocol.

d) Achieve higher performance by prefetching data.

(a) The log stores old values of the data to enable rollback.

(b) A commit makes the results permanent. Even after a restart, the system can report that it has committed the transaction. A garbage collector can trim the transaction log to remove committed data in the log

(c) Yes. This allows a restarted process to know what promises it made.

(d) No.

Question 22

To abort a distributed transaction:

a) At least one participant must vote to abort.

b) The majority of participants must vote to abort.

c) All of the participants must vote to abort.

d) All live participants must vote to abort.

A commit is an all-or-nothing agreement.

Question 23

The three-phase commit protocol adds this to the two-phase commit protocol:

a) It communicates the result of the *"can you commit?"* vote to every replica.

b) It enables a committed transaction to roll back if any participant dies.

c) It enables a committed transaction to roll back if the coordinator dies.

d) The coordinator contacts all participants to ask if they agree to commit their transaction.

With 2PC:

Phase 2: coordinator sends commit messages but doesn't get to everyone before it dies. A recovery coordinator cannot probe a client to get a reliable state of the transaction.

Question 24

The motivation for an eventual consistency model was:

a) It is impossible to have highly available replicated data that is fully consistent in a system that can survive network partitioning.

b) Data inconsistencies are highly undesirable, so the primary focus should be on ensuring that all replicas are consistent.

c) Distributed commit protocols can never work reliably, so data is bound to become inconsistent on some participants.

d) Data might be in an inconsistent state during the execution of transactions but will be made consistent when they commit.

Brewer's CAP theorem: if you want consistency, availability, and partition tolerance, you can get at most two out of three.

The eventual consistency model favors high availability and partition tolerance – at the expense of consistency.

Question 25

Which of these is *NOT* an advantage of the Chandy-Misra-Hass probe deadlock detection algorithm on a reliable but asynchronous network?

a) There is little computation needed at each node.

b) There is no need to construct and maintain a graph of resource dependencies.

c) The algorithm is not subject to false deadlock.

d) The algorithm makes it easy for a node to detect that deadlock does not exist.

The first three are advantages.

The last is not true.

> If a deadlock DOES exist, then the *probe* message comes back to the sender and the sender knows *not* to request the resource.
>
> If there is no deadlock, the *probe* never makes it to the sender.

Question 26

The wound-wait deadlock prevention algorithm prevents deadlock by ensuring this condition does not occur:

a) Mutual exclusion

b) Hold and wait

c) Non-preemption

d) Circular wait

Ensures that an old process will not wait on resources that a younger process holds.

Question 27

In contrast to two-phase locking, strict two-phase locking:

a) Uses a two-phase commit protocol to get locks.

b) Uses a three-phase commit protocol to get locks.

c) Ensures that no transaction will read uncommitted data.

d) Preserves serializability (the Isolated part of ACID).

(a) No.

(b) No.

(c) Yes. Two-phase locking enables a transaction to read data that has been unlocked by another transaction. This can lead to ***cascading aborts***.

(d) Yes, but so does two-phase locking

Question 28

With optimistic concurrency control, transactions:

a) Do not lock the data that they modify.

b) Agree to commit at the start of the transaction and do not need to run a commit protocol.

c) Only lock data that they modify, but not data they read.

d) Use shared locks instead of exclusive locks on data.

Optimistic concurrency control idea:

Locking is too resource intensive and unduly limits concurrency.

In many environments, the chances of one transaction actually interfering with another is very small.

Question 29

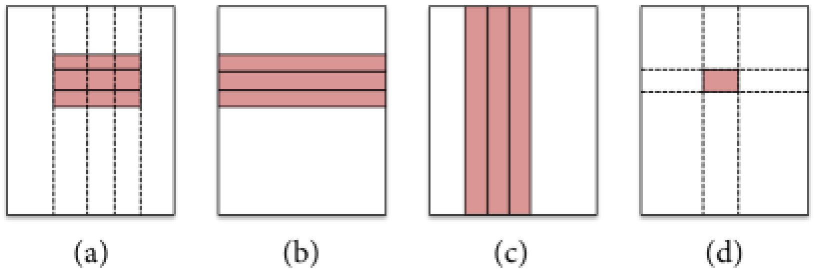
In Bigtable, each server is responsible for serving tablets. A *tablet* is:

a) A subset of rows and column families.

b) A subset of rows but stores all column family data for those rows.

c) A set of one or more column families but it stores all row data for those column families.

d) Exactly one row and one column family.



Each worker handles multiple tablets

A tablet is a horizontal slice of a full table

Question 30

In the Bulk Synchronous Parallel (BSP) framework:

a) Processes may have different amounts of work to do and a long running process may span multiple supersteps.

b) One long-running process will cause every other process to wait before starting the next superstep.

c) Processes are load-balanced so that no one process takes substantially more time to run than any other.

d) A process may be distributed among multiple processors if it is taking a long time to complete.

Supersteps end in a barrier. If a superstep takes a long time on one process, everyone else waits.

Question 31

A failed process in Pregel means that:

a) The entire job has to be restarted from the beginning.

b) The entire job has to be restarted from the last checkpoint.

c) Only the failed process has to restart from the beginning.

d) Only the failed process has to restart from the last checkpoint.

Processes communicate.

Not (c) or (d): Restarting only a failed process will have that process resend messages that other processes have already processed.

Not (a): The point of *checkpoints* is to avoid this.

The End