Distributed Systems

2016 Exam 2 Review

Paul Krzyzanowski
Rutgers University
Fall 2016

Question 1

What makes a message unstable? How does an unstable message become *stable*?

In virtual sychrony, a message is unstable if a group member is not certain that it has been received by every other group member. It becomes stable when the group member transmits it to every other group member and receives acknowledgements from all of them.

Bad answers:

- *Not all members confirmed receipt of the message*
  - *Except during a view change, a group member does not get confirmations from other members*
- *It's unknown if the recipient receives the message*
  - *The "unstable" property is at the recipient, not at the sender.*
  - *It's not enough for the coordinator to get confirmation – that information needs to be propagated to each group member.*

2

Question 2

Why is a transaction log crucial in providing fault tolerance in a two-phase commit protocol?
Hint: the answer has nothing to do with aborts

It allows the transaction to continue from where it left off when the system restarts.

A transaction cannot change its mind even if it dies and restarts. The log keeps the transaction's state.

Bad answers:

- *"Revert if a sub-transaction fails" or "enable rollback"*
  - *That's only done in the case of aborts*
- *Recovery server can take over*
  - *A recovery server cannot access a failed coordinator's log*

3

Question 3

How does Eric Brewer's CAP theorem affect the design of highly-available systems that can withstand network partitions

If you need availability and partition tolerance, you cannot have consistency.

4

Question 4

How did callbacks in AFS enable it to scale to support more clients than NFS?

They allow each client to support long-term caching – no need to check with the server; it will tell you if a file has been modified.

Bad answer:

Explain callbacks without explaining why they enable AFS to scale

5

Question 5

Under what conditions would consistent hashing be unnecessary for a distributed hash table?

If you never need to expand or shrink the table – i.e., you never need to add or remove servers

Bad answers:

- *"If there are fewer keys than the number of slots"*
- *"If there are no collisions" or any answer related to collisions.*

6

Question 6

The two-army problem illustrates that:

a) Achieving reliable communication between two parties requires the use of a third party.
b) You may need to use multiple levels of acknowledgements when using an unreliable asynchronous network.
c) You cannot guarantee agreement when communicating via an unreliable, asynchronous network.
d) An asynchronous network can be made to look like a synchronous network through the use of timeouts.

Messages may be lost and acknowledgements can be lost.
Infinite acknowledgements for certainty

7

Question 7

In virtual synchrony, the *group membership service*:

a) Is responsible for pinging servers to see if they are still alive.
b) Is used to notify servers if a new member joins a group.
c) Is a central coordinator that implements reliable multicasts.
d) All of the above.

The GMS is used to keep track of group ownership & propagating that knowledge to group members.

Group members report non-responding members to the GMS

Group members communicate with the group directly.

8

Question 8

A *Paxos proposer* may change its mind and pick a different proposal:

a) If it receives a higher-numbered proposal from any acceptor.
b) If it receives a different proposal number from the majority of acceptors.
c) If it receives a delayed but earlier proposal from a client.
d) Only after consensus has been reached.

If *any* acceptor responds with a higher # proposal, the proposer is obligated to use that for phase 2 of the protocol.

This is a way to share knowledge of the highest proposal received so far.

9

Question 9

Unlike Paxos, Raft:

a) Requires all requests to go through an elected leader.
b) Does not require a majority of servers to agree to the proposed value.
c) May not always succeed in achieving consensus even with a majority of servers functioning.
d) Provides a fault-tolerant framework for consensus.

Paxos makes an elected leader ("distinguished proposer") optional; Raft requires it.

10

Question 10

To achieve consensus, Paxos requires the functioning of:

a) The majority of proposers
b) The majority of acceptors.
c) The majority of learners.
d) All of the above.

Only one proposer is usually used (the "distinguished proposer"). Otherwise, there's a greater risk of the proposal being rejected with a higher proposal number from another proposer (which is OK, but will require another round).

At least one learner is needed to propagate the knowledge.
Multiple learners will simply propagate redundant messages.

11

Question 11

Which is not an ACID property of transactions?

a) *Atomic*. The intermediate state of a transaction is not visible.
b) *Consistent*. A transaction cannot leave the system in an inconsistent state.
c) *Isolated*. The net effect of concurrent transactions should be the same as if they ran in serial order.
d) *Distributed*. A transaction is made up of sub-transactions running on multiple systems.

Distributed – transactions do not have to be distributed
D = Durable

12

Question 12

The *two-phase commit protocol* uses two phases to:

a) Perform a tentative commit followed by a permanent commit.
b) Give a chance for a sub-transaction to request more time.
c) Make sure everyone's vote is received before taking action.
d) Release resources used by each sub-transaction prior to committing.

Phase 1: Get a vote from everyone
Phase 2: Tell everyone what to do

(a) There are no tentative commit.

(b) If a sub-transaction needs more time, it can choose to abort or delay responding.

(c) That's part of the commit/abort process.

13

Question 13

The *three-phase commit protocol*:

a) Allows a coordinator to abort all sub-transactions if any one does not respond to a vote.
b) Enables a recovery coordinator to query any sub-transaction for the state of the protocol.
c) Allows a participant to query another participant to decide to commit or abort if it does not hear from the coordinator.
d) All of the above.

3PC was designed to:
- Enable the use of a recovery coordinator
- Enable sage timeout-based aborts/commits (in cases where possible)

(a) If no commit/abort directives were issued, the coordinator can tell everyone to abort if there's a delay from any participant
(b) If any sub-transaction has a "pre-commit" vote, that means consensus was reached. If not, the recovery coordinator knows it can abort
(c) If a participant queries any other participant and gets the "pre-commit" vote, it knows the guaranteed outcome of the transaction.

14

Question 14

Two-phase locking means a transaction:

a) First checks if any other transaction holds the lock before requesting it.
b) First requests a read lock, followed by a write lock only if it needs to modify the resource.
c) Becomes the owner of the lock and can grant it to another transaction when it is finished.
d) Cannot request a lock if it already released any lock.

Phase 1: acquire all locks
Phase 2: release all locks

15

Question 15

Unlike two-phase locking, *strict two-phase locking*:

a) Prohibits a transaction from getting a lock unless it first checks if it is held by another process.
b) Requires a transaction to get a lock for any resource it accesses.
c) Ensures that other transactions cannot access uncommitted data.
d) Ensures that transactions maintain serial order.

Phase 1: acquire all locks
Phase 2: release all locks at commit/abort

No need for cascading aborts

16

Question 16

Optimistic concurrency control is a technique that:

a) Tries to run all transactions at once.
b) Schedules transactions in an optimal sequence to avoid deadlock.
c) Does not require transactions to grab locks for resources while they are working on the transaction.
d) Requires that, once granted, locks will not be revoked from a transaction.

Optimistic concurrency control: assume there will be no conflict for resources

Take no locks but be prepared to abort if it turns out that some other transaction modified the data while you were using it.

17

Question 17

The main benefit of a stateless file server design is:

a) Improved consistency across multiple clients that access the same file.
b) Shorter requests for higher throughput.
c) Ability for increased client-side caching.
d) Simplified recovery from client or server crashes.

(a) No. There are no mechanisms for invalidating/updating caches.

(b) No. Stateless servers need all information in their requests.

(c) No – see (a)

(d) Nothing to clean up or restore if a client or server crashes.

18

Question 18

The NFS *automounter*:

a) Automatically remounts previously used remote directories after a system crash.
b) Enables an NFS server to tell a group of clients to mount specific remote directories.
c) Looks at past activity on the client to figure out what remote directories should be mounted in the future.
d) Avoids the overhead of mounting remote directories that will not be used.

Automounter – mount remote directories on first access

19

Question 19

If two clients modify the same file concurrently using AFS:

a) Two distinct versions of the file will be created on the server.
b) The last client to close the file overwrites the other client's changes.
c) Changes will be applied to the server in the order that they are made by the clients.
d) The last client to open the file will be considered to have the latest version and its changes will win.

Session semantics: last one to close a modified file wins.

20

Question 20

Coda's *client modification log* is used to:

a) Allow servers to keep track of which clients modified which files.
b) Make it easy to merge changes to the same file from multiple clients.
c) Propagate client changes to the server.
d) Provide a version control mechanism on the server.

When disconnected, clients log files that were change to the CML.

When reconnected, the log contains a list of files that need to be uploaded to the server(s).

21

Question 21

SMB *oplocks* (opportunistic locks) are:

a) Locks pushed by the server to force a client to wait before opening a file.
b) A technique to maximize concurrent file access without getting locks, resolving conflicts when the file is closed.
c) Permissions granted by the server to tell the client how it can cache file data.
d) Exclusive locks on a region of a file to ensure that only one client can modify data in that region.

oplocks control client caching behavior

22

Question 22

Which storage systems keep file metadata (information) and file data on separate servers?

a) GFS and Chubby.
b) Dropbox and GFS.
c) SMB and Chubby.
d) AFS and Coda.

Chubby: single-server file system – but replicated

SMB: protocol for single-server file systems

AFS: servers contain volumes, which store data & related metadata (VLDB identifies where a volume lives)

Coda: Same as AFS

'Dropbox: database for metadata and Amazon S3 for file data

GFS: master stores metadata; chunkservers store data

23

Question 23

GFS minimizes the time spent locking a file during appends by:

a) Using optimistic concurrency control techniques.
b) Allowing only one process to open a file at any time.
c) Using strict two-phase locking.
d) Uploading the data fully before making it part of the file.

Two phase (NOT two-phase locking) process used for writing
1. Send the data: propagated to N chunkserver replicas serially
2. Make the data part of the file:
   Master locks the file; primary chunkserver sends write request to all replicas – ensures consistent order is maintained

Data flow is separated from the control flow.

24

Question 24

A DNS referral is used:

a) When a domain name moves from one IP address to another.
b) To identify another name server that is closer to the query.
c) When a name server is no longer responsible for a given domain name.
d) To find the root name servers responsible for a domain.

A referral is sent when a DNS server knows another server that is responsible for a greater (closer) component of the queried name

25

Question 25

Chubby is typically deployed as a set of five servers:

a) To provide replicas in case of failures.
b) To distribute load for high-volume operations.
c) To provide increased storage capacity.
d) All of the above.

At any one time, all requests go to the elected master, which replicates changes to the Chubby replicas in the cell.

26

Question 26

A key programming approach in the Google cluster architecture is to:

a) Perform queries against a huge database that spans multiple servers but is accessed via a single API.
b) Send a query to any one of a huge number of replicated systems for processing.
c) Minimize overall load by forwarding queries from one small database to another until the query is resolved.
d) Convert a query of a large database into many concurrent queries of small databases and merge the results.

Searching/sorting is time consuming
Merging is easy

27

Question 27

Which provides the most direct access to content?

a) Central coordinator.
b) Query flooding.
c) Content-Addressable Network (CAN).
d) Chord with finger tables.

Central coordinator: no hops needed – it knows where everything is.

b, c, d may require multiple hops to find the content.

28

Question 28

An *overlay network*:

a) Makes access more efficient by using only the fastest network links.
b) Allows computers to communicate only with systems they know about.
c) Avoids the need for back propagation.
d) Reduces latency by establishing direct connections between systems.

29

Question 29

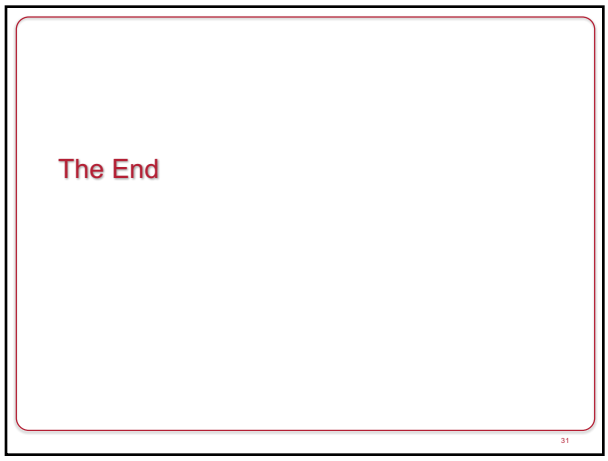
GFS file access is most similar to:

a) Central coordinator.
b) Query flooding.
c) Distributed hash table.
d) A single, centralized server.

GFS master knows about all files and which chunks make up a file.

Chunkservers don't know about files; they just serve chunks.

30


The End
31