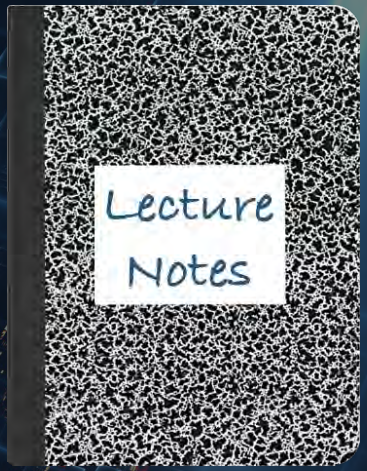
CS 417 – DISTRIBUTED SYSTEMS

Exam 3 Review



Paul Krzyzanowski

Question 1

Which of the following best describes the data model of Bigtable?

(a) Key-value store.
(b) Wide-column store.
(c) Document store.
(d) Multi-table database.

A wide-column store is a table where names and format of the columns can vary from row to row in the same table.

Question 2

Bigtable allows:

(a) Splitting the contents of a large row across more than one tablet server.

(b) Performing atomic, isolated transactions on a group of rows.

(c) Dynamically adding a new column to a row.

(d) Nesting a table within a table.

Bigtable splits tablets only by rows – an individual row is never split across tablets.

Bigtable only provides atomic operations on a single row – no locking of rows.

Question 3

Unlike Bigtable, a Cassandra table:

(a) Supports ACID semantics for transactions.

(b) Uses an SSTable (Sorted Strings Table) as the underlying structure for a table.

(c) Is a wide-column data store.

(d) Is split and distributed based on a user-defined key.

(a) Cassandra is a NoSQL database, providing *eventual consistency.*

(b) Both Bigtable & Cassandra store table data in SSTable structures.

(c) Both Bigtable & Cassandra are wide-column data stores.

(d) The partition key determines which server a row of a table gets stored.

Question 4

Cassandra's architecture is most closely inspired by:

(a) Amazon Dynamo & Google Bigtable.
(b) Google Bigtable & GFS.
(c) Google Bigtable & Spanner.
(d) Amazon Dynamo & Google Spanner..

Cassandra uses a wide column data store implemented with SSTables for table storage, like Bigtable.

It uses a Chord-like DHT similar to Dynamo.

Question 5

Spanner's *TrueTime* API:

(a) Enables timestamp-based detection of conflicts due to concurrent updates.

(b) Enables a transaction to wait until it is certain that its commit timestamp is in the past.

(c) Provides clients with a highly accurate timestamp via the use of atomic clocks and GPS.

(d) Enables any computer in the network to convert a globally consistent time value into the local timezone.

- TrueTime is used by Spanner to provide *external consistency*:
  - The order of transactions reflects their true time order.
  - If a transaction T_1 commits before a transaction T_2 starts, based on physical time, then the serial ordering of commits should reflect that and T_1 must get a smaller timestamp than T_2.
- Spanner implements this by using physical timestamps and commit wait: delay a commit until the timestamp is definitely in the past..

Question 6

Spanner does not require locks for a read-only transaction because:

(a) Transactions are serialized via two-phase locking.
(b) Optimistic concurrency control is used at commit time to detect modifications.
(c) The transaction does not modify data.
(d) The transaction will not read data newer than its timestamp.

(a) Spanner uses two-phase locking for r/w transactions – this implies locking.

(b) No.

(c) Even if a transaction didn't modify data, it would usually need a lock to ensure that others don't modify the data.

(d) Spanner uses multi-version concurrency control and stores multiple timestamped versions of data.

Question 7

What is the process of *shuffling* in MapReduce?

(a) Transferring data from the map workers to the reduce workers.
(b) Distributing input data to the nodes in the cluster.
(c) Reducing the size of the input data.
(d) Converting map workers into reduce workers.

Shuffling & sorting is the work done by the MapReduce framework between the *map* phase and *reduce* phase.

Shuffling is when each *reduce* worker contacts every *map* worker via remote procedure calls to transfer the <*key, value*> data that was targeted for its partition.

This data is then sorted so all values can be grouped by unique keys.

Question 8

The *map* step in MapReduce is responsible for:

(a) Aggregating results.
(b) Processing and transforming input data.
(c) Sorting data.
(d) Distributing tasks to workers.

The *map* phase parses the data to generate <*key, value*> pairs.

Question 9

In BSP, *barriers* are used to:

(a) Prevent any process from generating too much data.
(b) Optimize network traffic.
(c) Synchronize processes at the end of each superstep.
(d) Force processes to stop execution.

Barriers are the synchronization points between supersteps.

The next superstep cannot not start until all processes complete the previous superstep.

Messages from the previous superstep are delivered to processes at the start of the next superstep.

Question 10

Fault tolerance in Pregel is achieved through:

(a) Checkpointing.
(b) Replication of processing nodes.
(c) Data mirroring: replicating stored data.
(d) Automatic retries if a process fails to complete.

Like BSP, Pregel operates in supersteps. The barrier at the end of each superstep is a time when user processes perform no computation, so the state of the system is stable.

Pregel allows processes to save their state in stable storage (a distributed file system) every *N* supersteps.

If a machine fails, vertices are reassigned to available machines and all the vertex state is restored from the last checkpoint.

Question 11

In Pregel, *vertex-centric programming* means:

(a) Each vertex has its own processor.

(b) The program is written from the perspective of a single vertex.

(c) Vertices in a graph are processed sequentially.

(d) Each vertex is considered an independent entity and unaware of others.

The user subclasses a Vertex class and implements a *Compute* method that will be executed for each vertex.

This method can get the vertex value, outgoing edges, and incoming messages.

Question 12

In Spark, a *Resilient Distributed Dataset*, or RDD, is:

(a) A set of data that is replicated across multiple servers for fault tolerance.

(b) A distributed collection of objects representing either original data or the output of a transformation.

(c) The original input data that will be processed by Spark.

(d) Output data generated by a Spark action.

An RDD is the input data to a transformation or action.

Transformations produce RDDs.

An RDD can be sharded across multiple systems but is not replicated – it can be regenerated if necessary by reapplying the transformation that created it.

Question 13

What is one benefit of *lazy evaluation* in Apache Spark?

(a) It allows a transformation to be distributed across multiple executors.

(b) It ensures that transformations are executed exactly in the order specified by the programmer.

(c) It allows the framework to optimize the execution of transformations.

(d) It enables even load distribution across all executors (processing nodes).

Instead of executing transformations immediately, Spark constructs a Directed Acyclic Graph (DAG) that identifies the sequence of transformations required for each action.

This allows Spark to optimize the processing pipeline, such as combining multiple transformations into a single operation or avoiding running unnecessary transformations, thus reducing computational overhead and improving overall performance.

Question 14

How does Kafka ensure message durability?

(a) By replicating messages across multiple nodes.
(b) By affixing Message Authentication Codes (MACs) to each message.
(c) By allowing multiple consumers to read the same message.
(d) By being implemented on top of a fault-tolerant distributed file system like HDFS or GFS.

Kafka writes all received messages into stable storage.

Each message may be replicated *N* ways via state machine replication (a modified version of the Raft consensus protocol).

Question 15

What is the purpose of a *consumer group* in Kafka?

(a) To enable multiple consumers in the group to receive the same stream of messages.
(b) To enable members of the group to subscribe to different topics.
(c) To create a pipeline where one group member can receive a message and forward results to the next member.
(d) To distribute the processing of messages across multiple consumers in the group.

Consumers in a consumer group share the message queue:

- When one consumer gets a message, it is out of the queue

Allows the workload to be distributed across multiple consumers

Question 16

How does Akamai use DNS in optimizing content delivery?

(a) To resolve domain names to the IP addresses of the closest edge server.

(b) To find the most efficient route from the user making the query to the origin server.

(c) To use the worldwide set of DNS servers to cache content that will be returned with a DNS query.

(d) To make sure that access to content is secure.

DNS is used to to return an IP address of the preferred server to a customer's domain name query

- This will generally be the server with the lowest latency to the customer

Question 17

Akamai's *overlay network* enhances content delivery by:

(a) Bypassing the Internet.

(b) Improving routing.

(c) Routing traffic to a central storage server.

(d) Redirecting user traffic to caching servers.

(a) The Internet is not bypassed.

(c) There is no central storage server.

(d) User traffic is redirected to caching servers via the DNS query.

The overlay network is the worldwide collection of Akamai servers.
They maintain open TCP connections between them and periodically measure latency and bandwidth.

- This enables Akamai to make more efficient routing decisions by explictly forwarding packets through its servers on different networks

Question 18

In BitTorrent, what is a *torrent* file?

(a) The original version of a large file that is being shared on BitTorrent.
(b) An incomplete version of the file that is being downloaded.
(c) A file that contains information about a file that is being shared.
(d) One or more complete copies of files that are being shared on BitTorrent.

A torrent file contains information about the content:

- File name
- File size, SHA-256 hash of the file
- Piece size, SHA-256 hash of each piece of the file
- Address of a tracker server

(a,d) The original version is the seed.

Question 19

How does BitTorrent achieve high data transfer speeds?

(a) By compressing a file before downloading it.
(b) By having a peer test and select the lowest latency server that has a copy of the file.
(c) By deploying a large number of caching servers.
(d) By having peers make their partially-downloaded content available to other peers that need it.

Peers that have downloaded some pieces of the file are leechers: they are still downloading but make their content available to other peers.

The more peers are downloading content, the more places there is for a peer to get it.

Question 20

For Alice to send data that only Bob can read, Alice would encrypt it with:

(a) Alice's public key.
(b) Alice's private key.
(c) Bob's public key.
(d) Bob's private key.

Alice needs to encrypt it in a way that only Bob would be able to decrypt it.

(a) Only Alice would be able to decrypt this with her private key.

(b) Anybody would be able to decrypt this with Alice's public key.

(d) Alice shouldn't have access to Bob's private key.
If she did, anybody would be able to decrypt this with Bob's public key.

Question 21

To prevent intruders from generating a new hash of an altered message, a Message Authentication Code (MAC):

(a) Is encrypted with the sender's private key.
(b) Is encrypted with the sender's public key.
(c) Uses a secret key as part of the input to the hash function.
(d) Is sent separately from the message.

Message Authentication codes are a hash of not just the message but the message and a shared secret key.

You need to know the key to be able to validate the message.

Public key cryptography is not used.

Question 22

The *Diffie-Hellman Key Exchange* algorithm:

(a) Allows one party to send an encryption key securely to another party.

(b) Provides a secure mechanism so that two parties can get each other's secret keys.

(c) Allows two parties to come up with a common key that nobody else can compute.

(d) Allows a system to update its encryption key periodically.

Diffie-Hellman allows two parties to compute a shared common key.

(a) This can be used to encrypt another key so it can be sent securely.

(b) Same as (a).

(d) Users can periodically use D-H to come up with a new key, if desired.

Question 23

A *digital signature* differs from a Message Authentication Code (MAC) because:

(a) It contains the message and the authentication code.

(b) It identifies the origin but does not detect that the corresponding content has been modified.

(c) It does not rely on a hash function.

(d) It can only be created by one party.

MACs use shared keys

- If Alice & Bob communicate, either party can generate the same MAC

Digital signatures use public key cryptography.

Only the owner of the private key can generate a valid signature for that user

- If Alice & Bob communicate, Alice cannot generate Bob's signature and Bob cannot generate Alice's signature … but they can validate them

Question 24

A *digital certificate*:

(a) Binds information about someone together with their public key.

(b) Is a tamperproof data structure that stores a user's identity and their private key.

(c) Is a secure packaging of a file, its MAC, and a digital signature.

(d) Is a hash of the content encrypted by the owner of the content.

A digital certificate is a data structure signed by a certification authority (CA):

$$\{ \text{user\_identity}, \text{user\_public\_key}, \text{CA\_name} \}_{\text{CA\_signature}}$$

It allows public keys to be distributed along with the corresponding identity and the signature enables detection of modifications.

Question 25

Passkey authentication makes use of:

(a) Diffie-Hellman Key Exchange.
(b) Symmetric cryptography.
(c) Message authentication codes.
(d) Public key cryptography.

Passkeys were designed to move users away from passwords.

Passkeys are a public key authentication protocol:

Server sends a challenge

Client returns a signature of the challenge
(essentially the challenge encrypted with the user's private key)

The End