Storage Resources (File Systems)

- “block” and “page” have different meanings for SSD internals than we’ve seen for operating systems and interactions with block devices. What are the differences?
  - block (for SSDs): unit of erasure (perhaps 100s of KBs)
  - block (for using a block device): unit of transfer to/from device, perhaps 512 bytes
  - page (for SSDs): unit of read/write within the define (few KBs)
  - page (for system memory): unit of mapping between virtual memory and physical memory (usually 4 KB)
- How can we break a block device into smaller sections?
  - with partitions
- How can we treat multiple individual block devices like one bigger, more reliabel one?
  - With RAID (redundant array of inexpensive disks)
- Memory is byte-addressible. But what is the unit of access for an HDD, SSD, etc?
  - These are block devices, so the unit is a block. A typical block size might be about 512 bytes.
- On UNIX, how do we make many smaller file systems appear as one big one?
  - By mounting them on top of each other
- What are the names for these two data access patterns?
  - reading lots of data in order
    - sequential
  - jumping around and reading from many different places
    - random
- What are two ways we can use memory to make access to block devices more efficient?
  - caching: keep some disk data in memory, so we don’t need to go back to disk if the data is reused
  - buffering: when doing many smaller reads/writes, accumulate more data in memory before doing the transfer between disk and memory (because block devices are more efficient for bigger accesses).
- What is the data structure in a file system that points to blocks containing file data?
  - inode

Storage Resources (Formats and DBs)

- Compression works by looking for repetition over some window, say 32 KB. What would be the advantage and disadvage of increasing the window size?
  - Advantage: find more repetition, and use that to get a smaller file size
  - Disadvantage: spend more computation time finding the repetition
- Considering CSVs and parquets: when do we need to infer the schema?
  - With CSVs we need to infer it, because everything in a CSV is a string (need to detect that all the strings in a given column look like integers, for example).
  - Parquets have schema information built in, so there is no need to infer the schema.
- For analytics work, what are some tradeoffs between storing data in a database vs. storing it in some files (e.g., parquet files)?
  - Database advantage: a database will often have a special storage format co-designed to work with the analytics engine – the tight integration can lead to great performance.
  - Files advantage: if you want to go beyond doing SQL queries, having your data in a bunch of files makes it easier to use other tools. For example, it will probably be faster for some machine learning package to pull data from Parquet files (than do a big query on the database to extract everything).
- What does ETL stand for?
  - Extract Transform Load. Often used to get data into a data warehouse. For example, an ETL process might extract data from an OLTP database, transform it in some way, then load it into an OLAP database (the data warehouse).
- What does it mean for a format to be “column oriented”?
  - The data in the column is layout out consecutively in memory or within a file.
- What does OLAP stand for?
  - Online Analytics Processing
- What does snappy prioritize, high speed, or high compression ratios?
  - high speed. From the docs: “[Snappy] does not aim for maximum compression, or compatibility with any other compression library; instead, it aims for very high speeds and reasonable compression.”
- What is a data warehouse?
  - An OLAP database where we collect data from multiple sources (e.g., OLTP databases) for analysis.
- What is analytics processing?
  - Trying to answer questions about the data, which involves computation over whole columns of data.
- What is the orientation of CSVs and parquet files?
  - CSVs: row oriented. Parquet: column oriented.
- What is the tradeoff between text encoded and binary encoded formats?
  - text: easier to debug (because you can just look at it)
  - binary: usually more efficient (often less space, and numbers usually already in a format on which CPUs can directly operate without further conversion)
- What is transactions processing?
  - Operating on individual rows or a small number of rows at a time (e.g., adding, modifying, deleting rows).

SQL Databases (MySQL)

- If conn is a SQLAlchemy connection, and you’ve made some changes as part of a transaction, how do you abandon that transaction?
  - conn.rollback()
- If conn is a SQLAlchemy connection, and you’ve made some changes as part of a transaction, how do you finalize that transaction?
  - conn.commit()
- In SQL, SUM, AVG, and other operations that work over all rows (or groups of rows) are called?
  - aggregates
- What are two SQL clauses that let you filter down what rows are included based on some condition, and how do these two clauses differ?
  - WHERE and HAVING
  - WHERE is prior GROUP BY (if GROUP BY is used), so filters input rows
  - HAVING is after GROUP BY, so filters groups
- What does ACID stand for?
  - atomiticy: it all happens or nothing happens (partial progress is rolled back)
  - consistency: application invariants (like no negative bank accounts) are supported
  - isolation: others cannot see a transaction in progress (aka atomicity when talking about locks)
  - durability: once finished, it persists (even if machine crashes+restarts)
- What SQL clause lets you combine two tables?
  - JOIN
- What SQL clause lets you limit results to the first N rows?
  - LIMIT
- What SQL clause pulls together related rows?
  - GROUP BY
- What SQL clause sorts rows?
  - ORDER BY

Hadoop

- HDFS has a boss/worker architecture. What are the specific names given to the boss and workers in an HDFS cluster?
  - boss: NameNode
  - worker: DataNode
- How can you get an HDFS cluster report for `hdfs://main:9000/`?
  - `hdfs dfsadmin -fs hdfs://main:9000/ -report`
- How can you start a DataNode and connect it to a NameNode with hostname ”main” on port 9000?
  - `hdfs datanode -fs hdfs://main:9000`
- How do you format an HDFS NameNode?
  - `hdfs namenode -format`
- How do you start running an HDFS NameNode with hostname ”main” and port 9000?
  - `hdfs namenode --fs hdfs://main:9000`
- If you want to use commands like mkdir, cp, ls, cat, and du on HDFS, how do you do it?
  - `hdfs dfs -COMMAND ARGS`
  - For example, to run ”ls” on the ”data” directory: `hdfs dfs -ls hdfs://main:9000/data/`
- If you write 1 MB to a triply replicated file, how much disk I/O does that generate? What if you read 1 MB from a triply replicated file?
  - Write: 3 MB (need a copy on each machine, for safety)
  - Read: 1 MB (the three machines have identical data, so there is no point in requesting it from more than one)
- What are pipelined writes?
  - If a client wants to write 3 replicas of a block to DataNodes DN1, DN2, and DN3, it does not send the data directly to each of them.
  - Instead, it sends the data to DN1, which forwards to DN2, which forwards to DN3.
  - This prevents any one computer from having too much work (most importantly, the client isn’t doing all the work, because the client may be a weaker machine than the servers).
- What does GFS stand for?
  - Google File System
- What is a distributed analytics system that many people have started using instead of MapReduce?
  - Spark
- What is a Hadoop system similar to Google’s BigTable?
  - HBase
- What is the Hadoop system similar to GFS?
  - HDFS
- What is the technical word for breaking our data (for example, a file) into smaller pieces that are more manageable?
  - partitioning
- What is the technical word for having multiple copies of our data on different systems?
  - replication
- When a client wants to read an HDFS file, how does it interact with the NameNode and DataNodes?
  - First it tells the NameNode what it wants to read, and the NameNode responds with information about where the client can find the data.
  - Then, the client fetches the data directly from the DataNodes.
  - Data does not flow through the NameNode because otherwise having a single NameNode would be a major bottleneck.
- When Google MapReduce and BigTable were first created, where did they store their data?
  - GFS

MapReduce

- A reduce task might have the reduce function called several times. What can we say about the keys across these multiple calls for a single task?
  - The keys will be sorted
- A Spark task:
  - runs on a single CPU core
  - operates on a single partition, which is loaded entirely to memory
- If `df` is a Spark DataFrame, how would we start/stop caching it?
  - `df.cache()` # start
  - `df.unpersist()` # stop
- If you have a regular Python list, how can you convert it into a Spark RDD?
  - `rdd = sc.parallelize(YOUR LIST)`
- In MapReduce, what parameters do map and reduce functions take?
  - `def map(key, value)`
  - `def reduce(key, values)`
  - Note that with reduce, all values with the same key will go to a single reduce call on a single machine. During map, different values with the same key go to different map calls, perhaps even on different machines.
- What are some performance advantages Spark has over MapReduce?
  - intermediate data is handled better (could directly pass from one computation to the next, without materializing in HDFS)
  - a chain of RDDs lets us specify many steps to get to a desired end result, but lazy execution means that an optimizer could find more efficient ways to get to that same result
- What are two categories of Spark transformations?
  - narrow transformation: one input partition corresponds to one output partition
  - wide transformation: each output partition might need data from multiple input partitions (and vice versa)
  - if partitions are on different physical machines, wide transformations often cause network I/O (rows need to be shuffled/exchanged across the network)
- What determines the number of output files in a MapReduce job?
  - The number of reducers, because each reducer produces one output file.
- What does RDD stand for, in the context of Spark?
  - Resilient Distributed Dataset
- What is the difference between a data warehouse and data lake?
  - Both are a place where we collect data for analysis (OLAP workloads).
  - A warehouse is a database with integrated storage+computation.
  - A data lake is a combination of distributed storage (for example, HDFS) and decoupled analytics engine (for example, Spark)
- What is the relationship between actions, operations, and transformations?
  - actions and transformations are both kinds of operations
  - transformations are lazy (no work done yet), and describe how to produce an output RDD given an input RDD
  - actions actually materialize some output (on the screen, in an HDFS file, etc), triggering whatever work is necessary to compute that output. An action is often at the end of a chain of transformations.
- Why might you want bigger Spark partitions?
  - Each partition corresponds to a task, and there is an overhead to starting tasks. Bigger partitions => fewer partitions => less overhead.
- Why might you want smaller Spark partitions?
  - use more cores that are available in the cluster
  - balance work more evenly across executors
  - use less memory at any given time

Spark DataFrames

- How can you see the number of RDD partitions for a Spark DataFrame `df`?
  - `df.rdd.getNumPartitions()`
- How do you create a SparkData corresponding to a CSV at location PATH?
  - `df = spark.read.format("csv").load(PATH)`
- If you want to select the “x” column of a Spark DataFrame `df`, add 1 to it, and call the result column y, how would you do that?
  - `df.select(expr("x+1").alias("y"))`
- Say you run this: `spark.read.format("csv").load("test.csv")`
  And `test.csv` looks like this: “a,b\n1,2\n3,4”
  What will be the first row of data?
  - a,b. Because headers are off by default. Add this: `spark.read.format("csv").option("header", True).load("test.csv")`
- Say you run this: `spark.read.format("csv").option("header", True).load("test.csv")`
  And `test.csv` looks like this: “a,b\n1,2\n3,4”
  What will be the data types for the columns?
  - strings, because everything in a CSV is a string. To get ints, use this: `spark.read.format("csv").option("header", True).option("inferSchema", True).load("test.csv")`
- Spark gives an error about a file already existing when you run this: `df.select(columns).write.format("parquet").save(PATH)`
  What can you add to replace the previous file?
  - `df.select(columns).write.format("parquet").mode("overwrite").save(PATH)`

Spark SQL

- How are grouping and windowing similar and different?
  - In each case, we pull together related data into groups/partitions.
  - In grouping, there is one output row per group (e.g., an AVG stat).
  - In windowing, all the rows in the partitions can be part of the output, but they can have stats relative to the rest of the partition (e.g., a row_number() relative to the row’s position with the partition).
- How do you write a Spark DataFrame (say df) as a Hive table?
  - df.write.saveAsTable(”TABLE_NAME”)
- If df is a Spark DataFrame, what are two ways you can make it appear as a view in Hive (for the purpose of upcoming SQL queries)?
  - df.createTempView(”VIEW_NAME”)
  - df.createOrReplaceTempView(”VIEW_NAME”)
  - The former raises an exception if the view already exists.
- If tv is a table OR view in Hive, how can you get the corresponding Spark DataFrame?
  - spark.table(”tv”)
- If you have a query, say ”SELECT * FROM data”, how can you run it with Spark SQL? Assume your Spark session is in a variable ”spark”.
  - spark.sql(”SELECT * FROM data”)
- If you only want to get the unique values in column C of table T, what SQL can you write?
  - SELECT DISTINCT C FROM T;
- JOINs typically have ON clauses. Say your ON clause is something like this: `ON tbl1.colA = tbl2.colB`
  What is the technical name for this kind of JOIN?
  - ”equi join”

- Say you want to filter the results of a Spark SQL query based on a value computed with a windowing function. Why is this tricky, and what are some solutions?
  - We can’t use WHERE – that runs prior to the windowing function. We can’t use HAVING – that is only for GROUP BY.
  - Options:
    - nested SQL query (the outer query can have a WHERE clause to filter)
    - add a where filter using the DataFrame API after the SQL query (this is an excellent case for mixing and matching Spark SQL with Spark DataFrames).
  - Example of option 2:

```
spark.sql("""
SELECT area, Call_Number, row_number()
    OVER (PARTITION BY area ORDER BY Call_Number ASC) AS num
FROM calls
""").where("num <= 3").toPandas()
```

- What does spark.sql(”SHOW TABLES”) do?
  - Gives you a DataFrame listing all the tables AND all the views. You’ll need to do .show(), .toPandas(), or similar to actually see it.
- What kind of Spark operation is .count()?
  - It depends.
  - In `df.count()` where df is a DataFrame, then it is an ACTION, because it actually triggers the work to get a number back.
  - In `groups.count()` where groups is a GroupedData, then it is a TRANSFORMATION, because the counts per group might only be an intermediate result that will be further processed.
- What type does spark.sql(”QUERY”) return?
  - a Spark DataFrame

Spark Internals and Performance

- How does a distributed JOIN work with BHJ?
  - This is best for a join between a big table and small table.
  - The small table is loaded to a dictionary and sent to every machine that will work on a partition of the big table.
  - Tasks operating on partitions of the big table can independently loop over rows, matching keys with the dictionary.
  - The dictionary needs to fit in memory on every machine, so this is only a good strategy when the small table is small enough.
- How does a distributed JOIN work with SMJ?
  - Table A and table B are both hash partitioned on the column we are joining on.
  - Then we shuffle data so that we get pairs of partitions (one from A, one from B) with keys in common on the same executor.
  - The rows in these partitions are sorted by key, so tasks doing the JOIN can loop over the rows together, efficiently finding matches.
- Is data that is cached at the DISK_ONLY level serialized?
  - Yes, so it will actually use less storage space than MEMORY_ONLY (not serialized) uses of memory.
- One optimization that is part of adaptive query execution (AQE) in Spark is ”partition coallescing”. What is it?
  - For GROUP BY and similar queries, Spark needs to bring together related data with hash partitioning, using `partition_num = hash(key) % partition_count` for each row.
  - `partition_count` could be anything, but needs to be the same for the entire Spark session. It is 200 by default, which often creates many very small partitions (which is slow later).
  - The idea of partition coalescing is that the boss checks the sizes of fragments of partition data before those partitions are sent over the network.
  - It then combines some partitions together to form bigger partitions, ideally similar in size.
  - Executors are informed of the combining decisions. So there will generally be <200 partitions in the end.
- What does BHJ stand for?
  - Broadcast Hash Join
- What does bucketBy do when Spark is writing parquet files?
  - Each task that is writing data will bucket the data by key (this is basically the same as partitioning, but in Spark ”partitioning” refers to memory/RDDs, and ”bucketing” refers to files).
  - Data for each bucket is written to a different Parquet file.
  - This means that when we read the data back in, we can get an RDD that is already hash partitioned by that key.
  - If we want to do a GROUP BY (on the bucket key) followed by an aggregate, we can skip the network shuffle/exchange that would usually be necessary.
- What does SMJ stand for?
  - Shuffle Sort Merge Join
- What is `df.cache()` a shortcut for?
  - `df.persist(StorageLevel.MEMORY_ONLY)`
- What is the difference between normal partitioning and hash partitioning in Spark?
  - In normal partitioning, the rows are divided roughly evenly into partitions, and there is no guarantee about what row will be in what partition.
  - With hash partitioning, the user specifies a ”key” (usually one or more columns).
  - For each row, we compute `hash(key) % partition_count` to get a partition number.
  - This means that a partition can contain rows with different keys, but all rows for the same key will end up in the same partition.
- What is the difference between these Spark caching levels? MEMORY_ONLY vs. MEMORY_ONLY_SER
  - MEMORY_ONLY: data is represented as JVM objects on which computation can be done directly. Unfortunately, JVM objects are somewhat bloated (often using 2-5x more memory than the actual data).
  - MEMORY_ONLY_SER: data is serialized to a compact byte format. This saves space, but data partitions must be transformed to JVM objects every time computation is done on them.
- What is the difference between these two Spark caching levels, and what are the tradeoffs? MEMORY_ONLY vs. MEMORY_ONLY_2
  - MEMORY_ONLY only keeps the data in RAM on one worker.
  - MEMORY_ONLY_2 keeps it in RAM on two workers.
  - This uses more memory in the cluster (unfortunately), but it means we have more flexibility: when doing computation on an RDD partition, Spark has two workers with cached data it can choose from, so it can pick the worker that is less busy.
  - Also, if a worker with cached data crashes, we can use the other worker, avoiding a long delay to recompute the data that had been cached.
- What is the idea of partial aggregates in Spark?
  - For some aggregates (e.g., sum, count, avg), we can compute partial results prior to the exchange, often saving network I/O.
  - For example, instead of sending (“A”, 1), (“A”, 2), (“B”, 4) when we’re summing per key, a task could send this: (“A”, 3), (“B”, 4).
  - Of course, the receiving task might get “A” values from other executors too, so 3 is a partial sum, not the total sum.

Wide Tables: HBase and Cassandra

- A Spark ”Pipeline” object is an unfit model, consisting of multiple transformers and ending in an estimator. When we call ”fit” on it, we get a fitted PipelineModel. What does it contain?
  - All the transformers of Pipeline are copied to PipelineModel.
  - ”fit” is called on the estimator at the end of Pipeline, and this ”fit” returns a transformer.
  - The transformer is copied to the end of PipelineModel, such that PipelineModel is a series of transformers.
- Consider Spark’s DecisionTreeRegressionModel and DecisionTreeRegressor. Which one is fitted?
  - DecisionTreeRegressionModel is fitted. DecisionTreeRegressor is NOT fitted.
  - Usually, the class names ending in ”Model” represent fitted models in Spark.
- `dt = DecisionTreeRegressor(featuresCol="????", labelCol="????")` We can only pass in one featuresCol – how do we handle multiple features?
  - Use a vector assembler to transform a DataFrame we want to train/predict on, adding a column of vectors that contains data from multiple columns: `va = VectorAssembler(inputCols=["col1", "col2", ...], outputCol="features")`
- If we have a fitted model in Spark, how do we used it to make predictions?
  - `df_WITH_predictions = model.transform(df_WITHOUT_predictions)`. `transform` in Spark is used both for preprocessing AND prediction.
- Is this deterministic? `df.randomSplit([0.75, 0.25], seed=544)`
  - Not completely, because the deterministic split happens at the partition level.
  - If the number of partitions changes (e.g., because the number of workers changes and Spark decides a different partition count would be ideal), then the overall split won’t be deterministic.
- The traditional decision tree training algorithm considers splits on every column, for every possible split between rows (a threshold on a column is a split). Where does Spark consider splits?
  - It still looks at every column, but we configure a number of bins on each column, for example: `dt.setMaxBins(N)`.
  - A histogram is computed on this with N buckets, and the thresholds considered are those that split the buckets (so N-1 possible split points per column).
- What is an ”equi-depth” histogram?
  - Each bucket contains a similar number of data points, even if that means some buckets cover a wider range of values than others (e.g., 1-3, 3-10, 10-100, etc).
  - In contrast, most histograms have buckets with fixed intervals (e.g., 0-10, 10-20, 20-30, etc).

Cassandra Query Language (CQL)

- At what granularity can a user specify the replication factor, in both HDFS and Cassandra?
  - HDFS: the user specifies replication per file
  - Cassandra: the user specifies replication per keyspace
- For what kinds of columns are the same values shared across many rows of data?
  - Two cases:
    - partition key
    - static column
- HBase and BigTable are ”versioned sparse tables” – what does this mean?
  - There is not a fixed number of columns, and different rows can have different columns.
  - Looking up a cell is more like using a dictionary, something like `table["row:column"]`.
  - Each cell may also have multiple versions, and if we want a specific one, we can use `table["row:column:version"]`.
  - If we look at every possible combination of rows/columns in the table, many correspond to cells with no values – in a normal database, these missing values would be represented as NULL and representing those NULLs would take a little storage/memory space.
  - In versioned sparse tables, no space is used to represent the missing values (hence ”sparse”).
- How do HBase and Cassandra handle replication differently for N-times replicated data?
  - A row in HBase will belong to a single RegionServer, but that RegionServer will store that data in a N-times replicated HDFS file.
  - A row in Cassandra will belong to N different workers, each of which will store it in a single local file system (the local FS does not replicate it).
  - In other words: HBase replicates at the file system level, Cassandra replicates at the database level.
- What about a row determines which Cassandra workers are responsible for that row?
  - The value in the partition key
- What are two systems that inspired the design of Cassandra?
  - Dynamo (from Amazon)
  - BigTable (from Google)
  - Both these published papers, but not source code. Facebook designed Cassandra using the ideas in the papers, then open-sourced Cassandra.
- What determines the sort order of rows within a partition?
  - The values in the cluster key column(s)
- What happens to the regions an HBase RegionServer was responsible for if that server dies?
  - All of that dead server’s files (which contain region data) should still be safely stored in HDFS, so other healthy RegionServers can be assigned to take over these regions.
- What is a Cassandra keyspace, and where is the data it contains stored?
  - A collection of tables. Every keyspace has data spread across all workers in the cluster.
- What is the open source system most similar to Google’s BigTable?
  - HBase
- What is the primary key in a Cassandra table?
  - The combination of the partition key and cluster key.
- Why might you want all data for a particular user to live in a single row in an HBase table?
  - A single row will belong to a single RegionServer, meaning requests for that users data will go to one server instead of many.

Cassandra Partitioning

- Cassandra keyspaces contain tables. If you want to access table T in keyspace K, but you don’t want to type K.T every time you access T, what can you do?
  - Execute this first: `USE K`
- If your Cassandra keyspace K has a table T, what does this do in Spark?

  -
    ```
    spark.conf.set("spark.sql.catalog.C",
        "com.datastax.spark.connector.datasource.CassandraCatalog")
    spark.conf.set
        ("spark.sql.catalog.C.spark.cassandra.connection.host", "????")
    ```
  - It creates a Spark catalog C containing all the keyspaces of the Cassandra cluster specified on the ”host” line.
  - This means the T table can be queried from Spark as C.K.T.
- What are some advantages of prepared statements, in CQL or SQL?
  - Only need to parse the query once.
  - Easier to call (just specify a few values to plug in). We can also control default values more easily.
- What are two Cassandra functions to create a UUID value, and what is the difference between them?
  - UUID() and NOW()
  - NOW() does a better job of actually guaranteeing uniqueness because it uses several factors: MAC address (remember these are supposed to be globally unique), current time, and sequence number (in case it is called concurrently)
- What happens when you do a Cassandra INSERT with a key that already exists?
  - INSERT is really an upsert (update OR insert), so it will overwrite the previous value if already there. No error.
  - The key could be the whole primary key, or just a partition key.
- What is the cluster and partition key for this Cassandra table?

```
CREATE TABLE ???? (
  ......\
  PRIMARY KEY (x, y, z)
)
```

  - partition key: x
  - cluster key: (y,z)
  - If you wanted y to be part of the partition key, you would need to specify like this: `PRIMARY KEY ((x, y), z)`
- When can you do GROUP BY with Cassandra?
  - When you’re grouping on the columns of the primary key.
- Will this query be fast or slow in Cassandra if table T is large? `alter table T add (newcol sometype)`
  - Fast because Cassandra tables are sparse. Cassandra doesn’t need to write a bunch of NULL values to fill out the new column.

Cassandra Replication

- Given a Cassandra row’s token, how do we select a vnode to be responsible for that token? Assume single replication.
  - If a vnode has exactly the same token as the row, then that vnode owns the row. Having an exact match like this is rare.
  - Otherwise, we move to the right in the token ring, finding the first vnode token that is bigger than the row’s token.
  - We wrap around (from biggest to smallest token) if necessary.
- How do we update the token map in Cassandra? For example, we might need to insert some new vnodes to the token map if a new worker joins the cluster.
  - We can’t plan to update all the workers at once (remember, they all have a copy of the token map). At any given time, there are probably some machines offline (for example, rebooting) if we have a large cluster.
  - Instead, we update a few workers in the cluster.
  - All the workers engage in a ”gossip” protocol.
  - Once every second (or N seconds), a worker chooses a peer, and they exchange gossip about any new information their copies of the token map.
  - As with gossip or epidemics, the information will quickly spread to all the nodes that are alive.
  - If a node was down or rebooting, it will get the information soon after it comes back online and starts gossiping with its peers.
- If a client wants to interact with a Cassandra cluster, how does it start?
  - The client can connect to any worker, which will serve as a coordinator.
  - Every worker has a copy of the token map, so the coordinator will know what node (or nodes) need to be involved for operations on a given partiton of rows.

- In consistent hashing, how are tokens managed for (a) workers and (b) rows?
  - Workers: the system can choose tokens for workers (aka nodes) however it wants. Some options: randomly, or trying to space tokens evenly. Given the flexibility in the decision, whatever is decided needs to be remembered – that is, stored in some data structure. For Cassandra, that data structure is called the token map. If there are only a few thousand workers in the cluster (a lot, actually), the token map won't take that much space in RAM.
  - Rows: we want to handle MANY rows (billions, trillions, ...) in the system, so it's not reasonable to store a map of tokens for every row in a table. Instead, we take a hash of the row's key. For Cassandra, the key used is the "partition key". This means that all rows in the same partition will have the same token.
- What are a couple famous systems that use consistent hashing to partition data across workers?
  - Dynamo and Cassandra
- What are the strengths and weaknesses of hash partitioning?
  - Strength: it's highly scalable because anybody can figure out where data goes with a little math: `hash(key)%partition_count` – no need to ask a centralized boss.
  - Weakness: it's not elastic. `hash(key)%partition_count` rarely equals `hash(key)%(partition_count+1)`. This means that if you add a worker, most data in the cluster will need to be shuffled over the network to a different worker.
- What is "heterogeneity" in a cluster?
  - It means the workers are running on different hardware – some machines might be more powerful than others (more CPU, RAM, etc.).
- What is a weakness of HDFS's approach to partitioning?
  - You need a data structure somewhere that maps logical blocks to physical locations on DataNodes.
  - If there are many blocks in the system, the structure is large – in the case of HDFS, it is stored in memory in the NameNode, which is expensive.
- What is elasticity?
  - It means you can efficiently scale up/down (for example, adding or removing workers in a cluster).
  - Being efficient means we can't move the majority of the data over the network each time we gain/lose a worker.
- What is the "wrapping range" in Cassandra, and how do we handle it? Assume single replication.
  - This is the range of tokens that are bigger than any vnode token.
  - Having a row with a token in this range is a special case since we usually "walk the ring" to the right to find a vnode for a row.
  - We handle this corner case by assigning the row to the vnode with the smallest token in the cluster.
- Where is the token map stored in Cassandra?
  - We don't have a centralized boss, so every worker in the cluster stores a copy of the token map.
- Why does Cassandra have tokens for vnodes (virtual nodes) instead of just for nodes (the workers in the cluster)?
  - when we add a worker to the cluster, it's better if the new worker can take some work away from a few other nodes.
  - If we only reassign some work from just one old node, it will take longer to hand off the data, and the cluster will not be as balanced in the end.
  - some workers might have more CPU/RAM/etc.
  - We can give these stronger nodes more vnodes, which will result in them receiving more work.

Big Query 4 - Cost

- What are two BigQuery billing models?
  - Capacity and On-Demand
- What do on-demand and capacity billing charge for in BigQuery?
  - Both charge for Collosus storage. On-demand additionally charges for Collosus I/O. Capacity additionally charges for CPU+memory.
- What does it mean that a BigQuery task is preemptible?
  - It can be interrupted if resources are needed for something more important. For example, an on-demand task might be preempted by a capacity task (capacity is considered more important).
- What is a BigQuery slot?
  - A unit of resources, consisting of approximately 1 GB of RAM and 0.5 CPUs
- Which BigQuery billing model has priority access to resources (i.e., first "right of refusal")?
  - Capacity billing. This is reserved capacity, so it should be usable whenever needed. Any leftover resources (if any) are shared between on-demand jobs.

Big Query 3 - Machine Learning

- In BigQuery, how do we do non-linear transformations on our features?
  - When creating the model, add a line something like this: `TRANSFORM(POWER(x, 2), ...)`. In the parentheses, you can put things you would normally put after select: "SELECT ?????". This lets you add columns that are transformations of raw columns in the input data.
- In BigQuery, how do you handle OneHot encoding?
  - Do nothing: BigQuery will automatically do it for you if your training data has categorical features.
- In BigQuery, what are two reasons to do train/test splitting yourself (for ML) instead of letting BigQuery do it?
  - BigQuery has odd defaults depending on dataset size. For example, for <500 rows, all data is used for training.
  - BigQuery remembers the train/test split for 48 hours, so it will feel deterministic at first, but you'll be surprised later
- In BigQuery, what must be passed to `ML.EVALUATE(...)`?
  - `MODEL your_model_name, QUERY`. The query must return all the features needed to predict AND a label column indicating the correct label.
- In BigQuery, what must be passed to `ML.PREDICT(...)`?
  - `MODEL your_model_name, QUERY`. The query must have all the features needed to predict.
- In BigQuery, what must be passed to `ML.WEIGHTS(...)`?
  - `MODEL your_model_name`

Big Query 2 - Data Sources + Geo Data

- If column C is an array type, how do you get the Nth item from every array using BigQuery?
  - `SELECT C[OFFSET(N)] FROM some_table`
- If you have floats long and lat, what can you call in BigQuery to convert this to a geographic point that can be used with other spatial functions?
  - `ST_GEOGPOINT(long, lat)`
- Say you have a query like this: `SELECT * FROM myproj.mydataset.tbl1 WHERE x=3`. And you want to put the results in a new BigQuery table named `myproj.mydataset.tbl2`. How do you do this?
  - `CREATE OR REPLACE TABLE myproj.mydataset.tbl2 AS SELECT * FROM myproj.mydataset.tbl1 WHERE x=3`
- What are some examples of external tables we can register in BigQuery?
  - Parquet files in GCS, Google sheets, etc.
- What can we call if we want to get a pandas DataFrame from a BigQuery query? What about a geopandas GeoDataFrame? `bq.query(...).?????()`
  - `to_dataframe` or `to_geodataframe`
- What is the difference between ST_CENTROID_AGG(geo_column) and ST_CENTROID(geo_column)?
  - ST_CENTROID_AGG will give a single centroid for the shape representing the union of every shape in geo_column.
  - ST_CENTROID will give many centroids, one for each shape in a cell of geo_column.

Big Query 1 - Basics

- How can you activate/deactivate your GCP credentials on a GCP VM?
  - `gcloud auth`
- If you want to do a correlated cross join against a field C containing an array of structs, how do you first "expand" the arrays so each entry becomes its own row?
  - `UNNEST(C)`
- What are some factors you might consider when choosing a GCS bucket region?
  - Cost, carbon emissions, proximity to users, etc.
- What does this let you do? `%load_ext bigquery_magics`
  - It lets you write BiqQuery queries directly in Jupyter cells, like this: `%%bigquery SELECT * FROM some_table`
- What is a cross join?
  - Every row in table A is matched with every row in table B (there is no "ON" expression to limit the matches between tables).

The Cloud

- What is the newer distributed file system Google built and started using after GFS?
  - Colossus File System
- The parquet file format was inspired by the ______ format developed at Google, which was eventually replaced internally at Google with the _______ format.
  - ColumnIO, Capacitor
- What does cloud "lock in" mean?
  - It means it is difficult to move to another cloud (or out of the cloud), even if if that's desirable in the long run. Reasons for the difficulty:
    - need to pay a high network egress cost to get your data out
    - your code might be written to specifically use an API not available in other clouds
- IaaS: Infrastructure as a Service. You rent low-level hardware-like resources (e.g., VMs, virtual disks, private networks, etc). You use these to deploy systems yourself. For example, you might launch a bunch of VMs and start running HDFS DataNodes on them.
- PaaS: Platform as a Service. The cloud provider runs a specific system on your behalf. It could be something they built (e.g., S3), or something open source (e.g., MySQL). This is usually more convenient, but you often pay more for that convenience. (Also BigQuery)
- When a column contains large strings, and the same values show up multiple times, we can replace those strings with shorter numeric codes. We can use the numeric codes to lookup the original strings as needed. This is called dictionary encoding.
- When a value in a column is repeated many times consecutively, we can record just the value and count (instead of many copies of the value). This is called run-length encoding.

Streaming: Spark Concepts

- Most systems use replication to implement fault tolerance. Why doesn't Spark streaming need to do that?
  - Spark streaming is based on RDDs, which describe how to compute the data on demand, as needed. So as long as data can be re-read from the source, we can re-materialize any output previously computed for an RDD.
- Say you have "A ???? JOIN B", where A is static and B is a stream. What kind of JOINs are supported?
  - INNER and RIGHT are supported.
  - LEFT is NOT supported. The reason is that a left join requires us to output a row of A with NULL columns from B if there is no B row matching. But we can never conclude there is no B row matching because B keeps growing, and we don't know the future.
- Spark streaming supports most of the operations supported by batch jobs, but a "pivot" operation is an example of one that only works for batch (not streaming). Why doesn't it work?
  - If you allowed pivot for streaming, there are some new inputs that could appear that would cause an additional column to be added to the output. But Spark DataFrames (whether streaming or not) need to have a well-defined set of columns so that SQL queries and other operations can operate on them.
- What are some examples of sources/sinks in Spark streaming?
  - Kafka, Parquet files in HDFS, Cassandra, console, etc.
- What are the three output modes available for Spark streaming queries?
  - append, complete, update

Spark

- a simple (default) reduce task can combine output
  of multiple mappers to a single file
- MapReduce shuffles the data, bringing together intermediate outputs from mappers
  with the same key. All data with the same key is passed to a single "reduce" call. A
  reduce task will generally have reduce called many times with different keys.
- SQL => MapReduce
  - Map Phase
    - SELECT, WHERE, GROUP BY, JOIN
  - Shuffle Phase (bringing related data to same place)
    - ORDER BY, GROUP BY, JOIN
  - Reduce Phase
    - SELECT, AGGREGATE, HAVING, JOIN
  - MapReduce is more flexible. (for example, how to do a
    GROUP BY where one row goes to mutliple groups in SQL?)
  - Projects like HiveQL try to make MapReduce more accessible.
- Data Locality: Avoid Network Transfers
  - HDFS DataNodes and MapReduce executor run on same machines
  - Try to run mappers on machine where DataNode has needed data. Uses disk but not network.
- Speed of `spark.read.format`:
  - manually specify types < parquet < only read header < reads whole csv to guess types
- `df.cache()` is a shortcut for `df.persist(StorageLevel.MEMORY_ONLY)`.

Spark ML Lib

- models that do good on train data but bad on validation/test data have "overfitted"
- in Spark, a fitted model is a Transformer, and an unfitted model is an Estimator
- PLANET: Parallel Learner for Assembling Numerous Ensemble Trees
  - originally implemented as MapReduce jobs
  - Spark DecisionTreeRegressor and DecisionTreeClassifier use it too
- Hybrid Approach
  - in-memory splitting for nodes with few enough rows to fit in worker memory
  - simplified (fewer split points) and distributed approach for nodes with lots of data
- number of bins must be greater than the biggest number of unique values in any categorical column

Cassandra

- a region is assigned to ONE HBase "RegionServer" at any given time
- Rows are never split across regions
- HBase only support single-row transactions
- ideally a RegionServer is placed on the same machine as a DataNode holding most of its data
- HBase Storage Strategy:
  - store new data in memory until we have a lot of data
  - then do one big write to disk
- Cassandra clusters have many worker nodes
  - No centralized boss node (unlike HDFS, Spark)
  - Not necessarily same data center (could be geographically distributed)
  - Clusters are called "rings" because some nodes are defined to be "adjacent"
  - Ring organization doesn't necessarily correspond to network topology
- Getting Conflicting Versions
  - send all version back to the client, which will need specialized conflict resolution code
  - automatically combine them into a new row, and write that (if possible) to all replicas
  - Dynamo supports both. Cassandra uses second approach.

Exam Questions

- Which webhdfs operation makes use of HTTP redirects (status code 307)?
  - OPEN
- For which node type is a format necessary before the first launch?
  - NameNode only
- FLOPS => KFLOPS => MFLOPS => GFLOPS => TFLOPS => PFLOPS => EFLOPS
  - 1000
- A Cassandra table has three columns: X (first column, a partition key), Y (second column, a cluster key), and Z (third column, regular column). You insert these rows: (1, 1, 1) (1, 2, 3) (1, 2, 4) How many rows will be in the table?
  - 2, because rows that share the same combination of partition and clustering key are considered the same row.
- The data owned by a node in a decision tree has 10 rows and 5 columns of feature data. No value appears more than once in the table. How many different ways are there to split this node?
  - 45 (9 gaps * 5 unique columns)
- To what is the space usage of the token map proportional?
  - Only the number of nodes in a Cassandra ring
- The wrapping range of a Cassandra ring consists of tokens that are larger than biggest vnode token.
- Sequential I/O is generally faster than random 1/0 for block devices. The difference is larger for HDD.
- Which of the following Spark SQL clauses does NOT use hash partitioning? `WHERE`
- Spark uses the PLANET algorithm to train decision trees (DTs). The type of job that runs on a set of DT nodes depends on whether those nodes have few enough rows to run the in-memory algorithm. For the in-memory case, the job uses hash partitioning. What does it partition rows on?
  - the DT node to which the row belongs
- An HBase "compaction" primarily involves what operation?
  - merge sorting small files

- Which system uses pipelined writes to send data to all the workers that will store a new piece of data?
  - HDFS
- How does HBase assign data to RegionServers? Assume we are using 3x replication.
  - each region will belong to one RegionServer (Data replication is handled by HDFS, not HBase itself)

TopHat

1. Two consumer groups are both subscribed to the same topic. The first group has 3 consumer processes and the second group has 4 consumer processes. All consumers simply print consumed messages. When a message is written to the topic, how many times will it be printed in total (across all processes)?
Answer: 2

2. For which resource does Google charge under both BigQuery billing models? (capacity and on-demand)
Answer: Collosus Storage

3. What do we specify in the options for a "CREATE OR REPLACE MODEL" statement with regard to feature/label columns?
Answer: just label columns

4. For what type of cloud service is "lock in" usually a bigger concern?
Answer: PaaS

5. Spark is maintaining a count for an interval starting at 2pm. At what time could Spark reasonably discard the running count for events occurring in this window?
(animals.withWatermark("timestamp", "1 hours")
.groupBy(window("timestamp", "2 hours"))
.count())
Answer: 5PM

6. Is it stateless?
```
SELECT (x+y) AS total
FROM mystream;
```
Answer: Yes. (SELECT, FILTER, MAP operations that don't rely on past data.) (Keywords that indicate stateful are GROUP BY, COUNT, SUM, AVG, WINDOW, TUMBLE, HOP, SESSION, OVER, WITH WATERMARK, MAX)

7. Suppose RF=4 and min in-sync replicas are 2. There are currently 3 in-sync replicas and one lagging. A message is written to two replicas (leader and one follower). Is it committed?
Answer: No

8. If producers request strong acks and have retry, but you don't suppress duplicates, what are the semantics of a successful ack?
Answer: At least once

9. What is the FIRST print where we'll see the assignment of more than zero partitions?
```
consumer = KafkaConsumer(bootstrap_servers=[broker])
print(consumer.assignment()) # X
consumer.subscribe(["odd_nums"])
print(consumer.assignment()) # Y
_ = consumer.poll(1000)
print(consumer.assignment()) # Z
```
Answer: Z

10. Six messages are produced in the following order:
topic=A, key=X, value=8
topic=A, key=Y, value=8
topic=B, key=Y, value=8
topic=B, key=Y, value=7
topic=C, value=9
topic=C, value=10
What can we say about the order in which these will be consumed?
Answer: msg 3 before msg 4

11. To what is the space usage of the token map proportional?
Answer: Only the number of nodes in Cassandra ring

12. The wrapping range of a Cassandra ring consists of tokens that are...
Answer: bigger than biggest vnode token

13. Within a static column of a Cassandra table, there is at most one value corresponding to each partition key.

14. A feature column has these numbers, in the range of 0 to 100:
1, 2, 8, 9, 15, 16, 80, 90
What histogram would a Spark decision tree preferably compute for 4 bins?
Answer: 0-5, 5-10, 10-20, 20-100

14. What is a method you WILL NOT be using when working with Spark models?
Answer: predict

16. Your MapReduce job runs. There are M map tasks and R reduce tasks. The map tasks collectively emit X unique keys. The reduce tasks collectively emit Y unique keys. How many times is your reduce(...) function called?
Answer: X

17. Your MapReduce job runs. There are M map tasks and R reduce tasks. The map tasks collectively emit X unique keys. The reduce tasks collectively emit Y unique keys. How many output files are written to HDFS?
Answer: R

18. Your table has 100 rows. You want to calculate statistics related to each unique value of column A in the table. Column A contains 20 unique values. The result set of your query contains 20 rows. What kind of operations seem to have been performed?
Answer: The rows were grouped by A, then an aggregate function was applied to each group.

Quiz

1. Which JOIN types have an "ON" clause?
LEFT JOIN, RIGHT JOIN, INNER JOIN

2. In BigQuery, if you want to do a correlated cross join against a field C containing an array of structs, how do you first "expand" the arrays so each entry becomes its own row?
UNNEST(C)

3. What is the execution engine used inside BigQuery?
Dremel

4. In which system do read quorum configurations often require the involvement of multiple replicas to serve a read operation?
Cassandra

5. When is the spark.sql.shuffle.partitions configuration most important?
Streaming

6. Rows written to a Kafka topic use column X as the key. You try to create a Spark consumer that involves doing a "GROUP BY" on a different column, Y. What will happen?
The output will be correct because Spark will hash partition the data based on Y.

7. Where are topic partitions stored in Kafka?
Brokers

8. What can a consumer call when it wants partitions automatically assigned by Kafka?
Subscribe

9. Kafka message X is produced before message Y. You're asked whether X will be consumed before Y. What is the minimum you need to know about these messages to answer?
Topic and key

10. How does a broadcast hash join work in Spark?
only the small table is sent to every executor

11. A Spark Pipeline object p has a transformer X followed by an estimator Y. You call p.fit(...) to get a PipelineModel back. What will be in the PipelineModel?
X and the object returned by Y.fit(...)

12. The PLANET algorithm runs two kinds of jobs when constructing a decision tree. The are for:
Nodes that fit well in memory on one worker and bigger nodes

13. How does HBase assign data to RegionServers? Assume we are using 3x replication.
each region will belong to one RegionServer

14. Which system was most influenced by Amazon Dynamo?
Cassandra

15. Which operations are examples of actions?
count (on a DataFrame or grouped data), toPandas

16. What would be an example of a "wide" transformation in Spark?
sorting a DataFrame by column C

17. You have 10 Spark executors, each with 20 CPU cores and 64 GB of RAM. How many tasks can execute simultaneously?
200

18. Which one is usually faster if you want to query from it?
Hive table

19. Which Spark options serialize the cached data?
MEMORY_ONLY_SER, MEMORY_ONLY_SER_2
DISK_ONLY, DISK_ONLY_2

20. A file system on /dev/sda1 has these directories:
/a
/b
A file system on /dev/sda2 has these files/directories:
/x/y/z.txt
Say /dev/sda1 is already mounted as the root file system. Then, you run this:
mount /dev/sda2 /b
What absolute path can you use to open the existing z.txt?
/b/x/y/z.txt

21. What is the biggest limitation of MapReduce?
saving intermediate data after each job is inefficient

22. If you're running out of memory when running a Spark job, what is most likely to help?
Same data, but more partitions of it

23. In a networking stack, what is something that HTTP allows you to specify that lower layers do not directly support?
Resource

Exam

If you want to run BigQuery over data in the Capacitor format, how should you add tables to your dataset?
() load job
(b) external table (In original format)

Consider the following Kafka messages. What can we guarantee about which messages will go to the same partition?
topic="W", key="Z", value=""
topic="W", key="X", value="X"
topic="X", key="Z", value="Z"
(A) 1 and 2 will go to the same partition
(B) 1 and 3 will go to the same partition
(C) 2 and 3 will go to the same partition
() We can't guarantee anything

What does COS stand for, in the context of Google's cloud?
() Container-Optimized OS
(B) Cloud Operating System
(C) Container Orchestration Service

HDFS is most similar to which proprietary system?
(A) Artifact Registry
(B) BigQuery
() Colossus
(D) Dynamo

In Kafka, we can separately configure the replication factor and min in-sync replicas. When is a message considered committed?
(A) it has been written to min in-sync replicas
() it has been written to all the in-sync replicas
(C) it has been written to a number of replicas equal to the replication factor

In the clause FROM first, second, what kind of JOIN is done between the tables?
() CROSS
(B) INNER
(C) LEFT
(D) RIGHT

The query engine for BigQuery is internally based on what system?
(A) GFS
() Dremel
(C) Spark
(D) MapReduce

You attempt a Cassandra INSERT with a primary key that is already used by one row that is already in the table (the table was created with a cluster key). What happens?
(A) the insert is ignored
(B) an error is raised
() previous row is updated
(D) there will be two rows with the same primary key

The PLANET algorithm (implemented by Spark) sometimes collects all the rows corresponding to a single node of a decision tree on a single machine.
True

You want to do a streaming GROUP BY with Spark, using a Kafka topic as a source. However, the column you want to group by in Spark is different than the column used to set the key for the messages in the Kafka topic. Does Spark support such a query?
True

A Cassandra table has three columns: X (first column, a partition key), Y (second column, a cluster key), and Z (third column, regular column). You insert these rows:
- (1,1,1)
- (1,2,4)
- (1,3,4)

How many rows will be in the table?
(A) 0
(B) 1
(C) 2
() 3
(E) 4

Which non-cloud platform is most similar to Google's BigQuery?
() Spark
(B) Cassandra
(C) Kafka
(D) HBase
(E) BigTable

Say your bloom filter uses 2 hash functions and 10 bits. The bits contain this:
1000000001
hash1(X)%10=0, hash2(X)%10=9
Was X previously inserted?
(A) yes
(B) no
() maybe

Spark uses the PLANET algorithm to train decision trees (DTs). The type of job that runs on a set of DT nodes depends on whether those nodes have few enough rows to run the in-memory algorithm. For the in-memory case, the job uses hash partitioning. What does it partition rows on?
(A) the first column in the row
() the DT node to which the row belongs
(C) the value in the column on which we're splitting

Which of the following Spark SQL clauses does NOT use hash partitioning?
(A) JOIN
() WHERE
(C) GROUP BY

Cassandra uses consistent hashing. For what does Cassandra use a hash function to get a token on the token ring?
() only for data
(B) only for nodes
(C) for both data and nodes