# CSP 554 – Big Data Technologies

## Fall 2023 – All Sections

### Midterm Exam - Part 2

**Part I** – Short Answer (Show Points/Results) – 5 points each, 30 points total

1. Given a 514MB file residing on HDFS configured for a block size of 128MB, what is the total number of blocks used with a replication factor of 3? If the replication factor is increased by 1, how much additional storage is wasted/inefficiently allocated?

Answer- Number of blocks used with a replication factor of 3

Number of blocks = File size / Block size \* Replication factor

Number of blocks = 514MB / 128MB \* 3 = 12

Additional storage wasted with a replication factor of 4

Additional storage wasted = (New replication factor - Old replication factor) \* Number of blocks \* Block size

Additional storage wasted = (4 - 3) \* 12 \* 128MB = 512MB

Explanation-When a file is stored in HDFS, it is divided into blocks of the specified size. In this case, the block size is 128MB. The file size is 514MB, so the file is divided into 4 blocks.

The replication factor determines how many copies of each block are stored on different datanodes. In this case, the replication factor is 3, so each block is stored on 3 different datanodes.

If the replication factor is increased to 4, then each block will be stored on 4 different datanodes. This means that an additional 512MB of storage space will be used to store the additional copies of the blocks.

In summary

The total number of blocks used with a replication factor of 3 is 12.

Increasing the replication factor from 3 to 4 will result in 512MB of additional storage being wasted/inefficiently allocated.

1. How often is a custom Combiner task within the Map-Reduce framework guaranteed to execute? Under what conditions will it execute? Describe in terms of memory buffers/spill files.

Answer- A custom Combiner task within the Map-Reduce framework is not guaranteed to execute. The Hadoop framework may decide to not invoke the Combiner for a variety of reasons, such as:

* The input data is too small to benefit from combining.
* The Combiner's output is not significantly smaller than its input.
* The Combiner throws an exception.

If the Combiner is invoked, it will be executed once for each partition of the input data. A partition is a subset of the input data that is assigned to a single Map task.

The Combiner is typically executed in memory. However, if the Combiner's output is too large to fit in memory, it will be spilled to disk. Spilling is the process of writing intermediate results to disk.

The conditions under which the Combiner will execute are as follows:

* The input data is large enough to benefit from combining.
* The Combiner's output is significantly smaller than its input.
* The Combiner does not throw an exception.

The Combiner functions best in general when the input data is vast and the output is considerably smaller than the input. If the Combiner is used to sum the values for a specific key, for instance, and there are numerous records with the same key, the output will be significantly smaller than the input.

In conclusion, the execution of a custom Combiner task within the Map-Reduce framework is not guaranteed.

• The Combiner is normally run once for each input data partition.

• The Combiner generally runs in memory, but if the output is too huge for memory, it may spill to disk.

• If the input data is large enough to benefit from combining, the combinator will run;

1. Describe the process of writing a new file to HDFS in terms of interactions between Clients, Name Nodes, and Data Nodes. Assume a default replication factor of 3 and a single Name Node instance.

Answer-

Client

Initiates write operation by creating a `DistributedFileSystem` object.

Obtains DataNode locations from NameNode.

Opens an output stream to the new file.

Writes data to the output stream.

NameNode

Receives write request from client.

Determines DataNodes responsible for storing the new file.

Provides client with DataNode locations.

DataNodes

Receive data from client.

Store data in their local file systems.

Replicate data to other DataNodes (default replication factor of 3).

Example

Client creates a `DistributedFileSystem` object.

NameNode determines that DataNodes `dn1`, `dn2`, and `dn3` are responsible for storing the new file.

NameNode provides client with locations of `dn1`, `dn2`, and `dn3`.

Client opens an output stream to the new file.

Client writes data to the output stream.

Client closes the output stream.

DataNode `dn1` stores the first 33 bytes of data in its local file system.

DataNode `dn2` stores the next 33 bytes of data in its local file system.

DataNode `dn3` stores the remaining 34 bytes of data in its local file system.

DataNode `dn1` replicates the first 33 bytes of data to DataNode `dn2`.

DataNode `dn2` replicates the first 33 bytes of data to DataNode `dn3`.

DataNode `dn2` replicates the next 33 bytes of data to DataNode `dn1`.

DataNode `dn3` replicates the next 33 bytes of data to DataNode `dn2`.

DataNode `dn3` replicates the remaining 34 bytes of data to DataNode `dn1`.

Result

New file is stored on DataNodes `dn1`, `dn2`, and `dn3`.

Each DataNode stores a complete copy of the file.

Result

The new file is stored on DataNodes dn1, dn2, and dn3. Each DataNode stores a complete copy of the file.

1. Given the Pig commands of LOAD/STORE and FILTER/GROUP BY, which pair are functional transformations and which ones trigger evaluation? If a Relation contains the following: UserId, UserAge, UserIncome - which of the above should be used to obtain all records of users with an age >= 18?

Answer-

Functional transformations

LOAD

STORE

Transformations that trigger evaluation

FILTER

GROUP BY

Explanation

Functional transformations do not cause any data to be processed. They simply create or modify the structure of the data. For example, the LOAD command creates a relation from a data source, and the STORE command stores a relation to a data source.

Transformations that trigger evaluation cause data to be processed. For example, the FILTER command filters the data in a relation, and the GROUP BY command groups the data in a relation.

To obtain all records of users with an age >= 18, the following Pig commands could be used:

filtered\_users = FILTER users BY UserAge >= 18;

STORE filtered\_users INTO 'filtered\_users';

The FILTER command would filter the users relation to include only records of users with an age >= 18. The STORE command would store the filtered data in a new relation called filtered\_users.

In summary

Functional transformations do not cause any data to be processed.

Transformations that trigger evaluation cause data to be processed.

The FILTER command can be used to obtain all records of users with an age >= 18.

1. How does the CSV file Reader in the Hadoop libraries process a record (row) that spans the block boundary in HDFS? Explain in terms of the difference between Map-Reduce splits and HDFS blocks.

Answer - When a CSV file reader in the Hadoop libraries encounters a record that spans the block boundary in HDFS, it uses a technique called "record merging" to reconstruct the complete record.

Map-Reduce splits and HDFS blocks

A Map-Reduce split is a logical division of a file that is assigned to a Map task. A HDFS block is a physical division of a file that is stored on a DataNode. Map-Reduce splits are not necessarily aligned with HDFS blocks. This means that a Map-Reduce split can span multiple HDFS blocks, or a single HDFS block can be divided into multiple Map-Reduce splits.

Record merging

When a CSV file reader encounters a record that spans the block boundary, it reads the portion of the record that is in the current block. The reader then looks ahead to the next block to see if the rest of the record is in that block. If the rest of the record is in the next block, the reader reads the remaining portion of the record and merges it with the portion of the record that was read from the previous block. The reader then returns the complete record to the Map task.

Example

Consider a CSV file that contains the following records:

1,John Doe,100

2,Jane Doe,200

The file is divided into two HDFS blocks, with the first block containing the first record and the second block containing the second record.

The Map task is assigned a split that spans both blocks. The CSV file reader first reads the first record from the first block. The reader then looks ahead to the second block and sees that the second record begins in the second block. The reader then reads the remaining portion of the second record from the second block. The reader then merges the two portions of the record together and returns the complete record to the Map task.

In summary

The CSV file reader in the Hadoop libraries uses record merging to reconstruct records that span the block boundary in HDFS. This is done by reading the portion of the record that is in the current block and then looking ahead to the next block to see if the rest of the record is in that block. If the rest of the record is in the next block, the reader reads the remaining portion of the record and merges it with the portion of the record that was read from the previous block. The reader then returns the complete record to the Map task.

1. Given a Map-Reduce task where we are provided with records (rows) containing: UserId, TransactionAmount - define one (or more) Map-Reduce tasks which would provide a per-user average spending amount: UserId, AverageSpend. Specify the Key-Value Pairs for all tasks.

Answer- Map Task

* Input Key-Value Pair: <UserId, TransactionAmount>
* Output Key-Value Pair: <UserId, TransactionAmount>

Reduce Task

* Input Key-Value Pair: <UserId, [TransactionAmount, TransactionAmount, ...]>
* Output Key-Value Pair: <UserId, AverageSpend>

The Map task simply emits the input key-value pair unchanged. This is necessary because the Reduce task needs to receive all of the transaction amounts for each user in order to calculate the average spending amount.

The Reduce task receives a list of transaction amounts for each user. The Reduce task then calculates the average spending amount by summing the transaction amounts and dividing by the number of transactions. The Reduce task then emits a key-value pair with the user ID as the key and the average spending amount as the value.

Example

* Input data:
  + <1, 100>
  + <1, 200>
  + <2, 50>
  + <2, 100>
* Map output:
  + <1, 100>
  + <1, 200>
  + <2, 50>
  + <2, 100>
* Reduce input:
  + <1, [100, 200]>
  + <2, [50, 100]>
* Reduce output:
  + <1, 150>
  + <2, 75>

The resulting key-value pairs are:

* <1, 150>
* <2, 75>

These key-value pairs represent the per-user average spending amounts.

**Part II** – Long Answer (Show Reasoning/Calculations) – 10 points each, 20 points total

1. Given the following key-value pair results from a Map step of a Map-Reduce job, what would be the resulting key-value pairs that would be the input to a Reduce job after the Shuffle/Sort step: <hello, 1>, <there, 1>, <hello, 1>, <again, 1> ? Would this result be different if a Combiner step was run? If so, provide the resulting key-value pairs in that case.

Answer-

In the Map-Reduce paradigm, the Shuffle/Sort step groups key-value pairs by their keys before passing them to the Reducer. In wer case, we have the following key-value pairs from the Map step:

1. <hello, 1>

2. <there, 1>

3. <hello, 1>

4. <again, 1>

After the Shuffle/Sort step, the key-value pairs are sorted by key and grouped together:

1. <again, 1>

2. <hello, 1>

3. <hello, 1>

4. <there, 1>

If a Combiner step is run before the Reducer, its purpose is to perform a local aggregation on the Mapper's output to reduce data transfer during the Shuffle/Sort step. In this case, the Combiner would combine the values for the same keys within a Mapper's output. However, the Combiner's output is not guaranteed to be the same as the Reducer's input because the Reducer will perform another round of aggregation on the shuffled and sorted data.

With a Combiner, the output would look like this:

1. <again, 1>

2. <hello, 2>

3. <there, 1>

These key-value pairs are the input to the Reducer, which will further aggregate and process the data. The use of a Combiner can help reduce network traffic and improve the efficiency of the Map-Reduce job by performing some initial aggregation at the Mapper level. In summary

The Shuffle/Sort step groups key-value pairs with the same key together and sorts them by key. The Combiner can be used to reduce the amount of data that needs to be shuffled and sorted.

1. Which storage format, row-oriented or column-oriented, is best suited to a website order management system where new transactions are added periodically? Would the same format also be best suited to a machine learning system which selects features for model fitting? Explain wer reasoning.

Answer- The choice between row-oriented and column-oriented storage formats depends on the specific requirements and access patterns of the data in data-driven systems.

Website Order Management System (WOMS)

* Scenario: A website order management system where new transactions are added periodically.
* Storage format: Row-oriented.
* Reason: Row-oriented storage is optimized for write-heavy workloads. When a new order is placed, the entire row of data for that order can be written to the database in a single operation. This is much more efficient than writing individual columns of data to a column-oriented database.

Machine Learning System (MLS)

* Scenario: A machine learning system that selects features for model fitting.
* Storage format: Column-oriented.
* Reason: Column-oriented storage is optimized for read-heavy workloads. When a machine learning system is selecting features, it needs to read data from a large number of columns. With a column-oriented database, only the columns that are needed for feature selection need to be read. This can significantly improve the performance of the machine learning system.

In summary

* Row-oriented storage is typically more suitable for applications with frequent updates, such as website order management systems.
* Column-oriented storage is typically more suitable for applications that focus on feature selection and analysis of large datasets, such as machine learning systems.

In summary, the choice between row-oriented and column-oriented storage formats depends on the primary use case. For a website order management system with frequent updates, row-oriented storage is preferred. For a machine learning system focusing on feature selection and analysis of large datasets, column-oriented storage is more efficient due to its read-heavy nature and better support for data analytics and processing.