Hadoop

What is Hadoop ?

=====Hadoop is a distributed framework which has distributed storage and distributed processing mechanism. It uses HDFS for distributed storage and Mapreduce for distributed Processing. Hadoop is highly durable and easily scalable.

Components of Hadoop?

Hadoop architecture consist of NameNode,Standby Name Node, Datanodes,Journal Nodes and Zookeeper.

Block size in Hadoop HDFS ?

128mb Default Block in Hadoop 2.0

64 mb in Hadoop 1.0

How fault tolerance is managed in Hadoop?

Hadoop has replication factor of 3 in which each block gets replicated in case of any node failure(Faults) can be handled.

What is Safe Mode in Hadoop ?

When Name Node gets restarted. To create FS image (To serve client) name node goes to safe mode and do checkpointing (Merge Edit Log and FS image) and leave safe mode. (Name Node in un serviceable that time)

What is Heart Beat interval?

3 Seconds Data sends heart beat to the Name Node.

What is FS image and Edit Log?

FS image in Name stores the Meta information of every transaction and about datanodes. Edit log is a temporary file which helps the checkpointing for every 1 hour default it . Secondary Name Node helps for checkpointing.

Under and Over replication in Hadoop

As default replication factor is 3. If any of the Nodes Goes down. The blocks in that node has only two replication (less than 3) in other Nodes which we call under replication.

If dead Nodes comes back which has blocks replicated already (more than 4) considering this as dead node . – Over replication.

Split brain scenario in hadoop

If dead Name Node becomes active and start writing the edit logs to the Journal Node and also when the recent active name node do the same .

Speculative execution in hadoop

In Hadoop during Speculative Execution, a certain number of duplicate tasks are launched. On a different slave node, multiple copies of the same map or reduce task can be executed using Speculative Execution

High availability and federation in Hadoop

The high availability feature in Hadoop **ensures the availability of the Hadoop cluster without any downtime**, even in unfavorable conditions like NameNode failure, DataNode failure, machine crash, etc. It means if the machine crashes, data will be accessible from another path.

HDFS Federation improves the existing HDFS architecture through a clear separation of namespace and storage, enabling generic block storage layer. It enables support for multiple namespaces in the cluster to improve scalability and isolation.

Rack awareness in hadoop

Rack Awareness in Hadoop is the concept to choose a nearby data node (closest to the client which has raised the Read/Write request), thereby reducing the network traffic. Hadoop supports the configuration of rack awareness to ensure the placement of one replica of the data block on a different rack

Small File issues in hadoop

Storing lot of small files which are extremely smaller than the block size cannot be efficiently handled by HDFS. Reading through small files involve lots of seeks and lots of hopping between data node to data node, which is inturn inefficient data processing.

sqoop and performance Tuning

What is Sqoop and its Features

Apache Sqoop is a big data tool for **transferring data between Hadoop and relational database servers**. Sqoop is used to transfer data from RDBMS (relational database management system) like MySQL and Oracle to HDFS (Hadoop Distributed File System).

Default mappers in sqoop

4

--split-by columnname

Controlling Imports

Types

Where – Can be applied over the sqoop command using –where “city=’Chennai’”

Sqoop import –connect < --username> --password –table –where “city=’Chennai’ and id>3”

Query - --query “select * from table where $CONDITIONS”

Sqoop import –connect < --username> --password –table –query

Default file format

Text File format

Sqoop eval

It **allows users to execute user-defined queries against respective database servers and preview the result in the console**. So, the user can expect the resultant table data to import. Using eval, we can evaluate any type of SQL query that can be either DDL or DML

Sqoop Incremental Append Imports

Sqoop incremental append imports helps to imports only the newly added data at the RDBMS table. We can give handle it using –incremental append –check-column id –last-

value 10

We can handle incremental using , -- int,date,timestamp columns as incremental column

Sqoop Incremental Modified Imports

Sqoop also has feature to import modified imports by running two jobs, one to import the data and other to apply the changes to the existing part file. Using –incremental last modified –check-column date –last-value 2022-03-04 –merge-key id

Here merge key helps to compare the data and for last value possible column types could be date or timestamp

Sqoop Job incremental Automation

We can automate sqoop incremental jobs using
Sqoop job –create jobname

Sqoop job –exec jobname

Sqoop job –list or –show to see the job details

For every import it will save the last value of itself. It would help for the next imports. You can see the jobs details under .sqoop/metastore.db.script

Password file

We can manage to have password file consist of Password can be passed to the sqoop command using --password-file file:///Path

Sqoop import –connect ←username> --password-file hdfs:/user/cloudera/passfile

Sqoop password authentication

We can JCEKS for password authentication (Java crypto Encrypted key store).But In my case I have used password file

Sqoop Staging Exports

Sqoop helps to export to the staging table first incase of any issues, only staging tables get effected instead of target sql table during the partial export issues.

Sqoop Import all tables

Yes sqoop can import all tables from a database and exclude few if required.

sqoop import-all-tables --connect jdbc:mysql://localhost/DATABASE --username root --password hadoop --warehouse-dir '/user/cloudera/alltables' –exclude-tables <TAB1>,<TAB2>

Mapcolumn java in Sqoop

MapColumn java is used in Sqoop to cast the columns during the Imports. For Example. During avro import , date columns does not support so we use map column java to import as a string using

--map-column-java “tdate=string”

Sqoop Hcatalogue

HCatalog is a table and the storage management service for Apache Hadoop, which enables the users with different data processing tools such as Hive, Pig, MapReduce to read and write data on a grid with ease.

SQOOP_HOME/bin/sqoop import --connect <jdbc-url> -table <table-name> --hcatalog-table emp <other sqoop options>

Performance (Fetch Size,Mappers,direct Mode, Exports Performance)

Import performance –

Fetch Size – Sqoop import default 1000 rows as at a time. It could be increased using –fetch-size 5000

Sqoop import connect <> --fetch-size 5000

Mappers -- Default numbers of mappers used in sqoop in 4 however we can increase like -m 20 depends on the connection availability in RDBMS. So that it would performance boundary query and divide data to each mapper accordingly.

Direct Mode – By using direct mode (--direct) in sqoop , it would be uses its native library for data import for mysql and Postgres instead of JDBC driver which enables for faster imports, however its available only for mysql and postgresql

Sqoop import connect <> --fetch-size 5000 --direct

Export Performance –

Sqoop export -Dsqoop.export.statements.per.transaction=1000 Dsqoop.export.records.per.statement =1000 --connect

Types of Serialized File formats

TextFile -- Readable and huge on size and its sqoop default file format.

Sequence file – Java file format helps for faster processing for Mapreduce.

Parquet – Columnar Format and has predicate pushdown. Used for Target system for faster query and support snappy compression of about 60-80 percent of compression ratio.

ORC -- -Columnar format, high on compression of about 90% due to ZLIB inbuilt used for legacy historical storage data

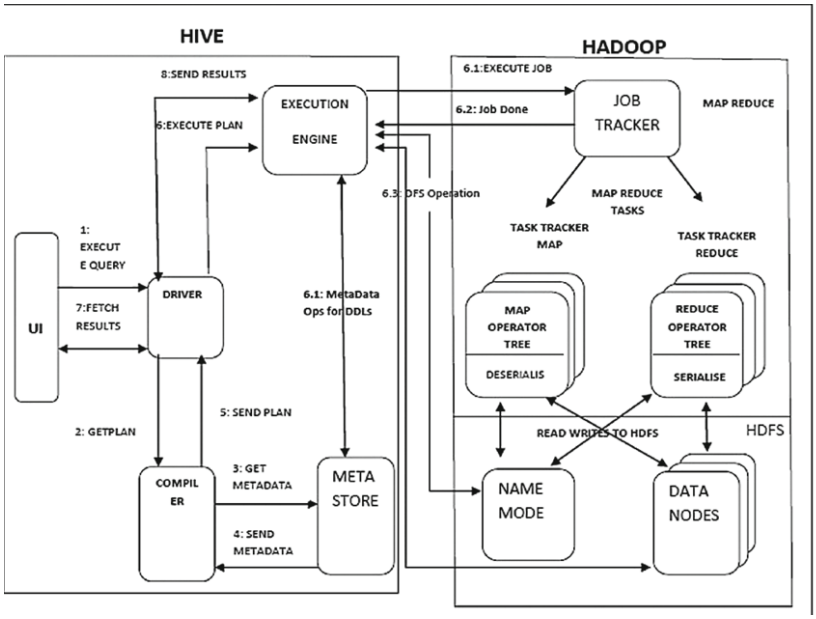
AVRO ---- Supports 50-60 compression ratio and its row file format , and its support schema evolution.

hive and performance Tuning

What is Hive ?

Hive allows users to read, write, and manage petabytes of data using SQL. Hive is built on top of Apache Hadoop, which is an open-source framework used to efficiently store and process large datasets.

Hive architecture



Once Hive query is triggered. First Driver takes the query and send to compiler to check syntax and semantic checks and sends to the execution Engine.

Then execution Engine query the data from Hadoop using Yarn and show you the data

Hive metastore

The Hive metastore is simply a relational database. It stores metadata related to the tables/schemas you create to easily query big data stored in HDFS. When you create a new Hive table, the information related to the schema (column names, data types) is stored in the Hive metastore relational database

Hive can use two metastore

Embedded Metastore - Derby Default metastore

Remote Metastore - SQL metastore

Types of Data Loads to hive table

We can load from Edge Node as well HDFS. Using

Edge Node load—Load data local inpath

Hdfs load – load data inpath

Types of Table and its difference:

We have two types of table – Managed and External

Managed Table – As every table is pointed to a hdfs directory, if we drop Managed table , the back end directory associated with the table gets **deleted.. Temporary Purpose**

External Table – As every table is pointed to a hdfs directory, if we drop External table , the back end directory do Exists and Only table gets deleted from metastore. **.. Permanent Purpose**

Also multiple tables can be created under same location (Managed /External)

Hive warehouse

Hive warehouse is like hive's personal room for creation of Directories and Table. If we create hive table without location the directory gets created in the name of the table and my table gets pointed to it and we also

Hive vs Sql

Sql	Hive
ETL	ELT
Load time parsing (Schema on Write)	Query time Parsing (Schema on read)
No serialized data support	Good serialized data support
Good for small data	Good for Large data
Licenced	Free

Sort By vs Order By

We should use SORT BY instead of ORDER BY when we have to sort huge datasets because SORT BY clause sorts the data using multiple reducers whereas ORDER BY sorts all of the data together using a single reducer. Therefore, using ORDER BY against a large number of inputs will take a lot of time to execute.

Default delimiter Hive

^A

Schema evolution in hive

Yes it Supported in hive provided if the table is created with Avro file format as we have avsc file managed the schema

Partitioning and Types

Hive supports partitions by diving the data into multiple folders or sub folders for efficient querying. Low cardinal columns (Low distinct columns) are chosen to perform partitions.

Partitions are folder

Types – Static and Dynamic

Bucketing

Bucketing is used for High cardinal Columns for query performance

We can use bucketing along with partitions.

Bucketing also helps for Join during Bucket to Bucket Join.

Buckets are Files

Default partition size

Hive create default partition of 100. To increase the number of partitions

hive.exec.max.dynamic.partitions

DML in hive

DML in hive is possible (Insert, Update ,delete) . But the table should be **transactional(true)** table along with **orc file format** and should be **bucketed Table**.

```
create table HiveTest1 (EmployeeID Int,FirstName String,Designation String, Salary
Int,Department String)
clustered by (department) into 3 buckets
stored as orc
TBLPROPERTIES ('transactional'='true');
```

Msck repair in hive

Msck repair is applied for the hive table to recognize the partitions which is created by the other table in the same location

Performance Tuning in hive

Vectorization -- hive.vectorized.execution.enabled=true . .Hive usually process 1 row at a time. But it process 1024 rows at a time by enabling vectorization

Parallel execution –Hive.exec.parallel=true, by default false, If we have sub queries, hive have to wait to process first sub query and to handle second sub query , but by enabling this property both the sub queries gets executed parallely.

Partitions and bucketing – Dividing the data into multiple chunks to ensure the query performance

Mapside Join – When two joins are getting joined (1 large table and 1 small table). Its better to perform map join with small table so the whole small data would traverse to the cache of all the datanodes of the First table for easy join.

Tez/Spark Engine- set hive.execution.engine=tez;

Sort Merge Bucket Join

It would help to join Bucket to Bucket of two tables but these two tables should have equal number of buckets.

Indexing

One of the Hive query optimization methods is Hive index. Hive index is used to speed up the access of a column or set of columns in a Hive database because with the use of index the database system does not need to read all rows in the table to find the data that one has selected.

CREATE INDEX table01_index ON TABLE table01 (column2) AS 'COMPACT';

Hive Views:

A view allows a query to be saved and treated like a table. It is a logical construct, as it does not store data like a table. In other words, materialized views are not currently supported by Hive.

Limitations of Hive

Not suitable for Unstructured or Small data

Real time streaming is not much supported

Updates are not easy (DML) as it have to support some conditions

Rename table in hive

ALTER TABLE db_name.test_1 RENAME TO db_name.test_2;

Change datatype of column in hive

ALTER TABLE tableA CHANGE ts BIGINT;

Serde in hive

A SerDe allows Hive to read in data from a table, and write it back out to HDFS in any custom format.

Avro,parquet,orc serde Example.

UDFS in hive

User Defined Functions, also known as UDF, allow you to create custom functions to process records or groups of records. Hive comes with a comprehensive library of functions. There are however some omissions, and some specific cases for which UDFs are the solution

Hive unix run (Hive -e,hive -f)

Hive -e “create table ; insert into ; drop table”

Hive -f data.hql

Thrift server

a service which permits a remote client to submit requests to Hive, with the help of a variety of programming languages, as well as retrieve results, is what we call HiveServer. Since, It is built on Apache Thrift, that's why we call it Thrift server sometimes

spark

What is Apache Spark?

Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size.

How is Apache Spark different from MapReduce?

Spark does process in Inmemory without hitting the harddisk however Mapreduce hits the harddisk for every action of Map Reduce Jobs till the Completion.

Lazy evaluation in Spark?

Till the action is triggered none of the transformations takes place. So spark is Lazy in nature.

Libraries in Spark or components of the Spark?

RDD,Dataframe,MLLIB,Streaming,Structured Streaming,Graphx

Ways to Process data spark?

RDD and Dataframes.

SchemaRDD in Spark RDD?

If any of the schema is imposed on Spark rdd using case class or NamedTuple we call that as schema rdd.

Save Modes in Spark

When I write the data using dataframe we have to specify save modes.

Error (Default)

Append

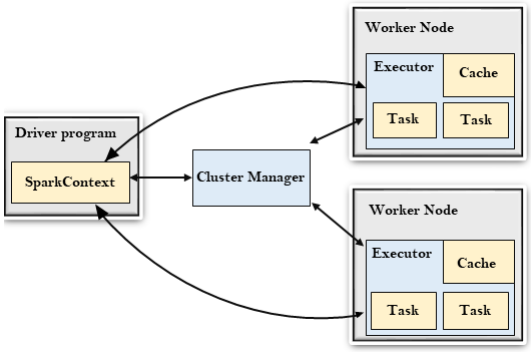
Overwrite

Ignore

Default file format supported in Spark

Parquet is default file format. Spark also supports JSON,CSV,ORC, for other file formats we need extra dependency.

Explain how Spark runs applications with the help of its architecture.



When Spark submit is done, Spark context gets created in which it runs Driver program which submits the Job to the cluster manager which inturn Communicate with the each worker node where

What are the different cluster managers available in Apache Spark?

Standalone cluster manager, Hadoop YARN and Apache Mesos

Resilient Distributed Datasets in Spark?

RDDs are the main logical data units in Spark. They are a distributed collection of objects, which are stored in memory or on disks of different machines of a cluster. A single RDD can be divided into multiple logical partitions so that these partitions can be stored and processed on different machines of a cluster

Shuffling in Spark? When does it occur?

The Spark SQL shuffle is a mechanism for redistributing or re-partitioning data so that the data is grouped differently across partitions, based on your data size you may need to reduce or increase the number of partitions of RDD/DataFrame using spark.

Shuffling is a mechanism helps to equalize the partition size during the repartitioning.

Session vs sqlcontext

Spark Session is SQL object to perform SQL or dataframe operations

Spark Context is RDD object to perform RDD operations.

Narrow and wide transformation in spark

Narrow transformations **transform data without any shuffle involved**. – Filter.FlatMap

Wider transformations are **the result of groupByKey() and reduceByKey() functions** and transform data with shuffles.

Cache and Persist

Caching or Persisting mean caching the Intermediate processed data to reduce the redundancy of the same execution.

Cache and persist do the same but cached data cannot be uncached . Persisted data can be unpersist and it has difference storage levels.

Different levels of persistence in Spark?

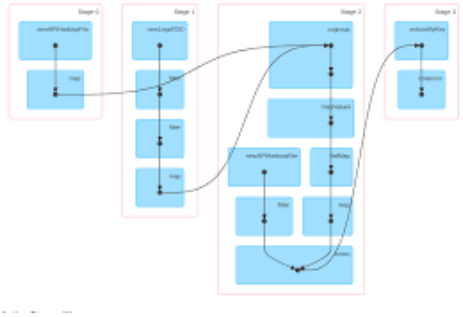
Storage Level	Description
MEMORY_ONLY	It stores the RDD as deserialized Java objects in the JVM. This is the default level. If the RDD doesn't fit in memory, some partitions will not be cached and recomputed each time they're needed.
MEMORY_AND_DISK	It stores the RDD as deserialized Java objects in the JVM. If the RDD doesn't fit in memory, store the partitions that don't fit on disk, and read them from there when they're needed.
MEMORY_ONLY_SER (Java and Scala)	It stores RDD as serialized Java objects (i.e. one-byte array per partition). This is generally more space-efficient than deserialized objects.
MEMORY_AND_DISK_SER (Java and Scala)	It is similar to MEMORY_ONLY_SER, but spill partitions that don't fit in memory to disk instead of recomputing them.
DISK_ONLY	It stores the RDD partitions only on disk.
MEMORY_ONLY_2, MEMORY_AND_DISK_2, etc.	It is the same as the levels above, but replicate each partition on two cluster nodes.
OFF_HEAP (experimental)	It is similar to MEMORY_ONLY_SER, but store the data in off-heap memory. The off-heap memory must be enabled.

Array Column processing in Spark

We use explode to process or filter array column which we call as Flattening of data.

DAG in spark

DAG(Direct Acyclic Graph) is a graphical representation of Transformations and actions in Spark. There are finitely many vertices and edges, where each edge directed from one vertex to another.



programmatically specify a schema for DataFrame

Struct Type

```
val schema = StructType(Array(
   StructField("firstname",StringType,true),
   StructField("middlename",StringType,true),
   StructField("lastname",StringType,true),
   StructField("id", StringType, true),
   StructField("gender", StringType, true),
   StructField("salary", IntegerType, true)
 ))
```

Catalyst Optimizer

The Catalyst optimizer is a crucial component of Apache Spark. It **optimizes structural queries – expressed in SQL, or via the DataFrame/Dataset APIs – which can reduce the runtime of programs and save costs**.

When we perform SQL operations catalyse optimizer design logical plan to execute the query with high performance.

Broadcast variable in Spark

Broadcasting a variable enables the variable data to cached in all the necessary worker nodes which helps for join instead of shuffling the data which is similar to Map Side Join in hive

Types of joins in Spark

Inner

An INNER JOIN is **such type of join that returns all rows from both the participating tables where the key record of one table is equal to the key records of another table**

Left

The LEFT JOIN command **returns all rows from the left table, and the matching rows from the right table**

Right

This join **returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join**.

Full

FULL JOIN **creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN**

Left anti

There are two types of anti joins: A left anti join :**This join returns rows in the left table that have no matching rows in the right table**

Left Semi

A semi-join between two tables **returns rows that match an EXISTS subquery without duplicating rows from the left side of the predicate when multiple rows on the right side satisfy the criteria of the subquery**

Cross Join

the CROSS JOIN is used **to combine each row of the first table with each row of the**

second table. It is also known as the Cartesian join since it returns the Cartesian product of the sets of rows from the joined tables.

Join Doc —Double Click to Download and Scroll Down

difference between map and flatMap

Map and FlatMap are the functions applied on Spark rdd or datasets.

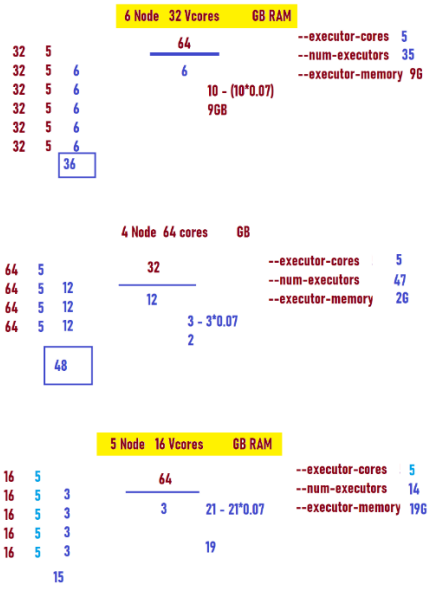
Map – Applies the function or operation

FlatMap – Applies the function or operation along with Flattening the data.

Memory Tuning in Spark

Spark executor Cores,Num executors and Executor Memory calculation

During Spark Submit we have to tune the spark memory for efficient resource allocation.



Dynamic Resource Allocation in Spark

Dynamic Resource Allocation. **Spark provides a mechanism to dynamically adjust the resources your application occupies based on the workload**. This means that your application may give resources back to the cluster if they are no longer used and request them again later when there is demand.

```
spark.dynamicAllocation.enabled=true
spark.shuffle.service.enabled=true
spark.dynamicAllocation.enabled=true
spark.dynamicAllocation.shuffleTracking.enabled=true
spark.dynamicAllocation.minExecutors=10
spark.dynamicAllocation.maxExecutors=500
spark-submit --master yarn-cluster \
    --driver-cores 2 \
    --driver-memory 2G \
    --num-executors 10 \
    --executor-cores 5 \
    --executor-memory 2G \
    --conf spark.dynamicAllocation.minExecutors=5 \
    --conf spark.dynamicAllocation.maxExecutors=30 \
    --conf spark.dynamicAllocation.initialExecutors=10 \ # same as --num-executors 10
  --conf  spark.dynamicAllocation.enabled=true
--conf  spark.shuffle.service.enabled=true


    --class com.spark.sql.jdbc.SparkDFtoOracle2 \
```

Coalesce vs Repartition

Both helps to change the Partition Size. Size of the partition size is Dynamic.

Coalesce helps to decrease the partition size within the node without shuffling the data.

Repartition helps to do both Increase or decrease the partition size but it shuffles the data between nodes to Maintain equal partition size

Spark Lineage

Lineage is an RDD process **to reconstruct lost partitions**. Spark not replicate the data in memory, if data lost, Rdd use linege to rebuild lost data. Each RDD remembers how the RDD build from other datasets.

Spark Hive integration

To Read –

Val df = spaı k.sql("select * from db.tablename")

Or

Val df = spark.table("db.tablename")

To write

Df.write.format("parquet").saveAsTable("db.tablename")

Stages and Tasks in Spark

For every actions in Spark that creates the stages with its respective Tasks.

Every transformation creates a Stage and for every shuffle operation a Stage gets created.

Collection of Tasks means stage –

Number of Tasks ==== Number of Partitions of that RDD.

Accumulators in Spark

Accumulators are the variable which are used aggregate information from Executors. Every Executor provides statistics which gets collected in the Accumulators.

Its not aggregate the data (Not Business Logic) , it shows aggregation of statistics of the Job from every executor of the Worker Nodes.

It's a counter on every Executor

Spark Submit Deployment

Code would be exported as a jar in scala or Py File in Python and would be placed over the cluster and gets triggered using Spark Submit command.

Spark-submit –master yarn –deploy-mode cluster –executor-cores 6 –executor-memory 5g –num-executors 40 –class pack.obj /home/cloudera/ss-0.0.1-SNAPSHOT.jar

Debugging in Spark

Spark Job Debugging or Status is Known **from Spark UI or Using Yarn Logs.**

Deploy Mode --- Client and Cluster

In Client Mode --- The driver program runs in Client Node (Edge Node)

In cluster Mode – the driver program runs in one of the Data Nodes to track the jobs.

Spark checkpointing

Spark checkpointing mean writing the intermediate transformed data to the Harddisk

Eg- df.persist(StorageLevel.DISK_ONLY)

Performance Tuning in Spark

Memory Tuning (Formula)

Cache Persist (Persisting in better storage level)

Dynamic Resource Allocation

Coalesce and Repartitions

Joins Utilization whenever required

Using serialized data (Especially Parquet)

Broadcast variables

Data skewness in Spark (What is skewness and solution for it)

If One executor got the lot of Load in a work node after the data shuffling we call it as a data skewness.

	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	S
+details	2018/02/18 15:12:11	47 h	199/200			

Example. I have used Monotonically Increasing Id I have removed it and used zipWithIndex solved the Problem.

It also can be solved using Spark Salting.

df.withColumn("salt_random_column", (rand * n).cast(IntegerType)) // n is the size of partition you'd like to have

.groupBy(groupByFields, "salt_random_column")

.agg(aggFields)

.groupBy(groupByFields)

.agg(aggFields)

groupBykey vs Reduceby key?

Both reduceByKey and groupByKey result in wide transformations which means both triggers a shuffle operation. The key difference between reduceByKey and groupByKey is that **reduceByKey does a map side combine and groupByKey does not do a map side combine**

Logical vs Physical Plan

Physical Plan—Expected memory might be needed before the action

Logical Plan --- Query design for the transformations

Spark Partitions.

Similar to Hive partitions but it can be achieved using **partitionBy** api

Df.write.format(“parquet”).partitionBy(“country”,”spendby”).save(“”)

Spark Bucketing

File creation for high cardinal data where partitions cannot help.

Similar to Hive bucketing

Helps for Bucket to Bucket Join.

Df.write.format(“parquet”) .bucketBy(4, "id").partitionBy(“country”,”spendby”).save(“”)

After 2.4.3 version of Spark

Spark Windowing

Spark Window functions are used to calculate results such as the rank, row number e.t.c over a range of input rows


```
val windowf = Window
  .partitionBy($"department")
  .orderBy($"assetValue" desc)

df.withColumn("col3", dense_rank() over windowf)
  .filter("col3=2")
```

Spark UDF

If any of the API's are not available for doing a transformation. We create a custom User Defined Function to solve the problem but (Its very very rare)

Out of Memory Issues

Can be drive out of memory

Collect() --- Better to use cluster mode.

Broadcast large data – avoid if the data is huge

Can be Executor Out of Memory

Yarn Memory Overhead – Increase over memory manually -- spark.executor.memoryOverhead=8G

Improper core allocation or memory ---- Please allocated max 5-6 Executor Cores

Dataframe vs Dataset

Dataframe are SQL api of Spark. It uses Kryo serializer at the backend. It uses Tungsten for optimization

Dataset is kind of combination of RDD and Dataframe. It uses Encoders for optimization.

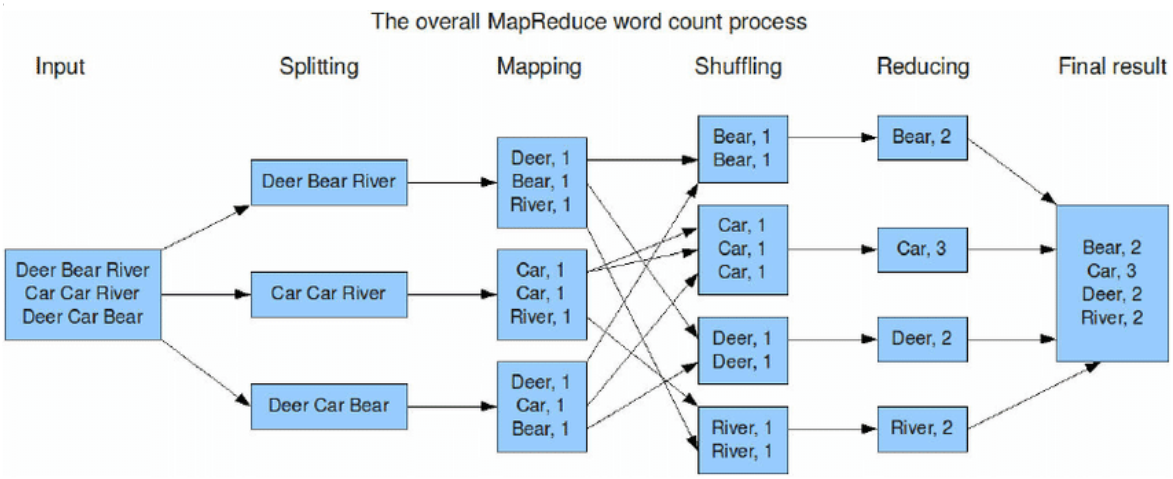
Build Tool

Instead of adding jars to the project we can provide any build tool reference to download directory from repository.

General Questions

MR and Yarn

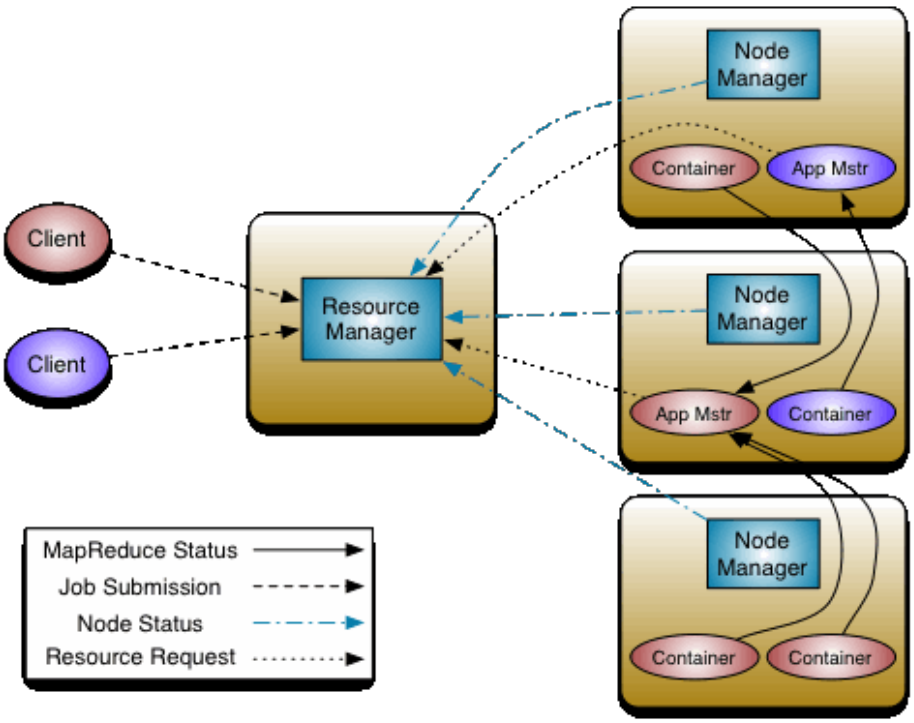
What is Mapreduce?



Spilling In Mapreduce

A spill is **when a mapper's output exceeds the amount of memory which was allocated for the MapReduce task**. Spilling happens when there is not enough memory to fit all the mapper output

Yarn in Hadoop 2.0



When we submit Spark or MR job Client submits to RM, in which RM contacts Name Node for Data Location and activates its respective

Node Manager to Create containers to run MR jobs. RM also create AM to track the job in any of the nodes.

Block vs Input split

Block—Physical Block

Input split – Logical Block

General

Cluster size

On Prem - 120-150 Node Cluster – 64 GB RAM, 32 Vcores- 4 TB Hardisk

Cloud – 25 Node Cluster R Series in AWS EMR for every job

Team size

5-8 Team members including me

Schedule

We use nifi to schedule as we use nifi for many usecases for which we go with Nifi. Before it used to be Control M and Cron

Data size

Our Job Runs Weekly, we process approximate 2.5 to 3TB customer data.

What you do Daily (Day to Day activities)

We follow Agile Methodology

we have scrum Master who create user stories for the whole team in the JIRA board.

We have sprint for 2 Weeks, after scrum master assigns the Stories we progress each day report in the scrum call.

Unit Testing

We recently started implementing Unit testing using Scala Test and Pyspark code with unittest Module and we started checking necessary assertions.

(Ensure that you already left the job say that was implemented mid way)

GIT

We use GIT to handle version Controlling of Code.

We have all the Code checked in to the GIT with Master and Developer branch.

For every implementation we create our own branch

CLONE IT → DO the implementation → Check out the code to the Branch and create necessary pull request to the Team to Merge the Code.

Incase of any conflicts we team sit together to Resolve it.

https://youtu.be/OAC0XUA3a2E