**Hadoop Questions**

1. **HDFS**

[Hadoop Distributed File System](https://www.dezyre.com/hadoop-course/hdfs) is the java based file system for scalable and reliable storage of large datasets. Data in HDFS is stored in the form of blocks and it operates on the Master Slave Architecture.

1. **Hadoop MapReduce**

This is a java based programming paradigm of Hadoop framework that provides scalability across various Hadoop clusters. [MapReduce](https://www.dezyre.com/hadoop-course/mapreduce) distributes the workload into various tasks that can run in parallel. Hadoop jobs perform 2 separate tasks- job. The map job breaks down the data sets into key-value pairs or tuples. The reduce job then takes the output of the map job and combines the data tuples to into smaller set of tuples. The reduce job is always performed after the map job is executed.

1. **What is a block and block scanner in HDFS?**

Block - The minimum amount of data that can be read or written is generally referred to as a “block” in HDFS. The default size of a block in HDFS is 64MB.

Block Scanner - Block Scanner tracks the list of blocks present on a DataNode and verifies them to find any kind of checksum errors. Block Scanners use a throttling mechanism to reserve disk bandwidth on the datanode.

1. **Explain the difference between NameNode, Backup Node and Checkpoint**

**NameNode**:

NameNode is at the heart of the HDFS file system which manages the metadata i.e. the data of the files is not stored on the NameNode but rather it has the directory tree of all the files present in the HDFS file system on a hadoop cluster. NameNode uses two files for the namespace-

fsimage file- It keeps track of the latest checkpoint of the namespace.

edits file-It is a log of changes that have been made to the namespace since checkpoint.

**Checkpoint Node-**

<http://blog.cloudera.com/blog/2014/03/a-guide-to-checkpointing-in-hadoop/>

Checkpoint Node keeps track of the latest checkpoint in a directory that has same structure as that of NameNode’s directory. Checkpoint node creates checkpoints for the namespace at regular intervals by downloading the edits and fsimage file from the NameNode and merging it locally. The new image is then again updated back to the active NameNode.

**BackupNode:**

Backup Node also provides check pointing functionality like that of the checkpoint node but it also maintains its up-to-date in-memory copy of the file system namespace that is in sync with the active NameNode.

1. **Explain what happens if during the PUT operation, HDFS block is assigned a replication factor 1 instead of the default value 3**.

Replication factor is a property of HDFS that can be set accordingly for the entire cluster to adjust the number of times the blocks are to be replicated to ensure high data availability. For every block that is stored in HDFS, the cluster will have n-1 duplicated blocks. So, if the replication factor during the PUT operation is set to 1 instead of the default value 3, then it will have a single copy of data. Under these circumstances when the replication factor is set to 1, if the DataNode crashes under any circumstances, then only single copy of the data would be lost.

1. **Whenever a client submits a hadoop job, who receives it?**

NameNode receives the Hadoop job which then looks for the data requested by the client and provides the block information. Job Tracker takes care of resource allocation of the hadoop job to ensure timely completion.

1. **Explain the usage of Context Object**.

Context Object is used to help the mapper interact with other Hadoop systems. Context Object can be used for updating counters, to report the progress and to provide any application level status updates. Context Object has the configuration details for the job and also interfaces, that helps it to generating the output.

1. **Explain about the partitioning, shuffle and sort phase**

**Shuffle Phase-**Once the first map tasks are completed, the nodes continue to perform several other map tasks and also exchange the intermediate outputs with the reducers as required. This process of moving the intermediate outputs of map tasks to the reducer is referred to as Shuffling.

**Sort Phase**- Hadoop MapReduce automatically sorts the set of intermediate keys on a single node before they are given as input to the reducer.

**Partitioning Phase-**The process that determines which intermediate keys and value will be received by each reducer instance is referred to as partitioning. The destination partition is same for any key irrespective of the mapper instance that generated it.

1. **How to write a custom partitioner for a Hadoop MapReduce job?**

Steps to write a Custom Partitioner for a Hadoop MapReduce Job-

* A new class must be created that extends the pre-defined Partitioner Class.
* getPartition method of the Partitioner class must be overridden.
* The custom partitioner to the job can be added as a config file in the wrapper which runs Hadoop MapReduce or the custom partitioner can be added to the job by using the set method of the partitioner class.
* **What are the different operational commands in HBase at record level and table level?**
* Record Level Operational Commands in HBase are –put, get, increment, scan and delete.
* Table Level Operational Commands in HBase are-describe, list, drop, disable and scan.

1. **What is Row Key?**

Every row in an HBase table has a unique identifier known as RowKey. It is used for grouping cells logically and it ensures that all cells that have the same RowKeys are co-located on the same server. RowKey is internally regarded as a byte array.

1. **Explain the difference between RDBMS data model and HBase data model.**

RDBMS is a schema based database whereas HBase is schema less data model.

RDBMS does not have support for in-built partitioning whereas in HBase there is automated partitioning.

RDBMS stores normalized data whereas HBase stores de-normalized data.

1. **Explain about the different catalog tables in HBase?**

The two important catalog tables in HBase, are ROOT and META. ROOT table tracks where the META table is and META table stores all the regions in the system.

1. **What is column families? What happens if you alter the block size of Column Family on an already populated database?**

The logical deviation of data is represented through a key known as column Family. Column families consist of the basic unit of physical storage on which compression features can be applied. In an already populated database, when the block size of column family is altered, the old data will remain within the old block size whereas the new data that comes in will take the new block size. When compaction takes place, the old data will take the new block size so that the existing data is read correctly.

1. **Explain the difference between HBase and Hive.**

HBase and Hive both are completely different hadoop based technologies-Hive is a data warehouse infrastructure on top of Hadoop whereas HBase is a NoSQL key value store that runs on top of Hadoop. Hive helps SQL savvy people to run MapReduce jobs whereas HBase supports 4 primary operations-put, get, scan and delete. HBase is ideal for real time querying of big data where Hive is an ideal choice for analytical querying of data collected over period of time.

1. **Explain the process of row deletion in HBase.**

On issuing a delete command in HBase through the HBase client, data is not actually deleted from the cells but rather the cells are made invisible by setting a tombstone marker. The deleted cells are removed at regular intervals during compaction.

1. **What are the different types of tombstone markers in HBase for deletion?**

There are 3 different types of tombstone markers in HBase for deletion-

1)Family Delete Marker- This markers marks all columns for a column family.

2)Version Delete Marker-This marker marks a single version of a column.

3)Column Delete Marker-This markers marks all the versions of a column.

1. **Is it possible to do an incremental import using Sqoop?**

Yes, Sqoop supports two types of incremental imports-

1)Append

2)Last Modified

To insert only rows Append should be used in import command and for inserting the rows and also updating Last-Modified should be used in the import command.

1. **Explain about the core components of Flume.**

The core components of Flume are –

Event- The single log entry or unit of data that is transported.

Source- This is the component through which data enters Flume workflows.

Sink-It is responsible for transporting data to the desired destination.

Channel- it is the duct between the Sink and Source.

Agent- Any JVM that runs Flume.

Client- The component that transmits event to the source that operates with the agent.

1. **Does Flume provide 100% reliability to the data flow?**

Yes, Apache Flume provides end to end reliability because of its transactional approach in data flow

1. **How can Flume be used with HBase?**

Apache Flume can be used with HBase using one of the two HBase sinks –

* HBaseSink (org.apache.flume.sink.hbase.HBaseSink) supports secure HBase clusters and also the novel HBase IPC that was introduced in the version HBase 0.96.
* AsyncHBaseSink (org.apache.flume.sink.hbase.AsyncHBaseSink) has better performance than HBase sink as it can easily make non-blocking calls to HBase.

1. **Explain about the different channel types in Flume. Which channel type is faster?**

The 3 different built in channel types available in Flume are-

MEMORY Channel – Events are read from the source into memory and passed to the sink.

JDBC Channel – JDBC Channel stores the events in an embedded Derby database.

FILE Channel –File Channel writes the contents to a file on the file system after reading the event from a source. The file is deleted only  after the contents are successfully delivered to the sink.

MEMORY Channel is the fastest channel among the three however has the risk of data loss. The channel that you choose completely depends on the nature of the big data application and the value of each event.

1. **Which is the reliable channel in Flume to ensure that there is no data loss?**

FILE Channel is the most reliable channel among the 3 channels JDBC, FILE and MEMORY.

1. **Explain about the replication and multiplexing selectors in Flume.**

Channel Selectors are used to handle multiple channels. Based on the Flume header value, an event can be written just to a single channel or to multiple channels. If a channel selector is not specified to the source then by default it is the Replicating selector. Using the replicating selector, the same event is written to all the channels in the source’s channels list. Multiplexing channel selector is used when the application has to send different events to different channels.

1. **How multi-hop agent can be setup in Flume?**

Avro RPC Bridge mechanism is used to setup Multi-hop agent in Apache Flume.

1. **Differentiate between FileSink and FileRollSink**

The major difference between HDFS FileSink and FileRollSink is that HDFS File Sink writes the events into the Hadoop Distributed File System (HDFS) whereas File Roll Sink stores the events into the local file system.

1. **Can Apache Kafka be used without Zookeeper?**

It is not possible to use Apache Kafka without Zookeeper because if the Zookeeper is down Kafka cannot serve client request.

1. **Explain about ZooKeeper in Kafka**

Apache Kafka uses ZooKeeper to be a highly distributed and scalable system. Zookeeper is used by Kafka to store various configurations and use them across the hadoop cluster in a distributed manner. To achieve distributed-ness, configurations are distributed and replicated throughout the leader and follower nodes in the ZooKeeper ensemble. We cannot directly connect to Kafka by bye-passing ZooKeeper because if the ZooKeeper is down it will not be able to serve the client request.

1. **Explain how Zookeeper works**

ZooKeeper is referred to as the King of Coordination and distributed applications use ZooKeeper to store and facilitate important configuration information updates. ZooKeeper works by coordinating the processes of distributed applications. ZooKeeper is a robust replicated synchronization service with eventual consistency. A set of nodes is known as an ensemble and persisted data is distributed between multiple nodes.

3 or more independent servers collectively form a ZooKeeper cluster and elect a master. One client connects to any of the specific server and migrates if a particular node fails. The ensemble of ZooKeeper nodes is alive till the majority of nods are working. The master node in ZooKeeper is dynamically selected by the consensus within the ensemble so if the master node fails then the role of master node will migrate to another node which is selected dynamically. Writes are linear and reads are concurrent in ZooKeeper.

1. **List some examples of Zookeeper use cases.**

* Found by Elastic uses Zookeeper comprehensively for resource allocation, leader election, high priority notifications and discovery. The entire service of Found built up of various systems that read and write to   Zookeeper.
* Apache Kafka that depends on ZooKeeper is used by LinkedIn
* Storm that relies on ZooKeeper is used by popular companies like Groupon and Twitter.

1. **What problems can be addressed by using Zookeeper?**

In the development of distributed systems, creating own protocols for coordinating the hadoop cluster results in failure and frustration for the developers. The architecture of a distributed system can be prone to deadlocks, inconsistency and race conditions. This leads to various difficulties in making the hadoop cluster fast, reliable and scalable. To address all such problems, Apache ZooKeeper can be used as a coordination service to write correct distributed applications without having to reinvent the wheel from the beginning.

1. **Explain about the SMB Join in Hive.**

In SMB join in Hive, each mapper reads a bucket from the first table and the corresponding bucket from the second table and then a merge sort join is performed. Sort Merge Bucket (SMB) join in hive is mainly used as there is no limit on file or partition or table join. SMB join can best be used when the tables are large. In SMB join the columns are bucketed and sorted using the join columns. All tables should have the same number of buckets in SMB join.

1. **What is SerDe in Hive? How can you write your own custom SerDe?**

SerDe is a Serializer DeSerializer. Hive uses SerDe to read and write data from tables. Generally, users prefer to write a Deserializer instead of a SerDe as they want to read their own data format rather than writing to it. If the SerDe supports DDL i.e. basically SerDe with parameterized columns and different column types, the users can implement a Protocol based DynamicSerDe rather than writing the SerDe from scratch.

1. **What is Apache Hadoop YARN?**

YARN is a powerful and efficient feature rolled out as a part of Hadoop 2.0.YARN is a large scale distributed system for running big data applications.

1. **What are the additional benefits YARN brings in to Hadoop?**

* Effective utilization of the resources as multiple applications can be run in YARN all sharing a common resource. In Hadoop MapReduce there are separate slots for Map and Reduce tasks whereas in YARN there is no fixed slot. The same container can be used for Map and Reduce tasks leading to better utilization.
* YARN is backward compatible so all the existing MapReduce jobs.
* Using YARN, one can even run applications that are not based on the MaReduce model

1. **Explain the differences between Hadoop 1.x and Hadoop 2.x**

* In Hadoop 1.x, MapReduce is responsible for both processing and cluster management whereas in Hadoop 2.x processing is taken care of by other processing models and YARN is responsible for cluster management.
* Hadoop 2.x scales better when compared to Hadoop 1.x with close to 10000 nodes per cluster.
* Hadoop 1.x has single point of failure problem and whenever the NameNode fails it has to be recovered manually. However, in case of Hadoop 2.x StandBy NameNode overcomes the SPOF problem and whenever the NameNode fails it is configured for automatic recovery.
* Hadoop 1.x works on the concept of slots whereas Hadoop 2.x works on the concept of containers and can also run generic tasks.

1. **What are the core changes in Hadoop 2.0?**

Hadoop 2.x provides an upgrade to Hadoop 1.x in terms of resource management, scheduling and the manner in which execution occurs. In Hadoop 2.x the cluster resource management capabilities work in isolation from the MapReduce specific programming logic. This helps Hadoop to share resources dynamically between multiple parallel processing frameworks like Impala and the core MapReduce component. Hadoop 2.x Hadoop 2.x allows workable and fine grained resource configuration leading to efficient and better cluster utilization so that the application can scale to process larger number of jobs

1. **Explain the major difference between HDFS block and InputSplit.**

In simple terms, block is the physical representation of data while split is the logical representation of data present in the block. Split acts a s an intermediary between block and mapper.  
Suppose we have two blocks:  
**Block 1: ii nntteell  
Block 2: Ii ppaatt**  
Now, considering the map, it will read first block from ii till ll, but does not know how to process the second block at the same time. Here comes Split into play, which will form a logical group of Block1 and Block 2 as a single block.

It then forms key-value pair using inputformat and records reader and sends map for further processing With inputsplit, if you have limited resources, you can increase the split size to limit the number of maps. For instance, if there are 10 blocks of 640MB (64MB each) and there are limited resources, you can assign ‘split size’ as 128MB. This will form a logical group of 128MB, with only 5 maps executing at a time.

However, if the ‘split size’ property is set to false, whole file will form one inputsplit and is processed by single map, consuming more time when the file is bigger.

1. **What is Speculative Execution in Hadoop?**

One limitation of Hadoop is that by distributing the tasks on several nodes, there are chances that few slow nodes limit the rest of the program. Tehre are various reasons for the tasks to be slow, which are sometimes not easy to detect. Instead of identifying and fixing the slow-running tasks, Hadoop tries to detect when the task runs slower than expected and then launches other equivalent task as backup. This backup mechanism in Hadoop is Speculative Execution.

It creates a duplicate task on another disk. The same input can be processed multiple times in parallel. When most tasks in a job comes to completion, the speculative execution mechanism schedules duplicate copies of remaining tasks (which are slower) across the nodes that are free currently. When these tasks finish, it is intimated to the JobTracker. If other copies are executing speculatively, Hadoop notifies the TaskTrackers to quit those tasks and reject their output.

Speculative execution is by default true in Hadoop. To disable, set mapred.map.tasks.speculative.execution and mapred.reduce.tasks.speculative.execution  
JobConf options to false.

1. **How to compress mapper output but not the reducer output?**

To achieve this compression, you should set:

conf.set("mapreduce.map.output.compress", true)

conf.set("mapreduce.output.fileoutputformat.compress", false)

1. **What is the difference between Map Side join and Reduce Side Join?**

Map side Join at map side is performed data reaches the map. You need a strict structure for defining map side join. On the other hand, Reduce side Join (Repartitioned Join) is simpler than map side join since the input datasets need not be structured. However, it is less efficient as it will have to go through sort and shuffle phases, coming with network overheads

1. **How can you transfer data from Hive to HDFS?**

By writing the query:

hive> insert overwrite directory '/' select \* from emp;

You can write your query for the data you want to import from Hive to HDFS. The output you receive will be stored in part files in the specified HDFS path.

1. **Does Hadoop replace my existing Data Warehouse?**

Panel says: No. Hadoop can be an extremely valuable extension to your data warehouse and even off-load some services from your data warehouse (such as ETL), but it does not replace it. Hadoop is not a RDBMS, it’s not an ACID compliant database, it’s not even a database. It is a file system (Hadoop Distributed File System or HDFS) and analytic/calculation engine (MapReduce). Yes, we can add SQL services like Hive and other processing engines like Spark but it still doesn’t replace an Enterprise Data Warehouse. Hive and other SQL on Hadoop tools are not full ANSI SQL standard, rather a sub-set of ANSI SQL 1992 features – which would have significant speed/performance implications. Hadoop is complementary to your data warehouse.

Of course, if we really wanted to complicate things, we could dig deeper into what you consider to be a data warehouse -and we would get a variety of answers that run the spectrum. And if the answer was something like “our data warehouse is really just a repository of data from a handful of sources, without any complex schemas or modeling” – then maybe you “could” actually move everything to Hadoop. But since that is fairly academic and probably of limited applicability to most enterprise customers, I’ll stick with my original answer of: no.

1. **What about Spark, does it replace Hadoop?**

Once again: No. Spark is an in-memory processing engine that can run on top of HDFS or stand-alone. As an in-memory engine, Spark is much faster than the traditional MapReduce approach. Spark can process data from HDFS, Hive, Flume and other data sources extremely fast, allowing Hadoop to be an effective streaming or real-time analytics platform. Spark can replace MapReduce as the right tool for many jobs, but it is just one part of the Hadoop ecosystem, which includes tools such as MapReduce, Spark, Storm, Hive, Hbase, Flume etc.

1. **What is a heartbeat in HDFS?**

A heartbeat is a signal indicating that it is alive. A datanode sends heartbeat to Namenode and task tracker will send its heart beat to job tracker. If the Namenode or job tracker does not receive heart beat then they will decide that there is some problem in datanode or task tracker is unable to perform the assigned task.

Are job tracker and task trackers present in separate machines?

Yes, job tracker and task tracker are present in different machines. The reason is job tracker is a single point of failure for the Hadoop MapReduce service. If it goes down, all running jobs are halte

1. **What are the components of Region Server?**

The components of a Region Server are:

* ***WAL***: Write Ahead Log (WAL) is a file attached to every Region Server inside the distributed environment. The WAL stores the new data that hasn’t been persisted or committed to the permanent storage.
* ***Block Cache***: Block Cache resides in the top of Region Server. It stores the frequently read data in the memory.
* ***MemStore***: It is the write cache. It stores all the incoming data before committing it to the disk or permanent memory. There is one MemStore for each column family in a region.
* ***HFile***: HFile is stored in HDFS. It stores the actual cells on the disk.

1. **Explain “WAL” in HBase?**

Write Ahead Log (WAL) is a file attached to every Region Server inside the distributed environment. The WAL stores the new data that hasn’t been persisted or committed to the permanent storage. It is used in case of failure to recover the data sets.

1. **How Spark function interfaces are implemented in Java?**

**// Anonymous way of invocation (Java 7)**

RDD<String> errors = lines. Filter (new Function<String, Boolean>() {

public Boolean call (String x) {

return x.contains ("error");

}

});

**// Named class way of invocation (Java 7) [Constructor can also be defined in this approach]**

class ContainsError implements Function<String, Boolean> () {

public Boolean call (String x) {

return x.contains ("error");

}

}

RDD<String> errors = lines. filter (new ContainsError());

**// Lambda expressions can be used starting from java 8**

1. **What is a Block Manager in Spark?**

Basically, a Block Manager manages the storage for most of the data in spark, name a few: block that represent a cached RDD partition, intermediate shuffle data, broadcast data etc. it is per executor, while in standalone mode, normally, you have one executor per worker.   
  
You don't control how many worker you have at runtime, but you can somehow manage how many executors your application will launch Check different running mode's documentation for details (but control where? Hardly, yarn mode did some works based on data locality, but this is done by framework not user program).

1. **Considerations while designing a data lake?**

https://knowledgent.com/whitepaper/design-successful-data-lake/

1. **Steps behind put operation behind the scenes?**
2. **Steps behind get operation behind the scenes?**
3. **How to set permissions for a Hive table?**
4. **What is Kafka Broker?**

All participating nodes in Kafka Cluster are called Kafka Brokers. They interact with Zookeeper which is also a Broker.

<https://sookocheff.com/post/kafka/kafka-in-a-nutshell/>

1. **What Data modelling would you use when designing a columnar database?**
2. **In Spark 2.0, what is tungsten spark?**

* Tungsten is a new Spark SQL component that provides more efficient Spark operations by working directly at the byte level.
* Tungsten became the default in Spark 1.5 and can be enabled in earlier versions by setting spark.sql.tungsten.enabled to true (or disabled in later versions by setting this to false). Even without Tungsten, Spark SQL uses a columnar storage format with Kryo serialization to minimize storage cost.
* The goal of Project Tungsten is to improve Spark execution by optimizing Spark jobs for CPU and memory efficiency (as opposed to network and disk I/O which are considered fast enough).

1. **Difference between copyFromLocal and put**

hadoop fs -copyFromLocal is similar to hadoop fs -put command. But, for copyFromLocal, the source(s) has to be a local file.  But, there are no such restrictions for put.

1. **Difference between copyToLocal and get**

hadoop fs -copyToLocal is similar to hadoop fs -get command. But, for copyToLocal, the destination(s) has to be a local file.  But, there are no such restrictions for get.

1. **What are the map reduce steps for performing a Join in Hive?**
2. **What are the steps used in deploying a UDF in Hive/Pig?**

**Pig**

1. Java class extends EvalFunc – Write the business logic here and create a JAR
2. Register the JAR

**Hive**

1. Java class extends UDF – Write the business logic here and create a JAR
2. add jar my\_jar.jar;
3. **Kafka compared with Flume?**

|  |  |
| --- | --- |
| **Kafka** | **Flume** |
| It is publish-subscribe messaging system, which offers strong durability, scalability and fault-tolerance support. | Flume is a distributed, reliable, and available system for efficiently collecting, aggregating, and moving large amounts of data from many different sources to a centralized data store, such as HDFS |
| Use Kafka if you need a highly reliable and scalable enterprise messaging system to connect many multiple systems, one of which is Hadoop | Use Flume if you have a non-relational data sources such as log files that you want to stream into Hadoop. |
| High availability of events(recoverable in case of failures) | Flume does not replicate events - in case of flume-agent failure, you will lose events in the channel Flume does not replicate events - in case of flume-agent failure, you will lose events in the channel |

1. **What tools has to be used for Lambda architecture?**

One company attempted to solve a streaming data problem by implementing the Lambda Architecture as follows:

The speed layer enlisted Kafka for ingestion, Storm for processing, Cassandra for state, and Zookeeper for distributed coordination.

The batch layer loaded tuples in batches into S3, then processed the data with Cascading and Amazon Elastic MapReduce.

The serving layer employed a key/value store such as ElephantDB.

Each component required at least three nodes; the speed layer alone needed 12 nodes. For a situation requiring high speed and accuracy, the company implemented a fragile, complex infrastructure.

In-memory databases can be designed to fill the gaps left by the Lambda Architecture. I’ll finish this article by looking at a solution involving in-memory databases, using VoltDB as a model.

1. **Why AVRO format is a better one?**

**Advantage**

1. Inter operable API’s available between different languages
2. Avro compresses the data. Hence occupies less space
3. Notated using JSON (easier)
4. Schema is sent along with the message. Hence it is very useful in compatibility of versions

**Disadvantage**

1. slower serialization
2. To read/write data, need a schema
3. **What are different types of transformations in Spark?**

* Narrow Transformation
* Wide Transformation

1. **Avro Vs Parquet**

Avro is a row-based storage format for Hadoop.

Parquet is a column-based storage format for Hadoop.

If your use case typically scans or retrieves all of the fields in a row in each query, Avro is usually the best choice.

If your dataset has many columns, and your use case typically involves working with a subset of those columns rather than entire records, Parquet is optimized for that kind of work.

1. **Spark Chapter 3 Study**

**Aggregate usage**

val data1 = sc.parallelize(List(1,2,3,4,5,6),2);

Here , 2 partitions are created. Let us assume [1,2,3] belongs to one partition and [4,5,6] belongs to another partition.

**Three computations would be performed here**

1) Tuple(0,0)

2)  ( x.\_1 + y  , x.\_2 + 1)   this is nothing but x.\_1 -> denotes first element of the tuple , x.\_2 denotes second element of tuple ( Here it is 0,0)

**1st partition computation**

      0 + 1 , 0 + 1                       now the tuple becomes (1,1)

      1 + 2 , 1 + 1                       now the tuple becomes (3,2)

      3 + 3 , 2 + 1                       now the tuple becomes  (6,3)

1st partition result is (6,3)

**2nd  partition computation**

(4,5,6)

      0 + 4 , 0 + 1                       now the tuple becomes (4,1)

      4 + 5 , 1 + 1                       now the tuple becomes (9,2)

      9 + 6 , 2 + 1                       now the tuple becomes  (15,3)

2nd  partition result is (15,3)

3)  Summing up multiple partitions here

x.\_1 +y.\_1, x.\_2+y.\_2

From the results above  (6,3) and (15,3)

6+15 , 3+3 = (21,6)

**Code**

scala> val data1=sc.parallelize(List(1,2,3,4,5,6),2)

scala> val out = data1.aggregate((0,0))((x,y)=> (x.\_1 + y , x.\_2 + 1),(x,y)=> (x.\_1 +y.\_1, x.\_2+y.\_2))

**Tuple usage**

scala> val name = ("sunil”, arumugam")

scala> print(name.\_1)

sunil

scala> print(name.\_2)

arumugam

**Parallelize  RDD creation instead of using a file (Test purpose)**

val data2 = sc.parallelize(List("sunil","arumugam"))

**Union**

scala> val temp2 = data2.filter(\_.contains("arumugam"));

temp2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[45] at filter at <console>:28

scala> val temp3 = data2.filter(\_.contains("sunil"));

temp3: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[46] at filter at <console>:28

scala> val temp4 = temp2.**union**(temp3)

temp4: org.apache.spark.rdd.RDD[String] = UnionRDD[47] at union at <console>:32

scala> temp4.collect()

class SearchFunctions(**val query: String**)  Constructor

{

**// Normal Function definition**

def isMatch(s: String): Boolean = {

s.contains(query)

}

def getMatchesFunctionReference(rdd: RDD[String]): RDD[String] = {

**// Problem: "isMatch" means "this.isMatch", so we pass all of "this"**

rdd.map(isMatch)

}

def getMatchesFieldReference(rdd: RDD[String]): RDD[String] = {

**// Problem: "query" means "this.query", so we pass all of "this"**

rdd.map(x => x.split(query))

}

def getMatchesNoReference(rdd: RDD[String]): RDD[String] =

{

**// Safe: extract just the field we need into a local variable**

val query\_ = this.query

rdd.map(x => x.split(query\_))

}

}

scala> val sample = sc.parallelize(List(1,2,3,5))

scala> sample.map(\_.to(5)).collect()

res110: Array[scala.collection.immutable.Range.Inclusive] = Array(Range(1, 2, 3, 4, 5), Range(2, 3, 4, 5), Range(3, 4, 5), Range(5))

scala> val sample = sc.parallelize(List(1,2,3,5))

sample: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[104] at parallelize at <console>:26

scala> val temp = data2.reduce**((x,y) => {println(x+","+y);x+y})**      Multiple lines can be implemented between { }

5,4

9,6

temp: Int = 15

**This result cannot be collected because the output is just a Int. Only RDD can be collect()**

scala> val data1 = sc.parallelize(List("sunil","arumugam"));

data1: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[106] at parallelize at <console>:26

scala> val temp = data1.fold("")((x,y)=>{ println(x+y);x+y });

sunil

sunilarumugam

16/08/17 22:22:44 INFO scheduler.DAGScheduler: ResultStage 89 (fold at <console>:28) finished in 0.013 s

sunilarumugam

16/08/17 22:22:44 INFO scheduler.DAGScheduler: Job 88 finished: fold at <console>:28, took 0.020877 s

temp: String = sunilarumugam

**Ordering elements**

data1.takeOrdered(1)

scala> data1.takeOrdered(1)(Ordering[String].reverse)

res153: Array[String] = Array(sunil)

1. **What is RDD lineage**

Spark does not support data replication in the memory. In the event of any data loss, it is rebuilt using RDD Lineage. It is the process of reconstructing the partitions

1. **Can we do real time processing using Spark Sql**

Not directly but we can register an existing RDD as SQL table and trigger SQL queries.

1. **Can we re-broadcast in Spark**

Yes it can be done.

1. **Difference between RDD, dataset and data frame**

**Must read before interview**

<http://stackoverflow.com/questions/37301226/difference-between-dataset-api-and-dataframe>

**Problems faced in FRESH project**

1. **Versions used in FRESH Project**

Kafka Cluster – 0.9

Apache Spark - 1.6

Cloudera – 5.7

Hbase – 1.2.0

Oracle – 11G

Java – 1.7

**Current version**

2.0

2.4

6.0

5.8

1. **Online interface and REST interface fetch from Kafka is asynchronous because both send and receive were from different topics. How this design issue was resolved?**
   1. By using a HashMap which would maintain a common session ID
   2. This HashMap is common among different Web Service Calls
   3. Two threads are spawned here. One thread would write to Kafka and another thread would pull message from Kafka output topic and assign it against appropriate session id map
2. **Fresh system when reboots, what will happen to message sent in Kafka?**

Kafka Check pointing has to be enabled. By doing this, messages will be consumed when Fresh starts again. (Logic is check pointing maintains offset for each consumer read)

1. **What tools were used to check performance?**

Kafka Jmeter and Kafka performance shell (built in shell) were used to send multiple messages

1. **Reprocessing in FRESH?**

Whenever there is a change in look up data (ex: USD-INR) value this will be refreshed

1. **Lazy evaluation of Kafka output write?**

Kafka connection was created lazily at run time for each of the Partition once. This was used to create Kafka Connection. And for each of this partition kafka handle is used and closed. In similar fashion, it is done in all the remaining executors.

**< This increased Spark performance by 50 times>**

1. **Choosing a JSON instead of CSV?**

JAXON parser and JOLT API’s are easy to parse the data

1. **Not possible scenarios**

* Creating a database connection in driver and broadcasting the handle
* Broadcasting Spark Context
* Inside a Streaming context, using a normal spark context to read from file ( Tried for batch operation and failed)

1. **What we did to enhance performance?**

<http://blog.cloudera.com/blog/2015/03/how-to-tune-your-apache-spark-jobs-part-1/>

<http://blog.cloudera.com/blog/2015/03/how-to-tune-your-apache-spark-jobs-part-2/>

The below parameters were adjusted tuned to enhance the performance in FRESH application

1. Number of min/max executors were adjusted. Also (dynamic was set to = yes).

**Note:** Only upon the load the number of executors will be chosen

1. Number of concurrent jobs is tuned.
2. Tune number of messages per Batch size/Batch duration for Streaming interface
3. Number of cores/memory is adjusted by trial/error method

There were many lines of Java code which were not part of Spark Transformation functions.

1. Moved the normal Java code to Spark transformations.
2. **RDD repartition?**

Partitioning will not be helpful in all applications—for example, if a given RDD is scanned only once, there is no point in partitioning it in advance. It is useful only when a dataset is reused multiple times in key-oriented operations such as joins.

**< In Fresh it is scanned only once. Hence this wasn’t required>**

1. **Configurable design?**
2. **How Kyro Serialization helped FRESH?**

When Spark is transferring data over the network or spilling data to disk, it needs to serialize objects into a binary format. This comes into play during shuffle operations, where potentially large amounts of data are transferred. By default Spark will use Java’s built-in serializer. Spark also supports the use of Kryo, a third-party serialization library that improves on Java’s serialization by offering both faster serialization times and a more compact binary representation, but cannot serialize all types of objects “out of the box.” Almost all applications will benefit from shifting to Kryo for serialization.

To use Kryo serialization, you can set the spark.serializer setting to org.apache.spark.serializer.KryoSerializer.

For best performance, you’ll also want to register classes with Kryo that you plan to serialize, as shown in

Example 8-12. Registering a class allows Kryo to avoid writing full class names with

individual objects, a space savings that can add up over thousands or millions of serialized

records. If you want to force this type of registration, you can set

spark.kryo.registrationRequired to true, and Kryo will throw errors if it encounters

an unregistered class.

156 | Chapter 8: Tuning and Debugging Spark

*Example 8-12. Using the Kryo serializer and registering classes*

**val** conf **= new SparkConf**()

conf.set("spark.serializer", "org.apache.spark.serializer.KryoSerializer")

*// Be strict about class registration*

conf.set("spark.kryo.registrationRequired", "true")

1. **How findbugs tool helped FRESH?**
2. **CITI Spark cluster has Kafka 0.9 libraries whereas Kafka cluster has 0.10 libraries?**
3. kafka cluster  - version is 0.10
4. Spark cluster  - 1.6 which has 0.9 Kafka jars

We are trying to produce () and consume () in spark cluster mode. (via spark-submit)

While running spark-submit job, spark chooses 0.9 version of kafka. The following is our observation

1. Producer – works fine ( 0.9 api and 0.10 api producer is compatible )
2. Streaming Kafka Consumer using KafkaUtils – works fine ( seems here also 0.9 api and 0.10 api producer is compatible)

Consumer using subscribe () API – Errors out with the following message. Can someone help us know why is it failing?

**Solution** :

Since FRESH uses only createDirectStream () API, we did not encounter any issues. (Luckily this works)

1. **Sample execution of spark-submit?**

. /spark-submit --master yarn --deploy-mode cluster --class com.citi.fresh.FreshDriver --files /tmp/log4j.properties#myfile.properties --conf spark.executor.extraJavaOptions='-Dlog4j.configuration=file:myfile.properties' --conf spark.driver.extraJavaOptions='-Dlog4j.configuration=file:myfile.properties' /home/sameeksha/WordCount2.jar

**Web Services**

1. **Core Concept of Web Service?**

Web Service once exposed can be consumed by any platform. It is inter operable.

Ex: Writing a Web service via Java and hosting it. This can be consumed by a .NET platform without any issues.

1. **Different protocols of Web Service?**
   1. SOAP
   2. XML
   3. RESTFUL
   4. Java Web Service
   5. JAX-RS
   6. JAX-WS
2. **What type of architecture is used in Web Services?**

Client- Server Architecture

1. **Quick history on Web Services?**

Word Wide Web Consortium (W3C) came up with a protocol on how to communicate between web services. The method was to use

* 1. SOAP – Simple Object Access Protocol which uses XML
  2. Provides in built WS Security

This XML which is used to communicate between client/server is called as WSDL

WSDL contains all the required elements like

* Method Name
* Method Parameter
* Security parameters etc.

**SOAP Disadvantages**

* xml parsing is tedious and is slow

1. **Differences between REST and SOAP**

|  |  |  |
| --- | --- | --- |
| **No.** | **SOAP** | **REST** |
| 1) | SOAP is a **protocol**. | REST is an **architectural style**. |
| 2) | SOAP stands for **Simple Object Access Protocol**. | REST stands for **REpresentational State Transfer**. |
| 3) | SOAP **can't use REST** because it is a protocol. | REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP. |
| 4) | SOAP **uses services interfaces to expose the business logic**. | REST **uses URI to expose business logic**. |
| 5) | **JAX-WS** is the java API for SOAP web services. | **JAX-RS** is the java API for RESTful web services. |
| 6) | SOAP **defines standards**to be strictly followed. | REST does not define too much standards like SOAP. |
| 7) | SOAP **requires more bandwidth** and resource than REST. | REST **requires less bandwidth** and resource than SOAP. |
| 8) | SOAP **defines its own security**. | RESTful web services **inherits security measures** from the underlying transport. |
| 9) | SOAP **permits XML** data format only. | REST **permits different** data format such as Plain text, HTML, XML, JSON etc. |
| 10) | SOAP is **less preferred** than REST. | REST **more preferred** than SOAP. |

1. **RESTFUL**

* REST stands for REpresentational State Transfer. (This is an architectural style and not a protocol).
* It revolves around resources where every component is a resource and a resource is accessed by a common interface using HTTP standard methods.
* In REST architecture, a REST Server simply provides access to resources and the REST client accesses and presents the resources.
* Here each resource is identified by URIs/ Global IDs. REST uses various representations to represent a resource like Text, JSON and XML. JSON is now the most popular format being used in Web Services.

1. **HTTP Methods of RESTFUL**

The following HTTP methods are most commonly used in a REST based architecture.

* **GET** − Provides a read only access to a resource.
* **PUT** − Used to create a new resource.
* **DELETE** − Used to remove a resource.
* **POST** − Used to update an existing resource or create a new resource.

1. **What is a Resource?**

REST architecture treats every content as a resource. These resources can be Text Files, Html Pages, Images, Videos or Dynamic Business Data. REST Server simply provides access to resources and REST client accesses and modifies the resources. Here each resource is identified by URIs/ Global IDs. REST uses various representations to represent a resource where Text, JSON, XML. The most popular representations of resources are XML and JSON.

1. **Representation of Resources in REST?**

A resource in REST is a similar Object in Object Oriented Programming or is like an Entity in a Database. Once a resource is identified then its representation is to be decided using a standard format so that the server can send the resource in the above said format and client can understand the same format.

1. **Good Resources Representation**

REST does not impose any restriction on the format of a resource representation. A client can ask for JSON representation whereas another client may ask for XML representation of the same resource to the server and so on. It is the responsibility of the REST server to pass the client the resource in the format that the client understands.

Following are some important points to be considered while designing a representation format of a resource in RESTful Web Services.

**Understandability** − Both the Server and the Client should be able to understand and utilize the representation format of the resource.

**Completeness** − Format should be able to represent a resource completely. For example, a resource can contain another resource. Format should be able to represent simple as well as complex structures of resources.

**Linkablity** − A resource can have a linkage to another resource, a format should be able to handle such situations.

However, at present most of the web services are representing resources using either XML or JSON format. There are plenty of libraries and tools available to understand, parse, and modify XML and JSON data.