**Q) Write Kafka Producer?**

A) Before going to write Producer program can you give me minute to explain what is Kafka.

Kafka is open source stream processing platform.

**Kafka Includes Core API’s** - Producer API, Consumer API, Streams API, Connect API

**Use cases** – Messaging, Website Activity Tracking, Metrics, Log Aggregation, Stream Processing, Event Sourcing, Commit Log (Ref link - https://kafka.apache.org/documentation/ )

Now we talk about Producer API:

This is mainly used for Publishing and Consuming Messages Using a Java Client.

Apache Kafka Producer API having “**KafkaProducer**” Class, This class provides to connect Kafka broker in its constructor with the following methods.

Producer class having Send method, Flush method, Metrics .

**producer.send(new ProducerRecord<byte[],byte[]>(topic, partition, key1, value1) , callback);**

In the above method

**ProducerRecord** − The producer manages a buffer of records waiting to be sent.

**Callback** − A user-supplied callback to execute when the record has been acknowledged by the server (null indicates no callback).

Flush Method – for sent messanges // public void flush()

Metrics - provides partitionFor for getting the Partition meta data for given topic. Used for custom partitioning. // public Map metrics()

public void close() – this method close method blocks until previously sent requests are completed.

**Overview of API :**

There are 2 types of producers i.e., **Synchronous(Sync) and Asynchronous (Async)**

**Sync -** Send message directly, but message in background.

**Async -** upto 0.8 there is no send() and call back methods for register error handlers. Available only 0.9 onwards

The Producer API that wraps the 2 low-level producers - kafka.producer.SyncProducer and kafka.producer.async.AsyncProducer.

class Producer {

/\* Sends the data, partitioned by key to the topic using either the \*/

/\* synchronous or the asynchronous producer \*/

public void send(kafka.javaapi.producer.ProducerData<K,V> producerData);

/\* Sends a list of data, partitioned by key to the topic using either \*/

/\* the synchronous or the asynchronous producer \*/

public void send(java.util.List<kafka.javaapi.producer.ProducerData<K,V>> producerData);

/\* Closes the producer and cleans up \*/

public void close();

}

**Q) Monad class ?**

A) Monad class means wrapping of objects. (identity with Unit, Bind with Map)

Monads provide us with the following two operations (if you read about monads somewhere and see the term “two natural transformations”, this is what they are talking about):

**identity** (return in Haskell, unit in Scala)

**bind** (>>= in Haskell, flatMap in Scala)

Scala doesn’t come with a built-in monad type like Haskell so we will model the monad ourselves. If you take a look at some cool functional programming libraries like Scalaz you will find monads there, along with the rest of the category theory family (functors, applicatives, monoids and so on), but in plain Scala there’s no such thing out of the box.

We will model a monad with a generic trait that provides methods unit() and flatMap(). We can call it M instead of Monad simply to be more concise. Here it is:

trait M[A] {

def flatMap[B](f: A => M[B]): M[B]

}

def unit[A](x: A): M[A]

Referred link - https://www.haskell.org/tutorial/monads.html

**3) How much Flume-NG data is reliable?**

Apache Flume is a distributed, reliable, and available system for efficiently collecting, aggregating and moving large amounts of log data from many different sources to a centralized data store.

The work necessary to make this change began a few months ago under the JIRA issue FLUME-728. This work currently resides on a separate branch by the name flume-728, and is informally referred to as Flume NG. At the time of writing this post Flume NG had gone through two internal milestones – NG Alpha 1, and NG Alpha 2 and a formal incubator release of Flume NG is in the works.

**Core Concept – Event, Flow, Client, Agent, Source, Channel, Sink**

The architecture of Flume NG is based on a few concepts that together help achieve this objective. Some of these concepts have existed in the past implementation, but have changed drastically. Here is a summary of concepts that Flume NG introduces, redefines, or reuses from earlier implementation

Ref link - http://blog.cloudera.com/blog/2011/12/apache-flume-architecture-of-flume-ng-2/

**4) what is Interceptor?**

Flume Plugin listen in any Incoming and Alter events content on the Fly.

Interceptor Implementation – for JSon, Project dependencies.

Ref Link - https://hadoopi.wordpress.com/2014/06/11/flume-getting-started-with-interceptors/

**5) Flume-ng Channel types?**

Memory Channel, JDBC Channel, Kafka Channel, File Channel, Spillable Memory Channel,

Pseudo Transaction Channel

These are the Flume-NG channel types.

In Flume channel types only – memory, jdbc, file and Kafka

Ref Links - https://www.tutorialspoint.com/apache\_flume/apache\_flume\_configuration.htm

https://www.cloudera.com/documentation/enterprise/5-5-x/topics/cdh\_ig\_flume\_supported\_sources\_sinks\_channels.html

**6) Base class in java?**

A base class is a class, in an object-oriented programming language, from which other classes are derived. It facilitates the creation of other classes that can reuse the code implicitly inherited from the base class (except constructors and destructors).

Ex. Inheritance Base class

**7) Base class in scala?**

Base class concept same both java and scala is Same. But the thing is syntactically different. In Scala Base and Derived Keywords available.

Ex. abstract class Base( val x : String )

final class Derived( x : String ) extends Base( "Base's " + x )

{

override def toString = x

}

Ref link - https://stackoverflow.com/questions/6515931/idiomatic-scala-way-to-deal-with-base-vs-derived-class-field-names

**8) what is RDD?**

Resilient Distributed Dataset is the primary data abstraction in Apache Spark and the core of Spark.

The features of RDDs (decomposing the name):

Resilient, i.e. fault-tolerant with the help of RDD lineage graph and so able to recompute missing or damaged partitions due to node failures.

Distributed with data residing on multiple nodes in a cluster.

Dataset is a collection of partitioned data with primitive values or values of values, e.g. tuples or other objects (that represent records of the data you work with).

Ref Link - https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd.html

**9)what is Fault tolarence?**

It is used for computing large data sets with parallel and distributed algorithms in the cluster. MapReduce is the heart of Hadoop. MATERIALS AND METHODS. FAULT TOLERANCE. Fault tolerance is defined as, when the system functions properly without any data loss even if some hardware components of the system has failed.

**10) Immutable data?**

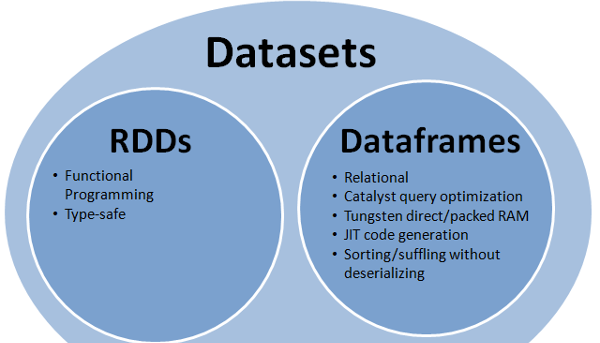
Immutable Data means we can’t able to change.

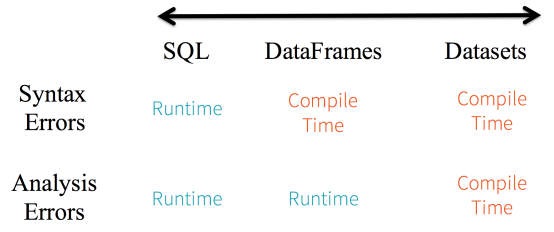
11) **dataframes vs datasets?**

Dataframe - Spark 1.3 introduced a new DataFrame API as part of the Project Tungsten initiative which seeks to improve the performance and scalability of Spark. The DataFrame API introduces the concept of a schema to describe the data, allowing Spark to manage the schema and only pass data between nodes, in a much more efficient way than using Java serialization.

Dataset - The Dataset API, released as an API preview in Spark 1.6, aims to provide the best of both worlds; the familiar object-oriented programming style and compile-time type-safety of the RDD API but with the performance benefits of the Catalyst query optimizer. Datasets also use the same efficient off-heap storage mechanism as the DataFrame API.

The below image give more clarity.





Ref link - <http://www.agildata.com/apache-spark-rdd-vs-dataframe-vs-dataset/>

https://www.linkedin.com/pulse/apache-spark-rdd-vs-dataframe-dataset-chandan-prakash

**12) Types of faults in RDD?**

RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.

Ref link - https://www.tutorialspoint.com/apache\_spark/apache\_spark\_rdd.htm

**13) No of executors in your project?**

Sample - 3

**14) What trade in scala?**

Big Data technology offers tremendous potential for all areas of financial services, including the ability to help solve business challenges and analyze market data on the trading side of the house.

In trading we analyse the data we will give most probability which share is grown up based on the historical data.

**15) map vs flatemap?**

As per the definition, difference between map and flatMap is:

map: It returns a new RDD by applying given function to each element of the RDD. Function in map returns only one item.

flatMap: Similar to map, it returns a new RDD by applying a function to each element of the RDD, but output is flattened.

Ref Link - https://stackoverflow.com/questions/22350722/can-someone-explain-to-me-the-difference-between-map-and-flatmap-and-what-is-a-g

**16) pyspark?**

PySpark is the python binding for the Spark Platform and API and not much different from the Java/Scala versions

**17) Dstream?**

A Discretized Stream (DStream), the basic abstraction in Spark Streaming, is a continuous sequence of RDDs (of the same type) representing a continuous stream of data (see spark.RDD for more details on RDDs). DStreams can either be created from live data (such as, data from HDFS, Kafka or Flume) or it can be generated by transformation existing DStreams using operations such as map, window and reduceByKeyAndWindow. While a Spark Streaming program is running, each DStream periodically generates a RDD, either from live data or by transforming the RDD generated by a parent DStream.

Ref link - https://spark.apache.org/docs/0.7.2/api/streaming/spark/streaming/DStream.html

**18) Zero vs Null vs None in scala?**

http://alvinalexander.com/scala/scala-null-values-option-uninitialized-variables

**19) Transformations?---**

transformations are functions that take a RDD as the input and produce one or many RDDs as the output.

Ref Link - https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd-transformations.html

**20) Actions?---**

Unlike Transformations which produce RDDs, action functions produce a value back to the Spark driver program. Actions may trigger a previously constructed, lazy RDD to be evaluated.

Ref Link - https://www.supergloo.com/fieldnotes/apache-spark-examples-of-actions/

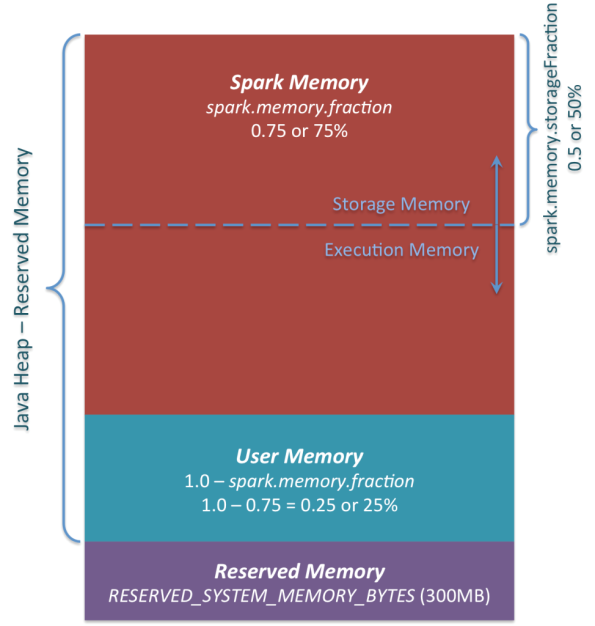
**21) Yarn mode?**

There are two deploy modes in spark ie., Client Mode and Cluster Mode.

Ref Link - https://spark.apache.org/docs/latest/running-on-yarn.html

**22) execute Momery?**

Peak Execution memory refers to the memory used by internal data structures created during shuffles, aggregations and joins. The value of this accumulator should be approximately the sum of the peak sizes across all such data structures created in this task. For SQL jobs, this only tracks all unsafe operators, broadcast joins, and external sort.



Ref Link - <https://stackoverflow.com/questions/39503484/peak-execution-memory-in-spark>,

https://0x0fff.com/spark-memory-management/

**23) Driver memory?**

In a Spark Application, Driver is responsible for task scheduling and Executor is responsible for executing the concrete tasks in your job.

Ref Link - https://stackoverflow.com/questions/27181737/how-to-deal-with-executor-memory-and-driver-memory-in-spark

**24) what type of data losses in RDD?**

Spark does not support data replication in the memory. In the event of any data loss, it is rebuilt using the “RDD Lineage”. It is a process that reconstructs lost data partitions

**25) Flume-ng ?**

Apache Flume is a distributed, reliable, and available system for efficiently collecting, aggregating and moving large amounts of log data from many different sources to a centralized data store.

Source and Sink, in between the process of channel how to contacting the data to HDFS.

Flume-Ng is Advanced of Flume only.

Ref Link - https://blogs.apache.org/flume/entry/flume\_ng\_architecture

**26) Apache Kafka?**

Refer 1 ans.

**27) Concurrency and Parallelism in scala?**

Concurrency and parallelism are related concepts. Concurrency is two or more tasks progressing with only one task executing at any given point in time on a single processing unit. Parallelism is two or more tasks executing simultaneously on two different processing units.

Ref Link - https://www.packtpub.com/mapt/book/application\_development/9781783984343/4/ch04lvl1sec35/Concurrency+and+parallelism

**28) How does HashPartitioner work?**

Hash partitioning maps data to partitions based on a hashing algorithm that Oracle applies to the partitioning key that you identify. The hashing algorithm evenly distributes rows among partitions, giving partitions approximately the same size.

Hash partitioning is the ideal method for distributing data evenly across devices. Hash partitioning is also an easy-to-use alternative to range partitioning, especially when the data to be partitioned is not historical or has no obvious partitioning key.Note: You cannot change the hashing algorithms used by partitioning.

Ref Link - https://docs.oracle.com/cd/B28359\_01/server.111/b32024/partition.htm

**29) RDD(string,value).leftOuterJoint(RDD(string, boolean)?**

**30) i want unique values from two array list of RDD?**

val result = rdd.flatMap(\_.\_2).distinct //if you want the result in an RDD,

or

val result = rdd.flatMap(\_.\_2).distinct.collect //if you want the result in a local collection.

**31) create new column with function in spark dataframe?**

type(randomed\_hours) # => list

# Create in Python and transform to RDD

new\_col = pd.DataFrame(randomed\_hours, columns=['new\_col'])

spark\_new\_col = sqlContext.createDataFrame(new\_col)

my\_df\_spark.withColumn("hours", spark\_new\_col["new\_col"])

Ref Link - https://stackoverflow.com/questions/33681487/how-do-i-add-a-new-column-to-a-spark-dataframe-using-pyspark

**32) how to read multiple text files into a single RDD?**

Use union as follows:

val sc = new SparkContext(...)

val r1 = sc.textFile("xxx1")

val r2 = sc.textFile("xxx2")

...

val rdds = Seq(r1, r2, ...)

val bigRdd = sc.union(rdds)

**33) Repartition vs Coalesce?**

coalesce uses existing partitions to minimize the amount of data that's shuffled. repartition creates new partitions and does a full shuffle.

Ref Link - https://stackoverflow.com/questions/31610971/spark-repartition-vs-coalesce

**http://www.bigdatatrunk.com/top-spark-rdd-interview-questions/**