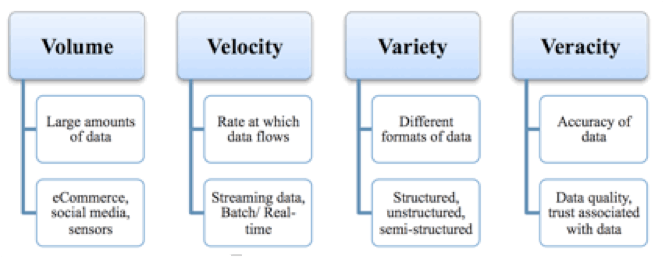
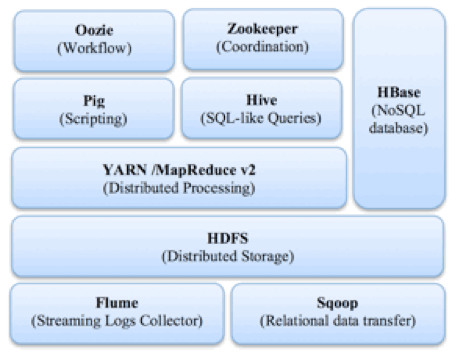
**Q. What is big data?**  
**Big Data** is really large amount of data that exceeds the processing capacity of conventional database systems, and requires special parallel processing mechanism. The data is too big and grows rapidly. This data can be either structural or unstructured data. To retrieve meaningful information from this data, we must choose an alternative way to process it.  
***Characteristics of Big Data:***  
Data that has very large volume, comes from variety of sources and formats and flows into an organization with a great velocity is normally referred to as Big Data.



**Q. What is Hadoop?**  
Hadoop is a framework that allows distributed processing of large data sets across clusters of computers using simple and fault tolerant programming model. It is designed to scale up from a very few to thousands of machines, each machine provides local computation and storage. The Hadoop software library itself is designed to detect and handle failures at the application layer.  
**Hadoop** is written in java by Apache Software Foundation.  It process data very reliably and fault-tolerant manner.  
**Core components of Hadoop:**  
HDFS (Storage) + MapReduce/YARN (Processing)



**Q. What are the sources generating big data?**  
Employers,Users and Machines  
**Employees:** Historically, employees of organizations generated data.  
**Users:**Then a shift occurred where users started generating data. For example, email, social media, photos, videos, audio and e-Commerce.  
**Machines:**Smart phones, intelligent kitchen appliances, CCTV cameras, smart meters, global satellites, and traffic flow sensors.

**Q. Why do we need a new framework for handling big data?**  
Most of the traditional data was organized neatly in relational databases. Data sets now are so large and complex that they are beyond the capabilities of traditional storage and processing systems.  
The following challenges demand cost-effective and innovative forms of handling big data at scale:  
**Lots of data**  
Organizations are increasingly required to store more and more data to survive in today’s highly competitive environment. The sheer volume of the data demands lower storage costs as compared to the expensive commercial relational database options.  
**Complex nature of data**  
Relational data model has great properties for structured data but many modern systems don’t fit well in row-column format. Data is now generated by diverse sources in various formats like multimedia, images, text, real-time feeds, and sensor streams. Usually for storage, the data is transformed, aggregated to fit into the structured format resulting in the loss of the original raw data.  
**New analysis techniques**  
Previously simple analysis (like average, sum) would prove to be sufficient to predict customer behavior. But now complex analysis needs to be performed to gain insightful understanding of data collected. For example, prediction models for effective micro-segmentation needs to analyse the customer’s purchase history, browsing behavior, likes and reviews on social media website to perform micro-segmentation. These advanced analytic techniques need the framework to run on.  
**Hadoop to rescue:** Framework that provides low-cost storage and complex analytic processing capabilities

**Q. Why do we need Hadoop framework, shouldn’t DFS be able to handle large volumes of data already?**  
Yes, it is true that when the datasets cannot fit in a single physical machine, then Distributed File System (DFS) partitions the data, store and manages the data across different machines. But, DFS lacks the following features for which we need Hadoop framework:  
**Fault tolerant:**  
When a lot of machines are involved chances of data loss increases. So, automatic fault tolerance and failure recovery becomes a prime concern.  
**Move data to computation:**  
If huge amounts of data are moved from storage to the computation machines then the speed depends on network bandwidth.

**Q. What is the difference between traditional RDBMS and Hadoop?**

|  |  |
| --- | --- |
| **RDBMS** | **Hadoop** |
| Schema on write | Schema on read |
| Scale up approach | Scale out approach |
| Relational tables | Key-value format |
| Structured queries | Function programming |
| Online Transactions | Batch processing |

**Q. What is HDFS?**  
Hadoop Distributed File Systems (HDFS) is one of the core components of Hadoop framework. It is a distributed file system for Hadoop. It runs on top of existing file system (ext2, ext3, etc.)  
**Goals:** Automatic recovery from failures, Move Computation than data.  
**HDFS features:**  
1. Supports storage of very large datasets  
2. Write once read many access model  
3. Streaming data access  
4. Replication using commodity hardware

**Q. What is difference between regular file system and HDFS?**

|  |  |
| --- | --- |
| **Regular File Systems** | **HDFS** |
| Small block size of data (like 512 bytes) | Large block size (orders of 64mb) |
| Multiple disk seeks for large files | Reads data sequentially after single seek |

**Q. What HDFS is not meant for?**  
HDFS is not good at:  
1. Applications that requires low latency access to data (in terms of milliseconds)  
2. Lot of small files  
3. Multiple writers and file modifications

**Q. What is HDFS block size and what did you chose in your project?**  
By default, the HDFS block size is 64MB. It can be set to higher values as 128MB or 256MB. 128MB is acceptable industry standard.

**Q. What is the default replication factor?**  
Default replication factor is 3

**Q. What are different hdfs dfs shell commands to perform copy operation?**  
$ hadoop fs -copyToLocal  
$ hadoop fs -copyFromLocal  
$ hadoop fs -put

**Q. What are the problems with Hadoop 1.0?**  
1. NameNode: No Horizontal Scalability and No High Availability  
2. Job Tracker: Overburdened.  
3. MRv1: It can only understand Map and Reduce tasks

**Q. What comes in Hadoop 2.0 and MapReduce V2 (YARN)?**  
NameNode: HA and Federation  
JobTracker: Cluster and application resource

**Q. What different type of schedulers and type of scheduler did you use?**  
**Capacity Scheduler**  
It is designed to run Hadoop applications as a shared, multi-tenant cluster while maximizing the throughput and the utilization of the cluster.  
**Fair Scheduler**  
Fair scheduling is a method of assigning resources to applications such that all apps get, on average, an equal share of resources over time.

**Q. Steps involved in decommissioning (removing) the nodes in the Hadoop cluster?**  
1. Update the network addresses in the dfs.exclude and mapred.exclude  
2. $ hadoop dfsadmin -refreshNodes  and hadoop mradmin -refreshNodes  
3. Check Web UI it will show “Decommissioning in Progress”  
4. Remove the Nodes from include file and then run again the step 2 refreshNodes.  
5. Remove the Nodes from slave file.

**Q. Steps involved in commissioning (adding) the nodes in the Hadoop cluster?**  
1. Update the network addresses in the dfs.include and mapred.include  
2. $ hadoop dfsadmin -refreshNodes  and hadoop mradmin -refreshNodes  
3. Update the slave file.  
4. Start the DataNode and NodeManager on the added Node.

**Q. How to keep HDFS cluster balanced?**  
Balancer is a tool that tries to provide a balance to a certain threshold among data nodes by copying block data distribution across the cluster.

**Q. What is distcp?**  
1. distcp is the program comes with Hadoop for copying large amount of data to and from Hadoop file systems in parallel.  
2. It is implemented as MapReduce job where copying is done through maps that run in parallel across the cluster.  
3. There are no reducers.

**Q. What are the daemons of HDFS?**  
1. NameNode  
2. DataNode  
3. Secondary NameNode.

**Q. Command to format the NameNode?**  
$ hdfs namenode -format

**Q. What are the functions of NameNode?**  
The NameNode is mainly responsible for:  
**Namespace**  
Maintain metadata about the data  
**Block Management**  
Processes block reports and maintain location of blocks.  
Supports block related operations  
Manages replica placement

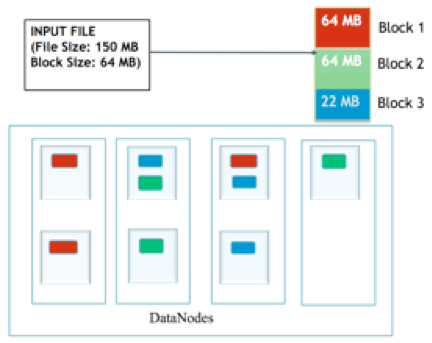
**Q. What is HDFS Federation?**  
1. HDFS federation allows scaling the name service horizontally; it uses multiple independent NameNodes for different namespaces.  
2. All the NameNodes use the DataNodes as common storage for blocks.  
3. Each DataNode registers with all the NameNodes in the cluster.  
4. DataNodes send periodic heartbeats and block reports and handles commands from the NameNodes

**Q. What is HDFS High Availability?**  
1. In HDFS High Availability (HA) cluster; two separate machines are configured as NameNodes.  
2. But one of the NameNodes is in an **Active** state; other is in a **Standby** state.  
3. The Active NameNode is responsible for all client operations in the cluster, while the Standby is simply acting as a slave, maintaining enough state to provide a fast failover if necessary  
4. They shared the same storage and all DataNodes connects to both the NameNodes.

**Q. How client application interacts with the NameNode?**  
1. Client applications interact using Hadoop HDFS API with the NameNode when it has to locate/add/copy/move/delete a file.  
2. The NameNode responds the successful requests by returning a list of relevant DataNode servers where the data is residing.  
3. Client can talk directly to a DataNode after the NameNode has given the location of the data

**Q. What is a DataNode?**  
1. A DataNode stores data in the Hadoop File System HDFS is a slave node.  
2. On startup, a DataNode connects to the NameNode.  
3. DataNode instances can talk to each other mostly during replication.

**Q. What is rack-aware replica placement policy?**  
1. Rack-awareness is used to take a node’s physical location into account while scheduling tasks and allocating storage.  
2. Default replication factor is 3 for a data blocks on HDFS.  
3. The first two copies are stored on DataNodes located on the same rack while the third copy is stored on a different rack.



**Q. What is the main purpose of HDFS fsck command?**  
fsck a utility to check health of the file system, to find missing files, over-replicated, under-replicated and corrupted blocks.  
Command for finding the blocks for a file:  
**$ hadoop fsck -files -blocks –racks**

**Q. What is the purpose of DataNode block scanner?**  
1. Block scanner runs on every DataNode, which periodically verifies all the blocks stored on the DataNode.  
2. If bad blocks are detected it will be fixed before any client reads.

**Q. What is the purpose of dfsadmin tool?**  
1. It is used to find information about the state of HDFS  
2. It performs administrative tasks on HDFS  
3. Invoked by hadoop dfsadmin command as superuser

**Q. What is the command for printing the topology?**  
It displays a tree of racks and DataNodes attached to the tracks as viewed by the .hdfs dfsadmin -printTopology

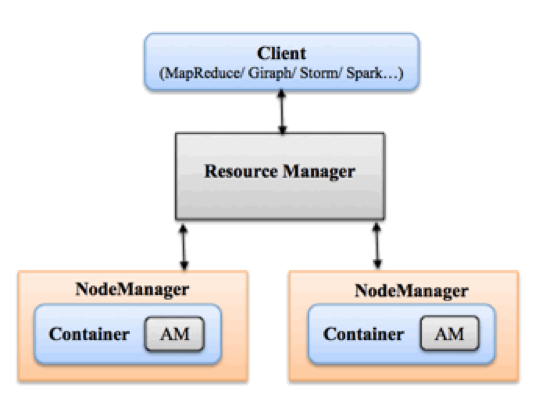
**Q. What is RAID?**  
RAID is a way of combining multiple disk drives into a single entity to improve performance and/or reliability. There are a variety of different levels in RAID  
For example, In RAID level 1 copy of the same data on two disks increases the read performance by reading alternately from each disk in the mirror.

**Q. Does Hadoop requires RAID?**  
1. In DataNodes storage is not using RAID as redundancy can be achieved by replication between the Nodes.  
2. In NameNode’s disk RAID is recommended.

**Q. What are the site-specific configuration files in Hadoop?**  
1. conf/core-site.xml  
2. conf/hdfs-site.xml  
3. conf/yarn-site.xml  
4. conf/mapred-site.xml.  
5. conf/hadoop-env.sh  
6. conf/yarn-env.sh

**Q. What is MapReduce?**  
MapReduce is a programming model for processing on the distributed datasets on the clusters of computer.  
**MapReduce Features:**  
1. Distributed programming complexity is hidden  
2. Built in fault-tolerance  
3. Programming model is language independent  
4. Parallelization and distribution are automatic  
5. Enable data local processing

**Q. What is the fundamental idea behind YARN?**  
In YARN (Yet Another Resource Allocator), JobTracker responsibility is split into:  
1. Resource management  
2. Job scheduling/monitoring having separate daemons.  
Yarn supports additional processing models and implements a more flexible execution engine.



**Q. What MapReduce framework consists of?**  
**ResourceManager (RM)**  
1. Global resource scheduler  
2. One master RM  
**NodeManager (NM)**  
1. One slave NM per cluster-node.  
**Container**  
1. RM creates Containers upon request by AM  
2. Application runs in one or more containers  
**ApplicationMaster (AM)**  
1. One AM per application  
2. Runs in Container

**Q. What are different daemons in YARN?**  
1. ResourceManager: Global resource manager.  
2. NodeManager: One per data node, It manages and monitors usage of the container (resources in terms of Memory, CPU).  
3. ApplicationMaster: One per application, Tasks are started by NodeManager

**Q. What are the two main components of ResourceManager?**  
**Scheduler**  
It allocates the resources (containers) to various running applications: Container elements such as memory, CPU, disk etc.  
**ApplicationManager**  
It accepts job-submissions, negotiating for container for executing the application specific ApplicationMaster and provides the service for restarting the ApplicationMaster container on failure.

**Q. What is the function of NodeManager?**  
The NodeManager is the resource manager for the node (Per machine) and is responsible for containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the ResourceManager

**Q. What is the function of ApplicationMaster?**  
ApplicationMaster is per application and it has the responsibility of negotiating appropriate resource containers from the Scheduler, tracking their status and monitoring for progress.

**Q. What are the minimum configuration requirements for a MapReduce application?**  
The job configuration requires the  
1. input location  
2. output location  
3. map() function  
4. reduce() functions and  
5. job parameters.

**Q. What are the steps to submit a Hadoop job?**  
Steps involved in Hadoop job submission:  
1. Hadoop job client submits the job jar/executable and configuration to the ResourceManager.  
2. ResourceManager then distributes the software/configuration to the slaves.  
3. ResourceManager then scheduling tasks and monitoring them.  
4. Finally, job status and diagnostic information is provided to the client.

**Q. How does MapReduce framework view its input internally?**  
It views the input as a set of pairs and produces a set of pairs as the output of the job.

**Q. Assuming default configurations, how is a file of the size 1 GB (uncompressed) stored in HDFS?**  
Default block size is 64MB. So, file of 1GB will be stored as 16 blocks. MapReduce job will create 16 input splits; each will be processed with separate map task i.e. 16 mappers.

**Q. What are Hadoop Writables?**  
Hadoop Writables allows Hadoop to read and write the data in a serialized form for transmission as compact binary files. This helps in straightforward random access and higher performance. Hadoop provides in built classes, which implement Writable: Text, IntWritable, LongWritable, FloatWritable, and BooleanWritable.

**Q. Why comparison of types is important for MapReduce?**  
Comparison is important as in the sorting phase the keys are compared with one another. For comparison the WritableComparable interface is implemented.

**Q. What is the purpose of RawComparator interface?**  
RawComparator allows the implementors to compare records read from a stream without deserialization them into objects, so it will be optimized, as there is not overhead of object creation.

**Q. What is a NullWritable?**  
It is a special type of Writable that has zero-length serialization. In MapReduce, a key or a value can be declared as NullWritable if we don’t need that position, storing constant empty value.

**Q. What is Avro Serialization System?**  
Avro is a language-neutral data serialization system. It has data formats that work with different languages. Avro data is described using a language independent schema (usually written in JSON). Avro data files support compression and are **splittable**.  
Avro provides AvroMapper and AvroReducer to run MapReduce programs.

**Q. Explain use cases where SequenceFile class can be a good fit?**  
When the data is of type binary then SequenceFile will provide a persistent structure for binary key-value pairs. SequenceFiles also work well as containers for smaller files as HDFS and MapReduce are optimized for large files.

**Q. What is MapFile?**  
A MapFile is an indexed SequenceFile and it is used for look-ups by key.

**Q. What is the core of the job in MapReduce framework?**  
Core of a job:  
*Mapper interface*: map method  
*Reducer interface* reduce method

**Q. What are the steps involved in MapReduce framework?**  
1. Firstly, the mapper input key/value pairs maps to a set of intermediate key/value pairs.  
2. Maps are the individual tasks that transform input records into intermediate records.  
3. The transformed intermediate records do not need to be of the same type as the input records.  
4. A given input pair maps to zero or many output pairs.  
5. The Hadoop MapReduce framework creates one map task for each InputSplit generated by the InputFormat for the job.  
6. It then calls map(WritableComparable, Writable, Context) for each key/value pair in the InputSplit for that task.  
7. All intermediate values associated with a given output key are grouped passed to the Reducers.

**Q. Where is the Mapper Output stored?**  
The mapper output is stored on the Local file system of each individual mapper nodes. The intermediate data is cleaned up after the Hadoop Job completes.

**Q. What is a partitioner and how the user can control which key will go to which reducer?**  
Partitioner controls the partitioning of the keys of the intermediate map-outputs by the default. The key to decide the partition uses hash function. Default partitioner is HashPartitioner.  
A custom partitioner is implemented to control, which keys go to which Reducer.  
public class SamplePartitioner extends Partitioner {  
@Override  
public int getPartition(Text key, Text value, int numReduceTasks) {  
}  
}

**Q. What are combiners and its purpose?**  
1. Combiners are used to increase the efficiency of a MapReduce program. It can be used to aggregate intermediate map output locally on individual mapper outputs.  
2. Combiners can help reduce the amount of data that needs to be transferred across to the reducers.  
3. Reducer code as a combiner if the operation performed is commutative and associative.  
4. Hadoop may or may not execute a combiner.

**Q. How number of partitioners and reducers are related?**  
The total numbers of partitions are the same as the number of reduce tasks for the job.

**Q. What is IdentityMapper?**  
IdentityMapper implements the mapping inputs directly to output. IdentityMapper.class is used as a default value when JobConf.setMapperClass is not set.

**Q. What is IdentityReducer?**  
In IdentityReducer no reduction is performed, writing all input values directly to the output. IdentityReducer.class is used as a default value when JobConf.setReducerClass is not set

**Q. What is the reducer and its phases?**  
Reducer reduces a set of intermediate values, which has same key to a smaller set of values. The framework then calls reduce().  
**Syntax:**  
reduce(WritableComparable, Iterable, Context) method for each pair in the grouped inputs.  
Reducer has three primary phases:  
1. Shuffle  
2. Sort  
3. Reduce

**Q. How to set the number of reducers?**  
The number of reduces for the user sets the job:  
1. Job.setNumReduceTasks(int)  
2. -D mapreduce.job.reduces

**Q. Detail description of the Reducer phases?**  
**Shuffle:**  
Sorted output (Mapper) à Input (Reducer). Framework then fetches the relevant partition of the output of all the mappers.  
**Sort:**  
The framework groups Reducer inputs by keys. The shuffle and sort phases occur simultaneously; while map-outputs are being fetched they are merged.  
**Secondary Sort:**  
Grouping the intermediate keys are required to be different from those for grouping keys before reduction, then Job.setSortComparatorClass(Class).  
**Reduce:**  
reduce(WritableComparable, Iterable, Context) method is called for each pair in the grouped inputs.  
The output of the reduce task is typically written using Context.write(WritableComparable, Writable).

**Q. Can there be no Reducer?**  
Yes, the number of reducer can be zero if no reduction of values is required.

**Q. What can be optimum value for Reducer?**  
Value of Reducers can be: 0.95  
1.75 multiplied by ( \* < number of maximum containers per node>)  
Increasing number of reducers  
1. Increases the framework overhead  
2. Increases load balancing  
3. Lowers the cost of failures

**Q. What is a Counter and its purpose?**  
Counter is a facility for MapReduce applications to report its statistics. They can be used to track job progress in a very easy and flexible manner. It is defined by MapReduce framework or by applications. Each Counter can be of any Enum type. Applications can define counters of type Enum and update them via counters.incrCounter in the map and/or reduce methods.

**Q. Define different types of Counters?**  
**Built in Counters:**  
Map Reduce Task Counters  
Job Counters  
**Custom Java Counters:**  
MapReduce allows users to specify their own counters (using Java enums) for performing their own counting operation.

**Q. Why Counter values are shared by all map and reduce tasks across the MapReduce framework?**  
Counters are global so shared across the MapReduce framework and aggregated at the end of the job across all the tasks.

**Q. Explain speculative execution.**  
1. Speculative execution is a way of dealing with individual machine’s performance. As there are lots of machines in the cluster, some machines can have low performance, which affects the performance of the whole job.  
2. Speculative execution in Hadoop can run multiple copies of the same map or reduce task on different task tracker nodes and the results from first node to finish are used.

**Q. What is DistributedCache and its purpose?**  
DistributedCache is a facility provided by the MapReduce framework to cache files (text, archives, jars etc.) needed by applications. It distributes application-specific, large, read-only files efficiently. The user needs to use DistributedCache to distribute and symlink the script file.

**Q. What is the Job interface in MapReduce framework?**  
Job is the primary interface for a user to describe a MapReduce job to the Hadoop framework for execution. Some basic parameters are configured for example:  
1. Job.setNumReduceTasks(int)  
2. Configuration.set(JobContext.NUM\_MAPS, int)  
3. Mapper  
4. Combiner (if any)  
5. Partitioner  
6. Reducer  
7. InputFormat  
8. OutputFormat implementations  
9. setMapSpeculativeExecution(boolean))/ setReduceSpeculativeExecution(boolean))  
10. Maximum number of attempts per task (setMaxMapAttempts(int)/ setMaxReduceAttempts(int)) etc.  
11. DistributedCache for large amounts of (read-only) data.

**Q. What is the default value of map and reduce max attempts?**  
Framework will try to execute a map task or reduce task by **default 4 times** before giving up on it.

**Q. Explain InputFormat?**  
InputFormat describes the input-specification for a MapReduce job. The MapReduce framework depends on the InputFormat of the job to:  
Checks the input-specification of the job.  
It then splits the input file(s) into logical InputSplit instances, each of which is then assigned to an individual Mapper.  
To extract input records from the logical InputSplit for processing by the Mapper it provides the RecordReader implementation.  
Default: **TextInputFormat**

**Q. What is InputSplit and RecordReader?**  
InputSplit specify the data to be processed by an individual Mapper.  
In general, InputSplit presents a byte-oriented view of the input.  
Default: **FileSplit**  
RecordReader reads pairs from an InputSplit, then processes them and presents record-oriented view

**Q. Explain the Job OutputFormat?**  
OutputFormat describes details of the output for a MapReduce job.  
The MapReduce framework depends on the OutputFormat of the job to:  
It checks the job output-specification  
To write the output files of the job in the **pairs**, it provides the RecordWriter implementation.  
Default: **TextOutputFormat**

**Q. How is the option in Hadoop to skip the bad records?**  
Hadoop provides an option where a certain set of bad input records can be skipped when processing map inputs. This feature can be controlled through the **SkipBadRecords** class.

**Q. Different ways of debugging a job in MapReduce?**  
1. Add debug statement to log to standard error along with the message to update the task’s status message. Web UI makes it easier to view.  
2. Create a custom counter, it gives valuable information to deal with the problem dataset  
3. Task page and task detailed page  
4. Hadoop Logs  
5. MRUnit testing

**PROGRAM 1: Counting the number of words in an input file**  
**Steps**  
1. **Create the input file**  
Create the input.txt file with sample text.  
$ vi input.txt  
Thanks Lord Krishna for helping us write this book  
Hare Krishna Hare Krishna Krishna Krishna Hare Hare  
Hare Rama Hare Rama Rama Rama Hare Hare  
2. **Move the input file into HDFS**  
Use the –put or –copyFromLocal command to move the file into HDFS  
$ hadoop fs -put input.txt  
3. **Code for the MapReduce program**  
Java files:  
WordCountProgram.java  // Driver Program  
WordMapper.java         // Mapper Program  
WordReducer.java        // Reducer Program  
————————————————–  
WordCountProgram.java File: Driver Program  
————————————————–  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.conf.Configured;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Job;  
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;  
import org.apache.hadoop.util.Tool;  
import org.apache.hadoop.util.ToolRunner;  
public class WordCountProgram extends Configured implements Tool{  
@Override  
public int run(String[] args) throws Exception {  
Configuration conf = new Configuration();  
Job job = new Job(conf, “wordcountprogram”);  
job.setJarByClass(getClass());  
// Configure output and input source  
TextInputFormat.addInputPath(job, new Path(args[0]));  
job.setInputFormatClass(TextInputFormat.class);  
job.setMapperClass(WordMapper.class);  
job.setReducerClass(WordReducer.class);  
// Configure output  
TextOutputFormat.setOutputPath(job, new Path(args[1]));  
job.setOutputFormatClass(TextOutputFormat.class);  
job.setOutputKeyClass(Text.class);  
job.setOutputValueClass(IntWritable.class);  
return job.waitForCompletion(true) ? 0 : 1;  
}  
public static void main(String[] args) throws Exception {  
int exitCode = ToolRunner.run(new WordCountProgram(), args);  
System.exit(exitCode);  
}  
}  
————————————————–  
WordMapper.java File: Mapper Program  
————————————————–  
import java.io.IOException;  
import java.util.StringTokenizer;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Mapper;  
public class WordMapper extends Mapper {  
private final static IntWritable count = new IntWritable(1);  
private final Text nameText = new Text();  
@Override  
protected void map(LongWritable key, Text value, Context context) throws IOException,  
InterruptedException {  
StringTokenizer tokenizer = new StringTokenizer(value.toString(),” “);  
while (tokenizer.hasMoreTokens()) {  
nameText.set(tokenizer.nextToken());  
context.write(nameText, count);  
}  
}  
}  
———————————————–  
WordReducer.java file: Reducer Program  
————————————————–  
import java.io.IOException;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Reducer;  
public class WordReducer extends Reducer {  
@Override  
protected void reduce(Text t, Iterable counts, Context context)  
throws IOException, InterruptedException {  
int sum = 0;  
for (IntWritable count : counts) {  
sum += count.get();  
}  
context.write(t, new IntWritable(sum));  
}  
}

4. **Run the MapReduce program**  
Create the jar of the Code in Step 3 and use the following command to run the MapReduce program  
$ hadoop jar WordCount.jar WordCountProgram input.txt output1  
Here,  
WordCount.jar: Name of jar exported having the all the methods.  
WordCountProgram: Driver Program having the entire configuration  
input.txt: Input file  
output1: Output folder where the output file will be stored

5. **View the Output**  
View the output in the output1 folder  
$ hadoop fs -cat /user/cloudera/output1/part-r-00000  
Hare  8  
Krishna     5  
Lord  1  
Rama  4  
Thanks      1  
book  1  
for   1  
helping     1  
this  1  
us    1  
write 1

**Q. What problem does Apache Flume solve?**  
**Scenario:**  
1. There are several services producing a huge number of logs that run in different servers. These logs need to be accumulated, stored and analyzed together.  
2. Hadoop has emerged as a cost effective and scalable framework for storage and analysis for big data.  
**Problem:**  
1. How can these logs be collected, aggregated and stored to a place where Hadoop can process them?  
2. Now there is a requirement for a reliable, scalable, extensible and manageable solution.

**Q. What is Apache Flume?**  
Apache Flume is a distributed data collection service that gets flows of data (like logs) from the systems that generates them and aggregates them to a centralized data store where they can be processed together.  
**Goals:** reliability, recoverability and scalability  
**Flume features:**  
1. Ensures guaranteed data delivery  
2. Gather high volume data streams in real time  
3. Streaming data is coming from multiple sources into Hadoop for analysis  
4. Scales horizontally

**Q. How is Flume-NG different from Flume 0.9?**  
**Flume 0.9:**  
Centralized configuration of the agents handled by Zookeeper.  
Input data and writing data are handled by same thread.  
Flume 1.X (Flume-NG):  
No centralized configuration. Instead a simple on-disk configuration file is used.  
Different threads called **runners** handle input data and writing data.

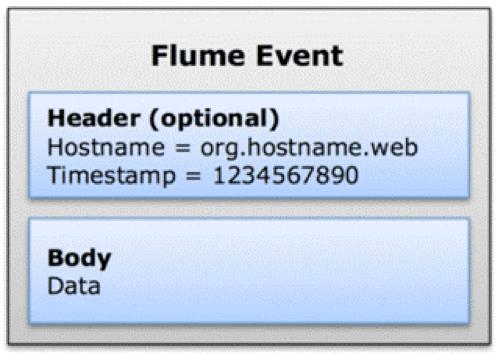
**Q. What is the problem with HDFS and streaming data (like logs)?**  
1. In a regular filesystem when you open a file and write data, it exists on disk even before it is closed.  
2. Whereas in HDFS, the file exists only as a directory entry of zero length till it is closed. This implies that if data is written to a file for an extended period without closing it, you may be left with an empty file if there is a network disconnect with the client.  
3. It is not a good approach to close the files frequently and create smaller files as this leads to poor efficiency in HDFS.

**Q. What are core components of Flume?**

**Flume Agent:**  
1. An agent is a daemon (physical Java virtual machine) running Flume.  
2. It receives and stores the data until it is written to a next destination.  
3. Flume source, channel and sink run in an agent.  
**Source:**  
1. A source receives data from some application that is producing data.  
2. A source writes events to one or more channels.  
3. Sources either poll for data or wait for data to be delivered to them.  
4.**For Example:** log4j, Avro, syslog, etc.  
**Sink:**  
1. A sink removes the events from the agent and delivering it to the destination.  
2. The destination could be different agent or HDFS, HBase, Solr etc.  
3.**For Example:**Console, HDFS, HBase, etc.  
**Channel:**  
1. A channel holds events passing from a source to a sink.  
2. A source ingests events into the channel while sink removes them.  
3. A sink gets events from one channel only.  
4.**For Example:** Memory, File, JDBC etc.

**Q. Explain a common use case for Flume?**  
***Common Use case:***Receiving web logs from several sources into HDFS.  
Web server logs → Apache Flume → HDFS (Storage) → Pig/Hive (ETL) → HBase (Database) → Reporting (BI Tools)  
1. Logs are generated by several log servers and saved in local hard disks, which need to be pushed into HDFS using Flume framework.  
2. Flume agents, which are running on, log servers collect the logs, which are pushed into HDFS.  
3. Data analytics tools like Pig or Hive then process this data.  
4. The analysed data is stored in structured format in HBase or other database.  
5. Business intelligence tools will then generate reports on this data.

**Q. What are Flume events?**  
**Flume events:**  
1. Basic **payload of data** transported by Flume (typically a single log entry)  
2. It has zero or more headers and a body



Event Headers are key-value pairs that are used to make routing decisions or carry other structured information like:  
1. Timestamp of the event  
2. Hostname of the server where event has originated  
**Event Body**  
Event Body is an array of bytes that contains the actual payload.

**Q. Can we change the body of the flume event?**  
Yes, editing Flume Event using interceptors can change its body.

**Q. What are interceptors?**

**Interceptor**  
An interceptor is a point in your data flow where you can inspect and alter flume events. After the source creates an event, there can be zero or more interceptors tied together before it is delivered to sink.

**Q. What are channel selectors?**  
**Channel selectors:**  
Channel selectors are responsible for how an event moves from a source to one or more channels.  
Types of channel selectors are:  
**1. Replicating Channel Selector:** This is the default channel selector that puts a copy of event into each channel  
**2. Multiplexing Channel Selector:** Routes data into different channel depending on header information and/or interceptor logic

**Q. What are sink processors?**  
**Sink processor:**  
Sink processor is a mechanism for failover and load balancing events across multiple sinks from a channel

**Q. How to Configure an Agent?**  
1. An agent is configured using a simple Java property file of key/value pairs  
2. This configuration file is passed as an argument to the agent upon startup.  
3. You can configure multiple agents in a single configuration file. It is required to pass an agent identifier (called a name).  
4. Each agent is configured starting with:  
agent.sources=  
agent.channels=  
agent.sinks=  
5. Each source, channel and sink also has a distinct name within the context of that agent.

**Q. Explain the “Hello world” example in flume.**  
In the following example, the source listens on a socket for network clients to connect and sends event data. Those events were written to an in-memory channel and then fed to a log4j sink to become output.  
Configuration file for one agent (called a1) that has a source named s1, a channel named c1 and a sink named k1.  
# netcatAgent.conf: Logs the netcat events to console  
# Name of the components on this agent  
a1.sources=s1  
a1.channels=c1  
a1.sinks=k1  
# Configure the source  
a1.sources.s1.type=Netcat  
   
**Q. What is Hive?**  
Hive is a Hadoop based system for querying and analyzing large volumes of Structured data which is stored on HDFS or in other words Hive is an query engine built to work on top of Hadoop that can compile queries into Map Reduce jobs and run them on the cluster.

**Q. In which scenario Hive is good fit?**  
1. Data warehousing applications where more static data is analyzed.  
2. Fast response time is not the criteria.  
3. Data is not changing rapidly.  
4. An abstract to underlying MapReduce programs  
5. Like SQL

**Q. What are the limitations of Hive?**  
Hive does not provide:  
1. Record-level operations like INSERT, DELETE or UPDATE.  
2. Cannot be used for low latency jobs.  
3. Transaction.

**Q. What are the differences between Hive and RDBMS?**  
HIVE:  
Schema on Read  
Batch processing jobs  
Data stored on HDFS  
Processed using MapReduce  
RDBMS:  
Schema on write  
Real time jobs  
Data stored on internal structure  
Processed using database

**Q. What are the components of Hive architecture?**  
Hive Driver  
Metastore  
Hive CLI/HUE/HWI

**Q. What is the purpose of Hive Driver?**  
Hive Driver is responsible for compiling, optimizing and then executing the HiveQL.

**Q. What is a Metastore and what it stores?**  
1. It is a database by default Derby SQL server  
2. Holds metadata about table definition, column, and data types partitioning information,  
3. It can be stored in MySQL, derby, oracle etc.

**Q. What is the purpose of storing the metadata?**  
People want to read the dataset with a particular schema in mind.  
For e.g.: BA and CFO of a company look at the data with a particular schema.  
BA may be interested in say IP addresses and timings of the clicks in a weblog while the CFO may be interested in say the clicks that were direct clicks on the website or from paid Google adds.  
Underneath it’s the same dataset that is accessed. This schema is used again and again. So it makes sense to store this schema in a RDBMS.

**Q. List the various options available with the Hive command.**  
**Syntax:**  
$ ./hive –service serviceName  
where  
serviceName options are:  
cli  
help  
hiveserver  
hwi  
jar  
lineage  
metastore  
rcfile

**Q. Explain the different services that can be invoked using the Hive command.**  
cli  
1. default service  
2. used to define tables, run queries, etc.  
hiveserver  
1. Daemon that listens for Thrift connections from other processes  
hwi  
1. Simple web interface for running queries  
jar  
1. Extension of the hadoop jar command  
metastore  
1. External Hive metastore service to support multiple clients  
rcfile  
1. Tool for printing the contents of an RFFile

**Q. Can you execute Hadoop dfs Commands from Hive CLI? How?**  
Hadoop dfs commands can be run from within the hive CLI by dropping the hadoop work from the command and adding a semicolon in the end.  
For Example:  
Hadoop dfs command:  
hadoop dfs -ls /  
From within hive  
hive > dfs -ls / ;

**Q. How to give multiline comments in Hive Scripts?**  
Hive does not support multiline comments. All lines of comments should start with the string —  
For e.g.  
— This is first line of comment  
— This is second line of comment !!

**Q. What is the reason for creating a new metastore\_db whenever Hive query is run from a different directory?**  
**Embedded mode:**  
Whenever Hive runs in embedded mode, it checks whether the metastore exists. If the metastore does not exist then it creates the local metastore.  
**Property:** Default value  
**javax.jdo.option.ConnectionURL** = “jdbc:derby:;databaseName=metastore\_db;create=true”

**Q. When Hive is run in embedded mode, how to share the metastore within multiple users?**  
No.  
For sharing use the standalone database (like MySQL, PostGresQL) for metastore

**Q. How can an application connect to Hive run as a server?**  
Thrift Client: Hive commands can be called hive command from programming languages like Java, PHP, Python, Ruby, C++  
JDBC Driver: Type 4 (pure Java) JDBC Driver  
ODBC driver:  ODBC protocol

**Q. List the Primitive Data Types?**  
**DataTypes:**  
TINYINT

**Q. What problem does Apache Pig solve?**  
**Scenario**  
1. MapReduce paradigm presented by Hadoop is low level and rigid so developing can be challenging.  
2. Jobs are (mainly) in Java where developer needs to think in terms of map and reduce  
**Problem**  
1. Many common operations like filters, projections, joins requires a custom code  
2. Not everyone is a Java expert!!!  
3. MapReduce has a long development cycle

**Q. What is Apache Pig?**  
Apache Pig is a platform for analyzing large data sets that consists high-level language for expressing data analysis programs, with infrastructure for evaluating these programs.  
**Goals:** Ease of programming, Improved Code readability, Flexible, Extensible  
**Pig Features:**  
Ease of programming:  
1. Generates MapReduce programs automatically  
2. Fewer lines of code  
Flexible:  
1. Metadata is optional  
Extensible:  
1. Easy extensible by UDFs  
Resides on the client machine

**Q. In which scenario MapReduce is a better fit than Pig?**  
Some problems are harder to express in Pig. For example:  
1. Complex grouping or joins  
2. Combining lot of datasets  
3. Replicated join  
4. Complex cross products  
In such cases, Pig’s MAPREDUCE relational operator can be used which allows plugging in Java MapReduce job.

**Q. In which scenario Pig is better fit than MapReduce?**  
Pig provides common data operations (joins, filters, group by, order by, union) and nested data types (tuple, bag and maps), which are missing from MapReduce.

**Q. Where not to use Pig?**  
1. Completely unstructured data. For example: images, audio, video  
2. When more power to optimize the code is required  
3. Retrieving a single record in a very large dataset

**Q. What can be feed to Pig?**  
We can input structured, semi-structured or unstructured data to Pig.  
*For example*, CSV’s, TSV’s, Delimited Data, Logs

**Q. What are the components of Apache Pig platform?**  
**Pig Engine**  
Parser, Optimizer and produces sequences of MapReduce programs  
**Grunt**  
Pig’s interactive shell  
It allows users to enter Pig Latin interactively and interact with HDFS  
**Pig Latin**  
High level and easy to understand dataflow language  
Provides ease of programming, extensibility and optimization.

**Q. What are the execution modes in Pig?**  
Pig has **two** execution modes:  
**Local mode**  
No Hadoop / HDFS installation is required  
All processing takes place in only one local JVM  
Used only for quick prototyping and debugging Pig Latin script  
pig -x local  
**MapReduce mode (Default)**  
Parses, checks and optimizes locally  
1. Plans execution as one MapReduce job  
2. Submits job to Hadoop  
3. Monitors job progress  
pig or pig -x mapreduce

**Q. Different running modes for running Pig?**  
Pig has **two** running modes:  
**Interactive mode**  
Pig commands runs one at a time in the grunt shell  
**Batch mode**  
Commands are in pig script file.

**Q. What are the different ways to develop PigLatin scripts?**  
Plugins are available which features such as syntax/error highlighting, auto completion etc.  
Eclipse plugins  
1. PigEditor  
2. PigPen  
3. Pig-Eclipse  
Vim, Emacs, TextMate plugins also available

**Q. What are the Data types in Pig?**  
**Scalar Types**  
Int, long, float, double, chararray, bytearray, boolean (since Release 0.10.0)  
**Complex Types**  
Map, Tuple, Bag

**Q. Which type in Pig is not required to fit in Memory?**  
1. Bag is the type not required to fit in memory, as it can be quite large.  
2. It can store bags to disk when necessary.

**Q. What is a Map in Pig?**  
Map is a chararray to data element mapping, where data element be of any Pig data type.  
It can also be called as a set of key-value pairs where  
     Keys → chararray and Values → any pig data type  
For example [‘student’#’Mahi’, ’Rank’#1]

**Q. What is a Tuple in Pig? (~ RDBMS row in a table)**  
A tuple is an ordered set of fields; fields can be of any data type.  
It can also be called as a sequence of fields of any type