Apache MapReduce

Apache MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster.

1) How does Hadoop process records split across block boundaries? Suppose a record line is split across two blocks (b1 and b2). The mapper processing the first block (b1) will notice that the last line doesn't have a EOL separator and fetches the remaining of the line from the next block of data (b2).How does the mapper processing the second block (b2) determine that the first record is incomplete and should process starting from the second record in the block (b2)?

Answer)Map Reduece algorithm does not work on physical blocks of the file. It works on logical input splits. Input split depends on where the record was written. A record may span two Mappers.The way HDFS has been set up, it breaks down very large files into large blocks (for example, measuring 128MB), and stores three copies of these blocks on different nodes in the cluster.HDFS has no awareness of the content of these files. A record may have been started in Block-a but end of that record may be present in Block-b.To solve this problem, Hadoop uses a logical representation of the data stored in file blocks, known as input splits. When a MapReduce job client calculates the input splits, it figures out where the first whole record in a block begins and where the last record in the block ends.

The Map-Reduce framework relies on the InputFormat of the job to:

Validate the input-specification of the job.

Split-up the input file(s) into logical InputSplits, each of which is then assigned to an individual Mapper.

Each InputSplit is then assigned to an individual Mapper for processing. Split could be tuple. InputSplit[] getSplits(JobConf job,int numSplits) is the API to take care of these things.

FileInputFormat, which extends InputFormat implemented getSplits() method.

2) In mapreduce each reduce task write its output to a file named part-r-nnnnn where nnnnn is a partition ID associated with the reduce task. How to merge output files after reduce phase

Answer)We can delegate the entire merging of the reduce output files to hadoop by calling:

hadoop fs -getmerge /output/dir/on/hdfs/ /desired/local/output/file.txt

3) Can you set number of map task in Map reduce?

Answer)The number of map tasks for a given job is driven by the number of input splits. For each input split a map task is spawned. So, over the lifetime of a mapreduce job the number of map tasks is equal to the number of input splits.

4) If your Mapreduce Job launches 20 task for 1 job can you limit to 10 task?

Answer)Yes by setting mapreduce.jobtracker.maxtasks.perjob to 10 in mapred-default.xml file

5) What is the default value set for mapreduce.jobtracker.maxtasks.perjob ?

Answer)Default value is -1 indicates that there is no maximum

6) Have you ever faced Container is running beyond memory limits? For example Container [pid=28921,containerID=container_1389136889968_0001_01_000121] is running beyond virtual memory limits. Current usage: 1.2 GB of 1 GB physical memory used; 2.2 GB of 2.1 GB virtual memory used. Killing container. How to handle this issue?

Answer) For our example cluster, we have the minimum RAM for a Container (yarn.scheduler.minimum-allocation-mb) = 2 GB. We'll thus assign 4 GB for Map task Containers, and 8 GB for Reduce tasks Containers.
In mapred-site.xml:
mapreduce.map.memory.mb: 4096
mapreduce.reduce.memory.mb: 8192

Each Container will run JVMs for the Map and Reduce tasks. The JVM heap size should be set to lower than the Map and Reduce memory defined above, so that they are within the bounds of the Container memory allocated by YARN.
In mapred-site.xml:
mapreduce.map.java.opts: -Xmx3072m
mapreduce.reduce.java.opts: -Xmx6144m

The above settings configure the upper limit of the physical RAM that Map and Reduce tasks will use.

7))What is Shuffling and Sorting in Hadoop MapReduce?

Answer)Shuffling in MapReduce

The process of transferring data from the mappers to reducers is known as shuffling i.e. the process by which the system performs the sort and transfers the map output to the reducer as input. So, MapReduce shuffle phase is necessary for the reducers, otherwise, they would not have any input (or input from every mapper). As shuffling can start even before the map phase has finished so this saves some time and completes the tasks in lesser time.

Sorting in MapReduce

The keys generated by the mapper are automatically sorted by MapReduce Framework, i.e. Before starting of reducer, all intermediate key-value pairs in MapReduce that are generated by mapper get sorted by key and not by value. Values passed to each reducer are not sorted; they can be in any order.

8) What steps do you follow in order to improve the performace of Mapreduce Job?

Answer) There are some general guidelines to improve the performance.

If each task takes less than 30-40 seconds, reduce the number of tasks

If a job has more than 1TB of input, consider increasing the block size of the input dataset to 256M or even 512M so that the number of tasks will be smaller.

Number of reduce tasks per a job should be equal to or a bit less than the number of reduce slots in the cluster.

Some more tips :
Configure the cluster properly with right diagnostic tools
Use compression when you are writing intermediate data to disk
Tune number of Map and Reduce tasks as per above tips
Incorporate Combiner wherever it is appropriate
Use Most appropriate data types for rendering Output (Do not use LongWritable when range of output values are in Integer range. IntWritable is right choice in this case)
Reuse Writables
Have right profiling tools

9)What is the purpose of shuffling and sorting phase in the reducer in Map Reduce Programming?

Answer)First of all shuffling is the process of transfering data from the mappers to the reducers, so I think it is obvious that it is necessary for the reducers, since otherwise, they wouldn't be able to have any input (or input from every mapper). Shuffling can start even before the map phase has finished, to save some time. That's why you can see a reduce status greater than 0% (but less than 33%) when the map status is not yet 100%.

Sorting saves time for the reducer, helping it easily distinguish when a new reduce task should start. It simply starts a new reduce task, when the next key in the sorted input data is different than the previous, to put it simply. Each reduce task takes a list of key-value pairs, but it has to call the reduce() method which takes a key-list(value) input, so it has to group values by key. It's

easy to do so, if input data is pre-sorted (locally) in the map phase and simply merge-sorted in the reduce phase (since the reducers get data from many mappers).

Partitioning, that you mentioned in one of the answers, is a different process. It determines in which reducer a (key, value) pair, output of the map phase, will be sent. The default Partitioner uses a hashing on the keys to distribute them to the reduce tasks, but you can override it and use your own custom Partitioner.

10)How do I submit extra content (jars, static files, etc) for Mapreduce job to use during runtime?

Answer)The distributed cache feature is used to distribute large read-only files that are needed by map/reduce jobs to the cluster. The framework will copy the necessary files from a URL on to the slave node before any tasks for the job are executed on that node. The files are only copied once per job and so should not be modified by the application.

11)How do I get my MapReduce Java Program to read the Cluster's set configuration and not just defaults?

Answer)The configuration property files ({core|mapred|hdfs}-site.xml) that are available in the various conf/ directories of your Hadoop installation needs to be on the CLASSPATH of your Java application for it to get found and applied. Another way of ensuring that no set configuration gets overridden by any Job is to set those properties as final

12)How do I get each of a jobs maps to work on one complete input-file and not allow the framework to split-up the files?

Answer) Essentially a job's input is represented by the InputFormat[interface] or FileInputFormat[base class].

For this purpose one would need a non-splittable FileInputFormat i.e. an input-format which essentially tells the map-reduce framework that it cannot be split-up and processed. To do this you need your particular input-format to return false for the isSplittable call.

E.g.
org.apache.hadoop.mapred.SortValidator.RecordStatsChecker.NonSplitableSequenceFileInputFormat in src/test/org/apache/hadoop/mapred/SortValidator.java

In addition to implementing the InputFormat interface and having isSplitable() returning false, it is also necessary to implement the RecordReader interface for returning the whole content of the input file. Default is LineRecordReader, which splits the file into separate lines

13)You see a maximum of 2 maps/reduces spawned concurrently on each TaskTracker, how do you increase that?

Answer) Use the configuration knob: mapred.tasktracker.map.tasks.maximum and mapred.tasktracker.reduce.tasks.maximum to control the number of maps/reduces spawned simultaneously on a TaskTracker. By default, it is set to 2, hence one sees a maximum of 2 maps and 2 reduces at a given instance on a TaskTracker.

14)How do Map/Reduce InputSplit's handle record boundaries correctly?

Answer)It is the responsibility of the InputSplit's RecordReader to start and end at a record boundary. For SequenceFile's every 2k bytes has a 20 bytes sync mark between the records. These sync marks allow the RecordReader to seek to the start of the InputSplit, which contains a file, offset and length and find the first sync mark after the start of the split. The RecordReader continues processing records until it reaches the first sync mark after the end of the split. The first split of each file naturally starts immediately and not after the first sync mark. In this way, it is guaranteed that each record will be processed by exactly one mapper.

Text files are handled similarly, using newlines instead of sync marks.

15) How do I change final output file name with the desired name rather than in partitions like part-00000, part-00001?

Answer)You can subclass the OutputFormat.java class and write your own. You can locate and browse the code of TextOutputFormat, MultipleOutputFormat.java, etc. for reference. It might be the case that you only need to do minor changes to any of the existing Output Format classes. To do that you can just subclass that class and override the methods you need to change.

16)When writing a New InputFormat, what is the format for the array of string returned by InputSplit\#getLocations()?

Answer)It appears that DatanodeID.getHost() is the standard place to retrieve this name, and the machineName variable, populated in DataNode.java\#startDataNode, is where the name is first set. The first method attempted is to get "slave.host.name" from the configuration; if that is not available, DNS.getDefaultHost is used instead.

17)How do you gracefully stop a running job?

Answer)hadoop job -kill JOBID

18)How do I limit Limiting Task Slot Usage

Answer)There are many reasons why one wants to limit the number of running tasks.

Job is consuming all task slots

The most common reason is because a given job is consuming all of the available task slots, preventing other jobs from running. The easiest and best solution is to switch from the default FIFO scheduler to another scheduler, such as the FairShareScheduler or the CapacityScheduler. Both solve this problem in slightly different ways. Depending upon need, one may be a better fit than the other.

Job has taken too many reduce slots that are still waiting for maps to finish

There is a job tunable called mapred.reduce.slowstart.completed.maps that sets the percentage of maps that must be completed before firing off reduce tasks. By default, this is set to 5% (0.05) which for most shared clusters is likely too low. Recommended values are closer to 80% or higher (0.80). Note that for jobs that have a significant amount of intermediate data, setting this value higher will cause reduce slots to take more time fetching that data before performing work.

Job is referencing an external, limited resource (such as a database)

In Hadoop terms, we call this a 'side-effect'.

One of the general assumptions of the framework is that there are not any side-effects. All tasks are expected to be restartable and a side-effect typically goes against the grain of this rule.

If a task absolutely must break the rules, there are a few things one can do:

Disable SpeculativeExecution .

Deploy ZooKeeper and use it as a persistent lock to keep track of how many tasks are running concurrently

Use a scheduler with a maximum task-per-queue feature and submit the job to that queue, such as FairShareScheduler or CapacityScheduler

19)How to increase the number of slots used?

Answer)There are both job and server-level tunables that impact how many tasks are run concurrently.

Increase the amount of tasks per node

There are two server tunables that determine how many tasks a given TaskTracker will run on a node:
mapred.tasktracker.map.tasks.maximum sets the map slot usage
mapred.tasktracker.reduce.tasks.maximum sets the reduce slot usage

These must be set in the mapred-site.xml file on the TaskTracker. After making the change, the TaskTracker must be restarted to see it. One should see the values increase (or decrease) on the JobTracker main page. Note that this is not set by your job.

Increase the amount of map tasks

Typically, the amount of maps per job is determined by Hadoop based upon the InputFormat and the block size in place. Using mapred.min.split.size and mapred.max.split.size settings, one can provide hints to the system that it should use a size that is different than the block size to determine what the min and max input size should be.

Increase the amount of reduce tasks

Currently, the number of reduces is determined by the job. mapred.reduce.tasks should be set by the job to the appropriate number of reduces. When using Pig, use the PARALLEL keyword.

20)When do reduce tasks start in Hadoop?Do they start after a certain percentage (threshold) of mappers complete? If so, is this threshold fixed? What kind of threshold is typically used?

Answer)The reduce phase has 3 steps: shuffle, sort, reduce. Shuffle is where the data is collected by the reducer from each mapper. This can happen while mappers are generating data since it is only a data transfer. On the other hand, sort and reduce can only start once all the mappers are done. You can tell which one MapReduce is doing by looking at the reducer completion percentage: 0-33% means its doing shuffle, 34-66% is sort, 67%-100% is reduce. This is why your reducers will sometimes seem "stuck" at 33%- it's waiting for mappers to finish.

Reducers start shuffling based on a threshold of percentage of mappers that have finished. You can change the parameter to get reducers to start sooner or later.

Why is starting the reducers early a good thing? Because it spreads out the data transfer from the mappers to the reducers over time, which is a good thing if your network is the bottleneck.

Why is starting the reducers early a bad thing? Because they "hog up" reduce slots while only copying data and waiting for mappers to finish. Another job that starts later that will actually use the reduce slots now can't use them.

You can customize when the reducers startup by changing the default value of mapred.reduce.slowstart.completed.maps in mapred-site.xml. A value of 1.00 will wait for all the mappers to finish before starting the reducers. A value of 0.0 will start the reducers right away. A value of 0.5 will start the reducers when half of the mappers are complete. You can also change mapred.reduce.slowstart.completed.maps on a job-by-job basis. In new versions of Hadoop (at least 2.4.1) the parameter is called is mapreduce.job.reduce.slowstart.completedmaps

Typically, I like to keep mapred.reduce.slowstart.completed.maps above 0.9 if the system ever has multiple jobs running at once. This way the job doesn't hog up reducers when they aren't doing anything but copying data. If you only ever have one job running at a time, doing 0.1 would probably be appropriate.

21) How do you do chaining of multiple Mapreduce job in Hadoop? In many real-life situations where you apply MapReduce, the final algorithms end up being several MapReduce steps.i.e. Map1 , Reduce1 , Map2 , Reduce2 , and so on.So you have the output from the last reduce that is needed as the input for the next map.The intermediate data is something you (in general) do not want to keep once the pipeline has been successfully completed. Also because this intermediate data is in general some data structure (like a

'map' or a 'set') you don't want to put too much effort in writing and reading these key-value pairs.What is the recommended way of doing that in Hadoop?

Answer)You use the JobClient.runJob(). The output path of the data from the first job becomes the input path to your second job. These need to be passed in as arguments to your jobs with appropriate code to parse them and set up the parameters for the job.

I think that the above method might however be the way the now older mapred API did it, but it should still work. There will be a similar method in the new mapreduce API but i'm not sure what it is.

As far as removing intermediate data after a job has finished you can do this in your code. The way i've done it before is using something like:

FileSystem.delete(Path f, boolean recursive);

Where the path is the location on HDFS of the data. You need to make sure that you only delete this data once no other job requires it.

There are many ways you can do it.

(1) Cascading jobs

Create the JobConf object "job1" for the first job and set all the parameters with "input" as inputdirectory and "temp" as output directory. Execute this job:

JobClient.run(job1).

Immediately below it, create the JobConf object "job2" for the second job and set all the parameters with "temp" as inputdirectory and "output" as output directory. Execute this job:

JobClient.run(job2).

(2) Create two JobConf objects and set all the parameters in them just like (1) except that you don't use JobClient.run.

Then create two Job objects with jobconfs as parameters:
Job job1=new Job(jobconf1);
Job job2=new Job(jobconf2);
Using the jobControl object, you specify the job dependencies and then run the jobs:
JobControl jbcntrl=new JobControl("jbcntrl");
jbcntrl.addJob(job1);
jbcntrl.addJob(job2);
job2.addDependingJob(job1);
jbcntrl.run();

(3) If you need a structure somewhat like Map+ | Reduce | Map*, you can use the ChainMapper and ChainReducer classes that come with Hadoop version 0.19 and onwards.

22)Can you explain me how secondary sorting works in hadoop ? Why must one use GroupingComparator and how does it work in hadoop ?

Answer)Grouping Comparator

Once the data reaches a reducer, all data is grouped by key. Since we have a composite key, we need to make sure records are grouped solely by the natural key. This is accomplished by writing a custom GroupPartitioner. We have a Comparator object only considering the yearMonth field of the TemperaturePair class for the purposes of grouping the records together.

```
public class YearMonthGroupingComparator extends WritableComparator {
public YearMonthGroupingComparator() {
super(TemperaturePair.class, true);
}
@Override
public int compare(WritableComparable tp1, WritableComparable tp2) {
TemperaturePair temperaturePair = (TemperaturePair) tp1;
TemperaturePair temperaturePair2 = (TemperaturePair) tp2;
return temperaturePair.getYearMonth().compareTo(temperaturePair2.getYearMonth());
}
}
```

Here are the results of running our secondary sort job:

```
new-host-2:sbin bbejeck$ hdfs dfs -cat secondary-sort/part-r-00000


190101 -206
190102 -333
190103 -272
190104 -61
190105 -33
190106 44
190107 72
190108 44
```

While sorting data by value may not be a common need, it's a nice tool to have in your back pocket when needed. Also, we have been able to take a deeper look at the inner workings of Hadoop by working with custom partitioners and group partitioners.

23)Explain about the basic parameters of mapper and reducer function.

Answer)Mapper Function Parameters

The basic parameters of a mapper function are LongWritable, text, text and IntWritable.

LongWritable, text- Input Parameters

Text, IntWritable- Intermediate Output Parameters

Here is a sample code on the usage of Mapper function with basic parameters –

```
public static class Map extends MapReduceBase implements Mapper {
private final static IntWritable one = new IntWritable (1);
private Text word = new Text () ;}
```

Reducer Function Parameters

The basic parameters of a reducer function are text, IntWritable, text, IntWritable

First two parameters Text, IntWritable represent Intermediate Output Parameters

The next two parameters Text, IntWritable represent Final Output Parameters

24)How data is spilt in Hadoop?

Answer)The InputFormat used in the MapReduce job create the splits. The number of mappers are then decided based on the number of splits. Splits are not always created based on the HDFS block size. It all depends on the programming logic within the getSplits () method of InputFormat.

25)What is the fundamental difference between a MapReduce Split and a HDFS block?

Answer)MapReduce split is a logical piece of data fed to the mapper. It basically does not contain any data but is just a pointer to the data. HDFS block is a physical piece of data.

26) When is it not recommended to use MapReduce paradigm for large scale data processing?

Answer)It is not suggested to use MapReduce for iterative processing use cases, as it is not cost effective, instead Apache Pig can be used for the same.

27) What happens when a DataNode fails during the write process?

Answer)When a DataNode fails during the write process, a new replication pipeline that contains the other DataNodes opens up and the write process resumes from there until the file is closed. NameNode observes that one of the blocks is under-replicated and creates a new replica asynchronously.

28) List the configuration parameters that have to be specified when running a MapReduce job.

Answer)Input and Output location of the MapReduce job in HDFS.

Input and Output Format.

Classes containing the Map and Reduce functions.

JAR file that contains driver classes and mapper, reducer classes.

29) Is it possible to split 100 lines of input as a single split in MapReduce?

Answer)Yes this can be achieved using Class NLineInputFormat

30) Where is Mapper output stored?

Answer)The intermediate key value data of the mapper output will be stored on local file system of the mapper nodes. This directory location is set in the config file by the Hadoop Admin. Once the Hadoop job completes execution, the intermediate will be cleaned up.

31) Explain the differences between a combiner and reducer.

Answer)Combiner can be considered as a mini reducer that performs local reduce task. It runs on the Map output and produces the output to reducers input. It is usually used for network optimization when the map generates greater number of outputs.

Unlike a reducer, the combiner has a constraint that the input or output key and value types must match the output types of the Mapper.

Combiners can operate only on a subset of keys and values i.e. combiners can be executed on functions that are commutative.

Combiner functions get their input from a single mapper whereas reducers can get data from multiple mappers as a result of partitioning.

32) When is it suggested to use a combiner in a MapReduce job?

Answer)Combiners are generally used to enhance the efficiency of a MapReduce program by aggregating the intermediate map output locally on specific mapper outputs. This helps reduce the volume of data that needs to be transferred to reducers. Reducer code can be used as a combiner, only if the operation performed is commutative. However, the execution of a combiner is not assured.

33)What is the relationship between Job and Task in Hadoop?

Answer)A single job can be broken down into one or many tasks in Hadoop.

34)Is it important for Hadoop MapReduce jobs to be written in Java?

Answer)It is not necessary to write Hadoop MapReduce jobs in Java but users can write MapReduce jobs in any desired programming language like Ruby, Perl, Python, R, Awk, etc. through the Hadoop Streaming API.

35) What is the process of changing the split size if there is limited storage space on Commodity Hardware?

Answer)If there is limited storage space on commodity hardware, the split size can be changed by implementing the "Custom Splitter". The call to Custom Splitter can be made from the main method.

36) What are the primary phases of a Reducer?

Answer)The 3 primary phases of a reducer are –
1) Shuffle
2) Sort
3) Reduce

37) What is a TaskInstance?

Answer)The actual Hadoop MapReduce jobs that run on each slave node are referred to as Task instances. Every task instance has its own JVM process. For every new task instance, a JVM process is spawned by default for a task.

38) Can reducers communicate with each other?

Answer)Reducers always run in isolation and they can never communicate with each other as per the Hadoop MapReduce programming paradigm.

39) What is the difference between Hadoop and RDBMS?

Answer)In RDBMS, data needs to be pre-processed being stored, whereas Hadoop requires no pre-processing.

RDBMS is generally used for OLTP processing whereas Hadoop is used for analytical requirements on huge volumes of data.

Database cluster in RDBMS uses the same data files in shared storage whereas in Hadoop the storage is independent of each processing node.

40) Can we search files using wildcards?

Answer)Yes, it is possible to search for file through wildcards.

41) How is reporting controlled in hadoop?

Answer)The file hadoop-metrics.properties file controls reporting.

42) What is the default input type in MapReduce?

Answer) Text

43) Is it possible to rename the output file?

Answer)Yes, this can be done by implementing the multiple format output class.

44) What do you understand by compute and storage nodes?

Answer) Storage node is the system, where the file system resides to store the data for processing.

Compute node is the system where the actual business logic is executed.

45) When should you use a reducer?

Answer) It is possible to process the data without a reducer but when there is a need to combine the output from multiple mappers – reducers are used. Reducers are generally used when shuffle and sort are required.

46) What is the role of a MapReduce partitioner?

Answer)MapReduce is responsible for ensuring that the map output is evenly distributed over the reducers. By identifying the reducer for a particular key, mapper output is redirected accordingly to the respective reducer.

47) What is identity Mapper and identity reducer?

Answer) IdentityMapper is the default Mapper class in Hadoop. This mapper is executed when no mapper class is defined in the MapReduce job.

IdentityReducer is the default Reducer class in Hadoop. This mapper is executed when no reducer class is defined in the MapReduce job. This class merely passes the input key value pairs into the output directory.

48) What do you understand by the term Straggler ?

Answer) A map or reduce task that takes unsually long time to finish is referred to as straggler.

49) What is a MapReduce Combiner?

Answer)Also known as semi-reducer, Combiner is an optional class to combine the map out records using the same key. The main function of a combiner is to accept inputs from Map Class and pass those key-value pairs to Reducer class

50)What is RecordReader in a Map Reduce?

Answer)RecordReader is used to read key/value pairs form the InputSplit by converting the byte-oriented view and presenting record-oriented view to Mapper.

51)What is OutputCommitter?

Answer)OutPutCommitter describes the commit of MapReduce task. FileOutputCommitter is the default available class available for OutputCommitter in MapReduce. It performs the following operations:

Create temporary output directory for the job during initialization.

Then, it cleans the job as in removes temporary output directory post job completion.

Sets up the task temporary output.

Identifies whether a task needs commit. The commit is applied if required.

JobSetup, JobCleanup and TaskCleanup are important tasks during output commit.

52)Explain what happens when Hadoop spawned 50 tasks for a job and one of the task failed?

Answer)It will restart the task again on some other TaskTracker if the task fails more than the defined limit.

53)What is the key difference between Fork/Join and Map/Reduce? Do they differ in the kind of decomposition and distribution (data vs. computation)?

Answer)One key difference is that F-J seems to be designed to work on a single Java VM, while M-R is explicitly designed to work on a large cluster of machines. These are very different scenarios.

F-J offers facilities to partition a task into several subtasks, in a recursive-looking fashion; more tiers, possibility of 'inter-fork' communication at this stage, much more traditional programming. Does not extend (at least in the paper) beyond a single machine. Great for taking advantage of your eight-core.

M-R only does one big split, with the mapped splits not talking between each other at all, and then reduces everything together. A single tier, no inter-split communication until reduce, and massively scalable. Great for taking advantage of your share of the cloud.

54)What type of problems can mapreduce solve?

Answer)For problems requiring processing and generating large data sets. Say running an interest generation query over all accounts a bank hold. Say processing audit data for all transactions that happened in the past year in a bank. The best use case is from Google - generating search index for google search engine.

55)How to get the input file name in the mapper in a Hadoop program?

Answer)First you need to get the input split, using the newer mapreduce API it would be done as follows:

context.getInputSplit();

But in order to get the file path and the file name you will need to first typecast the result into FileSplit.

So, in order to get the input file path you may do the following:

```
Path filePath = ((FileSplit) context.getInputSplit()).getPath();
String filePathString = ((FileSplit) context.getInputSplit()).getPath().toString();
```

Similarly, to get the file name, you may just call upon getName(), like this:

```
String fileName = ((FileSplit) context.getInputSplit()).getPath().getName();
```

56)What is the difference between Hadoop Map Reduce and Google Map Reduce?

Answer)Google MapReduce and Hadoop are two different implementations (instances) of the MapReduce framework/concept. Hadoop is open source , Google MapReduce is not and actually there are not so many available details about it.

Since they work with large data sets, they have to rely on distributed file systems. Hadoop uses as a standard distributed file system the HDFS (Hadoop Distributed File Systems) while Google MapReduce uses GFS (Google File System)

Hadooop is implemented in java. Google MapReduce seems to be in C++.