Q1) Explain in your own words the main differences between Hadoop 1.0 and Hadoop 2.0

Ans 1.

Main Differences:

1. In Hadoop 1.0 computation model is MapReduce whereas in Hadoop 2.0 computation model is MapReduce and also supports other Applications as well.
2. In Hadoop 1.0 JobTracker manages scheduling and resource management while in Hadoop 2.0 the work of JobTracker is divided between the Application Master and the Scheduler.
3. In Hadoop 1.0 we have slots that run map or reduce tasks whereas in Hadoop 2.0 we have containers which run more generic tasks.
4. In Hadoop 1.0 we have a single namenode to manage the entire namespace while in Hadoop 2.0 we have multiple namenodes which can handle multiple namespaces.
5. In Hadoop 1.0 scales very less compared to Hadoop 2.0
6. In Hadoop 1.0 there is a single namenode which is single point of failure while in Hadoop 2.0 the HDFS comes with High Availability feature now, which solves this problem by providing the option of running two redundant Name Nodes in the same cluster in an Active/Passive way (one primary Name Node and other a hot standby Name Node) They both share an edits log. All namespace edits are logged to a shared NFS storage and there is only a single writer to this shared storage at any point of time. The passive Name Node reads from this storage and keeps updated metadata information for cluster. In case of Active Name Node failure, the passive Name Node becomes the Active Name Node and starts writing to the shared storage. There is only one write to the shared storage at any point of time.
7. Hadoop 1.0 treated all storage devices (be it spinning disks or SSDs) on a datanode as a single uniform pool; although one could store data on an SSD, one could not control which data. Heterogeneous storage is part of Hadoop 2.0 onwards, where the system will distinguish between storage types and also make the storage type information available to frameworks and applications so that they can take advantage of storage properties.

Source: Book: Hadoop: The definitive guide.

Q 2. Define *Yarn* in your own words

Ans 2 YARN,short for Yet Another Resource Negotiator solves the scalability problem of “classic”

MapReduce by splitting the responsibilities of the jobtracker into separate entities. The jobtracker takes

care of both job scheduling (matching tasks with tasktrackers) and task progress monitoring (keeping

track of tasks, restarting failed or slow tasks, and doing task bookkeeping, such as maintaining counter

totals).YARN separates these two roles into two independent daemons: a global resource manager and a

per application application master. The idea is that an application master negotiates with the resource

manager for cluster resources—described in terms of a number of containers, each with a certain memory limit—and then runs application-specific processes in those containers. The containers are overseen by node managers running on cluster nodes, which ensure that the application does not use more

resources than it has been allocated.

Description of YARN ResourceManager (RM):The ResourceManager has two main components: Scheduler and ApplicationsManager.

The Scheduler is responsible for allocating resources to the various running applications subject to familiar constraints of capacities, queues etc. The Scheduler is pure scheduler in the sense that it performs no monitoring or tracking of status for the application. Also, it offers no guarantees about restarting failed tasks either due to application failure or hardware failures. The Scheduler performs its scheduling function based on the resource requirements of the applications

The ApplicationsManager is responsible for accepting job-submissions, negotiating the first container for executing the application specific ApplicationMaster and provides the service for restarting the ApplicationMaster container on failure.

No application or tasks can be launched if RM is unavailable.

Description of YARN NodeManager (NM):The NodeManager is the per-machine framework agent who is responsible for containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the ResourceManager/Scheduler.If a node manager fails, it will stop sending heartbeats to the resource manager, and the node manager will be removed from the resource manager’s pool of available nodes.

Description of YARN ApplicationMaster (AM): The ApplicationMaster one per application has the responsibility of negotiating appropriate resource containers from the Scheduler, tracking their status and monitoring for progress.An application master sends periodic heartbeats to the resource manager, and in the event of application master failure, the resource manager will detect the failure and start a new instance of the master running in a new container.

Description of YARN JobHistoryServer (JHS): The JobHistoryServer is used for retaining job history data.Job information is archived by the job history server to enable later interrogation by users if desired.It provides users with details of past job runs.

Source-Hadoop Definitive Guide

Q 3. What are the main advantages of adopting Hadoop as infrastructure for data science projects?

Ans 3. The main advantages of adopting Hadoop are-

1. Scalable

Hadoop is a platform that is highly scalable. This is largely because of its ability to store as well as distribute large data sets across plenty of servers. These servers can be inexpensive and can operate in parallel. And with each addition of servers one adds more processing power. Relational database management systems (RDBMS) that cannot scale in order to process huge amounts of data.

2. Cost effective

Hadoop comes across as a very cost-effective solution for businesses that need to store ever growing data. In the case of traditional relational database management systems, it becomes really difficult to scale to the degrees possible with Hadoop, just to process data. Many of the businesses would have to downsize data and further implement classifications based on assumptions of how certain data could be more valuable that the other.

In the process, raw data would have to be deleted. This basically serves short term priorities, and if a business happens to change its plans somewhere down the line, the complete set of raw data would be unavailable for later usage.Hadoop’s scale-out architecture with MapReduce programming, allows the storage and processing of data in a very affordable manner. It can also be used in later times. In fact, the cost savings are huge.

3. Flexible

Hadoop enables businesses to easily access new data sources and tap into different types of data (both structured and unstructured) to generate value from that data. This means businesses can use Hadoop to derive valuable business insights from data sources such as social media, email conversations. Hadoop can be used for a wide variety of purposes, such as log processing, recommendation systems, data warehousing, market campaign analysis and fraud detection.

4. Fast

Hadoop’s unique storage method is based on a distributed file system that basically ‘maps’ data wherever it is located on a cluster. The tools for data processing are often on the same servers where the data is located, resulting in much faster data processing. If you’re dealing with large volumes of unstructured data, Hadoop is able to efficiently process terabytes of data in just minutes, and petabytes in hours.

5. Fault Tolerance

A key advantage of using Hadoop is its Fault Tolerance. When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of failure, there is another copy available for use.

6 Parallel processing

One of the primary aspects of the working of MapReduce programming is that it divides tasks in a manner that allows their execution in parallel.Parallel processing allows multiple processors to take on these divided tasks, such that they run entire programs in less time.

source

<http://blogs.mindsmapped.com/bigdatahadoop/hadoop-advantages-and-disadvantages/>

http://www.tutorialspoint.com/articles/advantages-of-hadoop-mapreduce-programming

Q 4. When would you NOT use MapReduce or Hadoop for data science projects?

Ans 4. These are the reasons for not using Hadoop or Mapreduce

1)You Need Answers in short time

Hadoop is probably not the ideal solution if you need really fast access to data. The various SQL engines for

Hadoop have made big strides in the past year, and will likely continue to improve. That’s more out of

necessity and to enable the huge amount of existing business intelligence tools that speak SQL to gain access

into Hadoop.

2) Your Queries Are Complex and Require Extensive Optimization

Hadoop is great because it gives you a massively parallel cluster for low-cost Lintel servers (or Wintel servers in the case of Hortonworks’ distribution) and scads of cheap hard disk capacity. While the hardware and scalability is straightforward, getting the most out of Hadoop typically requires a hefty investment in the technical skills required to optimize queries. According to a paper written by Hortonworks and Teradata, the software-based optimizers that are included with traditional data warehouse platforms can often outperform Hadoop.

3) You Require Random, Interactive Access to Data

The pushback from the limitations of the batch-oriented MapReduce paradigm in early Hadoop led the community to improve SQL performance and boost its capability to serve interactive queries against random data. Products like Cloudera’s Impala and Hortonworks Stinger initiative to improve the Hive SQL engine have emerged and are making headway. While SQL on Hadoop is getting better, in most cases it’s not a reason in of itself to adopt Hadoop.

4) You Want to Store Sensitive Data

Hadoop is evolving quickly and is able to do a lot of things that it couldn’t do just a few years ago. But one of the things that it’s not particularly good at today is storing sensitive data. Hadoop today has basic data and use access security. And while these features are improving by the month, the risks of accidentally losing personally identifiable information due to Hadoop’s less-than-stellar security capabilities is probably not worth the risk.

5) You Want to Replace Your Data Warehouse

A lot has been said about how Hadoop is decimating the market for traditional data warehouse platforms. And while there may be a grain of truth to that–it appears that Teradata customers are putting off upgrades until they can figure out this Hadoop thing–most data pros will tell you that Hadoop is complementary to a traditional data warehouse, not a replacement for it. The superior economics of Hadoop-based storage make it an excellent place to land raw data and pre-process it before siphoning it over to a traditional data warehouse to run analytic workloads.

Source- <http://www.datanami.com/2014/01/27/when_to_hadoop_and_when_not_to/>

Q 5. Explain in your own words in no more than half a page the workflow of running a MapReduce job

Ans 5. A MapReduce job is a unit of work that the client wants to be performed: it consists of the input data, the MapReduce program, and configuration information. Hadoop runs the job by dividing it into tasks, of which there are two types: map tasks and reduce tasks.

Map phase

-Process key-value input pairs, output key-value output pairs.Sort and group key-value pairs by key before sending the output to the Reduce phase .Each map task outputs an intermediate file on the local hard drive. Hadoop does its best to run the map task on a node where the input data resides in HDFS. This is called the data locality optimization because it doesn’t use valuable cluster bandwidth.

Reduce phase

- Process key-value input pairs, output key-value output pairs.Specify processing to perform on data collected from distributed map nodes .Data is received as input is grouped by key .Each reduce task creates an output file in HDFS and writes its output to it . The programmer decides the type of these Key-Value Pairs. Reduce tasks don’t have the advantage of data locality; the input to a single reduce task is normally the output from all mappers

There are two types of nodes that control the job execution process: a jobtracker and a number of tasktrackers. The jobtracker coordinates all the jobs run on the system by scheduling tasks to run on tasktrackers. Tasktrackers run tasks and send progress reports to the jobtracker, which keeps a record of the overall progress of each job. If a task fails, the jobtracker can reschedule it on a different tasktracker.Hadoop divides the input to a MapReduce job into fixed-size pieces called input splits, or just splits. Hadoop creates one map task for each split, which runs the user defined map function for each record in the split. When there are multiple reducers, the map tasks partition their output, each creating one partition for each reduce task. There can be many keys (and their associated values) in each partition, but the records for any given key are all in a single partition. Finally The output of the reduce is normally stored in HDFS for reliability

Source- Hadoop Definitive Guide