Assignment 4
Hadoop: Distributed Processing of Big Data

Instructor: Venkat Mavram
Course Number: 30088.(027)
Student: Sheetal Gangakhedkar **Date:** 03/27/2017

What is Hadoop? Explain briefly Hadoop core components. Why do we need Hadoop?
Hadoop is an Open Source distributed computation model and a distributed file system, based on Google MapReduce and GFS. It is designed to handle processing of large amounts of unstructured and semi-structured data. Hadoop has two main components, the HDFS which is the Hadoop File System and the MapReduce the distributed computation model. Hadoop is best suited for batch jobs (non real-time), that have to deal with terabytes to petabytes of Big Data, and applicable to almost all industries that are embarked on digital transformation, and want to generate actionable results for their business. Large files are stored in HDFS in blocks of 64MB or 128MB, and are replicated on three different nodes for high availability and fault tolerance. The MapReduce compute logic is called a job, these jobs are executed nearest to the node on which the application data blocks are found.

Explain briefly the various Hadoop daemons and their roles in a Hadoop cluster.
Hadoop uses is a master/slave architecture for distributed computing and distributed storage. Hadoop cluster has NameNode daemon, that is the master server that manages the metadata about the files and its distributed blocks, and provides access to the application or client. The 2.x version supports a secondary NameNode daemon, but it is in passive state, in case the master Namenode fails, the secondary Namenode takes over. File blocks (64MB in v1.0 and 128MB in v2.x of Hadoop) are distributed across DataNodes daemons, usually replicated on 3 DataNodes. DataNodes daemon's job is to store and serve these file blocks when requested by the client application, transparently. Data block distribution is usually 2 blocks on one rack, and 1 block on a different rack. JobTracker process runs on the Namenode server, that manages job submission by the client and task delegation to the TaskTrackers running on the DataNode where the data is. The TaskTracker runs on the same machine as the DataNode (move-compute-near-data), as instructed by the JobTracker. In case of v1.0 of Hadoop the Secondary NameNode in passive state is used for ensuring high-availability, but in v2.x Hadoop allows multiple NameNode daemons, but scoped to a namespace by HDFS Federation concept.

Hadoop v2.x added two additional daemons, one is called the "Resource Manager", the other is called "Node Manager".

Resource Manager daemon runs on the master server/Namenode server, it is responsible for the job getting submitted by the client, and scheduling it on the cluster. It also monitors and reserves the needed resources for the running job on the cluster from start to end (success/failure). Resource Manager uses two other processes on the master server, named Application Manager and Scheduler for MapReduce task and resource management.

Node Manager daemon runs on the slave node, and coordinates with Resource Manager task scheduling, resource utilization tracking on the slave node. Node Manager uses other processes like Application Master and Container for MapReduce task scheduling and execution on the slave node.

What is speculative execution in Hadoop?
Map tasks are distributed across nodes, but if some nodes are nowhere near completing their task and others are nearing completion, Hadoop schedules the slow node tasks on the the faster nodes that are available, in a way duplicating the tasks. Once one of the task replicated across nodes completes, Hadoop instructs the rest to stop their processing, and chooses to use the Map output from this completed task to pass it to the Reducer. This method of scheduling some slow tasks, across faster and freedup nodes is called “speculative execution”.

What is a Mapper, Combiner, Partitioner and Reducer?
Mapper: The first stage of MapReduce job is the Mapper task, this is initiated on the the partitioned data blocks, carried on DataNodes by the TaskTracker. The output of a Mapper is a block of tuples or <key, value> pairs. Mapper works on data blocks from HDFS.
Combiner: MapReduce comes with a default combiner, it combines the tuples or <key, value> pairs into <key, [value]+> for the tuples in that input partitioned data block.
Partitioner: MapReduce comes with a default partitioner, it partitions the mapper output, blocks of tuples by hash of the “key” it sees, across the Reducer tasks. A user can overwrite the default partitioner so the routing of the mapper output tuples is done based on the custom partitioner logic to the appropriate Reducer task, that can Reduce for that group of data.
Reducer: This is the last stage of the MapReduce job, it aggregates the Mapper output and reduces the output by a user defined aggregation logic defined in the Reducer. Reducer outputs to HDFS.

When do you use file cache in Hadoop?
File Cache is used to do lookups to map a value in the <name, value> pairs outputted by the mapper or reducer. This data is maintained in the in-memory FileCache and can be used across multiple Map and Reduce tasks, without triggering a reload. For example a “value” representing SSNs can be transformed to actual names, using a FileCache lookup.

What is the main use of Sqoop in Hadoop?
Sqoop is used to import and/or export data from traditional relational DB into/out-of the Hadoop system and participate in the Hadoop MapReduce tasks pipeline to accomplish a job. It simplifies the ETL tasks between a data warehouse and Hadoop data processing system.

What is Hive? What is the Hive Metastore?
Hive is a columnar NoSQL data store, with HiveQL as a SQL like query language on this data store. Each Hive query maps to a MapReduce job, thus providing a well known SQL interface on a distributed data store. Hive MetaStore is the meta-data store, that includes schema of the Hive data tables and the partition meta-data of the tables, this increases flexibility in schema design. It is well suited for storing the Views, so visualization softwares that familiar with SQL or JDBC interfaces can work with Hive data store. Also the Hive Metastore uses a relational data-base to store this critical meta-data about the Hive data tables.

Compare and contrast Hive with Relational Databases?
Hive is a columnar NoSQL Database, and supports HiveQL, which is a SQL like query language. This store is suited for distributed storage, but does not support ACID transactions like the relational DBs. Relational DB the schema is defined and embedded with the table when written, while Hive NoSQL DB is schema-free, the schema can be defined after the tables are created, and is stored externally, in this case in the Hive Metastore.

What is Spark? What is the advantage of using Spark along with Hadoop?
Spark is a in-memory compute engine, that includes support for MapReduce kind of jobs, it also has other components that are supported on this Spark Compute framework, they are GraphX, MLib, Spark Streaming, Spark SQL. Spark can access many data sources, including Hadoop HDFS, and many other NoSQL DBs. Spark can run on Hadoop, Mesos or standalone, those are its modes of operation. Because it is in-memory, and there is very little I/O as compared to Hadoop I/O intense HDFS access. Spark is more suited for real-time Big Data analytics and stream processing, while Hadoop is more suited for batch/non-real-time jobs.

Define Spark RDD. What operations does the Spark RDD support?
RDD stands for Resilient Distributed Dataset, Spark operates on RDDs. A Spark job takes as input RDDs, and outputs transformed RDDs. RDD is the fundamental data structure of Spark, it is a collection of objects required or created by the Spark jobs. Spark achieves speed by ensuring the collection of objects needed for a compute job are in that RDD, and Spark ensures the RDD is available in-memory. RDDs are used for intermediate results that can be shared between multiple Map Reduce tasks, as long as the Spark job is running.