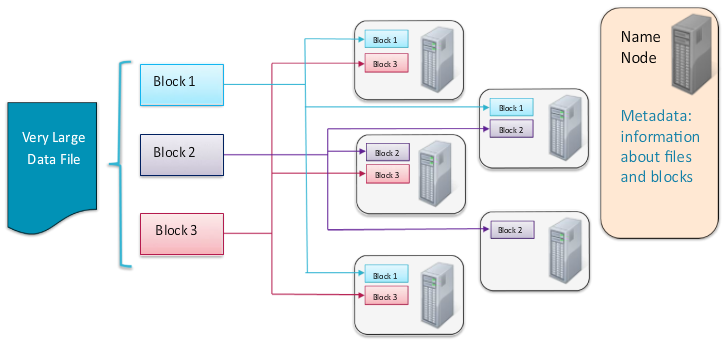
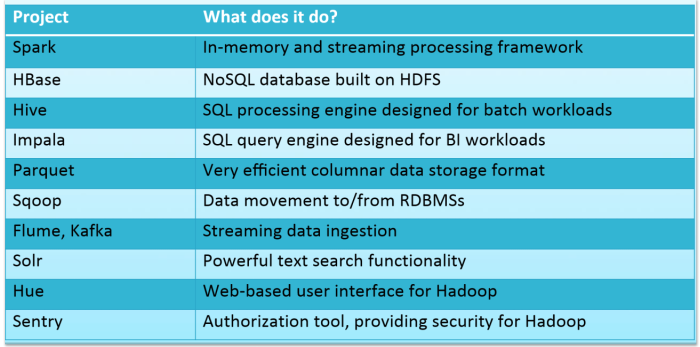
**Hadoop**

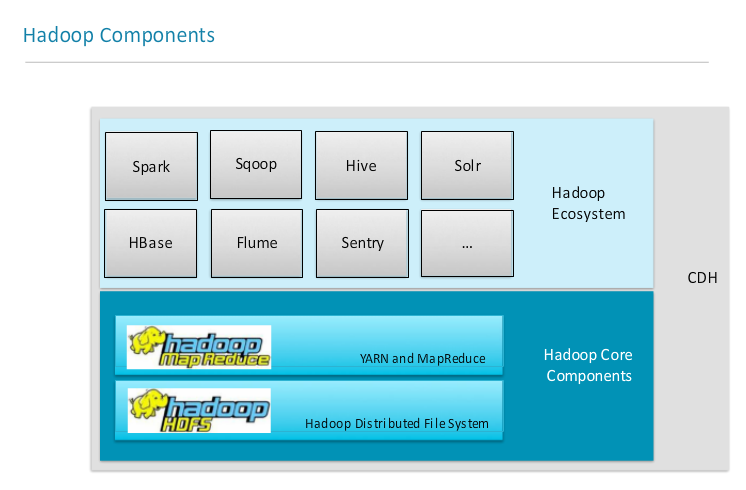
* Hadoop is a distributed data storage and processing platform.
* Core Hadoop is a File system and processing framework
* HDFS – Hadoop Distributed File system – stores data on the cluster
  + Any type of file can be stored
  + Data is split into chunks and replicated as it is written
  + Files in HDFS are “write-only”
    - Appends are permitted
    - No random writes are allowed
* Basic concept of HDFS
  + HDFS is a filesystem written in Java
  + Sits on top of a native filesystem
  + Supports efficient processing with MapReduce, Spark and other frameworks
  + Data files are split into blocks and distributed to data nodes
  + Each block is replicated on multiple nodes
  + NameNode stores the metadata info about the files and blocks

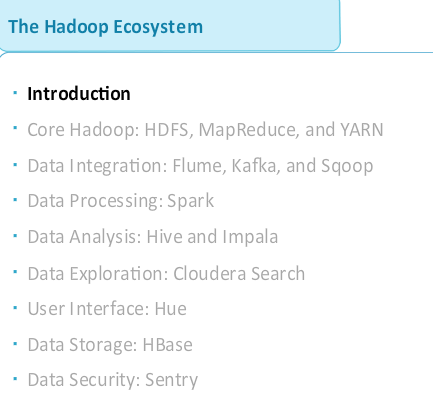


* MapReduce – process data on the cluster
  + A distributed processing framework / programming model
  + Neither platform – nor language – specific
  + Record oriented data processing (key and value)
* YARN – Yet Another Resource Negotiator – Schedules work on cluster
  + Manages the processing resources of the Hadoop cluster
  + Schedules jobs
  + Runs processing frameworks
* Motivation for YARN
  + Compatible with the application written in MapReduce
  + Can write user defined application in YARN (with client and master)
  + Multiple frameworks may exist on a single cluster
  + Each framework competes for compute and memory resources on the nodes
  + YARN allocates resources to different frameworks based on demand and on system admin settings
  + Supports framework like TEZ and SPARK
* Adding node adds capacity. Node failure is inevitable
  + System continues to function
  + Master re-assigns work to different node
  + Data replication happens
  + Nodes which recovers rejoins the cluster automatically
* Hadoop ecosystem – simplifies distributed computing so programmers can focus on the application

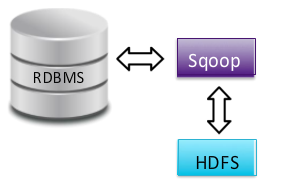
Example



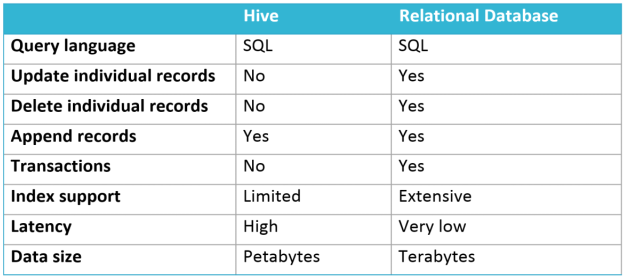




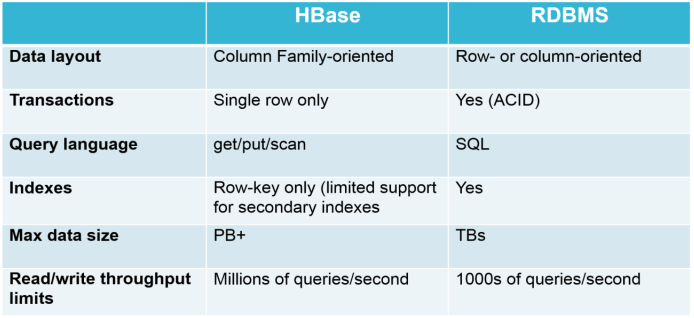
* Flume and Kafka – tools for ingesting event data into Hadoop as the data is being generated (eg. Files, logs, sysout)
  + Flume is typically easier to configure but Kafka provides more functionality
  + Flume generally provides a path from a data source to HDFS or to a streaming framework like Spark
  + Kafka uses a “Publish/Subscribe” model - allows data to be consumed by diff systems, including writing to HDFS
* Sqoop – rapidly moves large amount data from RDBMS to HDFS

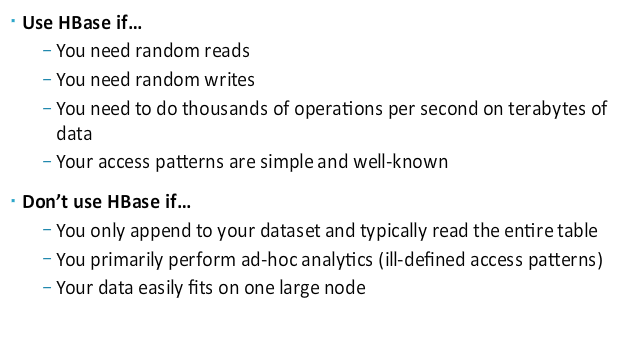


* + Uses JDBC to connect to the database
* Spark – large scale data processing engine
  + Provides the ability to process data as they are being generated
  + Faster than MapReduce
  + Code can be written using Java, Python or Scala
  + Well suited to iterative processing algorithms
* Hive – Abstraction layer on top of Hadoop
  + Uses SQL language called HiveQL
  + Uses MapReduce or Spark to process data
  + JDBC and ODBC connection are available
  + Data can be loaded before the table is defined (Do not need to know the data’s structure prior to loading it)
  + Suited for structured data



* Impala – high performance SQL engine
  + Supports a dialect of SQL similar to Hive
* Impala or Hive
  + Impala is best suited for ad-hoc analytics and situations where multiple people querying the cluster simultaneously
  + Hive is suited for batch processing
* Couldera Search – provides interactive full-text search for data in Hadoop cluster
  + Supports real time and batch intdexing
  + Dynamic web-based dashboard Search interface with Hue
  + Allows non-technical users to access data
* Hue – provides web front-end to a Hadoop
  + Upload, Browse data, Query tables in Hive and Impala, Search, etc.
  + Provides access control to users before accessing the data
* HBase – NoSQL distributed database
  + Stores data into HDFS

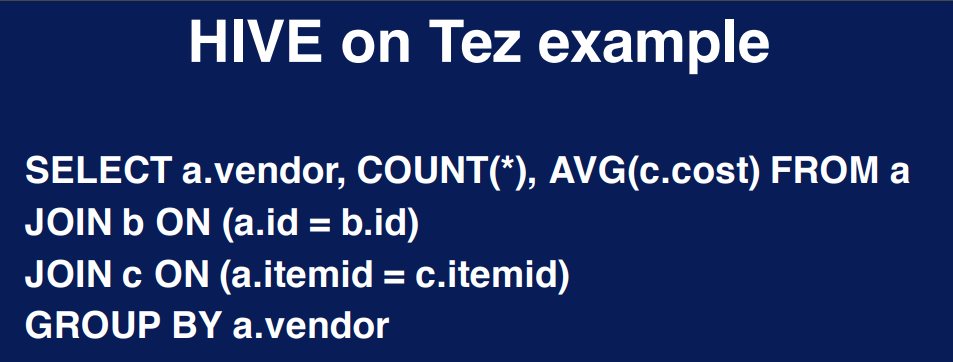


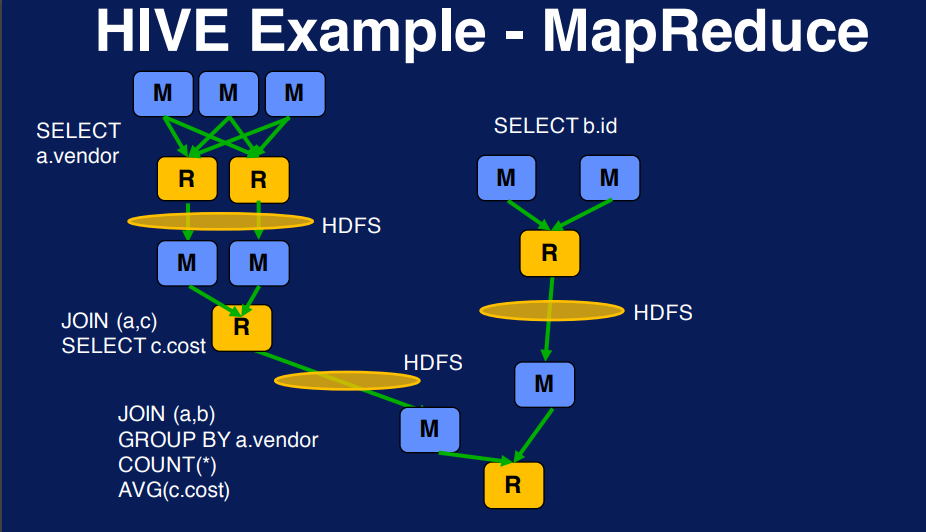


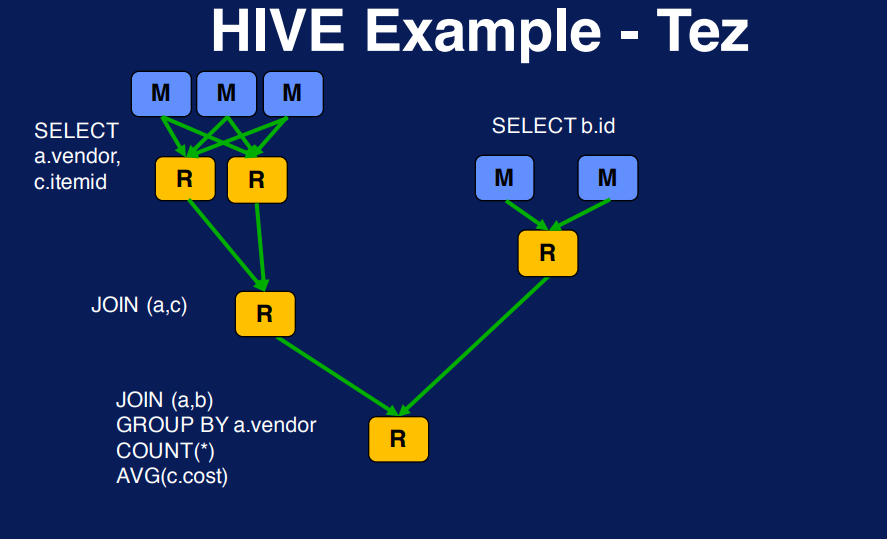
* Sentry – provides fine grained access control(authorization) to Hadoop ecosystem components (Hive, Impala)
  + Combined with Kerberos authentication, Sentry authorization provides a complete cluster security solution

Hadoop Architecture

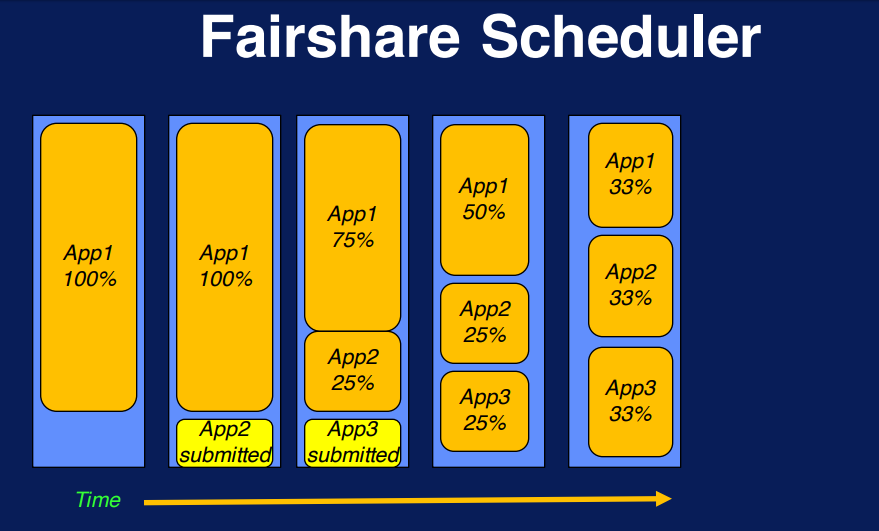
* Hadoop Cluster – is a group of computers working together
* Node – is an individual computer in the cluster
* Daemon – is a program running on a node
* Master Node – manage the work
  + Daemons running on master node ensures that the entire cluster works
  + A failed daemon could cause the entire cluster to become unstable
* Worker Node – do the work
  + Daemons running on worker node handle actual data processing
* TEZ
  + Resides on top of YARN
  + Handles dataflow graphs using API
  + Allows to customize application logic and data formats(no restriction like key-value pair as in MapReduce)
  + Runs complex DAG and dynamic DAG caching



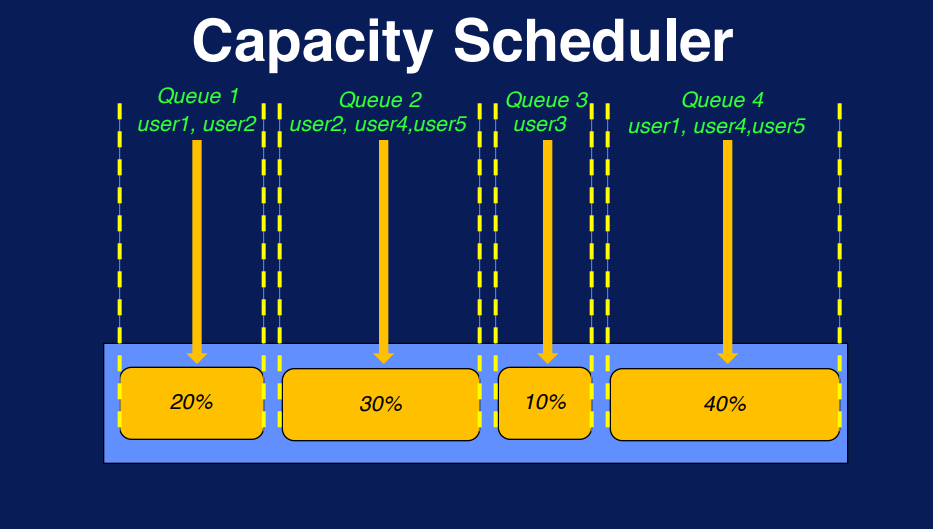




* SPARK
  + Advanced DAG execution engine
  + Supports cyclic dataflow
  + Keeps track of data – In memory computing
  + Supports JAVA, Scala, Python
  + Contains rich existing optimized libraries for handing machine learning, streaming applications and intensive data processing
* Hadoop Resource Scheduling
  + Motivation for Schedulers
    - Numerous execution engines
    - Control of resources between various components
  + Types of schedulers
    - First in First out(FIFO) – Default scheduler
    - Fairshare – balances resource allocation across application over time
      * By default it is based on memory but can add CPU as a resource
      * Guarantees minimum share



* + - Capacity – guarantees capacity using queues with fractional capacity and no of users. Restricts users from looking at and changing other users’ jobs
      * Queues and Sub Queues
      * Capacity with elasticity
      * Runtime changes and draining of apps
      * Resource based scheduling



PIG

* Two Components
  + Pig Latin – high level language
  + Infrastructure layer – processes which are written in Pig Latin and transforms it to MapReduce jobs or Tez
* Pig execution engine – can be local or MapReduce or Tez
* Various in built operators and functions are available
* Also extensible to write custom functions
* Interacts using Grunt command

Hive

* Apache dataware software
* Query and manages data using HiveQL – SQL like language
* Execution environment – MapReduce, Tez, Spark
* Runs interactively using beeline command

HBase

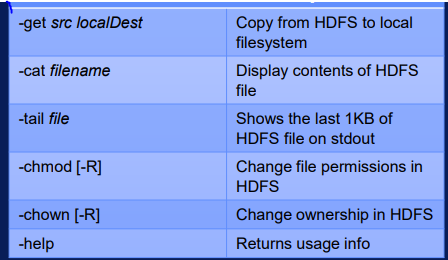
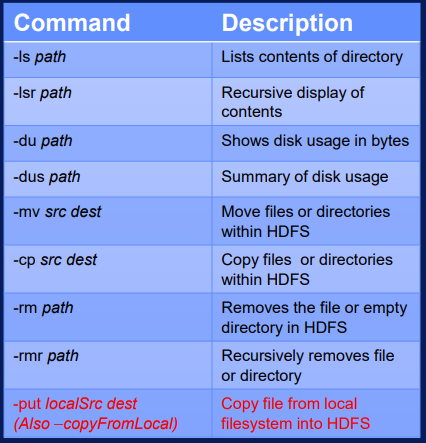
* Scalable data store and non-rational distributed database
* Runs on top of HDFS
* Caching mechanisms –
  + MemStore
  + BlockCache
* Uses HBase shell for interacting

HDFS Configuration parameters

* Replication parameter – default is 3
* Block size – default is 64MB and goes up to 128MB

HDFS Commands

* List files in HDFS – ***hdfs dfs –ls /***
* Make directory – hdfs dfs –mkdir /user/test
* Copy local file into HDFS – hdfs dfs –put sample.txt /user/test
* Details of a file – hdfs fsck /user/test/sample.txt



To be noted:

* Difference between HDFS and HDFS2
* Difference between MapReduce and YARN
* Diffrence between Pig and Hive
* Disadvantage of MpReduce over YARN
  + When we want to perform interactive data exploration – loading data from the disk every time you are trying to look at it.
  + When we want to perform iterative processing – this happens in some machine learning programs
* YARN supports complex DAG (directed acyclic graph)
* Also in memory caching of data
* Week -2: Pig, Hive and HBase working sessions
* Week – 3: Changes in Configuration files of HDFS using console
* Week -4: Python word count assignment

HBase

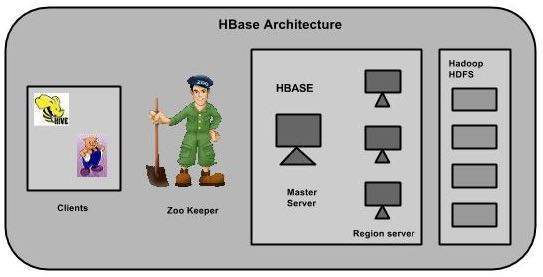
* provide quick random access to huge amounts of structured data similar to Google’s big table.
* a distributed column-oriented database built on top of the Hadoop file system

|  |  |
| --- | --- |
| **HDFS** | **HBase** |
| HDFS is a distributed file system suitable for storing large files. | HBase is a database built on top of the HDFS. |
| HDFS does not support fast individual record lookups. | HBase provides fast lookups for larger tables. |
| It provides high latency batch processing; no concept of batch processing. | It provides low latency access to single rows from billions of records (Random access). |
| It provides only sequential access of data. | HBase internally uses Hash tables and provides random access, and it stores the data in indexed HDFS files for faster lookups. |

* Storage Mechanism in HBase
  + HBase is a **column-oriented database** and the tables in it are sorted by row.
  + The table schema defines only column families, which are the key value pairs.
  + A table have multiple column families and each column family can have any number of columns
* HBase and RDBMS

|  |  |
| --- | --- |
| **HBase** | **RDBMS** |
| HBase is schema-less, it doesn't have the concept of fixed columns schema; defines only column families. | An RDBMS is governed by its schema, which describes the whole structure of tables. |
| It is built for wide tables. HBase is horizontally scalable. | It is thin and built for small tables. Hard to scale. |
| No transactions are there in HBase. | RDBMS is transactional. |
| It has de-normalized data. | It will have normalized data. |
| It is good for semi-structured as well as structured data. | It is good for structured data. |

Architecture



* The master server Assigns regions to the region servers and takes the help of Apache ZooKeeper for this task.
* Regions are nothing but tables that are split up and spread across the region servers.
* The store contains memory store and HFiles. Memstore is just like a cache memory. Anything that is entered into the HBase is stored here initially. Later, the data is transferred and saved in Hfiles as blocks and the memstore is flushed.