# Table of Content

[Table of Content 1](#_Toc7107615)

[WHY HBASE? 3](#_Toc7107616)

[WHY NOT HDFS? 3](#_Toc7107617)

[What is Flume? 4](#_Toc7107618)

[What is Kafka? 5](#_Toc7107619)

[What is Spark? 6](#_Toc7107620)

[Types of Spark deployment 7](#_Toc7107621)

[How will we have decided number of executers in Spark/Resource allocation in Spark? 7](#_Toc7107622)

[How Static and Dynamic resource allocation work? 8](#_Toc7107623)

[What is Sqoop? 9](#_Toc7107624)

[Scheduling 9](#_Toc7107625)

[Direct Import 9](#_Toc7107626)

[YARN vs Mesos 10](#_Toc7107627)

[YARN: 10](#_Toc7107628)

[MESOS: 10](#_Toc7107629)

[Kafka vs Spark Streaming vs Nifi vs Flink vs Storm 12](#_Toc7107630)

[Hive vs TEZ vs Impala vs Spark SQL 14](#_Toc7107631)

[Types of files formats in Hadoop 16](#_Toc7107632)

[Pig Vs Hive 19](#_Toc7107633)

[Hive vs Impala 20](#_Toc7107634)

[HDFS vs Hbase 21](#_Toc7107635)

[Data Processing 22](#_Toc7107636)

[Memory Management 22](#_Toc7107637)

[Data Flow 22](#_Toc7107638)

[Command Line Interface (CLI) 22](#_Toc7107639)

[Support for Other Streaming Products 22](#_Toc7107640)

[Capacity planning 23](#_Toc7107641)

[CLOUD vs In-house Cluster deployment 24](#_Toc7107642)

[Few recent Issues Faced 26](#_Toc7107643)

[Backup and Disaster Recovery Options (BDR) 27](#_Toc7107644)

[Hadoop Upgrades Process 28](#_Toc7107645)

[Cloudera Upgrades 0 above 29](#_Toc7107646)

[Security Components to Study: 31](#_Toc7107647)

[Networking 31](#_Toc7107648)

[Linux 32](#_Toc7107649)

[Authentication 32](#_Toc7107650)

[Authorization 35](#_Toc7107651)

[**Ranger Service Failure Scenarios** 35](#_Toc7107652)

[**Sentry Service Failure Scenarios** 35](#_Toc7107653)

[Data Protection 36](#_Toc7107654)

[Difference between Trust store and Key StoreHow to setup the trust store and Keystore? **Wrong Answers** 36](#_Toc7107655)

[What are the Data Masking components in Hortonworks and Cloudera? 36](#_Toc7107656)

[What is your Understanding about OS Hardening, few examples about it? 37](#_Toc7107657)

[Have you integrated AD with ecosystem components? How do you setup Ad? 39](#_Toc7107658)

[Which SSL is better OpenSSL or CA Signed 40](#_Toc7107659)

[Hortonworks 41](#_Toc7107660)

[Cloudera 42](#_Toc7107661)

What is Hbase?

* 1. It is based on Googles Big Table. It is a completely NoSQL Datastore and it is built over HDFS FS.
  2. It can also use the MapReduce framework to retrieve/store data.
  3. It is column family oriented
  4. It’s a master/slave architecture and the components it contains are:
     1. HMaster Server (Master deamon) –
        1. it manages and maintains the cluster
        2. it also administers in creating the tables in Hbase
        3. it also handles the failover
        4. It requests changes from clients like schema or metadata change then Hmaster is responsible for changing it in all these region servers
     2. Zookeeper which client connects and then ZK redirects the request to Hmaster or region server
        1. It will track all the region servers in the cluster and it is going to be like a service registry
        2. It will have all the information about the different region servers and Hmaster in that particular cluster.
     3. Region Server which stores the data and retrieves it as per request
        1. Block Cache – frequently accessed files are loaded into cache for faster reads
        2. Mem Store – This is the write cache where data that is to be written is present in cache before writing it on Region server
        3. WAL – **Write ahead lock** this is a file where the data is written but it is not persisted on permanent storage (Temporary storage)
        4. Hfile – This is the final/actual file that store the sorted or actual Key:value in region server.

## WHY HBASE?

* + 1. It can be used where you want to process huge amount of data like TB/PB usually
    2. If you have less GB of files, Hbase will not yield you more value because the number of instances of nodes that you have deployed is completely high and the cost for maintaining Hbase will be extremely large
    3. THROUGHPUT is extremely high and reads are faster
    4. You can also do Random reads/writes in an optimized way

## WHY NOT HDFS?

* + 1. **HDFS** When you want to write something to the disk HDFS rewrites the whole file
       1. In Hadoop, you cannot update the data once written. You can only append.
    2. **HBASE** comparatively writes only the part of the file.
       1. Example: it is like a Datastore where you can modify the data inside Hbase Database. You can Read you can Write and you can update in Hbase.
       2. It is built over Hadoop in order overcomes the drawbacks of HDFS
       3. Hbase has multiple Hmasters unlike Hadoop which has two Namenodes. Once one Hmaster goes down the other Hmaster will take over the responsibility and we can have multiple Hmasters configured for High Availablity.
       4. In case of Hadoop we will be reading the whole file
       5. But in case of Hbase we will be reading data that will be in any part of the cluster node.

# What is Flume?

1. It is a distributed service for collecting and aggregating streaming data into HDFS
2. It is tightly coupled to HDFS
3. Newer version of Flume 1.9 supports Kafka, Hbase
4. They can collect application logs, sensor, machine, geo-location, social media data.
5. IN Flume it is a simple architecture
   1. Source
   2. Channel
      1. File Channel: where data is maintained in temporary storage before it is committed to HDFS permanently
      2. Memory Channel: Data is maintained in memory before writing on Disk
   3. Sink

# What is Kafka?

1. It is a distributed streaming platform
2. The streaming platform has three capabilities
   1. It Publishes and subscribes to a stream of records, similar to a message queue
   2. The records are stored in a fault-tolerant durable way.
   3. Processes streams of records as they occur.
3. It is generally used for two broad class applications
   1. Building real-time streaming data pipelines
   2. Building real-time streaming applications that transform or react to the stream of data
4. Kafka cluster stores record in a category called as topic
5. And every record has a key, value and a timestamp

# What is Spark?

1. It is the eventual successor to MapReduce for data processing.
2. It allows to process very large amounts of data, with high-level, easy-to-use APIs.
3. You can also run Spark on other computing frameworks, such as Mesos
4. MapReduce has well-known problems with processing iterative algorithms that require multiple passes over the same data.
5. MapReduce is also not suitable for interactive computational work.
6. Spark is designed to overcome both of these issues with MapReduce processing—it is designed for both fast interactive queries, as well as iterative algorithms.
7. Spark relies on an in-memory storage mode and efficient fault recovery.
8. **Using Spark you can do batch processing, near realtime processing, machine learning, sql, graphx**

**Cluster Manager**

1. [Standalone](https://spark.apache.org/docs/latest/spark-standalone.html) – a simple cluster manager included with Spark that makes it easy to set up a cluster.
2. [Apache Mesos](https://spark.apache.org/docs/latest/running-on-mesos.html) – a general cluster manager that can also run Hadoop MapReduce and service applications.
3. [Hadoop YARN](https://spark.apache.org/docs/latest/running-on-yarn.html) – the resource manager in Hadoop 2.
4. [Kubernetes](https://spark.apache.org/docs/latest/running-on-kubernetes.html) – an open-source system for automating deployment, scaling, and management of containerized applications.
5. Spark

**What are the modes in which you can run Spark**

* + 1. Client / Interactive Mode (Like Spark or Hive shell)
       1. It is preferred when you are doing development work
       2. The driver will be on the same node from where the job is submitted
       3. Debugging becomes easy
    2. Cluster Mode
       1. It is preferred when you are running spark in Production cluster or Post-Development
       2. The driver can be on any node including the node from which you have submitted the job

#### How to submit spark job

$ ./bin/spark-submit --class org.apache.spark.examples.SparkPi \

--master yarn \

--deploy-mode cluster \

--driver-memory 4g \

--executor-memory 2g \

--executor-cores 1 \

--queue thequeue \

examples/jars/spark-examples\*.jar \

10

### Types of Spark deployment

* + 1. Standalone
    2. YARN
       1. Client modes:
       2. Cluster modes: The driver can be on any node including the node from which you have submitted the job
    3. MESOS
    4. Kubernetes

### How will we have decided number of executers in Spark/Resource allocation in Spark?

* + 1. Lets say for example we have 6 nodes and each node has 16 cores and 64 GB RAM
       1. We will evaluate the number of cores we should allocate first to the executors
          1. As per cloudera and other organisations research its advised that any application with mode than 5 concurrent task will lead to a bad show so the optimal number of cores to be assigned to an executor is 5
          2. Now that we got our cores as per the above configurations we get upto 3 executors on one node, So with 6 nodes we get total number of executor as 18
          3. But we cannot configure 18 executors as one executor (Java process) will be for Application Master
          4. So final number of executors we have is 17 as we have given one executor for AM
          5. Now we will need to set memory for each executor as we have 3 executors and 63 GB of RAM remaining. Each executor get 21 GB of ram

However out of this small overhead memory for the JVM is required about 10% so we will remove that from the total memory

We are left with 18.9 (Spark will always round up the decimal value so the value becomes 19)

* + - * 1. Hence the final number of executors we have is 17, per executor core – 5, memory – 19

### How Static and Dynamic resource allocation work?

In Dynamic two most important things is enable dynamic allocation parameter configuration

And configure external shuffle service

* 1. Statestore/ checkpoint
  2. Shared Variables (Broadcast variable , Accumulators)

<http://site.clairvoyantsoft.com/understanding-resource-allocation-configurations-spark-application/>

# What is Sqoop?

It is used to import rdbms data from HDFS or hive. Also export vise versa

1. Types of Incremental load
   1. Last modified
      1. You have to provide a date column and the load is performed as per the date
   2. Append

## Scheduling

1. We can do scheduling of Sqoop through Sqoop jobs and you can configure that Sqoop job in Oozie
   1. What Sqoop job will do is based on your last modified date it will import the data when you are performing an incremental load.

### Direct Import

1. When you are doing bulk uploading/insert
2. Sqoop imports from RDBMS to HDFS
   1. Generally, it imports the data by its internal MapReduce APIs Internally
   2. But in Direct Import it calls the Native code of the Database

# YARN vs Mesos

## YARN:

* Yarn was built as a necessity to move the Hadoop mapreduce API to the next iteration and lifecycle
* It had to remove the resource manager out of that embedded framework (Job Tracker) into its own container management lifecycle model.
* **JOB REQUEST -** When a job comes in Yarn Resource Manager it evaluates all the available resources and places the job.
  + Here YARN decides where job should go and thus it is modelled as a monolithic Scheduler.
* **WHEN TO PREFER YARN -** When you want to evaluate how to manage your entire datacentre as whole
  + **When you have YARN which can safely manage Hadoop jobs**
  + **This means that YARN is not capable of managing entire Datacenter**

## MESOS:

* It was built to be a global resource manager for your entire data centre
* **JOB REQUEST -** When a job comes in to the Mesos Master, what Mesos does is it determines what the resources that are available and it makes offer back.
  + These offers can be accepted or rejected
  + This allows the frame work to decide what the best fit is for the job that needs to be run
    - The framework has the option to reject the offer and wait for another offer to come in
    - Once the framework accepts the job then Mesos Master places the job on that particular slave.
  + This model is very scalable as per googles whitepapers who has used it. Also, this is a non-monolithic scheduler
* **WHEN TO PREFER MESOS –** When you want to evaluate how to manage your entire datacentre as whole
  + **When you have Mesos to manage every single resource in your datacentre**
* **IF you want to use both then you will need to create a static partition so that a specified resource will be managed by YARN and Mesos.**
* **This is possible with Apache Myriad**
  + **In this it enables Mesos Myriad executor to launch and Manage YARN Node Managers**
  + **HOW IT WORKS:**
    - When a job comes in to YARN, it will send a request to Mesos.
    - Mesos will internally pass it on to Mesos Slaves and there is this Myriad executor that runs near the YARN node manager in the Mesos Slaves
    - What it does is it advertises to the YARN Node Manager How many resources it has available.
    - This actually makes YARN dynamic because it gives Resources to YARN Resource Manager to place resources where he sees fit
    - And so from Mesos Side if you want to add or remove resources from Yarn, it becomes easy to dynamically control your data centre.
  + **Benefit of Apache Myriad**
    - When you have production, operations managed globally by Mesos. You can have the people on the Data Analytics side running their jobs in any fashion that they see fit via YARN for Job placement.
    - This means YARN will be limited in the production environment, and from a global perspective if you need to take resources away Hadoop resiliency with the job placement will allow those jobs to be places elsewhere on the cluster
    - You can Kill instances of YARN and take back those resources to make them available to Mesos
    - This really is to the best of worlds to remove static partitioning that running two of these running independently in a data centre would create.
    - Overall benefit is Apache Myriad is going to enable you to deploy both technologies in your datacentre
      * Levrage Mesos as the resource manager for the datacentre as a whole
      * Levrage YARN to manage Hadoop jobs you may need to get them deployed faster where you don’t care about the accept and reject capabilities of Mesos for the Jobs
      * Where datalocality is your primary concern for Hadoop Data only

# Kafka vs Spark Streaming vs Nifi vs Flink vs Storm

There is nothing right or wrong answer it completely depends on your use case also these libraries are constantly changing and evolving.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Kafka | Spark Stream | Flume | Flink | Storm |
| USAGE | Most of the time when you are doing realtime streaming Kafka is used.  It can be optimized based on your use case by tweaking parameters for durability, availability, throughput, etc | It does processing in batch and micro batch and guarantees 5 millisecond sla.  In new version of spark we are getting continuous streaming | It is a simple log colletor. It has a source, channel, sink.  Channel is of two types file and memory. | It is mostly preferred for realtime streaming.  It does processing on per record bases as it comes. | It is for realtime streaming |
|  | It is a publish subscribe distributed system & it allows pull architecture. It’s the responsibility of consumer to fetch the data.  Kafka pushes the data into topic which can later be retried from consumers |  | It is a push architecture means if the velocity of data is high and consumer is not able to handle that data then overflow will happen. |  |  |
| Feature | Fault-tolerant, through-put, durability, latency, |  | No durability, and it is not a Datastore like kafka |  |  |
|  | **Backpressure:** when we have a use case we will need to study what is the purpose of my application and what is the need from the above feature and then we can configure one or two features that can be considered by kafka | We don’t have backpressure in Spark streaming |  |  |  |
| Windowing |  | Supports time based windowing |  | Supports time base, record based windowing |  |
|  | We also have k-stream if you want some filtering of the data.  We also have kafka connect which offers readymade connectors |  |  |  |  |

# Hive vs TEZ vs Impala vs Spark SQL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Hive | TEZ | Impala | Spark SQL |
| About | It is a batch processing system |  | It is used for having a realtime feel or interactive querying |  |
|  | In hive the query processing is in batch |  | Impala uses MPP architecture through which you can get quick response. It will not be low latency but it will be efficient than hive |  |
| USECASE | If you have a usecase where in you want to analyse complete data then Hive is a good choice |  | If you want to analyse only few columns and want to have realtime feel then impala is a good option. Impala internally uses HIVE Metastore |  |
| Execution model | MapReduce/ in new TEZ or spark |  | MPP architecture | In Memory / catalyst optimizer |
| Suitable for | Large Batches | Interactive Mode | Interactive Mode | Interactive & Medium size batches |

# Types of files formats in Hadoop

They are the key factors in Bigdata processing & query performance

* 1. Schema Evolution
  2. Compression and Splittability
  3. Data Processing
     1. Write Performance
     2. Partial Read
     3. Full Read

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| File Type | Behaviour | Read/write | Compression | Spllitable | Schema Evolution |
| Text (csv,tsv) | Each line is a record and terminated by \n character | Good write but slow reads | Do not support block compression. Compressing text file often comes at a read cost. If want to compress then file level compress format like bzip2 will work | Inherently splittable on \n line character | Limited support of schema evolution (new fields can be appended but old fields can never be deleted)  Since every record is a data index file so **there is no metadata** attached to it. |
| Sequence | Each record is stored as a key-value pair in a binary format. **Eventually it occupied less space.** | Good write performance than text files. Good read performance when reading full rows of data. | Block level Compression so you can compress contents of a file and save a good amount of space. Also, No extra codes or steps are required if you want to work with these files. | Sequence files are splittable. So you can split the files into segments and assign them into multiple tasks. | Limited support of schema evolution (new fields can be appended but old fields can never be deleted)  Since every record is a data index file so **there is no metadata** attached to it. |
| Avro | It is a file format plus a serialization & deserialization framework. It uses JSON for defining data types and serializes data in a compact binary format. | Average read /write performance.  If you are **dealing with I/O then Avro is not the best choice.** | Supports block level compression. | Avro files are splittable. | Mainly designed to support full schema evolution. Avro files store metadata with their data and thus support full schema evolution. Fields can be renamed, added, deleted, while old files can still be read with new schema. |
| RC  (Row columnar) | These are flat files consisting of key /value pairs, and it shares much similarity with sequence file. | **Good for faster reads** but writing an RC file requires more memory and computation than non-columnar file formats. | Provides significant block compression, **data can be compressed with high compression ratios** | RC files are splittable | **No Schema evolution support**. In order to add a column to our data we must write every pre-existing RC file. |
| ORC (optimized row columnar) | These are successors to RC file. (**better version of RC file**) | Good for faster reads but with a **compromise with write performance. (Better than RC file)** | Provides significant block compression, **data can be compressed with high compression ratios** (Better than RC files) | Files are splittable at stripe levels. | **No Schema evolution support.Hence schema cannot be updated/deleted or changed.** |
| Parquet | Similar to RC and ORC. Stores nested data structures in a flat columnar format. | Faster reads with slow writes. | Supports compression with Snappy algorithm/compression codec. | Parquet files are conditionally splittable | Support limited schema evolution.  (new fields can be appended but old fields can never be deleted) |

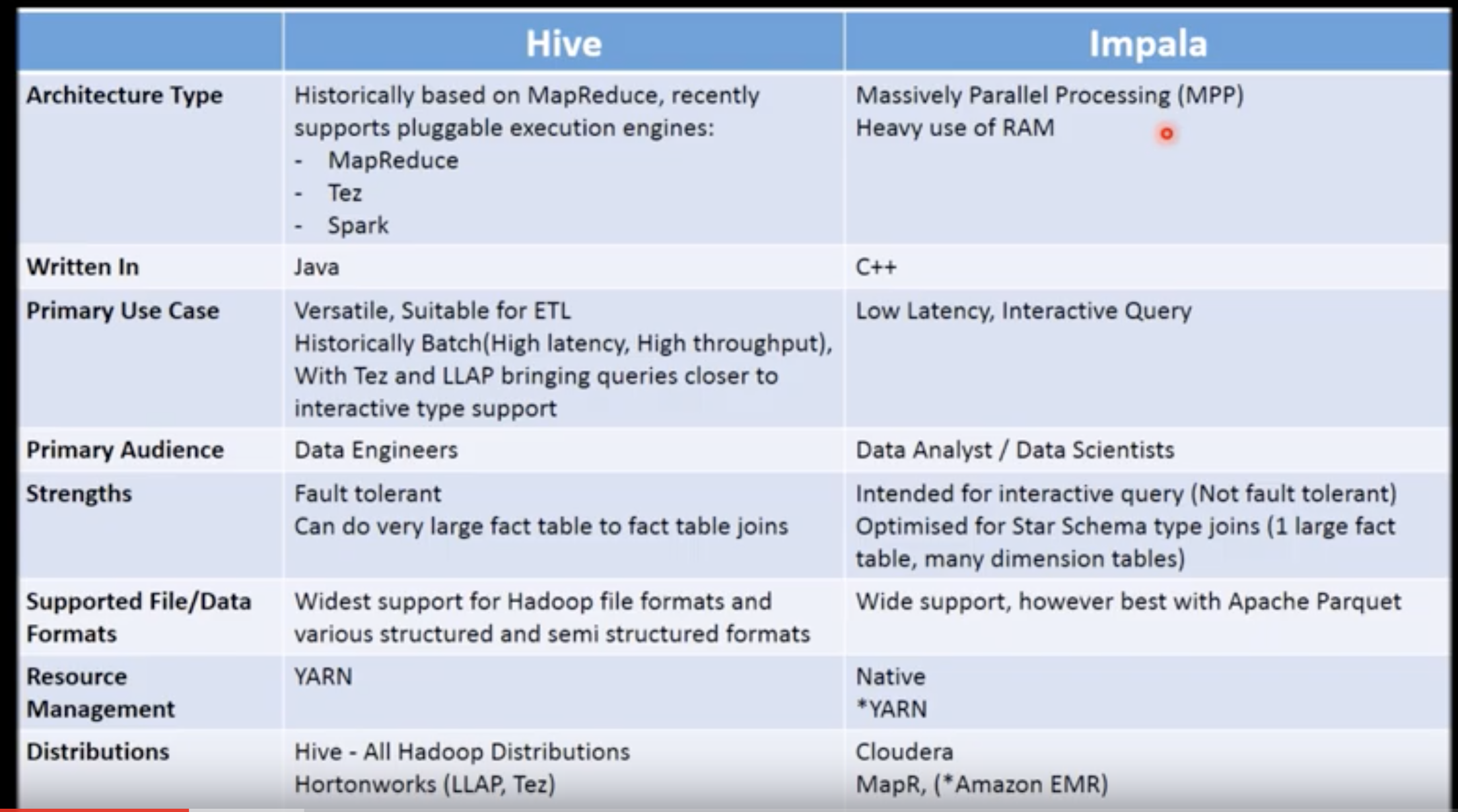
* Sometimes we may have committed to a column-based format (ORC or Parquet) but as we start getting into a project, the I/O pattern starts to shift towards a more write-heavy presence. In that case, it might be better to switch over to row-based storage (Avro)
* **ORC is commonly used with Apache Hive, and since Hive is essentially managed by engineers working for Hortonworks**. **Presto is also affiliated with ORC files**.
* Similarly, **Parquet is commonly used with Impala, and since Impala is a Cloudera project**, Parquet is also used in **Apache Drill, which is MapR‘s favoured SQL-on-Hadoop solution**; And **Apache Spark**,.
* **Avro, by comparison, is the file format often found in Apache Kafka clusters**. **Avro is also the favoured big data file format used by Druid**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| File Type | ABOUT THE FORMAT | Similarities | Read & Write | Usage / Preference | Comparisons | Schema Evolution |
| **TEXT (csv, tsv)** |  |  |  |  |  |  |
| **ORC** | N/A | N/A | n/A | n/A | n/A | n/A |
| **PARQUET** | It is a self-describing **Column based** storage format  **ADVANTAGE:**   1. It stores all column of same types 2. It achieves good level of compression hence saves a lot of storage space 3. When your downstream applications require only few columns then Parquet is a good choice. Example: average, standard deviations etc | 1. High performance because it uses compression 2. Due to compression, **storage benefit** 3. **Compute resources** are less required as data is compressed. 4. Computation and storage operations are much faster 5. Parquet and Avro has its information, metadata, schema embedded with the file 6. Parquet and Avro both support Schema Evolution to a certain degree | 1. Performs slower in write when compared to Avro 2. Columns-oriented datastores are optimized for read-heavy analytical workload | 1. Ideal choice for Analytical Querying 2. **It’s mostly suited for write once and read many times** 3. Faster Query operations on subset of Data 4. Highly recommended when using Impala due to its MPP engine | 1. Compression **are much better than Avro** 2. **It is Ideal when you are Querying few columns from an entire table** 3. Example like **SELECT few, columns from TABLE** | 1. Supports **SE** only in the case of Appends 2. Does not support changes to an existing column |
| **AVRO** | It is a self-describing **Row based** storage format  **ADVANTAGE:**   1. For ex we have 60 columns in a table: if we want to do queries like select \* from table then Avro is a good option | 1. Performs better in write operations when compared to Parquet 2. Row-based databases are best for write-heavy transactional workloads | 1. **More preferred** when we are looking **at ETL kind of work loads** 2. Slower queries on Data subset compared to Parquet | 1. **Compression are not better than** Parquet 2. **It is Ideal when you are Querying few columns from an entire table** 3. Example like **SELECT \* from TABLE** like queries | 1. Supports much richer and more featured schema evolution such as when you are appending or modifying columns avro supports full wealth of schema evolution |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

# Pig Vs Hive

|  |  |  |
| --- | --- | --- |
|  | PIG | HIVE |
| Type of Flow | Procedural Language | Declarative Language |
| Ease of use | Learning Curve | Less Learning Curve |
| Nature of Usage | When data needs Computing | Used in Analytical Area |
| Type of Data | Structured / Unstructured | Structured |
| Intermediate Layers | Represents data in Variables | Represents data in Tables |
| Debugging Facility | Code can be Debugged Locally | Complex and time consuming |
| Versatility | UDF is Easy | UDF is Complex |
| Maintenance | More Maintenance | Less Maintenance |
| Persistent | Variable values are not retained. Program will need to be run again to get the values | Tables will remain and can be accessed later |
| Development Time | May require more time due to Pig latin language needs to be learnt | Time will be less as hive being an SQL language |
| RDBMS Compatibility | Complex, since code syntax is completely different | Less needs to be modified |
| Handling Big Data | Efficiently handles more amounts of Data | Sometimes leads to memory overflow or impact in performance while handling large data. (There are parameters that can be adjusted to address this issue) |

# Hive vs Impala



# HDFS vs Hbase

|  |  |  |
| --- | --- | --- |
| Feature | HDFS | HBASE |
|  | When you have data in huge size and you want to dump that data then HDFS is good choice | When you have lot of records in small size Hbase is good choice. Later you can apply schema and read the data |
|  | Data in HDFS is immutable that means we cannot update or make change to the data. **It does not allows updation but only append** | Data in Hbase is mutable that means you can update or make changes to the data. **It allows updation of data** |
|  | It is a File system | It is a No-SQL Database. It is a key value pair & columnar database |
|  |  |  |

Sparks vs. Flink

* **Flink and Spark are in-memory databases that do not persist their data to storage**. **They can write their data to permanent storage, but the whole point of streaming is to keep it in memory, to analyze current data. All of this lets programmers write big data programs with streaming data**. They can take data in whatever format it is in, join different sets, reduce it to key-value pairs (map), and then run calculations on adjacent pairs to produce some final calculated value. They also can plug these data items into machine learning algorithms to make some projection (predictive models) or discover patterns (classification models).

Here's a comparison of Spark and Flink based on data processing, memory management, data flow, CLI, and support for other streaming products.

### Data Processing

**Spark processes data in micro batch mode** while **Flink processes streaming data in real time**. **Spark processes chunks of data, known as RDDs** while **Flink can process rows after rows of data in real time**. So, while a **minimum data latency is always there with Spark**, it is **not so with Flink**.

### Memory Management

Flink can automatically adapt to varied datasets but Spark needs to optimize and adjust its jobs manually to individual datasets. Also, Spark does manual partitioning and caching. So, expect some delay in processing. Flink has a different approach to memory management. Flink pages out to disk when memory is full, which is what happens with Windows and Linux too. Spark crashes that node when it runs out of memory. But it does not lose data since it is fault tolerant.

### Data Flow

**Flink is able to provide intermediate results on its data processing whenever required.** While **Spark follows a procedural programming system**, **Flink follows a distributed data flow approach**. So, whenever intermediate results are required, broadcast variables are used to distribute the pre-calculated results through to all the worker nodes.

### Command Line Interface (CLI)

Spark has CLIs in Scala, Python, and R. Flink does not really have a CLI, but the distinction is subtle.

To have a Spark CLI means users can **start up Spark, obtain a SparkContext, and write programs one line at a time. That makes walking through data and debugging easier**. Walking through data and running map and reduce processes, and doing that in stages, is how data scientists work.

Flink has a Scala CLI, too, but it is not exactly the same. **With Flink, you write code and then run print() to submit it in batch mode and wait for the output**.

### Support for Other Streaming Products

**Both Flink and Spark work with Kafka**, the streaming product written by LinkedIn. Flink also works with Storm topologies.

# Capacity planning

* 1. **Information gathering**
     1. We check what is the **volume of data to be processed**
     2. If there will be ingestion then at **what frequency and volume are we expecting the data**
     3. We check the **Retention policy for how long the customer want to store the data**
     4. Later we will try to **evaluate the type of data processing** that will be performed on the cluster

This will help you figure out the throughput speed you will need to efficiently process data

* + - 1. **CPU Intensive (Performing joints, queries, fact table joints etc)**
         1. If expected workload is CPU intensive then Disk speed and network speed are less important
         2. If you cluster is going to perform large amount of Heavy MR processing then network bandwidth becomes important factor

Maybe you’ll need to get multiple NICs for each node

* + - 1. **I/O intensive (Lot of data is going to be ingested and fetched)**
      2. Memory Intensive (Interactive or Realtime processing)
    1. What is the storage mechanism that we are going to prefer (Avro, Parquet, ORC)
       1. Based on the type of file format we will also achieve decent level of compression that will reduce storage cost
    2. What percent of data will be processed daily as it will again occupy space (Temporary data)
  1. **Based on these details we get the total storage for our HDFS**
  2. Once we obtain the size for HDFS we will also need to allocate around 20% storage for the FS on which the HDFS DN and other deamons are going to run (Server nodes)
  3. It is a benchmark that a node should not exceed more that 24 TB and a disk should be at max 2 TB in size, also we will take how many applications we are going to run and the type of workload on the cluster and then **this will help us to come up with the number of disks per node that we should go for.**
     1. **If we have IO intensive workload then we would require more disks per box**
     2. **If we have CPU intensive then we would require more memory and CPU perbox but less disks**
  4. Also depending on what type of processing is going to happen on the percentage of data we can get the cores that we would require on the nodes
     1. **If we have CPU intensive then we would require more memory and CPU**
     2. Also, we will need to evaluate the number of applications that we are going to deploy on OS and Hadoop
        1. We will need to check their requirements of CPU and Memory
        2. Once we get the evaluation of the memory and cores required by the applications and OS then we can determine the number of tasks we will be running on one node.
           1. For a heavy CPU intensive tasks requires 1 core
           2. Medium heavy CPU intensive tasks requires .70 cores per tasks
        3. Based on how many cores you are left with you can then determine the number of tasks you will be running on your cluster also it will help you to plan to take the cores accordingly
  5. After these calculations, you also obtain the remaining amount of memory that you can use to allocate for the type of processing that you want to perform on the cluster
  6. There is also a utility script for Yarn which advises the optimal amount of resource allocation to be done for Yarn components

# CLOUD vs In-house Cluster deployment

* Whenever we are considering between Cloud and In-house there are many things to consider
* The **First factor** is how important the **uptime is for your business**
* Cloud solutions are **more expensive than in-house**, but the **benefits of being in the cloud are less expensive for some businesses**
  + For example, an online business that is dependent on web-based transactions will consider uptime an extremely important factor
  + **therefore**, they will likely be willing to pay more for a cloud-based solution that can guarantee a certain level of uptime.
* Other businesses not as dependent on uptime may be more suited to an in-house set up.

|  |  |  |
| --- | --- | --- |
| In – House | PROS | CONS |
| Security | Critical data is in house and no third party can access it | Needs space in your office for a rack or server room/closet, in addition to dedicated IT support |
| Hardware | Gives you physical control over your backup | Requires a capital investment in hardware and infrastructure |
| Internet connectivity | No need to rely on an Internet connection for access to data. | There may be data loss disaster situations due to its in-house location if you don’t take data offsite. |
| Cost | It may be cost-effective for small to mid-sized companies | No uptime or recovery time guarantees |

|  |  |  |
| --- | --- | --- |
| Cloud | PROS | CONS |
| Cost | No need for onsite hardware or capital expenses. Well-suited to smaller companies that may outgrow storage too quickly. | The costs of the data recovery could outweigh the benefits for companies that are not as dependent on uptime and instant recovery. |
| Scalability | Storage can be added as needed. Solutions are often on-demand, so you only pay for what you need. | Every organization will have a limit to data that can be stored in the cloud due to storage availability and cost. |
| Backup | Backup and restore can be initiated from anywhere, using any computer, tablet, or smartphone. | If the Internet goes down on your side or on your cloud provider’s side, you won’t have access to any of your information. |
| Data Recovery | Data can be backed up in the cloud as regularly as 15-minute intervals, minimizing data losses in disaster situations. Small data set recovery time is improved. | Full data recovery could prove very time-consuming and impactful on systems. |

**CONCLUSION**

* **Third party cloud services provide a quick start for new businesses with lowest initial cost**
* **In the long run cost of running the business is much lower in having in-house cloud system**

# Few recent Issues Faced

1. **Dependency Hell issue faced while installing SSSD on Centos 6 server due to outdated libraries**
   1. **When did rpm install the dependencies where not resolved**
   2. **This took a lot of time to resolve the issue**
2. **Active directory integrated with Ambari but partial accounts where being retrieved**
   1. **This was caused due to SSSD used to create a cache and whenever it was unable to connect to few load balancers in AD then it used to use the built cache to perform authentication**
   2. **Resolution was when troubleshooted the connection to the few load balancers were blocked by firewall**
   3. **Rasied the issue and then resolved the issue with the networking team**
3. **Users from active directory where not properly resolving user groups when retrieving from AD when using SSSD**
   1. **This was because of the AD not preconfigured to work with Linux servers**
   2. **It required the AD admins to enable feature in AD for compatibility with Linux servers**
      1. **Enabling UID and GUID feature**

# Backup and Disaster Recovery Options (BDR)

* 1. Hortonworks
     1. Hortonworks Data Lifecycle Manager (DLM)
     2. Falcon

Falcon is deprecated post HDP 2.6

Wandisco (company)

Fusion

* 1. Cloudera
     1. HDFS Backup (Via Cloudera Manager Console > Backup option > create replication > HDFS Backup)
     2. Hive Backup (Via Cloudera Manager Console > Backup option > create replication > Hive Backup)

1. Cloud vs In-house Deployment options

# Hadoop Upgrades Process

1. Hortonworks
   * 1. Ambari Upgrade
        1. Minor Upgrade
        2. Major Upgrade
     2. **HDP Upgrade**
        1. **Express upgrade**
           1. Take backup of the databases managed by Hadoop components
           2. Take backup of namenode metadata by metasave command
           3. Login to ambari
           4. Run service checks and make sure all the services are running and resolve any warnings or issues ongoing
           5. Disable auto service restart feature in Ambari
           6. Register new HDP version using the ambari web UI
           7. Install new version of HDP using Ambari web UI
           8. Monitor the progress until installation is complete
           9. Perform the upgrade using the Ambari web UI click on “Upgrade”
           10. Click on Express Upgrade
           11. Click on the checkbox that says I have performed the manual steps

Also if you are using slider so please stop al the submitted application deployed using slider

* + - * 1. After upgrade the process will pause for sometime for running the service checks
        2. Once completed successfully you can click on finalize upgrade or downgrade if the upgrade fails

# Cloudera Upgrades 0 above

1. Upgrading from CDH 4 to CDH 5 Parcels
2. 1) Pre-Upgrade
3. Upgrade to Cloudera Manager 5 before upgrading to CDH 5
4. Ensure Java 1.7 is installed across the cluster.
5. Make sure No job is running and no user is using hadoop cluster
6. Run the Host Inspector and fix every issue.
7. If using security, run the Security Inspector.
8. Run hdfs fsck / and hdfs dfsadmin -report and fix every issue(if any).
9. Stop All Services
10. Stop the Cloudera Management Service
11. Perform Service-Specific Prerequisite Actions
12. Back up HDFS metadata on the NameNode (tar -cvf /root/nn\_backup\_data.tar /data/dfs/nn/\*)
13. Back up the Hive and Sqoop metastore databases. (Ref: https://www.cloudera.com/documentation/enterprise/5-6-x/topics/cm\_ag\_backup\_dbs.html#xd\_583c10bfdbd326ba--6eed2fb8-14349d04bee--7e98)
14. 2) Upgrade CM4 to CM5
15. Stop Cloudera Manager Server, Database, and Agent
16. sudo service cloudera-scm-server stop
17. sudo service cloudera-scm-server-db stop
18. sudo service cloudera-scm-agent hard\_stop\_confirmed
19. Upgrade the JDK on Cloudera Manager Server and Agent Hosts
20. Upgrade Cloudera Manager
21. Find the Cloudera repo file for your distribution by starting at https://archive.cloudera.com/cm5/ and navigating to the directory that matches your operating system.
22. Example:
23. for Red Hat or CentOS 6, you would go to "https://archive.cloudera.com/cm5/redhat/6/x86\_64/cm/" and download "cloudera-manager.repo"(For ubuntu it would be "cloudera.list").
24. Copy cloudera-manager.repo to /etc/yum.repos.d/.
25. Run the following commands:
26. $ sudo yum clean all
27. $ sudo yum upgrade cloudera-manager-server cloudera-manager-daemons cloudera-manager-server-db-2 cloudera-manager-agent
28. Start Cloudera Manager Server
29. sudo service cloudera-scm-server-db start
30. sudo service cloudera-scm-server start
31. Run the Upgrade Wizard
32. Log into the Cloudera Manager Admin console.
33. From the Home > Status tab, click next to the cluster name and select Upgrade Cluster. The Upgrade Wizard starts.
34. click the Use Parcels radio button.
35. The selected parcels are downloaded and distributed. Click Continue.
36. Click Continue. Cloudera Manager performs all service upgrades and restarts the cluster.
37. Click Finish to return to the Home > Status tab.
38. Done
39. Have you done any Performance Optimization of Hadoop Components?(OS performance tuning, Application specific like yarn, hive (ORC), Mapreduce, Spark)
    1. OS Performance tunning
       1. Disable THP – because THP feature is known to perform poorly in Hadoop cluster and results in excessively high CPU utilization
          1. Disable Host swappiness - Swap space is much slower than memory as it is backed by disk instead of RAM. Processes that are swapped to disk are likely to experience pauses, which may cause issues and missed SLAs
          2. Configure CPUs for Performance Scaling – from power saving change to performance mode for better performance (edit /sys/devices/system/cpu/cpu\*/cpufreq/scaling\_governor and set the content to 'performance')
    2. **YARN**
       1. Enabling RM HA will provide high availability for the resource manager inYARN which is critical for resource negotiations
       2. When you have multi tenant cluster and you want guaranteed resources you can configure capacity scheduler queues
       3. Enable log aggregation yarn.log-aggregation.enable (This will keep the Application logs got a longer time for us to access and troubleshoot)
          1. This may impact namenode memory as lots of logs will be maintained
    3. **Hive**
       1. Use TEZ
          1. set hive.execution.engine=tez; **( Tez execution engine will always perform better and give faster performance comparative to MR)**
          2. Using ORC file. **(Using ORC file in Hive extremely beneficial to get fast response times for your HIVE queries.)** 
             1. Also it supports compression of about 70 % which can save a lot of storage costs
    4. MapReduce
       1. Speculative Execution
       2. Implement a [combiner](https://data-flair.training/blogs/hadoop-combiner-tutorial/)to reduce data which enables faster data transfer.
       3. Avoid **RAID** on **TaskTracker** and datanode machines, it generally reduces performance.
       4. set **mapred.compress.map.output** to **true (It saves time by reducing the amount of disk IO during the shuffle)**
    5. Spark
       1. <https://community.hortonworks.com/articles/80301/spark-configuration-and-best-practice-advice.html>
    6. Sqoop
       1. You can import data into Hive directly by direct import in this instead of using sqoop api it will use native libraries provided by the vendors and this will improve performance

# Security Components to Study:

## Networking

* 1. **DNS** - The Domain Name Systems (DNS) is the phonebook of the Internet. Humans access information online through domain names, like nytimes.com or espn.com. Web browsers interact through Internet Protocol (IP) addresses. DN S translates domain names to IP addresses so browsers can load Internet resources.
     1. Each device connected to the Internet has a unique IP address which other machines use to find the device. DNS servers eliminate the need for humans to memorize IP addresses such as 192.168.1.1 (in IPv4), or more complex newer alphanumeric IP addresses such as 2400:cb00:2048:1::c629:d7a2 (in IPv6)
  2. **DHCP** - DHCP (Dynamic Host Configuration Protocol) is a network management protocol used to dynamically assign an Internet Protocol (IP) address to any device, or node, on a network so they can communicate using IP. DHCP automates and centrally manages these configurations rather than requiring network administrators to manually assign IP addresses to all network devices. DHCP can be implemented on small local networks as well as large enterprise networks.
  3. **Ulimit** - Ulimit is the number of open file descriptors per process. It is a method for restricting the number of various resources a process can consume. Sometimes you will get the error message is like “too many files open “, it is because you have reached the limits of opened files, so you need to increase the ulimit parameters
  4. **THP** - Huge Pages in Linux-based operating systems create pre-allocated contiguous memory space designed to assist application performance. Transparent Huge Pages (THP) is a Linux OS feature that conceals much of the complexity of using actual Huge Pages as well as automates the creation of contiguous memory space.
  5. **SELinux** - SELinux is an acronym for Security-enhanced Linux. It is a security feature of the Linux kernel. It is designed to protect the server against misconfigurations and/or compromised daemons. It put limits and instructs server daemons or programs what files they can access and what actions they can take by defining a security policy.
  6. **Firewall** - A firewall is a network security program that controls the incoming and outgoing connections based on the rules that are set. Linux has a default firewall I.e iptables. Using iptables an administrator can set the rules of the firewall.
  7. **Tunnelling** - Network Tunnels with Linux. ... Most people are familiar with tunnels as they are used to create virtual private networks (VPN) to connect remote sites, or remote hosts, to a central site securely, allowing the remote network to be accessed as if it is local.

## Linux

# Authentication

* 1. What is service ticket? How does client identify which service(NN,Hive etc) to be contacted for a particular request after getting TGT?
* When we type Hadoop fs -ls / for example the client always looks for the Hadoop site xml files from there it gets to know the Principal of Namenode and Datanode.
* It reads these principals and provide it to the KDC and then after verifying these principals the client gets the Service Ticket
* Using which the client gets an access to the service.
* Incase of Hive when we access it from beeline we explicitly mention the principal name when providing the beeline command.(in this case the client knows the Principal from beeline)
  1. Explain Kerberos Architecture

OR

* 1. Explain the Process of How a User gets access to a service i.e. HTTP in Kerberos

<http://www.roguelynn.com/words/explain-like-im-5-kerberos/>

Basically, Kerberos comes down to just that it is a:

* a protocol for authentication
* it uses tickets to authenticate
* it avoids storing passwords locally or sending them over the internet
* and involves a trusted 3rd-party
* it is built on symmetric-key cryptograph

**Points:**

* You have a ticket – your proof of identity encrypted with a secret key for the service requested – on your local machine
* Rather than re-entering your user/password credentials, your ticket (cached on your system) is used to authenticate allowing for single sign-on.
* Your ticket is refreshed when you sign on to your computer, or when you kinit USER within your terminal.
* it takes a third-party (a Key Distribution Center) to authenticate between a client and a service or host machine.

**Process:**

1. You introduce yourself to the Authentication Server. Logging into your computer, or kinit USERNAME, initiates that introduction via a plaintext request for a Ticket Granting Ticket (TGT)
   * message contains: (**Optional**)
   * your name/ID
   * the name/ID of the requested service (in this case, service is the Ticket Granting Server),
   * your network address (may be a list of IP addresses for multiple machines, or may be null if wanting to use on any machine), and
   * requested lifetime for the validity of the TGT,
   * The above details are sent to the Authentication Server
2. The Authentication Server will check if you are in the KDC database. This check is only to see if you exist; At this point no credentials are checked
3. If there are no errors (e.g. user is not found), it will randomly generate a key called a session key for use between you and the Ticket Granting Server (TGS).
4. The Authentication Server will then send two messages back to you. One message is **the TGT and is encrypted with the TGS Secret Key**.
   * The other message is encrypted with your Client Secret Key and contains:
     + the TGS name/ID,
     + timestamp,
     + lifetime, and
     + TGS Session Key
5. Your Client Secret Key is determined by prompting you for your password, appending a salt (made up of user@REALMNAME.COM) and hashing the whole thing. Now you can use it for decrypting the second message to obtain the TGS Session Key. If the password is incorrect, then you will not be able to decrypt the message. Please note that this is the step in which the password you enter is implicitly validated.
6. At this point, you have the TGT that you can not read because you do not have the TGS Secret Key to decrypt it. You do, however, have the TGS Session Key.
7. It’s now your turn to send two messages to the ticket Granting Server
   * Authenticator, encrypted with the TGS Session Key containing (your name/ID, and timestamp)
   * Encrypted Authenticator and TGT
8. The Ticket Granting Server will first check the KDC database to see if the HTTP Service exists
9. The TGS will then compare your details with Authenticator to that of the TGT
10. The Ticket Granting Server then randomly generates the HTTP Service Session Key, and prepares the HTTP Service ticket for you and encrypts it with the HTTP Service Secret Key
11. Then the TGS sends you two messages. One is the encrypted HTTP Service Ticket; the other that is encrypted with the TGS Session Key.
12. Your machine decrypts the message with the TGS Session Key and obtain the HTTP Service Session Key
13. To now access the HTTP Service, your machine prepares another Authenticator message that contains HTTP Service Session Key. Your machine then sends the Authenticator and the still-encrypted HTTP Service Ticket received from the TGS.
14. The HTTP Service then decrypts the Ticket with its Secret Key to obtain the HTTP Service Session Key. It then uses that Session Key to decrypt the Authenticator message you sent.
15. Similar to the TGS, the HTTP Server will compare your Authenticator details to that of the Ticket
16. The HTTP Service then sends an Authenticator message containing its ID and timestamp to confirm its identity to you and is encrypted with the HTTP Service Session Key
17. Your machine reads the Authenticator message by decrypting with the cached HTTP Service Session Key, and knows that it must receive a message with the HTTP Service’s ID and timestamp.
18. And now you have been authenticated to use the HTTP Service.

# Authorization

* 1. I've created some policies in Ranger and Ranger KMS (**Sentry**). What happen if my Ranger service is down? Will those policies still work? How?

**Ranger Service Failure Scenarios**

It stores the policies and information in the database but also in a JSON file

This file is located in “/etc/ranger/(hdfs repository)/policy.json”

**Sentry Service Failure Scenarios**

If the Sentry service fails and you attempt to access the Hive warehouse, Hive, Impala and HDFS will behave as follows:

* Hive: Queries to the Hive warehouse will fail with an authentication error.
* Impala: The Impala Catalog server caches Sentry privileges. If Sentry goes down, Impala queries will continue to work and will be authorized against this cached copy of the metadata. However, authorization DDLs such as CREATE ROLE or GRANT ROLE will fail.
* HDFS/Sentry Synchronized Permissions: Affected HDFS files will continue to use a cached copy of the synchronized ACLs for a configurable period, after which they will fall back to NameNode ACLs. The timeout value can be modified by adding the sentry.authorization-provider.cache-stale-threshold.ms parameter to the hdfs-site.xml Safety Valve in Cloudera Manager.

While the default timeout value is very conservative (60 seconds), if you're working with a large cluster, these timeouts can be increased to a much longer period, anywhere between a few minutes to a few hours, as required by the cluster if Sentry fails.

* Solr: Solr does not use the Sentry service, hence there will be no impact.

# **Data** Protection

What is the difference between Symmetric and Asymmetric Keys?

* Asymmetrical encryption is also known as public key cryptography, which is a relatively new method, compared to symmetric encryption.
* Asymmetric encryption uses two keys to encrypt a plain text. Secret keys are exchanged over the Internet or a large network. It ensures that malicious persons do not misuse the keys.
* Anyone with a secret key can decrypt the message and therefore asymmetrical encryption uses two related keys to boosting security.
* A public key is made freely available to anyone who might want to send you a message. The second private key is kept a secret so that you can only know.
* A message that is encrypted using a public key can only be decrypted using a private key, while also, a message encrypted using a private key can be decrypted using a public key.
* Asymmetric encryption is mostly used in day-to-day communication channels, especially over the Internet.
* Popular asymmetric key encryption algorithm are RSA, DSA, PKCS.

### Difference between Trust store and Key StoreHow to setup the trust store and Keystore? **Wrong Answers**

* Trust Store and Key Store are used in context of setting up SSL connection in Java application between client and server.
* They are used to store SSL certificates in Java but there is a minor difference between them.
* Trust store is used to store public certificates while keystore is used to store private certificates of client or server.
* In SSL, the purpose of **trust Store is to verify credentials** and purpose of**keyStore is to provide credential**.

### What are the Data Masking components in Hortonworks and Cloudera?

* **Ranger in Hortonworks**
  + We can achieve this using Resource based or Tag based policies in Hive.
* Sensitive Data Masking in Cloudera is an option in which things like Cloudera Manager and Passwords, Cloudera Manager Server Database Password, Cloudera Manager API, Log and Query,
* **Cloudera Navigator is a component for Masking the Data**
  + **Apache Record Service (Opensource in Beta Version not yet released)**

# What is your Understanding about OS Hardening, few examples about it?

Link : <https://www.tecmint.com/linux-server-hardening-security-tips/>

1. Configure the BIOS to disable booting from CD/DVD, External Devices, Floppy Drive in BIOS. Next, enable BIOSpassword & also protect GRUB with password to restrict physical access of your system.
2. It’s important to have different partitions to obtain higher data security in case if any disaster happens. By creating different partitions, data can be separated and grouped. When an unexpected accident occurs, only data of that partition will be damaged, while the data on other partitions survived. Make sure you must have following separate partitions and sure that third party applications should be installed on separate file systems under /opt.
3. avoid installing useless packages to avoid vulnerabilities in packages. This may minimize risk that compromise of one service may lead to compromise of other services. Find and remove or disable unwanted services from the server to minimize vulnerability. Use the ‘chkconfig‘ command to find out services which are running on runlevel 3
4. With the help of ‘netstat‘ networking command you can view all open ports and associated programs. Then you can use ‘chkconfig‘ command to disable all unwanted network services from the system.
5. Disable root Login
6. Only allow Specific Users
7. Keep System updated
8. Lockdown Cronjobs
9. Disable USB stick to Detect
10. Turn on SELinux
11. Turn Off IPv6
12. Restrict Users to Use Old Passwords
13. Enable Iptables (Firewall)
14. Display SSH Banner Before Login
15. Review Logs Regularly
16. Important file Backup
17. **How does Hadoop Namenode failover process works?**

The ZKFailoverController (ZKFC) is a new component which is a ZooKeeper client which also monitors and manages the state of the NameNode. Each of the machines which runs a NameNode also runs a ZKFC, and that ZKFC is responsible for:

**Health monitoring** - the ZKFC pings its local NameNode on a periodic basis with a health-check command. So long as the NameNode responds in a timely fashion with a healthy status, the ZKFC considers the node healthy. If the node has crashed, frozen, or otherwise entered an unhealthy state, the health monitor will mark it as unhealthy.

**ZooKeeper session management** - when the local NameNode is healthy, the ZKFC holds a session open in ZooKeeper. If the local NameNode is active, it also holds a special "lock" znode. This lock uses ZooKeeper's support for "ephemeral" nodes; if the session expires, the lock node will be automatically deleted.

**ZooKeeper-based election** - if the local NameNode is healthy, and the ZKFC sees that no other node currently holds the lock znode, it will itself try to acquire the lock. If it succeeds, then it has "won the election", and is responsible for running a failover to make its local NameNode active.

1. **What happens when two clients try to access the same file in the HDFS?**

HDFS supports exclusive writes only.

When the first client contacts the “NameNode” to open the file for writing, the “NameNode” grants a lease to the client to create this file. When the second client tries to open the same file for writing, the “NameNode” will notice that the lease for the file is already granted to another client, and will reject the open request for the second client.

1. What will you do when NameNode is down? Gave a wrong answer

* The NameNode recovery process involves the following steps to make the Hadoop cluster up and running:
* Use the file system metadata replica (FsImage) to start a new NameNode.
* Then, configure the DataNodes and clients so that they can acknowledge this new NameNode, that is started.
* Now the new NameNode will start serving the client after it has completed loading the last checkpoint FsImage (for metadata information) and received enough block reports from the DataNodes.
* Whereas, on large Hadoop clusters this NameNode recovery process may consume a lot of time and this becomes even a greater challenge in the case of the routine maintenance. Therefore, we have HDFS High Availability Architecture which is covered in the HA architecture blog.

# Have you integrated AD with ecosystem components? How do you setup Ad?

Steps

Create the AD machine

* + Click on Add role and features
* Select Active Directory Domain services
* Click Next and at last select restart automatically if restart required
  + In the alerts section click on Promote this server to a domain controller
* Click on Add a new forest
* Provide a password
* Click Next and complete the promotion
* Restart Server & Login back
  + In Local Server we can see that the Domain shows what we have set
  + Click on tools > Users and Computers
  + Create a certificate for secured communication with ldap
* Click on Add Roles and Features
* Role Based installation in Roles seclect Active Directory Certificate Services and click on Next Select Role Services Select Certificate Authority
* In Alerts click on Configure AD certificate services
* Specify the Administrator profile "Domain\useraccount"
* Click Next let it be enterprise CA
* Click Next Type of CA keep it Root CA
* Create a Private Key (SHA1)
* Verify the Cn and DN and specify years "99"
* Restart the server
  + On command line type "ldp"
* this will help you verify if the communication with ldap and ldaps are working

# Which SSL is better OpenSSL or CA Signed

* 1. Certificate Hierarchy / Chain of Trust
     1. Digital Certificate: the certificate that we create with the help of PKI and Certificate Authority are verified using chain of trust. **The trust anchor** for digital certificate hierarchy **is known as Root CA**. All the CA’s are grouped into multiple levels of CA hierarchy.
  2. Cross certification: when two Root CA’s exchange their certification and are from different domains.
  3. HTTP
     1. Hyper text transfer protocol is when we browser internet without sharing our personal details.
  4. HTTPS
     1. Hyper text transfer protocol secured is when we share our personal details over the internet and it is protected by encrypting/ unscrambling our data and then sending to the source. Preventing it from hackers to steal our information.
  5. SSL
     1. Secure socket layer is a protocol used to ensure security on internet. It uses public key encryption to secure data. How it works is when we browse internet our browser requests a certificate to the website to prove its identity. Once the webserver provides an ssl certificate the browser will then verify the identity and proceed with secure communication.
  6. TLS
     1. It is the latest industry standard cryptographic protocol. The successor to SSL and based on same specifications. Unlike SSL it also authenticates the server, client and encrypts the data.

NOTE: A lot of websites are now using HTTPS by default, regardless if sensitive data is going to be exchanged or not. Not only, **Google** is flagging websites as “not secure” if they are not SSL protected but also, Google is penalizing websites that are not SSL protected.



## Hortonworks

* 1. Administration
     1. Ranger
  2. Authentication
     1. Kerberos AD
     2. MIT KDC
     3. Knox
  3. Authorization
     1. Ranger
  4. Auditing
     1. Ranger
  5. Data Protection
     1. HDFS Encryption
     2. Ranger Transparent Data Encryption
  6. Data Governance
     1. Atlas

## Cloudera

* 1. Administration
     1. Sentry
  2. Authentication
     1. Kerberos AD
     2. MIT KDC
  3. Authorization
     1. Sentry
  4. Auditing
     1. Cloudera Navigator
  5. Data Protection
     1. HDFS Encryption
     2. Cloudera Navigator Encrypt
     3. Cloudera Navigator Key Trustee Server
  6. Data Governance
     1. Cloudera Navigator