Lecture 5: Distributed computing and EMR

Contents

Gittu George, March 4 2024

5.1. Announcements

- Tutorial 2 (by Daniel) is going to be ***extremely** useful for milestone 3. Apr 8th - Details will be posted by Daniel.

5.2. About mid-block feedback:

Thanks for the feedback you gave to the block reps. We will be using the feedback to improve the course.

- We will take care of windows users with but more care.
- Finally, please feel free to approach me during lab time (we have a lot of instructor time available) if you want to discuss anything covered.

5.3. Learning objectives

- Understanding when to opt for a scale-out solution.
- Details of Hadoop and its main components.
- Various Hadoop distributions and their uses.
- Details of EMR cluster architecture.

5.4. Refresher questions

- Let's clarify cloud systems and some pieces we have learned:
  - Username/password to log in to an AWS account.
  - Access key and secret key to use AWS services.
  - S3, EC2, and other services that I set up in my AWS account.

- Public and private keys to log in to an EC2 instance.

5.4.1. Theory

- What were the 5 categories of AWS services we discussed?
- What is an EC2 instance, and can you explain it?
- How do S3, EFS, and EBS differ from each other?
- Which AWS storage option would you choose for building your database?
- What is the maximum file size allowed for a single file in S3?
- Which AWS storage option is best suited for storing videos?
- In what scenarios is it not recommended to use the S3 web interface for uploading data?
- When should you consider using the S3 Glacier storage option?

5.4.2. Practical

- Can you spin up an EC2 instance?
- How can you perform a scale-up operation on an EC2 instance?
- What is the process for creating an AMI? (You want to explore this in today's worksheet)
- Why are key pairs important, and how can they be used?
- Do you have experience with basic UNIX commands on a server, and are you able to SSH into a server?

Answers for theory:

- What were the 5 categories of AWS services we discussed? Here are the 5 main categories of AWS services:
  - Compute
  - Storage
  - Database
  - Networking
  - Identity and access management
- What is an EC2 instance, and can you explain it?
  - EC2 is a virtual machine in the cloud. It is a server that you can use to run your applications. The full form is Elastic Compute Cloud.
- How do S3, EFS, and EBS differ from each other?
  - S3 is a simple storage service. It is an object storage service.
  - EFS is an elastic file system. It is a file storage and is designed to scale to petabytes.
  - EBS is an elastic block storage. It is block storage and is similar to the regular storage that we use on our laptops.
- Which AWS storage option would you choose for building your database?
  - EBS
- What is the maximum file size allowed for a single file in S3?
  - 5 TB
- Which AWS storage option is best suited for storing videos?
  - S3
- In what scenarios is it not recommended to use the S3 web interface for uploading data?

    - When you have a large amount of data to upload.
- When should you consider using the S3 Glacier storage option?
    - When you have a large amount of archived data that you don't need to access frequently.

5.5. Halfway check

Let's summarize Big Data solutions and whatever we learned in our previous classes;

5.5.1. Local solutions (Lectures 1 and 2)

5.5.1.1. Storage

- Use an efficent file format - parquet
    - Partition pruning
    - Predicate pushdown
    - Projection pushdown
    - Self-describing file (schema included)
    - Schema evolution
    - Decrease the size of the file - saves storage space and I/O
    - Splittable for parallel processing

5.5.1.2. Processing

- Bring just what is needed to memory (usecols, dtype, working on individual files, chunksize)
- Parallel processing so that pieces of files can be processed at the same time. (packages like Arrow does all these for you)
- Some packages that make use of Arrow as a backend are;
    - Dplyr in R
    - Polars in Python
    - Duckdb in R and Python as you can SQL query on top of the data.

The local solutions are applicable everywhere, but when we get to a much larger data set, then these techniques alone won't help. Hence we wanted to learn about scale-UP and scale-OUT solutions.

5.5.2. Scale UP (Lectures 3 and 4)

With cloud computing, a scale-up solution is just one click away. Before the rise of cloud computing, it was considered very difficult as it had to be done in an on-premise setup. As a student, you can think about buying a new 32 GB RAM laptop when things don't work anymore in your 16 GB laptop. You can, of course, think about all the difficulties involved in it.

We learned about cloud computing and spinning up an EC2 instance and how we can scale UP (or even scale DOWN) instances by changing the instance type. We also learned about various options to control (stop, terminate, etc.) your instance.

Even though a scale-up solution sounds easy with cloud computing, it might not be appropriate in many situations, especially when dealing with big data. Here are some of the disadvantages you can think of with a scale-up solution:

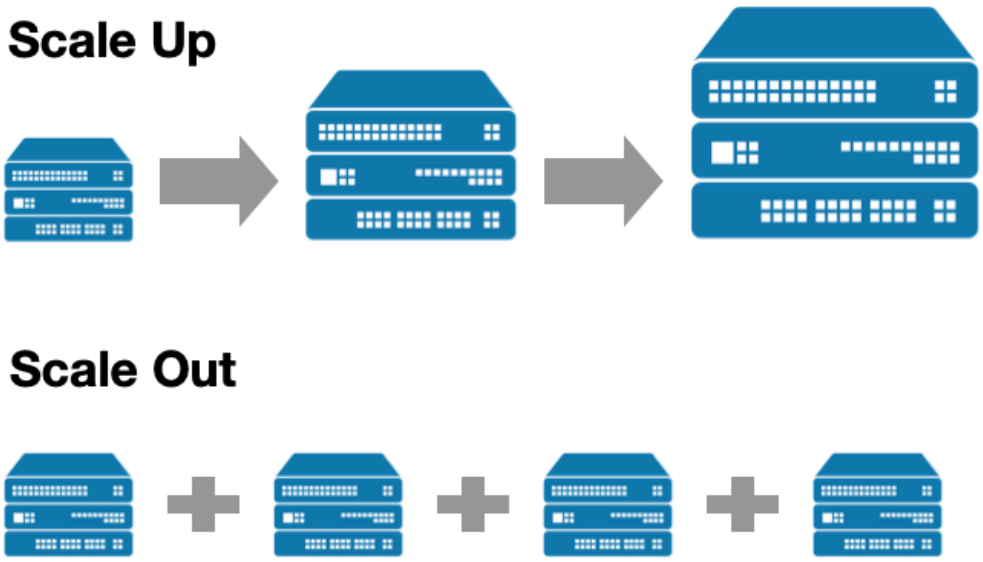
- No default high-availability
- Single-point failure
- Server upgrade can be tricky (I agree with cloud computing, it is easy!)
- Downtime related to upgrade

Adding some advantages related to the scale-out solution;

- Scale-out is easy - as you want to add servers when the load is high, and you can remove servers when the load is less.
- You can make use of distributed frameworks and their advantages for processing.

To summarize: a scale-up solution makes more sense when thinking about just an individual working on some data on a research project. However, looking from a large-scale or enterprise solution standpoint, a scale-up solution is not a great option in most cases as there is always an expectation of huge and steady growth in data in the long run. So, let's get into the details of a scale-out solution.

Here is a diagram that shows the difference in scale-up and scale-out solutions.



5.5.3. Scale OUT (8 min)

Add more computers so that it behaves as a single machine. There are many scale OUT solutions available, like cluster computers, HPC, etc... but we will focus on Hadoop and Spark.

5.6. HADOOP 🐘

- Hadoop is an open-source framework for storing data and running applications in clusters.
- Horizontal scalability
- Failure is normal and expected
- Data Locality - Compute should move to the data ❤️

Fun Fact

5.6.1. STORAGE (Hadoop Distributed File System - HDFS)

- Data automatically is divided into small blocks and is distributed across all the machines.

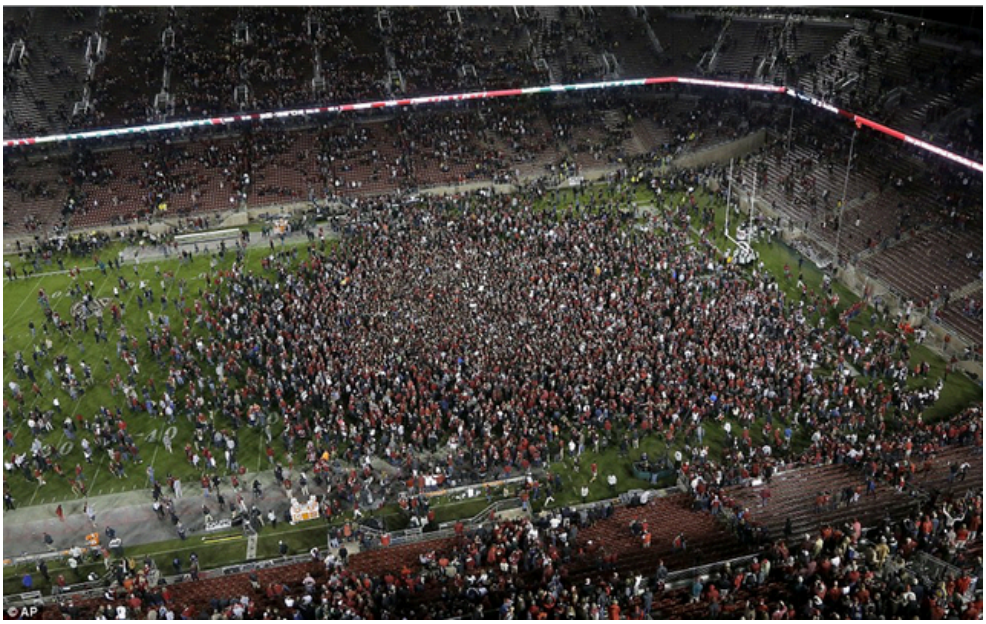
5.6.2. PROCESSING (MapReduce)

- A Programming paradigm that consists of mappers and reducers to process data stored in HDFS

5.6.3. ANALOGY for Hadoop

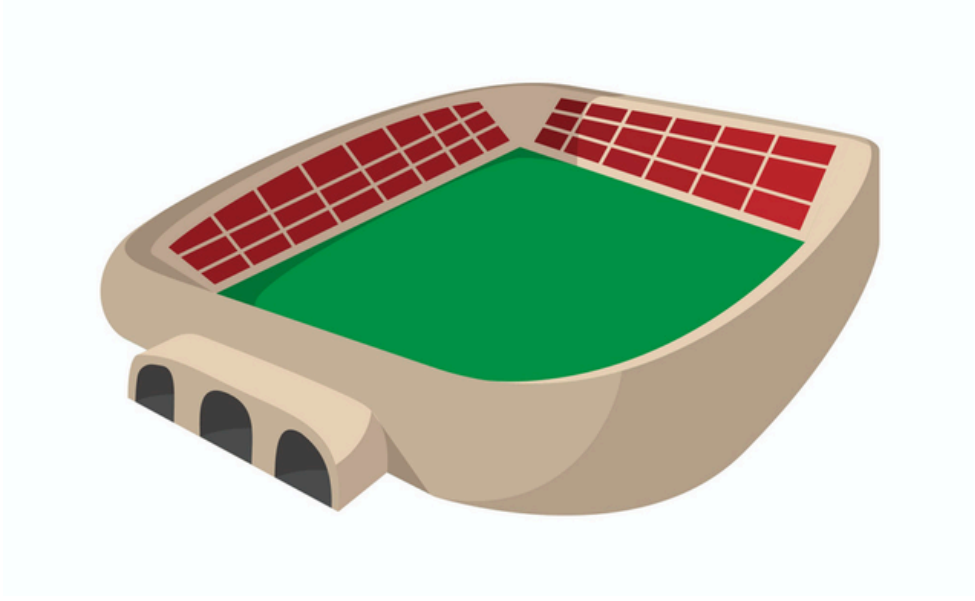
5.6.3.1. PROBLEM 😔

Lots of people in stadium

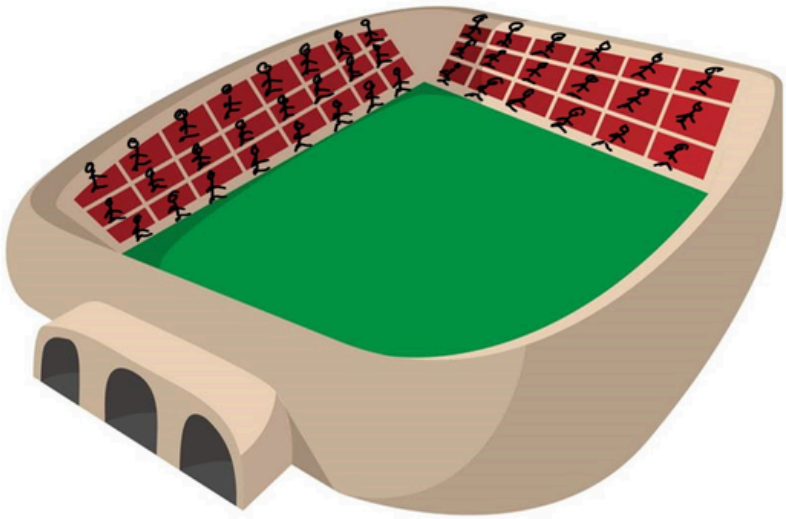


Mess when the referee called them for polling

5.6.3.2. HADOOP SOLUTION 😀

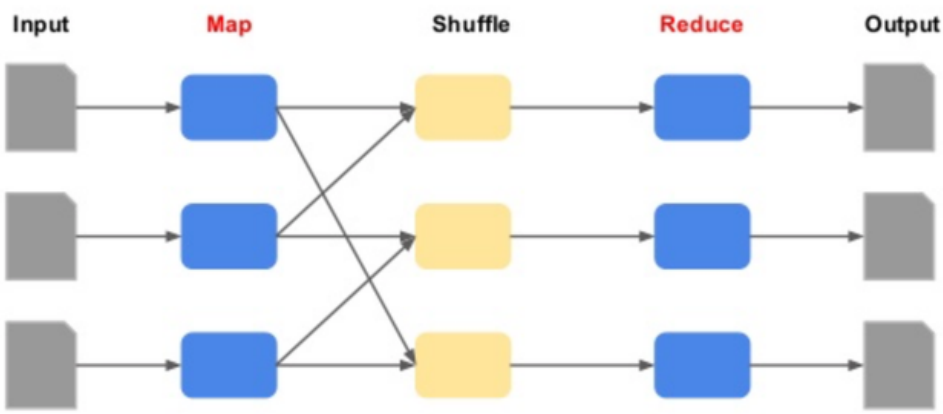


Seating Arrangement



Sending ball boys for counting

Ball boys returning back to referee with poll



Steps in Map Reduce

5.6.3.3. Code/Question : Are there more TeamA or TeamB supporters?

STADIUM :::: HADOOP

- Lots of People :::: Big Data
- Seats :::: Storage (HDFS)
- Seat Blocks :::: Blocks in HDFS
- Ball boys :::: Mappers
- Referee :::: Reducer

5.6.4. Spark

Now that we know about Hadoop. What is Spark?

- Spark runs up to 100x faster in memory and even 10x faster on disk than Hadoop
- Easier to Use – more high-level operators.
- Provides rich APIs in Java, Scala, Python, and R.

- What is the difference between Hadoop & Spark? ❌
- What is the difference between MapReduce and Spark? ✅

See also

We will revisit Spark with more details in our next class.

Knowledge Check 1:

- Write down some points on the benefits of Hadoop.
- Write down some disadvantages of Hadoop. (I haven't covered this in the class, but I want to hear your thoughts)

5.6.5. Hadoop distributions

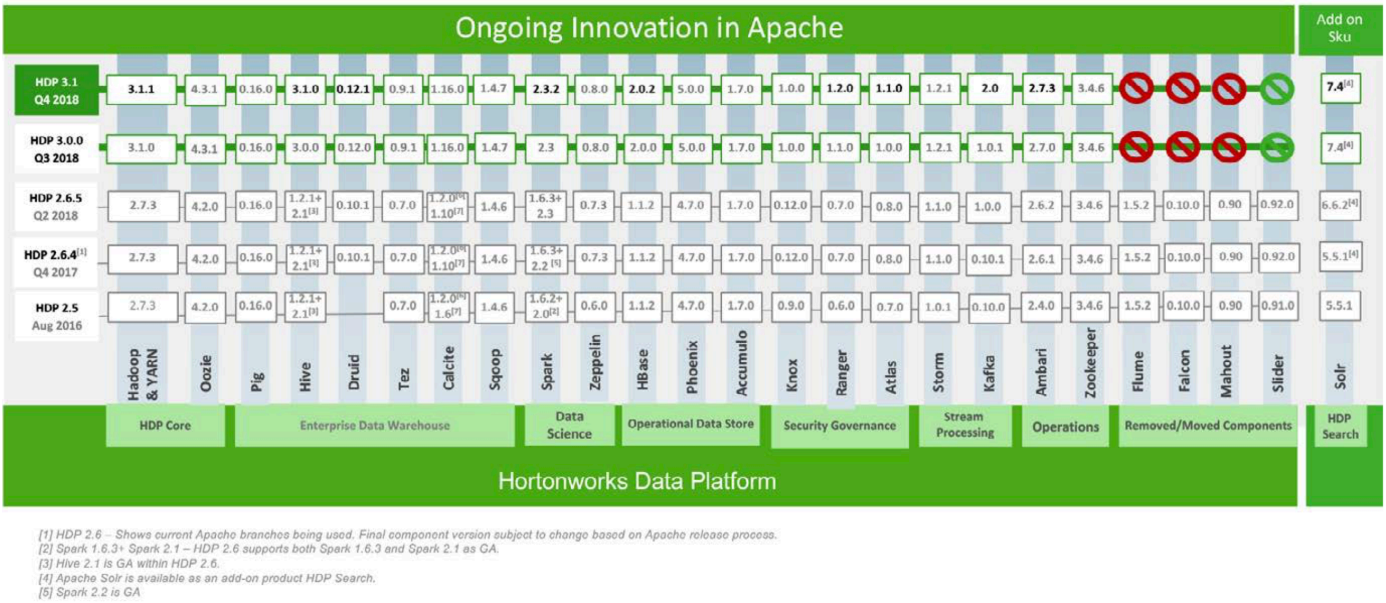
Now we know about Hadoop. You also want to know that there are various components of Hadoop and various other elements(toolsets) in the Hadoop ecosystem. Each is built to handle different workloads. Here are some tools in the Hadoop ecosystem and their uses...

Tool/package	Use
Sqoop	To transfer data from RDBMS to HDFS
Flume	To transfer data from other filesystems to HDFS
Mahout	Machine learning library to work with MapReduce
Storm	Real-time processing in Hadoop
Pig	high-level program for processing over the MapReduce
Apache Giraph	Graph processing that runs on Hadoop

The big question is: How can we set up a Hadoop system with the tools we want? All of these projects are open-source, and we can install them directly from their source pages. However, setting it up correctly is complicated because we need to ensure that all the independent packages/tools we install are properly connected to Hadoop and are able to communicate with each other. Recognizing this difficulty, many vendors have developed solutions to automate the building of the entire Hadoop stack, including all the desired elements. They have contributed to the open-source community and offer enterprise solutions that are widely used in the industry. These vendors distribute and manage these services. I am listing a few of these vendors...

- Hortonworks
- Cloudera
- MapR
- IBM
- Azure
- AWS

Here is what the Hortonworks distribution stack used to look like...



[1] HDP 2.6 – Shows current Apache branches being used. Final component version subject to change based on Apache release process.
[2] Spark 1.6.3+ Spark 2.1 – HDP 2.6 supports both Spark 1.6.3 and Spark 2.1 as GA.
[3] Hive 2.1 is GA within HDP 2.6.
[4] Apache Solr is available as an add-on product HDP Search.
[5] Spark 2.2 is GA

AWS's solution to distribute and manage these services is EMR (Elastic Map Reduce). EMR cluster provides us with an easy way to install and manage these services.

> **Note**
>
> I haven't mentioned Spark in the above Hadoop ecosystem, as Spark emerged as a project to be independent and run independently. But most people include spark as part of the Hadoop ecosystem as in enterprise solutions (or in the industry), spark is tied up with the Hadoop cluster to utilize HDFS (you already know) and YARN (cluster manager for Hadoop)

5.7. EMR cluster architecture

Here is the architectural diagram for EMR;

- Master node

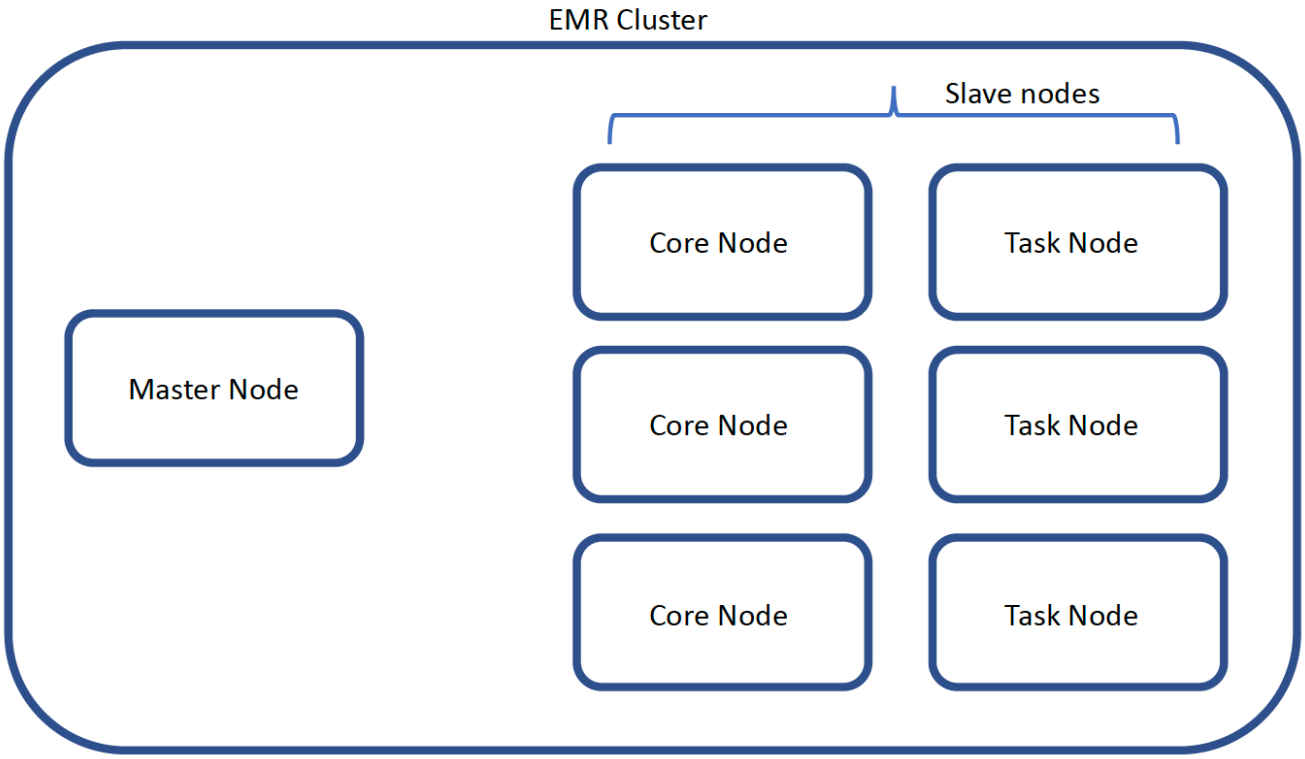
This node is like the conductor of the EMR cluster, coordinating the work of the other nodes and making sure everything runs smoothly. It keeps an eye on the cluster's health, and if anything goes wrong, it'll take care of the problem.

- Core node

Core nodes are responsible for storing and processing data, so they do the majority of the heavy lifting. They're always present in an EMR cluster, and the more you have, the more data you can process.

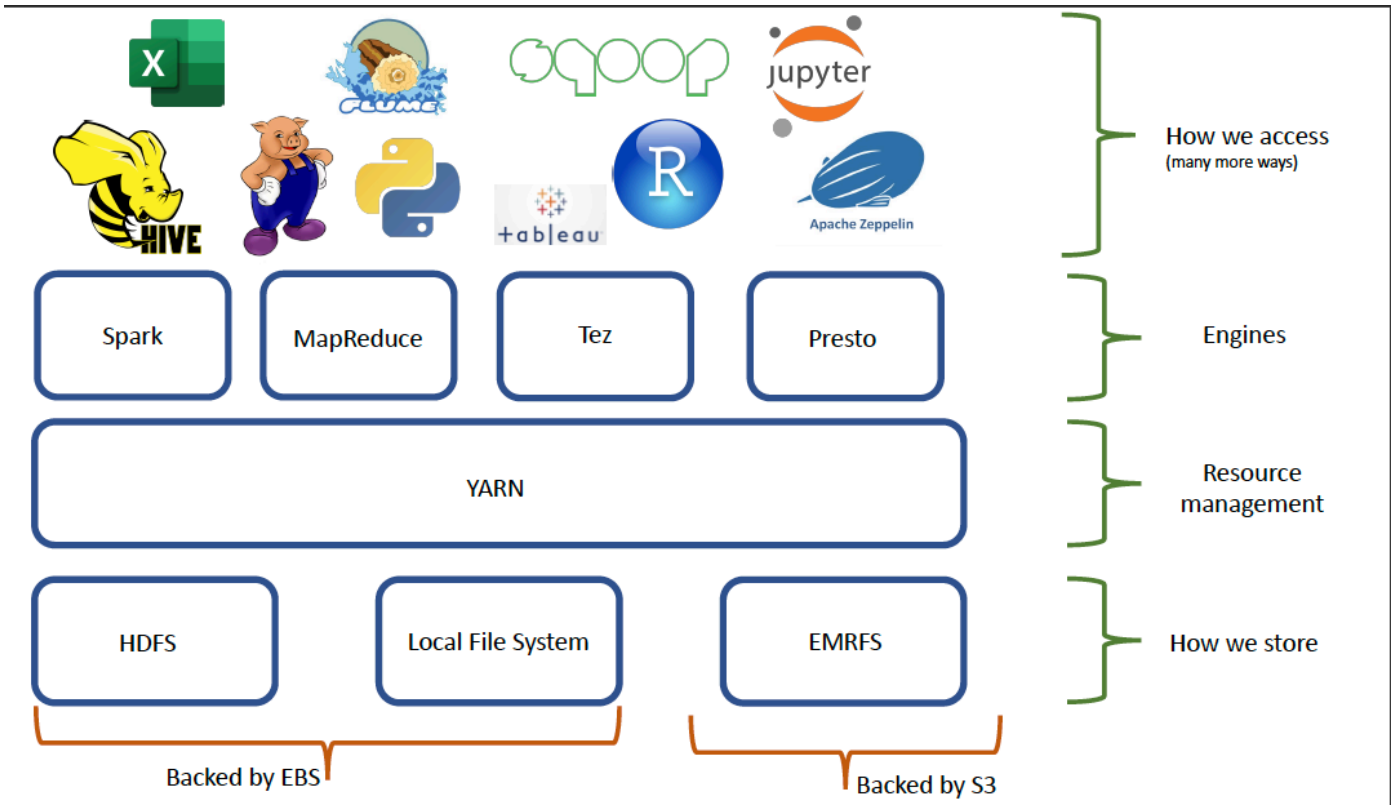
- Task node

Task nodes, on the other hand, don't store data. Instead, they're there to add more processing power to your cluster. They're not required, but if you need extra processing power, you can add them to your cluster.



Check out details on these concepts here

Here are the different layers of the EMR setup;



Tip

Best practices for EMR include using Spark and EMRFS.

5.8. What we learned today?

- We learned about situations where we want to go for the scale-out solution and about its advantages.
- The working of HDFS and MapReduce to understand how it helps to solve the big data problem.
- We learned about the architecture of the EMR cluster.