Hadoop Introduction

Department of Computer Science,

National University of Computer & Emerging Sciences,
Islamabad Campus

Hadoop - Why ?

- Need to **process** **huge datasets** on **large** **clusters** of **computers**
- **Very** **expensive** to **build** **reliability** into **each** **application**
- **Nodes failure** is **common**

- **Need a common infrastructure**
  - **Efficient, reliable, easy to use**
  - **Open Source**

So what is this Hadoop?

- A **framework** for **distributed processing** of **large data sets across clusters** of computers based on **simple programming models**

- Can **scale up** from **single server** to **thousands** of **machines** (each **offering local computation** and **storage**).

- **Designed** to **detect** and **handle failures** at the **application layer**, so **delivering** a **highly-available service** on top of a cluster of computers

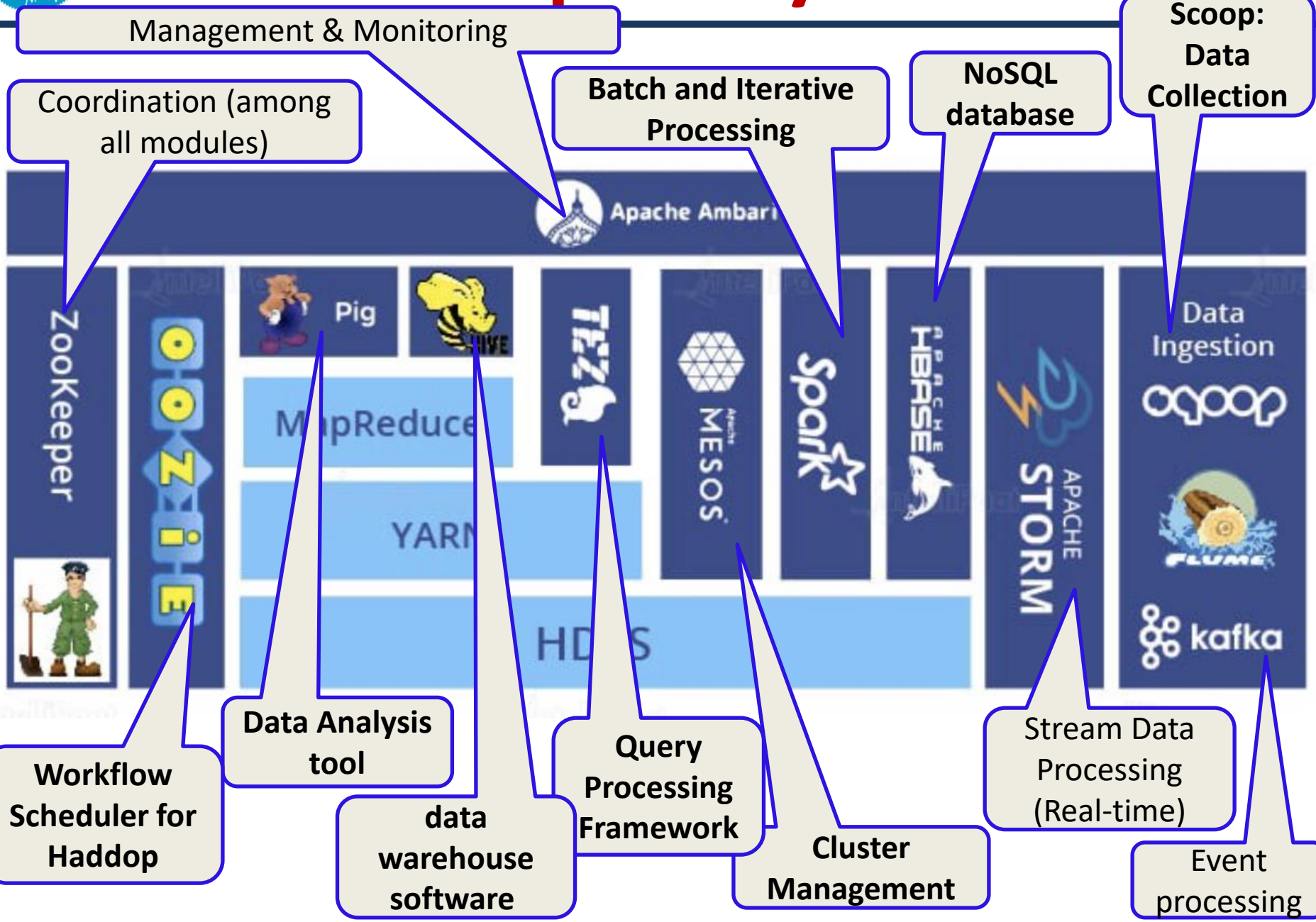
Who uses Hadoop?

- Amazon, Facebook, Google, Twitter, New York Times, Veoh, Yahoo! …. many more

Core components

- **The project includes these modules:**
  - **Hadoop Common**: The **common utilities** that **support** the **other Hadoop modules**
  - **Hadoop Distributed File System (HDFS)**: A **distributed file system** that **provides high-throughput** access to **application data**.
  - **Hadoop YARN**: A **framework** for **job scheduling** and cluster **resource management**.
  - **Hadoop MapReduce**: A **programming model** for **large scale data processing**.

Hadoop Ecosystem



Hadoop Distributed File System (HDFS)

Goals of HDFS

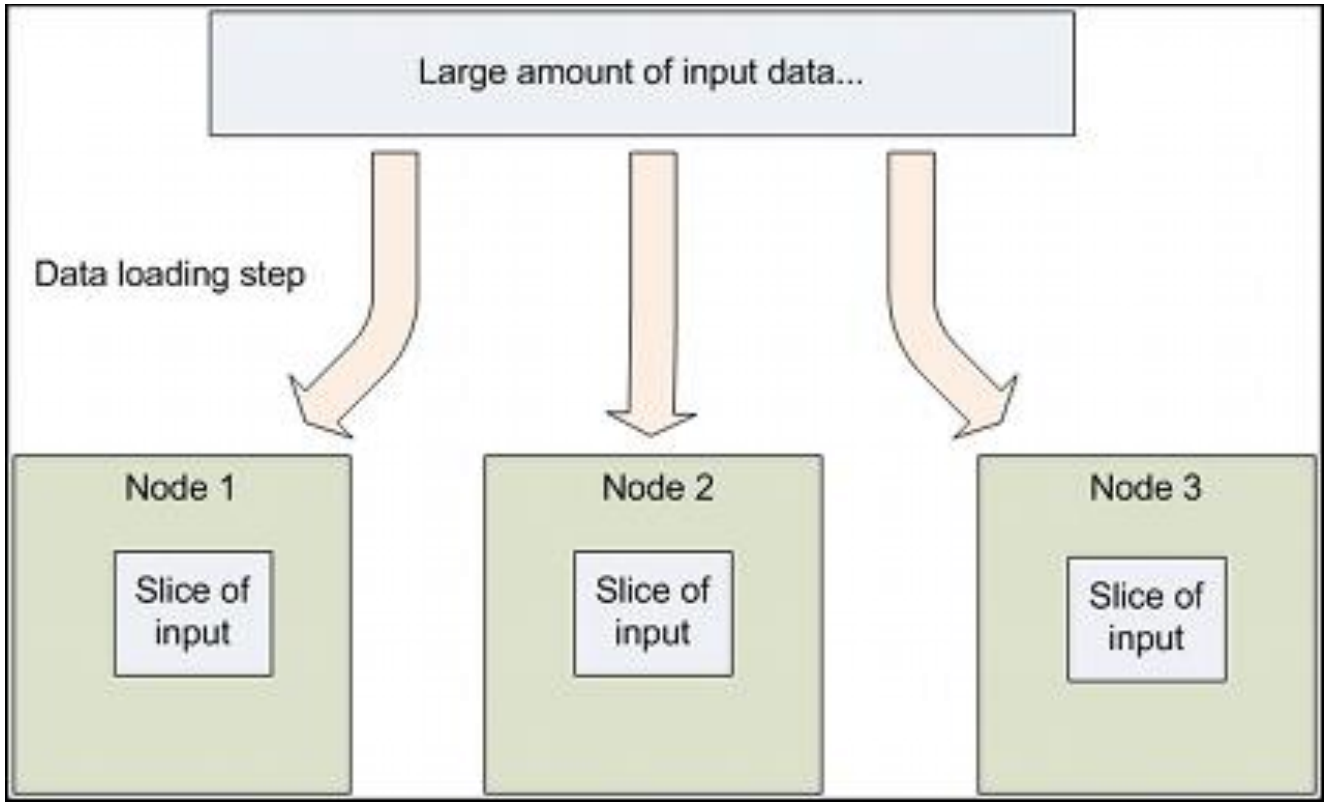
- A **filesystem** designed for **storing very large files** with **streaming data access** patterns, running **on clusters**
- "**Very large**" in this context means **files** that are **hundreds** of ***megabytes***, ***gigabytes***, or ***terabytes*** in size
  - **Streaming data access** - **write once, read-many-times** pattern
    - Supports **efficient batch processing** (***no need for synchronization***)

Hadoop Distributed File System (HDFS)

- **HDFS** has **demonstrated** **production** **scalability** of up to **200 PB of storage** and a single **cluster** of **4500 servers**,
  - A billion files and blocks

- **HDFS** is a **scalable**, **fault-tolerant**, **distributed storage system**.

Hadoop Distributed File System (HDFS)



Hadoop MapReduce

Hadoop MapReduce

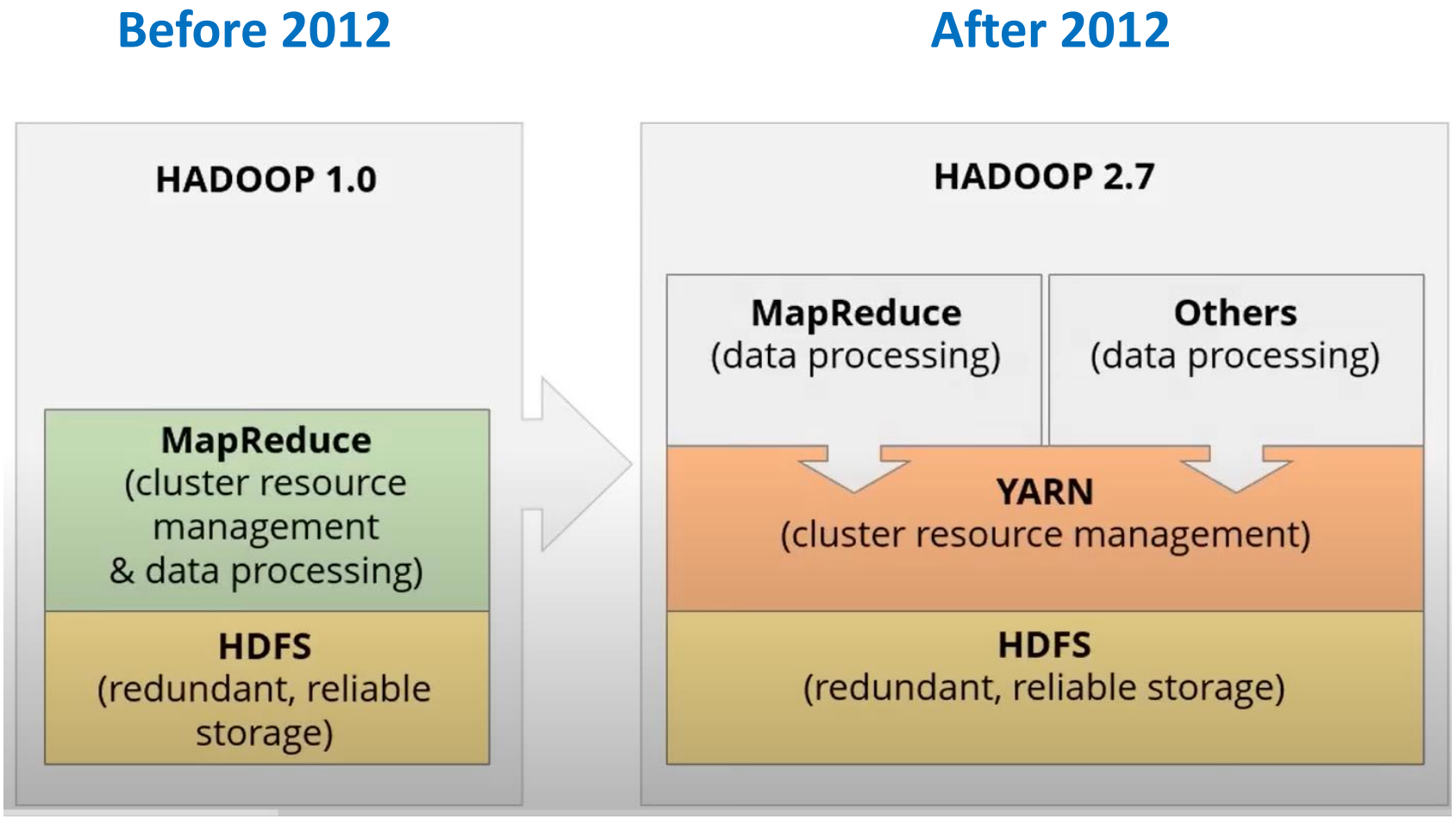
- A **framework** for **writing applications** that **process large amounts of data** stored in the **Hadoop Distributed File System (HDFS).**
- **MapReduce works** by **breaking** the **processing** into **two phases**:
  - the **(1) Map phase** and the **(2) Reduce phase.**
  - Each **phase** has **key-value pairs** as **input** and **output** (type chosen by the programmer).
  - The **programmer specifies two functions**: the **map function** and the **reduce function**.

MapReduce

Benefit	Description
Simplicity	Developers can write applications in their language of choice, such as Java, C++ or Python, and MapReduce jobs are easy to run
Scalability	MapReduce can process petabytes of data, stored in HDFS on one cluster
Speed	Parallel processing means that MapReduce can take problems that used to take days to solve and solve them in hours or minutes
Recovery	MapReduce takes care of failures. If a machine with one copy of the data is unavailable, another machine has a copy of the same key/value pair, which can be used to solve the same sub-task. The JobTracker keeps track of it all.
Minimal data motion	MapReduce moves compute processes to the data on HDFS and not the other way around. Processing tasks can occur on the physical node where the data resides. This significantly reduces the network I/O patterns and contributes to Hadoop's processing speed.

YARN(Yet Another Resource Negotiator)

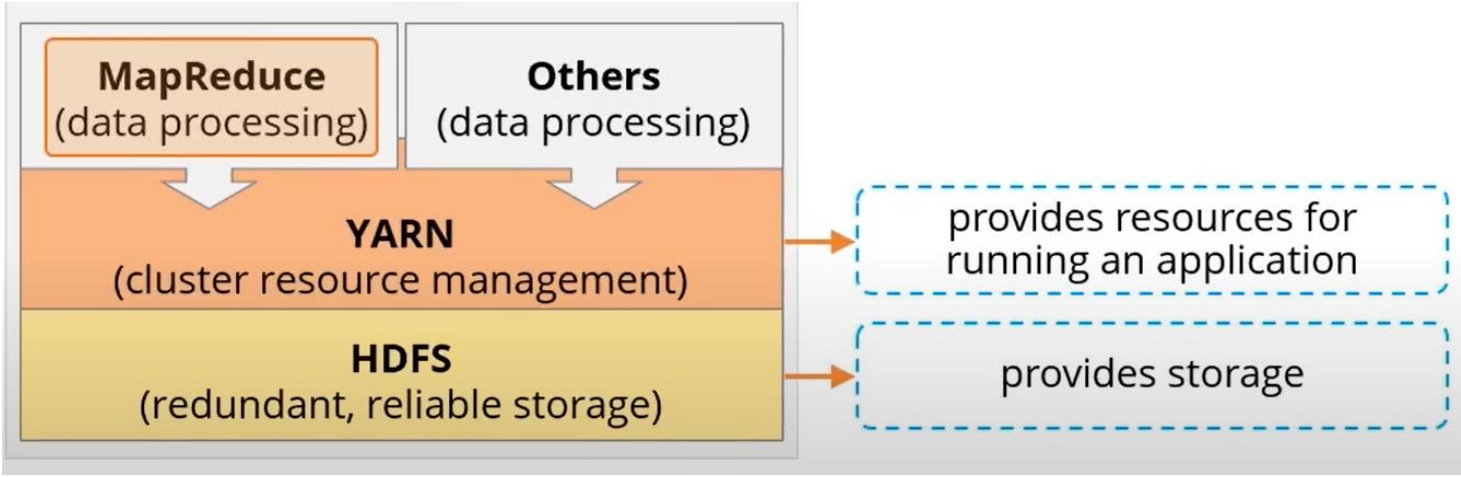
YARN (Yet Another Resource Negotiator)



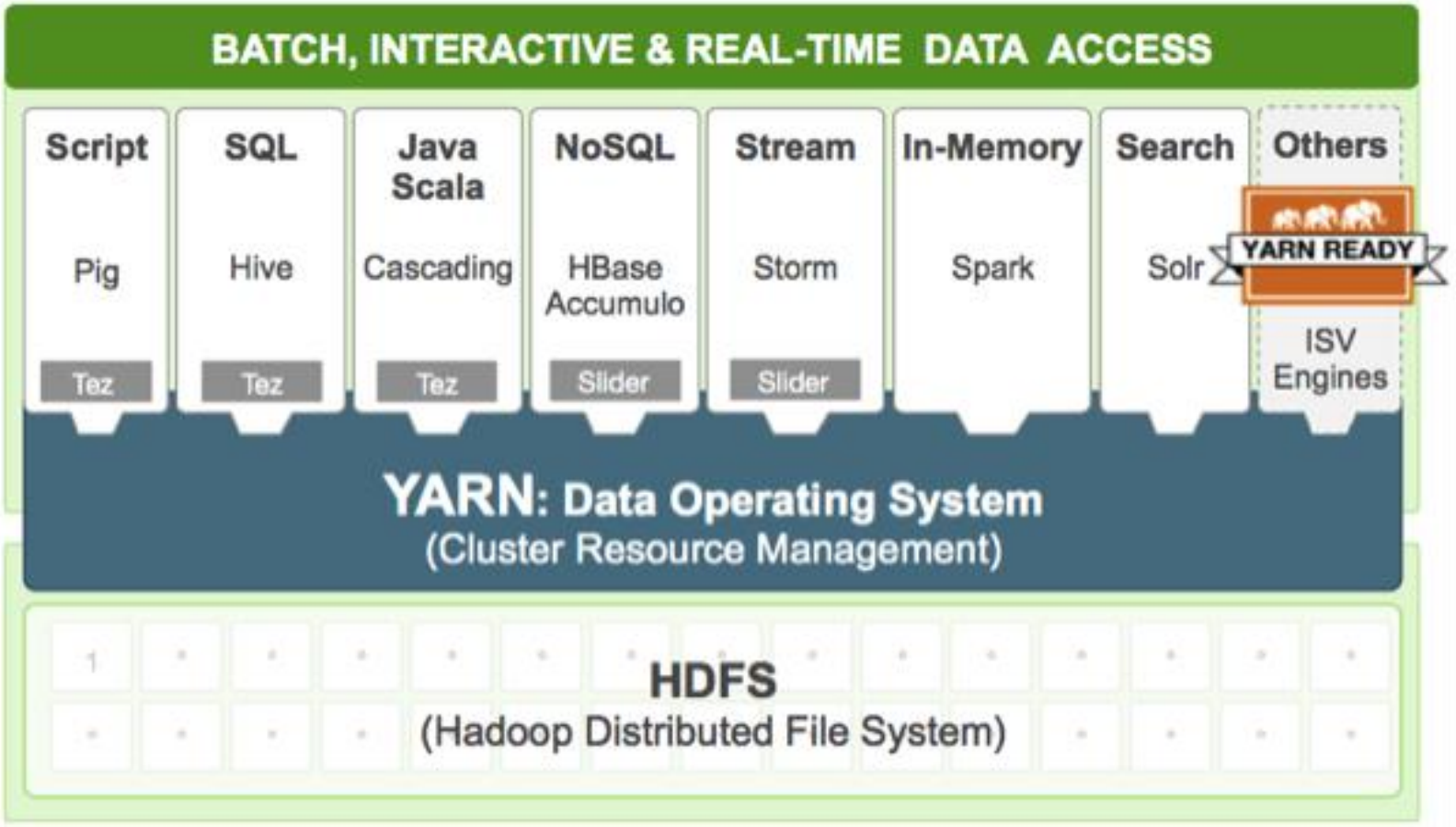
Apache YARN (Yet Another Resource Negotiator)

- **YARN mainly provides resource management**

- **Allows multiple data processing engines** using a single platform:
  - **Interactive SQL**,
  - **Real-time streaming**,
  - **Batch processing**
  - **…**



Apache YARN(Yet Another Resource Negotiator)



Five *Pillars* of Hadoop

- The term "**Hadoop**" is **sometimes** used to **refer to a larger ecosystem** of **projects**,
  - **not just HDFS** and **MapReduce**
  - **All fall under** the **umbrella** of **infrastructure** for **distributed computing** and **large-scale data processing.**

- We can organize them in **five broad categories**

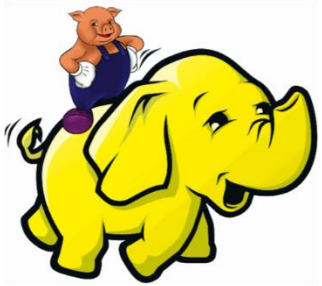
1. Data Management

- **Hadoop Distributed File System (HDFS)** is a **Java-based file system** that provides **scalable** and **reliable data storage**.

- **Apache Hadoop YARN** provides the **resource management** and **pluggable scheduling architecture** for **enabling** a **wide variety** of **resource management stretegies.**

2. Data Access

- **Apache Hive** – is a **data warehouse** that **enables easy data summarization** and **ad-hoc queries** via an **SQL-like** interface for **large datasets stored** in HDFS.



- **Apache Pig** – A **scripting platform** for **processing and analyzing large data-sets**. **Pig** consists of a **high-level language** (called ***Pig Latin***) for **expressing data analysis programs.**

2. Data Access

- ***MapReduce*** – is a **procesing framework** for **writing applications** that **computes large amounts of data**.

- ***Apache Spark*** – is **ideal** for **in-memory data processing**. It **allows** data scientists to **implement fast**, **iterative algorithms** for **advanced analytics** such as **clustering** and **classification (Machine Learning)** of **datasets**.

2.Data Access

- ***Apache Storm*** – is a **distributed real-time computation system** for **processing fast, large streams** of **data adding reliable real-time data processing capabilities** to Apache Hadoop® 2.x

- ***Apache Hbase*** – A **column-oriented NoSQL data storage** system that **provides random real-time read/write access** to **big data** for **user applications**.

- ***Apache Tez*** – Used with **Hive & Pig** on top for **speeding up query processsing**. An **alternative** to **MapReduce** to **speedup Query processing** for **near real-time big data processing**.

2. Data Access

- ***Apache Kafka*** – is a **fast** and **scalable publish-subscribe messaging system** having **higher throughput**, **replication**, and **fault tolerance**.

- ***Apache HCatalog*** – A **table** and **metadata management service** that **provides** a **centralized way** for **data processing systems** to **understand** the **structure** and **location of the data** stored within Apache Hadoop.

- ***Apache Slider*** – A **framework** for **deployment of long-running data access applications** in Hadoop.

2. Data Access

- ***Apache Solr*** – is the **open source platform** for **searches** of **data stored** in Hadoop. Solr **enables** powerful **full-text search** and **near real-time indexing** on many of the **world's largest Internet sites**.

- ***Apache Mahout*** – provides **scalable machine learning algorithms** for Hadoop which **aids** with **data science** for **clustering**, **classification** and **batch based collaborative filtering**.

- ***Apache Accumulo*** – is a **high performance data storage** and **retrieval system**.

3. Data Governance and Integration

- ***Apache Falcon*** – is a **data management framework** for **simplifying data lifecycle management**.

- ***Apache Flume*** – allows you to **efficiently aggregate** and **move large amounts of log data** from **many different sources** to Hadoop.

- ***Apache Sqoop***– is a **tool** that **speeds and eases movement** of **data** in and **out** of **Hadoop**

4. Security

- ***Apache Knox*** – provides a **single point of authentication** and **access** for **Apache Hadoop services** in a cluster.

- ***Apache Ranger*** – **delivers** a **comprehensive approach** to **security** for a **Hadoop cluster**. It **provides central security policy administration** across the core enterprise security requirements of **authorization**, **accounting** and **data protection**.

5. Operations

- ***Apache Ambari*** – A **web-based tool** for **monitoring** **Apache Hadoop clusters**

- ***Apache Oozie*** – Oozie **Java Web application** used to **schedule** **Apache Hadoop jobs**. Oozie **combines** multiple **jobs sequentially** into **one logical unit of work**.

- ***Apache ZooKeeper*** – A **highly available** system for **coordinating distributed processes** **(different hadoop subsystems)**. Distributed applications use **ZooKeeper** to **store** and **mediate** **updates** to **important configuration information**.

Getting Started

- We **start** with the **Hadoop installation**, **many ways** to do it.
  - **Standalone (or local) mode,** there are **no daemons running** and **everything runs** in a **single process**.
  - **Pseudo-distributed mode**, the Hadoop **daemons run** on the **local machine**, thus **simulating a cluster** on a **small scale.**
  - **Fully distributed mode,** the **Hadoop daemons run** on a **cluster of machines**.

Getting Started

- We obviously start with the Hadoop installation, many ways to do it.
  - **Standalone (or local) mode, there are no daemons running and everything runs in a single process.**
  - **Pseudo-distributed mode**, the Hadoop daemons run on the local machine, thus simulating a cluster on a small scale.
  - **Fully distributed mode,** the Hadoop daemons run on a cluster of machines.

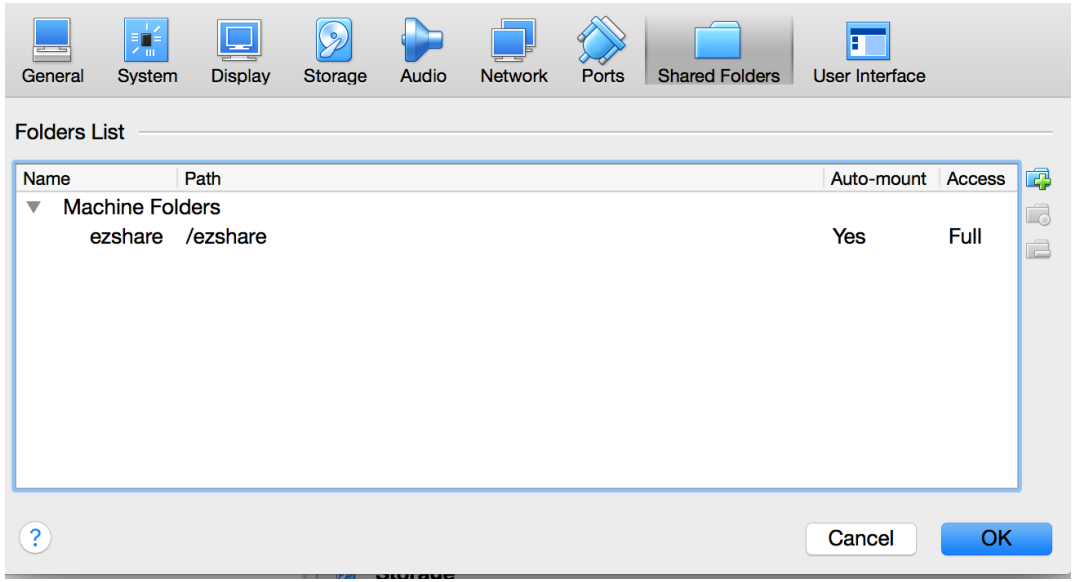
Standalone Hadoop Installation

- **Very simple** and you should have it up and running in few minutes

- Here are the steps to follow

Standalone Installation - Step 1

- **Create** a new **VM** with **Ubuntu Desktop**
  - You can share files between VM and Host



  - Once logged into new VM you would be able to view shared folder, sf_xxxx, if you have access issues use this command, sudo adduser *username* vboxsf

Standalone Installation - Step 2

- **Install Java download** compressed **file** from **Oracle site**, jdk-8u77-linux-x64.tar.gz, and extract to some folder, for me ~/java

- **Add java to your path** and **add** a **JAVA_HOME** environment variable. For this **edit** the file **/etc/profile** and **add following contents** to the end of file.

```
export JAVA_HOME=/home/ehtesham/java
export PATH=$PATH:$JAVA_HOME/bin
```

- **Run** ***source /etc/profile***, you should then be able to both echo the content of ***JAVA_HOME*** and **run** the command ***java -version***.

Standalone Installation - Step 3

- **Download** and **unzip hadoop** to a folder, for me the version was hadoop-2.6.4.tar.gz and I extracted it to ~/hadoop

- Add some more environment variables, /etc/profile

export HADOOP_HOME=/home/ehtesham/hadoop
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar

update the path environment variable to this one
Export PATH=$PATH:$HOME/bin:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

- Run ***source /etc/profile*** and verify your environment settings by executing, ***hadoop version***

Standalone Installation - Completion

- **Setup is complete**, we need to just run a simple **MapReduce job** to make sure if things work out as expected,

- **Download wordcount.jar** from SLATE and on your VM create a folder **(~/hadoopSrc**, in my case) and put the **wordcount.jar** in that **folder**.

- Next, **create a subfolder named input** and put **any text file** (for instance, the one uploaded on SLATE) in that folder.

Standalone Installation - Completion

- Run the following command, ***hadoop jar ~/hadoopSrc/wordcount.jar WordCount ~/hadoopSrc/input ~/hadoopSrc/output***

- It should show some statistics on the terminal and would create a folder ~/hadoopSrc/output with the output.

Congratulations !!