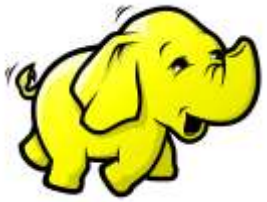
Hadoop Ecosystem

BY RAHIM A.

History of Hadoop



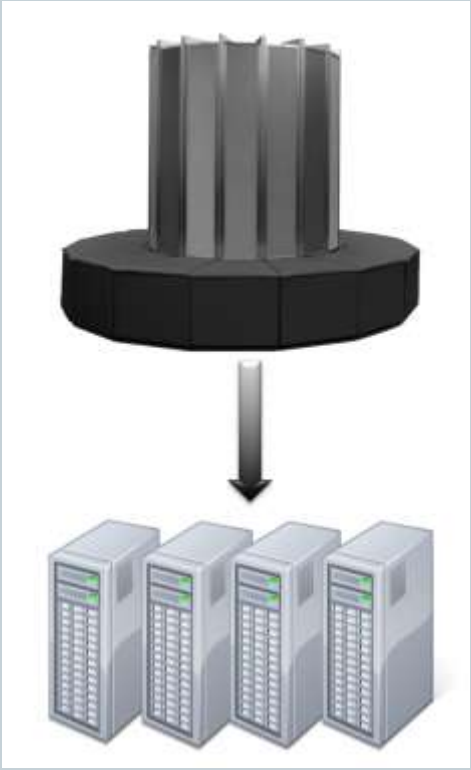
- Hadoop was created by Doug Cutting, the creator of Apache Lucene, the widely used text search library.
- Hadoop has its origins in Apache Nutch, an open source web search engine, itself a part of the Lucene project.
- In 2003, Google published a paper described Google's distributed file system GFS.
- In 2004, Nutch's developers implemented open source version on GFS, the Nutch Distributed File System (NDFS) – which evolved to HDFS
- In 2004, Google published the paper that introduced MapReduce
- In 2005, the Nutch developers implemented MapReduce in Nutch, and ported all the major Nutch algorithms to run using MapReduce and NDFS
- Thanks to Google, Hadoop including HDFS and MapReduce, was born in 2005

The Motivation for Hadoop

- "In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, we didn't try to grow a larger ox. We shouldn't be trying for bigger computers, but for *more systems* of computers."

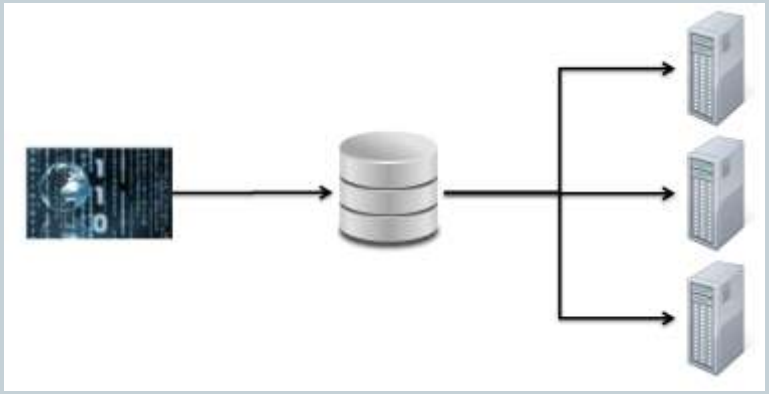
  – Grace Hopper

- Main motivation for Hadoop is to utilize distributed systems for single job

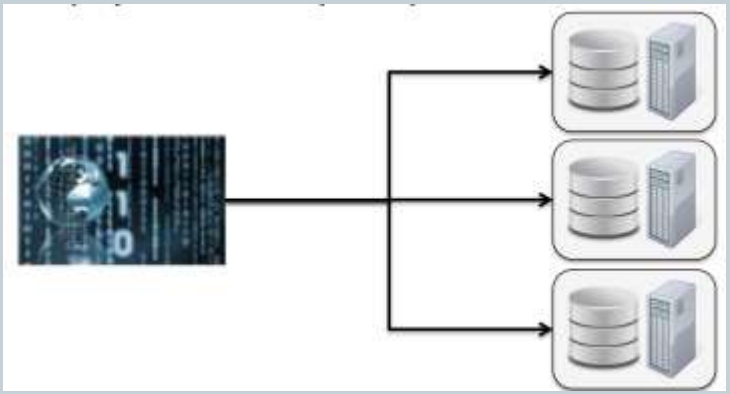


Hadoop Vs. Traditional systems

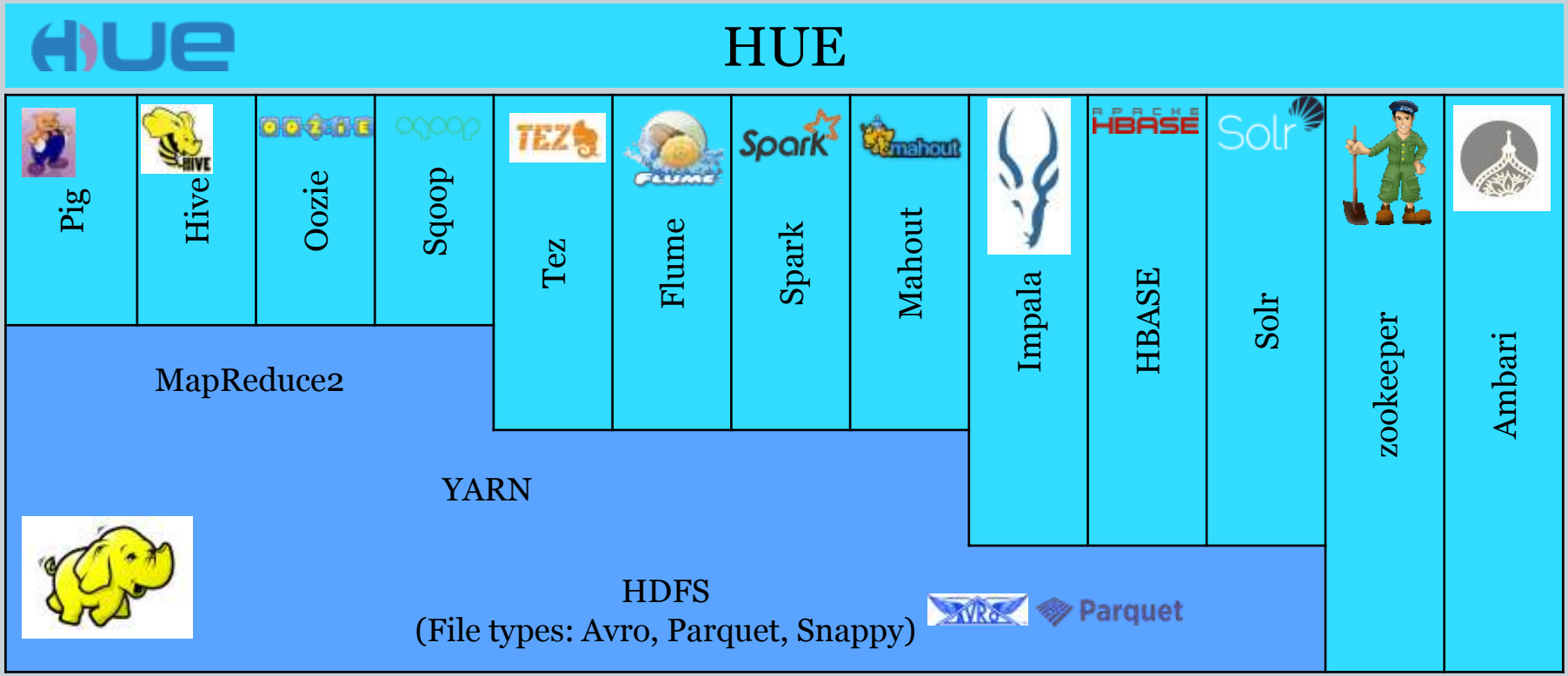
- Traditional systems
  - Data is stored in a central location
  - Data is copied to processors at runtime
  - Fine for limited amounts of data



- Hadoop
  - Distribute data when the data is stored
  - Run computation where the data is - bring the computation to the data
  - Data is replicated
  - Hadoop is scalable and fault tolerant



Hadoop Related Project



** *Apart them there are also projects – **Storm, Kafka, Samza** for streaming and message processing; **Chukwa** – for data collection; **Drill** – for interactive data analysis; **Cascading, Flink, Whirl…***

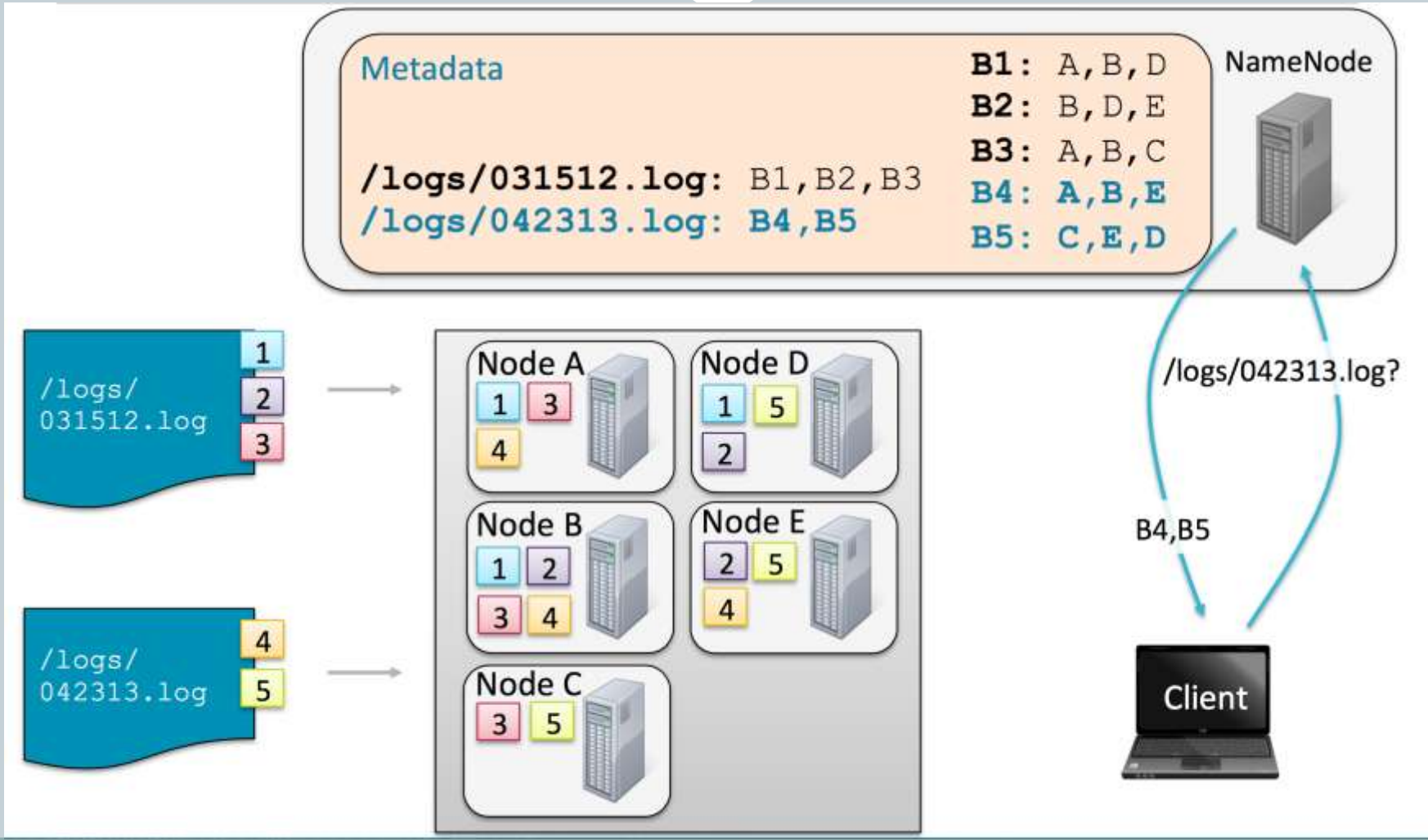
Hadoop core components

- Hadoop common
  - Common utilities to support other Hadoop modules
- Hadoop Distributed File System (HDFS)
  - Distributed file system
- Yet Another Resource Negotiator (YARN)
  - Framework for job scheduling and cluster resource manager
- MapReduce2
  - YARN based framework for parallel processing
- MapReduce1
  - Older and depreciated map reduce framework, based on own job scheduler –forget about it!

HDFS

- Properties:
  - Sits on top of a native filesystem - *such as ext3, ext4 or xfs*
  - HDFS performs best with a 'modest' number of large files – *millions, rather than billions, of files, Each file typically 100MB or more*
  - Files in HDFS are 'write once' - *no random writes to files*
  - HDFS is optimized for large, streaming reads of files – *rather than random reads*
- HDFS deamons
  - Datanode – stores file blocks
  - Namenode – store filesystem's metadata (file name and path, number of blocks, number of block replicas, blocks locations)
  - Secondary namenode – bookkeeping and check points

HDFS: file storage



YARN

- Concepts
  - YARN is cluster resource management system
  - Provides APIs for requesting and working with cluster resources – CPU and Memory
- YARN daemons
  - Resource manager (one per cluster) - to manage the use of resources across the cluster
  - Node managers - running on all the nodes in the cluster to launch and monitor containers. A ***container*** - executes an application-specific process with a constrained set of resources (memory, CPU)
  - Application's history server – only for map reduce jobs, stores job running histories

YARN features

- **Scalability**

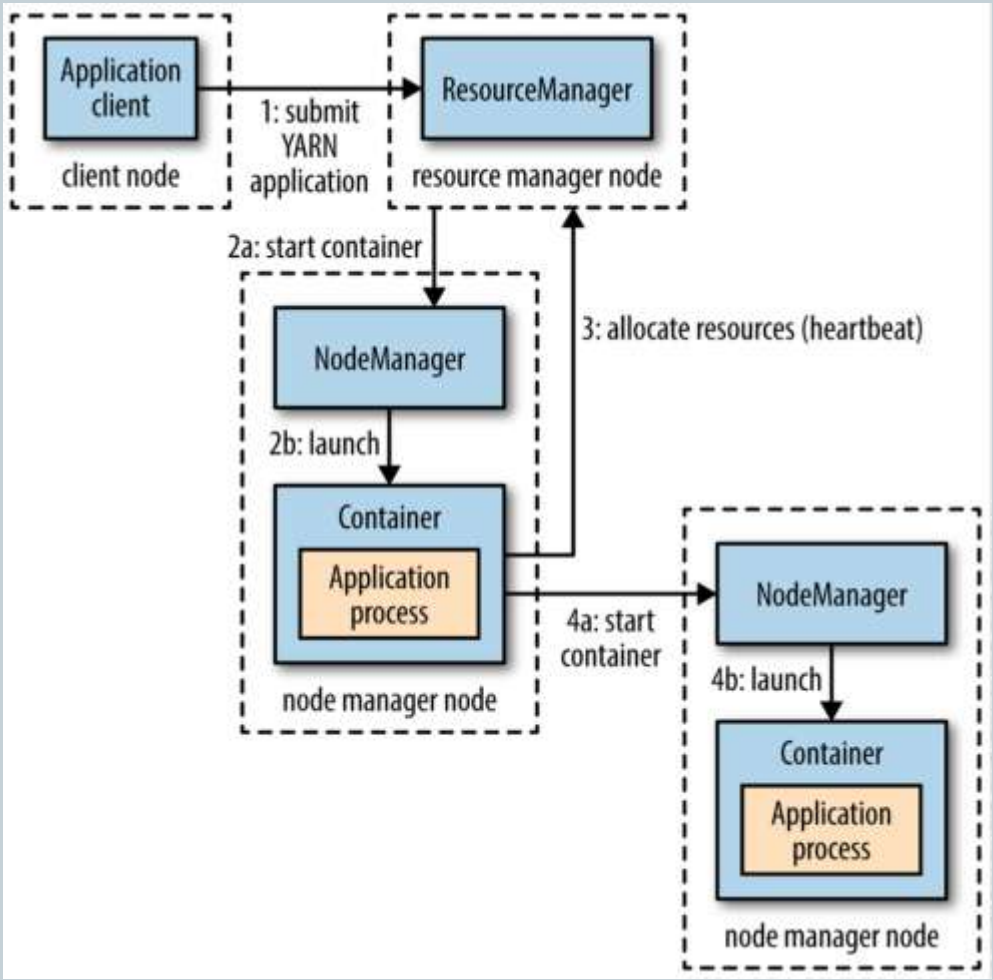
  can run on larger clusters than MapReduce 1. It is designed to scale up to 10,000 nodes and 100,000 tasks.
- **Availability**

  provide HA for the resource manager, then for YARN applications
- **Utilization**

  In YARN, a node manager manages a pool of resources, rather than a fixed number of designated slots
- **Multitenancy**

  the biggest benefit of YARN is that it opens up Hadoop to other types of distributed application beyond MapReduce”

YARN: how applications are run



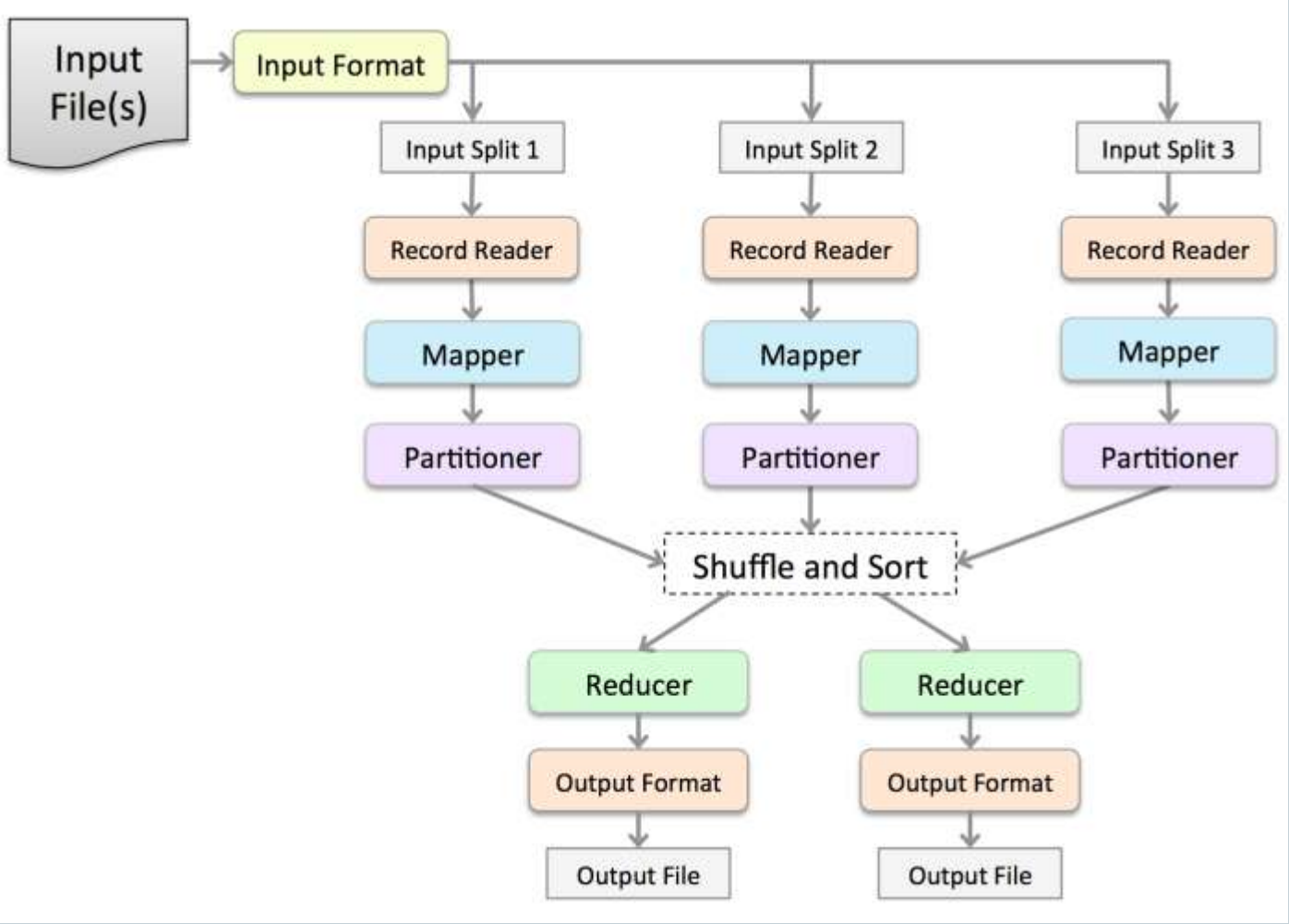
MapReduce

- MapReduce is a method for distributing a task across multiple nodes
- Each node processes data stored on that node – Where possible
- Consists of two phases:
  - Map
  - Reduce
- Features of MapReduce
  - Automatic parallelization and distribution
  - Fault tolerance
  - A clean abstraction for programmers
    - MapReduce programs are usually written in Java
    - Can be written in any language using *Hadoop Streaming*
  - MapReduce abstracts all the 'housekeeping' away from the developer

MapReduce Key stages

- **The Mapper**
  - Each Map task (typically) operates on a single HDFS block
  - Map tasks (usually) run on the node where the block is stored
- **Shuffle and Sort**
  - Sorts and consolidates intermediate data from all mappers
  - Happens after all Map tasks are complete and before Reduce tasks start
- **The Reducer**
  - Operates on shuffled/sorted intermediate data (Map task output)
  - Produces final output

MapReduce Workflow

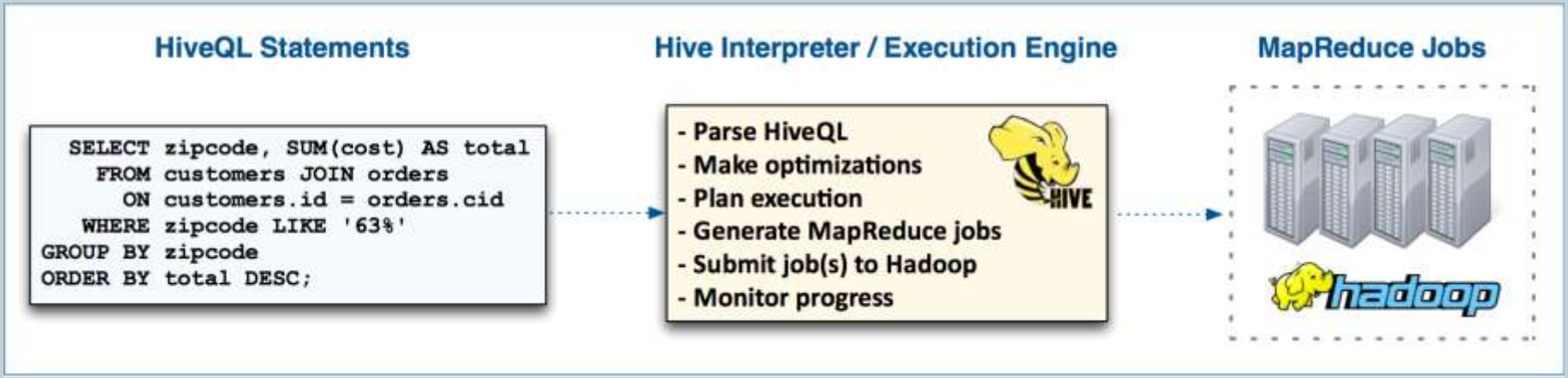


Motivation for Hadoop related projects

- Complexity of writing MapReduce jobs
- Not all tasks fits MapReduce
  - it is not good for complex directed-acyclic-graphs
  - It is not suitable for iterative algorithms
- Address business and data analyst needs
- Performance issues
- Data ingestion and integration with traditional systems
- Provide ready to use frameworks and system build on big data

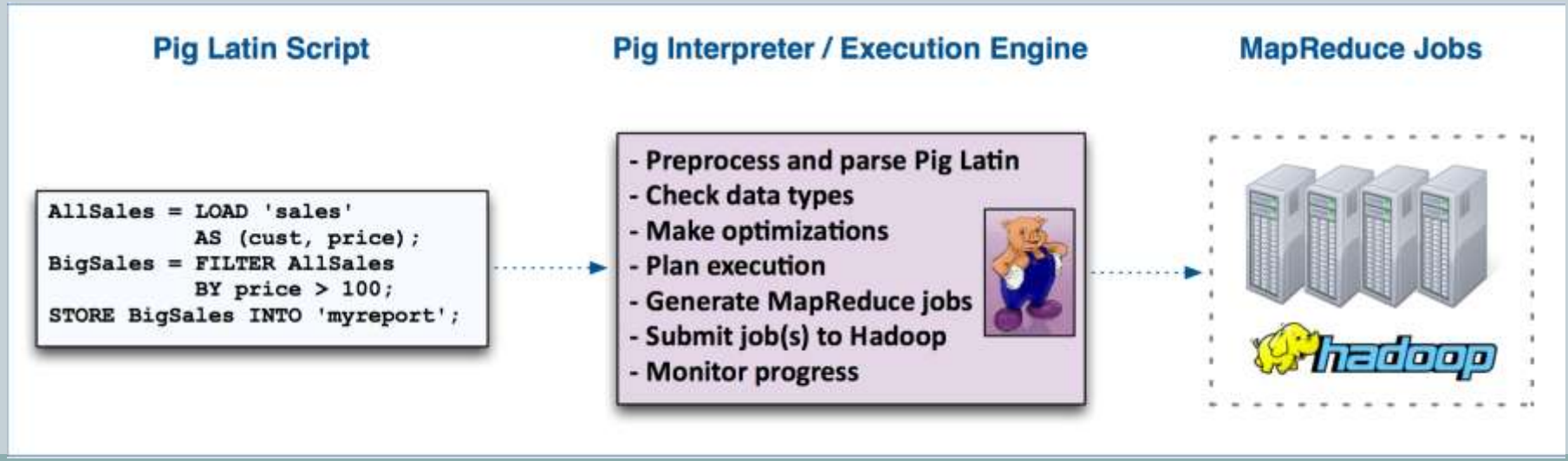
HIVE

- Apache Hive is a high level abstraction on top of MapReduce
- Uses an SQL like language called HiveQL
- Generates MapReduce jobs that run on the Hadoop cluster
- Originally developed by Facebook for data warehousing
- Hive Limitations
  - Not all 'standard' SQL is supported – No correlated subqueries
  - No support for UPDATE or DELETE
  - No support for INSERTing single rows



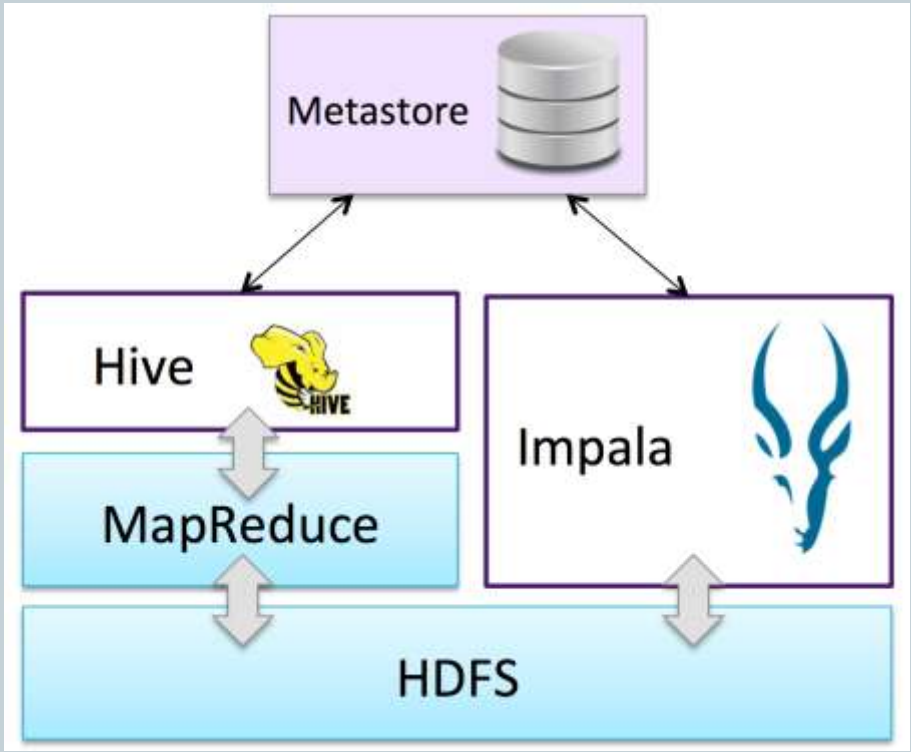
PIG

- Apache Pig is a platform for data analysis and processing on Hadoop
- Offers an alternative to writing MapReduce code directly
- Originally developed at Yahoo
- Pig Goals: flexibility, productivity, and maintainability
- Main components:
  - The data flow language (Pig Latin)
  - The interactive shell (Grunt)
  - The Pig interpreter and execution engine



Impala

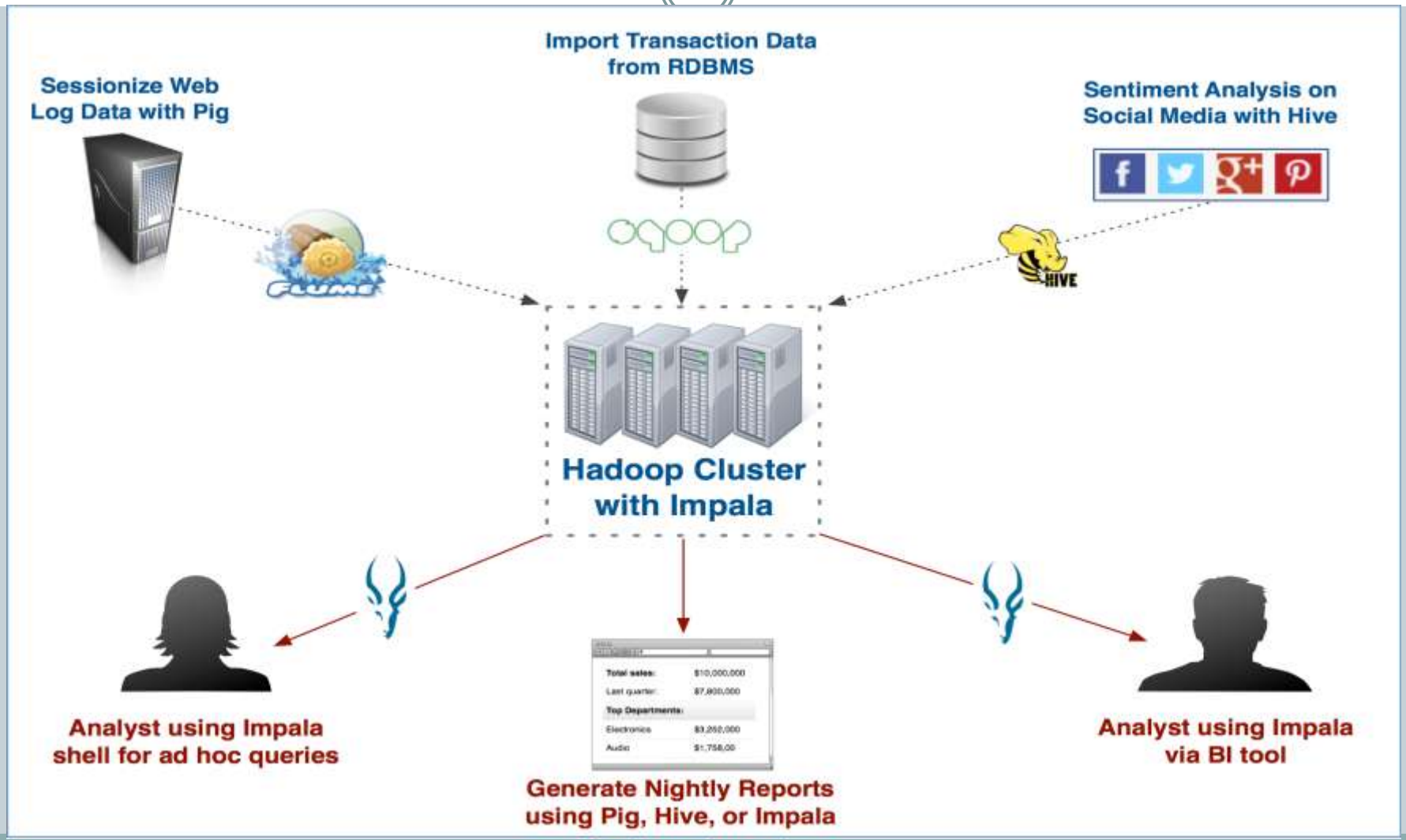
- High performance SQL engine for vast amounts of data
  - Similar query language to HiveQL
  - 10 to 50+ Ames faster than Hive, Pig, or MapReduce
- Developed by Cloudera
- MapReduce is not optimized for interactive queries -High latency – even trivial queries can take 10 seconds or more
- Impala does not use MapReduce
- Uses the same Metastore as Hive



SQOOP and Flume

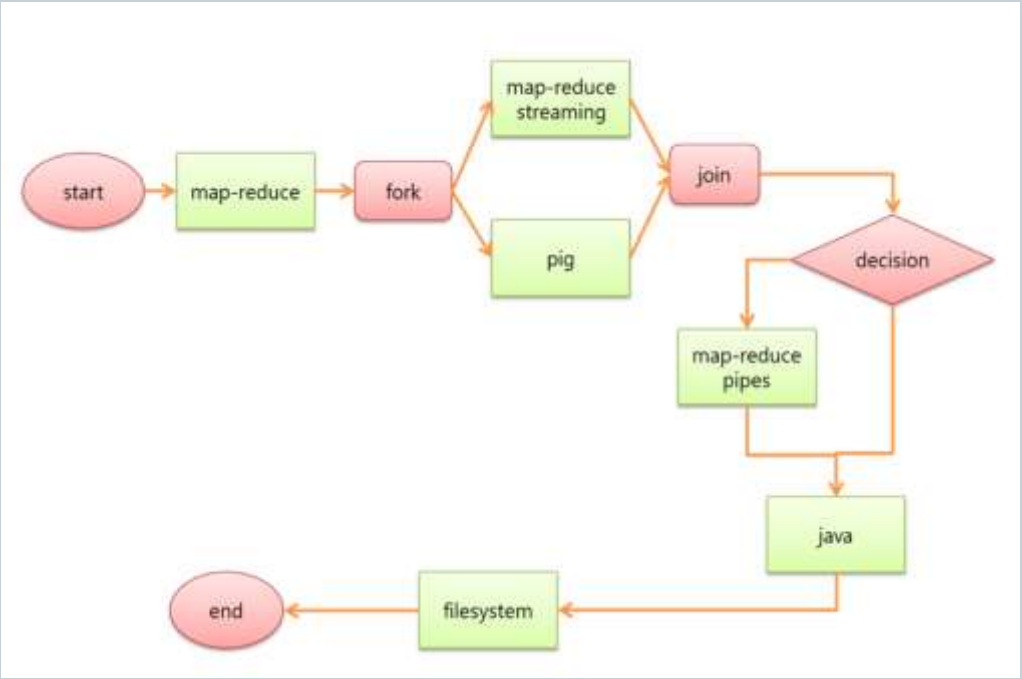
- Sqoop
  - Sqoop stands for “SQL for Hadoop”
  - Imports tables from an RDBMS into HDFS and vice-versa
    - Just one table
    - All tables in a database
    - Sqoop supports a WHERE clause
  - Uses MapReduce to actually import the data to RDBMS
  - Uses a JDBC interface
  - Supports Oracle Database (connector developed with Quest Software)
- Flume
  - Flume is a distributed, reliable, available service for efficiently moving large amounts of data as it is produced
  - Ideally suited for file ingestions (web logs, network device logs etc.)

Sample Workflow



Oozie

- Oozie is a ‘workflow engine’
- Runs on a server, typically outside the cluster
- Runs workflows of Hadoop jobs
  - Including Pig, Hive, Sqoop jobs
  - Submits those jobs to the cluster based on a workflow definition
- Workflow definitions are in XML and submitted via HTTP
- Jobs can be scheduled
  - One-off or recurring jobs



HBASE

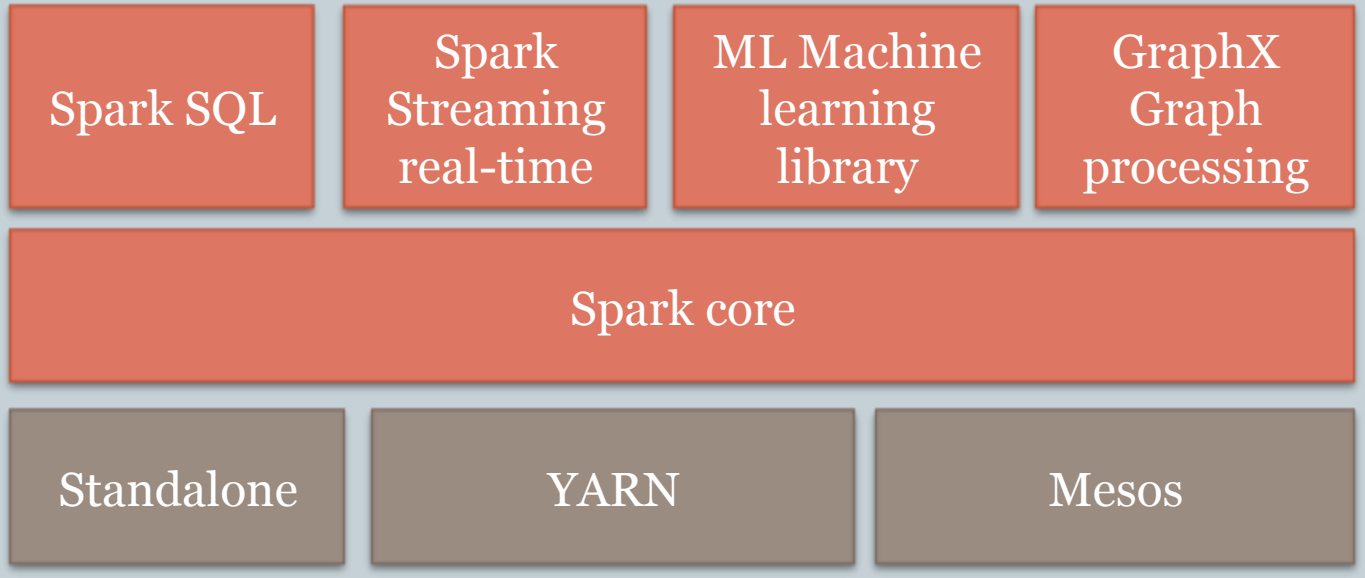
- HBase is the Hadoop database - A 'NoSQL' datastore
- Developed as open source version of Google's BigTable
- Can store massive amounts of data – Petabytes+
- High write throughput Scales to 100K inserts per second
- Handles sparse data well – No wasted spaces for empty columns in a row
- Limited access model
  - Optimized for lookup of a row by key rather than full queries
  - No transactions: single row operations only
  - Only one column (the 'row key') is indexed

HBASE vs RDBMSs

	RDBMS	HBASE
Data layout	Row oriented	Column oriented
Transaction	Supported	Single row only
Query language	SQL	Put/get/scan
Security	Buit-in Authentication-Authorization	Kerberos
Indexes	Any column	Row key only
Max data size	TBs	PB+
Read-write throughput	Thousands	Millions

Spark

- Spark is an open source project, started in 2009 as a research project in the UC Berkeley
- Spark is a cluster computing platform designed to be fast and general-purpose
- 10-20 times faster than MapReduce – so it is MapReduce killer
- Written in Scala, using Akka framework
- Spark can run as standalone cluster, on YARN, and on Apache Mesos cluster
- Spark components:



Spark components

- Spark Core
  - task scheduling, memory management, fault recovery, interacting with storage systems, and more.
  - API that defines resilient distributed datasets (RDDs), -
    - RDDs represent a collection of items distributed across Spark cluster nodes that can be manipulated in parallel.
- Spark SQL
  - Spark's package for working with structured data
- Spark Streaming
  - Spark component that enables processing of live streams of data
- MLlib
  - Provides classification, regression, clustering, collaborative filtering
- GraphX
  - Library for manipulating graphs and performing graph-parallel computations