hadoop

Big Data Analytic Platform

Instructor, Nero Chan Zhen Yu

Forward School

What is Apache Hadoop

- A software framework for storing, processing, and analyzing “big data”
  - Distributed
  - Scalable
  - Fault--tolerant
  - Open source

Something about Apache Hadoop

- Open source software
- Around 60 committers from companies and volunteer developers
  - Cloudera, Yahoo!, Facebook, Apple and more
- A large software ecosystem

HIVE
FLUME
APACHE HBASE
mahout
HUE
hadoop
Pig
Zookeeper
sqoop
AVRO
Apache Solr
OOZIE
Impala

Forward School

Why do we need something like Hadoop?

- To solve problems that exist in traditional large-scale analytics systems
    - Computation has been processor (and memory) –bound
        - Can’t just keep buying bigger computers
    - Programming for traditional distributed systems is complex
        - Synchronization for data exchange? Partial failure problem
    - Distributed systems
        - Programming complexity

- Solutions before Hadoop - ?
    - New approach needed!

Hence we need a system that can….

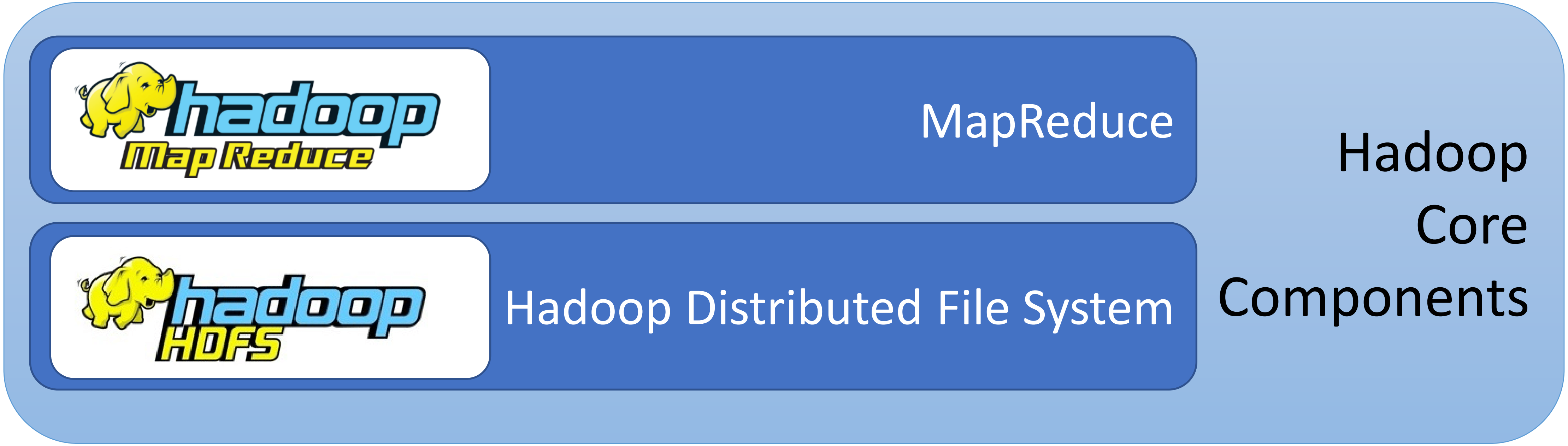
- Support partial failure
  - Failure of a component will not cause complete failure of the entire system
- Data recoverability
  - Automatic restart task and recover lost data, or better – no lost of data even when there is a component failure
- Components can "come and go"
  - No restart of system is required
- Consistency
  - Outcome of the job is not affected even if there is a component failure
- Scalability
  - Additional load will not bring down the system
  - Increase resources can be added flexibly

hadoop

A little history

- Hadoop is based on Google File System in 2003, later MapReduce in 2004
- Core concept:
  - distribute data across the nodes in a distributed system
  - processing is only done on local machine/node itself
  - Programming – high-level code
  - Communication between nodes – minimal
  - Replication of data – redundancy to improve availability and reliability

Hadoop Core Components



MapReduce

Hadoop Distributed File System

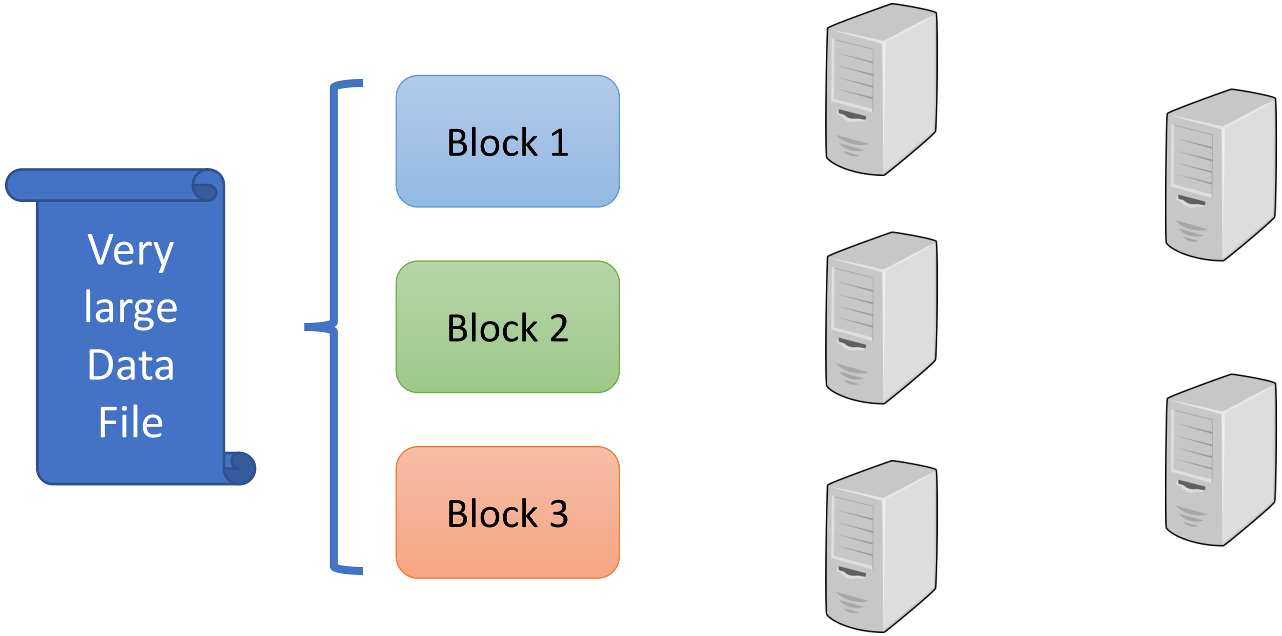
Hadoop Core Components

Hadoop Distributed File System

- Based on a presented white paper on Google File System (GFS)
- Initially developed as a storage infrastructure for Apache Nutch web search engine project
- Some characteristics:
  - Extremely fault-tolerant
  - Can hold large number of datasets
  - Provides redundant storage for massive amounts of data

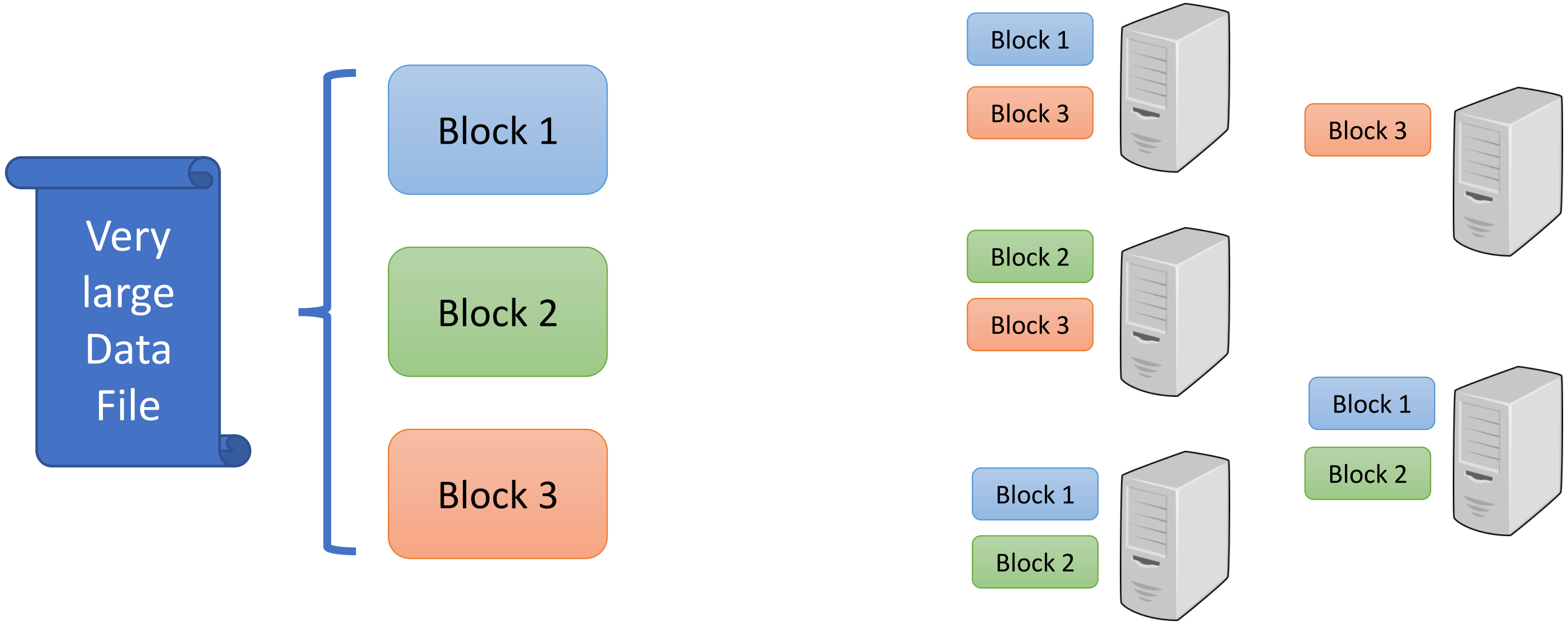
How files are stored in HDFS

- Files are split into blocks of 64MB or 128MB (Typically)
- Data is distributed across many machines at load time
  - The blocks will be stored on at least 3 machines across the cluster
  - Provides local processing, especially for efficient MapReduce processing



Forward School

Distribution of data



Very large Data File

Block 1

Block 2

Block 3

Block 1

Block 3

Block 3

Block 2

Block 3

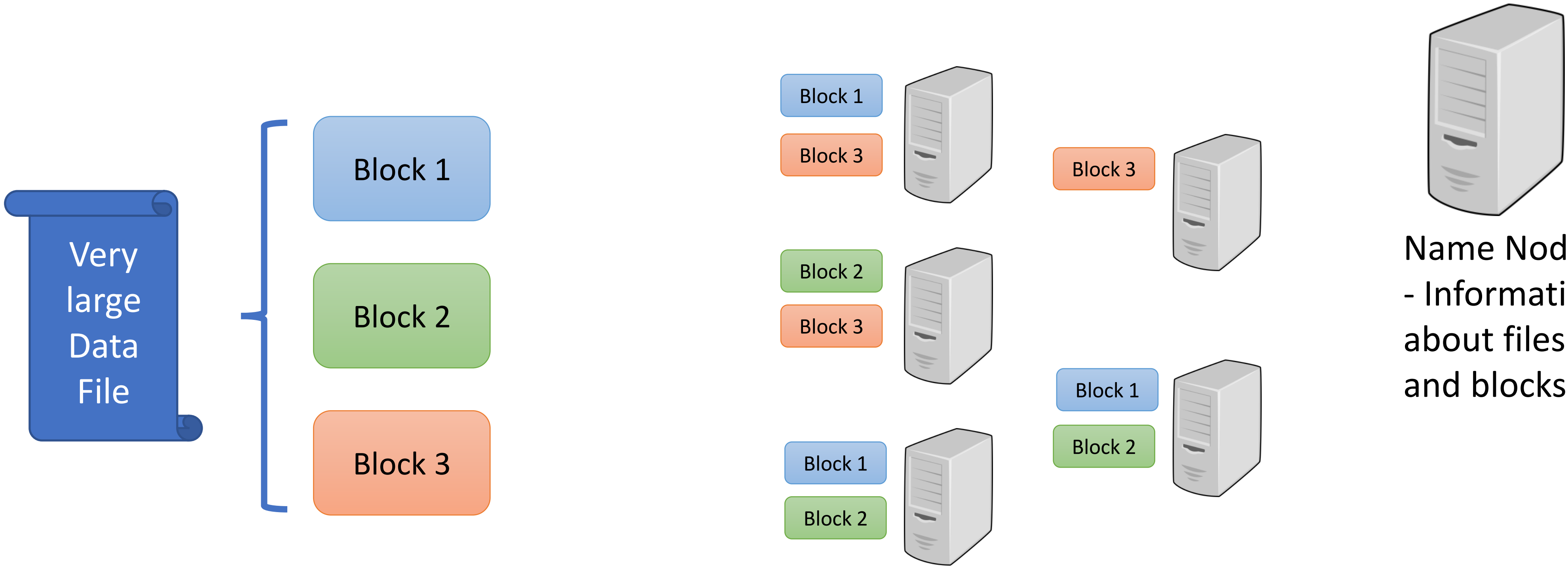
Block 1

Block 2

Block 1

Block 2

Name Node to manage metadata



Name Node
- Information about files and blocks

File access

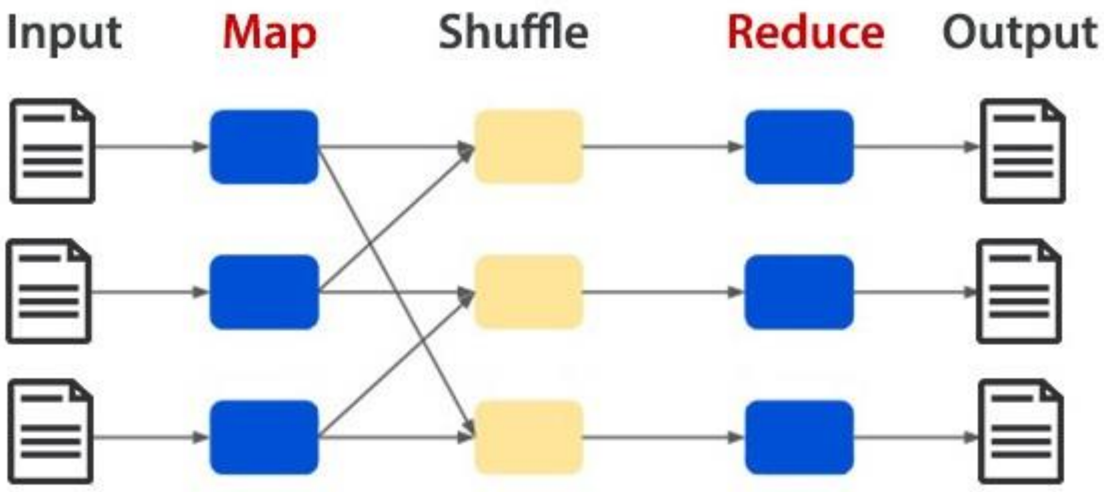
- To users, they are just access a file
  - Via FsShell command (hadoop fs)
  - Java API
  - Ecosystem projects (Flume, Sqoop and Hue)
- Hadoop/HDFS manages the access to the respective blocks

Hadoop MapReduce

- Hadoop's implementation of MapReduce
- MapReduce
  - A method for distributing a task across multiple nodes in a cluster
  - Each node processes data (remember the blocks?) stored at that particular node
  - Components - Mapper, Reducer and Shuffle and Sort
  - Terminology
    - A job – a full programme
    - A task – execution of a task (a single Mapper/Reducer) on a block
    - A task attempt – a particular instance of an attempt to execute a task

Image from: https://bigdatapath.wordpress.com/2019/03/06/hadoop-mapreduce-framework/

Hadoop MapReduce



- Mapper
  - Process data on a single HDFS block at the node where the block is stored
  - Usually the first part of the manipulation of <key,value> in a certain processing needs
- Shuffle and sort
  - Sorts and consolidates intermediate data from all mappers
- Reducer
  - Process data from Shuffle and Sort to produce the final output (second part of the processing/manipulation of data)

Forward School

Image from: https://bigdatapath.wordpress.com/2019/03/06/hadoop-mapreduce-framework/

A word count example

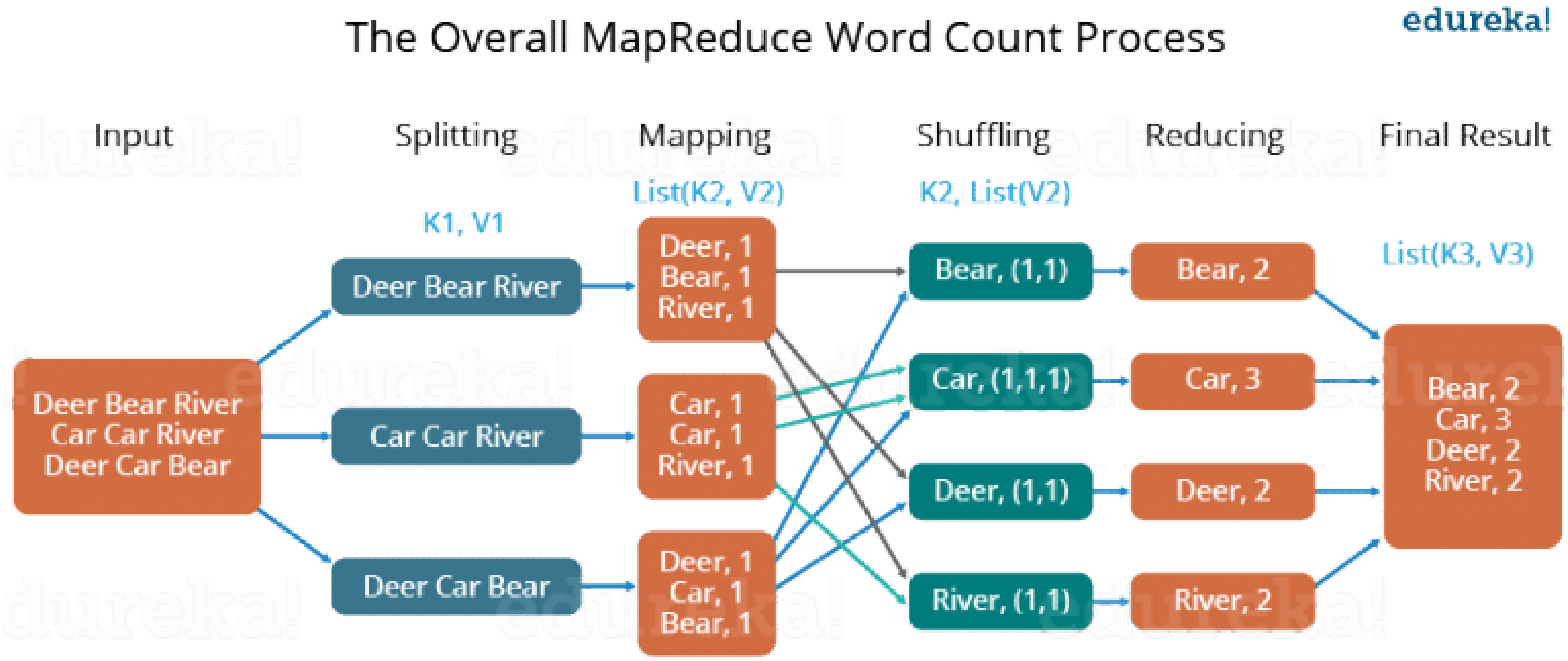
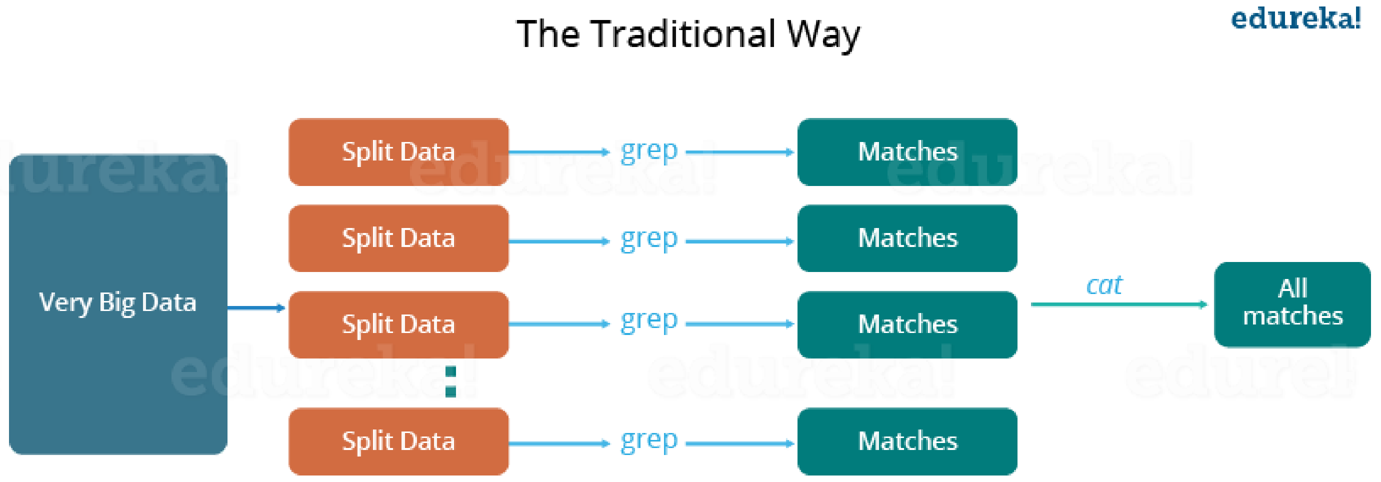
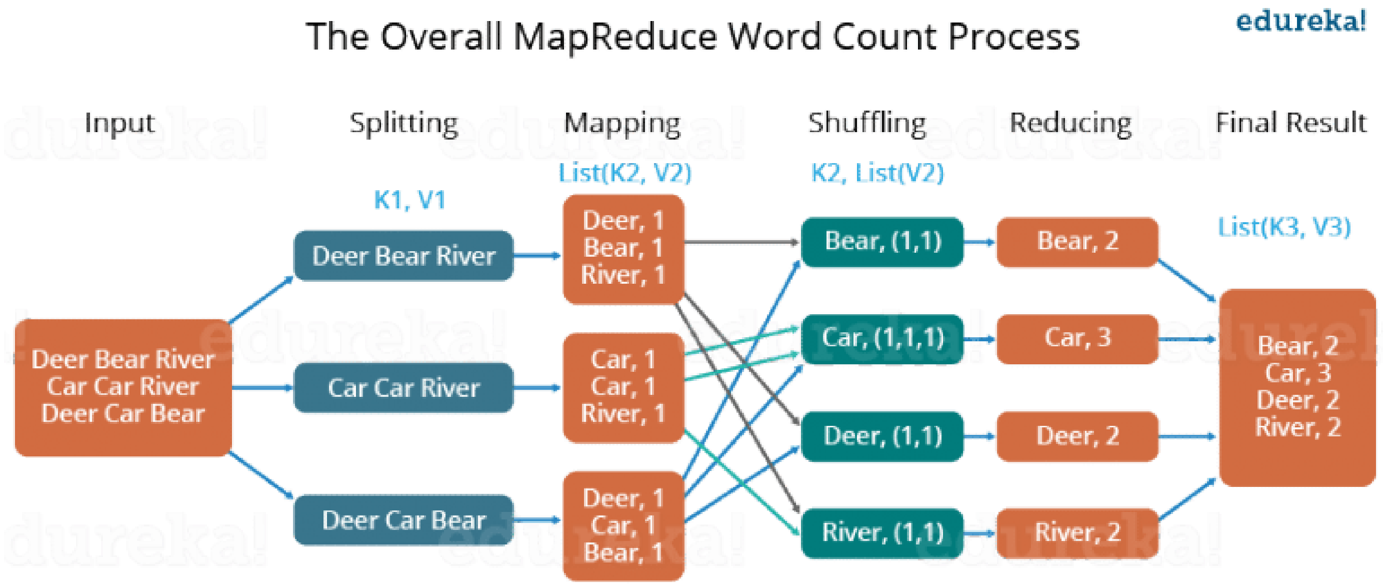


Image from: https://medium.com/edureka/mapreduce-tutorial-3d9535ddbe7c

Forward School

Comparison





Forward School

Image from: https://medium.com/edureka/mapreduce-tutorial-3d9535ddbe7c

If word count is boring….

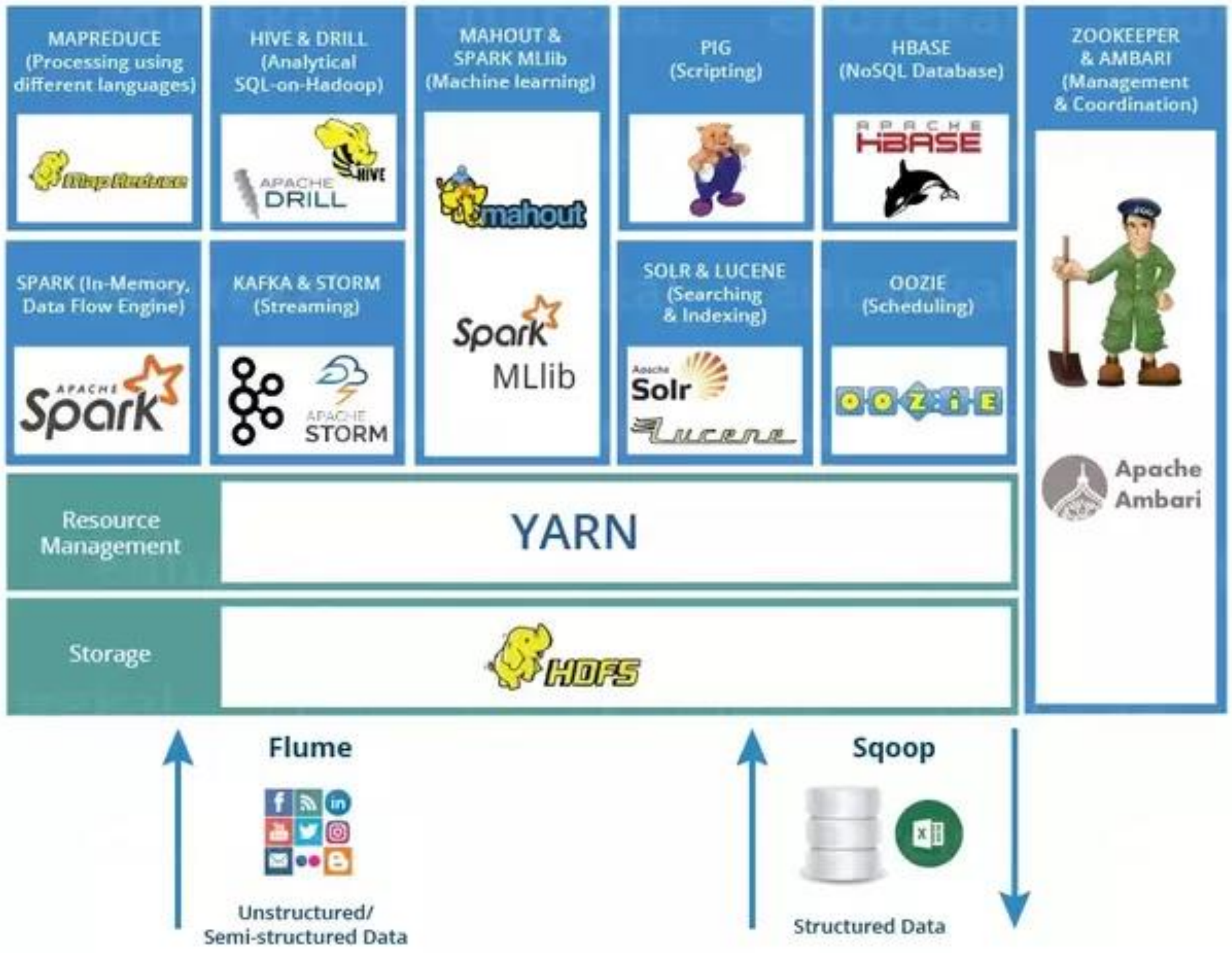
- Processing web log
- Find Top Ten records
- Find distinct values

Hadoop Ecosystem

Forward School

Ecosystem….

- Built on HDFS
- Built on HDFS and MapReduce
- Designed to integrate with or support Hadoop


MAPREDUCE (Processing using different languages)
MapReduce
HIVE & DRILL (Analytical SQL-on-Hadoop)
HIVE
APACHE DRILL
MAHOUT & SPARK MLlib (Machine learning)
mahout
Spark MLlib
PIG (Scripting)
HBASE (NoSQL Database)
APACHE HBASE
ZOOKEEPER & AMBARI (Management & Coordination)
SPARK (In-Memory, Data Flow Engine)
APACHE Spark
KAFKA & STORM (Streaming)
APACHE STORM
SOLR & LUCENE (Searching & Indexing)
Apache Solr
Lucene
OOZIE (Scheduling)
OOZIE
Apache Ambari
Resource Management
YARN
Storage
HDFS
Flume
Unstructured/ Semi-structured Data
Sqoop
Structured Data

Forward School

Data Storage: HBase

- Hadoop Database
- NoSQL Datastore
- Can store more than Petabytes of data (Massive amount of data)
- High write throughput rate
- Handles sparse data well – no wasted spaces for empty columns in a row
- Limitations
  - Only optimized for row look up by key – no FULL queries like SQL
  - No transaction: single row operations only
  - Only the key is indexed

Comparison with Traditional RBDMS

	RDBMS	Hbase
Data layout	Row-oriented	Column-oriented
Transactions	Yes	Single row only
Query language	SQL	Get/put/scan
Security	Authentication/Authorization	Kerberos
Indexes	Any column	Only row-key
Max data size	TBs	PB+
Read/write throughput (queries per second)	Thousands	Millions

When you should use Hbase

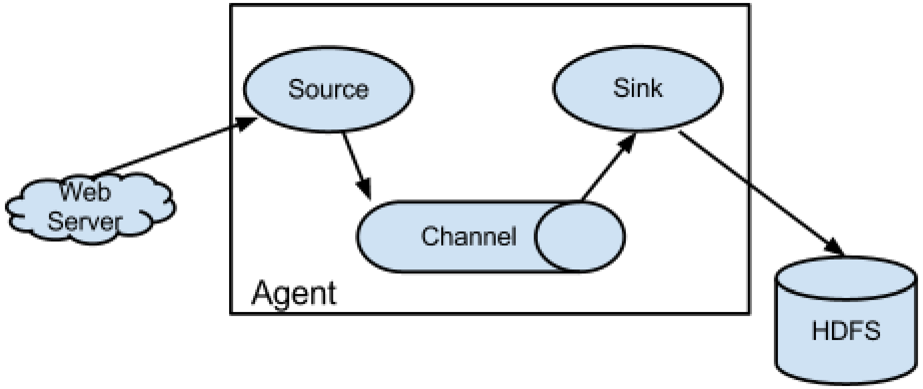
- Use HDFS if
  - You only append to your dataset (no random write)
  - You usually read the whole dataset (no random read)
- Use HBase if
  - You need random write and/or read
  - You do thousands of operations per second on TB+ of data
- Use an RDBMS if
  - Your data fits on one big node
  - You need full transaction support
  - You need real-time query capabilities

Data Integration: Flume

- What is Flume?
  - A service to move large amounts of data in real time
  - Example: storing log files in HDFS
- Flume imports data into HDFS as it is generated
  - Instead of batch-processing it later
  - For example, log files from a Web server
- Flume is
  - Distributed
  - Reliable and available
  - Horizontally scalable
  - Extensible

Flume – High level overview

- Source may be files, logs, stdout or custom
- Scalable throughput to write in parallel
- Store in any format
  - Text, compressed, binary or custom



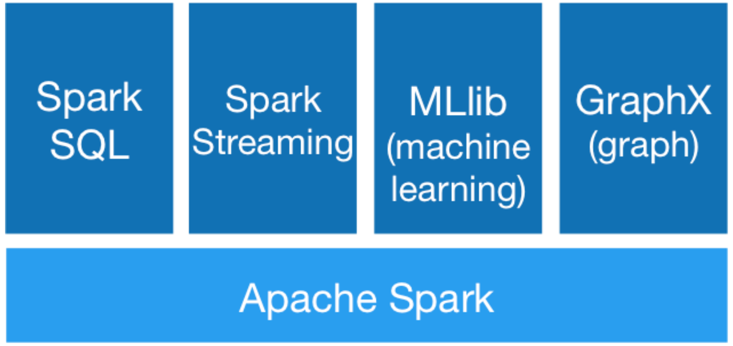
Data Integration: Sqoop

- Exchanging data with RDBMS
- Sqoop transfer data between RDBMS and HDFS very efficiently
- Supports JDBC, ODBC, and other specific databases
- Custom connectors
  - MySQL, Postgres, Netezza, Teradata, Oracle
- Not open source but free to use

Forward School

Data Processing: Spark

- a unified analytics engine for large-scale data processing
- Sparks demonstrates high performance for batch and streaming data
- Uses DAG scheduler
- Supports Java, Scala, Python, R and SQL
- Runs on Hadoop, and more – Apache Mesos, Kubernetes, standalone



Forward School

Spark

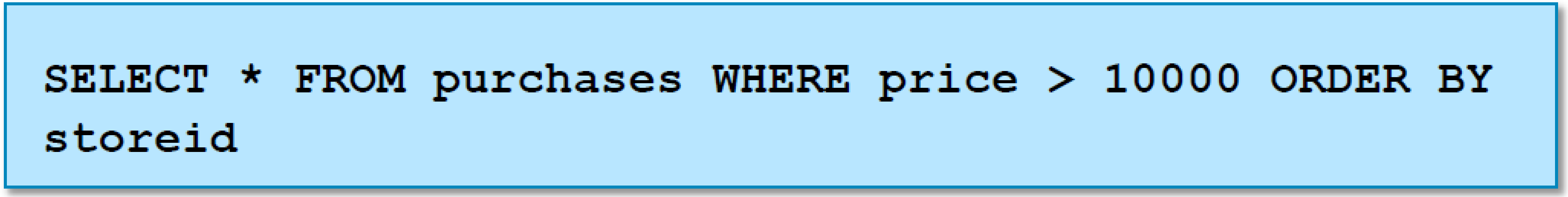
- Originally developed in UC Berkely’s AMPLab
- Benefits over MapReduce
  - Speed – way faster than MapReduce
  - Better suited for iterative algorithms
    - Can hold intermediate data in RAM, resulting in much better performance
  - Easier API
  - Supports real-time streaming data processing

Data Analysis: Hive, Pig and Impala

- MapReduce is powerful, but hard to code/master
- High level programming to perform MapReduce
  - Hive and Pig – Languages for querying and manipulating data
  - Support/leverage on existing skillsets
    - SQL
    - Programmers
- Open source Apache projects
- Interpreter turns queries into MapReduce jobs

Hive

- HiveQL – An SQL-like interface to Hadoop/MapReduce

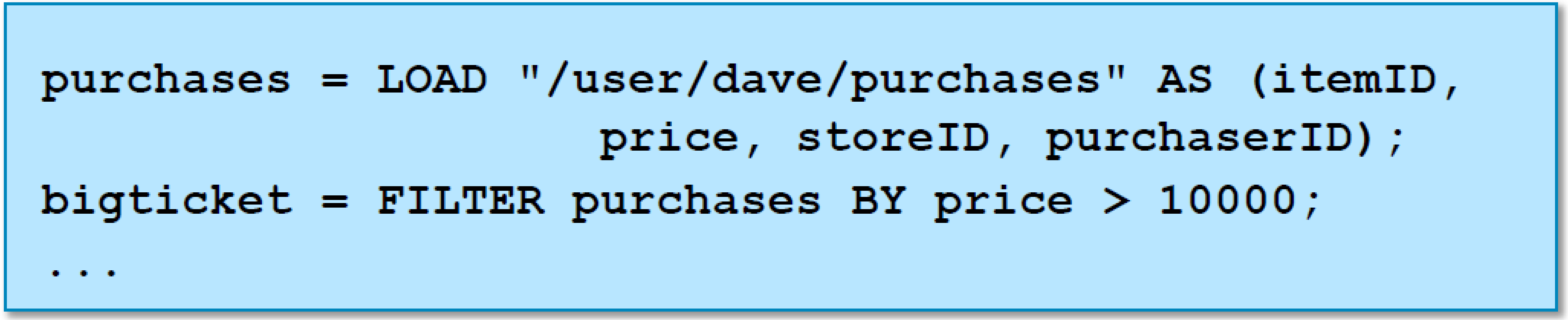


```
SELECT * FROM purchases WHERE price > 10000 ORDER BY
storeid
```

Pig

Apache Pig

- A scripting dataflow language (called Pig Latin) for transforming large data sets



```
purchases = LOAD "/user/dave/purchases" AS (itemID,
                   price, storeID, purchaserID);
bigticket = FILTER purchases BY price > 10000;
...
```

Comparison

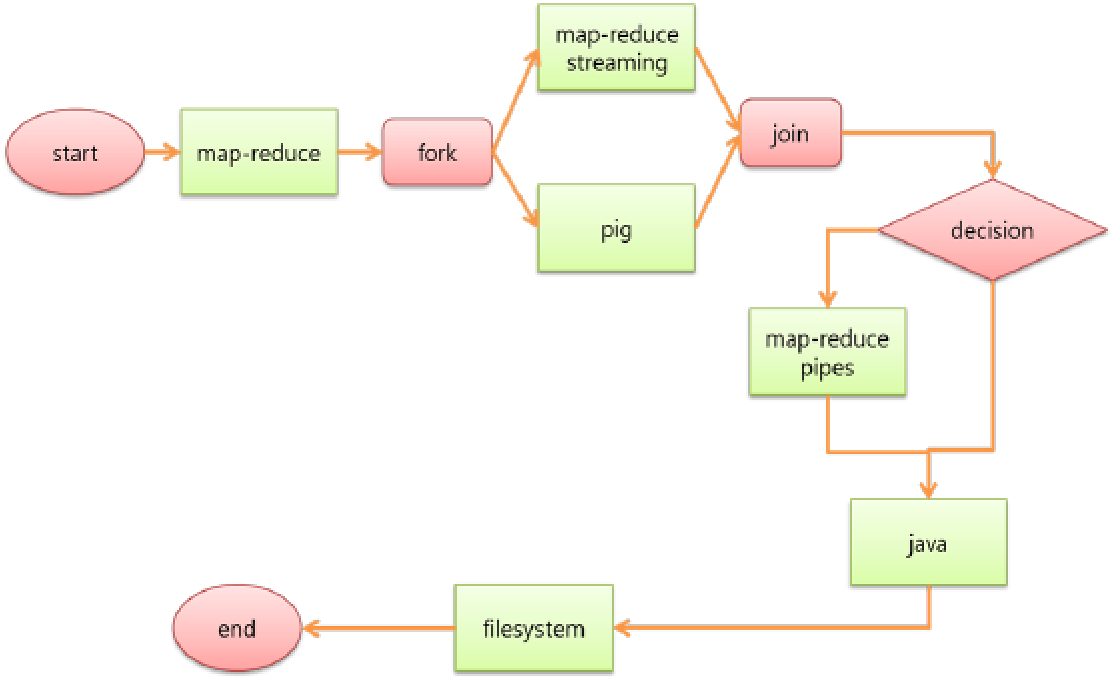
	Hive	Pig
Language	HiveQL (SQL-like)	Pig Latin (dataflow language)
Schema	Table definitions stored in a metastore	Schema optionally defined at runtime
Programmatic access	JDBC, ODBC	PigServer (Java API)

Impala

- High performance SQL engine for vast amounts of data
  - Similar query language to HiveQL
  - 10 to 50+ times faster than Hive, Pig or MapReduce
- Impala runs on Hadoop clusters
  - Data stores in HDFS
  - Does not use MapReduce
- 100% opensource but developed by Cloudera

Workflow Engine: Oozie

- Workflow engine for MapReduce jobs
- Defines dependencies between jobs
- Ensure jobs are submitted in the correct sequences



Machine Learning: Mahout

- Mahout is a Machine Learning library written in Java
- Use for
  - Collaborative filtering
  - Clustering
  - Classification
- Why use Hadoop for Machine Learning?
  - Because of the ability to store large amount of data – and the machine learning outcome may have the advantage