Big Data Technologies

Hadoop and its Ecosystem

Hala El-Ali
EMIS/CSE 8331
SMU
helali@smu.edu

Agenda

- Introduction
- Hadoop Core
- Demo
- Hadoop Ecosystem
- Demo
- QA

Big Data

- **Big data** is the term for a collection of **structured** and **unstructured** data sets so **large** and **complex** that it becomes **difficult to process** using on-hand database management tools or traditional data processing applications.
- Big Data 3 Vs
  - Volume: large amount of data
  - Velocity: needs to be analyzed quickly
  - Variety : structured and unstructured

Scale of Big Data

Bytes (8 Bits)

Kilobyte (1000 Bytes)

Megabyte (1 000 000 Bytes)

Gigabyte (1 000 000 000 Bytes)

Terabyte (1 000 000 000 000 Bytes) ← **Traditional tech.**

Petabyte (1 000 000 000 000 000 Bytes) ← Hadoop

Exabyte (1 000 000 000 000 000 000 Bytes)

Zettabyte (1 000 000 000 000 000 000 000 Bytes)

Yottabyte (1 000 000 000 000 000 000 000 000 Bytes)

The Perfect Storm (2000's)

- Data is growing at exponential rates:
  - Generated in many ways (social media, consumer transactions, scientific data, etc.)
  - Acquired in many ways (GPS, sensors, scanners, etc.)
- The demand for "knowledge" is growing at the same rate.
  - IN EVERY ASPECT!!!
  - Need to process data in real-time
- Hardware performance growth slowed:
  - CPU performance growth hit a wall
  - Storage density continues to grow, but linearly
  - Networks continue to be a bottleneck
- Hitting technical limits created a big push to parallel process and distribute

What is Hadoop and Why it Matters?

- A programming framework for Big Data that is:
  - Distributed (runs on master-slave cluster)
  - Scalable (1000's of nodes)
  - Fault-Tolerant (built-in redundancy)
  - Cost-efficient (commodity hardware)
  - Open-Source (Apache projects)
  - Free (Apache License 2.0)
- Used and supported by major corporation (Google, Yahoo!, IBM, ebay, facebook, etc.)
- Commercial distributions from companies like Cloudera and Hortonworks.
- **The World's de facto enterprise-viable Big Data solution**.

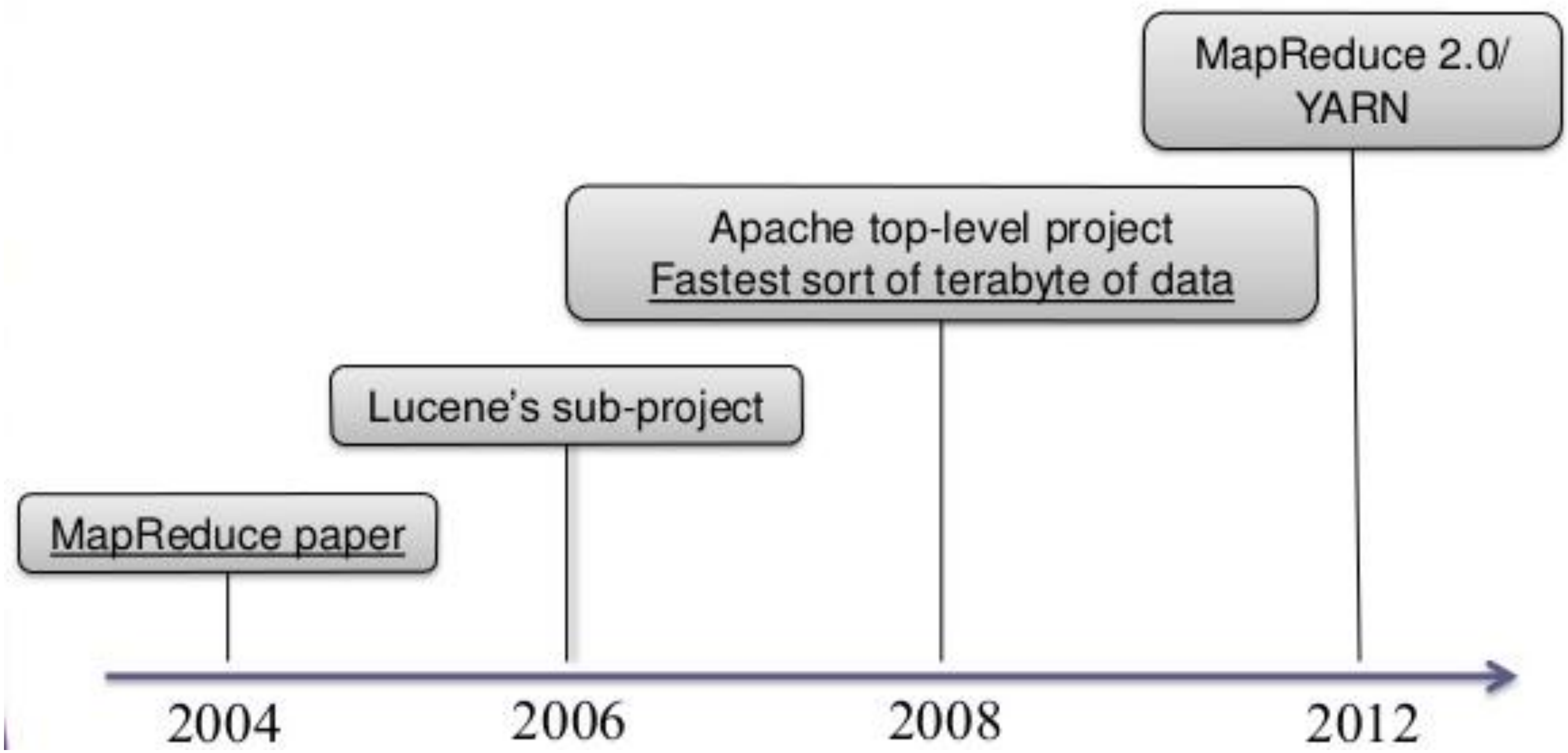
How it does it?

- Distribute:
  - Scale-out compute and storage using clusters of inexpensive commodity hardware.
  - Provide a platform for highly-available distributed storage (Hadoop Distributed File System)
  - Provide a platform for highly-available distributed compute (MapReduce).
- Localize (reduce reliance on networks)
  - Reduce network traffic by moving process where the data is.

Brief History

- First the web was invented, then searching the web became an idea.
- The web crawlers were invented and automated search (search engines).
- Nutch project: crawl and index on multi-machines. Google was working on the same idea.
- Nutch split into crawler (Nutch) and distributed computing and processing (Hadoop).
- 2004 Google published white papers on Google File System, Map-Reduce, and Big Table.
- Hadoop is an implementation of all three.

Really Brief History



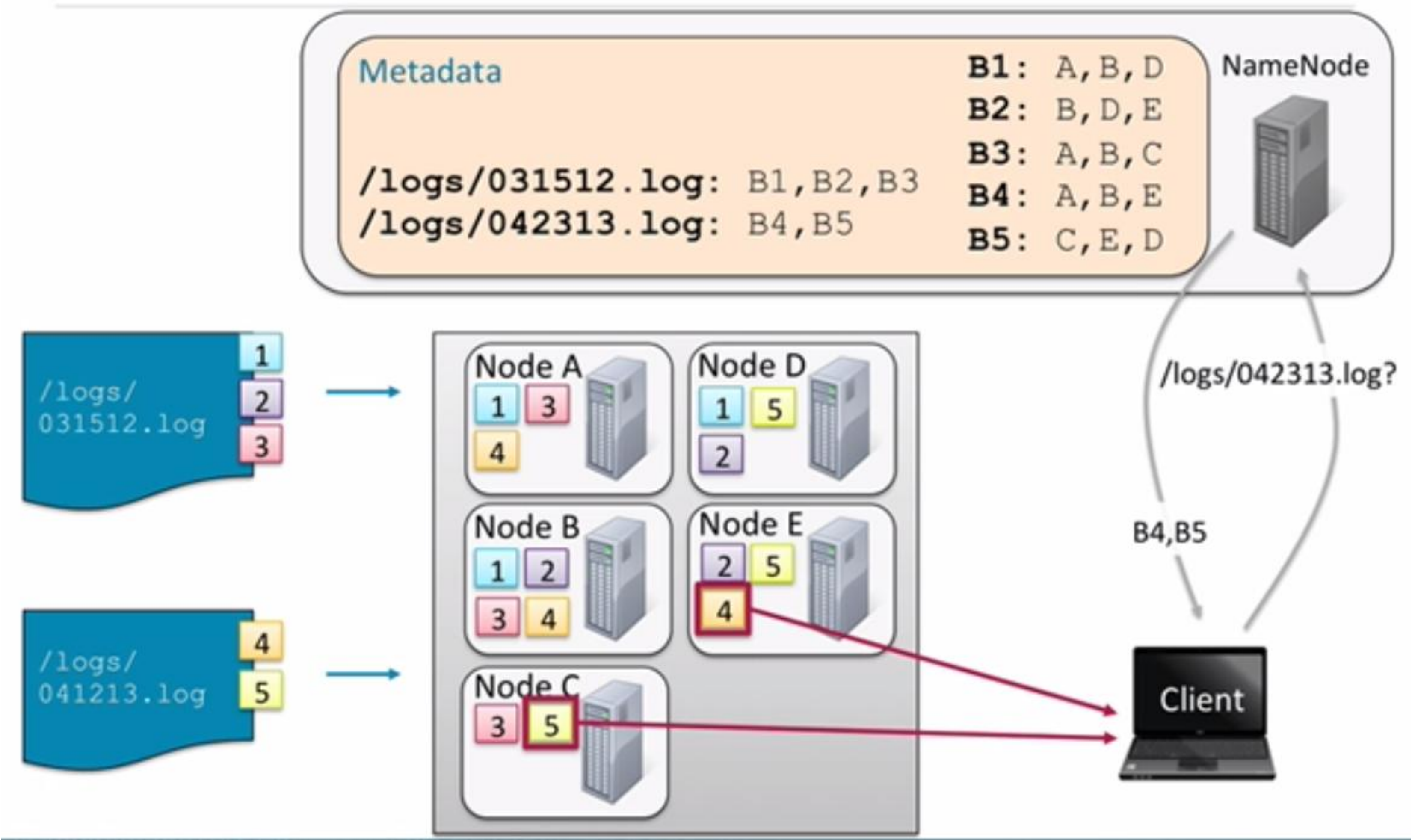
Hadoop Core

- **HDFS** Hadoop Distributed File System (follows Google File System)
- **Hadoop MapReduce**: an implementation of MapReduce programming model
- **Yarn (Hadoop 2.0)** A framework for job scheduling and cluster resource management. Yarn can be used for other than MapReduce jobs.

HDFS

- Hadoop Distributed File System that runs of file systems
- Large files are split into blocks or partitions
- Default is to store 3 copies of each partition. Two local copies and 1 remote copy (rack or subnet aware).
- Master-slave architecture
- Name Node (Master): manages files and blocks and runs on master node. Stores the metadata of the HDFS. One Name Node per cluster.
- Data Node (Slave): stores blocks and runs on slave nodes. There can be thousands of those.
- Hadoop 2 added Backup Node and Checkpoint Node (replaces Secondary Node).
- Clients access HDFS via API or command line.

HDFS



Metadata
/logs/031512.log: B1,B2,B3
/logs/042313.log: B4,B5
B1: A,B,D
B2: B,D,E
B3: A,B,C
B4: A,B,E
B5: C,E,D
NameNode
/logs/031512.log
/logs/041213.log
Node A
Node B
Node C
Node D
Node E
/logs/042313.log?
B4,B5
Client

HDFS Commands

[-appendToFile <localsrc> … <dst>]

[-cat [-ignoreCrc] <src> …]

[-checksum <src> …]

[-chgrp [-R] GROUP PATH…]

[-chmod [-R] <MODE[,MODE]… | OCTALMODE> PATH…]

[-chown [-R] [OWNER][:[GROUP]] PATH…]

[-copyFromLocal [-f] [-p] <localsrc> … <dst>]

[-copyToLocal [-p] [-ignoreCrc] [-crc] <src> … <localdst>]

[-count [-q] <path> …]

[-cp [-f] [-p] <src> … <dst>]

[-createSnapshot <snapshotDir> [<snapshotName>]]

[-deleteSnapshot <snapshotDir> <snapshotName>]

[-df [-h] [<path> …]]

[-du [-s] [-h] <path> …]

[-expunge]

[-get [-p] [-ignoreCrc] [-crc] <src> … <localdst>]

[-getfacl [-R] <path>]

[-help [cmd …]]

[-ls [-d] [-h] [-R] [<path> …]]

[-mkdir [-p] <path> …]

[-moveFromLocal <localsrc> … <dst>]

[-moveToLocal <src> <localdst>]

[-mv <src> … <dst>]

[-put [-f] [-p] <localsrc> … <dst>]

[-renameSnapshot <snapshotDir> <oldName> <newName>]

[-rm [-f] [-r|-R] [-skipTrash] <src> …]

[-rmdir [--ignore-fail-on-non-empty] <dir> …]

[-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]|[--set <acl_spec> <path>]]

[-setrep [-R] [-w] <rep> <path> …]

[-stat [format] <path> …]

[-tail [-f] <file>]

[-test -[defsz] <path>]

[-text [-ignoreCrc] <src> …]

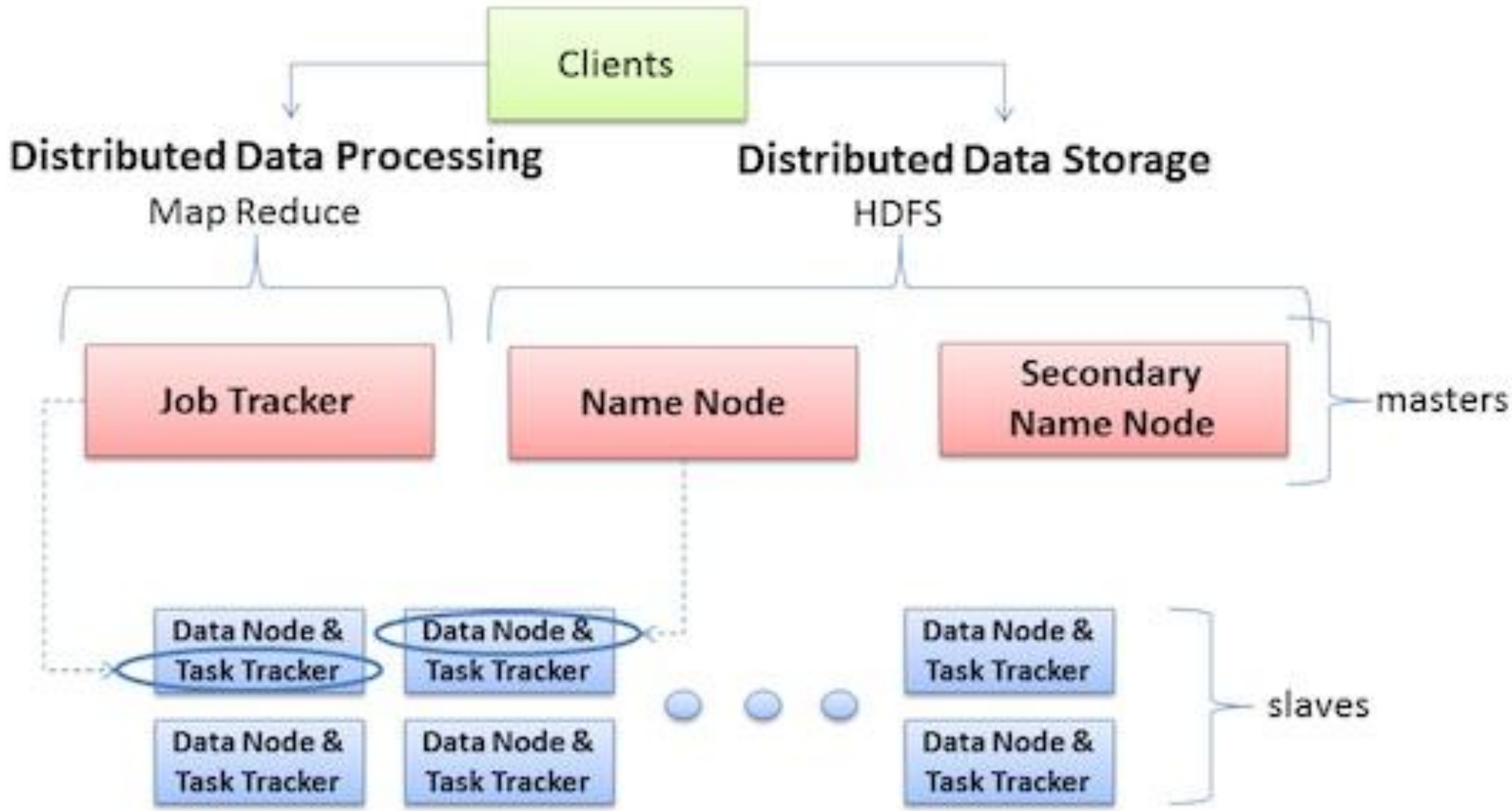
[-touchz <path> …]

[-usage [cmd …]]

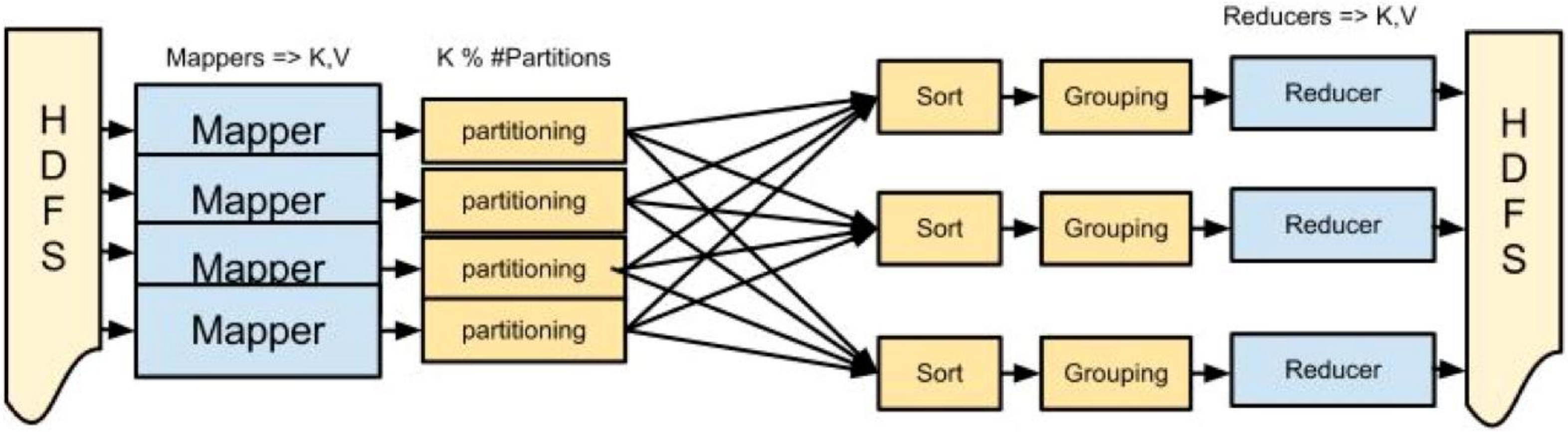
MapReduce

- Programming model for distributed processing.
- MapReduce engine consists of JobTracker and TaskTracker
- JobTracker runs on master node, client submit jobs to it. It pushes work to TaskTracker. Rack aware, tries to assign tasks to node that contains the data or a node near it.
- TaskTracker: runs on slave node, spawns a JVM for each task, holds 'available' slots where tasks are pushed by the JobTracker. Speculative scheduling for slow running TaskTracker.
- Scheduling: Default is FIFO. Fair and Capacity Scheduling was added as options.

MapReduce


Clients
Distributed Data Processing
Map Reduce
Distributed Data Storage
HDFS
Job Tracker
Name Node
Secondary
Name Node
masters
Data Node &
Task Tracker
Data Node &
Task Tracker
Data Node &
Task Tracker
Data Node &
Task Tracker
Data Node &
Task Tracker
Data Node &
Task Tracker
slaves

MapReduce



The MapReduce Pipeline

A mapper receives (Key, Value) & outputs (Key, Value)
A reducer receives (Key, Iterable[Value]) and outputs (Key, Value)
Partitioning / Sorting / Grouping provides the Iterable[Value] & Scaling

Yarn

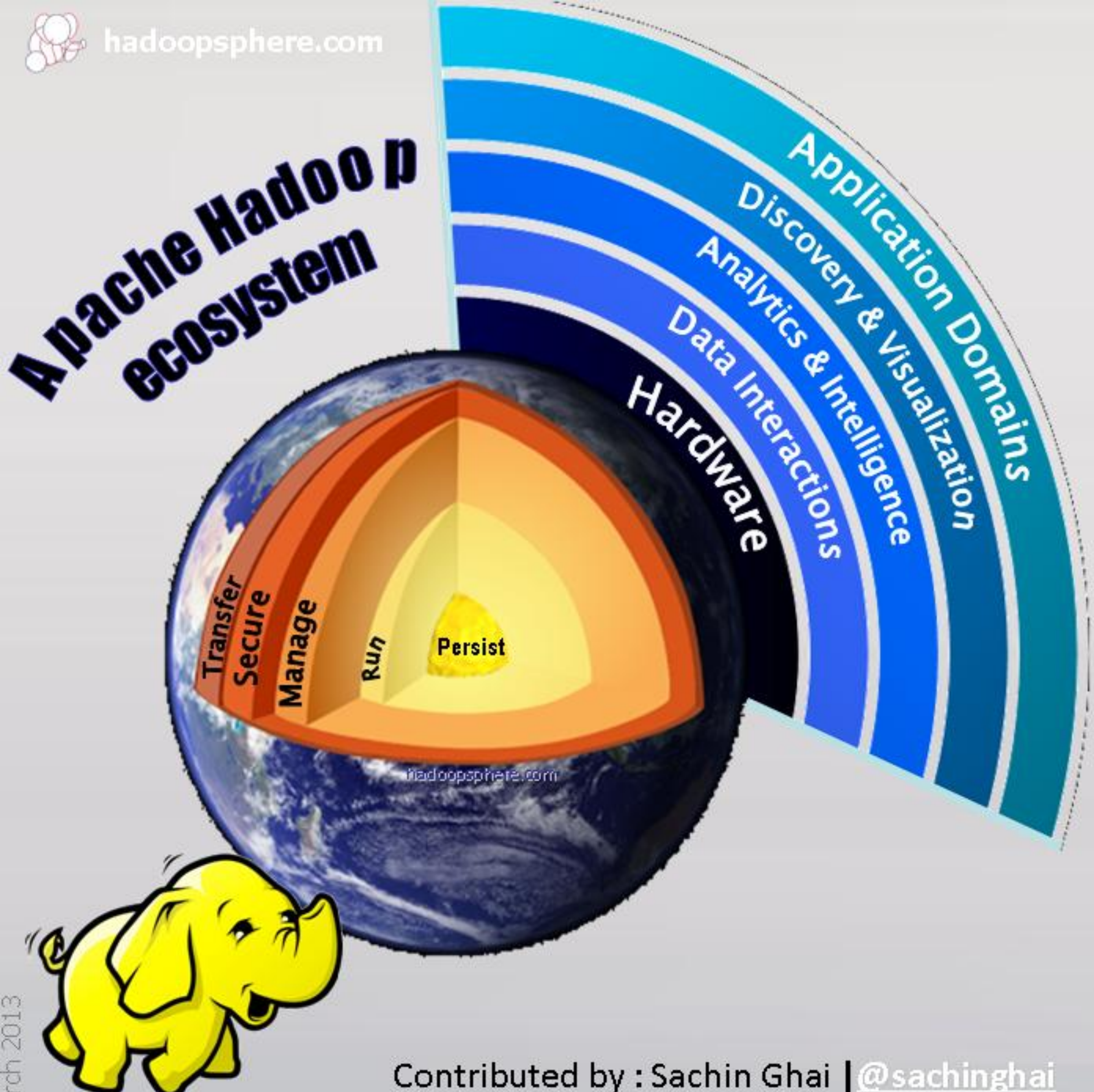
- MapReduce V2 split the JobTracker into two daemons: scheduling/monitoring and resource management.
- Yarn addresses issues with MapReduce V1.
- Yarn enables fine grained memory allocation.
- Enables larger clusters
- Yarn enables running jobs other than MapReduce on the cluster.

demo

MapReduce demo

What Hadoop is Not

- Apache Hadoop is not a substitute for a database
  - No ACID
  - No SQL
  - No real-time
- MapReduce is not always the best algorithm
  - No shared states
  - No shared memory
  - No flow, locks, dependency
- HDFS is not a complete POSIX filesystem
  - No seek to the middle and write into file



'Atmospheric' Layers	Application Domains	Distribution, Financial, Government, Heavy Industry, Internet, Oil & Energy, Research, Telecom
	Discovery & Visualization	Lucene, Blur, Giraph
	Analytics & Intelligence	Mahout, Drill
	Data Interactions	Pig, Hive, HCatalog, Tez, Gora
	Hardware (& Appliances)	Commodity H/w
'Core' Layers	Distribution	Apache
	Secure	Knox
	Manage	Oozie, Zookeeper, Crunch, MRUnit, HDT, Ambari, Vaidya, BigTop, Whirr
	Run	MapReduce, YARN, Hama
	Persist	HDFS, HBase, Cassandra, Accumulo, Avro, Trevni, Thrift
	Transfer	Flume, Sqoop, Chukwa, Kafka

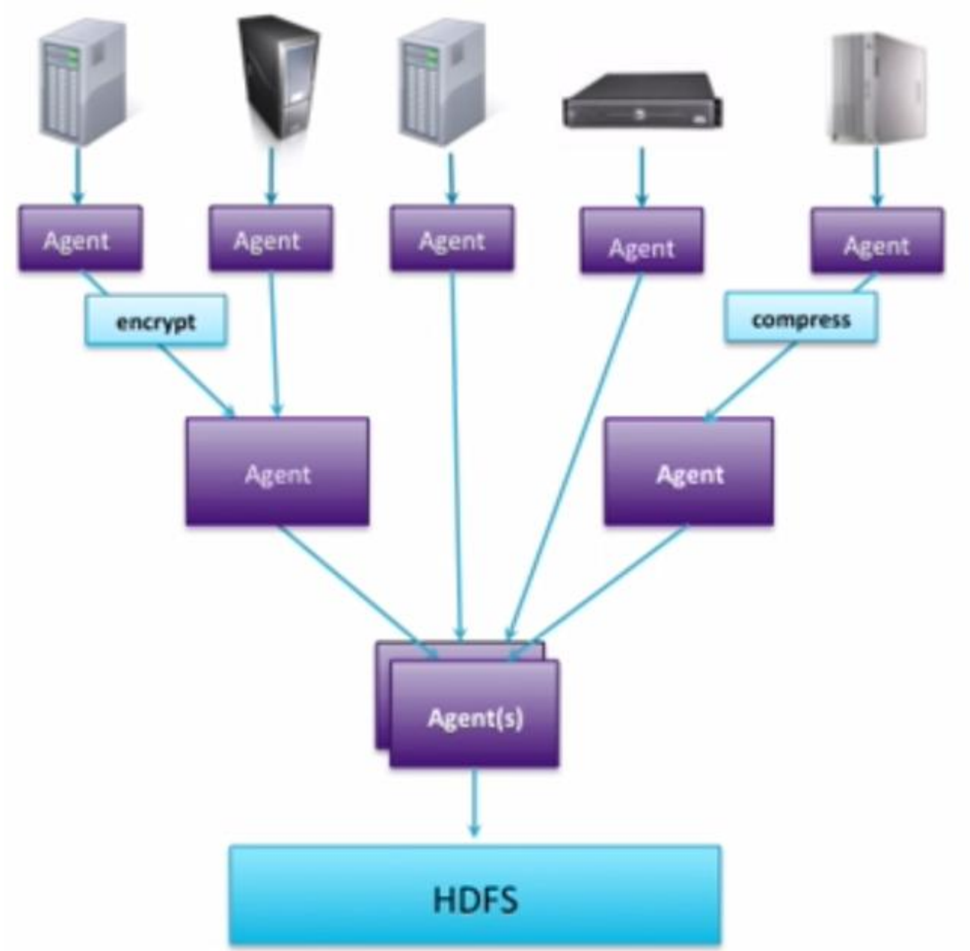
Contributed by : Sachin Ghai | @sachinghai

March 2013

Transfer – Flume

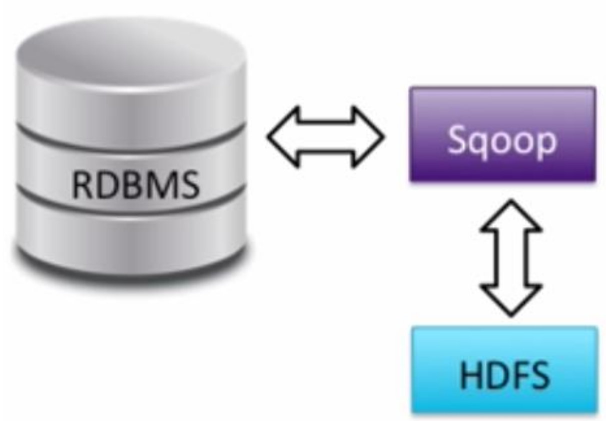
- Service for efficiently collecting, aggregating, and moving large amounts of ~~log~~ data (flume.apache.org)
- Runs in its own cluster
- Simple architecture: Source-Channel-Sink
- Supported sources are: file, logs, syslog, stdout, user-defined (events)
- Supported sinks are: HDFS, files system, user-defined (Hbase).
- Agents collect data from source computers. Data is processed (encrypted, compressed, ..).
- Agent can perform more pre-persist processing (cleaned, enriched, transformed) then parallel stored in any format (text, binary, ..).

Transfer – Flume

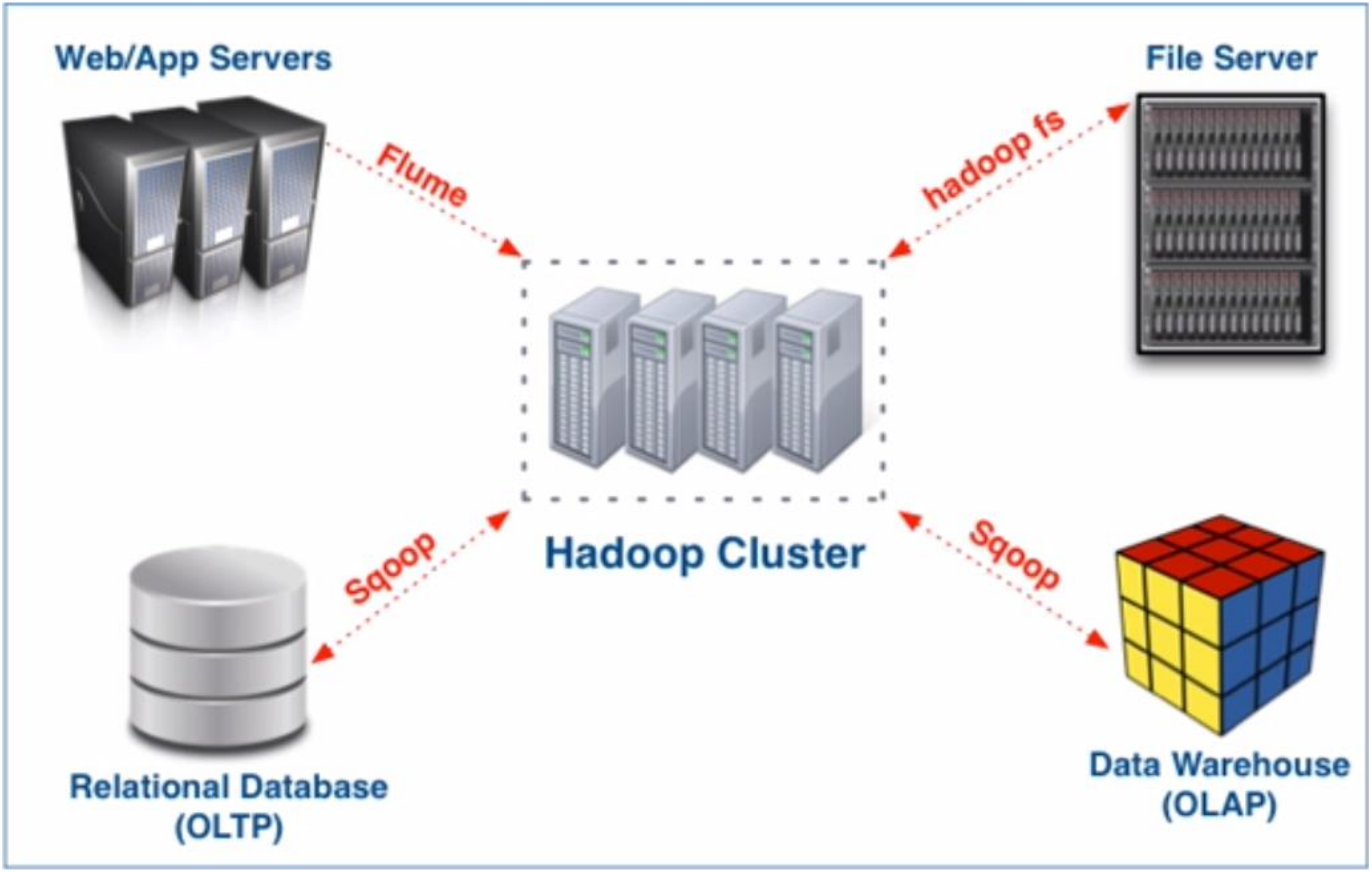


Transfer – Sqoop

- A tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases (sqoop.apache.org).
- It is capable of transferring the data in both directions.
- Supports incremental imports.
- Supports any database that supports JDBC and allows for custom connectors.
- Free but not open source.



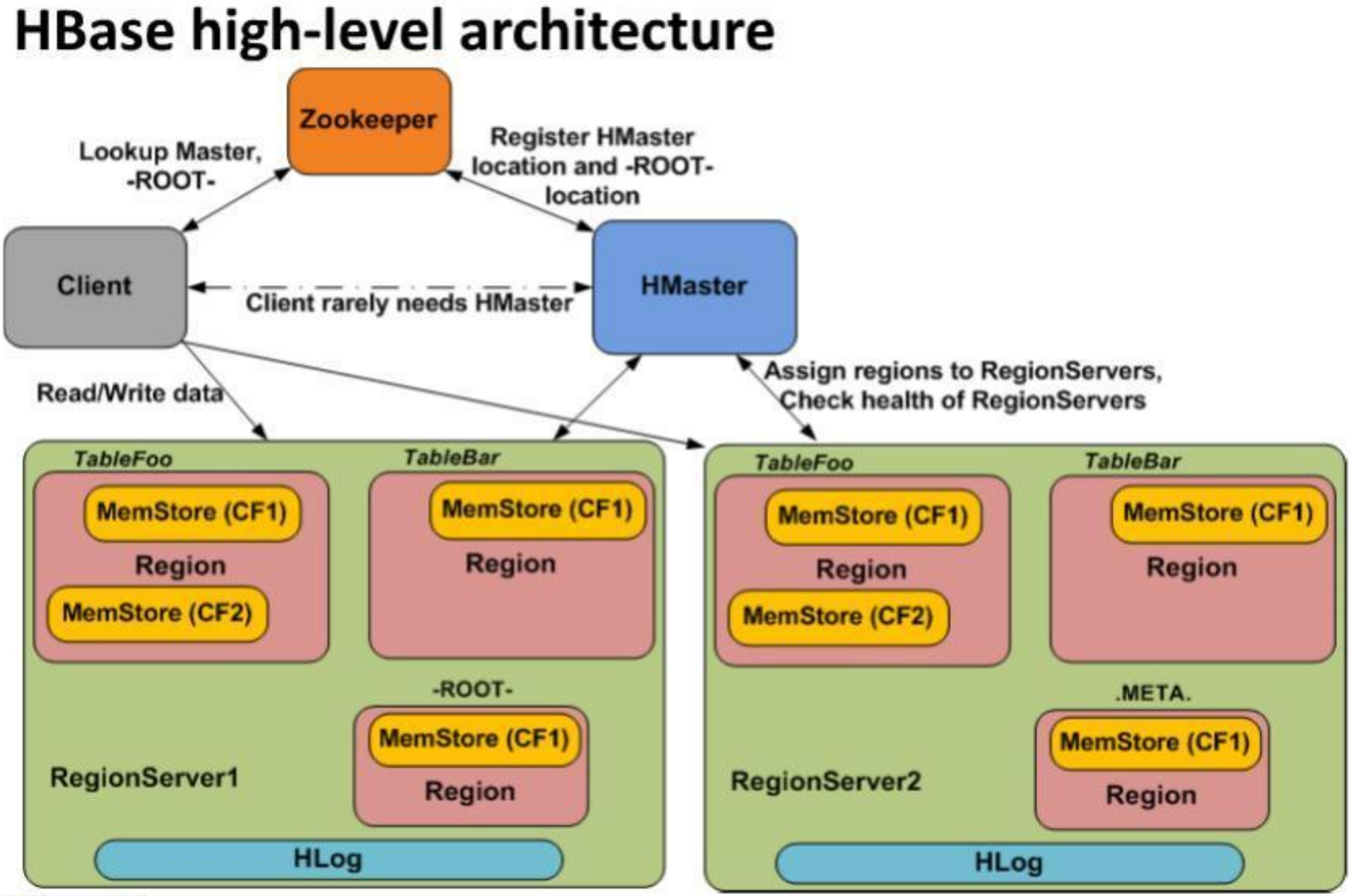
Transfer


Web/App Servers
File Server
Flume
hadoop fs
Hadoop Cluster
Sqoop
Sqoop
Relational Database
(OLTP)
Data Warehouse
(OLAP)

Persist – Hbase (Hadoop Database)

- Built on top of HDFS to provide **real-time, random read/write access** to the data.
- Modeled after Google Bigtable, suitable for massive **sparse** data.
- Column-Family based, schema consists of column-families definitions. A column-family is like a table except columns can be added without schema change. Column families are stored together on HDFS
- Access is via a primary key (row key) only, no indexes.
- Data sorted by row key and partitioned (sharded) into regions by row-key range.
- Cells are versioned usually timestamped.
- New data is held in memory, flushed to disk in files, compaction is run to remove expired and deleted cells.

Persist – Hbase Diagram



Hbase Vs. RDBMS

	RDBMS	HBase
Transactions	Yes	Single row only
Query language	SQL	get/put/scan (or use Hive or Impala)
Indexes	Yes	Row-key only
Max data size	TBs	PBs
Read/write throughput (queries per second)	Thousands	Millions

Data Interactions - Hive

- Data warehouse software facilitates **<u>querying</u>** and managing large datasets residing in distributed storage (hive.apache.org)
- Initially developed by Facebook.
- Provides a SQL-like language called HiveQL
- Command line interface, web and JDBC/ODBC.
- Runs on your computer.
- It creates MapReduce jobs in the background to execute the queries.
- Leverage SQL skills
- Reduce programming and testing time.

Data Interactions - Hive

Hive SQL Semantics
SELECT, LOAD INSERT from query
Expressions in WHERE and HAVING
GROUP BY, ORDER BY, SORT BY
Sub-queries in FROM clause
GROUP BY, ORDER BY
CLUSTER BY, DISTRIBUTE BY
ROLLUP and CUBE
UNION
LEFT, RIGHT and FULL INNER/OUTER JOIN
CROSS JOIN, LEFT SEMI JOIN
Windowing functions (OVER, RANK, etc)
INTERSECT, EXCEPT, UNION, DISTINCT
Sub-queries in WHERE (IN, NOT IN, EXISTS/ NOT EXISTS)
Sub-queries in HAVING

Color Key
Hive 0.10
Hive 0.11
FUTURE

Data Interactions – Pig

- **Apache Pig** is a platform for analyzing large data sets that consists of a **high-level language for expressing data analysis programs**, coupled with **infrastructure for evaluating** these programs. The salient property of Pig programs is that their structure is amenable to substantial **parallelization**, which in turns enables them to handle very large data sets. (pig.apache.org)
- Components are:
  - A **compiler** that generates MapReduce jobs .
  - A dataflow, script-like language for transforming large datasets called **PigLatin**.
- Interfaces are command line and Java API **PigServer**

PigLatin

- Scripting-like language
- Defines data types like primitive datatype, tuple, sets and bags.
- Provides a list of commands for transforming data.
- Each command is executed as a MapReduce job (newer version would optimize the execution of a block of commands).
- Runs on your computer but submits the execution to a hadoop cluster
- Extendible (users can define their own processing UDF) (See Piggybank for community UDF)

PigLatin

Pig Command	What it does
load	Read data from file system.
store	Write data to file system.
foreach	Apply expression to each record and output one or more records.
filter	Apply predicate and remove records that do not return true.
group/cogroup	Collect records with the same key from one or more inputs.
join	Join two or more inputs based on a key.
order	Sort records based on a key.
distinct	Remove duplicate records.
union	Merge two data sets.
split	Split data into 2 or more sets, based on filter conditions.
stream	Send all records through a user provided binary.
dump	Write output to stdout.
limit	Limit the number of records.

PigLatin

PigLatin demo (using grunt interactive shell).

Mahout

- Per http://hortonworks.com/hadoop/mahout/:

  "Mahout supports four main data science use cases:

  Collaborative filtering – mines user behavior and makes product recommendations (e.g. Amazon recommendations)

  Clustering – takes items in a particular class (such as web pages or newspaper articles) and organizes them into naturally occurring groups, such that items belonging to the same group are similar to each other

  Classification – learns from existing categorizations and then assigns unclassified items to the best category

  Frequent itemset mining – analyzes items in a group (e.g. items in a shopping cart or terms in a query session) and then identifies which items typically appear together "

Mahout

Algorithm	Category	Description
Distributed Item-based Collaborative Filtering	Collaborative Filtering	Estimates a user's preference for one item by looking at his/her preferences for similar items
Collaborative Filtering Using a Parallel Matrix Factorization	Collaborative Filtering	Among a matrix of items that a user has not yet seen, predict which items the user might prefer
Canopy Clustering	Clustering	For preprocessing data before using a K-means or Hierarchical clustering algorithm
Dirichlet Process Clustering	Clustering	Performs Bayesian mixture modeling
Fuzzy K-Means	Clustering	Discovers soft clusters where a particular point can belong to more than one cluster
Hierarchical Clustering	Clustering	Builds a hierarchy of clusters using either an *agglomerative* "bottom up" or *divisive* "top down" approach
K-Means Clustering	Clustering	Aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean
Latent Dirichlet Allocation	Clustering	Automatically and jointly cluster words into "topics" and documents into mixtures of topics
Mean Shift Clustering	Clustering	For finding modes or clusters in 2-dimensional space, where the number of clusters is unknown
Minhash Clustering	Clustering	For quickly estimating similarity between two data sets
Spectral Clustering	Clustering	Cluster points using eigenvectors of matrices derived from the data
Bayesian	Classification	Used to classify objects into binary categories
Random Forests	Classification	An ensemble learning method for classification (and regression) that operate by constructing a multitude of decision trees
Parallel FP Growth Algorithm	Frequent Itemset Mining	Analyzes items in a group and then identifies which items typically appear together

http://hortonworks.com/hadoop/mahout/

QA

References

- White, Tom. *Hadoop The Definitive Guide*. 3rd ed. O'Reilly, 2012. Print.
- Rathbone, Matthew. "A Beginners Guide to Hadoop." *Blog RSS*. 17 Apr. 2013. Web. 30 Mar. 2015. http://blog.matthewrathbone.com/2013/04/17/what-is-hadoop.html
- Allouche, Gil. "Hadoop 101: An Explanation of the Hadoop Ecosystem." *Hadoop 101: An Explanation of the Hadoop Ecosystem*. DZone, 17 Dec. 2014. Web. 30 Mar. 2015. <http://architects.dzone.com/articles/hadoop-101-explanation-hadoop>.
- Garment, Victoria. "Hadoop 101: The Most Important Terms, Explained." *Plotting Success*. 27 Mar. 2014. Web. 30 Mar. 2015. <http://www.plottingsuccess.com/hadoop-101-important-terms-explained-0314/>.
- Jobke, Morris. "Hadoop - NameNode, Checkpoint Node and Backup Node - Morris Jobke." 11 Dec. 2013. Web. 30 Mar. 2015. <http://morrisjobke.de/2013/12/11/Hadoop-NameNode-and-siblings/>.
- Li, Heifeng. "Distributed NoSQL: HBase and Accumulo." *Dataconomy*. 18 Aug. 2014. Web. 30 Mar. 2015. <http://dataconomy.com/distributed-nosql-hbase-and-accumulo/>.
- "The Zettabyte Era-Trends and Analysis." *Cisco*. Web. 30 Mar. 2015. <http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.html>.
- Sutter, Herb. "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software." *The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software*. 30 Mar. 2005. Web. 30 Mar. 2015. <http://www.gotw.ca/publications/concurrency-ddj.htm>.

References

- "Apache Mahout." *Hortonworks*. Web. 30 Mar. 2015. <http://hortonworks.com/hadoop/mahout/>.
- "Introduction to Apache Pig." *- Online Hadoop Training Video*. 4 Oct. 2011. Web. 30 Mar. 2015. <http://www.cloudera.com/content/cloudera/en/resources/library/training/introduction-to-apache-pig.html>.
- "Cloudera Essentials for Apache Hadoop." *Hadoop Essentials Training*. Web. 30 Mar. 2015. <http://www.cloudera.com/content/cloudera/en/training/library/hadoop-essentials.html>.
- Ghai, Sachin. "HadoopSphere." *Apache Hadoop Ecosystem*. Web. 30 Mar. 2015. <http://www.hadoopsphere.com/2013/03/apache-hadoop-ecosystem-march-2013_12.html>.
- "Hadoop Tutorial: Map-Reduce on YARN Part 1 -- Overview and Installati..." *Hadoop Tutorial: Map-Reduce on YARN Part 1 -- Overview and Installati...* Web. 30 Mar. 2015. <http://www.slideshare.net/martyhall/hadoop-tutorial-mapreduce-on-yarn-part-1-overview-and-installation>.
- "Cheat Sheet Hive for SQL Users." Web. 30 Mar. 2015. <http://hortonworks.com/wp-content/uploads/downloads/2013/08/Hortonworks.CheatSheet.SQLtoHive.pdf>.
- "Welcome to Apache Pig!" *Welcome to Apache Pig!* Web. 30 Mar. 2015. <https://pig.apache.org/>.
- "General." *Apache Hive TM*. Web. 30 Mar. 2015. <https://hive.apache.org/>.
- "Welcome to Apache Flume¶." *Welcome to Apache Flume — Apache Flume*. Web. 30 Mar. 2015. <https://flume.apache.org/>.
- "Sqoop -." *Sqoop -*. Web. 30 Mar. 2015. <https://sqoop.apache.org/>.