

Chapter 2: Tools and storing to Analyze Big Data

Course Title: Big Data Analysis

Specific solutions

Finance, Health, Social Media, IoT, ...

5. Analytics solutions

Analytics techniques and functions

Advanced statistics, machine learning, text processing, ...

4. Analytics tools

Data stream analytics platforms

Drools Fusion, Esper, Event Swarm, InfoSphere Streams, Azure Stream Analytics, ...

Stored data analytics platforms

Elastic Search, Pentaho, ...

3. Analytics platforms

Data stream processing platforms

Storm, Kafka, Flume, ActiveMQ, AWS Kinesis, Microsoft Azure, ...

Stored data processing platforms

Hadoop, Spark, ...

2. Big data platforms

Cloud

Azure, Amazon, ...

Enterprise distributed systems

IBM, Oracle, Tibco, ...

Others

1. Infrastructure platforms

Distributed processing is non-trivial

- How to assign tasks to different workers in an efficient way?
- What happens if tasks fail?
- How do workers exchange results?
- How to synchronize distributed tasks allocated to different workers?



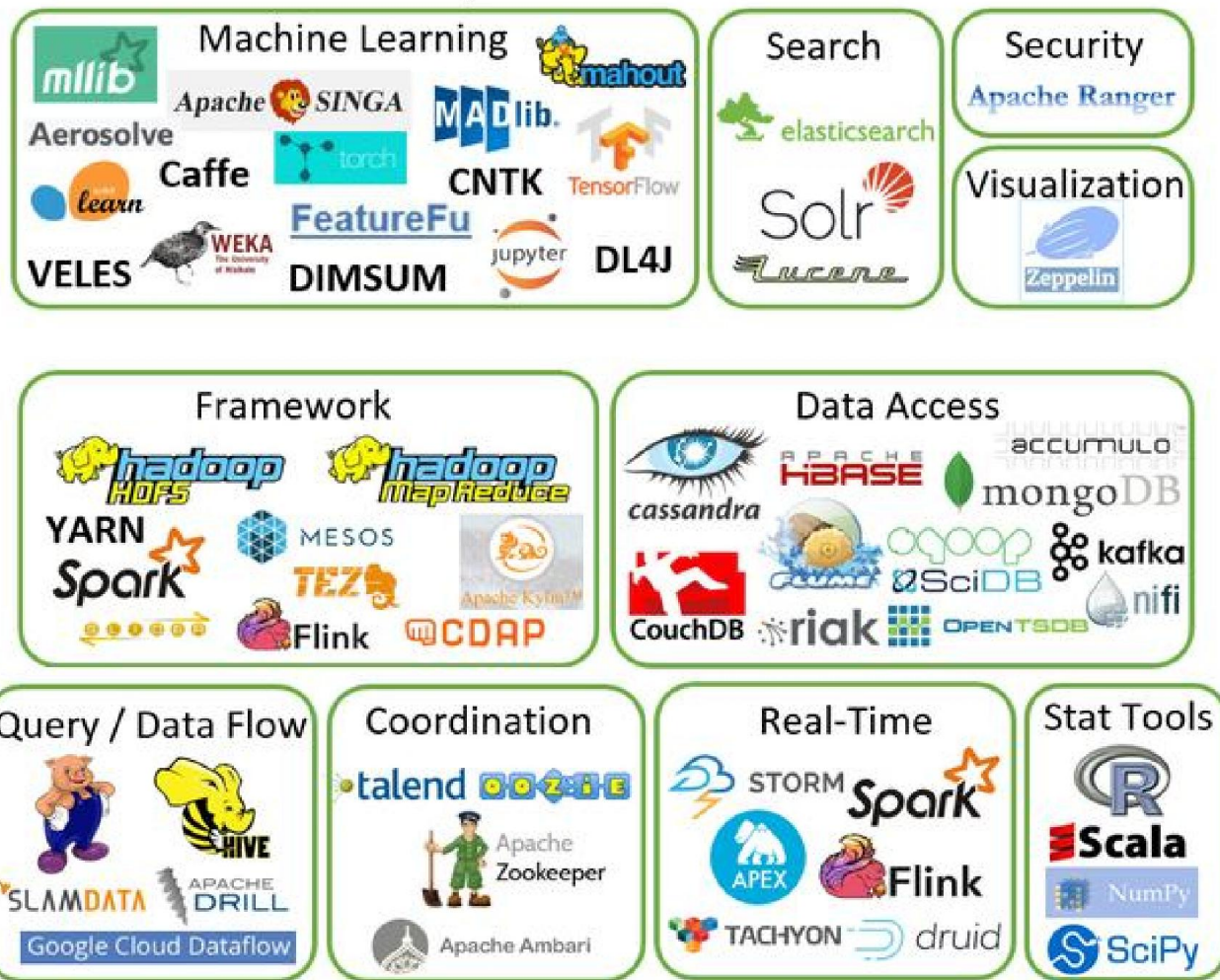
Big data storage is challenging

- Data Volumes are massive
- Reliability of Storing data is challenging
- All kinds of failures: Disk/Hardware/Network Failures
- Probability of failures simply increase with the number of machines
- Complexity of data is increased



Big Data Open Source Tools

Open Source



Tools and its applications

- Big data tools includes

- **Apache Hadoop**

- MapReduce
 - HDFS
 - HBase
 - Hive and Pig
 - Mahout
 - Spark

What is Hadoop

☰ It is an open-source data storage and processing platform

● Before the advent of Hadoop, storage and processing of big data was a big challenge.

☰ Massively scalable, automatically parallelizable

● Based on work from Google: **similarity and differences**

● Google: GFS + MapReduce + BigTable (Not open)

● Hadoop: HDFS + Hadoop MapReduce + HBase (opensource)

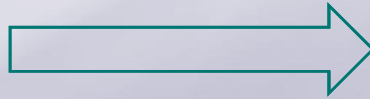
Hadoop offers

- Redundant, Fault-tolerant data storage
- Parallel computation framework
- Job coordination



Programmers

*No longer need to
worry about*



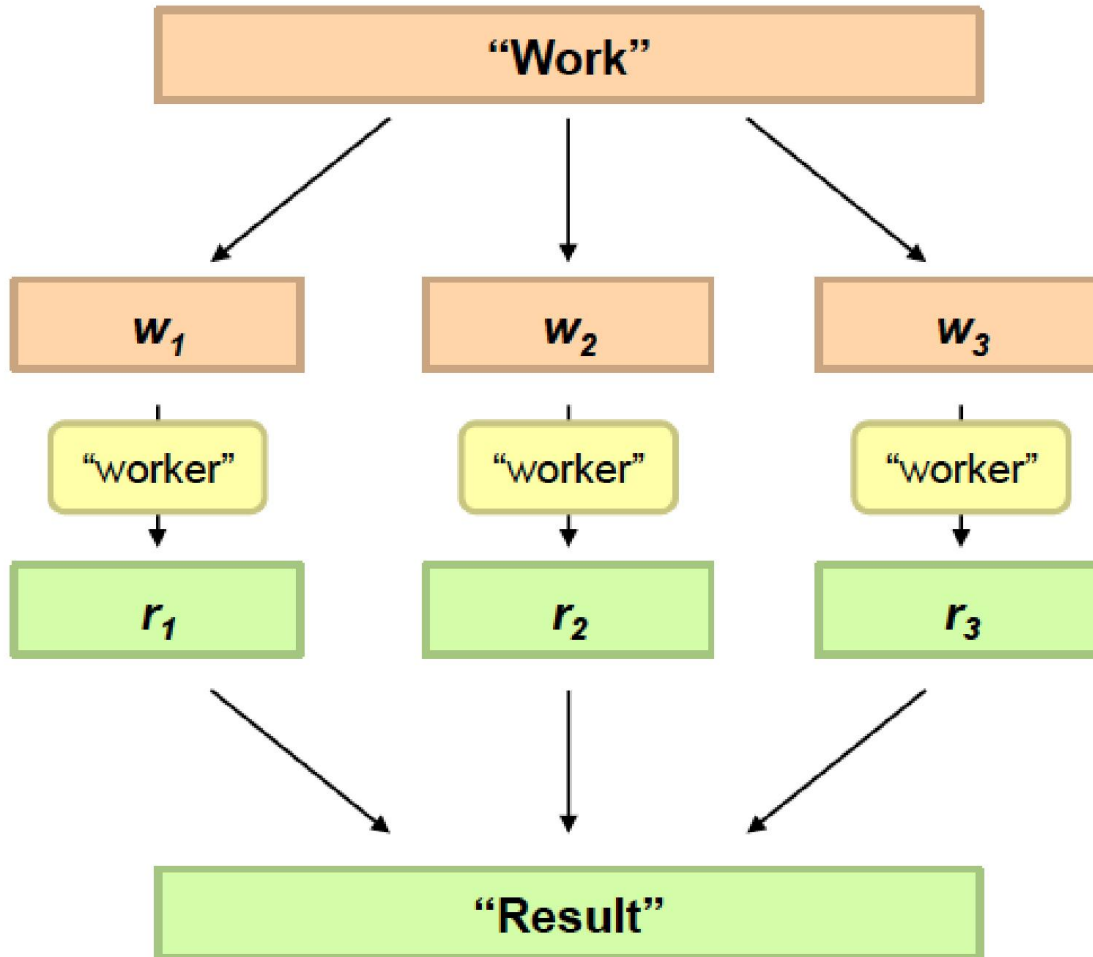
**Q: Where file is
located?**

**Q: How to handle
failures & data
lost?**

**Q: How to divide
computation?**

**Q: How to
program for
scaling?**

Philosophy to Scale for Big Data Processing



Divide Work



Combine Results

Why Use Hadoop?

☰ Cheaper

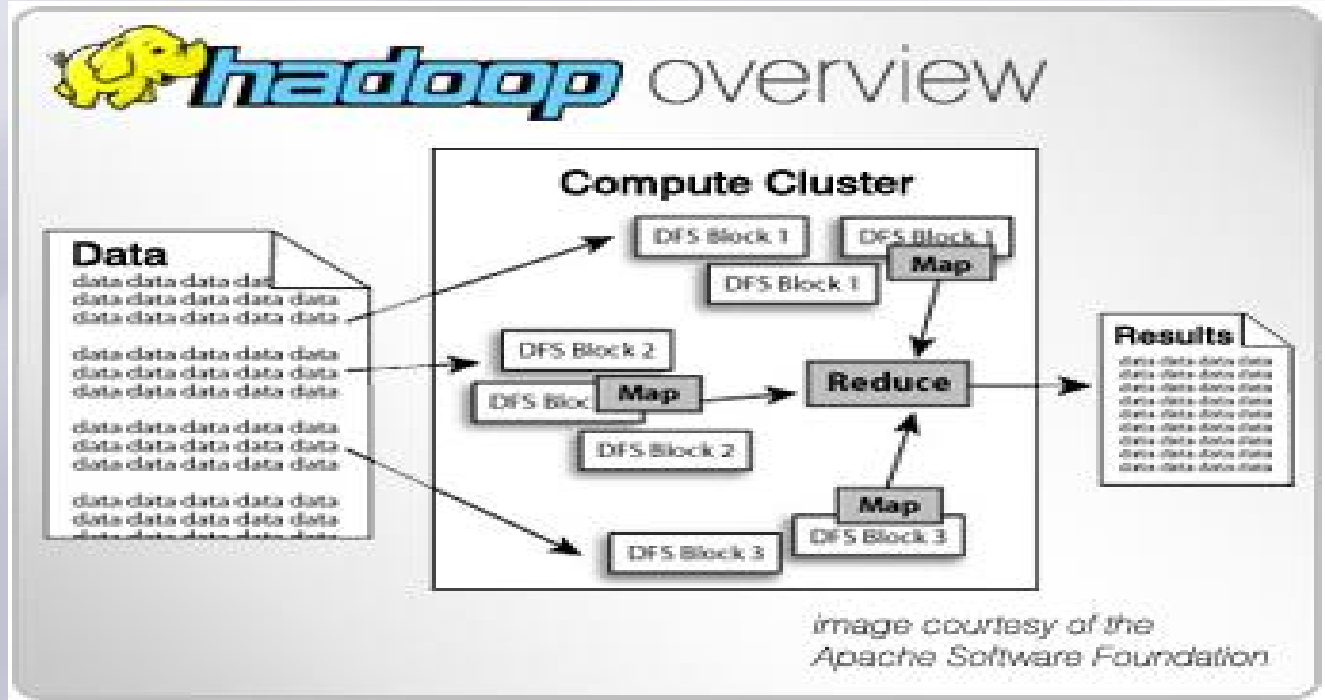
- Scales to Petabytes or more easily

☰ Faster

- Parallel data processing

☰ Better

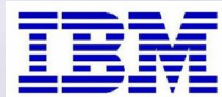
- Suited for particular types of big data problems



Companies Using Hadoop



facebook



The New York Times



eHarmony

twitter



NETFLIX

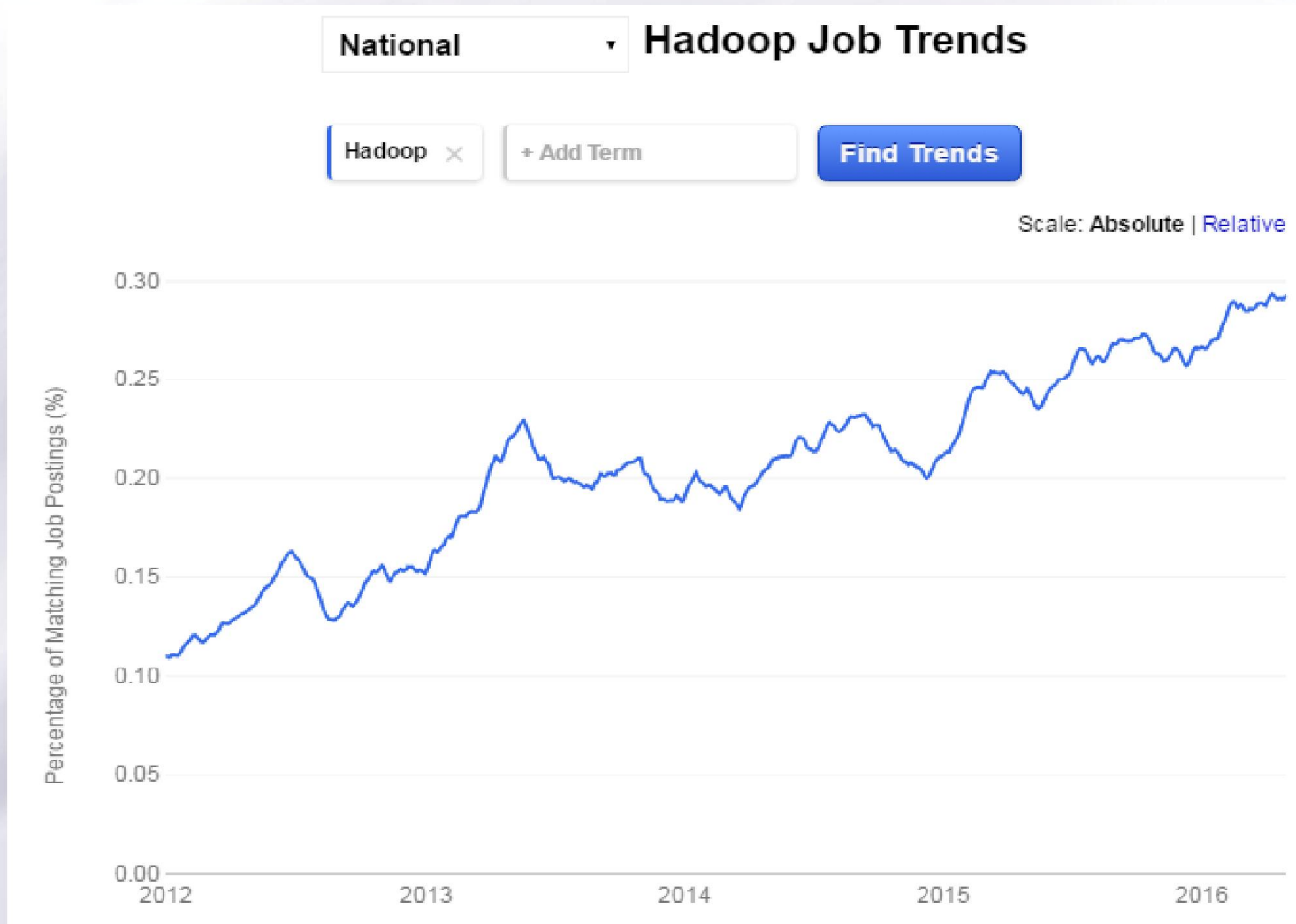


amazon.com



YAHOO!

Forecast growth of Hadoop Job Market



Hadoop is a set of Apache Frameworks: **core parts**

☰ Data storage (**HDFS**)

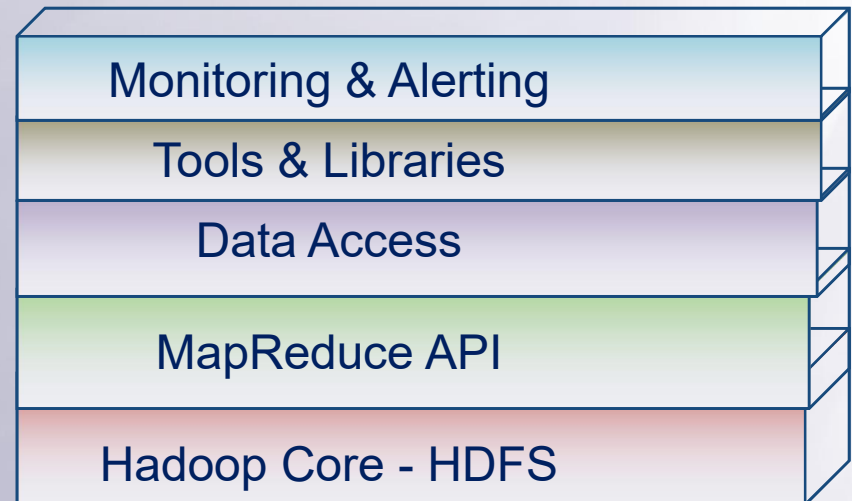
- Runs on commodity hardware (usually Linux)
- Horizontally scalable

☰ Processing (**MapReduce**)

- Parallelized (scalable) processing
- Fault Tolerant

☰ Other Tools / Frameworks

- Data Access
 - **HBase, Hive, Pig, Mahout**
- Tools
 - Hue, Sqoop
- Monitoring
 - Greenplum, Cloudera



What are the core parts of a Hadoop distribution?

HDFS Storage

- Redundant
- For large files – large blocks
- 64 or 128 MB / block
- Can scale to 1000s of nodes

MapReduce API

- Batch (Job) processing
- Distributed and Localized to clusters (Map)
- Auto-Parallelizable for huge amounts of data
- Fault-tolerant (auto retries)
- Adds high availability and more

Other Libraries

Pig
Hive
HBase
Others

Common Hadoop Distributions

Open Source

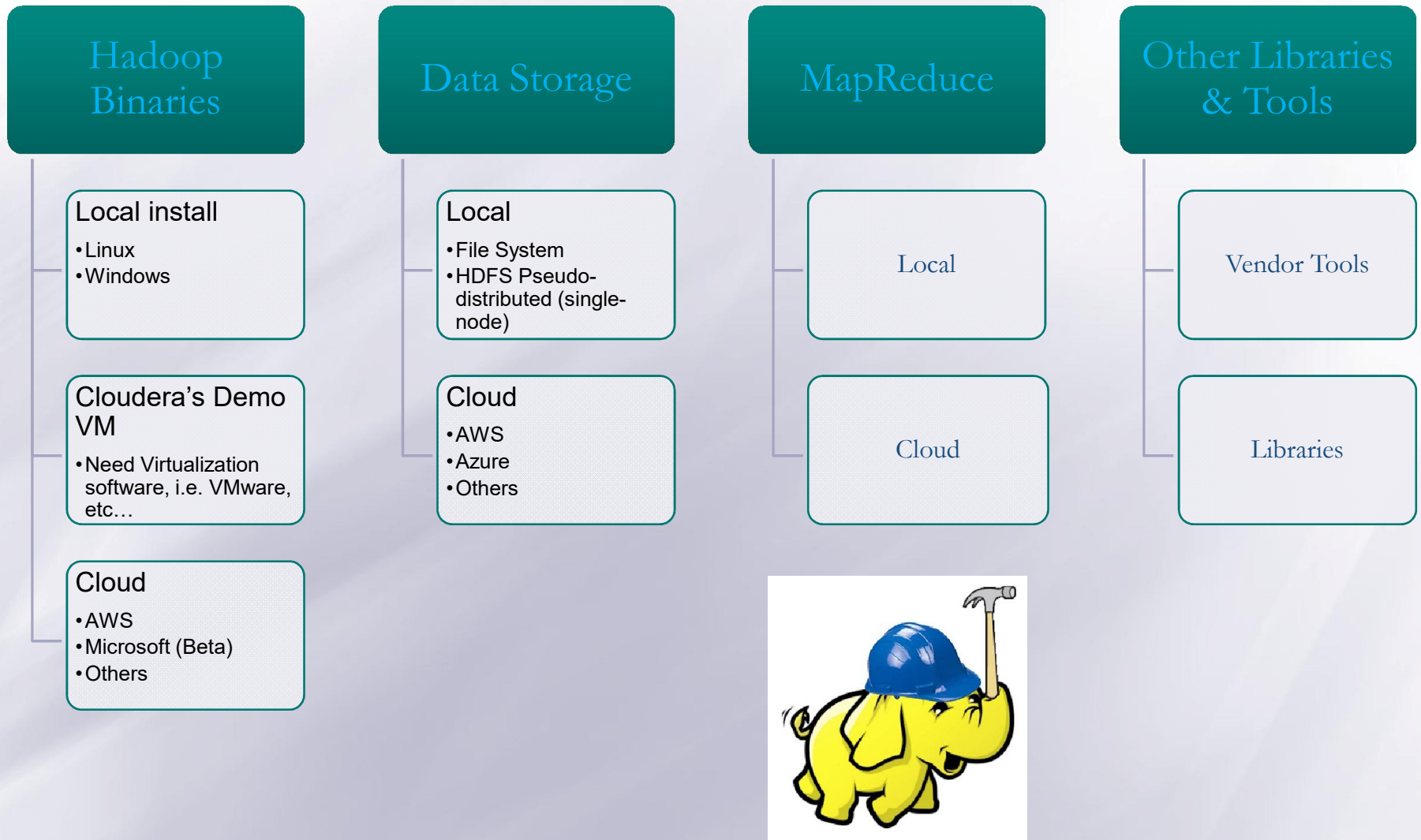
- Apache

Commercial

- Cloudera
- Hortonworks
- MapR
- AWS MapReduce
- Microsoft Azure HDInsight (Beta)

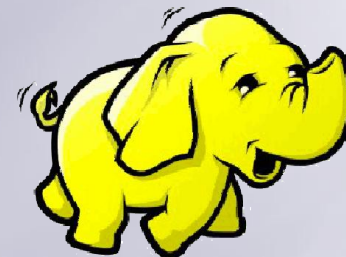


Setting up Hadoop Development



Comparing: RDBMS vs. Hadoop

	Traditional RDBMS	Hadoop / MapReduce
Data Size	Gigabytes (Terabytes)	Petabytes (Hexabytes)
Access	Interactive and Batch	Batch – NOT Interactive
Updates	Read / Write many times	Write once, Read many times
Structure	Static Schema	Dynamic Schema
Integrity	High (ACID)	Low
Scaling	Nonlinear	Linear
Query Response Time	Can be near immediate	Has latency (due to batch processing)



Storing Big Data

- Big Data NoSQL(Stands for **Not Only SQL**) databases were founded by top internet companies like Amazon, Google, LinkedIn and Facebook to overcome the drawbacks of RDBMS.
 - RDBMS is not always the best solution for all situations as it cannot meet the increasing growth of unstructured data.
 - As data processing requirements grow exponentially, NoSQL is a dynamic and cloud friendly approach to dynamically process unstructured data with ease.
- IT professionals often debate the merits of SQL vs. NoSQL but with increasing business data management needs, **NoSQL is becoming the new darling of the big data movement.**
- As per the web statistics report in 2016, there are about 4 billion people who are connected to the World Wide Web.
 - The amount of time that the internet users spend on the web is somewhere close to 40 billion hours per month, which is increasing.



- With the availability of several mobile and web applications, it is pretty common to have billions of users- who will generate a lot of unstructured data.

- ☰ There is a need for a database technology that can render 24/7 support to store, process and analyze this data using NoSQL.

- ☰ **NoSQL:**

- Class of non-relational data storage systems
- Usually do not require a fixed table schema nor do they use the concept of joins
- All NoSQL offerings relax one or more of the ACID properties

What kinds of NoSQL

☰ NoSQL solutions fall into two major areas:

- Key/Value or ‘the big hash table’.
 - Amazon S3 (Dynamo)
 - Voldemort
 - Scalaris
 - Memcached (in-memory key/value store)
 - Redis
- Schema-less which comes in multiple flavors, column-based, document-based or graph-based.
 - Cassandra (column-based)
 - CouchDB (document-based)
 - MongoDB (document-based)
 - Neo4J (graph-based)
 - HBase (column-based)

Key/Value


Pros:

- very fast
- very scalable
- simple model
- able to distribute horizontally

Cons: - many data structures (objects) can't be easily modeled as key value pairs

Common Advantages

- Cheap, easy to implement (open source)
- Data are replicated to multiple nodes (therefore identical and fault-tolerant) and can be partitioned
 - Down nodes easily replaced
 - No single point of failure
- Easy to distribute
- Don't require a schema
 - Can scale up and down
- Relax the data consistency requirement (CAP)

- 
- Several different varieties of NoSQL databases have been created to support specific needs and use cases. **These databases can broadly be categorized into four types:**

1. Key-value store NoSQL database

- From an API perspective, key-value stores are the simplest NoSQL data stores to use. The client can either get the value for the key, assign a value for a key or delete a key from the data store.
- Since key-value stores always use primary-key access, they generally have great performance and can be easily scaled.
- The key-value database uses a hash table to store unique keys and pointers (in some databases it's also called the inverted index) with respect to each data value it stores.
- The implementation is easy. Key-value databases give great performance and can be very easily scaled as per business needs.

☰ However key-value databases are **not the ideal choice for every use case** when:

- We have to query the database by specific data value.
- We need relationships between data values.
- We need to operate on multiple unique keys.
- Our business needs updating a part of the value frequently.
- Examples of this database are Redis, MemcacheDB and Riak.

Key/Value Store

Key: 1	ID: sj	First Name: Sam	
Key: 2	Email: jb@gmail.com	Location: London	Age: 37
Key: 3	Facebook ID: jkirk	Password: xxx	Name: James

2. Document store NoSQL database

Document store NoSQL databases are similar to key-value databases in that there's a key and a value.

- Data is stored as a value.

Its associated key is the unique identifier for that value. The difference is that, in a document database, the value contains structured or semi-structured data.

This structured/semi-structured value is referred to as a document and can be in XML, JSON or BSON format.

Use cases: Document store databases are preferable for:

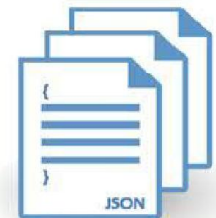
- E-commerce platforms, Content management systems.
- Analytics platforms, Blogging platforms

Examples of document store NoSQL databases are MongoDB, Apache CouchDB and Elastic search.



Relational data model

Highly-structured table organization with rigidly-defined data formats and record structure.



Document data model

Collection of complex documents with arbitrary, nested data formats and varying "record" format.

3. Column store NoSQL database

- In column-oriented NoSQL databases, data is stored in cells grouped in columns of data rather than as rows of data.



- Columns are logically grouped into column families. Column families can contain a **virtually** unlimited number of columns that can be created at runtime or while defining the schema.

 - Read and write is done using columns rather than rows.

- Column families are groups of similar data that is usually accessed together. As an example, we often access customers' names and profile information at the same time, but not the information on their orders.

- The main advantages of storing data in columns over relational DBMS are fast search/access and data aggregation. Relational databases store a single row as a continuous disk entry.

- Different rows are stored in different places on the disk while columnar databases store all the cells corresponding to a column as a continuous disk entry, thus making the search/access faster.



Each column family can be compared to a container of rows in an RDBMS table, where the key identifies the row and the row consists of multiple columns.

- The difference is that various rows do not have to have the same columns, and columns can be added to any row at any time without having to add them to other rows.

Use cases: Developers mainly use column databases in:

- o Content management systems
- o Blogging platforms
- o Systems that maintain counters
- o Services that have expiring usage
- o Systems that require heavy write requests (like log aggregators)

A Representation of Column store on NoSQL

Relational DB						Column Families DB	
	A	B	C	D	E	1	A → foo; B → bar; C → hello
1	foo	bar	hello			2	B → Tom
2		Tom				3	C → Java; D → Scala; E → Perl
3			Java	Scala	Perl		

N.B:

Column store databases should be avoided if you have to use complex querying or if your querying patterns frequently change. Also avoid them if you don't have an established database requirement, a trend which we are beginning to see in new systems.

Examples of column store NoSQL databases are Cassandra and Apache Hadoop Hbase.

4. Graph base NoSQL database

≡ Graph databases are basically built upon the Entity/node – Attribute – Value model.

≡ It is a very flexible way to describe how data relates to other data. Nodes store data about each entity in the database, relationships describe a relationship between nodes, and a property is simply the node on the opposite end of the relationship.

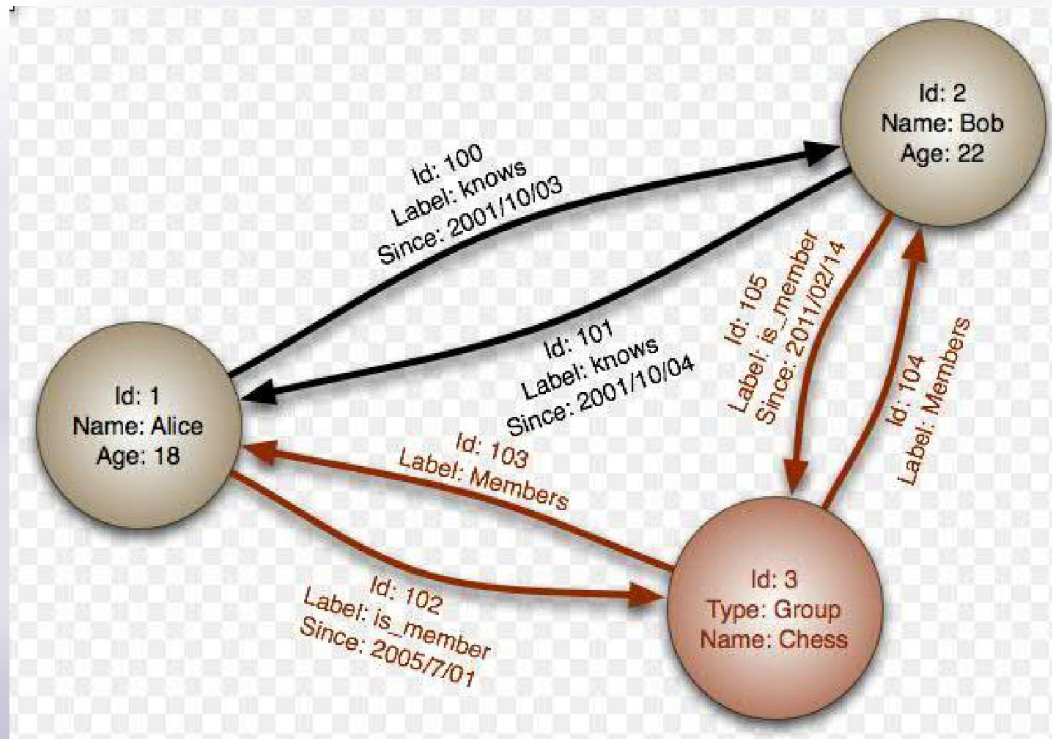
≡ Whereas a traditional database stores a description of each possible relationship in foreign key fields or junction tables, graph databases allow for virtually any relationship to be defined on-the-fly.



Use cases: Graph base NoSQL databases are usually used in:

- o Fraud detection
- o Graph based search
- o Network and IT operations
- o Social networks, etc

Examples of graph base NoSQL databases are Neo4j, ArangoDB and OrientDB.



A Representation of Graph base on NoSQL

Review Questions

1. What makes Hadoop and NoSQL are more unique than the conventional tools? Show by examples.
2. Describe the working principle of Big data processing?
3. Advantages and disadvantages of big data?
4. What are the main category of big-data databases? Show the use cases of each by graphical example.

Assignment1:

- Configure an Hadoop software and demonstrate/ present how it works (20%).
- Assignment 2 (10 %) due on 18/10/2013:
 - Read one article that has been done in big data using machine learning, and present in ppt.