Storage Infrastructures for Large Data Volumes

Gianluca Demartini

DATA7201 Data Analytics at Scale

Week 3

Week	Date	Lecture	Prac	Assessment
1	21-Feb	Introduction to DATA7201 - Data Analytics at Scale	-	
2	28-Feb	Supporting Infrastructures and Use Cases	-	
3	6-Mar	Storage Infrastructures for Large Data Volumes	Intro to Cluster and HDFS	
4	13-Mar	Analytics Queries for Large Data Volumes	PIG(1)	
5	20-Mar	Distributed Data Processing	PIG (2)	
6	27-Mar	Processing Large Data Streams	PySpark (1)	**Quiz 1 Due (5)**
Semester Break				
7	10-Apr	Processing Large Graph Data (1) + use cases	PySpark (2)	
8	17-Apr	Processing Large Graph Data (2) + use cases	Project support	
9	24-Apr	Recommender Systems	Project support	**Quiz 2 Due (5)**
10	1-May	Opinion Mining + use cases	Project support	
11	8-May	Health Data Analytics (guest speaker)	Project support	
12	15-May	Large Language Models?	Project support	**Report Due (45)**
13	22-May	Course Revision	-	**Quiz 3 Due (5)**

Last Week

- Example applications and data products that can be developed using Big Data
- Infrastructures required to handle Big Data and enable Data Analytics
- Cloud computing
- Scalable architectures (e.g., MapReduce, Hadoop, Pig, HBase)
  - Volume
- Open source vs. commercial software/services

Lecture Outline

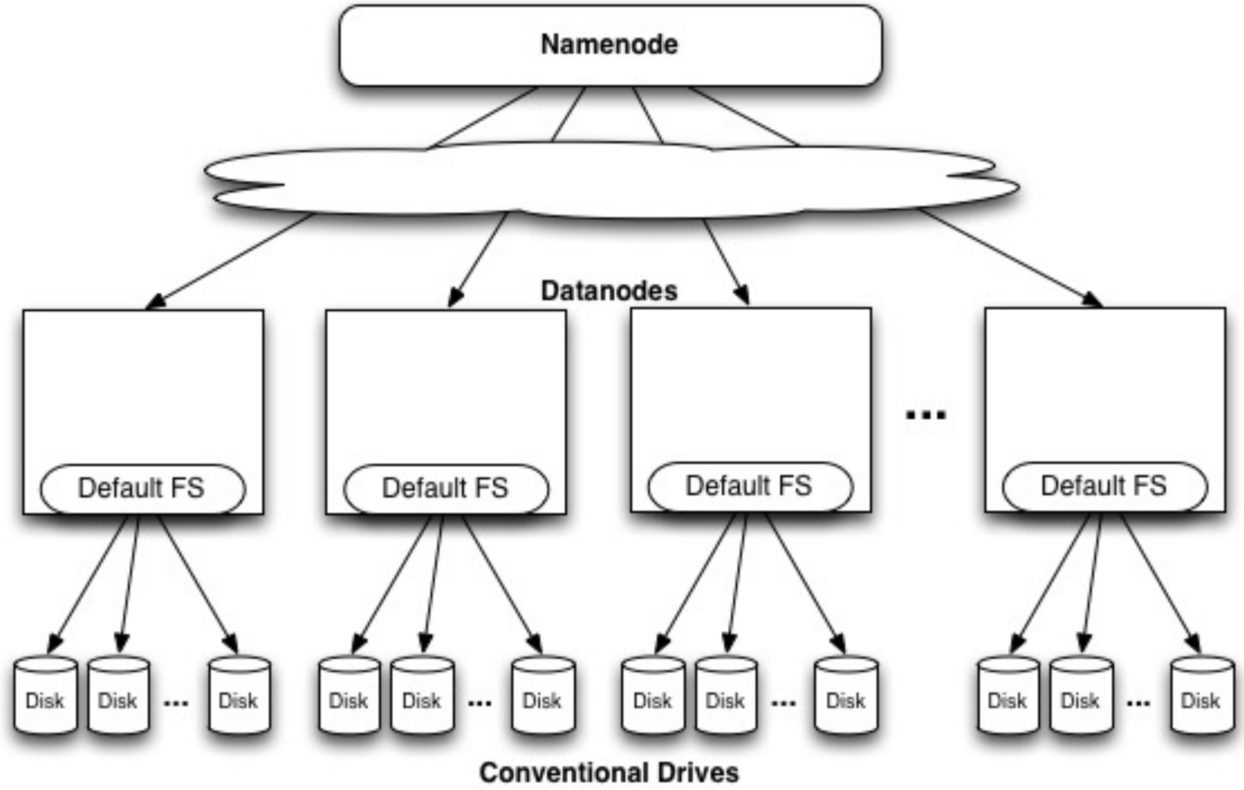
- Distributed Infrastructures
  - CAP Theorem
- Google File System
- HDFS
- Spark RDDs
- Column-stores / noSQL

Distributed Systems

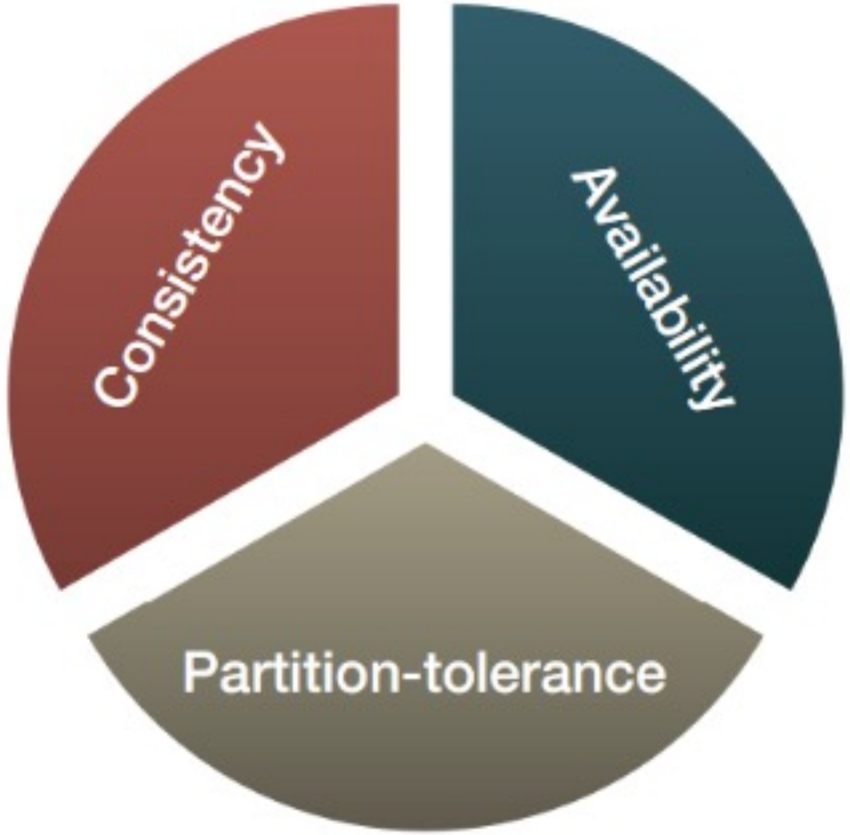
- [Wikipedia] A distributed system is a model in which components located on **networked computers** communicate and **coordinate their actions** by passing **messages**. The components interact with each other in order to achieve a **common goal**.

Distributed Systems

- Large Data Volumes
- Store them over a distributed system



CAP properties of distributed systems

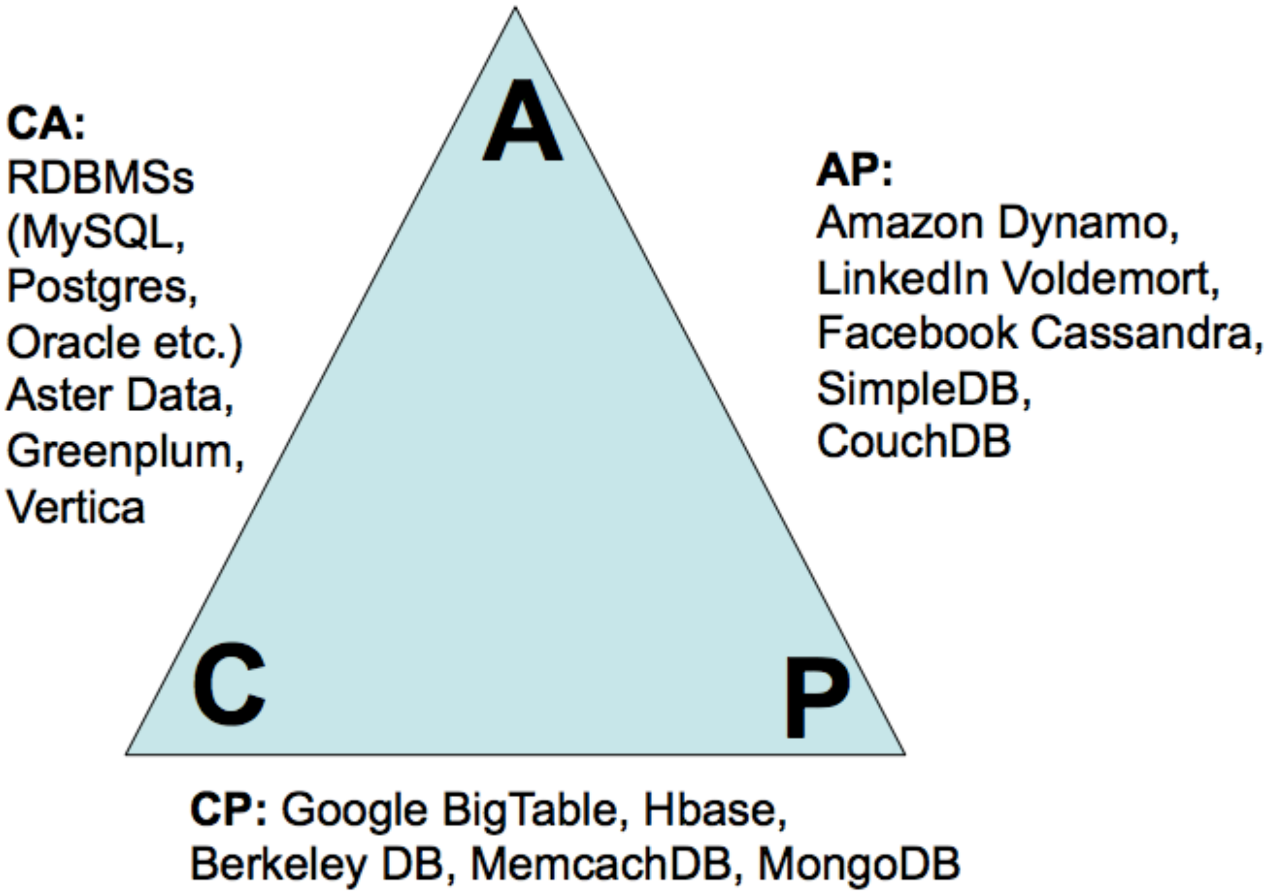


CAP Theorem

- **Consistency** means that each client always has the same view of the data.
- **Availability** means that all clients can always read and write.
- **Partition tolerance** means that the system works well across physical network partitions.

- Only 2 out of 3 can be implemented

Some recent systems



http://blog.nahurst.com/visual-guide-to-nosql-systems

Map/Reduce and Hadoop

> "MapReduce is a programming model for expressing **distributed** computations on **massive amounts of data** and an execution framework for large-scale data processing on clusters of **commodity servers**."
>
> -Jimmy Lin

GFS

Hadoop is an **open-source implementation** of the MapReduce framework.

HDFS

Distributed File Systems

Google File System

- Research papers:
  - *The **Google file system*** by Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung (2003)
  - *The Hadoop distributed file system* by Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler (2010)

What is a file system?

- File systems determine **how** data is stored and retrieved
- **Distributed file systems** manage the storage across a network of machines
  - GFS/HDFS are distributed file systems

GFS Assumptions

- Hardware **failures are common** (commodity hardware)
- **Files are large** (GB/TB) and their number is limited (millions, not billions)
- Two main types of reads: **large streaming reads** and **small random reads**
- Workloads with **sequential writes** that **append** data to files
- Once written, files are **seldom modified** (!=append) again
  - Random modification in files possible, but not efficient in GFS

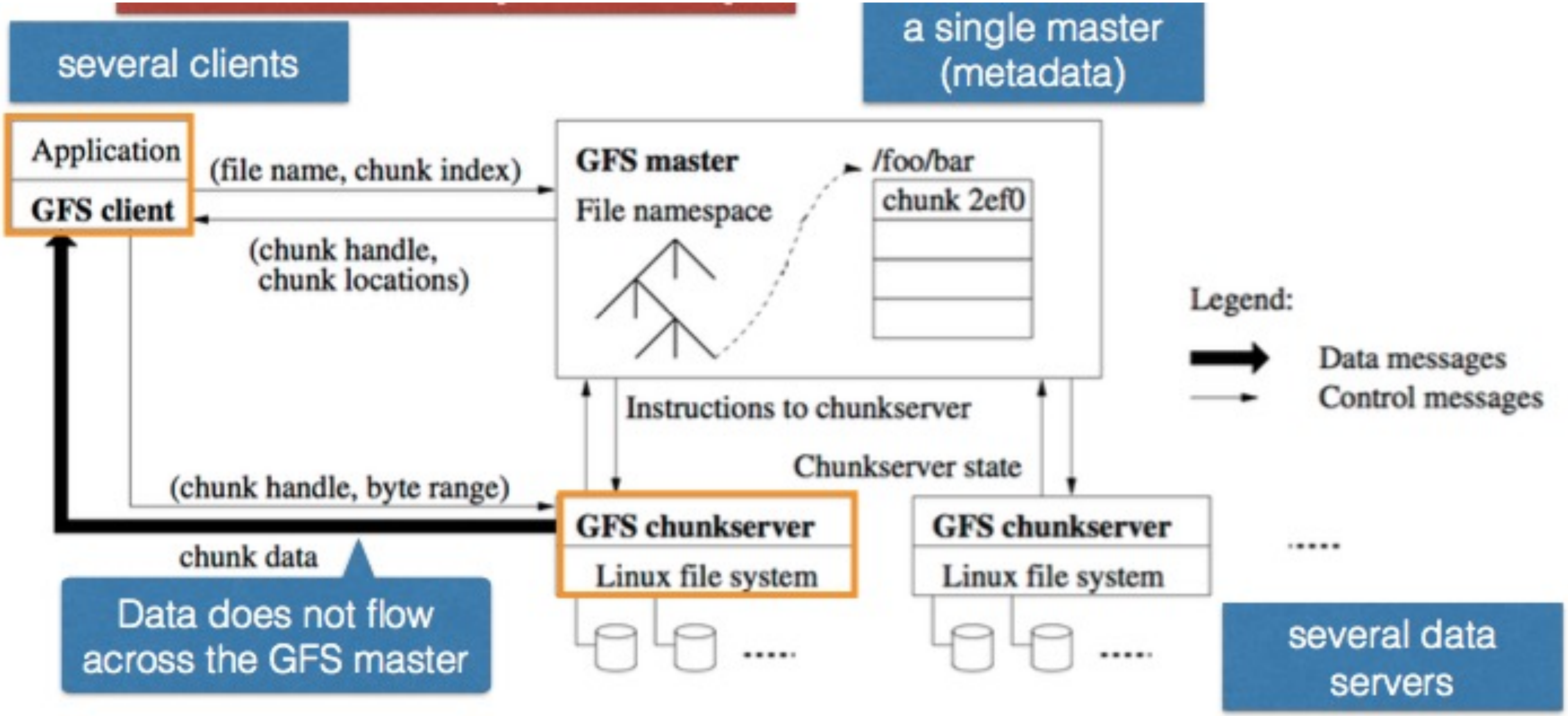
GFS is not good for

- **Low latency data access** (in the milliseconds range)
- **Many small files**
- **Constantly changing data**

Files in GFS

- A **single file** can contain **many objects**
- Files are divided into **fixed size chunks** (64MB)
  - In Hadoop 128MB
- **chunkservers** store chunks on local disk as “normal” files
- Files are **replicated** (by default 3 times) across all chunk servers
- **master** maintains all **file system metadata**
- **To read/write data**: client communicates with master (metadata)

GFS Architecture



Single master architecture

- Single master **simplifies the design** tremendously
  - Chunk placement and replication with **global knowledge**
- Single master in a large cluster can become a **bottleneck**

Hadoop Distributed File System (HDFS)

- Inspired by Google File System
- Scalable, distributed, portable file system written in Java for Hadoop framework
- Primary distributed storage used by Hadoop applications
- HDFS can be part of a Hadoop cluster or can be a stand-alone general purpose **distributed file system**
- Reliability and **fault tolerance** ensured by replicating data across multiple hosts
- Zookeeper for the distributed coordination

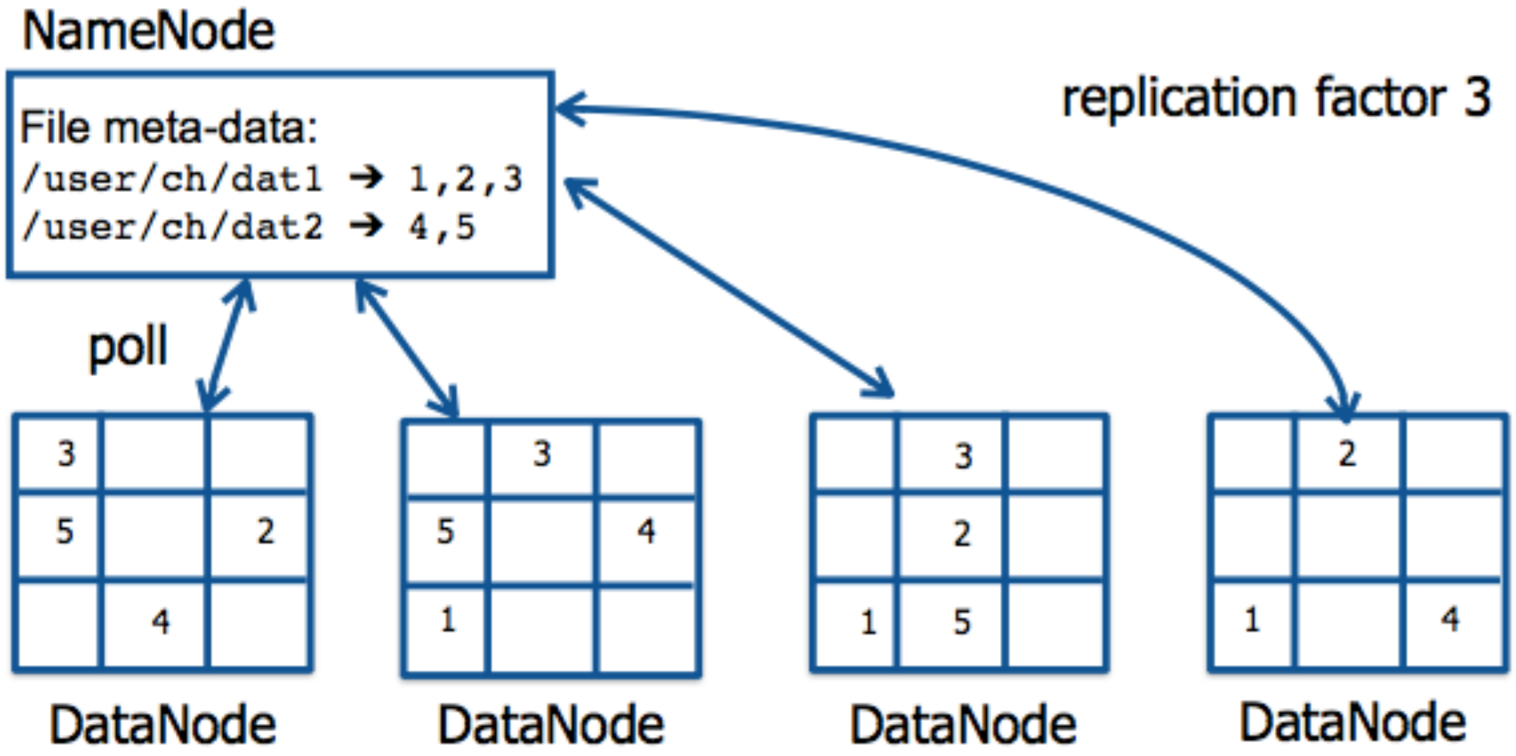
GFS vs HDFS

GFS	HDFS
Master	NameNode
chunkserver	DataNode
operation log	journal, edit log
chunk	block
random file writes possible	**only append is possible**
multiple writer, multiple reader model	single writer, multiple reader model
chunk: 64KB data and 32bit checksum pieces	per HDFS block, two files created on a DataNode: data file & metadata file (checksums, timestamp)
default block size: 64MB	default block size: 128MB

HDFS

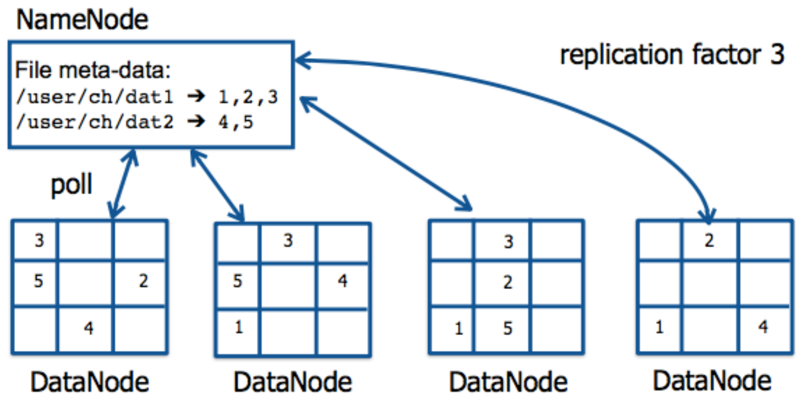
- **NameNode**
  - **Master** of HDFS, directs the slave DataNode daemons to perform low-level I/O tasks
  - Keeps track of file splitting into blocks, replication, block location, etc.
- **Secondary NameNode**: takes snapshots of the NameNode
- **DataNode**: each slave machine hosts a DataNode daemon

NameNodes and DataNodes



HDFS Replication

- Files divided in blocks and replicated
  - Block size and replication factors are configurable
- NameNode makes all replication decisions
  - It receives Hearbeats and Blockreport from DataNodes



- Replica placement
  – E.g., on unique racks (good for reliability, bad for latency)
  – Satisfy a read request from a replica that is closest to the reader

https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#Data+Replication

Hadoop in practice: Yahoo! (2010)

- **40 nodes/rack** sharing one IP switch
- **16GB RAM** per cluster node, 1-gigabit Ethernet
- **70% of disk space allocated to HDFS**
- **NameNode**: up to **64GB RAM**
- **Total storage**: 9.8PB -> **3.3PB** net storage (replication: 3)
- **60 million files**, 63 million blocks
- Cluster with 3500 nodes
  - **1-2 nodes lost per day**
  - **Time for cluster to re-replicate lost blocks: 2 minutes**

Apache Spark Data Structures

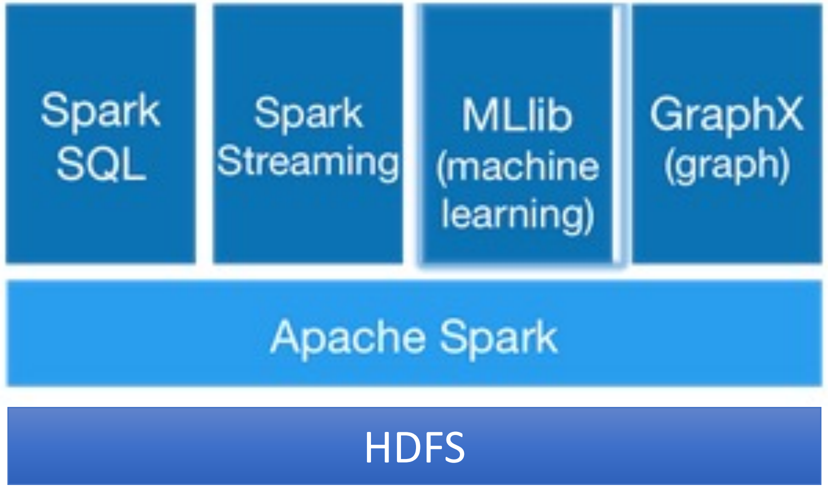
Apache Spark

- https://spark.apache.org/ by databricks.com
- Started by students at UC Berkeley
- General system for large-scale data processing
- Faster (in-memory), interactive
- Version 3.3 released in 2022 (v3 since 2020)
  - SparkR

  But also:

  sparklyr

  etc.



- pyspark

Spark RDDs (Ch. 3, Spark book)

- Based on HDFS
- Resilient distributed dataset (RDD)
  - Distributed collection of elements
  - Immutable
  - Lazy evaluation
- Spark distributes data across the cluster
  - Partitions
- Operations
  - Transformations
  - Actions

In Python: https://spark.apache.org/docs/latest/api/python/

Transformations

- Generates a new RDD partition
- Lazy evaluation (computed only when needed)
- Filter: select certain rows in a log file
- Map, Filter, flatMap, Sample, Union, Intersection, Distinct, groupByKey, reduceByKey, sortByKey, Join, Cogroup, cartesian

Transformations

- **Map(function):** Return a new distributed dataset formed by passing each element of the source through a function
- **FlatMap(function):** each input item can be mapped to 0 or more output items (instead of one as for map)
- **reduceByKey(function):** When called on a dataset of (K, V) pairs, returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function, which must be of type (V,V) => V.

https://spark.apache.org/docs/latest/rdd-programming-guide.html#working-with-key-value-pairs

Actions

- Return a final value
- Force the evaluation of transformation operations

- Reduce, Collect, Count, First, Take, takeSample, saveAsTextFile, foreach

- **Collect():** Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.

NoSQL

- Non-relational Databases: noSql
  - [Wikipedia] **NoSQL** is a term used to designate database management systems that differ from classic relational database management systems (RDBMS) in some way. These data stores may **not require fixed table schemas**, usually **avoid join** operations, **do not** attempt to **provide ACID** properties and typically **scale horizontally**.
- Distributed: scale-out

ACID Properties

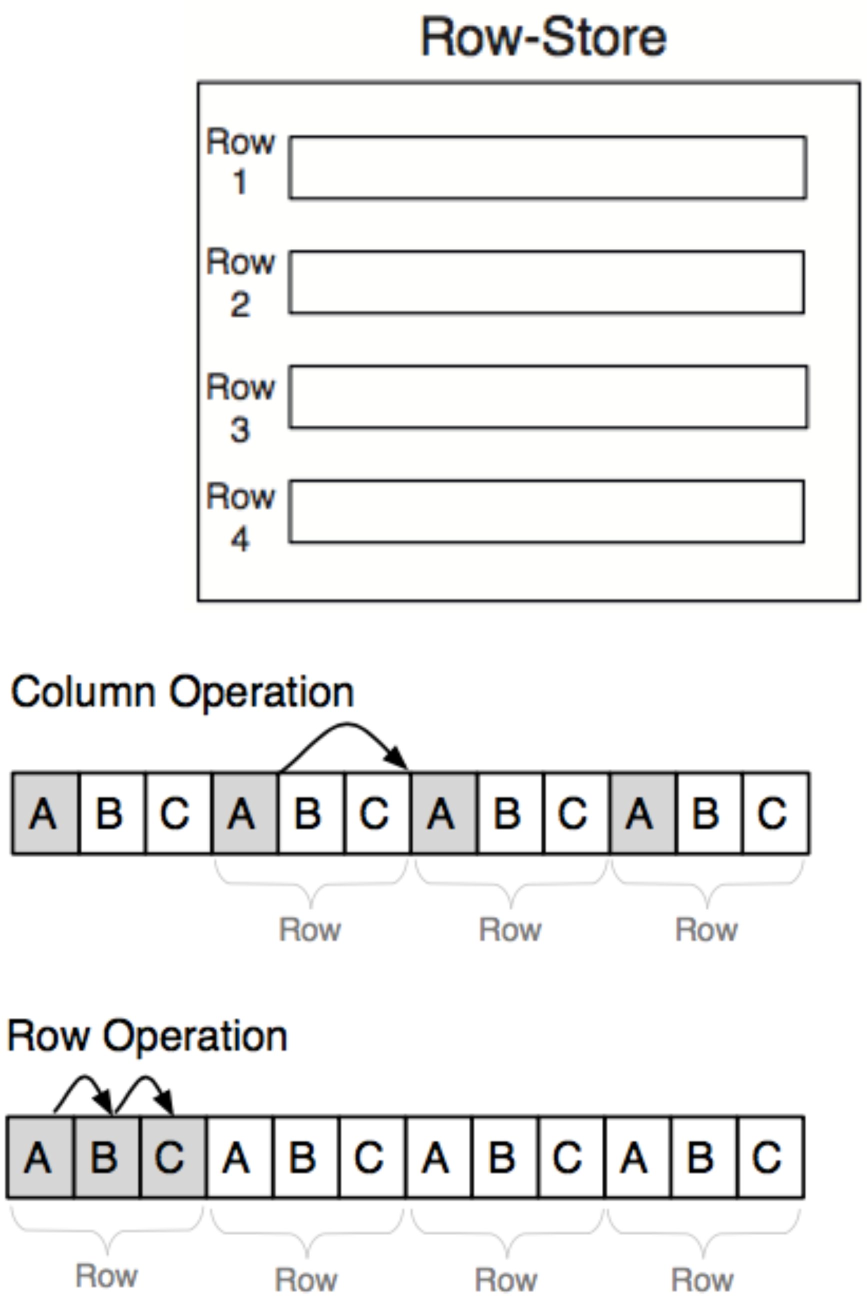
- **A**tomicity
  - A transaction is completely executed or not at all
- **C**onsistency
  - A DB is in a consistent state before and after a transaction is executed
- **I**solation
  - A transaction execution is not affected by other concurrent transactions
- **D**urability
  - Changes made during a transaction are permanent

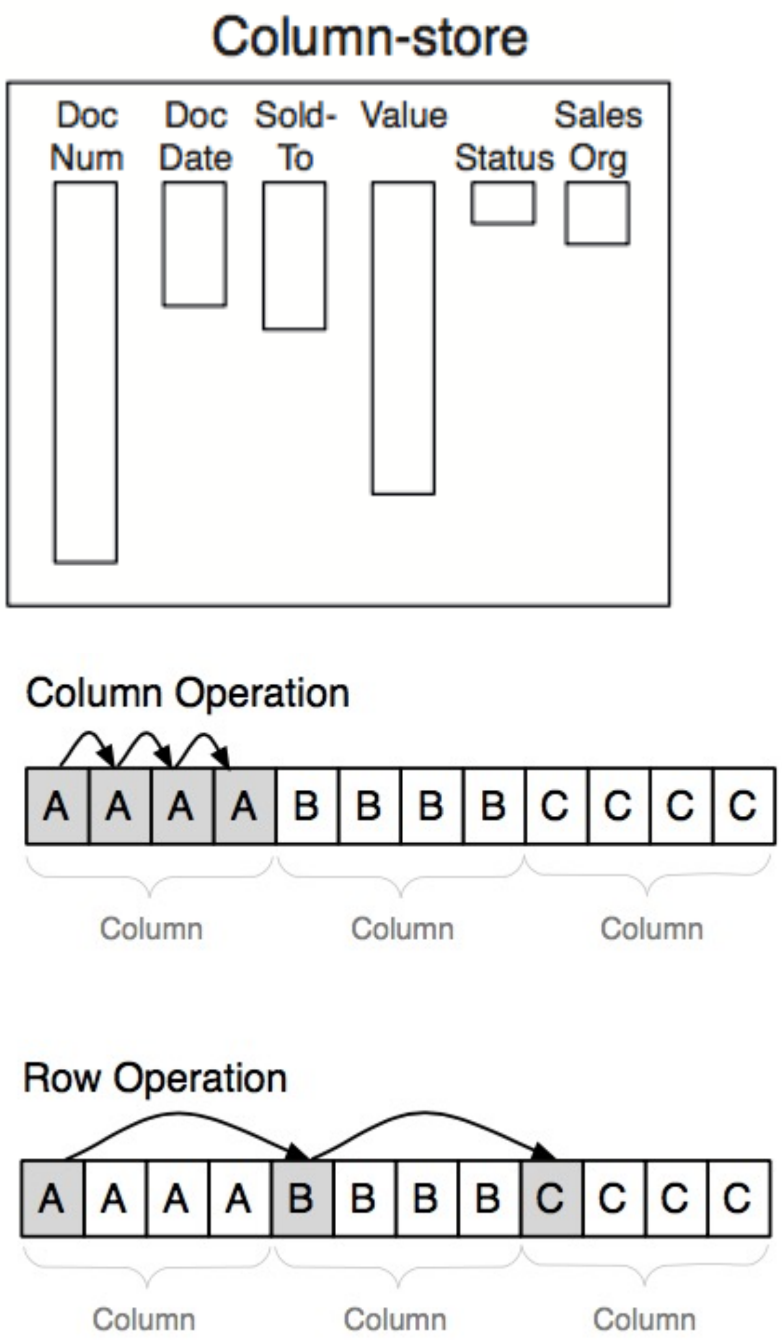
NoSQL

- Four approaches:
  - Key-value (BigTable (2004), Dynomite (2008), Voldemort (2009))
  - Column-oriented (Hbase (2007), Cassandra (2008))
  - Graph-based (Neo4j (2007), ArangoDB(2011))
  - Document-oriented (CouchDB(2005), MongoDB (2009))

Column Stores

- **Row** store:
  - Rows are stored consecutively
  - Optimal for row-wise access (e.g. SELECT *)
- **Column** store:
  - Columns are stored consecutively
  - Optimal for attribute focused access (e.g. SUM, GROUP BY)


Row-Store
Row 1
Row 2
Row 3
Row 4
Column Operation
A B C A B C A B C A B C
Row
Row
Row
Row Operation
A B C A B C A B C A B C
Row
Row
Row
Row


Column-store
Doc Num
Doc Date
Sold-To
Value
Status
Sales Org
Column Operation
A A A A B B B B C C C C
Column
Column
Column
Row Operation
A A A A B B B B C C C C
Column
Column
Column

Apache HBase

- “Apache HBase is the Hadoop database, a distributed, scalable, big data store.”
- On top of HDFS
- Billions of rows * millions of columns
- Non-relational DB / NoSQL
- Column store: columns instead of tables
- Key-value cells

Apache HBase

HBase is not ACID compliant

"HBase is a **distributed column-oriented** *database* built on top of HDFS. HBase is the Hadoop application to use when you require **real-time** read/write **random** access to very **large** datasets." (Tom White)

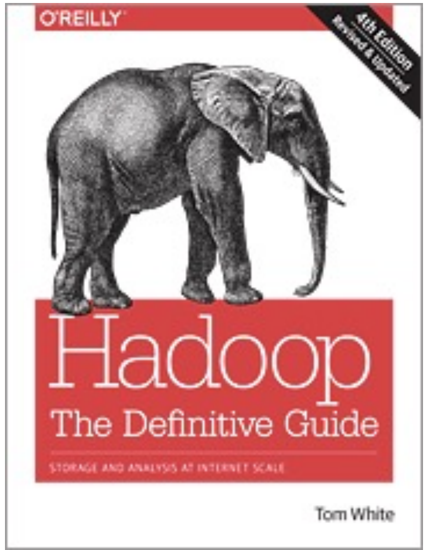
"HBase tables are like those in an RDBMS, only cells are **versioned**, rows are **sorted**, and columns can be **added on the fly**…" (Tom White)

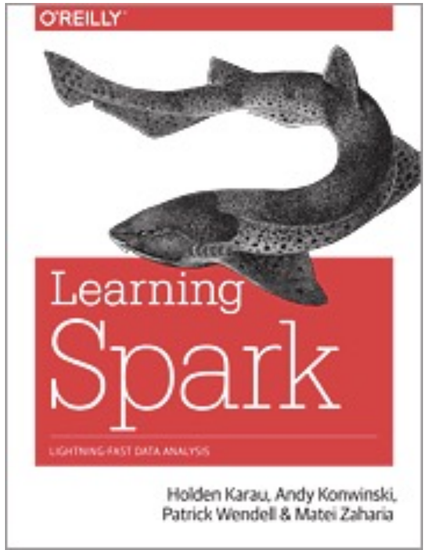
Summary

- Distributed Infrastructures
  - CAP Theorem
- Google File System
- HDFS
- Spark RDDs
- Column-stores / noSQL

References

- Tom White. *Hadoop: The Definitive Guide.* O'Reilly.
- Zaharia et al. *Learning Spark*. O'Reilly.
- *The **Google file system*** by Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung (2003)
- **Bigtable: A Distributed Storage System for Structured Data.** Fay Chang et. al. OSDI 2006





DATA7201 Cluster

- Amazon EMR set up operates the Hadoop cluster
- Student notebooks on UQcloud zones (JupyterLab, new this year)
- A VPN link joins the two systems together
- m5.4xlarge EC2 instances (each have 64 GB RAM and 16 CPU cores)
- Resizing the cluster during the semester

Time span	Instances	Compute memory [5]	Available disk
Weeks 1-6	3	128 GB	600 GB
Week 7	6	320 GB	900 GB
Week 8	9	512 GB	1.75 TB
Week 9	25	1.5 TB	4.8 TB
Weeks 10-12	49	3 TB	9.3 TB [3]
Weeks 12-14	9	512 GB	1.75 TB
(Base + Avg. Spot)	3 + 9.42	128 + 667 GB	600 GB + 1.77 TB

- ~$30K AUD