Large Scale Data Processing-MapReduce

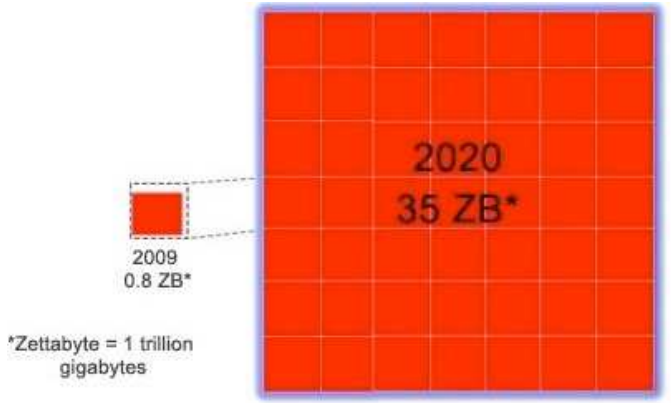
Introduction

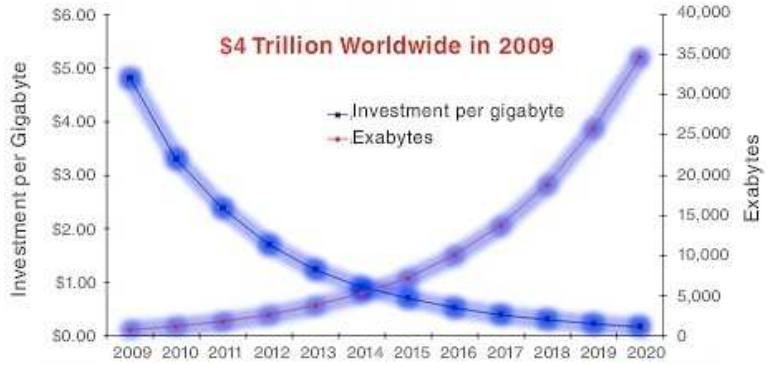
Dr. Wenceslao PALMA
wenceslao.palma@ucv.cl

August 2013

Large Data

- Websites
  - 266 million- The number of websites by december 2010.
  - 21.4 million- The number of added websites in 2010.
- Social Media
  - 152 million- The number of blogs on the Internet (as tracked by BlogPulse 2010).
  - 25 billion- The number of sent tweets on Twitter in 2010.
  - 175 million- People on Twitter as of September 2010.
  - 600 million- People on Facebook at the end of 2010.





Large Scale

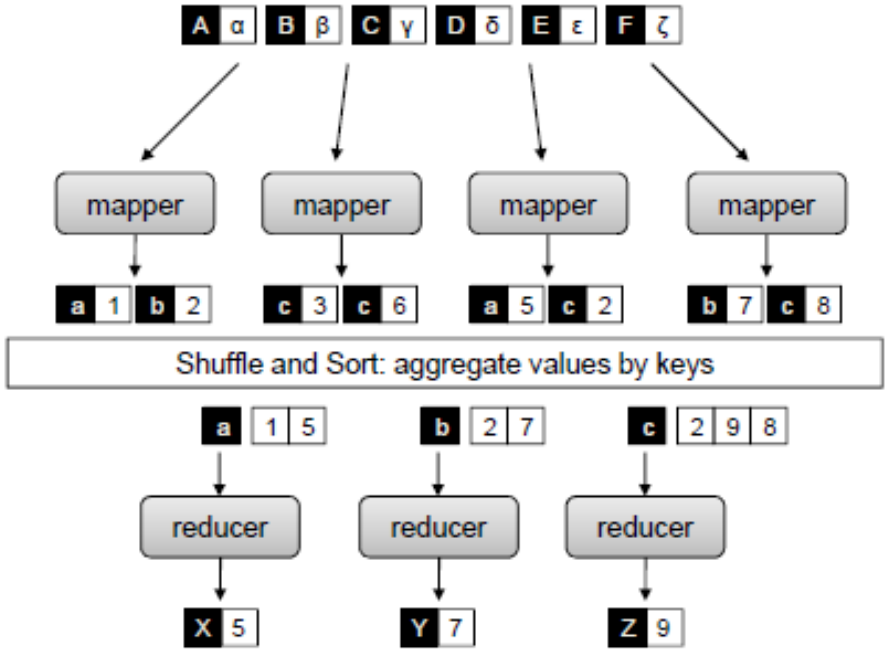
- Google grew from processing 100 TB of a data day in 2004 to processing 20 PB a day in 2008.
- Facebook reports 2.5 PT of user data growing at about 15 TB/day in 2009.

- Moving beyond the commercial sphere
  - The Large Hadron Collider will produce roughly 15 petabytes (15 million gigabytes) of data annually.
  - When the Large Synoptic Survey Telescope (LSST) comes online around 2015 in Chile, its 3.2 gigapixel primary camera will produce approximately half a petabyte of archive images every month.

- The only feasible approach to tackling large-data problems today is to divide and conquer.
- The general principles behind divide-and-conquer algorithms are broadly applicable to a wide range of problems in many different application domains.
- There are many issues that need to be addressed:
  - How to break up a large problem into smaller tasks?
  - How to assign tasks across a potentially large number of computer nodes.?
  - How to coordinate synchronization among the different nodes?
  - How to share partial results from one node that is needed by another?
  - How do we accomplish all of the above in the face of software errors and hardware faults?

MapReduce

MapReduce is a programming model for data processing introduced by Google (2004) to support parallel and fault-tolerant computations over large data sets on clusters of computers. It provides an abstraction that hides many system-level details from the programmer.



Apache Hadoop

A project that develops open-source software for realiable, scalable, distributed computing, including:

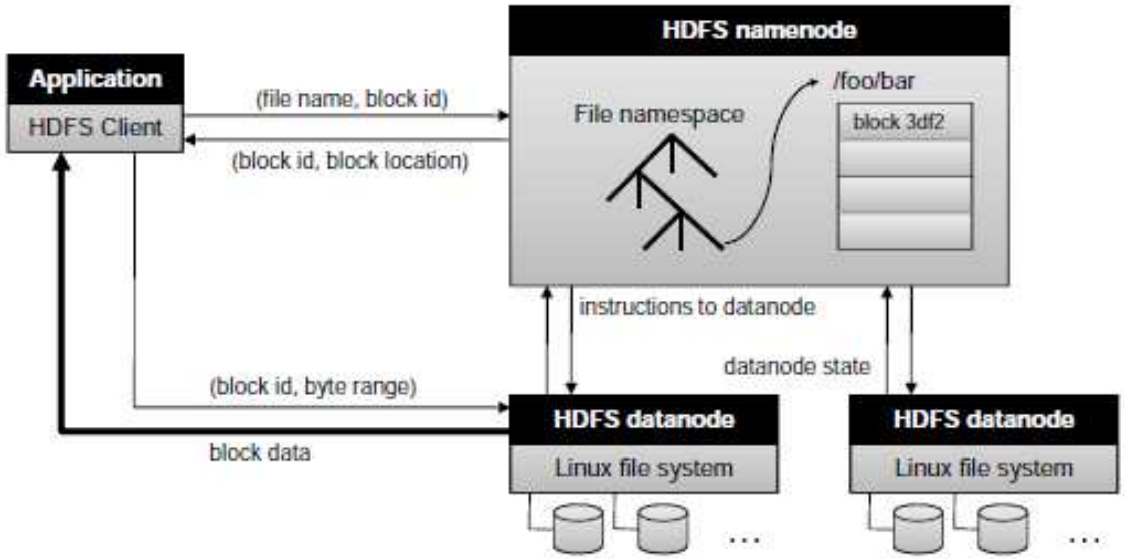
- Hadoop core: MapReduce support which is refered as only Hadoop.
- Hadoop Distributed File System (HDFS)
- Pig: A high-level data-flow language and execution framework for parallel computation.

Hadoop $\neq$ Mapreduce

- A MapReduce program breaks a job into small pieces and processes each of them in a parallel fashion.
- Hadoop core refers to the overall system for running MapReduce programs.

HDFS

Hadoop Distributed File System (HDFS) is the primary storage system used by Hadoop applications. HDFS creates multiple replicas of data blocks and distributes them on compute nodes throughout a cluster to enable reliable, extremely rapid computations.



HDFS

- By default, HDFS stores three separate copies of each data block to ensure both reliability, availability, and performance.
- In large clusters, the three replicas are spread across diffeerent physical racks, so HDFS is resilient towards two common failure scenarios: (1) individual datanode crashes and (2) failures in networking equipment that bring an entire rack offine.

Pig

- Pig raises the level of abstraction for processing large datasets.
- The language used to express data flows called *Pig Latin*.
- Pig transforms a data flow into a series of MapReduce jobs.

Pig

- Pig raises the level of abstraction for processing large datasets.
- The language used to express data flows called *Pig Latin*.
- Pig transforms a data flow into a series of MapReduce jobs.

Pig allows to focus on the data rather than the nature of the execution.

Pig

- Pig raises the level of abstraction for processing large datasets.
- The language used to express data flows called *Pig Latin*.
- Pig transforms a data flow into a series of MapReduce jobs.

Pig allows to focus on the data rather than the nature of the execution.

"The [Hofmann PLSA E/M] algorithm was implemented in pig in 30-35 lines of pig-latin statements. **Took a lot less compared to what it took in implementing the algorithm in Map-Reduce Java**. Exactly that's the reason I wanted to try it out in Pig. It took 3-4 days for me to write it, starting from learning pig."
– Prasenjit Mukherjee, Mahout project [a]

[a] Olston C. et al. Programmming and Debugging Large-Scale Data Processing Workflows

Traditional DBMS vs MapReduce

	Traditional DBMS	**MapReduce**
Data size	Gigabytes	Petabytes
Access	Interactive and batch	Batch
Updates	Read and write many times	Write once, read many times
Structure	Static schema	Dynamic schema
Integrity	High	Low