CMPT 354: Database System I

Lecture 12. Where to go from here

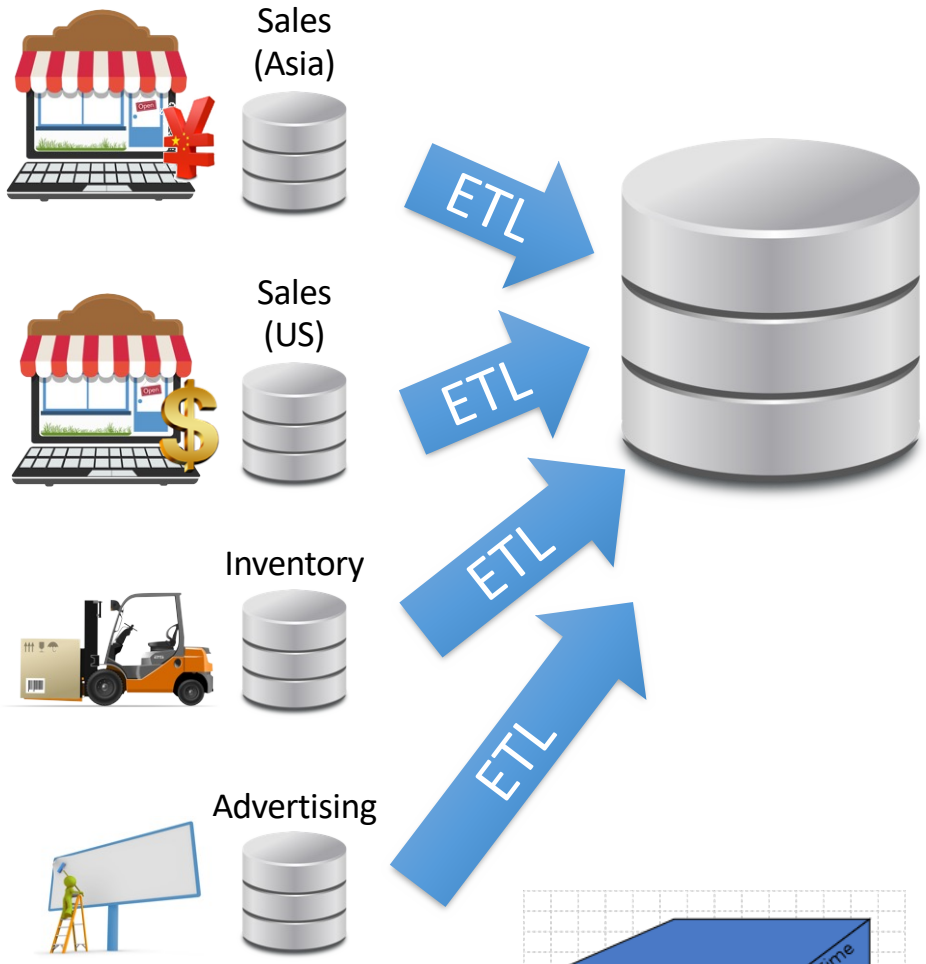
Where are we now

- **How to query a database**
  - SQL (Lec 3-4)
- **How to design a database**
  - ER Model (Lec 8)
  - Design Theory (Lec 9)
  - Relational Model (Lec 2)
- **How to develop a database application**
  - Database Application (Lec 10)
  - Transactions (Lec 12)
- **How to process a SQL query**
  - Relational Algebra (Lec 5)
  - Query Processing (Lec 6)
  - Query Optimization (Lec 12)

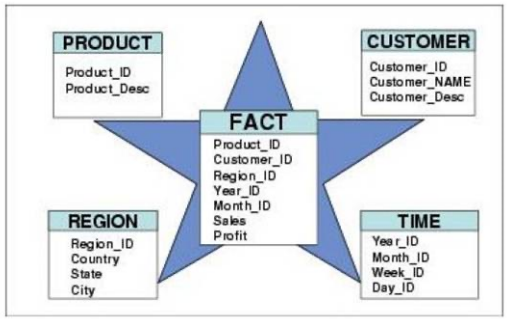
Where to go from here

- Data Warehouse
- Parallel Databases
- Big Data Processing
- Cloud Databases
- Data Science

Data Warehouse

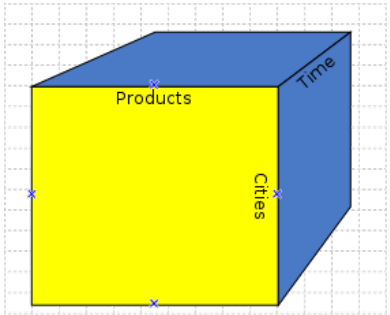


Collects and organizes historical data from multiple sources

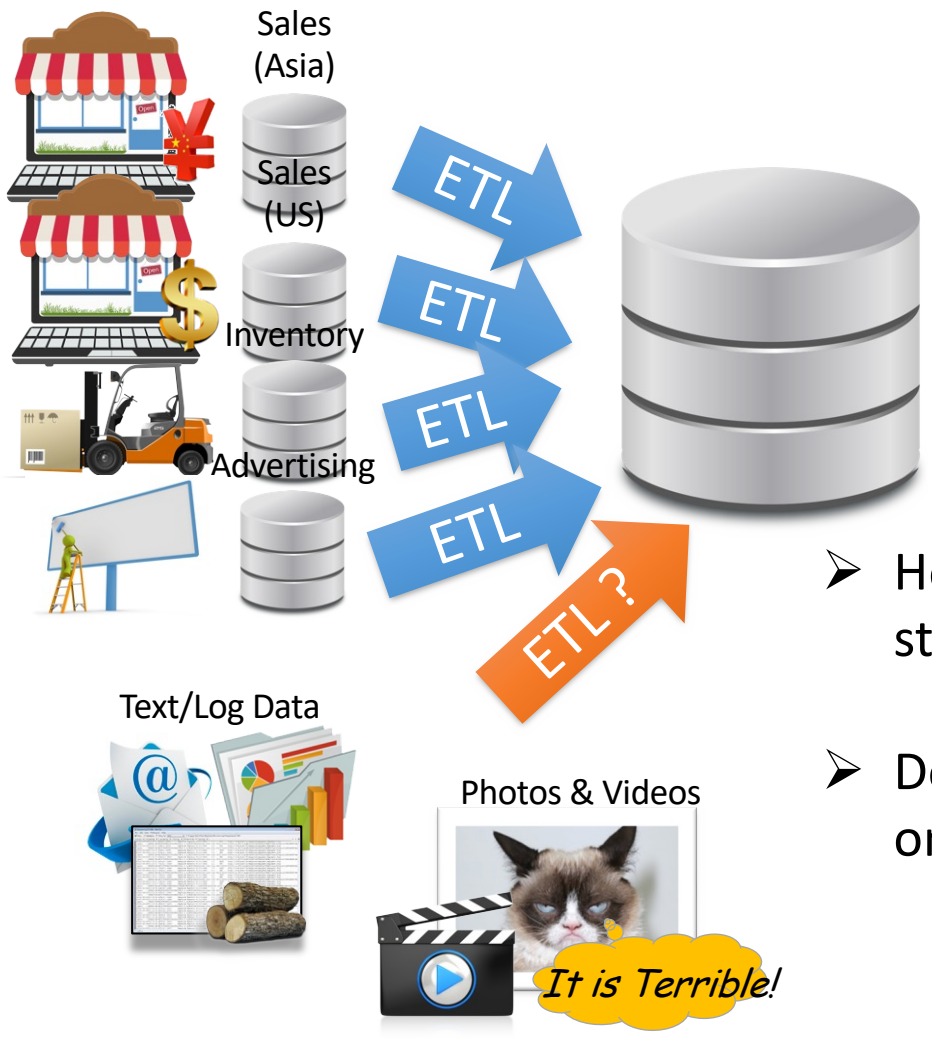


So far …

- Star Schemas
- Data cubes



Data Warehouse



Collects and organizes historical data from multiple sources

- How do we deal with semi-structured and unstructured data?
- Do we really want to force a schema on load?

Sales (Asia)

Advertising

ETL

ETL

ETL

Data Warehouse

How do we **clean** and **organize** this data?

Depends on use …

Collects and organizes historical data from multiple sources

ETL ?

How do we **load** and **process** this data in a relational system?

Depends on use …
Can be difficult …
Requires thought …

Text/Log Data

Photos & Videos

Data Lake *

Store a copy of all the data

- in one place
- in its original "natural" form

Enable data consumers to choose how to transform and use data.

- *Schema on Read*

Enabled by new Tools:
Map-Reduce & Distributed Filesystems

What could go wrong?

*Still being defined...[Buzzword Disclaimer]

The Dark Side of Data Lakes

- Cultural shift: *Curate → Save Everything!*
  - Noise begins to dominate signal
- Limited data governance and planning

  Example:
  hdfs://important/joseph_big_file3.csv_with_json
  - **What** does it contain?
  - **When** and **who** created it?
- No cleaning and verification → lots of dirty data
- New tools are more complex and old tools no longer work

Enter the data engineer

A Brighter Future for Data Lakes

Enter the data engineer

- Data engineers bring new skills
  - Distributed data processing and cleaning
  - DevOps and MLOps
- Technologies are improving
  - SQL over large files
  - Self describing file formats & catalog managers
- Organizations are evolving
  - Tracking data usage and file permissions
  - New job title: data engineers

Where to go from here

- Data Warehouse
- **Parallel Databases**
- Big Data Processing
- Cloud Databases
- Data Science

Why compute in parallel?

- Multi-cores:
  - Most processors have multiple cores
  - This trend will likely increase in the future

- Big data: too large to fit in main memory
  - Distributed query processing on 100x-1000x servers
  - Widely available now using cloud services

Parallel DBMSs

- How to evaluate a parallel DBMS?
- How to architect a parallel DBMS?
- How to partition data in a parallel DBMS?

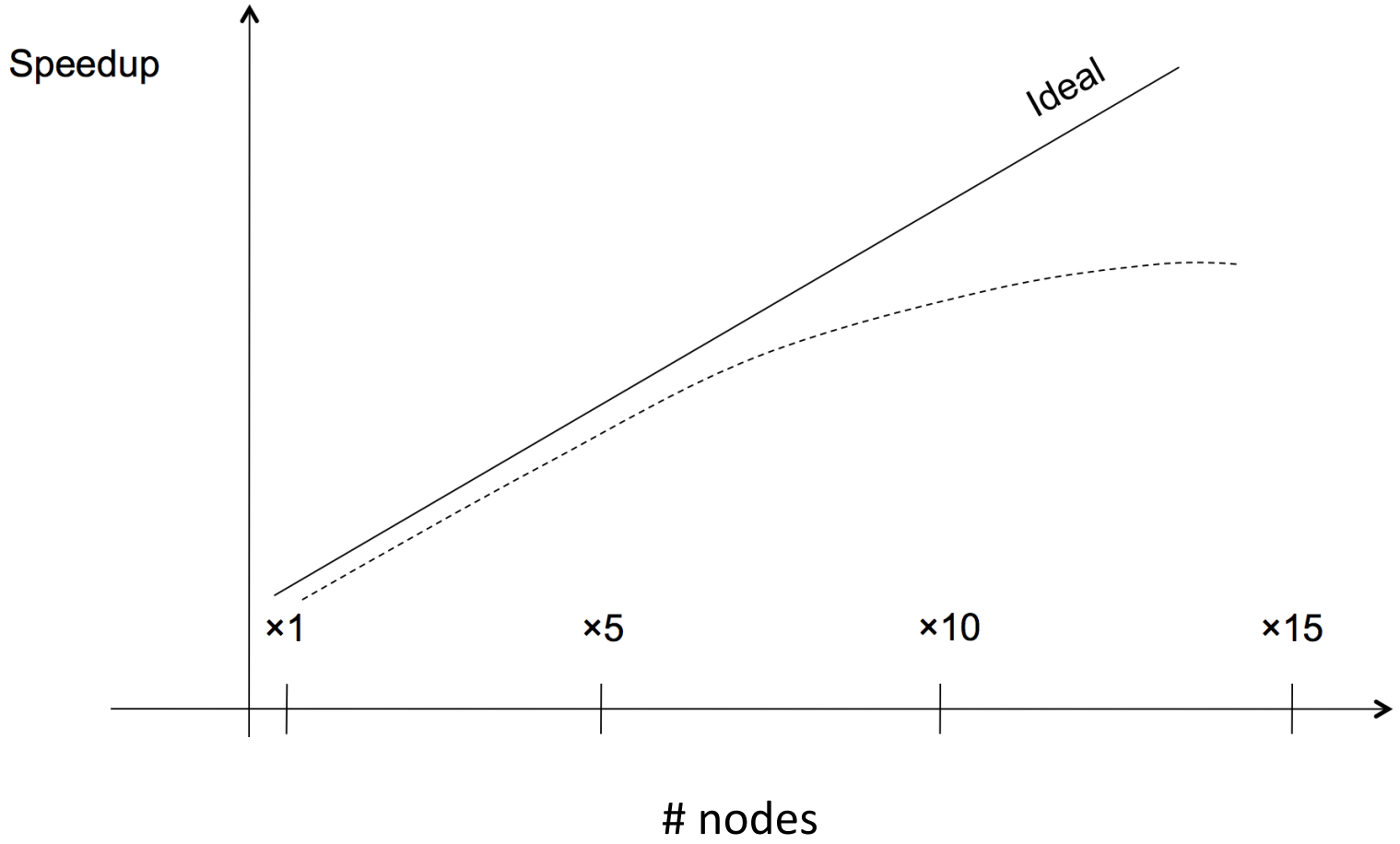
Parallel DBMSs

- **How to evaluate a parallel DBMS?**
- How to architect a parallel DBMS?
- How to partition data in a parallel DBMS?

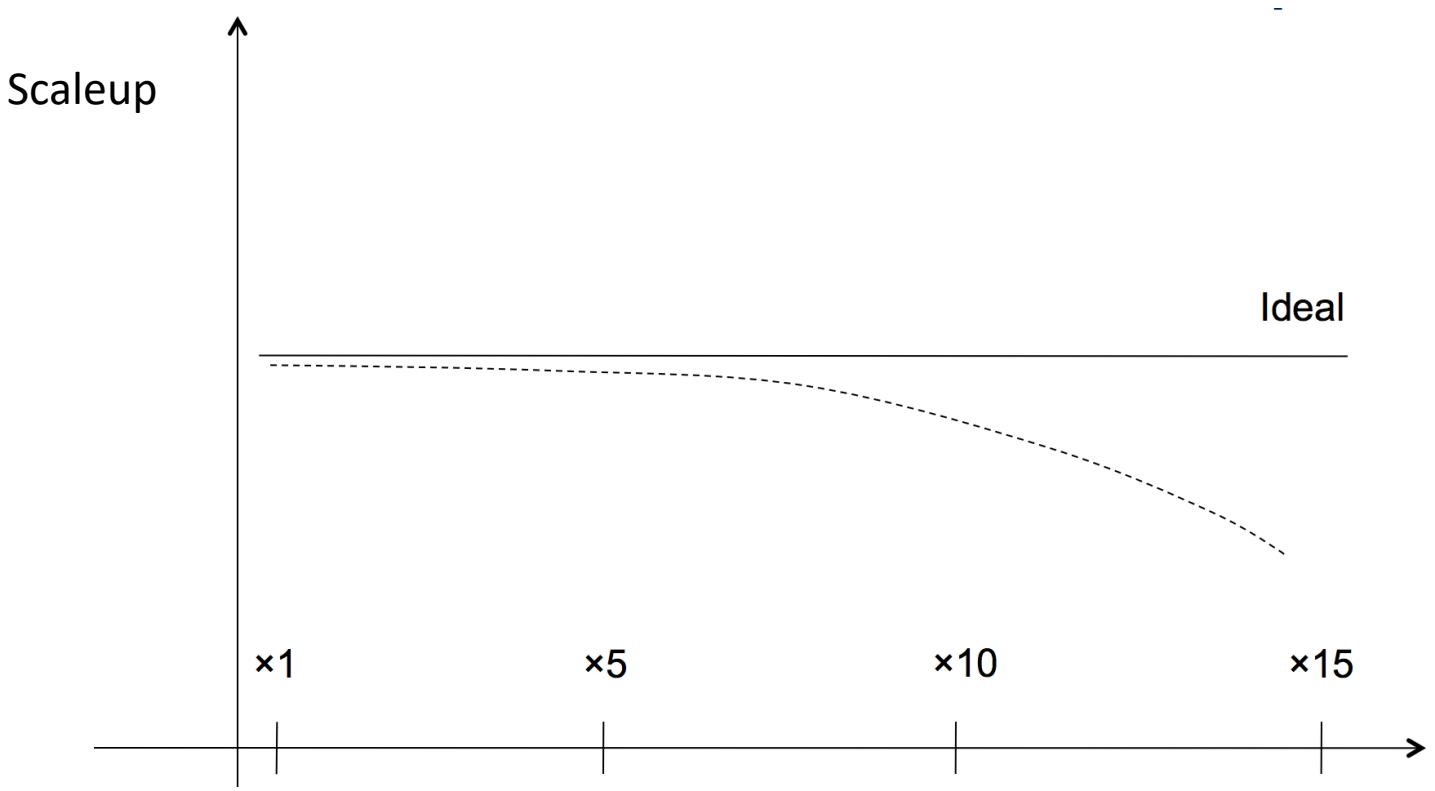
Performance Metrics for Parallel DBMSs

- Nodes = processors, compute
- **Speedup**
  - More nodes, same data → Higher speed
- **Scaleup**
  - More nodes, more data → same speed

Linear v.s. Non-linear Speedup



Linear v.s. Non-linear Scaleup



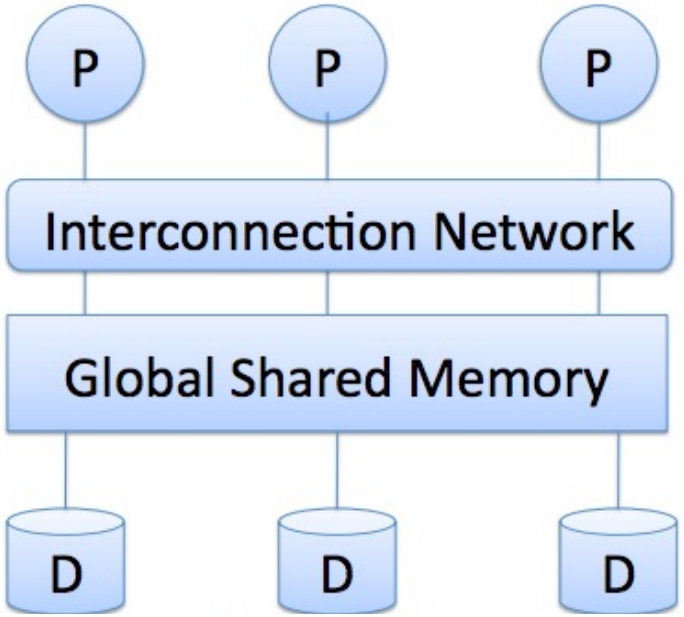
Parallel DBMSs

- **How to evaluate a parallel DBMS?**
- **How to architect a parallel DBMS?**
- How to partition data in a parallel DBMS?

Three Architectures

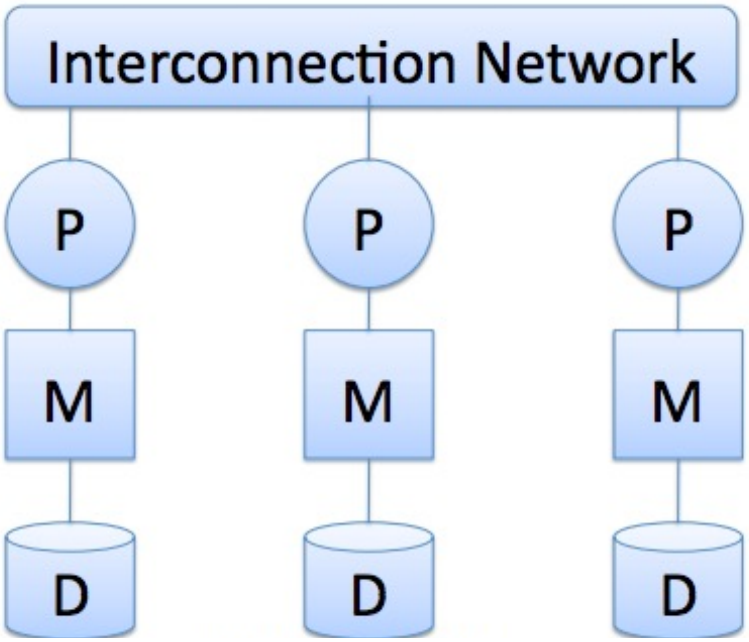
- **Shared Memory**
- **Shared Nothing**
- **Shared Disk**

Shared Memory



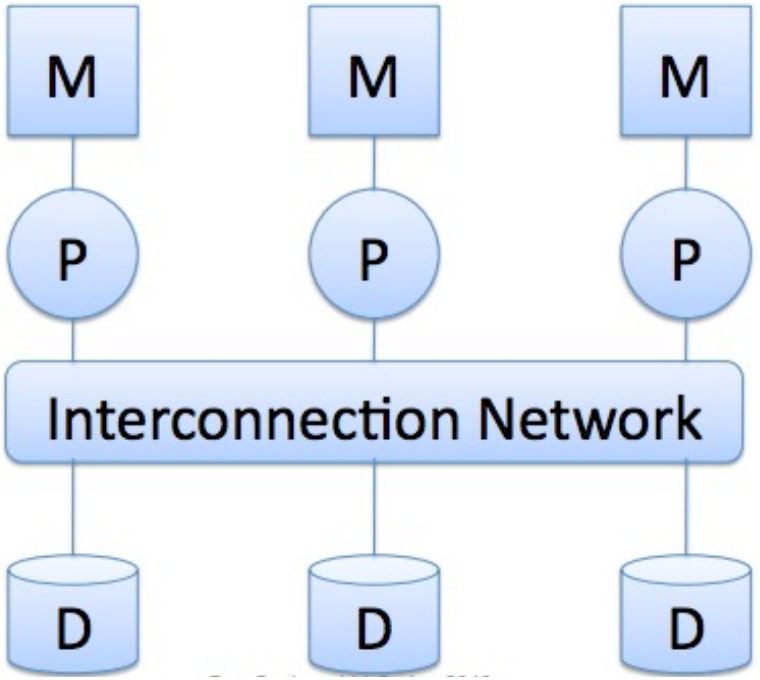
GPU

Shared Nothing



Parallel DBMSs, MapReduce, Spark

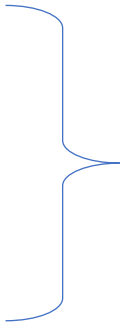
Shared Disk



Azure Synapse Analytics

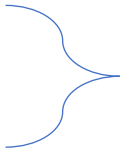
Three Architectures

- **Shared Memory**
- **Shared Nothing**



Computation vs. Communication Trade-offs

- **Shared Disk**



Economic Consideration

Parallel DBMSs

- How to evaluate a parallel DBMS?
- How to architect a parallel DBMS?
- **How to partition data in a parallel DBMS?**

Horizontal Data Partitioning

- **Round Robin**
  - ☺ Load Balancing
  - ☹ Bad Query Performance
- **Range Partitioning**
  - ☺ Good for range/point queries
  - ☹ Data Skew (i.e., Bad Load balancing)
- **Hash Partitioning**
  - ☺ Load Balancing, Good for point queries
  - ☹ Hard to answer range queries

Where to go from here

- Data Warehouse
- Parallel Databases
- **Big Data Processing**
- Cloud Databases
- Data Science

3 Vs of Big Data

- Volume: data size
- Velocity: rate of data coming in
- Variety: data sources, formats, workloads

Big Data Systems

Shared Nothing Architecture

How was Spark created?

UC Berkeley's Research Centers

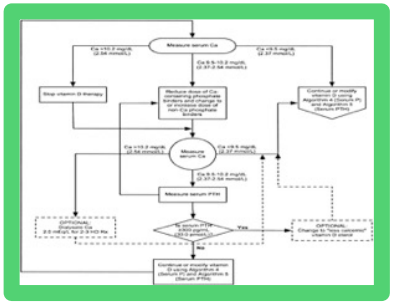
Requirements

- A common vision
- About 5 years
- At least three faculty
- A dozen students

Years	Title	Profs: Director, Co-PIs	Students
1977–1981	X-Tree: Tree Multiprocessor	Despain, Patterson, Sequin	12
1980–1984	RISC: Reduced Instructions	Patterson, Ousterhout, Sequin	17
1983–1986	SOAR: Smalltalk On A RISC	Patterson, Ousterhout	22
1985–1989	SPUR: Symbolic Processing Using RISCs	Patterson, Fateman, Hilfinger, Hodges, Katz, Ousterhout	21
1988–1992	RAID: Redundant Array of Inexpensive Disks	Katz, Ousterhout, Patterson, Stonebraker	16
1993–1998	NOW: Network of Workstations	Culler, Anderson, Brewer, Patterson	25
1997–2002	IRAM: Intelligent RAM	Patterson, Kubiatowicz, Wawrzynek, Yelick	12
2001–2005	ROC: Recovery Oriented Computing Systems	Patterson, Fox	11
2005–2011	RAD Lab: Reliable Adaptive Distributed Computing Lab	Patterson, Fox, Jordan, Joseph, Katz, Shenker, Stoica	30
2007–2013	Par Lab: Parallel Computing Lab	Patterson, Asanovic, Demmel, Fox, Keutzer, Kubiatowicz, Sen, Yelick	40
2011–2017	AMP Lab: Algorithms, Machines, and People	Franklin, Jordan, Joseph, Katz, Patterson, Recht, Shenker, Stoica	40
2013–2018	ASPIRE Lab	Asanovic, Alon, Bachrach, Demmel, Fox, Keutzer, Nikolic, Patterson, Sen, Wawrzynek	40

AMPLab's Vision

Make sense of BIG DATA by tightly integrating algorithms, machines, and people

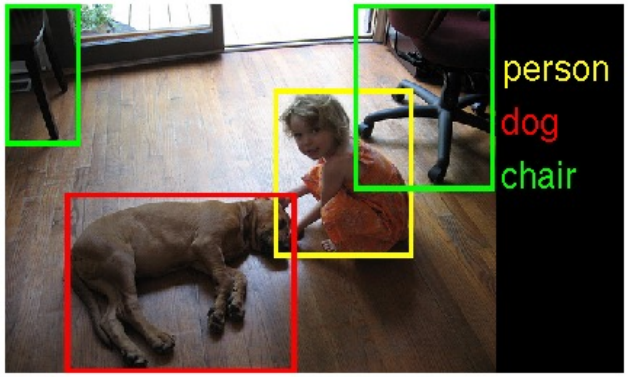


+

+

Example: Extract Value From Image Data

What are in the image?



How to solve the problem?

Deep Learning (Algorithms)
GPU Cluster (Machines)
ImageNet (People)

Spark's Initial Idea

Matei Zaharia

- Algorithms + Machines
  - Run ML Algorithms on Hadoop
- Why is it slow?
  1. The algorithms are iterative (i.e., multiple scans of data)
  2. MapReduce writes/reads data to/from **disk** at each iteration
- Solution
  - Keep data in **memory**

How About Fault Tolerance?

Resilient **Distributed Datasets (RDD)**

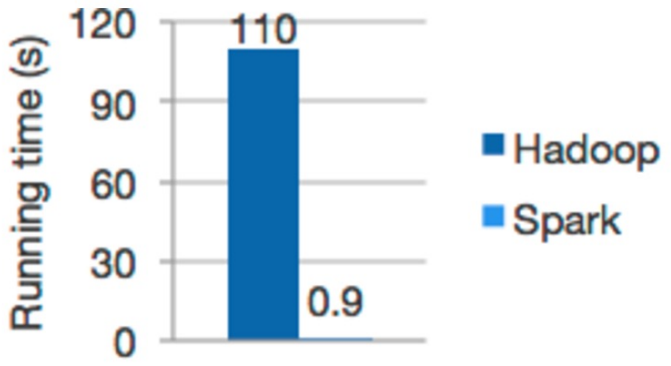
Main Idea: Logging the transformations (used to build an RDD) rather than the RDD itself

Zaharia et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. NSDI 2012: 15-28

Why Spark?

What Makes Spark *Fast*?

- In-memory Computation



Logistic regression in Hadoop and Spark

What you save?

- Serialization/Deserialization
- Compression/Decompression
- I/O cost

“CPU (and not I/O) is often the bottleneck”

Ousterhout et al. Making Sense of Performance in Data Analytics Frameworks. NSDI 2015: 293-307

Hardware Trends

	2010	2016	
Storage	50+MB/s (HDD)	500+MB/s (SSD)	10X
Network	1Gbps	10Gbps	10X
CPU	~3GHz	~3GHz	☹

What Makes Spark *Fast*?

Project Tungsten: Bringing Apache Spark Closer to Bare Metal

by Reynold Xin and Josh Rosen
Posted in **ENGINEERING BLOG** | April 28, 2015

1. Memory Management and Binary Processing
2. Cache-aware computation
3. Code generation

Why Spark?

What Makes Spark *Easy-to-Use*?

- Over 80 High-level Operators

WordCount (Mapreduce)

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WordCount extends Configured implements Tool {

    public static class TokenizerMapper
    extends Mapper<LongWritable, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        @Override
        public void map(LongWritable key, Text value, Context context
                ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
```

```
    public static class IntSumReducer
    extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        @Override
        public void reduce(Text key, Iterable<IntWritable> values,
                Context context
                ) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }

    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new WordCount(), args);
        System.exit(res);
    }

    @Override
    public int run(String[] args) throws Exception {
        Configuration conf = this.getConf();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);

        job.setInputFormatClass(TextInputFormat.class);

        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        TextInputFormat.addInputPath(job, new Path(args[0]));
        TextOutputFormat.setOutputPath(job, new Path(args[1]));

        return job.waitForCompletion(true) ? 0 : 1;
    }
}
```

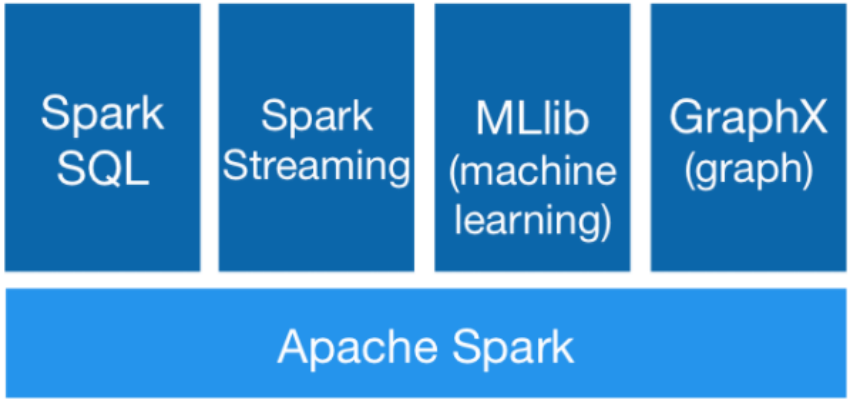
WordCount (Spark)

```
text_file = spark.textFile("hdfs://...")

text_file.flatMap(lambda line: line.split())
    .map(lambda word: (word, 1))
    .reduceByKey(lambda a, b: a+b)
```

What Makes Spark *Easy-to-Use*?

- **Unified Engine**



Easy to manage, learn, and combine functionality

Analogy

Specialized Devices

Unified Device

What Makes Spark *Easy-to-Use*?

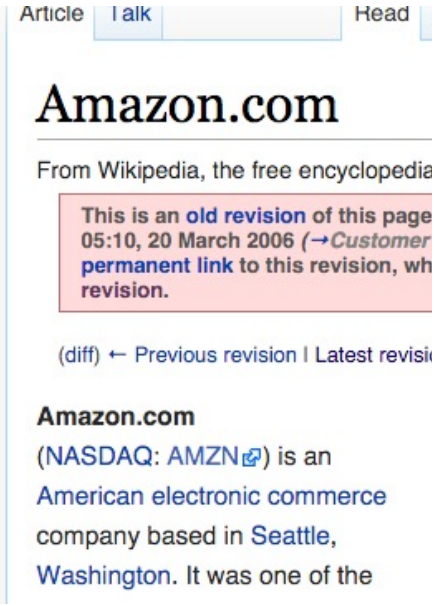
- **Integrate Broadly**

Where to go from here

- Data Warehouse
- Parallel Databases
- Big Data Processing
- **Cloud Databases**
- Data Science

Amazon

- **From Wikipedia 2006**


Article Talk Head

Amazon.com

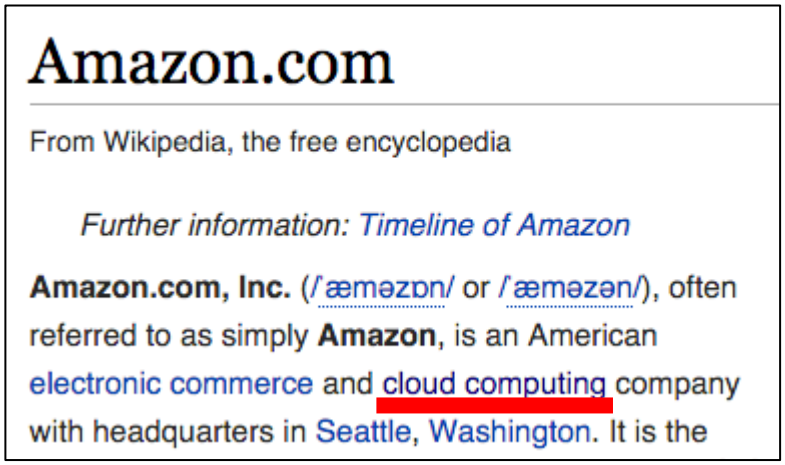
From Wikipedia, the free encyclopedia

This is an old revision of this page 05:10, 20 March 2006 (→Customer permanent link to this revision, wh revision.

(diff) ← Previous revision | Latest revisi

Amazon.com
(NASDAQ: AMZN) is an American electronic commerce company based in Seattle, Washington. It was one of the

From Wikipedia 2016


Amazon.com

From Wikipedia, the free encyclopedia

Further information: Timeline of Amazon

Amazon.com, Inc. (/ˈæməzɒn/ or /ˈæməzən/), often referred to as simply **Amazon**, is an American electronic commerce and cloud computing company with headquarters in Seattle, Washington. It is the

What is Cloud Computing?

- The buzz word before “Big Data”
  - Larry Ellison’s response in 2009 (https://youtu.be/UOEFXaWHppE?t=7s)
  - Berkeley RADLab’s paper in 2009 (https://youtu.be/IJCxqoh5ep4)
- A technical point of view
  - Internet-based computing (i.e., computers attached to network)
- A business-model point of view
  - Pay-as-you-go (i.e., rental)

Three Types of Cloud Computing

CourSys — Application + Cloud = **SaaS** (Software as a service)

Database — Platform + Cloud = **PaaS** (Platform as a service)

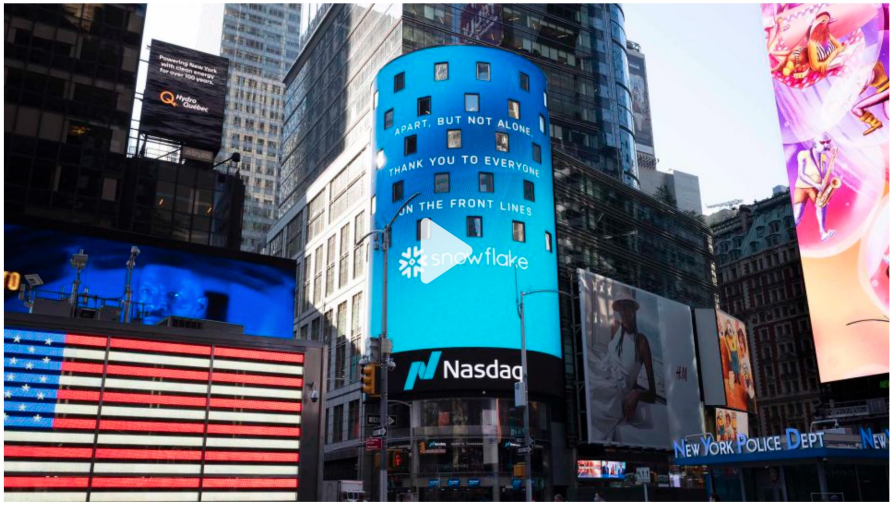
Servers — Infrastructure + Cloud = **IaaS** (Infrastructure as a service)

Cloud Databases

- **Amazon**
  - Aurora
  - Redshift
- **Microsoft**
  - Azure SQL Databases
  - Azure Synapse Analytics
- **Google**
  - Spanner
  - BigQuery

Snowflake shares more than double. It's the biggest software IPO ever

By Paul R. La Monica, CNN Business
Updated 3:37 AM EDT, Thu September 17, 2020



Where to go from here

- Data Warehouse
- Parallel Databases
- Big Data Processing
- Cloud Databases
- **Data Science**

Summary

- Data Warehouse
- Parallel Databases
- Big Data Processing
- Cloud Databases

Acknowledge

- Some lecture slides were copied from or inspired by the following course materials
  - “W4111: Introduction to databases” by Eugene Wu at Columbia University
  - “CSE344: Introduction to Data Management” by Dan Suciu at University of Washington
  - “CMPT354: Database System I” by John Edgar at Simon Fraser University
  - “CS186: Introduction to Database Systems” by Joe Hellerstein at UC Berkeley
  - “CS145: Introduction to Databases” by Peter Bailis at Stanford
  - “CS 348: Introduction to Database Management” by Grant Weddell at University of Waterloo
  - UC Berkeley DS100 Fall 2017