Big Data platforms and cloud elasticity: use cases

Dr Ioannis Konstantinou

ikons@cslab.ntua.gr

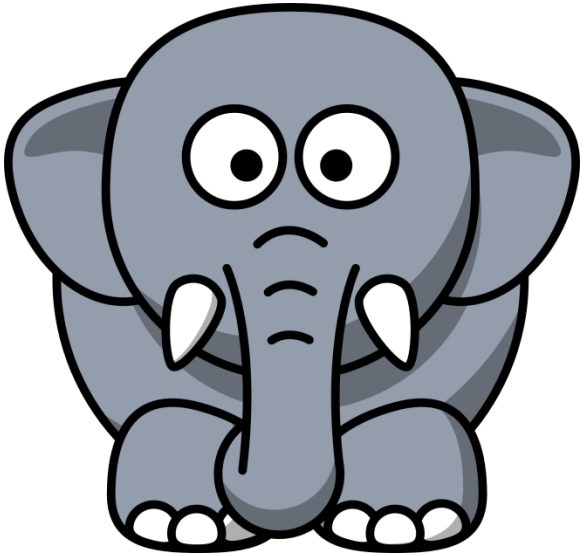
Outline

- **Big Data facts**
- Hadoop Ecosystem
  - MapReduce
  - Hadoop Distributed File System
  - MapReduce example: Word count
  - HIVE and PIG
  - NoSQL datastores and HBase
  - Apache Mahout
- CSLAB Big Data projects

Big Data

Volume

Velocity

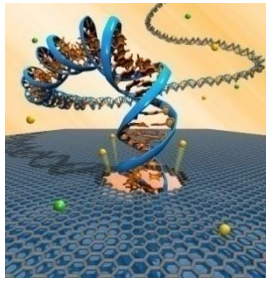


Variety

Veracity

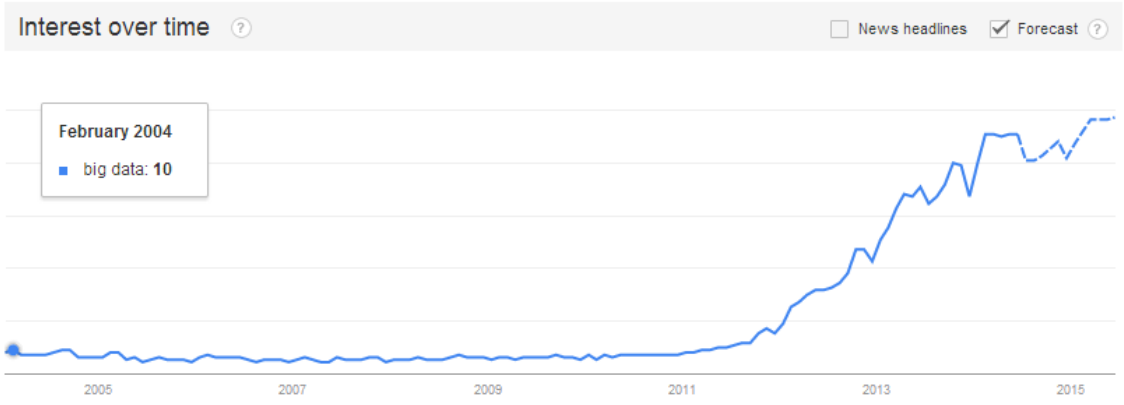
Big Data facts 1/2

- 90% of today's data was created in the past 2 years
- Moore's Law: Data size doubles every 18 months
- YouTube: 13M hours and 700B playbacks in 2010. 48h/min -> 8years/day upload
- Facebook: 20TB/day zipped. 1B shares/day
- CERN/LHC: 40TB/day (15PB/year)



Big Data facts 2/2

- By 2015, 4.4M new Big Data IT jobs
- By 2018, a shortage of 190K big-data experts and 1.5M analysts
- Bad/poor data costs US 600B $/year
- For a Fortune 1000 company, a 10% increase on data usability -> 2B USD increase



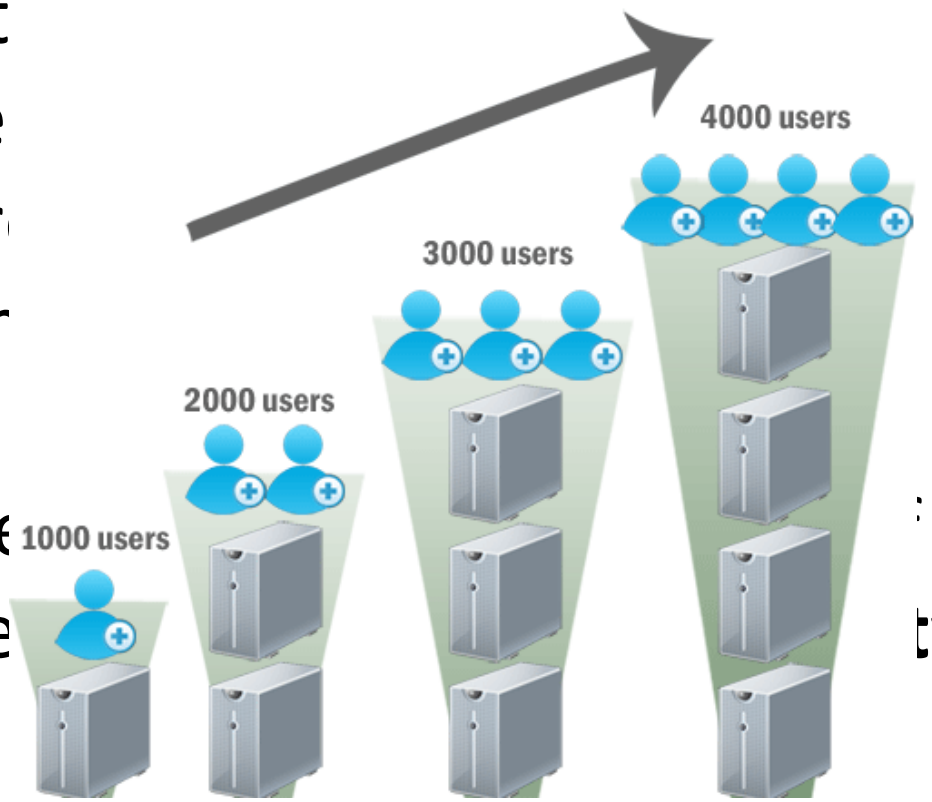
Big Data Application Use Cases

- Retail
  - Monitoring social media to get preferences, customer behavior, product perception, etc.
- Banking/Insurance
  - Risk management, fraud detection, etc.
- Manufacturing
  - Maintenance and repair, supply chain management, etc.
- Advertising/Marketing
  - Responsiveness to campaigns, etc.
- Government
  - Publicizing data
- Media
  - Personalization, archiving, etc.
- Telecommunications
  - Failure prevention, etc.

Not only Google and Facebook have big data needs!!!!

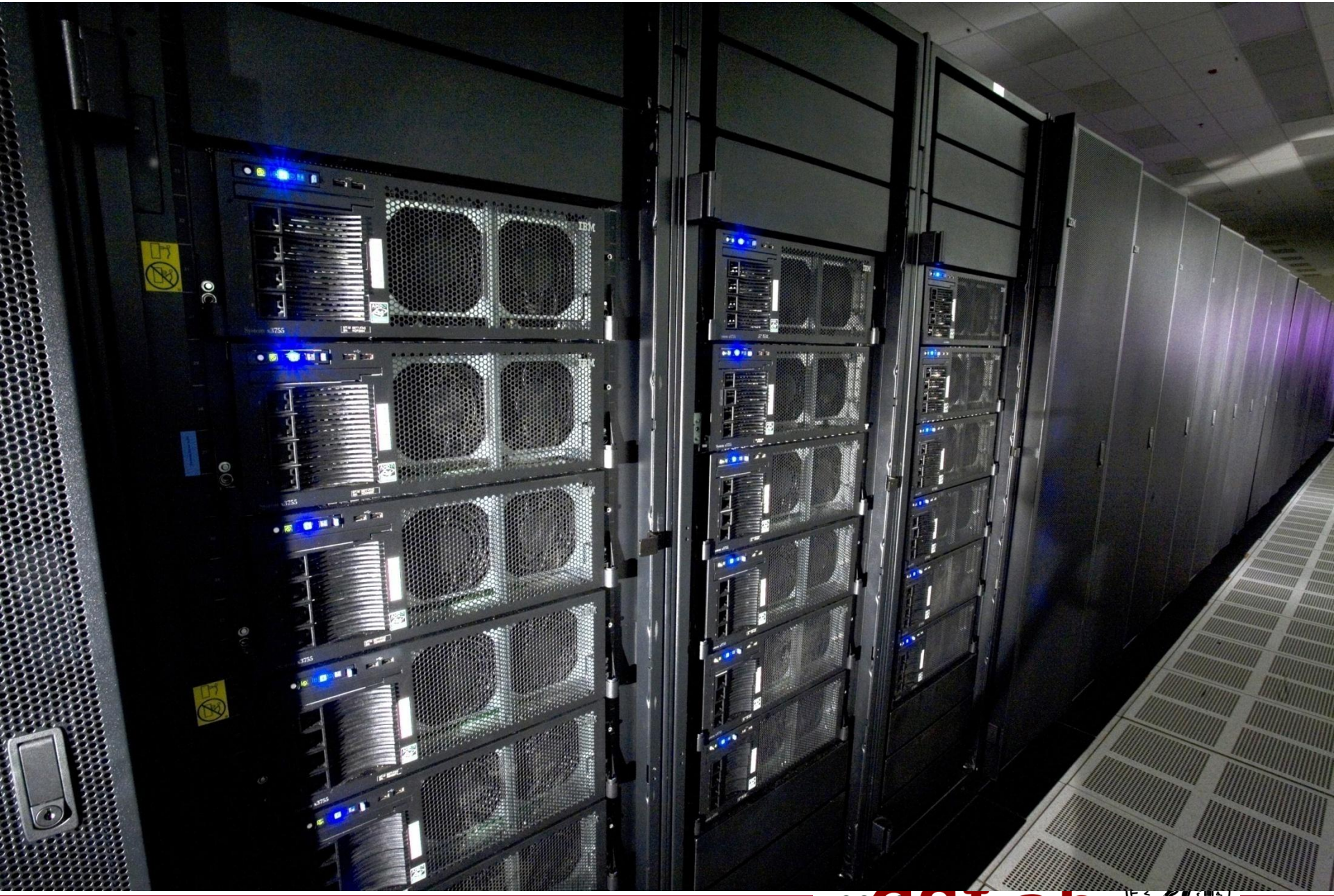
Big Data Sources

- Social net
- Public we
- Data War
- Monitorir

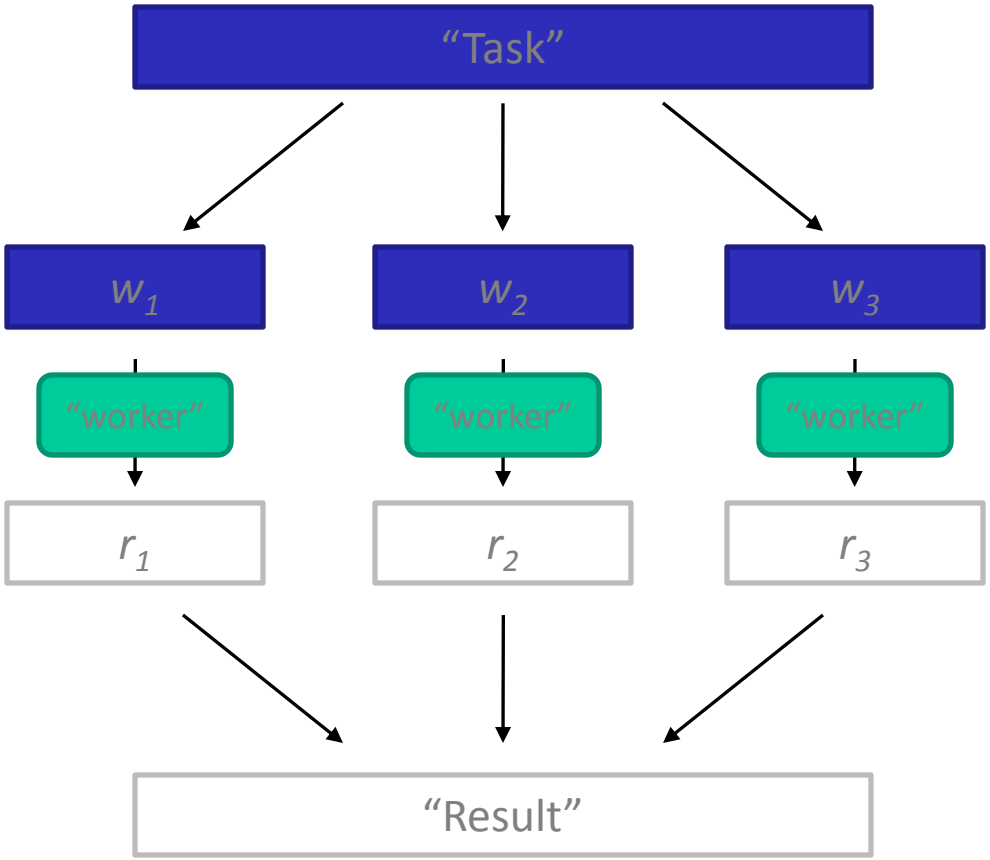


- How to de data?
- Distribute ty.

Source: Wikipedia (IBM Roadrunner)



divide and conquer



Partition

Combine

National Technical University of Athens CSLab

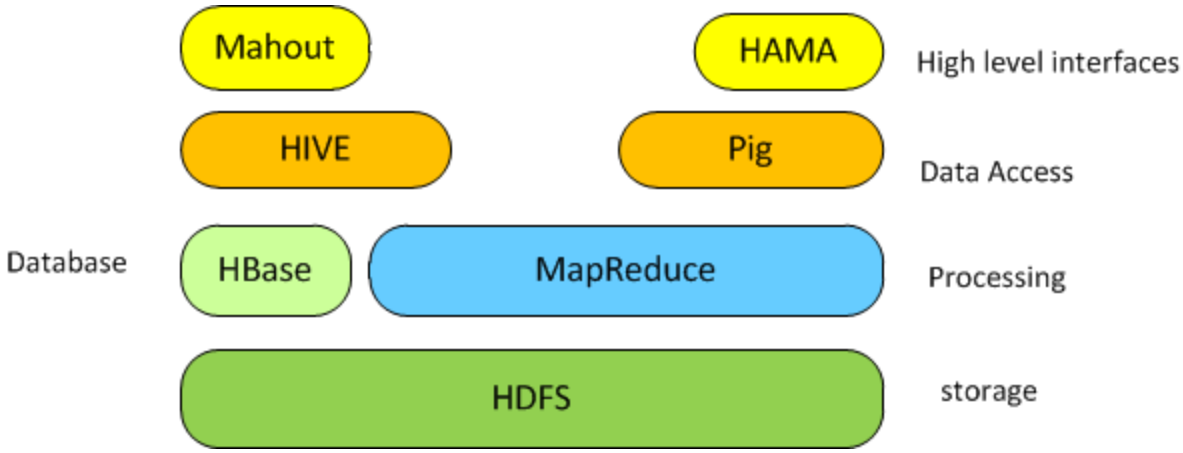
Parallelization Challenges

- How do we assign tasks to workers?
- What happens when
  - We have more tasks than workers?
  - Workers need to share intermediate results?
- How do we summarize intermediate results?
- How do we know that workers have finished?
- What happens when some workers have been terminated?

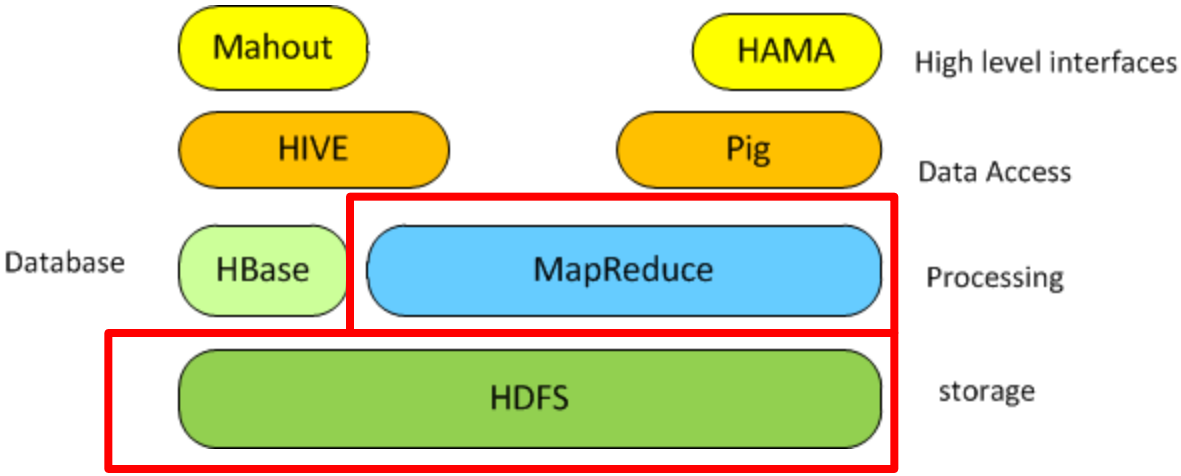
Outline

- Big Data facts
- **Hadoop Ecosystem**
  - MapReduce
  - Hadoop Distributed File System
  - MapReduce example: Word count
  - HIVE and PIG
  - NoSQL datastores and HBase
  - Apache Mahout
- CSLAB Big Data projects

Hadoop Ecosystem



Hadoop Ecosystem



What is MapReduce?

- A programming model and framework
- For the development of applications for
  - Fast and parallel processing of huge data sets
  - Using clusters of commodity hardware
  - Closed-source Google Implementation
  - ’03 and ’04 Scientific papers describing its architecture
- Hadoop: open-source MapReduce implementation
  - http://hadoop.apache.org/

What is Hadoop?

- 2 subsystems: data management (I/O) and computation (CPU):
  - I/O: HDFS (Hadoop Distributed File System)
  - CPU: MapReduce the computation framework running on top of HDFS
  - HDFS is MapReduce's I/O
- A collection of java processes running in a set of different machines.
  - hadoop streaming allows many different programming langs.
- Who uses it?
  - Yahoo!, Amazon, Facebook, Twitter
  - And many others...

HDFS – Distributed File System

- A distributed scalable file system for applications that manage large data sets.
  - Distributed: running on top of a computer cluster
  - Scalable: 10K nodes, 100K files 10PB storage
  - Closed-source optimizations (MapR M5-M7, cloudera)
- Apps view a singe file-system and are un-aware of the underlying cluster.
- Files are split into blocks
- Typical block size 128 MB.

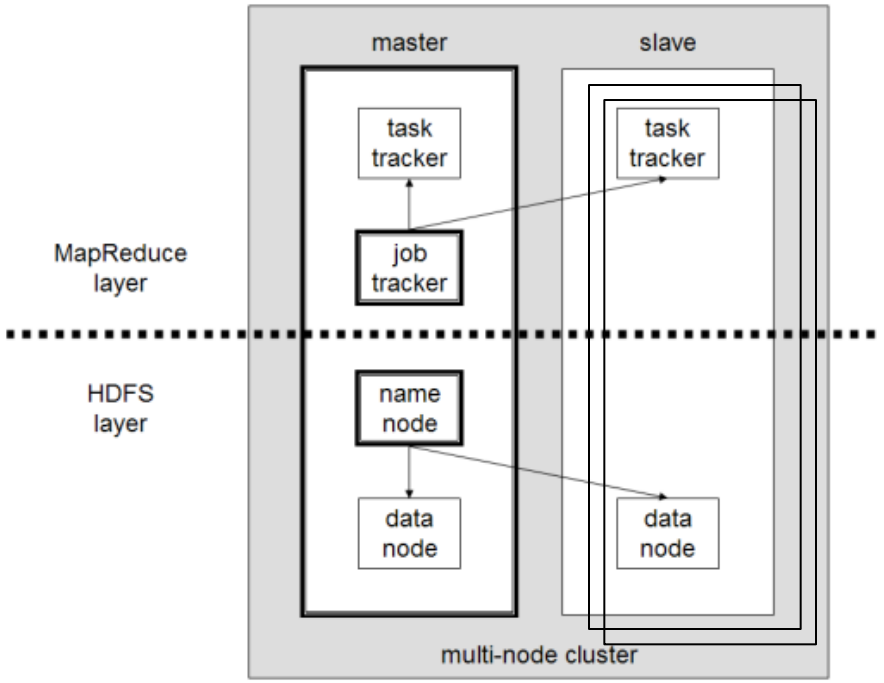
HDFS/MapReduce Architecture

- Master/Slave
  - HDFS: One NameNode for many DataNodes
    - NameNode: registry keeping File->Datanode locations (like FAT or ext3 block metadata)
    - DataNodes: only raw file chunks
  - MapReduce: One JobTracker to manage Many TaskTrackers

-NameNode and JobTracker running on master

-DataNode and TaskTracker running on slaves

- Data locality

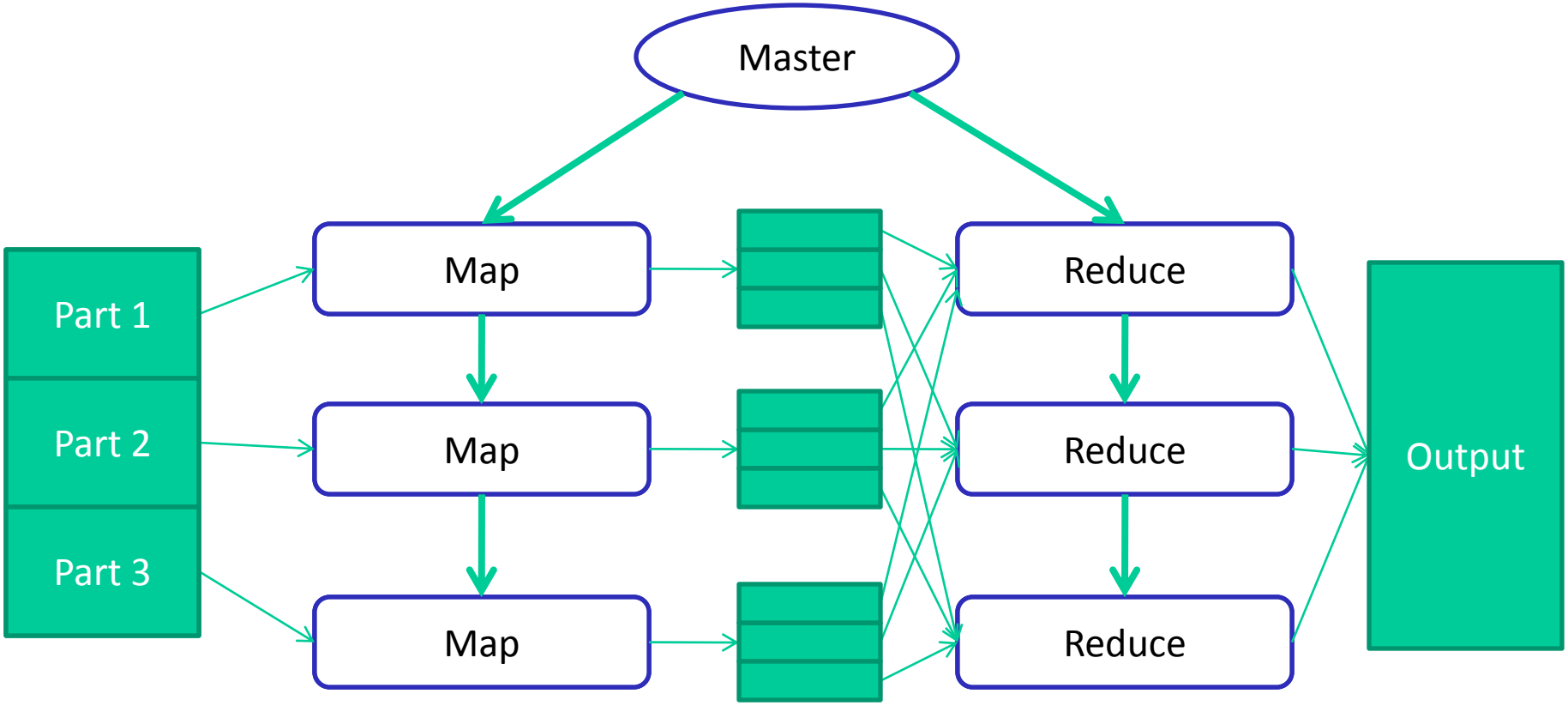


MapReduce

- Problem “breaks “ in two phases, Map and Reduce
- **Map**: Non overlapping input data chunks that contain <key,value> pairs are assigned in different processes (mappers) which produce a set of intermediate <key,value> results
- **Reduce:** Map results are fed to a usually smaller number of processes (reducers) which “summarize” input results into less <key,value> pairs.

Job Execution

- Data is uploaded to HDFS and are split into MB size chunks.
  - Every chunk contains a number of <key,value> pairs
- Tasktrackers execute a copy of the M/R program in different data chunks (data parallel)
- One node is the master JobTracker. It assigns tasks to workers. Tasks can be map or reduce



Example: Word Count 1/3

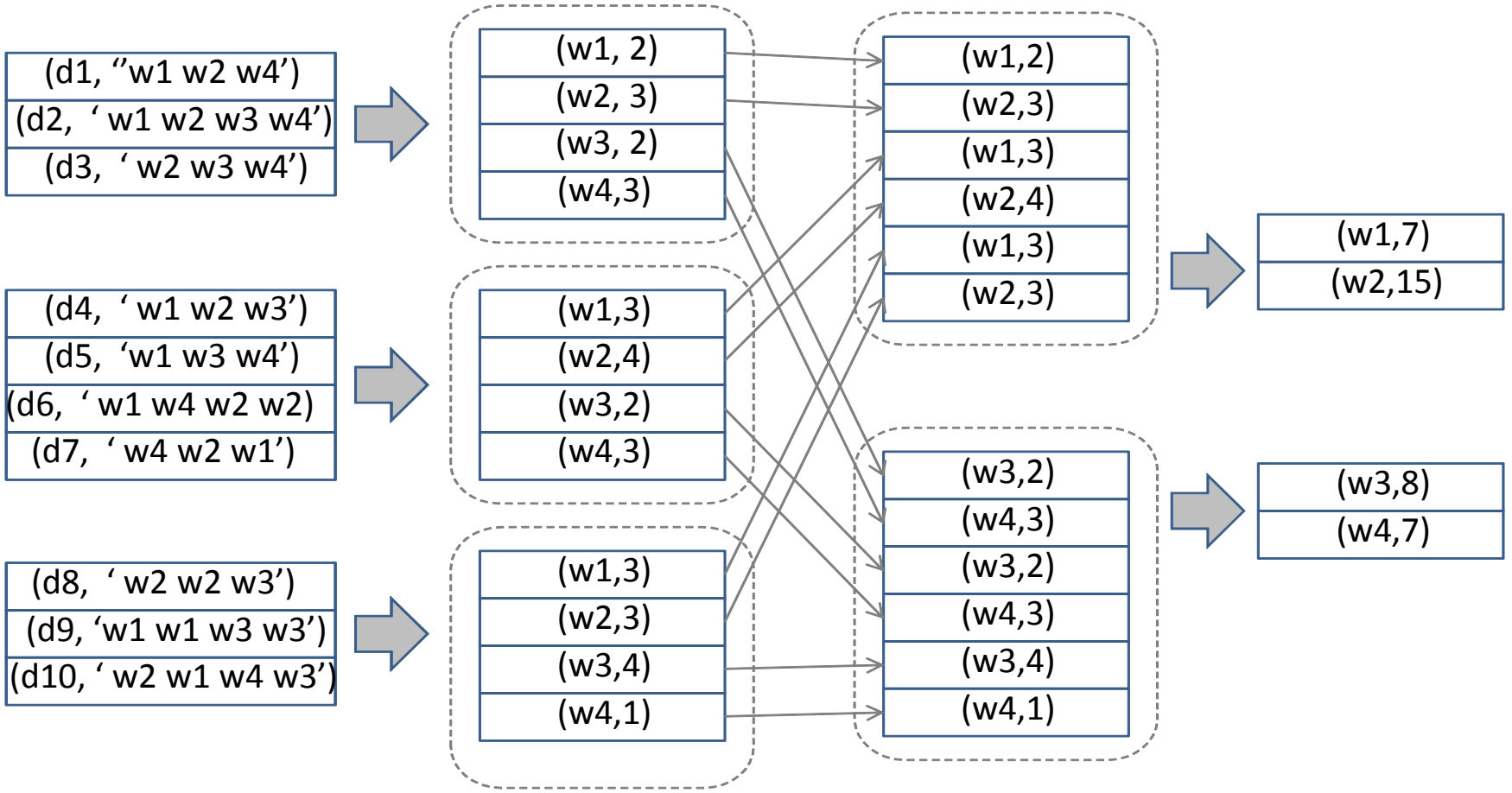
- Target: count word frequency in a large text corpus
- Possible use: Find popular urls in webserver logfiles
- Implementation plan:
  - "Upload" texts to MapReduce HDFS
  - Write a map and a reduce function
  - Execute a MapReduce job
  - Collect results

Example: Word Count 2/3

```
map(key, value):
// key: document name; value: text of document
    for each word w in value:
     emit(w, 1)

reduce(key, values):
// key: a word; value: an iterator over counts
     result = 0
     for each count v in values:
          result += v
     emit(result)
```

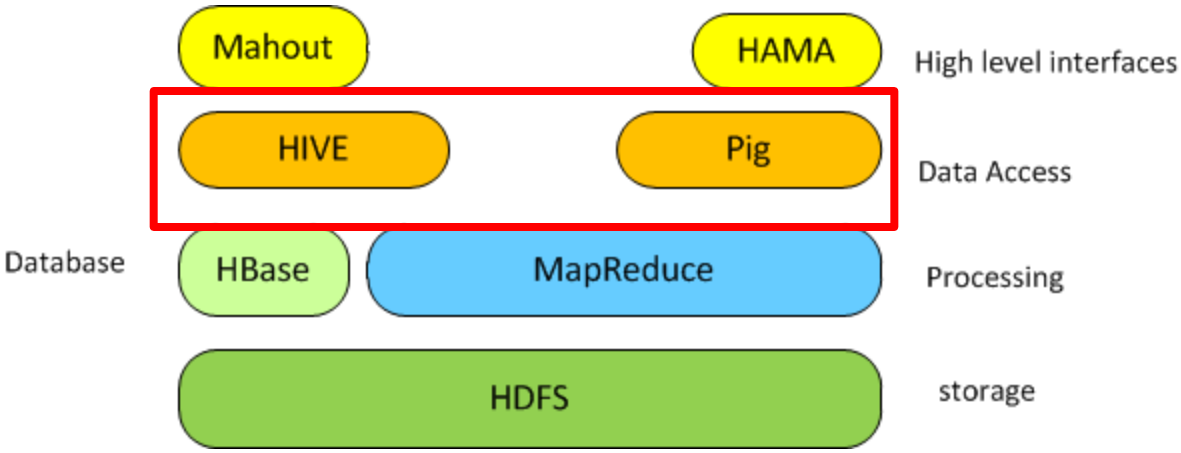
Example: Word Count 3/3



Who uses it? What for?

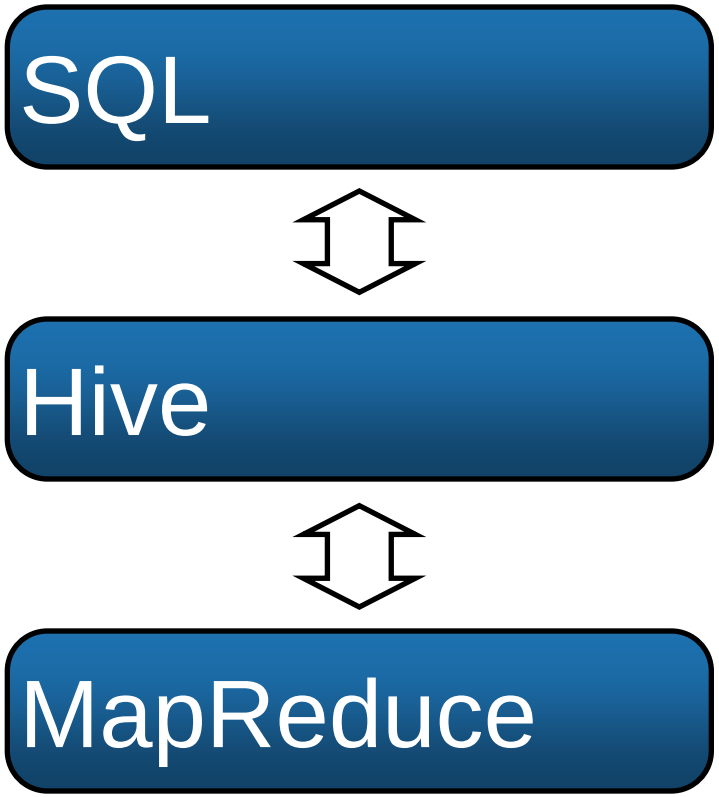
- Amazon, e-bay
  - product search indices, analytics
- AOL
  - behavioral analysis and targeting
- LinkedIn
  - discovering People You May Know
- Facebook
  - Storage of internal logs for reporting/analytics and machine learning
- Twitter
  - Processing of tweets

Hadoop Ecosystem



Hadoop HIVE 1/2

- Data Warehouse System for Hadoop
- Data Aggregation
- Ad-Hoc Queries
- SQL-like Language (HiveQL)
- Developed at facebook

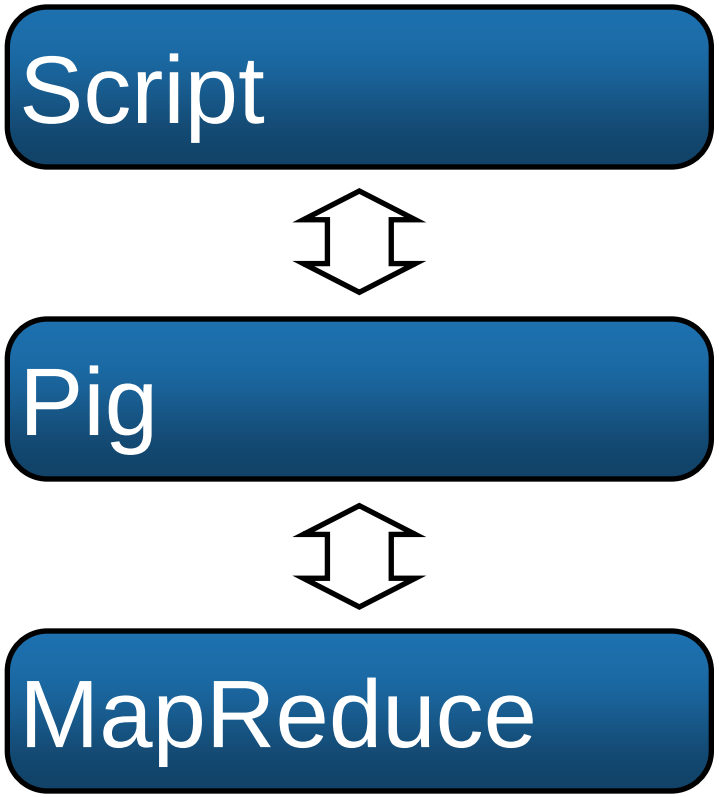


Hadoop HIVE 2/2

- Data Warehouse on top of Hadoop
- **SQL-like** language HiveQL
- Queries implicitly converted into **MR jobs**
  - Distributed query execution
- Different storage types
  - plain text, RCFile, Hbase ...
- Supports all major primitive types
  - integers, floats, doubles, strings
- Supports complex types
  - maps, lists, structs
- Built-in extensible user defined functions

Hadoop PIG

- Data Warehouse System for Hadoop
- Data Aggregation
- Ad-Hoc Queries
- High-Level Scripting Language (Pig Latin)
- Developed at Yahoo



HIVE/PIG examples

2 Datasets: movies (movie info), movierating (user ratings for movies)
Problem: find average rating for each movie.

HIVE

```
CREATE TABLE newmovie (id INT, name STRING, year INT, numratings INT,
    avgrating FLOAT);
INSERT OVERWRITE TABLE newmovie
      SELECT id, name, year, COUNT(1), AVG(rating)
      FROM movie
      JOIN movierating ON movie.id = movierating.movieid
          GROUP BY id, name, year;
```

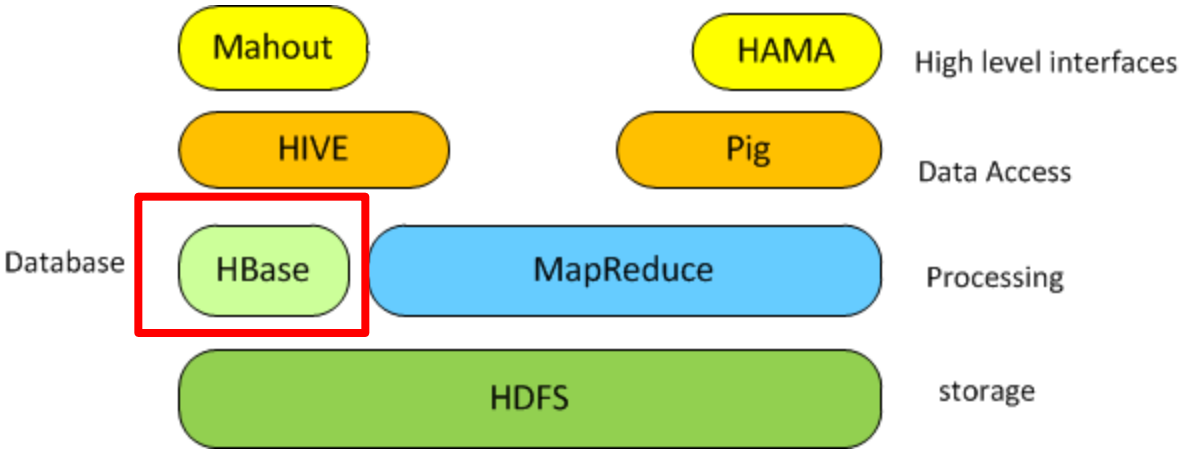
PIG

```
movierating = LOAD 'movierating' AS (userid, movieid, rating:INT);
groupmr = GROUP movierating BY movieid;
ratings = FOREACH groupmr GENERATE group AS movieid,
     COUNT(movierating.rating) AS numratings, AVG(movierating.rating) AS avgrating;
movie = LOAD 'movie' AS (id, name, year);
mr = JOIN movie BY id, ratings BY movieid;
result = FOREACH mr GENERATE id, name, year, numratings, avgrating;
STORE result INTO 'ratedmovie';
```

Who uses them? What for?

- AOL, LinkedIn, Salesforce, Twitter, Yahoo
  - Analytics and batch data processing
  - People you may know (Linked in)
  - Custom UDF development
  - Usage log processing, tweet mining
  - >40% of Yahoo's Hadoop Jobs are Pig jobs

Hadoop Ecosystem



NoSQL

- Modify legacy RDBMS model to something more
  - Scalable
  - Available/Fault tolerant
- No–SQL = threw away SQL
- Not–only–SQL = Require more for different applications
- No relations, but simple <key,value> pairs
- A value can be anything from a string entry, to multiple columns or files.
- Usually indexed by key.

NoSQL Characteristics 1/2

- Different consistency levels (non ACID)
- Schema-less
- Queries
  - Complex ones are translated into M/R jobs
  - Support for simple get/select key
- Data insertion
  - Simple puts (key, value)
  - Bulk insert support

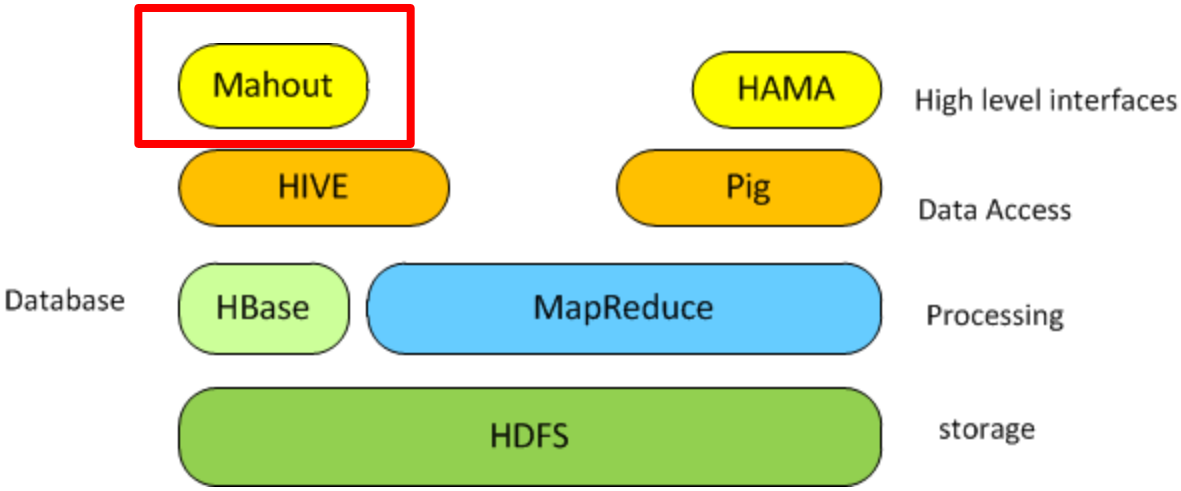
Hadoop NoSQL: HBase

- Google Bigtable Clone (NoSQL “pioneer”)
- HBase = Bigtable open sourced
- Top level apache project
- Goes with Hadoop
- Can be used by Hadoop MapReduce as I/O
- Java written

Who uses it? What for?

- Facebook
  - Provide messages infrastructure
- Twitter
  - read/write backup of all mysql tables in backend
  - timeseries DB for cluster-wide monitoring/performance
- Yahoo
  - document fingerprint store for detecting near-duplications
- Celer technologies
  - Fast and adaptable financial trading framework
  - storing all financial data for trading, risk, clearing in a single data store

Hadoop Ecosystem



Apache Mahout

- A library collection for scalable machine learning
  - "Machine Learning is programming computers to optimize a performance criterion using example data or past experience"
    Intro. To Machine Learning by E. Alpaydin
- Scalability with Hadoop (latest version is based on Spark)
- Mahout algos also utilize Map-Reduce algos: linear scalability
- Latest release version: Mahout 0.9

What does it support?

- 3C + FPM + 0 = Mahout
  - Clustering
  - Classification
  - Collaborative Filtering
  - Frequent Pattern Mining
  - Other

Clustering

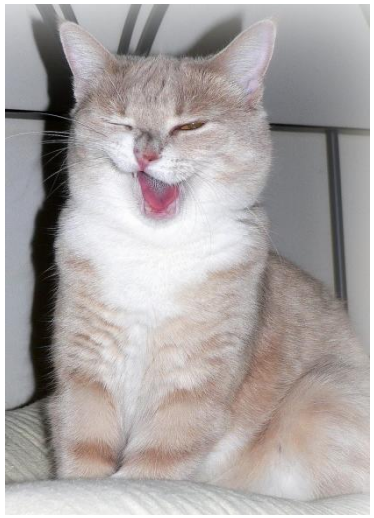
- Grouping based on a similarity notion

- Grouping similar objects
- Meaning of similarity: distance metric:
  - Euclidean
  - Cosine
  - Tanimoto
  - Manhattan

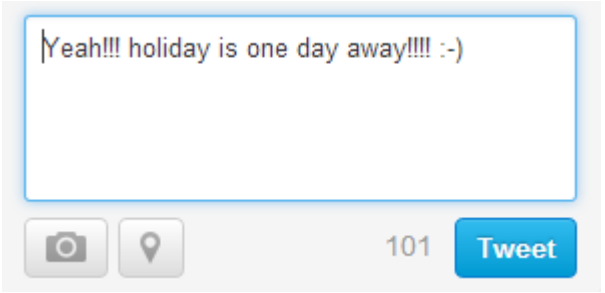
Classification

- Identify the “type” of a new object based on its characteristics (features)
- Types are predetermined

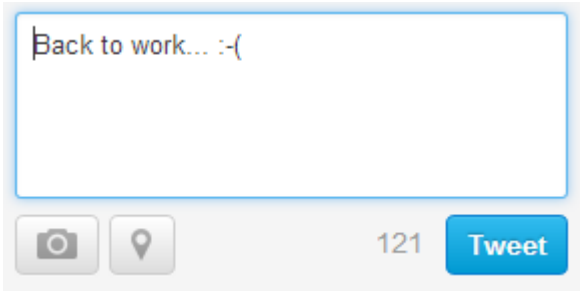
Dog



Cat



Positive



Negative

National Technical University of Athens CSLab

Classification

- Supervised learning: offline model training from manually annotated objects (training set)
- “Guess” a type of a new object according to its extracted features and existing model.
  - Twitter sentiment analysis
  - Many ☺=> Positive, Many ☹ => Negative

Collaborative filtering (CF)

- Recommenders
- User preferences prediction according to:
- Its options so far
- Options of people similar to her (collaborative)
- Propose limited options (filtering)

National Technical University of Athens CSLab

Who uses it? What for?

- AOL
  - Shopping recommendations
- Amazon
  - Amazon personalization platform
- Foursquare
  - Use it for its recommendation engine
- Twitter
  - User interest modeling (Using Mahout's Latent Dirichlet allocation algorithm implementation)
- Yahoo
  - Mail uses Mahout's Frequent Pattern Set Mining for anti-spam filtering

Outline

- Big Data facts
- Hadoop Ecosystem
  - MapReduce
  - Hadoop Distributed File System
  - MapReduce example: Word count
  - HIVE and PIG
  - NoSQL datastores and HBase
  - Apache Mahout
- **CSLAB Big Data projects**

Infrastructure to play with 1/2

- Clusters
  - 32-node dual Intel Xeon E5335 (256 CPUs), Gbit Ethernet / 10Gbps Myrinet
  - 6-node dual Intel Xeon X5650 (72 cores, 144 CPUs), Gbit Ethernet, Infiniband and NVIDIA Fermi FM105 GPUs
  - 8-node, dual Intel Xeon (16 cores, 32 CPUs, SMT enabled), Gbit Ethernet / Myrinet
- Servers
  - 5 Intel Xeon-based servers (various generations) totaling 168 CPUs
  - 2 UltraSparc T1 and T2 servers, totaling 88 CPUs
- Openstack private cloud IaaS
  - big data virtual infrastructures on the fly
  - A total of 160 vCPUS, 368GB RAM on 6 physical servers

In total: *more than 650 CPUs , more than 30TB of storage*

Infrastructure to play with 2/2

- ~okeanos Public Cloud http://okeanos.grnet.gr
- IaaS resources for the academic community

~20 racks

18860 vCPUs

31995.7 GB memory

661077.3 GB storage

On-going Big data projects

- EU and GR funded research programs
  - CELAR, ASAP, ARCOMEM, MoDisSENSE, Clarin
- Research works
  - Tiramola, H2RDF, Datix, DBalancer

CELAR

- **Elasticity**: Ability of a big-data system to expand or contract dedicated resources using cloud infrastructure
- Fully automated, Fine-grained, Real-time
- Use cases: Cloud gaming and cancer detection pipeline

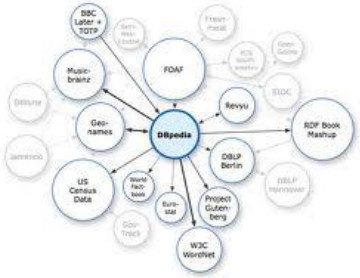
ASAP

- Adaptable Scalable Analytics Platform
- 1 V of Big-data: Variety (data structure, systems, processing models, etc)
- Offer a unified framework for analytics
  - No single engine, datastore
  - Adaptable to user defined settings :cost, time, etc
- Use cases: BI on Telecom Data and Analytics on Web Data

ARCOMEM

- Tools for archivists to crawl the social web and enrich archives with social semantics
- Our contribution: the ARCOMEM Database
  - Hbase for raw crawled content

  - H2RDF for semantic annotations (in the form of rdf triples)



National Technical University of Athens CSLab

MoDisSENSE

- Big Data from: Social networking footprints and Mobile-related generated content
- Provide **personalized user experience**
  - socially enhanced queries: “Find restaurants near me that my friends like”
- **Extract knowledge** from textual/spatial information
  - Find “trending” points of interest (POIs)
  - Auto-augment POI info with user created content
  - Detect new POIs from user trajectories

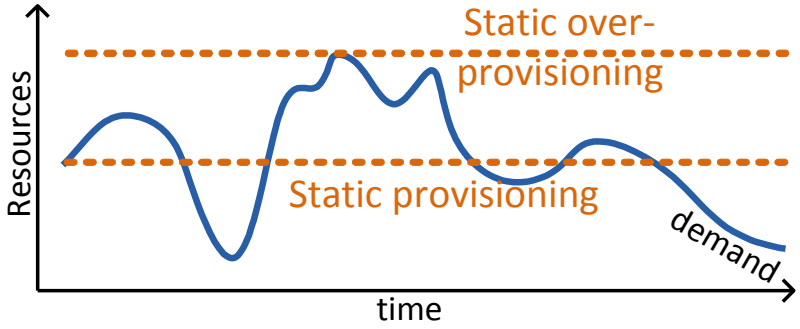
CLARIN

- Repository for linguistic datasets/services
  - Storing, analyzing and processing large datasets
- On top of ~okeanos
- Elastic architecture
  - Web farm for serving queries
  - DB cluster and Solr Cloud for metadata management (~GB scale)
  - Pithos (S3 like storage) for dataset storage (~TB scale)
  - Ansible scripts automate infra deployment
  - Automatic deployment of Hadoop clusters for arbitrary service execution using workflows (CAMEL)

TIRAMOLA

- Unpredictable and variable workloads
  - Sport web-sites, web-apps like taxisnet, etc.
  - Over-provisioning is costly
  - Under-provisioning leads to outages

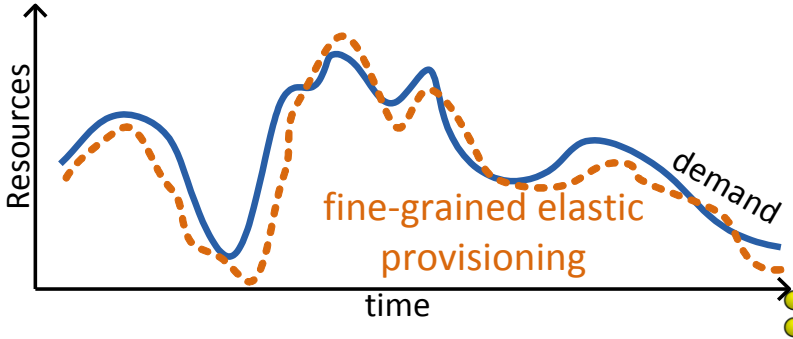
Static



Costly: Idle resources

Outage Risk

TIRAMOLA Elasticity



True Elasticity
Resources follow demand

National Technical University of Athens CSLab

TIRAMOLA

- Cloud enabled framework that allows apps
  - Define elasticity requirements
  - Monitor performance
  - Adaptively grow/shrink resources (multi-grained elasticity)
- Machine learning detects optimal state
  - Based on reward functions, current and prev. system states
- Best Paper award at CCGRID 2013

H_2RDF

- Elastic, Join-Scalable Billion Triple Store
  - Based on Hadoop and Hbase
  - Open sourced at h2rdf.googlecode.com
- Adaptive query planner/executor
  - Centralized (fast) vs MapReduce (scalable) execution
  - MapReduce based executions of multi-way Merge and Sort-merge Join algorithms
- Adding more nodes increase:
  - Both query throughput and execution time
  - Storage space
- Easily scales up to 14B triples (2.5 TB) in a 75-node cluster
- Presented at WWW'2012 and IEEE BIGDATA 2013

DATIX: Scalable network analytics

- Passive sflow IXP traffic data (TB scale)
  - Lines containing ip pairs, payload, protocol info, etc
- External data sources: IP to ASN, DNS, Country, etc
- Distributed processing using HIVE/Hadoop
- Combine sflow/external data to extract lists with:
  - Top contributors for a specific time period
  - Contributors can be unique IPs, ASes with high traffic, most visited Web servers, etc

DBalancer: Balancing NoSQL datastores

- Skewed data access/placement affects NoSQL data-stores
- Current balancing approaches: non automated, system specific, centralized
- DBalancer: Generic, configurable system for real-time on-line adaptive load balancing
  - Support for Cassandra
  - Hooks to support any other NoSQL
  - Implementation of 3 existing algorithms

Questions