Big Data Infrastructures & Technologies

Mark Raasveldt

About Me

- PhD in Computer Science from Leiden University
- Senior Researcher at CWI Database Architectures
- CTO and co-founder of DuckDB Labs

http://markraasveldt.com

@ mraasveldt

Topics Today

1. Introduction
   Practicum
2. Data Storage & ETL
   Practicum
3. SQL on Big Data
   Practicum

Organization

1. Materials & Slides online

https://github.com/Mytherin/bigdata-infrastructure-technology

Big Data Infrastructures & Technology

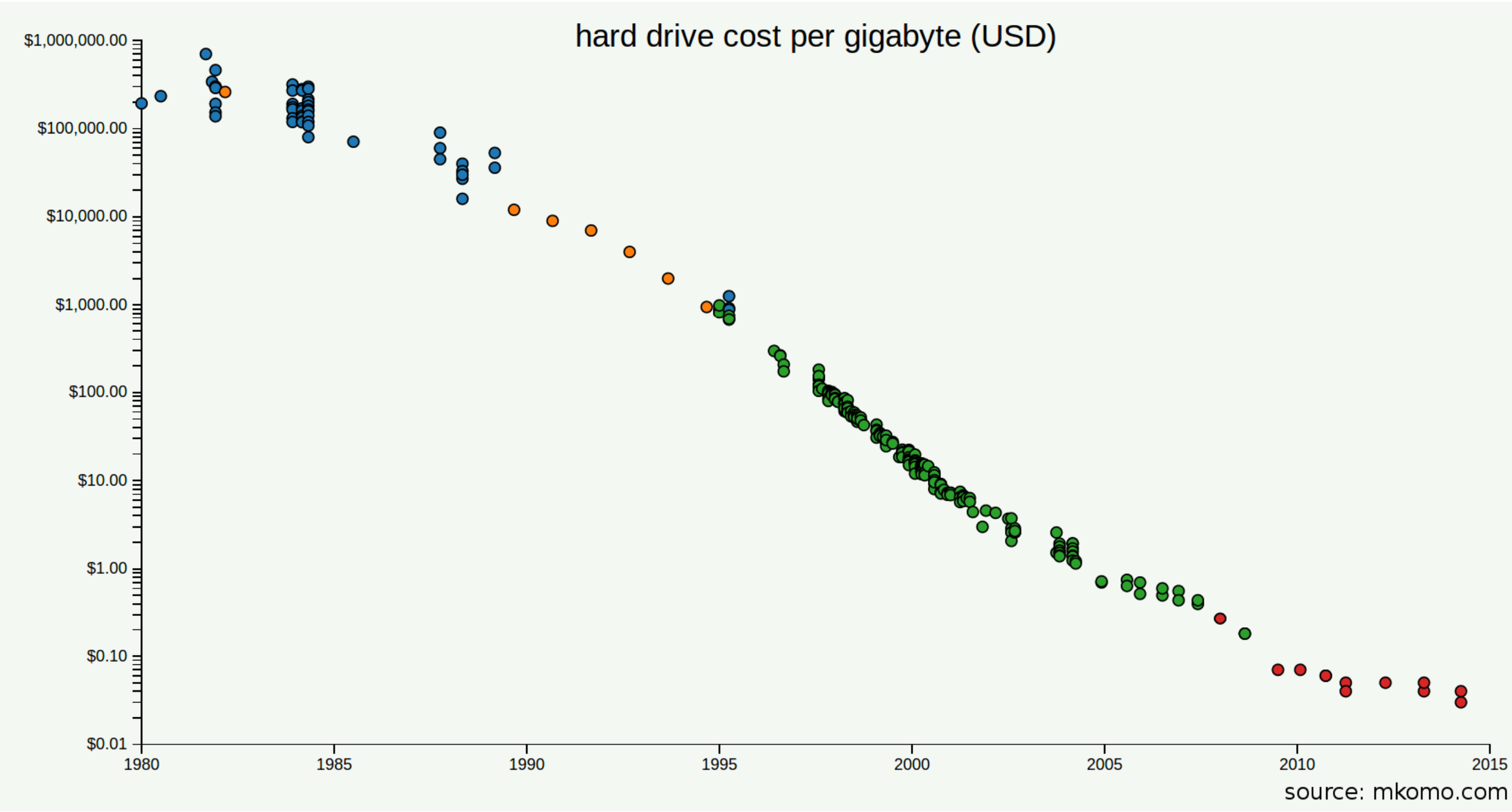
History

- 1956 IBM hard disk
- Capacity: 5 MB
- Size: ~ 3 m^3
- Weight: ~ 1 t
- Cost: ~ 1.000.000 EUR

- 2022 WD hard disk
- Capacity: 8.000.000 MB
- Size: ~ 0.00045 m^3
- Weight: ~ 0.001 t
- Cost: 100 EUR

Full year of 720p video!

Log Scale!



[Matt Komorowski]

Storage became free*

- Google one of first movers
- **Plan**: Index entire web, build search engine
- **Step 1**: Collect Websites
  - Terabytes back then, now petabytes
- **Step 2**: Build inverted (search) index
- **Step 3**: ~~Unclear~~ Advertising Profit

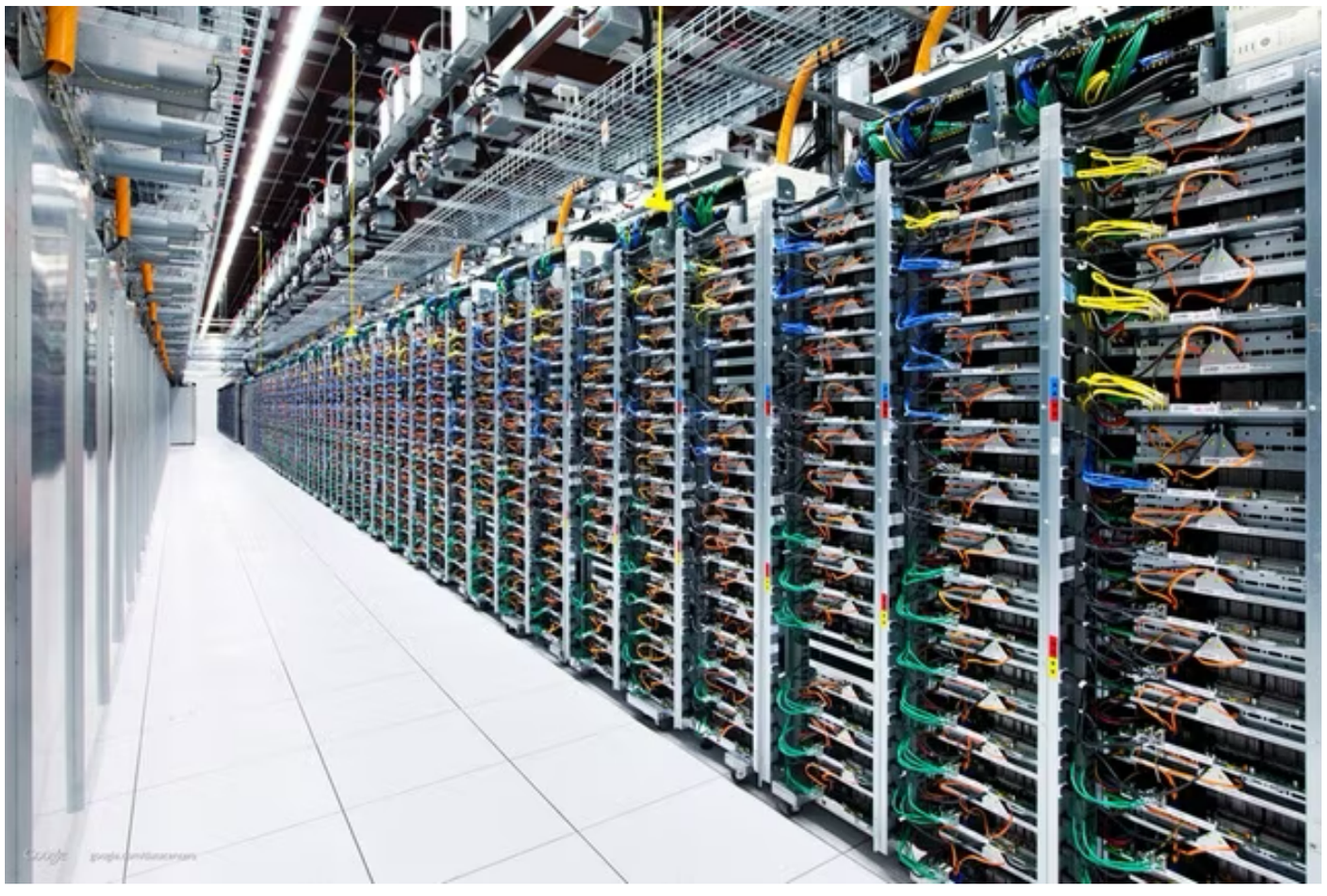
*almost

Distributed Data Processing

- Internet back then was already rather large
  - Terabytes!
- Search index has to be kept up to date…
  - Processing has to happen fast!
- **Problem**: Too much data for a single computer

Distributed Data Processing

- **Solution**: Use more computers!



- … **Problem**: Hard to use more computers!
  - Coordination, Concurrency, Debugging, Hardware Failures

Distributed Data Processing

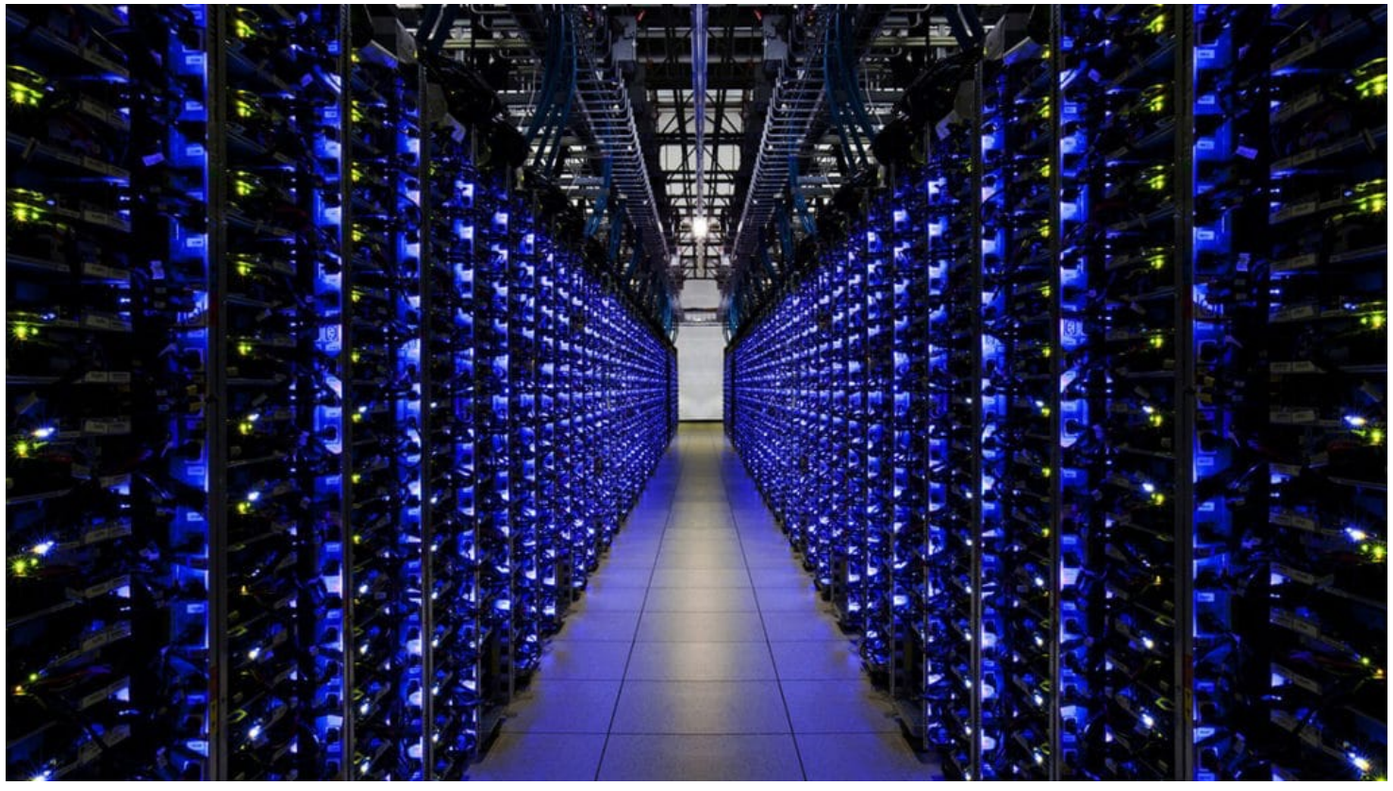
- **Rare failures become common**
  - 1 computer: 1% chance of hard disk failure every year
  - 10 computers: 10% chance of hard disk failure every year
  - 100 computers: 64% chance of hard disk failure every year
  - …

The Joys of Real Hardware

Typical first year for a new cluster:

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)
~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)
~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)
~1 network rewiring (rolling ~5% of machines down over 2-day span)
~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
~5 racks go wonky (40-80 machines see 50% packetloss)
~8 network maintenances (4 might cause ~30-minute random connectivity losses)
~12 router reloads (takes out DNS and external vips for a couple minutes)
~3 router failures (have to immediately pull traffic for an hour)
~dozens of minor 30-second blips for dns
~1000 individual machine failures
~thousands of hard drive failures
slow disks, bad memory, misconfigured machines, flaky machines, etc.

[Jeff Dean]



?

TECHNOLOGY

Google bolsters underwater cables to combat shark bites

https://www.submarinecablemap.com

[Jeff Dean]

Distributed Data Processing

- Difficult to write software that works correctly in distributed env
  - Bugs happen in rare circumstances
  - Hard to track down errors/debug
- … but consequences can be catastrophic
  - Data loss
  - Service outage
- **Very error prone**!

Distributed Data Processing

- Idea: Write a **framework** that handles the hard parts
  - “Which code runs on which machine”
  - “What happens if a computer crashes”
  - “What happens if data gets corrupted"
- **MapReduce**

OSDI 2004

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the *map* and *reduce* primitives present in Lisp and many other functional languages. We realized that most of our computations involved applying a *map* operation to each logical “record” in our input in order to compute a set of intermediate key/value pairs, and then

Distributed Data Processing

- **Example**: Count the number of times words occur
  - Useful for search engines
- Define **map** and **reduce** functions

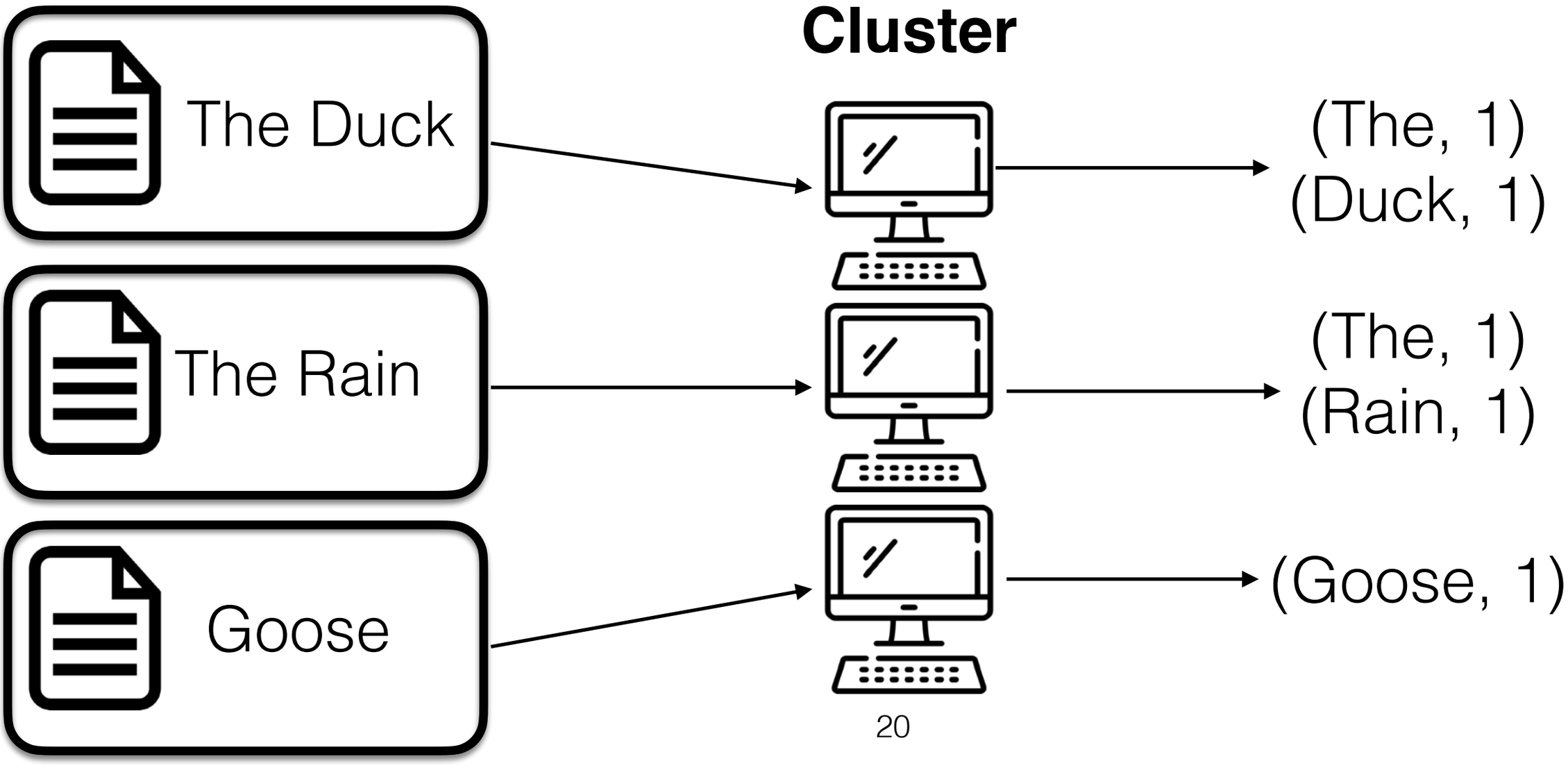
Map
Split document into word pairs
(word, 1)

Reduce
Merge pairs by adding counts
(word, X) + (word, Y) = (word, X + Y)

Distributed Data Processing

Map
Split document into word pairs
(word, 1)

Reduce
Merge pairs by adding counts
(word, X) + (word, Y) = (word, X + Y)

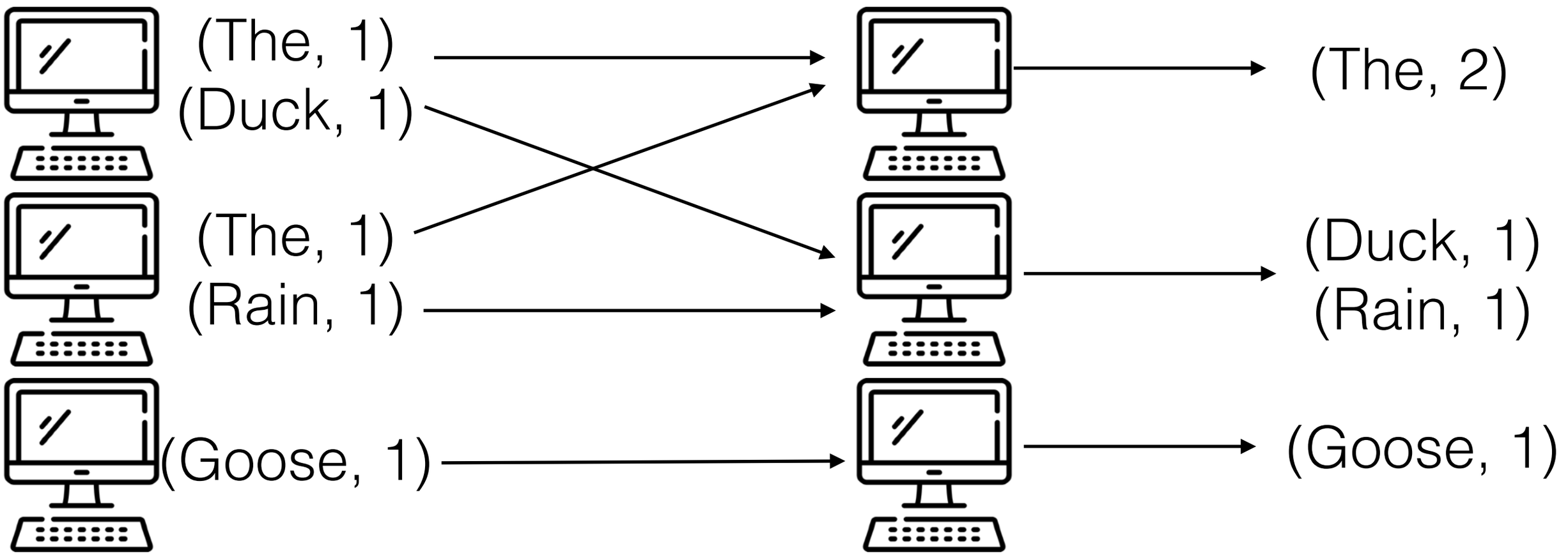


Distributed Data Processing

Map
Split document into word pairs
(word, 1)

Reduce
Merge pairs by adding counts
(word, X) + (word, Y) = (word, X + Y)

Shuffle



Distributed Data Processing

- MapReduce simplifies writing distributed programs
  - But does not solve all issues!
- Functionality is limited
  - Often need multiple MapReduce jobs for basic tasks
- **Slow**: Communication overhead (shuffle) dominates

2006

Hadoop

- MapReduce was never released outside of Google
- **Hadoop**: Open source version of MapReduce
- Beginning of **Big Data Hype**


The Economist
MAY 6TH–12TH 2017
Theresa May v Brussels
Ten years on: banking after the crisis
South Korea's unfinished revolution
Biology, but without the cells
The world's most valuable resource
amazon
Google
Data and the new rules of competition

Infrastructure

Hadoop On-Premise
cloudera, Hortonworks, MAPR, Pivotal, IBM InfoSphere, bluedata, jethro

Hadoop in the Cloud
amazon web services, Microsoft Azure, Google Cloud Platform, IBM InfoSphere, CAZENA, TREASURE DATA, altiscale, Qubole

Spark
databricks, GridGain, TACHYON NEXUS

Cluster Services
amazon web services, kubernetes, HPCC SYSTEMS, docker, MESOSPHERE, CoreOS, pepperdata, StackIQ

NoSQL Databases
amazon DynamoDB, Google Cloud Platform, ORACLE, Microsoft Azure, MarkLogic, mongoDB, DATASTAX, EROSPIKE, Couchbase, SequoiaDB, redislabs, influxdata

NewSQL Databases
SAP HANA, Clustrix, Pivotal, paradigm4, NUODB, memsql, splice machine, MariaDB, VOLTDB, citusdata, deepdb, Trafodion, Cockroach LABS

Graph Databases
neo4j, GIRAPH, OrientDB, InfiniteGraph

MPP Databases
TERADATA, VERTICA, An HP Company, NETEZZA, an IBM Company, Qlction, kognitio, EXASOL, dremio

Cloud EDW
amazon web services, Google Cloud Platform, Microsoft Azure, Pivotal, snowflake, WATERLINE DATA, Infoworks

Data Transformation
alteryx, talend, TRIFACTA, tamr, Paxata, StreamSets, Alation

Data Integration
informatica, Put potential to work:, MuleSoft, snapLogic, BedrockData, xplenty

Management / Monitoring
New Relic, APPDYNAMICS, amazon web services, actifio, Numerify, splunk>, DATADOG, Procana, DRIVEN, Anodot

Security
TANIUM, illumio, CODE42, DataGravity, CipherCloud, VECTRA, sqrrl, BlueTalon

Storage
amazon web services, Google Cloud Platform, Microsoft Azure, panasas, nimblestorage, COHO DATA, Qumulo

App Dev
apigee, CASK, Keen IO, Typesafe, DRIVEN

Crowd-sourcing
amazon mechanical turk, CrowdFlower, WorkFusion

Analytics

Analyst Platforms
Palantir, AYASDI, Quid, enigma, Digital Reasoning, ORBITAL INSIGHT

Analytics Platforms
Microsoft, guavus, Datameer, Bottlenose, interana

Data Science Platforms
context relevant, CONTINUUM ANALYTICS, DataRobot, Alpine DATA LABS, ARIMO, MODE, plotly, dataiku, nutonian, DOMINO, sense, yhat, ALGORITHMIA

Visualization
tableau, Google Cloud Platform, Qlik, looker, Roambi, SISENSE, ZOOMDATA, datorama, CHARTIO

BI Platforms
Power BI, amazon web services, DOMO, salesforce, Wave Analytics, birst, GoodData, kyvos insights, platfora, atscale, ARCADIA, SISENSE

Statistical Computing
sas, SPSS, MATLAB

Log Analytics
splunk>, sumologic, kibana, CLOUD PHYSICS, loggly

Social Analytics
Hootsuite, NETBASE, DATASIFT, tracx, bitly, synthesio, simplereach

Real-Time
amazon web services, METAMARKETS, striim, confluent, DATATORRENT, dataArtisans

Machine Learning
Azure Machine Learning, H2O.ai, amazon web services, Dato, SKYTREE, rapidminer, DATARPM, deepsense.io, ViSENZE, PredictionIO, glowfi.sh

Speech & NLP
NarrativeScience, NUANCE, WolframAlpha, semanticmachines, Gridspace, ARRIA, api.ai, cortical.io, maluuba, MindMeld, iDIBON, yseop

Horizontal AI
IBM Watson, Cortana., sentient technologies, VIV, nervana SYSTEMS, vicarious, nara, Numenta, HyperScience, SI SCALED INFERENCE, Descartes Labs, clarifai, DEXTRO, Geometric Intelligence, MetaMind

Search
Autonomy, ORACLE, ENDECA, EXALEAD, Lucidworks, elastic, ThoughtSpot, MAANA, swiftype, Algolia, SINEQUA

Data Services
OPERA SOLUTIONS, Mu Sigma, EXL analytics, DATASCIENCE, BILLION VALUE DATA SCIENCE, kaggle, datascope, DataKind

For Business Analysts
OrigamiLogic, ClearStory, CIRRO, import.io

Web / Mobile / Commerce
Google Analytics, mixpanel, RJMetrics, BLUECORE, AMPLITUDE, granify, sumall, Airtable, retention SCIENCE, custora

Applications

Sales & Marketing
RADIUS, Gainsight, bloomreach, Zeta, EVERSTRING, livefyre, blueyonder, Lattice, kahuna, infer, SAILTHRU, persado, AVISO, 6sense, QUANTIFIND, ACTIONIQ, fuse machines, ENGAGIO

Customer Service
MEDALLIA, ATTENSITY, CLARABRIDGE, ClickFox, STELLAService, NGDATA, Preact, DigitalGenius, appuri, Wise.io

Human Capital
gild, Connectifier, textio, entelo, hiQ

Legal
RAVEL, JUDICATA, Everlaw, Brevia, PREMONITION

Ad Optimization
AppNexus, MediaMath, criteo, OpenX, rocketfuel, Integral Ad Science, theTradeDesk, Adgorithms, dstillery, LiveIntent, TAPAD, DataXu, Appier, MOAT

Security
CYLANCE, CounterTack, cybereason, AREA 1 SECURITY, ThreatMetrix, SentinelOne, Recorded Future, Guardian Analytics, FORTSCALE, siftscience, Keybase, feedzai, SIGNIFYD

Vertical AI Applications
facebook, Clara, KASISTO, lumiata

Publisher Tools
outbrain, Taboola, quantcast, Chartbeat, yieldbot, Yieldmo

Govt / Regulation
Socrata, OPENGOV, FiscalNote, PREDPOL, enigma, mark43, OpenDataSoft

Finance
Affirm, LendingClub, OnDeck, Kreditech, zestfinance, LendUp, Kabbage, tidemark, Payoff, INSIKT, zuora, Dataminr, Lenddo, KENSHO, AIDYIA, iSENTIUM, Quantopian, sentient technologies

Education / Learning
KNEWTON, Clever, declara, PANORAMA EDUCATION, knowre

Life Sciences
23andMe, Counsyl, PATHWAY GENOMICS, Recombine, deep genomics, KYRUUS, FLATIRON, zymergen, HealthTap, METABIOTA, ZEPHYR HEALTH, ovia, Ginger.io, transcriptic, Glow, enlitic, AiCure, Atomwise

Industries
OPOWER, eHarmony, RetailNext, duetto, STITCH FIX, WorkFusion, BLUE RIVER, TACHYUS, SwiftKey, Seeq, FarmLogs, HowGood, celect, SIGHT MACHINE, statmuse, BOXEVER

Cross-Infrastructure/Analytics

amazon web services, Google, Microsoft, IBM, SAP, sas, 1010data, hp, Autonomy, VERTICA, An HP Company, vmware, TIBCO, TERADATA, ORACLE, NetApp

Open Source

Framework
hadoop HDFS, hadoop MapReduce, YARN, Spark, MESOS, TEZ, Apache Kylin, Flink, CDAP

Query / Data Flow
SLAMDATA, HIVE, APACHE DRILL, Google Cloud Dataflow

Data Access
cassandra, APACHE HBASE, accumulo, mongoDB, CouchDB, riak, OPENTSDB, kafka, nifi

Coordination
talend, ZOOKEEPER, Apache Zookeeper, Apache Ambari

Real-Time
STORM, Spark, APEX, Flink, TACHYON, druid

Stat Tools
ScalaLab, NumPy, SciPy

Machine Learning
mllib, mahout, Apache SINGA, Aerosolve, MADlib, Caffe, torch, CNTK, TensorFlow, learn, FeatureFu, VELES, WEKA, DIMSUM, jupyter, DL4J

Search
elasticsearch, Solr, Lucene

Security
Apache Ranger

Visualization
Zeppelin

Data Sources & APIs

Health
JAWBONE, GARMIN, practicefusion, fitbit, Withings, VALIDIC, netatmo, kinsa, Human API

IOT
UPTAKE, ThingWorx, helium, samsara, AUGURY, estimote

Financial & Economic Data
Bloomberg, DOW JONES, THOMSON REUTERS, YODLEE, PREMISE, S&P CAPITAL IQ, quandl, xignite, CB INSIGHTS, mattermark, StockTwits, estimize, PLAID

Air / Space / Sea
PLANET LABS, spire, WINDWARD, CRUISE, SKYCATCH, Airware, DroneDeploy

Location / People / Entities
acxiom, Experian, EPSILON, InsideView, GARMIN, foursquare, STREETLINE, esri, Crimson Hexagon, CARTODB, factual, PlaceIQ, CIRCULATE, placemeter, BASIS, Sense 360

Other
qualtrics, panjiva, DATA.GOV

Incubators & Schools

GA, PLURALSIGHT, DataCamp, INSIGHT, DataElite, The Data Incubator, METIS

Last Updated 3/23/2016

FIRSTMARK

Pokemon or Big Data?

https://pixelastic.github.io/pokemonorbigdata/

Since 2004

- Basically still MapReduce
- Faster, nicer to use, more abstraction, more efficient, ..
- But fundamentally still the same
- Limitations also still the same

- **Apache Spark** current state of things

Apache Spark

- Distributed processing framework
  - “Nicer Map Reduce”

- Spark requires a **cluster of computers**
  - **In practicum**: local cluster runs on your own machine
  - **In practice**: rent from cloud provider (AWS/Azure/…) or run on own servers
- Your computer gives commands to the cluster

Apache Spark

spark_example.py

```
from pyspark import SparkConf, SparkContext
config = SparkConf().setMaster("local").setAppName("test").set('spark.ui.enabled','false')

sc = SparkContext.getOrCreate(config)
```

Connect to the cluster

For practicum: local cluster

Apache Spark

spark_example.py

```
from pyspark import SparkConf, SparkContext
config = SparkConf().setMaster("local").setAppName("test").set('spark.ui.enabled','false')

sc = SparkContext.getOrCreate(config)
```

```
shakespeare = sc.textFile("shakespeare.txt")
```

Load a text file into Spark (stored on local disk)

Apache Spark

spark_example.py

```
from pyspark import SparkConf, SparkContext
config = SparkConf().setMaster("local").setAppName("test").set('spark.ui.enabled','false')

sc = SparkContext.getOrCreate(config)

shakespeare = sc.textFile("shakespeare.txt")
```

```
words = shakespeare.flatMap(lambda line: line.split(' '))
tuples = words.map(lambda word: (word, 1))
counts = tuples.reduceByKey(lambda count_left, count_right: count_left + count_right)
```

Perform map/reduce using Spark

1. Split the text file into words by splitting on space (‘ ‘)
2. **Map:** Construct the (word, 1) pairs
3. **Reduce:** Add the counts together

Apache Spark

spark_example.py

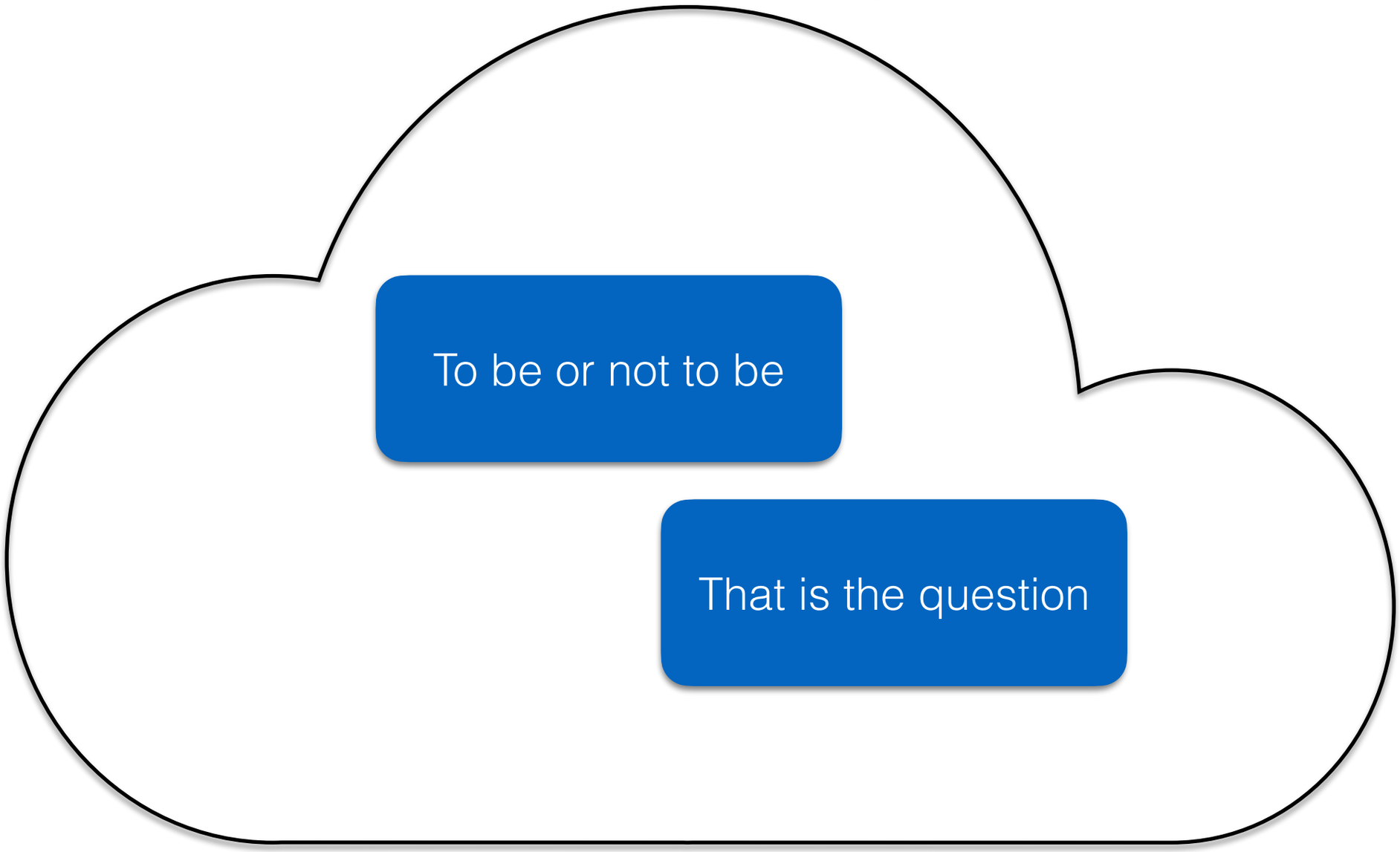
```python
from pyspark import SparkConf, SparkContext
config = SparkConf().setMaster("local").setAppName("test").set('spark.ui.enabled','false')

sc = SparkContext.getOrCreate(config)

shakespeare = sc.textFile("shakespeare.txt")

words = shakespeare.flatMap(lambda line: line.split(' '))
tuples = words.map(lambda word: (word, 1))
counts = tuples.reduceByKey(lambda count_left, count_right: count_left + count_right)

results = counts.takeOrdered(10, key=lambda x: -x[1])
for entry in results:
    print(entry)
```

Fetch the 10 most frequently occurring words and print them

Apache Spark


```
shakespeare = sc.textFile("shakespeare.txt")
```

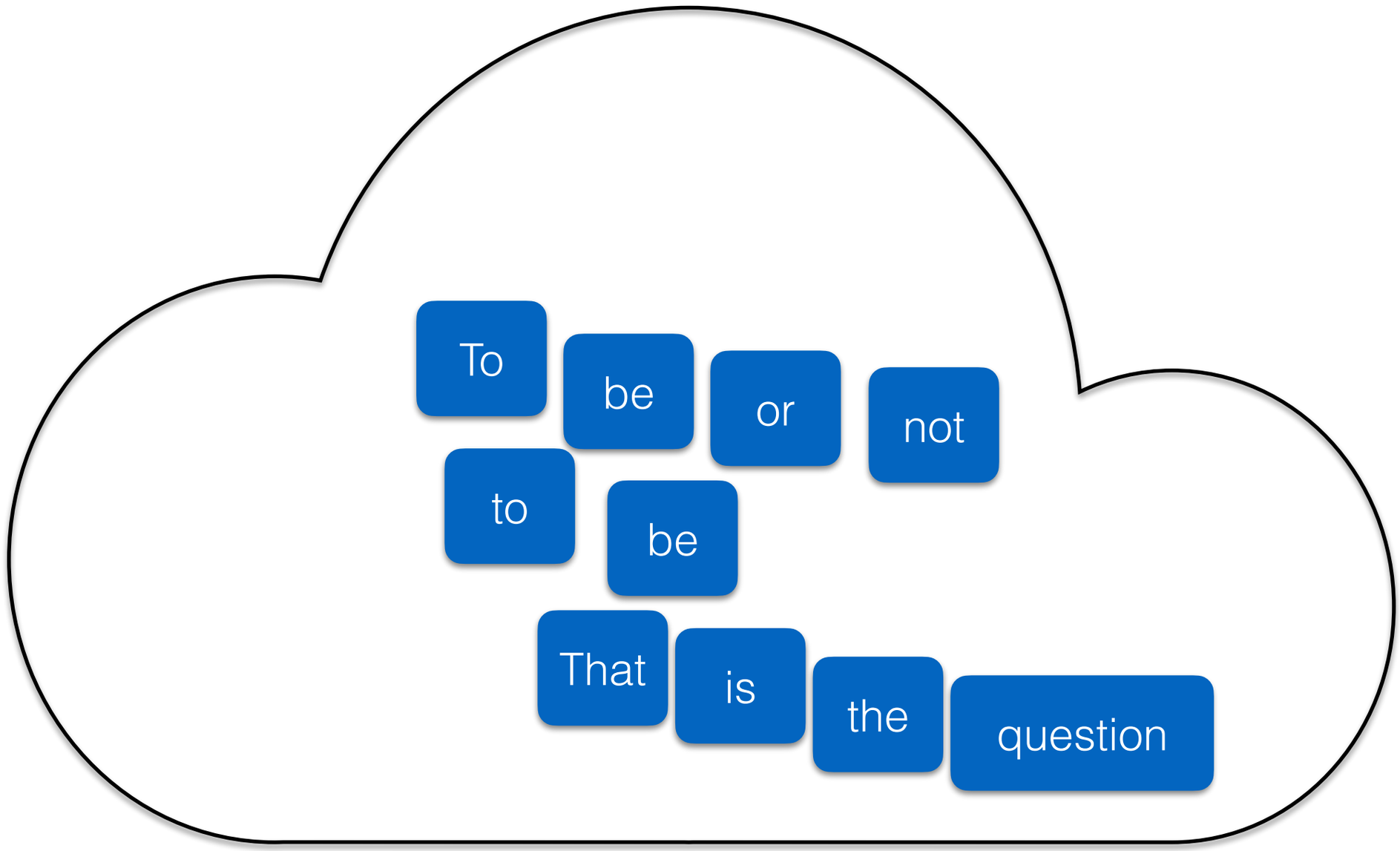
textFile: read lines from file as objects



Apache Spark

```
words = shakespeare.flatMap(lambda line: line.split(' '))
```

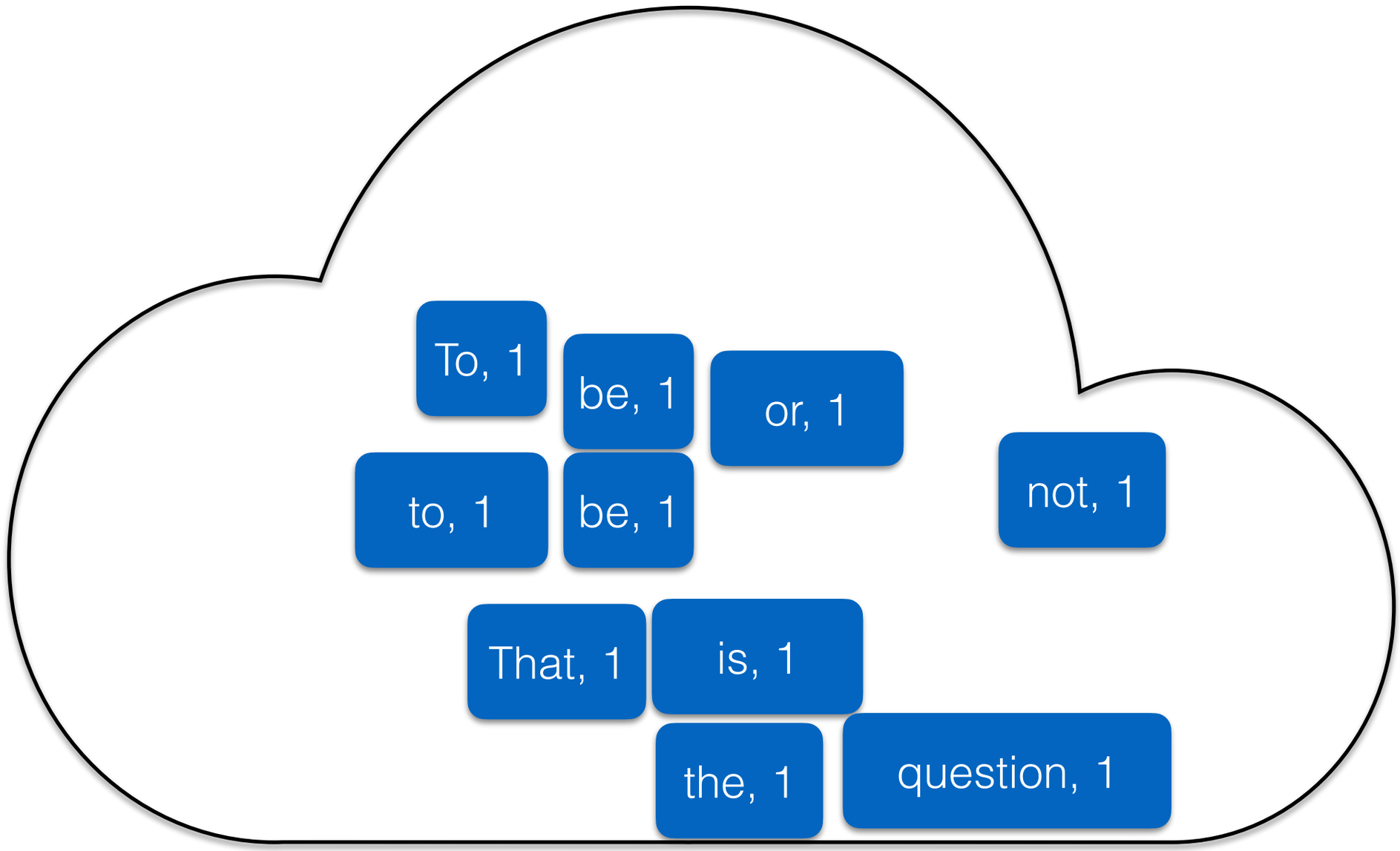
flatMap: Apply function to all objects and flatten



Apache Spark

```
tuples = words.map(lambda word: (word, 1))
```

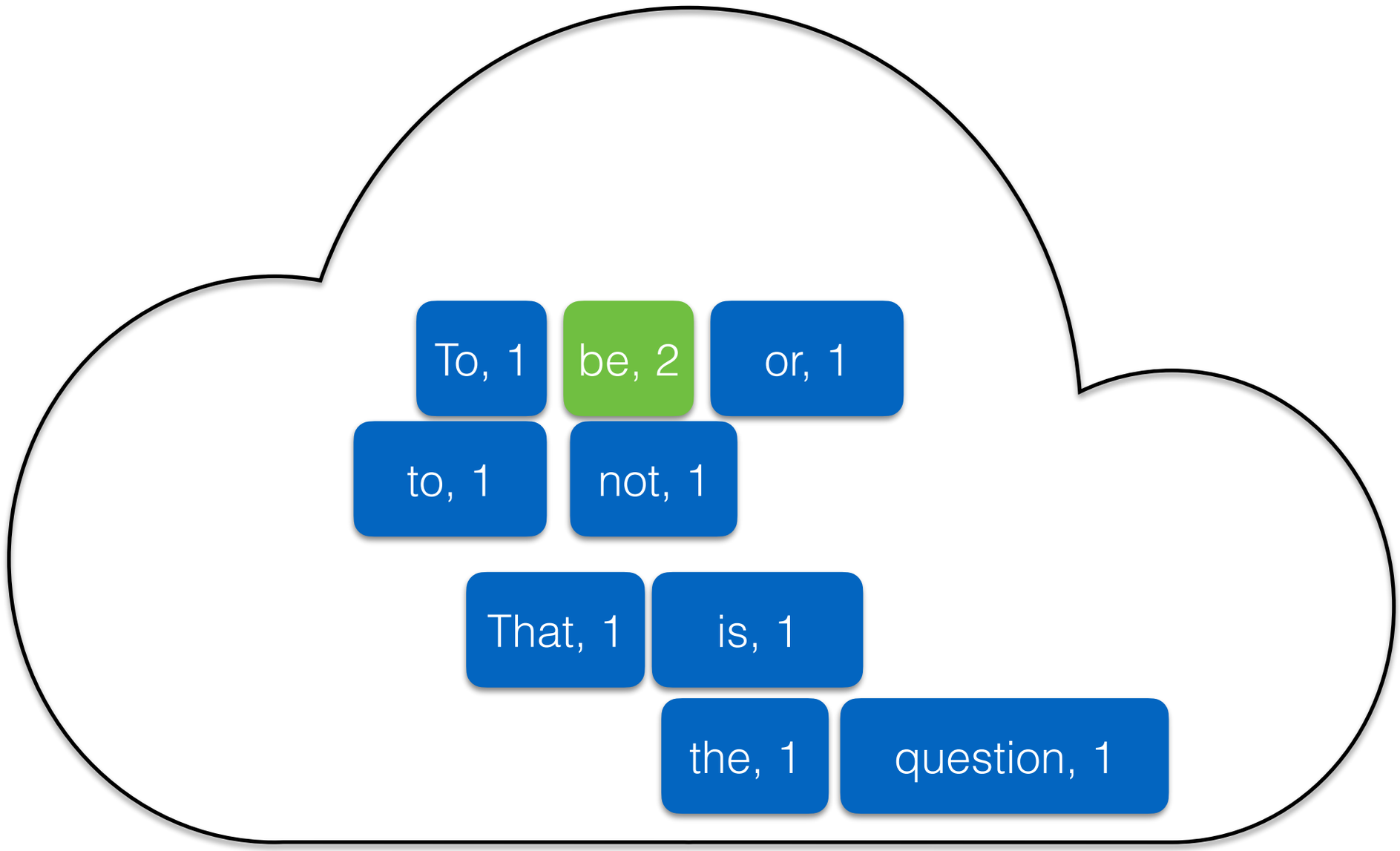
map: Apply function to all objects



Apache Spark


```
counts = tuples.reduceByKey(lambda count_left, count_right: count_left + count_right)
```

reduceByKey: Group by key and combine using function



Embarrassingly parallel?

- Distributed processing works best when problem can be partitioned independently with minor communication
- Web Index? ETL? Filtering? Perfect.
- Graph Analysis? Terrible.

https://en.wikipedia.org/wiki/Embarrassingly_parallel

Scalability! But at what COST?

Frank McSherry
Unaffiliated

Michael Isard
Unaffiliated*

Derek G. Murray
Unaffiliated†

Abstract

We offer a new metric for big data platforms, COST, or the Configuration that Outperforms a Single Thread. The COST of a given platform for a given problem is the hardware configuration required before the platform outperforms a competent single-threaded implementation. COST weighs a system's scalability against the overheads introduced by the system, and indicates the actual performance gains of the system, without rewarding systems that bring substantial but parallelizable overheads.

We survey measurements of data-parallel systems recently reported in SOSP and OSDI, and find that many systems have either a surprisingly large COST, often hundreds of cores, or simply underperform one thread for all of their reported configurations.

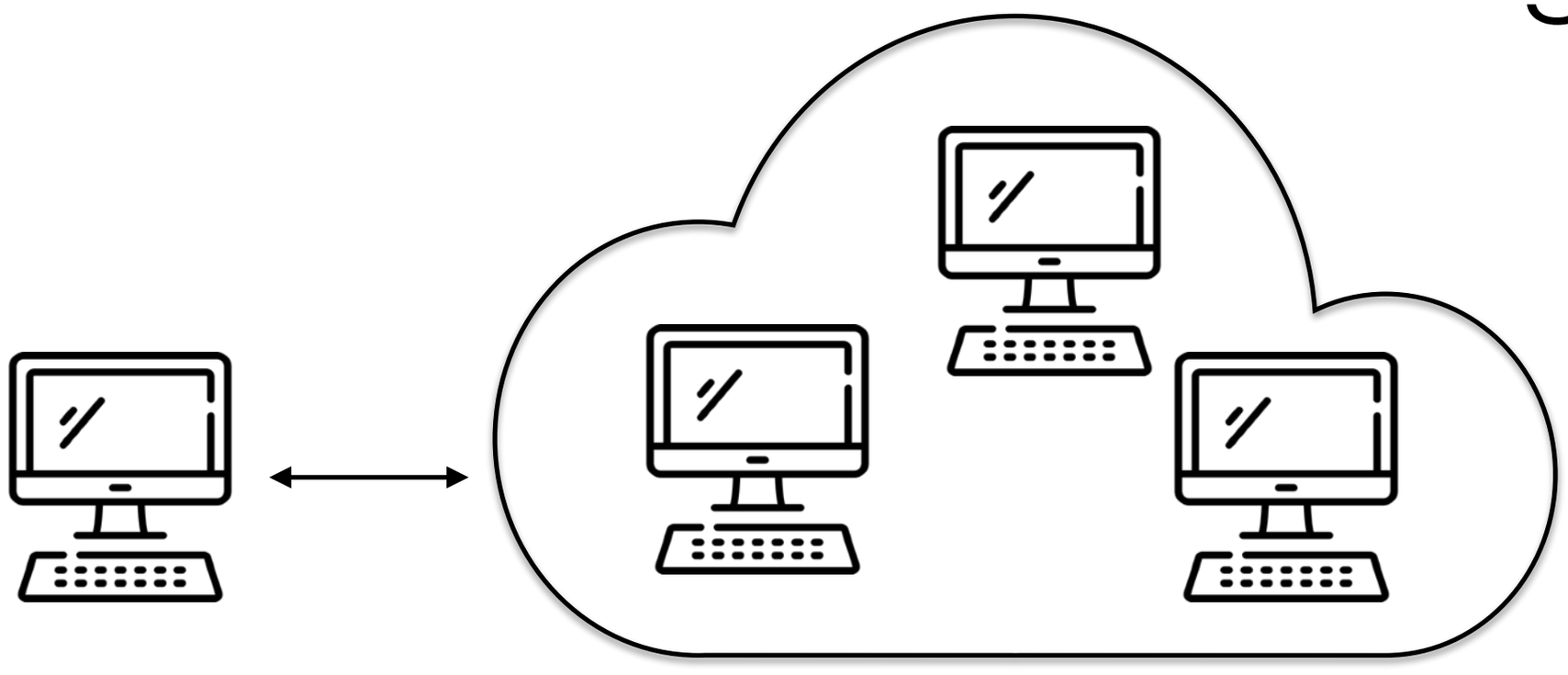
scalable system	cores	twitter	uk-2007-05
GraphLab	128	242s	714s
GraphX	128	251s	800s
Single thread (SSD)	1	153s	417s
Union-Find (SSD)	1	15s	30s

scalable system	cores	twitter	uk-2007-05
GraphLab	128	249s	833s
GraphX	128	419s	462s
Vertex order (SSD)	1	300s	651s
Vertex order (RAM)	1	275s	-
Hilbert order (SSD)	1	242s	256s
Hilbert order (RAM)	1	110s	-

1 Introduction

> "You can have a second computer once you've shown you know how to use the first one."
>
> –Paul Barham

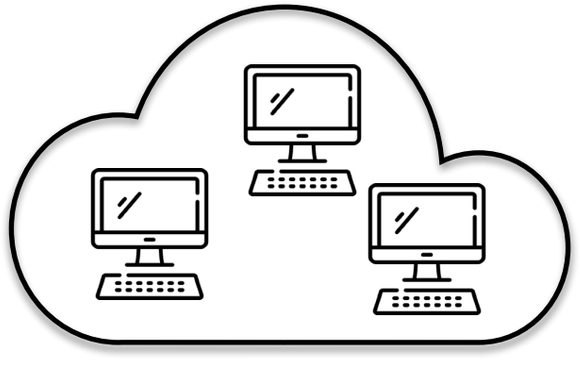
Distributed Processing



- Even if problem is parallel there are overheads to distributed computing
  - Send task to cluster
  - Run task
  - Send back results
- Never **extremely fast**, even for small data sets

Distributed Processing

€



€ € € €

- Distributed processing is **expensive**
  - The more computers the more expensive!
- Harder to debug/iterate than local development

Single Machine

- If we can use a single machine to process data in acceptable time, we should always!
- Single-machine solutions always cheaper and easier
- Machines have gotten **much more powerful** since 2004!

Macbook Pro

64GB Ram
10 cores

AWS u-24tb1.metal

24 TB Ram
224 cores

Right Tool for the Right Job

- Different tools have different use cases/purposes
  - Spark/Distributed processing is one such tool
- Enables processing gigantic data sets
- But not as effective for smaller data sets

Unscientific Rule of Thumb

Single Machine ↔ Distributed

1MB 1GB 1TB 1PB

Practicum 1

Topics Today

1. Introduction
2. **Data Storage & ETL**

    Practicum
3. SQL on Big Data

    Practicum

Data Storage

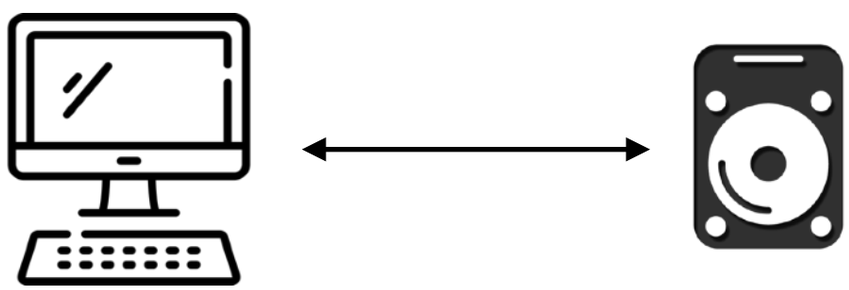
- Traditional way: Database “owns” data.
- Modern way: Data is stored in shared storage in open formats (“Data lake”).
- **Advantage**: No vendor lock-in
- Multiple tools can operate on same data

Data Storage

- Traditional way: Rent/own **servers**
- Modern way: Rent **storage** and **compute** separately
- **Advantage**: Can scale both independently.
- Cheaper

Traditional Server

EC2 **S3**



Modern Data Warehouse

- Store data in **Blob Storage**
  - S3**,** Azure Blob Storage, Google Cloud Storage
- Data is stored in **Open File Formats**
  - Parquet, CSV, Plaintext
- Analyze data using various processing tools
  - **Managed:** Spark cluster
  - **Hosted Service:** Amazon Athena
  - **Locally:** DuckDB or Pandas

Storage System: S3

- “**S**imple **S**torage **S**ervice”, Amazon Web Services
  - "Dropbox for Data”
  - Key-Value store for files
- Pay for storage space and requests
- Azure Blob Storage & Google Cloud Storage variants from Microsoft/Google

Storage System: S3

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (6)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

Find objects by prefix

1

Name	Type	Last modified	Size	Storage class
AthenaQuery/	Folder	-	-	-
lineitem_sample.csv	csv	September 22, 2020, 09:42:44 (UTC+02:00)	1.3 KB	Standard
lineitem.db	db	May 9, 2023, 18:48:58 (UTC+02:00)	248.3 MB	Standard
lineitem.parquet	parquet	May 9, 2023, 19:17:35 (UTC+02:00)	258.3 MB	Standard
storage_version.db	db	May 9, 2023, 17:47:40 (UTC+02:00)	2.0 MB	Standard
Unsaved/	Folder	-	-	-

- Files can be uploaded through Web GUI
- For managing data usually through programs/tools
  - Most modern data tools can read and write from S3
- e.g. Spark, DuckDB, Pandas

File Formats: Plaintext

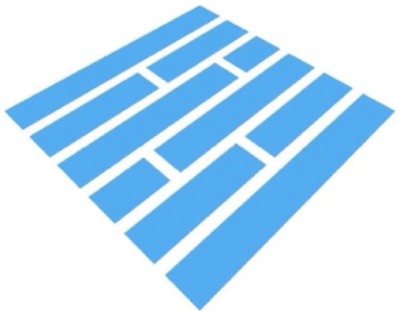
- Just text, e.g. logfiles
- Completely unstructured
- Needs ETL pass to proceed
- Very common
- Typical starting point in practise

File Formats: CSV

- Tables as text
- Structured, but often be broken
- Human readable-ish
- Most tools can process CSV
- But inefficient

```
id,name,released_on,price,created_at,updated_at
24,1000 Piece Jigsaw Puzzle,2012-07-03,14.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
30,360° Protractor,2012-05-03,3.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
17,7 Wonders,2012-04-21,28.75,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
13,Acoustic Guitar,2012-06-06,1025.0,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
15,Agricola,2012-05-22,45.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
22,Answer to Everything,2012-07-03,42.0,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
23,Box Kite,2012-05-19,63.0,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
29,CanCan Music Record,2012-05-09,2.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
12,Chocolate Pie,2012-04-12,3.14,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
9,Dog Toy Bone,2012-06-13,2.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
11,Flux Capacitor,2012-06-01,19.55,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
6,Game Console,2012-06-06,299.95,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
10,Heated Blanket,2012-07-19,27.95,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
19,Knights of Catan,2012-06-10,19.95,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
8,Lawn Chair,2012-05-29,34.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
21,Millennium Falcon,2012-04-10,3597200.0,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
14,Model Enterprise,2012-04-18,27.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
28,Model Train Rails,2012-06-30,45.0,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
3,Oak Coffee Table,2012-07-08,223.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
5,Oh's Cereal,2012-04-17,3.95,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
```

File Formats: Parquet



- Tables, split up by row groups and columns
- Binary format, compressed
- Self-describing
- **Advantages**:
  - Compression reduces file size drastically (5-10X smaller)
  - Can be read very efficiently
- Recommended format

ETL: Basic Principle

	c1	c2	c3
r1	42		
r2			
r3			

ETL = First step in Analysis

- Dataset from somewhere, almost always messy
  - Making data usable/“analysable”
- Often want to combine multiple sources
  - e.g. Stock Price + Weather
  - Need compatible representation

ETL = First step in Analysis

- **ETL = Messy gunk to usable table(s)**
- Extremely Task-specific!
  - ETL can turn Big Data into Small Data (single node!)
- Typically **takes majority of time!**
- Revolves around files

Big Data ETL Process

1. Load (e.g. zipped plaintext)
    - Decompress
    - Iterate over Records
2. Decode
3. Normalize
4. Store (e.g. Parquet)
    - Chunk Records
    - Compress
5. (Analyze)

Parallel!

Typical Issues

- All the classical ETL issues
  - Date formats (non-ISO 8601)
  - Number formats
  - Encoding
  - Units
  - NULLs
  - …

Typical Issues 2

- Container formats (ZIP, tar, etc.)
  - also nested, ZIP of ZIP of TAR etc.
  - Exotic containers, HDF5, AVRO, …
  - Data encoded in filenames
  - Third-party readers often needed

Example: AIS

- “Automatic Identification System”, ships send out radio signals with position etc.
- Mainly for safety
- Recorded, ~ 1 TB/year

AIS Visualization

https://www.marinetraffic.com

Raw Data (Compressed Text)

s3://ais/raw_data/00-00.txt.gz
s3://ais/raw_data/00-01.txt.gz
s3://ais/raw_data/00-02.txt.gz
s3://ais/raw_data/00-03.txt.gz
s3://ais/raw_data/00-04.txt.gz
s3://ais/raw_data/00-05.txt.gz
s3://ais/raw_data/00-06.txt.gz
s3://ais/raw_data/00-07.txt.gz
s3://ais/raw_data/00-08.txt.gz
s3://ais/raw_data/00-09.txt.gz
…

Decompress

```
!AIVDM,1,1,,B,146HbN?P00NqNp>@66`qjgv`0>`<,0*3A
!AIVDM,1,1,,A,B69>@lh07B:p2v5HVGpsswW6kP06,0*6D
!AIVDM,1,1,,A,Dh3OwjR<Tnfp2MF9H35F9H3eF9H,2*33
!AIVDM,1,1,,A,13aEObh0000E=1:N1H3Hi0:PP>`<,0*7B
!AIVDM,1,1,,A,13aBNegP050G9;vM`Q>8AgvP0>`<,0*35
!AIVDM,1,1,,B,13aFh1@P1=PFUCNMvCsbFOvR2>`<,0*1F
!AIVDM,1,1,,A,13KCpp3vhD0Q:9@OPM?uo;6p0>`<,0*3F
!AIVDM,1,1,,B,Dh3OwjQMdnfq<QF9I=9F9I=iF9H,2*61
!AIVDM,1,1,,A,14eGkT0001o8okhL8I9725`p0>`<,0*72
!AIVDM,1,1,,A,13bV2L0P?w<tSF0l4Q@>4?wv0>`<,0*04
!AIVDM,1,1,,B,14eGCm@001rQfhnICBnpU38t2>`<,0*32
!AIVDM,1,1,,B,15Mj3<002To?T>DK<>eS12Lv0>`<,0*05
…
```

AIVDM protocol messages

Decode

```
!AIVDM,1,1,,B,146HbN?P00NqNp>@66`qjgv`0>`<,0*3A
```

libais_example.py

```
import ais
print(ais.decode('146HbN?P00NqNp>@66`qjgv`0>`<'))

{'id': 1, 'repeat_indicator': 0, 'mmsi': 275131000, 'nav_status': 15, 'rot_over_range':
True, 'rot': -731.386474609375, 'sog': 0.0, 'position_accuracy': 0, 'x':
-15.404788333333334, 'y': 28.128698333333332, 'cog': 250.60000610351562, 'true_heading'
511, 'timestamp': 20, 'special_manoeuvre': 0, 'spare': 0, 'raim': False, 'sync_state':
0, 'slot_timeout': 3, 'received_stations': 10764}
```

Decode

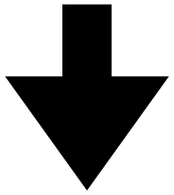
```
!AIVDM,1,1,,B,146HbN?P00NqNp>@66`qjgv`0>`<,0*3A
```

```
MMSI      = 275131000
Longitude = -15.404
Latitude  = +28.128
…
```

Normalize/Store

Convert into **structured table**

```
Msg,RepInd,MMSI,NavStat,ROT,SOG,PosAcc,Long,E/W,Lat,N/S,COG,Heading,TimeStamp,Special,Spare,Raim
1,0,636016199,0,+0.0,14.4,1,3.5954833,E,51.9660333,N,73.0,75,29,0,0,0
1,0,227729770,0,+0.0,0.0,1,0.1198500,E,49.4838833,N,277.0,282,31,0,0,0
1,0,261020840,15,-128.0,0.0,1,18.4185500,E,54.7969500,N,302.0,511,30,0,0,1
1,0,431004315,0,+0.0,0.0,1,135.4497267,E,34.6464667,N,203.8,71,30,0,0,0
1,0,255805588,5,+0.0,19.5,1,135.1636500,E,34.4905000,N,224.7,225,26,0,0,0
…
```



Store as **parquet files**

```
s3://ais/2023/08/01/signals.parquet
s3://ais/2023/08/02/signals.parquet
s3://ais/2023/08/03/signals.parquet
…
```

Ready for analysis!

Practicum 2

Topics Today

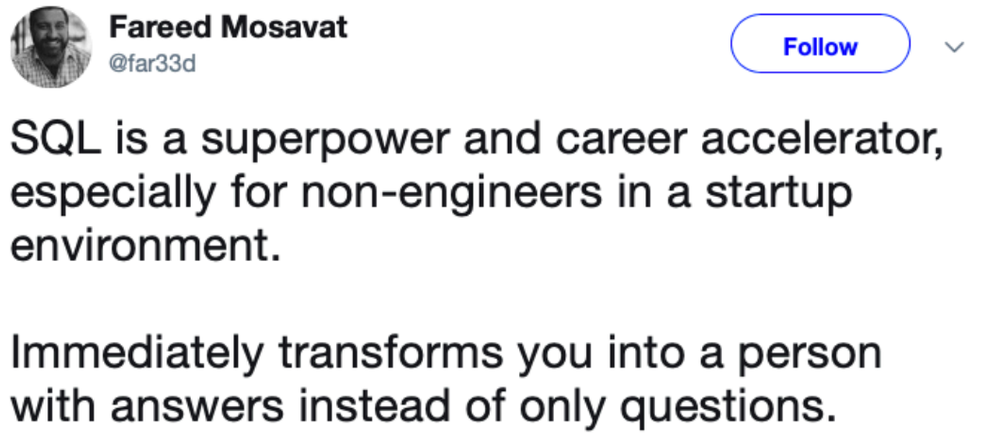
1. Introduction
2. Data Storage & ETL

   Practicum
3. **SQL on Big Data**

   Practicum

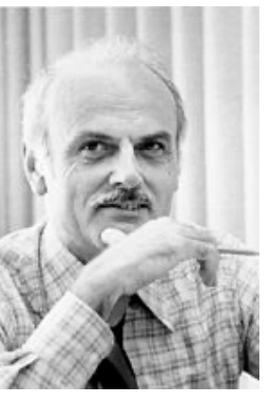
Why SQL

- Decoupling of query and execution, "Declarative"
- "I want these rows, figure out how to get them for me"
- Allows for optimisation
- Allows for changes to physical storage without changing queries
- Huge renaissance of SQL in ~ last 5 years
- SQL is being retrofitted to systems that swore it off initially



1970

Brief History of SQL



Information Retrieval P. BAXENDALE, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. Data Dependencies in Present Systems

SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

1974

by

Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California

ABSTRACT: In this paper we present the data manipulation facility for a structured English query language (SEQUEL) which can be used for accessing data in an integrated relational data base. Without resorting to the concepts of bound variables and quantifiers SEQUEL identifies a set of simple operations on tabular structures, which can be shown to be of equivalent power to the first order predicate calculus. A SEQUEL user is presented with a consistent set of keyword English templates which reflect how people use tables to obtain information. Moreover, the SEQUEL user is able to compose these basic templates in a structured manner in order to form more complex queries. SEQUEL is intended as a data base sublanguage for both the professional programmer and the more infrequent data base user.

Brief History of SQL

- SEQUEL was a pun on the then-popular QUEL
  - QUEL stands for QUEry Language
- Trademark dispute - name was later changed to S.Q.L.
  - Structured Query Language
- Source of divergent pronunciation (S.Q.L. vs Sequel)
- Official pronunciation is S.Q.L.

“Standard” SQL

- SQL was first proposed in 1974
- Implemented in several systems (System R, DB2, Oracle)

- Standardized by ANSI/ISO only in 1986
  - Already many differences between dialects…

“Standard” SQL

- There is a standard…
  - But created after multiple implementations

- Not everything is well defined by the standard
  - Different systems often have different behavior

Postgres

```
   name
---------
 Hannes
 Mark

(3 rows)
```

```
SELECT *
FROM names
ORDER BY name;
```

SQLite

```
name
------
NULL
Hannes
Mark
```

Sample Dataset

Students

SID	FIRST	LAST
101	Ann	Smith
102	Michael	Jones
103	Richard	Turner
104	Maria	Brown

Results

SID	CAT	ENO	POINTS
101	H	1	10
101	H	2	8
101	M	1	12
102	H	1	9
102	H	2	9
102	M	1	9
103	H	1	5
103	M	1	7

Basic Query Syntax

Projection

Column Reference with Alias

```
SELECT   SID, S.FIRST AS F
FROM     STUDENTS AS S
WHERE    S.LAST = 'Smith'
```

Table with Alias

Selection

Condition

String Literal

```
SELECT   SID, S.FIRST
FROM     STUDENTS AS S
WHERE    S.LAST = 'Smith'
```

S

SID	FIRST	LAST
101	Ann	Smith
102	Michael	Jones
103	Richard	Turner
104	Maria	Brown

```
WHERE    S.LAST = 'Smith'
```

Row Selection

SID	FIRST	LAST
101	Ann	Smith
102	Michael	Jones
103	Richard	Turner
104	Maria	Brown

```
SELECT   SID, S.FIRST
```

Column Projection

Expressions

Arithmetic in Projection

String Functions

Integer Literal

```
SELECT  SID + 42, LENGTH(FIRST)
FROM    STUDENTS
WHERE   LAST = 'Smith' AND (SID + 1) > 102
```

Logical Conjunction

Functions? Documentation!

Bells & Whistles

No Duplicates

```
SELECT DISTINCT FIRST
FROM            STUDENTS
WHERE           LAST = 'Smith'
ORDER BY        FIRST DESC
LIMIT 10
```

Result Sorting

Top 10

Aggregates

```
SELECT COUNT(*)
FROM   RESULTS
```

Aggregation Function

COUNT
MIN
MAX
AVG
SUM
STDEV
VAR
...

```
SELECT MIN(POINTS), MAX(POINTS)
FROM   RESULTS
WHERE  CAT = 'H'
```

Changes Result Cardinality!

```
SELECT MIN(POINTS), MAX(POINTS)
FROM   RESULTS
WHERE  CAT = 'H'
```

Results

SID	CAT	ENO	POINTS
101	H	1	10
101	H	2	8
101	M	1	12
102	H	1	9
102	H	2	9
102	M	1	9
103	H	1	5
103	M	1	7

MIN(POINTS)	MAX(POINTS)
5	10

```
WHERE  CAT = 'H'
```

```
SELECT MIN(POINTS), MAX(POINTS)
```

Grouped Aggregates

```
SELECT ENO, AVG(POINTS)
FROM RESULTS
WHERE CAT = 'H'
GROUP BY ENO
```

Grouping Attribute

As many result rows as groups

```
SELECT ENO, AVG(POINTS)
FROM RESULTS
WHERE CAT = 'H'
GROUP BY ENO
```

Results

SID	CAT	ENO	POINTS
101	H	1	10
101	H	2	8
101	M	1	12
102	H	1	9
102	H	2	9
102	M	1	9
103	H	1	5
103	M	1	7

ENO=1

...	ENO	POINTS
	1	10
	1	9
	1	5

ENO=2

...	ENO	POINTS
	2	8
	2	9

ENO	AVG(POINTS)
1	8
2	8,5

WHERE CAT = 'H'

GROUP BY ENO

SELECT ENO,
AVG(POINTS)

Filtering Grouped Aggregates

```
SELECT ENO, AVG(POINTS)
FROM RESULTS
WHERE CAT = 'H'
GROUP BY ENO
HAVING AVG(POINTS) > 8.2
```


Group filter criteria
no column references!

```
SELECT ENO, AVG(POINTS)
FROM RESULTS
WHERE CAT = 'H'
GROUP BY ENO
HAVING AVG(POINTS) > 8.2
```

ENO=1

...	ENO	POINTS
	1	10
	1	9
	1	5

ENO=2

...	ENO	POINTS
	2	8
	2	9

GROUP BY ENO

ENO	AVG(POINTS)
1	8
2	8,5

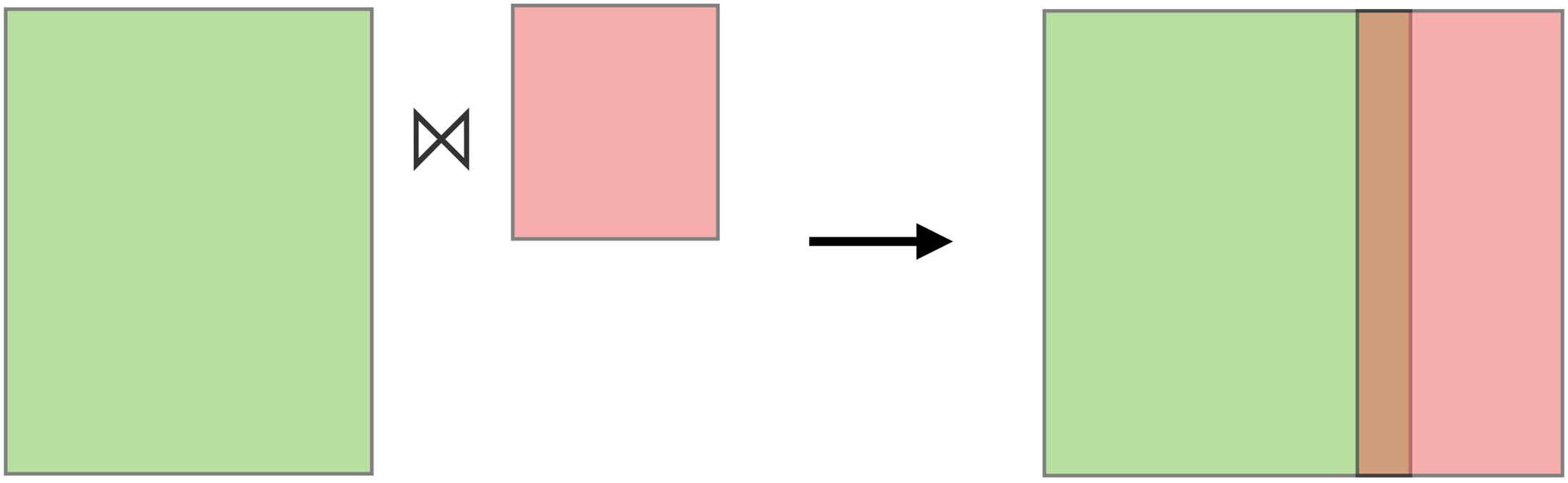
HAVING
AVG(POINTS) > 8.2

ENO	AVG(POINTS)
2	8,5

SELECT ENO,
AVG(POINTS)

Joins

Horizontal Combination of Tables



Rectangular Result

Inner Joins

```
SELECT R.ENO, R.POINTS
FROM   STUDENTS S JOIN RESULTS R
  ON   (S.SID = R.SID)
WHERE  S.LAST = 'Smith'
```

Join Condition

Students

SID	FIRST	LAST
101	Ann	Smith
102	Michael	Jones
103	Richard	Turner
104	Maria	Brown

$\bowtie_{SID}$

Results

SID	CAT	ENO	POINTS
101	H	1	10
101	H	2	8
101	M	1	12
102	H	1	9
102	H	2	9
102	M	1	9
103	H	1	5
103	M	1	7

S.SID	S.FIRST	S.LAST	R.SID	R.CAT	R.ENO	R.POINTS
101	Ann	Smith	101	H	1	10
101	Ann	Smith	101	H	2	8
101	Ann	Smith	101	M	1	12
102	Michael	Jones	102	H	1	9
102	Michael	Jones	102	H	2	9
102	Michael	Jones	102	M	1	9
103	Richard	Turner	103	H	1	5
103	Richard	Turner	103	M	1	7

```
SELECT R.ENO, R.POINTS
FROM   STUDENTS S JOIN RESULTS R
  ON   (S.SID = R.SID)
WHERE  S.LAST = 'Smith'
```

S.SID	S.FIRST	S.LAST	R.SID	R.CAT	R.ENO	R.POINTS
101	Ann	Smith	101	H	1	10
101	Ann	Smith	101	H	2	8
101	Ann	Smith	101	M	1	12
102	Michael	Jones	102	H	1	9
102	Michael	Jones	102	H	2	9
102	Michael	Jones	102	M	1	9
103	Richard	Turner	103	H	1	5
103	Richard	Turner	103	M	1	7

```
SELECT R.ENO, R.POINTS
FROM   STUDENTS S JOIN RESULTS R
  ON   (S.SID = R.SID)
WHERE  S.LAST = 'Smith'
```

Students

SID	FIRST	LAST
101	Ann	Smith
102	Michael	Jones
103	Richard	Turner
104	Maria	Brown

⋈ SID

Results

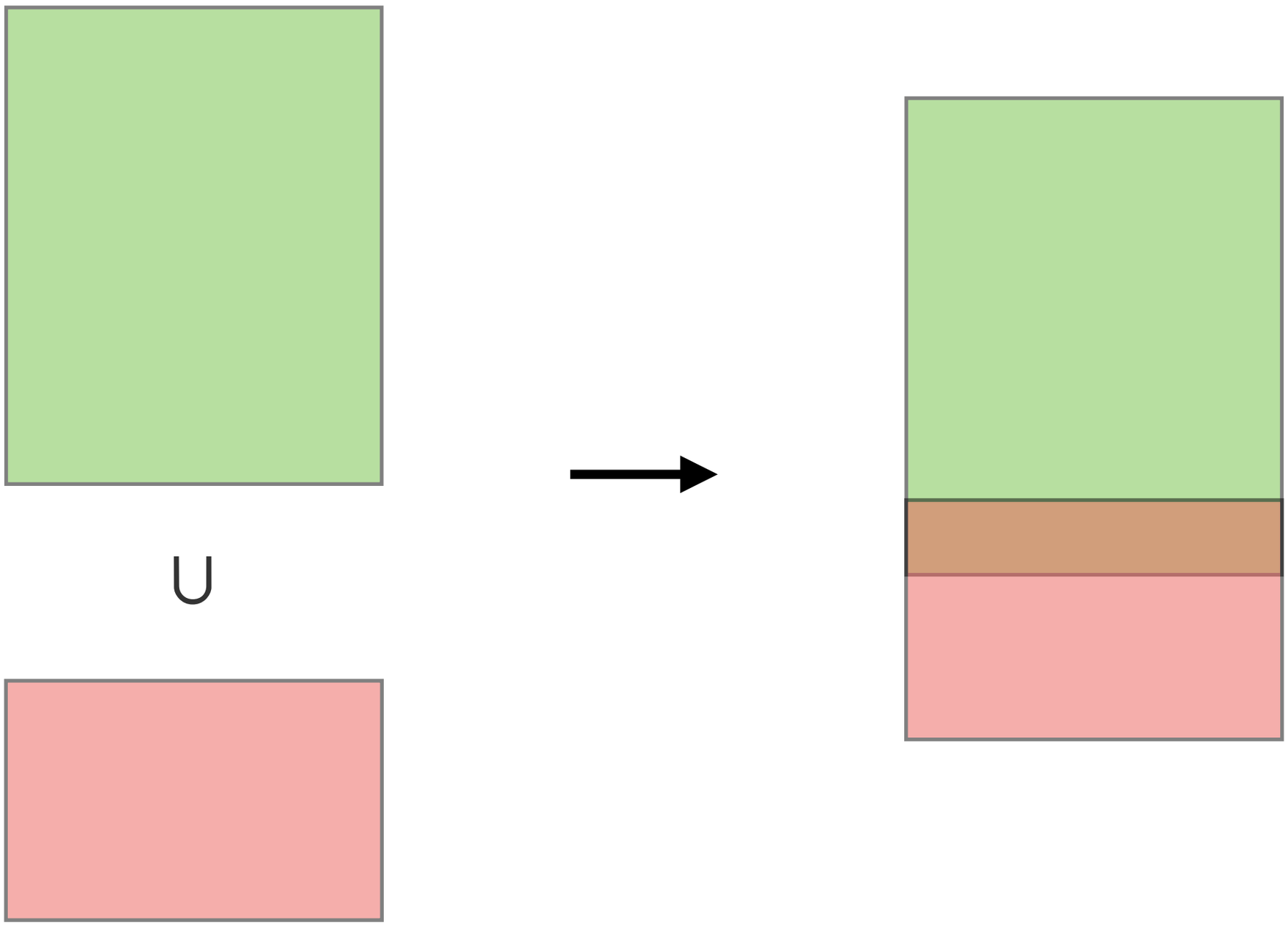
SID	CAT	ENO	POINTS
101	H	1	10
101	H	2	8
101	M	1	12
102	H	1	9
102	H	2	9
102	M	1	9
103	H	1	5
103	M	1	7

```
WHERE  S.LAST = 'Smith'
```

S.SID	S.FIRST	S.LAST	R.SID	R.CAT	R.ENO	R.POINTS
101	Ann	Smith	101	H	1	10
101	Ann	Smith	101	H	2	8
101	Ann	Smith	101	M	1	12

Set Operations

Vertical Combination of Tables



Set Operations

- Compare content of two tables and produce result by comparing all attributes
  - `UNION` / `UNION ALL`: Combine two independent tables
  - `EXCEPT`: Rows from the first table except rows found is in the second table
  - `INTERSECT`: Only rows appearing in both tables
- Number of columns and types have to match between tables

Combining Results

First Query

```
SELECT SID, CAT
FROM   RESULTS
WHERE  SID = 103
```

Set Operation

```
UNION ALL
```

```
SELECT ENO, POINTS
FROM   RESULTS
WHERE  POINTS >= 10
```

← Second Query

Columns need to match!

SID	CAT	ENO	POINTS
101	H	1	10
101	H	2	8
101	M	1	12
102	H	1	9
102	H	2	9
102	M	1	9
103	H	1	5
103	M	1	7

SID	CAT	ENO	POINTS
101	H	1	10
101	H	2	8
101	M	1	12
102	H	1	9
102	H	2	9
102	M	1	9
103	H	1	5
103	M	1	7

```
SELECT SID, CAT
FROM   RESULTS
WHERE  SID = 103
```

```
SELECT POINTS, CAT
FROM   RESULTS
WHERE  POINTS >= 10
```

?	?
103	H
103	M
10	H
12	M

UNION ALL

Results makes no sense...

Evaluation Order

```
4     SELECT   R.ENO, AVG(POINTS) AS AV
      FROM     STUDENTS S JOIN RESULTS R
1       ON     (S.SID = R.SID)
2     WHERE      S.LAST = 'Smith'
3     GROUP BY R.ENO
      ORDER BY AV
5     LIMIT      10
```

Conceptually, not actually

Subqueries

- Nested queries
  - Filters, Comparisons, `EXISTS` / `x IN`
  - Table-Producing, `FROM (SELECT ... )`
- Somewhat similar to functions in programming
- Can refer to attributes from outer query ("correlated")

Scalar Subqueries

No duplicates

"Get the exercise number for which the highest score was given"

```
SELECT DISTINCT ENO
FROM   RESULTS
WHERE  POINTS = 12
```

Hard-Coded Constant :(

```
SELECT DISTINCT ENO
FROM   RESULTS
WHERE  POINTS = (
   SELECT MAX(POINTS) FROM RESULTS )
```

Scalar Subquery

Correlated Subqueries

"Get the students without results"

```
SELECT FIRST, LAST
FROM STUDENTS
WHERE NOT EXISTS (
   SELECT *
   FROM RESULTS
   WHERE RESULTS.SID = STUDENTS.SID )
```

Set Subquery

Column from outer query!

```sql
SELECT FIRST, LAST
FROM STUDENTS
WHERE NOT EXISTS (
   SELECT *
   FROM RESULTS
   WHERE RESULTS.SID = STUDENTS.SID )
```

Subquery is re-run for every row in STUDENTS*

```
NOT EXISTS (SELECT * FROM RESULTS WHERE RESULTS.SID = 101) → False
NOT EXISTS (SELECT * FROM RESULTS WHERE RESULTS.SID = 102) → False
NOT EXISTS (SELECT * FROM RESULTS WHERE RESULTS.SID = 103) → False
NOT EXISTS (SELECT * FROM RESULTS WHERE RESULTS.SID = 104) → True
```

Students

SID	FIRST	LAST
101	Ann	Smith
102	Michael	Jones
103	Richard	Turner
104	Maria	Brown

Results

SID	CAT	ENO	POINTS
101	H	1	10
101	H	2	8
101	M	1	12
102	H	1	9
102	H	2	9
102	M	1	9
103	H	1	5
103	M	1	7

Window Functions

Window Functions

- SQL expressions are **position-independent**
- The **order** of rows does not matter
- Rows cannot reference their neighbors
  - But sometimes you want to do this!

- In come **window functions**

Window Functions

- Window functions allow computation of aggregates over a **window**

Transactions

Account	Date	Amount
Ann	Feb 10	100
Michael	Feb 12	25
Ann	Feb 15	-50
Ann	Feb 16	-25
Michael	Feb 18	15
Ann	Feb 20	30
Michael	Feb 22	-50
Michael	Feb 23	70

- Example: Has any account ever gone below 0?

Window Functions

Partition information

```
SELECT account, date, amount,
   SUM(amount) OVER
      (PARTITION BY account
         ORDER BY date)
FROM transactions;
```

Aggregate

Order in which we want to view the rows

Window Functions

```
SELECT account, date, amount,
    SUM(amount) OVER
        (PARTITION BY account
            ORDER BY date)
FROM transactions;
```

Start by **partitioning** and **ordering** as specified

Transactions

Account	Date	Amount
Ann	Feb 10	100
Ann	Feb 15	-50
Ann	Feb 16	-25
Ann	Feb 20	30
Michael	Feb 12	25
Michael	Feb 18	15
Michael	Feb 22	-50
Michael	Feb 23	70

Window Functions

```
SELECT account, date, amount,
    SUM(amount) OVER
        (PARTITION BY account
            ORDER BY date) AS total
FROM transactions;
```

Transactions

Account	Date	Amount	Total
Ann	Feb 10	100	**100**
Ann	Feb 15	-50	**50**
Ann	Feb 16	-25	**25**
Ann	Feb 20	30	**55**
Michael	Feb 12	25	**25**
Michael	Feb 18	15	**40**
Michael	Feb 22	-50	**-10**
Michael	Feb 23	70	**60**

Within each partition, the aggregate is computed

Note that unlike aggregates, window functions **do not** change cardinality

Window Functions

- The **frame specification** can be specified
  - **Which set of neighboring rows** to consider on
- By default the **frame specification** is `ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW`

Window Functions

- **`ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW`**
- **UNBOUNDED PRECEDING**: All rows **before this row**
- **CURRENT ROW:** The current row

Transactions

Account	Date	Amount	Total
Ann	Feb 10	100	100
Ann	Feb 15	-50	50
Ann	Feb 16	-25	**25**
Ann	Feb 20	30	55
Michael	Feb 12	25	25
Michael	Feb 18	15	40
Michael	Feb 22	-50	-10
Michael	Feb 23	70	60

← To compute this row, we look at all rows before it (within the current partition)

Window Functions

```
SELECT account, date, amount,
    SUM(amount) OVER (
    PARTITION BY account
        ORDER BY date
      ROWS BETWEEN
        1 PRECEDING AND
        1 FOLLOWING)
FROM transactions;
```

Change which neighboring rows are considered

Window Functions

- **`ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING`**
- **1 PRECEDING**: The previous row
- **1 FOLLOWING:** The next row

Transactions

Account	Date	Amount	Total
Ann	Feb 10	100	50
Ann	Feb 15	-50	25
Ann	Feb 16	-25	**-45**
Ann	Feb 20	30	5
Michael	Feb 12	25	40
Michael	Feb 18	15	-10
Michael	Feb 22	-50	35
Michael	Feb 23	70	25

← Look at the row before it, the current row, and the next row

Why SQL on Big Data?

- People know SQL, no need for re-training
- Declarative query language allows optimisation

SQL partitioning

- SQL has no notion of parallelism
  - Does not care how result is produced
- Data is on multiple computers
- Need to rewrite query to run in parallel
  - Old & hard problem, especially considering slow network
  - "Exchange operator"

```
SELECT CAT, COUNT(CAT)
FROM RESULTS
GROUP BY CAT;
```

Results

SID	CAT	ENO	POINTS
101	H	1	10
101	H	2	8
101	M	1	12
102	H	1	9
102	H	2	9
102	M	1	9
103	H	1	5
103	M	1	7

Query Result

CAT	COUNT
H	5
M	3

Partition by row

```
SELECT CAT, COUNT(CAT)
FROM RESULTS
GROUP BY CAT;
```

Results-1

SID	CAT	ENO	POINTS
101	H	1	10
101	H	2	8
101	M	1	12
102	H	1	9

Results-2

SID	CAT	ENO	POINTS
102	H	2	9
102	M	1	9
103	H	1	5
103	M	1	7

Query Result 1

CAT	COUNT
H	3
M	1

← Merge →

Query Result 2

CAT	COUNT
H	2
M	2

Sometimes impossible :/

```
SELECT CAT, COUNT(CAT)
FROM RESULTS
GROUP BY CAT;
```

Partition by CAT

Results-1

SID	CAT	ENO	POINTS
101	H	1	10
101	H	2	8
101	H	1	12
102	H	1	9
103	H	1	5

Results-2

SID	CAT	ENO	POINTS
101	M	1	12
102	M	1	9
103	M	1	7

Query Result 1

CAT	COUNT
H	5

← Union →

Query Result 2

CAT	COUNT
M	3

Partitioning Decision

- Partitioning relevant for distributed aggregations, joins, ...
- How should system decide on how to partition?
  - Easy: Make user tell you
  - Pitfall: "Works" for good & bad partitioning
    - But orders of magnitude speed difference
    - Learn to interpret query plans!
- Partitioning that works for one query might not work for others :/
- Big Data: **Ad-Hoc** repartitioning & replication!

Performance Enablers

- Schema, Query & Partitioning: User
  - directly influenced
- Execution Performance: System
  - influenced by choice of system
- Relevant factor: Cycles/Value
  - How many CPU cycles does it take to process 1 data field
- Topics
  - Data Layout & Compression
  - Indexing

Data Layout & Compression

- Columnar & Lightweight compression state of the art
- Trade-Off IO / CPU
  - CPU typically starved of data to process
    - Data Lake: Medium compression (GZIP, Snappy)
    - RAM: lightweight compression (RLE, Dict, Diff)
- Synergy! Columnar data increases compression efficiency
  - Same-distribution data together
- Even better: Operate directly on compressed data

Practicum 3