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. ETL on Big Data
   Practicum

   Lunch

3. SQL on Big Data
   Practicum

4. Machine Learning on Big Data?

Organization

1. Materials & Slides on Google Docs

CWI

Database Architectures

Big Data Infrastructures & Technology

Introduction

Topics Today

1. **Introduction**
2. ETL on Big Data

   Practicum

   Lunch
3. SQL on Big Data

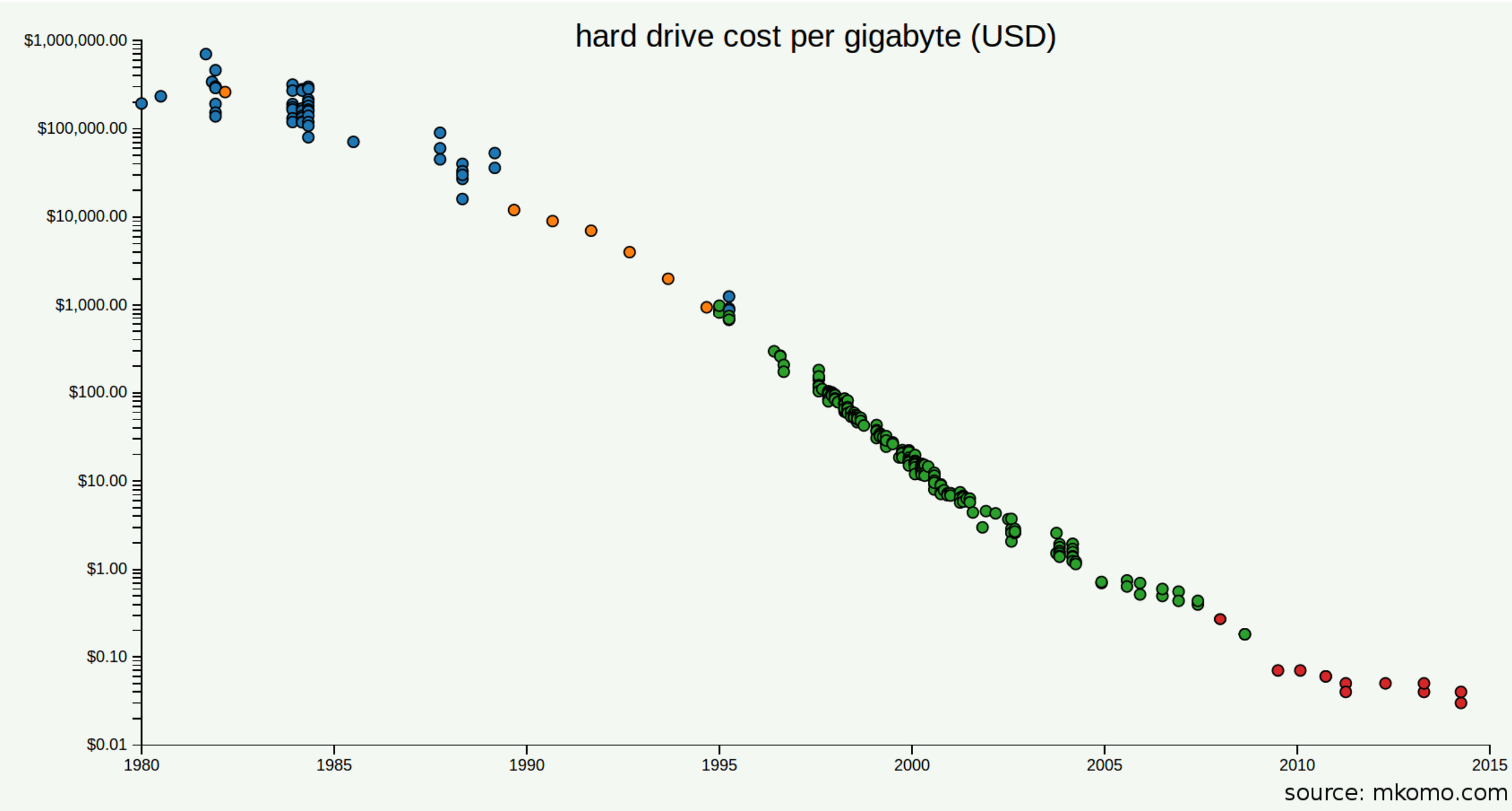
   Practicum
4. Machine Learning on Big Data

- 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*

- First movers: Index entire web, build search engine (ca. 2000)
- Step 1: Collect Websites (Terabytes, Petabytes)
- Step 2: **?**
- Step 3: ~~Unclear~~ Advertising Profit

*almost

Large-Scale Data Processing

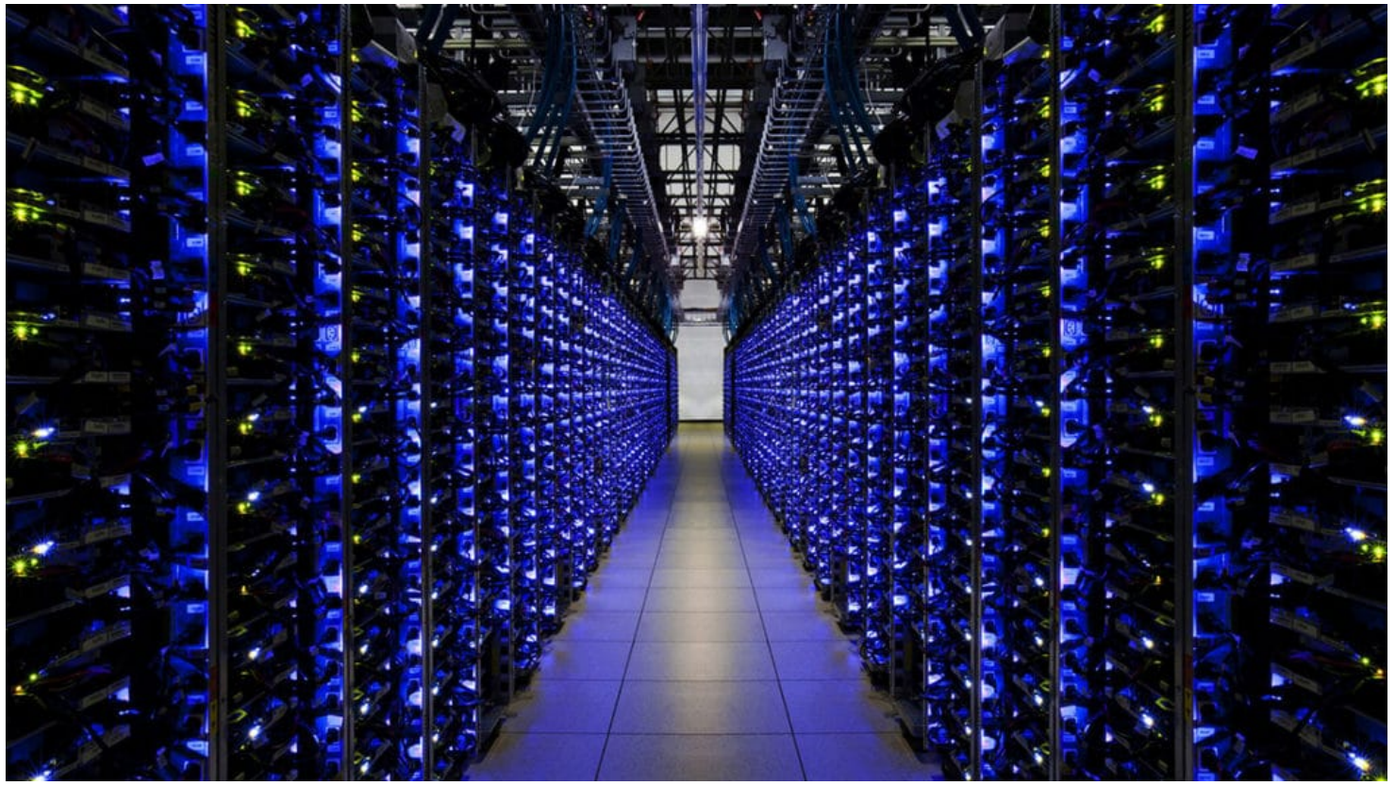
- Step 2 : Build search index (inverted index)
- Problem: No business model yet, so needs to be cheap
- Solution: COTS hardware, lots of it
- Problem: Hard to program distributed systems
  - Coordination, Concurrency, Debugging, Hardware

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]

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

Generalised Data Processing

- Iterate over large number of records
- Extract something of interest from each (**MAP**)
- Shuffle and sort intermediate results
- Aggregate intermediate results (**REDUCE**)
- Generate final output
- Handle failures

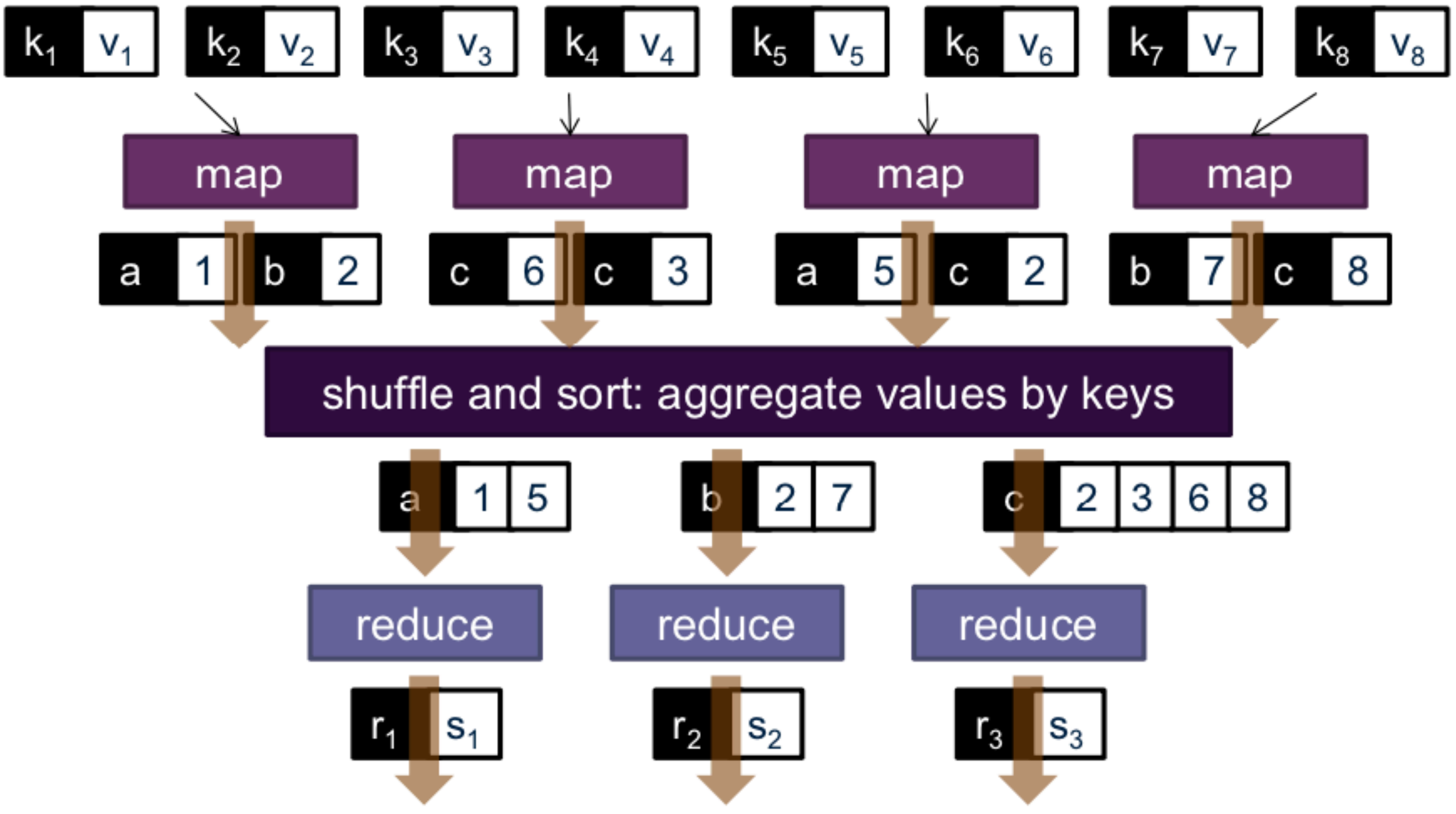
- MapReduce key insight: Provide abstraction!

- Programmers specify two functions:

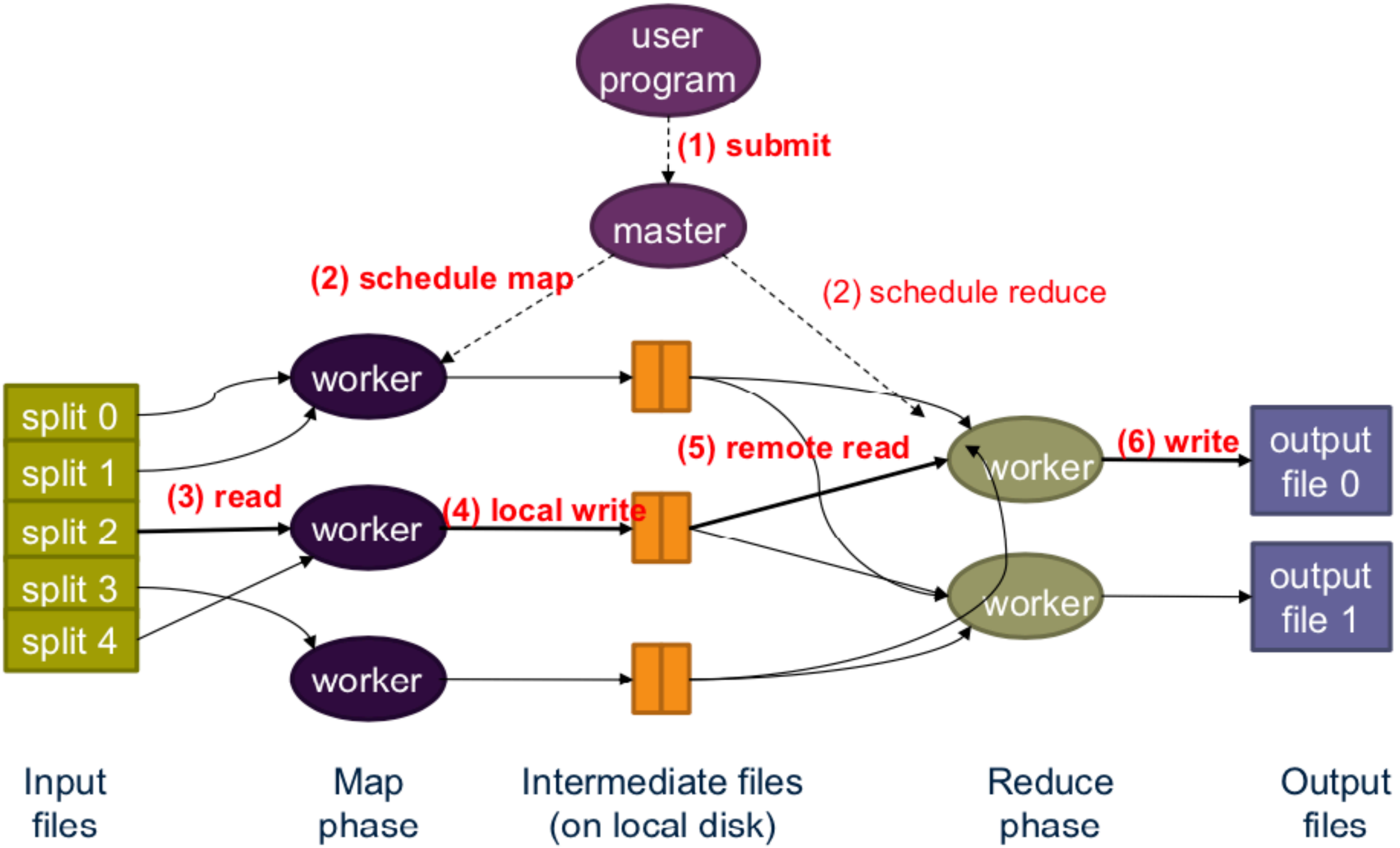
  map $(k_1, v_1) \rightarrow [<k_2, v_2>]$

  reduce $(k_2, [v_2]) \rightarrow [<k_3, v_3>]$

  – All values with the same key are sent to the same reducer



Technical View



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

Embarrassingly parallel?

- MapReduce (and **all** other distributed Big Data tools) only work 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.

1 Introduction

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

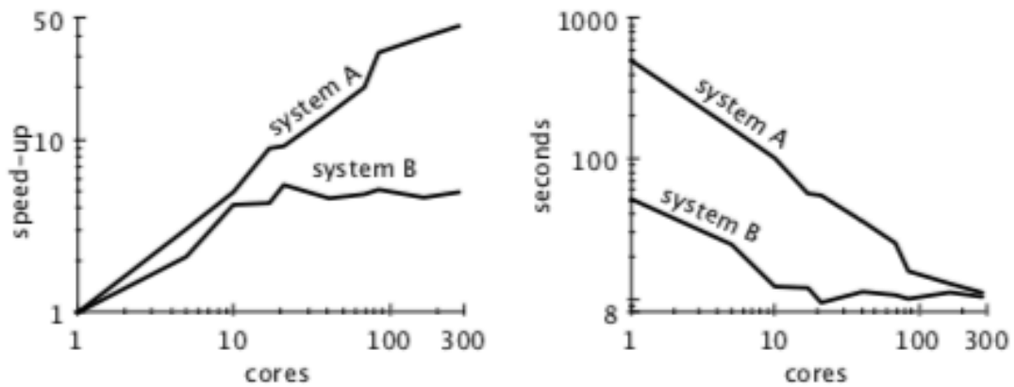


Figure 1: Scaling and performance measurements for a data-parallel algorithm, before (system A) and after (system B) a simple performance optimization. The unoptimized implementation "scales" far better, despite (or rather, because of) its poor performance.

While this may appear to be a contrived example, we will argue that many published big data systems more closely resemble system A than they resemble system B.

1.1 Methodology

In this paper we take several recent graph processing papers from the systems literature and compare their reported performance against simple, single-threaded im-

Generalised Data Processing

- Once tools available, applied to lots of large datasets
  - Everyone wants to be Google

EXPERT OPINION

Contact Editor: **Brian Brannon,** bbrannon@computer.org

The Unreasonable Effectiveness of Data

Alon Halevy, Peter Norvig, and Fernando Pereira, *Google*

Eugene Wigner's article "The Unreasonable Effectiveness of Mathematics in the Natural Sciences"[1] examines why so much of physics can be neatly explained with simple mathematical formulas such as $f = ma$ or $e = mc^2$. Meanwhile, sciences that involve human beings rather than elementary particles have proven more resistant to elegant mathematics. Economists suffer from physics envy over their inability to neatly model human behavior. An informal, incomplete grammar of the English

behavior. So, this corpus could serve as the basis of a complete model for certain tasks—if only we knew how to extract the model from the data.

Learning from Text at Web Scale

The biggest successes in natural-language-related machine learning have been statistical speech recognition and statistical machine translation. The reason for these successes is not that these tasks are easier than other tasks; they are in fact much harder than tasks such as document classification that extract just a few bits of information from each doc-

2009

Google

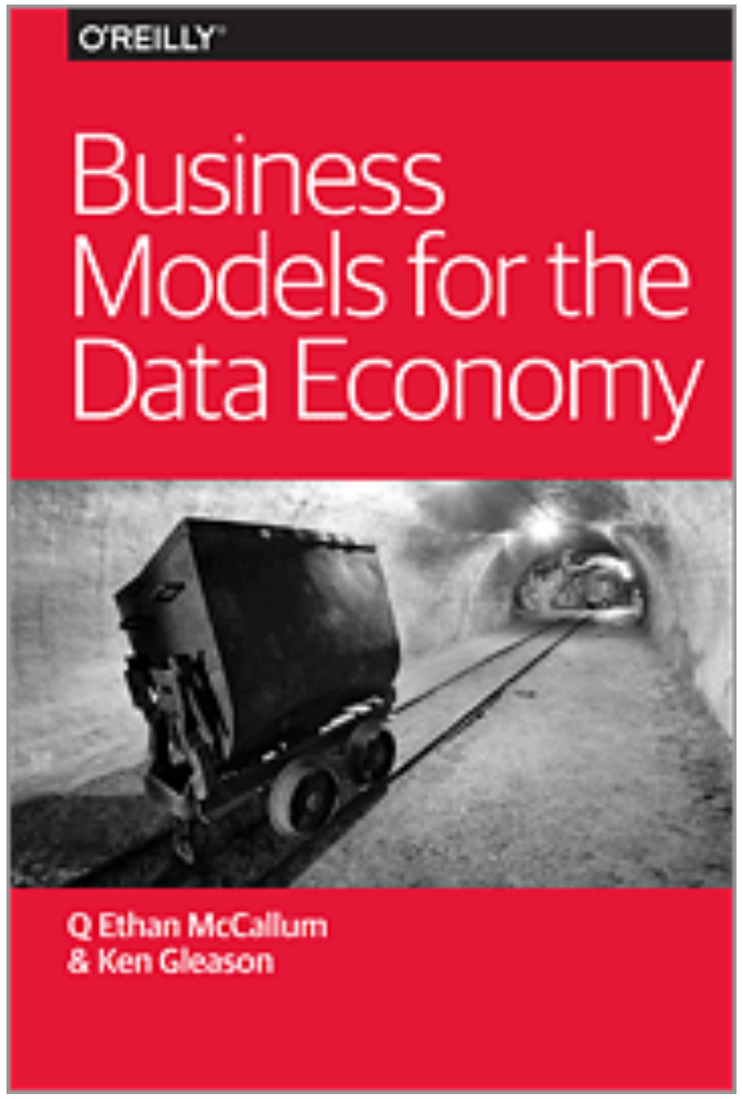
Detecting influenza epidemics using search engine query data

Jeremy Ginsberg[1], Matthew H. Mohebbi[1], Rajan S. Patel[1], Lynnette Brammer[2], Mark S. Smolinski[1] & Larry Brilliant[1]

[1]Google Inc. [2]Centers for Disease Control and Prevention

Epidemics of seasonal influenza are a major public health concern, causing tens of millions of respiratory illnesses and 250,000 to 500,000 deaths worldwide each year[1]. In addition to seasonal influenza, a new strain of influenza virus against which no prior immunity exists and that demonstrates human-to-human transmission could result in a pandemic with millions of fatalities[2]. Early detection of disease activity, when followed by a rapid response, can reduce the impact of both seasonal and pandemic influenza[3,4]. One way to improve early detection is to monitor health-seeking behavior in the form of online web search queries, which are submitted by millions of users around the world each day. Here we present a method of analyzing large numbers of Google search queries to track influenza-like illness in a population. Because the relative frequency of certain queries is highly correlated with the percentage of physician visits in which a patient presents with influenza-like symptoms, we can accurately estimate the current level of weekly influenza activity in each region of the United States, with a reporting lag of about one day. This approach may make it possible to utilize search

2009


O'REILLY
Business Models for the Data Economy
Q Ethan McCallum
& Ken Gleason


Turning Government Data into Gold
ec.europa.eu

BIG DATA

The Parable of Google Flu: Traps in Big Data Analysis

Large errors in flu prediction were largely avoidable, which offers lessons for the use of big data.

2014

David Lazer,[1,2]* Ryan Kennedy,[1,3,4] Gary King,[3] Alessandro Vespignani[3,5,6]

In February 2013, Google Flu Trends (GFT) made headlines but not for a reason that Google executives or the creators of the flu tracking system would have hoped. *Nature* reported that GFT was predicting more than double the proportion of doctor visits for influenza-like illness (ILI) than the Centers for Disease Control and Prevention (CDC), which bases its estimates on surveillance reports from laboratories across the United States (*1*, *2*). This happened despite the fact that GFT was built to predict CDC reports. Given that GFT is often held up as an exemplary use of big data (*3*, *4*), what lessons can we draw from this error?

The problems we identify are not limited to GFT. Research on whether search or social media can predict *x* has become commonplace (*5*–*7*) and is often put in sharp contrast with traditional methods and hypotheses. Although these studies have shown the value of these data, we are far from a place where they can supplant more traditional methods or theories (*8*). We explore two

surement and construct validity and reliability and dependencies among data (*12*). The core challenge is that most big data that have received popular attention are not the output of instruments designed to produce valid and reliable data amenable for scien-

the algorithm in 2009, and this model has run ever since, with a few changes announced in October 2013 (*10*, *15*).

Although not widely reported until 2013, the new GFT has been persistently overestimating flu prevalence for a much longer time. GFT also missed by a very large margin in the 2011–2012 flu season and has missed high for 100 out of 108 weeks starting with August 2011 (see the graph). These errors are not randomly distributed. For example, last week's errors predict this week's errors (temporal autocorrelation), and the direction and magnitude of error varies with the time of year (seasonality). These patterns mean that GFT overlooks considerable information that could be extracted by traditional statistical methods.

Even after GFT was updated in 2009, the comparative value of the algorithm as a stand-alone flu monitor is questionable. A study in 2010 demonstrated that GFT accuracy was not much better than a fairly simple projection forward using already avail-

Big Data 2022

- Hype is over, thankfully.
- Your Gemeente is not Google
- Problems:
  - Illegal. GDPR requires a reason to store something
  - Lots of people store lots of data. What are cool applications in the real world?
  - You might leak this data. Big problem.
  - Once the data exists, shady entities want access

No Magic

- Big Data Technology ≈ Parallel Data Processing
- When should one use this technology?
- **Actual** large amount of data and effort to process.
  - 10 GB maybe?
  - 10 TB maybe?
  - 10 PB maybe?
- Absolute dataset size meaningless. What part is relevant? How expensive is processing?

Single Machine

- If we can use a single machine to process data in acceptable time, we should always!
  - Setup overhead
  - Processing overhead (Communication etc.)
  - Terrible to debug
- Single-machine solutions always cheaper and easier
  - Rent 12 TB RAM AWS VM if MacBook too small
- So when **should** we use Parallel Data Processing?

Example 1: Log Data

- Webserver log data (e.g. Apache)
  127.0.0.1 - peter [9/Feb/2017:10:34:12 -0700] "GET /sample-image.png HTTP/2" 200 1479
- Every HTTP request creates log entry
- 1 TB of data (text files)
- Example question: What part of site is most unstable (Error code 500).
- What do we do?

Example 1: Log Data

- Only a tiny amount of requests will be errors (hopefully)
- Need to read all the data, filter, aggregate errors by path
- Hard disk drive read speed: 300 MB/s (max!)
- 1 TB at 300 MB/s ~ 1h of data reading
- Rest has negligible overhead
- 1h on **single** standard computer
- **No need** for parallel data processing!

Example 2: Pictures

- Picture collection, 1M pictures, 4 GB of data
- Example question: Get pictures showing Julian Assange
- What do we do?

Example 2: Pictures

- Need to detect faces on all pictures,
  return those who show Assange.
- Face detection is computationally expensive.
  Takes 0.5s/pic.
- Disk read speed irrelevant
- 1M pics at 2 pics/s ~ 1 week on single computer
- **Need** parallel data processing!

Example 3: Speech Recognition

- Train the Alexa speech recognition neural network
- Have 100MB/h * 10 years of audio ~ 1 TB of input data
- Example question: Optimise network to minimise errors
- What do we do?

Example 3: Speech Recognition

- Training neural network processes 10 GB/hour
- Disk read speed irrelevant
- 4 days on single computer
- However, this problem cannot be parallelised
  - every training step depends on all previous ones.
- **Cannot use** parallel data processing!

Should we use parallel data processing

- Is the problem parallelizable at all?
  - **No** = Cannot use!
- Does the data fit on a single machine?
  - **No** = Use parallel data processing
- How long will it take to process data on single machine
  - **Fast**? = Do not use!
  - **Slow**? = Use parallel data processing

Very unscientific rules of thumb

- Business data (orders, customers) will fit on single machine
  - You are not Werner Vogels (CTO of Amazon)
- Computer-generated data (server logs, high-frequency sensor data) will likely not fit on single machine
- Anything involving ML will be slow, so parallelise if possible

Delta Example

https://thepointsguy.com/news/this-is-the-reason-you-arent-feeling-as-much-turbulence-on-delta-flights/

Pokemon or Big Data?

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

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, AEROSPIKE, Couchbase, SequoiaDB, redislabs, influxdata

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

Graph Databases
neo4j, GRAPH, OrientDB, InfiniteGraph

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

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

Data Transformation
alteryx, Preferred Partner, 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, Trust in the Cloud, 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, Artificial Artificial Intelligence, 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, plotly, ARIMO, MODE, 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, SCALED INFERENCE, Descartes Labs, clarifai, DEXTRO, Geometric Intelligence, MetaMind

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

Data Services
Mu Sigma, DO THE MATH, OPERA SOLUTIONS, EXL analytics, DATASCIENCE, SILICON VALLEY DATA SCIENCE, kaggle, datascope, DataKind

For Business Analysts
OrigamiLogic, ClearStory Data, Now You See It, 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, eBrevia, PREMONITION

Ad Optimization
AppNexus, MediaMath, criteo, OpenX, rocketfuel, Artificial Intelligence. Real results., 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, FN FiscalNote, enigma, PREDPOL, 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, Comprehensive In-Store Analytics, 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, Apache Parquet, cascading, SciDB, kafka, nifi, riak, OPENTSDB

Coordination
talend, DOZER, 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, WEKA, The University of Waikato, VELES, 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, CONNECTING THE REAL WORLD, esri, Crimson Hexagon, CARTODB, factual, PlaceIQ, CIRCULATE, placemeter, BASIS TECHNOLOGY, Sense 360

Other
qualtrics, panjiva, DATA.GOV

Incubators & Schools

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

Last Updated 3/23/2016

FIRSTMARK

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, VOLTDB, MariaDB, citusdata, deepdb, Trafodion, CockroachDB

Graph Databases
neo4j, GIRAPH, OrientDB, InfiniteGraph

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

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

Transformation
alteryx, Preferred Partner, 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, Trust in the Cloud, 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, Artificial Artificial Intelligence, 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, ACADIA, SIGNIFY

Statistical Computing
sas, SPSS, MATLAB

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

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

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

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

Speech & NLP
Nara, NUANCE, WolframAlpha, semantic machines, Gridspace, ARRIA, api.ai, cortical.io, maluuba, MindMeld, iDIBON, yseop

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

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

Data Services
Mu Sigma, OPERA SOLUTIONS, DATASCIENCE, EXL analytics, SILICON VALLEY DATA SCIENCE, kaggle, datascope, DataKind

For Business Analysts
OrigamiLogic, ClearStory DATA, 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, theTradeDesk, Ad Science, Adgorithms, dstillery, LiveIntent, TAPAD, DataXu, Appier, MOAT

Security
CYLANCE, CounterTack, cybereason, AREA 1 SECURITY, ThreatMetrix, SentinelOne, Recorded Future, Guardian Analytics, FORTSCALE, sift science, Keybase, feedzai, SIGNIFY

Vertical AI Applications
facebook, Clara, KASISTO, lumiata

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

Govt / Regulation
Socrata, OPENGOV, FiscalNote, enigma, PREDPOL, 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, knowre

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

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

~ This Course

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, OOZIE, Apache Zookeeper, Apache Ambari

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

Stat Tools
ScalaLab, NumPy, SciPy

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

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

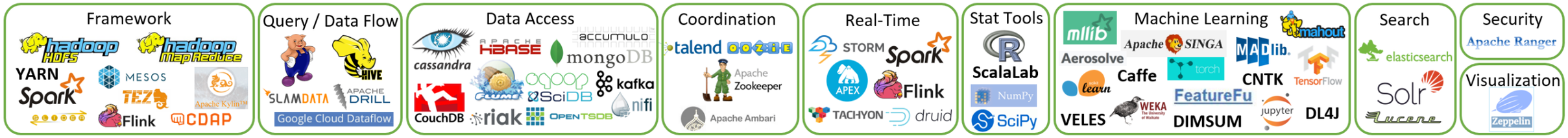
Air / Space / Sea
PLANET LABS, spire, WINDWARD, CRUISE, SKYCATCH, Airware, DroneDeploy

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

FIRSTMARK

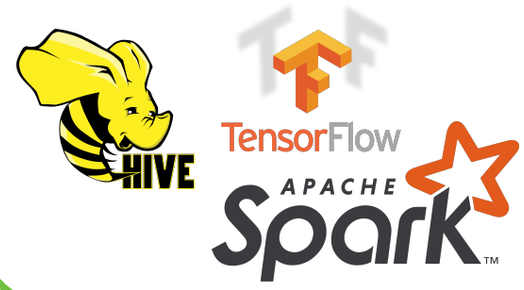


? ? ? ?

Storage

Framework

Query/DataFlow



“Hard Disk” “Operating System” “Excel”

“The datacenter is the computer”

Data Storage

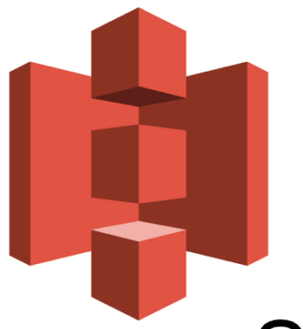
- Traditional way: Database “owns” data.
- Big Data way: Data is stored in shared storage in open formats. **Need storage system and file format.**
  - Big advantage
    - no vendor lock-in, multiple tools can operate on same data
- Popular storage systems: HDFS, S3, CEPH, …
- Popular formats: Plaintext, CSV, Parquet
- Metadata storage: HCatalog (for tables)

Storage System: HDFS

- “Hadoop Distributed File System”
  - Ripoff of GFS, the “Google File System”
- De-facto implementation of “Data Lake”
- Cluster of computers that provide one large virtual file system
- Redundant storage to cope with hardware failure
- Normal files and folders
- Quirk:
  - Works best if files > 10 MB
  - cannot change file contents once they are written

Storage System: S3

- “Simple Storage Service”, Amazon Web Services
  - Key-Value store for files
- Service, users pay for storage space and requests
- Eventually-consistent
- Only file names (can be hierarchically organized)
- Storage foundation of entire Amazon Cloud



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

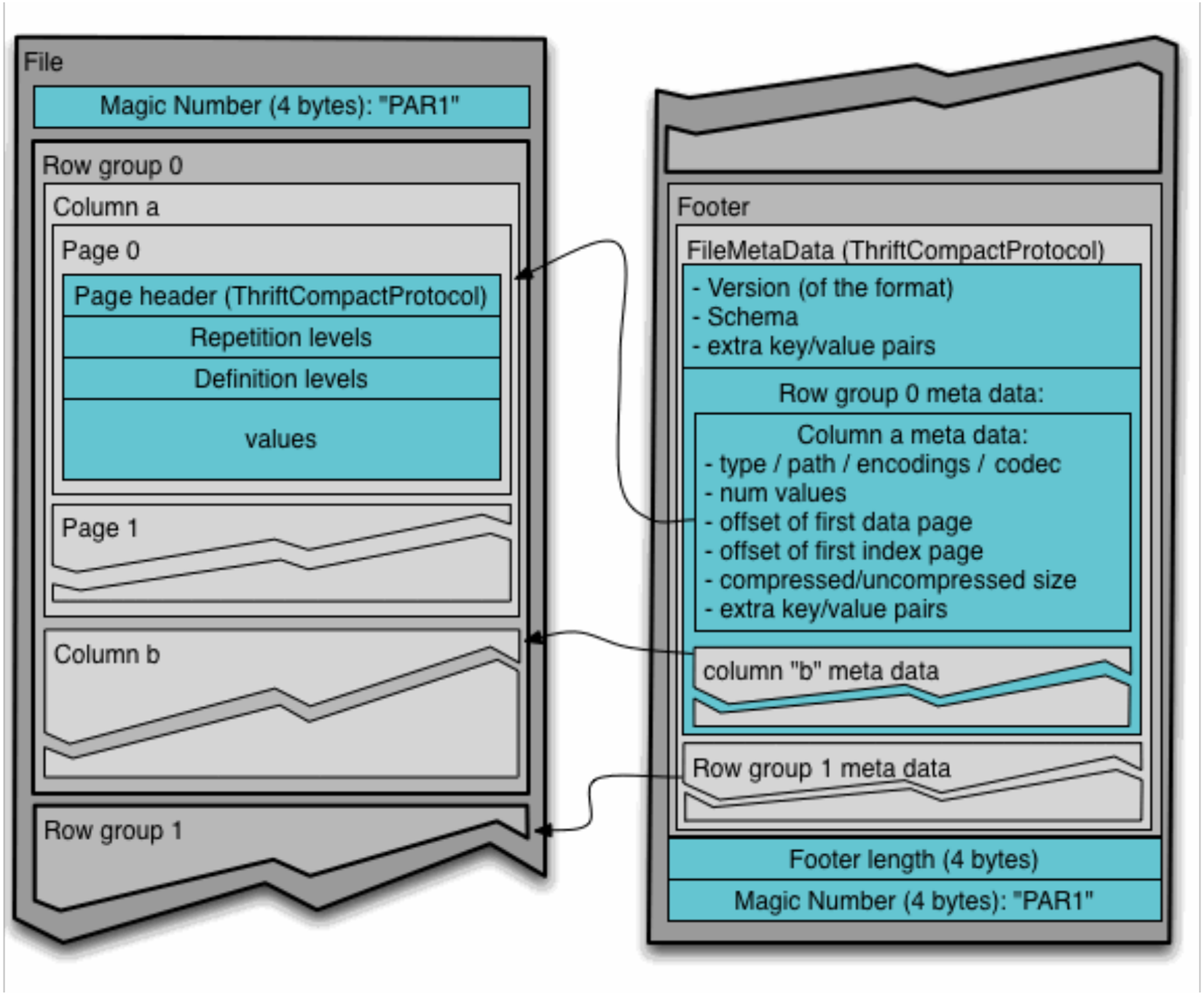
- Tables as text
- Structure, but might be broken
  - Can be hard to figure out structure
- Most tools can process CSV
  - But inefficient
- Very common

```
id,name,released_on,price,created_at,updated_at
24,1000 Piece Jigsaw Puzzle,2012-07-03,14.99,2012-07-
30,360° Protractor,2012-05-03,3.99,2012-07-09 16:50:49
17,7 Wonders,2012-04-21,28.75,2012-07-09 16:50:49 UTC
13,Acoustic Guitar,2012-06-06,1025.0,2012-07-09 16:50
15,Agricola,2012-05-22,45.99,2012-07-09 16:50:49 UTC,
22,Answer to Everything,2012-07-03,42.0,2012-07-09 16
23,Box Kite,2012-05-19,63.0,2012-07-09 16:50:49 UTC,2
29,CanCan Music Record,2012-05-09,2.99,2012-07-09 16:
12,Chocolate Pie,2012-04-12,3.14,2012-07-09 16:50:49
9,Dog Toy Bone,2012-06-13,2.99,2012-07-09 16:50:49 UTC
11,Flux Capacitor,2012-06-01,19.55,2012-07-09 16:50:4
6,Game Console,2012-06-06,299.95,2012-07-09 16:50:49
10,Heated Blanket,2012-07-19,27.95,2012-07-09 16:50:4
19,Knights of Catan,2012-06-10,19.95,2012-07-09 16:50
8,Lawn Chair,2012-05-29,34.99,2012-07-09 16:50:49 UTC
21,Millennium Falcon,2012-04-10,3597200.0,2012-07-09
14,Model Enterprise,2012-04-18,27.99,2012-07-09 16:50
28,Model Train Rails,2012-06-30,45.0,2012-07-09 16:50
3,Oak Coffee Table,2012-07-08,223.99,2012-07-09 16:50
5,Oh's Cereal,2012-04-17,3.95,2012-07-09 16:50:49 UTC
```

File Formats: Parquet

- Tables, split up by row groups and columns
- Binary format, compressed
- Self-describing
- Very popular because can be read efficiently (by Big Data standards)

File Formats: Parquet



Data Processing

- Original Hadoop “Ecosystem” (aka obsolete Java gunk)
  - MapReduce, Pig, Hive, Mahout, Cascading
- Distributors: Cloudera, Hortonworks, etc.

Data Processing

- Improved Hadoop
  - Impala
  - Spark
  - Flink
  - HBase

Data Processing

- Managed Cloud Services
  - Snowflake
  - Amazon **Athena**
  - Google BigTable
  - DataBricks (Spark++)

Data Processing

- Stand-Alone distributed data processing
- Presto
- Spark on Mesos

Data Processing

- Traditional Parallel Data Warehousing
  - Teradata
  - Vertica
  - Oracle PDW
  - SQL Server

Trend: Higher-Level Interfaces

1. Job Schedulers
   - YARN, Mesos, Kubernetes
2. Single-Job
   - MapReduce
3. Workflow
   - Pig, Spark, Cascading, Flink

Avoid

4. High-Level Interfaces

- Spark SQL, Spark DataFrames, GraphX, TensorFlow

Recommendations

- Reminder: Don't use Big Data technology unless you **absolutely have to**
- General data processing: SQL or DataFrame interface
- Graph/ML: Hard to parallelise efficiently, hence avoid
  - Make problem smaller then use single node ML

Practicum 1

Topics Today

1. Introduction
2. **ETL on Big Data**

   Practicum
3. SQL on Big Data

   Practicum
4. Machine Learning on Big Data

Basic Principle

	c1	c2	c3
r1	42		
r2			
r3			

ETL = First step in Analysis

- Dataset from somewhere, almost always messy
- ETL in Big Data = Making data usable/“analysable”
- Often want to combine multiple sources
  - e.g. Stock Price + Weather
  - Need compatible representation
- Big data tools cannot read arbitrary file formats

ETL = First step in Analysis

- Traditional: ETL from operations to Data Warehouse
  - Controlled process
- Big Data: **ETL from messy gunk to usable table(s)**
- Task-specific!
  - ETL = subsetting (mostly)
  - 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. CSV or 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

Typical Issues 3

- Need extremely robust decoder
  - Huge dataset **will** have broken record somewhere
    - Can take days to get to this entry
  - Third-party decoders often needed (e.g. AIS, XLS, …)
- Count errors!

Automated ETL Pipeline

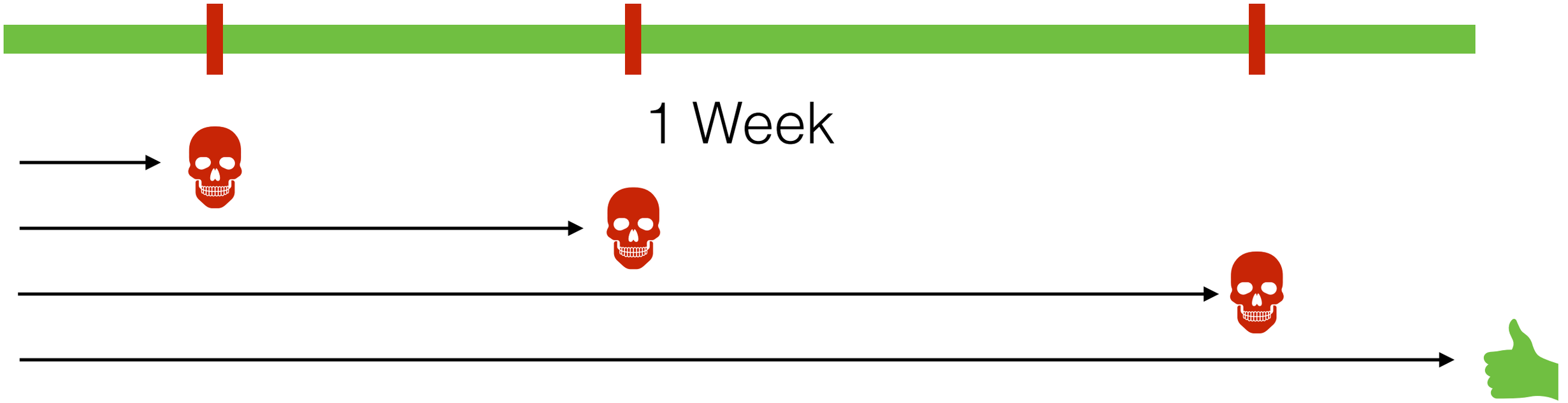
- You **will** have to do the same thing again
  - Crash/Corrupt data
  - New data arrived
  - Requirements change
- Automate everything (no interactive ETL!)
- Document everything (can be script) for future self

Automated ETL Pipeline

- ETL consists of multiple steps
  - Separate ETL from analysis
  - ETL most likely much slower than analysis
  - Don’t want to run ETL every time analysis code changes
  - Downside: Need to keep extracted data
    - Likely smaller than original

Resumability

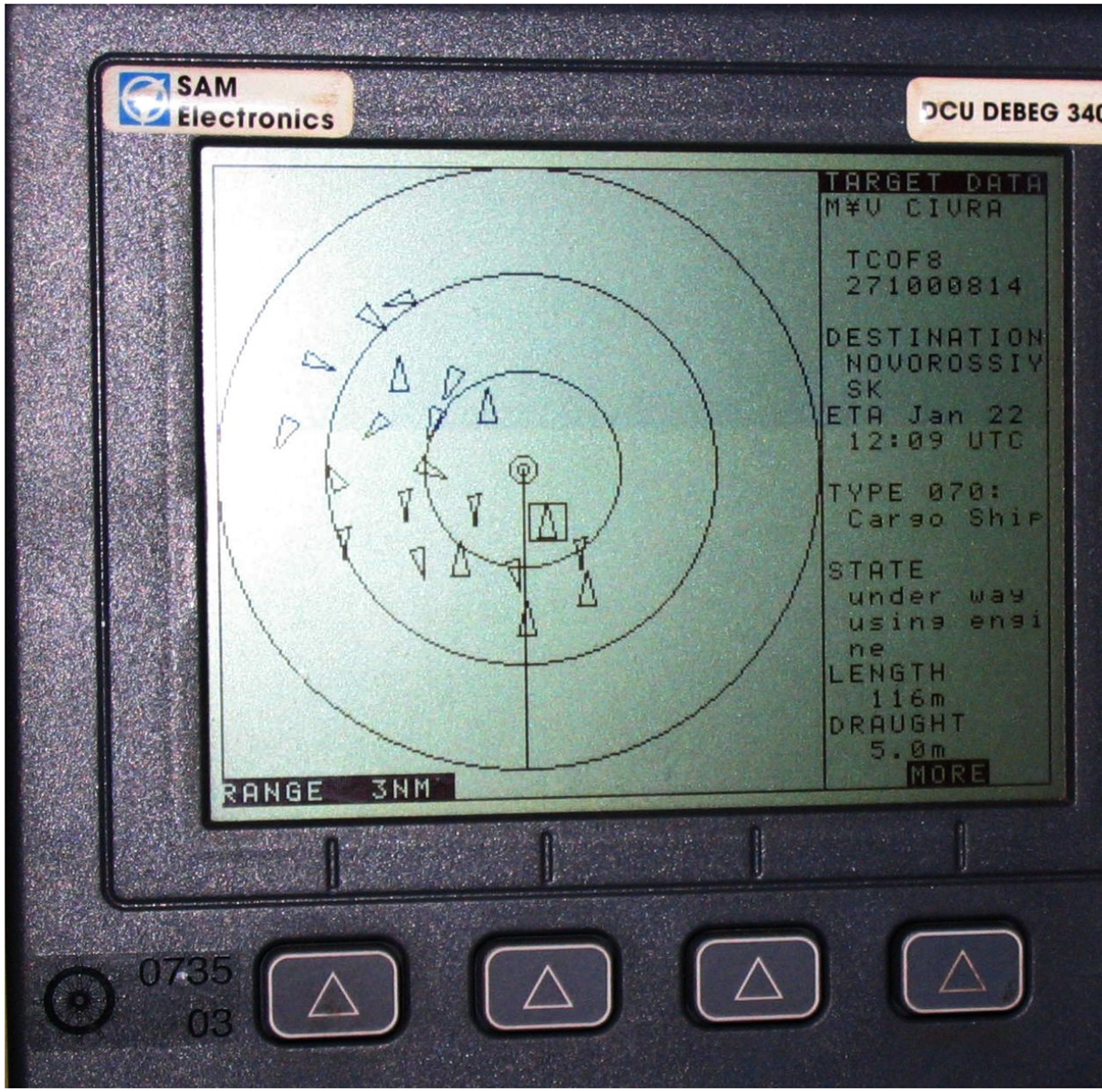
- Make sure pipeline is **resumable**
  - Pipeline might take days
  - And it **will** crash at first



17 days vs 7!

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

Decompress

!AIVDM,1,1,,B,146HbN?P00NqNp>@66`qjgv`0>`<,0*3A
!AIVDM,1,1,,A,B69>@Ih07B: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

MMSI = 275131000
Longitude = -15.404
Latitude = +28.128
...

e.g. libais

Normalize/Store

Msg,RepInd,MMSI,NavStat,ROT,SOG,PosAcc,Long,E/W,Lat,N/S,COG,Heading,TimeStamp
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
1,0,219001125,7,-720.0,3.0,1,12.1691233,E,56.6607600,N,153.8,156,28,0,0,1
1,0,266057000,0,+0.0,0.0,0,12.3524983,E,56.9291867,N,148.6,277,29,0,0,0
1,0,564201000,15,-128.0,0.3,0,103.7503267,E,1.2806883,N,39.7,511,31,0,0,0
1,0,235088195,0,-128.0,0.2,1,1.4226267,E,51.6333650,N,10.7,511,31,0,0,1
1,0,244850098,15,-128.0,0.0,1,5.4381500,E,53.1748717,N,360.0,511,30,0,0,1
1,0,367345780,15,-128.0,0.0,0,90.7178183,W,29.5813033,N,165.2,511,29,0,0,0
1,0,366851250,0,+0.0,2.4,1,90.7053250,W,29.6122367,N,41.9,62,30,0,0,0
1,0,367462460,15,-128.0,0.0,0,90.7233400,W,29.5643217,N,271.2,511,30,0,0,1
1,0,367033420,15,-128.0,0.0,0,90.6722233,W,29.6006417,N,59.8,511,29,0,0,0
1,0,366816890,0,-128.0,0.0,0,90.7013450,W,29.5459183,N,161.9,511,29,0,0,0
1,0,367558180,0,+0.0,0.0,1,75.2074883,W,39.8945333,N,258.7,249,27,0,0,1
1,0,564754000,0,+0.0,10.4,1,74.8771450,W,38.6102650,N,145.0,145,28,0,0,0
1,0,247222500,15,-128.0,0.0,0,9.8328433,E,44.1044783,N,217.4,511,30,0,0,0
1,0,265579090,15,+0.0,0.0,1,11.9664600,E,57.7132083,N,278.4,46,30,0,0,0

Practicum 2

Lunch

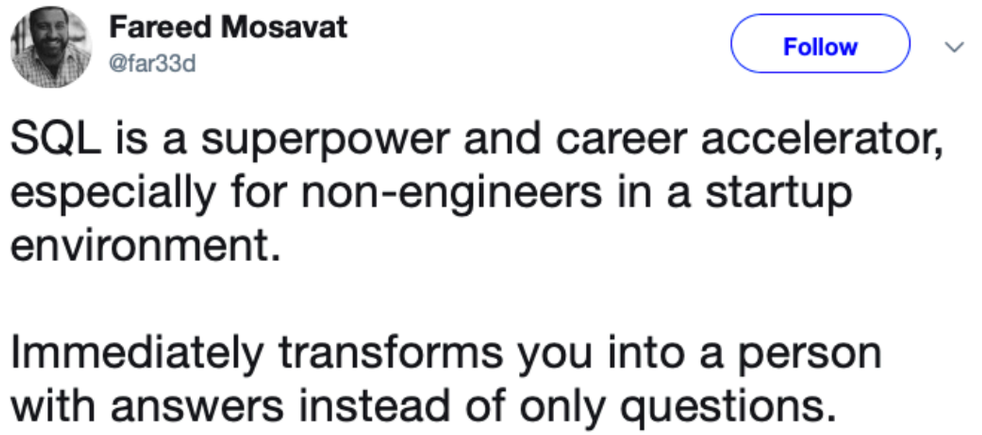
Topics Today

1. Introduction
2. ETL on Big Data
   Practicum
3. **SQL on Big Data**

   Practicum
4. Machine Learning on Big Data

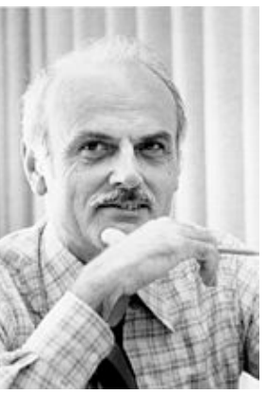
Why SQL

- 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
```

Online SQL learning tool

https://homepages.cwi.nl/~hannes/vudbcoursetool/select.html

Sample Dataset

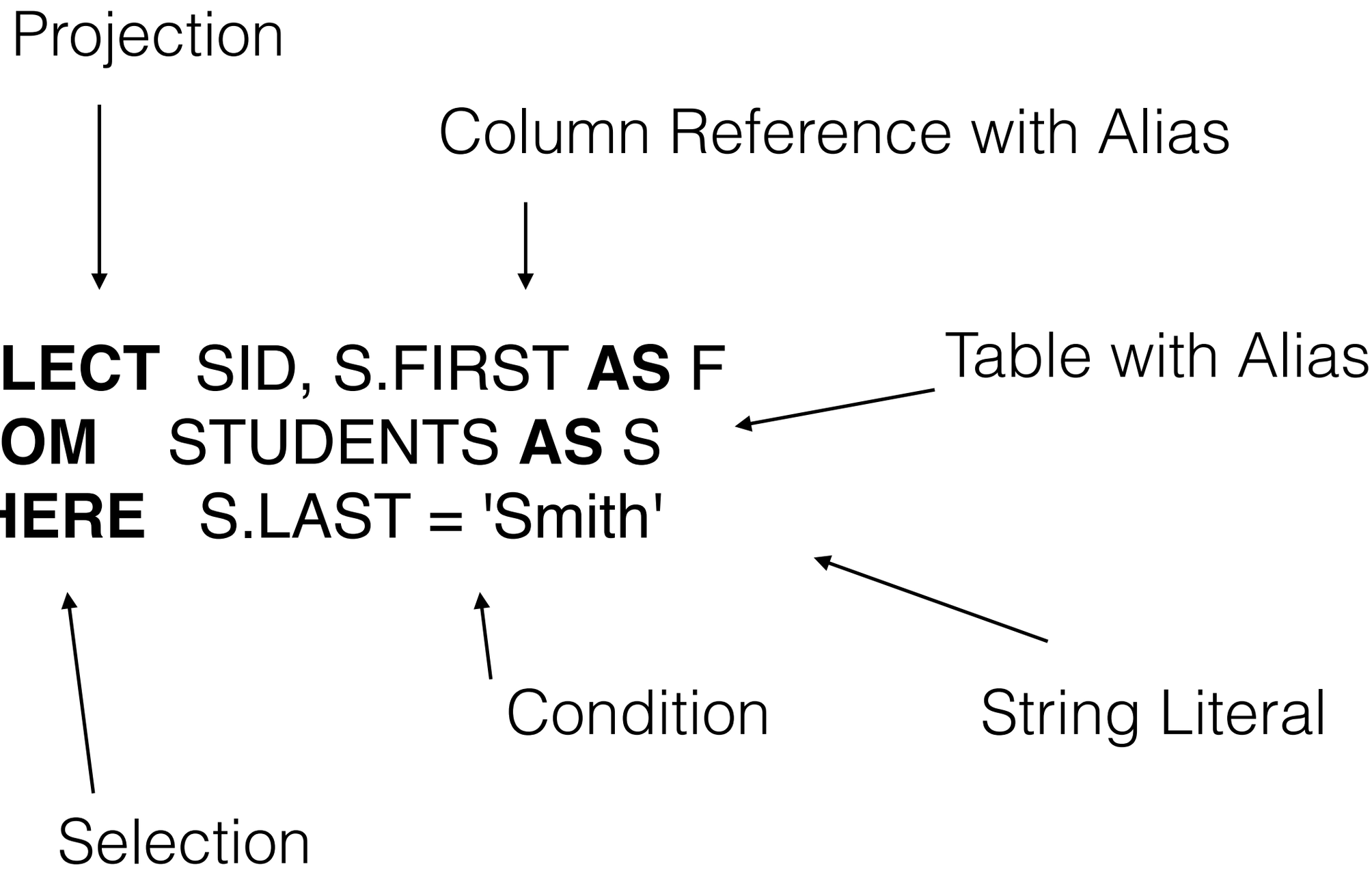
Students

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



```
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'

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

SELECT SID, S.FIRST

Row Selection

Column Projection

Quoting

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

Safe bet!

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

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

COUNT
MIN
MAX
AVG
SUM
STDEV
VAR
...

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

WHERE CAT = 'H'

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

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

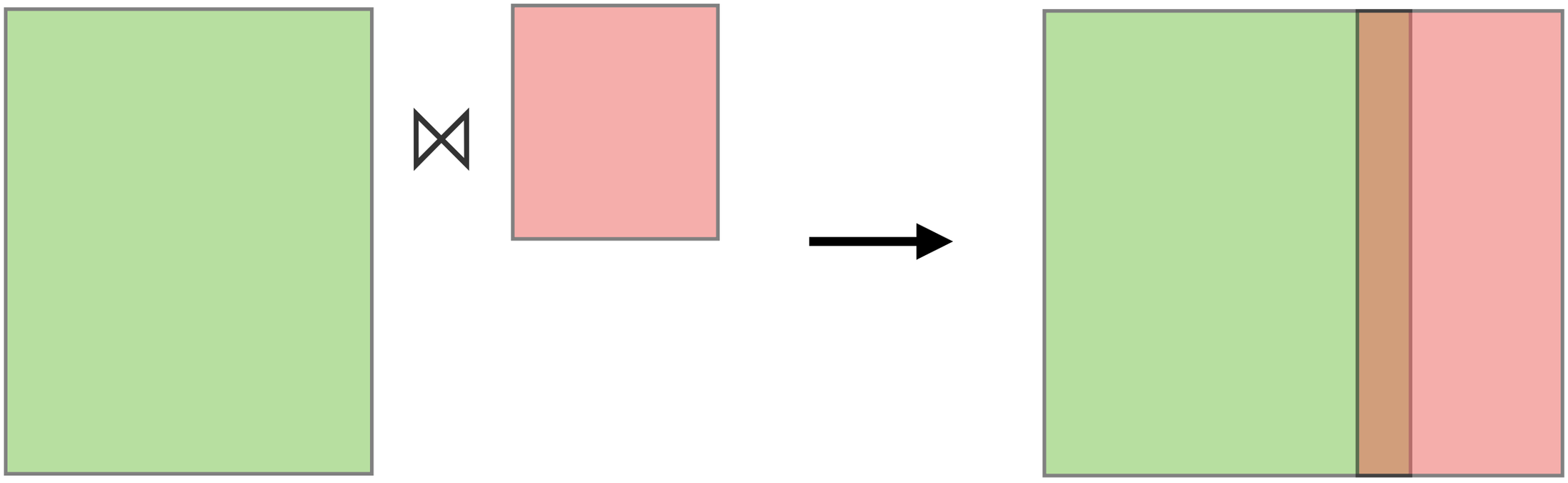
HAVING **AVG**(POINTS) > 8.2

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

Outer Joins

SELECT S.LAST, R.POINTS
FROM STUDENTS S **LEFT OUTER JOIN** RESULTS R
ON (S.SID = R.SID)

https://blog.jooq.org/2016/07/05/say-no-to-venn-diagrams-when-explaining-joins/

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
104	Maria	Brown	**NULL**	**NULL**	**NULL**	**NULL**

```
SELECT S.LAST, R.POINTS
FROM   STUDENTS S LEFT OUTER JOIN RESULTS R
  ON   (S.SID = R.SID)
```

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
104	Maria	Brown	NULL	NULL	NULL	**NULL**

Eeek!

More Joining

- Natural Join: Automatic condition from matching attributes
- Cross Join: Cartesian product (Big Data)
- Theta Join: More complex condition than **A=B**
- ...

Join Syntax

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

Explicit Join condition in **FROM**

```
SELECT R.ENO, R.POINTS
FROM   STUDENTS S, RESULTS R
WHERE  S.SID = R.SID
  AND  S.LAST = 'Smith'
```

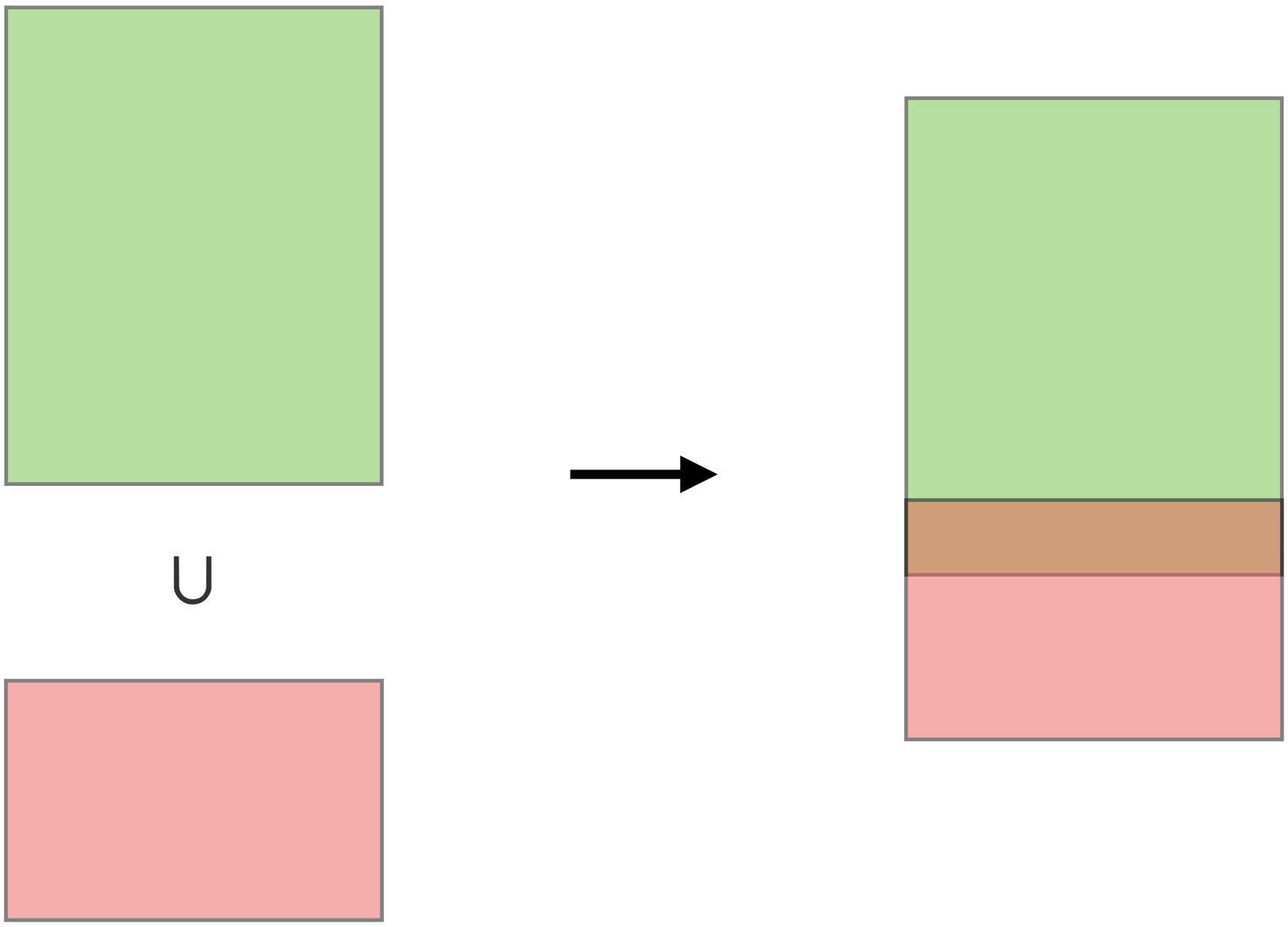
Implicit Join condition in **WHERE**

```
SELECT R.ENO, R.POINTS
FROM   STUDENTS S, RESULTS R
WHERE  S.LAST = 'Smith'
```

Forgot Join condition what happens?

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

SELECT SID, CAT
FROM RESULTS
WHERE SID = 103

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 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
1    FROM     STUDENTS S JOIN RESULTS R
      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!

```
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

Table-Producing Subqueries

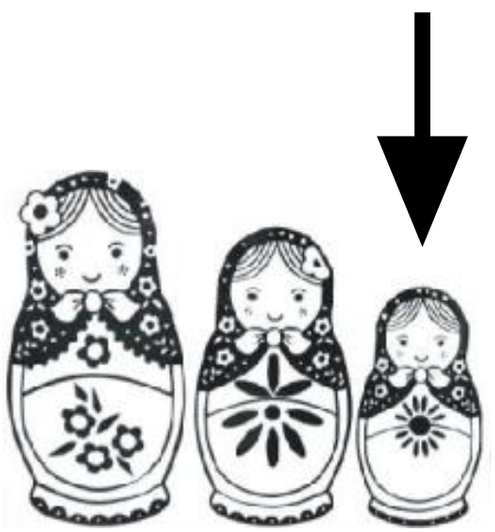
```
SELECT   ENO, AVG(POINTS) AS AV
FROM     STUDENTS JOIN RESULTS USING(SID)
WHERE    LAST = 'Smith'
GROUP BY ENO
ORDER BY AV LIMIT 10
```

```
SELECT ENO, AVG(POINTS) AS AV FROM (
  SELECT ENO, POINTS FROM (
    SELECT LAST, ENO, POINTS
    FROM STUDENTS S JOIN RESULTS R USING (SID))
  WHERE LAST='Smith')
GROUP BY ENO ORDER BY AV LIMIT 10
```

ProTip

```
SELECT ENO, AVG(POINTS) AS AV FROM (
  SELECT ENO, POINTS FROM (
    SELECT LAST, ENO, POINTS
    FROM STUDENTS S JOIN RESULTS R USING (SID))
  WHERE LAST='Smith')
GROUP BY ENO ORDER BY AV LIMIT 10
```

LAST	ENO	POINTS
Smith	1	10
Smith	2	8
Smith	1	12
Jones	1	9
Jones	2	9
Jones	1	9
Turner	1	5
Turner	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

More Window Functions

Aggregates

AVG
COUNT
MAX
MIN
SUM
...

Rank-Based

ROW_NUMBER
RANK
DENSE_RANK
PERCENT_RANK
CUME_DIST
NTILE
LAG
LEAD
FIRST_VALUE
LAST_VALUE
NTH_VALUE
...

TPCDS Q5

https://github.com/duckdb/duckdb/blob/master/extension/tpcds/dsdgen/queries/05.sql

There's more

- Common Table Expressions
- Data cube operators
- User-Defined Functions
- ...
- Random SQL Questions?

Google

sql on big data

All Images Videos News Shopping More Settings Tools

About 128.000.000 results (0,56 seconds)

SQL Server **big data** clusters provide flexibility in how you interact with your **big data**. You can query external **data** sources, store **big data** in HDFS managed by **SQL** Server, or query **data** from multiple external **data** sources through the cluster. You can then use the **data** for AI, machine learning, and other analysis tasks. Dec 5, 2018

SQL Server 2019 big data clusters - Microsoft Docs
https://docs.microsoft.com/en-us/sql/big-data-cluster/big-data-cluster-overview

About this result Feedback

People also ask

Is SQL good for big data?
What is Oracle Big Data SQL?
What database does big data use?
What is SQL on Hadoop?

Feedback

SQL Server 2019 big data clusters - Microsoft Docs
https://docs.microsoft.com/en-us/sql/big-data-cluster/big-data-cluster-overview
Dec 5, 2018 - **SQL** Server **big data** clusters provide flexibility in how you interact with your **big data**. You can query external **data** sources, store **big data** in HDFS managed by **SQL** Server, or query **data** from multiple external **data** sources through the cluster. You can then use the **data** for AI, machine learning, and other analysis tasks.

Big Data SQL | Database | Oracle
https://www.oracle.com/database/big-data-sql/
Extends Oracle **SQL** to Hadoop and NoSQL and the security of Oracle Database to all your **data**. It also includes a unique Smart Scan service that minimizes **data** movement and maximizes performance, by parsing and intelligently filtering **data** where it resides. ... Seamlessly query **data** ...

Nice SEO job, Oracle & Microsoft

Scope

- Many claim "SQL on Big Data"
  - Traditional Data Warehouse
  - MPP Databases
  - Hadoopified/Cloudified Legacy
  - Vendor lies
- Scope here: **No** dedicated data load necessary
  - Exclusively operates directly on files (CSV, Parquet, ...)
  - Open Source

Why SQL on Big Data?

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

F1: A Distributed SQL Database That Scales

Jeff Shute
Chad Whipkey
David Menestrina
Radek Vingralek
Eric Rollins
Stephan Ellner
Traian Stancescu
Bart Samwel
Mircea Oancea
John Cieslewicz
Himani Apte
Ben Handy
Kyle Littlefield
Ian Rae*

Google, Inc.
*University of Wisconsin-Madison

ABSTRACT

F1 is a distributed relational database system built at Google to support the AdWords business. F1 is a hybrid database that combines high availability, the scalability of NoSQL systems like Bigtable, and the consistency and usability of traditional SQL databases. F1 is built on Spanner, which provides synchronous cross-datacenter replication and strong consistency. Synchronous replication implies higher commit latency, but we mitigate that latency by using a hierarchical schema model with structured data types and through smart application design. F1 also includes a fully functional distributed SQL query engine and automatic change tracking and publishing.

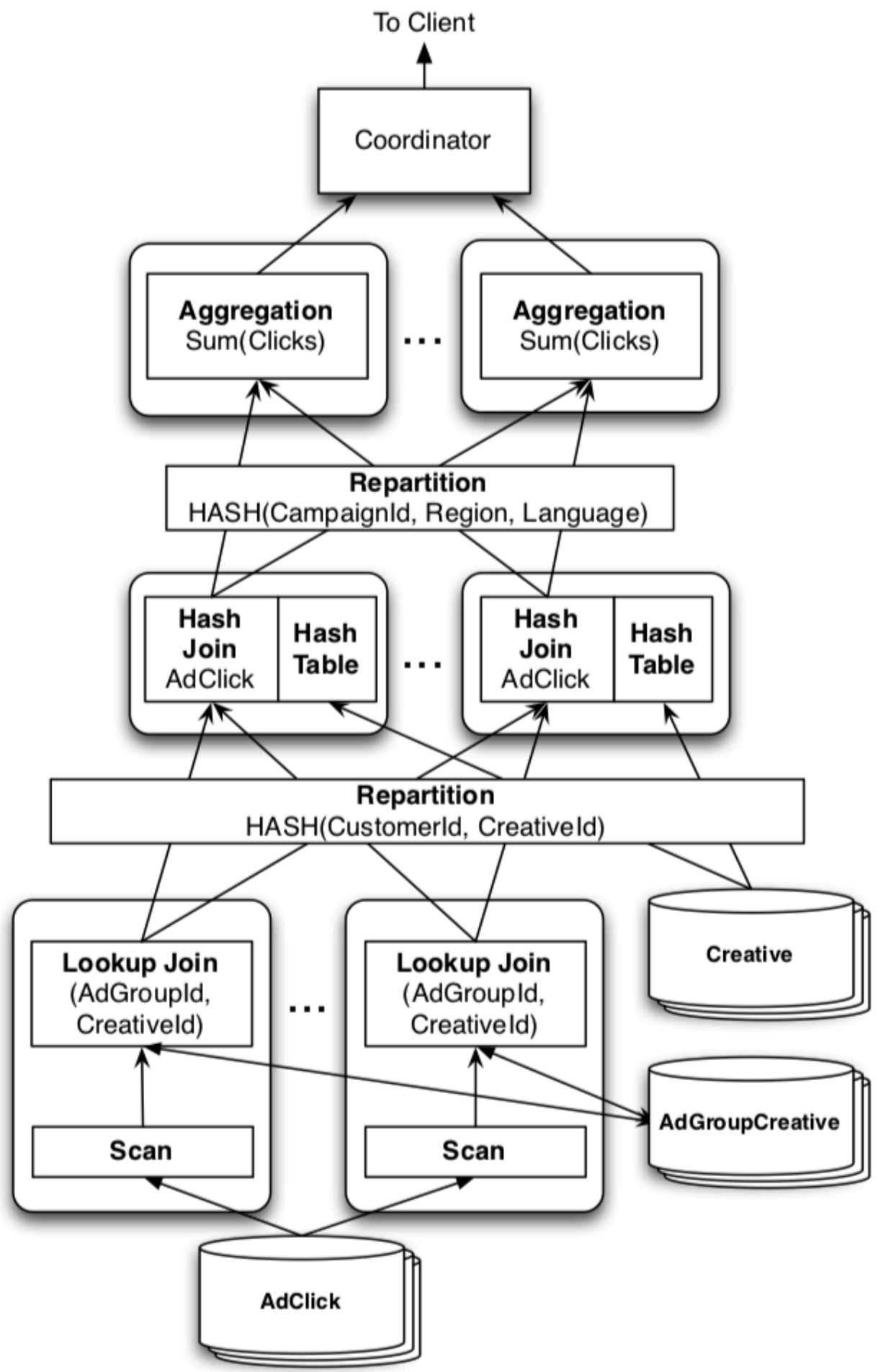
consistent and correct data.

Designing applications to cope with concurrency anomalies in their data is very error-prone, time-consuming, and ultimately not worth the performance gains.

4. **Usability**: The system must provide full SQL query support and other functionality users expect from a SQL database. Features like indexes and ad hoc query are not just nice to have, but absolute requirements for our business.

Recent publications have suggested that these design goals are mutually exclusive [5, 11, 23]. A key contribution of this

```
SELECT agcr.CampaignId, click.Region,
  cr.Language, SUM(click.Clicks)
FROM AdClick click
  JOIN AdGroupCreative agcr
    USING (AdGroupId, CreativeId)
  JOIN Creative cr
    USING (CustomerId, CreativeId)
WHERE click.Date = '2013-03-23'
GROUP BY agcr.CampaignId,
  click.Region, cr.Language
```



Hive

- 2010, Facebook
- Runs on top of Hadoop
  - SQL → MapReduce jobs
- Innovation: Schema
  - **HCatalog**, de-facto standard
- Downside: slooow

CADE METZ BUSINESS 10.24.12 09:00 AM

MAN BUSTS OUT OF GOOGLE, REBUILDS TOP-SECRET QUERY MACHINE

Cloudera just unleashed Impala, a Google-inspired tool for analyzing massive amounts of data. IMAGE: FLICKR/EWANR

YOU CAN THINK of Google as the research lab for the internet.

Every so often, the company releases a research paper describing one of the sweeping software platforms that help drive its online empire, and a few years later, this paper will spawn an open source software project that seeks to share Google's creation with the rest of the world.

https://www.wired.com/2012/10/cloudera-impala-hadoop/

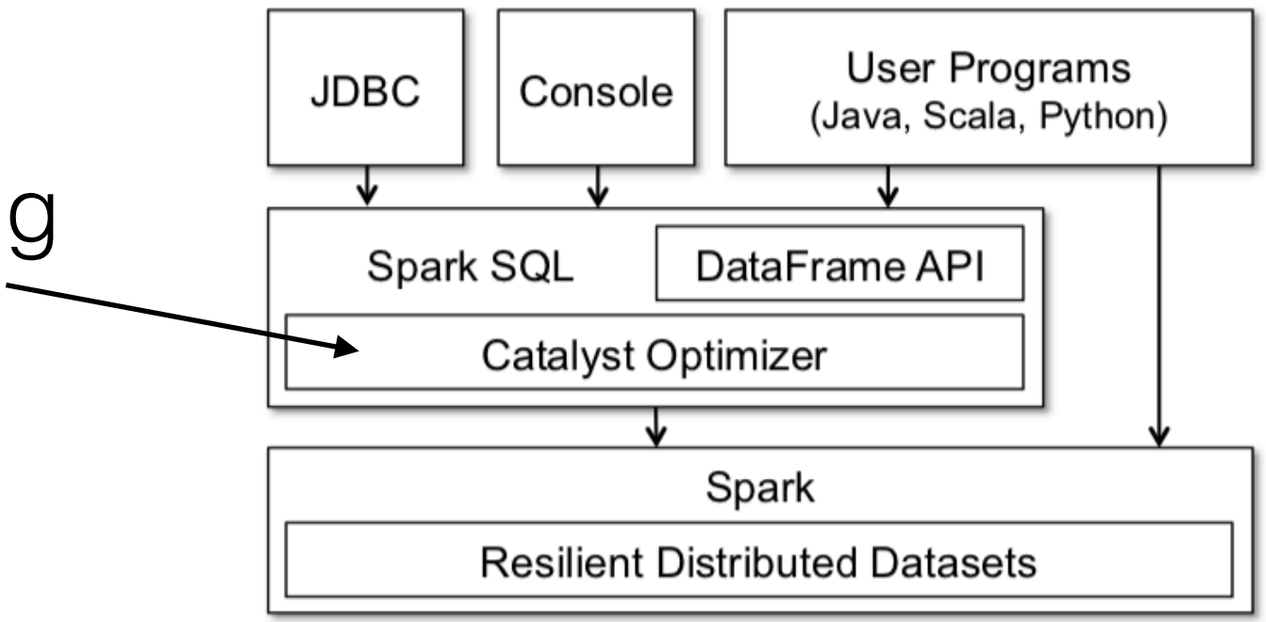
Impala/Presto

- 2013, Cloudera/Facebook
- F1 Clone, C++,
- Innovation: "Interactive"
  - Bypass Hadoop scheduling → Low(er)-Latency
  - Similar to Spark
- JIT, compression, columnar, ...
- Athena = Hosted Presto

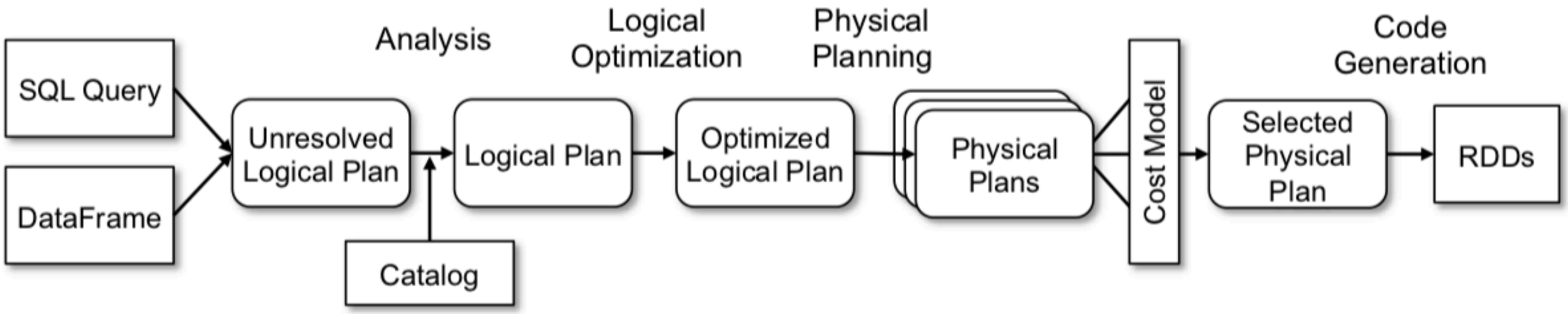
Spark SQL

- 2015, Databricks/UC Berkeley
- SQL → Spark jobs
  - RDDs etc.
- Uses HCatalog

Pattern Matching



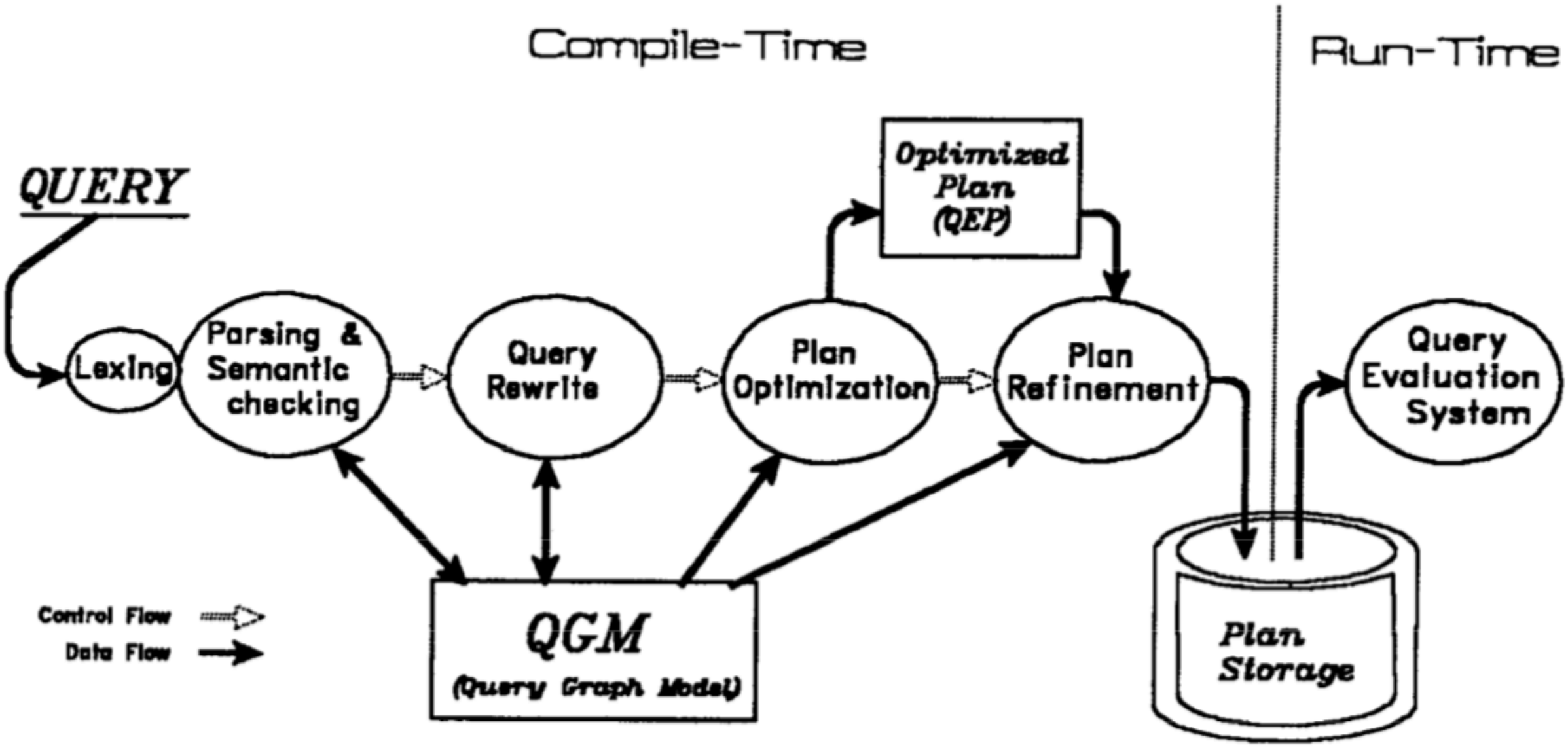
Spark Optimizer



https://github.com/apache/spark/blob/master/sql/catalyst/src/main/scala/org/apache/spark/sql/catalyst/optimizer/expressions.scala

Extensible Query Processing in Starburst

Laura M. Haas, J.C. Freytag[1], G.M. Lohman, and H. Pirahesh
IBM Almaden Research Center, San Jose, CA 95120



Big Data SQL Observations

- Overall enabler: Shared Schema (HCatalog)
- SQL standard support varying
- Good support for User-Defined Functions
- Optimizers generally limited

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 :/

Partition by **CAT**

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

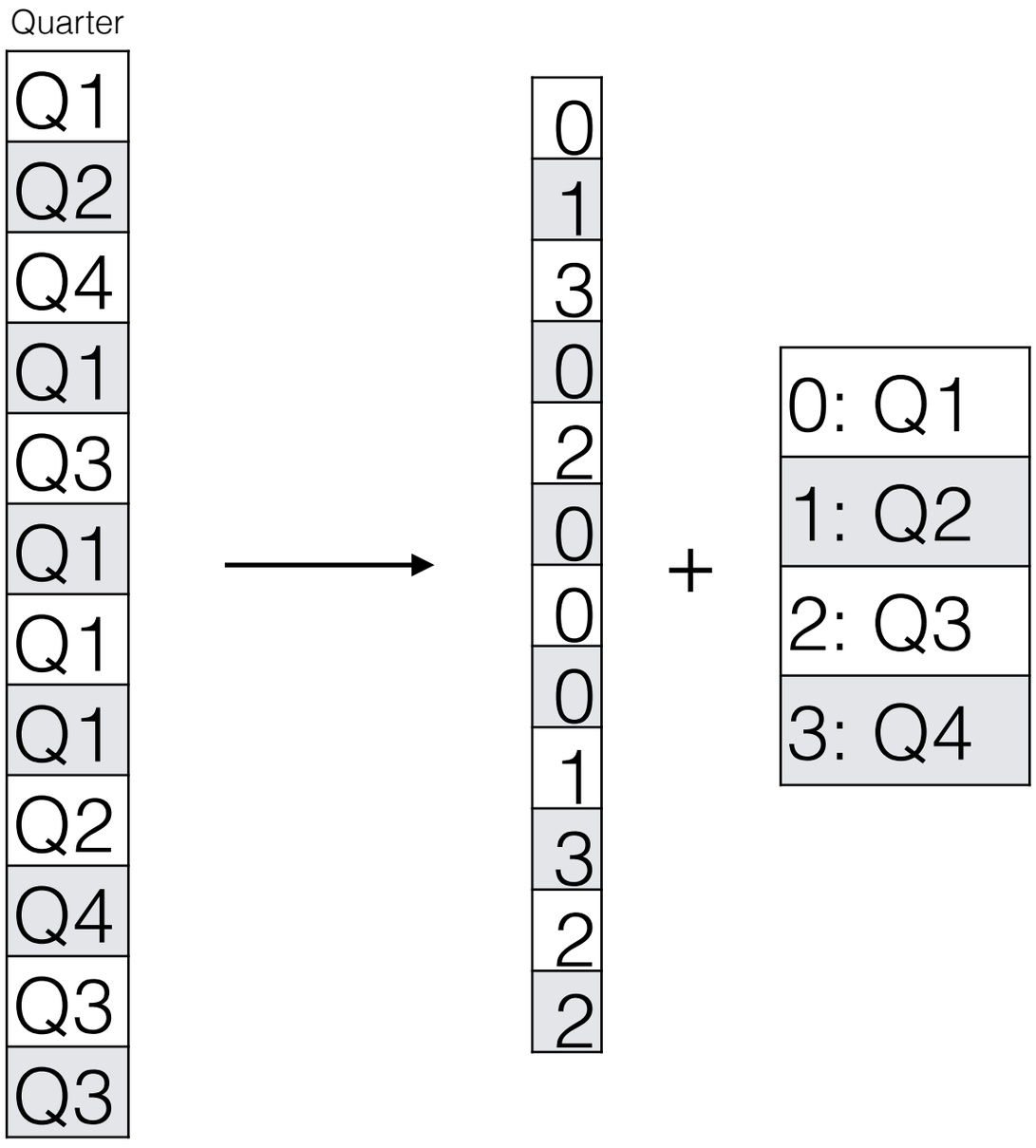
Data Layout & Compression

- 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

Example: Dictionary Compression

CREATE TABLE A
(QUARTER STRING)

SELECT * FROM A
WHERE QUARTER='Q2'



2 Bit/Value

Integer vs. String comparison

Lightweight Indexing

- Data often "naturally" ordered, e.g. by date
- Data often correlated
  - orderdate < paydate < shipdate
  - date > marketing campaign
  - ...
- Create zone maps
  - Keep min/max for every column
  - Allows skipping entire chunks of data

Example: Zonemap

Accounts

KEY	ACCTNO	BALANCE	
0	19	269,38	Zone 1
1	38	914,11	
2	72	346,61	
3	156	266,55	
4	153	850,90	Zone 2
5	282	521,60	
6	389	647,38	
7	134	119,40	
8	332	526,08	Zone 3
9	203	497,19	
10	533	22,03	
11	592	140,67	
12	808	383,69	Zone 4
13	896	899,41	

Accounts-Zonemap

COL	ZONE	MIN	MAX
KEY	1	0	3
KEY	2	4	7
KEY	3	8	11
KEY	4	12	13
ACCTNO	1	19	156
ACCTNO	2	153	389
ACCTNO	3	332	592
ACCTNO	4	808	896
BALANCE	1	266,55	914,11
BALANCE	2	119,40	850,90
BALANCE	3	22,03	536,08
BALANCE	4	383,69	889,41

Accounts

KEY	ACCTNO	BALANCE	
0	19	269,38	Zone 1
1	38	914,11	
2	72	346,61	
3	156	266,55	
4	153	850,90	Zone 2
5	282	521,60	
6	389	647,38	
7	134	119,40	
8	332	526,08	Zone 3
9	203	497,19	
10	533	22,03	
11	592	140,67	
12	808	383,69	Zone 4
13	896	899,41	

Accounts-Zonemap

COL	ZONE	MIN	MAX
KEY	1	0	3
KEY	2	4	7
KEY	3	8	11
KEY	4	12	13
ACCTNO	1	19	156
ACCTNO	2	153	389
ACCTNO	3	332	592
ACCTNO	4	808	896
BALANCE	1	266,55	914,11
BALANCE	2	119,40	850,90
BALANCE	3	22,03	536,08
BALANCE	4	383,69	889,41

SELECT * FROM A
WHERE ACCTNO BETWEEN 150 AND 200

Accounts

KEY	ACCTNO	BALANCE
0	19	269,38
1	38	914,11
2	72	346,61
3	156	266,55
4	153	850,90
5	282	521,60
6	389	647,38
7	134	119,40
8	332	526,08
9	203	497,19
10	533	22,03
11	592	140,67
12	808	383,69
13	896	899,41

Zone 1

Zone 2

Zone 3

Zone 4

Accounts-Zonemap

COL	ZONE	MIN	MAX
KEY	1	0	3
KEY	2	4	7
KEY	3	8	11
KEY	4	12	13
ACCTNO	1	19	156
ACCTNO	2	153	389
ACCTNO	3	332	592
ACCTNO	4	808	896
BALANCE	1	266,55	914,11
BALANCE	2	119,40	850,90
BALANCE	3	22,03	536,08
BALANCE	4	383,69	889,41

SELECT * FROM A
WHERE ACCTNO BETWEEN 150 AND 200

Accounts

KEY	ACCTNO	BALANCE
0	19	269,38
1	38	914,11
2	72	346,61
3	156	266,55
4	153	850,90
5	282	521,60
6	389	647,38
7	134	119,40
8	332	526,08
9	203	497,19
10	533	22,03
11	592	140,67
12	808	383,69
13	896	899,41

Zone 1

Zone 2

Zone 3

Zone 4

Accounts-Zonemap

COL	ZONE	MIN	MAX
KEY	1	0	3
KEY	2	4	7
KEY	3	8	11
KEY	4	12	13
ACCTNO	1	19	156
ACCTNO	2	153	389
ACCTNO	3	332	592
ACCTNO	4	808	896
BALANCE	1	266,55	914,11
BALANCE	2	119,40	850,90
BALANCE	3	22,03	536,08
BALANCE	4	383,69	889,41

SELECT * FROM A
WHERE ACCTNO BETWEEN 150 AND 200

Practicum 3

Topics Today

1. Introduction
2. ETL on Big Data

   Practicum
3. SQL on Big Data

   Practicum
4. **Machine Learning on Big Data**

Single Model

- **One** model trained on a large dataset
  - e.g. NN, ALS, ...
- Problem: **Hard** to partition!
  - Each step depends on all previous steps
  - Needs copying state around constantly :(
  - Cannot just train NNs on parts of data
    - Or can you?

ACCELERATING RECURRENT NEURAL NETWORK TRAINING VIA TWO STAGE CLASSES AND PARALLELIZATION

Zhiheng Huang, Geoffrey Zweig, Michael Levit, Benoit Dumoulin, Barlas Oguz and Shawn Chang*

Speech at Microsoft, Mountain View, CA
*Microsoft Research, Redmond, WA
{zhihuang,gzweig,Michael.Levit,bedumoul,barlaso,Shawn.Chang}@microsoft.com

- Idea 1:
  - Copy model to nodes
  - Train on subset of data
  - Copy model back to master
  - Compute average & repeat
- Paper finds improved recognition performance (?)
- Quality depends on partitioning!

TensorFlow: A system for large-scale machine learning

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng

Google Brain

- Idea 2:
  - Split up learning into small operators
    - e.g. matrix multiplication, averaging etc.
  - Create schedule on available resources but not split up data
  - Downside: Data movement!

TensorFlow: Neural Net

First **define** computational graph

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.placeholder(tf.float32, shape=(D, H))
w2 = tf.placeholder(tf.float32, shape=(H, D))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))

grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])
```

Then **run** the graph many times

```
with tf.Session() as sess:
    values = {x: np.random.randn(N, D),
              w1: np.random.randn(D, H),
              w2: np.random.randn(H, D),
              y: np.random.randn(N, D),}
    out = sess.run([loss, grad_w1, grad_w2],
                   feed_dict=values)
    loss_val, grad_w1_val, grad_w2_val = out
```

http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture08.pdf

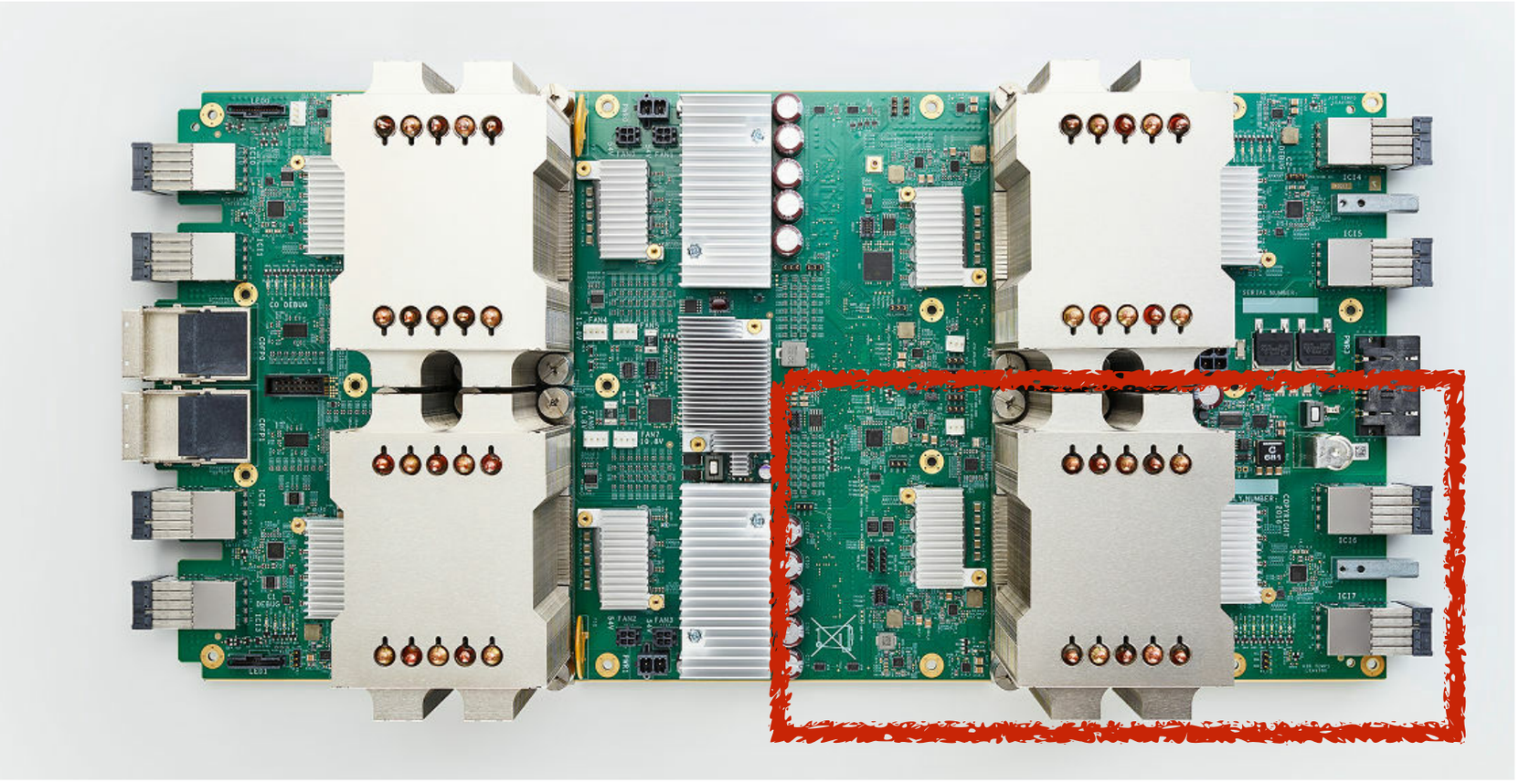
Single Model

- Idea 3:
  - Use special hardware! (GPU/TPU)
  - High memory bandwidth
  - Massive parallelism in operations
  - Good match with e.g. NN training
  - Support in libraries (e.g. Caffe, TensorFlow)

Graphics Processing Unit (GPU)

GeForce GTX 1080
320 GB/sec
10 TFLOPS
8GB RAM
592 €

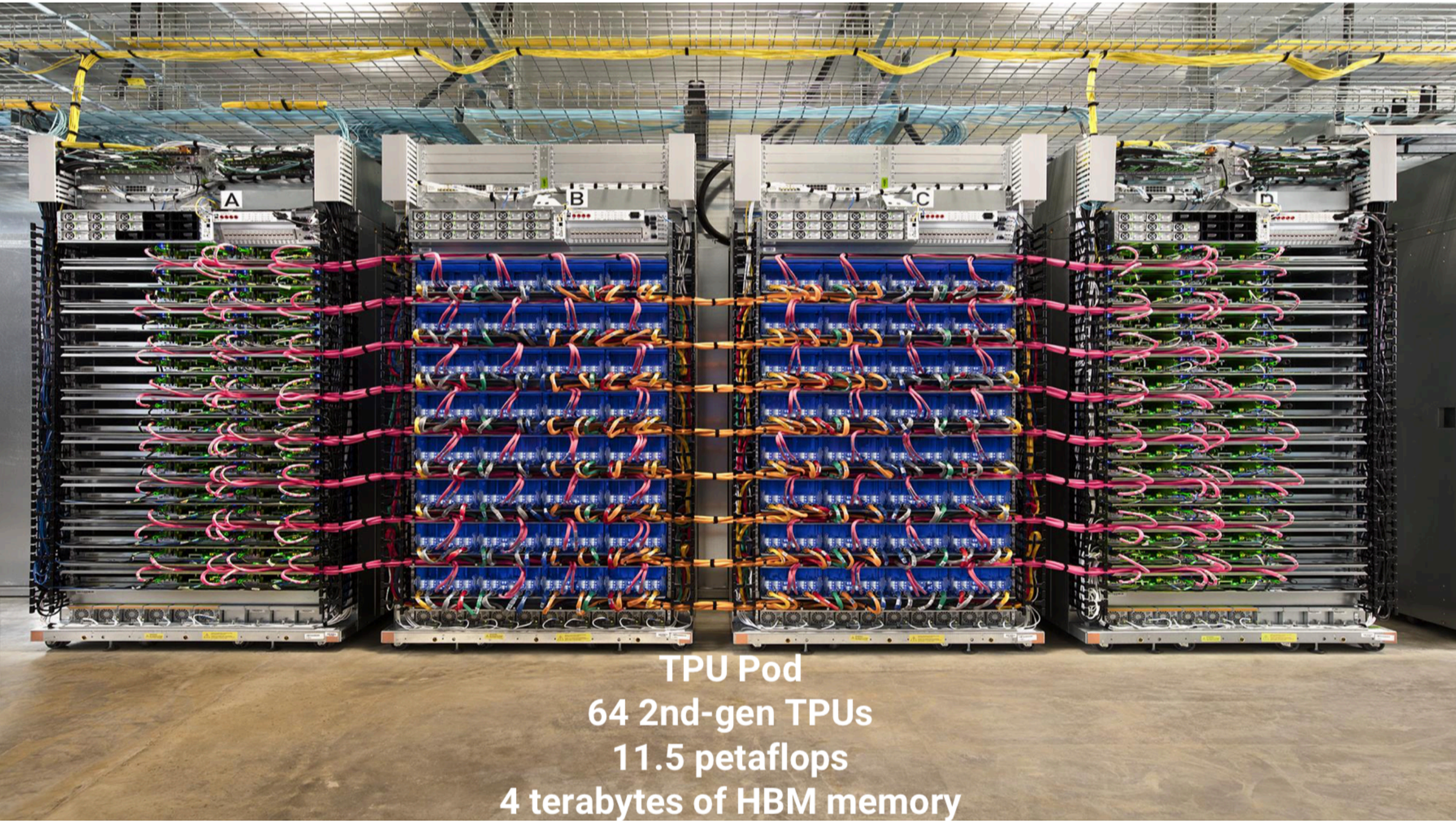
Tensor Processing Unit (TPU)



GeForce GTX 1080
320 GB/sec
8GB RAM
10 TFLOPS

Google 2Gen TPU
600 GB/sec
16GB RAM
45 TFLOPS

http://learningsys.org/nips17/assets/slides/**dean**-nips17.pdf


A
B
C
D
TPU Pod
64 2nd-gen TPUs
11.5 petaflops
4 terabytes of HBM memory

Jeff Dean:

"For large models, model parallelism is important. But getting good performance given multiple computing devices is **non-trivial** and non-obvious"

MLlib

- Idea 4:
  - Use Spark to implement **few** ML algorithms
  - Algorithms with ~~fairly efficient~~ barely working distributed version

Lies!

- MLlib is also comparable to or even better than other libraries specialized in large-scale machine learning.

https://stanford.edu/~rezab/sparkworkshop/slides/xiangrui.pdf

Spark MLlib

- Implemented Algorithms
  - Classification: logistic regression, linear support vector machine (SVM), naive Bayes
  - Regression: generalized linear regression (GLM)
  - Collaborative Filtering: alternating least squares (ALS)
  - Clustering: k-means
  - Decomposition: singular value decomposition (SVD), principal component analysis (PCA)
- General tip: Avoid!

Single Model & Big Data

- Instead of running distributed ML, spend time on
  - Data cleaning / ETL
  - Feature engineering
  - Sampling

Simpler Model?

- Use single machine tools, perhaps involving GPU/TPU
  - TensorFlow best bet in 2022
- Read & Watch
  - https://github.com/szilard/benchm-ml
  - https://www.youtube.com/watch?v=8wyOwUNw7D8

Many (Independent) Models

- Train multiple models on subsets of data
  - E.g. by country, language, customer, ...
- Embarrassingly parallel, so can efficiently distribute
  - Great for expensive models, e.g. NN
  - **map()** to collect all data for subset
  - **reduce()** phase to train separate models
- Models become data
  - Opaque blob or parameter columns

Parameter Tuning

- Performance of ML models heavily depends on parameters
  - Esp. NN
- Can have many parameters, unclear of their interactions
  - Giant parameter space
    - Can be non-continuous too, so no gradient descent :/
- Insight: Individual models are independent.
- Distribute data, partition parameter space
- **Parallelize** training & evaluation!

Topics Today

1. Introduction
2. ETL on Big Data

   Practicum
3. SQL on Big Data

   Practicum
4. Machine Learning on Big Data

   Party!