History of Database Systems

JIANNAN WANG

SIMON FRASER UNIVERSITY

Database Systems in a Half Century

(1960s – 2020s)

When	What
Early 1960 – Early 1970	The Navigational Database Empire
Mid 1970 – Mid 1980	The Database World War I
Mid 1980 – Early 2000	The Relational Database Empire
Mid 2000 – Now	The Database World War II

References.

- https://en.wikipedia.org/wiki/Database#History
- What Goes Around Comes Around (Michael Stonebraker, Joe Hellerstein)
- 40 Years VLDB Panel

The Navigational Database Empire

(Early1960 – Early 1970)

Data Model

1. How to organize data
2. How to access data

Navigational Data Model

1. Organize data into a multi-dimensional space (i.e., A space of records)
2. Access data by following pointers between records

Inventor: Charles Bachman

1. The 1973 ACM Turing Award
2. Turing Lecture: "The Programmer As Navigator"

The Navigational Database Empire

(Early1960 – Early 1970)

Representative Navigational Database Systems

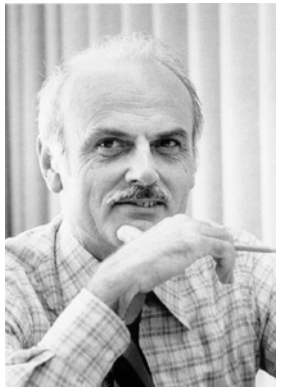
- Integrated Data Store (IDS), 1964, GE
- Information Management System (IMS), 1966, IBM
- Integrated Database Management System (IDMS), 1973, Goodrich

CODASYL

- Short for "Conference/Committee on Data Systems Languages"
- Define navigational data model as standard database interface (1969)

The Birth of Relational Model

Ted Codd



- Born in 1923
- PHD in 1965
- "A Relational Model of Data for Large Shared Data Banks" in 1970

Relational Model

- Organize data into a collection of relations
- Access data by a declarative language (i.e., tell me what you want, not how to find it)

Data Independence

The Database World War I: Background

One Slide (Navigational Model)

- Led by Charles Bachman (1973 ACM Turing Award)
- Has built mature systems
- Dominated the database market

The other Slide (Relational Model)

- Led by Ted Codd (mathematical programmer, IBM)
- A theoretical paper with no system built
- Little support from IBM

The Database World War I:
Three Big Campaigns

1. Which data model is better in theory?
   (Mid 1970)

2. Which data model is better in practice?
   (Late 1970 – Early 1980)

3. Which data model is better in business?
   (Early 1980 – Mid 1980)

The "Theory" Campaign

A debate at ACM SIGFIDET (precursor of SIGMOD) 1974

Navigational model is bad

- Data Organization: So complex
- Data Access: No declarative language

Relational model is bad

- Data Organization: A special case of navigational model
- Data Access: No system proof that declarative language is viable

The "Practice" Campaign

The Big Question

- Can a relational database system perform as good as a navigational system?

System prototypes

- Ingres at UC Berkeley (early and pioneering)
- System R at IBM (arguably got more stuff "right")

The System R Team

- Query Optimization (Patricia P. Griffiths et al.)
- SQL (Donald D. Chamberlin et al.)
- Transaction (Jim Gray et al.)

The “Business” Campaign

Commercialization of Relational Database Systems

Not as easy as we thought

Three reasons (required) that led relational database systems to won

- The minicomputer revolution (1977)
- Competing products (e.g. IDMS) could not be ported to the minicomputer
- Relational front end was not added to navigational database systems

What Can We Learn?

Lesson 1.

The winning of theory ≠ The winning of practice

Lesson 2.

The winning of practice ≠ The winning of business

Lesson 3.

Everyone can get a chance to win

The Relational Database Empire
(Mid1980 – Early 2000)

Parallel and distributed DBs (1980 – 1990)

- SystemR*, Distributed Ingres, Gamma, etc.

Objected-oriented DB (1980 – 1990)

- Objects: Data/Code Integration
- Extensibility: User-defined functions, User-defined data types

MySQL and PostgresSQL (1990s)

- Widely used open-source relational DB systems

Database Systems in a Half Century

(1960s – 2010s)

When	What
Early 1960 – Early 1970	The Navigational Database Empire
Mid 1970 – Mid 1980	The Database World War I
Mid 1980 – Early 2000	The Relational Database Empire
Mid 2000 – Now	The Database World War II

References.

- https://en.wikipedia.org/wiki/Database#History
- What Goes Around Comes Around (Michael Stonebraker, Joe Hellerstein)
- 40 Years VLDB Panel

The Database World War II: Background

Internet Boom (Early 2000)

- Larger data volume that cannot be fit in a single machine
- Faster data updates that cannot be handled by a single machine

Commercial distributed database systems are expensive ☹

Open-source database systems do not support distributed computing well ☹

OLTP

OnLine Transaction Processing

Workload

High-frequent Updates + Small Queries

OLAP

OnLine Analytical Processing

Workload

Low-frequent Updates + Big Queries

The Database World War II:
Two Big Campaigns

1. Which is better for (distributed) OLTP:
   NoSQL vs. Relational DBMS?

2. Which is better for (distributed) OLAP:
   MapReduce vs. Relational DBMS?

The Database World War II:
Two Big Campaigns

1. Which is better for OLTP:
   NoSQL vs. Relational DBMS?

2. Which is better for OLAP:
   MapReduce vs. Relational DBMS?

What Happened To OLTP?

OldSQL (1970 – Now)

NoSQL (2000 – Now)

NewSQL (2010 – Now)

OldSQL (1970 – Now)

Traditional SQL vendors

. . .

Still very big market!!!

Limitation 1: Not Scalable

Limitation 2: Pre-defined Schema

The advent of Web 2.0

Read-only Web → Read-write Web

Highly Scalable

- Scale to1,000,000 users and 1000 servers

Highly Available

- Available 24 hours a day, 7 days a week

Highly Flexible

- Flexible schema and flexible data types

NoSQL Pioneers

Memcached [Fitzpatrick 2004]

- In-memory indexes can be highly scalable

BigTable [Chang et al. 2006]

- Persistent record storage could be scaled to thousands of nodes

Dynamo [DeCandia et al. 2007]

- Eventual consistency allows for higher availability and scalability

NoSQL Categories

NoSQL	Data Model	Example Systems
Key-value Stores	Hash	DynamoDB, Riak, Redis, Membase
Document Stores	Json	SimpleDB, CouchBase, MongoDB
Wide-column Stores	Big Table	Hbase, Cassandra, HyperTable
Graph Database	Graph	Neo4J, InfoGrid, GraphBase

NoSQL Limitations

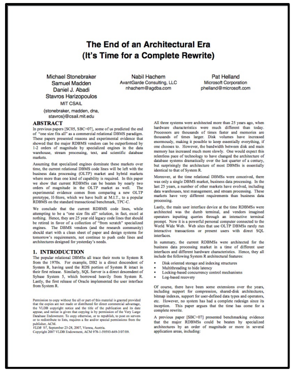
Low-level Language
- Simple read/write database operators

Weak Consistency
- Eventual Consistency

Lack of Standardization
- 100+ NoSQL systems

NewSQL

Strong Consistency + High Scalability


The End of an Architectural Era
(It's Time for a Complete Rewrite)

Michael Stonebraker
Samuel Madden
Daniel J. Abadi
Stavros Harizopoulos
MIT CSAIL

Nabil Hachem
AvantGarde Consulting, LLC

Pat Helland
Microsoft Corporation

The end of an architectural era:(it's time for a complete rewrite)

M Stonebraker, S Madden, DJ Abadi… - Proceedings of the 33rd …, 2007 - dl.acm.org

Abstract In previous papers [SC05, SBC+ 07], some of us predicted the end of" one size fits all" as a commercial relational DBMS paradigm. These papers presented reasons and experimental evidence that showed that the major RDBMS vendors can be outperformed ...

Cited by 580 Related articles All 55 versions Cite Save

90% query time spent on overhead

NewSQL Market

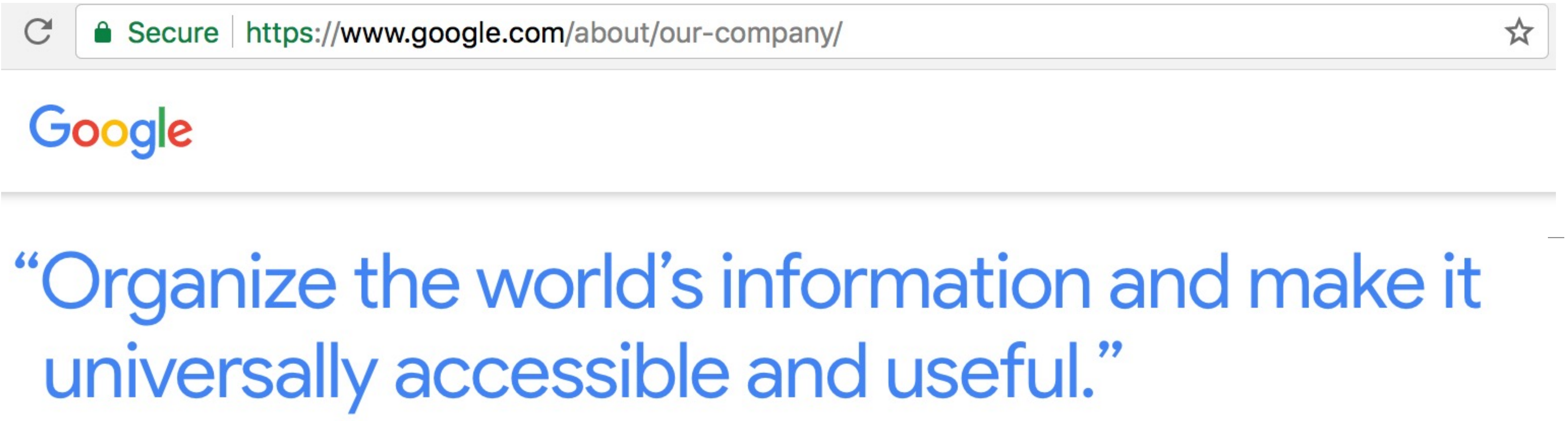
Limitations

- Scalable but not highly scalable
- Available but not highly available
- Flexible but not highly flexible

The Database World War II:
Two Big Campaigns

1. Which is better for OLTP:
   NoSQL vs. Relational DBMS?

2. Which is better for OLAP:
   MapReduce vs. Relational DBMS?



1 PB = 100GB *10,000 Machines

How to store 1PB using 10,000 machines? → GFS (HDFS)

How to process 1PB using 10,000 machines? → MapReduce

Why MapReduce?

1. Fault Tolerant

(Your program will be OK when failures happen)

# of machines	Failure Probability
1	0.1%
10	0.9%
100	9.5%
1000	63.2%
10,000	??? 99.9%

Cost for 5 nodes	Failure Probability
$100/day	0.1%
$1/day	10%

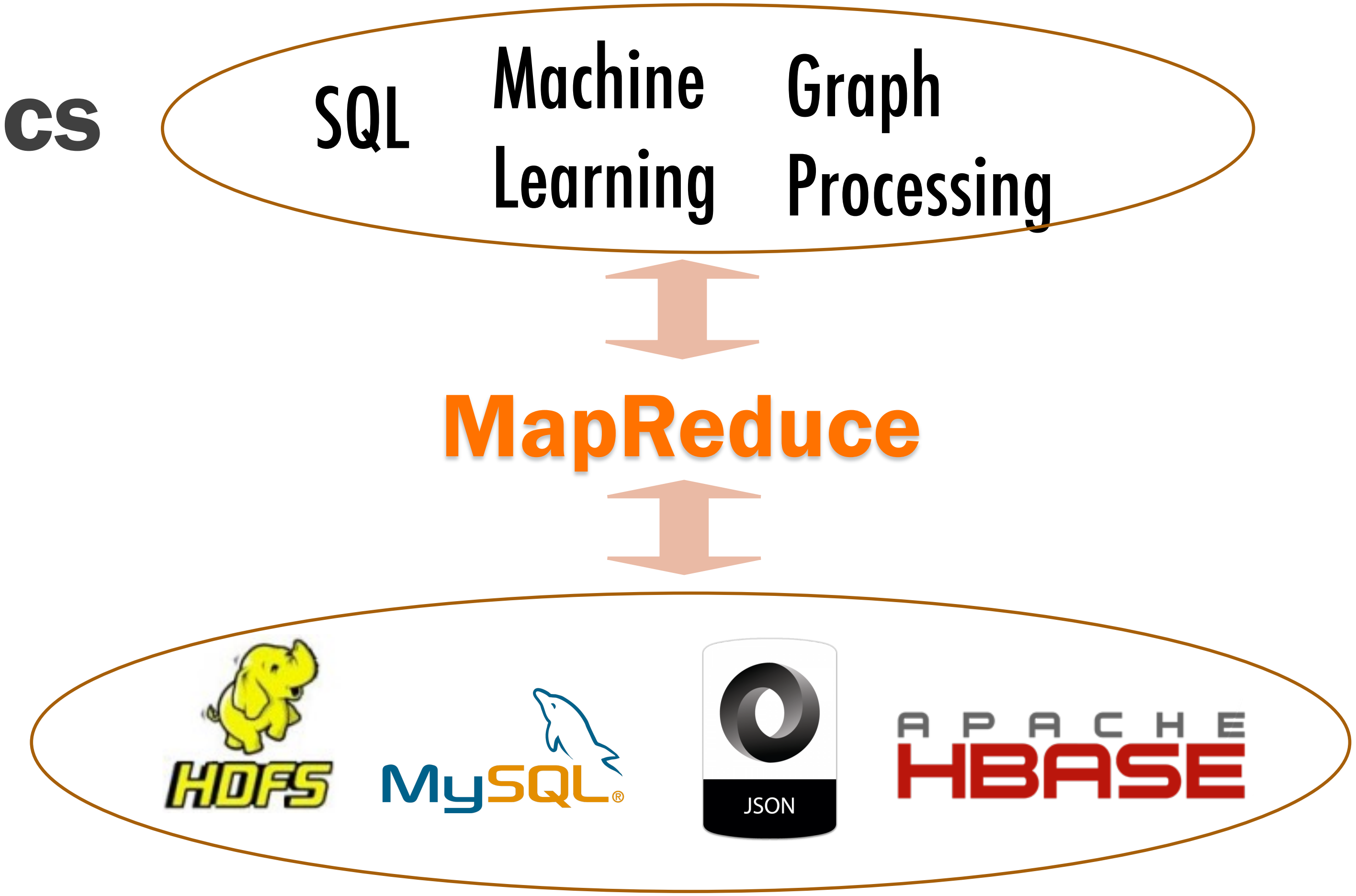
Reserved Instance → $100/day

Spot Instance → $1/day

Why MapReduce?

2. Complex Analytics

3. Heterogeneous Storage Systems



The Great MapReduce Debate

(2008-2010)

http://tiny.cc/mapreduce-debate

MapReduce vs. SQL

Map (k, v) = SELECT Map(v)
FROM Table

Reduce (k, v_list) = SELECT Reduce(v)
FROM Table
GROUP BY k

MapReduce: A Major Step Backwards

1. MapReduce is a step backwards in database access
2. MapReduce is a poor implementation
3. MapReduce is not novel
4. MapReduce is missing features
5. MapReduce is incompatible with the DBMS tools

Dewitt, D. and Stonebraker, M. *MapReduce: A Major Step Backwards blogpost*; January 17, 2008

Comments From The Other Side

VS.

MapReduce is a program model rather than a **database** system

COMMUNICATIONS OF THE ACM

CACM.ACM.ORG 01/2010 VOL.53 NO.01

From Stonebraker et al.

From Dean and Ghemawat

MapReduce complements DBMSs since databases are not designed for extract-transform-load tasks, a MapReduce specialty.

BY MICHAEL STONEBRAKER, DANIEL ABADI, DAVID J. DEWITT, SAM MADDEN, ERIK PAULSON, ANDREW PAVLO, AND ALEXANDER RASIN

MapReduce and Parallel DBMSs: Friends or Foes?

MapReduce advantages over parallel databases include storage-system independence and fine-grain fault tolerance for large jobs.

BY JEFFREY DEAN AND SANJAY GHEMAWAT

MapReduce: A Flexible Data Processing Tool

What They Agree On?

Advantages of MapReduce:

1. Fault Tolerant
2. Complex Analytics
3. Heterogeneous Storage Systems
4. No Data Loading Requirement

Both should Learn from Each Other

Who won the debate?

Nobody is writing MapReduce code right now.

Many new systems (e.g., Spark, HIVE) were built on MapReduce

Summary

Why Relational Database? (Mid 1970 – Now)

Why NoSQL? (Mid 2000 – Now)

Why MapReduce? (Mid 2000 – Now)