Dirk Habich

Scalable Data Management (SDM)

Introduction

TECHNISCHE UNIVERSITÄT DRESDEN

Dresden Database Research Group

What this course is about

Dresden Database Research Group

Learn how to design scalable and efficient data processing (cloud-native) systems

- Understand the demands of novel (data) workloads and the economies and challenges at scale.
- Get to know the internals of modern data centers and emerging technologies and trends.
- Learn the fundamental principles for building scalable system software for data processing.

Build a modern data processing system

- Work across multiple layers of the stack: storage, synchronization, caching, compute, etc.
- Tailor the system for given workload requirements: data management, ML, etc.
- Think in terms of performance, scalability, fault tolerance, elasticity, high availability, cost, privacy, etc.
- Use modern (cloud) constructs like containers or serverless functions.

Apply the knowledge

- Within the exercises.

Focus:
Relational Database Systems

DBMS Sandwich

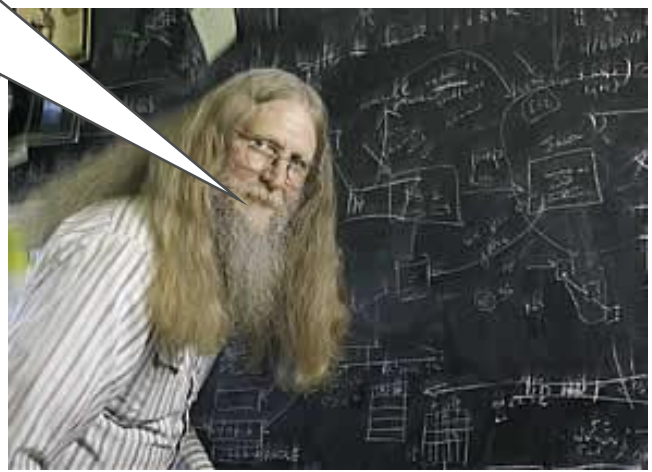
Applications

(Relational) Database Management System (DBMS)

Hardware

Why Relational Databases???

Relational databases are the foundation of western civilization.



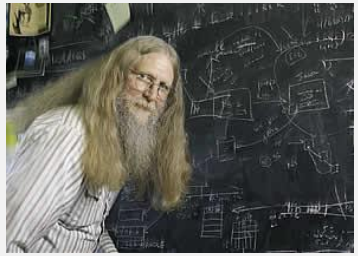
Bruce Lindsay,
IBM Fellow @ IBM Almaden Research Center

Why Relational Databases???

Dresden Database Research Group

"Relational Databases are the foundation of western civilization."

Bruce Lindsay,
IBM Fellow @ IBM Almaden Research Center



Relational Database Management Systems (DBMS) are

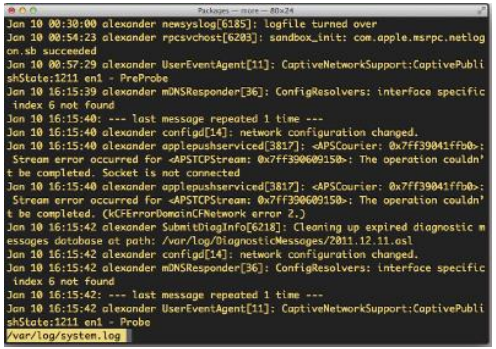
- Established state-of-the-art and well researched
- Universally applicable due to its flexibility
- Available in all sizes and at all prices
- Supported by many tools

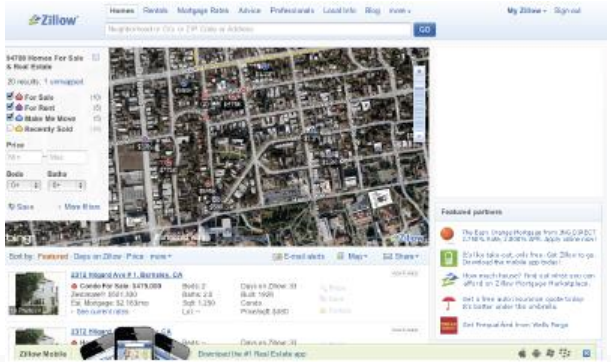
Common Concepts

- Data Modeling, Recovery, Transaction processing, Query optimization

Spot the database!!!

Dresden Database Research Group





Spot the database!!! (2)

Dresden Database Research Group

http://newmail-ng.com/facebook-records-1-3bn-users-daily-says-zuckerberg/

Facebook uses MySQL to store posts

Twitter uses MySQL for tweets and users

LinkedIn uses Oracle Database

Youtube uses MySQL

Wikipedia uses MySQL

WordPress uses MySQL to manage components of a website (pages, links, menus, etc.)

Things are changing dramatically:

(1) Evolving Data-Driven Applications

Data is the New Oil of the 21st Century

Dresden Database Research Group

[The Economist, 2010]

[IBM, 2018]

[Wired, 2014]

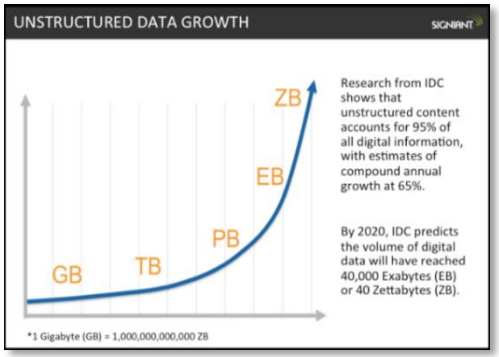
New Oil of the Digital Economy

Evolving Data-Driven Applications

Evolving
Data-Driven Applications

Data Management System (DMS)

Data Volume



HYBRID

Transactional & Analytical Processing (HTAP)

Throughput/Latency

Information at the fingertips

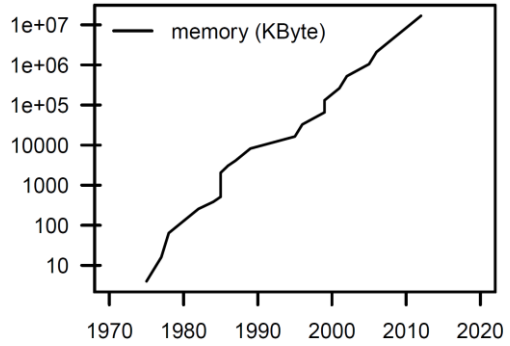
Things are changing dramatically:

(2) Evolving Hardware Landscape

Evolving Hardware

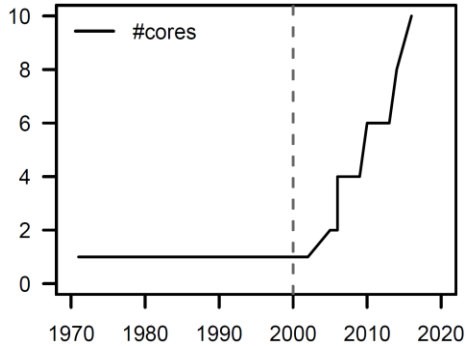
Dresden Database Research Group

Increasing Main Memory Capacity



Increasing Heterogeneity

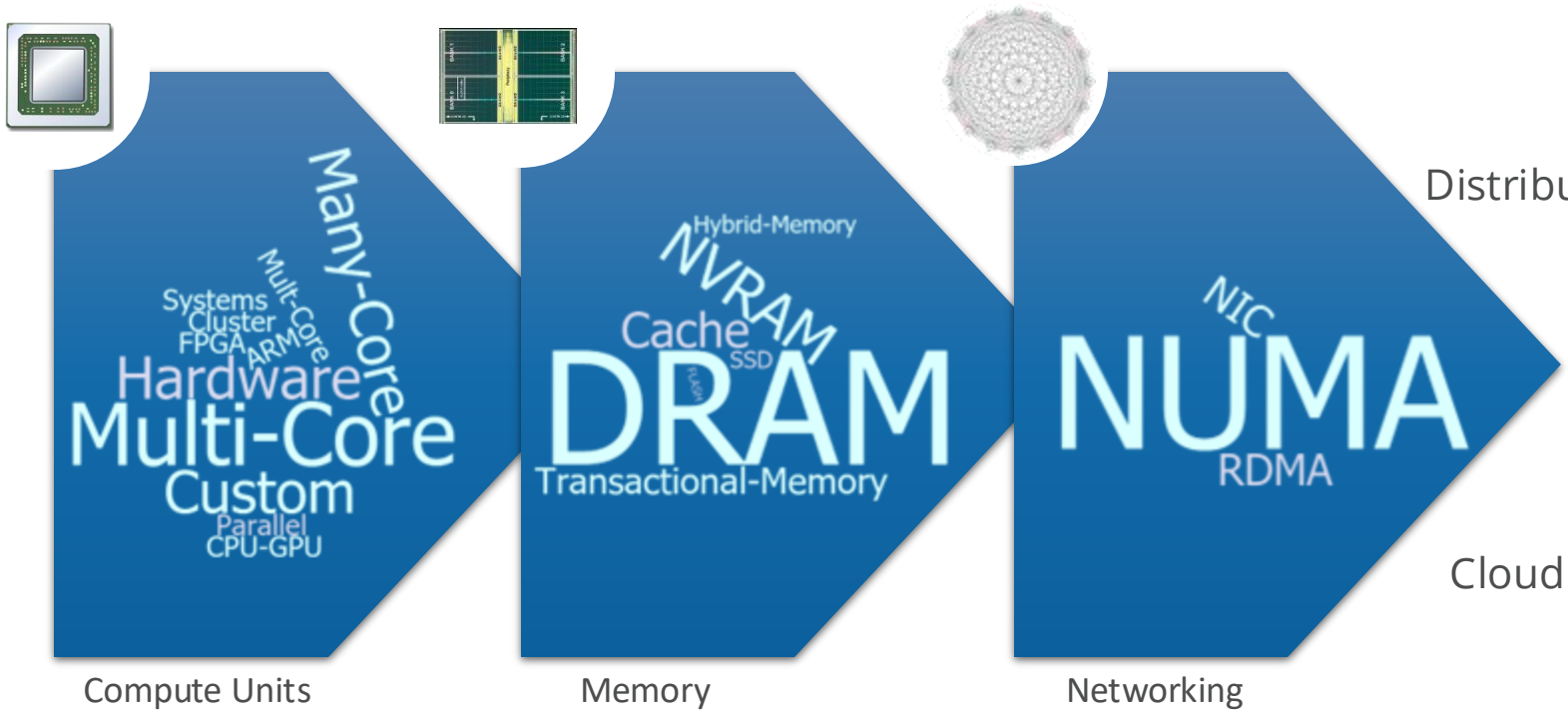
Increasing Number of Cores



Opportunities

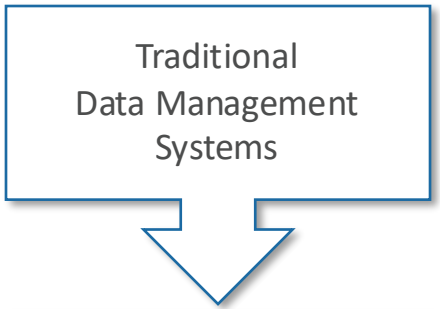
Modern Hardware

Modern Hardware Landscape

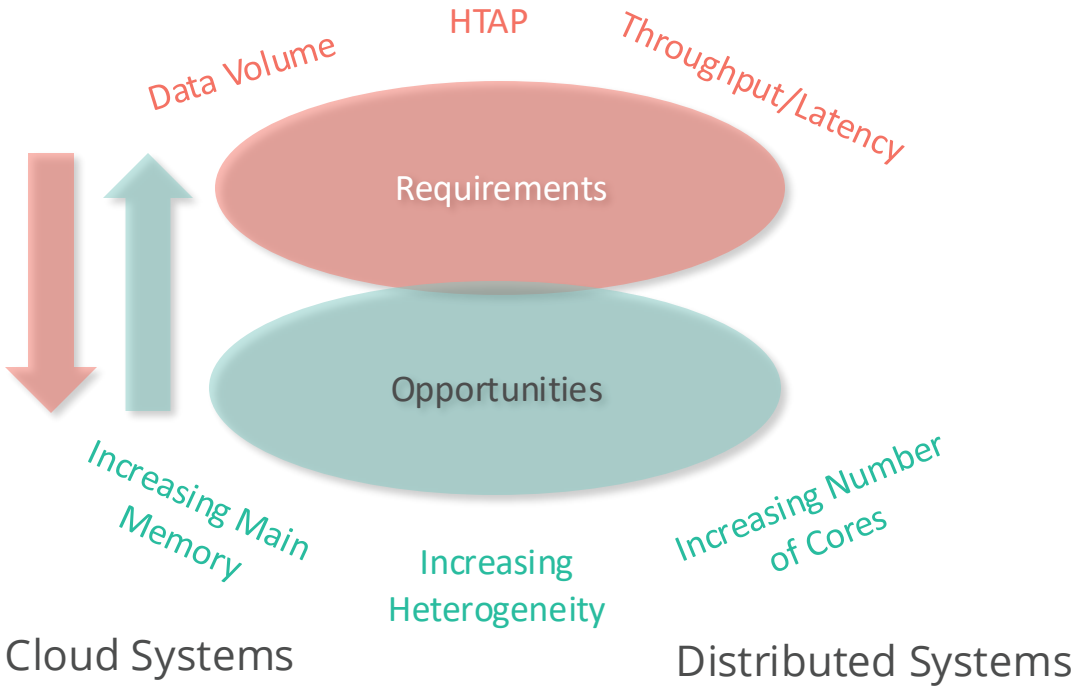


Research Field from a DBMS-Perspective

Dresden Database Research Group



Rethinking the whole architecture



What are the challenges??

Overview of Challenges

Scalability

- Independent parallel processing of sub-requests or tasks
- E.g., adding more servers permits serving more concurrent requests

Fault Tolerance

- Must mask failures and recover from hardware and software failures
- Must replicate data and service for redundancy

High Availability

- Service must operate 24/7

Consistency

- Data stored / produced by multiple services must lead to consistent results

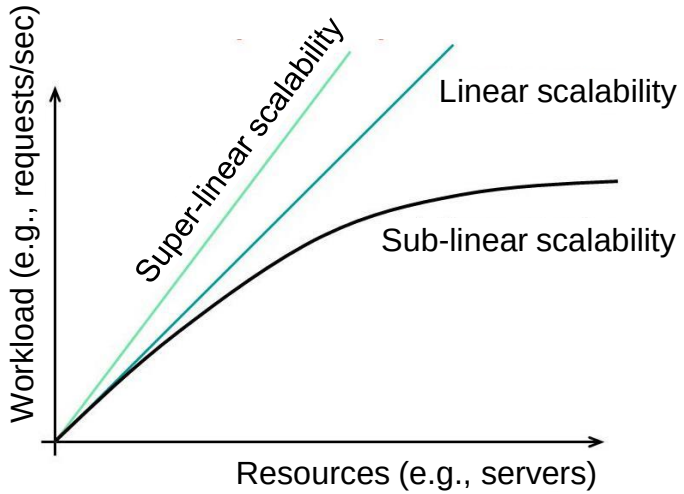
Scalability matters

Dresden Database Research Group

Ideally, adding N more servers should support N more users!

But, linear scalability is hard to achieve:

- Overheads + synchronization
- Load-imbalances create hot-spots
- (e.g., due to popular content, poor hash function)
- Amdahl's law → a straggler slows everything down



Approach for scalability

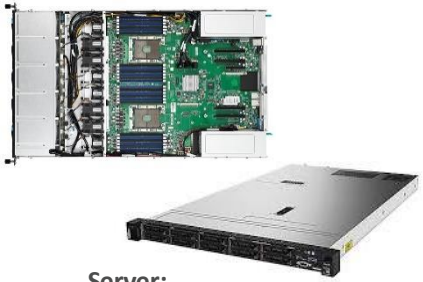
- Partition both data and compute

Fault Tolerance

Think of failure as the common case.

- Anatomy of a Data Center (Cloud provider)

Commodity CPU:
Xeon E5-2440: 6/12 cores
Xeon Gold 6148: 20/40 cores



Server:
Multiple sockets, RAM, disks

Rack:
16-64 servers + top-of-rack switch

Cluster:
Multiple racks + cluster switch

Data Center:
>100,000 servers

[Google Data Center, Eemshaven, Netherlands]

Fault Tolerance

Dresden Database Research Group

Think of failure as the common case.

[Christos Kozyrakis and Matei Zaharia: CS349D: Cloud Computing Technology, lecture, **Stanford 2018**]

Yearly Data Center Failures

- ~0.5 overheating (power down most machines in <5 mins, ~1-2 days)
- ~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hrs)
- ~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hrs)
- ~1 network rewiring (rolling ~5% of machines down over 2-day span)
- ~20 rack failures (40-80 machines instantly disappear, 1-6 hrs)
- ~5 racks go wonky (40-80 machines see 50% packet loss)
- ~8 network maintenances (~30-minute random connectivity losses)
- ~12 router reloads (takes out DNS and external vIPs for a couple minutes)
- ~3 router failures (immediately pull traffic for an hour)
- ~dozens of minor 30-second blips for dns
- ~1000 individual machine failures (2-4% failure rate, at least twice)
- ~thousands of hard drive failures (1-5% of all disks will die)

Hard disks have mean time to failure (MTTF) of about 10 to 50 years. In storage cluster with 10,000 disks, we should expect on average one disk to die per day.

Fault Tolerance

Dresden Database Research Group

Think of failure as the common case.

Full redundancy is too expensive → use failure recovery.

- Impossible to build redundant systems at scale
- Rather reduce the cost of failure recovery

Failure recovery: replication or re-computation

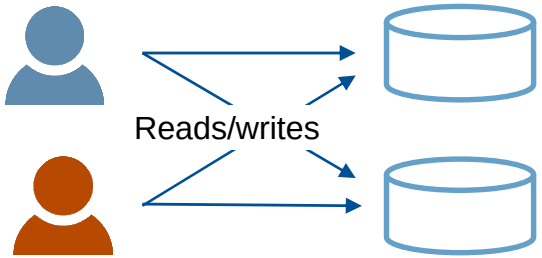
- Which one is better, depends on the respective costs and application use case

- **Replication**:
  - Need to replicate data and service
  - Introduces the consistency issues
- **Re-computation**
  - Easy for stateless services
  - Remember data lineage for compute jobs

Consistency

Dresden Database Research Group

Many applications need state replication across a wide area, for reliability, availability and low latency.



CAP-Theorem

- It is impossible for a distributed data store to simultaneously provide more than two out of the three guarantees:
  - Consistency
  - Availability
  - Partition tolerance

High Availability

Dresden Database Research Group

Observation

- Downtime → bad customer experience, and loss in revenue.
- According to Gartner, a minute of IT downtime costs companies $5’600 on average.

For a high available service, the design have to consider:

- Eliminate single point of failure by adding redundancy in the system.
- Have a reliable crossover.
- Have an efficient way to monitor and detect failures when they occur.

Overview of Challenges (recap)

Dresden Database Research Group

Scalability

- Being able to elastically scale (out and in) to meet the load demand is crucial.

Fault Tolerance

- Accept the reality that faults are common and build for quick detection and recovery.

High Availability

- Minimize costs for downtime.

Consistency

- Keep CAP-Theorem in mind.

The modern revolution

Dresden Database Research Group

Impact on how database systems should be designed

Traditional Systems	Modern Cloud Systems
Monolithic	Decomposed
Designed for predictable scalability	Designed for elastic scale
Relational Database	Mix of storage technologies
Synchronized processing	Asynchronous processing
Design to avoid failures	Design for failure recovery
Occasional large updates	Frequent small updates
Manual management	Automated self-management
Snowflake servers	Immutable infrastructure

Design principles /1

Dresden Database Research Group

Design for self-healing.

- In a distributed system, failures happen all the time. Design the application to be self-healing.

Make all things redundant.

- Build redundancy into your application to avoid having single points of failure.

Minimize coordination.

- Minimize coordination between application services to achieve better scalability.

Design to scale out/scale-in/scale-up

- Design your application so that it can scale in different dimensions.

Partition around limits.

- Use partitioning to work around database, network and compute limits.

Design principles /2

Dresden Database Research Group

Use of stateless services.

- Scaling without having a state is trivial.

Caching

- Latency is king. Caching helps to significantly reduce the job's latency.

Use the best data store for the job.

- Pick the storage technology that is the best fit for your data and how it will be used.

Distribute computation

- Partition/Aggregate compute pattern is one that scales pretty well.

Design for evolution

- An evolutionary design is key for continuous innovation.

Structure of the Course

Dresden Database Research Group

Foundations

- Relational Data and SQL
- Traditional Database Architecture
- Transactions

Parallelism

- Data-oriented Architecture
- Coarse- and Fine-grained parallelism
- Data Fragmentation

Transactions

- Data Replication
- Distributed Transactions
- CAP-Theorem

Additional

- Map/Reduce and Hadoop
- Spark and its Ecosystem
- Database in the cloud

Homework

Dresden Database Research Group

Please read the following paper

- 01-TheSeattleReportOnDatabaseResearch.pdf (folder papers in OPAL)

DOI:10.1145/3524284

DANIEL ABADI
ANASTASIA AILAMAKI
DAVID ANDERSEN
PETER BAILIS
MAGDALENA BALAZINSKA
PHILIP A. BERNSTEIN
PETER BONCZ
SURAJIT CHAUDHURI
ALVIN CHEUNG
ANHAI DOAN
LUNA DONG
MICHAEL J. FRANKLIN
JULIANA FREIRE
ALON HALEVY
JOSEPH M. HELLERSTEIN
STRATOS IDREOS

Every five years, a group of the leading database researchers meet to reflect on their community's impact on the computing industry as well as examine current research challenges.

The Seattle Report on Database Research