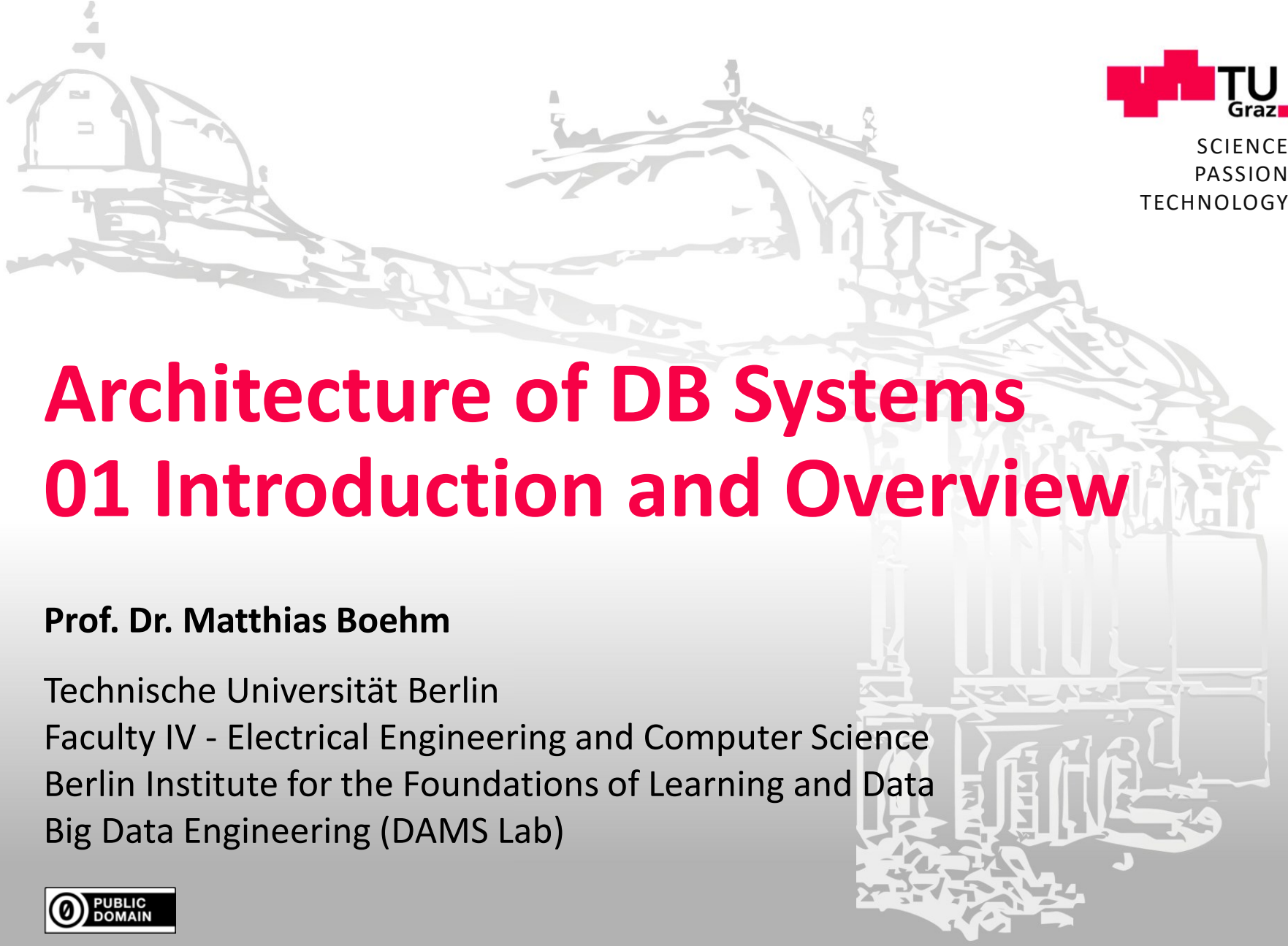


SCIENCE
PASSION
TECHNOLOGY

Architecture of DB Systems
01 Introduction and Overview

Prof. Dr. Matthias Boehm

Technische Universität Berlin
Faculty IV - Electrical Engineering and Computer Science
Berlin Institute for the Foundations of Learning and Data
Big Data Engineering (DAMS Lab)

Last update: Oct 18, 2023

ISDS

Announcements/Org

- **#1 Lecture Format**
  - Introduction virtual, remaining lectures blocked **Dec 04 - Dec 07**
  - Optional attendance
  - **Hybrid**, in-person but live-streaming / video-recorded lectures
    - **HS i10** + Zoom: https://tu-berlin.zoom.us/j/9529634787?pwd=R1ZsN1M3SC9BOU1OcFdmem9zT202UT09

- **#2 Course Registration** (as of Oct 12)
  - **Architecture of Database Systems** (ADBS)

WS20/21: **73 (0)**
WS21/22: **94 (0)**
WS23/24: **101 (0)**

Agenda

- **Data Management Group**
- **Course Organization**
- **Course Motivation and Goals**
- **Course Outline and Projects**
- **Excursus: DAPHNE Project**

Data Management Group

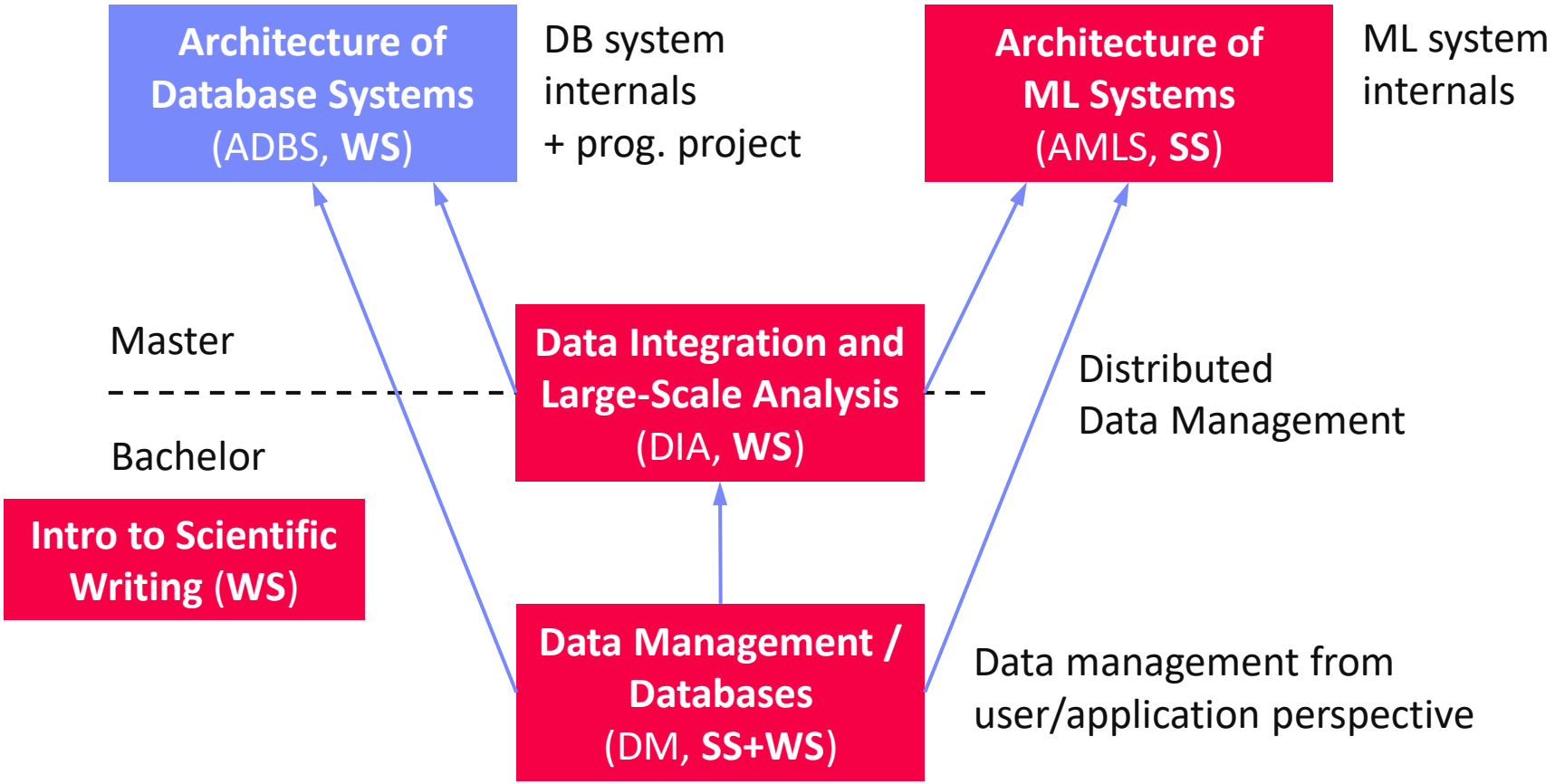
https://damslab.github.io/

About Me

- **Since 09/2022 TU Berlin**, Germany
  - University professor for Big Data Engineering (DAMS)
- **2018-2022 TU Graz**, Austria
  - BMK endowed chair for data management & RAM
  - **Data management for data science** (DAMS), **SystemDS & DAPHNE**
- **2012-2018 IBM Research – Almaden**, CA, USA
  - Declarative large-scale machine learning
  - Optimizer and runtime of **Apache SystemML**
- **2007-2011 PhD TU Dresden**, Germany
  - Cost-based optimization of integration flows
  - Time series forecasting / indexing & query processing

DB group

Data Management Courses (at TU Graz)



Course Organization

Basic Course Organization


- **Staff**
  - Lecturer: Prof. Dr. Matthias Boehm (TU Berlin)
  - Assistants: M.Tech. Arnab Phani (TU Berlin)
- **Language**
  - Lectures and slides: **English**
  - Communication and examination: **English**/**German**
- **Course Format**
  - VU 2/1, **5 ECTS** (2x 1.5 ECTS + 1x 2 ECTS), bachelor/master
  - **Lectures** (**Wed 6pm** kickoff + blocked, including **Q&A**), **attendance optional**
  - **Mandatory programming project** (~2 ECTS)
  - **Recommended papers** for additional reading on your own
- **Prerequisites**
  - **Preferred:** course Data Management / Databases is very good start
  - **Sufficient:** basic understanding of SQL / RA (or willingness to fill gaps)
  - Basic programming skills in low-level language (C, C++)

Course Logistics

- **Website**
  - https://mboehm7.github.io/teaching/ws2324_adbs/index.htm
  - All course material (lecture slides) and dates
- **Live-Streaming / Video Recording** (on website)
- **Communication**
  - **Informal language** (first name is fine)
  - Please, **immediate feedback** (unclear content, missing background)
  - Newsgroup: N/A – email is fine, summarized in following lectures
  - **Office hours:** by appointment or after lecture
- **Exam**
  - **Completed programming project** (checked by me/staff)
  - **Final written exam** (oral exam if <=35 students take the exam)
  - **Grading** (30% project/exercises completion, 70% exam)

Course Logistics, cont.

- **Course Applicability**
  - **Master** programs computer science (CS), as well as software engineering and management (SEM)
    - Catalog Data Science (elective course in major/minor)
    - Catalog Software Technology (elective course in major/minor)
  - **Free subject course** in any other study program or university

Course Motivation and Goals

History 1970/80s (relational)

Ingres @ UC Berkeley (Stonebraker et al., Turing Award '14)

QUEL

SQL Standard (SQL-**86**)

SEQUEL

System R @ IBM Research – Almaden (Jim Gray et al., Turing Award '98)

Oracle, IBM DB2, Informix, Sybase → MS SQL

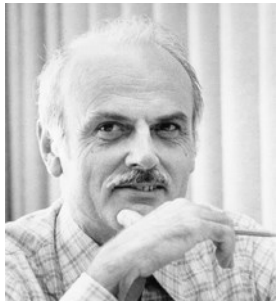
Tuple Calculus

Relational Algebra

Relational Model

Goal: Data Independence (physical data independence)

- Ordering Dependence
- Indexing Dependence
- Access Path Dependence



Edgar F. "Ted" Codd @ IBM Research (Turing Award '81)

[E. F. Codd: A Relational Model of Data for Large Shared Data Banks. Comm. ACM 13(6), **1970**]

Success of SQL / Relational Model

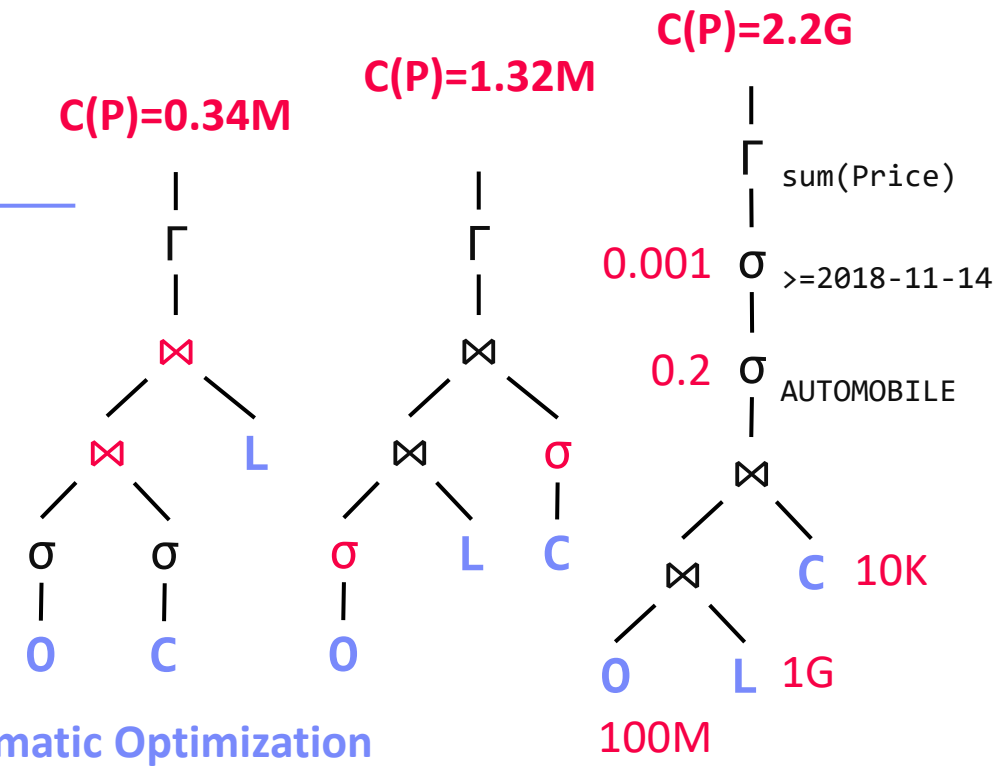
Query:

```
SELECT O_OID, sum(L_Price)
FROM Orders, Lineitem, Customer
WHERE O_OID = L_OID AND O_CID = C_CID
   AND O_Odate >= '2018-11-14'
   AND C_Msegment = 'AUTOMOBILE'
GROUP BY O_OID
```

#1 Declarative:
what not how

#2 Flexibility:
closure property
→ composability

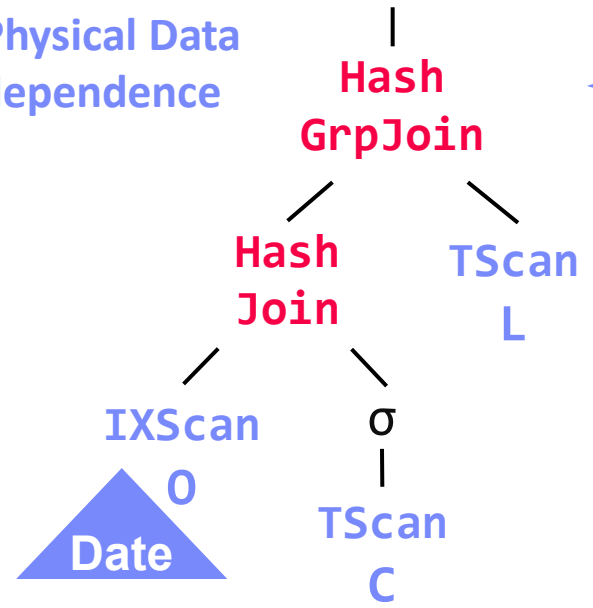
Logical Query Plans



#3 Automatic Optimization

#4 Physical Data Independence

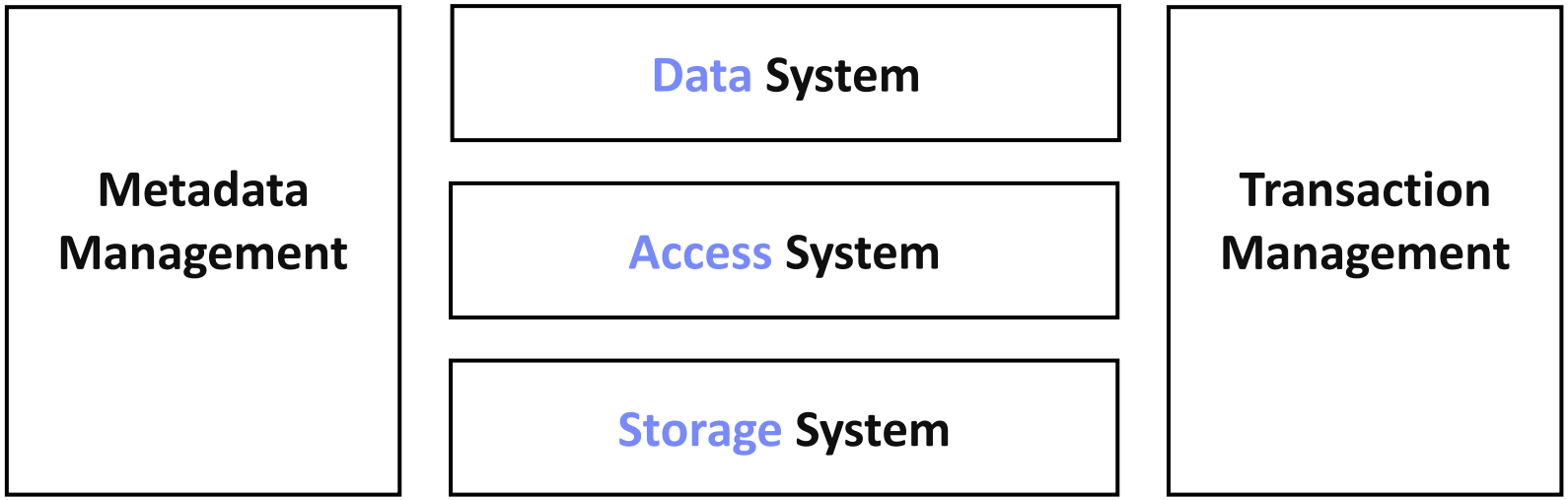
Physical Query Plan



DBMS Architecture

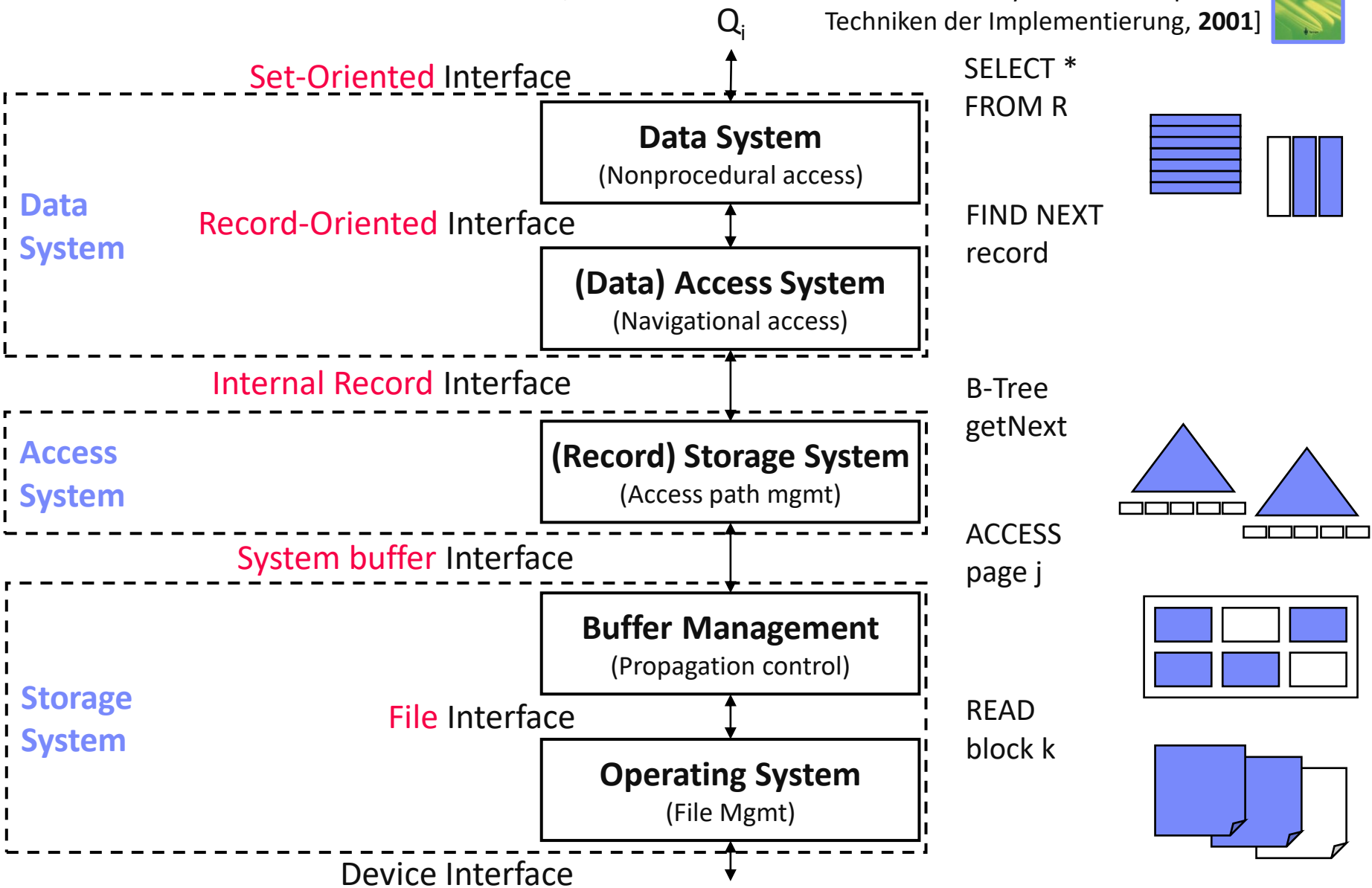
[Theo Härder, Erhard Rahm: Datenbanksysteme: Konzepte und Techniken der Implementierung, **2001**]

- **Coarse-grained System Architecture**



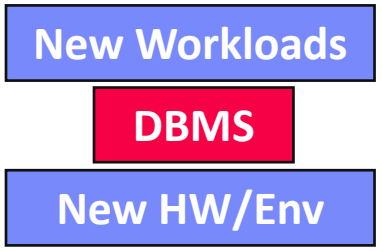
DBMS Architecture, cont.

[Theo Härder, Erhard Rahm: Datenbanksysteme: Konzepte und Techniken der Implementierung, **2001**]



Course Goals

- **Constantly Changing Environment**
  - New application and **data analysis workloads**
  - Heterogeneous and changing **hardware characteristics**



- **#1 Architecture and internals of traditional/modern DB systems**
- **#2 Understanding of DB characteristics → better evaluation / usage**
- **#3 Understanding of effective techniques → build/extend DB systems**
  (these fundamental techniques are **broadly applicable** in other systems)

Course Outline and Projects

Course Outline

A: System Architecture and Data Access

- **01 Introduction and Overview** [Oct 18, 6pm]
- **02 DB System Architectures** [Dec 04, 9.30am]
- **03 Data Layouts and Bufferpool Management** [Dec 04, 12.30pm]
- **04 Index Structures and Partitioning** [Dec 04, 3pm]
- **05 Compression Techniques** [Dec 04, 5.30pm]

B: Query Processing and Optimization

- **06 Query Processing (operators, execution models)** [Dec 05, 9am]
- **07 Query Compilation and Parallelization** [Dec 05, 11.30pm]
- **08 Query Optimization (rewrites, costs, join ordering)** [Dec 05, 2.30pm]
- **09 Adaptive Query Processing** [Dec 07, 9am]

Course Outline, cont.

C: Emerging Topics

- **10 Cloud Database Systems** [Dec 07, 12.30pm]
- **11 Modern Concurrency Control** [Dec 07, 3pm]
- **12 Modern Storage and HW Accelerators** [Dec 07, 5.30pm]

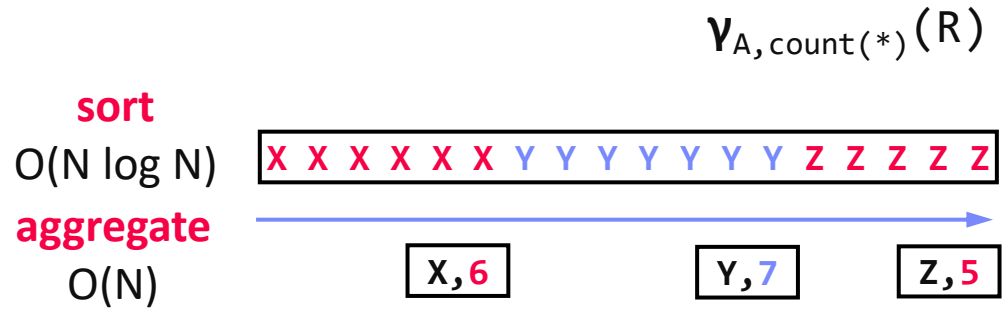
Overview Programming Project

- **Team**
  - **1-3 person teams** (w/ clearly separated responsibilities)
- **Task: Efficient Group-by Aggregation**
  - Column-oriented frame storage w/ materialized intermediates
  - Multi-threaded group-by aggregation w/ multiple group-by columns, additive aggregation functions, different data types / characteristics
  - C test / performance suites → correct and minimum perf
  - Programming language: no restrictions, but **C or C++** recommended
- **Timeline**
  - Test drivers, reference implementation available on website
  - **Jan 21, 11.59pm:** Final programming project deadline (submission via email)
- **Price Top-1 Submission** (and min perf target)
  - 1x Attendance **SIGMOD'25 in Berlin** (travel, hotel, conf registration)

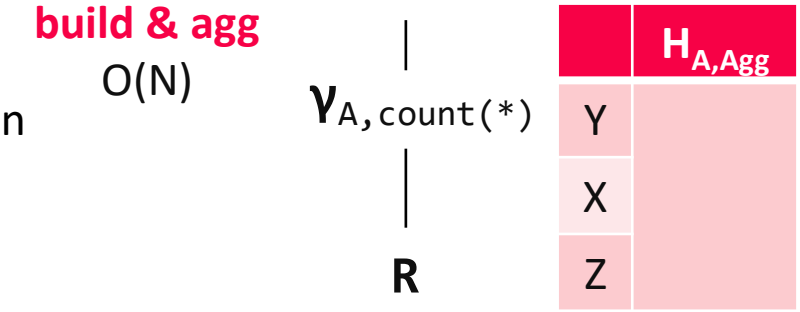
Overview Programming Project, cont.

- **Recap: Classification of Aggregates (DM, DIA)**
  - Additive, semi-additive, additively-computable, others

- **#1 Sort Group-By**
  - Similar to sort-merge join (Sort, GroupAggregate)
  - Sorted group output



- **#2 Hash Group-By**
  - Similar to hash join (HashAggregate)
  - Higher temporary memory consumption
  - Unsorted group output
  - **#1** w/ **tuple grouping**
  - **#2** w/ **direct aggregation** (e.g., count)
  - **Beware:** cache-unfriendly if many groups (size(H) > L2/L3 cache)



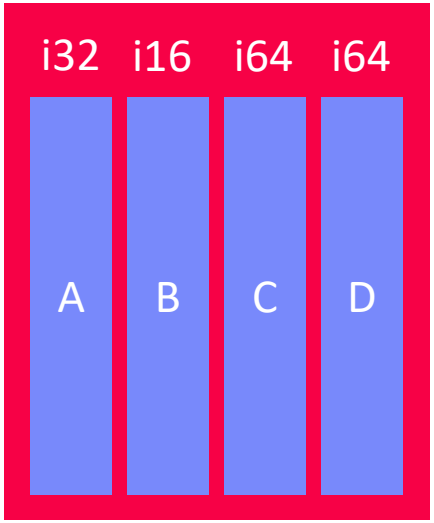
Overview Programming Project, cont.

- **API Sketch**
  - Materialized inputs, outputs
  - Multiple group by attributes
  - Multiple aggregated attributes
  - Aggregation functions: sum/min/max

$$\gamma_{sum(C)}(R)$$
$$\gamma_{A,sum(C)}(R)$$
$$\gamma_{A,B,sum(C)}(R)$$
$$\gamma_{A,B,sum(C),sum(D)}(R)$$

- **Data Characteristics**
  - Frames w/ column-oriented storage
  - Data Types: INT16, INT32, INT64
  - Varying # distinct values, skew, missing values

```
getNumRows()
getNumCols()
getRow(size_t r)
getCol(size_t c)
```



- **Performance Target**
  - Relative to [naïve] reference implementation
  - **Perf target: score >= # physical-cores / 2**

DAPHNE:

Integrated **D**ata **A**nalysis **P**ipelines for Large-Scale DM, **H**PC, and ML

Motivation, Vision, and System Architecture

https://daphne-eu.github.io/

DAPHNE

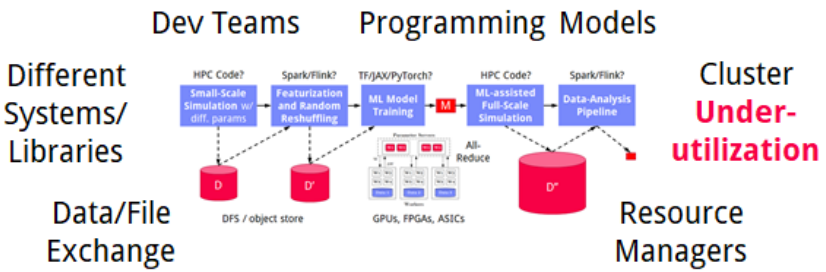
[Louvre, Paris]

➔ DAPHNE Overall Objective: Open and extensible system infrastructure

Motivation

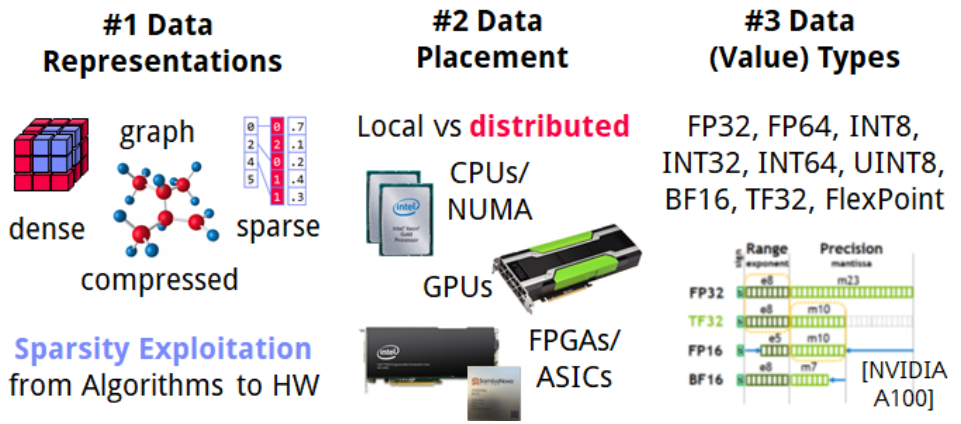
- **Integrated Data Analysis Pipelines**
  - Open data formats, query processing
  - Data preprocessing and cleaning
  - ML model training and scoring
  - HPC, custom codes, and simulations

Deployment Challenges



- **Hardware Challenges**
  - DM+ML+HPC share compilation and runtime techniques / converging cluster hardware
  - **End of Dennard scaling:** $P = \alpha CFV^2$ (power density 1)
  - **End of Moore's law**
  - **Amdahl's law:** sp = 1/s

➔ **Increasing Specialization**



DAPHNE System Architecture

[**CIDR 2022**]

[Patrick Damme et al.: DAPHNE: An Open and Extensible System Infrastructure for Integrated Data Analysis Pipelines, **CIDR 2022**]

DaphneLib (API)

DaphneDSL (Domain-specific Language)

MLIR

MLIR-Based Compilation Chain

DaphneIR (MLIR Dialect)

Optimization Passes

New Runtime Abstractions for Data, Devices, Operations

Hierarchical Scheduling

Device Kernels (CPU, GPU, FPGA, Storage)

Vectorized Execution Engine (Fused Op Pipelines)

Sync/Async I/O Buffer/Memory Management

Local (embedded) and Distributed Environments (standalone, HPC, data lake, cloud, DB)

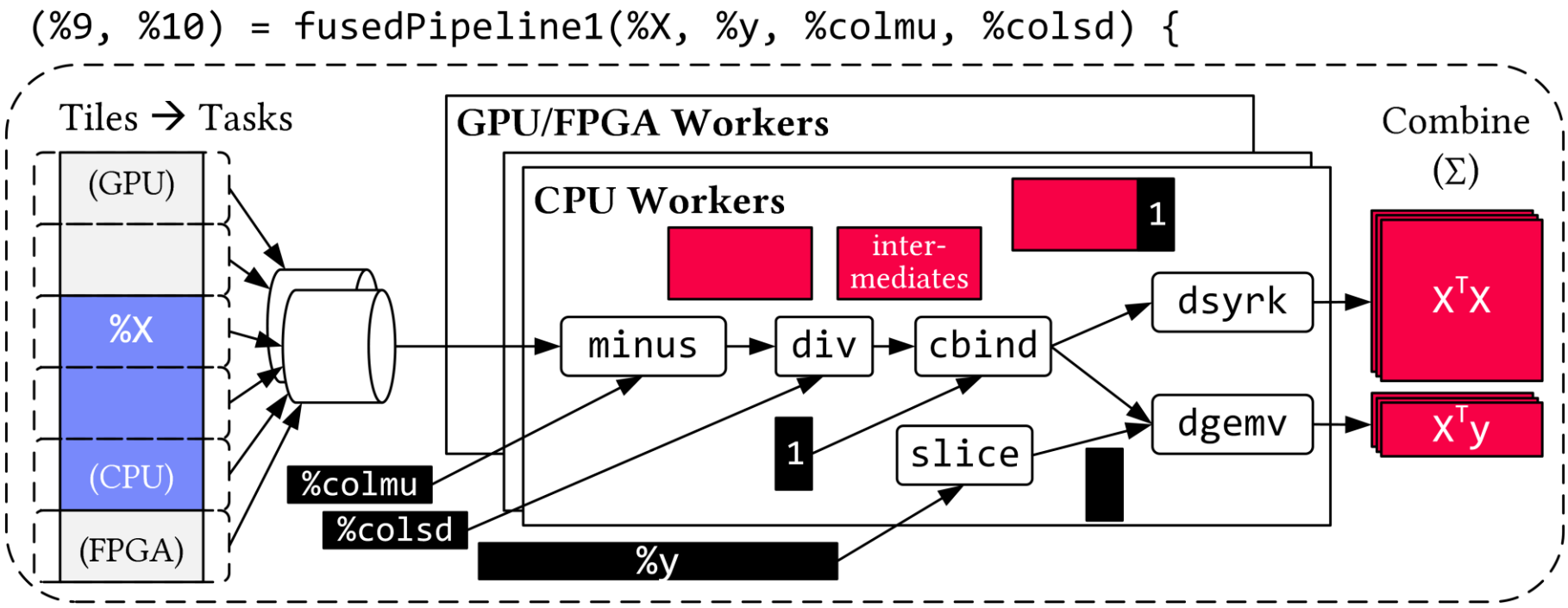
Extensible Infrastructure

MLIR Dialects, Extension Catalog (new dtypes, kernels, scheduling algs)

Fine-grained Fusion and Parallelism

Integration w/ Resource Mgmt & Prog. Models

Vectorized (Tiled) Execution



Default Parallelization Frame & Matrix Ops

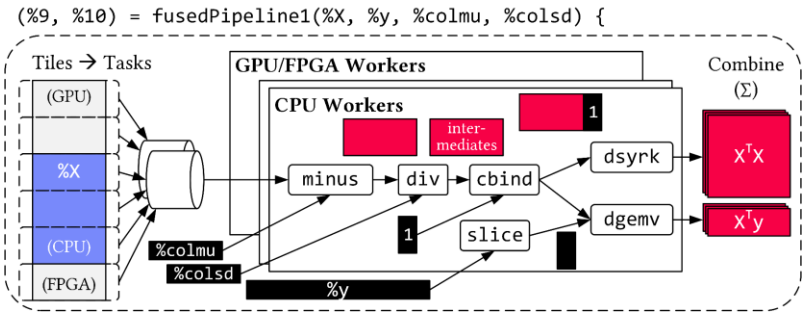
Locality-aware, Multi-device Scheduling

Fused Operator Pipelines on Tiles/Scalars + Codegen

[PVLDB 2018]

Vectorized (Tiled) Execution, cont.

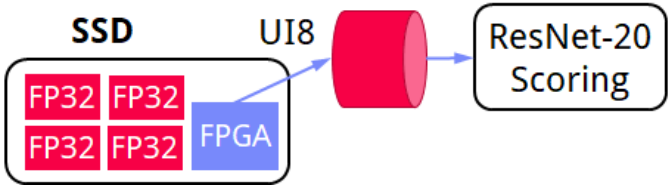
- **#1 Zero-copy Input Slicing**
  - Create view on sliced input (no-op)
  - All kernels work on views



- **#2 Sparse Intermediates**
  - Reuse dense/sparse kernels
  - Sparse pipeline intermediates for free

- **#3 Fine-grained Control**
  - Task sizes (dequeue, data access) vs data binding (cache-conscious ops)
  - Scheduling for load balance (e.g., sparse operations)

- **#4 Computational Storage**
  - Task queues connect eBPF programs, async I/O into buffers, and op pipelines



Distributed Vectorized Execution

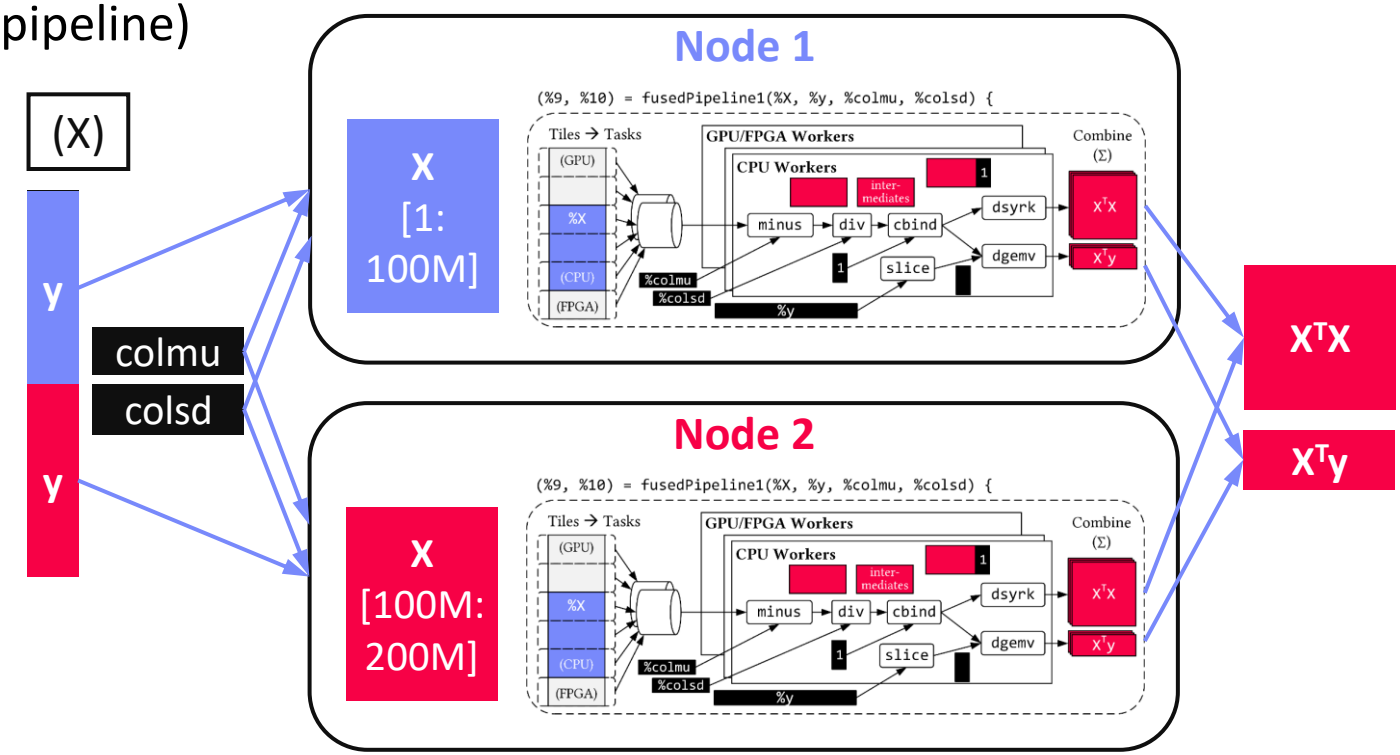
- **Federated matrices/frames + distribution primitives**
- **Hierarchical vectorized pipelines and scheduling**

Coordinator
(spawns fused pipeline)

#1 Prepare Inputs

#2 Coarse-grained Tasks Scheduling

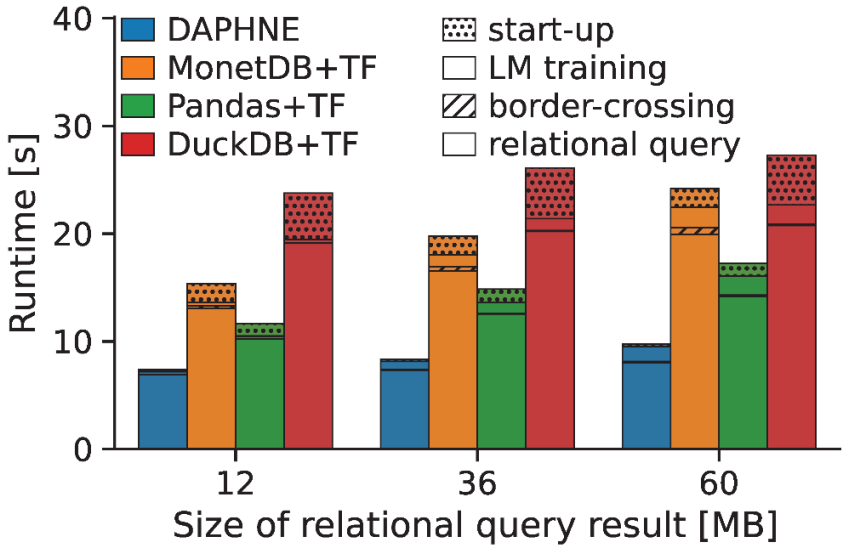
#3 Combine Outputs



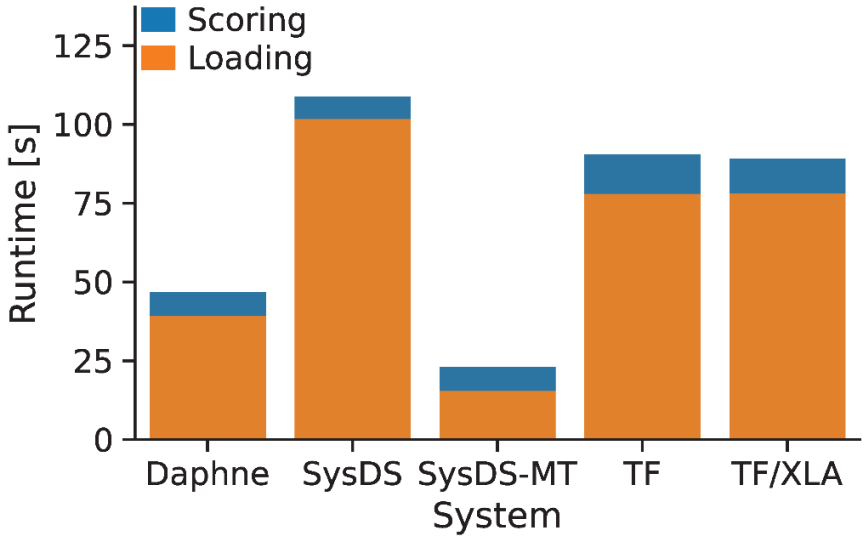
DAPHNE – Experiments (simple IDA pipelines)

- **Setup:** Single node w/ 2x Intel Xeon Gold 6238 (112 vcores, 7.7 TFLOP/s), 768 GB DDR4 RAM, 12x 2TB SSDs (data), NVIDIA **T4 GPU** (8.1 TFLOP/s, 16 GB), and Intel FPGA PAC D5005 (w/ Stratix **10SX FPGA**, 32 GB) since Dec 29

P1: TPC-H SF10 csv, query processing + linear regression training on CPUs



P2: So2Sat LCZ42 csv (testset), ResNet-20 scoring on GPU



Summary and Q&A

- **Course Goals**
  - **#1** Architecture and internals of traditional/modern DB systems
  - **#2** Understanding of DB characteristics → **better evaluation / usage**
  - **#3** Understanding of effective techniques → **build/extend DB systems** (these fundamental techniques are **broadly applicable** in other systems)
- **Programming Project**
  - Column-oriented frame storage w/ materialized intermediates
  - **Multi-threaded group-by aggregation** w/ multiple group-by columns, additive aggregation functions, different data types / characteristics
- **Next Lectures**
  - 02 **DB System Architectures** [Dec 04]
  - 03 **Data Layouts and Bufferpool Management** [Dec 04]
  - 04 **Index Structures and Partitioning** [Dec 04]
  - 05 **Compression Techniques** [Dec 04]