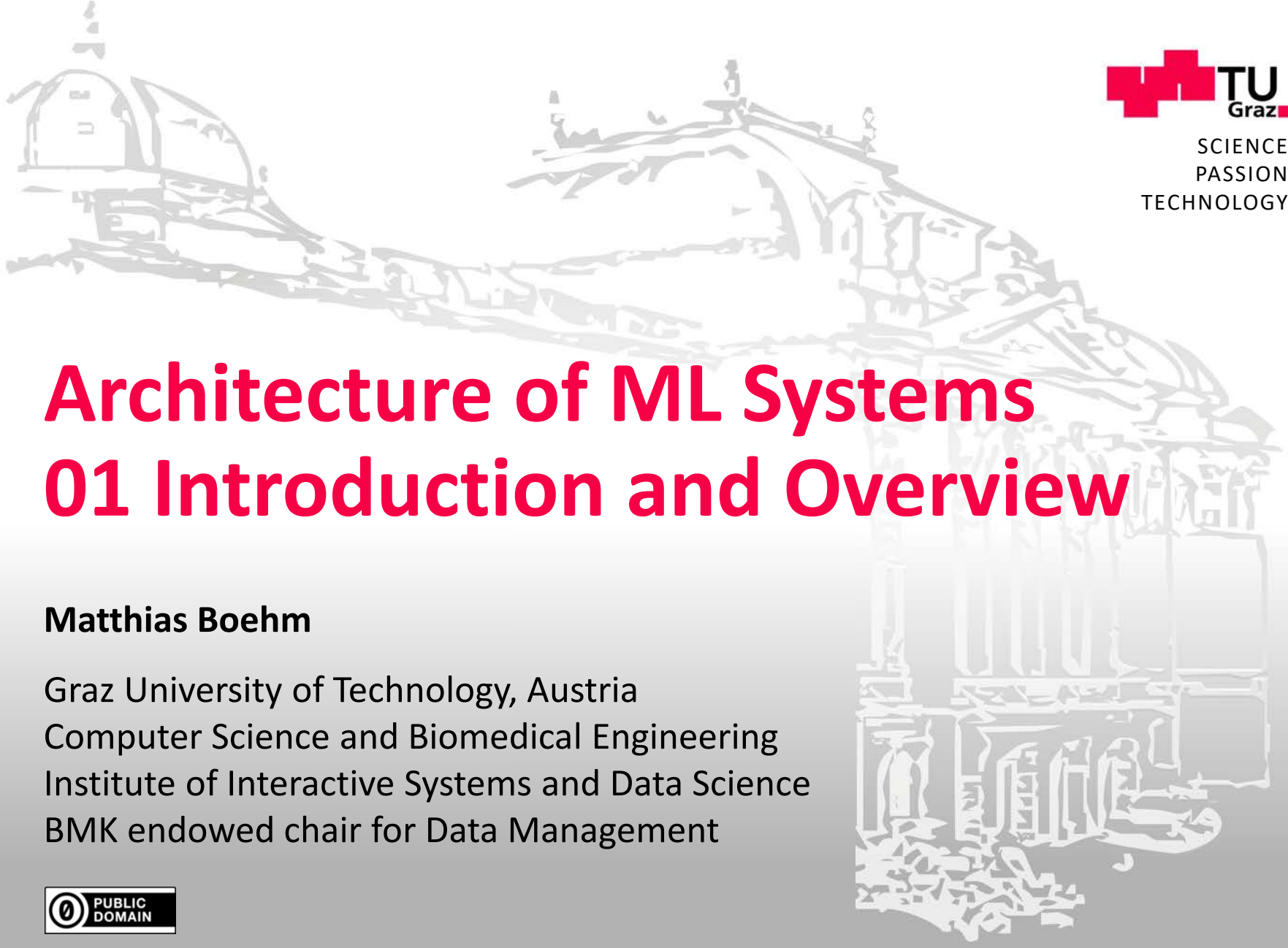


SCIENCE
PASSION
TECHNOLOGY

Architecture of ML Systems
01 Introduction and Overview

Matthias Boehm

Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMK endowed chair for Data Management

Last update: Mar 02, 2022

ISDS

Announcements/Org

- **#1 Video Recording**
  - Link in **TeachCenter** & **TUbe** (lectures will be public)
  - Optional attendance (independent of COVID)
  - Hybrid, in-person, TUbe video-recording, and webex live: https://tugraz.webex.com/meet/m.boehm
  - **Update:** status **ORANGE**, max 50% capacity, 2.5G rule

- **#2 Course Registrations** (as of Mar 02)
  - **Architecture of Machine Learning Systems** (AMLS): **111 (7)**
  - Bachelor/master/PhD ratio? CS/other ratio?

- **#3 Gründungsgarage Volume XVIII**
  - Academic Startup Accelerator
  - https://www.gruendungsgarage.at/
  - Next application deadline: **Mar 13, 2022**

Agenda

- **Data Management Group**
- **Motivation and Goals**
- **Course Organization**
- **Course Outline, and Projects**
- **Apache SystemDS and DAPHNE**

Data Management Group

https://damslab.github.io/

About Me

- **2018-2022 TU Graz**, Austria
  - BMK endowed chair for data management
  - **Data management for data science**
    (ML systems internals, end-to-end data science lifecycle)

https://github.com/apache/systemds

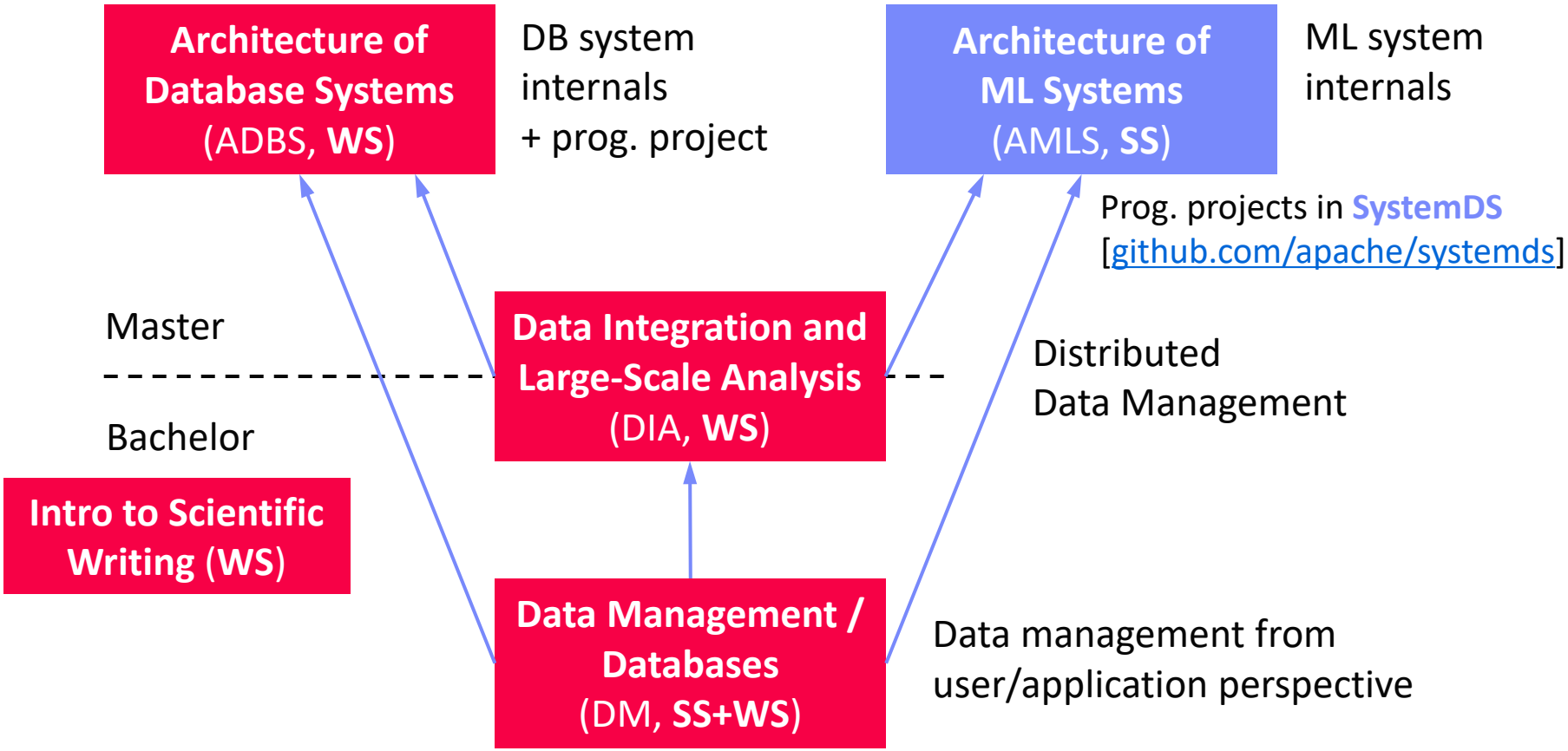
- **2012-2018 IBM Research – Almaden**, USA
  - Declarative large-scale machine learning
  - Optimizer and runtime of **Apache SystemML**

- **2011 PhD TU Dresden**, Germany
  - Cost-based optimization of integration flows
  - Systems support for time series forecasting
  - In-memory indexing and query processing

TECHNISCHE UNIVERSITÄT DRESDEN

DB group

Data Management Courses

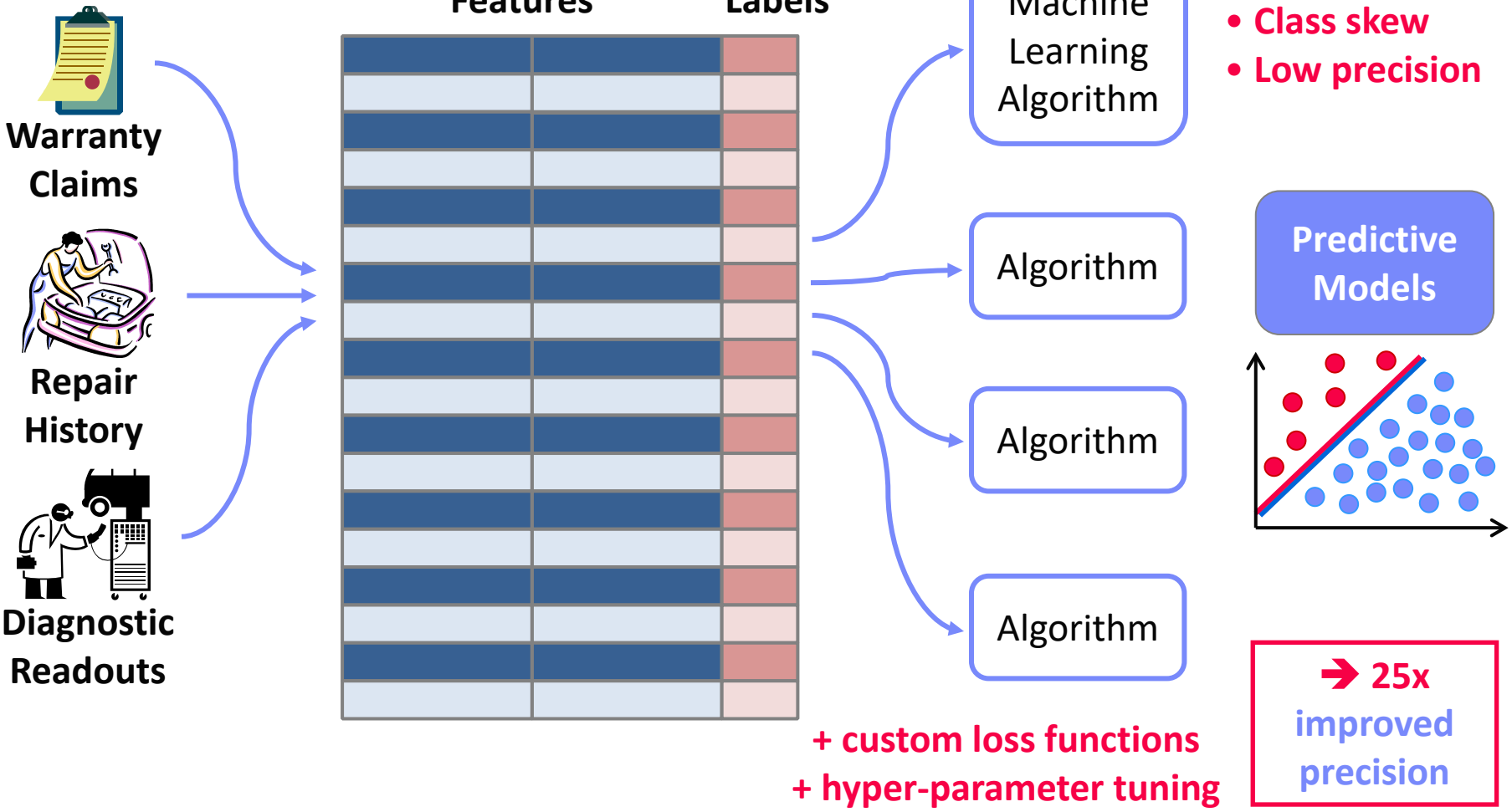


Motivation and Goals

Example ML Applications (Past/Present)

- **Transportation / Space**
  - **Lemon car detection and reacquisition** (classification, seq. mining)
  - **Airport passenger flows from WiFi data** (time series forecasting)
  - **Data analysis for assisted driving** (various use cases)
  - **Automotive vehicle development** (ML-assisted simulations)
  - Satellite senor analytics (regression and correlation)
  - Earth observation and **local climate zone classification** and monitoring
- **Finance**
  - Water cost index based on various influencing factors (regression)
  - **Insurance claim cost per customer** (model selection, regression)
  - **Financial analysts survey correlation** (bivariate stats w/ new tests)
- **Health Care**
  - **Breast cancer cell grow from histopathology images** (classification)
  - **Glucose trends and warnings** (clustering, classification)
  - Emergency room diagnosis / patient similarity (classification, clustering)
  - Patient survival analysis and prediction (Cox regression, Kaplan-Meier)

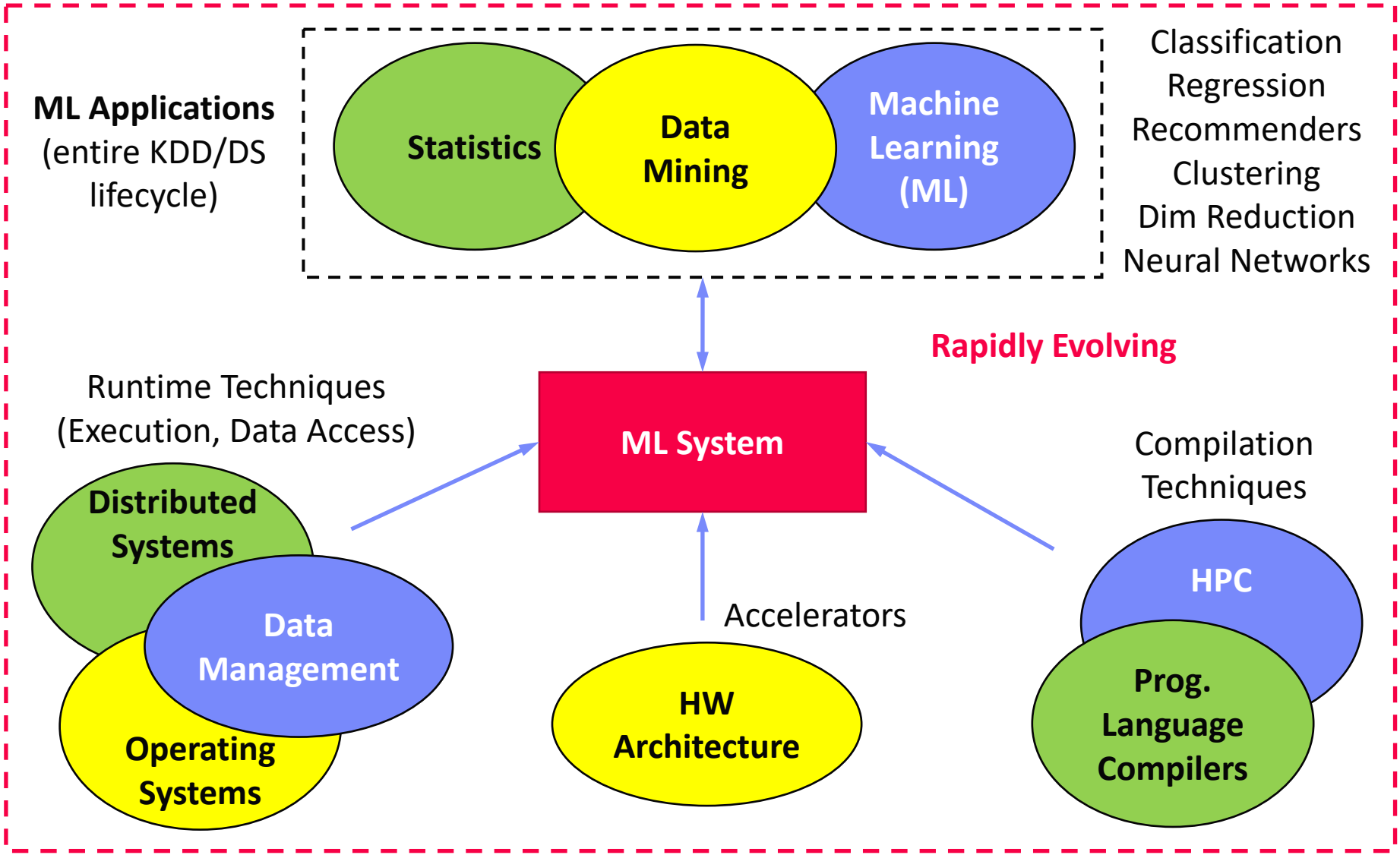
A Car Reacquisition Scenario



Example ML Applications (Past/Present), cont.

- **Production/Manufacturing**
  - **Paper and fertilizer production** (regression/classification, anomalies)
  - **Semiconductor manufacturing**, and **material degradation** modeling
  - **Mixed waste sorting and recycling** (composition, alignment, quality)
- **Other Domains**
  - **Machine data: errors and correlation** (bivariate stats, seq. mining)
  - Smart grid: energy demand/RES supply, weather models (forecasting)
  - **Elastic flattening via sparse linear algebra** (spring-mass system)
- **Information Extraction**
  - **NLP contracts → rights/obligations** (classification, error analysis)
  - **PDF table recognition and extraction, OCR** (NMF clustering, custom)
  - **Learning explainable linguistic expressions** (learned FOL rules, classification)
- **Algorithm Research** (+ various state-of-the art algorithms)
  - **User/product recommendations** via various forms of NMF
  - Localized, supervised metric learning (dim reduction and classification)
  - Learning word embeddings via orthogonalized skip-gram

What is an ML System?



What is an ML System?, cont.

- **ML System**
  - **Narrow focus:** SW system that executes ML applications
  - **Broad focus:** Entire system (HW, compiler/runtime, ML application)

  ➔Trade-off **runtime/resources** vs **accuracy**

  ➔Early days: no standardizations (except some exchange formats), lots of different languages and system architectures, but many shared concepts

- **Course Objectives**
  - Architecture and internals of modern (large-scale) ML systems
    - **Microscopic view** of ML system internals
    - **Macroscopic view** of ML pipelines and data science lifecycle
  - **#1** Understanding of characteristics → **better evaluation / usage**
  - **#2** Understanding of effective techniques → **build/extend ML systems**

Course Organization

Basic Course Organization

- **Staff**
  - Lecturer: Univ.-Prof. Dr.-Ing. Matthias Boehm, ISDS
  - Assistants: M.Sc. Shafaq Siddiqi, M.Sc. Sebastian Baunsgaard
- **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
  - **Weekly lectures** (**start 12.15pm**, including **Q&A**), **attendance optional**
  - **Mandatory programming project** or exercises (2 ECTS)
  - **Recommended papers** for additional reading on your own
- **Prerequisites (preferred)**
  - Basic courses Data Management/Databases, and
  - Basic courses on applied ML / Knowledge Discovery and Data Mining

Course Logistics

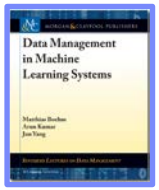
- **Website**
  - https://mboehm7.github.io/teaching/ss22_amls/index.htm
  - All course material (lecture slides) and dates
- **Video Recording / Live Streaming Lectures (TUbe, webex)**
- **Communication**
  - **Informal language** (first name is fine)
  - Please, **immediate feedback** (unclear content, 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 <=25 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 (compulsory course in major, and elective)
    - Catalog Machine Learning (elective course)
    - Catalog Interactive and Visual Information Systems (elective course)
    - Catalog Software Technology (elective course)
  - **PhD** CS doctoral school list of courses
  - **Free subject course** in any other study program or university

Course Outline and Projects

Partially based on



[Matthias Boehm, Arun Kumar, Jun Yang: Data Management in Machine Learning Systems. Synthesis Lectures on Data Management, Morgan & Claypool Publishers 2019]

Major updates in SS2020, SS2021, and SS2022

Part A: Overview and ML System Internals

- **01 Introduction and Overview** [Mar 04]
- **02 Languages, Architectures, and System Landscape** [Mar 11]

- **03 Size Inference, Rewrites, and Operator Selection** [Mar 18]
- **04 Operator Fusion and Runtime Adaptation** [Mar 25]
- **05 Data- and Task-Parallel Execution** [Apr 01]
- **06 Parameter Servers** [Apr 08]

- **07 Hybrid Execution and HW Accelerators** [Apr 29]
- **08 Caching, Partitioning, Indexing, and Compression** [May 06]

Part B: ML Lifecycle Systems

- **09 Data Acquisition, Cleaning, and Preparation** [May 13]
- **10 Model Selection and Management** [May 20]
- **11 Model Debugging, Fairness, and Explainability** [Jun 03]
- **12 Model Serving Systems and Techniques** [Jun 10]

- **13 Q&A and Exam Preparation**

Programming Projects

- **Open Source Projects**
  - Programming project in context of open source projects
    - **Apache SystemDS:** https://github.com/apache/systemds
    - **DAPHNE:** https://daphne-eu.eu/
      (private repo but OSS release end 03/2022)
    - Other OSS projects possible, but harder to merge PRs
  - Commitment to **open source and open communication** (PRs, mailing list)
  - **Remark:** Don't be afraid to ask questions / develop code in public
- **Objectives**
  - Non-trivial feature in an ML system (**2 ECTS → 50 hours**)
  - **OSS processes:** Break down into subtasks, code/tests/docs, PR per project, code review, incorporate review comments, etc
- **Team**
  - Individuals or up to **three-person teams** (w/ separated responsibilities)

Alternative Exercise

- **Task: ML Pipeline** [to be published Mar 11]
  - Given the **TBD** dataset(s) (e.g., UCI/Kaggle Wine or others)
  - **Data Prep:** Setup train/test/validation splits; perform data validation, data augmentation, feature engineering
  - **Modeling:** Compare multiple baseline models using an **OSS ML system**
  - **Tuning:** hyper-parameter tuning and cross validation
  - **Parallelization:** parallelize your ML pipeline (at least the tuning part)
  - **Debugging:** Perform model debugging and investigate explainability

- **Objectives**
  - End-to-end development of an ML pipeline on real data
  - Handle data issues, under-specified objectives, model training and debugging

- **Team**
  - Individuals or up to **three-person teams** (w/ separated responsibilities)

Apache SystemDS:
A Declarative ML System for the End-to-End Data Science Lifecycle

Background and System Architecture

https://github.com/apache/systemds

Landscape of ML Systems

- **Existing ML Systems**
  - **#1 Numerical computing** frameworks
  - **#2 ML Algorithm libraries** (local, large-scale)
  - **#3 Linear algebra ML systems** (large-scale)
  - **#4 Deep neural network** (DNN) frameworks
  - **#5 Model management, and deployment**

- **Exploratory Data-Science Lifecycle**
  - **Open-ended problems** w/ underspecified objectives
  - Hypotheses, data integration, run analytics
  - **Unknown value** → lack of system infrastructure
    → **Redundancy of manual efforts and computation**

"Take these datasets and show value or competitive advantage"

[NIPS 2015]
[DEBull 2018]

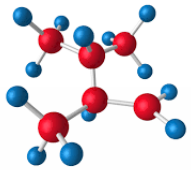
- **Data Preparation Problem**
  - **80% Argument:** 80-90% time for finding, integrating, cleaning data
  - Diversity of tools ➔ boundary crossing, lack of optimization

The Data Science Lifecycle

Data-centric View:
Application perspective
Workload perspective
System perspective

Data extraction, schema alignment, entity resolution, data validation, data cleaning, outlier detection, missing value imputation, semantic type detection, data augmentation, feature selection, feature engineering, feature transformations

Data Scientist



Data Integration
Data Cleaning
Data Preparation

Model Selection
Training
Hyper-parameters

Validate & Debug
Deployment
Scoring & Feedback

Data/SW Engineer

Exploratory Process
(experimentation, refinements, ML pipelines)

DevOps Engineer

Key observation: SotA data integration/cleaning based on ML

Example: Linear Regression Conjugate Gradient

Note:
#1 Data Independence
#2 Implementation-Agnostic Operations

```
1:  X = read($1); # n x m matrix
2:  y = read($2); # n x 1 vector
3:  maxi = 50; lambda = 0.001;
4:  intercept = $3;
5:  ...
6:  r = -(t(X) %*% y);
7:  norm_r2 = sum(r * r); p = -r;
8:  w = matrix(0, ncol(X), 1); i = 0;
9:  while(i<maxi & norm_r2>norm_r2_trgt)
10: {
11:    q = (t(X) %*% (X %*% p))+lambda*p;
12:    alpha = norm_r2 / sum(p * q);
13:    w = w + alpha * p;
14:    old_norm_r2 = norm_r2;
15:    r = r + alpha * q;
16:    norm_r2 = sum(r * r);
17:    beta = norm_r2 / old_norm_r2;
18:    p = -r + beta * p; i = i + 1;
19: }
20: write(w, $4, format="text");
```

Read matrices from HDFS/S3

Compute initial gradient

Compute conjugate gradient

Compute step size

Update model and residuals

➔ **"Separation of Concerns"**

Apache SystemML/SystemDS

APIs: Command line, JMLC, Spark MLContext, Spark ML, (20+ Scalable Algorithms)

DML Scripts

Language

Compiler

Runtime

Write Once, Run Anywhere

[SIGMOD'15,'17,'19,'21abc]
[PVLDB'14,'16ab,'18]
[ICDE'11,'12,'15]
[CIDR'17,'20]
[VLDBJ'18]
[DEBull'14]
[PPoPP'15]

Apache SystemML™

07/2020 Renamed to SystemDS
05/2017 Apache Top-Level Project
11/2015 Apache Incubator Project
08/2015 Open Source Release

In-Memory Single Node
(scale-up)

Hadoop or Spark Cluster
(scale-out)

In-Progress:

GPU

since 2014/16

Java C++

since 2012

hadoop

since 2010/11

APACHE Spark

since 2015

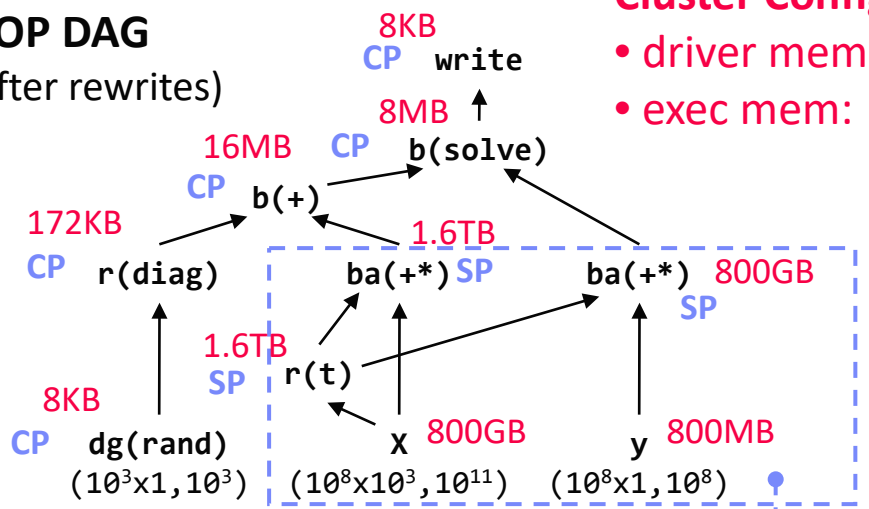
Basic HOP and LOP DAG Compilation

LinregDS (Direct Solve)

```
X = read($1);
y = read($2);
intercept = $3;
lambda = 0.001;
...
if( intercept == 1 ) {
  ones = matrix(1, nrow(X), 1);
  X = append(X, ones);
}
I = matrix(1, ncol(X), 1);
A = t(X) %*% X + diag(I)*lambda;
b = t(X) %*% y;
beta = solve(A, b);
...
write(beta, $4);
```

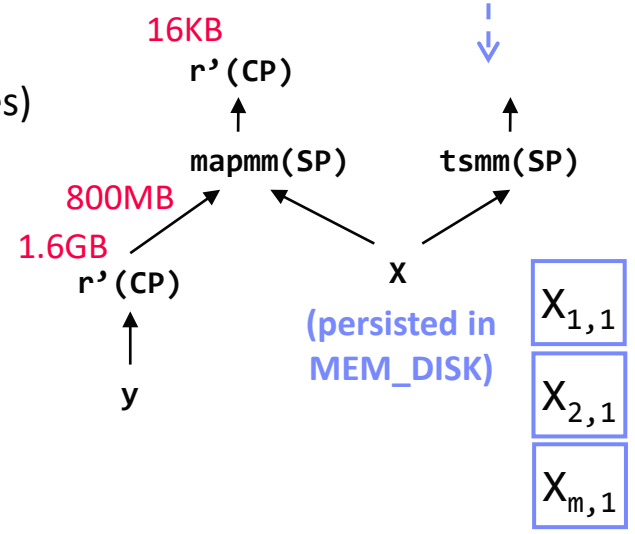
Scenario:
X: 10^8 x 10^3, 10^{11}
y: 10^8 x 1, 10^8

HOP DAG
(after rewrites)



Cluster Config:
- driver mem: 20 GB
- exec mem: 60 GB

LOP DAG
(after rewrites)



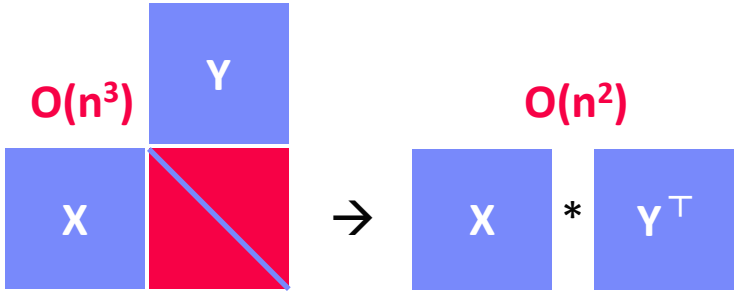
➔ **Hybrid Runtime Plans:**
- Size propagation / memory estimates
- Integrated CP / Spark runtime
- Dynamic recompilation during runtime

➔ **Distributed Matrices**
- Fixed-size (squared) matrix blocks
- Data-parallel operations

Static and Dynamic Rewrites

- **Example Static Rewrites** (size-indep.)
  - Common Subexpression Elimination
  - Constant Folding / Branch Removal / Block Sequence Merge
  - Static Simplification Rewrites
  - Right/Left Indexing Vectorization
  - For Loop Vectorization
  - Spark checkpoint/repartition injection

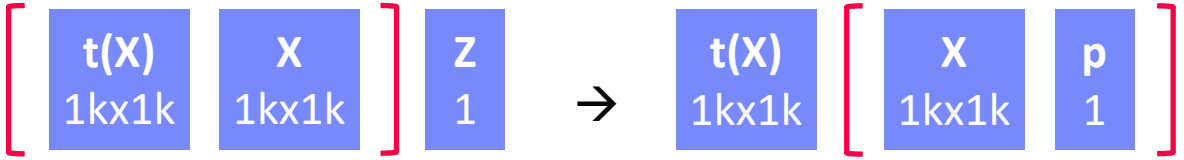
`trace(X%*%Y)` → `sum(X*t(Y))`



`sum(λ*X)` → `λ*sum(X)`
`sum(X+Y)` → `sum(X)+sum(Y)`

- **Example Dynamic Rewrites** (size-dep.)
  - Dynamic Simplification Rewrites
  - Matrix Mult Chain Optimization

`rowSums(X)` → `X, iff ncol(X)=1`
`sum(X^2)` → `X%*%t(X), iff ncol(X)=1`



2,002 MFLOPs

4 MFLOPs

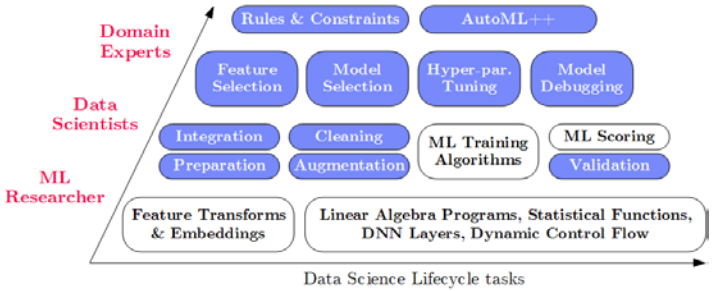
Size propagation and sparsity estimation

Apache SystemDS Design

Apache SystemML (since 2010)
→ SystemDS (09/2018)
→ **Apache SystemDS** (07/2020)

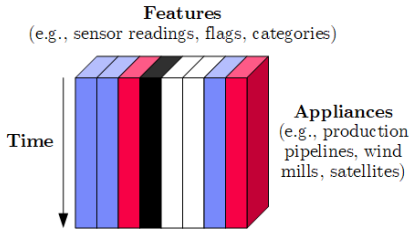
- **Objectives**
  - Effective and efficient **data preparation, ML, and model debugging at scale**
  - High-level abstractions for different lifecycle tasks and users

- **#1 Based on DSL for ML Training/Scoring**
  - Hierarchy of **abstractions for DS tasks**
  - ML-based SotA, interleaved, performance



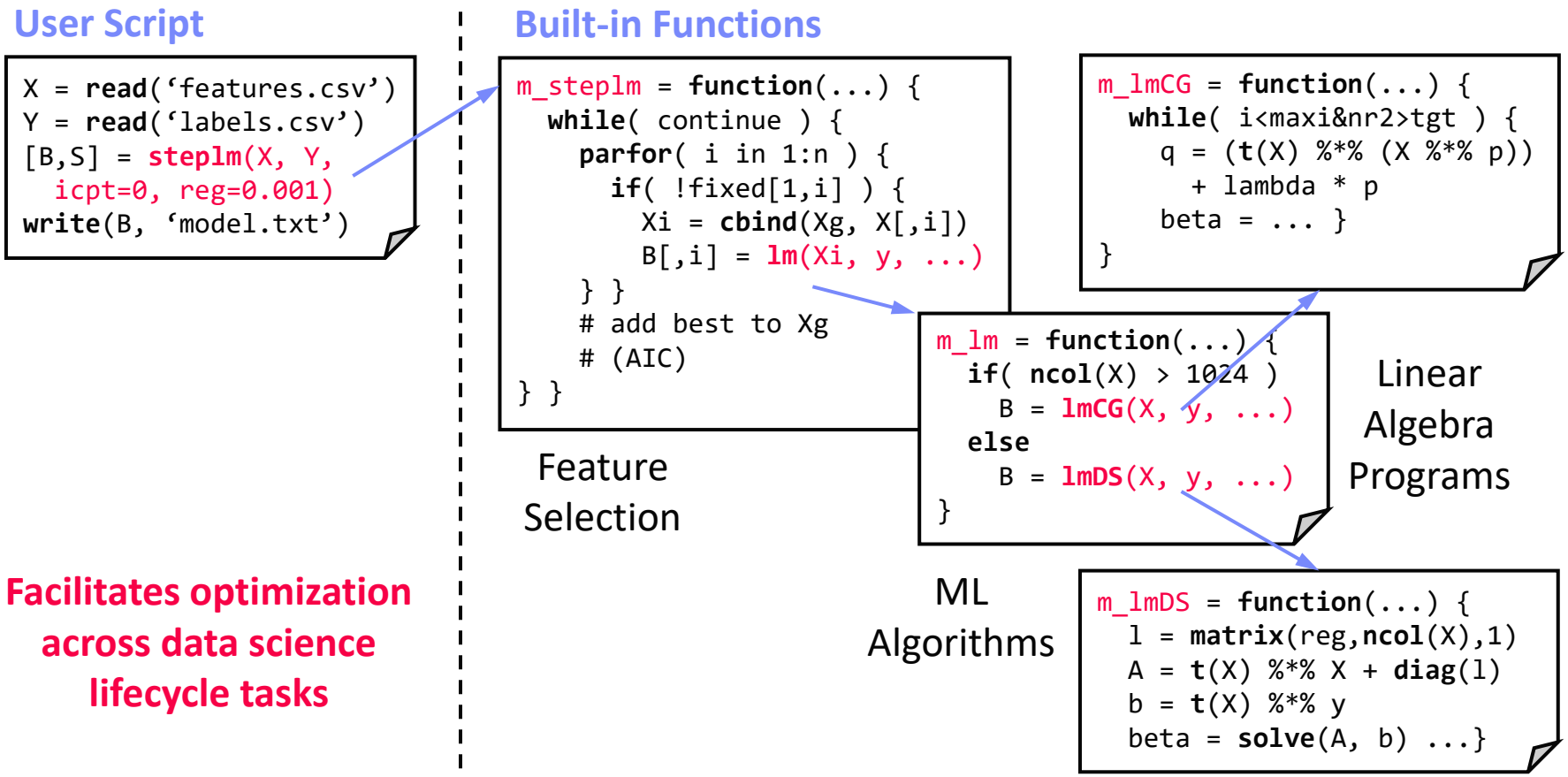
- **#2 Hybrid Runtime Plans and Optimizing Compiler**
  - System infrastructure for diversity of algorithm classes
  - Different parallelization strategies and new architectures (**Federated ML**)
  - Abstractions → redundancy → automatic optimization

- **#3 Data Model: Heterogeneous Tensors**
  - Data integration/prep requires **generic data model**



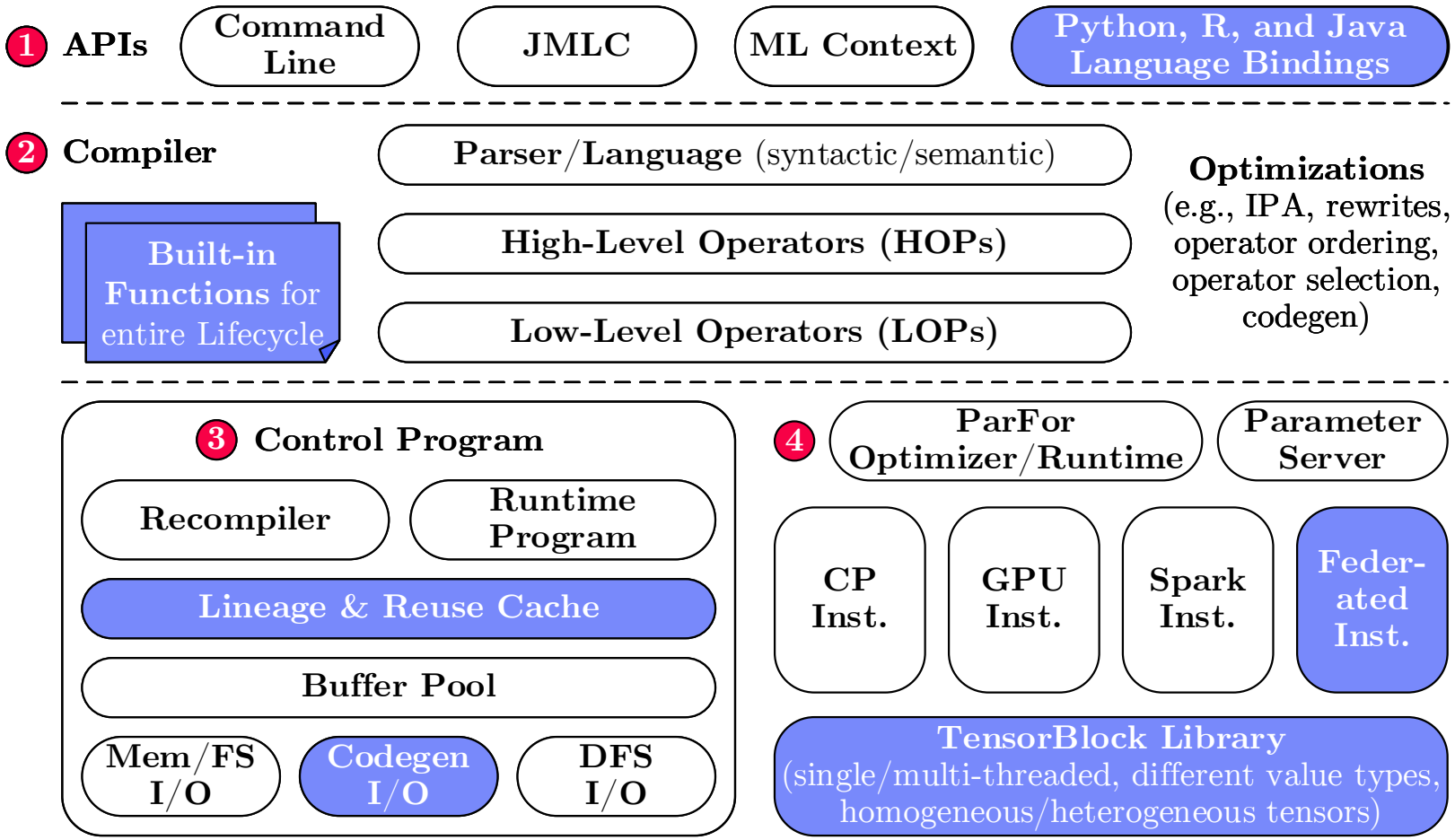
Language Abstractions and APIs, cont.

- **Example: Stepwise Linear Regression**



Apache SystemDS Architecture

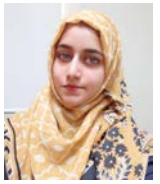
> 83,400 tests
> 8,500 DSL tests



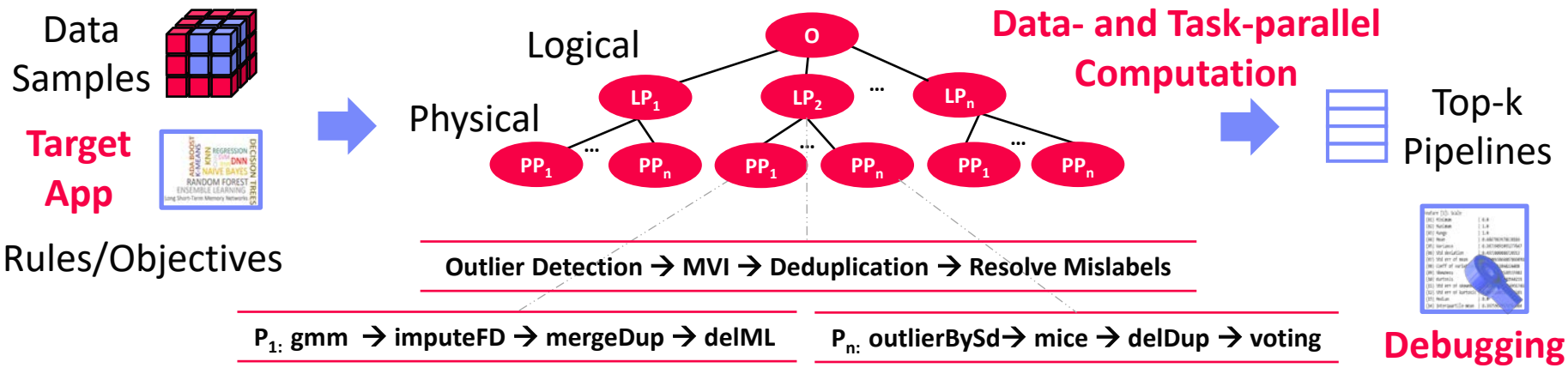
[M. Boehm, I. Antonov, S. Baunsgaard, M. Dokter, R. Ginthör, K. Innerebner, F. Klezin, S. N. Lindstaedt, A. Phani, B. Rath, B. Reinwald, S. Siddiqui, S. Benjamin Wrede: SystemDS: A Declarative Machine Learning System for the End-to-End Data Science Lifecycle. **CIDR 2020**]

Data Cleaning Pipelines

[WIP] **WashHouse**:
Data Cleaning Benchmark



- **Automatic Generation of Cleaning Pipelines**
  - Library of robust, parameterized **data cleaning primitives**
  - **Enumeration of DAGs** of primitives & **hyper-parameter optimization** (HB, BO)



University	Country
TU Graz	Austria
TU Graz	Austria
TU Graz	Germany
IIT	India
IIT	IIT
IIT	Pakistan
IIT	India
SIBA	Pakistan
SIBA	null
SIBA	null

Dirty Data

University	Country
TU Graz	Austria
TU Graz	Austria
TU Graz	Austria
IIT	India
IIT	India
IIT	India
IIT	India
SIBA	Pakistan
SIBA	Pakistan
SIBA	Pakistan

After **imputeFD(0.5)**

A	B	C	D
0.77	0.80	1	1
0.96	0.12	1	1
0.66	0.09	null	1
0.23	0.04	17	1
0.91	0.02	17	null
0.21	0.38	17	1
0.31	null	17	1
0.75	0.21	20	1
null	null	20	1
0.19	0.61	20	1
0.64	0.31	20	1

Dirty Data

A	B	C	D
0.77	0.80	1	1
0.96	0.12	1	1
0.66	0.09	17	1
0.23	0.04	17	1
0.91	0.02	17	1
0.21	0.38	17	1
0.31	0.29	17	1
0.75	0.21	20	1
0.41	0.24	20	1
0.19	0.61	20	1
0.64	0.31	20	1

After **MICE**

SliceLine for Model Debugging

[SIGMOD'21c]

[**Credit:** sliceline, Silicon Valley, HBO]

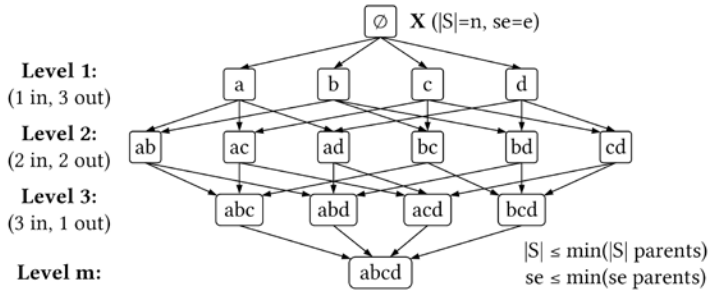
- **Problem Formulation**
  - Intuitive slice scoring function
  - Exact top-k slice finding
  - $|S| \geq \sigma \wedge sc(S) > 0$
  - $\alpha \in (0,1]$

$$sc = \alpha\left(\frac{\bar{e}(S)}{\bar{e}(X)} - 1\right) - (1-\alpha)\left(\frac{|X|}{|S|} - 1\right)$$

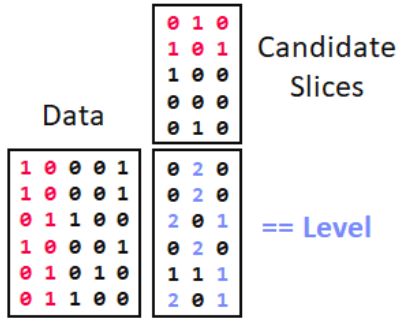
$$= \alpha\left(\frac{|X|}{|S|} \cdot \frac{\sum_{i=1}^{|S|} es_i}{\sum_{i=1}^{|X|} e_i} - 1\right) - (1-\alpha)\left(\frac{|X|}{|S|} - 1\right)$$

slice error — slice size

- **Properties & Pruning**
  - Monotonicity of slice sizes, errors
  - Upper bound sizes/errors/scores → pruning & termination



- **Linear-Algebra-based Slice Finding**
  - Recoded matrix **X**, error vector e
  - Vectorized implementation in linear algebra (join & eval via sparse-sparse matrix multiply)
  - Local and distributed task/data-parallel execution



Multi-Level Lineage Tracing & Reuse

[SIGMOD'21a]

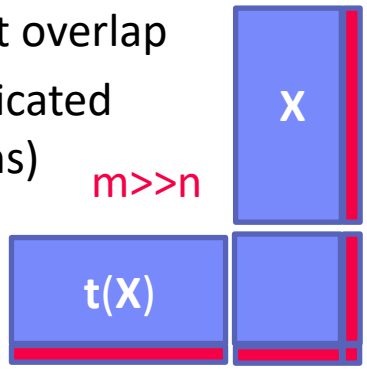

- **Lineage as Key Enabling Technique**
  - Trace lineage of operations (incl. non-determinism), dedup for loops/functions
  - Model versioning, data reuse, incremental maintenance, autodiff, debugging

- **Full Reuse of Intermediates**
  - Before executing instruction, probe output lineage in cache `Map<Lineage, MatrixBlock>`
  - Cost-based/heuristic caching and eviction decisions (compiler-assisted)

```
for( i in 1:numModels )
  R[,i] = lm(X, y, lambda[i,], ...)
```

```
m_lmDS = function(...) {
  l = matrix(reg,ncol(X),1)
  A = t(X) %*% X + diag(l)
  b = t(X) %*% y
  beta = solve(A, b) ...}
```

- **Partial Reuse of Intermediates**
  - **Problem:** Often partial result overlap
  - Reuse partial results via dedicated rewrites (compensation plans)
  - Example: steplm

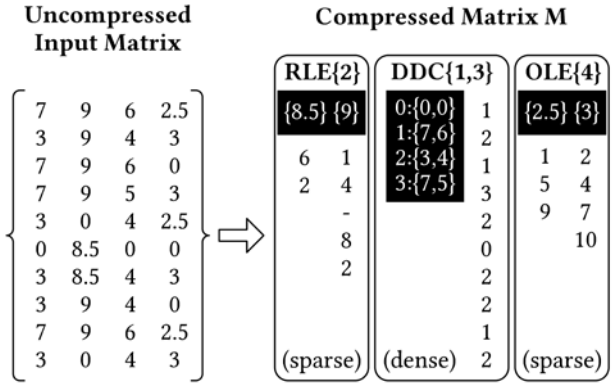



```
m_steplm = function(...) {
  while( continue ) {
    parfor( i in 1:n ) {
      if( !fixed[1,i] ) {
        Xi = cbind(Xg, X[,i])
        B[,i] = lm(Xi, y, ...)
    } }
    # add best to Xg
    # (AIC)
} }
```

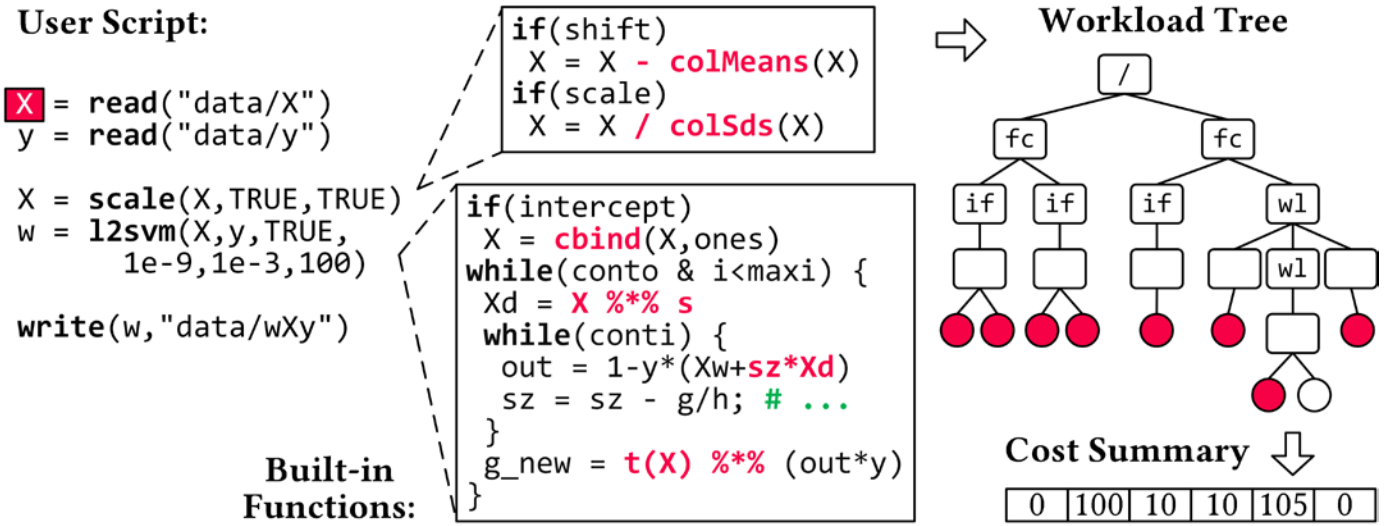
Compressed Linear Algebra Extended

[under submission]

- **Lossless Matrix Compression**
  - **Improved general applicability** (compression time, new compression schemes, new kernels, intermediates, workload-aware)
  - Sparsity → **Redundancy exploitation** (data redundancy, structural redundancy)



- **Workload-aware Compression**
  - Workload summary → compression
  - Compression → execution planning

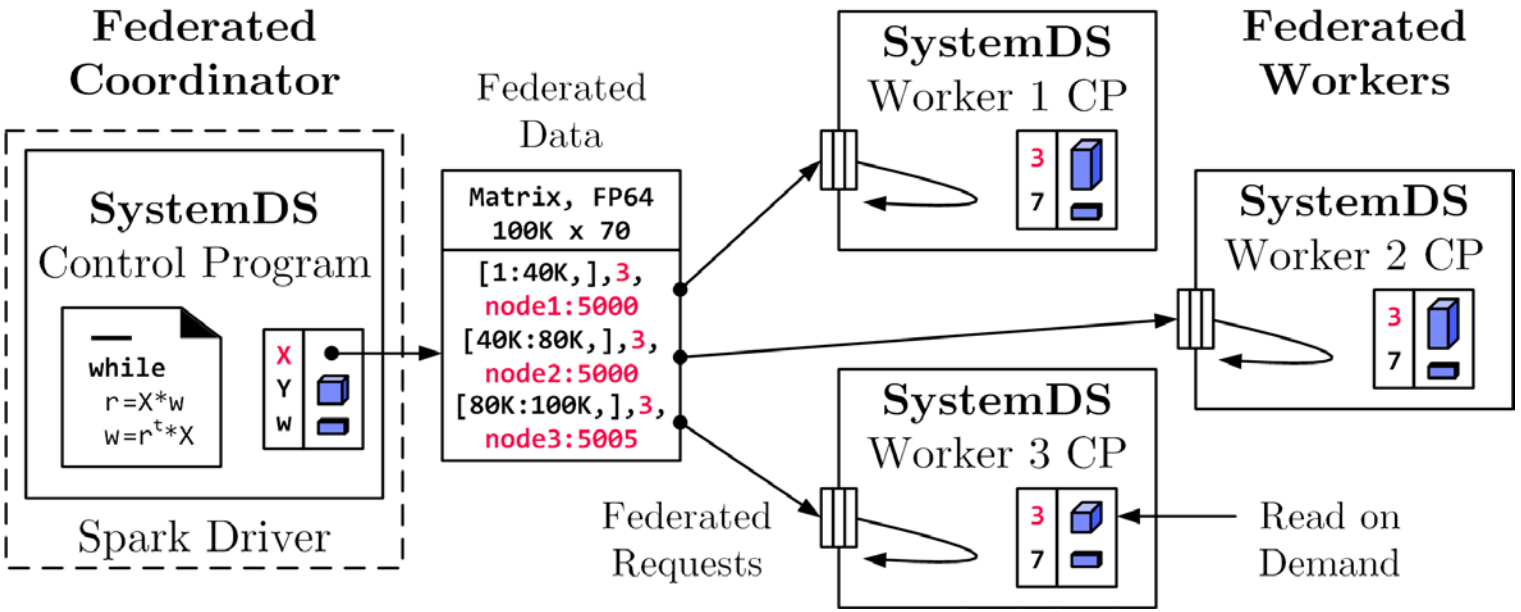


Federated Learning

[SIGMOD 2021b]

- **Federated Backend**
  - **Federated data** (matrices/frames) as meta data objects
  - **Federated linear algebra**, (and **federated parameter server**)

```
X = federated(addresses=list(node1, node2, node3),
  ranges=list(list(0,0), list(40K,70), ..., list(80K,0), list(100K,70)));
```



- **Federated Requests:** `READ, PUT, GET, EXEC_INST, EXEC_UDF, CLEAR`

Integrated **D**ata **A**nalysis **P**ipelines for Large-scale Data Management, **H**PC, and Machi**ne** Learning; DAPHNE daughter of river god Peneus (fountains, streams), chased by Apollo

[Louvre, Paris]

The DAPHNE project is funded by the European Union's Horizon 2020 research and innovation program under grant agreement number 957407 for the time period from Dec/2020 through Nov/2024.

DAPHNE

An Open and Extensible System Infrastructure for Integrated Data Analysis Pipelines

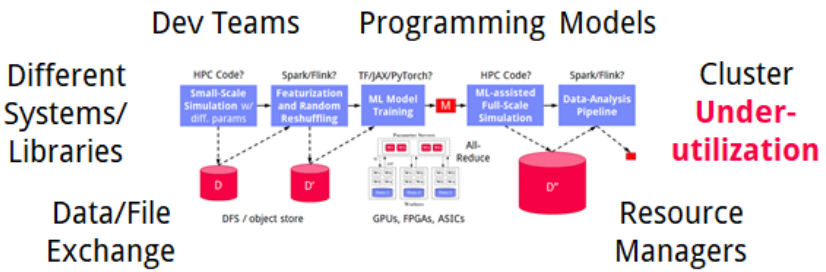
https://daphne-eu.eu/

➔ **DAPHNE Overall Objective: Open and extensible system infrastructure**

Motivation

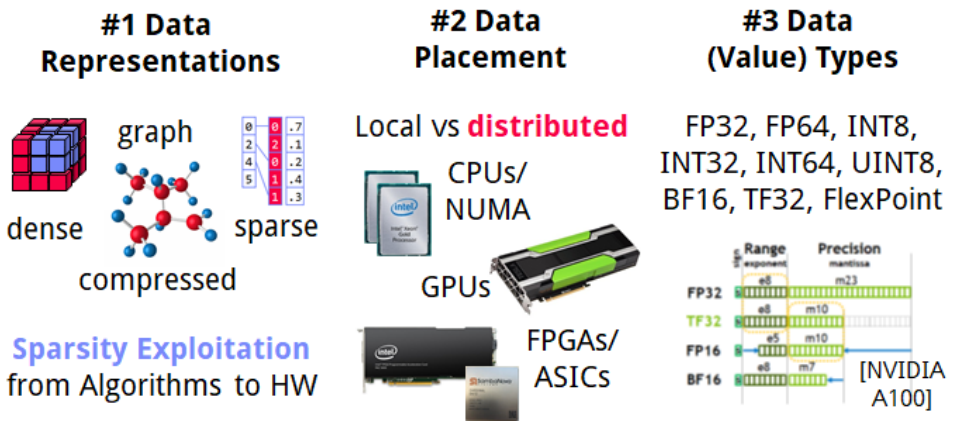
- **Integrated Data Analysis Pipelines**
  - Open data formats, query processing
  - Data preprocessing and cleaning
  - ML model training and scoring
  - HPC, custom codes, and simulations

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**

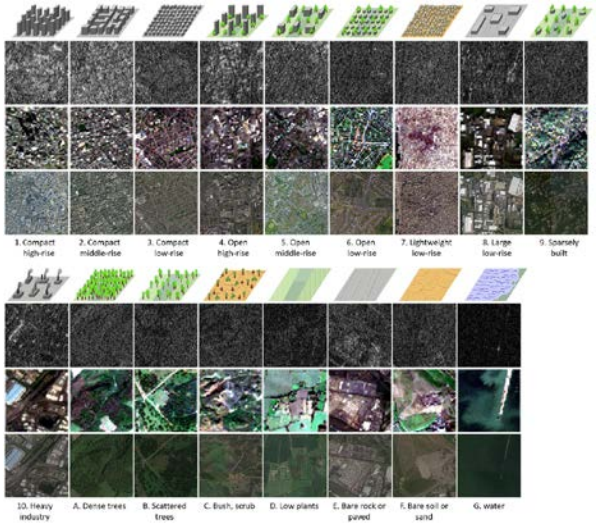


DAPHNE Use Cases

- **DLR Earth Observation**
  - **ESA Sentinel-1/2** datasets → 4PB/year
  - Training of local climate zone classifiers on **So2Sat LCZ42** (15 experts, 400K instances, 10 labels each, 85% confidence, ~55GB H5)
  - **ML pipeline:** preprocessing, ResNet18, climate models

[Xiao Xiang Zhu et al: So2Sat LCZ42: A Benchmark Dataset for the Classification of Global Local Climate Zones. **GRSM 2020**]

[So2Sat LC42 Dataset https://mediatum.ub.tum.de/1454690]

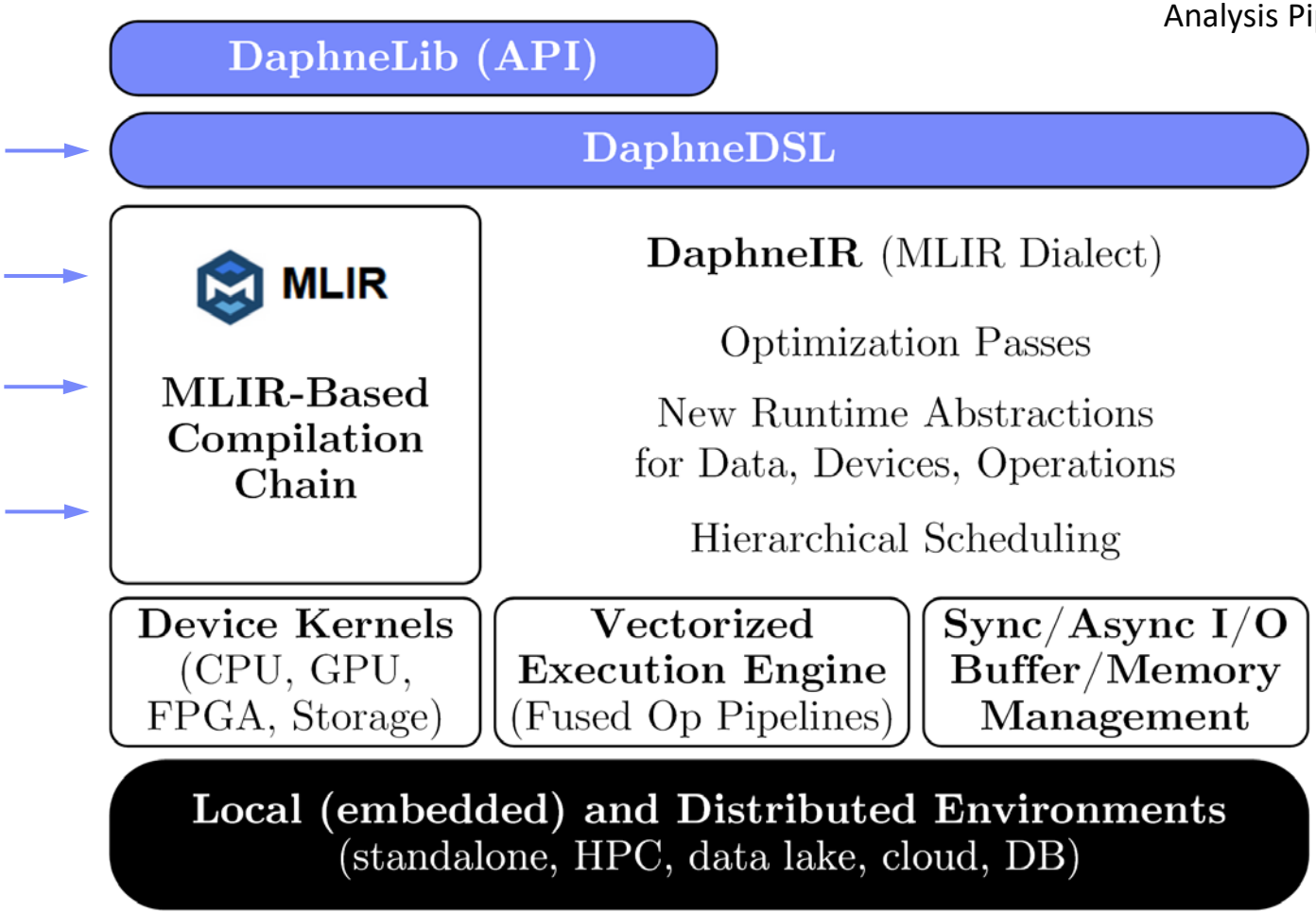


- **IFAT Semiconductor Ion Beam Tuning**
- **KAI Semiconductor Material Degradation**
- **AVL Vehicle Dev Process** (ejector geometries, KPIs)

- **ML-assisted simulations, data cleaning, augmentation**

DAPHNE System Architecture

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



Extensible Infrastructure

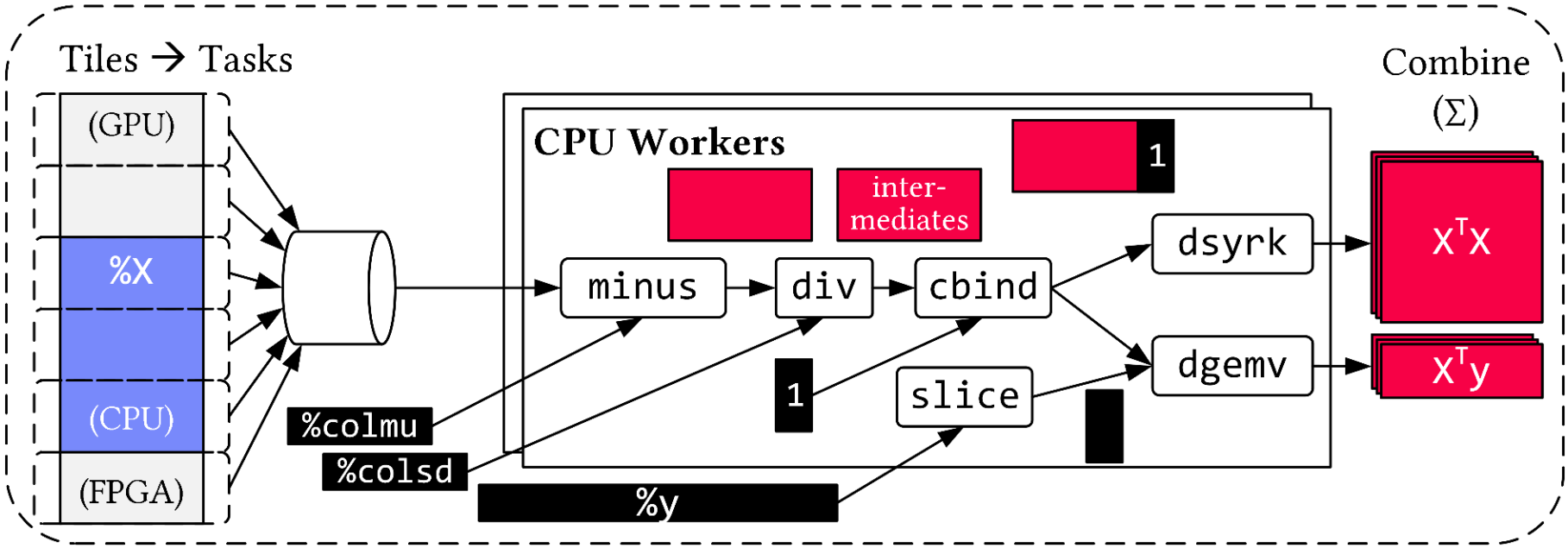
Multi-level Compilation/ Runtime

Fine-grained Fusion and Parallelism

Integration w/ Resource Mgmt & Prog. Models

Vectorized (Tiled) Execution

```
(%9, %10) = fusedPipeline1(%X, %y, %colmu, %colsd) {
```



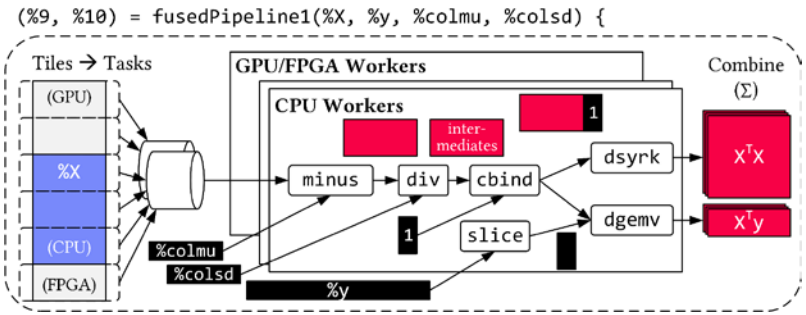
Default Parallelization Frame & Matrix Ops

Locality-aware, Multi-device Scheduling

Fused Operator Pipelines on Tiles/Scalars + Codegen

Vectorized (Tiled) Execution, cont.

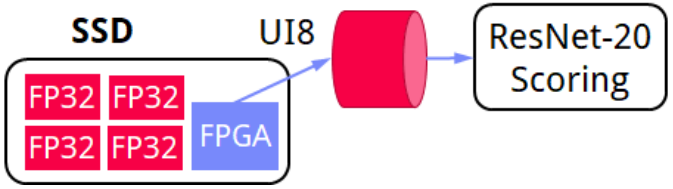
- **#1 Zero-copy Input Slicing**
  - Create view on sliced input (no-op)
  - All kernels work on views



- **#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



Summary & Q&A

Thanks

- **Data Management Group**
- **Motivation and Goals**
- **Course Organization**
- **Course Outline, and Projects**
- **Apache SystemDS and DAPHNE**

Programming Projects in Apache SystemDS, DAPHNE, or Exercise on ML Pipeline

- **Next Lectures** (Part A)
  - **02** Languages, Architectures, and System Landscape [Mar 11] **+ projects**
  - **03** Size Inference, Rewrites, and Operator Selection [Mar 18]
  - **04** Operator Fusion and Runtime Adaptation [Mar 25]
  - **05** Data- and Task-Parallel Execution [Apr 01]
  - **06** Parameter Servers [Apr 08]
  - **07** Hybrid Execution and HW Accelerators [Apr 29]
  - **08** Caching, Partitioning, Indexing and Compression [May 06]