Architecture of ML Systems (AMLS)
01 Introduction and Overview

Prof. Dr. Matthias Boehm
Technische Universität Berlin
Berlin Institute for the Foundations of Learning and Data
Big Data Engineering (DAMS Lab)

Last update: Apr 15, 2023

Announcements / Org

- **#1 Hybrid & Video Recording**
  - Hybrid lectures (in-person, zoom) with optional attendance
    https://tu-berlin.zoom.us/j/69052921909?pwd=Wlp0ekhLdERHQ08vV2lOd0ZPYk5VZz09 (first lecture)
    https://tu-berlin.zoom.us/j/9529634787?pwd=R1ZsN1M3SC9BOU1OcFdmem9zT202UT09 (other lectures)
  - Zoom **video recordings**, links from website
    https://mboehm7.github.io/teaching/ss23_amls/index.htm

- **#2 Course Registrations**
  - TU Berlin: 135 (TUB ISIS registrations)
  - TU Graz: 30? (TUGonline registrations)
  - Bachelor/Master/PhD ratio? CS/other ratio?

~165

Agenda

- FG Big Data Engineering (DAMS Lab)
- Motivation and Goals
- Course Organization and Logistics
- Course Outline, and Projects
- Apache SystemDS and DAPHNE

FG Big Data Engineering (DAMS Lab)

https://www.tu.berlin/dams

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 + research area manager
  - **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 / in-memory indexing & query processing

DB group

FG Big Data Engineering (DAMS Lab) – Team

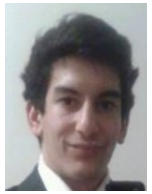
- **Postdoc** (01/2021)
  Patrick Damme

- **PhD Student** (04/2019)
  Arnab Phani

- **PhD Student** (01/2020)
  Sebastian Baunsgaard

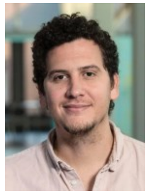
- **PhD Student** (06/2023)
  Christina Dionysio

- **PhD Student** (06/2023)
  Philipp Ortner

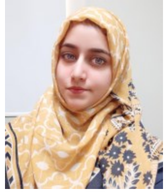


- **PhD Student** (03/2023)
  David Justen
  SAP

- **PhD Student** (02/2023)
  Carlos E. Muniz Cuza
  [visitor Aalborg University]



- **PhD Student** (09/2019)
  Shafaq Siddiqi
  TU Graz



- **PhD Student** (04/2021)
  Saeed Fathollahzadeh
  TU Graz

- **3x Student Assistants**
  N.N. (1x ~06/2023)
- **Bachelor & Master Students**

FG Big Data Engineering (DAMS Lab) – Teaching

Architecture of Database Systems (ADBS, **WS**)
DB system internals + prog. project

Architecture of ML Systems (AMLS, **SoSe**)
ML system internals, data science lifecycle + prog. project

Master

Bachelor

Data Integration and Large-Scale Analysis (DIA, **WiSe**)

Distributed Data Management

SE/PR Large-scale Data Engineering (LDE, **SoSe+WiSe**)

Intro to Scientific Methods (WS)

Supervision of bachelor/master theses
(currently 3+12)

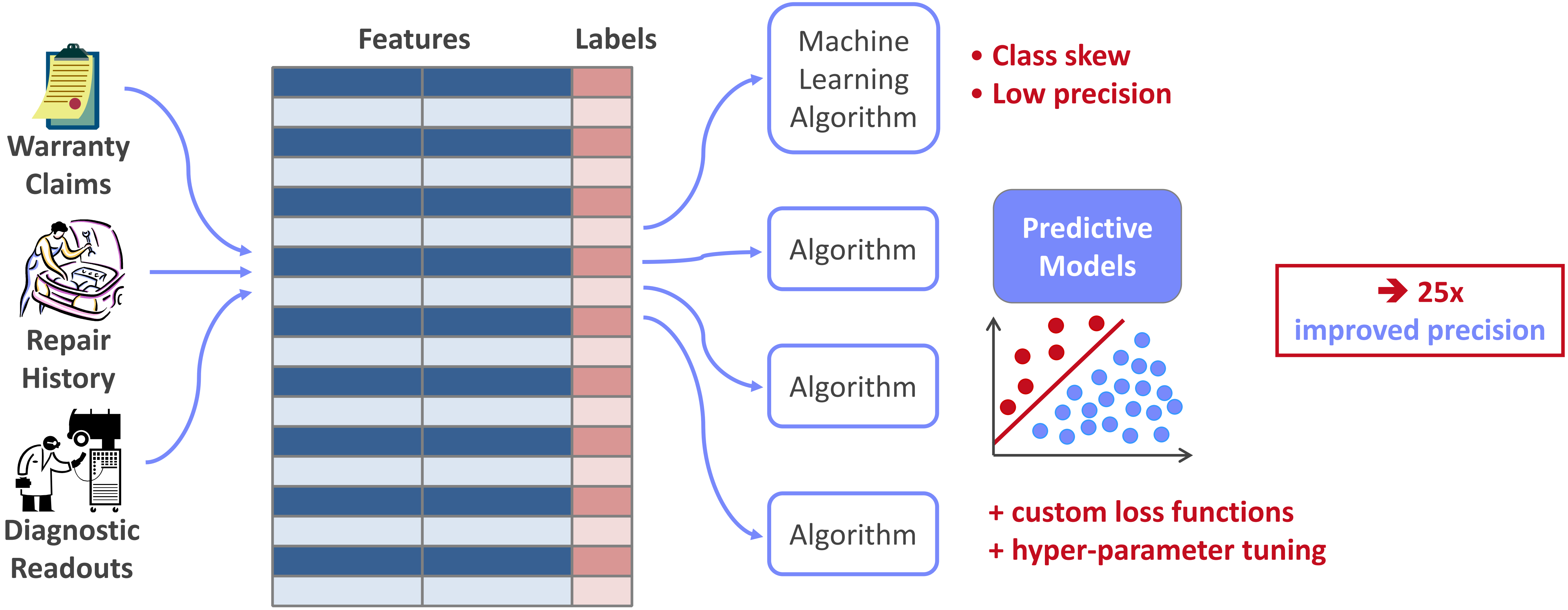
Data Management / Databases (DM, **SS+WS**)
Data management from user/application perspective

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 driving assistance** (blind spot detection, emergency braking, lane centering)
  - **Automotive vehicle development** (ML-assisted simulations, hyper-parameter tuning, ejector optimization)
  - Earth observation and **local climate zone classification** and monitoring, compression
- **Finance**
  - **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)

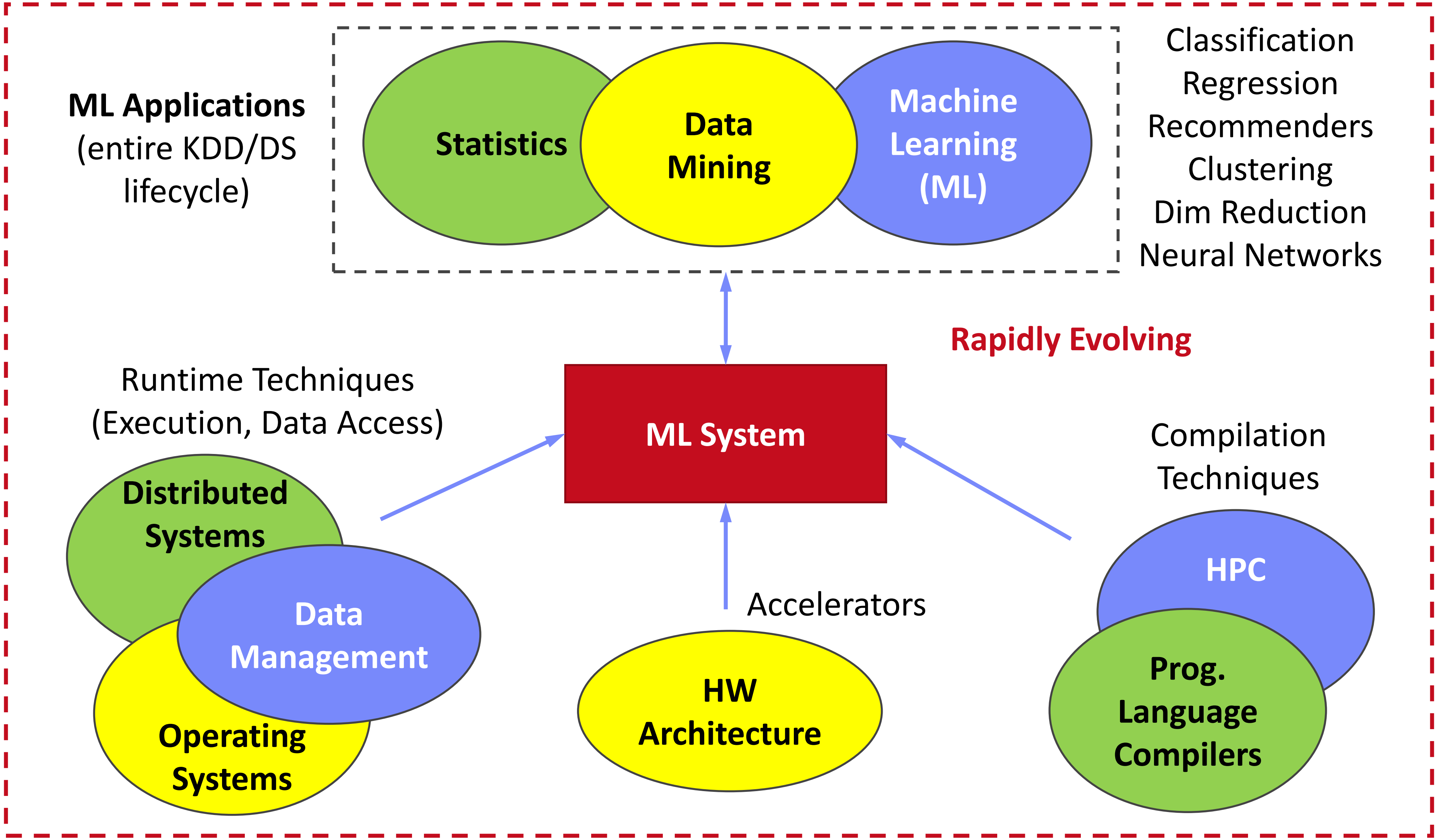
A Car Reacquisition Scenario



Example ML Applications

- **Production/Manufacturing**
  - **Paper and fertilizer production** (regression/classification, anomalies)
  - **Semiconductor manufacturing** (ion beam tuning), and **material degradation** modeling (survival analysis)
  - **Mixed waste stream 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)
- **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; word-embeddings via orthogonalized skip-gram
  - Localized, supervised metric learning (dim reduction and classification)

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 and Logistics

Basic Course Organization



- **Staff**
  - Lecturer: Prof. Dr. Matthias Boehm (replacement: Dr. Patrick Damme)
  - Teaching Assistants: M.Sc. Sebastian Baunsgaard (projects/exercises)
- **Language**
  - Lectures and slides: **English**
  - Communication and examination: **English**/**German**
- **Course Format**
  - TU Berlin: VL+UE 3+2 SWS, **6 ECTS** (3x 1 ECTS + 2x 1.5 ECTS), master / TU Graz: VU 3 SWS, 5 ECTS
  - **Weekly lectures** (**start 4pm**, including **Q&A**), **attendance optional**
  - **Mandatory programming project** or exercises (~3 ECTS)
  - **Recommended papers** for additional reading on your own
- **Prerequisites (preferred)**
  - Basic courses data Management/databases, distributed systems, applied ML
  - Completed bachelor (not mandatory)

Course Logistics

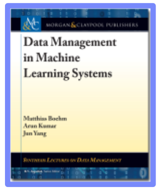
- **Website**
  - https://mboehm7.github.io/teaching/ss23_amls/index.htm
  - All course material (lecture slides) and dates
- **Video Recording / Live Streaming Lectures (zoom)**
- **Communication**
  - **Informal language** (first name is fine)
  - Please, **immediate feedback** (unclear content, missing background)
  - **ISIS forum** for offline questions, after lecture, and via email/PR discussions
  - **Office hours:** by appointment or after lecture
- **Exam**
  - **Completed project / exercise** (checked by me/staff, **no plagiarism** incl *-GPT)
  - **Final oral exam** (written exam or delegated oral exams, if >70 students take the exam)
  - **Grading** (project/exercises completion, 100% exam)

Course Logistics, cont.

- **Course Applicability TU Berlin**
  - Master programs computer engineering, computer science, electrical engineering, information systems management
    - Area Data and Software Engineering
    - Area Cognitive Systems
    - Area Distributed Systems
- **Course Applicability TU Graz** (remote)
  - 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**

Outline and Projects

Created SoSe 2019, 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 SoSe 2020 – SoSe 2023

Part A: Overview and ML System Internals

- **01 Introduction and Overview** [Apr 20]
- **02 Languages, Architectures, and System Landscape** [Apr 27]

- **03 Size Inference, Rewrites, and Operator Selection** [May 04]
- **04 Operator Fusion and Runtime Adaptation** [May 11]
- **05 Data- and Task-Parallel Execution** [May 25]
- **06 Parameter Servers** [Jun 01]

- **07 Hybrid Execution and HW Accelerators** [Jun 08]
- **08 Caching, Partitioning, Indexing, and Compression** [Jun 15]

Part B: ML Lifecycle Systems

- **09 Data Acquisition, Cleaning, and Preparation** [Jun 22]
- **10 Model Selection and Management** [Jun 29]
- **11 Model Debugging, Fairness, and Explainability** [Jul 06]
- **12 Model Serving Systems and Techniques** [Jul 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://github.com/daphne-eu/daphne
    - 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

Apache SystemML™

DAPHNE

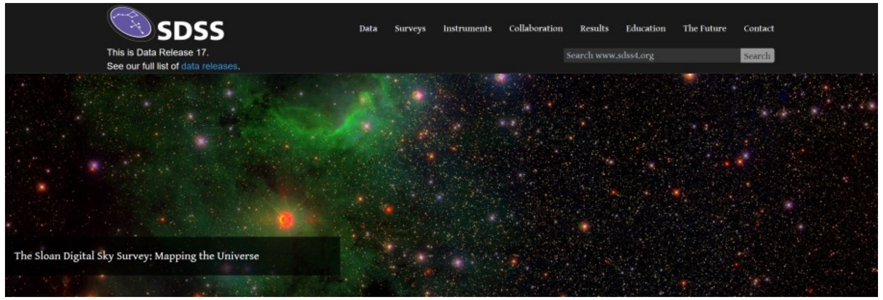
Lists of projects available on website and discussed in 2nd lecture

- **Objectives**
  - Non-trivial feature in an ML system (**3 ECTS → 75-85 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 **Apr 26**]
  - Given the **sky-survey** dataset (SDSS), prepared at https://github.com/damslab/datasets
  - **Data Prep:** Setup train/test/validation splits; perform data validation, data augmentation, feature engineering
  - **Modeling:** Train/compare 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



[https://www.sdss4.org/]

- **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 and DAPHNE

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

https://github.com/apache/systemds

Landscape of ML Systems

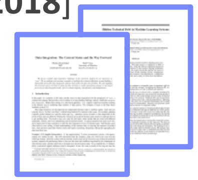
- **Existing ML Systems**
  - **#1 Numerical computing** frameworks
  - **#2 ML Algorithm libraries** (local, large-scale)
  - **#3 Linear algebra ML systems** (large-scale)
  - **#4 Deep neural network** (DNN) frameworks
  - **#5 Model management, and deployment**

- **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**

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

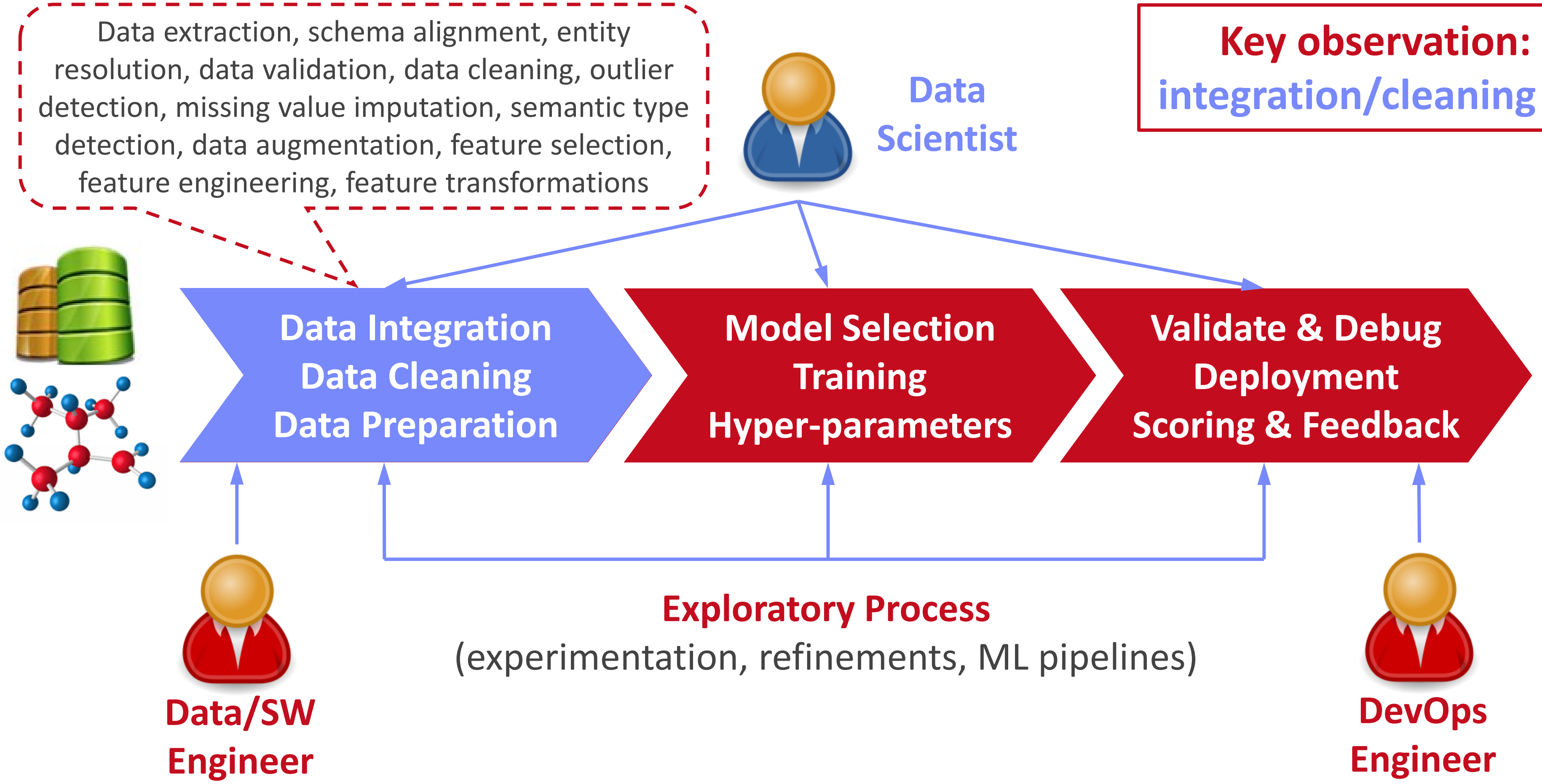
"Take these datasets and show value or competitive advantage"

[NIPS 2015]
[DEBull 2018]

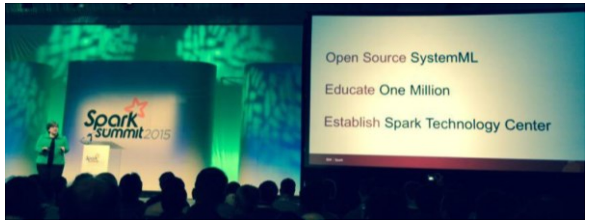


The Data Science Lifecycle
(aka KDD Process, aka CRISP-DM)

Key observation: SotA data integration/cleaning based on ML



Apache SystemDS [https://github.com/apache/systemds]



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

DML Scripts

Language

Compiler

Runtime

Write Once,
Run Anywhere

[SIGMOD'15,'17,'19,'21abc,'23abc]
[PVLDB'14,'16ab,'18,'22]
[ICDE'11,'12,'15]
[CIDR'17,'20]
[VLDBJ'18]
[CIKM'22]
[DEBull'14]
[PPoPP'15]

Apache SystemML™

07/2020 Renamed to **Apache 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)

Federated
(LA progs, PS)

In-Progress:

GPU

since 2014/16

Java

since 2012

hadoop

since 2010/11

Apache Spark

since 2015

exdra

since 2019

Language Abstractions and APIs

Data Independence + Impl-Agnostic Ops
➔ **"Separation of Concerns"**

- **Example:** Stepwise Linear Regression

User Script

```
X = read('features.csv')
Y = read('labels.csv')
[B,S] = steplm(X, Y,
  icpt=0, reg=0.001)
write(B, 'model.txt')
```

Facilitates optimization across data science lifecycle tasks

Built-in Functions

```
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)
} }
```

Feature Selection

```
m_lm = function(...) {
  if( ncol(X) > 1024 )
    B = lmCG(X, y, ...)
  else
    B = lmDS(X, y, ...)
}
```

ML Algorithms

```
m_lmCG = function(...) {
  while( i<maxi&nr2>tgt ) {
    q = (t(X) %*% (X %*% p))
      + lambda * p
    beta = ... }
}
```

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

Linear Algebra Programs

Basic HOP and LOP DAG Compilation

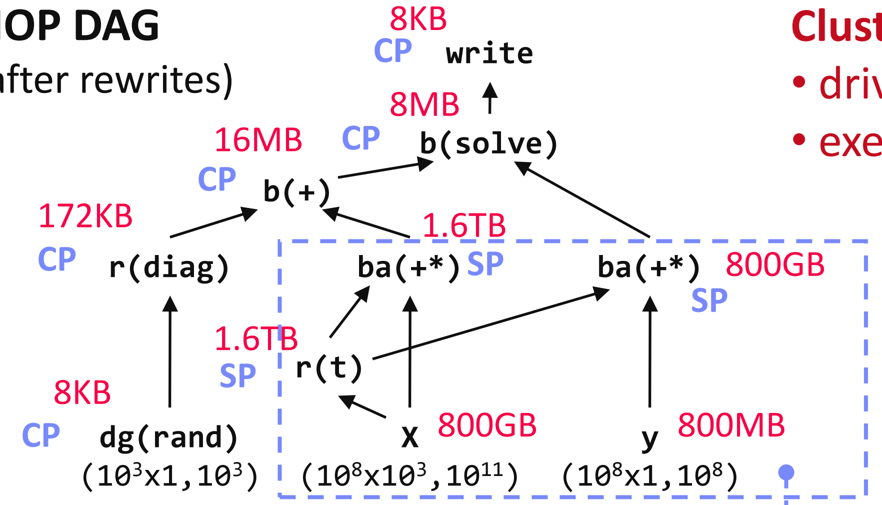
LinregDS (Direct Solve)

```
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: $10^8 \times 10^3$, 10^{11}
y: $10^8 \times 1$, 10^8

HOP DAG

(after rewrites)

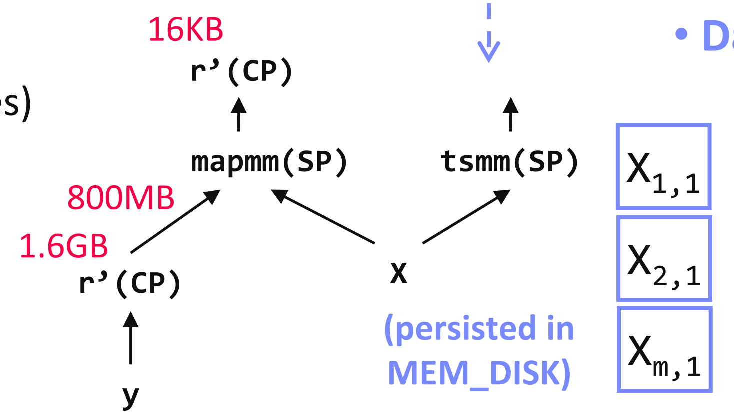


Cluster Config:

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

LOP DAG

(after rewrites)



→ Distributed Matrices

- Fixed-size matrix blocks
- Data-parallel operations

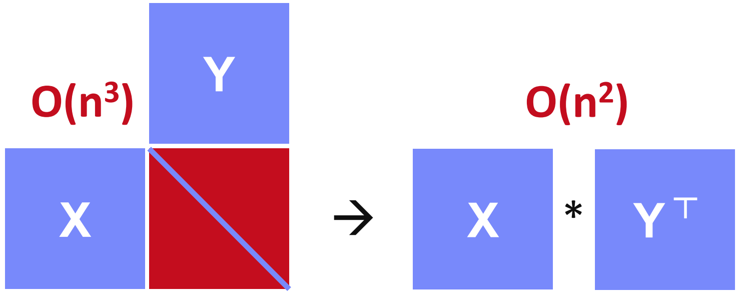
→ Hybrid Runtime Plans:

- Size propagation / memory estimates
- Integrated CP / Spark runtime
- Dynamic recompilation during runtime

Static and Dynamic Rewrites

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

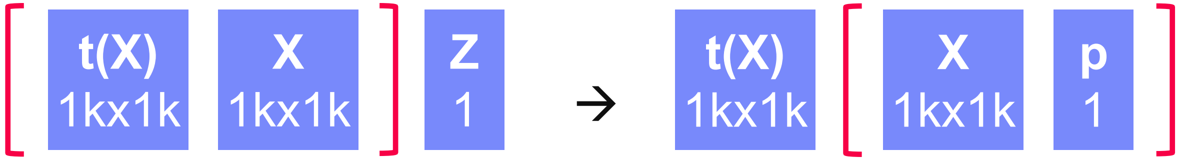
`trace(X%*%Y) → sum(X*t(Y))`



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

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

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



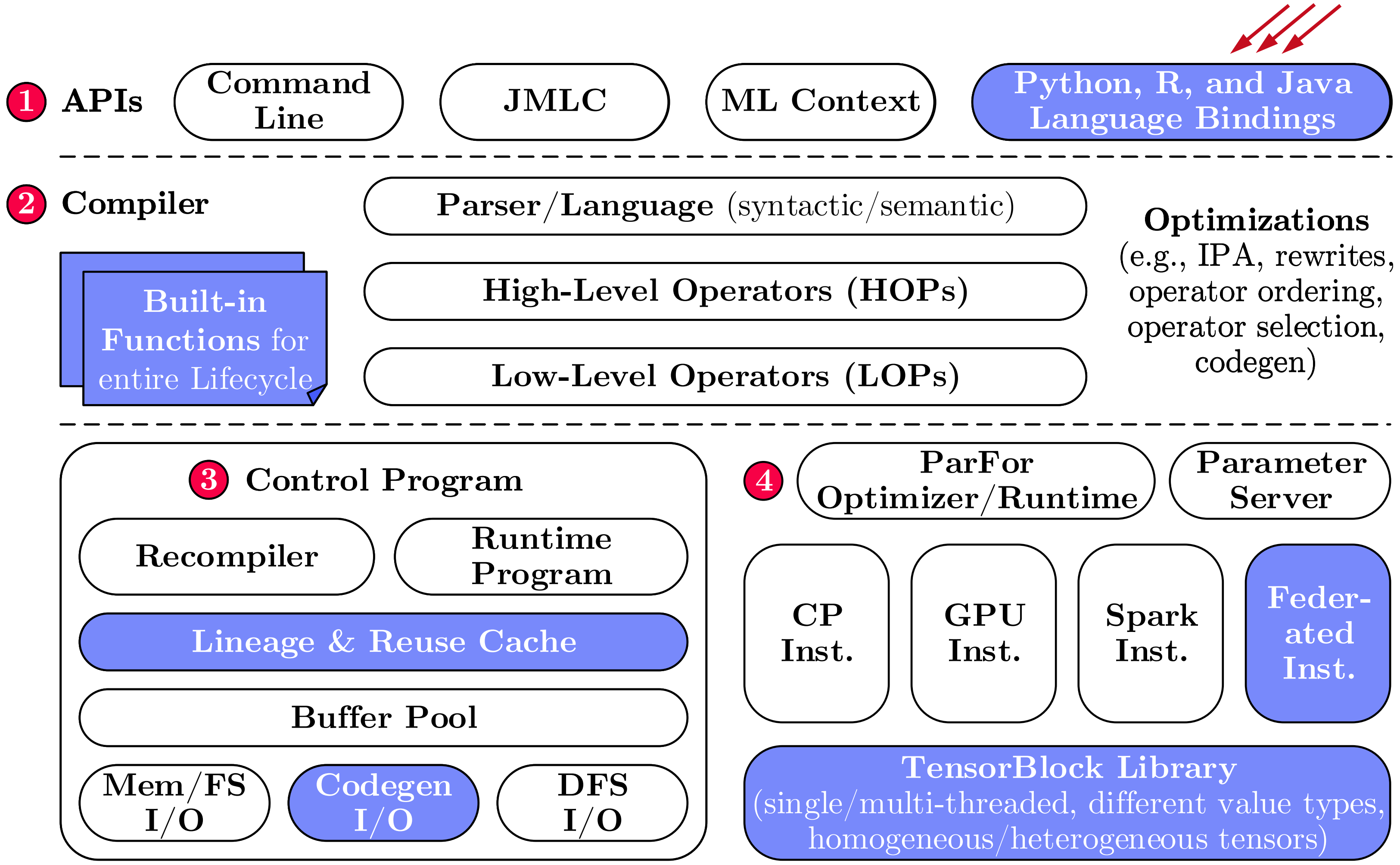
2,002 MFLOPs **4 MFLOPs**

Size propagation and sparsity estimation

Sparsity Estimation & Sparse DP Enum [**SIGMOD'19**]

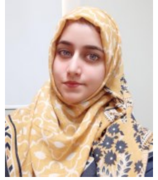
Apache SystemDS Architecture [CIDR'20]

> 125,200 tests
> 8,100 DSL tests



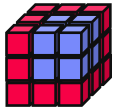
Data Cleaning Pipelines [under submission]

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



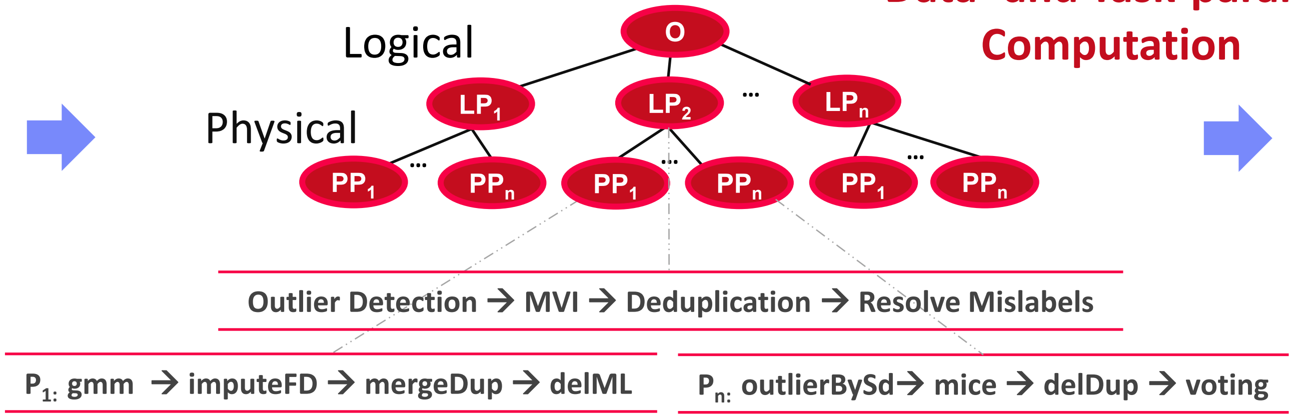
- **Automatic Generation of Cleaning Pipelines**
  - Library of robust, parameterized **data cleaning primitives**
  - **Enumeration of DAGs** of primitives & **hyper-parameter optimization** (evolutionary, HB)

Data Samples



Target App

Rules/Objectives



Data- and Task-parallel Computation

Top-k Pipelines

Debugging

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'21b]

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

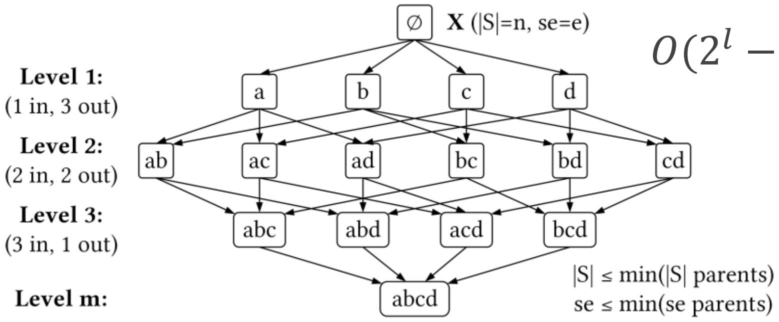
- **Problem Formulation**
  - **Intuitive slice scoring function**
  - Exact **top-k slice finding**
  - $|S| \geq \sigma \wedge sc(S) > 0,\ \alpha \in (0,1]$

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

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

slice error **slice size**

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



$$O(2^l - \sum_{j=1}^{m} 2^{d_j} + l + m)$$

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

Candidate Slices

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

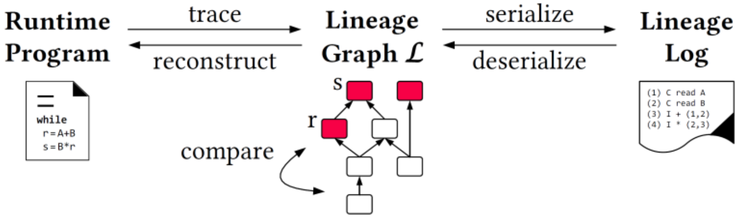
Data

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 2 & 0 \\ 0 & 2 & 0 \\ 2 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 1 & 1 \\ 2 & 0 & 1 \end{bmatrix}$$

== Level

Multi-level Lineage Tracing & Reuse [CIDR'20, SIGMOD'21a]


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



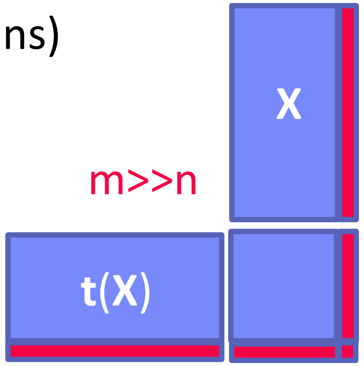
- **Full Reuse of Intermediates**
  - Before executing instruction, probe output lineage in cache `Map<Lineage, MatrixBlock>`
  - Cost-based/heuristic caching and eviction decisions (compiler-assisted)

```
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

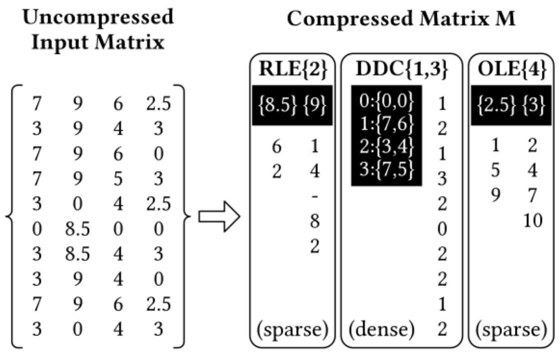
```
m_steplm = function(...) {
  while( continue ) {
    parfor( i in 1:n ) {
      if( !fixed[1,i] ) {
        Xi = cbind(Xg, X[,i])
        B[,i] = lm(Xi, y, ...)
    } }
    # add best to Xg (AIC)
} }
```



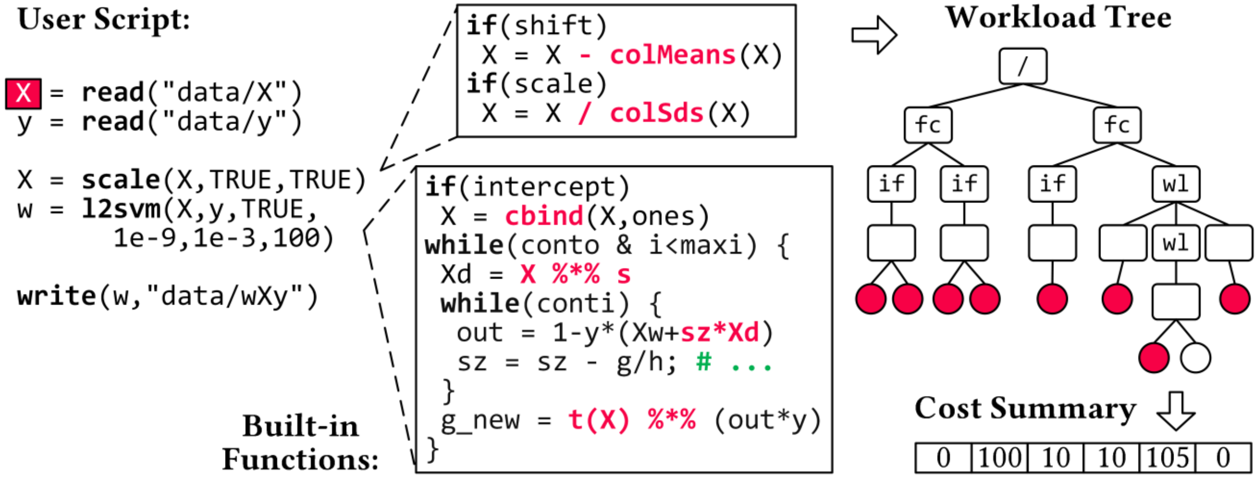
- **Next Steps:** multi-backend, unified mem mgmt

Compressed Linear Algebra Extended [PVLDB'16a, VLDBJ'18, SIGMOD'23a]

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

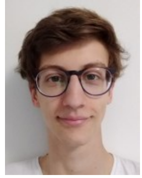
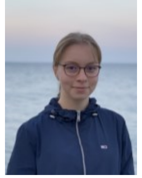
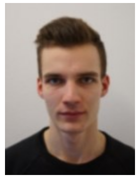


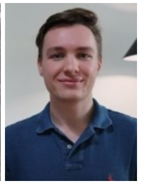
- **Workload-aware Compression**
  - Workload summary → compression
  - Compressed Representation → execution planning



- **Next Steps**
  - Frame compression, compressed I/O
  - Compressed feature transformations
  - Morphing of compressed data

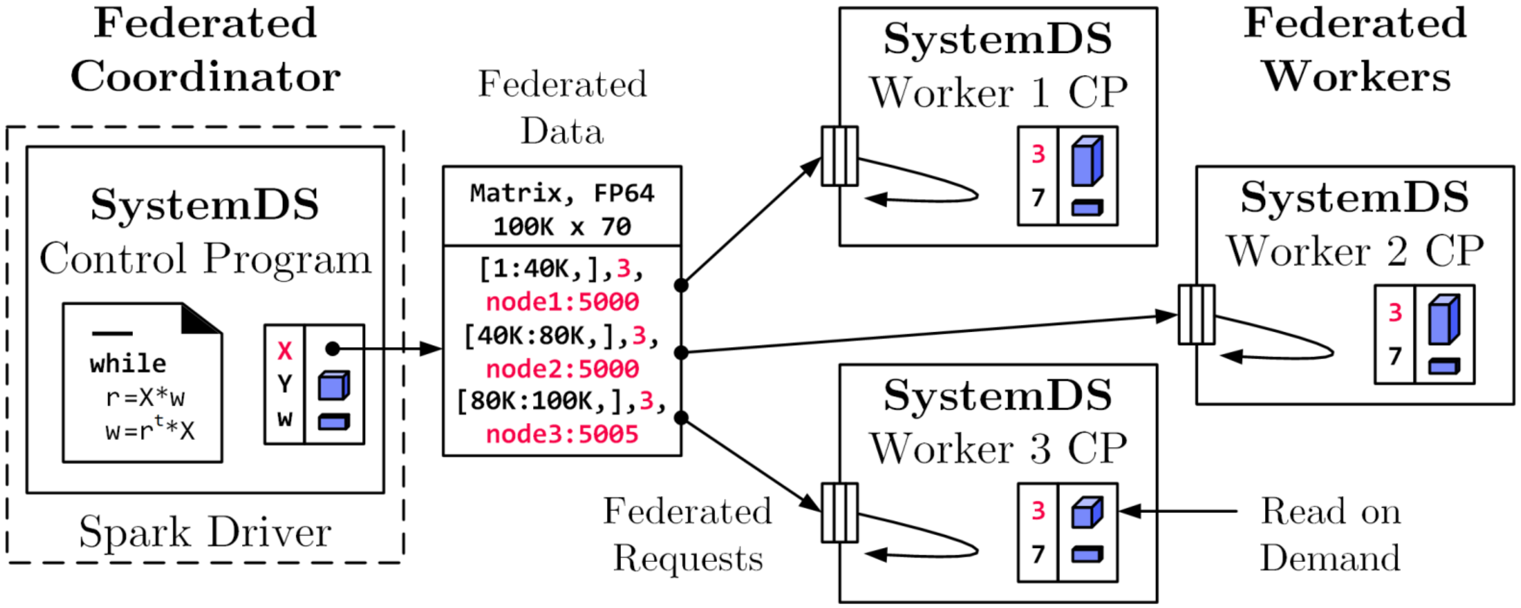
Federated Learning [SIGMOD'21c]



- **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`

➔ **Design Simplicity:**
(1) reuse instructions
(2) federation hierarchies

DAPHNE: An Open and Extensible System Infrastructure for Integrated Data Analysis Pipelines

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

DAPHNE

Motivation

➔ DAPHNE Overall Objective:
Open and extensible system infrastructure

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

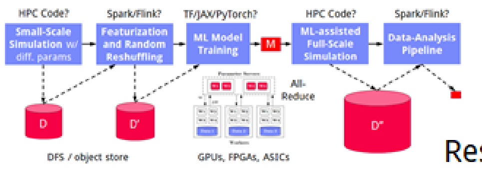
- **Hardware Challenges**
  - DM+ML+HPC share compilation and runtime techniques / converging cluster hardware
  - **End of Dennard scaling:** $P = \alpha\, CFV^2$ (power density 1)
  - **End of Moore's law**
  - **Amdahl's law:** $sp = 1/s$
  - ➔ **Increasing Specialization**

Deployment Challenges

Dev Teams

Programming Models

Different Systems/ Libraries

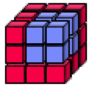


Cluster **Under-utilization**

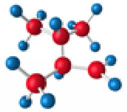
Data/File Exchange

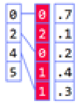
Resource Managers

#1 Data Representations



graph





dense

sparse

compressed

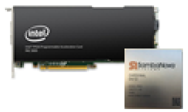
Sparsity Exploitation from Algorithms to HW

#2 Data Placement

Local vs **distributed**

CPUs/ NUMA

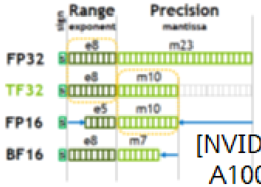
GPUs



FPGAs/ ASICs

#3 Data (Value) Types

FP32, FP64, INT8, INT32, INT64, UINT8, BF16, TF32, FlexPoint

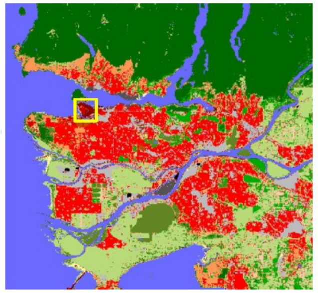


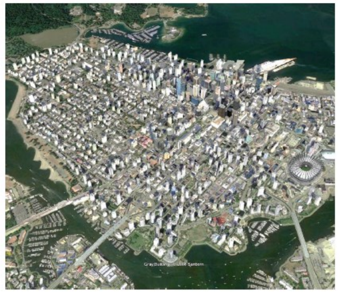
[NVIDIA A100]

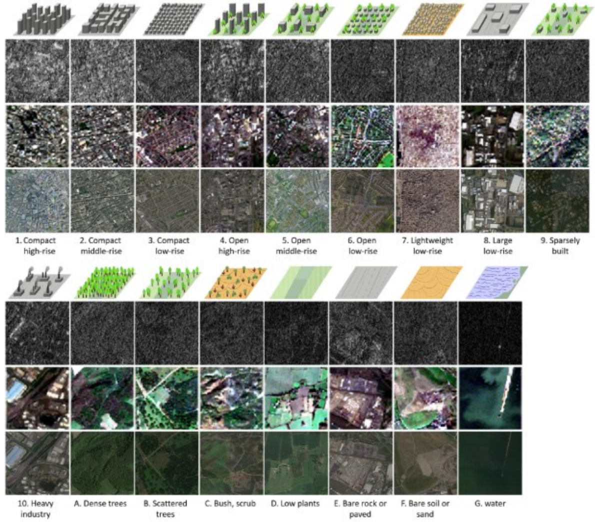
DAPHNE Use Cases

[Xiao Xiang Zhu et al: So2Sat LCZ42: A Benchmark Dataset for the Classification of Global Local Climate Zones. **GRSM 8(3) 2020**]
[So2Sat LC42: https://mediatum.ub.tum.de/1454690]

- **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, ResNet20, climate models



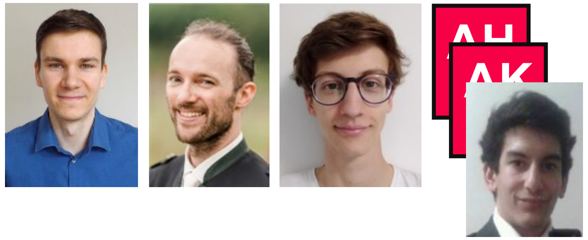




- **IFAT Semiconductor Ion Beam Tuning**
- **KAI Semiconductor Material Degradation**
- **AVL Vehicle Development Process (ejector geometries, KPIs)**

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

DAPHNE System Architecture [CIDR'22]



DaphneLib (API)

Python API w/ lazy evaluation

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

Multi-level Compilation/ Runtime

Fine-grained Fusion and Parallelism

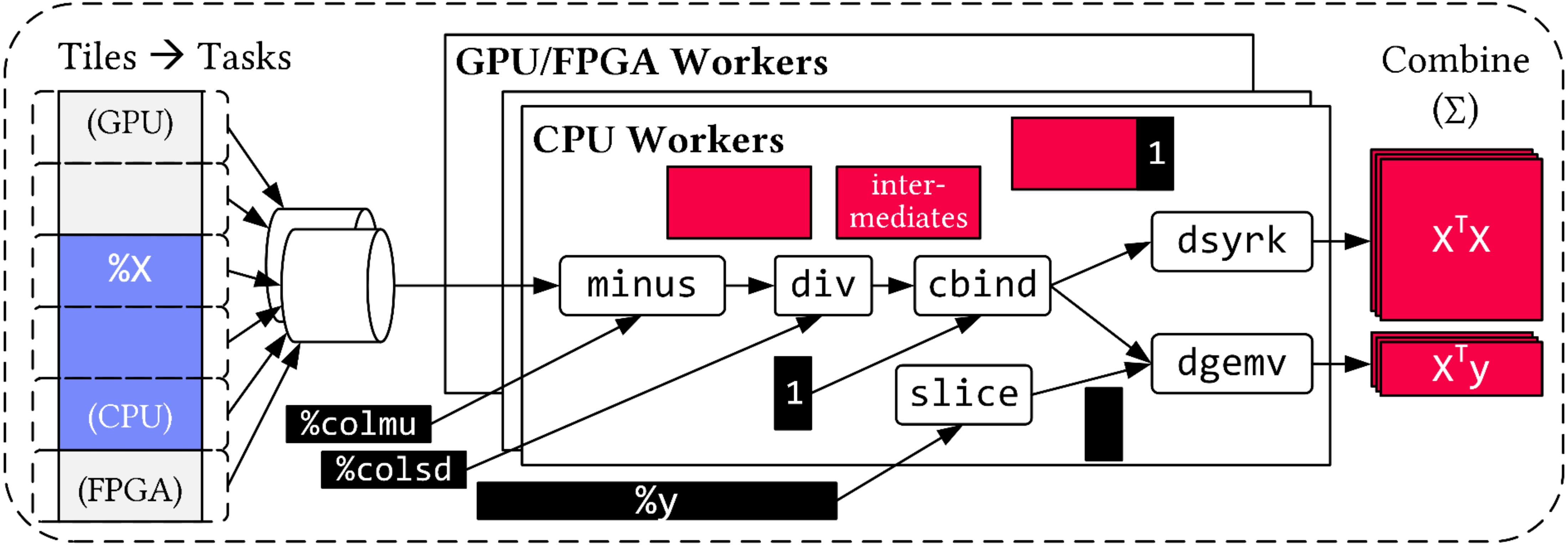
Integration w/ Resource Mgmt & Prog. Models

MLIR Dialects, Extension Catalog (new data types, kernels, scheduling algs)

Sideways Entry, DSL-level constraints (e.g., data/op placement)

Vectorized (Tiled) Execution

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



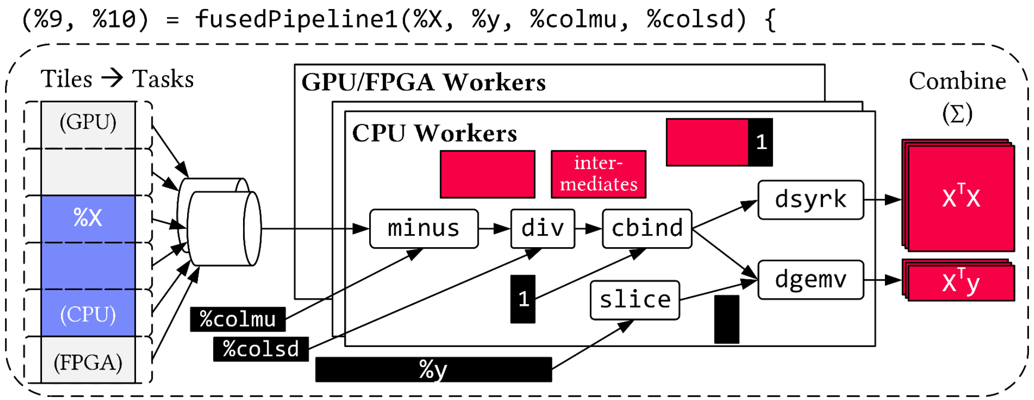
Default Parallelization Frame & Matrix Ops

Locality-aware, Multi-device Scheduling

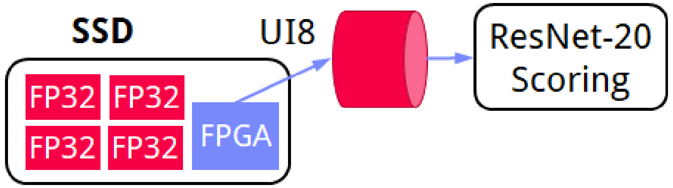
Fused Operator Pipelines on Tiles/Scalars + Codegen

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 & QA

Thanks

- FG Big Data Engineering (DAMS Lab)
- Motivation and Goals
- Course Organization and Logistics
- Course Outline, and Projects
- Apache SystemDS and DAPHNE

→ **Programming Projects** in **Apache SystemDS**, **DAPHNE**, or Exercise on ML Pipelines

- Next Lectures (Part A)
  - **02 Languages, Architectures, and System Landscape** [Apr 27] **+ projects**
  - **03 Size Inference, Rewrites, and Operator Selection** [May 04]
  - **04 Operator Fusion and Runtime Adaptation** [May 11]
  - **05 Data- and Task-Parallel Execution** [May 25]
  - **06 Parameter Servers** [Jun 01]