Project Large-scale Data Engineering (LDE)
Project Kick-off Meeting

Dr.-Ing. Patrick Damme
Technische Universität Berlin
Berlin Institute for the Foundations of Learning and Data
Big Data Engineering (DAMS Lab)

Last update: Oct 13, 2024

Announcements/Org

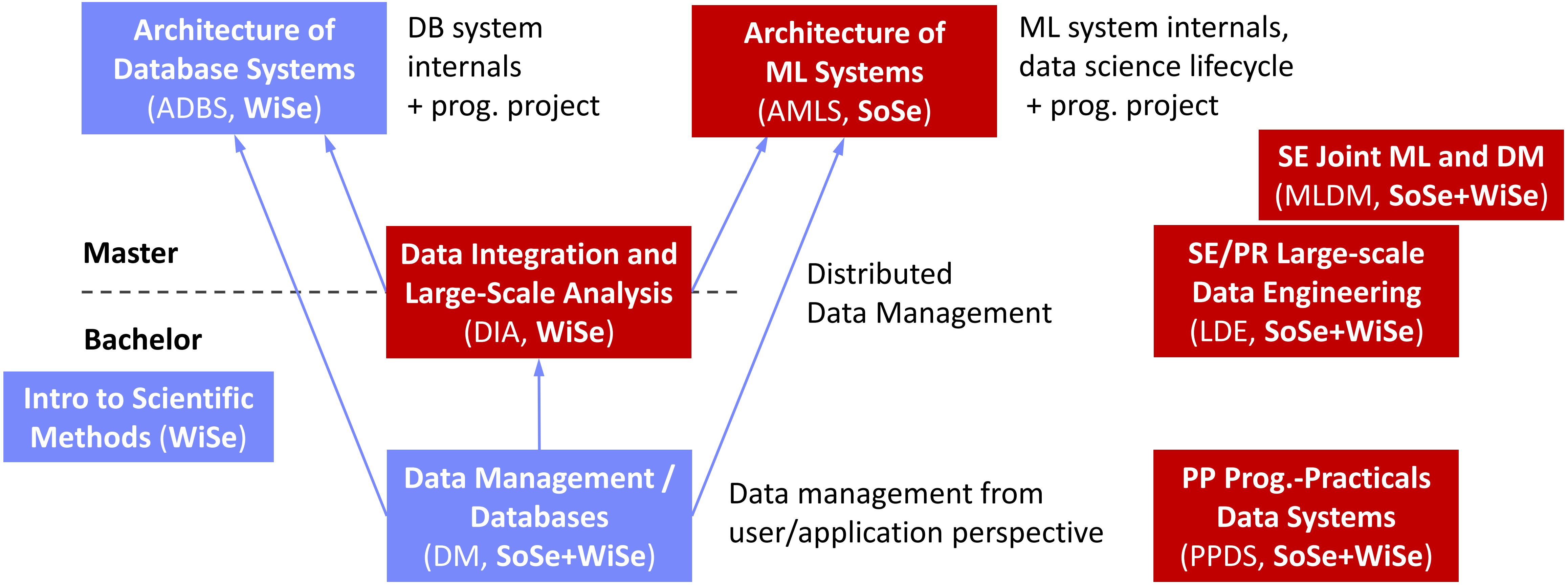
- **Hybrid Setting with Optional Attendance**
  - In-person in MAR 0.015
  - Virtual via zoom
    https://tu-berlin.zoom.us/j/67376691490?pwd=NmlvWTM5VUVWRjU0UGI2bXhBVkxzQT09

About Me

- **Since 10/2022: Postdoc at TU Berlin**, Germany
  - FG Big Data Engineering (DAMS Lab) headed by Prof. Matthias Böhm
  - Continuing work on integrated data analysis pipelines
  - Research interests in the fields of database and ML systems (especially compiler & runtime techniques, extensibility)
- **2021-2022: Postdoc at TU Graz & Know-Center GmbH**, Austria
  - Data Management group headed by Prof. Matthias Böhm
  - Started work on integrated data analysis pipelines
- **2015-2020: PhD student at TU Dresden**, Germany
  - Dresden Database Research Group headed by Prof. Wolfgang Lehner
  - PhD thesis on making complex analytical database queries more efficient through lightweight compression of intermediate results

MorphStore & TVL

FG Big Data Engineering (DAMS Lab) – Teaching



Architecture of Database Systems (ADBS, **WiSe**)

DB system internals
+ prog. project

Architecture of ML Systems (AMLS, **SoSe**)

ML system internals,
data science lifecycle
+ prog. project

SE Joint ML and DM (MLDM, **SoSe+WiSe**)

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 (WiSe)

Data Management / Databases (DM, **SoSe+WiSe**)

Data management from
user/application perspective

PP Prog.-Practicals Data Systems (PPDS, **SoSe+WiSe**)

Agenda

- **Course Organization, Outline, and Deliverables**
- **Apache SystemDS and DAPHNE**
- **List of Project Topics (Proposals)**

Course Organization, Outline, and Deliverables

Large-scale Data Engineering: Module Overview

20+5 seats in total

bachelor + master

#41086: LDE Seminar + Project (12 ECTS) — 12 students

13 students — #41095: Seminar LDE (3)

bachelor-only

#41183: Project LDE (9 ECTS) — 7 students

bachelor-only

Mon, 14:00-16:00
MAR 0.015 & zoom

Seminar LDE

- Reading & writing scientific papers
- Giving presentations on papers
- Summary paper
- Presentation
- Lecturer & seminar mentor

Mon, 16:00-18:00
MAR 0.015 & zoom

Project LDE

- Building & evaluating prototypes
- Giving presentations on prototypes
- Prototype design/impl/tests/doc
- Presentation
- Project mentors

→ In the context of systems for data engineering, data management, machine learning

→ In combination: Ideal preparation for a bachelor/master thesis with our group

Course Organization

- **General Contact Person**
  - Dr.-Ing. Patrick Damme (patrick.damme@tu-berlin.de)
- **Course Website**
  - https://pdamme.github.io/teaching/2024-25_winter/lde/lde_winter2024-25.html
  - One site for seminar and project
  - All material, schedule, **deadlines**
- **ISIS course**
  - https://isis.tu-berlin.de/course/view.php?id=39897
  - Announcements, discussion forum, polls for topic selection
- **Language**
  - Lectures and slides: **English**
  - Communication: **English**/**German**
  - Submitted paper and presentation: **English**
  - **Informal language** (first name is fine), immediate feedback is welcome

Semester Schedule & Deadlines

- **Kick-off Meeting Oct 14** (optional)
- **Recommended Introductory Lecture** (optional)
  - Oct 28, 14:00: Experiments, Reproducibility, and Giving Presentations
- **Self-organized Project Work**
  - Office hours for any questions (optional)
- **Intermediate Presentations** (prerequisite)
  - Dec 16, 16:00-18:00: All teams and individuals
- **Final Presentations** (mandatory)
  - Feb 24, 14:00-18:00: All teams and individuals

- **List of Project Topics**
  - Presented today, take your time to select afterwards
- **Topic Selection**
  - **Deadline: Oct 31, 23:59 CET** (in 2½ weeks)
  - Ranked list of **5 topics** via poll on the ISIS course, plus preferences on individual/team work, plus optionally concrete team members
  - Global topic assignment based on preferences
  - **Notification of assigned topics: Nov 7** (in 3½ weeks)
- **Submission of Impl/Tests/Doc**
  - **Deadline: Feb 17, 23:59 CET** (in 18 weeks)
  - As a pull request on GitHub (exceptionally by email)
- **Submission of Presentation Slides**
  - **Deadline: The day before you present, 23:59 CET**
  - Presentation slides (PDF) to Patrick Damme and project mentor

Project Deliverables

- **Individual/Team Project Work**
  - Teams of up to 3 students **strongly encouraged**
- **Design/Implementation/Tests/Documentation**
  - Get familiar with the given task/problem
  - Develop an initial design for discussion
  - Implement your design, plus tests and docs
  - Conduct experiments and analyze/visualize results
- **Final Presentations**
  - Summarize the problem and your solution (design, implementation, experimental results)
  - **1 student: 10 min talk + 5 min discussion = 15 min**
  - **2 students: 13 min + 7 min = 20 min**
  - **3 students: 16 min + 9 min = 25 min**
  - Audience: engage in the discussion

- **Grading**
  - **#41086 (seminar + project)**
    - Graded portfolio exam
    - 25 pts: summary paper
    - 15 pts: presentation
    - 50 pts: design/impl/tests/doc
    - 10 pts: presentation
  - **#41183 (project-only)**
    - Graded portfolio exam
    - 85 pts: implementation/tests/documentation
    - 15 pts: presentation
- **Academic Honesty / No Plagiarism** (incl LLMs like ChatGPT)

Intermediate Presentations

- **Introduced in Response to Students' Feedback (Course Evaluation)**
- **Expectations**
  - Slide presentation of 5-10 min per individual/team
  - Briefly **motivate the problem** you work on
  - Explain **how you plan to solve it** (conceptually and technically)
  - Outline **your planned experiments**
  - Should be the result of **prior discussions with your project mentor**
- **Benefits for You**
  - Become aware of the **complexity of your project** early on for **improved time management**
  - **Get feedback** by project mentors and other students for improving the **quality of your solution**
- **Prerequisite for the Portfolio Exam**
  - **Ungraded** to create a context where you can make mistakes and learn from them

Portfolio Exam Registration

- **Portfolio exam registration: Nov 04 – Dez 02**
  - Binding registration in Moses/MTS
  - Including selection of seminar presentation date (first-come-first-serve)
- **Portfolio exam de-registration**
  - **Until 3 days before the first graded exam part**
    - Modules “LDE”/”Seminar LDE”: until **Jan 10**
    - Module “Project LDE”: until **Feb 14**
    - De-register yourself in Moses/MTS
  - **With sufficient reason: Until the day of the exam**
    - In case of sickness etc.
    - Modules “LDE”/”Seminar LDE”: until **Jan 12**
    - Module “Project LDE”: until **Feb 16**

- **Missing deadlines/exam without de-registration**
  - Zero points in the respective exam part (!)
  - **Approach us early in case of problems**
- **If you don’t want to take LDE anymore**
  - Let me know asap to give students in the queue a chance to fill in

LDE Project Characteristics

- **Unique Characteristics**
  - Each team/individual gets a different topic
- **Advantages**
  - Topics are real open issues in existing systems
  - Meaningful contributions to open-source systems
  - Your work will be used by others (impact)
  - You earn 9 ECTS (~270 h of work)
  - **~6.75 weeks of full-time work**
- **Practice Open-source Processes**
  - Breakdown into subtasks
  - Code/tests/docs
  - Pull requests
  - Code review
  - Incorporate feedback to improve code

- **Remarks on Topic Descriptions**
  - Lots of open topics to work on in the two systems we develop in our group
  - Initial topic descriptions of varying level of detail
  - If there is interest in a specific topic, we will provide more detailed descriptions with some pointers (please approach project mentors directly)
  - Open to alternative topic proposals

LDE Projects in the Context of Two Open-source Systems

DAPHNE

THE APACHE SOFTWARE FOUNDATION

Apache SystemDS™

- **DAPHNE EU-project**
  https://github.com/daphne-eu/daphne
  - Focus on integrated data analysis pipelines
  - Project implementation in C++ and Python

- **Apache SystemDS**
  https://github.com/apache/systemds
  - Focus on the end-to-end data science lifecycle
  - Project implementation in Java, Python, and DML

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**

DASK
NumPy
julia
R
scikit learn
VOWPAL WABBIT
APACHE Spark
MAHOUT
Apache SystemML™
PYTORCH
mxnet
TensorFlow
mlflow
Keras

- **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**

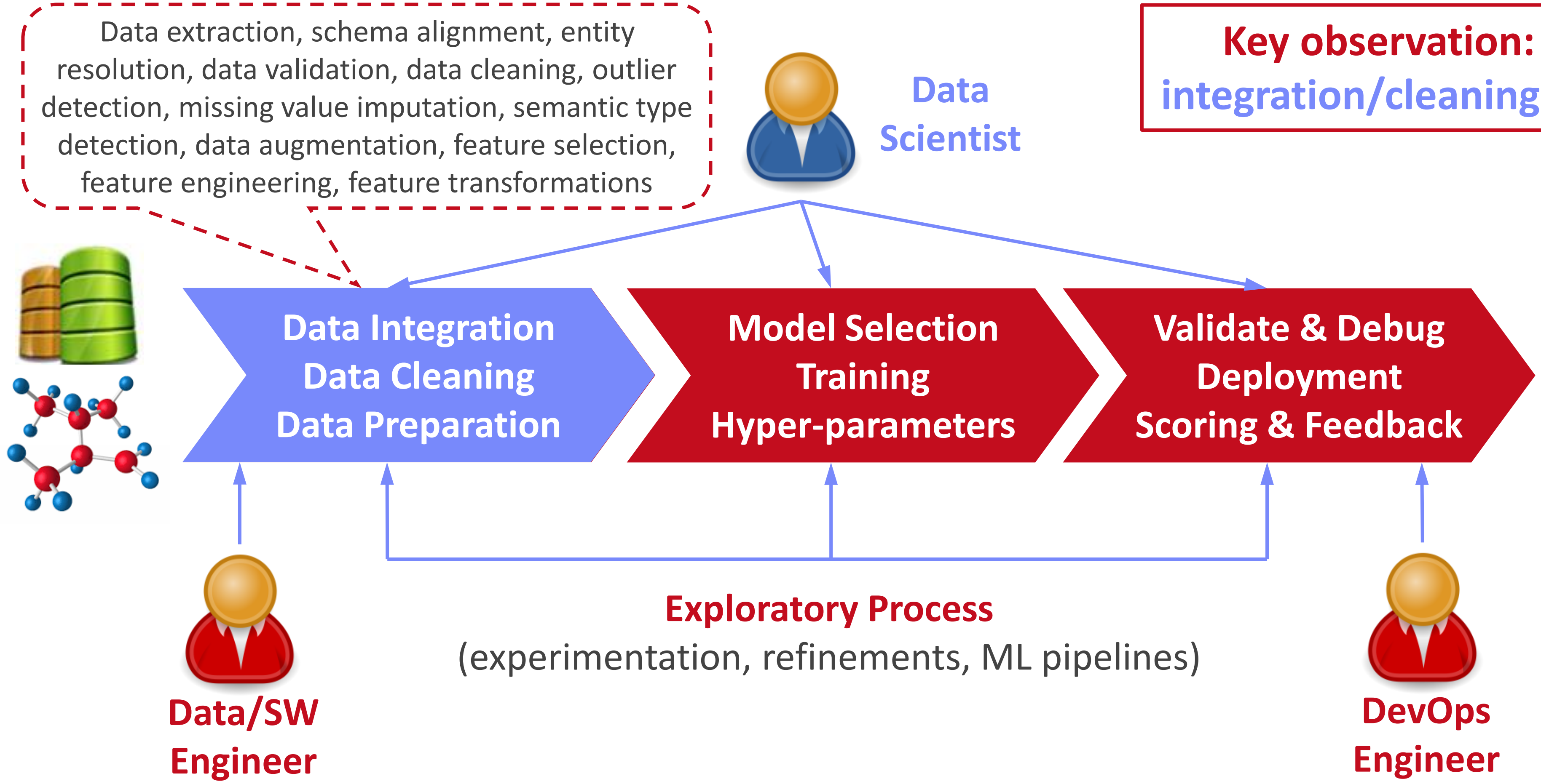
"Take these datasets and show value or competitive advantage"

[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
(aka KDD Process, aka CRISP-DM)

Key observation: SotA data integration/cleaning based on ML

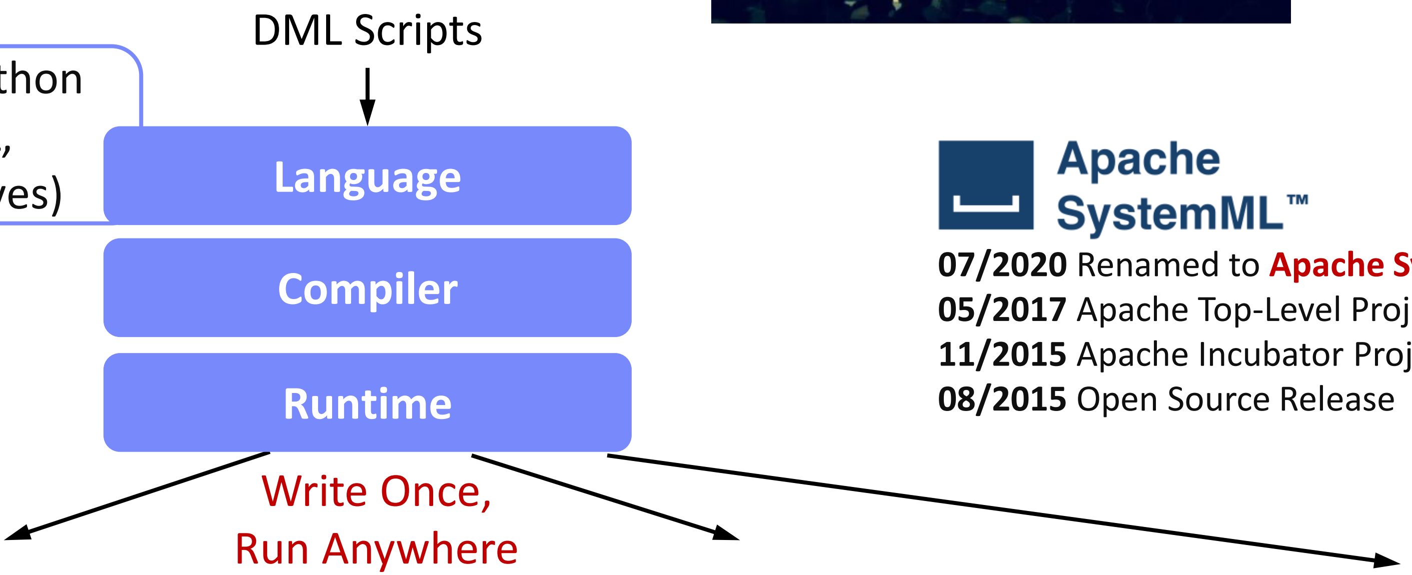


Data extraction, schema alignment, entity resolution, data validation, data cleaning, outlier detection, missing value imputation, semantic type detection, data augmentation, feature selection, feature engineering, feature transformations

Exploratory Process
(experimentation, refinements, ML pipelines)

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

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



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

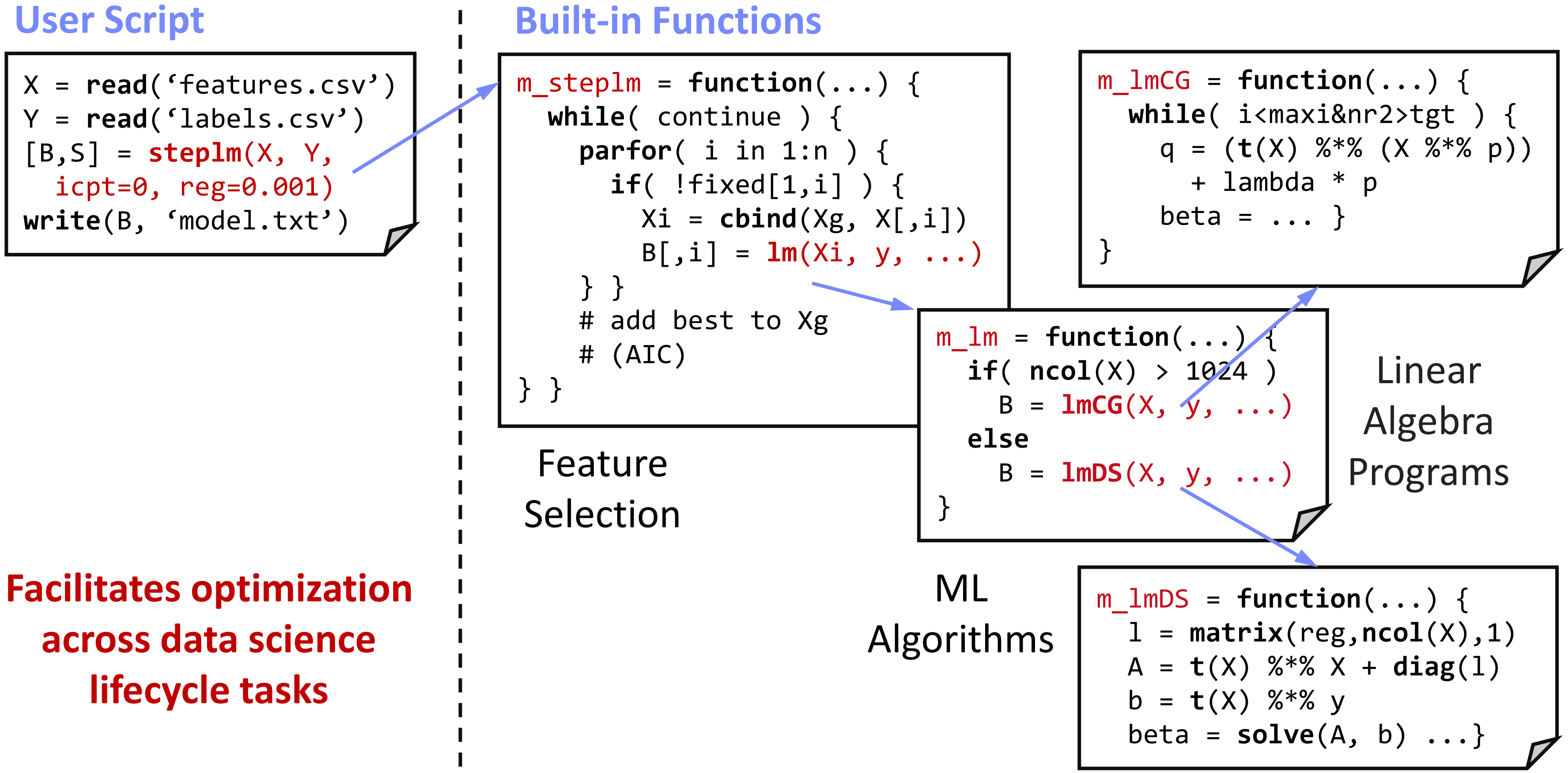
exdra

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’)
```

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

Facilitates optimization across data science lifecycle tasks

Basic HOP and LOP DAG Compilation

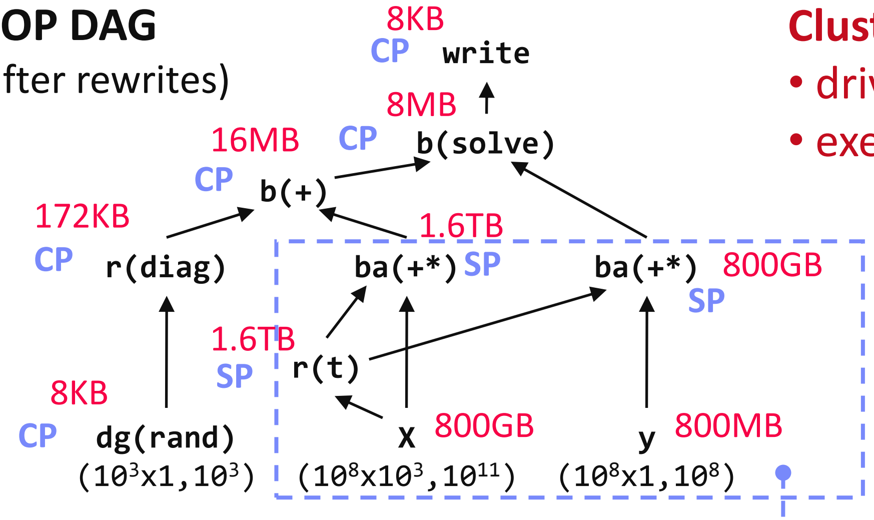
LinregDS (Direct Solve)

```
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)

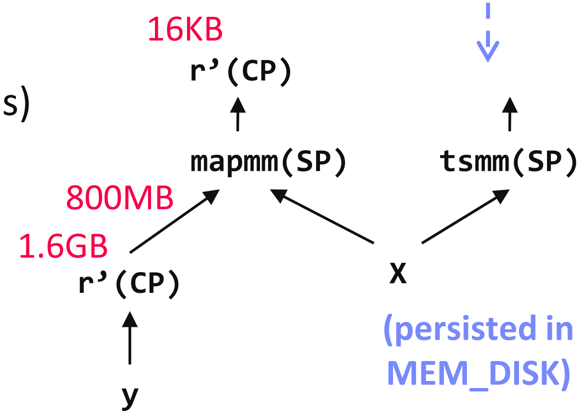


Cluster Config:

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

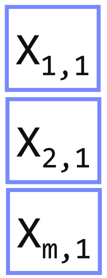
LOP DAG

(after rewrites)



→ **Distributed Matrices**

- Fixed-size matrix blocks
- Data-parallel operations



→ **Hybrid Runtime Plans:**

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

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

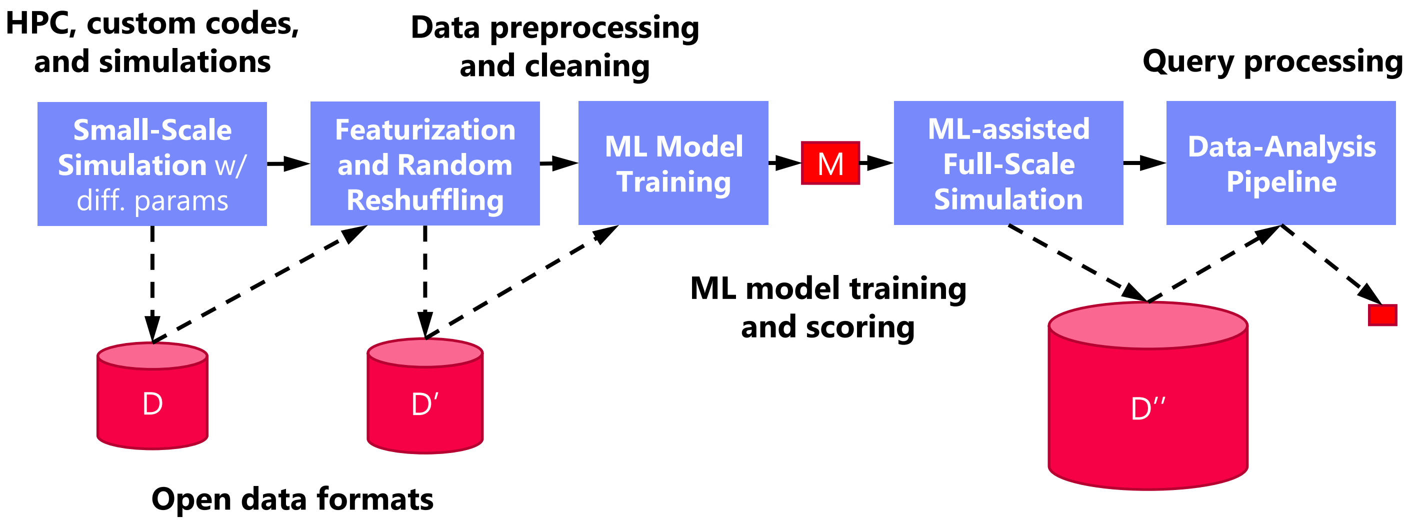
https://github.com/daphne-eu/daphne

Integrated Data Analysis (IDA) Pipelines

DM + ML + HPC

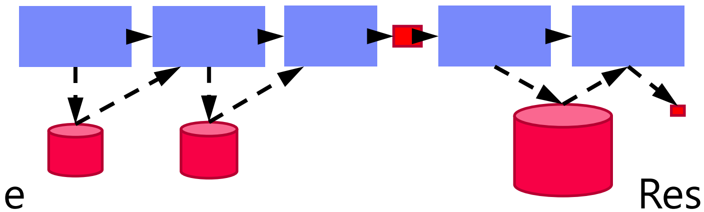
Data Management, Machine Learning, High-Perf. Computing

Example: ML-assisted Simulation


HPC, custom codes, and simulations
Data preprocessing and cleaning
Query processing
Small-Scale Simulation w/ diff. params
Featurization and Random Reshuffling
ML Model Training
M
ML-assisted Full-Scale Simulation
Data-Analysis Pipeline
ML model training and scoring
D
D′
D′′
Open data formats

Challenges

Deployment Challenges



Dev Teams

Programming Models

Different Systems/ Libraries

Cluster Under-utilization

Data/File Exchange

Resource Managers

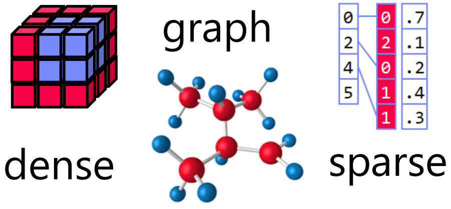
DAPHNE: An open and extensible system infrastructure for IDA pipelines

Increasing Specialization

Hardware Challenges

- DM+ML+HPC share compilation and runtime techniques / converging cluster hardware
- **End of Dennard scaling**
- **End of Moore's law**
- **Amdahl's law**

Data Representations



dense

graph

sparse

compressed

Sparsity Exploitation from Algorithms to HW

Data Placement

Local vs **distributed**

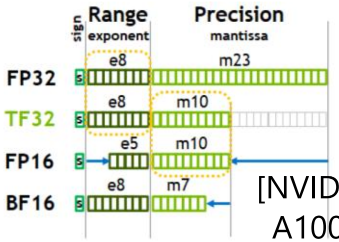
CPUs/ NUMA

GPUs

FPGAs/ ASICs

Data (Value) Types

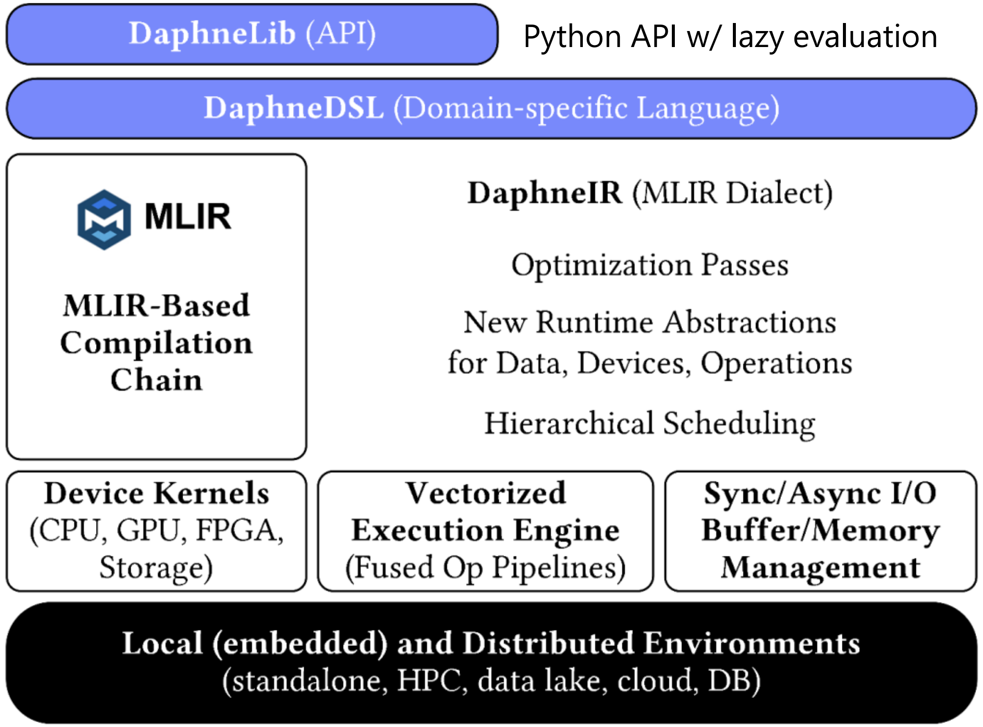
FP32, FP64, INT8, INT32, INT64, UINT8, BF16, TF32, FlexPoint



[NVIDIA A100]

System Architecture

System Architecture



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)

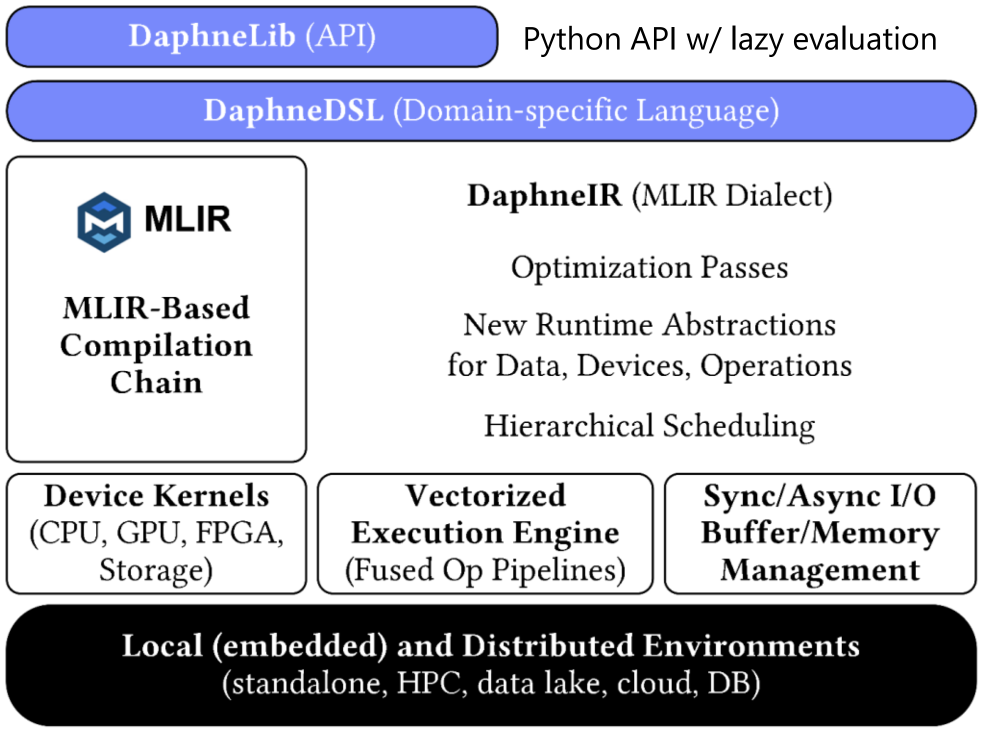
Vectorized Execution Engine (Fused Op Pipelines)

Sync/Async I/O Buffer/Memory Management

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

Language Abstractions

System Architecture



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)

DSL for linear and relational algebra

- Coarse-grained **matrix/frame operations**
- Built-in operations for **linear and relational algebra**
- **High-level operations** (e.g., SQL, parameter servers, map)
- Conditional **control flow** (branches, loops)
- Typed and untyped **user-defined functions**

- **Hierarchy of primitives** for data science tasks

- **Physical data independence**

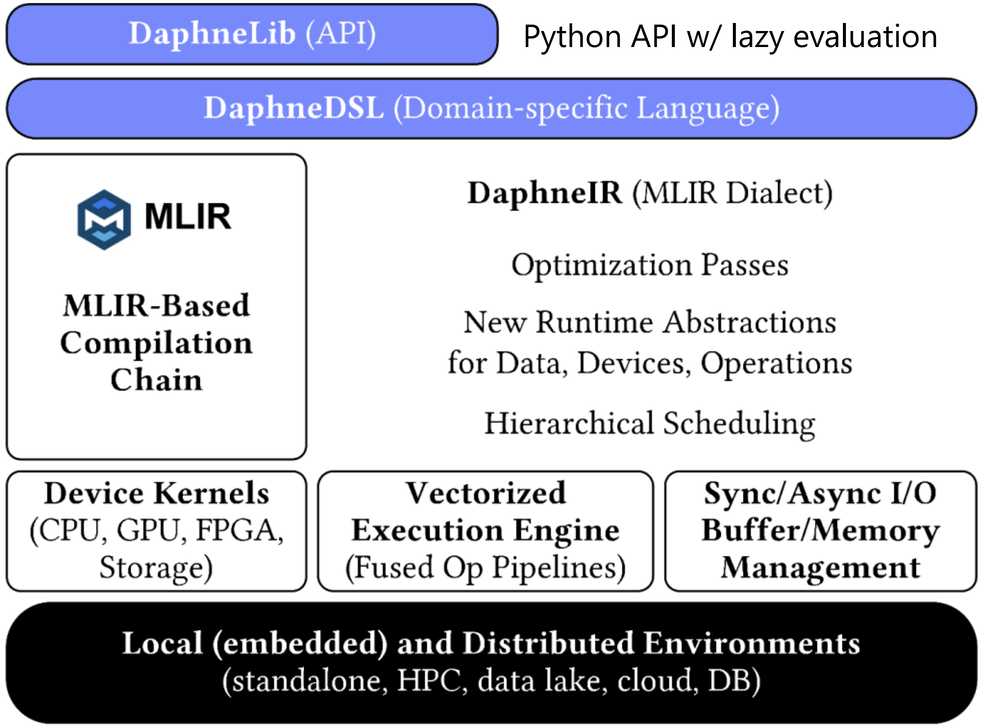
Example: **linear regression model training** (simplified)

```
def lm(X, y) { // X feature matrix, y labels
  colmu = mean(X, 1);       // column means
  colsd = stddev(X, 1);     // column stddevs
  X = (X - colmu) / colsd;  // shift and scale
  X = cbind(X, 1);          // append column of ones
  A = t(X) @ X;             // t for transpose
  b = t(X) @ y;             // @ for matrix mult
  return solve(A, b);       // system of linear eq
}
```

```
my_model = lm(my_X, my_y);
```

Optimizing Compiler

System Architecture



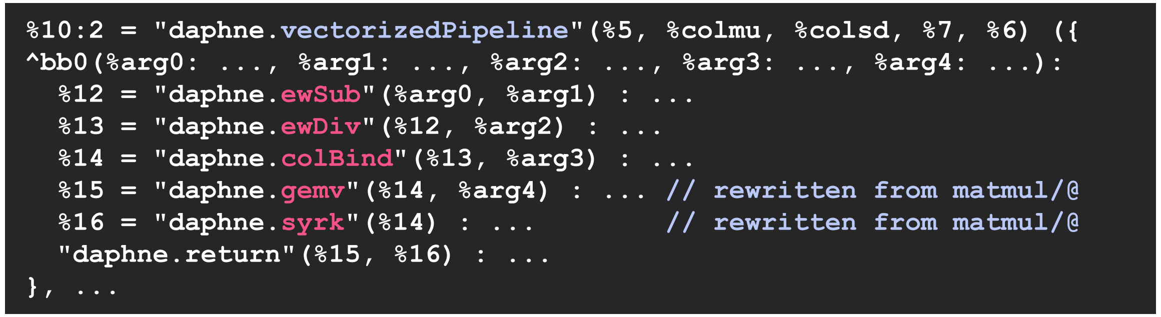
DaphneLib (API)
Python API w/ lazy evaluation
DaphneDSL (Domain-specific Language)
MLIR
MLIR-Based Compilation Chain
DaphneIR (MLIR Dialect)
Optimization Passes
New Runtime Abstractions for Data, Devices, Operations
Hierarchical Scheduling
Device Kernels (CPU, GPU, FPGA, Storage)
Vectorized Execution Engine (Fused Op Pipelines)
Sync/Async I/O Buffer/Memory Management
Local (embedded) and Distributed Environments (standalone, HPC, data lake, cloud, DB)

MLIR-based Optimizing Compiler

- Intermediate representation **DaphneIR (MLIR dialect)**
- **Systematic lowering** from **domain-specific operations** to calls to pre-compiled **kernels for heterogeneous hardware**

- Traditional **programming language rewrites**
- Type & property **inference**, inter-procedural analysis
- **Domain-specific rewrites** from linear and relational algebra
- Memory management & garbage collection
- **Device placement** & **physical operator selection**

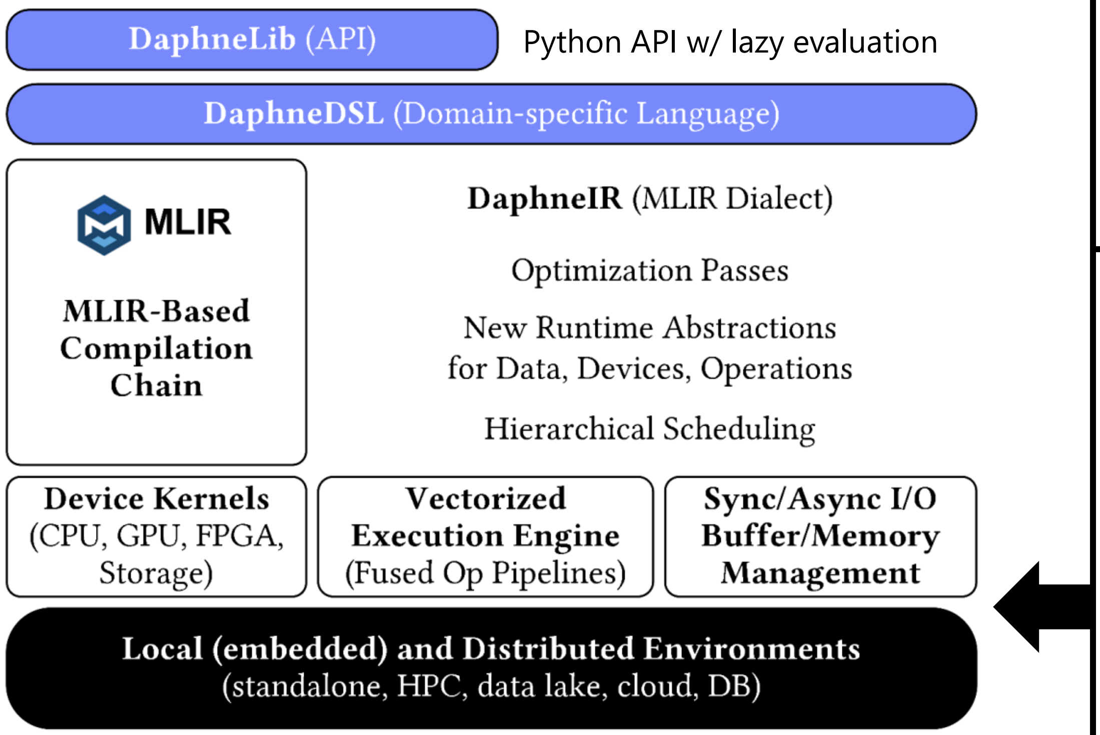
Example: **linear regression model training** (simplified)



```
%10:2 = "daphne.vectorizedPipeline"(%5, %colmu, %colsd, %7, %6) ({
^bb0(%arg0: ..., %arg1: ..., %arg2: ..., %arg3: ..., %arg4: ...):
  %12 = "daphne.ewSub"(%arg0, %arg1) : ...
  %13 = "daphne.ewDiv"(%12, %arg2) : ...
  %14 = "daphne.colBind"(%13, %arg3) : ...
  %15 = "daphne.gemv"(%14, %arg4) : ... // rewritten from matmul/@
  %16 = "daphne.syrk"(%14) : ...        // rewritten from matmul/@
  "daphne.return"(%15, %16) : ...
}, ...
```

Runtime

System Architecture



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)

Distributed and Local Vectorized Execution

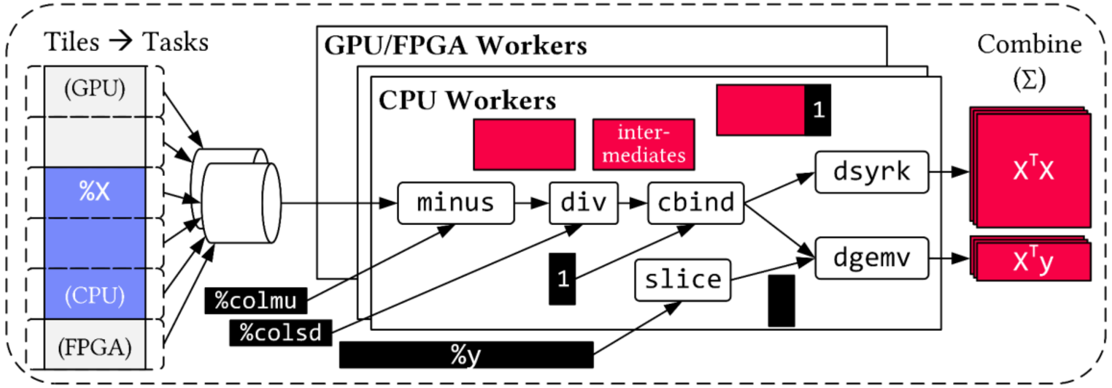
- **Fused operator pipelines** on tiles/vectors of data
- Coarse-grained tasks and cache-conscious data binding

- Device **kernels for heterogeneous hardware**
- Integration of **computational storage** (e.g., eBPF programs)

- **Scheduling for load balancing** (e.g., for ops on sparse data)
- Different **distributed backends** (e.g., gRPC, OpenMPI)

Example: linear regression model training (simplified)

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



Which System to Choose for Your LDE Project: SystemDS or DAPHNE?

- **Lot’s of Similarities**
  - Open-source systems, with major influence of our research group
  - Declarative DSL for linear (and relational) algebra
  - Domain-specific compiler
  - Runtime with local and distributed execution, hardware accelerators
  - Focus on efficient and effective execution of machine learning and data science tasks
  - …

- **Some Differences**
  - **SystemDS**
    - More mature system (since 2010, including history from SystemML)
    - Mainly written in Java and Python
    - Tasks in system internals and DSL scripts
  - **DAPHNE**
    - Younger system (since 2021)
    - Mainly written in C++ and Python
    - Tasks in system internals
    - Based on MLIR (compiler framework)

How to Get Started

- **Suggested Initial Steps**
  - Navigate to the GitHub repo of the respective system
  - Browse the documentation
  - Set up your development environment and try to build and run the system
  - Browse the source code, identify the points related to your task
  - Read the contribution guidelines
  - Start early to identify blocking issues

List of Project Topics (Proposals)

See list at https://pdamme.github.io/teaching/2024-25_winter/lde/ProjectTopics.pdf

Summary and Q&A

- **Course Organization, Outline, and Deliverables**
- **Apache SystemDS and DAPHNE**
- **List of Project Topics (Proposals)**

- **Remaining Questions?**

- **Reminder: Seminar Introductory Lecture Recommended for the Project**
  - 03 Experiments, Reproducibility, and Giving Presentations [Oct 28, 14:00]

- **See you during the office hours and intermediate presentations ☺**