Data Integration and Large-scale Analysis (DIA)
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)

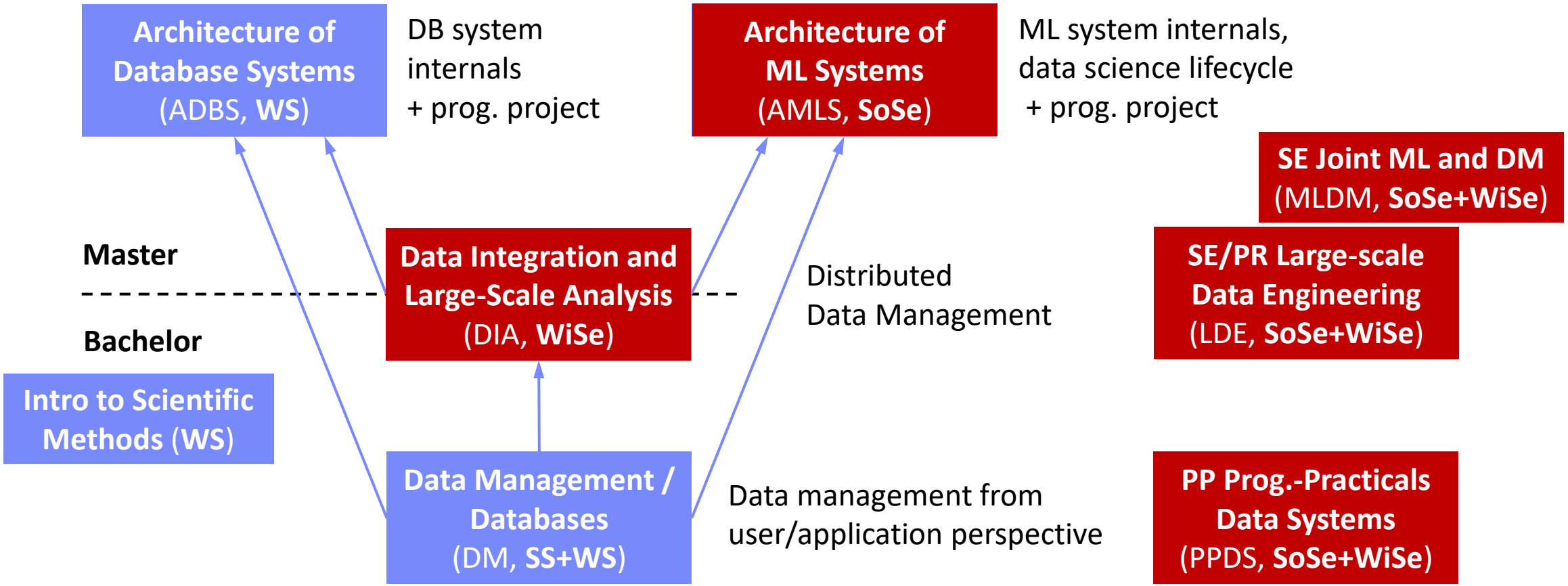
Last update: Oct 16, 2025

FG Big Data Engineering (DAMS Lab) – 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) – Teaching



Faculty IV - Team Awareness and Antidiscrimination

https://www.tu.berlin/eecs/awan

- **Goal**
  - **Low-barrier approachability** for spectrum of awareness and antidiscrimination issues
- **Team**
  - **Irene Hube-Achter** (MTSV)
  - **Matthias Boehm** (professors)
  - **Nadine Karsten** (scientific personnel)
  - **Tom Hersperger** (students)




- **Mission Statement**
  - Account for heterogeneity and complexity of modern societies at TU Berlin
  - All employees and students are committed to
    - **#1** Treat all persons with fairness and respect
    - **#2** Ensure a safe environment for all
    - **#3** Comply with our duty of care towards others
    - **#4** Actively support the implementation of the above guidelines and contribute

Contact: private email, eecs-TB-awareness@win.tu-berlin.de, or AwAn@dams.tu-berlin.de

Agenda

- **Course Organization**
- **Course Motivation and Goals**
- **Course Outline and Projects/Exercise**
- **Excursus: Apache SystemDS**

Course Organization

Course Logistics

- **Staff**
  - **Lecturer:** Prof. Dr. Matthias Boehm, DAMS
  - **Teaching Assistant:** Carlos E. Muniz Cuza, DAMS

- **Language**
  - Lectures and slides: English
  - Communication and examination: English/German
- **Course Format**
  - VL/UE 3/2 SWS, **6 ECTS** (3 ECTS + 3 ECTS), bachelor/master; no capacity restrictions **279 Reg** (as of Oct 16)
  - **Weekly lectures** (**Thu 4.15pm sharp**, in-person & zoom livestreaming/recording), **optional attendance**
  - **Mandatory exercises or programming project** (3 ECTS), office hour **Wed 5pm-6pm** (sharp)
  - **Recommended papers** for additional reading on your own
- **Prerequisites**
  - Basic understanding of SQL / RA (or willingness to fill gaps)
  - Basic programming skills (Python, R, Java, C++)

Course Logistics, cont.

- **Website / ISIS Course / Zoom**
  - https://mboehm7.github.io/teaching/ws2526_dia/index.htm (public)
  - https://isis.tu-berlin.de/course/view.php?id=44995 (TUB-internal)
  - https://tu-berlin.zoom.us/j/9529634787?pwd=R1ZsN1M3SC9BOU1OcFdmem9zT202UT09
- **Communication**
  - **Informal language** (first name is fine); **immediate feedback** welcome
  - ISIS Forum for offline Q&A on projects/exercises as well as
  - **TA Office hours:** TBD second week
- **Academic Honesty / No Plagiarism** (incl LLMs like ChatGPT)

- **Exam**
  - **Exam Prerequisite: Completed exercises or project** (checked by teaching assistants)
  - **Final written exam** (oral exam if <35 students take the exam): **Feb 05, 4pm** / **Feb 12, 4pm** / **Mar 12, 4pm**
  - **Grading** (project/exercises pass/fail, 100% exam) → **5 extra points in exam** if exercises with >= 90%

Course Applicability

- **Bachelor study programs computer science, information systems management, computer engineering, and electrical engineering**
- **Master study programs computer science, information systems management, computer engineering, and electrical engineering**
  - Data and software engineering
  - Cognitive systems
  - Distributed systems and networks
- **Free subject course in any other study program or university**

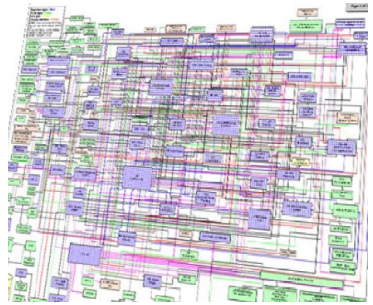
- **(currently reorganization StuPO WS26/27 bachelor computer science → DIA in ”Data Systems” catalog)**
- **Different than “Data Integration: Algorithms and Systems (DI)”**

Course Motivation and Goals

Data Sources and Heterogeneity

- **Terminology**
  - **Integration** (Latin integer = whole): consolidation of data objects / sources
  - **Homogeneity** (Greek homo/homoios = same): similarity
  - **Heterogeneity**: dissimilarity, different representation / meaning

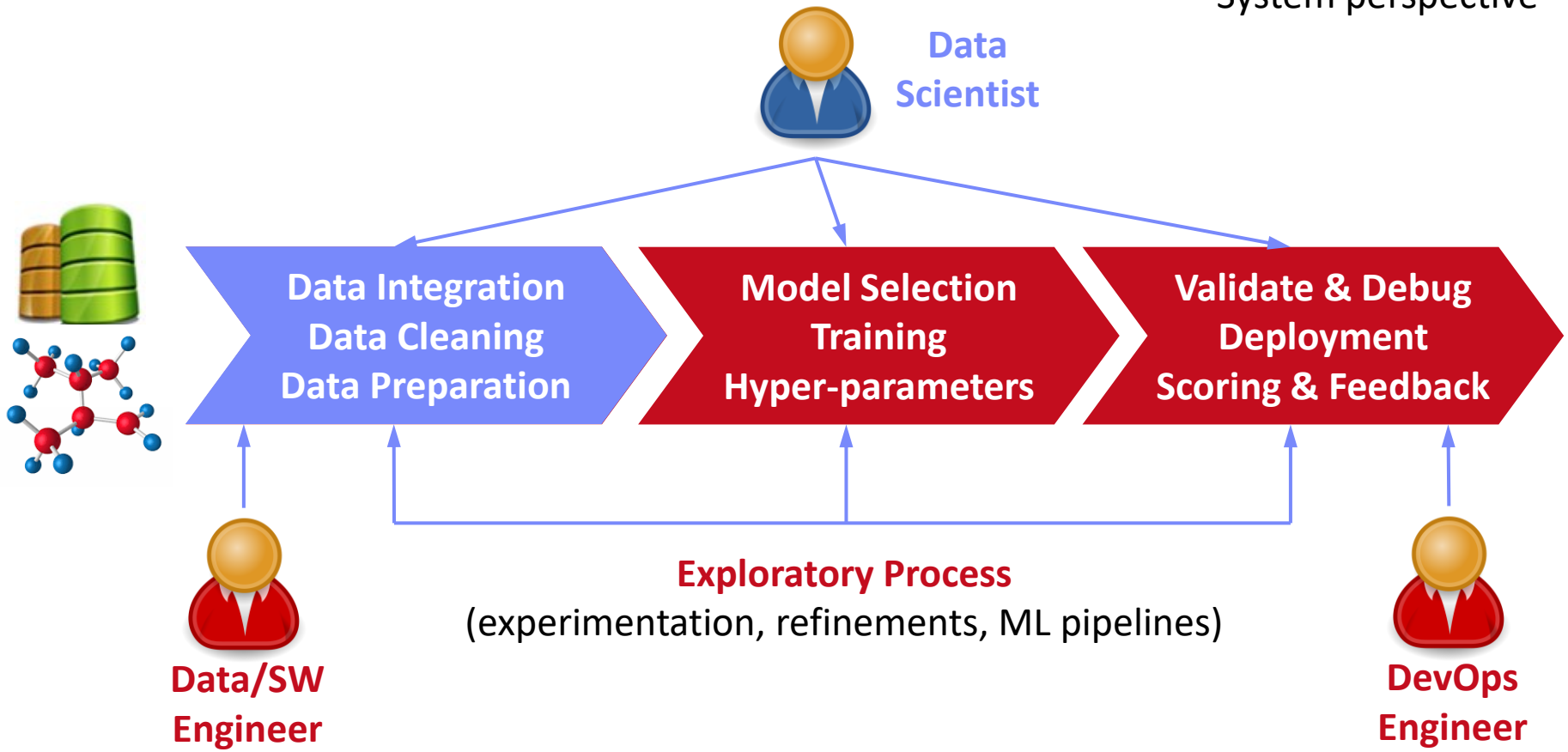
[**Credit:** Albert Maier]



- **Heterogeneous IT Infrastructure**
  - Common enterprise IT infrastructure contains >100s of heterogeneous and distributed systems and applications
  - E.g., health care data management: 20 - 120 systems
- **Multi-Modal Data (example health care)**
  - Structured patient data, patient records incl. prescribed drugs
  - Knowledge base drug APIs (active pharmaceutical ingredients) + interactions
  - Doctor notes (text), diagnostic codes, outcomes
  - Radiology images (e.g., MRI scans), patient videos
  - Time series (e.g., EEG, ECoG, heart rate, blood pressure)

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

Data-centric View:
Application perspective
Workload perspective
System perspective

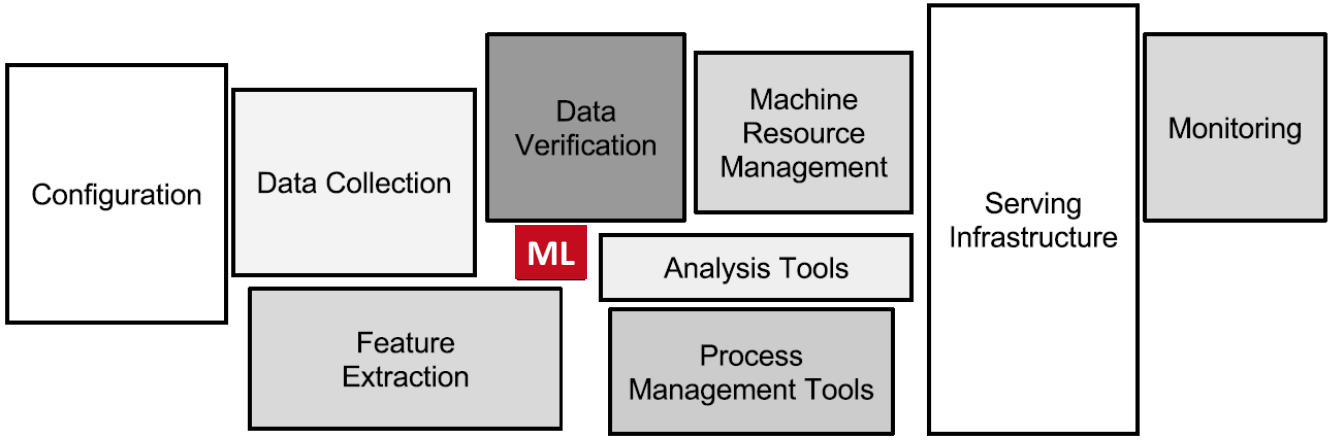


The 80% Argument

- **Data Sourcing Effort**
  - Data scientists spend **80-90% time** on finding, integrating, cleaning datasets

[Michael Stonebraker, Ihab F. Ilyas: Data Integration: The Current Status and the Way Forward. **IEEE Data Eng. Bull. 41(2) (2018)**]

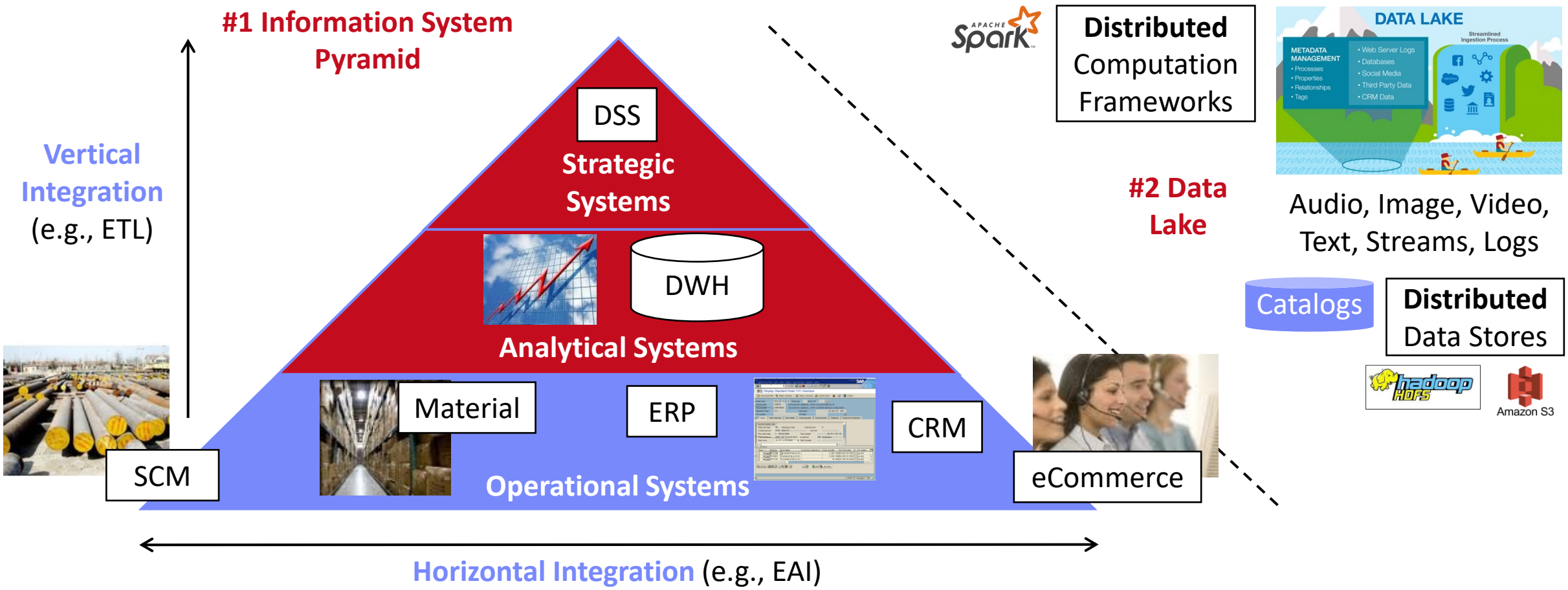
- **Technical Debts in ML Systems**



[D. Sculley et al.: Hidden Technical Debt in Machine Learning Systems. **NeurIPS 2015**]

  - Glue code, pipeline jungles, dead code paths
  - Plain-old-data types (arrays), multiple languages, prototypes
  - Abstraction and configuration debts
  - Data testing, reproducibility, process management, and cultural debts

Complementary System Architectures



Course Goals

- **Common Data and System Characteristics**
  - **Heterogeneous** data sources and formats, often distributed
  - **Large** data collections → **distributed** data storage and analysis

- **#1** Major data integration architectures

- **#2** Key techniques for data integration and cleaning

- **#3** Methods for large-scale data storage and analysis

Course Outline and Projects/Exercise

Part A: Data Integration and Preparation

Data Integration Architectures

- **01 Introduction and Overview** [Oct 16]
- **02 Data Warehousing, ETL, and SQL/OLAP** [Oct 23]
- **03 Message-oriented Middleware, EAI, and Replication** [Oct 30]

Key Integration Techniques

- **04 Schema Matching and Mapping** [Nov 06]
- **05 Entity Linking and Deduplication** [Nov 13]
- **06 Data Cleaning and Data Fusion** [Nov 20]
- **07 Data Provenance and Catalogs** [Nov 27]

Part B: Large-Scale Data Management & Analysis

Cloud Computing

- **08 Cloud Computing Foundations** [Dec 04]
- **09 Cloud Resource Management and Scheduling** [Dec 11]
- **10 Distributed Data Storage** [Dec 18]

Large-Scale Data Analysis

- **11 Distributed, Data-Parallel Computation** [Jan 15]
- **12 Distributed Stream Processing** [Jan 22]
- **13 Distributed Machine Learning Systems** [Jan 29]
  + Q&A and Exam Preparation

Overview Projects or Exercises

- **Team**
  - **1-3 person teams** (w/ clearly separated responsibilities)

- **Objectives**
  - Non-trivial programming project in DIA context (**3 ECTS → 80-90 hours**)
  - **Preferred:** Open-source contribution to **Apache SystemDS** https://github.com/apache/systemds (from HW to high-level scripting)
  - https://issues.apache.org/jira/secure/Dashboard.jspa?selectPageId=12335852#Filter-Results/12365413
  - **Alternative Exercise: "Berlin Public Transport Data Analysis"**

- **Timeline**
  - **Oct 31:** Binding project/exercise selection (via google forms)
  - **Jan 30:** Project/exercise submission deadline

https://tinyurl.com/aytk6bw6

DIA Exercise (alternative to projects), cont.

DIA WiSe2025: Exercise – Berlin Public Transport Data Analysis

Published: Oct 13, 2025
Deadline: Jan 30, 2025; 11.59pm

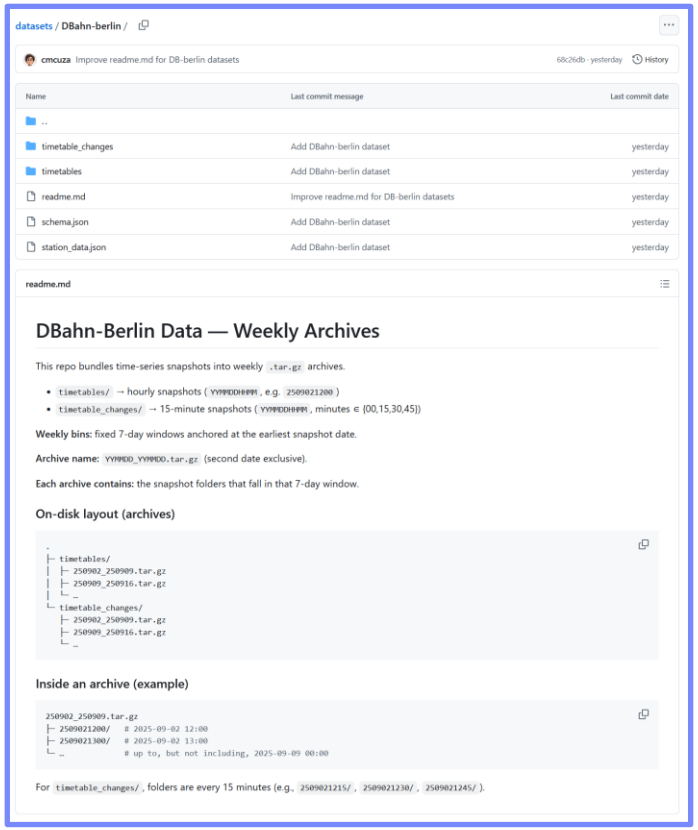
This exercise is an alternative to the DIA programming projects and provides practical experience in the development of ETL pipelines for data integration and analytics. The task of this semester is to ingest and analyze data from Berlin's public transport system, extracted via the Deutsche Bahn (DB) API marketplace[1]. We collected real-world information for 133 Berlin stations, including train connections and disruptions, from Sep 02, 2025 through Oct 15, 2025. The objectives are to design a schema capable of accommodating this data and to implement queries that demonstrate proficiency in data integration and large-scale analysis. The final submission is a zip archive named `DIA_Exercise_<student ID>.zip` (max 5MB), containing: (1) the source code used to solve the individual sub-tasks, as well as (2) a PDF report of up to 8 pages (10pt), including the names of all team members, a summary of how to run the code, and a description of the solutions to the individual sub-tasks.

Data Source: The dataset consists of three main components:

- `Stations:` A `.json` file with metadata for the 133 Berlin stations.
- `Timetables:` `.xml` files containing planned train movements (arrival/departure times, platforms, lines, numbers, routes), collected once per hour at `HH:01`.
- `Timetable Changes:` `.xml` files containing disruptions (delays, cancellations, modification messages), collected every 15 minutes at `HH:01`, `HH:16`, `HH:31`, and `HH:46`.

We also provide a `schema.json` file describing all fields. All timetable files are stored as: `/timetables/{date_hour_00}/{station_name}_timetable.xml`, where `date_hour_00` encodes the download time, e.g. `2509051100` for Sep 05, 2025 at 11:00. Each file covers one hour of data. Furthermore, all timetable change files are stored as: `/timetable_changes/{date_hour_minute}/{station_name}_change.xml`, where `date_hour_minute` encodes the download time, e.g. `2509051116` for Sep 05, 2025 at 11:16. Each file covers 15 minutes of data.

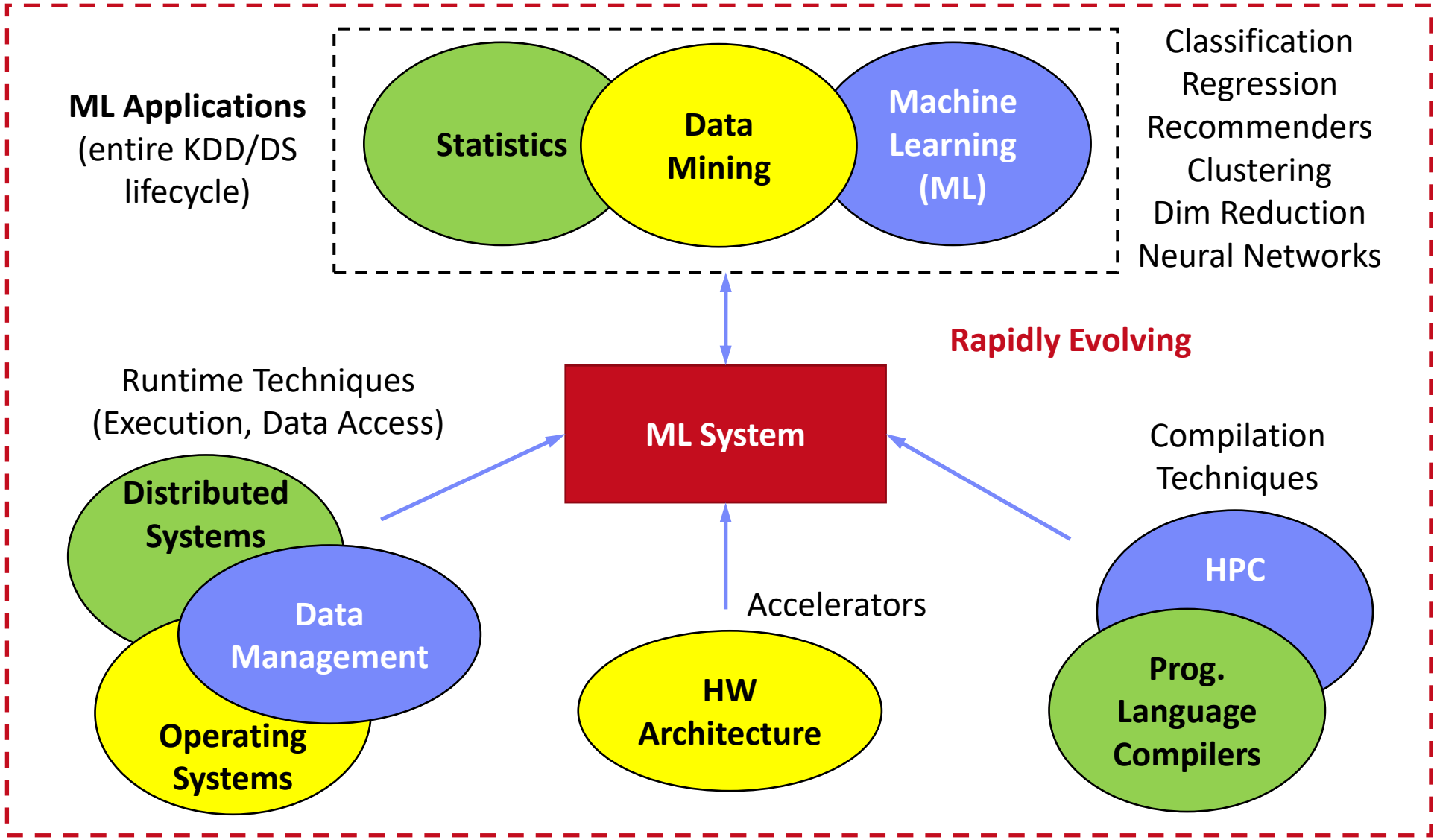
[https://github.com/damslab/datasets/tree/master/DBahn-berlin]



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

https://github.com/apache/systemds

What is an ML System?



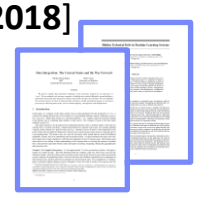
Landscape of ML Systems

- **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**

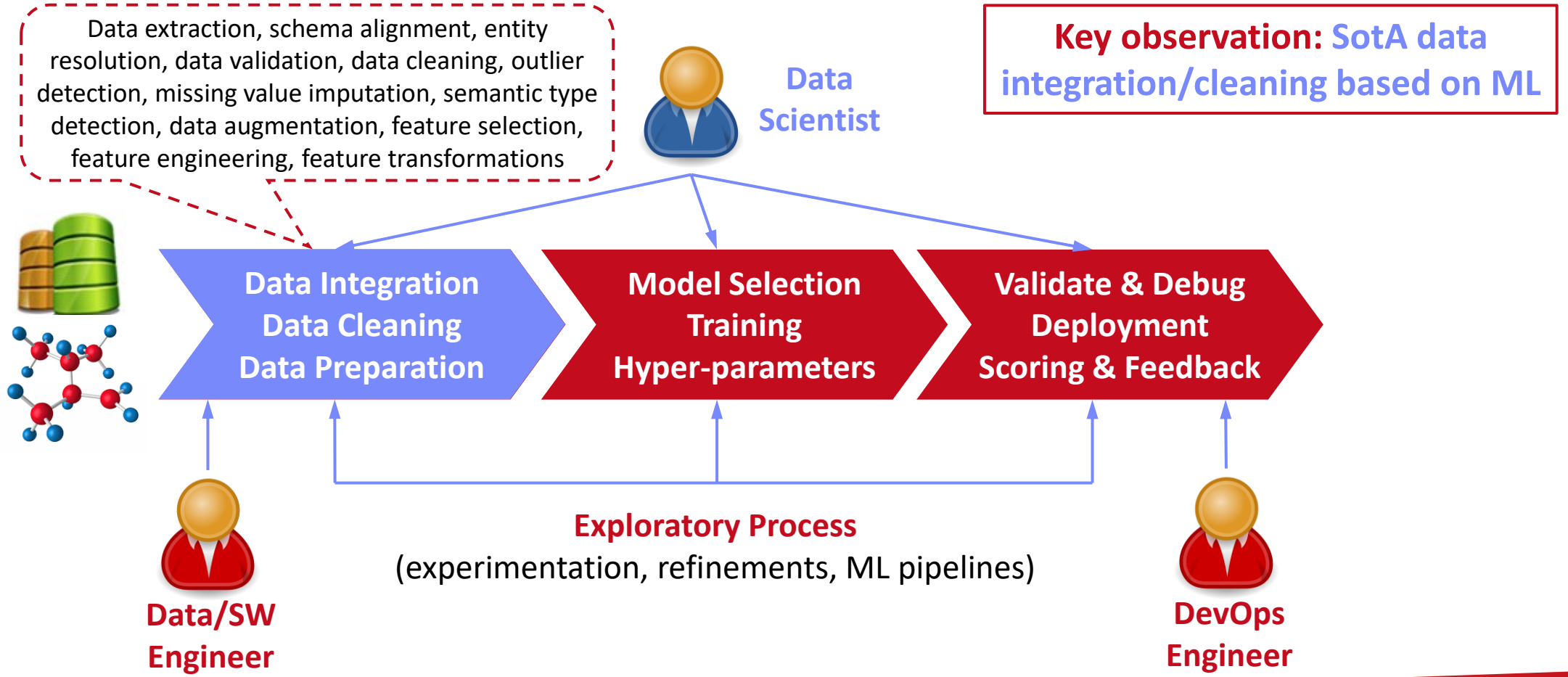
"Take these datasets and show value or competitive advantage"

[NIPS 2015]
[DEBull 2018]

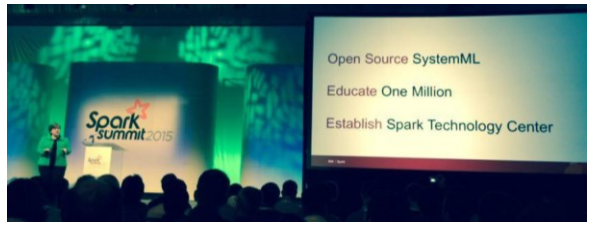


- **Data Preparation Problem**
  - **80% Argument:** 80-90% time for finding, integrating, cleaning data
  - Diversity of tools → boundary crossing, lack of optimization

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



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,'24ab]
[PVLDB'14,'16ab,'18,'22]
[ICDE'11,'12,'15]
[EDBT'25][BTW'25ab]
[CIDR'17,'20]
[VLDBJ'18] [TODS'26]
[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)

Java

since 2012

Hadoop or Spark Cluster
(scale-out)

since 2010/11

since 2015

Federated
(LA progs, PS)

since 2019

In-Progress:

GPU

since 2014/16

Others:
Netezza
Apache Flink

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

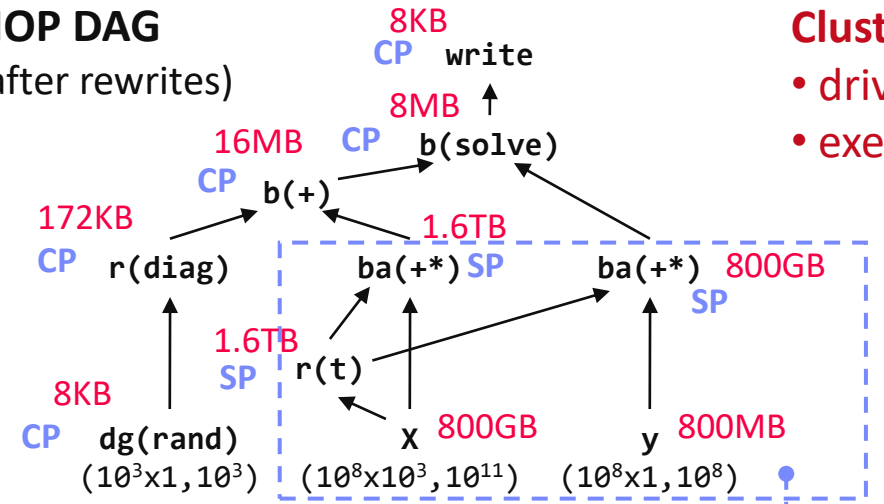
Basic HOP and LOP DAG Compilation

LinregDS (Direct Solve)

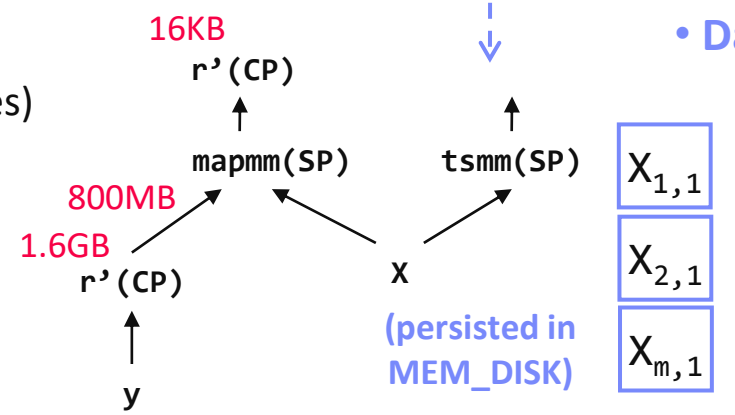
```
X = read($1);
y = read($2);
intercept = $3;
lambda = 0.001;
...
if( intercept == 1 ) {
  ones = matrix(1, nrow(X), 1);
  X = append(X, ones);
}
I = matrix(1, ncol(X), 1);
A = t(X) %*% X + diag(I)*lambda;
b = t(X) %*% y;
beta = solve(A, b);
...
write(beta, $4);
```

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

HOP DAG
(after rewrites)



LOP DAG
(after rewrites)

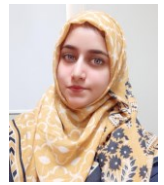


Cluster Config:
- driver mem: 20 GB
- exec mem: 60 GB

➔ Distributed Matrices
- Fixed-size matrix blocks
- Data-parallel operations

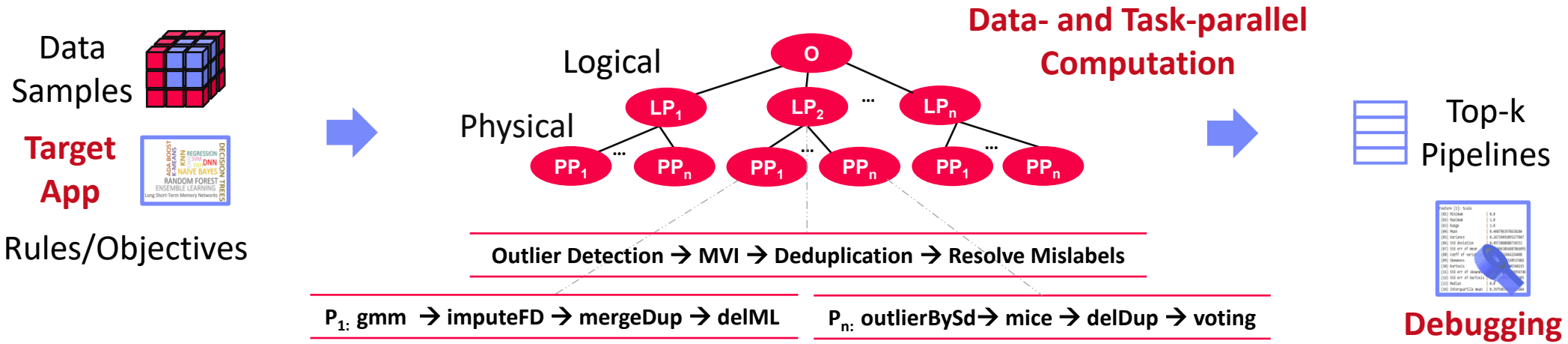
➔ Hybrid Runtime Plans:
- Size propagation / memory estimates
- Integrated CP / Spark runtime
- Dynamic recompilation during runtime

Data Cleaning Pipelines [SIGMOD'24a, TODS'26]




[**best paper runner-up** w/ Shafaq and Roman]

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



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, BTW'25a]

sliceline

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

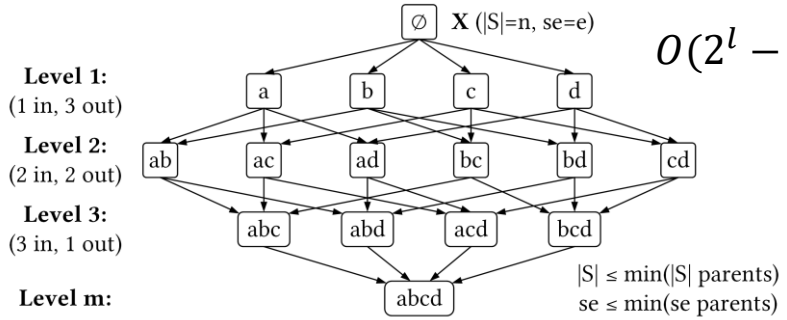
- **Problem Formulation**
  - Intuitive slice scoring function
  - Exact **top-k slice finding**
  - $|S| \geq \sigma \wedge sc(S) > 0, \alpha \in (0,1]$

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

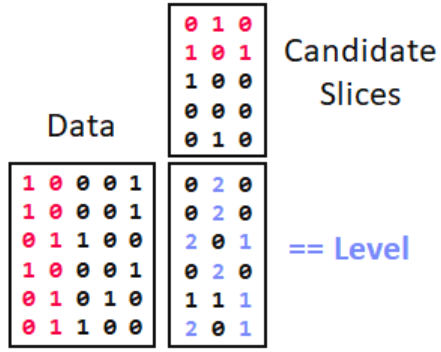
$$= \alpha\left(\frac{|X|}{|S|} \cdot \frac{\sum_{i=1}^{|S|} es_i}{\sum_{i=1}^{|X|} e_i} - 1\right) - (1-\alpha)\left(\frac{|X|}{|S|} - 1\right)$$

slice error **slice size**

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



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



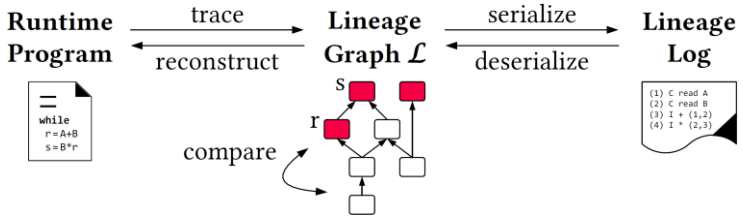
- **Linear-Algebra-based Slice Finding**
  - Recoded/binned matrix **X**, error vector e
  - Vectorized implementation in linear algebra (join & eval via sparse-sparse matmult)
  - Local and distributed task/data-parallel execution

BIFOLD

Multi-level Lineage Tracing & Reuse

[CIDR'20, SIGMOD'21a, EDBT'25]


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

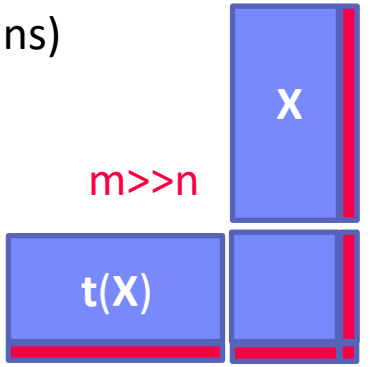


- **Full Reuse of Intermediates**
  - Before executing instruction, probe output lineage in cache
    `Map<Lineage, MatrixBlock>`
  - Cost-based/heuristic caching and eviction decisions (compiler-assisted)

```
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



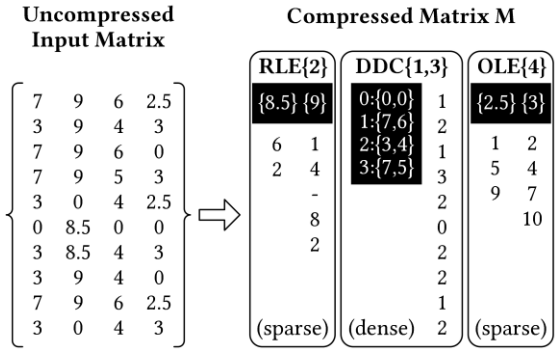

```
m_steplm = function(...) {
  while( continue ) {
    parfor( i in 1:n ) {
      if( !fixed[1,i] ) {
        Xi = cbind(Xg, X[,i])
        B[,i] = lm(Xi, y, ...)
    } }
    # add best to Xg (AIC)
} }
```

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

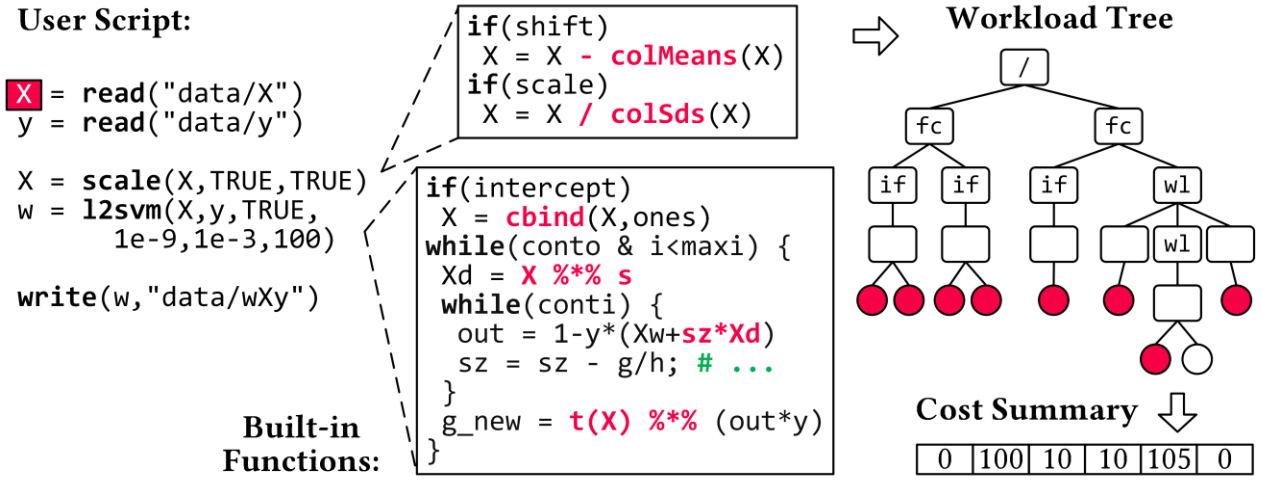
Compressed Linear Algebra Extended

[PVLDB'16a, VLDBJ'18, SIGMOD'23a, under submission]

- **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

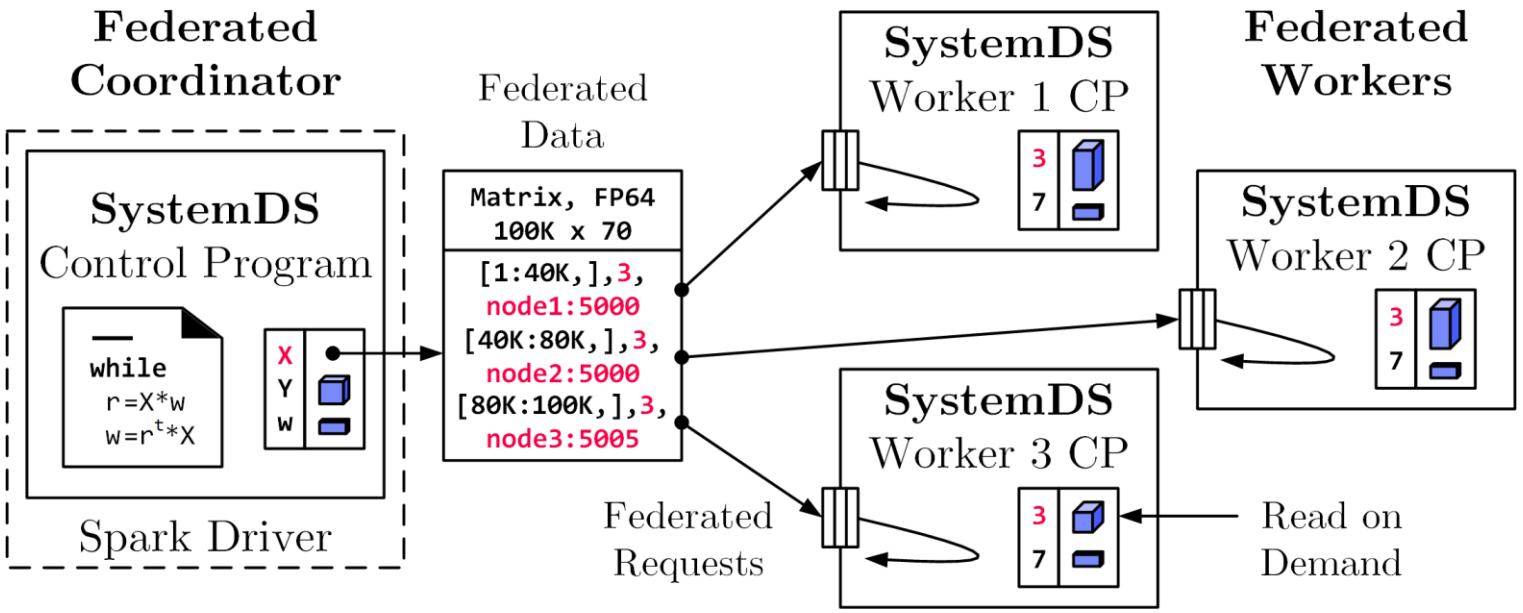


- **Next Steps**
  - Frame compression, compressed I/O
  - Compressed feature transformations
  - Morphing of compressed data

Federated Learning [SIGMOD'21c, CIKM'22]

- **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)));
```



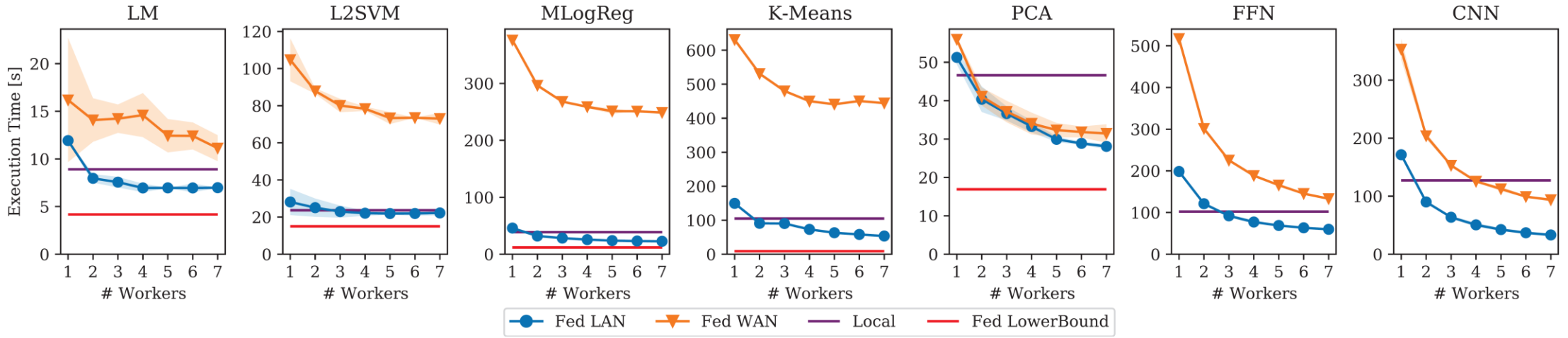
Federated Requests:
`READ, PUT, GET, EXEC_INST, EXEC_UDF, CLEAR`

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

Federated Learning – Experiments

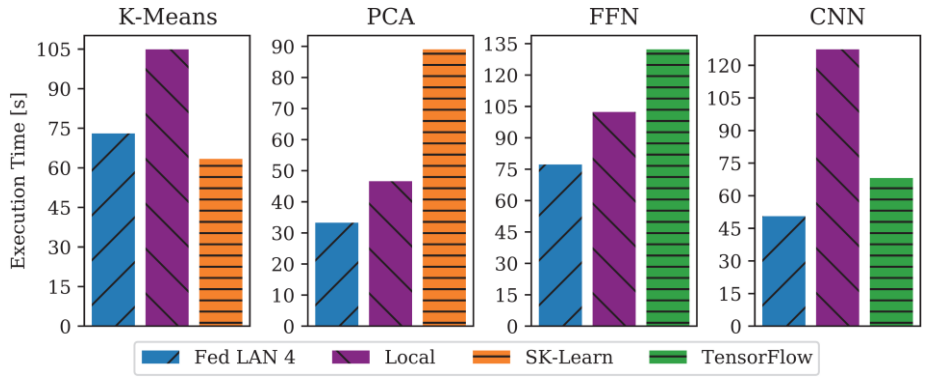
Reproducible Results

OPEN ACCESS



- **Workloads and Baselines**
  - **LM:** linear regression, lmCG
  - **L2SVM:** l2-regularized SVM
  - **MLogReg:** multinomial logreg
  - **K-Means:** Lloyd's alg. w/ K-Means++ init
  - **PCA:** principal component analysis
  - **FFN:** fully-connected feed-forward NN
  - **CNN:** convolutional NN

Comparisons w/ **Scikit-learn** and **TensorFlow**



Summary and Q&A

- **Course Goals**
  - **#1** Major data integration architectures
  - **#2** Key techniques for data integration and cleaning
  - **#3** Methods for large-scale data storage and analysis

- **Programming Projects**
  - Unique project in **Apache SystemDS** (teams or individuals), **or**
  - Exercise on **data engineering /ML pipeline**
  - Project selection by Oct 31 EOD

- **Next Lectures**
  - **02** Data Warehousing, ETL, and SQL/OLAP [Oct 23]
  - **03** Message-oriented Middleware, EAI, and Replication [Oct 30]

Thanks

https://tinyurl.com/aytk6bw6