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 17, 2024

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

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

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 (WS)

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

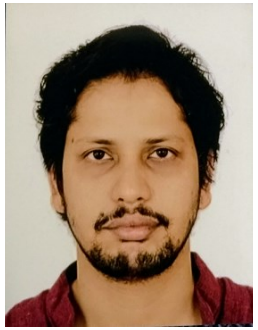
PP Prog.-Practicals Data Systems (PPDS, **SoSe+WiSe**)

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:** M.Tech. Arnab Phani, 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

    219 Reg (as of Oct 17)
  - **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/ws2425_dia/index.htm (public)
  - https://isis.tu-berlin.de/course/view.php?id=39740 (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 06, 4pm** / **Feb 13, 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**

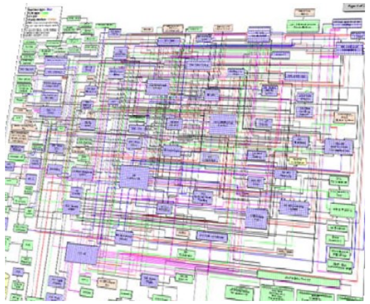
- **(currently reorganization StuPO WS25/26 bachelor computer science → DIA in ”Data Systems” catalog)**
- **Different than “Data Integration: Algorithms and Systems (DI)”**

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
- **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

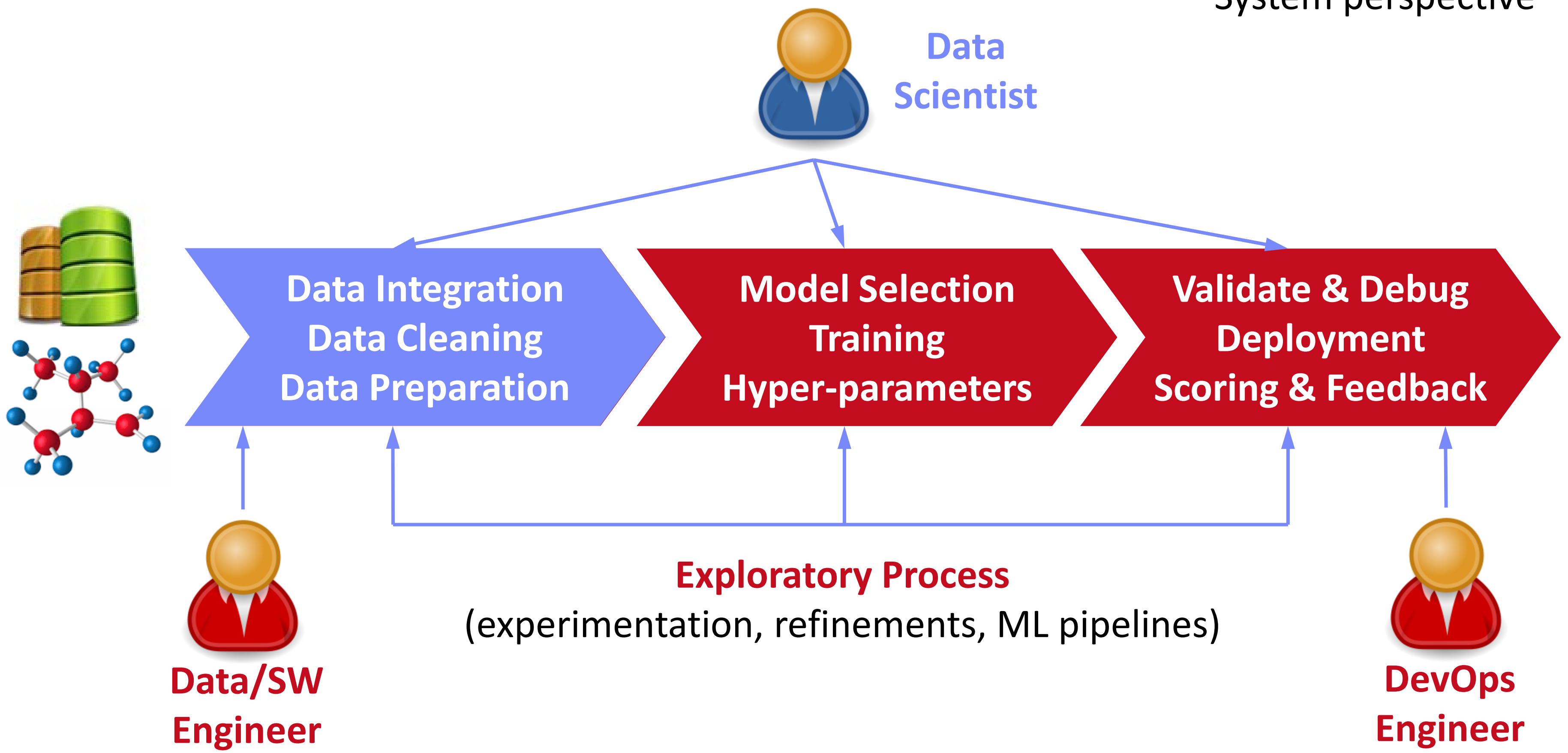
[**Credit:** Albert Maier]



- **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

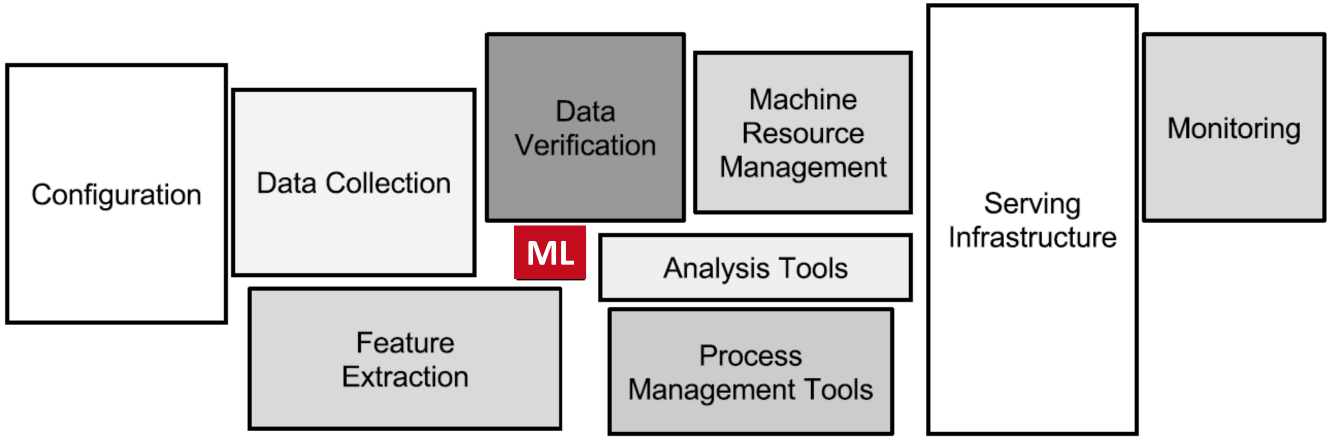


The 80% Argument

- **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)**]

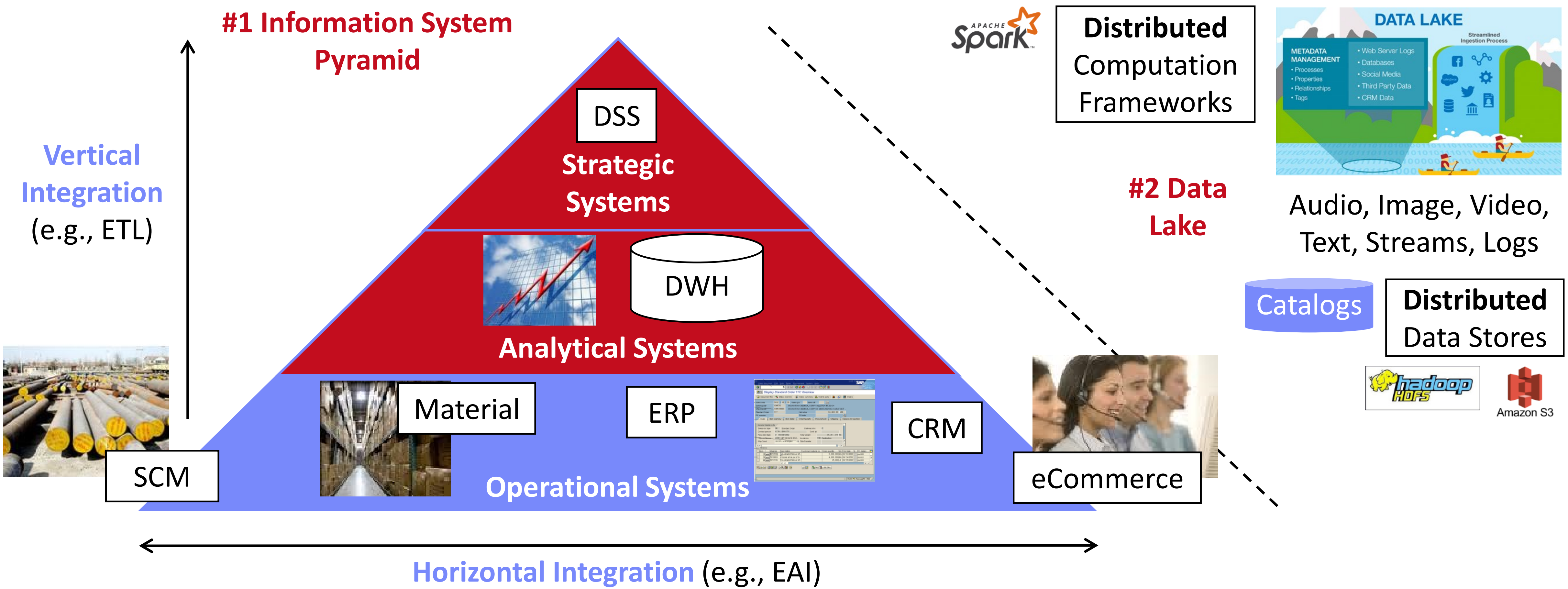
- **Technical Debts in ML Systems**



[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 17]
- **02 Data Warehousing, ETL, and SQL/OLAP** [Oct 24]
- **03 Message-oriented Middleware, EAI, and Replication** [Oct 31]

Key Integration Techniques

- **04 Schema Matching and Mapping** [Nov 07]
- **05 Entity Linking and Deduplication** [Nov 14]
- **06 Data Cleaning and Data Fusion** [Nov 21]
- **07 Data Provenance and Catalogs** [Nov 28]

Part B: Large-Scale Data Management & Analysis

Cloud Computing

- **08 Cloud Computing Foundations** [Dec 05]
- **09 Cloud Resource Management and Scheduling** [Dec 12]
- **10 Distributed Data Storage** [Dec 19]

Large-Scale Data Analysis

- **11 Distributed, Data-Parallel Computation** [Jan 09]
- **12 Distributed Stream Processing** [Jan 16]
- **13 Distributed Machine Learning Systems** [Jan 23]

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: "Streaming Full Text Search"** (ACM SIGMOD 2013 Programming Contest)

03 Replication and Message-oriented Middleware
09 Cloud Resource Management and Scheduling
11 Distributed Data-parallel Computation
12 Distributed Stream Processing

- **Timeline**
  - **Oct 31:** Binding project/exercise selection (via email to matthias.boehm@tu-berlin.de)
  - **Jan 30:** Project/exercise submission deadline

DIA Exercise (alternative to projects), cont.

[https://mboehm7.github.io/teaching/ws2425_dia/DIA_2024_Exercise.pdf]

Univ.-Prof. Dr.-Ing. Matthias Boehm
Technische Universität Berlin
Faculty IV - Electrical Engineering and Computer Science
Berlin Institute for the Foundations of Learning and Data (BIFOLD)
Big Data Engineering (DAMS Lab) Group

1 DIA WiSe2024: Exercise – Streaming Full Text Search

Published: Oct 16, 2024 (last update: Oct 16)
Deadline: Jan 30, 2025, 11.59pm

This exercise is an alternative to the DIA programming projects, and aims to provide practical experience in the development of data engineering and ML pipelines. The task of this semester is to filter a stream of documents using a dynamic set of exact and approximate continuous keyword match queries. This task resembles the SIGMOD Programming Contest 2013. You may use any programming language(s) of your choosing, and utilize existing open-source ML frameworks and libraries. The expected result is a zip archive named `DIA_Exercise_<student_ID>.zip` (replace `<student_ID>` by your student ID) of max 5 MB, containing:

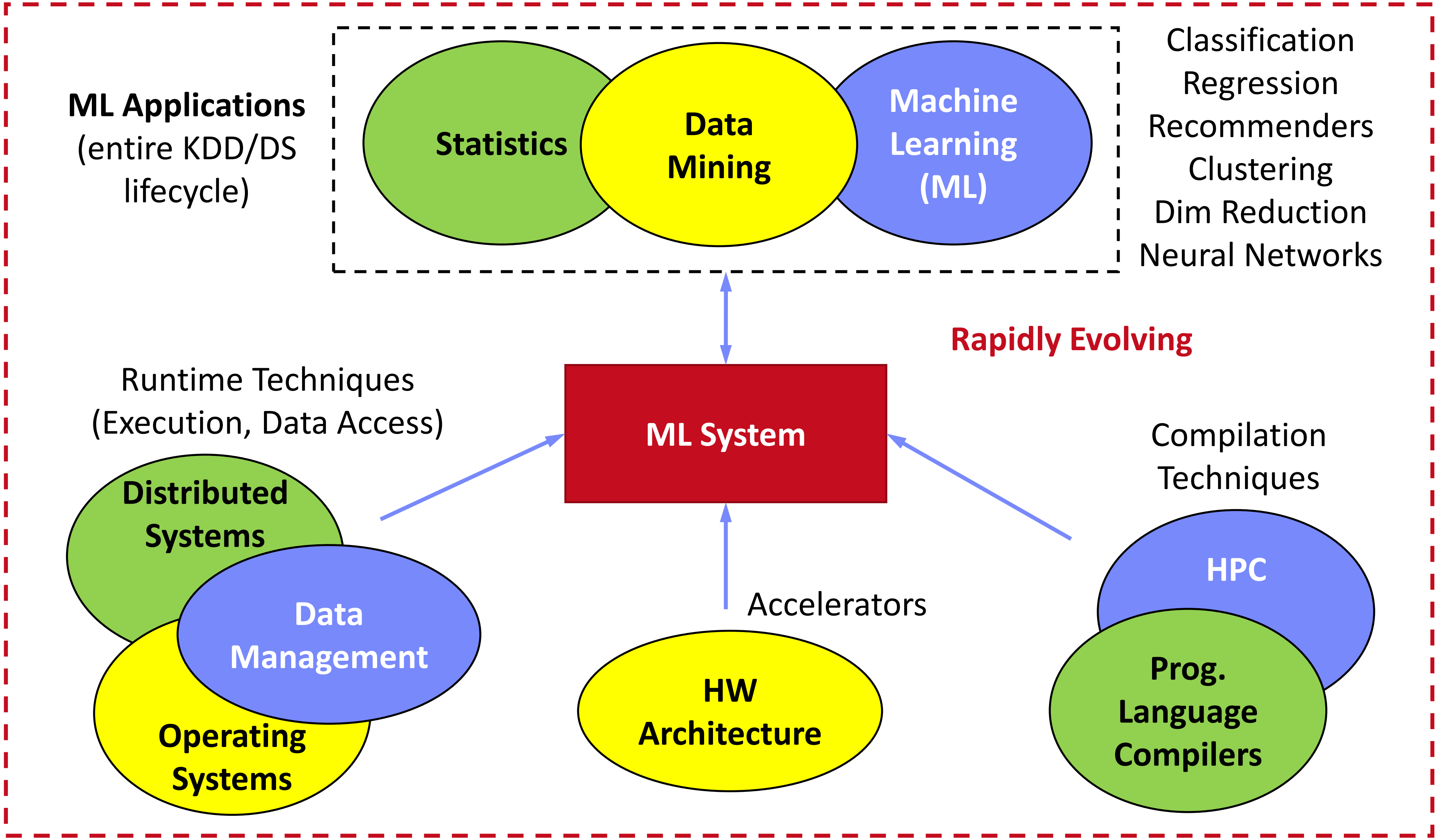
- The source code used to solve the individual sub-tasks
- A PDF report of up to 8 pages (10pt), including the names of all team members, a brief summary of how to run your code, and a description of the solutions to the individual sub-tasks.

Data and Reference Implementation: The task API header file, a reference implementation of the task interface, the test driver along with an example workload, and a Makefile are available at https://github.com/transactionalblog/sigmod-contest-2013. A detailed task description can be found here. Additionally, you can find both smaller and bigger datasets here.

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

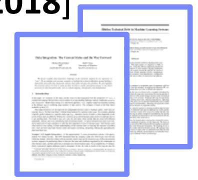
- **Existing ML Systems**
  - **#1 Numerical computing** frameworks
  - **#2 ML Algorithm libraries** (local, large-scale)
  - **#3 Linear algebra ML systems** (large-scale)
  - **#4 Deep neural network** (DNN) frameworks
  - **#5 Model management, and deployment**

- **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**

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

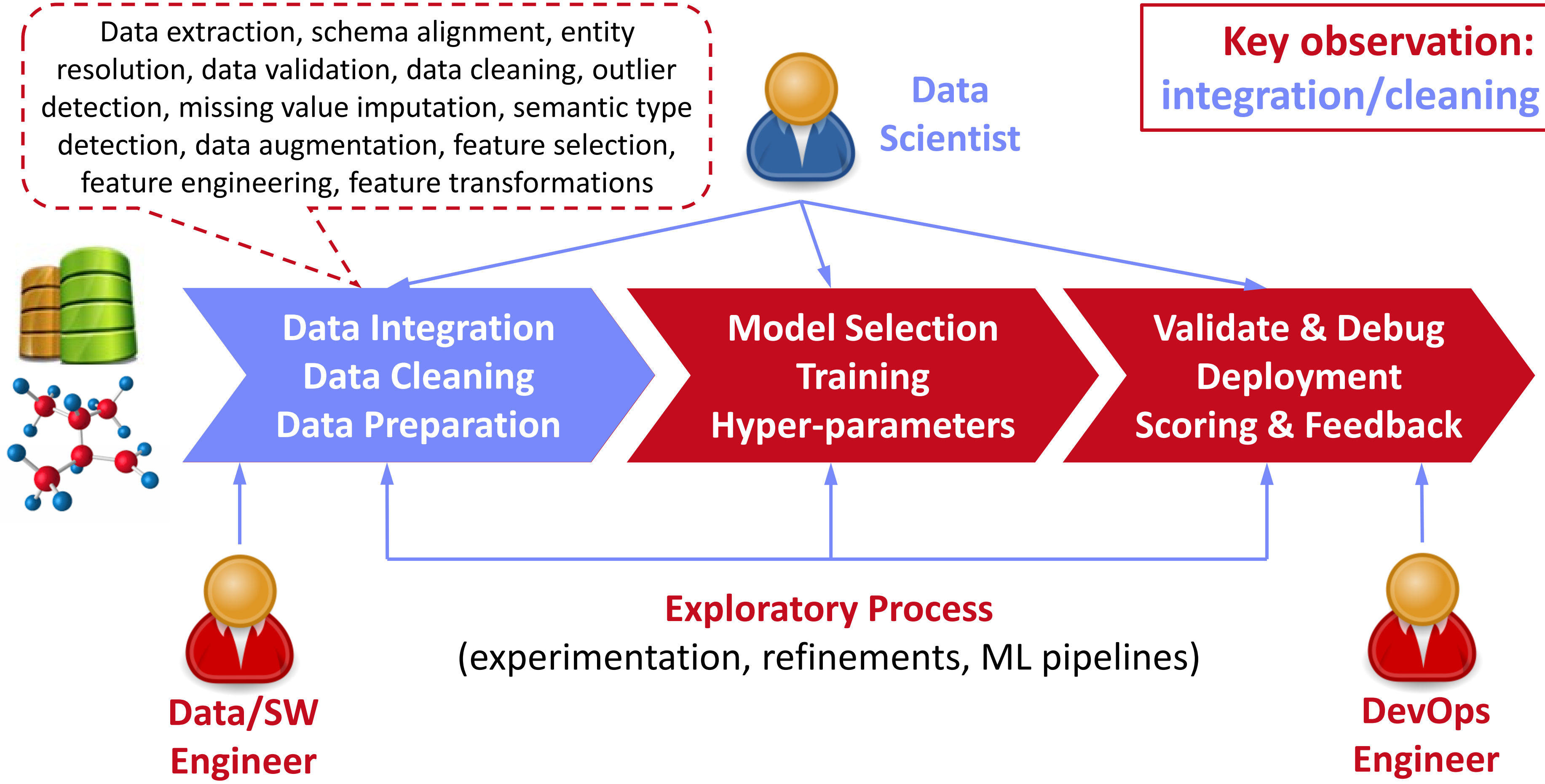
"Take these datasets and show value or competitive advantage"

[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

Apache SystemML™

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

[SIGMOD'15,'17,'19,'21abc,'23abc,'24ab]
[PVLDB'14,'16ab,'18,'22]
[ICDE'11,'12,'15]
[EDBT'25]
[CIDR'17,'20]
[VLDBJ'18]
[CIKM'22]
[DEBull'14]
[PPoPP'15]

In-Memory Single Node
(scale-up)

Hadoop or Spark Cluster
(scale-out)

Federated
(LA progs, PS)

Others:
Netezza
Apache Flink

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

Feature Selection

```
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)
} }
```

ML Algorithms

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

Linear Algebra Programs

```
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) ...}
```

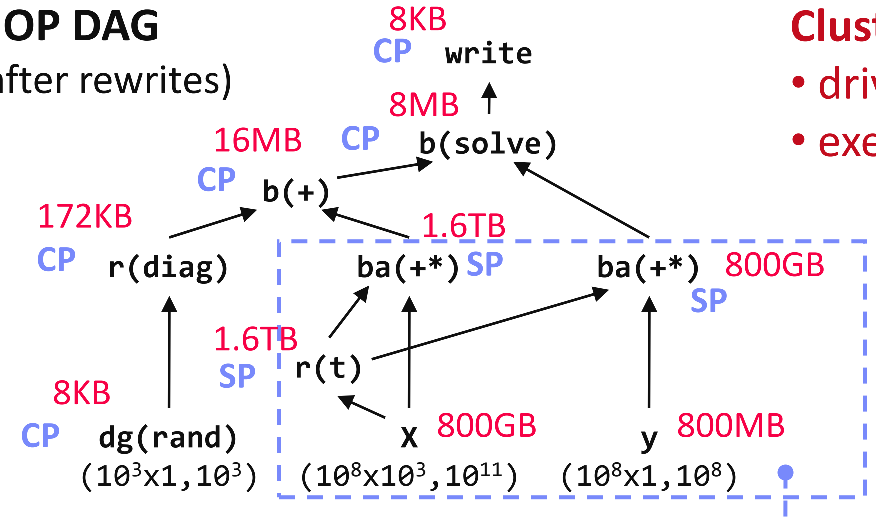
Basic HOP and LOP DAG Compilation

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);
```

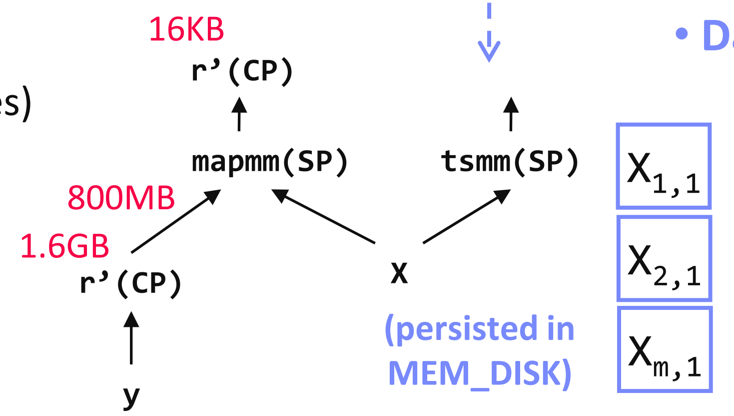
Scenario:
X: 10^8 x 10^3, 10^{11}
y: 10^8 x 1, 10^8

HOP DAG
(after rewrites)



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

LOP DAG
(after rewrites)

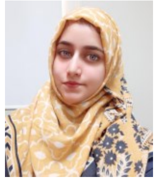


➔ **Distributed Matrices**
- **Fixed-size matrix blocks**
- **Data-parallel operations**

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

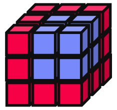
Data Cleaning Pipelines [SIGMOD'24a]

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



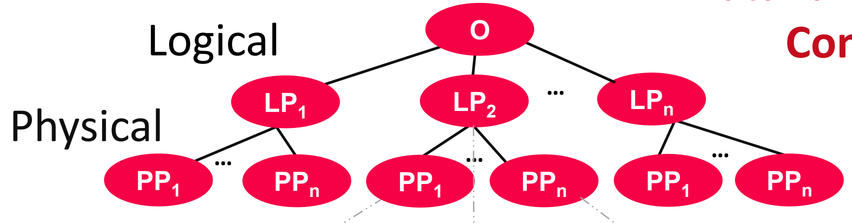
- **Automatic Generation of Cleaning Pipelines**
  - Library of robust, parameterized **data cleaning primitives**,
  - **Enumeration of DAGs** of primitives & **hyper-parameter optimization** (evolutionary, HB)

Data Samples



Target App

Rules/Objectives

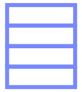


Data- and Task-parallel Computation

Outlier Detection → MVI → Deduplication → Resolve Mislabels

P_1: gmm → imputeFD → mergeDup → delML

P_n: outlierBySd → mice → delDup → voting



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]

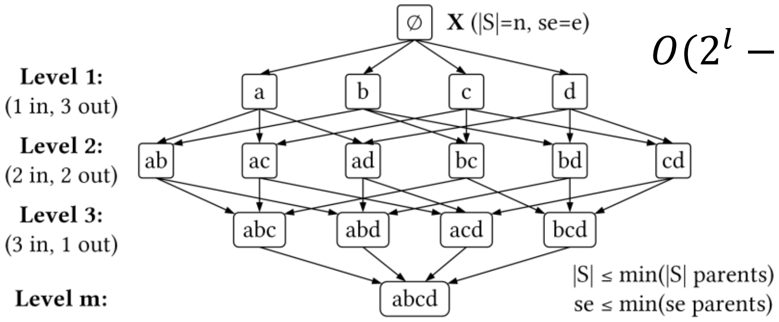
- **Problem Formulation**
  - **Intuitive slice scoring function**
  - Exact **top-k slice finding**
  - $|S| \geq \sigma \wedge sc(S) > 0,\ \alpha \in (0,1]$

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

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

slice error **slice size**

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



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

- **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

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

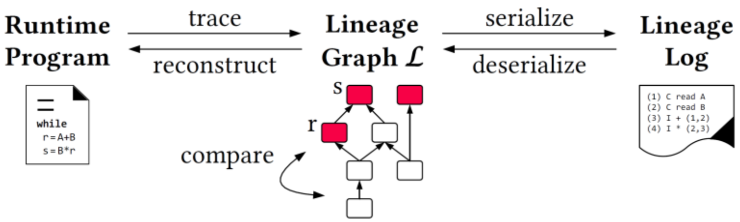
Data

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

== Level

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

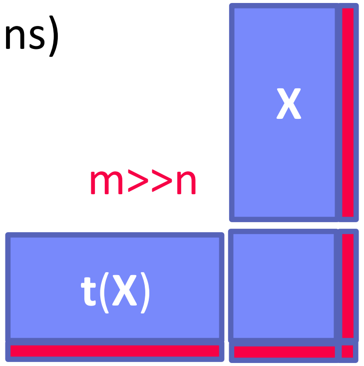


- **Full Reuse of Intermediates**
  - Before executing instruction, probe output lineage in cache `Map<Lineage, MatrixBlock>`
  - Cost-based/heuristic caching and eviction decisions (compiler-assisted)

```
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



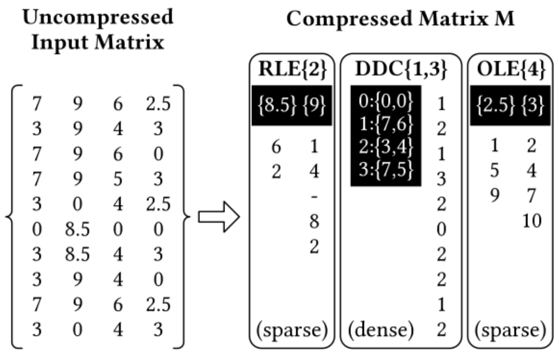

```
m_steplm = function(...) {
  while( continue ) {
    parfor( i in 1:n ) {
      if( !fixed[1,i] ) {
        Xi = cbind(Xg, X[,i])
        B[,i] = lm(Xi, y, ...)
    } }
    # add best to Xg (AIC)
} }
```

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

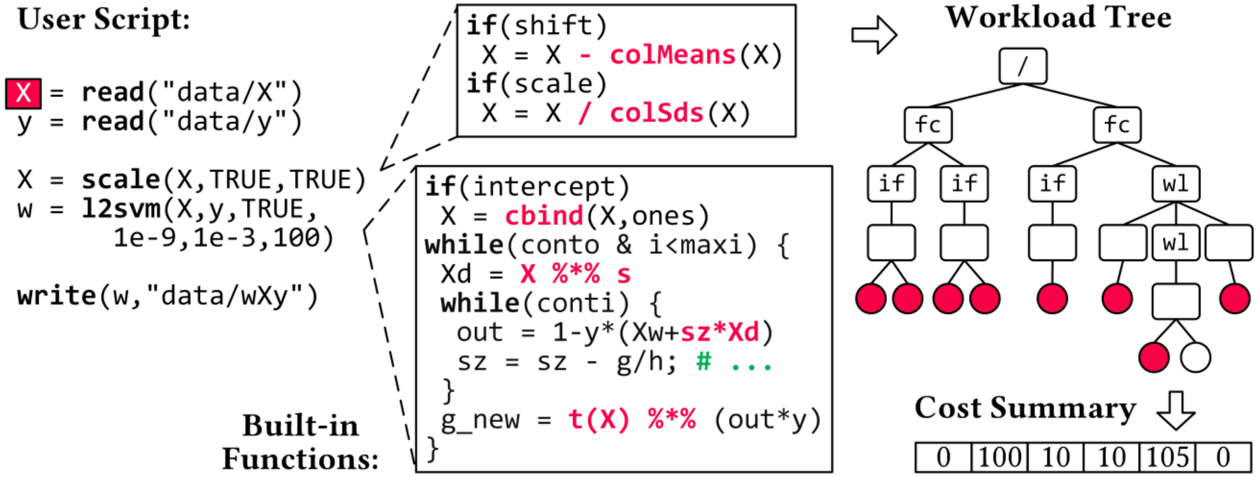
Compressed Linear Algebra Extended

[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)

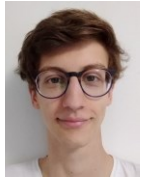


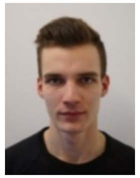
- **Workload-aware Compression**
  - Workload summary → compression
  - Compressed Representation → execution planning

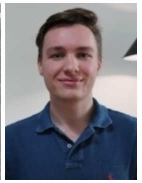


- **Next Steps**
  - Frame compression, compressed I/O
  - Compressed feature transformations
  - Morphing of compressed data

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

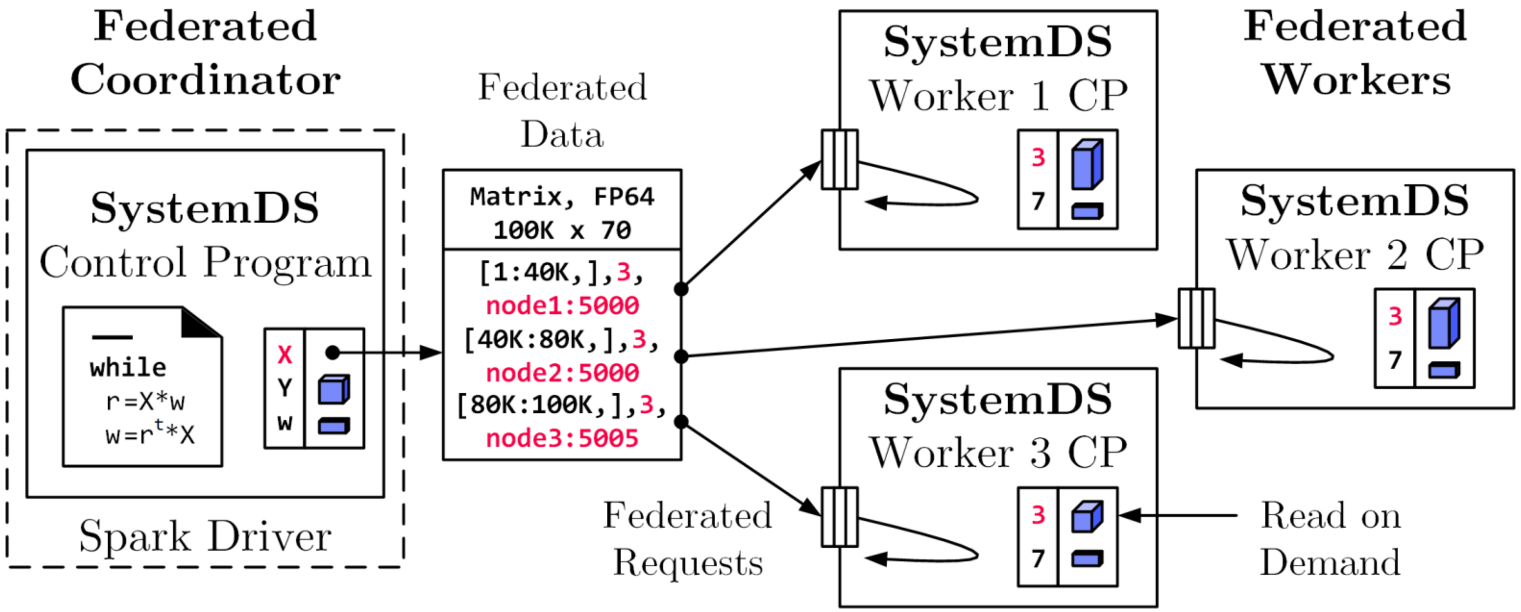







- **Federated Backend**
  - **Federated data** (matrices/frames) as meta data objects
  - **Federated linear algebra**, (and **federated parameter server**)

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



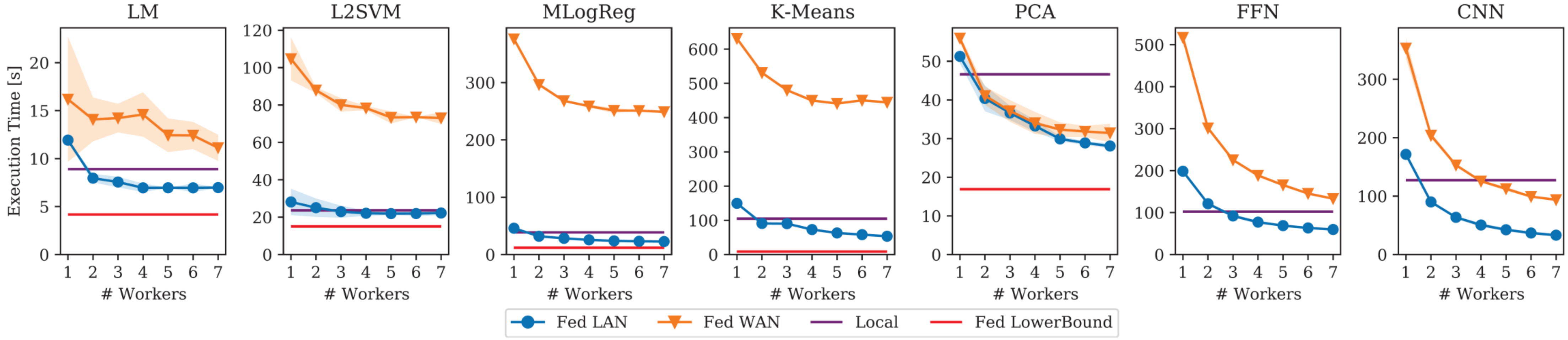
Federated Requests:
READ, PUT, GET, EXEC_INST,
EXEC_UDF, CLEAR

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

Federated Learning – Experiments

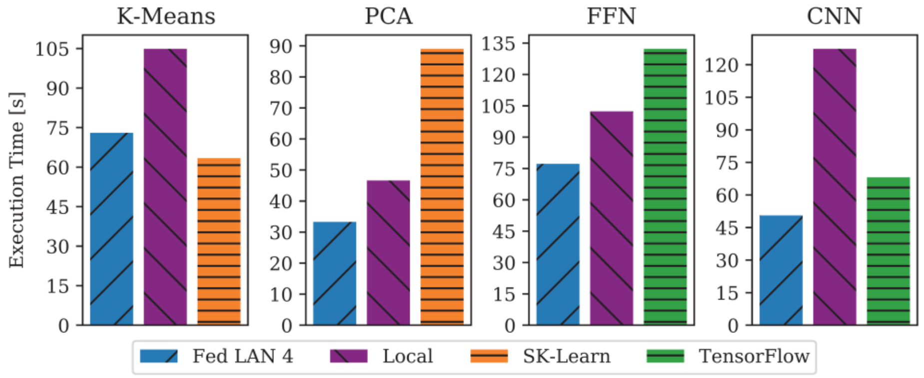
Reproducible Results

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

Thanks

- **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 and ML pipeline**

- **Next Lectures**
  - **02 Data Warehousing, ETL, and SQL/OLAP** [Oct 24]
  - **03 Message-oriented Middleware, EAI, and Replication** [Oct 31]