

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)

Announcements / Org



■ #1 Hybrid & Video Recording

- Hybrid lectures (in-person, zoom) with optional attendance

<https://tu-berlin.zoom.us/j/9529634787?pwd=R1ZsN1M3SC9BOU1OcFdmem9zT202UT09>

- Zoom **video recordings**, links from website

https://mboehm7.github.io/teaching/ws2627_dia/index.htm



■ #2 Course Registrations

- TU Berlin ISIS registrations as of Oct 11
- Bachelor/Master, CS/ISM/other ratios

~300

■ #3 Hackathon on **AI in Medicine**

- BIFOLD/Charité hackathon in planning
- Help us find suitable dates that work for you



#4 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)
- Ceenu George (professors)
- Nadine Karsten (scientific staff)
- Maria Fleßner-Jung (mentoring)
- N.N. (students)



- Mission Statement

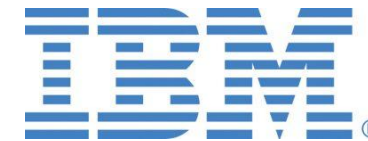
- Account for heterogeneity and complexity of modern societies at TU Berlin
- #1 **Treat all persons with fairness and respect**
- #2 **Ensure a safe environment for all**
- #3 **Comply with our duty of care towards others**
- #4 **Actively support the implementation of the above guidelines and contribute**

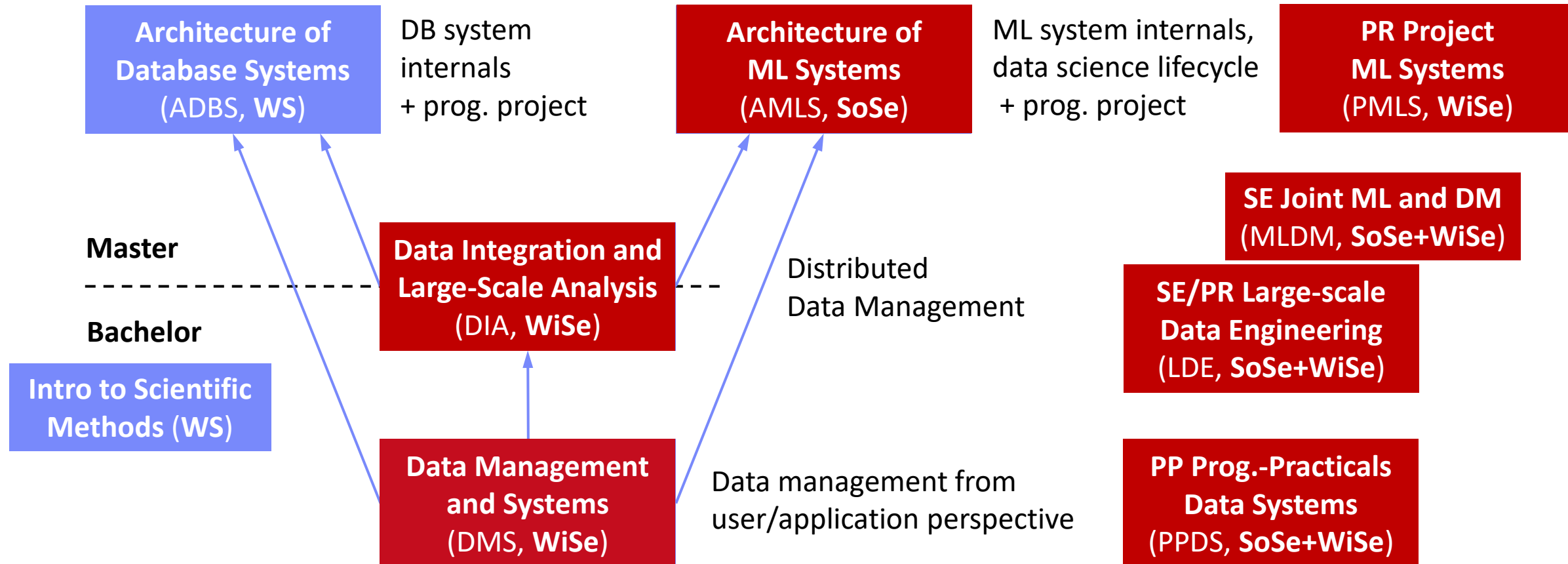
Contact: private email,
eeecs-TB-awareness@win.tu-berlin.de,
or AwAn@dams.tu-berlin.de

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





Agenda



- Course Organization
- Course Motivation and Goals
- Course Outline and Projects/Exercise
- Excursus: [Apache SystemDS](#)

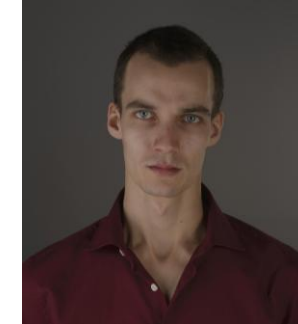
Course Organization

Course Logistics



■ Staff

- **Lecturer:** Prof. Dr. Matthias Boehm, DAMS
- **Teaching Assistant:** David Weissteiner, 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
- **Weekly lectures** (**Thu 4.15pm sharp**, in-person & zoom livestreaming/recording), **optional attendance**
- **Mandatory exercises or programming project** (3 ECTS), office hour **Wed 2pm-3pm** (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/ws2627_dia/index.htm (public)
- <https://isis.tu-berlin.de/user/index.php?id=49228> (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:** Wednesday 2-3pm, starting in 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 04, 4pm** / **Feb 25, 4pm** / **Mar 18, 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
- Different than “Data Integration: Algorithms and Systems (DI)”

Course Motivation and Goals

■ Terminology

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

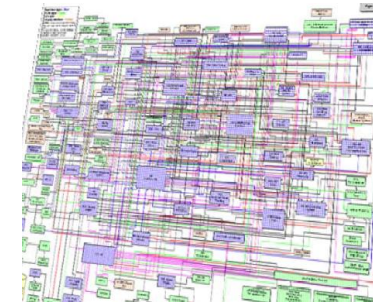
■ Heterogeneous IT Infrastructure

- Common enterprise IT infrastructure contains >100s of **heterogeneous and distributed systems and applications**
- E.g., health care data management: 20 - 120 systems

■ Multi-Modal Data (example health care)

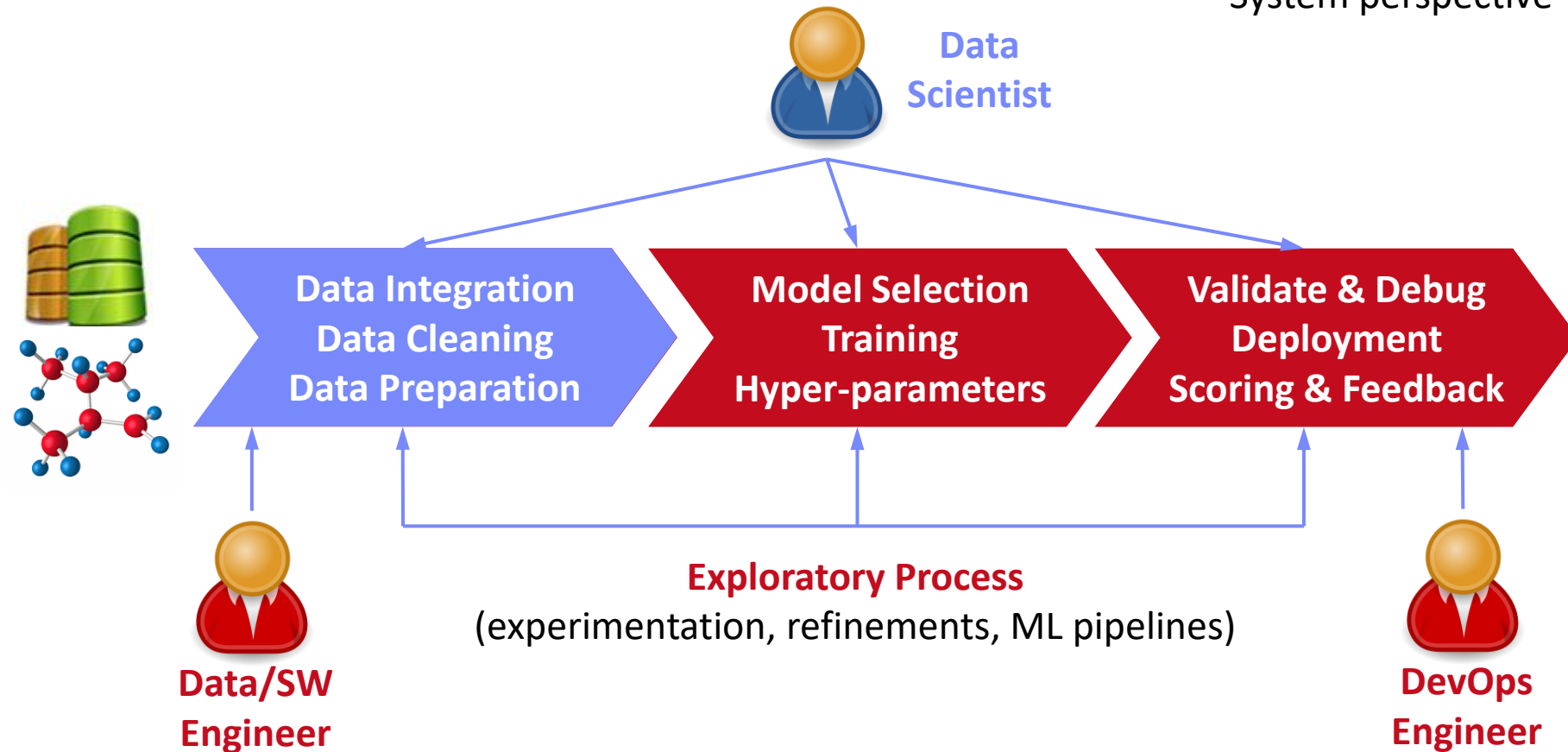
- **Structured patient data**, patient records incl. prescribed drugs
- **Knowledge base** drug APIs (active pharmaceutical ingredients) + interactions
- **Doctor notes** (text), diagnostic codes, outcomes
- **Radiology images** (e.g., MRI scans), **patient videos**
- **Time series** (e.g., EEG, ECoG, heart rate, blood pressure)

[Credit: Albert Maier]



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

Data-centric View:
Application perspective
Workload perspective
System perspective



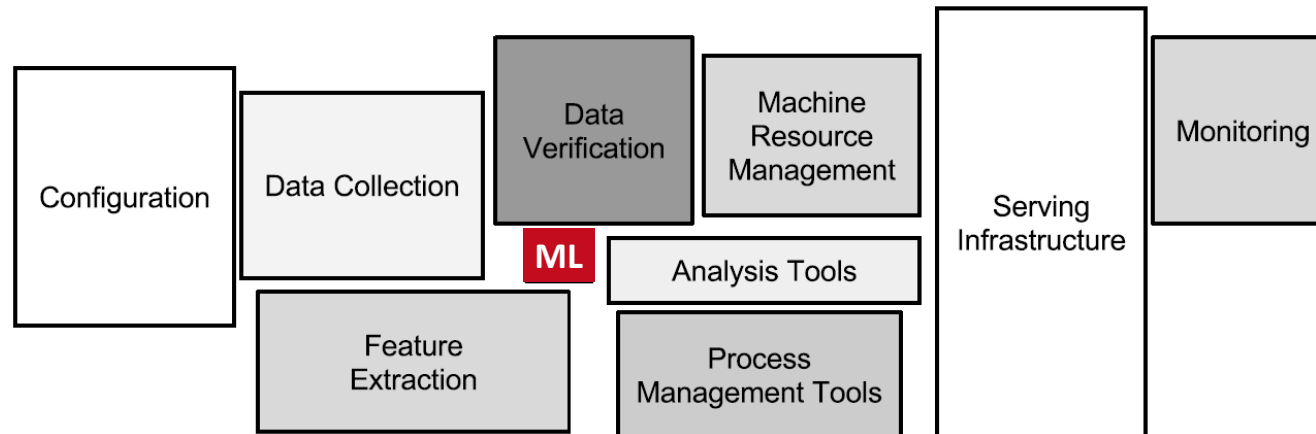
The 80% Argument



- **Data Sourcing Effort**

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

- **Technical Debts in ML Systems**



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



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

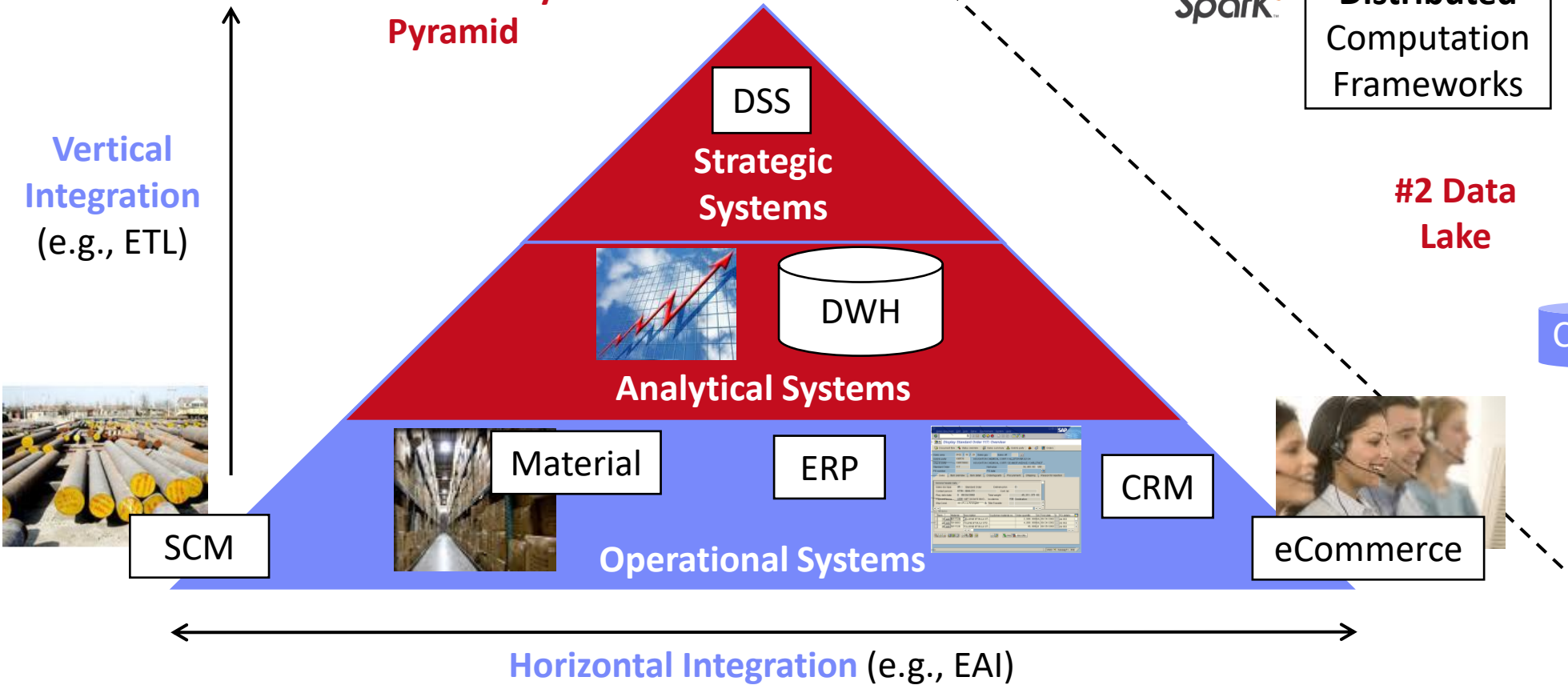


- Glue code, pipeline jungles, dead code paths
- Plain-old-data types (arrays), multiple languages, prototypes
- Abstraction and configuration debts
- Data testing, reproducibility, process management, and cultural debts

Complementary System Architectures

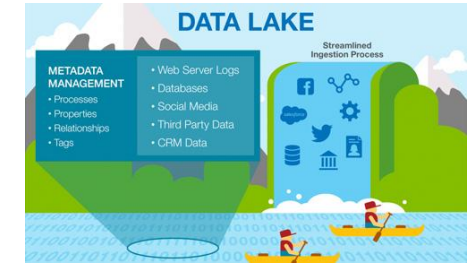


#1 Information System Pyramid



Distributed Computation Frameworks

#2 Data Lake



Audio, Image, Video, Text, Streams, Logs

Catalogs

Distributed Data Stores



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 15]
- **02 Data Warehousing, ETL, and SQL/OLAP** [Oct 22]
- **03 Message-oriented Middleware, EAI, and Replication** [Oct 29]

Key Integration Techniques

- **04 Schema Matching and Mapping** [Nov 05]
- **05 Entity Linking and Deduplication** [Nov 12]
- **06 Data Cleaning and Data Fusion** [Nov 19]
- **07 Data Provenance and Catalogs** [Nov 26]

Part B: Large-Scale Data Management & Analysis



Cloud Computing

- **08 Cloud Computing Foundations** [Dec 03]
- **09 Cloud Resource Management and Scheduling** [Dec 10]
- **10 Distributed Data Storage** [Dec 17]

Large-Scale Data Analysis

- **11 Distributed, Data-Parallel Computation** [Jan 14]
- **12 Distributed Stream Processing** [Jan 21]
- **13 Distributed Machine Learning Systems** [Jan 28]
+ Q&A and Exam Preparation

Overview Projects or Exercises



■ Team

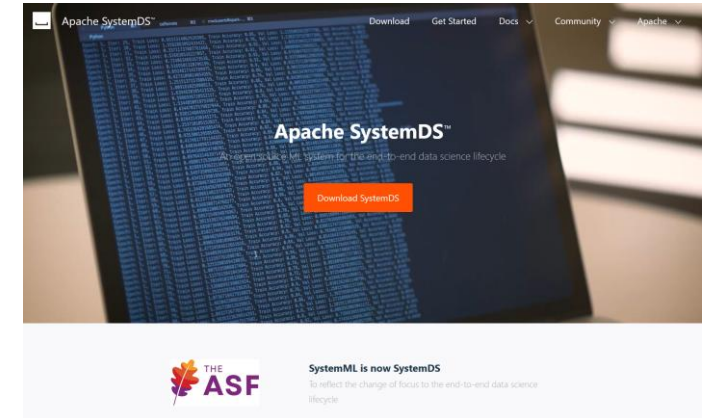
- **1-3 person teams** (w/ clearly separated responsibilities)

■ 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:** “Ship Collision Detection” (most students)

■ Timeline

- **Oct 23:** Binding project/exercise selection (via google forms)
- **Jan 28:** Project/exercise submission deadline



<https://forms.gle/XcmRZxmxQ9AZvVjj7>

DIA Exercise (alternative to projects), cont.



DIA WiSe 2026/27: Exercise – Ship Collision Detection

Published: Oct 11, 2026 (Last Update: Oct 11, 2026)

Deadline: Jan 28, 2027; 11.59pm

This exercise is an alternative to the DIA programming projects and provides practical experience with PySpark. The task is to identify near-collision scenarios in AIS¹ data. The objectives are to implement data analysis methods for both local and distributed execution, and to integrate optimization techniques to improve runtime and scalability. The final submission (in ISIS) is a zip archive named `DIA_Exercise_<studentID>.zip` (max. 5MB), containing:

1. The source code used to solve the tasks, comprising the files `collisionDetectionPandas.py` and `collisionDetectionSpark.py`.
2. A PDF report of up to 3 pages (12pt), including the concise description of the implemented tasks, as well as a contribution statement table with the names of all team members and task-specific percentages.

Data Source: The Danish Maritime Authority provides historical AIS data for free². The data are provided as CSV files for each individual day, each containing about 15 million AIS reports which corresponds to a file size of approximately 2.5GB. Since these data sizes are too large for local experiments, we provide a pre-filtered version of the dataset from 01/01/2025 that includes the most interesting vessels for our task. The dataset contains 26 features, from which we are only interested in MMSI³, # Timestamp, Longitude, Latitude, SOG⁴, Heading⁵.

[https://github.com/damslab/datasets/tree/master/ais_data]

The screenshot shows the GitHub repository page for 'ais_data' by 'damslab'. The repository is described as 'Add pre-filtered AIS data for DIA 2026 Exercise'. It has 4d19406 commits, 20 hours ago, and a history link. The file list shows a directory '..', a 'README.md' file, and a 'aisdk-2025-01-01-prefiltered.csv' file, all with the commit message 'Add pre-filtered AIS data for DIA 2026 Exercise' and a commit date of 20 hours ago. The 'README.md' file is expanded, showing the title 'AIS Data', an 'Overview' section, and a 'Structure' section. The 'Overview' section states that the Danish Maritime Authority provides historical AIS data free for download on their website, and the dataset here is a filtered subset from 01/01/2025. The 'Structure' section states that the data is provided as a CSV file with the following columns:

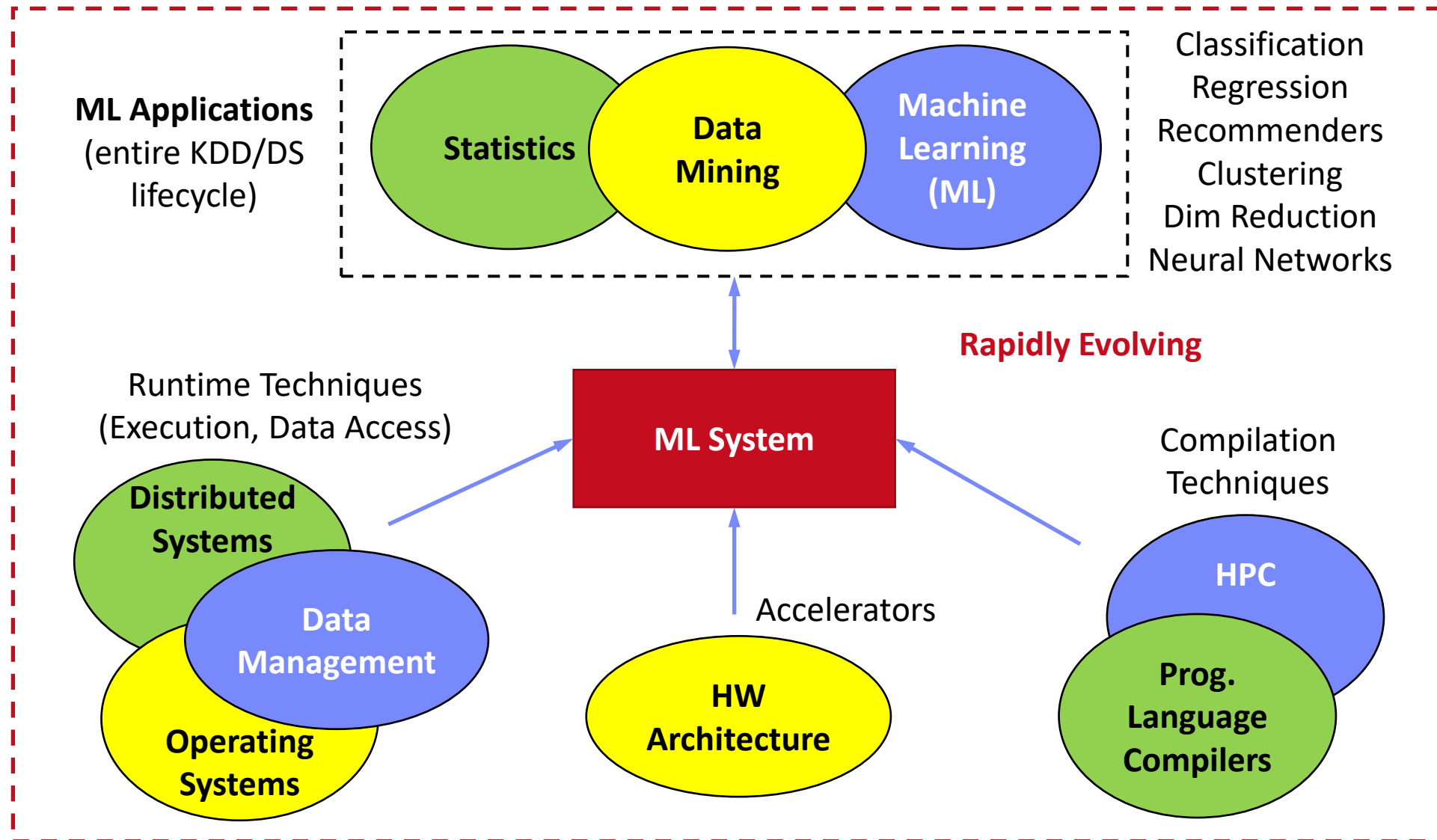
Column name	Description
MMSI	MMSI number of vessel
# Timestamp	Timestamp from the AIS basestation, format: 31/12/2015 23:59:59
Latitude	Latitude of message report (e.g. 57.8794)
Longitude	Longitude of message report (e.g. 17.9125)
SOG	Speed over ground from AIS message if available
Heading	Heading from AIS message if available

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



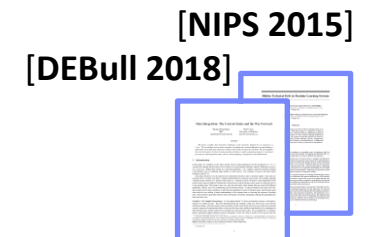
Exploratory Data-Science Lifecycle

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

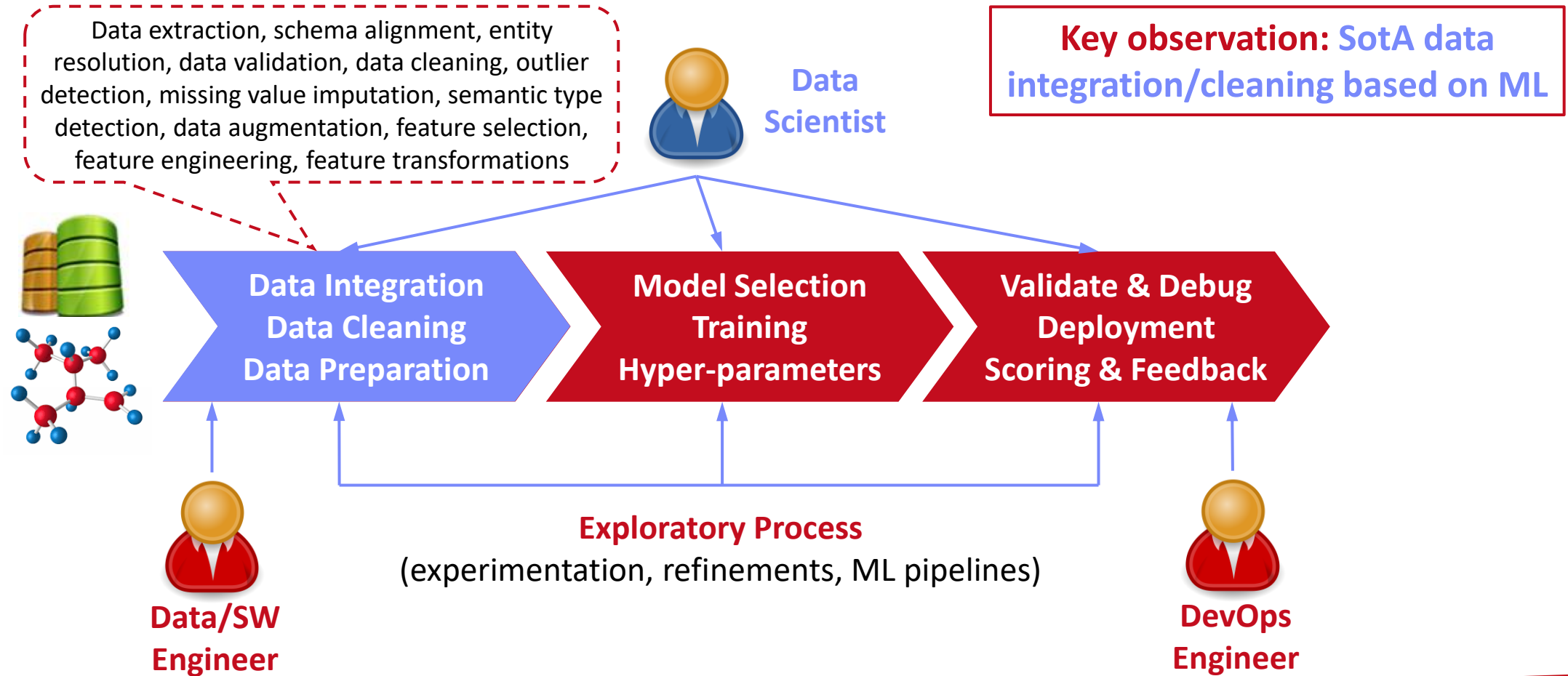
“Take these datasets
and show value or
competitive advantage”

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

In-Memory Single Node
(scale-up)

Hadoop or Spark Cluster
(scale-out)

Federated
(LA progs, PS)



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][BTW'25ab]
[CIDR'17,'20]
[VLDBJ'18] [TODS'26]
[CIKM'22]
[DEBull'14]
[PPoPP'15]

In-Progress:

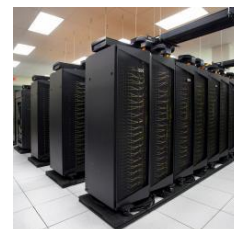
GPU



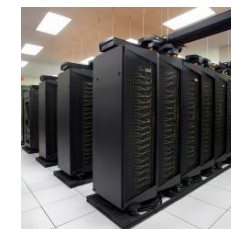
since 2014/16



since 2012



since 2010/11



since 2015



since 2019

Others:

Netezza
Apache Flink

- Example:
Stepwise
Linear
Regression

User Script

```
X = read('features.csv')
Y = read('labels.csv')
[B,S] = step1m(X, Y,
  icpt=0, reg=0.001)
write(B, 'model.txt')
```

Built-in Functions

```
m_step1m = 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_lmCG = function(...) {
  while( i<maxi&nr2>tgt ) {
    q = (t(X) %**% (X %**% p))
      + lambda * p
    beta = ...
  }
}
```

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

Linear
Algebra
Programs

ML
Algorithms

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

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;
...
```

Scenario:

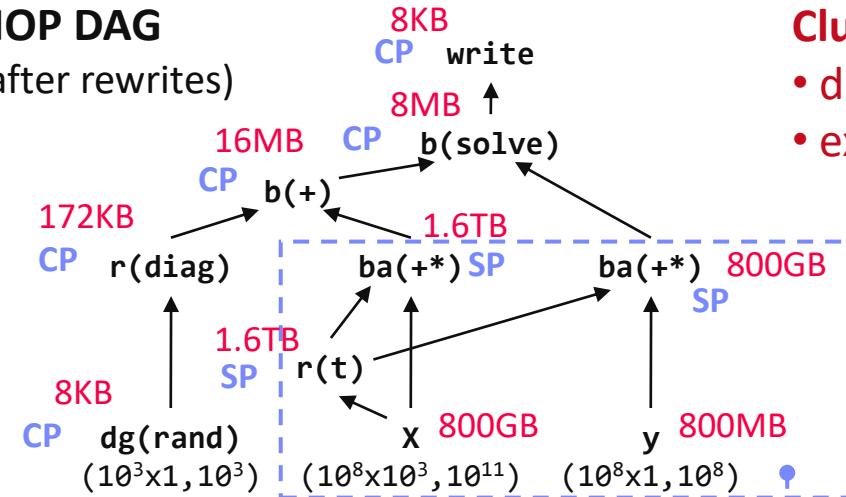
X: $10^8 \times 10^3, 10^{11}$
y: $10^8 \times 1, 10^8$

```
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);
```

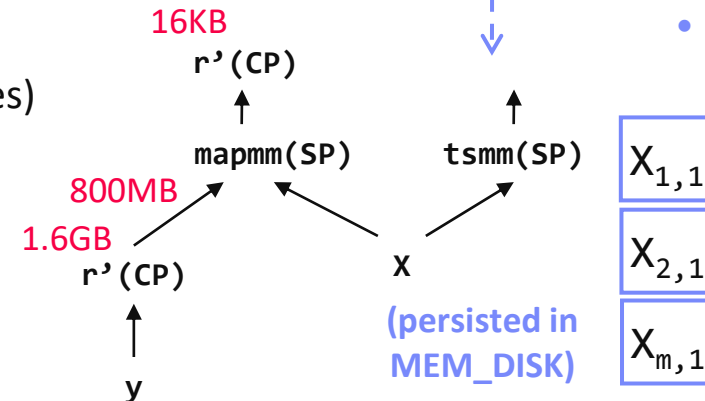
HOP DAG

(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

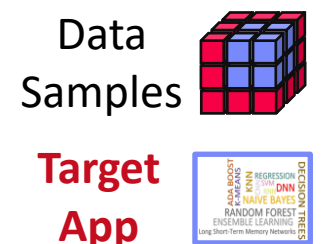


[best paper runner-up
w/ Shafaq and Roman]

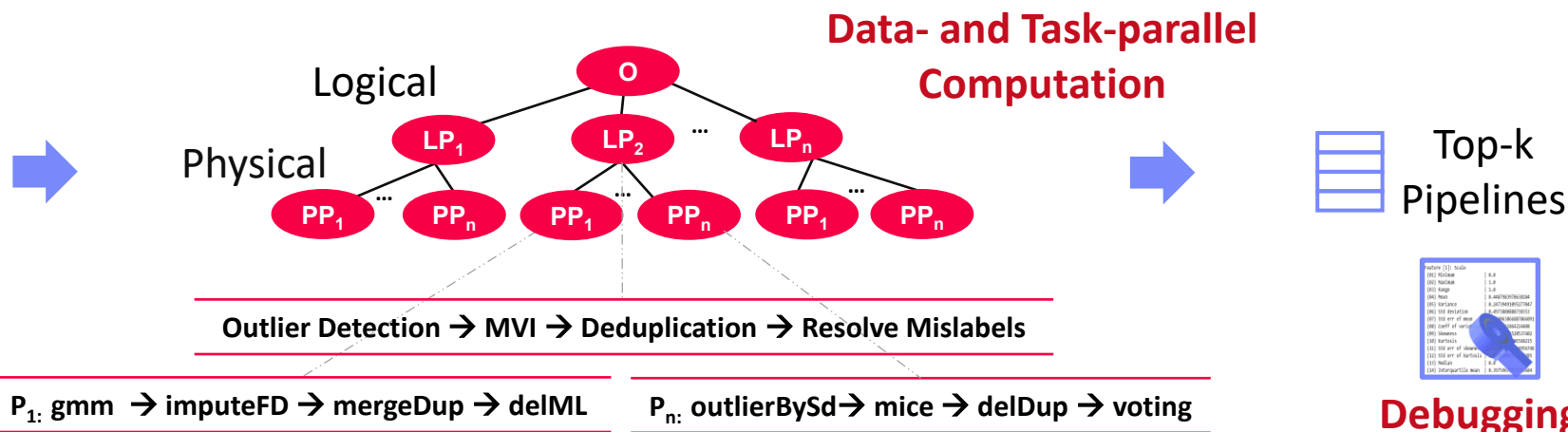


Automatic Generation of Cleaning Pipelines

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



Rules/Objectives



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



■ Problem Formulation

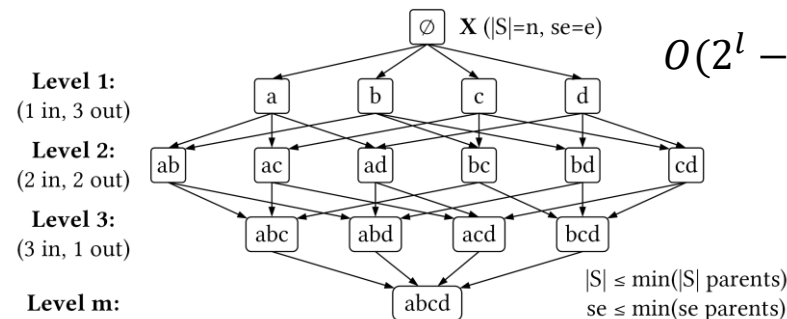
- Intuitive slice scoring function
- Exact **top-k slice finding**
- $|S| \geq \sigma \wedge sc(S) > 0, \alpha \in (0,1]$

$$\begin{aligned}
 sc &= \alpha \left(\frac{\bar{e}(S)}{\bar{e}(X)} - 1 \right) - (1 - \alpha) \left(\frac{|X|}{|S|} - 1 \right) \\
 &= \alpha \left(\frac{|X|}{|S|} \cdot \frac{\sum_{i=1}^{|S|} es_i}{\sum_{i=1}^{|X|} e_i} - 1 \right) - (1 - \alpha) \left(\frac{|X|}{|S|} - 1 \right)
 \end{aligned}$$

slice error
slice size

■ Properties & Pruning

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



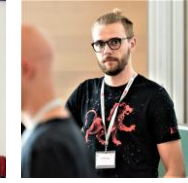
■ Linear-Algebra-based Slice Finding

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

	<table><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr></table>	0	1	0	1	0	1	1	0	0	0	0	0	0	1	0	Candidate Slices																																		
0	1	0																																																	
1	0	1																																																	
1	0	0																																																	
0	0	0																																																	
0	1	0																																																	
Data	<table><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr></table>	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	<table><tr><td>0</td><td>2</td><td>0</td></tr><tr><td>0</td><td>2</td><td>0</td></tr><tr><td>2</td><td>0</td><td>1</td></tr><tr><td>0</td><td>2</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>2</td><td>0</td><td>1</td></tr></table>	0	2	0	0	2	0	2	0	1	0	2	0	1	1	1	2	0	1	== Level
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																																															
0	2	0																																																	
0	2	0																																																	
2	0	1																																																	
0	2	0																																																	
1	1	1																																																	
2	0	1																																																	

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

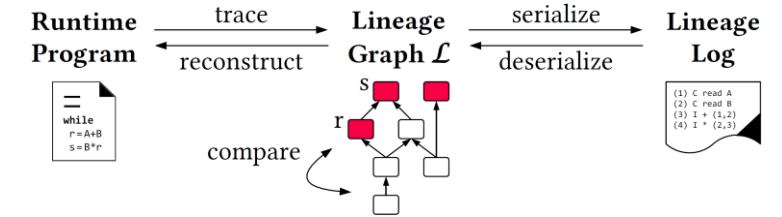
■ Full Reuse of Intermediates

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

■ Partial Reuse of Intermediates

- **Problem:** Often partial result overlap
- Reuse partial results via dedicated rewrites (compensation plans)
- Example: step1m

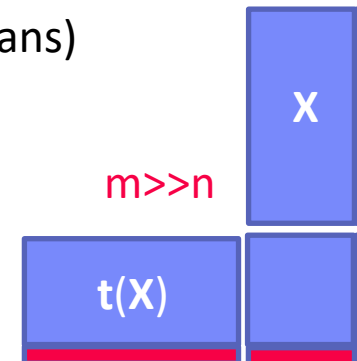
■ Next Steps: multi-backend, unified mem mgmt



```
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) ...}
```

```
m_step1m = function(...) {
  while( continue ) {
    parfor( i in 1:n ) {
      if( !fixed[1,i] ) {
        Xi = cbind(Xg, X[,i])
        B[,i] = lm(Xi, y, ...)
      }
    }
    # add best to Xg (AIC)
  } }
```



Compressed Linear Algebra Extended

[PVLDB'16a, VLDBJ'18, SIGMOD'23a,
PVLDB'26a]



Lossless Matrix Compression

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

Uncompressed Input Matrix	Compressed Matrix M		
$\begin{Bmatrix} 7 & 9 & 6 & 2.5 \\ 3 & 9 & 4 & 3 \\ 7 & 9 & 6 & 0 \\ 7 & 9 & 5 & 3 \\ 3 & 0 & 4 & 2.5 \\ 0 & 8.5 & 0 & 0 \\ 3 & 8.5 & 4 & 3 \\ 3 & 9 & 4 & 0 \\ 7 & 9 & 6 & 2.5 \\ 3 & 0 & 4 & 3 \end{Bmatrix}$	$\begin{Bmatrix} \text{RLE}\{2\} \\ \{8.5\} \{9\} \\ 6 & 1 \\ 2 & 4 \\ - \\ 8 \\ 2 \end{Bmatrix}$ (sparse)	$\begin{Bmatrix} \text{DDC}\{1,3\} \\ 0:\{0,0\} \\ 1:\{7,6\} \\ 2:\{3,4\} \\ 3:\{7,5\} \end{Bmatrix}$ (dense)	$\begin{Bmatrix} \text{OLE}\{4\} \\ \{2.5\} \{3\} \\ 1 & 2 \\ 5 & 4 \\ 9 & 7 \\ 10 \end{Bmatrix}$ (sparse)

Workload-aware Compression

- Workload summary
→ compression
- Compressed Representation
→ execution planning

User Script:

```
X = read("data/X")
y = read("data/y")

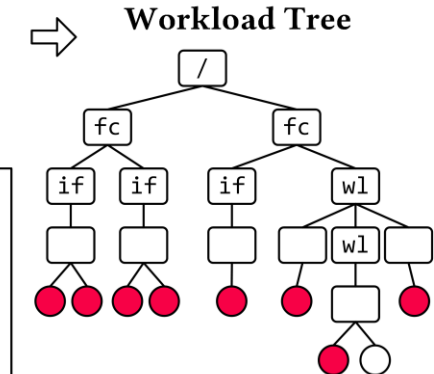
X = scale(X, TRUE, TRUE)
w = l2svm(X, y, TRUE,
          1e-9, 1e-3, 100)

write(w, "data/wXy")
```

Built-in
Functions:

```
if(shift)
  X = X - colMeans(X)
if(scale)
  X = X / colSds(X)

if(intercept)
  X = cbind(X, ones)
while(conto & i<maxi) {
  Xd = X %*% s
  while(conti) {
    out = 1-y*(Xw+sz*Xd)
    sz = sz - g/h; # ...
  }
  g_new = t(X) %*% (out*y)
}
```



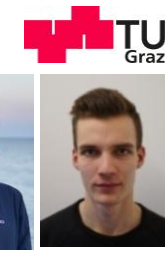
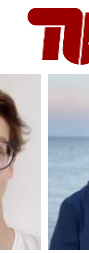
Cost Summary

0	100	10	10	105	0
---	-----	----	----	-----	---

BWARE: Compressed Feature Engineering

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

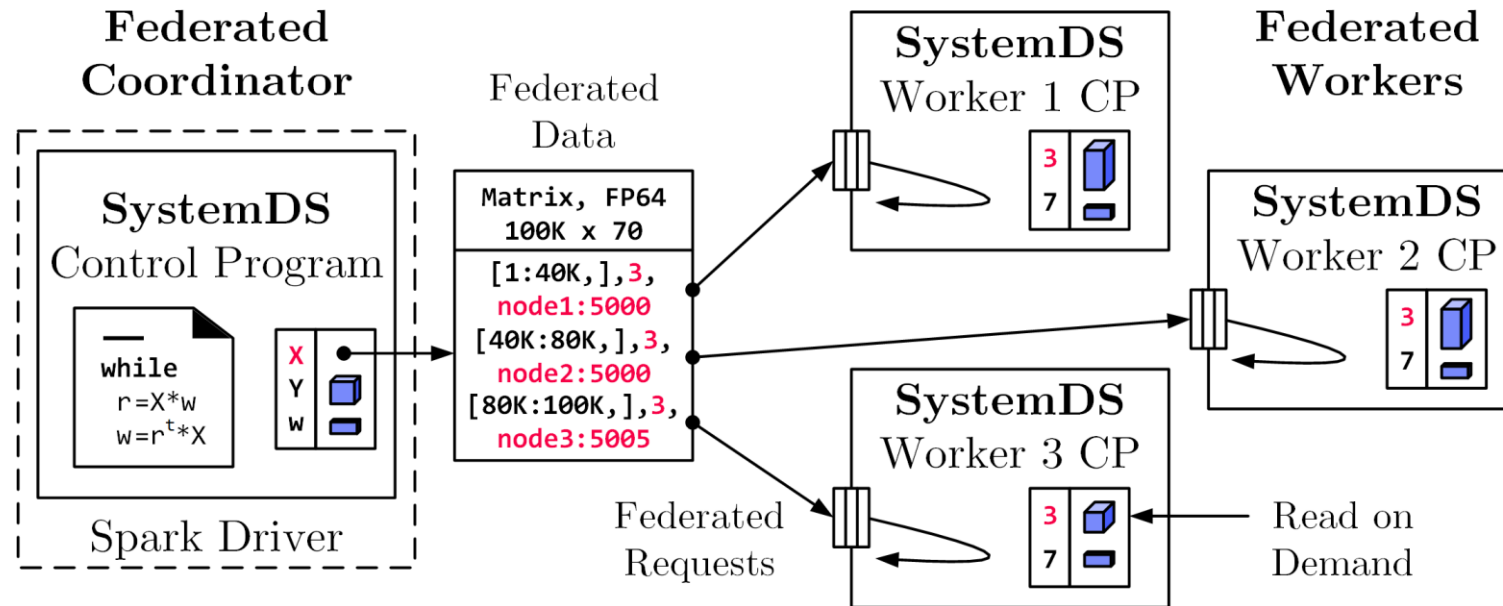
Federated Learning [SIGMOD'21c, CIKM'22]



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)));
```



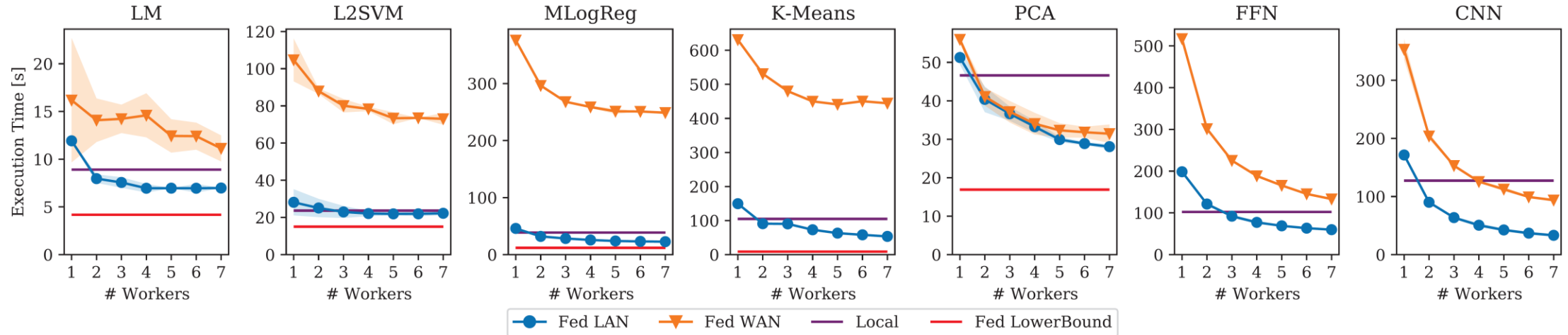
Federated Requests:
READ, PUT, GET, EXEC_INST,
EXEC_UDF, CLEAR

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

Federated Learning – Experiments

Reproducible Results

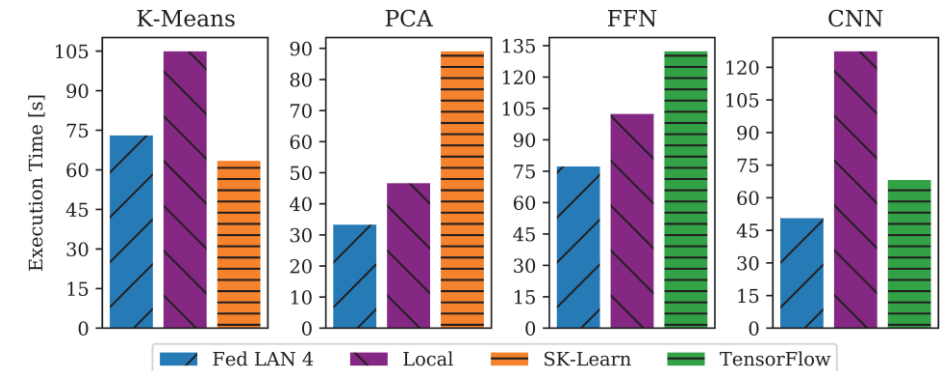
OPEN ACCESS



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



■ 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 / analytics**
- Project selection by Oct 23 EOD

■ Next Lectures

- 02 **Data Warehousing, ETL, and SQL/OLAP** [Oct 22]
- 03 **Message-oriented Middleware, EAI, and Replication** [Oct 29]

Thanks

<https://forms.gle/XcmRZxmxQ9AZvVjj7>