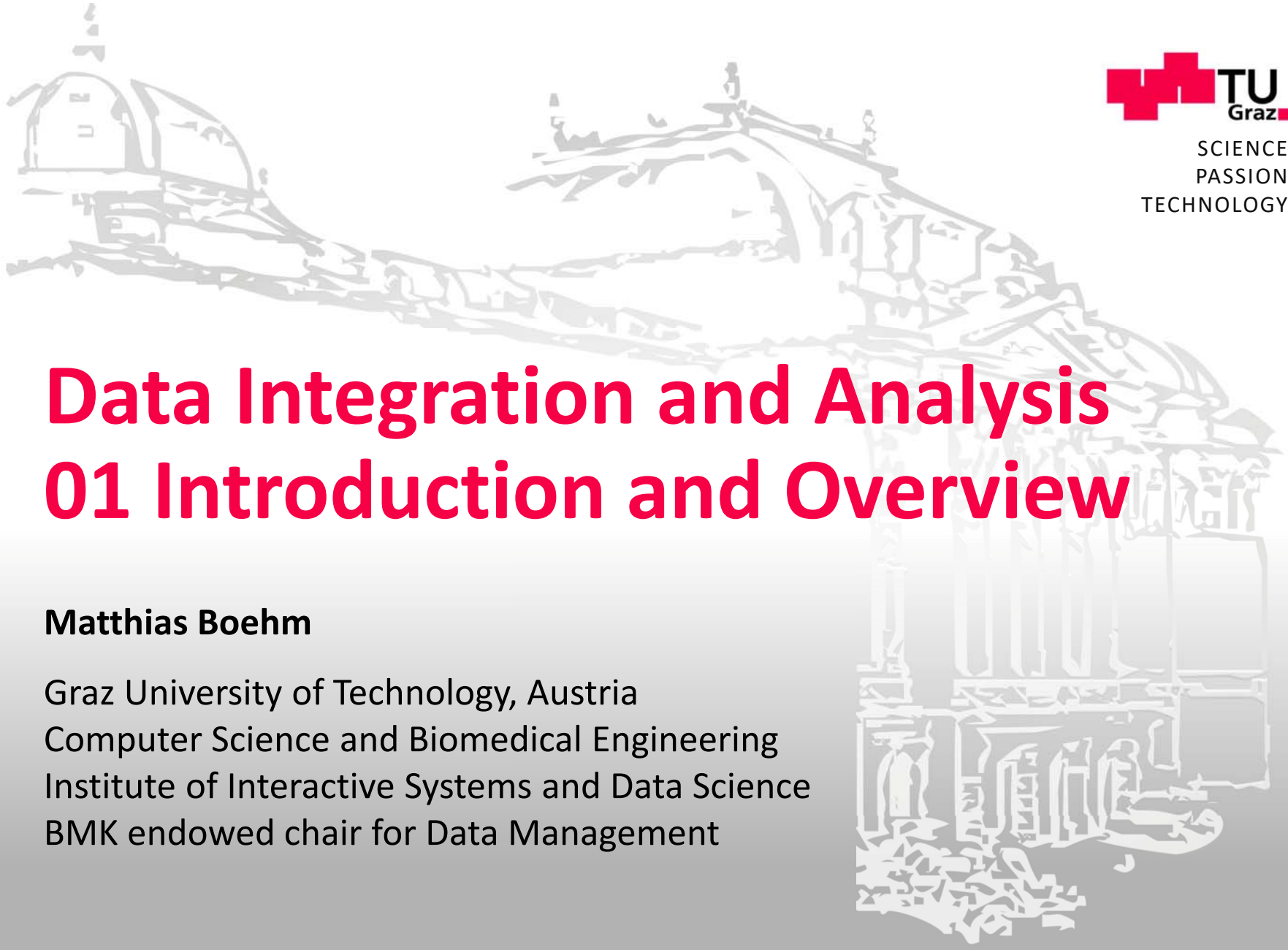


SCIENCE
PASSION
TECHNOLOGY

Data Integration and Analysis
01 Introduction and Overview

Matthias Boehm

Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMK endowed chair for Data Management

Last update: Oct 08, 2020

ISDS

Announcements/Org

- **#1 Video Recording**
  - Link in TeachCenter & TUbe (lectures will be public)
  - Optional attendance (independent of COVID)

TUbe

- **#2 Course Registration** (as of Oct 08)
  - Data Integration and Large-Scale Analysis:

96 (2)

- **#3 COVID-19 Restrictions** (HS i5)
  - Max 25% room capacity (TC registrations)

- **#4 Startup Incubator "Gründungsgarage"**
  - 5min-overview by Martin Glinik in DM (see recording)
  - If interested, apply by **Oct 11**

Gründungs garage

Agenda

- **Data Management Group**
- **Course Organization**
- **Course Motivation and Goals**
- **Course Outline and Projects**
- **Excursus: Apache SystemDS**

Data Management Group

https://damslab.github.io/

About Me

- **09/2018 TU Graz**, Austria
  - BMK endowed chair for data management
  - **Data management for data science**
    (ML systems internals, end-to-end data science lifecycle)

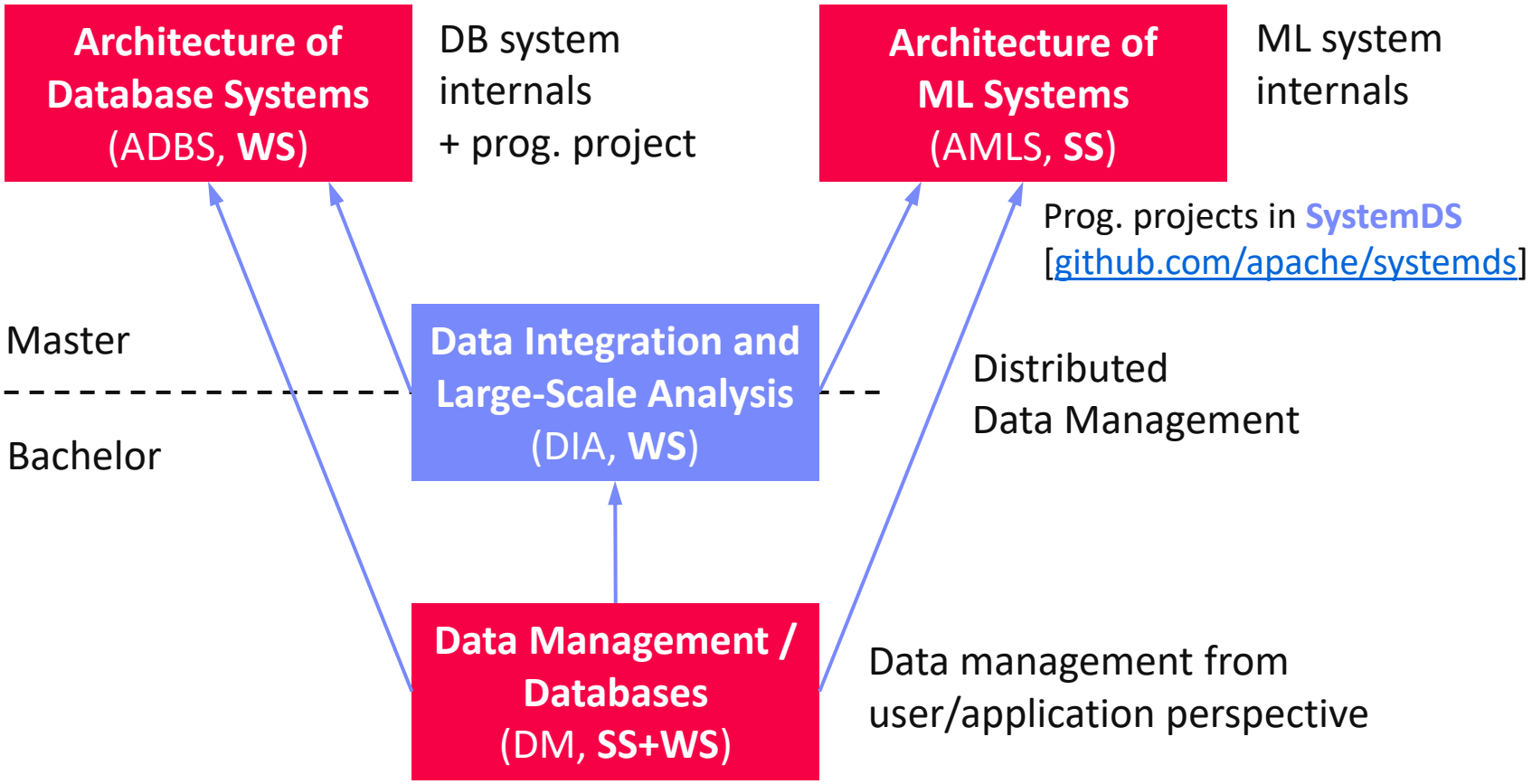
https://github.com/apache/systemds

- **2012-2018 IBM Research – Almaden**, USA
  - Declarative large-scale machine learning
  - Optimizer and runtime of **Apache SystemML**

- **2011 PhD TU Dresden**, Germany
  - Cost-based optimization of integration flows
  - Systems support for time series forecasting
  - In-memory indexing and query processing

TECHNISCHE UNIVERSITÄT DRESDEN
DB group

Data Management Courses



Course Organization

Basic Course Organization

- **Staff**
  - Lecturer: Univ.-Prof. Dr.-Ing. Matthias Boehm, ISDS
  - Assistant: M.Sc. Shafaq Siddiqi, ISDS
- **Language**
  - Lectures and slides: **English**
  - Communication and examination: **English**/**German**
- **Course Format**
  - VU 2/1, **5 ECTS** (2x 1.5 ECTS + 1x 2 ECTS), bachelor/master
  - **Weekly lectures** (**Fri 3pm**, including **Q&A**), **attendance optional**
  - **Mandatory exercises or programming project** (2 ECTS)
  - **Recommended papers** for additional reading on your own
- **Prerequisites**
  - **Preferred:** course Data Management / Databases is very good start
  - **Sufficient:** basic understanding of SQL / RA (or willingness to fill gaps)
  - Basic programming skills (Python, R, Java, C++)

Course Logistics

- **Website**
  - https://mboehm7.github.io/teaching/ws2021_dia/index.htm
  - All course material (lecture slides) and dates
- **Video Recording Lectures (TUbe)?**
- **Communication**
  - **Informal language** (first name is fine)
  - Please, **immediate feedback** (unclear content, missing background)
  - Newsgroup: N/A – email is fine, summarized in following lectures
  - **Office hours:** by appointment or after lecture
- **Exam**
  - **Completed exercises or project** (checked by me/staff)
  - **Final written exam** (oral exam if <15 students take the exam)
  - **Grading** (40% project/exercises completion, 60% exam)

Course Logistics, cont.

- **Course Applicability**
  - **Bachelor** programs computer science (CS), as well as software engineering and management (SEM)
  - **Master** programs computer science (CS), as well as software engineering and management (SEM)
    - Catalog Data Science: compulsory course in major/minor
  - **Free subject course** in any other study program or university

Course Motivation and Goals

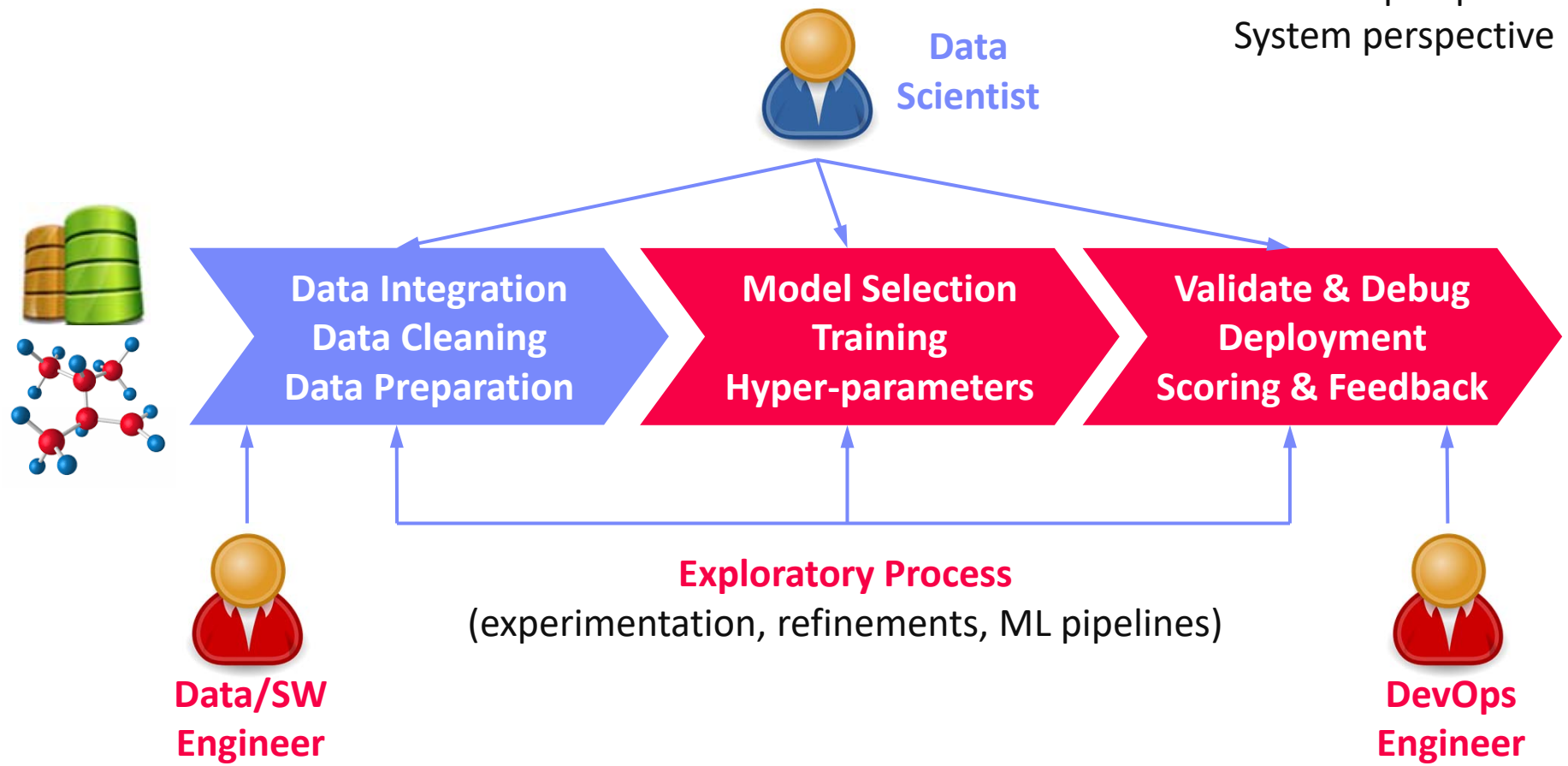
Data Sources and Heterogeneity

- **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

- **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)

The Data Science Lifecycle

Data-centric View:
Application perspective
Workload perspective
System perspective

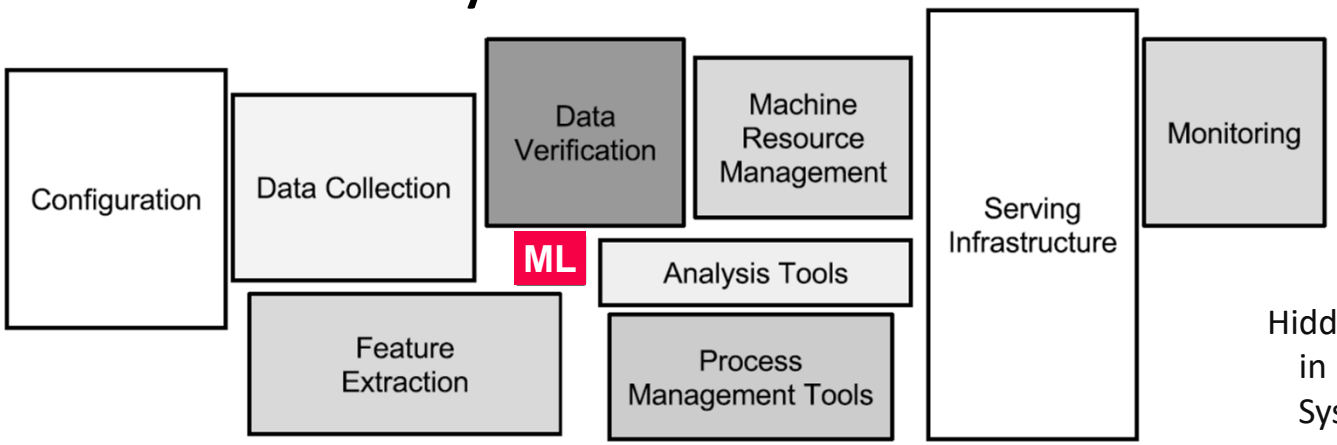


The 80% Argument

- **Data Sourcing Effort**
  - Data scientists spend **80-90% time** on finding relevant datasets and data integration/cleaning.

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

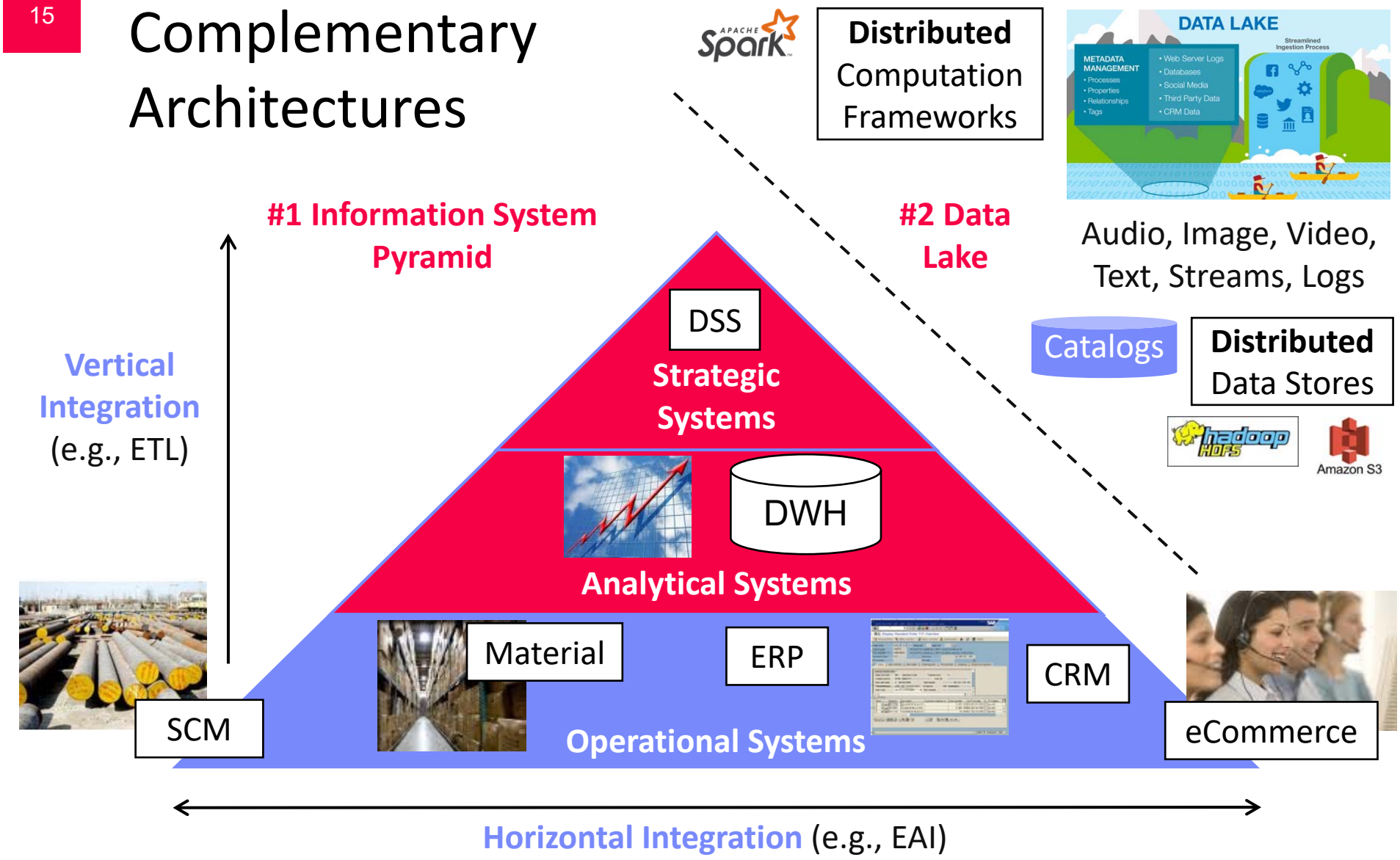
- **Technical Debts in ML Systems**



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

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

Complementary 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

Part A: Data Integration and Preparation

Data Integration Architectures

- **01 Introduction and Overview** [Oct 09]
- **02 Data Warehousing, ETL, and SQL/OLAP** [Oct 16]
- **03 Message-oriented Middleware, EAI, and Replication** [Oct 23]

Key Integration Techniques

- **04 Schema Matching and Mapping** [Oct 30]
- **05 Entity Linking and Deduplication** [Nov 06]
- **06 Data Cleaning and Data Fusion** [Nov 13]
- **07 Data Provenance and Blockchain** [Nov 20]

Part B: Large-Scale Data Management & Analysis

Cloud Computing

- **08 Cloud Computing Foundations** [Nov 27]
- **09 Cloud Resource Management and Scheduling** [Dec 04]
- **10 Distributed Data Storage** [Dec 11]

Large-Scale Data Analysis

- **11 Distributed, Data-Parallel Computation** [Jan 08]
- **12 Distributed Stream Processing** [Jan 15]
- **13 Distributed Machine Learning Systems** [Jan 22]
- **14 Q&A and exam preparation** [Jan 22]

Overview Projects or Exercises

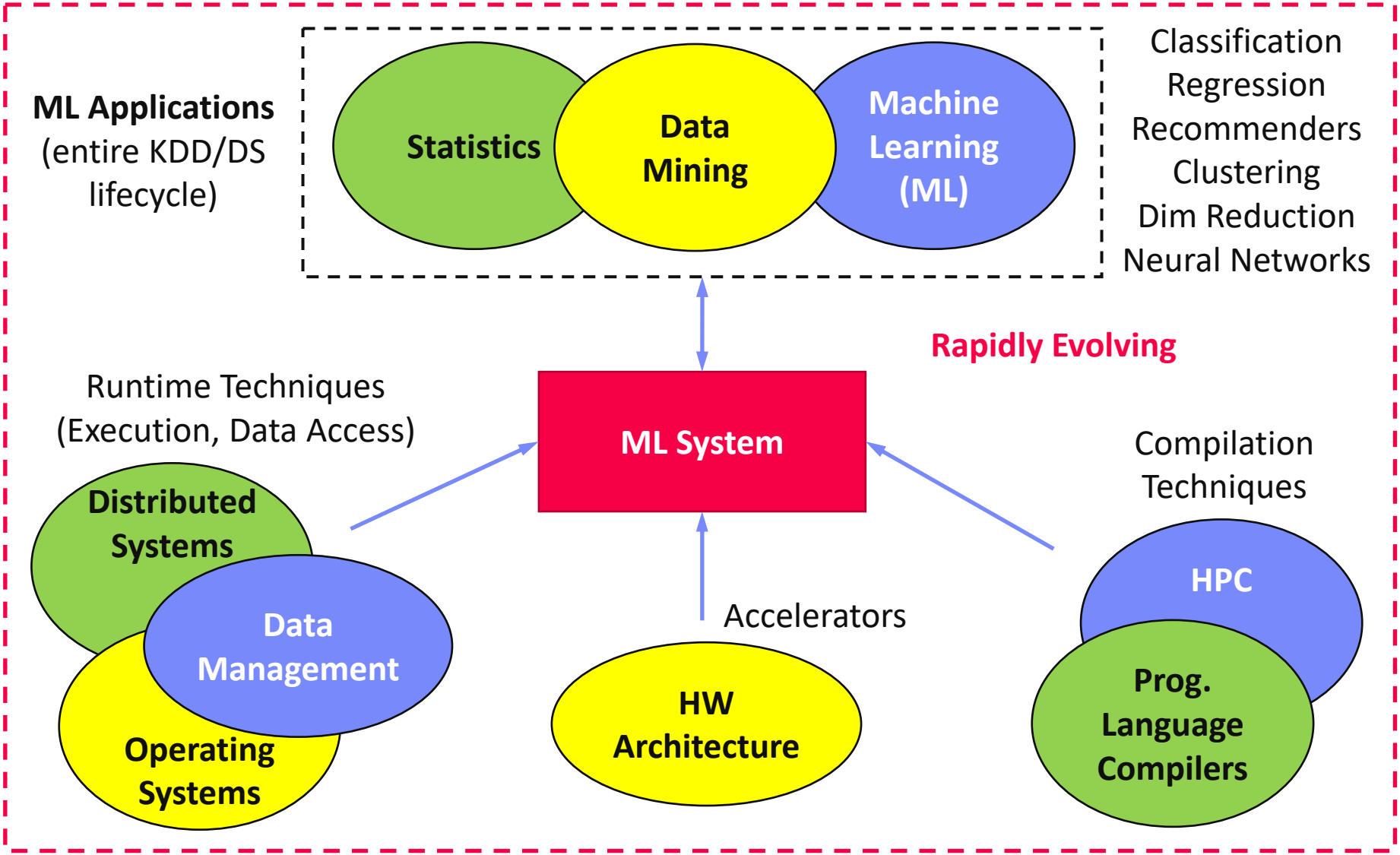
- **Team**
  - 1-3 person teams (w/ clearly separated responsibilities)
  - In exceptions also larger teams (e.g., Data Cleaning Benchmark)
- **Objectives**
  - Non-trivial programming project in DIA context (**2 ECTS → 50 hours**)
  - **Preferred:** Open source contribution to **Apache SystemDS**
    - https://github.com/apache/systemds
    - Topics throughout the stack (from HW to high-level scripting)
  - **Alternatively:** Exercise – Distributed entity resolution on Apache Spark
- **Timeline**
  - **Oct 16:** List of projects proposals, feel free to bring your own
  - **Oct 30:** Binding project/exercise selection
  - **Jan 15:** Final project/exercise deadline (**soft deadline**)

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

Background and System Architecture

https://github.com/apache/systemds

What is an ML System?

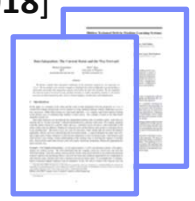


Landscape of ML Systems

- **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]



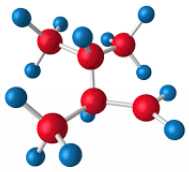
- **Data Preparation Problem**
  - **80% Argument:** 80-90% time for finding, integrating, cleaning data
  - Diversity of tools ➔ boundary crossing, lack of optimization

The Data Science Lifecycle

Data-centric View:
Application perspective
Workload perspective
System perspective

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

Data Scientist



Data Integration
Data Cleaning
Data Preparation

Model Selection
Training
Hyper-parameters

Validate & Debug
Deployment
Scoring & Feedback

Data/SW Engineer

Exploratory Process
(experimentation, refinements, ML pipelines)

DevOps Engineer

Key observation: SotA data integration/cleaning based on ML

Example: Linear Regression Conjugate Gradient

Note:
#1 Data Independence
#2 Implementation-Agnostic Operations

```
1:  X = read($1); # n x m matrix
2:  y = read($2); # n x 1 vector
3:  maxi = 50; lambda = 0.001;
4:  intercept = $3;
5:  ...
6:  r = -(t(X) %*% y);
7:  norm_r2 = sum(r * r); p = -r;
8:  w = matrix(0, ncol(X), 1); i = 0;
9:  while(i<maxi & norm_r2>norm_r2_trgt)
10: {
11:    q = (t(X) %*% (X %*% p))+lambda*p;
12:    alpha = norm_r2 / sum(p * q);
13:    w = w + alpha * p;
14:    old_norm_r2 = norm_r2;
15:    r = r + alpha * q;
16:    norm_r2 = sum(r * r);
17:    beta = norm_r2 / old_norm_r2;
18:    p = -r + beta * p; i = i + 1;
19: }
20: write(w, $4, format="text");
```

Read matrices from HDFS/S3

Compute initial gradient

Compute conjugate gradient

Compute step size

Update model and residuals

➔ **"Separation of Concerns"**

Background Apache SystemML

DML Scripts

APIs: Command line, JMLC, Spark MLContext, Spark ML, (20+ Scalable Algorithms)

Language

Compiler

Runtime

Write Once, Run Anywhere

In-Memory Single Node
(scale-up)

Hadoop or Spark Cluster
(scale-out)

Apache SystemML™

05/2017 Apache Top-Level Project
11/2015 Apache Incubator Project
08/2015 Open Source Release

[SIGMOD'15,'17,'19]
[PVLDB'14,'16a,'16b,'18]
[ICDE'11,'12,'15]
[CIDR'17]
[VLDBJ'18]
[DEBull'14]
[PPoPP'15]

In-Progress:

GPU

since 2014/16

Java

since 2012

hadoop

since 2010/11

APACHE Spark

since 2015

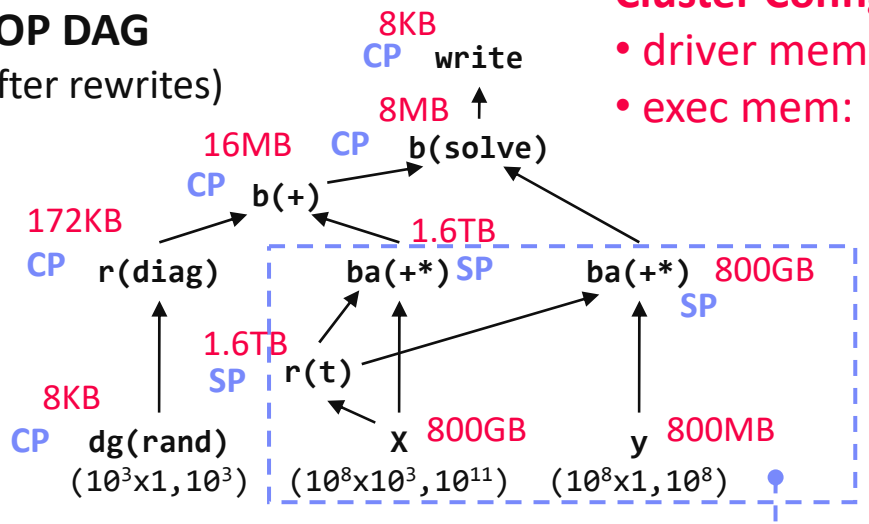
Basic HOP and LOP DAG Compilation

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);
```

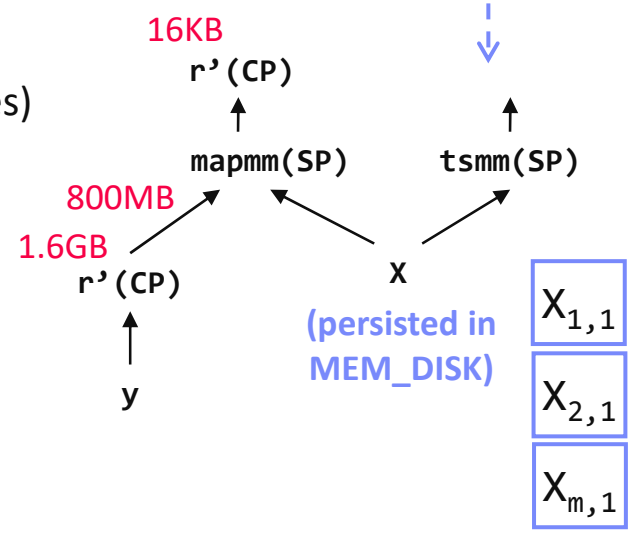
Scenario:
X: 10^8 x 10^3, 10^{11}
y: 10^8 x 1, 10^8

HOP DAG
(after rewrites)



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

LOP DAG
(after rewrites)



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

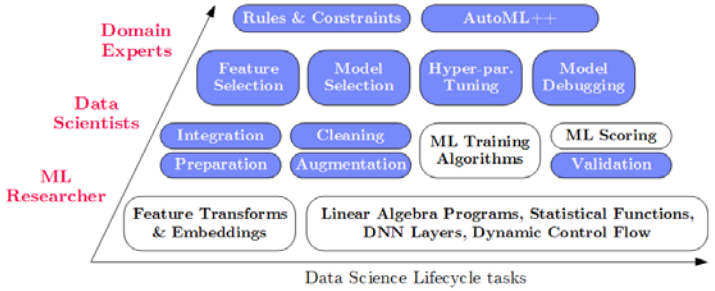
➔ **Distributed Matrices**
- Fixed-size (squared) matrix blocks
- Data-parallel operations

Apache SystemDS Design

Apache SystemML (since 2010)
→ SystemDS (09/2018)
→ Apache SystemDS (04/2020)

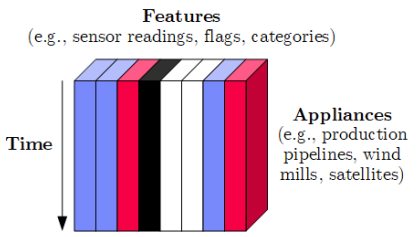
- **Objectives**
  - Effective and efficient **data preparation, ML, and model debugging at scale**
  - High-level abstractions for different lifecycle tasks and users

- **#1 Based on DSL for ML Training/Scoring**
  - Hierarchy of **abstractions for DS tasks**
  - ML-based SotA, interleaved, performance



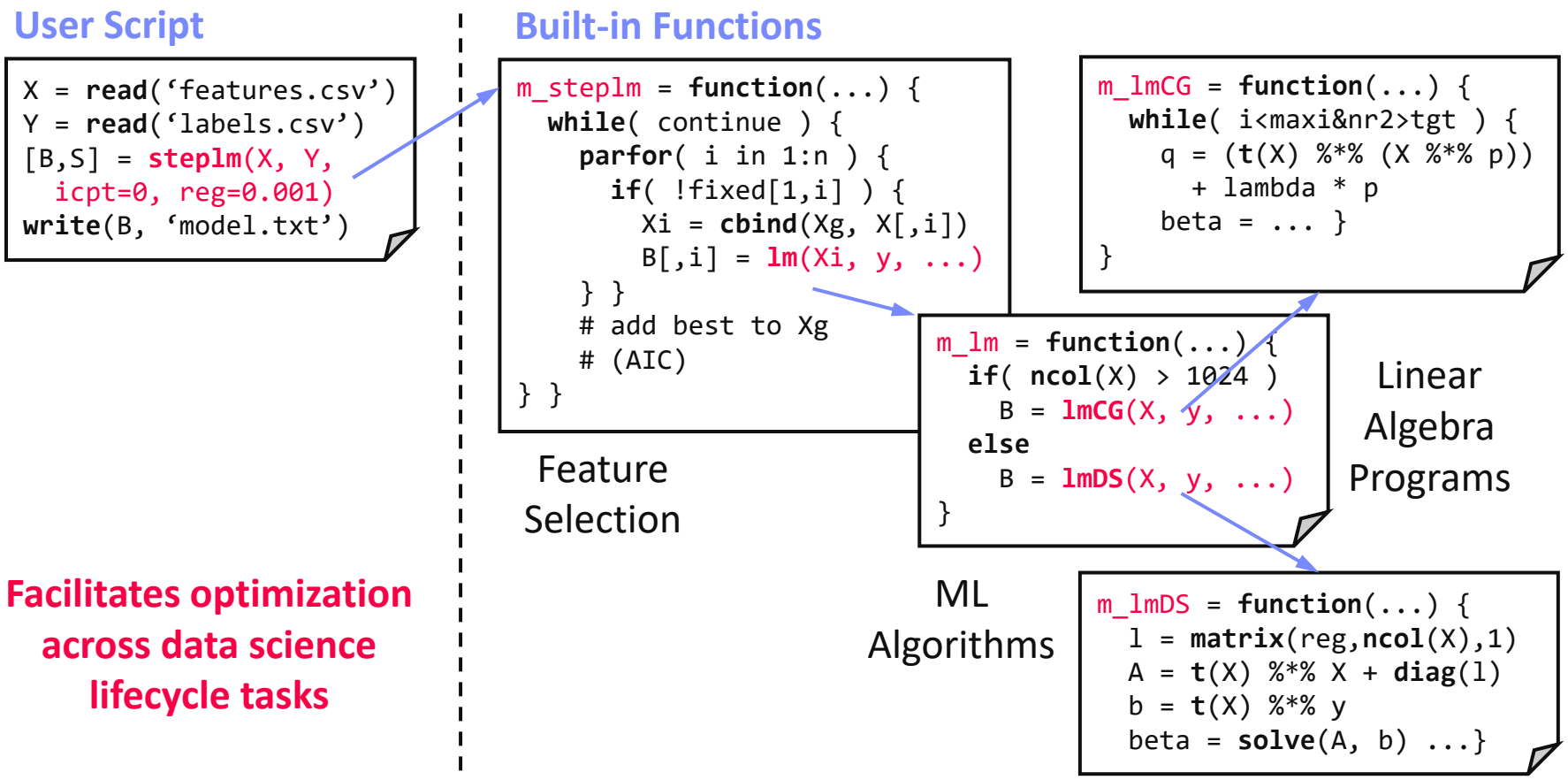
- **#2 Hybrid Runtime Plans and Optimizing Compiler**
  - System infrastructure for diversity of algorithm classes
  - Different parallelization strategies and new architectures (**Federated ML**)
  - Abstractions → redundancy → automatic optimization

- **#3 Data Model: Heterogeneous Tensors**
  - Data integration/prep requires **generic data model**



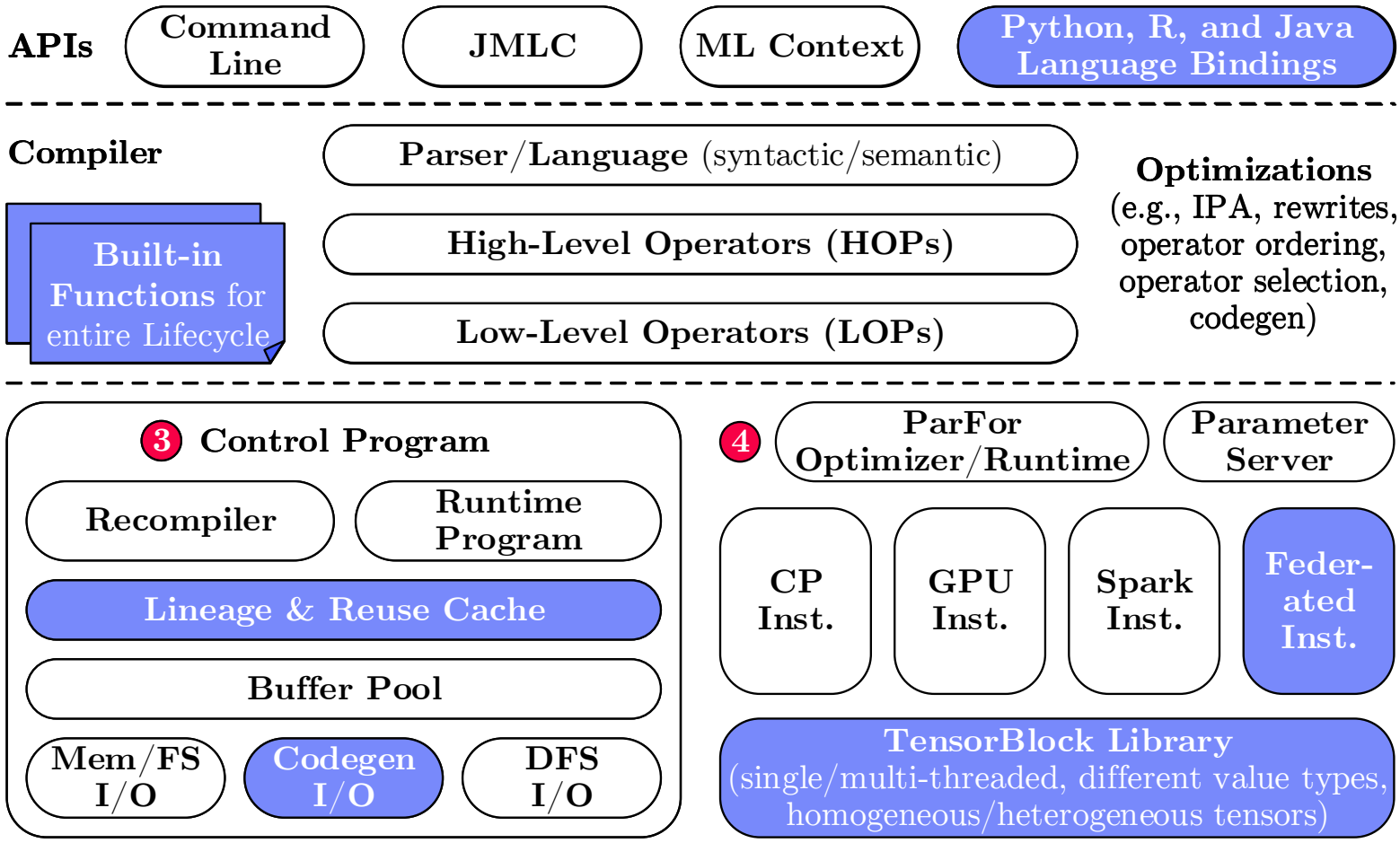
Language Abstractions and APIs, cont.

- **Example: Stepwise Linear Regression**



Apache SystemDS Architecture

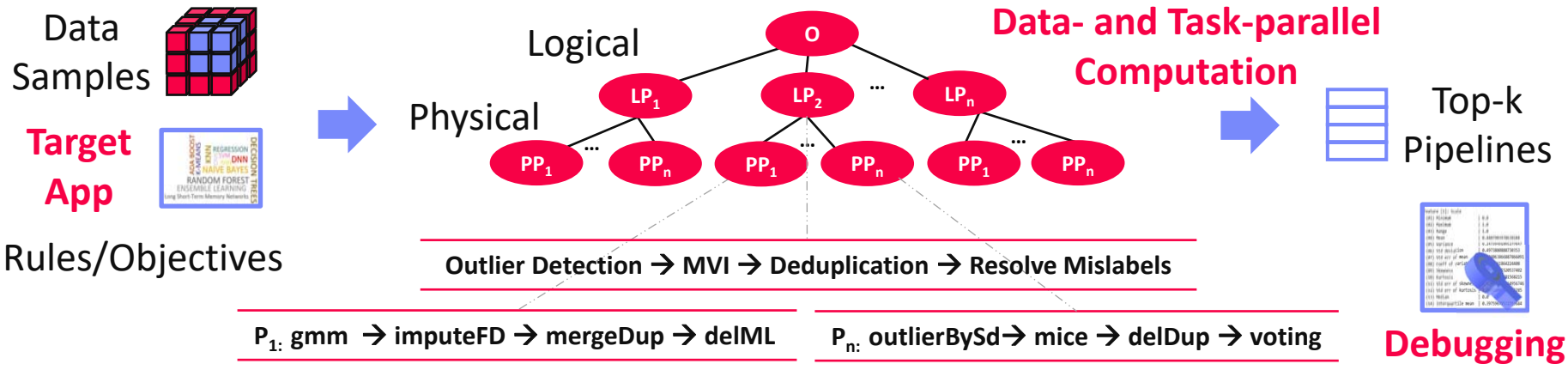
> 15,500 tests



[M. Boehm, I. Antonov, S. Baunsgaard, M. Dokter, R. Ginthör, K. Innerebner, F. Klezin, S. N. Lindstaedt, A. Phani, B. Rath, B. Reinwald, S. Siddiqui, S. Benjamin Wrede: SystemDS: A Declarative Machine Learning System for the End-to-End Data Science Lifecycle. **CIDR 2020**]

Data Cleaning Pipelines

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



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**

Multi-Level Lineage Tracing & Reuse

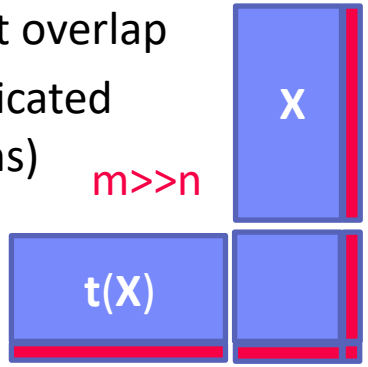

- **Lineage as Key Enabling Technique**
  - Trace lineage of operations (incl. non-determinism), dedup for loops/functions
  - Model versioning, data reuse, incremental maintenance, autodiff, debugging

- **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,], ...)
```

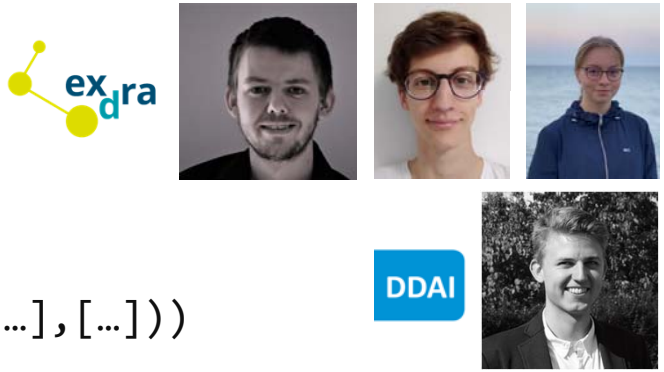
```
m_lmDS = function(...) {
  l = matrix(reg,ncol(X),1)
  A = t(X) %*% X + diag(l)
  b = t(X) %*% y
  beta = solve(A, b) ...}
```

- **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)
} }
```

Federated Learning



- **Python API**
  - Federated data objects and lazy evaluation

```
features = federated(sds, [node1,node2], ([…],[…]))
model = features.l2svm(labels).compute()
```

- **Example Federated Execution**

```
while(continueOuter & iter<maxi) {
  Xd = X %*% s (federated MV)
  # ...
  while(continueInner) {
    out = 1-Y* (Xw+step_sz*Xd);
    sv = (out > 0);
    out = out * sv;
    g = wd + step_sz*dd
          - sum(out * Y * Xd);
    h = dd + sum(Xd * sv * Xd);
    step_sz = step_sz - g/h;
  }
  g_new = t(X) %*% (out * Y)
        - lambda * w
  # ...
} ...
```

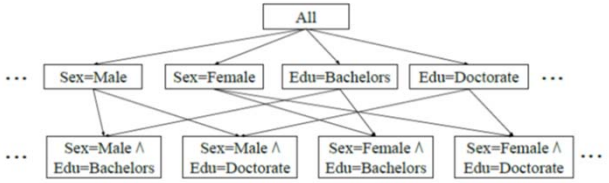

```
# At all workers
0. load Xi if not loaded
1. Send s → tmp1
2. Exec Xi %*% tmp1 → tmp2
3. Retrieve tmp2 as Xdi

# At master
Xd = rbind(Xd1, Xd2)
```

Model Debugging

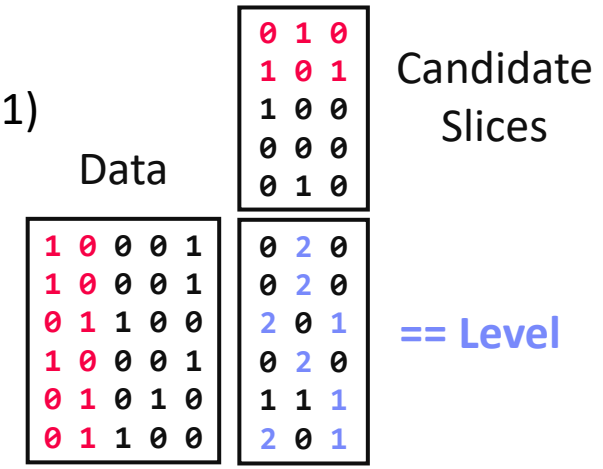
- **Problem:** Model M with 85% accuracy
  - **Find top-k data slices where model performs worse** than average
  - Data slice: S^{DG} := D=PhD ∧ G=female (subsets of features)

[Yeounoh Chung et al.: Slice Finder: Automated Data Slicing for Model Validation. CoRR 2018/**ICDE2019**]

- **Existing Algorithms**
  - Binning + One-Hot Encoding of **X**
  - Lattice search w/ heuristic, level-wise termination



- **Extensions**
  - Score: **α***(n/|S^{DG}|*e(S^{DG})/e(X)-1) - (**1-α**)*(n/|S^{DG}|-1)
  - **#1 Lower/upper bounds sizes/errors** → pruning & termination
  - **#2 Scalable implementation in linear algebra** (enum & eval via sparse-sparse matrix multiply)



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 distributed entity resolution pipeline on Apache Spark

- **Next Lectures**
  - 02 **Data Warehousing, ETL, and SQL/OLAP** [Oct 16]
  - 03 **Message-oriented Middleware, EAI, and Replication** [Oct 23]