Architecture of ML Systems
01 Introduction and Overview

Matthias Boehm

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

Last update: Mar 06, 2020

Announcements/Org

- **#1 Video Recording**
  - Link in TeachCenter & TUbe (lectures will be public)

- **#2 Course Registrations** (as of Mar 06)
  - Architecture of Machine Learning Systems (AMLS): **34**
  - Bachelor/master/PhD ratio?

- **#3 CS Talks x7** (**Mar 10, 5pm**, Aula Alte Technik)
  - Claudia Müller-Birn (Freie Universität of Berlin)
  - Title: Collaboration is Key – Human-Centered Design of Computational Systems

Agenda

- **Data Management Group**
- **Motivation and Goals**
- **Course Organization**
- **Course Outline, and Projects**
- **Overview SystemDS**

Data Management Group

About Me

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

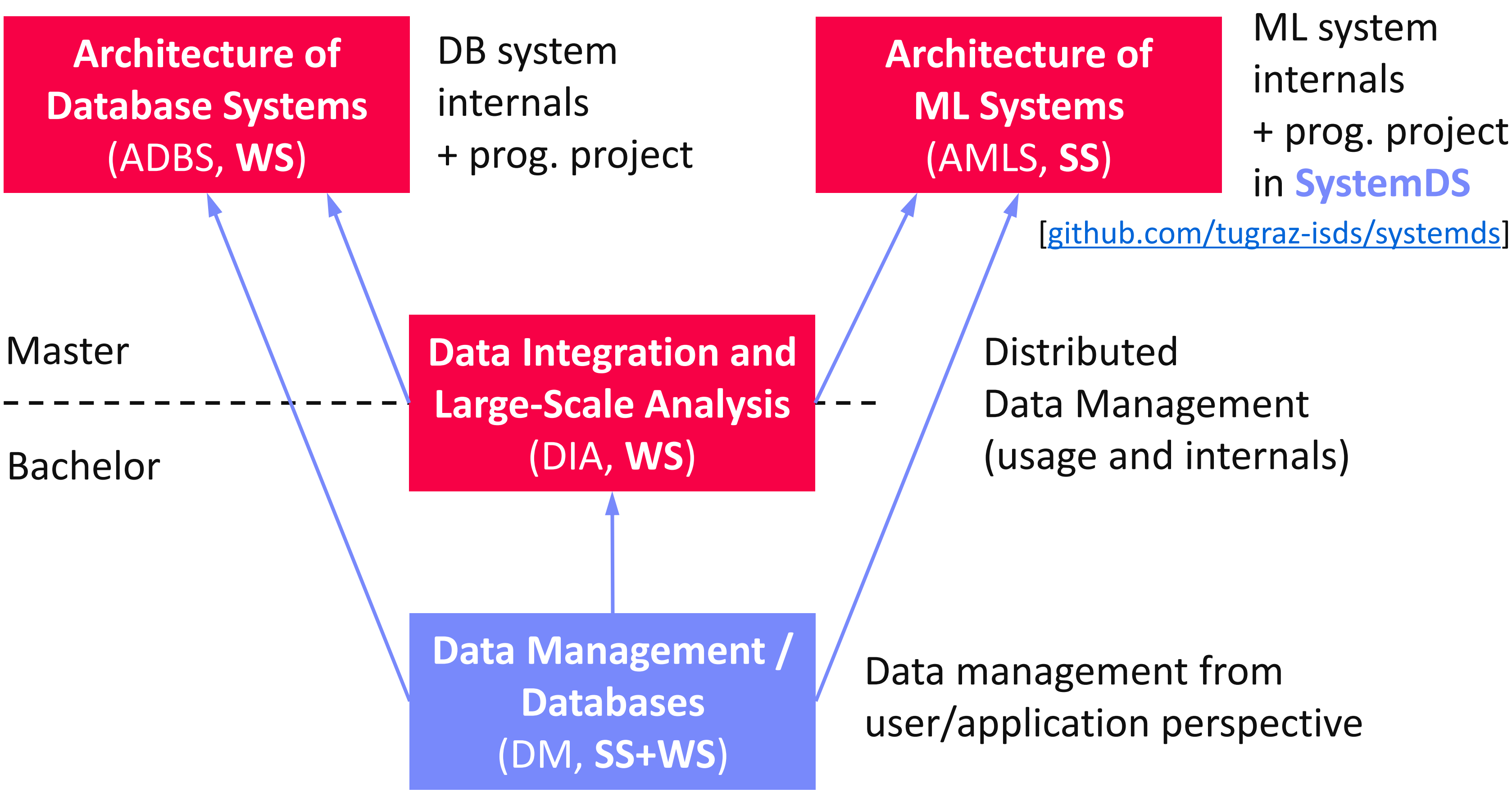
https://github.com/tugraz-isds/systemds

- **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

DB group

Data Management Courses

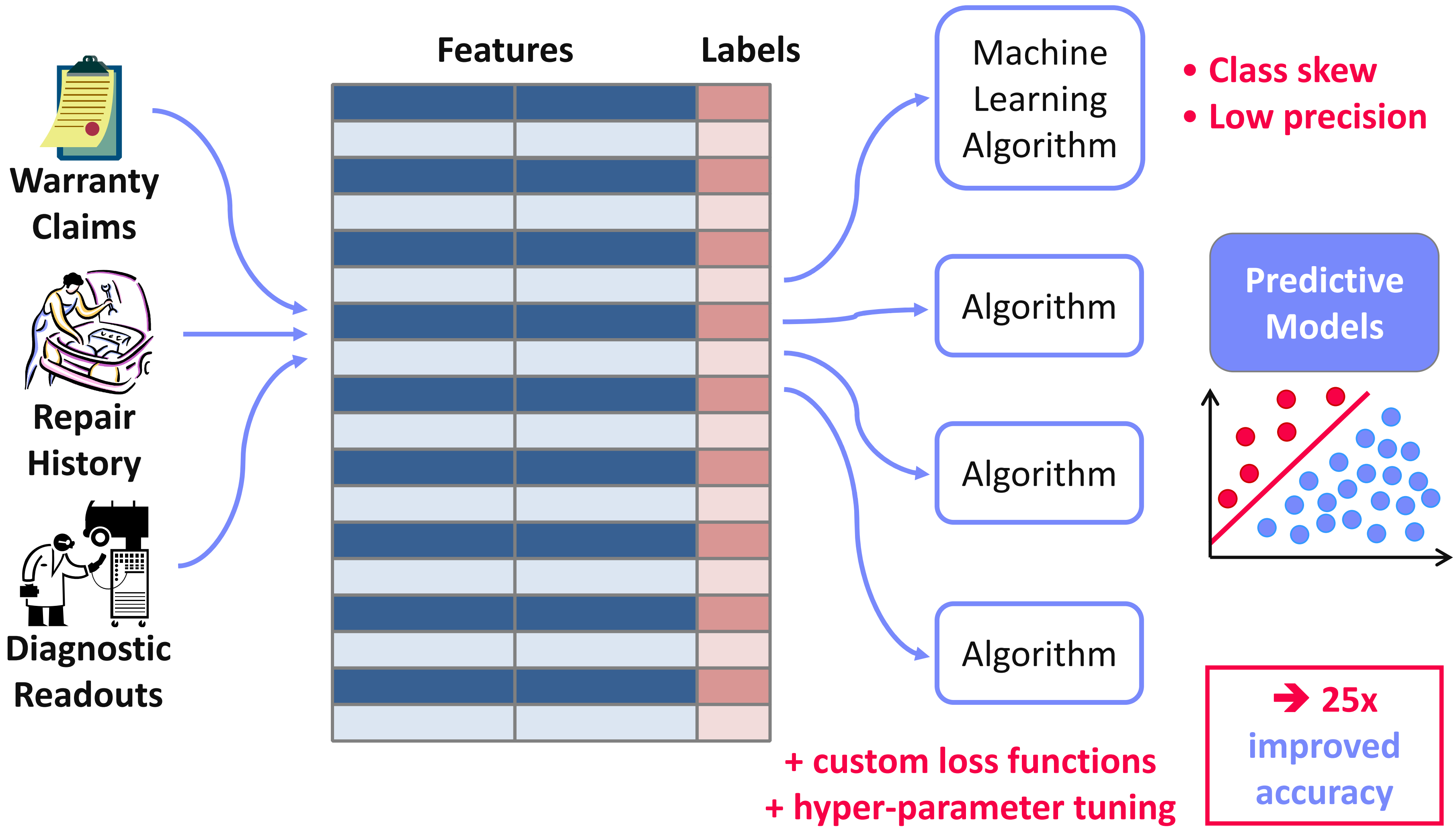


Motivation and Goals

Example ML Applications (Past)

- **Transportation / Space**
  - Lemon car detection and reacquisition (classification, seq. mining)
  - Airport passenger flows from WiFi data (time series forecasting)
  - Satellite senor analytics (regression and correlation)
  - Data analysis for automated driving (various use cases)
- **Finance**
  - Water cost index based on various influencing factors (regression)
  - Insurance claim cost per customer (model selection, regression)
  - Financial analysts survey correlation (bivariate stats w/ new tests)
- **Health Care**
  - Breast cancer cell grow from histopathology images (classification)
  - Glucose trends and warnings (clustering, classification)
  - Emergency room diagnosis / patient similarity (classification, clustering)
  - Patient survival analysis and prediction (Cox regression, Kaplan-Meier)

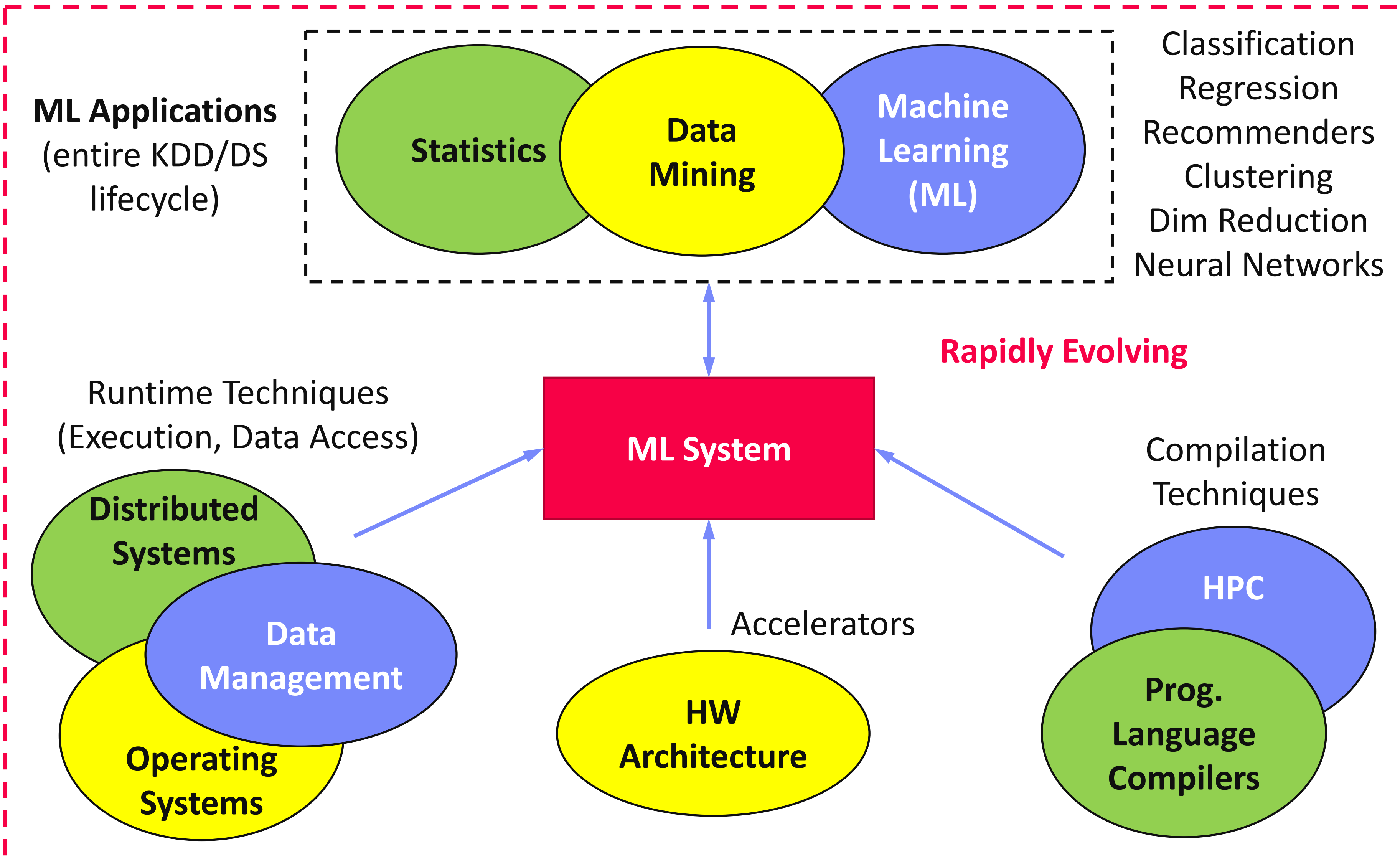
A Car Reacquisition Scenario



Example ML Applications (Past), cont.

- **Other Domains**
  - **Machine data: errors and correlation** (bivariate stats, seq. mining)
  - Smart grid: energy demand/RES supply, weather models (forecasting)
  - Visualization: dimensionality reduction into 2D (auto encoder)
  - **Elastic flattening via sparse linear algebra** (spring-mass system)
- **Information Extraction**
  - **NLP contracts → rights/obligations** (classification, error analysis)
  - **PDF table recognition and extraction** (NMF clustering, custom)
  - OCR: optical character recognition (preprocessing, classification)
- **Algorithm Research** (+ various state-of-the art algorithms)
  - **User/product recommendations** via various forms of NMF
  - Localized, supervised metric learning (dim reduction and classification)
  - Learning word embeddings via orthogonalized skip-gram
  - Learning first-order rules for explainable classification

What is an ML System?



What is an ML System?, cont.

- **ML System**
  - **Narrow focus:** SW system that executes ML applications
  - **Broad focus:** Entire system (HW, compiler/runtime, ML application)
  - ➔Trade-off **runtime/resources** vs **accuracy**
  - ➔Early days: no standardizations (except some exchange formats), lots of different languages and system architectures, but many shared concepts

- **Course Objectives**
  - Architecture and internals of modern (large-scale) ML systems
    - **Microscopic view** of ML system internals
    - **Macroscopic view** of ML pipelines and data science lifecycle
  - **#1** Understanding of characteristics → **better evaluation / usage**
  - **#2** Understanding of effective techniques → **build/extend ML systems**

Course Organization

Basic Course Organization

- **Staff**
  - Lecturer: Univ.-Prof. Dr.-Ing. Matthias Boehm, ISDS
  - Assistant: M.Sc. Sebastian Baunsgaard, 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** (**start 12.15pm**, including **Q&A**), **attendance optional**
  - **Mandatory programming project** (2 ECTS)
  - **Recommended papers** for additional reading on your own

Course Logistics

- **Exam**
  - **Completed project** (merged PRs)
  - **Final oral exam** (via doodle slot pocking)
  - **Grading** (40% project, 60% exam)

- **Communication**
  - **Informal language** (first name is fine)
  - Please, **immediate feedback** (unclear content, missing background)
  - Newsgroup: news://news.tugraz.at/tu-graz.lv.amls (email for private issues)
  - Office hours: by appointment or after lecture

- **Website**
  - https://mboehm7.github.io/teaching/ss20_amls/index.htm
  - All course material (lecture slides, list of projects) and dates

Course Logistics, cont.

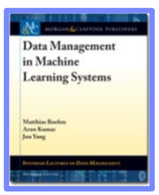
- **Open Source Projects**
  - Programming project in context of open source projects
    - SystemDS: https://github.com/tugraz-isds/systemds
    - Other open source projects possible, **but harder to merge PRs**
  - Commitment to **open source and open communication** (discussion on PRs, mailing list, etc)
  - **Remark:** Don't be afraid to ask questions / develop code in public

- **Objectives**
  - Non-trivial feature in an open source ML system (**2 ECTS → 50 hours**)
  - **OSS processes:** Break down into subtasks, code/tests/docs, PR per project, code review, incorporate review comments, etc

- **Team**
  - Individuals or two-person teams (w/ clearly separated responsibilities)

Course Outline and Projects

Partially based on



[Matthias Boehm, Arun Kumar, Jun Yang: Data Management in Machine Learning Systems. Synthesis Lectures on Data Management, Morgan & Claypool Publishers 2019]

Major updates coming (compared to SS19)

Part A: Overview and ML System Internals

- **01 Introduction and Overview** [Mar 06]
- **02 Languages, Architectures, and System Landscape** [Mar 13]

- **03 Size Inference, Rewrites, and Operator Selection** [Mar 20]
- **04 Operator Fusion and Runtime Adaptation** [Mar 27]
- **05 Data- and Task-Parallel Execution** [Apr 03]
- **06 Parameter Servers** [Apr 24]

- **07 Hybrid Execution and HW Accelerators** [May 08]
- **08 Caching, Partitioning, Indexing, and Compression** [May 15]

Part B: ML Lifecycle Systems

- **09 Data Acquisition, Cleaning, and Preparation** [May 29]
- **10 Model Selection and Management** [Jun 05]
- **11 Model Debugging Techniques** [Jun 12]
- **12 Model Serving Systems and Techniques** [Jun 19]

- **13 Trends and Research Directions 2020** [Jun 26]
- **14 Q&A and Exam Preparation**

Preliminary Example Projects

- **#1 Extended Python and Java Language Bindings**
- **#2 Auto Differentiation** (builtin function and compiler)
- **#3 Built-in Functions for Regression, Classification, Clustering**
- **#4 Built-in Functions for Time Series Missing Value Imputation**
- **#5 DL-based Entity Resolution Primitives** (baseline implementation)
- **#6 Model Selection Primitives** (BO, multi-armed bandit, hyperband)

- **#7 Documentation and Tutorials** (for different target users)
- **#8 Extended Test Framework** (comparisons, caching, remove redundancy)
- **#9 Performance Testsuite** (extend algorithm-level suite)
- **#10 ONNX Graph Importer** (DML script / HOP DAG generation)

Preliminary Example Projects, cont.

- **#11 Loop Vectorization Rewrites** (more general framework)
- **#12 Canonicalization Rewrite Framework** (refactoring, new rewrites)
- **#13 Extended CSE & Constant Folding** (commutativity, one-shot)
- **#14 Extended Matrix Multiplication Chain Opt** (sparsity, rewrites)
- **#15 Extended Update In-Place Framework** (reference counting)

- **#16 SLIDE Operators and Runtime Integration** (Sub-LInear DL Engine)
- **#17 Compression Planning Extensions** (co-coding search algorithm)
- **#18 Feature Transform: Equi-Height/Custom Binning** (local, distributed)
- **#19 Extended Intel MKL-DNN Runtime Operations** (beyond conv2d)
- **#20 Extended I/O Framework for Other Formats** (e.g., NetCDF, HDF5, Arrow)
- **#21 Protobuf reader/writer into Data Tensor** (local, distributed)

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

Matthias Boehm[1,2], Iulian Antonov[2], Sebastian Baunsgaard[1], Mark Dokter[2], Robert Ginthör[2], Kevin Innerebner[1], Florijan Klezin[2], Stefanie Lindstaedt[1,2], Arnab Phani[1], Benjamin Rath[1], Berthold Reinwald[3], Shafaq Siddiqi[1], Sebastian Benjamin Wrede[2]

[1] **Graz University of Technology**; Graz, Austria
[2] **Know-Center GmbH**; Graz, Austria
[3] **IBM Research – Almaden**; San Jose, CA, USA

SCIENCE
PASSION
TECHNOLOGY

Motivation SystemDS

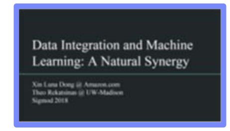
- **Existing ML Systems**
  - **#1 Numerical computing** frameworks
  - **#2 ML Algorithm libraries** (local, large-scale)
  - **#3 Linear algebra ML systems** (large-scale)
  - **#4 Deep neural network** (DNN) frameworks
  - **#5 Model management, and deployment**
- **Exploratory Data-Science Lifecycle**
  - **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
  - **In-DBMS ML** toolkits **largely unsuccessful** (stateful, data loading, verbose)

Motivation SystemDS, cont.

[Xin Luna Dong, Theodoros Rekatsinas: Data Integration and Machine Learning: A Natural Synergy. **SIGMOD 2018**]



- **Key Observation**
  - **SotA data integration based on ML**
    (e.g., data extraction, schema alignment, entity linking)
  - **Similar:** data cleaning, outlier detection, missing value imputation, semantic type detection, data augmentation, feature selection, hyper parameter optimization, model debugging

- **A Case for Declarative Data Science**
  - High-level abstractions (**R/Python**, **stateless**) for lifecycle tasks, implemented **in DSL for ML training/scoring**
  - **Avoid boundary crossing** and **optimizations across lifecycle**
  - Control compiler and runtime of utmost importance

Apache SystemML → SystemDS

Architecture and Preliminary Results

SystemML Background

Apache SystemML™

Example: Linear Regression Conjugate Gradient

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

```
X = read($1); # n x m matrix
y = read($2); # n x 1 vector
maxi = 50; lambda = 0.001;
intercept = $3;
...
r = -(t(X) %*% y);
norm_r2 = sum(r * r); p = -r;
w = matrix(0, ncol(X), 1); i = 0;
while(i<maxi & norm_r2>norm_r2_trgt)
{
   q = (t(X) %*% (X %*% p))+lambda*p;
   alpha = norm_r2 / sum(p * q);
   w = w + alpha * p;
   old_norm_r2 = norm_r2;
   r = r + alpha * q;
   norm_r2 = sum(r * r);
   beta = norm_r2 / old_norm_r2;
   p = -r + beta * p; i = i + 1;
}
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"

High-Level SystemML Architecture

APIs: Command line, JMLC, Spark MLContext, Spark ML, (20+ scalable algorithms)

DML Scripts

Language

Compiler

Runtime

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-Memory Single Node
(scale-up)

Hadoop or Spark Cluster
(scale-out)

In-Progress:

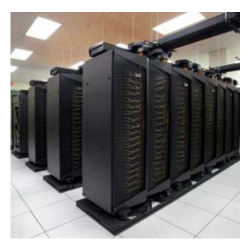
GPU

since 2014/16

Java

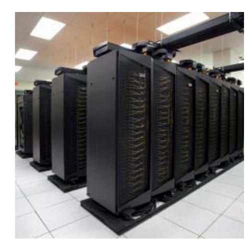
since 2012

hadoop



since 2010/11

APACHE Spark



since 2015

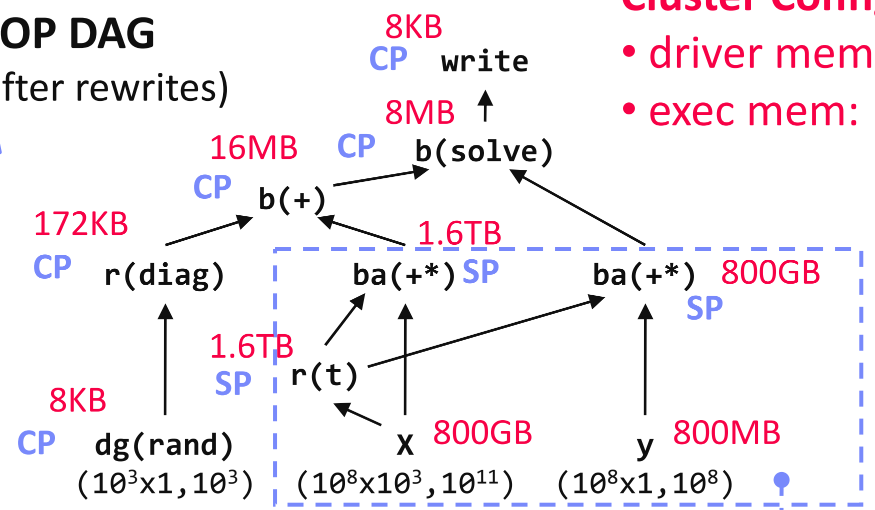
Basic HOP and LOP DAG Compilation

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);
```

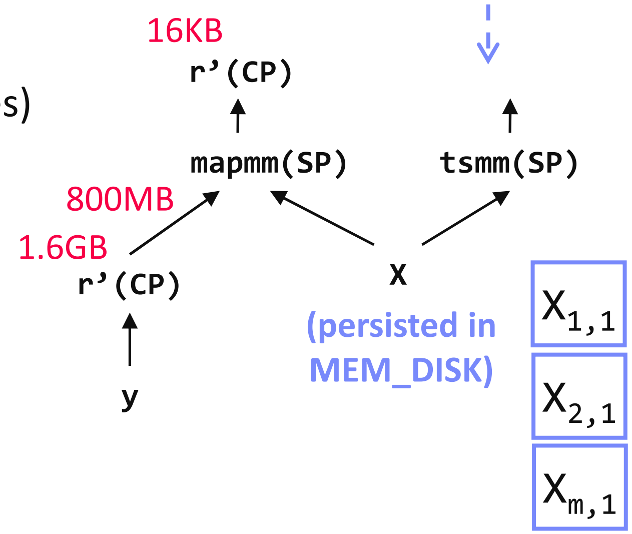
Scenario:
X: 10^8 x 10^3, 10^{11}
y: 10^8 x 1, 10^8

HOP DAG
(after rewrites)



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

LOP DAG
(after rewrites)



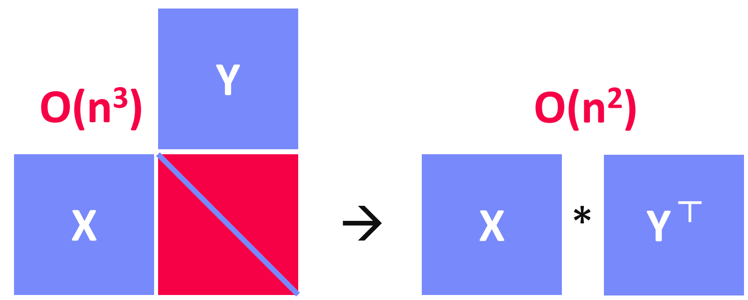
➔ **Hybrid Runtime Plans:**
- **Size propagation / memory estimates**
- **Integrated CP / Spark runtime**
- **Dynamic recompilation during runtime**

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

Static and Dynamic Rewrites

- **Example Static Rewrites** (size-indep.)
  - Common Subexpression Elimination
  - Constant Folding / Branch Removal / Block Sequence Merge
  - Static Simplification Rewrites
  - Right/Left Indexing Vectorization
  - For Loop Vectorization
  - Spark checkpoint/repartition injection

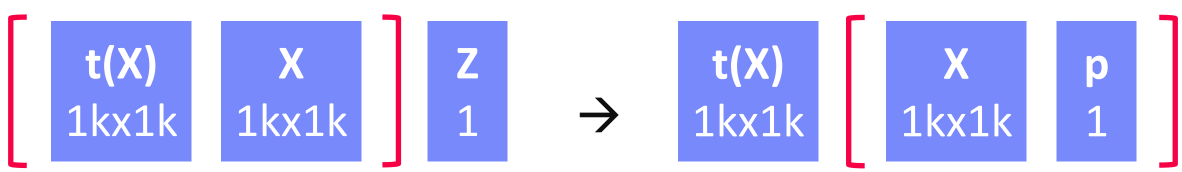
```
trace(X%*%Y) → sum(X*t(Y))
```




```
sum(λ*X) → λ*sum(X)
sum(X+Y) → sum(X)+sum(Y)
```

- **Example Dynamic Rewrites** (size-dep.)
  - Dynamic Simplification Rewrites
  - Matrix Mult Chain Optimization

```
rowSums(X) → X, iff ncol(X)=1
sum(X^2) → X%*%t(X), iff ncol(X)=1
```



2,002 MFLOPs

4 MFLOPs

Size propagation and sparsity estimation

Selected Research Results

What-If

#3 Resource Optimization
for automatic resource provisioning
(SIGMOD'15)

#4 Compressed Linear Algebra
(PVLDB'16, SIGMOD Record'17, VLDB Journal'18, CACM'19)

parfor

#2 Task-Parallel Parfor Loops
hybrid parallelization strategies
(PVLDB'14)

Apache SystemML™

#5 Optimizing Operator Fusion Plans
(PPoPP'15, CIDR'17, PVLDB'18)

#1 SystemML's Optimizer
rewrites, operator selection, size propagation, memory estimates, dynamic recompilation (DEBull'14)

ΣΠ **#6 Advanced Optimization**
sum-product (CIDR'17),
sparsity estimation (SIGMOD'19)

Google Summer of Code

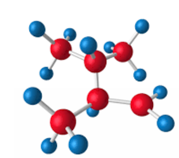
GPU, meta, numerical stability, **parameter servers**, etc

Lessons Learned from SystemML

Why was SystemML not adopted in practice?

- **L1 Data Independence & Logical Operations**
  - Independence of **evolving technology stack** (MR → Spark, GPUs)
  - **Simplifies development** (libs) and **deployment** (large-scale vs. embedded)
  - **Enables adaptation** to cluster/data characteristics (dense/spare/compressed)
- **L2 User Categories** (|Alg. Users| **>>** |Alg. Developers|)
  - **Focus on ML researchers** and algorithm developers **is a niche**
  - Data scientists and domain experts **need higher-level abstractions**
- **L3 Diversity of ML Algorithms & Apps**
  - **Variety of algorithms** (batch 1st/2nd, mini-batch DNNs, hybrid)
  - Different parallelization, ML + rules, numerical computing

- **L4 Heterogeneous Structured Data**
  - Support for **feature transformations on 2D frames**
  - Many apps deal with **heterogeneous data and various structure**



SystemDS Architecture

(An open source ML **System** for the end-to-end **D**ata **S**cience lifecycle)

https://github.com/tugraz-isds/systemds,

forked from Apache SystemML 1.2 in Sep 2018

SystemDS 0.1 published Aug 31, 2019

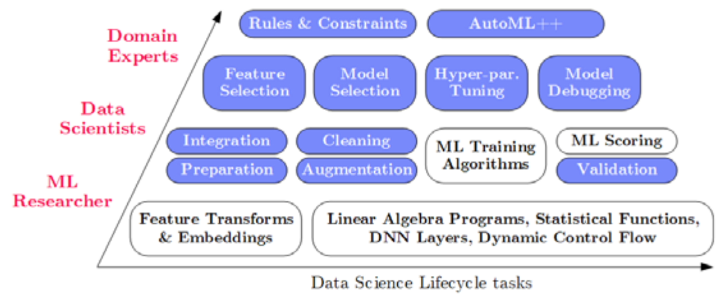
SystemDS 0.2 upcoming (next week)

Upcoming merge into Apache SystemML

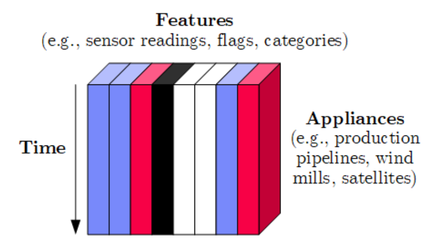
SystemDS Vision and Design

Game Plan

- **Objectives**
  - Effective and efficient **data preparation, ML, and model debugging at scale**
  - High-level abstractions for lifecycle tasks (L3/L4) and users (L2)
- **#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**

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

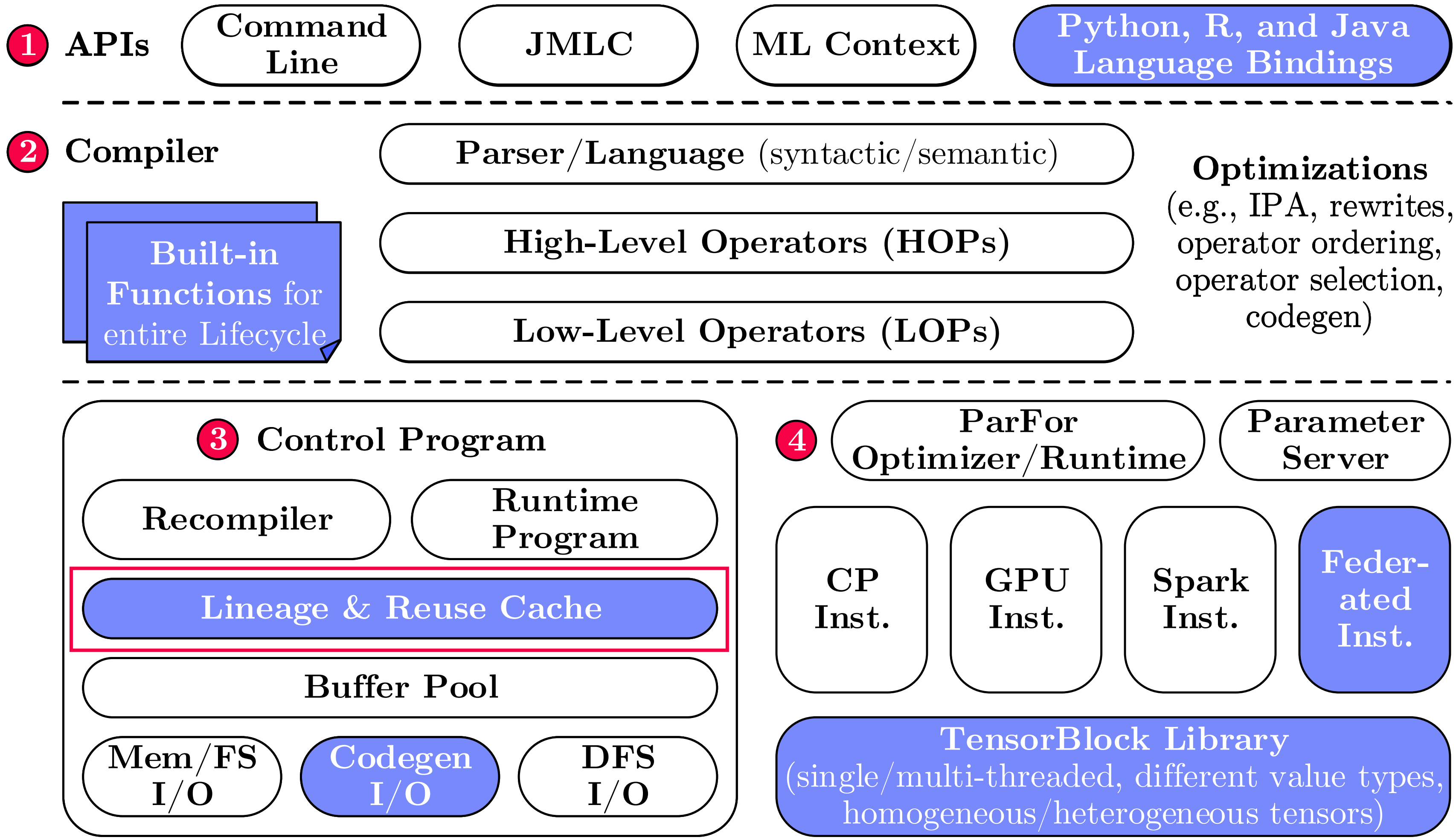
```
m_lm = function(...) {
  if( ncol(X) > 1024 )
    B = lmCG(X, y, ...)
  else
    B = lmDS(X, y, ...)
}
```

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) ...}
```

System Architecture



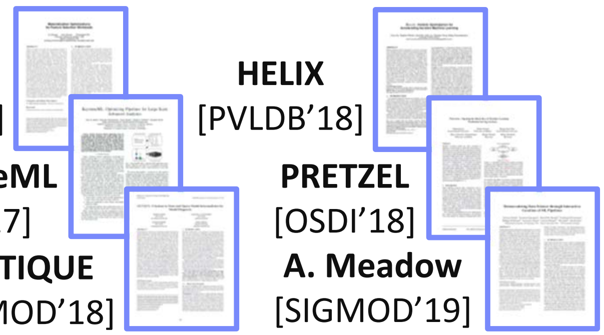
Lineage and Reuse

- **Problem**
  - **Exploratory data science** (data preprocessing, model configurations)
  - **Reproducibility** and **explainability** of trained models (data, parameters, prep)

➔ **Lineage as Key Enabling Technique**

- Model versioning, **data reuse**, incremental maintenance, auto diff, debugging (e.g., queries over lineage)

COLUMBUS [SIGMOD'14]
KeystoneML [ICDE'17]
MISTIQUE [SIGMOD'18]
HELIX [PVLDB'18]
PRETZEL [OSDI'18]
A. Meadow [SIGMOD'19]



- **a) Efficient Fine-Grained Lineage Tracing**
  - Tracing of inputs, literals, and **non-determinism**
  - **Trace lineage of logical operations** for all live variables, store along outputs, program/output reconstruction possible:

```
X = eval(deserialize(serialize(lineage(X))))
```

  - **Proactive deduplication** of lineage traces for loops, (and functions)

Lineage and Reuse, cont.

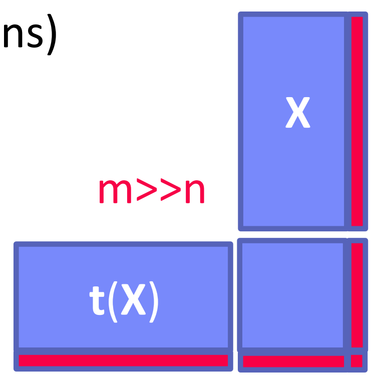
- **b) Full Reuse of Intermediates**
  - Before executing instruction, probe output lineage in cache `Map<Lineage, MatrixBlock>`
  - Cost-based/heuristic caching and eviction decisions (compiler-assisted)

$O(k(mn^2+n^3)) \rightarrow O(mn^2+kn^3)$

```
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) ...}
```

- **c) 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)
} }
```

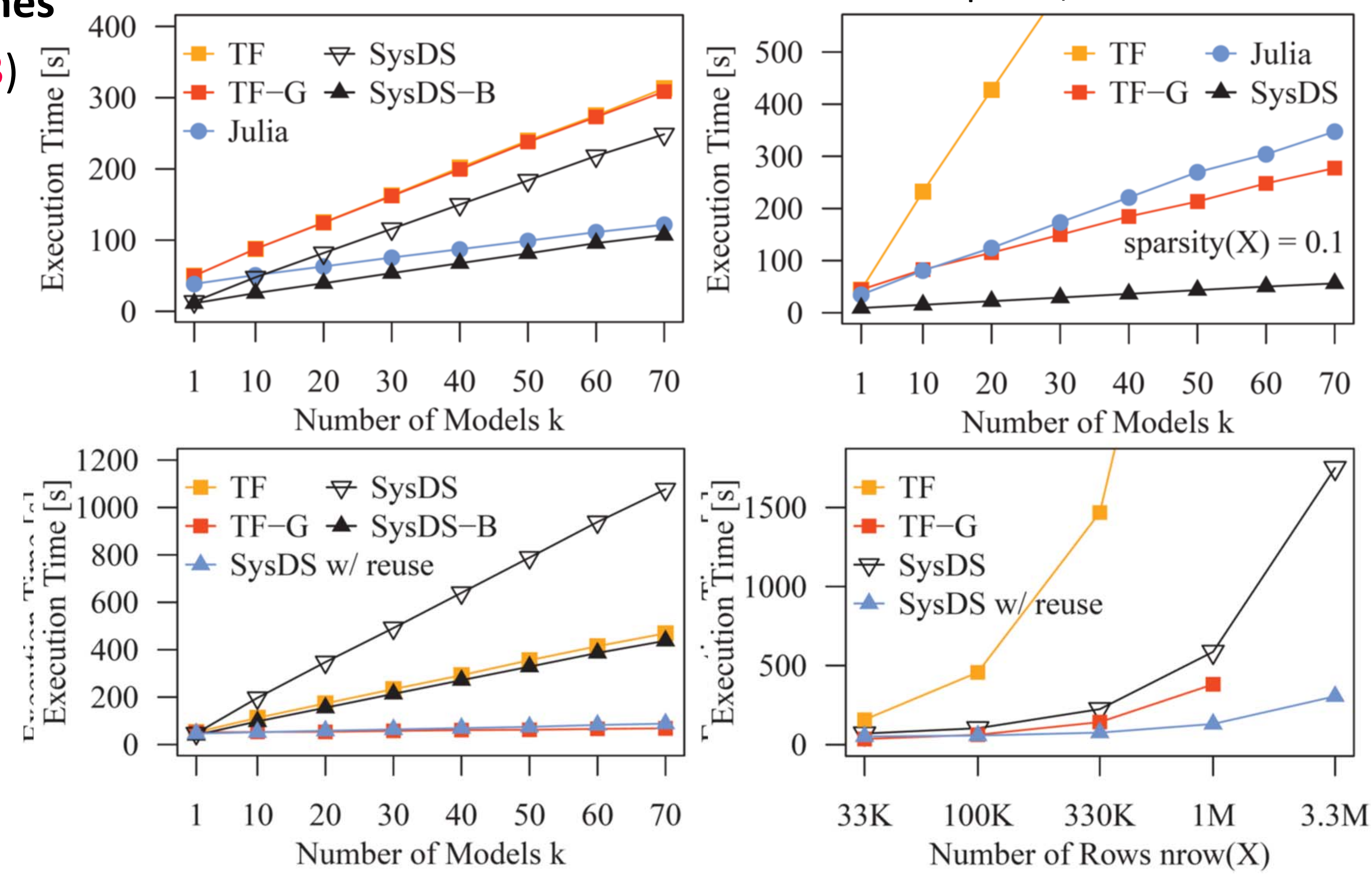
$O(n^2(mn^2+n^3)) \rightarrow O(n^2(mn+n^3))$

Experiments (Hyper-Param Opt)

- **Baselines** (**TF1.13**)
- **Full Reuse** (**TF2.0**)

Dense, 100K x 1K

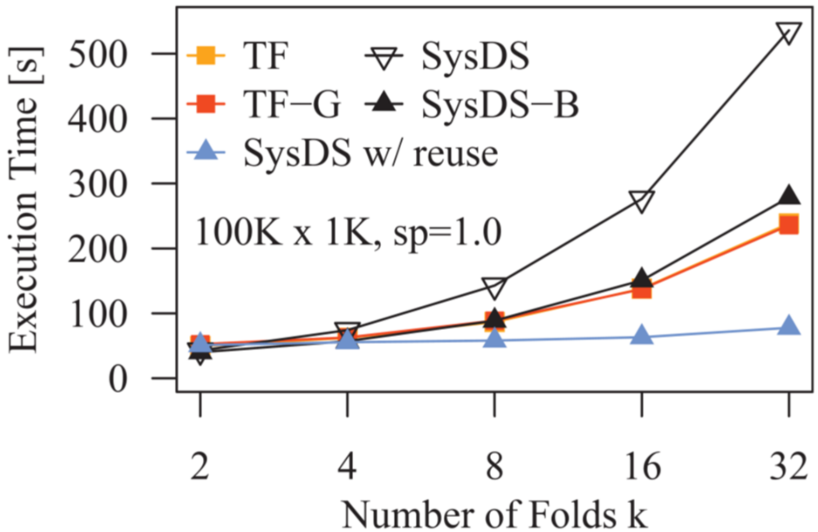
Sparse, 100K x 1K



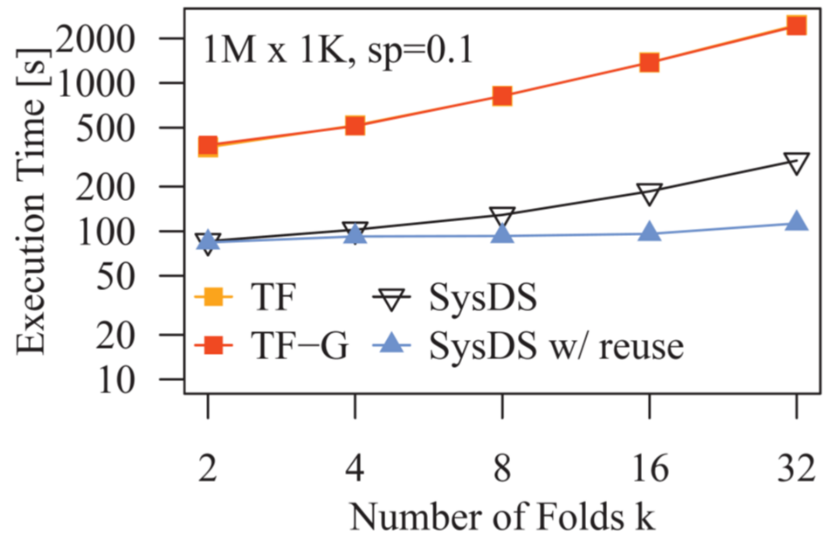
Experiments (Cross Validation)

- **Full Reuse** (**TF2.0**)

Dense, 100K x 1K



Sparse, 1M x 1K

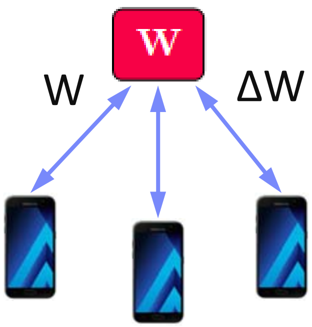


#1 **Competitive baseline performance** ML training (dense, sparse)

#2 Large improvements due to **fine-grained redundancy elimination**

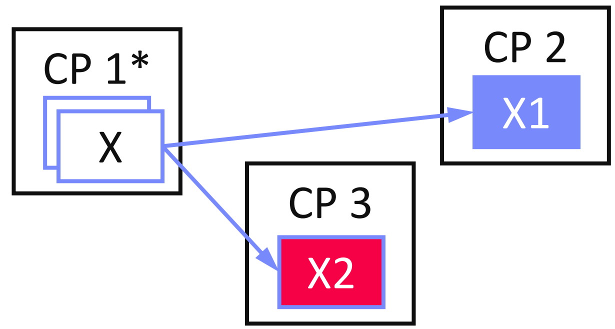
Federated ML

[Keith Bonawitz et al.: Towards Federated Learning at Scale: System Design. SysML 2019]



- **Motivation Federated ML**
  - Model training **w/o central data consolidation**
  - Data Ownership ➔ Federated ML in the enterprise (machine vendor – middle-person – customer equipment)

- **Federated ML Architecture**
  - Multiple control programs w/ single master
  - Federated tensors (metadata handles)
  - **Federated linear algebra** and **parameter server**
  - PET integration (MPC, homomorphic encryption)



- **ExDRa Project** (Exploratory Data Science over Raw Data)
  - **Basic approach:** Federated ML + ML over raw data
  - System infra, integration, data org & reuse, Exp DB, geo-dist.

SIEMENS

Bundesministerium Verkehr, Innovation und Technologie

FFG Forschung wirkt.

Gefördert im Programm "IKT der Zukunft" vom Bundesministerium für Verkehr, Innovation, und Technologie (BMVIT)

TU Graz

Technische Universität Berlin

Conclusions

- **Summary: SystemML is dead, long live SystemDS**
  - Vision and system architecture of SystemDS
  - Selected research directions and preliminary results

→ **Apache SystemDS** (Mar 2020)

- **#1 Support for data science lifecycle tasks** (data prep, training, debugging), **users w/ different expertise** (ML researcher, data scientist, domain expert)
- **#2 Support for local, distributed, and federated ML**, optimizing compiler and parallelization strategies
- **#3 Underlying data model of heterogeneous tensors** w/ native support for lineage tracing and exploitation, and automatic data reorganization and specialization
- **We're open: early adopters, comparisons, collaborations**