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 15, 2019

Agenda

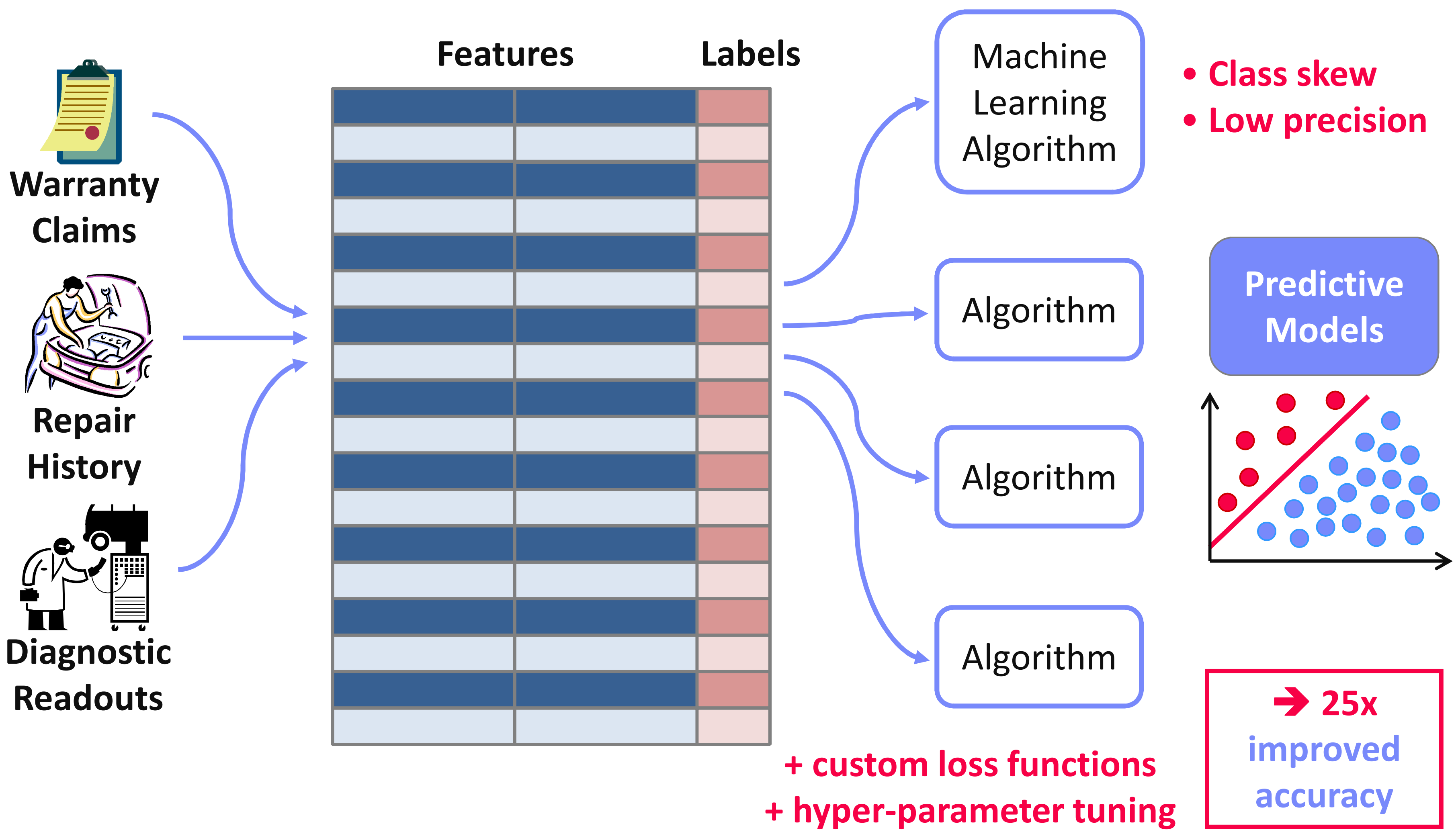
- **Motivation and Goals**
- **Data Management Group**
- **Course Organization**
- **Course Outline and Projects**
- **Overview Apache SystemML**

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)
- **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)

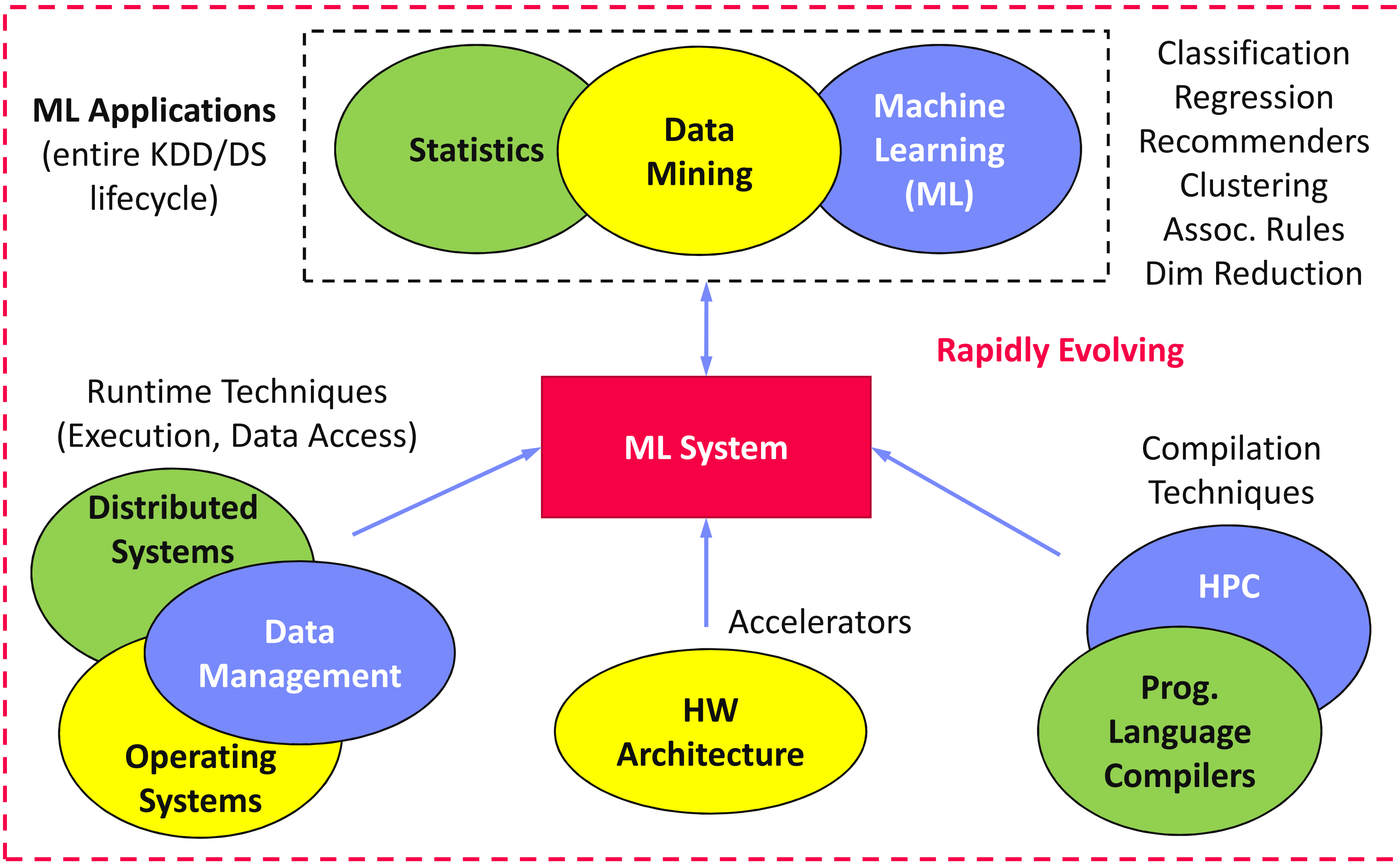
A Car Reacquisition Scenario



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)
- **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?

- **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, lots of different languages and system architectures, but many shared concepts

- **Course Objectives:**
  - Architecture and internals of modern (large-scale) ML systems
  - **#1** Understanding of characteristics → **better evaluation / usage**
  - **#2** Understanding of effective techniques → **build/extend ML systems**

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)
- **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

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

DB group

Data Management Courses

- **SS: Databases / Databases 1 (DM)**
  - Data management from user/application perspective
  - VU 1.5/1.5 (4 ECTS), and VU 1/1 (3 ECTS)
- **SS: Architecture of ML Systems (AMLS)**
  - Internals of machine learning systems
  - VU 2/1 (5 ECTS), master, github.com/tugraz-isds/systemds

- **WS: Data Integration and Large-Scale Analysis (DIA)**
  - Distributed data and information systems
  - VU 2/1 (5 ECTS), bachelor/master
- **WS: Architecture of Database Systems (ADBS)**
  - Internals of database management systems
  - VU 2/1 (5 ECTS), master

Course Organization

Basic Course Organization

- **Staff**
  - Lecturer: Univ.-Prof. Dr.-Ing. Matthias Boehm, ISDS
  - Assistant: M. Tech. Arnab Phani, 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), master only
  - **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 and project presentation)
  - **Final oral exam** (by appointment)
  - **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/ss19_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
    - SystemML: https://github.com/apache/systemml
    - 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

Course Outline

Part A-C: Architecture, Compiler, Runtime

A: Introduction

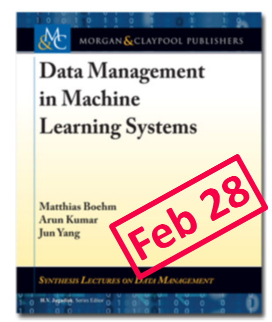
- **01 Introduction and Overview** [Mar 15]
- **02 Languages, Architectures, and System Landscape** [Mar 22]

B: Rewrites and Optimization

- **03 Size Inference, Rewrites, and Operator Selection** [Mar 29]
- **04 Operator Fusion and Runtime Adaptation** [Apr 05]

C: Execution Strategies

- **05 Data- and Task-Parallel Execution** [Apr 12]
- **06 Parameter Servers** [May 03]
- **07 Hybrid Execution and HW Accelerators** [May 10]



Part D-F: Data Access and ML Lifecycle

D: Data Storage and Access

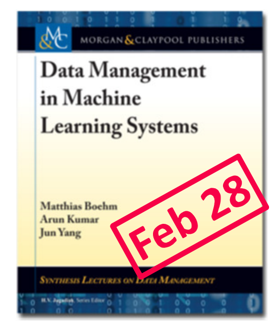
- **08 Formats, Caching, Partitioning, and Indexing** [May 17]
- **09 Lossy and Lossless Compression** [May 24]

E: ML Lifecycle Systems

- **10 Data Acquisition, Cleaning, and Preparation** [Jun 07]
- **11 Model Selection and Management** [Jun 14]
- **12 Model Deployment and Serving** [Jun 21]

F: Wrap-Up

- **14 Project Presentations, Conclusions, Q&A** [Jun 28]



Project Overview

- **Team**
  - Individuals or two-person teams (w/ clearly separated responsibilities)
- **Objectives**
  - Non-trivial feature in an open source ML system (**2 ECTS → 50 hours**)
  - OSS processes: Break down into 3-7 tasks, code/tests/docs, PR per task, code review, incorporate review comments, etc
- **Target Systems**
  - Preferred: **SystemDS**, or **Apache SystemML**
  - Other options: Julia, TensorFlow, PyTorch, <your_favorite_project>
- **Timeline**
  - **Mar 22:** List of projects and discussions
  - **Apr 05:** Project selection
  - Last lecture: 5-10min **project presentation**, including demo!

Example Projects

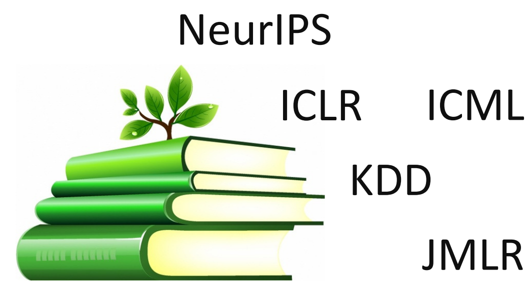
- **#1: Auto Differentiation**
  - Implement auto differentiation for deep neural networks
  - Integrate auto differentiation framework in compiler or runtime
- **#2: Sparsity-Aware Optimization of Matrix Product Chains**
  - Integrate sparsity estimators into DP algorithm
  - Extend DP algorithm for DAGs and other operations
- **#3 Parameter Server Update Schemes**
  - New PS update schemes: e.g., stale-synchronous, Hogwild!
  - Language and local/distributed runtime extensions
- **#4 Extended I/O Framework for Other Formats**
  - Implement local readers/writers for NetCDF, HDF5, libsvm, and/or Arrow
- **#5: LLVM Code Generator**
  - Extend codegen framework by LLVM code generator
  - Native vector library, native operator skeletons, JNI bridge

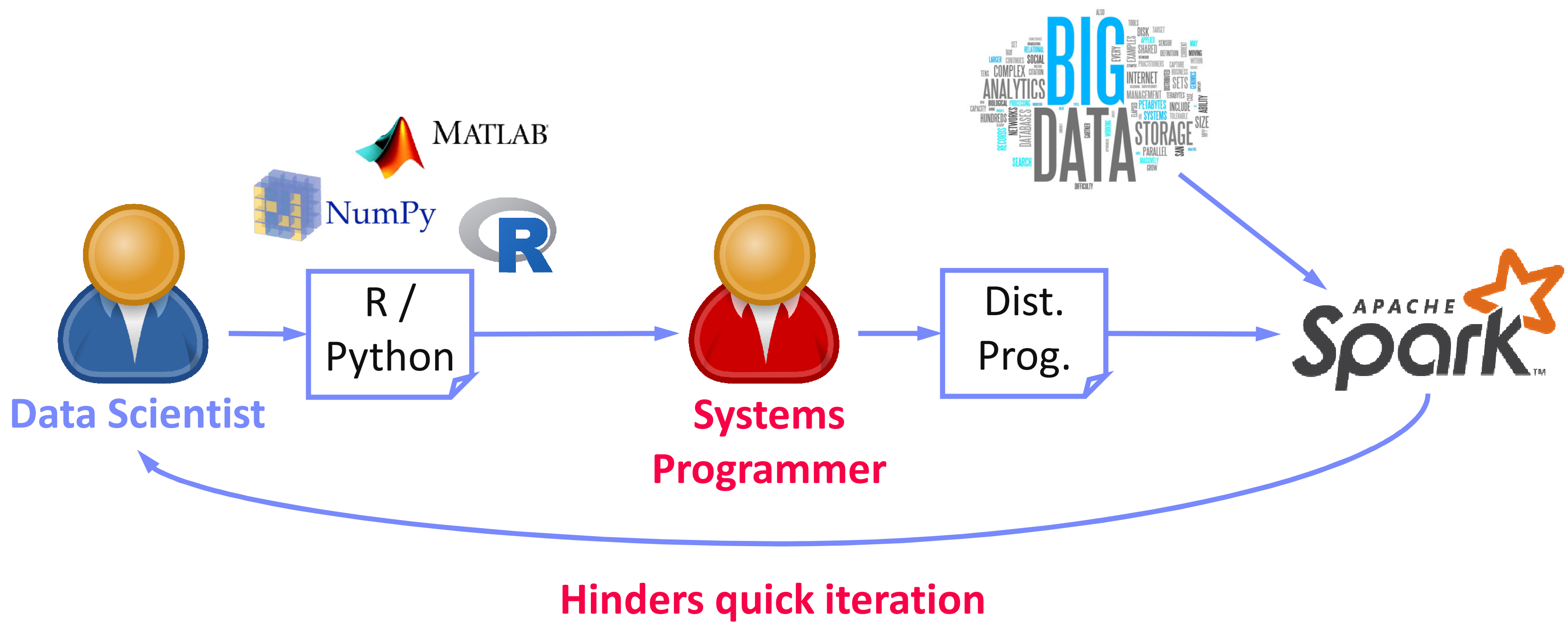
Overview Apache SystemML

Declarative Large-Scale Machine Learning

Common Large-Scale ML Challenges

- **#1 Custom ML Algorithms**
  - Huge diversity of existing ML algorithms
  - Cutting- / bleeding-edge algorithms
  - Domain-specific extensions init/loss





Common Large-Scale ML Challenges

- **#1 Custom ML Algorithms**
  - Huge diversity of existing ML algorithms
  - Cutting- / bleeding-edge algorithms
  - Domain-specific extensions init/loss

- **#2 Changing Environment**
  - Sample vs large-scale datasets (data size)
  - Dense/sparse, #features (data characteristics)
  - Single-node vs cluster (cluster characteristics)

"Hellerstein's Inequality"

$$\frac{\Delta env}{\Delta t} \gg \frac{\Delta app}{\Delta t}$$

- **#3 Integration and Deployment**
  - Data preparation and feature engineering
  - **Batch** and mini-batch training/scoring
  - Low-latency scoring (streaming)
  - Scale-up, **scale-out**, GPUs (hardware)

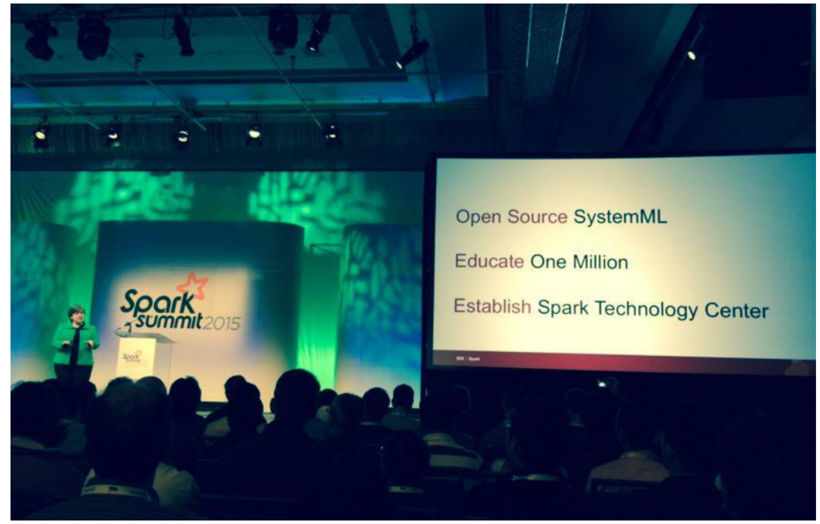
R / Python

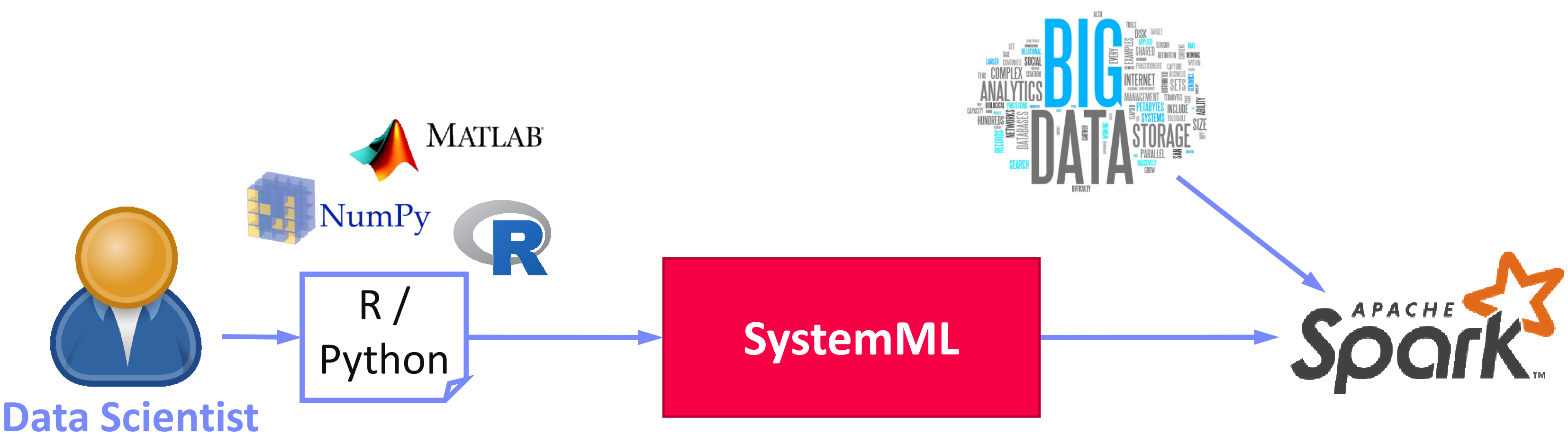
"Write Once, Run Anywhere"

Apache SystemML

Apache SystemML™

05/2017 Apache Top-Level Project
11/2015 Apache Incubator Project
08/2015 Open Source Release
01/2012 Integration in IBM BigInsights
01/2010 Project Kickoff





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

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

DML (**D**eclarative **M**achine **L**earning **L**anguage)

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:

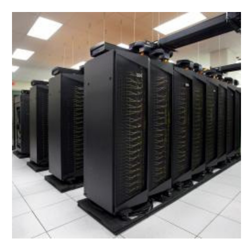
GPU

since 2014/16

Java

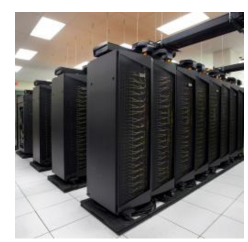
since 2012

hadoop



since 2010/11

APACHE Spark



since 2015

Basic HOP and LOP DAG Compilation

LinregDS (Direct Solve)

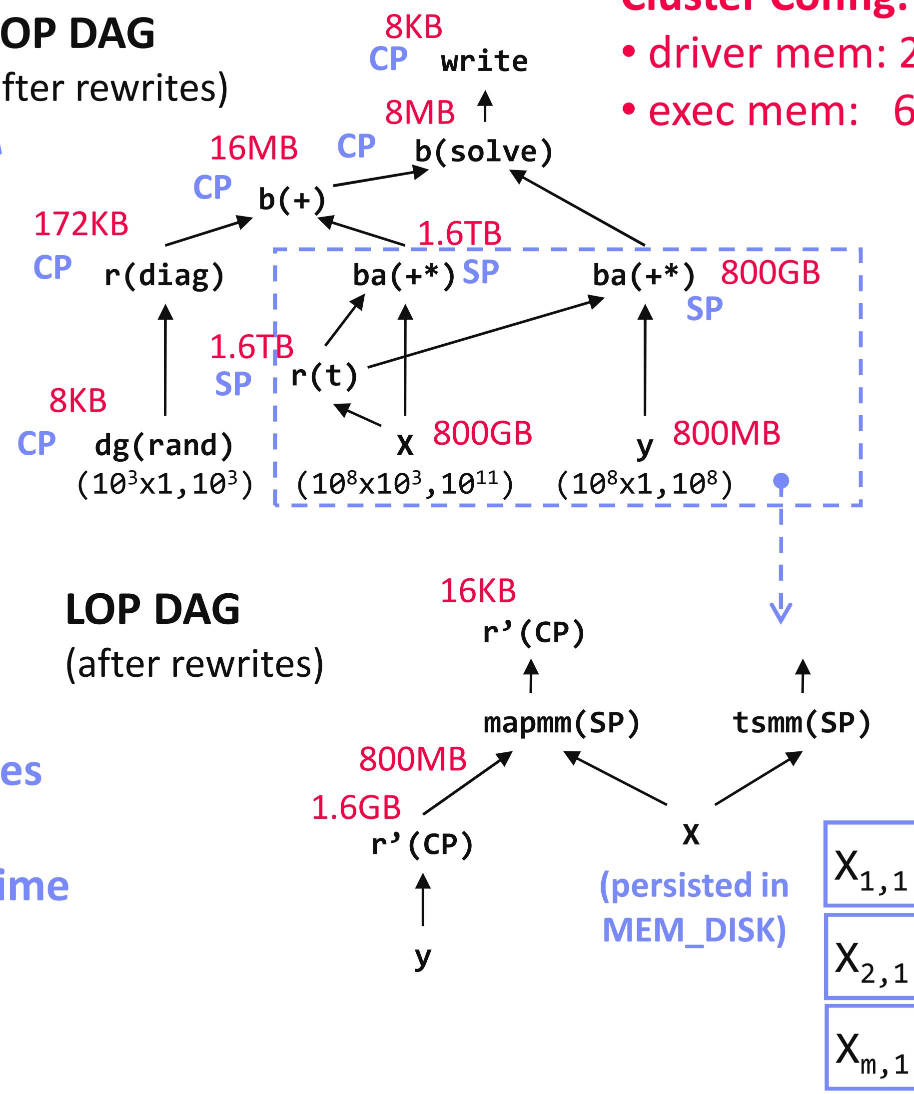
```
X = read($1);
y = read($2);
intercept = $3;
lambda = 0.001;
...
if( intercept == 1 ) {
  ones = matrix(1, nrow(X), 1);
  X = append(X, ones);
}
I = matrix(1, ncol(X), 1);
A = t(X) %*% X + diag(I)*lambda;
b = t(X) %*% y;
beta = solve(A, b);
...
write(beta, $4);
```

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

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

HOP DAG
(after rewrites)

LOP DAG
(after rewrites)



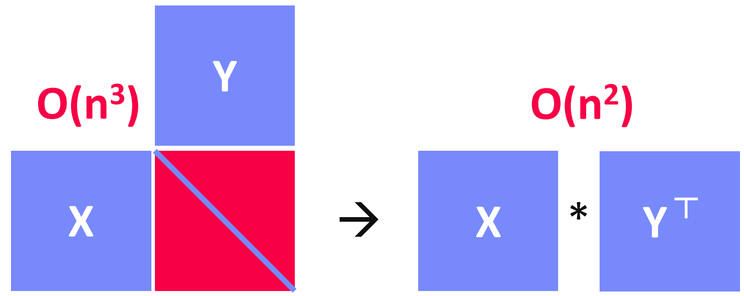
➔ Hybrid Runtime Plans:
- Size propagation / memory estimates
- Integrated CP / Spark runtime
- Dynamic recompilation during runtime

➔ 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

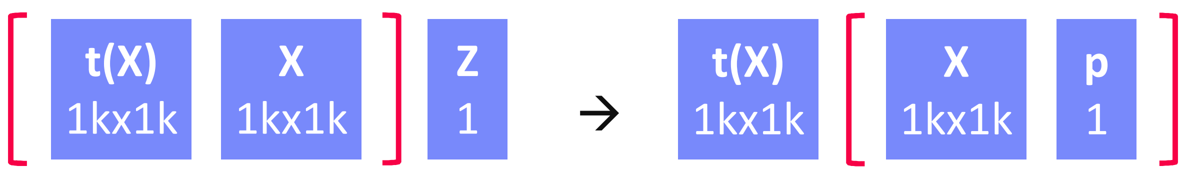
`trace(X%*%Y) → sum(X*t(Y))`



`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`

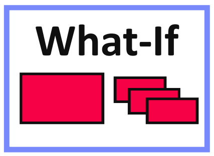


2,002 MFLOPs

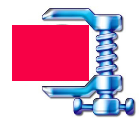
4 MFLOPs

Size propagation and sparsity estimation

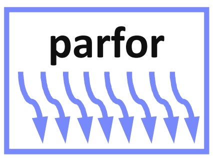
Selected Research Results



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

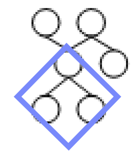


#4 Compressed Linear Algebra
(PVLDB'16,
SIGMOD Record'17,
VLDB Journal'18, CACM'19)



#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)

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

Lessons on Declarative Specification

- **L1: Importance of Data Independence and Logical Operations**
  - **Protection of investments** (adaptation to changing technology stack)
  - Simplification of **development** (especially libs) and **deployment**
  - Adaptation to **data/cluster characteristics**, **but** harder to optimize
  - Allows optimizations such as **resource op**, **compression** and **fusion**

- **L2: User Categories** (|Alg. Users| **>>** |Alg. Developers|)
  - Algorithm developers/researchers → Linear algebra
  - Algorithm users → ML libraries
  - Domain experts → ML tasks / AutoML

Alg. Users

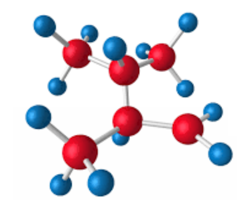
- **L3: Importance of Real Applications and Users**
  - Language for ML is wild west, **no standards** (PMML, PFA, ONNX)
  - **Unseen data and algorithm characteristics**
  - **Source of new APIs, features and optimizations**
  - Variety of apps / use cases → **balance generality / specialization**

Lessons on Data Model

- **L4:** Diversity of ML Algorithms / Applications
  - Broad range of algorithms (stats, ML, 2^{nd}-order optim)
  - Model choice often a **cost-benefit tradeoff**
  - Complex ML applications (rules, models, etc)
  - Opportunities of data programming and augmentation

- **L5: Users want Consolidated Lifecycle / Structured Data**
  - **Boundary crossing** for data integration, cleaning, feature engineering, training, and scoring is obstacle
  - **Heterogeneous input/output data**, with **structure**
  - Poor support for **provenance and model versioning**
  - APIs for embedded, low-latency scoring

TFX



Incl. Meta Data

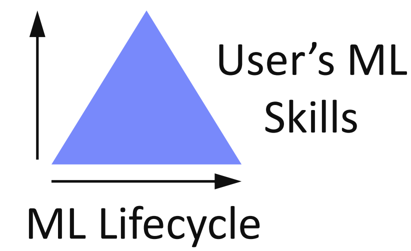
- **L6: Data Model very Difficult to Change**
  - Internal format extensions (e.g., dense/sparse, type) are major efforts
  - All combinations of data representations virtually impossible to test
  - **Deep integration of tensors** equivalent to new system

SystemDS™ Overview

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

- **Overview**
  - Open source **System** for end-to-end **D**ata **S**cience lifecycle
  - Data integration/cleaning, ML training, serving

- **Stack of Declarative Languages**
  - **Language hierarchy for tasks and users**
  - **Unified DSL and layering** for interop., reuse, opt
  - **Data model:** Heterogeneous tensors (w/ schema)



- **Key Features**
  - **#1: Data integration and cleaning**, outliers, feature engineering
  - **#2: ML model training**, tuning, validation, and **serving**
  - **#3: Data provenance and model versioning** → explainability
  - **#4: ML+Rules:** incorporate domain-expert and compliance rules
  - **Hybrid runtime plans**: local/distributed, data/task/PS/federated
  - **Horizontal and vertical optimization; sparsity exploitation**