Apache SystemML: Declarative, Large-Scale Machine Learning
From System Overview to Lessons Learned

Matthias Boehm
Graz University of Technology
Graz, Austria

Acknowledgements: Apache SystemML Team,
IBM Research – Almaden, IBM Spark Technology Center
About Me

- **09/2018 TU Graz, Austria**
  - BMVIT endowed chair for Data Management
  - **Data management** for data science
    (ML systems internals, data systems integration, deployment)

- **2012-2018 IBM Research – Almaden, USA**
  - Declarative large-scale machine learning
  - Optimizer and runtime of [Apache SystemML](https://systemml.apache.org)

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

ML Applications (entire KDD/DS lifecycle)
- Statistics
- Data Mining
- Machine Learning (ML)

Classification
- Regression
- Recommenders
- Clustering
- Assoc. Rules
- Dim Reduction

Runtime Techniques (Execution, Data Access)
- Distributed Systems
- Data Management
- Operating Systems

“Anwendungsübergreifende IKT-Forschung (5.6M €)”

Compilation Techniques
- HPC
- Prog. Language Compilers

ML System
- Accelerators
- HW Architecture

Hardware
- Accelerators
- HW Architecture

Software
- Prog. Language Compilers
- HPC

Data Science
- Machine Learning (ML)
- Data Mining
- Statistics

Applications
- ML Applications (entire KDD/DS lifecycle)
- Classification
- Regression
- Recommenders
- Clustering
- Assoc. Rules
- Dim Reduction

Deep Learning
- Machine Learning (ML)
- Data Mining
- Statistics

Natural Language Processing
- Prog. Language Compilers
- HPC

High Performance Computing
- Accelerators
- HW Architecture
Example ML Applications

- **Transportation / Space**
  - Lemon car detection and reacquisition (classification, sequence mining)
  - Airport passenger flows from WiFi data (time series forecasting)
  - Satellite sensor analytics (regression and correlation)

- **Finance**
  - Water cost index based on various influencing factors (regression)
  - Insurance claim cost per customer (model/feature 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 and patient similarity (classification, clustering)
  - Patient survival analysis and prediction (Cox regression, Kaplan-Meier)
Example ML Applications, cont.

- **Other Domains**
  - **Machine data: errors and correlation** (bivariate stats, sequence mining)
  - Smart grid: energy demand/RES supply and weather models (forecasting)
  - Visualization: dimensionality reduction into 2D (auto encoder)

- **Information Extraction**
  - **NLP contract sentences → rights/obligations** (classification, error analysis)
  - **PDF table recognition and extraction** (NMF clustering, custom processing)
  - OCR: optical character recognition (preprocessing, classification)

- **Algorithm Research**
  - **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
  - (Dozens of state-of-the-art algorithms from the literature)
Common Large-Scale ML Challenges

- **#1 Custom ML Algorithms**
  - Huge diversity of existing ML algorithms
  - Cutting-/bleeding-edge algorithms
  - Domain-specific extensions (e.g., initializations, loss functions)
Common Large-Scale ML Challenges

- **#1 Custom ML Algorithms**
  - Huge diversity of existing ML algorithms
  - Cutting-/ bleeding-edge algorithms
  - Domain-specific extensions (e.g., initializations, loss functions)

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

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

\[ \frac{\Delta env}{\Delta t} \gg \frac{\Delta app}{\Delta t} \]

“Hellerstein’s Inequality”

“Write Once, Run Anywhere”
Apache SystemML™

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

SystemML: Overview and Architecture


An 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");

Compute matrices from HDFS
Compute initial gradient
Compute conjugate gradient
Update model and residuals

"Separation of Concerns"
High-Level SystemML Architecture

**APIs:** Command line, JMLC, Spark MLContext, Spark ML, (Scala, Python, Java DSLs)

Library of 20+ scalable algorithms

DML/PyDML Scripts

Language

Compiler

Runtime

In-Memory Single Node (scale-up)

Hadoop or Spark Cluster (scale-out)

In-Progress:

- GPU since 2014/16
- Python since 2012
- DML (Declarative Machine Learning Language) since 2010/11
- Spark since 2015

Apache SystemML™
Basic HOP and LOP DAG Compilation

LinregDS (Direct Solve)

\[ X = \text{read}(\$1); \]
\[ y = \text{read}(\$2); \]
\[ \text{intercept} = \$3; \]
\[ \text{lambda} = 0.001; \]

\[
\begin{align*}
\text{if} ( \text{intercept} == 1 ) \{ \\
\quad \text{ones} = \text{matrix}(1, \text{nrow}(X), 1); \\
\quad X = \text{append}(X, \text{ones}); \\
\}
\]

\[ I = \text{matrix}(1, \text{ncol}(X), 1); \]
\[ A = \text{t}(X) \%\% X + \text{diag}(I)\%\%\text{lambda}; \]
\[ b = \text{t}(X) \%\% y; \]
\[ \text{beta} = \text{solve}(A, b); \]

\[ \text{write}(\beta, \$4); \]

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

Hybrid Runtime Plans:
- Size propagation over ML programs
- Worst-case sparsity / memory estimates
- Integrated CP / Spark runtime
- Dynamic recompilation during runtime
Selected Research Results
DB-Inspired Data Management in ML Systems

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

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

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

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

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

#6 Sum-Product Optimization
holistic framework for automatic rewrites (CIDR’17)

GPU, meta learning, numerical stability, parameter servers, etc
Data Management in ML Systems


- Lessons Learned (subset)
- SystemDS at TU Graz
Lessons on Declarative Specification
(“The notion of declarative specification is evolving”)

- **L1: Importance of Data Independence and Logical Operations**
  - Protection of investments (adaptation to changing technology stack)
  - Simplification of development (especially library algorithms), and deployment (e.g., large-scale vs embedded training/scoring)
  - Adaptation to data/cluster characteristics, but harder to optimize
  - Allows optimizations such as resource op, compression and fusion

  - Algorithm developers/researchers → Linear algebra
  - Algorithm users → ML libraries
  - Domain experts → ML tasks / AutoML

- **L3: Importance of Real Applications and Users**
  - Language abstractions for ML is wild west, no standards
  - Unseen data and algorithm characteristics
  - Source of new APIs, features and optimizations
  - Variety of applications / use cases → balance generality / specialization
Lessons on Data Model

- **L4: Diversity** of ML Algorithms / Applications
  - DL + mini-batch SGD + parameter server sufficient? **NO!**
  - a) **Broad range of algorithms** (stats, ML, 2\textsuperscript{nd}-order optim)
  - b) Model choice often a **cost-benefit tradeoff**
  - c) **Complex ML applications** (rules, models, etc)

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

- **L6: Data Model very hard 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 and Language

- **Overview**
  - *System* support for entire *Data* *Science* lifecycle
  - Data integration/cleaning, ML training, serving

- **Stack of Declarative Languages**
  - Two-dimensional *language hierarchy* for tasks and users
  - *Unified DSL and layering* for interoperability, reuse, and automatic optimization

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

("IKT-Themenfelder:
A) Systems of Systems
B) Intelligente Systeme
D) Schnittstellen von Systemen")
Limitations of Existing ML Systems

- ML lifecycle requires management of **heterogeneous, structured data**
- Existing systems limited to **homogeneous tensors** (ML systems -> scalar cells, array databases → structured cells) or 2D datasets/frames (Spark, SystemML, TF)
- Data model hardest aspect to change in existing system

DataTensors

- Generic data model as basis for hierarchy of specification languages
- Specialization during runtime (data reorganization, compression, etc)

Status

- Forked SystemML 1.2
- In progress of building basic system
- Open source soon

We’re hiring PhD students
Open for Collaborations