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

**Matthias Boehm**

Graz University of Technology

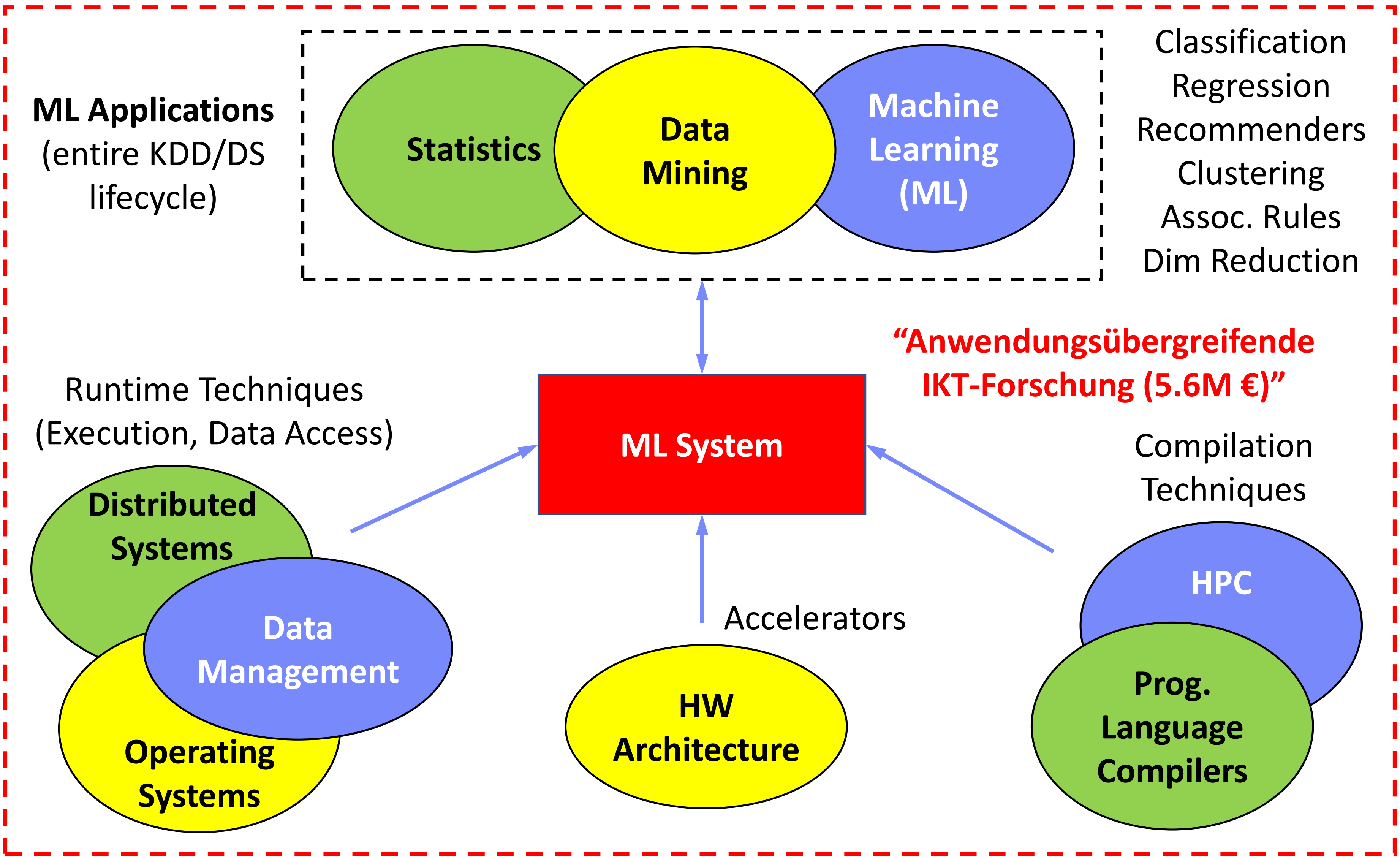
Graz, Austria

**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
- **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?



# Example ML Applications

- **Transportation / Space**
  - **Lemon car detection and reacquisition** (classification, sequence 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/feature selection, regression)
  - **Financial analysts survey correlation** (bivariate stats w/ new tests)

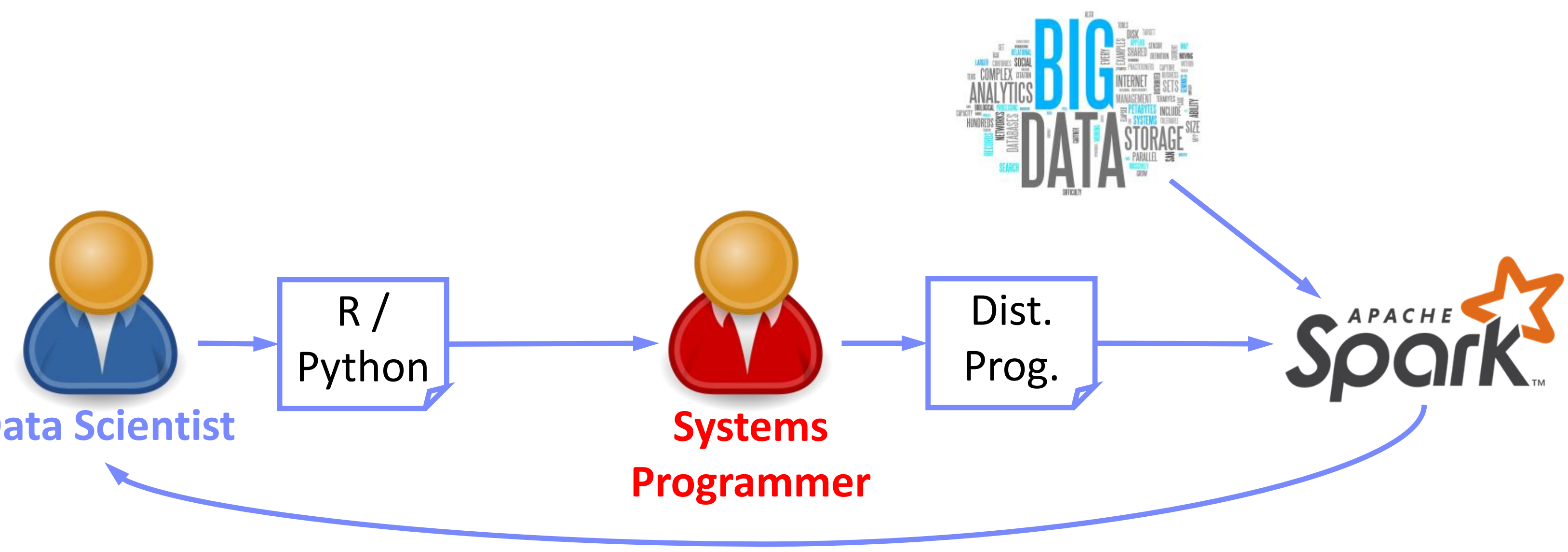
- **Health Care**
  - **Breast cancer cell grow from histopathology images** (classification)
  - **Glucose trends and warnings** (clustering, classification)
  - Emergency room diagnosis and patient similarity (classification, clustering)
  - Patient survival analysis and prediction (Cox regression, 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)


Hinders quick iteration

# 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)

"Hellerstein's Inequality"

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

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

R / Python

"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


N. Pansare, M. Dusenberry, N. Jindal, M. Boehm, B. Reinwald, P. Sen: Deep Learning with Apache SystemML. **SysML 2018**.

M. Boehm, M. Dusenberry, D. Eriksson, A. V. Evfimievski, F. Makari Manshadi, N. Pansare, B. Reinwald, F. Reiss, P. Sen, A. Surve, S. Tatikonda: SystemML: Declarative Machine Learning on Spark. **PVLDB 2016**.

B. Huang*, M. Boehm, Y. Tian, B. Reinwald, S. Tatikonda, F. R. Reiss: Resource Elasticity for Large-Scale Machine Learning. **SIGMOD 2015**.

M. Boehm, D. R. Burdick, A. V. Evfimievski, B. Reinwald, F. R. Reiss, P. Sen, S. Tatikonda, Y. Tian: SystemML's Optimizer: Plan Generation for Large-Scale Machine Learning Programs. **IEEE Data Eng. Bull. 2014**.

M. Boehm, S. Tatikonda, B. Reinwald, P. Sen, Y. Tian, D. Burdick, S. Vaithyanathan: Hybrid Parallelization Strategies for Large-Scale Machine Learning in SystemML. **PVLDB 2014**.

A. Ghoting, R. Krishnamurthy, E. P. D. Pednault, B. Reinwald, V. Sindhwani, S. Tatikonda, Y. Tian, S. Vaithyanathan: SystemML: Declarative Machine Learning on MapReduce. **ICDE 2011**.

# An 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

DML/PyDML Scripts

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

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

Library of 20+ scalable algorithms

**Language**

**Compiler**

**Runtime**

Apache SystemML™

**In-Memory Single Node**
(scale-up)

**Hadoop or Spark Cluster**
(scale-out)

**In-Progress:**

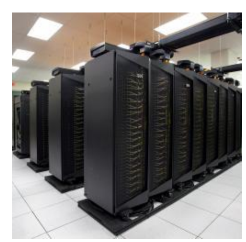
GPU

**since 2014/16**

Java

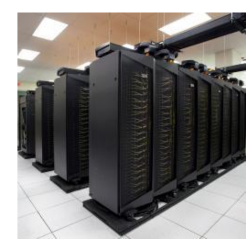
**since 2012**

hadoop



**since 2010/11**

APACHE Spark



**since 2015**

# Basic HOP and LOP DAG Compilation

**LinregDS (Direct Solve)**

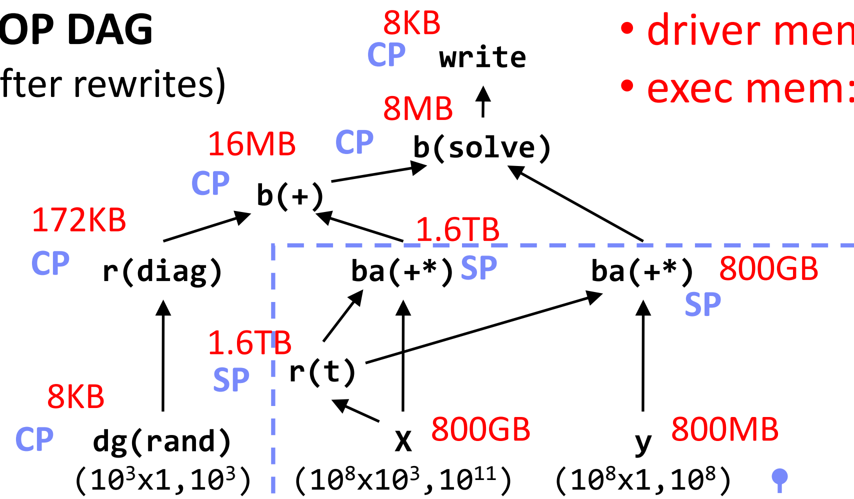
```
X = read($1);
y = read($2);
intercept = $3;
lambda = 0.001;
...
```

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

```
if( intercept == 1 ) {
  ones = matrix(1, nrow(X), 1);
  X = append(X, ones);
}
```

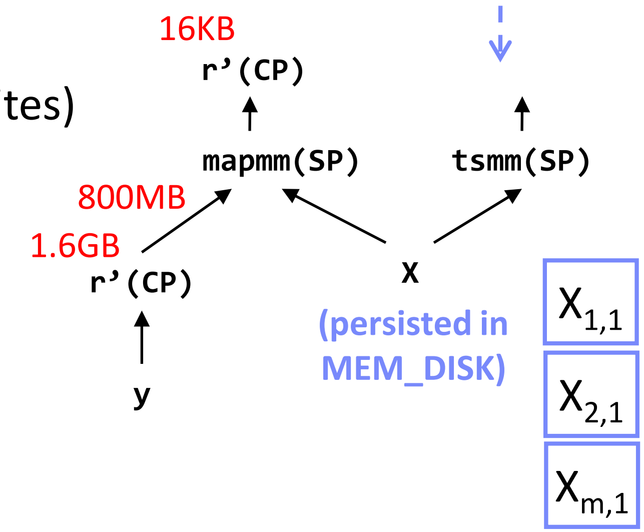
```
I = matrix(1, ncol(X), 1);
A = t(X) %*% X + diag(I)*lambda;
b = t(X) %*% y;
beta = solve(A, b);
...
write(beta, $4);
```

**HOP DAG**
(after rewrites)




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

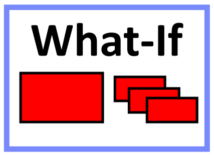
**LOP DAG**
(after rewrites)



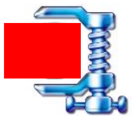

➔ **Hybrid Runtime Plans:**
- **Size propagation over ML programs**
- **Worst-case sparsity / memory estimates**
- **Integrated CP / Spark runtime**
- **Dynamic recompilation during runtime**

# Selected Research Results

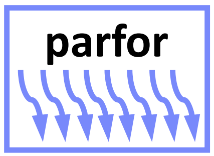
## DB-Inspired Data Management in ML Systems




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

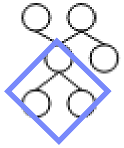


**#4 Compressed Linear Algebra**
(PVLDB'16,
SIGMOD Record'17,
VLDB Journal'17, CACM'18)




**#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)

$\Sigma\Pi$ **#6 Sum-Product Optimization**
holistic framework for automatic rewrites (CIDR'17)

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

# Data Management in ML Systems

M. Boehm, A. Kumar, J. Yang: Data Management in Machine Learning Systems. **Morgan & Claypool Publishers, 2019 (in preparation)**.

A. Kumar, M. Boehm, J. Yang: Data Management in Machine Learning: Challenges, Techniques, and Systems (Tutorial). **SIGMOD 2017**.

MORGAN & CLAYPOOL PUBLISHERS

- 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**

- **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 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, $2^{nd}$-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**

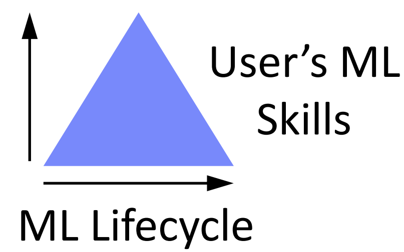
**Incl. Meta Data**

- **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

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

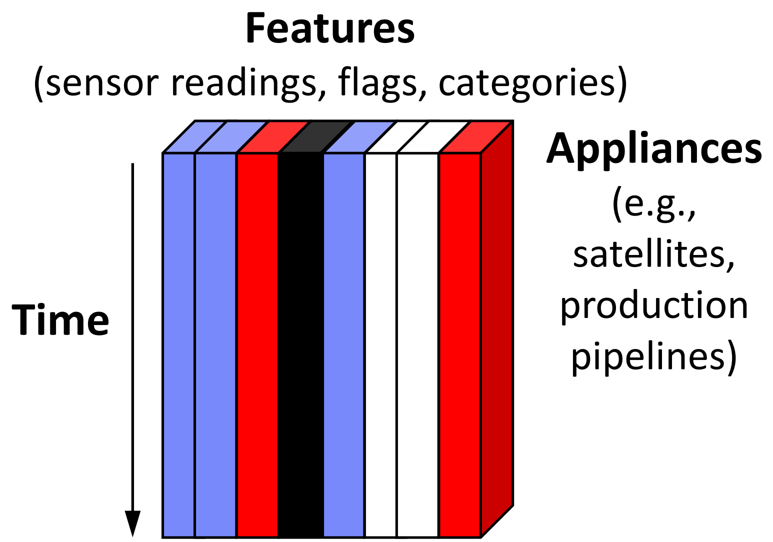
- **Overview**
  - System support for entire Data Science lifecycle
  - Data integration/cleaning, ML training, serving
- **Stack of Declarative Languages**
  - Two-dimensional **language hierarchy for tasks and users**
  - **Unified DSL and layering** for interoperability, reuse, and automatic optimization




- **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**

# SystemDS™
# Data Model and Status

- **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**