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
BMVIT endowed chair for Data Management


Last update: Oct 04, 2019

# Announcements/Org

- **#1 CS Talks x5 (Oct 15, 5pm**, Aula Alte Technik)
  - **Margarita Chli** (ETH Zurich)
  - Title: **How Robots See – Current Challenges and Developments in Vision-based Robotic Perception**


- **#2 Course Architecture of DB Systems**
  - **Canceled due to <10 students** and overload w/ other courses
  - Will be offered in WS2020/21, 706.543

- **#3 Course Intro International Entrepreneurship**
  - Basic and systematic understanding of international business, as well as markets and the people
  - Lecturer: Univ.-Prof. Dr. techn. Hongying Foscht
  - **Beginning Oct 9, 2019**; 4 ECTS, 706.319

# Announcements/Org, cont.

- **#4 Master Thesis – JOANNEUM RESEARCH Health**
  - **Thesis topic:** Development and validation of a hybrid decision model to identify frailty in older adults with care needs in geriatric care facilities
  - **Supervisors:** Klaus Donsa (JOANNEUM RESEARCH), Matthias Boehm (TU Graz), Peter Mrak (QiGG)
  - 60% part-time employment JOANNEUM RESEARCH, **8 months**, monthly salary of **€ 831**

# Agenda

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

**Architecture of Database Systems** (ADBS, **WS**)

DB system internals
+ prog. project

**Architecture of ML Systems** (AMLS, **SS**)

ML system internals
+ prog. project
in **SystemDS**

[github.com/tugraz-isds/systemds]

Master

Bachelor

**Data Integration and Large-Scale Analysis** (DIA, **WS**)

Distributed Data Management
(usage and internals)

**Data Management / Databases** (DM, **SS+WS**)

Data management from user/application perspective

# Team

## Staff Members

### Head

Matthias Boehm

Email: m.boehm@tugraz.at

Personal Website: matthiasboehm.org

### Researchers

Mark Dokter

Email: mdokter@know-center.at

### PhD Students

Arnab Phani

Email: arnab.phani@tugraz.at

Shafaq Siddiqi

Email: shafaq.siddiqi@tugraz.at

### Undergraduate Students

Benjamin Rath

Email: benjamin.rath@student.tugraz.at

Kevin Innerebner

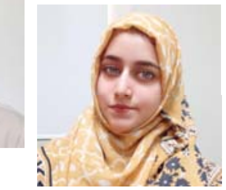
Email: innerebner@student.tugraz.at

Florijan Klezin

Email: fklezin@know-center.at

# 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

# Course Logistics

- **Website**
  - https://mboehm7.github.io/teaching/ws1920_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 staff)
  - **Final written exam** (oral exam if <10 students take the exam)
  - **Grading** (40% project/exercises, 60% exam)

# Course Logistics, cont.

- **Course Applicability**
  - **Bachelor** programs computer science (CS), as well as software engineering and management (SEM)
  - **Master** programs CS catalog “Knowledge Technologies”, and SEM catalog “Web and Data Science”
  - **Free subject course** in any other study program or university
  - Future master CS/SEM catalog “Data Science” (**unconfirmed**) → compulsory course in major/minor

# Course Motivation and Goals

# Data Sources and Heterogeneity

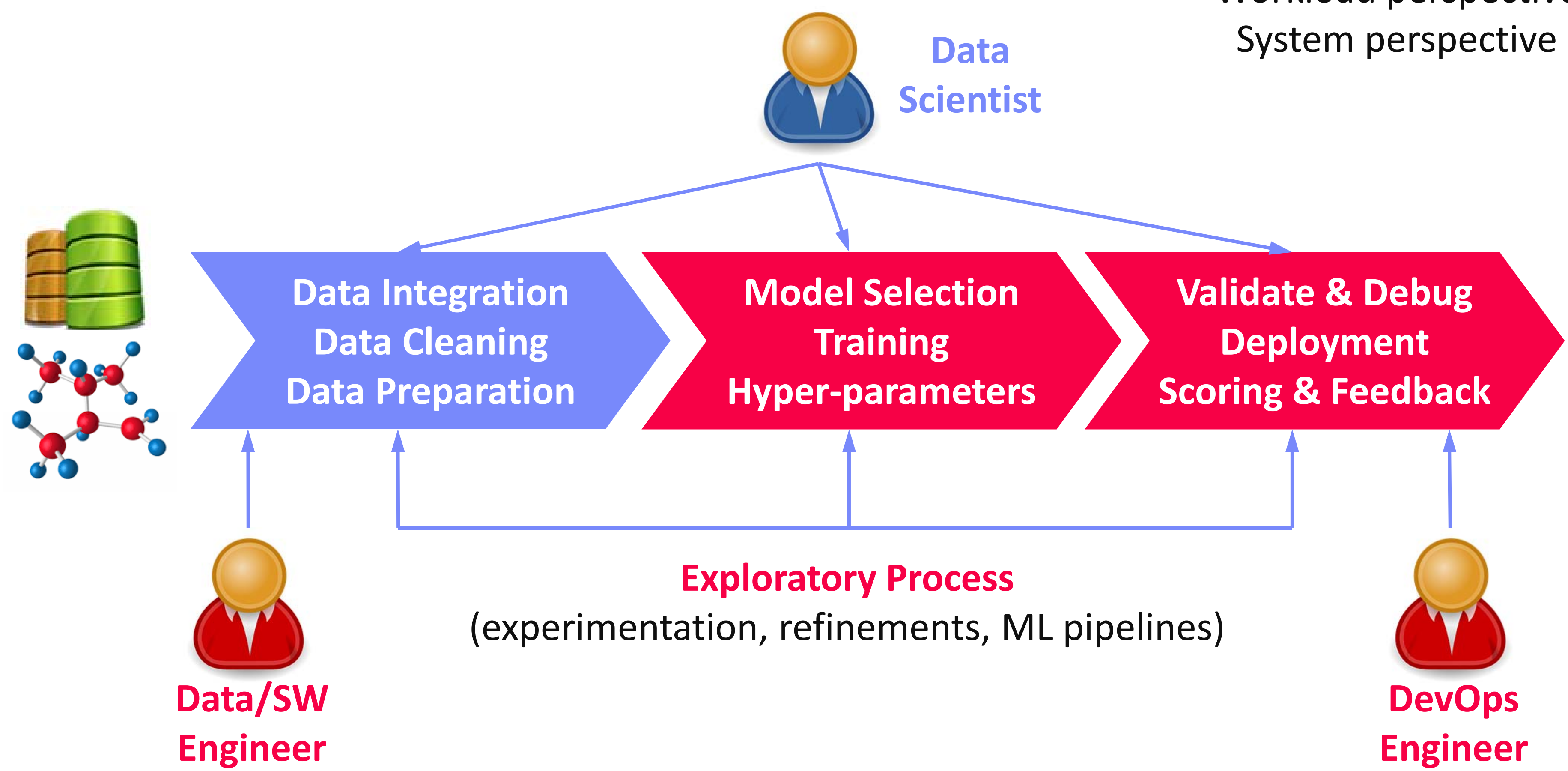
- **Terminology**
  - **Integration** (Latin integer = whole): consolidation of data objects / sources
  - **Homogeneity** (Greek homo/homoios = same): similarity
  - **Heterogeneity**: dissimilarity, different representation / meaning

- **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

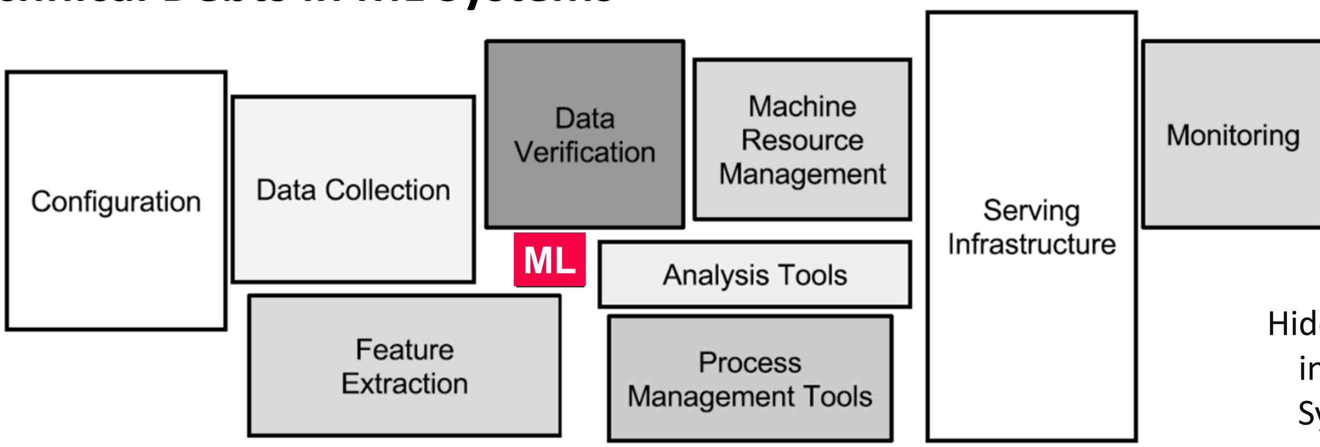


# The 80% Argument

- **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)**]

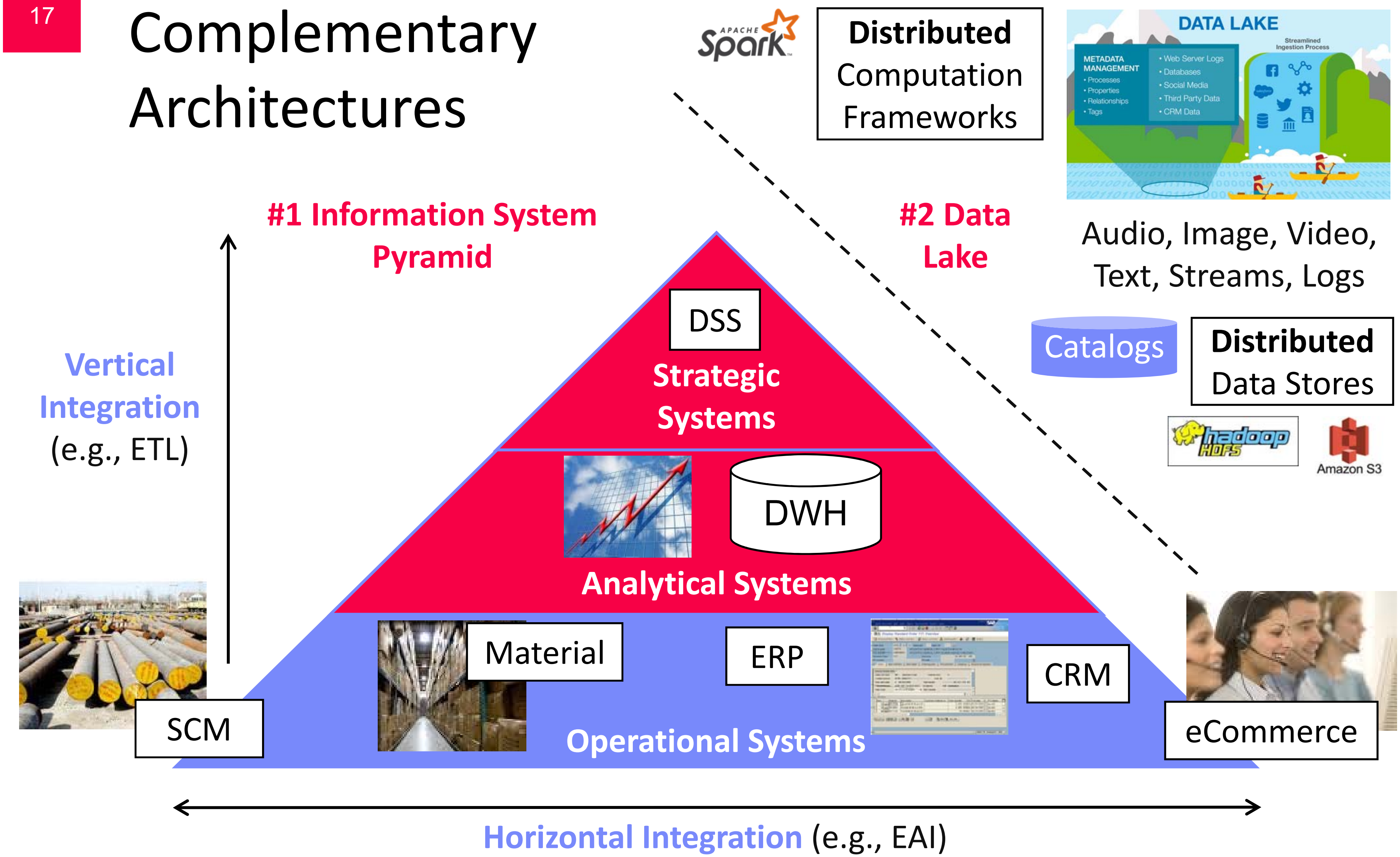
- **Technical Debts in ML Systems**




[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 04]
- **02 Data Warehousing, ETL, and SQL/OLAP** [Oct 11]
- **03 Message-oriented Middleware, EAI, and Replication** [Oct 18]

## Key Integration Techniques

- **04 Schema Matching and Mapping** [Oct 25]
- **05 Entity Linking and Deduplication** [Nov 08]
- **06 Data Cleaning and Data Fusion** [Nov 15]
- **07 Data Provenance and Blockchain** [Nov 22]

# Part B: Large-Scale Data Management & Analysis

## Cloud Computing

- **08 Cloud Computing Foundations** [Nov 29]
- **09 Cloud Resource Management and Scheduling** [Dec 06]
- **10 Distributed Data Storage** [Dec 13]

## Large-Scale Analysis

- **11 Distributed, Data-Parallel Computation** [Jan 10]
- **12 Distributed Stream Processing** [Jan 17]
- **13 Distributed Machine Learning Systems** [Jan 24]
- **14 Q&A and exam preparation** [Jan 31]

# Overview Projects or Exercises

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

- **Objectives**
  - Non-trivial programming project in DIA context (**2 ECTS → 50 hours**)
  - **Preferred:** Open source contribution to **SystemDS**
    - https://github.com/tugraz-isds/systemds
    - Topics throughout the stack (from HW to high-level scripting)
  - **Alternatively:** 3 of 4 provided exercises (2 per part)

- **Timeline**
  - **Oct 25:** List of projects proposals, feel free to bring your own
  - **Nov 08:** Binding project/exercise selection
  - **Jan 31:** Final project/exercise deadline

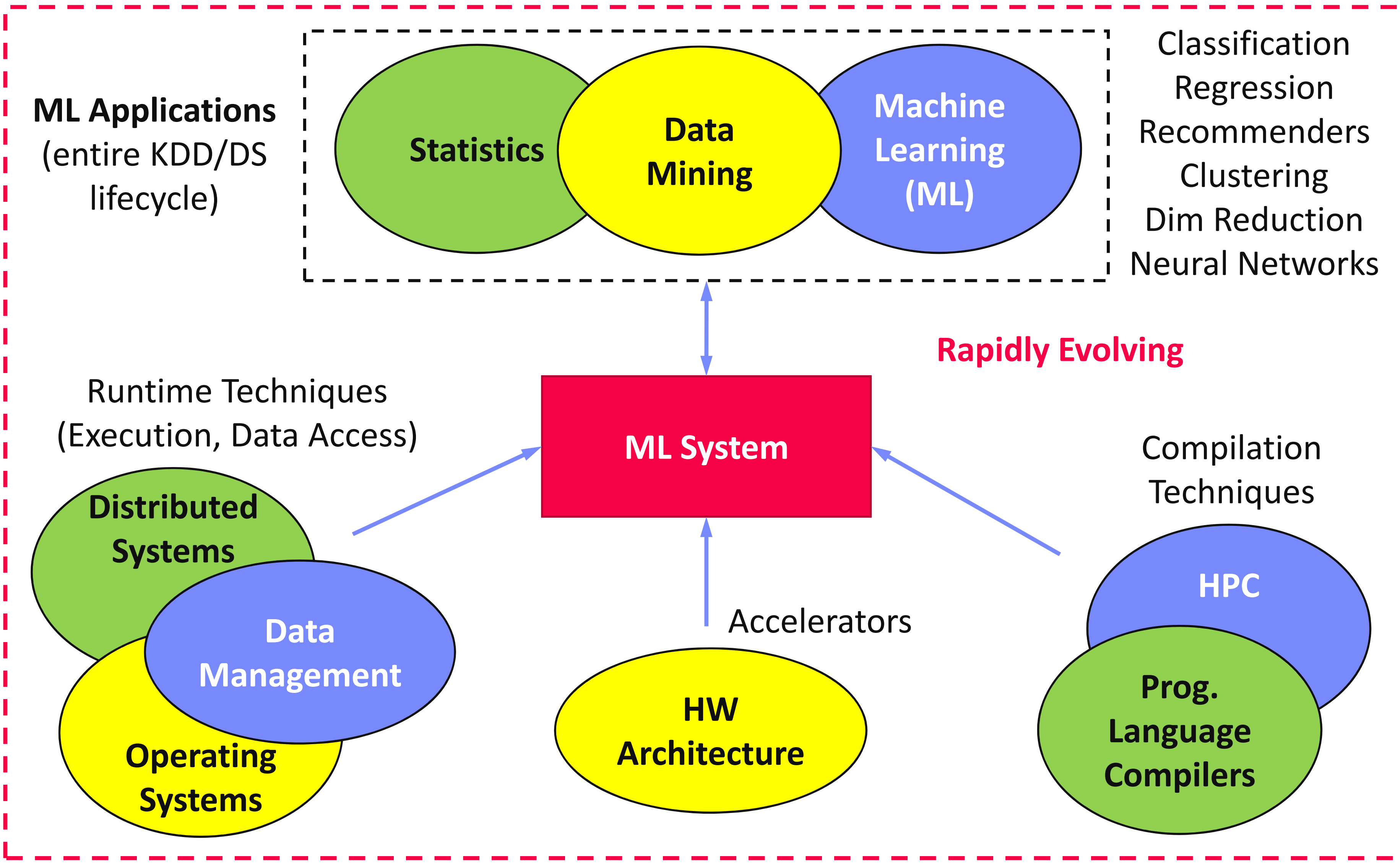
# Excursus: SystemDS

(An open source ML **system** for the end-to-end **d**ata **s**cience lifecycle )

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

https://arxiv.org/pdf/1909.02976.pdf

# What is an ML System?



# Motivation SystemDS

- **Existing ML Systems** (primarily ML training/scoring)
  - Variety of ML algorithms and lack of standards
  - **#1 Numerical computing** frameworks
  - **#2 ML Algorithm libraries** (local, large-scale)
  - **#3** Large-scale **linear algebra systems**
  - **#4 Deep neural network** (DNN) frameworks
- **Exploratory Data-Science Lifecycle**
  - **Open-ended problems** w/ underspecified objectives
  - Wide variety of heterogeneous data sources
  - Hypotheses, integrate data, run analytics, look for interesting patterns/models
  - **Unknown value** → lack of system infrastructure
    ➔ **Redundancy of manual efforts and computation**

**"Take these datasets and show value or competitive advantage"**

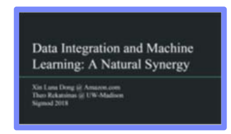
# Motivation SystemDS, cont.

- **Data Preparation Problem**
  - **80% Argument:** 80-90% of time for finding, integrating, cleaning data
  - Dedicated subsystems for data collection, verification, and extraction
  - Diversity of tools ➔ boundary crossing, lack of optimization
  - In-DBMS ML toolkits **largely unsuccessful** (stateful, data loading, verbose)

TFX

- **A Case for Declarative Data Science**
  - Specify data science lifecycle in **R or Python** syntax and **use stateless systems**
  - **Key observation**: SotA data integration based on ML
  - Data cleaning, outlier detection, data augmentation, feature and model selection, hyper-parameter optimization, model debugging
  - **Our approach:** High-level abstractions for data science lifecycle tasks, implemented in DSL for ML training/scoring
    ➔ **Avoid boundary crossing** and **optimizations across lifecycle**

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


Data Integration and Machine Learning: A Natural Synergy

**SystemDS**

# 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

DML Scripts

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

**Language**

**Compiler**

**Runtime**

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


**Apache SystemML™**

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

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

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

**In-Progress:**

GPU

**since 2014/16**

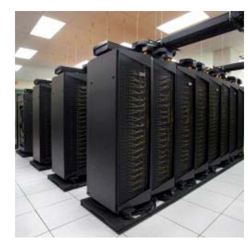
Java

**since 2012**

hadoop

**since 2010/11**

APACHE Spark



**since 2015**

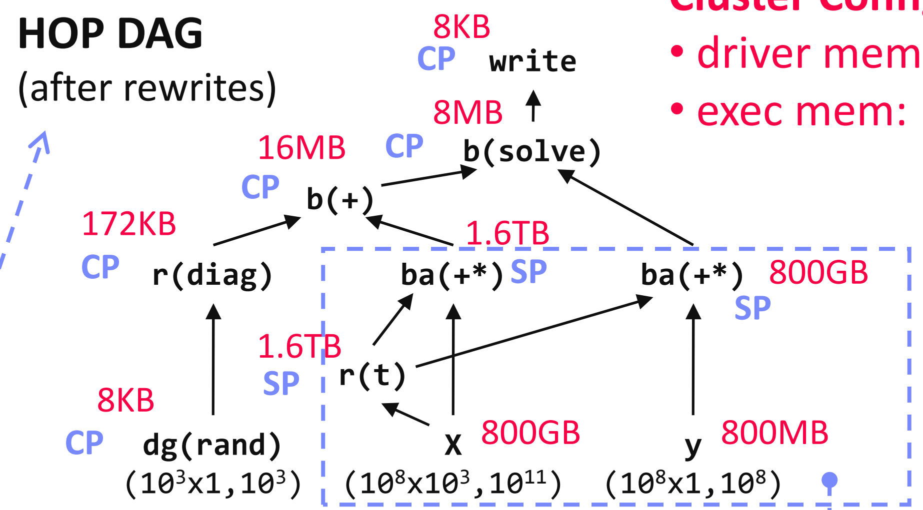
# Basic HOP and LOP DAG Compilation

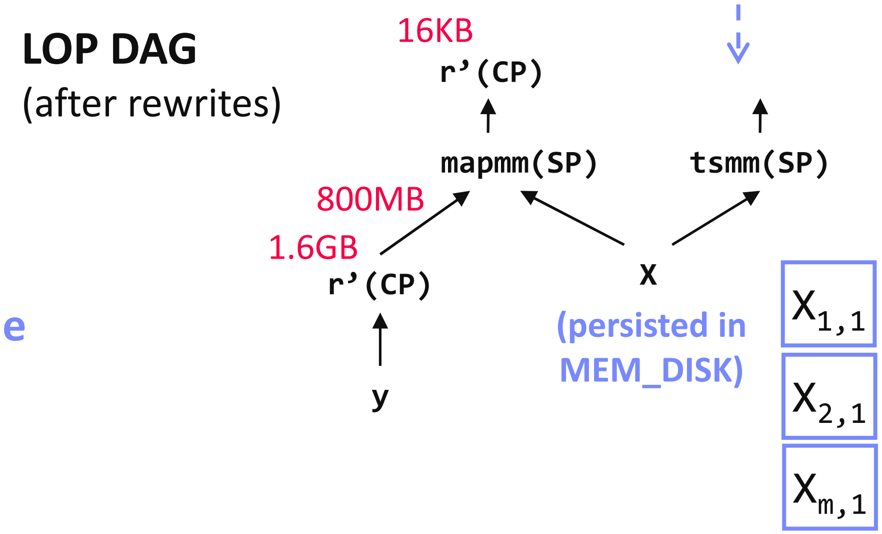
**LinregDS (Direct Solve)**

```
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



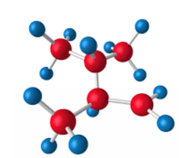

→ **Hybrid Runtime Plans:**
- Size propagation / memory estimates
- Integrated CP / Spark runtime
- Dynamic recompilation during runtime

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

# Lessons Learned from SystemML

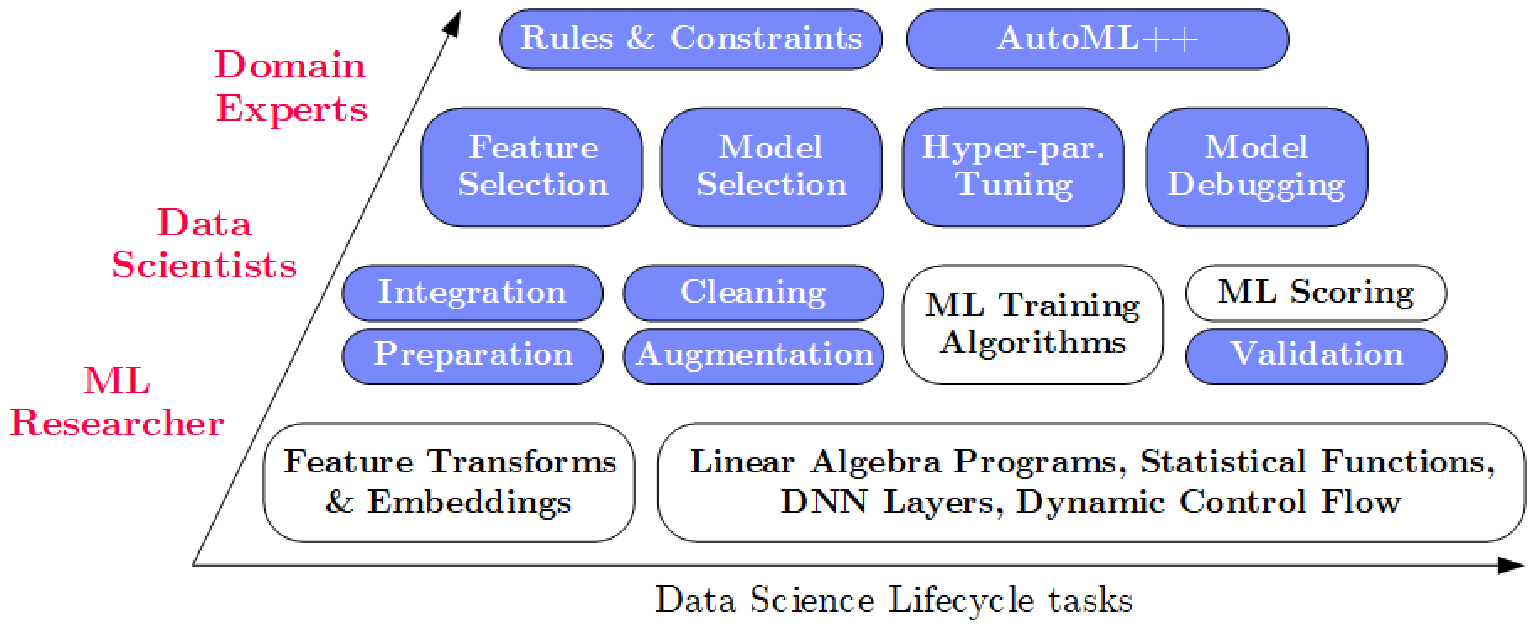
- **L1 Data Independence & Logical Operations**
  - Independence of **evolving technology stack** (MR → Spark, GPUs)
  - **Simplifies development** (libs) and **deployment** (large-scale vs. embedded)
  - **Enables adaptation** to cluster/data characteristics (dense/spare/compressed)
- **L2 User Categories** (|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**



# Language Abstractions and APIs

- **DSL with R-like Syntax**
  - Leverage SystemML's DML lang for linear algebra control flow programs (**L1**)
  - Extended by stack of declarative abstractions for **different users** (**L2**)
  - Mechanism for registering **DML-bodied built-in functions**



# 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

```
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)
} }
```

Feature Selection

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

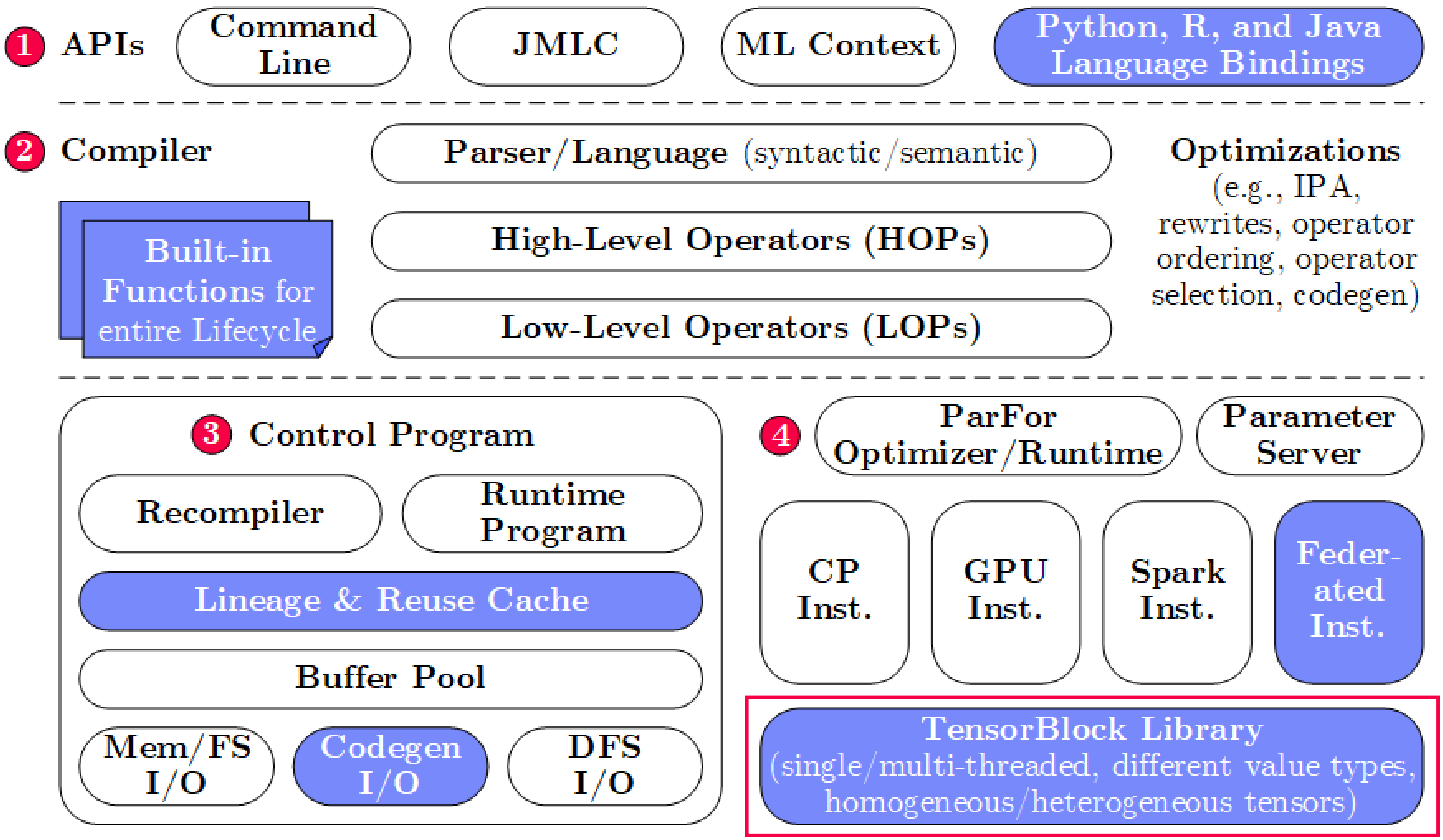
ML Algorithms

```
m_lmCG = function(...) {
  while( i<maxi&nr2>tgt ) {
    q = (t(X) %*% (X %*% p))
      + lambda * p
    beta = ... }
}
```

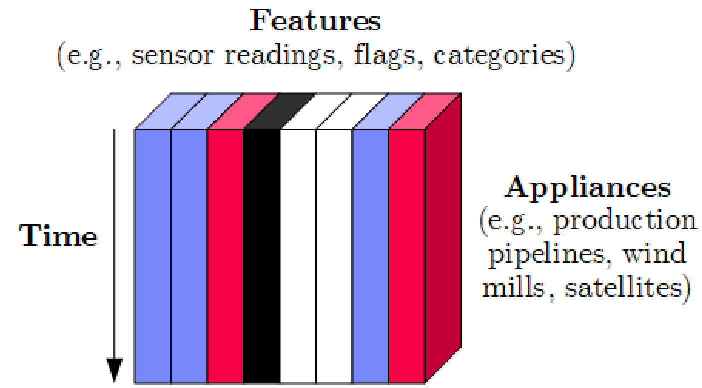
```
m_lmDS = function(...) {
  l = matrix(reg,ncol(X),1)
  A = t(X) %*% X + diag(l)
  b = t(X) %*% y
  beta = solve(A, b) ...}
```

Linear Algebra Programs

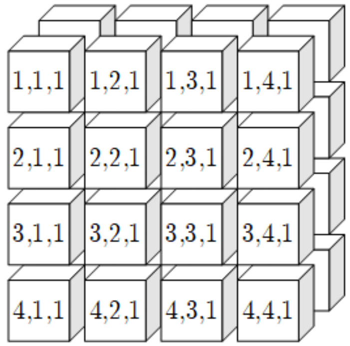
# System Architecture



# Data Model: Heterogeneous Tensors




- **Basic Tensor Block**
  - **BasicTensorBlock:** homogeneous tensors (FP32, FP64, INT32, INT64, BOOL, STRING/JSON)
  - **DataTensorBlock:** composed from basic TBs
  - Represents local tensor (CPU/GPU)




- **Distributed Tensor Representation**
  - Collection of **fix-sized tensor blocks**
  - Squared blocking schemes in n-dim space (e.g., 1024^2, 128^3, 32^4, 16^5, 8^6, 8^7)
  - PairRDD<TensorIndex,TensorBlock>
- **Federated Tensor Representation**
  - Collection of meta data handles to `TensorObjects`, each of which might refer to data on a different worker instance (local or distributed)
  - Generalizes to federated tensors of CPU and GPU data objects

# #1 Lineage and Reuse

- **Problem**
  - **Exploratory data science** (data preprocessing, model configurations)
  - **Reproducibility** and **explanability** of trained models (data, parameters, prep)
- **Lineage/Provenance as Key Enabling Technique**
  - Model versioning, reuse of intermediates, incremental maintenance, auto differentiation, and debugging (results and intermediates, convergence behavior via query processing over lineage traces)

- **a) Efficient 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

# #1 Lineage and Reuse, cont.

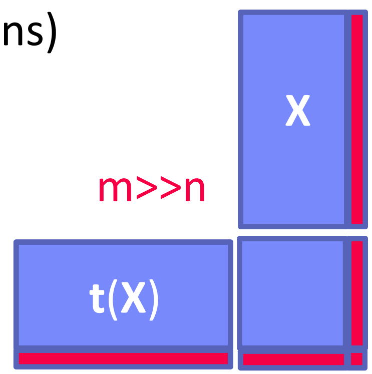
$O(k(mn^2+n^3)) \rightarrow O(mn^2+kn^3)$

- **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)

```
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)
} }
```

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

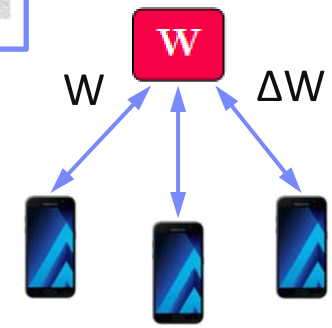
# #2 Data Integration and Cleaning

- **a) Semi-automated Data Preparation**
  - Provide abstractions for **composing data preparation pipelines** (built-in functions: **vectorized** & **pruning via sparsity exploitation**)
  - **ML-assisted data integration and cleaning** (extraction, schema alignment, entity linking, outlier detection, data augmentation, and feature transforms)
  - Design choice: retain **stateless appearance** (consume models as tensors)

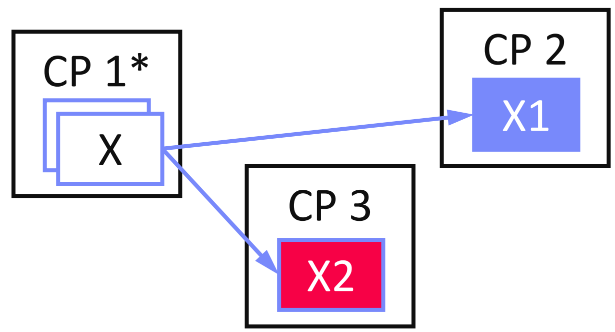
- **b) Efficient Data Ingestion**
  - **Codegen of efficient readers/writers** from high-level descriptions
  - **Avoid unnecessary parsing** on data extraction
  - **Avoid unnecessary shuffling** on distributed data preparation
  - Leverage lineage-based reuse and access methods for **LA over raw data**

# #3 Federated ML

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




- **Motivation Federated ML**
  - Learn model **w/o central data consolidation**
  - **Privacy + data/power caps** vs **personalization and sharing**
- **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 instructions and parameter server




- **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.

Bundesministerium Verkehr, Innovation und Technologie

FFG Forschung wirkt.

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

# #4 Compiler and Runtime

**Stack of declarative abstractions requires major extensions**

- **a) ML & Rules**
  - Complex ML apps often combine ML models and rules in meta model
  - Dedicated compilation and verification techniques
- **b) Size Propagation**
  - Better size propagation (dims, sparsity) over conditional control flow for cost-based optimization of complex pipelines
- **c) Operator Fusion & Code Generation**
  - Automatic operator fusion (composed ops) to avoid unnecessary intermediates, scan sharing, and sparsity exploitation across operations
- **d) Lossless and Lossy Compression**
  - Lossless matrix compression (CLA, TOC) and quantization for DNN workloads
  - Reconsideration for data tensors (n-dim, types) and federated ML
- **e) Cloud and Auto Scaling**
  - Resource optimization still an obstacle, especially for domain experts
  - Stateless design and size propagation simplifies auto scaling

# Conclusions

- **Summary: SystemML is dead, long live SystemDS**
  - **#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**, as well as hybrid parallelization strategies
  - **#3 Underlying data model of heterogeneous data tensors** w/ native support for lineage tracing, and automatic data reorganization and specialization

- **Next Lectures (Data Integration Architectures)**
  - **02 Data Warehousing, ETL, and SQL/OLAP** [Oct 11]
  - **03 Message-oriented Middleware, EAI, and Replication** [Oct 18]