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)

- **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
Data Management Courses

- **Architecture of Database Systems (ADBS, WS)**
- **Data Integration and Large-Scale Analysis (DIA, WS)**
- **Data Management / Databases (DM, SS+WS)**
- **Architecture of ML Systems (AMLS, SS)**

**Master**
- Distributed Data Management (usage and internals)

**Bachelor**
- DB system internals + prog. project
- ML system internals + prog. project in SystemDS
  [github.com/tugraz-isds/systemds]

Data management from user/application perspective

---

706.520 Data Integration and Large-Scale Analysis – 01 Introduction and Overview
Matthias Boehm, Graz University of Technology, WS 2019/20
Data Management Group

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**
  - 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

Data Scientist

Data Integration
- Data Cleaning
- Data Preparation

Model Selection
- Training
- Hyper-parameters

Validate & Debug
- Deployment
- Scoring & Feedback

Exploratory Process
(experimentation, refinements, ML pipelines)

Data/SW Engineer

DevOps Engineer
The 80% Argument

- **Data Sourcing Effort**
  - Data scientists spend 80-90% time on finding relevant datasets and data integration/cleaning.

- **Technical Debts in ML Systems**
  - 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 Motivation and Goals

#1 Information System Pyramid

Vertical Integration (e.g., ETL)

#2 Data Lake

Audio, Image, Video, Text, Streams, Logs

Distributed Data Stores

Catalogs

Distributed Computation Frameworks

Operational Systems

Material

ERP

CRM

eCommerce

Analytical Systems

DSS

DWH

Strategic Systems

Horizontal Integration (e.g., EAI)

Horizontal Integration (e.g., EAI)
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

- 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](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 data science lifecycle)

https://github.com/tugraz-isdts/systemds
What is an ML System?

ML Applications (entire KDD/DS lifecycle)

- Statistics
- Data Mining
- Machine Learning (ML)

- Classification
- Regression
- Recommenders
- Clustering
- Dim Reduction
- Neural Networks

Rapidly Evolving

Runtime Techniques (Execution, Data Access)

- Distributed Systems
- Data Management
- Operating Systems

- HW Architecture
- Accelerators

Compilation Techniques

HPC

Prog. Language Compilers
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)

- **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]
Example: Linear Regression Conjugate Gradient

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

1: \[ X = \text{read}\($1\); \# n \times m \text{ matrix} \]
2: \[ y = \text{read}\($2\); \# n \times 1 \text{ vector} \]
3: \[ \text{maxi} = 50; \lambda = 0.001; \]
4: \[ \text{intercept} = $3; \]
5: \[ \ldots \]
6: \[ r = -(t(X) \%**\% y); \]
7: \[ \text{norm}_r^2 = \text{sum}(r \times r); \, p = -r; \]
8: \[ w = \text{matrix}(0, \text{ncol}(X), 1); \, i = 0; \]
9: \[ \text{while}(i < \text{maxi} \& \& \text{norm}_r^2 > \text{norm}_r^2_{\text{trgt}}) \]
10: \{ 
11: \quad q = (t(X) \%**\% (X \%**\% p)) + \lambda p; 
12: \quad \alpha = \text{norm}_r^2 / \text{sum}(p \times q); 
13: \quad w = w + \alpha \times p; 
14: \quad \text{old}_r^2 = \text{norm}_r^2; 
15: \quad r = r + \alpha \times q; 
16: \quad \text{norm}_r^2 = \text{sum}(r \times r); 
17: \quad \beta = \text{norm}_r^2 / \text{old}_r^2; 
18: \quad p = -r + \beta \times p; \, i = i + 1; 
19: \} 
20: \text{write}(w, $4, \text{format}=\text{"text"});

Read matrices from HDFS/S3

Compute initial gradient

Compute conjugate gradient

Update model and residuals

Compute step size

→ “Separation of Concerns”
High-Level SystemML Architecture

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

- **DML Scripts**
  - Language
  - Compiler
  - Runtime

- **In-Memory Single Node** (scale-up)
- **Hadoop or Spark Cluster** (scale-out)

- In-Progess: GPU
  - since 2014/16

- Apache SystemML™
  - 05/2017 Apache Top-Level Project
  - 11/2015 Apache Incubator Project
  - 08/2015 Open Source Release
LinregDS (Direct Solve)

\[
X = \text{read}(1); \\
y = \text{read}(2); \\
\text{intercept} = 3; \\
\text{lambda} = 0.001; \\
\ldots \\
\text{if} (\text{intercept} == 1) \{ \\
\quad \text{ones} = \text{matrix}(1, \text{nrow}(X), 1); \\
\quad X = \text{append}(X, \text{ones}); \\
\}
\]

\[
I = \text{matrix}(1, \text{ncol}(X), 1); \\
A = \text{t}(X) \times X + \text{diag}(I) \times \text{lambda}; \\
b = \text{t}(X) \times y; \\
\text{beta} = \text{solve}(A, b); \\
\ldots \\
\text{write}(\text{beta}, 4);
\]

→ 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

Apache SystemML Background

Basic HOP and LOP DAG Compilation

HOP DAG
(after rewrites)

LOP DAG
(after rewrites)

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

---

**SystemDS Architecture**

---

706.520 Data Integration and Large-Scale Analysis – 01 Introduction and Overview
Matthias Boehm, Graz University of Technology, WS 2019/20
Language Abstractions and APIs, cont.

- Example: Stepwise Linear Regression

**User Script**

```r
X = read('features.csv')
Y = read('labels.csv')
[B,S] = steplm(X, Y, icpt=0, reg=0.001)
write(B, 'model.txt')
```

**Built-in Functions**

- **m_steplm** = function(...)
  ```r
  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)
  }
  ```

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

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

**SystemDS Architecture**

Facilitates optimization across data science lifecycle tasks
System Architecture

1. APIs
   - Command Line
   - JMLC
   - ML Context
   - Python, R, and Java Language Bindings

2. Compiler
   - Built-in Functions for entire Lifecycle
   - Parser/Language (syntactic/semantic)
   - High-Level Operators (HOPs)
   - Low-Level Operators (LOPs)
   - Optimizations (e.g., IPA, rewrites, operator ordering, operator selection, codegen)

3. Control Program
   - Recompiler
   - Runtime Program
   - Lineage & Reuse Cache
   - Buffer Pool
   - Mem/FS I/O
   - Codegen I/O
   - DFS I/O

4. ParFor Optimizer/Runtime
   - CP Inst.
   - GPU Inst.
   - Spark Inst.
   - Federated Inst.

TensorBlock Library
(single/multi-threaded, different value types, homogeneous/heterogeneous tensors)
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:
    \[ X = \text{eval}(\text{deserialize}(\text{serialize}(	ext{lineage}(X)))) \]

- **Proactive deduplication** of lineage traces for loops
#1 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)

- **c) Partial Reuse of Intermediates**
  - **Problem:** Often partial result overlap
  - Reuse partial results via dedicated rewrites (compensation plans)
  - Example: steplm

\[
\text{O}(k(mn^2+n^3)) \rightarrow \text{O}(mn^2+kn^3)
\]
#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

## Motivation Federated ML
- Learn model \textit{w/o central data consolidation}
- Privacy + data/power caps vs personalization and sharing

## Data Ownership $\Rightarrow$ 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)
- \textbf{Basic approach:} Federated ML + ML over raw data
- System infra, integration, data org & reuse, Exp DB, geo-dist.

#4 Compiler and Runtime

- **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]