TECHNISCHE UNIVERSITÄT BERLIN

Seminar Large-scale Data Engineering (LDE)
01 Structure of Scientific Papers

Dr.-Ing. Patrick Damme
Technische Universität Berlin
Berlin Institute for the Foundations of Learning and Data
Big Data Engineering (DAMS Lab)

Last update: Apr 16, 2023

[**Credit:** Based on "Introduction to Scientific Writing"/ "01 Structure of Scientific Papers" by Matthias Boehm (TU Graz, winter 2021/22)]

BIFOLD

Announcements/Org

- **Hybrid Setting with Optional Attendance**
  - In-person in TEL 811 (~20 seats)
  - Virtual via zoom
    https://tu-berlin.zoom.us/j/67376691490?pwd=NmlvWTM5VUVWRjU0UGI2bXhBVkxzQT09

Agenda

- FG Big Data Engineering (DAMS Lab)
- Course Organization, Outline, and Deliverables
- Structure of Scientific Papers
- List of Seminar Topics (Proposals)

FG Big Data Engineering (DAMS Lab)

https://www.tu.berlin/dams

FG Big Data Engineering (DAMS Lab) – Team

- **Head of the Group**
  Matthias Boehm

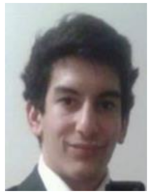
- **Postdoc** (01/2021)
  Patrick Damme

- **PhD Student** (04/2019)
  Arnab Phani

- **PhD Student** (01/2020)
  Sebastian Baunsgaard

- **PhD Student** (06/2023)
  Christina Dionysio

- **PhD Student** (06/2023)
  Philipp Ortner



- **PhD Student** (03/2023)
  David Justen
  SAP

- **PhD Student** (02/2023)
  Carlos E. Muniz Cuza
  [visitor Aalborg University]

- **PhD Student** (09/2019)
  Shafaq Siddiqi
  TU Graz

- **PhD Student** (04/2021)
  Saeed Fathollahzadeh
  TU Graz

- **3x Student Assistants**
  N.N. (1x ~06/2023)

- **Bachelor & Master Students**

BIFOLD

About Me

- **Since 10/2022: Postdoc at TU Berlin**, Germany
  - FG Big Data Engineering (DAMS Lab) headed by Prof. Matthias Böhm
  - Continuing work on integrated data analysis pipelines
  - Research interests in the fields of database and ML systems (especially compiler & runtime techniques, extensibility)
- **2021-2022: Postdoc at TU Graz & Know-Center GmbH**, Austria
  - Data Management group headed by Prof. Matthias Böhm
  - Started work on integrated data analysis pipelines
- **2015-2020: PhD student at TU Dresden**, Germany
  - Dresden Database Research Group headed by Prof. Wolfgang Lehner
  - PhD thesis on making complex analytical database queries more efficient through lightweight compression of intermediate results

MorphStore & TVL

FG Big Data Engineering (DAMS Lab) – Teaching

Architecture of Database Systems (ADBS, **WS**)
DB system internals
+ prog. project

Architecture of ML Systems (AMLS, **SoSe**)
ML system internals, data science lifecycle
+ prog. project

Master

Bachelor

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

Distributed Data Management

SE/PR Large-scale Data Engineering (LDE, **SoSe+WiSe**)

Intro to Scientific Methods (WS)

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

Data management from user/application perspective

Supervision of bachelor/master theses
(currently 3+12)

BIFOLD

Course Organization, Outline, and Deliverables

Large-scale Data Engineering: Module Overview

20+5 places in total

bachelor + master

#41086: LDE Seminar + Project (12 ECTS) — 13–2 students

12 students — #41095: Seminar LDE (3)

#41094: Project LDE (9 ECTS) — 12+2 students

bachelor-only

bachelor-only

Mon, 14:00-16:00
TEL 811 & zoom

Seminar LDE
- Reading & writing scientific papers
- Giving presentations on papers
- Summary paper
- Presentation
- Lecturer & seminar mentor

Project LDE
- Building & evaluating prototypes
- Giving presentations on prototypes
- Prototype design/impl/tests/doc
- Presentation
- Project mentors




Mon, 16:00-18:00
TEL 811 & zoom

→ In the context of systems for data engineering, data management, machine learning

→ In combination: Ideal preparation for a bachelor/master thesis with our group

Course Organization

- **General Contact Person**
  - Dr.-Ing. Patrick Damme (patrick.damme@tu-berlin.de)

- **Course Website**
  - https://pdamme.github.io/teaching/2023_summer/lde/lde_summer2023.html
  - One site for seminar and project
  - All material, news, **deadlines**, …

- **Language**
  - Lectures and slides: **English**
  - Communication: **English**/**German**
  - Submitted paper and presentation: **English**
  - **Informal language** (first name is fine), immediate feedback is welcome

Semester Schedule & Deadlines

- **Three Introductory Lectures** (optional)
  - Apr 17: Structure of Scientific Papers
  - Apr 24: Scientific Reading and Writing
  - May 8: Experiments, Reproducibility, and Giving Presentations
- **Self-organized Seminar Work**
  - Office hours for any questions (optional)
- **Final Presentations** (mandatory)
  - Jun 26: Session #1
  - Jul 03 : Session #2
  - Jul 10 : Session #3
  - Jul 17 : Session #4

- **List of Seminar Topics**
  - Presented today, take your time to select afterwards
- **Topic Selection**
  - **Deadline: May 1, 23:59 CEST** (in 2 weeks)
  - **First-come-first-serve**
  - List of preferences by email to Patrick Damme
- **Submission of Summary Paper**
  - **Deadline: Jun 19, 23:59 CEST** (in 9 weeks)
  - Summary paper (PDF) by email to Patrick Damme
- **Submission of Presentation Slides**
  - **Deadline: The day before you present, 23:59 CEST**
  - Presentation slides (PDF) by email to Patrick Damme

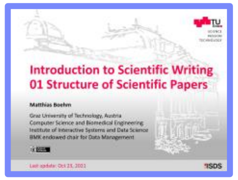
Seminar Deliverables

- **Individual Seminar Work**
  - 1 student = 1 paper, no team work
- **Summary Paper**
  - Read and understand selected paper
  - Search for related work for some context
  - Write summary paper (**6 pages**, excl. references)
    - including related work
    - make sure relation to umbrella is conveyed
  - LaTeX with **ACM acmart template** (**document class sigconf**, link on course website)
- **Presentation**
  - Summarize your paper
  - **15 min talk + 5 min discussion** (stay in time)
  - Audience: engage in the discussion

- **Grading**
  - Graded portfolio exam
  - **#41086 (seminar + project)**
    - 25 pts: summary paper
    - 15 pts: presentation
    - 50 pts: design/impl/tests/doc
    - 10 pts: presentation
  - **#41095 (seminar-only)**
    - 65 pts: summary paper
    - 35 pts: presentation

Structure of Scientific Papers

In Computer Science (Data Management)



[**Credit:** Based on “Introduction to Scientific Writing”/ ”01 Structure of Scientific Papers” by Matthias Boehm (TU Graz, winter 2021/22)]

Overview Types of Scientific Writing

- **Classification of Scientific/Technical Documents**
  - Formal vs informal writing, cumulative?, single vs multi-author, archival vs non-archival publications
- **Scientific Writing Skills are crucial**
  - Different types of docs share many similarities

Monographs
- BS Thesis
- MS Thesis
- PhD Thesis
- Habilitation

Scientific Papers
- by pub. channel
  - Preprint
  - Conference
  - Journal
  - Tech Report
- by contents
  - Research
  - Industry
  - Survey
  - E&A
  - Vision

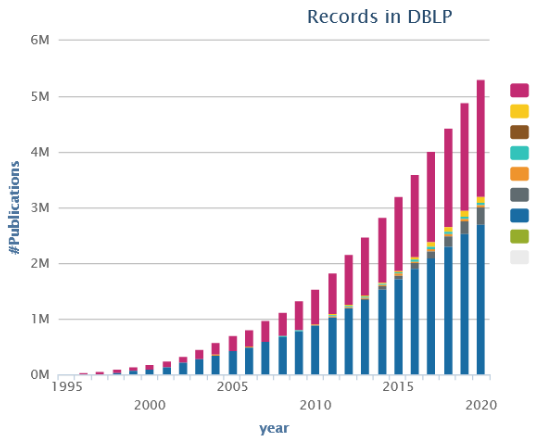
Others
- Grant Proposal
- Design doc
- White Paper
- Blog
- Website

Preparation

- **Know your Audience**

- **Get your Workflow in Order**
  - Writing: **LaTeX** (e.g., Overleaf, TeXnicCenter), **versioning** (e.g., git), **templates**
  - Plotting: **R** (e.g., plot, ggplot), **Python** (e.g., matplotlib, seaborn), **Gnuplot**, **LaTeX** (e.g., pgfplots)
  - Figures: e.g., MS Visio/**MS Powerpoint**, **Inkscape** → pdf, eps, svg (vector graphics)

- **Mindset: Quality over Quantity**
  - Aim for top-tier conferences/journals (act as filter)
  - Make the paper useful for others (ideas, evidence, code)



Paper Writing and Publication Process

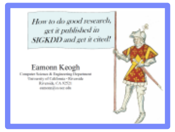
- **Research – Writing Cycle**
  - Read lots of papers
  - ~~Idea → Research → Writing → Document~~
  - Idea → Writing/Research → Document
  - Incremental refinement of drafts

- **Paper Submission Cycle**
  - Blind vs double-blind submission
  - Revisions and Camera-ready
  - **Similar: bachelor/master** thesis
    → drafts to advisor / final version

- **Example: SIGMOD 2024: Paper Submission Round 2**
  - **April 15, 2023**: Paper submission
  - **May 26 - 28, 2023**: Author feedback phase
  - **June 20, 2023**: Notification of accept/reject/review again
  - **July 20, 2023**: Revised paper submission
  - **August 23, 2023**: Final notification of accept/reject
  - **tba, 2023**: Camera ready due

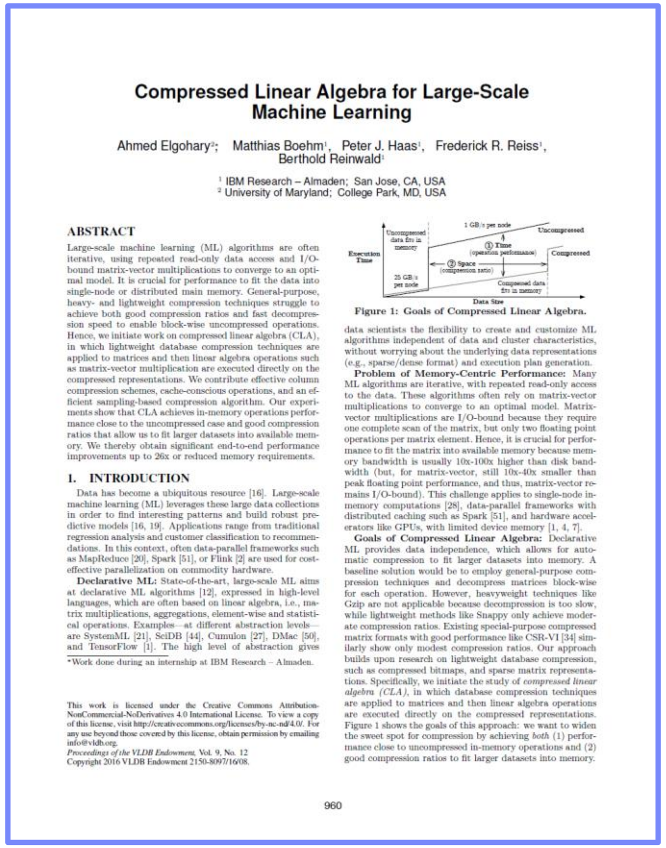
[Recommended Reading]

[Eamonn Keogh: How to do good research, get it published in SIGKDD and get it cited!, **KDD 2009**]



[Simon Peyton Jones: How to write a great research paper, MSR Cambridge]

Running Example



Compressed Linear Algebra for Large-Scale Machine Learning

Ahmed Elgohary[2]*, Matthias Boehm[1], Peter J. Haas[1], Frederick R. Reiss[1], Berthold Reinwald[1]

[1] IBM Research – Almaden; San Jose, CA, USA
[2] University of Maryland; College Park, MD, USA

ABSTRACT

Large-scale machine learning (ML) algorithms are often iterative, using repeated read-only data access and I/O-bound matrix-vector multiplications to converge to an optimal model. It is crucial for performance to fit the data into single-node or distributed main memory. General-purpose, heavy- and lightweight compression techniques struggle to achieve both good compression ratios and fast decompression speed to enable block-wise uncompressed operations. Hence, we initiate work on compressed linear algebra (CLA), in which lightweight database compression techniques are applied to matrices and then linear algebra operations such as matrix-vector multiplication are executed directly on the compressed representations. We contribute effective column compression schemes, cache-conscious operations, and an efficient sampling-based compression algorithm. Our experiments show that CLA achieves in-memory operations performance close to the uncompressed case and good compression ratios that allow us to fit larger datasets into available memory. We thereby obtain significant end-to-end performance improvements up to 26x or reduced memory requirements.

1. INTRODUCTION

Data has become a ubiquitous resource [16]. Large-scale machine learning (ML) leverages these large data collections in order to find interesting patterns and build robust predictive models [16, 19]. Applications range from traditional regression analysis and customer classification to recommendations. In this context, often data-parallel frameworks such as MapReduce [20], Spark [51], or Flink [2] are used for cost-effective parallelization on commodity hardware.

Declarative ML: State-of-the-art, large-scale ML aims at declarative ML algorithms [12], expressed in high-level languages, which are often based on linear algebra, i.e., matrix multiplications, aggregations, element-wise and statistical operations. Examples—at different abstraction levels—are SystemML [21], SciDB [44], Cumulon [27], DMac [50], and TensorFlow [1]. The high level of abstraction gives

*Work done during an internship at IBM Research – Almaden.

Proceedings of the VLDB Endowment, Vol. 9, No. 12

Figure 1: Goals of Compressed Linear Algebra.

data scientists the flexibility to create and customize ML algorithms independent of data and cluster characteristics, without worrying about the underlying data representations (e.g., sparse/dense format) and execution plan generation.

Problem of Memory-Centric Performance: Many ML algorithms are iterative, with repeated read-only access to the data. These algorithms often rely on matrix-vector multiplications to converge to an optimal model. Matrix-vector multiplications are I/O-bound because they require one complete scan of the matrix, but only two floating point operations per matrix element. Hence, it is crucial for performance to fit the matrix into available memory because memory bandwidth is usually 10x-100x higher than disk bandwidth (but, for matrix-vector, still 10x-40x smaller than peak floating point performance, and thus, matrix-vector remains I/O-bound). This challenge applies to single-node in-memory computations [28], data-parallel frameworks with distributed caching such as Spark [51], and hardware accelerators like GPUs, with limited device memory [1, 4, 7].

Goals of Compressed Linear Algebra: Declarative ML provides data independence, which allows for automatic compression to fit larger datasets into memory. A baseline solution would be to employ general-purpose compression techniques and decompress matrices block-wise for each operation. However, heavyweight techniques like Gzip are not applicable because decompression is too slow, while lightweight methods like Snappy only achieve moderate compression ratios. Existing special-purpose compressed matrix formats with good performance like CSR-VI [34] similarly show only modest compression ratios. Our approach builds upon research on lightweight database compression, such as compressed bitmaps, and sparse matrix representations. Specifically, we initiate the study of *compressed linear algebra (CLA)*, in which database compression techniques are applied to matrices and then linear algebra operations are executed directly on the compressed representations. Figure 1 shows the goals of this approach: we want to widen the sweet spot for compression by achieving *both* (1) performance close to uncompressed in-memory operations and (2) good compression ratios to fit larger datasets into memory.

960

- **Example paper used in the following**
  - Ahmed Elgohary, **Matthias Boehm**, Peter J. Haas, Frederick R. Reiss, Berthold Reinwald: **Compressed Linear Algebra** for Large-Scale Machine Learning. **PVLDB 2016**

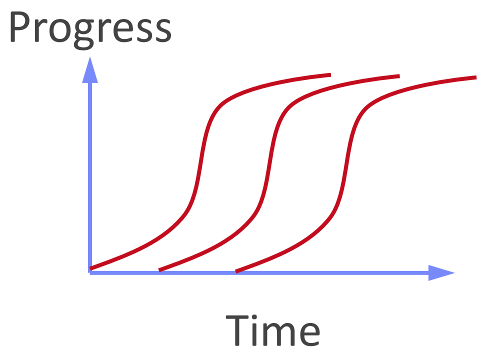
[Ahmed Elgohary, Matthias Boehm, Peter J. Haas, Frederick R. Reiss, Berthold Reinwald: Scaling Machine Learning via Compressed Linear Algebra. **SIGMOD Record 2017 46(1)**]

[Ahmed Elgohary, Matthias Boehm, Peter J. Haas, Frederick R. Reiss, Berthold Reinwald: Compressed Linear Algebra for Large-Scale Machine Learning. **VLDB Journal 2018 27(5)**]

[Ahmed Elgohary, Matthias Boehm, Peter J. Haas, Frederick R. Reiss, Berthold Reinwald: Compressed Linear Algebra for Large-Scale Machine Learning. **Commun. ACM 2019 62(5)**]

Ideas and Topic Selection

- **Problem-Oriented Research**
  - Focus on problem/observation first, not your solution
  - **Discuss early ideas** with collaborators and friends
  - Develop your taste for good research topics
  - Topic selection needs time → **pipeline model**



- **Ex. Compressed Linear Algebra**
  - **Problem:** Iterative ML algorithms + memory-bandwidth-bound operations → crucial to fit data in memory → automatic lossless compression
  - **Sub-problems:** #rows>>#cols, column correlation, column characteristics → column-wise compression w/ heterogeneous encoding formats

Prototypes and Experiments

- **Worst Mistake: Schrödinger's Results**
  - Postpone implementation and experiments till last before the deadline
  - No feedback, no reaction time (experiments require many iterations)
  - **Karl Popper:** falsifiability of scientific results
- **Continuous Experiments**
  - Run experiments during survey / prototype building
  - Systematic experiments → observations and ideas for improvements
  - Don’t be afraid of throw away prototypes that don’t work
- **Ex. Compressed Linear Algebra**
  - Data characteristics inspired overall design of encoding schemes
  - Initially slow compression → dedicated sampling schemes and estimators
  - Initially slow compressed operations → cache-conscious operations, selected operations with better asymptotic behavior

Prototypical Structure of a Scientific Paper

- **Title & Authors**
- **Sections and Subsections**
  - Abstract → short overview of problem and solution (part of meta data)
  - Introduction → context, problem, contributions
  - Background / Preliminaries → necessary background for understanding
  - Main Part → your technical core contributions
  - Main Part 2
  - Experiments → setting, micro benchmarks, end-to-end benchmarks
  - Related Work → areas of related work, differences to your own work
  - Conclusions → summary, conclusions, and future work
  - Acknowledgments → funding agencies, helpful people beyond co-authors
  - References → list of other works referenced throughout the paper
  - (Appendix) → any additional contents (e.g., proves of theorems, more results)

Title and Authors

- **List of Authors**
  - E.g., by contribution (main, …, advisor)
  - E.g., by last name
  - Affiliations, contact (corresponding author)

- **Title**
  - Descriptive yet concise
  - Short name if possible → easier to cite and discuss

Compressed Linear Algebra for Large-Scale Machine Learning

Ahmed Elgohary[2]; Matthias Boehm[1], Peter J. Haas[1], Frederick R. Reiss[1], Berthold Reinwald[1]

[1] IBM Research – Almaden; San Jose, CA, USA
[2] University of Maryland; College Park, MD, USA

SPOOF: Sum-Product Optimization and Operator Fusion for Large-Scale Machine Learning

Tarek Elgamal[2]; Shangyu Luo[3]; Matthias Boehm[1], Alexandre V. Evfimievski[1], Shirish Tatikonda[4]; Berthold Reinwald[1], Prithviraj Sen[1]

[1] IBM Research – Almaden; San Jose, CA, USA
[2] University of Illinois; Urbana-Champaign, IL, USA
[3] Rice University; Houston, TX, USA
[4] Target Corporation; Sunnyvale, CA, USA

MNC: Structure-Exploiting Sparsity Estimation for Matrix Expressions

Johanna Sommer, IBM Germany
Matthias Boehm, Graz University of Technology
Alexandre V. Evfimievski, IBM Research – Almaden
Berthold Reinwald, IBM Research – Almaden
Peter J. Haas, UMass Amherst

SliceLine: Fast, Linear-Algebra-based Slice Finding for ML Model Debugging

Svetlana Sagadeeva*, Graz University of Technology
Matthias Boehm, Graz University of Technology

[**Credit:** sliceline, Silicon Valley, HBO]

Abstract

```
% 1. State the problem
```

Large-scale machine learning (ML) algorithms are often iterative, using repeated read-only data access and I/O-bound matrix-vector multiplications to converge to an optimal model. It is crucial for performance to fit the data into single-node or distributed main memory.

```
% 2. Say why it's an interesting problem
```

General-purpose, heavy- and lightweight compression techniques struggle to achieve both good compression ratios and fast decompression speed to enable block-wise uncompressed operations.

```
% 3. Say what your solution achieves
```

Hence, we initiate work on compressed linear algebra (CLA), in which lightweight database compression techniques are applied to matrices and then linear algebra operations such as matrix-vector multiplication are executed directly on the compressed representations. We contribute effective column compression schemes, cache-conscious operations, and an efficient sampling-based compression algorithm. Our experiments show that CLA achieves in-memory operations performance close to the uncompressed case and good compression ratios that allow us to fit larger datasets into available memory.

```
% 4. Say what follows from your solution
```

We thereby obtain significant end-to-end performance improvements up to 26x or reduced memory requirements.

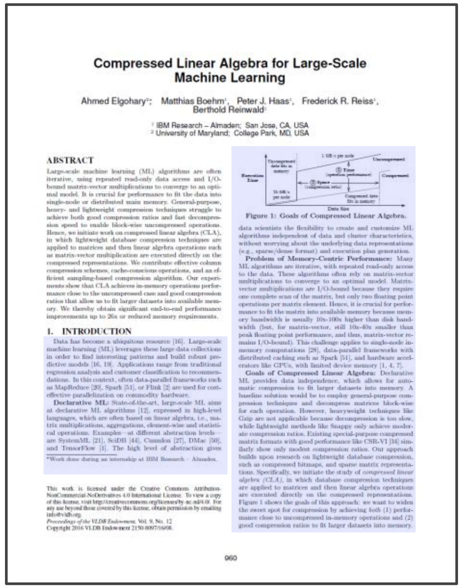
[Simon Peyton Jones: How to write a great research paper, MSR Cambridge]

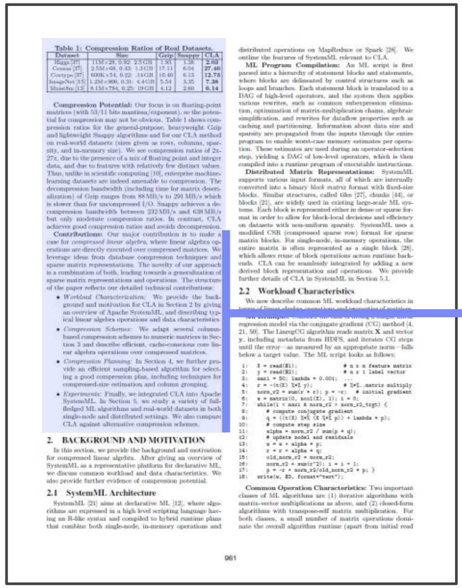
ABSTRACT

Large-scale machine learning (ML) algorithms are often iterative, using repeated read-only data access and I/O-bound matrix-vector multiplications to converge to an optimal model. It is crucial for performance to fit the data into single-node or distributed main memory. General-purpose, heavy- and lightweight compression techniques struggle to achieve both good compression ratios and fast decompression speed to enable block-wise uncompressed operations. Hence, we initiate work on compressed linear algebra (CLA), in which lightweight database compression techniques are applied to matrices and then linear algebra operations such as matrix-vector multiplication are executed directly on the compressed representations. We contribute effective column compression schemes, cache-conscious operations, and an efficient sampling-based compression algorithm. Our experiments show that CLA achieves in-memory operations performance close to the uncompressed case and good compression ratios that allow us to fit larger datasets into available memory. We thereby obtain significant end-to-end performance improvements up to 26x or reduced memory requirements.

Introduction

- **Prototypical Structure**
  - Context (1 paragraph)
  - Problems (1-3 paragraphs)
  - [Existing Work (1 paragraph)]
  - [Idea (1 paragraph)]
  - Contributions (1 paragraph)



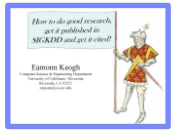


Contributions: Our major contribution is to make a case for *compressed linear algebra*, where linear algebra operations are directly executed over compressed matrices. We leverage ideas from database compression techniques and sparse matrix representations. The novelty of our approach is a combination of both, leading towards a generalization of sparse matrix representations and operations. The structure of the paper reflects our detailed technical contributions:

- *Workload Characterization:* We provide the background and motivation for CLA in Section 2 by giving an overview of Apache SystemML, and describing typical linear algebra operations and data characteristics.
- *Compression Schemes:* We adapt several column-based compression schemes to numeric matrices in Section 3 and describe efficient, cache-conscious core linear algebra operations over compressed matrices.
- *Compression Planning:* In Section 4, we further provide an efficient sampling-based algorithm for selecting a good compression plan, including techniques for compressed-size estimation and column grouping.
- *Experiments:* Finally, we integrated CLA into Apache SystemML. In Section 5, we study a variety of full-fledged ML algorithms and real-world datasets in both single-node and distributed settings. We also compare CLA against alternative compression schemes.

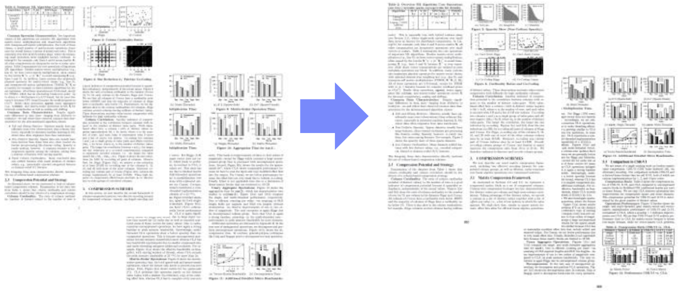
- **Introduction Matters**
  - **Anchoring**: most reviewers reach their opinion after reading introduction and motivation and then look for evidence

[Eamonn Keogh: How to do good research, get it published in SIGKDD and get it cited!, **KDD 2009**]



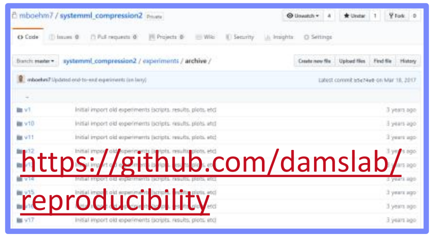
Writing the Paper (and more Experiments)

- **Easily Readable: Quality ∝ Time**
  - **Make it easy to skim the paper**
    → paragraph labels, self-explanatory figures (close to text), and structure
  - Avoid unnecessary formalism → as simple as possible
  - Shortening the text in favor of structure improves readability
  - **Ex. Compressed Linear Algebra**
    - Initial SIGMOD submission: **12+3 pages**
    - Final PVLDB submission: **12 pages**
      (+ more figures, experiments, etc.)



→ **02 Scientific Reading and Writing**

- **Solid, Reproducible Experiments**
  - Create, use, and share dedicated benchmarks / datasets
  - Avoid weak baselines, start early w/ baseline comparisons
  - Automate your experiments as much as possible
  - Keep repository of all scripts, results, and used parameters



https://github.com/damslab/reproducibility

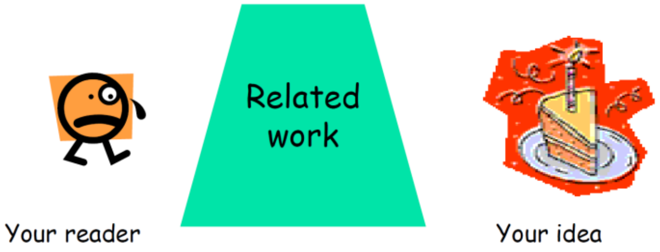
→ **03 Experiments, Reproducibility, and Giving Presentations**

Related Work

[Simon Peyton Jones: How to write a great research paper, MSR Cambridge]

- **Purpose of a “Related Work”-Section**
  - **Not** a mandatory task or to show you know the field
  - Put you work in context of related areas (~ 1 paragraph each)
  - Discuss closely related work
  - **Crisp separation from existing work** (what are the differences)

- **Placement**
  - Section 2 or **Section n-1**
  - Throughout the paper



- **Give Credit**
  - Cite broadly, **give credit to inspiring ideas**, create connections
  - Honestly acknowledge **limitations of your approach**

References

- **Setup**
  - Use LaTeX **\cite**{} and BibTeX
  - Use a consistent source of bibtex entries (e.g., DBLP)

```
inproceedings{StonebrakerBPR11,
  author    = {Michael {Stonebraker et al.}},
  title     = {{The Architecture of SciDB}},
  booktitle = {{SSDBM}},
  year      = {2011}
}
```

VLDB2016.bib

```
\bibliographystyle{abbrv}
\bibliography{VLDB2016}
```

- **Different References Styles**
  - But, **not in footnotes** (unless required)
  - 8. REFERENCES
    [1] M. Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *CoRR*, 2016.
    [2] A. Alexandrov et al. The Stratosphere Platform for Big Data Analytics. *VLDB J.*, 23(6), 2014.

References

[Al14] Alexandrov, A. et al.: The Stratosphere platform for big data J. 23/6, 2014.

[AS14] Arap, O.; Swany, M.: Offloading MPI Parallel Prefix Scan the NetFPGA. CoRR abs/1408.4939/, 2014.

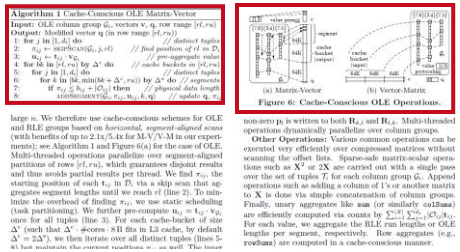
References

Jaume Amores. 2013. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*.

Dealing with Feedback / Criticism

- **Different Kinds of Feedback**
  - Casual discussion of early ideas
  - Comments on paper drafts
  - Reviewer comments (good and bad)

  - Always welcome feedback/criticism
  - Address all feedback w/ sincere effort
- **Example Compressed Linear Algebra**
  - SIGMOD Reviewer 2 (REJECT)
    - "The rewriting for q=Xv seems wrong: To compute q, one takes each row of the matrix X and multiplies it with the vector v."
  - PVLDB Reviewer 3 (WEAK ACCEPT)
    - "I kinda disagree with the broad definition of declarative ML […]."



Algorithm 1 Cache-Conscious OLE Matrix-Vector

Input: OLE column group $\mathcal{G}_i$, vectors $\mathbf{v}$, $\mathbf{q}$, row range $[rl, ru)$
Output: Modified vector $\mathbf{q}$ (in row range $[rl, ru)$)

```
1: for j in [1, d_i] do                              // distinct tuples
2:    π_ij ← SKIPSCAN(G_i, j, rl)                    // find position of rl in D_i
3:    u_ij ← t_ij · v_G_i                            // pre-aggregate value
4: for bk in [rl, ru) by Δ^c do                      // cache buckets in [rl, ru)
5:    for j in [1, d_i] do                           // distinct tuples
6:       for k in [bk, min(bk + Δ^c, ru)) by Δ^s do  // segments
7:          if π_ij ≤ b_ij + |O_ij| then             // physical data length
8:             ADDSEGMENT(G_i, π_ij, u_ij, k, q)     // update q, π_ij
```

large n. We therefore use cache-conscious schemes for OLE and RLE groups based on *horizontal, segment-aligned scans* (with benefits of up to 2.1x/5.4x for M-V/V-M in our experiments); see Algorithm 1 and Figure 6(a) for the case of OLE. Multi-threaded operations parallelize over segment-aligned partitions of rows $[rl, ru)$, which guarantees disjoint results and thus avoids partial results per thread. We find π_{ij}, the starting position of each $\mathbf{t}_{ij}$ in $\mathcal{D}_i$ via a skip scan that aggregates segment lengths until we reach rl (line 2). To minimize the overhead of finding π_{ij}, we use static scheduling (task partitioning). We further pre-compute $\mathbf{u}_{ij} = \mathbf{t}_{ij} \cdot \mathbf{v}_{\mathcal{G}_i}$ once for all tuples (line 3). For each cache-bucket of size Δ^c (such that $\Delta^c \cdot$ #cores $\cdot$ 8 B fits in L3 cache, by default $\Delta^c = 2\Delta^s$), we then iterate over all distinct tuples (lines 5-8) but maintain the current positions π_{ij} as well. The inner

Figure 6: Cache-Conscious OLE Operations.

non-zero $\mathbf{p}_l$ is written to both $\mathbf{R}_{k,l}$ and $\mathbf{R}_{l,k}$. Multi-threaded operations dynamically parallelize over column groups.

Other Operations: Various common operations can be executed very efficiently over compressed matrices without scanning the offset lists. Sparse-safe matrix-scalar operations such as $\mathbf{X}^2$ or $2\mathbf{X}$ are carried out with a single pass over the set of tuples $\mathcal{T}_i$ for each column group $\mathcal{G}_i$. Append operations such as adding a column of 1's or another matrix to $\mathbf{X}$ is done via simple concatenation of column groups. Finally, unary aggregates like `sum` (or similarly `colSums`) are efficiently computed via counts by $\sum_{i=1}^{|\mathcal{X}|} \sum_{j=1}^{d_i} |\mathcal{O}_{ij}| \mathbf{t}_{ij}$. For each value, we aggregate the RLE run lengths or OLE lengths per segment, respectively. Row aggregates (e.g., `rowSums`) are computed in a cache-conscious manner.

[Matthias Boehm et al.: Declarative Machine Learning - A Classification of Basic Properties and Types. **CoRR 2016**.]

- **Paper Rebuttal and/or Revision**
  - **Rebuttal**: seriously consider all feedback (in doubt agree), and answer with facts / ideas how to address the comments
  - **Revision** (conditional accept): address all revision requests

List of Seminar Topics (Proposals)

Last update: Apr 16, 2023

Seminar Umbrella Topic

- **Papers in the fields of systems for data engineering, data management, and machine learning**

- **This Semester's Umbrella Topic: "Extensible Data Systems"**
  - Motivation
    - Meet requirements of emerging applications (multimedia, machine learning, domain-specific, …)
    - Enable the timely adoption of novel techniques and technologies (algorithms, hardware, …)
    - Handle the increasing specialization (data models, data representations, algorithms, hardware, …)
    - Facilitate research at systems and algorithms level
  - Active field of research for decades (at least 1980s till today)
  - Interesting challenges
    - Which abstractions are needed?
    - How to enable users to extend the system?
    - How to reach efficiency in spite of these abstractions?
  - Concepts have been proposed at all levels of the system stack

Seminar Topics (1/4)

- **Extensible Database Management Systems**
  - Implementation of Data Abstraction in the Relational Database System Ingres (SIGMOD Rec, 1984)
  - The Design of POSTGRES (SIGMOD, 1986)
  - Design and Implementation of an Extensible Database Management System Supporting User Defined Data Types and Functions (VLDB, 1988)
  - Starburst Mid-Flight: As the Dust Clears (IEEE Trans. Knowl. Data Eng., 1990)
  - MIL Primitives for Querying a Fragmented World (VLDBJ, 1999)
  - Hyrise Re-engineered: An Extensible Database System for Research in Relational In-Memory Data Management (EDBT, 2019)

- **Extensible Machine Learning Systems**
  - TensorFlow: A System for Large-Scale Machine Learning (OSDI, 2016)
  - BAGUA: Scaling up Distributed Learning with System Relaxations (PVLDB, 2022)

Seminar Topics (2/4)

- **Component-based Database/ML Systems**
  - Flashlight: Enabling Innovation in Tools for Machine Learning (PMLR, 2022)
  - mutable: A Modern DBMS for Research and Fast Prototyping (CIDR, 2023)

- **Optimizer Generators / Compiler Frameworks**
  - GENESIS: An Extensible Database Management System (IEEE Trans. Software Eng., 1988)
  - The Volcano Optimizer Generator: Extensibility and Efficient Search (ICDE, 1993)
  - MLIR: Scaling Compiler Infrastructure for Domain Specific Computation (CGO, 2021)

- **Efficient Handling of User-defined Functions**
  - Opening the Black Boxes in Data Flow Optimization (PVLDB, 2012)
  - Extending database task schedulers for multi-threaded application code (SSDBM, 2015)
  - Processing Java UDFs in a C++ Environment (SOCC, 2017)

Seminar Topics (3/4)

- **Extensibility of Hardware Backends**
  - Voodoo - A Vector Algebra for Portable Database Performance on Modern Hardware (PVLDB, 2016)
  - TVM: An Automated End-to-End Optimizing Compiler for Deep Learning (OSDI, 2018)
  - Hardware-Oblivious SIMD Parallelism for In-Memory Column-Stores (CIDR, 2020)

Disclaimer:
Co-authored by the lecturer

- **Extensible Data Analytics**
  - Towards a unified architecture for in-RDBMS analytics (SIGMOD, 2012)
  - MLbase: A Distributed Machine-learning System (CIDR, 2013)

Seminar Topics (4/4)

- **Miscellaneous**
  - MorphStore: Analytical Query Engine with a Holistic Compression-Enabled Processing Model (PVLDB, 2020)
  - User-Defined Operators: Efficiently Integrating Custom Algorithms into Modern Databases (PVLDB, 2022)

Disclaimer:
Co-authored by the lecturer

- **Alternative: Propose yet another topic/paper**
  - Should be related to the umbrella topic of extensible data systems

Summary and Q&A

- FG Big Data Engineering (DAMS Lab)
- Course Organization, Outline, and Deliverables
- Structure of Scientific Papers
- List of Seminar Topics (Proposals)

- Remaining Questions?

- Next Lectures
  - 02 Scientific Reading and Writing [Apr 24]
  - 03 Experiments, Reproducibility, and Giving Presentations [May 8]