

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: Oct 11, 2026

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



Announcements/Org



- **Hybrid ~~Hybrid~~ Purely Virtual** Setting with Optional Attendance

- ~~In person in MAR 0.015~~
- Virtual via zoom

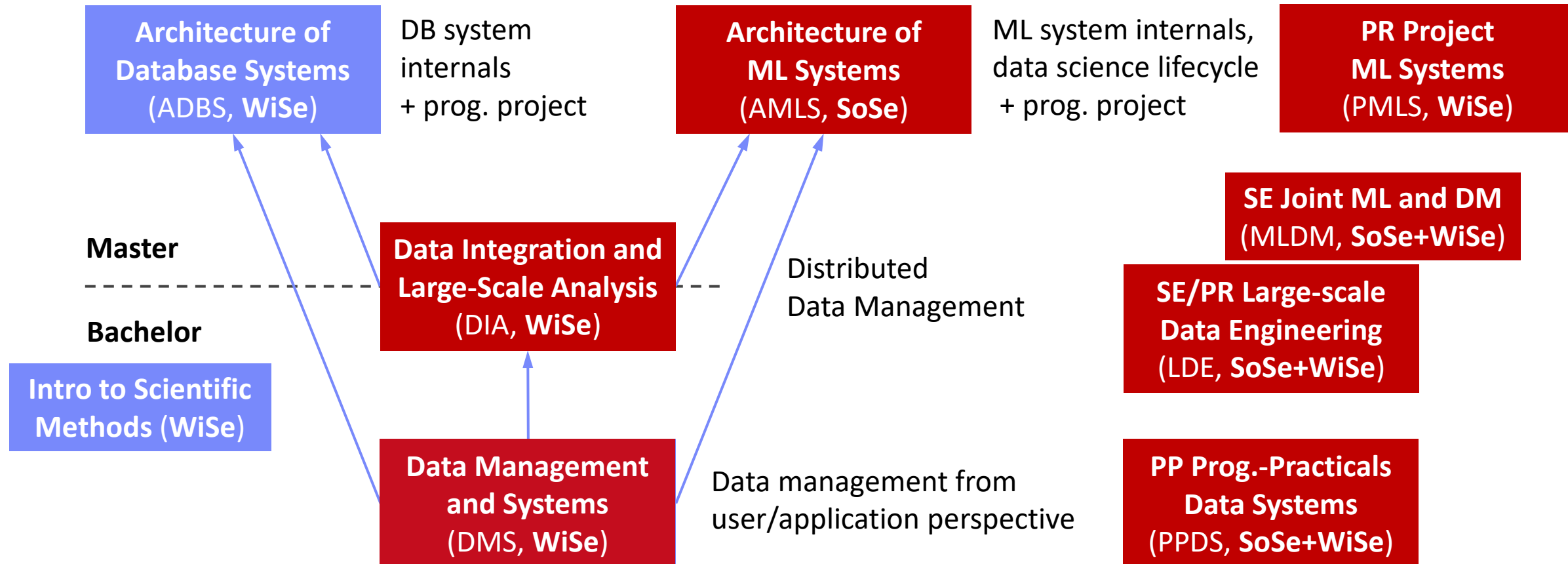
<https://tu-berlin.zoom-x.de/j/67376691490?pwd=NmlvWTM5VUVWRjU0UGI2bXhBVkxzQT09>



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, adaptive proc)
- **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





Agenda



- **Course Organization, Outline, and Deliverables**
- **Structure of Scientific Papers**
- **List of Seminar Topics**

Course Organization, Outline, and Deliverables

Large-scale Data Engineering: Module Overview



20+1 seats in total

master + bachelor

#41086: LDE Seminar + Project (12 ECTS)

10+1 students

10 students

#41095: Seminar LDE (3)

#41183: Project LDE (9 ECTS)

10 students

bachelor-only

bachelor-only

Mon, 14:00-16:00
MAR 0.015 & zoom

Seminar LDE

- Reading & writing scientific papers
- Giving presentations on papers
- Summary paper
- Presentation
- Lecturer & seminar mentors



Project LDE

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



Mon, 16:00-18:00
MAR 0.015 & 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

■ People

- **Lecturer** (+organizational matters): **Dr.-Ing. Patrick Damme** (patrick.damme@tu-berlin.de)
- **Teaching assistant** (+seminar-topic-related matters): **Dominic Stöcker, M.Sc.** (d.stoecker@campus.tu-berlin.de)

■ Course Website

- https://pdamme.github.io/teaching/2026-27_winter/lde/lde_winter2026-27.html
- One site for seminar and project; all material, schedule, **deadlines**

■ ISIS course

- <https://isis.tu-berlin.de/course/view.php?id=49696>
- Announcements, discussion forum, topic selection poll, submission of summary paper and presentation slides

■ 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)
 - Oct 12: Structure of Scientific Papers
 - Oct 19: Scientific Reading and Writing
 - Oct 26: Experiments, Reproducibility, and Giving Presentations
- **Self-organized Seminar Work**
 - Consultation hours for any questions (optional) (FR 768 and zoom)
- **Final Presentations** (mandatory, in-person)
 - Jan 25, 14:00-18:00: Session #1
 - Feb 01, 14:00-18:00: Session #2
- **List of Seminar Topics**
 - Presented today, take your time to select afterwards
- **Topic Selection**
 - **Deadline: Oct 23, 23:59** (in 1½ weeks)
 - Ranked list of **5 topics** via poll on the ISIS course
 - Global topic assignment based on preferences
 - **Notification of assigned topics: Oct 26** (in 2 weeks)
- **Submission of Summary Paper**
 - **Deadline: Jan 11, 23:59** (in 13 weeks)
 - Upload PDF in the ISIS course
- **Submission of Presentation Slides**
 - **Deadline: The day before you present, 23:59**
 - Upload PDF in the ISIS course

Portfolio Exam Registration



- **Registration: Nov 02 – Nov 30**
 - Binding registration in Moses/MTS
 - Including selection of seminar presentation date (first-come-first-serve)
- **De-registration**
 - **Until 3 days before the first graded exam part**
 - De-register yourself in Moses/MTS
 - Modules “LDE”/”Seminar LDE”: until **Jan 08**
 - Module “Project LDE”: until **Feb 18**
 - **With sufficient reason: Until the day of the exam**
 - In case of sickness etc.
 - Modules “LDE”/”Seminar LDE”: until **Jan 11/Jan 25/Feb 01**
 - Module “Project LDE”: until **Feb 21/Feb 22**
- **Missing Deadlines/Exam Without De-registration**
 - Zero points in the respective exam part (!)
 - **Approach us early in case of problems**
- **If You Don’t Want to Take LDE Anymore**
 - Let me know asap to give students in the queue a chance to fill in

- **Individual Seminar Work**
 - 1 student = 1 paper, no teamwork
- **Step 1: Understand the Topic**
 - Read and understand your selected paper
 - Read up on any unclear concepts
 - Search for related work to understand the context (at least 3-5 more papers, at least skim them)
- **Step 2: How Would You Explain the Topic?**
 - How would **you** structure the information?
 - Which helpful illustrations would **you** draw?
 - What key take-aways do **you** want to convey?
- **Step 3: Write the Summary Paper**
 - English, full text (sentences/paragraphs)
 - **4 pages** + unlimited references
- **FAQ**
 - **Title of the summary paper?**
=> Feel free to choose your own (e.g., “A Summary of XYZ”)
 - **Author(s) of the summary paper?**
=> Just you (not the original authors)
 - **Structure of the summary paper: Stick with the given paper’s?**
=> Use your own, but adhere to the prototypical paper structure
 - **Summary == copy the most important sentences?**
=> No, use your own words!
 - **Summary == slightly rephrase the text?**
=> No, present the topic in your own way!
 - **Writing perspective?**
=> “we” + “the authors”
 - **Where to reference the given paper?**
=> Reference after each sentence/paragraph usually unnecessary
=> Ensure that source of information is always clear, e.g., ref once at the beginning of the section
 - **Copying figures, tables, listings, etc. from the given paper?**
=> Create your own illustrations, tables, listings etc.
=> Exception (if necessary for accuracy): reuse plots (with ref)

Seminar Deliverables: Summary Paper: LaTeX Paper Template



■ Obtain the Template

- <https://www.acm.org/publications/proceedings-template>
- Download the ZIP archive `acmart-primary.zip` and unpack it

■ Select the Right Template

- The archive contains **all** ACM templates
- Use the **sigconf document class**

■ The Easiest Way to Set up Your Own Document

- Copy just the required files to a new directory
- Rename the .tex and .bib file as you like
(and adapt the `\bibliography{ }`)
- Open the .tex file in a text/LaTeX editor
- Remove example contents, replace it by yours
(title, authors, sections, paragraphs, figures, etc.)
- Replace the .bib contents by your BibTeX entries

```
acmart-primary/  
  samples/  
    sample-base.bib  
    sample-sigconf.tex  
  ...  
  acmart.cls  
  ACM-Reference-Format.bst  
  ...
```



```
lde-summary-paper/  
  literature.bib  
  summary-paper.tex  
  acmart.cls  
  ACM-Reference-Format.bst
```

Seminar Deliverables: Summary Paper: LaTeX Paper Template



ACM acmart template

document class **sigconf** (double-column)

CCS concepts

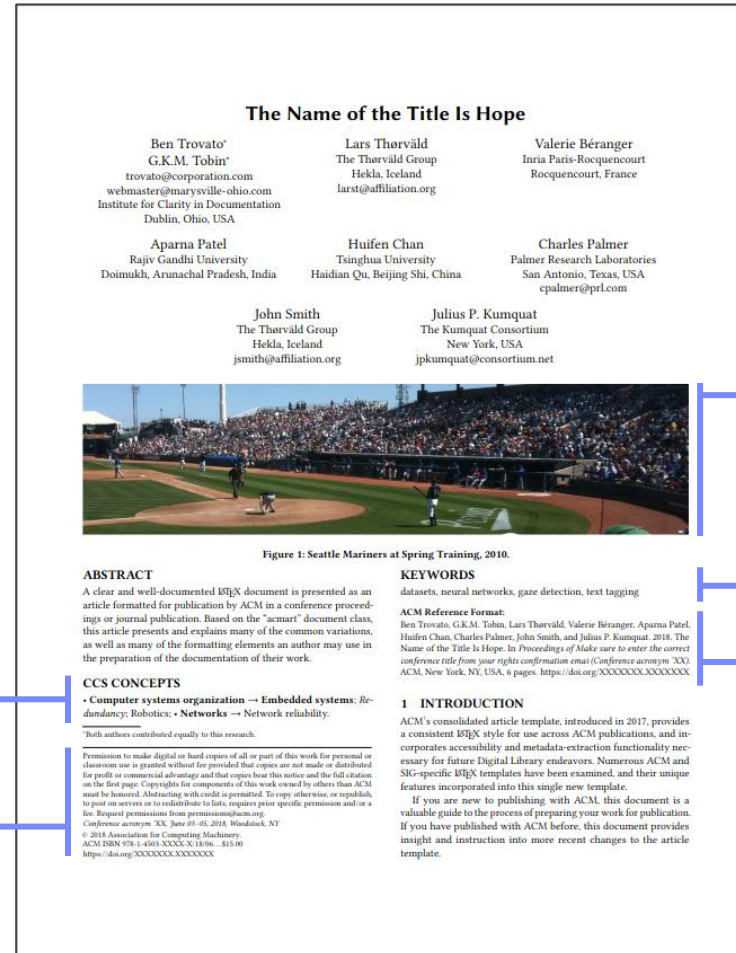
(ACM Computing Classification System)

Select concepts at <https://dl.acm.org/ccs>
and insert generated code:

```
220 %%  
221 %% The code below is generated by the tool at http://dl.acm.org/ccs.cfm.  
222 %% Please copy and paste the code instead of the example below.  
223 %%  
224 \begin{CCSXML}  
225 <ccs2012>  
226 <concept>  
227 <concept_id>10010520.10010553.10010562</concept_id>  
228 <concept_desc>Computer systems organization-Embedded systems</concept_desc>  
229 <concept_significance>500</concept_significance>  
230 </concept>  
231 <concept>  
232 <concept_id>10010520.10010575.10010755</concept_id>  
233 <concept_desc>Computer systems organization-Redundancy</concept_desc>  
234 <concept_significance>300</concept_significance>  
235 </concept>  
236 </CCSXML>
```

Copyright notice

Just keep as it is (ignore the dummy data)



Teaser image

Not required (especially no photograph)

Keywords

Specify meaningful keywords

```
255 %% Keywords. The author(s) should pick words that accurately describe  
256 %% the work being presented. Separate the keywords with commas.  
257 \keywords{datasets, neural networks, gaze detection, text tagging}
```

ACM reference format

Just keep as it is (ignore the dummy data)

Seminar Deliverables: Presentation & Grading



■ Presentation

- **Speaker:** Explain the topic to your fellow students, such that **they** can **understand** it
- Create appropriate slide deck
- Clear motivation, background, approach, experiments, conclusions, etc.
- **15 min talk + 5 min discussion** (stay in time)
- **Audience:** **engage in the discussion**

■ Academic Honesty / No Plagiarism

- Develop **your own** reading/writing/pres skills
- Use of **generative AI** (ChatGPT etc.) **is prohibited**

■ Graded Portfolio Exam

■ #41086 (seminar + project)

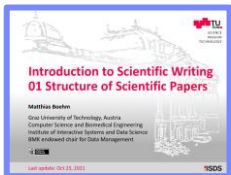
- 25 pts: summary paper
- 15 pts: presentation
- 50 pts: design/impl/tests/doc/exps
- 10 pts: presentation

■ #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

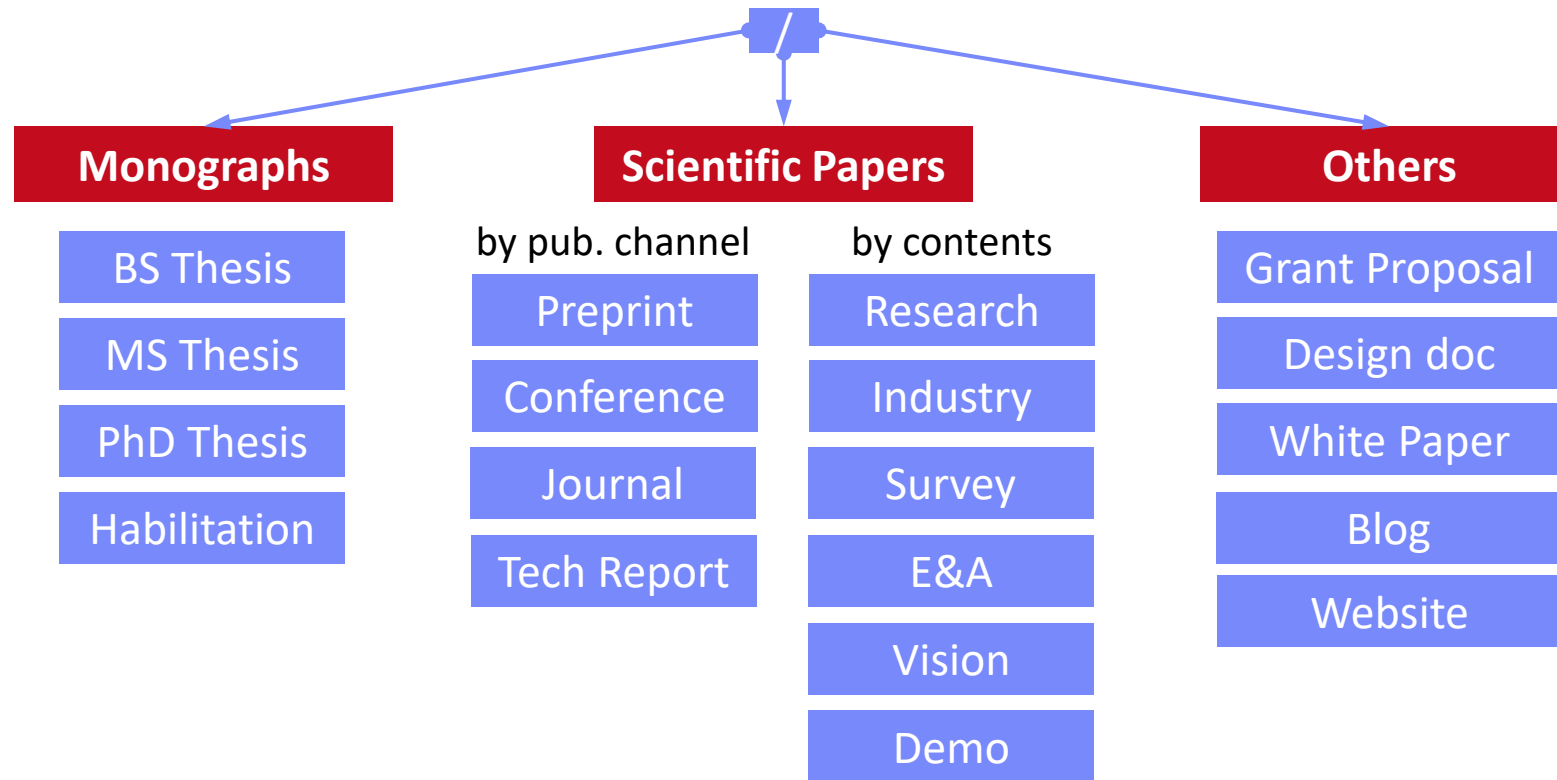


- **Classification of Scientific/Technical Documents**

- Formal vs informal writing, cumulative?, single vs multi-author, archival vs non-archival publications

- **Scientific Reading/Writing Skills Are **Crucial****

- Different types of docs share many similarities



■ 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 2025: Paper Submission Round 4

- **Oct 10, 2024:** Abstract Submission & Declaration of COIs
- **Oct 17, 2024:** Paper Submission
- **Nov 28, 2024:** Notification of Accept/Review/Reject
- **Dec 05, 2024:** Submission of Revision Plan
- **Dec 19, 2024:** Revision Feedback
- Submission of Revised Paper
- **Jan 30, 2025:** Final Notification
- Submission of Camera-ready Version



[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]



Compressed Linear Algebra for Large-Scale Machine Learning

Ahmed Elgohary¹, Matthias Boehm¹, Peter J. Haas¹, Frederick R. Reiss¹,
Berthold Reinwald²

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 [29], 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, aggregation, element-wise and statistical operations. Examples – at different abstraction levels – are SystemML [21], SciDB [14], Cumulon [27], DMac [50], and TensorFlow [1]. The high level of abstraction gives

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

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org.
Proceedings of the VLDB Endowment, Vol. 9, No. 12
Copyright 2016 VLDB Endowment 2150-8019/16/08.

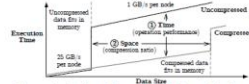


Figure 1: Goals of Compressed Linear Algebra.

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-V1 [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)**]

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 → 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)

Compressed Linear Algebra for Large-Scale Machine Learning

Ahmed Elgohary^{2*}, Matthias Boehm¹, Peter J. Haas¹, Frederick R. Reiss¹,
Berthold Reinwald¹

¹ IBM Research – Almaden; San Jose, CA, USA

² University of Maryland; College Park, MD, USA

■ Title

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

■ List of Authors

- E.g., by contribution (main, ..., advisor)
- E.g., by last name
- Affiliations, contact (corresponding author)

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

Tarek Elgamal^{2*}, Shangyu Luo^{3*}, Matthias Boehm¹, Alexandre V. Evfimievski¹,
Shirish Tatikonda⁴, Berthold Reinwald¹, Prithviraj Sen¹

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

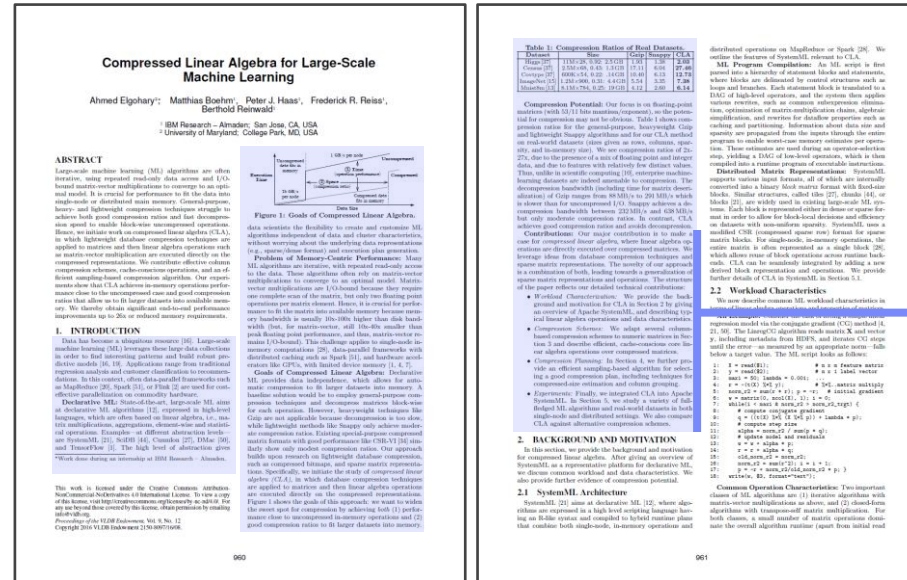
Large-scale machine learning (ML) algorithms are often iterative, using repeated read-only data access and I/O-bound matrix-vector multiplications to converge to an optimal model. It is crucial for performance to fit the data into single-node or distributed main memory. General-purpose, heavy- and lightweight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



■ 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]



■ Main Part: Core Technical Contributions

■ 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

→ 02 Scientific Reading and Writing

■ Solid, Reproducible Experiments

- Experimental setup and methodology
- Experimental results (visual presentation, interpretation/conclusions)
- Different kinds of experiments for different aspects of the problem

→ 03 Experiments, Reproducibility, and Giving Presentations

Related Work



■ Purpose of a “Related Work”-Section

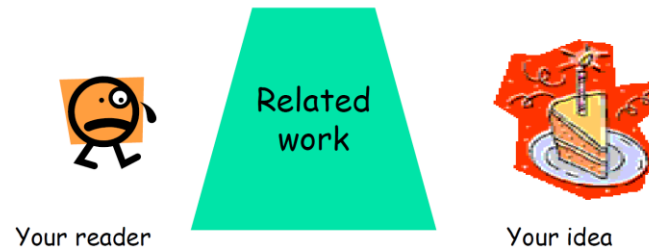
- **Not** a mandatory task or to show you know the field
- Put your work in context of related areas (~ 1 paragraph each)
- Discuss closely related work
- **Crisp separation from existing work** (what are the differences)

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



■ 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.

7. CONCLUSIONS

We have initiated work on compressed linear algebra (CLA), in which matrices are compressed with lightweight techniques and linear algebra operations are performed directly on the compressed representation. We introduced effective column encoding schemes, efficient operations over compressed matrices, and an efficient sampling-based compression algorithm. Our experiments show operations performance close to the uncompressed case and compression ratios similar to heavyweight formats like Gzip but better than lightweight formats like Snappy, providing significant performance benefits when data does not fit into memory. Thus, we have demonstrated the general feasibility of CLA, enabled by declarative ML that hides the underlying physical data representation. CLA generalizes sparse matrix representations, encoding both dense and sparse matrices in a universal compressed form. CLA is also broadly applicable to any system that provides block-matrix representations, linear algebra, and physical data independence. Interesting future work includes (1) full optimizer integration, (2) global planning and physical design tuning, (3) alternative compression schemes, and (4) operations beyond matrix-vector.

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.
- [3] A. Ashari et al. An Efficient Two-Dimensional Blocking Strategy for Sparse Matrix-Vector Multiplication on GPUs. In *ICIS (Int. Conf. on Supercomputing)*, 2014.
- [4] A. Ashari et al. On Optimizing Machine Learning Workloads via Kernel Fusion. In *PPoPP (Principles and Practice of Parallel Programming)*, 2015.
- [5] M. A. Bonnet, Data Compression in Scientific and Statistical Databases. *TSE (Trans. SW Eng.)*, 11(10), 1985.
- [6] W. Bell and M. Garland. Implementing Sparse Matrix-Vector Multiplication on Throughput-Oriented Processors. In *SC (Supercomputing Conf.)*, 2009.
- [7] J. Bergstra et al. Theano: a CPU and GPU Math Expression Compiler. In *SciPy*, 2010.
- [8] K. S. Beyer et al. On Synopses for Distinct-Value Estimation Under Matrix Operations. In *SIGMOD*, 2007.
- [9] B. Bhattacharjee et al. Efficient Index Compression in DR2. *VLDB*, 2(2), 2009.
- [10] S. Bhattacharjee et al. PBase: An Efficient Storage Framework for Managing Scientific Data. In *SSDDBM*, 2014.
- [11] C. Huang et al. Dictionary-based Order-preserving String Compression for Main Memory Column Stores. In *SIGMOD*, 2009.
- [12] M. Bollen et al. Declarative Machine Learning - A Classification of Basic Properties and Types. *CoRR*, 2016.
- [13] R. Barlow. The Infinite MNIST dataset. <http://data.dcc.utoronto.ca/projects/mnist/>.
- [14] M. Charlier et al. Towards Estimation Error Guarantees for Distinct Values. In *SIGMOD*, 2009.
- [15] R. Chittu et al. Approximate Kernel k-means: Solution to Large Scale Kernel Clustering. In *KDD*, 2011.
- [16] J. Cohen et al. MAD Skills: New Analysis Primitives for Big Data. *VLDB*, 2(2), 2009.
- [17] C. Constantinescu and M. Lu. Quick Estimation of Data Compression and De-compression for Large Storage Systems. In *CCP (Data Compression, Comm. and Process.)*, 2011.
- [18] C. V. Cormack. Data Compression on a Database System. *Commun. ACM*, 28(12), 1985.
- [19] S. Das et al. Riscv: Integrating R and Hadoop. In *SIGMOD*, 2010.
- [20] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*, 2004.

- [21] A. Ghosh et al. SystemML: Declarative Machine Learning on MapReduce. In *ICDE*, 2011.
- [22] I. J. Good. The Population Frequency of Species and the Estimation of Population Parameters. *Biometrika*, 1953.
- [23] G. Graefe and L. D. Shapiro. Data Compression and Database Performance. In *Applied Computing*, 1991.
- [24] P. J. Hsu and L. Shiao. Estimating the Number of Clones in a Finite Population. *J. Amer. Statist. Assoc.*, 93(444), 1998.
- [25] D. Hardek et al. Estimation of Deduplication Ratios in Large Data Sets. In *MSST (Mass Storage Sys. Tech.)*, 2012.
- [26] D. Hardek et al. To Zip or not to Zip: Effective Resource Usage for Real-Time Compression. In *PAST*, 2013.
- [27] B. Huang et al. Causal: Optimizing Statistical Data Analysis in the Cloud. In *SIGMOD*, 2013.
- [28] B. Huang et al. Resource Efficiency for Large-Scale Machine Learning. In *SIGMOD*, 2015.
- [29] S. Idroos et al. Estimating the Compression Fraction of an Index using Sampling. In *ICDE*, 2010.
- [30] N. L. Johnson et al. *Continuous Discrete Distributions*. Wiley, New York, 2nd edition, 1992.
- [31] V. Korth et al. An Extended Compression Format for the Optimization of Sparse Matrix-Vector Multiplication. *TPIOS (Trans. Par. and Dist. Systems)*, 24(10), 2013.
- [32] D. Kornet et al. SLACD - Sparse Linear Algebra in a Column-Oriented In-Memory Database System. In *SSDDBM*, 2014.
- [33] H. Kimura et al. Compression Aware Physical Database Design. *PVLDB*, 4(20), 2011.
- [34] K. Kourtis et al. Optimizing Sparse Matrix-Vector Multiplication Using Index and Value Compression. In *CF (Computing Frontiers)*, 2008.
- [35] H. Lang et al. Data Block: Hybrid OLTP and OLAP on Compressed Storage using both Vectorization and Compression. In *SIGMOD*, 2016.
- [36] P. Larson et al. SQL Server Column Store Indexes. In *SIGMOD*, 2011.
- [37] M. Leisner. UCI Machine Learning Repository. Higgs, Coopers, CS Census (1996). archive.ics.uci.edu/ml/.
- [38] P. E. O'Neill. Model 204 Architecture and Performance. In *High Performance Transaction Systems*, 1980.
- [39] Oracle. *Data Warehousing Guide, 11g Release 1*, 2007.
- [40] V. Ramana and G. Swart. How to Write a Table Driven: Entropy Compression of Relations and Querying of Compressed Relations. In *VLDB*, 2006.
- [41] V. Ramana et al. DR2 with B+ Acceleration: So Much More than Just a Column Store. *PVLDB*, 6(11), 2013.
- [42] Y. S. SPARKS. SPARKS: a basic tool kit for sparse matrix computations - Version 2, 1994.
- [43] M. Stonebraker et al. C-Store: A Column-oriented DBMS. In *VLDB*, 2005.
- [44] M. Stonebraker et al. The Architecture of SciDB. In *SSDDBM*, 2011.
- [45] Sysbase. *SQL 12.4 System Administration Guide*, 2013.
- [46] G. Valiant and P. Valiant. Estimating the Unseen: An In/Out-sample Estimator for Entropy and Support Size, Shows Optimal via New CLTs. In *STOC*, 2011.
- [47] T. Watson et al. The Implementation and Performance of Compressed Databases. *SIGMOD Record*, 29(3), 2000.
- [48] S. Williams et al. Optimization of Sparse Matrix-Vector Multiplication on Emerging Multiscale Platforms. In *SC (Supercomputing Conf.)*, 2007.
- [49] K. Wu et al. Optimizing Stream Indices With Efficient Compression. *TODS*, 31(1), 2006.
- [50] L. Ye et al. Exploring Matrix Dependency for Efficient Distributed Matrix Computation. In *SIGMOD*, 2015.
- [51] M. Zaharia et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In *KDD*, 2012.
- [52] C. Zhang et al. Materialized Optimizations for Feature Selection Workloads. In *SIGMOD*, 2014.

971

References

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

List of Seminar Topics

See list at https://pdamme.github.io/teaching/2026-27_winter/lde/SeminarTopics.pdf

Summary and Q&A



- Course Organization, Outline, and Deliverables
- Structure of Scientific Papers
- List of Seminar Topics (Topic Selection by Oct 23)
- Remaining Questions?
- Next Lectures
 - Hopefully back to hybrid setup...
 - 02 Scientific Reading and Writing [Oct 19]
 - 03 Experiments, Reproducibility, and Giving Presentations [Oct 26]