Seminar Large-scale Data Engineering (LDE)
03 Experiments, Reproducibility & Presentations

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

[**Credit:** Based on "Introduction to Scientific Writing"/ "03 Experiments and Reproducibility" by Matthias Boehm (TU Graz, winter 2021/22)]

Last update: May 4, 2025

Announcements/Org

- **Hybrid Setting with Optional Attendance**
  - In-person in MAR 0.010
  - Virtual via zoom
    https://tu-berlin.zoom-x.de/j/67376691490?pwd=NmlvWTM5VUVWRjU0UGI2bXhBVkxzQT09

- **Selection of Seminar and Project Topics**
  - Received participants' preferences
  - **Assignment ongoing**
    - **Seminar:** tba by May 5 (today)
    - **Project:** tba by May 12 (next Mon)

Agenda

- **Experiments and Result Presentation**
- **Reproducibility and RDM**
- **Scientific Presentations**

Experiments and Result Presentation

In Computer Science (Data Management)

[**Credit:** Ioana Manolescu, Stefan Manegold: Performance Evaluation in Database Research: Principles and Experiences, **ICDE 2008**]

Motivation

- **Worst Mistake: Schrödinger's Results**
  - Postpone implementation and experiments till last before the deadline
  - No feedback, no reaction time (experiments require many iterations)
- **Continuous Experiments**
  - Run experiments during survey / prototype building
  - Systematic experiments → observations and ideas for improvements
  - Don't be afraid of throwing away prototypes that don't work
- **Good Research Fires Itself**
  - Initial experiments give directions for further improvements
  - Problem-oriented methodology

Types of Experiments

- **#1 Exploratory Experiments**
  - Tests for functional correctness
  - Unstructured experiments for initial feedback → evaluate feasibility
- **#2 Micro Benchmarks**
  - Measure specific aspects in controlled and understandable scope
  - Bottom-up approach
- **#3 Benchmarks**
  - Evaluate on community/your own benchmarks
  - Examples: TPC-C, TPC-H, TPC-DS, JOB, MLPerf, TPCx-AI
- **#4 End-to-end Applications**
  - Evaluate in larger scope of real datasets and query workloads
  - Examples: Customer workload, ML pipelines (data preparation, training, evaluation)

From Idea to Experiments

[I. Manolescu, S. Manegold: Performance Evaluation in Database Research: Principles and Experiences, **ICDE 2008**]

- **Overview**
  - Proper planning helps to keep you from “getting lost”
  - Repeatable experiments simplify your own work
  - There is **no single way** how to **do it right**
  - There are **many ways** how to **do it wrong**

- **Basic Planning**
  - Which **data** / datasets should be used?
  - Which **workload** / queries should be run?
  - Which **baselines** are relevant?
  - Which **hardware** & **software** should be used?
  - **Metrics:** What to measure? How to measure?
  - **Comparison:** How to compare? How to find out what is going on?

Dataset Selection

- **Synthetic Data**
  - Generate data with **specific data characteristics**
  - Benchmarks with data generators (**inspired by real-world** use cases)
  - **Systematic evaluation** with data size, sparsity, distributions etc.
  - **Don't use too small data sizes** (should the data fit into CPU cache, main memory, single-node storage, cluster, etc.?)

Representative of real data distributions?

- **"Real" Data Repositories**
  - Wide selection of available **datasets with different characteristics**
  - UCI ML Repository: https://archive.ics.uci.edu/
  - SuiteSparse Matrix Collection: https://sparse.tamu.edu/
  - Google dataset search: https://datasetsearch.research.google.com/
  - Common Datasets in ML: ImageNet, Mnist, CIFAR, KDD, Criteo
  - Common Datasets in DM: Census, Taxi, Airlines, DBLP, benchmarks etc.

Representative for variety of workloads / common case?

Workloads: Benchmarks

- **Overview**
  - Remember: Types of experiments: (exploratory experiments), micro benchmarks, **benchmarks**, end-to-end applications

- **Benchmarks**
  - Community- and organization-driven creation of agreed benchmarks
  - **Benchmarks can define a field** and foster innovation
  - **#1 Data Management**
    - Query processing: 007, TPC-C, TPC-E, TPC-H, TPC-DS (w/ audit), SSB
    - Join ordering: JOB
  - **#2 "Big Data"**
    - MR/Spark: BigBench, HiBench, SparkBench
    - Array Databases: GenBase
  - **#3 Machine Learning Systems**
    - SLAB, DAWNBench, MLPerf, MLBench, AutoML Bench, Meta Worlds, TPCx-AI

[Michael J. Carey, David J. DeWitt, Jeffrey F. Naughton: The oo7 Benchmark. **SIGMOD 1993**]

[https://www.tpc.org/tpch/]

(See AMLS course for details)

Baselines

- **#1 Primary Baseline**
  - Existing algorithm or system infrastructure
  - Main comparison point, usually with same runtime operations
  - **Beware:** Avoid speedup-only results (need absolute numbers for grounding)
- **#2 Additional Baselines**
  - Alternative systems with different runtime and compiler
  - Usually, not directly comparable but important for grounding
  - E.g., SystemDS: R, Julia, Spark, TensorFlow, PyTorch, …
  - E.g., DAPHNE: the same, plus numpy, pandas, DuckDB, …
  - **Potential hindering factors:** commercial and closed-source systems, software licenses

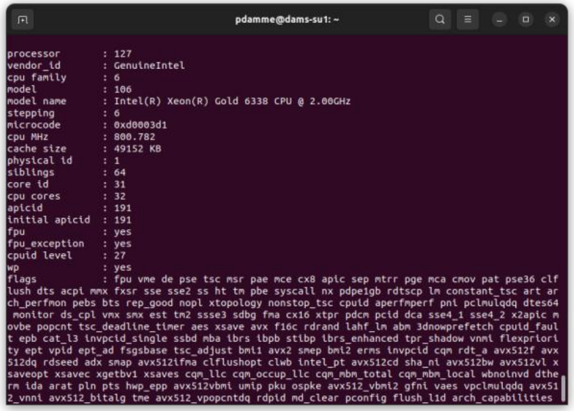
- **Problem of Weak Baselines**
  - Authors want to show improvements
  - Successive improvements over state-of-the-art don't add up

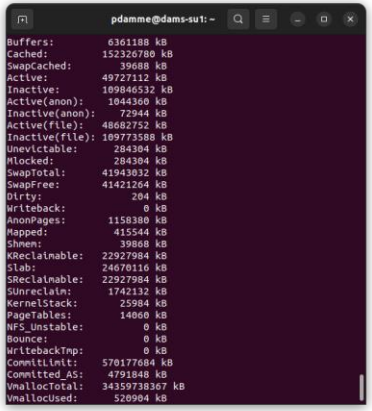
[Timothy G. Armstrong, Alistair Moffat, William Webber, Justin Zobel: Improvements That Don't Add Up: Ad-Hoc Retrieval Results Since 1998. **CIKM 2009**]

[Maurizio Ferrari Dacrema, Paolo Cremonesi, Dietmar Jannach: Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. **RecSys 2019**]

Experimental Setting (Hardware & Software)

- **Hardware Selection**
  - Multiple nodes for distributed computation
  - Avoid too outdated hardware (irrelevance)
- **Find Balanced Level of Detail**
  - Underspecified:
    `“We ran all experiments on an Intel CPU”`
  - Over-specified:
    `cat /proc/cpuinfo`
    `cat /proc/meminfo`





- **Recommendation**
  - **HW components:** #nodes, CPUs (#cores, clock freq., cache size), memory, network, I/O
  - **SW components:** OS, programming language, versions, other software, compiler flags
  - Baselines and configuration → Use **recent versions of baseline systems**
  - Data and workloads with data sizes, parameters, configurations

Metrics & Comparison

- **Typical Metrics**
  - Runtime (elapsed time) [e.g., ms]
  - Throughput [e.g., GB/s]
  - Memory/storage consumption [e.g., GB]
  - Performance counters [e.g., #L1d-cache misses]
  - Result quality, e.g.
    accuracy, recall/precision, various error metrics
  - **Useful metrics depend on goal of experiment**
- **How to Measure**
  - General tools: e.g., `time`, `perf`, `top`, `htop`, `iotop`
  - System-spefic tracing/statistics tools, e.g.
    - DAPHNE: `--timing`, `--statistics`
    - SystemDS: `-stats <count>`
- **Repeat Measurements & Calculate Mean/Median**
  - Reduce sensitivity to outliers
- **Ensure Fair Comparison**
  - Use **recent version** of all baselines
  - Use the right **compiler and prog language flags**
    (e.g., optimization levels: `g++ -O3`)
  - **Tune baselines** by hand to get the most out of them
    See their docs for the **args**, **config** and **tuning knobs**
- **Result Interpretation**
  - **Don't just report** the results
  - Try to **understand and explain** them

Metrics & Comparison: Measuring Time in DAPHNE

- **Example: k-Means Clustering on Small Random Dataset in DAPHNE**

script output

--timing

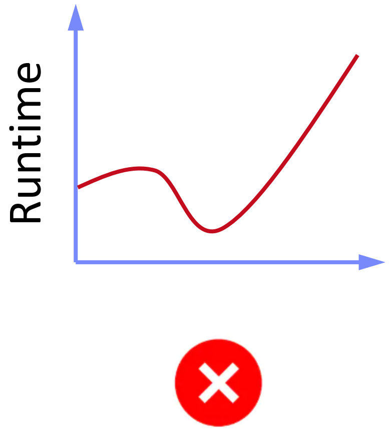
--statistics

time

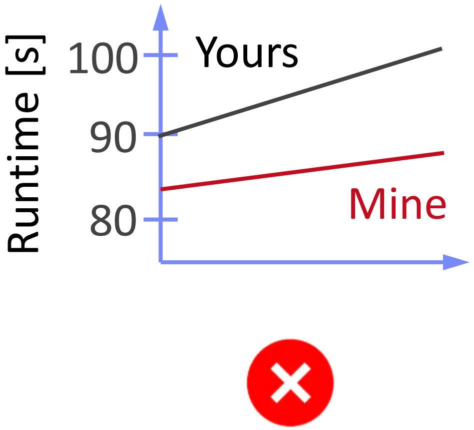
```
ubuntu@daphne-container:/daphne$ time bin/daphne --timing --statistics test/api/cli/algorithms/kmeans.daphne r=1000000 f=10 c=5 i=20
DenseMatrix(5x10, double)
0.513791 0.71707 0.477272 0.357556 0.482075 0.469744 0.477895 0.498584 0.70016 0.292175
0.510809 0.305214 0.487386 0.247159 0.498644 0.512785 0.514804 0.493601 0.557605 0.724351
0.505778 0.736792 0.494478 0.58006 0.490072 0.478354 0.478442 0.510815 0.274124 0.691537
0.488499 0.391096 0.538751 0.784757 0.531517 0.522385 0.516354 0.489182 0.727017 0.558672
0.482869 0.342111 0.502864 0.52036 0.49785 0.517561 0.51353 0.505505 0.253164 0.247353
{"startup_seconds": 0.0135434, "parsing_seconds": 0.00184906, "compilation_seconds": 0.0286434, "execution_seconds": 4.05648, "total_seconds": 4.10051}
[info]: DAPHNE operator execution runtime statistics.
[info]: #   Operator Name                                                  Time(s)  Count   Avg(s)  File:Line:Column
[info]: 0   _ewLe__DenseMatrix_double__DenseMatrix_double__DenseMatrix_double   0.72     20      0.04    test/api/cli/algorithms/kmeans.daphne:17:10
[info]: 1   _transpose__DenseMatrix_double__DenseMatrix_double                  0.63     20      0.03    test/api/cli/algorithms/kmeans.daphne:20:9
[info]: 2   _ewDiv__DenseMatrix_double__DenseMatrix_double__DenseMatrix_double  0.50     20      0.02    test/api/cli/algorithms/kmeans.daphne:18:10
[info]: 3   _ewAdd__DenseMatrix_double__DenseMatrix_double__DenseMatrix_double  0.50     20      0.02    test/api/cli/algorithms/kmeans.daphne:15:24
[info]: 4   _ewMul__DenseMatrix_double__DenseMatrix_double__double              0.46     20      0.02    test/api/cli/algorithms/kmeans.daphne:15:19
[info]: 5   _matMul__DenseMatrix_double__DenseMatrix_double__DenseMatrix_double__bool__bool  0.23  20  0.01  test/api/cli/algorithms/kmeans.daphne:15:11
[info]: 6   _matMul__DenseMatrix_double__DenseMatrix_double__DenseMatrix_double__bool__bool  0.20  20  0.01  test/api/cli/algorithms/kmeans.daphne:20:14
[info]: 7   _minRow__DenseMatrix_double__DenseMatrix_double                     0.17     20      0.01    test/api/cli/algorithms/kmeans.daphne:16:11
[info]: 8   _sumRow__DenseMatrix_double__DenseMatrix_double                     0.17     20      0.01    test/api/cli/algorithms/kmeans.daphne:18:12
[info]: 9   _sumCol__DenseMatrix_double__DenseMatrix_double                     0.15     20      0.01    test/api/cli/algorithms/kmeans.daphne:19:14
[info]: 10  _randMatrix__DenseMatrix_double__size_t__size_t__double__double__double__int64_t  0.11  1  0.11  test/api/cli/algorithms/kmeans.daphne:10:4

real    0m4.119s
user    0m20.831s
sys     0m40.832s
```

Presentation – Figures

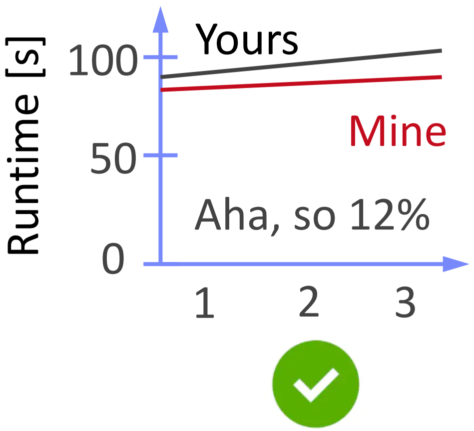
- **Axes**
  - Use informative axes labels with units (e.g., Total Execution Time [ms])
  - Don’t cheat or mislead readers and reviewers
  - Start y-axis at 0 for linear scale



What are the units?
Where are the tics?

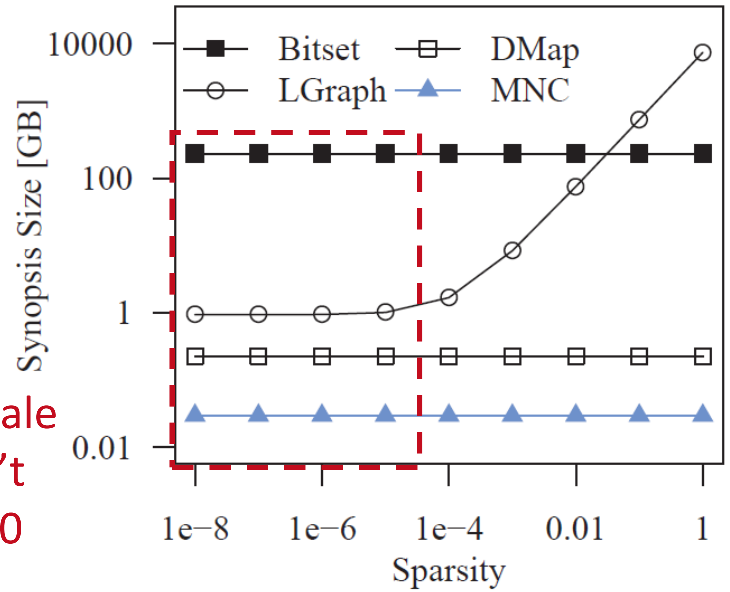


Misleading y axis



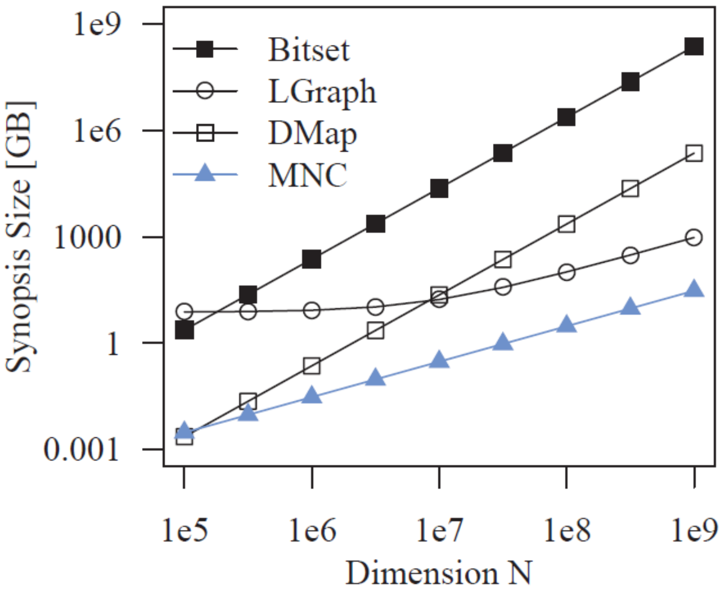
Presentation – Figures, cont.

- **Fair Ranges of Parameters**
  - Evaluate common ranges of values
  - Don’t hide important information



For log-scale you can’t start at 0

Don’t limit range to make you look good



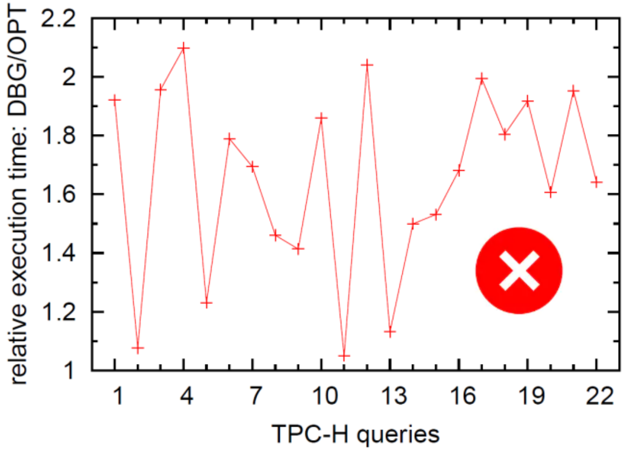
[J. Sommer, M. Boehm, A. V. Evfimievski, B. Reinwald, P. J. Haas: MNC: Structure-Exploiting Sparsity Estimation for Matrix Expressions. **SIGMOD 2019**]

If there are multiple relevant parameters, show them all

Presentation – Figures, cont.

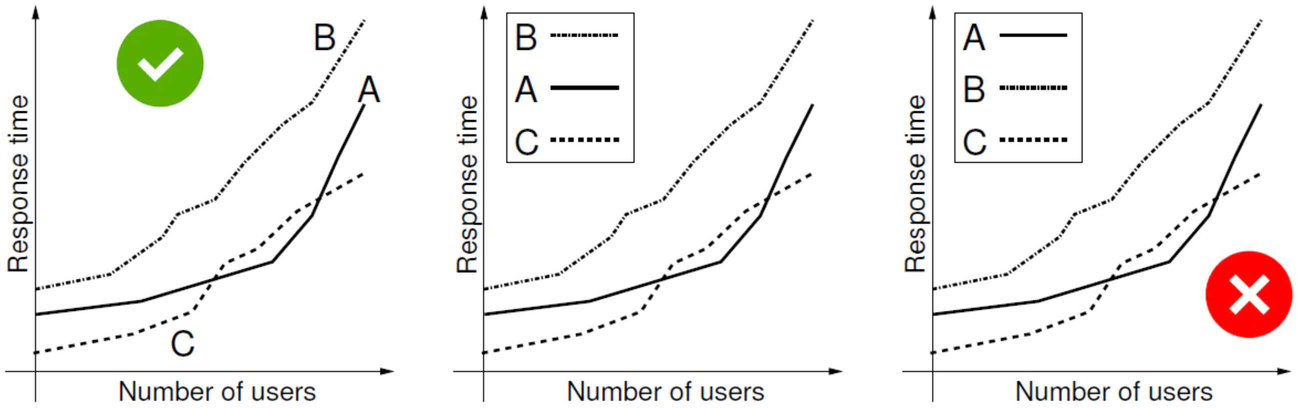
[I. Manolescu, S. Manegold: Performance Evaluation in Database Research: Principles and Experiences, **ICDE 2008**]

- **Plots Types**
  - **Barplot** for categories
  - **Plot** + Line/linepoints for continuous parameters
  - Visible font sizes (similar to text)



Categorical x-axis.
What do lines mean?

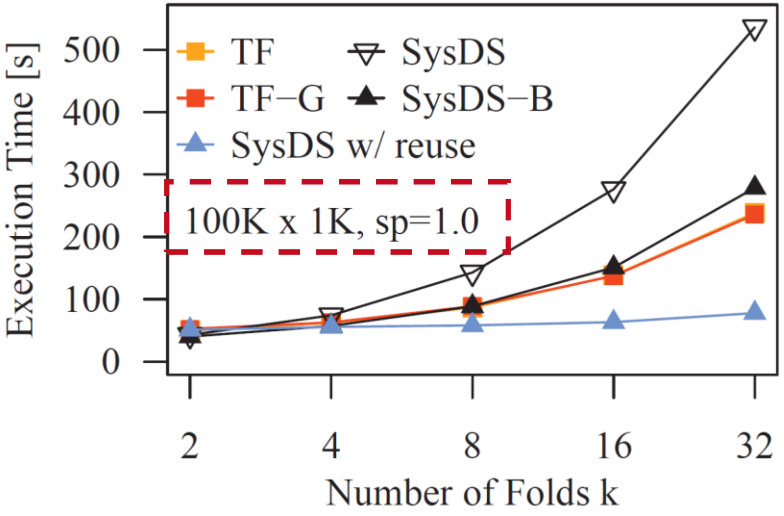
- **Legends**
  - Order them by appearance
  - Attach directly to graph

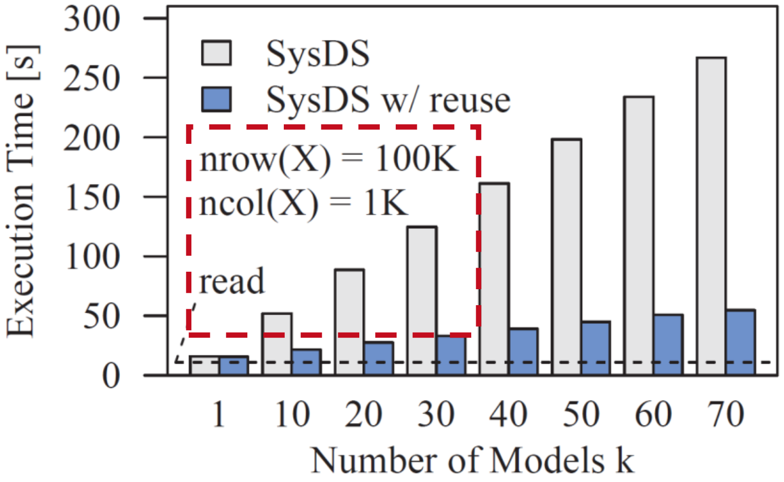


Human brain is a poor join processor
Humans get frustrated

Presentation – Figures, cont.

- **Diversity & Consistency**
  - Diversity: if applicable use mix of different plot types and tables
  - Consistency: use consistent colors and names for same baselines
- **Labeling**
  - Make the plots self-contained
  - Simplifies skimming and avoids join with text





[Matthias Boehm et al: SystemDS: A Declarative Machine Learning System for the End-to-End Data Science Lifecycle. **CIDR 2020**]

Reproducibility and RDM (Research Data Management)

In Computer Science (Data Management)

Research Data Management (RDM)

- **Overview**
  - Ensure reproducibility of research results and conclusions

  - **Common problem:** `“All code and data was on the student’s laptop and the student left / the laptop crashed.”`

  - Create value for others (compare, reuse, understand, extend)
  - EU Projects: Mandatory proposal section & deliverable on RDM plan

Excursus: FAIR Data Principles

[https://www.go-fair.org/fair-principles/]

- **Findable**
  - Metadata and data have globally unique **persistent identifiers**
  - Data described with rich **meta data**; registered/indexed and searchable
- **Accessible**
  - Metadata and data retrievable via open, free and universal **comm protocols**
  - Metadata accessible even when data no longer available
- **Interoperable**
  - Metadata and data use a formal, **accessible, and broadly applicable format**
  - Metadata and data use FAIR vocabularies and qualified references
- **Reusable**
  - Metadata and data described with plurality of accurate and relevant attributes
  - Clear license, **associated with provenance**, meets community standards

RDM in Practice @ DAMS Lab

- **Code and Artifacts**
  - Open-source systems
    - Apache SystemDS: https://github.com/apache/systemds
    - DAPHNE: https://github.com/daphne-eu/daphne
    - Complete code history, src/bin releases
    - LDE/AMLS/DIA programming projects in SystemDS and DAPHNE
  - Additional private GitHub repos for student projects / prototypes
  - Reproducibility for publications: https://github.com/damslab/reproducibility

- **Central Paper Repository**
  - All paper submissions with LaTeX sources, figures, reviews, rebuttals, etc.
  - All paper-related experiments
    - `Archive:` append-only experimental results
    - `Plots:` scripts and figures of plots
    - `Results:` latest results used for the current plots
    - `Scripts:` data preparation, baselines, benchmarks

→ **Automate your experiments as much as possible**

SIGMOD Reproducibility Process

[**Credit:** https://reproducibility.sigmod.org/]

- **Overview**
  - Accepted papers can submit package, verified by committee
  - “Artifacts Available”, “Artifacts Evaluated – Reusable”, “Results Replicated” badges
  - Most Reproducible Paper Award ($750, visibility)

- **#1 Artifact Availability and Evaluation (aka Repeatability)**
  - **Expected:** Prototype system, input data(gen), experiments, diagram creation
  - **Ideally:** Exceed minimal functionality, clear docs, facilitate reuse
- **#2 Reproducibility**
  - Central results and claims supported by the submitted experiments
  - **Expected:** similar behavior to that shown in the paper

SIGMOD Reproducibility Process, cont.

[**Credit:** https://reproducibility.sigmod.org/]

- **Ideal Reproducibility Submission**

  "At a minimum the authors should provide a complete set of scripts to **install the system**, **produce the data**, **run experiments** and **produce the resulting graphs** along with a **detailed Readme file** that describes the process step by step so it can be easily reproduced by a reviewer.

  The ideal reproducibility submission consists of a master [sic] script that:

  1. installs all systems needed,
  2. generates or fetches all needed input data,
  3. reruns all experiments and generates all results,
  4. generates all graphs and plots, and finally,
  5. recompiles the sources of the paper

  ... to produce a new PDF for the paper that contains the new graphs. "

- **Note: It takes time, plan from start**

Scientific Presentations

In Computer Science (Data Management)

Goals and Structure

- **Typical Goals of a Scientific Presentation**
  - Make audience aware of and interested in your work → visibility, get your paper cited
  - Show that you made significant contributions to relevant research area
  - Discuss problem/topic, get feedback from audience, foundation for offline discussion

- **Structure**
  - No single best structure, but best practices
  - Commonalities with scientific papers
    - Introduction/motivation (including necessary background)
    - Main part (your own contributions)
    - Experimental results
    - Summary & outlook

Limit the Scope

- **Typical Duration**
  - Lecture: ~90-120 min
  - Thesis defense: ~45 min
  - **Seminar talk: ~20 min**
  - **Conference talk ~5-15 min**

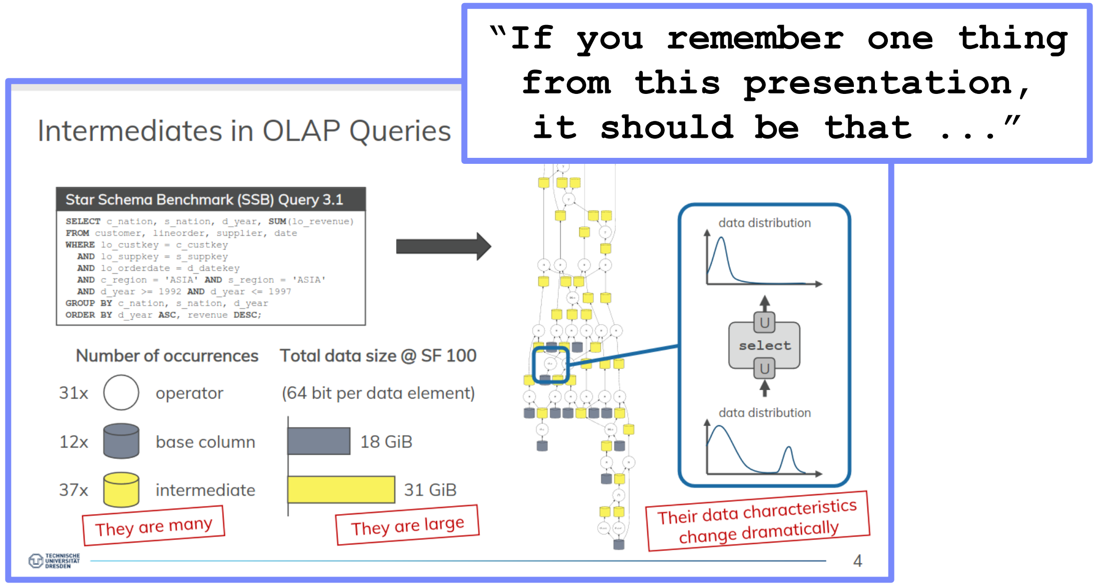
→ **Challenge: Usually not to fill the time, but to not go over time**

- **Limit the Scope, You Cannot Talk about Everything**
  - In terms of **breadth**
    - E.g., focus on a subset of the contributions
    - E.g., not all experiments
    - But: give overview of everything
  - In terms of **depth**
    - I.e., don't show all details
    - Present simplified results

→ **Challenge: Select the most important aspects**

Know Your Audience

- **Audience Characteristics**
  - How much can you expect them to know about your topic? Don't assume too much…
  - Need to adjust to you as a speaker in the beginning
- **Help Audience Not to Get Lost**
  - Clear motivation (don't rush through it)
  - Clear presentation outline (after motivation, otherwise hard to comprehend)
  - Outline and current position again after each section of the talk
  - Repeat important assumptions
  - Illustrate theory with concrete (running) examples
  - Take necessary time for complex diagrams, formulas, etc.
- **Steer Audience's Attention**
  - Make the most important points pop out



  - (Simple) animations
    - Reveal complex slide contents incrementally
    - But: avoid "Powerpoint Poisoning"

Slide Design

- **Avoid Slides Full of Text**
- **Avoid Complex Formulas and Source Code**
  - Unless they really contribute to the understanding
- **Avoid too Small Font Size**
- **Avoid too Many Effects**
- **Use Varied Layouts**
  - Not just lists of bullet points
  - Convey information in diagrams, figures, etc.
- **Use Simple Set of Colors**
- **Use Conscious Line Breaks**

- **Don't Simply Reuse the Figures from Your Paper**

paper

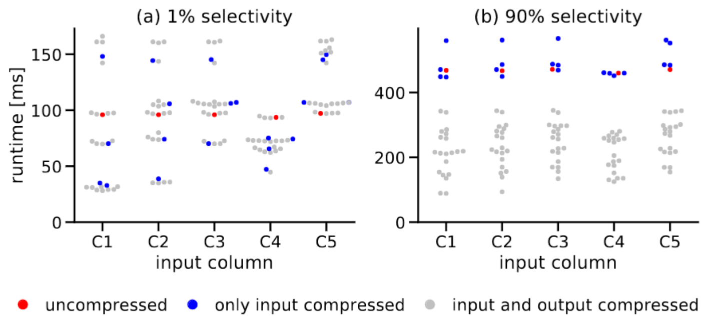
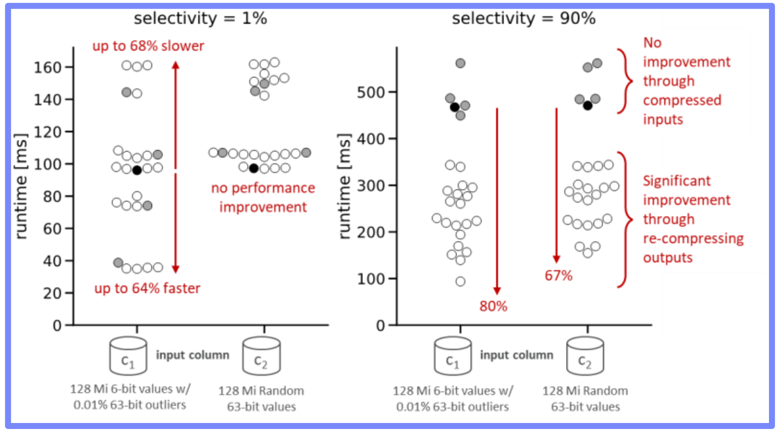


Figure 5: select **with on-the-fly de/re-compression.**

slides



Preparing a Presentation

- **#1 Planning**
  - Who's your **audience**?
  - What are the **key takeaways** you want to convey?
  - **Structure** of the talk, running examples
- **#2 Slide Creation**
  - Create **initial slide deck** (doesn't need to be pixel-perfect yet)
  - Should contain all planned content, consciously divided into slides
- **#3 Practice**
  - **Rehearse** aloud, ideally with audience (ask for max constructive criticism)
  - Does the **timing** fit? Is the talk **comprehensible**?
- **#4 Slide Finalization**
  - Make the slides **pixel-perfect**

Iterate steps
if necessary

Handling Questions

- **Basic Mindset**
  - **Always welcome questions as well as feedback/criticism**
  - **Take all questions constructively**
- **You Don’t Need to Always Have an Answer**
  - Answer as good as you can
  - Honestly admit if you don’t know the answer, e.g., if it needs further investigation
- **Take longer discussions offline**
  - Don’t bore the rest of the audience with too specific discussions
- **Page Numbers on Slides**
  - Help audience to refer to specific point in your presentation
- **Prepare Back-up Slides**
  - Extra slides not shown in the main presentation
  - Can be useful when answering questions

Summary and Q&A

- **Experiments and Result Presentation**
- **Reproducibility and RDM**
- **Scientific Presentations**
- **Remaining Questions?**

- **Seminar/Project Topic Assignment TBA by May 5 (Seminar) and May 12 (Project)**

- **Self-organized Seminar/Project Work**
- **Seminar/Project Submission Deadlines & Presentation Dates on Course Website**

- **Optional Consultation Hours**
  - **Seminar:** Mondays 14:00 – 16:00 hybrid in room B 120 and zoom
  - **Project:** Arrange individual meetings with project mentor