SCIENCE
PASSION
TECHNOLOGY

Introduction to Scientific Writing 03 Experiments & Reproducibility

Matthias Boehm

Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMK endowed chair for Data Management

Last update: Nov 11, 2021

ISDS

Announcements/Org

- **#1 Virtual Lectures**
  - https://tugraz.webex.com/meet/m.boehm
  - Optional attendance (independent of COVID)

- **#2 Course Registrations** (as of Nov 05)
  - Changes in WS20/21, now max constraints
  - **Introduction to Scientific Writing**

ISDS Group Boehm
39/40

- **#3 Timeline**
  - **Nov 11:** project selection via email to m.boehm@tugraz.at (11.59pm)
    subject: [Scientific Writing] Project Selection
  - **Dec 23:** paper submission via email to m.boehm@tugraz.at (11.59pm)
  - **Jan 13:** Final project presentation (all students)

Agenda

- **Experiments and Result Presentation**
- **Reproducibility and RDM**
- **Reminder: Paper Project Selection**

Experiments and Result Presentation

In Computer Science (Data Management)

[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)
  - **Karl Popper:** falsifiability of scientific results → refutable by evidence
- **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
- **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 → eval feasibility
- **#2 Micro Benchmarks**
  - Measure specific aspects in controlled and understandable scope
  - Bottom-up approach
- **#3 Benchmarks**
  - Evaluate on community/own benchmarks
  - Examples: TPC-C, TPC-H, TPC-DS, JOB, MLPerf
- **#4 End-to-end Applications**
  - Evaluate in larger scope of real datasets and query workloads
  - Examples: Customer workload, ML pipelines (dataprep, training, eval)

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 / data sets should be used?
  - Which workload / queries should be run?
  - Which hardware & software should be used?
  - **Metrics:** What to measure? How to measure?
  - **Comparison:** How to compare? CSI: How to find out what is going on?

Dataset Selection

- **Synthetic Data**
  - Generate data with specific data characteristics
  - Systematic evaluation w/ datasize, sparsity, etc
  - **Inappropriate for certain topics**: compression, ML accuracy

Representative of real data distributions?

- **"Real" Data Repositories**
  - Wide selection of available datasets w/ different characteristics
  - UCI ML Repository: https://archive.ics.uci.edu/ml/index.php
  - Florida Sparse 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?

Benchmarks

- **Overview**
  - 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)
  - Join ordering: JOB

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

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

- **#2 "Big Data"**
  - MR/Spark: BigBench, HiBench, SparkBench
  - Array Databases: GenBase

(See AMLS course for details)

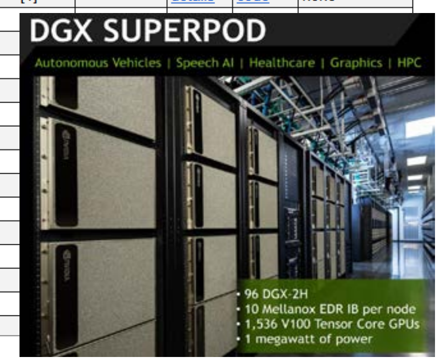
- **#3 Machine Learning Systems**
  - SLAB, DAWNBench, MLPerf, MLBench, AutoML Bench, Meta Worlds, TPCx-AI

Benchmarks, cont.

[**MLPerf** v0.6: https://mlperf.org/training-results-0-6/]

Closed Division Times

#	Submitter	System	Processor	#	Accelerator	#	Software	Image classification / ImageNet / ResNet-50 v1.5	Object detection, light-weight / COCO / SSD w/ ResNet-34	Object detection, heavy-wt. / COCO / Mask-R-CNN	Translation, recurrent / WMT E-G / NMT	Translation, non-recur. / WMT E-G / Transformer	Recommendation / MovieLens-20M / NCF	Reinforcement Learning / Go / Mini Go	Details	Code	Notes
Available in cloud																	
0.6-1	Google	TPUv3.32			TPUv3	16	TensorFlow, TPU 1.14.1.dev	42.19	12.61	107.03	12.25	10.20	[1]		details	code	none
0.6-2	Google	TPUv3.128			TPUv3	64	TensorFlow, TPU 1.14.1.dev	11.22	3.89	57.46	4.62	3.85	[1]		details	code	none
0.6-3	Google	TPUv3.256			TPUv3	128	TensorFlow, TPU 1.14.1.dev	6.86	2.76	35.60	3.53	2.81	[1]		details	code	none
0.6-4	Google	TPUv3.512			TPUv3	256	TensorFlow, TPU 1.14.1.dev	3.85	1.79		2.51	1.58	[1]		details	code	none
0.6-5	Google	TPUv3.1024			TPUv3	512	TensorFlow, TPU 1.14.1.dev	2.27	1.34		2.11	1.05	[1]		details	code	none
0.6-6	Google	TPUv3.2048			TPUv3	1024	TensorFlow, TPU 1.14.1.dev	1.28	1.21			0.85	[1]		details	code	none
Available on-premise																	
0.6-7	Intel	32x 2S CLX 8260L	CLX 8260L	64			TensorFlow						[1]	14.43	details	code	none
0.6-8	NVIDIA	DGX-1			Tesla V100	8	MXNet, NGC19.05	115.22					[1]		details	code	none
0.6-9	NVIDIA	DGX-1			Tesla V100	8	PyTorch, NGC19.05		22.36	207.48	20.55	20.34	[1]		details	code	none
0.6-10	NVIDIA	DGX-1			Tesla V100	8	TensorFlow, NGC19.05						[1]	27.39	details	code	none
0.6-11	NVIDIA	3x DGX-1			Tesla V100	24	TensorFlow, NGC19.05						[1]	13.57	details	code	none
0.6-12	NVIDIA	24x DGX-1			Tesla V100	192	PyTorch, NGC19.05			22.03			[1]		details	code	none
0.6-13	NVIDIA	30x DGX-1			Tesla V100	240	PyTorch, NGC19.05		2.67				[1]		details	code	none
0.6-14	NVIDIA	48x DGX-1			Tesla V100	384	PyTorch, NGC19.05				1.99		[1]		details	code	none
0.6-15	NVIDIA	60x DGX-1			Tesla V100	480	PyTorch, NGC19.05					2.05	[1]		details	code	none
0.6-16	NVIDIA	130x DGX-1			Tesla V100	1040	MXNet, NGC19.05	1.69					[1]		details	code	none
0.6-17	NVIDIA	DGX-2			Tesla V100	16	MXNet, NGC19.05	57.87									
0.6-18	NVIDIA	DGX-2			Tesla V100	16	PyTorch, NGC19.05		12.21	101.00	10.94	11.04					
0.6-19	NVIDIA	DGX-2H			Tesla V100	16	MXNet, NGC19.05	52.74									
0.6-20	NVIDIA	DGX-2H			Tesla V100	16	PyTorch, NGC19.05		11.41	95.20	9.87	9.80					
0.6-21	NVIDIA	4x DGX-2H			Tesla V100	64	PyTorch, NGC19.05		4.78	32.72							
0.6-22	NVIDIA	10x DGX-2H			Tesla V100	160	PyTorch, NGC19.05					2.41					
0.6-23	NVIDIA	12x DGX-2H			Tesla V100	192	PyTorch, NGC19.05			18.47							
0.6-24	NVIDIA	15x DGX-2H			Tesla V100	240	PyTorch, NGC19.05		2.56								
0.6-25	NVIDIA	16x DGX-2H			Tesla V100	256	PyTorch, NGC19.05				2.12						
0.6-26	NVIDIA	24x DGX-2H			Tesla V100	384	PyTorch, NGC19.05				1.80						
0.6-27	NVIDIA	30x DGX-2H, 8 chips each			Tesla V100	240	PyTorch, NGC19.05		2.23								
0.6-28	NVIDIA	30x DGX-2H			Tesla V100	480	PyTorch, NGC19.05					1.59					
0.6-29	NVIDIA	32x DGX-2H			Tesla V100	512	MXNet, NGC19.05	2.59									
0.6-30	NVIDIA	96x DGX-2H			Tesla V100	1536	MXNet, NGC19.05	1.33									



96 x DGX-2H = 96 * 16 = 1536 V100 GPUs

➔ ~ 96 * $400K = **$35M – $40M**

[https://www.forbes.com/sites/tiriasresearch/2019/06/19/nvidia-offers-a-turnkey-supercomputer-the-dgx-superpod/#693400f43ee5]

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 w/ different runtime and compiler
  - Usually, not directly comparable but important for grounding
  - E.g.,: for SystemDS → R, Julia, Spark, TensorFlow, PyTorch

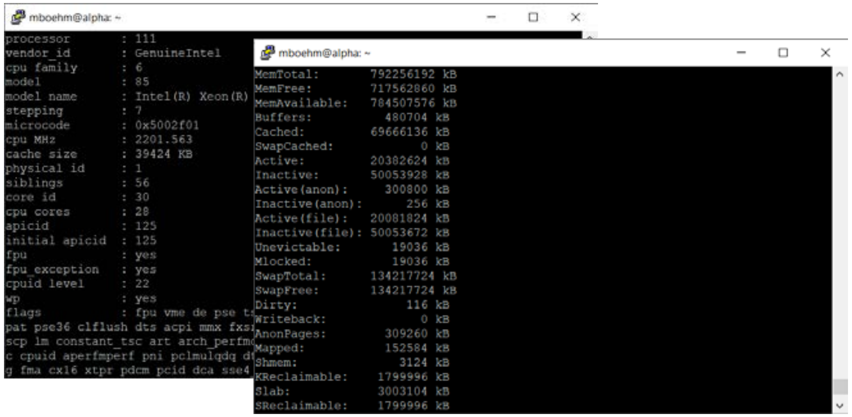
- **Problem of Weak Baselines**
  - Authors want to show improvements
  - Successive improvements over state-of-the-art don’t add up

[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**]

Presentation – Experimental Setting

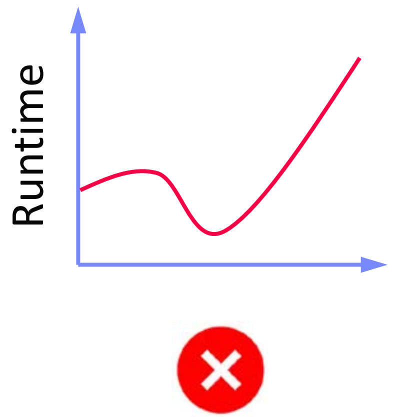
- **Hardware Selection**
  - Multiple nodes for distributed computation
  - Avoid too outdated HW (irrelevance)
- **Find Balanced Level of Detail**
  - Underspecified: “`We ran all experiments on an Intel CPU`”
  - Over-specified:
    `cat /proc/cpuinfo`
    `cat /proc/meminfo`



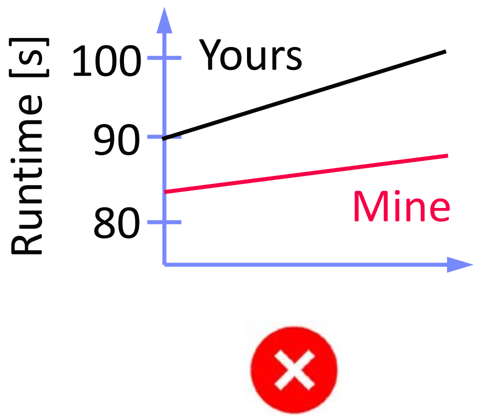
- **Recommendation**
  - **HW components:** #nodes, CPUs, memory, network, I/O
  - **SW components:** OS, programming language, versions, other software
  - **Baselines** and configuration → Use **recent versions of baseline systems**
  - **Data and workloads** w/ data sizes, parameters, configurations

Presentation – Figures

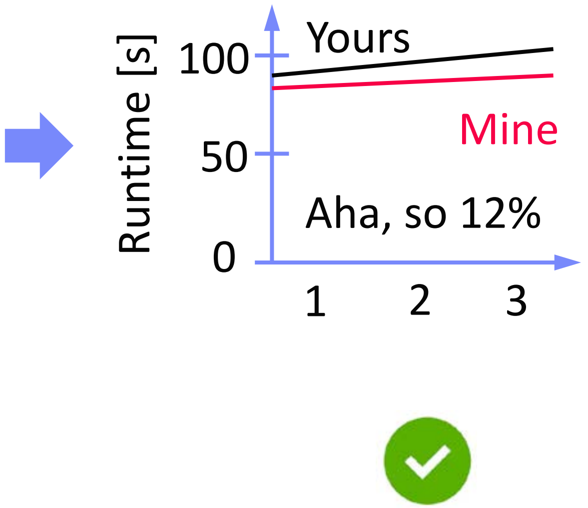
- **Axes**
  - Use Informative axes labels with units (e.g., Total Execution Time [ms])
  - Don’t cheat or mislead readers and reviewers
  - Start y-axis at 0 for linear scale



What are the units?
Where are the tics?



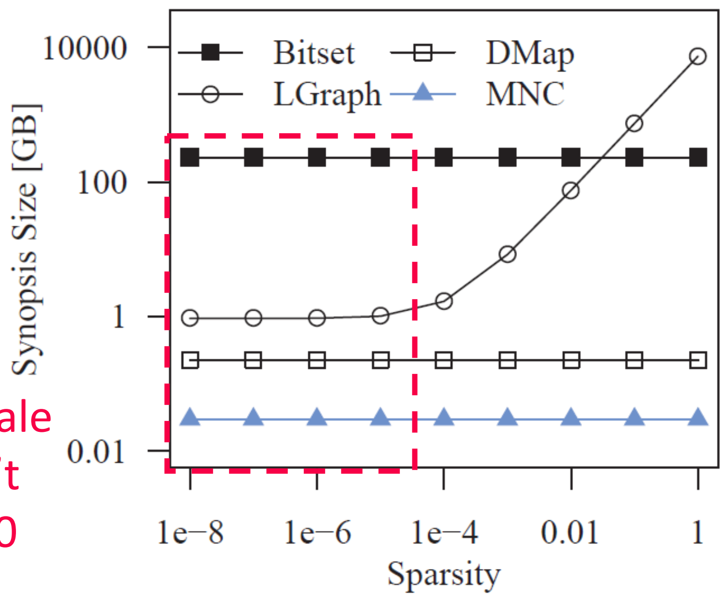
This is a misleading y axis



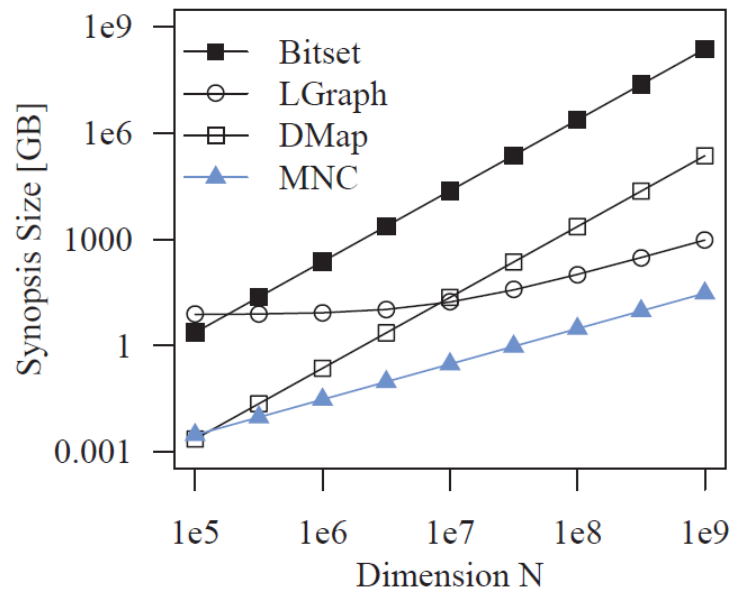
Presentation – Figures, cont.

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

If there are multiple relevant parameters, show them all



For log-scale you can't start at 0



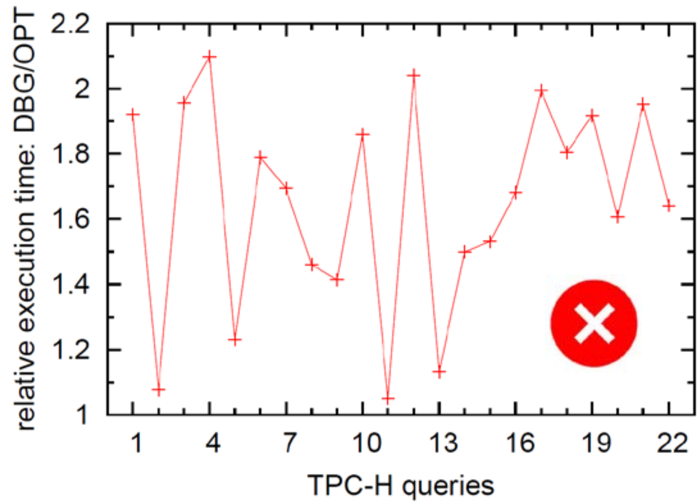
Don't limit range to make you look good

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

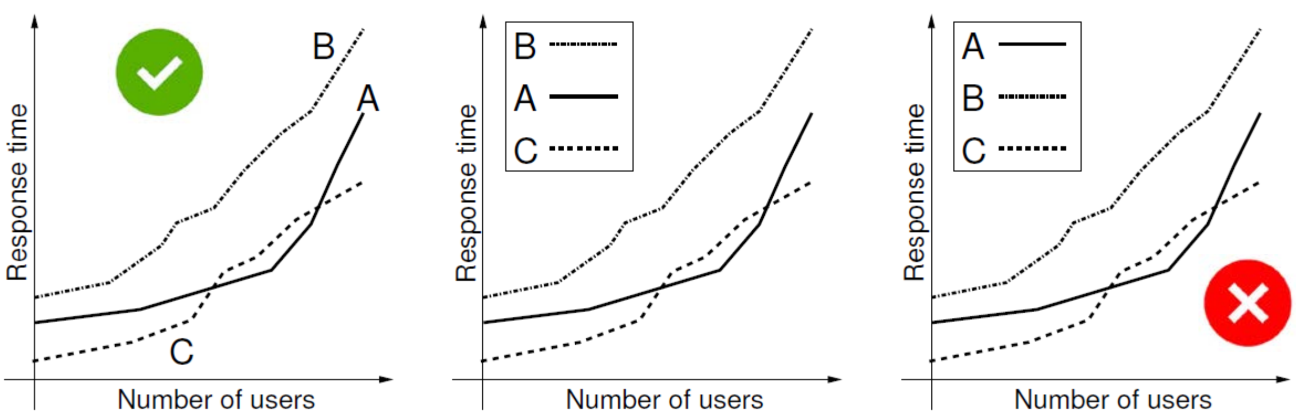
Presentation – Figures, cont.

[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)



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

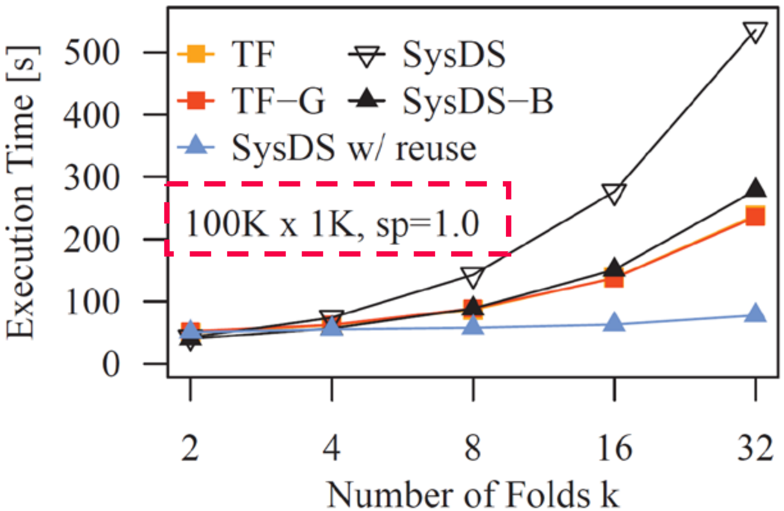


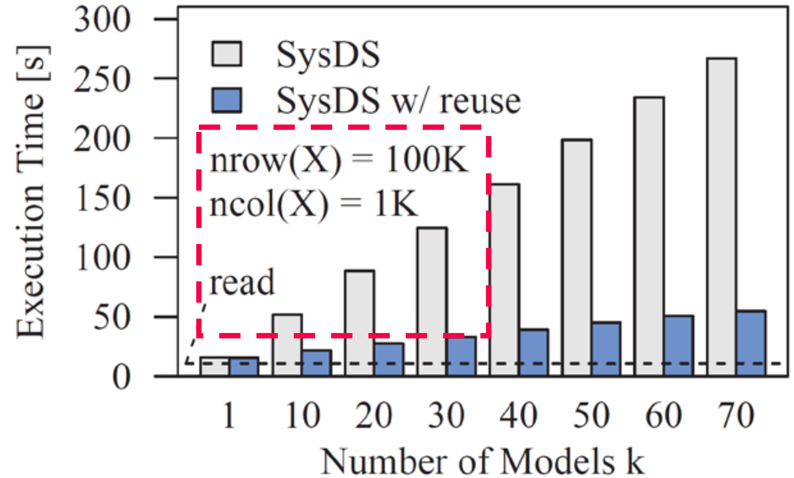
Human brain is a poor join processor
Humans get frustrated

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

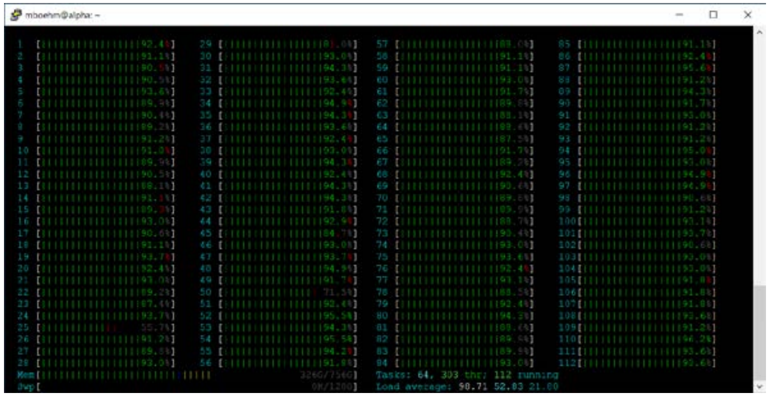
[Matthias Boehm et al: SystemDS: A Declarative Machine Learning System for the End-to-End Data Science Lifecycle. **CIDR 2020**]





Presentation – Result Interpretation

- **Use the Right OS Tools**
  - System-specific tracing/statistics
  - `top / htop / iotop`
    (looks **CPU bound**)
  - `perf -stat -d ./run.sh`
    (no, it's **memory-bandwidth bound**)



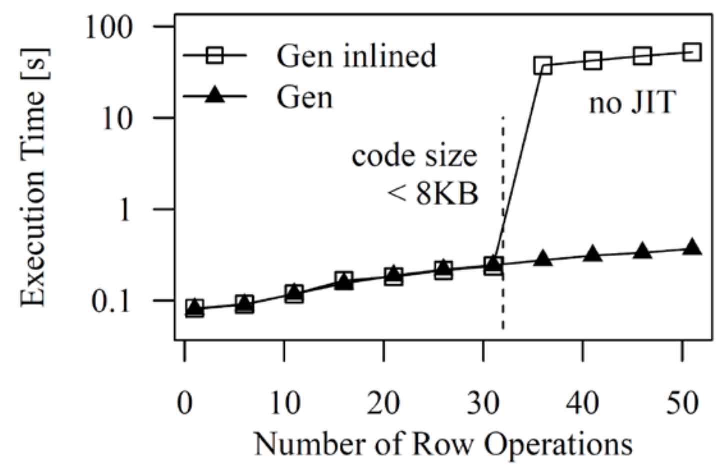
```
 Performance counter stats for './run.sh':
       12721364.53 msec task-clock              #   83.640 CPUs utilized
            463352      context-switches        #    0.036 K/sec
     5455536095415      instructions            #    0.14  insn per cycle         (62.50%)
      335314473273      branches                #   26.358 M/sec                  (62.50%)
        1463380955      branch-misses           #    0.44% of all branches        (62.50%)
     2185062643097      L1-dcache-loads         #  171.763 M/sec                  (62.50%)
      142845949268      L1-dcache-load-misses   #    6.54% of all L1-dcache hits  (62.50%)
        3375555316      LLC-loads               #    0.265 M/sec                  (50.00%)
        1016330404      LLC-load-misses         #   30.11% of all LL-cache hits   (50.00%)

     152.096000108 seconds time elapsed
   12052.466691000 seconds user
     674.704421000 seconds sys
```

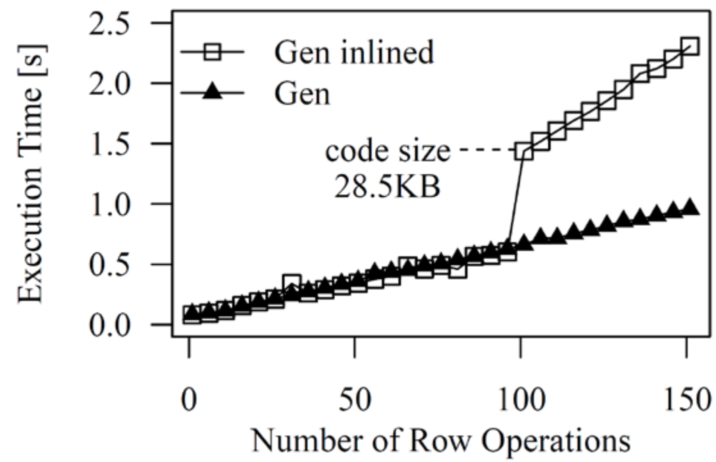
Don't just report the results but try to understand and explain them

Presentation – Result Interpretation, cont.

- **Use the Right PL Tools / Flags**
  - E.g., Understanding Java JIT compilation `-XX:+PrintCompilation`



  - E.g., Understanding HW Cache Hierarchy (L1i 32KB) `-XX:-DontCompileHugeMethods`



[Matthias Boehm et al: On Optimizing Operator Fusion Plans for Large-Scale Machine Learning in SystemML. **PVLDB 11(12) 2018**]

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
- **RDM @ TU Graz:** https://www.tugraz.at/sites/rdm/home/
  - Toni Ross-Hellauer → Ilire Hasani-Mavriqi / Sarah Stryeck (ISDS): Open and Reproducible Research Group (ORRG)
  - TU Graz RDM Policy since 12/2019, towards faculty-specific RDM policies

"Ensure that research data, code and any other materials needed to reproduce research findings are appropriately documented, stored and shared in a research data repository in accordance with the FAIR principles (Findable, Accessible, Interoperable and Reusable) for at least 10 years from the end of the research project, unless there are valid reasons not to do so. [...]

Develop a written data management strategy for managing research outputs within the first 12 months of the PhD study as part of their supervision agreements."

Excursus: FAIR Data Principles

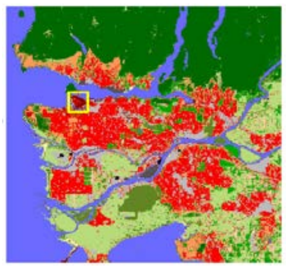
[https://www.go-fair.org/fair-principles/]

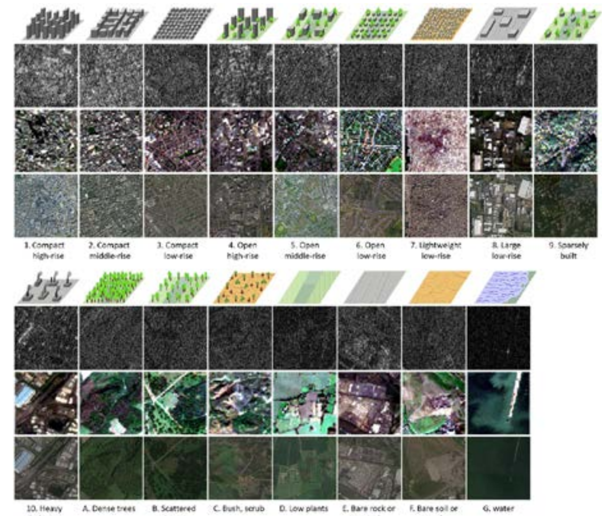
- **#1 Findable**
  - Metadata and data have globally unique **persistent identifiers**
  - Data describes w/ rich **meta data**; registered/indexes and searchable
- **#2 Accessible**
  - Metadata and data retrievable via open, free and universal **comm protocols**
  - Metadata accessible even when data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

RDM in Practice – Example So2Sat LCZ42

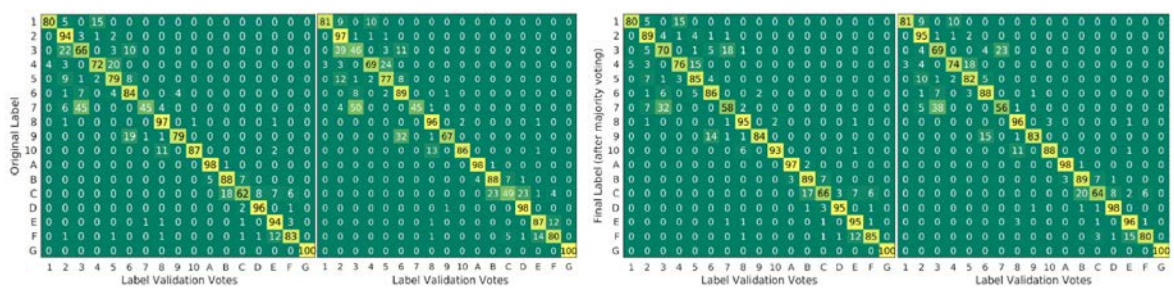
[Xiao Xiang Zhu et al: So2Sat LCZ42: A Benchmark Dataset for the Classification of Global Local Climate Zones. **GRSM 2020**]

- **Data and ML Pipelines**
  - ESA Sentinel-1/2 datasets → 4PB/year
  - Training of local climate zone classifiers on So2Sat LCZ42 (15 experts, 400K instances, 10 labels each, 85% confidence, ~55GB H5)
  - **ML pipeline:** preprocessing, ResNet18, climate models

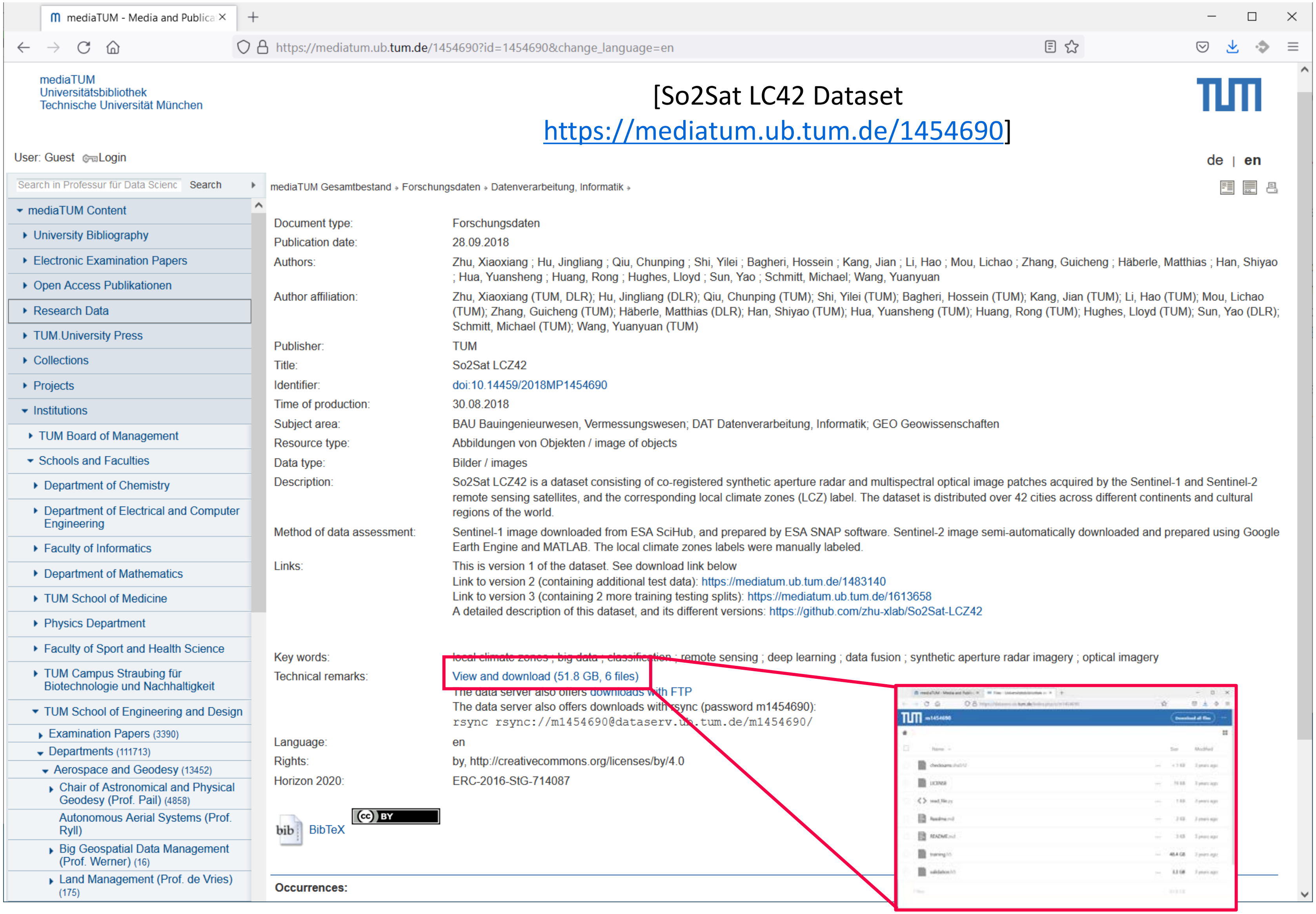




- **Label Creation/ Validation**
  - Team learning
  - Labeling w/ checks
  - Label validation
  - Quantitative validation w/ 10 expert votes on correctness



RDM in Practice – Example So2Sat LCZ42, cont.

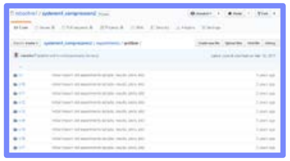


RDM in Practice @DAMSLab

- **Code and Artifacts**
  - Apache SystemDS: https://github.com/apache/systemds (OSS)
    - Complete code history, src/bin releases (**SystemDS 2.2.0 in Oct 2021**)
    - DIA / AMLS programming projects in SystemDS
  - Additional private github repos for student projects / prototypes

- **Central Paper Repository**
  - All paper submissions w/ 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

- **Overview**
  - Accepted papers can submit package, verified by committee
  - ACM Results Replicated / ACM Artifacts Available labels
  - Most Reproducible Paper Award ($750, visibility)

- **#1 Replicability (aka Repeatability)**
  - Recreate result data and graphs shown in the final paper
  - **Expected:** same trend of baseline comparisons, parameter influence

- **#2 Reproducibility**
  - Verify robustness of results wrt parameters and environments
  - **Examples:** different data and workload characteristics, hardware

SIGMOD Reproducibility Process, cont.

- **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 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. "

[**Credit:** db-reproducibility.seas.harvard.edu/#Guidelines]

- **Note: It takes time, plan from start**
  - We prepared for SIGMOD 2019 Repro, but finally, not submitted (ran out of time)

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

Excursus: SIGMOD Contributions Award 2020

The SIGMOD 2020 Contributions Award recognizes the innovative work in the data management community to encourage scientific reproducibility of our publications. Reproducibility was introduced at the 2008 SIGMOD Conference and since then has influenced how the community approaches experimental evaluation.

Philippe Bonnet
(ITU Copenhagen)

Stratos Idreos
(Harvard)

Ioana Manolescu
(Ecole Polytechnique)

Dennis Shasha
(NYU)

Stefan Manegold
(CWI Amsterdam)

Juliana Freire
(NYU)

Tools for Automation

- **Motivation**
  - Tooling for running artifacts in a self-contained manner
  - Metadata storage in semi-structured formats like JSON/XML

- **Examples**

Name	Target	Link
CK	ML Systems	http://ctuning.org
CWL	Analysis Workflows	http://commonwl.org
Popper	Container Workflows	https://github.com/systemslab/popper
ReproZip	General-purpose Bundles	http://reprozip.org
Sciunit	Self-contained Experiments Bundles	http://sciunit.run
Sumatra	Numerical Simulations	https://github.com/open-research/sumatra

Reminder: Paper Projects

In Computer Science (Data Management)

Overview Paper Projects

Alternative: LV combined with bachelor thesis

- **Team**
  - 1-4 person teams (w/ clearly separated responsibilities)

See **01 Structure of Scientific Papers** for project list

- **Project**
  - Pick from a given list of papers / groups of papers
  - **#1** Write short summary paper (#pages = 2 * team-size, written in LaTeX, ACM acmart template, document-class sigconf, PDF)
  - **#2** Prepare and present talk on paper summary (7min + 3min Q&A)
- **Timeline**
  - **Oct 28:** List of projects proposals, feel free to bring your own, or ask for extended proposals (e.g., ML systems, distributed systems)
  - **Nov 11:** project selection via email to m.boehm@tugraz.at (11.59pm) subject: [Scientific Writing] Project Selection
  - **Dec 23:** paper submission via email to m.boehm@tugraz.at (11.59pm)
  - **Jan 13:** Final project presentation (all students)

Summary and Q&A

- **Experiments and Result Presentation**
- **Reproducibility and RDM**
- **Reminder: Paper Project Selection**

- **Remaining Questions?**