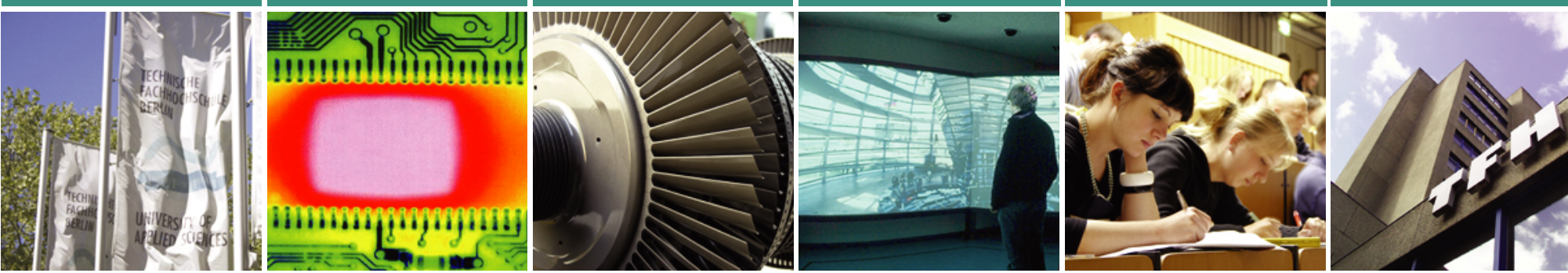
Version August 31, 2020

A Framework for Supporting Repetition and Evaluation in the Process of Cloud-based DBMS Performance Benchmarking

Patrick K. Erdelt

Beuth Hochschule fuer Technik Berlin, Luxemburger Strasse 10, Berlin, Germany
patrick.erdelt@beuth-hochschule.de



Framework for DBMS Performance Benchmarking

Identify the need for a framework to **support the process** [1, 4, 5]:

1. Provide dynamic testbeds - cloud orchestration tools
2. Automated collection of metadata
3. Help in statistical evaluation

$\longrightarrow$ New framework

Content of Talk

1. Design Decisions
2. Two Packages
3. Two Experiments

Design Decisions

1. DBMS as Docker Containers
   - Brytlyt, Exasol, Kinetica, MariaDB, MemSQL, MonetDB, MS SQL Server, MySQL, OmniSci, Oracle Database, PostgreSQL, SQreamDB, ...
   - Reproducibility and portability are very high

Design Decisions

1. DBMS as Docker Containers
   - Brytlyt, Exasol, Kinetica, MariaDB, MemSQL, MonetDB, MS SQL Server, MySQL, OmniSci, Oracle Database, PostgreSQL, SQreamDB, ...
   - Reproducibility and portability are very high
2. Monitoring via Prometheus / Grafana
   - CPU, RAM, GPU, VRAM, FS, Network, ..., and extensible
   - Monitoring easier and more schematic than profiling and logging

Design Decisions

1. DBMS as Docker Containers
   - Brytlyt, Exasol, Kinetica, MariaDB, MemSQL, MonetDB, MS SQL Server, MySQL, OmniSci, Oracle Database, PostgreSQL, SQreamDB, ...
   - Reproducibility and portability are very high
2. Monitoring via Prometheus / Grafana
   - CPU, RAM, GPU, VRAM, FS, Network, ..., and extensible
   - Monitoring easier and more schematic than profiling and logging
3. Workflow Management for Kubernetes
   - Pod = Collection of containers, sharing host and network interface
   - Automated collection of metadata

Design Decisions

1. DBMS as Docker Containers
   - Brytlyt, Exasol, Kinetica, MariaDB, MemSQL, MonetDB, MS SQL Server, MySQL, OmniSci, Oracle Database, PostgreSQL, SQreamDB, ...
   - Reproducibility and portability are very high
2. Monitoring via Prometheus / Grafana
   - CPU, RAM, GPU, VRAM, FS, Network, ..., and extensible
   - Monitoring easier and more schematic than profiling and logging
3. Workflow Management for Kubernetes
   - Pod = Collection of containers, sharing host and network interface
   - Automated collection of metadata
4. Python
   - Powerful Data Analysis Scripting Language

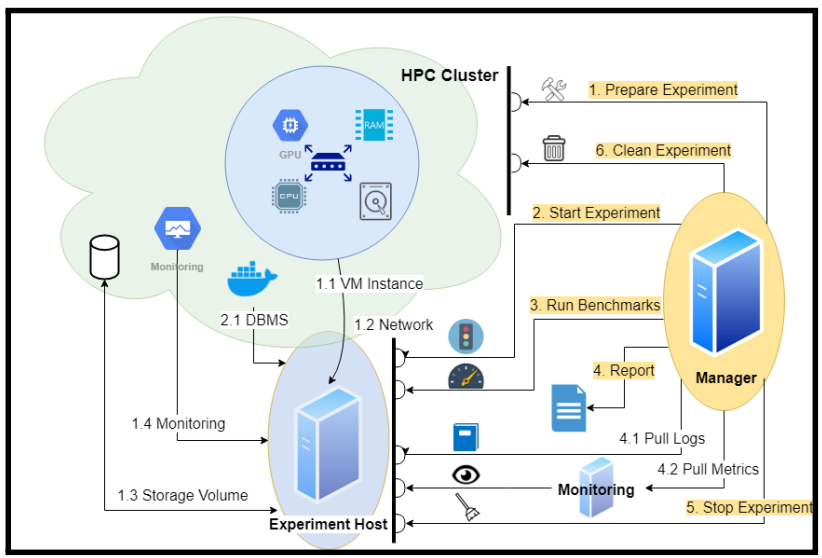
Bexhoma

Bexhoma helps managing DBMS benchmarking experiments in a HPC cluster environment [2].

It provides DBMS as ready-to-use black box (Pod) having

- **default port - for monitoring**
- **default port - for JDBC**
- **metadata - for classification of result**

Automate as much as possible!

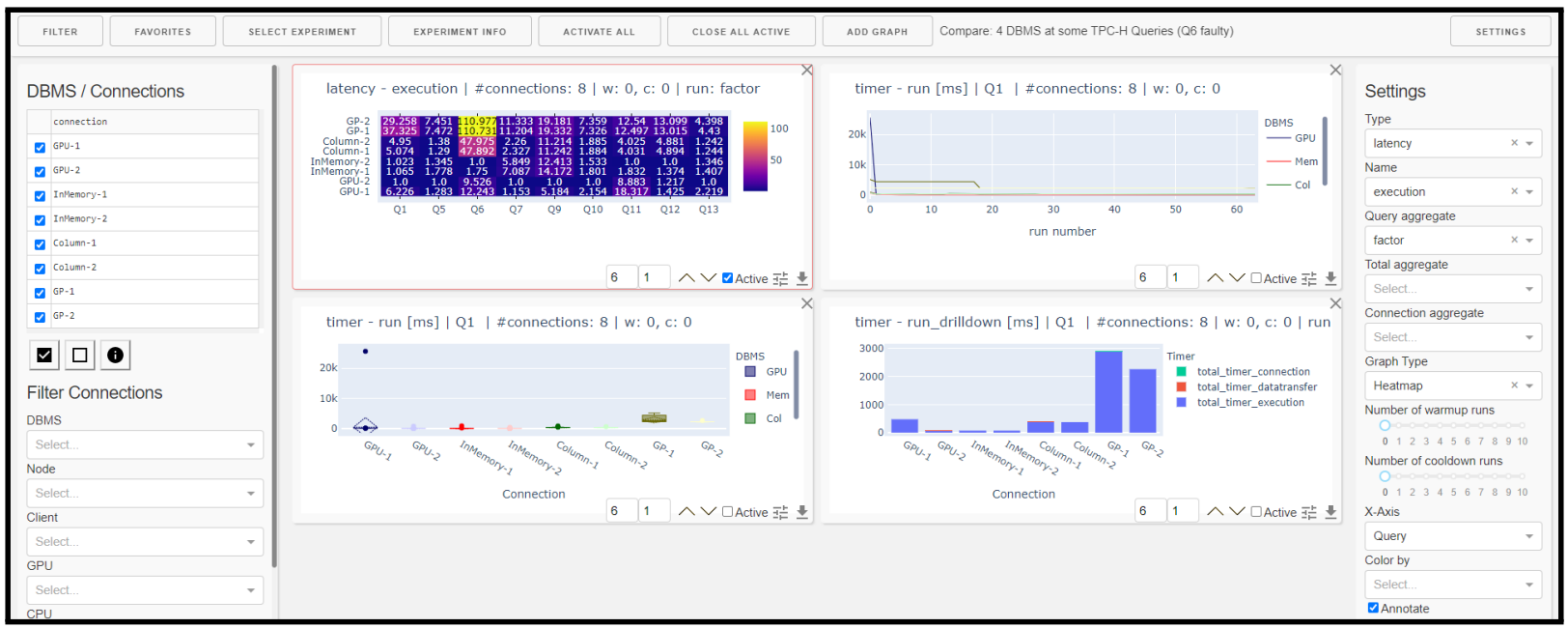


Experiment Workflow

DBMSBenchmarker

DBMSBenchmarker is an application-level benchmarking tool for DBMS [3]

- connects via JDBC
- runs a given list of (parametrized and randomized) SQL queries
- collects timing and hardware metrics
- helps in verification and evaluation of results



Result Dashboard

Use Case: Compare two DBMS

```
# basic configs
cluster  = bexhoma.cluster_manager()

# set  experiment basics
cluster.setExperiment(
        volume="tpch",
        script="SF=1",
        queryfile="queries.config",
        cpu_type="epyc-7542",
        cpu = "4000m",
        memory = "16Gi")

# run experiments
cluster.runExperiment(docker="PostgreSQL")
cluster.runExperiment(docker="MySQL")
```

1. Workload TPC-H
   - Data Source
   - DDL Script
   - Query List
2. Hardware Environment
   - CPU Type
   - CPU Cores
   - Memory
3. DBMS

First Experiment: Compare DBMS

	DBMS	Processor	Install	Data	CPU
γ_1	In-Memory	AMD EPYC 7542	true	true	4
γ_2	In-Memory	AMD EPYC 7542	false	true	4
γ_3	General-Purpose	AMD EPYC 7542	true	true	4
γ_4	General-Purpose	AMD EPYC 7542	false	true	4
γ_5	Columnwise	AMD EPYC 7542	true	true	4
γ_6	Columnwise	AMD EPYC 7542	false	true	4
γ_7	GPU-enhanced	NVIDIA V100	true	true	4
γ_8	GPU-enhanced	NVIDIA V100	false	true	4

First Experiment

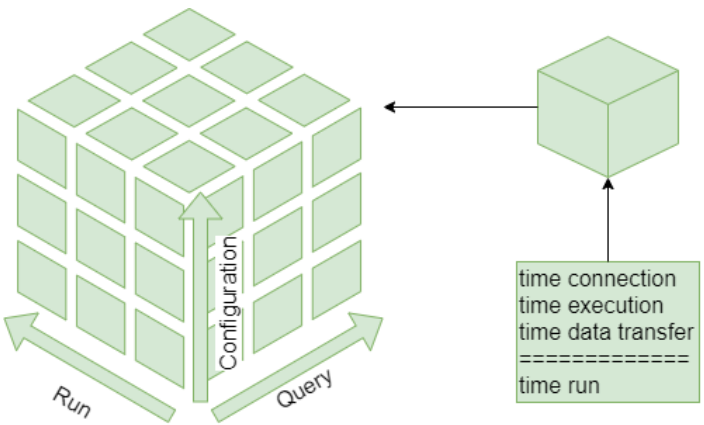
- 4 DBMS, 8 Configs
- 9 TPC-H Queries: Q1, Q5 - Q7, Q9 - Q13
- 64 runs each $\longrightarrow$ confidence
- $13.824 = 8 \cdot 9 \cdot 64 \cdot 3$ measured times
- Limit to 4 CPUs, do not reinstall for second workload

First Experiment: Compare DBMS

	DBMS	Processor	Install	Data	CPU
γ_1	In-Memory	AMD EPYC 7542	true	true	4
γ_2	In-Memory	AMD EPYC 7542	false	true	4
γ_3	General-Purpose	AMD EPYC 7542	true	true	4
γ_4	General-Purpose	AMD EPYC 7542	false	true	4
γ_5	Columnwise	AMD EPYC 7542	true	true	4
γ_6	Columnwise	AMD EPYC 7542	false	true	4
γ_7	GPU-enhanced	NVIDIA V100	true	true	4
γ_8	GPU-enhanced	NVIDIA V100	false	true	4

First Experiment

- 4 DBMS, 8 Configs
- 9 TPC-H Queries: Q1, Q5 - Q7, Q9 - Q13
- 64 runs each $\longrightarrow$ confidence
- $13.824 = 8 \cdot 9 \cdot 64 \cdot 3$ measured times
- Limit to 4 CPUs, do not reinstall for second workload



Dimensions

- Runs (64)
- Queries (9)
- Configurations (8)
  - DBMS
  - CPU / GPU
  - assigned cluster node
  - ...

Aggregations

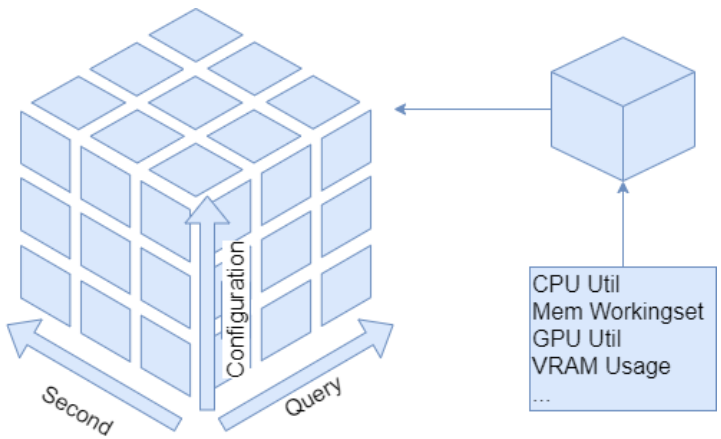
- First, Last
- Min, Max, Sum
- Mean, StdDev
  sensitive outlier
- Median, IQR
  insensitive outlier
- Geom Mean
  heterogeneous scales
- CV, QCOD
  scaleless variation

Second Experiment: Compare CPUs

	DBMS	Processor	Install	Data	CPU
γ_1	In-Memory	EPYC 7542	true	false	∞
...	...	...	...	...	...
γ_{11}	In-Memory	Xeon CPU E5-2630 v3	true	false	∞
...	...	...	...	...	...
γ_{21}	In-Memory	Xeon CPU E5-2630 v4	true	false	∞
...	...	...	...	...	...
γ_{31}	In-Memory	Xeon Silver 4110 CPU	true	false	∞
...	...	...	...	...	...
γ_{41}	In-Memory	Opteron Processor 6378	true	false	∞
...	...	...	...	...	...
γ_{50}	In-Memory	Opteron Processor 6378	true	false	∞

Second Experiment

- 1 DBMS, 50 Configs
- 9 TPC-H Queries: Q1, Q5 - Q7, Q9 - Q13
- 10 times on 5 different CPUs
- 1024 runs each $\longrightarrow$ stress test
- $1.382.400 = 50 \cdot 9 \cdot 1024 \cdot 3$ measured times
- No limit to CPUs, reinstall each time, ignore data



Dimensions

- Second of execution
- Queries
- Configurations
  - DBMS
  - CPU / GPU
  - assigned cluster node
  - ...

Aggregations

- First, Last
- Min, Max, Sum
- Mean, StdDev
  sensitive outlier
- Median, IQR
  insensitive outlier
- Geom Mean
  heterogeneous scales
- CV, QCOD
  scaleless variation

Conclusion

- Support for Repetition and Evaluation
- Presentation of Framework
- Motivation Docker, Monitoring, Kubernetes, Python
- Feasibility of seeing characteristics

[1] Brent, L., Fekete, A.: A versatile framework for painless benchmarking of database management systems. In: Chang, L., Gan, J., Cao, X. (eds.) Databases Theory and Applications. pp. 45–56. Springer International Publishing, Cham (2019)

[2] Erdelt, P.: Benchmark-Experiment-Host-Manager (Sep 2020), `https://github.com/Beuth-Erdelt/Benchmark-Experiment-Host-Manager`, [Online; accessed 1. Aug. 2020]

[3] Erdelt, P.: DBMS-Benchmarker (Aug 2020), `https://github.com/Beuth-Erdelt/DBMS-Benchmarker`, [Online; accessed 1. Aug. 2020]

[4] Raasveldt, M., Holanda, P., Gubner, T., Mühleisen, H.: Fair benchmarking considered difficult: Common pitfalls in database performance testing. In: Proceedings of the Workshop on Testing Database Systems. pp. 2:1–2:6. DBTest'18, ACM, New York, NY, USA (2018), `http://doi.acm.org/10.1145/3209950.3209955`

[5] Seybold, D., Domaschka, J.: Is distributed database evaluation cloud-ready? In: New Trends in Databases and Information Systems. pp. 100–108. Springer International Publishing, Cham (2017)