DBMSBenchmarker

Release 0.1

Patrick Erdelt

Aug 31, 2021

TABLE OF CONTENTS:

1 Key Features **3**

2 Installation **5**

3 Basic Usage **7**
- 3.1 Configuration . . . 7
- 3.2 Perform Benchmark . . . 8
- 3.3 Evaluate Results in Dashboard . . . 8

4 Benchmarking in a Kubernetes Cloud **9**

5 Limitations **11**

6 References **13**

DBMS-Benchmarker is a Python-based application-level blackbox benchmark tool for Database Management Systems (DBMS). It aims at reproducible measuring and easy evaluation of the performance the user receives even in complex benchmark situations. It connects to a given list of DBMS (via JDBC) and runs a given list of (SQL) benchmark queries. Queries can be parametrized and randomized. Results and evaluations are available via a Python interface. Optionally some reports are generated. An interactive dashboard assists in multi-dimensional analysis of the results.

See the homepage for more documentation.

CHAPTER

ONE

KEY FEATURES

DBMS-Benchmarker

- is Python3-based
- connects to all DBMS having a JDBC interface - including GPU-enhanced DBMS
- requires *only* JDBC - no vendor specific supplements are used
- benchmarks arbitrary SQL queries - in all dialects
- allows planning of complex test scenarios - to simulate realistic or revealing use cases
- allows easy repetition of benchmarks in varying settings - different hardware, DBMS, DBMS configurations, DB settings etc
- investigates a number of timing aspects - connection, execution, data transfer, in total, per session etc
- investigates a number of other aspects - received result sets, precision, number of clients
- collects hardware metrics from a Grafana server - hardware utilization, energy consumption etc
- helps to evaluate results - by providing
  - standard Python data structures
  - predefined evaluations like statistics, plots, Latex reporting
  - an inspection tool
  - an interactive dashboard

In the end this tool provides metrics that can be analyzed by aggregation in multi-dimensions, like maximum throughput per DBMS, average CPU utilization per query or geometric mean of run latency per workload.

For more informations, see a *basic example*, take a look at help for a full list of options or take a look at a demo report.

The code uses several Python modules, in particular jaydebeapi for handling DBMS. This module has been tested with Brytlyt, Citus, Clickhouse, DB2, Exasol, Kinetica, MariaDB, MariaDB Columnstore, MemSQL, Mariadb, MonetDB, MySQL, OmniSci, Oracle DB, PostgreSQL, SingleStore, SQL Server and SAP HANA.

CHAPTER

TWO

INSTALLATION

Run `pip install dbmsbenchmarker`

CHAPTER

THREE

BASIC USAGE

The following very simple use case runs the query `SELECT COUNT(*) FROM test` 10 times against one local MySQL installation. As a result we obtain an interactive dashboard to inspect timing aspects.

3.1 Configuration

We need to provide

- a DBMS configuration file, e.g. in `./config/connections.config`

```
[
  {
    'name': "MySQL",
    'active': True,
    'JDBC': {
      'driver': "com.mysql.cj.jdbc.Driver",
      'url': "jdbc:mysql://localhost:3306/database",
      'auth': ["username", "password"],
      'jar': "mysql-connector-java-8.0.13.jar"
    }
  }
]
```

- the required JDBC driver, e.g. `mysql-connector-java-8.0.13.jar`
- a Queries configuration file, e.g. in `./config/queries.config`

```
{
  'name': 'Some simple queries',
  'queries':
  [
    {
      'title': "Count all rows in test",
      'query': "SELECT COUNT(*) FROM test",
      'numRun': 10
    }
  ]
}
```

3.2 Perform Benchmark

Run the CLI command:

`dbmsbenchmarker run -e yes -b -f ./config`

- `-e yes`: This will precompile some evaluations and generate the timer cube.
- `-b`: This will suppress some output
- `-f`: This points to a folder having the configuration files.

This is equivalent to `python benchmark.py run -e yes -b -f ./config`

For more options, see the documentation

After benchmarking has been finished we will see a message like

```
Experiment <code> has been finished
```

The script has created a result folder in the current directory containing the results. <code> is the name of the folder.

3.3 Evaluate Results in Dashboard

Run the command:

`dbmsdashboard`

This will start the evaluation dashboard at `localhost:8050`. Visit the address in a browser and select the experiment <code>.

This is equivalent to `python dashboard.py`.

Alternatively you may use a Jupyter notebook.

CHAPTER

FOUR

BENCHMARKING IN A KUBERNETES CLOUD

This module can serve as the **query executor** [2] and **evaluator** [1] for distributed parallel benchmarking experiments in a Kubernetes Cloud, see the orchestrator for more details.

CHAPTER

FIVE

LIMITATIONS

Limitations are:

- strict black box perspective - may not use all tricks available for a DBMS
- strict JDBC perspective - depends on a JVM and provided drivers
- strict user perspective - client system, network connection and other host workloads may affect performance
- not officially applicable for well known benchmark standards - partially, but not fully complying with TPC-H and TPC-DS
- hardware metrics are collected from a monitoring system - not as precise as profiling
- no GUI for configuration
- strictly Python - a very good and widely used language, but maybe not your choice

Other comparable products you might like

- Apache JMeter - Java-based performance measure tool, including a configuration GUI and reporting to HTML
- HammerDB - industry accepted benchmark tool, but limited to some DBMS
- Sysbench - a scriptable multi-threaded benchmark tool based on LuaJIT
- OLTPBench -Java-based performance measure tool, using JDBC and including a lot of predefined benchmarks

CHAPTER

SIX

REFERENCES

[1] A Framework for Supporting Repetition and Evaluation in the Process of Cloud-Based DBMS Performance Benchmarking

> Erdelt P.K. (2021) A Framework for Supporting Repetition and Evaluation in the Process of Cloud-Based DBMS Performance Benchmarking. In: Nambiar R., Poess M. (eds) Performance Evaluation and Benchmarking. TPCTC 2020. Lecture Notes in Computer Science, vol 12752. Springer, Cham. https://doi.org/10.1007/978-3-030-84924-5_6

[2] Orchestrating DBMS Benchmarking in the Cloud with Kubernetes

(old, slightly outdated docs)