Bexhoma

Release 0.1

Patrick Erdelt

Sep 01, 2021

TABLE OF CONTENTS:

`{include} ../README.md`

This Python tools helps **managing benchmark experiments of Database Management Systems (DBMS) in a Kubernetes-based High-Performance-Computing (HPC) cluster environment**. It enables users to configure hardware / software setups for easily repeating tests over varying configurations.

It serves as the **orchestrator** [2] for distributed parallel benchmarking experiments in a Kubernetes Cloud. This has been tested at Amazon Web Services, Google Cloud, Microsoft Azure, IBM Cloud und Oracle Cloud and at Minikube installations.

The basic workflow is [1,2]: start a virtual machine, install monitoring software and a database management system, import data, run benchmarks (external tool) and shut down everything with a single command. A more advanced workflow is: Plan a sequence of such experiments, run plan as a batch and join results for comparison.

See the homepage and the documentation.

CHAPTER

ONE

INSTALLATION

1. Download the repository: https://github.com/Beuth-Erdelt/Benchmark-Experiment-Host-Manager
2. Install the package `pip install bexhoma`
3. Make sure you have a working `kubectl` installed (Also make sure to have access to a running Kubernetes cluster - for example Minikube)
4. Adjust configuration [tbd in detail]
    1. Rename `k8s-cluster.config` to `cluster.config`
    2. Set name of context, namespace and name of cluster in that file
5. Install data [tbd in detail] Example for TPC-H SF=1:
    - Run `kubectl create -f k8s/job-data-tpch-1.yml`
    - When job is done, clean up with `kubectl delete job -l app=bexhoma -l component=data-source` and `kubectl delete deployment -l app=bexhoma -l component=data-source`.
6. Install result folder Run `kubectl create -f k8s/pvc-bexhoma-results.yml`

CHAPTER TWO

QUICKSTART

The repository contains a tool for running TPC-H (reading) queries at MonetDB and PostgreSQL.

1. Run `tpch run -sf 1 -t 30`.
2. You can watch status using `bexperiments status` while running. This is equivalent to `python cluster.py status`.
3. After benchmarking has finished, run `bexperiments dashboard` to connect to a dashboard. You can open dashboard in browser at `http://localhost:8050`. This is equivalent to `python cluster.py dashboard` Alternatively you can open a Jupyter notebook at `http://localhost:8888`.

CHAPTER

THREE

MORE INFORMATIONS

For full power, use this tool as an orchestrator as in [2]. It also starts a monitoring container using Prometheus and a metrics collector container using cAdvisor. It also uses the Python package dbmsbenchmarker as query executor [2] and evaluator [1]. See the images folder for more details.

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

FOUR

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)