ISEngineering

Technische Universität Berlin

Enterprise Computing: Exercise 7 – Benchmarking II

Marco Peise, Stefan Tai

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Agenda

Exercise 7 Benchmarking II

– BenchFoundry

Technische Universität Berlin

Exercise 7

Task 1

Name four relevant reasons why benchmarks such as TPC, OLTPBench, YCSB or YCSB++ are not optimal for benchmarking cloud database systems or cloud database services.

- Strict functional and non-functional requirements on supported database systems and services (only by RDBMS but not NoSQL Systems)
- mimick application workloads realistically not testing isolated database features
- Multi-quality measurements
- Geo-distribution out of the box
- Fine-grained result collection

Task 2 BenchFoundry

Technische Universität Berlin

BenchFoundry: A Benchmarking Framework for Cloud Database Services

David Bermbach and Jörn Kuhlenkamp
Information Systems Engineering Research Group
TU Berlin
Berlin, Germany
Email: {db,jk}@ise.tu-berlin.de

Akon Dey
Awake Networks Inc.
Mountain View, California, USA
Email: akon@awakenetworks.com

***Abstract*—Understanding quality of cloud database systems and services is often crucial. However, current cloud database benchmarks either come without an implementation, have strong disadvantages, or can only be used with relational database systems and services.**

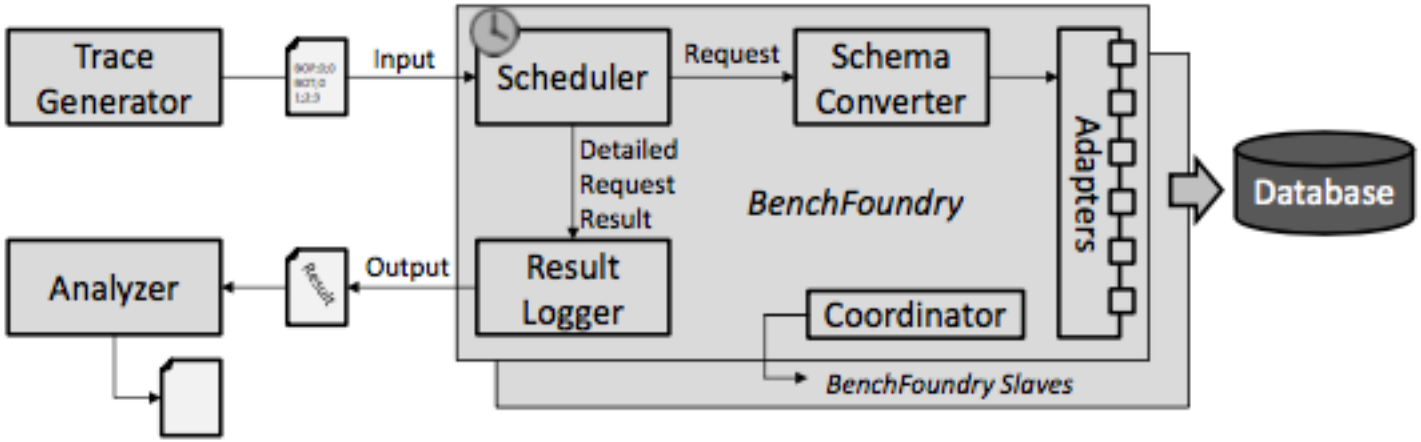
In this paper, we present BenchFoundry which is not a benchmark itself but rather is a benchmarking framework that can execute arbitrary application-driven benchmark workloads in a distributed deployment while measuring multiple qualities at the same time. BenchFoundry can be used or extended for every kind of OLTP database system or service. Specifically, BenchFoundry is the first system where workload specifications become mere configuration files instead of code. In our design, we have put special emphasis on ease-of-use

actually implemented. Other approaches, e.g., YCSB [2] or YCSB++ [3] are essentially micro-benchmarks. While these are useful for understanding how tiny changes in workloads affect system quality or for testing isolated database features, they are not a good fit for use cases such as selecting or optimizing the configuration of a database system since micro-benchmarks rarely mimick application workloads realistically. Other criteria where existing approaches are lacking are aspects like extensibility in terms of workloads, multi-quality measurements, (geo-)distribution support of the benchmark out of the box, fine-grained result collection, or ease-of-use. We will discuss requirements for modern benchmarks in detail and give an overview of related work in section 2.

References:

• https://github.com/dbermbach/BenchFoundry
• D. Bermbach, J. Kuhlenkamp, A. Dey "BenchFoundry: A Benchmarking Framework for Cloud Database Services “, to appear , 2017

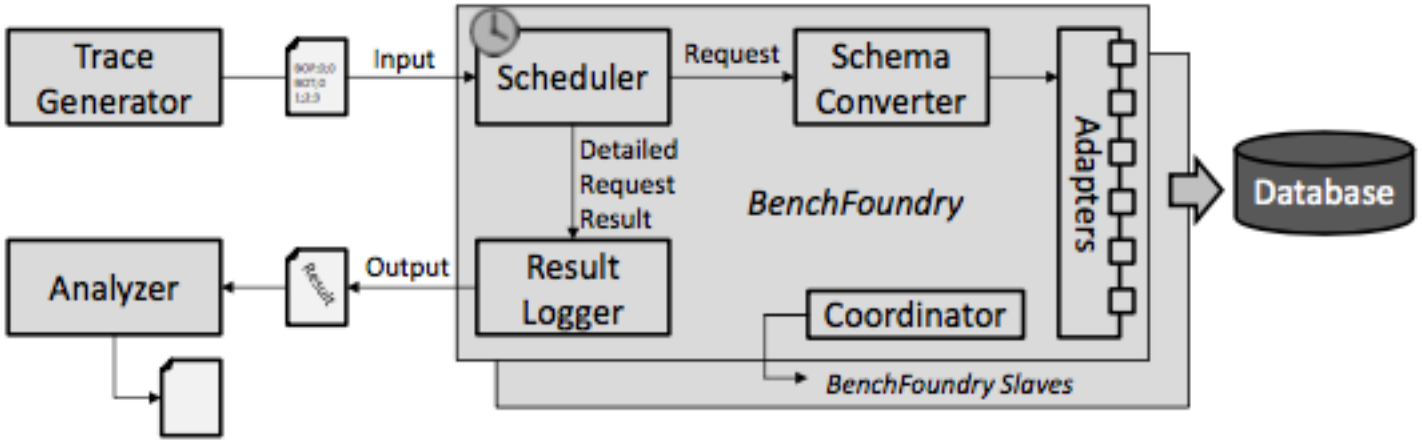
BenchFoundry



References:

- https://github.com/dbermbach/BenchFoundry
- D. Bermbach, J. Kuhlenkamp, A. Dey "BenchFoundry: A Benchmarking Framework for Cloud Database Services “, to appear , 2017

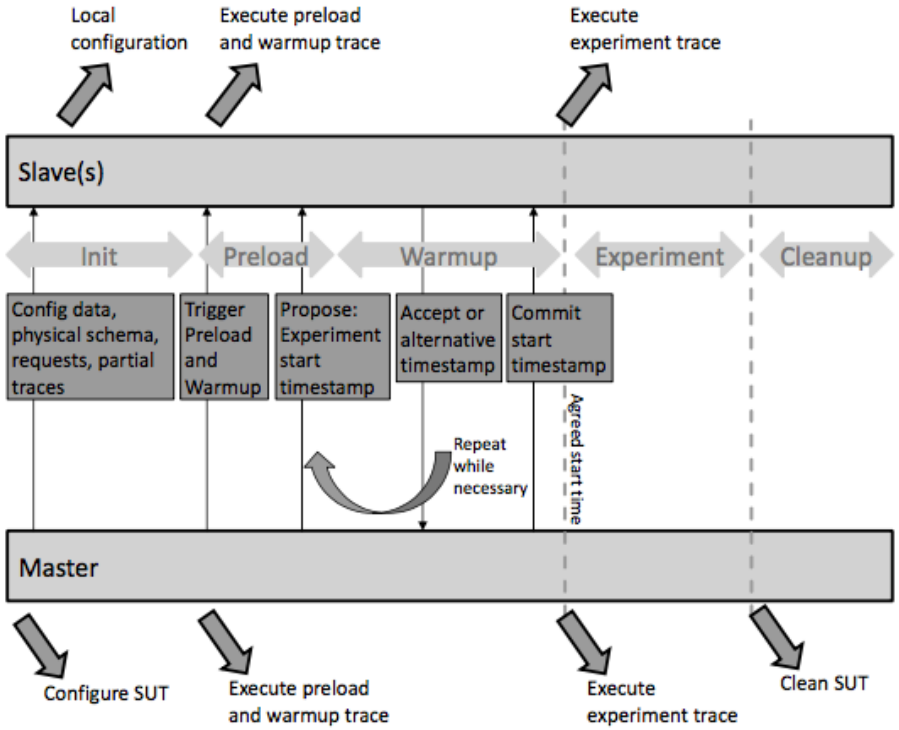
BenchFoundry



References:

- https://github.com/dbermbach/BenchFoundry
- D. Bermbach, J. Kuhlenkamp, A. Dey "BenchFoundry: A Benchmarking Framework for Cloud Database Services “, to appear , 2017

BenchFoundry

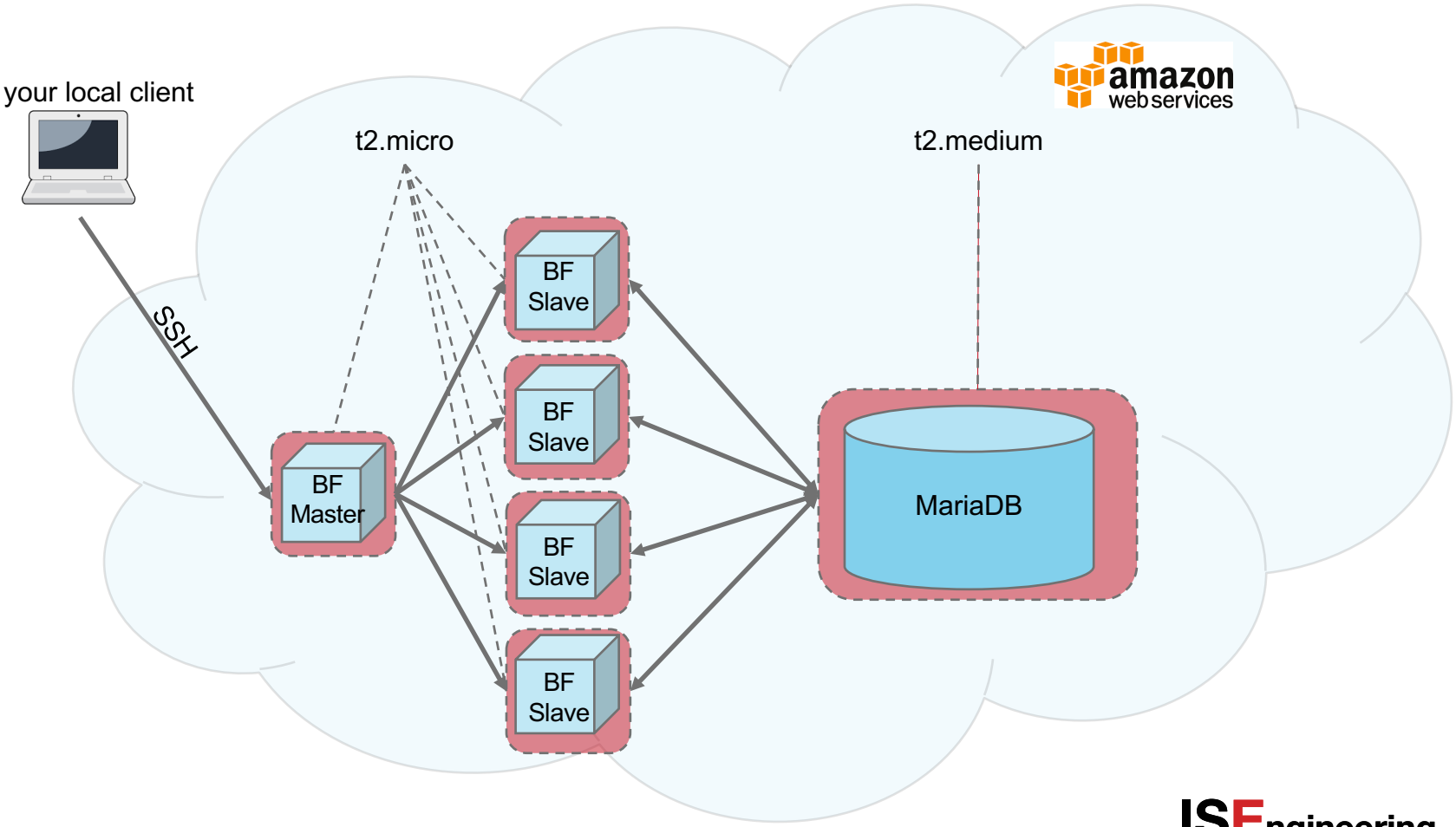


References:

- https://github.com/dbermbach/BenchFoundry
- D. Bermbach, J. Kuhlenkamp, A. Dey "BenchFoundry: A Benchmarking Framework for Cloud Database Services “, to appear , 2017

Technische Universität Berlin

Setup BenchFoundry with MariaDB



ISEngineering
Wirtschaftsinformatik – Information Systems Engineering

BenchFoundry

Technische Universität Berlin

install & configure 5 t2.micro instances

1. Create 5 t2.micro instances which are reachable from any TCP Port outside of AWS
2. Update the Operating System (sudo apt-get update, sudo apt-get upgrade)
3. OPTIONAL: Install Thrift compiler (sudo apt-get install thrift-compiler) if you want to build the original BF
4. OPTIONAL: Install Maven (sudo apt-get install maven) if you want to build the original BF
5. Install Java (sudo apt-get install default-jre, sudo apt-get install default-jdk)

download tar for current BenchFoundry on each mircro instance

Original from: https://github.com/dbermbach/BenchFoundry

Already build: https://github.com/marcopeise/BenchFoundry

On Master instance edit:

1. benchfoundry.properties and slaves.properties

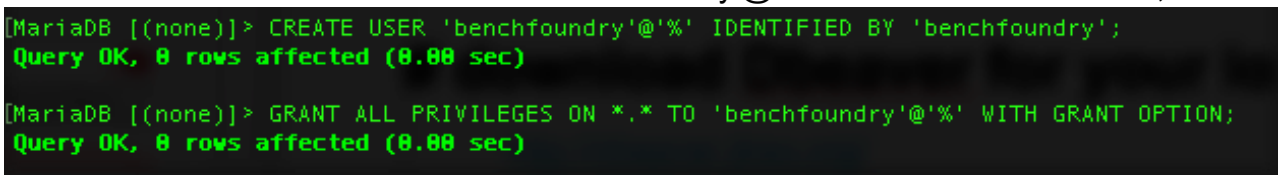
BenchFoundry

Technische Universität Berlin

install & configure MariaDB for your t2.medium instance

1. https://mariadb.com/ (sudo apt-get install mariadb-server)
2. Create User “benchfoundry” with password “benchfoundry” and grant privileges on the SQL command line:

CREATE USER 'benchfoundry'@'%' IDENTIFIED BY 'benchfoundry‘;
GRANT ALL PRIVILEGES ON *.* TO 'benchfoundry'@'%' WITH GRANT OPTION;



3. Create a Database called “test” with the SQL command line
4. Make DB available to the clients (/etc/mysql/mariadb.conf.d/50-server.cnf -> comment out bind-address)

download Dbeaver for your local client

http://dbeaver.jkiss.org/

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

BenchFoundry

Technische Universität Berlin

run Master & Slave instances

On **each Slave** start BenchFoundry:

java -jar target/BenchFoundry-1.0-SNAPSHOT-jar-with-dependencies.jar <port>

(should correlate with slaves.properties on Master Instance)

On the **Master** instance start BenchFoundry:

java -jar target/BenchFoundry-1.0-SNAPSHOT-jar-with-dependencies.jar benchfoundry.properties

analyse the outcome