UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Faculty for Computer Science, Electrical Engineering and Mathematics
Department of Computer Science
Research Group DICE Group

Bachelor's Thesis Proposal

Submitted to the DICE Group Research Group
in Partial Fullfilment of the Requirements for the Degree of

Bachelor of Science

Basilisk – Continuous Benchmarking for Triplestores

by
Fabian Rensing

Thesis Supervisor:
Prof. Dr. Axel-Cyrille Ngonga Ngomo

Paderborn, November 3, 2021

1 Description and Motivation

Triplestores are the database backend of knowledge graphs. They are needed to store and query data of knowledge graphs used in the Semantic Web. Since knowledge graphs can contain huge amount of data and can also be subject to many changes, Triplestores need to be able to handle many different workloads. To better test and compare Triplestores in these diverse scenarios, benchmarks are performed to allow an appropriate comparison between different Triplestores. Often Triplestores are developed in long iterations and are bench-marked only in a late stage of such an development iteration. Today benchmarks and the evaluation of their results are usually done manually and bind developers time. Thus, performance regressions are found very late or never.

Benchmarks in general are used to measure and compare the performance of computer programs and systems with a defined set of operations. Often they are designed to mimic and reproduce a particular type of workload to the system. In the context of Triplestores, a benchmark usually consists of creating a big knowledge graph and performing multiple queries and operations on the data.

Several benchmarks for Triplestores have been proposed. IGUANA is a benchmark-independent execution framework [1] that can measure the performance of Triplestores under several parallel query request. Currently the benchmark execution framework needs to be installed and benchmarks need to be started manually. Basilisk is a continuous benchmarking service for Triplestores which internally uses IGUANA to perform the benchmarks. The idea is that the Basilisk service will check automatically for new versions of Triplestores and start benchmarks with the IGUANA framework. Further it should be possible to start custom benchmarks on demand. If a new version is found in a provided GitHub- or DockerHub-repository, Basilisk will automatically setup a benchmark environment and starts a benchmarking suite.

This means that developers do not have to worry about performing benchmarks at different stages of development.

—

- What general topic are we interested in - why is this topic interesting in general - where is it used - why does it matter - who implements it - what problem does it solve?

- What is your personal motivation to deal with this topic? - Which interesting problems do you expect?

Explain some more background for the topic, help the reader, who may have never heard of the topic, understand what you are talking about.

Current state of the research: - What is a trivial way to solve x? - How is x usually solved in practice? - what are typical ingredients / techniques to construct the systems we are interested

in? - How does ? solve the problem? - What does the algo in ? do roughly?

The topic I have to offer has the working title is "Basilisk – Continuous Benchmarking for Triplestores". At the core, it is developing and deploying a CI/CD tool that hooks into github and/or docker registries.

Described in more detail:

Triplestores – the database backend of knowledge graphs – are typically developed in long iterations and are bench-marked – if at all – only in a very late stage of such an iteration. Benchmarking and evaluation of benchmarking results are typically done manually and binds developer's time. Thus, performance regressions are found very late or never.

With Basilisk we started to develop a continuous benchmarking platform for triplestore which hooks into github and docker image registries.

On events like pull requests or newly published versions of triplestores, a benchmarking suite is run automatically.

The first version of Basilisk (https://github.com/dice-group/Basilisk, https://github.com/dice-group/basilisk-frontend) is already implemented. It is based on the benchmarking tool IGUANA (https://github.com/dice-group/IGUANA) and Docker. (It requires triple stores to be dockerized).

The thesis task is to:

The thesis can be extended to a paper in a scientific journal (e.g. ISWC) on the resource track with you as first author.

2 Formulation of Target Setting

The target of this thesis is to describe the software architecture, deploy the system and perform some benchmarks of different triple stores.

The software architecture of Basilisk was developed through a master thesis beforehand.
and review the software architecture that has been used for the Basilisk platform.
—

Goals of the thesis: - Whats the problem with existing solutions? with the stuff you explained in the motivation - What's the goal of the thesis in a nutshell? solve that problem / analyze the solution / compare potential solutions / ... - Then go into detail. What are the specifics of what you are going to do? Explain intermediate goals - Explain how your evaluation will look like. Describe your test environment. - Provide optional goals

Evaluation - was macht basilisk was vorher nicht da war - versch systeme mit selben zweck vergleichen - suche? eigentlich niht verfügbar - richtung: einfaches seetting - benchmark - 3 triplestores - wie lange mit system / Ohne system - schnell im basilisk zw benchmarks wechseln - im voraus planen was zu testen

3

Tasks Descriptions

1. State of the art review?
2. software architecture analysis
3. Describe and review software architecture
4. Solution design
   How to perform the missing implementations
5. Solution implementation
6. Deploy Basilisk and its frontend
7. Perform benchmarks of requested Triplestores
   - 2 versions of Tentris
   - one version of another Triplestore
8. Fix critical bugs, document non-critical
9. Thesis writing

—

1. describe and review the software architecture 2. deploy Basilisk and its frontend on a publicly available VM 3. benchmark 2 versions of Tentris (https://github.com/dice-group/tentris, via a github hook) and one version from another triple store (via a docker image registry, e.g. https://hub.docker.com/r/ontotext/graphdb/ or https://hub.docker.com/r/openlink/virtuoso-opensource-7). 4. fix critical bugs in 1.-3. and document non-critical

4

Schedule

—

21 weeks of work Gantt chart: pgfgantt package

5

Preliminary Outline of the Thesis

The thesis will comprise the following chapters

1. Introduction:
    - Description of the topic and the context in which it is inserted
    - Topic motivation
2. Related Work
    - Description of existing techniques and approaches
3. Background
    - Description of foundational knowledge required to understand the thesis
4. Approach
    - Description of the designed system
5. Implementation
    - Implementations details
6. Evaluation
    - Experiment setup, requirements and results
7. Summary and Discussion
    - Summary of the work
    - Highlighting the key findings of the evaluation stage

Bibliography

[1] Felix Conrads, Jens Lehmann, Muhammad Saleem, Mohamed Morsey, and Axel-Cyrille Ngonga Ngomo. Iguana: A generic framework for benchmarking the read-write performance of triple stores. In Claudia d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange, and Jeff Heflin, editors, *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*, volume 10588 of *Lecture Notes in Computer Science*, pages 48–65. Springer, 2017.