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 29, 2021

1 Description and Motivation

In the field of Semantic Web, knowledge graphs are an important structure to represent data and its relationships. To easily store and query the data in these knowledge graphs, some data structure or database is needed. The special kind of database developed to store knowledge graphs are called Triplestores.

Since knowledge graphs can contain huge amounts of data which can also be subject to many changes, Triplestores need to be able to handle many different workloads. Some scenarios need to handle huge amount of data being added, while others need to handle a lot of changes on the current data. To better test and compare Triplestores in these diverse scenarios, benchmarks are performed to allow an appropriate comparison between different Triplestores[8].

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 given knowledge graph on which multiple queries and operations are performed.

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.

Several benchmarks for Triplestores have been proposed [8]. IGUANA is a benchmark-independent execution framework [4] 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.

2 Formulation of Target Setting

The target of this thesis is to setup and deploy an instance of the Basilisk service platform. Since Basilisk is not yet fully developed, the missing parts will be designed and implemented first, before the system can be deployed publicly.

The platform will be deployed on a virtual machine provided by the IRB-department of the Paderborn University. The hardware requirements are 8-16 CPU-cores, 256GB of RAM, and 4TB of Storage for different datasets and benchmarks.

Different benchmarks will be performed on the Triplestores under test. Synthetic benchmarks will be generated with WatDiv[2]. FEASIBLE[7] will be used to generate benchmarks from real-world datasets of varied sizes and structures. Real-world datasets that will be used are a subset of the WikiData and DBpedia. For these datasets there are query logs contained in LSQ[6] which will be provided to the FEASIBLE framework. Other benchmarks that will be available on the platform are LUBM[5] and SP²Bench[9].

The metrics that are calculated after each benchmark, are the ones provided by the IGUANA framework [1][4], since the IGUANA framework is used in the Basilisk platform to perform the benchmarks. These metrics provided can be seen in table 2.1:

Name	Description
NoQ	number of successfully executed Queries
QMPH	number of executed Query Mixes per Hour
NoQPH	NoQ per hour
QPS	queries per second
penalized QPS	failed queries get a penalty that is included in this metric
AvgQPS	average QPS
penalizedAvgQPS	AvgQPS with penalty included

Table 2.1: Metric provided by the IGUANA framework[1].

At least two Triplestores will be evaluated with the framework of which one should be Tentris[3]. The goal of this use-case is to assess how well the system helps automating the execution and evaluation of benchmarks. This includes examining which steps could be further automated and which interfaces could be added to get specific data.

If unforeseen problems arise, these will be documented and evaluated, to learn for further attempts.

3 Tasks Descriptions

The enumerated tasks for the thesis are defined here and correspond to the schedule shown in chapter 4.

Task 1	Software architecture analysis and review
Time Required	3 Weeks
Description	Analysis and review of the current software architecture for the Basilisk service.

Task 2	Solution Design
Time Required	3 Weeks
Description	The missing parts of the implementation for the Basilisk service will be designed and planned out.

Task 3	Solution Implementation
Time Required	4 Weeks
Description	The previously designed solution will be implemented into the existing architecture.

Task 4	Deploy Basilisk and its frontend
Time Required	2 Weeks
Description	The Basilisk service and the frontend will be deployed on a publicly available VM.

Task 5	Perform benchmarks for Triplestores
Time Required	3 Weeks
Description	Benchmarks for two versions of the Tentris Triplestore[3] and one different Triplestore will be benchmarked.

Task 6	Fix critical bugs, document non-critical
Time Required	2 Weeks
Description	Critical Bugs and errors found in Tasks 1-5 will be fixed and non-critical will be documented.

Task 7	Thesis writing
Time Required	19 Weeks
Description	The thesis will be written parallel to the work on the first Tasks.

4 Schedule

The Schedule for the proposed task in Chapter 3 is shown in figure 4.1. The last point "Buffer" is meant as a buffer for unforeseeable events that could alter the planned time schedule.

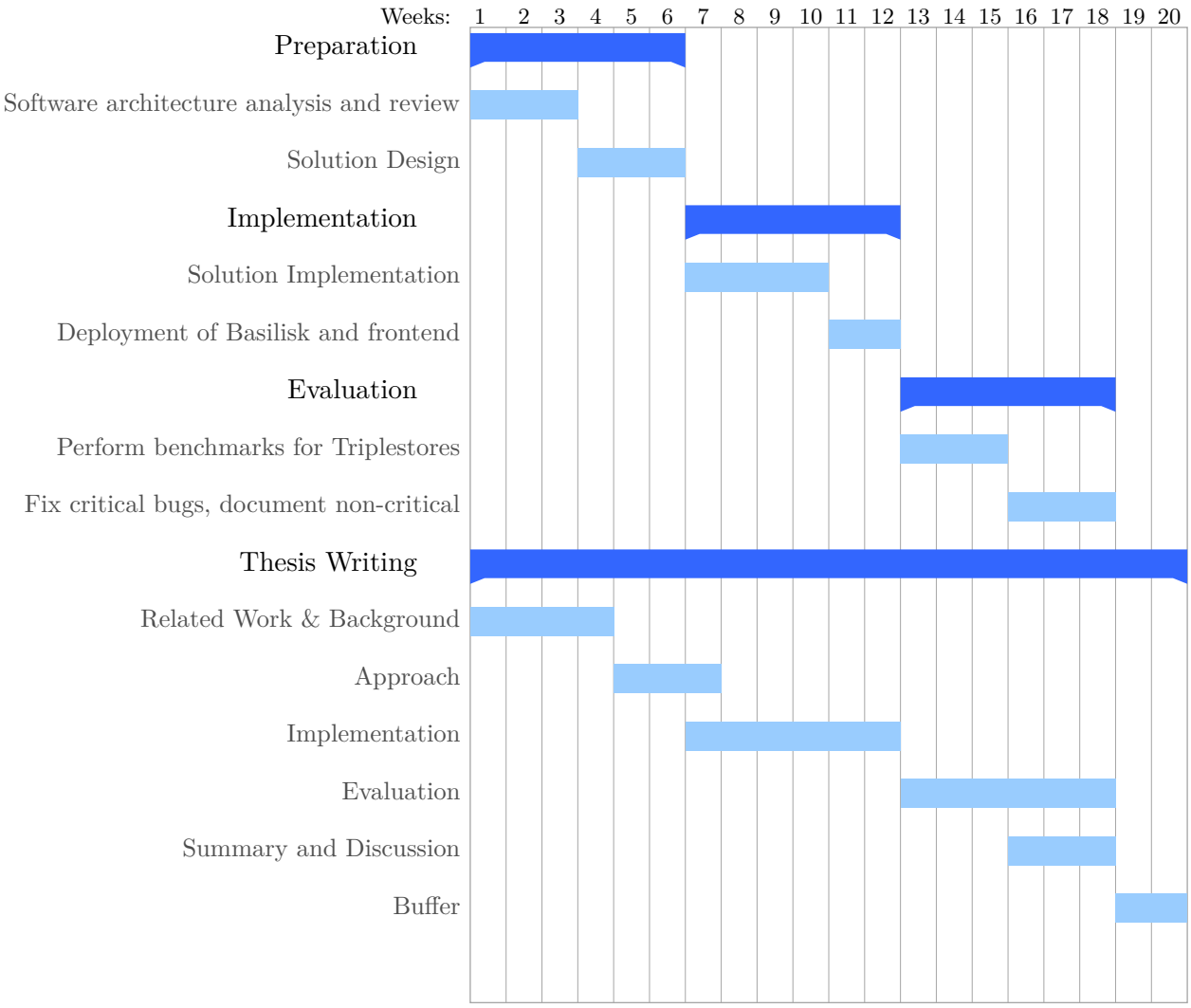

Figure 4.1: Proposed schedule for the work on the thesis.

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
    - Analysis and review of the current software architecture
5. Implementation
    - Solution design and implementation
    - Deployment of the service
6. Evaluation
    - Experiment setup, requirements
    - Performing of benchmarks
    - Result evaluation
7. Summary and Discussion
    - Summary of the work
    - Highlighting the key findings of the evaluation stage

Bibliography

[1] Metrics - Iguana 3.3 Documentation, http://iguana-benchmark.eu/docs/3.3/usage/metrics/.

[2] Güneş Aluç, Olaf Hartig, M. Tamer Özsu, and Khuzaima Daudjee. Diversified Stress Testing of RDF Data Management Systems. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandečić, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble, editors, The Semantic Web – ISWC 2014, Lecture Notes in Computer Science, pages 197–212. Springer International Publishing.

[3] Alexander Bigerl, Felix Conrads, Charlotte Behning, Mohamed Ahmed Sherif, Muhammad Saleem, and Axel-Cyrille Ngonga Ngomo. Tentris – A Tensor-Based Triple Store. In Jeff Z. Pan, Valentina Tamma, Claudia d' Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana Kagal, editors, The Semantic Web – ISWC 2020, volume 12506 of Lecture Notes in Computer Science, pages 56–73. Springer International Publishing.

[4] 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 Fernandez, Valentina Tamma, Freddy Lecue, Philippe Cudré-Mauroux, Juan Sequeda, Christoph Lange, and Jeff Heflin, editors, The Semantic Web – ISWC 2017, volume 10588 of Lecture Notes in Computer Science, pages 48–65. Springer International Publishing.

[5] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. 3(2):158–182.

[6] Muhammad Saleem, Muhammad Intizar Ali, Aidan Hogan, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. LSQ: The Linked SPARQL Queries Dataset. In Marcelo Arenas, Oscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d' Aquin, Kavitha Srinivas, Paul Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Steffen Staab, editors, The Semantic Web - ISWC 2015, Lecture Notes in Computer Science, pages 261–269. Springer International Publishing.

[7] Muhammad Saleem, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. FEASIBLE: A Feature-Based SPARQL Benchmark Generation Framework. In Marcelo Arenas, Oscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d' Aquin, Kavitha Srinivas, Paul Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, Krishnaprasad Thirunarayan, and Steffen Staab, editors, The Semantic Web - ISWC 2015, Lecture Notes in Computer Science, pages 52–69. Springer International Publishing.

[8] Muhammad Saleem, Gábor Szárnyas, Felix Conrads, Syed Ahmad Chan Bukhari, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. How Representative Is a SPARQL Benchmark?

an Analysis of RDF Triplestore Benchmarks. In The World Wide Web Conference, pages 1623–1633. ACM.

[9] Michael Schmidt, Thomas Hornung, Georg Lausen, and Christoph Pinkel. SP2Bench: A SPARQL Performance Benchmark.