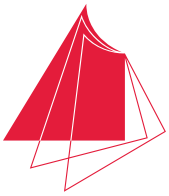

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Benchmarking, analysis, and optimization of Python-based graph library performance

Bachelor Thesis von

Fabian Sorn

an der Fakultät für Informatik und Wirtschaftsinformatik
Fachrichtung Verteilte Systeme (VSYS)

Erstgutachter:	Prof. Dr. rer. nat. Christian Zirpins
Zweitgutachter:	Prof. B
Zweiter Betreuer:	Dipl.-Inform. D

10. November 2019 – 10. March 2020

Hochschule Karlsruhe Technik und Wirtschaft
Fakultät für Informatik und Wirtschaftsinformatik
Moltkestr. 30
76133 Karlsruhe

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Änderungen entnommen wurde.

PLACE, DATE

. . . . . .

(Fabian Sorn)

Zusammenfassung

Deutsche Zusammenfassung

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Introduction

The following chapter will provide an introduction into this work. First the setting in which the work is done, will be described, starting with the orgranisation followed by the team. Afterwards the fundamental problem and the goal of this work will be explained, rounded up by an overview about the structure of the following chapters.

1.1. Motivation

The big advantage that the raise of computing brought with it, was, that complex mathematical tasks, could be performed in very little time. Since the beginnings, a lot has happened and computers got much more powerful. Even with these improvements, the question of good performance could not be more relevant as today. We have to make sure, that the hardware and our algorithms are fast enough, to keep up with the tasks we we want to accomplish. This demand is especially relevant in the scientific world, where often gigantic data sets have to be filtered, recorded and analyzed. A popular tool for such work are software products, that allow us to visualize data as graphs, since visualization allows us to have a much deeper insight into the data we want to understand. As with any type of software, the performance of graphs has to keep up with our high demands. One of the places, where this couldn't become more clear, is CERN, where the fundamental question the following work is based on, was researched.

1.2. CERN

The European Organization for Nuclear Research (CERN) is one of the biggest and most well known research facilities in the world. It is most known as the host of one of the world most complex and astonishing machines, the Large Hadron Collider (LHC) as well as the birth place of the World Wide Web (WWW), on which we rely on daily everywhere around the world. [2] These and many more achievements and projects all contribute to the central mission at CERN: Finding out, what our universe is made of. To find answers to this question, CERN brings together over 17500 people from all over the world, to work together in many different fields including physics, engineering, computer science and more. Today, CERN counts 23 member states that collaborate on decisions made at in the organisation every day [5].

CERN's roots can be traced back to the 1940's, when a hand full of scientists saw the needs for Europe to advance its role in the scientific world by hosting its own research facility for physics. Starting with 12 original member states, CERN originally was founded based on this vision in 1954 located at the franco-swiss border, as the *Conseil Européen*

pour la Recherche Nucléaire, leading to today's well recognized acronym CERN. Until today, these member states are contributing to CERN's financing, organisation and foundations to achieve its goals to expand the boundaries of human knowledge. [3]

1.3. Beams Controls Applications

CERN's main focus for research is particle physics. To continuously improve our knowledge in this field, CERN is operating the world's most powerful particle accelerator called LHC. The LHC allows us to gain a much deeper insight into the subatomic structure of the world around us bringing us closer to understanding the inner workings of our universe. CERN itself is devided into different departments which have their own purposes. The Beams Department (BE) is responsible for developing software and hardware instruments for the accelerator complex. [1]

The LHC's task is to produce and accelerate two beams of charged particles, travelling in opposite directions. To achieve this,it is constructed as two circular pipes containing a vacuum, which are surrounded by magnets. The magnetic field created by these magnets can accelerate and steer particles passing by. If a beam of particles is injected into the accelerator, the strength of the magnetic field is increased with every round the beam travels in the accelerator, until the beam reaches speeds very close to the speed of light. Is this level of energy reached, the next step is to make particles from the two beams collide with each other. CERN is operating four experiments, Atlas, CMS, Alice and LHCb, where the particles can be led to collission. The particles detectors then can record the results of the collisions in great detail for later analysis. The operation of the LHC is under one roof, the Cern Control Center (CCC). [4, 7]

Particle accelerators are set up from many different components, ranging from power converters that provide energy to the magnets to instruments responsible for monitoring all metrics describing the state of the beam. To operate all these different parts, a control system is necessary, which allows operators to change settings of components and monitor the resulting behaviour. The development of this control system for the accelerator complex is done by the Controls Group (BE-CO) which is part of BE. [6] The control system itself is composed of different components that work together. Responsible for these components are different sections within the controls group. This work has been conducted in the Applications Section (BE-CO-APS), whose responsibility it is, to provide software solutions for the many tasks of the control system. One of these products are Graphical User Interfaces (GUIs), that are vital tools for the operators' work. A Graphical User Interface (GUI) application allow to monitor the current state of the machine and react to occuring problems by altering the setting of these machines. To develop such monitoring applications, BE-CO-APS is providing different reusable GUI widgets, from which more complex monitoring applications can be developed.

1.4. Problem

A recent decision of BE-CO-APS at CERN was, to move from Java to Python for GUIs. This decision opens the door for many people, which aren't primarily software developers, to write their own GUI applications for their specific use cases. The framework of widgets which BE-CO-APS is providing, does also contain graph components, that allow operators to visualize data in their GUIs.

Choosing a library to implement graph widgets is not a trivial topic. Especially in python, there are many charting libraries, from which you can choose from, including matplotlib, Seaborn, PyQtGraph, Plotly and more [8]. The comparison of offered features is in most cases not enough. Many users have specific needs and use cases, which rely heavily on the performance of the library. Compared to the evaluation of needed features and the offerings in libraries, evaluating the performance is not just a decision between *is available* and *is not available*. To answer the question, if a library is fast enough for a specific use case, we have to provide metrics, realistic use cases and a reliable way of testing the performance, that allow us to take a sound decision.

Benchmarking is a good way of answering such performance questions. Most known in these cases are benchmarks, which allow us to compare the speed of different hardware components, by running the same sequence operations on them and comparing the times, that they required to complete these tasks [9]. To compare the perfromance of different implementations of software, we can utilize a very similar approach. Instead of running the same code on different hardware, we can run different implementations of the same tasks on the same hardware and measure each's performance. For more complex operations, like visualizing data, this inevitably raises the question, how we can implement such a measurment and what metrics describe the libraries performance.

Benchmarking for different implementations would not only allow us to compare the same high level operations between different libraries, but also the development of certain operations in the same library over time. For the user, such a benchmarking possibility would also make a decision between libraries much easier, since he could actively test his demands and use cases on different libraries to find the library that fits his performance needs the best.

1.5. Planned Solution

Goal of our work is, to develop a benchmarking framework for python graph libraries. The benchmark framework will be used to develop a suite of benchmarks, that can be used to verify the performance of a graph library written in Python in real world use cases. The benchmark suite should not only give use comparable results to objectively judge performance, but also help us to find and improve slow operations. Afterwards, we will implement potential improvements for found performance deficits. To evaluate our framework, we will run the same benchmarks, but now based on our changed implementation and compare the results to the original implementation.

1.6. Structure of this work

In the beginning of this work, we will create a common knowledge base that is necessary to understand the following chapters, by going through the basics of data visualization. To understand thechnical decisions we took and implementations of our solution, we will have a look at the technologies we will use. The next chapter will deal with real use cases, that users of charting libraries at CERN have provided. From those we will derive metrics, which we can use for our later implementation. Following that, we will have a look at the basics of benchmarking as well as already existing benchmarking solutions, from which we can derive concepts, we can use for our own implementation. The next chapter then guides through the design and implementation of a benchmark framework and benchmarking suite, which allows us to run our graph library of choice against the collected use cases. Following that, we can use the benchmarks detailed results, to plan ways to remove performance bottle-necks in the implementation. Finally the exact same benchmarks will be run again, to objectively judge, which impact our changes had on the tested operations.

2. Fundamentals

This chapter will give an introduction into all major topics this work is based on. In the beginning we will have a look at Data Visualization and charting in general

2.1. Charting

2.1.1. Data Visualization Pipeline

…

2.2. Charting at CERN

2.2.1. Monitoring Applications

…

2.2.2. JDataViewer

…

2.3. Benchmarking

…

2.4. Python

…

2.5. Qt

2.5.1. Qt Basics

…

2.5.2. Event Loop

…

2.5.3. Signals and Slots

...

2.5.4. Python bindings

...

2.6. PyQtGraph

2.6.1. Fundamentals

...

2.6.2. AccPyQtGraph

...

3. Use Cases

This chapter will guide us through selected use cases that we can use to find metrics to evaluate performance.

3.1. Line Charts in section BE-CO-HT

…

3.2. Line Charts in section BE-CO-LHC

…

3.3. Use case in Linac4 Source GUI

…

3.4. Metrics from use cases

…

4. Design and Implementation of a Benchmark Framework

In this chapter, the findings from the previous chapters will be combined to design and implement a benchmarking framework, which allows the user to benchmark charting operations, he has defined himself.

4.1. Design

4.1.1. Architecture

…

4.1.2. Redraw Mechanism

…

4.1.3. Profiling

…

4.2. Implementation

4.2.1. Implementation of the Framework

…

4.2.2. Implementation of the Use Cases

…

5. Performance optimization

5.1. Optimizing line graph performance

5.1.1. GPU accellerated Rendering

...

5.2. Optimizing scatter plot performance

5.2.1. Incremental data updates

...

6. Evaluation

…

6.1. Benchmark changes to Line Graph

…

6.2. Benchmark changes to Scatter Plot

…

7. Conclusion

7.1. Summary

…

7.2. Outlook

…

Literatur

[1] CERN. *Beams Department*. URL: https://beams.web.cern.ch.

[2] CERN. *Home*. URL: https://home.cern.

[3] CERN. *Our history*. URL: https://home.cern/about/who-we-are/our-history.

[4] CERN. *The Large Hadron Collider*. URL: https://home.cern/science/accelerators/large-hadron-collider.

[5] CERN. *Who we are*. URL: https://home.cern/about/who-we-are.

[6] Stéphane Deghaye und Eve Fortescue-Beck. *Introduction to the BE-CO Control System*. 2019 Edition. CERN, 2019.

[7] Ben Dotson. *How Particle Accelerators Work*. 2014. URL: https://www.energy.gov/articles/how-particle-accelerators-work.

[8] Quincy Smith. *The Best Python Data Visualization Libraries*. Nov. 2019. URL: https://www.fusioncharts.com/blog/best-python-data-visualization-libraries/.

[9] Reinhold P. Weicker. "An overview of common benchmarks". In: *Computer* 23.12 (1990), S. 65–75. DOI: 10.1109/2.62094.

A. Anhang

A.1. First Appendix Section

Abbildung A.1.: A figure

...