Optimization of server solution and performance measurement

Jakub Trmal, Jindřich Prokop, Daniel Novák

Faculty of Electrical Engineering, Czech Technical University in Prague

Motivation

In our case, we have a server-side web application which was created to provide a **virtual therapist** for an addiction treatment plan. This application uses a framework prepared as a platform to build a universal web application based on **logic-driven** concept. In the early phase of the project, evolution has this approach significant benefits and stable framework also provide robust debug environment. In later, when the project becomes more stable with verified functionality, we can focus on huge optimization potential.

Assignment and Requirements

The ambition of this work is to find a particular working implementation of optimization techniques, which guarantee runtime of memory consumption improvement. It is also desirable to update the project with the latest technologies and also try to enhance deployment processes which make the application more flexible, portable and easy to update.

Analysis

The fundamental part of optimization is thoroughgoing and complex analysis to help us understand every process of application concept. Our analysis is focused on two levels:

Design level

On this level, we look at the application as a complex, and we try to detect possible weak points such as actions with no effect, repeated procedures or dead ends. We go through particular use cases usually discuss these steps with the product owner.

Source code level

This level is a more technical and specific level where the analysis goes deeper in algorithms, data structures and try to find feasible improvements. Includes performance analysis, memory analysis and think over algorithms.

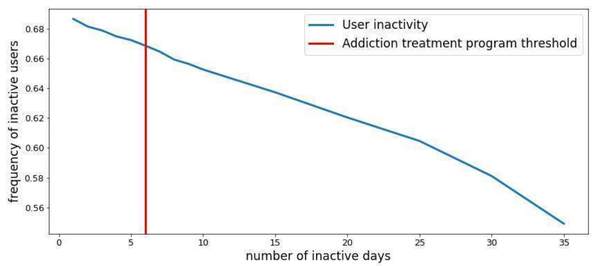

Figure 1. User activity trend in time with addiction treatment program threshold.

Optimization techniques

In figure 1 we can see result of user activity analysis based on addiction treatment program design [1] which results in this model. If the user is inactive more than 7 days we can remove the user from the program, which can save large amout of computational sources.

The main idea behind appropriate technique selection is to think about the way how to convert general models and processes to fit a more specific use case.

The application core consist of multiple components includes notification module, expression parser, decision tree traversal algorithm or worker queue. From the shallow analysis, we can infer the worker queue as one of the most significant crucial parts.

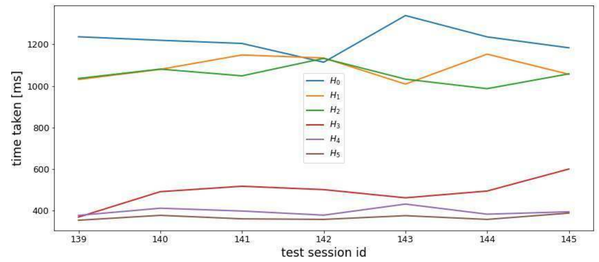

Figure 2. Computational time reduction on each optimization.

In figure 2 we can notice computational time save after particular optimization is applied. It is clear to see that one of the greatest improvement is an update from H_2 to H_3, which represent internal **instances reduction**. This technique helps us to obtain up to double speedup application.

Another optimization was the reduction of communication with database module, general improvement of database queries or better usage of database indexing. Other presented optimization are listed below.

From the developer's point of view, the important advancement is the implementation of automated CI/CD and configuration of multiple environments for deployment.

Used Optimizations

H_0 - application without optimization
H_1 - add slots in data holding classes
H_2 - optimize expression parser and evaluator
H_3 - reduction number of internal instances per request
H_4 - interaction with database improvement
H_5 - optimize core engine internal iterations

Conclusion

Combining all the optimization methods helps us to reach local speedup `4.3`. The graph 3 shows optimization techniques impact in computational time on each of few test sessions.

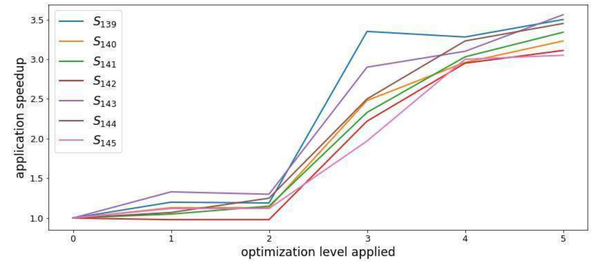

Figure 3. Speedup gained on application of particular optimization level.

The overall optimization process was a point at speed up the computational part of the application, but it ensures locally decrease memory consumption due to better data structures selection.

Last but not least, the analysis shows us another potential in the communication part, where we can save much in transferred data by simple reduction as present the table below.

Page name	size	size reduced	gzip
`E01.5a`	32 kB	5.9 kB	2.5 kB
`EE01.2`	23 kB	4.1 kB	1.9 kB

Table 1. A table caption.

Outcome Highlights

- Main job computational time reduced from 56 minutes to approximately 4 minutes
- Proper setup of CI/CD results in flexible and more reliable deployment process
- Containerization ensure better scalability as well as cost-effective managment

References

[1] A Kulhánek, R Gabrhelík, D Novák, V Burda, and H Brendryen. ehealth intervention for smoking cessation for czech tobacco smokers: Pilot study of user acceptance. pages 81–85, 2018.