DD2482 - Automated Software Testing and DevOps

Essay Assignment

The Purpose and Advantages of Adding Scalability Testing to Your CI/CD Pipeline

KTH Royal Institute of Technology

Einar Oddur Pall Runarsson
eopru@kth.se

May 2023

Contents

I certify that generative AI, incl. ChatGPT, has not been used to write this essay. Using generative AI without permission is considered academic misconduct.

1 Introduction

This essay is about the purpose and advantages that come along with setting up scalability testing as a part of the testing stage in your CI/CD pipeline.

Imagine being in a start-up that is doing good. It has started to grow rapidly, the users sign up by the thousands each day and you start to worry about how the system will handle this magnitude of scaling up that is happening. Is it going to start crashing? Is it going to slow down significantly affecting the user's experience dramatically? Endless questions race through your mind as you start looking for answers. That is when you find out about scalability testing. Why wait to find out the hard way when you can simply test for it by creating simulated traffic and measuring the results?

1.1 Software Testing

When developing software it is crucial to make sure everything works as expected, and that is what software testing enables. Software testing can help to uncover errors or faults in an application, it can reduce costs of development, and result in an overall improvement of performance. To maximize these wanted attributes there are multiple types of testing, for instance, unit testing, integration testing, acceptance testing, usability testing, and so many more [1].

Traditionally, software testing was a separate practice from development. It was not until the later stages that the almost finished product was tested and bugs were found shortly before the release. This caused products to often be released with bugs that the development team did not have time to fix. As the development traditions advanced through the years from this traditional waterfall methodology towards a more agile environment, testing of software was conducted earlier. This leads teams to discover bugs sooner rather than later, making it easier and less expensive to fix since other stuff has not been stacked upon it. As traditions develop the DevOps approach is introduced. What that meant for testing was continuousness [1].

1.2 Continous Testing

As mentioned in the previous subsection, along with DevOps came continuous testing. Products were no longer tested just before release or after manually building each version, but rather automatically and continuously throughout the lifespan of the product! That is why continuous testing is considered a crucial driver in the effectiveness of continuous integration and continuous development. If the continuous tests fail the development team is notified and fixes can be made to the source code before the error has an impact elsewhere. When the

changes have been applied, the source code gets reintegrated and the testing starts again. If the tests pass the next stage of the DevOps cycle is reached, which would be continuous delivery or "release" as shown on figure 1 [2].

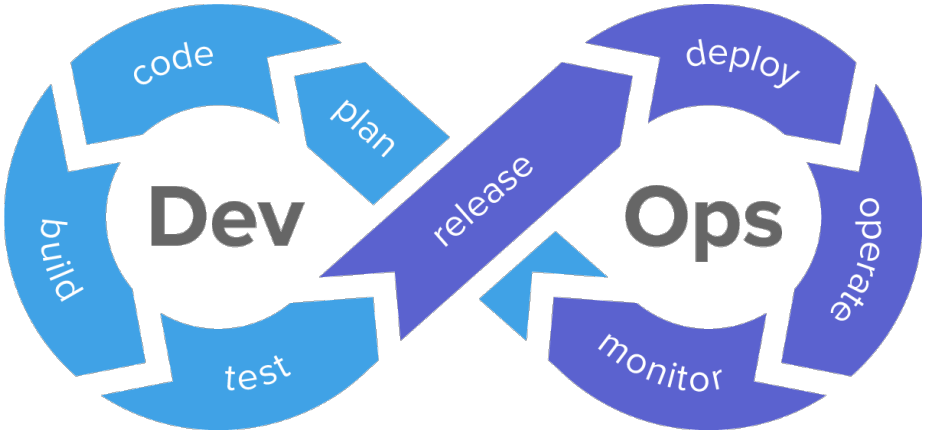

Figure 1: DevOps lifecycle phases
source: https://devopspage.com/what-is-ci-cd-in-devops-world/

1.3 Scalability

The word scalability in the context of software means how well a system adapts its capacity by increasing or decreasing its consumption of resources. The capacity of a system is the maximum workload it can process while satisfying a performance goal. A system that is scalable has three qualities: being able to handle an increase in usage, an increase in data, and being maintainable with good performance. Performance and scalability are not to be confused with each other. Performance is the measure of speed and how efficiently the system completes a task, while scalability is the measure of the trend of performance with growing/shrinking load [3] [4].

When it comes to scaling, two methods are important to explain, vertical scaling and horizontal scaling. Vertical scaling, or scaling up, is the method of increasing the resources of a single component to handle the load. This can range from upgrading the physical hardware of a machine to optimizing algorithms in a source code. This method has its advantages and disadvantages, however, it is worth noting that the cost grows exponentially here due to the curve of cost behind computational processing, and that having only one highly optimized and powerful machine does not guarantee availability as it is a single point of failure. The other method, horizontal scaling, or scaling out, is the practice of adding components to the system so the load can be spread over multiple resources resulting in less stress on the single unit. This method does not have a single point of failure but rather keeps the system working if one or multiple components go down. The disadvantage with horizontal scaling is the addition

of maintenance and adjusting the source code to parallelization [3].

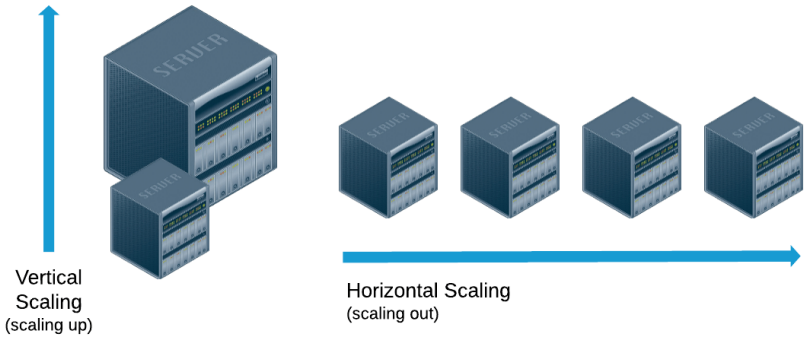

Figure 2: Vertical vs. horizontal scaling
source: https://www.section.io/blog/scaling-horizontally-vs-vertically/

1.4 The Problem

As software solutions grow in popularity, or go through waves of traffic bursts, they need to be able to handle it appropriately since an application needs to be robust and have a good and stable user experience. That is where the problem lies, testing how well a system handles these situations.

2 Scalability Testing

Scalability testing measures how well the system handles scaling up or down, and its purpose is to provide information about how well it performs to meet those demands. The information provided could for example be the hard limit to how much traffic can go through the system at once, how many users can be logged in the application, or their experience using the solution. Some of the attributes measured to perform the tests are as follows: time (response time, session time, reboot time, transaction time), usage (network, memory, throughput), performance (with different numbers of users, under different sizes of load), units per second (requests, transactions, hits) [5].

In this chapter the process of setting up scalability testing is explained, the advantages and disadvantages that follow are explained in more detail, and finally, three alternatives, load testing, stress testing, and capacity testing, are compared to scalability testing.

2.1 The Process

The first step of setting up scalability testing is the definition of three key variables; the criteria for scaling, the process to be repeated for testing executions,

and the virtual users of the system. The next step is configuring the software requirements for the testing environment. This includes hardware and software specifications, and it is important to make sure it can handle the planned testing cycles [6].

After definitions and configurations are set the next step is creating the repeatable test scenarios and test plans. Good test plans include detailed steps of the actions performed by the users, any run-time data that is required to be able to interact with the system, and have some data-driven approaches where data is varying for each run [5]. The plan varies depending on the type of application being tested. If it is a web application where multiple simulated users are sending requests at the same time, it might be reasonable to measure the response time of requests. If it is a cache-driven database application dealing with multiple queries at the same time, checking if the cache is expanded is reasonable [7].

Before execution of the tests, it is important to verify the components of the test plan. Make sure the scenarios work under different situations for hardware or software settings by repeating them after each change [6]. Another factor to verify is the performance data collected. The tests are likely interested in multiple metrics and it is better to collect more information rather than less to have a better understanding of what exactly is happening in the system [7].

Now all the planning has been dealt with and time for the exciting part, the execution of the scalability tests. It is a good idea to start with a fairly simple scaling factor that gets incremented by the same factor after each time. The incrementation is repeated until either the scaling comes to a stop or resources run out. When either has been reached the detailed information generated is ready for analysis [7].

One optional step that was not mentioned is utilizing existing tools, for example:

- Load View
- Apache JMeter
- LoadUI Pro
- Neo Load
- SmartBear

In figures 3 and 4 we can see an example of before and after optimization using data collected from scalability testing using SmartBear. After analyzing the bottleneck of a server with 15 concurrent users, the problem was rooted in database accesses, and after optimizing the database calls the average response time and time spent on testing was reduced significantly [8].

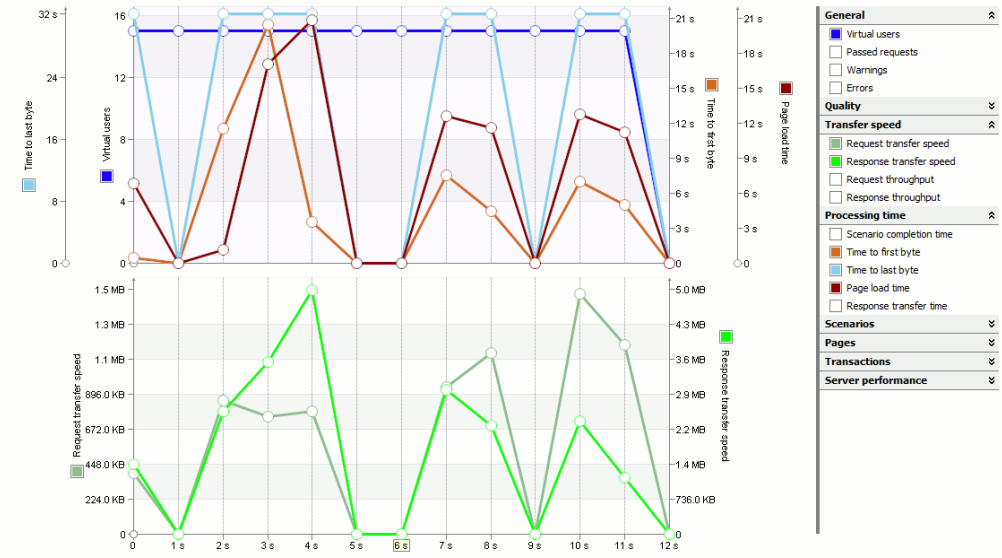

Figure 3: Results of scalability testing before optimization
source: https://support.smartbear.com/loadcomplete/docs/use-cases/scalability-testing.html

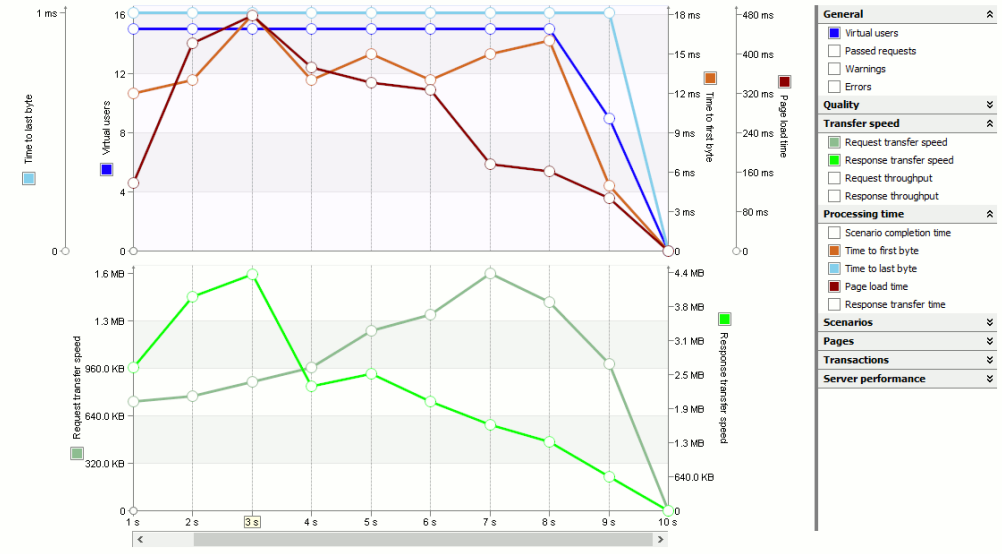

Figure 4: Results of scalability testing after optimization
source: https://support.smartbear.com/loadcomplete/docs/use-cases/scalability-testing.html

Most scalability testing tools have their own unique features and approaches to testing a system. Choosing the right tool for a project can depend on multiple factors to consider: ease of use, efficiency, network protocols, cost, and support [9] [10].

2.2 Advantages and Disadvantages

Benefits in general from scalability testing have been mentioned in the essay before but now three specific examples are listed and discussed in more detail:

1. *Better User Experience* - Using scalability testing gives insight into the eyes of the user when the system is under stress. If the responsiveness or performance of the system is lacking it can lead to bad customer experience. 40% of users will abandon web pages with loading time longer than three seconds, so having data from scalability testing to improve the user's experience can have a significant effect on the traffic of a web page [11].

2. *Lower Resource Cost* - Scalability testing helps optimize the computing power needed for a system, preventing spending more than needed on resources. This is most noticeable during the scaling down of a system - making sure that the resources are freed accordingly when shrinking [11].

3. *Faster Bug Detection* - Being able to detect and make changes during testing allows the developer to spot bugs sooner, leading to a lower cost since the longer the bug exists the more expensive it tends to get to resolve it [11].

The disadvantages of scalability testing mainly connect to the time and cost of setting up. The cost of this technologically advanced method of mimicking traffic might cause a project to go over budget - especially if an expensive existing tool is chosen to perform scalability testing. The time spent setting everything up for scalability testing might cause a delay in meeting a deadline. It is also important to know that scalability testing does not detect all critical errors, meaning it is not recommended to be the stand-alone testing method for a system. [10]

2.3 Related Alternatives

Scalability testing is often compared with three other testing methods, load testing, capacity testing, and stress testing. Although they are closely related there are some differences between them which sets them apart. Knowing these differences can help a developer determine which method fits the situation to test.

- Load testing focuses on finding how much load is needed for the system to fail, whereas scalability testing focuses more on the trend of performance as the load changes. However, load testing is also used to test how the

system handles a specified anticipated number of users or requests at once and see how it performs [5][12].

- Capacity testing tests the maximum amount of users simultaneously using the system or the maximum amount of transactions happening and compares the result with definitions in a service-level agreement (SLA) [11].
- Stress testing is performed to test the system beyond its defined limits. The load mimicked is varied quickly and the interest lies in the user's experience using the system and how recoveries from abrupt failures are handled. This testing method is unique in finding bugs, however, it is not considered to test realistic situations [9][12].

3 Conclusion

This essay has answered the problem laid forth of being able to measure how well a system handles scaling. Enter, scalability testing. Scalability testing gives software developers insight and a better overview of how well the system responds to scaling, and can perhaps lead to the discovery of weaknesses that need to be fixed. The process behind setting up scalability testing was described in 5 steps, defining variables, configuring software requirements, creating the test scenarios and plans, verification, and execution. Adding scalability testing to the continuous testing stage in a pipeline has its benefits. Three examples of benefits were discussed which most software project value greatly, "better user experience", "lower resource cost", and "faster bug detection", as well as the disadvantages of using scalability testing which is related to cost, time and not being able to depend solely on it as a stand-alone testing method. Load testing, capacity testing, and stress testing, three alternatives to scalability testing were explored and their differences were explained.

3.1 Reflection

The advantages of scalability testing seem desirable for most software projects that deal with scaling on a regular basis. It can be set up to run before each continuous deployment as a part of a CI/CD pipeline and give meaningful information about how well the system scales up or down that can be used to improve and optimize.

References

[1] "What is Software Testing and How Does it Work? | IBM." [Online]. Available: https://www.ibm.com/topics/software-testing

[2] "What is continuous testing? | IBM." [Online]. Available: https://www.ibm.com/topics/continuous-testing

[3] "A Fresh Graduate's Guide to Software Development Tools and Technologies - a free online book for Software Engineering students and fresh graduates." [Online]. Available: https://www.comp.nus.edu.sg/ seer/book/2e/

[4] G. Brataas, G. K. Hanssen, N. Herbst, and A. van Hoorn, "Agile scalability engineering: The scrumscale method," *IEEE Software*, vol. 37, no. 5, pp. 77–84, 2019.

[5] T. Hamilton, "What is Scalability Testing? Learn with Example," Feb. 2020. [Online]. Available: https://www.guru99.com/scalability-testing.html

[6] "What is Scalability Testing and How Does It Impact Performance?" Mar. 2022. [Online]. Available: https://www.copado.com/devops-hub/blog/what-is-scalability-testing-and-how-does-it-impact-performance-crt

[7] perfwork, "Scalability Testing," Mar. 2012. [Online]. Available: https://perfwork.wordpress.com/2012/03/14/scalability-testing/

[8] "Scalability Testing: Checking Whether a Site Performance Can Scale Up | LoadComplete Documentation." [Online]. Available: https://support.smartbear.com/loadcomplete/docs/use-cases/scalability-testing.html

[9] "Scalability Testing Tutorial: A Comprehensive Guide With Examples And Best Practices." [Online]. Available: https://www.lambdatest.com/learning-hub/scalability-testing

[10] QAonCloud, "A step by step guide to scalability testing: Setup, Execution, and Reporting," Oct. 2021. [Online]. Available: https://www.qaoncloud.com/scalability-testing/

[11] "Demystifying Scalability Testing: The Complete Guide," Dec. 2020. [Online]. Available: https://www.radview.com/blog/demystifying-scalability-testing-the-complete-guide/

[12] M. E. Khan, "Different forms of software testing techniques for finding errors," *International Journal of Computer Science Issues (IJCSI)*, vol. 7, no. 3, p. 24, 2010.