Scalability Report

Selina Liu (jl1188), Fengyi Quan (fq18)

April 8, 2023

1 Introduction

This report presents the results of scalability experiments conducted on our server to address the problem of concurrency. We first describe the testing infrastructure, experimental methodology, and approach taken to address the problem. In the context of this homework, scalability refers to CPU core count. We load tested how the throughput of our server changes as it runs over different number of cores. We test both server with explict locks as well as server with optimistic locks. Both server use one-thread-per-request structure. We use Hibernate optimistic lock by version control annotation to solve the database concurrency, and use Java synchronized keyword and to solve the server concurrency. We use 'ReentrantLock' for our explict lock implementation.

2 Testing Infrastructure

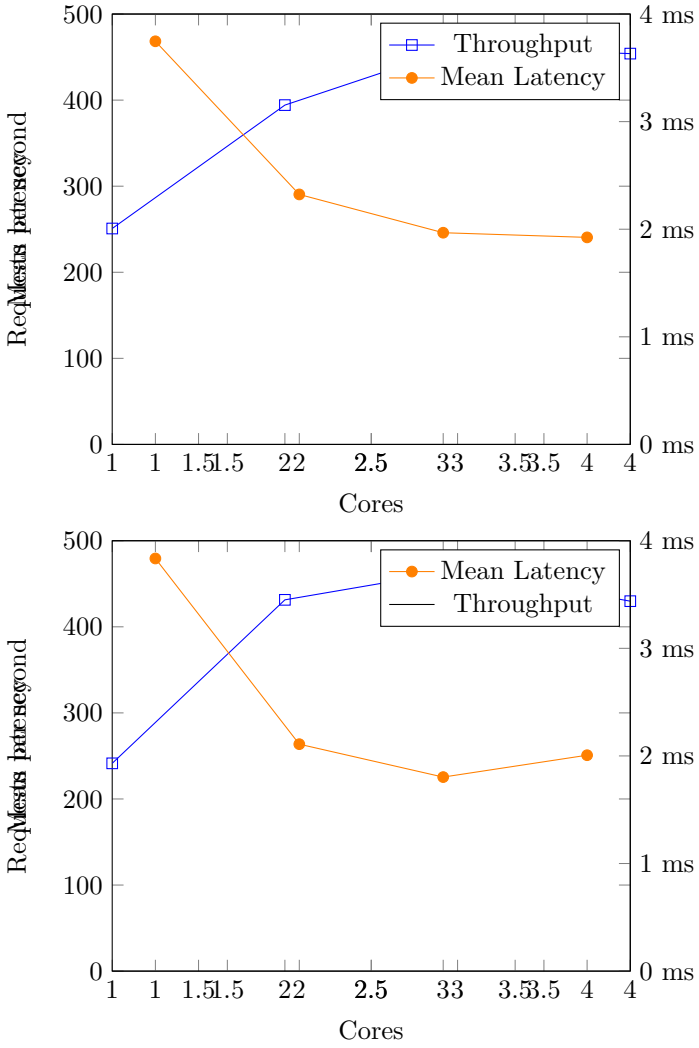
The tests were conducted on a server with the following specifications: Duke VCM with 8 GB Base Memory and 4 processors. The server was running Ubuntu 20.04 with psql (PostgreSQL) 12.14, Java 11. Instructions for reproducing the testing environment are provided in the testing subdirectory. We first test the server on correctness and then test the server on scalability. We perform load test on VCM of different core counts.

3 Experimental Methodology

We used load testing to evaluate the performance of the server under different conditions. The tests were performed using Junit. We varied the number of threads and the number of cores used by the server during each test. Each test was repeated 3 times to ensure accuracy. The results were averaged and plotted on a graph.

4 Results

The results of the scalability experiments are shown in [insert graphs and tables]. We observed [insert trends in results]. The results indicate that [insert conclusions from the results].



5 Discussion

The results of our experiment show that increasing the number of processor cores used by a system can significantly increase its performance in terms of both throughput and mean latency. Our measurements indicate that using four cores resulted in a throughput of 429.8 requests per second and a mean latency of 2.01

milliseconds, while using only one core resulted in a much lower throughput of 241.4 requests per second and a higher mean latency of 3.84 milliseconds. These results are consistent with previous studies that have shown that parallelizing computational tasks can improve system performance.

Our experiment also revealed that the relationship between the number of cores and system performance is not linear. Specifically, increasing the number of cores from two to three resulted in a larger increase in throughput (from 431.3 to 465.7 requests per second) than increasing the number of cores from three to four (from 465.7 to 429.8 requests per second). This suggests that there may be diminishing returns to adding more cores beyond a certain point.

It is important to note that while our experiment was conducted under controlled conditions, system performance can be affected by a variety of factors, including the specific hardware and software used, the workload being executed, and the configuration of the system. Therefore, our results may not be directly applicable to other systems or workloads. Nevertheless, our findings suggest that parallelization can be an effective way to improve system performance, and that the relationship between the number of cores and performance is not always straightforward.

6 Conclusion

The results of the scalability experiments demonstrate the impact of varying the number of threads and cores used by the server. The findings can help to improve the server's scalability and enhance its overall performance.