ECE568 – HW4 Report

When building and creating software systems, scalability must be taken into account. It speaks to a system's capacity to manage an increase in workload without suffering performance effects. Scalability is especially important in circumstances where the workload could increase dramatically or unpredictably, such in cloud computing or web-based applications.

The number of threads and cores the system is using are the two main variables that affect scalability. The independent execution units known as threads enable a system to carry out several tasks at once. On the other hand, a system's physical processors are referred to as cores.

1.0 Test setup

In our design pattern, we've used a thread pool with 100 threads in it. To test our server's scalability, we created a great number of clients that send requests to the server under the situations of only 1, 2 and 4 cores of my computer being visible to the threads of my processes. Because we cannot deploy the app into the docker container, we conducted the experiments in a Windows system. The script to test the server scalability is located in app/client/Client.java.

2.0 Scalability vs Threads

This information suggests that the running time typically increases as the overall number of requests and the number of requests per thread rise. The pace of growth is not always linear, though, and in certain instances, the running time actually lowers as the number of requests per thread rises.

It is also important to remember that the running duration varies greatly based on the number of threads utilized. In particular, 100 threads consistently beat both higher and lower thread counts for all request levels, yielding the highest performance.

Overall, these findings imply that, in order to maximize performance, the recommended setup for your process would involve employing 100 threads and splitting the workload across many cores.

3.0 Scalability vs Cores

According to the statistics, as the number of cores rises, the process's running time falls, showing that additional cores make the process more scalable. This

is especially apparent when there are many threads and requests, as the running time on four cores is much shorter than that on one or two cores in those situations.

It's crucial to remember that scalability is not always linear and that as the number of cores rises, the rate of advancement falls. Beyond a certain point, adding more cores might not always result in noticeable performance improvements.

4.0 Conclusion

To determine the ideal number of threads and cores for your workload, benchmark testing on various hardware configurations while optimizing the performance of your process is crucial. By finding the optimal balance between threads and cores, you can ensure that your system is scalable and can handle increased workloads effectively.