Homework 4 Scalability Report

Kexin Liao(kl460) Xianjing Huang(xh135)

1. Introduction

In this report, we will introduce the methods we implement in this assignment and the scalability analysis based on this work. We build a server that can receive requests from clients and give responses to them. This information is stored in a series of database tables. Then we tested our matching engine from functionality and scalability.

2. XML Parser

We use tinyxml2 as the XML parser library.

3. Database Design

We created a database called 'matching' in postgres, and designed 5 relations to help handle the matching engine. These five tables are:

ACCOUNT(ACCOUNT_ID, BALANCE);
POSITION(POSITION_ID, ACCOUNT_ID, SYMBOL, AMOUNT);
OPEN_ORDER(ORDER_ID, ACCOUNT_ID, SYMBOL, AMOUNT, PRICE, TIME)
EXEC(EXEC_ID, BUYER_ID, SELLER_ID, BUY_ID, SELL_ID, SYMBOL, PRICE, AMOUNT, TIME)
CANCEL(CANCEL_ID, ACCOUNT_ID, TRANS_ID, SYMBOL, AMOUNT, TIME)

Trans_id is the order_id in OPEN_ORDER table which is its primary key.

4. Database Concurrency

In order to handle database concurrency, we used row level locking 'SELECT … FOR UPDATE'. It helps lock the value we chose when it modifies and writes in one session, but it also might cause deadlock so we need to use it carefully.

5. Scalability Experiments

(1) Method

To test the scalability of our server, we create 7000, 4000, and 1000 clients to connect and send requests to the server concurrently. And we run the server on 1 core, 2 cores and 4 cores respectively. Then we record the running time for each test and compute the throughput.

Since every measurement has some degree of uncertainty, we run each test 5 times and take the average time as the value. Error bars are visual representations of the variability or uncertainty of the data. They can indicate standard deviation, standard error, or confidence intervals. So we also calculate standard deviation and standard error for error bars, adding them to the bar chart.

To limit server to running on a certain number of cores, we use following commands to make cpu groups:

```
sudo cgcreate -g cpu:/cpu_0
sudo cgcreate -g cpu:/cpu_01
```

sudo cgcreate -g cpu:/cpu_0123
sudo cgset -r cpuset.cpus="0" cpu_0
sudo cgset -r cpuset.cpus="0,1" cpu_01
sudo cgset -r cpuset.cpus="0,1,2,3" cpu_0123
And then run the server using:
sudo cgexec -g cpu:/cpu_0 ./server
sudo cgexec -g cpu:/cpu_01 ./server
sudo cgexec -g cpu:/cpu_0123 ./server

(2) Results

The screenshots of the original running time are displayed in the appendix. The average time is computed by 5 test results.

Table 1: Throughput Performance

	Average Time(/second)			Throughput(request/second)		
# Requests	1 core	2 cores	4 cores	1 core	2 cores	4 cores
7k	28.237	27.631	27.206	247.902	253.339	257.296
4k	16.330	16.175	15.697	244.948	247.295	254.826
1k	4.088	4.000	3.921	244.618	250.000	255.037

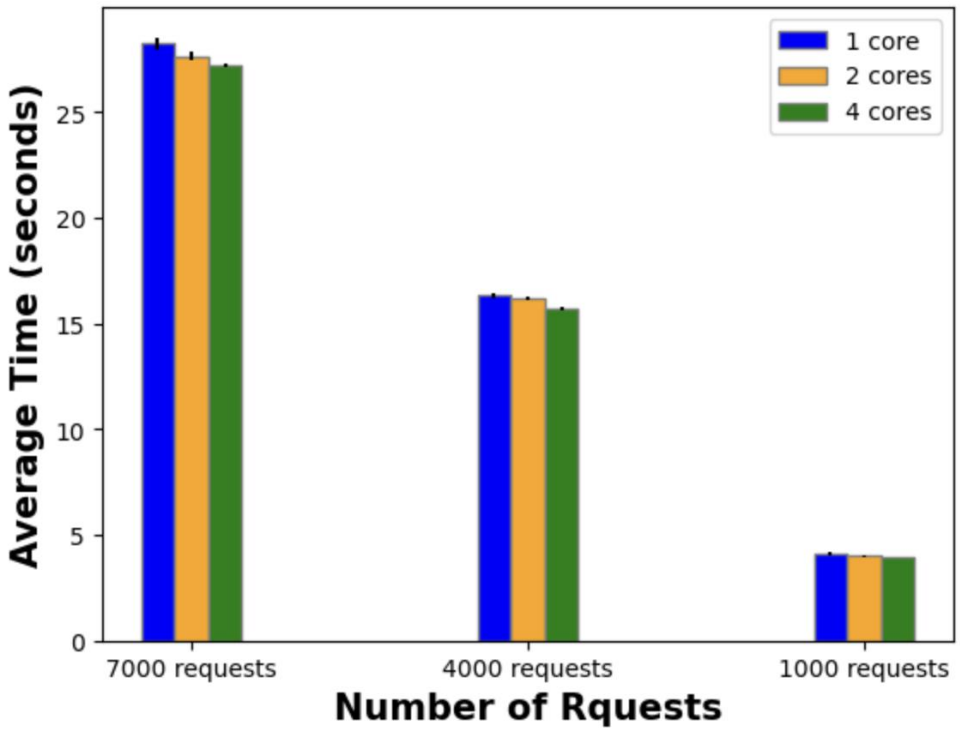

Figure 1: Average Execution Time with Error Bars

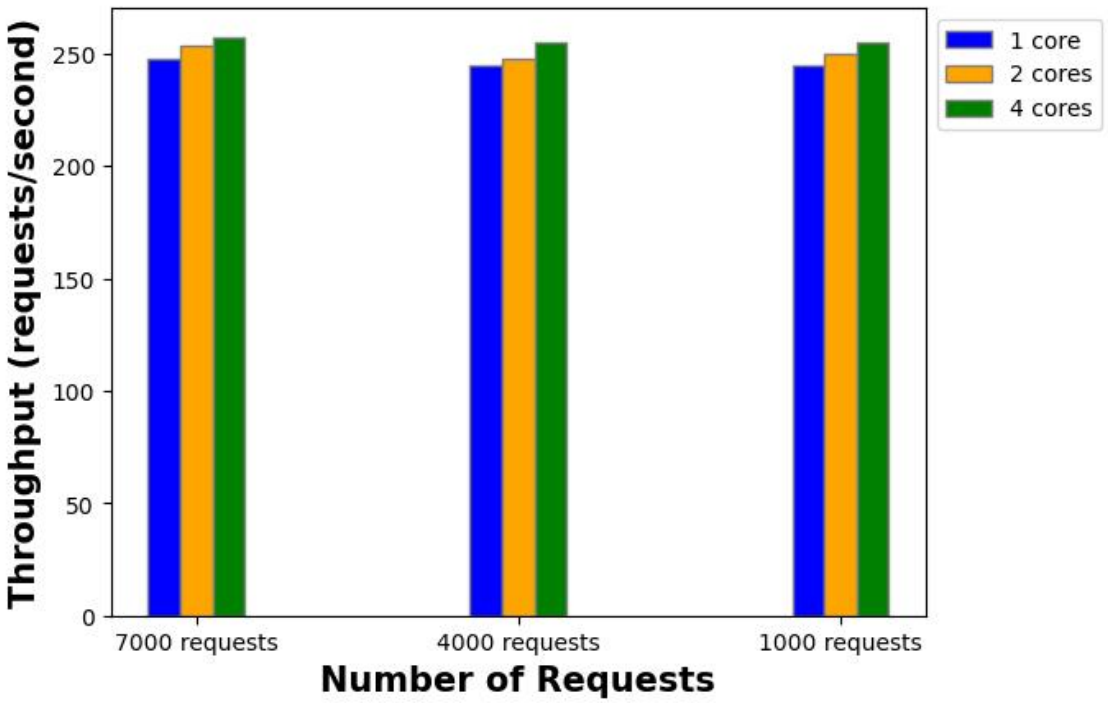

Figure 2: Throughput Performance for Different Cores

(3) Analysis

From the “Average Execution Time” and “Throughput Performance” bar chart and table, we can see that as the number of cores increases, the average time for the server to process the same number of requests decreases, and the throughput increases. As the number of requests increases, the difference in running time among servers with different numbers of cores becomes greater. However, the server’s performance does not improve significantly when increasing the number of CPU cores. We may analyze the reasons as follows:

a. Server implementation also has a significant impact on server performance, which may limit server scalability. Further optimizing server design can achieve better scalability.
b. The total runtime includes not only the time required to process requests, but also the time required for system startup, library loading, and so on. The longer the running time, the greater the proportion of processing request time, and the more it can be seen that increasing the number of cores improves server performance.
c. The content for shared resources, like memory or network bandwidth, may limit the scalability of the server.
d. An increase in the number of cores will lead to an increase in overhead for managing multiple cores, which will also affect scalability.

Appendix: screenshots of the original running time

1 core, 7k requests:

```
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 27431 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 29176 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 28283 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 28393 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 27903 milliseconds
```

2 cores, 7k requests:

```
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 27291 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 28186 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 27493 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 28009 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 27175 milliseconds
```

4 cores, 7k requests:

```
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 26960 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 27280 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 27089 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 27100 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 27605 milliseconds
```

1 core, 4k requests:

```
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 16668 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
^[[ATime: 16489 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 16082 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 16200 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 16205 milliseconds
```

2 cores, 4k requests:

```
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 16119 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 16011 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 16332 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 16280 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 16132 milliseconds
```

4 cores, 4k requests:

```
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 15865 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 15544 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 15554 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 15572 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 15955 milliseconds
```

1 core, 1k requests:

```
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 3938 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 4214 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 4242 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 4097 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 3947 milliseconds
```

2 cores, 1k requests:

```
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 4027 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 3946 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 4002 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 4047 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 3975 milliseconds
```

4 cores, 1k requests:

```
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 3935 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 3950 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 3894 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 3933 milliseconds
xh135@vcm-38181:~/erss-hwk4-kl460-xh135/testing$ ./test_scalability 127.0.0.1
Time: 3896 milliseconds
```