COMP 512: Performance Analysis Report

Jin Dong, 260860634; Shiquan Zhang, 260850447

1 Test Bed

The performance test has two parts: single client and multiple clients. In the multiple client test, we measure the performance with 2, 4, 6 and 8 clients in the system.

In each client, we use 200 *customers*, 200 *flights* and 100 *locations/cities*. At the preparation phase, we add these *customers*, *flights*, *cars* and *rooms* to the server. Then in the test phase, we run 100 iterations with different itineraries. In each itinerary, client sends 9 requests in one transaction: *start*, *queryCustomer*, *queryFlight*, *reserveFlight*, *queryCar*, *reserveCar*, *queryRoom*, *reserveRoom* and *commit*. In the single RM test, in order to keep the same operation in one transaction, we make 3 $query - reserve$ pair on the *flight* RM.

We measure the response time of every request in the itinerary and calculate the $averagequerytime$ of 4 queries and $averagereservetime$ of 3 reservations. Besides, we also record the time of every transaction/itinerary. When calculating the overall $averagetime$, we leverage the time in the last 80 iterations to get the stable result.

In the multiple client test, we need to avoid deadlock. We use $(i * 8 + clientNum)\%cityNum$ as the index, which can avoid different clients reading and writing the same object at the same time. Additionally, we add a ±1% variation in every transaction time interval.

2 Test Result

2.1 Single Client Test

The overall average start time, query time, reserve time and commit time with single RM and multiple RM in single client test is shown in Table 1. The unit of the values is millisecond.

/ms	Start Time	Query Time	Reserve Time	Commit Time	Tx Time
Single RM	0.31	0.533768	0.650833	0.61	5.02
Multiple RM	0.3375	0.903125	1.0575	2.2875	9.425

Table 1: Response Time of the Single Client Test with Single/Multiple RM

2.2 Multiple Client Test

The average transaction time of multiple client test with different transaction rates and different number of clients are shown in Figure 1. Since the minimum time interval between two transactions is 1 ms, the maximum transaction rate varies from 2000 tx/s to 8000 tx/s.

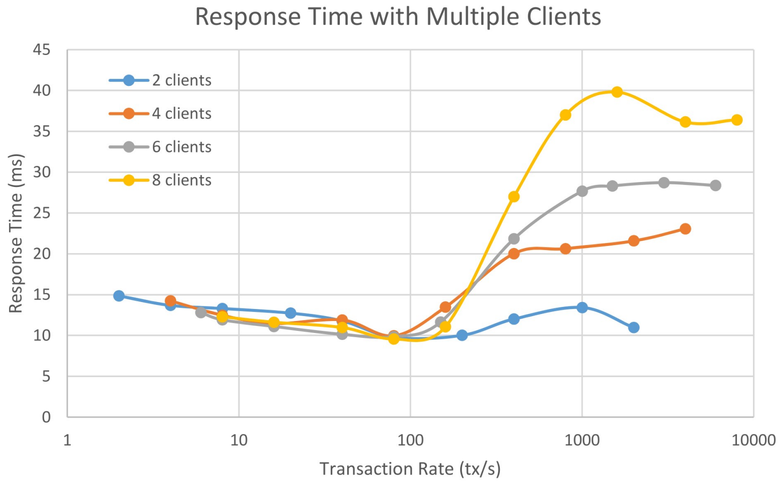

Figure 1: Response Time of Multiple Client Test

3 Performance Analysis

3.1 Single Client Test

From Table 1, we can see that the total transaction time is roughly the linear combination of all requests time: $T_{start} + 4 \times T_{query} + 3 \times T_{reserve} + T_{commit} \approx T_{tx}$. Consider the differece between the query time and reserve time, the reserve operation not only reads the data but also writes new data to item RM and customer RM, thus the difference (around 0.1 ms) is casused by the write operation.

Comparing the response time between single RM and multiple RM, the differences are mainly in query, reservation and commit. In query and reserve, the operations of two types of transactions are different mainly in lock management and saving undo hashmap. In every round of multiple RM test, the query requests have 2 more locks to grant and the reserve requests have 2 more locks to convert and 2 more items to save the undo hashmap. In commit requests, the cost difference is relatively large, since in the multiple RM test, Middleware needs to communicate with 2 more RMs, which costs much more time. Therefore, we can infer that the time is mainly spent in lock management in Middleware when querying, in data writing and undo hashmap saving in RM when reserving, and in communication when committing.

3.2 Multiple Client Test

From Figure 1, we can see that when the transaction rate is less than 100 tx/s, the response time remains low. When the transaction rate increases to $[100, 1000]$ tx/s, the response time goes up. When the transaction rate is more than 100 tx/s, the system is saturated and the response time remains in a high level.

When the workload is low, the system can handle all the requests from different clients in time, so the average response time is close to that in the single client test. When the workload increases, the server is no longer be able to handle all the transactions in time, so the response time increases. When the workload goes beyond some point around 1000 tx/s, the time interval between transactions is less than the response time, thus the client will continuously send the request and the system is saturated.

Furthermore, if we look into the response time with high workload by time slots, we can see that there are some spikes in the response time, which means that sometime the transactions from all clients are congested in the server at the same time and causing the overall average response time larger than normal case. This may be caused by the data switching between memory and disk in Middleware.

For custom functionality, we also test the system with 2, 4, 6 and 8 clients. The response time with different number of clients are the same when the workload is low and all reach the lowest point at around 100 tx/s. Then all the response time increase until saturated, while the response time with 2 clients increases little. Besides, the saturated response time are roughly linear to the number of clients, because although the transaction rate is the same, the middleware needs to handle more requests at the same time with more clients in the system.