Content

Report

1 Introduction

This report provides test results and analysis of the performance of the Online book store application developed in assignment 2 in the course COMP9321. We start out by an analysis of relevant application including what components are designed and how they work in order to serve user requests. Then details about this test such as hardware info, operating environment and Database used is going to be given. Next, we will analyse the performance of the application based on our test results, discuss possible performance improvements, and in the end give the conclusion of this test.

2 Application Details

2.1 User Login

User inputs his user name and password and hits the “login” button, and then a call is made to the database to check whether the username or password is valid. To be specific, the call will invoke the function userValidate() in the controller, and that function will build up a PreparedStatement for the query. After executing the query, the function will firstly check whether there exists that username, and then check whether the password is valid. The whole process involves one call to the database. The figure shows below:

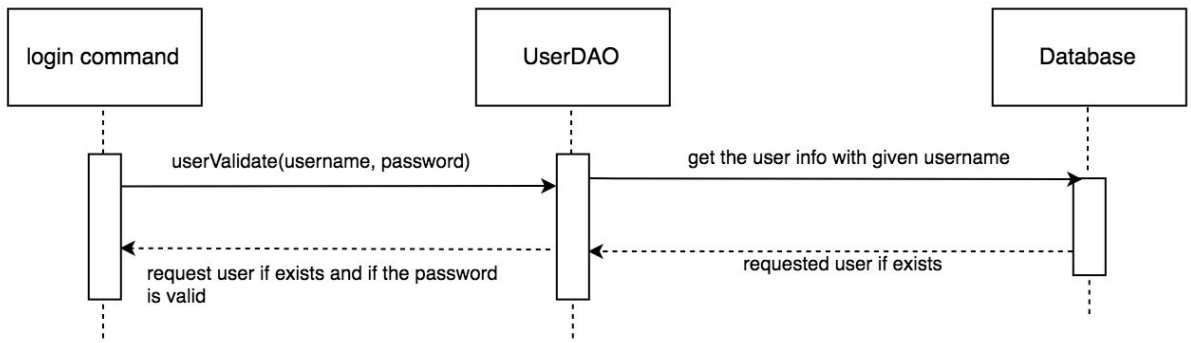


Figure1: sequence diagram for user’s login

2.2 user add book

User fills the book information, adding a picture and then hits the “add” button. The page will send these information to the controller, invoking the function addBook() in bookDAO to insert the book information into book table and sell table. At the same time, it stores the picture of book to the local hard disk. This function will involve three interactions with the database: 1. An update operation: Insert the book basic information into the book table except book picture and invoke pstmt.getGeneratedKeys() to get the bookid of that book. 2. An update operation: Insert the picture into the record of that book. 3. An update operation: Insert the userid and bookid into the sell table. The figure shows below:

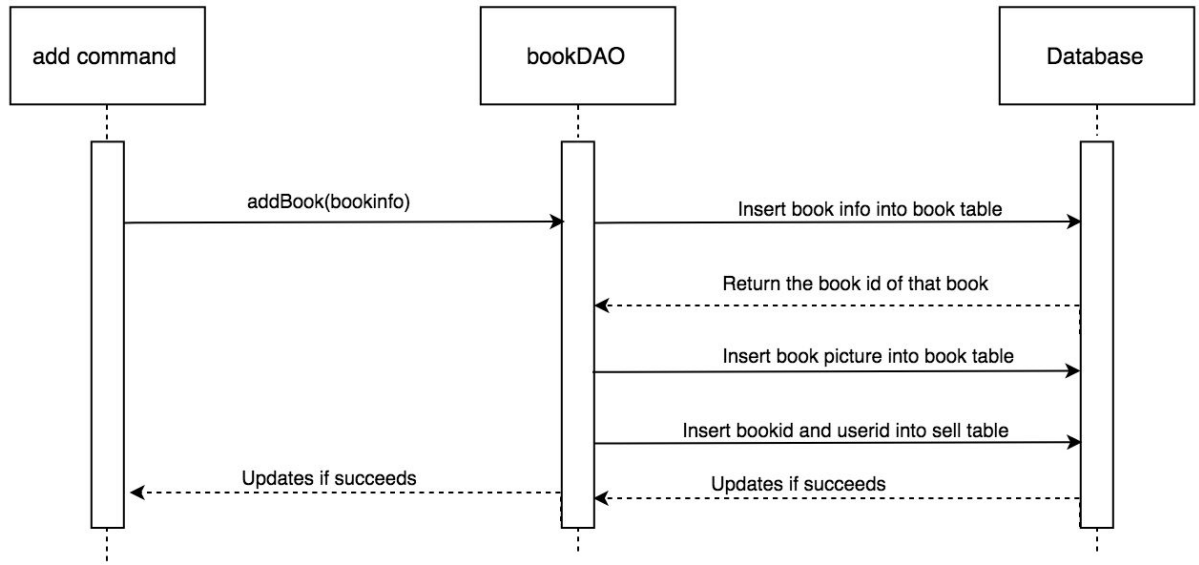


Figure2: sequence diagram for adding a book

2.3 hold books by users

User have the privilege to hold or recover the books they have put on the website. Since the principle of holding or recovering books is exactly the same. I will explain the operation of holding books only. A user can tick the checkboxes to select the books to be held. When he hits the “hold” button, the form which contains the bookids of those selected books will be transferred to the controller, invoking the function holdBook() to change the status of those selected books. And then invokes the function getMyBookList() to get all the books of that user and go back to the

“myBooks.jsp” page. The whole process involves two calls to the database: 1. An update operation: update the status of those selected books to be 0, which means those books will not be seen in the search results. 2. An query operation, get all the books which belongs to that user. The figure shows below:

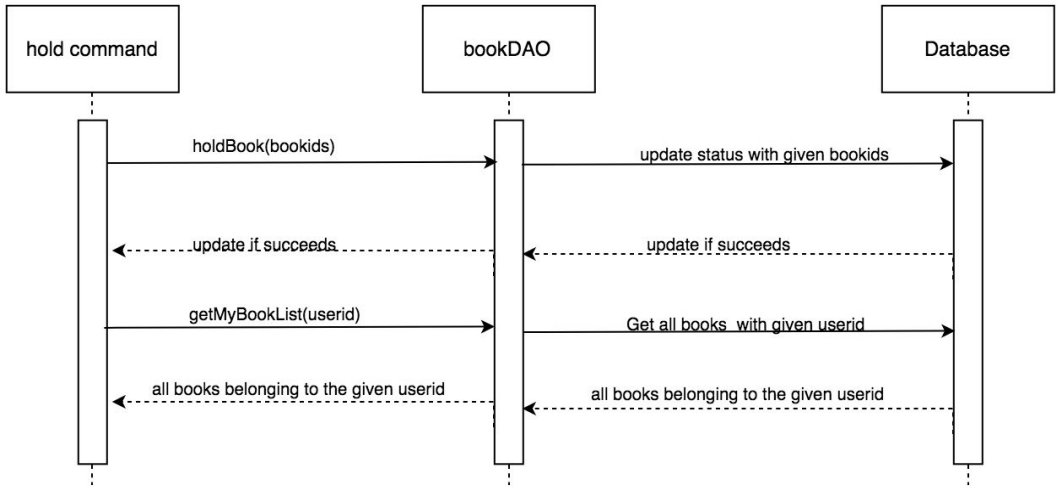
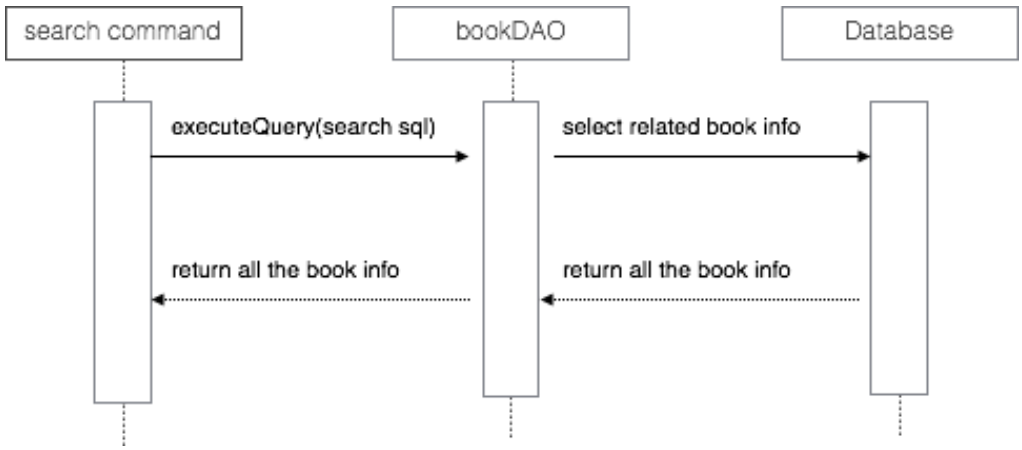


Figure3: sequence diagram for holding books

2.4 User search books

A user can search for a particular item. First user enter some search detail such as publication, author, title and etc. when he clicks on the ‘search’ button the form which contains all the search keywords will be sent to controller and invoking the function makeSQL() to return a SQL query to result page and result page will use javaBean connecting to database and invoking function executeQuery() to select related result to user. The figure shows below:

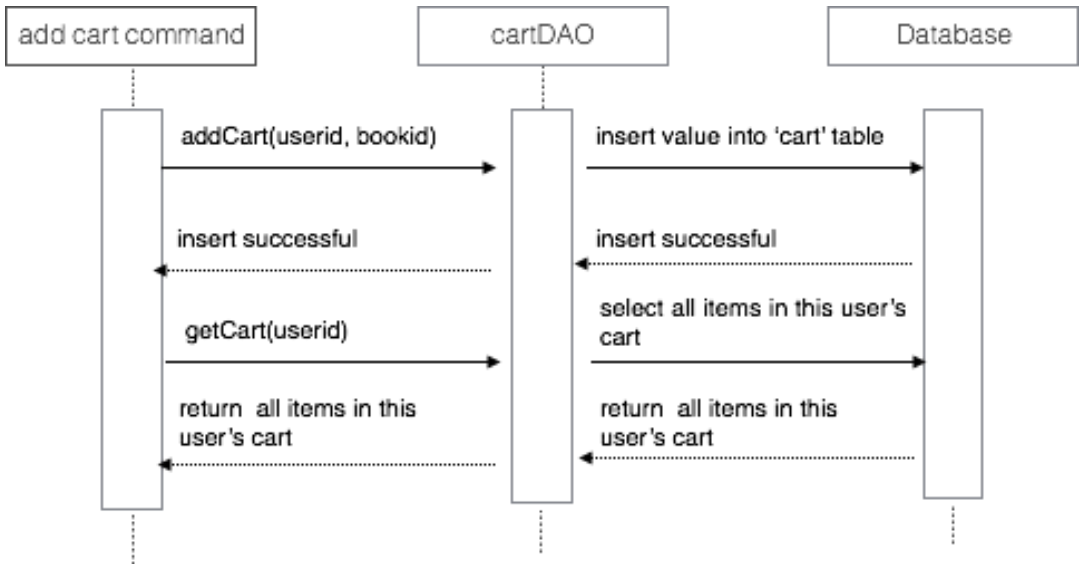


Search parameter in this part

Search parameter	DB returned records
publication = article	187
year ≥ 2000	
title contains 'network'	
price ≤ 100	

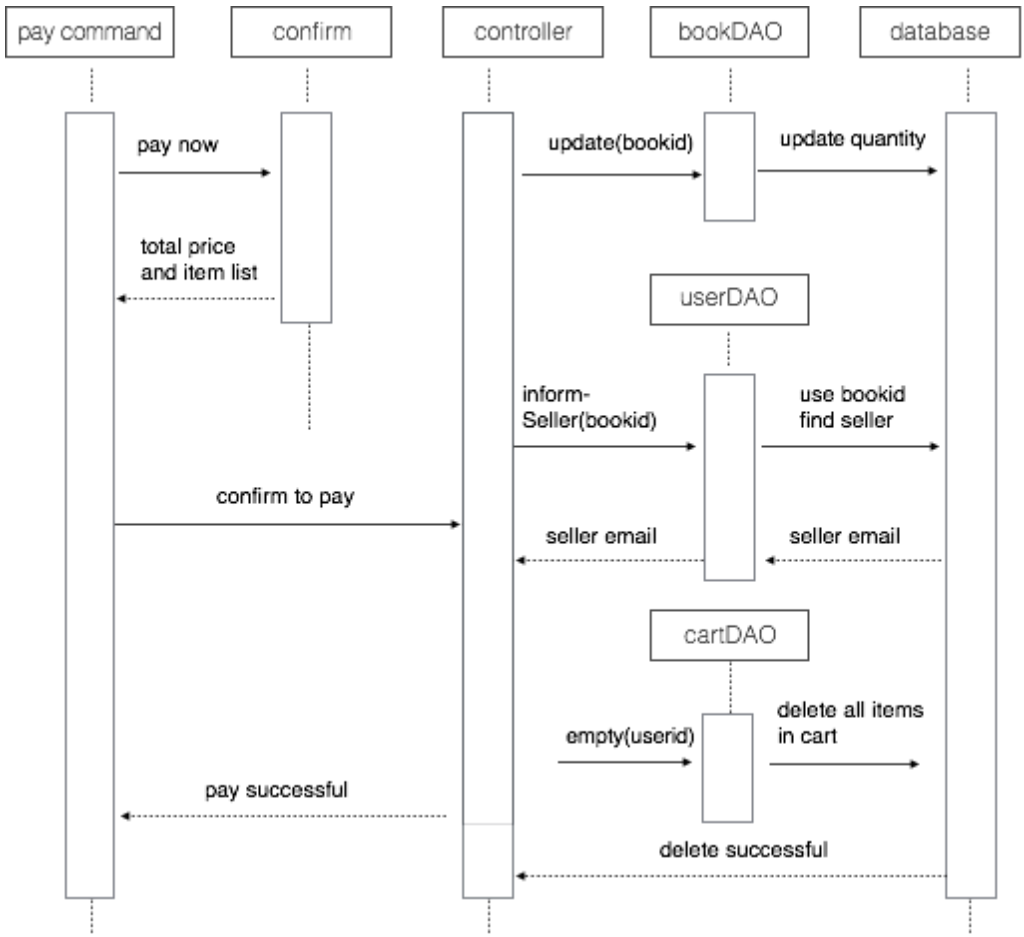
2.5 User add book to shopping cart

In the result page and book info page user can click on the button 'add to cart' to add item in their shopping cart.And once he dose this, it will send the bookid to controller and invoke the method addCart() to add this item to their shopping cart and get all the items in their shopping cart by using method getcart() and go to 'cart.jsp' page. The whole process involves two calls to the database: 1. An insert operation: insert userid and bookid into cart table. 2. An query operation, get all the bookid from cart table which belongs to that user. The figure shows below:



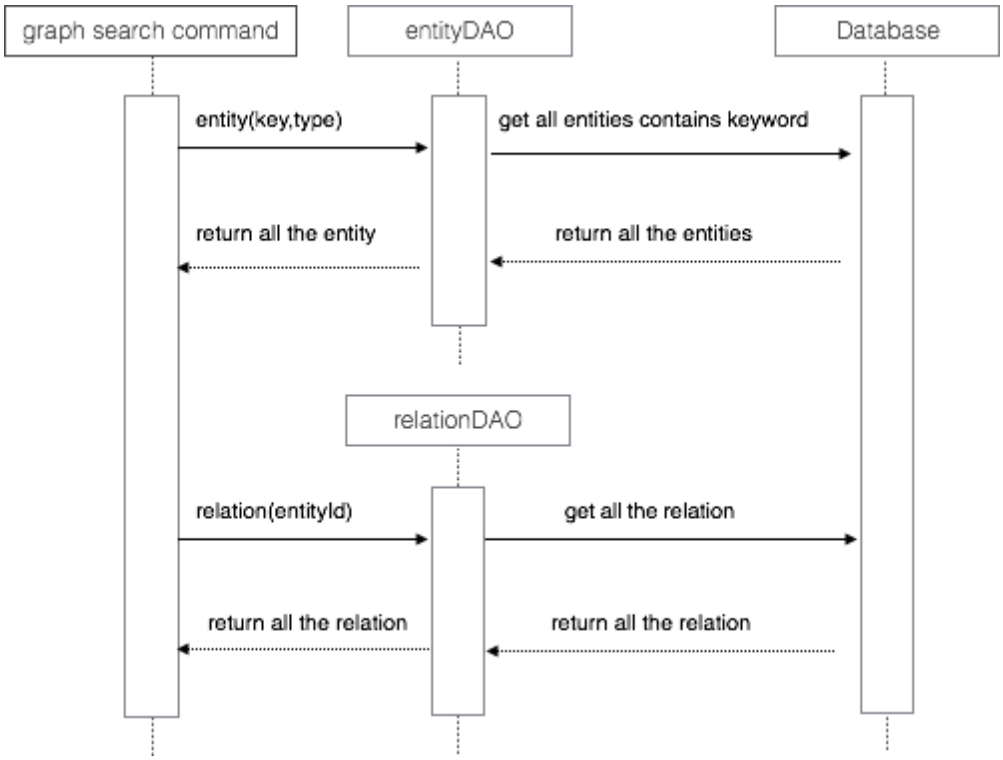
2.6 Shopping cart payment

In the shopping cart page, user can pay for all the item by clicking the button 'Pay Now', once he does, this page will goto a confirm page which shows all the item name he is going to buy and the total price, when user confirm to buy, this page will send the command to the controller and call three functions:1. update() this will update book available quantity in database 2. informSeller() which will send email to the book seller if exist. 3. emptyCart() which will empty this user's shopping cart. The figure shows below:



2.7 Graph and Visualisation

In the ‘graph_form.jsp’ page, when user enter a keyword of entity(three choices: author, publication and venue), controller will receive parameter ‘search_key’ and ‘search_type’ and call function ‘entity()’ return all the entity and next for each entity goto relation table in database and get all this entity’s relationship by using method ‘relation()’, then return all relations in the database. The figure shows below:



Search parameter in this part

Search parameter	DB returned records
search key= web development	Entity : 5
search type = publication	Relation : 5

3 Test setup

3.1 Client side detail

MacBook Air (version 10.11.6)

Hardware variable	value
CPU	1.6 GHz Intel Core i5
RAM	8 GB 1600 MHz DDR3
SSD	120GB

3.2 Server side detail

MacBook Pro 10.11.3

Hardware variable	Value
CPU	2.7 GHz Intel Core i5
RAM	8 GB 1867 MHz DDR3
SSD	128GB

Operating variable	Value
Operating System	Mac OS 10.11.3
Java version	1.8.0_91
JVM Server	Java HotSpot(TM) 64-Bit Server VM)
JVM Server version	1.8.0_91-b14
JVM maximum memory	1.78GB
Apache Tomcat Server	7.0.75
Database	MySQL
MySQL version	Ver 14.14 Distrib 5.7.18

3.3 Details of test plan

The test consists of two parts: client side and server side, which can be observed on GUI of Jmeter and Probe page in Tomcat respectively.The tests were run over a computer-to-computer network.
There are some scenarios that are going to be simulated:

1. A group of users try to log in the website.
2. They are going to add a book to be sold through this website.
3. They want to hold some books they have added.
4. They want to search some particular books.
5. They can add books they like to shopping cart.
6. They can buy items in their shopping cart.
7. They can also use graph page to visualize relations between different items.

The range of concurrent users are as the following numbers: 10, 50, 100, 150, 200, 250, 300, 400, 500, 800. And each of those group of users will be handled in 5 seconds. Since we would like to challenge our websites, we enlarged the concurrent

user number to 800 per 5 seconds(And from the test result, we cannot handle 800 users, which can be seen in the result part).

On the client side the test will be conducted in Jmeter, by adding a “Thread Group”, “Simple Controller”, “HTTP request sampler” and some listeners used to monitor those data, such as Average response time, median response time, throughput etc. Each time after running a test for a group of users, we need to clean the records of those listeners. For each scenario, we are going to test its performance by starting from 10 users. We use server machine’s IP address as test path in client’s Jmeter to ensure that we run JMeter and application on separate machines.

On the server side, we focus on the Probe page. Each time we can observed the average response time, request numbers and processing time generated during the test on the server. These data can be used in the result part of our report. But we need to restart the server between each test as well.

4 Test Result

4.1 Response time

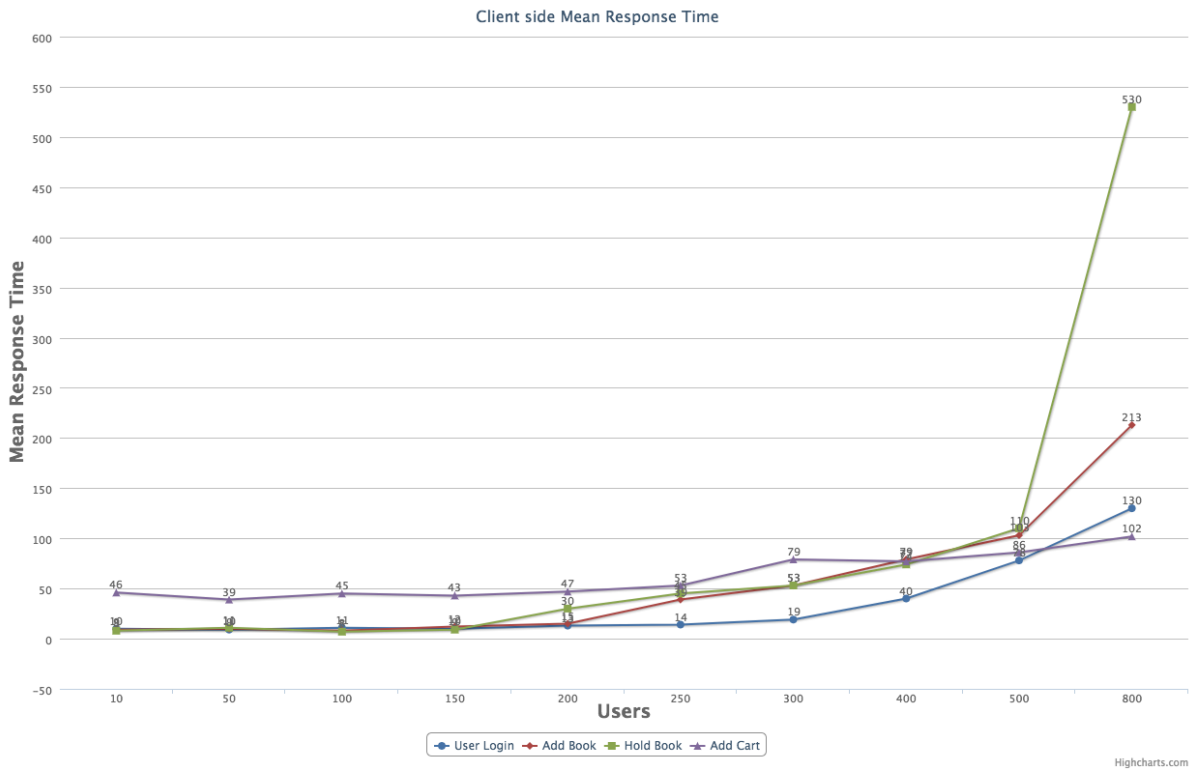


Figure 4.1 Client side Mean Response Time Part 1

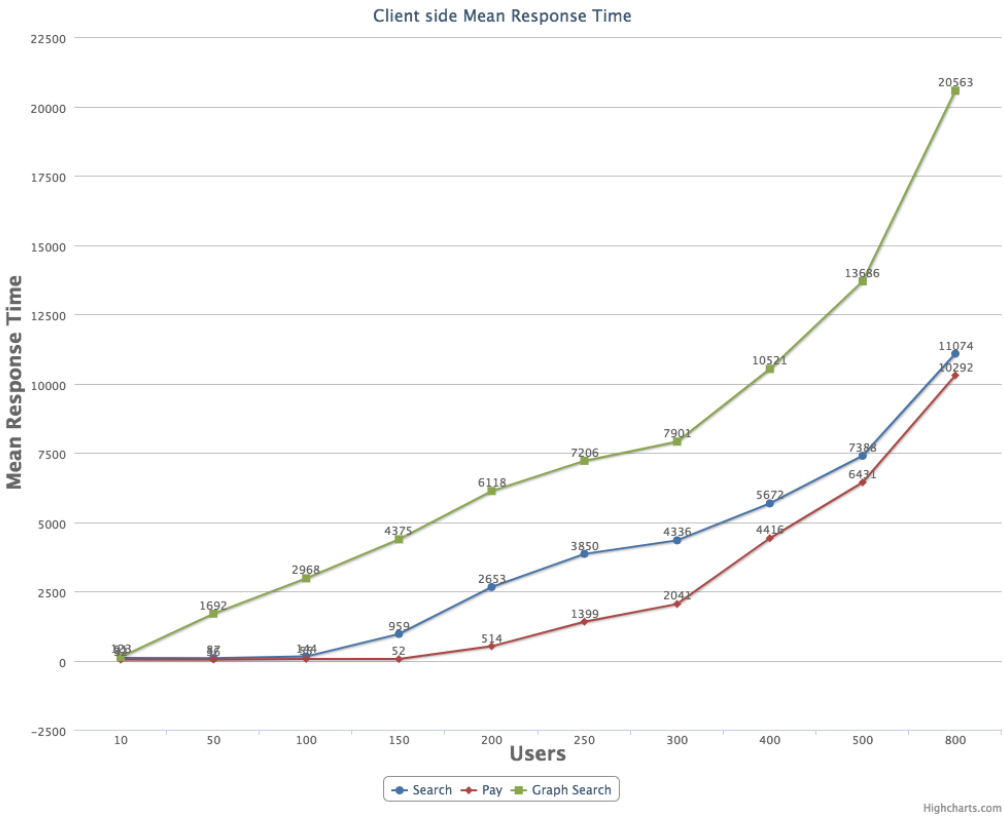


Figure 4.2 Client side Mean Response Time Part 2

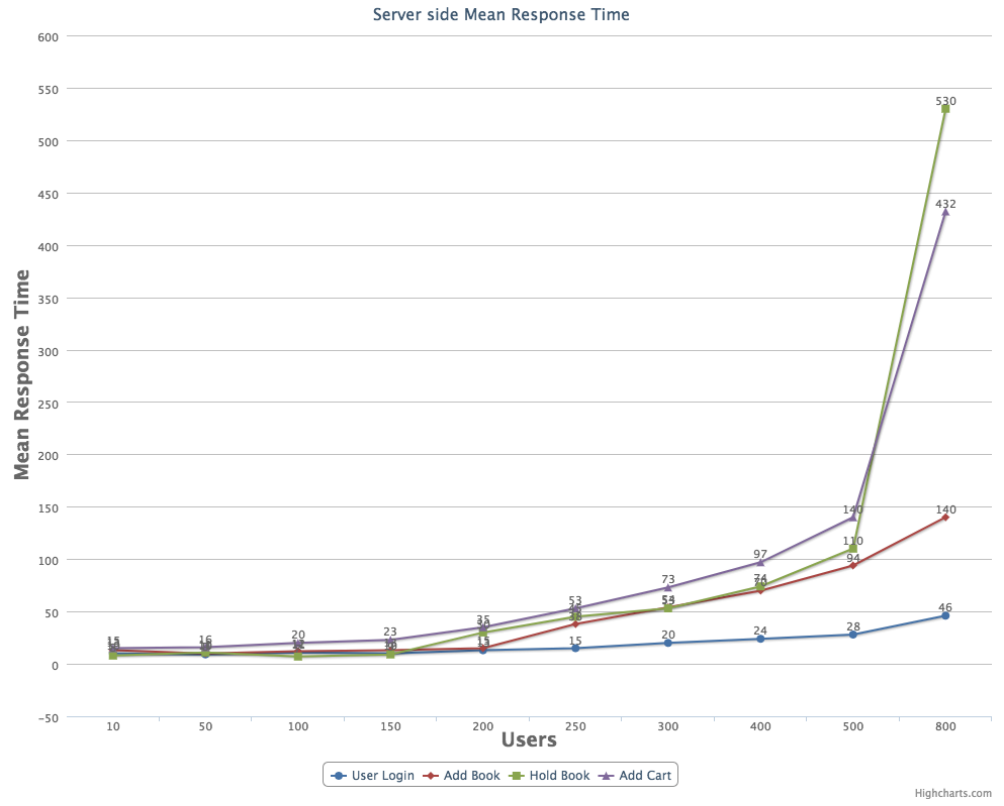


Figure 4.3 Server side Mean Response Time Part 1

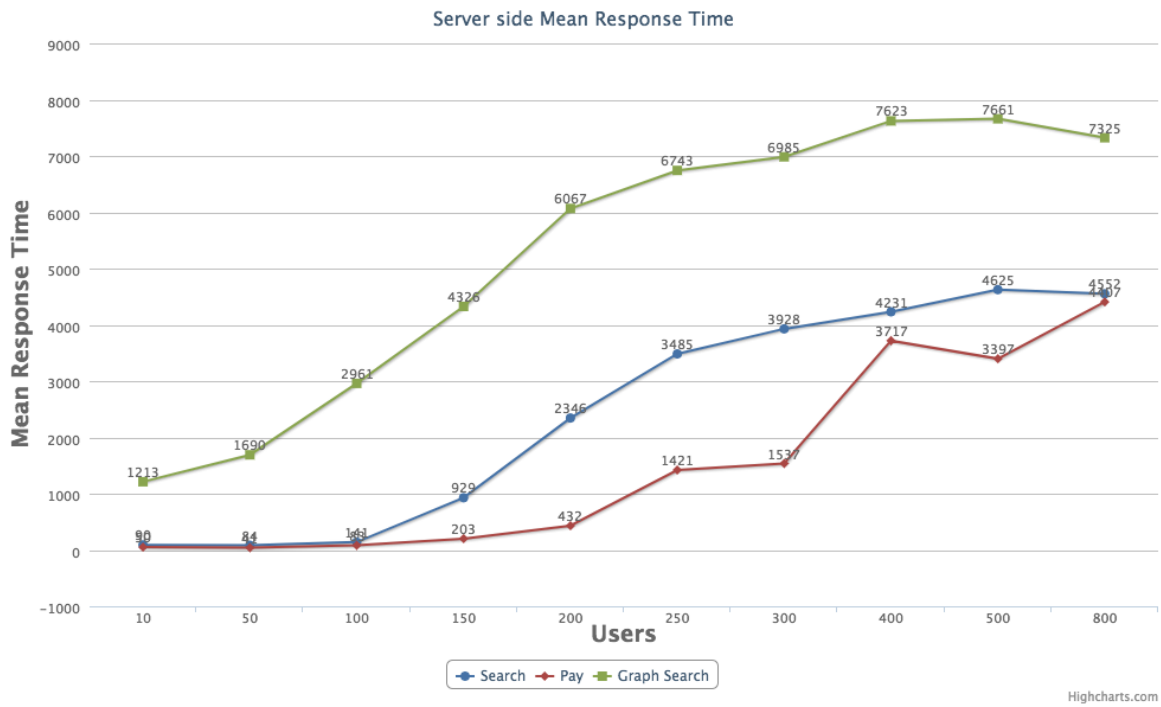


Figure 4.4 Server side Mean Response Time Part 2

4.2 Throughput

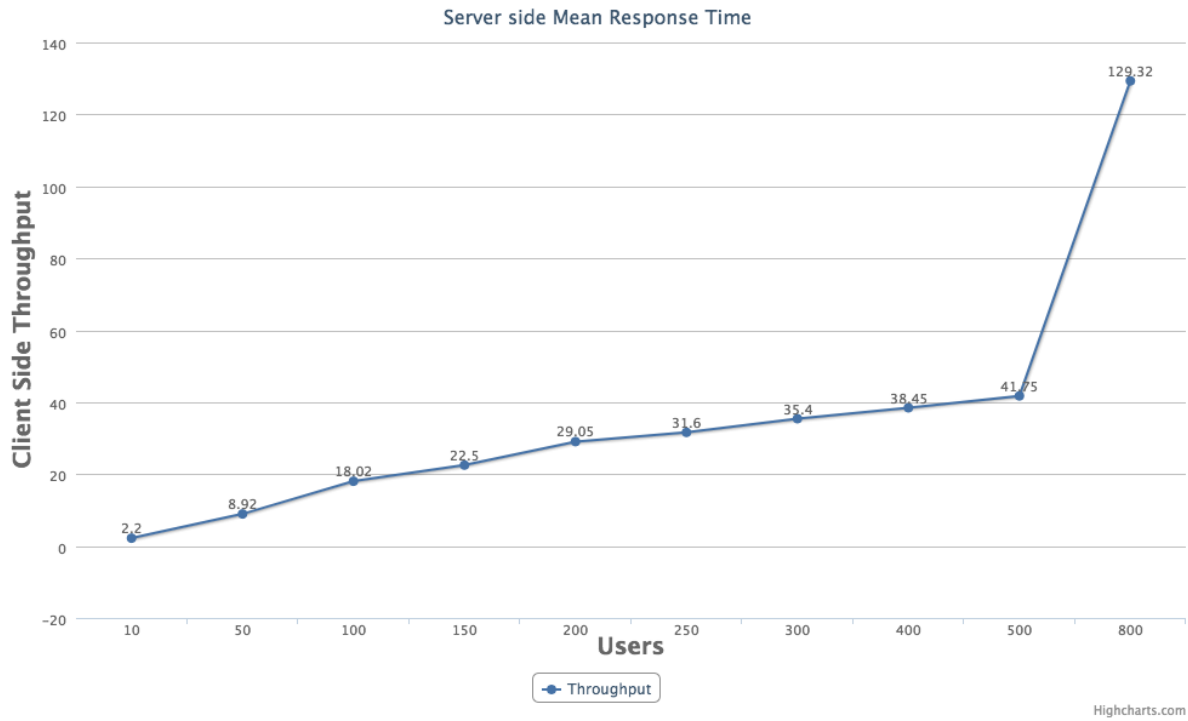


Figure 4.5 Client side Throughput

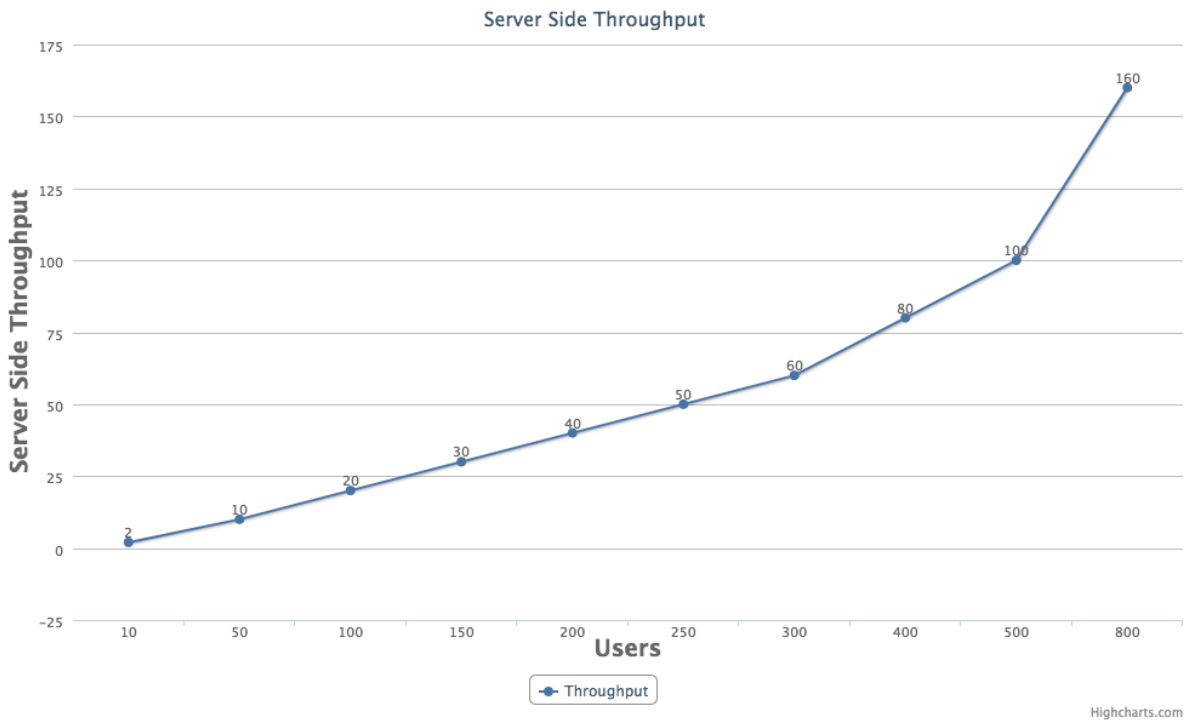


Figure 4.6 Client side Throughput

4.3 Number of requests in the web application

4.3.1 User Login

number of users	Average Response time [Sec]	Throughput[req/sec]	Q = R * X [req]
10	0.009	2.2	0.0198
50	0.010	10.2	0.102
100	0.008	20.1	0.1608
150	0.012	30	0.36
200	0.030	39.8	1.194
250	0.039	49.6	1.9344
300	0.036	59.6	2.1456
400	0.046	78.6	3.6156
500	0.081	97.9	7.9299
800	0.130	150.5	16.555

4.3.2 Add Book

users	Average Response time [Sec]	Throughput[req/sec]	Q = R * X [req]
10	0.013	2.2	0.286
50	0.011	10.2	0.102

100	0.012	20.1	0.2412
150	0.013	30	0.39
200	0.015	39.8	0.597
250	0.038	49.7	1.8886
300	0.054	59.4	3.2076
400	0.07	78.9	5.523
500	0.11	98.1	10.791
800	0.213	150.6	32.0778

4.3.3 Hold Book

users	Average Response time [Sec]	Throughput[req/sec]	Q = R * X [req]
10	0.08	2.2	0.176
50	0.011	10.2	0.1122
100	0.07	20.1	1.407
150	0.09	29.8	2.682
200	0.03	39.8	1.194
250	0.045	50	2.25
300	0.053	59.2	3.1376
400	0.074	78.6	5.8164
500	0.11	97.4	10.714
800	0.53	147.5	78.175

4.3.4 Search

users	Average Response time [Sec]	Throughput[req/sec]	Q = R * X [req]
10	0.091	2.2	0.2002
50	0.087	10	0.87
100	0.144	19.8	2.8512
150	0.959	27.2	26.0848
200	2.653	22	58.366
250	3.85	26.2	100.87
300	4.336	30.1	130.5136
400	5.672	29.9	169.5928
500	7.388	33	243.804
800	11.074	34.2	378.7308

4.3.5 Add Chart

users	Average Response time [Sec]	Throughput[req/sec]	Q = R * X [req]

10	0.046	2.2	0.1012
50	0.039	10.1	0.3939
100	0.045	19.9	0.8955
150	0.043	16.6	0.7138
200	0.047	22	1.034
250	0.053	27.6	1.4628
300	0.079	30.7	2.4253
400	0.077	43.6	3.3572
500	0.086	51.4	4.4204
800	0.102	75.3	7.6806

4.3.6 Pay

users	Average Response time [Sec]	Throughput[req/sec]	Q = R * X [req]
10	0.032	2.2	0.0704
50	0.036	10	0.36
100	0.055	20	1.1
150	0.052	29.5	1.534
200	0.514	22.1	11.3594
250	1.399	49.2	68.8308
300	2.041	52.2	106.5402
400	4.416	37.9	167.3664
500	6.431	48.6	312.5466
800	10.292	34.3	353.0156

4.3.7 Graph Search

users	Average Response time [Sec]	Throughput[req/sec]	Q = R * X [req]
10	0.123	2.2	0.2706
50	1.692	5.6	9.4752
100	2.968	12.4	36.8032
150	4.375	16.7	73.0625
200	6.118	9.6	58.7328
250	7.206	13.1	94.3986
300	7.901	13.4	105.8734
400	10.521	15.5	163.0755
500	13.686	20.8	284.6688

800	20.563	23.2	477.0616

4.4 Utilization of the system

4.4.1 User login

number of users	B [sec]	τ [sec]	U = B/τ
10	0.257	5	0.0514
50	0.448	5	0.0896
100	1.029	5	0.2058
150	1.266	5	0.2532
200	1.506	5	0.3012
250	4.275	5	0.855
300	5.574	5	1.1148
400	9.873	5	1.9746
500	14.283	5	2.8566
800	37.117	5	7.4234

4.4.2 Add Book

number of users	B [sec]	τ [sec]	U = B/τ
10	0.223	5	0.0446
50	1.201	5	0.2402
100	2.241	5	0.4482
150	3.642	5	0.7284
200	7.117	5	1.4234
250	8.478	5	1.6956
300	9.618	5	1.9236
400	16.615	5	3.323
500	37.786	5	7.5572
800	90.05	5	18.01

4.4.3 Hold Book

number of users	B [sec]	τ [sec]	U = B/τ
10	0.339	5	0.0678
50	0.901	5	0.1802
100	2.685	5	0.537
150	4.15	5	0.83
200	8.984	5	1.7968
250	10.402	5	2.0804
300	14.564	5	2.9128
400	27.737	5	5.5474
500	57.175	5	11.435
800	153.2	5	30.64

4.4.4 Add Cart

number of users	B [sec]	τ [sec]	U = B/τ
10	0.43	5	0.086
50	1.794	5	0.3588
100	3.925	5	0.785
150	6.252	5	1.2504
200	13.675	5	2.735
250	21.344	5	4.2688
300	32.105	5	6.421
400	56.684	5	11.3368
500	80.673	5	16.1346
800	161.234	5	32.2468

4.4.5 Search

number of users	B [sec]	τ [sec]	U = B/τ
10	0.903	5	0.1806
50	4.18	5	0.836
100	14.11	5	2.822

150	139.362	5	27.8724
200	469.301	5	93.8602
250	853.867	5	170.7734
300	1178.55	5	235.71
400	1696.973	5	339.3946
500	2289.419	5	457.8838
800	3619.513	5	723.9026

4.4.6 Pay

number of users	B [sec]	τ [sec]	U = B/τ
10	0.5	5	0.1
50	2.062	5	0.4124
100	8.355	5	1.671
150	30.057	5	6.0114
200	86.44	5	17.288
250	355.258	5	71.0516
300	461.148	5	92.2296
400	1486.839	5	297.3678
500	1691.723	5	338.3446
800	3490.433	5	698.0866

4.4.7 Graph search

number of users	B [sec]	τ [sec]	U = B/τ
10	0.121	5	0.0242
50	84.504	5	16.9008
100	296.072	5	59.2144
150	468.943	5	93.7886
200	1207.512	5	241.5024
250	1685.891	5	337.1782
300	2095.632	5	419.1264
400	3026.593	5	605.3186
500	3848.1	5	769.62
800	5779.56	5	1155.912

4.5 Throughput of Database

Since on average one operation will cause two calls to the database, X(db) = V *X(sys) according to Forced Flow Law. Therefore we get the following graph:

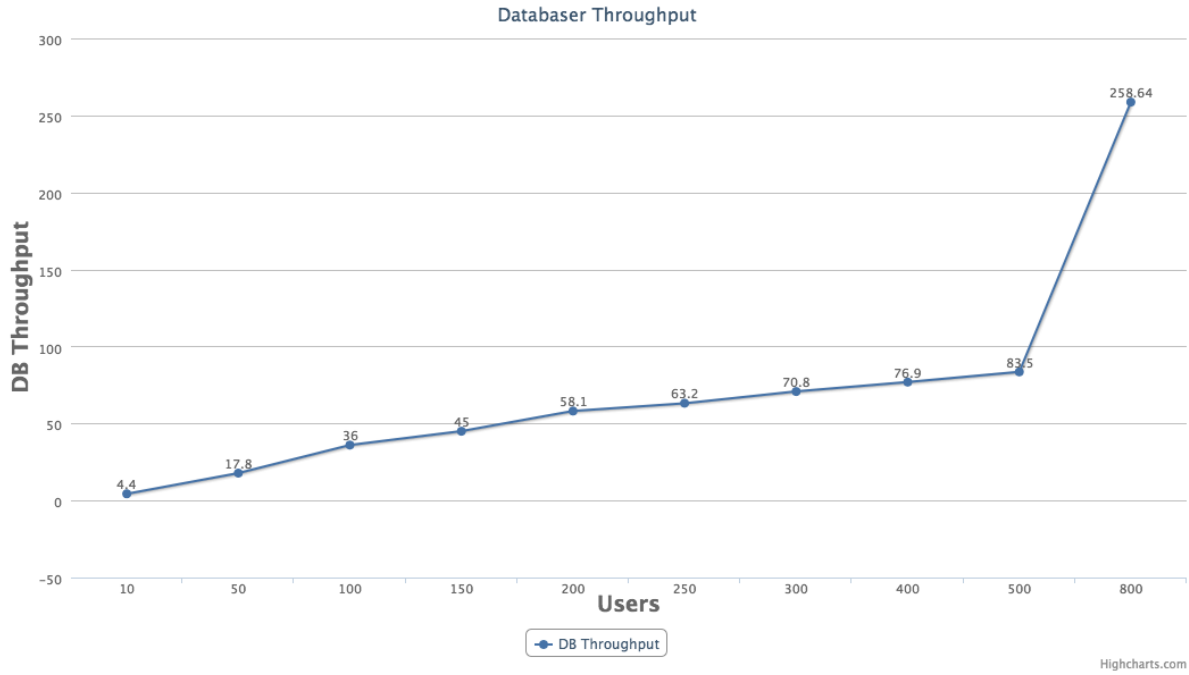


Figure 4.7 Database Throughput

5 Discussion

5.1 Performance analysis

5.1.1 User login

When it reaches 400 users per 5 seconds, the avg Response time starts to ascend and accord to the part 4.3 the number of users in application, it is clear that when the number of users is less than 200, there will be few people in the queue and until 500 users can the system hold the requests made by those users. In general , the function of log in is rather efficient because it only involve one time of calling database.

5.1.2 Add Book

From Figure 4.1, the avg Response time starts to grow when it comes to 250 users per 5 seconds. And from Part 4.3, we can see there is hardly a user in the queue when the number of users is less than 200. And the bottleneck should be around 300 users per 5 seconds since the utilisation of system is more than 100% and the avg response time starts to grow linearly.

5.1.3 Hold Book

The average response time of holding books is very steady before 200 users per 5 seconds. But it grows distinctly after 200 users per 5 seconds. But its efficiency is not so good as “Add Book” because when more than 100 users, the number of users in system is 1.4. When it reaches 150 users per 5 seconds, the utilisation of system is nealy 100%.

5.1.4 Search

This operation is rather time consuming because when the number of users reaches 50, the queue is not quite empty: the number of users in system is around 0.8. And the utilisation nearly reaches 100% when the users is around 60.

5.1.5 Pay

The average response time of this operation goes smoothly from 10 to 100 users. But after that it grows dramatically. And the utilisation of systems has already reached 100% when it comes to 100 users per seconds. Therefore, when the number of users is more than 100 users, the running state will become not so stable.

5.1.6 Add cart

This operation goes well before the number of users reaches 250. But the utilisation becomes more than 100% when the number of users is 150.

5.1.7 Graph search

This operation is the most time consuming one. Its average response time is far more than other operations from the beginning 10 users. When there are only 20 users, the queue starts to grow dramatically,and so is the utilisation of system, which reaches 100% from the very beginning of the test.

5.2 Slow operations analysis

5.2.1 Search

On the server side,due to the large size of book table, database search will take more time than other db operation, such as insert. Meanwhile on client side, when servlet return all the records, the JSP page need to form all info into http elements and use JPage to realise page function, which means client receive all the results when search is done, if number of returned value is huge it will be very slow to load this data.

5.2.2 Pay

Once user confirm to pay, the server not only has to update all the book quantity in book table and empty this user’s shopping cart but also need to send inform email to book seller(if exists) and insert user activity into related table. All these actions together will take long time on the server side, which also cost long waiting time for clients.

5.2.3 Graph search

Due to the structure designed in the entity and relation table, when user enter a key word in the search bar, database will first find all the entity and then for each entity, it will go through the whole relation table to find all related relation. For example, in our experiment, there are 5 entity of article with key word ‘web development’, and in database, there are 18408 tuples together , so actually database will scan 5 * 18408 times.

5.3 Possible Improvement

5.3.1 Client side improvement

For book search part we can obvious find that when servlet send all request to user it takes lots of space and time for jsp page to load this data, generate html elements and then present to user. We could add another function on the servlet side to receive page number that return only 10 result on each page every time, this would reduce the client side response time.
All for all, for other part in our application, we should also follow this rule, that simplify the jsp page, short website loading time, like reduce page size, remove unnecessary elements and etc.

5.3.2 Database interaction

Previous part shows that the most two cost-time parts of our application is graph search and book search, because these two operations need to scan large amount of data in database. One way to solve this problem is to improve our database schema design. For example, in the current database, we put all relation in one table, and for every entity the data base need to scan whole relation table to get result. We could simply split the table into three: author relation, publication relation and venue relation. When user select a particular search type database will only search in this kind relation table not others. This may cost space redundancy but will decrease processing time in database. So is book search, don’t put all book in one table but each publication one table.

5.4 Conclusion

In this report we have analysed the performance of the Online Book Store Web Application which we developed for assignment 2 in COMP9321. During this performance evaluation experiment, we understand the structure and pattern of our app better. More importantly, this test helps us find slowest part and some shortcakes of application, then inspires us to think of possible improvement to make our app work better.

Part3 link: https://www.youtube.com/watch?v=pYZwO5a1NBw