Performance Analysis of Social Media Web Application
Group Name: Eat Sleep Code and Repeat
Members: Andrew Wirjaputra, Cheng Yin, Weitao Wang, Xingjian Guo, Zhenyu Ye, Zhi He

1. Introduction

This report provides test results and analysis of the performance of the Social Media application developed in assignment 2 of COMP9321. We will start by describing how the test was set up in order to produce consistent and measurable results. We will then describe how each relevant components of the application are designed in order to serve user requests. And lastly, we will analyze the performance of the application based on our test results and discuss possible performance improvements.

2. Test Setup

A Sony VAIO SVS15126PGB running Apache Tomcat 8.045 was used as the Server and a SONY VAIO SVS15116GGB running JMeter was used as the Client.

2.1 Server Details

Hardware variable	Details
Processor	Intel® Core™ i7-3632QM CPU @ 2.20GHz
RAM	12.0 GB
System type	64-bit

Table 1: Hardware details

Operating variable	Details
Operating System	Windows 8.1 Pro
Java	1.8.0.144
Apache Tomcat Server	8.0.45
MySQL	5.7.19 (1 connection)

Table 2: Operating Environment

2.2 Client Details

Hardware variable	Details
Processor	Intel® Core™ i7-3612QM CPU @ 2.10GHz
RAM	4.0 GB
System type	64-bit

Table 3: Hardware details

2.3 Details of Test Plan

Both the Server and Client are connected to the same network to minimize disturbance. For the simulation, the following scenarios are executed in the order listed below with a delay of 5 seconds between each user action:

1. A user logs into the system.
2. The user search for other users.
3. The user opens another user profile.

4. The user adds the other user as friend.
5. The user posts a new message on the wall.

By using the “Thread Group” feature in JMeter, we can construct a series of HTTP requests to mimic the actions of an actual User. The test was conducted incrementally for a wide range of concurrent users: 5, 10, 20, 25, 40, 50, 75, 80, 100, and 200, while constantly monitoring the error rate in JMeter. The test is stopped when the error rate is higher than 2%.

The exact testing procedure can be seen below:

1. Recreate and repopulate all databases with dummy user data.
2. Reload the web application using PSI Probe.
3. Load the web application in the web browser (1st request takes longer time).
4. Restart JMeter and clear aggregate report.
5. Run JMeter test.

3. Application Details

3.1 User Login

A user can simply login into the system by inputting the correct username and password in the login page. After the user pressed the login button, calls will be made to the database in order to extract the specified user information.

The login process is show in Figure 1. The first call to the database is used to make sure that user with the specified username exists, and that his/her account is verified and not banned. The second call to the database is used to make sure that the specified password is correct. If any of these terms is violated, the page will be redirected to the login page, showing the respective error notification. Upon successful login, the specified user session will be opened and the user is redirected into their home page.

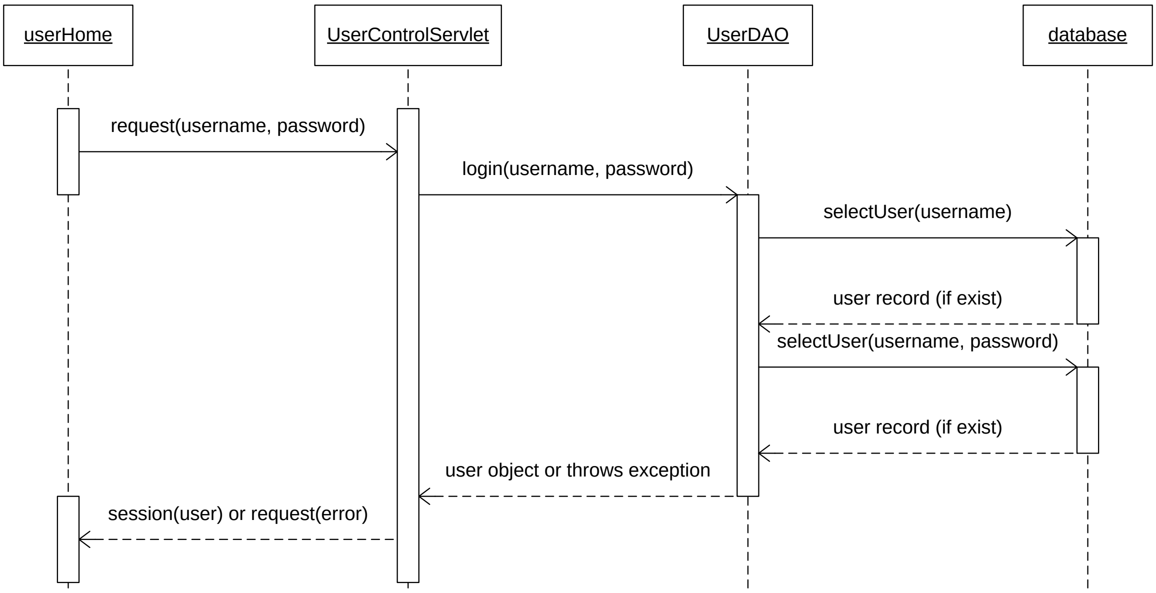

userHome
UserControlServlet
UserDAO
database
request(username, password)
login(username, password)
selectUser(username)
user record (if exist)
selectUser(username, password)
user record (if exist)
user object or throws exception
session(user) or request(error)

Figure 1: User Login Process

3.2 User Search

A user can search for other users based on their firstname, surname, gender, or date of birth. Simply select one from the dropdown menu at the top of the page, input the search value, and click on the search button. The system will then query the database based on the specified search information and display the result page. Details of the search process can be seen in Figure 2 below.

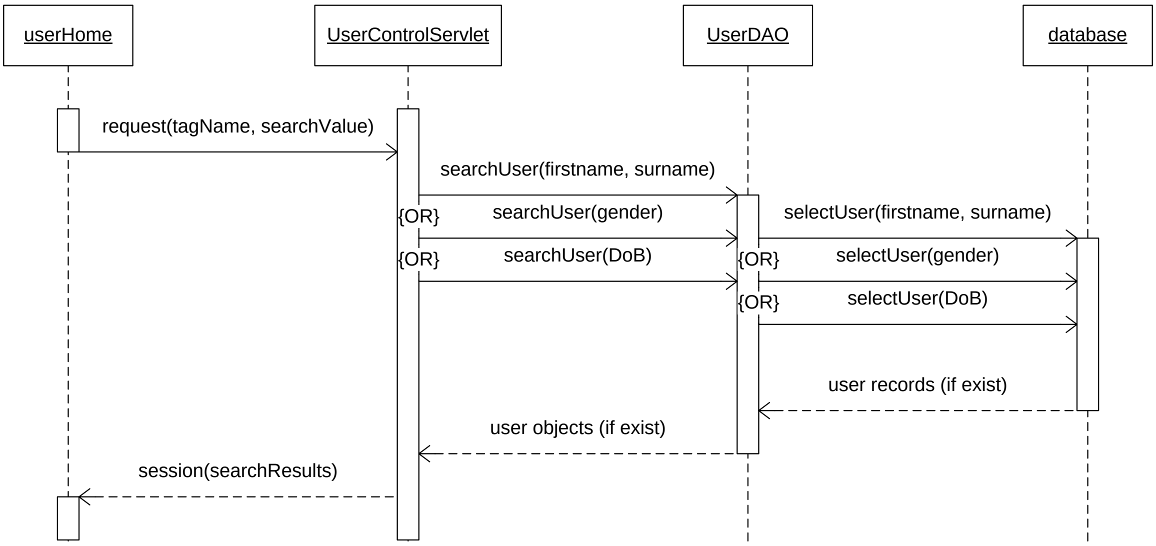


Figure 2: User Search Process

3.3 Other User Profile

On the search result page, user can select any of the users found to view their profile. A query will then be sent to the database to select user with the specified username (username is unique). The other user object will be created based on the query result, and the page will redirect to the other user profile, displaying their information based on the other user object.

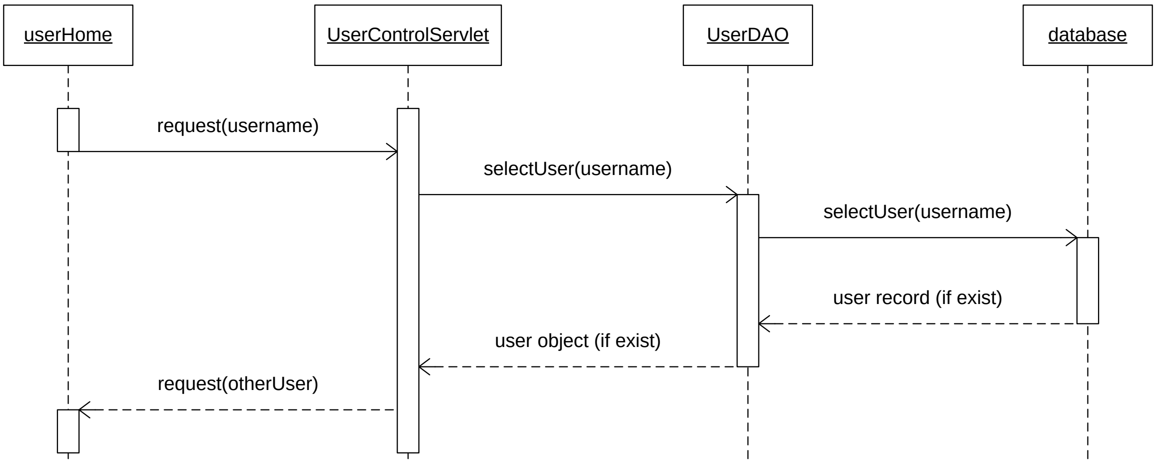


Figure 3: Process of Opening another User Profile

3.4 Add Friend

An add friend button will be displayed in another user profile, if the other user is not yet in your friend list. By clicking the add friend button, a query will be sent to the database to find user with the specified username and email. If the user information exists, this add friend action will then be recorded in the log database, and a friend request email will be sent to the other user.

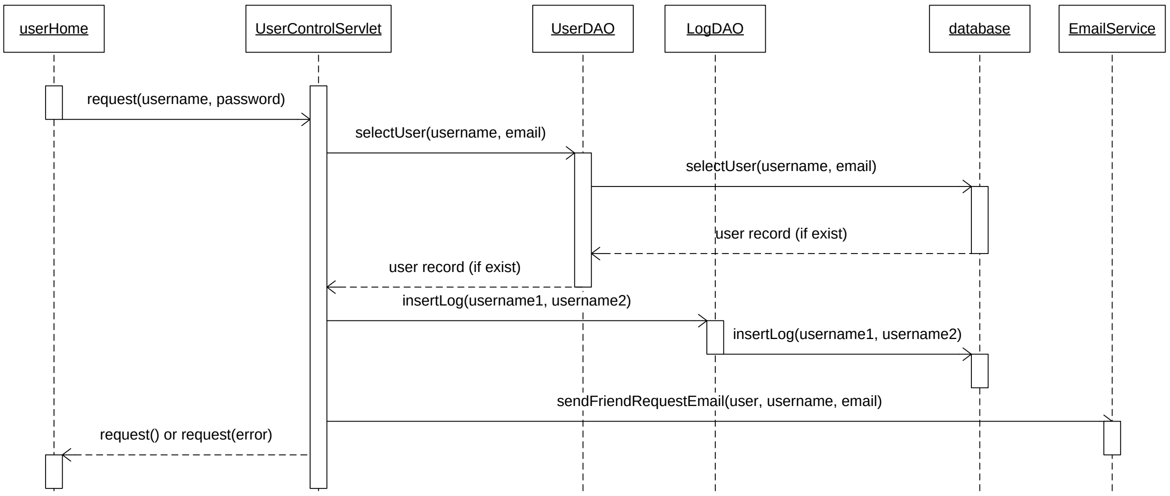

userHome
UserControlServlet
UserDAO
LogDAO
database
EmailService
request(username, password)
selectUser(username, email)
selectUser(username, email)
user record (if exist)
user record (if exist)
insertLog(username1, username2)
insertLog(username1, username2)
sendFriendRequestEmail(user, username, email)
request() or request(error)

Figure 4: Add Friend Process

3.5 Post Message

Users can post messages in their wall for their friends to see. When a user post a message, their username and the content of the message will be saved in the post database. Since the post ID is automatically incremented by the database, another query will be sent to retrieve the post ID, which will then be used to record the post action in the log database.

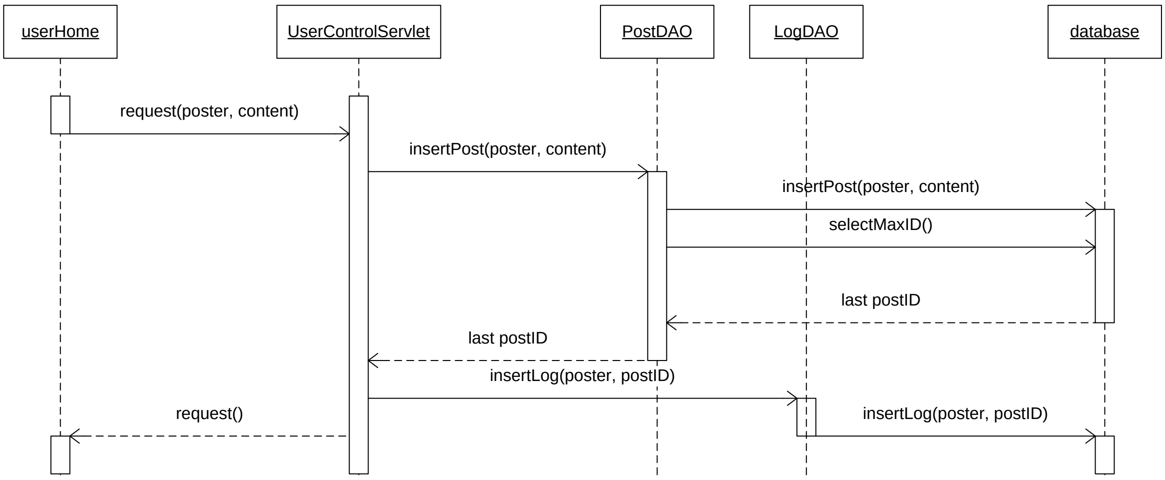
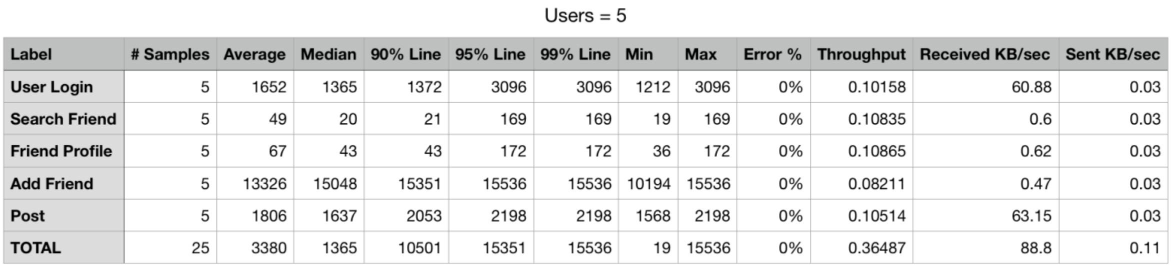


Figure 5: Post Message Process

4. Results and Discussion

4.1 JMeter Results

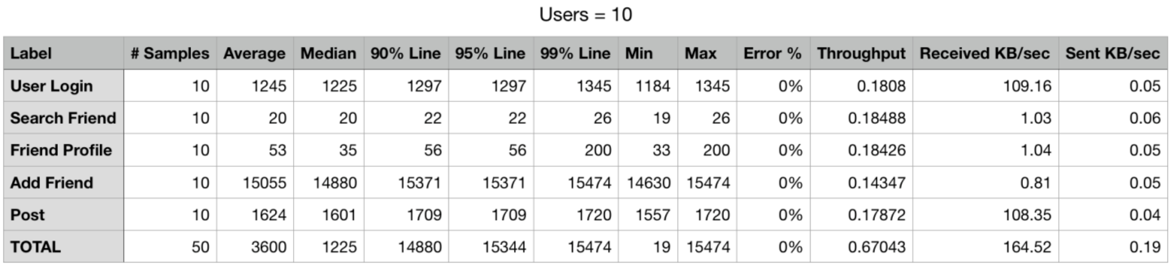
Users = 5, time = 68 seconds



Users = 5

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
User Login	5	1652	1365	1372	3096	3096	1212	3096	0%	0.10158	60.88	0.03
Search Friend	5	49	20	21	169	169	19	169	0%	0.10835	0.6	0.03
Friend Profile	5	67	43	43	172	172	36	172	0%	0.10865	0.62	0.03
Add Friend	5	13326	15048	15351	15536	15536	10194	15536	0%	0.08211	0.47	0.03
Post	5	1806	1637	2053	2198	2198	1568	2198	0%	0.10514	63.15	0.03
TOTAL	25	3380	1365	10501	15351	15536	19	15536	0%	0.36487	88.8	0.11

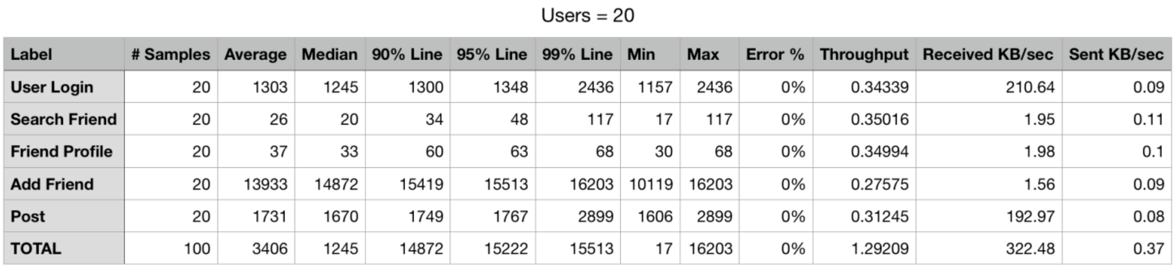
Users = 10, time = 74 seconds



Users = 10

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
User Login	10	1245	1225	1297	1297	1345	1184	1345	0%	0.1808	109.16	0.05
Search Friend	10	20	20	22	22	26	19	26	0%	0.18488	1.03	0.06
Friend Profile	10	53	35	56	56	200	33	200	0%	0.18426	1.04	0.05
Add Friend	10	15055	14880	15371	15371	15474	14630	15474	0%	0.14347	0.81	0.05
Post	10	1624	1601	1709	1709	1720	1557	1720	0%	0.17872	108.35	0.04
TOTAL	50	3600	1225	14880	15344	15474	19	15474	0%	0.67043	164.52	0.19

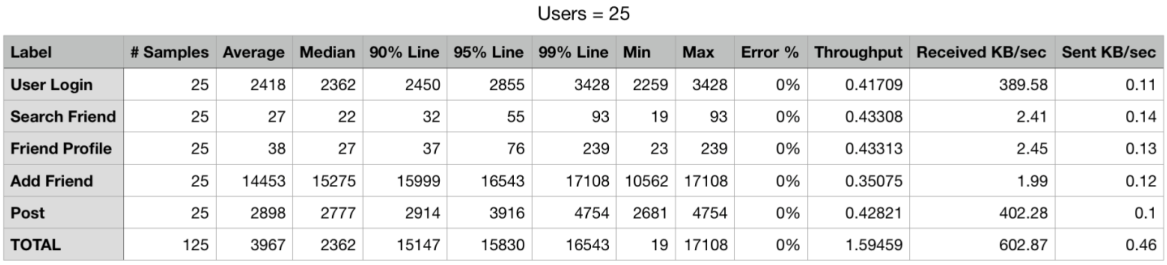
Users = 20, time = 77 seconds



Users = 20

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
User Login	20	1303	1245	1300	1348	2436	1157	2436	0%	0.34339	210.64	0.09
Search Friend	20	26	20	34	48	117	17	117	0%	0.35016	1.95	0.11
Friend Profile	20	37	33	60	63	68	30	68	0%	0.34994	1.98	0.1
Add Friend	20	13933	14872	15419	15513	16203	10119	16203	0%	0.27575	1.56	0.09
Post	20	1731	1670	1749	1767	2899	1606	2899	0%	0.31245	192.97	0.08
TOTAL	100	3406	1245	14872	15222	15513	17	16203	0%	1.29209	322.48	0.37

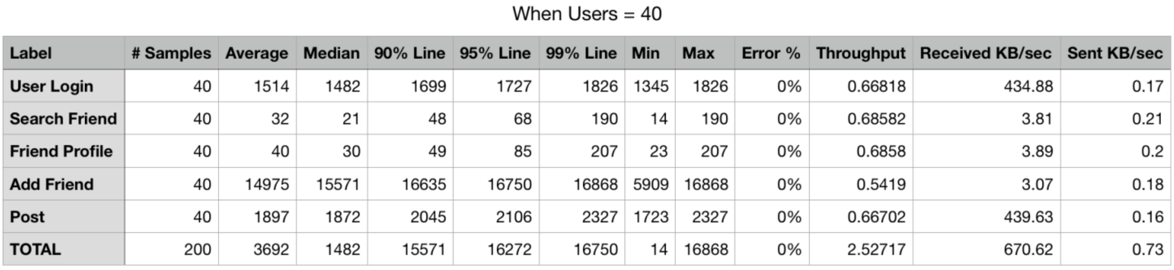
Users = 25, time = 78 seconds



Users = 25

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
User Login	25	2418	2362	2450	2855	3428	2259	3428	0%	0.41709	389.58	0.11
Search Friend	25	27	22	32	55	93	19	93	0%	0.43308	2.41	0.14
Friend Profile	25	38	27	37	76	239	23	239	0%	0.43313	2.45	0.13
Add Friend	25	14453	15275	15999	16543	17108	10562	17108	0%	0.35075	1.99	0.12
Post	25	2898	2777	2914	3916	4754	2681	4754	0%	0.42821	402.28	0.1
TOTAL	125	3967	2362	15147	15830	16543	19	17108	0%	1.59459	602.87	0.46

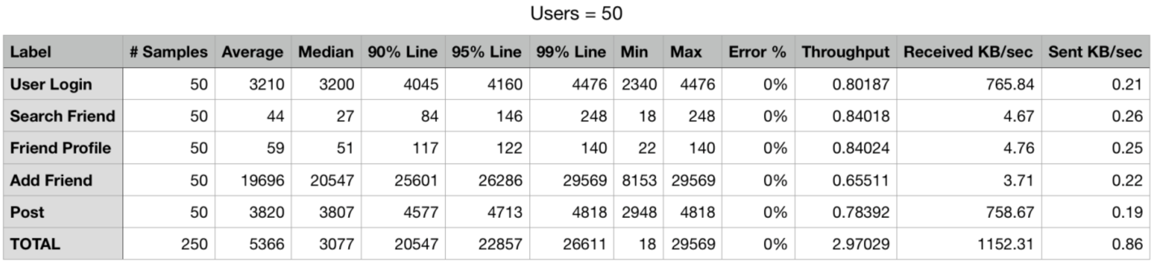
Users = 40, time = 80 seconds



When Users = 40

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
User Login	40	1514	1482	1699	1727	1826	1345	1826	0%	0.66818	434.88	0.17
Search Friend	40	32	21	48	68	190	14	190	0%	0.68582	3.81	0.21
Friend Profile	40	40	30	49	85	207	23	207	0%	0.6858	3.89	0.2
Add Friend	40	14975	15571	16635	16750	16868	5909	16868	0%	0.5419	3.07	0.18
Post	40	1897	1872	2045	2106	2327	1723	2327	0%	0.66702	439.63	0.16
TOTAL	200	3692	1482	15571	16272	16750	14	16868	0%	2.52717	670.62	0.73

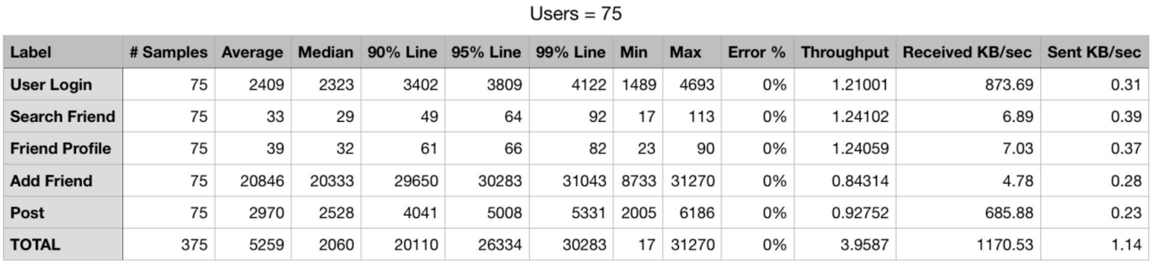
Users = 50, time = 85 seconds



Users = 50

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
User Login	50	3210	3200	4045	4160	4476	2340	4476	0%	0.80187	765.84	0.21
Search Friend	50	44	27	84	146	248	18	248	0%	0.84018	4.67	0.26
Friend Profile	50	59	51	117	122	140	22	140	0%	0.84024	4.76	0.25
Add Friend	50	19696	20547	25601	26286	29569	8153	29569	0%	0.65511	3.71	0.22
Post	50	3820	3807	4577	4713	4818	2948	4818	0%	0.78392	758.67	0.19
TOTAL	250	5366	3077	20547	22857	26611	18	29569	0%	2.97029	1152.31	0.86

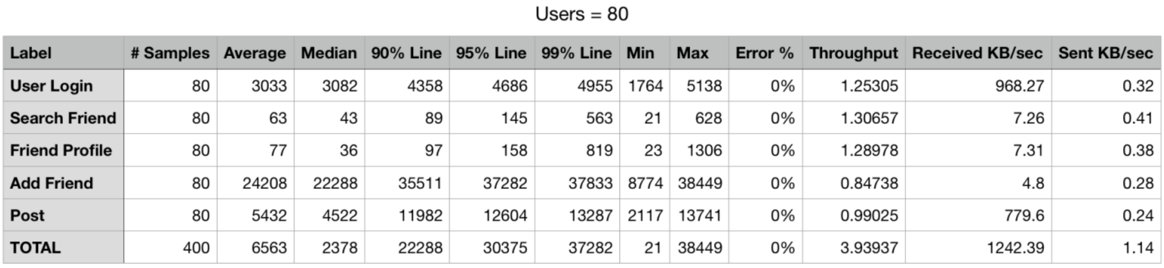
Users = 75, time = 94 seconds



Users = 75

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
User Login	75	2409	2323	3402	3809	4122	1489	4693	0%	1.21001	873.69	0.31
Search Friend	75	33	29	49	64	92	17	113	0%	1.24102	6.89	0.39
Friend Profile	75	39	32	61	66	82	23	90	0%	1.24059	7.03	0.37
Add Friend	75	20846	20333	29650	30283	31043	8733	31270	0%	0.84314	4.78	0.28
Post	75	2970	2528	4041	5008	5331	2005	6186	0%	0.92752	685.88	0.23
TOTAL	375	5259	2060	20110	26334	30283	17	31270	0%	3.9587	1170.53	1.14

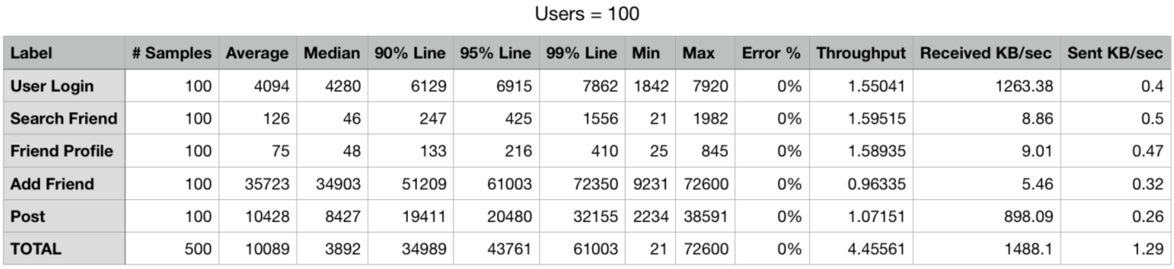
Users = 80, time = 102 seconds



Users = 80

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
User Login	80	3033	3082	4358	4686	4955	1764	5138	0%	1.25305	968.27	0.32
Search Friend	80	63	43	89	145	563	21	628	0%	1.30657	7.26	0.41
Friend Profile	80	77	36	97	158	819	23	1306	0%	1.28978	7.31	0.38
Add Friend	80	24208	22288	35511	37282	37833	8774	38449	0%	0.84738	4.8	0.28
Post	80	5432	4522	11982	12604	13287	2117	13741	0%	0.99025	779.6	0.24
TOTAL	400	6563	2378	22288	30375	37282	21	38449	0%	3.93937	1242.39	1.14

Users = 100, time = 113 seconds



Users = 100

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
User Login	100	4094	4280	6129	6915	7862	1842	7920	0%	1.55041	1263.38	0.4
Search Friend	100	126	46	247	425	1556	21	1982	0%	1.59515	8.86	0.5
Friend Profile	100	75	48	133	216	410	25	845	0%	1.58935	9.01	0.47
Add Friend	100	35723	34903	51209	61003	72350	9231	72600	0%	0.96335	5.46	0.32
Post	100	10428	8427	19411	20480	32155	2234	38591	0%	1.07151	898.09	0.26
TOTAL	500	10089	3892	34989	43761	61003	21	72600	0%	4.45561	1488.1	1.29

Users = 200, time = 201 seconds

Users = 200

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
User Login	200	29992	32322	46853	49437	53628	2525	55375	7.5%	2.24009	1779.84	0.58
Search Friend	185	1988	365	3896	5512	39153	23	43478	4.324%	2.13185	11.54	0.67
Friend Profile	177	824	279	1229	3308	8455	28	21334	1.13%	2.04006	11.47	0.6
Add Friend	175	72551	68875	102489	106538	135434	16874	136896	0%	1.07939	6.12	0.36
Post	175	26038	13329	51180	54980	62896	2456	66357	0%	1.3934	1267.36	0.34
TOTAL	912	26058	12456	68738	89698	106538	23	136896	2.741%	4.52844	1594.22	1.3

4.2 Response Times

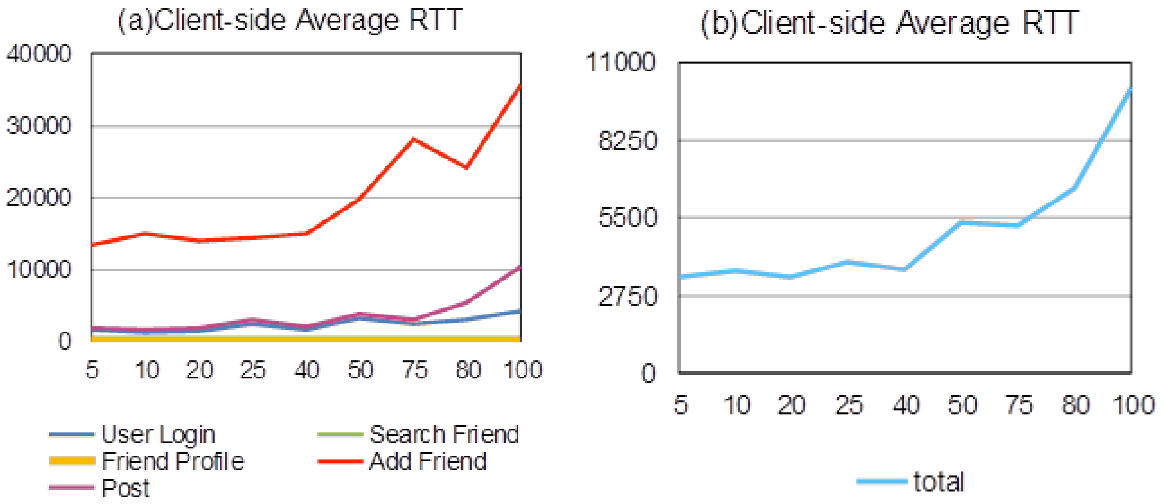


Figure 6: Figure a shows the average response time of each type of user action, for each set of users. Figure b shows the average response time of all user actions, for each set of users.

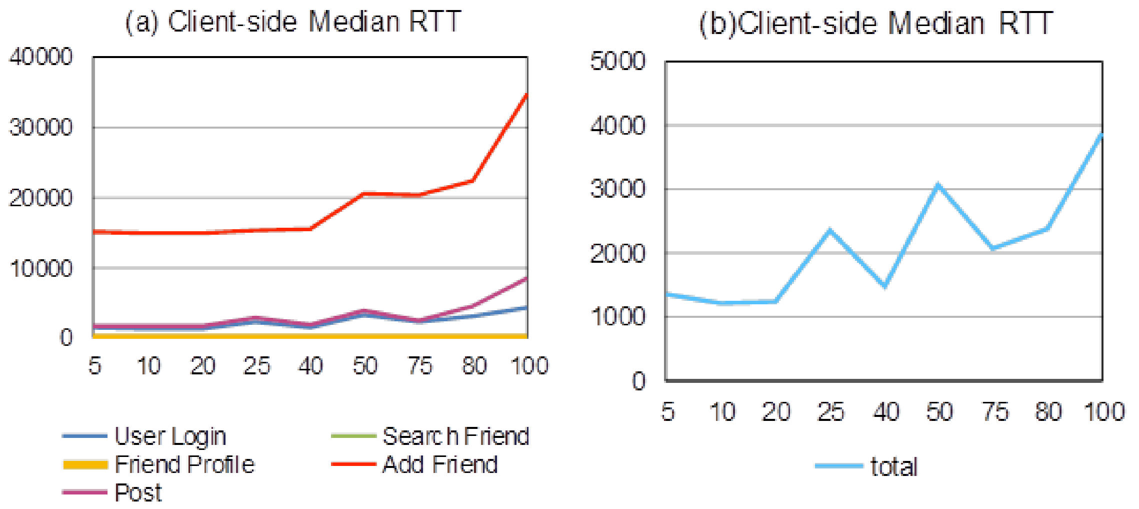


Figure 7: Figure a shows the median response time of each type of user action, for each set of users. Figure b shows the median response time of all user actions, for each set of users.

4.3 Throughput

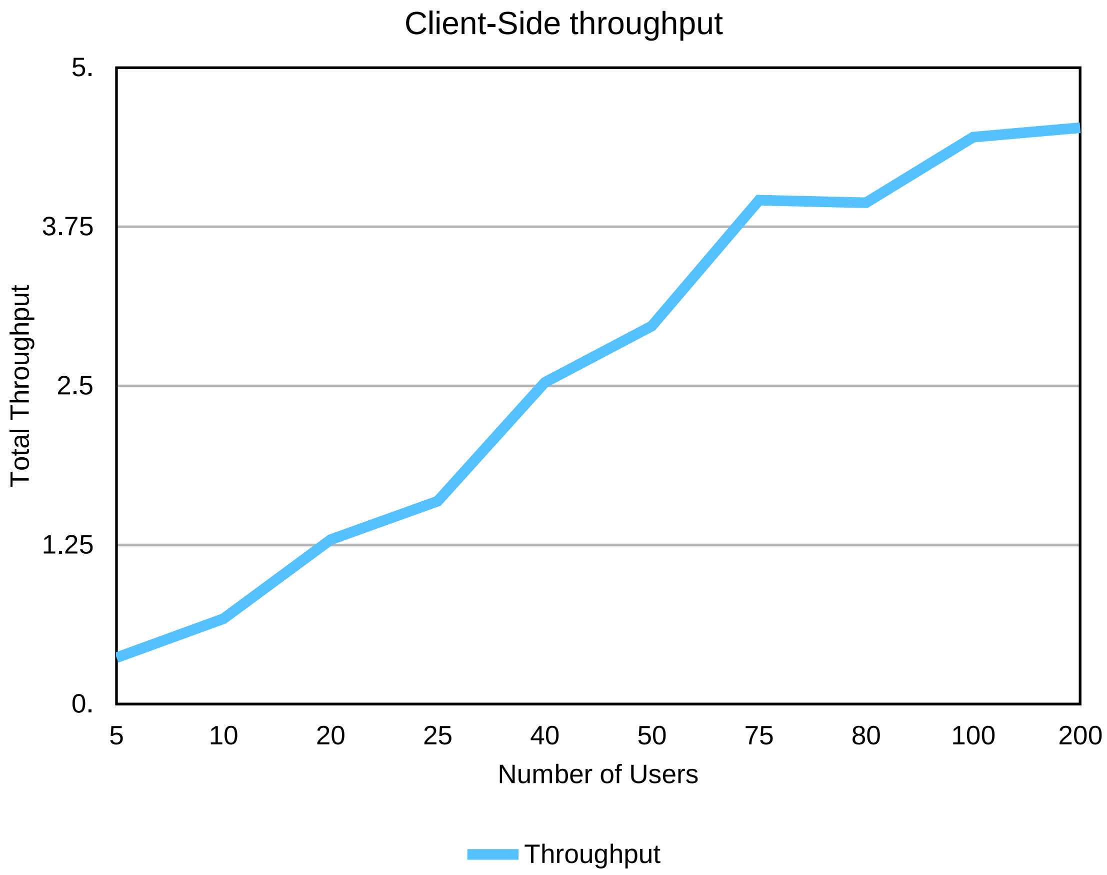


Figure 8: The client-side throughput for each set of users.

4.4 Number of requests in the web application

No. of Users	R (sec)	X (req/sec)	Q = R x X (req)
5	3.380	0.36487	1.23326
10	3.600	0.67043	2.41355
20	3.406	1.29209	4.40086
25	3.967	1.59459	6.32574
40	3.692	2.52717	9.33031
50	5.366	2.97029	15.93858
75	5.259	3.95870	20.81880
80	6.563	3.93937	25.85409
100	10.089	4.45561	44.95265
200	26.058	4.52844	118.00209

Table 4: Number of requests in the web application (average queue length), computed using Little's Law

4.5 Utilization of Tomcat Container and System

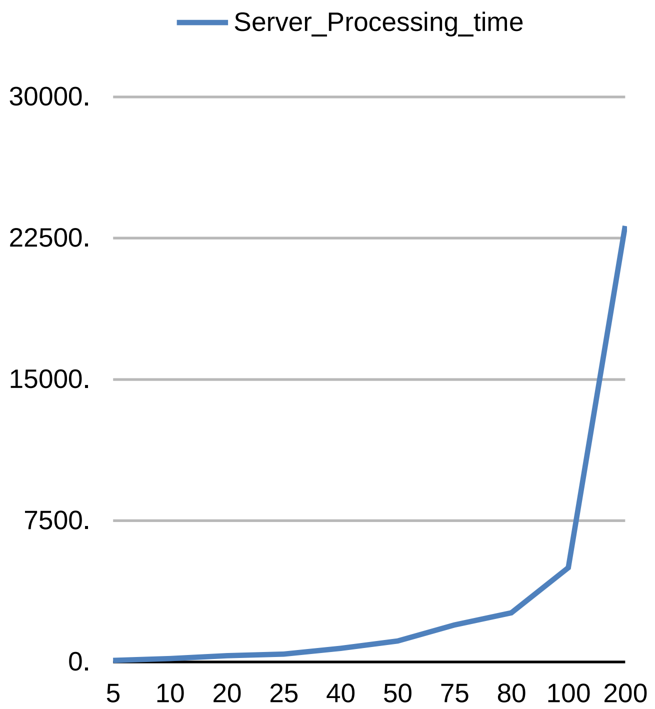


Figure 9: The server-side processing time for each set of users.

No. of Users	B (sec)	τ (sec)	U = B / τ
5	82.267	68	120.98%
10	178.877	74	241.73%
20	338.158	77	439.17%
25	416.672	78	534.19%
40	729.418	80	911.77%
50	1107.679	85	1303.15%
75	1961.618	94	2086.83%
80	2603.689	102	2552.64%
100	5012.057	113	4435.45%
200	23150.532	201	11517.68%

Table 5: Utilization of Tomcat Container and System. B is the total processing time given by PSI probe. τ is the time it took to run the tests.

4.6 Throughput of Database

No. of Users	V (avg. no. of visits)	X_{sys} (req/sec)	$X_{db} = V \times X_{sys}$ (req/sec)
5	1.8	0.36487	0.65677
10	1.8	0.67043	1.20677
20	1.8	1.29209	2.32576
25	1.8	1.59459	2.87026
40	1.8	2.52717	4.54891
50	1.8	2.97029	5.34652
75	1.8	3.95870	7.12566
80	1.8	3.93937	7.09087
100	1.8	4.45561	8.02010
200	1.795	4.52844	8.12855

Table 6: Throughput of the database, computed using Forced Flow Law.

4.7 Analysis of the System Utilization

We have made sure that all unnecessary process on the server was ended before conducting the tests. The CPU usage on the server before the tests commenced is observed at ~2%. The CPU usage caps at ~54% for all test cases, peaking under heavy load, as one would expect. We believe that this constant maximum CPU usage is due to the processing limit of the server. So the number of simultaneous process on the server remains the same regardless of how many users are tested, although queue will build up faster when testing more users.

4.8 Analysis of RTT and the Application Capacity

RTT is an important metric for measuring performance in terms of user experience. RTT for individual requests measures the amount of time a user spends waiting between pages. The sum of these RTTs yields the time a user spends inactive while navigating the application. Based on figure 6.a and 6.b, we can see that the RTT is good for up to 80 users. The average and median RTT increase rapidly beyond 80 users, whereas the gap between the minimum and maximum RTT becomes wider and wider. The increase in response time is linked with the queue on the server (Table 6). As the average queue in the server grows, the response time observed in the client will also increase.

4.9 Analysis of the Different Steps in the Operation

As we expected, the step in the operation that takes the longest time is the add friend function. This action is the only one from the five that involves sending an email. Even when testing the application ourselves, we notice the page taking a few seconds to load after clicking the add friend button. For the rest of the actions, the ones that have a larger number of database queries (see application details) will take more time than the others.

4.10 Error Analysis

When testing with 200 users, we recorded some errors. The most obvious one is the add friend action, in which we received less than 100 emails. The number of posted messages added to the database is also less than the number of samples observed in JMeter. These errors are likely due to the email and database server not being able to accommodate the quick succession of requests.

4.11 Possible Performance Improvements

- **Multithreading email**
  Seeing that this action is the one that's taking the longest time, we believe that performance improvements could be best achieved by improving how email requests are handled. Since the performance of this part of the system is reliant on the performance of the mail server (which is out of our hands), the solution that we can come up with is threading and timing the upcoming email requests. By doing so, simultaneous email requests will be queued on the application server and sent to the mail server in a timely manner, therefore preventing the mail server from dropping the requests (or at least minimize it).
- **Database Interaction, reducing redundancy**
  Another performance improvement could be done on the data access layer, by creating a more efficient query and by reducing redundancy in the database itself.
- **Caching search results**

5. Conclusion

In this report we have analyzed the performance of the Social Media Web Application which we developed for assignment 3 in COMP9321. Our analysis found that the web application worked well up to around 80 concurrent users, the average response time will increase rapidly afterwards. Our tests also revealed that the application made some errors when the user load increased beyond 100 concurrent users. These errors cause several email and database access requests to be dropped. Furthermore, we found that the bottleneck of our application is the email service. Actions that involved more database queries will also take more time. Based on this, our best suggestion for improving performance is by multithreading the email requests on the application server side. This will prevent or at least minimize the amount of requests dropped by the mail server, in which we have no control of.

You can check our Solution Video in the link below:
https://www.youtube.com/watch?v=ocqsKAYf-ts