

**Project**  
**on**  
**Client to Client Chat System**

**Avais Ahmad**

173050043

**Abhijit Patil**

173050085

October 9, 2017

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Architecture Overview</b>	<b>4</b>
2.1	Architecture Diagram . . . . .	4
<b>3</b>	<b>Types Of Request</b>	<b>5</b>
<b>4</b>	<b>Load Generator</b>	<b>7</b>
4.1	System View . . . . .	7
4.2	System Flow . . . . .	7
4.3	Load Test . . . . .	7
4.4	Observations . . . . .	8
4.4.1	Registration Request . . . . .	8
4.4.2	Login Request . . . . .	10
4.5	Screen shots . . . . .	12
<b>5</b>	<b>Conclusion</b>	<b>17</b>

# 1 Introduction

Our project provides a platform for client to client chatting in a very efficient manner by using multi-threading and socket programming. In this two clients can chat after authentication from server. User have to register using a user name and password. A unique port number will be assigned to it by the server which will be later used for creating a new connection for chatting. User name should be unique. After registration client have to login. Once his user-name and password are verified by the server he can either wait for other clients to connect or he can request other client who is waiting To request other client to connect the client have to enter the user-name of the client to whom he wants to chat . Server will send the port of that client. Using this port a new socket can be created for both clients to chat. Both client have to send message “bye” to end the chat connection.

## 2 Architecture Overview

### 2.1 Architecture Diagram

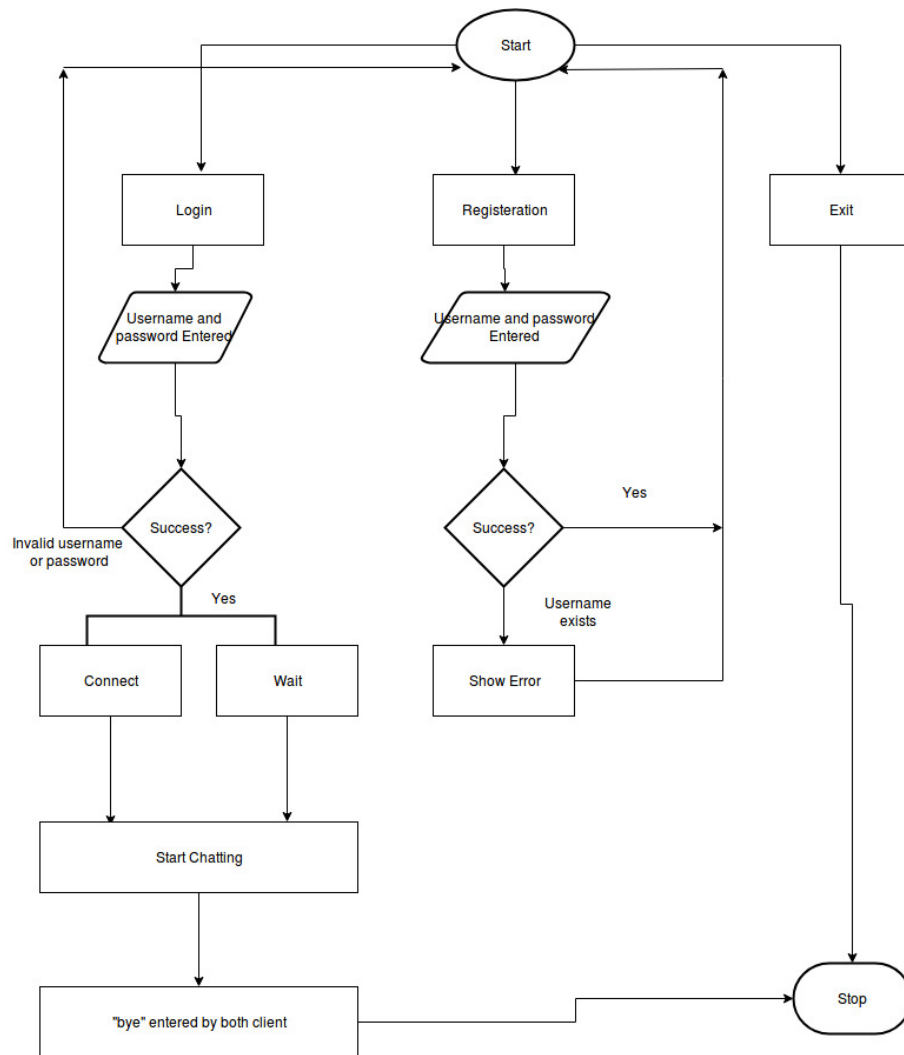
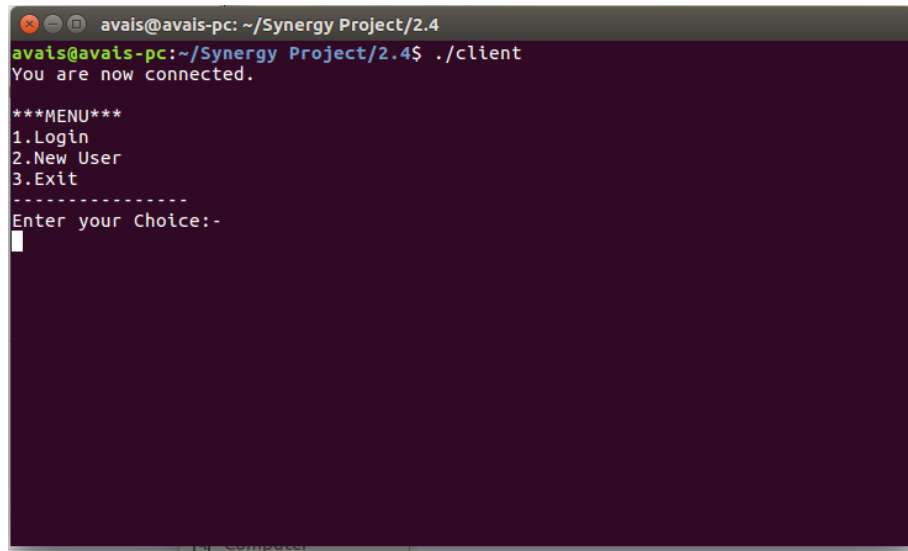


Figure 1: Architecture

### 3 Types Of Request

In our project, we have a different type of requests:-



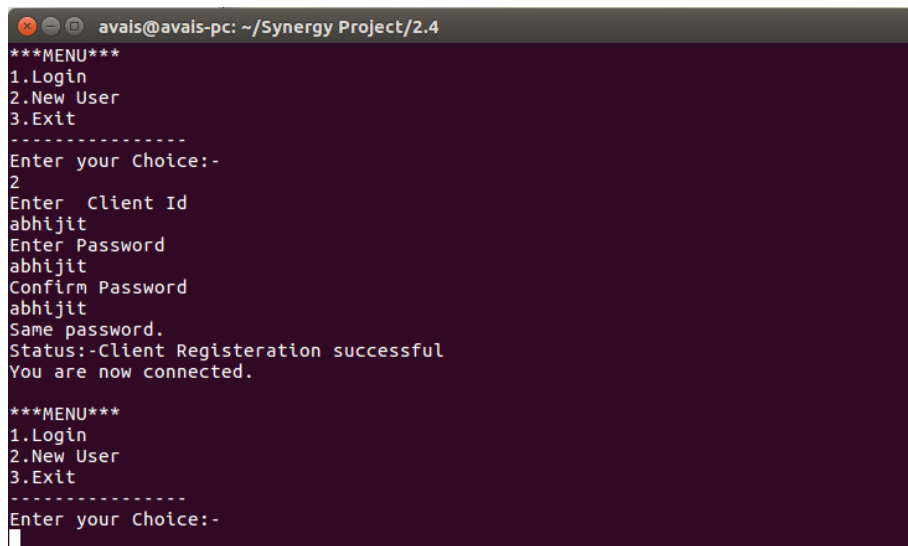
```
avals@avals-pc: ~/Synergy Project/2.4
avals@avals-pc:~/Synergy Project/2.4$ ./client
You are now connected.

***MENU***
1.Login
2.New User
3.Exit
-----
Enter your Choice:-

```

Figure 2: Menu

1. Registration:- In this request, the client establishes the connection with Server and do registration operation with user id and password. On completion of the request, Server gives response message.



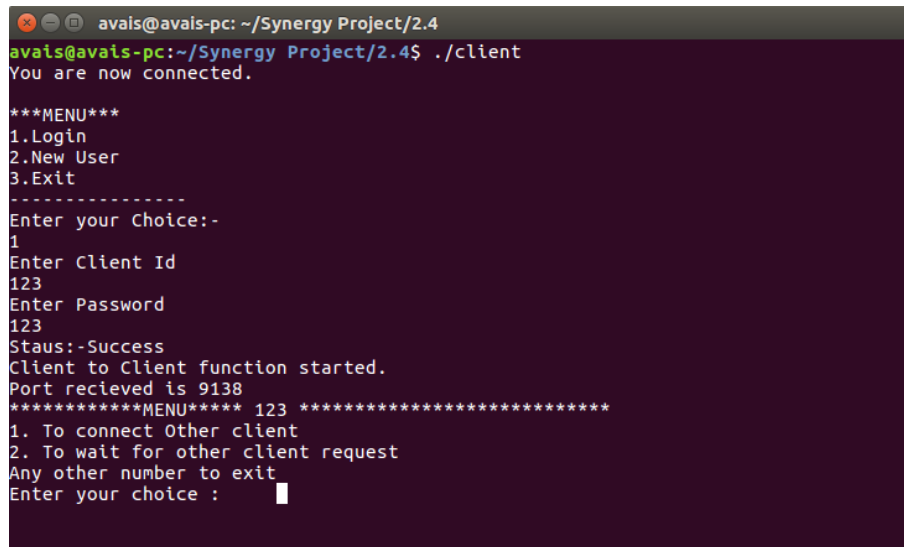
```
avals@avals-pc: ~/Synergy Project/2.4
***MENU***
1.Login
2.New User
3.Exit
-----
Enter your Choice:-
2
Enter Client Id
abhijit
Enter Password
abhijit
Confirm Password
abhijit
Same password.
Status:-Client Registration successful
You are now connected.

***MENU***
1.Login
2.New User
3.Exit
-----
Enter your Choice:-

```

Figure 3: Registration Request

2. In this request, client login with credentials and ask for a self-port number from the server. then client creates a socket on self-port number wait for a client request
3. In this request, client login with credentials and ask for an another client port number whom it wants to connect. then the client sends a request on that port number.

A terminal window titled 'avals@avals-pc: ~/Synergy Project/2.4' showing the execution of a client program. The user runs './client' and receives 'You are now connected.' A menu is displayed with options: 1.Login, 2.New User, 3.Exit. The user enters '1'. Then 'Enter Client Id' is prompted, and '123' is entered. Then 'Enter Password' is prompted, and '123' is entered. The output shows 'Staus:-Success' (note the typo), 'Client to Client function started.', and 'Port recieved is 9138' (note the typo). A second menu is shown: '\*\*\*\*\*MENU\*\*\*\*\* 123 \*\*\*\*\*' with options: '1. To connect Other client', '2. To wait for other client request', and 'Any other number to exit'. The prompt 'Enter your choice :' is shown with a cursor.

```
avals@avals-pc: ~/Synergy Project/2.4
avals@avals-pc:~/Synergy Project/2.4$ ./client
You are now connected.

***MENU***
1.Login
2.New User
3.Exit
-----
Enter your Choice:-
1
Enter Client Id
123
Enter Password
123
Staus:-Success
Client to Client function started.
Port recieved is 9138
*****MENU***** 123 *****
1. To connect Other client
2. To wait for other client request
Any other number to exit
Enter your choice : █
```

Figure 4: Login Request

4. After the establishment of Connection to another client, it breaks the connection with the server and starts message on their private ports.

## 4 Load Generator

We've written a closed-loop load generator to test the system, in which there are N threads continuously sending requests to the servers without any think time in between the subsequent requests.

### 4.1 System View

- The load generator runs for 300 seconds and measures average response time and throughput of the system for these 300 seconds.
- The Front end and Back end servers run on one machine with multiple CPU Cores on it, while the Load Generator runs on another.
- Front End will interact with Back end server which is MYSQL database
- We have implemented Multi-Threaded server and Load Generator.

### 4.2 System Flow

- The main thread of the load generator spawns N threads.
- Each thread runs in loop for 300 seconds. each thread creates client socket. After completion of First request it generates new request.
- For each of these request, we maintain a TCP socket connection to the Front end Server, which in turn opens MYSQL connections to the Back end Server, one for each connection from the Load Generator.
- Requests are sent continuously from the Load Generator to the Front end Server for these N threads without any delay in between the requests, Front end Server assigns these requests to the Back end Server using the corresponding outgoing connection of the current thread.

### 4.3 Load Test

- The main thread calls join and waits for each of the N user threads, each of which runs for 300 seconds.
- Within loop, we take the system time stamp before calling a request and after receiving a response respectively, to get the response time for that request.
- We noted all such response times to get cumulative response time for number of requests as 'R' .
- We calculate the average response time 'i' across 'r' requests

$$i = \frac{totalResponsetime(R)}{NumberOfRequests(r)} \quad (1)$$

- Now, we calculate the throughput for closed loop systems as per formula:

$$X = \frac{NumberOfRequestServed}{TotalTime} \quad (2)$$

## 4.4 Observations

Upon increasing the number of users to the system we calculated average response time per request . Upon observation, it is clear that Average Response Time per request increases on increasing the number of users N in the system as defined by the number of threads spawned on the load generator initially. After a certain point response time increases gradually as when the CPU hits its maximum capacity and could not perform better. Hence it takes more amount of time to service any particular request by the system.

### 4.4.1 Registration Request

As it is shown in Figure3 on Page5 , new user has to provide Unique Client ID and password to register into the database. Our load generator generates new client socket and bombards registration request on Server.

Load generator creates 'N' threads, each thread generates request in a loop. For each request, it creates client socket and establishes the connection with server. then it sends "Registration" message to inform server that client wants to perform registration operation. On receiving appropriate response from server, Client sends Client Id and password. After completion of a request new request will generate.

In Load generator, we started with low number of threads and gradually increase the number of threads till it reaches to stable value of throughput. For N=35 , our system reaches full %CPU Utilization . Our system is bottleneck by Our CPU as you can see in Figure10 where all 4 cores are reached to 100% utilization and for very large number threads MYSQL is also giving low performance as it is unable to maintain more connection at a time. In load generator ,Thread has to wait in loop for MYSQL connection till it gets connected successfully.



**1. Throughput Of the System**  
**Observation are in Table shown below**

No of Threads	Throughput
1	20.79
2	27.53
5	62.65
10	89.72
15	104.70
18	108.09
20	112.11
25	114.54
30	115.47
35	117.22
40	117.62
50	118.49
100	118.69

And the corresponding graph plotted against these values is shown above:

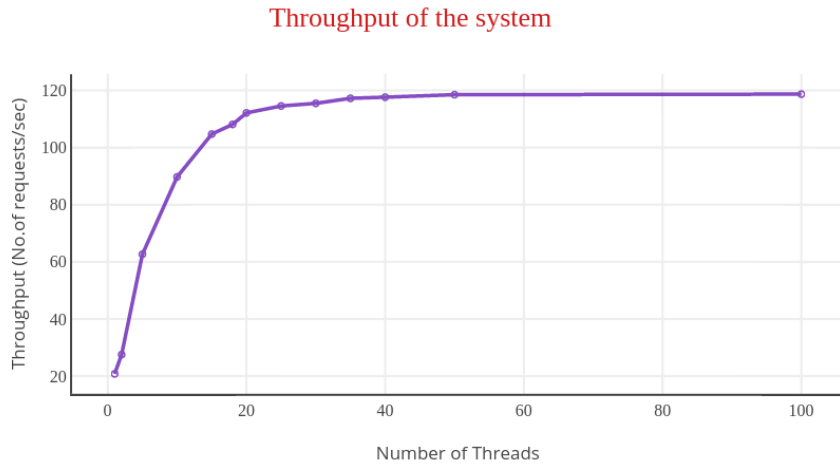


Figure 5: Throughput

## 2.Latency Of the System

No of Threads	Latency
1	48032
2	72585
5	79763
10	111472
15	143315
18	166693
20	178480
30	260227
35	320779
45	380646
50	423090

And the corresponding graph plotted against these values is shown above:

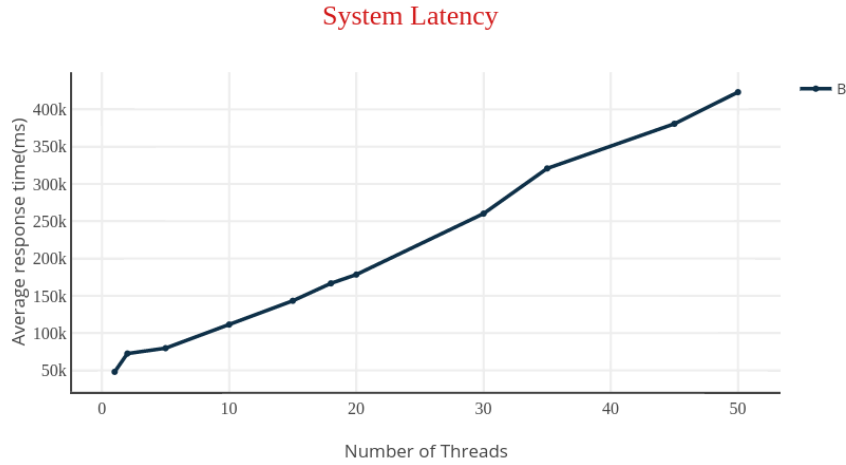


Figure 6: Latency of the System

### 4.4.2 Login Request

As it is shown in Figure4 on Page6, Client has to provide Unique Client ID and password to login into the database. Our load generator generates new client socket and bombards registration request on Server.

Load generator creates 'N' threads, each thread generates request in a loop. For each request, it creates client socket and establishes the connection with server. Then it sends "Login" message to inform server that client wants to perform Login operation. On receiving appropriate response from server, Client sends Client Id and password. Then client has two option either wait on self port or request another client . In our Load generator , we check the load of "Wait on self port" request. In which after login operation, client ask for self port to server which is unique for all registered clients. After getting the self

port from server, it suppose to create new socket on that port number. we After completion of a request new request will generate.

In Load generator, we started with low number of threads and gradually increase the number of threads till it reaches to stable value of throughput. For N=8 , our system reaches full %CPU Utilization . Our system is bottleneck by Our CPU utilization as you can see in Figure12 where all 4 cores are reached to 100% utilization .On further increasing the load, there is full CPU utilization which may be accounted for thrashing occurring in the system and hence CPU may now busy servicing page faults rather than actual user requests. On further increasing the load this dip increases a lot more.For very large number of threads, MYSQL is also giving low performance as it is unable to maintain more connection at a time. In load generator ,Thread has to wait in loop for MYSQL connection till it gets connected successfully.

### 1. Throughput Of the System

No of Threads	Throughput
1	14.40666667
2	25.22
3	28.51
4	29.22333333
5	29.38
8	29.51
20	29.54666667

And the corresponding graph plotted against these values is shown above:

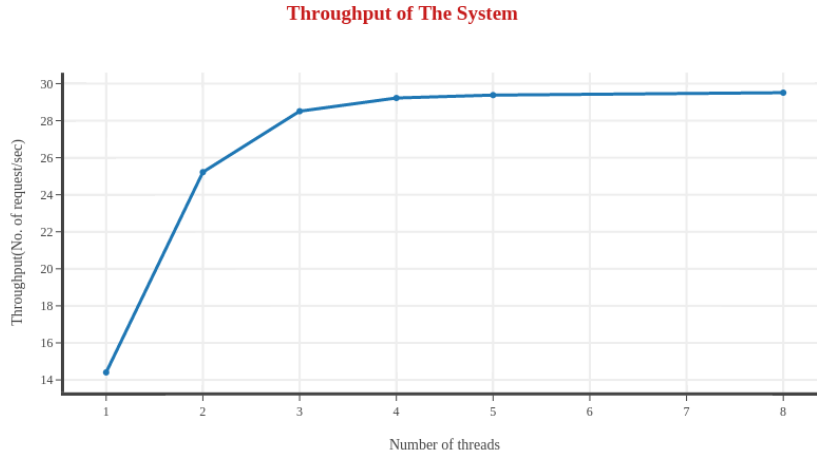


Figure 7: Throughput

## 2.Latency Of the System

No of Threads	Latency
1	69429
2	79324
3	105273
4	136973
5	170334
8	271523
20	678913

And the corresponding graph plotted against these values is shown above:

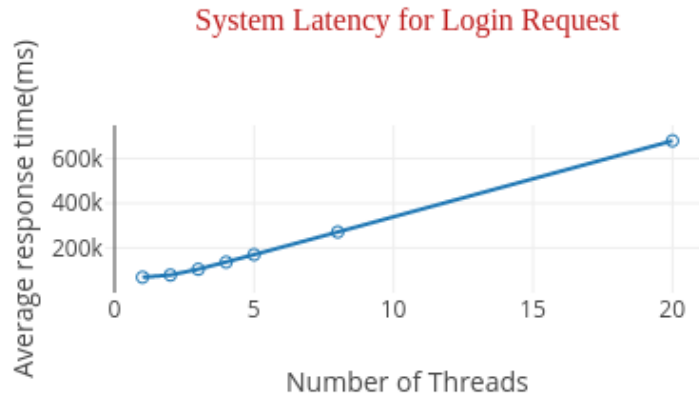


Figure 8: Latency of the system

## 4.5 Screen shots

Following are the screen shots captured at various values of number of users N, refer to them to observe CPU utilization by process Server. back end server process spawns several child processes and their utilization is seen individually. Number of child processes is equal to the number of users in the system. Total CPU Utilization by Back end server process should be calculated by summing up percentage utilization of parent process and all the child process together. Since system has 4 cores so, total of 400% confirms CPU bottleneck.

System Performance Analysis using "htop" command

## 1. Performance at low number of threads

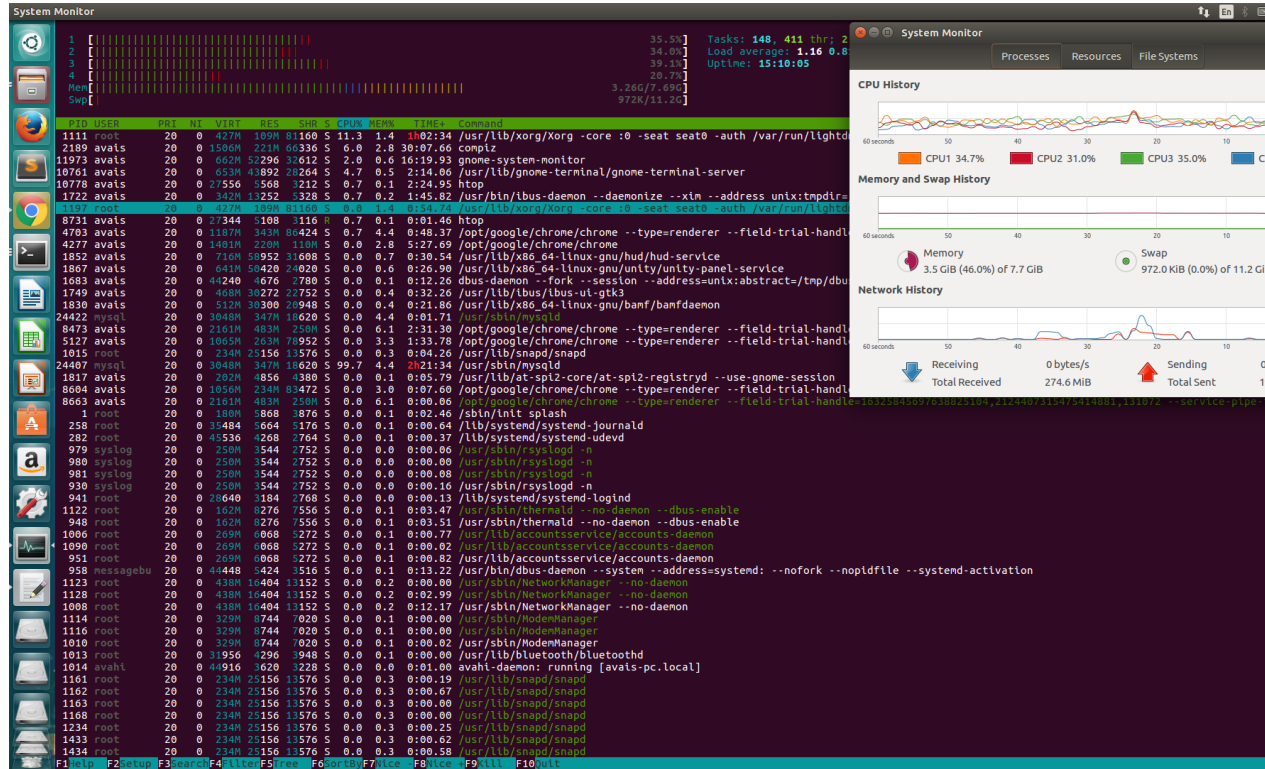
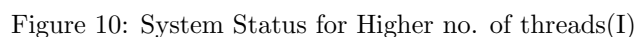


Figure 9: System Status for Lower no. of threads

There is still scope to increase the number of threads as CPU is not fully utilized



## 2. Increase in Number of threads but not fully utilized

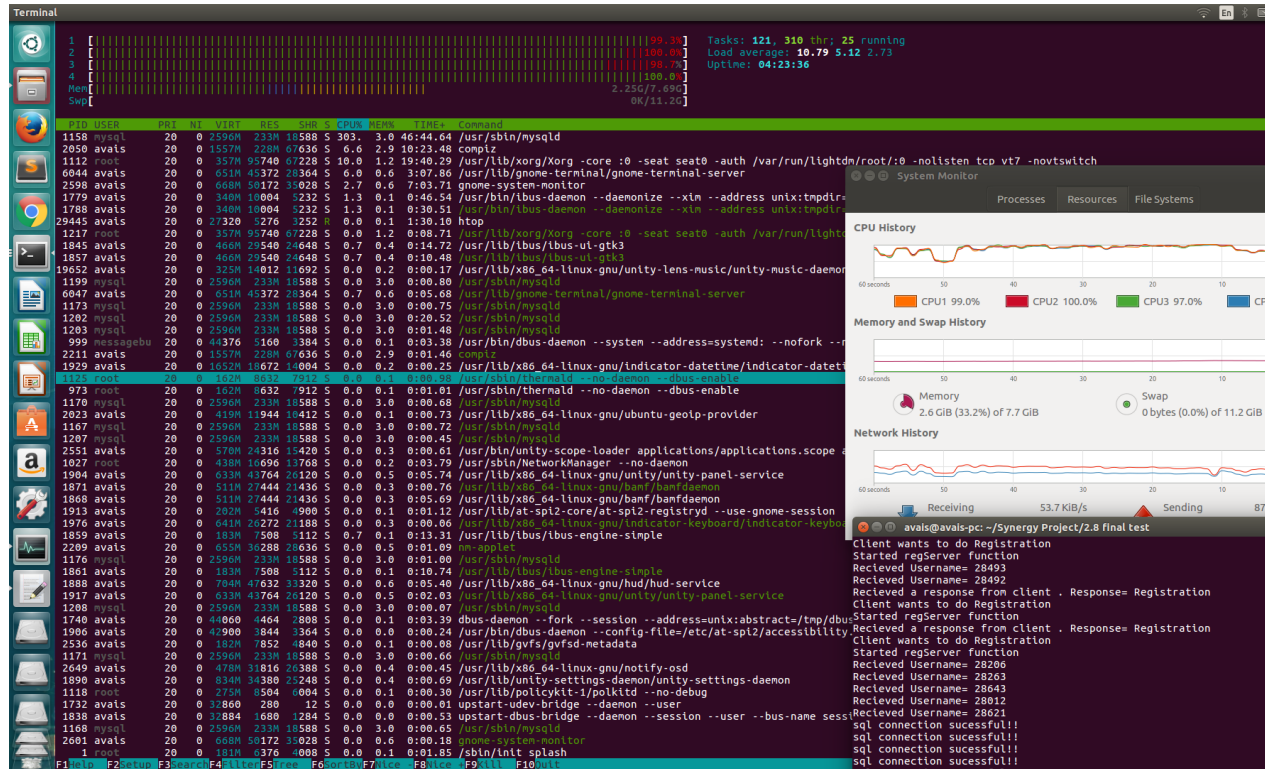


Figure 11: System Status for Higher no. of threads(II)

### 3.Full CPU Utilization which lowers the performance.

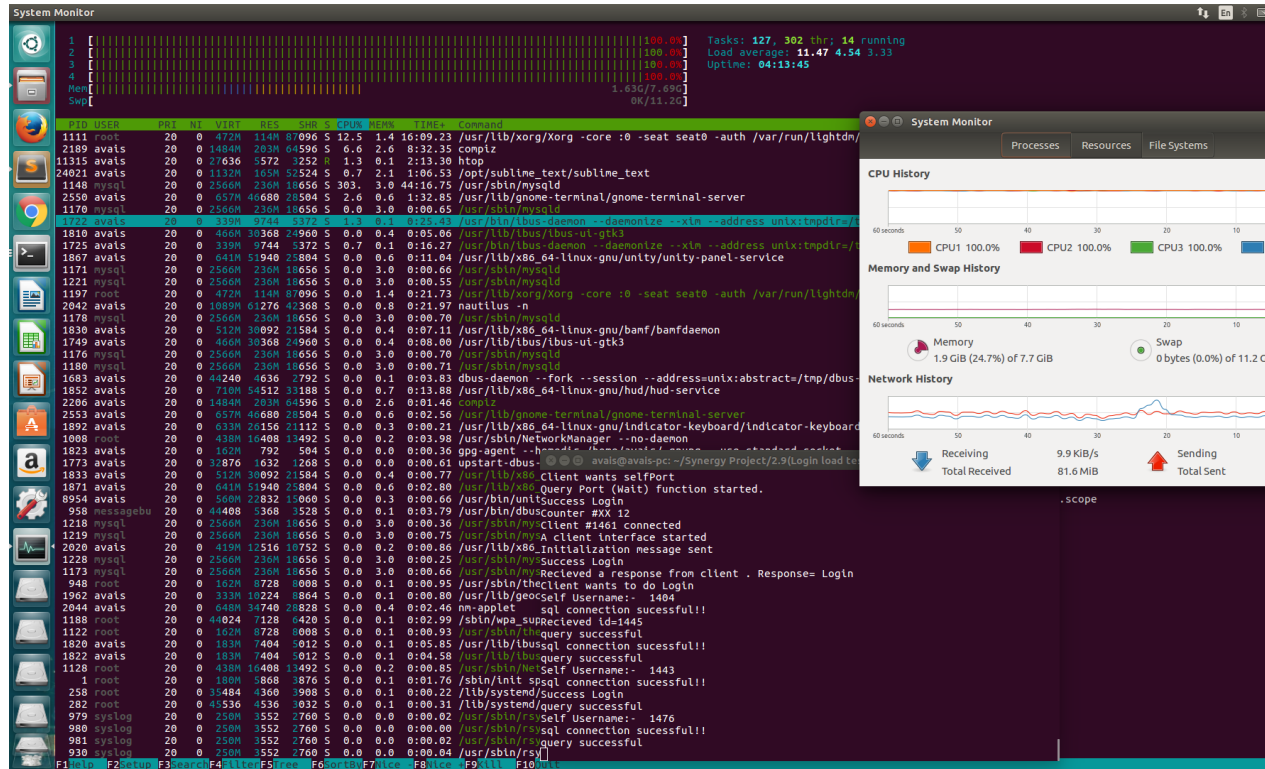


Figure 12: System Status for Full CPU utilization



## 5 Conclusion

From this project phase , we learned how to test our system capacity. Through-put of the system gradually increases as increase in number of threads but after some threshold value it fully utilized the system. Average latency of system keeps on increasing because of waiting time for threads increases. we come to know about bottleneck of system which we can improve in next phase.