File server with backend authentication

Phase 3 : Optimisation

Major Amit Pathania 163054001
Nivia Jatain 15305R007

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

November 2017

Contents

List of Figures

List of Tables

1 Introduction

File server with backend authentication server has been implemented using TCP sockets in C++. There are two servers: Server1 and Server2. Server1 (File server) receive TCP requests from client and requests a username and password from client for either creating new account or for requesting a file. Server1 will send this username and password to Server2 (Backend authentication server) to authenticate the user or add new user account. Once authenticated, the client can then request files from the server1 and if file is present, server1 will send the file to client. **Server1 is multithreaded** to handle multiple client requests. **Server2 is multiprocess** to handle multiple authentication requests from Server1. The broad architecture of our system is described in figure1.

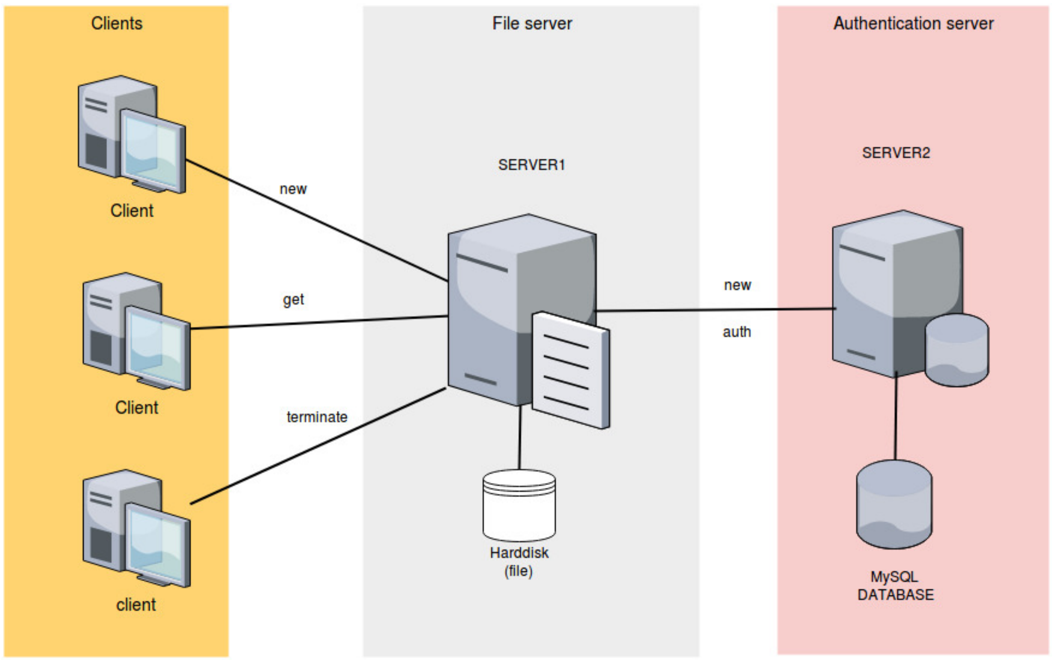


Figure 1: System Architecture

Our load generator performs close loop test using N threads.The response time, T , in a closed system is defined to be the time from when a request is submitted until it is received. In the case where the system is a single server (e.g. a web server), the server load, denoted by ρ , is defined as the fraction of time that the server is busy, and is the product of the mean throughput X and the mean service demand (processing requirement) E[S].

2 Background

2.1 Design of load generator

We are performing a closed loop test where new job arrivals are only triggered by job completions.In order to simulate N concurrent users, we create N threads in our load generator. Each of the N threads emulate one user by issuing one request, waiting for it to complete, and issues the next request immediately afterwards.There is no think time between requests.

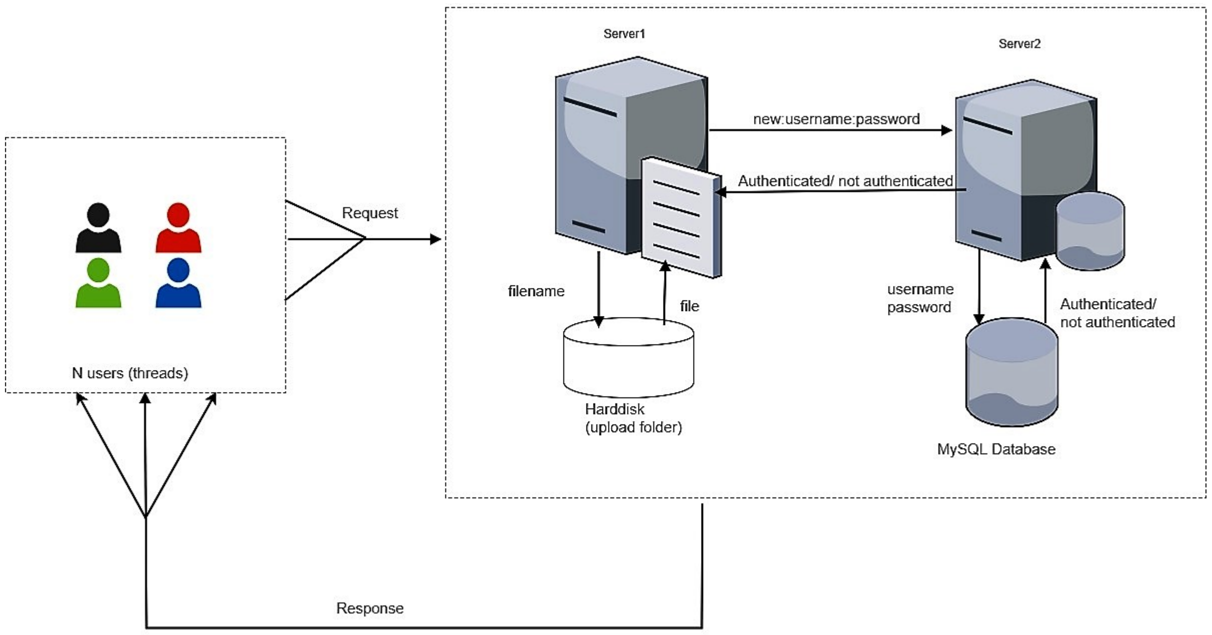


Figure 2: Design of load generator

2.2 Type of load/requests

Load generator can generate two types of load (or requests):

- **To create new user account**. In this case, "new" keyword followed by username and password will be sent to server1 (**new: uname: upassword**). Server1 will send "new" keyword followed with username and password to Server2 . Server2 will try to create user account. In this case,**there is no file transfer (disk access at Server1) and only new socket connection** is created between Client, Server 1 and server2 for each request. **Server2 will connect to mySQL database for each request** and try to create account. For creating new account, Server2 will write into the database. Hence, **disk access at Server2**.

  For each N, we defined total runtime ie 100-120 seconds.Each thread sent request to create account till runtime is over. We counted the number of requests completed and based on this time, we calculated throughput as Total Number of requests/ Actual elapsed time. For each thread,we generated the new user's username and password which was sent to server.

- **To request file from server1**. In this case, "get" keyword followed by username and password will be sent to server1 (**get: uname: upassword**). Server1 will send "auth" keyword followed with username and password to Server2 . Server2 will check the database for user account.If user details are correct, server1 will be informed and server1 will request filename from user. Client will then send filename with extension to server1(File server) and then file will be sent to client if that file exists at server1. In this case,**there is file transfer (ie disk access at Server1) and new socket connection** is created between Client, Server 1 and server2 for each request. **Server2 will connect to mySQL database** for each request and authenticate the user.

In this load test, we each user thread fetched different file of given file size everytime.We generated large number of test files of given size(1KB, 10KB,100KB, 1MB, 10MB, 100MB). In order to ensure that different files from server are fetched by different user threads with less probability of file being in cache, for each user we defined number of total fetch file requests.For instance, 2000 file request for each user. So, N=1,there were total 2000 requests,for N=2,total 4000 requests, for N=10, total 20000 requests and so on. The total time taken to complete these requests was calculated.And based on this time, we calculated throughput as Total Number of requests/ Total time taken.This approach helped us to ensure that total bytes of data downloaded by each user remained same for testing. In order to ensure that no user thread fetches the already fetched file and read the file from disk, we generated file name for each request as follows:

```
int r = (long(threadid)* 2000) +num_req+1;
strcpy(file_name,"500KB");
sprintf(count,  r);
strcat(file_name,count);
strcat(file_name,".txt");
num_req++;
```

This ensured that there is offset of 2000 between files demanded by each thread as per thread id and the same file is not requested again by some other thread. But, we had to generate 2000*N files for such test case. Also, after every one test for some value of N for given file, we cleared the buffer cache using following commands:-

```
sync; sudo sh -c "echo 3 > /proc/sys/vm/drop_caches"
```

Generating test files. We have to create files of varying sizes to check system load for different file sizes.Since, we don't care about contents of file and only file size matters, we used /dev/urandom option to create file. The shell code for same is as follows:

```
#! /bin/bash
for n in {1..5000}; do
 base64 /dev/urandom$|$head -c 200000000 $>$  200MB\$n.txt
done
```

2.3 Testing environment

Earlier in phase2, we had setup Server1 (File server) and Server2 (Authentication server) on Virtual Machine. Client is running on host machine. Client connects to server machine over bridged network between host and VM. Server contains upload folder which contains files which client can download.

However, for Phase3, we setup Servers on different system from client PC. The experiments done for load testing in Phase2 were again conducted on new system to calculate Throughput and Response Time values.

Client and Server Hardware Configuration. The hardware specifications of server and client machine for Phase 3 is as follows:

	Load Generator	**Server1 & 2**
Processor	4x Intel(R) Core(TM) i5-4440 CPU @ 3.10GHz	Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz
CPU(s)	4	8
Thread(s) per core	1	2
Core(s) per socket	4	4
Socket(s)	1	1
L1d cache	32K	32K
L1i cache	32K	32K
L2 cache	256K	256K
L3 cache	6144K	8192K

Table 1: Client and Server Hardware Configuration

Tuning MySQL. By default, maximum number of simulatneous conenctions in MySQL is 150. Can be viewed using command:

show variables like 'max_connections';

The limit can be increased using command:

SET GLOBAL max_connections = 15000;

Tuning sockets: In order to make our server handle many concurrent connections, there is requirement to tune socket parameters. In listen system call in server, we define number of maximum incoming connections that can be handled. By default, in ubuntu this limit is 128 which is very less if we want to saturate the server.

To view maximum connections handled by socket:
cat /proc/sys/net/core/somaxconn
128(default)

Increasing ulimit parameters. By default the number of open files for user in Ubuntu is 1024. We have to increase it to measure maximum throughput.This is done with the ulimit command:

ulimit -a // see all the kernel parameters
ulimit -n // see the number of open files
ulimit -n 9000 // set the number open files to 9000

To make these changes permanent, add the following line to /etc/security/limits.conf:
useracct hard nofile 64000

3 Identification of bottleneck

The bottleneck of our system is **disk**. We monitored disk IO activity using nmon and iostat tools. For **New** request, there was **"Disk Write"** for each new user account creation in MySQL. At saturation, there was MySQL database connection error. For **Get** request, there was **"Disk Read"** at server for uploading file to the client.

The summary of results :

Load type	File size	N*	Mean Response Time at N=N*	Max Through-put at N=N*	Bottleneck
new		800	0.219622	3363.49	Disk I/O (Write) and MySQL
get	1KB	17	0.0054039	3093.7	Disk I/O (Read)
get	10KB	10	0.00390524	2547.3	Disk I/O (Read)
get	100KB	5	0.0113418	430.355	Disk I/O (Read)
get	1MB	2	0.0265555	75.1187	Disk I/O (Read)
get	10MB	1	0.0984429	10.1575	Disk I/O (Read)
get	100MB	1	0.880579	1.1356	Disk I/O (Read)

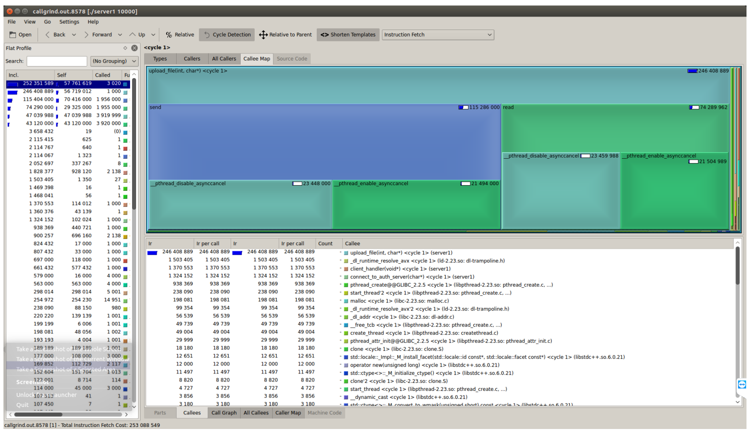
Table 2: Test results before optimisation

3.1 Profiling the code

In order to find which portion of code/function, our server1 is spending most of the time, we used valgrind tool.

valgrind –tool=callgrind ./server1 10000

It generated a file called callgrind.out.xxxx. We used kcachegrind tool to read this file and get graphical analysis.


callgrind.out.8578 [./server1 10000]
File View Go Settings Help
Open Back Forward Up
Relative Cycle Detection Relative to Parent Shorten Templates Instruction Fetch
Flat Profile
Search: (No Grouping)
<cycle 1>
Types Callers All Callers Callee Map Source Code
upload_file(int, char*) <cycle 1> 246 408 889
send 115 296 000
read 74 289 962
__pthread_disable_asynccancel 23 448 000
__pthread_enable_asynccancel 21 494 000
__pthread_disable_asynccancel 23 459 988
__pthread_enable_asynccancel 21 504 989
Parts Callees Call Graph All Callees Caller Map Machine Code

```
Incl.         Self        Called
252 351 589   57 761 619  3 020
246 408 889   56 719 012  1 000
115 404 000   70 416 000  1 956 000
 74 290 000   29 325 000  1 955 000
 47 039 988   47 039 988  3 919 999
 43 120 000   43 120 000  3 920 000
  3 658 432           19  (0)
  2 115 415          625  1
  2 114 767          640  1
  2 114 067        1 323  1
  2 052 697      337 267  8
  1 828 377      928 120  2 138
  1 503 405        1 350  27
  1 469 398           16  1
  1 468 041           56  1
  1 370 553      114 012  1 000
  1 360 376       43 139  1
  1 324 152      102 024  1 000
    938 369      440 721  1 000
    900 257      696 160  2 138
    824 432       17 000  1 000
    807 432       33 000  1 000
    697 000      118 000  1 000
    661 432      577 432  1 000
    579 000       16 000  4 000
    563 000      563 000  4 000
    298 014      298 014  5 001
    254 972      254 230  14 951
    238 090       88 150  980
    220 220      139 139  1 001
    199 199        6 006  1 001
    198 081       48 056  1 002
    193 193        4 004  1 001
    189 189      189 189  1 001
    177 000      108 000  3 000
    169 852      112 729  2 117
    152 604      151 704  1 013
    122 001        8 714  114
    114 000       45 000  3 000
    107 513           41  1
    107 450            7  1
```

```
Ir           Ir per call  Ir           Ir per call  Count  Callee
246 408 889  246 408 889  246 408 889  246 408 889         upload_file(int, char*) <cycle 1> (server1)
  1 503 405    1 503 405    1 503 405    1 503 405         _dl_runtime_resolve_avx <cycle 1> (ld-2.23.so: dl-trampoline.h)
  1 370 553    1 370 553    1 370 553    1 370 553         client_handler(void*) <cycle 1> (server1)
  1 324 152    1 324 152    1 324 152    1 324 152         connect_to_auth_server(char*) <cycle 1> (server1)
    938 369      938 369      938 369      938 369         pthread_create@@GLIBC_2.2.5 <cycle 1> (libpthread-2.23.so: pthread_create.c, ...)
    238 090      238 090      238 090      238 090         start_thread'2 <cycle 1> (libpthread-2.23.so: pthread_create.c, ...)
    198 081      198 081      198 081      198 081         malloc <cycle 1> (libc-2.23.so: malloc.c)
     99 354       99 354       99 354       99 354         _dl_runtime_resolve_avx'2 <cycle 1> (ld-2.23.so: dl-trampoline.h)
     56 539       56 539       56 539       56 539         _dl_addr <cycle 1> (libc-2.23.so: dl-addr.c)
     49 739       49 739       49 739       49 739         __free_tcb <cycle 1> (libpthread-2.23.so: pthread_create.c, ...)
     49 004       49 004       49 004       49 004         create_thread <cycle 1> (libpthread-2.23.so: createthread.c)
     29 999       29 999       29 999       29 999         pthread_attr_init@@GLIBC_2.2.5 <cycle 1> (libpthread-2.23.so: pthread_attr_init.c)
     18 180       18 180       18 180       18 180         clone <cycle 1> (libc-2.23.so: clone.S)
     12 651       12 651       12 651       12 651         std::locale::_Impl::_M_install_facet(std::locale::id const*, std::locale::facet const*) <cycle 1> (libstdc++.so.6.0.21)
     12 000       12 000       12 000       12 000         operator new(unsigned long) <cycle 1> (libstdc++.so.6.0.21)
     11 497       11 497       11 497       11 497         std::ctype<>::_M_initialize_ctype() <cycle 1> (libstdc++.so.6.0.21)
      8 820        8 820        8 820        8 820         clone'2 <cycle 1> (libc-2.23.so: clone.S)
      4 727        4 727        4 727        4 727         start_thread <cycle 1> (libpthread-2.23.so: pthread_create.c, ...)
      3 856        3 856        3 856        3 856         __dynamic_cast <cycle 1> (libstdc++.so.6.0.21)
      3 180        3 180        3 180        3 180         std::ctype<>::_M_convert_to_wmask(unsigned short) const <cycle 1> (libstdc++.so.6.0.21)
```

callgrind.out.8578 [1] - Total Instruction Fetch Cost: 253 088 549

Figure 3: Code profiling

We can see in the profile output that our file server spends maximum time in "**upload_file**" function, ie reading the file and sending the file to client.

4 Optimization

4.1 Step1: Design changes

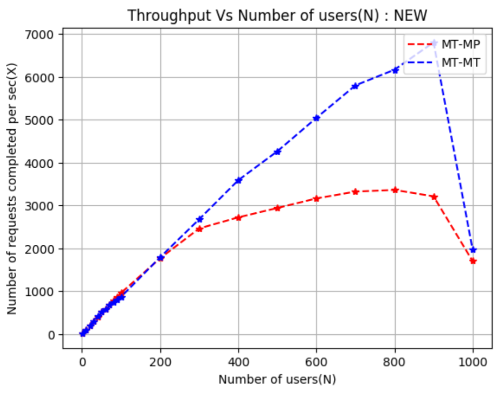
In order to compare performance benefits between Multi-process and multi-threaded architecture, we changed Server2 from Multi-process to multi-threaded.

4.1.1 Performance after design change

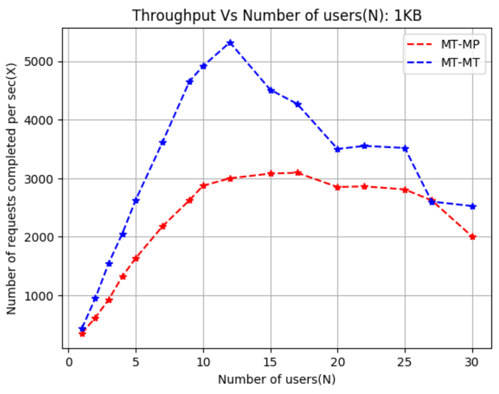
Load type	File size	Before change (MT-MP)		After Change (MT-MT)	
		N*	Max Throughput	N*	Max Throughput
new		800	3363.49	900	6806.03
get	1KB	17	3093.7	12	5323.32
get	10KB	10	2547.3	9	3325.93
get	100KB	5	430.355	5	465.524
get	1MB	2	75.1187	1	81.0029
get	10MB	1	10.1575	2	10.3492
get	100MB	1	1.1356	1	1.14016

Table 3: Test results after design change

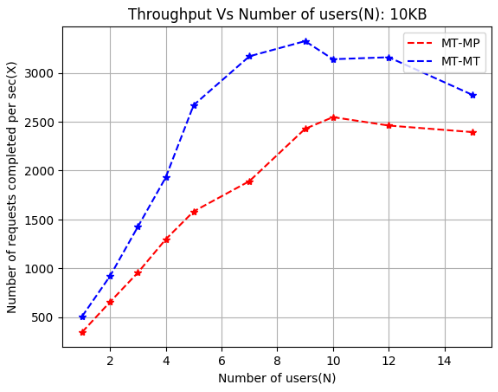
Hence, we conclude that forking a new process per connection has more overhead than creating thread per connection. As each client thread executes independently and don't have shared data/ variables, there is no overhead of synchronization using mutex/locks.



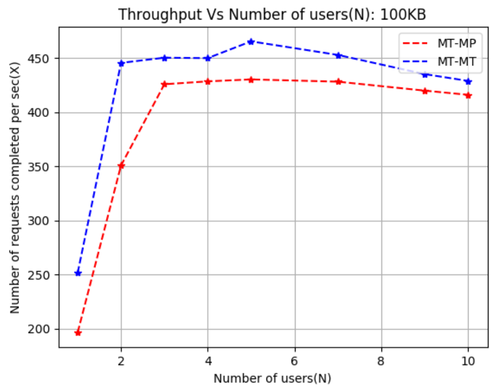
(a)



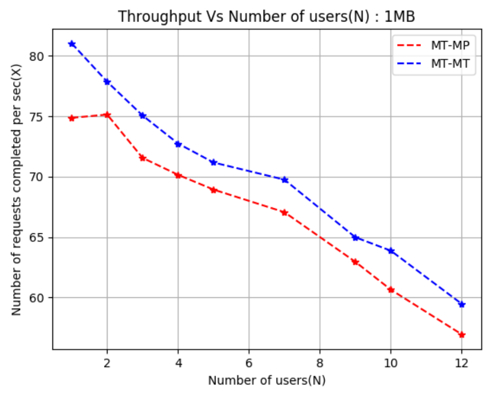
(b)



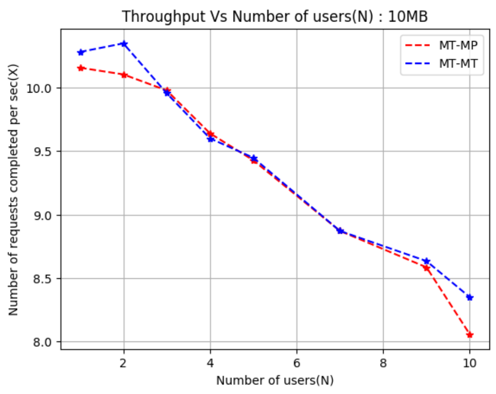
(c)



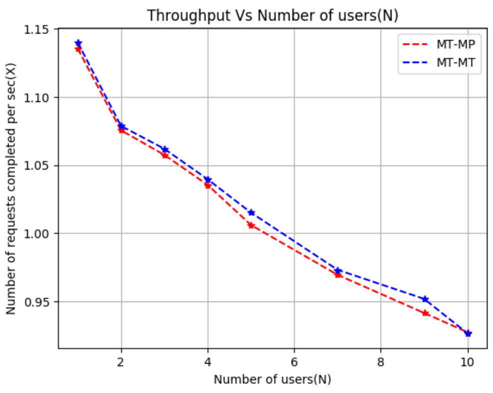
(d)



(e)



(f)



(g)

Figure 4: Comparison in performance between Multi threaded server and Multi process server for different requests

4.2 Step2: Disk IO optimization

When file reading system calls are used, data is usually transferred first from disk into the disk cache and then in the process memory. By prefetching the files into user space and saving it into user space using malloc, we can bypass disk access and disk buffer cache. Direct I/O is a feature of the file system whereby file reads and writes go directly from the applications to the storage device, bypassing the operating system read and write caches. We opened files in O_DIRECT mode and read every file in one I/O, i.e. create buffer in memory of the file size and read into it. We used malloc and created variables to store file contents when server started and all further client request were fetched from these without disk access. This reduced disk IO very much. This increased our throughput for 100KB file to 1154-1160 from 460-465.

However, as we increased N throughput got flattened but CPU was only 25-30% utilised. We used iperf tool to measure Bandwidth between our client and server machines and compared it with results of our load generator :

Test	Interval	Data Transferred	Bandwidth (Mbps)	Bandwidth (MBps)
iperf	10.0 sec	1.09 GBytes	935	116.875
load generator	30.0997 sec	3.4742 GBytes	923.384	115.423

Hence, we concluded **network as our bottleneck**. In order to **shift bottleneck from network to CPU**, we moved servers and load generator to same machine and pinned each server process, MySQL process and load generator to different core of the machine.

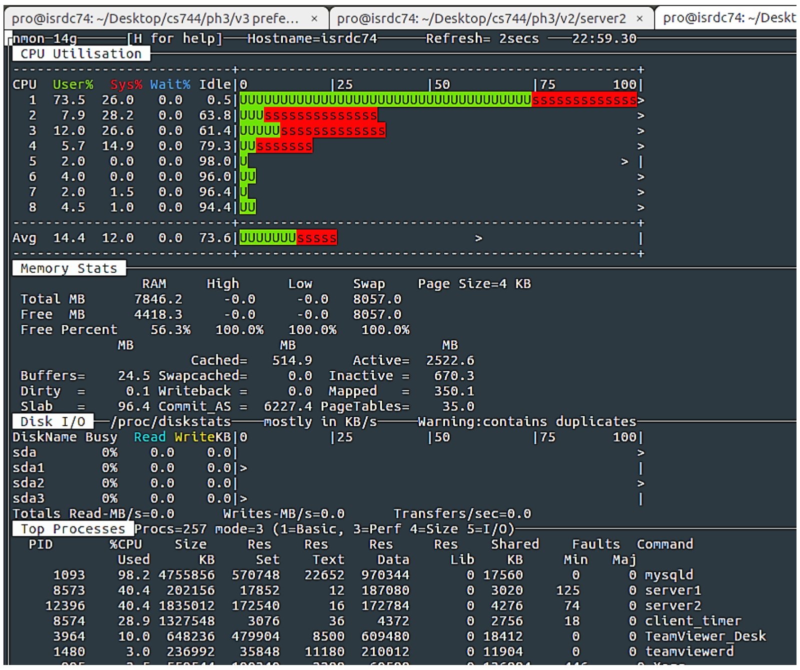
4.3 Step3: Code optimization

Code optimization to improve performance and reduce cache misses:

- Keeping variables that will be accessed together (or right after another) next to each other in memory by declaring them together. Also, created structure to include items accessed together.
- Removing loops to access contents of malloc(ed) files for uploading file. Used memcopy with offset to read and copy contents of file to send buffers.

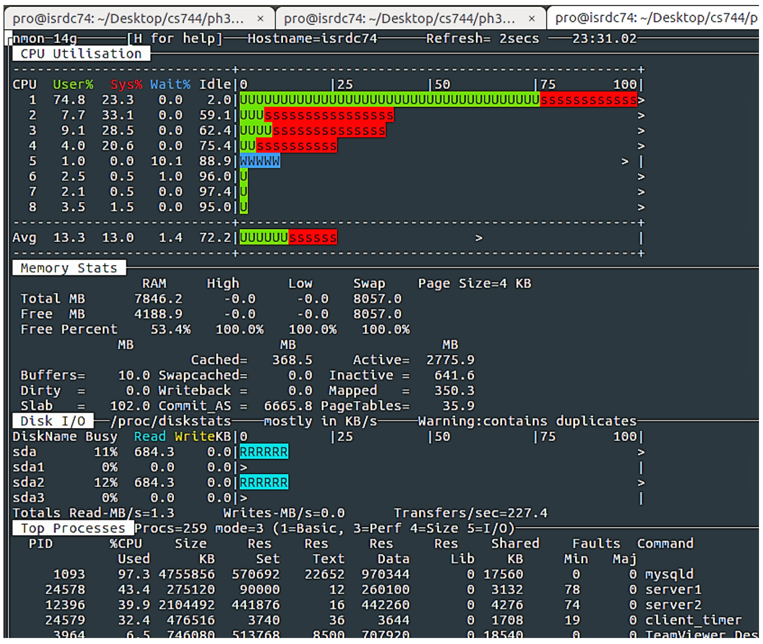
4.3.1 Performance after disk I/O and code Optimization

New bottleneck: For files with size less than 1MB, MySQL was bottleneck and for files with larger file sizes, server1 process was bottleneck.



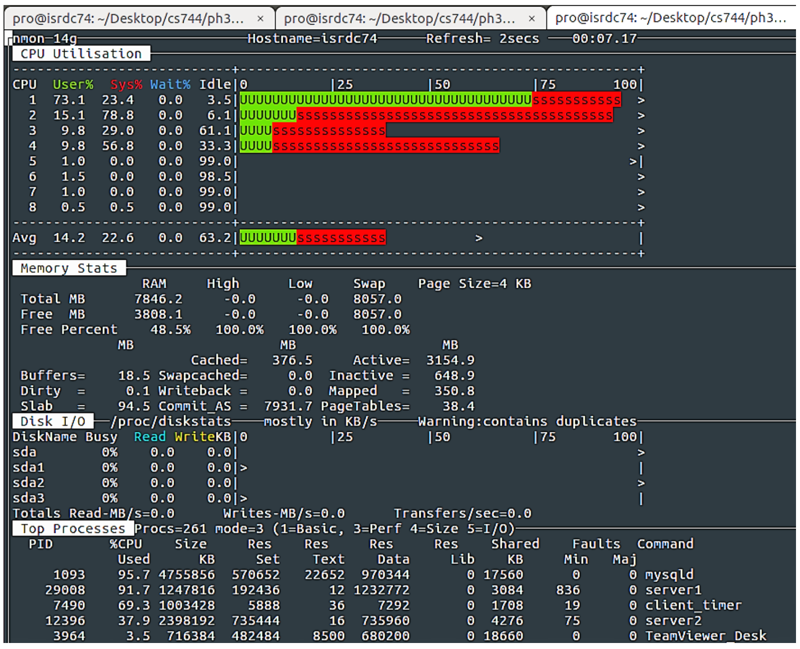
```
pro@isrdc74: ~/Desktop/cs744/ph3/v3 prefe... ×   pro@isrdc74: ~/Desktop/cs744/ph3/v2/server2 ×   pro@isrdc74: ~/Deskt
nmon—14g————[H for help]——Hostname=isrdc74————Refresh= 2secs ——22:59.30——
 CPU Utilisation
---------------------------+-------------------------------------------------+
CPU  User%  Sys% Wait% Idle|0          |25         |50          |75       100|
  1  73.5  26.0   0.0   0.5|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUsssssssssssss>
  2   7.9  28.2   0.0  63.8|UUUssssssssssssss                                >
  3  12.0  26.6   0.0  61.4|UUUUUsssssssssssss                               >
  4   5.7  14.9   0.0  79.3|UUsssssss                                        >
  5   2.0   0.0   0.0  98.0|U                                              > |
  6   4.0   0.0   0.0  96.0|UU                                               >
  7   2.0   1.5   0.0  96.4|U                                                >
  8   4.5   1.0   0.0  94.4|UU                                               >
---------------------------+-------------------------------------------------+
Avg  14.4  12.0   0.0  73.6|UUUUUUUsssss                >                    |
---------------------------+-------------------------------------------------+
 Memory Stats
                RAM     High      Low     Swap    Page Size=4 KB
 Total MB      7846.2     -0.0     -0.0   8057.0
 Free  MB      4418.3     -0.0     -0.0   8057.0
 Free Percent    56.3%   100.0%   100.0%   100.0%
             MB                   MB                   MB
                     Cached=   514.9     Active=  2522.6
 Buffers=    24.5 Swapcached=     0.0  Inactive =   670.3
 Dirty  =     0.1 Writeback =     0.0  Mapped   =   350.1
 Slab   =    96.4 Commit_AS =  6227.4 PageTables=    35.0
 Disk I/O ——/proc/diskstats————mostly in KB/s————Warning:contains duplicates——
DiskName Busy  Read WriteKB|0          |25         |50          |75       100|
sda         0%    0.0    0.0|                                                >
sda1        0%    0.0    0.0|>                                               |
sda2        0%    0.0    0.0|                                                >
sda3        0%    0.0    0.0|>                                               |
Totals Read-MB/s=0.0      Writes-MB/s=0.0      Transfers/sec=0.0
 Top Processes Procs=257 mode=3 (1=Basic, 3=Perf 4=Size 5=I/O)——————
  PID       %CPU    Size     Res     Res     Res     Res   Shared    Faults  Command
             Used      KB     Set    Text    Data     Lib     KB     Min  Maj
     1093    98.2 4755856  570748   22652  970344       0 17560      0      0 mysqld
     8573    40.4  202156   17852      12  187080       0  3020    125      0 server1
    12396    40.4 1835012  172540      16  172784       0  4276     74      0 server2
     8574    28.9 1327548    3076      36    4372       0  2756     18      0 client_timer
     3964    10.0  648236  479904    8500  609480       0 18412      0      0 TeamViewer_Desk
     1480     3.0  236992   35848   11180  210012       0 11904      0      0 teamviewerd
```

Figure 5: New Bottleneck for file size 1KB



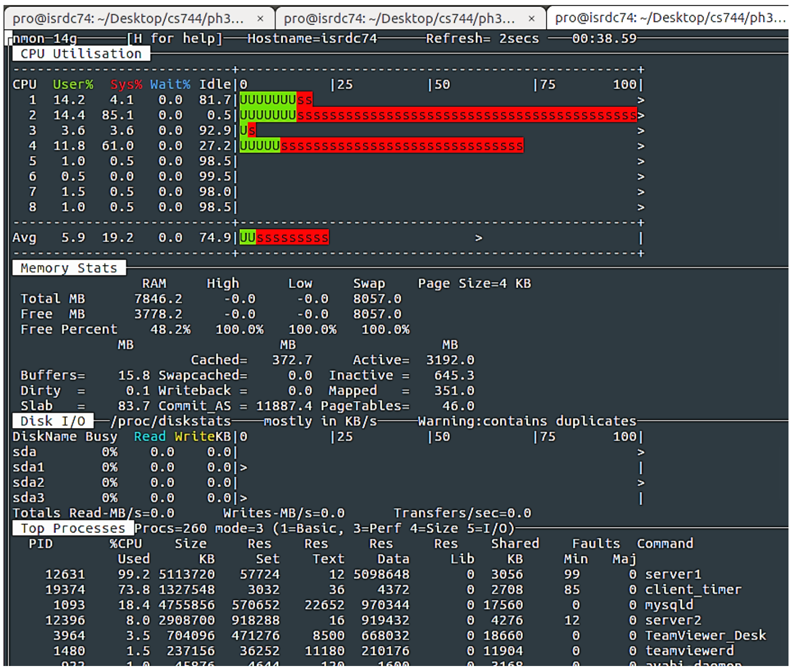
```
pro@isrdc74: ~/Desktop/cs744/ph3...  ×  pro@isrdc74: ~/Desktop/cs744/ph3...  ×  pro@isrdc74: ~/Desktop/cs744/p
nmon—14g———————[H for help]——Hostname=isrdc74————Refresh= 2secs ——23:31.02——
 CPU Utilisation
---------------------------+-------------------------------------------------+
CPU  User%  Sys% Wait% Idle|0          |25         |50          |75       100|
  1  74.8  23.3   0.0   2.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUsssssssssssss>
  2   7.7  33.1   0.0  59.1|UUUsssssssssssssssss                              >
  3   9.1  28.5   0.0  62.4|UUUUsssssssssssssss                               >
  4   4.0  20.6   0.0  75.4|UUssssssssss                                      >
  5   1.0   0.0  10.1  88.9|WWWWW                                           > |
  6   2.5   0.5   1.0  96.0|U                                                 >
  7   2.1   0.5   0.0  97.4|U                                                 >
  8   3.5   1.5   0.0  95.0|U                                                 >
---------------------------+-------------------------------------------------+
Avg  13.3  13.0   1.4  72.2|UUUUUUssssss                  >                  |
---------------------------+-------------------------------------------------+
 Memory Stats
               RAM      High      Low      Swap    Page Size=4 KB
 Total MB      7846.2     -0.0     -0.0    8057.0
 Free  MB      4188.9     -0.0     -0.0    8057.0
 Free Percent    53.4%   100.0%   100.0%   100.0%
             MB                  MB                  MB
                   Cached=   368.5     Active=  2775.9
 Buffers=    10.0 Swapcached=    0.0  Inactive =   641.6
 Dirty  =     0.0 Writeback =    0.0  Mapped   =   350.3
 Slab   =   102.0 Commit_AS =  6665.8 PageTables=    35.9
 Disk I/O ——/proc/diskstats———mostly in KB/s————Warning:contains duplicates——
DiskName Busy  Read WriteKB|0          |25         |50          |75       100|
sda       11%  684.3    0.0|RRRRRR                                            >
sda1       0%    0.0    0.0|>                                                 |
sda2      12%  684.3    0.0|RRRRRR                                            >
sda3       0%    0.0    0.0|>                                                 |
Totals Read-MB/s=1.3      Writes-MB/s=0.0      Transfers/sec=227.4
 Top Processes Procs=259 mode=3 (1=Basic, 3=Perf 4=Size 5=I/O)————————————————
  PID      %CPU    Size     Res     Res     Res     Res   Shared   Faults  Command
            Used      KB     Set    Text    Data    Lib     KB    Min   Maj
     1093   97.3 4755856  570692   22652  970344      0 17560     0      0 mysqld
    24578   43.4  275120   90000      12  260100      0  3132    78      0 server1
    12396   39.9 2104492  441876      16  442260      0  4276    74      0 server2
    24579   32.4  476516    3740      36    3644      0  1708    19      0 client_timer
     3964    6.5  746080  513768    8500  707920      0 18540     0      0 TeamViewer_Des
```

Figure 6: New Bottleneck for file size 10KB



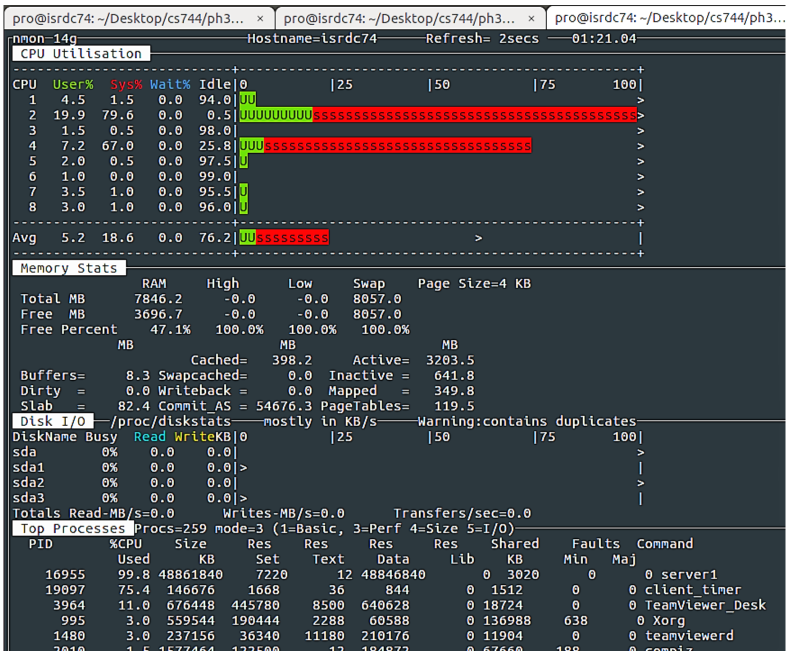
```
pro@isrdc74: ~/Desktop/cs744/ph3...  ×  pro@isrdc74: ~/Desktop/cs744/ph3...  ×  pro@isrdc74: ~/Desktop/cs744/ph3...
nmon—14g——————————————————————Hostname=isrdc74————Refresh= 2secs ——00:07.17——
 CPU Utilisation
---------------------------+-------------------------------------------------+
CPU  User%  Sys% Wait% Idle|0          |25         |50          |75       100|
  1  73.1  23.4   0.0   3.5|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUsssssssssss  >
  2  15.1  78.8   0.0   6.1|UUUUUUUsssssssssssssssssssssssssssssssssssssss   >
  3   9.8  29.0   0.0  61.1|UUUUssssssssssssss                                >
  4   9.8  56.8   0.0  33.3|UUUUssssssssssssssssssssssssssss                  >
  5   1.0   0.0   0.0  99.0|                                                 >|
  6   1.5   0.0   0.0  98.5|                                                  >
  7   1.0   0.0   0.0  99.0|                                                  >
  8   0.5   0.5   0.0  99.0|                                                  >
---------------------------+-------------------------------------------------+
Avg  14.2  22.6   0.0  63.2|UUUUUUUsssssssssss            >                  |
---------------------------+-------------------------------------------------+
 Memory Stats
               RAM      High      Low      Swap    Page Size=4 KB
 Total MB      7846.2     -0.0     -0.0    8057.0
 Free  MB      3808.1     -0.0     -0.0    8057.0
 Free Percent    48.5%   100.0%   100.0%   100.0%
             MB                  MB                  MB
                   Cached=   376.5     Active=  3154.9
 Buffers=    18.5 Swapcached=    0.0  Inactive =   648.9
 Dirty  =     0.1 Writeback =    0.0  Mapped   =   350.8
 Slab   =    94.5 Commit_AS =  7931.7 PageTables=    38.4
 Disk I/O ——/proc/diskstats———mostly in KB/s————Warning:contains duplicates——
DiskName Busy  Read WriteKB|0          |25         |50          |75       100|
sda        0%    0.0    0.0|                                                  >
sda1       0%    0.0    0.0|>                                                 |
sda2       0%    0.0    0.0|                                                  >
sda3       0%    0.0    0.0|>                                                 |
Totals Read-MB/s=0.0      Writes-MB/s=0.0      Transfers/sec=0.0
 Top Processes Procs=261 mode=3 (1=Basic, 3=Perf 4=Size 5=I/O)————————————————
  PID      %CPU    Size     Res     Res     Res     Res   Shared   Faults  Command
            Used      KB     Set    Text    Data    Lib     KB    Min   Maj
     1093   95.7 4755856  570652   22652  970344      0 17560     0      0 mysqld
    29008   91.7 1247816  192436      12 1232772      0  3084   836      0 server1
     7490   69.3 1003428    5888      36    7292      0  1708    19      0 client_timer
    12396   37.9 2398192  735444      16  735960      0  4276    75      0 server2
     3964    3.5  716384  482484    8500  680200      0 18660     0      0 TeamViewer_Desk
```

Figure 7: New Bottleneck for file size 100KB



```
pro@isrdc74: ~/Desktop/cs744/ph3...  ×   pro@isrdc74: ~/Desktop/cs744/ph3...  ×   pro@isrdc74: ~/Desktop/cs744/ph3...
nmon─14g────────[H for help]───Hostname=isrdc74────────Refresh= 2secs ───00:38.59──
 CPU Utilisation
---------------------------+-------------------------------------------------+
CPU  User%  Sys% Wait% Idle|0          |25         |50          |75       100|
  1  14.2   4.1   0.0  81.7|UUUUUUUss                                        >
  2  14.4  85.1   0.0   0.5|UUUUUUUssssssssssssssssssssssssssssssssssssssssss>
  3   3.6   3.6   0.0  92.9|Us                                               >
  4  11.8  61.0   0.0  27.2|UUUUUssssssssssssssssssssssssssssss              >
  5   1.0   0.5   0.0  98.5|                                                 >
  6   0.5   0.0   0.0  99.5|                                                 >
  7   1.5   0.5   0.0  98.0|                                                 >
  8   1.0   0.5   0.0  98.5|                                                 >
---------------------------+-------------------------------------------------+
Avg   5.9  19.2   0.0  74.9|UUsssssssss                  >                   |
---------------------------+-------------------------------------------------+
 Memory Stats
               RAM     High      Low     Swap    Page Size=4 KB
 Total MB     7846.2    -0.0     -0.0   8057.0
 Free  MB     3778.2    -0.0     -0.0   8057.0
 Free Percent   48.2%  100.0%  100.0%  100.0%
            MB                  MB                  MB
                     Cached=   372.7     Active=  3192.0
 Buffers=     15.8 Swapcached=     0.0  Inactive =   645.3
 Dirty  =      0.1 Writeback =     0.0  Mapped   =   351.0
 Slab   =     83.7 Commit_AS = 11887.4 PageTables=    46.0
 Disk I/O ──/proc/diskstats────mostly in KB/s─────Warning:contains duplicates──
DiskName Busy  Read WriteKB|0          |25         |50          |75       100|
sda         0%    0.0    0.0|                                                 >
sda1        0%    0.0    0.0|>                                                |
sda2        0%    0.0    0.0|                                                 >
sda3        0%    0.0    0.0|>                                                |
Totals Read-MB/s=0.0      Writes-MB/s=0.0      Transfers/sec=0.0
 Top Processes Procs=260 mode=3 (1=Basic, 3=Perf 4=Size 5=I/O)─────────────────
  PID       %CPU    Size      Res     Res      Res      Res    Shared    Faults  Command
             Used      KB      Set    Text     Data      Lib     KB      Min   Maj
    12631    99.2 5113720   57724      12 5098648        0   3056      99      0 server1
    19374    73.8 1327548    3032      36    4372        0   2708      85      0 client_timer
     1093    18.4 4755856  570652   22652  970344        0  17560       0      0 mysqld
    12396     8.0 2908700  918288      16  919432        0   4276      12      0 server2
     3964     3.5  704096  471276    8500  668032        0  18660       0      0 TeamViewer_Desk
     1480     1.5  237156   36252   11180  210176        0  11904       0      0 teamviewerd
```

Figure 8: New Bottleneck for file size 1MB



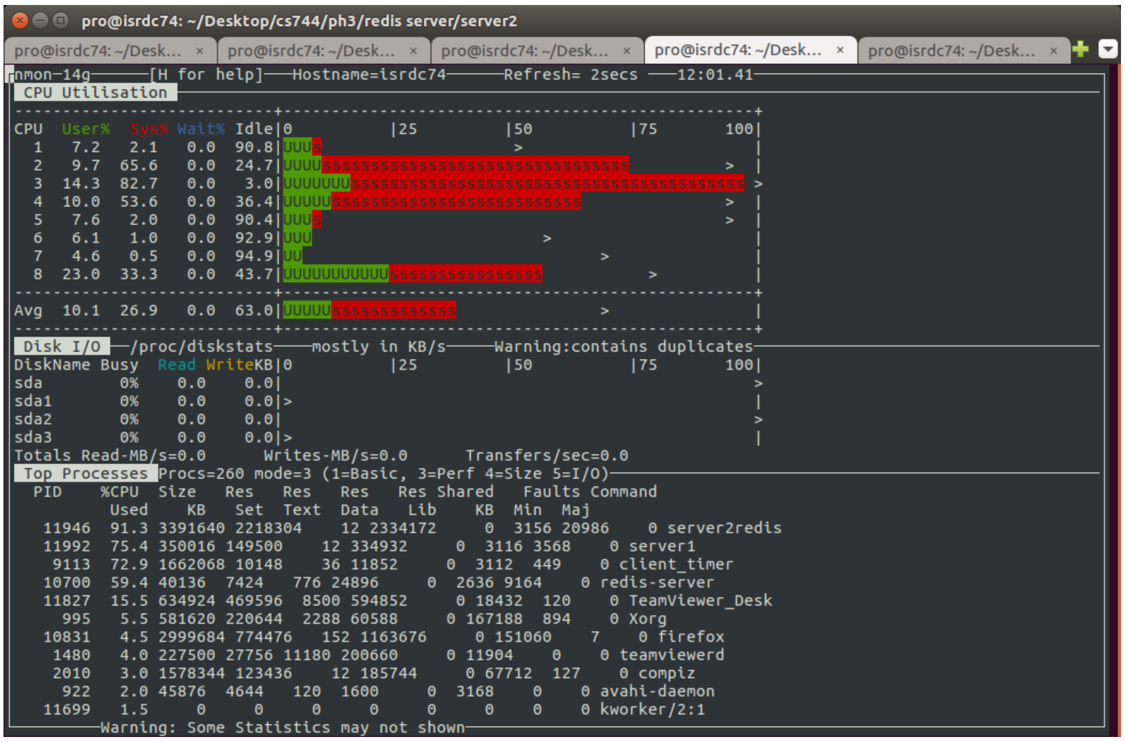
```
pro@isrdc74: ~/Desktop/cs744/ph3...  ×   pro@isrdc74: ~/Desktop/cs744/ph3...  ×   pro@isrdc74: ~/Desktop/cs744/ph3...
nmon─14g─────────────────────────Hostname=isrdc74────────Refresh= 2secs ───01:21.04──
 CPU Utilisation
---------------------------+-------------------------------------------------+
CPU  User%  Sys% Wait% Idle|0          |25         |50          |75       100|
  1   4.5   1.5   0.0  94.0|UU                                               >
  2  19.9  79.6   0.0   0.5|UUUUUUUUUssssssssssssssssssssssssssssssssssssssss>
  3   1.5   0.5   0.0  98.0|                                                 >
  4   7.2  67.0   0.0  25.8|UUUsssssssssssssssssssssssssssssssss              >
  5   2.0   0.5   0.0  97.5|U                                                >
  6   1.0   0.0   0.0  99.0|                                                 >
  7   3.5   1.0   0.0  95.5|U                                                >
  8   3.0   1.0   0.0  96.0|U                                                >
---------------------------+-------------------------------------------------+
Avg   5.2  18.6   0.0  76.2|UUsssssssss                  >                   |
---------------------------+-------------------------------------------------+
 Memory Stats
               RAM     High      Low     Swap    Page Size=4 KB
 Total MB     7846.2    -0.0     -0.0   8057.0
 Free  MB     3696.7    -0.0     -0.0   8057.0
 Free Percent   47.1%  100.0%  100.0%  100.0%
            MB                  MB                  MB
                     Cached=   398.2     Active=  3203.5
 Buffers=      8.3 Swapcached=     0.0  Inactive =   641.8
 Dirty  =      0.0 Writeback =     0.0  Mapped   =   349.8
 Slab   =     82.4 Commit_AS = 54676.3 PageTables=   119.5
 Disk I/O ──/proc/diskstats────mostly in KB/s─────Warning:contains duplicates──
DiskName Busy  Read WriteKB|0          |25         |50          |75       100|
sda         0%    0.0    0.0|                                                 >
sda1        0%    0.0    0.0|>                                                |
sda2        0%    0.0    0.0|                                                 >
sda3        0%    0.0    0.0|>                                                |
Totals Read-MB/s=0.0      Writes-MB/s=0.0      Transfers/sec=0.0
 Top Processes Procs=259 mode=3 (1=Basic, 3=Perf 4=Size 5=I/O)─────────────────
  PID       %CPU    Size      Res     Res      Res      Res    Shared    Faults  Command
             Used      KB      Set    Text     Data      Lib     KB      Min   Maj
    16955    99.8 48861840    7220      12 48846840      0   3020       0      0 server1
    19097    75.4  146676    1668      36      844        0   1512       0      0 client_timer
     3964    11.0  676448  445780    8500   640628        0  18724       0      0 TeamViewer_Desk
      995     3.0  559544  190444    2288    60588        0 136988     638      0 Xorg
     1480     3.0  237156   36340   11180   210176        0  11904       0      0 teamviewerd
```

Figure 9: New Bottleneck for file size 100MB

Hence, we replaced MySQL with Redis database.

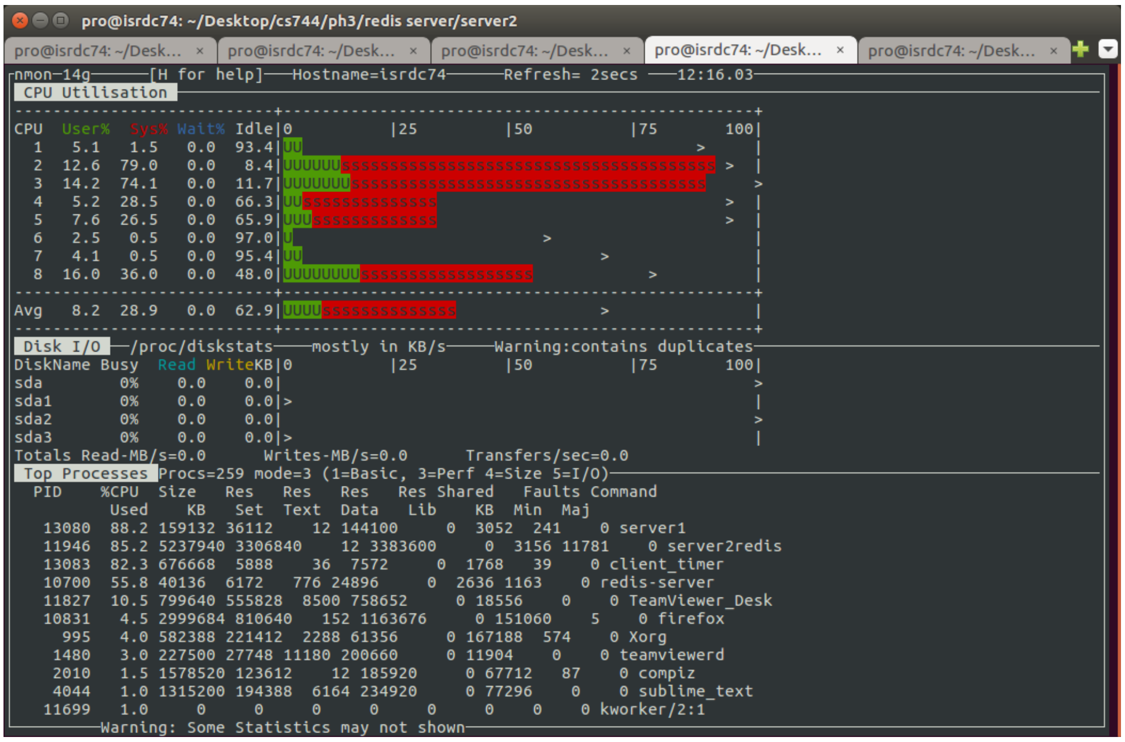
4.4 Step4: Database optimisation using Redis Database

We created username-password as key-value in Redis database and changed server2 code to connect to Redis-server. For "new" request and "get" requests where file size is less than 1MB, there is increase in throughput of system and database was no longer a bottleneck.



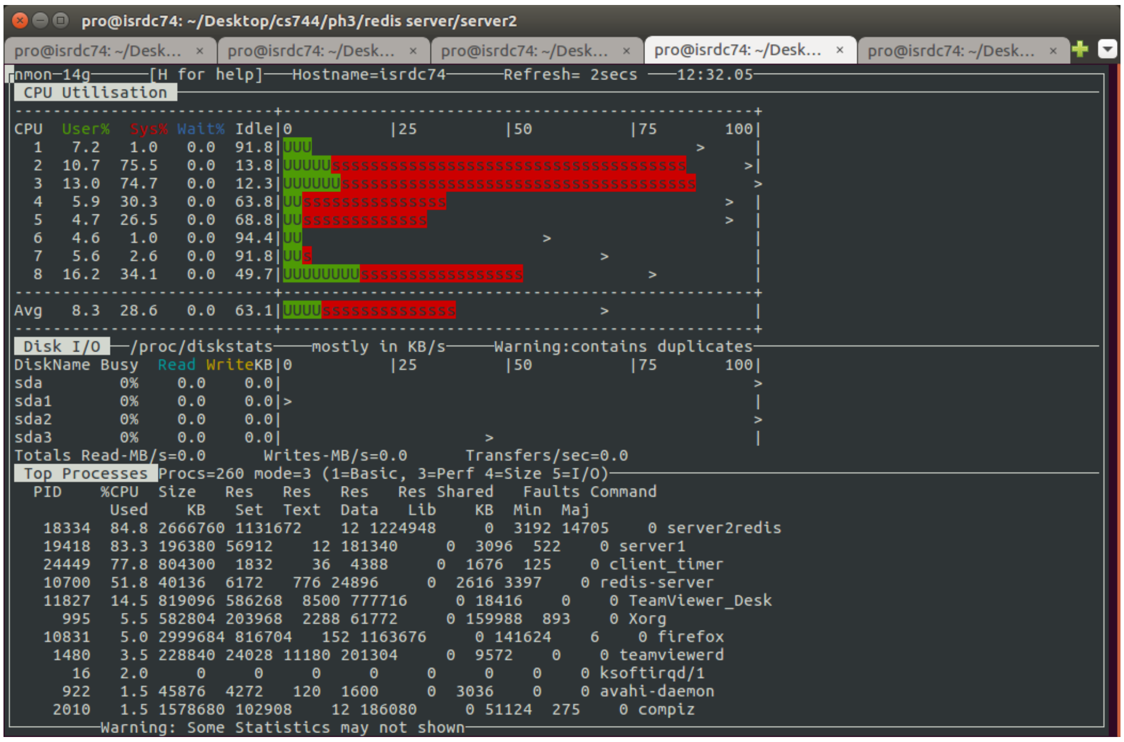
```
pro@isrdc74: ~/Desktop/cs744/ph3/redis server/server2
pro@isrdc74: ~/Desk... × pro@isrdc74: ~/Desk... × pro@isrdc74: ~/Desk... × pro@isrdc74: ~/Desk... × pro@isrdc74: ~/Desk... ×
nmon—14g———[H for help]——Hostname=isrdc74———Refresh= 2secs ——12:01.41—
 CPU Utilisation
---------------------------+-------------------------------------------------+
CPU  User%  Sys% Wait% Idle|0          |25          |50          |75       100|
  1   7.2   2.1   0.0  90.8|UUUs                       >                     |
  2   9.7  65.6   0.0  24.7|UUUUsssssssssssssssssssssssssssssssss          >  |
  3  14.3  82.7   0.0   3.0|UUUUUUUsssssssssssssssssssssssssssssssssssssssssss>
  4  10.0  53.6   0.0  36.4|UUUUUsssssssssssssssssssssssssss                >  |
  5   7.6   2.0   0.0  90.4|UUUs                                            >  |
  6   6.1   1.0   0.0  92.9|UUU                            >                   |
  7   4.6   0.5   0.0  94.9|UU                                   >             |
  8  23.0  33.3   0.0  43.7|UUUUUUUUUUUsssssssssssssssss              >          |
---------------------------+-------------------------------------------------+
Avg  10.1  26.9   0.0  63.0|UUUUUsssssssssssss                   >            |
---------------------------+-------------------------------------------------+
 Disk I/O ——/proc/diskstats———mostly in KB/s———Warning:contains duplicates—
DiskName Busy  Read WriteKB|0          |25          |50          |75       100|
sda         0%    0.0    0.0|                                                 >
sda1        0%    0.0    0.0|>                                                |
sda2        0%    0.0    0.0|                                                 >
sda3        0%    0.0    0.0|>                                                |
Totals Read-MB/s=0.0      Writes-MB/s=0.0      Transfers/sec=0.0
 Top Processes Procs=260 mode=3 (1=Basic, 3=Perf 4=Size 5=I/O)—
  PID     %CPU  Size   Res   Res   Res   Res Shared   Faults Command
           Used    KB   Set  Text  Data   Lib    KB  Min  Maj
   11946  91.3 3391640 2218304    12 2334172     0  3156 20986    0 server2redis
   11992  75.4 350016 149500    12 334932     0  3116 3568    0 server1
    9113  72.9 1662068 10148    36 11852     0  3112  449    0 client_timer
   10700  59.4 40136  7424   776 24896     0  2636 9164    0 redis-server
   11827  15.5 634924 469596  8500 594852     0 18432  120    0 TeamViewer_Desk
     995   5.5 581620 220644  2288 60588     0 167188  894    0 Xorg
   10831   4.5 2999684 774476   152 1163676     0 151060    7    0 firefox
    1480   4.0 227500 27756 11180 200660     0 11904    0    0 teamviewerd
    2010   3.0 1578344 123436    12 185744     0 67712  127    0 compiz
     922   2.0 45876  4644   120  1600     0  3168    0    0 avahi-daemon
   11699   1.5     0     0     0     0     0     0    0    0 kworker/2:1
—————————Warning: Some Statistics may not shown—
```

Figure 10: New bottleneck for file size 1KB



```
pro@isrdc74: ~/Desktop/cs744/ph3/redis server/server2
pro@isrdc74: ~/Desk... × pro@isrdc74: ~/Desk... × pro@isrdc74: ~/Desk... × pro@isrdc74: ~/Desk... × pro@isrdc74: ~/Desk... ×
nmon—14g———[H for help]——Hostname=isrdc74———Refresh= 2secs ——12:16.03—
 CPU Utilisation
---------------------------+-------------------------------------------------+
CPU  User%  Sys% Wait% Idle|0          |25          |50          |75       100|
  1   5.1   1.5   0.0  93.4|UU                                         >      |
  2  12.6  79.0   0.0   8.4|UUUUUUsssssssssssssssssssssssssssssssssssssss >  |
  3  14.2  74.1   0.0  11.7|UUUUUUUsssssssssssssssssssssssssssssssssssss     >
  4   5.2  28.5   0.0  66.3|UUssssssssssssss                                >  |
  5   7.6  26.5   0.0  65.9|UUUsssssssssssss                                >  |
  6   2.5   0.5   0.0  97.0|U                              >                   |
  7   4.1   0.5   0.0  95.4|UU                                   >             |
  8  16.0  36.0   0.0  48.0|UUUUUUUUssssssssssssssssss                >          |
---------------------------+-------------------------------------------------+
Avg   8.2  28.9   0.0  62.9|UUUUsssssssssssssss                  >            |
---------------------------+-------------------------------------------------+
 Disk I/O ——/proc/diskstats———mostly in KB/s———Warning:contains duplicates—
DiskName Busy  Read WriteKB|0          |25          |50          |75       100|
sda         0%    0.0    0.0|                                                 >
sda1        0%    0.0    0.0|>                                                |
sda2        0%    0.0    0.0|                                                 >
sda3        0%    0.0    0.0|>                                                |
Totals Read-MB/s=0.0      Writes-MB/s=0.0      Transfers/sec=0.0
 Top Processes Procs=259 mode=3 (1=Basic, 3=Perf 4=Size 5=I/O)—
  PID     %CPU  Size   Res   Res   Res   Res Shared   Faults Command
           Used    KB   Set  Text  Data   Lib    KB  Min  Maj
   13080  88.2 159132 36112    12 144100     0  3052  241    0 server1
   11946  85.2 5237940 3306840    12 3383600     0  3156 11781    0 server2redis
   13083  82.3 676668  5888    36  7572     0  1768   39    0 client_timer
   10700  55.8 40136  6172   776 24896     0  2636 1163    0 redis-server
   11827  10.5 799640 555828  8500 758652     0 18556    0    0 TeamViewer_Desk
   10831   4.5 2999684 810640   152 1163676     0 151060    5    0 firefox
     995   4.0 582388 221412  2288 61356     0 167188  574    0 Xorg
    1480   3.0 227500 27748 11180 200660     0 11904    0    0 teamviewerd
    2010   1.5 1578520 123612    12 185920     0 67712   87    0 compiz
    4044   1.0 1315200 194388  6164 234920     0 77296    0    0 sublime_text
   11699   1.0     0     0     0     0     0     0    0    0 kworker/2:1
—————————Warning: Some Statistics may not shown—
```

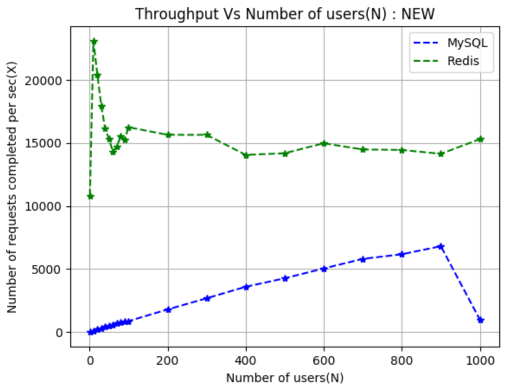
Figure 11: New bottleneck for file size 10KB



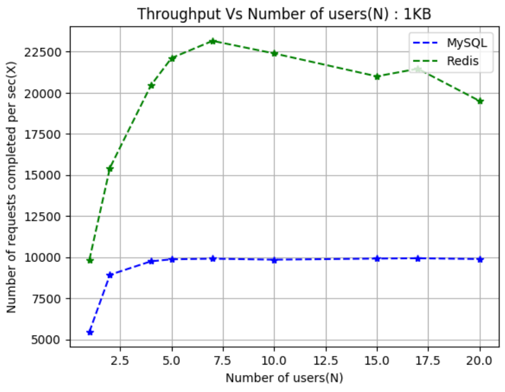
```
pro@isrdc74: ~/Desktop/cs744/ph3/redis server/server2
pro@isrdc74: ~/Desk... ×  pro@isrdc74: ~/Desk... ×  pro@isrdc74: ~/Desk... ×  pro@isrdc74: ~/Desk... ×  pro@isrdc74: ~/Desk... ×
┌nmon─14g──────[H for help]───Hostname=isrdc74──────Refresh= 2secs ───12:32.05─
│ CPU Utilisation ─────────────────────────────────────────────────────────────
│---------------------------+-------------------------------------------------+
│CPU  User%  Sys% Wait%  Idle|0          |25         |50          |75       100|
│  1   7.2   1.0   0.0  91.8|UUU                                         >   |
│  2  10.7  75.5   0.0  13.8|UUUUUsssssssssssssssssssssssssssssssssssss      >|
│  3  13.0  74.7   0.0  12.3|UUUUUUsssssssssssssssssssssssssssssssssssss      >
│  4   5.9  30.3   0.0  63.8|UUsssssssssssssss                            >   |
│  5   4.7  26.5   0.0  68.8|UUsssssssssssss                              >   |
│  6   4.6   1.0   0.0  94.4|UU                         >                     |
│  7   5.6   2.6   0.0  91.8|UUs                              >               |
│  8  16.2  34.1   0.0  49.7|UUUUUUUUsssssssssssssssss             >          |
│---------------------------+-------------------------------------------------+
│Avg   8.3  28.6   0.0  63.1|UUUUssssssssssssss               >               |
│---------------------------+-------------------------------------------------+
│ Disk I/O ──/proc/diskstats────mostly in KB/s─────Warning:contains duplicates─
│DiskName Busy  Read WriteKB|0          |25         |50          |75       100|
│sda         0%    0.0    0.0|                                                >
│sda1        0%    0.0    0.0|>                                               |
│sda2        0%    0.0    0.0|                                                >
│sda3        0%    0.0    0.0|                      >                         |
│Totals Read-MB/s=0.0       Writes-MB/s=0.0       Transfers/sec=0.0
│ Top Processes Procs=260 mode=3 (1=Basic, 3=Perf 4=Size 5=I/O)───────────────
│  PID    %CPU  Size    Res    Res    Res    Res Shared    Faults Command
│          Used     KB    Set   Text   Data    Lib     KB   Min   Maj
│   18334  84.8 2666760 1131672    12 1224948      0  3192 14705     0 server2redis
│   19418  83.3 196380 56912     12 181340      0  3096   522     0 server1
│   24449  77.8 804300  1832     36  4388      0  1676   125     0 client_timer
│   10700  51.8 40136  6172    776 24896      0  2616 3397     0 redis-server
│   11827  14.5 819096 586268  8500 777716      0 18416     0     0 TeamViewer_Desk
│     995   5.5 582804 203968  2288 61772      0 159988   893     0 Xorg
│   10831   5.0 2999684 816704   152 1163676      0 141624     6     0 firefox
│    1480   3.5 228840 24028 11180 201304      0  9572     0     0 teamviewerd
│      16   2.0      0      0      0      0      0      0     0     0 ksoftirqd/1
│     922   1.5 45876  4272    120  1600      0  3036     0     0 avahi-daemon
│    2010   1.5 1578680 102908    12 186080      0 51124   275     0 compiz
└─────────Warning: Some Statistics may not shown──────────────────────────────
```

Figure 12: New bottleneck for file size 100KB

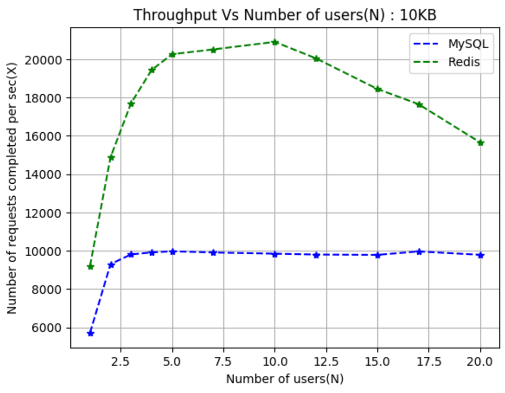
4.4.1 Performance after database optimisation



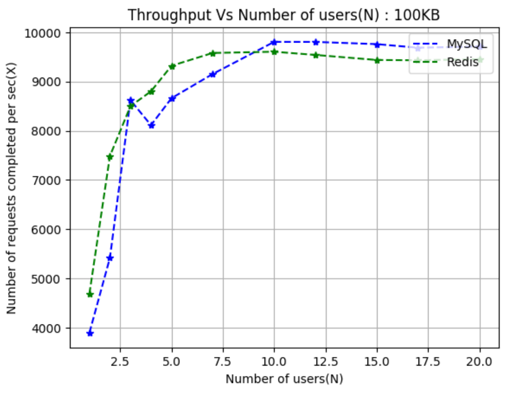
(a)



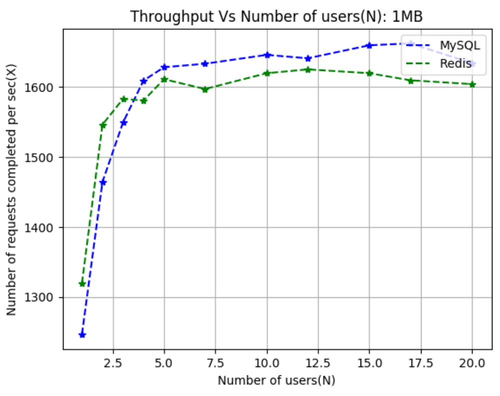
(b)



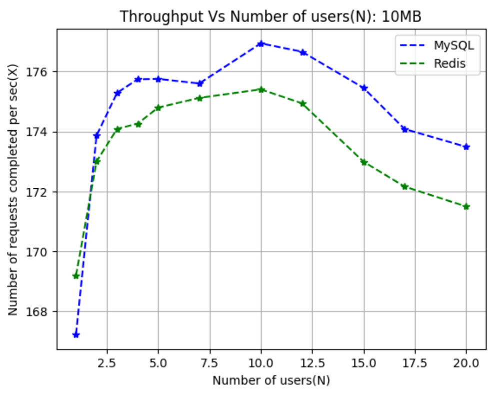
(c)



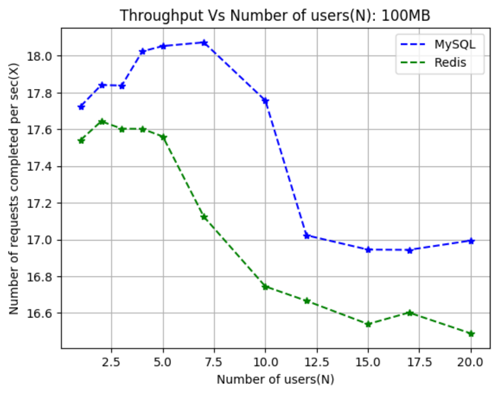
(d)



(e)



(f)

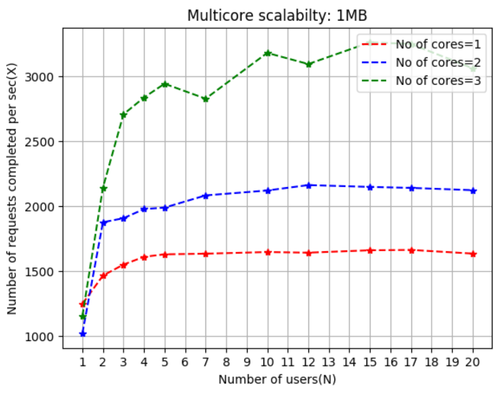


(g)

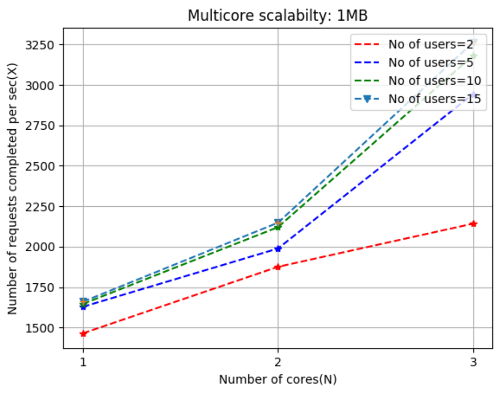
Figure 13: Performance after change in database

5 Multicore scalability

For file size=1MB, the server1 was limiting component and utilizing 96-98% CPU , so we increased number of cores using taskset. As we increased number of cores from 1 to 3, the throughput increased.



(a)



(b)

Figure 14: Multicore scalabilty

6 Summary of results

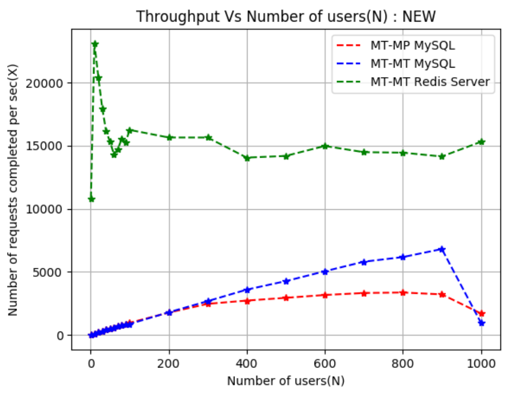
We first changed our server2 from multi process to multi thread and compared their performance. It was found that multi-threaded systems give better performance when there is no shared data and overhead of synchronization.

Then, in order to remove disk as bottleneck, we pre-fetched the files into user space by allocating memory to file contents using malloc. This improved the throughput a lot and we removed disk as bottleneck. We found that new bottleneck is MySQL for file sizes less than 100KB and for other cases, bottleneck was CPU core running server. So we changed our backend database from MySQL to Redis.

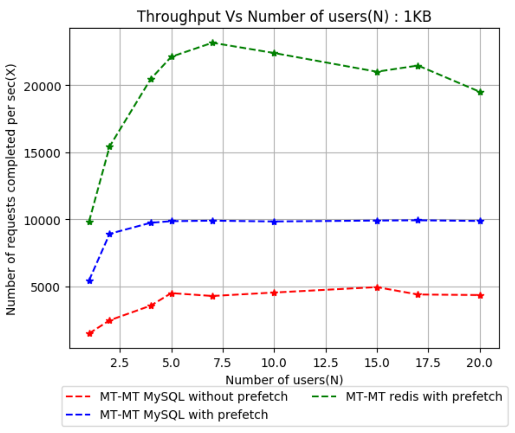
In cases where MySQL was bottleneck ie for file size less than 100KB, using Redis database increased the throughput. Also, for "new" request where disk was bottleneck for MySQL server, Redis server gave better performance because Redis is in-memory database platform and it don't write to disk continuously but saves the data to disk periodically from memory as per configuration.However for files with larger size(>100KB) the performance of systems with MySQL database and Redix Database was comparable. In such cases, bottleneck was CPU core running server.

Load type	File size	**Max Throughput**		
		MT-MT without prefecting and MySQL DB	**MT-MT with prefetching and MySQL DB**	**MT-MT with prefetching and RedisDB**
new		6806.03	6806.03	23089.8
get	1KB	4954.06	9932.58	23154.2
get	10KB	3453.44	9973.76	20907.1
get	100KB	1139.57	9804.96	9604.82
get	1MB	135.965	1633.88	1624.99
get	10MB	12.508	176.932	175.399
get	100MB	1.14016	18.0733	17.6434

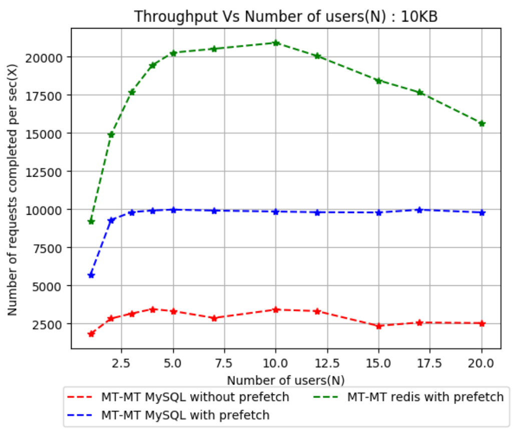
Table 4: Summary of results


Throughput Vs Number of users(N) : NEW
MT-MP MySQL
MT-MT MySQL
MT-MT Redis Server
Number of requests completed per sec(X)
Number of users(N)

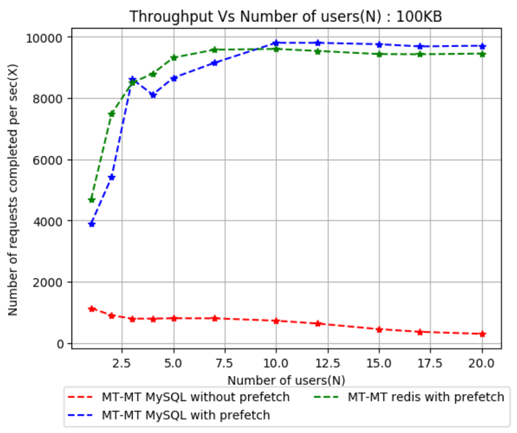
(a)


Throughput Vs Number of users(N) : 1KB
Number of requests completed per sec(X)
Number of users(N)
MT-MT MySQL without prefetch
MT-MT MySQL with prefetch
MT-MT redis with prefetch

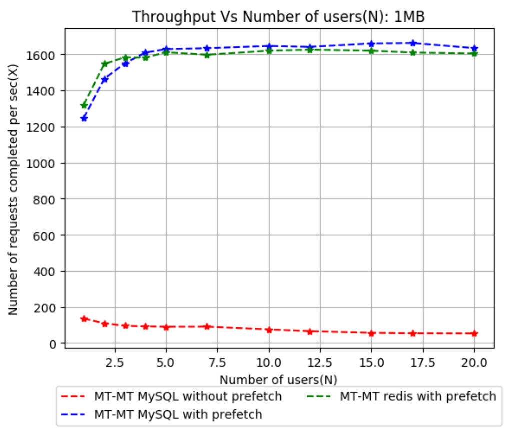
(b)


Throughput Vs Number of users(N) : 10KB
Number of requests completed per sec(X)
Number of users(N)
MT-MT MySQL without prefetch
MT-MT MySQL with prefetch
MT-MT redis with prefetch

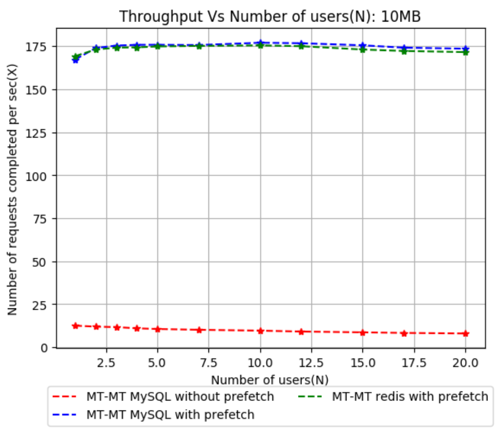
(c)


Throughput Vs Number of users(N) : 100KB
Number of requests completed per sec(X)
Number of users(N)
MT-MT MySQL without prefetch
MT-MT MySQL with prefetch
MT-MT redis with prefetch

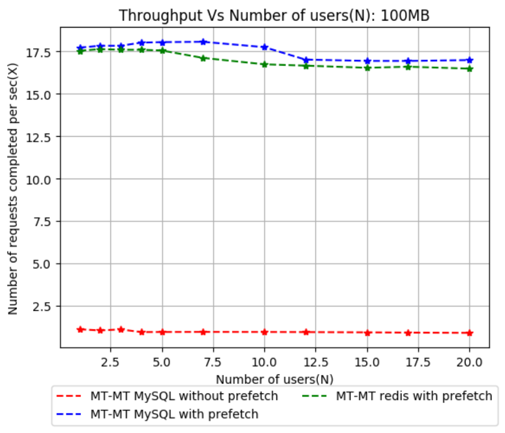
(d)



(e)



(f)



(g)

Figure 15: Summary of results

7 Submission folder

There are five folders inside : loadgen, ph2, multithreaded ,prefetching and redis.The ph2, multithreaded, prefetching and redis folder contains server1 and server2 sub-folders for respective stages of optimsations.

The folder/directory structure and files are listed below. The loadgen folder contains:

- Makefile
- client.cpp : Load generator (with Number of request as input)
- client_timer.cpp : Load generator (with run time as input)
- "downloads" folder : to store received files.

To run client/ load generator:
./client Server1_IP Server1_Listening-port request_type
Each server1 sub-folder contains:

- Makefile

- server1.cpp : File server code.
- "uploads" folder : to store files available for sharing/download.
- filecreate.sh : To create files with random content of specified size.
- clearbuffer.sh : To clear contents of cache buffer.
- tunesocket.sh : To tune the socket parameters.

Each server2 sub-folder of ph2, multithreaded and prefetching folder contains:

- Makefile
- server2.cpp : Authentication server code.
- installmysql.sh : script to install MySQL server and required libraries for header files.
- importdatabase.sh : script to import database and user table from cs744.sql file.
- cs744.sql : contains MySQL database and table for import.

Each server2 sub-folder of redis folder contains:

- Makefile
- server2.cpp : Authentication server code.
- installredis.sh : script to install Redis server and required libraries for header files.