CS 258 Assignment 1

Tongzhou Gu, Li Zhou

1 Question 1

In quick sort, all the data have to be load into the memory, then there will be a base data been selected and compared with the rest of the data, other data will switch position with the base data. Since the CPU cache size, CPU cache schedule algorithm have an impact on the performance, to simply the problem, we don't consider it here.

By considering this term, the operation can be categorized into 2 steps, compare and switch.

1. The compare operation consist of reading data from memory to register, then compare.
2. (a) The switch operation consist of reading data from the memory and write the data to the new position. In total there're 2 reads and 2 writes

Compare operation takes 1 cycle, and memory read/write need 100 more cycles. So, the bottleneck will be in the memory speed.

Considering the CPU has 3GHz speed, and per comparation has 50% possibility to trigger a data switch operation, so per second, there are 1.5×10^9 data switch needed, which means 24GB/s Memory bandwidth is needed, the modern DDR4-3200Mhz Memory support 25.6GB/s, so the memory bandwidth is not the problem, the memory latency is.

2 Question 2

the index of array range from 1 to $1 << 24$. Considering it's a 32-bit system so the integer will take 32 bits memory. the size of all the data is $4 \times (1 << 24) = 64\,MB$.

If we count the probability that the next random operation will happen in the same cache will help to reduce the execution time. The L1 cache is 32KB, this means there's only 0.049%, the L2 cache is 2MB, the probability is 3.125%, it's still not that much. But if with a 16MB L3, the probability is 25%, it's better. So adding a 16MB L3 will help.

3 Question 3

Here the `random()` function returns a random value from 0 to $2^{31} - 1$, so here we have 2 cases.

Case 1,$n < 2^{31} - 1$.

This case is simpler, since the return value r is ranged from 0 to $2^{31} - 1$, we can easily get a random number ranged from 0 to n by doing

$$\frac{r}{2^{31} - 1} \times n$$

So that the random number is ranged from 0 to n

Case 2, $n > 2^{31} - 1$

In this case, the `int` type, if the n is bigger, the `unsigned long long` type can't present the n. We need to implement our own number system to hold bigger numbers.

In Case 1 we already have an uniform random number generator with a range n. So and the $2^{31}-1$ means `0x7fffffff`, which means `1` repeated 31 times in binary representation. So for a bigger number, we can divide the bigger number as binary blocks

```
a = |block 1 | block 2 |...| block n |
```

Each block $a_n$means a binary representation ranged from 0 to `0x7fffffff`. Then we generate a random number with a range from 0 to a_n , since each block, the random number is uniform, then chain them all, the result is also uniform. So that by chaining the blocks, we have a bigger range random number generator

4 Question 4

Here we don't need to consider the big number as mean value. By searching we got the algorithm to generate the Poisson distribution

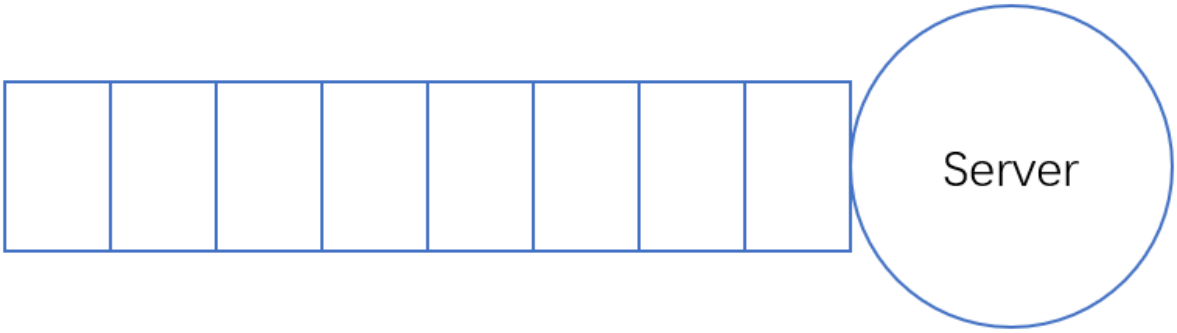
```
algorithm poisson random number (Knuth):
    init:
         Let L ← e-λ, k ← 0 and p ← 1.
    do:
         k ← k + 1.
         Generate uniform random number u in [0,1] and let p ← p×u.
    while p > L.
    return k − 1.
```

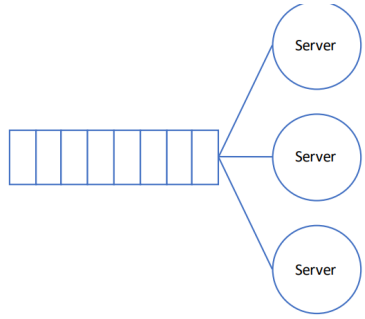
When mean value λ is very big, the $e^{-\lambda}$ is close to 0, when λ is big enough, no matter how big the λ is, the L will not change much.

We write and give out our code in the attachment `a1q4.cpp`

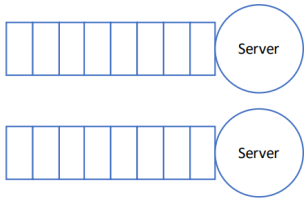
5 Question 5

5.1 a.

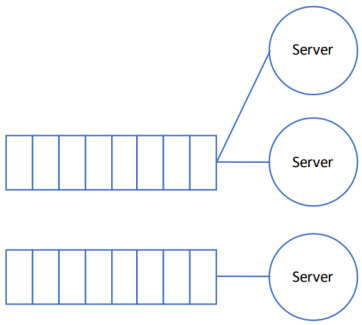




Q5.c. Single Queue Multi Server

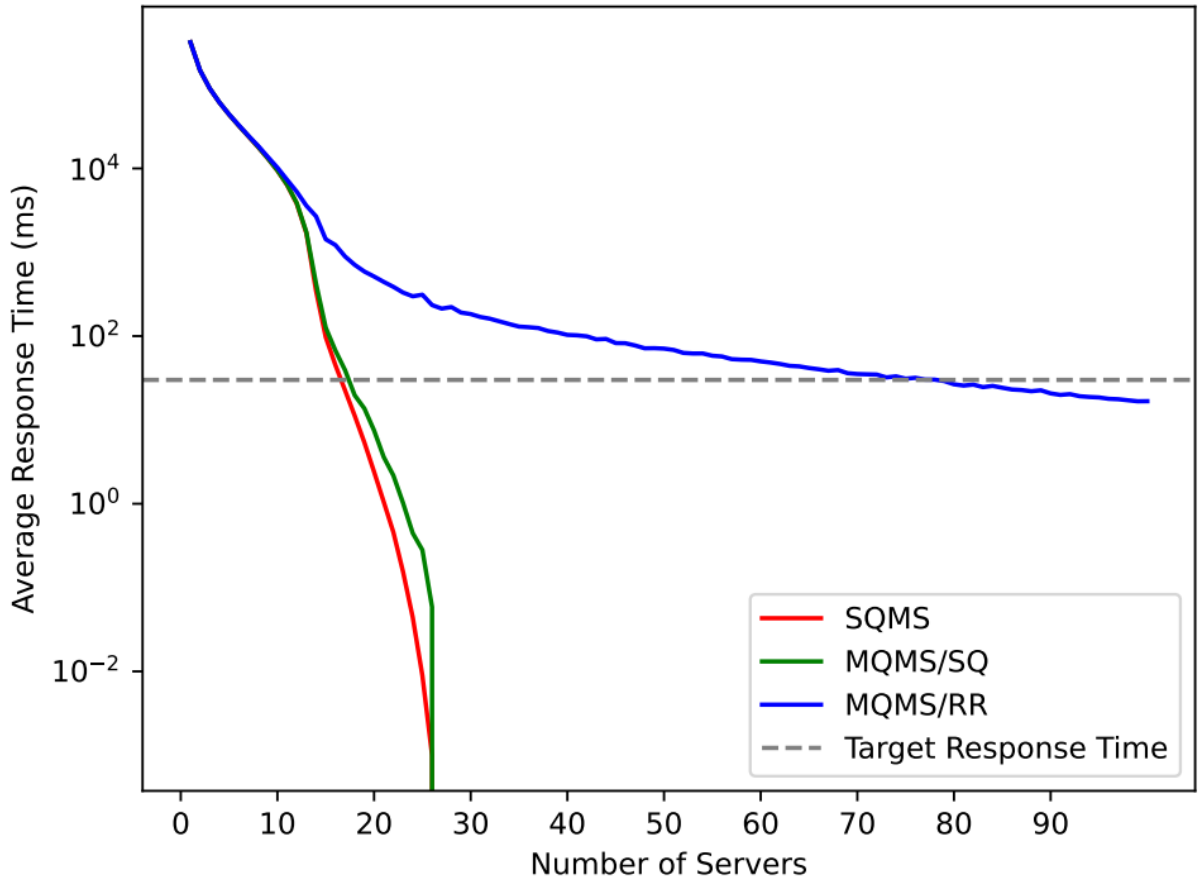


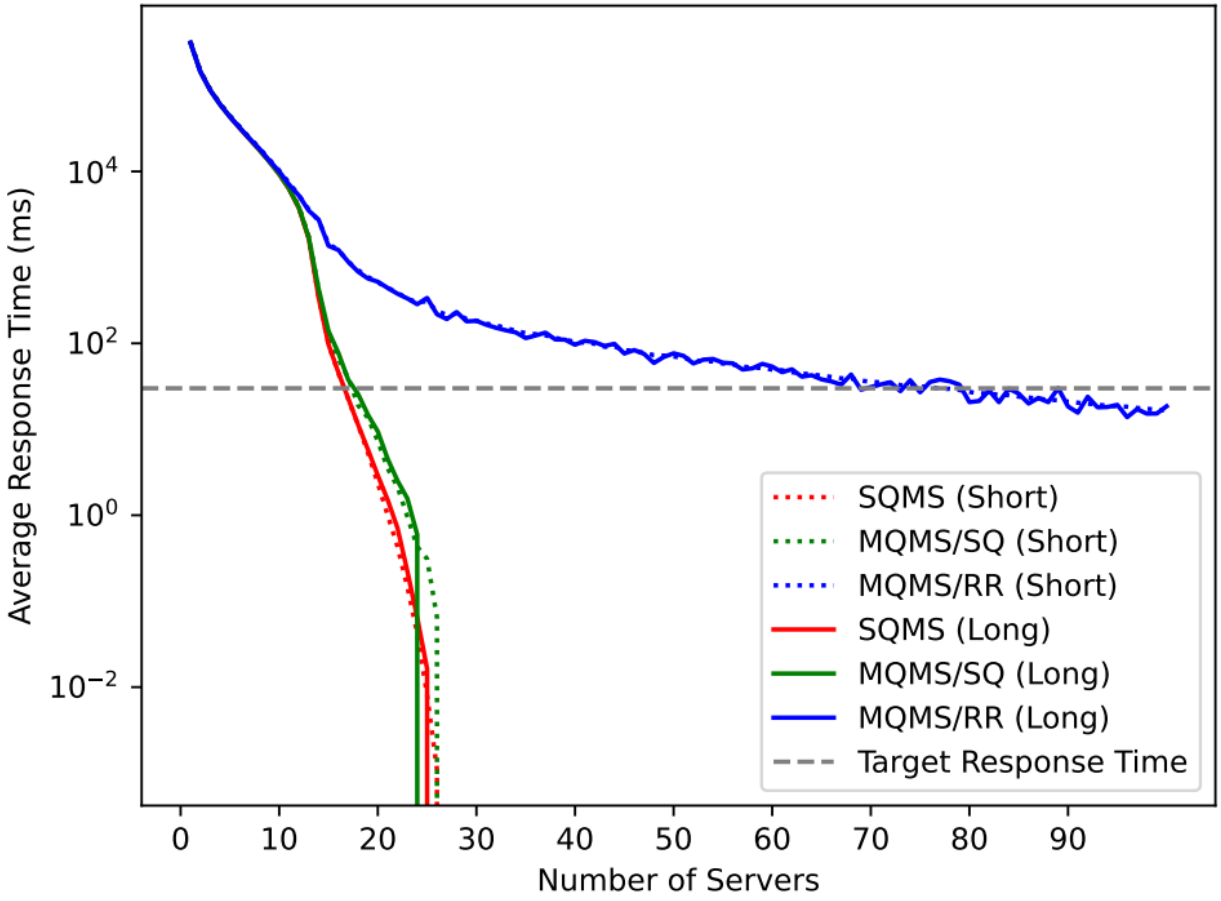
Q5.d/e. Multi Queue
Single Server per Queue

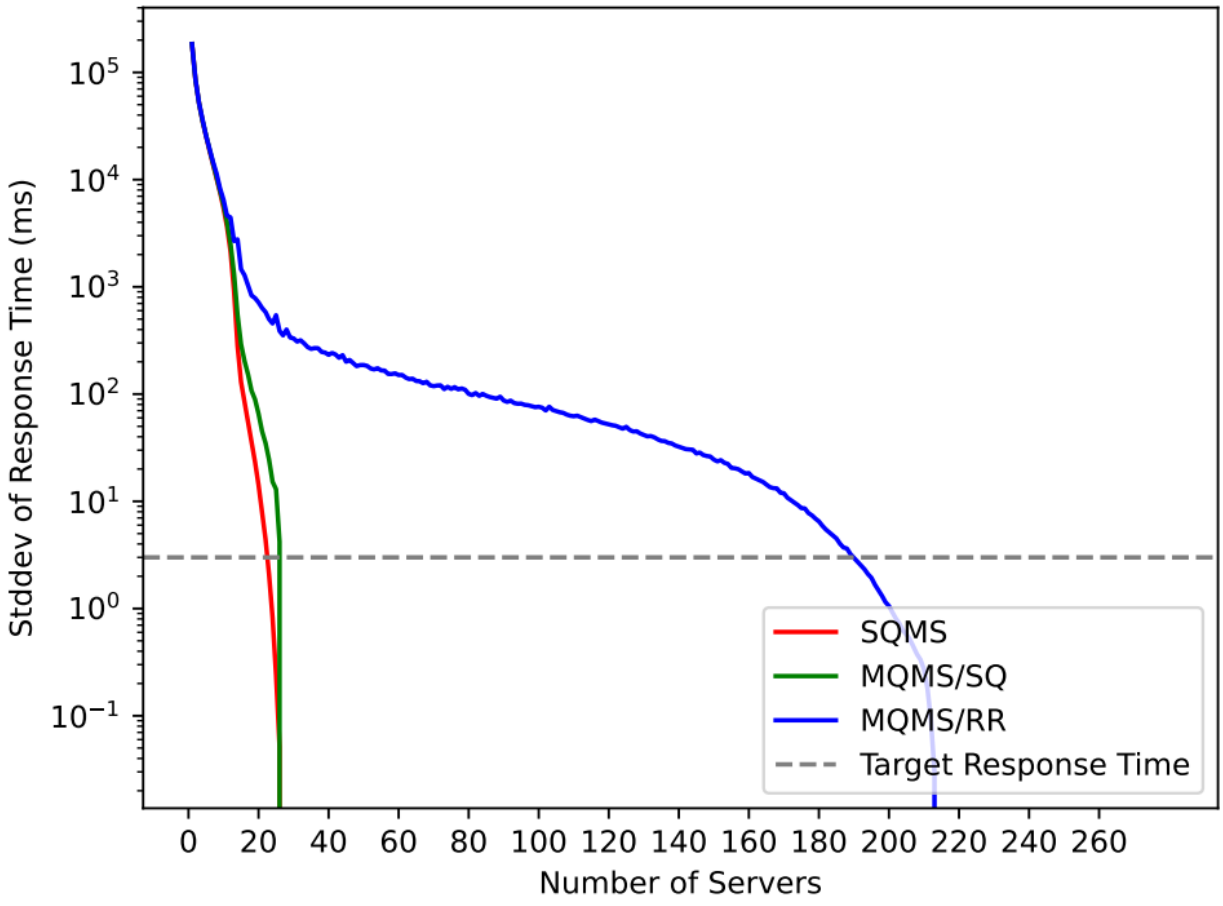


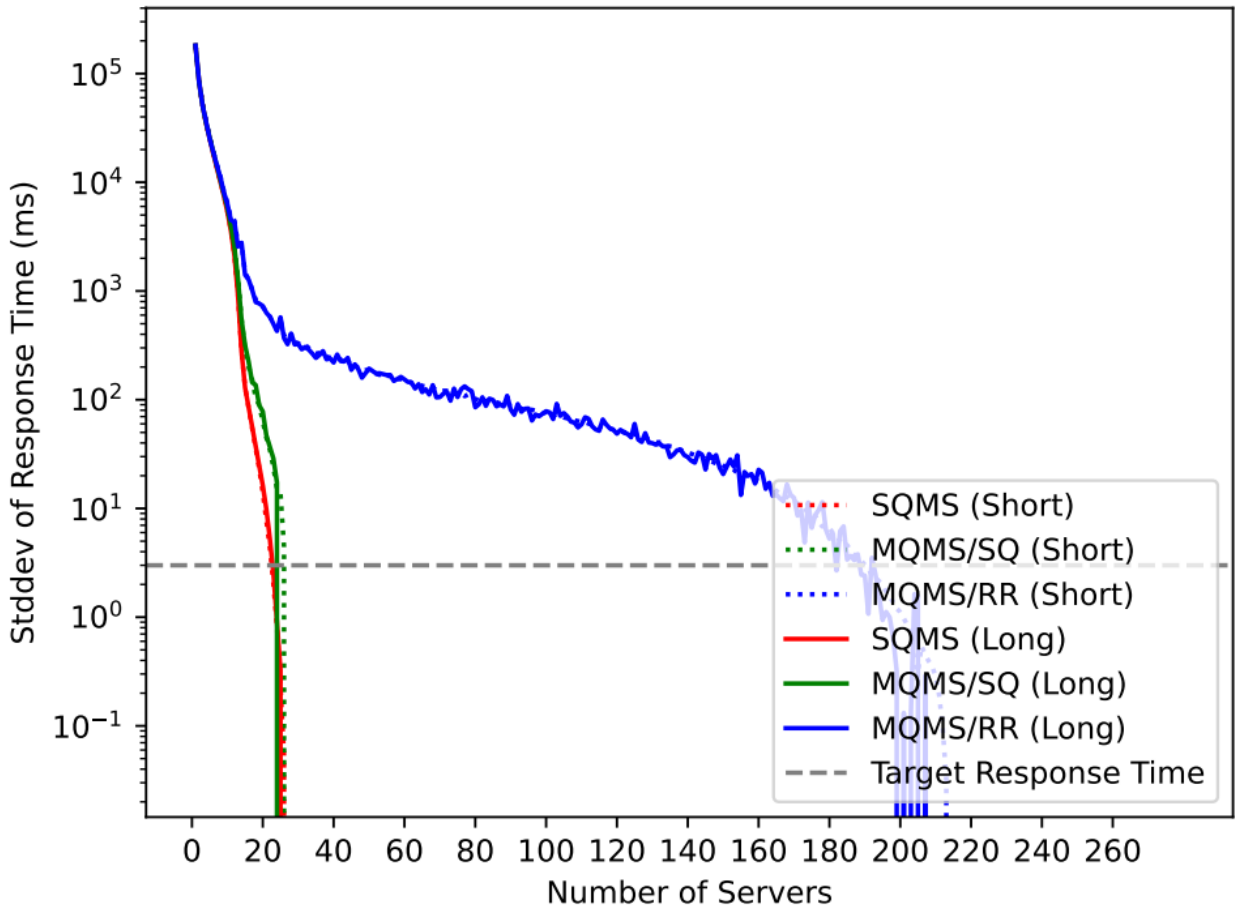
Q5.f. Multi Queue Multi Server

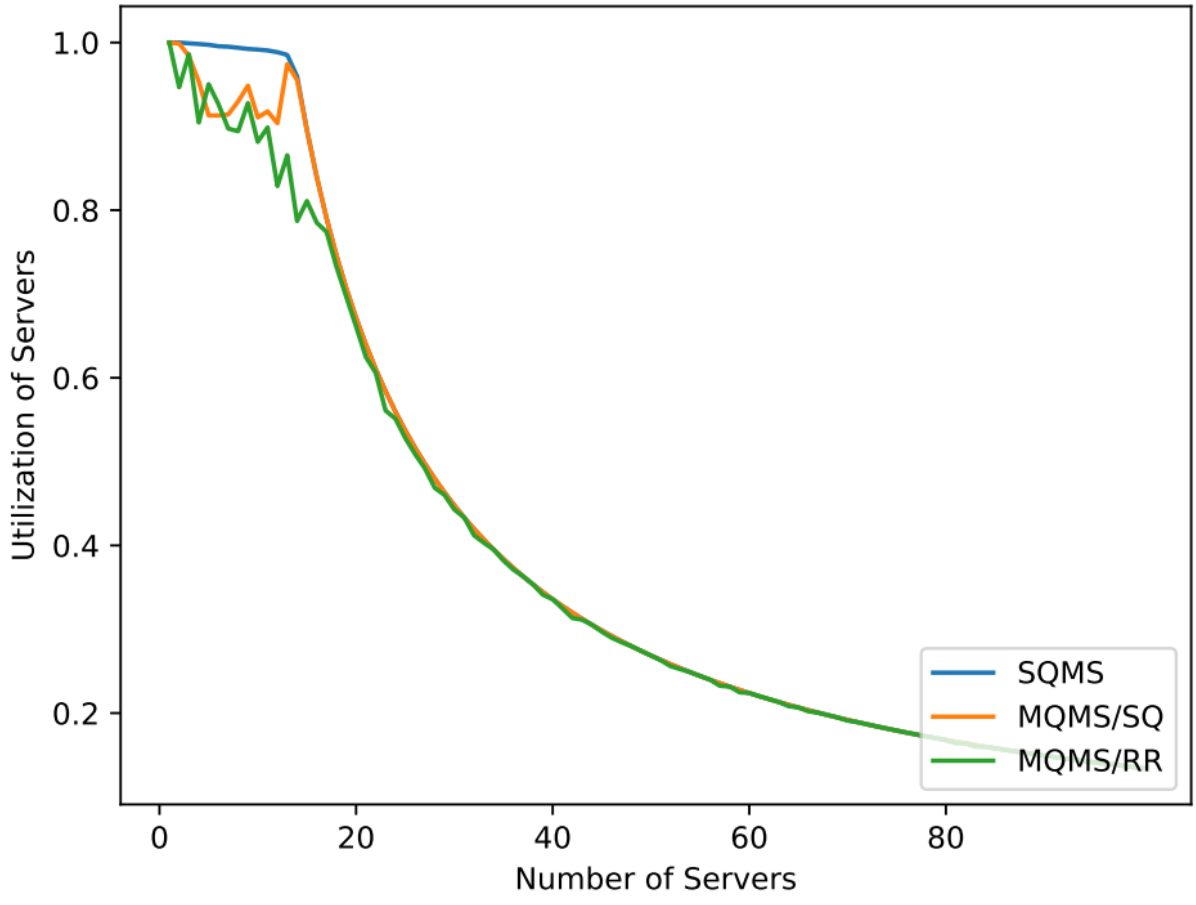
5.2 Figures for b./c./d./e.




Average Response Time (ms)
10^4
10^2
10^0
10^{-2}
0
10
20
30
40
50
60
70
80
90
Number of Servers
SQMS (Short)
MQMS/SQ (Short)
MQMS/RR (Short)
SQMS (Long)
MQMS/SQ (Long)
MQMS/RR (Long)
Target Response Time


Stddev of Response Time (ms)
10^5
10^4
10^3
10^2
10^1
10^0
10^{-1}
0
20
40
60
80
100
120
140
160
180
200
220
240
260
Number of Servers
SQMS
MQMS/SQ
MQMS/RR
Target Response Time


Stddev of Response Time (ms)
10^5
10^4
10^3
10^2
10^1
10^0
10^{-1}
0 20 40 60 80 100 120 140 160 180 200 220 240 260
Number of Servers
SQMS (Short)
MQMS/SQ (Short)
MQMS/RR (Short)
SQMS (Long)
MQMS/SQ (Long)
MQMS/RR (Long)
Target Response Time


Utilization of Servers
1.0
0.8
0.6
0.4
0.2
0 20 40 60 80
Number of Servers
SQMS
MQMS/SQ
MQMS/RR

b.

Here we set there are 10000 requests.

The average response time is 331003.5ms the standard deviation of response time is 191682.3ms

For the slow requests, the average and std of response time is 328439.1ms and 190182.9ms

For the fast requests, the average and std of response time is 331293.5ms and 191849.3ms

The average utilization of the server is 100%

c.

24 Servers. Utilization: ~55%

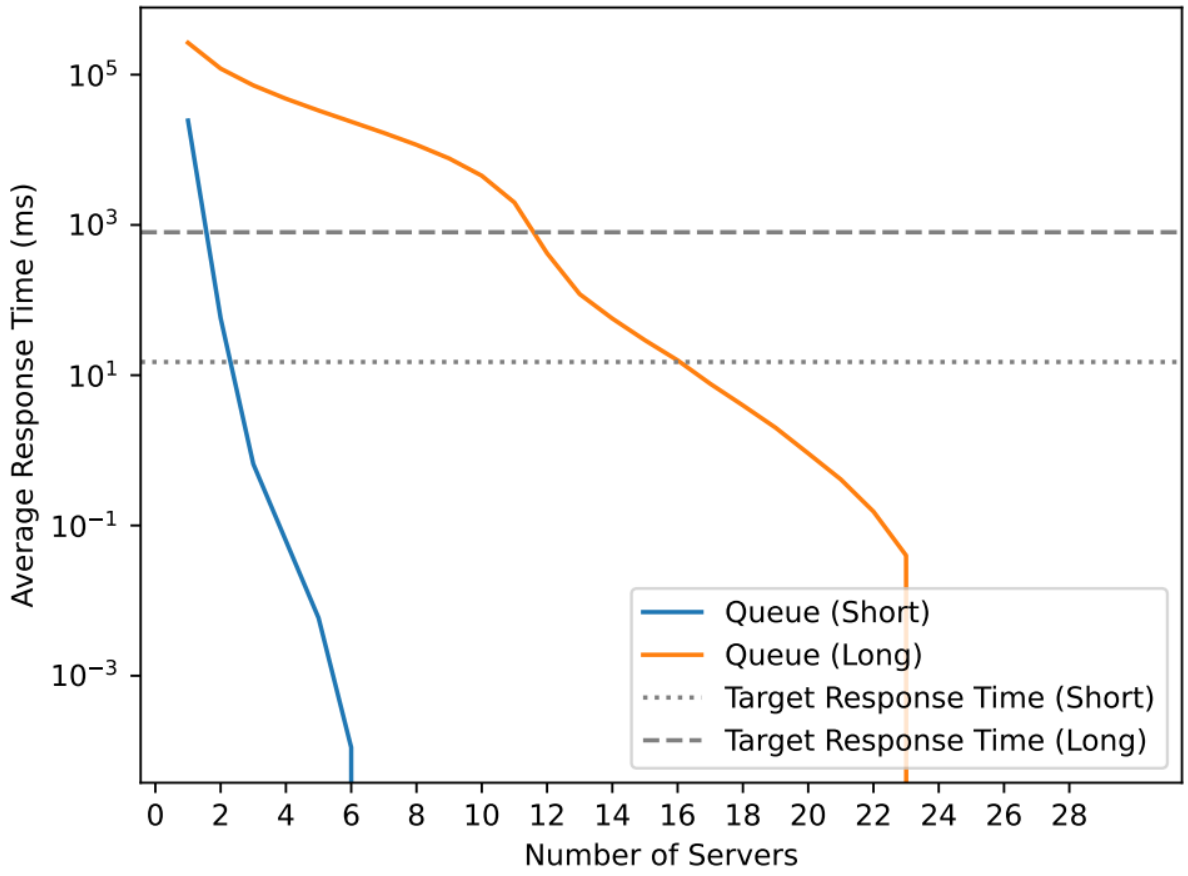
d.

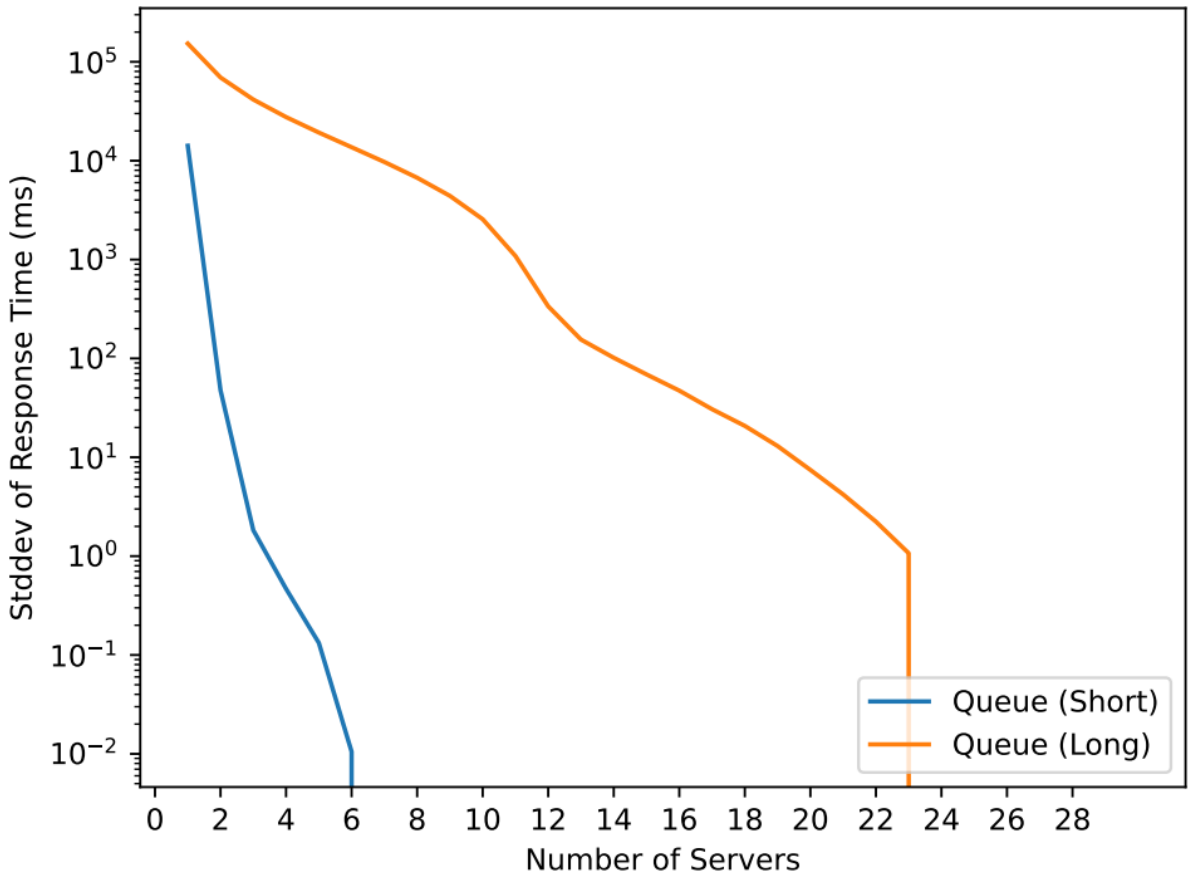
27 Servers. Utilization: ~49%

e.

192 Servers. Utilization: ~7%

5.3 f.





3 Servers for Short Requests. Utilization: ~65%

13 Servers for Long Requests. Utilization: ~88%

5.4 g.

f scheme: relatively easy to achieve performance goal if percentages of each type of workloads are fixed
c, d scheme: could provide good performance compared to e scheme