COMP9334 Project Report

System Design:

Outline:

The fork-join system consists of three modules, preprocessor, servers, and joint. The preprocessor and servers all obey the first in first serve rule. **So, I use the part of M/M/1 program provided in mathlab files.** The process should be like this, when the request arrive in preprocessor just say request_1, if it's busy join in queue, else serve it. After serving it, it breaks up n parts(just say n = 3), and send this sub_requests to the selected servers(selecte n servers randomly among 10 servers).After serving from each selected servers, the sub_request would add into joint. When the number of subtasks equal to n, there would be one integrity task.

Pre_processor:

This module is much like the MM1 but with different arrival rate and service rate. Note that I used the built in function in python to generate the random.

Based on the spec, the next arrival time = master_clock + random.expovariate(0.85) + random.uniform(0.05, 0.25)

The service time next arrival = random.expovariate(10 / n)

The whole simulation is much like MM1, except the arrival and service rate and most importantly, the request would break Into n subtasks, and each of them went to the selected servers.

Server:

This module is also much like MM1 but with different arrival and service rate.Also, I just the built in function in python to generate the random, when the sub_request has been served, the server will sent it to joint directly.

And I just create calss of server with some relevant properties to record each of server.

Based on the spec, the next arrival time = processor.departure_time

Because the service time obey in pareto distribution. The steps below are how I get the service time.

$$f(t_s) = \begin{cases} 0 & \text{for } t_s < \frac{t_m}{n^{1.65}} \\ \frac{1}{n^{1.65k}} \frac{k t_m^k}{t_s^{k+1}} & \text{for } t_s \geq \frac{t_m}{n^{1.65}} \end{cases} \quad (1)$$

1. Based on the formula ,I can compute the ts inversely.The service time ts required by each sub-request At each server is distributed which probability density function is f(ts).Therefore the cumulative distributed function is F(ts) = 1 – ts ^ (-k) * tm ^ k / n ^ (1.65 * k). cof =tm ^k / n ^ (1.65 * k)
2. The inverse transform approach: Y = F(ts): ts = tm ^ k / n ^ (1.65 * k)

Joint:

This module is maintain with a list. The format of it just like that:

Joint_status: [[orig_time_1, number], [orig_time_2, number], […], …]

If the number specified by the orig_time equal to n, which means there is one integrity request.

Then the response time of this request += master_clock – orig_time

Then the number of finished tasks : N += 1

Connection between three modules and make sure right simulatioin:

1. I create the class of preprocessor and servers with some relevant parameters
   such as next_arrival_time, next_departure_time, buffer_content …
2. Initialise the server and processor, I used the server_list = [] to store the 10 servers
   Just like it: server_list[server1, server2, ….]
3. If the master_clock < Tend, store the arrival and departure time of processor and store the departure time of each server in the event_time as a list.
4. Find the min time and specified the event happened on processor or servers and update the master_clock = min_time
5. Based on the event_type and specified server, processing the request just like MM1
6. When the server departure, the sub_request would append in joint directly.And then check the joint status whether meet the condition.
7. Loop 3 – 6
8. Because the processor and servers are much like MM1, but with different rate,**I used the part of MM1 program in mathlab to simulate**. I think the challenge would be how to break up tasks and track each of subtask .First, I use random generator to generate n numbers in[1,10].Second, when the subtask leave to each selected servers, I used orig_time_save to save the arrival time in the processor so that we can compute the response time when it append in joint.
9. You can see more detail in project.py.
10.

How to use program to simulate:

Note that the input format when using the program to simulate

$python 2.7 project.py [n] [Tend] [seed]

For example:

If we want to simulate when n = 3, Tend = 20000, seed = 3:

$python2.7 project.py 3 20000 3

Statistical Analysis:

In order to get how long should the simulation is, I first draw the graph which shows how the mean response time changes based on the simulation for each n(1,2,3,4,5,6,7,8,9,10).
Then through the visual inspection, I can easily find during which period the state is steady.
Next, choose two simulation times are the steady and transient state, and repeat the simulation time but with different random numbers, I just try 10 times.
Next, use the data(T,N) of steady state – data(T,N) of transient state
Next, compute the mean response time, standard deviation and 95% confident interval.
Finally, compare the confident interval with n(1,2,3..10), the n with smallest response time is the result.

Remove transient: $\frac{T(m+1)+T(m+2)+...+T(N)}{N-m}$

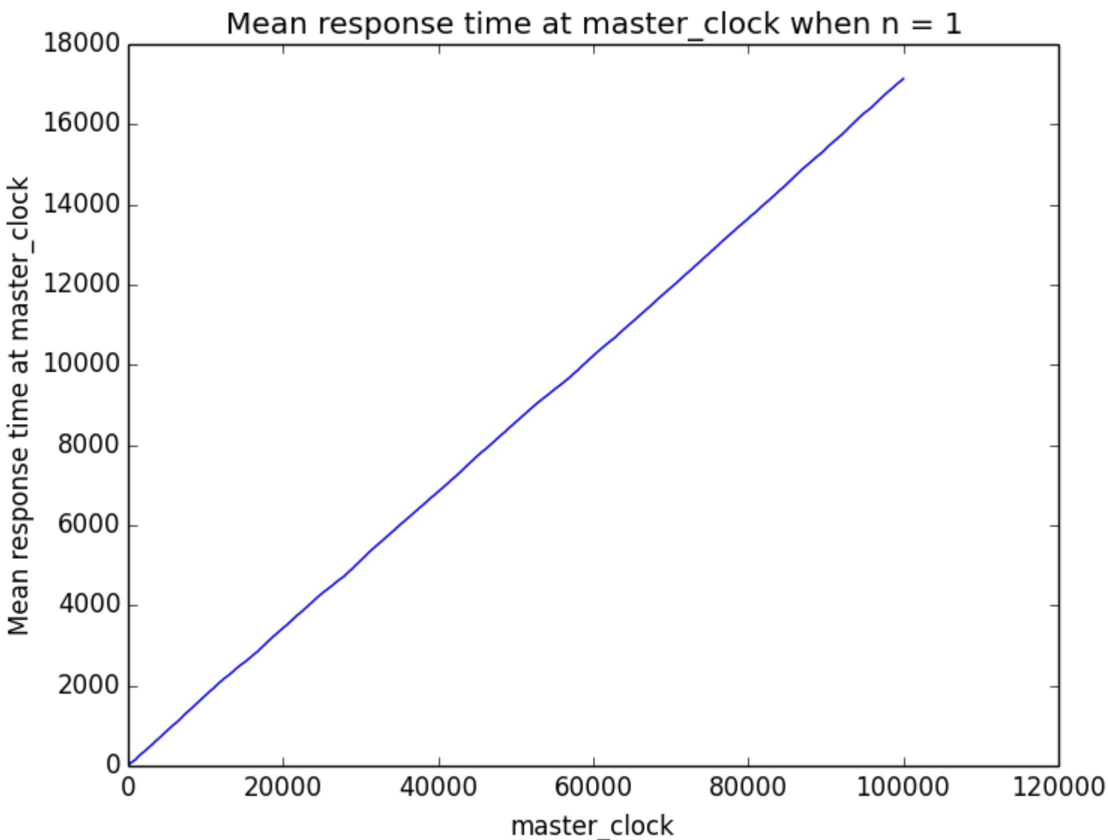
Mean response time: $\hat{T}=\frac{\sum_{i=1}^{n}T(i)}{n}$

Standard deviation: $\hat{S}=\sqrt{\frac{\sum_{i=1}^{n}(\hat{T}-T(i))^2}{n-1}}$

Confident interval: $[\hat{T}-t_{n-1,1-\frac{\alpha}{2}}\frac{\hat{S}}{\sqrt{n}},\hat{T}+t_{n-1,1-\frac{\alpha}{2}}\frac{\hat{S}}{\sqrt{n}}]$

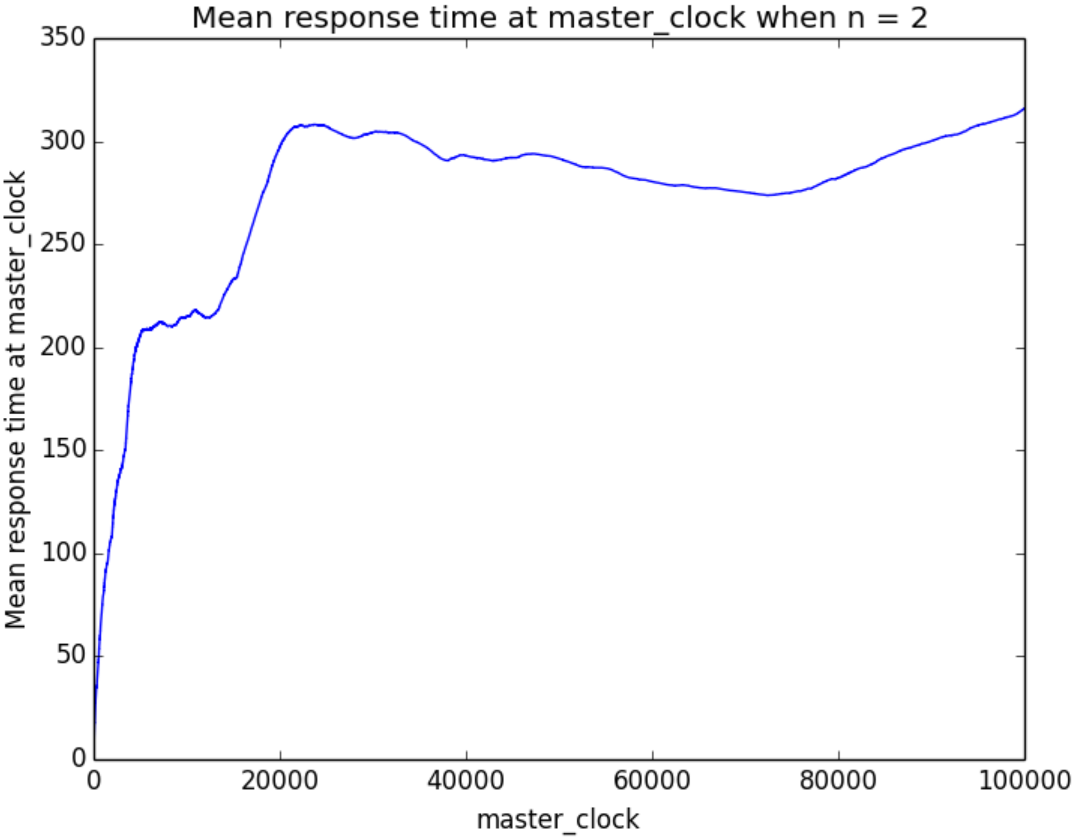
Note that the statistics below can be found in **Reproducibility** part.

n = 1:



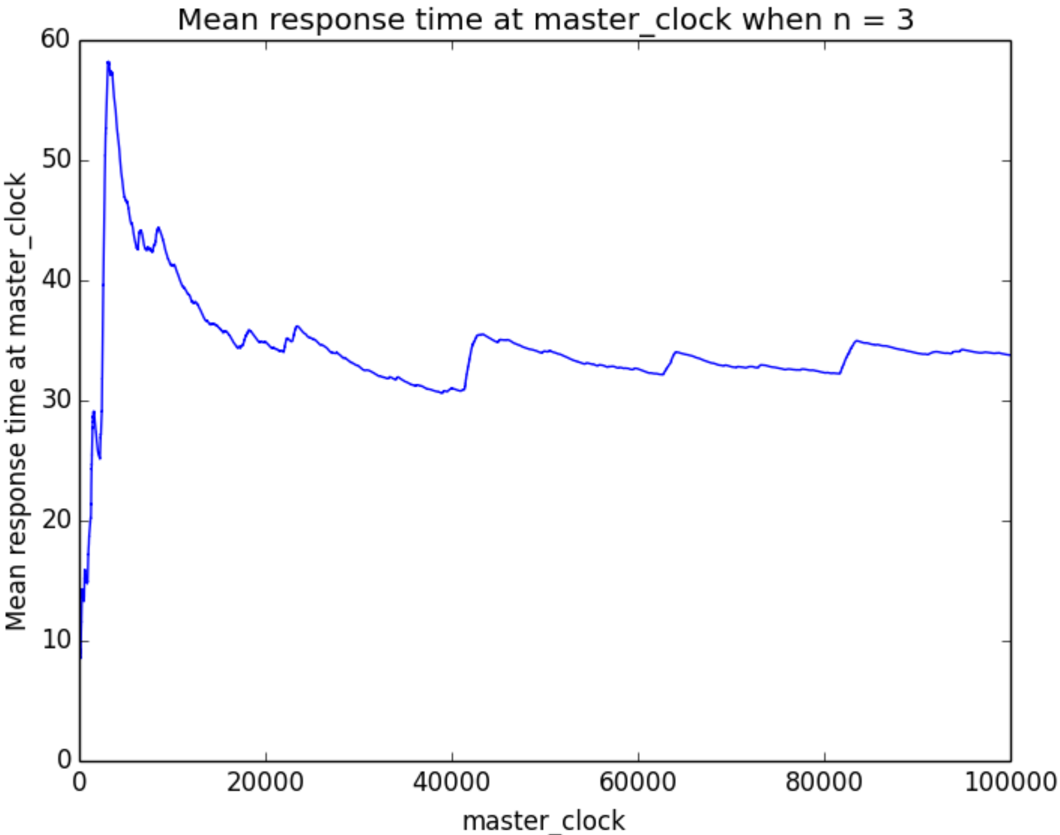
When n = 1, we can find that the mean response time and master_clock is linear function, which means if we increase the simulation time, our mean response time would also be increased.

n = 2:



From the picture, we can found that the mean response time is increased by master_clock, although it’s not the linear relation,I can’t find the there are any steady status. Therefore I think it’s meaningless to analyze it with some random numbers.

n = 3:



Similarly, from the picture, we can found that the state is steady after master_clock = 20000, therefore I choose master_clock = 20000 as the end of transient and master_clock = 40000, repeat the simulation 10 times(same seed) with different randoms.

Note that the statistics below can be found on **Reproducibility** part.

T1 = (931234.0 - 526903.0) / (30037.0 - 15103.0) = 27.0745
T2 = (806113.0 - 385348.0) / (30320.0 - 15135.0) = 27.7092
T3 = (913326.0 - 488096.0) / (29644.0 - 14857.0) = 28.7570
T4 = (10530940.0 - 9988659.0) / (30156.0 - 15088.0) = 35.9889
T5 = (983929.0 - 398199.0) / (30452.0 - 15197.0) = 38.3953
T6 = (1084422.0 - 569958.0) / (30096.0 - 15025.0) = 34.1360
T7 = (796766.0 - 374638.0) / (30210.0 - 15121.0) = 27.9758
T8 = (849863.0 - 360342.0) / (30493.0 - 15214.0) = 32.0388
T9 = (940848.0 - 462281.0) / (30052.0 - 14977.0) = 31.7457
T10 = (1214036.0 - 601767.0) / (30113.0 - 15134.0) = 40.8751

T= (T1 + T2 + … + T10) / 10
= (27.0745+ 27.7092+ 28.7570+ 35.9889+ 38.3953+
34.1360+ 27.9758+ 32.0388+ 31.7457+ 40.8751) / 10
= 32.4696

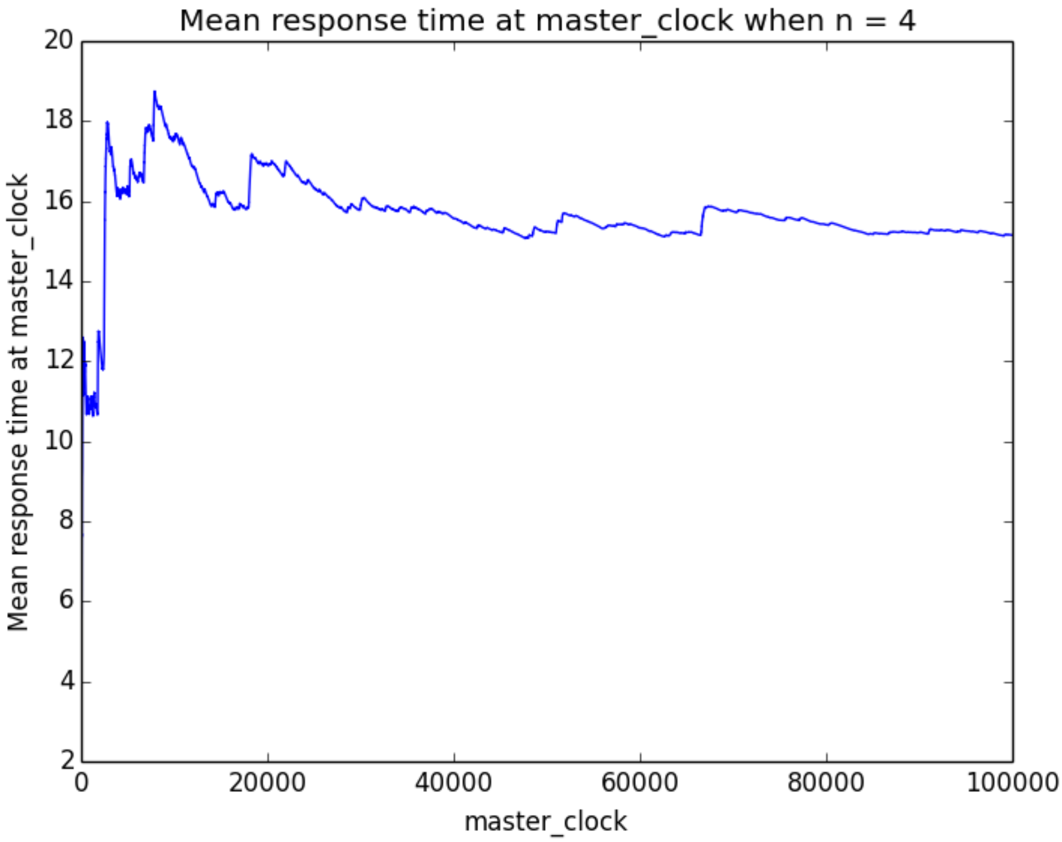
S = (((32.4696- 27.0745) ** 2 + (32.4696- 27.7092) ** 2 +(32.4696- 28.7570) ** 2 +(32.4696- 35.9889) ** 2 +
(32.4696- 38.3953) ** 2 +(32.4696- 34.1360) ** 2 +(32.4696- 27.9758) ** 2 +
(32.4696- 32.0388) ** 2 +(32.4696- 31.7457) ** 2 +(32.4696- 40.8751) ** 2) / 9) ** (1.0 / 2)
= 4.8002

The sample mean of 10 replications = 32.4696
The sample standard deviation of 10 replications is 0.3616
I want to compute the 95% confidence interval, a = 0.05
Since we did 10 independent experiments and want to 95% confidence interval, I use t(9, 0.975)

From the t-distribution table, the value of t(9, 0,975) = 2.262,therefore the 95% confidence interval:
[32.4696- 2.262* (4.8002/ ((10) ** (1.0 / 2))), 32.4696+ 2.262* (4.8002/ ((10) ** (1.0 / 2)))]
Therefore the confident interval is:

c

n = 4:



Similarly, from the picture, we can found that the state is steady after master_clock = 30000, therefore I choose master_clock = 30000 as the end of transient and master_clock = 40000, repeat the simulation 10 times(same seed) with different randoms.

Note that the statistics below can be found on **Reproducibility** part.

T1 = (467933.0 - 357701.0) / (30060.0 - 22546.0) = 14.6702
T2 = (422135.0 - 289771.0) / (30018.0 - 22425.0) = 17.4323
T3 = (464098.0 - 360041.0) / (29874.0 - 22406.0) = 13.9337
T4 = (446204.0 - 333687.0) / (30293.0 - 22790.0) = 14.9962
T5 = (642665.0 - 508875.0) / (30055.0 - 22508.0) = 14.7275
T6 = (475651.0 - 382785.0) / (30039.0 - 22585.0) = 12.4585
T7 = (513958.0 - 360689.0) / (30116.0 - 22637.0) = 15.2354
T8 = (577275.0 - 450945.0) / (30198.0- 22627.0) = 16.6860
T9 = (437927.0 - 335598.0) / (30261.0 - 22711.0) = 13.5535
T10 = (418482.0 - 323864.0) / (29778.0 - 22337.0) = 12.7157

T= (T1 + T2 + … + T10) / 10
= (14.6702+ 17.4323+ 13.9337+ 14.9962+ 14.7275+
12.4585+ 15.2354+ 16.6860+ 13.5535+ 12.7157) / 10
= 14.6409

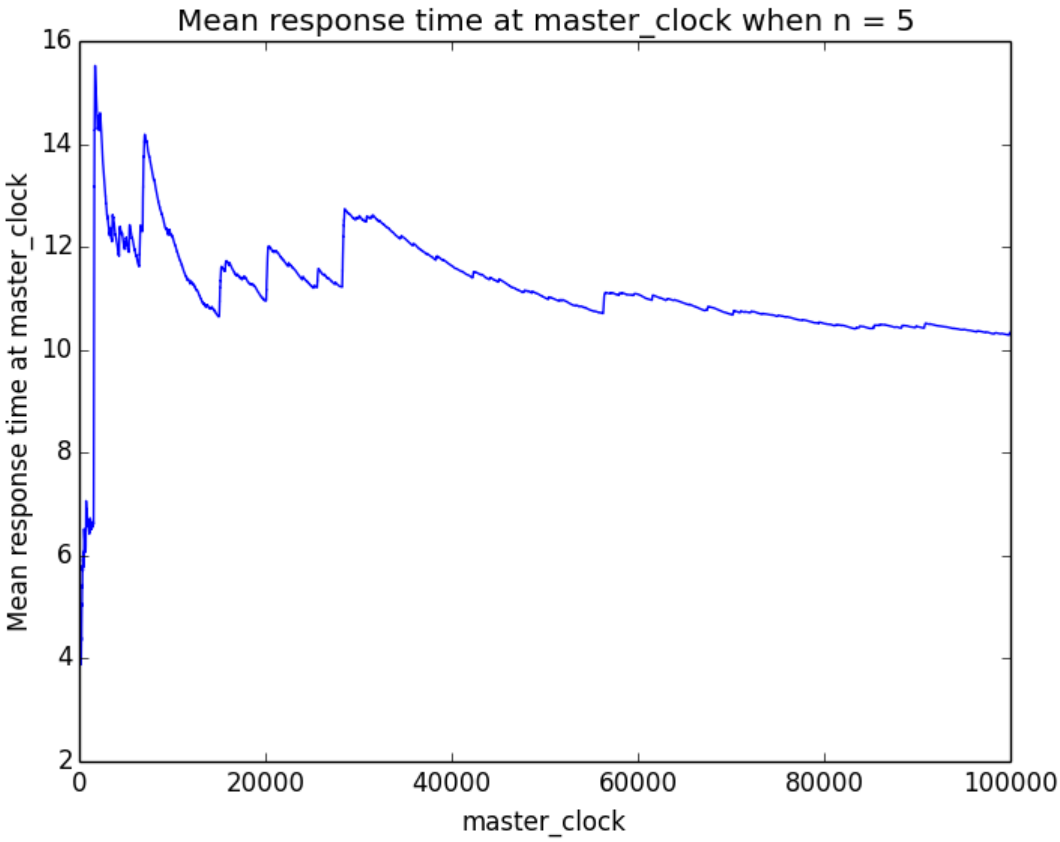
S = (((14.6409 - 14.6702) ** 2 +(14.6409 - 17.4323) ** 2 +(14.6409 - 13.9337) ** 2 +(14.6409 - 14.9962) ** 2 +
(14.6409 - 14.7275) ** 2 +(14.6409 - 12.4585) ** 2 +(14.6409 - 15.2354) ** 2 +
(14.6409 - 16.6860) ** 2 +(14.6409 - 13.5535) ** 2 +(14.6409 - 12.7157) ** 2) / 9) ** (1.0 / 2)
= 1.5851

The sample mean of 10 replications = 14.6409
The sample standard deviation of 15 replications is 1.5851
I want to compute the 95% confidence interval, a = 0.05
Since we did 10 independent experiments and want to 95% confidence interval, I use t(9, 0.975)

From the t-distribution table, the value of t(9, 0,975) = 2.262,therefore the 95% confidence interval:
[14.6409- 2.262* (1.5851/ ((10) ** (1.0 / 2))), 14.6409+ 2.262* (1.5851/ ((10) ** (1.0 / 2)))]
Therefore the confident interval is:

[13.5070, 15.7747]

n = 5:



Similarly, from the picture, we can found that the state is steady after master_clock = 60000, therefore I choose master_clock = 60000 as the end of transient and master_clock = 80000, repeat the simulation 10 times(same seed) with different randoms.

Note that the statistics below can be found on **Reproducibility** part

T1 = (634456.0 - 501383.0) / (60372.0 - 45262.0) = 8.8096
T2 = (773997.0 - 573399.0) / (60345.0 - 45245.0) = 13.2842
T3 = (593507.0 - 439876.0) / (60281.0 - 45011.0) = 10.0609
T4 = (706742.0 - 478489.0) / (60145.0 - 44979.0) = 13.0503
T5 = (665271.0 - 483276.0) / (60267.0 - 44981.0) = 11.9059
T6 = (679675.0 - 479531.0) / (59928.0 - 44837.0) = 13.2624
T7 = (990208.0 - 529661.0) / (60481.0 - 45408.0) = 15.5544
T8 = (631150.0 - 475605.0) / (60408.0 - 45355.0) = 10.3331
T9 = (567426.0 - 444042.0) / (60343.0 - 45249.0) = 13.2354
T10 = (568479.0 - 409444.0) / (60206.0 - 45124.0) = 12.5446

T= (T1 + T2 + … + T10) / 10
= (8.8096+ 13.2842+10.0609+ 13.0503+ 11.9059+ 13.2624+
15.5544+ 10.3331+ 13.2354+ 12.5446) / 10
= **12.2040**

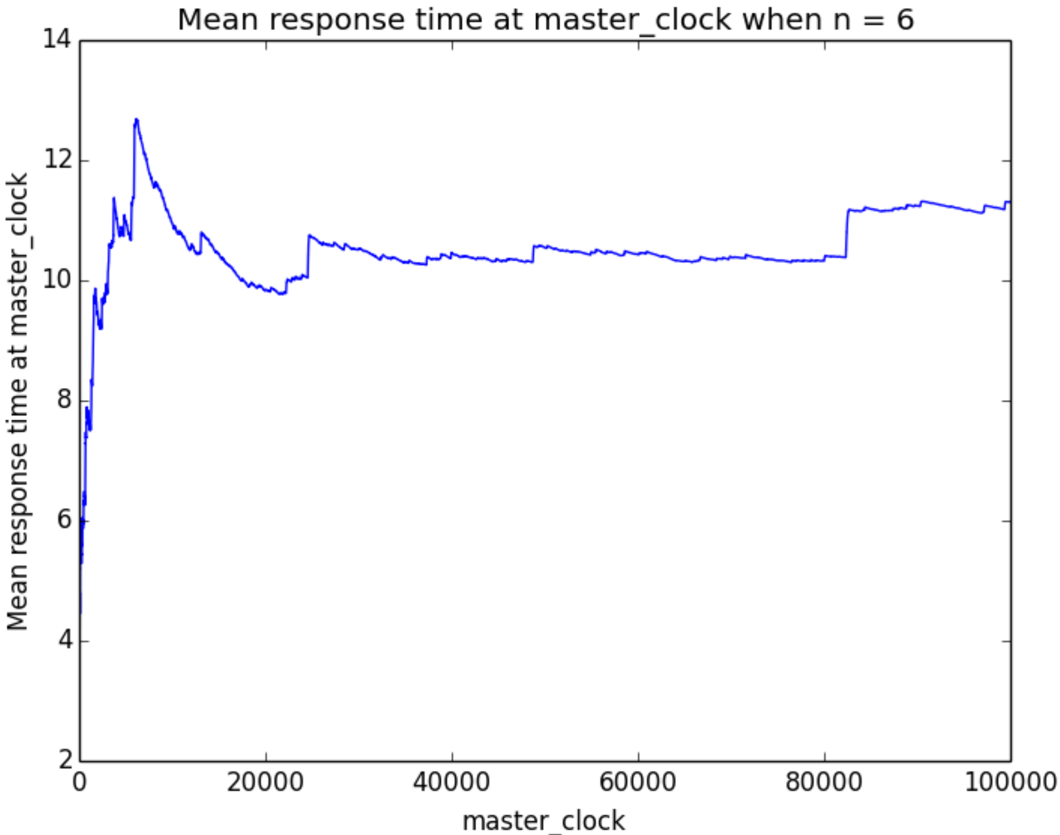
S = (((12.2040- 8.8096) ** 2 +(12.2040- 13.2842) ** 2 +(12.2040- 10.0609) ** 2 +(12.2040- 13.0503) ** 2 +
(12.2040- 11.9059) ** 2 +(12.2040- 13.2624) ** 2 +(12.2040- 15.5544) ** 2 +
(12.2040- 10.3331) ** 2 +(12.2040- 13.2354) ** 2 +(12.2040- 12.5446) ** 2) / 9) ** (1.0 / 2)
= 1.9751

The sample mean of 10 replications = **12.2040**
The sample standard deviation of 10 replications is 1.9751
I want to compute the 95% confidence interval, a = 0.05
Since we did 10 independent experiments and want to 95% confidence interval, I use t(9, 0.975)

From the t-distribution table, the value of t(9, 0,975) = 2.262,therefore the 95% confidence interval:
[12.2040 - 2.262* (1.9751/ ((10) ** (1.0 / 2))), 12.2040 + 2.262* (1.9751/ ((10) ** (1.0 / 2)))]
Therefore the confident interval is:

[10.7911, 11.6186]

n = 6:



Similarly, from the picture, we can found that the state is steady after master_clock = 40000, therefore I choose master_clock = 40000 as the end of transient and master_clock = 60000, repeat the simulation 10 times(same seed) with different randoms.

Note that the statistics below can be found on **Reproducibility** part

T1 = (371143.0 - 316148.0) / (45165.0 - 30218.0) = 13.6793
T2 = (520762.0 - 327159.0) / (45066.0 - 30139.0) = 12.9699
T3 = (471551.0 - 319881.0) / (44895.0 - 30161.0) = 10.2938
T4 = (520861.0 - 319163.0) / (45075.0 - 30162.0) = 13.5249
T5 = (607574.0 - 409353.0) / (45203.0 - 30078.0) = 13.1055
T6 = (462468.0 - 300452.0) / (44945.0 - 26846.0) = 11.1023
T7 = (534186.0 - 330832.0) / (44997.0 - 30009.0) = 13.5677
T8 = (474284.0 - 320946.0) / (45284.0 -30208.0) = 10.1710
T9 = (392332.0 - 327236.0) / (45472.0 - 30331.0) = 12.4532
T10 = (428863.0 - 282922.0) / (45405.0 - 30117.0) = 9.5461

T= (T1 + T2 + … + T10) / 10
= (13.6793+ 12.9699+ 10.2938+ 13.5249+ 13.1055+
11.1023+ 13.5677+ 10.1710+ 12.4532+ 9.5461) / 10
= **12.0413**

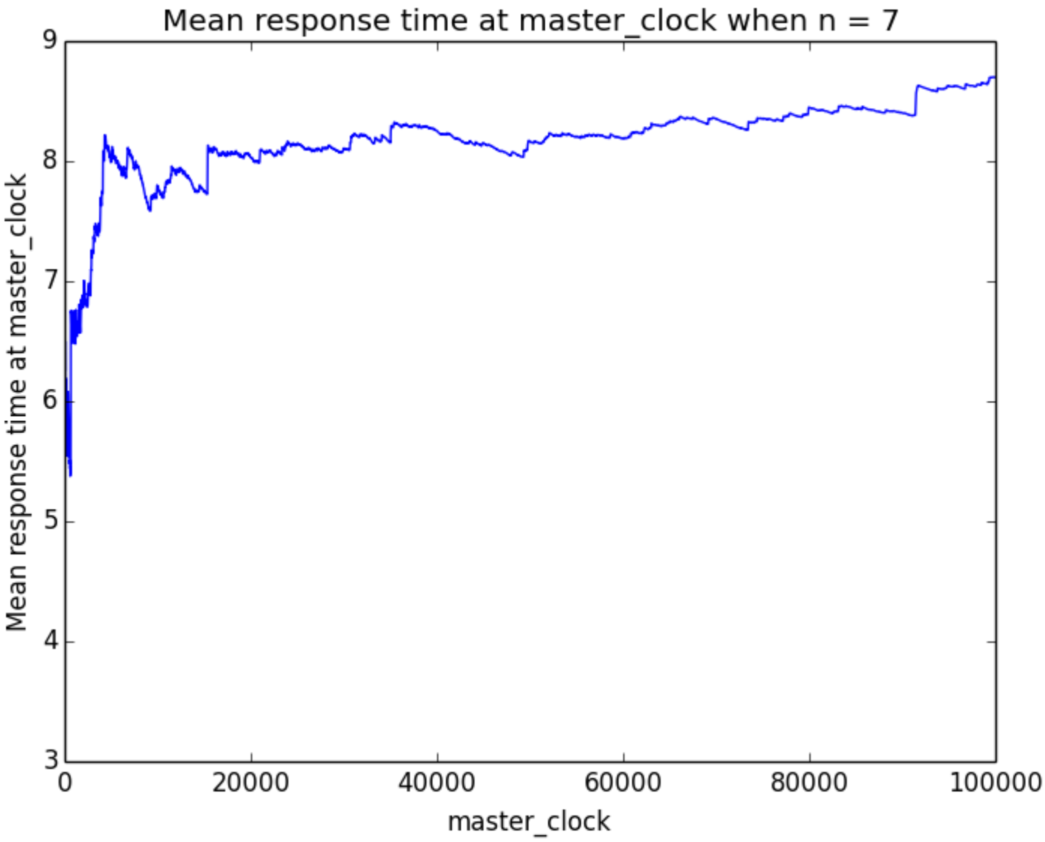
S = (((12.0413- 13.6793) ** 2 +(12.0413- 12.9699) ** 2 +(12.0413- 10.2938) ** 2 +(12.0413- 13.5249) ** 2 +
(12.0413- 13.1055) ** 2 +(12.0413- 11.1023) ** 2 +(12.0413- 13.5677) ** 2 +
(12.0413- 10.1710) ** 2 +(12.0413- 12.4532) ** 2 +(12.0413- 9.5461) ** 2) / 9) ** (1.0 / 2)
= **0.6938**

The sample mean of 10 replications = **12.0413**
The sample standard deviation of 10 replications is 0.6938
I want to compute the 95% confidence interval, a = 0.05
Since we did 10 independent experiments and want to 95% confidence interval, I use t(9, 0.975)

From the t-distribution table, the value of t(9, 0,975) = 2.262,therefore the 95% confidence interval:
[12.0413- 2.262* (0.6938/ ((10) ** (1.0 / 2))), 12.0413+ 2.262* (0.6938/ ((10) ** (1.0 / 2)))]
Therefore the confident interval is:

[11.8641, 12.3423]

n = 7:



Similarly, from the picture, we can found that the state is steady after master_clock = 40000, therefore I choose master_clock = 40000 as the end of transient and master_clock = 60000, repeat the simulation 10 times(same seed) with different randoms.

T1 = (371732.0 - 251194.0) / (45386.0 - 30445.0) = 8.0675
T2 = (428784.0 - 277421.0) / (45211.0 - 30209.0) = 9.5326
T3 = (375086.0 - 251241.0) / (45217.0 - 30050.0) = 8.1654
T4 = (527628.0 - 240691.0) / (44765.0 - 29737.0) = 8.3621
T5 = (423132.0 - 282388.0) / (45344.0 - 30264.0) = 9.3315
T6 = (408591.0 - 279212.0) / (44944.0 - 29916.0) = 8.6091
T7 = (434556.0 - 266860.0) / (45119.0 - 30099.0) = 11.1648
T8 = (416867.0 - 293716.0) / (45221.0 - 30250.0) = 8.2259
T9 = (392332.0 - 259396.0) / (45472.0 – 30386.0) = 8.8118
T10 = (369433.0 - 254715.0) / (45221.0 - 30180.0) = 7.6270

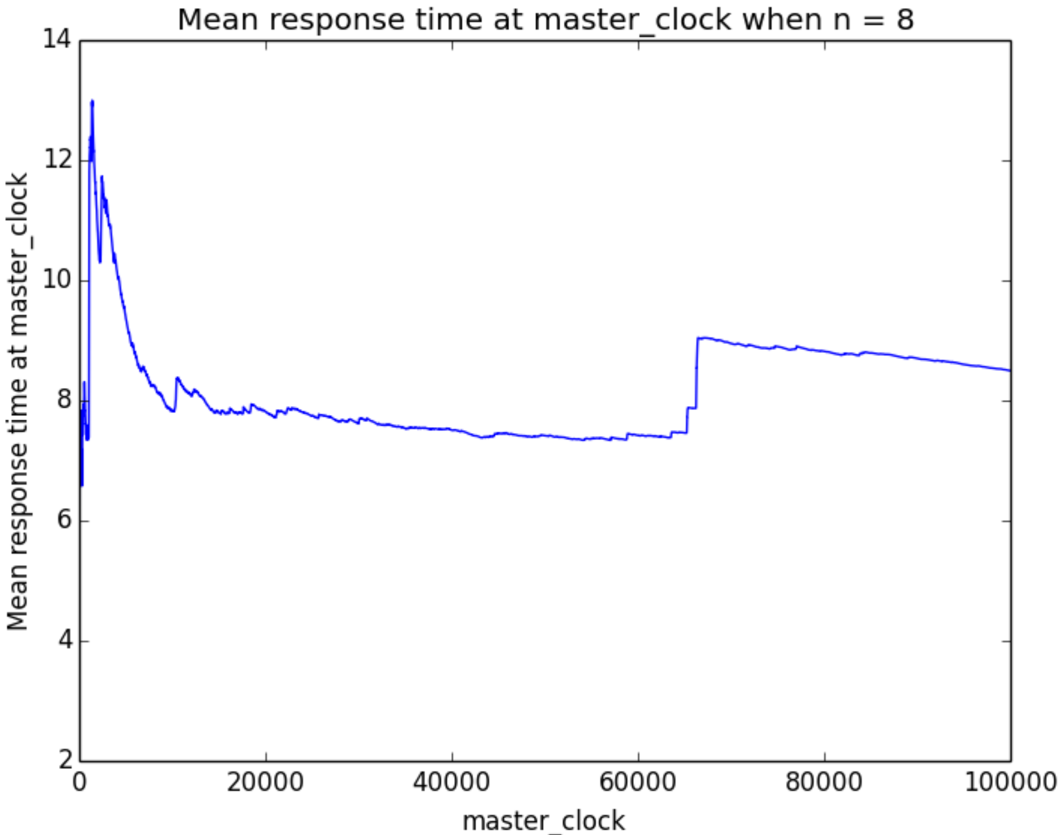
T= (T1 + T2 + … + T10) / 10
= (8.0675+ 9.5326+ 8.1654+ 8.3621+9.3315+ 8.6091
+ 11.1648+8.2259+ 8.8118+ 7.6270) / 10
= **8.7897**

S = (((8.7897- 8.0675) ** 2 + (8.7897- 9.5326) ** 2 +(8.7897- 8.1654) ** 2 +(8.7897- 8.3621) ** 2 +
(8.7897- 9.3315) ** 2 +(8.7897- 8.6091) ** 2 +(8.7897- 11.1648) ** 2 +
(8.7897- 8.2259) ** 2 +(8.7897- 8.8118) ** 2 +(8.7897- 7.6270) ** 2) / 9) ** (1.0 / 2)
= **1.0156**

The sample mean of 10 replications = **8.7897**
The sample standard deviation of 10 replications is **1.0156**
I want to compute the 95% confidence interval, a = 0.05
Since we did 10 independent experiments and want to 95% confidence interval, I use t(9, 0.975)

From the t-distribution table, the value of t(9, 0,975) = 2.262,therefore the 95% confidence interval:
[8.7897- 2.262* (1.0156/ ((10) ** (1.0 / 2))), 8.7897+ 2.262* (1.0156/ ((10) ** (1.0 / 2)))]
Therefore the confident interval is:
[8.0632,9.5161]

n = 8:



Similarly, from the picture, we can found that the state is steady after master_clock = 25000, therefore I choose master_clock = 25000 as the end of transient and master_clock = 30000, repeat the simulation 10 times(same seed) with different randoms.

T1 = (333456.0 - 225312.0) / (44939.0 - 30019.0) = 7.2482
T2 = (348308.0 - 238189.0) / (44932.0 - 29995.0) = 7.3722
T3 = (380395.0 - 242586.0) / (45504.0 - 30190.0) = 8.9988
T4 = (901271.0 - 673909.0) / (45432.0 - 30282.0) = 15.0079
T5 = (383019.0 - 248790.0) / (45445.0 - 30207.0) = 8.8088
T6 = (376157.0 - 249524.0) / (45222.0 - 30056.0) = 8.3497
T7 = (424167.0 - 267052.0) / (45150.0 - 30071.0) = 10.4194
T8 = (335617.0 - 221184.0) / (45281.0 - 30174.0) = 7.5748
T9 = (343079.0 - 237417.0) / (45263.0 - 30214.0) = 7.0211
T10 = (364088.0 - 225686.0) / (45301.0 - 30131.0) = 9.1234

T= (T1 + T2 + … + T10) / 10
= (7.2482+ 7.3722+ 8.9988+ 15.0079+ 8.8088+ 8.3497+
10.4194+7.5748+ 7.0211+ 9.1234) / 10
= 8.9924

S = (((8.9924 – 7.2482) ** 2 + (8.9924 – 7.3722) ** 2 + (8.9924 – 8.9988) ** 2 + (8.9924 – 15.0079) ** 2 +
(8.9924 – 8.8088) ** 2 + (8.9924 – 8.3497) ** 2 + (8.9924 – 10.4194) ** 2 +
(8.9924 – 7.5748) ** 2 + (8.9924 – 7.0211) ** 2 + (8.9924 – 9.1234) ** 2) / 9) ** (1.0 / 2)

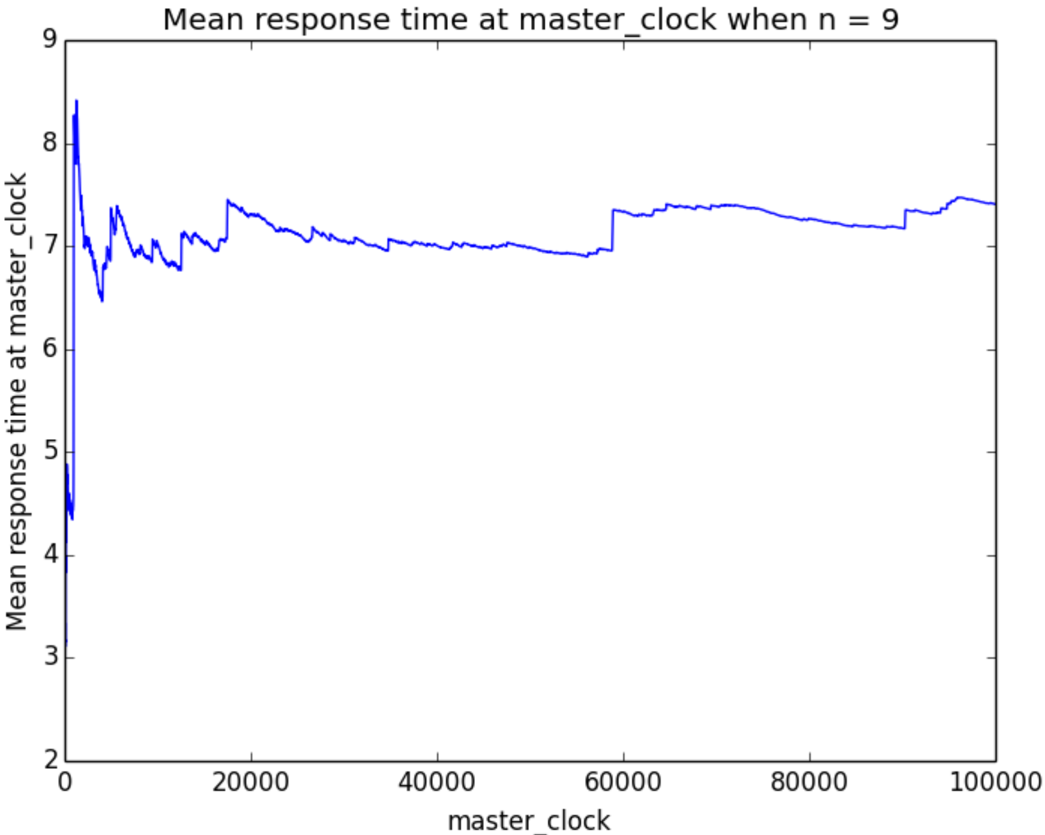
= 0.5435

The sample mean of 10 replications = 8.9924
The sample standard deviation of 10 replications is 0.5435
I want to compute the 95% confidence interval, a = 0.05
Since we did 10 independent experiments and want to 95% confidence interval, I use t(9, 0.975)

From the t-distribution table, the value of t(9, 0,975) = 2.262,therefore the 95% confidence interval:
[8.9924- 2.262* (0.5435/ ((10) ** (1.0 / 2))), 8.9924+ 2.262* (0.5435/ ((10) ** (1.0 / 2)))]
Therefore the confident interval is:

[8.6036,9.3811]

n = 9:



Similarly, from the picture, we can found that the state is steady after master_clock = 30000, therefore I choose master_clock = 30000 as the end of transient and master_clock = 40000, repeat the simulation 10 times(same seed) with different randoms.

Note that the statistics below can be found on **Reproducibility** part

T1 = (221448.0 - 159638.0) / (30194.0 - 22642.0) = 8.1845
T2 = (228329.0 - 161990.0) / (30219.0 - 22713.0) = 8.8381
T3 = (216065.0 – 158890.0) / (30029.0 – 22499.0) = 7.5929
T4 = (214482.0 - 164958.0) / (30243.0 – 22715.0) = 6.5786
T5 = (223717.0 - 172594.0) / (30278.0 - 22789.0) = 6.8264
T6 = (193988.0 - 142965.0) / (29785.0 - 22336.0) = 6.8496
T7 = (221553.0 - 152890.0) / (30296.0 - 22736.0) = 9.0824
T8 = (214945.0 -158686.0) / (30116.0 - 22695.0) = 7.5810
T9 = (194340.0 - 142383.0) / (30107.0 - 22636.0) = 6.9544
T10 = (210487.0 - 160916.0) / (30368.0 - 22740.0) = 6.4985

T= (T1 + T2 + … + T10) / 10
= (8.1845+ 8.8381+ 7.5929+ 6.5786+ 6.8264+
6.8496+ 9.0824+ 7.5810+ 6.9544+ 6.4985) / 10
= 7.4986

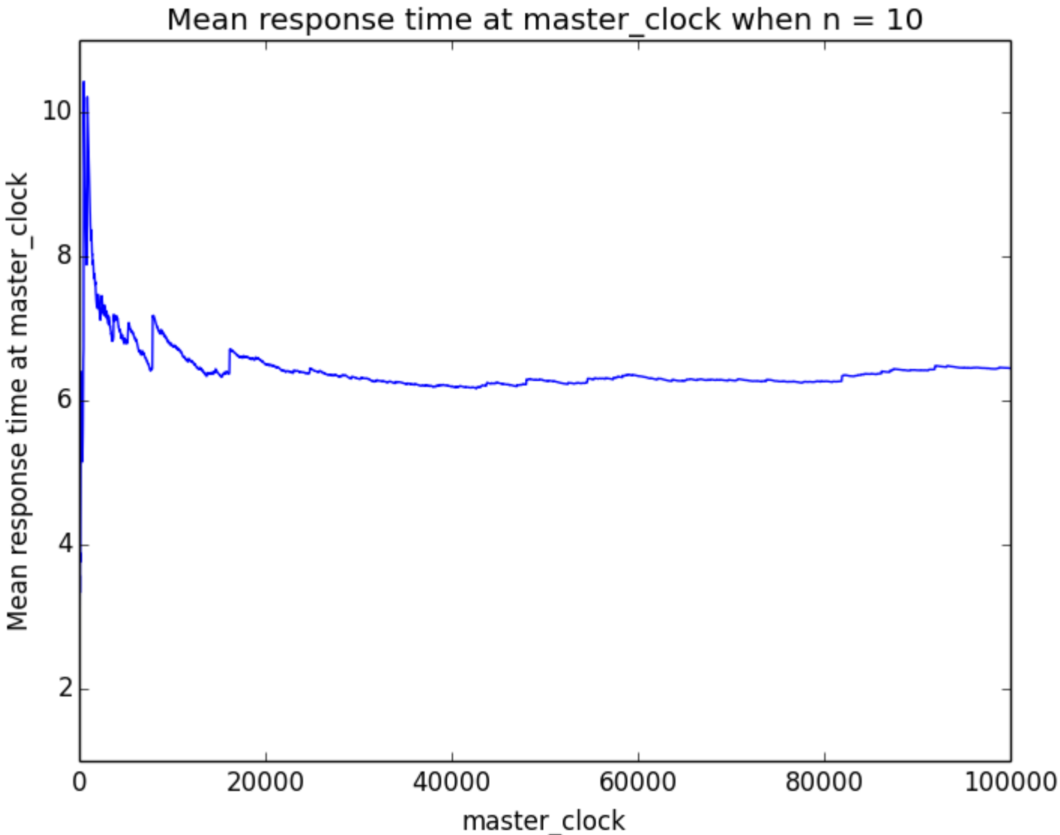
S = (((7.4986-8.1845) ** 2 + (7.4986-8.8381) ** 2 +(7.4986-7.5929) ** 2 +(7.4986-6.5786) ** 2 +
(7.4986-6.8264) ** 2 +(7.4986-6.8496) ** 2 +(7.4986-9.0824) ** 2 +
(7.4986-7.5810) ** 2 +(7.4986-6.9544) ** 2 +(7.4986-6.4985) ** 2) / 9) ** (1.0 / 2)
= 0.9312

The sample mean of 10 replications = 7.4986
The sample standard deviation of 10 replications is 0.9312
I want to compute the 95% confidence interval, a = 0.05
Since we did 10 independent experiments and want to 95% confidence interval, I use t(9, 0.975)

From the t-distribution table, the value of t(9, 0,975) = 2.262,therefore the 95% confidence interval:
Therefore the confident interval is:
[7.4986- 2.262* (0.9312/ ((10) ** (1.0 / 2))), 7.4986+ 2.262* (0.9312/ ((10) ** (1.0 / 2)))]

[6.8325, 8.1646]

n = 10:



Similarly, from the picture, we can found that the state is steady after master_clock = 30000, therefore I choose master_clock = 30000 as the end of transient and master_clock =40000, repeat the simulation 10 times(same seed) with different randoms.

T1 = (185553.0 - 141582.0) / (29924.0 – 22460.0) = 5.8910
T2 = (209443.0 - 157038.0) / (29990.0 – 22342.0) = 6.8521
T3 = (176713.0 -127185.0) / (29936.0 – 22471.0) = 6.6346
T4 = (205954.0 - 156822.0) / (30227.0 - 22635) = 6.4715
T5 = (197036.0 - 145059.0) / (29910.0 – 22315.0) = 6.8435
T6 = (207584.0 - 154115.0) / (29951.0 - 22377.0) = 7.0595
T7 = (241256.0 - 189066.0) / (30135.0 – 22574.0) = 6.9025
T8 = (227038.0 - 175470.0) / (30500.0 - 22997.0) = 6.8729
T9 = (192490.0 - 144014.0) / (30149.0 - 22609.0) = 6.4291
T10 = (192596.0 - 145178.0) / (30220.0 - 22602.0) = 6.2244

T= (T1 + T2 + … + T10) / 10
= (5.8910 + 6.8521 +6.6346+ 6.4715+ 6.8435+
7.0595+ 6.9025+ 6.8729+ 6.4291+6.2244) / 10
= **6.6181**

S = (((6.6181 - 5.8910) ** 2 + (6.6181 - 6.8521) ** 2 +(6.6181 - 6.6346) ** 2 +(6.6181 - 6.4715) ** 2 +
(6.6181 - 6.8435) ** 2 +(6.6181 - 7.0595) ** 2 +(6.6181 - 6.9025) ** 2 +
(6.6181 - 6.8729) ** 2 +(6.6181 - 6.4291) ** 2 +(6.6181 - 6.2244) ** 2) / 9) ** (1.0 / 2)

= **0.3632**

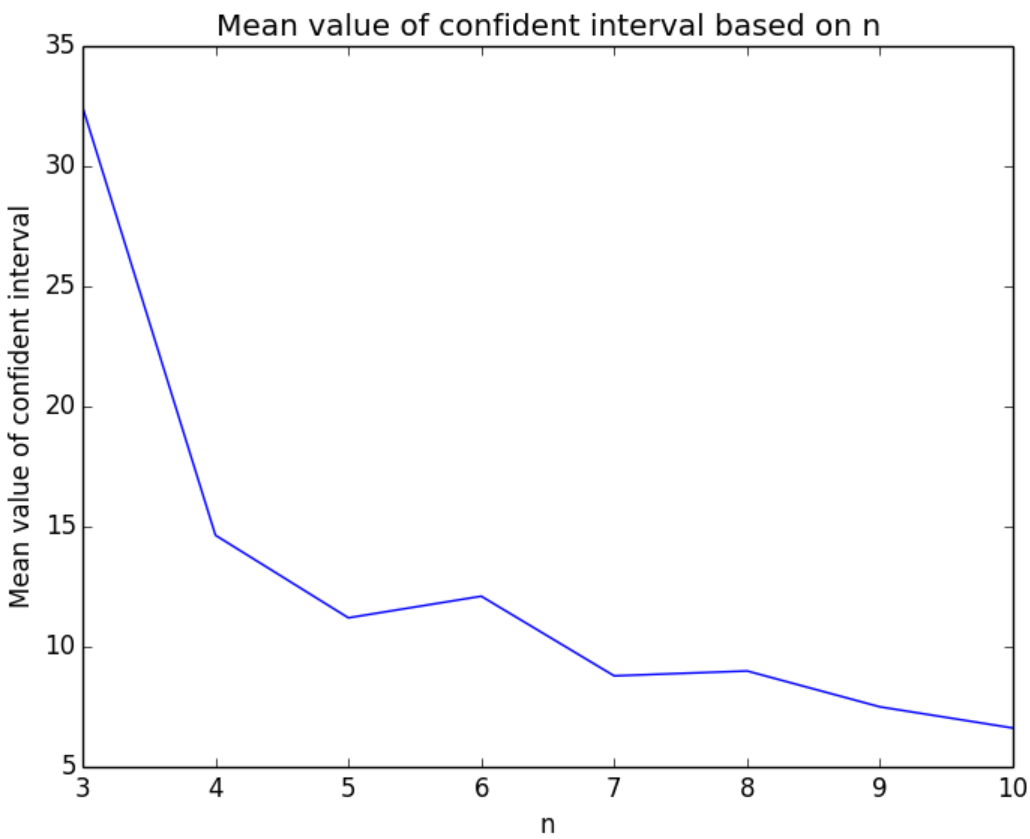
The sample mean of 10 replications = 6.6181
The sample standard deviation of 10 replications is 0.3632
I want to compute the 95% confidence interval, a = 0.05
Since we did 10 independent experiments and want to 95% confidence interval, I use t(9, 0.975)

From the t-distribution table, the value of t(9, 0,975) = 2.262,therefore the 95% confidence interval:
Therefore the confident interval is:
[6.6181- 2.262* (0.3632/ ((10) ** (1.0 / 2))), 6.6181+ 2.262* (0.3632/ ((10) ** (1.0 / 2)))]

[6.3583, 6.8778]

Summary of Statistical Analysis:

n	confident interval	mean value of confident interval
1	**Inf**	**Inf**
2	**Inf**	**Inf**
3	**[29.0359,35.9032]**	**32.4695**
4	**[13.5070, 15.7747]**	**14.6408**
5	**[10.7911, 11.6186]**	**11.2048**
6	**[11.8641, 12.3423]**	**12.1032**
7	**[8.0632,9.5161]**	**8.7896**
8	**[8.6036,9.3811]**	**8.9923**
9	**[6.8325, 8.1646]**	**7.4985**
10	**[6.3583, 6.8778]**	**6.6185**



Therefore in order to get the smallest response time**, the n should be equal 10.**

Reproducibility:

Note that when use my program to simulate the system, the format should be like that:

$ python project.py [n] [Tend] [seed]

For example:

If we want to simulate the T = 20000, n = 3, seed = 3

$ python project.py 3 20000 3

Note that, when n =1,2, there are no data, NaN.

The raw data below are T and N of of different n (10, 9, 8) based on different seed.

n = 10	T = 30000	T = 40000	n = 9	T = 30000	T = 40000	n = 8	T = 40000	T = 60000
seed = 1	T: 141582 N: 22460 Response: 6.3037	T: 185553 N: 29924 Response: 6.2008		T: 159638 N: 22642 Response: 6.2008	T: 221448 N: 30194 Response: 7.0029		T: 225312 N: 30019 Response: 7.5056	T: 333456 N: 44939 Response: 7.4202
seed = 2	T: 157038 N: 22342 Response: 7.0288	T: 209443 N: 29990 Response: 6.9837		T: 161990 N: 22713 Response: 7.1320	T: 228329 N: 30219 Response: 7.5558		T: 238189 N: 29995 Response: 7.9409	T: 348308 N: 44932 Response: 7.7519
seed = 3	T: 127185 N: 22471 Response: 5.6599	T: 176713 N: 29936 Response: 5.9030		T: 158890 N: 22499 Response: 7.0621	T: 216065 N: 30029 Response: 7.1952		T: 242586 N: 30190 Response: 8.0353	T: 380395 N: 45504 Response: 8.3595
seed = 4	T: 156822 N: 22635 Response: 6.9283	T: 205954 N: 30227 Response: 6.8136		T: 164958 N: 22715 Response: 7.2621	T: 214482 N: 30243 Response: 7.0919		T: 673909 N: 30282 Response: 7.5056	T: 901271 N: 45432 Response: 19.8378
seed = 5	T: 145059 N: 22315 Response: 6.5005	T: 197036 N: 29910 Response: 6.5876		T: 172594 N: 22789 Response: 7.5735	T: 223717 N: 30278 Response: 7.3887		T: 248790 N: 30207 Response: 8.2361	T: 383019 N: 45445 Response: 8.4281
seed = 6	T: 154115 N: 22377 Response: 6.8872	T: 207584 N: 29951 Response: 6.9307		T: 142965 N: 22336 Response: 6.4006	T: 193988 N: 29785 Response: 6.5129		T: 249524 N: 30056 Response: 8.3019	T: 376157 N: 45222 Response: 8.3180
seed = 7	T: 189066 N: 22574 Response: 8.3753	T: 241256 N: 30135 Response: 8.0058		T: 152890 N: 22736 Response: 6.7245	T: 221553 N: 30296 Response: 7.3129		T: 267052 N: 30071 Response: 8.8798	T: 424167 N: 45150 Response: 9.3946
seed = 8	T: 175470 N: 22997 Response: 7.6301	T: 227038 N: 30500 Response: 7.4438		T: 158686 N: 22695 Response: 6.9921	T: 214945 N: 30116 Response: 7.1372		T: 221184 N: 30174 Response: 7.3303	T: 335617 N: 45281 Response: 7.4118
seed = 9	T: 144014 N: 22609 Response: 6.3698	T: 192490 N: 30149 Response: 6.3846		T: 142383 N: 22636 Response: 6.2901	T: 194340 N: 30107 Response: 6.4549		T: 237417 N: 30214 Response: 7.8578	T: 343079 N: 45263 Response: 7.5796
seed = 10	T: 145178 N: 22602 Response: 6.4232	T: 192596 N: 30220 Response: 6.3731		T: 160916 N: 22740 Response: 6.0763	T: 210487 N: 30368 Response: 6.9312		T: 225686 N: 30131 Response: 7.4901	T: 364088 N: 45301 Response: 8.0371

The raw data below are T and N of of different n (7, 6, 5,4) based on different seed(1-10).

n = 7	T = 40000	T = 60000	n = 6	T= 40000	T = 60000	n = 5	T = 60000	T = 80000	n = 4	T = 30000	T = 40000
	T: 251194 N: 30445 Response: 8.2507	T: 371732 N: 45386 Response: 8.1904		T: 316148 N: 30218 Response: 10.4622	T: 371143 N: 45165 Response: 10.4315		T: 501383 N: 45262 Response: 11.0773	T: 634456 N: 60372 Response: 10.5091		T: 357701 N: 22546 Response: 15.8654	T: 467933 N: 30060 Response: 15.5666
	T: 277421 N: 30209 Response: 9.1834	T: 428784 N: 45211 Response: 9.3976		T: 327159 N: 30139 Response: 10.8550	T: 520762 N: 45066 Response: 11.5555		T: 573399 N: 45245 Response: 12.6732	T: 773997 N: 60345 Response: 12.8262		T: 289771 N: 22425 Response: 12.9218	T: 422135 N: 30018 Response: 14.0627
	T: 251241 N: 30050 Response: 8.3607	T: 375086 N: 45217 Response: 8.2951		T: 319881 N: 30161 Response: 10.605	T: 471551 N: 44895 Response: 10.5034		T: 439876 N: 45011 Response: 9.7726	T: 593507 N: 60281 Response: 9.8456		T: 360041 N: 22406 Response: 16.0689	T: 464098 N: 29874 Response: 15.5352
	T: 240691 N: 29737 Response: 8.0940	T: 527628 N: 44765 Response: 11.7866		T: 319163 N: 30162 Response: 10.5816	T: 520861 N: 45075 Response: 11.5554		T: 478489 N: 44979 Response: 10.6380	T: 706742 N: 60145 Response: 11.7506		T: 333687 N: 22790 Response: 14.6418	T: 446204 N: 30293 Response: 14.7296
	T: 282388 N: 30264 Response: 9.3308	T: 423132 N: 45344 Response: 9.3316		T: 409353 N: 30078 Response: 13.6097	T: 607574 N: 45203 Response: 13.4410		T: 483276 N: 44981 Response: 10.7440	T: 665271 N: 60267 Response:11.0387		T: 508875 N: 22508 Response: 22.6086	T: 642665 N: 30055 Response:21.3829
	T: 279212 N: 29916 Response: 9.3332	T: 408591 N: 44944 Response: 9.0911		T: 300452 N: 26846 Response: 10.0667	T: 462468 N: 44945 Response: 10.2896		T: 479531 N: 44837 Response: 10.6950	T: 679675 N: 59928 Response: 11.3415		T: 382785 N: 22585 Response: 16.9486	T: 475651 N: 30039 Response: 15.8344
	T: 266860 N: 30099 Response: 8.8611	T: 434556 N: 45119 Response: 9.6313		T: 330832 N: 30009 Response: 11.0244	T: 534186 N: 44997 Response: 11.8715		T: 529661 N: 45408 Response: 11.6645	T: 990208 N: 60481 Response: 16.3722		T: 360689 N: 22637 Response: 15.9336	T: 513958 N: 30116 Response: 27.0274
	T: 293716 N: 30250 Response: 9.7096	T: 416867 N: 45221 Response: 9.2184		T: 320946 N: 30208 Response: 10.6245	T: 474284 N: 45284 Response: 10.4735		T: 475605 N: 45355 Response: 10.4862	T: 631150 N: 60408 Response: 10.4481		T: 450945 N: 22627 Response: 19.9295	T: 577275 N: 30198 Response: 19.1163
	T: 259396 N: 30386 Response: 8.5367	T: 392332 N: 45472 Response: 8.6280		T: 327236 N: 30331 Response: 10.7888	T: 392332 N: 45472 Response: 8.6280		T: 444042 N: 45249 Response: 9.8133	T: 567426 N: 60343 Response: 9.4033		T: 335598 N: 22711 Response: 14.7769	T: 437927 N: 30261 Response: 14.4716
	T: 254715 N: 30180 Response: 8.4398	T: 369433 N: 45221 Response: 8.1695		T: 282922 N: 30117 Response: 9.3940	T: 428863 N: 45405 Response: 9.4452		T: 409444 N: 45124 Response: 9.0737	T: 568479 N: 60206 Response: 9.4422		T: 323864 N: 22337 Response: 14.4990	T: 418482 N: 29778 Response: 14.0534

The raw data below are T and N of n = 3 based on different seed(1-10).

n = 3	T = 20000	T = 40000
	T: 526903 N: 15103 Response: 34.8873	T: 931234 N: 30037 Response: 31.0029
	T: 385348 N: 15135 Response: 25.4607	T: 806113 N: 30320 Response: 26.5868
	T: 488096 N: 14857 Response: 32.8529	T: 913326 N: 29644 Response: 30.9140
	T: 9988659 N: 15088 Response: 662.0267	T: 10530940 N: 30156 Response: 349.2154
	T: 398199 N: 15197 Response: 26.2025	T: 983929 N: 30452 Response: 32.3108
	T: 569958 N: 15025 Response: 37.9340	T: 1084422 N: 30096 Response: 36.0321
	T: 374638 N: 15121 Response: 24.7760	T: 796766 N: 30210 Response: 26.3742
	T: 360342 N: 15214 Response: 23.6849	T: 849863 N: 30493 Response: 27.8707
	T: 462281 N: 14977 Response: 30.8660	T: 940848 N: 30052 Response: 31.3073
	T: 601767 N: 15134 Response: 39.7626	T: 1214036 N: 30113 Response: 40.3160

Optimization:

I think we can optimize the part that the processor randomly chooses a number of servers to process the sub-tasks.

I think we can add a module just like system register that record the server status and queue length of each server. After the processor served the task, then it should check the system register to find is there any idle servers, if yes then check whether the number of idle servers larger than n, if yes just choose the n idle servers randomly among all the idle servers, otherwise select all the idle servers and put the remaining subtask to the other servers with smaller queue length which can get from system register. Otherwise, find the servers with smaller queue length among all the busy servers and sent these subtasks to them.