COMP9334

Project, Session 1 2017

Getting the best response time out of your power bill

Elliot Griffiths

z3332776

Generating Probability Distributions

To generate the inter-arrival times of the requests arriving at the high-speed router we take the product of two random numbers, a_{1k} and a_{2k}. Such that the inter-arrival time is $a_k = a_{1k}a_{2k}$.

To generate the exponentially distributed values a_{1k} we use the inverse transform method. We generate a random number u, which is uniformly distributed in (0,1) and compute the number $F^{-1}(u)$ where $F(x)$ is the Cumulative Density Function (CDF)

We know the CDF for an exponential distribution is $F(x) = 1 - e^{-\lambda x}$
Thus, $u = 1 - e^{-\lambda x}$ and therefore our generated number is $x = -\frac{\log(1-u)}{\lambda}$
(where $\lambda = 7.2$)

To generate the uniformly distributed values a_{2k} in the interval [0.75, 1.17] we simply generate a number between 0 and 1.17 - 0.75 then we add that number to 0.75 giving us the required range.

Finally generating the service time t is slightly more involved. We are given the probability density function g(t).

Where Probability Density Function of service time is:

$$g(t) = \begin{cases} 0 & \text{for } \alpha_1 \leq t \\ \frac{\gamma}{t^\beta} & \text{for } \alpha_1 \leq t \leq \alpha_2 \\ 0 & \text{for } t \geq \alpha_2 \end{cases}$$

Where, $\gamma = \frac{1-\beta}{\alpha_2^{1-\beta} - \alpha_1^{1-\beta}}$

For generating random numbers with this probability distribution, we first need to find the Cumulative Density Function G(t). To do this we integrate over the curve $\frac{\gamma}{t^\beta}$ from α_1 to t

This gives the CDF(t) = G(t)

$$G(t) = \begin{cases} 0 & \text{for } \alpha_1 \leq t \\ \frac{t^{1-\beta} - \alpha_1^{1-\beta}}{(\alpha_2^{1-\beta} - \alpha_1^{1-\beta})} & \text{for } \alpha_1 \leq t \leq \alpha_2 \\ 1 & \text{for } t \geq \alpha_2 \end{cases}$$

We can see that if $t = \alpha_2$ then the cumulative probability of getting a value of 0.98 or less is 100% thus it is correct.

$$u = \frac{t^{1-\beta} - \alpha_1^{1-\beta}}{\alpha_2^{1-\beta} - \alpha_1^{1-\beta}}$$

Now we need to arrange the function so we can find the 't-intercepts' from given 'u' values

Therefore our values for t given u between 0 and 1 are

$$t = \left(u * \left(\alpha_2^{1-\beta} - \alpha_1^{1-\beta}\right) + \alpha_1^{1-\beta}\right)^{\frac{1}{1-\beta}}$$

Simulating the PS Server

The simulation is written in perl and stored in file ps_server_sim.pl. To run the simulation two arguments must be given after the program name. The first is the simulation time and the second is the number of servers. For example:

```
Elliots-MacBook-Pro:Project Elliot$ ./ps_server_sim.pl 10000 10

7495 jobs completed by 1 server so we have 74950 jobs completed by
10 servers,
in 4064.88435799293 time therefore mean response time is
0.542346145162499
```

Verifying the Simulation

To verify the code I have wrote an alternative versions of the simulation with the random number generation removed and replaced with two lists containing the Arrival time of the request and the service time of each request. These values are popped from the lists when required. For simplicity the frequency has been set to 1GHz

CODE: ps_server_sim_ver.pl

SERVERS: 1 at 1GHz

Job Number	Request Arrival Time	Service Time
1	1	2.1
2	2	3.3
3	3	1.1
4	5	0.5

Expected Mean Response Time: 3.6

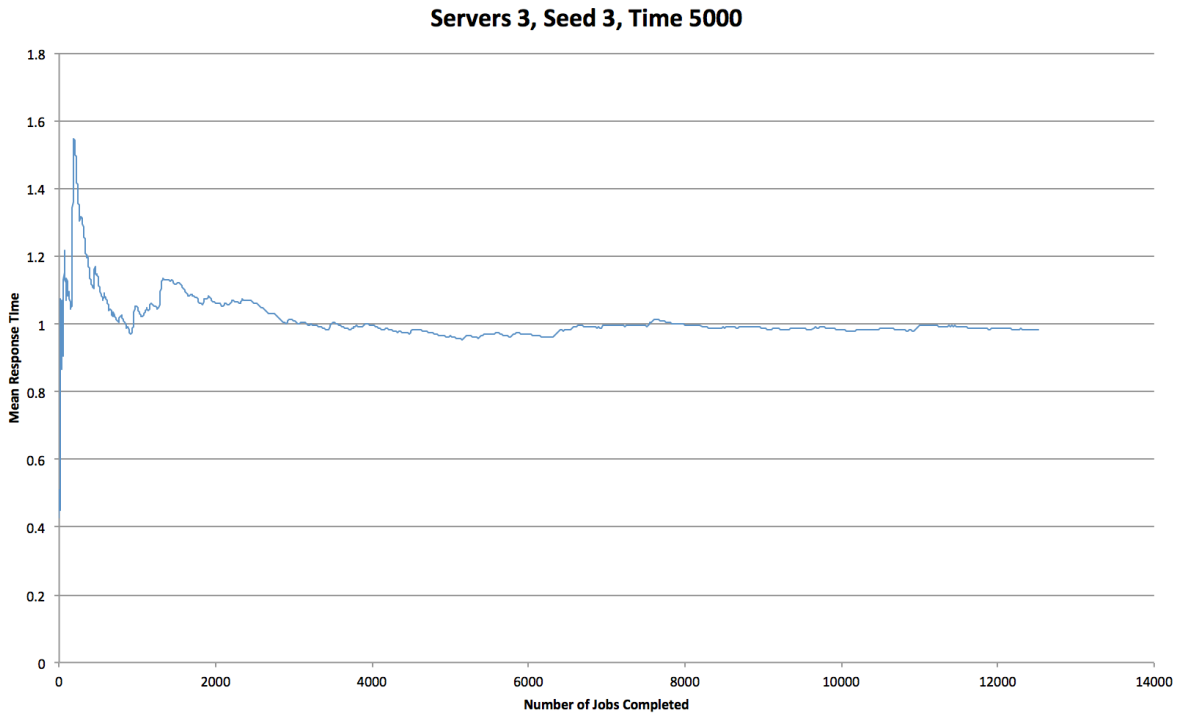
Mean Response Time Simulation:

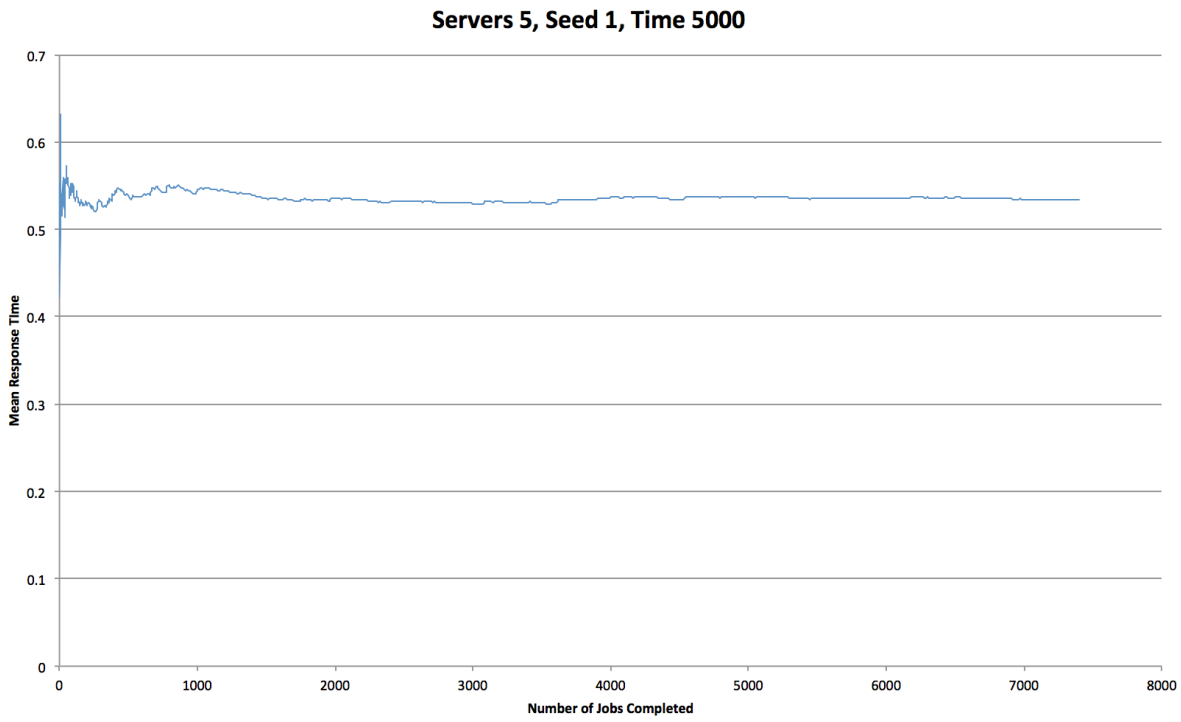
```
Elliots-MacBook-Pro:Validating Elliot$ ./ps_server_sim_ver.pl

4 jobs completed by 1 server(s) at 1GHz frequency in 14.4 time
therefore mean response time is 3.6
```

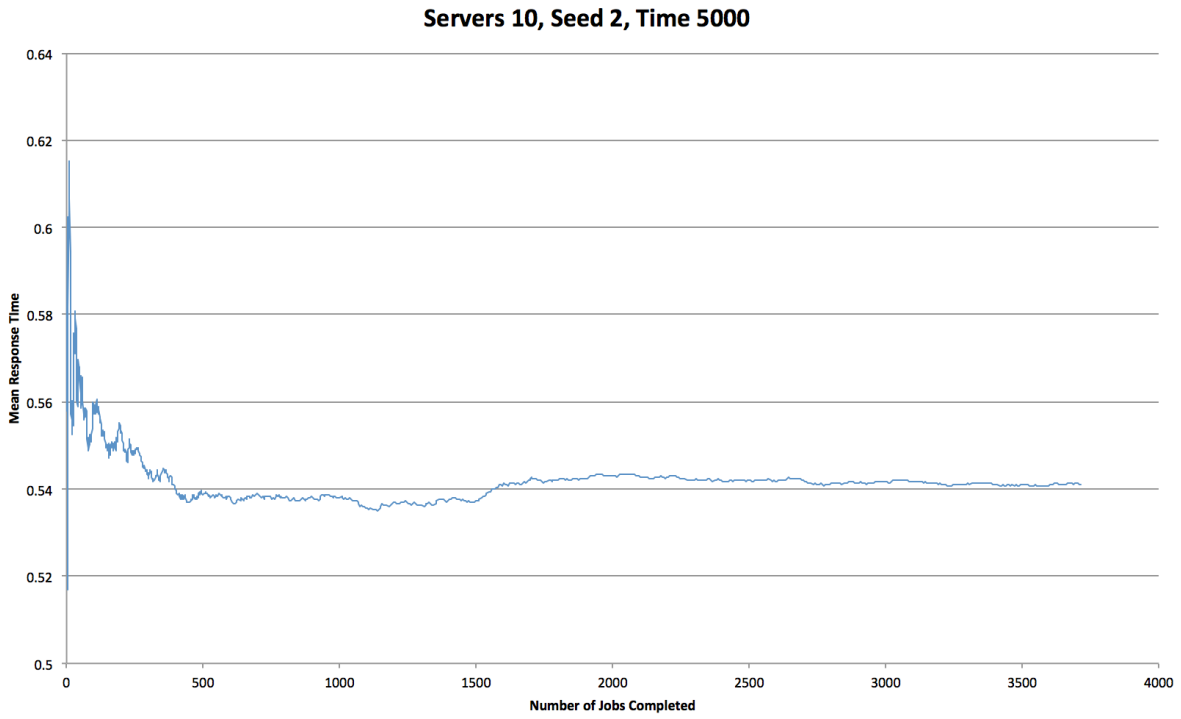
Choosing Simulation Parameters

SIM TIME

First we want to make sure we have simulated enough jobs such that the mean response time has reached steady state behaviour and does not show any transient behaviour. We do this by visually inspecting the mean response time as the number of jobs completed increases.


Servers 3, Seed 3, Time 5000
Mean Response Time
Number of Jobs Completed
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
0
2000
4000
6000
8000
10000
12000
14000

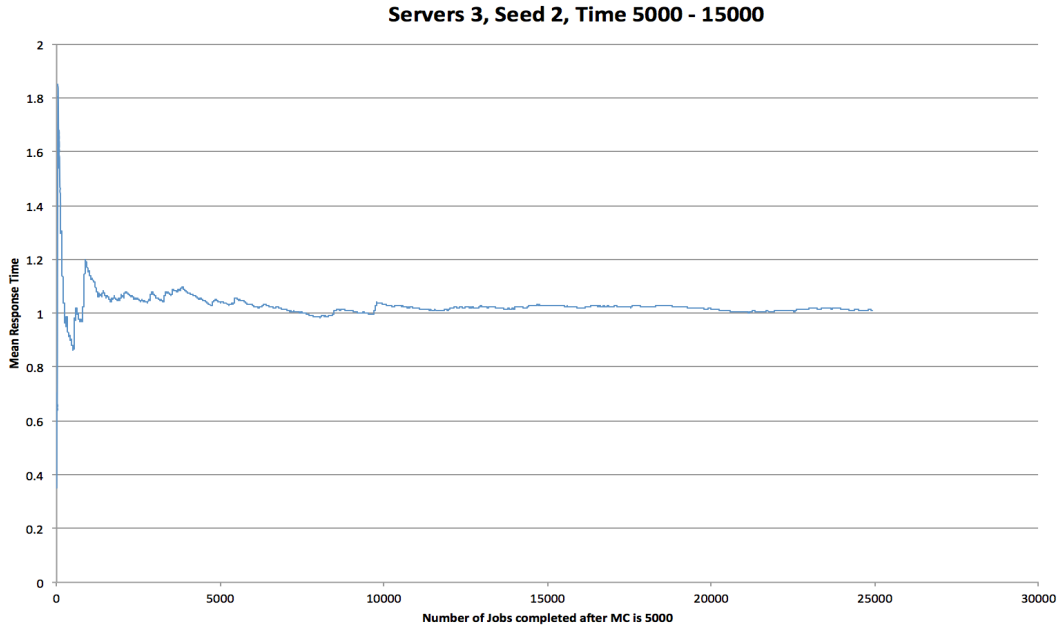

Servers 5, Seed 1, Time 5000
Mean Response Time
Number of Jobs Completed
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
1000
2000
3000
4000
5000
6000
7000
8000

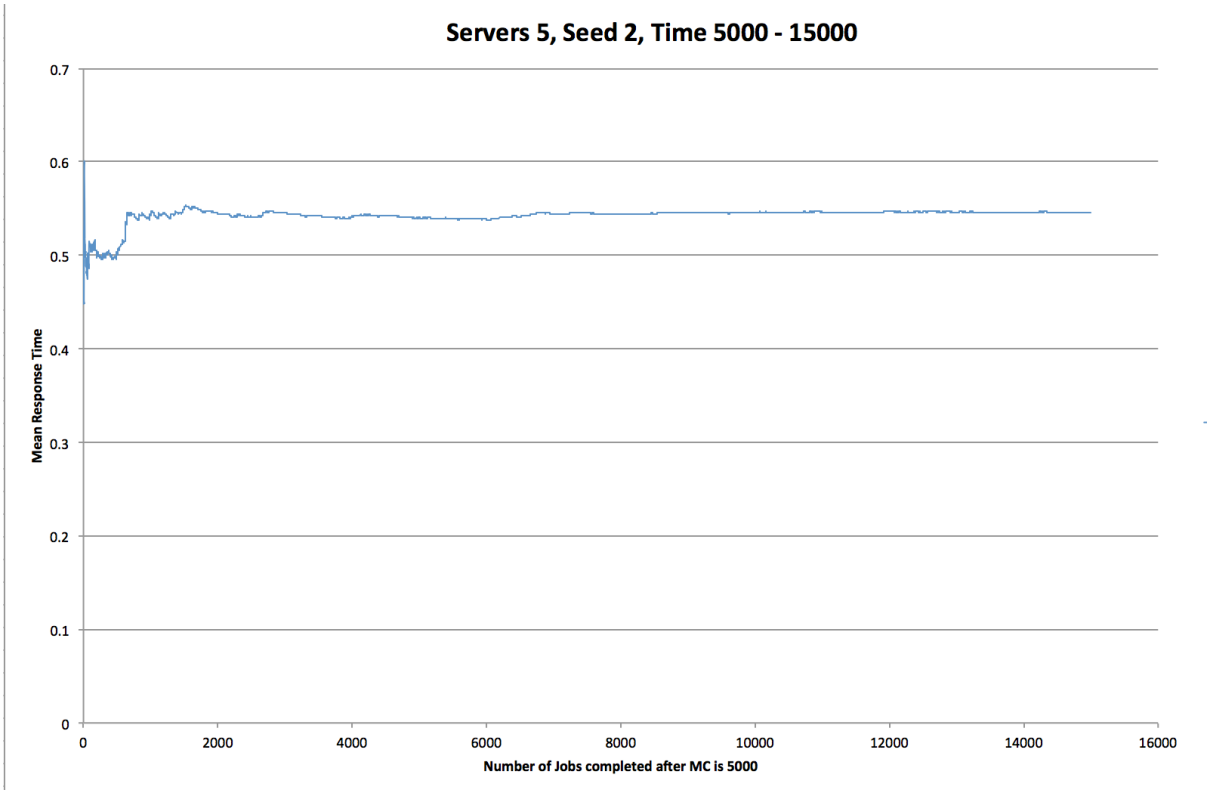


We can see from inspection of the above graphs that the response time leaves the transient section and reaches steady state after around 4000 jobs completed.

To ensure our simulated response times have left the transient section we will start recording the response times after 5000 time units have past.

We now need to decide how long to run the simulation. i.e From 5000 time units to X time units.





From inspection we can see that after removal of the first 5000 time units the mean response time returns to a steady state after a following 10000 time units. Thus we will simulate for 0-15000 time units and remove the first results from 0 - 5000 time units.

INDEPENDENT REPLICATIONS

Now we need to find an appropriate number of independent replications to achieve a confidence interval of 95% that has a small enough range to avoid overlap with the confidence intervals simulations.

First we will try with n = 30 independent replications. If we need to tighten the confidence interval we can either increase n or simulate for longer time to reduce the standard deviation.

3 Servers

Seed	Mean Response Time
1	0.968082436
2	1.01165843
3	1.003298418
4	0.978125311
5	0.977540504
6	1.025093734
7	1.047546367
8	1.035121274
9	1.035590128
10	1.005032247
11	0.975570673
12	1.022300154
13	1.00178242
14	0.971986812
15	1.023872489
16	0.998126317
17	1.001315833
18	0.982632928
19	1.013761194
20	1.018326441
21	1.010792162
22	0.997688981
23	1.070610585
24	0.97995469
25	1.033074155
26	1.046638712
27	1.060239365
28	1.009126017
29	0.982738248
30	1.00362445

Sample Mean
1.009708383

Sample SD
0.026795794

alpha
0.05

scale factor (t 29, 0.975)
2.04523

+/- Scale factor * (S / √n)
0.010005716

Lower bound
0.999702667

Upper bound
1.019714098

4 Servers

Seed	Mean Response Time
1	0.63464628
2	0.61785193
3	0.632199762
4	0.631507602
5	0.633128034
6	0.636228298
7	0.63478278

Sample Mean
0.630695443

Sample SD
0.006631602

alpha
0.05

8	0.623928146	
9	0.638921156	**scale factor (t 29, 0.975)**
10	0.629290194	2.04523
11	0.626338276	
12	0.635655258	**+/- Scale factor * (S / √n)**
13	0.627965373	0.002476281
14	0.633041872	
15	0.64273823	
16	0.631995816	**Lower bound**
17	0.627915688	0.628219162
18	0.635698368	
19	0.634197564	**Upper bound**
20	0.620890478	0.633171724
21	0.629271311	
22	0.611771915	
23	0.637602432	
24	0.625897801	
25	0.625393143	
26	0.62595185	
27	0.630068343	
28	0.634728134	
29	0.631629221	
30	0.639628035	

5 Servers

Seed	Mean Response Time	Sample Mean
1	0.552066987	0.543144946
2	0.545688779	
3	0.541961551	**Sample SD**
4	0.547222257	0.003528652
5	0.545623846	
6	0.538960804	**alpha**
7	0.535176456	0.05
8	0.543329568	
9	0.542350073	**scale factor (t 29, 0.975)**
10	0.539715454	2.04523
11	0.543365202	
12	0.542884593	**+/- Scale factor * (S / √n)**
13	0.547354775	0.001317621
14	0.542595902	
15	0.544027293	
16	0.543604611	**Lower bound**
17	0.544115126	0.541827326
18	0.542584239	
19	0.540711322	**Upper bound**
20	0.534807522	0.544462567

21	0.543894231
22	0.545053141
23	0.540259757
24	0.538289794
25	0.546829823
26	0.544219026
27	0.543302512
28	0.544165087
29	0.544070339
30	0.546118319

6 Servers

Seed	Mean Response Time
1	0.519493955
2	0.512292896
3	0.515657311
4	0.516548809
5	0.518659901
6	0.520016773
7	0.519386681
8	0.521785967
9	0.519000391
10	0.51306208
11	0.522765528
12	0.516034485
13	0.516082033
14	0.520432514
15	0.519474935
16	0.51189244
17	0.517458395
18	0.518170709
19	0.517412674
20	0.516660272
21	0.521097394
22	0.519237953
23	0.516866164
24	0.512226835
25	0.520994809
26	0.519302828
27	0.518061165
28	0.520910787
29	0.515990025
30	0.518503144

Sample Mean
0.517849328

Sample SD
0.002847537

alpha
0.05

scale factor (t 29, 0.975)
2.04523

+/- Scale factor * (S / √n)
0.001063288

Lower bound
0.51678604

Upper bound
0.518912617

7 Servers

Seed	Mean Response Time
1	0.515411347
2	0.513182531
3	0.516323247
4	0.515375042
5	0.514134054
6	0.516335001
7	0.515388019
8	0.514500065
9	0.512841075
10	0.515036258
11	0.513948175
12	0.518645305
13	0.513379182
14	0.513353974
15	0.516150726
16	0.515437287
17	0.514619214
18	0.515071589
19	0.513762196
20	0.51538042
21	0.519487834
22	0.516566629
23	0.515849065
24	0.51644742
25	0.513766393
26	0.512436924
27	0.514437123
28	0.515623028
29	0.517458794
30	0.519328719

Sample Mean
0.515322555

Sample SD
0.001780174

alpha
0.05

scale factor (t 29, 0.975)
2.04523

+/- Scale factor * (S / √n)
0.000664728

Lower bound
0.514657826

Upper bound
0.515987283

8 Servers

Seed	Mean Response Time
1	0.517892523
2	0.5222454
3	0.525498342
4	0.520281803
5	0.523288245
6	0.522867144
7	0.522848173
8	0.52267093
9	0.521174723
10	0.523640597
11	0.521642368

Sample Mean
0.521656586

Sample SD
0.001783123

alpha
0.05

scale factor (t 29, 0.975)
2.04523

12	0.519979925	**+/- Scale factor * (S / √n)**
13	0.518126153	0.000665829
14	0.521842568	
15	0.522346787	
16	0.520702666	**Lower bound**
17	0.523664192	0.520990757
18	0.519905695	
19	0.521143364	**Upper bound**
20	0.524575655	0.522322416
21	0.521755262	
22	0.523228462	
23	0.520509697	
24	0.52192408	
25	0.5213105	
26	0.523209305	
27	0.520556322	
28	0.52199049	
29	0.519157887	
30	0.51971833	

9 Servers

Seed	Mean Response Time	Sample Mean
1	0.532533926	0.530814307
2	0.530896274	
3	0.528453199	**Sample SD**
4	0.532117399	0.001386869
5	0.530169895	
6	0.532422614	**alpha**
7	0.533467524	0.05
8	0.532582667	
9	0.530076181	**scale factor (t 29, 0.975)**
10	0.530547282	2.04523
11	0.531262269	
12	0.531956396	**+/- Scale factor * (S / √n)**
13	0.531663306	0.000517866
14	0.527761558	
15	0.532351624	
16	0.531459757	**Lower bound**
17	0.531878382	0.530296442
18	0.528406667	
19	0.529181691	**Upper bound**
20	0.529580274	0.531332173
21	0.530920419	
22	0.528903073	
23	0.531481745	
24	0.529880904	

25	0.531507067
26	0.53049731
27	0.530368592
28	0.530209344
29	0.531223934
30	0.53066794

10 Servers

Seed	Mean Response Time
1	0.539673373
2	0.543188946
3	0.541158319
4	0.542584734
5	0.539650387
6	0.540318834
7	0.541489529
8	0.540681794
9	0.541800349
10	0.543544824
11	0.542381336
12	0.544311077
13	0.541246143
14	0.541673835
15	0.54326916
16	0.542493609
17	0.542638447
18	0.541528388
19	0.543732403
20	0.538383764
21	0.543832035
22	0.541411981
23	0.544446093
24	0.540692801
25	0.540613651
26	0.53954409
27	0.544608901
28	0.540431381
29	0.543735935
30	0.542501884

Sample Mean
0.541918933

Sample SD
0.001630058

alpha
0.05

scale factor (t 29, 0.975)
2.04523

+/- Scale factor * (S / √n)
0.000608674

Lower bound
0.54131026

Upper bound
0.542527607

We can see that there is minimal overall between adjacent number of servers so we do not need to increase n. If wish to confirm which is better out of 10 vs 5 servers we would need to increase our number of replications or increase our simulation time. However we can see the fastest number of servers with 95% confidence so we will stay with n = 30 and sim time set to 5000 - 15000

Comparing Response Times

From the upper and lower bounds calculated above and the charts below we can conclude with 95% confidence that the optimal number of servers is 7 with a 95% probability of mean response time between [0.514658, 0.515987]. We can conclude this because we have no overlap in the confidence intervals closest to the 7 server interval. Thus, this is the correct choice with >95% probability as there is space between 7 server interval and the next fastest interval (6 servers). This means we could increase the width of the interval and the confidence percentage above 95% while still avoiding any overlap between 6 and 7 intervals. In order from best to worst response time the order of servers is: 7, 6, 8, 9, 10*, 5*, 4, 3
*Note: that we cannot see any clear difference between 5 and 10 servers