



COMP 9334

Processor Sharing Project

Ivan Teong (z3386180)



Reproducibility and Test Cases

2

Removing Transient

3

Conclusion

9

Reproducibility and Test Cases

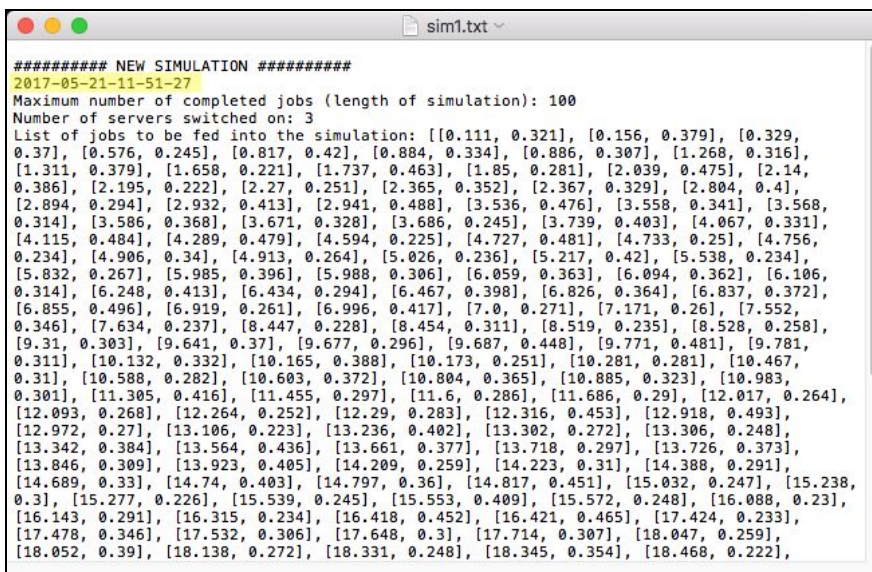
In our program for **simulation.py**, there is a choice to choose between 1) using the current time as a random seed to generate random arrival and service times for each job, or 2) reproduce a past simulation using an input for the seed stored in `sim*.txt`, but varying other parameters.

```
>>>
RESTART: /Users/ivanteong/Desktop/MIT/2017/COMP9334 Systems Capacity Planning/Assignments/proj/simulation.py
Choose name of text file to save values for recording seed values for simulation (e.g. sim1): sim1
Choose 1 to run simulation based on random seed of current time, or choose 2 to reproduce past simulation: 1
Seed of date and time stored in seed.txt: 2017-05-21-21-41-18
Choose the maximum number of completed jobs after a job departure (length of simulation): 100
Choose 1 to generate jobs for trace-driven simulation, or choose 2 to use jobs from test case: 1
Choose the number of jobs to generate that will be fed into the simulation in a list: 400
Choose the number of servers to switch on (3, 4, 5, 6, 7, 8, 9, 10): 4
Choose name of trace file to save values of each departing job's response time to (e.g. trace1): trace1
```

The random seed is stored inside **sim1.txt**, **sim2.txt**, **sim3.txt**, **sim4.txt** and **sim5.txt** inside the respective **s*** folder, where ***** = number of servers turned on. The other parameters such as the maximum number of completed jobs (100), number of jobs to generate to be fed into the simulation in a list (400) will remain the same for the trace-driven process-sharing simulations for the whole experiment.

Reproducing the random seed:

```
>>>
RESTART: /Users/ivanteong/Desktop/MIT/2017/COMP9334 Systems Capacity Planning/Assignments/proj/simulation.py
Choose name of text file to save values for recording seed values for simulation (e.g. sim1): sim1
Choose 1 to run simulation based on random seed of current time, or choose 2 to reproduce past simulation: 2
Enter the seed of current time recorded: 2017-05-21-21-36-14
```



```
##### NEW SIMULATION #####
2017-05-21-11-51-27
Maximum number of completed jobs (length of simulation): 100
Number of servers switched on: 3
List of jobs to be fed into the simulation: [[0.111, 0.321], [0.156, 0.379], [0.329, 0.37], [0.576, 0.245], [0.817, 0.42], [0.884, 0.334], [0.886, 0.307], [1.268, 0.316], [1.311, 0.379], [1.658, 0.221], [1.737, 0.463], [1.85, 0.281], [2.039, 0.475], [2.14, 0.386], [2.195, 0.222], [2.27, 0.251], [2.365, 0.352], [2.367, 0.329], [2.804, 0.4], [2.894, 0.294], [2.932, 0.413], [2.941, 0.488], [3.536, 0.476], [3.558, 0.341], [3.568, 0.314], [3.586, 0.368], [3.671, 0.328], [3.686, 0.245], [3.739, 0.403], [4.067, 0.331], [4.115, 0.484], [4.289, 0.479], [4.594, 0.225], [4.727, 0.481], [4.733, 0.25], [4.756, 0.234], [4.906, 0.34], [4.913, 0.264], [5.026, 0.236], [5.217, 0.42], [5.538, 0.234], [5.832, 0.267], [5.985, 0.396], [5.988, 0.306], [6.059, 0.363], [6.094, 0.362], [6.106, 0.314], [6.248, 0.413], [6.434, 0.294], [6.467, 0.398], [6.826, 0.364], [6.837, 0.372], [6.855, 0.496], [6.919, 0.261], [6.996, 0.417], [7.0, 0.271], [7.171, 0.26], [7.552, 0.346], [7.634, 0.237], [8.447, 0.228], [8.454, 0.311], [8.519, 0.235], [8.528, 0.258], [9.31, 0.303], [9.641, 0.37], [9.677, 0.296], [9.687, 0.448], [9.771, 0.481], [9.781, 0.311], [10.132, 0.332], [10.165, 0.388], [10.173, 0.251], [10.281, 0.281], [10.467, 0.31], [10.588, 0.282], [10.603, 0.372], [10.804, 0.365], [10.885, 0.323], [10.983, 0.301], [11.305, 0.416], [11.455, 0.297], [11.6, 0.286], [11.686, 0.29], [12.017, 0.264], [12.093, 0.268], [12.264, 0.252], [12.29, 0.283], [12.316, 0.453], [12.918, 0.493], [12.972, 0.27], [13.106, 0.223], [13.236, 0.402], [13.302, 0.272], [13.306, 0.248], [13.342, 0.384], [13.564, 0.436], [13.661, 0.377], [13.718, 0.297], [13.726, 0.373], [13.846, 0.309], [13.923, 0.405], [14.209, 0.259], [14.223, 0.31], [14.388, 0.291], [14.689, 0.33], [14.74, 0.403], [14.797, 0.36], [14.817, 0.451], [15.032, 0.247], [15.238, 0.3], [15.277, 0.226], [15.539, 0.245], [15.553, 0.409], [15.572, 0.248], [16.088, 0.23], [16.143, 0.291], [16.315, 0.234], [16.418, 0.452], [16.421, 0.465], [17.424, 0.233], [17.478, 0.346], [17.532, 0.306], [17.648, 0.3], [17.714, 0.307], [18.047, 0.259], [18.052, 0.39], [18.138, 0.272], [18.331, 0.248], [18.345, 0.354], [18.468, 0.222],
```

We also enabled us to check that the round robin assignment is working properly by using the test cases given in the Appendix.

```
>>>
RESTART: /Users/ivanteong/Desktop/MIT/2017/COMP9334 Systems Capacity Planning/Assignments/proj/simulation.py
Choose name of text file to save values for recording seed values for simulation (e.g. sim1): test
Choose 1 to run simulation based on random seed of current time, or choose 2 to reproduce past simulation: 1
Seed of date and time stored in seed.txt: 2017-05-21-22-50-01
Choose the maximum number of completed jobs after a job departure (length of simulation): 5
Choose 1 to generate jobs for trace-driven simulation, or choose 2 to use jobs from test case: 2
Choose name of trace file to save values of each departing job's response time to (e.g. trace1): testtrace
```

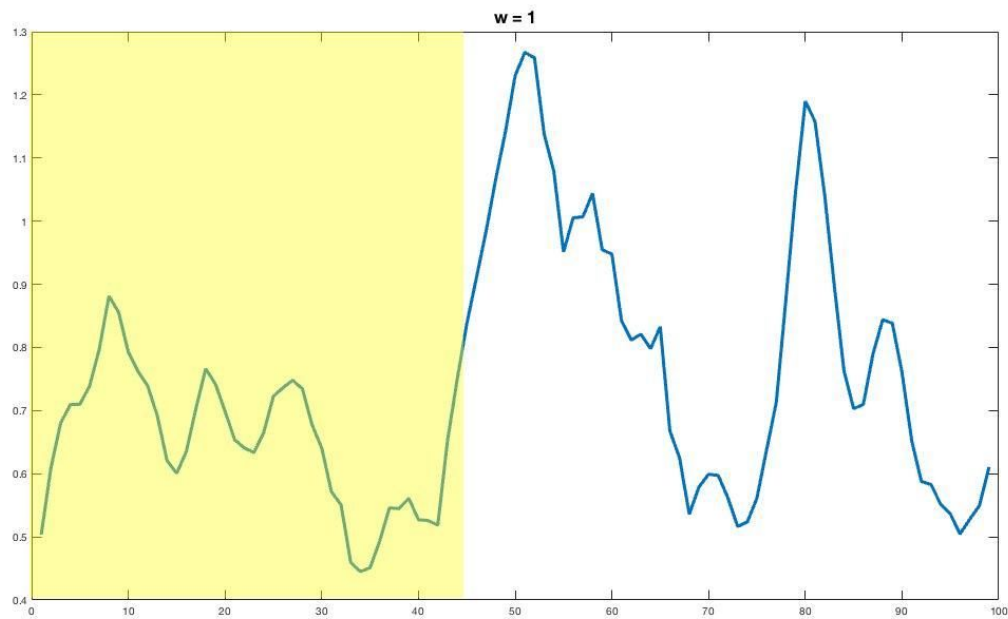
Removing Transient

We describe the steps here to remove the transient for our simulation with the following parameters. The simulation is in simulation.py and will generate the relevant simulation files with random seed details, sim* and its respective trace file containing response times of each job, trace*, where * is the nth replication:

- **Length of simulation determined by number of completed jobs** = 100
- **Number of jobs fed into the processor-sharing simulation system** = 400
- **Number of independent replications seeded by random time using current date and time** = 5
- **Type of simulation** = trace-driven, a list of jobs are generated first with random arrival times and service times, then fed into the simulation
- **Servers switched on** = 3

A program that implements the transient removal procedure in Section 9.5.1 Law and Kelton can be found in the matlab file, **week07_q1_a.m**. The value of m is the number of data points, which in our case is the number of completed jobs for each simulation. We loaded up the **trace1** to **trace5** files into the matlab file to compute the array for the transient removal. They contain the response times for the 100 completed jobs we chose to simulate for the length of simulation. There are a total of 5 simulations, with the seed values and details of each simulation recorded in **sim1.txt** to **sim5.txt**.

We adjusted w until we got a smoothed curve. There are no hard rules as to how to choose w. This is done by trial-and-error.



We tried $w = 1$ till $w = 10$. We settled on **$w = 1$** as the graph is smoother, but it still oscillates. It is difficult to get the ideal shape, where the graph rises up initially and then settles down to a steady state value. From the figure, we can see that the curve oscillates around the response time of 0.85. That is probably the mean value. Based on that, the suggestion is to cut away the first 45 points of the total 100 data points, which equates to the response times of the first 45 completed jobs. This is done before we compute the mean response time for each of the 5 simulations.

Since we have decided to take the first 45 points as the transient, I will use the 46th to 100th data point for each simulation to compute the mean response time. Referencing the 46th to 100th response times for each job in the trace files, I was able to compute the following mean response time for each simulation after removal of the transient:

Mean Response Time for Simulations

Using **trace_calc.py**, I calculated the mean response time for different simulations by inputting the trace file name and the data point that I want to start iterating from in order to calculate the mean. In this case, I'm iterating from the 46th data point (which equates to the 46th line in the trace file) till the 100th data point, and using that to calculate the mean by dividing it over the number of data points (which equals to the number of completed jobs after removing the first 45 jobs that belong to the transient).

- **Mean response time of sim1:** 0.9788
- **Mean response time of sim2:** 0.775090909091
- **Mean response time of sim3:** 0.716563636364
- **Mean response time of sim4:** 0.786036363636
- **Mean response time of sim5:** 0.726654545455

The steady state mean response time (sample mean) is calculated to be **0.796629090909 sec**, after summing the 5 values and dividing by 5.

$$\hat{T} = \frac{\sum_{i=1}^n T(i)}{n}$$

The sample standard deviation is calculated to be **0.106147601585** using the following formula:

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

Confidence Interval

- There is a probability $(1-\alpha)$ that the mean response time that you want to estimate lies in the interval

$$\left[\hat{T} - t_{n-1, 1-\frac{\alpha}{2}} \frac{\hat{S}}{\sqrt{n}}, \hat{T} + t_{n-1, 1-\frac{\alpha}{2}} \frac{\hat{S}}{\sqrt{n}} \right]$$

where $t_{n-1, 1-\frac{\alpha}{2}}$ is the upper $(1 - \frac{\alpha}{2})$ critical point for the Student t distribution with $(n - 1)$ degrees of freedom

To compute a 95% confidence interval for our 5 independent replications, we need the value of $t_{4, 0.975}$ which is calculated from $n - 1 = 4$ ($n = 5$ independent replications) and $\alpha = 1 - 0.95$ for 95% confidence interval, and $1 - \alpha/2 = 0.975$. Using **t-table.pdf** to get the t-value, we have **2.776** for the t-value for $t_{4, 0.975}$.

Therefore, the 95% confidence interval is:

[**0.796629090909** - **2.776** (**0.106147601585**/ $\sqrt{5}$), **0.796629090909** + **2.776** (**0.106147601585**/ $\sqrt{5}$)] = [**0.6648505649582391**, **0.9284076168599423**] for this particular sample, where which we switched on 3 processor-sharing servers ($s = 3$).

Simulation Results for Different Systems with Varying s

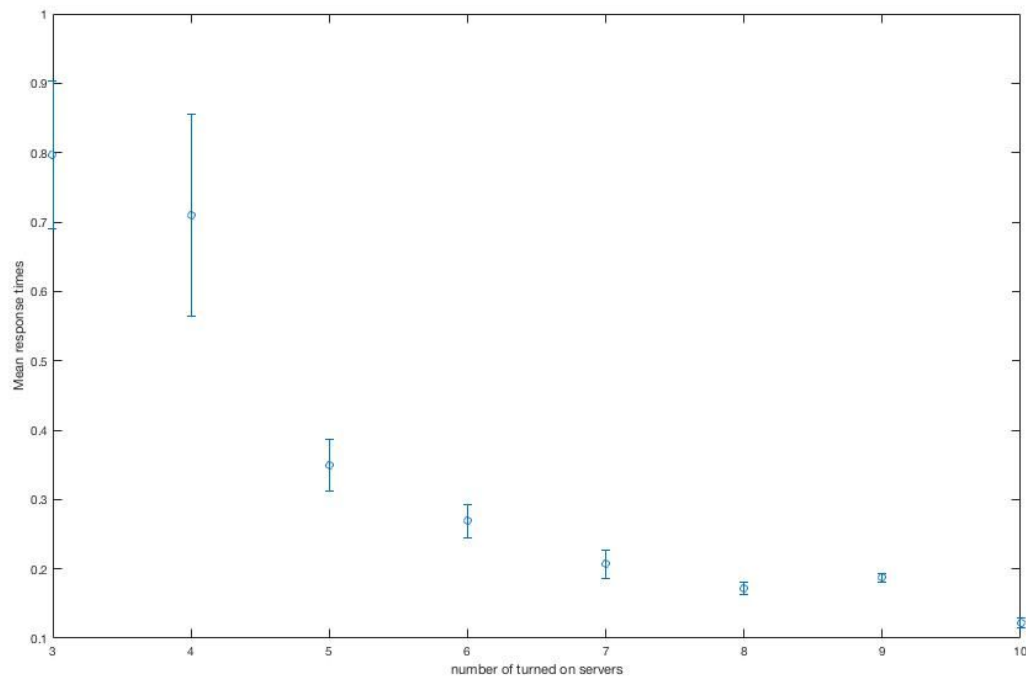
Now that we have covered the steps of how we removed the transient and computing the confidence intervals, we will use the steps to compute the different confidence intervals in order to compare the different systems where number of servers switched on will vary, to find out which system is better. The values are calculated using **trace_calc.py**, in the different folders of $s3$ to $s10$ that contains the respective trace files from trace1 to trace5 representing the 5 replications. The total data points reduced over time, due to having to skip jobs that do not belong to the server that we are simulating. This may have impacted some part of the data. (Note: if I was to redo it over again, I would have fed more jobs into the server so that the total data points will be consistently 100)

Computed results:

s	Transient	Total data pts	Mean Response Times for 5 replications	Sample Mean	Sample Standard Deviation	Confidence Interval
3	45	100	sim1: 0.9788 sim2: 0.775090909091 sim3: 0.716563636364 sim4: 0.786036363636 sim5: 0.726654545455	0.796629090909	0.106147601585	[0.6648505649582391, 0.9284076168599423]
4	35	100	sim1: 0.691492307692 sim2: 0.935015384615 sim3: 0.5386 sim4: 0.735246153846 sim5: 0.645815384615	0.709233846154	0.145865105633	[0.5281474313790796, 0.8903202609286137]
5	40	80	sim1: 0.283933333333 sim2: 0.361066666667 sim3: 0.371366666667 sim4: 0.362716666667 sim5: 0.369566666667	0.34973	0.0370405641006	[0.30374543827027567, 0.39571456172972475]
6	25	67	sim1: 0.250306666667 sim2: 0.265893333333 sim3: 0.3098 sim4: 0.259546666667 sim5: 0.25788	0.268685333333	0.0236445391393	[0.23933146694320712, 0.29803919972345944]
7	30	58	sim1: 0.188485714286 sim2: 0.187285714286 sim3: 0.221342857143 sim4: 0.233242857143	0.20704	0.02018961493	[0.18197530088997588, 0.23210469911002407]

			sim5: 0.204842857143			
8	25	50	sim1: 0.16004 sim2: 0.180093333333 sim3: 0.171853333333 sim4: 0.16608 sim5: 0.179653333333	0.171544	0.00867677 052057	[0.160772093596 67147, 0.1823159064033 2865]
9	13	45	sim1: 0.195034482759 sim2: 0.191356321839 sim3: 0.183770114943 sim4: 0.180298850575 sim5: 0.185137931034	0.18711954 023	0.00596372 085548	[0.179715789869 25493, 0.1945232905905 153]
10	22	40	sim1: 0.112782051282 sim2: 0.128397435897 sim3: 0.117871794872 sim4: 0.128615384615 sim5: 0.122358974359	0.12200512 8205	0.00683432 306298	[0.113520555726 2384, 0.1304897006840 1797]

Using **systems_compare.m**, I was able to plot the confidence intervals and visually inspect the different systems with different values of s to see which will result in a better performance, indicated by a lower mean response time.



We can see approximate the performance of different systems with varying s by visually inspecting the confidence intervals and seeing if the mean and confidence intervals overlap with each other. It seems that using $s = 10$ will give us the lowest response time, followed by $s = 8$.

We use the following rules to help us:

- If 2 systems' confidence intervals overlaps with each other and the mean of one system is in the confidence interval of the other, then the 2 systems are not different.
- If 2 systems' confidence intervals overlaps with each other, but the mean of one system is not in the confidence interval of the other, then do the t-test.

We can see from visual inspection that almost all the systems are not similar, besides the systems for $s = 7$ and $s = 8$, where there might be slight overlap of the confidence intervals upon closer inspection.

Conclusion

From what we have observed, we can conclude that $s = 10$ will most likely result in the lowest mean response time, which gives us the best performance. However, this may be slightly skewed as the total data points from $s = 5$ to $s = 10$ starts to drop from $m = 100$. This is due to the round robin job assignment, where we ignore any jobs that are not routed to our server as for this simulation, we are only simulating one of the servers.