M/M/N Queue Simulation

Supermarket Model Extension

Milo Galli S4793560
Giorgio Rengucci S4483986
12 December 2022

Simulation's results analysis

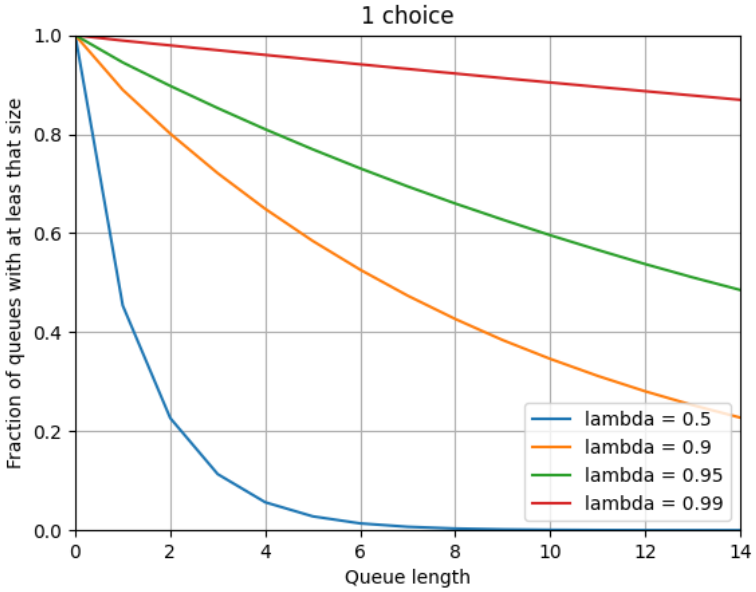
Once we finished implementing the code we tested it right away with different lambdas in order to see if our implementation matched the theoretical result.

Test run with the following arguments:

- `lambdas = [ 0.5, 0.9, 0.95, 0.99 ]`
- `n = 10`
- `d = 1`

```
Average time spent in the system: 2.0058426112511913
Theoretical expectation for random server choice: 2.0
Average time spent in the system: 9.859321204182024
Theoretical expectation for random server choice: 10.000000000000002
Average time spent in the system: 19.287585642344588
Theoretical expectation for random server choice: 19.999999999999982
Average time spent in the system: 86.64736023972176
Theoretical expectation for random server choice: 99.99999999999991
```

At a first glance at the results given by the terminal it's possible to have an over all understanding of the performance of our implementation.
Since the theoretical expectation, that is based on the random queue choice model, matched our result obtained choosing only one queue at the time for the job schedule, we proceeded on looking at the distribution of jobs in the queues.



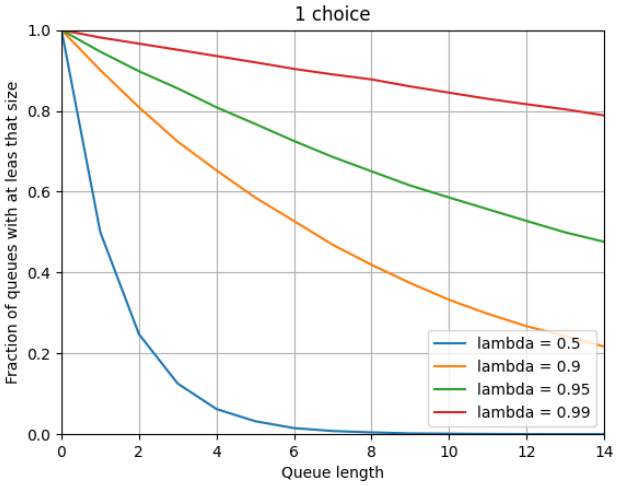
From the graph it's possible to see how incrementing the lambda, the arrival rateo of the jobs inside the system, and maintaining a stable mu, the rateo at which the jobs exit the system, the crowdedness of queues seem to have a much higher average.

For lambda equal to 0.5 half of the queues have at least one element or exactly one element, on the other hand with lambda equal to 0.99 the majority of the queues have at least 14 jobs in them. As we learn from theory and looking at the results obtained we can state that the random queue choice is not the best approach when it comes to load balancing of the jobs.

In a second moment after implementing the supermarket model we analysed once again the results with this arguments:

- `lambdas = [ 0.5, 0.9, 0.95, 0.99 ]`
- `n = 1000`
- `d = [1, 2, 5, 10]`
- `max-t = 10_000`

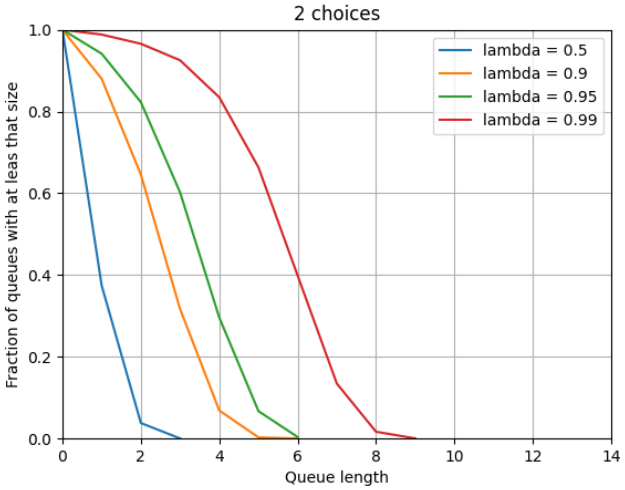
`d = 1`




```
Average time spent in the system: 1.9981636462239336
Theoretical expectation for random server choice: 2.0
Average time spent in the system: 9.884563540772264
Theoretical expectation for random server choice: 10.000000000000002
Average time spent in the system: 19.33247866951701
Theoretical expectation for random server choice: 19.999999999999982
Average time spent in the system: 53.03521113877056
Theoretical expectation for random server choice: 99.99999999999991
```

From here it's possible to see that even if we incremented the number of queues the average time spent in the system and the distribution of jobs is not affected when it comes to random queue choice.

`d = 2`

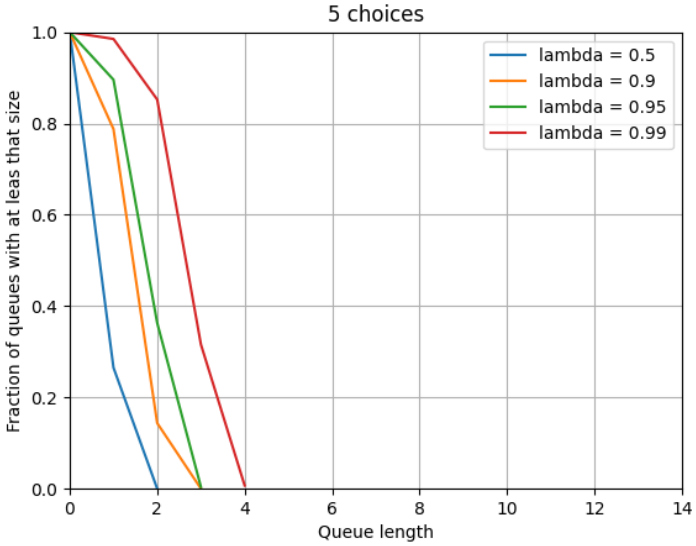


Incrementing the value of d with the changes applied for the supermarket model we begin to see some changes mostly from the graph.

From a deeper analysis we can state that the distribution of the jobs in the queues is highly influenced by size of the queues' subset considered every time a job arrives in the system.
A major change is represented by the steepness of the curves which means that on average the queues are less crowded than the random queue choice model.

So we can say that the job load is better balanced than before and we the maximum amount of jobs reached by the queues is significantly less.

`d = 5`



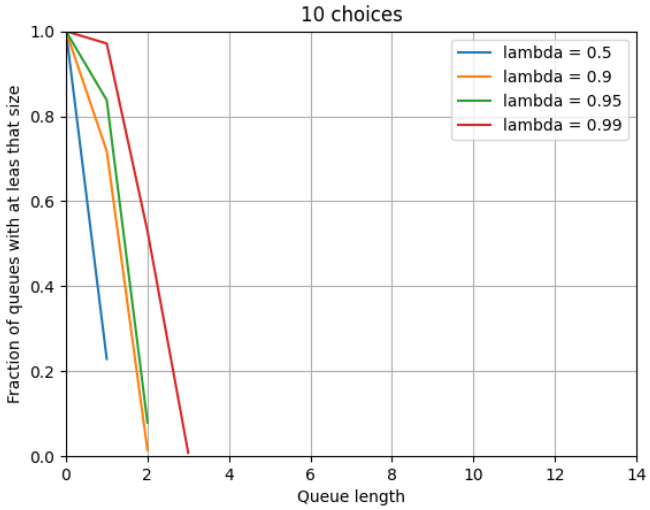

```
Average time spent in the system: 1.4391854719357122
Theoretical expectation for random server choice: 2.0
Average time spent in the system: 2.6675484251654416
Theoretical expectation for random server choice: 10.000000000000002
Average time spent in the system: 3.42501360549378
Theoretical expectation for random server choice: 19.999999999999982
Average time spent in the system: 5.489010702274706
Theoretical expectation for random server choice: 99.99999999999991
```

As we continue to increment the value of d the average length of the queues continues to drop down and we can see the true advantages in the use of the supermarket model.
Here the maximum length value of the queues is 4!

Major changes here can also be seen in the average time spent in the system.

Since the workload is better distributed among the queues each job waits the least amount of time needed before its execution and it's trivial to understand that contributes to speed up the system.

`d = 10`

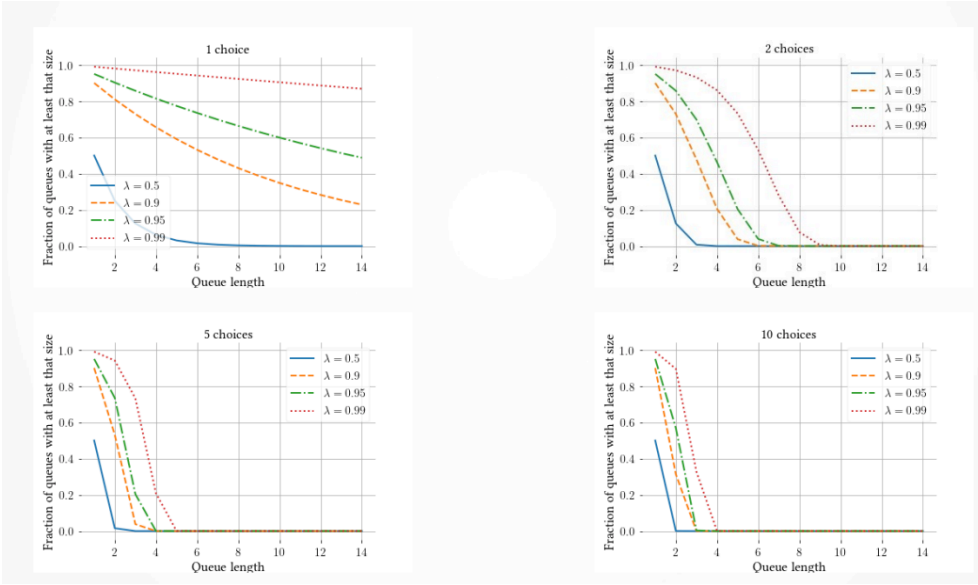



```
Average time spent in the system: 1.380208081172062
Theoretical expectation for random server choice: 2.0
Average time spent in the system: 1.733753800926515
Theoretical expectation for random server choice: 10.000000000000002
Average time spent in the system: 1.8788483098359974
Theoretical expectation for random server choice: 19.999999999999982
Average time spent in the system: 2.43101206805838
Theoretical expectation for random server choice: 99.99999999999991
```

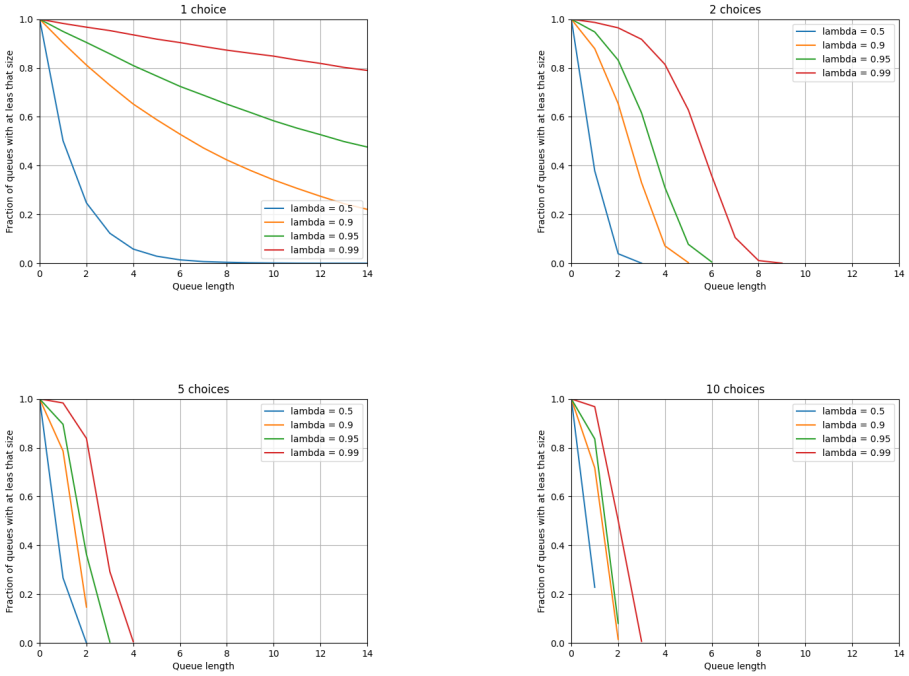
The changes here are clearer than ever and all the observations made until this point in the paper find another solid find confirmation.

Final considerations

Theoretical results taken from the slides



Practical results obtained from our simulation



Theoretical values have been obtained using an infinite number of queues and 1000_000 as max running time, our results on the other hand are the product of testing the simulation with 1000 queues and 10_000 as max running time.
That's the reason behind the negligible differences in the graphs.
Apart from that it's still possible to understand the benefits brought by load balancing in the simulation