Francesco Stucci - Matricola N° S4234595

Report Queue Simulator

M/M/1 analysis

For this simulation, our environment is composed by a single queue of jobs. This is the simplest example and it is scheduled as a **FIFO queue**. My goal is to compare theorical values with practical ones, therefore I make tests with different values of arrival and completion rates.
We define **LAMBDA** as the mean arrival rate and, for simplicity, our completion rate is set to 1.
In order to give our parameters an exponential scale, we leverage on **Python Expovariate** function, which means the more LAMBDA is close to 1, the less time new jobs take to arrive.
The ***LAW*** theory tells us the average queue length (*L*) is equal to the arrival rate (*A*) times the average time spent (*W*).

The following picture proves the simulator is working as theory claims:

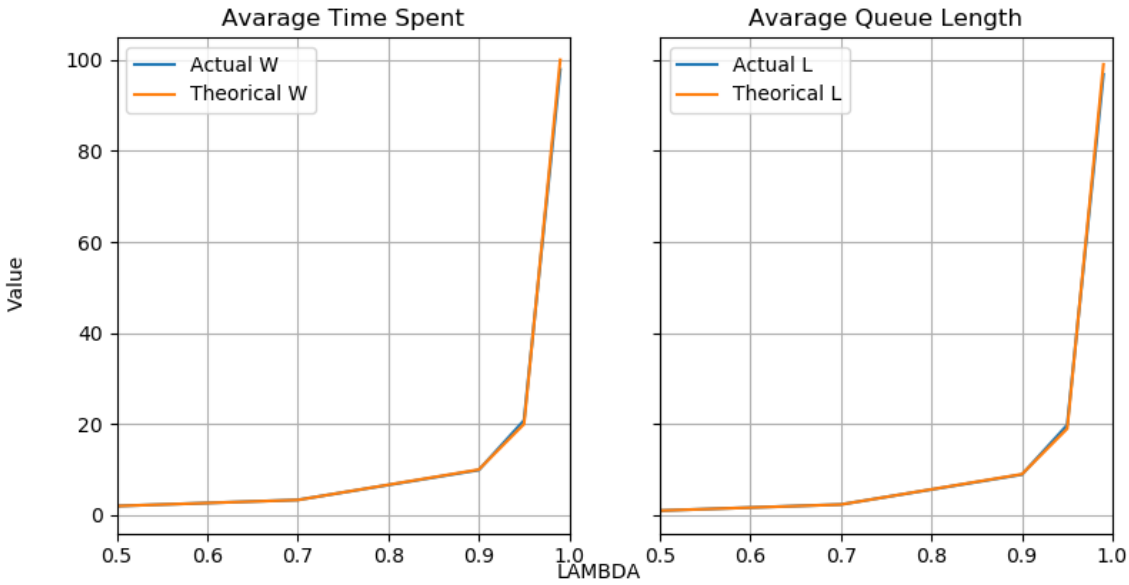

Figure 1: comparison between Theorical and Obtained data

This graph shows a trivial concept: the more quickly a new job arrives, the more long the average queue is and the more time the job waits until its completion.

Finally, I make a graph of the fraction of queue having a certain value of length:

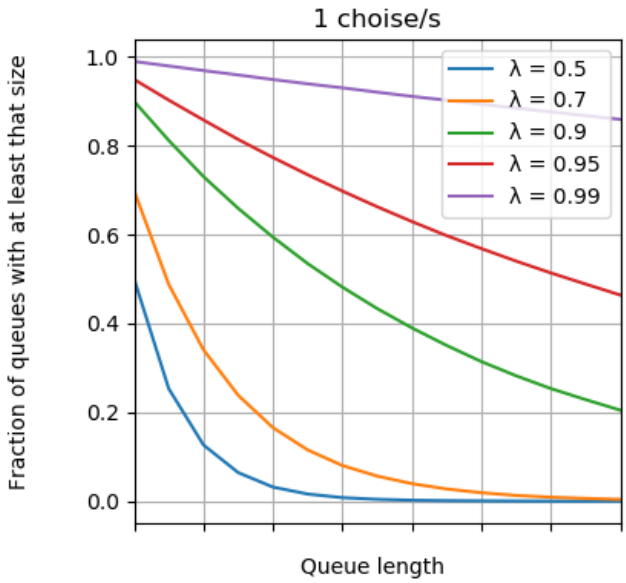

Figure 2: queue length values

This graph as well shows how the queue length grows by making the arrival time shorter.

M/M/N analysis

In this phase we analyze the case of **multiple queues** of jobs instead of only one. Before on trying some scheduling variants, the first attempt will be a **random choice** of the queue for arrival jobs, and all queues are **FIFO**

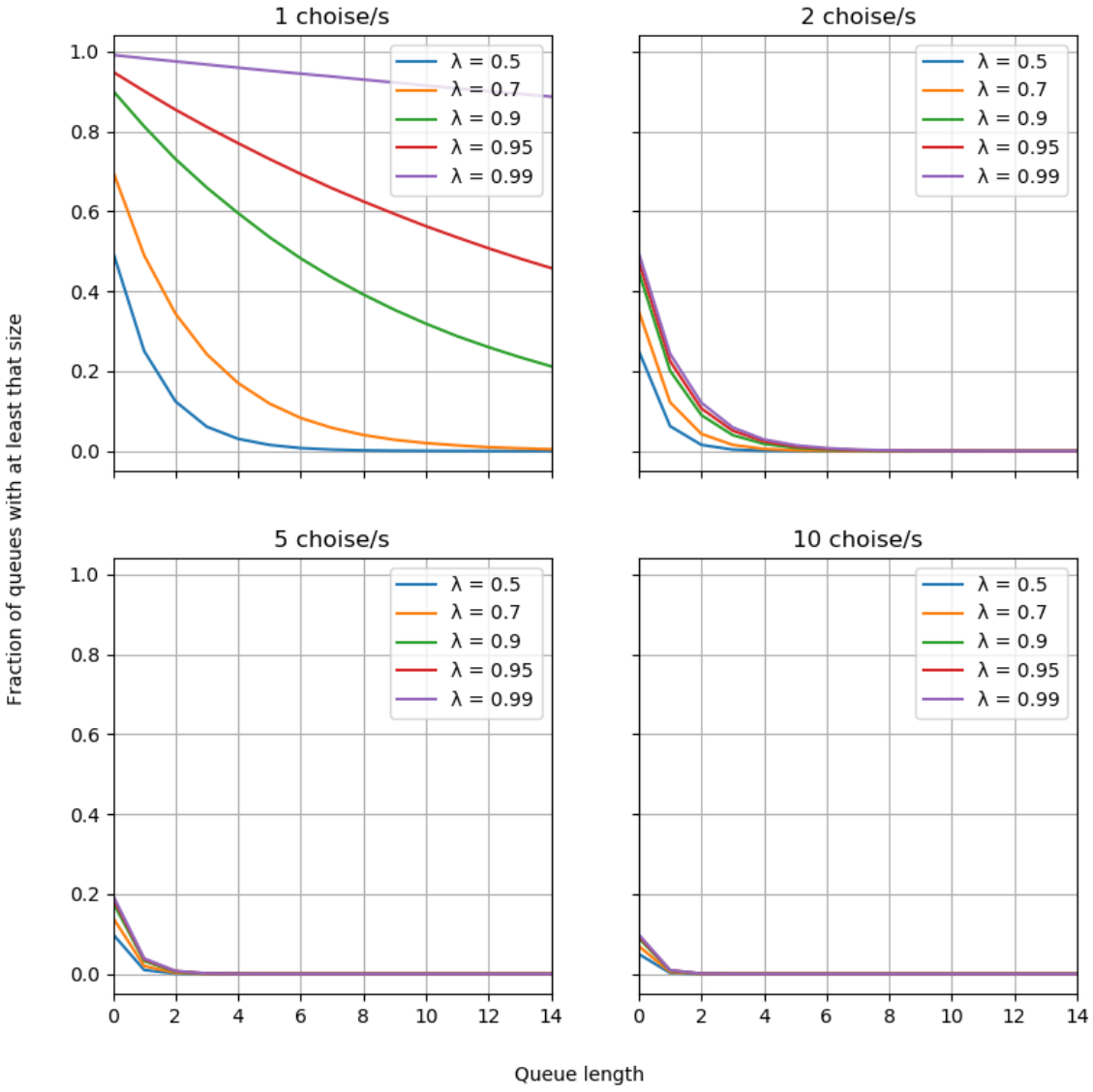

Figure 3: FIFO, random choice, no overload

We can notice we get much better result by using additional queue of course. However, in this case, we have no overload, which means the arrival time is not multiplied by the number of servers.

If we add overload, we get the following graph:

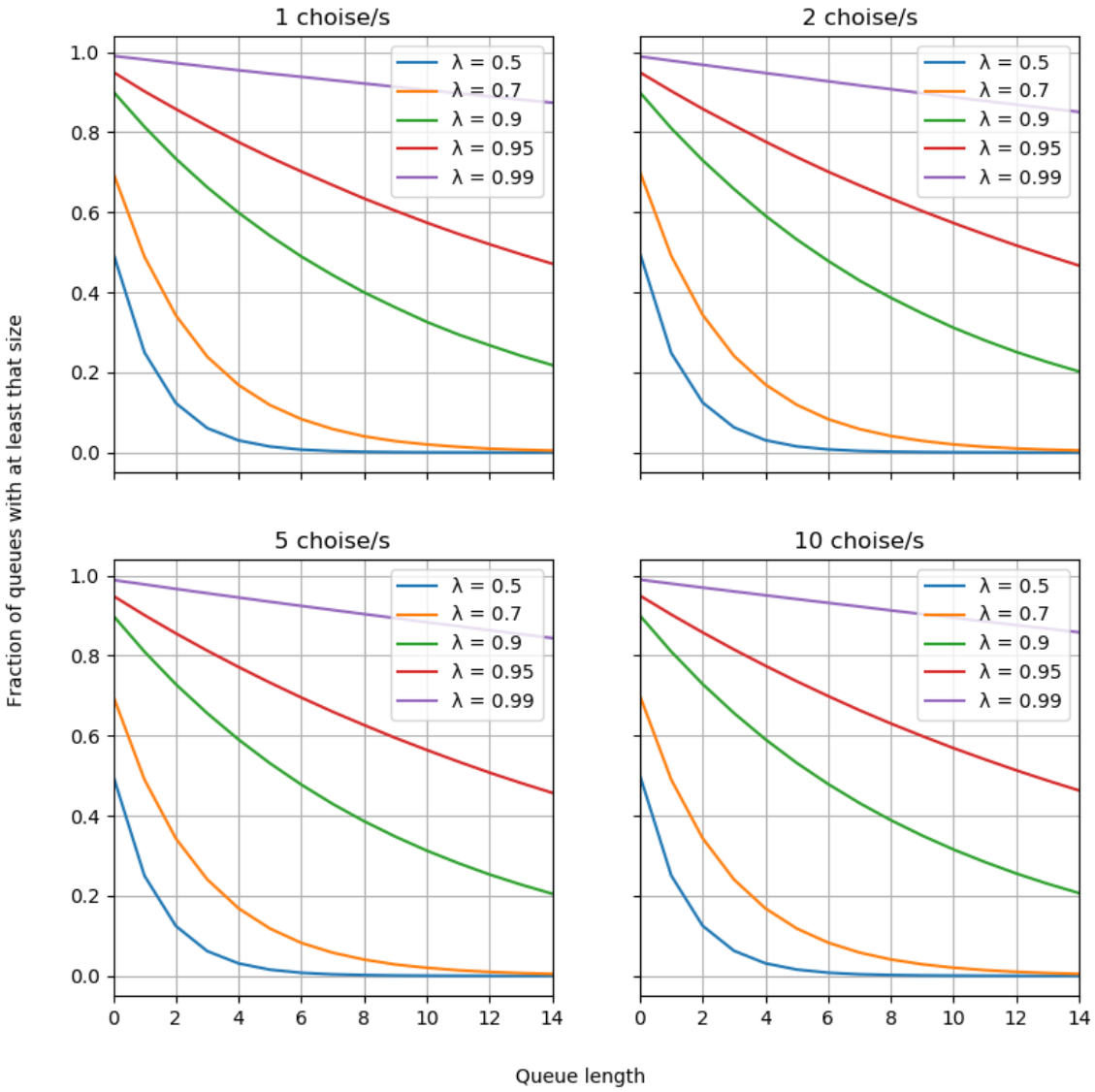

Figure 4: FIFO, random choice with overload

By adding overloading and multiplying LAMBDA for the number of servers, we can notice we are not enhancing our system.

In order to obtain some better result, I try to use another queue choice policy, the so-called supermarket queue choice. This makes the new arrival jobs choice not a random queue but the shortest one.

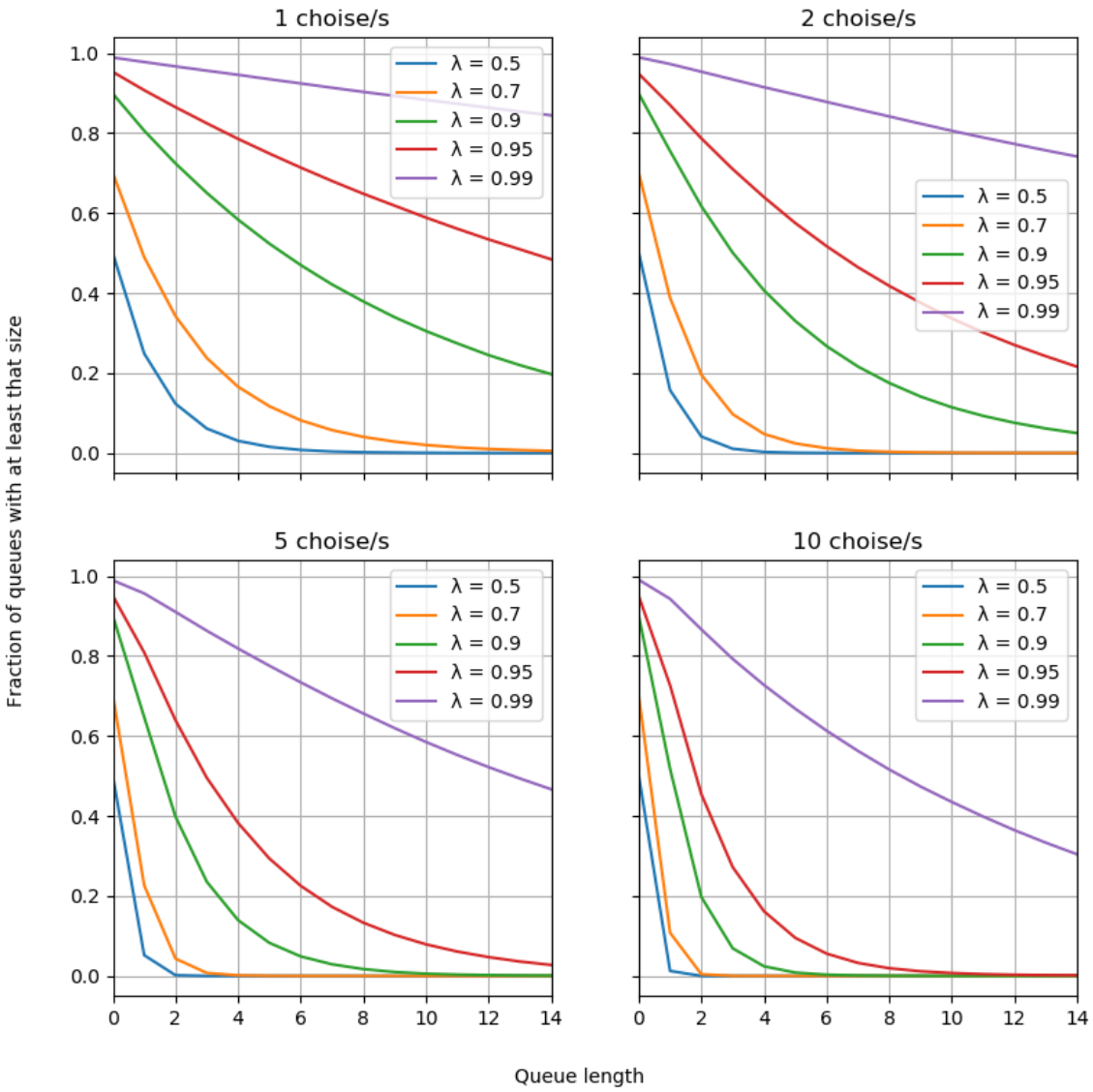

Figure 5: FIFO, supermarket choice with overload

This is a great result. By changing the choice policy, we enhanced a lot our system. We can notice the more server we have, the shorter our queues are even though we have a proportional arrival time.

Finally, I changed the scheduling policy. Previously I've always used **FIFO**, so the first arrival job is the first to be served independently form its duration, now I switch to **SJF** (shortest job first), which means the less time demanding jobs are served before others.
Notice this is hard to make in practice since its hard to "predict" how much time a job takes!

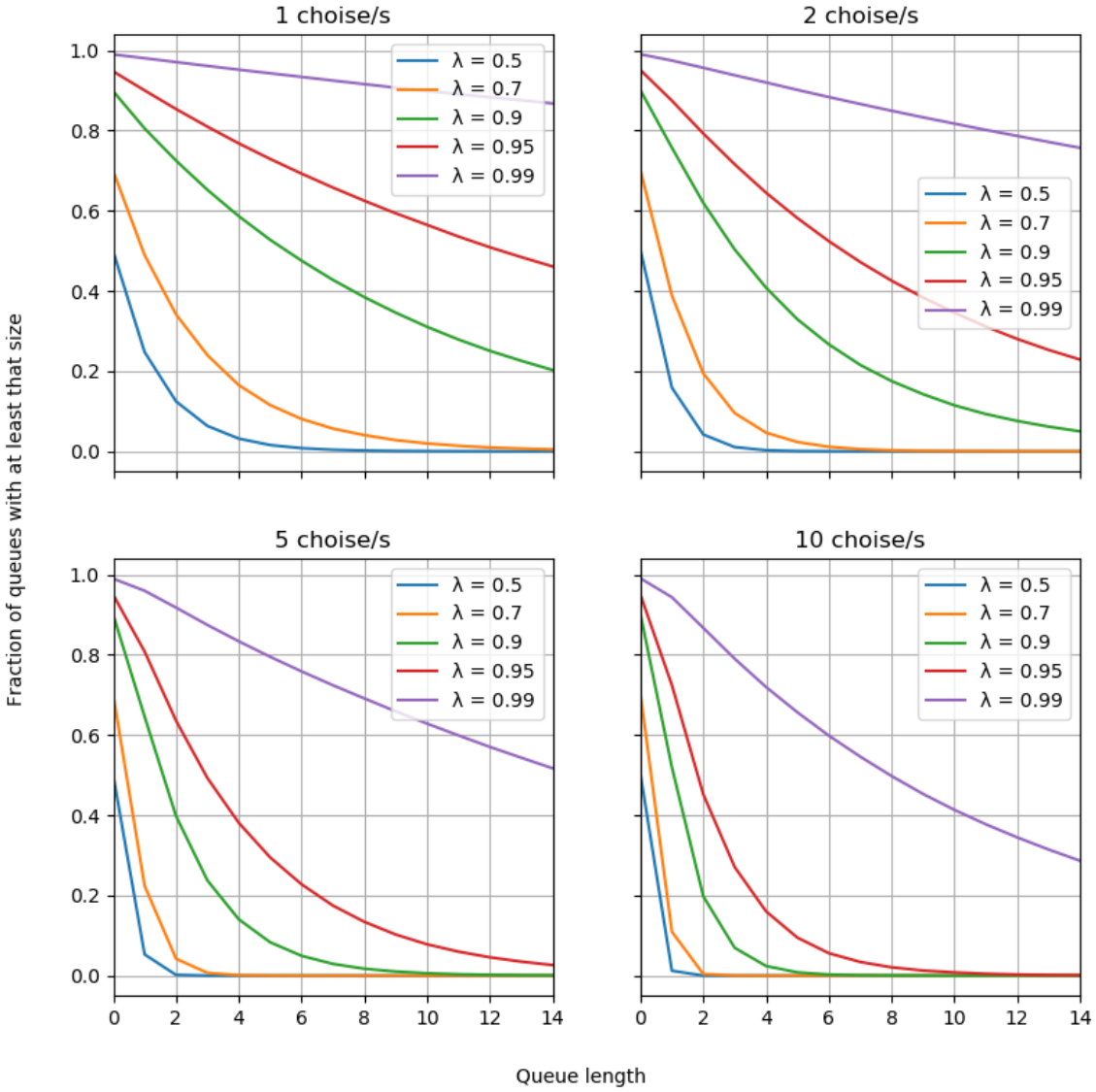

Figure 6: SJF, supermarket choice with overload

By using **SJF** it seems we got no improvement on our system!

Real Data analysis

For this analysis I gather some traces from https://www.pdl.cmu.edu/ATLAS/ data.
Given the list of jobs with their starting and ending time, I tried to compute the **LAMBDA** values of the queue. I used the *LAW* formula applied in the previous analysis.

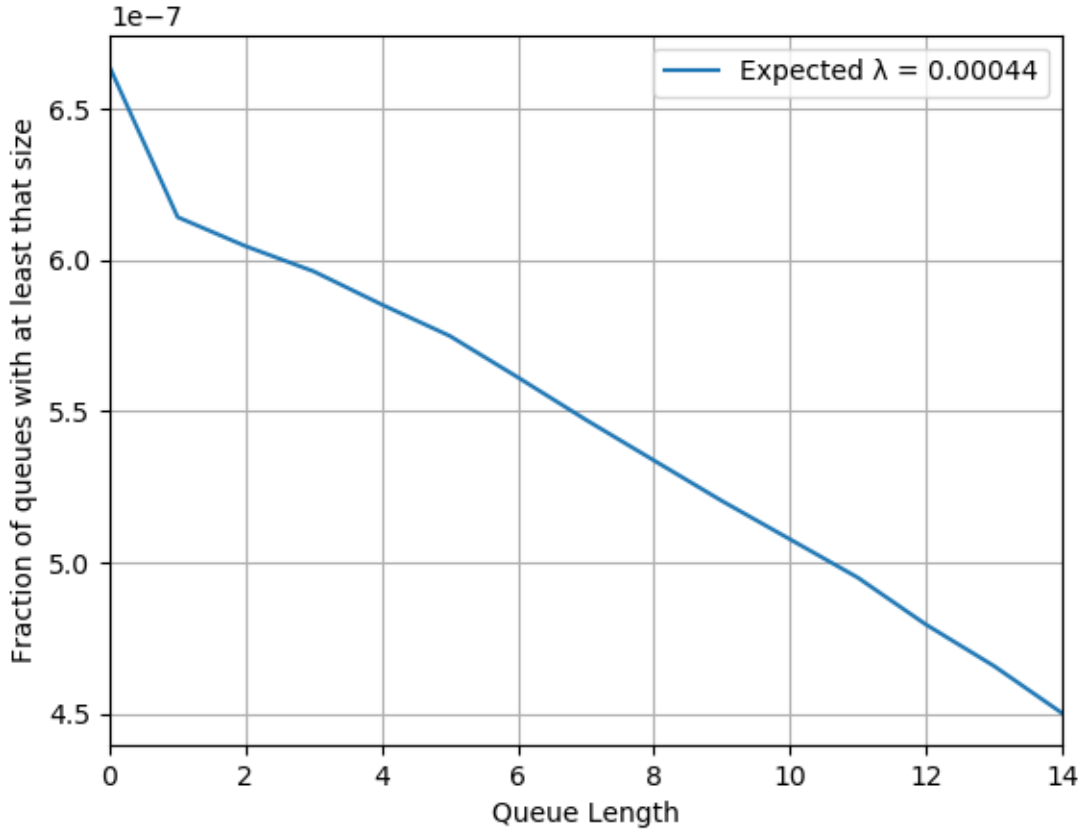

Figure 7: real data analysis

The **LAMBDA** results to be so much low! I conclude the jobs arrive in a much longer time with respect to the time server needs to complete them!