Smart Scheduling in Queues

Athanasakis Evangelos 2019030118
Fragkogiannis Yiorgos 2019030039

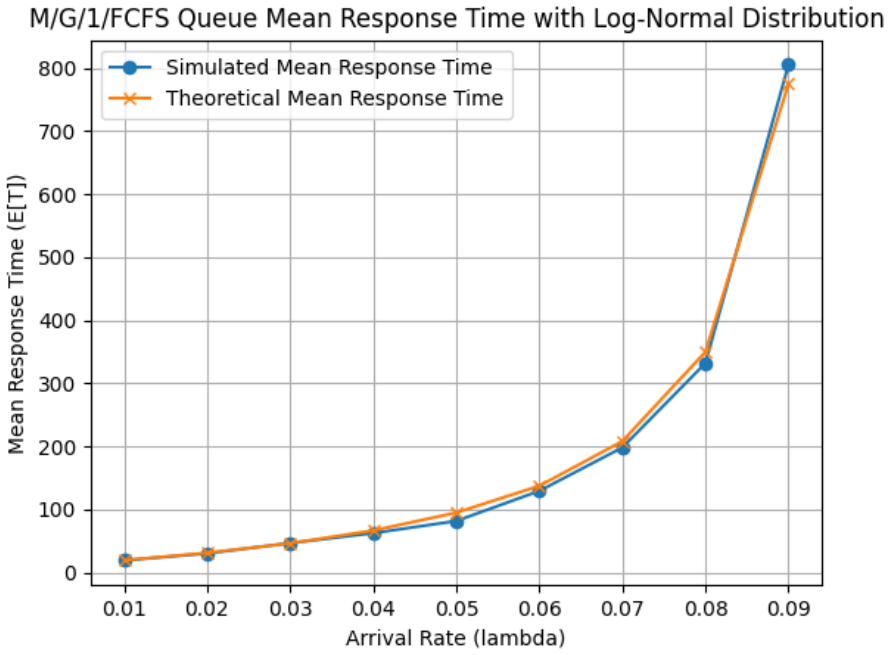
July 14, 2024

1 Introduction

In this project we have simulated in Python some standard queueing systems. In the first scenario we create an M/G/1/FCFS queue and compare its response time to the theoretical one. In the second scenario, using the SITA policy where we split the incoming jobs into two servers based on a set cutoff service time.

2 Simulating an M/G/1/FCFS Queue

Simulation of an M/G/1/FCFS queue, where jobs arrive as a Poisson process with rate λ jobs/sec (λ is a parameter we will experiment with). Jobs are distributed with an average duration of 10 seconds (E[S]). The squared coefficient of variation for the jobs is equal to 16. In order to simulate the network we choose the Log-normal distribution to generate service times for each incoming job. This distribution was used because it has a "long right tail", which means it can model the occurrence of a few very large jobs while also minimise the extreme scenarios. In other words, it can exhibit very high variability. During our experiment we plotted both the simulated response times of our queuing system and the theoretical response times for different λ values that led to different utilization ratios for the system. The following plots were generated:



For this plotted data, first of all we can observe that the simulated response time follows the expected (theoretical) mean response time:

$$E[T] = E[S] + \frac{\lambda * E[S^2]}{2 * (1 - \rho)}$$

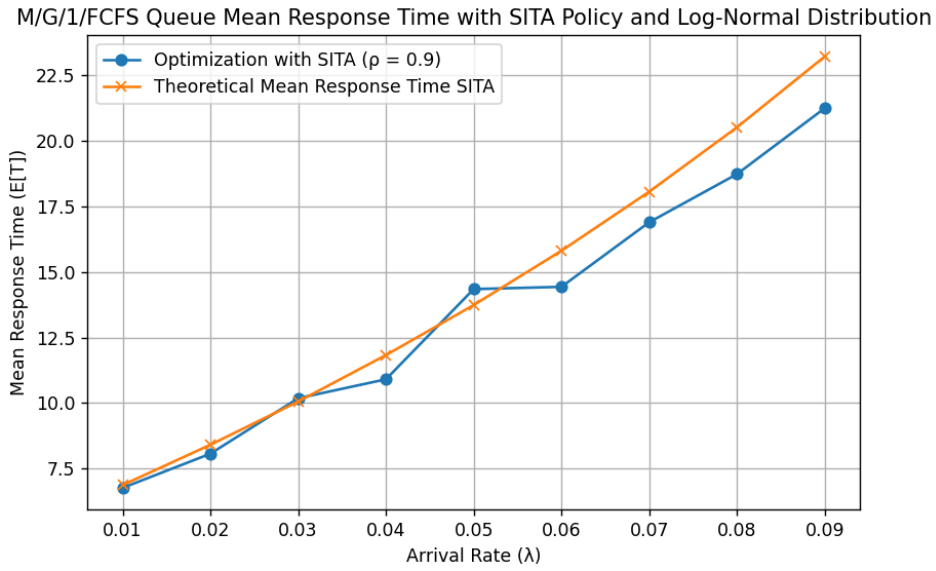
which means our model works correctly. We can see that as the utilization ρ of the system increases so does the mean response time. We can also observe that the response time grows abruptly for arrival rates greater than 0.07 arrivals/sec.

2.1 Generating the queue

We started by generating exponentially distributed arrival times for the incoming jobs. For each job that we will simulate we generate its size (service time) through the log-normal distribution. Then, we generate the service times for each job, using again the log-normal distribution that is defined by the server's capacity. The start time of each job is defined either as its time of arrival in the system or the finish time of the previous job (this way we also simulate the queuing time). By calculating the start time and the service time for each job, we can calculate the finish time and hence can calculate the response time as well ($finish\_time - start\_time$). In this way we can find the mean response time of the server.

3 Implementing SITA policy

In the following section, we split the incoming jobs to 2 servers, the cumulative capacity of which will be the same as the system described above. For our system, we split jobs based on their service time into "small" and "large". An optimised cutoff point and optimised server capacities (for the 2 new servers) were chosen to optimise the mean response time of the system (will analyze this further). The generated plot of the simulation for an optimal server capacity ratio and an optimal cutoff point for utilization $\rho = 0.5$ and $\rho = 0.9$ can be seen below:



The optimal values for the cutoff and the ratio of the μ_1 and μ_2 is the same for both $\rho = 0.5$ and $\rho = 0.9$. Their optimal values where found: cutOff $= 10$, $ratio = 0.1 \Rightarrow \mu_1 = 0.1 * \mu_2$.

3.1 Simulating M/G/1/FCFS with SITA policy

We started by generating exponentially distributed arrival times for the incoming jobs. For each job that we will simulate we generate its size (service time) through the log-normal distribution. If the job's size is greater than the cutoff point we have set for this system then we assign the job to the

server for "large" jobs, otherwise it is assigned to the server for "small" jobs. Then, after labeling the incoming jobs as "large" or "small", we generate the service times for each one of them, using again the log-normal distribution but this time with the server specific service rate (μ_1, μ_2) that is defined through the server capacity ratio. We acquire the start time as the time of arrival of a job in the system or the finish time of the previous job in the system (that way we simulate also the queuing time). By calculating the start time for each job and the service time, we can calculate the finish time and hence can calculate the response time as well (finish time - start time). In this way we can find the mean response time for each individual server. Finally, in order to get the total mean response time of the system, we calculate the cumulative mean response time for both servers.

3.2 Calculating the theoretical mean response time for our system

In order to calculate the theoretical mean response time of the system we need to use the formula:

$$E[T_i] = E[S_i] + \frac{\lambda_i * E[S_i^2]}{2 * (1 - \rho_i)}$$

In order to use this formula we must first calculate a theoretical mean service time for each server. Each mean service time calculated depends on the service rate (μ_1, μ_2) of its server. Furthermore, we must find $E[S_i^2]$. Using the variance equation we get that:

$$variance_i = E[S_i^2] - E[S_i]^2 \Rightarrow$$

$$\Rightarrow E[S_i^2] = variance_i + E[S_i]^2$$

The variance of the service time of each server is equal to

$$variance_i = scv * E[S_i]^2$$

prob_small is the probability of a job to be classified as "small" based on the respective cutOff. This probability is calculated at the simulation step and it is passed to the theoretical analysis for better comparison. λ_i are the arrival rates of each one of the 2 servers.

$$\lambda_1 = prob\_small * \lambda$$

$$\lambda_2 = (1 - prob\_small) * \lambda$$

Finally we need to find the utilization of each server ρ_1 and ρ_2, where:

$$\rho_i = \lambda_i * E[S_i]$$

Using the values calculated above we can find the theoretical mean response time for each server as:

$$E[T] = prob\_small * E[T_1] + (1 - prob\_small) * E[T_2])$$

3.3 Choosing optimal server capacity ratio and cutoff point

In order to optimise our system we created the function *optimise_environment*(*lambda_rate*). This function will go through server capacity ratio and cutoff combinations to find which provides the optimal mean response time for a given utilization. The function returns the optimal ratio and cutoff combination it has found for the system. Because this script needs to be able to run on Google Collab, the testing range of the parameters we have used is limited, but sufficient to get good results. More specifically, for the ratio we only need to check for values in (0,0.5]. We used a step of 0.1 for better performance. For the cutoff point, we first started testing with very high and very low values and ended up searching for optimality in the range of [5,50] with a step of 5, for better performance.

The optimal values for the cutoff and the ratio of the μ_1 and μ_2 where found: $cutOff = 10$**,** $ratio = 0.1 => \mu_1 = 0.1 * \mu_2$. For this cuttOff value, the "small" server takes approximately 80% of jobs while the "large" server only the 20% of jobs. The calculated cutoff point fits our simulation well enough, since its can optimally split the incoming jobs into small and large ones and thus route them to the server with the slowest and fastest service time respectively. The selected ratio is also optimal, since we can define a "fast" server (which can handle the larger jobs) and a "slow" server (which handles the bulk of the smaller jobs).

4 Results

From the plotted diagrams, we can see that by implementing the SITA policy, the estimated response time for the system drops dramatically. More specifically, for utilization equal to $\rho = 0.5$ the mean response time drops from about 100 time units to 13.5 time units, achieving a decrease of 86.5%. For utilization equal to $\rho = 0.9$ the mean response time drops from about 780 time units to 23 time units, achieving a decrease of 97.1%. So, the bigger the utilization of the system, the better the performance advantages of using the SITA policy to route the incoming jobs.