1 Introduction and experimental setup

This report describes the implementation and the detailed performance analysis realized with a closed loop test, of one web application that resembles the "IMDb" site from Amazon company. This developed system in particular is able to search for a given movie title and to show its main details like writers, directors, year of publishing, rating and actors information. The used data is a replica of the original "IMDb" database that could be found on the official website [3].

The experimental setup is composed by two machines:

1. The system under test (SUT), where the application services run, is realized in Python. The web framework for the request managing and the return of responses used is Flask [2], and the library for handling the movies' information retrieval is Polars [4]. In particular, Polars is not a true DBMS system but include some useful features like Lazy evaluation and efficient parallel thread computation for managing CSV database's files directly. The SUT used for this experiment is a PC with 8 core CPU AMD Ryzen 7 3700x @ 3.6GHz, 16GB RAM and NVMe SSD.

2. The testing machine, where Tsung evaluation software runs for simulating the behavior of the users that trying to access the application. The device used is a PC with 6 core CPU Intel Core i7-8750H @ 2.20GHz, 16GB RAM and NVMe SSD

2 System architecture and critical choices

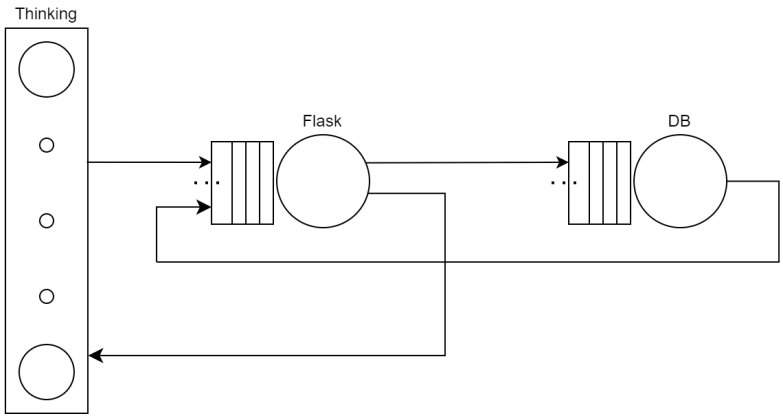


Figure 1: *System theoretical model*

The system architecture, sketched in figure 1, from the theoretical point of view can be seen as an interactive system. Some properties of this queuing network are that is always stable, it has a constant number of customers, its workload is characterized by a fixed number of jobs, and it has a thinking phase that is executed independently.

The specific components, instead, are defined as an infinite server queue (Thinking) and as two M/G/1/PS queues (Flask and Database). In particular, the Processor Sharing (PS) discipline used to model these two last stations (both physically residing in SUT machine), is considered a good approximation of the standard Operative Systems scheduling policies (indeed it assumes the context switch time is negligible with respect to the processing time) and also guarantee some theoretical results like the insensitivity to service time distribution, that is fundamental for this specific case analysis. In fact, thanks to this property it can be proved that results obtained with M/M/1/FIFO are the same as the M/G/1/PS one's even without the additional exponential assumption of service time.

The most crucial choice of this analysis is to model the system as single class to make every job statistically identical to the others. More in details, as can be seen from the *closed_loop_test.xml* file and as better explained in section 3.1, after every user request is present also a thinking time; so for the final evaluation only the average of the requests' statistics are taken into account. This allows a huge simplification of the theoretical analysis, in particular with regard to the asymptotic bound computation and the Mean Value Analysis (MVA).

$$\bar{R} = \frac{1}{\mu - \lambda}; \quad \bar{N} = \frac{\rho}{1-\rho}; \quad \rho = U = \frac{\lambda}{\mu}; \quad \bar{X} = \frac{\bar{N}}{\bar{R}}$$

Equation 1: *M/G/1/PS - M/M/1/FIFO queue formulas*

$\bar{R}$: expected response time, $\bar{N}$: expected number of users, X: system throughput, ρ: system load factor, U: system utilization, λ: arrival rate, μ: service rate

3 System evaluations

3.1 Test implementation

The closed loop workload test executed, that also allows an easier management of the system results even in saturation condition, was designed to resemble a real word case usage: indeed movie title issued by the simulated clients are sampled from a pool of 10000 queries where films are present in order to be proportional to the number of rating that they have received.
Considering the specific test implementation, where users perform many operations at SUT before leaving and alternate thinking phase with commit phase, is composed of 2 main parts:

1. Users arrival: in which users are generated according to a Poisson arrival process, with 5 seconds of inter-arrival for 5 minutes of duration and that stops when number of user is equal to 25. Notice that this phase configuration guarantees to avoid unnatural congestions (not too short) and transient effects (not too long) [1]

[1] More details can be found in *testing/closed_loop_test.xml* file

2. Cyclic requests: in which users cyclic behavior is implemented thanks to for-loop feature of Tsung. This part allows to define the steps of what must be done inside the application and their specific requests' parameters. Hence, as in one real word example, the simulated user repentantly:

   (a) Access first the site homepage

   (b) Thinks for 10 seconds what film is looking for

   (c) Search for it and wait 10 seconds once movie list is returned to visualize it

   (d) Clicks one list's link to see the complete movie details

   (e) Thinks one again

   The whole process stop after 30 minutes and, based on some other tests, this is sufficient to take the system under a steady state condition.
   Note also that the discussion has implicitly assumed that sessions never terminate and, in practice, the number of sessions being serviced will vary as existing sessions terminate and new sessions arrive [5].

3.2 Experimental results

The below images summarize the results obtained by the load test with Tsung software. In particular, from figures 3 and 4 can be observed that number of competing users after reaching value 25 remain stable until the test stop and the testing machine surely isn't the bottleneck because its mean load always stays under threshold of 20% CPU.
From image 2, that shows a plot in which the different HTTP requests are divided, can be seen that the longest and probably the most computational demanding operation is the movie details retrieval: intuitively, to accomplish this task, the server has to merge a lot of information from different files, and it requires a massive usage of the physical memory (SSD in this case), the PC's bottleneck. Whereas the non-use of dedicated DBMS due to the staticity of the data collection and due to the difficulties in importing the schema, the results are considered satisfying.

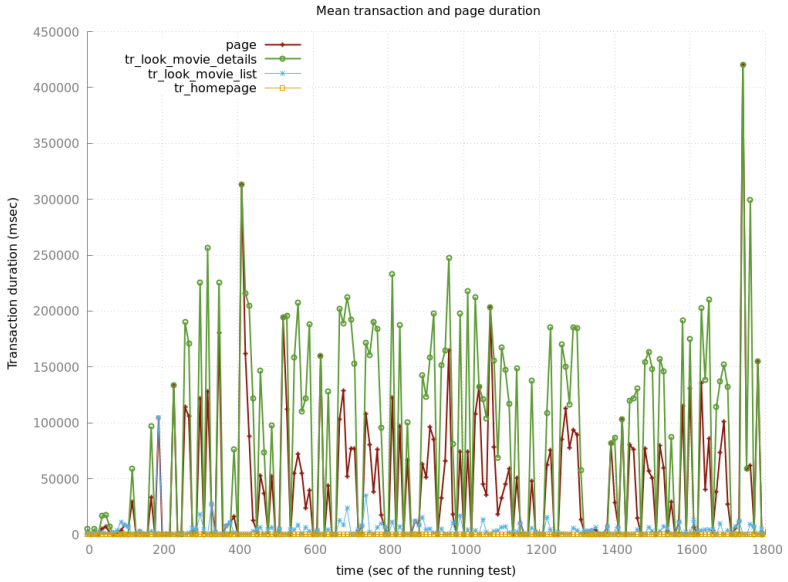


Figure 2: *Example of expected response time split by transactions' type*

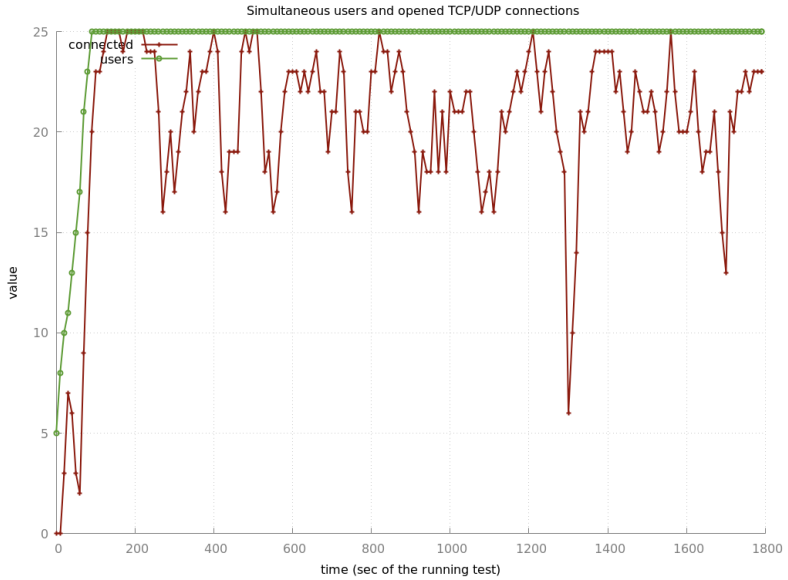


Figure 3: *Example of simultaneous users in system during the test*

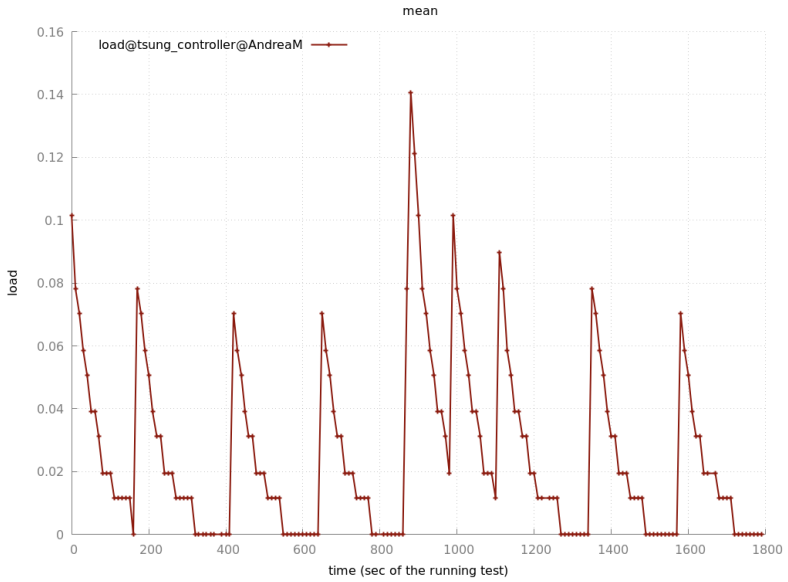


Figure 4: *Example of testing machine mean load*

Table 1 also displays some average numeric data obtained and the confidence interval at 95% using T test on 5 executions.

N° of test: 5	System $\bar{R}$ (sec)	DB $\bar{R}$ (sec)
Average system $\bar{R}$ with 25 users (sec)	80,26	79,00
Standard deviation $\bar{R}$ (sec)	2,71	2,66
Alpha	0,05	0,05
Confidence interval T test value (sec)	3,37	3,31

Table 1: *Test summary table*

In order to carry out the analysis, also the service time of the different components are needed. To obtain these values, a smart tuning strategy is adopted: some closed loop test with the same Tsung configuration but with only single user were executed, and the mean of the expected response time for each component was taken as service time value. This procedure was developed in order to maintain the correct service time proportion above each the component based on the entire system structure, exploiting the fact that in this case there wasn't any competition between customers. Obviously, some adjustments were needed to impose values producing the same performance indices of the 25 user's test, but this strategy was considered a better tuning procedures with respect to a simple trial and error.
The precise numbers chosen are ($\mu = \frac{1}{service\_time}$):

$$\mu_{db} = \frac{1}{3.5} job/s; \quad \mu_{server} = \frac{1}{0.5} job/s$$

4 System theoretical analysis

Recalling that the web application can be represented as an interactive system and its performance metrics can be defined with the equations 2, the first fundamental task of a system analysis is to spot the bottleneck.

$$\bar{T}(N) = \bar{Z} + \bar{R}(N); \quad \bar{X}(N) = \frac{N}{\bar{T}(N)}; \quad \bar{R}(N) = \frac{N}{\bar{X}(N)} - \bar{Z}; \quad \bar{V}_i = e_i \; \forall i = 1..K$$

Equation 2: *Interactive closed system formulas*

$\bar{T}(N)$ is the system time, that depends on the number of users, and is the sum of thinking time $\bar{Z}$ and expected response time $\bar{R}$. Notice that also $\bar{X}$ and $\bar{R}$ are related to N. $\bar{V}_i$ instead is the relative visit ratio of the station i and in this case it's equal to the out flow of the station i.

It should be also reminded that the bottleneck can be identified as the component with the highest service demand, $\bar{D}_b$, that as can be observed by equations 3 depends both on relative visit ratio and service time $\frac{1}{\mu_i}$

$$\bar{D}_i = \bar{V}_i \cdot \frac{1}{\mu_i}; \quad \rho_i = X_1 \cdot \bar{D}_i \text{ and } X_1 < \frac{1}{\bar{D}_b}$$

Equation 3: *Bottleneck formulas*

The required values of relative visit ratio in interactive systems can be derived solving the system's traffic equation, where for simplicity the reference station is the thinking station and its out flow e_i is fixed to 1:

$$\begin{cases} e_1 = 1 \\ e_2 = e_1 + e_3 \\ e_3 = e_2 \cdot 0.5 \end{cases} \quad \begin{cases} e_1 = 1 \\ e_2 = 2 \\ e_3 = 1 \end{cases}$$

After some computation, finally it's possible to obtain the service demands that for the different components are:

$$\bar{D}_{server} = \frac{2}{\frac{1}{0.5}} = 1$$

$$\bar{D}_{db} = \frac{1}{\frac{1}{3.5}} = 3.5$$

Therefore, the bottleneck of the system is the database.

Another crucial step is the computation of the performance metrics in the limit case, when system saturation is very low or very high. These asymptotic bounds

are therefore calculated with the following important assumptions, all fulfilled by the theoretical architecture defined in section 2:

- Stations are single server, except for the thinking station which is infinite servers. This assumption can be also extended in order to model also multiple servers, by considering a single server with speed equal to single server speed multiplied by the number of servers (approximate well the measures of system in heavy load phase but not so much in low load)
- The number of visits at a station does not affect the service time
- There is a single class of users (very complex to extends analysis to multi-class case)

Summarizing, the asymptotic bounds for an interactive system can be expressed more formally as follows:

$$\bar{R} \geq \max(\bar{D},\, N\bar{D}_b - \bar{Z}) \quad \text{and} \quad X \leq \min(\frac{\bar{N}}{\bar{D}+\bar{Z}},\, \frac{1}{\bar{D}_b}) \quad \text{where} \;\; \bar{D} = \sum_{i=2}^{K} \bar{D}_i$$

$$\bar{D} = 1 + 3.5 = 4.5$$

$$\bar{R} \geq \max(4.5,\, 3.5 \cdot N - 10) \quad \text{and} \quad X \leq \min(\frac{N}{4.5+10},\, \frac{1}{3.5})$$

Equation 4: *Asymptotic bounds for interactive system*

Thanks to these results, also the optimal number of users can also be calculated: this value represents a situation where the response time is not excessively big, while the throughput remains reasonably high. In practice, this indicates that the system is neither underutilized nor saturated.
From the graphical point of view, this number corresponds to the intersection of the bounds referring to the response time and the throughput.

$$\bar{N}_{opt} = \frac{\bar{D}+\bar{Z}}{\bar{D}_b} = \frac{4.5+10}{3.5} = 4.143$$

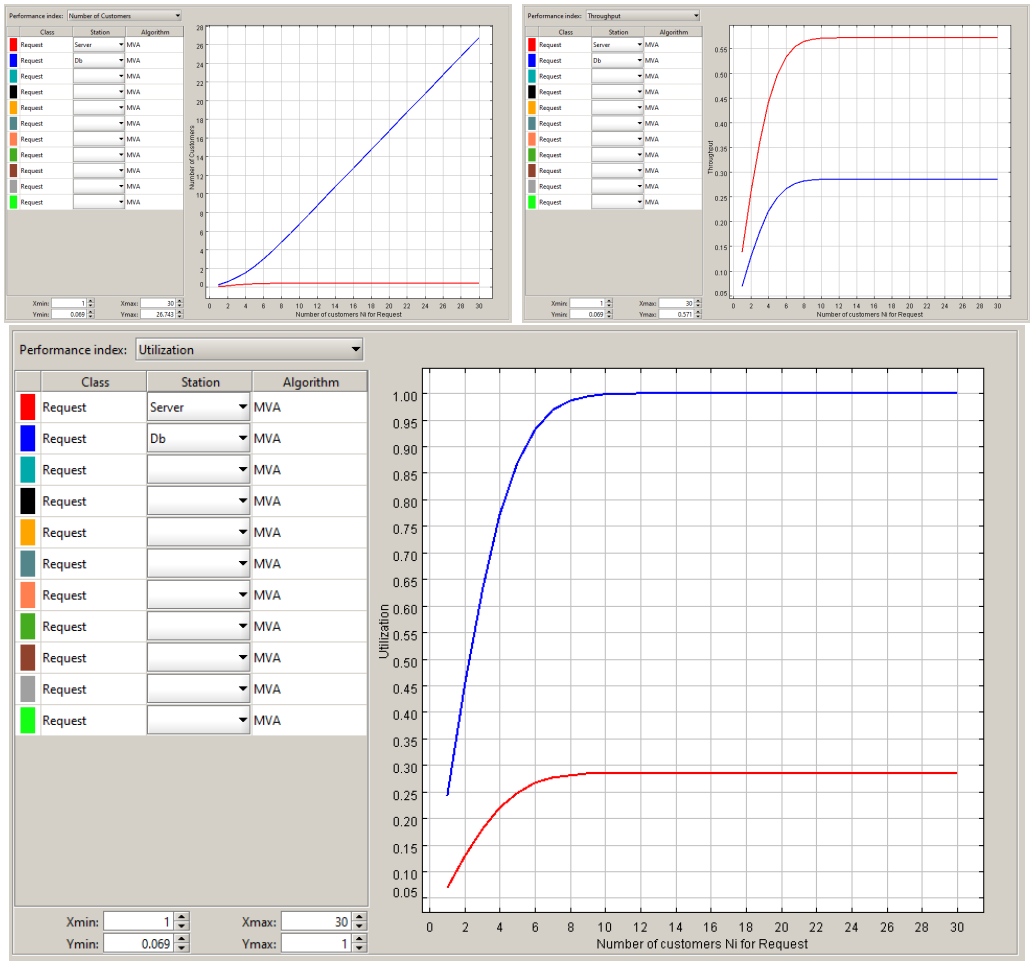
Equation 5: *Optimal number of users for the system*

Since the system theoretical architecture is a closed queuing network that respect the assumption of independent and exponential service time (see section 2, M/G/1/PS part) and the probabilistic routing is irreducible[2], it's possible to apply the Gordon-Newell theorem and the Mean Value Analysis (MVA) to it. This last statistical tool in particular is very powerful because it allows computing the mean

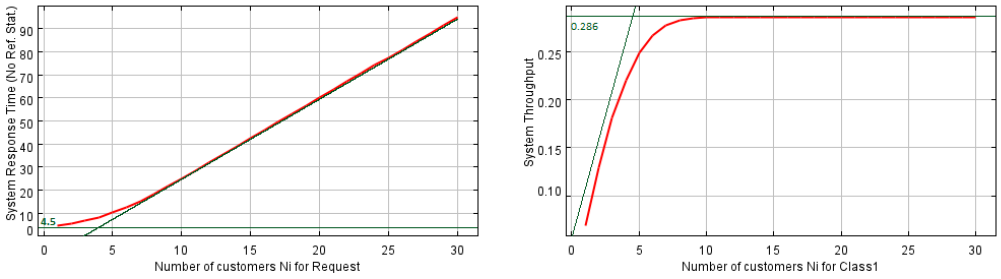
[2]If this condition is not met, the network can be split or pieces can be removed

performance indices for a closed product form queuing networks without directly compute the distribution's normalizing constant.
Figure 5 show the plots obtained from MVA and not surprisingly confirms the results obtained from the theoretical formulas and display the indices trend varying the number of customers:

- The bottleneck can be identified by observing that the database (db) is the first component that get the highest utilization. It is also important to note that users tend to accumulate in the slowest station.
- The expected response time and throughput adhere to the theoretical asymptotic bounds (as can be clearly seen in figure 5b).
- Looking at the plots, considering the indices values related to the optimal number of users, the system is observed to be in good state.
- System expected response time has an increasing trend as the number of user gets higher, as in the standard closed loop test plot



(a)



(b) *Green line: asymptotic bounds values*

Figure 5: *MVA plots of system architecture,*

The MVA was performed using the Java Modelling Tools (JMT) software [1], which speedup and automatize a lot of different various computations.

5 System architecture improvement

In order to improve the system performance and scalability, it's required to make the bottleneck scalable and perform a better resource balancing. The solution proposed

to overcome this problem is replicating the database component, splitting equally in some sense the workload seen by the single element. The new system model is illustrated in Figure 6.

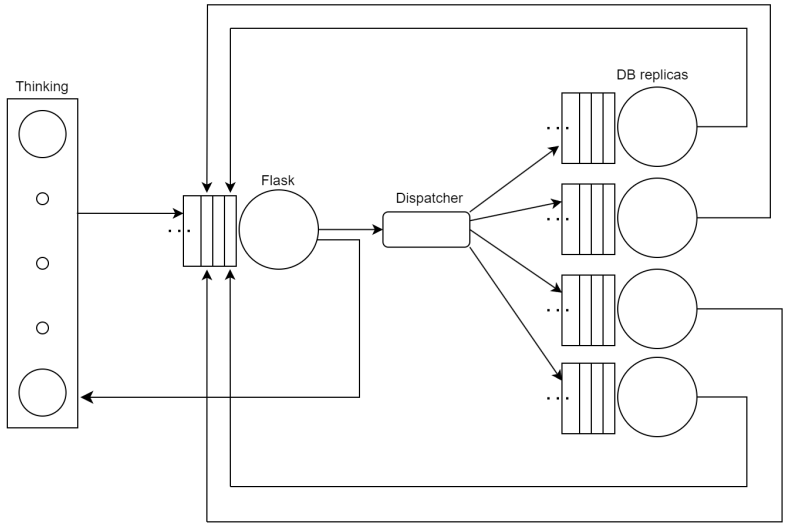


Figure 6: *New system model*

The other important concept to consider is how requests are dispatched from the server to the new database replicas. The simplest and one of the most effective choice is to employ a random dispatcher. In this scenario, where all database replicas have the same service time distribution, a random dispatcher proves to be satisfactory, also due to its stateless property.

As before, the first step of these new theoretical system analysis is to solve the traffic equations and determine the relative visit ratio for each station as follows:

$$\begin{cases} e_1 = 1 \\ e_2 = e_1 + e_3 + e_4 + e_5 + e_6 \\ e_3 = e_2 \cdot 0.5 \cdot 0.25 = 0.125 \cdot e_2 \\ e_4 = e_2 \cdot 0.5 \cdot 0.25 = 0.125 \cdot e_2 \\ e_5 = e_2 \cdot 0.5 \cdot 0.25 = 0.125 \cdot e_2 \\ e_6 = e_2 \cdot 0.5 \cdot 0.25 = 0.125 \cdot e_2 \end{cases} \quad \begin{cases} e_1 = 1 \\ e_2 = 2 \\ e_3 = 0.25 \\ e_4 = 0.25 \\ e_5 = 0.25 \\ e_6 = 0.25 \end{cases}$$

Once obtained all the values of e_i, the next step is to calculate the service demand for each station:

$$\bar{D}_{server} = \frac{2}{\frac{1}{0.5}} = 1$$

$$\bar{D}_{db1} = \frac{0.25}{\frac{1}{3.5}} = 0.875$$

$$\bar{D}_{db2} = \frac{0.25}{\frac{1}{3.5}} = 0.875$$

$$\bar{D}_{db3} = \frac{0.25}{\frac{1}{3.5}} = 0.875$$
$$\bar{D}_{db4} = \frac{0.25}{\frac{1}{3.5}} = 0.875$$

It is evident that the bottleneck has shifted. The highest service demand is now related to the Flask component $\bar{D}_{\text{server}}$. With this information, the asymptotic bounds can also be computed in the following way and can be seen un figure 7b:

$$\bar{D} = 1 + 0.875 * 4 = 4.5$$
$$\bar{R} \geq \max(4.5,\, N - 10) \quad \text{and} \quad X \leq \min(\frac{N}{4.5 + 10},\, 1)$$

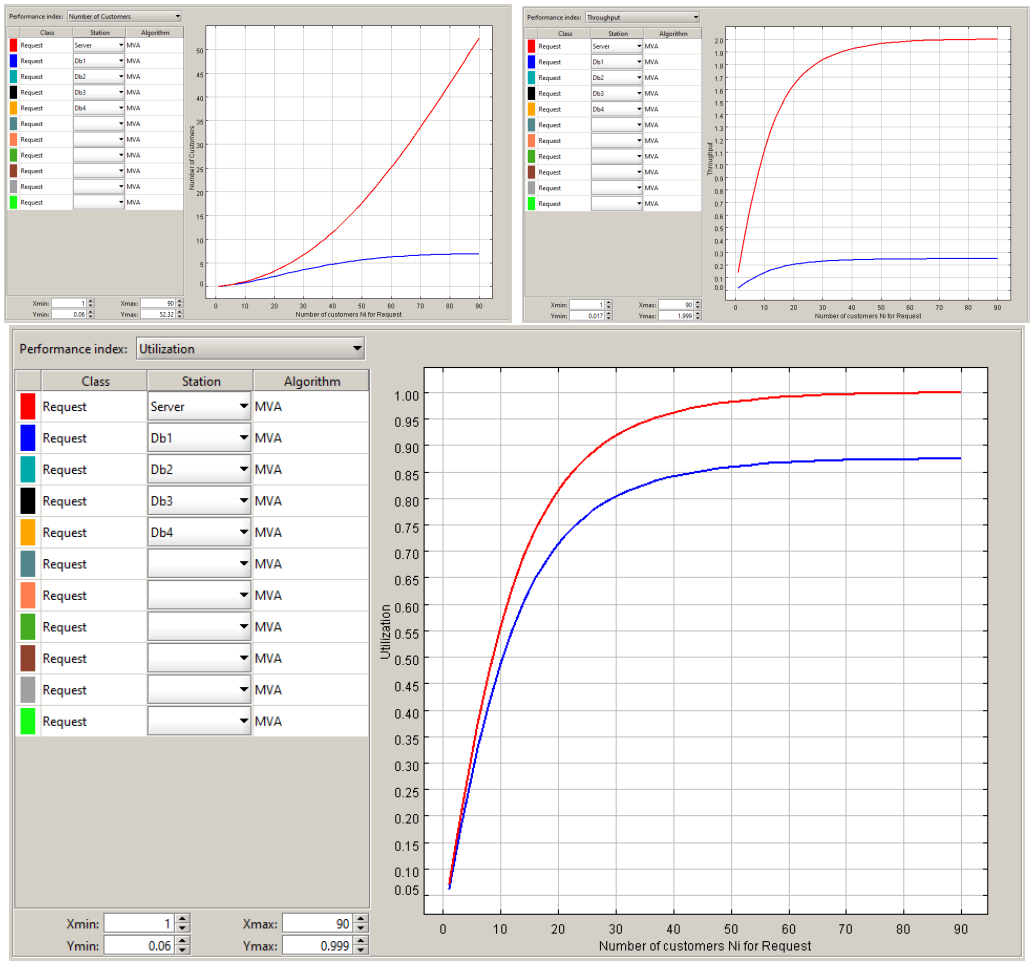
Equation 6: *Asymptotic bounds for new interactive system*

Furthermore, the new optimal number of users can be determined as:

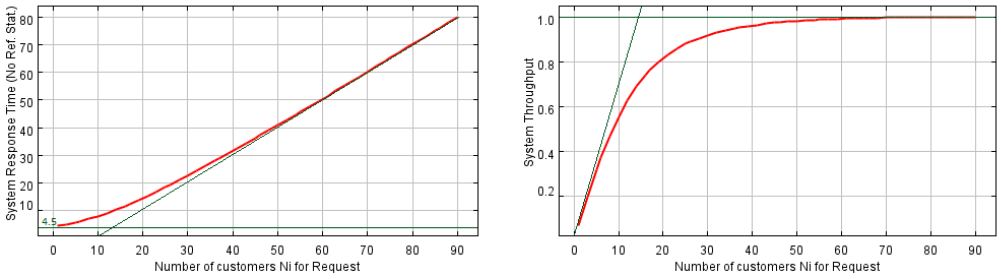
$$\bar{N}_{opt} = \frac{\bar{D} + \bar{Z}}{\bar{D}_b} = \frac{4.5 + 10}{1} = 14.5$$

Equation 7: *Optimal number of users for the new system*

This modification leads to an increment of the optimal number of users value from 4.1 to 14.5. These results are confirmed also by the Mean Value Analysis (MVA) performed using JMT. The MVA plots of the improved system architecture are shown in figure 7. Also these additional statistics provide further evidence that supports the theoretical formulas.



(a)



(b) *Green line: asymptotic bounds values*

Figure 7: *MVA plots of improved system architecture*

6 Conclusions

Summarizing, starting from the theoretical model of the web application system, the report has shown the effectiveness of a correct and formal performance analysis; in particular with the purposes of spotting the queuing network bottleneck and finding all the data needed to be able to improve the original system. It's also needed to remark the fundamental role of the assumptions used to make the computations,

and the convenience of using some software to automatize the calculation.
Some future works could be for example trying some other dispatching disciplines in order to understand if the improvement given by their usage is convenient.
In the end, the whole analysis and the related result are considered satisfying.