Chapter 2
Systems with Homogeneous Workloads

2.1 A Web Server with External Workload

tags: open, single class, Source/Queue/Sink, JMVA.

The models for analyzing the performance of a system can be developed at different levels of detail and with a single element that can represent the system as a whole or just one of its components. In spite of their high level of aggregation, models in which the resources of a system are *collectively* represented with a *single component* (i.e., the system is modeled as a black box) yields in many cases interesting results. These models can also provide useful insights for the evaluation of more complex scenarios.

To solve the model presented in this section we use the analytical tool JMVA that applies the classical Queuing Networks equations.

2.1.1 Problem Description

A capacity planning study is required to model a web server utilized for the distribution of technical documentation concerning the products of a company and accessible by a high number of users through Internet. Requests arrive at the server from the network, compete for the resources, and once executed leave the system, see Fig. 2.1a. These models are usually referred to as *open models*. The workload consists of a *single request class*. The requests have similar service demands, are independent each other and arrive to the server with exponentially distributed interarrival times. We consider a simple high-level aggregated model, i.e., a single queue station, representing the web server accessed by a request only once before leaving the station, see Fig. 2.1b. This single-station model may seem inadequate to describe a web server that has at least two resources, a CPU and a storage, that are visited many times by the requests during their execution.

G. Serazzi, *Performance Engineering*,
https://doi.org/10.1007/978-3-031-36763-2_2

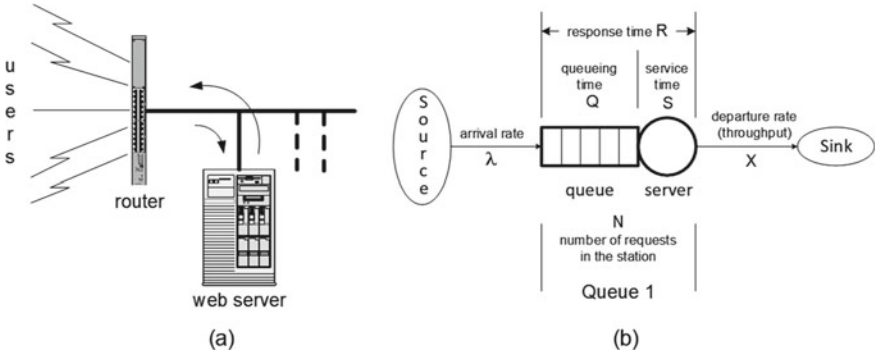


Fig. 2.1 Web server model at high level of abstraction

However, there are several problems in which single-station models yield interesting results. Among the motivations that make them useful in many situations are:

- it is common the case in which only one resource of a system is the dominant concern for the performance objectives, while the remainder components have a negligible impact on them. Modeling this critical resource clearly provides useful information about overall system performance.
- usually, one resource is much more utilized than the others (i.e., it is the *bottleneck*) and is largely responsible for the overall system performance. Models with only this resource can provide accurate predictions of the overall system performance.
- a technique to implement large models is to partition them in smaller submodels and to study them in isolation. The solutions of the submodels are then combined in order to analyze their impact on the behavior of the global model. The station used to represent collectively the stations of a single submodel is called FES, *Flow Equivalent Server*. The objective of a FES station is to introduce in the flow of requests the same delay as the submodel it represents (see, e.g., Chap. 8 of [25]).

The assumption of *single class workload* is important in many situations for the accuracy of the models. When the workload components have significant differences in resource requirements, i.e., when there are *multiple class* requests, the bottleneck may migrate among resources as a function of the fluctuations of the *mix of requests* in execution (see, e.g., [2, 3]). The effect of this migration may be dramatic for the accuracy of the results. With single class workloads the bottleneck does not switch among resources provided that all of them are *load independent*, i.e., their service time is not a function of the number of requests that are in the resource (waiting in queue and in service).

Concerning the single visit hypothesis, this should not be a concern. Depending on the abstraction level of the model, it may not be necessary to explicitly describe the load of each component r of a system at the lower levels of detail using the `Service times` S_r and the `Visits` V_r but it is sufficient to consider the *global* `Service`

`demand` D_r placed on each of them during a complete execution (i.e., $D_r = V_r\ S_r$). To reduce the number of parameters and the effort required by their measurement, we will parameterize most of the models with the `Service demands` D_r.

2.1.2 Model Implementation

We consider a simple *open model* at high level of abstraction, i.e., system-level, (see Fig. 2.1b) consisting of a single *queue* station `Queue 1`, representing the web server that is accessed by the requests generated by the `Source`. Once executed, the requests leave the *queue* station for the `Sink`. In these *open models* the number of requests in execution is *not controlled* by the system itself but depends on the characteristics of the traffic generated by the `Source` (rate and fluctuations of arrivals, service requirements). Depending on these parameters, a system can be flooded with requests whose number can suddenly grow to very high values.

Requests are assumed to be independent of each other and arrive at the server at random times. This is equivalent to saying that `Interarrival times` are *exponentially* distributed. All the requests are considered statistically equal, i.e., are indistinguishable each other, and leave the server at random times. The randomness of the departure times has as the consequence that the `Service times` S, i.e., the time requirement per visit, are *exponentially* distributed.

All requests arriving at the station can be accepted for execution, i.e., there is *enough space* to store them all that can grow indefinitely. This type of station is usually referred to as M/M/1 station (see, e.g., [36, 37]). The arrival rate is $\lambda = 0.2$ req/s, thus the average `Interarrival time` is $1/\lambda$. The average time required by a complete execution of a request is 1 s. This time usually is referred to as `Service demand` D of a request, but since the number of visits to the server in the aggregated model is one, its value coincides with the `Service time` S. Thus, for simplicity, we will use the notation S instead of D in this example. The requests are served according to their order of arrival, i.e., with a FCFS scheduling.

With the hypotheses considered, this model can be solved analytically with the classical *Queueing Networks* equations implemented in the JMVA.

2.1.3 Results

In what follows we will describe the operations required to achieve some of the **objectives** (referred to as *Obj.1–Obj.4*) of the capacity planning study.

Obj.1: implement a model of the server and compute the performance indexes with the parameters above described
In Fig. 2.2 the input parameters for the model solved with QN (Queueing Networks) equations are shown. Some of the performance indexes computed by the model are

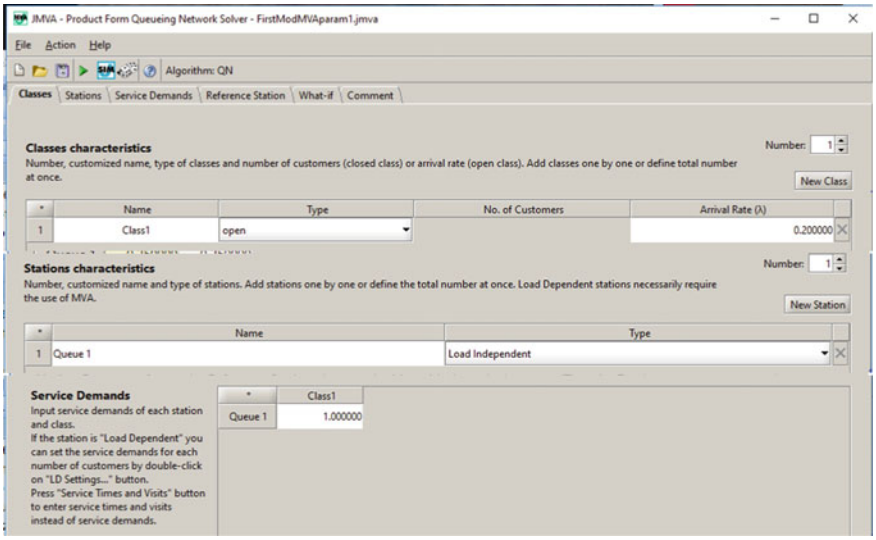


Fig. 2.2 Input parameters of the JMVA for the open model of Fig. 2.1b

shown in Fig. 2.3. The mean `number of requests N` in the server is 0.25 *req* and the mean `Response time` is 1.25 s. To check the correctness of the results we computed the values of the same indexes with the exact equations of Queueing Networks:

$$N = \frac{U}{1-U} = \frac{\lambda\, S}{1-\lambda\, S} = 0.25\, req. \qquad R = \frac{S}{1-U} = \frac{S}{1-\lambda\, S} = 1.25\ \text{s} \quad (2.1)$$

Obj.2: compute the behavior of the performance indexes when the workload increases to λ = 0.9 req/s.

A `What-if analysis` is required with `Arrival rate` as `Control Parameter` ranging from 0.2 to 0.9 req/s. In Fig. 2.4 the parameterization of the `What-if` (100 models are requested) and the behavior of two performance indexes, i.e., the `Throughput` X and the `Response time` R, are shown. Since in the model there is only the `Queue1` station, its `Throughput` and `Response time` coincide with the ones of the `System`. The linear behavior of the `Throughput` X is correct since we increase linearly the `Arrival rate` λ from 0.2 to 0.9 req/s and the models are in equilibrium, i.e., it is $\lambda = X$. The *maximum* `Throughput` of the server corresponds to the *saturation load* $\lambda^{sat} = 1/S = 1$ req/s.

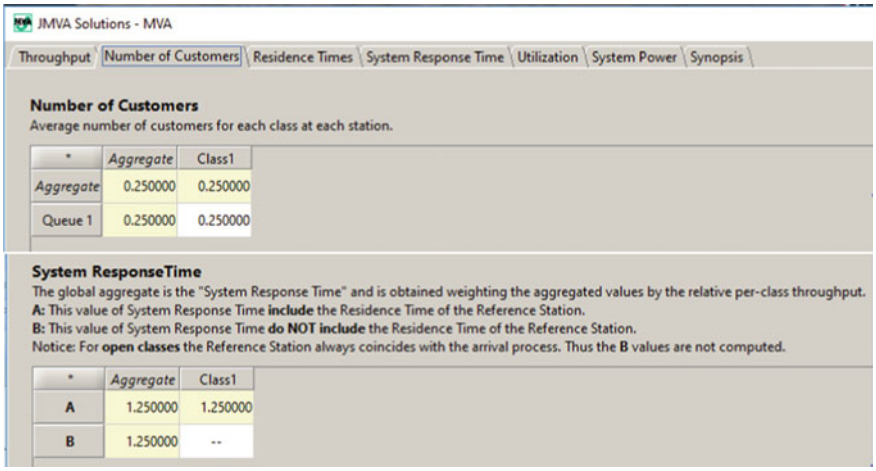


Fig. 2.3 Some of the performance indexes computed by the JMVA

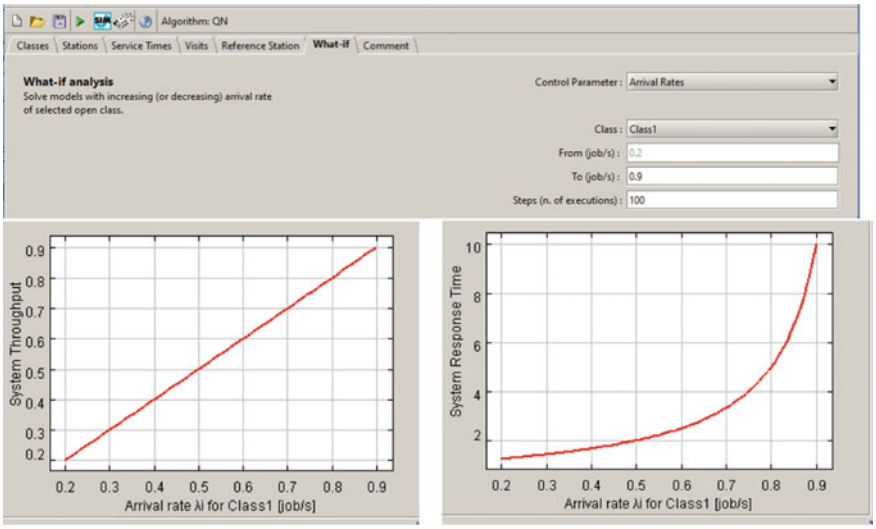


Fig. 2.4 `System Throughput` and `System Response Time` of 100 models with `Arrival rates` λ ranging from 0.2 and 0.9 req/s obtained with a `What-if` analysis

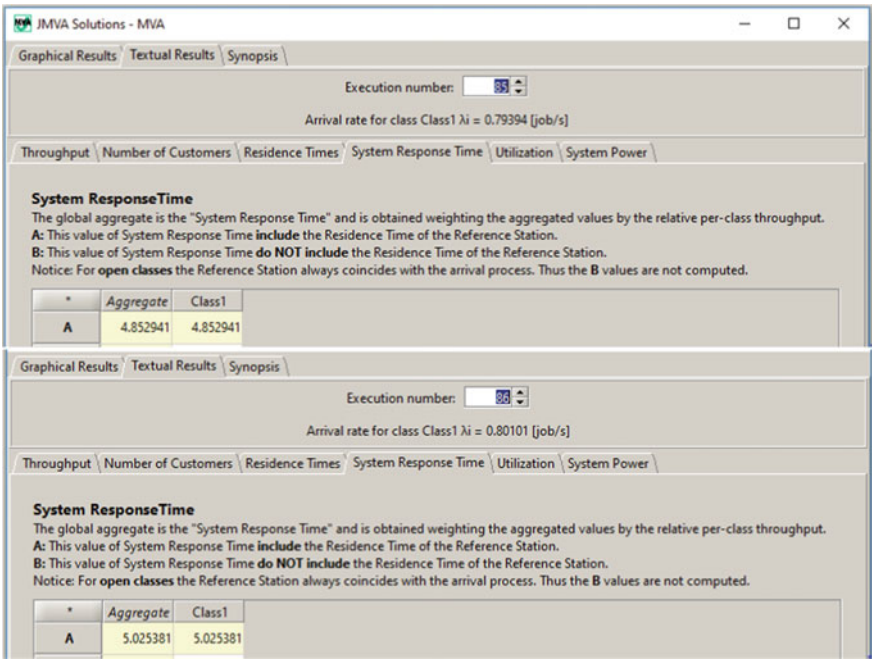


Fig. 2.5 `What-if` analysis: tabular results of the `System Response Times` corresponding to the `Arrival rates` $\lambda = 0.793$ and 0.801 req/s

Obj.3: according to the marketing department forecast, a maximum `Response time` R = 5 s can be tolerated. Compute the maximum increase of the workload that satisfy this constraint (% **with respect to the original** $\lambda = 0.2$ **req/s**).
We can use the results provided in *tabular* format from the `What-if` analysis made in the previous step. Figure 2.5 shows the `Response times` of models 85 and 86 that are just above and just below the value $R = 5$ s (4.85 and 5.02, respectively). The `arrival rates` used in the two models are 0.793 and 0.801 respectively. So we are sure that $\lambda = 0.793$ req/s satisfy the constraint. Just as a simple check, we may use Eq. 2.1 to derive the value of λ corresponding to R = 5 s and S = 1 s. We obtain $\lambda = 0.8$ req/s. Thus, the increment of the workload *tolerated* is 300%.

Obj.4: a new set of complex technical manuals are expected in the near future whose `Service demand` is assumed to be S = 2 s. What will be the `System Response Time` R with an expected arrival rate double the actual one (i.e., $\lambda = 0.4$ **req/s)**?
New values for the input parameters S = 2 s and $\lambda = 0.4$ req/s must be set. The corresponding value of `System Response Time` is R = 10 s.

2.2 A Computing Infrastructure with a Closed Workload

tags: closed, single class, Delay/Queue, JSIMg.

In this section we describe a model of a computing infrastructure with a *closed* workload (see Sect. 1.2) solved with the simulation technique. The main characteristic of this type of workload is that the number of customers in execution is constant. A new customer enter the system when a customer complete its execution.

On the basis of the assumptions made, this model could also be solved analytically with JMVA. However, we have used the simulation technique to provide a first simple example of implementing a model with a simulator. Furthermore, it should be noted that simulation is by far the most popular modeling technique used in performance engineering. Indeed, *simulators are very powerful tools* and the set of models they can implement is practically unlimited given the great generality offered in terms of characteristic of the systems and type of assumptions that can be represented.

2.2.1 Problem Description

A computing infrastructure, located in a large data center, is used to execute applications that are very critical to the company's business. This infrastructure adopts very high security techniques to control accesses that are reserved *only* to a *limited* number of authorized employees. It mainly consists of three servers: a `Web Server (WS)` and two servers (AS_1 and AS_2) dedicated to the `Application` and `Storage` functions, see Fig. 2.6a.

Due to the apps executed, the resource requirements of the user requests are similar, i.e., the workload is *single-class*. The `Service times` of the three servers have different mean values, and are assumed exponentially distributed. The probabilities (i.e., the *routing probabilities*) that the requests in output from the `Web Server` are routed to servers AS_1 and AS_2 are known. In some problems, instead of the routing probabilities, the *visits* that a request perform to each resource during its execution

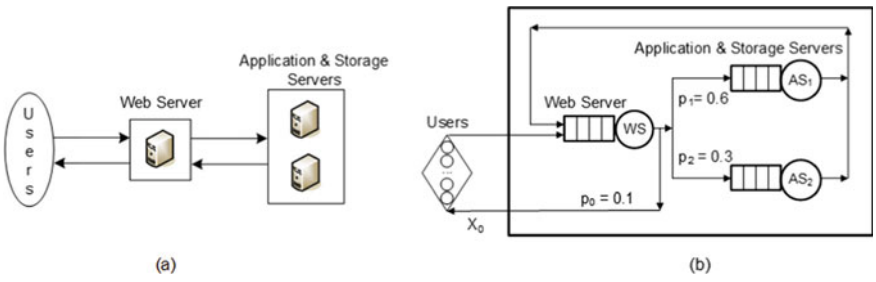


Fig. 2.6 The computing infrastructure considered (**a**) and the corresponding queueing network (**b**)

are known. These two sets of values are related each other and to derive one set from the other it is required to know the *topology* of the network. In Appendix A.1 it is described how to obtain the relationships between the *routing probabilities* and the *visits* for the topology considered in Fig. 2.6b. Assuming that when a request leaves the model it has been completely executed, i.e., that it is $V_0 = 1$, we have:

$$V_{WS} = \frac{1}{p_0} = 10 \qquad V_{AS_1} = \frac{p_1}{p_0} = 6 \qquad V_{AS_2} = \frac{p_2}{p_0} = 3 \tag{2.2}$$

Models can be parameterized with one set of values or the other. JSIMg accept *both* types of parameters. The scheduling algorithm adopted by the resources is FCFS.

2.2.2 Model Implementation

Since the *number of users* (i.e., the employees authorized to access the computing infrastructure) is *constant*, we implement a *closed model* with four stations: one *delay* and three *queue*, see Fig. 2.6b. Each user submit one request. The probabilities p_i's that after a visit to the `Web server` WS a request is routed to `App&Storage servers` AS_i are known. The index 0 is used to represent the world outside the system, and the metrics with index 0 are at *system-level*. Therefore, X_0 and R_0 represent the `Throughput` and the `Response time` of the *global system*, and p_0 is the probability that a request leaves the system as it has completed its execution. We assume that a request is routed to this path only *once* in his lifetime, so the `number of visits` V_0 that it performs outside the system is one. According to the layout of the model it is $\sum_{i=0}^{2} p_i = 1$.

The workload is generated by a station external to the system representing the `Users`, that we consider as `Reference station`. This station is used to compute the `System Response Time` R_0 and the `System Throughput` X_0 . R_0 is defined as the period of time between the instant in which a request enters the model (leaving the `Reference station`) and the one in which it leaves the model (entering the `Reference station`). X_0 is the rate of completed requests that leave the model and enter the `Reference station`. Others performance indexes are also influenced by the selection of the station that will be considered as reference (see Appendix A.1). The mean `Service time` for each Visit to servers `WS`, `AS1` and `AS2` are: $S_{WS} = 0.005$ s, $S_{AS1} = 0.020$ s, and $S_{AS2} = 0.025$ s, respectively. The *think time* of the delay station `Users` is $Z = 1$ *s*. All the values are exponentially distributed. The JSIMg model of Fig. 2.7 was solved with simulation. The *routing probabilities* of the requests leaving the `Web Server` are: $p_0 = 0.1$, $p_1 = 0.6$, and $p_2 = 0.3$, see Fig. 2.8.

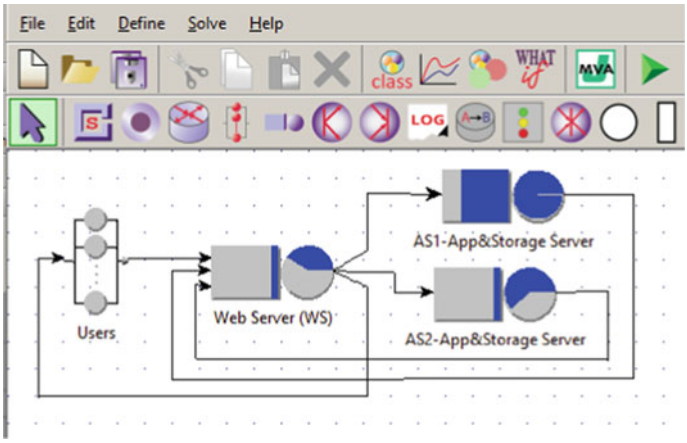


Fig. 2.7 The JSIMg model of the computing infrastrucure of Fig. 2.6b

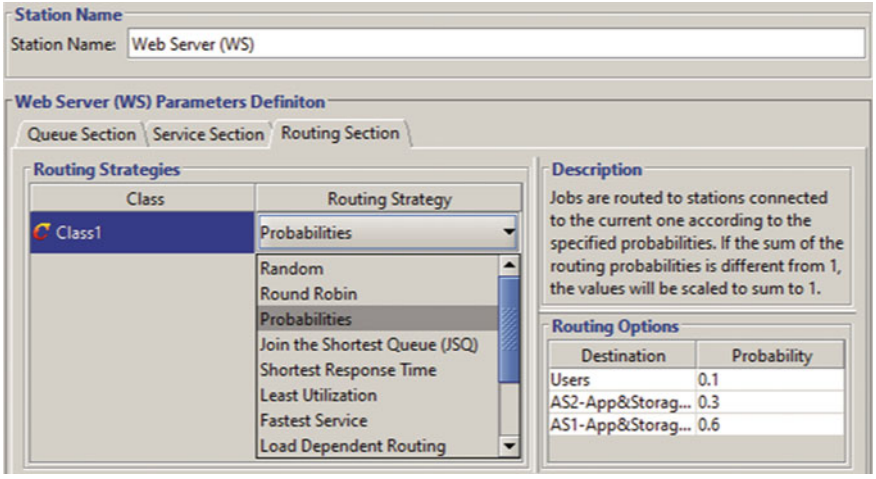
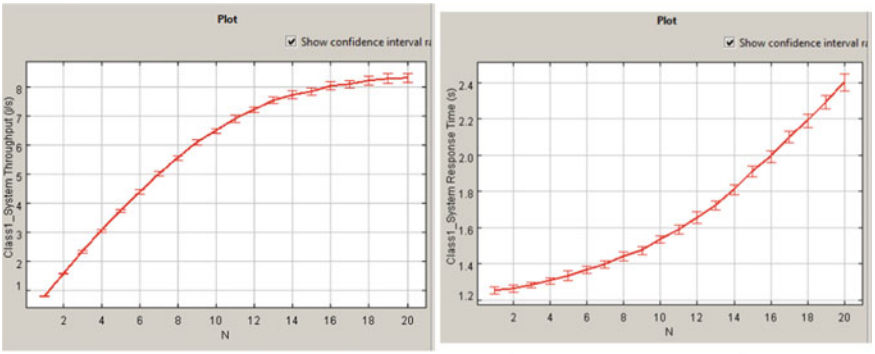


Fig. 2.8 Settings of the `Routing Probabilities` of the `Web Server WS`

2.2.3 Results

Several objectives of the capacity planning study were set. In what follows we will describe the results of some of them referred to as *Obj.1–Obj.4.*

Obj.1: Implement the model of the computing infrastructure with the parameters assigned. Investigate the behavior of `System Throughput` X_0 and `System Response Time` R_0 for the `Number of Customers` N_0 ranging from 1 to 20. Which will be the 90th percentile of R_0 with $N_0 = 20$?



(a) System Throughput X_0 (b) System Response Time R_0

Fig. 2.9 System Throughput and System Response Time versus Number of customers

A What-if analysis is performed by setting the Number of customers $N_0 = 1 \div 20$ as *control parameter*. Figure 2.9 show the behavior of System Throughput X_0 and System Response Time R_0, respectively, with respect to N_0. Please note that the R_0 values computed by JSIMg *include* the time spent in the Reference station, i.e., the Users station, that is $Z = 1$ s. For $N_0 = 20$ we have $X_0 = 8.32$ req/s and $R_0 = 2.4$ s. As N_0 increases from 1 to 20, X_0 becomes flat and tends to its horizontal upper bound, while R_0 becomes linear and tends to its lower bound which is a oblique line. These behaviors are typical of closed systems when a resource is approaching *saturation*. In the following *Objs. 2, 3* we will analyze this condition in detail.

The values of some percentiles of the System Response Times, for example the 90th or the 95th, are often requested in performance studies. Let us recall that the 90th percentile Π_{90} of a variable Y is the value below which can be found 90% of all the values assumed by Y, i.e., it is $P(Y \leq \Pi_{90}) = 0.9$. To obtain the percentile values in JSIMg it is necessary to flag the check box Stat.Res. (see, e.g., Fig. 1.8) in the window of the metrics to be collected. A CSV file with *all* collected values of the selected metric is then generated and stored. Various statistical indexes are computed by clicking on the Statistical Results button (see Fig. 1.9) in the window of the analyzed metric. Selecting Distribution as a drawing option, the values are sorted in increasing order and are grouped in intervals. For example, 300 intervals have been selected in Fig. 2.10. The percentiles corresponding to each interval are calculated and stored in a CSV file. A sample of this file for the intervals $70 \div 76$ with the corresponding percentiles (from 88.9 to 91.3) is shown in Fig. 2.11. The 90.1 percentile corresponds to $R_0 = 4.88$ s. It should be noted that if the values of a variable Y are exponentially distributed it is $\Pi_{90} \simeq 2.3 \times (mean\ value\ of\ Y)$. In our case, the values of R_0 are hypo-exponentially distributed (the coefficient of variation is $0.76 < 1$, see Fig. 2.10). Their variance is less than that of an exponentially distributed variable with the same mean. Thus, it seems correct to obtain the value of

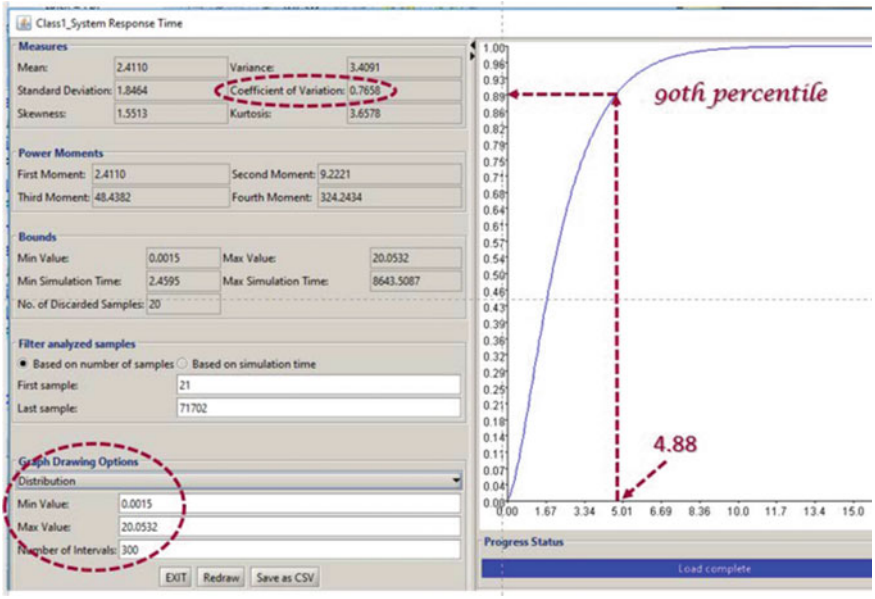


Fig. 2.10 Statistical indexes of the `System Response Times`

70	4.613391	4.68023	0.889174
71	4.68023	4.747069	0.893597
72	4.747069	4.813908	0.897754
73	4.813908	4.880747	0.901953
74	4.880747	4.947586	0.905887
75	4.947586	5.014425	0.909417
76	5.014425	5.081264	0.913198

90.19th percentile

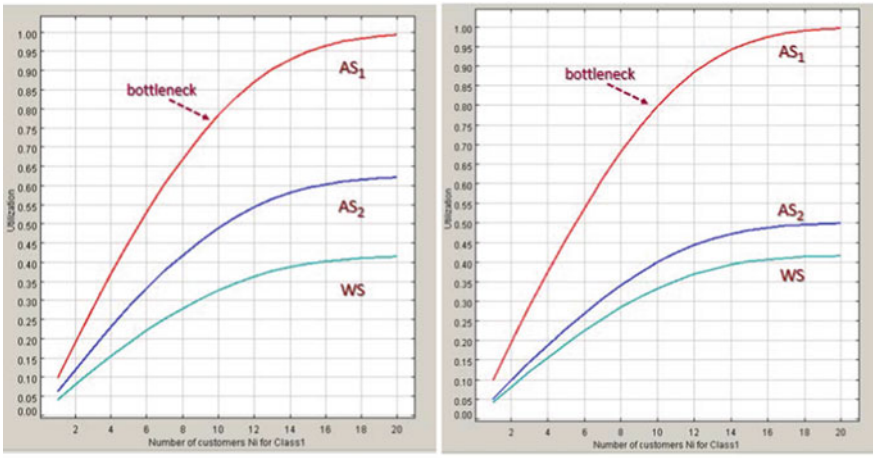
Fig. 2.11 Sample of the CSV file with the values of R_0 sorted in increasing order and subdivided into 300 intervals. The four columns refer respectively to: the *id* of the intervals, the *extremes* of each interval, and the *percentile* corresponding to the extreme with maximum value

4.88 s for the 90th percentile of R_0 which is less than 5.52 s (2.3 x 2.4), as it would be if they were exponentially distributed. By increasing the number of intervals, more detailed percentiles can be obtained.

Obj.2: To improve the computing infrastructure performance, one of the first actions that seems natural is to replace AS_2, the slowest of the `App&Storage` servers, with a new model that is 20% faster (that is, the same as AS_1). Evaluate the effects on X_0 and R_0.

The mean `Service time` of server AS_2 of the original model must be modified decreasing its value from 0.025 to 0.020 s. The model with the `What-if` for $N_0 = 1 \div 20$ *users* is executed again.

As expected, the `Utilization` of AS_2 decreased, e.g., from 61.8% to 50% with $N_0 = 20$. However, surprisingly *NO improvements* are obtained on X_0 and R_0.



(a) Utilizations of the original servers (b) Utilizations with the upgraded server AS_2

Fig. 2.12 `Utilizations` of the three servers `AS1`, `AS2`, and `WS` *versus* N_0

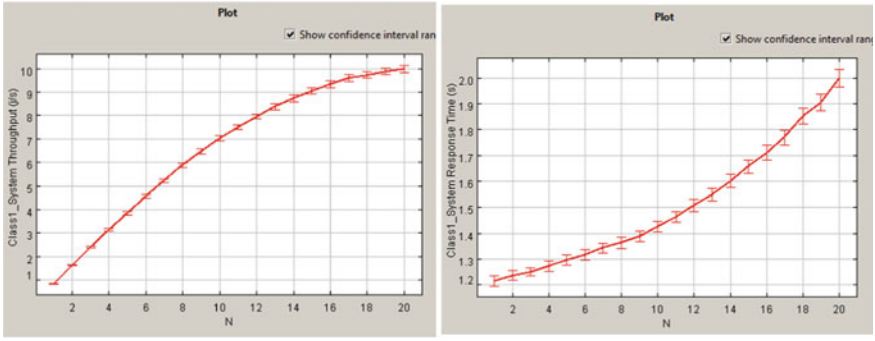
Indeed, with the new fast server AS_2 we have $X_0(20) = 8.27$ req/s and $R_0(20) = 2.42$ s while with the slow one we had $X_0 = 8.32$ req/s and $R_0 = 2.40$ s, respectively. The two values of X_0 can be considered equally likely estimates of the exact throughput value since they are both in the same 99% confidence interval. The same observation applies to R_0 (see Appendix A.2).

Analyzing the `Utilizations` of the three servers, in Fig. 2.12a with the original configuration and in Fig. 2.12b with the new AS_2, we have an answer to this *unexpected* result. From Fig. 2.12a it is possible to see that the utilizations of AS_1 and AS_2 are *unbalanced*, and that AS_1 is the *bottleneck* of the computing infrastructure despite being the *faster* of the two. Indeed, its utilization is the highest of all servers and for heavy load it is close to saturation (e.g., with $N_0 > 15$ it is $U_{AS1} > 0.95$).

This is the main motivation of the *uselessness* of the action we have done:

> *improving any station but the bottleneck do not generate any performance gain with heavy workload.* It is known that performance improvements can `only` be achieved by *reducing its contention.* Actions that reduce the load of stations `other than` the bottleneck produce *minimal improvements* (if any) only under very light workload (see *Obj.3*).

Obj.3: Given the insignificant results obtained in Obj.2, we want to evaluate the performance improvements that can be achieved by replacing the AS_1 server with a new model 20% faster (the same increase considered in Obj.2 for AS_2). We recompute the original model (Fig. 2.7) settings the mean `Service Time` of server AS_1 to a value 20% faster (from 0.020 s to 0.016 s). We then execute again the `What-if` for $N_0 = 1 \div 20$ *users* obtaining the values of X_0 and R_0 reported in Fig. 2.13. For $N_0 = 20$, with respect to the original system, X_0 increases of 20%, from



(a) System Throughput X_0 (b) System Response Time R_0

Fig. 2.13 X_0 and R_0 with the new server AS_1 20% faster

8.32 to 9.99 req/s, and R_0 drops of 17% from 2.4 to 1.99 s. The bottleneck remain the server AS_1, its utilization is 0.95 (in the original model was 0.99).

Let us remark that these *positive* results were obtained because we *improved* the station that is the *bottleneck* of the system, i.e., the server AS_1. Indeed, as seen in the previous *Obj.2*, improving other stations do not produce any significant results on performance.

Obj.4: According to the management, the number of internal employees authorized to access the computing infrastructure may increase to 40 in a semester. Which will be R_0 and X_0 with the actual configuration with $N_0 = 40$ users?
We recompute the original model (Fig. 2.7) settings the `Number of Customers` in the closed class definition window to 40. The behavior of the mean value of R_0 and of the confidence intervals during the simulation are shown in Fig. 2.14. As can be seen, the mean value of R_0 is 4.821 s obtained from the model is very close to the lower bound 4.8 s given by $N_0 D_{max} = N_0 V_{AS1} S_{AS1} = 40 \times 0.12$. The $X_0(40)$ is 8.325 req/s, very close to its upper bound $1/D_{AS1} = 1/0.12 =$ 8.333 req/s (see Sect. 2.3).

2.3 Equivalent Model with Service Demands

tags: closed, single class, Delay/Queue, JSIMg

In this section we describe a model, solved with JSIMg, parameterized with `Service demands`. This model is *equivalent* to the one solved in the previous section using the `Visits` and `Service times` of the stations. The granularity at the *system-level* is here adopted compared to that at the *station-level* adopted in the previous model.

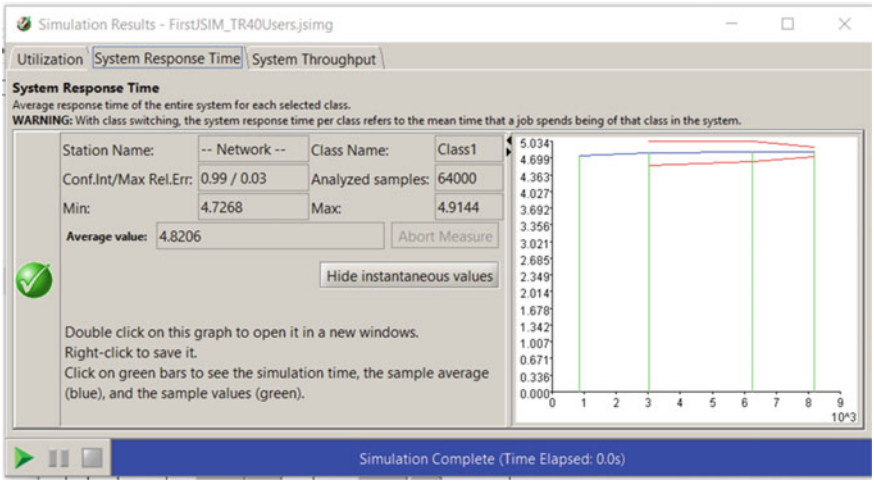


Fig. 2.14 `System Response Time` of the computing infrastructure with the initial configuration and $N_0 = 40$ users

2.3.1 Problem Description

The models can be designed at different levels of granularity (see Chap. 1), from the single component (station) to the entire system. In Sect. 2.2 we implemented a station-level model using the `Routing probabilities` p_{ij}, i.e., the probability that a request in output from station i is routed to station j. We have also described how to obtain from the $p_{ij}s$ the number of `Visits` V_r that a request makes to each station during its complete execution (see Appendix A.1). In the models parameterized at this level of detail we may compute all the performance indexes describing the behavior of each station, including its `Throughput` and `Response time`. However, measuring the p_{ij}, or the V_r, is difficult or in some cases impossible. A parameter that is often used is the `Service demand` D_r of a request to station r, which represents the *total amount* of `Service time` that a request requires from station r to complete its execution. The D_r values may be obtained by multiplying the `Service time` S_r required by one visit to the number of `Visits` V_r that a request makes to station r during its execution, i.e., $D_r = S_r\ V_r$. There are many motivations that make the models parameterized with `Service demands` so *popular*. Among them are:

- the *limited effort* required to obtain the mean values of `Service demands` D_r from *measurements*. Indeed, the system log file usually shows the D_r values for every request executed. Recall that several executions are needed to have a *reliable estimate* of the mean values of D_r and the *confidence intervals* of the measured values must be computed, (see Appendix A.2 and, e.g., [36, 37]). The $D_r s$ can

also be obtained dividing the *busy time* B_r of a resource by the number C_0 of user requests executed;

- the models that use `Service demands` are less expensive to parameterize than more detailed models using `Service times` and `Visits` as the number of parameters required is significantly lower. The measurement of even one more parameter may require a non-trivial effort;
- a large part of queueing networks considered in performance studies are of the *separable* type (see Sect.1.2 and, e.g., [4, 25]) and can be solved by knowing *only* the values of D_r and not those of its single factors V_r and S_r. The paths followed by requests between the resources can be unknown, only the *global amount* of service time required to each resource (i.e., the `Service demands`) counts. According to this property, for example, a model in which a job make 1000 visits to a station whose service time is 5 milliseconds, is equivalent to one in which the job make a single visit to that station requiring 5 sec of service time. Clearly the equivalence must be applied also to all the other stations of the system. The performance indexes obtained with this equivalent model are the same as the more detailed model with regard to the indexes at the system-level, i.e., `System Throughput` and `System Response Time`. The same is true also for the `Utilization` and the `Residence time` of the single stations. However, in this case, due to the *high level* of granularity adopted, we *cannot compute* the `Throughput` and the `Response time` at the station-level (their computation requires the `Visits` and `Service times` of each station).

Based on the described advantages, when possible, the models are preferably parameterized in terms of `Service demands` D_r instead of `Visits` and `Service times`. Clearly, with this high level of granularity we lose the structural similarity with the considered system, but the models are easier to implement, the solution algorithms are faster, and the performance indexes that can be computed (not all, but almost) are correct.

2.3.2 Model Implementation

To illustrate the practical applicability of the `Service demands` we consider again the closed model solved in Sect.2.2 using the `Routing probabilities` p_{ij} and the `Service times` S_r, see Fig. 2.6b. In this Section we implement a new version of it using `Service demands`. From the p_{ij}s it is possible to derive the `Visits` to the three servers $V_{WS} = 10,\ V_{AS1} = 6,\ V_{AS2} = 3$ (see Appendix A.1) and knowing the `Service times` $S_{WS} = 0.005$ s, $S_{AS1} = 0.020$ s and $S_{AS2} = 0.025$ s, we can compute the `Service demands` ($D_r = V_r\ S_r$) $D_{WS} = 0.050$ s, $D_{AS1} = 0.120$ s, $D_{AS2} = 0.075$ s.

The implemented model of Fig. 2.15 consists of three servers, having as `Service times` the D_r, which are visited only once during the execution of a user request. The structure of this new model is clearly simpler than that of Fig. 2.7.

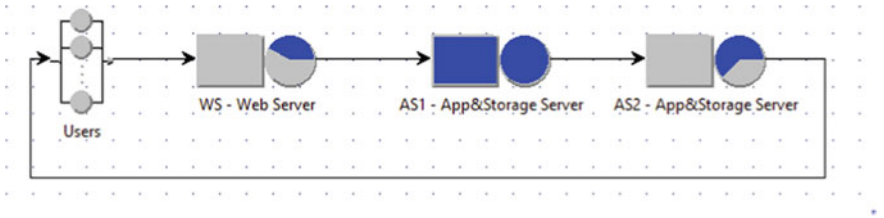


Fig. 2.15 Model parameterized with `Service demands` equivalent to that of Fig. 2.7 which uses `Visits` and `Service times`

2.3.3 Results

We solve the model of Fig. 2.15 using the `What-if` analysis with $N_0 = 1 \div 20$ `customers` as `control parameter`. The performance measures obtained are practically the same, i.e., they lie in the same 99% confidence interval of the corresponding indexes obtained with the model of Fig. 2.7. Table 2.1 compares the values of some performance indexes computed by the two models with $N_0 = 20$ *customers*.

It should be noted that with the high level parameterization, and the consequent simplified layout of the equivalent model, it is NOT possible to compute the `Throughput` and the `Response time` of *each station*. Indeed, at the system level, we do not model the `Visits`, thus *only* the parameters at high level of aggregation can be computed for each station, i.e., the global `Utilization` and the `Residence time`. The values inside the boxes in Table 2.1 emphasize that in the equivalent model the three servers have the *same* `Throughput` (measured in user requests per time unit), that coincide with the `System Throughput` X_0. This is because the three servers in this model are visited only once, requiring the `Service demand` D_r to each of them, are connected in series. Thus, we may compute only the `Residence times` Rd_r of the servers and *not* their `Response times` R_r (since it is $Rd_r = V_r \, R_r$).

Table 2.1 Performance metrics for $N_0 = 20$ *customers* obtained from the two equivalent models parameterized at different levels of granularity (`Visits and Service times` and `Service demands`, respectively)

Parameters used	Performance metrics										
	R_0	X_0	U_{WS}	U_{AS1}	U_{AS2}	Rd_{WS}	Rd_{AS1}	Rd_{AS2}	X_{WS}	X_{AS1}	X_{AS2}
`Visits and Service time` (Fig. 2.7)	**2.4**	**8.32**	0.419	0.993	0.618	0.082	1.14	0.184	82.13	49.32	24.74
`Service demands` (Fig. 2.15)	**2.4**	**8.31**	0.415	0.992	0.622	0.083	1.14	0.185	8.31	8.31	8.31

The *operational analysis laws* can be applied also to models parameterized with `Service demands`. For example, using the *Forced Flow law* $X_r = X_0\, V_r$, the *Utilization law* becomes

$$U_r = X_r\, S_r = X_0\, V_r\, S_r = X_0\, D_r \tag{2.3}$$

From Table 2.1 it is possible to see that Eq. 2.3 is verified with the metrics obtained from the model parameterized with the D_r. For example, for the server WS it is $U_{WS} = 8.31 \times 0.05 = 0.415$ that coincides with the measured value of U_{WS}. Note that the results of the two models may not coincide exactly as we are in simulation and we know only the confidence intervals of the computed variables. We may verify also that, according to Eq. 2.3, the ratio of the U_r coincides with the ratio of the D_r:

$$\frac{U_{WS}}{U_{AS1}} = 0.418 \simeq \frac{D_{WS}}{D_{AS1}} = 0.416 \qquad \frac{U_{AS1}}{U_{AS2}} = 1.594 \simeq \frac{D_{AS1}}{D_{AS2}} = 1.6 \tag{2.4}$$

The *Little law* applied to resource r using the `Residence times` becomes:

$$N_r = X_r\, R_r = X_0\, V_r\, R_r = X_0\, Rd_r \tag{2.5}$$

It should be recalled that the `System Response Time` R_0 provided by JSIMg comprises the time spent by a user request in the `Reference station`, that in our model is the `Users` with $Z = 1$ s. So, it is $R_0 = Rd_{WS} + Rd_{AS1} + Rd_{AS2} + Z = 2.409$ s. Applying *Little law* at the system-level we have: $N_0 = X_0\, R_0 = 8.31 \times 2.409 = 20\, customers$, as expected.

From the analysis of the D_r we can derive that the server `AS1` is the *most utilized* of the resources since $D_{AS1} = 0.120$ s is the *largest* of the `Service demands`. As N_0 increases it will be the first resource to saturate, i.e., it becomes the *bottleneck* of the system, limiting the `System Throughput` to $X_0 \leq 1/D_{max} = 8.333$ req/s. With $N_0 = 20\, customers$ we obtained $X_0 = 8.31$ req/s (see Table 2.1) since `AS1` is not completely saturated (it is $U_{AS1} = 0.992$).

2.4 Optimal Operating Point of a Server

tags: open, single class, Source/Queue/Sink, JSIMg.

We describe how to identify the optimal operating condition of a system that is characterized by the highest `Throughput` with the shortest `Response time`. A system in this condition, referred to as *optimal operating point*, operates with *maximum of efficiency*, that is, it maximizes its productivity by introducing the minimum delay. We consider a simple model of a system that is solved using both analytical techniques (Queueing Networks) and simulation techniques (JSIMg model).

2.4.1 Problem Description

Identifying the *optimal operating point* of a digital infrastructure is a problem that IT managers face in their daily lives. Nowadays, this task is becoming more and more important as the size of data centers and clouds continues to grow in the number of servers, applications and users. The basic idea is that a small profit on a single server can translate into a large profit when evaluated on hundreds of servers with thousands of users.

Clearly, depending on the context considered, the notion of *optimal operating point* assumes various definitions that translate into different actions. For example, it can refer to the operating conditions that minimize the energy required to execute a workload while meeting the performance goals, or to the operating point that satisfies the SLA (Service Level Agreement) by minimizing the number of allocated servers.

To simplify the description, let's consider a single server that we assume is operating under the *optimal* conditions when its `Throughput` X is *maximized and* its `Response time` R is *minimized*. The load corresponding to this *optimal condition* will be referred to as *optimal load*.

In this simple case, when the goal of the performance study is to identify the load that *maximizes* the `Throughput` X or the one that *minimizes* the `Response time` R, the answers are easily provided. In fact, a resource generates the *maximum* X when it is *saturated* and provides the *minimum* R when only one request is in execution, that is, there is *no contention*. However, when X and R are to be compared at the *same time*, a new metric must be used that considers the trade-off between the two. To this end, below we will consider the `System power` Φ, a metric introduced in [19] and extensively studied by Kleinrock [23, 24], defined as

$$\Phi = \frac{X}{R} \tag{2.6}$$

The behavior of Φ may be considered in some way related to that of the *Quality of Service*. Indeed, an increase in `Throughput` or a decrease in `Response time` increases the `System power`, that may be considered in the *SLA* as indicator of the *Quality of Service* delivered to the users.

In the next section we consider a system consisting of a single server modeled with a `Queue` station (see, e.g., Fig. 2.1) that execute a homogeneous workload (single class) with `Interarrival times` and `Service times` exponentially distributed.

For this simple case, we describe the analytical computation of the optimal load, and then we implement the correspondent model with JSIMg. The analytical derivation of Φ for more complex systems is not easy to obtain. However, it should be emphasized that the JSIMg provides the `System Power` behavior for *all simulated models*, regardless of *their complexity*. Φ is one of the metrics available in the tool.

2.4.2 Model Implementation

In this section we address the problem of the identification of the optimal operating point of a single server executing requests with `Interarrival times` and `Service times` exponentially distributed. For its simplicity, we initially present the analytical solution of this model that was derived in [23]. Below, we describe the simulation results obtained with the corresponding model implemented with JSIMg.

The `System Power` Φ for the considered M/M/1 model is

$$\Phi = \frac{\lambda}{R} = \frac{\lambda\,(1-\lambda S)}{S} \tag{2.7}$$

where λ is the arrival rate and S is the mean service time of the requests. To find the load λ^{opt} that *maximizes* Φ it is sufficient to set to zero its first derivative Φ' with respect to λ and derive the value of λ^{opt}. We have: $\Phi' = (1/S) - 2\lambda^{opt} = 0$, thus it will be

$$\lambda^{opt} = \frac{1}{2}\,\frac{1}{S} \tag{2.8}$$

Therefore, according to Eq. 2.8, the *optimal operating point* is obtained with a load λ^{opt} equal to *half* of the one corresponding to the `maximum Throughput` 1/S. The R^{opt}, U^{opt} and N^{opt} are

$$R^{opt} = \frac{S}{1-\lambda^{opt}S} = 2S \qquad U^{opt} = \lambda^{opt}S = 0.5 \qquad N^{opt} = \frac{0.5}{1-0.5} = 1\ req$$

Let us remark that R^{opt} is *twice* its minimum value S, the server is utilized at 50% and the mean number of customers in the server is 1, 0.5 in queue and 0.5 in execution. In this optimal condition, an arriving request has 50% of probability to find the queue empty and the server idle. Figure 2.16 shows the behavior of Φ of a server with $S = 1$ s.

An interesting observation can be obtained from the analysis of Fig. 2.16b which shows that `System Power` is maximized with the load corresponding to the *tangent point* of the straight line $R = m\lambda$ from the origin to the `Response time` curve. Equating to zero the discriminant of the equation that compute the intersection of the two functions we obtain $m = 4S^2$. Replacing it in the equation of the intersection we obtain $\lambda = 0.5(1/S)$, which has already been found as *optimal load* in Eq. 2.8.

This property allows to define the optimality condition as the one corresponding to a load λ^{opt} for which the relative increase of the `Throughput` X is equal to that of the `Response time` R. When it is $\lambda < \lambda^{opt}$ it will be $dX/X > dR/R$ therefore an increase in λ increases the `Throughput` more than the `Response time`, so the gain is higher than the loss. The opposite situation occurs when it is $\lambda > \lambda^{opt}$ since in this condition an increase of λ generates losses greater than the gains, i.e., it is $dX/X < dR/R$.

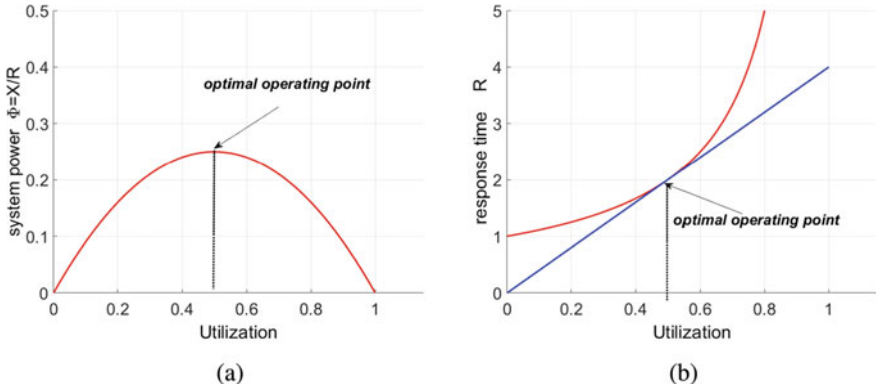


Fig. 2.16 `Power` Φ (**a**) and `Response time` R (**b**) versus `Utilization` with S = 1 s

This definition of the optimal operating point is valid for *any* `Throughput` and *any* `Response time` functions. The metric `System Power` can be very useful to implement load balancing policies based on machine learning and as a target function in autoscaling components.

The implemented JSIMg open model consists of three stations: `Source`, `Queue`, `Sink` (see, e.g., Fig. 2.1). The `Service times` of the Queue are exponentially distributed with mean $S = 1$ s. The `Interarrival times` of the requests generated by the `Source` have exponential distribution with arrival rates ranging from 0.1 to 0.9 req/s. Figure 2.17 emphasizes the selection of `System Throughput`, `System Response time`, and `System Power` indexes (the last two are shown in the graphs of Fig. 2.18).

2.4.3 Results

A `What-if` analysis is used with the `arrival rate` λ of `requests` as *control parameter* with values ranging from 0.1 to 0.9 in 9 models. As can be seen from Fig. 2.18, the values of R and Φ corresponding to the optimal load $\lambda^{opt} = 0.5$ req/s are very close to the exact ones obtained analytically ($R = 2$ s, and $\Phi = 0.25$). The confidence intervals are very small.

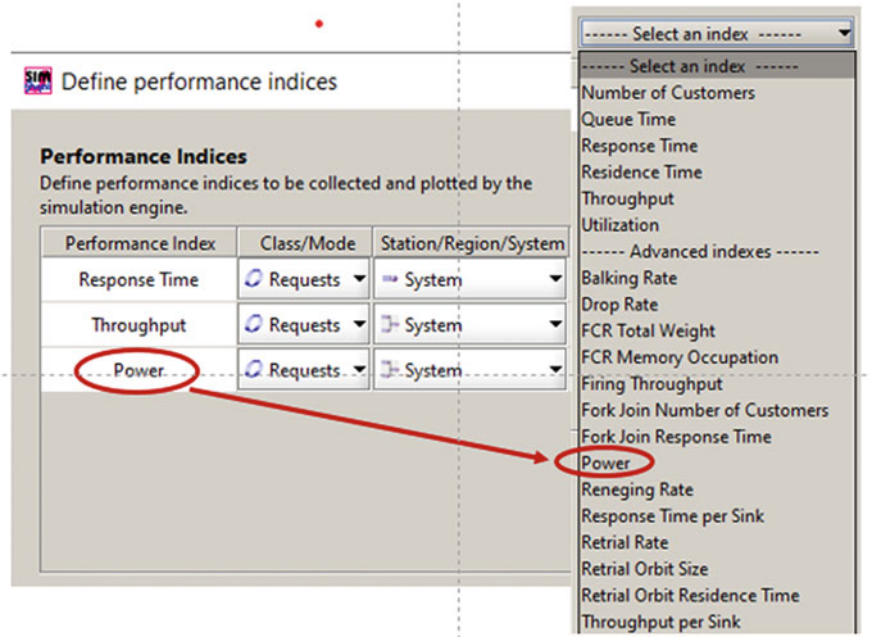
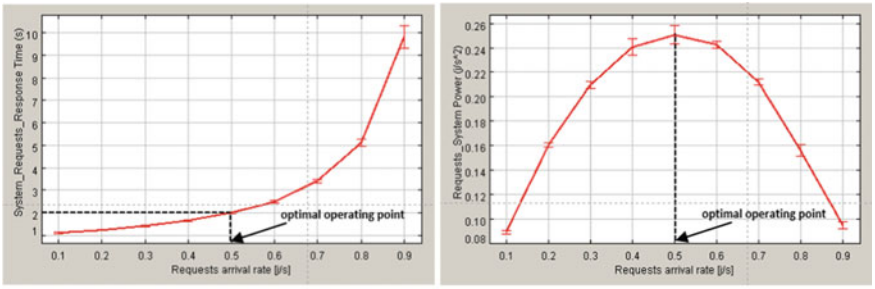


Fig. 2.17 Selection of the `System Power` index



(a) (b)

Fig. 2.18 `Response time` R and `System Power` Φ obtained with JSIMg

2.4.4 *Limitations and Improvements*

- *High variability of* `Service times`: In [24] it is described the analytical derivation of Φ in simple models with exponential `Interarrival times` and high variability of `Service times` (for M/G/1 stations).

- *Models with complex structure*: Although we have described the use of `System power` in models with a single resource, it should be clear that all the considerations made can be applied also to open and closed models with *more complex* structure and multiple resources. The identification of the analytical expression of Φ in these models is clearly not so simple as the one of Eq. 2.7.
- *Availability of Power index*: JSIMg compute and plot the values of Φ for all the simulated models, *independently* of their complexity.