UNIVERSITÀ DI PISA

Computer Engineering

Performance Evaluation Computer Systems and Networks

Multi-Programmed Server

STUDENTS:

Fernando De Nitto

Nicola Ferrante

Simone Pampaloni

Academic Year 2019/2020

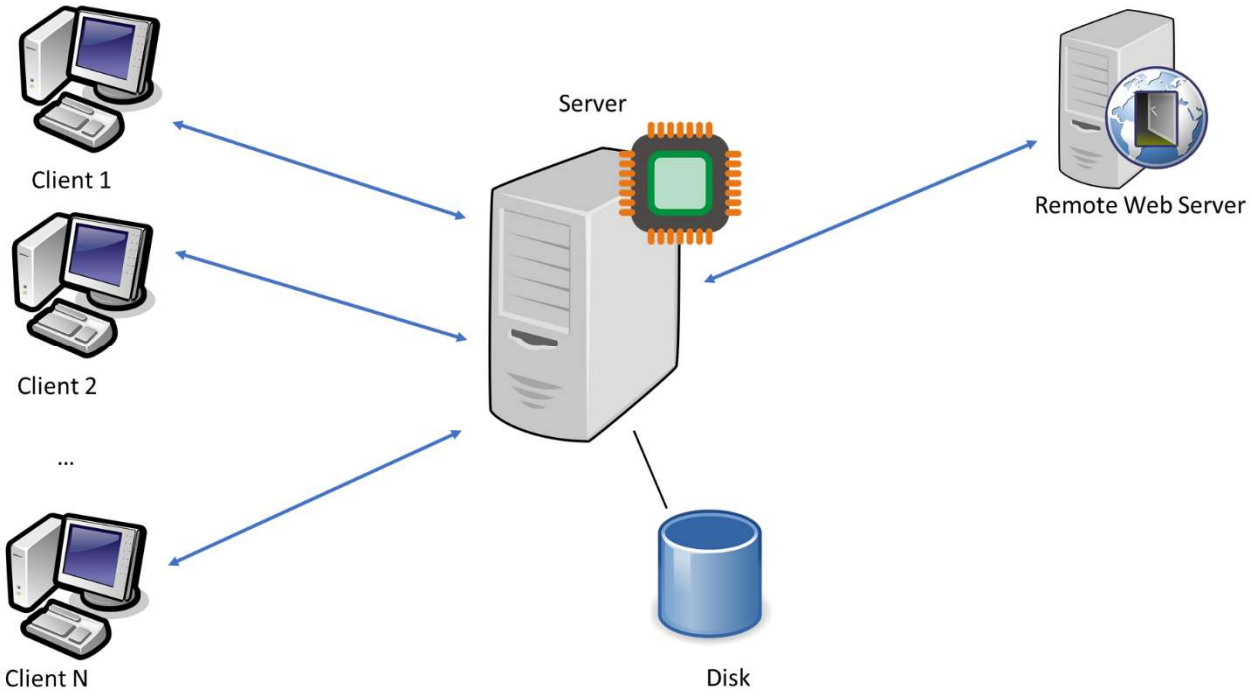
Contents

Overview

A multi-programmed server provides service to different concurrent clients. The server has access to a disk and can communicate with a remote web server. Clients send requests to the server and each request is processed one at a time by the server following a FIFO order. After an initial pre-processing phase, each request can be handled in the following ways:

- The request has finished processing and a response is sent to the client.
- The request requires a disk access and then a new processing is required.
- The request is sent to a remote web server and after its response a new processing is required.

Even disk and the remote web server handle one request at a time in a FIFO order.
A client that receives a reply immediately issues another request.



The aim of this study is to evaluate the performance of the system described above with particular emphasis on throughput. This study must be done considering various scenarios by varying the level of multiprogramming and making sure that the service demands at the three components (server processor, disk, and remote web server) have a considerably different mean.

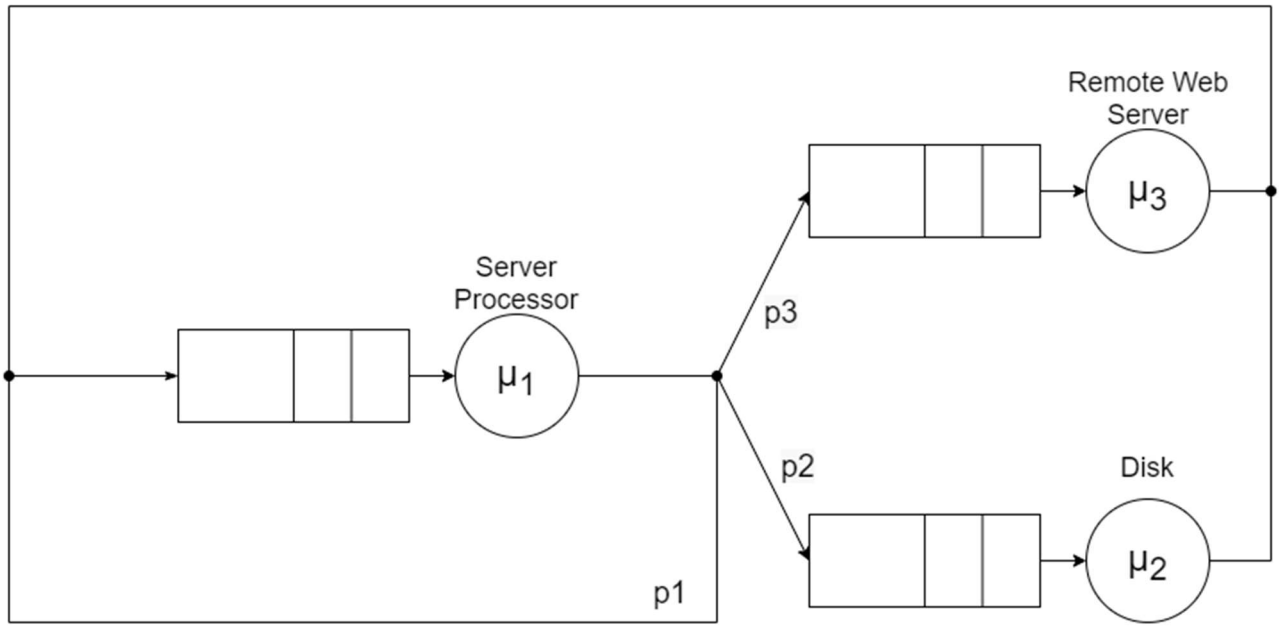
Performance Indexes

Since the objective of this study is to evaluate the performance of the system described above, the following performance indexes have been defined:

- Throughput: the number of served requests divided by the operating time.
- Mean response time: the mean time it takes between the departure of a request and the arrival of the response to the client
- Utilization: How long each system component (server processor, disk, and remote web server) processes requests during operating time.

Model

the following model represents an abstraction of the system to be studied:



The server has been modeled into two service centers (SCs) that represent the two main components that are needed for this specific study: the server processor, and the disk. The remote web server was modeled into a single SC.
Since clients send instantly a new request whenever they receive a response, there is always the same number of service requests in the system. For this reason, as can be seen from the diagram, the system is designed as a Closed Jackson's Network.

So, transactions may involve processing in the main server, access to the disk, and remote queries to a remote web server. As mentioned above, a new transaction always requires some processing time as a first step. After the processing has occurred:

- With a probability p_1 the transaction is terminated, and a reply is sent to the client that originated it.
- With a probability p_2 a disk access is required. After that, a new processing is required.
- With a probability $p_3 = 1-p_1-p_2$ a query to remote web server is required. After that, a new processing is required.

The server processor, the disk, and the remote web server handle one request at a time in a FIFO order. Processing, disk access and query to remote server service demands (with rates μ_1, μ_2 and μ_3) are exponential IID RVs, and they are different from one iteration to another, even for the same transaction.

Factors

For the system model designed, the following factors have been defined.

- Number of clients
- Mean service demands:
  - Mean service demand of server processor
  - Men service demand of disk
  - Mean service demand of remote web server
- Forwarding Probabilities

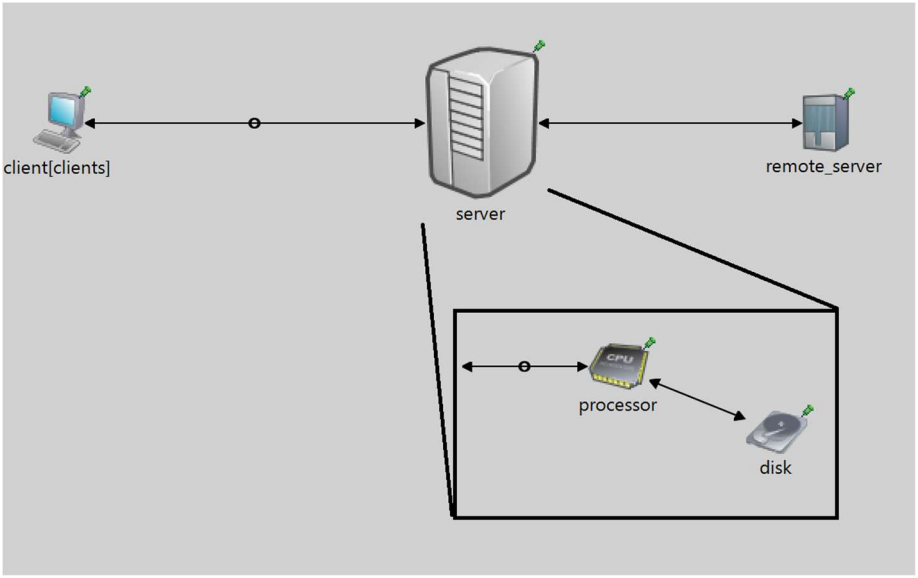
Tools

In the following list there are all the tools used during this study.

- Simulation Tool:
  - **Oment++**: simulation tool used to implement the system simulator.
- Data Analysis and Graph Creation:
  - **MS Excel**
  - **Matlab**
  - **DB Browser**
  - **Python Scripts** (Jupyter NoteBook + Pandas)

Implementation

The simulation tool *Omnet++* was used to implement the system simulator.
The system was implemented as a single network (as you can see in the following figure), consisting of N clients, the server, and the remote web server.



The connections between client and server have been set up with a delay of 40ms, the connection between server and remote web server has been set up with a delay of 80ms. Finally, connections within the server (between processor and disk) have been set up with a delay of 5ms.

Delays have been set based on the average delay within a LAN and the average delay on the Internet.

The system is composed as follows:

- **Client**: composed by a *cSimpleModule*. When initialized, it prepares and sends the first request (*cMessage* type) to the server. When the reply arrives, immediately prepare a new request to send to the server. Whenever a reply arrives, a signal is issued by the client to record the response time of that message.

- **Server**: represented by a *cCompoundModule* composed of two *cSimpleModule*, the processor and the disk.

    - **Processor**: each time it receives a request, if the it is busy, the request is queued, otherwise a pre-processing phase begins (the pre-processing time has an exponential distribution). Once this phase is finished, the processor determines one of the three possible forwarding alternatives using a uniform distribution (0,1) and comparing it with the forwarding probabilities p_1, p_2, p_3. The server emits a signal every time a transaction ends (forwarding probabilities equal to p_1), which allows to calculate throughput.

    - **Disk**: each time it receives a request, if the it is busy, the request is queued, otherwise a processing phase begins (the processing time has an exponential distribution). Once this phase is finished, disk sends a response to the processor and if its queue is not empty, it starts a new processing with a new request.

- **Remote Web Server**: composed by a *cSimpleModule*. Each time it receives a request, if the it is busy, the request is queued, otherwise a processing phase begins (the processing time has an exponential distribution). Once this phase is finished, remote web server sends a response to the processor and if its queue is not empty, it starts a new processing with a new request.

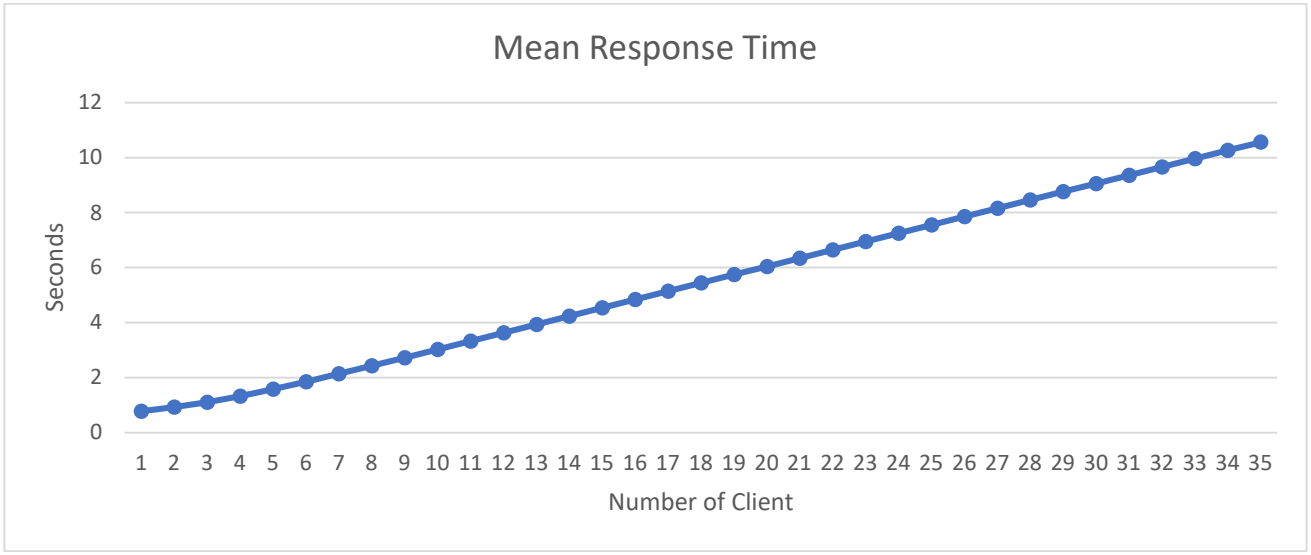
All the queues of the system are represented by a *cQueue*.
Each SC emits a signal every time it finishes processing a request and the queue is empty, indicating the time that has passed in processing. This signal is useful for calculating the utilization of each SC.

Verification

Various methods have been used to verify the correctness of the code and the correctness of the implementation of the model.
First, **Valgrind** was used to verify that there were no checking bad memory accesses or memory leaks.
Then, tests were made by running the simulation. All tests have been carried out considering the mean service demands of each SC equal to 1 second and with a simulation time of one hour (3600s). Each test has been repeated 10 times.

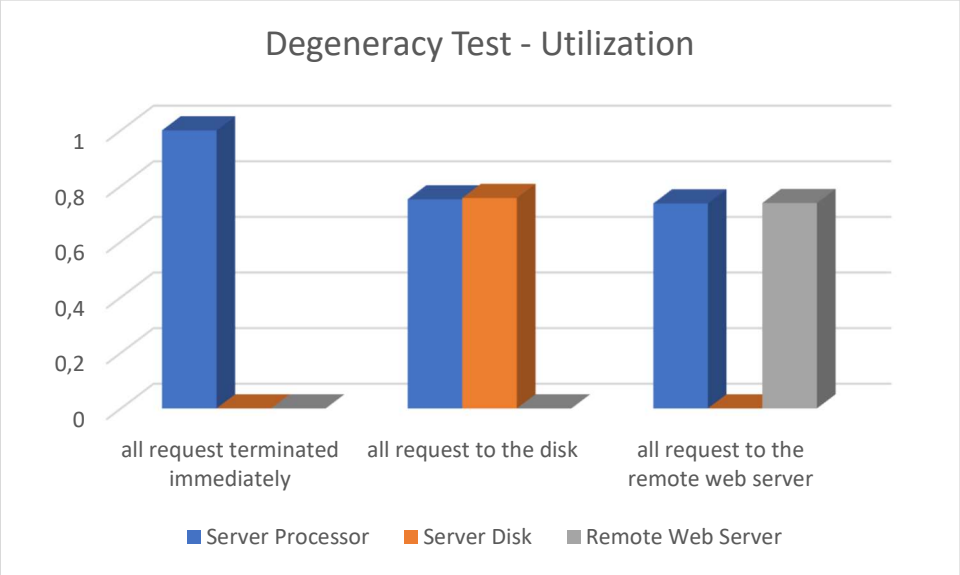
In this initial phase, *MS Excel* was used to process the data and plot the graphs.

- **Continuity Test:** To verify the correctness of the code, we have seen how the mean response time behaves with small progressive changes in the number of clients.



As can be seen from the graph, the mean response time increases linearly with the number of clients.

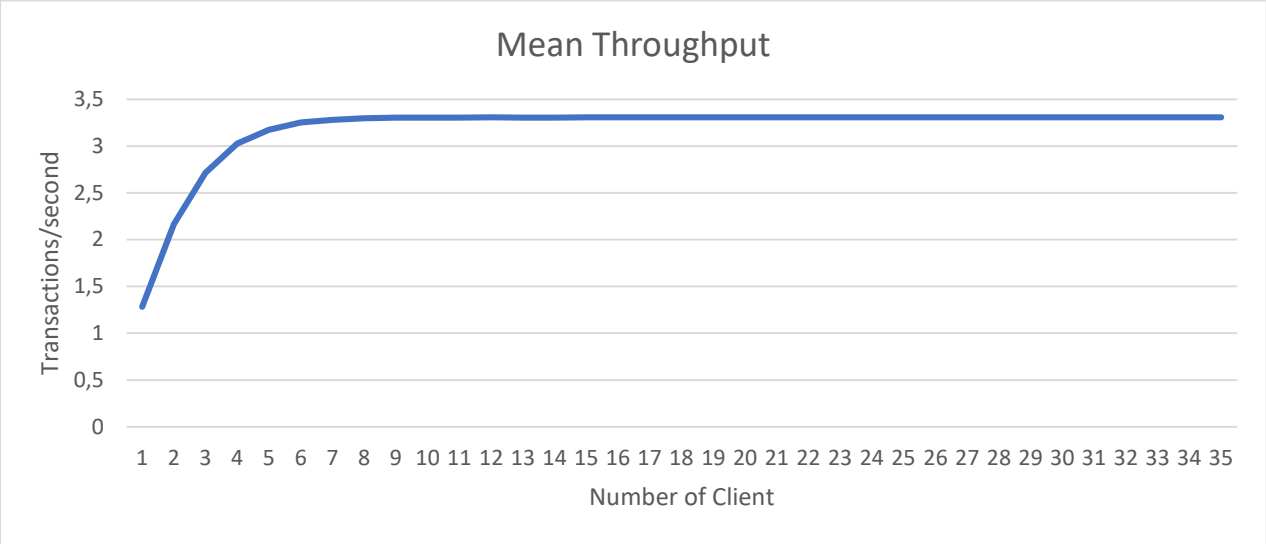
- **Degeneracy Test:** We analyzed the cases where all the requests were forwarded to a single SC. We considered the case in which all requests ended immediately after the pre-processing phase ($p_1 = 1$, $p_2 = p_3 = 0$), all requests are sent to the disk ($p_1 = 0$, $p_2 = 1$, $p_3 = 0$), and all the requests are sent to the remote web server($p_1 = p_2 = 0$, $p_3 = 1$).



As can be seen from the isogram, the degeneracy tests were correct. It is important to note that processor utilization is always high in all three cases because for each type of request there is a pre-processing phase.

Calibration

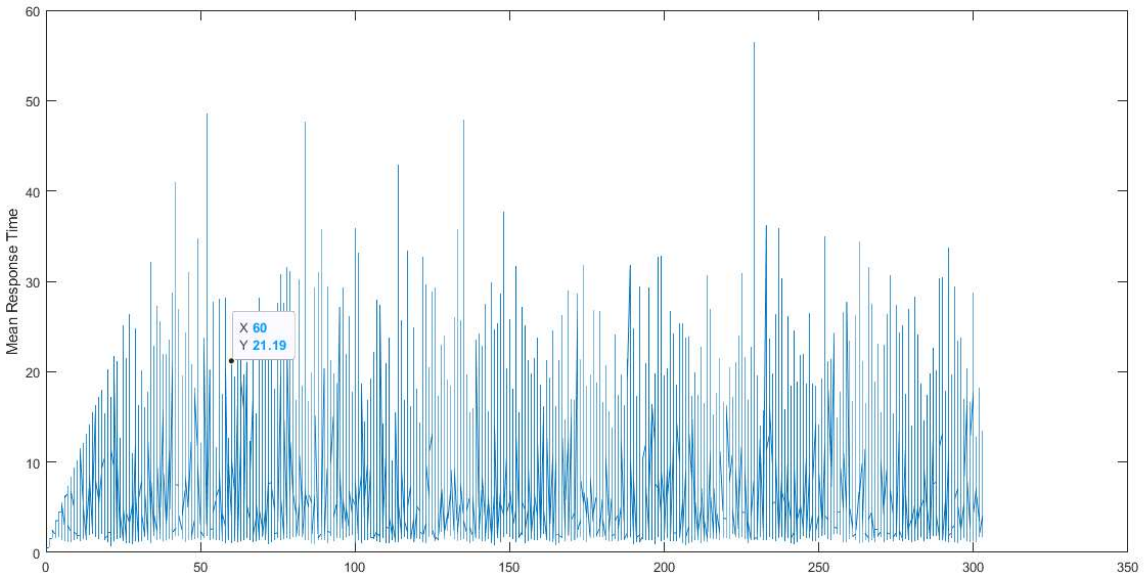
To understand the maximum number of clients to be set in order to have values of interest in the simulations. The mean throughput has been calculated for an increasing number of clients (from 1 to 35). Ten tests were carried out for each number of clients. The graph below shows the average throughput trend as the number of clients increases.



Throughput becomes almost constant as the number of clients increases. For this reason, we have chosen to set the maximum value of the clients to 21.

Warm-up period and Simulation Time

In this phase is important to identify the warm-up period of the system. To do this, a simulation has been made with the maximum number of clients previously selected. Ten tests have been carried out. The mean response time has been analyzed to see when it would stabilize. The maximum number of clients was chosen because it is the worst case, as with increasing clients the average response time increases.



As can be seen from the graph, the mean response time begins to stabilize after 60 seconds. To ensure that the system is stable, a 180 second warm-up time has been chosen.

Consequently, a simulation time of 180+ 3600 ≅ 3800 seconds was chosen, time necessary to have the data for one hour of simulation.

Experiment Design

The experiments were designed by combining forward probability values trying to create realistic scenarios. In particular, the following 4 scenarios have been chosen:

- **S1**: p_1=0.34, p_2=0.33, p_3=0.33
- **S2**: p_1=0.60, p_2=0.25, p_3=0.15
- **S3**: p_1=0.15, p_2=0.60, p_3=0.25
- **S4**: p_1=0.15, p_2=0.25, p_3=0.6

The S1 scenario has been chosen to have a balanced distribution of the various types of requests. The S2, S3 and S4 scenarios have been chosen because they represent respectively a scenario in which a large number of one of the requests is required. These four scenarios allow to analyze various types of applications, from those more oriented to calculation (with a large use of the processor), to those more oriented towards downloading or uploading data from disk, to those that require remote queries.

Various configurations were applied to each scenario by selecting a combination of different service time means of processor, disk, and remote web server. For each SC, two different response time values were chosen:

- **Processor**: 0.001 second or 0.01 second
- **Disk**: 0.01 second or 0.1 second
- **Remote Web Server**: 0.1 second or 1 second

//2^k r analysis

The following configurations were chosen from the previous analysis:

- **C1**: Processor=0.01, Disk=0.01, Remote Web Server=0.01
- **C2**: Processor=0.001, Disk=0.01, Remote Web Server=0.1
- **C3**: Processor=0.001, Disk=0.1, Remote Web Server=0.1
- **C4**: Processor=0.1, Disk=0.01, Remote Web Server=0.001

The above-described scenarios and configurations have been analyzed on varying the number of clients (from 1 to 21) and for each of them ten repetitions were made.

Data and Simulation Analysis