Ho Chi Minh City University of Techonology
Faculty of Computer Science and Engineering

– CO3007 - System Performance Evaluation –

QUEUING MODEL EVALUATION
M/M/1/B=n

Instructor: Dr. Le Hong Trang

Name	Student ID
Le Nguyen An Khuong	1752305
Ho Le Thuc Quyen	1752454

Ho Chi Minh City, Dec 2019

Contents

List of Figures

1 Introduction

Performance is a key criterion in the design, procurement, and use of computer systems. As such, the goal of computer systems engineers, scientists, analysts, and users is to get the highest performance for a given cost. To achieve that goal, computer systems professionals need, at least, a basic knowledge of performance evaluation terminology and techniques. In this assignment, we will evaluate a common and widely implemented in many system: queuing model.

2 System Evaluation

2.1 Goal and System Definition

In this assignment, our goal is aiming to evaluate the relation between job arrive and serve rate, point out the properties of *M/M/1* system and the most important is to make a conclusion if this system consists of stable state. But first, we need to know a little knowledge of our system.

Queueing theory is the key analytical modeling technique used for computer systems performance analysis. The literature on queueing theory is vast. Several hundred papers are published every year. Fortunately, a large percentage of day-to-day performance questions can be answered using only a few techniques. To begin with, you need a knowledge of the queueing notation and some results on single-queue systems. For systems with multiple queues, operational analysis, mean-value analysis, and convolution are very useful. In addition, the technique of hierarchical modeling is helpful in analyzing large systems.

Queue Notation: The *M/M/1/n* notation is interpreted from the following model *A/S/m/B/K/SD* where:

1. **A**: Arrival Process
   - Arrival time $t_1, t_2, ..., t_j$
   - Interarrival time $\tau_j = t_j - t_{j-1}$
   - τ_j form a sequence of Independent and Identically Distributed (IID) random variables.
   - τ_j are IID and exponentially distributed $\Rightarrow$ τ_j follows Poisson distribution.
   - Distribution notation
     - M: Memoryless = Poisson(exponential)
     - E_k: Erlang with parameter k
     - H_k: Hyperexponential with parameter k
     - D: Deterministic
     - G: General $\rightarrow$ Result valid for all distributions.
2. **S**: Service time distribution
   Distribution notation
   - M: Memoryless = Poisson(exponential)

- E_k: Erlang with parameter k
- H_k: Hyperexponential with parameter k
- D: Deterministic
- G: General → Result valid for all distributions.

Exponential distributions is mostly used.

3. **m**: Number of servers
4. **B**: Number of buffer
   Number of buffer is the number of jobs that can stay in the waiting queue.
5. **K**: Population size
   Population size is the total number of jobs
6. **SD**: System Discipline
   System Discipline is the order in which jobs are served. Some common system discipline are:
   - First Come First Served (FCFS)
   - Last Come First Served (LCFS)
   - Last Come First Served Preempt and Resume (LCFS-PR)
   - Round-Robin (RR) with a fixed quantum
   - Round-Robin (RR) with a small quantum → Process Sharing (PS)
   - Infinite Server (IS) = fixed delay
   - Shortest Processing Time first (SPT)
   - Shortest Remaining Processing Time first (SRPT)
   - Shortest Expected Processing Time first (SEPT)
   - Shortest Expected Remaining Processing Time first (SERPT)
   - Biggest In First Served (BIFS)
   - Loudest Voice First Served (LVFS)

The queue that we are going to use in the assignment is *M/M/1/B = n*, with the notation above, our system should have these characteristics (other characteristics that were not mentioned are assumed to be infinity):

- *Arrival Process:* Exponential
- *Service Time Distribution:* Exponential
- *Number of Server:* 1
- *System Capacity:* n (user define)

2.2 Services and Outcomes

This queuing model is following the FCFS (First Come First Serve) service discipline, means that which job comes first will be served first, based on its arrival time in the queue.
This model will provide us:

- Number of jobs done
- Number of jobs drop (when it gets over the buffer size)

2.3 Metrics

For each service, the rate at which the service can be performed, the waiting time for the service and the queue length will be studied. Since the queue length is limited, jobs that arrive when queue is full will be dropped. In this situation, we come up with another metric to measure the reliability of the model.
This lead to the following performance metrics:

- Average service time per job
- Arrival rate
- Average queue length (in stable state)
- Average waiting time
- Average utilization

2.4 Parameters

Parameters that can affected the performance of a system are divided into 2 types:

- **System parameters**:
  - Number of servers.
  - Queue length.
  - Population size.
- **Workload parameters**:
  - Seed for random job generator.
  - Arrival rate and service rate.
  - Number of replications for simulation and validation.

2.5 Factors

In this evaluation, the key factors we are gonna considered about are listed below:

- **Queue Length:** Number of jobs in queue at a specific time
- **Buffer (B):** Total number of jobs including the one is been served and those are in waiting queue.
- **Mean arrival rate (λ):** Calculated by the formula $\frac{1}{E[\tau]}$, where τ is the time between two successive arrivals.
- **Mean service rate (μ):** Calculated by the formula $\frac{1}{E[s]}$, where s is the time service per job.

2.6 Evaluation Technique

The key consideration in deciding the evaluation technique is the life-cycle stage in which the system is. Measurements are possible only if something similar to the proposed system already exists, as when designing an improved version of a product. If it is a new concept, analytical modeling and simulation are the only techniques from which to choose. Analytical modeling and simulation can be used for situations where measurement is not possible, but in general it would be more convincing to others if the analytical modeling or simulation is based on previous measurement.

Because so, the technique we are going to use is simulation since the tools for such method is computer language, very familiar for us to use. Also others criterion are at medium-level so it is reliable to evaluate. To be more specific about the evaluation tool, we are going to use Python to implement the system on our computer since it is heavily mathematics-relate.

Along with Python, there is a simulation supporting tool which is Simpy. SimPy is a process-based discrete-event simulation framework based on standard Python. Processes in SimPy are defined by Python generator functions and may, for example, be used to model active components like customers, vehicles or agents. SimPy also provides various types of shared resources to model limited capacity congestion points.

Figure 1: Discrete event simulation for Python

Server is generated for a several times so that the analysis can be more reliable. We generate m different servers with the same properties which are called replications. In each replication, the information of the number of jobs in buffer is recorded every 1 unit of time. After that, we plot all replications on one graph and use analysis technique to evaluate our system. When running the simulation, Transient Removal technique will be used.

- **Transient Removal** is a technique where we will try to identify the steady-state performance of the simulation as the initial part are unsteady and should not be included in the final computations (this initial part is called transient state. In this technique, there are several methods to use but in our opinion the most suitable one is Initial Data Deletion.

- **Initial Data Deletion**: requires studying the overall average after some of the initial observations are deleted from the sample. During steady state, the average does not change much as the observations are deleted. However, the randomness in the observations does cause the averages to change slightly even during the steady state. To reduce the effect of randomness, the method requires first averaging across several replications. Each replication consists of a complete run of the simulation with no change in input parameter values. The replications differ only in the seed values used in the random-number generators. Averaging across the replications results in a smoother trajectory. This simulation analysis is performed under the following steps:
  Step 1: Calculating the mean value across m replications.

$$\overline{x_j} = \frac{1}{m}\sum_{i=0}^{m-1} x_{ij} \qquad j = 0, 1, ..., n-1$$

  Step 2: Calculating the overall mean value of $\overline{x_j}$.

$$\overline{x} = \frac{1}{n}\sum_{j=0}^{n-1} \overline{x_j}$$

  Step 3: Removing the first "l+1" ("l" begins at 0) observations and calculating the overall mean value of "n-l-1" remaining values of $\overline{x_j}$.

$$\overline{x_l} = \frac{1}{n-l-1}\sum_{j=1+1}^{n-1} \overline{x_j} \qquad l = 0, 1, ..., n-2$$

  Step 4: Computing the relative change between $\overline{x}$ and each $\overline{x_l}$.

$$\frac{\overline{x_l} - \overline{x}}{\overline{x}} \qquad l = 0, 1, 2, ...n-2$$

  Plotting the results from step 1, step 3 and step 4.

  Step 5: Finding the knee which is neither minimum or maximum to get the length of transient interval. The system supposed to be stable after "knee" point.

- **Independent Replications:** This method is based on the assumption that the means of independent replications are independent even though observations in a single replication are correlated.The method consists of conducting m replications of size $n + n_0$ each, where n_0 is the length of transient phase.The first n_0 observations of each replication are discarded. The remaining steps are as follows:
  Step 1: Compute a mean of each replication:

$$\overline{x}_i = \frac{1}{n}\sum_{j=n_0+1}^{n_0+n} x_{i,j}, i = 1, 2, ..., m$$

Step 2: Compute an overall mean for all replications:

$$\overline{\overline{x}} = \frac{1}{m}\sum_{i=1}^{m}\overline{x}_i$$

Step 3: Calculate the variance of replicate means:

$$Var(\overline{x}) = \frac{1}{m-1}\sum_{i=1}^{m}(\overline{x}_i - \overline{\overline{x}})^2$$

Step 4: Confidence interval for the mean response is:

$$\overline{\overline{x}} \pm z_{1-\alpha/2}\sqrt{\frac{Var(\overline{x})}{m}}$$

2.7 Workloads

The system's workload consists of 2 parts:

- **The arrival process generator:** This process will generate arrival process for the system following the Poisson arrivals, which simply means that the inter-arrival times are independent and identically distributed (IID) and and are exponentially distributed.
- **The time service generator:** This process will random time for each service and recorded it to a log file.

2.8 Experimental Design

For testing the system, we set "key variables" and divide them into some specific cases and in each case the simulation will be run five times to make the result of the simulation objective:

For every case below, we make experiments when $\lambda < \mu$ and $\lambda > \mu$.
E1: Buffer is small.
E2: Buffer is large enough.
E3: Buffer is very large and supposed to equal to ∞.

2.9 Analyze and interpret Data

The following is analytical result from the $M/M/1/n$ queuing network:

1. **Parameters:**
   - λ: arrival rate of jobs per unit time.
   - μ: service rate in jobs per unit time.
   - B: number of buffer.
2. **Traffic intensity:** $\rho = \dfrac{\lambda}{\mu}$

3. **Stable condition:** $\rho < \infty$
4. **Probability of zero jobs in the system:** $p_0 = \begin{cases} \dfrac{1-\rho}{1-\rho^{B+1}}, & \rho \neq 1 \\ \dfrac{1}{B+1}, & \rho = 1 \end{cases}$
5. **Probability of n jobs in the system:** $p_n = \begin{cases} \dfrac{1-\rho}{1-\rho^{B+1}} \times \rho^n, & \rho \neq 1, 0 \leq n \leq B \\ \dfrac{1}{B+1}, & \rho = 1 \\ 0 & n > B \end{cases}$
6. **Mean number of jobs in the system:** $E[n] = \dfrac{\rho}{1-\rho} - \dfrac{(B+1)\rho^{B+1}}{1-\rho^{B+1}}$
7. **Mean number of jobs in queue:** $E[n_q] = \dfrac{\rho}{1-\rho} - \rho \times \dfrac{1+B\rho^B}{1-\rho^{B+1}}$
8. **Effective arrival rate:** $\lambda' = \sum_{n=0}^{B-1} \lambda p_n$
9. **Mean response time:** $E[r] = \dfrac{E[n_q]}{\lambda'}$
10. **Mean waiting time:** $E[w] = E[r] - \dfrac{1}{\mu}$

2.10 Result

For each experiment's result, we show the graphs of our system where the left one is for $\lambda < \mu$ and the right one is for $\lambda > \mu$. A graph "Queue" which is on the top left illustrates the number of jobs in the queue over time and the next graph "Mean across replications" displays the mean value of all value at one time of the previous graph. The others two graph depict the mean $\overline{x_j}$ and relative change of initial data deletion simulation technique. In all experiments, we run the system for 10000 unit of time in SimPy.

- Experiment E1: Buffer equals to 10 in two cases $\lambda = 2.0 < \mu = 4.3$ and $\lambda = 4.0 > \mu = 2.3$.

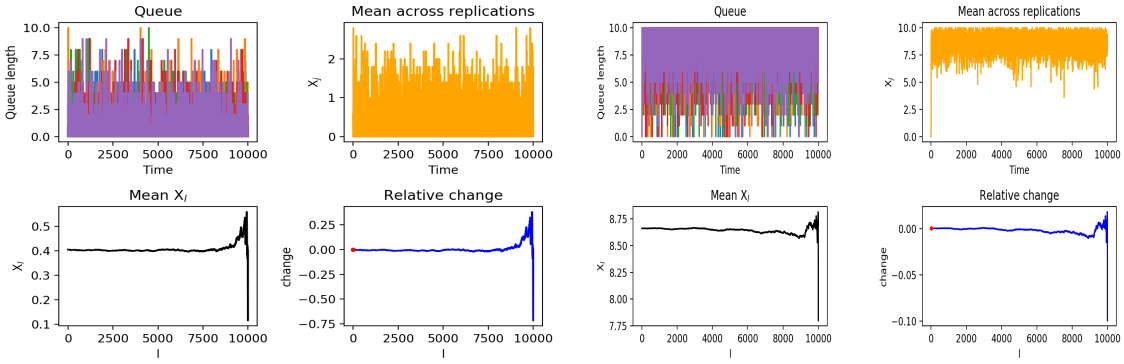

Figure 2: System simulation with a small buffer

- Experiment E2: Buffer equals to 200 in case $\lambda = 2.1 < \mu = 4.5$ and equals to 800 in case $\lambda = 5.1 > \mu = 3.5$.

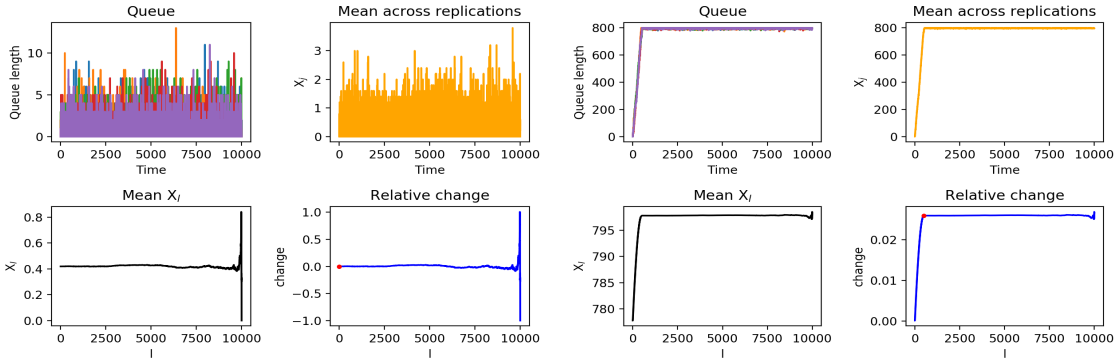

Figure 3: System simulation with a reasonable buffer

- Experiment E3: Buffer equals to 2000000 in case $\lambda = 3.1 < \mu = 5.8$ and equals to 800 in case $\lambda = 3.1 > \mu = 2.8$.

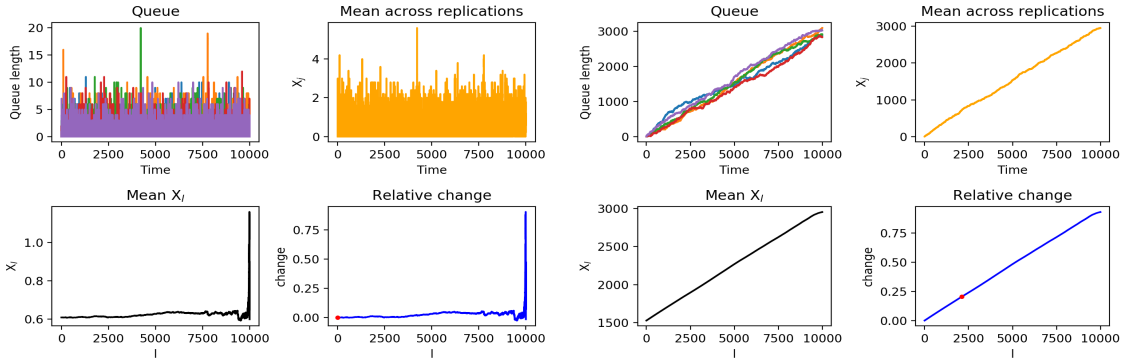

Figure 4: System simulation with a very large buffer

3 Validation

To validate the system, we will use Theoretical Results as we validate the model by checking the similarity of theoretical result and simulation result. Model validation consists of validating the three key aspects of the model which are as follows:

- **Assumptions:** Service discipline is Last Come First Serve, the buffer is limited and traffic intensity (ρ) must be less than 1.
- **Input parameter values and distributions:** mean arrival rate $\lambda = 2.0$ and mean service rate $\mu = 4.3$ are exponentially distributed. The buffer equals to 10.
- **Output values and conclusions:** The tables below compare some of the theoretical results to that of the simulation model.

	Theoretical result	**Practical result**
Traffic intensity	0.465	0.467
Mean number of jobs in queue	0.402	0.415
Mean number of jobs in system	0.867	0.884
Mean waiting time	0.201	0.207
Mean response time	0.434	0.439

Figure 5: Buffer is small and $\lambda < \mu$

Interpretation from the table show that our model does behave like a real model. This helps increase in the degree of confidence in our model result.

In this experiment, we also applies "Independent Replications" technique to calculate the confidence interval. We uses that in case where $\lambda < \mu$ and buffer equals to 10.
The result approximately equals to $\overline{\overline{x}} \pm 0.009$, where $\overline{\overline{x}} \approx 0.415$ and $Var(\overline{x}) \approx 0.0001137$.

4 Conclusion

The system condition after the Transient Removal, it still hardly reach the stable state, after many experiences we find this system is not efficient as the conditions to achieve the stable state are nearly impossible in real life system (which is the service will come as soon as the previous job is served). Another reason why this system is not suitable for real life implementation which is its ability to work in a longer time, the queue will eventually exceed the maximum buffer capacity no matter how large it is and the system is forced to stop before serving all the jobs.

References

[1] R. K. Jain, *The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation, and Modeling*, Wiley, 1991

[2] Paul Fortier, *Howard Michel Computer Systems Performance Evaluation and Prediction*, Digital Press, 2003.