Abstract:

Cloud computing has become a dominant paradigm in the IT environment in recent years. Its principle is to provide services to customers on demand, and allows them to pay only for what they need. The increasing need for this type of service brings the providers to increase the size of their infrastructure, as energy consumption and associated costs become very significant.

A cloud service provider must provide all the different types of requests. Infrastructure managers need then to host all the types of services together. That's why during this thesis, we focus on the modeling and analysis of management resources in data center environments in cloud computing. Analytical models are developed to improve the quality of services deployed in the cloud and minimize the energy consumption. Two great analytical tools have been used to achieve this purpose, queuing theory and dynamic programming.

To achieve this, we first modeled and studied the problem of initial resource allocation, proposing two algorithms approached via heuristics, and then exploiting the concept of virtual machine migration in a cloud data center.

The notion of reserve servers is implemented in our models in order to manage energy consumption. Servers are divided into two types: In run mode and in sleep mode as a reserve when there is no work to be done. Usually, the service receives customers in one way or another, we have focused on M/M/K queue system with a setup cost that describes a delay when a new reserve server is turning on. In another approach, a controller is placed in front of the arriving clients to manage their placement in the system and he is the one who decides whether to turn on a reserved server or not depending on the client's needs.

Typically, services have to deal with different load levels. This is why we have developed the resource allocation models including the dynamical nature of requests and resource usage, as well as the notion of the cost of switching servers ON and OFF and the cost of migrating services from one server to another.

The results of the models used during the works have been validated by both analytical and numerical solutions and some of them are compared qualitatively with other recent works.

Keywords: Queueing theory, Dynamic programming, Resource allocation and management, Cloud computing, Performance evaluation.

N° d'ordre

OUAMMOU Abdellah

Queuing Theory and Dynamic Programming to Model Resources Allocation in a Cloud Computing Environment

2021, Mathématiques et Informatique Appliquées

Université Hassan 1er
Centre d'Études Doctorales
Faculté des Sciences et Techniques
Settat

THÈSE DE DOCTORAT

Pour l'obtention de grade de Docteur en Sciences et Techniques
Formation Doctorale : Mathématiques et Informatique Appliquées (MIA)
Spécialité : Mathématiques Appliquées

Sous le thème

Queuing Theory and Dynamic Programming to Model Resources Allocation in a Cloud Computing Environment

Présentée par :
OUAMMOU Abdellah

Soutenue le : 08 octobre 2021
A la Faculté des Sciences et Techniques de Settat devant le jury composé de :

HAQIQ Abdelkrim	PES	FST Settat	Président
AIT BABRAM Mohamed	PES	FST Marrakech	Rapporteur
NAFIDI Ahmed	PES	ENSA Berrechid	Rapporteur
FAKHAR Rachid	PH	FP Khouribga	Examinateur
HANINI Mohamed	PH	FST Settat	Co-Directeur de thèse
BEN TAHAR Abdelghani	PES	FST Settat	Directeur de thèse

Année Universitaire : 2020/2021

Remerciement

Avant tous, je remercie ALLAH, le tout puissant, le miséricordieux, de m'avoir donné la santé et tout dont je nécessitais pour l'accomplissement de cette thèse.

Je tiens à exprimer ma profonde gratitude et mes remerciements à mes directeurs de thèse, Pr. Abdelghani BEN TAHAR et Pr. Mohemed HANINI. Je n'oublierai jamais ces quatres années durant lesquelles ils m'ont fait découvrir le monde de la recherche et transmis leur passion pour la recherche. Ils se sont montrés disponibles et patients et m'ont encouragé tout au long de cette thèse.

Je remercie de tout coeur Pr. Saida AMINE, Pr. Mohammed AIT BABRAM et Pr. Ahmed NAFIDI pour avoir accepté de rapporter cette thèse, de leur intérêt pour mon travail, ainsi que pour leurs remarques positives et encourageantes.

C'est avec grand plaisir que je remercie Pr. Abdelkrim HAQIQ qui m'a fait l'honneur de présider le jury de cette thèse. Je suis aussi reconnaissant enver Pr. Rachid FAKHAR pour avoir consenti à être membre de mon jury. C'est pour moi un grand honneur de partager mon travail avec eux.

Je remercie également tous mes amis , également à tous mes collègues doctorants de la FST Settat, sans lesquels l'ambiance n'aurait jamais été la même. Sans être exhaustif, je pense à Adnane, Hamid, Ahmed, Allal, Said, Azzedine, Fatima Ezzahra, Salma, Hiba et Amal, ainsi que d'autres que j'ai sans doute oublié de citer mais que je remercie.

Les années de thèse ne se résumant pas à un unique travail de recherche, je tiens à mentionner que j'ai aussi pris beaucoup de plaisir à effectuer ma mission d'enseignement au sein du Département de Mathématiques et Informatique de la FST Settat et je voudrais à ce titre remercier les Professeurs pour qui j'ai effectué des TD et TP, Pr. mohamed HANINI, Pr. Abdelghani BEN TAHRA, Pr. Mohamed TOUHAMI Pr. El Ghalia BOUZIANE, Pr. Amine BEN MAKHLOUF, Pr. Abdessamad TALHA et Pr. Jaouad DABOUNO . Ce fut un véritable plaisir de travailler avec eux et j'en garderai d'excellents souvenirs. Je les remercie pour cette expérience unique et inoubliable.

Enfin, je tiens à exprimer mes derniers remerciements aux personnes qui me sont les plus chères, mes parents, mon frère ainsi que mes sœurs pour leurs soutien pendant ces longues années d'études, et sans qui rien de tout ceci n'aurait été possible. C'est à cette charmante et solide famille que je dédie cette thèse.

Abstract

Cloud computing has become a dominant paradigm in the IT environment in recent years. Its principle is to provide services to customers on demand, and allows them to pay only for what they need. The increasing need for this type of service brings the providers to increase the size of their infrastructure, as energy consumption and associated costs become very significant. A cloud service provider must provide all the different types of requests. Infrastructure managers need then to host all the types of services together. That's why during this thesis, we focus on the modeling and analysis of management resources in data center environments in cloud computing. Analytical models are developed to improve the quality of services deployed in the cloud and minimize the energy consumption. Two great analytical tools have been used to achieve this purpose, queuing theory and dynamic programming. To achieve this, we first modeled and studied the problem of initial resource allocation, proposing two algorithms approached via heuristics, and then exploiting the concept of virtual machine migration in a cloud data center. The notion of reserve servers is implemented in our models in order to manage energy consumption. Servers are divided into two types: In run mode and in sleep mode as a reserve when there is no work to be done. Usually the service receives customers in one way or another, we have focused on M/M/K queue system with a setup cost that describes a delay when a new reserve server is turning on. In another approach, a controller is placed in front of the arriving clients to manage their placement in the system and he is the one who decides whether to turn on a reserved server or not depending on the client's needs. Typically, services have to deal with different load levels. This is why we have developed the resource allocation models including the dynamical nature of requests and resource usage, as well as the notion of the cost of switching servers ON and OFF and the cost of migrating services from one server to another. The results of the models used during the works have been validated by both analytical and numerical solutions and some of them are compared qualitatively with other recent works.

Keywords: Queueing theory, Dynamic programming, Resource allocation and management, Cloud computing, Performance evaluation.

Résumé

Ces dernières années le cloud computing est devenu un paradigme dominant dans l'environnement informatique. Son principe est de fournir des services aux clients à la demande, et leur permettre de payer seulement ceux dont ils ont besoin. Le développement de ce type de services amène les fournisseurs à augmenter la taille de leur infrastructure, à tel point que la consommation d'énergie et les coûts associés deviennent très importants. Un fournisseur de cloud services doit fournir toutes les différentes demandes. De leur côté, les gestionnaires d'infrastructure doivent héberger tous les types de services ensemble. C'est pourquoi, au cours de cette thèse, nous nous concentrons sur la modélisation et l'analyse de la gestion des ressources dans les environnements de centres de données dans le cadre de l'informatique en nuage. Des modèles analytiques sont développés pour améliorer la qualité des services déployés dans le nuage et minimiser la consommation d'énergie. Deux grands outils analytiques ont été utilisés à la réalisation de cet objectif, la théorie des files d'attente et la programmation dynamique. Dans ce but, nous avons dans un premier temps modélisé et étudié le problème de l'allocation initiale des ressources, en proposant deux algorithmes abordées via des heuristiques, puis en exploitant le concept de migration des machines virtuelles dans un centre de données en nuage. La notion de serveurs de réserve est implémentée dans nos modèles afin de gérer la consommation d'énergie. Les serveurs sont divisés en deux types : En mode exécution et en mode veille comme réserve lorsqu'il n'y a pas de tâches à exécuter. Généralement, le service reçoit des clients d'une manière ou d'une autre, nous nous sommes concentrés sur les files d'attente M/M/K avec un coût de setup qui désigne un délai lorsqu'un nouveau serveur de réserve est mis en service. Dans une autre approche, un contrôleur est placé devant les clients qui arrivent pour gérer leur placement dans le système et c'est lui qui décide de mettre en marche ou non un serveur de réserve en fonction des besoins du client. En général, les services doivent gérer différents niveaux de demande. C'est pourquoi nous avons développé les modèles d'allocation des ressources incluant la nature dynamique des demandes et l'utilisation des ressources, ainsi que la notion de coût de mise en fonction et d'arrêt des serveurs et le coût de migration des services d'un serveur à l'autre. Les modèles mathématiques proposés sont validés numériquement et certains d'entre eux sont comparés qualitativement à d'autres travaux récents.

Mots-clés: Théorie des files d'attente, Programmation dynamique, Allocation et gestion des ressources, Cloud computing, Évaluation des performances.

Contents

List of Abreviations

- API : Application Programming Interfaces
- BFD : Best Fit Decreasing
- CDC : Cloud Data Center
- CM : Combinations of Migrations
- CTMC : Continuous Time Markov Chain
- DES : Discrete Event Simulator
- DVFS : Dynamic Voltage and frequency Scaling
- DyT : Dynamic Threshold
- FCFS : First Come First Served
- FFDS : First Fit Decreasing Scheduling
- FIFO : First In First Out
- GA : Greedy Algorithm
- HOL : Hold On Line
- HP : High Priority
- HPG : Highest Potential Growth
- IaaS : Infrastructure as a Service
- I/O : Input/Output
- IT : Information Technology
- LCFS : Last Come First Served
- LIFO : Last In First Out
- LP : Location Precedence
- MBFD : Modified Best Fit Decreasing

- MDP : Markov Decision Process
- MILP : Mixed Integer Linear Programming
- MIPS : Millions of Instructions Per Second
- MM : Minimisation of Migration
- MOGA : Multi-Objective Genetic Algorithm
- MPM : Main Physical Machine
- NPA : No Power Aware
- NIST : National Institute for Standards and Technology
- OGC : Open Green Cloud
- PaaS : Platform as a Service
- PM : Physical Machine
- PR : Preemption
- PS : Processor Sharing
- QED : Quality and Efficiency Driven
- QoS : Quality of Service
- RC : Random Choice policy
- RPM : Reserve Physical Machine
- SaaS : Software as a Service
- SLA : Service Level Agreement
- SMDP : Semi Markov Decision Process
- ST : Single Threshold
- TT : Two Thresholds
- VCC : Cloud Computing for Vehicles
- VI : Value Iteration
- VM : Virtual Machine
- VMM : Virtual Machine Monitor

List of figures

List of tables

Introduction

Contents

1 Thesis context

Cloud computing has become an important part of the IT environment in recent years. Its principle role is to provide a distributed service and enable customers to consume resources on a pay-per-use model. The growing need for this type of service is leading service providers to increase the size of their infrastructures, to the extent that energy consumption and operating costs are becoming significant. However, cloud computing environment have attracted a lot of attention in recent years, a lot of researches have been interested in this topic and many algorithms have been proposed to manage cloud resources to balance QoS and energy cost of the system. Although, the virtualization of resources and migration are among basic techniques used to improve the energy efficiency of cloud computing. The virtual machine (VM) allocation problem: determining the best placement of VMs on Physical machines (hosts) taking into account the Quality of Service (QoS) guarantees, the cost associated with using the hosts and the energy consumed takes a lot of interest in this field. This mainly concerns for the owner looking to maximize revenue or minimize operating costs, for example to reduce energy consumption in data centers, just turning off unused PM, even a significant amount of energy is consumed when the Physical Machine (PM) is turned off. But in reality, it is not as simple as it seems. Cloud data centers are very complex and highly dynamic systems, so we do not know which PMs need to be off or not ..., so that is a real cloud management system that must deal with uncertainty aspect of either demand and behavior of cloud computing.

2 Problem statement and objectives

The growing demand for infrastructure in cloud computing environments is increasing the energy consumed by data centers, which has become a critical issue. High energy consumption translates into lower supplier revenue costs and CO2 emissions, and much research has been done on the energy consumption of data centers. If this consumption continues unchecked, the

cost of energy consumed by a server over its lifetime exceeds the cost of the equipment. As a result, effective solutions are needed to minimize the impact of cloud provider profits as well as ensuring the reliable quality of service (QoS) defined by Service Level Agreements (SLAs) is essential for cloud computing environments, so cloud providers have to find a compromise between the energy performance of data centers and SLA contracts.

The objective of this thesis is to study the state of the art in dynamic allocation of VMs in Cloud Data Centers (CDCs). This research aims to present new techniques, models, algorithms and policies for distributed dynamic allocation of virtual machines in CDCs. The goal is to improve the use of IT resources and reduce power consumption under workload-independent QoS constraints, Dynamic VM allocation takes into account the characteristics of data centers in a cloud computing system, while considering the random aspects of incoming service demand from customers and those of the functioning of the data center architecture.

To achieve this objective mathematical modeling tools, namely Queuing theory and dynamic programming, as models that can take into account, the dynamic and stochastic aspect of cloud computing environment are used to analytically study this concerns.

3 Contributions

The contributions of this thesis can be mainly divided into three categories: Novel algorithms for distributed dynamic VM allocation, a formulation of the dynamic virtual machine management problem as a Markov decision process problem using reserves servers and an analysis of a M/M/k system with exponential setup times and reserves servers. The main contributions are:

1. An overview of the previous dynamic VM allocation algorithms and discussion about their advantages and inconvenients, lead to propose an algorithm that has been a result of a numerous algorithms in literatures. Dynamic VM consolidation exploits the highly workloads and continuously reallocates VMs using migration to minimize the number of active physical nodes to serve the current resources demand. The proposed algorithm is based on double thresholds that manages the selection and placement of VMs in PMs, this management deals with power/performance trade off. Numerical results demonstrate that it could be enable to reduce the energy consumed and minimize the execution time in CDC.
   This contribution has been subject of publications :

   - A.Ouammou, M.Hanini, S.El Kafhali, A.Ben Tahar, Energy Consumption and Cost Analysis for Data Centers with Workload Control International Conference on Innovations in Bio-Inspired Computing and Applications, (2017), pp.92-101. Springer.
   - A.Ouammou, A.Ben Tahar, M.Hanini, S.El Kafhali, Modeling and Analysis of Quality of Service and Energy Consumption in Cloud Environment International Journal of Computer Information Systems and Industrial Management Applications, ISSN 2150-7988, vol.10, (2018),pp. 98–106.

2. Turning off the unused PMs is one of effective and old way to reduce energy but what happens if we think differently, if we start with a minimum number of active PMs and

depending on the state of the system, we turn on another PM, consider the M/M/K/Setup, the K servers are divided in two types: In run mode and in sleep mode as a reserve, the reserves PM could be in run mode in any time with a setup cost in form of a delay. The proposed mechanism and its performance parameters are described by a mathematical model, in addition the effect of the proposed mechanism on energy consumption behavior is assessed.
This contribution has been subject of publication :

- A. Ouammou, A. Ben Tahar, M. Hanini, S. El Kafhali, Analysis of a M/M/k system with exponential setup times and reserves server, BDIoT'19: Proceedings of the 4th International Conference on Big Data and Internet of Things October 2019 Article No.:58 Pages 1–5

3. A stochastic model is proposed while controlling the flow of customers into the system, using dynamic programming and heuristic algorithm to assess the best management of system resources. The proposed mechanism helps to manage the resources by improving QoS and minimizing energy consumption based on a "PM reserves" approach. That trade-off can be evaluated by analyzing and solving a Semi Markov Decision Process (SMDP) model of the system. However, the optimal policy of the requests is dynamically achieved by applying the value iteration algorithm (VI). As an implementation of our algorithm, we studied the optimal choice of the controller in order to control the performance of the system and to minimize energy consumed.
This contribution has been subject of publications :

   - A. Ouammou, A. Ben Tahar, M. Hanini, S. El Kafhali, A dynamic programming approach to manage virtual machines allocation in cloud computing, International Journal of Engineering and Technology, ISSN: 2227-524X, vol.7, No 4.6 (2018), pp. 128-132.
   - Abdellah Ouammou, Abdellah Zaaloul, Mohamed Hanini, and Abdelghani Bentahar, "Modeling Decision Making to Control the Allocation of Virtual Machines in a Cloud Computing System with Reserve Machines," IAENG International Journal of Computer Science, vol. 48, no.2, pp285-293, 2021

4 Thesis organization

The structure of this thesis is organized into five chapters.

- Chapter 1 is devoted to mathematical modeling tools including queuing theory and dynamic programming for modeling different system behaviors.
- Chapter 2 represents a survey on cloud computing technology and related to management resources issues. This chapter will also present some of the scientific works being done by researchers working on the same research area.

Chapters 3, 4 and 5 are intended to address the three research contributions described in the above sections.

- Chapter 3 presents the algorithms proposed and describe the operating principle of the two phases "selection and allocation". In the same chapter, We will describe the architecture of the proposed model, the mechanisms used which are necessary to compare the results obtained with other existing approaches.
- Chapter 4 introduces the concept of reserve server and consider the M/M/k queuing system with setup costs: M/M/k/Setup to deal with coming clients. The proposed mechanism and its performance parameters are described by a mathematical model.
- Chapter 5 presents a modeling decision to control the allocation of VM and its resolution using dynamic programming and heuristic algorithm to assess the best management of system resources.
- Finally, a conclusion gives a summary of this thesis contributions and presents possible lines of research to refine this work and continue the research.

Chapter 1

Mathematical Modelling tools of performance evaluation of IT systems

Contents

The purpose of this chapter is to provide an overview of queuing theory and dynamic programming and also briefly discuss the problems of computer system performance modelling.

1.1 Introduction

Today, some IT systems are so complex so that intuition alone is not enough to predict their performance. Thus, various tools, such as simulation and mathematical modelling, play an important role in the model of these systems. Performance is a key factor that must be taken into account. Several methods can be used to evaluate this performance. For example, hardware and/or software monitors can be used, either in a user environment or under controlled conditions (test facility). However, it is necessary to make performance predictions based on statistical measures using models. In addition to quantitative predictions, modelling also provides a better understanding of the structure and behaviour of the system during development. This can ultimately help to discover and correct model failures.

The development of a performance model consists of characterising the hardware and software resources that make up the system by means of parameters. The choice of these parameters and a way to represent them are the basic elements of the modeling. It should be noted that during the development of performance models, several hypotheses can be introduced in order to simplify the complexity of a system. The validation of these models becomes essential. Often several models with different levels of detail are developed to represent a system. For example, there may be a very detailed simulation model and a simpler analytical model. If the system has not yet been realized, then the validity of the simpler model could be obtained by comparing its predictions with those obtained with the more detailed model.

A set of machines can be considered as a collection of hardware (CPUs, memories, etc.) and software (system programs, compilers, etc.) as a resources. The performance of a system depends directly on the organisation and management of these resources. A task (an element of the program located in the machine's memory) is the processing of a transaction. Usually only one task can use a resource at any one time. Therefore, all other tasks that need to use that resource must wait in a queue. Queuing theory is particularly well suited to analyzing the performance of IT systems. For example, it is possible, using this theory, to estimate the time required for tasks to be waiting in queues on a system. These waiting times can then be analyzed to predict the response time of the system. Note that response time is generally an important indicator of system performance. An introduction to queue theory will be presented in the next section 1.2. Therefore, in order to carry out this work, we will provide the theoretical background for dynamic programming and Markov decision processes in section 1.3 and 1.4.

1.2 Queuing theory

Queueing can be considered a typical phenomenon of contemporary life. They can be found in the most diverse fields of activity (post office counters, road traffic, call centers,...etc.).

The mathematical study of waiting phenomena constitutes an important field of application of stochastic processes. We speak of waiting phenomena whenever certain units called " clients " present themselves in a random way to " service stations " in order to receive a service, the duration is generally random. The objective of this lecture is to discuss the structure and calculate characteristic values to describe the performance of such systems.

1.2.1 Simple file

A simple queue (or station) is a system made up of one or more servers and a waiting space as shown in figure 1.1. Customers arrive from outside, eventually wait in the queue, receive service, and then leave the station. In order to completely specify a simple queue, we must characterize the customer arrival process, the service time, and the structure and service discipline of the queue.

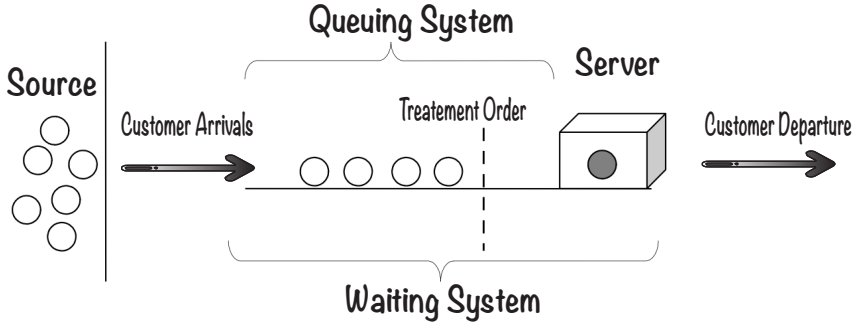

Figure 1.1 – General structure of a queuing system

1.2.2 Arrival process

The arrival of customers at the station will be described using a stochastic counting process. $(N_t \geq 0)$. [Note indifferently N_t or $N(t)$]. If A_n designates the random variable measuring the time of arrival of the n^{th} customer in the system, we will have: $A_0 = 0$ (by agreement) and $A_n = \inf\{t; N_t = n\}$.

If W_n designates the random variable measuring the time separating the arrival of the $(n-1)^{th}$ customer and the n^{th}, then we have: $W_n = A_n - A_{n-1}$.

Definition : A counting process $(N_t)t \geq 0$ is a renewal process if and only if the random variables (W_n) are independent and identically distributed variables. The law describing the interarrival time is then sufficient to characterize the renewal process.

Most of the time, the arrival of customers to a simple queue is assumed to be described by a renewal process. The simplest and most commonly arrival process used is the Poisson

process. It is a renewal process where the arrivals are distributed according to an exponential process.
We note λ the rate of arrivals: $\frac{1}{\lambda}$ is the average interval between two successive arrivals.

1.2.3 Service time

First of all, consider a single-server queue. Note D_n the random variable measuring the starting time of the n^{th} client of the system and W_n the random variable measuring the service time of the n^{th} client (time between the beginning and the end of the service). A start time always corresponds to an end of service, but does not necessarily correspond to a beginning of service. It is possible that a customer who leaves the service station may leave it empty. The server is then unoccupied until the next client arrives. Only stations with continuous service times described by independent and identically distributed (i.i.d.) variables will be considered. Note that μ is the service rate: $\frac{1}{\mu}$ is the average time of service.

The simplest service time distribution to study is the exponential distribution. However, the "memoryless" property of the exponential distribution makes it generally not very realistic for modeling real phenomena. We are therefore often obliged to use other service distributions.

1.2.4 Queue structure and discipline

A. Number of servers

A station can have several servers in parallel. Let C be the number of servers. As soon as a client arrives at the station, either one server is free and the client instantly goes into service, or all the servers are occupied and the client is placed in the queue waiting for one of the servers to be released. Most of the time, the servers are supposed to be identical (they have the same distribution) and independent from each other. A special station is the infinite servers IS station in which the number of servers is infinite. This station does not have a queue. As soon as a client comes to this station, he can find an available server instantaneously and directly enter in service. It allows to represent systems with more servers than the number of clients that can be present.

B. Queue capacity

The queue capacity for making customers wait the service can be finite or infinite. Let K be the capacity of the queue (including the customers in service). A queue with finite capacity : customer arrives and if the queue is full, the customer is rejected. In other hand the queue with infinite capacity $K = +\infty$ there is no reject.

C. Service Discipline

Service discipline determines the order in which customers are placed in the queue and allowed to be served. The most common disciplines are :

- FIFO (first in, first out) or FCFS (first come first served): this is the standand queue in which customers are served in their order of arrival. Note that the FIFO and FCFS disciplines are not equivalent when the queue contains several servers. In the first one, the first arrived client will be the first to leave the queue while in the second one, it will be the first to start its service. Nothing prevents a client who starts his service after him, in another server, finishes before him.
- LIFO (last in, first out) or LCFS (last come, first served). In which the last client arrived will be the first processed. Again, the LIFO and LCFS disciplines are only equivalent for a single-server queue.
- RANDOM The next customer to be served is randomly selected from the queue.
- Round-Robin (cyclic). All customers in the queue take turns in the service, taking a portion Q of the service time and then are returned to the queue again until their service is fully completed. This service discipline was introduced to model computer systems.
- Processor Sharing. This is the limiting case of the Round Robin distribution when the time Q tends towards 0. All customers are served at the same time, but with a speed inversely proportional to the number of customers simultaneously present. If the server rate is equal to μ and that at a given time, there are n clients at the station, all the clients are served simultaneously with a $\frac{\mu}{n}$ rate (Be careful, saying that the n clients are served simultaneously does not mean that they will be released simultaneously).

D. Kendall's Notations

Kendall's notation normalizes the description of a single queue :

$$T/Y/C/K/m/Z$$

Where :

- T : Interarrival distribution
- Y : Service distribution
- C : Number of servers
- K : Capacity of the queue

- m : User population
- Z : Service discipline.

When the last three elements of Kendall's notation are not specified, it is assumed that $K = +\infty$, $m = +\infty$, and Z= FIFO. The parameter m specifies the maximum number of users expected to arrive in the queue.

E. Notion of customer classes

A queue can be run by different classes of customers. These different classes will be distinguished by :

- Different arrival processes ;
- Different service times;
- A different order in the queue according to their class.

To define a multi-class queue, you need to define for each customer class the arrival process and the associated service time distribution. It is also necessary to specify how the customers of the different classes are ordered in the queue. This allows the implementation of new priority service disciplines: PR (preemption) and HOL (hold on line). For example, in a queue with two classes of customers, the customers in class 1 will always be "in advance" of the customers in class 2. To characterize a priority service discipline, it is also necessary to specify if the service is prioritized: when a class 2 client is in service and a class 1 client arrives in the queue, does the client have to wait for the class 2 client to finish its service (without preemption: HOL) or is the class 2 client instantly put back in the queue (with preemption: PR) in order to leave the server for the clients of class 1 client. In the latter case, two further cases must be distinguished. Does the client of class 2 report the time of service performed or not?

F. Some examples of waiting systems

- $T/Y/C$: 1 queue and C stations offering the same service; the waiting customer goes to the first one that becomes free.
  Example: Toilets in a public place, post office counters...
- $T/Y/+\infty$: Infinity of servers; no waiting.
  Example: Telephone lines.
- 2 different stations, one faster than other.
  If both stations are free, the client goes with the probability p on one station and with the probability $1-p$ on the other.
  $\rightarrow$ If $p = \frac{1}{2}$, passing customers : they do not see the difference.

→ If $p = 1$, for all customers have the same choice
→ In fact, $p \in [0, 1]$: There are several factors that define the choice (quality of service, execution time ...)

- No waiting $T/Y/C/C$
  Example: Telephone exchange where unanswered calls are rejected.

- Systems $T/Y/C/K$ where $K > C$: limited queue length, depending on capacity.
  Example: Gas station, waiting room...

- Closed systems: number of customers is constant.
  Example: m machines running or broken down, C workers to repair them, n broken down.
  → $m > C$: most frequent case ;
  → $m < C$: not very interesting because there would be workers who might never be busy;
  → $m = C$: one worker per machine.

1.2.5 Operational performance parameters

A. Transient performance parameters

Considering the behavior of the system over a given period of time, for example between $t = 0$ and $t = \Theta$. Let X_t be the total number of customers in the system at the moment t. Looking at the behavior of the system over the time interval $[0, \Theta]$ is like looking at the "transient state" of the system as presented in figure 1.2.

The following "operational parameters" are defined :

- W_k : Stay time of the k^{th} client in the system: $W_k = D_k - A_k$.

- Θ : Total observation time.

- $T(n, \Theta)$: Total time during which the system contains n clients; of course we have:

$$\sum_{n \geq 0} T(n, \Theta) = \Theta \tag{1.1}$$

- $P(n, \Theta) = \frac{T(n,\Theta)}{\Theta}$: Proportion of time when the system contains n customers.

- $\alpha(\Theta)$: number of clients arriving in the system during the period $[0, \Theta]$.

- $\delta(\Theta)$: number of clients departing the system during the period $[0, \theta]$.

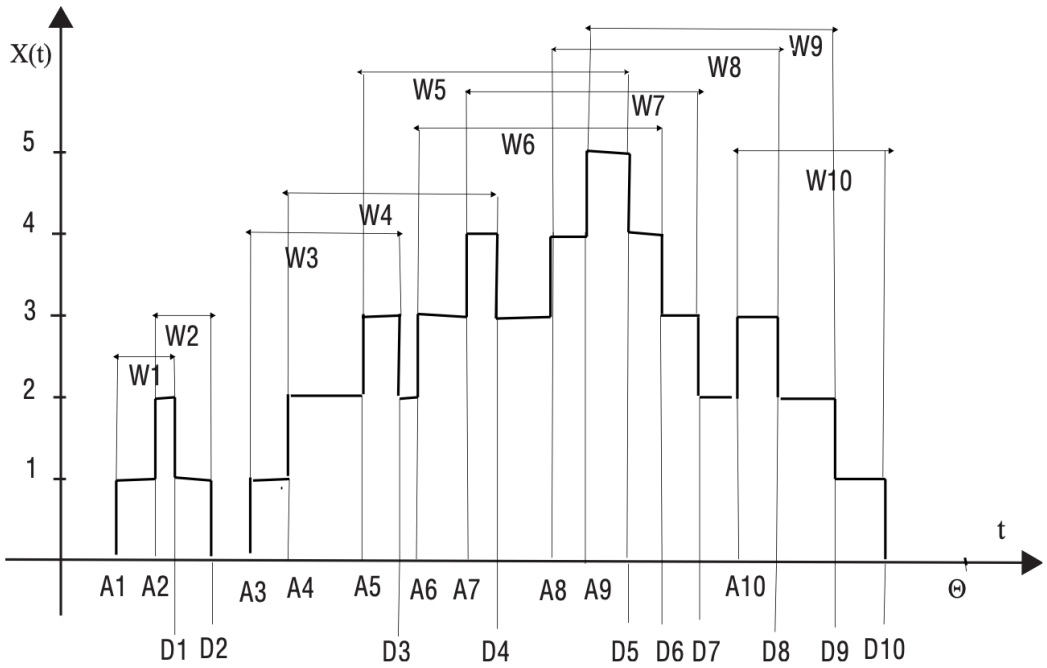

Figure 1.2 – The number of clients in the system in a period of time

From these parameters, the following transient performance parameters are defined:

Average incoming rate: The average incoming throughput is the average number of customers entering the system per unit of time. Over the observation period $[0, \Theta]$, it is then :

$$d_e(\Theta) = \frac{\alpha(\Theta)}{\Theta} \tag{1.2}$$

Average output rate: The average output rate is the average number of customers leaving the system per unit of time. Over the observation period $[0, \Theta]$, it is then :

$$d_e(\Theta) = \frac{\delta(\Theta)}{\Theta} \tag{1.3}$$

Average number of customers : The average number of clients present in the system is the time average of X_t (or $X(t)$) over the observation period $[0, \Theta]$, so it is the area under the curve of X_t:

$$L(\Theta) = \frac{1}{\Theta} \sum_{n=0}^{+\infty} nT(n, \Theta) = \sum_{n=0}^{+\infty} nP(n, \Theta) \tag{1.4}$$

Average stay time : The average stay time of a client in the system is, by definition, the average times of the clients who arrived in the system during the time interval. $[0, \Theta]$:

$$W(\Theta) = \frac{1}{\alpha(\Theta)} \sum_{k=1}^{\alpha(\Theta)} W_k \tag{1.5}$$

Rate of utilization U**:** For a single queue with a single server, in addition to the above parameters, average incoming rate, average output rate, average number of clients, average stay time, an additional performance parameter can be defined: the server utilization rate. It is defined as the proportion of time the server is occupied during the time $[0, \Theta]$:

$$U(\Theta) = \sum_{n=1}^{+\infty} P(n, \Theta) = 1 - P(0, \Theta) \tag{1.6}$$

B. Steady-state performance parameters

All of the above parameters define the performance of the system in the transient state (after a finite Θ time). In steady state, we will be interested in the existence and (possible) values of the limits when Θ tends towards infinity of all these parameters:

$$d_e = \lim_{\theta \to +\infty} d_e(\Theta), \quad d_s = \lim_{\theta \to +\infty} d_s(\Theta)$$

$$L = \lim_{\theta \to +\infty} L(\Theta), \quad W = \lim_{\theta \to +\infty} W(\Theta) \tag{1.7}$$

$$U = \lim_{\theta \to +\infty} U(\Theta)$$

C. Stability

Definition : A system is stable if and only if the average asymptotic output rate of the customers in the system is equal to the average input rate of the customers in the system:

$$\lim_{\theta \to +\infty} d_s(\Theta) = \lim_{\theta \to +\infty} d_e(\Theta) = d \tag{1.8}$$

Based on the previous relation, this implies that the total number of clients arriving $\alpha(\Theta)$ in the system during the interval $[0.\Theta]$, does not grow faster than the total number of clients departing the system $\delta(\Theta)$ when θ tends towards infinity:

$$\lim_{\theta \to +\infty} \frac{\delta(\Theta)}{\alpha(\Theta)} = 1 \tag{1.9}$$

Example : In order to fully understand the notion of stability of a single queue, we will consider the example of a $D/D/1$ queue (interarrivals and services are deterministic, meaning that customers arrive exactly every $t_a = \frac{1}{\lambda}$ units of time and all remain in service for a time $t_s = \frac{1}{\mu}$.
First consider the case where $t_a > t_s$ ($\lambda < \mu$), for example: $t_a = 10s$, i.e. $\lambda = 0.1$ clients/second, $t_s = 5s$, i.e. $\mu = 0.2$ clients/second. Over the observation period $[0.100[$, the average output rate $d_s = \frac{\delta(\Theta)}{\Theta} = \frac{10}{100} = 0,1$ is well equal to the average input rate $d_e = \lambda$. The queue is stable: when t tends towards infinity, the number of customers in the queue X_t remains finite (and in this case equal alternately to 0 or 1).
Now consider the case where $t_a < t_s$ ($\lambda > \mu$), for example : $t_a = 10s$, which $\lambda = 0,1$ clients/second, $t_s = 20$, and $\mu = 0,05$ clients/second. Over the observation period $[0, 100[$, the average output rate $d_s = \frac{\delta(\theta)}{\Theta} = \frac{5}{100} = 0,05$ is different from the average input rate $d_e = \lambda$. The queue is unstable: when t tends towards infinity, the number of customers in the queue, X_t tends towards infinity.

The problem is exactly the same if, instead of considering deterministic times, we consider random times (for example if we consider a $G/G/1$ queue, for which the interarrival time and the service time have a general distribution). The clients are grouped in the queue and the server cannot serve them. This phenomenon occurs when $t_a < t_s$ (so when $\lambda > \mu$) and is known as instability.

D. Ergodicity

Ergodicity is a very important concept in the field of stochastic processes. On the one hand, operational analysis is interested in a particular evolution of a system between two instants $t = 0$ and $t = \Theta$. This means that making Θ tend towards infinity and considering the limits of all the operational performance parameters is equivalent to being interested in the steady state of the system. In fact, it means looking at the steady state of a particular evolution of the system. Then it is possible to study different evolutions of the system. The question that must be asked of course: Do all the performance parameters considered have the same limit whatever the evolution of the system under consideration?

On the other hand, stochastic analysis will associate random variables and stochastic processes to the system:

- A_k: Random variable measuring the arrival time of the k^{th} client in the system;
- D_k : Random variable measuring the starting time of the k^{th} client of the system;
- W_k : Random variable measuring the time of stay of the k^{th} client in the system:

$$W_k = D_k - A_k$$

- (α_t) : Process measuring the number of customers arriving in the system at the moment t.
- (δ_t) : Process measuring the number of customers who have left the system at the moment t.
- (X_t) : Stochastic process measuring the number of customers in the system at the moment t:

$$X_t = \alpha_t - \delta_t$$

- $\pi_n(t)$: Probability that the system contains n clients at the moment t :

$$\pi_n(t) = P([X_t = n]).$$

Then, as was done in the operational analysis, all stochastic performance parameters can be calculated, in both steady state and transient conditions. The average number of customers present in the system at time t is calculated, for example, as follows:

$$L(t) = \sum_{n=0}^{+\infty} n\pi_n(t) \tag{1.10}$$

1.2.6 Little's Law

Little's law is a very general relation that applies to a large class of systems. It concerns only the permanent regime of the system. No hypothesis on the random variables that characterize the system (interarrival time, service time,...) is necessary. The only condition for the application of Little's law is that the system be stable. The flow rate of the system is then indifferently either the input rate or the output rate: $d_s = d_e = d$. Little's law is expressed as in the following property :

Theorem 1.1. *The average number of customers, the average time spent in the system, and the average throughput of a steady-state system are related as follows :*

$$L = W \times d \tag{1.11}$$

Proof. Consider, first of all, an observation duration Θ such that the system is empty at the beginning and at the end of the observation. Under these conditions, the number of clients who left the system during $[0, \Theta]$ is equal to the number of clients who arrived: $\delta(\Theta) = \alpha(\Theta)$. The quantities $L(\Theta)$ and $W(\Theta)$ can then be described as follows:

$$L(\Theta) = \sum_{n=0}^{+\infty} nP(n, \Theta) = \frac{1}{\Theta} \sum_{n=0}^{+\infty} nT(n, \Theta) \tag{1.12}$$

$$W(\Theta) = \frac{1}{\delta(\theta)} \sum_{k=1}^{\delta(\Theta)} W_k \tag{1.13}$$

Little's formula is based on the following result:

$$\sum_{n=0}^{+\infty} nT(n, \Theta) = \sum_{k=1}^{\delta(\Theta)} W_k \tag{1.14}$$

It can be formally demonstrated that the two summations involved in this equality represent two ways of calculating the area under the X_t curve. Since the system debit $d(\Theta)$ is the ratio of the number of customers exiting $\delta(\Theta)$ to the average observation time Θ , we immediately deduce from the three relations giving $L(\Theta)$, $W(\Theta)$ and $d(\Theta)$ that: $L(\Theta) = W(\Theta) \times d(\Theta)$. We can finally get rid of the hypothesis that the system is empty at the beginning and at the end of observation, making Θ tend towards infinity and reminding that if the system is stable.

$$\lim_{\Theta \to +\infty} \frac{\delta(\Theta)}{\alpha(\Theta} = 1 \tag{1.15}$$

■

Little's Law is very important in the analysis of queuing systems. It makes it possible to deduce one of the three quantities (L, W, d) according to the knowledge of the two others. It is applied in a wide variety of ways. Considering here the case of a simple queue with a single server, the Little's law can be applied in different ways.
Little's Law tells us that there is a relationship between the average number of clients in the queue (waiting or in service) and the average total time that a client stays in the queue (waiting time + service time):

$$L = W \times d \tag{1.16}$$

Little's law can also be applied by considering only the waiting in the queue (without the service). It then allows to link the average number of customers waiting L_q, to the average time of waiting of a customer before service W_q, by the relation:

$$L_q = W_q \times d$$

Also, we can apply Little's law by considering only the server. In this case, it connects the average number of clients in service L_S, to the average time of stay of a client in the server which is nothing else than the average time of service $\frac{1}{\mu}$, by the relation:

$$L_S = \frac{1}{\mu} \times d \tag{1.17}$$

Since there is never more than one client in service, L_S is simply expressed as:

$$L_S(\Theta) = 0P(0,\Theta) + 1[P(1,\Theta) + P(2,\Theta) + P(3,\Theta) + ...] = 1 - P(0,\Theta). \tag{1.18}$$

L_S is nothing more than the server utilization rate, U, defined as the probability that the server is busy (or the proportion of time the server is busy). Thus, in this case, Little's law is written :

$$U = \frac{1}{\mu} \times d$$

Three relations were obtained by applying Little's law successively to the whole system, to the queue alone and, finally, to the server alone. These three relations are of course not independent. We can indeed deduce one of them from the two others by noticing that :

$$W = W_q + \frac{1}{\mu} \text{ and } L = L_q + L_S = L_q + U \tag{1.19}$$

1.3 Markovian queue

The mathematical study of a waiting system is most often done by introducing an appropriate stochastic process.
→ First, we are interested in the number X_t of clients in the system at the moment t. According to the parameters that define the structure of the system, we try to calculate:

- The state probabilities $\pi_n(t) = P([X_t = n])$ which define the transitional regime of the process $(X_t)t \geq 0$;
- The stationary regime of the process, defined by:
  $\pi_n = \lim_{t \to +\infty} \pi_n(t) = \lim_{t \to +\infty} P([X_t = n])$.

→ From the stationary process distribution (X_t) $t \geq 0$, we can obtain other operating characteristics of the system such as :

- The average number of customers L in the system ;
- The average number of customers L_q in the queue;
- The average waiting time of a customer W_q ;
- Average stay time in the system W (waiting + service) ;
- The occupation rate of the service stations;
- Percentage of clients who could not be served;

- The duration of an active period, i.e. the time interval during which there is always at least one customer in the system...

It is important to note that the explicit calculation of the transitional regime is difficult, not to say impossible, for most of the models considered. With the exception of certain models that are particularly simple to manipulate, we will limit ourselves to determining the stationary regime of this waiting phenomenon.
The analysis results are basically average values, in particular, the minimum and maximum values, are not significant, for example, for the size of storage areas. In this case, the knowledge of the state probabilities π_n, can be very useful.

Markovian queues

The Makovian queues are those for which the interarrivals and service times are exponential. Their Kendall notation will be in the form $M/M/\ldots$ (M for markovian ...)

A. Process of birth and death in general

This process was discussed in the previously indirectly. Recall that it is characterized by a number n of entities that evolves as follows:

- The arrivals and departures of entities follow exponential laws of $\lambda(n)$ and $\mu(n)$ rates respectively
- The probability of two events occurring in a time interval dt is very small (regularity hypothesis: two events cannot occur at the same time).

There is a transition to a nearby state, either by the arrival of a client (birth) or by the departure of a client (death).
If $\pi_n(t)$ is the probability that there are n clients in the system at time t, Kolmogorov's equation is written, for $n > 0$:

$$\pi_n(t+dt) = (1 - (\lambda_n + \mu_n)\, dt)\, \pi_n(t) + \mu_{n+1}\pi_{n+1}(t)dt + \lambda_{n-1}\pi_{n-1}(t)dt + o(dt) \qquad (1.20)$$

that is to say, by making dt tend towards 0, for $n > 0$:

$$\frac{d}{dt}\pi_n(t) = -(\lambda_n + \mu_n)\, \pi_n(t) + \mu_{n+1}\pi_{n+1}(t) + \lambda_{n-1}\pi_{n-1}(t) \qquad (1.21)$$

In the same way, we obtain for $n = 0$:

$$\frac{d}{dt}\pi_0(t) = -\lambda_0\pi_0(t) + \mu_1\pi_1(t) \qquad (1.22)$$

It is practically impossible to calculate the general expression of $\pi_n(t)$, except by simulation from the values $\pi_n(0)$ which characterize the initial state of the system. However, if we assume that a steady state can be established, the probabilities become independent of time and we then have :

$$\pi_n = \pi_0 \prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}} \tag{1.23}$$

with

$$\pi_0 = \frac{1}{1 + \sum_{n=1}^{+\infty} \prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}}} \tag{1.24}$$

The Poisson process is a special case of the process of birth and death for which $\mu_n = 0$ and $\lambda_n = Cte = \lambda$ but, in this case, there is no stationary regime.
The differential equations are then written:

$$\frac{d}{dt}\pi_0(t) = -\lambda \pi_0(t) \tag{1.25}$$

where $\pi_0(t) = e^{-\lambda t}$

$$\frac{d}{dt}\pi_n(t) = -\lambda \left(\pi_n(t) - \pi_{n-1}(t)\right) \tag{1.26}$$

whose solution is $\pi_n(t) = \frac{(\lambda t)^n e^{-\lambda t}}{n!}$

B. The Queue $M/M/1$

This queue is characterized by an arrival poissonian rate λ and a service time with exponential rate μ. Let $\rho = \frac{\lambda}{\mu}$. Queuing can be seen as a process of birth and death, for which :

$$\lambda_n = \lambda \qquad \mu_n = \begin{cases} \mu & \text{if } n \neq 0 \\ 0 & \text{if } n = 0 \end{cases} \tag{1.27}$$

The probability of state (taking into account that $\rho < 1$ to be a permanent regime) is given by :

$$\begin{cases} \pi_n = \pi_0 \rho^n \\ \pi_0 = \frac{1}{\sum_{n=0}^{+\infty} \rho^n} = 1 - \rho \end{cases} \tag{1.28}$$

So

$$\pi_n = (1 - \rho)\rho^n \tag{1.29}$$

All performance parameters are calculated if the queue is stable. ($\lambda < \mu$ i,e $\rho < 1$) and

for the stationary regime of the queue.

Throughput d: Here $d = \lambda$ as $\lambda_n = \lambda$ for any $n \geq 0$. Another way of looking at things is to notice that the service is carried out with a μ rate in each state where the system contains at least one client :

$$d = \text{ Proba([file not empty])}\mu = \sum_{n=1}^{\infty} \pi_n \mu = [1 - \pi_0]\,\mu = \rho\mu = \lambda \tag{1.30}$$

We find that if the queue is stable, the average output flow rate is equal to the average input flow rate.

Server utilization rate U : By definition, the utilization rate is the probability that the server in the queue will be busy.

$$U = \sum_{n=1}^{+\infty} \pi_n = 1 - \pi_0 = \rho = \frac{\lambda}{\mu} \tag{1.31}$$

Average number of customers L: The average number of customers is calculated from the stationary probabilities in the following way:

$$\begin{aligned} L &= \sum_{n=1}^{+\infty} n\pi_n = \sum_{n=1}^{+\infty} n(1-\rho)\rho^n = \rho(1-\rho)\sum_{n=0}^{+\infty}(n+1)\rho^n = \rho(1-\rho)\left(1 + 2\rho + 3\rho^2 + \cdots\right) \\ &= \rho(1-\rho)\frac{d}{d\rho}\left(\rho + \rho^2 + \rho^3 + \cdots\right) = \rho(1-\rho)\frac{d}{d\rho}\left(\frac{1}{1-\rho} - 1\right) \end{aligned} \tag{1.32}$$

So

$$L = \frac{\rho}{1-\rho} \tag{1.33}$$

Average stay time W : This parameter is obtained using Little's law:

$$W = \frac{L}{d} = \frac{1}{\mu(1-\rho)} \tag{1.34}$$

which can be decomposed into :

$$W = \frac{1}{\mu} + \frac{\rho}{\mu(1-\rho)} \tag{1.35}$$

The average time spent in the queue is calculated as follows W_q :

$$W_q = \frac{\rho}{\mu(1-\rho)} \tag{1.36}$$

Average number of customers in the queue L_q :

$$L_q = \lambda W_q = \frac{\rho^2}{1-\rho} \tag{1.37}$$

Proof of Little's formulas in the case of $M/M/1$ **:** We are looking for the average waiting time $W_q = \mathbb{E}(T_q)$ of an individual: this is a function of the number of customers already present when he arrives. Either In the event "there are n customers in the system when the individual arrives". In this case the following occurs

$$W_q = \mathbb{E}\left(T_q\right) = \sum_{n=0}^{+\infty} \mathbb{E}\left(T_q/E_n\right) P\left(E_n\right) \tag{1.38}$$

with $\mathbb{E}\left(T_q/E_n\right) = \frac{n}{\mu}$ (no memory of the exponential law) and $P\left(E_n\right) = \pi_n = (1-\rho)\rho^n$ where $\rho = \frac{\lambda}{\mu}$. Then we have

$$W_q = \sum_{n=1}^{+\infty} \frac{n}{\mu}(1-\rho)\rho^n = \frac{L}{\mu} \text{ and } W = W_q + \frac{1}{\mu} = \frac{L+1}{\mu}. \tag{1.39}$$

Where

$$L_q = \sum_{n=1}^{+\infty}(n-1)\pi_n = \sum_{n=1}^{+\infty} n\pi_n - \sum_{n=1}^{+\infty} \pi_n = L - (1-\pi_0) = L - \rho \tag{1.40}$$

as

$$L = \sum_{n=1}^{+\infty} n\pi_n = \frac{\rho}{1-\rho}, \tag{1.41}$$

So

$$L = \rho(L+1) = \rho\mu W \tag{1.42}$$

and we have

$$L = \lambda W \tag{1.43}$$

then,

$$L_q = L - \frac{\lambda}{\mu} = \lambda\left(W - \frac{1}{\mu}\right) = \lambda W_q. \tag{1.44}$$

C. The Queue $M/M/1/K$

Consider a single-server system identical to the $M/M/1$ queue except that the capacity of the queue is finite. So we still have the following assumptions: the process of arriving clients in the queue is a poisson process with rate λ and the service time of a client is an exponential random variable of rate μ.

Let K be the capacity of the queue: this is the maximum number of customers that can be present in the system, either waiting or in service. When a customer arrives while there are already K customers present in the system, he is rejected. This system is known as the $M/M/1/K$ queue. The E state space is now finite: $E = \{0, 1, 2, ..., K\}$. As the capacity of the queue is limited. The number of clients in the queue can therefore never go to infinity. Moreover, as soon as a client is allowed to enter, he will leave and his time in the queue is finite, since it corresponds to the service time of all the clients in front of him and this number is limited by K.
Over a very long time, the exit flow will be equal to the entry flow, which corresponds well to the unconditional stability of the system.

The birth and death process modelling this type of queue defined as follows:

$$\begin{aligned} \lambda_n &= \begin{cases} \lambda & \text{if } n < K \\ 0 & \text{if } n = K \end{cases} \\ \mu_n &= \begin{cases} \mu & \text{if } n \neq 0 \\ 0 & \text{if } n = 0 \end{cases} \end{aligned} \tag{1.45}$$

The integration of the recurrent equation to calculate π_n is then done as follows:

$$\pi_n = \begin{cases} \pi_0 \rho^n & \text{if } n \leq K \\ 0 & \text{if } n > K \end{cases} \tag{1.46}$$

$$\pi_0 = \frac{1}{\sum_{n=0}^{K} \rho^n} = \frac{1-\rho}{1-\rho^{K+1}} \quad \text{if } \lambda \neq \mu \left(\text{ and } \frac{1}{K+1} \text{ if } \lambda = \mu \right) \tag{1.47}$$

Throughput d : The system throughput can be calculated in two equivalent ways: either by measuring the rate at which customers leave the server d_s, or by measuring the actual rate at every customers are accepted into the system d_e. Of course, it is expected that these two rates will be equal.

The output rate of the server is equal to μ as soon as the queue is not empty:

$$d_s = \text{ Proba ([file not empty])}\mu = \sum_{n=1}^{K} \pi_n \mu = [1 - \pi_0]\,\mu = \frac{\rho - \rho^{K+1}}{1-\rho^{K+1}}\mu \tag{1.48}$$

The effective rate of entry into the queue is equal to λ as soon as a customer arrives when the queue is not full:

$$d_e = \text{Proba([queue not full at the moment of arrival]} \tag{1.49}$$

$$\lambda = \sum_{n=0}^{K-1} \pi_n \lambda = [1 - \pi_K]\lambda = \frac{1-\rho^K}{1-\rho^{K+1}}\lambda \tag{1.50}$$

Since $\rho = \frac{\lambda}{\mu}$, we have $d_e = d_s = d$, where d is the average throughput of the queue (entry or exit):

$$d = \frac{1-\rho^K}{1-\rho^{K+1}}\lambda \tag{1.51}$$

Note that when K tends towards the infinite, We find the results of the $M/M/1$, i.e. $d = \lambda$, provided that $\rho < 1$, which corresponds to the condition of stability of the $M/M/1$.

Server utilization rate $U(K)$:

$$U(K) = \sum_{n=1}^{K} \pi_n = 1 - \pi_0 = \frac{\rho - \rho^{K+1}}{1-\rho^{K+1}} = \rho\frac{1-\rho^K}{1-\rho^{K+1}} \tag{1.52}$$

Thus, in the case of a queue with limited capacity, the utilisation rate is no longer equal to ρ. Indeed, the utilisation rate is always equal to the ratio of the average input flow rate to the average service rate (Little's law): $U = \frac{d}{\mu}$. But here d is no longer equal to λ.
Note that when $K \to +\infty$, $U(K)$ tends towards ρ if $\rho < 1$, and towards 1 if $\rho > 1$.

Average number of customers L:

$$\begin{aligned} L &= \sum_{n=0}^{K} n\pi_n = \frac{1-\rho}{1-\rho^{K+1}} \sum_{n=0}^{K} n\rho^n = \frac{\rho(1-\rho)}{1-\rho^{K+1}} \sum_{n=1}^{K} n\rho^{n-1} \\ &= \frac{\rho(1-\rho)}{1-\rho^{K+1}} \frac{d}{d\rho}\left(\frac{1-\rho^{K+1}}{1-\rho} - 1\right) = \frac{\rho(1-\rho)}{1-\rho^{K+1}} \frac{1-(K+1)\rho^K + K\rho^{K+1}}{(1-\rho)^2} \\ &= \frac{\rho}{1-\rho} \frac{1-(K+1)\rho^K + K\rho^{K+1}}{1-\rho^{K+1}} \end{aligned} \tag{1.53}$$

Again, when K tends towards infinity and $\rho < 1$, the results of the Queue $M/M/1$:

$$L = \frac{\rho}{1-\rho} \tag{1.54}$$

Average stay time W: Here we consider the average stay of a client actually admitted to the queue. This value can be obtained by application of Little's law:

$$W = \frac{L}{d} \tag{1.55}$$

Particular case where $\lambda = \mu : \pi_n = \pi_0 = \frac{1}{K+1}$ for $n \leq K$

$$U(K) = 1 - \pi_0 = \frac{K}{K+1} \text{ and } d = \frac{K\lambda}{K+1} \tag{1.56}$$

Average number of customers in the queue : $L = \sum_{n=0}^{K} \frac{n}{K+1} = \frac{K}{2}$

Average time in the waiting system: $W = \frac{L}{d} = \frac{K+1}{2\lambda}$

Proof of Little's formulas in the case of $M/M/1/K$ **:**

Proof. Here we have

$$\mathbb{E}\left(T_q\right) = \sum_{n=1}^{K-1} \frac{n}{\mu} \pi_n^* = \frac{1}{\mu}\left(\frac{L - K\pi_K}{1 - \pi_k}\right), \tag{1.57}$$

with $\pi_n^* = \frac{\pi_n}{1-\pi_K}$ (rejected customer if there is already K) $clients$

$$\begin{aligned} L - K\pi_K =& \rho \frac{1 - (K+1)\rho^K + K\rho^{K+1} - K\rho^{K-1}(1-\rho)^2}{(1-\rho)\left(1-\rho^{K+1}\right)} \\ =& \rho \frac{1 - (K+1)\rho^K + K\rho^{K+1} - K\rho^{K-1}\left(1 - 2\rho + \rho^2\right)}{(1-\rho)\left(1-\rho^{K+1}\right)} = \rho \frac{1 - K\rho^{K-1} + (K-1)\rho^K}{(1-\rho)\left(1-\rho^{K+1}\right)} \\ 1 - \pi_K = 1 - & \frac{\rho^K(1-\rho)}{1-\rho^{K+1}} = \frac{1 - \rho^{K+1} - \rho^K + \rho^{K+1}}{1-\rho^{K+1}} = \frac{1-\rho^K}{1-\rho^{K+1}} \end{aligned} \tag{1.58}$$

$$\begin{aligned} W = W_q + \frac{1}{\mu} &= \frac{1}{\mu}\left[\frac{\rho\left(1 - K\rho^{K-1} + (K-1)\rho^K\right)}{(1-\rho)\left(1-\rho^K\right)} + \frac{1 - \rho - \rho^K + \rho^{K+1}}{(1-\rho)\left(1-\rho^K\right)}\right] \\ &= \frac{1}{\mu}\left[\frac{1 - (K+1)\rho^K + K\rho^{K+1}}{(1-\rho)\left(1-\rho^K\right)}\right] = \frac{L\left(1-\rho^{K+1}\right)}{\rho\mu\left(1-\rho^K\right)} = \frac{L}{\lambda(1-\pi_K)} \end{aligned} \tag{1.59}$$

Where $d = \lambda \sum_{n=0}^{K-1} \pi_n = \lambda(1 - \pi_K)$ and we have $L = d \times W$

Also, $L - L_q = \sum_{n=1}^{K} n\pi_n - \sum_{n=1}^{K}(n-1)\pi_n = \sum_{n=1}^{K} \pi_n = 1 - \pi_0$ so, as $\bar{\lambda} = \bar{\mu} = \mu(1 - \pi_0) = d$,

$$L_q = L - (1 - \pi_0) = d \times W - \frac{d}{\mu} = d\left(W - \frac{1}{\mu}\right) = d \times W_q \tag{1.60}$$

■

D. The Queue $M/M/C$

We consider a system identical to the $M/M/1$ queue except that it has C identical and independent servers. We keep the hypotheses: arrival of clients with λ rate and the exponential service time with μ rate (for each of the servers). This system is known as the $M/M/C$ queue. The state space E is, as for the infinite $M/M/1$: $E = \{0, 1, 2, ...\}$. We have a birth and death process of rate :

$$\lambda_n = \lambda \tag{1.61}$$

$$\mu_n = \begin{cases} 0 & \text{if } n = 0 \\ n\mu & \text{if } 0 < n < C \\ C\mu & \text{if } n \geq C \end{cases} \tag{1.62}$$

Indeed, when the process is in a state $n < C$, all clients are in service and are able to quit the queue. To pass from n clients to $n-1$ clients in a dt time, it is necessary that one of the n clients finishes his service and the others do not finish theirs, this can happen for the first client, second client ,..., or n_{th} client. To be precise, it is also necessary to add that no client arrives during this time dt. The characteristic property of the exponential law tells us that the probability that a client will complete his service in a time dt is $\mu dt + o(dt)$, the probability that a client will not complete his service is $1 - \mu dt + o(dt)$ and the probability that no client arrives is $1 - \lambda dt + o(dt)$. The required probability is calculated as follows:

$$p_{n,n-1}(dt) = \left(\sum_{j=1}^{n} (\mu dt + o(dt))(1 - \mu dt + o(dt))^{n-1} \right) (1 - \lambda dt + o(dt)) \tag{1.63}$$

A Taylor polynomial to the first order immediately gives us that

$$p_{n,n-1}(dt) = n\mu dt + o(dt) \tag{1.64}$$

The transition rate from the n state to the $n-1$ state is therefore equal to $n\mu$. Similarly, when $n \geq C$, only C clients are in service and are able to leave the queue, thus switching the process from the n state to the $n-1$ state. The corresponding transition rate is equal to $C\mu$. In all cases, a transition from a state n to a state $n+1$ corresponds to a client arrival, that is to say in a time dt, with a probability $\lambda dt + o(dt)$. The transition rate is equal to λ .

The stability condition here is $\lambda < C\mu$ and expresses the fact that the average number of clients arriving in the queue per unit of time must be less than the average number of clients that the servers in the queue are able to process per unit of time.

The π_n can be calculated as follows:

$$\begin{cases} \pi_{n-1}\lambda = \pi_n n\mu & \text{for } n = 1, \cdots, C-1 \\ \\ \pi_{n-1}\lambda = \pi_n C\mu & \text{for } n = C, C+1, \cdots \end{cases} \tag{1.65}$$

Let

$$\begin{cases} \pi_n = \frac{\rho}{n}\pi_{n-1} & \text{for } n = 1, \cdots, C-1 \\ \\ \pi_n = \frac{\rho}{C}\pi_{n-1} & \text{for } n = C, C+1, \cdots \end{cases} \quad \text{where } \rho = \frac{\lambda}{\mu} \tag{1.66}$$

We can then express all the probabilities as a function of π_0 :

$$\begin{cases} \pi_n = \frac{\rho^n}{n!}\pi_0 & \text{for } n = 1, \cdots, C-1 \\ \\ \pi_n = \frac{\rho^n}{C!C^{n-C}}\pi_0 & \text{for } n = C, C+1, \cdots \end{cases} \tag{1.67}$$

The normalization condition allows us to calculate the probability π_0. We can easily verify that the convergence condition of this series is identical to the stability condition of the queue, i.e. $\lambda < C\mu$.

$$\pi_0 = \frac{1}{\sum_{n=0}^{C-1} \frac{\rho^n}{n!} + \frac{\rho^C}{(C-1)!(C-\rho)}} \tag{1.68}$$

When $C = 1$, We find well the results of the queue $M/M/1$:

$$\pi_n = (1-\rho)\rho^n \tag{1.69}$$

All performance parameters can be calculated if the queue is stable. ($\lambda < C\mu$) so $\rho < C$.

Throughput d : The service is performed with a rate of $n\mu$ in each state where the system contains less than C customers and with a rate of $C\mu$ in each state where the system contains more than C customers :

$$d = \sum_{n=1}^{C-1} \pi_n n\mu + \sum_{n=C}^{+\infty} \pi_n C\mu \tag{1.70}$$

By replacing the expressions obtained for the probabilities π_n and π_0, we find that the queue is stable, the average output rate is equal to the average input rate:

$$d = \lambda$$

For the $M/M/C$ queue, it is simpler (in terms of the calculations involved) to first calculate the average stay time and then deduct the average number of clients.

Average stay time W **:** The average stay of a client can be divided into the average time in the queue and the average service time. It is then sufficient to apply Little's law to the queue alone:

$$W = W_q + S = \frac{L_q}{d} + \frac{1}{\mu} = \frac{L_q}{\lambda} + \frac{1}{\mu} \tag{1.71}$$

It then remains to calculate the average number of customers waiting in the queue, L_q :

$$\begin{aligned} L_q &= \sum_{n=C}^{+\infty}(n-C)\pi_n = \sum_{n=C}^{+\infty}(n-C)\frac{\rho^n}{C!C^{n-C}}\pi_0 = \frac{\rho^{C+1}}{C!C}\sum_{n=C}^{+\infty}(n-C)\left(\frac{\rho}{C}\right)^{n-C-1}\pi_0 \\ &= \frac{\rho^{C+1}}{C!C}\frac{1}{\left(1-\frac{\rho}{C}\right)^2}\pi_0 = \frac{\rho^{C+1}}{(C-1)!(C-\rho)^2}\pi_0 \end{aligned} \tag{1.72}$$

The average stay time can be calculated as follows

$$W = \frac{\rho^C}{\mu(C-1)!(C-\rho)^2}\pi_0 + \frac{1}{\mu} \tag{1.73}$$

Average number of customers L**:** The average number of clients is then obtained by applying Little's law to the whole queue:

$$L = W \times d = W \times \lambda = \frac{\rho^{C+1}}{(C-1)!(C-\rho)^2}\pi_0 + \rho \tag{1.74}$$

E. The Queue $M/M/+\infty$

A system is considered to be composed of an unlimited number of identical and independent servers. As soon as a customer arrives, it is instantly in service. In this specific queue, there is no waiting time. It is always assumed that the customer arrival process is poissonian of rate λ and that the service times are exponential of rate μ (for all servers). This system is known as the $M/M/\infty$ queue. As was done for the $M/M/C$ queue, it can easily be shown that the transition rate from any n state to the $n-1$ state is equal to $n\mu$ and corresponds to the exit rate of one of the n clients in service. Similarly, the rate of transition from any n state to the $n+1$ state is equal to λ and corresponds to the rate of arrival of a client.
Intuitively, the processing capacity of the queue is infinite as any new customer arriving at the entrance of the queue is instantly processed. The stability condition that "the average number of customers arriving in the queue per unit of time must be less than the capacity of the queue" is always verified.
Let π_n be the stationary probability of being in the n state. The equilibrium equations give us:

$$\pi_{n-1}\lambda = \pi_n n\mu \qquad \text{for } n = 1, 2, \cdots \tag{1.75}$$

Let

$$\pi_n = \frac{\rho}{n}\pi_{n-1} \qquad \text{for } n = 1, 2, \cdots, \text{ where } \rho = \frac{\lambda}{\mu} \tag{1.76}$$

All probabilities can then be expressed in terms of π_0 :

$$\pi_n = \frac{\rho^n}{n!}\pi_0 \qquad \text{for } n = 1, 2, \cdots$$

The condition of normalization gives us immediately π_0 :

$$\pi_0 = \frac{1}{\sum_{n=0}^{+\infty} \frac{\rho^n}{n!}} = e^{-\rho} \tag{1.77}$$

Note that the series $\sum_{n=0}^{+\infty} \frac{\rho^n}{n!}$ converges for all values of ρ (so to λ and to μ), which is consistent with the unconditional stability of the queue. The final result is :

$$\pi_n = \frac{\rho^n}{n!}e^{-\rho} \qquad \text{for } n = 1, 2, \cdots \tag{1.78}$$

Throughput d : The service is provided at a rate of $n\mu$ in each state where the system contains n clients:

$$d = \sum_{n=1}^{+\infty} \pi_n n\mu = e^{-\rho} \sum_{n=1}^{+\infty} \frac{\rho^n}{(n-1)!}\mu = e^{-\rho}\rho e^{\rho}\mu = \rho\mu = \lambda \tag{1.79}$$

The unconditional stability of the queue is refound.

Average number of customers L:

$$L = \sum_{n=1}^{+\infty} n\pi_n = e^{-\rho} \sum_{n=1}^{+\infty} \frac{\rho^n}{(n-1)!} = e^{-\rho}\rho e^{\rho} = \rho \tag{1.80}$$

Average stay time W : Intuitively, the average time spent in the system is reduced to the average service time, which is $\frac{1}{\mu}$. We can demonstrate this result again by using Little's law:

$$W = \frac{L}{d} = \frac{\rho}{\lambda} = \frac{1}{\mu} \tag{1.81}$$

1.4 Dynamic programming

Every day, we interact with our environment and learn something from it. In some situations, we are supposed to take decisions, regarding the state of our environment. dynamic programming explore a computational approach to learning from this interaction. Precisely, it can be defined

as a sub-area of machine learning concerned with how to take a decision according to a specific situation in order to maximise the reward obtained by taking the decision. In dynamic programming, the entity which takes the decision is called agent or controller . It can be a human, a robot, a part of a machine or anything susceptible to take a decision, and the environment is what is all around the agent and which can influence its decision. A good way to understand the concept of dynamic programming is to see some examples of situations at which it can be applied.

In a chess game, the agents are the players and they act by considering a goal which is to win. So, every action of a player is considered as a decision which influences its environment by putting the game at a new state. dynamic programming can be applied to learn how to make good decision in every possible states of the game. dynamic programming can also be used in a more technical situation, like the approach of learning a performant decision strategy to control the functioning of allocation of virtual Machines. We will discuss this example of use of dynamic programming in this work. In these examples, the agent has a goal to be reached and acts according to it and the available state of its environment.

dynamic programming has two main features [1]:

- Trial and error search:

  The agent has to discover by itself which actions lead to the biggest reward, by trying them.

- Delayed reward:

  It considers more strongly the goal of a long term reward , rather than maximising the immediate reward.

1.4.1 Elements of dynamic programming

Dynamic programming has four principal elements:

1. The policy: It is a map from the perceived states of the environment to the set of actions that can be taken.

2. The reward function: It is a function that assigns a real number to each pair (s, a) where s is a state in the set of possible states S and a is an action in the actions set A.

$$\begin{aligned} R : S \times A &\longrightarrow \mathbb{R} \\ (s, a) &\longmapsto R(s, a) \end{aligned}$$

   $R(s, a)$ represents the reward obtained by the agent when performing the action a at state s. Generally, we note the value of the reward at time t as r_t.

3. The value function: It is the total amount of expected rewards from a specific state of the system, from a specific time, and over the future. It is an evaluation of what is good in the long term. The value taken by this function is generally named return and is denoted R_t at time t.

   We have two ways to count the return:

   - We can sum up all the rewards. This is especially used for episodic tasks, when there is a time stop T. In that case:

     $$R_t = r_{t+1} + r_{t+2} + \ldots + r_T$$

   - Concept of discounting: It is used to weight present reward more heavily than future ones with a discount rate $\gamma, 0 < \gamma \geq 1$. The discount rate γ is necessary to have a present value of the future rewards. In this case, the return is given by the infinite sum:

     $$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

     Since the discount rate γ is less than 1 and the series of rewards r_k bounded, this infinite sum has a finite value. If γ tends to 0, this agent is said to be myopic: it sees only immediate rewards. When γ tends to 1, the future rewards are more taken into account.

4. A model of environment: It is set to plan the future behaviour of the environment.

   The most important task of dynamic programming is determining a good policy which maximises the return. Markov Decision Processes (MDPs) are a mathematical model for dynamic programming that is commonly used nowadays. Current theories of dynamic programming are usually restricted to finite MDPs.

1.4.2 Markov decision processes

A. Definition and characteristics

A learning process is said to have the Markov property when it is retaining all relevant information about the future. Precisely, when all needed information to predict the future can be available at the present, and the state of the system at the present can resume past information. So, in that case, we can predict the future values of the rewards simply by iteration.

This statement can be written as:

$$P\left\{S^{t+1}=s', r_{t+1}=r \mid S^t, a^t, r_t, S^{t-1}, a^{t-1}, r_{t-1}, \ldots\right\} = P\left\{S^{t+1}=s', r_{t+1}=r \mid S^t, a^t, r_t\right\} \tag{1.82}$$

where

$$P\left\{S^{t+1}=s', r_{t+1}=r \mid S^t, a^t, r_t, S^{t-1}, a^{t-1}, r_{t-1}, \ldots\right\} \tag{1.83}$$

is the probability that at time $(t+1)$, we are at the state s', wining a reward r given that we took the action $a^t, a^{t-1}, a^{t-2}, \ldots$ at the states $S^t, S^{t-1}, S^{t-2}, \ldots$ respectively.

A decision making which has this property can be treated by MDPs. When the space of the states is finite, the process is referred to as finite MDP, and consists of a 4-tuple: (S, A, R, P)

- The set of states S: It is the finite set of all possible states of the system.
- The finite set of actions A.
- The reward function: $R(s, a)$ depends on the state of the system and the action taken. We assume that the reward function is bounded.
- The Markovian transition model: It is represented by the probability of going from a state s to another state s' by doing the action a: $P(s'|s, a)$.

B. Policy and value functions

Recall that the main task of a dynamic programming is finding a policy that optimises the value of rewards. This policy can be represented by a map π:

$$\begin{aligned} \pi : S &\longrightarrow A \\ s &\longmapsto \pi(s) \end{aligned} \tag{1.84}$$

where $\pi(s)$ is the action the agent takes at the state s.
Most MDP algorithms are based on estimating value functions. The state-value function according to a policy π is given by:

$$\begin{aligned} v^\pi : S &\longrightarrow \mathbb{R} \\ s &\longmapsto v^\pi(s) \end{aligned} \tag{1.85}$$

If we note here and in the remaining of this work $N = S$, the cardinal of the set of states S, the state-value function can be represented as a vector $V \in \mathbb{R}^N$ where each component corresponds respectively to the value $v^\pi(s)$ at each state $s \in S$.

Let us note $E_\pi\{R_t\}$, the expected value of the return R_t when the policy π is followed in the process. The value of the state-value function $v_\pi(s)$ at a particular state s is in fact the

expected discounted return when starting at the state s and following the policy π.

$$v^{\pi}(s) = E_{\pi}\left\{R_t \mid S^t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid S^t = s\right\} \tag{1.86}$$

The return R_t has the recursive property that:

$$\begin{aligned} R_t &= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \sum_{k=1}^{\infty} \gamma^k r_{t+k+1} \\ &= r_{t+1} + \sum_{k=0}^{\infty} \gamma^{k+1} r_{t+(k+1)+1} = r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+(k+1)+1} \\ &= r_{t+1} + \gamma R_{t+1} \end{aligned} \tag{1.87}$$

Thus,

$$\begin{aligned} v^{\pi}(s) &= E_{\pi}\left\{R_t \mid S^t = s\right\} = E_{\pi}\left\{r_{t+1} + \gamma R_{t+1} \mid S^t = s\right\} \\ &= E_{\pi}\left\{r_{t+1} \mid S^t = s\right\} + \gamma E_{\pi}\left\{R_{t+1} \mid S^t = s\right\} \end{aligned} \tag{1.88}$$

But $E_{\pi}\{r_{t+1}|S_t = s\}$ is the expected value of the reward r_{t+1} when the policy π is followed and such that at time t, the state is s. This is the same as the reward obtained by doing the action $\pi(s)$ at the state s : $R(s, \pi(s))$. Therefore, equation 1.88 becomes:

$$v^{\pi}(s) = R(s, \pi(s)) + \gamma E_{\pi}\left\{R_{t+1} \mid S^t = s\right\} \tag{1.89}$$

But $E_{\pi}\{R_{t+1}|S^t = s\}$ is the expected value of R_{t+1} known that $S^t = s$. and by definition of the conditional expectation, we get:

$$E_{\pi}\left\{R_{t+1} \mid S^t = s\right\} = \sum_{s'}\left[P\left(s' \mid s, \pi(s)\right) E_{\pi}\left\{R_{t+1} \mid S^{t+1} = s'\right\}\right] \tag{1.90}$$

Putting the equation 1.90 into the equation 1.89 yields to:

$$v^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'}\left[P\left(s' \mid s, \pi(s)\right) E_{\pi}\left\{R_{t+1} \mid S^{t+1} = s'\right\}\right] \tag{1.91}$$

And in fact,

$$E_{\pi}\left\{R_{t+1} \mid S^{t+1} = s'\right\} = v^{\pi}\left(s'\right) \tag{1.92}$$

Finally, we define the state-value function:

$$v^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'}\left[P\left(s' \mid s, \pi(s)\right) v^{\pi}\left(s'\right)\right] \tag{1.93}$$

One can also define another form of value function called state-action value function according to a policy π : $Q^{\pi}(s, a)$ which depends not only on the state but on the action taken as well.

Definition : [2] The decision process operator T_π for a policy π is defined by:

$$T_\pi v(s) = R(s, \pi(s)) + \gamma \sum_{s'} \left[P\left(s' \mid s, \pi(s)\right) v\left(s'\right) \right] \tag{1.94}$$

It yields from the definition of the state-value function in equation 1.93 that $v^\pi(s)$ is the fixed point of the decision process operator T_π. That is to say, $T_\pi v^\pi(s) = v^\pi(s)$

Definition : [2] The Bellman operator T^* is defined as:

$$T^* v(s) = \max_a \left[R(s, a) + \gamma \sum_{s'} P\left(s' \mid s, a\right) v\left(s'\right) \right] \tag{1.95}$$

We denote v^* , the fixed point of the Bellman operator. So, $T^* v^*(s) = v^*(s)$. And $v^*(s)$ is the maximum, according to actions, of $v(s)$ for a particular state s. When considering this maximum for every state $s \in S$, we can therefore define v^*, which is called the optimal state-value function.

Then, finding the appropriate action a which maximises the reward at each state is equivalent to find an action a verifying $T^* v^*(s) = v^*(s)$. We denote this action $greedy(v)(s)$

$$\text{greedy}(v)(s) = \arg\max_a \left[R(s, a) + \gamma \sum_{s'} P\left(s' \mid s, a\right) v\left(s'\right) \right] \tag{1.96}$$

It is worth nothing that for each state s of the system, we have one equation like 1.96. Therefore, finding the policy π to get the right action to do for every state of the system is now equivalent to solving N equations, each given by 1.96, with N unknowns.
Solving a MDP consists of looking for an efficient algorithm to solve this system.

1.4.3 Dynamic programming methods

A. Solving a MDP by value iteration algorithm

Recall that solving a MDP is equivalent to solving a system of N Bellman equations with N unknowns, where N is the number of possible states of the system. That is solving non linear equations. Then, iterative approach is suitable. Value iteration, also called Backward induction is one of the simplest method to solve a finite MDP. It was established by Bellman in 1957. Because our main objective in MDP is to find an optimal policy, value iteration works on it by finding the optimal value function first, and then the corresponding optimal policy.

The basic idea is to set up a sequence of state-value functions $\{v_k(s)\}_k$ such that this sequence converges to an optimal state-value function $v^*(s)$. In fact, a state-value function v can be written as a vector in $\mathbb{R}^N$: each component corresponds to $v(s)$ where s is a particular

state. Recall that the Bellman operator T^* is defined by:

$$T^*v(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P\left(s' \mid s,a\right) v\left(s'\right) \right] \tag{1.97}$$

The Bellman operator is in fact a max-norm contraction with contraction factor γ [3]. It implies that $\left\| T^*v - T^*\mu \right\|_\infty \leq v - \mu_\infty \forall v$ and μ state-value functions $\in \mathbb{R}^N$

Theorem 1.2 (Banach fixed point theorem (contraction mapping principle)). *Let T be a mapping $T : X \to X$ from a non-empty complete metric space X to itself. If T is a contraction, it admits one and only one fixed point.*

Furthermore, this fixed point can be found as follows: starting with an arbitrary element $x_0 \in X$, an iteration is defined by : $x_1 = T(x_0)$ and so on $x_{k+1} = T(x_k)$. This sequence converges and its limit is the fixed point of T.

In our case, $\mathbb{R}^N$ is a non-empty complete metric space and the Bellman operator T^* is a contraction. So, from theorem, the sequence v_{kk} converges to the fixed point v^* of Bellman equation. We have already proven previously that the fixed point of the Bellman operator is in fact the optimal policy for our MDP. That is why the value iteration converges to the optimal policy. The whole algorithm is described in Algorithm 1.

Algorithm 1: Value Iteration Algorithm

Input: Set of states S, set of action A, vector of reward for every action $R(s,a)$, transition probability matrices for every action $P(s'|s,a)$, the discount rate γ, a very small number θ near to 0

Initialise v_0 arbitrarily for every state $s \in S$.(Example : $v_0(s) = 0, \forall s \in S$);

while $|v_k(s) - v_{k-1}(s)| < \theta, \quad \forall s \in S$ **do**

  forall $s \in S$ **do**

    forall $a \in A$ **do**

      $Q_{k+1}(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) v_k(s')$;

    $v_{k+1}(s) = \max_a Q_{k+1}(s,a)$;

    $\pi_k(s) = \operatorname{argmax}_a Q_{k+1}(s,a)$ {Gives the action a that has given v_{k+1} }

Output: v_k { Takes the v_k at the last iteration of the While Loop: Which is the optimal statue-value function

Output: π_k { Takes the π_k at the last iteration of the While Loop: Which is the optimal policy

1.5 Conclusion

In this chapter, the two approaches of Markov chains and dynamic programming are introduced in detail. To model the best management decision, the use of dynamic programming provides the correct strategy to be chosen by the agent to control the system. In the following chapters, the contributions in terms of modeling management resources issues in cloud computing will be developed, starting with static and then dynamic defense measures.

Chapter 2

Survey on Management task in cloud computing environment

Contents

In this chapter, the cloud computing environment is introduced by citing its architecture, its characteristics, its service and deployment models as well as its associated technologies for its implementation. In addition, the treatment related to this environments are treated in term of managing its resources and its uses. Based on this, there are several dynamic techniques for management of resources in cloud computing as the two most known technologies are described: Virtualization and migration. Some research work are presented at the end of this chapter, which aim to manage the resource and tasks in order to minimizing the energy consumption and ensuring the QoS.

2.1 Introduction

Cloud computing is one of the technologies modeled to provide a service [4] rather than a product. Services such as computing, software, data access and storage are provided to its user without any prior knowledge of the physical location and configuration of the server providing these services. Large virtualized data centers are established to meet this requirement. In general, we mention "Cloud Computing" when it is possible to access data or programs from the internet, or at least when this data is synchronized with other information on the Internet.

Cloud platform can deploy new digital customer experiences in days rather than months and can support analytic that would be uneconomical or simply impossible with traditional technology platforms. and the managers need to accelerate their business to the cloud in order to digitize quickly and effectively in the wake of Covid-19.

However, cloud computing is a complex system composed of a set of IT resources to respond the needs of customers. The realization of this service/need requires effective algorithms to use its resource as well as the deployment of specific security. These services are divided into three types as described in figure 2.1.

2.1.1 Types of cloud computing services

There are three main forms of as-a-service cloud computing :

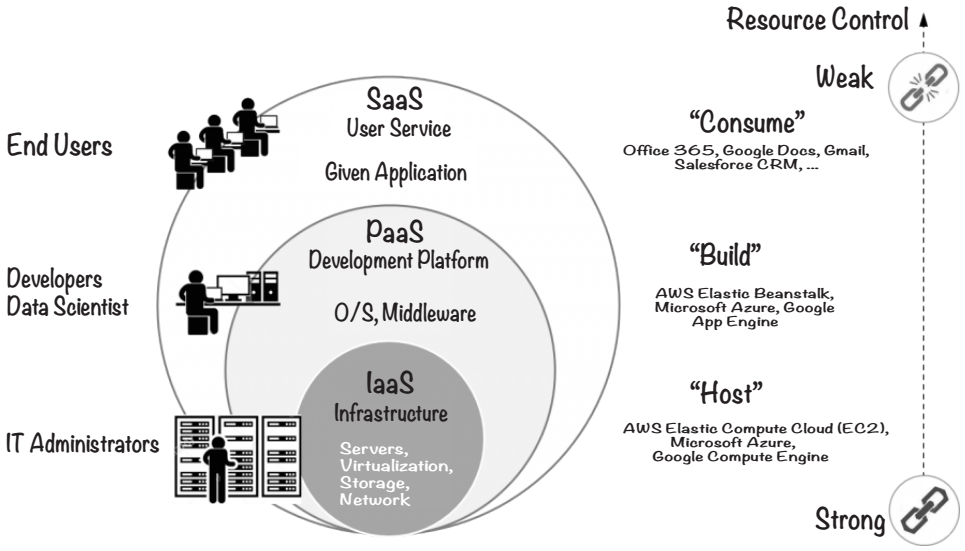

Figure 2.1 – Cloud Service Models

- **Infrastructure as-a-service (IaaS) :** As the most basic form of cloud computing, IaaS allows users to access the basic elements of the infrastructure, such as server space, data storage and network, which can be provisioned through an Application Programming Interface (API). This model comes closest to replicating the functionality of a traditional data center in a hosted environment.

- **Plateform as-a-service (PaaS) :** This model provides a complete development environment, eliminating the need for developers to deal directly with the infrastructure layer when deploying or updating applications.

- **Software as a Service (SaaS) :** SaaS applications are designed for end-users, with provisioning and infrastructure development taking place entirely in the background. From popular applications such as word processing and spreadsheets, photo editing suites, and video hosting platforms, SaaS applications offer a wide range of functionality in the cloud.

2.1.2 Deployment models

Three basic models of cloud computing are available as shown in figure 2.2:

- **Public cloud:** The public cloud is a shared cloud infrastructure that is owned, managed and maintained by a cloud provider, such as Amazon Web Services or Microsoft Azure. The main advantages of the public cloud are on-demand scalability and pay-per-use pricing.

- **Private cloud:** This type of cloud runs behind a firewall on a the company's IT network and is hosted in a data center dedicated to the organization. The private cloud infrastructure can be configured and managed according to an organization's individual needs.

- **Hybrid cloud:** As its name says, the hybrid cloud model allows enterprises to take advantage of both private and public cloud solutions. With the hybrid cloud, organizations can leverage the strengths of each cloud model - the flexibility and scalability associated with protecting operations and sensitive data.

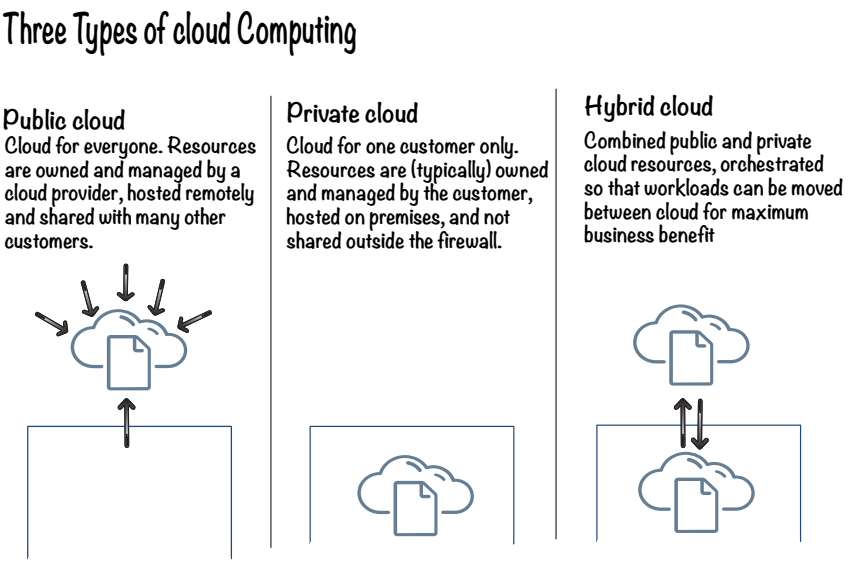

Figure 2.2 – Types of Clouds

2.1.3 The characteristics of cloud computing

According to the National Institute for Standards and Technology (NIST), the cloud offers several services with essential characteristics such as the elasticity of the resources offered by cloud services, their ease of access via the network, the pooling of resources, the increased agility of cloud information systems as well as billing for the use of cloud services:

- **Elasticity of resources:** In the cloud, new capacities can be automatically made available to users in the event of an increase in demand. Conversely, they can be quickly stop the use when they are no longer needed. This elasticity of cloud services creates the illusion of infinite capacity for the end user, which can be put into service at any time. All these operations can be performed automatically by scripts. These management mechanisms enable you to take full advantage of pay-per-use billing by adapting computing power to instantaneous traffic.

- **Invoicing per use:** With the cloud, a new way of invoicing usage is being implemented, which can be summarized simply: you only pay for what you actually use. So in the infrastructure cloud, you pay according to the number of processor cores consumed, the amount of memory used, the number of input/output operations performed or the amount of data stored. And in a SaaS mode, you pay according to the number of users and their use of resources. There is generally no cost for commissioning (the user carries out the operations). Invoicing is calculated according to the duration and quantity of resources used. A stopped treatment unit is not charged.

- **Access to services by the user on demand:** The implementation of the systems is fully automated, and the user accesses to the services on demand through a control console, which sets up and menages the configuration remotely.

- **Broadband network access:** These processing centers are usually connected directly to the internet for excellent connectivity. Major providers are allocating processing centers around the globe to provide access to systems in less than 50 ms from any location.

- **Resource pool (not localized):** Most of these centers have tens of thousands of servers and storage to allow for rapid scalability. It is often possible to choose a geographical area to put the data "close" to the users.

2.1.4 Advantages and disadvantages of cloud computing

This technology offers several advantages and benefits, including the following:

- **Low initial investment:** With cloud computing, a large part of the IT budget becomes an operating expense instead of a capital expenditure. Enterprises no longer need to build expensive data centers before opening their doors or undertaking new initiatives.

- **Cost-effectiveness :** Whether you're a small or large enterprise, you can reap the same benefits from the economies of scale of cloud service providers. Communication service providers are able to get the most out of the amount of hardware they use, save energy and other costs, and then pass the savings on to their customers.

- **Easier innovation:** Within the IT team as well as within the enterprise, cloud computing often facilitates the path to innovation. Freed from the operational burdens of racking and stacking, IT departments have the bandwidth they need to improve business processes, generating improvement with far-reaching effects. Meanwhile, their management-side counterparts are able to quickly and inexpensively find experimental resource programs and then build or scale them down without having to do detailed infrastructure planning or an initial long-term investment.

- **Better continuity of activities:** Due to the virtualized nature of the Cloud computing infrastructure, the creation of data and operating system backups and the switching of the switch over procedures can be automated. This provides much better data protection and availability than most on-premise systems can provide.
  - The security of data guaranteed by the provider cloud computing gives additional security guarantees. Indeed, the storage and backup of data is not done locally but on an external server and the services run on it.
  - The client does not have to worry about hardware issues anymore.
  - Moreover, the confidentiality and protection of data are sensitive points.

- **Ideal support for working in collaborative mode:** Clients are always more mobile and their objectives are achieved collectively. Thanks to the cloud, the information, services and applications they need to perform their tasks are easily accessible from anywhere and on any type of device. If the clients are a team, so they are connected to each other via a platform that centralizes all the channels they use. They communicate better and share their files and comments from anywhere. Document updates are made in real time. (Planning, storage, management, execution, project monitoring, cloud computing is the ideal choice for efficient collaborative work.)

- **Continuous innovation:** To keep your business at the forefront of innovation and performance, cloud service providers regularly update their offerings. They ensure you get the latest technologies across all your tools and applications. For example, the client can :

  - Always get the latest version of an application.
  - There is no risk of obsolescence
  - Tool upgrades are automatic and transparent

However, it is important to choose a service provider offering serious guarantees to ensure the confidentiality and integrity of data.

Despite all these advantages, the cloud computing system reveals some problems:

- Its computer networks are potentially attackable and virtual services can be shut down.

- The client depends on a service provider (Google Cloud Platform, AWS, Azure, etc.). It is fundamental to ensure that it offers the necessary guarantees and knows the expectations of their business.

- All services depend on the internet connection. An outage can disrupt the business of such organizations .

- The client lose part of the control of his computer system.
- As the Arabic proverb says: "One rotten fish ruins everything in the basket": that means a wrong client can destroy a large system.

2.2 Resources management

Task management is a discipline that is well known and can used in many different area, It is applied when there is a set of tasks to be managed, and these tasks are to be performed under constraints. These constraints can be of different kinds, they can be temporal constraints, when tasks must be performed before a defined deadline. Tasks can be also under resources constraints when the they must be completed using a limited resources.

In our case, all tasks are not identical, some are finite and others infinite, and they are not characterized by the same parameters. However, there is one characteristic common to all tasks: they all need a certain resources to be running, either explicitly when the task requires a certain number of machines or access to define resources, or implicitly when the task just requires the possibility to run on a specific resources.

Looking at the world of cloud, there are tasks that are weakly constrained by time, but strongly constrained by the resource they use such as the CPU and power that tasks will use.

In this thesis, we will focus on tasks as VM running in a cloud environment, these tasks will be interested in time constraints and by the energy consumption of the infrastructure, as well as the consumption of requested machine resources (CPU, memory, disk, etc.).

The management of these tasks will be based on different types of inputs, and need to solve several different problems. In addition, it may have different aims, different objectives, and therefore different forms. In the following sections, we will attempt to describe in detail the tools, virtualization and migration.

2.3 Virtualization

As systems become more and more complex, and architectures more and more diverse, it becomes obvious that some services need to be made available in a transparent way. It thus becomes necessary to facilitate development to have access to different resources in common. This is where virtualization comes in, representing the idea of making access identical to several different architectures. virtualization technologies represent a key factor in cloud computing. The most important characteristic is the possibility to install on the same physical machine (server) more than one operating systems on different virtual machines. In addition, this technology has the extra benefit of reducing overall cost by using a minimal equipment and consequently reducing the energy consumption.
The virtual machine (VM) concept was introduced in the 1960s by IBM as a way to provide simultaneous, interactive access to their mainframe computers. A VM is an projection of the physical machine and it gives the impression that users have direct access to a physical

machine (PM). VMs are used to enable resource sharing of very expensive hardware. Each VM is a fully protected and isolated copy of the system. Virtualization is an important tool for reducing material costs and improving global productivity by allowing multiple users to work simultaneously on the same PM. This allows virtualization to increase utilization of the machine. The main objective of virtualization is to share the physical characteristics of

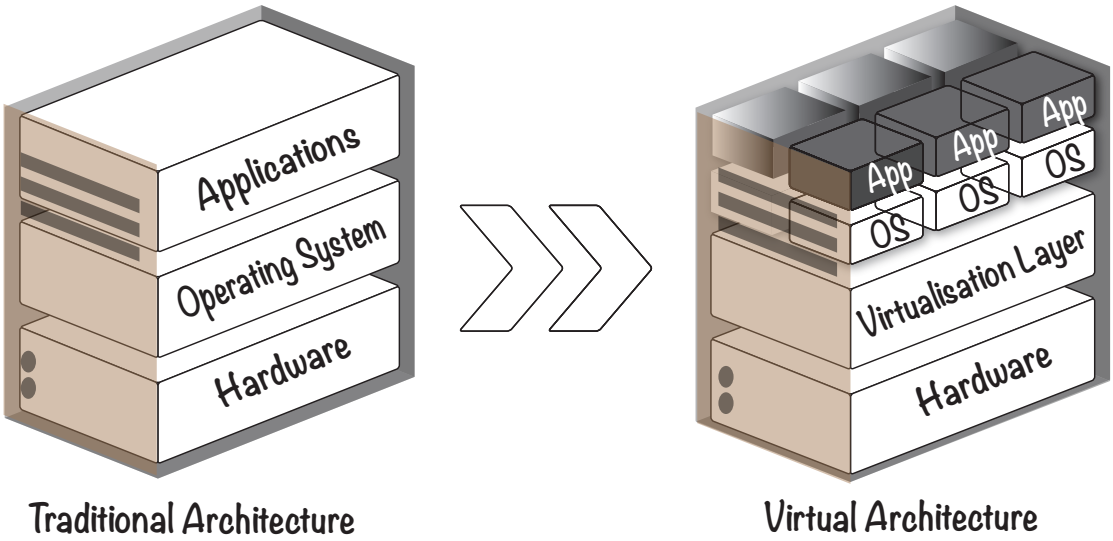

Figure 2.3 – Virtualisation

computing resources, so other systems, applications or end users interact with these resources. A main operating system (called the "host system") is installed on a single physical server. This system is used to host other operating systems. A virtualization software called "hypervisor" or VMM (Virtual Machine Monitor) is installed between the two hardware layers and the operating system (See figure 2.3) on the main operating system. It allows the creation of separate, independent operating system environments ("guest systems"). These environments are "virtual machines". A guest system is installed in a VM that runs independently of other guest systems in other virtual machines. Each VM has access to the physical server resources (CPU, memory, disk space, ...). The VMM takes control of resource management and is also used for system power management. This energy management is done either by applying Dynamic Voltage Frequency Scaling (DVFS) techniques or by applying energy management policies and approaches at the application level.

According to [5], DVFS is adjustment of power and speed. It successfully used in energy management of real-time system domain. DVFS is technique that aim at decreasing the dynamic power utilization by progressively altering voltage and recurrence of a CPU. The procedure of purposefully restricting the execution of the processor when it expending a lot of intensity while the errand can be practiced handling at lower frequency. Voltage and recurrence have an immediate relationship. It has been seen that server expend about 70% power when it is inert sparing this vitality will have incredible effect on energy reduction. DVFS can deal with this issue adequately DVFS is fundamentally expected and structure for vitality effectiveness in embedded system. DVFS methods generally consider homogeneous physical machine while less intrigue is appeared by analyst toward heterogeneity.

According to this flowchart of DVFS techniques 2.4 it can be creating a host and then crated virtual machine on host, balance the load of the user request and allocated host to

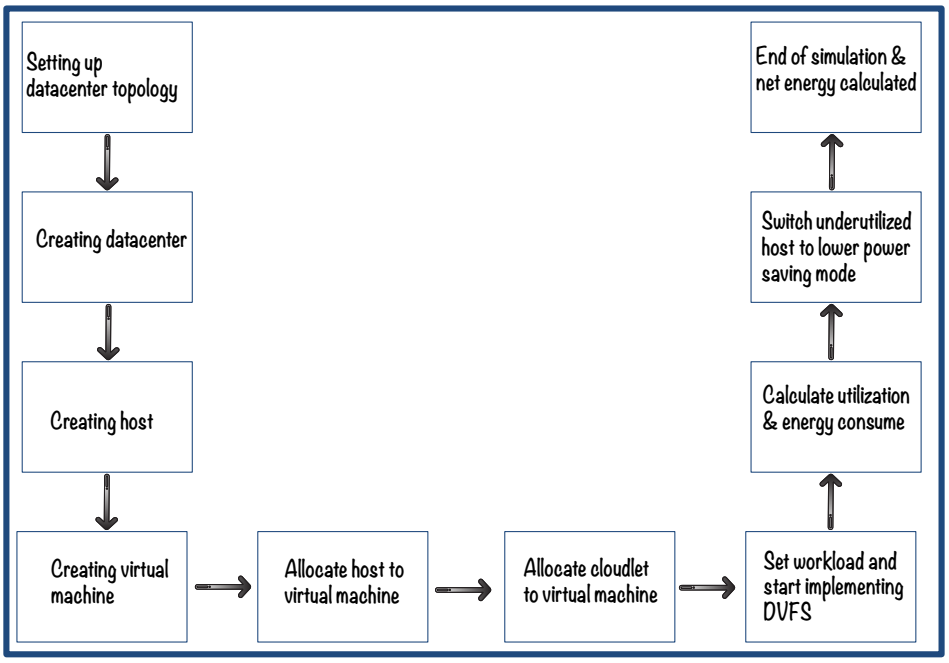

Figure 2.4 – Flowchart of DVFS

virtual machine. After completed this process calculating the energy then get proper result according that result it automatically shut down host and consume the energy.

2.3.1 Virtualization techniques

Before using virtualization in a cloud computing environment, the user must be aware of basic virtualization techniques such as emulation, hypervisor, full virtualization and paravirtualization [6].

A. Emulation

This is the oldest virtualization approach, usually called binary virtualization. In this technique, the hypervisor presents a complete virtual hardware (CPU, network interface, video card, ...) to the guest system which can be a completely different CPU architecture than the host system. The hypervisor then converts the CPU instructions from the virtual machine into instructions understandable by its CPU. Guest operating systems believe they are connecting directly to the hardware (See figure 2.5).
The emulation allows to properly isolate the guest operating systems. It provides great flexibility to the guest operating system, but the speed of the process is low compared to the hypervisor and requires a high configuration of hardware resources to run the software[7].

B. Full virtualization

In this virtualization technique, the Hypervisor creates an isolated environment between the guest or virtual server and the host or server hardware (See figure 2.6). Operating systems access directly to hardware and controller devices without being aware of the virtualized

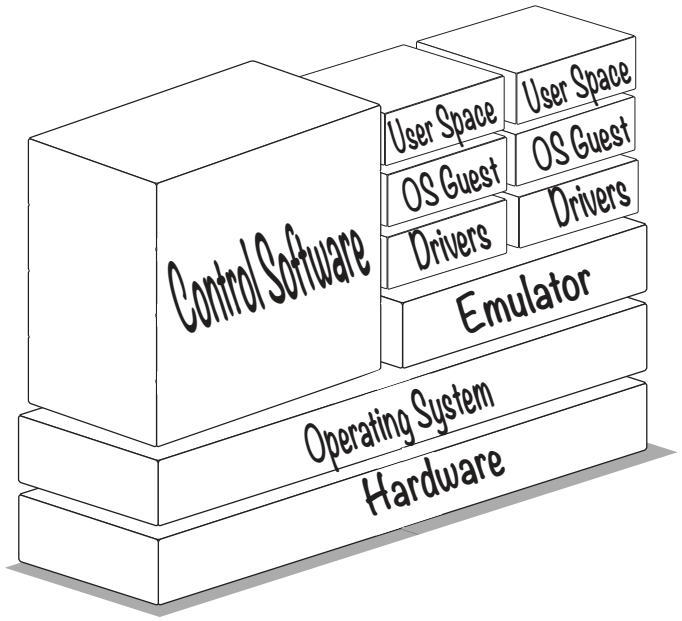

Figure 2.5 – Emulation

environment and the changes required. This virtualization intercepts and emulates privileged and sensitive instructions at runtime.
The main problem with full virtualization is the poor performance of the binary conversion and it is difficult to accelerate it [8].

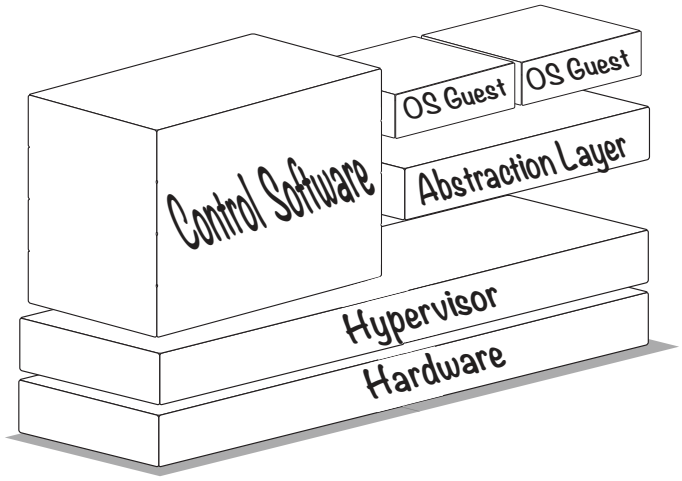

Figure 2.6 – Full Virtualization

C. Paravirtualization

Paravirtualization is the type of virtualization invented by the Xen project. This type of virtualization was made possible by free software, because it was possible to modify and adapt the Linux or BSD or Solaris kernel. Microsoft has put this method into practice with its Hyper-V product integrated into Windows Server 2008.

When guests are para-virtualized, they are supported by an intelligent compiler, which replaces non-virtualized OS instructions with hypercalls (See figure 2.7). It concerns the communication between the hypervisor and the guest operating system to improve efficiency and performance. Para virtualization resource access is better than the full virtualization model where all resources must be emulated in a full virtualization model.
Paravirtualization must modify the guest OS. It tries to reduce the processor resource consumption of virtualization and thus improve performance by changing only the guest OS core.

The disadvantage of this technique is that it modifies the Core of the guest exploitation system using hypercalls. This model is only suitable in open source operating systems. Many virtualization products use paravirtualization architecture. Popular products include: Xen, KMV and VMware ESX .

Unlike the full virtualization architecture, paravirtualization is not the same to manages instructions at compile time. Although paravirtualization reduces the consumption of processor resources, it leads to other problems. Firstly, its compatibility and portability may be questioned, as it must also support the unmodified OS. Second, the cost of managing paravirtualized OS is high, as they may need to retain OS core modifications. Finally, the performance benefits of paravirtualization can vary considerably depending on the load. Compared to full virtualization, paravirtualization is relatively easy and more convenient. As a result, many virtualization products employ the paravirtualization architecture [8].

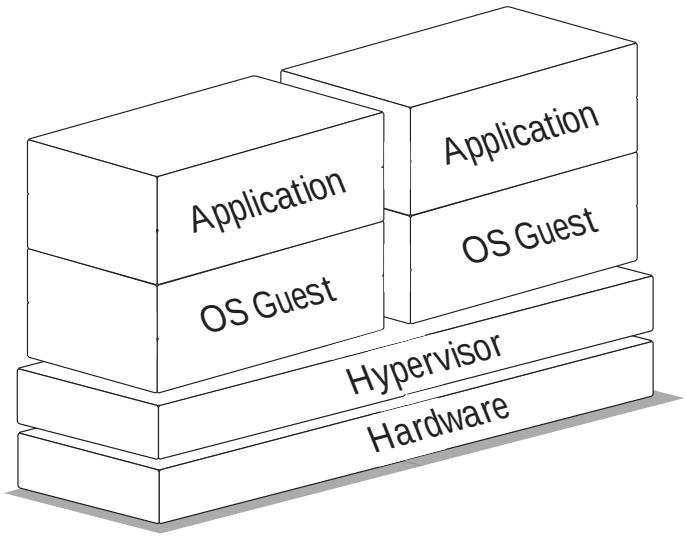

Figure 2.7 – Paravirtualization

D. Hypervisor

The hypervisor or VMM is a low-level program that allows multiple operating systems to run simultaneously on a single host machine. VMM creates isolated virtual machines, each one with its own operating system. The hypervisor manages requests for virtual machines to access the hardware. It creates an environment for VMs that is similar to the real environment. As an example of hypervisors, we have the Xen hypervisor which is an open source virtualization

technology, VMware which supports power management at the physical machine level using the DVFS technique, the Kernel-based virtual machine (KMV) which is a platform implemented as a module of the Linux kernel. Hypervisors can be classified into two types:

1. **Hypervisor type 1 (Bare Metal) :** In this type, the virtualization layer is installed directly on top of the hardware and controls the OS running above the hypervisor (See figure 2.8). This configuration provides response times similar to the non-virtualized case, as there is only one additional layer to go through. It is the only type of virtualization that can be used in production because it is robust and scalable. It currently offers better performance and security [9]. Its main task is the sharing and management of hardware resources between different operating systems. A main advantage is that not all problems of a virtual machine or guest operating system affect the other guest operating systems running on the hypervisor[10].

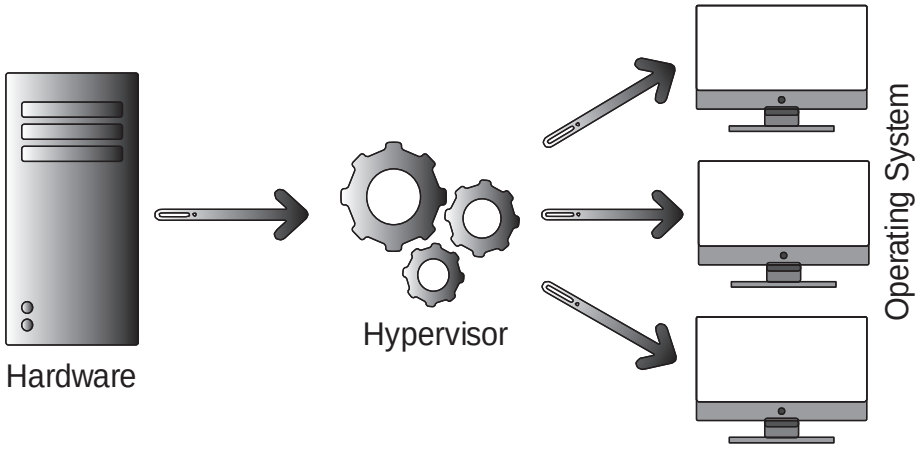

Figure 2.8 – Hypervisor type1 Bare Metal

2. **Hypervisor type 2 (Host Based) :** The use of this technique is limited to development (software for example). It is installed and run as an application. Indeed, the virtualization layer is installed on an operating system and not directly on the hardware layer and it supports other operating systems on it, while having a basic operating system that allows better policy specification (See figure 2.9). Problems in the base operating system affect the entire system even if the hypervisor running on top of the base operating system is secure [11],[10].

2.3.2 Advantages and disadvantages of virtualization

Virtualization offers several advantages, among them we cite [12]:

- Ability to install multiple operating systems on a single server (Windows, Linux, . . .), which reduces the number of hardware and the maintenance of multiple devices.
- Easily test, deploy and migrate virtual machines from one physical machine to another.
- **Cost reduction of electricity and air conditioning:** Many servers require electricity and produce a lot of heat. If the number of physical servers is reduced, the cost of electricity

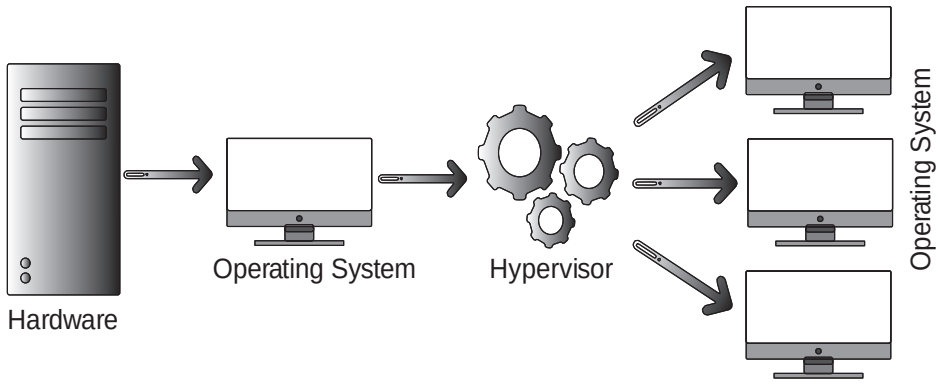

Figure 2.9 – Hypervisor type2 Host Based

is automatically reduced, and if heat production is reduced, air conditioning does not cost as much.

- **High speed to deploy a new server:** The installation of a physical server requires hardware and an OS license which is very expensive in time and money. Using this technology, we can create templates that are basic OS installations that are used to create servers. These models can be installed as desired. Deploying a new server is now counted in minutes rather than days or weeks.

- Microsoft, for example, offers a good advantage in terms of Windows Server licenses for virtual environments. If a data center license is purchased, it allows to install an unlimited number of windows OS in virtual mode. This is very practical when license management is not easy to get. We only need fewer licenses to be as efficient. Microsoft enables proactive management of license requirements.

- **Portability :** A VM can be transferred from one server to another using Migration techniques.

- Virtualization allows to reduce the time and cost of server management, which is usually high. The management of virtual machines is more manageable, thus helping to reduce the workload for system operators.

Despite all these advantages, there are a few inconveniences around virtualization:

- **Important cost:** To make a virtualized architecture work properly, the company must invest in a physical server with multiple processors and memory.

- **Generalized breakdowns:** if the physical server fails, the virtual machines also fail.

- **Generalized vulnerability:** if the hypervisor is corrupted or vulnerable to a security issue, the virtual machines may be corrupted as well and are no longer protected. By increasing the software layers, virtualization increases the company's attack surface.

2.4 Virtual machine migration

Virtualization allows the creation of multiple copies on the host machine so that several applications can run independently. It separates the operating system from the physical hardware using a hypervisor. Each VM has its own operating system called guest OS. Migration of operating system instances (VMs) from one physical machine to another is an important feature of virtualization. VM migration also includes the transfer of memory, CPU from a machine to another.

2.4.1 VM migration techniques

Migration techniques are basically classified into two types [13]: Live Migration and Non Live Migration.

A. Non live migration "Stop and copy migration"

Non live migration "stop and copy method" is a technique where the VM stops running on the original machine and its execution with its memory is migrated to the destination machine. Once all memory pages have been transferred, the VM is activated in the destination machine. The VM stays in a disabled state until all pages are transferred to the destination machine. In this type of migration, the total migration time is equal to the downtime.
Figure 2.10 illustrates the stop and copy migration steps. We take as an example the steps of migration of the black VM from source machine to destination machine :

1. The execution of the black VM is stopped in the source machine.
2. The black VM execution status and memory pages are transferred to Machine
3. The black VM is enabled in the destination machine.

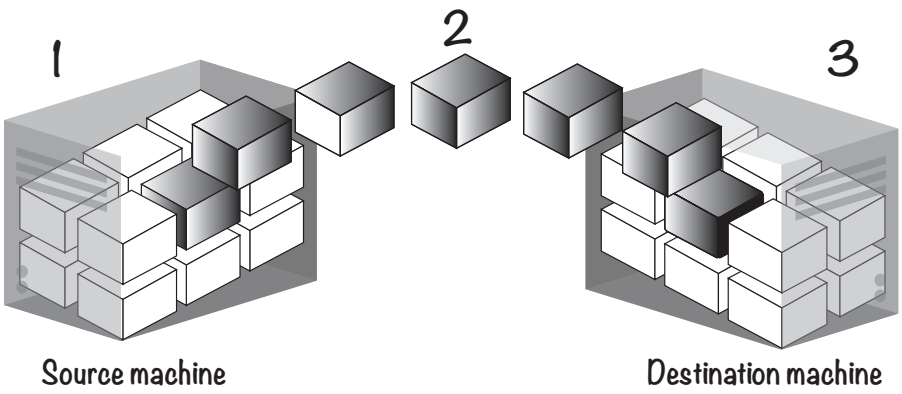

Figure 2.10 – Non Live Migration process

B. Live migration

Live migration is a valuable tool for the migration of complete operating systems and their applications between physical machines located far from data centers [14]. It is an impressive technology that facilitates load balancing, breakdown management, system maintenance and energy reduction. live migration of virtual machines is the process of moving one or more virtual machines from one physical machine or server to another without interrupting other VMs.
This technique is useful in several scenarios, including [15] :

- A VM in a crashed physical node can be migrated to another non-crashed node.
- VMs in idle mode on a physical machine can be migrated to other hosts to optimize resource utilization.
- VMs on a loaded physical node can be migrated to different nodes to balance loads.

Live migration techniques are characterized by 2 types based on the memory copy process. These techniques are: "Post-copy" and "Pre-copy".

a. **Live migration "Post-copy"** : In this technique, the VM is suspended in the source machine and its execution status (CPU, registers and memory pages needed to activate the VM in the destination machine) is transferred to the destination machine. The VM starts running on the destination machine even if the memory pages have not been copied completely. The page defects can occur when the memory pages of the VM are not available on the destination host. The disadvantage of this method is the appearance of memory page failures in the destination machine. Figure 2.11 shows how Post-copy migration works. We take the black VM migration as an example:

   1. The black VM is suspended in the source machine.
   2. The black VM execution status is transferred to the destination machine.
   3. The black VM is enabled in the destination machine.
   4. Generate memory page defects.
   5. Transfer the corresponding default pages.
   6. Steps 4 and 5 are repeated until all memory pages are transferred to the destination machine and the black VM runs.

b. **Live migration "Pre-copy"** : This is a technique of live migration used by hypervisors such as XEN, KMV and VMware. It sends the entire VM to the destination machine during the first iteration and then iteratively transfers the modified memory pages. This process continues until all sufficient memory pages are copied to the destination host. Figure 2.12 summarizes the principle of the black VM pre-copy migration in 5 steps as follows :

   1. Send all black VM memory pages from the source machine to the destination machine.
   2. Transfer modified memory pages in an iterative manner.
   3. Stop the black VM in the source machine.

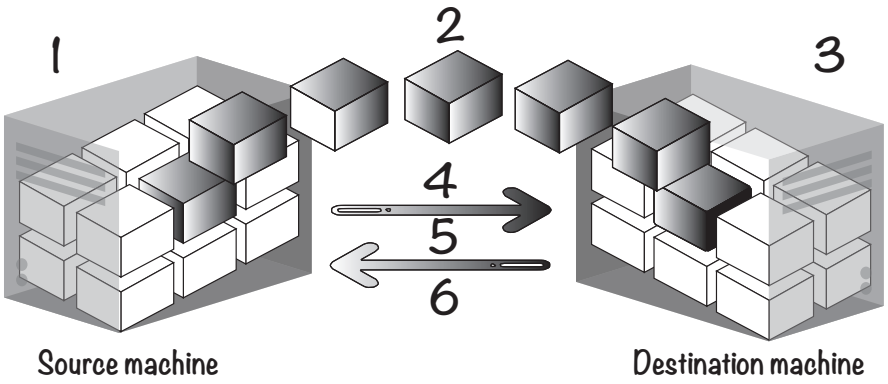

Figure 2.11 – Post-Copy Migration Process

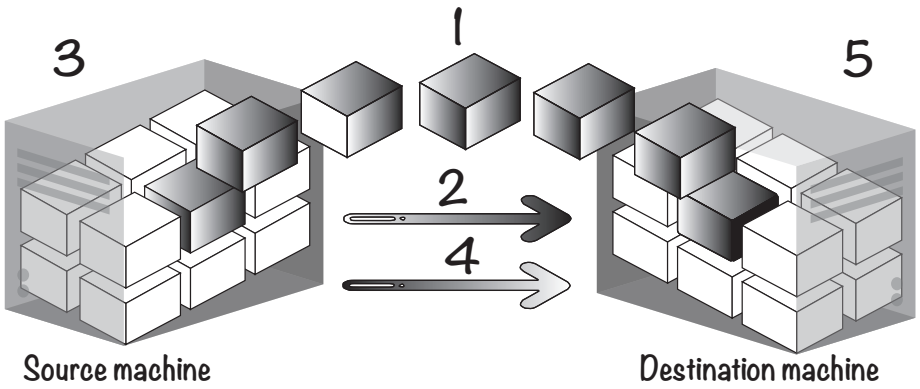

Figure 2.12 – Pre-Copy Migration Process

4. Send the remaining states of the black VM.
5. Activate black VM in the destination machine.

In case of a failure of the destination machine, the Pre-copy technique is much better than the Post-copy. The recovery of the black VM is possible by the Pre-Copy method because the last memory copy is kept in a safe place on the source machine including the CPU state.
The following measures are generally used to measure the performance of live migration [15] :

- **Preparation :** In this phase, the appropriate resources are reserved on the destination machine and various operations are performed on the source machine.
- **Holding time:** is the time when the virtual machine is suspended on the source host.
- **Time summary:** This step involves installing the VM on the destination host with the same state as the suspended VM. The time required to complete this phase varies according to different approaches.
- **Total migration time :** The total time taken after completion of all these phases is called total migration time.

2.4.2 Advantages of migration

The benefits of virtual machine migration include [14]:

- **Trouble-sensitive:** Trouble-sensitive allows virtual machines to continue their tasks even if any part of the system fails. This technique migrates the virtual machine from one physical server to another based on the prediction of the failure. The trouble-sensitive migration technique improves the availability of the physical server and prevents application performance degradation.

- **Load balancing:** The load-balancing migration technique aims to balance the workload on physical servers to improve the scalability of physical servers in a cloud environment.

- **Reduction of energy consumption:** The energy consumption of Data Centers is mainly based on the use of servers and their cooling systems. These servers typically use up to 70% of their maximum power consumption, even with low resource utilization. Therefore, migration techniques must be used that conserve the energy consumed by the servers through optimal use of resources.

Virtualization is a technology that enables cloud providers to solve the problem of energy inefficiency by creating multiple instances of virtual machines (VMs) on a single server. As a result, aggressive consolidation of virtual machines can lead to performance degradation when an application is subject to increasing demand, resulting in increased resource utilization. Ensuring the reliable quality of service (QoS) defined by Service Level Agreements (SLA) is essential for cloud computing environments, so cloud providers have to find a compromise between the energy performance of the data centers and SLAs.

2.5 Task allocation

Task allocation is also a way to manage the energy consumption of a system. So by allocating tasks wisely, we can use the infrastructures that consume the least amount of electricity.

This is the case of the consolidation techniques. The idea is to consolidate the task at the same place in order to reduce energy consumption. Consolidating tasks on a few machines can have adverse effects. it is also necessary to take into account the interactions that tasks can have with each other when they run on the same machine. In [16]the authors show that by profiling the resource consumption of a task running among others, the resource and energy consumption of each transaction of the task increases compared to the case when the task is running alone on the machine. The authors then show that it is possible to find an optimal zone where each task running on the machine will consume the least energy. Then they propose a consolidation algorithm based on the task interaction profile, to determine the optima consolidation, and to characterize which types of workloads should be combined.

Allocation can also be used as a tool for achieving the desired results, as a characterization of the different material resources can make a significant difference. Not all servers consume the same amount of power to run the same application, so using the task allocation is important.

Different techniques can be observed to take into account the different consumptions at different levels, whether at the processor level or at the machine itself.

2.5.1 Complexity of task allocation

In the context of all the tools available to us, it appears that the allocation of tasks has developed into something much more complex. Allocating tasks is no longer just deciding where to allocate a task on a set of machines, it is also the management of these machines, taking into account all the resources available to us after the allocation. We now also have to take into account the different available resources on the different hosts, the power consumption of these hosts, predict the needs over time, in order to have a sufficient supply of machines on.

To that, it must add the fact that there is a difference of scale in the different tools, which implies a potentially different management of the different mechanisms. For example, DVFS is done at the processor level, while switching on and off is done at the host level. Cooling must be managed at the machine level, while migration is done at the host level. So we have a multi-level approach, which has to take into account the whole set of possible approaches to reduce power consumption, from the Machine level.

It is easy to think of the case where the consolidation of virtual machines on a physical machine will have an unwanted effect on the provisioning of virtual machines (for example, in the case of too aggressive consolidation of virtual machines). Thus, if many virtual machines are allocated on a single physical machine, it will be difficult to increase the share of resources allocated to virtual machines, since few will remain idle, without counting on the interference between virtual machines when they are the same physical machine [17].

2.5.2 Tasks allocation under certain constraints

Task allocation can be applied in a variety of ways. Indeed, the same techniques can be used to optimize a system for one or more different objectives as well as to the type of management algorithms that will be used. Everything will depend on the constraints that we take into account for the management of our system.

The main idea is to have a management of resources that will be done according to certain parameters. This can include a monetary cost to maximize profits, or the use of a certain type of resource such as the use of renewable energy. We can also very well have internal considerations for applications running on the physical infrastructure, for example, tasks are strongly constrained in memory, or constrained in time by deadlines.

A. Constrained resource allocation

The management of tasks with resource constraints, which we will call resource-aware, represents the idea that we have a set of resources and these resources are not accessible from everywhere, at least not in the same way.

Task management will then give each task a place to be performed, for example to consume as much renewable energy as possible. The idea here is to optimize the overall electricity

consumption, in a particular direction, which is that of renewable energy. This is the case in [18], where will allocate HPC tasks according to different metrics such as CO2 emissions, cost of electricity, or processor efficiency.

However, this is just one example of what can happen. We could have also a set of geographically distributed servers, and want to carry out task management that takes this location into account, in order to bring a task as close as possible to its user, to improve the quality of service [19]. This will often be the case in content delivery networks, due to the volume of data to be transferred, the distance between the client and the server needs to be reduced as much as possible.

In general, the constrained allocation of tasks in resources allows the resulting allocation without a precise direction, according to the needs. Thus, the allocation will have to solve the various problems related to the different possible constraints.

B. Constrained energy allocation

The constrained energy allocation is related to the management of resource-aware tasks, since it can be defined as a constrained resource allocation, with the energy consumed or to be consumed as a resource. Task management can try to optimize the power consumption of the infrastructure on several levels. We refer to this allocation of tasks as power-awar or energy-awar (remember that power-awar is based on the instantaneous electrical power, and therefore optimizes at a given moment, while energy-awar is based on the energy consumed over a period of time, therefore is the integral of the instantaneous power).

We also have approaches at the machine level. In this case, it is necessary to manage the placement of tasks on the different machines, according to their consumption. It will also be necessary to manage the switching on and off of hosts, in order to reduce the overall consumption of the machines.

In [20] an optimization strategy is developed based on a formulation in a mixed integer linear program (MILP, Mixed Integer Linear Programming), in order to dynamically consolidate services in a virtual machine. This approach takes into account the cost of switching machines ON/OFF and is used in the context of an oversized system to be able to meet peak load requirements, leading to a 50% reduction in the energy consumed in the cluster.

A similar approach is developed by Viswanathan et al. in [21], where virtual machine allocation algorithms are based on an empirical model. A compromise is made between power and makespan (time between the beginning and the end of a set of tasks), and thus approaches manages to save 12% of power while reducing the makespan by 18% on average compared to a standard first fit approach (Algorithm where each task is allocated to the first host).

C. Constrained cost allocation

The cost constraint can mean two things. It means allocating tasks in a way that reduces the cost of maintenance for the infrastructure owner. It can also mean managing the way that reduces the cost of performing the tasks in order to improve the quality of service. In order to

reduce these costs, it is first necessary to understand them, as they can be complicated to take as a whole.

In [22], the authors attempt to assess the impact of data centers in terms of energy and cost by evaluating the energy consumed and the equipment used. They show that larger facilities often have a smaller carbon footprint, and that containerized data centers (servers, power and cooling are in a standard size container) have lower operational energy consumption.

In [23] on the contrary, the approach observed the maximization of the profit for service providers, using energy savings through the allocation of customer requests. This approach requires the implementation of policies for dynamic workloads, taking into account errors due to workload prediction. The first policy is to maximize revenue by calculating the optimal number of servers to turn on, taking into account the cost of changing state, as well as the revenue generated by each task. A policy based on a heuristic called QED (Quality and Efficiency Driven) allows to calculate a sub-optimal value of the number of servers to be turned on to answer the demand.

The cost of migration can be taken into account in the decision, so as not to make reallocations of virtual machines against performance. This is the case in GreenCloud [24], a management architecture for virtual machines with limited performance. It defines a heuristic of efficient replacement taking into account the cost of migration, the cost of energy consumption (here modeled by the number of machines turned on), as well as the cost of quality of service (modeled by the number of servers with a utilization of more than 90%).

In the following sections, a state of the art is presented by briefly describing the approaches proposed in the literature whose aim is to reduce the energy consumption of data centers or to improve QoS or to find a trade off between theme.

2.6 State of the art in energy management techniques

A. **Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing :** In order to manage efficiently the energy consumed in the cloud, the authors [25] proposed a model of the system architecture and three energy management policies. In cloud computing, power consumption is determined by the CPU, RAM, storage memory and network interfaces. In this work, the authors focused on the management of the energy consumption of processors. The technique DVFS was used. When virtual machines are not using all the resources provided, they can be logically grouped and transferred to the minimum number of physical nodes and inactive machines are powered down to eliminate unused power and reduce the total power consumption used by the data centers.

   In this article, the authors have proposed two phases in order to optimize the allocation of virtual machines. The first is the selection of VMs to migrate. In this phase, three policies were proposed: Minimisation of Migration Policy (MM), Highest Potential Growth policy (HPG) and Random Choice policy (RC).

   The basic idea of the three policies is to consider the upper and lower thresholds of

utilization as related to host utilization. If the host CPU usage is below the lower threshold, all virtual machines must be migrated and the machine is put into sleep mode to eliminate power consumption. If utilization exceeds the upper threshold, some virtual machines must be transferred from the host to reduce CPU usage. The second step is the implementation of the virtual machine selected by the proposed policies. The MBFD (Modified Best Fit Decreasing) algorithm was used in this second allocation phase.

To compare the efficiency of the algorithms, the authors used several performance metrics: the first metric is the total energy of the resources, the second performance metric is the percentage of Service Level Agreement (SLA) violation. SLA violations occur when a given VM cannot obtain the sufficient quantity of millions of instructions per second (MIPS). The third performance is the number of VM migrations. To evaluate the performance of the proposed heuristics, the CloudSim framework was used as a simulation platform. Several experiments have been implemented.

The authors compared power consumption by applying the NPA policy (all hosts run at 100% CPU usage and consume maximum power all the time). Another VM selection technique based on a single threshold is used called ST (Single Threshold). The results show that the energy consumption can be significantly reduced compared to NPA and DVFS policies by 77% and 53% respectively with 5.4% of SLA violations.

B. **Energy efficient utilization of resources in cloud computing systems :** In this article [26], the authors proposed two heuristics (ECTC and MaxUtil) for task consolidation in order to maximize the use of resources and to consider the energy of active and inactive resources. These heuristics assign each task to a resource on which the energy required to perform that task is explicitly or implicitly minimized without degrading performance. Virtualization was used and each server has several virtual machines. The objective of this work is to estimate the energy consumption. An energy model is used to calculate this energy consumption of a task. The consolidation technique allows to assign a set of tasks to a number of resources without violating the time constraint. For a given task, the heuristic checks each resource and identifies the resource that is the most energy efficient for that task. The ECTC heuristic calculates the actual energy consumption of the current task by subtracting the minimum energy consumption required to complete a task if there are other tasks running in parallel. The MaxUtil heuristic reduces both power consumption and the number of active resources.

   The authors compared the results between algorithms applied to a set of tasks (ECTC, MaxUtil and Random) and algorithms with task migration (ECTC-m, MaxUtil-m and Random-m). Task consolidation is applied to a different number of resource uses (high resource use, low or random). Experimental results show that the ECTC and MaxUtil heuristics reduce consumption of energy.

C. **Multi-criterion optimization for the allocation of distributed resources taking into account the energy :** In [27], the authors studied the problem of reducing energy consumption and CO2 emissions. Maximizing profit and ensuring quality of service (QoS) are the objectives of this article. They proposed a meta-scheduler that uses a multi-objective genetic MOGA algorithm to find the best scheduling for applications over time. The objective functions of the approach are designed to minimize energy consumption and CO2 emissions while maximizing the profit of the provider. Different parameters were used in the experiments:

the variation in the cost of arrivals customer and the price of the scheduling cycle. Four types of arrival rates were used (low, medium, high and very high speed). The experiments were conducted on realistic tracks from Feitelson's Parallel Workload Archives (PWA). A comparison was made between multi-objective genetic algorithms. The authors proposed a heuristic approach that maximizes the number of applications. . Experiments show that this heuristic reduces energy consumption by 4.66%, CO2 emissions by 10.85%, profit maximization by 1.62% and scheduling is improved by an average of 2.67%.

D. **Energy efficient management in virtualized cloud data centers :** The objective of this paper [28] is the management of energy resources. The authors proposed a decentralized architecture for resource management in the cloud. The technique of live migration of virtual machines was used. To save power, nodes are put in an enabled/disabled state using DVFS technique. This algorithm is applied to a cloud infrastructure consisting of several heterogeneous physical nodes. Each node has a CPU with high-performance processors and several virtual machines. The node in the Data Center is characterized by the amount of RAM and network bandwidth.

   The architecture of the proposed system consists of two managers located in each physical machine and a dispatcher. As soon as a user makes a request, the dispatcher distributes the request to the global manager. The global manager exchanges information on resource usage and virtual machines that need to be allocated and the placement of VM migration orders. Local managers send information to global managers about resource usage and virtual machines selected for migration. They also allow to commands to resize VMs. The VMM performs resizing, ordering and migration of virtual machines according to the orders received. Migration is used when: resource usage is too low or reaches 100%, the VM is in communication with another VM-intensive network located in another location or the temperature of a resource is too high.

   The problem of reallocating virtual machines is divided into two issues. The first is to select the virtual machine to migrate and the second is the implementation of this VM. For the selection of VMs, the authors proposed several heuristics. For the placement of the VMs, they applied the multidimensional bin packing problem (backpack problem). The authors simulated the policies: NPA (No Power Aware), DVFS, ST (Single Threshold) and TT (Two Thresholds) using the CloudSim simulator. The experimental results show the flexibility of the MM-policy.

E. **Energy efficient VM live migration in cloud data centers :** In [29], the authors proposed a method to detect over and under CPU utilization. When CPU "over-utilization" occurs, some virtual machines are migrated to another host to reduce server utilization below a specified threshold. In the case of CPU "under-utilization", all virtual machines are migrated from the source host to another to reduce the power consumed by these machines. The host in use is shut down after the migration is completed. For the allocation of VMs, they used heuristics. Migration policy and heuristics are implemented using the CloudSim simulator. The results obtained are better in energy performance than the heuristics already used in the literature.

F. **Designing and evaluating an energy efficient cloud :** The authors in [30] attempt to determine the energy cost of a virtual machine by measuring the cost when a task is running, the cost when the virtual machine has no task, and the cost of migration.

Virtualization is used to reduce energy consumption in large-scale distributed systems. The authors proposed a model called OGC (Open Green Cloud). The OGC architecture enables storage, computing and network resources to be put in an "OFF" state. To validate this infrastructure, two scheduling schemes were used: "Round-Robin" and "unbalanced". In the first scheduling, the first job is put in the first node, the second job in the second node, and so on. In the second scheduling, all jobs are put in the first node and if there are other jobs in the cloud, they use the second node.

For the experiments, four scenarios were run for each schedule to calculate the amount of energy consumed by the individual nodes in the cloud. The basic scenario has no technique to reduce energy. The "Balanced" scenario uses migration. In the third "ON/OFF" scenario, the unused nodes are switched to the "OFF" state. In the "Green" scenario, unused nodes are put in a "locked" state and the migration is used to balance the workload between the different nodes in the cloud.

To make a comparison between the different scenarios, the authors calculated the average energy consumption per node and for each schedule. Experiments show that the "Green" scenario is the scenario that consumes less energy in both types of scheduling. The "Green" Scenario with "unbalanced" scheduling consumes 25% less energy than the scenario with "Round-Robin" scheduling.

G. **Energy efficient allocation of virtual machines in cloud computing environments based on demand forecast :** Infrastructure in cloud computing consumes an enormous amount of energy. The requests for service from this system are often varied over time. The authors in [31] try to reduce this energy by making a prediction of the demands and an allocation of virtual machines. The first step is to use the Holt-Winter method to predict the different demands. The second step is to use the modified knapsack algorithm for the allocation of virtual machines in the different nodes. This algorithm takes into account two resources: CPU cores and memory capacity. In the third step, the self optimizing method was used to update the parameters of the Holt-Winter method and to find a reasonable prediction frequency. The authors compared the proposed approach with other approaches to find the most reasonable prediction period with low energy consumption and less iteration. Experimental results show that the proposed approach can save up to 60% of energy costs.

H. **Energy efficient dynamic integration of thresholds for migration at cloud data centers :** The authors in [32] proposed a dynamic threshold based on the use of the processor for a dynamic and unpredictable workload. They defined two phases: the migration phase and the placement phase. For the first phase, they proposed the Dynamic Threshold (Dyt) algorithm where they propose mathematical formulas to calculate this threshold and for the placement of virtual machines, they used the BFD (Best Fit Decreasing) algorithm. After the simulation, they compared the proposed approach (DyT) with different types of algorithms (DVFS, NPA, Single Threshold and Double Threshold). The results showed that the proposed Dyt approach attempted to minimize SLA violation by 35% to 40% and used 0.22 kWh of energy.

I. **Server farms with setup costs :** In this article [33], the authors consider server farms with an installation cost. This model is common in manufacturing systems and data centers,

where there is a cost to get the servers up and running. The setup cost always takes the form of a delay, and sometimes a power penalty, as in the case of data centers. Any server can be either ON, OFF, or in configuration mode. Although the authors analyze variants of server farms with configuration, such as models of server farms with staggered server startup, where a maximum of one server can be in configuration mode at a time, or server farms with an infinite number of servers. For some variants, the authors find that the response time distribution can be decomposed into the sum of the response time for a server farm without any configuration.

J. **Performance analysis of multi-core VMs hosting cloud SaaS applications:** Software-as-a-Service (SaaS) is the most popular cloud service model in use today, in which multi-core VMs are allocated efficiently to meet the workload offered, and in a way that avoids any violation of the agreed Service Level Agreement (SLA). This involves the need to model SaaS services in order to predict overall system performance and cost, and to estimate the required number of VM resources and their respective multi-core capacity prior to actual deployment. To this end, this paper [34] presents a mathematical queuing model to study and analyze the performance of multi-core VMs hosting SaaS applications in the cloud. There analytical model estimates, for any proposed workload, the number of instances of multi-core VMs required to meet the quality of service (QoS) parameters. The mathematical model is validated using Discrete Event Simulator (DES) simulations. The results obtained from there analysis and simulation models show that the proposed model is powerful and capable of correctly and efficiently predicting system performance and cost, as well as determining the number of VM cores needed for SaaS services to meet QoS objectives under different workload conditions.

K. **Energy efficient utilization of resources in cloud computing systems :** The energy consumption of unused resources, especially in a cloud environment, represents a significant part of the actual energy consumption. Inherently, a resource allocation strategy that takes into account resource usage would lead to greater energy efficiency; this, in the cloud, is further extended with virtualization technologies as tasks can be easily consolidated. Task consolidation is an effective method of increasing resource utilization and therefore reducing energy consumption. However, many studies have shown that server power consumption changes linearly with resource (CPU) usage. This encouraging fact underlines that task consolidation makes a significant contribution to reducing energy consumption. However, task consolidation can also lead to the release of resources that can remain inactive while continuing to consume energy. Notable efforts have been made to reduce power consumption in idle mode, typically by putting IT resources in standby or power-saving mode. In this paper [35], the authors present two power-conscious task consolidation heuristics that aim to maximize resource utilization and explicitly consider active and idle power consumption. the heuristics proposed assign each task to the resource on which energy consumption for the execution of the task is explicitly or implicitly minimized without degrading the performance of the task. On the basis of our experimental results, the heuristics demonstrate their promising energy-saving capacity.

L. **Dynamic VM allocation and traffic control to manage QoS and energy consumption in cloud computing environment :** In [36] Guaranteeing the desired quality of service (QoS) and controlling power consumption are two of the major challenges facing the cloud environment. In this paper, a mechanism is proposed that combines a virtual machine usage

scheme with a controlling access of incoming requests to the virtual machine monitor. The number of activated virtual machines is defined according to the workload, and the access control is based on the number of requests. The studied mechanism and its performance parameters are described by a mathematical model, and a power consumption model is built and evaluated. Numerical examples evaluating the parameters of quality of service. In addition, the impact of the proposed mechanism on the energy consumption behaviour is evaluated. The analysis of the results obtained shows the positive impact of the proposed mechanism.

M. **An SMDP-based resource allocation in vehicular cloud computing systems :** Among the task applications that can be handled by a cloud is Vehicle Ad-hoc Networks and should significantly improve road safety and transportation efficiency while providing a comfortable driving experience. However, the available communication, storage, and computing resources of connected vehicles are not well utilized to meet the service requirements of intelligent transportation systems (ITS). Cloud computing for vehicles (VCC) is a successful approach that takes the benefits of cloud computing and applies them to vehicle networks. In this paper, an optimal scheme for allocating computing resources to maximize the total reward expected from the VCC system in the long term is proposed. The system reward is obtained by taking into account both the revenues and costs of the VCC system as well as the variability of the available resources. Then, the optimization problem is formulated as a semi-markovian decision process (SMDP) with an infinite horizon with the defined state space, action space, reward model and transition probability distribution of the VCC system. The iteration algorithm is used to develop the optimal scheme that describes the action to be taken in a certain state. The numerical results show that the significant performance gain can be obtained by the SMDP-based scheme within an acceptable complexity.

N. **A utility-based resource allocation scheme in cloud-assisted vehicular network architecture :** Once again, in the area of the Internet of Vehicles (IoV), all components of an Intelligent Transportation System (ITS) can be connected to improve traffic safety and transportation efficiency. In order to maximize the use of resources, such as computing, communication, and storage resources, cloud computing technology could be integrated into vehicle networks. In the meantime, a cloud-assisted vehicle network could be proposed for efficient resource management. In this paper, the problem of resource allocation in the architecture of the Cloud Assisted Vehicle Network is studied. Each cloud in the architecture has its own specific characteristics, for example, the remote cloud has sufficient resources but suffers a high end-to-end delay, while the local cloud and the vehicle cloud have limited resources but a lower transmission delay is achieved. The optimal problem for maximizing the average reward expected by the system is formulated as a semi-markovian decision process (SMDP). Therefore, the corresponding optimal scheme is obtained by solving the SMDP problem by an iteration algorithm. The proposed scheme can provide guidelines that will be useful to decide in which cloud an application should be admitted and how many resources should be allocated. The numerical results indicate that the proposed system outperforms other resource allocation systems and improves system rewards and the experience of vehicle users.

0. **Managing performance and power consumption in a server farm :** where the author in [37] studied the problem of energy use when servers are powered up and down in a frame of the data centers, this means that after the setup time, all servers start running simultaneously even if the system does not require all servers. This gives us reason to reconsider the case where the configuration time of each server is considered different. A subset of servers is referred to as a "pool". Reserves are powered up when the number of jobs in the system is high enough, and are powered down when the number is low enough. Powering up takes a period of time during which the reserves consume power but do not serve the jobs. The answer to the question of how to choose the number of reserves and the rise and fall thresholds is to analyze an appropriate queuing model and minimize an appropriate cost function. Heuristic and numerical results are also presented.

2.7 Conclusion

Currently data centers provide to the global community an indispensable service, almost unlimited access to any type of information by supporting internet services such as: web hosting and e-commerce services. Energy/power consumption and QoS are the main concerns of data centers. Research into managing the power and energy consumed by data centers can make it easier to set up data centers, reduce costs and protect the environment. Given these benefits, researchers have made significant progress in conserving and reducing data center energy. In this chapter, we have presented a few techniques cited in the literature that enable the reduction of energy consumption in data centers. We are describe the two technologies, virtualization and migration used in our proposed approaches, their types, advantages and disadvantages. A state of the art on the approaches and models proposed by some of the research work was then presented. These researchers have the same goal, which is to obtain better energy consumption from data centers in order to reduce CO2 emissions and increase supplier revenues.

Chapter 3

Modeling and Analysis of Quality of Service and Energy Consumption in Cloud Environment

Contents

In this chapter, we study techniques to manage resources utilization by exploiting the concept of virtual machine migration in a cloud data center (CDC). The goal is to improve the QoS of the system, maximize resources utilization and reduce the overall energy consumption. Two algorithms are presented and studied (Combinations of Migrations (CM) and High Priority (HP)). The obtained numerical results demonstrate the effectiveness of the proposition in terms of makespan and energy efficiency while ensuring the quality of service.

3.1 Introduction

Cloud computing is an innovative technology that poses several challenges to all organizations around the world. The primary role of cloud providers is to provide a high quality of service (QoS) to their customers as long as they do not consume a lot of energy[38]. A Cloud data center (or CDC) is a focal and critical concept in cloud computing; it is a facility composed of networked servers or physical machines (PMs) used to organize, process, store and disseminate large amounts of data. Each PM can house a group of multiple virtual machines (VMs) [39]. The most important is the energy consumption concern. Indeed, the energy consumption of CDC worldwide is estimated at 26GW corresponding to about 1.4% of the global electrical energy consumption with a growth rate of 12% per year [40]. Depending on a recent study, data centers are the most energy consumers in the ICT ecosystem. Moreover, the initial cost of purchasing equipment for data center already exceeds the cost of its ongoing electricity consumption [41].

On other hand, the virtualization technique plays a central role in cloud environment, and it can be used to reduce the energy and improve the time efficiency [42]. This technology enables a single physical machine to run multiple VMs simultaneously. Moreover, through VMs migration and consolidation, virtualization reduces the total CDC energy consumption. It is known that a significant amount of power is consumed even when the PM is idle (approximately 70% of the power consumed by the PM running at full CPU utilization) [43, 44]. So, by migrating VMs and turns off the inactive PMs we can reduce energy cost. In addition, it can help to reduce the whole number of idle hosts and optimize task waiting time, execution time and operating costs [45]. Nevertheless, an aggressive consolidation of VMs can cause a delay in the execution time of a set of tasks. Therefore, could providers have to deal with energy consumption and Quality of Service trade-off. The achievement of this balance is the main objective of the work presented in this chapter. For that, we propose capacity balancing algorithms to improve system operation, reduce the waiting time, and minimize the makespan of the system. We also make a comparative analysis between our algorithms (Combination Migration Policy (CM) and the High Priority Policy (HP)) and random selecting algorithm.

This proposal technique is based on a double thresholds that manages the selection and placement of VMs in PMs, this management deal with power/performance trade-off. This chapter focused on the customer service in the system, especially the waiting time as well as the execution time and we try to minimize them with the same concept to designate two thresholds and keep the total CPU utilization between them. Furthermore, we provide numerical results with new figures and substantial discussion.

The rest of the chapter is organized as follows: Section 3.2 summarizes the General context and related work. The proposed model is presented in section 3.3. Section 3.4 presents the analysis of the proposed model. Section 3.5 presents the performance analysis and the numerical results. Finally, section 3.6 is devoted to the conclusion.

The results obtained in this chapter are presented in the research articles [46] and [47].

3.2 General context and related work

Some studies about cloud energy consumption and QoS analysis were proposed recently. For instance, Srikantaiah *et al.* [16] presented an energy-aware consolidation technique to

decrease the total energy consumption of a cloud computing system. The authors modeled the energy consumption of servers in CDC as a function of CPU and disk utilization. For that, they described a simple heuristic algorithm to consolidate the processing works in the cloud computing system. Performance of the proposed solution is evaluated only for very small input size. In [48], authors proposed a mathematical model for server consolidation in which each service was implemented in a VM to reduce number of used PM. Verma *et al.* [49] proposed and formulated a problem of energy consumption for heterogeneous CDC with workload control and dynamic placement of applications. They applied a heuristic for bin packing problem with variable bin sizes. In addition, they introduced the notion of cost of VM live migration, but the information about the cost calculation is not provided. The proposed algorithms do not handle SLA requirements.

A heuristic algorithm for dynamic adaption of VM allocation at run-time is proposed by Beloglazov *et al.* [25]. This algorithm based on the current resources utilization by applying live migration and switching idle nodes to sleep mode. The simulation results show that the proposed algorithm reduces significantly the global energy consumption. Authors in [50], proposed energy consumption formulas calculating the total energy consumption in cloud environments. They provided empirical analysis and generic energy consumption models for idle server and active server states. In [51], the authors proposed a dynamic and adaptive energy-efficient VM consolidation mechanism considering SLA constraints for CDC. Using live migration and switching idle servers to the sleep mode allow cloud providers to optimize resource utilization and reduce energy consumption, considering CPU as the main power consumers in servers. Arroba *et al.* [52] proposed an algorithm based on bin packing problem to minimize the number of bins used. The algorithm speeds up consolidation and the elastic scale out of the IT infrastructure, presenting a global utilization increase of up to 23.46% by reducing the number of active hosts by 44.91%. In [53], authors introduced a unique replication solution which considers both energy efficiency and bandwidth consumption of the geographically CDC systems. The proposed solution improves communication delay and network bandwidth between geographically dispersed CDC as well as inside of each CDC.

Other recent works in the literature focused on Markov chain and queueing theory to evaluate the performance of cloud systems. For example, Hanini *et al.* [54, 36] presented a scheme based on Continuous Markov Chain (CTMC) to manage VMs utilization in a PM with a workload control of the system. They analyzed the proposed scheme using mathematical evaluations of the QoS parameters of the system. Also, they modeled and evaluated the power consumption of the system under the proposed mechanisms. The obtained results demonstrate the usefulness of the proposed model to prevent overload in the system and to enhance its performances such as loss probability, number of jobs, and sojourn time in the system and throughput. In terms of power consumption, the proposed mechanism saves power significantly, and gives a command tool for this which is the arrival rate control parameter. Salah *et al.* [55] proposed a queuing model to predict the needed number of VMs to satisfy a predefined SLA requirement. They conducted experimental measurement on the AWS platform to validate the proposed model. The obtained results show that the proposed model can be extremely useful in achieving proper elasticity for cloud clients. In [34], the authors proposed a queuing mathematical model to study and analyze the performance of multi-core VMs hosting cloud SaaS applications. The proposed model estimates under any incoming workload the number of proper multi-core VM instances required to satisfy the QoS parameters. The queuing model is validated by simulation. The obtained results from analysis and simulation show that

the proposed model is powerful and able to predict the system performance and cost and to determine the number of VMs cores needed for SaaS services in order to achieve QoS targets under different workload conditions.

In contrast to the discussed studies, we propose efficient heuristics for dynamic adaption of allocation of VMs in runtime applying live migration according to current utilization of resources and thus minimizing energy consumption. The proposed approach can effectively handle strict QoS requirements, heterogeneous infrastructure and heterogeneous VMs. In addition, in our proposed model we have considered the time evolution of the system. The algorithms do not depend on a particular type of workload and do not require any knowledge about applications executed on VMs.

3.3 Proposed Model

3.3.1 Problem statement

In this work, we consider a data center with m homogeneous PMs, each PM contains a number of heterogeneous VMs, In order to manage the CDC, there are many objectives that have been defined in the literature, among theme, minimizing the number of PMs required or minimizing the number of VMs executed per unit of time. All these objectives contribute to minimize energy consumption or to increase the level of performance. The proposed model in this chapter aims to optimally consolidate VMs in a minimum number of PMs while minimizing the energy consumption. In addition, we aim to optimize the execution time of each PM as well as to minimize the withing time for each VM. However, consideration should be given to avoid excessive consolidation that gives us a minimal use of energy but also has a negative impact on the quality of service.

When the provider allocates several tasks to the overloaded PMs, the performance of the CDC system degrades. The idea is to define double thresholds in each PM; namely, the upper threshold and the lower threshold. These thresholds are used to keep the total CPU usage of all VMs in the PM between these two values. If the total use of a PM is below the lower threshold, then all of the VMs running in this PM must be migrated from that host, and that host must be disabled to eliminate active power consumption (Figure.3.1), in the other case if total CPU usage exceeds the upper threshold, some VMs must be migrated from that host to reduce usage and to avoid potential contract violation at the service level (Figure.3.2).

3.3.2 Mathematical formulation

In this section, we formulate our proposed solution using a mixed integer linear programming model. The notation used in this chapter are given in the Table 3.1 below.

The objective of this proposed problem is to balance the total execution time on each physical machine and minimize the number of PMs as much as possible. The objective function can be expressed as follows

$$\min \sum_{j=1}^{m} \int_{0}^{T} y_{jt} \mathrm{d}t \tag{3.1}$$

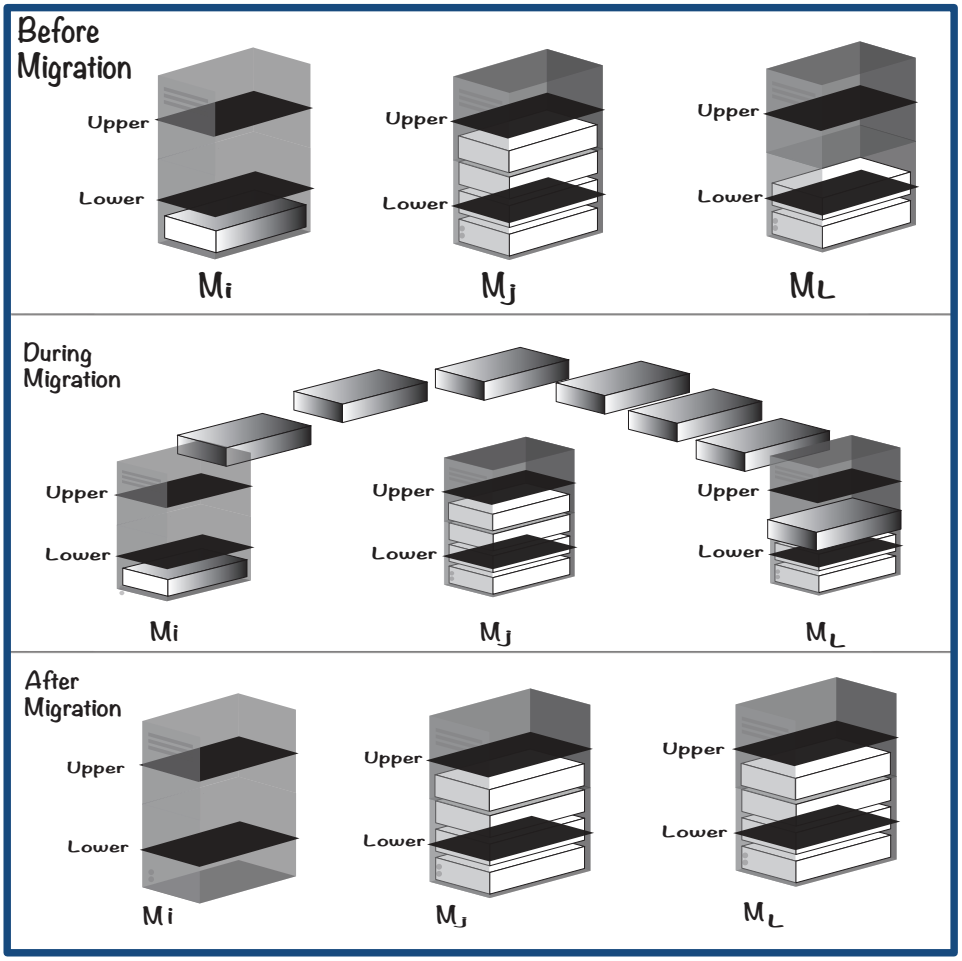

Figure 3.1 – A use case of the Lower threshold to control VM migration

Table 3.1 – Notation and Terminology

Notation	Description
m	Number of physical machines ($M_1, M_2, \ldots, M_m$)
n	Number of virtual machines in all physical machines
m'	Number of physical machines ($M_1, M_2, \ldots, M_{m'}$) possible to host the VM
v_j	Maximum number of virtual machines allocated to a physical machine M_j
x_{ijt}	bivalent variable indicating that VM_i is assigned to a M_j
C_{ij}	Capacity of each virtual machine i mapped in physical machine j
$Upper_j$	Up Threshold for each physical machine j
$Lower_j$	Down Threshold for each physical machine j
Cap_j	Total capacity of the physical machines j
$Upper$	Up Threshold for the total physical machines
P_i	Processing time of virtual machine i
s_i	Starting time of virtual machine i
T	The total execution time

where y_{jt} are intermediate variables defined by the following equation

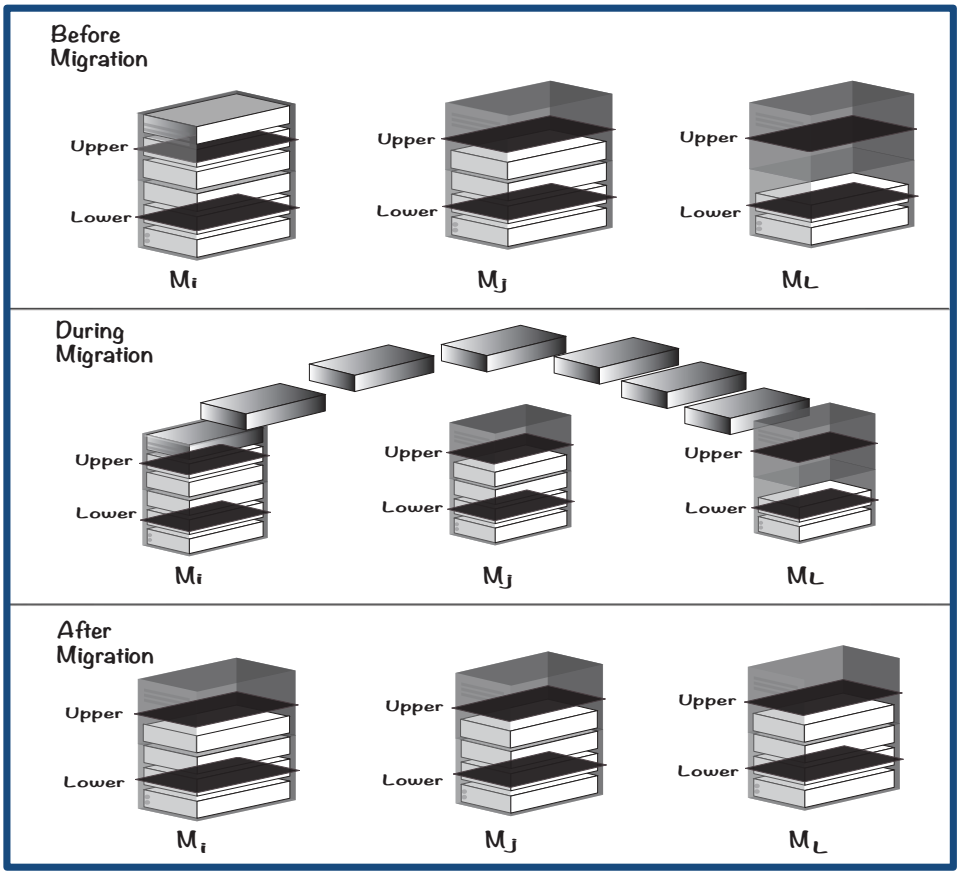

Figure 3.2 – A use case of the Upper threshold to control VM migration

$$y_{jt} = \begin{cases} 1, & \text{Machine } M_j \text{ is used at time } t \\ 0, & \text{otherwise} \end{cases} \qquad (3.2)$$
$$\forall j \in [1, m] \quad \forall t \in [0, T]$$

This optimization is subject to a number of linear constraints depending on the capacity of the host; the VM can exist only on one server at time t; and a server can host VMs according not only the amount of remaining capacity but also according to the effect of this hosting on the system [56].

- The total capacity does not exceed the capacity of system

$$\sum_{j=1}^{m} Cap_j y_{jt} \leq Upper, \quad \forall t \in [0, T] \qquad (3.3)$$

- Total capacity of virtual machines used for each physical machine M_j must be between the thresholds $Upper_j$ and $Lower_j$

$$Lower_j y_{jt} \leq \sum_{i=1}^{n} C_{ij} x_{ijt} \leq Upper_j y_{jt}, \qquad (3.4)$$
$$\forall j \in [1, m], \forall t \in [0, T]$$

- No virtual machine can exist in two physical machines at the same time t

$$\sum_{j=1}^{m} x_{ijt} \leq 1, \quad \forall i \in [1, n], \forall t \in [0, T] \tag{3.5}$$

where x_{ijt} are decision variables defined by

$$x_{ijt} = \begin{cases} 1, & i^{th} \text{ VM mapped to } j^{th} \text{ at time } t \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$
$$\forall j \in [1, m], \forall i \in [1, n], \forall t \in [0, T]$$

These can be summarized by combining the objective function with all constraints in the following set of equations

$$\begin{aligned} &\min \sum_{j=1}^{m} \int_{0}^{T} y_{jt} \mathrm{d}t \\ &\text{subject to} \quad \sum_{j=1}^{m} Cap_j y_{jt} \leq Upper, \\ &\qquad Lower_j y_{jt} \leq \sum_{i=1}^{n} C_{ij} x_{ijt} \leq Upper_j y_{jt} \\ &\qquad \sum_{j=1}^{m} x_{ijt} \leq 1, \\ &\qquad \sum_{i=1}^{n} x_{ijt} \leq v_j. \\ &\forall i \in [1, n], \forall j \in [1, m], \forall t \in [0, T] \end{aligned} \tag{3.7}$$

3.4 Model analysis

The proposed mechanism attempts to optimize the current state of the VMs in two phases. In the first phase, we select the VMs to be migrated. In the second step, using Bin Packing algorithm, the selected VMs are moved to another host that verify all constraints defined above.

A. VM selection:

When the total CPU utilization is no longer limited within the two thresholds, by either exceeding the upper threshold or being beneath the lower. Within the last case in redistributing, all the VMs must be moved to another host, while in the other case we choose just some VMs in order to migrate them to bring back the balance between the thresholds. This selection is acquainted with two policies: the Combinations of Migrations (CM) and High Priority (HP). The CM policy is an improvement of that proposed in [47] and HP is a new proposed policy based on VM utilization. The two policies are described below:

1. **The Combinations of Migrations (CM)** : This policy selects the necessary number of VMs to migrate from a host based on combination between the machines and choosing the minimum VMs they can return the CPU utilization below the upper threshold if the upper threshold is exceeded. The CM policy finds a set R_j of VMs which must be migrated from the host j. Let V_j the set of VMs currently allocated to the host j. For each VM_i in the host j, C_{ij} is its capacity. The R_j as a subset of V_j is defined by the following equations:

   if $Upper < Ucurr$:

$$R_j = \left\{ V_{ij} \in V_j \mid \min_{1 \leq k \leq |V_j|} \sum_{i=1}^{\binom{n}{k}} C_{ij} x_{ij} \geq \text{Ucurr}_j \cdot y_{jt} - \text{Upper}_j \cdot y_{jt} \right\} \tag{3.8}$$

   if $Ucurr < Lower$:

$$R_j = V_j \tag{3.9}$$

2. **High Priority (HP)** : This policy specifies a set R_j of VMs where the most used VMs have priority to stay in the same PM. All VMs having the lowest CPU usage are ordered from smallest to largest until we reach the value that exceeds the threshold are migrated. We can assume V_{ij} so that $C_{1j} < C_{2j}, \ldots$

$$R_j = \begin{cases} \{V_{1j}, \ldots, V_{kj}\} & \text{if } Upper < Ucurr \\ V_j & \text{if } Ucurr < Lower \\ \emptyset & \text{Otherwise} \end{cases} \tag{3.10}$$

   where

$$k = \text{argmin}_{1 \leq k \leq |V_j|} \sum_{i=1}^{k} C_{ij} x_{ijt} \geq Ucurr.y_{jt} - Upper.y_{jt}$$

B. VM Migration :

Finding the best location of the selected VMs can require a list of information about the latter, their number, capacity, and the former replacement of the VMs. To provide all that we need to define a bivalent variable D_{ijk} expressing replacement of VM_k from PM_i to PM_j. This bivalent variable D_{ijk} guide us to propose inequalities that have to be respected during the migration of VMs.

1. Once a VM is migrated to a server, we can not migrate it again from that server (Figure.3.3). This is reflected by the following inequality

$$D_{ijk} + D_{jlk} \leq 1, \forall l \tag{3.11}$$

2. To reinforce the previous condition, an inequality is added to ensure that when a VM_k is migrated from PM_i to PM_j at time t , the migration to the other nodes $l (l \neq j)$ is forbidden, in other words each machine has one and only one destination (Figure.3.4).

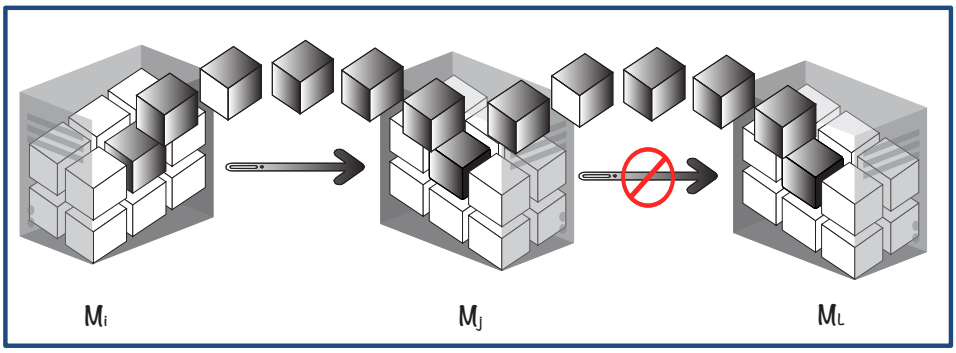

Figure 3.3 – The first constraint on VM migration

$$\sum_{j=1, j\neq i}^{m'} D_{ijk} \leq 1. \tag{3.12}$$

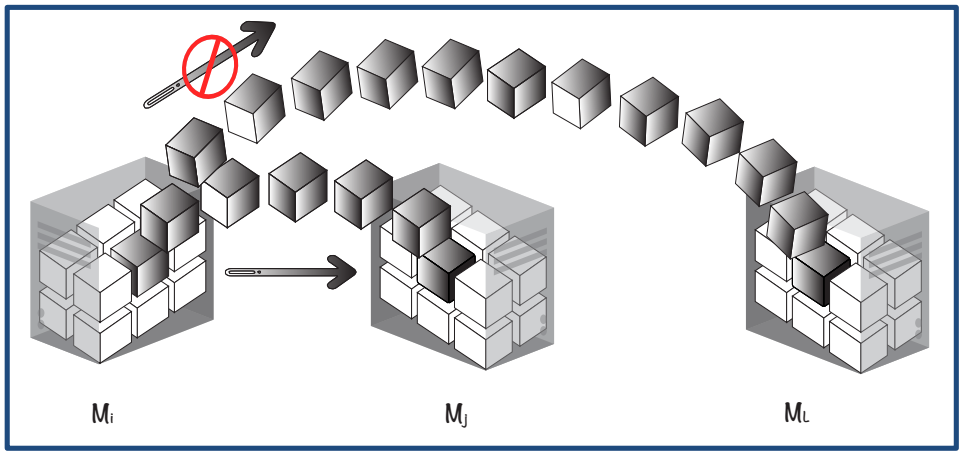

Figure 3.4 – The second constraint on VM migration

3. If a PM_i indicates that its usage is below the lower threshold, then it must migrate all of its hosted virtual machines to turn it off permanently:

$$\text{if} \quad Ucurr < Lower, \sum_{j=1, j\neq i}^{m'} \sum_{k=1}^{v_i} D_{ijk} = v_i y_{it}. \tag{3.13}$$

To solve this part we apply a First Fit Decreasing Scheduling (FFDS) bin packing algorithm. In favor of the heterogeneity of the nodes it is possible to choose the most effective power to suit the rhythm of the system. The pseudo code for the algorithm is presented in algorithm 2.

C. Decision problem :

In order to determine status of NP-hardness of this optimization problem, we describe decision problem VM-Migration as follow.

Proposition 3.1. *Given n VMs, $VM_1, \ldots, VM_n$ have to be executed without preemption and without interruption on m physical machines. Each machine is described by a starting time s_i and processing time p_i. All machines can support a number of virtual machines v_j.*

Question: Is there an allocation of virtual machines with different capacities such that the total execution time on each physical machine is does not exceed a certain value b ?

Theorem 3.1. *This decision problem will be proved to be NP-complete.*

Theorem 3.2. *Problem VM-Migration is NP-complete.*

Proof. We prove that Partition $\propto$ VM-Migration, i.e. decision problem VM-Migration belongs to NP-complete class by a reduction to Partition which is know to be is "NP-complete" [2 execution time]. . .
Recall that problem Partition is described as follow.
A finite set A of r elements $a_1, a_2, \ldots, a_r$ with integer sizes $s(a_k)$

$$\forall k, 1 \leq k \leq r, \sum_{k=1}^{r} s(a_k) = 2B. \tag{3.14}$$

Question: Is there a subset A_1 of indices such that

$$\sum_{k \in A_1} s(a_k) = \sum_{k \in \{1,2,\ldots,r\} \setminus A_1} s(a_k) = B \tag{3.15}$$

Proposition 3.2. *Given an arbitrary instance of Partition, we construct an instance of VM-Migration as follows:*

- $m = r$,
- *for each VM_i with $i \in \{1, 2, \ldots, n\}$* $\quad p_i = 1 \quad s(a_i) = C_{ij}$,
- *for physical machine M_j with $j \in \{1, 2, \ldots, m\}$:* $\quad v_j = B$,
- $b = 2$

($\Rightarrow$) Given a feasible solution to Partition, we can define a solution to VM-Migration by assigning a physical machine M_1 a subset of VMs corresponding to A_1, the remaining VMs will be allocated from M_2. This assignment satisfies the conditions and the answer to problem VM-Migration is 'Yes' since the execution time on M_1 (and on M_2 resp.) is 1.

($\Leftarrow$) If there exists a feasible solution to problem VM-Migration, then there exists a virtual machine allocation such that all virtual machines are allocated from two machines at most (due to $b = 2$ and all VMs allocated at the same time). Moreover, the maximum number of virtual machines could be used from these two machines is $v_1 + v_2 = B + B = 2B$ which corresponds exactly to the number of virtual machines allocated in PM. So, there is no redundant resource from these two machines. Hence, the total virtual machines executed on M_1 is equal to $v_1 = B$ and the subset of VMs defines then A_1. Consequently, the answer to Partition is 'Yes' ■

Algorithm 2: First Fit Decreasing Scheduling (FFDS)

```
Input: List of PMs
Output: List of VMs
SortDecreasing all virtual machines;
for Each VM in List of VMs do
    for Each PM in List of PMs do
        if Utilization of PM has not exceed upper threshold and has enough capacity
         for VM then
            Utilization of PM = Utilization of PM + Utilization of VM;
            Remain=Upper-Utilization of PM;
        if Remain is the minimum between the values provided by all VMs then
            Allocated VM in PM;
            Remove VM in List of VM;
return VMs allocation
```

3.5 Performance analysis

In migration algorithm, the upper threshold is set to avoid the SLA violations and ensure smooth task continuity. Each PM periodically executes an overload detection strategy to trigger migration. A PM is overload, when the resource utilization reaches the upper threshold. The upper threshold should be adjusted, depending on the specific system requirements, to avoid performance degradation and SLA violations. A PM is considered to be under loaded, when the resource utilization is under the lower threshold. The lower threshold significantly affects the energy efficiency and the amount of migrations.

We perform modelization to evaluate our model. Modelization has been chosen to evaluate the performance of the proposed algorithms, we have fixed the number of heterogeneous VMs to 100 and the number of PMs ranges from 10 to 100 by 10. Runs were performed to compare our methods with the one where no policy is used.

3.5.1 Analysis of energy consumption

To evaluate the efficiency of the proposed model, we evaluate the amount energy consumed in CDC. To achieve this end, the following formula is used to calculate the energy

$$E_{Tot} = E + \sum_{activePM_j} (P_j + \sum_{VM_{ij} \in V_j} \alpha \times U(VM_{ij})) \tag{3.16}$$

where E is the power needed for monitoring the CDC in idle state, P_j is the power consumption in idle state for a PM_j, $U(VM_{ij})$ is the utilization of the VM_{ij} and α is a power weight coefficient.

The influence of high and low thresholds is evaluated on energy efficiency in data center which consists of 100 VMs.

In Figure 3.5, we plot the amount of energy consumed under our proposed algorithms compared to the case where a random selecting algorithm is used. As the Figure 3.5 shows, the system energy consumption vary with the value of number of PMs. The tendencies of the

system energy consumption obtained from the two experiences are almost similar. We remark that, when the number of PM is more than 10, CM policy and HP policy are both minimizing the total consumed energy. However, it increases faster from 70 to 90 and the total energy consumption is always lower when using our algorithm, in this case of energy consumption, CM policy outperform HP policy by minimized more energy. The total consumed energy is minimizing by an average of 20% (respectively 16%) by applying CM policy (respectively HP policy), the saved energy is due to the way of migration proposed in our algorithm that turn off unneeded PMs.

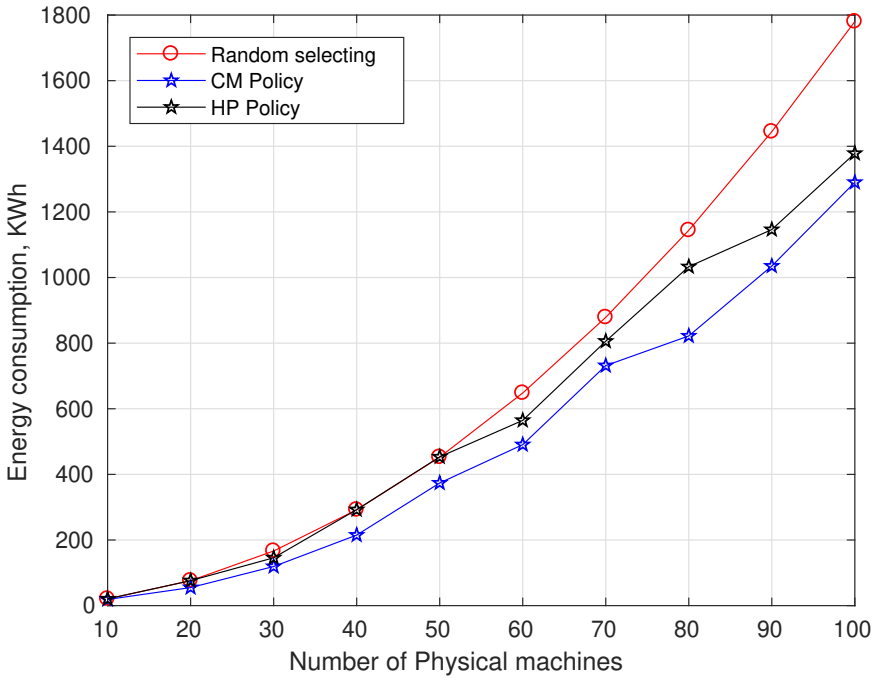

Figure 3.5 – Comparison of the system energy consumption

3.5.2 Analysis of execution time and CPU utilization

In this part, our prime aim is to minimize the makespan which is the time required to complete the execution of all input tasks by the system. It can be represented as follow

$$Makespan = \max_{1 \leq j \leq m} \sum_{i=1}^{v_j} AccoTime_{ij} \tag{3.17}$$

where, $AccoTime_{ij}$ is the accomplishment time of i^{th} VM on j^{th} PM [57]. During the execution of task, our proposed algorithm is about minimizing the total execution time on each physical machine. We assume that all the VMs are running respectively from he highest to smallest and each VM/task could be executed in the period $[s_{ij}, s_{ij} + p_{ij}]$.

where s_{ij} is the starting time and p_{ij} is the processing time.

The total running capacity of PM $_j$ is the sum of capacity of all VMs running in this PM.

$$Cap_j = \sum_{i=1}^{v_j} C_{ij} \tag{3.18}$$

Now, the duration time of i^{th} VM is estimated using

$$AccoTime_{ij} = C_{ij} \times S \tag{3.19}$$

Where S is the time needed to implement a unit capacity

To analyze the performance of our proposed algorithm FFDS in function of the waiting time earned by applying the HP policy, we perform an evaluation with 10 PMs and 60 VMs with $S = 0.02$ ms. Figure 3.6 illustrates the execution time in two cases: (1) in case of random selecting (2) in case of HP policy. We remark that the execution time has been decreased significantly by an average of 69%. These results show one other great advantage of our algorithm regrading the execution time of VMs in CDC.

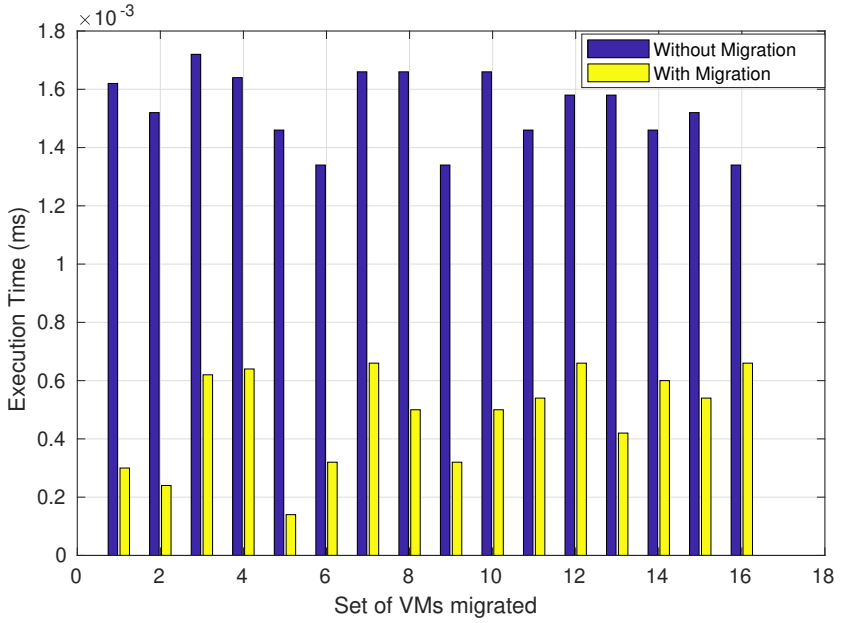

Figure 3.6 – The execution time vs migrated VMs

The next experiment focuses on the impact of our algorithm applying both policies CM and HP to manage the CPU utilization between 10 PMs mapping 60 VMs. As shown in Figure 3.7, the selecting random deal with allocation the VMs in a wrong way, where some PMs exceed the upper threshold while other PMs are active just to serve some VMs that have a small capacities, and that poses two problems, the first depends on the energy consumed, which will be higher because all the PMs are active, the second concerns the overload on some PMs which can have a negative impact on the quality of service (Figure 3.6). On the other hand, both policies find a way to improve significantly the CPU utilization of the system, and this is done by keeping the CPU utilization of all PMs between the upper and lower thresholds that are in our experiment 70% and 30% respectively.

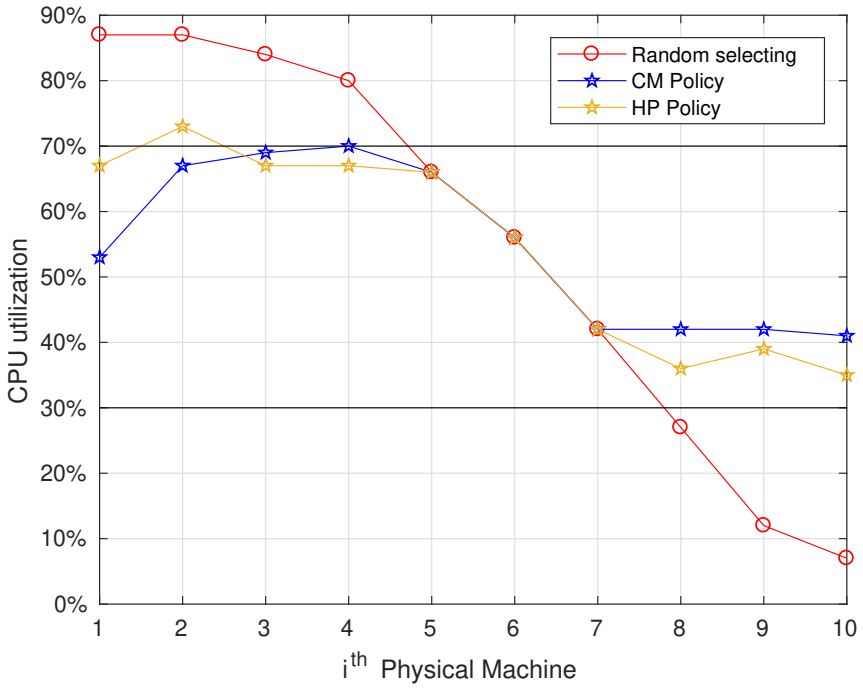

Figure 3.7 – Variation of CPU utilization on each physical machine

3.5.3 Cost of VM migration

VMs migrate at the beginning of each monitor period. It has negative impacts on performance of the running tasks. We assume that each VM migration costs the same amount of resources. So it is crucial to minimize the number of VM migrations ensuring the QoS and energy conservation. In this work we are interested in evaluating the cost of migration with a view to study and minimize it in future work. Each migration has a cost in QoS. However, when the migration is monitored due to exceeding of the upper threshold, the QoS of the migrated machine and of remained VMs in the PM where the migration is performed from is improved. Then, the total cost of this operation can be computed using the following formula

$$Cost_{Mig} = \beta \times NM_{Tot} - \gamma \times NM_{up} \tag{3.20}$$

where NM_{Tot} is the total number of migrated VM, NM_{up} is the number of VM migrated due to exceeding of the upper threshold, β is the cost of a migration and γ is the gain in QoS when the migration is monitored due to exceeding of the upper threshold.
Moreover, we can suppose that β is very small compared to γ based to the fact that migration techniques used in CDC are developed and permits minimizing the cost due to this operation. In our experimentation, we use $\beta = 2$ and $\gamma = 4$
Table 3.2 shows the cost of migration when the number of VMs is varying. We remark that when the number of VMs is increasing, the cost becomes not positive, which means that we gain at the QoS. This is due to the fact that there is more chance to have an exceeding of the upper threshold.

Figure 3.8 shows that when the number of PMs increases the provider has more gain than loss (negative cost). When we compare the two proposed policies (CM and HP), we remark

Table 3.2 – Cost of migration

Number of VM	Number of migration due to exceeding upper threshold	Cost
10	1	30
20	8	-32
30	12	-26
40	17	-50
50	14	-10
60	25	-80
70	29	-100
80	25	-58
90	38	-130
100	40	-122

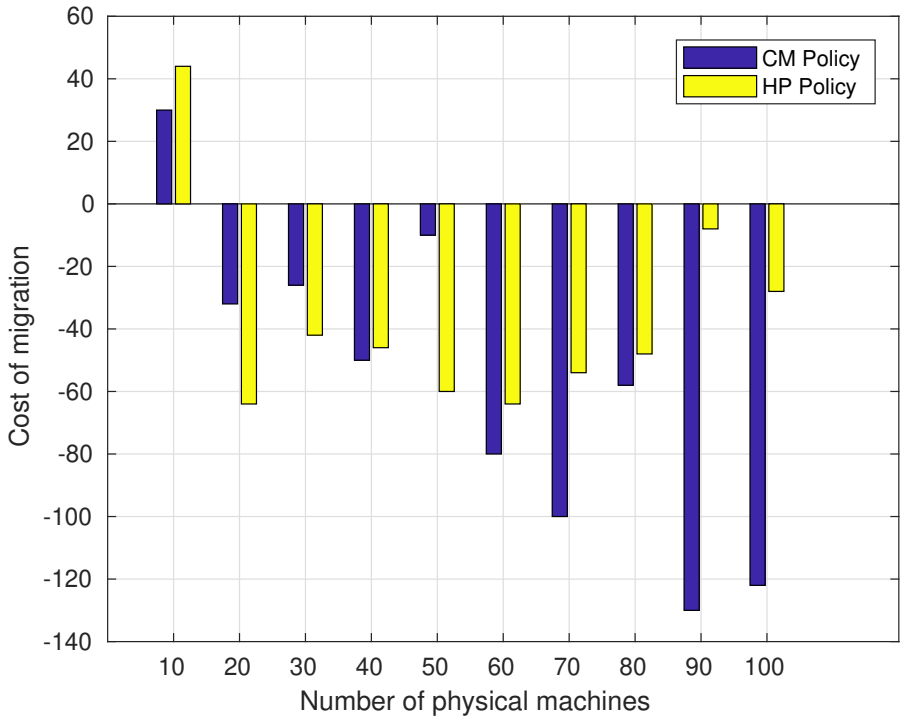

Figure 3.8 – The comparison of the double threshold policies in terms of cost

that HP policy performs better with lower number of PMs, and CM policy is better with large number of PMs.

3.6 Conclusion

In this chapter we have proposed a capacity balancing algorithm to manage the VM migration between the PM in a CDC in order to deal with power-performance trade-off. Our proposition is based on double thresholds that manage the selection and placement of VMs in physical

machines using both policies CM and HP. This model is presented as a mixed integer linear mathematical model. The numerical evaluation of the proposed technique showed that our model enables to save energy consumed and minimizes the execution time in CDC. However, this work has to be detailed in terms of QoS performances.

Chapter 4

Analysis of a M/M/k system with exponential setup times and reserves servers

Contents

In this chapter, we introduce the concept of reserve server and we consider the M|M|k queuing system with setup costs: M|M|k|Setup to deal with coming clients. The proposed mechanism and its performance parameters are described by a a mathematical model. In addition, the effect of the proposed mechanism on energy consumption behaviour is assessed. The discussion on the results shows the role of the consideration of reserves in the system.

4.1 Introduction

Cloud computing is now one of the major issues in the evolution of information technology by allowing the possibility of reserving IT resources online. Some commercial offers are already in existence for customers (residential or professional) supported by large data centres such as Amazon or Azure [58]. The amount of power consumed by server represents more than 1.5% of all electricity consumption in the United States, at a price of approximately $4.5 billion [59]. The cost of energy and the huge growth of data centres will lead to more energy consumption spending [60]. Most of the time, only 20 to 30% of the servers' total storage capacity of energy is used on average. The main cause of this wastage is the inactive machines in the server farms that are too large. A critical issue in modern server farm system architecture is to know how to efficiently allocate the number of servers to manage unpredictable demand patterns., in order to get the best performance without wasting energy.
Although it is demanded to shut down servers when they become inactive to save energy, the high configuration cost (in terms of setup time and energy consumption) required to restart the server can affect performance. The problem is even more complicated since today's servers offer several states of sleep or standby that allow the setup cost to be balanced by the power consumed while the server is "in sleep". The idle servers use about 60% of their maximum power [61], although it is possible to save a lot of energy by switching off the inactive servers when we don't need them [46, 44]. However, turning on an off server has a major cost. The setup cost may take the form of a time delay, which we call it the setup time. The option that we suggest in this work is to start with the minimum of servers and we put the others in a set of reserves and we turn on one by one when we need them and their situations are on sleep state. Although a server in idle state consumes more power than a server in off state, but the cost of setting up a server in idle state is lower than that of a server in off state state [62].

In this chapter, we focus on an M/M/k queue system with setup costs M/M/k/Setup. Servers are divided in two types: in run mode and in sleep mode as a reserve when there is no work to be done, but when the system needs more server to be on the state on we turning on a server among the reserves from in sleep mode and this involves setup costs. The setup cost is in the form of a delay. Since a reserve server consume a lot of energy, the number of servers that can be as a reserve at any time is often limited. In the configuration model, a maximum of one server can be configured at any one time. Although recent literature has analyzed an M/M/k system with exponentially distributed setup times [63], no closed-form solutions have been obtained. Because in our work we add the concept of reserves, we supply the primary analytic expressions for the limiting distribution of system states, response time distribution and average energy consumption for the system.

The rest of this chapter is organized as follows. The proposed system model is presented in Section 4.3. Section 4.4 formulates the proposed model. The performance measures are presented in Section 4.5. Section 4.6 presents the numerical results. Finally, Section 4.7 concludes the chapter.

The results obtained in this chapter are presented in the research article [64].

4.2 General context and related work

Setup times are a fundamental component of computer systems and manufacturing systems, and therefore they have always played an important role in queueing theoretic analysis. In manufacturing systems it is very common for a job that finds a server idle to wait for the server to "warm up" before service is initiated. In retail and hospitals, the arrival of customers may necessitate bringing in an additional human server, which requires a setup time for the server to arrive. In computer systems, setup times are once again at the forefront of research, as they are the key issue in dynamic capacity provisioning for data centers.
In data centers, it is desirable to turn idle servers off, or reallocate the servers, to save power. This is because idle servers burn power at 60%_70% of the peak rate, so leaving servers on and idle is wasteful [65]. Unfortunately, most companies are hesitant to turn off idle servers because the setup time needed to restart these servers is very costly; the typical setup times for servers is 200 seconds, while a job's service requirement is typically less than 1 second [66, 67]. Not only is the setup time prohibitive, but power is also burned at peak rate during the entire setup period, although the server is still not functional. Thus it is not at all obvious that turning off idle servers is advantageous. Many ideas have been proposed to minimize the number of times that servers in a data center must undergo setup. One major line of research involves load prediction techniques [66, 68, 69, 70]. In the case where load is unpredictable, research has turned to looking at policies such as delayed off, which delay turning off an idle server for some fixed amount of time, in anticipation of a new arrival [71, 72, 73]. Another line of research involves reducing setup times by developing low power sleep modes [72, 73]. Surprisingly, for all the importance of setup times, very little is known about their analysis. The M/G/1 with setup times was analyzed in 1964 by Welch [74]. The analysis of an M/M/k system with setup times, which we refer to as M/M/k/setup, however, has remained elusive, owing largely to the complexity of the underlying Markov chain. (Figure. 1.1 shows an M/M/k/setup with exponentially distributed setup times.) In 010, various analytical approximations for the M/M/k/setup were proposed in [33]. These approximations work well provided that either load is low or the setup time is low. The M/M/∞/setup was also analyzed in [33] and found to exhibit product form. Other than the above, no progress has been made on the M/M/k/setup. Even less is known about the M/M/k/setup/delayed off, where idle servers delay for a finite amount of time before turning off, or the M/M/k/setup/sleep, where idle servers can either be turned off (high setup time, zero power) or put to sleep (lower setup time, low power).

For the M/M/k/setup/delayed off, only iterative matrix-analytic approaches have been used [73]. No analysis exists for M/M/k/setup/sleep. We now discuss papers that have considered repeating Markov chains and have proposed techniques for solving these. We then comment on how these techniques might or might not apply to the M/M/k/setup.

Matrix-analytic based approaches : Matrix-analytic methods are a common approach for analyzing Markov chains with repeating structure. Such approaches are typically numerical, generally involving iteration to find the rate matrix, R. These approaches do not, in general, lead to closed forms or to any intuition, but are very useful for evaluating chains under different parameters. There are cases where it is known that the R matrix can be stated explicitly [75]. This typically involves using a combinatorial interpretation for the R matrix. As described in [75], the class of chains for which the combinatorial view is tractable is narrow. However, in

[76], the authors show that the combinatorial interpretation extends to a broader class of chains. Their class does not include the M/M/k/setup, however, which is more complicated because the transition (setup) rates are not independent of the number of jobs in system. Much research has been done on improving matrix-analytic methods to make the iteration faster. An example is [77], which develops a fast iterative procedure for finding the rate matrix for a broader class of chains than that in [76]. The authors in [77] also provide an explicit solution for the rate matrix in terms of infinite sums.

Generating function based approaches : Generating functions have also been applied to solve chains with a repeating structure. Like matrix-analytic methods these are not intuitive: Generating function approaches involve guessing the form of the solution and then solving for the coefficients of the guess, often leading to long computations. In theory, they can be used to solve very general chains (see for example [78]). We initially tried applying a generating function approach to the M/M/2/setup and found it to be incredibly complex and without intuition. This led us to seek a simpler and more intuitive approach

M/M/k with vacations : Many papers have been written about the M/M/k system with vacations, see for example [79, 80, 81, 82]. While the Markov chain for the M/M/k with vacations looks similar to the M/M/k/setup, the dynamics of the two systems are very different. A server takes a vacation as soon as it is idle and there are no jobs in the queue. By contrast, a setup time is initiated by jobs arriving to the queue. In almost all of the papers involving vacations, the vacation model is severely restricted, allowing only a fixed group of servers to go on vacation at once. This is very different from our system in which any number of servers may be in setup at any time. The model in [82] comes closest to our model, although the authors use generating functions and assume that all idle servers are on vacation, rather than one server being in setup for each job in queue, which makes the transitions in their chain independent of the number of jobs.

Restricted models of M/M/k with setup : There have been a few papers [83, 84, 33] that consider a very restricted version of the M/M/k/setup, where in at most one server can be in setup at a time. There has also been prior work [37] and [85] that considers an M/M/k system where in a fixed subset of servers can be turned on and off based on load. The underlying Markov chains for all of these restricted systems are analytically tractable and lead to very simple closed form expressions, since the rate at which servers turn on is always fixed. Our M/M/k/setup system is more general, allowing any number of servers to be in setup. This makes our problem much more challenging.

4.3 Model description

We model a server with setup time using M/M/k queuing system, with a Poisson arrival process with rate λ, and the service time is exponentially distributed μ. Let $\rho = \lambda/\mu$ denote the system load, where $\rho < k$, In this model a server can be in one of three states: On, in setup, or reserve. Servers are obviously shut down when there is no work to be done, but turning on a server that has been shut down will incur setup costs. The setup cost takes the shape of a delay and a power penalty, event that idle servers are shut down to save on running expenses.

Since servers in setup mode consume a lot of energy, the number of servers that can be in setup at any given time is often limited. In the configuration model, a maximum of one server can be setup at any time. A server is in the ON state when it is serving jobs, when the server is on, it consumes power P_{on}, if there are no jobs to serve, the server turns off instantaneously, While in the reserve state, the server consumes power P_s lower than P_{on}, and to switch on a server among the reserves, it must first put in setup mode. However, we suggest than, at most one server can be in setup at any time, while in setup mode , the time need to turn on a server is called the setup time, and during that entire time, power P_{on} is consumed and we model the setup time as an exponentially distributed random variable I with rate $\alpha = \frac{1}{E[I]}$.

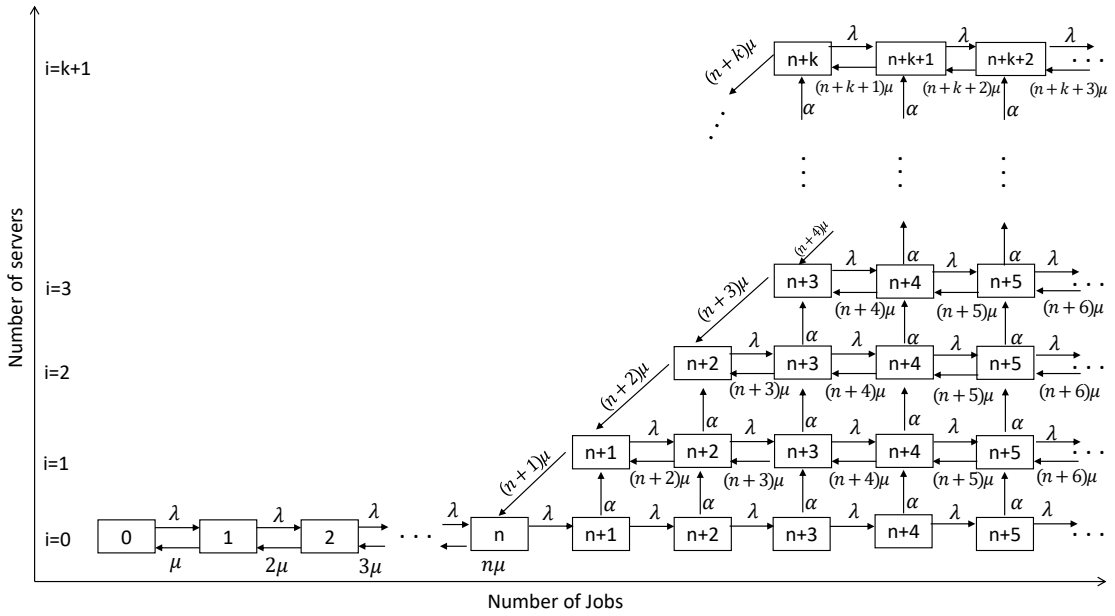

Figure 4.1 – Markov chain for an M/M/k/Setup

As in an M/M/k queuing system, we assume a First Come First Serve queue, from which servers serve jobs when they become available. Hence, the setup costs increase the complexity. From a job viewpoint, if a job comes in and finds a server in setup, it simply waits in the queue. However, if the job does not find any servers in the setup, then it randomly selects a server from the set of the reserves and puts it in the setup state. If the task cannot find any servers out of order, it simply waits in the queue. When a job ends the service on a server j, the first task in the queue is moved to server j, without the need for any setup, since server j is already enabled. Even if the job at head of the queue was already pending on another server i to setup, the task at head of the queue is still sent to the server j; server i is then either reconsidered as a reserve if there is no task in the queue, or remains in setup if the queue is not empty. We call this model the M/M/k/Setup model.

4.4 Model Analysis

In this section, we obtain the limit probabilities of the system states (theorem 1) for an M/M/k/Setup. Due to space constraints, we only present brief proofs. Additional details of the proofs will be found in the extending work.

The figure 4.1 shows the Markov chain for an M/M/k/Setup. The states of the Markov chain are designated by (a, b), where a denotes the number of servers that are powered on, and b represents the number of tasks in the system. The Markov chain is composed of $(k+1)$ rows. The first row (from the bottom) consists of states where we have that the main servers are on, the second row consists of states where we have exactly one server in setup from the reserves, and so on. For setup time, be reminded that only one server can be in setup at any time. Thus, the rate of passage from state $(i; j)$ to state $(i+1, j)$ is a constant α for all $0 \leq i < k$ and $i < j$.

Now we will solve the Markov chain illustrated in Figure 4.1 for the limiting probabilities of to be in any state. Firstly, we find the limit probabilities for the 1st row states, in terms of $\pi_{0,0}$. Then, we determine the limiting probabilities of being in the states of the second row. In relation to the first row solution, which in the meantime is given in terms of $\pi_{0,0}$. By going on in this manner, we are able to solve the limiting probabilities of all the states in the Markov chain in terms of $\pi_{0,0}$.

Theorem 4.1. *The limiting probabilities for an M/M/k/Setup are given by :*

$$\pi_{i,j} = \frac{\rho^n}{(n+1)!}\frac{\pi_{0,0}\cdot\gamma^i}{i!}\beta^j \quad \text{for } 0 \leq i < k \text{ and } j > n+i-1$$

$$\pi_{k,j} = \frac{\rho^n}{(n+1)!}\left[\frac{\pi_{0,0}\gamma^k k\mu\beta^j}{k!\cdot(k\mu-(\lambda+\alpha))} - \frac{\pi_{0,0}k^k(\lambda+\alpha)\left(\frac{\rho}{k}\right)^j}{k!\cdot(k\mu-(\lambda+\alpha))}\right] \quad \text{for } j > k-1$$

$$\pi_{0,0} = \left[\sum_{j=0}^{n}\frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!}\cdot\frac{1}{(1-\beta)}\cdot\left[\sum_{0\leq i<k}\frac{\rho^i}{i!} + \frac{\rho^k\mu}{(k-1)!\cdot(k\mu-\lambda)}\right]\right]^{-1}$$

where $\alpha = \frac{1}{\mathbb{E}[I]}, \beta = \frac{\lambda}{\lambda+\alpha}$ *and* $\gamma = \frac{\lambda+\alpha}{\mu}$

Proof. The relevant balance equations for the 1st row are given by :

$$\lambda\cdot\pi_{0,0} = \mu\cdot\pi_{0,1} \quad \text{for } j = 0 \tag{4.1}$$

$$\begin{aligned}(\lambda+j\mu)\cdot\pi_{0,j} &= \lambda\cdot\pi_{0,j-1} + (j+1)\mu\cdot\pi_{0,j+1}\\ \Longrightarrow \pi_{0,j} &= \pi_{0,0}\cdot\frac{\rho^n}{(n)!} \qquad \text{for } j = 1,\ldots,n,\end{aligned} \tag{4.2}$$

$$\begin{aligned}(\lambda+\alpha)\cdot\pi_{0,j} &= \lambda\cdot\pi_{0,j-1}\\ \Longrightarrow \pi_{0,j} &= \pi_{0,0}\cdot\frac{\rho^n}{(n)!}\cdot\beta^j \quad \text{for } j \geq n,\end{aligned} \tag{4.3}$$

The relevant balance equations for the 2nd row are given by, when j>1

$$(\lambda+\alpha+(n+1)\mu)\cdot\pi_{1,j} = \lambda\cdot\pi_{1,j-1} + \alpha\cdot\pi_{0,j} + (n+1)\mu\cdot\pi_{1,j+1} \tag{4.4}$$

The right-hand side of (4.4) above consists of states of the 2nd row as well as states of the 1st row. Thus, we use difference equation (see [63] for more information on difference equations) to solve for $\pi_{1,j}$

$$\pi_{1,j} = A_{1,1}x^j + A_{1,2}\beta^j \text{ for } j > 1 \tag{4.5}$$

where x is a solution of the homogeneous equation:

$$x \cdot (\lambda + \alpha + (n+1)\mu) = \lambda + x^2 \cdot (n+1)\mu \tag{4.6}$$

We now solve for $A_{1,2}$ by plugging in equations. (4.3) and (4.5) into (4.4) for $j > n+2$.

$$(A_{1,1}x_1^j + A_{1,2}\beta^j) \cdot (\lambda + \alpha + (n+1)\mu) = (A_{1,1}x_1^{j-1} + A_{1,2}\beta^{j-1}) \cdot \lambda + \pi_{0,0}\tfrac{\rho^n}{n!} \cdot \beta^j \cdot \alpha + (A_{1,1}x_1^{j+1} + A_{1,2}\beta^{j+1}) \cdot (n+1)\mu$$

$$A_{1,1}x_1^j(\lambda + \alpha + (n+1)\mu) + A_{1,2}\beta^j(\lambda + \alpha + (n+1)\mu) = A_{1,1}x_1^{j-1}\lambda + A_{1,2}\beta^{j-1}\lambda + \pi_{0,0}\tfrac{\rho^n}{n!}\beta^j\alpha + A_{1,1}x_1^{j+1}(n+1)\mu + A_{1,2}\beta^{j+1}(n+1)\mu$$

Where

$$A_{1,1}x_1^{j-1}(x_1[\lambda + \alpha + (n+1)\mu) - \lambda - x_1 2(n+1)\mu] = 0$$

after equation (4.6)

$$A_{1,2}\beta^j(\lambda + \alpha + (n+1)\mu) - A_{1,2}\beta^{j-1}\lambda - A_{1,2}\beta^{j+1}(n+1)\mu = \pi_{0,0}\frac{\rho^n}{n!}\beta^j\alpha$$

$$A_{1,2}(\beta(\lambda + \alpha + (n+1)\mu) - \lambda - \beta^2(n+1)\mu = \pi_{0,0}\frac{\rho^n}{n!}\beta\alpha$$

$$\implies A_{1,2}\frac{\lambda(n+1)\mu\alpha}{(\lambda+\alpha)^2} = \pi_{0,0}\frac{\rho^n}{n!} \cdot \frac{\lambda\alpha}{\lambda+\alpha}$$

$$\implies A_{1,2} = \pi_{0,0} \cdot \frac{\lambda+\alpha}{(n+1)\mu}\frac{\rho^n}{n!}$$

$$\implies A_{1,2} = \frac{\rho^n}{(n+1)!} \cdot \pi_{0,0} \cdot \gamma \quad \text{Where } \gamma = \frac{\lambda+\alpha}{\mu} \tag{4.7}$$

To get $A_{1,1}$, we use the balance equation for $\pi_{1,n+2}$, which will contain $\pi_{1,n+1}$. However, we first need to evaluate $\pi_{1,n+1}$. This requires us to use the balance equation for $\pi_{0,n}$

$$(\lambda + n\mu) \cdot \pi_{0,n} = \lambda \cdot \pi_{0,n-1} + (n+1)\mu \cdot \pi_{1,n+1}$$

$$\implies \pi_{1,n+1} = \frac{\lambda + n\mu}{(n+1)\mu} \cdot \pi_{0,n} - \frac{\lambda}{(n+1)\mu} \cdot \pi_{0,n-1}$$

$$\implies \pi_{1,n+1} = \frac{\lambda}{(n+1)\mu} \cdot \pi_{0,0}\frac{\rho^n}{(n+1)!} + \frac{n}{(n+1)\mu} \cdot \pi_{0,0}\frac{\rho^n}{(n)!} - \frac{\rho}{(n+1)} \cdot \pi_{0,0}\frac{\rho^{n-1}}{(n-1)!}$$

$$\implies \pi_{1,n+1} = \pi_{0,0} \cdot \frac{\rho^{n+1}}{(n+1)!} \cdot \beta^n \tag{4.8}$$

Now we use equations (4.3) and (4.5) into (4.4) for $j = n+2$

$$(4.4) \implies (\lambda + \alpha + (n+1)\mu) \cdot \pi_{1,n+2} = \lambda \cdot \pi_{1,n+1} + \alpha \cdot \pi_{0,n+2} + (n+1)\mu \cdot \pi_{1,n+3}$$

$$\Longrightarrow (\lambda+\alpha+(n+1)\mu)\cdot(A_{1,1}x^{n+2}+A_{1,2}\beta^{n+2}) = \pi_{0,0}\cdot\tfrac{\rho^{n+1}}{(n+1)!}\cdot\lambda\beta^n+\alpha\pi_{0,0}\cdot\tfrac{\rho^n}{n!}\cdot\beta^{n+2}+(n+1)\mu\cdot(A_{1,1}x^{n+3}+A_{1,2}\beta^{n+3})$$

$$\Longrightarrow (\lambda+\alpha+(n+1)\mu)\cdot A_{1,1}x^{n+2}-(n+1)\mu\cdot A_{1,1}x^{n+3} = \pi_{0,0}\cdot\tfrac{\rho^{n+1}}{(n+1)!}\cdot\lambda\beta^n+\alpha\pi_{0,0}\cdot\tfrac{\rho^n}{n!}\cdot\beta^{n+2}+A_{1,2}\beta^{n+3}(n+1)\mu-A_{1,2}\beta^{n+2}(\lambda+\alpha+(n+1)\mu)$$

$$\Longrightarrow (\lambda+\alpha+(n+1)\mu)\cdot A_{1,1}x_1^{n+2}-(n+1)\mu\cdot A_{1,1}x_1^{n+3} = \pi_{0,0}\cdot\tfrac{\rho^{n+1}}{(n+1)!}\cdot\lambda\beta^n+\alpha\pi_{0,0}\cdot\tfrac{\rho^n}{n!}\cdot\beta^{n+2}+A_{1,2}\beta^{n+2}\left((n+1)\mu\beta-\lambda-\alpha-(n+1)\mu\right)$$

$$(4.7)\Rightarrow A_{1,1}\left[(\lambda+\alpha+(n+1)\mu)x_1^{n+2}-(n+1)\mu x_1^{n+3}\right] = \pi_{0,0}\tfrac{\rho^{n+1}}{(n+1)!}\lambda\beta^n+\alpha\pi_{0,0}\tfrac{\rho^n}{n!}\beta^{n+2}+\tfrac{\rho^n}{(n+1)!}\pi_{0,0}\gamma\beta^{n+2}\left[(n+1)\mu\beta-\lambda-\alpha-(n+1)\mu\right]$$

$$\Longrightarrow A_{1,1}\left[(\lambda+\alpha+(n+1)\mu)x_1^{n+2}-(n+1)\mu x_1^{n+3}\right] = \pi_{0,0}\cdot\tfrac{\rho^{n+1}}{(n+1)!}\cdot\lambda\beta^n+\alpha\pi_{0,0}\cdot\tfrac{\rho^n}{n!}\cdot\beta^{n+2}-\tfrac{\rho^n}{(n+1)!}\cdot\pi_{0,0}\cdot\gamma\beta^{n+2}\left[\tfrac{\lambda^2+2\lambda\mu+\alpha^2+\alpha\mu(n+1)}{\lambda+\alpha}\right]$$

$$\Longrightarrow A_{1,1}\left[(\lambda+\alpha+(n+1)\mu)x_1^{n+2}-(n+1)\mu x_1^{n+3}\right] = \pi_{0,0}\cdot\tfrac{\rho^{n+1}}{(n+1)!}\cdot\lambda\beta^n+\alpha\pi_{0,0}\cdot\tfrac{\rho^n}{n!}\cdot\beta^{n+2}-\tfrac{\rho^n}{(n+1)!}\cdot\pi_{0,0}\cdot\gamma\beta^{n+2}\left[\tfrac{\lambda^2+2\lambda\mu+\alpha^2+\alpha\mu(n+1)}{\lambda+\alpha}\right]$$

$$\Longrightarrow A_{1,1}\left[(\lambda+\alpha+(n+1)\mu)x_1^{n+2}-(n+1)\mu x_1^{n+3}\right] = \pi_{0,0}\cdot\tfrac{\rho^n}{n!}\left[\tfrac{\lambda^2}{(n+1)\mu}\beta^n+\alpha\beta^{n+2}\right]-\tfrac{\rho^n}{(n+1)!}\cdot\pi_{0,0}\cdot\gamma\beta^{n+2}\left[\tfrac{\lambda^2+2\lambda\mu+\alpha^2+\alpha\mu(n+1)}{\lambda+\alpha}\right]$$

$$\Longrightarrow A_{1,1}\left[(\lambda+\alpha+(n+1)\mu)x_1^{n+2}-(n+1)\mu x_1^{n+3}\right] = \pi_{0,0}\cdot\tfrac{\rho^n}{n!}\beta^n\lambda^2\left[\tfrac{1}{(n+1)\mu}+\alpha\right]-\tfrac{\rho^n}{(n+1)!}\cdot\pi_{0,0}\cdot\gamma\beta^{n+2}\left[\tfrac{\lambda^2+2\lambda\mu+\alpha^2+\alpha\mu(n+1)}{\lambda+\alpha}\right]$$

$$\Longrightarrow A_{1,1}\left[(\lambda+\alpha+(n+1)\mu)x_1^{n+2}-(n+1)\mu x_1^{n+3}\right] = \pi_{0,0}\cdot\tfrac{\rho^n}{n!}\beta^n\tfrac{\lambda^2}{(\lambda+\alpha)^2(n+1)\mu}\left[(\lambda+\alpha)^2+\alpha\mu(n+1)\right]-\tfrac{\rho^n}{(n+1)!}\cdot\pi_{0,0}\cdot\gamma\beta^{n+2}\left[\tfrac{\lambda^2+2\lambda\mu+\alpha^2+\alpha\mu(n+1)}{\lambda+\alpha}\right]$$

$$\Longrightarrow A_{1,1}\left[(\lambda+\alpha+(n+1)\mu)x_1^{n+2}-(n+1)\mu x_1^{n+3}\right] = \pi_{0,0}\cdot\tfrac{\rho^n}{(n+1)!}\beta^{n+2}\tfrac{1}{\mu}\left[\lambda^2+2\lambda\alpha+\alpha^2+\alpha\mu(n+1)\right]-\tfrac{\rho^n}{(n+1)!}\cdot\pi_{0,0}\tfrac{\lambda\alpha}{\mu}\beta^{n+2}\left[\tfrac{\lambda^2+2\lambda\mu+\alpha^2+\alpha\mu(n+1)}{\lambda\alpha}\right]$$

$$\Longrightarrow A_{1,1}\left[(\lambda+\alpha+(n+1)\mu)x_1^{n+2}-(n+1)\mu x_1^{n+3}\right] = \cancel{\pi_{0,0}\cdot\tfrac{\rho^n}{(n+1)!}\beta^{n+2}\tfrac{1}{\mu}\left[\lambda^2+2\lambda\alpha+\alpha^2+\alpha\mu(n+1)\right]}-\cancel{\tfrac{\rho^n}{(n+1)!}\cdot\pi_{0,0}\beta^{n+2}\tfrac{1}{\mu}\left[\lambda^2+2\lambda\mu+\alpha^2+\alpha\mu(n+1)\right]}$$

$$\Longrightarrow A_{1,1}=0. \tag{4.9}$$

Thus, we have from equations (4.5), (4.7) and (4.9)

$$\pi_{1,j}=\frac{\rho^n}{(n+1)!}\cdot\pi_{0,0}\cdot\gamma\cdot\beta^j \qquad \text{for } j>n \tag{4.10}$$

Remarque:

$$\pi_{1,j} = \frac{\gamma}{(n+1)} \cdot \pi_{0,j} \quad \text{and} \quad \pi_{0,j} = \frac{\rho^n}{(n)!} \cdot \pi_{0,0} \cdot \beta^j$$

Note that the above equation is valid for $j = n+1$ since $\beta \cdot \gamma = \rho$

$$\pi_{1,n+1} = \frac{\rho^{n+1}}{(n+1)!} \cdot \pi_{0,0} \cdot \beta^n$$

For row i, $3 \leq i \leq k$, we again use difference equations, and derive the values of $A_{1,1}$ and $A_{1,2}$ as above. We find that $A_{1,1} = 0$ and $A_{1,2} = \frac{\pi_{0,0} \cdot \gamma^i}{i!}$. Thus:

$$\begin{aligned} \pi_{2,j} &= \frac{\pi_{0,0} \cdot \gamma^2}{2!} \cdot \frac{\rho^n}{(n+1)!} \cdot \beta^j \quad \text{for } j \geq n+1 \\ &\vdots \end{aligned} \tag{4.11}$$

$$\pi_{k-1,j} = \frac{\pi_{0,0} \cdot \gamma^{k-1}}{(k-1)!} \cdot \frac{\rho^n}{(n+1)!} \cdot \beta^j \quad \text{for } j \geq n+k-2 \tag{4.12}$$

In general

$$\pi_{i,j} = \frac{\pi_{0,0} \cdot \gamma^i}{i!} \cdot \frac{\rho^n}{(n+1)!} \cdot \beta^j \quad \text{for } j \geq n+i-1 \text{ and } 0 \leq i < k \tag{4.13}$$

For row (k+1), The relevant balance equations for the $(k+1)^{th}$ row are given by:

$$(\lambda + k\mu) \cdot \pi_{k,j} = \lambda \cdot \pi_{k,j-1} + \alpha \cdot \pi_{k-1,j} + k\mu \cdot \pi_{k,j+1} \quad \text{for } j > k \tag{4.14}$$

As before, the solution for such equations is given by:

$$\pi_{k,j} = A_{k,1} x_k^j + A_{k,2} \beta^j \quad \text{for } j > k \tag{4.15}$$

where x_k is a solution of the homogeneous equation:

$$x_k \cdot (\lambda + k\mu) = \lambda + x_k^2 \cdot k\mu \tag{4.16}$$

Solving Equation (4.16) for x_k, we find $x_k = k \cdot \rho$ (the other solution $x_k = 1$ is trivially discarded). Thus, we have:

$$\pi_{k,j} = A_{k,1} \left(\frac{\rho}{k}\right)^j + A_{k,2} \beta^j \quad \text{for } j > k \tag{4.17}$$

We now find $A_{k,1}$ and $A_{k,2}$ as we did for 2nd row and find that, for $j \geq k$:
Plugging in Equations (4.12) and (4.15) into (4.14)

$$(\lambda + k\mu) \cdot \left(A_{k,1} x_k^j + A_{k,2} \beta^j\right) = \lambda \cdot \left(A_{k,1} x_k^{j-1} + A_{k,2} \beta^{j-1}\right) + \alpha \cdot \left(\frac{\pi_{0,0} \cdot \gamma^{k-1}}{(k-1)!} \cdot \frac{\rho^n}{(n+1)!} \cdot \beta^j\right) + k\mu \cdot \left(A_{k,1} x_k^{j+1} + A_{k,2} \beta^{j+1}\right)$$

$$(\lambda + k\mu) \cdot A_{k,1} x_k^j + (\lambda + k\mu) \cdot A_{k,2} \beta^j = \lambda \cdot A_{k,1} x_k^{j-1} + \lambda \cdot A_{k,2} \beta^{j-1} + \alpha \cdot \left(\frac{\pi_{0,0} \cdot \gamma^{k-1}}{(k-1)!} \cdot \frac{\rho^n}{(n+1)!} \cdot \beta^j\right) + k\mu \cdot A_{k,1} x_k^{j+1} + k\mu \cdot A_{k,2} \beta^{j+1}$$

$$(4.16) \Longrightarrow A_{k,2} \cdot \left(\beta(\lambda + k\mu) - \lambda - k\mu\beta^2\right) = \alpha \cdot \left(\frac{\pi_{0,0} \cdot \gamma^{k-1}}{(k-1)!} \cdot \frac{\rho^n}{(n+1)!} \cdot \beta\right)$$

$$\Longrightarrow A_{k,2} \cdot \beta \left(\lambda + k\mu - \lambda - k\mu\frac{\lambda}{(\lambda+\alpha)}\right) = \alpha \cdot \left(\frac{\pi_{0,0} \cdot \gamma^{k-1}}{(k-1)!} \cdot \frac{\rho^n}{(n+1)!} \cdot \beta\right)$$

$$\Longrightarrow A_{k,2} \cdot \tfrac{\beta}{(\lambda+\alpha)} \left(\cancel{\lambda^2} + \cancel{\lambda\alpha} + \cancel{k\mu\lambda} + k\mu\alpha - \cancel{\lambda^2} - \cancel{2}\lambda\alpha - \alpha^2 - \cancel{k\mu\lambda}\right) = \alpha \cdot \left(\tfrac{\pi_{0,0} \cdot \gamma^{k-1}}{(k-1)!} \cdot \tfrac{\rho^n}{(n+1)!} \cdot \beta\right)$$

$$\Longrightarrow A_{k,2} \cdot \frac{\cancel{\beta} \cdot \cancel{\alpha}}{(\lambda+\alpha)} (k\mu - (\lambda+\alpha)) = \cancel{\alpha} \cdot \left(\frac{\pi_{0,0} \cdot \gamma^{k-1}}{(k-1)!} \cdot \frac{\rho^n}{(n+1)!} \cdot \cancel{\beta}\right)$$

$$\Longrightarrow A_{k,2} = \frac{\pi_{0,0} \cdot \gamma^{k-1}}{(k-1)!} \cdot \frac{\rho^n}{(n+1)!} \cdot \frac{(\lambda+\alpha)}{(k\mu - (\lambda+\alpha))}$$

Since $\gamma = \frac{\lambda+\alpha}{\mu}$

$$\Longrightarrow A_{k,2} = \frac{\pi_{0,0} \cdot \mu\gamma^k}{(k-1)!\,(k\mu - (\lambda+\alpha))} \cdot \frac{\rho^n}{(n+1)!} \tag{4.18}$$

As before to get $A_{k,1}$, we need to evaluate $\pi_{k,k}$, this requires us to use the balance equation for $\pi_{k-1,k-1}$, after a few steps of algebra, we find :

$$\pi_{k,k} = \frac{\pi_{0,0} \cdot \rho^k}{k!} \cdot \frac{\rho^n}{(n+1)!} \tag{4.19}$$

Now, we use equations (4.12) and (4.17) into (4.14) for $j = k+1$

$$(\lambda + k\mu) \cdot \left(A_{k,1}x_k^{k+1} + A_{k,2}\beta^{k+1}\right) = \lambda \cdot \pi_{k,k} + \alpha \cdot \left(\tfrac{\pi_{0,0} \cdot \gamma^{k-1}}{(k-1)!} \cdot \tfrac{\rho^n}{(n+1)!} \cdot \beta^{k+1}\right) + k\mu \cdot \left(A_{k,1}x_k^{k+2} + A_{k,2}\beta^{k+2}\right)$$

$$(4.19) \Longrightarrow (\lambda + k\mu) \cdot A_{k,1}x_k^{k+1} + (\lambda + k\mu) \cdot A_{k,2}\beta^{k+1} = \lambda \cdot \tfrac{\pi_{0,0} \cdot \rho^k}{k!} \cdot \tfrac{\rho^n}{(n+1)!} + \alpha \cdot \left(\tfrac{\pi_{0,0} \cdot \gamma^{k-1}}{(k-1)!} \cdot \tfrac{\rho^n}{(n+1)!} \cdot \beta^{k+1}\right) + k\mu \cdot A_{k,1}x_k^{k+2} + k\mu \cdot A_{k,2}\beta^{k+2}$$

$$\Longrightarrow A_{k,1}(\lambda + k\mu - k\mu x_k) \cdot x_k^{k+1} = \lambda \cdot \tfrac{\pi_{0,0} \cdot \rho^k}{k!} \cdot \tfrac{\rho^n}{(n+1)!} + \alpha \cdot \left(\tfrac{\pi_{0,0} \cdot \gamma^{k-1}}{(k-1)!} \cdot \tfrac{\rho^n}{(n+1)!} \cdot \beta^{k+1}\right) + A_{k,2} \cdot \beta^{k+1}(k\mu\beta - \lambda - k\mu)$$

Since $x_k = \frac{\rho}{k}$ and $\beta \cdot \gamma = \rho$

$$\Longrightarrow A_{k,1} \cdot k\mu \cdot x_k^{k+1} = \lambda \cdot \tfrac{\pi_{0,0} \cdot \rho^k}{k!} \cdot \tfrac{\rho^n}{(n+1)!} + \alpha \cdot \left(\tfrac{\pi_{0,0} \cdot \rho^{k+1}}{(k-1)!} \cdot \tfrac{\rho^n}{(n+1)!} \cdot \beta^2\right) - A_{k,2} \cdot \beta^{k+1}\left(\tfrac{k\mu\alpha}{\lambda+\alpha} + \lambda\right)$$

$$\Longrightarrow A_{k,1} \cdot k\mu \cdot x_k^{k+1} = \tfrac{\pi_{0,0} \cdot \rho^{k-1}}{(k-1)!} \cdot \left(\tfrac{\lambda^2\rho^n}{k\mu(n+1)!} + \tfrac{\alpha\lambda^2}{(\lambda+\alpha)^2} \cdot \tfrac{\rho^n}{(n+1)!}\right) - A_{k,2} \cdot \beta^{k+1}\left(\tfrac{k\mu\alpha}{\lambda+\alpha} + \lambda\right)$$

$$(4.18) \Longrightarrow A_{k,1} \cdot k\mu \cdot x_k^{k+1} = \frac{\pi_{0,0} \cdot \rho^{k-1}}{(k-1)!} \cdot \left(\frac{\lambda^2 \rho^n}{k\mu(n+1)!} + \frac{\alpha\lambda^2}{(\lambda+\alpha)^2} \cdot \frac{\rho^n}{(n+1)!} \right) - \frac{\pi_{0,0} \cdot \mu\gamma^k}{(k-1)!(k\mu-(\lambda+\alpha))} \cdot \frac{\rho^n}{(n+1)!} \left(\frac{k\mu\alpha}{\lambda+\alpha} + \lambda \right) \beta^{k+1}$$

Since $\beta \cdot \gamma = \rho$

$$\Longrightarrow A_{k,1} \cdot k\mu \cdot x_k^{k+1} = \frac{\pi_{0,0} \cdot \rho^{k-1}\lambda^2}{(k-1)!} \cdot \left(\frac{\rho^n}{k\mu(n+1)!} + \frac{\alpha\rho^n}{(\lambda+\alpha)^2(n+1)!} \right) - \frac{\pi_{0,0} \cdot \rho^{k-1}\mu}{(k-1)!} \cdot \frac{\beta\rho \cdot \rho^n}{(n+1)!} \left(\frac{k\mu\alpha+\lambda(\lambda+\alpha)}{((k\mu-(\lambda+\alpha))(\lambda+\alpha))} \right)$$

Since $\beta\rho\mu = \frac{\lambda^2}{\lambda+\alpha}$

$$\Longrightarrow A_{k,1} \cdot k\mu \cdot x_k^{k+1} = \frac{\pi_{0,0} \cdot \rho^{k-1}\lambda^2}{(k-1)!} \cdot \left(\frac{\rho^n}{k\mu(n+1)!} + \frac{\alpha\rho^n}{(\lambda+\alpha)^2(n+1)!} \right) - \frac{\pi_{0,0} \cdot \rho^{k-1}}{(k-1)!} \cdot \frac{\lambda^2 \cdot \rho^n}{(n+1)!} \left(\frac{k\mu\alpha+\lambda(\lambda+\alpha)}{\left((k\mu-(\lambda+\alpha))(\lambda+\alpha)^2\right)} \right)$$

$$\Longrightarrow A_{k,1} \cdot k\mu \cdot x_k^{k+1} = \frac{\pi_{0,0} \cdot \rho^{k-1}\lambda^2}{(k-1)!} \cdot \left(\frac{\rho^n}{k\mu(n+1)!} + \frac{\alpha\rho^n}{(\lambda+\alpha)^2(n+1)!} - \frac{\rho^n}{(n+1)!} \frac{k\mu\alpha+\lambda(\lambda+\alpha)}{\left(k\mu-(\lambda+\alpha)\right)(\lambda+\alpha)^2)} \right)$$

$$\Longrightarrow A_{k,1} \cdot k\mu \cdot x_k^{k+1} = \frac{\pi_{0,0} \cdot \rho^{k-1}\lambda^2}{(k-1)!} \cdot \frac{\rho^n}{(n+1)!} \cdot \left(\frac{(k\mu-(\lambda+\alpha))(\lambda+\alpha)^2+k\alpha\mu(k\mu-(\lambda+\alpha))-k\mu(k\mu\alpha+\lambda(\lambda+\alpha))}{k\mu\left(k\mu-(\lambda+\alpha))(\lambda+\alpha)^2\right)} \right)$$

$$\Longrightarrow A_{k,1} \cdot k\mu \cdot x_k^{k+1} = \frac{\pi_{0,0} \cdot \rho^{k-1}\lambda^2}{(k-1)!} \cdot \frac{\rho^n}{(n+1)!} \cdot \left(\frac{(k\mu-(\lambda+\alpha))(\lambda+\alpha)^2 - k\mu(\lambda+\alpha)^2}{k\mu\,(k\mu-(\lambda+\alpha))(\lambda+\alpha)^2)} \right)$$

$$\Longrightarrow A_{k,1} \cdot k\mu \cdot x_k^{k+1} = \frac{\pi_{0,0} \cdot \rho^{k-1}\lambda^2}{(k-1)!} \cdot \frac{\rho^n}{(n+1)!} \cdot \left(\frac{-(\lambda+\alpha)^3}{k\mu\,(k\mu-(\lambda+\alpha))(\lambda+\alpha)^2)} \right)$$

$$\Longrightarrow A_{k,1} \cdot k\mu \cdot \frac{\rho^{k+1}}{k^{k+1}} = \frac{\pi_{0,0} \cdot \rho^{k+1}}{k!} \cdot \mu \cdot \frac{\rho^n}{(n+1)!} \cdot \left(\frac{-(\lambda+\alpha)}{(k\mu-(\lambda+\alpha)} \right)$$

$$\Longrightarrow A_{k,1} = -\frac{\rho^n}{(n+1)!} \cdot \left(\frac{\pi_{0,0} k^{k-1}(\lambda+\alpha)}{(k-1)!(k\mu-(\lambda+\alpha))} \right) \tag{4.20}$$

Thus, we have from equations (4.17),(4.18) and (4.20)

$$\pi_{k,j} = \frac{\rho^n}{(n+1)!} \frac{\pi_{0,0}\gamma^k k\mu\beta^j}{k! \cdot (k\mu-(\lambda+\alpha))} - \frac{\rho^n}{(n+1)!} \frac{\pi_{0,0} k^k(\lambda+\alpha)\left(\frac{\rho}{k}\right)^j}{k! \cdot (k\mu-(\lambda+\alpha))} \quad \text{for } j > k-1$$

$$\pi_{k,j} = \frac{\rho^n}{(n+1)!} \left[\frac{\pi_{0,0}\gamma^k k\mu\beta^j}{k! \cdot (k\mu-(\lambda+\alpha))} - \frac{\pi_{0,0} k^k(\lambda+\alpha)\left(\frac{\rho}{k}\right)^j}{k! \cdot (k\mu-(\lambda+\alpha))} \right] \quad \text{for } j > k-1 \tag{4.21}$$

Now we need to find the $\pi_{0,0}$ to solve the limiting probabilities of all the states of the Markov chain, We use the equation $\sum_{i,j} \pi_{i,j} = 1$

$$\sum_{i,j} \pi_{i,j} = 1$$

$$\Longrightarrow 1 = \sum_{\substack{i,j \\ i<k}} \pi_{i,j} + \sum_{j} \pi_{k,j}$$

$$\Longrightarrow 1 = \sum_{\substack{0,j \\ j\leq n}} \pi_{0,j} + \sum_{\substack{i,j \\ i<k \\ j>n}} \pi_{i,j} + \sum_{j} \pi_{k,j}$$

$$\Longrightarrow 1 = \sum_{\substack{0,j \\ j\leq n}} \frac{\pi_{0,0} \cdot \rho^j}{j!} + \sum_{\substack{i,j \\ i<k \\ j>n}} \frac{\pi_{0,0} \cdot \gamma^i}{i!} \cdot \frac{\rho^n}{(n+1)!} \cdot \beta^j + \sum_{j} \frac{\rho^n}{(n+1)!} \left[\frac{\pi_{0,0}\gamma^k k\mu\beta^j}{k! \cdot (k\mu - (\lambda+\alpha))} - \frac{\pi_{0,0}k^k(\lambda+\alpha)\left(\frac{\rho}{k}\right)^j}{k! \cdot (k\mu - (\lambda+\alpha))} \right]$$

$$\Longrightarrow 1 = \pi_{0,0} \cdot \sum_{\substack{0,j \\ j\leq n}} \frac{\rho^j}{j!} + \pi_{0,0}\frac{\rho^n}{(n+1)!} \cdot \sum_{\substack{i,j \\ i<k \\ j>n}} \frac{\gamma^i}{i!} \cdot \beta^j + \pi_{0,0}\frac{\rho^n}{(n+1)!} \sum_{j} \left[\frac{\gamma^k k\mu\beta^j}{k! \cdot (k\mu - (\lambda+\alpha))} - \frac{k^k(\lambda+\alpha)\left(\frac{\rho}{k}\right)^j}{k! \cdot (k\mu - (\lambda+\alpha))} \right]$$

$$\Longrightarrow \pi_{0,0}^{-1} = \sum_{j=0}^{n} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!} \left[\sum_{0<i<k} \frac{\gamma^i}{i!} \cdot \sum_{i\leq j} \beta^j + \frac{\gamma^k k\mu}{k! \cdot (k\mu - (\lambda+\alpha))} \cdot \sum_{k\leq j} \beta^j - \frac{k^k(\lambda+\alpha)}{k! \cdot (k\mu - (\lambda+\alpha))} \cdot \sum_{k\leq j} \left(\frac{\rho}{k}\right)^j \right]$$

$$\Longrightarrow \pi_{0,0}^{-1} = \sum_{j=0}^{n} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!} \left[\sum_{0<i<k} \frac{\gamma^i}{i!} \cdot \frac{\beta^i}{1-\beta} + \frac{\gamma^k k\mu}{k! \cdot (k\mu - (\lambda+\alpha))} \cdot \frac{\beta^k}{1-\beta} - \frac{(\lambda+\alpha)}{k! \cdot (k\mu - (\lambda+\alpha))} \cdot \frac{k\mu\rho^k}{k\mu - \lambda} \right]$$

Since $\gamma \cdot \beta = \rho$

$$\Longrightarrow \pi_{0,0}^{-1} = \sum_{j=0}^{n} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!} \left[\frac{1}{1-\beta} \sum_{0<i<k} \frac{\rho^i}{i!} + \frac{\rho^k\mu}{(k-1)! \cdot (k\mu - (\lambda+\alpha))} \cdot \frac{1}{1-\beta} - \frac{(\lambda+\alpha)}{(k-1)! \cdot (k\mu - (\lambda+\alpha))} \cdot \frac{\mu\rho^k}{k\mu - \lambda} \right]$$

Since $\lambda + \alpha = \frac{\alpha}{1-\beta}$

$$\Longrightarrow \pi_{0,0}^{-1} = \sum_{j=0}^{n} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!} \left[\frac{1}{1-\beta} \sum_{0<i<k} \frac{\rho^i}{i!} + \frac{\rho^k\mu}{(1-\beta)(k-1)! \cdot (k\mu - (\lambda+\alpha))} - \frac{\rho^k\mu\alpha}{(1-\beta)(k-1)! \cdot (k\mu - (\lambda+\alpha)) \cdot (k\mu - \lambda)} \right]$$

$$\Longrightarrow \pi_{0,0}^{-1} = \sum_{j=0}^{n} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!} \left[\frac{1}{1-\beta} \sum_{0<i<k} \frac{\rho^i}{i!} + \frac{\rho^k\mu}{(1-\beta)(k-1)! \cdot (k\mu - (\lambda+\alpha))} \cdot \left(1 - \frac{\alpha}{k\mu - \lambda}\right) \right]$$

$$\Longrightarrow \pi_{0,0}^{-1} = \sum_{j=0}^{n} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!} \left[\frac{1}{1-\beta} \sum_{0<i<k} \frac{\rho^i}{i!} + \frac{\rho^k\mu}{(1-\beta)(k-1)! \cdot (k\mu - \lambda)} \right]$$

$$\Longrightarrow \pi_{0,0}^{-1} = \sum_{j=0}^{n} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!} \cdot \frac{1}{1-\beta} \left[\sum_{0<i<k} \frac{\rho^i}{i!} + \frac{\rho^k\mu}{(k-1)! \cdot (k\mu - \lambda)} \right]$$

$$\Longrightarrow \pi_{0,0} = \left[\sum_{j=0}^{n} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!} \cdot \frac{1}{(1-\beta)} \cdot \left[\sum_{0\leq i<k} \frac{\rho^i}{i!} + \frac{\rho^k\mu}{(k-1)! \cdot (k\mu - \lambda)} \right] \right]^{-1} \tag{4.22}$$

■

4.5 Performance Measures

Theorem 4.2. *The Mean number of jobs* $\mathbb{E}[N]$ *:*

$$\mathbb{E}[N] = \pi_{0,0}\rho \left[\sum_{j=0}^{n-1} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)! \cdot (1-\beta)} \sum_{j=0}^{k-1} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)! \cdot (1-\beta)^2} \sum_{j=0}^{k} \frac{\rho^j}{j!} + \frac{\rho^{n+k-1}k\mu}{(n+1)!k!} \cdot \frac{(k-1)(1-\beta)(k\mu-\lambda) + k\mu - \lambda\beta}{(1-\beta)^2(k\mu-\lambda)^2} \right]$$

Proof. After we get $\pi_{i,j}$, $\forall i, j$, We will proceed to calculate the mean number of jobs $E[N]$ in the M/M/K/Setup,

$$\mathbb{E}\left[N^{M/M/k/\text{setup}}\right] = \sum_{\substack{0\le i\le k\\ i\le j}} j\cdot\pi_{i,j}$$

$$\implies \mathbb{E}[N] = \sum_{j=0}^{n} j\cdot\pi_{0,j} + \sum_{0\le i<k}\left(\sum_{\substack{i\le j\\ j>n}} j\cdot\pi_{i,j}\right) + \sum_{\substack{k\le j\\ j>n}} j\cdot\pi_{k,j}$$

From equations (4.13), (4.14) and (4.21)

$$\begin{aligned}
\implies \mathbb{E}[N] &= \sum_{j=0}^{n} j\cdot\pi_{0,j} + \sum_{0\le i<k}\left(\sum_{\substack{i\le j\\ j>n}} j\cdot\frac{\pi_{0,0}\cdot\gamma^i}{i!}\cdot\frac{\rho^n}{(n+1)!}\cdot\beta^j\right) + \sum_{\substack{k\le j\\ j>n}} j\cdot\frac{\rho^n}{(n+1)!}\left[\frac{\pi_{0,0}\gamma^k k\mu\beta^j}{k!\cdot(k\mu-(\lambda+\alpha))} - \frac{\pi_{0,0}k^k(\lambda+\alpha)\left(\frac{\rho}{k}\right)^j}{k!\cdot(k\mu-(\lambda+\alpha))}\right]\\
&= \sum_{j=0}^{n} j\cdot\pi_{0,j} + \frac{\rho^n}{(n+1)!}\left(\sum_{0\le i<k}\frac{\pi_{0,0}\cdot\gamma^i}{i!}\cdot\left(\sum_{i\le j} j\beta^j\right) + \sum_{k\le j} j\left[\frac{\pi_{0,0}\gamma^k k\mu\beta^j}{k!\cdot(k\mu-(\lambda+\alpha))} - \frac{\pi_{0,0}k^k(\lambda+\alpha)\left(\frac{\rho}{k}\right)^j}{k!\cdot(k\mu-(\lambda+\alpha))}\right]\right)\\
&= \sum_{j=0}^{n} j\cdot\pi_{0,0}\frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!}\sum_{0\le i<k}\frac{\pi_{0,0}\cdot\gamma^i}{i!}\cdot\sum_{i\le j} j\beta^j + \frac{\rho^n}{(n+1)!}\frac{\pi_{0,0}\gamma^k k\mu}{k!\cdot(k\mu-(\lambda+\alpha))}\cdot\sum_{k\le j} j\beta^j - \frac{\rho^n}{(n+1)!}\frac{\pi_{0,0}k^k(\lambda+\alpha)}{k!\cdot(k\mu-(\lambda+\alpha))}\cdot\sum_{k\le j} j\cdot\left(\frac{\rho}{k}\right)^j
\end{aligned} \tag{4.23}$$

Now we find a closed form expression for $\sum_{i\le j} j\beta^j$ which will be useful for simplifying Eq (4.23).

$$\begin{aligned}
Z &= i\cdot x^i + (i+1)\cdot x^{i+1} + (i+2)\cdot x^{i+2} + (i+3)\cdot x^{i+3} + \dots\\
Z\cdot x &= 0\cdot x^i + (i+0)\cdot x^{i+1} + (i+1)\cdot x^{i+2} + (i+2)\cdot x^{i+3} + \dots
\end{aligned}$$

$$\begin{aligned}
Z\cdot(1-x) &= i\cdot x^i + 1\cdot x^{i+1} + 1\cdot x^{i+2} + 1\cdot x^{i+3} + \dots\\
\implies Z &= \frac{(i-1)\cdot x^i}{1-x} + \frac{x^i}{(1-x)^2}
\end{aligned}$$

So

$$\implies \sum_{i\le j} j\cdot x^j = \frac{(i-1)\cdot x^i}{1-x} + \frac{x^i}{(1-x)^2} \tag{4.24}$$

Using the equations (4.23) and (4.24).

$$\begin{aligned}
\mathbb{E}[N] &= \sum_{j=0}^{n} j\cdot\pi_{0,0}\frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!}\sum_{0\le i<k}\frac{\pi_{0,0}\cdot\gamma^i\beta^i}{i!}\cdot\left(\frac{i-1}{1-\beta} + \frac{1}{(1-\beta)^2}\right)\\
&\quad + \frac{\rho^n}{(n+1)!}\frac{\pi_{0,0}\gamma^k k\mu\beta^k}{k!\cdot(k\mu-(\lambda+\alpha))}\cdot\left(\frac{k-1}{1-\beta} + \frac{1}{(1-\beta)^2}\right) - \frac{\rho^n}{(n+1)!}\frac{\pi_{0,0}\cancel{k^k}(\lambda+\alpha)}{k!\cdot(k\mu-(\lambda+\alpha))}\cdot\frac{\rho^k}{\cancel{k^k}}\cdot\left(\frac{k\mu(k-1)}{(k\mu-\lambda)} + \frac{k^2\mu^2}{(k\mu-\lambda)^2}\right)
\end{aligned}$$

$$\begin{aligned}
&= \sum_{j=0}^{n} j\cdot\pi_{0,0}\frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!}\left[\sum_{0\le i<k}\frac{\pi_{0,0}\cdot\rho^i}{i!}\cdot\left(\frac{i-1}{1-\beta} + \frac{1}{(1-\beta)^2}\right) + \frac{\pi_{0,0}\rho^k k\mu}{k!\cdot(k\mu-(\lambda+\alpha))}\cdot\left(\frac{k-1}{1-\beta} + \frac{1}{(1-\beta)^2}\right)\right.\\
&\qquad \left. - \frac{\pi_{0,0}\rho^k(\lambda+\alpha)}{k!\cdot(k\mu-(\lambda+\alpha))}\cdot\left(\frac{k\mu(k-1)}{(k\mu-\lambda)} + \frac{k^2\mu^2}{(k\mu-\lambda)^2}\right)\right]
\end{aligned}$$

$$= \sum_{j=0}^{n} j \cdot \pi_{0,0} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!} \Bigg[\sum_{0 \le i < k} \frac{\pi_{0,0} \cdot \rho^i}{i!} \cdot \left(\frac{i-1}{1-\beta} + \frac{1}{(1-\beta)^2} \right) + \frac{\pi_{0,0} \rho^k (k\mu)}{k! \cdot (k\mu - (\lambda+\alpha))} \cdot \left(\left(\frac{k-1}{1-\beta} + \frac{1}{(1-\beta)^2} \right) - (\lambda+\alpha) \cdot \left(\frac{(k-1)}{(k\mu-\lambda)} + \frac{k\mu}{(k\mu-\lambda)^2} \right) \right) \Bigg]$$

$$= \sum_{j=0}^{n} j \cdot \pi_{0,0} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!} \sum_{0 \le i < k} \frac{\pi_{0,0} \cdot \rho^i}{i!} \cdot \left(\frac{i}{1-\beta} + \frac{\beta}{(1-\beta)^2} \right) + \frac{\rho^n}{(n+1)!} \cdot \frac{\pi_{0,0} \rho^k (k\mu)}{k! \cdot (k\mu - (\lambda+\alpha))} \cdot \left(\left(\frac{k-1}{1-\beta} + \frac{1}{(1-\beta)^2} \right) - (\lambda+\alpha) \cdot \left(\frac{(k-1)}{(k\mu-\lambda)} + \frac{k\mu}{(k\mu-\lambda)^2} \right) \right) \Bigg]$$

$$= \sum_{j=0}^{n} j \cdot \pi_{0,0} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!} \sum_{0 \le i < k} \frac{\pi_{0,0} \cdot \rho^i}{(i-1)!} \cdot \frac{1}{1-\beta} + \frac{\rho^n}{(n+1)!} \sum_{0 \le i < k} \frac{\pi_{0,0} \cdot \rho^i}{i!} \cdot \frac{1}{(1-\beta)^2} + \frac{\pi_{0,0} \rho^k (k\mu)}{k! \cdot (k\mu - (\lambda+\alpha))} \cdot \left(\left(\frac{k-1}{1-\beta} + \frac{1}{(1-\beta)^2} \right) - (\lambda+\alpha) \cdot \left(\frac{(k-1)}{(k\mu-\lambda)} + \frac{k\mu}{(k\mu-\lambda)^2} \right) \right) \Bigg]$$

$$= \frac{\pi_{0,0} \rho^k (k\mu)}{k! \cdot (k\mu - (\lambda+\alpha))} \cdot \left(\left(\frac{k-1}{1-\beta} + \frac{1}{(1-\beta)^2} \right) - (\lambda+\alpha) \cdot \left(\frac{(k-1)}{(k\mu-\lambda)} + \frac{k\mu}{(k\mu-\lambda)^2} \right) \right) \Bigg] + \sum_{j=0}^{n} j \cdot \pi_{0,0} \frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!} \frac{\pi_{0,0}}{(1-\beta)} \sum_{0 \le i < k} \frac{\rho^i}{(i-1)!} + \frac{\rho^n}{(n+1)!} \frac{\pi_{0,0}}{(1-\beta)^2} \sum_{0 \le i < k} \frac{\rho^i}{i!}$$

$$= \overbrace{\sum_{j=0}^{n} j \cdot \pi_{0,0} \frac{\rho^j}{j!} + \frac{\rho^{n+1} \pi_{0,0}}{(n+1)!(1-\beta)} \sum_{0 \le i < k} \frac{\rho^{i-1}}{(i-1)!} + \frac{\rho^n}{(n+1)!} \frac{\pi_{0,0}}{(1-\beta)^2} \sum_{0 \le i < k} \frac{\rho^i}{i!}}^{(\star)}$$
$$\underbrace{+ \frac{\pi_{0,0} \rho^k (k\mu)}{k! \cdot (k\mu - (\lambda+\alpha))} \cdot \left(\left(\frac{k-1}{1-\beta} + \frac{1}{(1-\beta)^2} \right) - (\lambda+\alpha) \cdot \left(\frac{(k-1)}{(k\mu-\lambda)} + \frac{k\mu}{(k\mu-\lambda)^2} \right) \right) \Bigg]}_{(\star\star)} <$$

We start to develop $(\star\star)$

$$\Longrightarrow \frac{\rho^{n+k} \pi_{0,0} k\mu(\lambda+\alpha)}{(n+1)!k! \cdot (k\mu - (\lambda+\alpha))} \cdot \left(\frac{k-1}{(\lambda+\alpha)(1-\beta)} + \frac{1}{(\lambda+\alpha)(1-\beta)^2} - \frac{k-1}{(k\mu-\lambda)} + \frac{k\mu}{(k\mu-\lambda)^2} \right)$$

$$\Longrightarrow \frac{\rho^{n+k} \pi_{0,0} k\mu(\lambda+\alpha)}{(n+1)!k! \cdot (k\mu - (\lambda+\alpha))} \cdot \left(\frac{(k-1)(k\mu-\lambda) - (k-1)(\lambda+\alpha)(1-\beta)}{(\lambda+\alpha)(1-\beta)(k\mu-\lambda)} + \frac{(k\mu-\lambda)^2 - k\mu(\lambda+\alpha)(1-\beta)^2}{(\lambda+\alpha)(1-\beta)^2(k\mu-\lambda)^2} \right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu(\lambda+\alpha)}{(n+1)!k!\cdot(k\mu-(\lambda+\alpha))}\cdot\left(\frac{(k-1)((k\mu-\lambda)-(\lambda+\alpha)(1-\beta))}{(\lambda+\alpha)(1-\beta)(k\mu-\lambda)}+\frac{k^2\mu^2+\lambda^2-2k\mu\lambda-k\mu\alpha(1+\beta^2-2\beta)-k\mu\lambda(1+\beta^2-2\beta)}{(\lambda+\alpha)(1-\beta)^2(k\mu-\lambda)^2}\right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu(\lambda+\alpha)}{(n+1)!k!\cdot(k\mu-(\lambda+\alpha))}\cdot\left(\frac{(k-1)\left(k\mu-\cancel{\lambda}-(\lambda+\alpha)+\overbrace{(\lambda+\alpha)\beta}^{=\lambda}\right)}{(\lambda+\alpha)(1-\beta)(k\mu-\lambda)}+\frac{k^2\mu^2+\lambda^2-2k\mu\lambda-k\mu\alpha-k\mu\alpha\beta^2+2k\mu\alpha\beta-k\mu\lambda-k\mu\lambda\beta^2+2k\mu\lambda\beta}{(\lambda+\alpha)(1-\beta)^2(k\mu-\lambda)^2}\right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu(\lambda+\alpha)}{(n+1)!k!\cdot(k\mu-(\lambda+\alpha))}\cdot\left(\frac{(k-1)\Big(k\mu-(\lambda+\alpha)\Big)}{(\lambda+\alpha)(1-\beta)(k\mu-\lambda)}+\frac{k\mu\Big[k\mu-(\lambda+\alpha)\Big]-2k\mu\lambda-\beta^2k\mu(\lambda+\alpha)+2k\mu\beta(\lambda+\alpha)+\lambda^2}{(\lambda+\alpha)(1-\beta)^2(k\mu-\lambda)^2}\right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu(\lambda+\alpha)}{(n+1)!k!\cdot(k\mu-(\lambda+\alpha))}\cdot\left(\frac{(k-1)\Big(k\mu-(\lambda+\alpha)\Big)}{(\lambda+\alpha)(1-\beta)(k\mu-\lambda)}+\frac{k\mu\Big[k\mu-(\lambda+\alpha)\Big]-2k\mu\lambda-\beta\cdot\overbrace{\beta(\lambda+\alpha)}^{=\lambda}k\mu+2k\mu\overbrace{\beta(\lambda+\alpha)}^{=\lambda}+\lambda^2}{(\lambda+\alpha)(1-\beta)^2(k\mu-\lambda)^2}\right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu(\lambda+\alpha)}{(n+1)!k!\cdot(k\mu-(\lambda+\alpha))}\cdot\left(\frac{(k-1)\Big(k\mu-(\lambda+\alpha)\Big)}{(\lambda+\alpha)(1-\beta)(k\mu-\lambda)}+\frac{k\mu\Big[k\mu-(\lambda+\alpha)\Big]-\cancel{2k\mu\lambda}-\beta k\mu\lambda+\cancel{2k\mu\lambda}+\lambda^2}{(\lambda+\alpha)(1-\beta)^2(k\mu-\lambda)^2}\right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu(\lambda+\alpha)}{(n+1)!k!\cdot(k\mu-(\lambda+\alpha))}\cdot\left(\frac{(k-1)\Big(k\mu-(\lambda+\alpha)\Big)}{(\lambda+\alpha)(1-\beta)(k\mu-\lambda)}+\frac{k\mu\Big[k\mu-(\lambda+\alpha)\Big]-\cancel{2k\mu\lambda}-\beta k\mu\lambda+\cancel{2k\mu\lambda}+\overbrace{\lambda^2}^{=\lambda\beta(\lambda+\alpha)}}{(\lambda+\alpha)(1-\beta)^2(k\mu-\lambda)^2}\right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu(\lambda+\alpha)}{(n+1)!k!\cdot(k\mu-(\lambda+\alpha))}\cdot\left(\frac{(k-1)\Big(k\mu-(\lambda+\alpha)\Big)}{(\lambda+\alpha)(1-\beta)(k\mu-\lambda)}+\frac{k\mu\Big[k\mu-(\lambda+\alpha)\Big]-\beta k\mu\lambda+\lambda\beta(\lambda+\alpha)}{(\lambda+\alpha)(1-\beta)^2(k\mu-\lambda)^2}\right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu(\lambda+\alpha)}{(n+1)!k!\cdot(k\mu-(\lambda+\alpha))}\cdot\left(\frac{(k-1)\Big(k\mu-(\lambda+\alpha)\Big)}{(\lambda+\alpha)(1-\beta)(k\mu-\lambda)}+\frac{k\mu\Big[k\mu-(\lambda+\alpha)\Big]-\lambda\beta\Big[k\mu-(\lambda+\alpha)\Big]}{(\lambda+\alpha)(1-\beta)^2(k\mu-\lambda)^2}\right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu(\lambda+\alpha)}{(n+1)!k!\cdot(k\mu-(\lambda+\alpha))}\cdot\left(\frac{(k-1)\Big(k\mu-(\lambda+\alpha)\Big)}{(\lambda+\alpha)(1-\beta)(k\mu-\lambda)}+\frac{\Big(k\mu-(\lambda+\alpha)\Big)\Big(k\mu-\lambda\beta\Big)}{(\lambda+\alpha)(1-\beta)^2(k\mu-\lambda)^2}\right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu(\lambda+\alpha)\cancel{\left(k\mu-(\lambda+\alpha)\right)}}{(n+1)!k!\cdot\cancel{\left(k\mu-(\lambda+\alpha)\right)}}\cdot\left(\frac{(k-1)}{(\lambda+\alpha)(1-\beta)(k\mu-\lambda)}+\frac{\left(k\mu-\lambda\beta\right)}{(\lambda+\alpha)(1-\beta)^2(k\mu-\lambda)^2}\right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu(\lambda+\alpha)}{(n+1)!k!(\lambda+\alpha)(1-\beta)(k\mu-\lambda)}\cdot\left((k-1)+\frac{\left(k\mu-\lambda\beta\right)}{(1-\beta)(k\mu-\lambda)}\right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu\cancel{(\lambda+\alpha)}}{(n+1)!k!\cancel{(\lambda+\alpha)}(1-\beta)^2(k\mu-\lambda)^2}\cdot\left((k-1)(1-\beta)(k\mu-\lambda)+k\mu-\lambda\beta\right)$$

$$\Longrightarrow \frac{\rho^{n+k}\pi_{0,0}k\mu}{(n+1)!k!(1-\beta)^2(k\mu-\lambda)^2}\cdot\left((k-1)(1-\beta)(k\mu-\lambda)+k\mu-\lambda\beta\right)\qquad(\star\star)'$$

For $(\star)$ from Mathematica where $\Gamma(a,x)$ is the upper incomplete gamma function

$$\sum_{j=0}^{n}\frac{\rho^j}{j!}=\frac{\exp(\rho)\cdot\Gamma(n+1,\rho)}{n!}$$

So

$$(\star)\Longrightarrow\pi_{0,0}\rho\sum_{j=0}^{n}\frac{\rho^{j-1}}{(j-1)!}+\frac{\rho^{n+1}}{(n+1)!}\cdot\frac{\pi_{0,0}}{(1-\beta)}\sum_{i=1}^{k}\frac{\rho^{i-1}}{(i-1)!}+\frac{\rho^n}{(n+1)!}\cdot\frac{\pi_{0,0}}{(1-\beta)^2}\sum_{i=0}^{k}\frac{\rho^{i-1}}{(i)!}$$

$$(\star)\Longrightarrow\pi_{0,0}\rho\frac{\exp(\rho)\cdot\Gamma(n,\rho)}{(n-1)!}+\frac{\rho^{n+1}}{(n+1)!}\cdot\frac{\pi_{0,0}}{(1-\beta)}\frac{\exp(\rho)\cdot\Gamma(k,\rho)}{(k-1)!}+\frac{\rho^n}{(n+1)!}\cdot\frac{\pi_{0,0}}{(1-\beta)^2}\frac{\exp(\rho)\cdot\Gamma(k+1,\rho)}{k!}\qquad(\star)'$$

From $(\star)'$ and $(\star\star)'$

$$\mathbb{E}[N]=\pi_{0,0}\rho\left[\frac{\exp(\rho)\cdot\Gamma(n,\rho)}{(n-1)!}+\frac{\rho^n}{(n+1)!\cdot(1-\beta)}\frac{\exp(\rho)\cdot\Gamma(k,\rho)}{(k-1)!}+\frac{\rho^n}{(n+1)!\cdot(1-\beta)^2}\frac{\exp(\rho)\cdot\Gamma(k+1,\rho)}{(k)!}+\frac{\rho^{n+k-1}k\mu}{(n+1)!k!}\cdot\frac{(k-1)(1-\beta)(k\mu-\lambda)+k\mu-\lambda\beta}{(1-\beta)^2(k\mu-\lambda)^2}\right]$$

$$\mathbb{E}[N]=\pi_{0,0}\rho\left[\sum_{j=0}^{n-1}\frac{\rho^j}{j!}+\frac{\rho^n}{(n+1)!\cdot(1-\beta)}\sum_{j=0}^{k-1}\frac{\rho^j}{j!}+\frac{\rho^n}{(n+1)!\cdot(1-\beta)^2}\sum_{j=0}^{k}\frac{\rho^j}{j!}+\frac{\rho^{n+k-1}k\mu}{(n+1)!k!}\cdot\frac{(k-1)(1-\beta)(k\mu-\lambda)+k\mu-\lambda\beta}{(1-\beta)^2(k\mu-\lambda)^2}\right]\tag{4.25}$$

■

Corollary 4.1. *The Mean response time* $\mathbb{E}[T]$

$$\mathbb{E}[T] = \frac{\pi_{0,0}}{\mu}\left[\sum_{j=0}^{n-1}\frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!\cdot(1-\beta)}\sum_{j=0}^{k-1}\frac{\rho^j}{j!} + \frac{\rho^n}{(n+1)!\cdot(1-\beta)^2}\sum_{j=0}^{k}\frac{\rho^j}{j!} + \frac{\rho^{n+k-1}k\mu}{(n+1)!k!}\cdot\frac{(k-1)(1-\beta)(k\mu-\lambda)+k\mu-\lambda\beta}{(1-\beta)^2(k\mu-\lambda)^2}\right] \tag{4.26}$$

Proof. We invoke Little's law to obtain the mean response time

$$\mathbb{E}\left[T^{M/M/k/\text{setup}}\right] = \frac{\mathbb{E}\left[N^{M/M/k/\text{setup}}\right]}{\lambda}$$

■

4.5.1 The expected number of servers either ON or in setup

We know that a server can be in any of the following three state : i/ OFF, ii/ ON and iii/ Setup, We are interested in the expected number of servers not in the OFF state. Let B_{busy} value to each of the states in the $M/M/k$ Markov chain shown in the figure .
For example , in the state $(0,0)$, $B_{busy}(0,0) = 0$, in the state $(0,1)$, $B_{busy}(0,1) = 1$, for state $(0,j)$, where $1 < j \leq n$, $B_{busy}(0,j) = j$, since we only allow one server to be in the setup mode at any time. In general, for state (i,j), we have :

$$B_{busy}(i,j) = \begin{cases} j & if \quad i = 0, \quad 0 \leq j \leq n \\ n & if \quad i = 0, \quad j \geq n \\ i+n & if \quad i > 0, \quad j \geq n+1 \\ k & if \quad i = k, \quad j \geq n+k \end{cases} \tag{4.27}$$

Thus, the expected number of servers either ON or in Setup, is given by :

$$\mathbb{E}[B_{busy}] = \sum_{i,j}\pi_{i,j}B_{busy}(i,j) \tag{4.28}$$

Using Eqs 4.2, 4.3, 4.13 and 4.21 in the above equation, we get :

$$
\begin{aligned}
\mathbb{E}[B_{busy}] &= \sum_{i,j} \pi_{i,j} B_{busy}(i,j) \\
&= \sum_{i=0}^{k-1} \sum_{j=0}^{+\infty} \pi_{i,j} B_{busy}(i,j) + \sum_{\substack{j=k+n \\ i=k}}^{+\infty} \pi_{k,j} B_{busy}(i,j) \\
&= \sum_{i=0}^{n} \pi_{0,j} \cdot j + \sum_{j=n+1}^{+\infty} \pi_{0,j} \cdot n + \sum_{i=1}^{k-1} \sum_{j=n+1}^{+\infty} \pi_{i,j}(i+n) + \sum_{j=k+n}^{+\infty} \pi_{k,j} k \\
&= \sum_{j=0}^{n} \pi_{0,0} \frac{\rho^j}{j!} \cdot j + \sum_{j=n+1}^{+\infty} \pi_{0,0} \frac{\rho^n}{n!} \beta^j \cdot n + \sum_{i=1}^{k-1} \sum_{j=n+1}^{+\infty} \pi_{0,0} \frac{\gamma^i}{i!} \frac{\rho^n}{(n+1)!} \beta^j (i+n) + k \sum_{j=k+n}^{+\infty} \pi_{k,j} \\
&= \pi_{0,0} \rho \sum_{j=1}^{n} \frac{\rho^{j-1}}{(j-1)!} + \pi_{0,0} \frac{\rho^n}{(n-1)!} \sum_{j=n+1}^{+\infty} \beta^j + \pi_{0,0} \frac{\rho^n}{(n+1)!} \sum_{i=1}^{k-1} \frac{\gamma^i}{i!} (i+n) \sum_{j=n+1}^{+\infty} \beta^j \\
&+ \frac{\rho^n}{(n+1)!} k \sum_{j=k+n}^{+\infty} \left[\frac{\pi_{0,0} \gamma^k k \mu}{k!(k\mu - (\lambda+\alpha))} \cdot \beta^j - \frac{\pi_{0,0} k^k (\lambda+\alpha)}{k!(k\mu - (\lambda+\alpha))} \cdot \left(\frac{\rho}{k} \right)^j \right] \\
&= \pi_{0,0} \rho \sum_{j=0}^{n} \frac{\rho^j}{j!} + \pi_{0,0} \frac{\rho^n}{(n-1)!} \frac{\beta^{n+1}}{1-\beta} + \pi_{0,0} \frac{\rho^n}{(n+1)!} \sum_{i=1}^{k-1} \frac{\gamma^i}{i!} (i+n) \frac{\beta^{n+1}}{1-\beta} \\
&+ \frac{\rho^n}{(n+1)!} k \left[\frac{\pi_{0,0} \gamma^k k \mu}{k!(k\mu - (\lambda+\alpha))} \cdot \sum_{j=k+n}^{+\infty} \beta^j - \frac{\pi_{0,0} k^k (\lambda+\alpha)}{k!(k\mu - (\lambda+\alpha))} \cdot \sum_{j=k+n}^{+\infty} \left(\frac{\rho}{k} \right)^j \right] \\
&= \pi_{0,0} \rho \sum_{j=0}^{n} \frac{\rho^j}{j!} + \pi_{0,0} \frac{\rho^n}{(n-1)!} \frac{\beta^{n+1}}{(1-\beta)} + \pi_{0,0} \frac{\rho^n}{(n+1)!} \cdot \frac{\beta^{n+1}}{(1-\beta)} \sum_{i=1}^{k-1} \frac{\gamma^i}{i!} (i+n) \\
&+ \frac{\rho^n}{(n+1)!} \left[\frac{\pi_{0,0} \gamma^k k^2 \mu}{k!(k\mu - (\lambda+\alpha))} \cdot \frac{\beta^{n+k}}{(1-\beta)} - k \frac{\pi_{0,0} k^k (\lambda+\alpha)}{k!(k\mu - (\lambda+\alpha))} \cdot \frac{(\frac{\rho}{k})^{n+k}}{(1-(\frac{\rho}{k}))} \right] \\
&= \pi_{0,0} \rho \left[\sum_{j=0}^{n-1} \frac{\rho^j}{j!} + \frac{\rho^{n-1}}{(n-1)!} \frac{\beta^{n+1}}{(1-\beta)} + \frac{\rho^{n-1}}{(n+1)!} \cdot \frac{\beta^{n+1}}{(1-\beta)} \gamma \sum_{i=1}^{k-1} \frac{\gamma^{i-1}}{(i-1)!} + \frac{\rho^{n-1}}{(n+1)!} \cdot \frac{\beta^{n+1}}{(1-\beta)} n \sum_{i=1}^{k-1} \frac{\gamma^i}{i!} \right] \\
&+ \frac{\beta^n}{(n+1)!} \frac{\pi_{0,0} \rho^{k+n}}{k!(k\mu - (\lambda+\alpha))} \left[\frac{k\mu}{\gamma^n (1-\beta)} - \frac{(\lambda+\alpha)}{(1-\frac{\rho}{k}) k^n} \right]
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}[B_{busy}] &= \pi_{0,0} \rho \left[\sum_{j=0}^{n-1} \frac{\rho^j}{j!} + \frac{\beta^{n+1}}{(1-\beta)} \left[\frac{\rho^{n-1}}{(n-1)!} + \frac{\rho^{n-1}}{(n+1)!} \cdot \gamma \sum_{i=1}^{k-1} \frac{\gamma^{i-1}}{(i-1)!} + \frac{\rho^{n-1}}{(n+1)!} \cdot n \sum_{i=1}^{k-1} \frac{\gamma^i}{i!} \right] \right] \\
&+ \frac{\rho^n \cdot k \cdot \rho^{n+k} \pi_{0,0}}{(n+1)! \cdot k!(k\mu - (\lambda+\alpha))} \left[\frac{k^n k\mu - \mu k^n \rho - \gamma^n \lambda - \gamma^n \alpha + \gamma^n \beta(\lambda+\alpha)}{\gamma^n (1-\beta)(1-\frac{\rho}{k}) k^n} \right] \\
&= \pi_{0,0} \rho \left[\sum_{j=0}^{n-1} \frac{\rho^j}{j!} + \frac{\beta^{n+1}}{(1-\beta)} \left[\frac{\rho^{n-1}}{(n-1)!} + \frac{\rho^{n-1}}{(n+1)!} \cdot \gamma \sum_{i=1}^{k-1} \frac{\gamma^{i-1}}{(i-1)!} + \frac{\rho^{n-1}}{(n+1)!} \cdot n \sum_{i=1}^{k-1} \frac{\gamma^i}{i!} \right] \right] \\
&+ \frac{\beta^n \cdot k \cdot \rho^{n+k} \pi_{0,0}}{(n+1)! \cdot k!(k\mu - (\lambda+\alpha))} \left[\frac{k^n (k\mu - \lambda) - \gamma^n \alpha}{(1-\beta)(1-\frac{\rho}{k}) k^n} \right]
\end{aligned}
$$

Since$(\star)$

$$\mathbb{E}[B_{busy}] = \pi_{0,0}\rho\left[\frac{\exp(\rho)\cdot\Gamma(n,\rho)}{(n-1)!} + \frac{\beta^{n+1}}{(1-\beta)}\left[\frac{\rho^{n-1}}{(n-1)!} + \frac{\rho^{n-1}}{(n+1)!}\cdot\gamma\frac{\exp(\gamma)\cdot\Gamma(k-1,\gamma)}{(k-2)!} + \frac{\rho^{n-1}}{(n+1)!}\cdot n\frac{\exp(\gamma)\cdot\Gamma(k,\gamma)}{(k-1)!}\right]\right]$$
$$+ \frac{\beta^n\cdot k\cdot\rho^{n+k}\pi_{0,0}}{(n+1)!\cdot k!(k\mu-(\lambda+\alpha))}\left[\frac{k^n(k\mu-\lambda)-\gamma^n\alpha}{(1-\beta)(1-\frac{\rho}{k})k^n}\right]$$

4.5.2 The mean power consumption

As stated earlier, our M/M/k with setup costs model is motivated by the problem of power management in server farms. Most server farm operators today are interested in minimizing both power usage and mean response time. While minimizing mean response time points to turning on many servers, minimizing power usage points to turning on few servers. To save on power usage, it is customary to turn servers OFF when they are not in use. However, this means that a setup cost is required every time a server needs to be turned on. This setup cost is both wasteful with respect to mean response time and with respect to power usage, since power is needed during the whole setup period. Since the setup cost can have a big effect on both power and response time, it is important that we take it into account when optimizing the configuration of our server farm. The performance metric we consider in this section is a weighted sum of the mean response time, $\mathbb{E}[T]$, and the mean power consumption, $\mathbb{E}\left[P^{M/M/k/\text{setup}}\right]$, given by

$$Perf = \mathbb{E}\left[P^{M/M/k/\text{setup}}\right] + c\cdot\mathbb{E}\left[T^{M/M/k/\text{setup}}\right]$$

The weight parameter c has units of Watts/sec, and can be thought of as the price required, in Watts, to lower the mean response time of the server farm by 1 second. A similar weighted linear combination has previously been used in literature [86], [87] and [88].

we know that a server can be in any of the following three states: (i) OFF, (ii) ON or (iii) SETUP. While in the OFF state, we assume that the power consumption of a server is 0. While in the ON or SETUP states, the power consumption of the server is assumed to be a constant, P_{max}. Given this information,we see that

$$\mathbb{E}\left[P^{M/M/k/\text{setup}}\right] = P_{max}.\mathbb{E}[B_{busy}]$$

4.6 Numerical results

In this section, we present some numerical results obtained from the proposed mechanism, illustrating and quantifying the performance of the studied system under different loading conditions and system parameters. Validation is performed based on numerical data presented.

We consider a M/M/K/Setup .where The maximum number of requests in the system is 18, the number of reserves is 8. To evaluate the power consumption, the power weight in the principal server and the idle state are 100 watts and 10 watts, respectively.

Figure 4.2 and 4.3 show the energy consumption behaviour of the system when the proposed mechanisms M/M/k/Setup is used and compares it to the M/M/k system.

For figure 4.2 shows that the M/M/K case is the worst case in terms of energy consumption. In addition, energy consumption becomes lower when we use the reserves. An M/M/k/Setup configuration is more efficient and saves power. Indeed, in this case the energy of the system

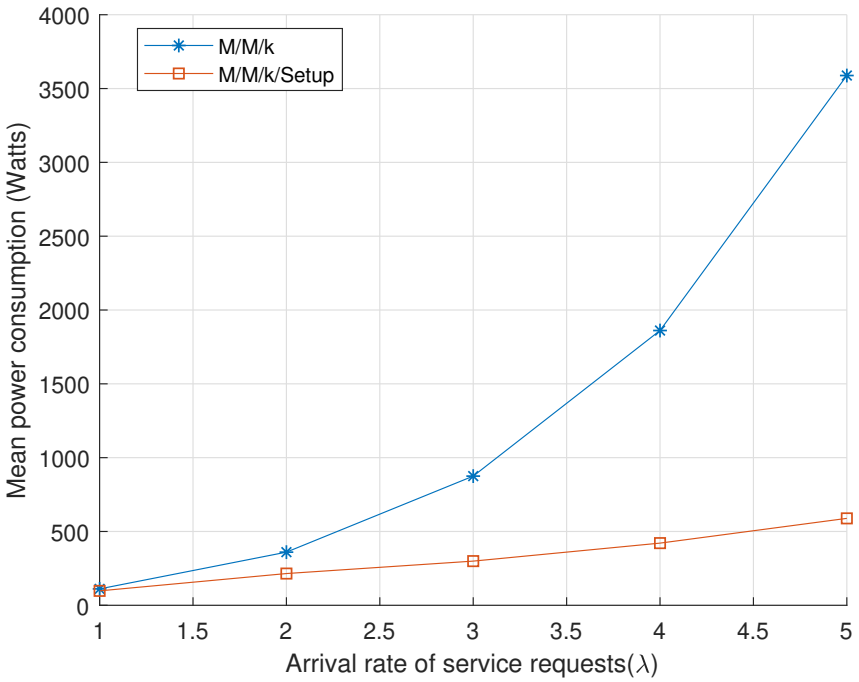

Figure 4.2 – Mean power consumption curves as functions of requests arrival rate

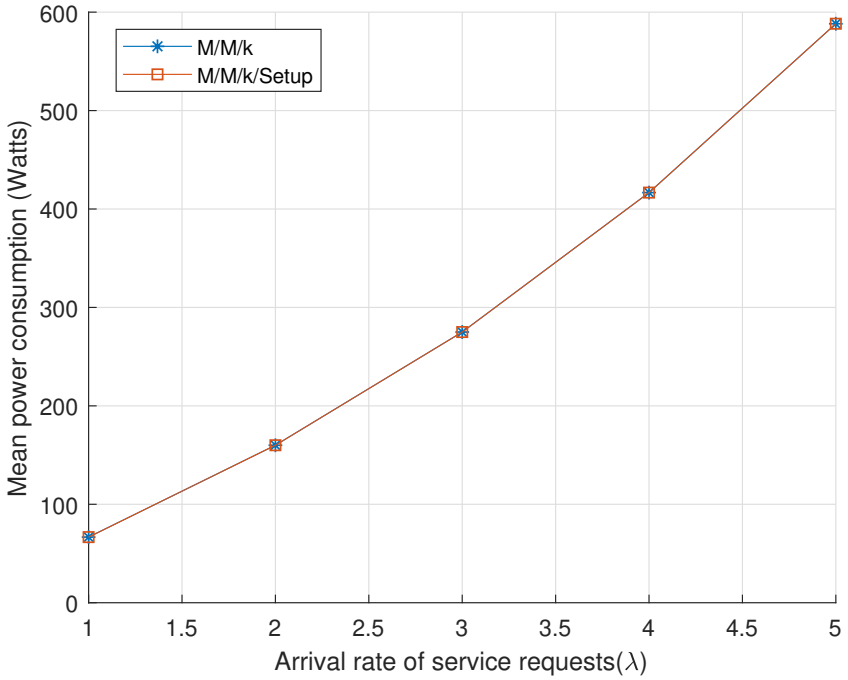

Figure 4.3 – Mean power consumption curves as functions of requests arrival rate without reserves

is low and the maximum capacity in the system is not reached. As far as the figure 4.3 shows a comparison of power consumption when we don't use any reserve in our system M/M/k/Setup and M/M/k, and that gives us a confirmation that our mechanism can reduce energy .

4.7 Conclusion

In this chapter, we began by examining the M/M/k queuing system with setup times. We provided the first analytical expressions of the closed forms for the average response time, limiting the distribution of the number of tasks in the system M/M/k/Setups.

The proposed mechanism is analyzed using a mathematical analyses of the system's parameters are derived. In addition, the system's energy consumption within the suggested mechanisms is modeled and evaluated. As future work, we plan to determine the optimal number of servers to use in a server farm with setup time, so as to reduce a weighted sum of the average power and the average response time.

Chapter 5

Modeling Decision Making to Control the Allocation of Virtual Machines in a Cloud Computing System with Reserve Machines

Contents

In this chapter, we propose an optimal model for allocating computing resources to assess the best management of system resources where a group of physical machines is defined as "reserves", the controller activates them one by one when the system has a high number of tasks. The objective is to maximize the reward of the cloud computing system, this reward is calculated based on the energy and execution time of each customer and the characteristics of the system. Finding the best allocation for such a complex system is a challenge. For this, we used a heuristic algorithm and dynamic programming. The results analysis showed the advantage of using our model to control the use of reserve machines to get a high quality of service and low energy consumption.

5.1 Introduction

Due to the complex and dynamic nature of virtual machine allocation in the cloud, it is difficult to manage and control the resources or to choose the best allocation of these resources[89]. Managing resources in such an environment is a difficult task, but it is an essential function of any dynamic system[90]. It demands sophisticated policies and decisions to manage complex multi-dimensional objectives such as CPU, memory and network bandwidth. Therefore, for the correct use of resources, efficient resource management can control the system resources and try to find an optimal balance between the QoS requirement and the quantity of energy consumption [91].

The energy consumption in the cloud data centers has grown exponentially in the IT sector in recent years. Data centres, as a principal component of current information and communication technology, have grown at an unprecedented rate as IT developers[92]. These data centers are loaded with thousands of servers and switches that consume huge amounts of energy, increasing operating costs and increasing carbon dioxide emissions into the environment[93]. On the other hand, cooling equipment must be used to manage the heat released by these data centres, which also consume energy [94, 95].

The idea in this work is to consider a group of physical machines (PM) as "reserve". The reserve status is controlled by the number of clients in the system on which they are turned on when the number of tasks in the system becomes large enough. The question, of course, is how many reserves we have to use and when we turn on a new PM. The answer depends on the demand parameters and the cost function to minimize, and this function should include both a performance cost (e.g. execution time) and the energy consumed by the physical machines of the server. Obviously, there is a compromise between the two parameters, as performance is improved by increasing the number of physical machines powered, while power consumption is improved by increasing the number of physical machines powered off. That trade-off can be evaluated by analyzing and solving a Semi Markov Decision Process (SMDP) model of the system. The solution provides average performance and average consumption measurements [96]. This makes it easy to calculate the cost function and minimize it by maximizing the total expected reward of the cloud computing system. However, the resolution of the SMDP problem on a large scale depends on the dimensionality constraint. Therefore, we make better use of the problem's characteristic and propose a dynamic resource management method. Since the system model contains a controller for allocation demand, the optimal policy of the requests is dynamically achieved by applying the value iteration algorithm (VI) [97]. As an implementation of our algorithm, we studied the optimal choice of the controller in order to control the performance of the system and to minimize energy consumed, and also to answer the question when we turn on a new physical machine.

The content of this chapter is organized as follows. It begins with a literature review in Section 5.2, followed by the development of the System Model in Section 5.3. Section 5.4 is

reserved for the formulation of the system using the semi-Markov decision process, and the reward of the system is calculated and solved in section 5.5, followed by a numerical analysis of the performance in section 5.6. Finally, in section 5.7 a conclusion.

The results obtained in this chapter are presented in the research article [98] and [99].

5.2 General context and related work

In the literature, there are some recent efforts devoted to managing resources in a cloud computing environment [100]. Gandhi et al. [33] analyse an ON-OFF policy in which a server is switched on if there is a pending task and is switched off if there are no pending tasks. The work in [60] Proposed a novel fitness function to maximize the energy efficiency (i.e., profit) of cloud data centers while minimizing the overall energy consumption with reserved VM requests. Several studies combining the quality of service analysis and cloud energy consumption were proposed recently. Ouammou et al [46] have suggested an energy-efficient load balancing algorithm. They used a policy to determine the upper and lower thresholds for each physical machine. This approach has managed the number of migrations. But the main problem with this approach is that they used fixed values of the upper and lower thresholds. In [34], the authors have presented a mathematical queuing model to study the performance of multi-core virtual machines hosting SaaS cloud services. The authors in [26]Maximize the use of resources by using two heuristics of consolidation noted energy-conscious taking into account both active and inactive energy consumption; the suggested model can be used to estimate the number of multi-core virtual machine instances needed to reach the QoS parameters under any additional workload. Another recent work in the research area has addressed Markov chain theory to estimate the efficiency of queues in the cloud servers. Hanini et al [36] introduced a continuous Markov chain (CTMC) approach to managing the use of virtual machines in an MP with a controlled system workload. They examined the proposed model on the base of mathematical analysis of the QoS parameters of the system. Also, there is some published work on the resources managements as [101], [102] and [103] they formulated their work using the Semi Markov decision process to manage the optimal action that will be taken under some conditions.The authors of [104] are interested in proposing an algorithm to schedule the client tasks to the resources of a data center in cloud computing. Their proposed solution, called OSACO algorithm, is concentrated on client QoS requirements and aimed to reduce the energy, cost and time.

In general, the majority of dynamic approaches are focused on heuristics, and consequently do not have sufficient theoretical guarantees of performance and, to our knowledge, a dynamic policy of the type proposed here has not been studied before. and perhaps the model closest to the one presented here is that examined by Isi mitarini [37] where the author studied the problem of energy use when servers are powered up and down in a frame of the data centers,

this means that after the setup time, all servers start running simultaneously even if the system does not require all servers. This gives us reason to reconsider the case where the configuration time of each server is considered different.

In this chapter, we consider resources management as a stochastic optimization problem. Our purpose in this work can be summarized as follows:

- To propose a mechanism that helps to manage the resources by improving QoS and minimizing energy consumption based on a "PM reserve" approach.
- To formulate the dynamic virtual machine management problem as a Markov decision process problem, in order to maximize the reward of the cloud computing system.

Based on the advanced algorithm, we show that our approach can significantly reduce the evaluation time of VM allocation solutions in each scalable iteration.

5.3 System model

5.3.1 Problem statement

The studied system is a server containing M physical machines, of which Q are designated as Main Physical Machines (MPM) and $(M - Q)$ considered as Reserves Physical Machines (RPM) $0 \leq Q \leq M$. Considering that each PM can host many Virtual machines (VM). In order to minimize the energy consumption even when we serve many clients at the same time, we optimally consolidate VMs in a minimum number of PMs. Clients arrive at the server in a Poisson process λ_p and find a controller in front of them that takes them through an MPM until the system gets n clients (Jobs) (n is called a "controller number"). At that time, the controller sends a feedback to the administrator to informe him to reduce the rate of clients sent to the servers from λ_p to λ_s, and then the controller must decide whether to send it to one of the MPMs that contain customers or to any of the RPM, based on system conditions. The service times are independent and distributed exponentially with the following parameters $\mu_1, \mu_2, \ldots, \mu_M$, such that $\mu_i = \mu_p$ with $i = 1, \ldots, Q$ and $\mu_j = \mu_s$ with $j = Q + 1, \ldots, M$ with $Q < M$.

We notice that when the client arrives, and at least one of the MPM is available to host this client under some conditions related to the workload in these MPs, the controller send it directly to one of the MPM, In the other case, when all MPM are noted empties, the controller can either send the customer to any of the MPMs. In this scenario, the system controller chooses to allocate multiple customers to a limited number of MPM in an attempt to reduce the total energy consumed, alternatively, the controller can use a new physical machine from the reserves, thus increasing the consumption of energy and in return improving the quality of service. To help the controller to decide what decision should be made for the benefits of the

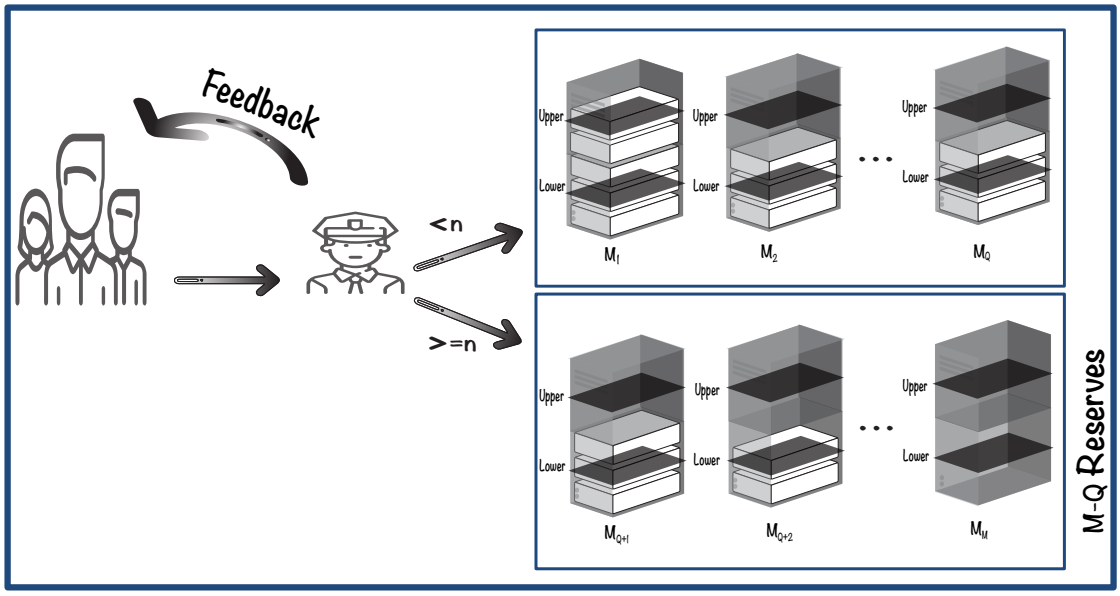

Figure 5.1 – Illustration of the proposed model

system, a decision policy is proposed in this chapter. For this, two thresholds are considered for each PM in order to maintain capacity below the upper threshold and above the lower, the details of this idea have been described in [46]. So if a client comes into the system, a VM describes the request of this client (job) and it has to be mapped to one of the available MPM as long as the capacity of this PM after allocation will not exceed the Upper threshold, and if this client has no place to mapped to any MPM, then the controller decides to turn on a new physical machine among the reserves and so on.

For our analysis, we assumed that the arrival rate of the clients follows a Poisson arrival and the service times are exponentially distributed. These assumptions has been usually used in the literature and can provide adequate approximation of real systems such cloud computing environment [105, 106, 107, 108].

The notations used in this document are listed in Table 5.1. and a graphical illustration for the considered model is presented in the figure 5.1.

5.3.2 Mathematical formulation

In this sub-section, using integer mixed linear programming, we will mathematically formulate the model that we have proposed. The objective is to reduce as much as possible the number of machines used from reserves.

The objective function is expressed as follows:

$$\min_{z_j} \sum_{j=1}^{M} z_j \tag{5.1}$$

Table 5.1 – List of important notations

Notation	Meaning
M	Number of all physical machines in the server
Q	Number of the PMs designated as Main PM
K	Maximal number of the VMs that the system can support
l	Number of the VMs that each PM can support
n	Number of clients used to decide when the system can turn on a new PM
x_k^p	Number of service requests that have been allocated in the k PM among the MPM
x_k^r	Number of service requests that have been allocated in the k PM among the RPM
λ_p	Arrival rate of a VM in MPM
μ_p	Service rate of a VM in MPM
λ_s	Arrival rate of a VM in RPM
μ_s	Service rate of a VM in RPM
C_{ij}	Capacity of the VM i located in j^{th} PM
$Upper_j$	Upper threshold for PM j
$Lower_j$	Down threshold for PM j

where z_j are integer variables defined by the following equation

$$\forall j \in [1, M] \qquad z_j = \begin{cases} 1, & \text{The } j^{th} \text{ PM is used} \\ 0, & \text{Otherwise} \end{cases} \tag{5.2}$$

This optimization is based on linear constrains; each VM can be hosted just on a PM, and each PM is able to host VMs not only based on the amount of remaining capacity but also on the impact of hosting them on the performance of the system [99].

- C_j : The sum of the capacity in all VMs used in a PM_d must be between its $Upper$ and $Lower$ thresholds.

$$Lower_j z_j \leq C_j = \sum_{i=1}^{l} C_{ij} V_{ij} \leq Upper_j z_j, \forall j \in [1, M], \tag{5.3}$$

where V_{ij} are decision variables defined by

$$V_{ij} = \begin{cases} 1, & \begin{cases} \text{The VM } i \text{ is mapped to } j^{th} PM \\ \forall j \in [1, M], \quad \forall i \in [1, l] \end{cases} \\ 0, & \text{Otherwise} \end{cases} \tag{5.4}$$

Note that if for a given i and j $V_{ij} = 1$ then $z_j = 1$

- The number of VMs that exist in a physical machines j as long as the sum of their capacities does not exceed the upper threshold is :

$$n_j = \sum_{i=1}^{l} V_{ij} \leq l, \quad \forall j \in [1, M] \tag{5.5}$$

- The number of VMs that exist in all MPM as long as the sum of their capacities does not exceed the upper threshold of each PM is:

$$n = \sum_{j=1}^{Q} n_j \leq l \times Q \tag{5.6}$$

- No virtual machine can exist in two physical machines.

$$\sum_{j=1}^{M} V_{ij} \leq 1, \quad \forall i \in [1, l], \tag{5.7}$$

We can summarize these by adding the objective function to all the constraints in the follows set of equations:

$$\min_{z_j} \sum_{j=1}^{M} z_j \quad \textbf{subject to}$$
$$\begin{cases} Lower_j z_j \ \leq \sum_{i=1}^{l} C_{ij} V_{ij} \leq Upper_j z_j \quad \forall j \in [1, M], \\ \sum_{j=1}^{M} V_{ij} \leq 1, \quad \forall i \in [1, l]. \\ n_j = \sum_{i=1}^{l} V_{ij} \leq l, \quad \forall j \in [1, M]. \\ n = \sum_{j=1}^{Q} n_j \leq l \times Q. \end{cases} \tag{5.8}$$

The system must always satisfy the constraints defined in the above equation. However, taking into account the dynamic character of the studied environment, it is also necessary that

the actions are chosen at any particular time be designed to minimise the cost caused by the operation of the system. For this, we will consider an expected reward to be maximized in the long run of the system. This maximization makes it possible to determine the optimal policy for deciding what action to take, this can be done using a Markov Decision Process [109], [110]. The resolution of such a problem will require an approach from the theory of dynamic programming. This is the purpose of the following in this chapter.

5.4 System formulation using Semi Markov Decision process

5.4.1 System states

Let S be the set of states that present the current requests with a variable number of VMs available in the system :

$$S = \{s | s = (x^p, x^r, Q, \epsilon) = (x_1^p, x_2^p, ...x_Q^p, x_{Q+1}^r, ..., x_M^r, Q, \epsilon)\}$$

Both x_k^p and x_k^r are described in Table 1, and let Y represents the set of events such as $\epsilon \in Y = \{A_p, A_r, D\}$ Where A_p is the arrival of the customer when at least one of the MPMs is available, A_r is the arrival of the customer when all the MPMs already have at least one customer, D denotes the leaving of a request allocated to one of the MPMs or RPMs.

5.4.2 Set of actions

We note that in our analysis model, a number of actions are possible to be taken as part of the set of actions A , i.e:

$$A = \{-1, 0, R, P_r\}$$

At the occurrence of an event, the system controller takes an action $a(s)$ from the set of actions A under system conditions.

$$a(s) = \begin{cases} -1 & \epsilon = D \in \{D_p, D_r\} \\ 0 \text{ or } R & \epsilon = A_r \\ P_r & \epsilon = A_p \end{cases} \tag{5.9}$$

When the system receives a new client, the following actions are determined by the system.

- $a(s) = P_r$: means that the request was accepted directly into an MPM before the system had n clients.
- $a(s) = R$: when the request became after the Q MPM have already been used by at least one client, and that client has been accepted.
- $a(s) = -1$: represents the instances when the service request terminated and leaves the system, and no service is requested, only that the information of the number of virtual machines available in the system must be updated.

- $a(s) = 0$: when the request is rejected.

5.4.3 Decision time

When a state s and an action a are given, then the service time to get the next state is indicated by $\tau(s, a)$. That determine by the average rate $\gamma(s, a)$ of the events, which is the opposite of the time interval between two decisions $\tau(s, a) = \gamma(s, a)^{-1}$.

So the average rate $\gamma(s, a)$ of the events is the sum of the rate of all system that can be expressed by:

$\gamma(s, a) =$

$$\begin{cases} Q\lambda_p + m\lambda_s + \sum_{i=1}^{Q} \mu_p + \sum_{i=Q+1}^{M} \mu_s & a(s) = 0, \quad \epsilon = A_r \\ Q\lambda_p + m\lambda_s + \sum_{i=1}^{Q} z_i\mu_p + \sum_{i=Q+1}^{M} z_i\mu_s & a(s) = -1, \quad \epsilon = D \\ Q\lambda_p + m\lambda_s + \sum_{i=1}^{Q} z_i\mu_p + \sum_{i=Q+1}^{M} z_i\mu_s + \mu_s & \begin{cases} a(s) = R, \epsilon = A_r, \\ \sum_{i=1}^{Q} V_{ij} > n \end{cases} \\ Q\lambda_p + m\lambda_s + \sum_{i=1}^{Q} z_i\mu_p + \mu_p & \begin{cases} a(s) = Pr, \epsilon = A_p \\ \sum_{i=1}^{Q} V_{ij} \leq n \end{cases} \end{cases} \tag{5.10}$$

In Eq (5.10), If the system does not accept any requests, there is $\sum_{i=1}^{Q} z_i$ services that exist in the system, then the service rate of the global system is $\sum_{i=1}^{Q} z_i\mu_p$. If a request is admitted, then there are $\sum_{i=1}^{Q} z_i + 1$ services in the system, if the number of clients does not reach n clients, so the leaving rate is $\sum_{i=1}^{Q} z_i\mu_p + \mu_p$. Similarly, when the system have already received n clients and start to use the RPM; then the rate of leaving is $\sum_{i=1}^{Q} z_i\mu_p + \sum_{i=Q+1}^{M} z_i\mu_s + \mu_s$

5.4.4 Transition probability

Now that the average event rate is calculated, the transition probability $p(\bar{s}/s, a)$ can be calculated from the state s to the next state $\bar{s}$ when the action a is happens, in the following we will discuss possible cases of $p(\bar{s}/s, a)$ according to the s statements as listed below.

When $s = (x^p, x^r, Q, A_p)$ where $a(s) = P_r$:

$P(\bar{s}/s,a) =$

$$\begin{cases} \frac{(x^p - l_i)\mu_p}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p - l_i, x^r, Q, D_p) \\ a(s) = -1, \quad i = 1,2,...,Q. \end{cases} \\ \frac{\lambda_s}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p, x^r + l_j, Q, A_r) \\ a(s) = R, \quad j = 1,2,...,(M-Q). \end{cases} \\ \frac{\lambda_p}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p + l_i, x^r, Q, A_p) \\ a(s) = P_r, \quad i = 1,2,...,Q. \end{cases} \end{cases} \tag{5.11}$$

We note that the vector l_i is a null vector with Q elements, except the i^{th} element which is 1, and the vector l_j is also a null vector of $M-Q$ elements, except the j^{th} element which is 1.

When $s = (x^p, x^r, Q, A_r)$ where $a(s) = 0$:

$P(\bar{s}/s,a) =$

$$\begin{cases} \frac{(x^p - l_i)\mu_p}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p - l_i, x^r, Q, D_p) \\ a(s) = -1, \quad i = 1,2,...,Q. \end{cases} \\ \frac{(x^r - l_j)\mu_s}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p, x^r - l_j, Q, D_r) \\ a(s) = -1, \quad j = 1,2,...,(M-Q). \end{cases} \\ \frac{\lambda_s}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p, x^r, Q, A_r) \\ a(s) = 0. \end{cases} \end{cases} \tag{5.12}$$

When $s = (x^p, x^r, Q, A_r)$ where $a(s) = R$:

$P(\bar{s}/s,a) =$

$$\begin{cases} \frac{(x^p - l_i)\mu_p}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p - l_i, x^r, Q, D_p) \\ a(s) = -1, \quad i = 1,2,...,Q. \end{cases} \\ \frac{(x^r - l_j)\mu_s}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p, x^r - l_j, Q, D_r) \\ a(s) = -1, \quad j = 1,2,...,(M-Q). \end{cases} \\ \frac{\lambda_s}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p, x^r, Q, A_r) \\ a(s) = 0. \end{cases} \\ \frac{\lambda_s}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p, x^r + l_j, Q, A_r) \\ a(s) = R, \quad j = 1,2,...,(M-Q). \end{cases} \end{cases} \tag{5.13}$$

When the state $s = (x^p, x^r, A_p)$, where $\epsilon = D$, no specific action is required except an update of active VMs, where $a(s) = -1$,The associated transition probabilities are given as follows

$P(\bar{s}/s,a) =$

$$\begin{cases} \frac{(x^p - l_i)\mu_p}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p - l_i, x^r, Q, D_p) \\ a(s) = -1, \quad i = 1,2,...,Q. \end{cases} \\ \frac{(x^r - l_j)\mu_s}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p, x^r - l_j, Q, D_r) \\ a(s) = -1, \quad j = 1,2,...,(M-Q). \end{cases} \\ \frac{\lambda_s}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p, x^r + l_j, Q, A_r) \\ a(s) = R, \quad j = 1,2,...,(M-Q). \end{cases} \\ \frac{\lambda_p}{\gamma(s,a)} & \begin{cases} \bar{s} = (x^p + l_i, x^r, Q, A_p) \\ a(s) = P_r, \quad i = 1,2,...,Q. \end{cases} \end{cases} \tag{5.14}$$

5.5 Cost and optimal solution

The system cost under the state s and the taken action a is noted by :

$$c(s,a) = r(s,a)\tau(s,a) \tag{5.15}$$

The $c(s, a)$ is the system cost estimated by taking the a action under the state s in the event ϵ happens. This combines both the revenue and the cost of the system. The main benefit of the system is to conserve energy consumption and ensure the quality of service. The function must reflect the impact of both [111]. To calculate the energy consumed in the system, the following equation is used :

$$E_{used} = E_Q + \sum_{j=1}^{M} \sum_{i=1}^{l} \delta \times C_{ij} + \sum_{j=Q+1}^{M} P_j z_j \tag{5.16}$$

where E_Q is the power needed for monitoring the system when just the MPM is running, P_j is the power consumption to start a new PM from RPM, and δ is a powerful weight coefficient.
Besides, $r(s, a)$ is defined by the number of occupied VMs in the PMs taking into account the total capacity of the system i.e.

$$c(s,a) = \begin{cases} E_Q + \sum_{i=1}^{M} C_i\delta + \sum_{i=Q+1}^{M} P_i & a(s) = 0, \quad \epsilon = A_r \\ E_Q + \sum_{i=1}^{M} C_i z_i\delta + \sum_{i=Q+1}^{M} P_i z_i & a(s) = -1, \quad \epsilon = D \\ E_Q + \sum_{i=1}^{M} C_i z_i\delta + \sum_{i=Q+1}^{M} P_i z_i + C_{ij}\delta & \begin{cases} a(s) = R, \quad \epsilon = A_r, \\ j = 1, ..., Q, \sum_{i=1}^{Q} V_{ij} \leq n. \end{cases} \\ E_Q + \sum_{i=1}^{Q} C_i\delta + C_{ij}\delta & \begin{cases} a(s) = P, \quad \epsilon = A_p, \\ jj = 1, ..., Q, \sum_{i=1}^{Q} V_{ij} > n. \end{cases} \end{cases} \tag{5.17}$$

5.5.1 Discounted cost model

Let us consider $\tau_n = t_n - t_{n-1}$ the n^{th} time of stay, between two successive instants of transition, and a_n is the decision taken at the moment t_n to get the moment t_{n+1}.
For a given policy π, each instant t, a cost $c(s_t, a_t)$ is assumed and is defined by :

$$c(s_t, a_t) = r(s_t, a_t)\tau(s_t, a_t) \tag{5.18}$$

$(s_t)_{t\in\mathbb{R}^+}$ is a Markov process of decision making for state spaces. $F = \mathbb{N}^M$ and actions $A = \{-1, 0, P, R\}$.

$$V\delta(s, \pi) = E_\pi \left[\int_0^\infty e^{-\delta t} c\left(s_t, a_t\right) dt / s_0 = s \right] \tag{5.19}$$

where E_π is the Mean under the policy π, $\delta > 0$ and s is an initial state of the system. the optimal policy exists because the cost function is positive so the set of actions is finite; according to assumption 2 of the following proposition (Existence of the optimal policy) [112].

5.5.2 Discretization of the problem

In this section, we're going to look for a transformation of the problem into an equivalent dynamic programming problem in discrete-time as the normalization procedure[113].
let $\nu = \lambda_p + \lambda_s + Q\mu_p + (M - Q)\mu_s$.
We define $0 < t_0 < t_1 < \ldots < t_n < \ldots$ as the moments of transition in the state of the system. The time intervals are independents and exponentially distributed:

$$P\left(t_{k+1} - t_k > t\right) = e^{-t\nu} \quad k = 0, 1, 2, \ldots \tag{5.20}$$

Proposition 5.1. *The cost $V^{\delta}(s, \pi)$ for any policy π and any initial condition s is :*

$$E_{\pi}\left(\int_0^{+\infty} e^{-\delta t} c\left(s_t, a_t\right) dt\right) = \frac{1}{\delta + \nu} \sum_{n=0}^{+\infty} \left(\frac{\nu}{\delta + \nu}\right)^n E_{\pi}\left(c\right)\left(s_n, a_n\right)) \tag{5.21}$$

where $s_n = s_{t_n}$ et $a_n = a_{t_n}$.

Proof. For any couple (s, a), and any policy π and any initial condition s, the cost $V^{\delta}(s, \pi)$ is :

$$\begin{array}{l} E_{\pi}\left(\int_0^{+\infty} e^{-\delta t} c\left(s_t, a_t\right) dt\right) \\ = E_{\pi}\left(\sum_{n=0}^{+\infty} \int_{t_n}^{t_{n+1}} e^{-\delta t} c\left(s_t, a_t\right) dt\right) \\ = \sum_{n=0}^{+\infty} E_{\pi}\left(\int_{t_n}^{t_{n+1}} e^{-\delta t} c\left(s_n, a_n\right) dt\right) \\ = \sum_{n=0}^{+\infty} E_{\pi}\left(\int_{t_n}^{t_{n+1}} e^{-\delta t} dt\right) E_{\pi}\left(c\left(s_n, a_n\right)\right) \end{array} \tag{5.22}$$

The equality (5.22) results from the fact that the sequence of states are independents of the stay times because the policy π is deterministic: it does not depend on τ_n.
So, we have got

$$\begin{array}{l} E_{\pi}\left(\int_0^{+\infty} e^{-\delta t} c\left(x_t, a_t\right) dt\right) \\ = \sum_{n=0}^{+\infty} E_{\pi}\left(\int_{t_n}^{t_{n+1}} e^{-\delta t} dt\right) E_{\pi}\left(c\left(s_n, a_n\right)\right) \end{array} \tag{5.23}$$

Since the variables τ_n are independents, so:

$$\begin{array}{ll} E_{\pi}\left(\int_{t_n}^{t_{n+1}} e^{-\delta t}\right) & = E_{\pi}\left(e^{-\delta t_n} \int_0^{\tau_{n+1}} e^{-\delta t} dt\right) \\ & = \frac{1}{\delta} E_{\pi}\left(e^{-\delta t_n}\right)\left(1 - E_{\pi}\left(e^{-\delta \tau_{n+1}}\right)\right) \\ & = \frac{1}{\delta}\left(\frac{\nu}{\delta+\nu}\right)^n \left(1 - \frac{\nu}{\delta+\nu}\right) \end{array} \tag{5.24}$$

Indeed, for all

$$n \geq 0, \tau_{n+1} \sim \exp(\nu) \Longrightarrow E_{\pi}\left(e^{-\delta \tau_{n+1}}\right) = \frac{\nu}{\delta + \nu}$$

and

$$\begin{array}{ll} E_{\pi}\left(e^{-\delta t_n}\right) & = E_{\pi}\left(e^{-\delta\left(\sum_{k=1}^{n} t_k - t_{k-1}\right)}\right) \\ & = \prod_{k=1}^{n} E_{\pi}\left(e^{-\delta \tau_k}\right) = \left(\frac{\nu}{\delta+\nu}\right)^n \end{array} \tag{5.25}$$

hence

$$\begin{array}{l} E_\pi \left(\int_0^{+\infty} e^{-\delta t} c\left(s_t, a_t\right) dt \right) \\ = \frac{1}{\delta+\nu} \sum_{n=0}^{+\infty} \left(\frac{\nu}{\delta+\nu} \right)^n E_\pi \left(c\left(s_n, a_n\right) \right) \end{array} \tag{5.26}$$

■

Based on this proposal, we write:

$$V^\delta(s,\pi) = \frac{1}{\delta+\nu} E_\pi \left(\sum_{n=0}^{+\infty} \left(\frac{\nu}{\delta+\nu} \right)^n c\left(s_n, a_n\right) / s_0 = s \right) \tag{5.27}$$

which is an expression of the discrete-time discounted cost under the policy π, with the discount rate $\beta = \frac{\nu}{\delta+\nu}, (0 < \beta < 1)$ and the cost function $\frac{c(s_n;a_n)}{\delta+\nu}$. Then the cost is defined by:

$$V_N^\beta(s,\pi) = E_\pi \left(\sum_{n=0}^{N-1} \beta^n c\left(s_n, a_n\right) / s_0 = s \right) \tag{5.28}$$

and the cost β-discounted for infinite horizon is defined by :

$$V^\beta(s,\pi) = E_\pi \left(\sum_{n=0}^{+\infty} \beta^n c\left(s_n, a_n\right) / s_0 = s \right) \tag{5.29}$$

5.5.3 Existence of the optimal policy

Proposition 5.2.

i.) For all $s \in F$, we have:

$$\begin{cases} V_N^\beta(s) = \min\limits_{a \in A(s)} \left\{ c(s,a) + \beta \sum\limits_{\bar{s} \in F} p(\bar{s}/s,a) V_{N-1}^\beta(\bar{s}) \right\}, \forall N \geq 1. \\ V_0(\cdot) = 0 \end{cases}$$

ii.) If the cost is positive and the set of actions is finite, then:

$$\lim_{N \to +\infty} V_N^\beta(s) = V_\infty^\beta(s) = V^\beta(s)$$

and V^β so the only solution is given by the following equation:

$$V^\beta(s) = \min_{a \in A(s)} \left\{ c(s,a) + \beta \sum_{\bar{s} \in F} P\left(\bar{s}/s,a\right) V^\beta(\bar{s}) \right\}$$

For the demonstration of this proposition, see [112, 113].

5.6 Numerical results and analysis

In this section, the suggested scheme (SMDP) is compared with other schemes and the numerical results are obtained and then discussed to support our performance analysis. In the meantime, the principal parameters of our analysis are given in Table 5.2.

Parameter	K	M	n	λ_p	λ_s	μ_p	μ_s	P_i	e_j
Value	64	16	8-32	1-9	7	8	6	40W	15W

Table 5.2 – System parameters

In order to compare, we evaluate the performance of the following two reference schemes.

Greedy Algorithm (GA) scheme: This algorithm is designed to maximize the reward of the system at each moment in any decision [114].

Location Precedence (LP) scheme:which is a heuristic algorithm that applies a priority policy to send requests to the MPM or RPM. The LP scheme always assigned the more virtual machines in a physical machine that can support [112].

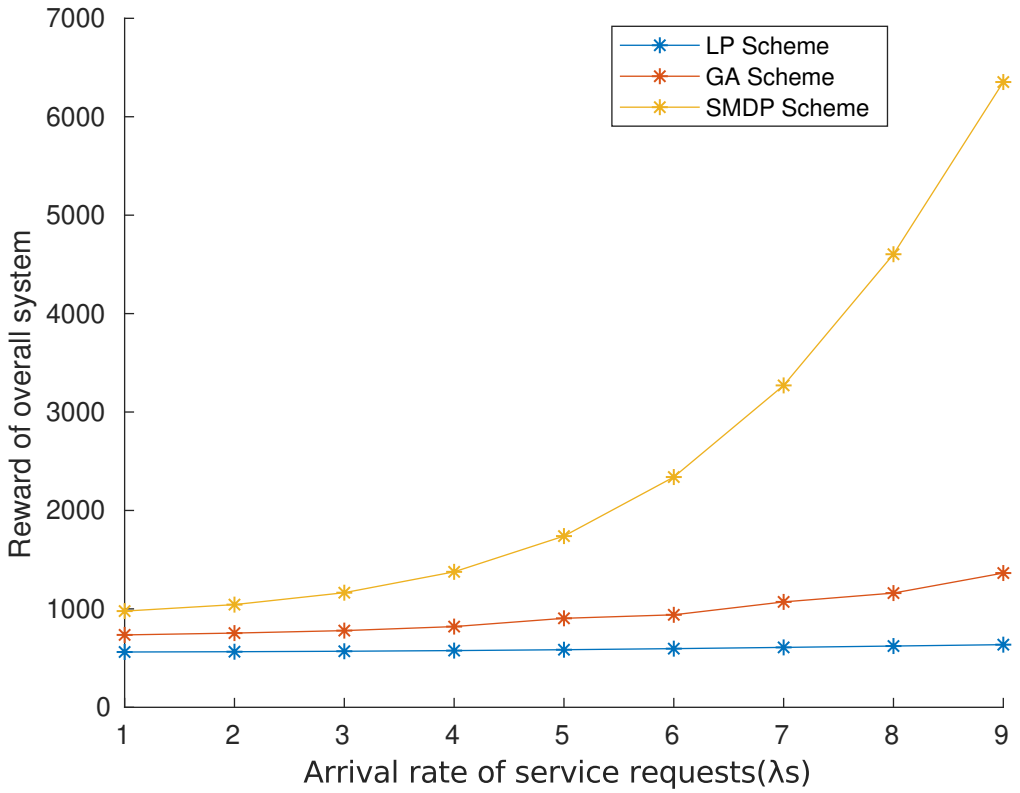

Figure 5.2 – System reward under different arrival rates for studied schemes.

The system reward under several schemes is illustrated in figure 5.2. When λ_p is low, the differences between the schemes are slight, because the virtual machines in the MPM and RPM are sufficient. However, with the increase of λ_p, the reward of all schemes increases gradually and the benefit of the

SMDP scheme becomes more evident. Besides, the LP scheme is less efficient, while the SMDP scheme is more efficient than the GA scheme. This is because the LP scheme does not consider the state of the reserves at all. However, the SMDP scheme sends the request to the RPM, which reduces the workload of the MPM in order to improve the quality of service expected from the system.

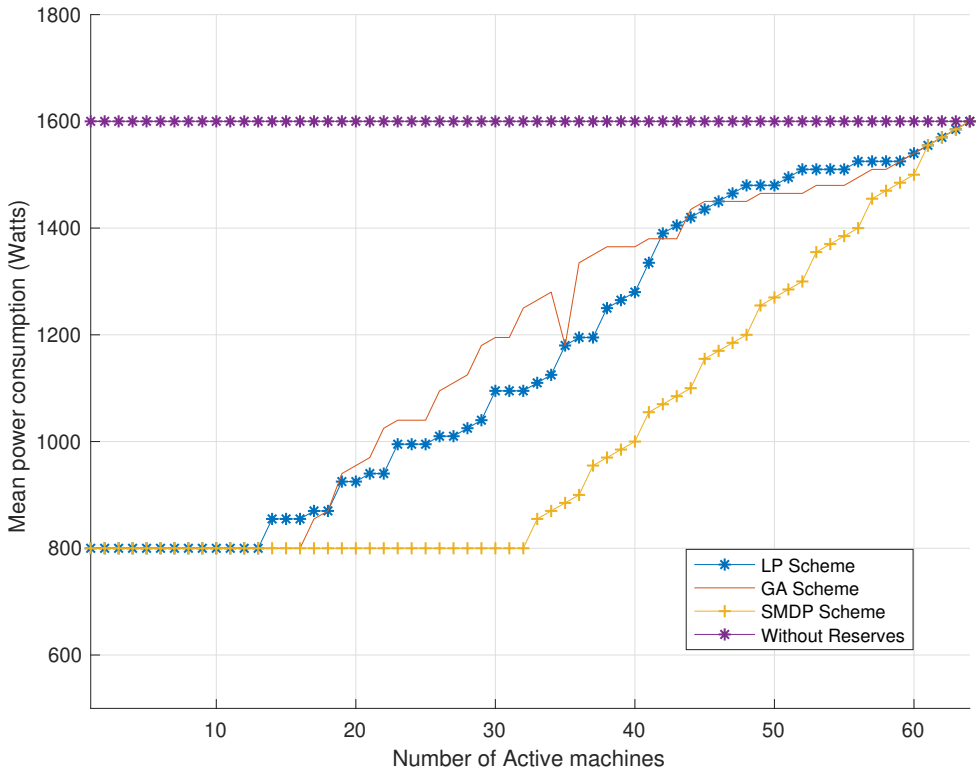

Figure 5.3 – System energy consumption of each scheme under different number of active machines .

In figure 5.3, we present the mean energy consumption in the system in relation to the number of activated machines. For all the three schemes, the system starts with a minimum consumption E_Q equal to 800 W. According to the decisions made by the algorithms GA and LP to use a new machine among the reserves, their energy consumption increases rapidly, because even if a machine is not fully loaded, the algorithm prefers to use a new machine for new customers until all machines are activated and then it recycles the possibility to send the incoming customers to the machines that still have the capacity. For our SMDP approach we observe that the energy consumption remains minimal by using an optimal number of MPMs, after that the controller decides to use the RPMs one by one until all machines are activated. The fourth graph shows the energy consumption without using the concept of reserve machines and it is clear that the consumption is high from the start. From this, we can say that our algorithm can conserve more energy consumption over time compared to other approaches.

Figure 5.4 presents the probability to send new clients to MPM for the three schemes. For our SMDP scheme we notice that the probability that the controller uses the main machines until the system has n clients is equal to 1. That means that the only decision possible for the controller is to use the MPM until he gets n clients. After that, the possibility to send the customers to any MPM is decreased

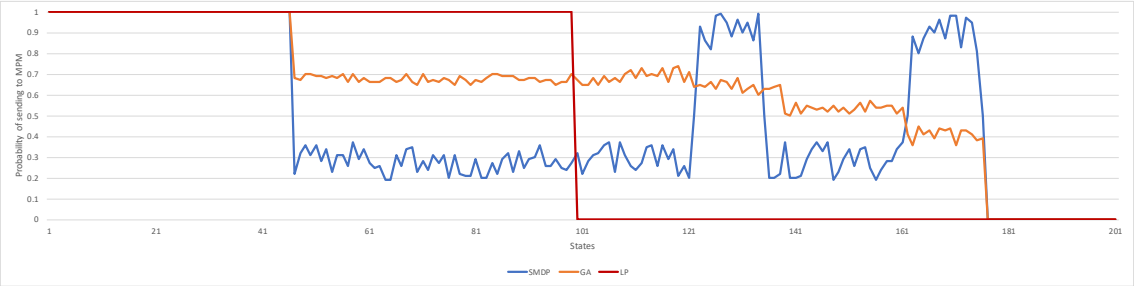

Figure 5.4 – The probability decision for sending the arrivals of customers to MPM with SMDP, GA and LP Schemes.

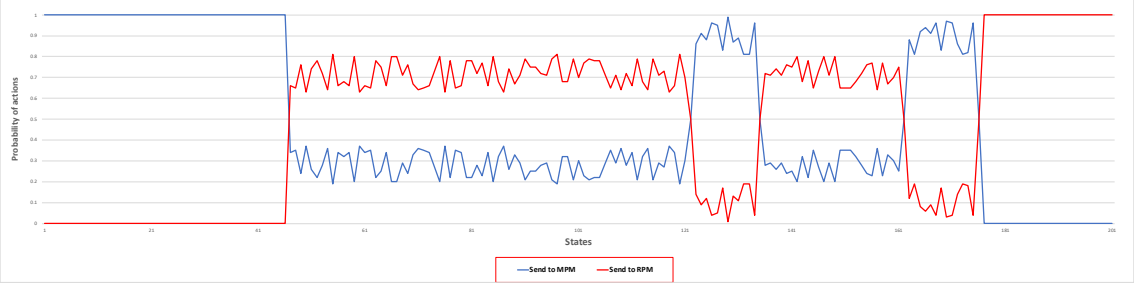

Figure 5.5 – The optimal policy for controlling the arrivals of customers with LP Scheme.

nearly to 0.3, which means that the probability to use a machine among the reserves is increased to 0.6. This means that the controller prefer to use new machine from RPM to ensure the QoS rather than use only the MPM to save energy. When all the RPM are busy and the customers still arrive, the probability to send uniformly to one of the PM is increased until all the MPM are full and then the probability to send to any MPM is equal to 0. For the other two algorithms their probabilities decision results are presented in the same figure (orange color and red color). It is clear that the GA algorithm is not stable; it means that it is zigzag between the choices of decision to maximize the instantaneous cost of the system until all the MPM are full. For the LP scheme, it is noted that it has no interest to use new machines reserved, except the already used, until all MPM are filled and that is obvious in the figure as the probability to send to MPM equal 1 when there is all MPM are full and then equal to 0.

For the probability to send to RPM for the three schemes, it is the compliment of the probabilities to send to PMP $(1 - p)$ for each scheme presented in figure 5.4.

The probabilities of decision to make each one from the two possible actions for our proposed scheme are presented in figure 5.5. This latter shows that before having n Customers, the obvious decision is to choose the main MPM machines with a probability equal to 1 and to use an RPM with a probability equal to 0. After, when the system has n clients the controller allows the possibility to turn on the reserve machines, in this case we notice that the probability of sending a new client to RPM is increasing. On the contrary the probability of sending clients to MPM is decreasing until a certain number of clients (when all MPM and RPM machines have at least $l * Q$ clients) where the controller chooses one more time to use the MPMs depending on the state of the system and taking into account the QoS. Towards the end, the probability to send to RPM is 1 and to send to MPM is 0, and this is normal because all the MPMs are busy and it only remains the possibility to install the VMs in RPM.

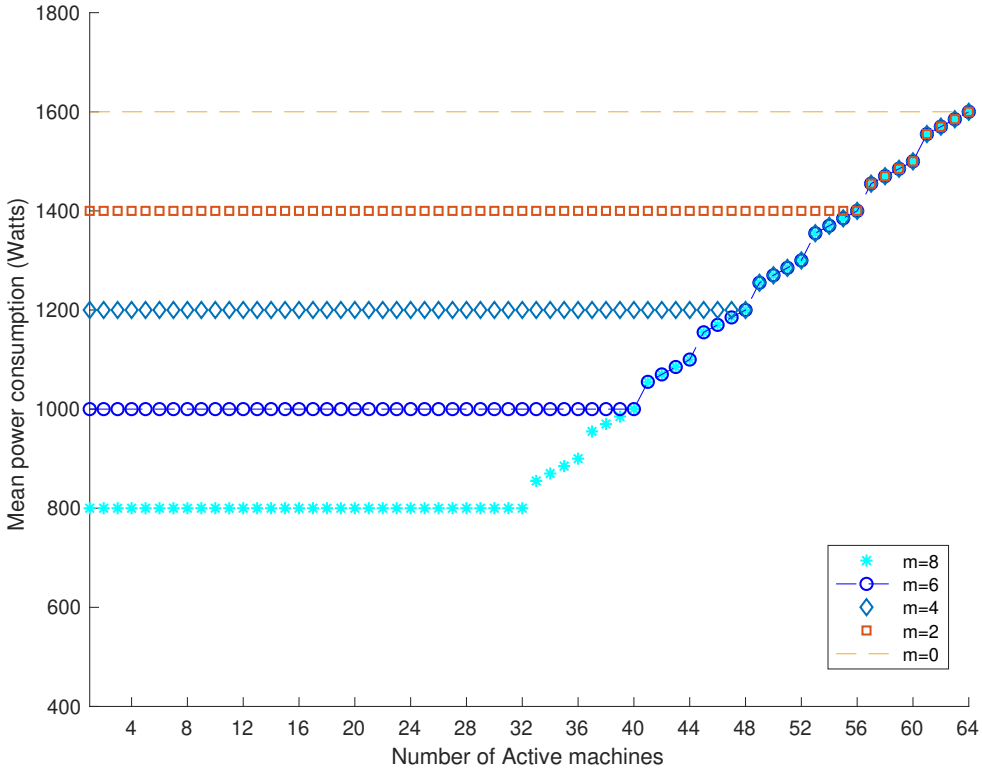

Figure 5.6 – Means power consumption curves as function of number of reserves .

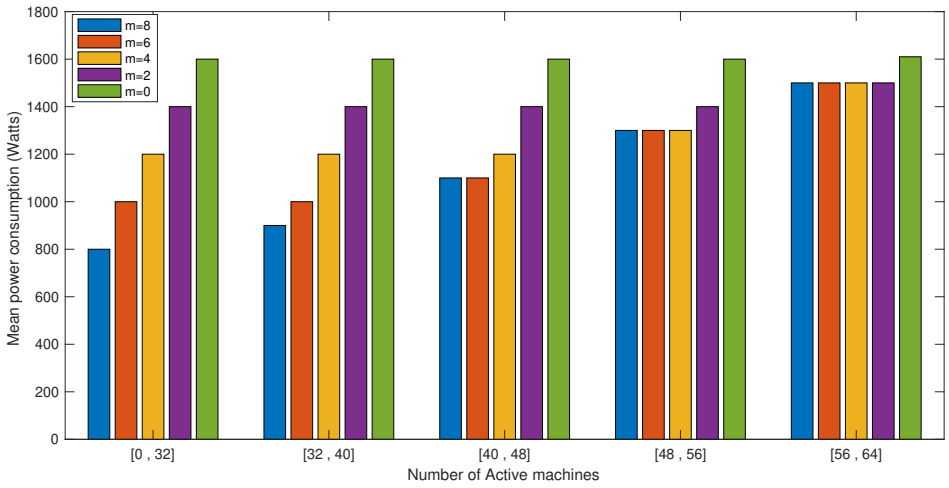

Figure 5.7 – Means power consumption curves as function of set of active machines .

Figure 5.6 shows the variation in the system's power consumption when the number of reserves $m = M - Q$ is changed and compares it to the static mode when all MPs are executed for the first time (there are no reservations). This figure shows that the last case is the worst in terms of energy consumption when all particles are running, even if they are not necessary. In addition, energy consumption becomes lower when a few PM are left in reserve. As shown in the figure, we notice that

when the number of reserves increases, the energy use is saved, and for example, when we use 8 PMs as reserves, we can save up to 50% of energy (with 32 customers) and 37.5% when we reserve 6 PMs and 12.5% when we reserve 2PMs. This reservation of PM leads to a reduction in energy consumption and optimal use of resources.

The variation in energy consumption as a function of the set of active VMs is illustrated in figure 5.7, this variation is presented in the form of intervals which present the role numbers of machines reserved in the system.

This figure confirms the effect of the number of active machines on system performance as well as the effect of the reservation of physical machines on energy consumption. But the most important fact deduced from this figure is that the number of RPMs has a good effect on the performance when the system load is not very large. However, the more the number of active machines increases the more the energy consumption tends to become independent of the number of reserved physical machines. This can be explained by the fact that, in order to ensure good performance, and when the number of requests increases, the system is forced to put RPMs in on mode, and these RPMs consume energy in the same way as the MPMs, which increases the energy consumption. In the case, without reserves (the green bar) the consumption remains high even if we have only one customer.

5.7 Conclusion

The mapping of virtual machines to physical machines in a cloud computing system poses several challenges due to the dynamic nature of such an environment and to energy and performance constraints in the system. In this chapter, we have been interested in modeling and evaluating the performance of a cloud computing system where some physical machines were considered as reserves and a controller turn them on one by one under some conditions. The optimal resource allocation problem with the objective of maximizing the expected reward is formulated as a Markov Decision Process problem. The optimal policy for controlling customer arrivals has been proven by the theory of dynamic programming. The numerical results show that the proposed scheme significantly outperforms all the considered schemes and improves the expected reward. Moreover, the proposed approach can significantly reduce the energy consumption of the system. These results are demonstrated in various scenarios. In perspective, it will be more interesting to be able to determine the optimal number of physical machines that should be reserved for a given system workload.

Conclusion

Contents

6.1 Summary of research contributions

Cloud computing is growing rapidly and is becoming one of the most important paradigms in the world dominant in the IT environment. The Infrastructures offering the services and the number and size of computing systems are becoming more and more bigger and larger to respond to the needs of the consumer and the business world. This increase leads to a growing demand for decentralized services. Obviously, that leads issues such as energy consumption, or the efficient use of energy and other resources. Consequently, techniques and tools must be developed to respond to these new needs.

In this thesis, several mathematical models based stochastic decision and Queuing theory were proposed. The idea is to help the provider of cloud computing to manage their resources. For this purpose we have attempted to solve some aspects of the problem by focusing on the management of virtual machines, more specifically on the allocation and reallocation of virtual machines.
In Chapter 3 we introduced a model of VM reallocation based on a combination of linear constraints. This model is based on double thresholds that manage the selection and placement of VMs in physical machines using both CM and HP policies related to resource usage thresholds. The proposed approaches are divided into two phases. The first phase called "selection phase", it allows to select a set of VMs to migrate them. The second phase is called "allocation phase", in this phase an allocation of the list of VMs is applied. The CM and HP algorithms are based on the upper and lower thresholds of CPU usage. The choice of the different CPU usage thresholds is fixed in the CM and HP approach. Our energy minimization policies solve the problem of Upper and Lower utilization of server resources. This reduces power consumption and also ensures a good quality of service described as execution time.

In chapter 4 we are focused on an M/M/k queue system with setup costs M/M/k/Setup. The servers are divided in two types: in run mode and in sleep mode as a reserve when there is no work to be done. The setup cost is in the form of a delay. Since a reserve server consume a lot of energy, the number of servers that can be as a reserve at any time is often limited. In

the configuration model, a maximum of one server can be configured at any time. We provided the first analytical expressions of the closed forms for the average response time, limiting distribution of the number of tasks in the system M/M/k/Setups. The proposed mechanism is studied using a mathematical analysis and the system's parameters are derived. In addition, the system's energy consumption within the suggested mechanisms is evaluated.

The mapping of virtual machines to physical machines in a cloud computing system poses several challenges due to the dynamic nature of such an environment and to energy and performance constraints in the system. In chapter 5, we have been interested in modeling and evaluating the performance of a cloud computing system where some physical machines were considered as reserves and a controller turn them on one by one under some conditions. The optimal resource allocation problem with the objective of maximizing the expected reward is formulated as a Markov Decision Process problem. The optimal policy for controlling customer arrivals has been proven by the theory of dynamic programming.

6.2 Further work

The works in this thesis had taken a small step towards modeling dynamic management resources in Cloud computing environments. However, during the preparation and writing of this thesis, we realized that many of the presented works open up new research questions and perspectives that require more attention. Some of these research tracks are:

- Add the dynamism of threshold to control the release of heat by physical machines in order to minimize the energy consumption of servers and cooling systems.
- Consider other types of resources in the two phases of selection and allocation, such as storage capacity and bandwidth.
- Study of the migration time will be interesting. The migration time plays a crucial role problems related to the reallocation of virtual machines. Test the impact of the migration time, with the impact of the virtual machine behavior and how implement during the migration, is necessary to adapt the algorithms, to be able to take some decision in an appropriate manner.
- Implement the two proposed approaches developed in chapter 3, in a real Cloud environment.
- One could think of extending the present model by introducing K blocks of reserves, of sizes $m_1, m_2, \ldots, m_K$, with associated upper and lower thresholds. Reserves 1 are powered ON when the queue passes level U_1, reserves 2 are powered ON when the queue passes level $U_2 > U_1$, etc. That would be an interesting topic for future work.
- Determine the optimal number of physical machines that should be reserved for a given system workload.

Thesis publications

Publications in journals

- Abdellah Ouammou, Abdellah Zaaloul, Mohamed Hanini, and Abdelghani BenTahar.*" Modeling Decision Making to Control the Allocation of Virtual Machines in a Cloud Computing System with Reserve Machines "*. IAENG International Journal of Computer Science, vol. 48, no.2, pp285-293, 2021.

- Abdellah Ouammou, Abdelghani Ben Tahar, Mohamed Hanini, and Said El Kafhali. *"Modeling and analysis of quality of service and energy consumption in cloud environment"*. International Journal of Computer Information Systems and Industrial Management Applications, ISSN 2150-7988, vol.10, (2018),pp. 98–106.

- Abdellah Ouammou, Mohamed Hanini, Abdelghani BenTahar, and Said El Kafhali.*"A dynamic programming approach to manage virtual machines allocation in cloud computing"*. International Journal of Engineering & Technology, ISSN: 2227-524X, vol.7, No 4.6 (2018),pp. 128-132.

Publications in international conferences and workshops

- Abdellah Ouammou, Abdelghani Ben Tahar, Mohamed Hanini, Said El Kafhali , *" Analysis of a M/M/k system with exponential setup times and reserves servers"* BDIoT'19: Proceedings of the 4th International Conference on Big Data and Internet of Things October 2019 Article No.:58 Pages 1–5

- ***Poster:*** Abdellah Ouammou, Abdelghani Ben Tahar, Mohamed Hanini *"Analytical approach to evaluate the impact of uncertainty in VM placement in cloud computing"*. First Winter School on Complex System Modeling and Simulation, Ben Guerir, Morocco (February 13-15, 2018)

- Abdellah Ouammou, Mohamed Hanini, Said El Kafhali, and Abdelghani BenTahar. *" Energy consumption and cost analysis for data centers with workload control"*. In International Conference on Innovations in Bio-Inspired Computing and Applications, pages 92–101. Springer, 2017

Participations in national and international scientific events

- ***" Participation in Winter School on Mathematical Statistics"***. University of Luxembourg, (7-10 December 2020)
- ***" Participation in the workshop and conference International Conference Dynamic, Games and Science CIRM (Centre International de Rencontres"***. CIRM (Centre International de Rencontres Mathématiques),Marseille, France (3-7 June 2019)
- ***" Participation in Probability days"***. CNRS and Polytechnic Institute of Paris, France. (24-28 June 2019)
- ***" Effective Control of Migration to Improve Quality of Service and Reduce Energy Consumption in Cloud Computing"***. The 6th edition of the "DOCTOR'S DAY", Settat, Morocco (April 05-2018)
- ***" Participation in the Spring School (CIMPA'17) under the theme Algebraic, combinatorial and analytic aspects of free probabilities"***.Faculty of Science and Technology, Settat, Morocco (17-28 April 2017)
- ***"Energy consumption analysis for data centers with workload control Scientific Day: Digital Methods and Decision Support"***, Khouribga, Morocco (May 20-2017)

Bibliography

[1] Richard S Sutton and Andrew G Barto. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999. 29

[2] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, 19:399–468, 2003. 33

[3] István Szita and András Lorincz. Factored value iteration converges. *arXiv preprint arXiv:0801.2069*, 2008. 34

[4] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009. 37

[5] Md Anit Khan, Andrew Paplinski, Abdul Malik Khan, Manzur Murshed, and Rajkumar Buyya. Dynamic virtual machine consolidation algorithms for energy-efficient cloud resource management: a review. *Sustainable cloud and energy services*, pages 135–165, 2018. 42

[6] Aaqib Rashid and Amit Chaturvedi. A study on resource pooling, allocation and virtualization tools used for cloud computing. *International Journal of Computer Applications*, 975:8887, 2017. 43

[7] Les différents types de virtualisation https://wapiti.telecom-lille.fr/commun/ens/peda/options/st/rio/pub/exposes/exposesrio2009/colin-desbureaux/la-virtualisation-diff%c3%a9rents-types-de-virtualisation.html, accessed October 29, 2020. 43

[8] *La virtualisations http://blog.usense.fr/la-virtualisation.html/ accessed October 29, 2020.* 44, 45

[9] *What's the difference between type 1 and type 2 hypervisors http://searchservervirtualization.techtarget.com/feature/Whats-the-difference-between-Type-1-and-Type-2-hypervisors accessed October 29, 2020.* 46

[10] *Ni real-time hypervisor architecture and performance details http://www.ni.com/white-paper/9629/en/ accessed October 29, 2020.* 46

[11] *Survey on hypervisors* http://salsahpc.indiana.edu/b534projects/sites,/default/files/public/6_Survey%20On%20Hypervisors_Alam%20Naveed%20Imran *accessed October 29, 2020.* 46

[12] *Les avantages de la virtualisation* http://www.agentsolo.com/ca/fr/fbsc/les-avantages-de-la-virtualisation *accessed October 29, 2020.* 46

[13] Lalithabhinaya Mallu and R Ezhilarasie. Live migration of virtual machines in cloud environment: A survey. *Indian Journal of Science and Technology*, 8(S9):326–332, 2015. 48

[14] Pradip D Patel, Miren Karamta, MD Bhavsar, and MB Potdar. Live virtual machine migration techniques in cloud computing: A survey. *International Journal of Computer Applications*, 86(16), 2014. 49, 51

[15] Rinal M Chawda1 Ompriya Kale. Virtual machine migration techniques in cloud environment: A survey. *International Journal for Scientific Research and Development*, 1(8), 2013. 49, 50

[16] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. Energy aware consolidation for cloud computing. In *USENIX Workshop on Power-Aware Computing and Systems*, 2008. 51, 62

[17] Ripal Nathuji, Aman Kansal, and Alireza Ghaffarkhah. Q-clouds: managing performance interference effects for qos-aware clouds. In *Proceedings of the 5th European conference on Computer systems*, pages 237–250, 2010. 52

[18] Toni Mastelic, Ariel Oleksiak, Holger Claussen, Ivona Brandic, Jean-Marc Pierson, and Athanasios V Vasilakos. Cloud computing: Survey on energy efficiency. *Acm computing surveys (csur)*, 47(2):1–36, 2014. 53

[19] Gihun Jung and Kwang Mong Sim. Location-aware dynamic resource allocation model for cloud computing environment. In *International Conference on Information and Computer Applications (ICICA), IACSIT Press, Singapore*. Citeseer, 2012. 53

[20] Vinicius Petrucci, Enrique V Carrera, Orlando Loques, Julius CB Leite, and Daniel Mosse. Optimized management of power and performance for virtualized heterogeneous server clusters. In *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 23–32. IEEE, 2011. 53

[21] Hariharasudhan Viswanathan, Eun Kyung Lee, Ivan Rodero, Dario Pompili, Manish Parashar, and Marc Gamell. Energy-aware application-centric vm allocation for hpc workloads. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pages 890–897. IEEE, 2011. 53

[22] Vlasia Anagnostopoulou, Heba Saadeldeen, and Frederic T Chong. Quantifying the environmental advantages of large-scale computing. In *International Conference on Green Computing*, pages 269–280. IEEE, 2010. 54

[23] Dmytro Dyachuk and Michele Mazzucco. On allocation policies for power and performance. In *2010 11th IEEE/ACM International Conference on Grid Computing*, pages 313–320. IEEE, 2010. 54

[24] Liang Liu, Hao Wang, Xue Liu, Xing Jin, Wen Bo He, Qing Bo Wang, and Ying Chen. Greencloud: a new architecture for green data center. In *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*, pages 29–38, 2009. 54

[25] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012. 54, 63

[26] Young Choon Lee and Albert Y Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280, 2012. 55, 101

[27] Yacine Kessaci, Nouredine Melab, and El-Ghazali Talbi. Optimisation multi-critère pour l'allocation de ressources sur clouds distribués avec prise en compte de l'énergie. In *Rencontres Scientifiques France Grilles 2011*, 2011. 55

[28] Anton Beloglazov and Rajkumar Buyya. Energy efficient resource management in virtualized cloud data centers. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 826–831. IEEE, 2010. 56

[29] Jyothi Sekhar, Getzi Jeba, and S Durga. A survey on energy efficient server consolidation through vm live migration. *International Journal of Advances in Engineering & Technology*, 5(1):515, 2012. 56

[30] Laurent Lefèvre and Anne-Cécile Orgerie. Designing and evaluating an energy efficient cloud. *The Journal of Supercomputing*, 51(3):352–373, 2010. 56

[31] Jian Cao, Yihua Wu, and Minglu Li. Energy efficient allocation of virtual machines in cloud computing environments based on demand forecast. In *International Conference on Grid and Pervasive Computing*, pages 137–151. Springer, 2012. 57

[32] Richa Sinha, Nidhi Purohit, and Hiteishi Diwanji. Energy efficient dynamic integration of thresholds for migration at cloud data centers. *IJCA Special Issue on Communication and Networks*, 1:44–49, 2011. 57

[33] Anshul Gandhi, Mor Harchol-Balter, and Ivo Adan. Server farms with setup costs. *Performance Evaluation*, 67(11):1123–1138, 2010. 57, 79, 80, 101

[34] Said El Kafhali and Khaled Salah. Performance analysis of multi-core vms hosting cloud saas applications. *Computer Standards & Interfaces*, 55:126–135, 2018. 58, 63, 101

[35] Young Choon Lee and Albert Y Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280, 2012. 58

[36] Mohamed Hanini, Said El Kafhali, and Khaled Salah. Dynamic vm allocation and traffic control to manage qos and energy consumption in cloud computing environment. *International Journal of Computer Applications in Technology*, 60(4):307–316, 2019. 58, 63, 101

[37] Isi Mitrani. Managing performance and power consumption in a server farm. *Annals of Operations Research*, 202(1):121–134, 2013. 60, 80, 101

[38] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf. Nist cloud computing reference architecture. *NIST special publication*, 500(2011):1–28, 2011. 62

[39] Said El Kafhali and Khaled Salah. Stochastic modelling and analysis of cloud computing data center. In *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, pages 122–126. IEEE, 2017. 62

[40] Jonathan G Koomey. Estimating total power consumption by servers in the us and the world, 2007. *Lawrence Berkeley National Laboratory, Berkeley, CA*, 2007. 62

[41] MP Mills. The cloud begins with coal–big data, big networks, big infrastructure, and big power, digital power group, 2013. 62

[42] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. Data center energy consumption modeling: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):732–794, 2015. 62

[43] Dara Kusic, Jeffrey O Kephart, James E Hanson, Nagarajan Kandasamy, and Guofei Jiang. Power and performance management of virtualized computing environments via lookahead control. *Cluster computing*, 12(1):1–15, 2009. 62

[44] Said El Kafhali and Khaled Salah. Modeling and analysis of performance and energy consumption in cloud data centers. *Arabian Journal for Science and Engineering*, 43(12):7789–7802, 2018. 62, 78

[45] Tamojit Chatterjee, Varun Kumar Ojha, Mainak Adhikari, Sourav Banerjee, Utpal Biswas, and Václav Snášel. Design and implementation of an improved datacenter broker policy to improve the qos of a cloud. In *Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014*, pages 281–290. Springer, 2014. 62

[46] Abdellah Ouammou, Abdelghani BenTahar, Mohamed Hanini, and Said El Kafhali. Modeling and analysis of quality of service and energy consumption in cloud environment. *International Journal of Computer Information Systems and Industrial Management Applications*, 10:098–106, 2018. 62, 78, 101, 103

[47] Abdellah Ouammou, Mohamed Hanini, Said El Kafhali, and Abdelghani Ben Tahar. Energy consumption and cost analysis for data centers with workload control. In *International Conference on Innovations in Bio-Inspired Computing and Applications*, pages 92–101. Springer, 2017. 62, 67

[48] Benjamin Speitkamp and Martin Bichler. A mathematical programming approach for server consolidation problems in virtualized data centers. *IEEE Transactions on services computing*, 3(4):266–278, 2010. 63

[49] Akshat Verma, Puneet Ahuja, and Anindya Neogi. pmapper: power and migration cost aware application placement in virtualized systems. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 243–264. Springer, 2008. 63

[50] Awada Uchechukwu, Keqiu Li, Yanming Shen, et al. Energy consumption in cloud computing data centers. *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, 3(3):145–162, 2014. 63

[51] Seyed Yahya Zahedi Fard, Mohamad Reza Ahmadi, and Sahar Adabi. A dynamic vm consolidation technique for qos and energy consumption in cloud environment. *The Journal of Supercomputing*, 73(10):4347–4368, 2017. 63

[52] Patricia Arroba, José M Moya, Jose L Ayala, and Rajkumar Buyya. Dynamic voltage and frequency scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers. *Concurrency and Computation: Practice and Experience*, 29(10):e4067, 2017. 63

[53] Dejene Boru, Dzmitry Kliazovich, Fabrizio Granelli, Pascal Bouvry, and Albert Y Zomaya. Energy-efficient data replication in cloud computing datacenters. *Cluster computing*, 18(1):385–402, 2015. 63

[54] Mohamed Hanini and Said El Kafhali. Cloud computing performance evaluation under dynamic resource utilization and traffic control. In *Proceedings of the 2nd international Conference on Big Data, Cloud and Applications*, pages 1–6, 2017. 63

[55] Khaled Salah, Khalid Elbadawi, and Raouf Boutaba. An analytical model for estimating cloud resources of elastic services. *Journal of Network and Systems Management*, 24(2):285–308, 2016. 63

[56] Quyet Thang Nguyen, Nguyen Quang-Hung, Nguyen Huynh Tuong, Van Hoai Tran, and Nam Thoai. Virtual machine allocation in cloud computing for minimizing total execution time on each machine. In *2013 International Conference on Computing, Management and Telecommunications (ComManTel)*, pages 241–245. IEEE, 2013. 66

[57] Sambit Kumar Mishra, Md Akram Khan, Bibhudatta Sahoo, Deepak Puthal, Mohammad S Obaidat, and Kuei-Fang Hsiao. Time efficient dynamic threshold-based load balancing technique for cloud computing. In *2017 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 161–165. IEEE, 2017. 72

[58] Ankita Choudhary, Shilpa Rana, and KJ Matahai. A critical analysis of energy efficient virtual machine placement techniques and its optimization in a cloud computing environment. *Procedia Computer Science*, 78:132–138, 2016. 78

[59] SM Ali, I Hussain, Z Ullah, I Sami, R Asghar, U Farid, B Khan, CA Mehmood, and A Haider. Need for mutual services interaction between smart grid and cloud data centers. In *2018 International Conference on Power Generation Systems and Renewable Energy Technologies (PGSRET)*, pages 1–5. IEEE, 2018. 78

[60] Xinqian Zhang, Tingming Wu, Mingsong Chen, Tongquan Wei, Junlong Zhou, Shiyan Hu, and Rajkumar Buyya. Energy-aware virtual machine allocation for cloud with resource reservation. *Journal of Systems and Software*, 147:147–161, 2019. 78, 101

[61] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. The datacenter as a computer: Designing warehouse-scale machines. *Synthesis Lectures on Computer Architecture*, 13(3):i–189, 2018. 78

[62] Anshul Gandhi, Mor Harchol-Balter, and Ivo Adan. Decomposition results for an m/m/k with staggered setup. *ACM SIGMETRICS Performance Evaluation Review*, 38(2):48–50, 2010. 78

[63] Mark W Storer, Kevin M Greenan, Ethan L Miller, and Kaladhar Voruganti. Pergamum: Replacing tape with energy efficient, reliable, disk-based archival storage. In *Fast*, volume 8, pages 1–16, 2008. 78, 82

[64] Abdellah Ouammou, Mohamed Hanini, Abdelghani Ben Tahar, and Said El Kafhali. Analysis of a m/m/k system with exponential setup times and reserves servers. In *Proceedings of the 4th International Conference on Big Data and Internet of Things*, pages 1–5, 2019. 78

[65] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007. 79

[66] Andrew Krioukov, Prashanth Mohan, Sara Alspaugh, Laura Keys, David Culler, and Randy H Katz. Napsac: Design and implementation of a power-proportional web cluster. In *Proceedings of the first ACM SIGCOMM workshop on Green networking*, pages 15–22, 2010. 79

[67] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's highly available key-value store. *ACM SIGOPS operating systems review*, 41(6):205–220, 2007. 79

[68] Wubi Qin and Qian Wang. Modeling and control design for performance management of web servers via an lpv approach. *IEEE Transactions on Control Systems Technology*, 15(2):259–275, 2007. 79

[69] Malu Castellanos, Fabio Casati, M-C Shan, and Umeshwar Dayal. ibom: A platform for intelligent business operation management. In *21st International Conference on Data Engineering (ICDE'05)*, pages 1084–1095. IEEE, 2005. 79

[70] Tibor Horvath and Kevin Skadron. Multi-mode energy management for multi-tier server clusters. In *2008 international conference on parallel architectures and compilation techniques (pact)*, pages 270–279. IEEE, 2008. 79

[71] Jihong Kim and Tajana Simunic Rosing. Power-aware resource management techniques for low-power embedded systems. In *of Design, Automation and Test in Europe (DATE'02)*, pages 788–794, 2002. 79

[72] Anshul Gandhi, Mor Harchol-Balter, and Michael A Kozuch. Are sleep states effective in data centers? In *2012 international green computing conference (IGCC)*, pages 1–10. IEEE, 2012. 79

[73] Anshul Gandhi and Mor Harchol-Balter. How data center size impacts the effectiveness of dynamic power management. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1164–1169. IEEE, 2011. 79

[74] Peter D Welch. On a generalized m/g/1 queuing process in which the first customer of each busy period receives exceptional service. *Operations Research*, 12(5):736–752, 1964. 79

[75] Guy Latouche and Vaidyanathan Ramaswami. *Introduction to matrix analytic methods in stochastic modeling*. SIAM, 1999. 79

[76] JSH Van Leeuwaarden and EMM Winands. Quasi-birth-and-death processes with an explicit rate matrix. *Stochastic models*, 22(1):77–98, 2006. 80

[77] Benny Van Houdt and Johan SH van Leeuwaarden. Triangular m/g/1-type and tree-like quasi-birth-death markov chains. *INFORMS Journal on Computing*, 23(1):165–171, 2011. 80

[78] Ivo Jean-Baptiste François Adan and Jacobus Adrianus Cornelis Resing. *A class of Markov processes on a semi-infinite strip*. Eindhoven University of Technology, Department of Mathematics and Computing ..., 1999. 80

[79] Zhe George Zhang and Naishuo Tian. Analysis on queueing systems with synchronous vacations of partial servers. *Performance Evaluation*, 52(4):269–282, 2003. 80

[80] Xiuli Xu and Naishuo Tian. The m/m/c queue with (e, d) setup time. *Journal of Systems Science and Complexity*, 21(3):446–455, 2008. 80

[81] Naishuo Tian, Quan-Lin Li, and Jinhua Gao. Conditional stochastic decompositions in the m/m/c queue with server vacations. *Stochastic Models*, 15(2):367–377, 1999. 80

[82] Yonatan Levy and Uri Yechiali. An m/m/s queue with servers' vacations. *INFOR: Information Systems and Operational Research*, 14(2):153–163, 1976. 80

[83] Ivo JBF Adan and Jan Van der Wal. Combining make to order and make to stock. *Operations-Research-Spektrum*, 20(2):73–81, 1998. 80

[84] Jesus R Artalejo, Antonis Economou, and Maria Jesus Lopez-Herrero. Analysis of a multiserver queue with setup times. *Queueing Systems*, 51(1):53–76, 2005. 80

[85] Anshul Gandhi, Sherwin Doroudi, Mor Harchol-Balter, and Alan Scheller-Wolf. Exact analysis of the m/m/k/setup class of markov chains via recursive renewal reward. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, pages 153–166, 2013. 80

[86] Susanne Albers and Hiroshi Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms (TALG)*, 3(4):49–es, 2007. 95

[87] Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs. Speed scaling with an arbitrary power function. *ACM Transactions on Algorithms (TALG)*, 9(2):1–14, 2013. 95

[88] Adam Wierman, Lachlan LH Andrew, and Ao Tang. Power-aware speed scaling in processor sharing systems. In *IEEE INFOCOM 2009*, pages 2007–2015. IEEE, 2009. 95

[89] Temidayo Abayomi-Zannu and Isaac Odun-Ayo. Cloud identity management–a critical analysis. In *Lecture Notes in Engineering and Computer Science: Proceedings of the International MultiConference of Engineers and Computer Scientists*, pages 13–15, 2019. 100

[90] Shahrzad Kananizadeh and Kirill Kononenko. Improving on linear scan register allocation. *International Journal of Automation and Computing*, 15(2):228–238, 2018. 100

[91] Xuanyu Liu, Yao Zhang, and Kaiju Zhang. Optimization control of energy consumption in tunneling system of earth pressure balance shield tunneling machine. *Engineering Letters*, 28(2), 2020. 100

[92] Hussein El Ghor and Maryline Chetto. Energy guarantee scheme for real-time systems with energy harvesting constraints. *International Journal of Automation and Computing*, 16(3):354–368, 2019. 100

[93] Usman Wajid, Cinzia Cappiello, Pierluigi Plebani, Barbara Pernici, Nikolay Mehandjiev, Monica Vitali, Michael Gienger, Kostas Kavoussanakis, David Margery, David Garcia Perez, et al. On achieving energy efficiency and reducing co 2 footprint in cloud computing. *IEEE transactions on cloud computing*, 4(2):138–151, 2015. 100

[94] Saad Mustafa, Babar Nazir, Amir Hayat, Sajjad A Madani, et al. Resource management in cloud computing: Taxonomy, prospects, and challenges. *Computers & Electrical Engineering*, 47:186–203, 2015. 100

[95] Dan Liu, Xin Sui, and Li Li. An energy-efficient virtual machine placement algorithm in cloud data center. In *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 719–723. IEEE, 2016. 100

[96] El Mehdi Kandoussi, Mohamed Hanini, Iman El Mir, and Abdelkrim Haqiq. Toward an integrated dynamic defense system for strategic detecting attacks in cloud networks using stochastic game. *Telecommunication Systems*, 73(3):397–417, 2020. 100

[97] Bruce Hajek. Optimal control of two interacting service stations. *IEEE transactions on automatic control*, 29(6):491–499, 1984. 100

[98] Abdellah Ouammou, Abdellah Zaaloul, Mohamed Hanini, and Abdelghani BenTahar. Modeling decision making to control the allocation of virtual machines in a cloud computing system with reserve machines. *IAENG International Journal of Computer Science*, 48(2):285–293, 2021. 101

[99] Abdellah Ouammou, Mohamed Hanini, Abdelghani BenTahar, and Said El Kafhali. A dynamic programming approach to manage virtual machines allocation in cloud computing. *International Journal of Engineering & Technology*, 7(4.6):128–132, 2018. 101, 104

[100] Isaac Odun-Ayo, Toro-Abasi Williams, Olusola Abayomi-Alli, and Jamaiah Yahaya. Systematic mapping study of economic and business models of cloud services. *Indonesian Journal of Electrical Engineering and Computer Science*, 18(2):987–994, 2020. 101

[101] Kan Zheng, Hanlin Meng, Periklis Chatzimisios, Lei Lei, and Xuemin Shen. An smdp-based resource allocation in vehicular cloud computing systems. *IEEE Transactions on Industrial Electronics*, 62(12):7920–7928, 2015. 101

[102] Hanlin Meng, Kan Zheng, Periklis Chatzimisios, Hui Zhao, and Lin Ma. A utility-based resource allocation scheme in cloud-assisted vehicular network architecture. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 1833–1838. IEEE, 2015. 101

[103] Said El Kafhali and Mohamed Hanini. Stochastic modeling and analysis of feedback control on the qos voip traffic in a single cell ieee 802.16 e networks. *IAENG International Journal of Computer Science*, 44(1):19–28, 2017. 101

[104] GB Bindu, K Ramani, and C Shoba Bindu. Optimized resource scheduling using the meta heuristic algorithm in cloud computing. *IAENG International Journal of Computer Science*, 47(3), 2020. 101

[105] Said El Kafhali and Khaled Salah. Efficient and dynamic scaling of fog nodes for iot devices. *The Journal of Supercomputing*, 73(12):5261–5284, 2017. 103

[106] Mikael Andersson, Anders Bengtsson, Martin Höst, and Christian Nyberg. Web server traffic in crisis conditions. In *In Proceedings of the Swedish National Computer Networking Workshop, SNCNW 2005*, 2005. 103

[107] Kaiqi Xiong and Harry Perros. Service performance and analysis in cloud computing. In *2009 Congress on Services-I*, pages 693–700. IEEE, 2009. 103

[108] Zhikui Wang, Yuan Chen, Daniel Gmach, Sharad Singhal, Brian J Watson, Wilson Rivera, Xiaoyun Zhu, and Chris D Hyser. Appraise: application-level performance management in virtualized server environments. *IEEE Transactions on Network and Service Management*, 6(4):240–254, 2009. 103

[109] Chao Wei. Almost sure exponential stability of nonlinear stochastic delayed systems with markovian switching and lévy noises. *IAENG International Journal of Applied Mathematics*, 49(3):1–8, 2019. 106

[110] Dongsheng Xu, Huaxiang Xian, Xiangxiang Cui, and Yanran Hong. A new single-valued neutrosophic distance for topsis, mabac and new similarity measure in multi-attribute decision-making. *IAENG International Journal of Applied Mathematics*, 50(1):1–8, 2020. 106

[111] Mona Nasseri, Mansoor Alam, and Robert C Green. Mdp based optimal policy for collaborative processing using mobile cloud computing. In *2013 IEEE 2nd international conference on cloud networking (CloudNet)*, pages 123–129. IEEE, 2013. 109

[112] Peter Whittle. *Optimal control: basics and beyond*. John Wiley & Sons, Inc., 1996. 109, 111, 112

[113] Sheldon M Ross. *Introduction to stochastic dynamic programming*. Academic press, 2014. 110, 111

[114] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009. 112