

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Blida -1-(Saad Dahleb)



Faculté des Sciences
Département d'Informatique

Mémoire de fin d'étude présenté en vue d'obtention du diplôme

MASTER EN INFORMATIQUE

Option : Systèmes Informatiques & Réseaux

Thème

An energy saving model for Cloud Computing

Présenté par :

**El hadj Khalaf Abderrahmane
Djemai Zakarya**

Devant le jury:

Ould-Khaoua Mohamed	(President of the jury)
Benyahia Mohamed	(Examinator)
Boutoumi Bachira	(Promoter)

The graduation year 2019/2020

Abstract

Cloud computing is getting more popular day by day. Nowadays, we have an energy shortage worldwide. Thus, more researches concern about finding ways to improve the energy efficiency and sustainability of cloud computing. This thesis presents a performance analysis of the cloud computing system based on the energy-efficient task scheduling strategy. This strategy consists of waking up threshold policy and a sleep delay timer.

To model the system in an energy-efficient task scheduling strategy, we use a vacation queue system with an N-policy threshold. In this work, the system is considered as a physical machine with a limited buffer, this physical machine contains a set of virtual machines that represent the servers in our modelization, as well we constructed a continuous-time Markov chain and an infinitesimal generator that allows us to calculate stationary distribution vector. After that, we analyze the steady-state system to calculate the system's different performance measures like energy consumption and the delay of the system.

We tested our system with different scenarios and with different parameter values to discover the proposed strategy performance and validate the system model according to the performance measure.

The experimental study has proven that the proposed solution improves energy efficiency and latency without affecting availability in the system.

Key words: Cloud Computing, energy efficiency, task scheduling, waking up threshold sleep delay, queue system, Markov chain

ملخص

تزداد شعبية الحوسبة السحابية يوماً بعد يوم. واليوم أصبح لدينا نقص في الطاقة في أنحاء العالم. وبالتالي، فإن المزيد من الأبحاث تهتم بإيجاد طرق لتحسين كفاءة استهلاك الطاقة واستدامة الحوسبة السحابية. تقدم هذه المذكرة تحليلاً للأداء لنظام الحوسبة السحابية بناءً على استراتيجية جدولة المهام من أجل توفير الطاقة. تتألف هذه الاستراتيجية من سياسة تنبيه الرجوع للخوادم إلى العمل ومؤقت تأخير الاطفاء الجزئي للخوادم.

لنمذجة النظام في استراتيجية جدولة مهام فعالة من حيث استهلاك الطاقة، نستخدم نظام قائمة انتظار تسمح باخذ عطل مع تنبيه الرجوع للخوادم. وفي هذا العمل، نعتبر النظام جهازاً فعلياً مزود بمخزن محدود، ويحتوي هذا النظام على مجموعة من الأجهزة الافتراضية التي تمثل الخوادم في النمذجة المتبعة، فضلاً عن ذلك فقد قمنا ببناء سلسلة ماركوف ذات الوقت المتواصل حتى يتسنى لنا حساب احتمالات الحالة الثابتة للنظام. وبعد ذلك، نقوم بتحليل نظام احتمالات الحالة الثابتة لحساب مقاييس الأداء المختلفة للنظام مثل استهلاك الطاقة و التأخيرات الحاصلة داخل النظام.

لقد قمنا باختبار نظامنا باستخدام سيناريوهات مختلفة وبقيم مختلفة لاكتشاف أداء الاستراتيجية المقترحة والتحقق من نموذج النظام وفقاً لمعايير قياس الاداء وقد أثبتت الدراسة التجريبية أن الحل المقترح يحسن من كفاءة استهلاك الطاقة ويقلل من وقت التأخيرات الحاصلة داخل النظام دون التأثير على امكانية توفر النظام للاستعمال.

الكلمات المفتاحية : الحوسبة السحابية, كفاءة استهلاك الطاقة, جدولة المهام, سياسة تنبيه الرجوع للخوادم, ومؤقت تأخير الاطفاء الجزئي للخوادم, نظام قائمة انتظار, سلسلة ماركوف

Résumé

Cloud computing devient de plus en plus populaire. Aujourd'hui, nous connaissons une pénurie d'énergie dans le monde entier. C'est pour cette raison que de plus en plus de chercheurs se préoccupent de trouver des moyens d'améliorer l'efficacité énergétique et la durabilité de cloud computing. Cette thèse présente une analyse des performances du système de cloud computing basée sur la stratégie d'ordonnancement des tâches à faible consommation d'énergie. Cette stratégie consiste en une politique de seuil de réveil et une minuterie de retardement du sommeil.

Pour modéliser le système dans une stratégie de planification des tâches économe en énergie, nous utilisons un système de file d'attente de vacances avec un seuil de politique N. Dans ce travail, le système considéré comme une machine physique avec un tampon limité, cette machine physique contient un ensemble de machines virtuelles qui représentent les serveurs dans notre modélisation, ainsi nous avons construit une chaîne de Markov en temps continu et un générateur infinitésimal qui nous permet de calculer un vecteur de distribution stationnaire. Ensuite, nous analysons le système en état stationnaire pour calculer les différentes mesures de performance du système comme la consommation d'énergie et le retard du système.

Nous avons testé notre système avec différents scénarios et avec différentes valeurs de paramètres pour découvrir la performance de la stratégie proposée et valider le modèle du système en fonction de la mesure de performance.

L'étude expérimentale a prouvé que la solution proposée améliore l'efficacité énergétique et le temps de latence sans affecter la disponibilité dans le système

Mots clés : Cloud Computing, l'efficacité énergétique, ordonnancement des tâches, politique de seuil de réveil, minuterie de retardement du sommeil, file d'attente de vacances, chaîne de Markov

Table of content

Introduction.....	1
Chapter 1 Introduction to Cloud Computing systems	3
1.1 Introduction	4
1.2 History of Cloud Computing.....	4
1.3 Cloud Computing	5
1.4 Essential Cloud Computing properties	6
1.5 Cloud service models	7
1.5.1 Infrastructure as a service (IaaS)	7
1.5.2 Platform as a service (PaaS)	8
1.5.3 Software as a service (SaaS).....	9
1.6 Cloud deployment models.....	9
1.6.1 Public cloud	9
1.6.2 Private cloud	10
1.6.3 Hybrid cloud	11
1.6.4 Community cloud	11
1.7 Cloud workload types	12
1.7.1 Static workloads.....	12
1.7.2 Periodic workloads	13
1.7.3 Unpredictable workloads.....	13
1.7.4 Hybrid Workloads	14
1.8 Cloud virtualization.....	14
1.8.1 Hypervisor	15
1.9 Advantages and disadvantages of Cloud Computing	17
1.10 Energy consumption in Cloud Computing	17
1.11 Conclusion.....	18
Chapter 2 Studies of Markov chain and queueing models	19
2.1 Introduction	20

2.2 Markov chains	20
2.2.1 Random variables	20
2.2.2 Stochastic process	21
2.2.3 Poisson process	22
2.2.4 Definition of Markov chains.....	22
2.2.5 Types of Markov chains	23
2.2.6 Conditional probability	23
2.2.7 Discrete-time Markov chain	24
2.2.8 Continuous-time Markov chain	28
2.3 Queueing models.....	34
2.3.1 Kendall's notation.....	35
2.3.2 Little's law	36
2.3.3 Standard queueing models.....	36
2.3.4 Queues with call-back.....	39
2.3.5 Vacation queueing models.....	41
2.3.6 Queuing systems with threshold policies	42
2.3.7 Arrivals and service	42
2.3.8 Poisson arrivals and exponential service	46
2.3.9 Performance measures.....	47
2.3.10 Birth-Death processes: The M/M/1 queue	49
2.3.11 Description and steady-state solution	50
2.3.12 Matrix formulation of the M/M/1 queue	51
2.3.13 The M/M/c queue	52
Chapter 3 : Modeling of the system.....	54
3.1 Introduction	55
3.2 Related works.....	55
3.3 Cloud model with n-policy.....	57
3.3.1 Description.....	57

3.3.2 Continuous time Markov chain of system Cloud model with n-policy.....	60
3.3.3 Resolution	62
3.3.4 Analysis of Cloud model with n-policy.....	62
3.3.5 Infinitesimal generators	65
3.4 Cloud model with n policy and sleep delay state.....	67
3.4.1 Description:.....	67
3.4.2 Continuous time Markov chain of system Cloud model with n-policy and sleep delay state	70
3.4.3 Resolution	73
3.4.4 Infinitesimal generator.....	75
3.5 Performance measures	76
3.6 Conclusion.....	80
Chapter 4 Experimental	81
4.1 Introduction	82
4.2 Experimental studies	82
4.3 Development tools.....	83
4.3.1 C-sharp.....	83
4.3.2 Visual studio	83
4.3.3 Matlab	84
4.4 Model 1 (Cloud model with n-policy)	85
4.5 Model 2 (Cloud model with n-policy and sleep delay state)	88
4.6 Model 1 and model 2 comparison.....	91
4.7 Model2 with different sleep delay parameter value.....	94
4.8 Conclusion.....	97
Conclusion	98
Bibliography	99

Table of Figures

Figure 1-1.1 Data center	5
Figure 1-2 Application stack and associated cloud service models	7
Figure 1-3 Infrastructure as a service	8
Figure 1-4 Platform as a service	8
Figure 1-5 Software as a Service	9
Figure 1-6 Public Cloud	10
Figure 1-7 Privat cloud	10
Figure 1-8 Hybrid cloud	11
Figure 1-9 Community cloud	12
Figure 1-10 Static workloads example	13
Figure 1-11 Periodic workloads example	13
Figure 1-12 Unpredictable workloads example	14
Figure 1-13 Mainframe Virtualization Architecture	15
Figure 1-14 Hypervisor types using virtualization and para-virtualization ..	16
Figure 2-1 example of n-step transition.....	27
Figure 2-2 example of reachable and communicating states	30
Figure 2-3 example of absorbing states.....	31
Figure 2-4 Example of Periodic and aperiodic states.....	32
Figure 2-5 Little's Law representations	36
Figure 2-6 Finite population queue model	37

Figure 2-7 Infinite population queue model.....	37
Figure 2-8 The M/M/1 queue	50
Figure 2-9 The M/M/C queue.....	52
Figure 3-1 The general representation of the system	57
Figure 3-2 description of some model 1 transition.....	59
Figure 3-3 Model 1 CTMC with 5 dimensions	61
Figure 3-4 Model 1 CTMC with 3 dimensions	64
Figure 3-5 description of somme model 2 transition	69
Figure 3-6 Model 2 CTMC with 6 dimensions	71
Figure 3-7 Model 2 CTMC with 4 dimensions	74
Figure 4-1energy consumption in model 1.....	85
Figure 4-2: mean waiting time in model 1	86
Figure 4-3: blocking probability in model 1.....	87
Figure 4-4: energy consumption in model 2.....	88
Figure 4-5: mean waiting time in model 2	89
Figure 4-6: blocking probability in model 2.....	90
Figure 4-7: energy consumption in model 1 and 2.....	91
Figure 4-8: mean waiting time in model 1 and 2.....	92
Figure 4-9: blocking probability in model 1 and 2.....	93
Figure 4-10: energy consumption in model 2 varying sleep delay	94
Figure 4-11: mean waiting time in model 2 varying sleep delay	95
Figure 4-12: blocking probability in model 2 varying sleep delay	96

List of acronyms and abbreviations

CTMC: continuous time Markov chain

DTMC: discrete time Markov chain

DVFS: dynamic voltage frequency scaling

DVS: dynamic voltage scaling

FIFO: first in first out

IaaS: infrastructure as a service

LIFO: last in first out

PaaS: platform as a service

PM: physical machine

SaaS: software as a service

VM: virtual machine

Introduction

Cloud Computing has become an increasingly used concept referring to memory, computing capacity, computers, and servers distributed worldwide and linked by the network.

Cloud is an On-demand self-service that provides users with high network access and allows them to request more additional computing resources and pay per use basis with make it a powerful technology to do any required tasks. Although the cloud provides a good performance and service time, Cloud data center consumes much energy. In this work, we try to lower the amount of energy consumption and get better performance.

Problematic

Energy consumption and latency are two of the leading cloud computing factors. Generally, energy consumption and latency have an inverse relationship while ensuring better energy consumption latency will be affected. And the same with latency. If we ensure a better latency, we will consume more energy.

In this work, the problem is that we try to reduce the energy consumption of cloud computing and minimize the latency of tasks.

Objectives

The objectives pursued through this work are the following:

- Studying cloud computing services in general. and concentrate on infrastructure as a service
- Examine Markov chains and queue models concentrating on queues with vacation and queue with a threshold.
- Define a solution in cloud computing infrastructure as a service that represents a system of a queue with vacation and sleep-delay timer as well as threshold policy.

Introduction

- Validation of the proposed model solution, through an experimental study, using c-sharp an application and Matlab tools.

Thesis organization

This document consists of a general introduction, four chapters, and a general conclusion. this work will begin with a general introduction, in which we have explained the general concept of this paper, specified the problem, and set the different objectives.

First chapter: the first chapter is a chapter of generalities. It represents a global view of the Cloud Computing domain. It includes definitions for the different concepts, and it gives the characteristics, models, advantages, and disadvantages of the cloud, as well as the evolution of Cloud Computing.

Second chapter: in the second chapter, we talked about some mathematical concepts like Markov chain, random variable, probability, stochastic process...e.g., Then we talked about queueing theory models like a queue with vacation, queue with threshold policy. We also talked about the infinitesimal generator and performance measures.

Third chapter: in this chapter, we define model 1 and model 2, specifying each model CTMC, infinitesimal generator, and the methods we use to find the system steady-state solution.

Fourth chapter: this chapter is the last, dedicated to experiments. It starts with a presentation of the tools and the development environment. After that, we test different experimentation scenarios when measuring model 1 and model 2 performances, then discussing the results obtained. At the end of this brief, a general conclusion presented as a summary of this work.

Chapter 1 Introduction to Cloud Computing systems

1.1 Introduction

Cloud computing is a novel paradigm for the provision of computing infrastructure, on the cloud we move the location of infrastructure and machine from our location to the network in order to reduce the costs of management and maintenance of hardware and software resources. The cloud providers by shifting resources to the network transfers management, maintenance, and investment from the customer to the provider. Cloud computing is a model for enabling on-demand network access to a shared pool of configurable computing resources (e.g., high-speed network, on-demand self-service, measured service (pay-per-use), resource pooling and rapid elasticity) to any customers in the word that can rapidly provisioned and released with minimal management effort or service provider interaction, largest cloud provider companies are amazon, google and Microsoft

We introduce the fundamentals required to understand cloud computing in this chapter. We describe basic cloud properties, cloud service models (IaaS, PaaS, SaaS), deployment models and the virtualization technology that used to provide powerful virtual machines. It is essential to understand why the cloud has these properties, how these properties delivered on different levels and how its benefits attract customers to use cloud computing rather than traditional computing.

1.2 History of Cloud Computing

At around in 1961, John MacCharty suggested in a speech at MIT that computing can be sold like a utility, just like electricity on a monthly basis. Back then it was a new and brilliant idea, but in that time this idea, it was ahead of its time, as for the next few decades, despite people and technology industry interest in the model, the technology was not ready for it. However, of course, many years later the technology developed and caught that idea and began to work on it, after a few years we mentioned that:

Introduction to Cloud Computing systems

In 1999, Salesforce.com started delivering applications to users using a simple website. Enterprises use Salesforce applications over the Internet by paying monthly rate price, and this way, the old idea of computing sold as a utility was right (1).

In 2002, Amazon started Amazon Web Services, providing services like storage, database, computation and even human intelligence. However, only starting with the launch of the Elastic Compute Cloud in 2006 commercial service open to everybody existed. Elastic Compute Cloud was a considerable success (1).

In 2009, Google Apps started Google Cloud to provide cloud computing enterprise applications. Google now provide some free cloud services such as google drive(1).

In 2009, Microsoft started Windows Azure, and big technology companies like Oracle and HP have all joined the game trying to dominate the market. Today this proves that cloud computing has become mainstream(1).

1.3 Cloud Computing

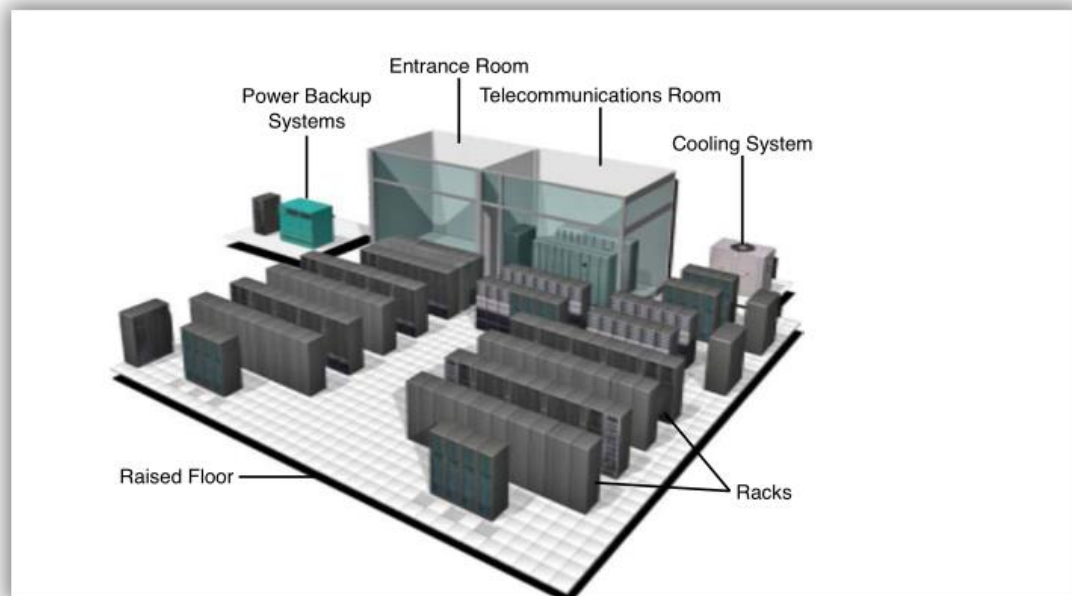


Figure 1-1.1 Data center (30)

Introduction to Cloud Computing systems

Cloud computing is delivering hosted services like servers, networking storage, databases, software, analytics and intelligence over the internet to offer faster innovation, flexible resources, and economies of scale. You typically pay only for computing services you use, For the duration you used it actively (pay-per-use), which lower your costs, manage infrastructure more efficiently and scale as your business needs change due to flexible resources offering.

1.4 Essential Cloud Computing properties

The following Properties make cloud computing technology so powerful and overcome a lot of its disadvantages:

- **On-demand self-service (2):** customers may reserve and release computing resources independently exactly as needed, which allows the customer to reduce the cost.
- **High network access (2):** the integration of distributed computing resources in an application need high speed and low latency network in order to reduce data access times and become less dependent on the physical location where data is stored.
- **Pay-per-use (2):** the use of cloud computing resources storage, processing, or data exchange is measured. This metering is used to enable pay-per-use pricing models the benefit of measured service, and the enabled pay-per-use pricing models for the customer is quite evident as no more investments in non-used or under-utilized IT resources is necessary. However, the cloud provider must deal with the fact that resources can be returned by customers when they do not need them and make sure they available to be assigned to other customers.
- **Rapid elasticity(2):** rapid elasticity allows users to automatically request additional space in the cloud or other types of services. Because cloud infrastructure setup, provisioning can be seamless for the customer or organization. Cloud provider allocation and de-allocation of computing resources are often irrelevant on the customer's side. This is an essential aspect of cloud technology. In a sense, cloud computing resources appear to be infinite or automatically available.

Introduction to Cloud Computing systems

- **Resource pooling(2):** to deal with the demand for pay-per-use, cloud providers offers IT resources using a large IT resource pool that is shared by multiple customers. To be able to assign resources of the resource pool dynamically to customers, it required that the resource pool supports elasticity, i.e., customers can rapidly grow or shrink the share of the resource pool assigned to them. The cloud provider can automatically detect underutilized IT resources or increased demand of customers and assigns IT resources accordingly.

1.5 Cloud service models

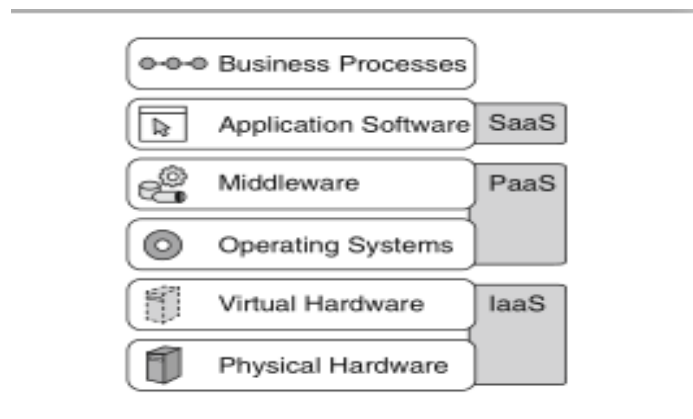


Figure 1-2 Application stack and associated cloud service models (2)

There are different cloud service models according to the layers of the application stack for which they provide cloud resources.

1.5.1 Infrastructure as a service (IaaS)

Infrastructure as a service means only the infrastructure is given to you, you control everything else and manage it the way you want it, and when you use it. IaaS provides computing architecture and infrastructure apart from that data storage, virtualization servers, and networking.

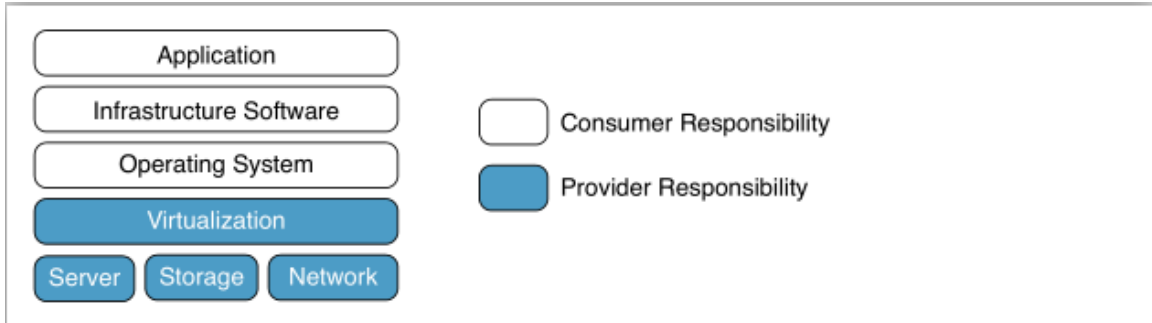


Figure 1-3 Infrastructure as a service (30)

1.5.2 Platform as a service (PaaS)

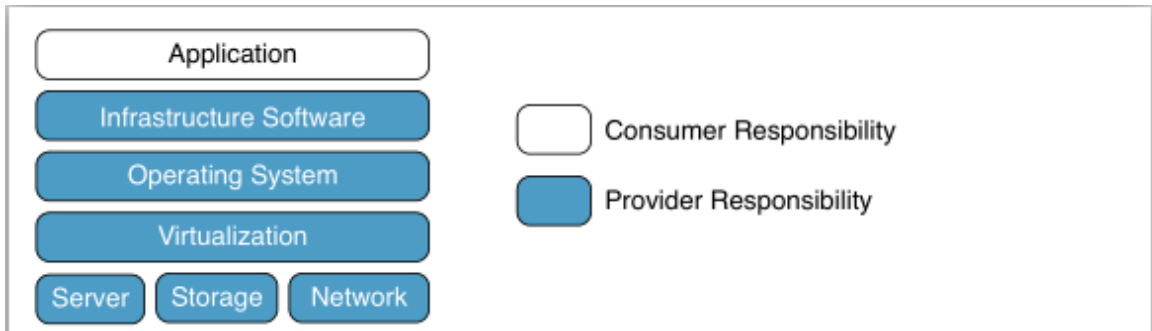


Figure 1-4 Platform as a service (30)

Platform as a Service (PAAS) is a cloud computing model that delivers infrastructure software over the internet. A cloud provider hosts the hardware, operating systems and software on its infrastructure. As a result, PAAS users do not worry about installing in-house hardware, operating system and software to develop or run a new application. Instead, they focus only on application development. PAAS providers for key services, such as Java development, website or application hosting. A PAAS provider, however, supports all the underlying computing and software; users then login and start using the platform usually through a Web browser interface. PAAS providers then charge users on a pay-per-use basis or on a monthly basis(3).

Introduction to Cloud Computing systems

1.5.3 Software as a service (SaaS)

In SAAS, cloud provider offers a complete software application to customers who may use it on-demand via a self-service interface (2). The provider, therefore, provides customers with an entire application stack with GUI to support their business. Users log in and access only to the application software, and they use it the way you want it, but do not have to install and manage an application required to support these processes. Accesses to this application billed on a pay-per-use basis.

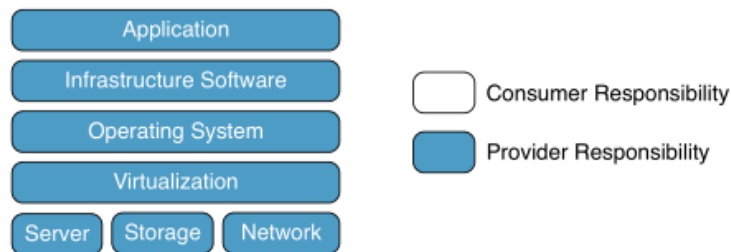


Figure 1-5 Software as a Service (30)

1.6 Cloud deployment models

1.6.1 Public cloud

This type of cloud deployment model can support all users who want to benefit from computing on a pay-per-use basis or a subscription basis. Customers connect to the cloud through a public network which means security is a significant risk in this cloud. Most common uses of public clouds are for non-mission-critical tasks such as file sharing, e-mail service, application development and testing, simulation labs (4).

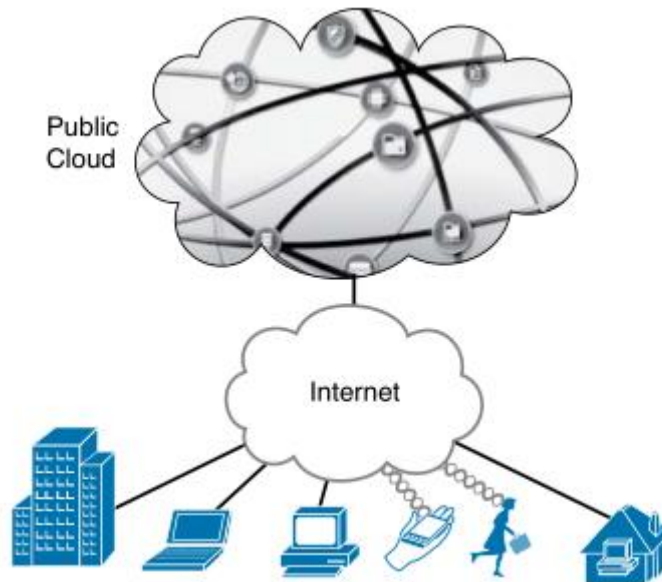


Figure 1-6 Public Cloud (30)

1.6.2 Private cloud

Cloud infrastructure is used by a single organization in private cloud environment. This infrastructure managed by the organization itself to support various user groups or a cloud provider could manage it. Users connect to this cloud model through a private network. Private clouds are more expensive than public clouds due to the extra cost involved in maintaining them (4). However, private clouds are better in security and privacy to organizations. We can use this cloud in critical tasks that contain sensitive data.

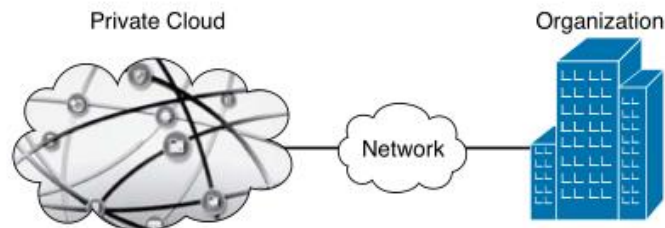


Figure 1-7 Privat cloud (30)

1.6.3 Hybrid cloud

In a hybrid cloud, we use interconnected private and public cloud infrastructure. Many organizations use this model when they need to scale up their IT infrastructure rapidly. For example, if an organization hosting Web application on the cloud needs more computing resources during the holiday season to run the application, it may attain those resources via public clouds (4).

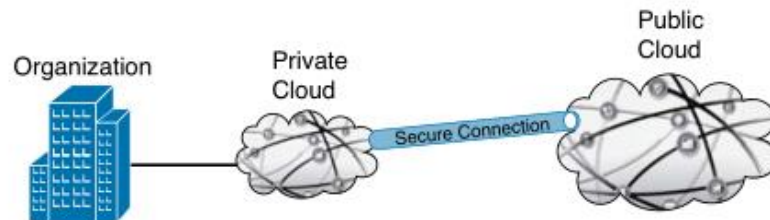


Figure 1-8 Hybrid cloud (30)

1.6.4 Community cloud

In the community cloud, multiple organizations that are part of a community sharing computing resources, examples include universities cooperating in big research projects, or police departments within a county or state sharing computing resources. Access to a community cloud environment (4) is restricted only to the members of the community.

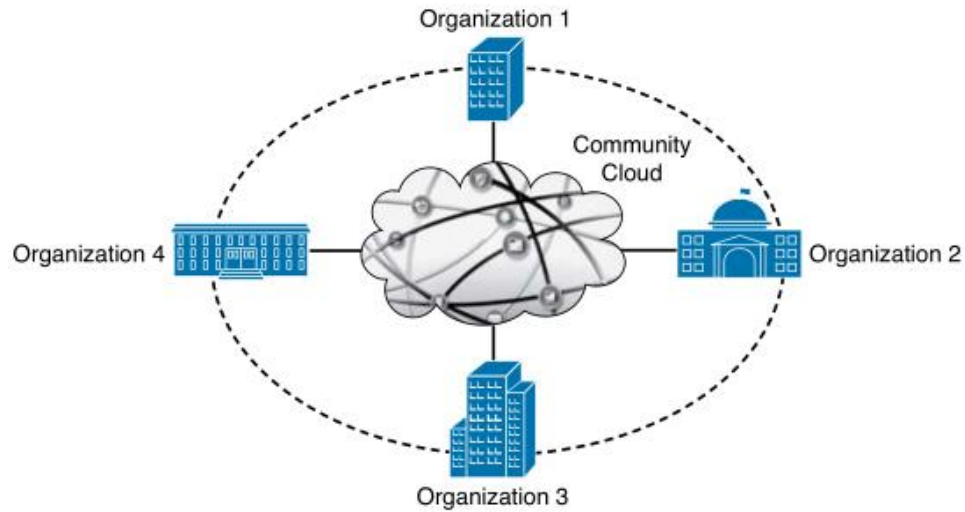


Figure 1-9 Community cloud (30)

1.7 Cloud workload types

1.7.1 Static workloads

Customers in static workload have predictable and pre-determine workload, meaning there are no surprises, no traffic spikes & rushes. This kind of workload can be a utility deployed on the cloud have a limited number of users in a private network, for example, an organization-wide tax-calculation utility or the enterprises with few numbers of customers.

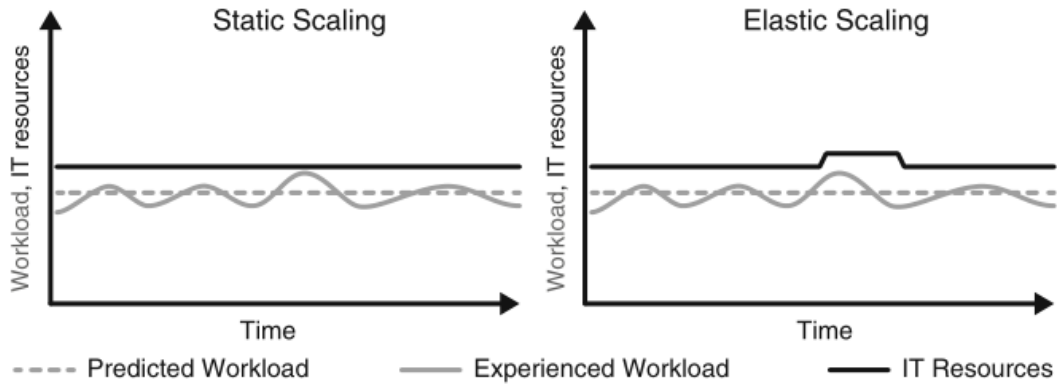


Figure 1-10 Static workloads example (2)

1.7.2 Periodic workloads

The utilization in these types of workloads happens at specific times only known to the cloud providers, maybe like an electricity bill payment app happen in a few days in a month. The best cloud models for these kinds of applications is Serverless compute models; customers do not need to pay for idle resources without using them, pay for the compute utilized.

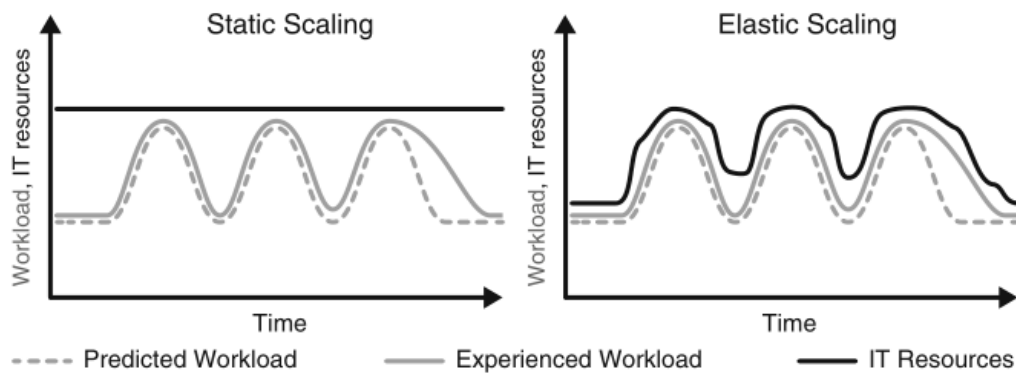


Figure 1-11 Periodic workloads example (2)

1.7.3 Unpredictable workloads

Popular huge apps like social networks have these workloads include based on the fact that many classes worldwide of customers use them at any moment, online multiplayer

Introduction to Cloud Computing systems

games, video, game streaming apps etc. Traffic can spike by any amount exponentially. Pokémon Go surpassed all traffic expectations by growing up to 50x the anticipated traffic. Likewise, on social networks, traffic spikes when any major global worldwide event occurs. The auto-scaling cloud ability and rapid elasticity in these kinds of scenarios save the day by dynamically adding additional instances when required.

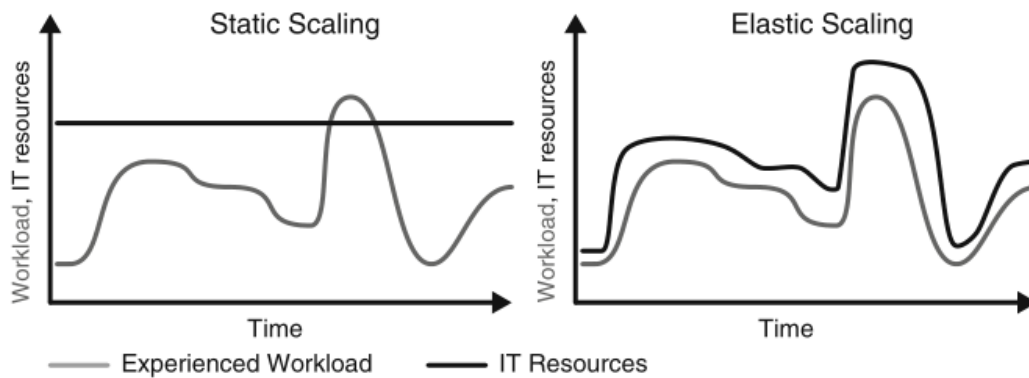


Figure 1-12 Unpredictable workloads example (2)

1.7.4 Hybrid Workloads

Hybrid workloads can be the mix of the above-stated workloads. Well, there is no limit to the architectural complexity in scalable applications.

1.8 Cloud virtualization

A virtual machine (VM) is an image file managed by the hypervisor that exhibits the behaviour of a separate computer, capable of performing tasks such as running applications and programs like a separate computer(5). In other words, a VM is a software application that performs most functions of a physical computer, actually behaving like a separate computer system.

Introduction to Cloud Computing systems

Deploying applications directly on physical servers presents several risks. It makes the application directly dependent on physical hardware failures, also known as a single point of failure if so many applications or VMs fail due to physical hardware fail. It leads to unavailability if the hardware has to be re-configured for updates or replacement.

Virtualization: in virtualization, we create a virtual version of hardware (physical), such as a server, CPU, a storage device, or network resources".

In cloud environments, Virtualization is a technique, which allows multiple customers to share a single hardware instance of a resource or an application. Assigning a logical name to physical storage and providing a pointer to that physical resource when demanded. By virtualization, we can install different operating systems on the same physical machine without any problems.

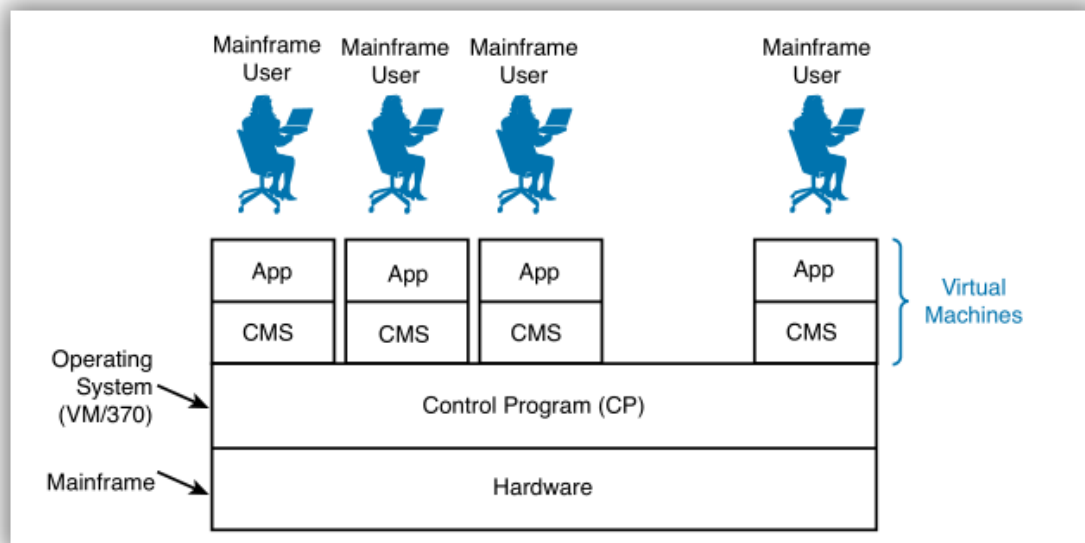


Figure 1-13 Mainframe Virtualization Architecture (30)

1.8.1 Hypervisor

A hypervisor is a software that makes virtualization technology possible. It abstracts the physical server into virtualized hardware where different operating systems and

Introduction to Cloud Computing systems

middleware installed to host applications sharing the physical server while being isolated from each other regarding the use of physical hardware(6).

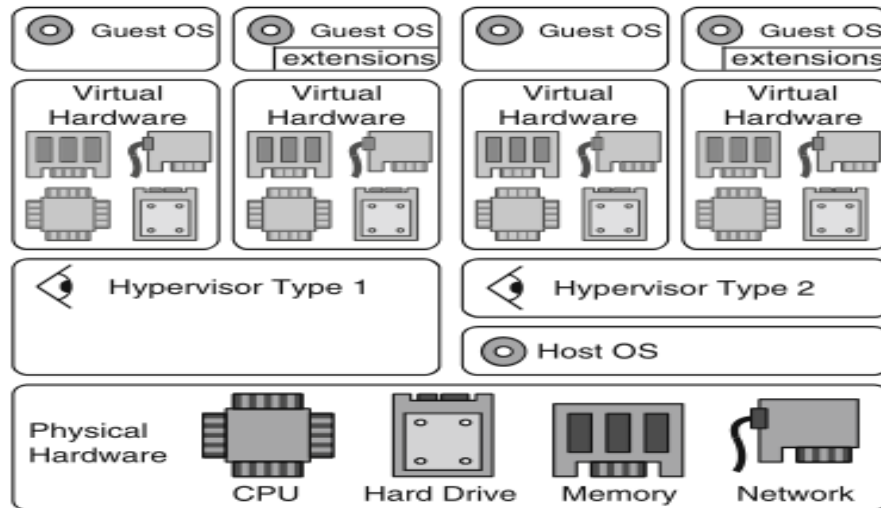


Figure 1-14 Hypervisor types using virtualization and para-virtualization (6)

- **Type-1 Hypervisor:** hypervisor runs directly on physical hardware. Known also as “Native Hypervisor” or “Bare metal hypervisor”. It does not require any operating system. It has direct access to hardware resources. We use this type in a cloud environment due to its smooth performance. Examples of Type 1 hypervisors include VMware ESXi, Citrix XenServer and Microsoft Hyper-V hypervisor (2).
- **Type-2 Hypervisor:** hypervisor runs on an underlying host operating system. It is also known as “Hosted Hypervisor”. It is software installed on an operating system. Hypervisor asks the operating system to make hardware calls. Example of Type 2 hypervisor includes VMware Player, virtual box or Parallels Desktop. Hosted hypervisors often found on endpoints like PCs (2).

1.9 Advantages and disadvantages of Cloud Computing

From what we have addressed earlier, we can conclude that Cloud has its advantages and disadvantages. Some of the cloud advantages are cost efficiency, High speed deploy and network, anywhere accessibility and On-demand computing resources.

Biggest cloud disadvantages are security and Energy consumption. The energy consumption of under-utilized resources, particularly in a cloud environment, accounts for a substantial amount of the actual energy use (7).

1.10 Energy consumption in Cloud Computing

The use of cloud computing is becoming widespread. Data centers have increased energy requirements for power and cooling. In 2012, for every \$1 spent on hardware there was \$1 spent for power and cooling. That is why reducing consumption has a substantial economic impact. Besides, there is also an ecological impact because the environmental footprint is not negligible. Indeed, in 2008, data centers emitted 116 million tons of carbon dioxide, which is more than Algeria's total emissions.

Researches of Cloud computing in the section of reducing energy has focused lately on reducing it physically like dynamic voltage and frequency scaling DVFS .and it is implemented in the architecture of cloud allowing the system to supply the voltage and adjust the frequency to a particular component within the physical system and this method shows a significant power and energy saving that means that those parts was before this method wasting a big amount of energy, with the same idea but with the system model of the cloud researchers searched for the gabs that waste energy in the system of cloud and try to optimize it.

By using the concept of VMs and introducing sleep state, sleep time, and many other concepts, researchers in both the physical field and the software field are trying to reduce the energy consumption in cloud computing.

1.11 Conclusion

In this chapter, we introduced cloud computing and brought up the essential cloud computing properties. We talked about cloud service models, Infrastructure as a service is one of the services models that we focus about in our work, after that, we address virtualization and its massive benefit to the cloud, and in the end, we talked about the problem of energy consumption in cloud computing.

In the next chapter, we will bring some mathematical concepts and speak about queueing theories and Markov chains

Chapter 2 Studies of Markov chain and queueing models

2.1 Introduction

Today, most systems have become more complex. It is necessary to check whether the future system meets the specified requirements described in our agenda. It is also necessary to modify or improve a system, which is already operational, by modifying or improving specific parameters of performance, such as the energy consumed in the system, the mean waiting time of tasks in the buffer, the system service throughout or the blocking probability in a system. The system can be used to determine the capacity, throughput and condition of the various machines in a production system, etc.

The process for determining the various performance parameters of the systems is known as performance evaluation or analysis. However, before to do this, we must go through a modelling phase that allows us to deduce a model, which represents a mathematical abstraction of the system. After we got our model, we can easily analyze it and figure out the performance of the reel model; thus, we try to improve it.

Our study on the performance of cloud machines, which will be addressed in the next chapter. Is essentially based on these mathematical formalisms: the queues and continuous-time Markov chain. However, the analysis of these models requires the mastery of stochastic processes and more particularly of Markov chains steady state.

2.2 Markov chains

2.2.1 Random variables

Any variable x defined on a sample space S for which the cumulative probabilities $Pr \{x \leq a\}$ can be defined for all real values of a , $-\infty < a < \infty$, is called a real random variable x . (8)

Studies of Markov chain and queueing models

A discrete random variable has a countable number of possible values, any random variable x which takes individually distinct values with nonzero probabilities is called a discrete random variable and in this case, the probability function, denoted by $f(x)$, is given by (8)

$$f(x) = \begin{cases} Pr\{X = x\}, & x \in S \\ 0, & x \notin S \end{cases} \quad \dots(2-1)$$

A continuous random variable takes all values in an interval of numbers, Any random variable x which is defined on a continuum of points, where the probability that x takes a specific value x' is zero, is called a continuous random variable and the density function is available from the cumulative density by differentiation, when differentiable, or the cumulative density is available by integration of the density. (8) That is:

$$F_x(a) = \int_0^a f_x(t) dt \quad \dots(2-2)$$

2.2.2 Stochastic process

A stochastic process is also called a random process. It describes how random variables evolve. (9)

Let X_t be the value of some characteristic at time t , X_t represents a random variable, and it is not known with certainty before time t , an example of X_t , is the number of students in the classroom at t minutes after the class starts, some students may come in late, so we do not know the value of X_t for sure before time t . (10)

X_t is called the state of the stochastic process. If a stochastic process can be observed at discrete time instants, it is called a discrete-time stochastic process. If the state of a stochastic process can be observed at any continuous time, it is called a continuous-time stochastic process (9)

Studies of Markov chain and queueing models

For example, the function $f: \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ given by $f(t) = t$ is a deterministic process, but a random function $f: \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ given by $f(t) = t$ with probability $1/2$ and $f(t) = -t$ with probability $1/2$ is a stochastic process.

2.2.3 Poisson process

The Poisson process is one of the most important models used in queueing theory. Often the arrival process of customers can be described by a Poisson process. In telegraphic theory, the “customers” may be calls or packets. The Poisson process is a viable model when the calls or packets originate from a large population of independent users. (8)

Mathematically the process is described by the counter process N_t or $N(t)$. The counter tells the number of arrivals that have occurred in the interval $(0, t)$ or, more generally, in the interval (t_1, t_2) .

In mathematics, a Markov chain is a Markov process with discrete-time, or continuous time, and discrete state space. A Markov process is a stochastic process with the Markov property: the information useful for predicting the future is entirely contained in the present state of the process, and it is not dependent on previous states (the system has no “memory”). Markov processes named after their inventor, Andrei Markov. (9)

2.2.4 Definition of Markov chains

The defining property of a Markov chain is the prediction of the future from the present only, it is not more precise by additional information about the past, because all the information that is useful for predicting the future is contained in the present state of the process. Its future and past states are independent. (11)

A Markov chain is a type of Markov process with either a discrete state space or a set of discrete indices (often representing time), but the precise definition of a Markov chain varies. For example, it is common to define a Markov chain as a discrete or continuous-

Studies of Markov chain and queueing models

time Markov process with a space of countable states (independent of the nature of time), but it is also common to define a Markov chain as having a discrete-time in the space of countable or continuous states (independent of the space of states). (11)

2.2.5 Types of Markov chains

The system time parameter and state space are specified in Markov chains. So, we have variation in parameters of the chains between the discrete-time and continuous-time (time parameter). The countable state space and the continuous state space, so we conclude the four deferent Markov chain types. In this study, we focus only on countable state space, so we have:

- Discrete-time Markov chain
- Continuous-time Markov chain

2.2.6 Conditional probability

The conditional probability of event A is the probability that the event will occur given the knowledge that B has already occurred. This probability is written $P(A|B)$, the notation for the probability of A given B. In the case where events A and B are independent (where event B does not affect the probability of event A), the conditional probability of event A given event B is simply the probability of event A, that is, $P(A)$. (12)

If events A and B are not independent, then the probability of the intersection of A and B (the probability that both events occur) is defined by:

$$P(A \text{ and } B) = P(A|B) P(B). \quad \dots(2-3)$$

From this definition, the conditional probability $P(A|B)$ is easily obtained by dividing by $P(B)$:

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)} \dots (2-4)$$

2.2.7 Discrete-time Markov chain

A discrete-time Markov process is an $X_0, X_1, X_2, X_3, \dots$ a sequence of random variables with values in state spaces, and E represents this sequence of random variables. The value X_n is the state of the process at the moment n . The applications where the state space of E is finite or countable are innumerable. In this study, the essential properties of general Markov processes, such as recurrence and ergodicity, are more simply stated or demonstrated in the case of discrete space Markov chains, in this study we focus, on discrete space Markov chains.

A discrete-time stochastic process is a Markov chain if, for $t = (0, 1, 2, \dots)$ And so on, the states have the following relation:

$$P(X_{t+1} = i_{t+1} | X_t = i_t, X_{t-1} = i_{t-1}, X_1 = i_1, X_0 = i_0) = P(X_{t+1} = i_{t+1} | X_t = i_t) \dots (2-5)$$

Last equation is an equation where the left-hand side is a conditional probability. It represents the probability that at a time $t+1$, the state is i_{t+1} , given that at a time 0, the state is i_0 At time 1 the state is i_1 and at time t the state is i_t . On the right-hand side, the equation is also a conditional probability. It represents the probability that at time $t+1$, the state is i_{t+1} , given that at time t the state is i_t .

This equation means the probability distribution of the state at time $t+1$ depends only on the state at time t . it does not depend on the states before time t .

- **Initial probability distribution**

An initial probability distribution can describe the initial state of the Markov chain

Studies of Markov chain and queueing models

q_i : the probability that the chain is in a state i at time 0: $P(X_0 = i) = q_i$

If we have a total of s different states, then we call the vector $q = [q_1, q_2, \dots, q_s]$ the initial probability distribution of the Markov chain where $\sum_{i=1}^s q_i = 1$.

- **Stationary assumption**

We will describe when the Markov chain is called a stationary Markov chain

Given:

$$P(X_{t+1} = j | X_t = i) \quad \dots(2-6)$$

The probability that the system will be in state j at time $t+1$, given it is in state i at time t .

A Markov chain is called a stationary Markov chain if this probability is independent of time t . that means the probability that the system will be in state j at time $t+1$, given it is in state i at time $t+1$ is equal to the probability that the system will be in state j at time t , given it is in state i at time $t-1$, which is also equal to the probability that the system will be in state j at time 1 given it is in state i a time 0:

$$P(X_{t+1} = j | X_t = i) = P(X_t = j | X_{t-1} = i) = \dots = P(X_1 = j | X_0 = i) = P_{i,j} \quad \dots(2-7)$$

Since his probability does not change, we give it a shorter name $P_{i,j}$ that is called the transition probability from state i at a previous time to state j at the current time. If we have a total of s states, the transition probability from one state to another can be displayed as an $s \times s$ matrix:

$$P = \begin{bmatrix} P_{1,1} & P_{1,2} & \dots & P_{1,s-1} & P_{1,s} \\ P_{2,1} & P_{2,2} & \dots & P_{2,s-1} & P_{2,s} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ P_{s-1,1} & P_{s-1,2} & \dots & P_{s-1,s-1} & P_{s-1,s} \\ P_{s,1} & P_{s,2} & \dots & P_{s,s-1} & P_{s,s} \end{bmatrix}$$

And:

$$\sum_{i=1}^{j=s} P_{i,j} = 1 \quad \dots(2-8)$$

As all these $P_{i,j}$ are probabilities they should all be greater or equal to 0, and less than or equal to 1. Also, the probability of each row must sum to 1. That means that if the Markov chain is in state i at time t , it must transit to one of the s states at time $t+1$. So, the sum should be equals to 1, and there is no such requirement for the columns.

- **n-step transition probability:**

if a Markov chain is in state i at time t , the probability that it will be in state j after n periods called the n -step probability.

$$P(X_{t+n} = j | X_t = i) = P(X_n = j | X_0 = i) = P_{i,j}(n) \quad \dots(2-9)$$

For a stationary Markov chain, this probability will be independent of t , so the probability that the chain is in state j at time $t + n$ given that it is in state i at time t is equal to the probability that the Markov chain is in state j at time n given that it is in state i at time 0, this n -step transition probability from state i to state j is denoted by $P_{i,j}(n)$.

example 1:

$$P = \begin{bmatrix} P_{11} & P_{1,2} & \cdots & P_{2,1} \\ P_{2,1} & P_{2,2} & \cdots & P_{2,1} \\ \vdots & \vdots & \ddots & \vdots \\ P_{2,1} & P_{2,1} & \cdots & P_{s,s} \end{bmatrix}$$

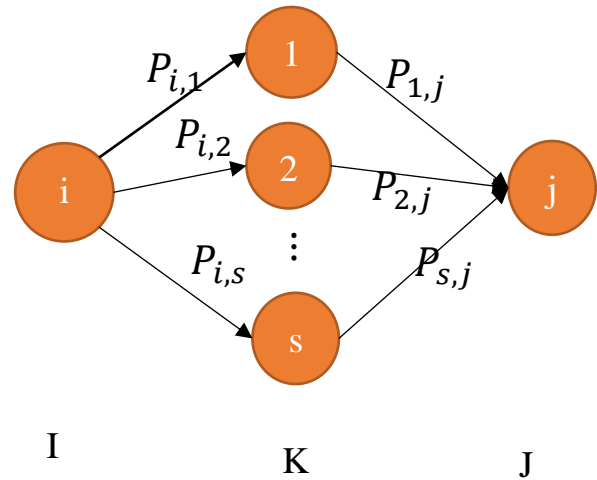


Figure 2-1 example of n-step transition

Assume that we know the one-step transition matrix, which is shown here we can think of a 2-step transition from i to j as two 1-step transition from i to an intermediate state K , and then from K to j . we have a total of s states so K can be any value between 1 and s .

- if the path $i-1-j$ is followed from i to j the probability is:

$$P_{i1} \times P_{1j}$$

- if the path $i-2-j$ is followed from i to j the probability is:

$$P_{i2} \times P_{2j}$$

⋮

- if the path $i-K-j$ is followed from i to j the probability is:

$$P_{iK} \times P_{Kj}$$

it is possible to go through any of the intermediate state, and the 2-step transition probability is equal to:

$$P_{ij}(2) = \sum_{k=1}^{k=s} P_{ik}P_{kj} = ij^{th} \text{ element of } P^2 \quad \dots (2-10)$$

This equation can be generalized as follow:

$$P_{ij}(n) = ij^{th} \text{ element of } P^n \quad n = 1,2,3, \dots \dots (2-11)$$

2.2.8 Continuous-time Markov chain

In probability theory, a continuous-time Markov process, or continuous-time Markov chain, is a continuous-time variant of the Markov process. More precisely, it is a mathematical model with a value in a countable set of states, in which the time spent in each of the states is a positive real random variable, following an exponential law. (9)

A continuous-time $(X_t)_{t \geq 0}$ Markov chain is characterized by:

- a finite or countable set S of states.
- An initial distribution that represent the set of states.
- A matrix Q of transition rates, also called an infinitesimal generator.

For $i \neq j$, the elements q_{ij} of the matrix, Q is positive real numbers that quantify the speed of transition from state i to state j . The elements q_{ii} are chosen so that the columns of each row sum to zero.

Studies of Markov chain and queueing models

$$q_{ii} = \sum_{j \neq i} q_{ij} \quad \dots (2-12)$$

Considering Continuous-time stochastic process $\{X(t)\}$ where for $t \geq 0$ And state space E is either finite or countable

$\{X(t)\}$ is called a continuous-time Markov chain if given time instances

$t_1 < t_2 < t_3 < \dots < t_n < s < s + t$ and integers $i_1, i_2, \dots, i_n, i, j \in E$ we have:

$$P(X(s + t) = j | X(s) = i, X(t_k) = i_k, k = 1, 2, \dots, n) = P(X(s + t) = j | X(s) = i)$$

The probability $p_{ij}(s, t) = P(\{X(s + t) = j | X(s) = i\})$ is called the transition probability.

- **homogenous continuous-time Markov chain**

if $p_{ij}(s, t)$ is independent of s but dependent on t we call the chain homogeneous continuous-time Markov chain, if $\{X(t), t \geq 0\}$, then

$$\begin{aligned} p_{ij}(t) &= P(X(s + t) = j | X(s) = i) \\ &= P(X(t) = j | X(0) = i) \end{aligned}$$

$$p_j(t) = P(X(t) = j)$$

$$= \sum_{k=0} p(0) p_{ij}(t) \quad \dots (2-13)$$

- **State holding time**

When the continuous-time Markov chain enters a state i , the time is spending before it leaves the state i is called the holding time in the state i that holding time t is a continuous random variable.

- **Structure of a homogenous CTMC**

Studies of Markov chain and queueing models

1. CTMC enters at state i . It stays at the state for a time t .

2. Once the CTMC leaves state i , it enters one of the states let us say j with the transition probability p_{ij} where $j \neq i$ and $\sum_{j \neq i} p_{ij} = 1$ (2-14)

The two events of leaving state i and entering the state j are independent because of Markov property.

- **Reachable and communicating states:**

In Markov chain, a path from state i to state j is a sequence of transitions that begins in i the sequence has a positive probability

- A state j is reachable from state i if there is a path leading from i to j
- the state i and the state j can communicate if j is reachable from i , and i is reachable from j
- a set of states S in a Markov chain is closed set if no state outside of S is reachable from any state in S .

Example 2:

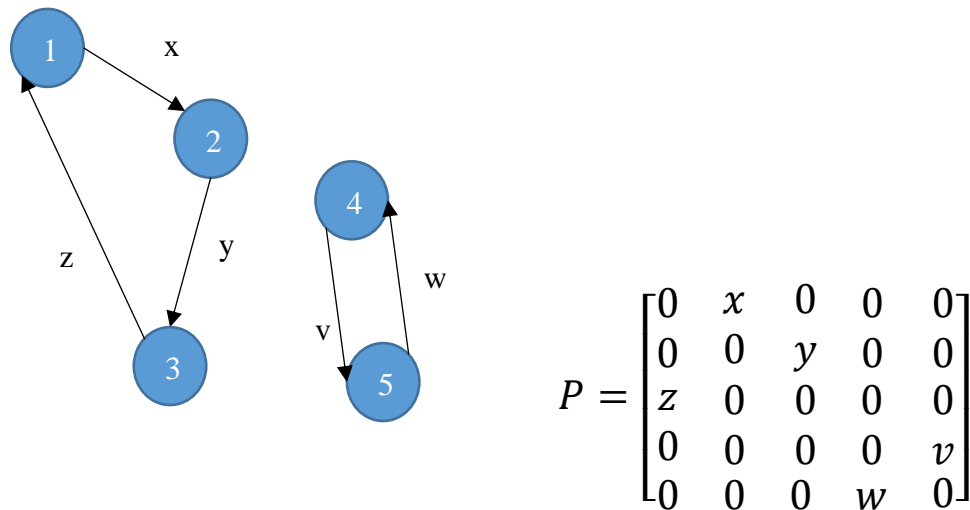


Figure 2-2 example of reachable and communicating states

Studies of Markov chain and queueing models

This Markov chain has 5 states, and P is the state transition matrix. In this example state 3 is reachable from state 1 via the path 1-2-3 but state 5 is not reachable from state 1, there is not path from 1 to 5, here we can say that states 1,2,3 are reachable from each other so S_1 which contains 3 states 1,2 and 3 is a closed set. And set S_2 which contains states 4 and 5 is closed set.

- **Absorbing state**

State i is an absorbing state if $p_{ii} = 1$

That means the probability of the transition from i to itself is 1, so whenever we enter an absorbing state, and we never leave the state. An absorbing state is also a closed set containing only one state.

here is an example:

example 3:

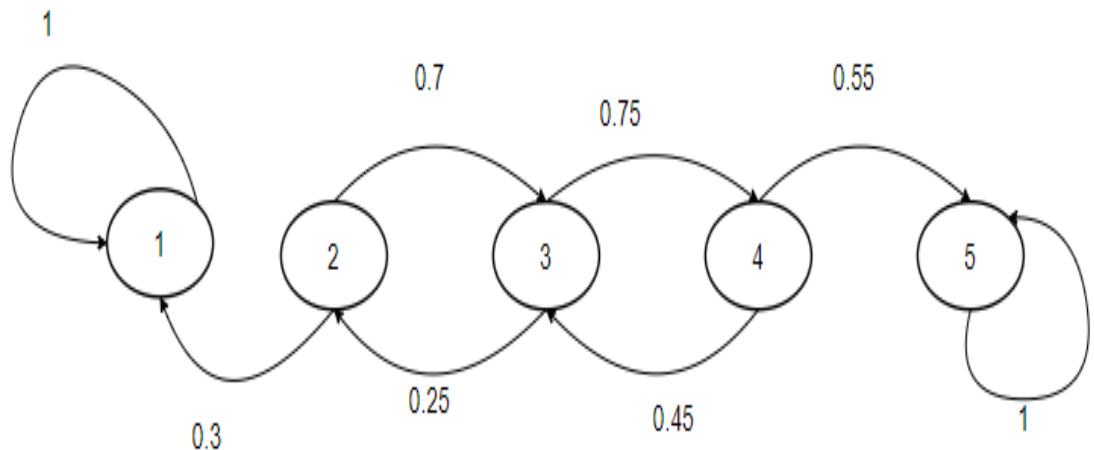


Figure 2-3 example of absorbing states

Studies of Markov chain and queueing models

in this Markov chain both the state 1 and the state 5 are absorbing states, other states are not absorbing states

- **Transition and recurrent states:**

A state i is a transient state if there exists a state j that is reachable from i , but state i is not reachable from state j , it means ones whenever we leave a transient state, it is possible that we will never return to this state.

If a state is not transient, then it is called a recurrent state.

In the last example of Markov chain, there are 3 transient states 2,3 and 4, for example, we can go from 3 to 2 and then from to 1 then we get trapped in state 1 and will never come back to state 3 again, the other two states 1 and 5 are not transient states, so they are recurrent states.

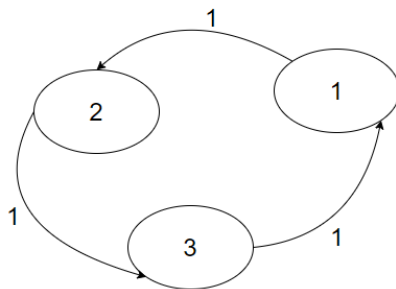
- **Periodic and aperiodic states:**

A state i is periodic with period $k > 1$, if k is the smallest number such that all paths leading from state i back to the state have a length that is multiple of k .

Absorbing states are aperiodic.

If we can return to a recurrent state at irregular times, it is aperiodic.

Example 4:



In this example we have 3 states they are all recurrent states, there is no absorbing state, all states are periodic with a period of 3

Figure 2-4 Example of Periodic and aperiodic states

Studies of Markov chain and queueing models

For example, starting from state 1 we need 3,6,9... steps to come back to state 1

Now we look at example 3 we know that the absorbing states 1 and 5 are not periodic, the transient states 2,3 and 4 are not periodic because we may not come back to these states again.

- **Ergodic Markov chain**

If all states in a Markov chain are recurrent, aperiodic, and communicate with each other, then this Markov chain is ergodic. (13)

In example 3 is not ergodic because it has some transient states 2,3 and 4 and state 1 cannot communicate with state 5.

In example 2 S1 is ergodic because none of the states is transient, there is no period, and every state can communicate with other states

S2 is ergodic also for the same reasons

But S that is represented from 1 to 5 is not ergodic because 1,2 and 3 cannot communicate with states 4 and 5

- **infinitesimal generator and Stationary Distributions**

An infinitesimal generator is a stochastic calculation tool, used in particular for continuous-time Markov chains.

That is constructed as follow:

$$Q [i,j] = \begin{cases} q(i,j), & \text{if } i \neq j \\ - \sum_{a \neq i} Q [i,a], & \text{if } i = j \end{cases} \dots (2-15)$$

Where $Q [i,j]$ is the transition rate from i to j

Studies of Markov chain and queueing models

A probability distribution $\pi = \pi_i \ i \in S$ is said a stationary for the probability transition matrix $P(t)$ if only:

$$\pi P(t) = \pi \quad \dots(2-16)$$

And the stationary distribution $\pi = \pi_i \ i \in S$ is stationary if and only if it satisfies the equation:

$$\pi Q = 0 \quad \dots(2-17)$$

Q is the infinitesimal generator matrix.

2.3 Queueing models

- **Queues:** is a system where units or customers arrive at a waiting area and wait at the end to acquire a service from a service channel if the service is not immediately available, and at the end, for their service to leave this system is called a queue system. The theory of queueing models is a form of probability that refers to the waiting tasks. It allows the analysis of incoming and outgoing tasks in a queued files system, and to calculate various system performance parameters, such as the probability that the service is immediately available to a new incoming customer, the average number of units in the system and waiting, and the time spent waiting in the system. In this way, decisions can be made based on system parameters, such as the number of resources making up the service, for example.

Historically, the theory of waiting files goes back to the beginning of the previous century, when Anger Kramp Erlang, a Danish engineer who worked for the company of Copenhagen, formulated a mathematical solution which made it possible to determine the number of lines necessary to handle a given number of telephone calls. Subsequently, he published several articles that represented the birth of the waiting line theory files. This theory is used in many fields such as commerce, engineering, computer systems and networks, etc. It is also used in the field of computer systems and networks.

Studies of Markov chain and queueing models

In fact, the files queuing theory described above is called classical or standard queuing theory, in which a new client arriving and finding the server(s) busy or unavailable behaves according to one of the two following scenarios:

- The client leaves the system without being served; this corresponds to the Erlang model with loss;
- or it waits to be served after the release of one of the servers, according to a certain discipline (FIFO, LIFO, ...).

There is another scenario, which corresponds to an intermediate situation, in which the customer calls back later to get the service, as many times as necessary and at time intervals distributed according to a certain law of probability. These systems are called call-back systems or systems with repeated calls.

In some situations, the service becomes temporarily unavailable to the customer. Such a system is called a standby system with vacation. In the next sections, we will discuss the models of the standard queueing system files, the call-back queueing system files and the vacation queueing system files

2.3.1 Kendall's notation

In 1951, David George Kendall introduced a set of notations, which have become standard in queuing models, which is a system of notation according to which the various characteristics of a queuing model are identified.

Kendall's notation is denoted by $a/b/c/d/e/f$ where,

- a: describe the distribution time of the arrival process
- b: describe the distribution time of the service process
- c: the number of servers (service channels)
- d: the number of places in the queue (length of the queue)
- e: the calling population (the size of which the customers comes from)
- f: the queue's discipline (it is how the queue is ordering service: FCFS, LCFS, SIRO, PQ)

Studies of Markov chain and queueing models

2.3.2 Little's law

In 1961, John Little published an article showing that the average number of units in the system, L , is related to the average time spent in the system, W , $L = \lambda \cdot W$, where λ is the client arrival rate. In the same way, the following two relationships are established:

- $L_s = \lambda \cdot W_s$, the average number of customers in service.
- $L_q = \lambda \cdot W_q$, the average number of customers in the waiting file.

W_s and W_q represent the average time spent in the waiting area and the average time spent in service, respectively.

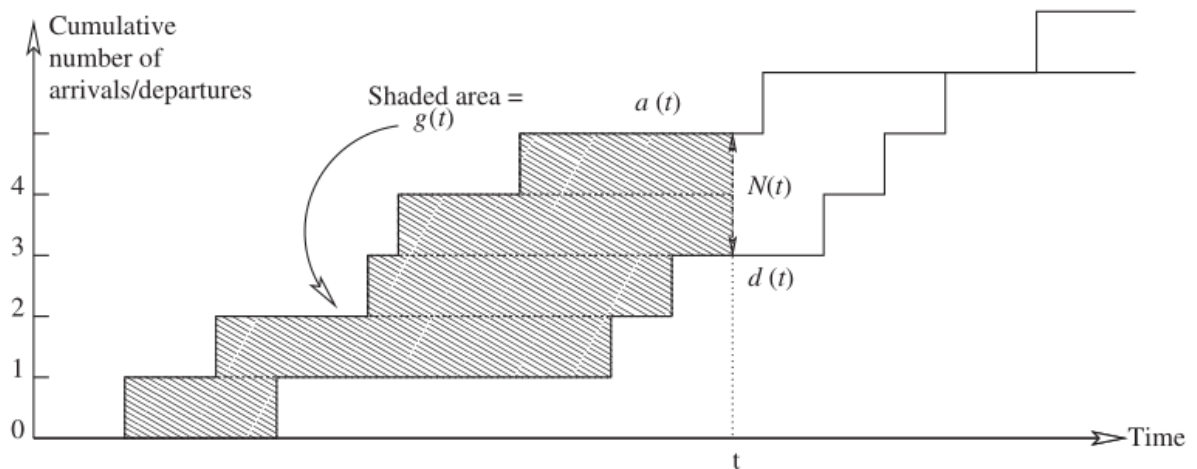


Figure 2-5 Little's Law representations (14)

2.3.3 Standard queueing models

Queueing models is often used in the simulation analysis, it provides the analyst with a powerful tool for designing and evaluating the performance of queueing systems.

Key elements of queueing systems:

- Customer: refers to anything that arrives at a facility and requires service, e.g., people, machines, trucks, emails, packets and frames (14).
- Server: refers to any resource that provides the requested service, e.g., machines, runways at the airport, host, switch, router, disk drive, algorithm.

Studies of Markov chain and queueing models

- Calling population: the population of potential customers may be assumed to be finite or infinite (15).
- Finite population model: if the arrival rate depends on the number of customers being served and waiting
- Infinite population model: if the arrival rate is not affected by the number of customers being served and waiting

Note: if the calling population is not mentioning then it is assumed to be infinite by default
system capacity: a limit on the number of customers that may be in the waiting line or Limited system capacity, e.g., a buffer of a NIC only has room for N packets to wait in line to enter the processing phase, if a system is full no packets are accepted anymore.

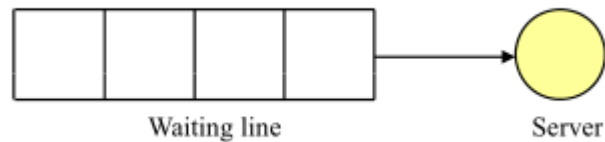


Figure 2-6 Finite population queue model

Unlimited capacity, e.g., concert ticket sales with no limit on the number of people allowed to wait to purchase tickets.

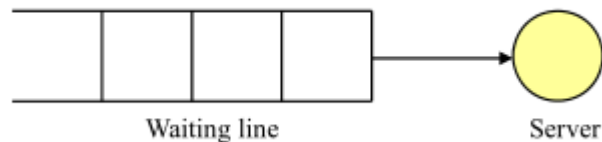


Figure 2-7 Infinite population queue model

For infinite-population model :

Studies of Markov chain and queueing models

Arrival types are:

- Random arrival: arrival times usually characterized by a probability distribution, most important model in random arrivals is Poisson arrival process that we talked about before (with rate λ), where a time represents the interarrival time between customer $n-1$ and customer n , and is exponentially distributed (with mean $1/\lambda$)(16).
- Scheduled arrivals: interarrival times can be constant or constant plus or minus a

small random amount to represent early or late arrivals.

Queue behaviour: the actions of customers in a queue while waiting for a service to begin, for example:

- **Balk:** leaving in the case when they see that the line is too long
- **Reneg:** leaving after being in line when it is moving too slow
- **Jockey:** moving from one line to another shorter line

Queue discipline: the logical ordering of customers in a queue that determines which customer is chosen for service when a server becomes free, for example:

- First-in-first-out (FIFO) (FIFO)
- Last-in-first-out (LIFO)
- Service in random order (SIRO)
- Shortest processing time first (SPT)
- Service according to priority (PR)

Service times of successive arrivals are denoted by S_1, S_2, S_3 it may be constant or random, it is usually characterized as a sequence of (IID) independent and identically distributed random variables, e.g.: exponential distribution.

A queueing system consists of a number of service centers and interconnected queues, each service center consists of some number of servers working in parallel upon getting to the head of the line a customer takes the in parallel, upon getting to the head of the line, a customer takes the 1st available server(14).

2.3.4 Queues with call-back

In the classical queueing theory seen in the previous section, we have found that a new client who arrives on the system and cannot be served immediately. The user either joins the waiting area and waits his turn, or leaves the final system. However, in reality, this is only a first approximation to real situations, where usually such a client returns to the system after a random amount of time and tries to acquire the information. This is known as the recall phenomenon.

Queues with recall have been introduced to address this deficiency, and have been widely used for several model problems in telephone systems, systems and networks in the field of telecommunications.

Since the former works of Kosten, Cohen, Wilkinson, and Riordan, a variety of techniques and results have been developed to resolve some particular problems.

Because of the complexity of queues with call-backs due to the presence of two streams of calls, analytical results are quite difficult to obtain and only exist for specific models, with binding assumptions on certain parameters, such as the number of servers, their reliability, etc... To this end, the researchers oriented towards numerical methods (algorithms), approximation methods, and simulation.

- **General model of queues with call-back:**

A system of file standby with call-back is composed of s ($s \geq 1$) parallel, and independent servers, available for processing clients and an imaginary space called an orbit. A client arriving at the system for the first time is considered a primary client (primary call). If a primary call finds at least one free server, it occupies it immediately and leaves the system as soon as its service is finished. If all the servers are busy, then this client will be blocked, and in this case, it joins the orbit and forms a source of secondary (repeated)

Studies of Markov chain and queueing models

calls and becomes a client in orbit. Each in-orbit client will call back for service at random time intervals until one server is free. In this case, the client is served and then leaves the system. Secondary clients are treated in the same way as primary calls. The time interval between two consecutive attempts made by the same client in orbit is called the call-back time. This time is independent of all previous call-back times.

- **Features of queues with call-back:**

A standby with callback system at file is characterized by a client arrival mechanism, a service mechanism, a callback mechanism, the number of servers, orbit capacity, and the size of the client source. Client arrival times, service times, and callback times are random, making the process described by this model a stochastic process. When the service station consists of a single server, the model is said to be single-server. When it is formed by two or more parallel servers, the model is said to be multi-server. On the other hand, there are two main types of standby with callback systems :

- Open systems (infinite sources): they are fed by a population Infinite. Thus, the number of arrivals is unlimited. As an example, we cite the number of programs submitted to a computer.
- Closed systems (finite sources): they are rather fed by a maximum number of fixed units, corresponding, for example, to the number of subscribers in a telephone network.

This model has several variants, among others the model of files on hold with callback and buffer.

2.3.5 Vacation queueing models

Servers are always available in the classical queueing model, but in many practical queueing systems and due to a variety of reasons servers may become unavailable for a period of time. This period of server absence may represent the servers working on some additional jobs being checked for maintenance or only taking a break.

- **Vacation policies:**

The classical queueing model has three main parts, which is the arrival process, the service process and the queue discipline; however, a vacation queueing model has an additional part, the vacation process controlled by a vacation policy that can be characterized by three aspects:

- The vacation start-up rule: This rule determined when the server starts a vacation, in this rule, there is to main types the exhaustive and the non-exhaustive services, in the exhaustive service, the server cannot take a vacation until the service finished and the system becomes empty. In a non-exhaustive service, the server can take a vacation even if the system is not empty. In a multi-server system, a semi-exhaustive service rule can be used for a part of servers that take a vacation. We can put in mind that the service interruption during the progress service is another vacation start-up rule. The service interruption may be a hardware failure.
- Vacation termination rule: This rule means how the server resumes serving the queue. There are two main rules which are the multiple vacation policy and the single vacation policy. A multiple vacation policy requires the server to keep taking vacations until it finds at least one customer waiting in the system at a vacation completion instant, and under a single vacation policy, the server takes only one vacation at the end of each busy period (17). After this vacation, the server either serves the waiting customers if there are customers or stays idle. There are more rules, such as the threshold policy (also called N-policy), more vacation termination rules are possible.

Studies of Markov chain and queueing models

In addition to start-up and termination rules in multi-server systems. There are other characteristics of a vacation policy. For example, all servers may take vacations together (synchronous vacations or servers may take vacations individually independently (asynchronous vacations). Another possible feature of a vacation policy is to allow some servers to take a vacation to ensure a minimum number of servers are always available

- vacation duration distribution: Vacations in the servers are assumed to be I.D.D (independent and identically distributed) random variables with a general distribution function. Beside some vacation, models require different types of vacation and follow different distributions.

2.3.6 Queuing systems with threshold policies

In recent years queuing models under various thresholds have been the subject of great interest for the queue theorists due to its significant role in performance prediction of various congestion systems.

In N-policy queuing system, the server (repairman) starts service (repair) to arriving customers or items when the number reaches up to some fixed value say 'N. The various applications of thresholds models can be made in day-to-day as well as industrial scenarios which motivate us for implementing this model in our work(18).

2.3.7 Arrivals and service

Congestion in a queueing system refers to the traffic customers on the system and depend on the system irregularities not just on average properties. In other words, the number of customers in the queue depends on the complete probabilistic description of the arrival and service processes. The customers placed on the queue then the server treat these demands. It would seem evident that, if the average arriving capacity of customers is greater than the system service capacity, the system will break down since unbounded queues will form. On the other hand, if the average arrival rate is less than the system service capacity,

Studies of Markov chain and queueing models

then, the current customers being served before new customer arrives; thus, we still get queues. Even when the average arrival and service rates are held constant, an increase in the variation of arrivals or service increases or decrease the congestion. Furthermore, as the average demand tends to the system service capacity, the effects of the fluctuations are magnified(14).

These fluctuations are described in terms of probability distributions. Thus, we use elementary probability theory to predict average waiting times, average queue length, distribution of queue length, etc., on the basis of

- The arrival pattern of customers to the resource.
- The service pattern of customers.
- The scheduling algorithm or the manner in which the next customer to be served is chosen.

In standard queueing models, the buffer size to hold customers waiting for service is limited. When the queue has reached its maximum capacity, it is said to be “full” customers who arrive to find the queue full are said to be “lost”.

- **The Arrival Process:**

The customer arrival process may be described in two ways:

- the number of arrivals per unit time (the arrival rate);
- the time between successive arrivals (the interarrival time).

We use the variable λ to denote the mean arrival rate. In this case, $1/\lambda$ denotes the mean time between arrivals. If the arrival pattern is not deterministic, the input process is a stochastic process, in that case, we need the associated probability distribution. The probability distribution of the interarrival time of customers is denoted by $A(t)$ where

$$A(t) = \text{Prob} \{ \text{time between arrivals} \leq t \} \quad \dots(2-18)$$

And

$$\frac{1}{\lambda} = \int_0^{\infty} t dA(t) \quad \dots(2-19)$$

Studies of Markov chain and queueing models

Where $dA(t)$ is the probability that the interarrival time is between t and $t + dt$, here we assume that these interarrival times are independent and identically distributed, which means that only $A(t)$ is of significance. If there are different types of customers, meaning each customer has its class, then each class may have its probability distribution function to describe its arrival process. The manner in which the arrival pattern changes in time may be important (e.g., the number of customers who arrive at a supermarket may be greater in the late afternoon than in the early morning.) when arrival pattern that does not change with time (i.e., the form and values of the parameters of $A(t)$ are time-independent) is said to be a homogeneous arrival process. If it is invariant to shifts in the time origin, it is said to be a stationary arrival process(19).

- **The Service Process**

Like we describe the arrival pattern, the service pattern may be described by a rate, the number of customers served per unit time, or by the time required to serve a new customer. The parameter μ is used to denote the mean service rate, and hence $1/\mu$ denotes the mean service time per one customer. We shall use $B(x)$ to denote the probability distribution of the demand placed on the system, i.e.,

$$B(x) = \text{Prob}\{\text{Service time} \leq x\}. \quad \dots(2-20)$$

Thus

$$\frac{1}{\mu} = \int_0^{\infty} x dAB(x) \quad \dots(2-21)$$

Where $dB(x)$ is the probability that the service time is between x and $x + dx$, notice that the service time is equal to the length of time spent in service and does not include the time spent waiting in the queue. Furthermore, service rates are conditioned on the fact that the system is not empty. If the system is empty, then the server must be idle.

Studies of Markov chain and queueing models

Although it is usual to associate the service time distribution with the server, the service time is actually the time that is requested or needed by the customer who is taken into service. Obviously, it does not make sense for a server to arbitrarily dispense service to customers without regard for their needs(19).

The service may be batch or single. For the batch service, several customers can be served simultaneously as is the case, for example, of customers who wait in line for taxis or buses, customers in the market. Also, the service rate may depend on the following factors:

- The number of customers present in the queue (called state-dependent or load-dependent service). For example, a server may speed up when the queue grows or starts to become full or slow down as it starts to empty.
- The time (called time-dependent or nonhomogeneous service). This is, for example, the case of a server that in the morning starts slowly and during the day start to become faster because of its workload.
- The total number of servers available at a queueing system is denoted by c . When there are more than one server two cases are possible:
 - Each server has its queue or buffer. For example, each supermarket checkout lane has its queue. However, the effect of jockeying and lane changing need that a supermarket checkout system to be more accurately modelled as a single queue in front of all checkout lanes.
 - There are fewer queues than servers. In most cases, there is a single queue for all servers. For example, a single queue usually forms in front of multiple bank tellers. The servers may or may not be identical, i.e., $B(x)$ may be different for different servers. Also, given multiple classes of customers, the same server may give different service to different classes.

The capacity that a service facility has to hold waiting for customers (called the system capacity) in many cases or models taken to be infinite. When this is not the case, then the system is referred to as a finite queueing system.

Studies of Markov chain and queueing models

2.3.8 Poisson arrivals and exponential service

In stochastic modelling, numerous random variables are frequently modelled as exponentials.

This is because it close to the real-world scenarios These include

- interarrival time λ ,
- service time μ ,
- time to failure of a component, and
- time required to repair a component.

The assertion that the above distributions are exponential should not be taken as fact, but as an assumption. Experimental verification of this assumption should be sought before relying on the results of any analyses that use them. the cumulative distribution function for an exponential random variable, X , with parameter $\lambda > 0$, is given by:

$$F(x) = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \dots (2-22)$$

And its corresponding probability density function obtained simply by taking the derivative of $F(x)$.

with respect to x , is:

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \dots (2-23)$$

If the Poisson provides an appropriate description of the number of occurrences per interval of time, then the exponential will provides a description of the length of time between occurrences (19).

2.3.9 Performance measures

When we analyze a queueing system, we do so for the purpose of evaluating the values of certain system properties. For example, we may want to find:

- the number of customers in the system;
- the mean waiting time in the buffer;
- the length of a busy or idle period;
- the duration of one cycle.

These are called measures of effectiveness. They are all random variables and, whereas we might wish to know their complete probabilistic descriptions (i.e., their PDFs).

- **Number of customers**

N is a random variable that represents a number of customers in the system at steady state. The probability that at steady state the number of customers present in the system is n is denoted by p_n (14),

$$p_n = \text{Prob}\{N = n\},$$

and the average number in the system at steady state is

$$L = E[N] = \sum_{i=0}^{\infty} np_n.$$

Within the queueing system, customers may be present in the queue waiting for their turn to receive service, or they may be receiving service. We shall let N_q be the random variable that describes the number of customers waiting in the queue and we shall denote its mean by $L_q = E[N_q]$ (14).

Studies of Markov chain and queueing models

- **System Time and Queueing Time**

The time that a customer spends in the system, from the instant of its arrival to the queue to the instant of its departure from the server, is called the response time or sojourn time. We shall denote the random variable that describes response time by R , and its mean value by $E[R]$. The response time is composed of the time that the customer spends waiting in the queue, called the waiting time, plus the time the customer spends receiving service, called the service time. We shall let Wq be the random variable that describes the time the customer spends waiting in the queue, and its mean will be denoted by $E[Wq]$. (14)

- **System Utilization**

In a queueing system with a single server ($c = 1$), the utilization U is defined as the fraction of time that the server is busy. If the rate at which customers arrive at, and are admitted into, a queueing facility is λ and if μ is the rate at which these customers are served, then the utilization is equal to λ / μ . Over a period of time T , this queueing system, in steady state, receives an average of λT customers, which are served in an average of $\lambda T / \mu$ seconds. In many queueing systems, the Greek letter ρ is defined as $\rho = \lambda / \mu$ and consequently is identified with the utilization.

However, λ is generally defined as the arrival rate to the system, and this may or may not be the rate at which customers actually enter the queueing facility. Thus, it is not always the case that λ / μ correctly defines utilization. Some customers may be refused admission (they are said to be “lost” customers) so that the effective arrival rate into the queueing facility is less than λ and hence the utilization is less than $\rho = \lambda / \mu$. However, unless stated otherwise, we assume that all customers who arrive at a queueing facility are admitted. In a G/G/1 queue where p_0 is the probability that the system is empty, it must follow that $U = 1 - p_0$. In a stable system (i.e., one in which the queue does not grow without bound), the server cannot be busy 100% of the time. This implies that we must have $\lambda / \mu < 1$ for the queueing system to be stable. Thus, in any time interval, the average number of customers that arrive must be strictly less than the average number of customers that the server can handle.

Studies of Markov chain and queueing models

In the case of queueing systems with multiple servers ($c > 1$), the utilization is defined as the average fraction of servers that are active—which is just the rate at which work enters the system divided by the maximum rate (capacity) at which the system can perform this work, i.e., $U = \lambda/(c\mu)$. In multiserver systems, it is usual to define ρ as $\rho = \lambda/(c\mu)$ with the same caveat as before concerning the identification of ρ as the utilization (14).

- **System Throughput**

The throughput of a queueing system is equal to its departure rate, i.e., the average number of customers that are processed per unit time. It is denoted by X . In a queueing system in which all customers that arrive are eventually served and leave the system, the throughput is equal to the arrival rate, λ . This is not the case in queueing systems with finite buffer, since arrivals may be lost before receiving service (14).

- **Traffic Intensity**

We define the traffic intensity as the rate at which work enters the system, so it is therefore given as the product of the average arrival rate of customers and the mean service time, i.e., $\lambda\bar{x} = \lambda/\mu$, where $\bar{x} = 1/\mu$ and μ is the mean service rate. Notice that in single-server systems, the traffic intensity is equal to the utilization. For multiple servers, the traffic intensity is equal to cU (14).

2.3.10 Birth-Death processes: The M/M/1 queue

Birth–death processes are continuous-time Markov chains with a very special structure. If the states of the Markov chain states are indexed by the integers $0, 1, 2, \dots$, if we are in the state we permitted only to move from this state to its nearest neighbours, namely, states $i - 1$ and $i + 1$. As for state $i = 0$, on exiting this state, the Markov chain must enter state 1. Such processes are also called skip-free processes because to go from any state i to any other state j , and each intermediate state must be visited: no state between these two can be skipped. Birth-death processes arise in a variety of simple single-server queueing systems, and due to its particular structure makes finding their stationary distributions states relatively easy to compute(19).



Figure 2-8 The M/M/1 queue (14)

2.3.11 Description and steady-state solution

the state of an M/M/1 queue at any time is completely described by specifying the number of customers present in the system. We shall use the integers $0, 1, 2, \dots$ to represent these states accurately: n denotes the state in which there are n customers in the system, including the one in service. We would like to be able to compute the state probabilities, i.e., the probability that the system is in any given state n at any time t . We write these as

$$p_n(t) = \text{Prob}\{n \text{ in system at time } t\}.$$

This is often a difficult task, even for this simplest of queueing processes. Instead, at least for the moment, we shall look for the steady-state probabilities,

$$p_n = \lim_{t \rightarrow \infty} p_n(t).$$

If this limit exists, then the probability of finding the system in any particular state eventually becomes independent of the starting state, so that no matter when we query the system after it settles into a steady state, the probability of finding n customers present does not change. The steady state probabilities p_n can be interpreted as the probability of finding n customers in the system at an arbitrary point in time after the process has reached a steady state. It is not true that all systems reach steady state, i.e., for some queueing systems it is possible that $\lim_{t \rightarrow \infty} p_n(t)$ may not yield a true probability distribution. Finding steady state by calculating limit is difficult and complex task instead, we can use an infinite infinitesimal generator to find it. (20)

Studies of Markov chain and queueing models

2.3.12 Matrix formulation of the M/M/1 queue

The M/M/1 queue with service rate μ and arrival rate λ has the following infinite infinitesimal generator:

$$Q = \begin{pmatrix} -\lambda & \lambda & & & & & \\ \mu & -(\lambda + \mu) & \lambda & & & & \\ & \mu & -(\lambda + \mu) & \lambda & & & \\ & & \mu & -(\lambda + \mu) & \lambda & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & & \ddots & \ddots \end{pmatrix}.$$

In the notation used for Markov chains, we have $\pi Q = 0$ (with $\pi_i = p_i$ for all i), $\sum \pi_i = 1$ and it is obvious that $-\lambda\pi_0 + \mu\pi_1 = 0$, i.e., that $\pi_1 = (\lambda/\mu)\pi_0$.

In general, we have $\lambda\pi_{i-1} - (\lambda + \mu)\pi_i + \mu\pi_{i+1} = 0$,

from which, by induction, we may derive

$$\pi_{i+1} = ((\lambda + \mu)/\mu)\pi_i - (\lambda/\mu)\pi_{i-1} = (\lambda/\mu)\pi_i.$$

Thus, once π_0 is known, the remaining values π_i , $i = 1, 2, \dots$, maybe determined recursively just as before. For the M/M/1 queue it has already been shown that the probability that the system is empty is given by $\pi_0 = (1 - \lambda/\mu)$.

Observe that the coefficient matrix is tridiagonal. and that once p_0 is known, the solution is just a forward elimination procedure(14). However, we show this formulation at this time because it will become useful in other, more complex cases.

- **Performance Measures**

We now turn our attention to computing various performance measures concerning the M/M/1 queue, such as mean number in system, mean queue length, and so on

- Traffic intensity: $p = \lambda/\mu$

Studies of Markov chain and queueing models

- Probability of n jobs in the system: $p_n = (1 - \rho)\rho^n$
- Mean number of jobs in the system: $E[n] = \rho/(1 - \rho)$
- Mean number of jobs in the queue: $E[n_q] = \rho^2/(1 - \rho)$
- Mean response time: $E[r] = (1/\mu)/(1 - \rho)$
- Mean waiting time: $E[w] = \rho(1/\mu)/(1 - \rho)$
- Mean number of jobs served in one busy period: $\frac{1}{1-\rho}$
- Mean busy period duration: $\frac{1}{\mu(1-\rho)}$

2.3.13 The M/M/c queue

An M/M/C queue is a shorthand notation for Markovian arrival rate, Markovian Service Rate, and C the number of resources is a system where arrivals form a single queue and are governed by a Poisson process, and job service times are exponentially distributed. (21)

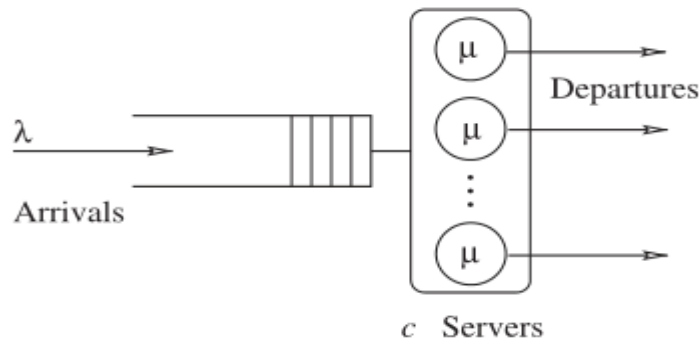


Figure 2-9 The M/M/C queue (14)

- **Performance Measures for the M/M/c Queue**

- Traffic intensity: $\rho = \lambda/(m\mu)$

Studies of Markov chain and queueing models

- Probability of n jobs in the system:
$$p_n = \begin{cases} p_0 \frac{(m\rho)^n}{n!} & n < m \\ p_0 \frac{\rho^n m^m}{m!} & n \geq m \end{cases}$$
- Mean response time: $E[r] = \frac{1}{\mu} \left(1 + \frac{\rho}{m(1-\rho)} \right)$

Conclusion

In this chapter, we have presented mathematical concepts necessary to the understanding of and Markov chain and queueing patterns. We introduce Markov chains and define some of its characteristics then we talked about the deferent types of Markov chains discrete-time Markov chain and continuous-time Markov chain and their characteristics so as concepts like ergodicity, infinitesimal generator and stationary distribution

And then We did see some models of queueing systems. We addressed the classical queue theory, and we have seen that the queueing model's standards do not allow to describe the real behaviour of the servers, where does the need to use vacation queueing models for modelling the vacation. We have seen why we use vacations, and it implemented in the real world, we also define the queueing policies and present the queueing system with a threshold.

Chapter 3 Modeling of the system

3.1 Introduction

Cloud computing is becoming a dominant field. The cloud computing services section is a need, especially for businesses as well as individuals. This is due to the divergence of the services.

In the first chapter we talked about cloud computing and the various services and the main architecture of the cloud we also talked about energy in the cloud section and this to understand the field and can describe a model with the cloud needs. Then we present Markov chains in the second chapter and focused on CTMC proprieties and also queueing models, and this chapter is the key that we can model the system correctly and can resolve the system.

In this chapter we are going to do two models, the first model, we represent the system with a working vacation and threshold policy, and the second model is the same as the first one with the addition of sleep-delay timer.

3.2 Related works

Cloud Computing is gaining much popularity nowadays, and it is getting implemented in many organizations very fast, and that need for cloud computing leads to many new ideas, and this section keeps in innovation, which leads to various research in the cloud computing section. One of the most trending cloud computing research topics is green cloud computing, which is saving energy and reducing the Carbone footprint.

E. Feller, L. Ramakrishnan, C. Morin, in "Performance and energy efficiency of big data applications in cloud environments: A Hadoop case study." They evaluate Hadoop

Modeling of the system

performance in the traditional model of collocated data and compute services considering the impact of separating the services. Their evaluation shows that: performance on physical clusters is significantly better than on virtual clusters, and application completion progress correlates with the power consumption. (22)

Xia, Y.; Zhou uses Dynamic voltage scaling (DVS) by exploiting the cloud data center's hardware characteristics to save energy by lowering the supply voltage and operating frequency. (23)

Chen, Y et al. also used dynamic voltage and scaling frequency DVFS that predicts the best voltage/frequency setting for the system. Their results show that the proposed DVFS could predict the suitable frequency which gives significant energy consumption and performance. (24)

All those previous researches talked about reducing energy without engaging the idea that the VMs could go to sleep state, which reduces energy. The next researches highlight this idea with various models.

Lawanyashri, M, uses energy-saving based on threshold activation in a wireless sensor network, based on a finite buffer queueing model with N-policy, which means that initialization of transmission starts after reaching the n packets waiting in the buffer.

Balusamy, B et al., proposed a system that uses both vacation and a threshold policy to control the workload level of each virtual machine in the data center, and reduces the energy consumption and cost accordingly. (25)

Xi, Wang et al, present a performance analysis and a system optimization of a cloud computing system with an energy efficient task scheduling strategy for satisfying the service level agreement of cloud users while at the same time improving the energy

Modeling of the system

efficiency in cloud computing system. so they proposed an energy-aware task scheduling strategy based on a sleep-delay timer and a waking-up threshold. so they combine a vacation-delay with a N-policy. (26)

Based on these related works we study a performance analysis and a system optimization of a cloud computing system with an energy efficient scheduling strategy, so we propose a strategy based on sleep-delay timer and a threshold policy with a limited buffer and no sleep period which mean that the only condition for sleep to awake is the threshold. And to observe the impact of the sleep delay timer on energy and performance, we model two models the first without the sleep delay timer and the second within the sleep delay time.

3.3 Cloud model with n-policy

3.3.1 Description

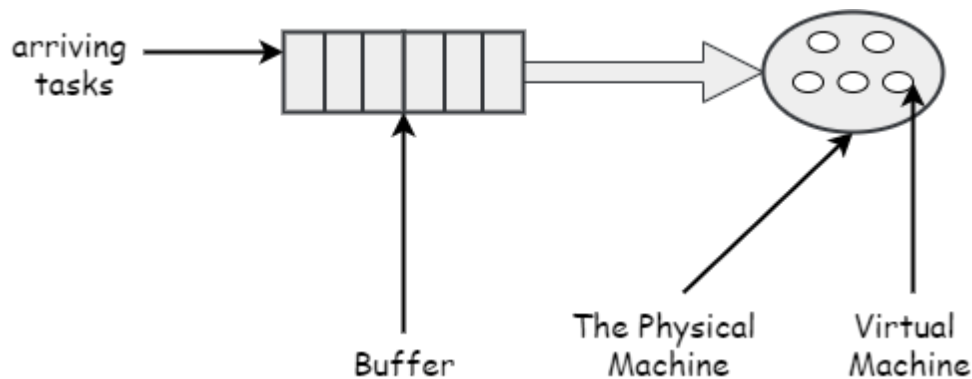


Figure 3-1 The general representation of the system

We propose in this section modelling of the physical machine on cloud system with infrastructure as a service (IaaS) using the formalism of queues with vacation and threshold

Modeling of the system

policies. Our model is general and does not fit to a specific cloud model, cloud topology or cloud provider company. Moreover, it concerns a general physical machine on the cloud, so that is valid for any machine on the cloud.

Each physical machine on the cloud have a limited size buffer (N) and a group of servers (S), these s servers represent the virtual machines (VMs) include in the physical machine, each one of these servers can handle a single task from the internet. To simplify the model, we assume that the tasks have equal size and the same service time. Each server from physical machine receives a task from buffer meaning one buffer for all the servers after it handles that task goes back to an idle state waiting for the arrival a new task to handle it.

The new characteristics in this model are that we take into consideration the limitation of physical machine buffer with n tasks only. If any task comes after the buffer having n task, this task will be lost. The queue also has a threshold policy; in this case, the servers in our model stay in a sleep state until a specific number of tasks n enter the buffer then all servers go to an idle state. The choice of the limited buffer size is motivated by the fact that in the real physical machine cloud buffer size is big but still limited. Thus, our model combines buffer limiting, queue with vacation and threshold policy.

Let's take a physical machine. In the initial state, our servers (VMs) are in a sleep state, and the buffer is empty, the tasks arrive with exponential arrival rate λ , once n task enters the buffer, all

servers in our model shift from the sleep state to the idle state. After that, each server starts to handle one task moving from idle to busy state with Poisson service rate μ in FIFO policy, and then once the complete server is handling that task, it moves again to an idle state waiting for a new task to come. On the other hand, if the buffer is full, all task arrive after that are lost, until the buffer becomes empty again then server shift from idle to sleep state thus, we return back to our initial state.

Modeling of the system

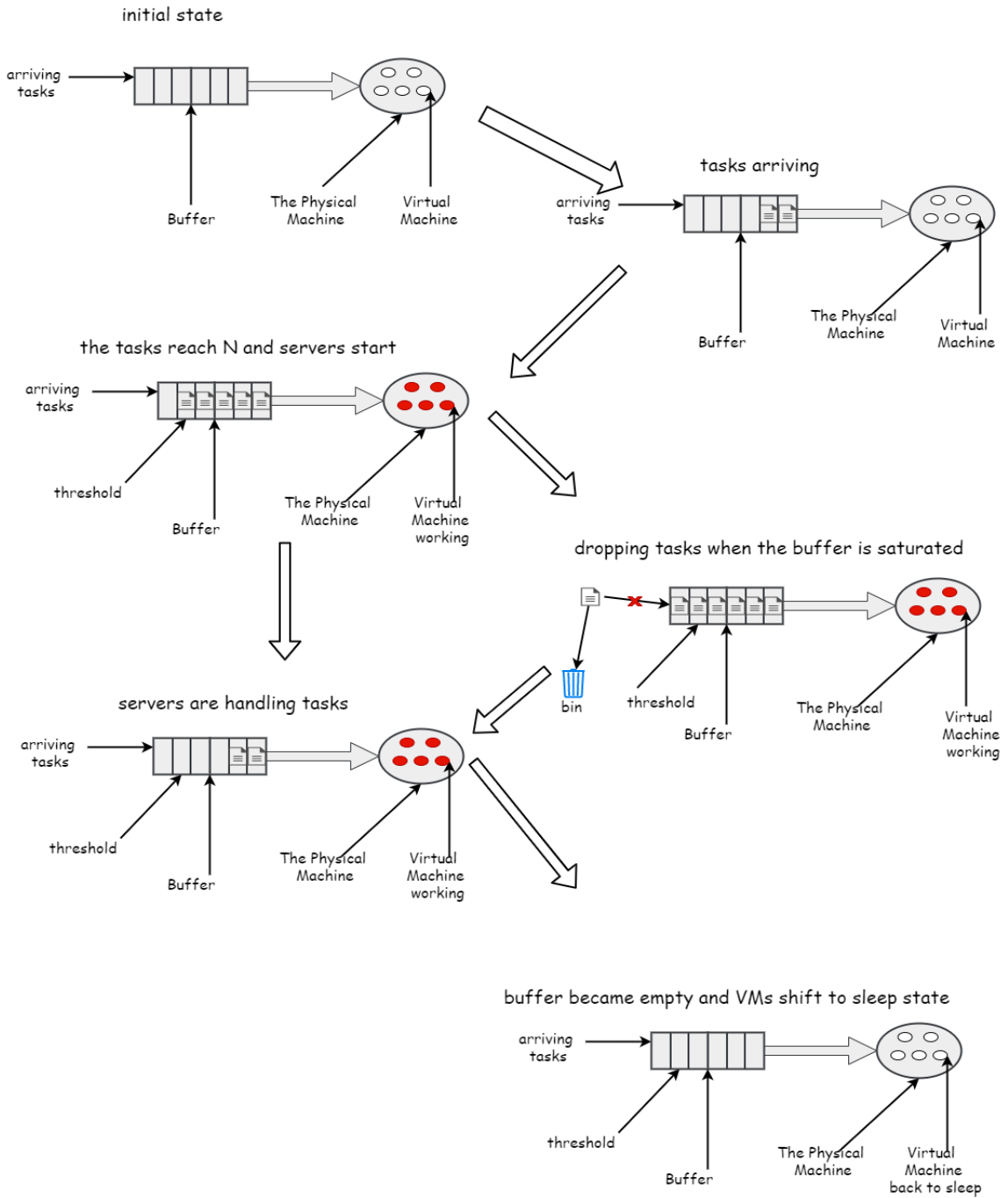


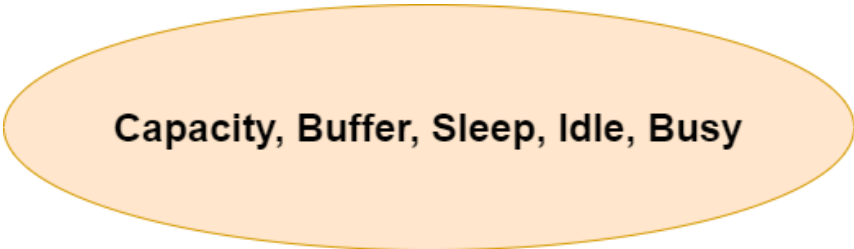
Figure 3-2 description of some model 1 transition

Modeling of the system

3.3.2 Continuous time Markov chain of system Cloud model with n-policy

As we talked above about some possible transition and states from our model, now we going to provide the Continuous-time Markov chain of system model that contain all possible state and transitions that can accrue in a real-world scenario, meaning, it is a complete model description.

We model our system using 5 different variables. Thus, every Markov chain state is described by the 5 variables: capacity, buffer, sleep, idle, busy.



Capacity, Buffer, Sleep, Idle, Busy

For abbreviation, we can index each one of these variables by a single letter. Our 5-dimension Markov chain state will be described by these letters (I, j, k, l, m)

- I: represent the limited size (capacity) of the buffer, a maximum of I task can be saved in the buffer.
- J: represent the current number of tasks in the buffer
- k: represent that a server (VM) is in sleep state, that state consumes the minimum energy to switch to busy when n task enters the buffer
- l: represent that a server (VM) is in idle state, server move to that state after completing the task, waiting for the arrival of a new task to start immediately handling it.
- m: represent that a server (VM) is in a busy state, in that state server start to handling the task.

Modeling of the system

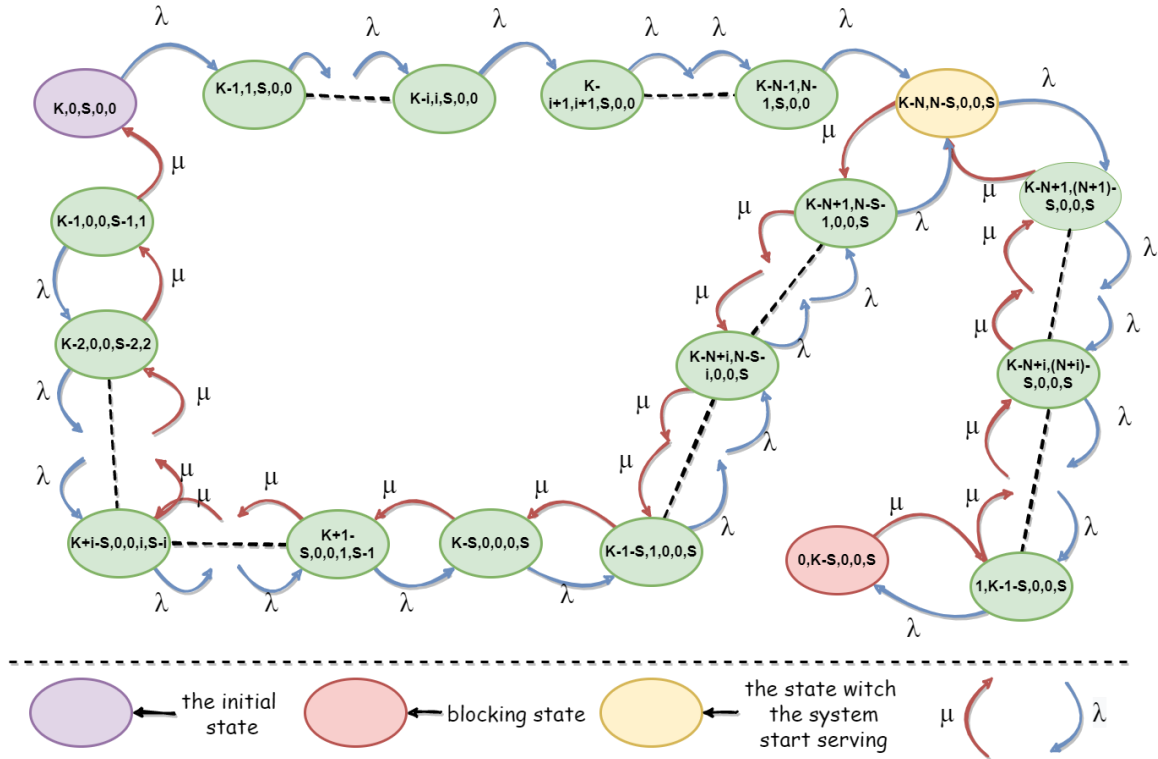


Figure 3-3 Model 1 CTMC with 5 dimensions

The initial Markov $(0, S, 0, 0, K)$ state describes our system when we first started it, in that state, we have 0 task in the buffer, and all servers are in sleep state, we can see that transitions from state to state happen only with the arrival rate λ and service rate μ . From initial state we keep moving by λ from state to the next state until we reach the state $(K-N, N-S, 0, 0, S)$, that state represents that n number of tasks enter in the buffer (number of tasks in the buffer reached the threshold), in that state, servers shift from a sleep state to busy state begin to handle s tasks from the buffer one by one in the duration of service rate μ leaving the only $n-s$ task in the buffer. Thus, we have to transition available:

transition with μ : when we move to the next state with μ transition, we finish serving the current task in the server allowing another task to be handled next. With this transition from state to the next state buffer decrease each transition until it becomes empty meaning, we service all the tasks, and then servers start to decrease each time going from busy to idle until we reach our initial state. We can go back from the next state to the current state at any of these transition by λ .

Modeling of the system

Transition with λ : or we can go the next state by λ transition, with this transition, new tasks are coming to the buffer before we finish serving the current ones. In each transition, the number of tasks in the buffer increase by one and servers stays in busy state until We reach to blocking state $(0, K-S, 0, 0, S)$ like we seed earlier every task coming in this state are lost (buffer overflow). We can go back from the next state to the current state at any of these transition by μ .

3.3.3 Resolution

Now, after we have our continuous-time Markov chain model, now, we want to find a system in the steady state. To do this, we have to analyze the CTMC in the steady-state. One of the key steps is to construct an infinitesimal generator. Since the state space increases as a function of the buffer size and the queue threshold. we seek to reduce the system solution cost, by designing algorithms that compute the infinitesimal generator Q directly as a function of the VM parameters and reduce Markov chain dimension.

3.3.4 Analysis of Cloud model with n-policy

Considering the proposed Model, whatever the value of K (The buffer size) and based on the fact that the VM can alternate between idle, busy and sleep states, we derive the following equations:

$$\begin{cases} T_i(Capacity) + T_i(Buffer) + V_i(Busy) = K \\ V_i(Idle) + V_i(Busy) + V_i(Sleep) = S \end{cases} \quad \dots(3-1)$$

Where:

- K is the buffer size, and S is the number of servers.
- Let $T_i(X)$ represent the number of tasks in state i in X ($X= \{capacity, buffer\}$)
- Let $V_i(X)$ represent the number of servers in state i in X ($X= \{idle, busy, sleep, sleep-delay\}$)

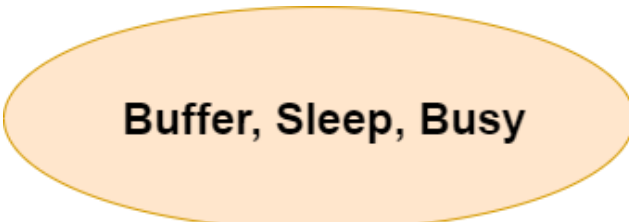
Modeling of the system

Using equation 1, we notice that the system state at steady-state can be described by means of 3 components (i, j, l) where:

- i : represents the number of tasks in the buffer. $0 \leq i \leq K$
- j : represents the number of servers in the place sleep. $0 \leq j \leq S$
- l : represents the number of servers in the place busy. $0 \leq l \leq S$

Thereby, having the (i, j, l) we are able to deduce the capacity and idle, given that:

$$\begin{cases} T_i(\text{Capacity}) = K - T_i(\text{Buffer}) - V_i(\text{Busy}) \\ V_i(\text{Idel}) = S - V_i(\text{Busy}) - V_i(\text{Sleep}) \end{cases} \dots(3-2)$$



Buffer, Sleep, Busy

After reduction, we got this Markov chain with 3 dimensions (i, j, l) and with the same number of states.

Modeling of the system

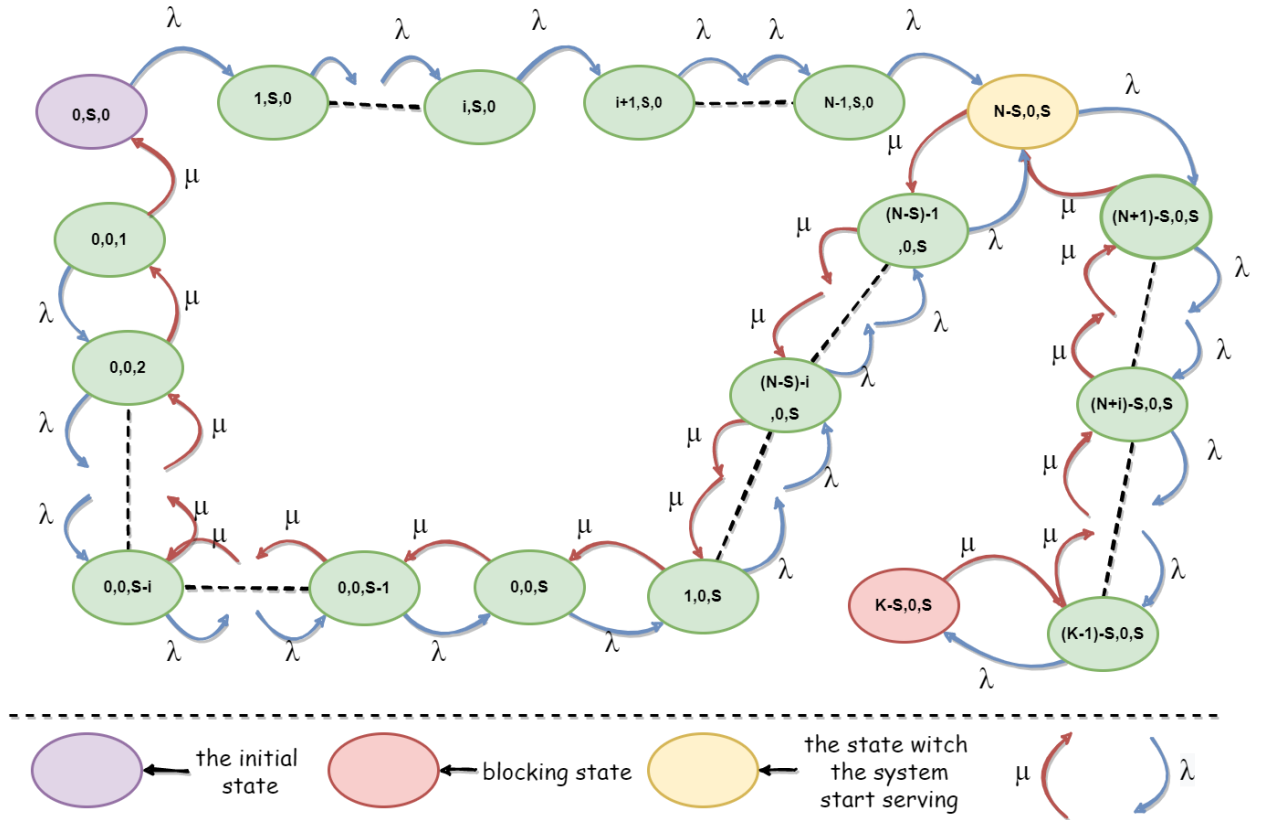


Figure 3-4 Model 1 CTMC with 3 dimensions

By analyzing the CTMC above, we have recognized that the total number of states of the CTMC is equal to M , where $M = K + N$. Hence, the infinitesimal generator is an $M \times M$ matrix Q , which can be constructed as follows:

$$Q[(i, j, l), (x, y, z)] = \begin{cases} q[(i, j, l), (x, y, z)] & \text{if } (i, j, l) \neq (x, y, z) \\ - \sum_{(i, j, l) \neq (k, m, n)} q[(i, j, l), (k, m, n)] & \text{if } (i, j, l) = (x, y, z) \dots (3-3) \end{cases}$$

The rates $q[(i, j, l), (x, y, z)]$ are the transition rates from the state (i, j, l) to state (x, y, z) , and are given by:

Modeling of the system

- $[0 \leq i \leq N - 1, j = s, l = 0]: (i, j, l) \xrightarrow{\lambda} (i + 1, j, l)$: The VM is in sleep state and number of tasks is less than the threshold N ;
- $[i = N, j = 0, l = s]: (i, j, l) \xrightarrow{\lambda} (i + 1, j, l)$ and $(i, j, l) \xrightarrow{\mu_1} (i - 1, j, l)$;
- $[N < i < K, j = 0, l = s]: (i, j, l) \xrightarrow{\lambda} (i + 1, j, l)$ and $(i, j, l) \xrightarrow{\mu_1} (i - 1, j, l)$;
- $[i = K, j = 0, l = s]: (i, j, l) \xrightarrow{\mu_1} (i - 1, j, l)$;
- $[0 < i < N, j = 0, l = s]: (i, j, l) \xrightarrow{\lambda} (i + 1, j, l)$ and $(i + 1, j, l) \xrightarrow{\mu_1} (i, j, l)$;
- $[i = 0, j = 0, l = s]: (i, j, l) \xrightarrow{\lambda} (i + 1, j, l), (i + 1, j, l) \xrightarrow{\mu_1} (i, j, l - 1)$;
- $[i = 0, j = 0, l = 0 < l < s]: (i, j, l) \xrightarrow{\lambda} (i, j, l + 1), (i, j, l + 1) \xrightarrow{\mu_1} (i, j, l)$;
- $[i = 0, j = 0, l = 1]: (i, j, l) \xrightarrow{\lambda} (i, j, l + 1), (i, j, l) \xrightarrow{\mu_1} (i, s, l - 1)$;

Therefore, the infinitesimal generator Q can be computed by mean of the algorithm below:

3.3.5 Infinitesimal generators

```
for  $L \leftarrow 0, S$  do  
  if  $(L = 0)$  then  
     $j \leftarrow s$   
    for  $i \leftarrow 0$  to  $n - 1$  do  
       $Q[(i, j, L), (i + 1, j, L)] \leftarrow \lambda$   
    end for  
  else  
     $i \leftarrow n - s, j \leftarrow 0, L \leftarrow s$ 
```

Modeling of the system

```

        for  $i \leftarrow N - S$  to  $K - S - 1$  do
 $Q[(i, j, L), (i + 1, j, L)] \leftarrow \lambda$ 
 $Q[(i, j, L), (i - 1, j, L)] \leftarrow \mu$ 
        end for
 $i \leftarrow K - S$ 
 $Q[(i, j, L), (i - 1, j, L)] \leftarrow \mu$ 
 $i \leftarrow N - S - 1$ 
        for  $i \leftarrow N - S - 1$  to 1 do
 $Q[(i, j, L), (i + 1, j, L)] \leftarrow \lambda$ 
 $Q[(i, j, L), (i - 1, j, L)] \leftarrow \mu$ 
        end for
 $i \leftarrow 0, L \leftarrow s$ 
 $Q[(i, j, L), (i + 1, j, L)] \leftarrow \lambda$ 
 $Q[(i, j, L), (i, j, L - 1)] \leftarrow \mu$ 
        For  $L \leftarrow S - 1$  to 2 do
 $Q[(i, j, L), (i, j, L - 1)] \leftarrow \mu$ 
 $Q[(i, j, L), (i, j, L + 1)] \leftarrow \lambda$ 
        end for
 $L \leftarrow 1, i \leftarrow 0, j \leftarrow 0$ 
 $Q[(i, j, L), (i, j + s, L - 1)] \leftarrow \mu$ 
 $Q[(i, j, L), (i, j, L + 1)] \leftarrow \lambda$ 

endif
end for
```

3.4 Cloud model with n policy and sleep delay state

3.4.1 Description:

As we did in model 1 our system is remaining the same with some changeset, we did propose in model 1 a modulization of a single physical machine on cloud system with infrastructure as a service using the formalism of queues with vacation and threshold policies. And we did say that Our model is general and does not fit to a specific cloud model, cloud topology or cloud provider company. so, it concerns a general physical machine on the cloud, so that is valid for any machine on the cloud.

In model 2 we introduce sleep-delay parameter β , so now our model consists of a queue with vacation and threshold policy as well as a sleep-delay parameter β .

After that we describe the system as follow, each physical machine on the cloud have a limited size buffer (K) and a number of servers (S), these S servers represent are virtual machines in the physical machine, and we did assume that the tasks have equal size and the same service time. Each server from physical machine receives a task from the buffer after it handles that task the virtual machine goes back to sleep state waiting for a new task to arrive and handle it.

Moreover, we talked about the characteristics that we take into consideration which are the limitation of physical machine buffer with n tasks only. If any task comes after the buffer is saturated with k tasks, the system is going to drop that task. We did say the queue have a threshold policy, and in this case, the servers in the first model and also the second model stay in a sleep state until a specific number of tasks N (the threshold) enter the buffer then all servers go to the idle state. Furthermore, we assume that the limited buffer size is related with the fact that in the real physical machine, the cloud buffer size is limited, so in addition to what we did mention before we suggest the sleep-delay period, so when the servers are busy, and the tasks end the servers go to sleep-delay state for taking some time before goes to a sleep state in the meanwhile if any tasks are arriving the servers go back too busy to handle the tasks and if the timer expires the tasks goes to sleep Thus, our

Modeling of the system

model combines the first model characteristics, buffer limiting, queue with vacation, threshold policy, and adding the sleep-delay timer.

Our system model is still working the same, a physical machine, in the initial state the servers (VMs) are in sleep state and the buffer is empty, the tasks arrive with exponential arrival rate λ , once the number of tasks reaches n task in the buffer, all servers change their state from sleep idle state. After that, servers start to handle tasks moving from idle to busy state with Poisson service rate μ in FIFS policy, then once the servers complete handling tasks in the buffer and the buffer is empty, the servers start a sleep-delay timer if any tasks arrive before the timer ends the servers go back to the idle state and doing the same thing, in another hand if the sleep-delay timer is expired then the servers change their state to sleep so the system goes back to the initial state, and the servers will not be idle until the threshold is reaching again. On the other hand, if the buffer is full, all tasks arrive after that are dropped.

Modeling of the system

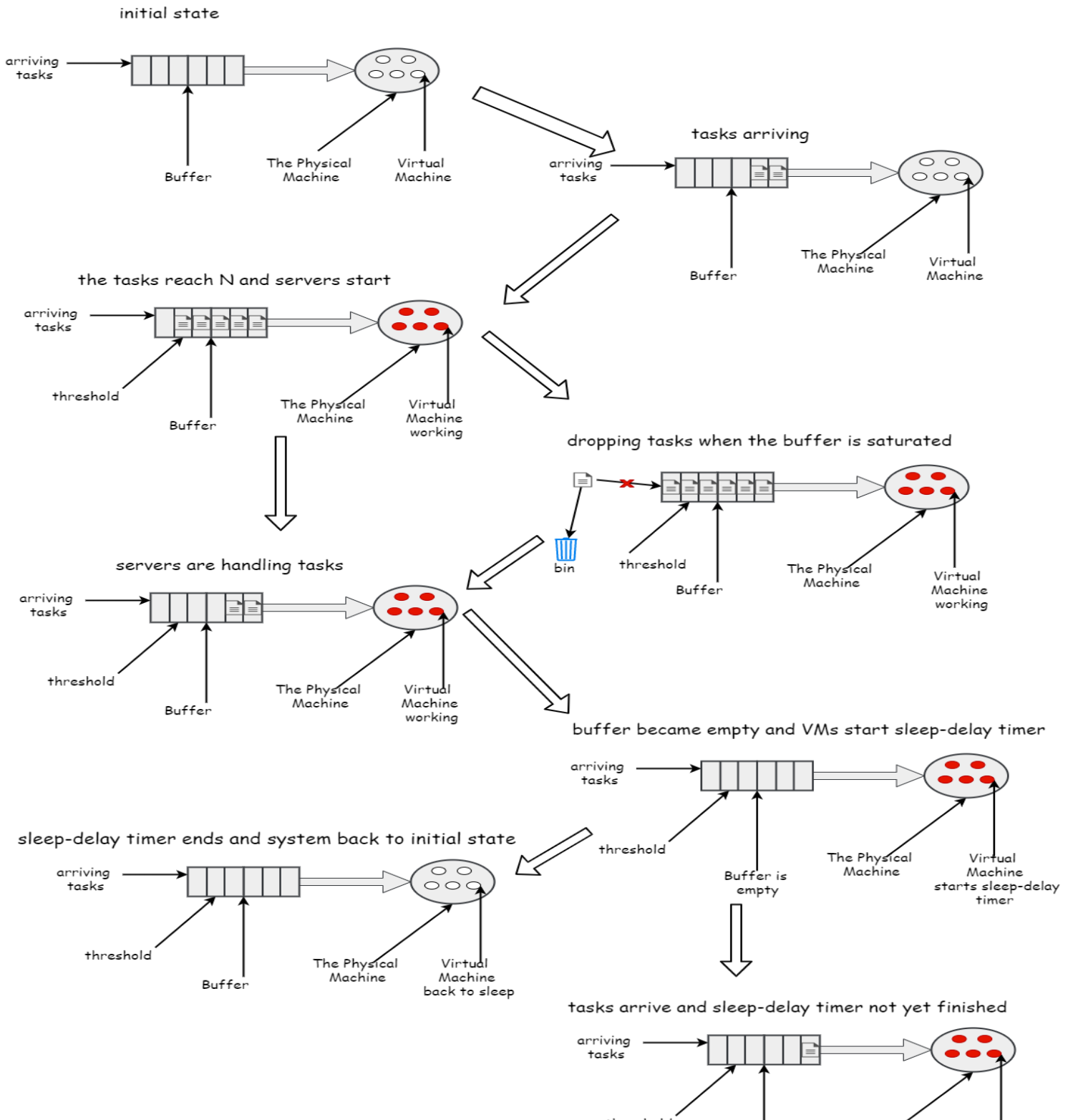


Figure 3-5 description of some model 2 transition

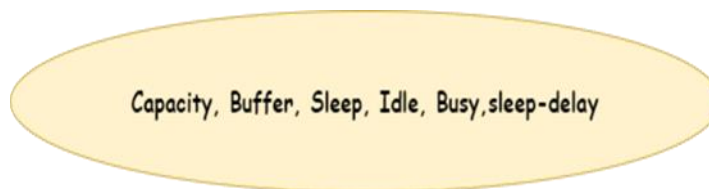
Modeling of the system

3.4.2 Continuous time Markov chain of system Cloud model with n-policy and sleep delay state

For simplify more and represent all the state of the system, we use a continuous-time Markov chain, and after explaining the system, we conclude that the system is a combination of:

- Capacity: which are the number of tasks that arrive with the rate
- Buffer: which represent the number of tasks in the buffer
- Sleep: represent the number of virtual machines that are sleep
- Idle: represent the number of servers that are awake ready to work
- Busy: represent the number of servers that are handling tasks
- Sleep-delay: represent the number of servers that are waiting for a timer to expire or a task to arrive
- S: is the number of servers in the physical machine
- K: is the number of tasks
- N: the threshold

Those 6 variables can represent one state as follow



The changeset in values of each variable in this representation represent another state.

Next, we represent the Markov chain with this modulization

Modeling of the system

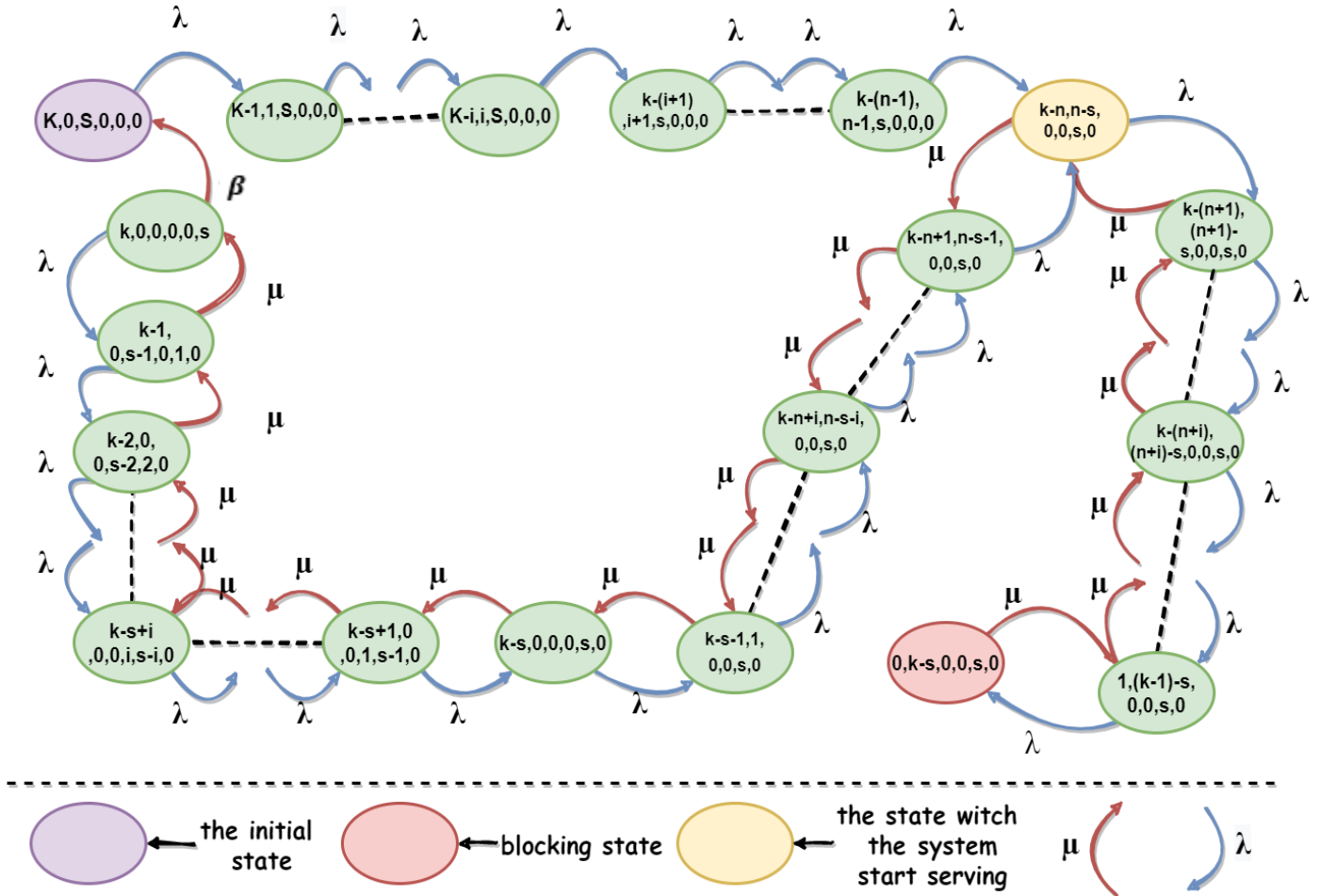


Figure 3-6 Model 2 CTMC with 6 dimensions

The initial state is $(K, 0, S, 0, 0, 0)$ which means that we have K available tasks that are not in the buffer yet, the number of tasks in the buffer is 0 and there are S servers that are asleep, all the rest places are empty so there are no servers in Idle, busy, and sleep-delay.

Next, with the arrival of a task, the capacity is $k-1$ and that task appears in the buffer and there are no changes in the rest of the system, tasks still arriving to the buffer by a rate of λ until it reaches the threshold N .

Modeling of the system

Now the system changes its behaviour, so the capacity again is now $k-n$, and the buffer reaches N in the same time servers are awake, and every server take a task to handle it, so the number that appears in the buffer is $N-S$, and all servers are busy so $\text{busy}=S$ and sleep remain 0, and others rest the same

We are now in the state that the service is started, from this state, the next step is either serving tasks with the rate μ or arriving tasks with the rate λ .

Let us take the approach of arriving tasks first the system stay receiving tasks and capacity -1 and buffer $+1$ until capacity is 0 then the buffer is full, so we reaches the blocking state any other task arrives now will be dropped or lost in our system we do not represent this to be obvious when capacity is 0 that means that the system has no other capacity to store or receive tasks.

When we take the service approach from the blocking state, and in every μ the system serve a task, after that it release space in the system, $\text{capacity}+1$ and $\text{buffer}-1$ the system keep serve until the buffer became empty here the capacity is $K-S$ because there is S tasks that are in busy with the servers, from knowing the servers that are handling tasks start to reduce so $\text{busy}-1$ $\text{capacity}+1$ and the servers that not busy is now idle do $\text{idle}+1$ the system keep like that until $\text{capacity}=k$, $\text{busy}=0$, $\text{idle}=s$ int the same time servers start sleep-delay timer so $\text{idle}=0$ and $\text{sleep-delay}=S$.

If no task arrives within the sleep-delay timer servers, go to sleep again, so the system is back to the initial state. Else the servers go back to busy again.

After modelling the system, we came to propose the resolution for this system in the next section.

Modeling of the system

3.4.3 Resolution

After understanding the system model, we denote that we can reduce our system variable with:

$$\begin{cases} T_i(Capacity) + T_i(Buffer) + V_i(Busy) = K & \dots (3) \\ V_i(Idle) + V_i(Busy) + V_i(Sleep) + V_i(Sleep - delay) = S & \dots (4) \end{cases} \dots (3-4)$$

We add $V_i(Busy)$ to the equation (4) because in our system, the busy means the V_m is handling a task so without the tasks in busy the tasks in buffer and capacity less than K .

Using equations (3):

$$T_i(Capacity) = K - (T_i(Buffer) + V_i(Busy)) \quad \dots (3-5)$$

$$V_i(Idle) = S - (V_i(Busy) + V_i(Sleep) + V_i(Sleep - delay)) \quad \dots (3-6)$$

Since we can get capacity and idle from busy, sleep, buffer and sleep delay, we can represent the model by following:

Buffer, Sleep, Busy, sleep-delay

Where:

- i : represent the number of tasks in the buffer $T_i(Buffer) \ 0 \leq i \leq K$
- j : represent the number of servers that are sleep $V_j(Sleep) \ 0 \leq j \leq S$
- l : represent the number of servers that are busy $V_l(Busy) \ 0 \leq l \leq S$
- d : represent the number of servers that are in sleep-delay $V_d(Sleep - delay) \ 0 \leq d \leq S$

Modeling of the system

after reducing the representation of the system, the CTMC is as follow:

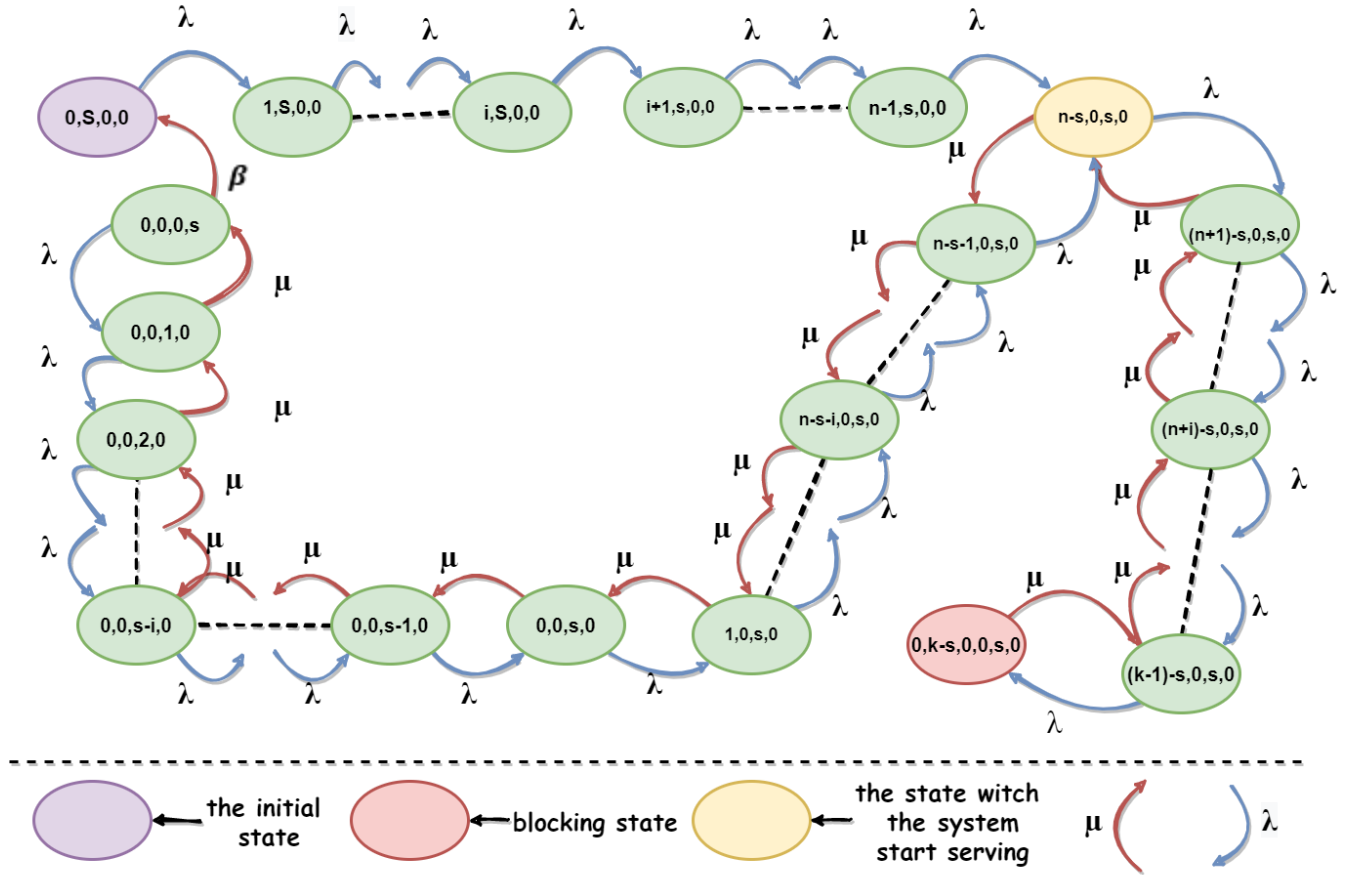


Figure 3-7 Model 2 CTMC with 4 dimensions

After analysing this CTMC, we conclude that the total number of tangible marking states M is: $M = K + N + 1$, and the infinitesimal generator Q dimensions equal to $M \times M$ which constructed as follows:

$$Q[(i, j, l, d), (x, y, z, v)] = \begin{cases} q((i, j, l, d), (x, y, z, v)), & \text{if } (i, j, l, d) \neq (x, y, z, v) \\ -\sum_{(i, j, l, d) \neq (a, b, c, d)} Q[(i, j, l, d), (a, b, c, d)] & \text{if } (i, j, l, d) = (x, y, z, v) \end{cases} \quad \dots (3-7)$$

Modeling of the system

3.4.4 Infinitesimal generator

```
for  $L \leftarrow 0, S$  do
  if  $(L = 0)$  then
     $j \leftarrow s$ 
     $d \leftarrow 0$ 
    for  $i \leftarrow 0$  to  $n-1$  do
       $Q[(i, j, L, d), (i+1, j, L, d)] \leftarrow \lambda$ 
    end for
  else
     $i \leftarrow n - s, j \leftarrow 0, L \leftarrow s$ 
    for  $i \leftarrow N - S$  to  $K - S - 1$  do
       $Q[(i, j, L, d), (i+1, j, L, d)] \leftarrow \lambda$ 
       $Q[(i, j, L, d), (i-1, j, L, d)] \leftarrow \mu$ 
    end for
     $i \leftarrow K - S$ 
     $Q[(i, j, L, d), (i-1, j, L, d)] \leftarrow \mu$ 
     $i \leftarrow N - S - 1$ 
    for  $i \leftarrow N - S - 1$  to  $1$  do
       $Q[(i, j, L, d), (i+1, j, L, d)] \leftarrow \lambda$ 
       $Q[(i, j, L, d), (i-1, j, L, d)] \leftarrow \mu$ 
    end for
     $i \leftarrow 0, L \leftarrow s$ 
     $Q[(i, j, L, d), (i+1, j, L, d)] \leftarrow \lambda$ 
     $Q[(i, j, L, d), (i-1, j, L-1, d)] \leftarrow \mu$ 
    For  $L \leftarrow S - 1$  to  $2$  do
       $Q[(i, j, L, d), (i, j, L-1, d)] \leftarrow \mu$ 
       $Q[(i, j, L, d), (i, j, L+1, d)] \leftarrow \lambda$ 
    end for
     $Q[(i, j, L, d), (i, j, L+1, d)] \leftarrow \lambda$ 
     $Q[(i, j, L-1, d), (i, j, L-2, d+S)] \leftarrow \mu$ 
     $L \leftarrow 0$ 
     $d \leftarrow S$ 

     $Q[(i, j, L, d), (i, j, L+1, d-S)] \leftarrow \lambda$ 
     $Q[(i, j, L, d), (i, j+S, L, d)] \leftarrow \beta$ 
  endif
end for
```

Modeling of the system

3.5 Performance measures

The states of the CTMC are all reachable and communicating states and none of the states is an absorbing state, as well as the states, are all recurrent and aperiodic then this CTMC is an ergodic Markov chain.

Based on the ergodicity of this CTMC, we can have the solution of this CTMC at steady-state by computing the stationary probability vector $\pi = (\pi_1, \pi_2, \pi_3, \dots)$ which is the solution of the linear system of equations:

$$\begin{cases} \pi Q = 0 \\ \sum_{i \in E} \pi_i = 1 \end{cases} \dots (3-8)$$

- π_i : steady state probability that the process in state i
- Q : the infinitesimal generator correspondent to the CTMC
- E : the states of the CTMC

Having a steady state vector, we can now calculate the various performance measures as follow:

- **The blocking probability of tasks (P_B):** It corresponds to the buffer saturation probability.

$$P_B = \sum_{i: T_i(\text{buffer})=k-s} \pi_i \quad \dots (3-9)$$

Where:

π Represent the steady vector, and i is the number of states

$T_i(X)$ Represent the number of tasks in the state i in place X

Modeling of the system

- **The Probability that a VM is on busy state P_s :** It corresponds to the probability that the place Busy contains at least one token.

$$P_s = \sum_{i:Vi(busy) \neq 0} \pi_i \quad \dots (3-10)$$

$Vi(X)$: represent the number of servers in state i in place X

- **The probability that the VM is on sleep state P_{sl} :** It corresponds to the probability that the place sleep contains at least one token.

$$P_{sl} = \sum_{i:Vi(sleep) \neq 0} \pi_i \quad \dots (3-11)$$

- **The probability that the VM is on sleep-delay state P_{sl-d}**

$$P_{sl-d} = \sum_{i:Vi(sleep-delay) \neq 0} \pi_i \quad \dots (3-12)$$

- **The mean number of tasks in the VM (\bar{Q}):** It represents the mean number of waiting tasks in the VM, including the tasks being transmitted. This corresponds to the mean number of tokens in the place Buffer

$$\bar{Q} = \sum_{i:Ti \in A} Ti(buffer) \cdot \pi_i \quad \dots (3-13)$$

Where A represents the set of reachable states in the system.

- **The task reception throughput $\bar{\lambda}$:** It corresponds to the effective rate of tasks reception by the VM.

$$\bar{\lambda} = \lambda \cdot \sum_{i:Ti \in E(T_arr)} \pi_i \quad \dots (3-14)$$

Modeling of the system

Where $E(T_Arr)$ represents the set of tasks where the transition T_Arr is enabled.

- **The Task service throughput during busy state ($\bar{\mu}$):**

$$\bar{\mu} = S \cdot \mu \cdot \sum_{i:Vi \in E(T_serv)} \pi_i \quad \dots (3-15)$$

Where:

$E(T_Serv)$ represents the set of servers where the transition T_Serv is enabled. And S is the number of servers

- **The average length of a sleep period \bar{S} :** It corresponds to the average duration of time of the system during the sleep period

$$\bar{S} = \frac{N}{\lambda} \quad \dots (3-16)$$

- **The average length of a busy period \bar{B} :** It corresponds to the average duration of service time

$$\bar{B} = \frac{\bar{Q}}{S \cdot \mu} \quad \dots (3-17)$$

- **The mean sojourn time of tasks in the buffer W :**

Modeling of the system

$$\bar{W} = \frac{\bar{Q}}{\lambda} \quad \dots (3-18)$$

- **The Average duration of a cycle \bar{C} :** from the previous equations we have

$$\bar{C} = \bar{B} + \bar{S} \quad \dots (3-19)$$

- **The number of cycles (N_c):** It corresponds to the number of transitions from sleep to busy per time unit.

$$N_c = \frac{1}{\bar{C}} \quad \dots (3-20)$$

- **The energy consumption physical machine PM (EC):**

$$EC1 = EC_S \cdot P_S + EC_b \cdot (1 - P_S) + EC_{SW} \cdot N_c + EC_{sr} \cdot \bar{Q} \quad \dots \text{model 1}$$

$$EC2 = EC_S \cdot P_S + EC_b \cdot (1 - P_S - P_{sl-d}) + EC_{SW} \cdot N_c + EC_{sr} \cdot \bar{Q} + P_{sl-d} \cdot EC_{sl-d} \quad \dots \text{Model 2}$$

Where

- EC_S : the energy consumption while the PM is in sleep.
- EC_b : the energy consumption while the PM is in busy.
- EC_{sl-d} : the energy consumption while the PM is in sleep delay
- EC_{SW} : the energy consumption when the PM switch from sleep to busy.
- EC_{sr} : the energy consumption for handling 1 task

3.6 Conclusion

In this chapter we talked about related works to our work and then modeled two systems the first one with threshold policy and working vacation and the second one with working vacation, threshold policy and sleep-delay timer using the Knowledge that we previously acquired in chapter 1 and 2.

We represent the systems using the CTMC graphs, and then we make a reduction for simplifying the resolution, in the resolution, we construct the infinitesimal generator and declare the sets of linear equations for calculating the stationary distribution. That allows us to calculate the various system performance.

In the next chapter we are going to set values for the parameters of the two systems and observe the changeset and compare the two models in specific performance and even compare the same system with different values of parameters.

Chapter 4 Experimental

Experimental

4.1 Introduction

After the modelling of the system with 2 different models approaches, we arrive now to implementation of the proposed solutions to test them according to different scenarios, retrieval of results for comparison between the two models, analysis and discussion.

In this chapter we will also see the tools and methods used to retrieve and these results.

4.2 Experimental studies

In this section, first, we are going to test model 1 and 2 separately in energy, mean waiting time in the buffer and blocking probability with different parameter values, then compare between them.

To measure these performance values, here are the steps and methods we followed:

- In our program implementation, we map each a state to a number (state number) to easily determine the states and find with the one we deal with.
- We define the Infinitesimal generator algorithm with show all the possible transition and save the Infinitesimal matrix Q.
- Based on the Infinitesimal matrix Q, we solve the linear system of equations:

$$\begin{cases} \pi Q = 0 \\ \sum_{i \in E} \pi_i = 1 \end{cases} \dots (4-1)$$

- As we said in the previous chapter using MATLAB to find the steady state of our CTMC model
- After that, we calculate the performance measures of our system using the steady state.

Experimental

4.3 Development tools

In our work, we used c-sharp as a programming language and visual studio as an IDE for c-sharp, with those we also used MATLAB for the Equation solutions and the operations on matrices.

4.3.1 C-sharp

C# is an object-oriented programming language, strongly typed, derived from C and C++, resembling the Java language. It is used to develop web applications, as well as desktop applications, web services, commands, widgets or class libraries. In C#, an application is a set of classes where one of the classes has a Main method, as it is done in Java.

C# is intended to develop on the .NET platform, a technology stack created by Microsoft to succeed COM.

C# Executables in C# are subdivided into assemblies, namespaces, classes and class4 members. An assembly is the compiled form, which can be a program (an executable) or a class library. An assembly contains the executable code and symbols. The code is translated into machine language at runtime by the just-in-time function of the .NET platform.(27)

4.3.2 Visual studio

Designed by Microsoft. The latest version is called Visual Studio 2019. Visual Studio is a complete set of development tools for generating ASP.NET web applications, XML web services, desktop and mobile applications. Visual Basic, Visual C++, Visual C# all use the same integrated development environment (IDE), which allows them to share tools and facilitates the creation of solutions using several languages. In addition, these languages enable them to better leverage the functionality of the .NET framework, which provides access to key technologies that simplify the development of ASP web applications and XML web services through Visual Web Developer.(28)

Experimental

4.3.3 Matlab

Matlab or matrix laboratory is a scripting language emulated by a development environment of the same name; it is used for numerical computation. Developed by The MathWorks Company, Matlab allows users to manipulate matrices, display curves and data, implement algorithms, create user interfaces, and can interface with other languages such as C, C++, and Java. Matlab users come from a wide variety of backgrounds, including engineering, science, and economics in both industrial and research settings. (29)

System parameters

parameter	value
Capacity of buffer (K)	40
Number of servers	10
Queue threshold	Range from 1 to K-1
Arrival rate	From 0.25 to 0.75
Service rate	2
Sleep delay rate	From 0.001 to 0.5
EC_S	10
EC_B	300
EC_{SL-D}	30
EC_{SR}	5
EC_{SW}	100

Note: We assume that energy is calculated with millijoule (mJ) and the time with millisecond (mS).

Experimental

4.4 Model 1 (Cloud model with n-policy)

- **Energy consumption:**

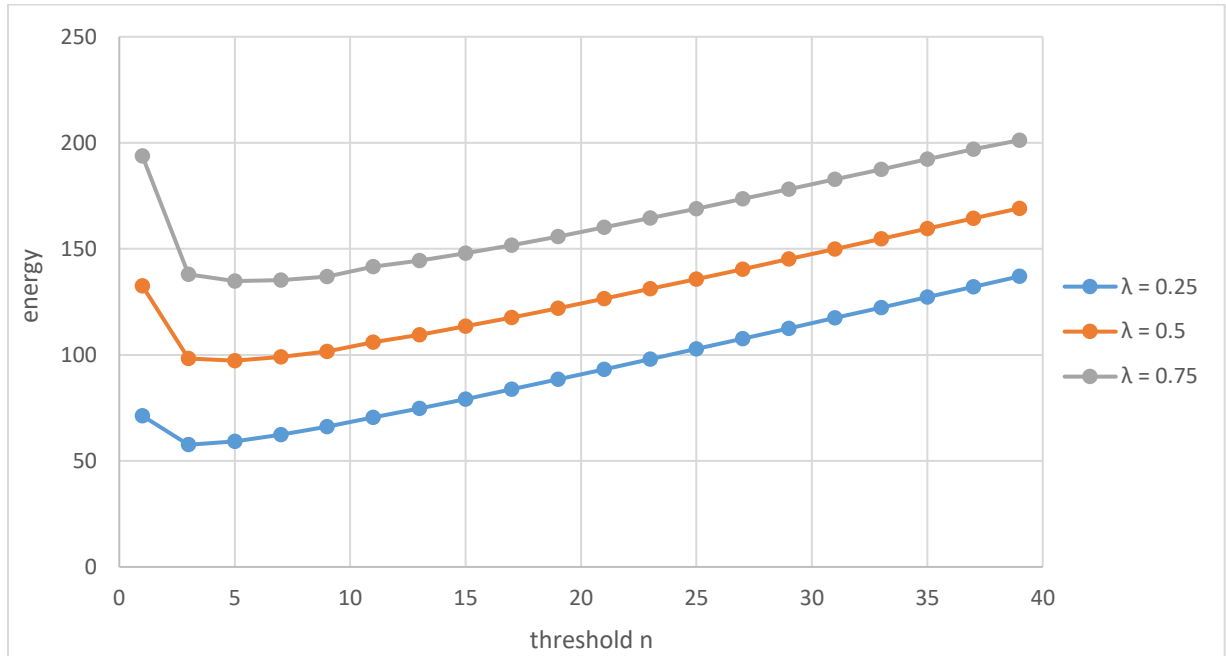


Figure 4-1energy consumption in model 1

Those charts represent the energy consumption by the threshold N while varying the arrival rate λ between (0.25, 0.5, 0.75) while the threshold N is between (1-40).

From the above chart, we test the average energy consumption with a different value of threshold n and arrival rate λ . We see that when $\lambda = 0.25$ the system consumes the minimum energy, while $\lambda = 0.75$ the system consumes the highest energy meaning the average energy consumption increases with the increase of λ .

For the threshold n, we see that when n range from 0 to 3 the average energy consumption decreases (n = 3 is the lowest value) but from 3 to 40, the average energy consumption increases with the increase of n. Because when λ and n increase the busy probability increase, thus the energy increase

Experimental

- Mean waiting time:

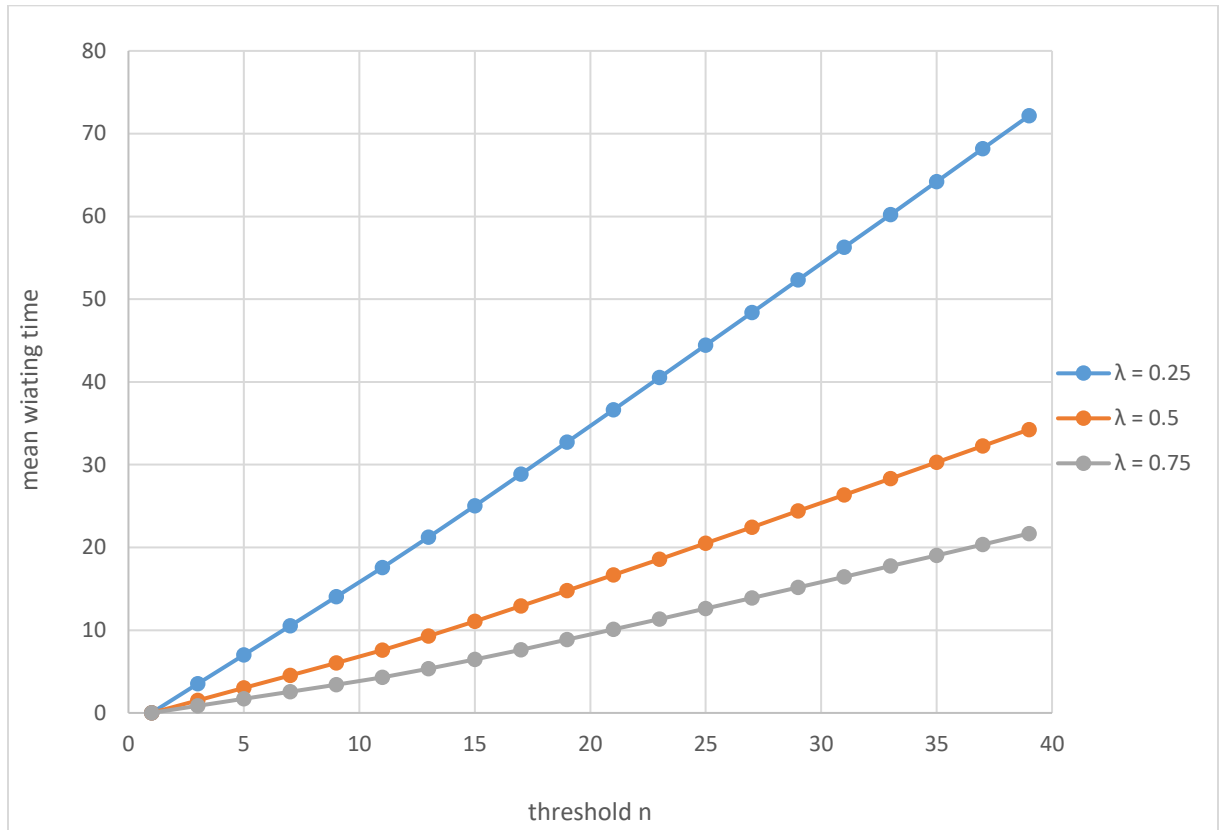


Figure 4-2: mean waiting time in model 1

In the same way, we test the mean waiting time in the buffer, We notice in those charts that when the threshold n is 1 the waiting time is 0 so there is no waiting because in every arrival the server's wakeup and this for any value of the given λ values, we see that when λ increase the mean waiting time decrease, from the chart $\lambda = 0.75$ registers the minimum waiting time. That is because when $\lambda = 0.75$ the interval between λ and μ decrease.

For the threshold n, in each increase of n the mean waiting time increase.

Experimental

- **Blocking probability:**

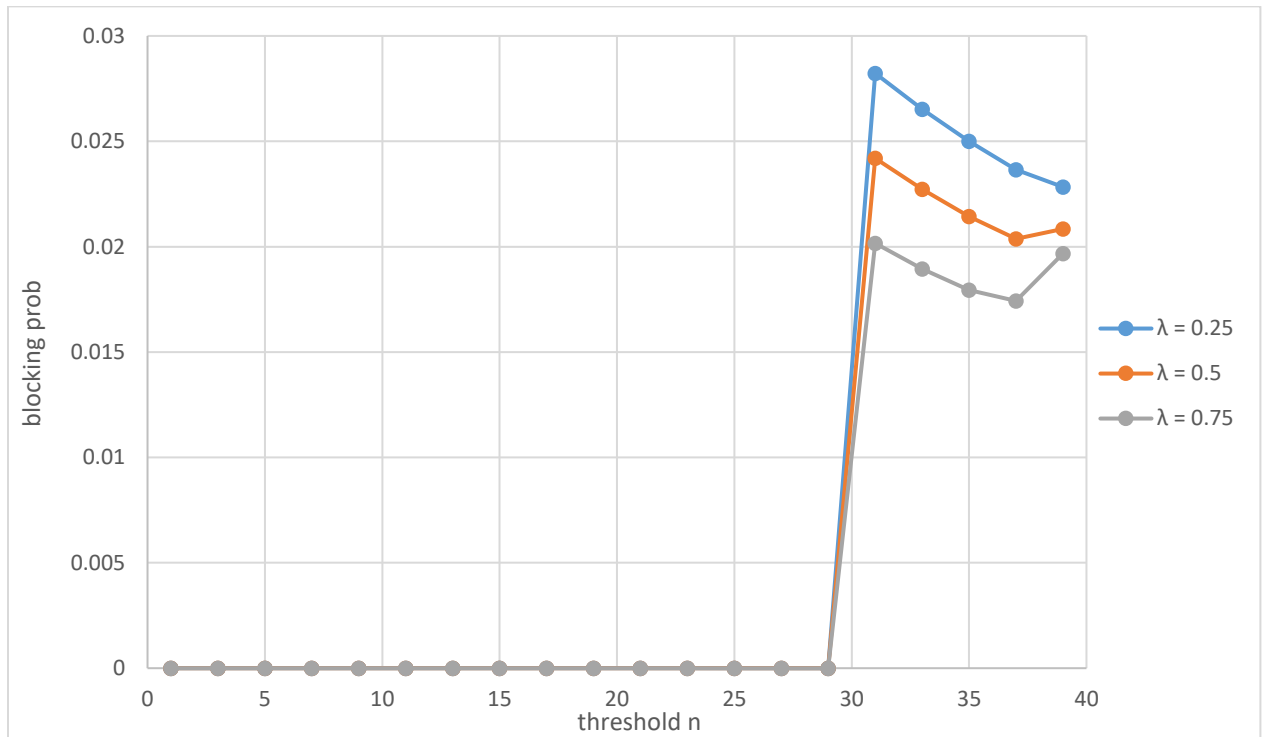


Figure 4-3: blocking probability in model 1

These charts represent the blocking probability by the threshold N while always varying λ (0.25, 0.5, 0.75) and the threshold N between 1 and 40.

We see in that the blocking probability is 0 from $N= 1$ to N equals 30 and this is because the buffer is going to be saturated if $N= 30$

when λ increase the probability decrease and when the threshold n increases the blocking probability increase.

4.5 Model 2 (Cloud model with n-policy and sleep delay state)

- **Energy consumption:**

In these tests, the values of parameters of the system are the same, and beta is equal to 10^{-3}

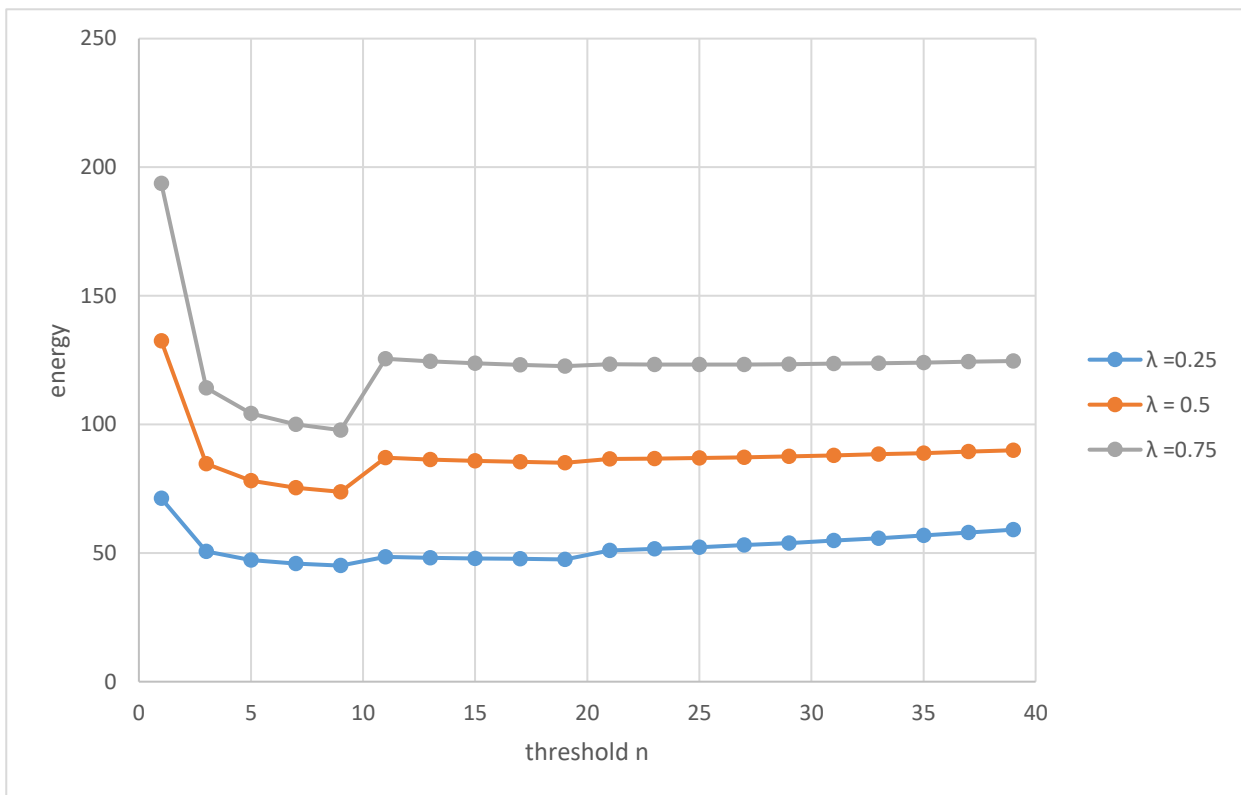


Figure 4-4: energy consumption in model 2

Those charts represent the energy consumption by the threshold N while varying the arrival rate λ between (0.25, 0.5, 0.75) while the threshold N is between (1-40)

We notice in this charts that the energy is in its highest level when the threshold is 1 and that means that whenever a task came the servers all going to be awake so there is so much energy wasted. After that and with the rise of the threshold N we notice that the

Experimental

energy goes down, and when the threshold reaches 10 which is the number of servers, we observe that the charts are stable and goes slightly up, with the increasing of the threshold.

By varying λ we see the differences between the three charts, so we note that with the increasing of λ , the system consumes more energy.

- **Mean waiting time:**

The same in these tests the values of parameters of the system are the same and beta is equal to 10^{-3}

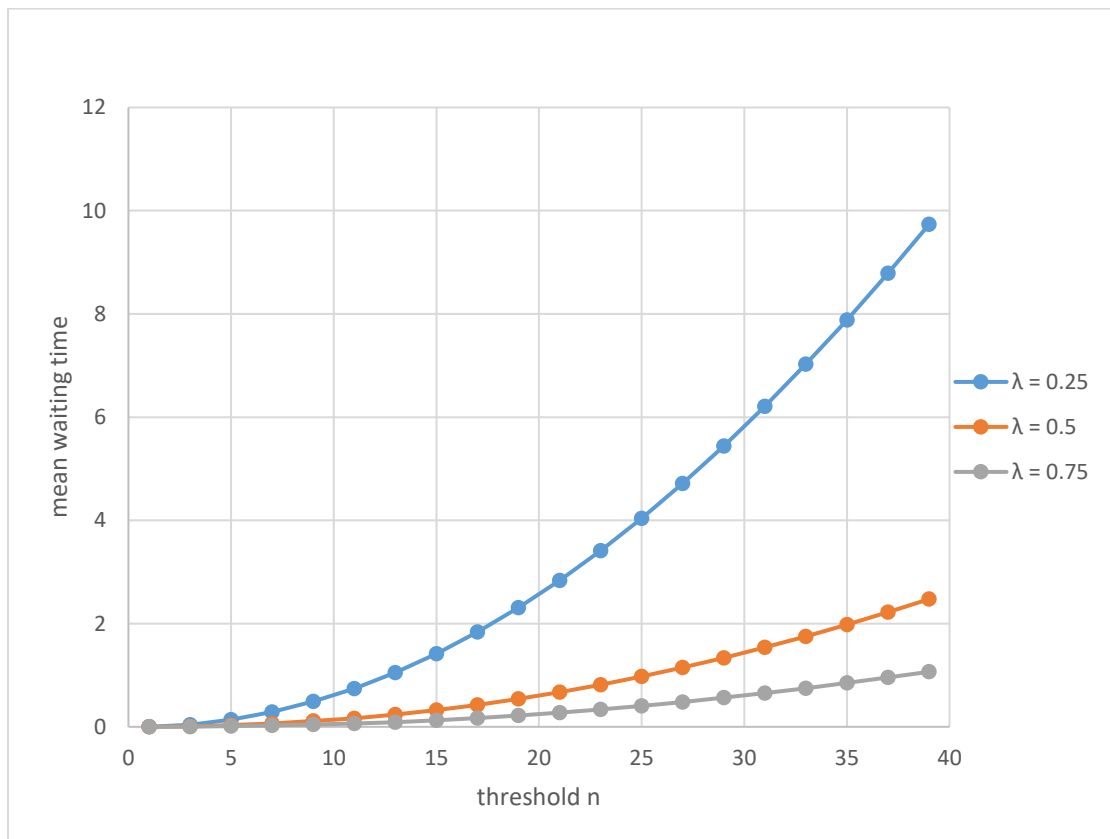


Figure 4-5: mean waiting time in model 2

Experimental

Those three charts represent the mean waiting time of the task in the buffer by the threshold N every chart represent mean waiting time with the change of λ . At the same time, N is between 1 and 40.

We notice in those charts that when the threshold is 1 the waiting time is 0 so there is no waiting because in every arrival the server's wakeup and this for any value of the given λ values.

With the increase of the threshold, we note that there is increase also of the waiting time in the buffer and the differences between the charts begin to expand we also notice that the less waiting time is in the chart with $\lambda=0.75$ and $\lambda=0.25$ gets the biggest value of the mean waiting time.

- **Blocking probability:**

We still the same, in these tests the values of parameters of the system are the same and beta is equal to 10^{-3}

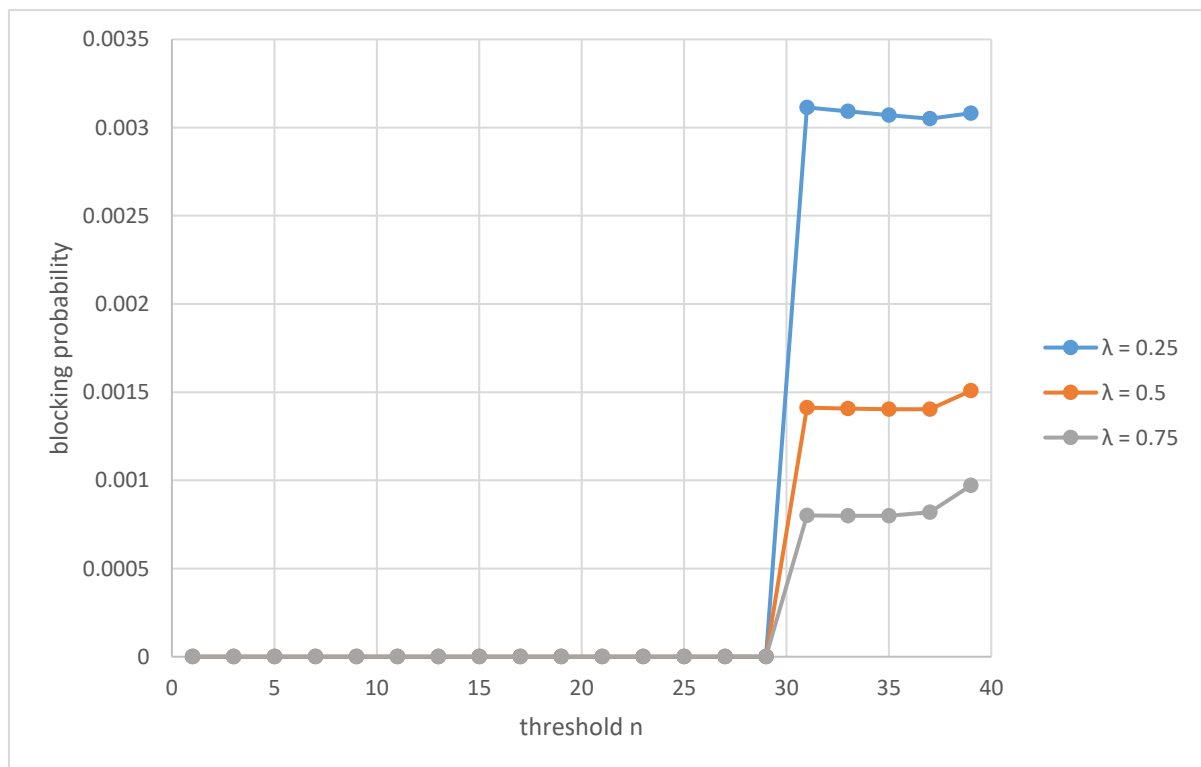


Figure 4-6: blocking probability in model 2

Experimental

These charts represent the blocking probability by the threshold N while always varying λ (0.25, 0.5, 0.75) and the threshold N between 1 and 40.

We see in these charts that the probability that the system is in blocking state is 0 from $N=1$ to N equals 30, and this is because that the buffer is going to be saturated if $N=30$ then the system is not working until 30 tasks arrive which increase the probability the system is in blocking state, expend we also notice that the less probability of being in blocking state is in the chart with $\lambda=0.75$ and $\lambda=0.25$ gets the biggest value of that probability.

4.6 Model 1 and model 2 comparison

Now we going to compare these to model side by side to determine with one is more useful in the same pervious performance measure.

- **Energy consumption:**

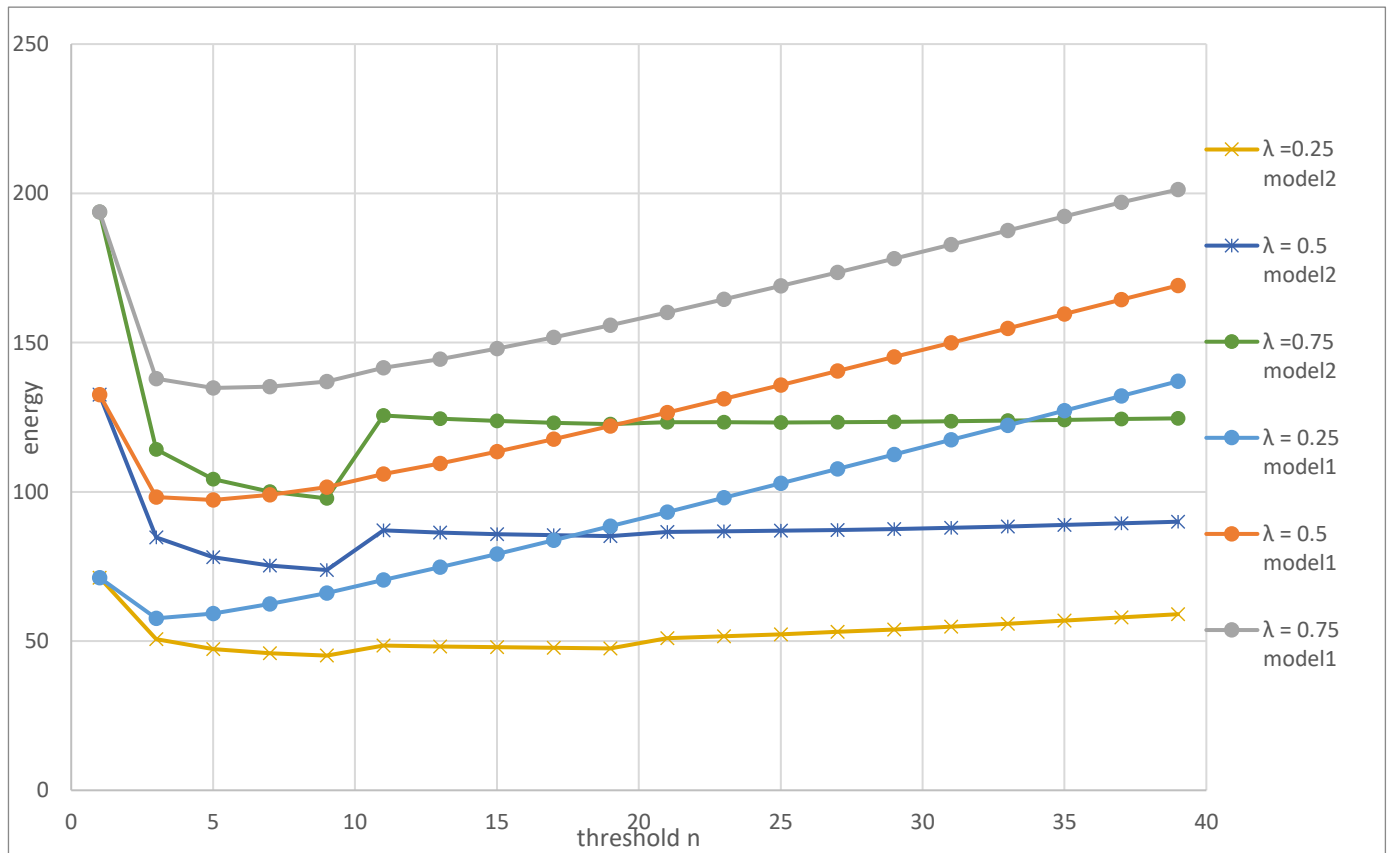


Figure 4-7: energy consumption in model 1 and 2

Experimental

Those charts represent the energy consumption of model1 (Cloud model with n-policy) and model2 (Cloud model with n-policy and sleep delay state) side by side by the threshold N while varying the arrival rate λ between (0.25, 0.5, 0.75) while the threshold N is between (1-40)

We start by comparing the mean energy consumption between the two models. By comparing each graph from model1 (Cloud model with n-policy) and model2 (Cloud model with n-policy and sleep delay state) with the same λ , we see clearly that model2 (Cloud model with n-policy and sleep delay state) consume less energy

- **Mean waiting time:**

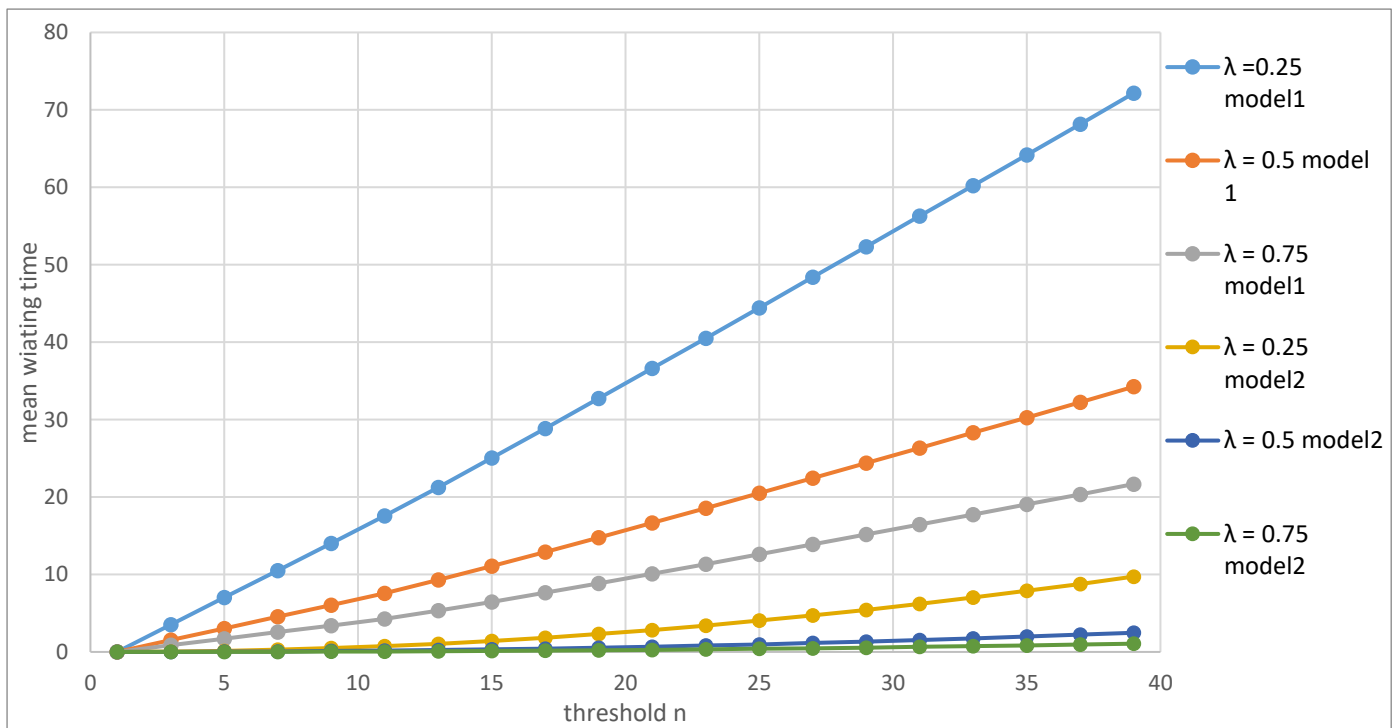


Figure 4-8: mean waiting time in model 1 and 2

In the same way with the same value of n and λ , we compare the mean waiting time in the buffer between model1 (Cloud model with n-policy) and model2 (Cloud model with n-

Experimental

policy and sleep delay state). We can see that all three graphs from model2 (with $\lambda=0.25, 0.5, 0.75$) are spending less time waiting in buffer compare to the rest graph from model1 (Cloud model with n-policy)

- **Blocking probability:**

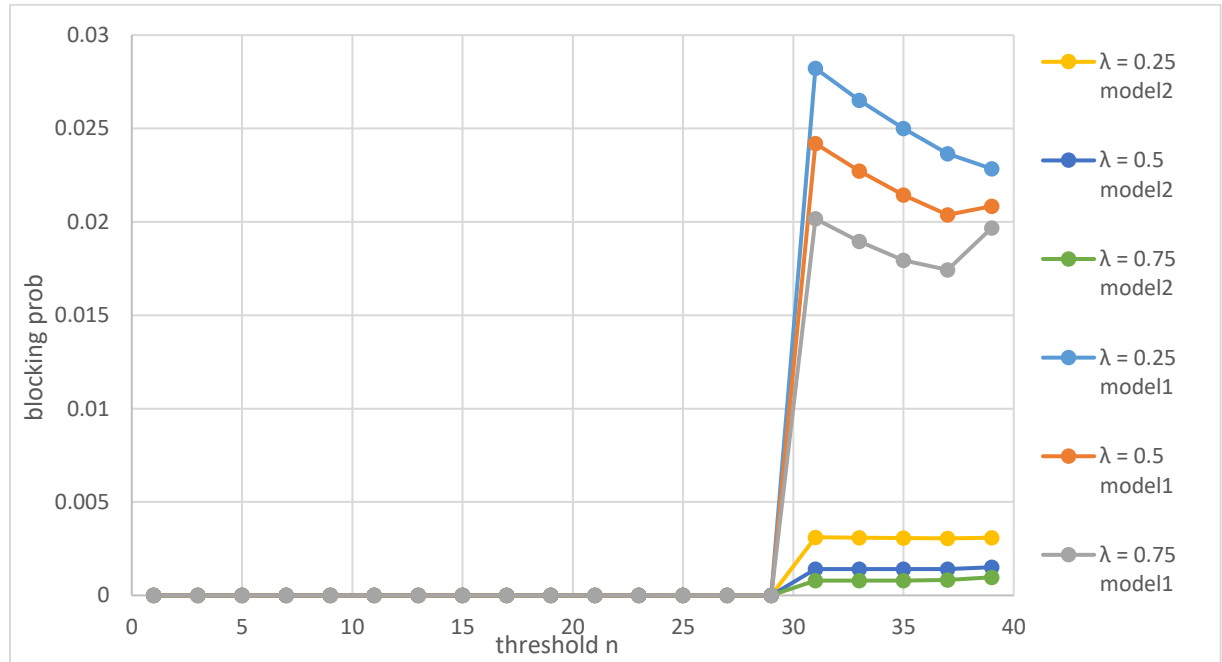


Figure 4-9: blocking probability in model 1 and 2

Those charts represent the blocking probability of model1 (Cloud model with n-policy) and model2 side by side by the threshold N while varying the arrival rate λ between (0.25, 0.5, 0.75) while the threshold N is between (1-40)

The same case as means waiting time happen in blocking probability, all three graphs from model2 have less blocking probability compare to model1 (Cloud model with n-policy).

Model 2 with different sleep delay (β)

Now that we know the differences between model 1 (Cloud model with n-policy) and model 2, we are going to make tests on model 2 (Cloud model with n-policy and sleep

Experimental

delay state) with the variation of beta with (0.001, 0.01, 0.5) with every of energy consumption, mean waiting time and the blocking probability.

4.7 Model2 with different sleep delay parameter value

- **Energy consumption:**

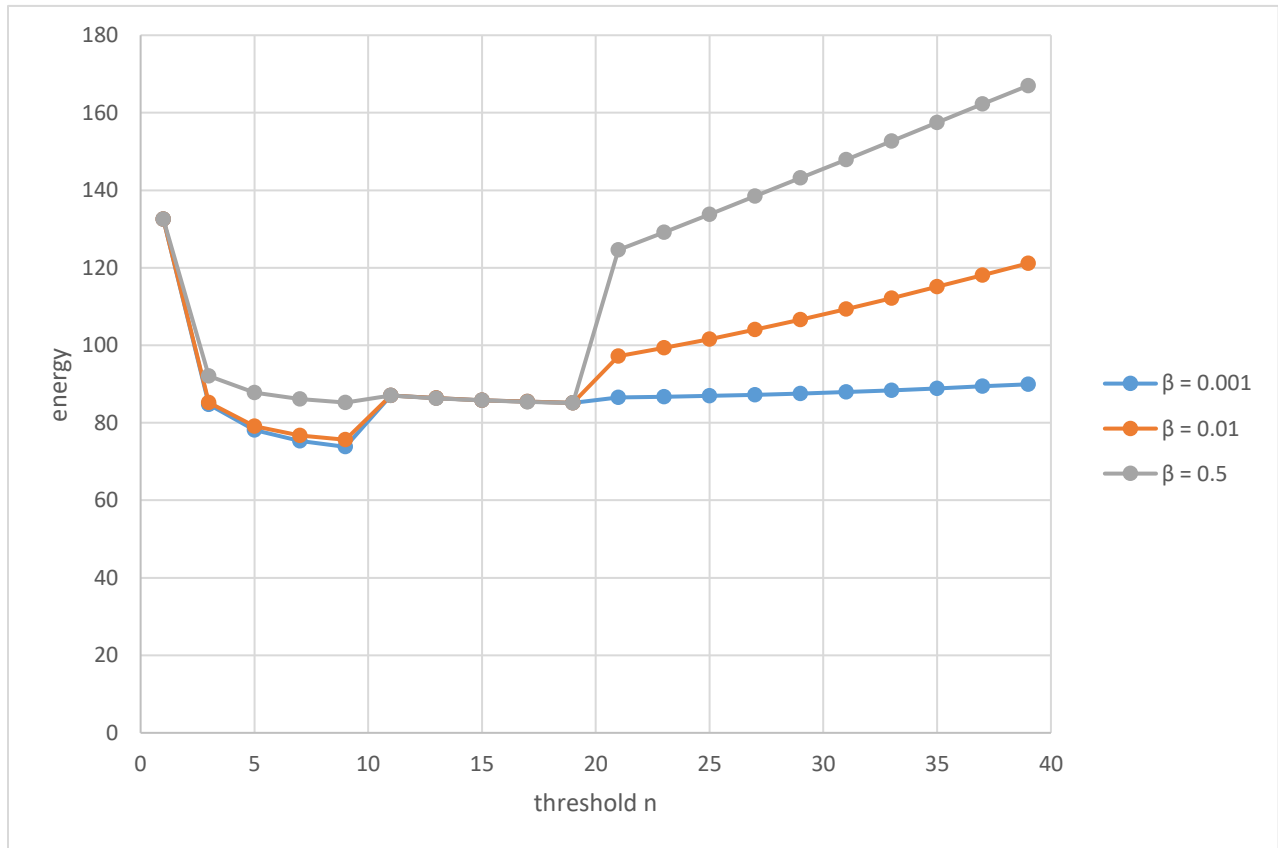


Figure 4-10: energy consumption in model 2 varying sleep delay

These charts represent the energy consumption by the threshold N while varying β and every chart represent one value of β and the threshold N between 1 and 40.

We notice in these charts that the energy is the same until N reaches 20. At that value, we see the difference between the charts where 0.001 is the best energy consumption, and 0.5 is the worst energy consumption from the given values.

Experimental

- Mean waiting time:

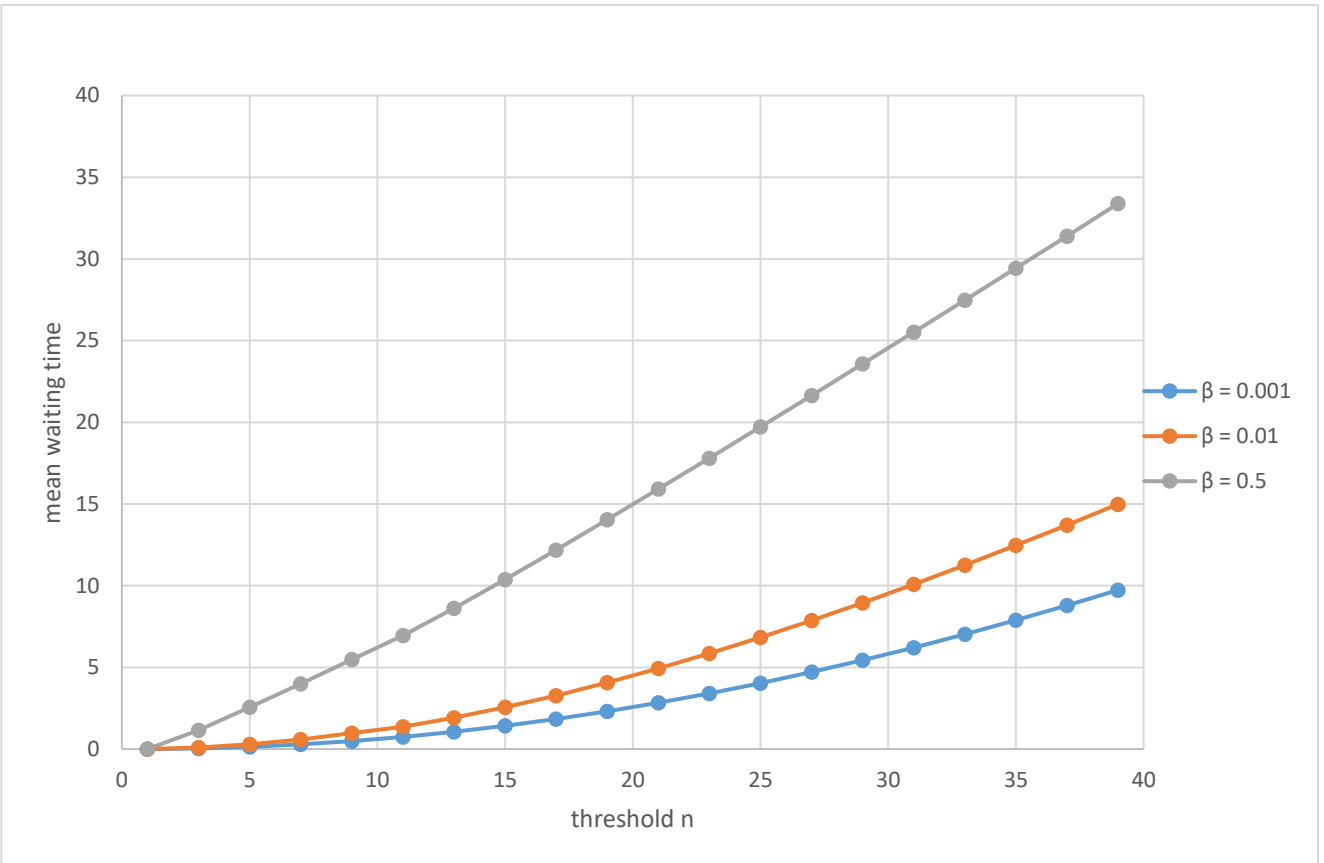


Figure 4-11: mean waiting time in model 2 varying sleep delay

These charts represent the mean waiting time in the buffer by the threshold N while varying β and every chart represent one value of β and the threshold N between 1 and 40.

We notice in those charts that when the threshold is 1, the waiting time is 0 so there is no waiting.

With the increase of the threshold, we note that there is increase also of the waiting time in the buffer and the differences between the charts begin to expand we also notice that the less waiting time is in the chart with $\beta = 0.001$ and $\beta = 0.5$ gets the biggest value of the mean waiting time.

Experimental

- **Blocking probability:**

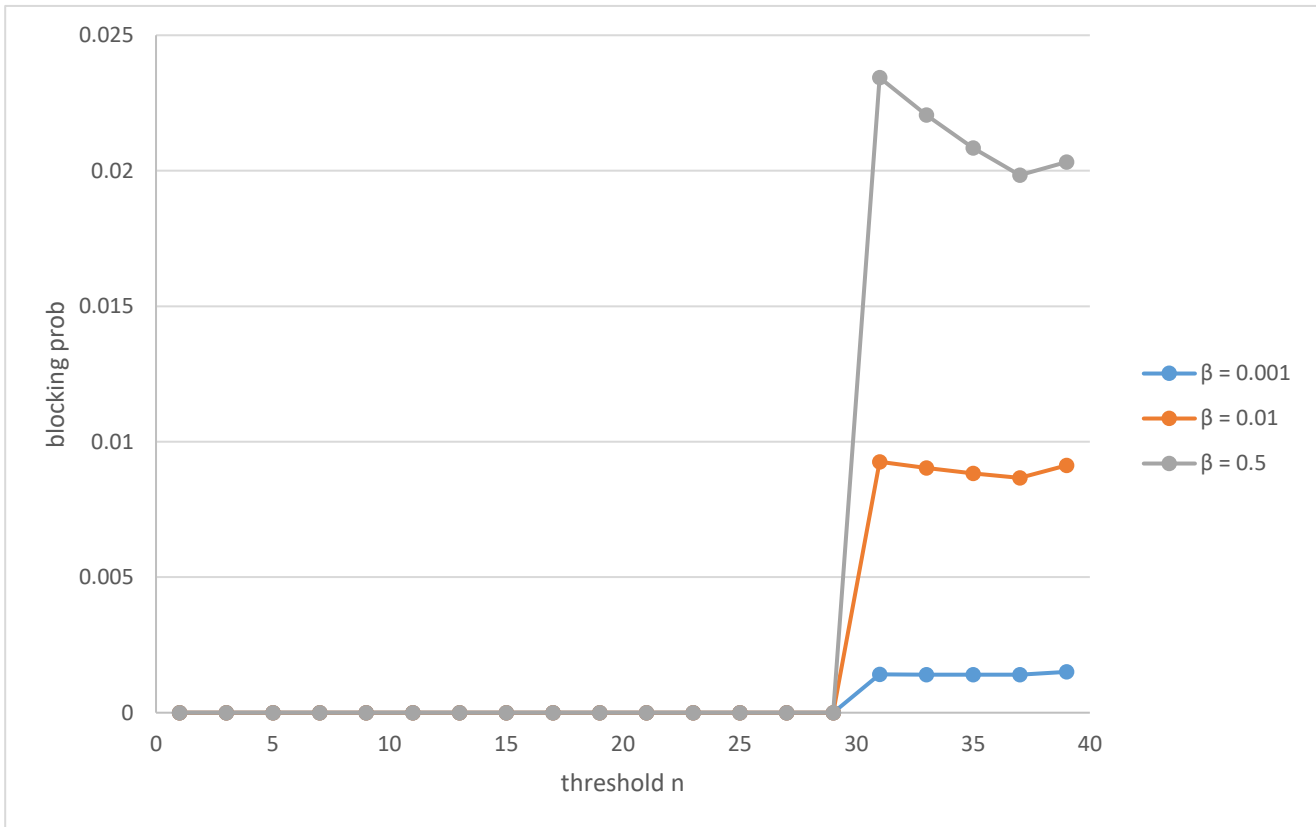


Figure 4-12: blocking probability in model 2 varying sleep delay

These charts represent the blocking probability by the threshold N while always varying β and the threshold N between 1 and 40.

We see in these charts that the probability that the system is in blocking state is 0 from $N= 1$ to N equals 30, and this is because that the buffer is going to be saturated if $N= 30$ then the system is not working until 30 tasks arrive which increase the probability the system is in blocking state, expend we also notice that the less probability of being in blocking state is the usage of $\beta= 0,001$ and when we rise β the probability of blocking increase and that according to the other 2 charts ($\beta=0.01$ and $\beta=0.5$).

4.8 Conclusion

At the end of this chapter and after we measure the performance analysis of each model which are energy consumption and mean waiting time of the tasks in the buffer and the blocking probability.

We see that the model 1 and model 2 has different results which we observe after comparing the two models that the model 2 is better in those performances analysis and after that, we did compare the same model 2 with itself while changing the sleep delay parameter β , we find that whenever β is smaller in our tests, we get better latency and lower energy consumption.

In the end, we conclude that model 2 is better than model 1, and we did discover the better range values of the parameters to use to get the best results.

Conclusion

Cloud computing is becoming a hotline topic in the information technology section, and it is scaling at an intense rate. This scalability demands more resources than ever. Also, one of the critical factors of using more resources is energy. At the same time, it is a major problem in cloud computing.

Dominating cloud providers like amazon web service and google cloud are consuming a significant amount of energy to provides their services. Thus, cloud customers pay a more amount of money to get their work done with less period of time on the cloud.

To have a better experience for providers and customers on the cloud, we introduce an energy-efficient task scheduling strategy in a cloud computing system using a woking up threshold and adding sleep delay parameter.

We firstly model our solution. After that, we compare it to other related work through different test scenarios and find that our proposed solution gives lower energy and better latency with a specific set of parameters.

At the same time, we tested the blocking state of our solution, and we fund that this solution does not affect the availability of the system, but it gives a minimum improvement

This work opens perspectives for reducing energy consumption and latency in cloud data centers.

Bibliography

1. History of Cloud Computing - javatpoint [Internet]. [cited 2020 Dec 3]. Available from: <https://www.javatpoint.com/history-of-cloud-computing>
2. Fehling C, Leymann F, Retter R, Schupeck W, Arbitter P, Fehling C, et al. Cloud Offering Patterns. In: Cloud Computing Patterns. Springer Vienna; 2014. p. 79–150.
3. PLATFORM AS A SERVICE [Internet]. [cited 2020 Dec 3]. Available from: <https://sites.google.com/site/platformasaservice3/>
4. Cloud Deployment Model - an overview | ScienceDirect Topics [Internet]. [cited 2020 Dec 3]. Available from: <https://www.sciencedirect.com/topics/computer-science/cloud-deployment-model>
5. What is a Virtual Machine (VM)? - Definition from Techopedia [Internet]. [cited 2020 Dec 3]. Available from: <https://www.techopedia.com/definition/4805/virtual-machine-vm>
6. Hypervisor | Cloud Computing Patterns [Internet]. [cited 2020 Dec 3]. Available from: <https://www.cloudcomputingpatterns.org/hypervisor/>
7. (PDF) Energy efficient utilization of resources in Cloud computing systems [Internet]. [cited 2020 Dec 3]. Available from: https://www.researchgate.net/publication/225140382_Energy_efficient_utilization_of_resources_in_Cloud_computing_systems
8. Mathai AM, Haubold HJ. 9. Collection of random variables. In: Probability and Statistics. 2017.
9. Bas E, Bas E. An Introduction to Markov Chains. In: Basics of Probability and Stochastic Processes. 2019.
10. Cassandras CG, Lafortune S. Introduction to discrete event systems. Introduction to Discrete Event Systems. 2008.
11. Nicolas P. Understanding Markov Chains: Examples and Applications. Numerical Methods for Partial Differential Equations. 2004.
12. Grinstead CM, Snell JL. Introduction to Probability: Second Revised Edition. Am Math Soc. 1997;
13. Sarig O. Lecture Notes on Ergodic Theory. Preprint [Internet]. 2008;2019:1–115. Available from:

Bibliography

- <http://www.math.psu.edu/sarig/506/ErgodicNotes.pdf%5Cnpapers2://publication/uuid/65795CDF-EEB0-4AF8-83CD-59F8BE88915E>
14. William J. Stewart. Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling. Princeton, NJ: Princeton University Press; 2009. 717 p.
 15. 08_Queueing_Models.pdf | Mathematical Model | Markov Chain [Internet]. [cited 2020 Dec 3]. Available from: <https://www.scribd.com/document/441495851/08-Queueing-Models-pdf>
 16. Queueing | Markov Chain | Poisson Distribution [Internet]. [cited 2020 Dec 3]. Available from: <https://www.scribd.com/presentation/260926259/Queueing>
 17. Ke JC, Lin CH, Huang HI, Zhang ZG. An algorithmic analysis of multi-server vacation model with service interruptions. *Comput Ind Eng*. 2011 Nov 1;61(4):1302–8.
 18. - MedCrave online [Internet]. [cited 2020 Dec 3]. Available from: <https://medcraveonline.com/OAJMTP/OAJMTP-01-00036>
 19. Probability, Markov chains, queues, and simulation. The mathematical basis of performance modeling - PDF Free Download [Internet]. [cited 2020 Dec 3]. Available from: <https://epdf.pub/probability-markov-chains-queues-and-simulation-the-mathematical-basis-of-perfor.html>
 20. Network Modeling | Stochastic Process | Markov Chain [Internet]. [cited 2020 Dec 3]. Available from: <https://www.scribd.com/document/71801211/Network-Modeling>
 21. Zhao W, Wang X, Jin S, Yue W, Takahashi Y. An energy efficient task scheduling strategy in a cloud computing system and its performance evaluation using a two-dimensional continuous time markov chain model. *Electron*. 2019;8(7).
 22. Feller E, Ramakrishnan L, Morin C. Performance and energy efficiency of big data applications in cloud environments: A Hadoop case study. *J Parallel Distrib Comput*. 2015;
 23. Xia YN, Zhou MC, Luo X, Pang SC, Zhu QS. A stochastic approach to analysis of energy-aware DVS-enabled cloud datacenters. *IEEE Trans Syst Man, Cybern Syst*. 2015;
 24. Chen YL, Chang MF, Liang WY, Lee CH. Performance and energy efficient dynamic voltage and frequency scaling scheme for multicore embedded system. In: 2016 IEEE International Conference on Consumer Electronics, ICCE 2016. 2016.
 25. LawanyaShri M, Balusamy B, Subha S. Threshold-based workload control for an under-utilized virtual machine in cloud computing. *Int J Intell Eng Syst*. 2016;9(4).
 26. Zhao W, Wang X, Jin S, Yue W, Takahashi Y. An energy efficient task scheduling

Bibliography

- strategy in a cloud computing system and its performance evaluation using a two-dimensional continuous time markov chain model. *Electron.* 2019;8(7).
27. Tutorialspoint. *C# Programing tutorial*. Tutorialspoint. 2014;
 28. Visual Studio Code. *Documentation for Visual Studio Code*. Visual Studio Code Documentation. 2015.
 29. Keviczky L, Bars R, Hetthéssy J, Bányász C. *Introduction to MATLAB*. *Advanced Textbooks in Control and Signal Processing*. 2019.
 30. Jackson, C., Jackson, C. and Wasko, S., 2017. *CCNA Cloud CLDADM 210-455 Official Cert Guide*. Indianapolis, IN: Cisco Press.