People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of Abbas Laghrour Khenchela

Master of End Study

Department Of Mathematics and Computer Science

Specialty: Security and Web Technology

Theme:

Tower a New Method to Unbalancing Load Detection in Cloud Computing

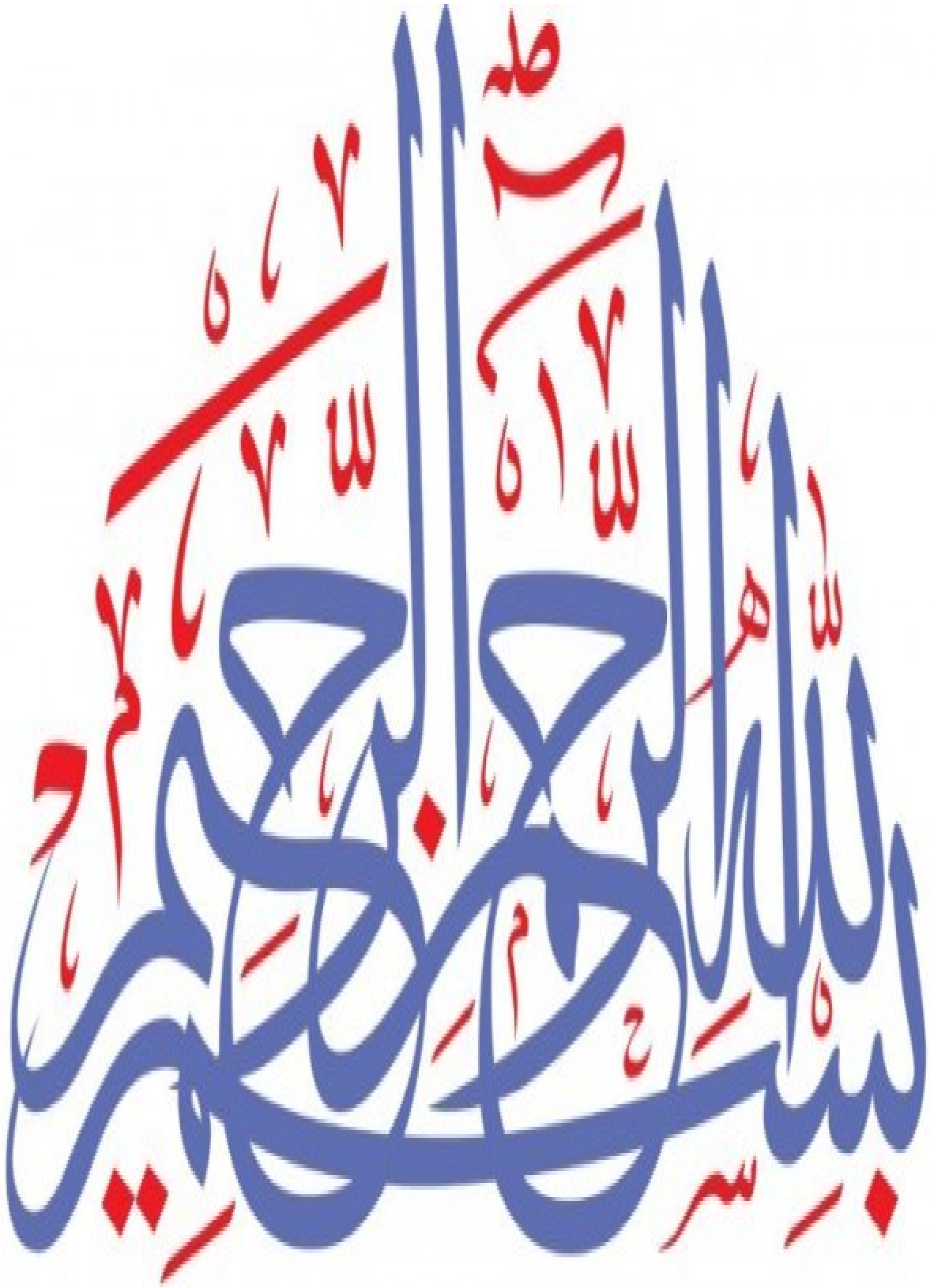
Supervised by :

Dr.Hemam Sofiane Mounine

Presented by :

Falek Mohamed

Mahfoud Soulaiman


بسم الله الرحمن الرحيم

Thanks

Our thanks go to all those who, from near and far, have helped make this work a reality.

We thank God for allowing us to complete this brief, our parents for their honest and infallible sacrifices.

We would particularly like to thank our mentor, Hemam Sofiane, for the support provided and express our gratitude and gratitude to him for having supervised this work and for the trust he has given us, his advice and encouragement.

We also thank the president of the jury and its members for agreeing to chair and take part in the evaluation of this work.

May all those we have not named find here the expression of our deep gratitude and our eternal salvation.

Dedications

Dedicate this thesis:

To the owner of a fragrant biography and an enlightened thought:

He had the first credit in attaining higher education (my beloved father), may God prolong his life

To those who put me on the path of life, made me calm, and took care of me until I became old (my dear mother), may God bless her soul.

To my brothers; Those who have had a great impact on many obstacles and difficulties. To all my esteemed teachers:

My university journey has come to an end after fatigue and hardship...I'm concluding my graduation research with all vigor and activity.

And I am grateful to everyone who has had a favor in my career, and who helped me, even if just a little.

Parents, family, friends, and respected professors, we especially thank the Supervising Professor Hemam Sofiane.

ملخص

موازنة الحمل في مركز البيانات هي محاولة القيام بموازنة حمل العمل او موازنة الحمل بين العقد (اي الجهاز الحقيقي او الافتراضي) التي عليها ضغط والعقد الاخرى التي لاتواجه ضغط في النظام الذي قام بالتوزيع مسبقا وذلك لتحسين استخدام الموارد ووقت الاستجابة للعملية التي يريدها العميل والتي ندعوها بالوظيفة وكل هذا مع وجوب منع المواقف التي يتم فيها تحميل بعض العقد بشكل كبير بينما تكون العقد الاخرى خامدة او تقوم بعمل قليل مقارنة بنظيراتها تفترض خوارزمية موازنة الحمل الديناميكية عدم وجود معرفة مسبقة بسلوك الوظيفة أو الحالة الكلية للنظام ، أي أن قرارات موازنة الحمل تعتمد فقط على الحالة الحالية للنظام. يتضمن تطوير خوارزمية فعالة لموازنة الحمل الديناميكي العديد من القضايا المهمة منها: تقدير الحمل ، ومقارنة مستويات الحمل ، ومؤشرات الأداء ، واستقرار النظام ، وكمية المعلومات المتبادلة بين العقد ، وتقدير متطلبات موارد العمل ، واختيار الوظيفة للنقل ، واختيار العقد البعيدة ، والمزيد . هدفنا هو اكتشاف عبء العمل على الأجهزة الافتراضية في الخوادم قبل حدوثها ومحاولة تقليل الضغط عليها

الكلمات المفتاحية: أنظمة الكمبيوتر الموزعة. حوسبة سحابية؛ توزيع الحمل؛ تقاسم الحمولة؛ تقييم الأداء؛ المزيد؛ مقارنة المستويات

Abstract

Load balancing is the process of redistributing the work load among nodes of the distributed system to improve both resource utilization and job response time while also avoiding a situation where some nodes are heavily loaded while others are idle or doing little work. A dynamic load balancing algorithm assumes no a priori knowledge about job behavior or the global state of the system, i.e., load balancing decisions are solely based on the current status of the system. The development of an effective dynamic load balancing algorithm involves many important issues: load estimation, load levels comparison, performance indices, system stability, amount of information exchanged among nodes, job resource requirements estimation, job's selection for transfer, remote nodes selection, and more. Our objective is to detect the workload on virtual machines in servers before happen and try to reduce the pressure on them.

Keywords: Distributed computer systems; Cloud computing; load balancing; load sharing; performance evaluation; stability; levels comparison.

Résumé

L'équilibrage de charge est le processus de redistribution de la charge de travail entre les nœuds du système distribué pour améliorer à la fois l'utilisation des ressources et le temps de réponse des travaux, tout en évitant une situation où certains nœuds sont fortement chargés tandis que d'autres sont inactifs ou effectuent peu de travail. Un algorithme d'équilibrage de charge dynamique ne suppose aucune connaissance a priori du comportement du travail ou de l'état global du système, c'est-à-dire que les décisions d'équilibrage de charge sont uniquement basées sur l'état actuel du système. Le développement d'un algorithme efficace d'équilibrage de charge dynamique implique de nombreux problèmes importants: estimation de la charge, comparaison des niveaux de charge, indices de performance, stabilité du système, quantité d'informations échangées entre les nœuds, estimation des besoins en ressources du travail, sélection du travail pour le transfert, sélection des nœuds distants, etc. .notre objectif est de détecter la charge de travail sur les machines virtuelles dans les serveurs avant de se produire et d'essayer de réduire la pression sur eux.

Mots clés: Systèmes informatiques distribués; Cloud computing; l'équilibrage de charge; partage de charge; évaluation des performances; stabilité; comparaison des niveaux.

Table Of Contents

Liste of Figure

Abbreviations

PM: Physical Machine

VMs: Virtual Machines

NIST: National Institute Standards and Technology

LB: Load Balancing

General Introduction

A cloud computing model is efficient if its resources are utilized in best possible way and such an efficient utilization can be achieved by employing and maintaining proper management of cloud resources. Resource management is achieved by adopting robust resource scheduling, allocation and powerful resource scalability techniques. These resources are provided to customers in the form of Virtual Machines (VM) through a process known as virtualization that makes use of an entity (software, hardware or both) known as hypervisor. The greatest advantage of cloud computing is that a single-user physical machine is transformed into a multi-user virtual machines. The major companies play a crucial role in service delivery to users and it is a complex task with given available virtual resources. While serving user requests, some VMs will get a heavy traffic of user tasks and some will get a lesser traffic. As a result, the Cloud service Company is left with unbalanced machines which have a huge gradient of user tasks and resource utilization. The problem of load unbalancing is an undesirable event in the companies' side, which degrades the performance and efficacy of the computing resources. Under these circumstances there arises need for load balancing (LB) and is a peculiar topic of research interest among researchers. The load balancing in cloud computing can be done at physical machine level or VM level .A task utilize resources of a VM and when a bunch of tasks arrive at a VM, the resources gets exhausted which means no resource is now available to handle the additional task requests. When such situation arises the VM is said to have entered into an overloaded state. At this point of time, tasks will either suffer from starvation or end up in deadlock with no hope of accomplishing them. Consequently there is necessity to migrate tasks to another resource on other VM. The workload migration process includes three basic steps: load balancing which checks the current load on machine resource, resource discovery which finds another suitable resource and workload migration which moves extra tasks to available resources. These operations are performed by three different units commonly known as load balancer, resource discovery and task migration units respectively. Load balancing is the process of redistribution of workload in a distributed system like cloud computing ensuring no computing machine is overloaded, under-loaded or idle. Load balancing tries to speed up different constrained parameters like response time, execution time, system stability etc. thereby improving performance of cloud. It is an optimization technique in which task scheduling is a hard problem. There are a large

number of load balancing approaches proposed by researchers where most of focus has been concerned on task scheduling, task allocation, resource scheduling, resource allocation, and resource management. To the best of our knowledge, we could not find an in-depth and comprehensive literature concerned with factors that cause load unbalancing situation. The survey papers based on load balancing could not provide a proper systematic classification of methods and techniques.in this thesis we will provide our model which is in line with the general concept, and will try improve the delay time and reduce the workload on the virtual machines to improve the performance and to optimize the available recourses utilization.

this thesis is arranged as follows : In the first chapter we will provide an overview of distributed computing systems and cloud computing.in second chapter will get to know the load balancing concept and explain them and its great importance in any distribute systems, in the third chapter we will show our model and explain its levels, and detailing how he react with the system, in the fourth and last chapter we will get to know the work environment and show how to implement our model and we present some obtained results.

Distributed System and Cloud Computing

Chapter Plan

1. *Introduction*
2. *Definition of a Distributed Computing*
3. *Architectures*
4. *Goals and challenges*
5. *Deference between Parallel and Distributed Computing*
6. *Definition of Cloud Computing*
7. *Characteristics of Cloud Computing*
8. *Service Models*
9. *Virtualization*
10. *Deployment models*
11. **Conclusion**

1. Introduction

Computing has passed through many transformations since the birth of the first computing machines. Developments in technology have resulted in the availability of fast and inexpensive processors, and progresses in communication technology have resulted in the availability of lucrative and highly proficient computer networks. Among these, the centralized networks have one component that is shared by users all the time. All resources are accessible, but there is a single point of control as well as a single point of failure. The integration of computer and networking technologies gave birth to new paradigm of computing called distributed computing in the late 1970s. Distributed computing has changed the face of computing and offered quick and precise solutions for a variety of complex problems for different fields. Nowadays, we are fully engrossed by the information age, and expending more time communicating and gathering information through the Internet. In this chapter we will provide an overview of distributed computing systems. The definition, architecture, goals and challenges. Finally, discusses about the difference between Parallel and distributed computing.

2. Definition of a Distributed Computing

Distributed computing is a field of computer science that studies distributed systems. A distributed system is a system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another [1]. The components interact with one another in order to achieve a common goal. Three significant characteristics of distributed systems are: concurrency of components, lack of a global clock, and independent failure of components [1]. Examples of distributed systems vary from SOA-based systems to massively multiplayer online games to peer-to-peer applications.

Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers [2], which communicate with each other via message passing [3].

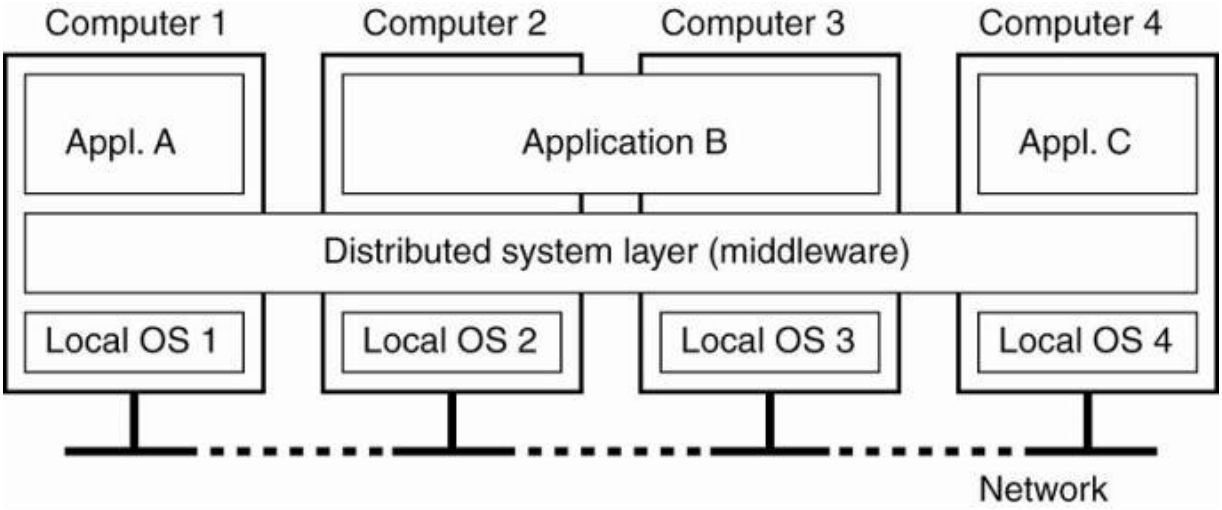


Figure 1 : Distributed System

3. Architectures

Various hardware and software architectures are used for distributed computing. At a lower level, it is necessary to interconnect multiple CPUs with some sort of network, regardless of whether that network is printed onto a circuit board or made up of loosely coupled devices and cables. At a higher level, it is necessary to interconnect processes running on those CPUs with some sort of communication system [4].

Distributed programming typically falls into one of several basic architectures: client–server, three-tier, *n*-tier, or peer-to-peer; or categories: loose coupling, or tight coupling [5].

Client–Server: architectures where smart clients contact the server for data then format and display it to the users. Input at the client is committed back to the server when it represents a permanent change.

Three-tier: architectures that move the client intelligence to middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are three-tier.

***N-tier*:** architectures that refer typically to web applications which further forward their requests to other enterprise services. This type of application is the one most responsible for the success of application servers.

Peer-to-peer: architectures where there are no special machines that provide a service or manage the network resources [6]. Instead all responsibilities are uniformly divided among all machines, known as peers. Peers can serve both as clients and as servers [7]. Examples of this architecture include BitTorrent and the bitcoin network.

Another basic aspect of distributed computing architecture is the method of communicating and coordinating work among concurrent processes. Through various message passing protocols, processes may communicate directly with one another, typically in a master/slave relationship. Alternatively, a "database-centric" architecture can enable distributed computing to be done without any form of direct inter-process communication, by utilizing a shared database [8]. Database-centric architecture in particular provides relational processing analytics in a schematic architecture allowing for live environment relay. This enables distributed computing functions both within and beyond the parameters of a networked database [9].

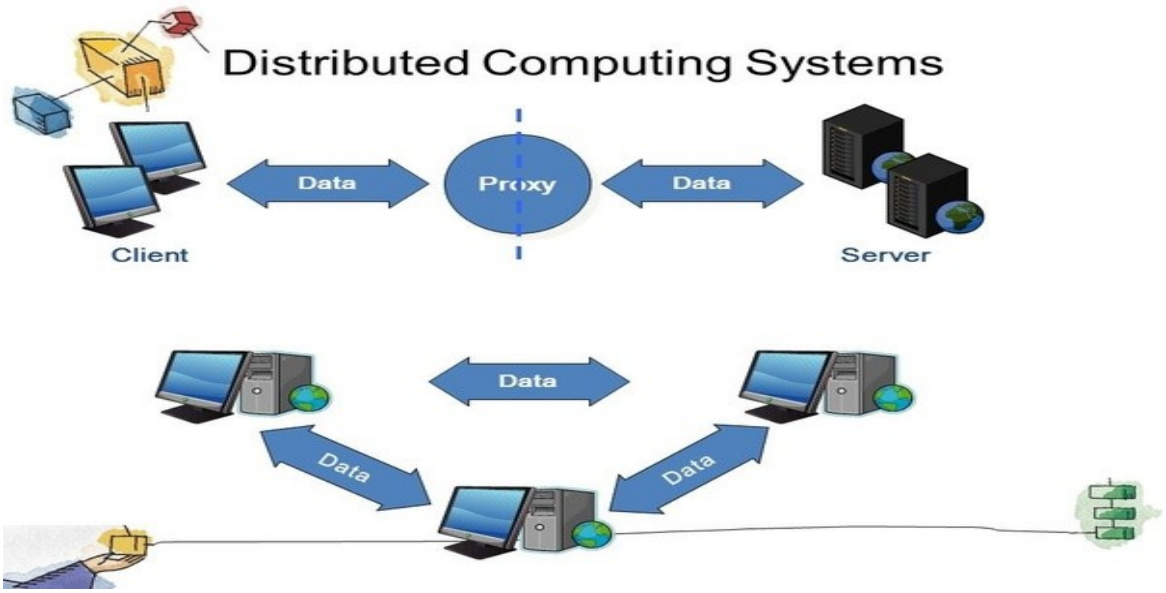


Figure 2 : Distributed computing Systems

4. Goals and challenges

4.1. Goals :

Connecting Users and Resources:

- The main goal of a distributed system is to make it easy for users to access remote recourses and to share them with others in a controlled way.
- It is cheaper to le a printer be shared by several users than buying and maintaining printers for each user.
- Collaborating and exchanging information can be made easier by connecting users and resource.

Transparence :

- It is important for a distributed system to hide the location of its process and resource. A distributed system that can portray itself as a single system is said to be transparent.
- The various transparencies need to be considered are access, location, migration, relocation, replication, concurrency, failure and persistence.
- Aiming for distributed transparency should be considered along with performance issues.

Openness :

- Openness is an important goal of distributed system in which it offers services according to standard rules that describe the syntax and semantics of those services.
- Open distributed system must be flexible making it easy to configure and add new components without affecting existing components.
- An open distributed system must also be extensible.

Scalable :

- Scalability is one of the most important goals which are measured along three different dimensions.
- First, a system can be scalable with respect to its size which can add more user and resources to a system.
- Second, users and resources can be geographically apart.
- Third, it is possible to manage even if many administrative organizations are spanned.

4.2 Challenges

Securities:

Many of the information resources that are made available and maintained in distributed systems have a high intrinsic value to their users. Their security is therefore of considerable importance. Security for information resources has three components:

confidentiality (protection against disclosure to unauthorized individuals)

integrity (protection against alteration or corruption),

availability for the authorized (protection against interference with the means to access the resources).

Fault-Tolerant:

software modules that are bound to fail in the long run. Such component failures can escort to service unavailability. Hence, the systems should be able to recover from component failures without performing erroneous actions. The goal of fault tolerance is to avoid failures in the system even in the presence of faults to provide uninterrupted service.

Heterogeneity:

The Internet enables users to access services and run applications over a heterogeneous collection of computers and networks. Heterogeneity (that is, variety and difference) applies to all of the following:

Hardware devices: computers, tablets, mobile phones, embedded devices, etc.
Operating System: Ms Windows, Linux, Mac, Unix, etc.
Network: Local network, the Internet, wireless network, satellite links, etc.
Programming languages: Java, C/C++, Python, PHP, etc.
Different roles of software developers: designers, system managers

Different programming languages use different representations for characters and data structures such as arrays and records. These differences must be addressed if programs written in different languages are to be able to communicate with one another. Programs written by different developers cannot communicate with one another unless they use common standards, for example, for network communication and the representation of primitive data

items and data structures in messages. For this to happen, standards need to be agreed and adopted – as have the Internet protocols.

5 Difference between Parallel And Distributed Computing

Distributed systems are groups of networked computers which share a common goal for their work. The terms "concurrent computing", "parallel computing", and "distributed computing" have much overlap, and no clear distinction exists between them [10]. The same system may be characterized both as "parallel" and "distributed"; the processors in a typical distributed system run concurrently in parallel [11]. Parallel computing may be seen as a particular tightly coupled form of distributed computing [12], and distributed computing may be seen as a loosely coupled form of parallel computing [13]. Nevertheless, it is possible to roughly classify concurrent systems as "parallel" or "distributed" using the following criteria :

- In parallel computing, all processors may have access to a shared memory to exchange information between processors[14].
- In distributed computing, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors.

The figures 3 and 4 illustrate the difference between distributed and parallel systems. Figure (a) is a schematic view of a typical distributed system; the system is represented as a network topology in which each node is a computer and each line connecting the nodes is a communication link. Figure (b) shows the same distributed system in more detail: each computer has its own local memory, and information can be exchanged only by passing messages from one node to another by using the available communication links. The Figure 4 shows a parallel system in which each processor has a direct access to a shared memory.

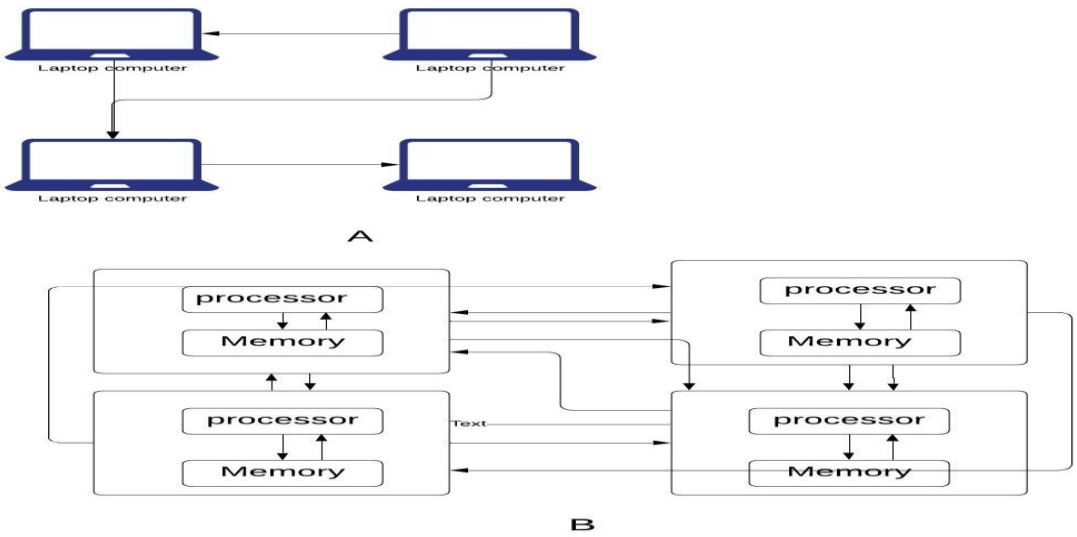


Figure 3 : A Distributed System

The situation is further complicated by the traditional uses of the terms parallel and distributed *algorithm* that do not quite match the above definitions of parallel and distributed *systems* (see below for more detailed discussion). Nevertheless, as a rule of thumb, high-performance parallel computation in a shared-memory multiprocessor uses parallel algorithms while the coordination of a large-scale distributed system uses distributed algorithms [15].

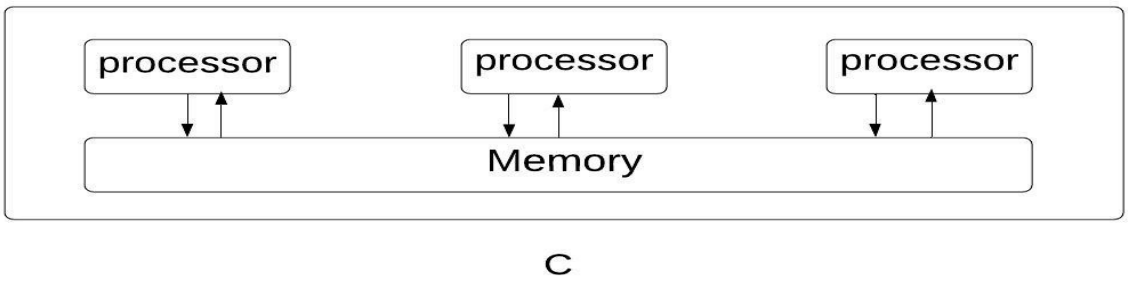


Figure 4 : a parallel system

6 Definition Of Cloud Computing

Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet [16]. Large clouds, predominant today, often have functions distributed over multiple locations from central servers. If the connection to the user is relatively close, it may be designated an edge server.

Clouds may be limited to a single organization (enterprise clouds [17] [18]), or be available to many organizations (public cloud).

Cloud computing relies on sharing of resources to achieve coherence and economies of scale.

7 Characteristics Of Cloud Computing

As cloud computing services mature both commercially and technologically, it will be easier for companies to maximize the potential benefits. Knowing what cloud computing is and what it does, however, is just as important. The National Institute of Standards and Technology (NIST) defines cloud computing as it is known today through five particular characteristics.

7.1 On-demand self-service

Cloud computing resources can be provisioned without human interaction from the service provider. In other words, a manufacturing organization can provision additional computing resources as needed without going through the cloud service provider. This can be a storage space, virtual machine instances, database instances, and so on. Manufacturing organizations can use a web self-service portal as an interface to access their cloud accounts to see their cloud services, their usage, and also to provision and de-provision services as they need to.

7.2 Broad network access :

Cloud computing resources are available over the network and can be accessed by diverse customer platforms. It other words, cloud services are available over a network—ideally high broadband communication link—such as the internet, or in the case of a private clouds it could be a local area network (LAN). Network bandwidth and latency are very important aspects of cloud computing and broad network access, because they relate to the quality of

service (QoS) on the network. This is particularly important for serving time sensitive manufacturing applications.

7.3 Multi-tenancy and resource pooling :

Cloud computing resources are designed to support a multi-tenant model. Multi-tenancy allows multiple customers to share the same applications or the same physical infrastructure while retaining privacy and security over their information. It's similar to people living in an apartment building, sharing the same building infrastructure but they still have their own apartments and privacy within that infrastructure. That is how cloud multi-tenancy works. Resource pooling means that multiple customers are serviced from the same physical resources. Providers' resource pool should be very large and flexible enough to service multiple client requirements and to provide for economy of scale. When it comes to resource pooling, resource allocation must not impact performances of critical manufacturing applications.

7.4 Rapid elasticity and scalability :

One of the great things about cloud computing is the ability to quickly provision resources in the cloud as manufacturing organizations, and then to remove them when they don't need them. Cloud computing resources can scale up or down rapidly and, in some cases, automatically, in response to business demands. It is a key feature of cloud computing. The usage, capacity, and therefore cost, can be scaled up or down with no additional contract or penalties. Elasticity is a landmark of cloud computing and it implies that manufacturing organizations can rapidly provision and de-provision any of the cloud computing resources. Rapid provisioning and de-provisioning might apply to storage or virtual machines or customer applications. With cloud computing scalability, there is less capital expenditure on the cloud customer side. This is because as the cloud customer needs additional computing resources, they can simply provision them as needed, and they are available right away. Scalability is more planned and gradual. For instance, scalability means that manufacturing organizations are gradually planning for more capacity and of course the cloud can handle that scaling up or scaling down. Just-in-time (JIT) service is the notion of requiring cloud elasticity either to provision more resources in the cloud or less. For example, if all of a sudden of a manufacturing organization needs more computing power to perform some kind of complex calculation; this would be cloud elasticity that would be a just-in-time service. On

the other hand, if the manufacturing organization needs to provision human-machine interface (HMI) tags in the database for a manufacturing project, that is not really just-in-time service, it is planned ahead of time. So it is more on the scalability side than elasticity. Another feature available for rapid elasticity and scalability in the cloud is related to testing of manufacturing applications. If a manufacturing organization needs, for example, a few virtual machines to test a supervisory control and data acquisition (SCADA) system before they roll it out in production, they can have it up and running in minutes instead of physically ordering and waiting for hardware to be shipped. In terms of the bottom line, when manufacturing organizations need to test something in the cloud, they are paying for what they use as they use it. As long as they remember to de-provision it, they will no longer be paying for it. There is no capital expense here for computer resources. Manufacturing organizations are using the cloud provider's investment in cloud computing resources instead. This is really useful for testing smart manufacturing solutions.

7.5 Measured service :

Cloud computing resources usage is metered and manufacturing organizations pay accordingly for what they have used. Resource utilization can be optimized by leveraging charge-per-use capabilities. This means that cloud resource usage—whether virtual server instances that are running or storage in the cloud—gets monitored, measured and reported by the cloud service provider. The cost model is based on "pay for what you use"—the payment is variable based on the actual consumption by the manufacturing organization.

8 Service Models

Though service-oriented architecture advocates "Everything as a service" (with the acronyms EaaS or XaaS [19], or simply aas), cloud-computing providers offer their "services" according to different models, of which the three standard models per NIST are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [20]. These models offer increasing abstraction; they are thus often portrayed as layers in a stack: infrastructure, platform and software-as-a-service, but these need not be related. For example, one can provide SaaS implemented on physical machines (bare metal), without using underlying PaaS or IaaS layers, and conversely one can run a program on IaaS and access it directly, without wrapping it as SaaS.

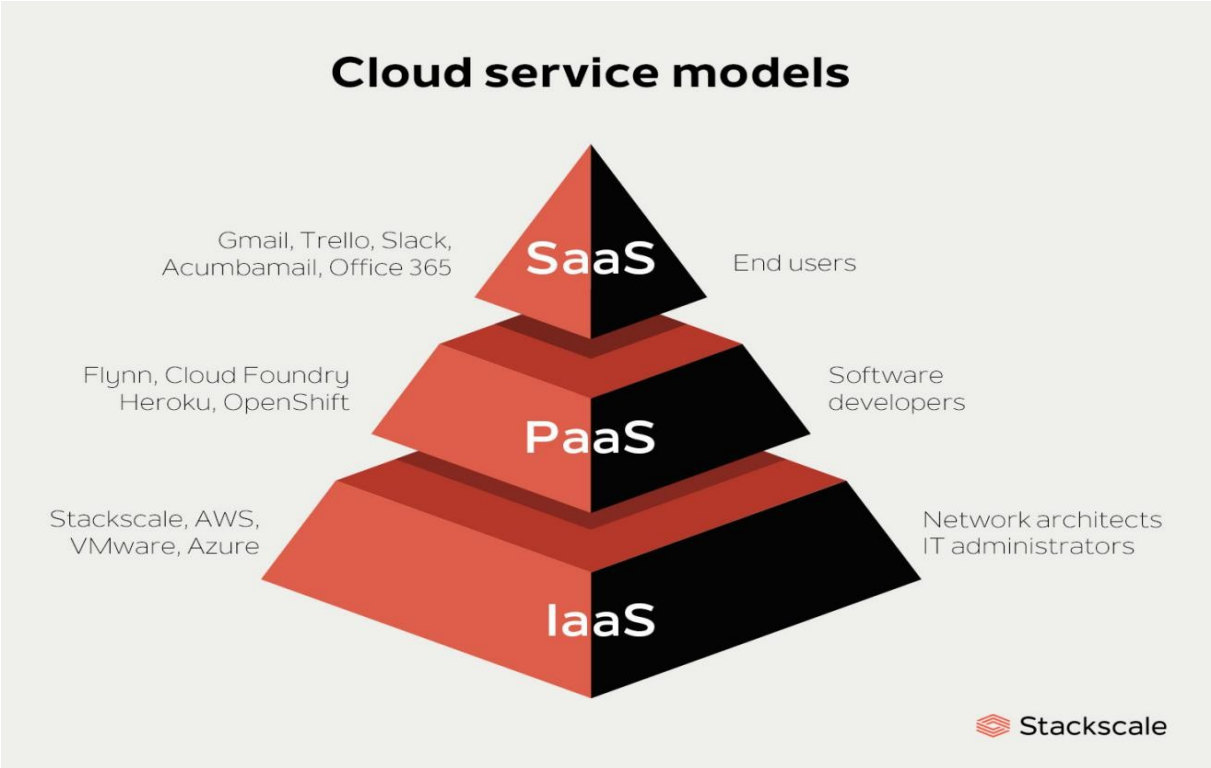


Figure 5 : Cloud Service Modles

8.1 Infrastructure as a Service (IaaS) :

"Infrastructure as a service" (IaaS) refers to online services that provide high-level APIs used to abstract various low-level details of underlying network infrastructure like physical computing resources, location, data partitioning, scaling, security, backup, etc. A hypervisor runs the virtual machines as guests.

Pools of hypervisors within the cloud operational system can support large numbers of virtual machines and the ability to scale services up and down according to customers' varying requirements. Linux containers run in isolated partitions of a single Linux kernel running directly on the physical hardware.

Linux groups and namespaces are the underlying Linux kernel technologies used to isolate, secure and manage the containers. Containerization offers higher performance than virtualization because there is no hypervisor overhead. Also, container capacity auto-scales dynamically with computing load, which eliminates the problem of over-provisioning and enables usage-based billing [21]. IaaS clouds often offer additional resources such as a virtual-machine disk-image library, raw block storage, file or object storage, firewalls, load

balancers, IP addresses, virtual local area networks (VLANs), and software bundles [22]. The NIST's definition of cloud computing describes IaaS as "where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls). IaaS-cloud providers supply these resources on-demand from their large pools of equipment installed in data centers [20]. For wide-area connectivity, customers can use either the Internet or carrier clouds (dedicated virtual private networks). To deploy their applications, cloud users install operating-system images and their application software on the cloud infrastructure. In this model, the cloud user patches and maintains the operating systems and the application software. Cloud providers typically bill IaaS services on a utility computing basis: cost reflects the amount of resources allocated and consumed [23].

8.2 Platform as a service (PaaS) :

The NIST's [20] defines Platform as a Service as the capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

PaaS vendors offer a development environment to application developers. The provider typically develops toolkit and standards for development and channels for distribution and payment. In the PaaS models, cloud providers deliver a computing platform, typically including operating system, programming-language execution environment, database, and web server. Application developers develop and run their software on a cloud platform instead of directly buying and managing the underlying hardware and software layers. With some PaaS, the underlying computer and storage resources scale automatically to match application demand so that the cloud user does not have to allocate resources manually [24].

8.3 Software as a service (SaaS) :

The NIST's definition of cloud computing defines Software as a Service [20] as

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either

a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

In the software as a service (SaaS) model, users gain access to application software and databases. Cloud providers manage the infrastructure and platforms that run the applications. SaaS is sometimes referred to as "on-demand software" and is usually priced on a pay-per-use basis or using a subscription fee [25]. In the SaaS model, cloud providers install and operate application software in the cloud and cloud users access the software from cloud clients. Cloud users do not manage the cloud infrastructure and platform where the application runs. This eliminates the need to install and run the application on the cloud user's own computers, which simplifies maintenance and support. Cloud applications differ from other applications in their scalability which can be achieved by cloning tasks onto multiple virtual machines at run-time to meet changing work demand [26]. Load balancers distribute the work over the set of virtual machines. This process is transparent to the cloud user, who sees only a single access-point. To accommodate a large number of cloud users, cloud applications can be multitenant, meaning that any machine may serve more than one cloud-user organization.

9 Virtualization

It is a very useful concept in context of cloud systems. Virtualization means ”something which isn’t real”, but gives all the facilities of a real. It is the software implementation of a computer which will execute different programs like a real machine.

Virtualization is related to cloud, because using virtualization an end user can use different services of a cloud. The remote datacenter will provide different services in a full or partial virtualized manner.

Two types of virtualization are found in case of clouds: Full virtualization, and Para virtualization.

9.1 Full Virtualization :

In case of full virtualization a complete installation of one machine is done on another machine. It will result in a virtual machine which will have all the softwares that are present in the actual server.

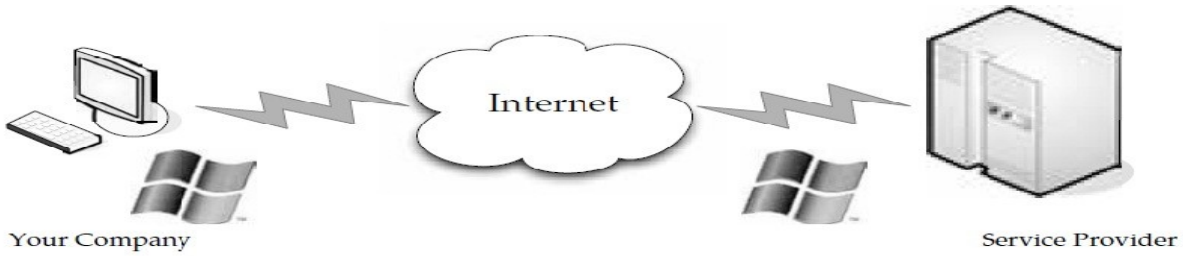


Figure 6 : Full Virtualization

Here the remote datacenter delivers the services in a fully virtualized manner. Full virtualization has been successful for several purposes as pointed out in:

- ✓ Sharing a computer system among multiple users
- ✓ Isolating users from each other and from the control program
- ✓ Emulating hardware on another machine

9.2 Para virtualization

In para virtualization, the hardware allows multiple operating systems to run on single machine by efficient use of system resources such as memory and processor. e.g. VMware software. Here all the services are not fully available, rather the services are provided partially.

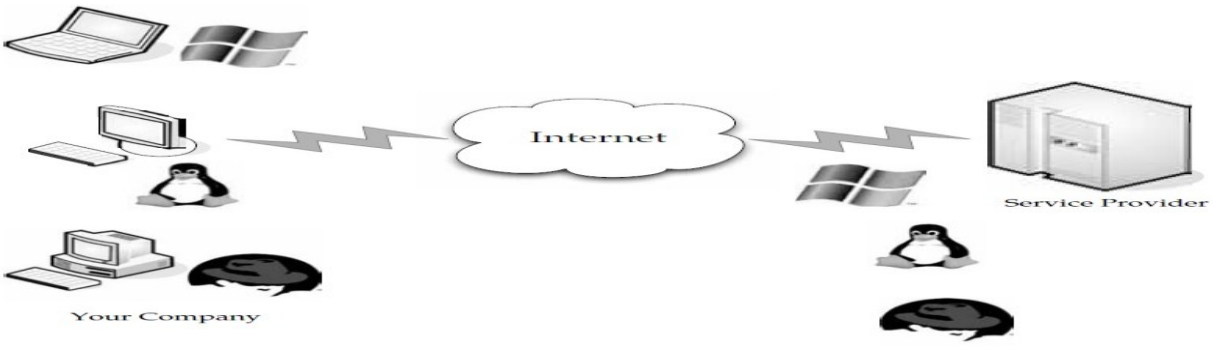


Figure 7 : Para virtualization

Para virtualization has the following advantages as given bellow:

Disaster recovery: In the event of a system failure, guest instances are moved to another hardware until the machine is repaired or replaced.

Migration: As the hardware can be replaced easily, hence migrating or moving the different parts of a new machine is faster and easier.

Capacity management: In a virtualized environment, it is easier and faster to add more hard drive capacity and processing power. As the system parts or hardwares can be moved or replaced or repaired easily, capacity management is simple and easier.

10 Deployment models

10.1 Private cloud :

Private cloud is cloud infrastructure operated solely for a single organization, whether managed internally or by a third party, and hosted either internally or externally [20]. Undertaking a private cloud project requires significant engagement to virtualize the business environment, and requires the organization to reevaluate decisions about existing resources. It can improve business, but every step in the project raises security issues that must be addressed to prevent serious vulnerabilities.

Self-run data centers [27] are generally capital intensive. They have a significant physical footprint, requiring allocations of space, hardware, and environmental controls. These assets have to be refreshed periodically, resulting in additional capital expenditures. They have attracted criticism because users "still have to buy, build, and manage them" and thus do not

benefit from less hands-on management [28], essentially "[lacking] the economic model that makes cloud computing such an intriguing concept" [29][30].

10.2 Public cloud:

A cloud is called a "public cloud" when the services are rendered over a network that is open for public use. Public cloud services may be free [31]. Technically there may be little or no difference between public and private cloud architecture, however, security consideration may be substantially different for services (applications, storage, and other resources) that are made available by a service provider for a public audience and when communication is effected over a non-trusted network. Generally, public cloud service providers like Amazon Web Services (AWS), IBM Cloud, Oracle, Microsoft, Google, and Alibaba own and operate the infrastructure at their data center and access is generally via the Internet. AWS, Oracle, Microsoft, and Google also offer direct connect services called "AWS Direct Connect", "Oracle FastConnect", "Azure ExpressRoute", and "Cloud Interconnect" respectively, such connections require customers to purchase or lease a private connection to a peering point offered by the cloud provider [32][33].

10.3 Hybrid cloud:

Hybrid cloud is a composition of a public cloud and a private environment, such as a private cloud or on-premises resources [34][35], that remain distinct entities but are bound together, offering the benefits of multiple deployment models. Hybrid cloud can also mean the ability to connect collocation, managed and/or dedicated services with cloud resources [20]. Gartner defines a hybrid cloud service as a cloud computing service that is composed of some combination of private, public and community cloud services, from different service providers [36]. A hybrid cloud service crosses isolation and provider boundaries so that it can't be simply put in one category of private, public, or community cloud service. It allows one to extend either the capacity or the capability of a cloud service, by aggregation, integration or customization with another cloud service. Varied use cases for hybrid cloud composition exist. For example, an organization may store sensitive client data in house on a private cloud application, but interconnect that application to a business intelligence application provided on a public cloud as a software service [37]. This example of hybrid cloud extends the capabilities of the enterprise to deliver a specific business service through the addition of externally available public cloud services. Hybrid cloud adoption depends on a number of factors such as data security and compliance requirements, level of control needed over data,

and the applications an organization uses [38]. Hybrid cloud infrastructure essentially serves to eliminate limitations inherent to the multi-access relay characteristics of private cloud networking. The advantages include enhanced runtime flexibility and adaptive memory processing unique to virtualized interface models [39].

11 Conclusion

In this chapter an overview of distributed systems are presented. The architecture, various characteristics, and the myths on distributed computing are discussed. Further to the difference between Parallel and distributed computing, and introduction to cloud computing and we talk about the important concept in this field like service model and deployment model. The future of distributed computing is still quite uncertain since it is one of many new types of computing. The technology has truly shown its worth as a useful tool for various complex applications. In conclusion, cloud computing is recently new technological development that has the potential to have a great impact on the world. It has many benefits that it provides to it users and businesses.

LAOD BALANCING

Chapter Plan

1.Introduction

2. Definition of load balancing

3. History of Load Balancing

4. How a load balancer is used

5. Load Balancing Algorithm

6. Load Balancer types and Benefits

7. Benefits of Load Balancing

8. Conclusion

1 Introduction

Internet technologies is fast growing and is being used extensively, with it Cloud Computing became a hot topic of industry and academia as an emerging new computing mechanism. It is supposed to provide computing as the utility to meet the everyday needs of the general community. Its infrastructure is used by businesses and users to access application services from anywhere in the world on demand. Though there is a glorious future of Cloud Computing, many crucial problems still need to be solved for the realization of cloud computing. Load balancing is one of these problems; it plays a very important role in the realization of Cloud Computing. Load balancing in cloud computing is to distribute the local workload evenly to the whole cloud. In fact it has become indispensable for cloud computing. It is used by Cloud service provider (CSP) in its own cloud computing platform to provide a high efficient solution for the user. Also, a inter CSP load balancing mechanism is needed to construct a low cost and infinite resource pool for the consumer. Thus Load balancing in cloud computing provides an organization with the ability to distribute application requests across any number of application deployments located in data centers and through cloud computing providers. In this chapter we will provide an overview of Load balancing, the definition, its history and methods, and its goals.

2 Definition of load balancing

Load Balancing Definition: Load balancing is the process of distributing network traffic across multiple servers. This ensures no single server bears too much demand. By spreading the work evenly, load balancing improves application responsiveness. It also increases availability of applications and websites for users. Modern applications cannot run without load balancers. Over time, software load balancers have added additional capabilities including security and application.

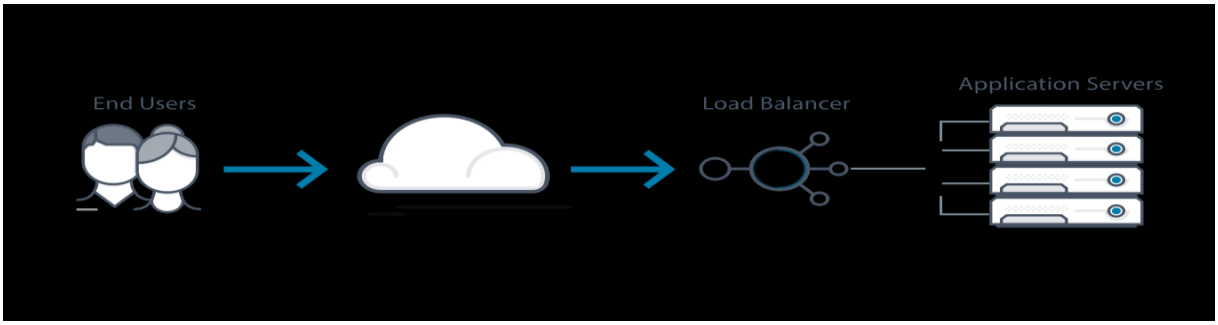


Figure 8 : Load Balancing

Another definition is: In computing, load balancing refers to the process of distributing a set of tasks over a set of resources (computing units), with the aim of making their overall processing more efficient. Load balancing techniques can optimize the response time for each task, avoiding unevenly overloading compute nodes while other compute nodes are left idle. Load balancing is the subject of research in the field of parallel computers.

3 History of Load Balancing

Load balancing got its start in the 1990s as hardware appliances distributing traffic across a network. Organizations wanted to improve accessibility of applications running on servers. Eventually, load balancing took on more responsibilities with the advent of Application Delivery Controllers (ADCs). They provide security along with seamless access to applications at peak times. ADCs fall into three categories: hardware appliances, virtual appliances (essentially the software extracted from legacy hardware) and software-native load balancers. As computing moves to the cloud, software ADCs performs similar tasks to hardware. They also come with added functionality and flexibility. They let an organization quickly and securely scale up its application services based on demand in the cloud. Modern ADCs allow organizations to consolidate network-based services. Those services include SSL/TLS offload, caching, compression, intrusion detection and web application firewalls. This creates even shorter delivery times and greater scalability.

4 How a load balancer is used

Usually, a load balancer is present between a client and a server and accepts incoming requests from the clients and distributes the requests evenly across multiple servers. The requests can be in the form of network or application traffic. This uniform distribution of load helps to avoid the single point of failure in the system. Think of a situation where most of your data lie in a single server and it stops working, sounds scary, right?

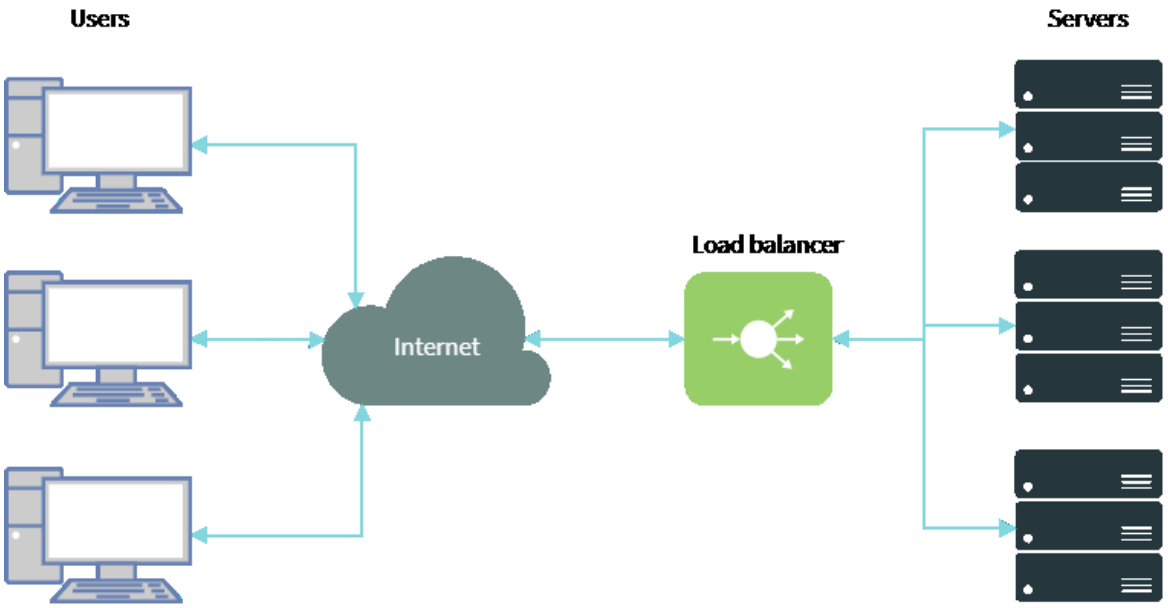


Figure 9 : Usag of Laod Balncing

The usage of load balancers is not limited between clients and servers. Load balancers can be used between every layer of the system to distribute load uniformly.

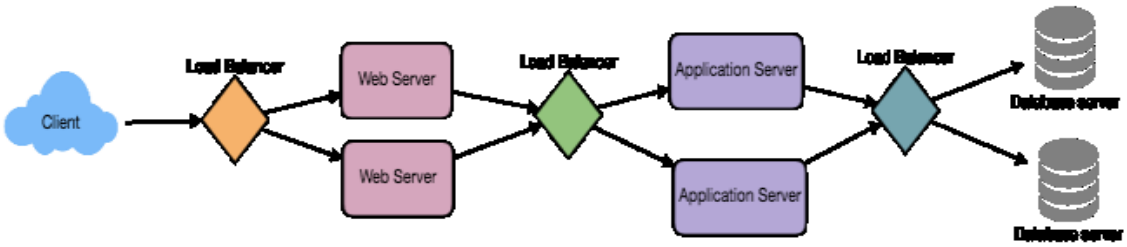


Figure 10 : Multiple place to use Load balancer

In the above picture, a load balancer is there between the client and the server to distribute network traffic. Another load balancer exists between the web server and application server to distribute application traffic and then a load balancer is used between the application layer and the database layer to handle database servers.

5 Load Balancer types and Benefits

5.1 Types

There are a number of specific types of load balancing you might need to consider for your network, including SQL Server load balancing for your relational database, global server load balancing for troubleshooting across multiple geographic locations, and DNS server load balancing to ensure domain name functionality. You can also think about types of load balancers in terms of the various cloud-based balancers available (including the well-known AWS Elastic Load Balancer

- ***Network Load Balancing:*** Network load balancing, as its name suggests, leverages network layer information to decide where to send network traffic. This is accomplished through layer 4 load balancing, which is designed to handle all forms of TCP/UDP traffic. Network load balancing is considered the fastest of all the load balancing solutions, but it tends to fall short when it comes to balancing the distribution of traffic across servers.
- ***HTTP(S) Load Balancing:*** HTTP(S) load balancing is one of the oldest forms of load balancing. This form of load balancing relies on layer 7, which means it operates in the application layer. HTTP load balancing is often dubbed the most flexible type of load balancing because it allows you to form distribution decisions based on any information that comes with an HTTP address.
- ***Internal Load Balancing:*** Internal load balancing is nearly identical to network load balancing but can be leveraged to balance internal infrastructure.

When talking about types of load balancers, it's also important to note there are hardware load balancers, software load balancers, and virtual load balancers.

- ***Hardware Load Balancer:*** A hardware load balancer, as the name implies, relies on physical, on-premises hardware to distribute application and network traffic. These devices can handle a large volume of traffic but often carry a hefty price tag and are fairly limited in terms of flexibility.
- ***Software Load Balancer:*** A software load balancer comes in two forms—commercial or open-source—and must be installed prior to use. Like cloud-based balancers, these tend to be more affordable than hardware solutions.

- ***Virtual Load Balancer:*** A virtual load balancer differs from software load balancers because it deploys the software of a hardware load balancing device on a virtual machine.

6 Load balancing algorithm

Depending on the current state of the system, load balancing algorithms can be divided into two categories:

Static: It doesn't depend on the current state of the system. Prior knowledge of the system is needed

Dynamic: Decisions on load balancing are based on current state of the system. No prior knowledge is needed. So it is better than static approach.

6.1 Dynamic Load balancing algorithm

In a distributed system, dynamic load balancing can be done in two different ways:
distributed and non-distributed. In the distributed one, the dynamic load balancing algorithm is executed by all nodes present in the system and the task of load balancing is shared among them. The interaction among nodes to achieve load balancing can take two forms: cooperative and non-cooperative [2]. In the first one, the nodes work side-by-side to achieve a common objective, for example, to improve the overall response time, etc. In the second form, each node works independently toward a goal local to it, for example, to improve the response time of a local task. Dynamic load balancing algorithms of distributed nature, usually generate more messages than the non-distributed ones because, each of the nodes in the system needs to interact with every other node. A benefit, of this is that even if one or more nodes in the system fail, it will not cause the total load balancing process to halt; it instead would affect the system performance to some extent. Distributed dynamic load balancing can introduce immense stress on a system in which each node needs to interchange status information with every other node in the system. It is more advantageous when most of the nodes act individually with very few interactions with others. In non-distributed type, either one node or a group of nodes do the task of load balancing. Non-distributed dynamic load balancing algorithms can take two forms: centralized and semi-distributed. In the first form, the load balancing algorithm is executed only by a single node in the whole system: the central node. This node is solely responsible for load balancing of the whole system. The other nodes interact only with the central node. In semi-distributed form, nodes of the system are partitioned into clusters, where the load balancing in each cluster is of centralized form. A central node is elected in each cluster by appropriate election technique which takes care of load

balancing within that cluster. Hence, the load balancing of the whole system is done via the central nodes of each cluster [40]. Centralized dynamic load balancing takes fewer messages to reach a decision, as the number of overall interactions in the system decreases drastically as compared to the semi-distributed case. However, centralized algorithms can cause a bottleneck in the system at the central node and also the load balancing process is rendered useless once the central node crashes. Therefore, this algorithm is most suited for networks with small size.

6.2 Policies or Strategies in dynamic load balancing

There are 4 policies [40]:

a) ***Transfer Policy:*** The part of the dynamic load balancing algorithm which selects a job for transferring from a local node to a remote node is referred to as Transfer policy or Transfer strategy

b) ***Selection Policy:*** It specifies the processors involved in the load exchange (processor matching)

c) ***Location Policy:*** The part of the load balancing algorithm which selects a destination node for a transferred task is referred to as location policy or Location strategy.

d) ***Information Policy:*** The part of the dynamic load balancing algorithm responsible for collecting information about the nodes in the system is referred to as Information policy or Information strategy:

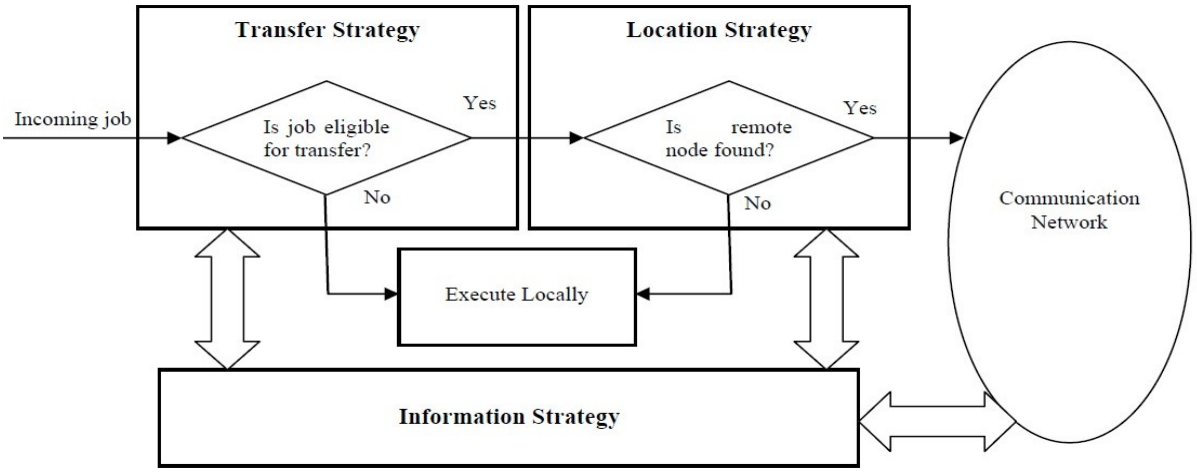


Figure 11 : Information Policy

7 Benefits of load balancer

- ✓ ***Less Response time****:* The system does not have to be stuck waiting for a response from an unhealthy server.
- ✓ ***Improved availability and higher throughput:*** Even in case of server failure, the load balancer will redirect the load and make sure the system is up.
- ✓ ***Troubleshoot incoming traffic:*** Load balancers allow the capability to do predictive analytics on incoming traffic and detect traffic bottlenecks.
- ✓ ***Avoid single point of failure****:* Prevents the system from going down due to failure of one component i.e. avoids dependency on a single component.

8 Conclusion

This chapter explains the concept of load balancing, types of load balancing algorithms, general idea about dynamic load balancing algorithms and the different policies that can be used in it and shows the benefits we got when using the load balancing concept.

Unbalancing Load Detection

Chapter Plan

1. Introduction
2. Proposed architecture
3. Operation of the Proposed Model
4. Model Activity Diagram and Pseudo Code
5. Conclusion

1 Introduction

Load balancing is a generic term used for distributing a larger processing load to smaller processing nodes for enhancing the overall performance of system. In a distributed system environment, it is the process of distributing load among various other nodes of distributed system to improve both resource utilization and job response time. An ideal load balancing algorithm should avoid overloading or under loading of any specific node. But, in case of a cloud computing environment the selection of load balancing algorithm is not easy because it involves additional constraints like security, reliability, throughput etc. So, the main goal of a load balancing algorithm in a cloud computing environment is to improve the response time of job by distributing the total load of system. The algorithm must also ensure that it is not overloading any specific node. The load balancing in clouds may be among physical hosts or VMs. This balancing mechanism distributes the dynamic workload evenly among all the nodes (hosts or VMs). The load balancing in the cloud is also referred as load balancing as a service (LBaaS). There are two versions of load balancing algorithms: static and dynamic. The static-based balancing algorithms are mostly fit for stable environments with homogeneous system. Dynamic-based balancing algorithms are more adaptable and effective in both homogeneous and heterogeneous environment. However, the application of static load balancing procedures has less system overhead as compared to the dynamic load balancing procedures. A large number of heuristics has been proposed in the literature to this problem. In cloud computing environment, the allocation of different tasks to VMs is known as the load. We can define the load balancing problem in various ways as follows. Task allocation: it represents the random distribution of a finite number of tasks into different Physical Machines (PMs) which again allocated to different VMs of respective PM. The efficiency of task allocation to the cloud determines the effectiveness of the load balancing algorithm; VM/Task Migration Management- in Cloud Computing Environment, VM Migration is nothing but the movement of a VM from one PM to another PM to improving the resource utilization of the data center for which the PM is overloaded. Similarly, the migration of the current state of a task from one VM to another VM or VM of one host to VM of another host is referred to as task migration. This is the reason; the VM or task migration plays a major role in load balancing of cloud computing. In this chapter, we propose a model that enables to reduce the workload on virtual machines in load balancing. The proposed model is made up of **three essential** components: the algorithm selection, the detection of workload in VMs (virtual

machines), and the final process is the Migrate. The results obtained through the case study show that our model effectively allows to reduce the workload on virtual Machines.

2 Proposed architecture

This section describes the proposed architecture (see Figure 12). The latter consists of two principal components: the datacenter broker and our middleware. In this section, we will explain the most important concept and component in our architecture.

2.1 **Cloudlet:**

This class models the Cloud-based application services (content delivery, social networking, business workflow), which are commonly deployed in the data centers. Our Architecture represents the complexity of an application in terms of its computational requirements. Every

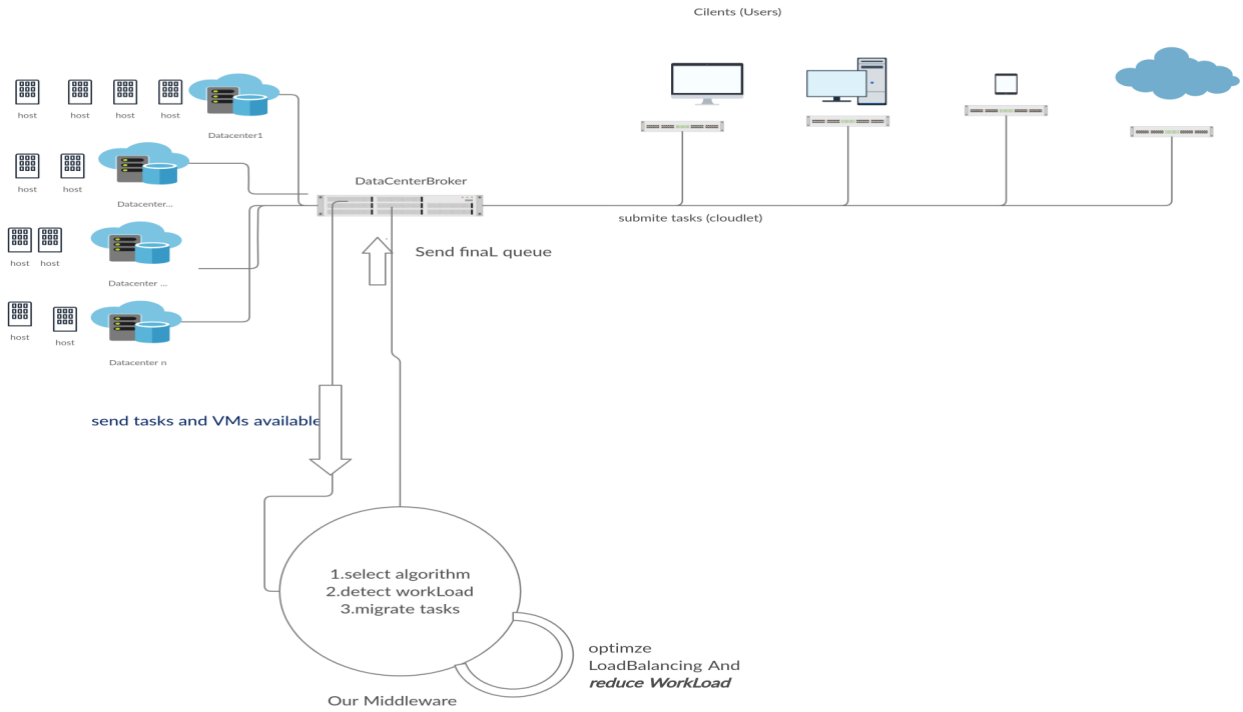


Figure 12 : Proposed architecture

application component has a pre-assigned instruction length and amount of data transfer that needs to be undertaken for successfully hosting the application.

2.1 Virtual Machine:

This class models an instance of a VM, whose management during its life cycle is the responsibility of the Host component. As discussed earlier, a host can simultaneously instantiate multiple VMs and allocate cores based on predefined processor sharing policies (space-shared, time-shared). Every VM component has access to a component that stores the characteristics related to a VM, such as memory, processor, storage, and the VM's internal scheduling policy, which is extended from the abstract component called VMScheduling.

2.2 **The queue:**

The queue is a structure that allows you to order tasks, on hold execution, according to their order of arrival. Its structure allows direct access to tasks.

2.3 MyDetectLoadBalncer :

This is the principle class it has a lot of functions like determine the best algorithm to choose from load Balancing algorithm and ability to do migration of task from virtual machine to another.

3 Operation of the Proposed Model

As shown in Figure 13 , the components of the proposed model interact with each other in order to guarantee a fair load between the different virtual machines, which allows reduce response and service times. Thus, the operation of our model is divided into three levels, The first level allows us to select best algorithm for load balancing depending on average time and number of virtual machine used by the algorithm and we calculate that depending on cloudlet size from task sent from user and cpu speed (mips) of virtual machine We call this level the *levels comparison* ,The second level allows us to determine and detect workload on virtual machines depending on speed of execution in virtual machine and the average time from load balancing algorithm and it is considered one of the most important steps in our architecture because she determine if we can handle the workload or not, The third level allows us to migrate a cloudlet(task) from virtual machine to another depending on the workload from heavy to light ,and that by taking into account cpu speed and task size into consideration, and this is the last step in our architecture that we have suggested for the detection of load Balancing.

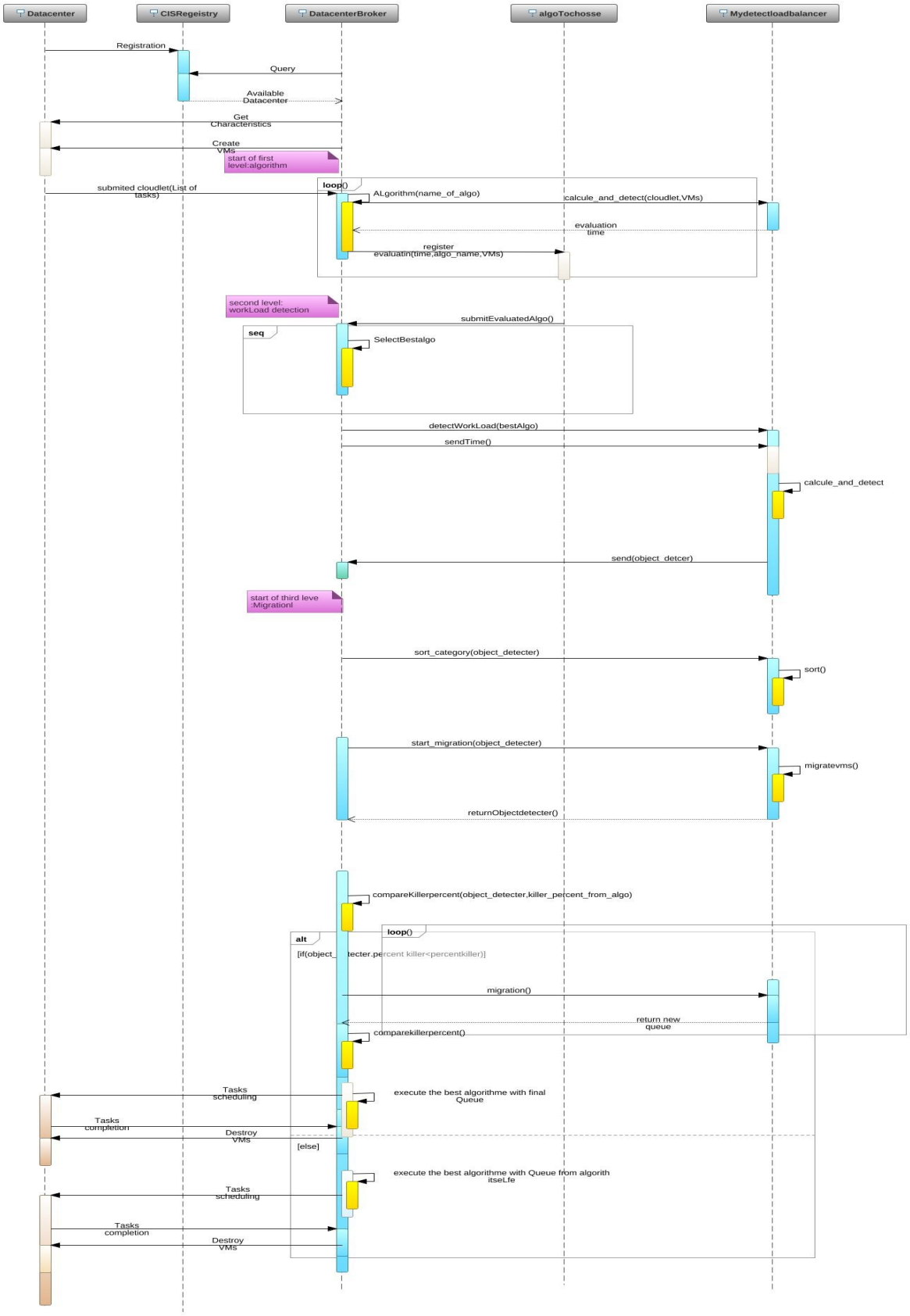


Figure 13 : sequance diagram of our Operation

3.1 Selection of algorithm

We implemented three algorithms in this work:

Round robin: is one of the simplest and most used load balancing algorithms. Client requests are distributed to application servers in rotation. For example, if you have three application servers: the first client request to the first application server in the list, the second client request to the second application server, the third client request to the third application server, the fourth to the first application server and so on. This load balancing algorithm does not take into consideration the characteristics of the application servers i.e. it assumes that all application servers are the same with the same availability, computing and load handling characteristics.

Randomly: in this algorithm we simply bind cloudlet to virtual machine randomly from VMs available.

Without load balancing: All cloudLet (tasks) will execute in one Virtual machine, that algorithm represent no Load Balancing at all.

After simulate execution we register the average time and the order of execution (task_id,vm_id) for each algorithm in AlgoToChosse CLass.

Execution time estimation (TE):

TE = size of cloudlet(task)/mips(speed of cpu on virtual machine)

Equation of the average time (avg_t):

$$\textbf{Avg\_t} = \frac{\sum_{k=0}^{n} \binom{n=\textbf{TE of the last task}}{k=\textbf{TE of first task}}}{number\ of\ virtual\ machine}$$

Queue: register every cloudlet and her virtual machine (ID of vm)

After calculate the average time we register this average in AlgoToChosse class with right algorithm name and the Queue.

3.2 detect workload on virtual machines

After the first level of our architecture now we move to second level ,which is detection of workload for each virtual machine exist on our datacenter, and that is by comparing the time

the virtual machine took and the average time of the best algorithm, and divide the results to three category according to percentage :

Percentage = $\sum_{TE=0}^{TE=n} \boldsymbol{VM}$ * 100 / **avg_t**

- **First category** : ***killer*** that's when the percentage go over 100%
- **Second category : *average*** that's when the percentage equal 100%
- **third category : *below*** that's when the percentage go below 100%

After set every virtual machine to their category we ready to go to the third level

3.3 Migration

in this level we transfer a task from heavy workload in killer category to below category , and that by arranging the virtual machines in both category from very heavy to heavy in killer category and vise versa in bellow category, after this operation we start with cloudlet(task) transfer process from very heavy to very light and so on. At the end of this process, we repeat the evaluation process if percentage in killer category is **less** than killer percentage from load balancing algorithm we keep the last one and repeat the whole level again until the percentage get bigger than the last one, but if the killer percentage is bigger than percentage from load balancing algorithm we stop and keep the original process.

4 activity diagram and pseudo code

4.1 General activity diagram:

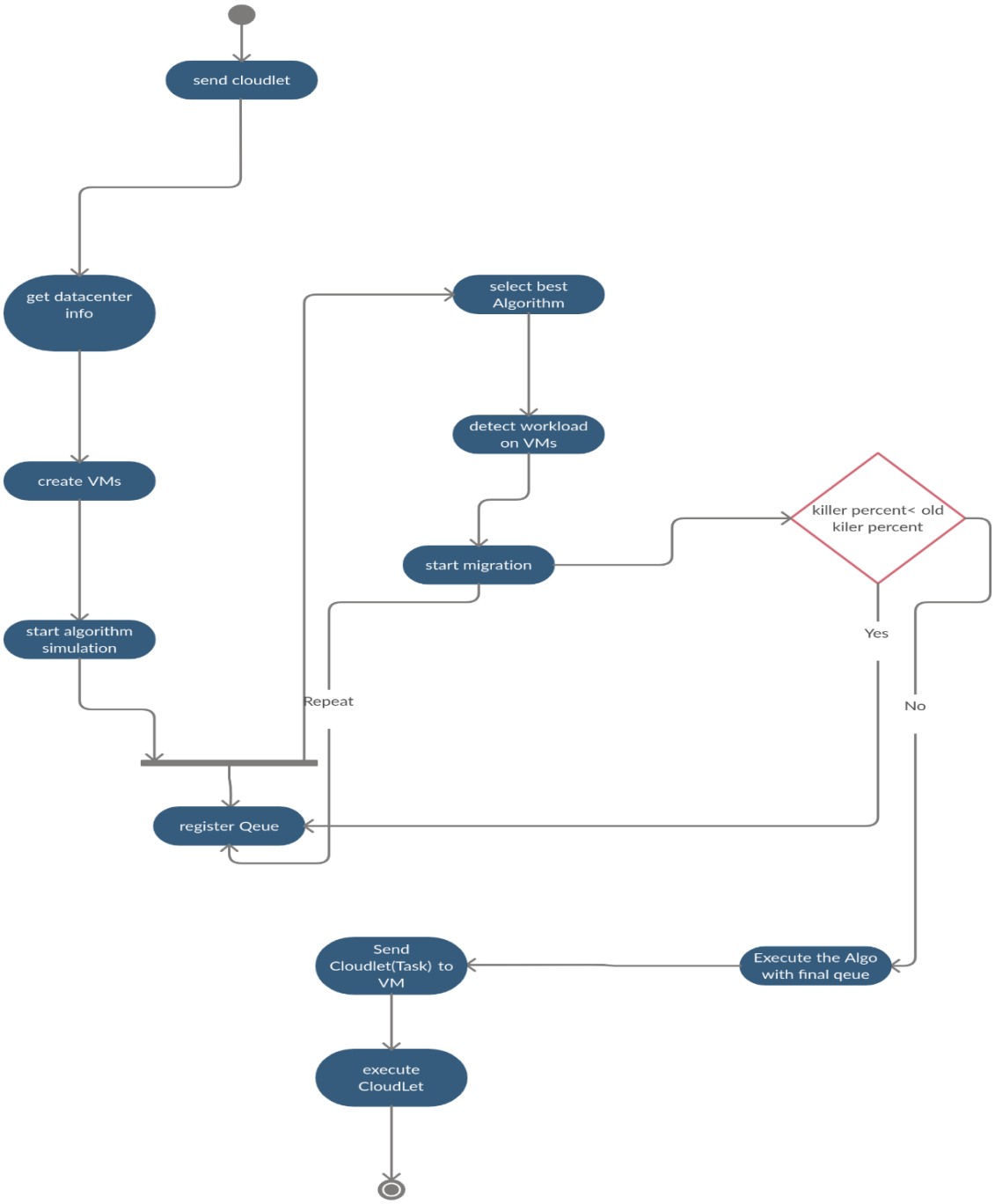


Figure 14 : general activity diagram

4.2 Selection diagram:

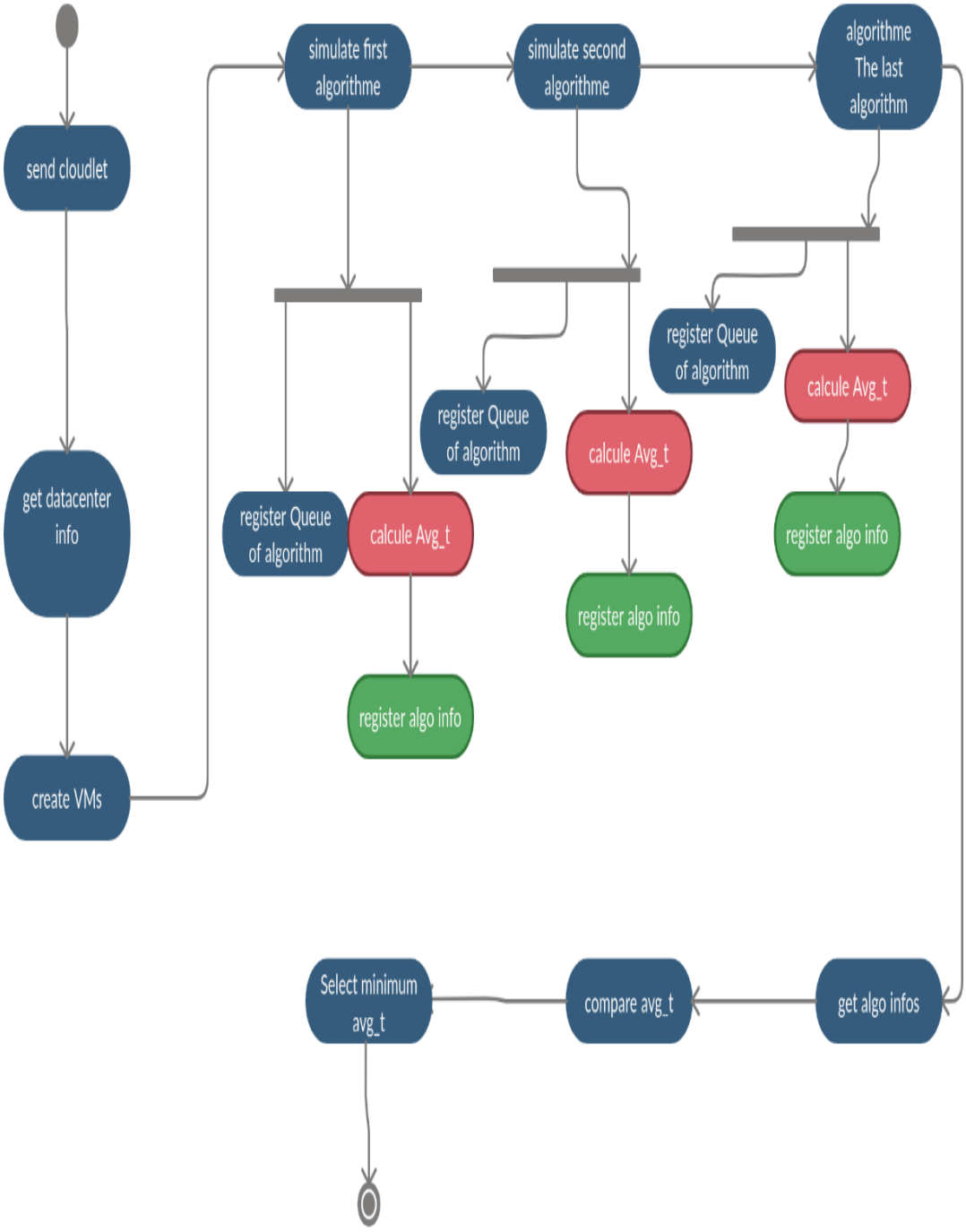


Figure 15 : Level one select best algorithm diagram

1. receive tasks
2. Get datacenter info from cloud information service (CIS)
3. Creat Vms from VMAllocationSimple class
4. Simulate first algorithm(round robin):

initial the queue of round robin algorithm so queue.size == 0

If(queue==0) : boolean paramter == false == don't execute tasks

Loop : n=0;n<tasks.size

conect task n to vm n

Remove task from task array

Register vm id on Queue

Get next VMs

If(vm==vmList.size) : vm return to first one

looop : calcule time for every VMs

Register time in time_array

Register algorithm info in algotochooseList<algotochosse>(time,algo_name,vm_used(size)).

5. Simulate seconde algorithm(random):

Like the first one but the deffrence we choose VMs randomly from VMs availble to us not in order

6.Simulate tird algorithm(no Load balancer)

We submite all the tasks to one VMs exist .. always the first one

7.after end simulate ..get algotochooseList and compare every average time from the three algorith

(the calcule happen inside the classe itself : **Avg_t** = $\frac{\sum_{k=0}^{n} \left(\substack{n=TE\ of\ the\ last\ task \\ k=TE\ of\ first\ task}\right)}{number\ of\ virtual\ machine}$

8.choose the small from thous three like the best algorith in this scenario.

4.3 Detect workload diagram:

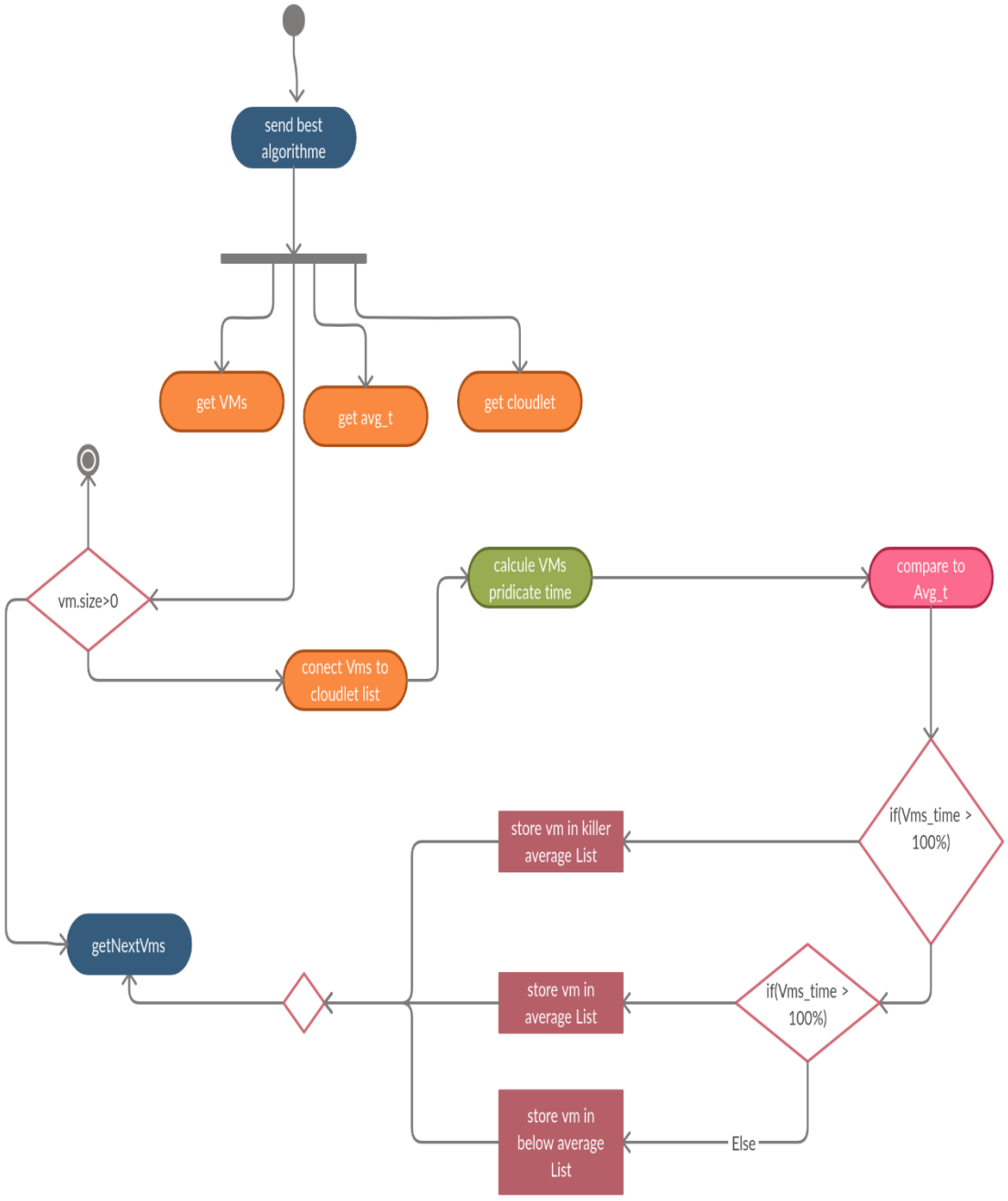


Figure 16 : *level two detection of workload diagram*

After select best algorithm from level 1 now we move to second level

1. get tasks ans avg_t from algo instance and Vms

2. we need to connect every task to her Vm

Loop : n=0;n<queue.size

Queue : represt order == the cell itself represnt the Vm_id

Time_of_current_vm =0;

2.1 get current vm_id (queue [n]) and get file size from cloudlet (cloudLetList.get(n))

2.2 Time_of_current_vm + = file_size / currentVm.mips(mips = million instraction per second)

3. Start detection Loop : n=0;n<Vms.size :

3.1 get time from current Vm(i)

3.2 Start calcule percent = time * 100 / avg_t

3.3 Depending on percent we class the Vm on 3 categorey :

Percent > 100%: killer caregry

Percent == 100% : average category

Percent < 100% : bellow average category

4.we need to return Object we three category store on it.

4.4 Migration diagram:

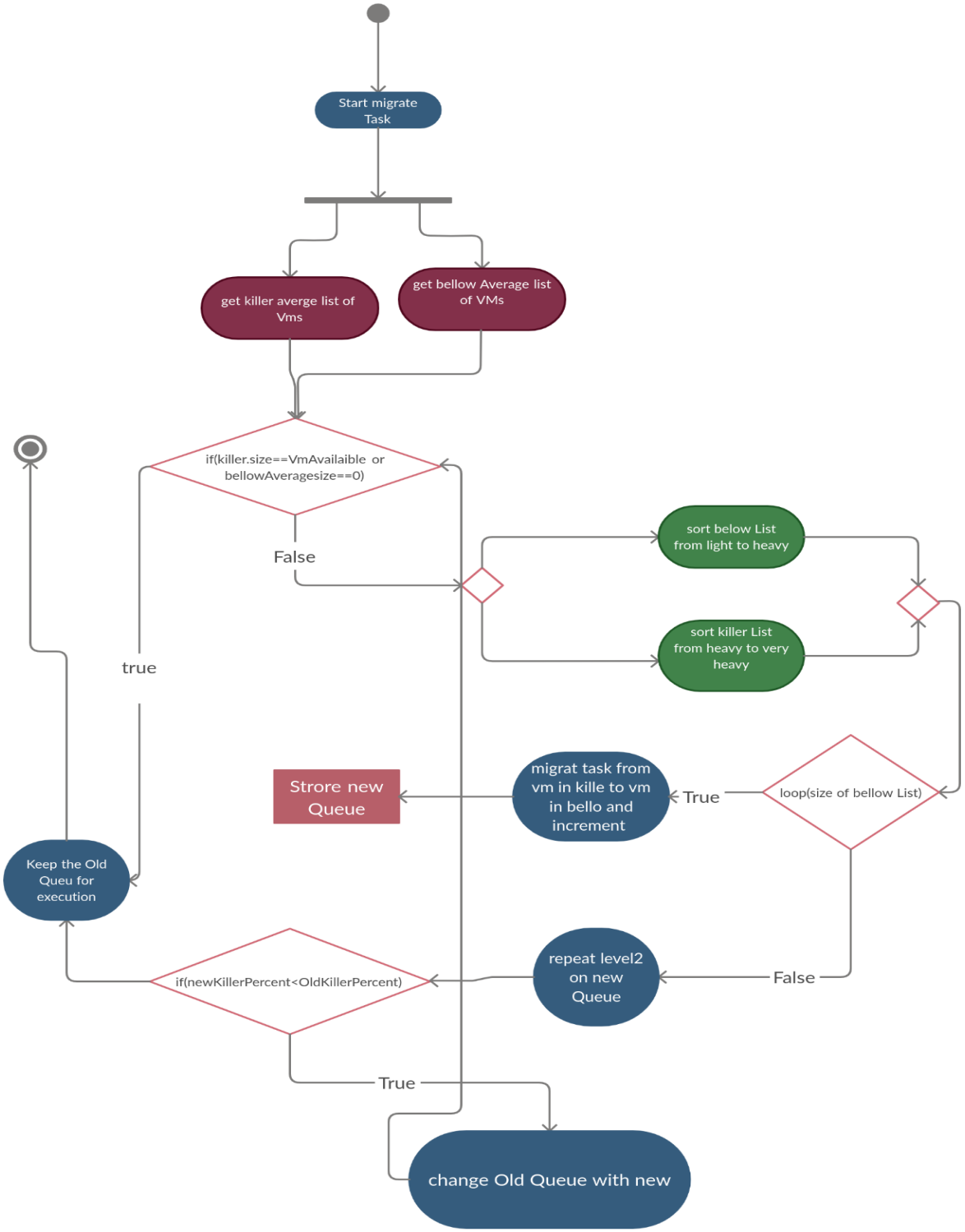


Figure 17 : level three migration diagram

After detect workload from level 2 now we move to third and the last level in our architecture

1. get list of killer VMs and List of VMs
2. Sort list of killer from very heavy vm to heavy
3. Sort list of bellow average from very light to light
4. Loop : n=0;n<bellow averageList.size

   migrate KillerList[n] to bellow_average[n]
5. Store the new Queue
6. after finish w need to repeat level 2 to get the new killer percent
7. Compare the new_percent to old percent

if(new_percent<old_percent) ::first thing success but not stopping here

Old_queue = new queue

Repeat the level 3 again

Else : we can't do migration and stop here and

Old_queue we don't touch him

5 Conclusion

In this chapter, we have addressed the problem of load imbalance in load balance process, by proposing an architecture based on two essential components. The operation of our model to balance the load takes place in three main phases: the first phase consists in select the best algorithm for the current scenario, the second phase is detect the workload in Virtual machine and this according to the characteristics of the tasks (task size) and VMs characteristics like mips (cpu power), as for the third phase, it balances VMs by emigrating process.

In the next chapter, we plan to validate our model by performing simulations in the CloudSim environment in order to show the relevance of our model on the one hand, and many other parameters that we will allow us to highlight the positive and negative points of our proposal.

CHAPTER FOUR

Implementation

Chapter Plan

1. Introduction

2. Definition of computer simulation

3. Language and development environment

4. Descrption of the application<<load balancing>>

5. Conclusion

1 Introduction

In the previous chapter, we proposed an architecture of our model based on the size of the task and execution time with three level, and in order to illustrate the different ideas and concepts included in the proposed architecture, we will unfold the main aspects of our architecture on a concrete example to show the feasibility and highlighting of our ideas. In this chapter we will first present the work environment, namely the different programming languages as well as the hardware and software environment that we will need to validate our architecture through simulation.

2 Definition of computer simulation

A computer simulation is the usage of a computer for the imitation of a real-world process or system. The dynamic responses of one system are represented by the behavior of another system, which is largely modeled on the former. A simulation requires a model, or a mathematical description of the real system. This is in the form of computer programs, which encompass the key characteristics or behaviors of the selected system. Here, the model is basically a representation of the system and the simulation process is known to depict the operation of the system in time.

3 Language and development environment

3.1 Java programming language:

The Java programming language was developed by Sun Microsystems in the early 1990s. Although it is primarily used for Internet-based applications, Java is a simple, efficient, general-purpose language. Java was originally designed for embedded network applications running on multiple platforms. It is a portable, object-oriented, interpreted language.

Java is extremely portable. The same Java application will run identically on any computer, regardless of hardware features or operating system, as long as it has a Java interpreter. Besides portability, another of Java's key advantages is its set of security features which protect a PC running a Java program not only from problems caused by erroneous code but also from malicious programs (such as viruses). You can safely run a Java applet downloaded from the Internet, because Java's security features prevent these types of applets from

accessing a PC's hard drive or network connections. An applet is typically a small Java program that is embedded within an HTML page.

3.2 CloudSim:

CloudSim is a simulator that allows the modeling, simulation and experimentation of new Cloud Computing infrastructures and application services. In the case of Cloud Computing, Simulation tools like CloudSim provide significant benefits to customers and suppliers. For customers, it allows them to test their services in a controllable environment with no cost and to verify performance before publishing them to the clouds. For providers, it allows them to check rental types based on various price and charge.

Without these tools, both customers and vendors have to rely on imprecise assessments, or trial-and-error approaches, these approaches can lead to inefficient service performance. Additionally, CloudSim helps researchers and developers test the performance of a developed application service.

There are many advantages relating to the use of CloudSim, such as: (i) time efficiency: it takes very less time and effort to implement cloud computing applications and (ii) flexibility: developers can easily model and test the performance of their applications and services in heterogeneous environments (Microsoft Azure, Amazon EC2).

3.3 CloudSim Architecture:

The software structure of CloudSim and its components is represented by a layered architecture as shown in Figure N. The lowest level is the SimJava discrete event simulation engine, which implements the basic functionality required for the simulation frameworks of the CloudSim. Higher level includes queues, event processing, creation of system components (services, host, Datacenter, Broker, virtual machines), communication between components and management of the simulation clock.

CloudSim supports modeling and simulation of the Cloud-based Datacenter environment, such as management interfaces dedicated to VMs, memory, storage and bandwidth. The CloudSim layer manages the instantiation and execution of basic entities (VMs, hosts, datacenters, applications) during the simulation period. In the topmost layer of the simulation stack, we find the user code which exposes the configuration of the functionalities linked to the hosts (ex: number of machines, their specifications), Broker scheduling policies, applications (ex: number of tasks and their needs), VM, number of users

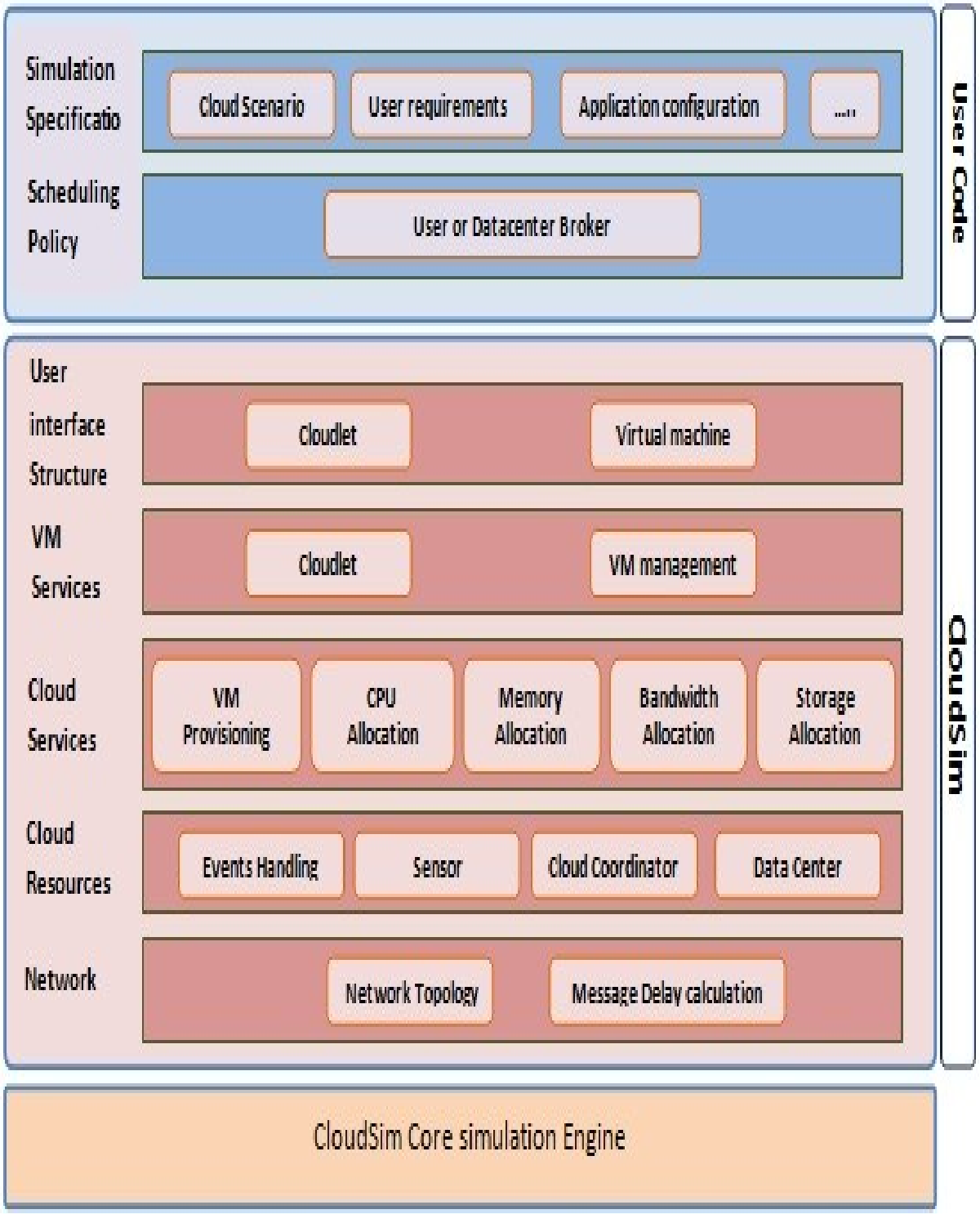


Figure 18 : CloudSim Architecture

Cloudlet:

In Cloudsim defined the workload, which is to be executed during the simulation run of the cloudsim simulation engine. Cloudlet in Cloudsim is a model class that exists inside the package 'org.cloudbus.cloudsim'. Cloudlet is one of the most important models which defined the specifications for a simulation engine corresponding to the real-life candidate application to be considered for moving to a Cloud-based system.

In contrast, we can also define this cloudlet as a single process or task being executed on the cloud-based system which is to be simulated through a Cloudsim simulation engine.

Virtual machine:

This class models a virtual machine (VM) instance, which is managed and hosted during its life cycle by the Cloud host component, a host can simultaneously instantiate multiple VMs and assign processor cores (shared space, shared time).

DataCenter:

This class models the core infrastructure of the service (hardware, software) offered by resource providers in a cloud computing environment. It encapsulates a set of computing machines which can be homogeneous or heterogeneous with regard to their resource configurations (memory, kernel, capacity and storage). In addition, each data center component instantiates a resource provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices. The HOST represents a physical computing server in a Cloud; it performs actions related to the management of virtual machines and has a policy defined for the provision of memory and bandwidth, as well as a policy of distribution of the PE (element processing) on virtual machines. A host is associated with a DataCenter, and can host virtual machines.

DataCenterBroker:

This class models the Broker, which is responsible for mediating between users and service providers according to the QoS conditions of users, and it deploys the service tasks through Clouds.

The Broker, acting on behalf of the users, identifies the appropriate cloud service providers through the Cloud Information Services (CIS) by negotiating with them for an allocation of resources that meet the QoS needs of the users.

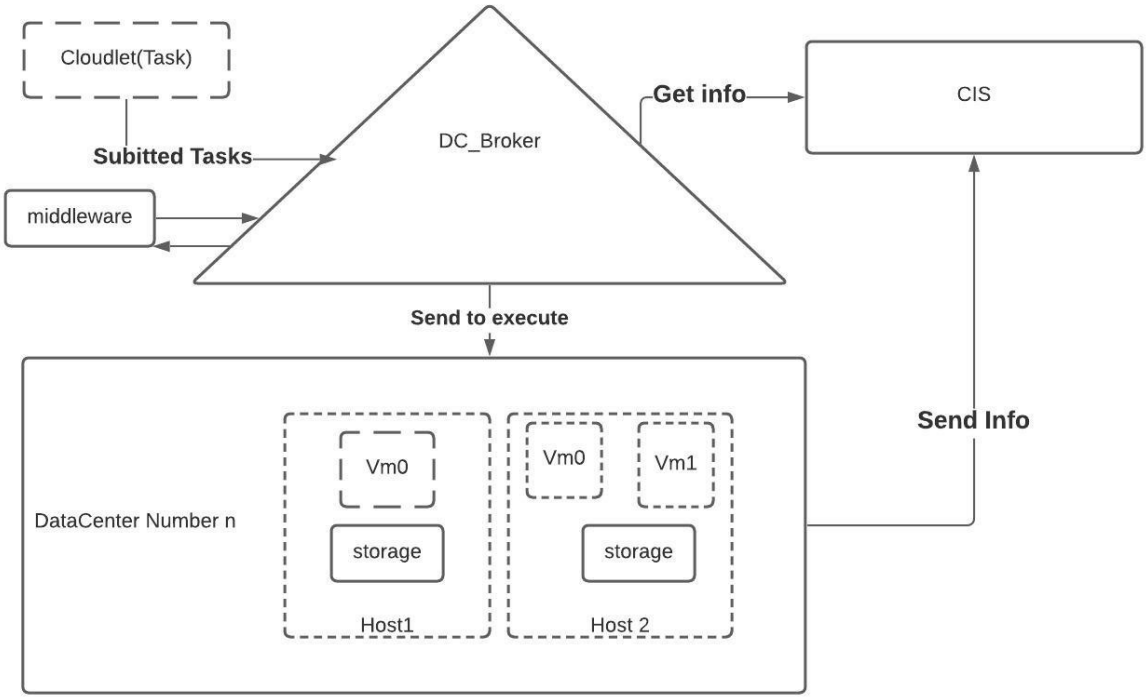


Figure 19 : interaction of our model and cloudsim architecture

4 Description of the application<<Loadbalancing Detection>>:

CloudSim version 3.0.3 has no graphical interface, it uses console mode while running, which makes it difficult for user to fully enjoy the simulator, such as display output graph (simulation result).

We have created a graphical interface that facilitates access and handling of the simulator, we will present the different steps carried out to carry out our simulation. Our “Load Balancing Detection” application is made up of a main interface, which contains eight scenarios and button to start simulation and two radio button for choose virtual machine scheduling policy. At the end of simulation we can click more information about the tasks and time of execution and datacenters.

Main interface:

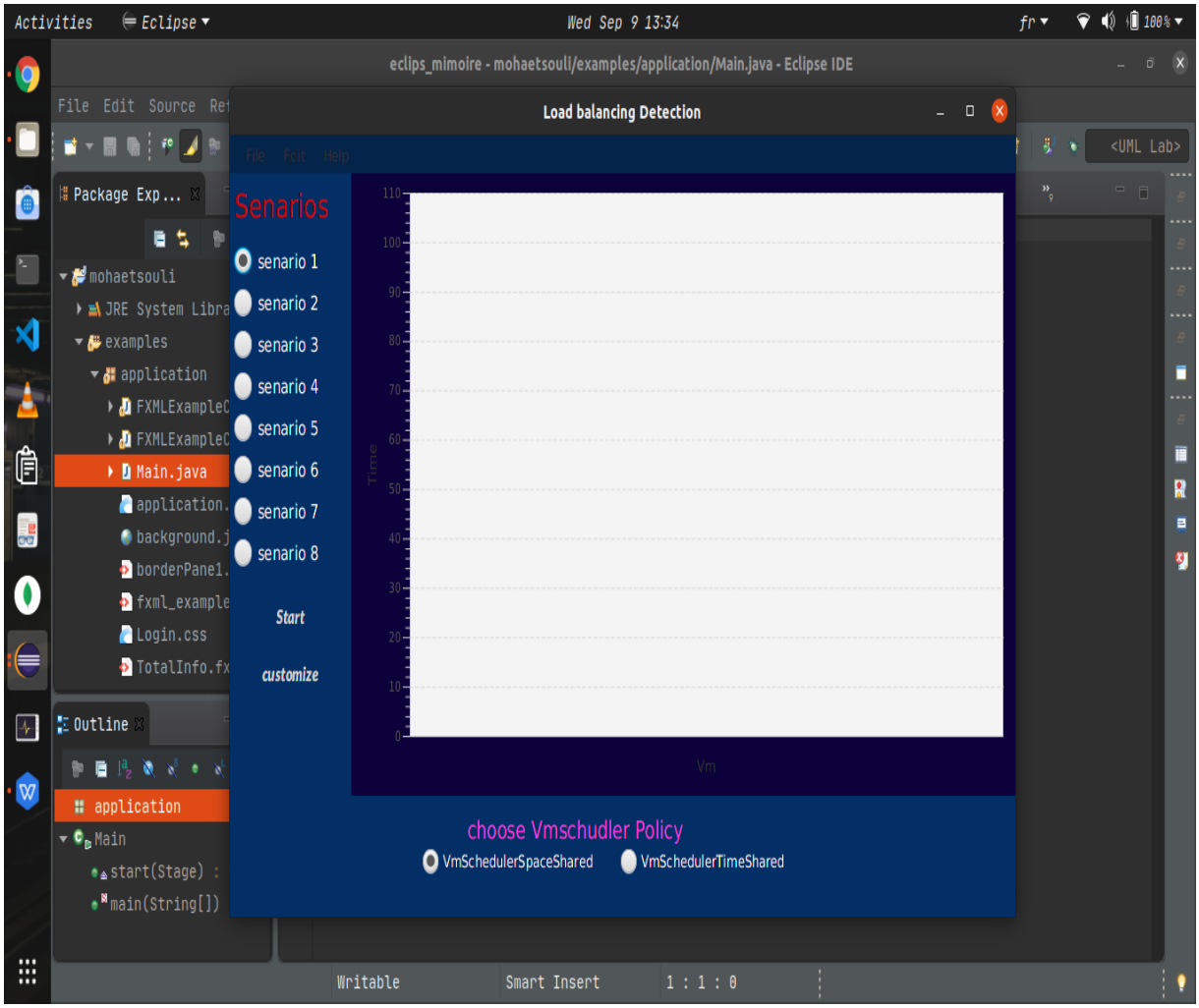


Figure 20 : main interface

4.1 How works:

first choose the scenario you want to simulate (when click to choose you will see the description of the scenario),and start the simulation after the simulation finish you will see the result like in figure 21 below You can find more information by read interface result and by click on more_info button.

At the end of the execution, we obtain the results as shown on figure 21:

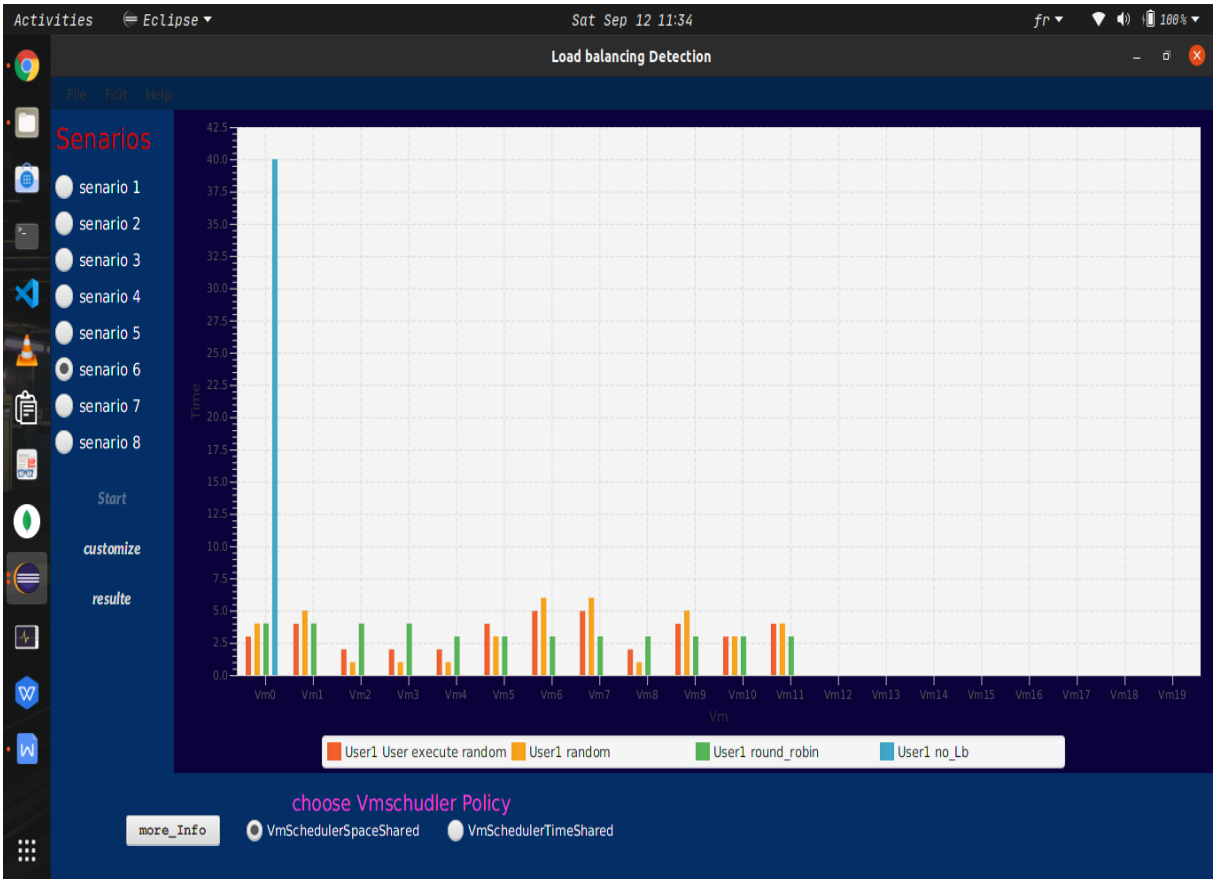


Figure 21 : The obtained results

Results interpretation :

The bar charts represent the virtual machines and their execution time. In this result, we have one user which sends 40 cloudlets (jobs/tasks) to the distributed system, the blue bar charts show the cloudlets performed on one virtual machine, and this expose the system to collapse, and this represents unbalancing load. The yellow bar chart represents the random load balancing algorithm and in this case it uses all available virtual machine (in this case, they are 11 virtual machines) and each of them has its own time. The green bar chats represent the round-robin algorithm for load balancing, the orange bar charts show the final execution for the user. At the first level of this scenario, it chooses the random algorithm as the best one for the execution, and we can note that it has changed the workload from virtual machines [vm0,vm1,vm6,vm7,vm9] to the others [vm2,vm3,vm4,vm5,vm8] at the second level, and finally it shows the difference between the random algorithm and the final execution.

More information about execution :

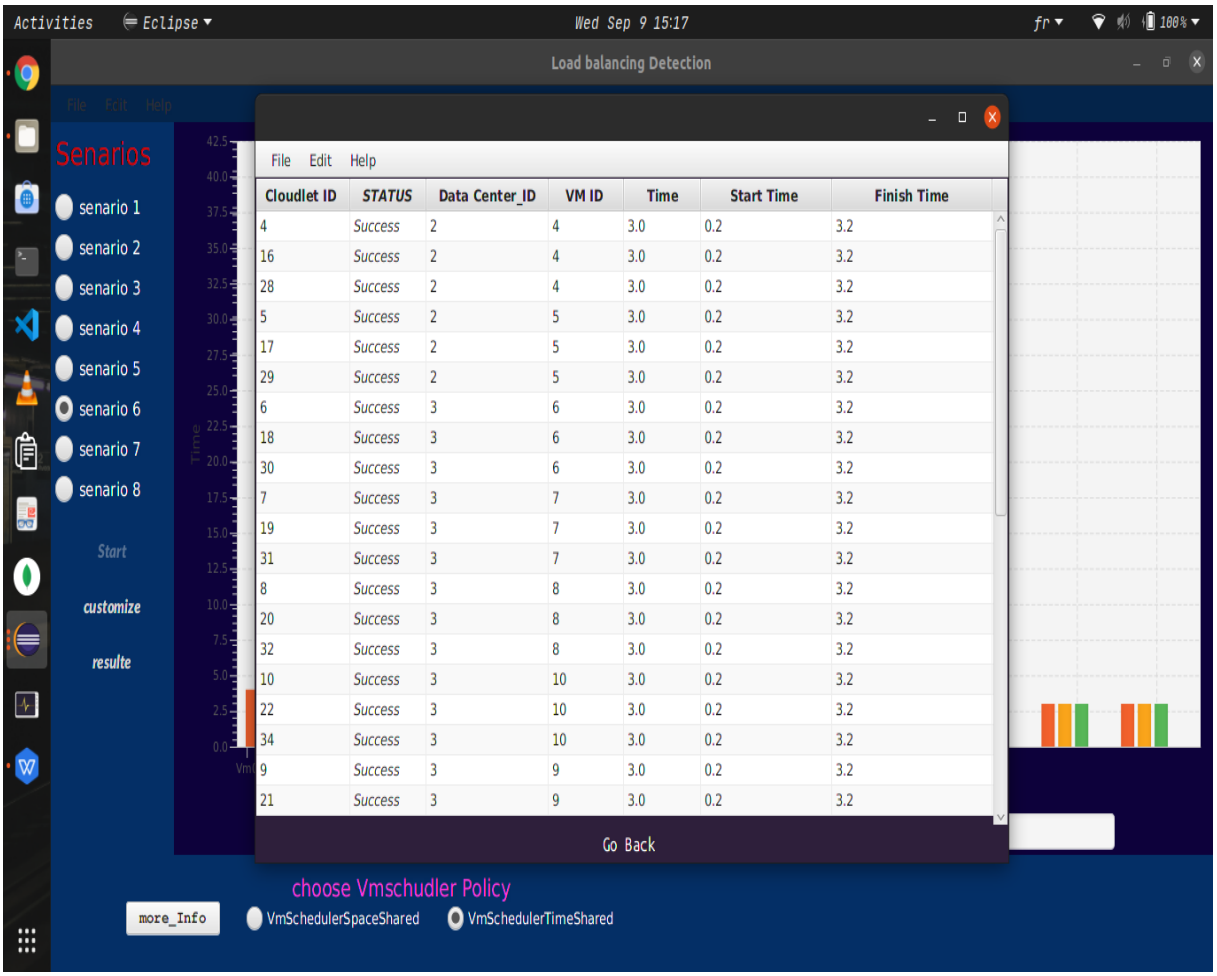


Figure 22 : more info interface

Before execution (only load balancer work):

```
 percent of killer averge is  :
58%

 end killer averge is  :

after filter ..........................................

we change : 1with : 0
```

Figure 23 : without our model result

With our model work with load balancer:

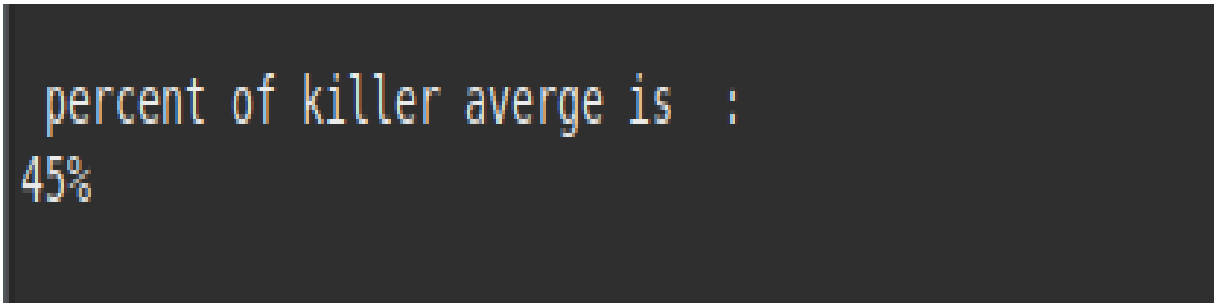


Figure 24 : result with our model work

5 Conclusion

In this chapter we have presented the results obtained as well as a quantitative comparison and qualitative. Our method is based on the pre-expectation and compares the results to reduce the workload on virtual machines, in fact our approach produces good results in terms of precision but we can do more in third level of our architecture to improve the quality of our product.

General Conclusion

General conclusion

Big companies offer their services over the Internet, but the issue here is load balancing. Growing The number of client requests to the server increases the load The system, which may be the cause of poor performance, which may also reduce the quality of service provided, so these companies need to find solutions to this problem, and here comes the role of the load balancing algorithm to provide continuous service without downtime or any depletion of service quality. This thesis provides a brief analysis of some load balancing algorithms for this problem with the main focus of load balancing. Different load balancing algorithms are compared on the basis of different types of parameters, but here we have focused on time and on status of virtual machine. We found that more improvements could be made on load balancing by do some trick like our proposed model and we encourage our self and our colleagues to create their own model to get more improve.

Bibliography

[1] : Tanenbaum, Andrew S.; Steen, Maarten van (2002).

Distributed systems: principles and paradigms

[2] : Godfrey, Bill (2002). "A primer on distributed computing"

[3] : Andrews (2000), p. 291–292. Dolev (2000), p. 5.

[4] : Ohlídal, M.; Jaroš, J.; Schwarz, J.; et al. (2006). "Evolutionary Design of OAB and AAB Communication Schedules for Interconnection Networks". In Rothlauf, F.; Branke, J.; Cagnoni, S. (eds.). Applications of Evolutionary Computing. Springer Science & Business Media. pp. 267–78.]

[5] : "Real Time And Distributed Computing Systems" (PDF).

ISSN 2278-0661. Retrieved 2017-01-09.

[6] : Vigna P, Casey MJ. The Age of Cryptocurrency: How Bitcoin and the Blockchain Are Challenging the Global Economic Order St. Martin's Press January 27, 2015 ISBN 9781250065636

[7] : Hieu., Vu, Quang (2010). Peer-to-peer computing : principles and applications. Lupu, Mihai., Ooi, Beng Chin, 1961-. Heidelberg: Springer. p. 16. ISBN 9783642035135. OCLC 663093862.

[8] : Lind P, Alm M (2006), "A database-centric virtual chemistry system", J Chem Inf Model, 46 (3): 1034–9, doi:10.1021/ci050360b, PMID 16711722.

[9] : Chiu, G (1990). "A model for optimal database allocation in distributed computing systems". Proceedings. IEEE INFOCOM'90: Ninth Annual Joint Conference of the IEEE Computer and Communications Societies.

[10] : Ghosh (2007), p. 10. Keidar (2008).

[11] : Lynch (1996), p. xix, 1–2. Peleg (2000), p. 1.

12 Peleg, David (2000), Distributed Computing: A Locality-Sensitive Approach,p.1.

[13] Ghosh, Sukumar (2007), Distributed Systems – An Algorithmic Approach, Chapman & Hall/CRC,p.10, ISBN 978-1-58488-564-1

[14] : Papadimitriou (1994), Chapter 15. Keidar (2008).

[15] : Bentaleb, A.; Yifan, L.; Xin, J.; et al. (2016). "Parallel and Distributed Algorithms" (PDF). National University of Singapore. Retrieved 20 July 2018.

[16] : The NIST Definition of Cloud Computing NIST

[17] : Wang (2012). "Enterprise cloud service architectures". Information Technology and Management. 13 (4): 445–454. doi:10.1007/s10799-012-0139-4. S2CID 8251298.

[18] : "What is Cloud Computing?". Amazon Web Services. 2013-03-19. Retrieved 2013-03-20.

[19] : Duan, Yucong; Fu, Guohua; Zhou, Nianjun; Sun, Xiaobing; Narendra, Nanjangud; Hu, Bo (2015). "Everything as a Service (XaaS) on the Cloud: Origins, Current and Future Trends". 2015 IEEE 8th International Conference on Cloud Computing. IEEE. pp. 621–628. doi:10.1109/CLOUD.2015.88. ISBN 978-1-4673-7287-9. S2CID 8201466.

[20] : Peter Mell; Timothy Grance (September 2011). The NIST Definition of Cloud Computing (Technical report). National Institute of Standards and Technology: U.S. Department of Commerce. doi:10.6028/NIST.SP.800-145. Special publication 800-145.

[21] : "ElasticHosts Blog". Elastichosts. 2014-04-01. Retrieved 2016-06-02.22 Amies, Alex; Sluiman, Harm; Tong, Qiang Guo; Liu, Guo Ning (July 2012). "Infrastructure as a Service Cloud Concepts". Developing and Hosting Applications on the Cloud. IBM Press. ISBN 978-0-13-306684-5.

[23] : Griffin, Ry'mone (2018-11-20). Internet Governance. Scientific e-Resources. p. 111. ISBN 978-1-83947-395-1.

[24] : Boniface, M.; et al. (2010). Platform-as-a-Service Architecture for Real-Time Quality of Service Management in Clouds. 5th International Conference on Internet and Web Applications and Services (ICIW). Barcelona, Spain: IEEE. pp. 155–160. doi:10.1109/ICIW.2010.91.

[24] : "Definition of: SaaS". PC Magazine Encyclopedia. Ziff Davis. Retrieved 14 May 2014.

[25] : Hamdaqa, Mohammad. A Reference Model for Developing Cloud Applications

[26] : "Self-Run Private Cloud Computing Solution – GovConnection". govconnection.com. 2014. Retrieved April 15, 2014.

[27] : "Private Clouds Take Shape – Services – Business services – Informationweek". 2012-09-09.

[28] : Haff, Gordon (2009-01-27). "Just don't call them private clouds"

[29] : "There's No Such Thing As A Private Cloud – Cloud-computing -". 2013-01-26

[30] : Rouse, Margaret. "What is public cloud?". Definition from Whatis.com. Retrieved 12 October 2014.

[31] : "Defining 'Cloud Services' and "Cloud Computing"". IDC. 2008-09-23

[32] : "FastConnect | Oracle Cloud Infrastructure". cloud.oracle.com. Retrieved 2017-11-15.

[33] : "What is hybrid cloud? - Definition from WhatIs.com". SearchCloudComputing. Retrieved 2019-08-10.

[34] : Butler, Brandon (2017-10-17). "What is hybrid cloud computing? The benefits of mixing private and public cloud services". Network World. Retrieved 2019-08-11.

[36] : "Mind the Gap: Here Comes Hybrid Cloud – Thomas Bittman". Thomas Bittman. Retrieved 22 April 2015.

[37] : "Business Intelligence Takes to Cloud for Small Businesses". CIO.com. 2014-06-04. Retrieved 2014-06-04.

[38] : Désiré Athow. "Hybrid cloud: is it right for your business?". TechRadar. Retrieved 22 April 2015.

[39] : Qiang, Li (2009). "Adaptive management of virtualized resources in cloud computing using feedback control". First International Conference on Information Science and Engineering.

[40] : Ali M. Alakeel, A Guide to Dynamic Load Balancing in Distributed Computer Systems, IJCSNS International Journal of Computer Science and Network Security,VOL.10 No.6, June 2010


Thank you !!!