إقـــــرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

An Approach for Bandwidth Guarantee in Multi-tenant Cloud Datacenter Network

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وإن هذه الرسالة ككل أو أي جزء منها لم يقدم من قبل لنيل درجة أو لقب علمي أو بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى.

DECLARATION

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification

Student's name:	اسم الطالب/ة: أيمن اسماعيل بارود
Signature:	التوقيع:
Date:	التاريخ: 1 / 02 / 2016

بسم الله الرحمن الرحيم

Islamic University – Gaza
Deanery of Graduate Studies
Faculty of Information Technology

الجامعة الإسلامية – غزة
عمادة الدراسات العليا
كلية تكنولوجيا المعلومات

An Approach for Bandwidth Guarantee in Multi-tenant Cloud Datacenter Network

A Thesis Submitted to the Faculty of Information Technology in Partial Fulfillments of the Requirements for the Degree of Master in Information Technology

By

Ayman Baroud

Supervisor

Dr. Rebhi Baraka

August 2015

الجامعة الإسلامية – غزة
The Islamic University - Gaza

مكتب نائب الرئيس للبحث العلمي والدراسات العليا
هاتف داخلي: 1150

الرقم ج س غ/35/ Ref

التاريخ 2015/09/13 Date

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة شئون البحث العلمي والدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ **أيمن اسماعيل خليل بارود** لنيل درجة الماجستير في **كلية *تكنولوجيا المعلومات*** **برنامج <u>تكنولوجيا المعلومات</u>** وموضوعها:

منهجية ضمان عرض النطاق في شبكات مراكز البيانات متعددة المستخدمين

An Approach for Bandwidth Guarantee in Multi-tenant Cloud Datacenter

وبعد المناقشة التي تمت اليوم الأحد 29 ذو القعدة 1436هـ، الموافق 2015/09/13م الساعة الحادية عشرة صباحاً، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

د. ربحي سليمان بركة	**مشرفاً و رئيساً**	....
د. توفيق سليمان برهوم	**مناقشاً داخلياً**	....
د. أحمد يحيى محمود	**مناقشاً خارجياً**	....

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في **كلية *تكنولوجيا المعلومات*/ برنامج تكنولوجيا المعلومات.**

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق،،،

نائب الرئيس لشئون البحث العلمي والدراسات العليا

أ.د. عبدالرؤوف علي المناعمة

ص.ب. 108 الرمال، غزة، فلسطين هاتف Tel: +970 (8) 286 0700 فاكس fax: +970 (8) 286 0800 P.O. Box 108, Rimal, Gaza, Palestine
public@iugaza.edu.ps www.iugaza.edu.ps

Table of Contents

List of Tables

List of Figures

Abbreviations and acronyms

IaaS	Infrastructure-as- a-Services
PaaS	Platform as a Services
SaaS	Software as a Services
SLA	Service Level Agreements
TCP	Transport Control Protocol
UDP	User Datagram Protocol
VM	Virtual Machine
SDN	Software Defined Network
DC	Data Center
DCN	Data Center Network
WAN	Wide Area Network
TAG	Tenant Application Graph
VMM	Virtual Machine Monitors
QoS	Quality of Service
Amazon EC2	Amazon Elastic Computing Cloud
VDC	Virtual Data Center
VN	Virtual network
SN	Substrate Network
InP	Infrastructure Provider
SP	Service Provider
TOR Switch	Top Of Rack Switch
PM	Physical Machine

عنوان البحث: منهجية ضمان عرض النطاق في شبكة مركز البيانات السحابية متعددة المستخدمين

ملخص

ان التطور المتسارع في البنية التحتية للحوسبة السحابية، سمح للشركات المالكة لهذه البنى بأن يعرضوا هذه القدرات الحاسوبية مثل المعالج المركزي والذاكرة والمساحة التخزينية للإجارة العامة، الا ان هذه الاجارة كانت وما زالت تفتقد تأمين خدمات الشبكات ضمن عقود الاجارة. وذلك كون مالكي الخدمة ينظرون الى خدمة الشبكات كمصادر عامة من الصعب ان يتم استغلالها من قبل جميع المستأجرين في نفس الوقت. على ان كثير من الباحثين اثبتوا ان هذه الفرضية غير صحيحة، وان هناك إمكانية لان يؤثر مستأجر على اخر نتيجة الصراع على الاستفادة من مصادر الشبكة. او ان قام أحدهما باستغلال هذه المصادر بشكل تعسفي. الامر الذي يؤدي في النهاية الى ان البرامج التطبيقية الخاصة بمستخدم ما لا تعمل بالكفاءة المتوقعة. ان هذه الظاهرة أدت الى عزوف بعض المستأجرين عن التوجه الى الحوسبة السحابية كونها لا تضمن توفير كفاءة متوقعة لبرامجهم التطبيقية، هذا من ناحية المستأجرين اما من ناحية المؤجرين فان هذه الظاهرة أدت الى انخفاض العائد من استثمارهم.

ان تشغيل البرامج التطبيقية ضمن الحوسبة السحابية وبالكفاءة المطلوبة يستلزم ان يقوم مزود خدمات الحوسبة السحابية بتوفير كل من القدرات الحاسوبية ومصادرة الشبكة معا. ان المنهجية الجديدة التي تدمج كل من القدرات الحاسوبية ومصادر الشبكة وتقدمها للمستأجر كحزمة واحدة يطلق عليها مراكز البيانات الافتراضية. في هذه المنهجية يقوم مزرد الحوسبة السحابية بتوفير واجهة بينية يقوم من خلالها المستأجر بتحديد عرض النطاق الذي يحتاجه بشكل صريح بالإضافة الى عدد الأجهزة الافتراضية المطلوبة ومواصفاتها. ويضمن مزود الخدمة توفير عرض النطاق المطلوب من خلال الاختيار المناسب لمكان تركيب الأجهزة الافتراضية بالإضافة الى استخدام محددات عرض النطاق. انه من الضروري هنا الإشارة الى عملية توزيع طلبات المستأجرين من أجهزة تخيلية وعرض نطاق على مراكز البيانات الفيزيائية هي توسيع لمشكله توزيع البضائع على الحاويات، والتي تعتبر من المشاكل التي من الصعب إيجاد حل لها في وقت معقول.

هذه الرسالة هي مساهمة في العمل القائم لحل مشكلة مشاركة عرض النطاق في مراكز البيانات المتعددة المستخدمين. حيث نقدم فيها خوارزمية مكونة من مرحلتين. في المرحلة الأولى يتم معالجة طلبات المستأجرين بشكل أولى عن طريق دمج ما يمكن دمجه من أجهزة تخيلية وعرض نطاق. في المرحلة الثانية من الخوارزمية. يتم ربط طلب المستأجر من أجهزة افتراضية وعرض النطاق بمركز بالخادمات الحقيقية في مركز البيانات. قمنا بتقييم عملنا هذا من خلال استخدام برامج المحاكاة للحوسبة السحابية ومقارنة النتائج مع اعمال أخرى في نفس المجال. وجدنا ان عملنا زاد من استغلال مصادر الشبكة في نفس الوقت وفر نوع من الحماية للمستأجر في مقابلة تعطل الخادمات.

كلمات افتتاحية: الحوسبة السحابية، البنية التحتية كخدمة، ضمان عرض النطاق، مراكز البيانات الافتراضية.

Abstract

The rapid development in cloud infrastructure allows cloud providers to offer computing power such as CPU, memory and storage to different customers, and serve multiple tenants at the same time. Unfortunately, due to network nature, cloud provider give no guarantee of network resource allocation to individual VM, relying on the fact that VMs are unlikely to simultaneously maximize their use of their nominally assigned bandwidth. However, many research shows that there is potential for VM to effect each other due to contention on network resource, or as side effect of noisy neighbor, and varying of bandwidth demand across time. Resulting in a severe lack of predictability and fairness of application performances. That cost tenants unpredictable cost and provider loss revenue.

Running cloud application without deprecation of performance require cloud provider to offer both computation power and network resource as bundle. The new approach that combine both computing and network resource in one form called virtual data center (VDC), formally known as VDC embedding problem. In this approach, cloud provider should provide tenants with an interface to explicit specifying their bandwidth demand. Cloud provider achieve deterministic bandwidth guarantee by proper placement of tenant VMs and static enforcement of rate limit. However, offering such service to customer is not easy task, and many researchers classify this problem as multiple dimension ben packing problem, which been as NP hard problem.

This research contribute to work done in this area. We introduce two stage heuristic algorithms. The first stage, preprocess tenant VDC request by consolidating VMs and link capacity. The second stage, we map tenants request to physical datacenter server and links. We evaluate our approach against other approach using cloud computing simulation tools, and find that our approach increase the utilization of network, while maximizing the number of accepting VDC request, in addition to provide some level of reliability to tenant request against server failure.

Keywords- *bandwidth guarantee; cloud computing; infrastructure-as-a-services; performance isolation; resource allocation; virtual data center; virtual network.*

Chapter 1 : Introduction

Cloud computing, the new paradigm in which computing is delivered as a service rather than a product, and where computing resources (i.e., networks, servers, storage, applications, etc.) are provisioned as metered on-demand services over networks, and can be rapidly allocated and released with minimal management effort. Cloud computing is expected to provide many benefits for consumers as well as for providers. Many companies such as Amazon, Google, Facebook, and Yahoo deploy this model of computing to support large-scale applications and to store large volumes of data using their data center to provide computing as utility.

One key enabling technology of cloud computing is the data center visualization, this technology enables creation of multiple virtual machines (VM) on one physical server, which in turn increases server utilization and reduces operation cost. Moreover, virtualization technologies provide performance isolation between collocated VMs to improve application performance and security. Nonetheless, server virtualization alone is insufficient to address all of the limitations of today's data center architectures. In particular, datacenter networking issues.

The performance of running user applications on cloud platform suffers unpredictable network performance, compared to the performance of running the same application on isolated environment, due to the impact of network congestion on many application run on cloud, i.e. guarantee on bandwidth [1][2][3]. should mention that, the bandwidth achieved between tenant VM depend on many factors outside control of tenant itself, such as network load, placement of tenant VM, and the natural oversubscription of cloud provider network [4]. As there is several commercial cloud providers (e.g., Amazon EC2) cannot provide the performance guarantee of network resource for tenants (or VMs).

Research shows that network bandwidth and latency between two VMs can vary significantly over time [4]. Which, in turn, has several negative consequences for both tenants and providers. Figure 1.1 illustrate the difficulty of sharing bandwidth between different tenants. For example, communication flow between X1 and X2 are interfered by communication flow from other tenants Y1, Y2 and Z1, Z2.

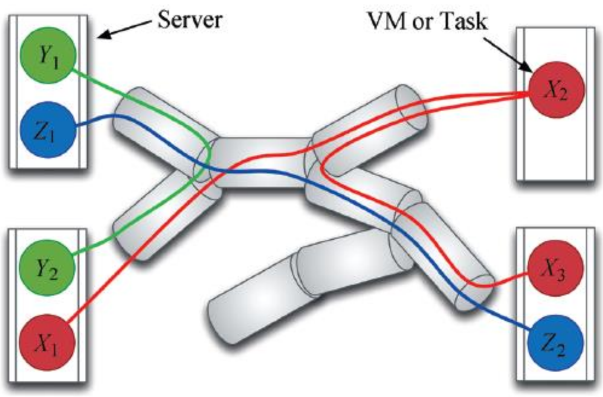


Figure 1.1: Bandwidth sharing among multiple tenant (X, Y, and Z) (source [5])

To overcome the performance degradation of cloud applications and deployed services due to bandwidth sharing problem. Recent research proposals have advocated offering both computing and networking resources in the form of Virtual Data Centers (VDCs). A VDC consists of virtual machines, routers and switches connected through virtual links with guaranteed bandwidth. Figure 1.2 depicts the idea of VDC and its mapping to physical data center.

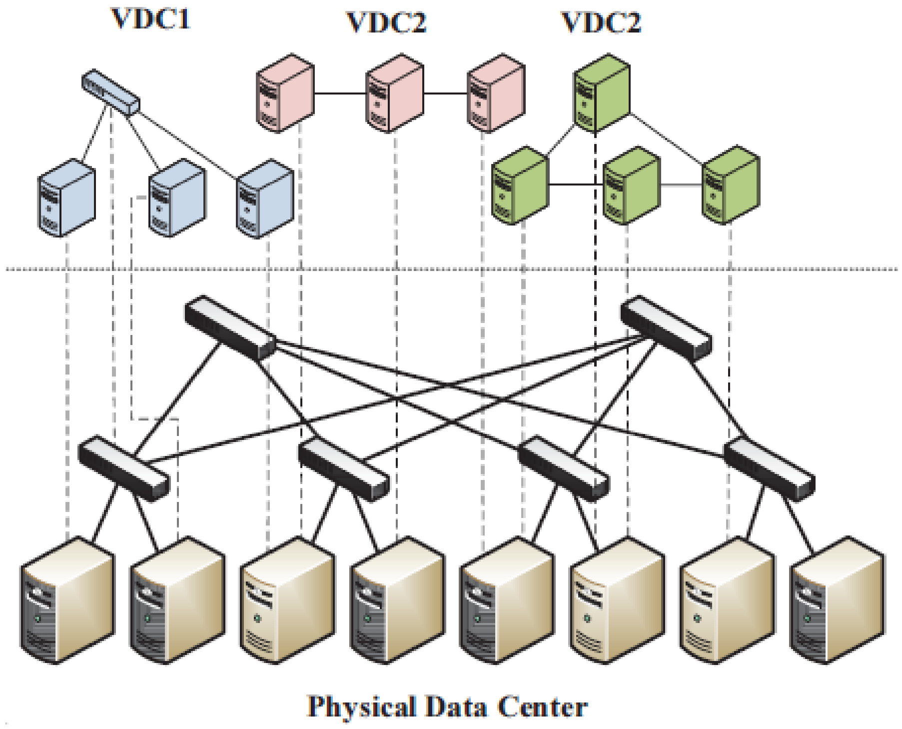


Figure 1.2: Multiple VDC mapping to same physical data center (source [6])

VDC aims for not only mapping of VM with its own properties (e.g., CPU, memory and storage), but also include mapping virtual switch and links to physical switch and link. This problem called VDC embedding problem. Which been known as NP-hard problem [7] [4] [8] [9]. Many solution exist [10] [4] [6] [11] [12] [13], both exact and heuristic embedding algorithms been proposed, each of which have its own pros and cons (refer to chapter three for more in deep comparison between different solution). However, finding a solution that can cope with complex tenant request and provide simple embedding algorithm is challenging task.

In this research, we propose an approach to solve the bandwidth-sharing problem based on static reservation. Our approach allows tenants to express their needs of both computing power and required network bandwidth as network graph, which in turn can be transformed into adjacency matrix. In the rest of this chapter, we explore and define the research problem. Then set the objective that our research try to achieve, and the limitation that define boundary of our research. In Section 1. We describe the methodology we use to achieve our objectives. Then we describe our contribution in this research. Finally, we give general description of whole structure of the thesis.

1.1 Statement of the Problem

Cloud computing paradigm brings noticeable performance degradation and variation of VMs performance to tenants, due to contention on network resource, which has become one of the primary issue of IaaS cloud. The new trend to solve this problem is by reserving the bandwidth alongside with computation resource. That is giving the tenants the illusion of having his own virtual data center (VDC). In this context, solving the bandwidth problem through implementation of VDC introduce an optimization problem that have many objective, our work try to solve the VDC problem, with constrain on bandwidth, reliability, and increasing cloud provider revenue and tenant satisfaction.

The problem can be decompose into the following sub problems:

1. How to simplify tenant VDC request, in such way that make embedding VDC request more easy.
2. How to maximize the number of accepted VDC request that share the same physical machine and network.
3. How to provide predictable network performance to tenant (bandwidth), in face of varying traffic pattern, varying of bandwidth demand across time, and noisy neighbor tenants?
4. How to enable tenant to describe his needs of computing resource, and network topology, as he would do if he build his own data center.
5. How to map tenant request of VM and network bandwidth to the datacenter physical infrastructure?

1.2 Objectives

1.2.1 Main Objective

To propose a new approach to embedding tenant VDC request into cloud provider data center that maximize the accepted VDC request, without violating tenant request of bandwidth, same time minimizing the total bandwidth consumption.

1.2.2 Specific Objectives

The specific objectives of the thesis are:

- Enable tenants to express his needs of virtual machine, network topology, and the bandwidth requirement between each two virtual machine. Using simple adjacency matrix.
- Simplify VDC request, in such way that make it easier to be embedded, minimize bandwidth consumption.
- Mapping of the tenant request –in the form of virtual data center VDC- components, which include virtual machine and link between them to physical data center infrastructure.

- Evaluate the proposed approach using network simulation tools based on some performance metrics, such as acceptance ratio, bandwidth consumption rate. and compare this to other similar approaches.

1.3 Importance of the Thesis

Conducting this study is important for both cloud computing provider and tenants. It allows cloud provider to maximize the number of accepted VDC request. On the other side, tenant will benefit by running their application without performance degradation due to bandwidth contention problem.

Maximizing the accepted tenant request on the same physical data center imply that network and physical server utilization will be improved, this has two main advantage: first, it minimize the operational cost by eliminating running unnecessary hardware if we can serve tenant with current hardware. Second, this minimize power consumption.

Our work introduce and prove that using simple heuristic algorithm for simplifying tenants VDC request, could be used to improve the performance of general VM placement algorithms

1.4 Scope and Limitations of the Thesis

- The scope of our solution will be limited to solve the problem of bandwidth contention on network within single data center (DC), for wide area networks (WAN) connecting data centers in different physical locations are not included and out of our scope.
- Our approach not include any bandwidth enforcement technique. Our solution provide static reservation of bandwidth at request initialization time, enforcing such reservation could be done at different level such as hypervisor level or switch level, but not include in our approaches.
- Our scope of this work is limited to Infrastructure as a Service (IaaS), other cloud computing model, like Software as a Service (SaaS) and Platform as a Service (PaaS) are not included.
- Our solution should not be limited to specific protocol, e.g. TCP.
- Our concern in this work is bandwidth, other network metrics “delay, jitter” are not included.

1.5 Methodology

Our methodology to achieve the objectives and hence solve the problem starts by enabling the tenants to express his needs of both computation resource alongside with required bandwidth using undirected weighted graph, where each vertex represents VM and edge represents bandwidth demand between linked VMs. The undirected weighted graph, which depict tenant request, represent our network abstraction model.

The tenant request -represented by adjacency matrix- should be simplified with a consolidation algorithm, and then tenant VDC request should be mapped to data center physical machine and physical links. We explain the methodology in the following stages:

Stage one: Simplify tenant VDC request. VM/link consolidation

By using heuristic algorithms, cloud provider read VDC request as an adjacency matrix, then transform it to simple graph. This is mandatory step that we based on to simplify the embedding.

Stage two: VM/Link placement

Design allocation algorithm to map tenant requests of both computing resource and network link to data center infrastructure. We think of the datacenter network as tree, - graph- where VM are the leaves and weighted edge to denote available bandwidth on this link. – This assumption is subject to change based on our study progress, - so the algorithm try to find where is the best place to allocate tenant request.

Stage three: Evaluation

As a Proof of concept, we use simulation tools to evaluate our solution on large-scale network. Evaluation will focus on performance of allocation algorithm. As baseline for evaluation, we use Random algorithm, OLRSA algorithm and SAE algorithm.

1.6 Contributions

- We present a comprehensive background about data centers, the basic concepts of data center virtualization. We also discuss the business model associated in cloud environments. We also provide an extensive literature survey on VDC embedding solution.
- We introduce new VDC embedding solution, with reasonable reliability objective.
- Introduce the idea of VDC request pre-processing, with both VMs and links consolidation.

1.7 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2: Theoretical and Technical Foundation, starts by defining cloud computing, cloud computing characteristic, service delivery models, and the deployment mode. Next provides an overview of conventional data centers, the basic concepts of data center virtualization, network virtualization, next we introduce the subject of scheduling and resource allocation in cloud computing enlivenment, the final section will stand for simulation tools, where we introduce FlexCloud toolkit.

Chapter 3: Related Works, we start by introducing the problem domain and distinguishing it from virtual network embedding domain, and then we introduce and analyze key research paper on our research domain.

Chapter 4: An Approach for Bandwidth Guarantee in Multi-tenant Data Center is dedicate for our own approach to solve the problem of sharing bandwidth in cloud data center. This chapter includes the mathematical formulation and algorithms used to simplify tenant VDC request and mapping of virtual resource to physical data center.

Chapter 5: Simulation and Performance Evaluation, we define the simulation environment and performance metrics. Then we introduce our results, analyze it and finally compare our work with others.

Chapter 6: A Conclusion and Future Work, we write down our study conclusion and point out some direction for future work.

Chapter 2 : Theoretical and Technical Foundation

The aim of this chapter is to present the theoretical and technical foundation as well as the terminology relevant to our research. We first provide some definitions related to cloud computing including its service and deployment model in Section 2.1. Section 2.2, introduces some key enabling technologies, that include virtualization, the data center, as it is the cloud computing platform and networking issue related to data center. In Section 2.3, we introduce basic concept related to scheduling and resource allocation. Finally, Section 2.4 is dedicated to simulation tools particularly FlexCloud toolkit.

2.1 Cloud Computing

2.1.1 Definition and Characteristics

There are many definitions about what cloud computing is, and it seems that experts from the industry, organizations and institutions all have their own understandings. Nevertheless, we strict our self to what been defined by the National Institute of Standards and Technology (NIST), quoted below:

- "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." - U.S. National Institute of Standards and Technology (NIST) [14].

Cloud computing make fundamental change to how computing services are developed, deployed, maintained and delivered. It also called the fifth generation of computing [15]. Next we introduce some qualitative and economical feature that cloud computing provide. Qualitative features refer to the qualities or properties of cloud computing, rather than specific technological requirements, while economic features are the feature that make cloud computing distinct compared with other computing paradigms form the business point of view:

- **Elasticity** means that the provision of services is elastic and adaptable, which allows the users to request the service near real-time without engineering for peak loads. The services are measured in fine-grain, so that the amount of offering can perfectly match the consumer's usage.
- ***High Availability.*** Making application highly available is usually difficult and expensive for companies, which requires specialized tools and highly trained staff. In a well-designed cloud computing, the more professional experts from cloud service providers can assist to improve reliability by for example simply running multiple redundant sites. By hiding many of the underlying complexities of disaster recovery, cloud computing can significantly reduce the burden of companies to maintain business continuity
- **Reliability** represents the ability to ensure constant system operation without disruption. Through using the redundant sites, the possibility of losing data and code dramatically decreases. Thus cloud computing is suitable for

business continuity and disaster recovery. Reliability is a particular QoS requirement, focusing on prevention of loss.

- **Pay-as-you-go** is the means of payment of cloud computing, only paying for the actual consumption of resource. Traditionally, users have to equip with all software and hardware infrastructure before computing starts, and maintain them during computing process. Cloud computing reduces cost of infrastructure maintenance and acquisition, so it can help enterprises, especially small to medium sized, reduce time to market and get return on the investment.
- **Operational expenditure** is greatly reduced and converted to operational expenditure. Cloud users enter the computing world more easily, and they can rent the infrastructure for infrequent intensive computing tasks. Minimal technical skills are required for implementation. Pricing on a utility computing basis is fine-grained with usage based options, so cloud providers should mask this pricing granularity with long-term, fixed price agreements considering the customer's convenience.
- **Energy-efficiency** is due to the ability that a cloud has to reduce the consumption of unused resources. Because of central administration, additional costs of energy consumption as well as carbon emission can be better controlled than in uncooperative cases. In addition, green IT issues are subject to both software stack and hardware level.

2.1.2 Cloud Computing Service Delivery Models and Stakeholder

Cloud computing provider rent out their IT resource to consumer in different ways and scenarios, commonly named, service delivery model. The most common and famous service delivery model enumerated below and depicted in Figure 2.1:

- ***Infrastructure as a Service (IaaS):*** provides processing, storage, networks, and other fundamental computing resources to users. IaaS users can deploy arbitrary application, software, operating systems on the infrastructure, the most appealing benefit is that users can dynamically increase or decrease several aspects of the environment in an on-demand fashion, along with business needs fluctuate. IaaS user sends programs and related data, while the vendor's computer does the computation processing and returns the result. The infrastructure is virtualized, flexible, scalable and manageable to meet user requirements. Examples of IaaS include Amazon EC2 [16] , Eucalyptus [17], and Rackspace [18] Cloud, etc.

- ***Platform as a Service (PaaS):*** offers a high-level integrated environment to build, test, deploy and host customer-created or acquired applications. Generally, developers accept some restrictions on the type of software that can write in exchange for built-in application scalability. Customers of PaaS do not manage the underlying infrastructure as SaaS users do, but control over the deployed applications and their hosting environment configurations. PaaS offerings mainly aim at facilitating application development and related management issues. Some are intended to provide a generalized development environment, and some only provide hosting-level services such as security and on-demand scalability. Typical examples of PaaS are Google App Engine [19], Windows Azure [20].

- ***Software as a Service (SaaS):*** Is a software delivery model in which applications are accessed by using a thin client (normally a web browser). The users are not concerned with the underlying cloud infrastructure including network, servers, operating systems, storage, platform, etc. this is because SaaS hosts software and the associated data centrally. SaaS has now become a common delivery model for most business applications; Enterprise applications include human resource management (HRM), enterprise resource planning (ERP), customer relationship management (CRM), accounting and invoicing. The popularity can also been seen from personal applications such as online documents (Google Docs [21]), and storage service (Dropbox [22]).

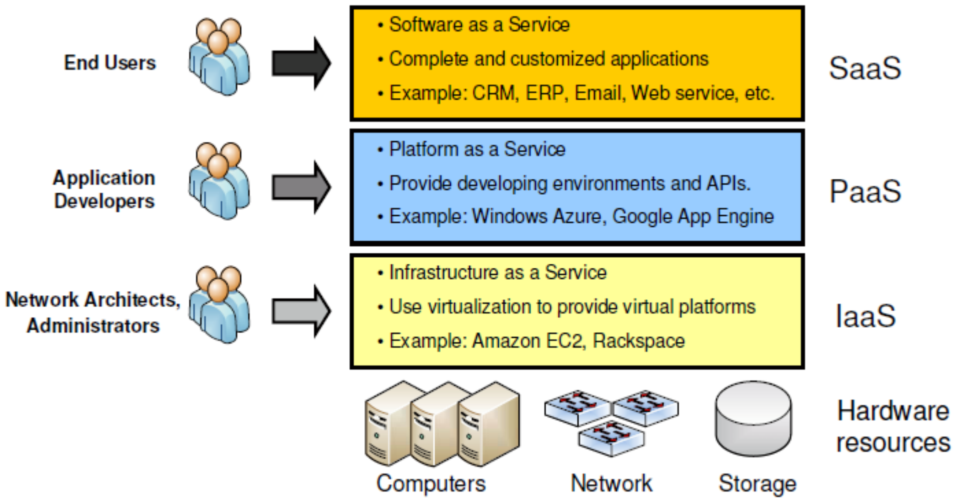


Figure 2.1: Cloud computing service delivery models.

Traditionally, ISP network and cloud computing environment define two roles, ISP or cloud provider roles, and end-user roles. Although, some researchers proposes separate the role of ISP / cloud provider into two different rules [10][23][7]. Below we define the each roles

- ***Infrastructure Providers (InPs):*** provision infrastructure resources such as virtual instances, networks, and storage to consumers usually by utilizing hardware virtualization technologies. In the IaaS model, a consumer rents resources from an infrastructure provider or multiple infrastructure providers, and establishes its own virtualized infrastructure, instead of maintaining an infrastructure with dedicated hardware. There are numerous infrastructure providers on the market, such as Amazon Elastic Compute Cloud (EC2) [3], and Rackspace.

- ***Service Providers (SPs)***: use either their own resources (taking both the SP and InP roles) or resources leased from one or multiple InPs to deliver end-user services to their consumers. SPs are not in charge of maintaining the underlying hardware infrastructures. SPs can use performance metrics (e.g., response time) to optimize their applications by scaling their rented resources from InPs, providing required Quality of Service (QoS) to the end users.

- ***Cloud End Users:*** who are the consumers of the services offered by SPs and usually have no concerns on where and how the services are hosted.

2.1.3 Type of Clouds

Every organization has different reasons to move their IT infrastructure to a cloud environment. While some clients might be looking to cut down on their day to day IT operational costs, others might like to project high quality and security as their unique selling points. To cater to these diverse demands and requirements, there exist four different kinds of cloud setup as described below [24][15][25]:

- ***Private cloud:*** The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise. A private cloud offers the highest degree of control over performance, reliability and security.
- ***Community cloud:*** The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise.
- ***Public cloud**:* The cloud infrastructure is made available to the public or a large industry group and is owned by an organization selling cloud services. Public clouds offer several key benefits to service providers, including no initial capital investment on infrastructure and shifting of risks to infrastructure providers.
- ***Hybrid cloud**:* The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability. In a hybrid cloud, part of the service infrastructure runs in private clouds while the remaining part runs in public clouds. This make hybrid cloud more flexible than public and private cloud.

2.1.4 Cloud Infrastructure Management Systems

Cloud computing services will net be available without cloud management system, such platforms are available in both proprietary and open source free forms, The proprietary cloud vendors deliver public cloud services as a pay-per-use model. Such as Amazon EC2 and Windows Azure. Open source cloud computing management system include Eucalyptus, OpenStack, OpenNebula, and Nimbus. These Cloud solutions provide various aspects of Cloud infrastructure management such as [26]:

1. Management services for VM life cycle, compute resources, networking, and scalability.
2. Distributed and consistent data storage with built-in redundancy, failsafe mechanisms, and scalability.
3. Discovery, registration, and delivery services for virtual disk images with support of different image formats (VDI, VHD, qcow2, VMDK).
4. User authentication and authorization services for all components of Cloud management.
5. Web and console-based user interface for managing instances, images, cryptographic keys, volume attachment/detachment to instances, and similar functions.

2.2 Key Enabling Technology

2.2.1 Virtualization Technology

"Virtualization, in computing, refers to the act of creating a virtual (rather than actual) version of something, including but not limited to a virtual computer hardware platform, operating system (OS), storage device, or computer network resources." [27]. Virtualization consider as main enabling technologies that paved the way of Cloud Computing towards its extreme success [26], and the underlying technology that cloud computing stand on [28].

Data center virtualization has three main components [29]:

- ***Server virtualization*** Server or machine virtualization abstracts the OS from the physical hardware. This has the benefit of running one or more OSes on one physical server, which increases the utilization of the hardware. This reduces the number of physical servers needed, which also reduces the power and cooling requirements of the data center.

- ***I/O virtualization:*** I/O virtualization abstracts the data flows paths from the physical network connections. By providing large physical links to each device in the data center and then defining the number, types, and capacity of the data paths in software, all aspects of I/O can be reconfigured without physically moving cables.

- ***Storage virtualization:*** Storage virtualization abstracts the data that the OS sees from the physical disk. This provides the ability to manage the data in different ways without the involvement of the OS. Storage virtualization, for example, can relocate data to another physical disk without participation of the OS.

We can summarize the benefits of virtualization technology as follow [30]:

- Implementing multiple VM in single machine (aka server consolidation), allow better utilization of hardware, and reduce physical space consumption and hardware cost.

- VM is a collection of files; Like all other files it can be moved or copied from one place to another while it is running, (live migration) that enables load balancing, gives higher availability ,makes maintenance and manageability easier.

- Increase the security level by isolating different application into different VM. Therefore, malfunction or malicious software of one VM does not affect the others. In short, virtualization provide high-level isolation among VM.

On the other hand, Virtualization, like other systems, has its own limitations, this is include magnifying physical failures, increased system complexity, introduces inefficiencies and weakens raw performance [30].

2.2.2 Server Virtualization

In this Section, we focus into server virtualization. Based on architectural perspective, server virtualization could be categorize as follows

1- ***Type I hypervisor***, *(also known as,* Hypervisor-based virtualization, Hypervisor system). Hypervisor is a piece of software that hosts and manage VMs on its *Virtual Machine Monitor* (VMM) components, it is installed and run on bare hardware and retains full control of the underlying physical system. The VMM partitions and shares the CPU, memory, and I/O devices to successfully virtualize the underlying physical system by multiplexes the hardware resources among the various running VMs in time and space sharing manner [26], Figure 2.2, depicted type I hypervisor virtualization.

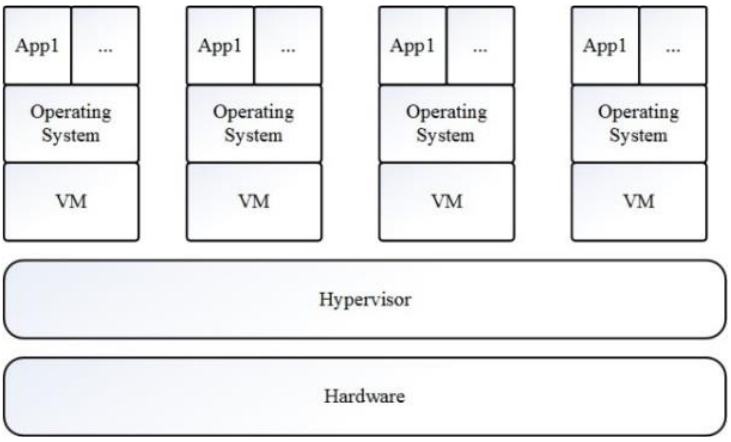


Figure 2.2: Type I hypervisor

2- ***Type II hypervisor***, (also known as Container-based virtualization, OS virtualization, Hosted system virtualization, Or application virtualization). The virtualization layer is installed and run as an individual application on top of an operating system and supports the broadest range of underlying hardware configurations. Example of such architecture includes VMware Player, and Oracle VM VirtualBox. However, since the virtualized execution is achieved heavily through emulation, it suffers great performance degradation especially for I/O. Type II hypervisors are mostly used in client environments (PCs), where performance is less critical. They allow users to gain rich functionalities by running several different types of operating systems like Linux, Windows and Mac. Figure 2.3 depicted type II hypervisor.

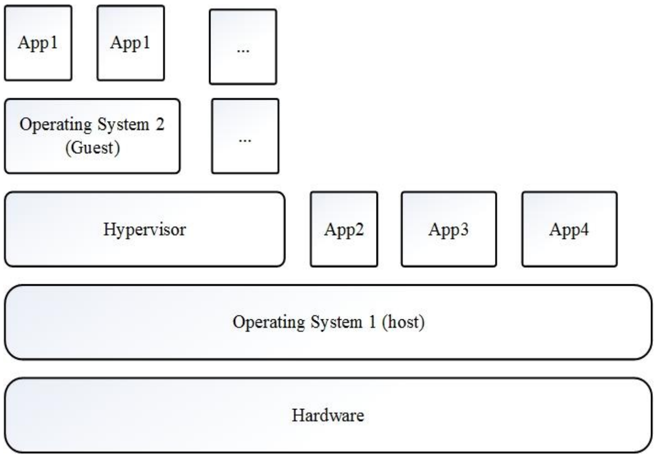


Figure 2.3: Type II hypervisor

2.2.3 X86 Architecture Virtualization

The functionality of the hypervisor varies greatly based on architecture and implementation [31]. We consider virtualization of Intel x86 architecture as it been the most successfully and widely adopted architecture. The Intel x86-instruction set consists of three kinds of instructions namely privileged, sensitive and non-privileged instructions. To protect the valuable resources like CPU, memory, input and output devices from unauthorized access, x86 uses four modes of operation (protection mechanism) numbered from '0'(privilege level/root level) to '3' (least privilege level/user mode), referred as ring. The ring '0' is normally used by OS, user application is run in ring '3' ,and the remaining are rarely used in commodity OS. See Figure 2.4.

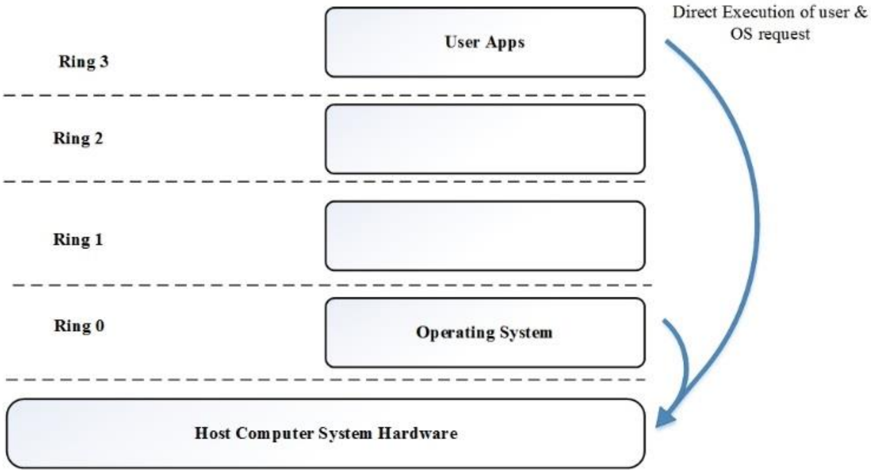


Figure 2.4: The x86 processor privilege rings without virtualization.

Hypervisor Virtualization supposed to be installed and run on bare hardware and bellow operating system so that VMs can be created and managed that would share the same physical resources. This means the virtualization layer needs to be placed in Ring 0; however unmodified operating systems assumes to be run in the same Ring. Moreover, there are some sensitive instructions that have different semantics when they are not executed in Ring 0 and thus cannot be effectively virtualized. As a consequence, the industry and research community have come up with the following three types of alternative virtualization techniques:

1- ***Full Virtualization:*** This type of virtualization technique provides full abstraction of the underlying hardware and facilitates the creation of complete VMs in which guest operating systems can execute. Full virtualization is achieved through a combination of binary translation and direct execution techniques that allow the VMM to run in Ring 0. VMware ESX Server and Microsoft Virtual Server are examples of full virtualization.

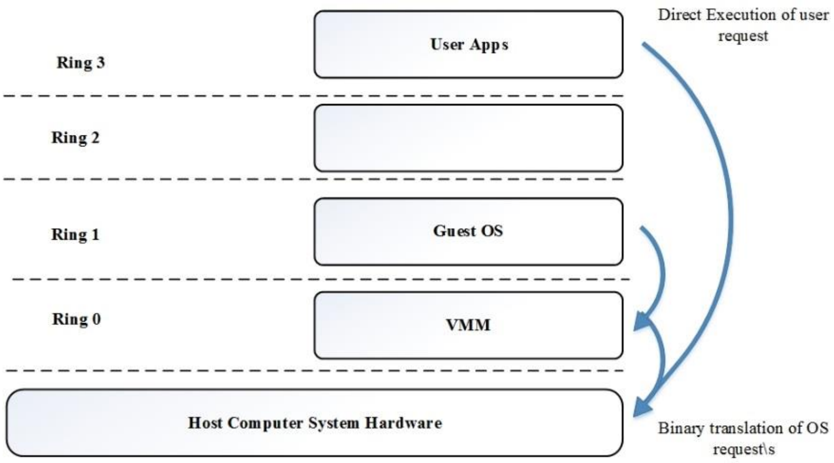


Figure 2.5: Full Virtualization

2- ***Paravirtualization:*** (also called *OS Assisted Virtualization*) works through the modification of the OS kernel code by replacement of the non-virtualizable instructions with hypercalls that communicate directly with the hypervisor virtualization layer. Thus, in paravirtualization each VM is presented with an abstraction of the hardware that is similar but not identical to the underlying physical machine. Example of paravirtualization is the open source Xen project

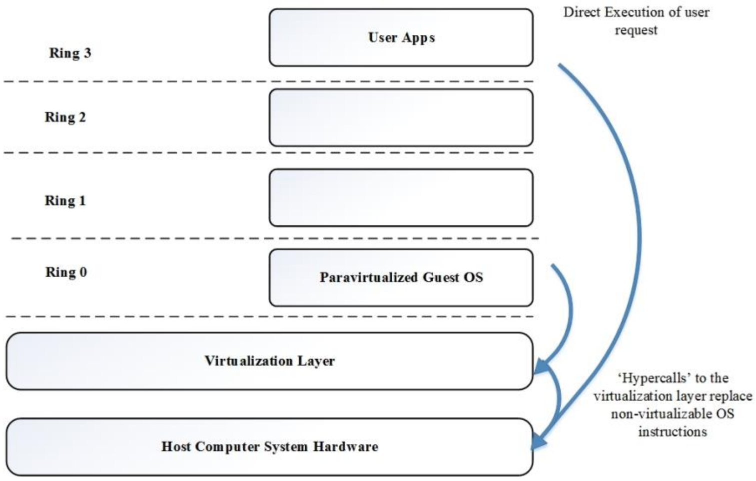


Figure 2.6: Paravirtualization

3- ***Hardware Assisted Virtualization:*** *H*ardware vendors have come up with new hardware features to help and simplify virtualization techniques. Intel Virtualization Technology (VT-x) and AMD-V are first generation virtualization supports allow the VMM to run in a new root mode below Ring 0 by the introduction of a new CPU execution mode. With this new hardware assisted feature, privileged and critical system calls are automatically trapped by the hypervisor and the guest OS state is saved in Virtual Machine Control Structures (VT-x) or Virtual Machine Control Blocks (AMDV),

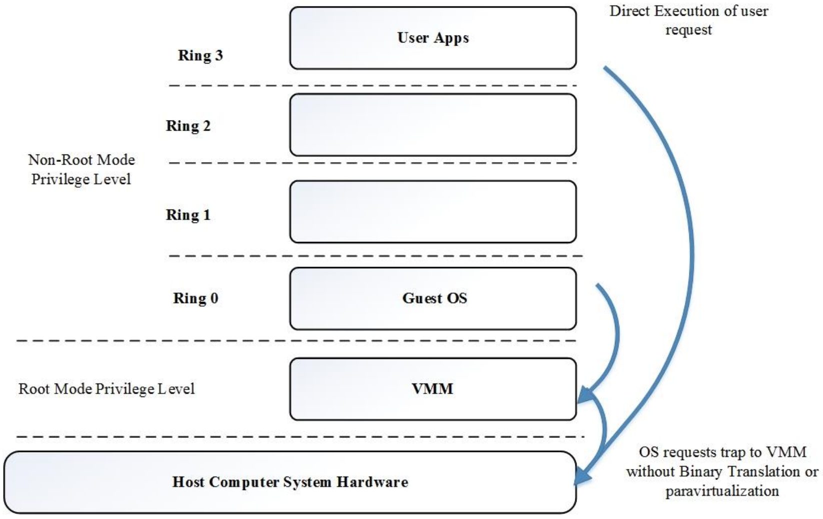


Figure 2.7: Hardware Assisted Virtualization

2.2.4 Datacenter Network

In recent years, datacenters (DC) become the foundations and the platform of cloud computing. DC have emerged as the cornerstones of modern communication and computing infrastructure. DC is a pool of computing resources clustered together using communication networks to host applications and store data [32]. DC consist of server (physical host), storage, network device, and power distributed system. Data Center network (DCN) is the communication infrastructure in DC and is described by network topology, routing/switch equipment, and the used protocol [33].

The major network fabrics for implementing data center are Ethernet and InfiniBand (IBA) [34]. Our primarily focus on data center network is based on Ethernet implementation. In such implementation, nodes can be configured to operate in Ethernet-switched mode or IP-routed mode. Nodes addressing schema could be ether flat or hierarchical, flat addressing use the Ethernet 6-bye MAC address, where an interface can be assigned any address (typically by the manufacturer) without consideration of its topological location (i.e., the switch/router to which it is connected). Hierarchical addressing are using internet protocol (IP). Which means that address is assigned to node by administrator based on nodes topological location. Both addressing schemes have its limitation, solutions should be hybrid approaches combining both flat and hierarchical addressing schemes [34].

In this section, we choose to primarily concentrate on a representative Standard tree based architectures and their variants are widely used in designing data center networks namely: basic tree topology and Fat tree topology.

1) ***Basic Tree:*** Basic tree topologies consist of three levels of switches/routers [34] [32], with the servers as leaves. There is a core tier at the root of the tree, an aggregation tier in the middle, and an edge tier of switches connecting to the servers. . There are no links between switches in the same tier, or in nonadjacent tiers. In such topology, each switch are responsible for supporting of all traffic between its children. As depicted in Figure 2.6.

To understand the problem of such DCN, consider the case where all servers in the left half of the tree are sending traffic to the server on the right half of the tree. We see that the links between the first level Top of Rack Switches (TOR) and the second level Aggregate switches will carry twice the load of the links connected to the servers. Furthermore, the links between the Aggregate switch and the Core switch will carry four times the load of the links connected to the servers. Resulting in an even larger multiplicative load factor higher in the tree. Ideally, nodes higher in the tree would leverage links with increasing capacities to match these demands. In practice switches with high capacity links are far more expensive per Gbps than lower capacity ones. To mitigate costs, datacenter networks resorted to high oversubscription rates, with switches at each hop having far more downstream bandwidth towards the servers than upstream bandwidth towards the Core. The resulting scarcity of cross datacenter bandwidth was a performance bottleneck, limiting the throughput of datacenter applications.

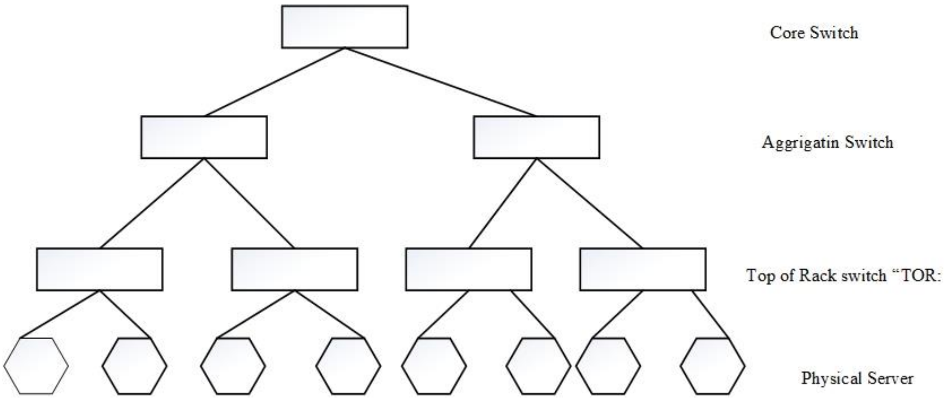


Figure 2.8: Basic tree topology

2) ***FatTree:*** Al-fares et al [35] propose new topology for DCN called FatTree that address basic tree topology limitation. FatTree is special instance of a Clos topology that use multiple commodity switch in palace of a single high-end., consider DCN build using k port switch, that form k pods, each pods have 2 layer switch, each layer have k/2 switch. At the first layer –edge switch- each switch is connected to k/2 hosts and the rest is connected to the second layer –aggregation switch. Each aggregation layer switch is connecting to k/2 core switch. Moreover, we have (k/2)2 core switch. In this topology that use k port switch it support (k3/4) hosts. Achieving an oversubscription ratio of 1:1

In Figure 2.9 is an example of FatTree topology, in this case we use 4 port switch. To build 4 pods, each pod have two layer of switch, the first one is edge layer and have 2 switch, and aggregation layer have 2 switch, edge switch are using 2 port to connect to hosts, and the other 2 port to connect to aggregation switch. This case connect 16 host and use 20 switch.

FatTree topologies enable datacenters to achieve far greater aggregate bandwidth in a cost-effective manner. The primary advantage of fat-tree topology is that it provide $k^2/4$ paths to route the traffic between any two servers, that is oversubscribing ratio is 1:1, and server can communicate with each other with full bandwidth -as switch port bandwidth- regardless of server location.

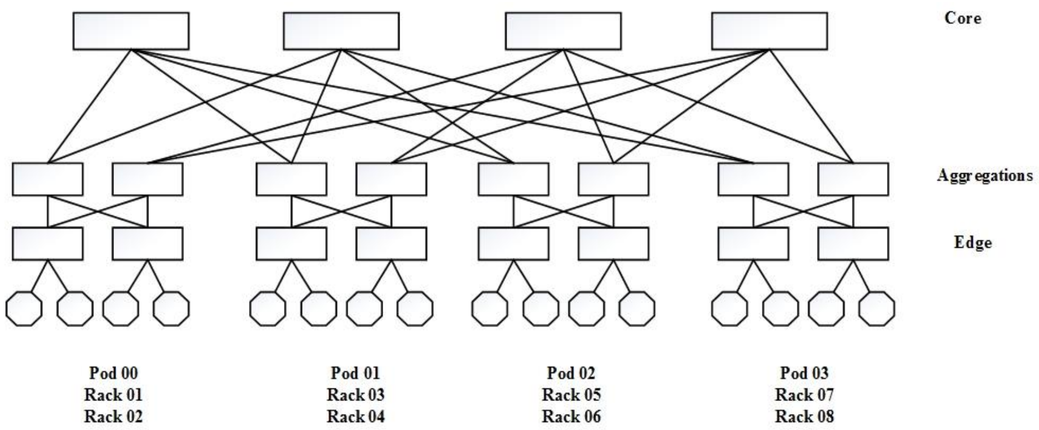


Figure 2.9: Fat tree topology.

2.3 Scheduling and Resource Allocation

According to Wikipedia, scheduling is the process of deciding how to allocate resources to a set of processes [36]. In general, scheduling is used to share system resource effectively or to achieve some quality of service level. In cloud computing scheduling is even more challenging, Also, these data center run many different kinds of applications with varying expectations from infrastructure. Cloud schedulers tries to optimize the utilization of data center as a whole. It is clear that in such environment, the role of a scheduler becomes very important in achieving high utilization without effecting application performance.

Cloud data centers typically make extensive use of virtualization technology, in order to ensure isolation of applications while at the same time allowing a healthy utilization of physical resources. Traditionally, achieving good utilization of server capacities was one of the key drivers behind the wide spread of virtualization technology. So that, Good VM allocation also helps to serve as many customer requests as possible with the given set of resources

An extensive research on the context of mapping of virtual resource to physical that could be classified into three main categories. The first category concern of mapping a VMs into physical host, the second concern on mapping virtual network links, in what been known as virtual network embedding problem (VNE). The last category, represented by handful of research that combine both filed, add new characteristic to it. Moreover, create what called virtual data center embedding research.

Network virtualization is a technology that enables hardware network resources to be shared among multiple concurrent software instances. That mean enabling multiple virtual instances to coexist on a common physical network infrastructure [37]. The network virtualization technology open the door for creating many virtual network, in which that virtual network is a collection of virtual nodes and virtual links that connect a subset of the underlying physical network infrastructure [37]. mapping a virtual network that connects a set of VMs onto a substrate network that connects a set of geographically distributed servers known as Virtual Network Embedding (VNE) problem, VNE problem been proofed to be (NP-hard) problem. On the other hand, VDC is defined as a set of VMs with an associated service level agreement (SLA), specifying the needs of CPU, memory, storage and inter-bandwidth [9]. VDC is composed of different types of virtual nodes (e.g. VMs, virtual switches and virtual routers) with diverse resources (e.g., CPU, memory and disk).

The problem of optimally allocating both servers and data center networks to multiple VDCs in order to maximize some objective like the total revenue, or minimize others like the power consumption. It is clear that designing an efficient resource management scheme for VDCs is a challenging problem. In addition, it is classified as an NP-hard problem as it generalizes the bin-packing problem.

As this problem is NP-hard, various heuristics have been proposed in the literature to solve this problem, such that using integer linear programming and Performing an exhaustive search of all possible solutions. Such approaches can be both time and resource consuming. On the other hand, greedy approaches as if the First-Fit algorithm that places a VM on is fast, but do not normally generate optimal solutions. Overall, approaches to solving the scheduling problem often lead to a trade-off between the time to find a solution and the quality of the solution found.

Before we dive into our main core research and stat of the art in the field of VDCE, I should clarify the confusion between this area of research and the area of VNE. Even it seems that both problem are related, but the VDC problem is significantly different than the VNE problem [38]. Following we explained this difference through point

The VN embedding different from VDC embedding [9].

- First, VN embedding generally treats all the substrate links equally without discrimination, while the significance of links varies in VDC embedding For instance, the core links are more important than edge links in data center since a core link may be used by more VMs than an edge link ordinarily [9].
- the VDC embedding has more constraints, as VM placement, virtual link allocation, switch resource consumption and special topologies are involved in simultaneously [9].
- VN embedding only consider CPU and network resource, whereas in VDC embedding other resources such as memory and disk also need to be considered [10].
- minimizing energy consumption has not been addressed in existing VN embedding models [10]. but it is significant to VDC problem.
- The number of nodes in ISP backbones is in order of hundreds, in case of data center number in order of thousands, Hence, the solutions to embedding problem in the VN domain can potentially raise scalability issues, and increase management complexity.
- Data center networks are built using topologies like the conventional tree, fat-tree, or Clos topologies, such topology have well-defined properties, researcher of VDC problem, may be relay on such properties to facilitate develop or optimized VDC. Substrate network of VN are heterogeneous and can't rely on it to optimize or develop VN embedding
- Propagation delay in VDC virtual links is negligible, as nodes are located in the same physical area. However, in case of NV virtual link, Low latency requirements are becoming increasingly important. Because VN connect nodes in different area, and through WAN link.

2.4 Simulation Tools

Researchers and industry-based developers of cloud computing are rely on simulation based experiments [39]. Academic researchers use simulation as risk-free and charge-free, while industry developers use simulation environments for the evaluation of different kinds of resource leasing scenarios under varying load and pricing distributions [40]. Other factors that push on the side of using simulation for cloud computing research are follow:

1. Using real cloud testbeds is expensive, since most research need allot of resource for long time.
2. Repetition of experiments on real cloud testbeds is not possible, since many factors are not under control of researchers.
3. Experiments on real clouds testbeds will be dependence on provider's specific infrastructure, so this will limit the scalability of application.

Different Cloud simulation tools have been developed. A full review of these tools is provided in [41]. Some of the well-known simulator, is CloudSim [42], CloudSim is extensible simulation toolkit, written in Java, served as the basis evaluation environment for many research works on Cloud. Although, CloudSim mainly consider application workload, that may be suitable for PaaS and SaaS. For IaaS simulation, where each virtual machine consider as resource by itself and could be requested and allocated to physical machine. Because of this, we try to find a simulation tools that concentrate on allocation of VM, and we find FlexCloud [43] most appropriate for us.

2.4.1 FlexCloud Architecture Model

FlexCloud use four main layers, as depicted in Figure 2.8 Brief description of each layer represented next.

Client Layer: provides the interface for user to configure requests properties and have results feedbacks from lower layers. In addition, FlexCloud will send results and comparison diagrams as feedback to Client Layer.

Broker Layer: implement Requests Broker Layer, acting as a mediator between Client Layer and Scheduler Layer. This Layer is responsible for verifying the inputs from Client Layer and transforming the settings into recognized commands at Scheduler Layer.

Scheduler Layer: implement three main components, VM Requests Generation component generates the VM requests with configured properties on user interface; Datacenter Scheduler component schedules the particular algorithms to allocate VMs to corresponding PM according to algorithms; VM Requests Allocation component manages the allocated VMs, including checking the allocation conditions and removing VMs at the end of their lifecycles.

Resource Layer: implements a Resource Management component providing resource that VM requests require and supporting services for higher levels. Such as physical resource, that include server and storage.

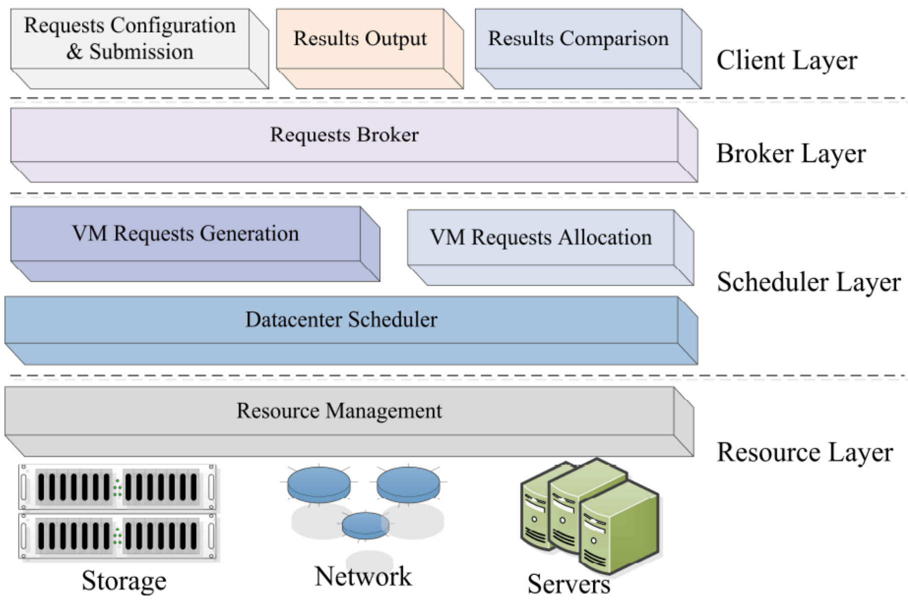


Figure 2.10: FlexCloud Architecture Model (source [43])

2.4.2 Modeling of Data Center

In current version of FlexCloud, data center consists of number of physical server (PM). A PM contains several kinds of resource, like CPU, memory and storage. The physical machine class is instantiated in FlexCloud before scheduling algorithms start. The number and property of PM can be defined in xml configuration file. The resource capacity decides whether a VM request can be allocated to PM or not. At initialization stage, PM has full capacity resource; mapping and later on removing VM from PM will update the available capacity value and influence the later requests allocation.

2.4.3 Modeling of VM Request

FlexCloud use predefined VM property, eight type of VM been defined, where each VM have its own property like CPU, Memory and storage. Beside this each VM have start and end time that define its life cycle by the number of time slot the VM will be a life. FlexCloud can be used to generate VM request, requests can be generated in Poisson, Normal and Random distribution. When the specific distribution is selected, the start time or duration of the generated requests would follow the distribution. Moreover, FlexCloud can import request data form real data file.

2.4.4 Scheduling Algorithms in FlexCloud

Based on scheduling goals and request type, FlexCloud provide four type of algorithms. Based on request type, algorithms can be classified as online and offline algorithms. The difference lies in whether the requests information is all known before scheduling. In online scheduling algorithms, request come and presses one by one. In offline scheduling algorithms all request been known before the starting of allocation process. Another division principle is via goal: FlexCloud consider load balancing and energy saving.

2.4.5 The Scheduling Process in FlexCloud

We summarize the major steps of scheduling in FlexCloud as follows:

1. Booting PM:
2. Generating traces (VM requests):
3. Comparing scheduling algorithms:
4. Output results:

2.4.6 FlexCloud Limitation

FlexCloud is simple simulation framework designed especially for resource allocation at infrastructure as a service level. Unfortunately, FlexCloud not supporting multiple user, so that all request belong to one user. Other main issue, FlexCloud not support modeling of user request as graph. We have to work around this limitation by extending and creating our own classes to support our simulation scenario.

Table below include summery of main future of different cloud simulation tools.

Table 2.1: comparison of different cloud computing simulation tools

Item	CloudSim	GreenCloud	iCanCloud	FlexCloud
Platform	Any	NS2	OMNET, MPI	Any
Programming language	Java	C++/OTcl	C++	Java
Availability	Open source	Open source	Open source	Open source
Physical Model	None	Limited	Full	Full
Model of public cloud	None	None	Amazon	Amazon

Chapter 3 : Related Works

Datacenter plays an essential role in realizing cloud computing paradigm. The essence of cloud computing is to allow applications of multiple entities to share resources in the underlying datacenters. It has been observed that relaying on traditional transport layer protocol to allocate bandwidth to different tenant application lead to interfere between these applications, resulting in severe lack of predictability and fairness of application performance.

Many researches [39,4,14,40,41,42,43,44] has done to address the problem of bandwidth allocation in data center with a number of new mechanisms to efficiently and fairly share data center network among multi-tenant. In this chapter, we try to explore such mechanisms and state of the art works, and address theirs advantage and disadvantage,

The network can have significant performance impact on applications in the cloud. Network performance has been cited as one of the prime concerns for many workloads in data center. Data centers are primarily composed of physical machines running virtualization software, with each physical machine hosting many virtual machines (VMs) simultaneously. The performance of the workload running inside a VM is affected not only by other VMs on the same physical machine, but in the case of a workload that uses the network, the location of the VM and other VMs it is communicating with in the network topology, and the utilization of all network links in between.

Traffic localization is an approach for minimizing the side effect of physical networking status, based on that internal bandwidth between VMs on the same physical machine is much more than available on the physical wire [44]. But here we should mention that deploying this approach will decrease the reliability of the system. Later on we go into discussion of how the trade-off between VDC bandwidth consumption and reliability.

Recent proposed solutions offers both computing and the network resource as virtual data center (VDC) [8]. A VDC is composed of a set of virtual machines interconnected by a set of virtual links [38]. The mapping of virtual resources to physical ones is commonly known as the VDC embedding problem. Next, we review specific efforts and works related to solving VDC embedding problem.

Guo et al [8] propose virtual data center (VDC) as the abstraction for resource allocation in multi-tenant cloud environments. VDC gives the illusion of dedicated physical data center. SecondNET mainly designed to provide high network utilization and low time complexity for embedding the VDC request. The authors differentiate the service provided by the cloud platform to three basic service types: a high priority end-to-end guaranteed service(type0), a better than best-effort service(type1) that offers bandwidth guarantees for the first/last hops of a path, and a best-effort service(type2). Initially, the type has been determined in SLA itself for unambiguous service guarantee.

VDC manager is a centralized entity to provide all kind of services like VDC creation, allocation, control and reconfiguration. VDC manager should maintain two type of data, first type is the data related to physical topology with all residual links capacity, and the second is resource allocation status, which include VM to physical mapping and bandwidth reserved for that links. VDC manager create VDCs based on a requirement matrix that defines the requested bandwidth between VM pairs.

SecondNet allocation algorithm consist of main three steps, first it partition the physical data center into clusters, and introduce the idea of hop-count as metric to group servers into clusters. Appropriate cluster is searched instead of the entire physical network when allocating the VDC request in order to minimize the allocation time. The second step is to build bipartite graph with the VMs at the left side and the physical servers at the right side, then use the min-cost network flow to get a matching. In the third step, the algorithm sort the requested bandwidth in descending order and allocate paths sequentially. Then allocate paths for the VM-pairs that have non-zero reserved bandwidths. SecondNET is simulated and verified the results in three network topologies like BCube, fat-tree and VL2.

Oktopus [4], propose two type of virtual data center abstraction, namely virtual cluster (VC) and virtual oversubscribed cluster (VOC), As depicted in Figure 3.1:

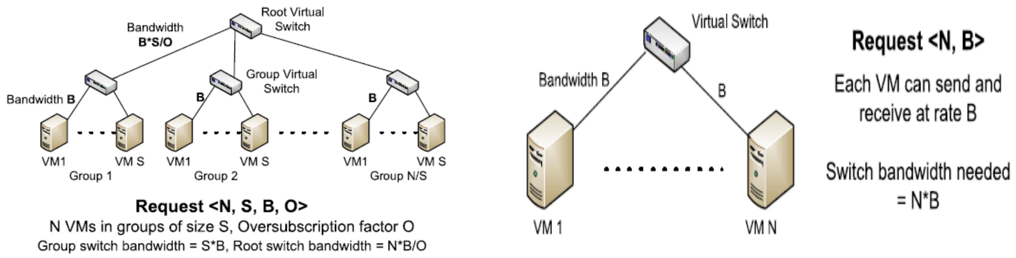


Figure 3.1: VDC request abstraction in Oktopus, (a) show VC, (b) show VOC. (source [4])

A virtual cluster provides the illusion of having all VMs connected to a single non-oversubscribed virtual switch. However, a virtual oversubscribed cluster emulates an oversubscribed two-tier cluster consisting of a virtual root switch that interconnects a set of virtual clusters. A tenant can choose the abstraction and the degree of the oversubscription of the virtual network based on the communication pattern of the applications to be deployed in the VDC. Initially Oktopus work with homogeneous environment that is all VM have the same capacity and bandwidth. However. Authors argue that both the abstractions and the algorithms can be easily extended to support multiple values of bandwidth and variable-sized groups within the same request.

Oktopus consist mainly of two components, management plane and data plane. In management plane implement the allocation algorithm, and in data plane Oktopus uses rate-limiting at endhost hypervisors to enforce the bandwidth available at each VM. Oktopus uses a greedy algorithm for the resource allocation to the VDCs that aims at finding the best trade-off between the performance guarantees offered to tenants, their costs, and the provider revenue. They try to map the VDC requests to the smallest sub-tree of the data center topology with the least amount of residual bandwidth on the links connecting the sub-tree to the rest of the topology. The main limitation of Oktopus is it only support two type of VDC request, moreover, VDC request can only applied to tree physical topology.

Zhani et al [10] argue that optimal allocation of server and network to multiple VDC in order to maximize the total revenue and minimize total energy consumption is not enough, and Suggest solutions that considered the possibility of using VM migration to dynamically adjust the resource allocation, in order to meet the fluctuating resource demand of VDCs. Zhani et al defines a set of objective for VDC embedding problem solution from an InP's perspective as follow: (1) maximizing the total revenue obtained from the embedded VDC requests, (2) minimizing request scheduling (i.e., queuing) delay, which refers to the time a request spends in the waiting queue before it is scheduled, and (3) minimizing the total energy consumed by the data center.

Increase the problem complexity, allowing SP to scale up and down their VDCs according to their needs. This issue not been addressed by previous work related to VDC embedding problem one solution to such problem is to re-embed the VDC from scratch. But more promising solution is to migrate some embedded VM from one server to another.

Zhani et al propose VDC Planner, a framework that supports migration-aware virtual data center embedding. That leverages migration techniques to achieve effective and efficient placement of VDCs over time. This problem is intractable because it requires solving a multi-dimensional bin-packing problem dynamically over time. Therefore, a more scalable yet cost-effective solution is needed. Figure 3.2 show the architecture of VDC planner.

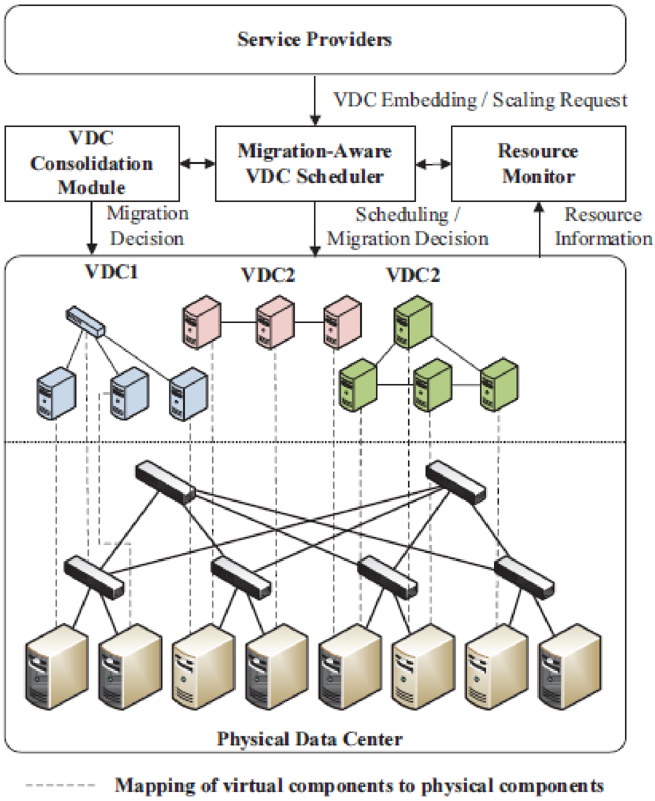


Figure 3.2: VDC Planner Architecture (source [10])

VDC planner consist of the following components: VDC scheduler, Resource Monitor, and VDC consolidation Module, and two heuristic algorithms The first heuristic is designed for migration-aware VDC embedding, The second heuristic is designed for dynamic VDC consolidation. The framework supports various usage scenarios, including VDC embedding, VDC scaling as well as dynamic VDC consolidation. By studying the algorithm VDC planner is based on, we note that algorithm are not accounting the virtual link between the virtual machine, mapping o f of virtual links to physical links have not been incorporated to service the incoming request which enables the communication between the VMs a guaranteed service. Moreover, VDC planner not include or use availability information for embedding.

Zhang et al [6] proposed an availability-aware VDC embedding framework called Venice “A**V**ailability-aware **E**mbeddi**N**g **I**n **C**loud **E**nvironments”. Venice try to address the availability aspects with VDC embedded. To achieving availability aware VDC embedding is a nontrivial problem, first, a single service often consists of multiple virtual components. Thus, it is necessary to capture the dependencies among virtual components in the VDC availability model. Second, that physical data center components have non-uniform failure characteristics in terms of failure rates, impact and repair costs. Venice include three main components, namely; reliability analysis module, reliability-aware VDC scheduler and monitoring module.

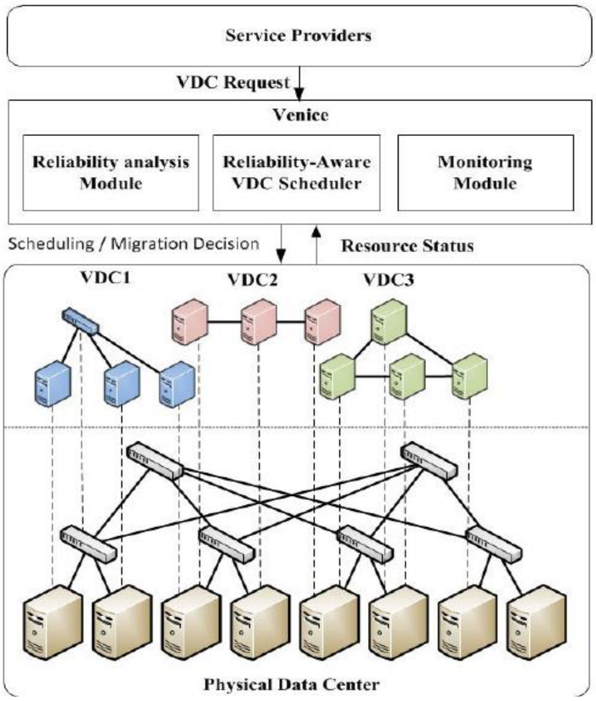


Figure 3.3: Venice Architecture. (source [6])

The framework promise to offer the following feature, Migration-based scheduling, Dynamic scaling, and Periodic consolidation. The framework use greedy approach for VDC scheduling. Venice availability-aware algorithm runs many times to achieve the best optimal embedding solution. The machine with lowest availability of resources will be dropped in each trial. The high availability of physical machine will be selected for embedding the current incoming VDC request. At the same time, over provisioning of resources in the name of high availability is also to be considered to avoid high resource wastage.

Venice been implemented through simulation, and evaluated its performance against VDC Planner. The simulation is based on VL2 topology, and authors use three type of topologies for experiments: Multi-Tiered, Partition-Aggregate, and MapReduce.

Xie et al [11] argue that simple VC abstraction model introduce by [4] using fixed bandwidth reservation waste network resource. Authors study traffic pattern of application –mainly MapReduce application- and show that such application only generate substantial traffic during only 30%-60% of the entire execution time. Thus reserving peak bandwidth requirement for the entire execution waste network resource and reduce the number of jobs that can fit in the datacenter to run concurrently. Based on this observation, authors propose new network abstraction that can express the time varying nature of application called temporally interleaved virtual cluster (TIVC). TIVC is very similar to VC, but instead of using fixed B to link VM to virtual switch, it use time-varying function B(t). Authors propose four type of this function model, The model generation algorithm takes as input the traffic profileof an application profiling run under bandwidth cap Bcap. As proof of concept, authors develop PROTEUS, a system that implements the new abstraction.

Guo et al [12] propose Falloc, a new bandwidth allocation protocol, with main objectives:(i) Guarantee bandwidth for VMs based on their base bandwidth requirements, and (ii) share residual bandwidth in proportion to weights of VMs. Authors solve the resource sharing problem in datacenter by modeling the bandwidth allocation process in datacenters as a asymmetric weighted Nash bargaining game of sharing. Contribution of Guo et al can be summarized as, present the design of Falloc, an application layer bandwidth allocation protocol to achieve fairness among VMs in datacenters. Where each VM is assigned with base bandwidth and weight. Should mention that Falloc prototype been implemented with OpenFlow and evaluated on MiniNet Test bed, using simulation technique with Mapreuce workload and find that Falloc can achieve a utilization approximate to the best effort manner while providing performance guarantees for VMs by enforcing a fair bandwidth allocation.

Jeyakumar et al [1] look to problem form anther point and try to solve it with platform to enforce predictable network bandwidth sharing within the data center, using minimum bandwidth guarantee to endpoints, named EyeQ, EyeQ fall under the umbrella of network Quality of service (QoS). EyeQ's design has two main components: (a) a rate meter at receivers that sends feedback to (b) rate limiters at senders. A combination of the above is needed to address both contention at the receiver indicated using feedback, as well as local contention at the sender. The rate limiters work in a distributed fashion using a control algorithm to iteratively converge to the 'right' rates.

Popa et al. [45] define how network sharing is difficult as it does not only depend on the VMs running on the same machine with X, but also on the other VMs that X communicates with. Authors add one more point to above mentioned problem, that network are not shared proportionally to payment. Authors argue that desirable solution for sharing cloud networks Should meet three requirements. First is min-guarantee, which can be define as the lower bounds for the worst-case performance of an application. Second is high-utilization, aims to maximize network utilization in the presence of unsatisfied demand. Third requirement is network proportionality, this mean that two tenant with the same number of VM should get the same aggregate bandwidth since they paid the same amount of money. Authors present three allocation policies that solve the tradeoff between above mentioned requirement, namely Proportional Sharing at Link-level (PS-L), Proportional Sharing at Network-level (PS-N) and Proportional Sharing on Proximate Links (PS-P).

Kumar et al [46] argue that static network reservation proposed before in literature have not been implemented in real world cloud because it lead to poor performance and resource wastage, and propose new adaptive network reservation system named Equinox. The proposed system worked at network level and consists of two major components, first, Flow Monitoring Service. Second, Network Reservation Service. Flow monitoring service is responsible for monitor the traffic in the network to estimate the demands of the VMs. Authors extend OpenStack Quantum to configure flow based monitoring at the Open vSwitch on all end hosts. The Network Reservation Service is responsible for computing and enforcing reservation through routing and end-host rate limits in the network.

Summary

The needs of running cloud application without deprecation of performance, lead to rise of research interests in cloud bandwidth allocation, which in turn depend on the characteristics of the underlying datacenter networks. In this chapter, we walked through existing efforts in sharing datacenter networks. We focus on works that try to provide deterministic bandwidth guarantees though using static reservation.

Deterministic of bandwidth guarantee each tenant to have predictable performance, irrespective of other tenant’s behavior. In such approach provider of cloud service should provide tenants with an interface to explicitly specify their bandwidth demand. Such guarantees are achieved by means of proper placement of tenant VMs and static enforcement of rate limits.

Chapter 4 : An Approach for Bandwidth Guarantee in Multi-tenant Data Center

Adapting virtual data center (VDC) allows user to express his needs in free way, such that he expresses his needs of computing power, bandwidth requirements, and the network topology. On the other hand, tackling VDC is tackling a set of NP-hard problem. This chapter is dedicated to propose our approach of how to deal with VDC embedding problem. In the first section, we introduce the mathematical model of VDC request and the physical data center. Then we setup our objective and constraints. We finish this section by describing the general properties of Fat Tree topology. In next section we introduce our proposed solution, alongside with rational of main idea our solution build on. Finally, our two phase algorithms to consolidate VM/links and mapping them to physical machine.

4.1 Mathematical model of VDC and Physical Data Center

This section contain the mathematical model of physical data center and VDC request.

The physical data center network is modeled as a weighted undirected graph, $G^p(N^p,E^p)$.Where, N^p is the set of physical node, E^p is the set of physical links. Each physical server in N^p is associated with a set of hardware feature, which typically include CPU, memory, and data storage, We associate with each physical server $n^p \in N^p$ a set of featured capacities $c_i(n^p)$, $i \in \{1,\ldots,k\}$ where k is the number of different reserved hardware capacities. For example, $c_1(n^p)$ can refer to CPU, and $c_2(n^p)$ refer to memory capacities. We also define the residual capacity of each hardware feature such that $\overline{c_i}(n^p)$, $i \in \{1,\ldots k\}$ to denote the residual capacity of hardware feature. For example $\overline{c_1}(n^p)$ can refer to number of residual CPU after some CPU been rented by tenants. For each physical link $e^p \in E^p$, we associate bandwidth capacity donated by $b(e^p)$. In addition, residual bandwidth capacity donated by $\overline{b}(e^p)$. Table 1 shows notation used for modeling the physical network infrastructure.

Table 4.1: Notation for physical network infrastructure

$G^p(N^p, E^p)$	The physical network infrastructure.
N^p	Set of physical nodes, nodes representing physical server
E^p	Set of physical link
$c_i(n^p)$	the physical node capacity
$\bar{c_i}(n^p)$	The residual of physical node capacity.
$b(e^p)$	Physical link capacity
$\bar{b}(e^p)$	The residual of physical link capacity

.

On the other hand, VDC request which donate what tenant wants to reserve has been modeled in similar way to physical data center mentioned above. The VDC request is also modeled as weighted undirected graph. $G^v(N^v, E^v)$, where N^v is a set of VMs, E^v is the set of virtual links. We associate each $n^v \in N^v$ with a set of feature that VM should have donated by $c_i(n^v)$. Virtual link $e^v(i, j) \in E^v$, associated with bandwidth capacity between two mentioned point (i,j) donated by $b(e^v)$. Table 4.2 summarize the used notation for VDC requesting modeling.

Table 4.2: notation for VDC request

$G^v(N^v, E^v)$	VDC request
N^v	Set of VMs
E^v	Set of virtual Links
$c_i(n^v)$	Capacity that VM should have.
$b(e^v)$	Bandwidth required between virtual nodes

4.2 Functional Objective and Constraints

From the tenant point of view, VDC is a virtual network abstraction model, that tenant use to express his needs of both computing resource alongside with bandwidth requirements. From the cloud provider point of view, VDC embedding is an optimization problem, in this context, cloud provider want to maximize revenue by maximizing number of accepted VDC request, same time minimizing his cost through minimizing bandwidth consumption. Based on this, we define a set of objective that our solution try to archives. Moreover, we define set of constraints that control our objectives. Next to that, we discuss some trade-off between different objective and constraints.

We define VDC embedding problem objective as follows:

1. The goal of cloud provider and more specifically the infrastructure provider (InP) is to maximize profits, by maximizing its infrastructure utilization. Which in turn implies maximizing the number VDC request that successfully embedded.
2. Provide tenants with certain level of reliability against server hardware failure. That is, limit the impact of server failure on the overall tenant VMs.
3. Minimize the bandwidth consumption by tenant's request. That is implies that we try to localize the tenant request.

Our solution is controlled by following constraints:

1. We are dealing with online version of VDC embedding problem, thus the algorithm will processed with incoming VDC request without knowledge of future VDCs request.
2. Each VM should be embedded on one and only one physical server –no redundancy- , at the same time all VMs belong to one VDC request, should be embedded. If not possible then the VDC request will be rejected.
3. Capacity of physical server cannot be overused by embedded VMs.
4. For successful mapping, Residual capacity of physical link should be equal or more than the capacity of virtual link
5. Capacity of physical link cannot be overused by virtual link,
6. Virtual link could go through multiple hops; the maximum capacity of such link is determined by the minimum residual capacity from source to destination.
7. Minimize hops counts. On other words, minimize communication cost.
8. We tack into account reliability of user request. We only consider server failure. We define reliability factor donated by (R) as (number of VMs failure) / total number of VMs > 75%. R used to determine the maximum number of VMs that can be mapped into one physical host.

Looking deep in our objectives, we consider the trade-off between providing reliability and minimizing the bandwidth consumption.

4.3 Fat-Tree Properties

Data center based on fat-tree topology is one of the most used topologies in current and new build data center. In addition to that, fat-tree topology provide non oversubscribing topology making the process of embedding tenant request more easy than traditional tree like topology.

By using fat-tree as data center network topology, we assume that if two physical machine v1, and v2, each one have b1 and b2 of network bandwidth that connect each physical server to its access server. Then there is existing path between both physical machines, the size of this path is min (b1, b2). That is imply, if required bandwidth between two VMs is equal or smaller than minimum of (b1, b2), then there is existing path between both VMs and no need for link mapping. Although, different VMs placement could lead to different bandwidth consumption.

Table 3 shows the general properties of fat-tree topology with an instance example of 6-port and 48-port switch. Some interesting propriety of fat-tree is number of hops is the same regardless number of switch ports. Figure 4.1 show graphical representation of fat tree network topology that use 6 port switches.

Table 4.3: fat-tree topology with different K-port switch, where K is number of switch ports.

Fat-Tree properties	Parameter	An example instance	
		6 port switch	48 port switch
Number of core switch	$(K/2)^2$	9	576
Number of pods	K	6	48
Number of aggregation switch in one pod	k/2	3	24
Number of edge switch “TOR” in one pod	k/2	3	24
Total number of switches	$5K^2/4$	45	2880
Number of hosts in each rack	K/2	3	24
Number of racks in each pod	K/2	3	24
Number of hosts in each pod	$K^2/4$	9	
Total number of hosts	$K^3/4$	54	27648
Number of path in case of intra-rack communication	1	1	1
Number of path in case of intra-pod communication	k/2	3	24
Number of path in case of inter-pod communication	$k^2/4$	9	576
Number of hop in case of intra-rack communication	1	1	1
Number of hop in case of intra-pod communication	3	3	3
Number of hop in case of inter-pod communication	5	5	5

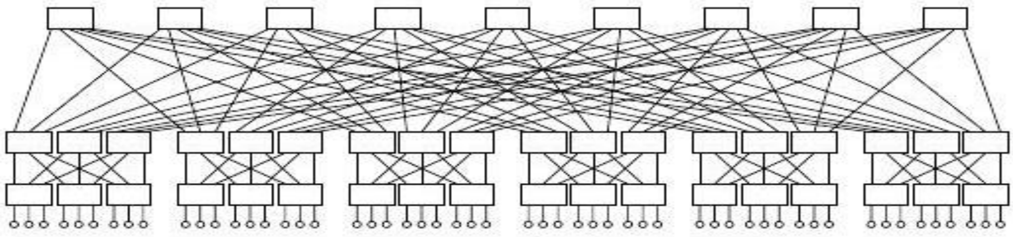


Figure 4.1: An instance example of 6-ary fat-tree

4.4 Proposed Solution

Neither VM placement nor virtual network embedding (VNE) are classified as NP-hard problem. For practical implementations, heuristics are needed. our heuristics are based on the following criteria:

- Finding the best groups
- Bandwidth consumption model
- Traffic aware placement
- Minimize defragmentation of resource

Next sub-section we discuss these criteria in more details, then we introduce our proposed two phase's algorithm.

4.4.1 Finding the Best Group

We observe that mapping the same VDC request into physical machine with different arrangements lead to different bandwidth consumption. Figure 4.2 depicted this idea. The figure show VDC request and the physical date center where the VDC request should be mapped. Next to that, two possible mapping. We see that in the first mapping we consume 22 unit of bandwidth, this is because node 4 needs 7 unit of bandwidth to communicate with node 3 and 2, also node 1 in the same physical server needs 3 unit of bandwidth to communicate with node 3 and 1. The total bandwidth required by node 4 and node 1 located in the same physical server is 11 unit of bandwidth. The second physical server hold both node 3 and 2, require 11 unit of bandwidth to communicate with nodes located in server holding node 4 and 1. While in the second mapping only consume 10 unit. This is because node 4 and 2 been holed in the same physical server, require only 5 unit of bandwidth for node 4 to communicate with node 3 and node 2 to communicate with node 1, while node 4 and 2 communicate internally, in the second physical server that hold node 3 and 1. Node 3 need to communicate with node 4 by 2 bandwidth unit and node 1 needs 3 unit of bandwidth to communicate with node 2.

We define the best groups to be the set of VMs that when allocated together in the same physical machine, the bandwidth consumption will be minimized. To find the best group, we use graph partitioning algorithms. We model VDC request as undirected weighted graph $G = (V, E)$ consists of a set of n vertices $V = \{v1, v2, \ldots, vn\}$ and a set of edges E. An edge ei, j represents a connection between vertex vi and vj .Both vertices and edges can have weights associated with them. Weighted vertex represent VM capacity such that, number of cores and RAM. Weighted edge represent bandwidth between vertexes. Note that as undirected graphs, the edges are symmetric (*ei, j =ej,i*).

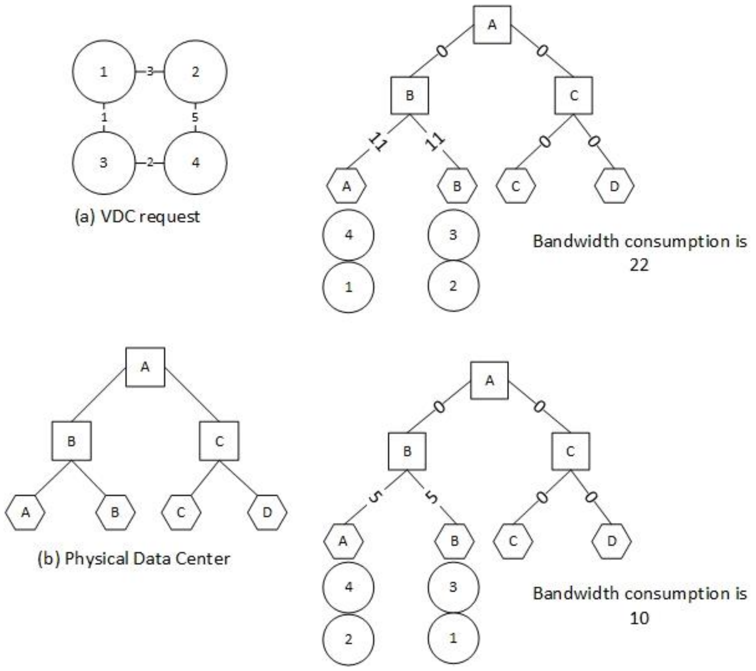


Figure 4.2: different mapping of the same VDC request into the same physical DC

the partitioning problem objectives is to partition the VDC request graph into sub set of graph that have high communication between them, and have low communication between different groups. We are not intended to do balance partition, where each sub graph have the same number of VMs, instead we most care about is to minimize the connection between different sub graph. The partitioning algorithm are constrained by three factors: F**irst**, the maximum number of node inside each sub graph should not exceed the reliability factor defined by customer, and calculated as proportion to total number of requested VMs. The number of VMs inside one group donate by M guarantee that single physical server failure will only impact small portion of the user VMs. **Second,** the total capacity of group should not exceed the capacity of standard host which been defined by InP provider. **Third**, the maximum bandwidth between groups should not exceed the capacity of physical link defined by provider.

4.4.2 Bandwidth Consumption Model

The bandwidth consumption of VDC i, that connect VM u and VM v, is the summation of the required bandwidth between the two VMs –based on VDC request-across all link between theses VM. Mathematically this is can be expressed as follow.

$$BC(i) = \sum_{u,v \in V^i} (Hops_{u,v} . BW_{u,v})$$

Where BC(i) is the bandwidth consumption by VDC I, Hops is the number of hops between the two VMs, BW is the bandwidth requested based on VDC request between the two mentioned VMs.

It is clear here, that increase the distance –number of hops- between VMs that have to communicate to each other will increase the bandwidth consumption. In this context, One of the main future of our solution is that enforce multiple VMs of the same request to be embedded in the same physical machine. This is because that bandwidth consumption of VMs located on the same physical server will be assumed to be equal to zero, that’s hops count is zero so total bandwidth consumption will be zero. Our justification for this is that the internal bandwidth is much larger than the network bandwidth. However, we should mention that increasing of VMs consolidation would decrease the reliability of the VDC.

4.4.3 Traffic Aware Placement

Based on Fat-Tree topology, we can identify four type of communication:

1. Inside host. In this type, both VMs are located into the same physical server, so that hop count is zero.
2. Intra-Rack communication. In this type of communication VMs located into different physical server, but both server are located into the same Rack, so that both server are connected through the same TOR switch. Hop count is one.
3. Intra-Pod communication, communication between different VM located into different physical server, and physical server are not connected directly though TOR switch, so that each physical server is connected with different TOR, but TOR switch are connected though aggregation switch, hop count is three. (TOR switch, Aggregation switch, TOR Switch).
4. Inter-Pod communication, VMs located into different physical server, each physical server are connected with different TOR switch, and TOR switch are not connected though the same aggregation switch. That is communication go through core switch, hop count is five. (TOR switch, Aggregation switch, Core switch, aggregation switch, TOR switch).

VM placement algorithm try to keep the locality of communication as much as possible by following the same order when placing VM. So first, we try to embed VM with heavy traffic into the same physical machine. If not possible, then try to embed VMs into different physical machine but at the same rack, and if not possible, then try to find another physical machine in different rack that belong to same pod. If not possible then separate the VDC request into different pod.

4.4.4 Minimize Defragmentation of Resource

It been observed that leaving some defragmentation of resource here and their lead to waste of resource. For example, suppose the case when we have two server, first with two CPU and the other with three CPU, and incoming request of VM that need 2 CPU, both physical server can embedding the requested VM. In both cases, the reminding is three CPU. However, we chose to embed the request in the server with two CPU, so that future incoming request have better chance in finding server with adequate resource available. Based on this idea, our mapping algorithm try to find server with minimal capacity to embedding VDC request to it.

4.5 VDC Embedding Algorithms

Our approach to the VDC embedding problem consist of two phases. Below we introduce each phase along with its objective, constraint and algorithm.

4.5.1 Phase I: VM/Link Consolidation

Phase I: VDC Request pre-processing and finding of best groups

The objective of this phase:

1. Find the best group, as been explained on Section 4.4.1, different arrangement of VMs leads to different bandwidth consumption. So finding the best group, will save bandwidth, and save the time and efforts needed to allocate virtual link.
2. Save bandwidth, Bandwidth consumption of VM to VM inside one physical host are equal to zero.
3. Simplify the next steps. First, consolidation of multiple VMs into one group by combining each VMs capacity together. For example if VM1 is 2 core and 2 Gig of RAM, VM2 have 3 core and 3 Gig of RAM, if both VMs been allocated on the same group, the next phases of our algorithm will treat both VMs as one super VM that have 5 core and 5 Gig of RAM. Second, virtual links connecting different groups will be consolidation to make super virtual link.
4. Partitioning of VDC request serve as reliability factor, by sitting the maximum number of VMs inside each group, this lead to increase the reliability of user VDC.

We build our own algorithm called simple vertices/edge consolidation. It based on consolidation of vertices that have strong communication into one super vertices with sum of both vertexes capacity. In the next phase of our algorithms, this super vertex is embedded in single host. Our algorithm obey to the following constrains.

1- Number of VM in super node should not exceed reliability number donated by reliability factor.
2- Summation of total VMs capacity in super node should not exceed standard physical host capacity.
3- Virtual link capacity should not exceed standard physical link capacity.

Figure 4.3 depicted our VM/link consolidation algorithm. The algorithm is following greedy approach, a brief explanation is following: suppose a new VDC request G^v arrives, represented as adjacency matrix. Algorithm start by biking heaviest link, if many exist, pick the first. Check the two end point of this link *u*, and *v*, if summation of both vertices capacity not exceed standard physical host then combine both vertices into one super node name it uv^-. – This will show that the new node is a result of summation of u and v node. If, as result of consolidation there is multiple path between vertices exists, then consolidate it if its capacity not exceed the physical link. At any point if capacity exceed the physical capacity, or number of vertices exceed reliability factor. Then pick the next heaviest link and start again. Stop when no more vertexes or link could be consolidate without violating our constrain. After finishing the consolidation stage, calculate the bandwidth of each VM.

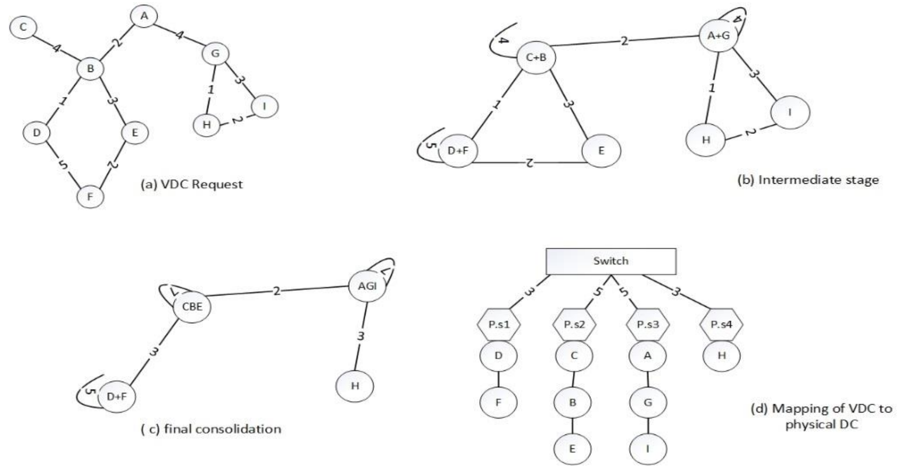


Figure 4.3: depiction of VM/link consolidation algorithm

Psedocode of VM/Link consolidation algorithm.

Algorithm: VM/Link consolidation

Input: VDC request, modeled as undirected weighted graph, and represented by adjacency matrix.

Output: simple form of VDC request.

Step 1: calculate the reliability factor <= 25% of total number of VMs. This number will determine the maximum number of VMs in one group.

Step 2: sort the edge value with its corresponding node in Descending order.

Step 3: pick the most weighted edge. If more than one exist, select one randomly. Determined what vertices are connected by this edge, u1 and u2.

Step 4: check if (u1 + u2) capacity < = standard host capacity && total number of node not exceed R “reliability factor”,

Step 5: check if u1 + u2 combination will produce multiple arc.

Step 6: check if summation of multiple arc <= standard physical link capacity.

Step 7: then combine both vertices into one super vertex named u1u2.

Step 8: Update adjacency matrix

Step 9: else, update the value of edge = -1; so this edge will not checked again.

Step 10: Go to step 3; until no more node can be combined together.

Step 11: calculate the new bandwidth requirements and associated to each VM.

4.5.2 Phase II: Virtual Machine Placement and Link Mapping

In this phase of our solution, we consider both VM placement and link mapping together; in fact, we treated bandwidth as a new property of VM. This is based on our assumption of using Fat Tree network topology. We assume that each VM is connected to central swath, bandwidth between VM and switch been calculated based on the output of phase one. We illustrate this idea by an example, suppose we have three VM, x, y, and z, connected in bus like this x—3—y—2—z. so that VM x connected with VM y with 3 unit of bandwidth, and VM z connected with VM y with 2 unit of bandwidth, the bandwidth required by VM y should be 5 unit of bandwidth, So that VM y can communicate with both x and z simultaneously.

We consider the VM placement problem as selecting the most suitable physical server to create the required VMs, with the following objectives: traffic aware placement, and minimal bandwidth consumption. More details on this objective in Section 4-4.

The input of this phase is the output of phase one, that is, simple matrix, define VMs groups. We consider VDC request as set of subgroups, even for individual VM, we account it as group with one element. The bandwidth of each groups will be calculated as been mention above. Start by randomly pick up one physical machine. Compare its capacity with VDC request groups capacity. If fit, map the group to physical machine (PM), and update resource, mark this physical machine so it will not be used to embedded any other groups of the same VDC request. If first group been embedded, then select another physical machine belong to same rack,

---

Algorithm: VM Placement.

Input: adjacency matrix of simplified VDC request, and array list of VMs with associated bandwidth based on host model.

1. Sort the VDC request in descend order. Based on number of CPU
2. For each incoming VDC request, select one pod randomly. Select one physical machine.
3. Check if residual capacity of PM is equal or bigger than VDC request groups.
4. If residual capacity of PM > VM group capacity,
    a. Embedding the largest group into smallest available host. “this is very important to minimize the defragmentation of resource”
    b. Update resource
    c. Mark this PM, so it will not been cheek to serve any future request belong to same VDC request.
    d. Select anther PM, belong to the same rack. Go to step 3.
5. Else, go to step 2.

4.6 Summary

This chapter been dedicated for presenting our approach to solve the bandwidth sharing problem. We first introduce our mathematical model of both VDC and data center, then we discuss functional objective and constrain. Next, we discuss the main idea that stand behind our heuristics solution. Finally we introduce our algorithms for simplifying VDC and VDC mapping. In next chapter, we show our evaluation of proposed solution.

Chapter 5 : Simulation and Performance Evaluation

In this chapter, we realize our proposed approach for bandwidth sharing in multi-tenant data center using simulation tools. First, we describe the simulation environment followed by defining the performance metrics. Next, we conduct the necessary experiments to explore stronger and weakness of our approach against others. Finally, we present our result and analyze it.

5.1 Simulation Environment

We implement our algorithms in FlexCloud, (see Section 2.5). Our data center network use Fat Tree topology. For all three level of switching, we use six-port switch, each port bandwidth is 1 Gbps. –for more detail regarding fat-tree specification and parameter refer to (Section 4.1.4). For the physical machine, our data center use one type of physical machine, for detailed specification see Table 5.1.

By using fat-tree Topology, we assume that there is an existing path between any two pair of VMs regardless the location of VM. Despite that, allocation of VMs pair in different rack or pod lead to different bandwidth consumption, which we try to minimize it. Refer to previous section for more details regarding bandwidth consumption model.

The bandwidth consumption model is key point to differentiate our algorithms with other work. Unfortunately, Due to limitation of FlexCloud simulator network model that not considering bandwidth utilization at switch level. We cannot calculate the bandwidth consumption of each VDC request. So, we based on FatTree property that define fixed hop count between different VMs pairs based on location of that VMs. then manually calculate bandwidth consumption of each VDC request.

For VDC request, we use two type of pre-defined VM, specification detailed in Table 5.1. For the network topology that link these VM, we use star, mesh, and tree topology. See Figure 5.1 for the main types of VDC network topology. The number of VMs in each topology are distributed from 5 to 19. Bandwidth between different VM belonging to same user –the same request- are distributed between 32 to 192 Mbps. All VMs belonging to same request have the same start time, but each VM have its own finish time, our algorithm will deal with each request online, and server request as it arrive the system. When some of VMs finish, it will leaf the system, and the system resource will be updated, even that other VMs belonging to same request still active.

Due to limitation of FlexCloud simulator of not supporting multiple user, and more important, not supporting tenant request as a graph. We built java classes that read tenants VDC request network as adjacency matrix, alongside with VDC VMs types, start time, and finish time. This class also implements our VM/Link consolidation algorithm. The output of this class is two text files, in the first one is a list of all requested VMs with the bandwidth requirements of each VM based on VDC request graph. The second output file contains a list of consolidating VMs, request Id and required bandwidth. Later on, we modify the FlexCloud to accept the new request format. To illustrate these input and the two outputs , following is a sample of input and output of this class.

The Input:

[101, 102, 103, 104, 105, 106, 107, 108, 109] // name of VMs

[1, 1, 1, 1, 1, 1, 1, 1, 1] // type of VMs

[2] // start time of VMs

[50] // finish time of VMs

[0, 32, 0, 0, 0, 0, 64, 0, 0] // adjacency matrix of VDC

[32, 0, 64, 64, 128, 0, 0, 0, 0]

[0, 64, 0, 0, 0, 0, 0, 0, 0]

[0, 64, 0, 0, 0, 192, 0, 0, 0]

[0, 128, 0, 0, 0, 32, 0, 0, 0]

[0, 0, 0, 192, 32, 0, 0, 0, 0]

[64, 0, 0, 0, 0, 0, 0, 64, 96]

[0, 0, 0, 0, 0, 0, 64, 0, 128]

[0, 0, 0, 0, 0, 0, 96, 128, 0]

The Output 1: // VDC without VM/link consolidation algorithm

1 1 1 101 2 50 96 1 /* sequence number, VDC id, VM weights, VM name, start time, finish time, bandwidth requirements, VM type */

2 1 1 102 2 50 288 1

3 1 1 103 2 50 64 1

4 1 1 104 2 50 256 1

5 1 1 105 2 50 160 1

6 1 1 106 2 50 224 1

7 1 1 107 2 50 224 1

8 1 1 108 2 50 192 1

9 1 1 109 2 50 224 1

The Output 2: // VDC with VM/Link consolidation algorithm

1 1 1 101 2 50 96 1 /* sequence number, VDC id, VM weights, VM name, start time, finish time, bandwidth requirements, VM type. */

2 1 1 103 2 50 64 1

3 1 2 104106 2 50 96 1

4 1 2 102105 2 50 192 1

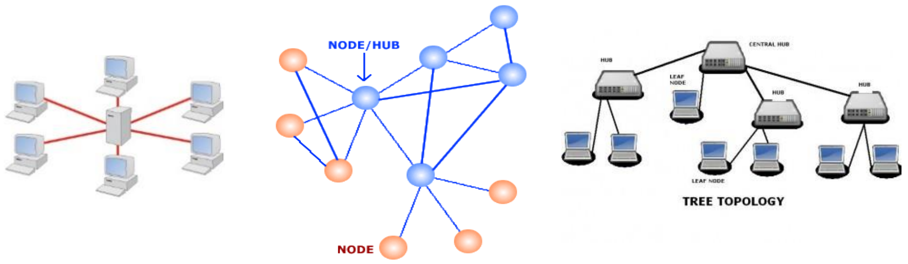
5 1 3 107108109 2 50 64 1

Note: VM specifications do not specify a value for bandwidth, because we set the bandwidth to be a property of the link that connects different VMs. However, In case of the physical machine, we specify bandwidth value as the value of physical link bandwidth connecting the physical machine to access switch.

Table 5.1: Physical and virtual machine specification

Specification	VM type 1	VM Type 2	Physical Machine
CPU	1	4	16
Memory	1.7	7.5	30
Storage	160	850	3380
Bandwidth	-	-	1024

Figure 5.1: VDC request represented ad physical network



We implement the second phase of our solution (placement algorithm, namely R-VDC algorithm) inside FlexCloud framework, by overriding the abstraction allocation method. This method accept the new request format, reading the requested VM file (the output of phase one file) at once, creating VM object of each individual line. As mentioned above, FlexCloud does not support multiple tenant requests, so we work around this limitation by grouping each object based on the value of VDC request ID. Each group that represent individual tenant VDC request will be mapped to the most appropriate physical machine based on VM placement algorithm.

5.2 Performance Metrics

We show the advantage of our algorithm by comparing a set of performance metrics against based line algorithms that based on randomized heuristics (provided by the FlexCloud itself). This algorithm pick one VM based on its start time, and then randomly pick a physical machine (PM) to map the VM to it. If PM have enough capacity to hold the VM then map this VM to PM, otherwise pick another PM. The algorithm will keep iterate until finish all VM, or until no PM have capacity to hold that VM.

We compare our algorithm with other based on the following metrics:

- VDC Acceptance ratio: it is the ratio of successfully embedded request to total number of request at certain point of time.
- Network bandwidth consumption: defined as summation of bandwidth required between pairs of VMs multiple by number of hops, between this pairs of VMs.
- Network bandwidth utilization: defined as the total bandwidth allocated to VDCs divided by the total link capacity.
- Server bandwidth utilization: defined as the total server bandwidth allocated to VDCs divided by the total server link capacity.
- Active server bandwidth utilization: defined as the total server bandwidth allocated to VDCs divided by the total active server link capacity

- Memory utilization: defined as the total server memory size allocated to VDC divided by total size of server memory. (for each PM)
- Storage utilization: defined as the total server storage size allocated to VDC divided by total size of server storage. (for each PM)
- Active physical machine: the number of physical machine that participate in severing VDC request.

The key difference between network bandwidth utilization and server bandwidth utilization is that the former defines utilization at aggregation and core switch, while the later only considers utilization between physical server and access switches.

5.3 Simulation Results

To study the performance of our approach, we run each phase of our approach as standalone phase, collecting results, and then combine both phases into one process. At each stage of our study, we collect results and analyze them.

In phase one of our approach (Section 4.5.1, Phase I: VM/Link consolidation) the input is set of adjacency matrix with other properties like start and finish time. Table 5.2 shows sample results of running this phase.

Table 5.2: Sample request size before and after consolidation.

VDC request ID	Number of VMs	Number of consolidated VMs groups	Bandwidth requirements (without consolidation)	bandwidth requirements (with consolidation)
1	9	5	1728	512
7	13	11	1920	992
11	18	15	2304	1536
22	7	3	1024	224
27	15	12	1536	1024

Results show that the number of VMs and bandwidth requirements are decreased. The order of decreasing is not specify by any specific parameter, instead it depend on the type of graph and weight of link between each VM pairs. The Table 5.3 show total number of VDC requests, the total number of individual VMs, and the total number of VMs groups.

Table 5.3: input/output VDC request to VM/link consolidation algorithm

Number of VDC request	Number of individual VMs	Number of VMs groups
38	399	289

We run phase two (Section 4.5.2, Phase II: VM placement algorithm) of our approach, using different load level (number of VDC request). At each load level, we calculate the utilization of each server resources, the number of active physical machine, and the number of rejected VDC request. Note that, we feed the placement algorithm with raw VDC request (without passing though VM/link consolidation algorithm). Table 5.4 shows the resource utilization at physical machine level. and Figure 5.2 illustrates them as chart.

Table 5.4: Resource utilization of R-VDC algorithm, second phase only

Load level	CPU utilization	Memory utilization	Storage utilization	Bandwidth utilization	Average utilization	Active PM	Rejected VMs
100	0.319867	0.290013	0.242266	0.745755	0.399475	19	0
200	0.333374	0.302259	0.252496	0.756715	0.411211	31	0
300	0.322124	0.292059	0.243975	0.72462	0.395694	44	0
400	0.332119	0.301121	0.251546	0.773564	0.414587	48	5

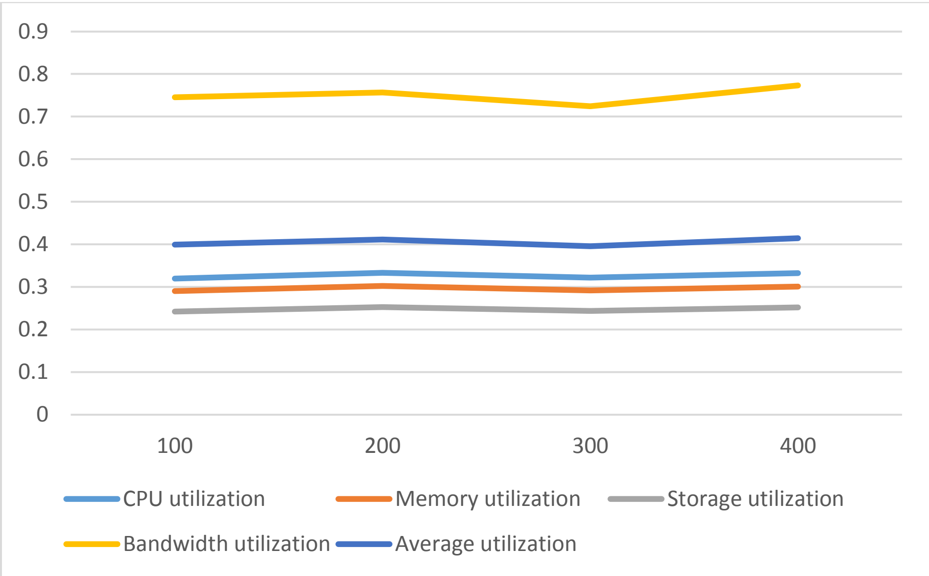


Figure 5.2: Resource utilization of R-VDC algorithm, second phase only

Next, we feed R-VDC request with modified VDC request. The result of simulation presented in Table 5.5 and illustrated in Figure 5.3.

Table 5.5: Resource utilization of R-VDC algorithm, two phases

Load level	CPU utilization	Memory utilization	Storage utilization	Bandwidth utilization	Average utilization	Active PM	Rejected VMs
100	0.416767	0.377869	0.315658	0.553886	0.416045	13	0
200	0.425733	0.385998	0.322449	0.593295	0.431869	25	0
300	0.408982	0.37081	0.309761	0.551334	0.410222	31	0
400	0.460876	0.417861	0.349066	0.592809	0.455153	33	3

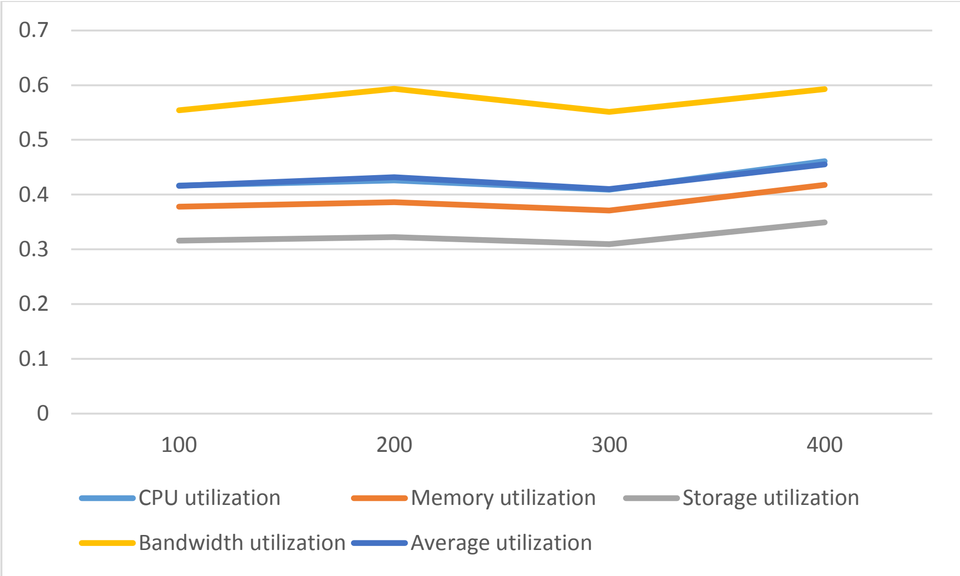


Figure 5.3: Resource utilization of R-VDC algorithm, two phases

Studying these simulation results, we note high utilization of all resources. More important, resource utilization is not affected by load level. The reason behind this is that our algorithm minimizes defragmentation, so the algorithm will keep mapping VMs to the same PM until no more space available. We note also that using the two phases together will increase the resource utilization. We also note that our algorithms starts to reject VMs even there is still PMs available.

Next, we perform a set of experiments to evaluate the effectiveness of our approach and comparing our approach with a set of placement algorithms. We choose three algorithms to compare with, namely Random Algorithm, OLRSA, and SAE Algorithms. These algorithms are part of FlexCloud framework.

In our first experiment, we use VDC request that have not been passed through our consolidation phase (we call it raw VDC request). Using the same VDC request, we run each algorithm four times, each time with different load level. Results are shown in Figure 5.4 to 5.7.

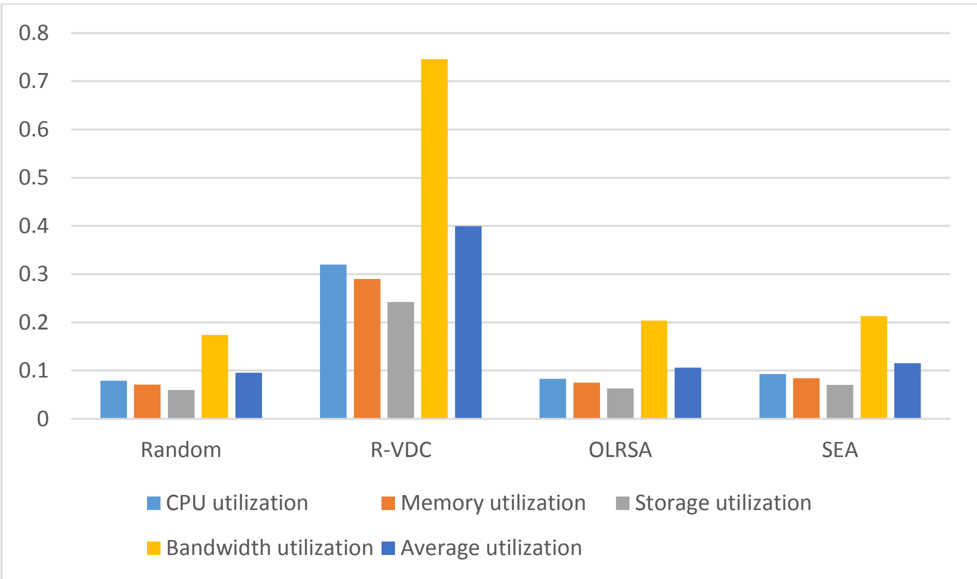


Figure 5.4: Utilization at 100 request

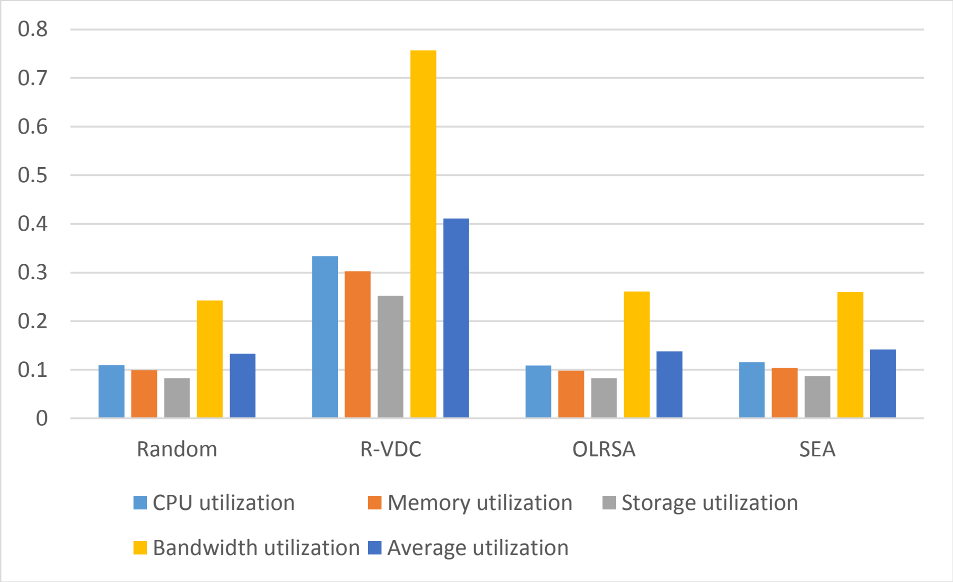


Figure 5.5: Utilization at 200 request

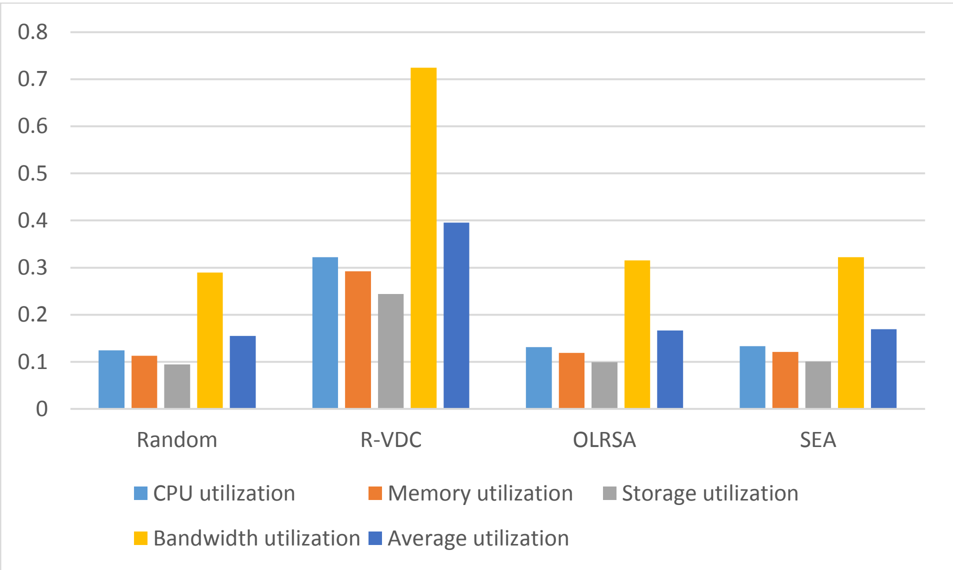


Figure 5.6: Utilization at 300 request

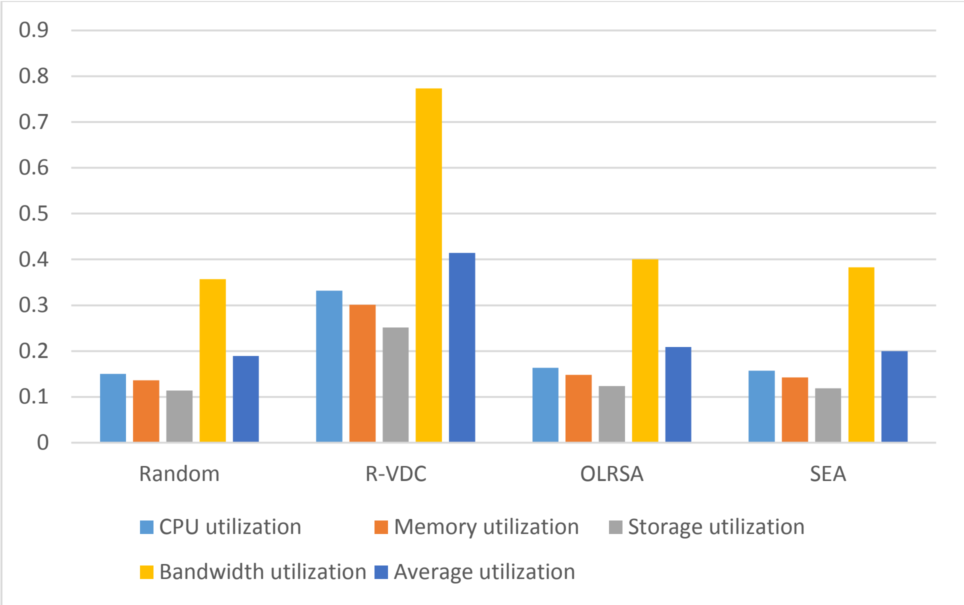


Figure 5.7: Utilization at 400 request

Next, we run the same experiment as before, but using VDC request that passed through VM/link consolidation phase. The results are shown in Figure 5.8 to 5.11

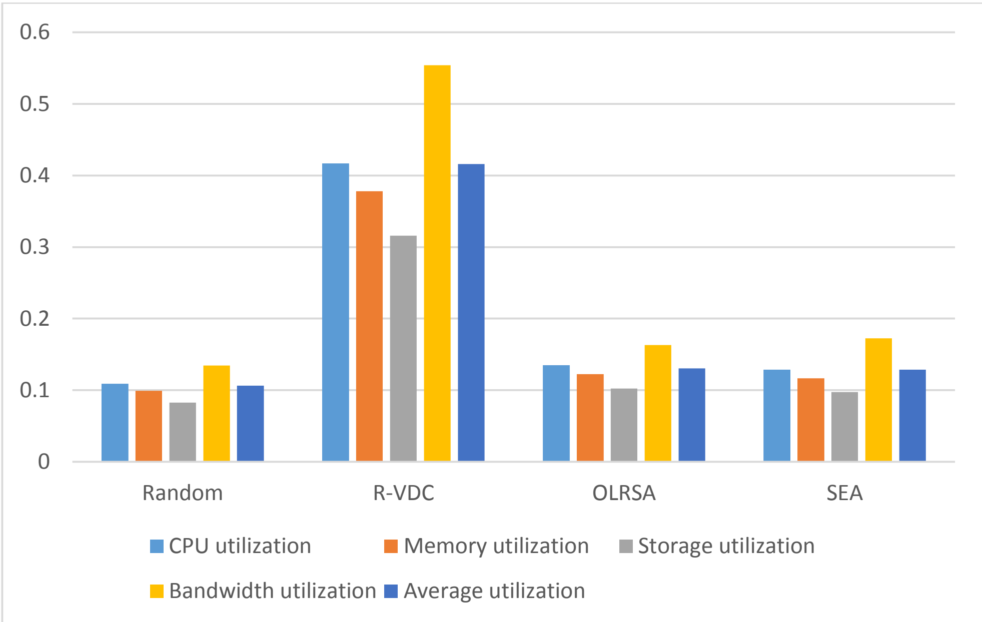


Figure 5.8: Utilization at 100 request with consolidated VDC

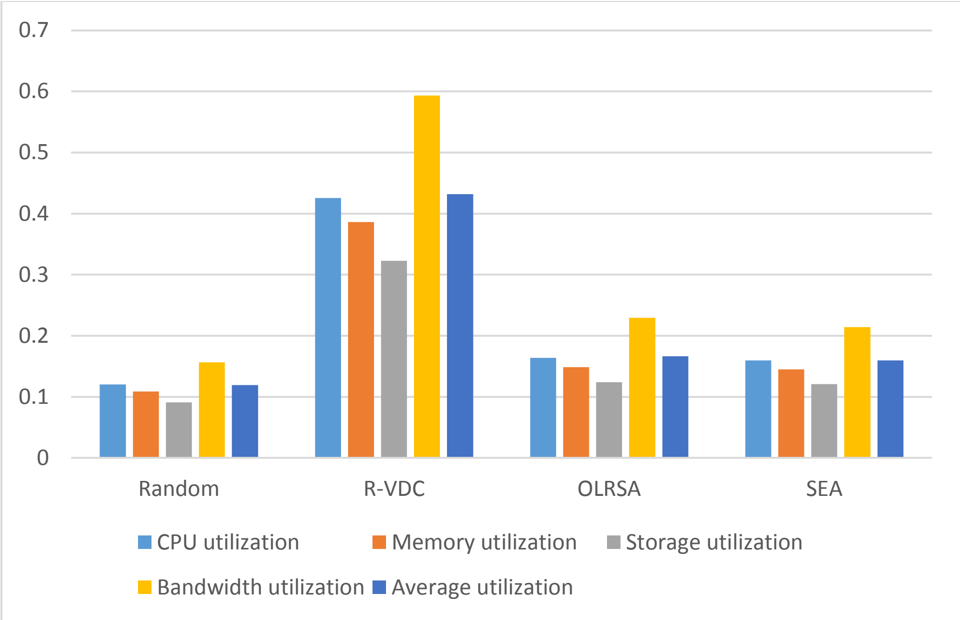


Figure 5.9: Utilization at 200 request with consolidated VDC

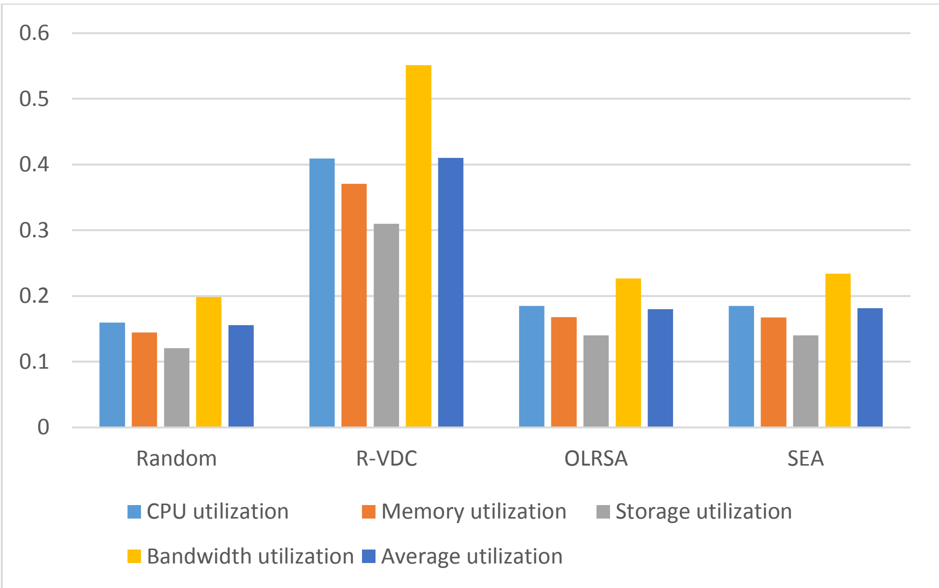


Figure 5.10: Utilization at 300 request with consolidated VDC

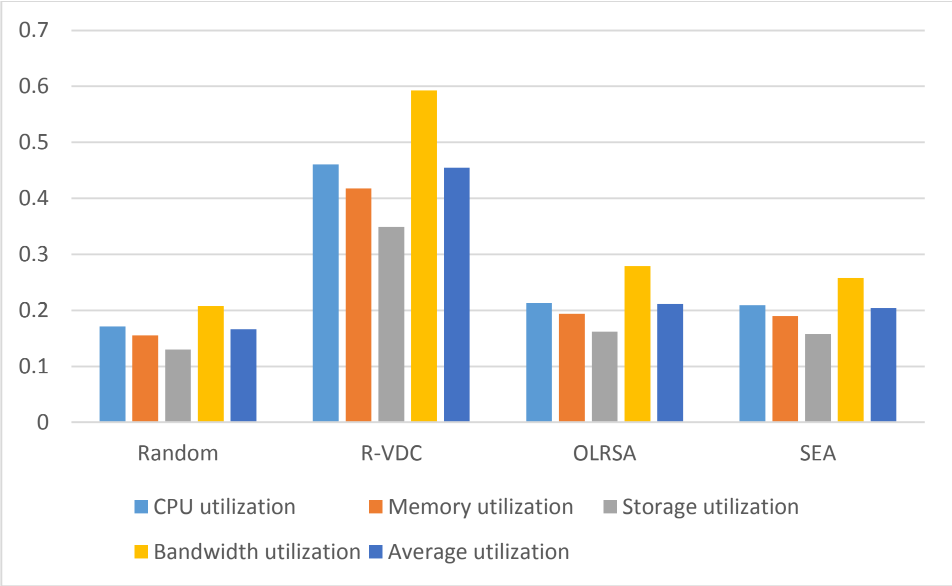


Figure 5.11: Utilization at 400 request with consolidated VDC

Studying the above results, we note that they follow the same pattern. We note that resource utilization of different resources increases as VDC request load increases. Except R-VDC algorithm, where utilization is stable and not affected by load. This is because R-VDC tries to use full capacity of the physical machine, while other algorithms will start mapping requests to a new physical machine even if the last PM still have residual capacity.

The results from last experiment can be interpreted in a different way to study the effect of consolidation phase on each algorithms. For example, Figure 5.12 show the utilization of Random algorithm with and without consolidation phase. We see that utilization of each resource are increased when using VDC request that passed though consolidation phase, We think this is because embedding one VM with high capacity is more efficient than embedding multiple VMs with the same capacity. However, bandwidth utilization is decreased. This is because consolidation phase remove some links so that decreasing bandwidth requirements. Figure 5.13 is another example that depict the same idea for OLRSA algorithm.

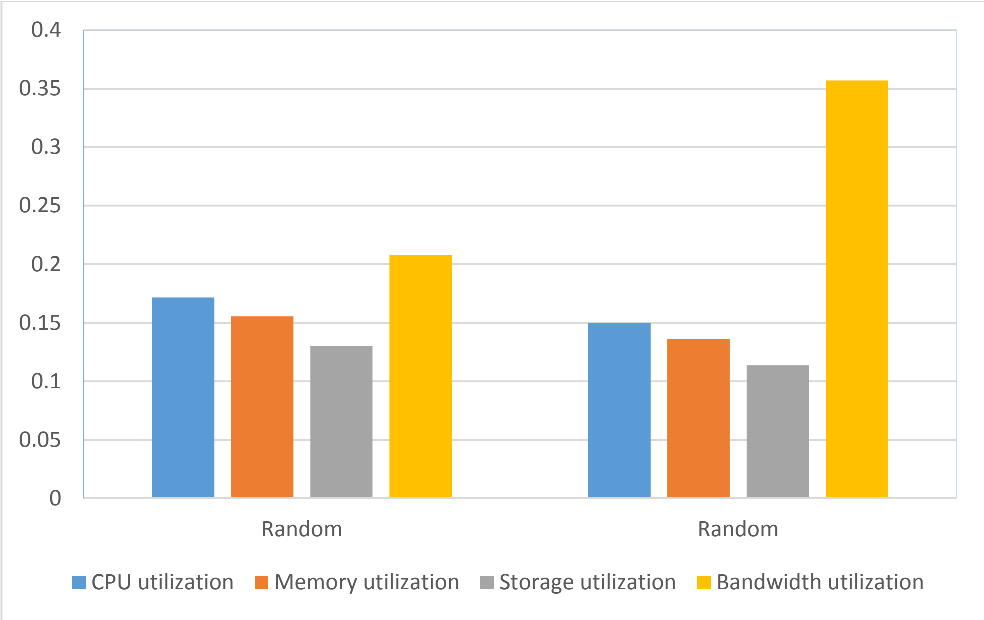


Figure 5.12: Random algorithm with/without VM/link consolidation

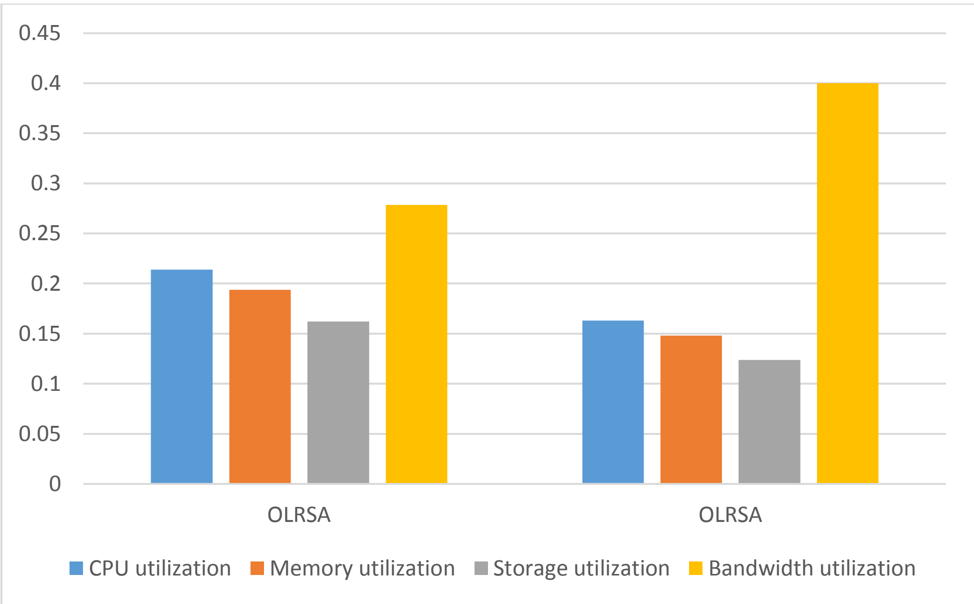


Figure 5.13: OLRSA algorithm with/without VM/link consolidation

Chapter 6 : Conclusion and Future Work

This thesis addressed the problem of application performance degradation due to contention on network resource. Many approaches been proposed to solve this problem. One of the most promising approaches is to offer both computation power and network bandwidth as bundle. This approach have its own drawback, that is, reserving bandwidth to tenant that may or may not use this bandwidth is wasting valuable resources. In the other hand, tenants will be sure that their application will run without performance degradation.

To tackle this problem which been known as virtual data center embedding (VDCE), we went through series of steps. First step was to figure out how to enable tenants to express their needs. We model the VDC request and data center resource (both physical machine and network paths) as undirected weighted graph, which can be represented as an adjacency matrix.

The process of mapping virtual machines and virtual path to physical machine and path is hard problem, so we think that may be if we could simplify tenant requests and hence will simplify next processing steps. We introduce simple heuristic algorithm that consolidate VMs and virtual links. In general, the output of this algorithm will be simple form of tenants' request. By simple, we mean that the number of nodes and links is less than the original request, while the capacity are the same. At the same time, this process helps in minimizing bandwidth requirements and consumptions.

After that, we designed an algorithm that maps virtual machines into physical machines. The design of this algorithm is based on the assumption that the data center is using fat-tree network topology. So that, if the physical like between physical machines and their access switches have enough bandwidth to satisfy VM requirements, then there is enough bandwidth in upper level switch that satisfy VMs pair requirements. However, the algorithm try to place individual VMs as close as possible to each other to save bandwidth in upper level switches.

We used simulation tools to study the performance of the proposed approach, and design multiple experiments to compare the performance and effectiveness of our approach against other proposed solution, namely, Random Algorithm, OLRSA and SAE algorithm.

Based on the simulation results, we proved that consolidation of user request is crucial part of successful embedding algorithms. We use the output of our VM/Link consolidation algorithm as input to other placement algorithms, and compare the results in both cases and find that placement algorithms perform better in case of using VDC requests that pass through the proposed VM/Link consolidation algorithms.

We also found that our placement algorithms have advantage over others from the resources utilization viewpoint. Simulation results showed that the utilization of different resources is higher than other algorithms at different data center load level. The reason behind these results is that our algorithm try to minimize defragmentation of resources, i.e., not to map VMs to new PM if current PM still have enough capacity.

The main drawback of our approach is that not considering switches in both VDC request and data center. This have two bad side effects. First, its limit the design of VDC request. Second, it limit our approach to fat tree topology. We consider this drawback to be new direction for future works.

In addition to modeling switches devices in both VDC request and at data center. I should expand the idea of consolidation/partition of VDC request into partitioning to physical machine, so that partitioning of physical data center into rack/pod. So that, instead of mapping VMs to individual physical machine, we try to map VDC request to rack or pod that have adequate resource available. This has two advantage, first it minimize the search time to locate individual physical machine and second, it keeps traffic locally, so that minimizing bandwidth consumption at higher switches level.

Finally, we should mention that, VDC embedding is a reservation process but not including any mechanism that enforce the bandwidth guarantee. It would be good idea to combine it with other to build complete solution that provides reservation service and enforces this reservation in face of tenant misuse.

References

[1] V. Jeyakumar, M. Alizadeh, D. Mazieres, B. Prabhakar, C. Kim, and A. Greenberg, “Eyeq: Practical network performance isolation at the edge,” *REM*, vol. 1005, no. A1, p. A2, 2013.

[2] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos, “Managing Performance Overhead of Virtual Machines in Cloud Computing: A Survey, State of the Art, and Future Directions,” *Proc. IEEE*, vol. 102, no. 1, pp. 11–31, Jan. 2014.

[3] M. Mishra, P. Dutta, P. Kumar, and V. Mann, “Managing Network Reservation for Tenants in Oversubscribed Clouds,” in *2013 IEEE 21st International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2013, pp. 50–59.

[4] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, “Towards predictable datacenter networks,” in *ACM SIGCOMM Computer Communication Review*, 2011, vol. 41, pp. 242–253.

[5] L. Chen, B. Li, and B. Li, “Allocating Bandwidth in Datacenter Networks: A Survey,” *J. Comput. Sci. Technol.*, vol. 29, no. 5, pp. 910–917, 2014.

[6] Q. Zhang, M. F. Zhani, M. Jabri, and R. Boutaba, “Venice: Reliable virtual data center embedding in clouds,” in *2014 Proceedings IEEE INFOCOM*, 2014, pp. 289–297.

[7] M. G. Rabbani, R. P. Esteves, M. Podlesny, G. Simon, L. Z. Granville, and R. Boutaba, “On tackling virtual data center embedding problem,” in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, 2013, pp. 177–184.

[8] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, “Secondnet: a data center network virtualization architecture with bandwidth guarantees,” in *Proceedings of the 6th International COnference*, 2010, p. 15.

[9] L. Shouxi, Y. Hongfang, L. Lemin, L. Dan, and S. Gang, “Traffic-aware VDC embedding in data center: A case study of fattree,” *Commun. China*, vol. 11, no. 7, pp. 142–152, Jul. 2014.

[10] M. F. Zhani, Q. Zhang, G. Simon, and R. Boutaba, “VDC Planner: Dynamic migration-aware Virtual Data Center embedding for clouds,” in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, 2013, pp. 18–25.

[11] D. Xie, N. Ding, Y. C. Hu, and R. Kompella, “The only constant is change: incorporating time-varying network reservations in data centers,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 199–210, 2012.

[12] J. Guo, F. Liu, H. Tang, Y. Lian, H. Jin, and J. C. S. Lui, “Falloc: Fair network bandwidth allocation in IaaS datacenters via a bargaining game approach,” in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, 2013, pp. 1–10.

[13] T. Huang, C. Rong, Y. Tang, C. Hu, J. Li, and P. Zhang, “VirtualRack: Bandwidth-aware virtual network allocation for multi-tenant datacenters,” in *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 3620–3625.

[14] P. Mell and T. Grance, “The NIST definition of cloud computing,” 2011.

[15] S. Rajan and A. Jairath, “Cloud Computing: The Fifth Generation of Computing,” 2011, pp. 665–667.

[16] “Amazon Elastic Compute Cloud (EC2).” [Online]. Available: http://aws.amazon.com/ec2/. [Accessed: 24-Mar-2015].

[17] “Eucalyptus,” *Eucalyptus*. [Online]. Available: https://www.eucalyptus.com/. [Accessed: 05-Apr-2015].

[18] “Rackspace.” [Online]. Available: http://www.rackspace.com/. [Accessed: 05-Apr-2015].

[19] “App Engine - Run your applications on a fully-managed Platform-as-a-Service (PaaS) using built-in services,” *Google Developers*. [Online]. Available: https://cloud.google.com/appengine/. [Accessed: 05-Apr-2015].

[20] “Microsoft Azure: Cloud Computing Platform & Services.” [Online]. Available: http://azure.microsoft.com/en-us/. [Accessed: 05-Apr-2015].

[21] “Google Docs.” [Online]. Available: https://docs.google.com/document/u/0/. [Accessed: 05-Apr-2015].

[22] “Dropbox - You’re invited to join Dropbox!” [Online]. Available: https://www.dropbox.com/. [Accessed: 05-Apr-2015].

[23] A. Amokrane, M. F. Zhani, R. Langar, R. Boutaba, and G. Pujolle, “Greenhead: Virtual Data Center Embedding across Distributed Infrastructures,” *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 36–49, Jan. 2013.

[24] Y. Jadeja and K. Modi, “Cloud computing-concepts, architecture and challenges,” in *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, 2012, pp. 877–880.

[25] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010.

[26] Md Hasanul Ferdaus, Manzur Murshed, Rodrigo N. Calheiros, and Rajkumar Buyya, “Network-aware Virtual Machine Placement and Migration in Cloud Data Centers.” .

[27] “Virtualization,” *Wikipedia, the free encyclopedia*. 02-Mar-2015.

[28] D. Armstrong and K. Djemame, “Towards quality of service in the cloud,” in *Proc. of the 25th UK Performance Engineering Workshop*, 2009.

[29]LeBlanc and Robert-Lee Daniel, “Analysis of Data Center Network Convergence Technologies,” Brigham Young University - BYU ScholarsArchive, 2014.

[30] I. S. Suresh and M. Kannan, “A Study on System Virtualization Techniques,” *IJARCST*, vol. 2, no. 1, 2014.

[31] “Understanding Full Virtualization, Paravirtualization, and Hardware Assist,” WP-028-PRD-01-01, 11 2007.

[32] K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, and M. Iqbal, “Quantitative comparisons of the state-of-the-art data center architectures,” *Concurr. Comput. Pract. Exp.*, vol. 25, no. 12, pp. 1771–1783, 2013.

[33] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data Center Network Virtualization: A Survey," *IEEE Commun. Surv. Tutor.*, vol. 15, no. 2, pp. 909–928, Second 2013.

[34] Yang Liu, Jogesh K. Muppala, and Malathi Veeraraghavan, "A Survey of Data Center Network Architectures." [Online]. Available: http://www.ece.virginia.edu/mv/pubs/recent-samples/Data-center-Survey.pdf. [Accessed: 11-Feb-2015].

[35] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, 2008, vol. 38, pp. 63–74.

[36] "Scheduling (computing)," *Wikipedia, the free encyclopedia*. 13-Apr-2015.

[37] F. Esposito, "A policy-based architecture for virtual network embedding," BOSTON UNIVERSITY, 2014.

[38] L. Shi, D. Katramatos, and D. Yu, "Virtual data center allocation with dynamic clustering in clouds," in *Performance Computing and Communications Conference (IPCCC), 2014 IEEE International*, 2014, pp. 1–10.

[39] R.Kanniga Devi and S.Sujan, "A Survey on Application of Cloudsim Toolkit in Cloud Computing," *Int. J. Innov. Res. Sci. Eng. Technol.*, vol. 3, no. 6, Jun. 2014.

[40] Jayshri Damodar Pagare and Nitin A Koli, "Design and simulate cloud computing environment using cloudsim," *Comput. Technol. Appl.*, vol. 6, pp. 35–42, Feb. 2015.

[41] W. Zhao, Y. Peng, F. Xie, and Z. Dai, "Modeling and simulation of cloud computing: A review," in *Cloud Computing Congress (APCloudCC), 2012 IEEE Asia Pacific*, 2012, pp. 20–24.

[42] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[43] W. W. Smari, Association for Computing Machinery, and Institute of Electrical and Electronics Engineers, Eds., "FlexCloud: A Flexible and Extendible Simulator for Performance Evaluation of Virtual Machine Allocation," 2015.

[44] "The Rise of Soft Switching Part II: Soft Switching is Awesome (tm)," *Network Heresy*. .

[45] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "FairCloud: sharing the network in cloud computing," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, 2012, pp. 187–198.

[46] P. Kumar, G. Choudhary, D. Sharma, and V. Mann, "Equinox: Adaptive network reservation in the Cloud," in *2014 Sixth International Conference on Communication Systems and Networks (COMSNETS)*, 2014, pp. 1–8.