Bournemouth University

Towards Effective Live Cloud Migration on Public Cloud IaaS

by

Ibrahim Ejdayid A.Mansour

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the

Department of Computing at Bournemouth University

November 2018

”Success is getting what you want, Happiness is wanting what you get”

Abstract

Cloud computing allows users to access shared, online computing resources. However, providers often offer their own proprietary applications, APIs and infrastructures, resulting in a heterogeneous cloud environment. This environment makes it difficult for users to change cloud service providers and to explore capabilities to support the automated migration from one provider to another. Many standards bodies (IEEE, NIST, DMTF and SNIA), industry (middleware) and academia have been pursuing standards and approaches to reduce the impact of vendor lock-in.

Cloud providers offer their Infrastructure as a Service (IaaS) based on virtualization to enable multi-tenant and isolated environments for users. Because, each provider has its own proprietary virtual machine (VM) manager, called the hypervisor, VMs are usually tightly coupled to the underlying hardware, thus hindering live migration of VMs to different providers. A number of user-centric approaches have been proposed from both academia and industry to solve this coupling issue. However, these approaches suffer limitations in terms of flexibility (decoupling VMs from underlying hardware), performance (migration downtime) and security (secure live migration).

These limitations are identified using our live cloud migration criteria which are represented by flexibility, performance and security. These criteria are not only used to point out the gap in the previous approaches, but are also used to design our live cloud migration approach, LivCloud. This approach aims to live migration of VMs across various cloud IaaS with minimal migration downtime, with no extra cost and without user's intervention and awareness. This aim has been achieved by addressing different gaps identified in the three criteria: the flexibility gap is improved by considering a better virtualization platform to support a wider hardware range, supporting various operating system and taking into account the migrated VMs' hardware specifications and layout; the performance gap is enhanced by improving the network connectivity, providing extra resources required by the migrated VMs during the migration and predicting any potential failure to roll back the system to its initial state if required; finally, the security gap is clearly tackled by protecting the migration channel using encryption and authentication.

This thesis presents: (i) A clear identification of the key challenges and factors to successfully perform live migration of VMs across different cloud IaaS. This has resulted in a rigorous comparative analysis of the literature on live migration of VMs at the cloud IaaS based on our live cloud migration criteria; (ii) A rigorous analysis to distil the limitations of existing live cloud migration approaches and how to design efficient live cloud migration using up-to-date technologies. This has led to design a novel live cloud migration approach, called LivCloud, that overcomes key limitations in currently available approaches, is designed into two stages, the basic design stage and the enhancement of the basic design stage; (iii) A systematic approach to assess LivCloud on

different public cloud IaaS. This has been achieved by using a combination of up-to–date technologies to build LivCloud taking the interoperability challenge into account, implementing and discussing the results of the basic design stage on Amazon IaaS, and implementing both stages of the approach on Packet bare metal cloud.

To sum up, the thesis introduces a live cloud migration approach that is systematically designed and evaluated on uncontrolled environments, Amazon and Packet bare metal. In contrast to other approaches, it clearly highlights how to perform and secure the migration between our local network and the mentioned environments.

Acknowledgements

This PhD thesis is the supervision of Prof. Hamid Bouchachia. My time at Bournemouth University has been influenced and guided by a number of people to whom I am deeply indebted. Without their help, friendship and support, this thesis would likely never have seen the light of day.

I would like to thank my thesis supervisor, Prof. Abdelhamid Bouchachia for his insights and guidance. I feel most fortunate to have had the opportunity to receive their support. My supervisor, Prof. Abdelhamid Bouchachia has had the greatest impact on my academic development during my thesis. He taught me how to do research, how to ask the right questions and how to answer them, how to have a clear vision and strategy.

I was also indeed fortunate to have Dr. Kendra Cooper as my second supervisor for limited time. I feel exceedingly privileged to have had her guidance and I owe her a great many thanks.

My deepest gratitude and appreciation are reserved for my wife, my brother, Ramadan Mansour and my daughters, Hiba, Arwa, Yara and Mais. Without their love, consistent support and patience, I would never have been able to produce this thesis. I dedicate this thesis to them. Thanks again to my brother and his endless support.

Contents

List of Figures

List of Tables

Abbreviations

SaaS	**S**oftware **a**s **a** **S**ervice
PaaS	**P**latform **a**s **a** **S**ervice
IaaS	**I**nfrastructure **a**s **a** **S**ervice
CaaS	**C**ontainer **a**s **a** **S**ervice
VMM	**V**irtual **M**achine **M**anager
KVM	**K**ernal **V**irtual **M**achine
QEMU	**Q**uick **E**mulator
LAN	**L**ocal **A**rea **N**etwork
VPN	**V**irtual **P**rivate **N**etwork
API	**A**pplication **P**rogramming **I**nterface
AWS	**A**mazon **W**eb **S**ervices
EC2	**E**lastic **C**ompute **C**loud
MPLS	**M**ulti**P**rotocol **L**abel **S**witching
SDN	**S**oftware **D**efined **N**etworking
NFV	**N**etwotking **F**unction **V**irtualization
NFS	**N**etwork **F**ile **S**ystem
OvS	**O**pen **vS**witch
ODL	**O**pen**D**ay**L**ight
SDNi	**SDN** controller **i**nterconnection
OFM	**O**pen**F**low **M**anager
LISP	**L**ocator **I**dentifier **S**eparation **P**rotocol
EID	**E**ndpoint **ID**endtifier
RLOC	**R**outing **LOC**ators
OOR	**O**pen **O**verlay **R**outer
UDT	**U**DP-based **D**ata **T**ransfer
SLA	**S**ervice **L**evel **A**greement

VPC	**V**irtual **P**rivate **C**loud
EPT	**E**xtended **P**age **T**able
VT-x	**V**irtual **T**echnology e**x**tension
AMI	**A**mazon **M**achine **I**mage
EBS	**E**lastic **B**lock **S**torage
LDAP	**L**ightweight **D**irectory **A**ccess **P**rotocol
xTR	ingress/egress **T**unnel **R**outers
REST	**RE**presentational **S**tate **T**ransfer
MR	**M**ap-**R**esolver
MS	**M**ap-**S**erver
GUI	**G**raphical **U**ser **I**nterface
D-H	**D**iffie-**H**ellman
AES	**A**dvanced **E**ncryption **S**tandard
RTT	**R**ound **T**rip **T**ime
DBT	**D**ynamic **B**inary **T**ranslation
APT	**A**dvanced **P**ersistent **T**hreats
NIC	**N**etwork **I**nterface **C**ard
ARP	**A**dress **R**esolution **P**rotocol

Dedicated to my parents, my beloved wife, lovely daughters and my supportive brother

Chapter 1

Introduction

There is a growing trend in adopting cloud computing services. The IDC's (International Data Corporation) Worldwide Semiannual Public Cloud Services Spending Guide reported that cloud services were predicted to increase from $70 billion in 2015 to more than $203 billion in 2020. This significant increase is about sevenfold of overall IT spending growth [10]. In 2017, RightScale conducted cloud computing trends survey in which 1,002 IT professionals at large and small enterprises were interviewed about their adoption of cloud infrastructure and related technologies [11]. 85 percent of enterprises deploy multi-cloud services, up from 82 percent in 2016. However, private cloud adoption decreased from 77 percent to 72 percent as enterprises focus more on public cloud services. Despite the notable upwards trend, there are still concerns about cloud computing security, interoperability and managing cost [11, 12]. However, the security concerns fell from 29 to 25 percent in comparison with 2016.

In 2013, Amazons US-EAST availability region remained unavailable for 59 minutes, resulting in users in U.S.A. and Canada not accessing Amazon.com and Audible.com. The reported loss was about $1,100 in net sales per second [13]. Moreover, Google reported that a 500ms delay in website page loading caused a 20% drop in traffic and revenue [14]. In 2018, Lloyd's of London and AIR Worldwide provides report and estimates on the losses from a major cloud services outage. According to this report, a cyber attack impacted the operations of one of the top three public cloud providers in the U.S. for three to six days. The total losses are estimated up to $19 billion. Only $1.1 to $3.5 billion can be insured, leaving organizations left to cover the rest of the costs [15].

If customer services had been able to rapidly become available by migrating to another provider, then the consequences would have been less disastrous and resources could have been saved. There are a number of advantages offered by live migration [16]:

1. High flexibility to change service providers, thereby, alleviating vendor lock-in.
2. Low-price services offered by certain providers.
3. Offering service continuity in case of ceasing due to various reasons including natural disasters.
4. Reducing latency by connecting cloud users to the nearest datacentre, regardless of the provider.
5. Choice to process sensitive data on a private trusted cloud, while processing less sensitive on a public cloud.
6. Borrowing resources from other providers in case of over-utilization or limited resources at the current provider.

In order to introduce the challenges of live migration, we organize the rest of this chapter as follows. Section 1.1 is a background on cloud computing IaaS and its related issues. Section 1.2 introduces cloud computing interoperability issues and benefits. Section 1.3 reviews the possible solutions to achieve cloud interoperability. Section 1.4 presents the aim and research questions of the thesis and it shows how the thesis reflects on these research questions. Section 1.5 discusses the major contributions of the thesis. The structure of the thesis is presented in Section. 1.6. A list of the publications on which the thesis is based is shown in Section. 1.7.

1.1 Background

Live cloud migration of VMs at IaaS is an active research area [10], working to overcome the lack of cloud interoperability among providers. Zhang et al. [12] conducted a survey on the lack of interoperability within the cloud at the IaaS level, open source cloud projects (i.e., OpenStack and OpenNebula), cloud standards, and a user-centric solution called Xen-Blanket [17]. The survey presented taxonomy of cloud infrastructure interoperability. However, the survey did not include any live cloud migration criteria at cloud infrastructure to assess previous live cloud migration approaches. Similarly,

Nadjaran et al. [16] conducted a broad survey on cloud interoperability for all levels, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) within the cloud and related open source projects (i.e. RESERVOIR, mOSAIC and OpenStack [18]). However, the survey did not evaluate any user-centric approach to facilitate interoperability or other approaches that could support live migration of VMs and lacked some important references such as [19] related to Ubuntu OpenStack. In 2015, OpenStack interoperability press announced that 32 companies signed up to adhere to OIL guidelines. Moreover, being one of the widely deployed open source cloud projects, OpenStack is supported by about 500 companies and 23,000 individuals across over 150 countries [19].

Much work has been done to provide live migrations of VMs to and within the cloud with minimum service interruption [20–22]. Live migration often requires the following [23]: memory state transfer between anonymous hosts, access of VMs to the storage at the destination host, without sharing storage among source and destination hosts; and access of the VM to the host's LAN at the destination without sharing the LAN. Virtualization is the foundation of the cloud IaaS. It allows cloud users to exploit multi-tenant resources (compute, network and storage) from a secure Cloud IaaS [21]. *Virtualization is the conversion of a physical machine to individual isolated spaces (VMs) that can be used by multiple users as per their needs. The isolation and resources provision is provided by a hypervisor* [24]. *Nested virtualization runs a VM inside another VM. The outer VM is run on the physical hardware, whereas the insider VM is run in the outer VM* [25].

Nested virtualization has been the main solution for separating VMs from underlying hardware in both legacy system and cloud IaaS despite the fact that it is not enabled on public cloud IaaS [17, 22, 26]. It was first proposed and refined in the 1970s [17] and now has become available on most platforms such as VMware [27]. In the context of this thesis, Citrix Xenserver [28] and VirtualBox hypervisors have been tested to evaluate thier support of running nested virtualization, but both hypervisors failed to enable a second virtualized layer. Public cloud IaaS is often described as a heterogeneous environment due to the fact that each cloud provider has their own hypervisor. Providers such as Amazon EC2 and Rackspace use the hypervisor Xen; while Fractus and Google Compute Engine rely on KVM. Windows Azure, on the other hand, uses the Microsoft hypervisor, Hyper-V [2, 20].

Despite that many providers leverage the same hypervisors for virtualization (e.g. Google and HP both use KVM), live cloud migration of VMs between those providers is still

challenging [20]. Every provider has been developing their own APIs and proprietary features to their selected hypervisor. This has made it difficult for cloud users to live migrate VMs to other providers - one aspect of vendor lock-in with substantial consequences [16].

Security and interoperability are the main challenges to cloud computing [11, 12]. However, according to [10], there are other challenges to cloud computing, such as resource management and scheduling, sustainability and usability. Despite the importance of these challenges, they are out side the scope of this research.

The main focus of this research is on a number of challenges that have direct impact on live cloud migration. They are as follows [10].

1. Heterogeneity and interconnected clouds

   It is still challenging to design an independent cloud platform that integrates and manages heterogeneity at all three levels, IaaS, PaaS and SaaS. Another challenge is related to the development of application software and APIs that are compatible with heterogeneous resources and that can be open-source and independent of the underlying hardware. However, cloud platforms still deliver their services independently. On the other hand, cloud interconnection is very demanding as it is about interfacing various clouds to enable communication [29].

   Although there are a number of standardization initiatives, the existing Cloud services are heterogeneous and not standardized [16]. Among such initiatives, Open Grid Forum's (OGF), Open Cloud Computing Interface (OCCI), Storage Networking Industry Association's (SNIA) Cloud Data Management Interface (CDMI), Distributed Management Task Force's (DMTF) Cloud Infrastructure Management Interface (CIMI), DMTF's Open Virtualization Format (OVF), IEEE's InterCloud and National Institute of Standards and Technology's (NIST) Federated Cloud [10, 16].

2. Security and privacy

   They are among the major challenges in cloud computing in terms of securing the providers' as well as the users' data and maintaining confidentiality, integrity and availability. Today providers maintain confidentiality by encrypting data before the storing process [10]. However, encryption affects the support of query evaluation at the provider side. To address such issue, encrypted database designs allowing SQL queries over encrypted data and indices associated with the encrypted

data are being developed [10, 30, 31]. There is another issue associated with confidentiality of data when cloud-based applications are utilized (e.g., applications for accurate social services, enhanced healthcare and discovering phishing) that access data over various sources across multiple domains. A major challenge of such applications is to preserve privacy, as data mining tools with across different domains may expose more personal information than needed, therefore deterring organizations to share their data [32]. Many approaches aiming to maintain data confidentiality assume that any legitimate user, who has the decryption key, can access the whole data. This assumption is not always correct because different users have various level of authorization. As a result, better solutions are needed to encrypt data according to their sensitivity [33].

In terms of integrity, various approaches including digital signatures and Provable Data Possession allow discovering of unauthorized modifications of data stored the cloud provider. Testing the integrity of stored data by authorized users is only one aspect of integrity. When multiple writers and queries are performed on data, this leads to change data dynamically which cause disruption to data integrity. A number of solutions have been investigated deploying authenticated data structures (deterministic approaches) or insertion of integrity checks (probabilistic approaches) [34] to maintain data integrity. Moreover, data security and privacy remain a concern with rules and regulations being increasingly imposed by different governments [35].

With respect to the availability, SLAs are predfined between the cloud provider and the cloud user. However, advanced cyberattacks in the cloud represent a serious breach that may compromise confidentiality, integrity and availability. The Advanced Persistent Threats (APTs) are an emerging class of cyberattacks. They are well-organized, technically-advanced, stealthy and persistent. Red October and Flame are examples of APTs. It is estimated that there will be a 50% increase of security budgets to avoid such a threat [36].

3. Data management

   It is one of the key selling points of Cloud computing is offering reliable and elastic storage. Also, cloud providers offer reliability and availability through multiple copies that are maintained transparently, along with disaster recovery with storage that can be replicated in various regions. A number of storage abstractions are also offered to accommodate certain needs. File-based storage (Amazon Simple Storage Service (S3), Azure File), block storage services (Azure Blob,

Amazon Elastic Block Store (EBS)) of a disk volume, and logical HDD (Hard Disk Drive) and SSD (Solid-State Drive) disks that are attached to VMs are common examples. Network latencies and bandwidth between VMs, and from VM to storage can be variable, causing bottlenecks in the local datacentres and across WAN. Different approaches such as Software Defined Networking (SDN) and Network Functions Virtualization (NFV) can provide better mechanisms to enhance the network and bandwidth issues [10].

4. Networking

   cloud networking is quite complicated and modern cloud datacentres face similar challenges to build the Internet due to their data capacity and the number of cloud users. The virtualized environment of these centres is also provoking different issues that have always existed within multi-tenant infrastructure. For example in terms of scalability, VLANs (Virtual Local Area Network) are limited to 4,096 segments. Thus, the scale is limited to approximately 4,000 tenants. In terms of IP address version 4, cloud providers such as Microsoft Azure reported that they ran out of addresses. This means that cloud networking needs technologies that offer high performance, robustness, reliability, scalability and security. Technologies such as SDN and NFV provide potential solutions of networking issues in the cloud. They can maintain traffic engineering mechanisms within and across different cloud providers' networks [37].

1.2 Cloud interoperability benefits and issues

One of the greatest challenges facing longer-term adoption of cloud computing services is interoperability, as this is necessary to provide cloud providers' services such as cloud federation, servers underutilization, maintenance and cease operations [13, 38]. To provide these services, live VMs migration is required within and between the clouds. Cloud interoperability can be viewed as a multi-layered model, where every layer has to interoperate with the next layer and with its counterpart in another provider. Cloud interoperability at the Platform as a Service (PaaS) and Software as a Service (SaaS) levels depends on the Infrastructure as a Service (IaaS) level. This indicates that interoperability at the IaaS level is of key importance [12].

Interoperability and portability have many aspects to a number of different components in the cloud architecture, each of which needs to be considered in its own right. These

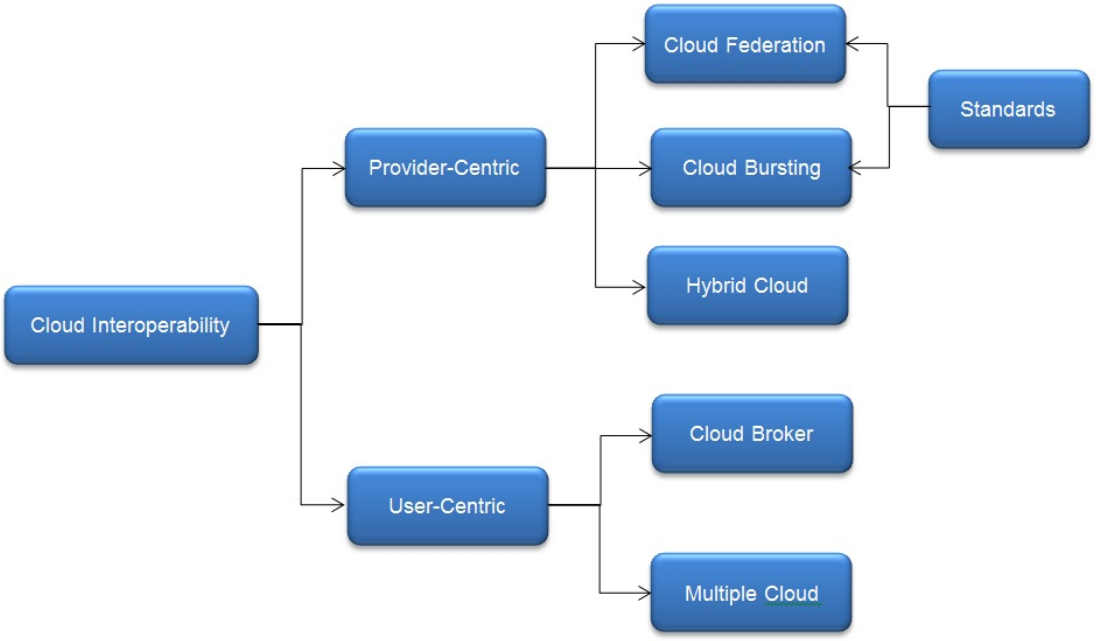


Figure. 1.1 A taxonomy on cloud interoperability approaches [1]

include standard interfaces, portable data formats and applications and international standards for service quality and security. The transparent provision, management and configuration of cross-site virtual networks to interconnect the on-premise cloud and the external provider resources is still an important challenge that is slowing down the full adoption of this technology [39]. Challenges in this area concern how to go beyond the minimum common denominator of services when interoperating across providers (and thus enabling richer Cloud applications); how to coordinate authorization, access, and billing across providers; and how to apply inter-cloud solutions in the context of Fog computing, IoT and other emerging trends [10].

1.3 Cloud interoperability approaches

Various approaches have been proposed to improve cloud interoperability for all the three levels (IaaS, PaaS and SaaS) [1, 12, 16]. Figure 1.1 illustrates a taxonomy organized around provider-centric and user-centric approaches [16].

1.3.1 Provider-centric approaches

Provider-centric approaches rely on the provider's agreement to adopt specific standards to achieve a specified level of interoperability. The development and widespread adoption of a set of standards is a long term vision for the community to support cloud federation, cloud bursting and hybrid clouds [16]. Cloud federation may be facilitated through network gateways that connect public, private clouds and/or community clouds, creating a hybrid cloud computing environment. Cloud bursting uses of a set of public or private cloud-based services to augment and handle peaks in IT system requirements at start-up or during runtime [40]. Hybrid clouds use of a combination of private and public clouds [41].

1.3.2 User-centric approaches

As standardization efforts proceed, alternative user-centric approaches to achieve cloud interoperability are being proposed to come up with more immediate and practical solutions. User-centric approaches do not rely on a provider's (standards based) agreement because cloud users either rely on their own in-house IT personnel or a third party (cloud broker) to achieve interoperability. There are two main possibilities. The first is a cloud broker, which provides a single interface through which users can access and manage cloud services across multiple providers [42]. The second is a multi-cloud, in which users may develop a separate layer to handle heterogeneity in cloud environments [12]. For example, a user may require deploying an adapter layer to communicate with different APIs or a cloud application may need an abstraction library, such as jcloud and LibCloud libraries [16].

1.4 Research questions

This research investigates effective live cloud migration for VMs at cloud IaaS with minimal service interruption. In particular, it attempts to address the following research questions:

Q1: What are the key challenges and factors to successfully perform live migration of VMs across different cloud IaaS?

This research addresses Q1 by introducing the live migration benefits to the cloud users and how cloud interoperability helps reduce the impact of live migration challenges, such as heterogeneous virtualization and networking complexity. Moreover, the key factors are clearly identified as the live cloud migration criteria to assess the effectiveness of any live cloud migration approach.

Q2: What are the limitations of existing live cloud migration approaches and how to design efficient live cloud migration using up-to-date technologies?
The research addresses Q2 by analysing previous live cloud migration approaches based on live cloud migration criteria to define the limitations of these approaches. Furthermore, a novel live cloud migration approach, LivCloud is designed based on these criteria to address the limitations resulted from the analysis.

Q3: Given a new live migration approach, how can it be assessed on different public cloud IaaS?
The research addresses Q3 by implementing and configuring LivCloud on two different cloud IaaS, Amazon EC2 and Packet bare metal cloud. Also, benchmark tools are carefully selected to help the assessment of approach's network throughput, network latency, CPU overhead and disk I/O performance.

1.5 Contributions

The major contribution of this research is design and develop a novel live cloud migration approach that live migrates VMs across various public cloud IaaS. Furthermore, answering the research questions helps highlight the other research contributions, which are distilled as follows.

1. Answering the first research question by identifing key factors to maintain cloud interoperability and benefits that are maintained from achieving cloud interoperability. Also, designing live cloud migration criteria at cloud IaaS.

2. These criteria have been effectively materialized to assess previously proposed live migration approaches. The assessment results have significantly contributed

to designing a fully functional prototype, **LivCloud**. This is the answer of the second research question. LivCloud is implemented into two stages. Firstly, the basic design which shows the two fundamental features to establish the live migration process, nested virtualization and network connectivity. Secondly, the enhancement of the basic design guarantees improving network throughput, maintaining existing VMs connections, reserving required migration resources and securing the migration process.

LivCloud is the live cloud migration approach that:

- is flexible, supporting multiple Operating Systems (OS) in comparison to current solutions (e.g., [20, 21]).
- provides rapid response time performance in comparison to current solutions (e.g., [20–22]).
- provides secured, private migration (e.g., Amazon VPC VPN IPsec peering & OpenSwan VPN IPsec [43, 44]).

3. The implementation of LivCloud in a test environment using open source applications and protocols is answering the third research question. Some of these technologies have never been used in this context, such as Cisco OpenFlow Manager and Zodiac OpenFlow Switch. Also, open-source technologies should help introducing LivCloud as a user-friendly system in terms of configuring and navigating the system's components. Moreover, the approach has been empirically evaluated in the test environment using a comprehensive collection of studies. This evaluation process provides a collection of repeatable benchmark migration tests, results for the community to adopt.

1.6 Structure of the thesis

The structure of this thesis is as follows:

- *Chapter 2* summarises the related work on live migration of VMs at both legacy and public cloud IaaS. It highlights the live cloud migration criteria that serve as a reference for requirements of live migration. The criteria are flexibility, performance and security.

- *Chapter 3* introduces LivCloud guided by the set of requirements introduced in Chapter 2. Technologies and applications that are needed to fulfil the requirements have been clarified in this chapter. Also, it illustrates in particular LivCloud to enable live migration of VMs across various hypervisors within LAN environment. Moreover, an evaluation study is proposed to highlight the effectiveness of LivCloud design.

- *Chapter 4* discuses the implementation of LivCloud basic design stage on Amazon EC2 instances, m3.2xlarge and c4.2xlarge in particular. Furthermore, it highlights alternative solutions to successfully implement this stage.

- *Chapter 5* shows the results of testing the basic stage on Packet bare metal cloud using Linux Bridge, KVM networking NAT and OpenSwan VPN IPsec. Packet has an advantage over Amazon, which is nested virtualization enabled by default.

- *Chapter 6* discuses the implementation of LivCloud enhancement stage on Packet. This chapter evaluates this stage using two different scenarios. Both scenarios architecture are explained in more detail in this chapter.

- *Chapter 7* concludes this thesis and proposes a better scenario to implement a live cloud migration approach without any downtime and better flexibility, performance and security.

1.7 List of publications

A number of publications have emerged from this research work with some of them still under review.

- The following publication is based on *Chapter 2*:

  - Ibrahim Mansour, Reza Sahandi, Kendra Cooper and Adrian Warman. 2016. Interoperability in the Heterogeneous Cloud Environment: A Survey of Recent User-centric Approaches. In proceedings of the ACM International Conference on Internet of things and Cloud Computing (ICC2016).

- The following publication is based on *Chapter 3*:

  - Ibrahim Mansour, Kendra Cooper and Hamid Bouchachia. ”Effective Live Cloud Migration”. In proceedings of The IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud 2016).

- The following publication is based on *Chapter 4*:

  - Ibrahim Mansour, Hamid Bouchachia and Kendra Cooper. ”Exploring Live Cloud Migration On Amazon EC2”. In proceedings of The IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud 2017).

- The following publication is based on *Chapter 5*:

  - Ibrahim Mansour and Hamid Bouchachia. ”Interoperability and Live Migration of VMs across Heterogeneous Cloud IaaS”. Journal of Network and Computer Applications, Elsevier. *Under review*

- The following publication is based on *Chapter 6*:

  - Ibrahim Mansour and Hamid Bouchachia. ”Enhancement of LivCloud for live cloud migration”. In proceedings of The IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC 2018). *Accepted*

Chapter 2

Literature review and live cloud migration analysis

This chapter summarises previous five cloud migration approaches, explains the design of live cloud criteria based on the related work, analyses certain popular approaches based on these criteria. This analysis helps design our live cloud migration approach, LivCloud [7]. Furthermore, a number of technologies related to cloud computing is discussed in the context of live cloud migration.

This chapter fulfils and answers the second research question that aims to state the limitations of the existing approaches. The organization of this chapter is as follows. Section 2.1 provides an elaborated description of the prominent live cloud migration approaches. Section 2.2 explains the design of live cloud migration criteria. Section 2.3 introduces an analysis of certain popular live cloud migration approaches based on these criteria. Section 2.4 discuses live migration in technologies related to cloud computing, containers, fog computing and Software Defined Networking. Finally, Section 2.5 concludes the chapter.

2.1 Approaches to live cloud migration

The literature review reveals that there are a number of approaches that aim to achieve live cloud migration at public cloud IaaS. In the following, we discus some of them to give a clearer insight.

In [20], a user-centric approach is introduced which is implemented on top of a number of cloud providers, including Amazon EC2, Rackspace and HP Cloud. It uses nested virtualization (Xen-Blanket [17] that overcomes cloud heterogeneity. Xen-Blanket leverages the paravirtualization (PV-on-HVM) drivers on Xen. However, Xen paravirtualization cannot run unmodified operating systems (i.e., Windows) [17]. Paravirtualization is an enhancement of virtualization technology in which a guest OS is recompiled prior to installation inside a virtual machine. Paravirtualization allows an interface of various hardware drivers that can differ somewhat from that of the underlying hardware to be exposed to the virtual machine [45].

The approach achieves relatively acceptable performance, about 1.4 seconds migration downtime [17]. The downtime is the time when services hosted on a VM become unavaliable as the VM is being moved to a new physical host [46]. Disk I/O drivers overhead caused by Xen-Blanket reached 30%, which may affect the physical machine and the other VMs residing on that machine [47]. Also, a security mechanism is not used during the process, so it opens the system to security attacks. As a result, the transmission channel is insecure and data flow is vulnerable to attacks, such as, ARP/ DHCP/DNS poising and IP/route hijack [48]. The approach relies on Xen as its nested virtualization platform, which has a number of issues. The Xensploit [48] tool is developed to execute man-in-the-middle attack during VM migration. It can modify the sshd memory segment to bypass sshd authentication. With such a tool, VM might be accessed and the system confidentiality and integrity may be compromised.

Another user-centric approach in [21] is implemented on OpenStack-based infrastructure. It uses a virtual switch and a Linux container (LXC) to live migrate nested VMs within the cloud. The approach cannot run on a variety of OS (i.e., Windows) because the containers (LXC) are Linux-based [49]. The study claims migrating a running application between the approach's infrastructure and Amazon within a few minutes and without any downtime [21]. The nested VMs in the study have a 3.2 GB virtual disk, which is migrated using OpenStack block migration. The disk size is not practical and small to run full Linux or Windows operating systems [7].

The last user-centric approach is presented in [22]. It is HVX hypervisor that can run unmodified operating systems (i.e., Windows). It is similar to VMware because both virtualization platforms use binary translation. Binary translation transfers the VM instructions directly to the underlying system and dynamically converts them to native x86 during runtime. However, the lack of a popular open-source binary translation hypervisor has allowed other approaches such as paravirtualization to be more popular

[20, 50]. The approach manages to incorporate various virtualization hypervisors, such as QEMU, Xen paravirtualization, KVM and VMware ESX. It can decouple the VM from underlying hardware [22]. This approach is the only one to run on a modified OS (Linux) and an unmodified OS (Windows). It is, nevertheless, seen as proprietary and it cannot be evaluated because it is protected by End User Agreement (EUA) [50]. It is described as robust and reliable because it leverages binary translation to achieve a better performance in a nested virtualization environment [22]. Nevertheless, many experts do not agree with the performance statement, as this technique imposes extra overhead on the guest kernel [20, 51].

In [52], a provider-centric approach is designed and evaluated in a controlled environment. It needs the provider's agreement to be implemented on their IaaS. It introduces Dichotomy which uses a new nested virtualization technology (ephemeral virtualization). This technology transfers control of VM between a layer one hypervisor (the cloud provider's hypervisor) and a second lightly modified hypervisor using memory mapping techniques. Dichotomy is implemented and tested in QEMU-KVM [52]. Dichotomy cleanly splits the role of the hypervisor into two parts, the hyperplexor and featurevisor. The hyperplexor is controlled by the cloud provider. It is a secure stable hypervisor designed only to multiplex physical hardware and support featurevisors. A featurevisor is a lightly-modified hypervisor that runs on top of the hyperplexor to allow cloud users managing their VMs [52].

There are other examples from literature review that aim to achieve live cloud migration using SDN technologies such as OpenFlow protocol [20]. In [53], an SDN architecture named, LIME, is introduced to live-migrate VMs and virtual switches. It is built on Floodlight controller. It runs and clones the virtual switches on multiple physical switches simultaneously. If this process is not implemented correctly, it may lead to services corruption. This architecture needs the provider's agreement to be implemented on top of public cloud IaaS. In [20], an interesting approach is discussed previously which is implemented on top of a number of cloud providers, including Amazon EC2, Rackspace and HP Cloud. It is claimed that OvS was used in, but without any details.

Another approach proposed in [54] uses an open LISP implementation for public transportation based on Open Overlay Router with an SDN controller, OpenDayLight. This approach is implemented on an emulated environment, GNS3 [55]. The real challenge is how to implement such design on uncontrolled environment, such as Amazon EC2 because the provider's networking system is highly complicated [56]. Also, networks are hard to manage because their configurations change during VMs re-instantiation on

the new location. In [57], migration of a VM cluster to various clouds is suggested based on different constraints such as computational resources and better economical offerings. It is designed based on SDN OpenFlow protocol and allows VMs to be paired in cluster groups that communicate with each other independently of the cloud IaaS. It separates the VM internal network from the cloud IaaS network. Consequently, VMs can be migrated to different clouds overcoming network complexity such as static IPs. The design also adopts SDN architecture for rerouting traffic when VMs relocation migration occurs. The design is evaluated on the OpenStack environment.

Finally, virtual network migration is designed and tested on the Global Environment for Networking Innovation (GENI) [58, 59] which is Wide-Area SDN-enabled infrastructure. The migration in this study is the process of remapping the virtual network to the physical network to dynamically allocate the resources during migration, manage hosts connected to the virtual network and minimize packet loss. However, maintaining transparent migration to the users and the running applications is still challenging.

In the following we provide a more elaborate description of some prominent approaches. These approaches are explained in the table below.

Reference	**Year of publication**	**Virtualization technology**	**Approach type**	**Description**
Inter-Cloud Mobility of Virtual Machines [23]	2011	Legacy nested virtualization	Provider-centric	The live migration of VMs with their virtual disks is carried out between three open-source clouds' IaaS within RESERVOIR project [22]. The approach has not been tested on an uncontrolled environment such as Amazon EC and Google Compute Engine [10, 20, 22].
RetroVisor: Nested Virtualization for Multi IaaS VM Availability [60]	2013	Legacy nested virtualization	Provider-centric	It runs VMs across various hypervisors within a controlled environment, which is less challenging than public cloud IaaS [10].

VirtualWire for Live Migration Virtual Networks across Clouds [61]	2013	Paravirtualization (Xen-Blanket)	User-centric	It leverages Xen-Blanket as its nested virtualization technique to live migrate VMs with its networks between Cornell University's IaaS and Amazon's IaaS. It inherits the same limitations of supercloud approach, which is discussed in Section 2.1.1.
Inception: Towards a Nested Cloud Architecture [26]	2013	Legacy nested virtualization	Provider-centric	It introduces a nested IaaS for live migration of VMs across cloud IaaS (Amazon & Rackspace). The design is relatively complicated, because the migrated VMs need at least three network cards and public IP addresses. It is a provider-centric approach that needs the cloud provider's agreement to be deployed.
Time-Constrained Live VM Migration in Shared-Nothing IaaS-Clouds [62]	2014	Legacy nested virtualization	User-centric	It introduces MigrateFS that live migrates VM with its disk within the same administration domain and predicts the resources required by the process. It envisions any potential failure by monitoring the network congestion. However, It is configured and implemented in an uncontrolled environment within one datacentre not across various datacentres.

Software Defining System Devices with the Banana Double-Split Driver Model [63]	2014	Paravirtualization (Xen-Blanket)	User-centric	It is similar to VirtualWire [61] in terms of the design. Both achieve migration downtime of 1.4s. It inherits the same limitations of supercloud approach, which is discussed in Section 2.1.1.
ViNO: SDN Overlay to Allow Seamless Migration Across Heterogeneous Infrastructure [64]	2015	Customized legacy virtualization	User-centric	ViNO connects OpevSwicthws and VMs using an overlay network based on VxLAN encapsulation.It creates VMS by customizing APIs calls to the underlying cloud IaaS. ViNO can migrate Linux services across VMs with low downtime, but migrating Windows services is not clear if this approach is capable of performing this migration.
Minimizing Live VM Migration Downtime Using OpenFlow based Resiliency Mechanisms [65]	2016	Legacy virtualization	Provider-centric	This study proposes several networking architectures based on OpenFlow features including stateful forwarding. Theses architectures aim to fast restoration of network connectivity of the migrated VMs. It is a provider-centric approach that needs the cloud provider's agreement to be deployed.

SDN-based IaaS for Mobile Computing [66]	2017	Legacy virtualization	Provider-centric	An IaaS framework with regional datacentres for mobile clouds is proposed in this study. The framework is designed based on software-defined networking (SDN) to address impacts on QoS during mobility by serving mobile user via the optimum datacenter. The testbed is developed and implemented based on Mininet [67] and KVM hypervisor. It is a provider-centric approach that needs the cloud provider's agreement to be deployed.

Table. 2.1 Summary of various related work

2.1.1 A proposal for interconnecting the clouds (Supercloud project)

In [2], an architecture to interconnect various cloud providers through an abstract layer is explained. Vendor lock-in is still out of reach of cloud users. Public cloud providers continue offering different interfaces for the same types of resources services. Different providers may run different virtual machine hypervisors with incompatible virtual machine images, different hardware with slightly different instruction sets and heterogeneous storage interfaces. As mentioned in Chapter 1, there have been calls for standardization to agree on a particular hardware and software platform. Networking faced similar issues in the 1960s to achieve interconnectivity between various operating system. The adoption of the layered design, The Open Systems Interconnection model (OSI model) that based on the Internet protocol (IP) layer helped abstract network hardware and applications and encouraged different networking provider such as Cisco and Juniper to develop unique services on top of this abstraction layer [68]. This abstraction

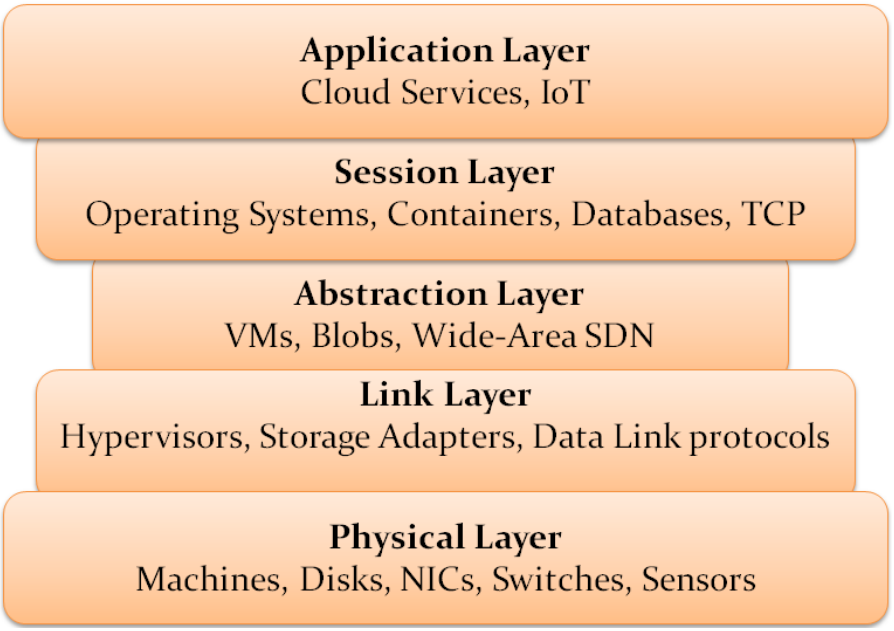


Figure. 2.1 Interconnected cloud stack architecture [2]

layer is thin in comparison to the other layers because it provides minimal communications with the upper and the lower layers. It supported a successful industry of hardware vendors, Internet providers, web services and eventually flourishing cloud industry [2].

Similarly, the cloud environment needs a thin layer to interconnect all these cloud platforms together into a standard infrastructure. This research [2] calls this layer, the Cloud Abstraction Layer (CAL). This layer is designed as simple as possible to host virtual resources for computation and storage (In form of data blobs) which all are connected by virtual network links. Cloud users will have the ability to choose their locations of the virtual resources to control latency, reliability, privacy or price.

This architecture has VMs and virtual storage blobs that can be controlled and migrated between various providers IaaS. CAL networking is considering the advancement of Software Defined Networking (SDN) [69]. Figure 2.1 highlights the architecture of interconnected cloud stack based on the IP stack. The lower layer has the available hardware and the next one is a thin layer of software that utilizes the available hardware resources. Then, the CAL that provides VMs, storage blobs and a software-defined internetwork layer that supports VMs migration. On top of this layer, operating systems and databases can be deployed and connected together with network protocols such as the transmission control protocol (TCP). Various applications, including cloud services and the IoT are provided on the top layer [2].

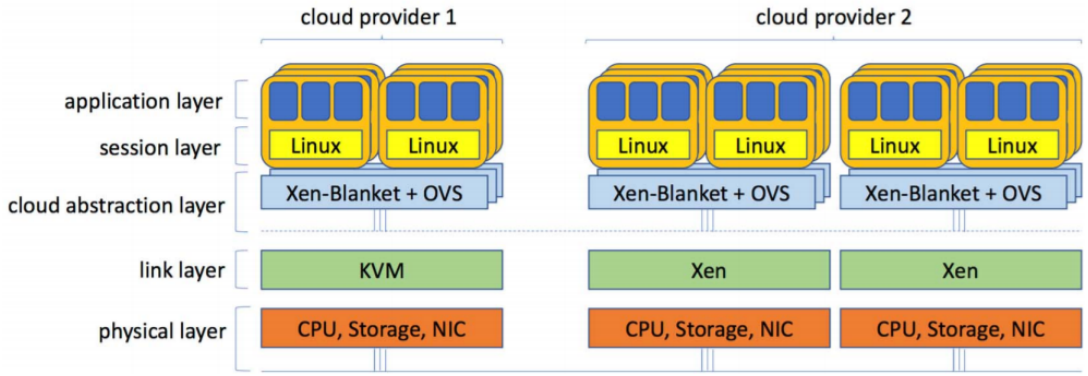


Figure. 2.2 Supercloud architecture [2]

At Cornell Uiversity, a prototype of the CAL has been implemented and is called the Supercloud [14, 70]. The Supercloud can be run across different availability zones of the same provider as well as availability zones of various cloud providers and private clusters as shown in Figure 2.2. To implement this design, there are two layers of hardware virtualization, the lower layer is the infrastructure managed by an IaaS cloud provider such as Amazon Elastic Compute Cloud (EC2). This layer provides VMs, storage and networking. The upper layer is a virtualization layer on top of the abstraction layer and is managed by the Supercloud. It leverages resources from the cloud provider and provides a single interface of OpenStack for the cloud users.

The link layer has the cloud provider's hypervisor. The CAL is implemented using Xen-Blanket [17] which provides a consistent Xen-based paravirtualized interface. Therefore, all major hypervisors including Xen, KVM, Hyper-V, and VMware are supported. Supercloud can be run on the major cloud providers including Amazon EC2, Rackspace, MicrosoftAzure, and Google Compute Engine [20].

2.2 Live cloud migration criteria

Based on a thorough analysis of the related work, we design our own live cloud migration. Our approach, LivCloud is designed based on our criteria which is explained in Table 2.2 [1]. Moreover, we use the criteria to compare our approach to a number of previous live cloud approaches [20–22]. There are three general categories of the criteria: flexibility, performance and security. In terms of flexibility criteria, we distinguish three subcriteria:

- **F1:** decouple the migrated VM from underlying system hardware by supporting wide range of hardware drivers, such as CPU drivers.
- **F2:** support various OS on the migrated VM, for instance, Windows.
- **F3:** consider the migrated VMs' specifications and architecture, including RAM and virtual disk sizes.

There are also three performance criteria:

- **P1:** live migration must be imperceptible to the migrated VM and its users.
- **P2:** predict the required resources to decide whether or not to proceed with live migration.
- **P3:** monitor resource utilization to avoid overutilization and to predict any possible failure.

With respect to security, the following criteria are considered:

- **S1:** maintain data privacy during live migration using encryption.
- **S2:** impose authentication during migration.

2.3 Analysis of three related approaches

To select the approaches for inclusion in the analysis, a thorough review of the literature is conducted to identify recent live user-centric migration approaches that explicitly address one or more of the live cloud criteria criteria. The sources used in the literature review included electronic databases (IEEE, ACM Digital Library, USENIX The Advanced Computing Systems Association and Springer). Three user-centric approaches are found that have successfully designed and implemented live cloud migration:

Table. 2.2 Live cloud migration criteria [1]

Criterion	Criterion Description	Details
Flexibility		
F1	Supporting multiple hardware platforms	Wide range of hardware drivers (CPU architecture and storage)
F2	Supporting mulitple Operating Systems (OS)	Unmodified OS /Modified OS
F3	Considering the migrated VMs' specifications and architecture	RAM, virtual disk sizes & 32 or 64 bit
Performance		
P1	Migration is imperceptible to VM and VM users	Acceptable / Unacceptable
P2	Predicting provision of required resources to decide whether or not to proceed with migration	Estimate resources, Reserve resources, both
P3	Monitoring resource utilization to avoid overutilization and to predict a potential failure	CPU overhead, network bandwidth consumption, disk I/O drivers overhead, memory dirty pages, downtime migration and total time migration
Security		
S1	Privacy (Channel encryption).	Advance Encryption Standard (AES)
S2	Authentication	Hash-based Message Authentication Code using the SHA1 (HMAC-SHA-1)

2.3.1 Supercloud:

In [20], an approach is developed using resources from a number of major cloud providers, including Amazon EC2, Rackspace, HP Cloud and other private clouds. It is named, Supercloud and it uses nested virtualization (Xen-Blanket [17]).

(a) *Flexibility*

F1: decouple VM from underlying system is achieved by using Xen-Blanket approach [17].

F2: Xen paravirtualization cannot run unmodified operating systems (i.e., Windows) [17].

F3: in this approach, the migrated VMs specifications and architecture are not clear.

(b) *Performance*

P1: the approach achieves relatively acceptable performance, about 1.4 seconds migration downtime [17].

P2: disk I/O drivers overhead caused by Xen-Blanket reached 30%, which may affect the physical machine and the other VMs residing on that machine [17, 22, 47].

P3: due to data size, security, cost saving and load balancing, a shared storage accessible by both source and destination is used during the live migration. This exposes the VM to overhead to access its disk over the network [7, 47]. The transport protocol used in the migration is TCP/IP. TCP has a slow start that can affect the migration process and impose extra overhead on the edge equipment. Consequently, it may affect the application's performance [71].

(c) *Security*

S1: the approach does not utilize an encryption algorithm. Also, a security mechanism is not used during the process, so it opens the system to security attacks [48].

S2: the approach does not utilize an authentication algorithm. The approach relies on Xen as its nested virtualization platform, which has a number of issues [48, 72].

2.3.2 Kangaroo:

In [21], there is an OpenStack-based infrastructure approach, called Kangaroo that uses a virtual switch and a Linux container (LXC) to live migrate nested VMs within the cloud.

(a) *Flexibility*

F1: decouple VMs is achieved by using nested virtualization (Qemu & LXC) [21].

F2: the approach cannot run on a variety of O/S (i.e., Windows) because the containers (LXC) are Linux-based.

F3: in this approach, only the size of the VM's vdisk is mentioned, but the VM architecture is not clear whether 32 or 64bit.

(b) *Performance*

P1: the study claims migrating a running application between the approachs local deployment and Amazon within a few minutes and without any downtime [21].

P2: the nested VMs in the study have a 3.2 GB virtual disk, which is migrated using OpenStack block migration. The disk size is not practical and small to run a full Linux or Windows operating systems [62].

P3: despite the achieved performance, the transporting protocol is still TCP/IP. In case of larger virtual disk, big data and low WAN connection bandwidth, it might be difficult to achieve the same result with such a protocol and without any load balancing tools for example [73].

(c) *Security*

S1: the approach does not utilize an encryption algorithm.

S2: the approach does not utilize an authentication algorithm. As the approach uses a layer 2 tunnelling technology to connect VMs, it has the same issues as the previous approach.

2.3.3 HVX:

In [22], an other interseting approach that introduces HVX hypervisor. HVX can run unmodified operating systems (i.e., Windows). It is similar to VMware because both virtualization platforms use binary translation. However, the lack of a popular open-source binary translation hypervisor has allowed other approaches (such as paravirtualization) to be more popular [17, 51].

(a) *Flexibility*

F1: the approach manages to incorporate various virtualization hypervisors, such as, Qemu, Xen paravirtualization, KVM and VMware ESX, therefore, it is able to decouple the VM from underlying hardware [22].

F2: this approach is the only one to run on a modified O/S (Linux) and an unmodified O/S (Windows). Despite, it is seen as a proprietary product and it cannot be evaluated [20].

F3: In this approach, the migrated VMs specifications and architecture are not clear.

(b) *Performance*

P1: there is not a quantitative evaluation of the approachs speed, but rather it is mentioned as robust and reliable [22].

P2: as for the storage migration, the study introduces a storage abstraction layer that copes with cloud storage heterogeneity. However, with large data size, which is most likely to reach a couple of hundreds of gigabytes, the approach may need optimization techniques, such as data compression [7].

P3: as the approach leverages binary translation to achieve a better performance in a nested virtualization environment, many experts do not agree with performance statement as this technique imposes extra overhead on the guest kernel [20, 51]. HVX introduces its own user-defined L2 overlay network (hSwitch). Yet, the transporting protocol is UDP, which is a best effort, connectionless protocol, but unreliable and it is not clear if the study uses a mechanism to recover lost packets due to use such a protocol [74]. Also, the layer 2 network is subject to broadcast storm as multiple clouds may span over the network.

(c) *Security*

S1: the approach does not utilize an encryption algorithm.

S2: the approach does not utilize an authentication algorithm.

2.3.4 Summary of analysis results

Table 2.3 provides a summary of the analysis results. Overall, the analysis shows that in order to gain a better performance, security mechanisms are not implemented. As a result, approaches, such as Supercloud approach proposes tinc VPN as a security mechanism to protect the migration channel because it has less implication on the performance and the local network can be extended across to the cloud using this VPN tunnel [20]. Despite security criteria (S1 and S2) and some performance criteria (P2 and P3) have not been considered, these solutions are still applicable to move VMs hosting publicly visible data (e.g., a Web Server that maintains a catalogue of books for sale). In such a scenario, security (especially, encryption) is not a main concern and in case of a web server migration failure, cloud users might be tolerant to longer time to access the corporate website.

Table. 2.3 Summary of analysis results

Supercloud [20]			**Kangaroo [21]**			**HVX [22]**		
Criteria Identifier		Assessment Values	Criteria Identifier		Assessment Values	Criteria Identifier		Assessment Values
F1	✓	Heterogeneous Hardware (CPU architecture (i.e. flags) & Disk I/O drivers)	F1	✓	Heterogeneous Hardware (CPU architecture (i.e. flags & Disk I/O drivers)	F1	✓	Heterogeneous Hardware (CPU architecture (i.e. flags) & Disk I/O drivers)
F2	×	Only modified O/S (Linux)	F2	×	Only modified O/S (Linux)	F2	✓	Modified (Windows) & Unmodified (Linux)
F3	×	-	F3	✓	Only VM's vdisk size	F3	×	-
P1	✓	Relatively acceptable ($\sim$ 1.4 seconds downtime)	P1	✓	acceptable (no downtime)	P1	✓	acceptable (no downtime)
P2	×	-	P2	×	-	P2	×	-
P3	×	-	P3	×	-	P3	×	-
S1	×	-	S1	×	-	S1	×	-
S2	×	-	S2	×	-	S2	×	-

2.4 Live migration in technologies related to cloud computing

Cloud computing has led to several advancements in technologies such as containers and fog computing. They have their own issues and challenges, such as live migration of containers and VMs across various fog computing providers. In the next section, containers and fog computing architectures as well as some prominent issues are discussed. Moreover, Software Defined Networking (SDN) is discussed as a potential solution to tackle many of the issues facing live cloud migration such as networking challenges.

2.4.1 Containers

A container is lightweight, stand-alone software that has the needed packages to run it, including code, runtime, and system libraries.It is the technology that has been widely adopted in academia and industry [10]. Containers rely on modern Linux operating systems' kernel facilities such as cgroups, LXC and libcontainer. Many cloud providers offer container as a service (CaaS), which allows a wide selection of containers based applications to be available online, such as UberCloud [35, 75]. Containers are fast at the start up and they are ready in less than a second. Also, in comparison to VMs, containers are faster because they consume a very small amount of hardware resources. Docker is the de facto container technology, uses Linux kernel's cgroups and namespaces to run isolated containers within a physical machine. cgroups provide isolation of resources such as CPU, memory, block I/O and network. On the other hand, namespaces isolate an application's view of the operating system environment [76].

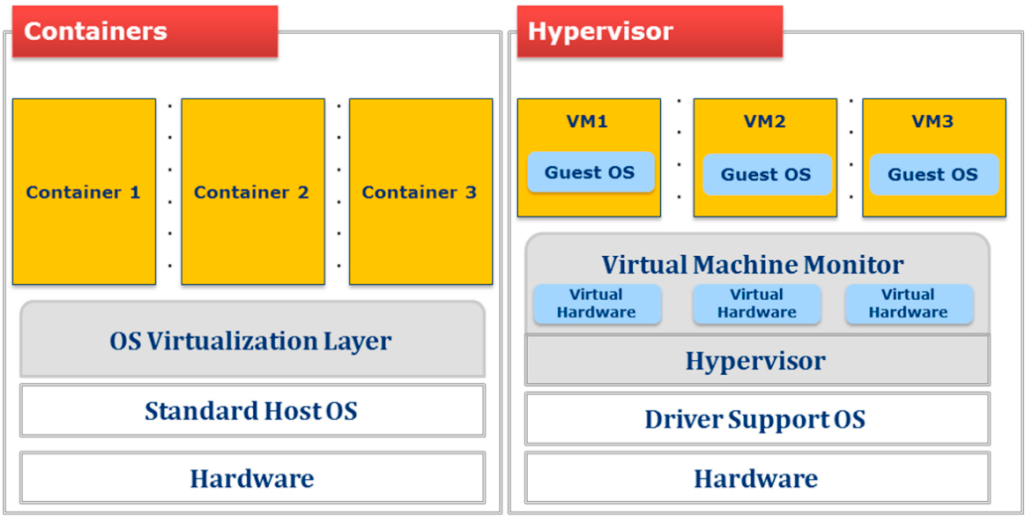


Figure. 2.3 Container and VMs architecture [3]

Although the container provides many benefits, there are still a number of challenges in order to fully adopt containers as a better alternative to VMs. First, due to the sharing of kernel, the isolation and security of containers is weaker than VMs [35] that permit workloads to be isolated from one another and for the resource usage to be controlled. As result, cloud computing relies extensively on the use of VMs than containers. Second, containers still need better capabilities to migrate from one physical machine to another in real time without affecting the applications running underneath. Figure 2.3 shows the general architecture of both VMs and containers [3].

The migration of a container is needed under the circumstances of balancing the load, updating OS and replacing or maintaining the hardware. There are issues (e.g., security and compatibility) for users when using old kernels. Moreover, containers' capability is not only in their individual value, but also in their collective functionality to build services with multiple building blocks [76]. There is an ongoing effort to build cluster-level management for containers. Docker Swarm [77] is a Docker-native clustering system that aims at exposing a cluster of Docker hosts as a single virtual host. However, Docker-Swarm is still in its incubation phase and it does not natively offer support for the availability of the Docker instances.

2.4.2 Fog computing

Fog computing paradigm is a key technology for the internet of things (IoT) by bringing the cloud services to the users regardless of their location with low latency [78]. Fog computing extends the cloud computing by integrating another layer between the clouds

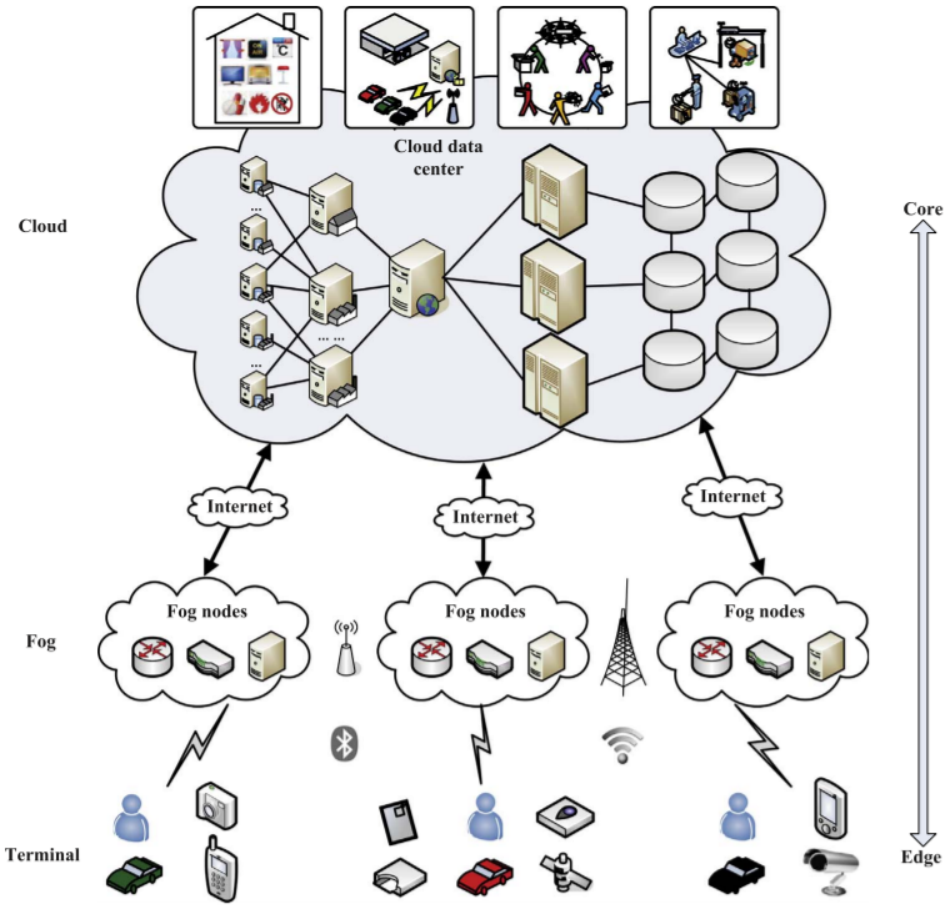


Figure. 2.4 Fog computing architecture [4]

and the users' mobile devices. This new architecture is known as device-fog-cloud. It contains a fog computing node which is a small to medium size computing device. It is usually located at the premises close to the users (e.g. shopping centres and airports). Also, there is a fog instance that is a physical or a virtualized resource on top of the fog node infrastructure to run users' customized applications. It is accessed by mobile devices over the wireless network. Moreover, different fog nodes can be connected to a cloud centre that provides coordination between various fog infrastructures and other services, such as extra computing capacity and large database management [9, 78].

Figure 2.4 shows the fog computing's architecture [4]. The architecture layers are described as follow: *Terminal layer* which consists of various users' mobile devices. *Fog layer* that has large number of fog nodes, which may include routers, gateways, switches and wireless access points. These fog nodes are geographically distributed between the end devices and cloud. The low latency can be achieved in this layer. Moreover, the fog nodes are also connected with cloud data center by IP core network and they are responsible for interaction with cloud to obtain extra computing and storage capabilities.

Cloud layer which consists of multiple high performance servers and storage devices to provide various services, such as smart home and smart transportation.

Fog computing shares many of the cloud computing challenges, including security, privacy and common standards to develop a unified and interoperable platform to manage fog computing from different providers. The Open Fog consortium is the first step in this direction [79, 80]. Similar to cloud computing, fog computing is a virtualized platform that offers proprietary computing, networking and storage services between cloud computing and end mobile devices. This has led to a heterogeneous fog computing environment. In such an environment, it is still challenging to support live migration of VMs across various providers [80].

2.4.3 Software Defined cloud computing

Software Defined Network (SDN) proposes isolating control planes from forward planes in network traffic. This is achieved by integrating a new control layer between applications and the infrastructure. This layer is known as the SDN controller that sends configurations and instructions to the equipment in the infrastructure layer. SDN and Network Functions Virtualization (NFV) have been adopted in cloud computing to optimize and automate configuration and physical resources abstraction by extending the concept of virtualization to all resources in a datacentre including compute, storage and network. Figure 2.5 shows software-defined cloud computing architecture [5].

SDN aims to overcome the limitations of traditional networks, such as multi-tenant environments where computing, storage, and network resources must be offered independent and isolated from one another [81, 82]. SDN decouples data forwarding functions from network control plane, which enables the network to become centrally manageable, programmable and facilitate live migration of services and VMs across different cloud providers [83]. Eramo et al. [84] proposed a consolidation algorithm based on a migration policy of virtualized network function instances to reduce energy consumption. Google adopted SDN in its B4 network to interconnect its Cloud DataCentres(CDC) with a globally-deployed software defined WAN [85]. SDN has been utilized in many previous approaches to facilitate VMs live migration across the WAN and the various cloud providers [20, 54, 58, 64, 66, 66]. SDN can be configured to create layers of network abstraction that can be used to run multiple separate, discrete virtualized network layers on top of the physical network. This configuration provides security benefits and eliminates IPv4 address limitations during VMs relocating [64, 66].

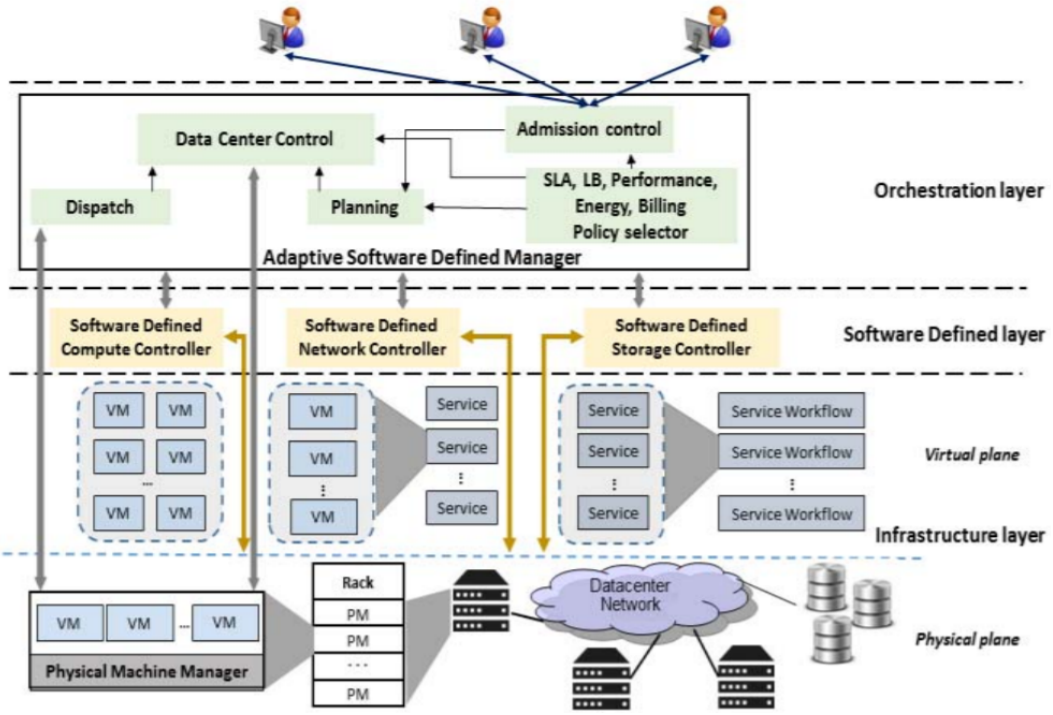


Figure. 2.5 Software-defined cloud computing architecture [5]

2.5 Conclusion

We overview the related work of live cloud migration of VMs across cloud IaaS. Furthermore, we propose live cloud criteria in this chapter and analyse a number of existing approaches based on live cloud migration criteria. This analysis reveals the existing gap in these approaches in terms of the migration downtime (performance), decoupling VMs from underlying systems (flexibility) and securing live migration channel (security). Although, all approaches managed to deploy nested virtualizations (Xen-Blanket, LXC and HVX), but they cause notable performance degradation to the underlying systems and limit VMs from running different operating systems (i.e., Windows). None of these approaches provide adequate security capabilities. moreover, some technologies that related and share many concerns with cloud computing are discussed. In the next chapter, we discuss the design of our live cloud migration approach, LivCloud. The design of this system is reliant on the mentioned criteria, performance (P1, P2 & P3), flexibility (F1, F2 & F3) and security (S1 & S2).

Chapter 3

LivCloud Architecture

Cloud providers offer their IaaS services based on virtualization to enable multi-tenant and isolated environments for cloud users. Currently, each provider has its own proprietary virtual machine (VM) manager, called the hypervisor. This has resulted in tight coupling of VMs to their underlying hardware hindering live migration of VMs to different providers. This chapter proposes a new approach, called LivCloud, to overcome the limitations of previous approaches as explained in Chapter 2. An open-source cloud orchestrator, a developed transport protocol, overlay network and secured migration channel are crucial parts of LivCloud to achieve effective live cloud migration. Moreover, an initial evaluation of LAN live migration in nested virtualization environment and between different hypervisors has been considered to show the migration impact on network throughput, network latency and CPU utilization.

The organization of this chapter is as follows. Section 3.1 presents an introduction. We discuss the related work and the motivation behind our work in Section 3.2. We describe LivCloud Architecture in Section 3.3. The experimental evaluation of LivCloud is described in Section 3.4. Section. 3.5 concludes this chapter.

3.1 Introduction

Cloud computing has been providing considerable capabilities for scalable, highly reliable, and easy-to-deploy environments. Live cloud migration of VMs at IaaS is an active research area, working to overcome the lack of cloud interoperability among

providers. Virtualization is the foundation of the cloud IaaS. It allows cloud users to exploit multi-tenant resources (compute, network and storage) from a secure Cloud IaaS [21]. Virtualization is the conversion of a physical machine to individual isolated spaces (VMs) that can be used by multiple users as per their needs. The isolation and resources provision is provided by hypervisor [24]. Public cloud IaaS is often described as a heterogeneous environment due to the fact that each cloud provider has their own hypervisor. Providers such as Amazon EC2 and Rackspace use the hypervisor Xen; while, Fractus and Google Compute Engine rely on KVM. Windows Azure, on the other hand, uses the Microsoft hypervisor, Hyper-V [20].

Despite that many providers leverage the same hypervisors for virtualization, for example Google and HP both use KVM, live cloud migration of VMs between those providers is still challenging [20]. Every provider has been developing their own APIs and proprietary features to their selected hypervisor. This has made it difficult for cloud users to live migrate VMs to other providers - one aspect of vendor lock-in with substantial consequences [16].

This chapter introduces LivCloud to address the limitations in the existing approaches with respect to flexibility, performance and security. Nested virtualization and network connectivity are the basic requirements of flexibility criteria to successfully start the live cloud migration. The performance criteria are defined by maintaining the connectivity to the migrated VM, the migrated VM's connections and configurations, reserving the necessary resources to the migration process and predicting any potential failure during the migration to save the process by rolling back to the original state. With respect to security criteria, these criteria ensure authenticating and authorizing the legitimate users and processes to interact with the migration. Tables 2.2 and 2.3 show these criteria and applying these criteria on specific live cloud migration approaches. Also, this chapter highlights a reference model of the key factors to successfully achieve live cloud migration.

LivCloud proposes to use different technologies, some of which have never been used in live cloud migration, such as the User Datagram Protocol based data transfer, -known as UDP-based data transfer or UDT, and inter-Software Defined Network (SDN) controller communication (ODL SDNi) [74, 86]. Moreover, it uses KVM for the first time to enable nested virtualization on the cloud IaaS as well as securing the migration channel, which has not been considered in live cloud migration [1]. A preliminary experimental study as well as benchmarking criteria are presented to validate and evaluate the design. Also, to demonstrate that the migration is successful, a live video on the migrated VM

will continue running if the process is completed correctly. This chapter reflects on the second question that aims to identify how to design an effective live migration approach.

3.2 Related work

Nested virtualization has been used to decouple the VM from public IaaS [16, 17, 21]. Nested virtualization is configuring one hypervisor (in the upper layer) within a virtual machine hosted by another hypervisor [25]. Most of legacy hypervisors, such as KVM, Xen, and VMware can run nested virtualization [26, 27, 86]. However, public cloud hypervisors do not allow running nested virtualization [17]. Two main techniques have been used to enable nested virtualization on the top of cloud IaaS, paravirtualization and binary translation. The Xen hypervisor can be configured to run paravirtualization concept, while VMware and hypervisor, HVX run binary translation [20, 22]. KVM is limited in running paravirtualization. However, OPENFV has been developing KVM for running Network Function Virtualization (NFV), which will help overcoming KVMs limitations [87]. A brief discussion of both paradigms is presented in the following.

3.2.1 Paravirtualization

It is a lightweight virtualization technique introduced by the Xen Project team, later adopted by other virtualization solutions such as KVM. It does not require virtualization extensions from the host CPU and thus enables virtualization on hardware architectures that do not support Hardware-assisted virtualization. Therefore, it allows different hardware architecture to be exposed to the VM. However, the VMs kernel has to be modified prior to OS installation, thus it does not support Windows OS [17].

Xen-Blanket is an academic approach is designed using Xen hypervisor. Paravirtulaization significantly helped Xen-Blanket enabling nested virtualization on Amazon EC2 instance. Xen-Blanket has been used by many academic live cloud migration approaches like in [20, 61, 63]. These approaches inherited drawbacks of Xen-Blanket, including, significant downtime during live migration (1.4 seconds) [17], overhead ($\sim$ 30%) on drivers I/O [47] and difficulty to run unmodified OS (Windows) [17]. Furthermore, securing live migration has not been taken into account due to extra latency caused by encryption and authentication [88]. Figure 3.1 shows the paravirtualization architecture.

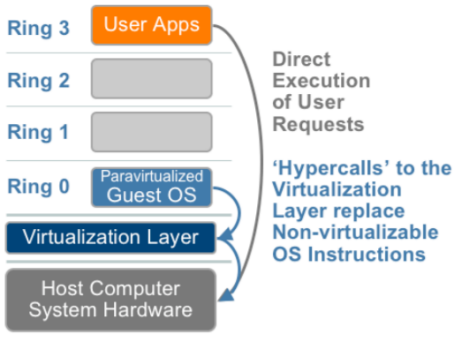


Figure. 3.1 Paravirtualization Architecture [6]

3.2.2 Binary translation

It transfers the VM instructions directly to the underlying system and dynamically converts them to native x86 during runtime. As instance of binary translation, HVX is a proprietary hypervisor designed by Ravello systems [22]. This hypervisor enables nested virtualization on the top of Amazon EC2 and Google Compute engine [22]. The main drawback of HVX is its proprietary status which hinders evaluating its performance. Many experts are sceptical about the performance of binary translation, because it imposes extra overhead on the guest kernel [20]. Moreover, security which has not yet been implemented could be done using IPsec VPN [22]. Figure 3.2 shows the binary translation architecture.

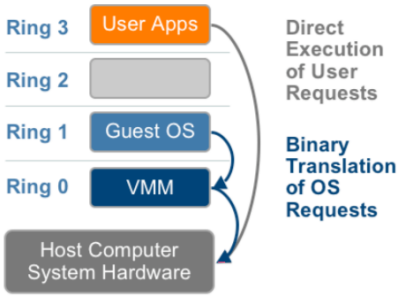


Figure. 3.2 Binary translation Architecture [6]

3.3 LivCloud architecture

LivCloud aims to achieve effective live cloud migration for VMs at cloud IaaS with minimal services interruption. It is similar to NFV Hypervisor-KVM Architecture using KVM as hypervisor and ODL as the SDN controller [87]. Figure 3.3 illustrates a

reference architecture of a successful live cloud migration.

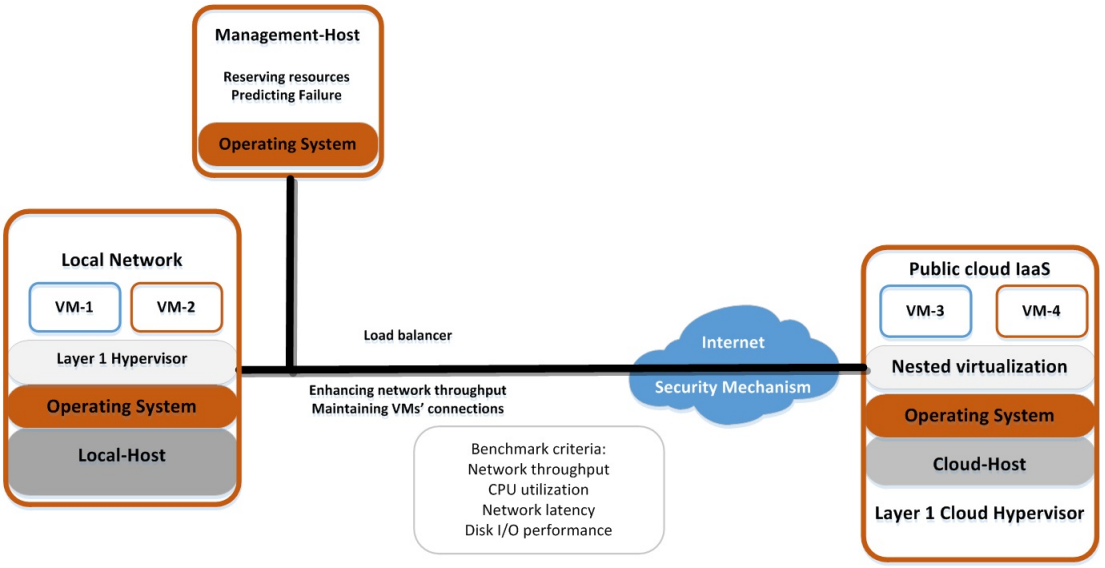


Figure. 3.3 A reference Architecture of live cloud migration

LivCloud is designed based on previously proposed criteria [1], which are in three general categories: performance, flexibility and security. There are three flexibility criteria; F1 decoupling the migrated VM from underlying system by supporting wide range of hardware drivers, such as CPU drivers; F2 supporting various OS on the migrated VM, for instance, Windows; F3 considering the migrated VMs specifications such as RAM and hard disk size and their architectures (64 or 32 bit). There are three performance criteria, denoted as- P1 live migration must be imperceptible to the migrated VM and its users; P2 predicting the required resources to decide whether or not to proceed with live migration; P3 monitoring resource utilization to avoid over utilization and to predict any possible failure. With respect to security, there are two security criteria, S1 maintaining data privacy during live migration using encryption; S2 imposing authentication during migration.

To support effective live cloud migration, the design needs a foundation that supports nested virtualization to decouple VMs from the cloud IaaS and connect hypervisors on the IaaS in order to facilitate live migration back and forth. In addition to this, the design needs to optimize live migration performance, prevent any potential failure, and protect the process against hijacking and penetration. Figure 3.4 illustrates LivCloud technical architecture.

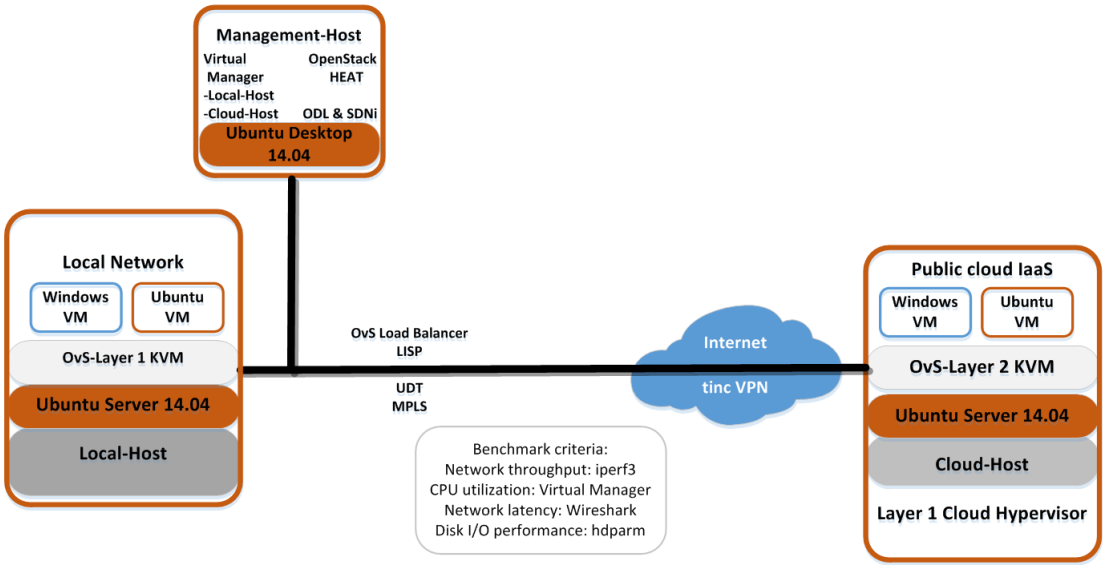


Figure. 3.4 LivCloud Architecture

3.3.1 The basic design stage: nested virtualization and network connectivity

Two fundamental features are necessary to establish network connectivity between LivCloud and the cloud IaaS. Firstly, nested virtualization needs a particular hypervisor installed and configured on source and destination machines. LivCloud uses KVM as a Layer one hypervisor on the source and the destination and as a Layer two installed on certain VMs. Linux virtual manager is a user interface for managing virtual machines mainly on KVM. Any physical or virtual machine that has KVM configured can be connected locally or remotely over SSH to virtual machine manager [89]. In LivCloud, two physical machines and three VMs are connected to the virtual manager installed on LivClouds management machine.

To fully achieve nested virtualization, KVM as a Layer 2 hypervisor must be enabled on either Amazon Ubuntu m3.2xlarge or c4.2xlarge instances. Secondly, both sides must be connected to LivClouds virtual manager in order to live migrate VMs between LivCloud and Amazon instance. KVM supports running modified and unmodified OS [86]. This step is currently underway, working closely with a cloud provider to enable nested KVM. The functional requirements help to fulfill F1, F2, F3 and P1.

3.3.2 The enhancement of the basic design stage: performance, flexibility and security

The optimizations of LivCloud are related to performance, flexibility and security criteria. LivCloud should guarantee enhancing network throughput, maintaining existing VMs connections, reserving required migration resources and securing the process which are briefly explained in the following:

1. Enhancing network throughput: Various technologies are leveraged to help to fulfill P1, including:

   (a) OpenvSwitch (OvS):

   It has flow classification, caching and better performance over the traditional Linux Bridge. Moreover, it has its own load balancer, which is used to distribute loads across available routes [90].

   (b) UDT:

   It is used as the transport protocol between LivCloud and Amazon instance instead of TCP. The protocol's developers claim that UDT has throughput about five times more than TCPs and is as reliable and secure as TCP. Until 2009, UDT was able to win *Supercomputing 2009 Bandwidth Challenge Winner* [91]. Moreover, in [74], a comparison study is conducted between TCP and UDT, which shows that UDT performance is far better performance than TCP, especially in Long Flat Network (LFN) which is known as Bandwidth-Delayed Network. In 2013, VMware announced that UDT can be used to speed up VMs live migration between Datacentres that deploy VMware vCloud Air [92].

   (c) Multiprotocol Label Switching (MPLS):

   It has improved QoS attributes over WAN connections and recently it has been integrated into SDN controllers [93]. LivCloud incorporates MPLS into ODL controller to improve the network performance.

2. Maintaining VMs connections and configurations:

   Live migration of a VM from one site to another over the Internet needs to keep existing connections and configurations, such as, Access Control List (ACL) and DNS records. The following technologies used to maintain these configurations:

(a) Inter SDN controller communication (ODL SDNi):

Different ODL controllers can use this feature to instantly communicate to each other and pass any changes in the topology to each other, such as VMs relocation [86, 94]. Two ODLs are configured on LivCloud and Amazon instance to deploy SDNi between both sites.

(b) ODL Locator/Identifier Separation Protocol (LISP): It is integrated into ODLs on both sides. LISP builds a routed layer on IP using two addresses. These two addresses, Endpoint Identifier (EIDs) and Routing Locator's (RLOCs) are used to decouple the VM from its fixed IP address and keep the existing connections when the VM is migrated [95].

3. Reserving resources and prediction of potential failure:

To help fulfill P2 and P3, LivCloud aims to utilize OpenStack orchestrator, HEAT [96] and a plug-in coded in Python [62]. These components will help to reserve enough resources for migration, finish the migration within predefined time, avoid any potential failure and prevent therefore throttling QoS attributes, such as Service Level Agreement (SLA).

4. Securing the migration channel:

Due to the extra overhead processing and migration downtime added by security mechanism, such as IPsec to live migration, it has been avoided in many live cloud migration approaches [1]. In many cases, the downtime is increased about 5 times when IPsec added to live migration as the study in [88] shows. The study illustrates the increase of both migration downtime and total time migration, from less than two seconds to almost 8 seconds downtime when IPsec VPN is implemented. Moreover, in studies [20], [61] and [63], the live migration is between a local deployment and Amazon services and there is no security mechanism used, despite the fact that Amazon offers load balanced IPsec VPN between its VPC and cloud users' local IaaS [43]. Data have to be encrypted during migration, thus it is protected from any penetration. Also, during migration, authentication has to be imposed in order to prevent any potential hijacking [72]. To maintain encryption (S1) and authentication (S2), LivCloud uses tinc VPN, which is able to provide encryption using Advanced Encryption Standard (AES) and authentication using Hash-based Message Authentication Code (HMAC-SHA1) [20].

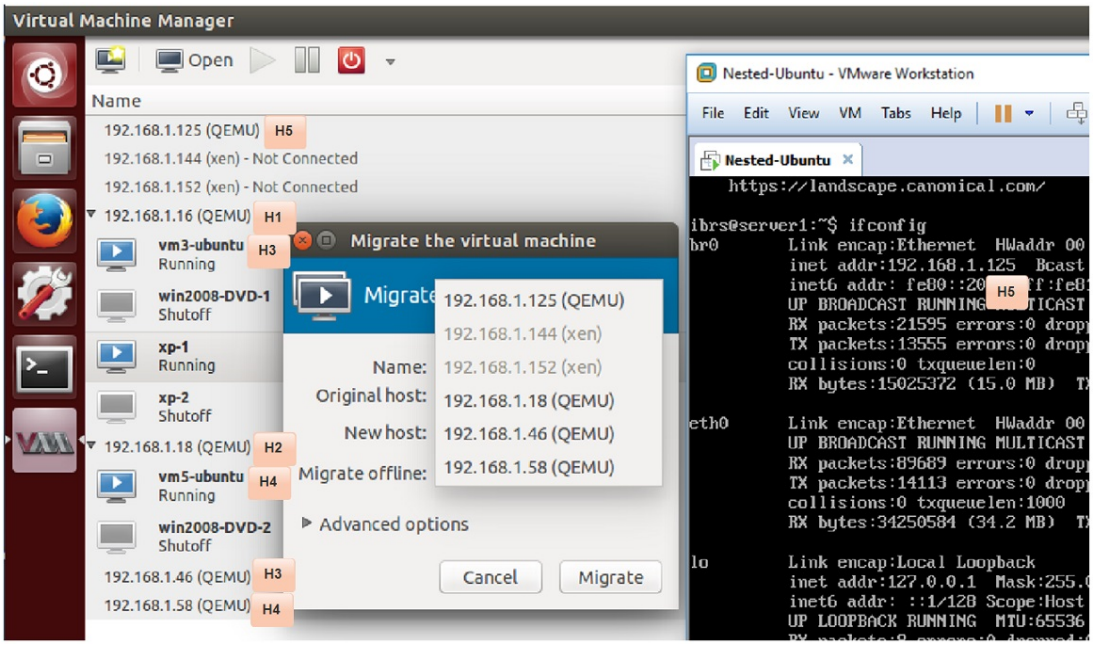


Figure. 3.5 Virtual Manager's connection to all hosts

3.4 Preliminary experimental evaluation

3.4.1 Experiment setup

The experiment aims to evaluate LivCloud within a LAN environment. Thus, the lab setup consists of four HP Z440 workstations are connected through Cisco L2 switch providing a 100 Mbps. Each machine has 32 GB of RAM, 1TB disk and 4-core 2.8GHz Intel(R) Xeon(R) E5-1603 v3 CPU. Ubuntu Server 14.04 LTS and KVM (a Layer 1 hypervisor) are installed and configured on two machines, H1 and H2. Using KVM, VMs, H3 is created on H1 and H4 on H2. KVM (a Layer 2 hypervisor) is configured on H3 and H4. Microsoft Windows 10 and VMware workstation 12 are installed on the third machine. Using VMware, H5 is created and equipped with Ubuntu Server 14.04 as well as KVM. The last machine is configured as a NFS server (FreeNAS 9.3) for LivCloud. Any VM can be configured with a local disk or a disk hosted on the NFS server. H3 and H4 have their disks hosted on the NFS server. Hosts H1, H2, H3, H4 and H5 are connected to the virtual manager as shown in Figure 3.5. A VM with Windows XP used as the migrated VM between all hosts. It has 1GB of RAM and 2vCPUs. The connection between LivCloud and Amazon instance is an essential part of live cloud migration. This connection is established through a VPN. Figure 3.6 shows how both sides are connected [43]. VPC is Amazon Virtual Network that helps building

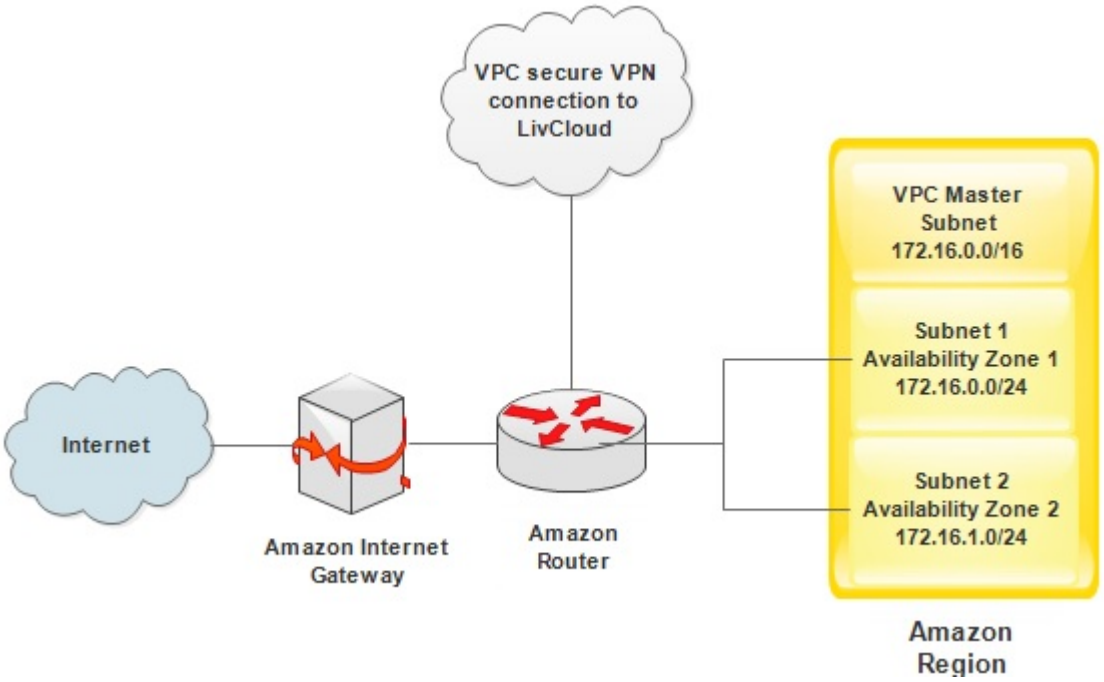


Figure. 3.6 LivCloud's connection to Amazon

user-defined private network subnets inside the cloud in order to facilitate controlling IP address changes.

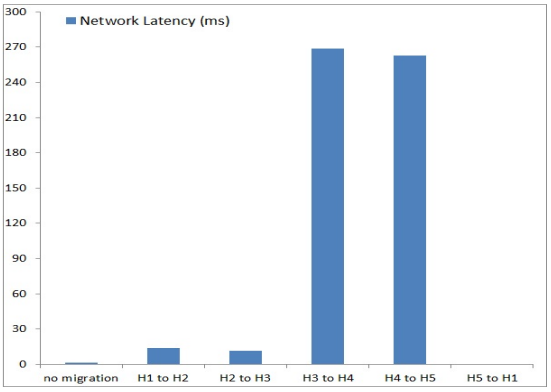
3.4.2 Experiment motivation

The main motivation behind conducting KVM live migration within the LAN environment is to illustrate that despite the LAN resources are less affected by the migration than the WAN, there is still a notable impact of the process on the LAN without the proposed criteria [1]. Network throughput, CPU utilization, network latency, migration downtime and disk I/O performance are the main parameters used to analyze the live migration impact. Network throughput is measured using iPerf version3 [97] and migration downtime is measured by using Wireshark protocol analyzer [98], whereas disk I/O performance is tested using hdparm command [99].

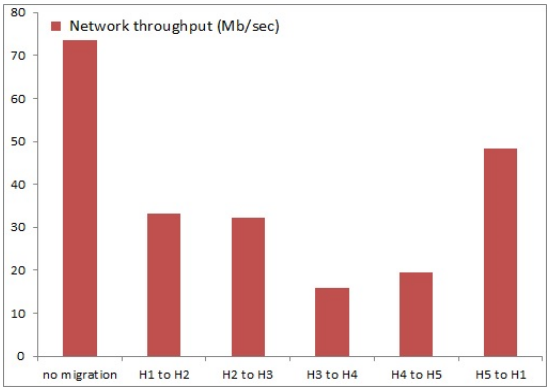
3.4.3 Experiment results discussion

Figure 3.7 illustrate the experimental results. These results are the average of conducting the experiment of a total of 15 runs. In terms of the experiment times, it is done during the morning, afternoon and during the night. This approach is used in any

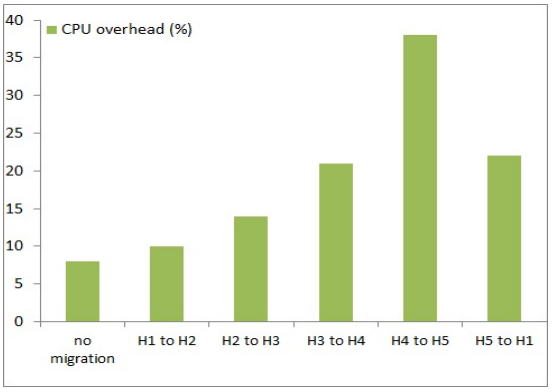
experiments carried out in this research. In particular, Figure 3.7(a) shows that there is downtime (0.07 seconds) during live migration between H5 and H1 (VMware to KVM) because of that there is no network latency.



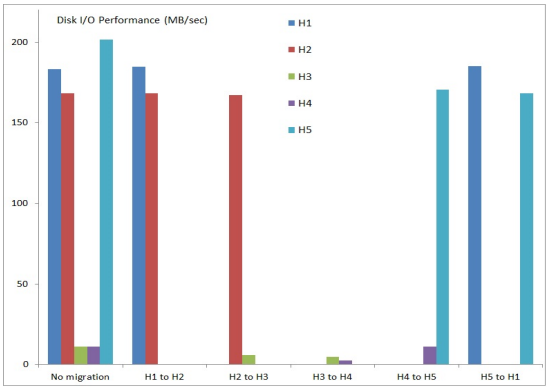
(a) Network latency



(b) Network throughput



(c) CPU overhead



(d) Disk I/O performance

Figure. 3.7 Results statistics

A remarkable increase in network latency is obtained when migrating from H3 to H4 and from H4 and H5 (a Layer 2 hypervisor). Figure 3.7(b) shows that the network throughput is highly affected, in particular when migrating between H3 and H4. In effect, the throughput decreases $\sim 25\%$ compared to the case of no migration.

In Figure 3.7(c), the CPU load increases by almost 20% during live migration between H1 and H5 (KVM to VMware). Figure 3.7(d) shows that the I/O performance of disks that are hosted on the NFS server (H3 and H4) are severely affected by the migration. These hosts access their disks through the network to write and read data. The impact is notable in accessing the disks through the LAN. Therefore, it will be more pronounced in the case of WAN and live cloud migration.

Live migration of VMs disks has been considered in many studies [11][74]. However, it is considered to be unreliable and needs synchronization between CPU processing speed and network bandwidth [100]. Moreover, many cloud users prefer keeping VMs disks

in-house for more control and privacy [20, 61, 63]. As mentioned earlier, LivCloud uses KVM that has a live block migration feature that allows migrating the disks state [101]. However, during the initial evaluation of LivCloud, this feature showed instability and the process crushed many times. On account of this, the disk live migration is cloud users decision to use this feature or leave the disk on the shared storage in LivCloud.

3.5 Conclusion

Given the requirements of cloud users, such as cloud service continuity and data privacy, there is a clear need for live migration of VMs at IaaS. The current cloud providers IaaS is heterogeneous and hence, hinders live migration of VMs. Every provider deploys their own developed virtualization platforms. Many user-centric approaches to overcome virtualization heterogeneity, including Xen-Blanket and HVX, attempted to achieve the migration with minimal service disruption. While they have managed to devise a customized nested virtualization such as paravirtualization and binary translation, they have shown shortfalls in terms of migration downtime, decoupling VMs from underlying systems and securing the live migration.

LivCloud is designed to address the existing issue using different components, including KVM, OpenDaylight controller, UDT and OpenStack orchestrator, Heat. The initial evaluation shows that live migration impacts the LAN resources, including network throughput and latency as well as CPU utilization and disk I/O performance before and during live migration. Optimization is needed to tackle the migration negative impact, in particular when live migrating between different hypervisors (KVM and VMware) and when VMs disks are hosted on an NFS server. The next chapter shows the implementation of basic design of LivCloud on Amazon EC2.

Chapter 4

The basic design of LivCloud on Amazon EC2

Having defined the design requirements of LivCloud, this design is configured and evaluated on a public cloud provider, Amazon EC2. Amazon is one of the most popular platforms [20, 22]. Cloud users may decide to live migrate their virtual machines from a public cloud provider to another due to a lower cost or ceasing operations. Currently, it is not possible to install a second virtualization platform on public cloud infrastructure (IaaS) because nested virtualization and hardware-assisted virtualization are disabled by default. As a result, cloud users' VMs are tightly coupled to providers IaaS hindering live migration of VMs to different providers. This chapter introduces LivCloud, a solution to live cloud migration. LivCloud is designed based on well-established criteria to live migrate VMs across various cloud IaaS with minimal interruption to the services hosted on these VMs. Also, it discusses the basic design of LivCloud which consists of a Virtual Machine manager and IPsec VPN tunnel introduced for the first time within this environment. It is also the first time that the migrated VM architecture (64-bit & 32-bit) is taken into consideration. In the context of this research, we evaluate the implementation of the basic design of LivCloud on Amazon EC2 M3 and C4 instances. M3 instance is similar to general workstation performance and C4 has a compute optimized instance and has high performance processors. First, we envision the instances as 32-bit machines to enable nested virtualization. Second, we explore three developed options. Theses options are being tested for the first time on EC2 to change the value of the EC2 instance's control registers. Changing the values of the registers will significantly help enable nested virtualization on Amazon EC2.

The organization of this chapter is as follows. Section 4.1 presents an introduction of this chapter. We introduce the LivCloud architecture on Amazon EC2 in Section 4.2. Implementing LivCloud architecture on EC2 is explained in Section 4.3. In Section 4.4, Configuring and testing HQEMU on Amazon EC2 instances is tested and evaluated. We discuss the implementation results in Section 4.5. In Section 4.6, the possible solutions to enable nested virtualization on Amazon EC2 are outlined. Section 4.7 concludes this chapter.

4.1 Introduction

In 2016, RightScale conducted cloud computing trends survey in which 1060 IT professionals were interviewed about their adoption of cloud infrastructure and related technologies. The survey showed 17% of enterprises had more than 1000 virtual machines (VMs) in public cloud, up from 13% in 2015 [102]. This number of VMs would have been reduced to 250 VMs hosting 4 VMs each if public cloud IaaS had not been deliberately locked (disabled nested virtualization or no hardware-assisted virtualization features enabled). This chapter evaluates the basic design of LivCloud on Amazon EC2 m3.2xlarge and c4.2xlarge instances [56]. By default nested virtualization or hardware-assisted virtualization features (Intel VT-x, Intel VT-d and Extended Page Tables) are not enabled on any Amazon instances [20, 22]. Consequently, enhanced QEMU, HQEMU is configured as a second layer hyperviosr. HQEMU [100] is an academic project to enhance QEMU performance by using dynamic binary translation (DBT). DBT is similar to binary translation mentioned in Section 3.2, but DBT is an open source technology. The implementation process has a number of twisted configurations to overcome Amazon network and KVM configuration challenges. For example, adding a second network interface with Elastic IP [103] is layer 3 networking with detailed steps to correctly enable this interface; whereas, in traditional operating system, adding a second interface is a simple layer 2 networking. Appendix A.2 shows more detail on this issue. Moreover, configuring IPsec VPN tunnel between Amazon VPC and the local network to secure the migration channel. The Virtual Machine manager (VMM) is used as GUI interface to connect *Cloud-Host* on the Amazon VPC to *Local-Host* on the local network. IPsec VPN and the virtual manager are the main contributions of implementing of LivCloud.

As mentioned in Section 3.2, two main techniques have been used to enable nested virtualization on the top of cloud IaaS: paravirtualization and binary translation. The

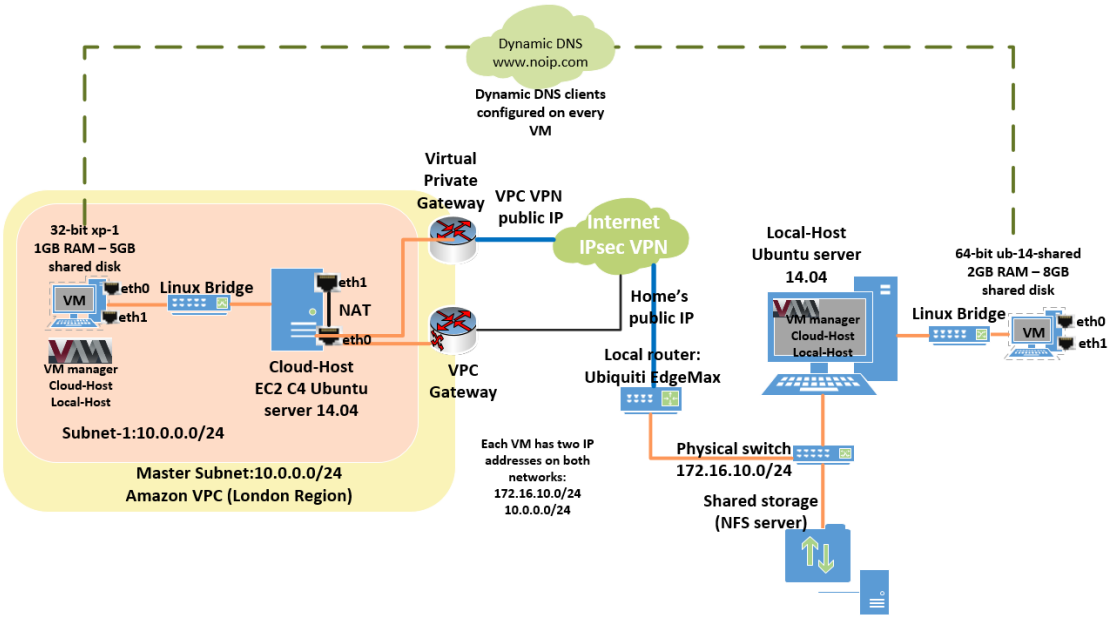


Figure. 4.1 LivCloud's implementation on Amazon EC2

Xen hypervisor can be configured to run paravirtualization concept, while VMware and hypervisor, HVX run binary translation [22]. KVM is limited in running paravirtualization. However, OPENFV has been developing KVM for running Network Function Virtualization (NFV), which will help overcoming KVMs limitations [87]. Two related user-centric approaches are already explained in Chapter 2, the fist one in [20] uses paravirtualization and the second runs binary translation [22].

4.2 LivCloud architecture on Amazon EC2

Figure 4.1 illustrates LivCloud architecture on Amazon EC2. LivCloud is designed based on live cloud migration criteria published in [1] and explained in Chapter 2.

To support effective live cloud migration, the design needs a foundation that supports nested virtualization in order to decouple VMs from the cloud IaaS and connect hypervisors on the IaaS in order to facilitate live migration back and forth. In addition to this, the design needs to optimize live migration performance, prevent any potential failure, and protect the process against hijacking and penetration [1]. The basic requirements help fulfill F1, F2, F3 and P1. In the basic design stage, *Dynamic DNS* is used to maintain the migrated VM's connections and configurations (P1). Dynamic DNS is used to keep a domain name pointing to the same physical or virtual server connected to the Internet regardless of any IP addresses changes [104]. Also, IPsec VPN tunnel is used

to fulfill S1 and S2. The secure connection between LivCloud and IaaS is an essential part of live cloud migration.

4.3 Implementing LivCloud on Amazon EC2

In this implementation, *Cloud-Host* in LivCloud architecture is Amazon instance. In this section, the cloud host is being configured to allow live migration and network connectivity between the local host and the cloud host (see Figure 4.1). We consider Qemu-KVM and network configurations of different Amazon instances.

4.3.1 Envision Amazon Ubuntu instance as 32 bit operating system

Amazon instance is envisioned as an x86 machine and use Qemu-system 32 and KVM paravirtualization to enable nested virtualization on the instance. In a simulated environment on VMware Workstation 12, a VM with 32-bit Ubutnu Desktop and KVM (layer 1 hypervisor) is created. Using virtual manager, a VM with 32-bit Windows XP is created on Ubuntu Desktop VM. There is a message stating that KVM is not working properly and it needs additional packages. It is similar to the message is encountered when installing Ubuntu Desktop and KVM on Amazon Ubuntu Server instances (t2.micro and m3.2xlarge). The instance is forced to ask for the same additional packages to help running KVM properly. It does ask for the same packages. Yet, when creating a new VM on the top of Amazon instance (m3.2xlarge), the message is shown (see Figure 4.2) states that KVM is not installed and further actions are needed.

Despite the warning message, a VM with Windows XP OS is created and the installation is successful. Following that, other issues are encountered, including VT-x and Extended Page Table (EPT) as well as an issue with the VM network (the bridge interface). Intel VT-x is necessary to create a 64-bit VM [105]. EPT provides a hardware assist to memory virtualization that includes the partitioning and allocation of physical memory between VMs. The guest OS stores the mapping between virtual and physical memory addresses in page tables [105]. The Amazon instance's architecture is almost identical to a 64-bit architecture. However, it is not possible to create a 64-bit VM on the top of it. Only, a 32-bit VM is supported. The VM has performance degradation because VT-x and EPT are not enabled. The system architecture of 32-bit Ubuntu Desktop with VT-x enabled and 32-bit Ubuntu Desktop with no VT-x to Amazon fully

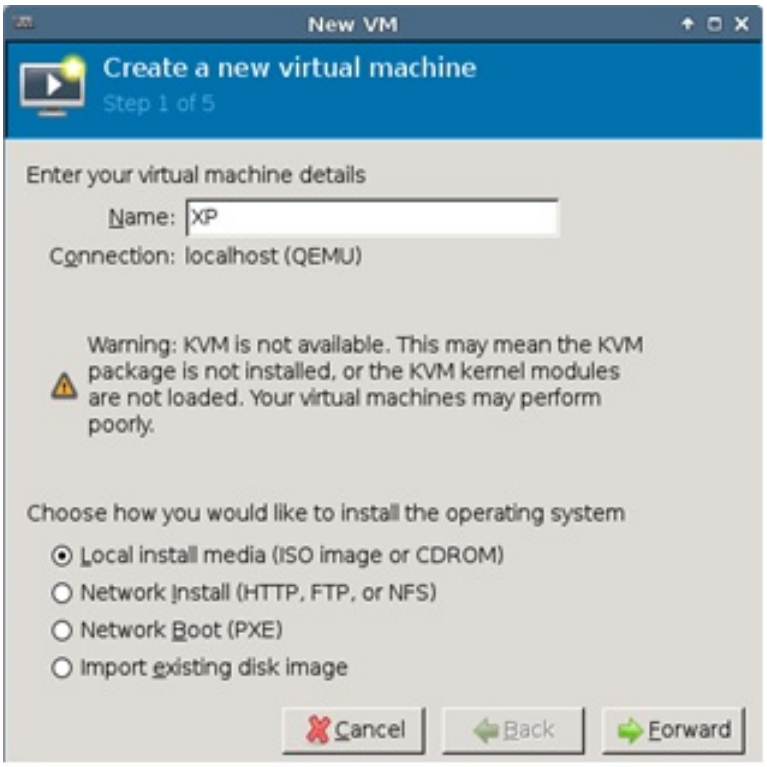


Figure. 4.2 KVM warning message on Amazon instance

virtualized (HVM) 64-bit Ubuntu Server are compared using lscpu command [106]. It is possible to live migrate a VM between 32-bit Ubuntu with VT-x and 32-bit Ubuntu with no VT-x as shown in Figure 4.3. The mentioned systems' architecture and a similar warning message to the message in Figure 4.2 are shown in Figure 4.4 and 4.5. Table 4.1 illustrates the architecture of the mentioned operating systems.

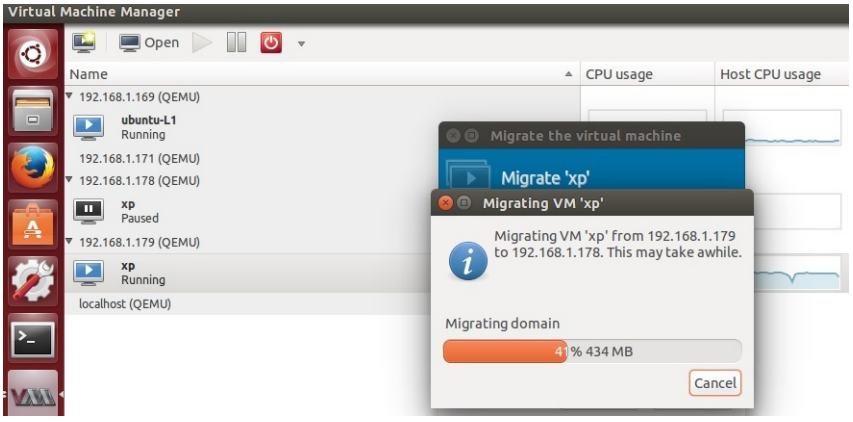


Figure. 4.3 Live migration between the two Ubuntu systems

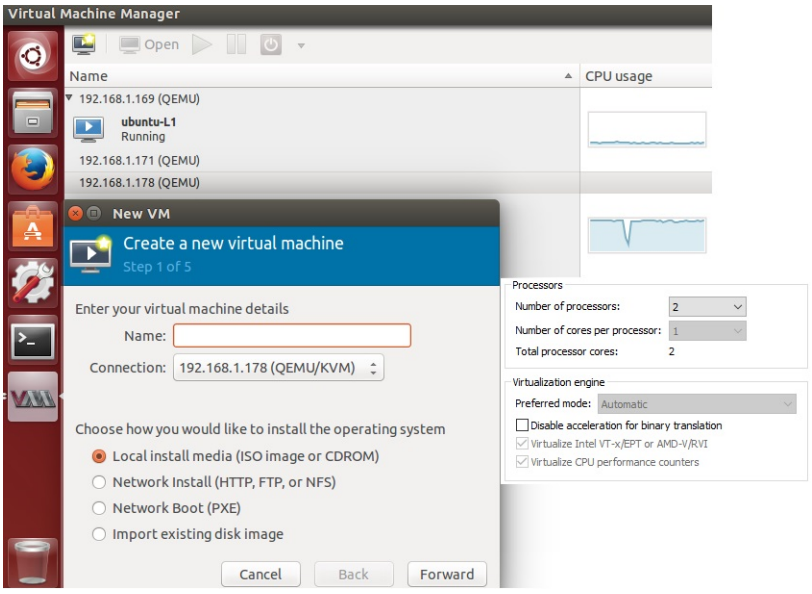


Figure. 4.4 32-bit Ubuntu desktop with VT-x enabled

4.3.2 Linux bridge issue

Linux bridge is used by KVM and Xen hypervisors for handling communication between VMs and the network interface card (NIC) of the hosting machine. It works as a layer-2 switch which forwards traffic based on the MAC addresses of NICs. Usually, when installing KVM, Linux bridge joints VMs to the physical network, so they can request IP addresses from the underlying network DHCP server [107]. In this case, the network card of the hosting machine is configured to pass its IP address to the bridge.

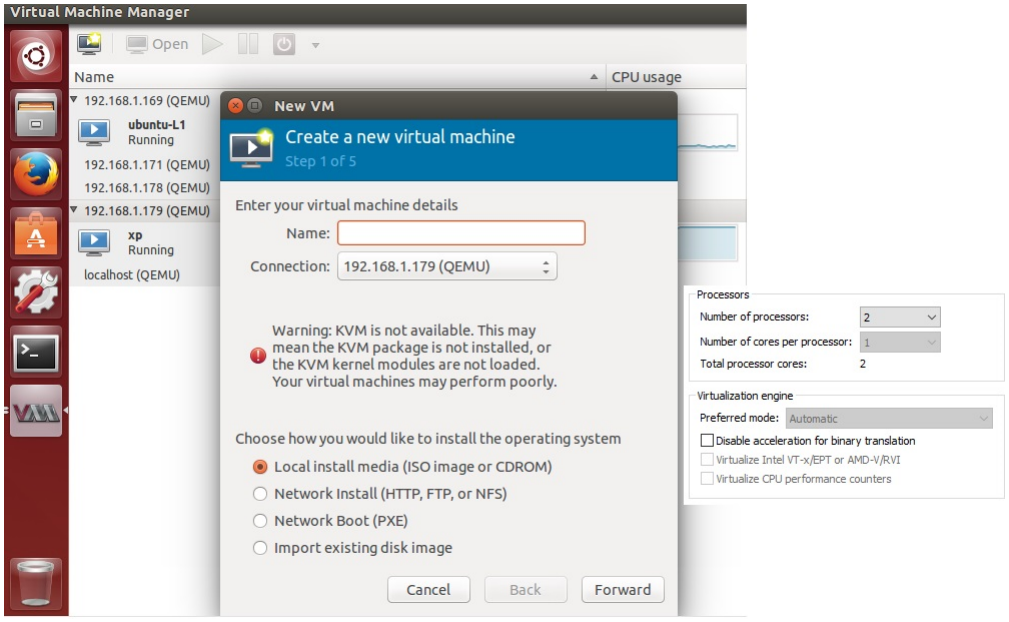


Figure. 4.5 32-bit Ubuntu desktop with no VT-x

Table. 4.1 Comparison between various Ubuntu architectures

Architecture Component	32-bit Ubuntu Desktop with VT-x	32-bit Ubuntu Desktop with no VT-x	Amazon t2.micro 64-bit Ubuntu
Architecture:	i686	i686	x86_64
CPU op-mode(s):	32-bit	32-bit	32-bit, 64-bit
Byte Order:	Little Endian	Little Endian	Little Endian
CPU(s):	2	2	1
Core(s) per socket:	1	1	1
Socket(s):	2	2	1
Vendor ID:	Genuine Intel	Genuine Intel	Genuine Intel
CPU MHz:	2394.459	2394.459	2500.042
Hypervisor vendor:	VMware	VMware	XEN
Virtualization type:	VT-x	0	Full

Figure 4.6 shows what it would be ideal configurations for Linux bridge on Amazon instance, the cloud host. However, when configuring the bridge on Amazon instance, connectivity to the Amazon instance is lost. Every Amazon VPC has DHCP server configured to handle any DHCP request from any network card of an instance within the VPC. The DHCP server sees the bridge as an undefined interface, so it does not give it an IP address. Subsequently, the connectivity with the instance is lost whenever the bridge is configured.

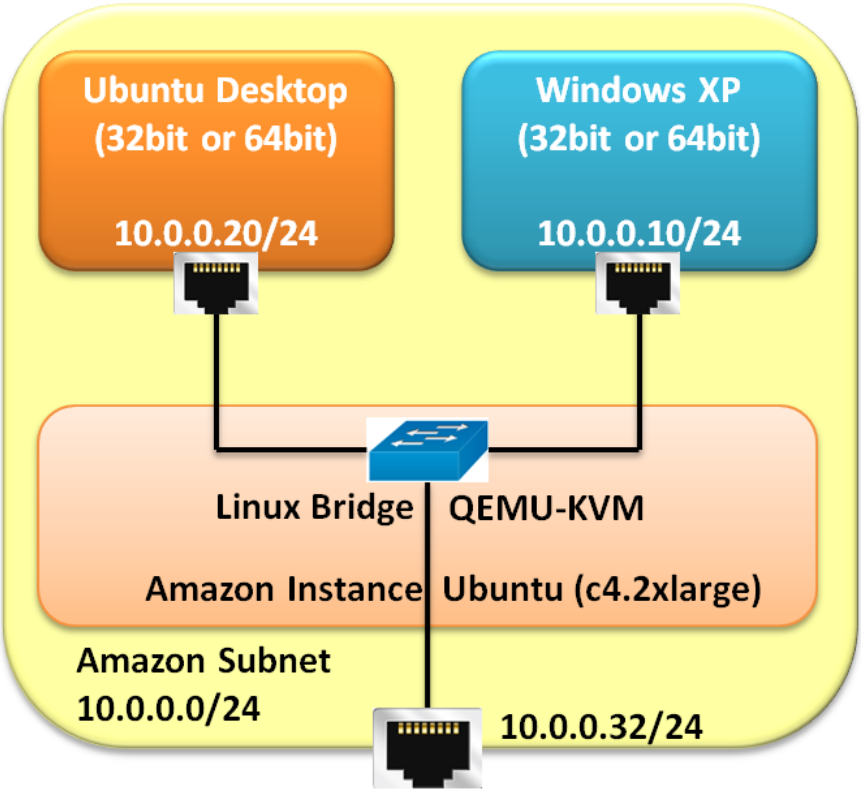


Figure. 4.6 Linux bridge ideal configurations of on Amazon (Cloud host)

Table. 4.2 Amazon C4 instance's specifications

C4 Feature	Specification
Processor Number	E5-2666 v3
Instruction Set	64-bit
Processor Base Frequency	2.9 GHz
Max All Core Turbo Frequency	3.2 GHz
Intel vPro Technology	Yes
Intel Virtualization Technology (VT-x)	Yes
Intel Virtualization Technology for Directed I/O (VT-d)	Yes
Intel VT-x with Extended Page Tables (EPT)	Yes
Intel 64	Yes

4.3.3 Enabling nested virtualization on C4 instance using KVM and VMware workstation

To overcome the issues mentioned earlier in relation to VT-x, EPT and the bridge connectivity, a different instance, c4.2xlarge is tested. C4 was launched in January, 2015 as a EC2 Compute-optimized instance. Table 4.2 shows the specifications of the instance. According to [108], the instance has the required features to enable layer 2 hypervisor and EPT to improve VMs performance, but the link had been modified and this feature has been removed. Appendix A.1 approves the case until 20/05/2016. After creating the instance, VMware Workstation 12 is installed on the instance. The NAT feature of VMware is used to hide the virtual network behind the IP address of the network card of the instance. Therefore, VMs do not have to request addresses from the VPC's DHCP server. It works and all VMs have Internet connectivity. However, KVM is installed on them with the same warning message. Figure 4.7 illustrates the configurations on Amazon C4 instance at this development stage. It is not possible to enable VT-x and EPT from inside VMware because Amazon hypervisor Xen prevents that. Figure 4.8 illustrates the error message. Both VMs are connected through Virtual Manager and a VM (XP with 1GB RAM and 2vCPU) is created on one of them. Then, this VM is live migrated to the other one. The XP VM has to be restarted upon finishing the migration because the VM is on a halted state.

According to Table 4.2, the instance is capable of running VT-x and EPT. During the initial evaluation it has been proved that these features have been disabled.

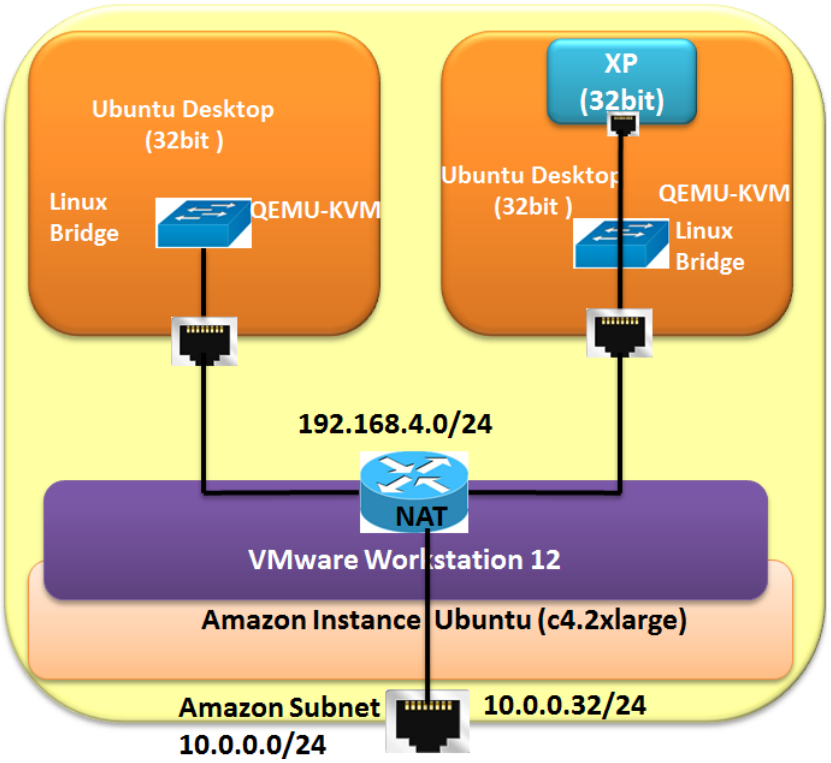


Figure. 4.7 The configurations on C4 at this development stage

4.4 Configuring HQEMU to implement LivCloud on EC2

The main motivation behind conducting HQEMU live migration between Local-Host and Cloud-Host across the Internet is to illustrate that LivCloud basic design can be implemented on uncontrolled environment, Amazon's datacentre without any enhancements from the next stage of LivCloud.

Figure. 4.8 EPT message on Amazon

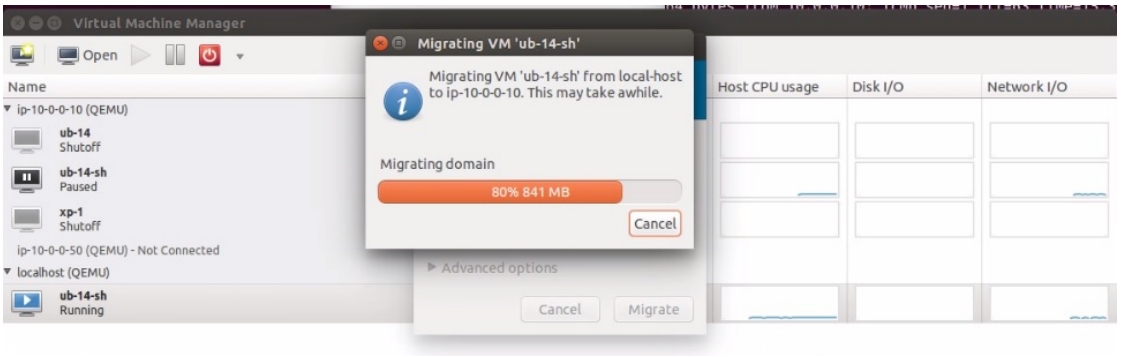


Figure. 4.9 Virtual manager's connection to both hosts

4.4.1 Experiment setup

To achieve the basic design stage, a private network (172.16.10.0/24) based in Bournemouth (UK), which has two physical servers (Local-Host and NFS server) is connected to a Ubuntu server 14.04 (private address, 10.0.0.10/24) on Amazon's datacentre in London, UK. The experiment aims to evaluate LivCloud's basic design within the mentioned environment. Thus, the lab setup as shown in Figure 4.1 consists of one HP Z440 workstation is connected to the Internet through EdgRouter X and Netgear L2 switch providing 1Gbps. The workstation has 32GB of RAM, 1TB disk and 4-core 2.8GHz Intel(R) Xeon(R) E5-1603 v3 CPU. 64-bit Ubuntu Server 14.04 LTS, HQEMU (Layer 1 hypervisor) and HQEMU routed network are installed and configured on the machine, *Local-Host*. The other machine on the private network is configured as an NFS server (FreeNAS 9.3) for the lab.

The Amazon c4.2xlarge instance 64-bit Ubuntu server 14.04, *Cloud-Host* is connected through a network card providing 1Gbps. The server has 15GB of RAM, 100GB disk and 8 vCPU 2.9GHz Intel(R) Xeon(R) Xeon E5-2666 v3. HQEMU (Layer 2 hypervisor) and HQEMU routed network are installed and configured on the instance. Any VM on either Local-Host or Cloud-Host can be configured with a local disk or a disk hosted on the lab NFS server. Using HQEMU, VMs, 2 VMs, 32-bit Windows XP, xp-1 and 64-bit Ubuntu server 14.04, ub-14-sh used as the migrated VMs between both hosts. Their disks are hosted on the NFS server. The Windows VM has 1GB of RAM, 2vCPUs and 5GB of disk. Whereas, the Ubuntu VM has 2GB of RAM, 2vCPU and 8GB of disk. The private network and the Amazon VPC network are securely connected via IPsec VPN tunnel. Local-Host and Cloud-host are connected through the tunnel via the VM manager that is installed on Local-Host as shown in Figure 4.9. VPC is Amazon Virtual Network that helps building user-defined private network subnets inside the cloud in

order to facilitate controlling IP address changes [43]. Furthermore, Dynamic DNS is used to maintain the migrated VMs' connections and configurations (P1).

Dynamic DNS is used to keep a domain name pointing to the same physical or virtual server connected to the Internet regardless of any IP addresses changes [104]. no.ip is a dynamic DNS provider that has been chosen to register ub-14-sh and xp-1 under the DNS name records, ub-14-sh.ddns.net and xp-1.ddns.net respectively. Dynamic DNS clients (noip-2.1.9-1) are installed and configured on both migrated VMs [109]. Also, to prove that it can achieve flexibility and security despite that it is not possible to conduct a successful migration at this stage. Moreover, the migrated VMs' hardware specifications in respect to RAM and disks sizes are larger than the migrated VMs in previous approaches [20–22].

4.5 Experiment results and discussion

In this scenario, both hosts have HQEMU bridged or routed network installed and configured because KVM modules cannot be loaded on Amazon EC2 instances. IPsec VPN tunnel is configured between the Local-Host's private network and Amazon VPC. Local-Host and EC2 Cloud-host are connected through the tunnel via the virtual manager that is installed on Local-Host as shown in Figure 4.9. The migration process of ub-14-sh is also shown in this figure. Amazon VPC provides two public IPs to VPN tunnel for load-balancing. Figure 4.10 shows a comparison between the latency (RTT) of a direct ping from Local-Host to Cloud-Host's public IP and the latency of a ping through

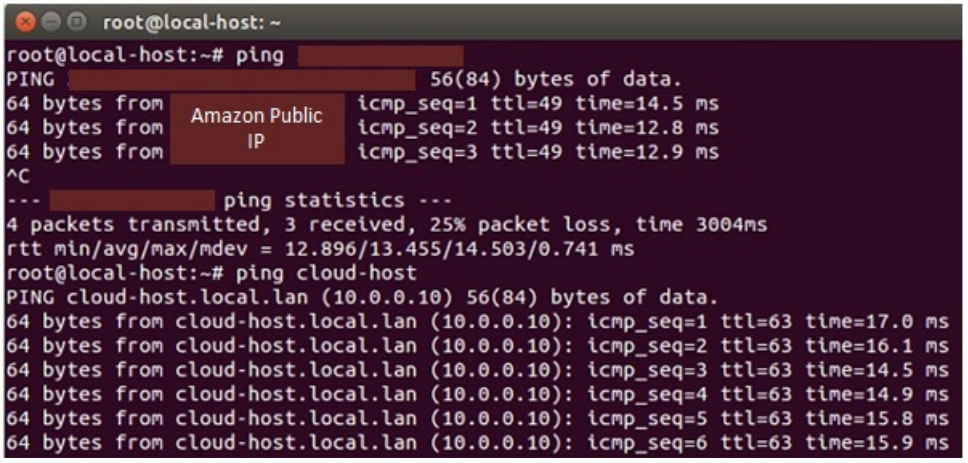


Figure. 4.10 Latency comparison between Internet connection and IPsec VPN

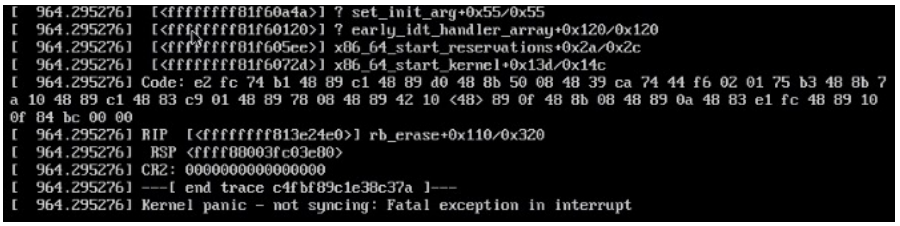


Figure. 4.11 Migrated Ubuntu VM's kernel panic

the IPsec VPN from Local-Host to Cloud-Host's private IP. The private network and the Amazon VPC network are securely connected via IPsec VPN tunnel.

4.5.1 Live migration with shared disk

Despite the successful completion of the migration of 2 VMs with shared disks (xp-1 & ub-14-sh) from Local-Host to Cloud-Host, it is necessary to restart both VMs to fix the halt state on xp-1 and the kernel panic on ub-14-sh. The average total migration time of ub-14-sh is just above 3 minutes, whereas, it is about 2 minutes in xp-1 migration. Furthermore, the performance of both VMS is rather slow despite compiling HQEMU [100] instead of QEMU. Figure 4.11 shows the kernel panic of ub-14-sh.

4.5.2 Live migration without shared disk

Live migration of VMs disks has been considered in many studies [17, 61, 63]. However, it is considered to be unreliable and needs synchronization between CPU processing speed and network bandwidth [71]. Moreover, many cloud users prefer keeping VMs disks in-house for more control and privacy [20]. As mentioned earlier, LivCloud uses HQEMU that is an enhancement of QEMU. QEMU has a live block migration feature that allows migrating the disks state [100]. However, during the evaluation of LivCloud, this feature showed instability and the process crushed many times. However, before crushing both VMs continue working for almost 2 minutes and Dynamic DNS's records are correctly update with the new public IP. The total migration time is

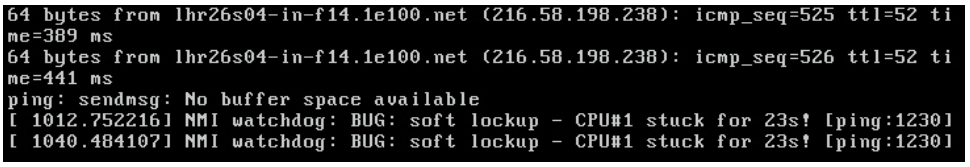


Figure. 4.12 Migrated Ubuntu VM's halt state

approximately 15 minutes of both VMs due to the disks sizes. As a result, live migration of the VM's disk is cloud users' decision to either use this feature or leave the disk on the shared storage in LivCloud. Figure 4.12 shows the crushing of the migrated Ubuntu VM.

In the next section, a number of solutions to enable nested virtualization on Amazon EC2 are discussed.

4.6 Possible solutions to enable nested virtualization on EC2

To load KVM modules on public IaaS, the hardware-assisted virtualization features must be enabled. To check if the IaaS has theses features, the KVM acceleration must be enabled, VMX or SVM flags's number should be larger than 0 and the nested virtulaition of kvm_intel must be set to 'Y'. Figure 4.13 proves that these features are not enabled on Amazon EC2 instances.

There is a number of possible solutions to enable these values and consequently, to enable the hardware-assisted virtualization on public cloud IaaS. The following solutions have been explored as part of the implementation process.

4.6.1 Recompiling Amazon C4 Linux instance's kernel

This solution aims to recompile the instance kernel with specific features enabled such as, KVM acceleration support as shown in Figure 4.14 using the latest version of Linux kernel [110] and *menuconfig* command [111]. The menuconfig command is a menu-based user interface that rebuilds Linux kernel with selected options. Because Amazon

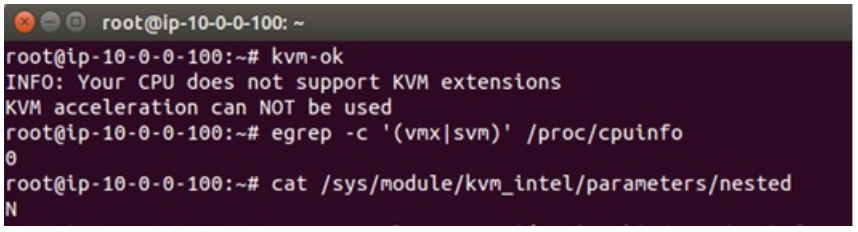


Figure. 4.13 Hardware-assisted virtualization features disabled on EC2

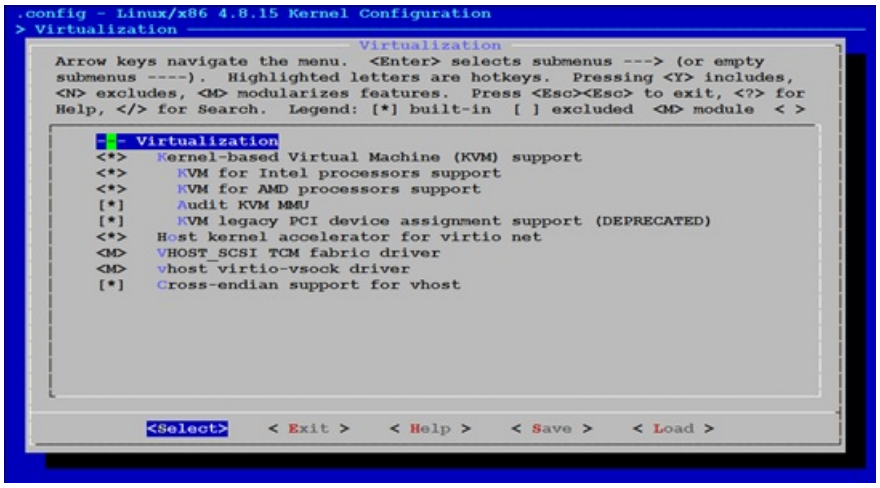


Figure. 4.14 Recompiling the EC2 instance's kernel

instances' BIOS cannot be reached, menuconfig is an alternative tool to enable many hardware features on the instances.

The rebuilding process of the kernel takes about two hours to finish and includes also upgrading the grub file. However, the result of this process changes only the nested virtualization of kvm_intel from 'N' to 'Y'. However, the other two features, the KVM acceleration and the VMX flags number, the process cannot change their values. This solution does not help improve the live migration process.

4.6.2 Compiling Bareflank on Amazon EC2

Bareflank is an open source, lightweight hypervisor, lead by Assured Information Security, Inc [112]. which provides the minimum requirements to install other complete/-complicated hypervisors, such as Xen, KVM and VirtualBox. To enhance Bareflank development, it is written in C++, and it can be run on various operating systems, including Windows and Linux. If the compilation of the hypervisor is successful, it converts the operating system into a VM [112]. It is installed and configured on Amazon c4.2xlarge instance because it can force enabling hardware-assisted virtualization, but the configuration process to convert the operating system to a VM has been repeatedly interrupted and stopped by the Amazon hypervisor, Xen.

Figure. 4.15 The output of running the script on C4 instance

4.6.3 Running a C script on Amazon EC2

Such a script written in C language and it had been used to enable Intel VT-x on Mac Pro and other operating systems in 2008 [113]. This code is programmed to enable hardware-assisted virtualization on the Intel based machines that have theses features in the CPU architecture, but there is no BIOS support for them provided that the BIOS does not lock these features. The result of running the script shows that the BIOS locks these features as shown in Figure 4.15.

There is a potential solution that may help activate this feature. It consists of reprogramming an existing Assembly code written to enable Intel VT-x and EPT on a Windows physical machine [114]. It switches on the values of the machine's CPU control registers to enable nested virtualization features on Amazon. The reprogramming is based on enabling the code to modify the vCPU's control registers values on Amazon Ubuntu instance. Through many attempts to run this code on Amazon, it is proved that it is challenging to master and customize the code.

4.7 Conclusion

Given the current state of public cloud IaaS in terms of hardware-assisted virtualization features, VMs live migration is still challenging to cloud users. LivCloud approach is introduced to help successfully live migrate cloud users' VMs without services disruption across different public cloud providers. The basic design stage of this approach is implemented and evaluated on Amazon M3 and C4 instances. Although, the connectivity is securely maintained between Local-Host and Cloud-Host through Virtual Machine manager and IPsec tunnel, the migration process is not successfully completed due to the lack of nested virtualization feature on Amazon IaaS. We explore 3 developed options to enable nested virtualization on Amazon EC2. None of them have yielded the desired results. In the next chapter, we implement the basic stage on a different provider, Packet bare metal cloud.

Chapter 5

The basic design of LivCloud on Packet

In this chapter, because none of the solutions proposed in Chpater 4 have yielded the desired results, we have moved to a different cloud provider that has the nested virtualization enabled by default. The basic design stage of LivCloud is evaluated on Packet bare metal cloud. The live migration with the basic requirements consists of five steps: (i) installing QEMU-KVM on the host on the local network, *Local-Host* and the host on Packet, *Cloud-Host*; (ii) establishing the connection between the two hosts through IPsec VPN; (iii) connecting the two hosts using virtual machine (VM) manager through SSH protocol; (iv) connecting both hosts to the shared storage on the local network; and (v) performing live migration between the two hosts. Each of these steps is validated using empirical studies. We show for the first time: (i) performing live cloud migration in these five steps; (ii) considering the migrated VM's architecture (32 or 64-bit) and (iii) deploying IPsec VPN tunnel in such environment. Our approach outperforms a number of previous approaches in terms of security and the migrated VMs hardware specifications (RAM & virtual disk sizes) despite its relatively acceptable performance. Furthermore, as far as the literature review of live cloud migration is concerned, it is the first time that the migration channel is protected by two secure layers, IPsec tunnel and SSH protocol.

The organization of this chapter is as follows. Section 5.1 we present an introduction to this chapter. We discuss introduces a brief summary of related work highlighting existing techniques to achieve nested virtualization on the cloud IaaS in Section 5.2. live cloud migration criteria of VMs at cloud IaaS and LivCloud architecture on Packet is presented in Section 5.3. We explain the experiment design of LivCloud in Section 5.4.

The empirical results of the experiment are summarised in Section 5.5. Section 5.6 concludes this chapter.

5.1 Introduction

Various approaches from industry and academia have been developed to address the lack of cloud interoperability. User-centric approaches are among these proposed solutions to achieve live cloud migration for VMs across public cloud IaaS. The implementation of these solutions are still challenging because they are implemented on top of uncontrolled public cloud IaaS. As result, a number of user-centric approaches succeeded in overcoming virtualization heterogeneity by devising a customized nested virtualization, such as paravirtualization and binary translation [17, 22]. However, they suffer limitations in terms of flexibility (decoupling VMs from underlying hardware), performance (migration downtime) and security (secure live migration). These three, flexibility, performance and security are our live cloud migration criteria [1]. They are used to identify the mentioned limitations and design our approach, *LivCloud* [1]. LivCloud is designed to address the limitations of the previous approaches. It is designed in two stages, the basic design and the enhancement of the basic design.

In this chapter, the basic design of LivCloud is implemented and evaluated. Our approach achieves better results in terms of flexibility and security. With respect to security, IPsec VPN is used for the first time in such an environment and it has no effects on performance. A study in [88] shows that the downtime is increased about 4 times when IPsec VPN is considered in live migration. The study illustrates the increase of both migration downtime and total time migration, from less than two seconds to almost 8 seconds downtime when IPsec VPN is implemented. Furthermore, it is the first time that the VM's architecture (64-bit or 32-bit) is taken into consideration in live cloud migration.

5.2 Related work

The literature review reveals that there are two of user-centric approaches that aim to achieve live cloud migration at public cloud IaaS that has nested virtualization by default. Both approaches are explained in Chapter 2. In [21], an approach that is implemented on OpenStack-based infrastructure. It uses a virtual switch and a Linux container (LXC) to live migrate nested VMs within the cloud. The approach cannot run on a variety of OS (i.e., Windows) because the containers (LXC) are Linux-based [49]. In [52], a provider-centric approach is designed and evaluated in a controlled environment. It needs the provider's agreement to be implemented on their IaaS. It introduces Dichotomy which uses a new nested virtualization technology (ephemeral virtualization). This technology transfers control of VM between a layer one hypervisor (the cloud provider's hypervisor) and a second lightly modified hypervisor using memory mapping techniques. Dichotomy is implemented and tested in QEMU-KVM [52].

5.3 LivCloud architecture on Packet

The LivCloud design is distilled into two stages, basic design and the enhancement of the basic design [7]. The basic design stage helps fulfill F1, F2, F3, P1, S1 and S2. The main motivation behind conducting live migration between LivCloud and public cloud IaaS across the Internet is to illustrate that LivCloud basic design can be implemented on an uncontrolled environment, cloud IaaS without any enhancements from the next stage of LivCloud. Also, to prove that it can achieve better results than previously proposed live migration approaches in terms of flexibility and security.

In this development stage, LivCloud is connected to the cloud IaaS through *nested virtualization* and *secure network connectivity*. Firstly, nested virtualization is achieved by configuring QEMU-KVM on LivCloud and public cloud IaaS. Nested virtualization is configuring one hypervisor (in the upper layer) within a virtual machine hosted by another hypervisor [24]. Most of legacy hypervisors, such as QEMU-KVM, Xen and VMware can run nested virtualization [25]. LivCloud uses QEMU-KVM as its hypervisor on the both sides. Virtual machine manager is a user interface for managing virtual machines mainly on QEMU-KVM. Any physical or virtual machine that has QEMU-KVM configured can be connected locally or remotely over SSH to virtual manager

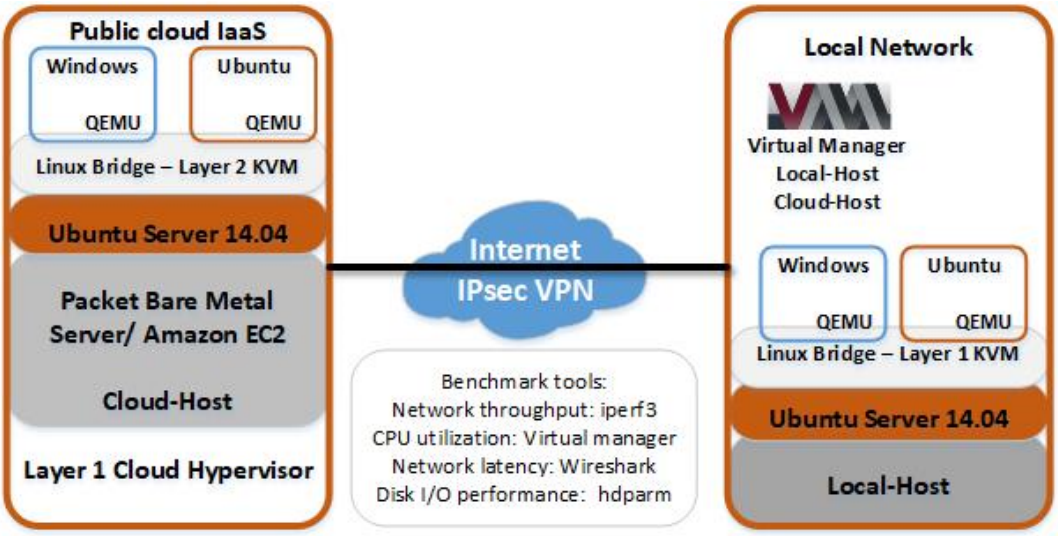


Figure. 5.1 The basic design architecture of LivCloud [7]

[89]. Figure 5.1 shows the basic design and the benchmark tools that are used in the implementation. The basic design architecture is distilled in the following steps:

1. QEMU-KVM is enabled on LivCloud and public cloud IaaS. QEMU-KVM supports running modified and unmodified OS. QEMU has high emulation capability of drivers (i.e network card driver) and KVM provides high acceleration to enhance drivers performance. Also, KVM needs to access the underlying CPU architecture to pass it to the virtualized CPU of the hosted VMs [20, 47].

2. IPsec VPN tunnel is configured to fulfill S1 and S2. The secure connection between LivCloud and IaaS is an essential part of live cloud migration.

3. Both sides are connected to virtual manager in order to live migrate VMs between LivCloud and cloud IaaS.

4. Both sides are connected to the shared storage on the local network.

5. At this stage, *Dynamic DNS* is used to maintain the migrated VM's connections and configurations (P1). Dynamic DNS is used to keep a domain name pointing to the same physical or virtual server connected to the Internet regardless of any IP addresses changes [109].

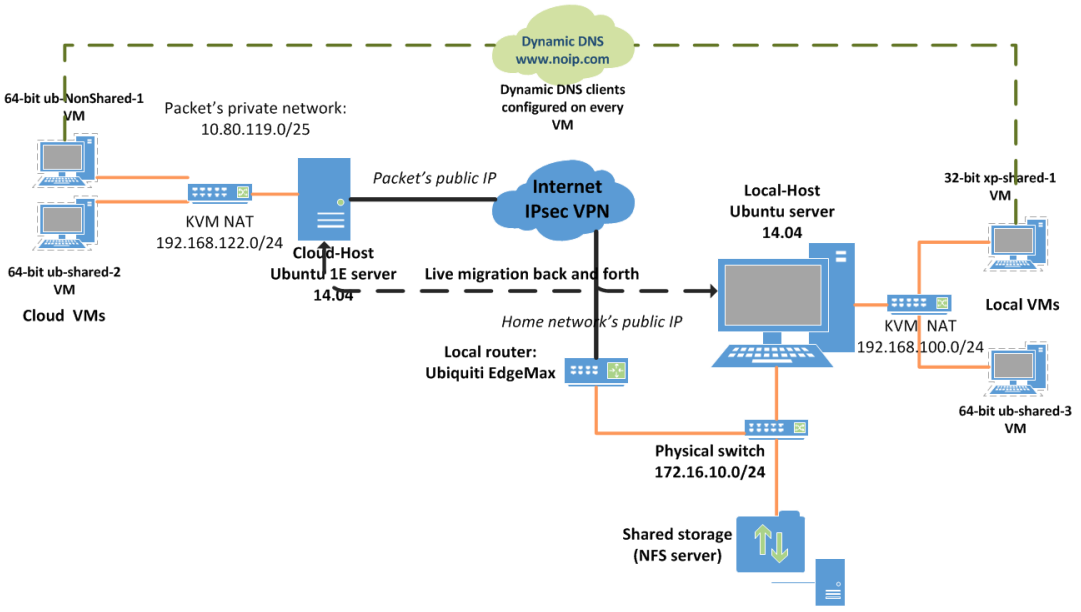


Figure. 5.2 The basic design implementation on Packet

5.4 Experimental design

To implement the basic design stage, a private network (172.16.10.0/24) based in Bournemouth, UK which has two physical servers (Local-Host and NFS server) is connected to a Ubuntu server 14.04 (private address, 10.80.119.3/31) on Packet's datacentre in Amsterdam, Holland.

Moreover, Network throughput, CPU utilization, network latency, migration downtime and disk I/O performance are the main parameters used to analyze the live migration impact. Network throughput is measured using iPerf version3 and network latency is measured by pinging the migrated VM's DNS record. Whereas disk I/O performance is tested on Local-Host and Cloud-Host using *hdparm* command [7].

Packet Bare Metal Cloud provides customers with dedicated single tenant-physical servers [115]. The bare metal server complements or substitutes virtualized cloud services with a dedicated server that eliminates the overhead of virtualization, but maintains flexibility, scalability and efficiency. The server's hardware is fully dedicated and the server can be provisioned using a web-based portal or API, providing access to high-performance dedicated servers on demand [115]. Figure 5.2 shows the basic design implementation on Packet.

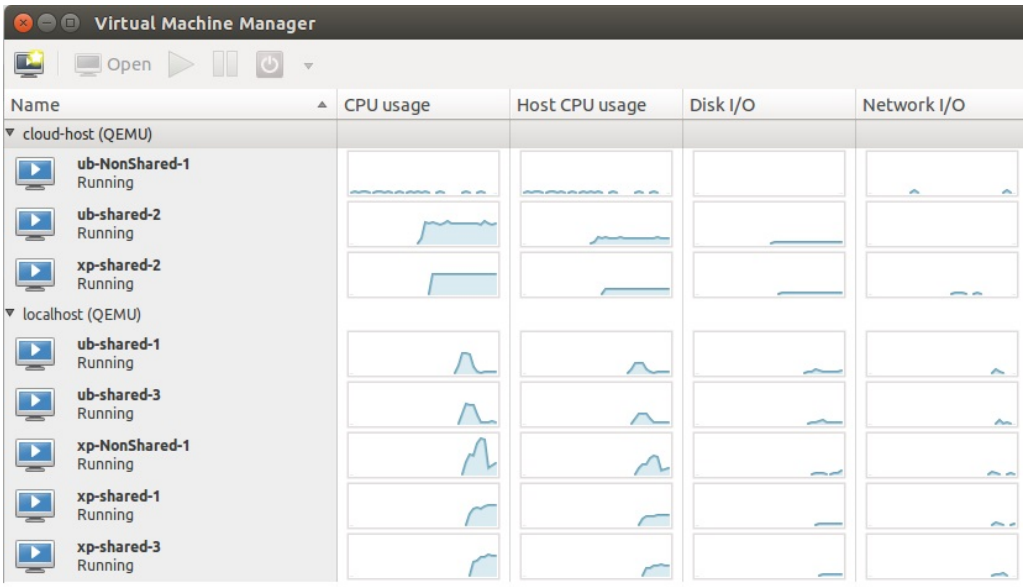


Figure. 5.3 Virtual manager connections to both hosts

5.4.1 Experimental setup

The lab setup as shown in Figure 5.2 consists of one HP Z440 workstation, *Local-Host* is connected to the Internet through EdgeRouter X and Netgear L2 switch providing a 1 Gbps. The workstation has 32 GB of RAM, 1TB disk and 4-core 2.8GHz Intel(R) Xeon(R) E5-1603 v3 CPU. 64-bit Ubuntu Server 14.04 LTS, QEMU-KVM (Layer 1 hypervisor) and QEMU-KVM NAT-based networking are installed and configured on the machine [116]. QEMU-KVM NAT is used here instead of QEMU-KVM Linux bridge due to the fact that Packet provider's private network does not allow layer 2 networking connectivity which is necessary to configure QEMU-KVM bridge [115]. The other machine on the private network is configured as a NFS server (FreeNAS 9.3) for the lab. The Packet 64-bit Ubuntu server 14.04, *Cloud-Host* is connected through two bonded network cards providing a 2 Gbps. The server has 32 GB of RAM, 240 GB disk and 4-physical core 3.4GHz Intel(R) Xeon(R) E3-1240 v3 CPU.

By default nested virtualization or hardware-assisted virtualization features (Intel VT-x, Intel VT-d and Extended Page Tables) are enabled on any Packet server [115]. QEMU-KVM (Layer 2 hypervisor) and QEMU-KVM NAT-based networking are installed and configured on the server. Any VM on either Local-Host or Cloud-Host can be configured with a local disk or a disk hosted on the private network NFS server. The private network and the Packet private network are securely connected via IPsec VPN tunnel. Local-Host and Cloud-host are connected through the tunnel via the virtual machine manager that is installed on Local-Host as shown in Figure 5.3.

In case of Local-Host is temporarily not accessible, both hosts can still be connected via the virtual machine manager installed on Cloud-Host. A remote Ubuntu desktop is installed on Cloud-Host using VNC server (vnc4server) and through TightVNC, cloud users can be remotely connected to Cloud-Host [117]. Six VMs, three 32-bit Windows XP and three 64-bit Ubuntu server 14.04 used as the migrated VMs with shared disks between both hosts. ub-NonShared-1 VM has 8GB of disk hosted on Cloud-Host and xp-NonShared-1 has 5GB of disk hosted on Local-Host. Both VMs are live migrated without shared disks. Table 5.1 shows the migrated VMs and their DNS records.

Table. 5.1 Migrated VMs' specifications and DNS names

DNS records	VM's Architecture	vCPU	RAM (GB)	Virtual disk (GB)	Shared disk/ non-Shared
ub-NonShared-1.ddns.net	64-bit	2	1	8	Non-Shared
ub-shared-1.ddns.net	64-bit	2	1	8	Shared
ub-shared-2.ddns.net	64-bit	2	2	10	Shared
ub-shared-3.ddns.net	64-bit	2	3	10	Shared
xp-NonShared-1.ddns.net	32-bit	1	2	5	Non-Shared
xp-shared-1.ddns.net	32-bit	2	1	8	Shared
xp-shared-2.ddns.net	32-bit	2	2	10	Shared
xp-shared-3.ddns.net	32-bit	2	3	10	Shared

At this stage, *Dynamic DNS* is used to maintain the migrated VMs' connections and configurations (P1). Dynamic DNS is used to keep a domain name pointing to the same physical or virtual server connected to the Internet regardless of any IP addresses changes [104]. no.ip is a dynamic DNS provider that is chosen to register the DNS records as shown in Table 5.1. Dynamic DNS clients (noip-2.1.9-1) are installed and configured on all migrated VMs [104]. Figure 5.4 illustrates how NFS server is connected to Local-Host and Cloud-host. The IPsec VPN tunnel is configured on the private network side. Whereas, on the Packet side, OpenSwan IPsec VPN [44] is configured on Cloud-Host. Appendix B.1 shows the OpenSwan's configurations in more detail.

5.5 Experimental results

5.5.1 Achieving flexibility criteria F1, F2 & F3

Because hardware-assisted virtualization features, Intel VT-x, Intel VT-d and Extended Page Tables (EPT) are enabled on Packet servers, KVM can support a wide range of

hardware architectures, such as CPU registers. This has been proved through live migrating of VMs between Local-Host and Cloud-Host. Both hosts have different architectures, yet the migration does not have any issues in this regard. As a result, F1 has been supported at this stage. KVM supports running modified OS, such as Linux and unmodified OS, such as Windows. Within the evaluation process, it has been possible to live migrate 64-bit Ubuntu VM (modified OS) and 32-bit Windows VM. It is the first time that the VM architecture (64-bit or 32-bit) has been taken into consideration in live cloud migration. Consequently, F2 has been successfully implemented. Table 5.1 highlights the migrated VMs' hardware specifications including virtual disks and RAM sizes. Table 5.1 helps achieve F3.

5.5.2 Achieving performance criterion, P1

For this criterion, live migration must maintain the continuity of delivering the hosted services on migrated VMs. Also, it must keep the existing connections to other VMs and cloud users. QEMU-KVM supports live migration with different networking options, including bridged or routed network and NAT network. The first option, routed

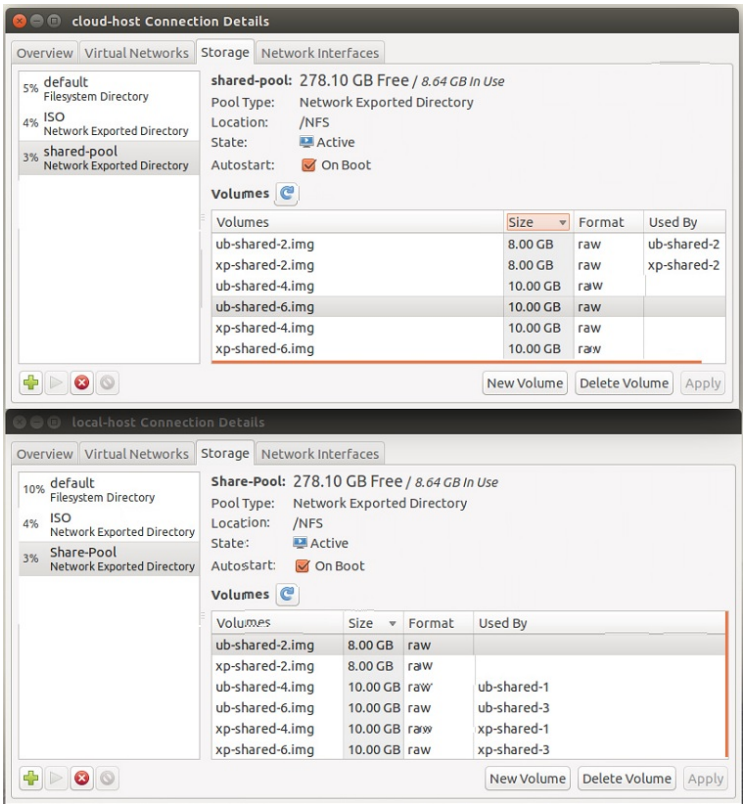


Figure. 5.4 NFS server connections to both hosts

network has not been possible to implement because as mentioned in Section 5.4.1, routed network needs a Linux bridge to be configured as a layer 2 switch, but layer 2 networking is not permissible in Packet network. The NAT is chosen to successfully implement the migration process. The other option to overcome networking issue is configuring an overlay network using the Software Defined Networking SDN protocol, OpenFlow [20, 93], this feature will be implemented in the enhancement of the basic design of LivCloud.

Dynamic DNS records are used to maintain the existing connections to the migrated VMS. VMs' dynamic DNS records are registered on the dynamic DNS provider, noip [109] associated with either the public IP of Local-Host or Cloud-Host. Once the migration to the host is completed, the dynamic DNS client installed on the migrated VMs updates the noip provider with this host's public IP, so that noip updates the name records accordingly. The other VMs and the cloud users are connected to these records not to the IP addresses. Therefore, any changes of public and private IP addresses, the DNS client and noip provider update the records accordingly. QEMU-KVM live migration is copying the RAM and CPU states in a number of iterations while the OS and the applications are running. This means the drivers' states, such as network cards (NICs) stay as they are on the sender side [7]. The migrated VMs' NICs are configured to request IP addresses from the NAT's DHCP server. During the migration, the VMs' NICS need to be triggered to renew their IPs on the receiver's network. To this end, a script written in C language to be run on Windows or Linux, is used to do the following:

1. Continuously testing the Internet connectivity by pinging Google server (8.8.8.8). If connectivity is maintained, the script does nothing.

2. If the connectivity is lost, the script forces the migrated VM to renew the IP address and trigger the dynamic DNS client to update the VM's record on the noip.

The script has the following structure:

During the evaluation process, this script is proved to function properly. The total migration time varies because the VM's hardware specifications and the Internet traffic. For example, live migrating the Ubuntu VM (ub-shared-2: 2GB RAM & 2 vCPU) it takes on average about *8 minutes* in terms of total migration time. Whereas, the XP VM (xp-shared-1: 1 GB RAM & 2vCPU), it takes about *5 minutes* in terms of total migration time. The migration process does not yield the desired results in case of xp-shared-3

Algorithm 2 Steps of C script

```
1:  Input: T
2:  while (true) do
3:      Sleep (T)
4:      if connection to 8.8.8.8 is false then
5:          if (Operation System is Windows) then
6:              - Trigger the network card to renew its IP address
7:              - Re-run Dynamic DNS client
8:          else if (Operation System is Unix) then
9:              - Trigger the network card to renew its IP address
10:             - Re-run Dynamic DNS client
11:         end if
12:     end if
13: end while
```

and xp-NonShared-1. However, the migration downtime in other VMs migration is just about *2 seconds* due to the latency in updating the public IP and the DNS records.

5.5.3 Achieving security criteria, S1 & S2

Figure 5.5 shows how the migration channel in LivCloud is protected. Due to the extra overhead processing and migration downtime added by security mechanism, such as IPsec to live migration, it has been avoided in many live cloud migration approaches. The downtime is increased about 5 times when IPsec added to live migration as the study in [88] shows. The study illustrates the increase of both migration downtime and total time migration, from less than two seconds to almost 8 seconds downtime when IPsec VPN is implemented. However, by comparing a direct ping through the Internet to Cloud-Host's public IP and ping to Cloud-Host's private IP (10.80.119.3) through the IPsec tunnel from Local-Host, the round trip time (RTT) is almost identical in both scenarios. In fact, the connection through the tunnel is slightly faster. Figure 5.6 supports the findings. Consequently, IPsec VPN achieves S1 and S2 in the live migration criteria without any extra penalty in terms of performance. In this design, IPsec starts the connection by exchanging a pre-shared key (password) using Diffie-Hellman protocol (D-H) [118]. On the sender side, D-H generates two keys, public and private keys. Then, it uses the public key to encrypt the pre-shared before sending it to the receiver. The receiver deciphers the pre-shared key using its own D-H private key. The receiver's private key is generated using the sender's public key. If this step is successfully completed, this means S1 is achieved. Now, the two parties have authenticated each other and any data sent between them is encrypted using AES-128 protocol [118].

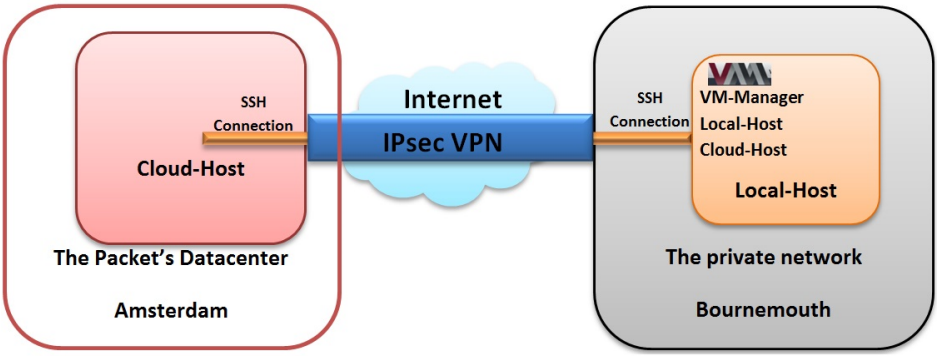


Figure. 5.5 Securing the migration channel via IPsec and SSH

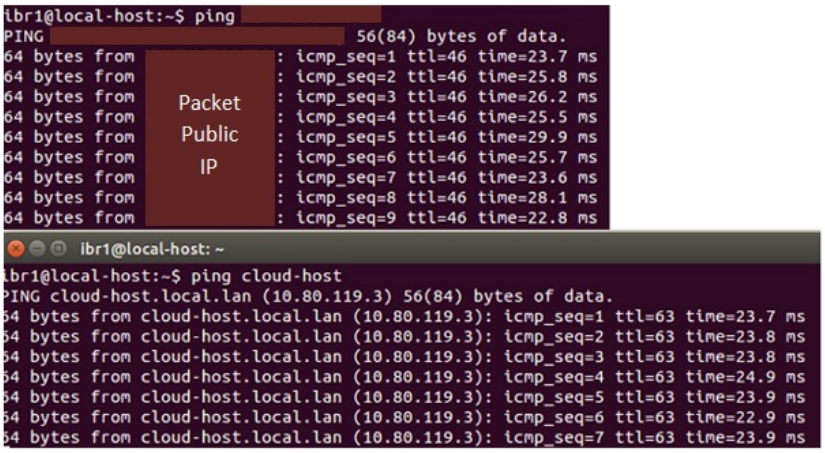


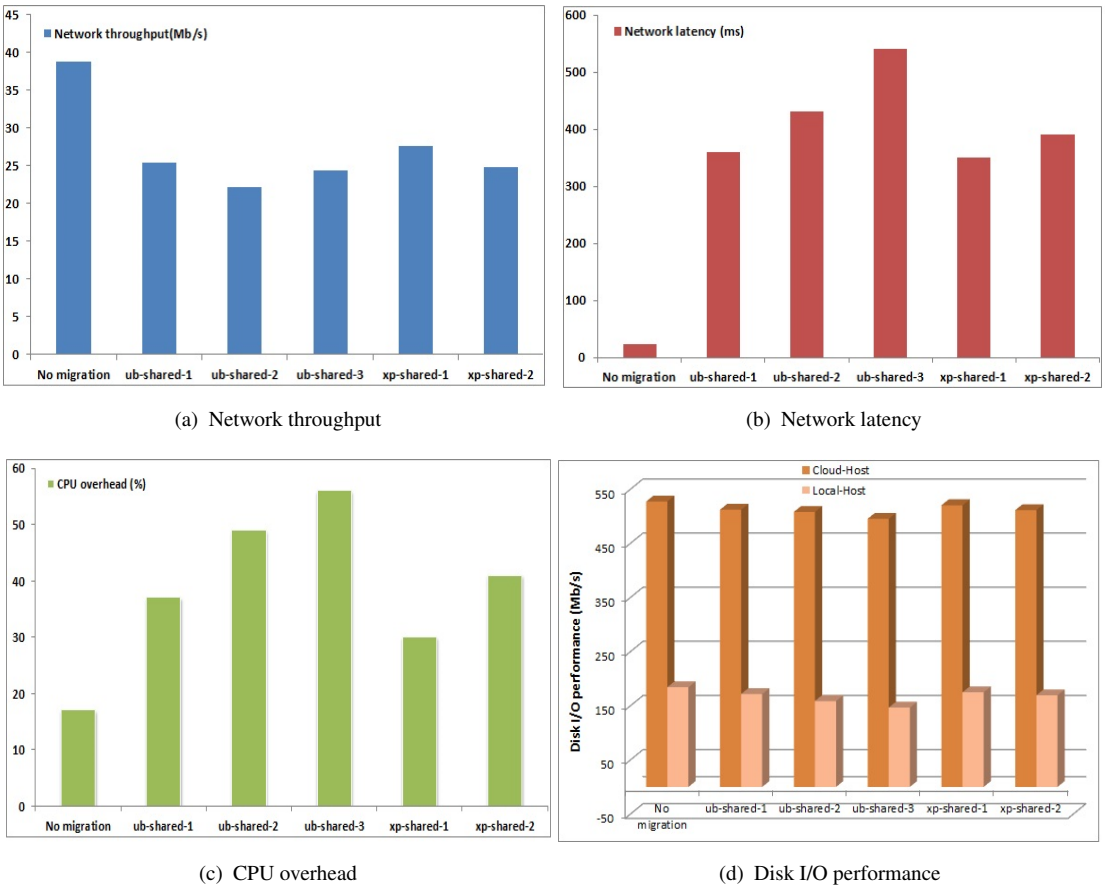
Figure. 5.6 A direct ping latency & IPsec VPN latency

This leads to maintain S2 and provides secure connectivity between the private network in Bournemouth and Cloud-Host in Amsterdam. This is the first layer of protection.

The second layer of protection is secured through the use of SSH protocol and passwords (another layer of encryption and authentication) to connect Local-Host to Cloud Host through IPsec tunnel. Virtual manager is a GUI interface that has many features regarding creating, editing and live migrating VMs. In this design, virtual manager is used to connect Local-Host and Cloud-Host through SSH protocol and VMs' user names and passwords as shown in Figure 5.3. This connection provides a secure SSH tunnel inside the IPsec tunnel.

5.5.4 Discussion

Live migration of the Ubuntu VMs and XP VMs mentioned earlier is performed back and forth between Local-Host and Cloud-Host. The experiments results are the average of conducting the experiment of a total of 15 runs. In terms of the experiment times, it



(a) Network throughput

(b) Network latency

(c) CPU overhead

(d) Disk I/O performance

Figure. 5.7 Results statistics

is done during the morning, afternoon and during the night. This approach is used in any experiments carried out in this research.

Appendix B.1 shows screen shots of performing the migration at different point of time. Only the most notable statistics are summarized in Figure 5.7: Figure 5.7(a) shows that the network throughput is considerably affected, in particular when migrating ub-share-2 VM. This VM has 2GB RAM which is less than ub-shared-3's RAM. It is most likely due to the Internet congestion at that time.

In particular, Figure 5.7(b) shows that there is notable increase in network latency during live migration ub-shared-3 VM because this VM has the largest RAM size, 3GB. The total migration time reaches about 15 minutes in this case. Figure 5.7(c) shows CPU load increases by almost 35% during live migration ub-shared-3. Figure 5.7(d) shows that I/O performance of disks of Local-Host and Cloud-Host are slightly effected by the migration process. As mentioned earlier, there is downtime ($\sim$ 2 seconds) during live migration back and forth between the two hosts. Based on conducting the migration in various time points, the process is slightly faster during the day than the night.

5.6 Conclusion

The evaluation of basic design on Packet shows that IPsec VPN is used for the first time in this environment with no impact on the system performance. Also, the evaluation shows that the migrated VMs' architecture (64 or 32-bit) is considered for the first time in live cloud migration. Finally, the migrated VMs' RAM and disks sizes are larger than any previous approaches. However, LivCloud performance is relatively acceptable due to losing of the connectivity to the migrated VMs for about 2 seconds. Therefore, LivCloud basic design stage is more flexible and more secure than any previous user-centric approach.

The next chapter shows how to implement the enhancement of the basic design of LivCloud on Packet. In this stage, OpenvSwitch (OvS), OpenDayLight (ODL) and Cisco OpenFlow Manager are considered. Theses technologies help enhance the network throughput, maintain the connectivity to the migrated VMs, and eliminate any disconnection between the cloud users and the migrated VMs by redirecting and re-routing the migrated VMs' new locations.

Chapter 6

The enhancement of the basic design on Packet

The implementation of the basic design has been introduced and evaluated on Amazon EC2 and Packet bare metal cloud in Chapter 5 and 4. This chapter discusses the implementation of the second stage, the enhancement of the basic design on Packet. In particular, it illustrates how LivCloud is implemented in two different scenarios. The first scenario deploys KVM bridge networking, OpenvSwitch and C scripts used to meet the network configuration changes during the VMs relocating. This scenario achieves better downtime of one second compared to the basic design of LivCloud. The second scenario uses OpenVPN, OpenDayLight (ODL) and Cisco OpenFlow Manager (OFM) to successfully live migrate VMs back and forth between LivCloud and Packet. This scenario achieves better downtime between 400 and 600 milliseconds. As part of the discussion, we propose a third potential scenario to successfully meet the live cloud migration requirements. This scenario aims to eliminate any downtime occurred in the first two scenarios by utilizing the Open Overlay Router (OOR), Locator Identifier Separator Protocol (LISP) and ODL.

The organization of this chapter is as follows. In Section 6.1, we present an introduction of this chapter. We present LivCloud's architecture that covers the enhancement of the basic design in Section 6.2. We also highlight the experimental setup in this chapter. We discus the implementation of the two live cloud migration scenarios on Packet and the empirical results of the experiments in Section 6.3. In Section 6.4, we conclude this chapter.

6.1 Introduction

Live migration across the Internet takes a notable amount of time due to transferring the storage, limited Internet bandwidth, traffic re-routing, faulty behavior of Internet links and IP address management [57, 119]. It must keep the existing connections of the migrated VM to other VMs and cloud users. As a result, the live migration process can maintain the continuity of delivering the hosted services on migrated VMs. Various approaches from industry and academia have been proposed to improve live cloud migration of VMs at cloud IaaS [12, 16]. The implementation of those solutions are still challenging because they are implemented on top of uncontrolled public cloud IaaS [1]. As a result, a number of approaches succeeded to overcome virtualization heterogeneity by devising Software Defined Networking (SDN) and nested virtualization [20, 22]. However, they suffer limitations in terms of flexibility (decoupling VMs from underlying hardware), performance (migration downtime) and security (secure live migration). Our proposed live cloud migration, LivCloud, considers these three criteria as critical. It is designed over two stages, the basic design and its enhancement of the basic design. The basic design has been implemented and evaluated in a previous paper [56].

The basic design evaluation outperforms a number of previous approaches in terms of security and the migrated VMs hardware specifications (RAM & virtual disk sizes) despite its relatively acceptable performance (downtime of 2 seconds).

In this chapter, the enhancement of the basic design is introduced and evaluated by conducting live cloud migration in two different scenarios. Despite both scenarios achieve better downtime than the basic design stage, Dynamic DNS and a script written in C are still needed to successfully finish the process. As a result, a third potential scenario is proposed to tackle these limitations of the first two scenarios by:

1. Using IPsec VPN and OpenvSwitch (OvS) [90].
2. Using OpenVPN Ethernet Bridging [120], OvS, Cisco OpenFlow Manager (OFM) [8] and OpenDayLight (ODL) controller [121].
3. Introducing ODL, OvS, Locator Identifier Separator Protocol (LISP) [54] and Open Overlay Router (OOR) [122].

With respect to security, IPsec VPN is used for the first time in such an environment and it has no effect on performance. A study in [7] shows that fully live migrating VMs

with their virtual disks and large RAM is still an ongoing effort to tackle instability and performance. Hence, the next step is implementing LivCloud using LISP, ODL, OvS and OOR.

The literature review reveals that there are a number of approaches that aim to achieve live cloud migration using SDN technologies such as OpenFlow protocol. In [53], an SDN architecture named, LIME, is introduced to live-migrate VMs and virtual switches. In [20], an interesting approach is introduced which is implemented on top of a number of cloud providers, including Amazon EC2, Rackspace and HP Cloud. Another approach in [54] proposes an open LISP implementation for public transportation based on Open Overlay Router with an SDN controller, OpenDayLight. In [57], Migration of a VM cluster is suggested to various clouds based on different constraints such as computational resources and better economical offerings. It is designed based on SDN OpenFlow protocol and allows VMs to be paired in cluster groups that communicate with each other independently of the cloud IaaS. In [66], an IaaS framework with regional datacentres for mobile clouds is presented. It is designed based on software-defined networking (SDN) to address the network bandwidth consumption during migration.

Finally, virtual network migration is designed and tested on the Global Environment for Networking Innovation (GENI) [58, 59] which is Wide-Area SDN-enabled infrastructure. All these approaches are disused in more detail in Chapter 2.

6.2 The final configurations of LivCloud

The LivCloud design is distilled into two stages: basic design and the enhancement of the basic design [7]. The basic design stage helps connecting the local network to the cloud IaaS through *nested virtualization* and *secure network connectivity*. Firstly, nested virtualization is achieved by configuring QEMU-KVM on the local network and public cloud IaaS. Nested virtualization is configuring one hypervisor (in the upper layer) within a virtual machine hosted on another hypervisor [24]. Most of legacy hypervisors, such as QEMU-KVM, Xen and VMware can run nested virtualization [25, 47]. LivCloud uses QEMU-KVM as a hypervisor on both sides. Virtual machine manager is a user interface for managing virtual machines mainly on QEMU-KVM. Any physical or virtual machine that has QEMU-KVM configured can be connected locally or remotely over SSH to virtual manager [89]. The basic design has been implemented and tested on Amazon EC2 [56].

At this development stage, an enhancement of basic design of LivCloud is implemented. It deploys various technologies such as OpenDayLight (ODL), OpenFlow and LISP protocols to:

1. Enhance network throughput.
2. Maintain VMs connections and configurations.
3. Reserve resources and prediction of potential failure.

Figure 6.1 shows the final configurations of LivCloud. Live cloud migration is implemented and evaluated in Scenario 1 and Scenario 2. The next section explains these scenarios in more detail. Both scenarios are built and tested on a general experimental setup that can be distilled as follows:

1. QEMU-KVM is enabled on the local network and public cloud IaaS. QEMU-KVM supports running modified and unmodified OS. QEMU has high emulation capability of drivers (i.e network card driver) and KVM provides high acceleration to enhance drivers performance. Also, KVM needs to access the underlying CPU architecture to pass it to the virtualized CPU of the hosted VMs [7, 17].
2. IPsec VPN tunnel is configured to secure the migration. The secure connection between local network and Packet's network is an essential part of live cloud migration.
3. Both sides are connected to Virtual Machine Manager (VMM) [89] in order to live migrate VMs between the local network and cloud IaaS.
4. Both sides are connected to the shared storage on the local network.
5. Dynamic DNS is used to maintain the migrated VM's connections and configurations. Dynamic DNS is used to keep a domain name pointing to the same physical or virtual server connected to the Internet regardless of any IP addresses changes [104].

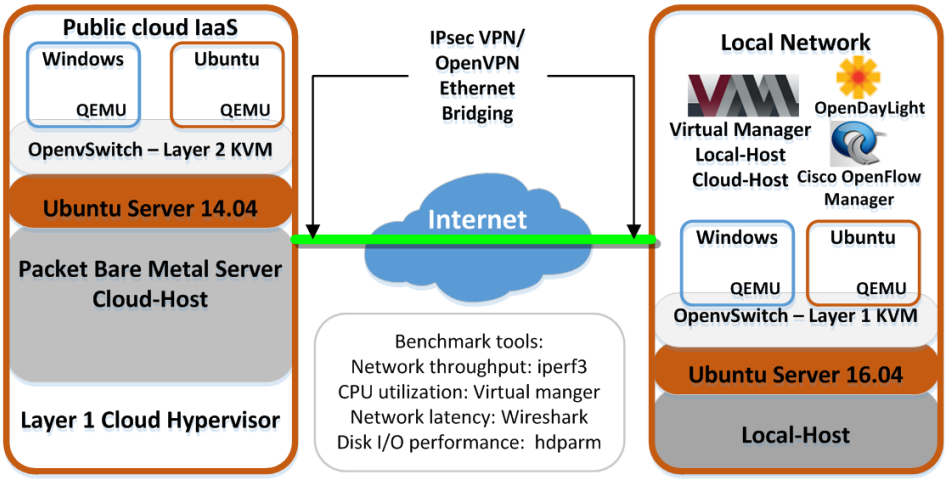


Figure. 6.1 The final configuratrion of LivCloud [7]

6.3 Live cloud migration scenarios

Two different live cloud migration scenarios are implemented and evaluated in this section. These scenarios are chosen to cover the potential solutions of live cloud migration. These solutions may help cloud users live migrate their VMs without any extra costs.

6.3.1 The general experimental setup

This setup is used in both scenarios and some elements may be added or removed accordingly. To implement the general setup, a local network (172.16.10.0/24) based in Bournemouth (UK) which has two physical servers (Local-Host and NFS server) is connected to a Ubuntu server (Cloud-Host) 14.04 (private address, 172.20.20.0/24) on Packet's datacentre in Frankfurt (Germany). Moreover, Network throughput, CPU utilization, network latency, migration downtime and disk I/O performance are the main parameters used to analyze the live migration impact. Network throughput is measured using iPerf [97], while network latency is measured by pinging the migrated VM's DNS record. Disk I/O performance is tested on Local-Host and Cloud-Host using *hdparm* command [99]. If any downtime happens during the process, Wireshark is used to calculate it [98].

Packet Bare Metal Cloud provides customers with dedicated single tenant-physical servers [109]. The bare metal server complements or substitutes virtualized cloud services with a dedicated server that eliminates the overhead of virtualization, but maintains flexibility, scalability and efficiency [115]. Figure 6.2 shows the enhancement

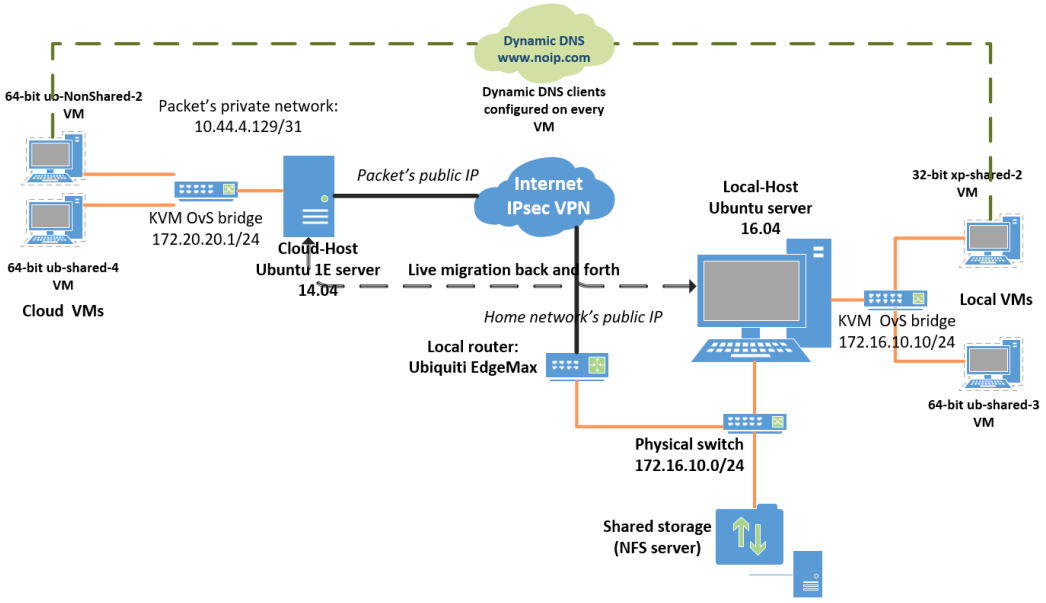


Figure. 6.2 The enhancement implementation on Packet

implementation on Packet.

The lab setup as shown in Figure 6.2 consists of one HP Z440 workstation, Local-Host is connected to the Internet through EdgeRouter X and Netgear L2 switch providing 1 Gbps. The workstation has 32 GB of RAM, 1TB disk and 4-core 2.8GHz Intel(R) Xeon(R) E5-1603 v3 CPU. 64-bit Ubuntu Server 16.04 LTS, QEMU-KVM (Layer 1 hypervisor), OpenvSwitch (OvS) and QEMU-KVM bridged networking are installed and configured on the machine [10]. OvS has flow classification, caching and better performance over the traditional Linux Bridge. Moreover, it has its own load balancer which is used to distribute loads across available routes [90].

The other machine on the private network is configured as NFS server (FreeNAS 9.3) for the lab. The Packet 64-bit Ubuntu TYPE 1E server 14.04, Cloud-Host is connected through two bonded network cards providing 20 Gbps. The server has 32 GB of RAM, 240 GB disk and 4-physical core 2.0GHz/3.4GHz burst Intel E3-1578L v3 CPU. By default nested virtualization or hardware-assisted virtualization features (Intel VT-x, Intel VT-d and Extended Page Tables) are enabled on any Packet server [115]. QEMU-KVM (Layer 2 hypervisor), OpenvSwitch (OvS) and QEMU-KVM bridged networking are installed and configured on the server.

Packet offers various types of bare metal servers including, Type 1 and Type 1E servers which both have similar hardware specifications as specifications of Local-Host [115].

As a result, the live migration has no issues in terms of hardware architecture. Previously, Type 1 in Packet's datacentre in Amsterdam, Holland, was used in implementing the basic design of LivCloud. However, KVM NAT networking had to be configured instead of the bridged option. Packet does not allow layer 2 networking in this type. This made the migration process more complicated in terms of VMs' networking and IPsec VPN configurations.

Type 1E Server is deployed and KVM bridge is possible thanks to the configuration with spare Ethernet network card (eth1) [115]. Layer 2 bridge is implemented through this interface and the cloud private network (172.20.20.0/24) is installed as shown in Figure 6.2. Many configurations are carefully considered including PAT behind the server's public IP and enabling IPv4 forwarding to have the bridge functions correctly. Appendix C.1 shows IPtables configurations which are a crucial part for enabling IPv4 forwarding between both sides.

Any VM on either Local-Host or Cloud-Host can be configured with a local disk or a disk hosted on the local network NFS server. The local network and the Packet private network are securely connected via IPsec VPN tunnel. Local-Host and Cloud-host are connected through the tunnel via the virtual machine manager that is installed on Local-Host. In the case of Local-Host being temporarily not accessible, both hosts can still be connected via the virtual machine manager installed on Cloud-Host. A remote Ubuntu desktop is installed on Cloud-Host using VNC server (vnc4server) and through TightVNC, cloud users can be remotely connected to Cloud-Host [117].

At this setup, *Dynamic DNS* is used to maintain the migrated VMs' connections and configurations. Dynamic DNS is used to keep a domain name pointing to the same physical or virtual server connected to the Internet regardless of any IP addresses changes [104]. no.ip is a dynamic DNS provider that is chosen to register the DNS records. Dynamic DNS clients (noip-2.1.9-1) are installed and configured on all migrated VMs [109]. Dynamic DNS records are used to maintain the existing connections to the migrated VMS. VMs' dynamic DNS records are registered on the dynamic DNS provider, noip [109] associated with either the public IP of Local-Host or Cloud-Host. Once the migration to the host is completed, the dynamic DNS client installed on the migrated VMs updates the provider with this host's public IP, so that the name records are updated accordingly. The other VMs and the cloud users are connected to these records not to the IP addresses. Therefor, any changes of public and private IP addresses, the

DNS client updates the records accordingly. Table 6.1 shows the migrated VMs' specifications, VMs' architecture and associated DNS names. As far as the related literature is concerned, the VMs' specifications are the highest in this environment.

Table. 6.1 Migrated VMs' specifications and DNS names

DNS records	VM's Architecture	vCPU	RAM (GB)	Virtual disk (GB)	Shared disk/ non-Shared
ub-NonShared-2.ddns.net	64-bit	2	2	12	Non-Shared
ub-shared-2.ddns.net	64-bit	2	2	15	Shared
ub-shared-3.ddns.net	64-bit	2	3	15	Shared
ub-shared-4.ddns.net	64-bit	2	4	15	Shared
xp-NonShared-2.ddns.net	32-bit	2	2	10	Non-Shared
xp-shared-2.ddns.net	32-bit	2	2	15	Shared
xp-shared-3.ddns.net	32-bit	2	3	15	Shared

6.3.2 Scenario 1:

The general setup described in Section 6.3.1 is used in this scenario without adding any technology to successfully live migrate the VMs mentioned in Table 6.1. QEMU-KVM supports live migration with different networking options, including bridged network and NAT network. Bridge network has successfully been implemented as mentioned in Section 6.3.1. Packet offers various server types including 1E server that its networking setup allows OpenvSwitch and the bridge configurations. QEMU-KVM live migration copies the RAM and CPU states over a number of iterations while the OS and the applications are running. This means the drivers' states, such as network cards (NICs) stay as they are on the sender side [56]. The migrated VMs' NICs are configured to request IP addresses from the NAT's DHCP server. During the migration, the VMs' NICS need to be triggered to renew their IPs on the receiver's network. To this end, we have written a script in C language to be run on Windows or Linux to enable the following:

1. Continuously testing the Internet connectivity by pinging Google server (8.8.8.8). If connectivity is maintained, the script does nothing.

2. If the connectivity is lost, the script forces the migrated VM to renew the IP address and trigger the dynamic DNS client to update the VM's record on the noip.

The script has the following structure:

Algorithm 4 Steps of C script in Scenario 1

```
1:  Input:
2:  while (true) do
3:      Sleep (T)
4:      if connection to 8.8.8.8 is false then
5:          if (Operation System is Windows) then
6:              - Trigger the network card to renew its IP address
7:              - Re-run Dynamic DNS client
8:          else if (Operation System is Unix) then
9:              - Trigger the network card to renew its IP address
10:             - Re-run Dynamic DNS client
11:         end if
12:     end if
13: end while
```

The total migration time varies because of the VM's hardware specifications and the Internet traffic. For example, live migrating the Ubuntu VM (ub-shared-4: 4GB RAM & 2 vCPU) takes on average about *7 minutes* in terms of migration time. The XP VM (xp-shared-2: 2 GB RAM & 2vCPU) takes about *3 minutes*. Unfortunately, the migration process does not yield the desired results in case of xp-shared-3 and xp-NonShared-2. However, the migration downtime in other VMs migration is just under *one second* due to the latency in updating the public IP and the DNS records.

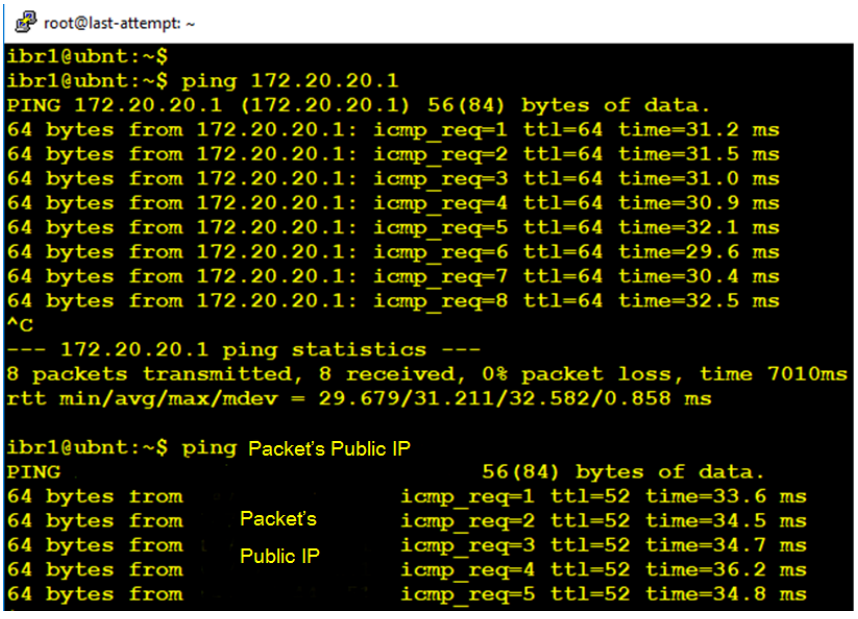


Figure. 6.3 A direct ping latency & IPsec VPN latency

Due to the extra overhead processing and migration downtime added by security mechanism, such as IPsec to live migration, it has been avoided in many live cloud migration approaches. The downtime is increased about 5 times when IPsec added to live migration as the study in [88] shows. The study illustrates the increase of both migration downtime and total time migration, from less than two seconds to almost 8 seconds downtime when IPsec VPN is implemented. However, by comparing a direct ping through the Internet to Cloud-Host's public IP and ping to Cloud-Host's private IP (172.20.20.1) through the IPsec tunnel from Local-Host, the round trip time (RTT) is almost identical in the first and the second scenarios. In fact, the connection through the tunnel is slightly faster. Figure 6.3 shows a direct ping latency & IPsec VPN latency .

Due to the extra overhead processing and migration downtime added by security mechanism, such as IPsec to live migration, it has been avoided in many live cloud migration approaches. The downtime is increased about 5 times when IPsec is added to live migration as in [88]. The study illustrates the increase of both migration downtime and total time migration, from less than 2 seconds to almost 8 seconds downtime when IPsec VPN is implemented. However, by comparing a direct ping through the Internet to Cloud-Host's public IP and ping to Cloud-Host's private IP (172.20.20.1) through the IPsec tunnel from Local-Host, the round trip time (RTT) is almost identical in the first and the second scenarios. In fact, the connection through the tunnel is slightly faster. Figure 6.3 shows A direct ping latency & IPsec VPN latency.

6.3.3 Scenario 2:

We update the general setup with the following technologies, **OpenVPN** [120], **Cisco OpenFlow Manager (OFM)** [8] and **Zodiac-FX OpenFlow** switch [123]. Figure 6.4 shows the changes made in this scenario. OpenVPN has the ability to extend one network across multiple sites (Ethernet bridging) [120]. The local network (172.16.10.0/24) is extended to the cloud network, so the migrated VM has an IP address within the local network range. OFM is connected to OpenDaylight controller through RESTCONF API [8] to re-route the migrated VM internally and Dynamic DNS is used to re-route it externally. This scenario uses OpenVPN tunnel instead of IPsec tunnel. The local network (172.16.10.0/24) is extended using OpenVPN across to Packet's private network (172.20.20.0/24) using **TAP interface** [120]. Zodiac switch is added to the general topology to configure OF protocol. Zodiac switch is connected to ODL [123]. Then, OFM is connected to ODL using RESTCONF API which is an application developed

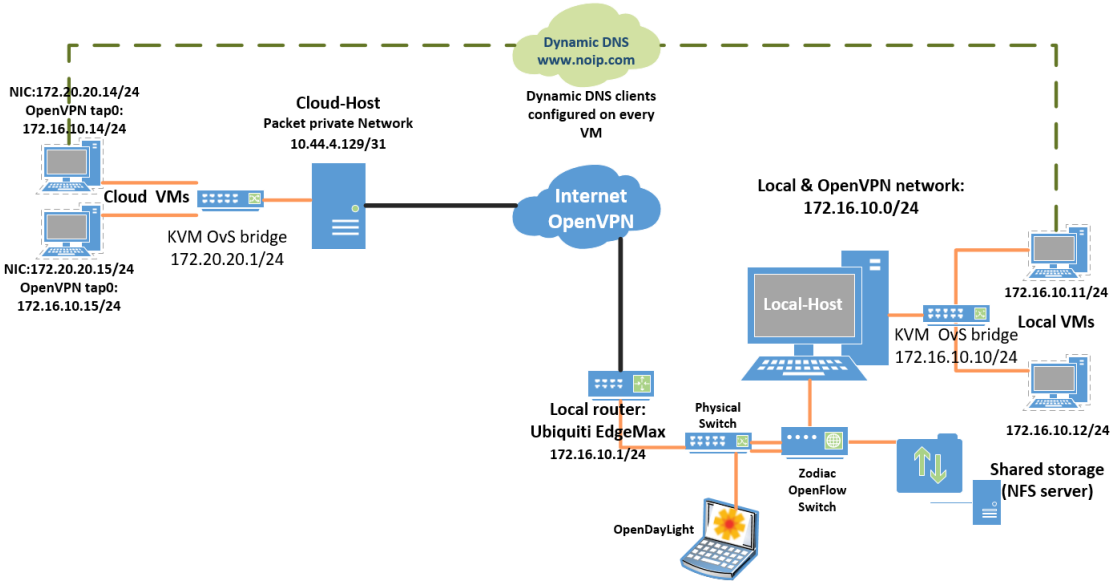


Figure. 6.4 The enhancement implementation on Packet using OpenVPN

by Cisco to run on top of ODL. It visualizes OpenFlow topologies, its program paths and gather its stats [8]. Figure 6.5 shows how OFM is connected to ODL. By configuring OFM, any changes of VMs or hosts location can be re-routed internally through Zodiac switch. However, Dynamic DNS is still needed to re-route the VMs' location to external users. Also, during the migration, the VMs' NICS and OpenVPN client file need to be triggered to renew their IPs on the receiver's network and update the OpenVPN configurations. This requires the modification of the C script used in Section 6.3.2 to yield the desired results.

During the evaluation process, OpenVPN bridging, OFM and the modified script are

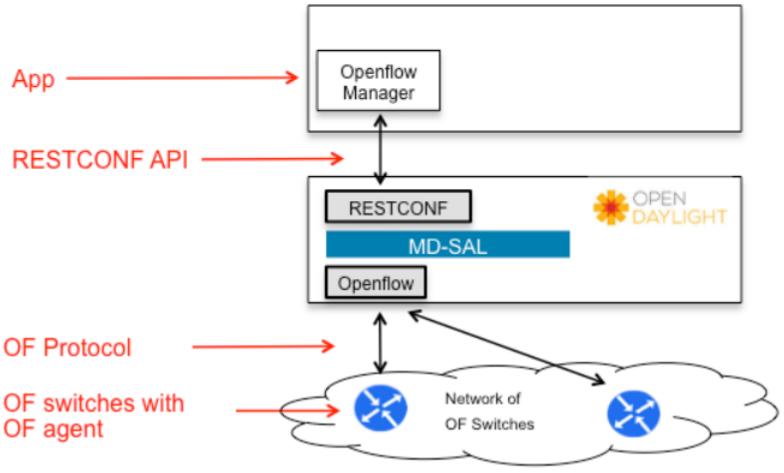


Figure. 6.5 The connection between OFM and ODL [8]

proved to function slightly better than the previous scenario. For example, live migrating the Ubuntu VM (ub-shared-3: 3GB RAM & 2 vCPU) takes on average about *5 minutes* in comparison to 7 minutes in the scenario 1. The XP VM (xp-shared-2: 2 GB RAM & 2vCPU) takes about the same time as the scenarion 1, *3 minutes*. Similar to the scenario 1, the migration process does not yield the desired results when live migrating xp-shared-3 and xp-NonShared-2. However, the migration downtime in other VMs migration mentioned in Table 6.1 is between *400 and 600 milliseconds* due to the latency in updating the public IP and the DNS records. The downtime is about 1 second in Scenario 1. Moreover, OpenVPN Bridging has limitations in terms of scalibility and Maximum Transmission Unit (MTU) tuning [20, 120]. The updated version of the script has the following structure:

Algorithm 6 Steps of C script in Scenario 2

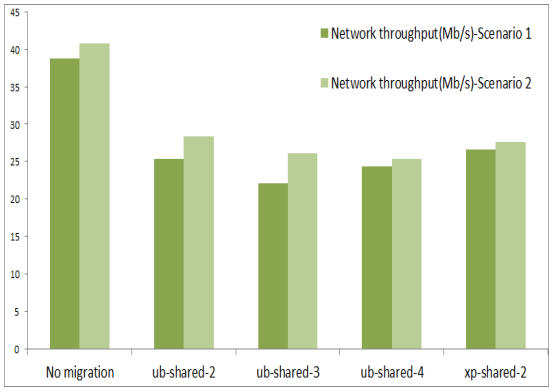
```
1:  Input:
2:  while (true) do
3:      Sleep (T)
4:      if connection to 8.8.8.8 is false then
5:          if (Operation System is Windows) then
6:              - Trigger the network card to renew its IP address
7:              - Re-run OpenVPN client
8:              - Re-run Dynamic DNS client
9:          else if (Operation System is Unix) then
10:             - Trigger the network card to renew its IP address
11:             - Re-run OpenVPN client
12:             - Re-run Dynamic DNS client
13:         end if
14:     end if
15: end while
```

6.3.4 Simulation results

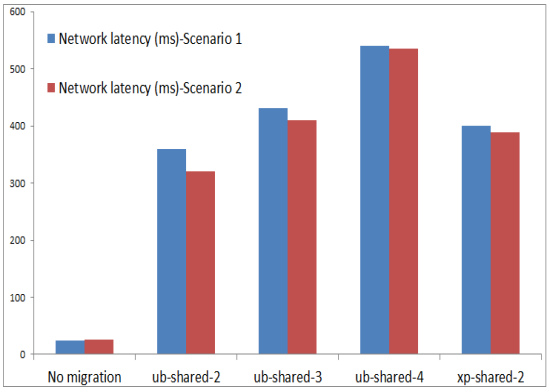
First, we compare Scenario 2 against Scenario 1 with respect to network throughput, network latency, CPU overhead and disk I/O performance. In both scenario, the experiments results are the average of conducting the experiment of a total of 15 runs. In terms of the experiment times, it is done during the morning, afternoon and during the night. This approach is used in any experiments carried out in this research.

Then, live migration of the Ubuntu VMs and XP VMs mentioned earlier is performed back and forth between Local-Host and Cloud-Host in both scenarios. Only the most

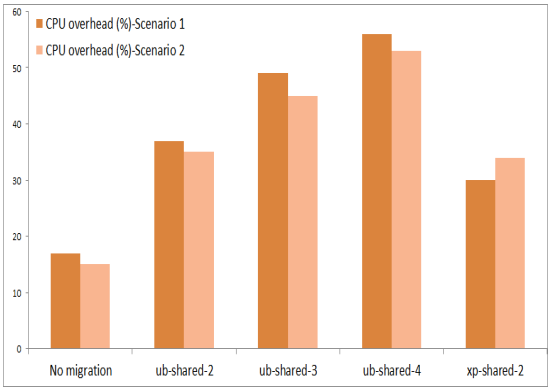
notable statistics are summarized in Figure 6.6. In summary, deploying OpenVPN Bridging is proved to outperform using only IPsec tunnel in all evaluation aspects. Figure 6.6(a) shows that the network throughput is considerably affected in the first scenario than the second scenario. In the first scenario, when migrating ub-share-3 VM that has 3GB RAM, the network throughput VM is more affected than ub-shared-4 that has 4GB RAM. It is most likely due to the Internet congestion at that time.



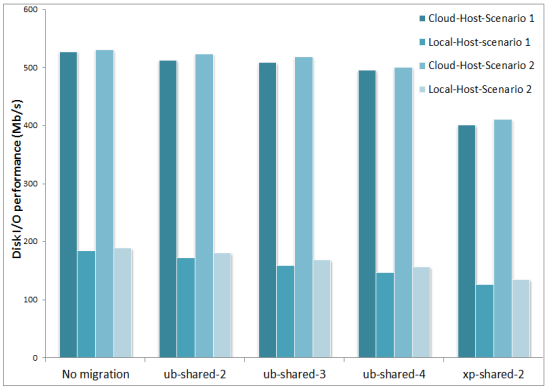
(a) Network throughput



(b) Network latency



(c) CPU overhead



(d) Disk I/O performance

Figure. 6.6 Simulation outcome

In the second scenario, when the VM's hardware size is larger the network throughput decreases. In particular, Figure 6.6(b) shows that there is notable increase in network latency during live migration ub-shared-4 VM in both scenarios because this VM has the largest RAM size, 4GB. The total migration time reaches about 7 minutes in the first scenario and 5 minutes in the second. In case of ub-shared-2 & 3, the network latency is fairly better in the second scenario than the first one.

Figure 6.6(c) shows that CPU load increases by about 39% in Scenario 1 and by 38% in the second one during live migration ub-shared-4. Figure 6.6(d) shows that I/O performance of disks of Local-Host and Cloud-Host are slightly effected by the migration

process in both scenario. However, It is affected more by the first scenario than the second one.

As mentioned earlier, in Scenario 1 there is downtime of roughly 1s during live migration back and forth between the two hosts. The downtime in the second scenario is between 400 to 600 milliseconds, which means using OpenVPN bridging is slightly faster. Table 6.2 shows a comparison between the scenario in Chapter 5 and the two scenarios in this chapter.

Table. 6.2 Summary of analysis results

Scenario in Chapter 5			**Scenario 1**		**Scenario 2**	
Criterion		Details	Criterion	Details	Criterion	Details
F1	✓	Heterogeneous Hardware (CPU architecture (i.e. flags) & Disk I/O drivers)	✓	Heterogeneous Hardware (CPU architecture (i.e. flags) & Disk I/O drivers)	✓	Heterogeneous Hardware (CPU architecture (i.e. flags) & Disk I/O drivers)
F2	✓	Unmodified (Windows) & Modified (Linux)	✓	Unmodified (Windows) & Modified (Linux)	✓	Unmodified (Windows) & Modified (Linux)
F3	✓	Table 5.1 shows VMs' specs	✓	Table 6.1 shows VMs' specs	✓	Table 6.1 shows VMs' specs
P1	✓	Relatively acceptable ($\sim$ 2 seconds downtime)	✓	Acceptable ($\sim$ 1 second downtime)	✓	Acceptable (400-600ms downtime)
P2	×	–	×	–	×	–
P3	×	–	×	–	×	–
S1	✓	IPsec AES-128 for encryption	✓	IPsec AES-128 for encryption	✓	IPsec AES-128 for encryption
S2	✓	IPsec Diffie-Hellman and shared key for authentication	✓	IPsec Diffie-Hellman and shared key for authentication	✓	IPsec Diffie-Hellman and shared key for authentication

6.4 Conclusion

LivCloud is designed to overcome the limitations of previously proposed live cloud migration approaches. The evaluation of enhancement design on Packet shows that live cloud migration can be improved by using various techniques such as, OpenVPN and Software Defined Network (SDN). Also, the evaluation shows the migrated VMs' RAM and disks sizes are larger than the previous stage of LivCloud and any previous approaches. Moreover, this stage performance outperforms any previous approaches. However, there is still improvement needed in maintaining the connectivity to the migrated VMs without using extra techniques such as Dynamic DNS. The migration downtime is most likely due to the time needed by Dynamic DNS to be propagated across both sites.

In a nutshell, we show: (i) performing two successful live cloud migration scenarios; (ii) considering the migrated VM's architecture (32 or 64-bit) and hardware specifications, (iii) deploying ODL and OFM in such environment and (iv) using a customized script to dynamically change network configurations and re-run the OpenVPN.

The next step of running LivCloud on Packet is to implement and evaluate Scenario 3 that is explained in the next chapter. It includes configuring LISP protocol on the OOR to eliminate the need for the customized script and Dynamic DNS. This scenario helps enhance the network throughput, maintain the connectivity to the migrated VMs and eliminate any disconnection between the cloud users and the migrated VMs by redirecting and re-routing the migrated VMs' new locations based on LISP and ODL LISP mapping feature.

Chapter 7

Conclusions and Future Work

This chapter summarizes the contributions of this dissertation and discusses the future directions. The thesis's aim is, to live migrate of VMs across various cloud providers' IaaS with minimal services disruption. To successfully achieve this aim, a number of objectives are considered. They are represented by live cloud migration criteria that consist of flexibility, performance and security criteria. Each criteria have a number of subcriteria, every subcriteria represent an objective towards achieving the main aim. In terms of flexibility, there are three subcriteria; firstly, decoupling the migrated VM from underlying system by supporting wide range of hardware drivers, such as CPU drivers; secondly, supporting various OS on the migrated VM, for instance, Windows; thirdly, considering the migrated VMs hardware specifications including RAM and hard disk size and their architecture, 64 or 32 bit. There are three performance subcriteria: firstly, live migration must be imperceptible to the migrated VM and its users; secondly, predicting the required resources to decide whether or not to proceed with live migration; thirdly, monitoring resource utilization to avoid over utilization and to predict any possible failure. With respect to security, there are security two subcriteria, firstly, maintaining data privacy during live migration using encryption; secondly, imposing authentication during migration.

7.1 Contributions

The research questions that this thesis revolves around achieving live cloud migration of VMs across public cloud IaaS. Many similar studies have been done in such an environment from both industry and academia. Also, Many standards bodies (IEEE,

NIST, DMTF and SNIA) have been pursuing standards to reduce the impact of vendor lock-in. Cloud providers offer their IaaS services based on virtualization to enable multi-tenant and isolated environments for cloud users. Currently, each provider has its own proprietary virtual machine (VM) manager, called the hypervisor. This has resulted in tight coupling of VMs to their underlying hardware hindering live migration of VMs to different providers. A number of user-centric and provider-centric approaches have been proposed to solve this issue.

The main contributions have been the result of four scenarios explained across five chapters answering the research questions. The core idea of these scenarios is to successfully live migrate VMs across various cloud providers with respects to flexibility, performance and security. The process must be done without extra cost and cloud users intervention or their awareness. In order to compare our approach, LivCloud to previous live cloud approaches [20–22], we used live cloud migration criteria mentioned in Table 2.2. Table 7.1 provides a summary of the comparison results between these approaches and LivCloud. Despite the relatively acceptable performance, LivCloud has deployed a better security mechanism than the previous approaches. Moreover, using the security mechanism has not affected the migration performance. Moreover, the migrated VMs' hardware specifications in respect to RAM and disks sizes are larger than the migrated VMs in these approaches. The implementation and the evaluation of LivCloud are explained in Sections 5.4 and 6.2.

Table. 7.1 Summary of analysis results

Supercloud [20]			**Kangaroo [21]**		**HVX [22]**		**Our approach [7]**	
Criterion		Details	Criterion	Details	Criterion	Details	Criterion	Details
F1	✓	Heterogeneous Hardware (CPU architecture (i.e. flags) & Disk I/O drivers)	✓	Heterogeneous Hardware (CPU architecture (i.e. flags) & Disk I/O drivers)	✓	Heterogeneous Hardware (CPU architecture (i.e. flags) & Disk I/O drivers)	✓	Heterogeneous Hardware (CPU architecture (i.e. flags) & Disk I/O drivers)
F2	×	Only modified O/S (Linux)	×	Only modified O/S (Linux)	✓	Unmodified (Windows) & Modified (Linux)	✓	Unmodified (Windows) & Modified (Linux)
F3	×	–	✓	3.2GB virtual disk	×	–	✓	Table 6.1 shows VMs' specs
P1	✓	Relatively acceptable (∼ 1.4 seconds downtime)	✓	acceptable (no downtime)	✓	acceptable (no downtime)	✓	Relatively acceptable (∼ 2 seconds downtime)
P2	×	–	×	–	×	–	×	–
P3	×	–	×	–	×	–	×	–
S1	×	–	×	–	×	–	✓	IPsec AES-128 for encryption
S2	×	–	×	–	×	–	✓	IPsec Diffie-Hellman and shared key for authentication

The main outcome is summarized in the next section.

7.2 Main outcome

- *Contribution 1:* the fist research question, *identifying the key challenges and factors to successfully perform live migration of VMs across different cloud IaaS*, is answered across Chapter 1 and Chapter 2. The outcomes of these chapters are, highlighting the challenges facing the cloud computing at the infrastructure level, a proposal of live cloud migration criteria which are in three categories, flexibility, performance and security. The criteria are the solid foundation on which we evaluate the effectiveness of previous live cloud migration approaches and design our approach, *LivCloud*. Furthermore, across these chapters, we discus the analysis results of certain user-centric approaches that are similar to our approach.

- *Contribution 2:* the second research question, *identifying the limitations of existing live cloud migration approaches and how to design efficient live cloud migration using up-to-date technologies*, is answered across Chapter 2 and Chapter 3. An introduction of our live cloud approach, LivCloud components and how they should interoperate with each other to help the approach function correctly as intended. LivCloud design is distilled into two stages, the basic design and the enhancement of the basic design. Experimental results within the LAN environment help to validate the different components of the design. As part of the experiment, live migration between two different virtualization platforms is evaluated and tested in order to find out how difficult the process across the cloud is.

- *Contribution 3:* the third research question, *how can our approach, LivCloud be assessed on different public cloud IaaS* , chapters 4, 5 and 6 explain in detail how this question is answer as follows.

    1. Implementing the basic design of LivCloud on Amazon EC2 infrastructure. The outcome of this chapter explain various challenges to successfully apply the approach on Amazon. Experimental results on Amazon M3 and C4 instances show the practicality of implementing the proposed design. Although, the networking complexity is dealt with here and the connection is secure, enabling nested virtualization on Amazon is proved to be still challenging. Application and developed model, such as VMware, HQEMU and various coding are used to efficiently enable nested virtualization, but none of them yield the desired result.

2. Implementing the basic stage of our approach on Packet bare metal cloud. This stage is evaluated and tested based on KVM NAT, Linux Bridge and other open-source technologies. Due to using NAT and Internet congestion, the down time (about 2 seconds) is quite notable in this implementation.
3. Implementing the enhancement stage of LivCloud. It gives a better results in terms of down time. Two scenarios are tested and evaluated, scenario 1 has down time of about 1 second and scenario 2 achieves better performance of 400 to 600 milliseconds down time.

7.3 Future work

In this section, we discuss the limitations of the all scenarios in this thesis and propose an alternative potential scenario that copes with these limitations that can be implemented in the near future.

7.3.1 The limitations of scenarios

To successfully finish the live migration, a number of steps have to be considered including Dynamic DNS, the C script and OpenVPN. Yet, there are still challenges to VMs relocating and migration downtime. In a nutshell, we have to implement the following steps to achieve successful migration and maintain the downtime as low as possible:

1. Configuring Dynamic DNS on the migrated VMs and the DNS provider to maintain the external and the internal connections to other VMs and cloud users. As shown in Section 6.3.4, there is still downtime in both scenarios because of updating and propagating any change in Dynamic DNS records.
2. Using the C script to cope with any change in network configurations helps update DNS name records and re-initiate OpenVPN in Scenario 2. These changes should have dynamically happened without any script.

Based on these limitations, the next section discusses two potential scenarios that cope with any of the mentioned challenges.

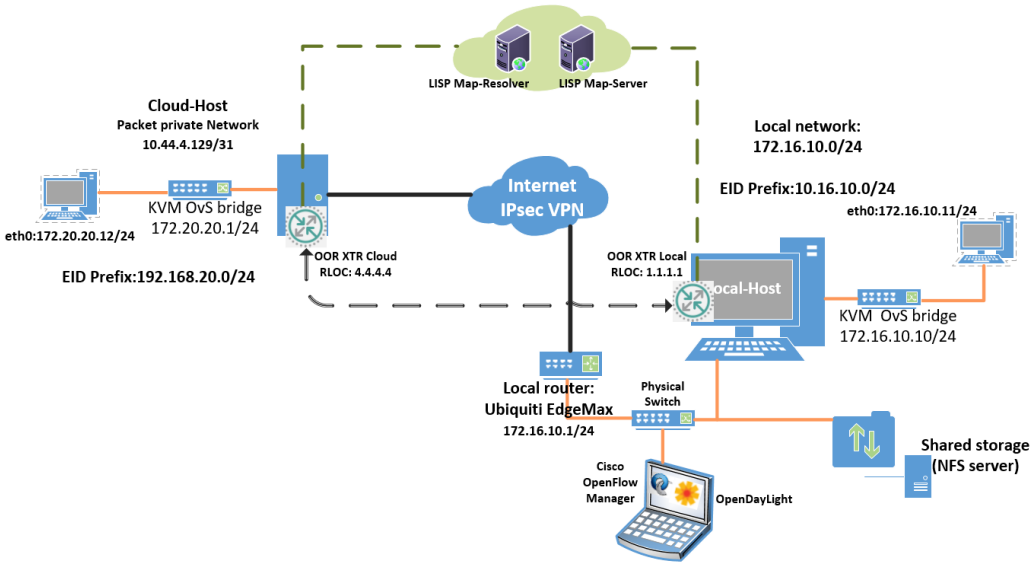


Figure. 7.1 The potential solution on Packet

7.3.2 Future scenario 1:

To improve LivCloud downtime and cope with VMs relocation, an alternative scenario is being investigated. This scenario adds to the general setup Open Overlay Router (OOR) that can be configured to run Locator Identifier Separator Protocol (LISP), OvS, ODL and Cisco OFM. OOR, which is an open source software router to deploy programmable overlay networks. OOR runs LISP to map overlay identifiers to underlay locators and to dynamically tunnel overlay traffic through the underlay network [122]. Figure 7.1 shows the scenario design.

LISP creates two different namespaces: endpoint identifiers (EIDs) and routing locators (RLOCs). Each host is identified by an EID, and its point of attachment to the network by an RLOC. Traffic is routed based on EIDs at LISP sites and on RLOCs at transit networks. At LISP site edge points, ingress/egress tunnel routers (xTRs) are deployed to allow transit between EID and RLOC space.

LISP follows a map-and-encap approach. EIDs are mapped to RLOCs and the xTRs encapsulate EID packets into RLOC traffic. LISP introduces a publicly accessible Mapping System, which is a distributed database containing EID-to-RLOC mappings. The Mapping System consists of both Map-Resolvers (MRs) and Map-Servers (MS). Map-Servers store mapping information and Map-Resolvers find the Map-Server storing a specific mapping [124].

OpenDayLight controller can use the northbound REST API to define the mappings and policies in the LISP Mapping Service. OOR can leverage this service through a southbound LISP plugin. It must be configured to use this OpenDayLight service as their Map Server and/or Map Resolver. The southbound LISP plugin supports the LISP control protocol (Map-Register, Map-Request, Map-Reply messages) and can also be used to register mappings in the OpenDayLight mapping service [125]. Each VM is assigned an EID, which represents the private IP address and RLOC which represents the public IP address. When the migration occurs the RLOC is maintained and the EID is updated through ODL LISP Mapping Service, MRs and MS servers.

The OOR configuration includes setting up two overlay networks, EID prefix (10.16.10.0/24) on the local network and EID prefix (192.168.20.0/24) on the cloud network. At this development stage, Packet's architecture does not allow configuring those prefixes. The configurations need flexibility in layer 2 networking, which is not possible on Packet's datacenters in either Amsterdam or Frankfurt. Both datacentres are the closest to LivCloud's location, Bournemouth, UK. Layer 2 networking is being considered in both centres very soon.

7.3.3 Future scenario 2:

Having evaluated different scenarios in Chapters 3, 4, 5 and 6 of both LivCloud's stages, the optimized and final design has become clearer in terms of the required technologies and their customized configurations. This design has the potential solution of the migration downtime and any possible VM hardware failure.

The scenario explained in Section 7.3.2 has been implemented and tested in a number of attempts, none of them yield the desired results. However, if there was cooperation with the OOR router developers, the desired results would have obtained. Therefore, upon successfully completing this scenario, other technologies explained in Chapter 3, including OpenStack dashboard (Horizon), OpenStack orchestration (Heat), UDT protocol and MPLS, can be integrated to improve the GUI interface, resource management and predication, and performance of live cloud migration.

Configuring OpenStack Horizon needs installing storage node, networking node, identity services (Keystone) and compute node. Cloud-Host and Local-Host are connected through the compute node [9]. Figure 7.2 shows a high level of how various design's

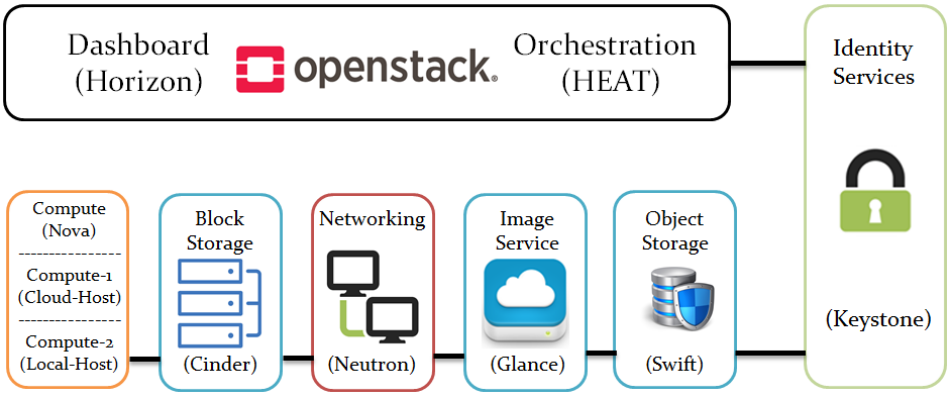


Figure. 7.2 LivCloud future architecture [9]

components are connected using OpenStack Horizon and other OpenSatck's components [9]. The figure shows how Cloud-Host and Local-Host are configure as Compute-1 and Compute-2 nodes in OpenStack dashboard, Horizon. As result, live migration is a Horizon component that it instructs both nodes' hypervisors to perform migration back and forth. Moreover, the figure shows the following components [9, 126]:

- *Identity services(Keystone):* it is an OpenStack identity service that manages user databases and OpenStack service catalogues and their API endpoints. It integrates with existing directory services like LDAP and supports multiple authentication mechanisms, such as username-and-password and token-based systems. It has different components including, user, tenant, role, credentials, authentication, token, service and endpoint.

- *Object storage(Swift):* it provides a cost effective, scalable and fully-distributed API-accessible storage platform that can be integrated directly into applications or used for backup, archiving and data retention. It is equivalent to Amazons S3.

- *Image service(Glance):* it stores virtual machine images in a number of formats. These images are used by compute service to create instances. It is comparable to Amazon Machine Image (AWS AMI).

- *Block storage(Cinder):* it provides persistent block storage to guest virtual machines for expanded storage, better performance and integration with enterprise storage platforms. It is comparable to Elastic Block Storage(AWS EBS).

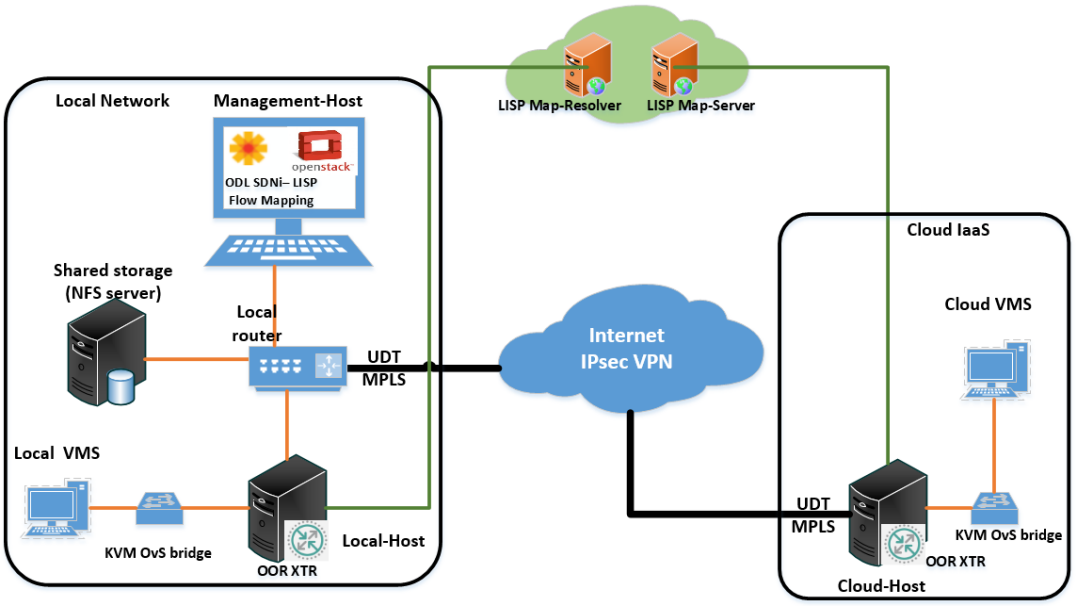


Figure. 7.3 The future implementation of LivCloud

- *Networking(Neutron):* it enables network connectivity interface devices managed by Compute. It enables users to create and attach interfaces to networks. It corresponds to AWS Networking, but it is more complicated and it needs more configurations. I the context of this research, we have worked with both networking and AWS networking is more flexible in terms of customization such as adding and removing any virtual network interface.

- *Compute(Nova):* it provisions instances on user demand. It supports most virtualization technologies. It provides similar functions to Amazon's Elastic Compute Cloud (EC2).

Figure 7.3 highlights technologies mentioned in Chapter 3 and this scenario's technologies combined together to achieve successfully live cloud migration. The server versions of all OpenSatck's components installed on the Management-Host, but the client versions are configured on the Cloud-Host and Local-Host. Once, the configurations are done correctly, Cloud-Host and Local-Host appear in the OpenStack's dashboard as Compute-1 and Compute-2 respectively. Every technology on this design accommodates subcriteria or criteria in live cloud migration criteria. LivCloud basic design needs nested Virtualization which is achieved by customizing QEMU-KVM package(performance criteria, F1, F2 & F3) and network connectivity which is achieved by configuring OpenStack's components and VPC IPsec(P1, S1 & S2).

With respect to the enhancement stage, enhancing network throughput that can be achieved by configuring OvS, MPLS & UDT (Enhancing performance subcriteria, P1); maintaining the migrated VMs connections and configurations by installing LISP Flow Mapping and ODL Inter SDN Controller Communication on ODL as well as configuring LISP on OOR routers(Enhancing performance subcriteria, P1); reserving the required migration resources and prediction of potential failure by installing OpenStack orchestrator, Heat(performance subcriteria P2 & P3).

Appendix A

This appendix highlights configurations related to Chapter 4 including, how Amazon C4 instance had the hardware-assisted virtualization enabled until 20/05/2016 as shown in Figure A.1, networking customization and HQEMU configuration issues.

A.1 C4 instance specifications

Amazon cloud provider offers various EC2 instances (VMs) including general purpose (T & M) and compute optimized (C4 & C5) [127]. In 2015, Amazon announced the availability of C4 instance with certain specifications, including hardware-assisted virtualization [128]. Figure A.1 shows these specifications, which were available on Amazon's website until 20/05/2016. In Section 4.4, we prove that the C4 instance does not have any virtualization related feature by running the related Linux commands and installing KVM that needs these features to run correctly. It would have been very promising from the research perspective to have these features enabled.

C4 Instances

Jan, 2015

C4 instances are ideal for compute-bound applications that benefit from high performance processors. C4 instances are well suited for the following applications:

- Batch processing workloads
- Media transcoding
- High-traffic web servers, massively multiplayer online (MMO) gaming servers, and ad serving engines
- High performance computing (HPC) and other compute-intensive applications

Contents

Hardware Specifications

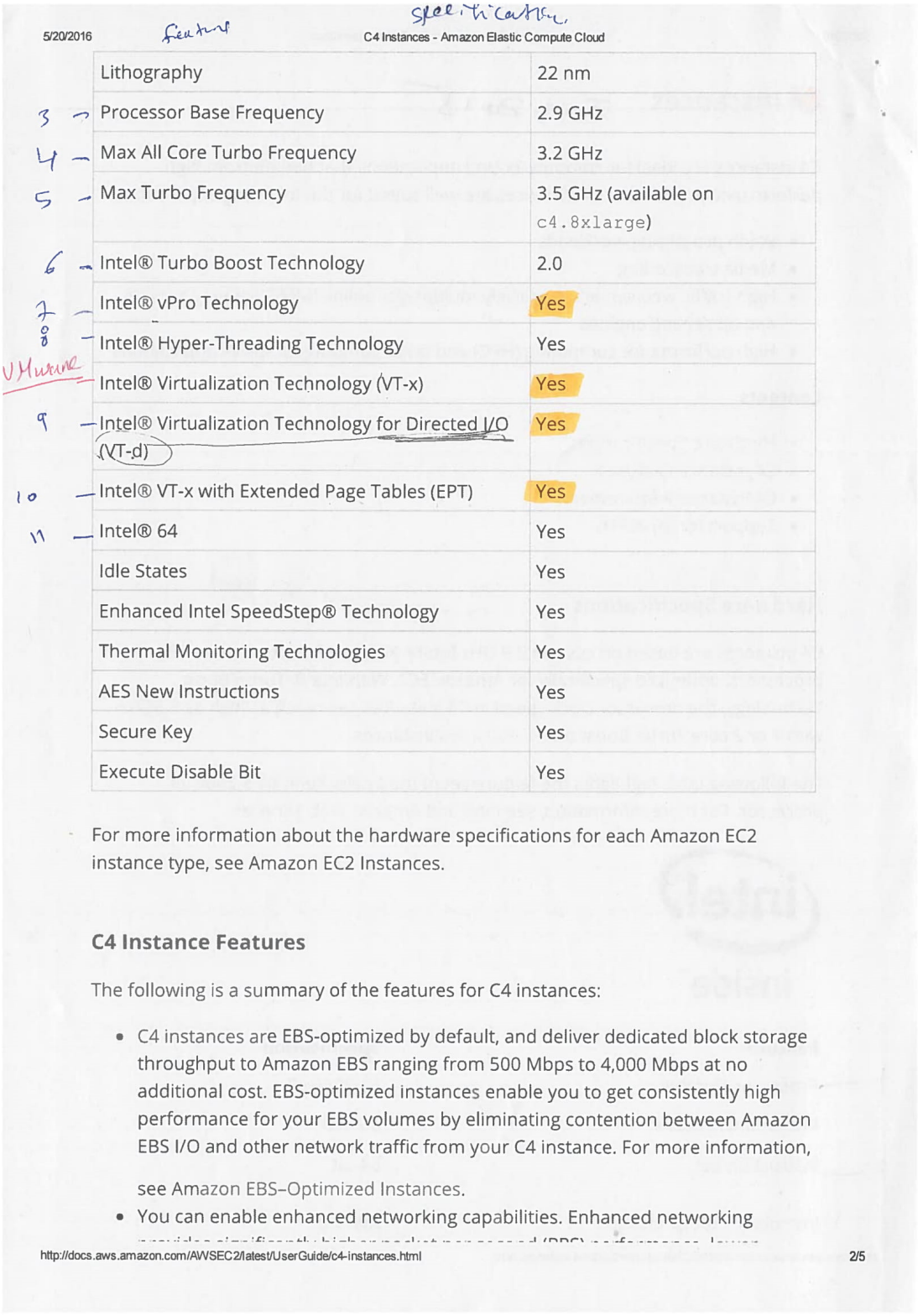
compute-optimized

C4 instances are based on custom 2.9 GHz Intel® Xeon® E5-2666 v3 (Haswell) processors, optimized specifically for Amazon EC2. With Intel® Turbo Boost Technology, the processor clock speed in C4 instances can reach as high as 3.5Ghz with 1 or 2 core Turbo Boost on `c4.8xlarge` instances.

The following table highlights the feature set of the Intel® Xeon® E5-2666 v3 processor. For more information, see Intel and Amazon Web Services.

intel inside™

Feature	Specification
1 Processor Number	E5-2666 v3
Intel® Smart Cache	25 MiB
2 Instruction Set	64-bit
Instruction Set Extensions	AVX 2.0



5/20/2016 C4 Instances - Amazon Elastic Compute Cloud

Lithography	22 nm
Processor Base Frequency	2.9 GHz
Max All Core Turbo Frequency	3.2 GHz
Max Turbo Frequency	3.5 GHz (available on `c4.8xlarge`)
Intel® Turbo Boost Technology	2.0
Intel® vPro Technology	Yes
Intel® Hyper-Threading Technology	Yes
Intel® Virtualization Technology (VT-x)	Yes
Intel® Virtualization Technology for Directed I/O (VT-d)	Yes
Intel® VT-x with Extended Page Tables (EPT)	Yes
Intel® 64	Yes
Idle States	Yes
Enhanced Intel SpeedStep® Technology	Yes
Thermal Monitoring Technologies	Yes
AES New Instructions	Yes
Secure Key	Yes
Execute Disable Bit	Yes

For more information about the hardware specifications for each Amazon EC2 instance type, see Amazon EC2 Instances.

C4 Instance Features

The following is a summary of the features for C4 instances:

- C4 instances are EBS-optimized by default, and deliver dedicated block storage throughput to Amazon EBS ranging from 500 Mbps to 4,000 Mbps at no additional cost. EBS-optimized instances enable you to get consistently high performance for your EBS volumes by eliminating contention between Amazon EBS I/O and other network traffic from your C4 instance. For more information, see Amazon EBS–Optimized Instances.
- You can enable enhanced networking capabilities. Enhanced networking

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/c4-instances.html 2/5

Figure. A.1 C4 instance specifications

A.2 Networking

As mentioned in Section 4.1 regarding adding another interface to the EC2 instance. This interface needs to be bridged to Linux Bridge, br0, so the migrated VMs can attain

private IP address from Amazon DHCP server. Figure A.2 shows the required configurations to bridge br0 through eth1.

```
# The loopback network interface
auto lo
iface lo inet loopback

# Source interfaces
# Please check /etc/network/interfaces.d before changing this file
# as interfaces may have been defined in /etc/network/interfaces.d
# NOTE: the primary ethernet device is defined in
# /etc/network/interfaces.d/eth0
# See LP: #1262951
source /etc/network/interfaces.d/*.cfg

auto br0
iface br0 inet dhcp
   bridge_ports eth1
   bridge_stp on
   bridge_fd 0

up route del -net 172.16.1.0 netmask 255.255.255.0 dev eth0

root@ip-172-16-1-142:~# cat /etc/network/interfaces.d/br0.cfg
# The bridge interface
auto br0
iface br0 inet dhcp
#gateway 172.16.1.142
dns-nameserver 172.16.1.142
dns-nameserver  8.8.8.8
bridge_ports eth1
bridge_stp off
root@ip-172-16-1-142:~# cat /etc/network/interfaces.d/eth0.cfg
# The primary network interface
auto eth0
iface eth0 inet dhcp
root@ip-172-16-1-142:~# cat /etc/network/interfaces.d/eth0.cfg
```

Figure. A.2 Network configuration of the hosted instance, C4

To attach the bridge, br0 to the instance, the following steps and commands are considered:

1. Create a config file for the eth1 and br0 interfaces:

```
cp /etc/network/interfaces.d/eth0.cfg  /etc/network/interfaces.d/eth1.cfg

cp /etc/network/interfaces.d/eth0.cfg /etc/network/interfaces.d/br0.cfg
```

2. Create a custom route table named, out by adding this line, 200 out is added to rt_tables file:

```
nano /etc/iproute2/rt_tables
```

3. Add a rule to route eth1 traffic via default gateway:

```
ip route add default via 10.20.0.1 dev br0 table out
```

4. Add routing rules to route all traffic from/to br0 IP address, which is assigned to the br0 interface, via the out routing table:

```
ip rule add from 10.20.0.11/32 table out
ip rule add to 10.20.0.11/32 table out
ip route flush cache
```

Also, the migrated VMs's networking configurations have to be customized as well. Figure A.3 shows these configurations of XP VM.

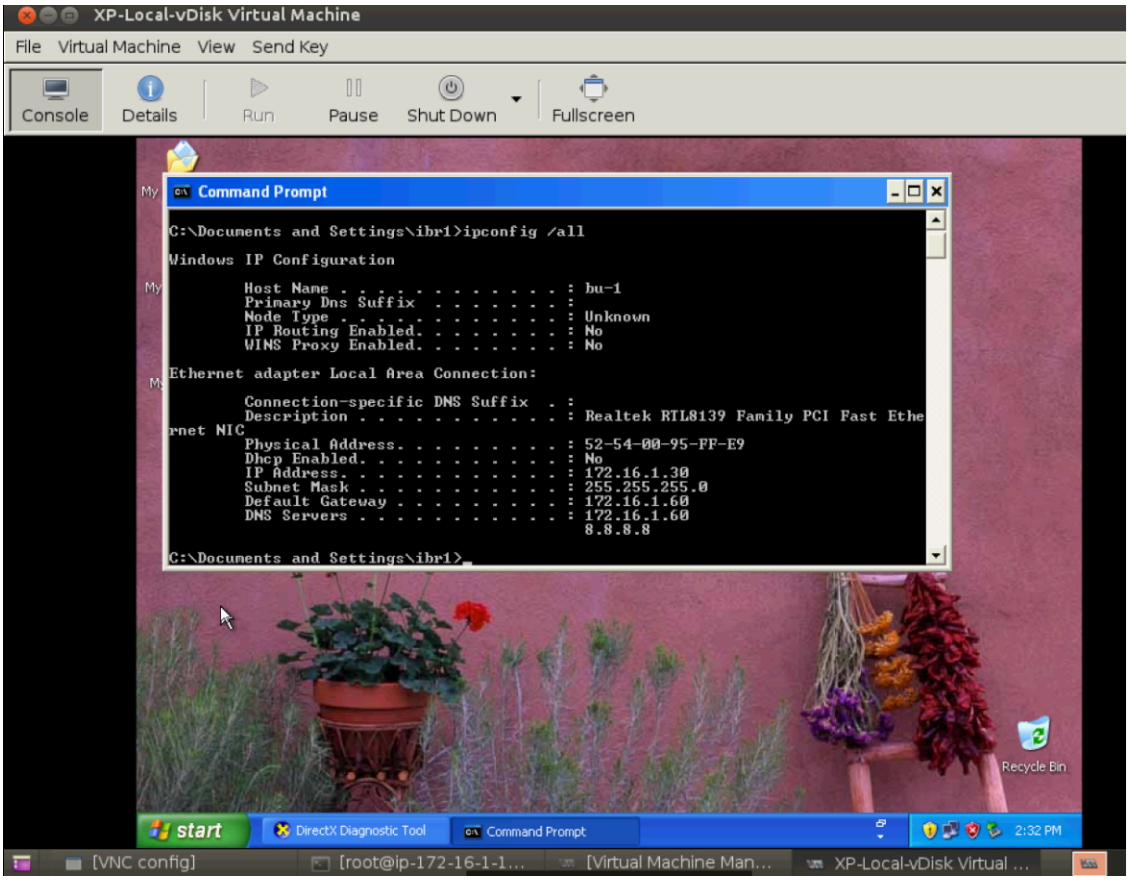


Figure. A.3 Network configuration of a migrated VM, XP on C4 instance

Upon successfully adding the bridge interface, the second step is, to configure NATting between the original interface of the instance, eth0 and br0.

A.3 HQEMU configuration issues

Prior to HQEMU installation and configurations, the instance's kernel has to be recompiled to facilitate HQEMU function on the instance. Figure A.4 highlights the main

commands to perform this operation. This operation takes on average between an hour and half and two hours. In the context of this project, this process has been done tens of times.

```
root@ip-10-0-0-100:~# history
    1  pwd
    2  wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.4.39.tar.xz
    3  tar -Jxvf linux-4.4.39.tar.xz -C /usr/src/
    4  cd /usr/src
    5  cd linux-4.4.39/
    6  make mrproper
    7  cp /boot/config-3.13.0-105-generic ./config
    8  mv config .config
    9  make menuconfig
   10  nproc
   11  make -j2
   12  make modules
   13  make modules_install
   14  make install
   15  modinfo kvm_intel | grep -i nested
   16  cat /sys/module/kvm_intel/parameters/nested
   17  apt-get install qemu qemu-kvm libvirt-bin virt-manager -y
   18  dmesg | grep -e DMAR -e IOMMU
   19  mointel_iommu=on
   20  dmesg | grep -e DMAR -e IOMMU
   21  modprobe pci_stub
   22  modprobe kvm
   23  modprobe kvm-intel
   24  cd sys
   25  ll
   26  cd /sys/module/
   27  ll
   28  apt-get update
   29  uname -r
   30  modprobe kvm
   31  modprobe kvm-intel
   32  lsmod | grep kvm
   33  cat /sys/module/kvm_intel/parameters/nested
   34  echo .options kvm-intel nested=1. > /etc/modprobe.d/kvm-intel.conf
   35  cat /sys/module/kvm_intel/parameters/nested
   36  kvm-intel.nested=1
   37  nano /etc/default/grub
   38  #
   39  update-grub
   40  ll /etc/default/grub.d/
   41  nano /etc/default/grub.d/50-cloudimg-settings.cfg
   42  update-grub
   43  nano /etc/modprobe.d/kvm-intel.conf
   44  cat /sys/module/kvm_intel/parameters/nested
   45  kvm-ok
   46  modprobe kvm
   47  modprobe kvm-intel
   48  lsmod | grep kvm
   49  apt-get install qemu qemu-kvm
   50  cat /etc/modprobe.d/kvm-intel.conf
   51  cat /etc/default/grub.d/50-cloudimg-settings.cfg
   52  cat /etc/default/grub
   53  dmesg | grep -e DMAR -e IOMMU
```

Figure. A.4 The Linux commands to recompile the kernel prior to HQEMU configuration

Appendix B

This Appendix highlights OpenSwan IPsec VPN configurations on Packet. These configurations are used in Chapter 5 and Chapter 6. Also, a number of screen shots of live migrating Ubuntu VM from Cloud-Host to Local-Host are shown in this appendix.

B.1 OpenSwan configuration on Packet

After installing OpenSwan on Cloud-Host, the following files are configured as follows:

1. ipsec.config:
    - config setup
        - nat_traversal=yes
        - oe=off
        - protostack=netkey
        - force_keepalive=yes
        - keep_alive=60
        - nhelpers=0
    - conn Packet2LocalConnection
        - left=10.80.135.131
        - leftsubnet=10.80.0.0/16,172.20.20.0/24
        - leftid=Packet public IP
        - leftsourceip=10.80.135.131
        - leftnexthop=Local network public IP
        - right=Local network public IP

- rightsubnet=172.16.10.0/24
- rightid=Local network public IP
- type=tunnel
- keyexchange=ike
- ikelifetime=480m
- keylife=60m
- ike=aes128-sha1;modp1024!
- phase2=esp
- phase2alg=aes128-sha1;modp1024
- pfs=yes
- forceencaps=yes
- authby=secret
- auto=start

2. ipsec.secrets:
   include /var/lib/openswan/ipsec.secrets.inc

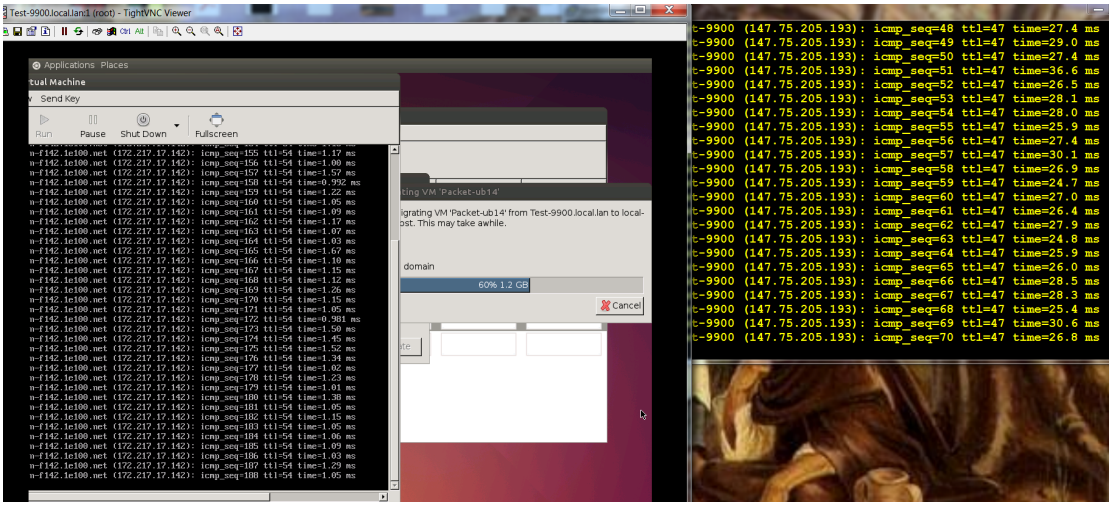
   Local network public IP Packet public IP: PSK ”PaSSwOrD”

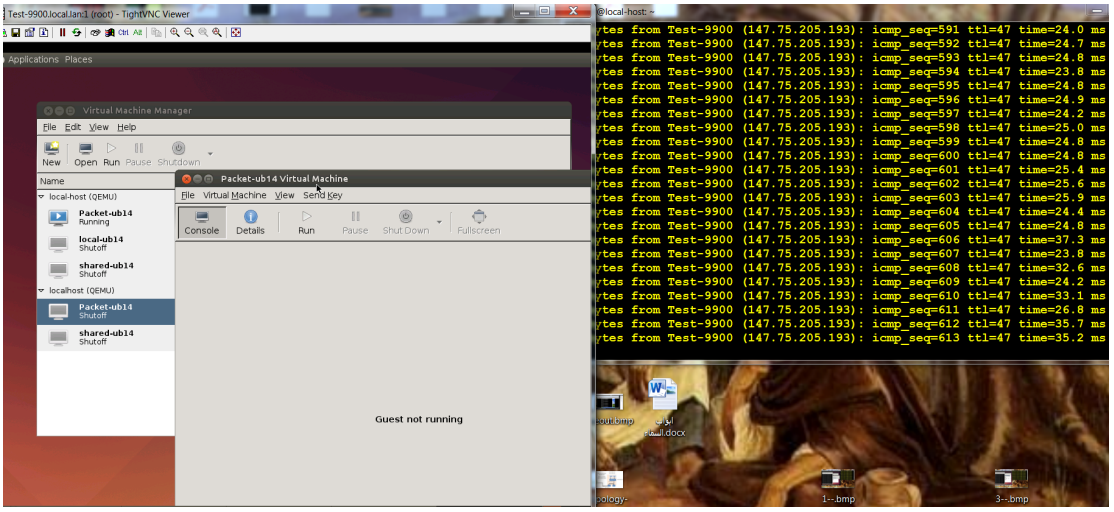
3. rc.local:

   - iptables -t nat -A POSTROUTING -s 10.80.0.0/16 ! -d 172.16.10.0/24 -o bond0 -j MASQUERADE
   - iptables -t nat -A POSTROUTING -s 172.30.255.0/24 ! -d 172.16.10.0/24 -o bond0 -j MASQUERADE

B.2 Screenshots of live migration attempts

The following figures show different points in time until Ubunt VM migrated between Cloud-Host to Local-Host.





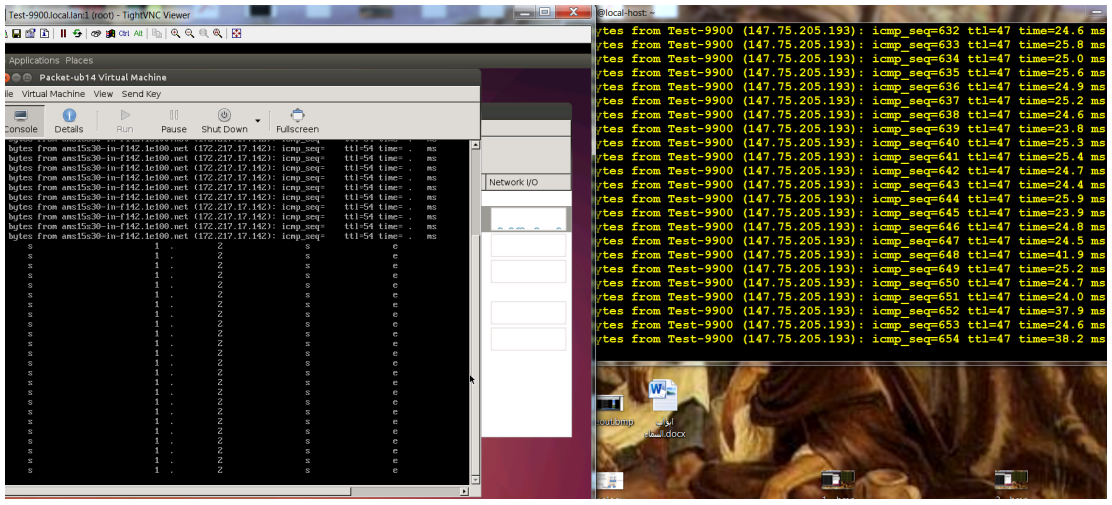


Figure. B.1 Live migration of Ubuntu VM at various points of time

Appendix C

C.1 IPtables configurations on Cloud-Host

The following configurations of IPtables on Cloud-Host show how the network behind KVM bridge, virbr0 is connected through IPsec VPN to the local network. These configurations have taken a number of attempts to correctly connect both sides. Despite Packet server used in Chapter 6 has a second interface attached to it, connecting the network behind the bridge was not possible until we customize IPtables configurations as follows:

```
# Generated by iptables-save v1.4.21 on Wed Nov 29 01:45:47 2017
*nat
:PREROUTING ACCEPT [1:60]
:INPUT ACCEPT [1:60]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -m policy --dir out --pol ipsec -j ACCEPT
-A POSTROUTING -s 172.20.20.0/24 -d 224.0.0.0/24 -j RETURN
-A POSTROUTING -s 172.20.20.0/24 -d 255.255.255.255/32 -j RETURN
-A POSTROUTING -s 172.20.20.0/24 ! -d 172.20.20.0/24 -p tcp -j MASQUERADE
   --to-ports 1024-65535
-A POSTROUTING -s 172.20.20.0/24 ! -d 172.20.20.0/24 -p udp -j MASQUERADE
   --to-ports 1024-65            535
-A POSTROUTING -s 172.20.20.0/24 ! -d 172.20.20.0/24 -j MASQUERADE
-A POSTROUTING -m policy --dir out --pol ipsec -j ACCEPT
-A POSTROUTING -s 10.80.0.0/16 ! -d 172.16.10.0/24 -o bond0 -j MASQUERADE
-A POSTROUTING -s 172.20.20.0/24 -o bond0 -m policy --dir out --pol ipsec -j ACCEPT
-A POSTROUTING -s 172.20.20.0/24 -o bond0 -j MASQUERADE
COMMIT
# Completed on Wed Nov 29 01:45:47 2017
# Generated by iptables-save v1.4.21 on Wed Nov 29 01:45:47 2017
*mangle
:PREROUTING ACCEPT [8476:1260941]
:INPUT ACCEPT [8378:1253221]
:FORWARD ACCEPT [98:7720]
:OUTPUT ACCEPT [9672:4537257]
```

```
:POSTROUTING ACCEPT [9787:4547396]
-A POSTROUTING -o virbr0 -p udp -m udp --dport 68 -j CHECKSUM --checksum-fill
COMMIT
# Completed on Wed Nov 29 01:45:47 2017
# Generated by iptables-save v1.4.21 on Wed Nov 29 01:45:47 2017
*filter
:INPUT ACCEPT [8367:1251212]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [9670:4536601]
-A INPUT -i virbr0 -p udp -m udp --dport 53 -j ACCEPT
-A INPUT -i virbr0 -p tcp -m tcp --dport 53 -j ACCEPT
-A INPUT -i virbr0 -p udp -m udp --dport 67 -j ACCEPT
-A INPUT -i virbr0 -p tcp -m tcp --dport 67 -j ACCEPT
-A INPUT -i virbr0 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -i bond0 -p udp -m policy --dir in --pol ipsec -m udp --dport 1701 -j
   ACCEPT
-A FORWARD -d 172.20.20.0/24 -p icmp -j ACCEPT
-A FORWARD -d 172.20.20.0/24 -p tcp -j ACCEPT
-A FORWARD -d 172.20.20.0/24 -p udp -j ACCEPT
-A FORWARD -d 172.20.20.0/24 -o virbr0 -m conntrack --ctstate RELATED,ESTABLISHED -j
   ACCEPT
-A FORWARD -s 172.20.20.0/24 -i virbr0 -j ACCEPT
-A FORWARD -i virbr0 -o virbr0 -j ACCEPT
-A FORWARD -o virbr0 -j REJECT --reject-with icmp-port-unreachable
-A FORWARD -i virbr0 -j REJECT --reject-with icmp-port-unreachable
-A OUTPUT -o virbr0 -p udp -m udp --dport 68 -j ACCEPT
COMMIT
# Completed on Wed Nov 29 01:45:47 2017
```

Bibliography

[1] Ibrahim Mansour, Reza Sahandi, Kendra Cooper, and Adrian Warman. Interoperability in the Heterogeneous Cloud Environment: A Survey of Recent User-centric Approaches. In *the International Conference on Internet of things and Cloud Computing (ICC2016)*. ACM, 2016.

[2] Hakim Weatherspoon Robbert van Renesse and Zhiming Shen. The Supercloud: Applying Internet Design Principles to Interconnecting Clouds. *Internet Computing*, 2018.

[3] Christophe Crin Thouraya Louati, Heithem Abbes and Mohamed Jemni. LXCloud-CR: Towards LinuX Containers Distributed Hash Table based Checkpoint-Restart. *Journal of Parallel and Distributed Computing*, 2018.

[4] Huansheng Ning Pengfei Hu, Sahraoui Dhelim and Tie Qiud. Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of Network and Computer Applications*, 2017.

[5] Eduard Zharikov Sergii Telenyk and Oleksandr Rolik. An approach to software defined cloud infrastructure management. In *XI International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*. IEEE, 2016.

[6] VMware. Understanding Full Virtualization, Paravirtualization, and Hardware Assist. URL https://www.vmware.com/techpapers/2007/understanding-full-virtualization-paravirtualizat-1008.html.

[7] Ibrahim Ejdayid A. Mansour, Kendra Cooper, and Hamid Bouchachia. Effective live cloud migration. In *4th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 2016.

[8] Cisco OpenFlow Manager. URL https://developer.cisco.com/site/devnetcreations/openflow-mgr.

[9] OpenStack. OpenStack Architecture Design Guide, . URL https://docs.openstack.org/arch-design/.

[10] Rajkumar Buyya, Satish Narayana Srirama, Giuliano Casale, Rodrigo Calheiros, Yogesh Simmhan, Blesson Varghese, Erol Gelenbe, Bahman Javadi, Luis Miguel Vaquero, Marco A. S. Netto, Adel Nadjaran Toosi, Maria Alejandra Rodriguez, Ignacio M. Llorente, Sabrina De Capitani di Vimercati, Pierangela Samarati, Dejan Milojicic, Carlos Varela, Rami Bahsoon, Marcos Dias de Assuncao, Omer Rana, Wanlei Zhou, Hai Jin, Wolfgang Gentzsch, Albert Zomaya, and Haiying Shen. A manifesto for future generation cloud computing: Research directions for the next decade. *Distributed, Parallel and Cluster Computing, Cornell University Library*, 2017.

[11] RIGHSCALE. Cloud computing trends: 2017 state of the cloud survey, 2017. URL https://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2017-state-cloud-survey.

[12] Zhizhong Zhang, Chuan Wu, and David W.L. Cheung. A survey on cloud interoperability: taxonomies, standards, and practice. In *ACM SIGMETRICS Performance Evaluation Review*. ACM, 2013.

[13] Zack Whittaker. Amazon web services suffers outage, takes down vine, instagram, others with it. URL http://www.zdnet.com/article/amazon-web-services-suffers-outage-takes-down-vine-instagram-\others-with-it/.

[14] Zhiming Shen, Qin Jia, Gur-Eyal Sela, Ben Rainero, Weijia Song, Robbert van Renesse, and Hakim Weatherspoon. Follow the Sun through the Clouds: Application Migration for Geographically Shifting Workloads. In *Proceedings of the Seventh ACM Symposium on Cloud Computing (SoCC'16)*. ACM, 2016.

[15] Sean Michael Kerner. Lloyd's Estimates the Impact of a U.S. Cloud Outage at $19 Billion. URL http://www.eweek.com/cloud/lloyd-s-estimates-the-impact-of-a-u.s.-cloud-outage-at-19-billion.

[16] Adel Nadjaran Toosi, Rodrigo N. Calheiros, and Rajkumar Buyya. Interconnected cloud computing environments: Challenges, taxonomy, and survey. In *ACM Computing Surveys (CSUR)*. ACM, 2014.

[17] Dan Williams, Hani Jamjoom, , and Hakim Weatherspoon. The Xen-Blanket: virtualize once, run everywhere. In *7th ACM European Conference on Computer Systems (EuroSys 12)*. ACM, 2012.

[18] OpenStack Summit. The OpenStack Summit in Hong Kong. URL https://www.openstack.org/summit/openstack-summit-hong-kong-2013.

[19] OpenStack. Two Milestones Mark the Beginning of the OpenStack Interoperability Era, . URL https://www.openstack.org/news/view/59/two-milestones-mark-the-beginning-of-the-openstack-\interoperability-era.

[20] Qin Jia, Zhiming Shen, Weijia Song, Robbert van Renesse, and Hakim Weatherspoon. Supercloud: Opportunities and Challenges. In *ACM SIGOPS Operating Systems Review-Special Issue on Repeatability and Sharing of Experimental Artifacts*. ACM, 2015.

[21] Kaveh Razavi, Ana Ion, Genc Tato, Kyuho Jeong, Renato Figueiredo, Guillaume Pierre, and Thilo Kielmann. Kangaroo: A Tenant-Centric Software-Defined Cloud Infrastructure. In *International Conference on Cloud Computing (IC2E)*. IEEE, 2015.

[22] Alex Fishman, Mike Rapoport, Evgeny Budilovsky, and Izik Eidus. HVX: Virtualizing the Cloud. In *USENIX Workshop on Hot Topics in Cloud Computing*. USENIX, 2013.

[23] Kenneth Nagin, David Hadas, Zvi Dubitzky, Alex Glikson, Irit Loy, Benny Rochwerger, and Liran Schour. Inter-cloud mobility of virtual machines. In *4th Annual International Conference on Systems and Storage (SYSTOR '11)*. ACM, 2011.

[24] Apprenda. Server Virtualization. URL https://apprenda.com/library/glossary/definition-server-virtualization/.

[25] Zhenhao Pan, Qing He, Wei Jiang, Yu Chen, and Yaozu Dong. NetCloud: Towards Practical Nested Virtualization. In *International Conference on Cloud and Service Computing (CSC)*. IEEE, 2011.

[26] Changbin Liu and Yun Mao. Inception: Towards a Nested Cloud Architecture. In *5th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 13)*. USENIX, 2013.

[27] VMware Technology Network. Running Nested VMs. URL https://communities.vmware.com/docs/DOC-8970.

[28] XenServer Open Source Technology. URL http://xenserver.org/.

[29] S. Crago, K.Dunn, P.Eads, L.Hochstein, D.-I.Kang, M.Kang, D.Modium, K.Singh, J.Suh, and J.P.Walters. Heterogeneous cloud computing. In *International Conference on Cluster Computing (CLUSTER)*. IEEE, 2011.

[30] K. Eguro-R. Kaushik D. Kossmann R. Ramamurthy A. Arasu, S. Blanas and R. Venkatesan. Orthogonal security with cipherbase. In *the 6th Biennial Conference on Innovative Data Systems Research (CIDR'13)*. Microsoft Research, 2013.

[31] N.Zeldovich R.A.Popa, C.Red
eld and H. Balakrishnan. Cryptdb: protecting con
dentiality with encrypted query processing. In *the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011.

[32] W. Zhou T. Zhu, G. Li and S. Y. Philip. Differentially private data publishing and analysis: a survey. *Transactions on Knowledge and Data Engineering*, 2017.

[33] S. Foresti S. Jajodia S. Paraboschi V. Ciriani, S. D. C. D. Vimercati and P. Samarati. Combining fragmentation and encryption to protect privacy in data storage. *Transactions on Information and System Security (TISSEC)*, 2010.

[34] S. Jajodia S. Paraboschi S. De Capitani di Vimercati, S. Foresti and P. Samarati. Effecient integrity checks for join queries in the cloud. *Journal of Computer Security*, 2016.

[35] F. D. Rossi T. C. Ferreto T. Lange M. G. Xavier, M. V. Neves and C. A. De Rose. Performance evaluation of container-based virtualization for high performance computing environments. In *the 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. IEEE, 2013.

[36] Ross Brewer and LogRhythm. Advanced persistent threats: minimising the damage. *Elsevier BV*, 2014.

[37] S.Azodolmolky, P.Wieder, and R.Yahyapour. Cloud computing networking: challenges and opportunities for innovations. *Communications Magazine*, 2013.

[38] Downdetector.uk (Amazon). URL http://downdetector.co.uk/problems/amazon.

[39] Eduardo Huedo, Rubn S. Montero, Rafael Moreno, Ignacio M.Llorente, Anna Levin, and Philippe Massonet. Interoperable Federated Cloud Networking. *Internet Computing*, 2017.

[40] Open Data Center Alliance. Open Data Center Alliance (2012) Developing Cloud-Capable Applications. URL http://www.opendatacenteralliance.org/docs/DevCloudCapApp.pdf.

[41] Jonathan Caldwell. Microsoft Azure announces expansion in cloud hybridization and hyperscaling for customers. URL http://www.winbeta.org/news/microsoft-expands-hybrid-cloud-hyper-scale-microsoft-azure-\announcements.

[42] Gartner. Cloud Services Brokerage (CSB). URL https://www.gartner.com/it-glossary/cloud-services-brokerage-csb.

[43] Steve Morad. Amazon Virtual Private Cloud Connectivity Options. URL https://media.amazonwebservices.com/AWS_Amazon_VPC_Connectivity_Options.pdf.

[44] OpenSwan. URL https://www.openswan.org/.

[45] Margaret Rouse. Paravirtualization. URL http://searchservervirtualization.techtarget.com/definition/paravirtualization.

[46] Peter Troger Felix Salfner and Andreas Polze. Downtime Analysis of Virtual Machine Live Migration. In *The Fourth International Conference on Dependability*. International Academy, Research and Industry Association (IARIA), 2011.

[47] Mark Shtern, Bradley Simmons, Michael Smit, and Marin Litoiu. An architecture for Overlaying Private Clouds on Public Providers. In *the 8th International Conference and Workshop on Network and Service Management (CNSM) and System Virtualization Management (SVM)*, 2012.

[48] Naveed Ahmad, Ayesha Kanwal, and Muhammad Awais Shibli. Survey on Secure Live Virtual Machine (VM) Migration in Cloud. In *2nd National Conference on Information Assurance (NCIA)*. IEEE, 2013.

[49] Canonical Ltd. What's LXC? URL https://linuxcontainers.org/lxc/introduction/.

[50] Dan Williams. *TOWARDS SUPERCLOUDS*. PhD thesis, Cornell University, 2013.

[51] Sandor Acs, Miklos Kozlovszky, and Peter Kacsuk. A Novel Cloud Bursting Technique. In *the 9th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, 2014.

[52] Dan Williams, Yaohui Hu, Umesh Deshpande, Piush K. Sinha, Nilton Bila, Kartik Gopalan, and Hani Jamjoom. Enabling Efficient Hypervisor-as-a-Service Clouds with Eemeral Virtualization. In *the 12th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE16)*. ACM, 2016.

[53] Soudeh Ghorbani, Cole Schlesinger, Matthew Monaco, Eric Keller, Matthew Caesar, Jennifer Rexford, and DavidWalker. Transparent, Live Migration of a Software-Defined Network. In *the Symposium on Cloud Computing (SOCC14)*. ACM, 2014.

[54] T.Balan, D.Robu, and F.Sandu. LISP Optimisation of Mobile Data Streaming in Connected Societies. *Mobile Information Systems*, 2016.

[55] Geraphic Network Simulator (GNS3). URL https://www.gns3.com/.

[56] Ibrahim Mansour, Hamid Bouchachia, and Kendra Cooper. Exploring Live Cloud Migration On Amazon EC2. In *5th International Conference on Future Internet of Things and Cloud (FiCloud 2017)*. IEEE, 2017.

[57] Stelios Sotiriadis, Nik Bessis, Euripides G.M. Petrakis, Cristiana Amza, Catalin Negrud, and Mariana Mocanud. Virtual machine cluster mobility in inter-cloud platforms. *Future Generation Computer Systems*, 2017.

[58] Yimeng Zhao, Samantha Lo, Ellen Zegura, Mostafa Ammar, and Niky Riga. Virtual Network Migration on the GENI Wide-Area SDN-Enabled Infrastructure. *Cornell University Library*, 2017.

[59] M.Berman, J.S.Chase, L.Landweber, A.Nakao, M.Ott, D.Raychaudhuri, R.Ricci, and I.Seskar. GENI: A federated testbed for innovative network experiments. *Special issue on Future Internet Testbeds Part I*, 2014.

[60] Aurlien Wailly, Marc Lacoste, and Herv Debar. RetroVisor: Nested Virtualization for Multi IaaS VM Availability. In *the third IT Conference on Conference in Parallelism, Architecture and system (ComPAS 2013)*, 2013.

[61] Dan Williams, Hani Jamjoom, Zhefu Jiang, and Hakim Weatherspoon. VirtualWires for Live Migration Virtual Networks across Clouds. Technical report, Cornell University and IBM T.J Watson Research Center, New York, 2013.

[62] Konstantinos Tsakalozos, Vasilis Verroios, Mema Roussopoulos, and Alex Delis. Time-Constrained Live VM Migration in Share-Nothing IaaS-Clouds. In *the 7th International Conference on Cloud Computing*. IEEE, 2014.

[63] Dan Williams, Hani Jamjoom, and Hakim Weatherspoon. Software defining system devices with the Banana double-split driver model. In *the 6th USENIX conference on Hot Topics in Cloud Computing (HotCloud14)*. USENIX, 2014.

[64] Spandan Bemby, Hongbin Lu, Khashayar Hossein Zadeh, Hadi Bannazadeh, and Alberto Leon-Garcia. ViNO: SDN overlay to allow seamless migration across heterogeneous infrastructure. *Communications Magazine*, 2013.

[65] Cristian Hernandez Benet, Kyoomars Alizadeh Noghani, and Andreas J.Kassler. Minimizing Live VM Migration Downtime Using OpenFlow based Resiliency Mechanisms. In *the 5th International Conference on Cloud Networking*. IEEE, 2016.

[66] Wijaya Ekanayake, Heli Amarasinghe, and Ahmed Karmouch. SDN-based IaaS for Mobile Computing. In *the 14th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2017.

[67] Mininet. URL http://mininet.org/.

[68] Microsoft TechNet. TCP/IP Protocol Architecture. URL https://technet.microsoft.com/en-gb/library/cc958821.aspx.

[69] Reaz Ahmed and Raouf Boutaba. Design considerations for managing wide area software defined networks. *Communications Magazine*, 2014.

[70] Hani Jamjoom Dan Williams and Hakim Weatherspoon. Plug into the Supercloud. *Internet Computing*, 2013.

[71] Ali Jose Mashtizadeh, Min Cai, Gabriel Tarasuk-Levin, Ricardo Koller, Tal Garfinkel, and Sreekanth Setty. XvMotion: Unified Virtual Machine Migration

over Long Distance. In *The Annual Technical Conference (USENIX ATC 14)*. USENIX, 2014.

[72] Mahdi Aiash, Glenford Mapp, and Orhan Gemikonakli. Secure Live Virtual Machines Migration: Issues and Solutions. In *the 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2014.

[73] Raja Wasim Ahmad, Abdullah Gani, Siti Hafizah Ab. Hamid, Muhammad Shiraz, Feng Xia, and Sajjad A.Madani. Virtual machine migration in cloud data centers: a review, taxonomy, and open research issues. *The International Journal of Supercomputing*, 2015.

[74] Tatikayala Sai Gopal, Neha Gupta, Rahul Jain, Srinivasa Reddy Kamatham, and Reddy Lakshmi Eswari P. Experiences in porting TCP application over UDT and their Performance Analysis. In *the International Conference in MOOC Innovation and Technology in Education (MITE)*. IEEE, 2013.

[75] S. Wu B. Ruan, H. Huang and H. Jin. A performance study of containers in cloud environment. In *In Advances in Services Computing: 10th Asia- Pacifi c Services Computing Conference, APSCC 2016*. Springer International Publishing, 2016.

[76] Hanglong Zhan Wei Cui and Bao Li. Cluster as a Service: A Container Based Cluster Sharing Approach with Multi-user Support. In *Symposium on Service-Oriented System Engineering (SOSE)*. IEEE, 2016.

[77] Wubin Li and Ali Kanso. Comparing Containers versus Virtual Machines for Achieving High Availability. In *the IEEE International Conference on Cloud Engineering*. IEEE, 2015.

[78] Ruben S. Montero Rafael Moreno-Vozmediano and Eduardo Huedo. Cross-Site Virtual Network in Cloud and Fog Computing. *Transactions on Cloud Computing*, 2017.

[79] OpenFog. URL https://www.openfogconsortium.org/.

[80] ZHENG YAN RONGXING LU KIM-KWANG RAYMOND CHOO OPEYEMI OSANAIYE, SHUO CHEN and MQHELE DLODLO. From Cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework. *SPECIAL SECTION ON RECENT ADVANCES IN CLOUD RADIO ACCESS NETWORKS*, 2017.

[81] Jungmin Son Amir Vahid Dastjerdi Rajkumar Buyya, Rodrigo N. Calheiros and Young Yoon. Software-defi
ned cloud computing: Architectural elements and open challenges. In *In International Conference on Advances in Computing, Communications and Informatics (ICACCI 2014)*. IEEE, 2014.

[82] Rastin Pries Peter Schneider Admela Jukan Wolfgang Bziuk Steffen Gebert Thomas Zinner Phuoc Tran-Gia Marco Hoffmann, Michael Jarschel. SDN and NFV as Enabler for the Distributed Network Cloud. *Journal of Mobile Networks and Applications*, 2017.

[83] Open Networking Foundation. URL https://www.opennetworking.org/.

[84] M. Ammar V. Eramo, E. Miucci and F. G. Lavacca. An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures. *Transactions on Networking*, 2017.

[85] Subhasree Mandal Joon Ong Leon Poutievski Arjun Singh Subbaiah Venkata Jim Wanderer Junlan Zhou-Min Zhu Jonathan Zolla Urs Holzle Stephen Stuart Sushant Jain, Alok Kumar and Amin Vahdat. B4: Experience with a globally-deployed Software De
fined WAN. *ACM SIGCOMM Computer Communication Review*, 2013.

[86] Rafat Jahan. Inter SDN Controller Communication (SDNi). URL https://events.static.linuxfound.org/sites/events/files/slides/ODL-SDNi_0.pdf.

[87] Mesut Ergin, Yunhong Jiang, Krishna Murthy, James Tsai, Wei Wang, Huawei Xie, and Yang Zhang. KVM as The NFV Hypervisor. URL http://www.linux-kvm.org/images/8/87/02x09-Aspen-Jun_Nakajima-KVM_as_the_NFV_Hypervisor.pdf.

[88] Anupam Tamrakar. Security in Live Migration of Virtual Machine with Automated Load Balancing. In *International Journal of Engineering Research & Technology (IJERT)*, 2014.

[89] Virtual Machine Manager (KVM). Manage virtual machines with virt-manager. URL https://virt-manager.org/.

[90] Ben Pfaff, Justin Petti, Teemu Koponen, Ethan J. Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Jonathan Stringer, Pravin Shelar, Keith Amidon, and Martin Casado. The Design and Implementation of Open vSwitch. In *the 12th Symposium on Networked Systems Design and Implementation (NSDI15)*. USENIX, 2015.

[91] UDT UDP based Data Transfer. UDT: Breaking the Data Transfer Bottleneck. URL http://udt.sourceforge.net/.

[92] Prasenjit Sarkar. Improving Performance with Path Optimization and UDT - VMware vCC 2.5. URL http://stretch-cloud.info/2013/06/improving-performance-with-path-optimization-and-udt-vmware-\vcc-2-5/.

[93] Gabriele Lospoto, Massimo Rimondini, Benedetto Gabriele Vignoli, and Giuseppe Di Battista. Rethinking virtual private networks in the software-defined era. In *International Symposium on Integrated Network Management (IM)*. IEEE, 2015.

[94] G. Carrozzo, R. Monno, B. Belter, R. Krzywania, K. Pentikousis, M.Broadben, T. Kudoh, A. Takefusa, A. Vico-Oton, C. Fernandez, and B.Puype J. Tanak. Large-scale sdn experiments in federated environments. In *International Conference on Smart Cmmunications in Network Technologies (SaCoNeT)*. IEEE, 2014.

[95] OpenDaylight. Whats New in Lithium. URL https://www.opendaylight.org/whats-new-lithium.

[96] OpenStack. OpenStack Orchestration Heat, . URL https://wiki.openstack.org/wiki/Heat.

[97] iPerf3. URL https://iperf.fr/iperf-download.php.

[98] Wireshark. URL https://www.wireshark.org/.

[99] hdparm. URL https://wiki.archlinux.org/index.php/hdparm.

[100] Ding-Yong Hong, Jan-Jan Wu, Pen-Chung Yew, Wei-Chung Hsu, Chun-Chen Hsu, Pangfeng Liu, Chien-Min Wang, and Yeh-Ching Chung. Efficient and Retargetable Dynamic Binary Translation on Multicores. In *Transactions on Parallel and Distributed Systems*. IEEE, 2014.

[101] Tom Fifield, Diane Fleming, Anne Gentle, Lorin Hochstein, Jonathan Proulx, Everett Toews, and Joe Topjian. OpenStack:Issues with Live Migration. URL http://www.tomsitpro.com/articles/openstack-operations-guide-compute-nodes,2-746-3.html.

[102] RIGHSCALE. Cloud computing trends: 2016 state of the cloud survey. URL http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey.

[103] Amazon Web Services. Elastic IP Addresses, . URL http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ipaddresses-eip.

[104] Steve Russell. DynamicDNS. URL https://help.ubuntu.com/community/DynamicDNS.

[105] Werner Fischer. Overview of the Intel VT Virtualization Features. URL https://www.thomas-krenn.com/en/wiki/Overview_of_the_Intel_VT_Virtualization_Features.

[106] Ubuntu Manuals. lscpu command. URL http://manpages.ubuntu.com/manpages/trusty/man1/lscpu.1.html.

[107] Ubuntu Documentation. Network Connection Bridge. URL https://help.ubuntu.com/community/NetworkConnectionBridge.

[108] Amazon Web Services. Compute Optimized Instances, . URL https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/compute-optimized-instances.html.

[109] noip. URL https://www.noip.com/.

[110] The Linux Kernel Archives. URL https://www.kernel.org/.

[111] Creating custom kernels with Debians kernel-package system. URL http://newbiedoc.sourceforge.net/tutorials/kernel-pkg/config-kernelpkg.html.en.

[112] Assured Information Security Inc. BitVisor. URL https://github.com/Bareflank/hypervisor.

[113] KVM. Enable VT-X on Mac Pro (Early 2008). URL https://www.linux-kvm.org/page/Enable_VT-X_on_Mac_Pro_(Early_2008).

[114] Code Project For those who code. Virtualization for System Programmers. URL https://www.codeproject.com/Articles/215458/Virtualization-for-System-Programmers.

[115] Packet. URL https://www.packet.net/.

[116] Libvirt Virtualization API. KVM/QEMU. URL https://wiki.libvirt.org/page/Main_Page#KVM_.2F_QEMU.

[117] TightVNC Software. URL http://www.tightvnc.com/.

[118] Cisco. VPNs and VPN Technologies. URL http://www.ciscopress.com/articles/article.asp?p=24833&seqNum=3.

[119] Anita Choudhary, Mahesh Chandra Govil, Girdhari Singh, Lalit K.Awasthi, and Emmanuel S.Pilliand Divya Kapil. A critical survey of live virtual machine migration techniques. *Cloud Computing: Advances, Systems and Applications*, 2017.

[120] OpenVPN. Openvpn ethernet bridging. URL https://openvpn.net/index.php/open-source/documentation/miscellaneous/76-ethernet-bridging.html.

[121] OPENDAYLIGHT, . URL https://www.opendaylight.org/.

[122] Alberto Rodriguez-Natal, Jordi Paillisse, and Florin Coras. Programmable Overlays via Open-OverlayRouter. In *Communications Magazine*. IEEE, 2017.

[123] Zodiac-FX OpenFlow Switch. URL https://northboundnetworks.com/products/zodiac-fx.

[124] Alberto Rodriguez-Natal, Lorand Jakab, and Vina Ermagan. Location and identity privacy for LISP-MN. In *International Conference on Communications (ICC)*. IEEE, 2015.

[125] OPENDAYLIGHT. LISP Flow Mapping User Guide, . URL http://docs.opendaylight.org/en/stable-nitrogen/user-guide/lisp-flow-mapping-user-guide.html.

[126] Flux7 Labs Technology Adoption Blog. An Introduction to Openstack Project. URL http://blog.flux7.com/blogs/openstack/openstack-tutorials-part-1-openstack-introduction.

[127] Amazon AWS. Amazon EC2 Instance Types, . URL https://aws.amazon.com/ec2/instance-types/.

[128] Amazon AWS. Now Available New C4 Instances, . URL https://aws.amazon.com/blogs/aws/now-available-new-c4-instances/.