A Review : Live Virtual Machine Migration Techniques

Literature Review
SCS 3216 : Research Methods

B.F. Ilma
Student of University of Colombo School of Computing
Email: 2019cs061@stu.ucsc.cmb.ac.lk
Supervisor: Dr. Dinuni K Fernando

January 12, 2024

Declaration

The literature review is my original work and has not been submitted previously for any examination/evaluation at this or any other university/institute. To the best of my knowledge, it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name : B.F.Ilma
Registration Number : 2019/CS/061
Index Number : 19000618

Signature & Date

This is to certify that this literature review is based on the work of Ms. B.F.Ilma under my supervision. The literature review has been prepared according to the format stipulated and is of acceptable standard.

Supervisor Name : Dr. D.K.Fernando

Signature & Date

Acknowledgement

I thank my supervisor Dr. Dinuni K. Fernando, for providing me with the necessary domain knowledge needed to conduct this survey and for guiding me throughout this process. I also thank Dr. Manjusri Wickramasinghe for his guidance on research methods and the literature review process.

Additionally, I would like to thank my friends and family, without whom this would not be possible.

Abstract

Cloud Computing (CC) has become a prominent technology that exists today. Storage facilities, software, identity and access management are one of the many things that could be provided as cloud services. Cloud service providers handle the large number of requests that arise with the popularity of CC through the use of Virtualization. Virtualization has become a vital technology in CC. It provides numerous benefits to cloud service providers including managing hardware and software resources in Cloud Data Centers (CDCs).

The major advantage of virtualization could be the migration of Virtual Machines (VMs). As VMs are independent of underlying hardware, they can be migrated among servers for various reasons such as fault tolerance, load balancing, availability and security. For interactive time-critical services, it is crucial that this migration of a VM happens in a transparent manner to the end user. Hence, majority of research focuses on *Live Migration*, which is migrating a VM while it continues to provide services. Today, there exists numerous variants of migration mechanisms in different settings. VMs with different workloads would perform better in different migration schemes. As such, there are many optimization mechanisms that researchers have come up with which can be evaluated using different performance metrics.

In this paper, state-of-the-art VM live migration methodologies are discussed. Initially, a background study of cloud computing, virtualization and migration mechanisms, in general, is given. In the following sections, live migration methodologies, examples of such in different settings and optimization techniques to improve the efficiency of the migration schemes are explained. The critical section analyses several migration schemes under different categories. The paper concludes with the future directions upon which more research could be conducted.

Key Words: Pre-copy Migration, Cloud Data Center, Virtualization, Migration, Live Migration, Post-copy Migration, Virtual Machines

Contents

List of Figures

Acronyms

CC Cloud Computing. 3, 6

CDC Cloud Data Center. 7

CDCs Cloud Data Centers. 3, 6–8, 11, 21

COW Copy-On-Write. 21

CPU Central Processing Unit. 14

DCLM Delta Compression Live Migration. 18

DoS Denial-of-Service. 8

DSB Dynamic Self Ballooning. 24

IP Internet Protocol. 21

IRLM Inter-rack Live Migration. 20

ISR Internet Suspend/Resume. 9

KVM Kernal-based Virtual Machine. 12, 17, 30

LAN Local Area Network. 11, 17, 18, 21, 23, 28, 29

MITM Man-In-The-Middle. 8

NAS Network-Attached Storage. 11, 28

OS Operating System. 12, 23

QoS Quality of Service. 7, 10–12, 24, 27, 28

SLA Service Level Agreement. 7–11

VM Virtual Machine. 3, 7–15, 17–21, 23–29, 31, 32

VMM Virtual Machine Monitor. 6, 10, 12, 15, 21, 24

VMMs Virtual Machine Monitors. 6, 12, 23, 24, 30

VMs Virtual Machines. 3, 6–8, 10–12, 21, 26, 29, 31

WAN Wide Area Network. 11, 17, 21, 23, 28, 29, 31

WWS Writable Working Set. 13, 18

1 Introduction

1.1 Cloud Computing

Throughout the previous years, CC has become a prominent technology. As technology has advanced, many applications require high storage and processing power. This increasing demand for resources has motivated researchers to share the available resources among the end users in an efficient manner so that it could maximize the service provider's profits. CC is such a technology. Its ability to provision hardware resources efficiently through the cloud and provide software services reliably has improved customer satisfaction immensely in many industries. In CC, services are provided in many forms such as hardware, software, infrastructure, storage, security etc. Some examples of cloud-based service platforms can be named as Microsoft Azure (Microsoft 2023*a*), Amazon Web Services (AWS 2023), Alibaba Cloud (Alibaba Cloud 2023), Google Cloud Platform (GCP 2023) etc. Due to the efficiency of cloud-based services, during the past few years there came to be an increasing demand for cloud resources. These resources are provided to the users via Cloud Data Centers (CDCs). Cloud Data Centers (CDCs) are nowadays equipped with more than thousands of server computers to allocate resources for the end users. The server computers are connected with high-speed communication links to provide services in different forms. Infrastructure, Platform and Software are the basic three models of cloud services. CDCs provide such services for the users over the internet based on a pay-per-use model for a period of time. As such, it is important to provide these services in a seamless manner to the users.

CDCs usually consume a large amount of electricity to manage the server computers. However, research shows that these servers consume a vast amount of energy without doing any useful work. In CDCs typically about 30% of the cloud servers were found to remain idle only consuming energy (Uddin et al. 2013). As such, a vast amount of energy is spent in CDCs for maintaining server computers. One solution for this scenario is to shut down the idle servers based on resource demands.

1.2 Virtualization

Virtualization is a technology adapted to solve the above problems. Generally, virtualization can be found in several areas of computer science, whether its hardware, software or firmware. In this study, virtualization of a complete computer environment with its processing capabilities, memory and communication channels is considered. To achieve this, there is a software layer known as a Hypervisor which runs on the Virtual Machines (VMs). This is otherwise known as a Virtual Machine Monitor (VMM). Its main task is allotting the available resources among the co-hosted VMs running on the host machine.

Multiple VMs can be hosted on the same hardware. This provides better resource utilization to CDCs by packing several VMs in under-utlized servers. VMMs make sure to map incoming requests from the guest operating systems to hardware resources. VMMs also provide performance isolation by making sure

co-hosted VMs are unaffected when one of the VMs running on the same machine fails (Medina & García 2014). Popular virtualization software include SolarWinds Virtualization Manager (SolarWinds Worldwide, LLC 2023), V2 Cloud (V2 Cloud Solutions, Inc 2023), Oracle VM Virtualbox (Oracle 2023), VM Ware Workstation (VMware, Inc 2023), QEMU (QEMU 2023), Microsoft Hyper-V (Microsoft 2023*b*), Red Hat Virtualization (Red Hat, Inc 2023) etc.

1.3 Why Live Migration needed in Cloud Computing

Hosting multiple VMs in one single server could lead to over-utilization of physical resources and therefore performance degradation in CDCs. Hosting different types of VMs in one machine is also a daunting task for researchers as interactive cloud applications experience frequently changing workload requests and have dynamic resource demands. As a result, some servers in the CDCs may be over-utilized while some remain idle. Additionally, all the VMs deployed within the server would also be impacted in case of a hardware failure, .
To solve the above problems migration technology was introduced. Migration technology solves the over-utilization and performance degradation issues by migrating VMs between physical servers. The hypervisor could relieve an over-utilized server by migrating a VM from it to an under-utilized or idle server. During VM migration, VMs continuously demand resources for its running applications. Additionally, VMs also require additional resources for migrating a VM. Hence, it is important that the migration of a VM takes least possible time so that in can dismiss the resources back to the server.

This mechanism originated from Process Migration (Osman et al. 2002). However, there are residual dependencies in process migration which prevents it from using it on cloud environment. Processes must be abstracted to run in isolation so that they could be migrated without having any ties left to the source server. Also in such a case, processes would have to be restricted in communicating with other processes to avoid creating host dependencies. In contrast, VMs make a migrating unit which is not dependent on the source server. VMs could be migrated from one server to another within the same CDC or from one CDC to another as well.

Doing the above process while continuously providing services to the end users is called ***Live Migration***. Live migration provides services to end users in a seamless manner without breaking the connection. This avoids violation of Service Level Agreement (SLA) and provides most efficient resource utilization. In contrast to live migration, non-live migration suspends the VM at the time of migration and only allows it to resume once the migration is complete. Through its definition, it can be easily understood that non-live VM migration cannot be used for time-critical applications which demand zero downtime. Using Live migration, CDCs can uphold their Quality of Service (QoS).

Live migration provides many benefits and use cases such as adaptive application resource remapping (Atif & Strazdins 2014), traffic engineering (Jiang et al. 2012), server consolidation (Padala et al. 2007), hardware maintenance with near-zero downtime (Boss et al. 2007), load balancing (migrating VMs from heavy-loaded servers to idle servers to improve application throughput), fault tolerance

(When a fault occurs in the host, the VM may be moved to a different host while it operates. Meanwhile the original host can be repaired and the VM can be migrated back again), providing seamless services to end users during periodic system maintenance, managing power by switching off idle servers after migrating workload from under-utilized servers to servers with available resources etc.

VM migration mechanism also brings performance overhead to the source host, destination host, the migrated VM and the other cohabited VMs within the two hosts (Voorsluys et al. 2009). Although this overhead happens to be at acceptable levels, it cannot be discarded. As such, there have been many attempts at optimizing the migration process. These will be discussed in detail in section 4. Additionally, live VM migration could also suffer from security threats like Denial-of-Service (DoS), Overflow Attack, Replay Attack and Man-In-The-Middle (MITM) (Princess et al. 2018). Hence it is important to take into consideration the security threats that are associated when adapting VM migration into CDCs.

1.4 Performance Metrics

In this section several attributes which are used to evaluate the efficiency of a migration methodology are discussed. An ideal migration scheme not only migrates a VM but also minimizes its side effects on the VM and other co-hosted VMs.

- Downtime

  Downtime represents the time period in which services cannot be provided due to the VM's migration. This attribute measures the migration scheme's transparency. For example, *Live migration* and *Pre-copy* migration are mechanisms that are adapted to minimize this aspect of a migration scheme.

- Migration duration

  Period of time in between the beginning and completion of a migration scheme is defined as the migration duration. Often times, the greater the migration duration, the larger the risk of affecting the VM in a negative way.

- Service degradation

  This attribute shows the impact of migrating a VM on the services executing on it. Service degradation can be evaluated by attributes such as response time and throughput. Often, severe service degradation can lead to SLA violations.

- Network traffic

  The total data transported through the migration scheme is given by the network traffic. This attribute becomes critical in cases where the VM runs a network-intensive task within it and a bandwidth contention occurs between the application and the migration process.

- Network bandwidth utilization

Network traffic and migration duration can be combined to measure the network bandwidth utilization (Zhang et al. 2018). Bandwidth utilization will be higher when the total network traffic is transferred with minimal migration duration.

The attributes discussed above can be used to assess and analyze various state-of-the-art migration strategies. A majority of the current VM migration mechanisms focus on optimizing one of these aspects or several at once.

2 Migration Techniques

In 2015, Ahmad et al. categorized the existing VM migration techniques considering seven characteristics associated with the process which are liveliness, objective function, migration granularity, network link, and hypervisor type. In this paper, currently existing migration mechanisms are considered under the above characteristics.

2.1 Liveliness

All VM migration schemes can be categorized mainly under the categories ***Hot migration (Live migration)*** and ***Cold migration (Non-live migration)***.

2.1.1 Non-live migration

As the name suggests, in this method, the VM is shut down before the migration process starts. Then, the migration takes place and the VM needs to be completely transferred to a different server before it can be resumed again. Figure 1 shows the state transition of non-live VM migration. This process clearly, has a lot of issues when it comes to time-critical interactive cloud services as it largely affects the SLA. One advantage of non-live migration mechanisms is that they enable migration of the whole memory at once and provide a predictable migration time .

The foundation for non-live VM migration is Process migration (Milojičić et al. 2000). A prime example for this is the Process Domain (Pod) abstraction method (Osman et al. 2002) where groups of processes are abstracted using a virtual layer that removes dependencies to the host system. These pods can be easily migrated to another machine while also preserving their network connections.

Over the years there have been a number of non-live VM migration mechanisms come forth. Kozuch et al. came up with Internet Suspend/Resume (ISR) mechanism where they copy the state of a VM resided in the local file system after suspension and resume it in another site. Initially this was tested out with NFS under LAN conditions and then improved it with Coda distributed file system to transfer the files over the internet.

Performance metrics of non-live VM migration can be taken as below.

- Resume Latency - Time it takes before being able to do any useful work after transferring to a new site.

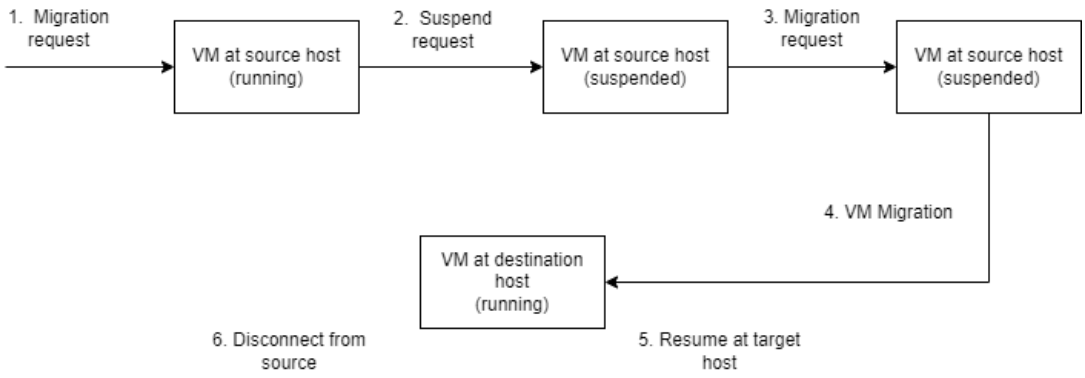

Figure 1: States of Non-live VM Migration

- Slowdown - Performance slowdown after resuming at a new site.

Currently, non-live migration trends are becoming obsoleted as a result of adapting to uninterruptible service delivery to customers.

Live VM migration mechanisms will be discussed in detail in section 3.

2.2 Objective Function

When considering the objective function of the VM migration schemes, **minimizing downtime** can be taken as a major objective of interactive applications. Downtime is defined as the period of time in which service will be unavailable to the users. Downtime of a cloud service can be affected by several factors including the capacity of network bandwidth, portion of system resources consumed by the VM migration process, dirty page(s) rate, page(s) size etc. In turn, it also affects QoS and SLA. **Pre-copy migration** (Clark et al. (2005);) which is a kind of live-migration, is one of the mechanisms which try to achieve this objective by migrating the memory pages before migrating the VM execution state. Several migration schemes adapts optimization mechanisms to reduce service downtime such as asynchronous data transfer of memory pages where pages with similar content are prevented from being migrated over and over and also compression of data before they are transferred. Additionally, compared to live migration schemes, non-live migration schemes have a greater downtime since it halts providing service during migration.

Minimizing the migration duration is another objective of some migration mechanisms. This defines the time taken for the VMM to complete migrating a VM. The duration depends on the quantity of memory that is moved as well as the frequency of dirty pages. When the VMM starts to migrate a VM, it allocates resources such as I/O devices, memory, CPU cycles and network bandwidth. Therefore, co-hosted VMs are greatly affected by the migration process. As such,

the longer the duration, the greater the degradation in the performance of the source host.

Additionally, when compared to live-migration schemes, non-live migration schemes have a higher migration duration as a whole VM is migrated at once (Osman et al.; Kozuch et al.; Kozuch & Satyanarayanan).

When considering QoS, **Application Performance** is another objective of migration schemes. Performance can be measured by the application's query response time, throughput, rate of dropped queries, jitter and service degradation time. The performance degradation duration is affected by the downtime and the duration of migration (Riteau et al.; Deshpande et al.; Zhu et al.; Deshpande et al.).

Maximizing bandwidth utilization specifies whether the network bandwidth has been efficiently used by the VM migration mechanism. As this is a shared resource among the co-located VMs, inefficient use of network bandwidth often produces performance issues in the other VMs (Ibrahim et al. 2011). Some of the VM migration schemes adopt **deduplication** (Al-Kiswany et al. (2011); Deshpande et al. (2012); Deshpande et al. (2011)), **compressing memory** (Hu et al. (2013)); Kozuch & Satyanarayanan (2002); Svärd, Hudzia, Tordsson & Elmroth (2011); Jin et al. (2009)), **filtering free memory pages** (Koto et al. 2012) and **dynamic rate limiting** (Svärd, Hudzia, Tordsson & Elmroth (2011); Deshpande et al. (2012); Jin et al. (2009); Deshpande et al. (2011); Gerofi et al. (2011)) to efficiently leverage the network bandwidth during the VMs migration. As such, all performance metrics like QoS, VM migration time and application downtime depends on the utilization of network bandwidth.

2.3 Network Link

VM migration schemes have adapted either Wide Area Network (WAN) based migrations or Local Area Network (LAN) based migrations. However, majority of existing migration mechanisms have considered LAN networks. This is due to the benefit of not considering storage migration which is needed for WAN but not for LAN as Network-Attached Storage (NAS) is easily accessible for both source host and target host in LAN migration schemes. Migration across WAN links is needed to make CDCs more scalable. Additionally, there's issues that should be considered with respect to WAN migration schemes such as limited network bandwidth, high packets-dropped rate, and higher latency (Bradford et al. (2007); Wood et al. (2011); Gerofi et al. (2011)).

WAN migration mechanisms also greatly affect the application QoS, downtime and migration time (Wood et al. 2011). As such, in a WAN setting, one would also need to face challenges such as minimizing downtime with respect to the migration of the disk state additional to the memory state, minimizing network configurations as different CDCs would support different IP address ranges, minimizing the latency and cost of migration and operating efficiently over low-bandwidth links.

For the above stated reasons, the possibility of SLA violation is much higher in migraiton schemes within WAN setting.

2.4 Hypervisor Type

VM migration schemes can also be classified under the type of hypervisor tool used for the migration. Most of the existing VMMs support Live migration of VMs. The first virtualization solution for the x86 architecutre, **VMWare Workstation** also supports transparent live migration (Bugnion et al. 2012). Barham et al. (2003) introduces **Xen** as a high-performance VMM which uses *paravirtualization* and contrasts it with other traditional VMMs such as **VMWare's ESX Server**. In case of Xen, one would require to have modifications to the guest OS to support virtualization.

Kivity et al. (2007) proposes a VMM called **Kernal-based Virtual Machine (KVM)** which provices VMM capabilities to Linux. KVM provides its own *dirty page log facility* for live migration where pages modified after copying can easily be tracked. KVM has configured as its termination criteria for the pre-copying phase of live migration as when two passes (not necessarily consecutive) were made where there's an increase in the amount of memory content transferred or when more than thirty iterations have passed.

A majority of the existing migration schemes have considered Xen and KVM.

2.5 Migration Granularity

A VMM can either migrate a single VM or multiple VMs. Migrating multiple VMs greatly affects the application QoS, downtime and other co-hosted VMs (Deshpande et al. 2012). Additionally, there's also a queuing delay at the source end when considering multiple VMs at once which contributes to accumulated total migration time being greater.

3 Live Migration Techniques

As mentioned in this paper before, *live migration* refers to migrating a VM while it continues to provide its services to end users. Live migration schemes are adapted to efficiently utilize the network bandwidth during the migration process (Svärd, Hudzia, Tordsson & Elmroth 2011), decreasing the performance degradation during migration (Ibrahim et al. 2011) and minimizing the service downtime (Zhu et al. 2013).

Live migration mechanisms can be categorized under three types based on the method adapted to transfer memory content. They are, *Post-copy migration*, *Pre-copy migration* and *Hybrid migration*. As the primary goal of pre-copy migration is minimizing service downtime, memory is iteratively migrated. In contrast, in post-copy migration methods, a minimum system state is transferred before the memory content to optimize the migration duration. Hybrid migration mechanisms adapts attributes of both post-copy and pre-copy migration.

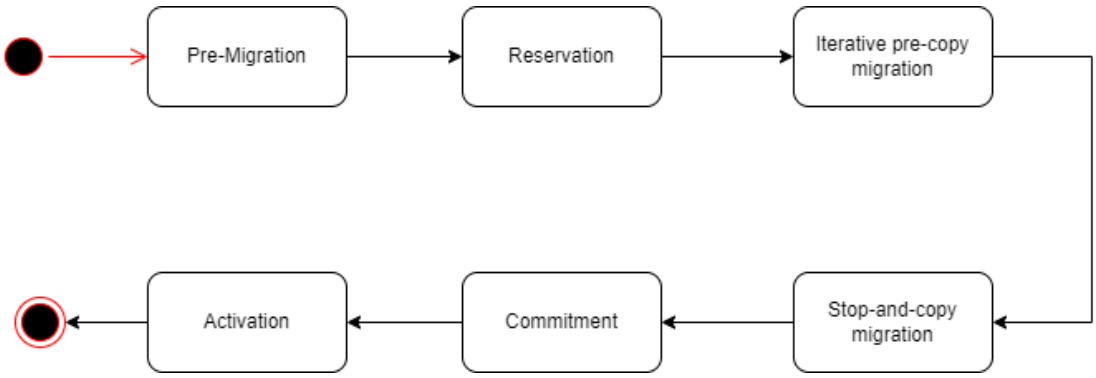

Figure 2: States of Pre-copy VM Migration

3.1 Pre-copy migration

When a VM provides live services to end users, it is crucial to have the migration happen transparently. The requirements of pre-copy migration are basically *minimizing the service downtime* and *migration duration*.

The memory transfer of a VM can basically be broken down into its major three phases.

- Push phase.
- Stop-and-copy phase
- Pull phase

Something to be noted is that, most of the existing pre-copy migration schemes only adopt one or two of the above phases in their practical implementations. For example, methods adapted by Constantine et al. (2002) and Kozuch et al. (2002) only adapt the latter two phases. However, this could have severe consequences if the application provides live services.

Generally in pre-copy migration's *push phase*, pages are migrated in many rounds. This occurs when the VM remains active at the source while the target receives its memory content. The reason it takes several rounds is that since the VM continues to provide services, the pages continue to be modified. Hence modified pages will have to be resent.

The convergence of the push phase is indeed an issue with the pre-copy migration as it is unclear on when exactly the transfer of pages must be stopped. In case of pages never being modified, copying the memory pages a single time will suffice. Alternatively, if the VM continues to modify memory, then Writable Working Set (WWS) should be considered. WWS refers to the set of pages which are modified by the VM frequently. In this case, they must be moved in the stop-and-copy phase as they are the least optimal choice for the pre-copy phase.

In general, different migration schemes have defined their own criteria of converging this iterative process such as, for example, setting a limit to the number of rounds executed.

When the push phase is completed, the stop-and-copy phase begins. Here, the VM suspends and its execution state is migrated.

At last during the pull phase, the migrated VM resumes providing services at the target host. Any page fault (memory which is not yet transferred) is pulled from the source host.

Clark et al. (2005) presents in their study, the effectiveness of pre-copy migration for different workloads. The experiments revealed that when considering only one iteration of pre-copy, it always performs better than just a stop-and-copy migration. When considering two, three or four iterations, a reduction in downtime could be observed albeit diminishing with higher bandwidth rates. The study suggests that increased bandwidth at the latter iterations would optimize the downtime as then pages can be transferred faster than the page dirtying rate. **Dynamic rate limiting** is a technique used to obtain the optimal network bandwidth for migrating a VM. The migration starts with a minimum bandwidth initially set by the system administrator. With each iteration, the dirtying rate is calculated while also incrementing the network bandwidth to accommodate it. The push phase is terminated once the bandwidth gets higher than the maximum bandwidth set by the administrator.

Figure 2 depicts the design overview of pre-copy migration. During the Pre-Migration stage the VM remains active within the source. At the Reservation stage, the request for the migration of the VM is issued. During the third stage, the memory content is transferred in an iterative manner until it converges. During the next stage, VM's execution is suspended. The memory pages that are leftover, are subsequently migrated with it across the network. During the Commitment stage, the target host acknowledges that it has received the VM. All remaining ties with the source host is now terminated. During the Activation stage, the VM resumes service at the target host.

3.2 Post-copy migration

When considering read-intensive VM workload, the pre-copy migration seems to minimize the service downtime and performance degradation well. However, this is not the case for write-intensive workload. This is because in write-intensive workload, the memory content is frequently modified which results in them needing to be migrated multiple times. In this case, usage of post-copy migration schemes would prefer to be more advantageous.

Contrasting to pre-copy, post-copy transfers the pages only after the VM Central Processing Unit (CPU) state has been migrated. However, in this case it should be noted that pre-copy migration schemes provide assurance of aborting the migration process should any fault arise within the target since the whole of VM still remains at the original host. In post-copy however, there's no clean way. After resuming the VM, the memory pages are migrated. During this time, any page faults occurring (since the VM is resumes providing services on the target server) is *demand-paged* from the source. Hence, it should be noted that unlike in pre-copy, each memory page would only be moved a maximum of once in post-copy migration.

Different methods can be used to bring the pages from the source. This causes variations in post-copy migration. The most inefficient and simplest method to fetch pages would be **Demand Paging**. Page faults will be resulted when the VM resumes its execution after the migration since all of memory is yet to be transferred to the destination. These pages must be migrated through the network link. However, this method will significantly cause a performance degradation in the application (Hines et al. 2009). Thus, this variant is deemed as primitive and insufficient for the goal of optimizing the application degradation and migration duration.

The question arises on how to detect the page faults occurring at the target host. In this case, Hines et al. (2009) discusses three different ways.

- Shadow Paging.

  Shadow page tables are maintained by the VMM for each VM. All pages are marked as read-only. The VMM can track the updates made to each page and propagate the changes to the shadow page table. This method can be used to track access to pages that do not exist yet at the target host (Liu, Xu, Jin, Gong & Liao 2011).

- Page Tracking

  During downtime, the page tracking technique marks all the pages that are within the VM's memory in the target server as *not present* in the corresponding entries within the page table. This causes a page fault exception. The purpose of this is to identify which pages are being used by the VM and which pages are not. By forcing the VM to fault on the pages, the system can identify which pages can be swapped out to free up memory resources.

  Once a page fault happens, a third party service can step in to resolve the issue and fix the page table entry.

- Pseudo Paging

  VM's pageable memory is first transferred before the actual VM instance to an in-memory-pseudo-paging tool that inhabits within the target host. When the VM is successfully moved and resumes service at the target, it can retrieve the pages that are swapped out through the default page fault mechanism. This reduces the amount of memory that is required to be migrated over the network.

Figure 3 depicts the flow of post-copy migration. As illustrated, the VM fetches pages that are not already fetched from source to target. The target server remains connected to the source server until all of the memory content is transferred. The connection is severed when the source and target are completely synced.

3.3 Hybrid migration

Features of post-copy and pre-copy methods are combined in hybrid VM migration schemes.This is so that it achieves optimal performance of VM migration. Hybrid

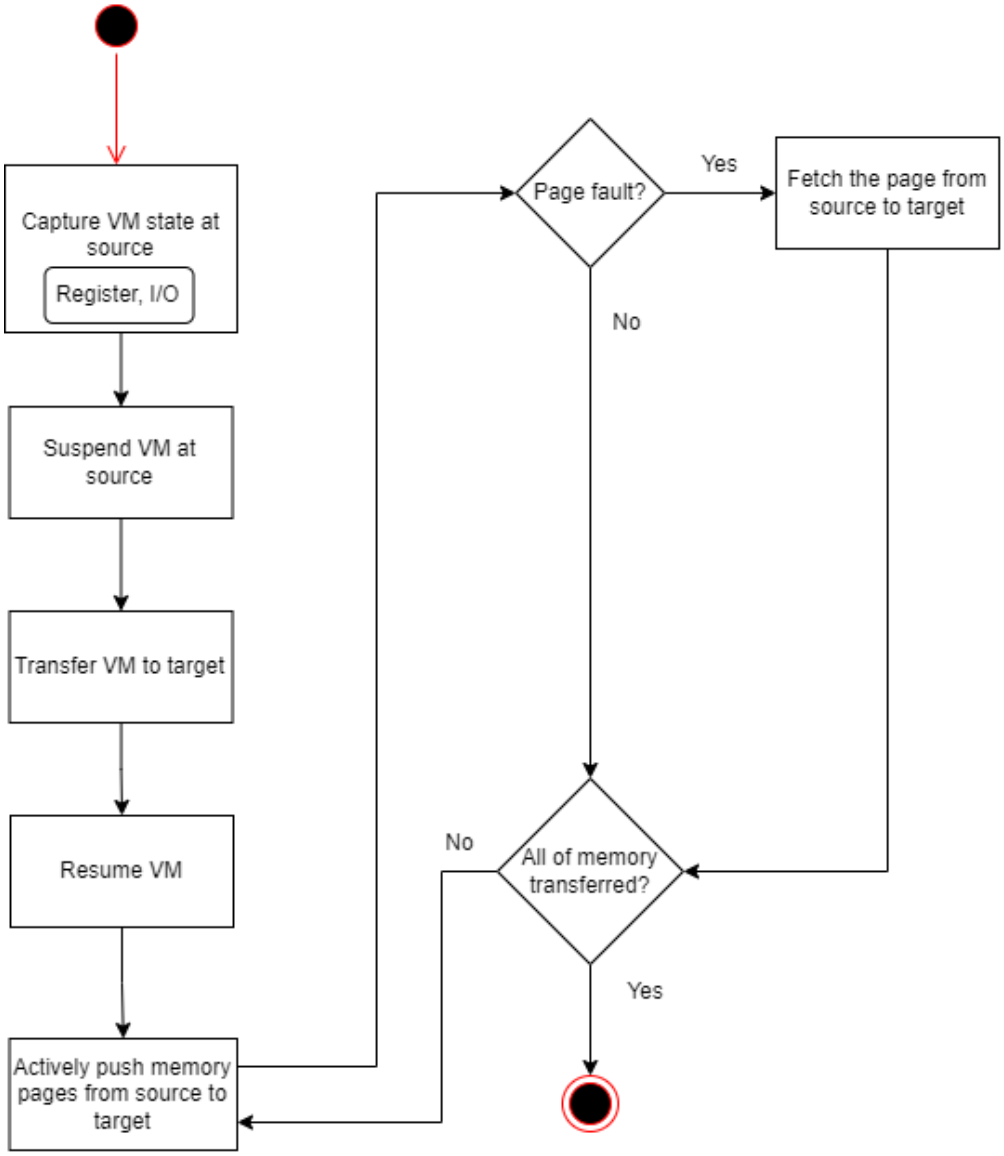

Figure 3: Post-copy VM Migration

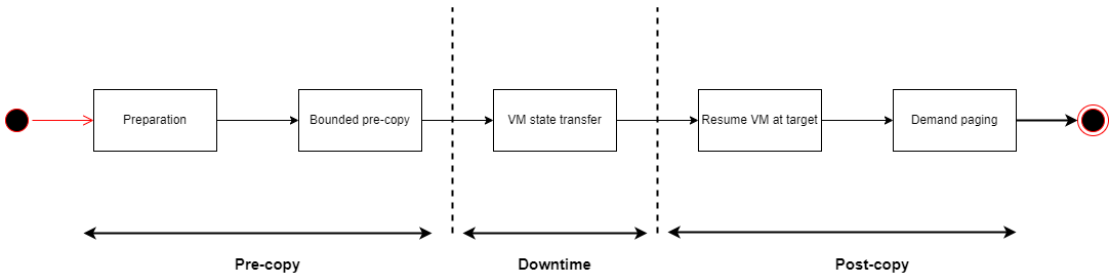

Figure 4: Hybrid VM Migration

mechanisms adopt pre-copy rounds which are bounded, to transfer a set of pages so as to lessen the frequency of page faults. After this, according to post-copy, a minimal state of the VM is transferred which then resumes service on target.

This mechanism can be broken down into five main steps. As Figure 4 illustrates, system resources that are needed for the VM are allocated within the target at the Preparation stage. Next, the bounded pre-copy iterations migrate a set of memory content from source to receiver. VM's minimum state is then captured and migrated to the target where it resumes. Finally, according to the application's read/write requests, remaining pages are fetched through page faults and demand paging. After both the source and target are synchronized, the connection between is cut off.

3.4 Live migration variants

In this section, several existing live migration mechanisms will be discussed in detail under the two categories LAN and WAN.

3.4.1 LAN Setting

Ibrahim et al. (2011) presents a live migration scheme which is optimized for memory intensive (scientific) applications. The proposed scheme uses the pre-copy variant with KVM. The study comes up with a novel condition for the termination of the iterative pre-copy rounds and the start of transferring the VM's execution state. This switching (between two stages) is triggered by analyzing the memory update patterns that would indicate no furthur progress would be made by allowing the iterative rounds to continue. The analysis can result in three cases.

- No. of modified pages is not minimized by continuing the iterative rounds.

  This indicates that the transfer rate of pages is lower than the page dirtying rate. Allowing further iterations would not prove to be useful in this case.

- Application's page modification rate drops.

  In this case, a small downtime could be attained.

- Most transmitted pages are identical.

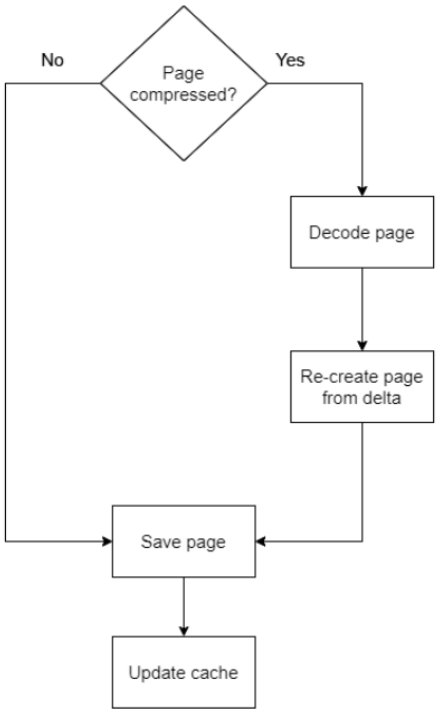

Figure 5: XBRLE Source Side

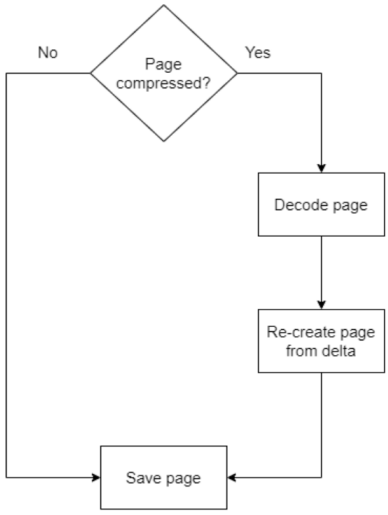

Figure 6: XBRLE Target Side

> In this case, there's a high retransmisson of pages. If the percentage of retransmission exceeds 90% of the transmitted pages, the migration can move on to the next stage.

Clark et al. (2005) also presents a similar mechanism (pre-copy) over LAN using Xen's shadow page tables. By extension of a previous study, Clark et al. presents the solution of not transferring WWS (frequently modified pages) to further optimize the iterative pre-copy rounds. Koto et al. (2012) also proposes a memory optimization mechanism with **SonicMigration** scheme that strives to shorten the total migration duration by reducing the memory content to be transferred (*soft pages*).

Additionally, transferring large VMs across the network would also be a challenge. Svärd, Hudzia, Tordsson & Elmroth (2011)'s Delta Compression Live Migration (DCLM) provides a new compression algorithm (**XBRLE**) to minimize the amount of data migrated and therefore increase the throughput. When a page needs to be transferred, the source server searches for cached previous versions of the memory page. If none is found, the page is sent and cached. If cache exists, a delta (XOR) version is created between the page and the previous version and cached, compressed and sent. From the target side, it checks for the *delta flag*. If the flag is set, the target decodes the page and recreates the page from delta. Figure 5 and 6 illustrates the workflow.

Hines et al. (2009)'s post-copy mechanism contrasts the above schemes by differentiating the way memory content is transferred over the network (after the transfer of the VM state). The proposed scheme is much more beneficial (than pre-

copy migration) for write intensive applications as pages are frequently modified. The goal of this method is to minimize the migration duration. The study however doesn't present a solution for failure of the target host during migration. The study suggests that implementing checkpoints at the target host where the VM's execution state can be sent (as a backup) to the source host would be a solution in case the target host fails. However, this would also induce additional reverse network traffic overhead.

Fernando et al. (2019) in their study illustrates this solution of transmiting **reverse incremental checkpoints** back to the source with *PostCopyFT*. PostCopyFT would support either periodic checkpoints or event-based checkpoints. This occurs parallel to the ongoing forward post-copy migration. During the migration process, if the network link or the target VM fails, the migrated VM can be recovered by the source host from its latest consistent checkpoint sent by the target host. Evaluation reveals that this method causes no change to the duration of migration although an increase in the downtime of the VM is resulted in case of reverse incremental checkpoints. This overhead could be minimized by using event-based checkpoints rather than using periodic checkpoints.

The design aspects of the above solution is as follows.

- Reverse Incremental Checkpoints

  PostCopyFT uses a structure called **checkpoint store** for this purpose. This is a key-value store which can be located at the source or an external node. When the VM resumes after migration, its initial memory and execution state is captured and sent by PostCopyFT to the checkpoint store. From this point onward, periodic or event-based incremental checkpoints are transferred to the checkpoint store. When the VM's migration is complete, the *checkpointing* scheme can be terminated.

- Recovering From Failure.

  When the network link or the target host fails during the migration process, the recovering process can be started. This can be done by restoring the VM's last consistent image within the checkpoint store. The VM is then resumed from its last checkpointed execution state.

- Network Buffering.

  All incoming packets are delivered directly to the VM without delay. However, all outgoing packets are buffered in between checkpoints. This is to ensure that the external world always communicates with a consistent VM state.

- Checkpoint Overhead Reduction.

  PostCopyFT obtains a migration duration similar to the post-copy method by running the active-push phase and the checkpointing mechanism concurrently. A separate thread executes the reverse checkpointing. To not increase the downtime through this mechanism, PostCopyFT divides the checkpointing into two stages in write-intensive applications. In the first stage, only

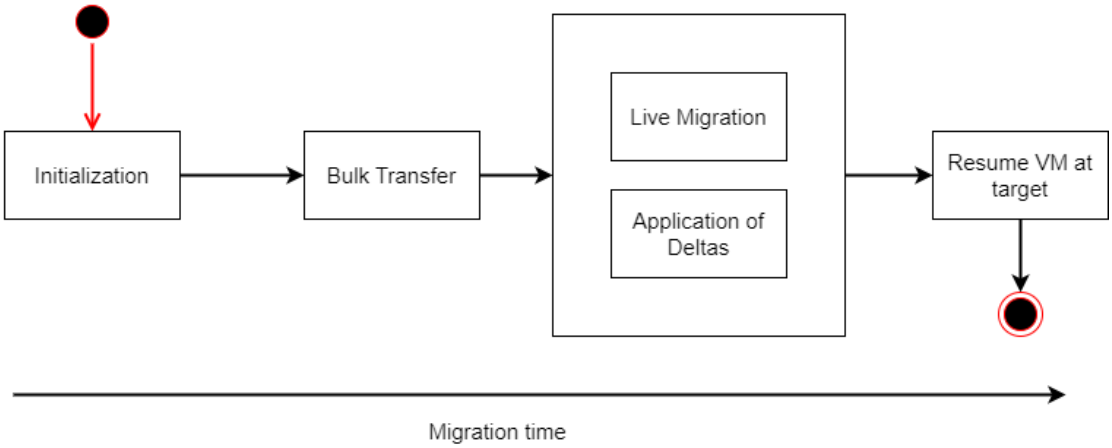

Figure 7: WAN Migration

modified pages (excluding the VM execution state) are checkpointed and stored locally while in the second stage the VM execution state along with the pages are checkpointed and transferred.

Often, groups of VMs are placed together in one source to reduce the communication cost among them. In this case, it would be beneficial migrating the VMs together as a cluster. Deshpande et al. (2011) provides a method to migrate multiple VMs parallely. VMs within a single server often have identical memory pages. Hence this mechanism is optimized to minimize the total memory content to be transferred by migrating identical pages only once (*deduplication).*

Considering migration of multiple VMs, Deshpande et al. (2012) provides **Inter-rack Live Migration (IRLM)** which is a distributed system for migrating several VMs parallely. *Deduplication* mechanism is used to minimize the amount of pages that need to be moved over the network (network traffic). Comparing this with **Gang Migration** Deshpande et al. (2011), it uses distributed deduplication (VMs running on different physical servers) as opposed to deduplication only within a single server.

As mentioned before in this paper, different migration mechanisms work for different types of workload. Addressing this issue Li et al. (2021)'s **AdaMig** is capable of stopping inefficient migration and changing the adapted migration scheme in the middle of the migration process. The study mentions of not needing to resend duplicated dirty pages or terminating the migration process. When migration is initiated, AdaMig starts transferring memory pages. There's a *Tendency Detector* which analyzes the data collected throughout this process and through this, AdaMig predicts whether the migration would fail or is in need of some parameter adjustments. If there's a tendency for the migration to fail, AdaMig chooses a different migration scheme. This provides a solution for unpredictable workload in VM applications.

3.4.2 WAN Setting

Most state-of-the-art migration mechanisms consider LAN migration as it is much more simpler than WAN. This is because in WAN migration it is necessary that the network connections and storage are transferred additional to the VM. Bradford et al. (2007) addresses both these concerns in their study.

The proposed solution operates on Xen VMM and the source and target are connected by TCP/IP protocol. All writes made while migration by the VM is intercepted by the solution and applied to the file system at the source server. Then, through the delivery of network packets in a reliable and orderly fashion, the updates are later applied to the file system of the target server. Fetching of the local persistent state is also done in a pre-copy manner rather than fetching-on-demand to further minimize downtime and performance degradation.

Additionally, network redirection scheme is introduced by the solution where all network connections are kept alive when the VM's IP address is updated during migration. Hence, new requests are redirected automatically to the updated IP address.

Looking at the architecture in detail (Figure 7), the *Initialization* stage is where the logistics are handled with respect to the target host such as setting up the required resources. In the next stage, *Bulk Transfer* transfers the persistent state (disk image) of the VM while it keeps running at the source. The next stage is where the live migration of the execution state of the VM happens. All writes made to the disk during this stage and the previous stage (bulk transfer), are intercepted and applied on the source which are also later applied on the target file system (*Application of Deltas* stage). This happens concurrently with the live migration. Throughout this process, the network redirection scheme also works if the VM migrates through the internet and uses a different IP than the source IP.

The issue of storage migration over WAN has been addressed in different ways over different studies. Similar to the above method by Bradford et al. (2007), Luo et al. (2008) uses a bitmap to keep record of all the write accesses in their **Three-phase migration** scheme (Figure 8). The disk image is then transferred concurrently with the memory pages to the target server.

Sapuntzakis et al. (2002) proposes a method based on Copy-On-Write (COW) disks. Whenever a *capsule* (A VM in a running state) is started, all disk updates from that point onwards are saved in a hierarchical manner. When capsules need to be migrated, the latest disk image can be fetched (not at once) on demand while the capsule runs at the target. However, Hirofuchi et al. (2009) argues in their study that this method is not ideal for virtualized CDCs since it is difficult to define a generalized approach which must suit for all kinds of VMs in a virtualized datacenter.

Hirofuchi et al. (2009) presents their own solution of storage migration in their study which is independent of VMM implementations. A *target server* and a *proxy server* is used for this mechanism. The target server is connected to both source and proxy servers. At the pre-copying stage of live migration, the target server continues to update the disk image at the source server. The disk is then migrated by to destination by the proxy server after the execution state of the VM

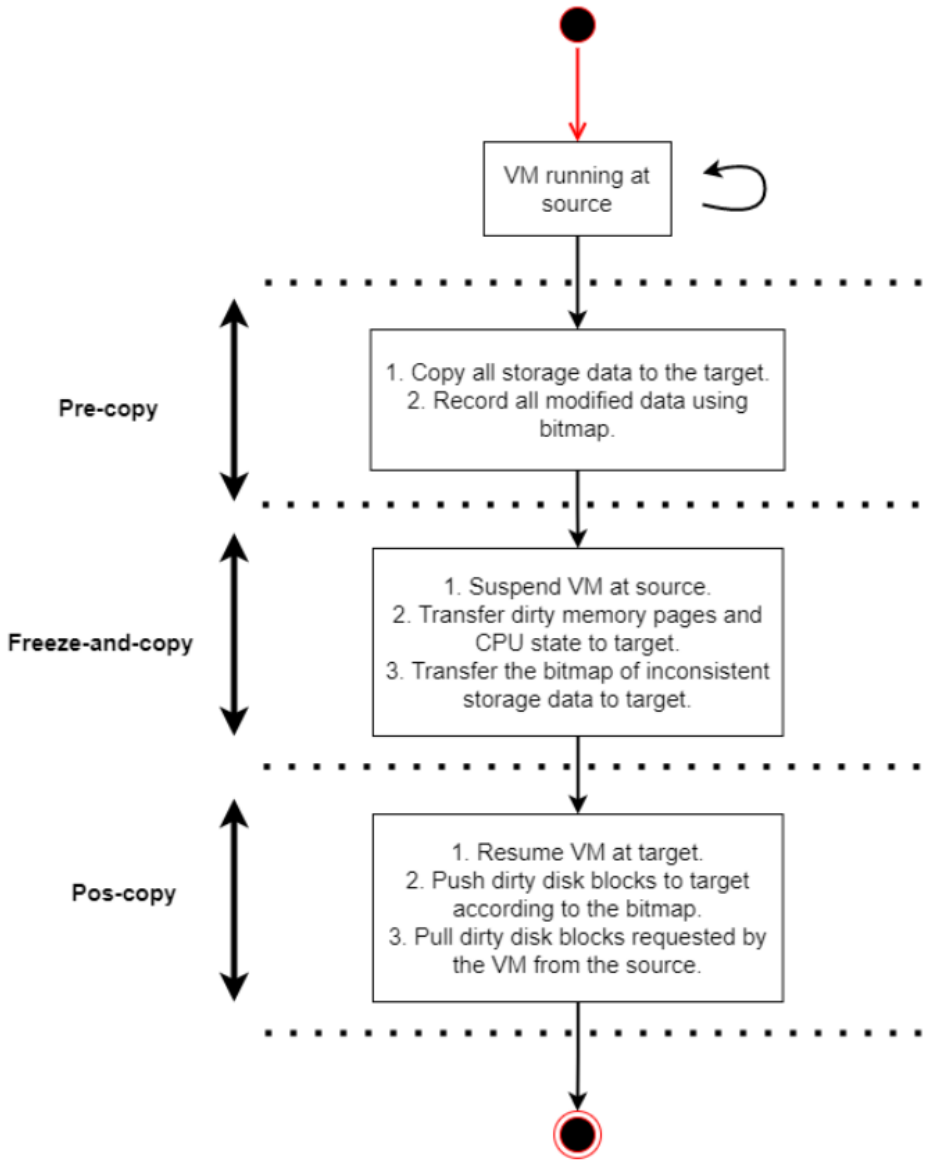

Figure 8: Three-Phase-whole-system Live Migration

is migrated. Relevant I/O requests are also redirected to the target server by the proxy server for blocks that aren't fetched yet. The proxy server then terminates the connection when all blocks are successfully migrated over.

CloudNet which is Wood et al. (2011)'s solution for WAN migration, provides a unique solution to connection geographically separated servers. The solution abstracts the existing resources in to a dynamic pool of resources similar to being connected over a LAN such that network configuration required for this migration scheme is minimal. Considering the disk state migration, initially, the disk image is copied (replicated) to a remote server in an asynchronous mode. Once the disk state becomes consistent, the memory transfer and the VM state transfer is carried out.

4 Optimization Techniques on Live Migration

State-of-the-art VM migration methodologies have been subjected to optimization techniques by academics throughout their lifetime. These techniques can be categorized with respect to the performance metric they try to optimize. In this section, few of the most noted optimization mechanisms will be discussed in detail.

Downtime of a VM is the attribute which measures the transparency of a migration scheme. For this reason, research enthusiasts have taken a lot of focus on this metric. Several studies have decreased the downtime of a VM by suppressing the amount of memory that is to be migrated. Migration performance can be significantly optimized by reducing the amount of memory to be migrated. One such mechanism is **Deduplication** of memory pages. VMMs could avoid sending memory pages which are identical within hosts with identical or similar OSs. Riteau et al. presents 'Shrinker' which compares hashes of the pages before migrating and only migrates if a page with the same hash value is not found within the destination server. Jin et al. (2009) also presents data compression mechanism to **compress memory pages** to provide fast migration. Migration schemes such as Koto et al. (2012) optimize the memory content to be transferred by cutting down the *soft pages* (**free pages filtering**) which are unnecessary for the VM to work accurately after the migration.

During pre-copy migration, convergence of the iterative pre-copy rounds is further optimized by Zhu et al. (2013) in their study. Termination of pre-copy rounds too late in the migration process for memory-intensive applications would severely cause performance degradation and consumption of network resource. The study presents a **smart iteration-termination** mechanism to know when to stop the iterations in live migration.

Liu, Jin, Liao, Yu & Xu (2011), in their study proposes a novel method of logging the execution of a VM. This record of execution trace is then migrated (instead of the actual memory pages) and the target and source are synchronized. The method drastically reduces the network traffic and migration duration.

Migration duration of a scheme could be reduced immensely by adapting compression and deduplication mechanisms Deshpande et al. (2012). **Quick Eviction** by Fernando et al. (2016) is a method to minimize the migration duration in which

snapshots of memory are sent to the destination host regularly. This happens prior to the migration process. When a migration needs to happen, the pages that need to be migrated are only the ones which are modified from the last consistent snapshotted state. Hence, the total memory that needs to be transferred in this case are significantly less.

When considering the application QoS degradation, the degradation duration can be minimized by,

- allocating system resources required for the VMM,
- choosing simple and fast data compression algorithms,
- deduplication is adapted and,
- application workload suits the type of migration adapted.

Bandwidth utilization of a migration scheme can be optimized by adapted several of the methods mentioned above including deduplication, compression, altering free memory pages Koto et al. (2012) and dynamic rate-limiting (Svärd, Hudzia, Tordsson & Elmroth (2011); Deshpande et al. (2012); Jin et al. (2009); Deshpande et al. (2011))

When considering post-copy migration, the way in which memory content is fetched from the source after the transfer of the VM state could provide benefits in different regards. Hence there exists several variants of this migration scheme.

The dependency with the source host is reduced much faster with the **Active Push** mechanism that is adapted. The source 'pushes' pages actively while the VM continues to run at the target. Page faults can also be delivered concurrently.

The amount of page-faults must be minimized to optimize the post-copy migration as addressing page-faults is quite expensive. This can be accomplished by pushing the memory before the target server experiences a page fault. One such mechanism which allows this is **adaptive prepaging** (Hines et al. 2009). Prepaging algorithms analyze the page faults and predicts the locality of the page access and transfers memory within the neighbourhood of a page fault prior to them being required by the VM.

Dynamic Self Ballooning (DSB) is an approach to prevent transferring invaluable pages. Invaluable in this case means free, un-allocated pages. To prevent the unnecessary transfer of these free pages, DSB analyzes the memory usage of a VM and adjusts the amount of memory allocated to it in real-time. When the VMM detects that the guest kernel is not using a this amount of memory, it can reclaim that memory and mark the corresponding pages as free. When migration happens, the VMM only transfers pages that are allocated (not the free pages). As a result, the migration becomes much faster. DSB is typically a feature implemented in VMMs.

Often times, when considering the migration of multiple VMs, majority of migration schemes treat all of the VMs same. However, VMs differ according to their application workload. The effect of the order of migration when migrating multiple VMs is studied by Fernando et al. (2020). The study proposes **SOLive** which

analyzes the usage of resources of each VM that is to be migrated, before starting the migration. A specific order of migration is then created by the component *migration scheduler*. The study reveals that this method reduces the migration duration significantly.

5 Critical Analysis of Work

In this section, several VM migration methods are compared against one another under several categories. The categories that are considered will be the Duration of Migration, Downtime, Duration of Performance Degradation, Network Link, Bandwidth Utilization Efficiency, Hypervisor Type and Migration Granularity.

5.1 Duration of Migration

The migration duration would be relatively low for the following reasons.

- Adapting fast compression algorithms.
- Adapting deduplication.
- Allocating resources for VM migration daemon.
- Write throttling.

It would be relatively high for the following reasons.

- No compression or deduplication methods adapted.
- VM memory is more diverse.
- Iterative pre-copy rounds are terminated forcefully.
- Write-intensive applications are migrated with pre-copy (non-optimized) migration schemes.
- High contention for resources.

Additionally, contrasting to live migration, non-live migration mechanisms undergo higher migration duration as a result of transferring a complete VM across the network at once.

Migration Scheme	Migration Duration
IRLM (Deshpande et al. 2012)	High
XBRLE (Svärd, Hudzia, Tordsson & Elmroth 2011)	Medium
CloudNet (Wood et al. 2011)	High
Checkpoint Recovery (Gerofi et al. 2011)	Low
Gang Migration (Deshpande et al. 2011)	High
Shrinker (Riteau et al. 2011)	High
Delta-Pre-copy (Svard, Tordsson, Hudzia & Elmroth 2011)	Low
Sonic Migration (Koto et al. 2012)	Low
Internet Suspend/Resume (Kozuch & Satyanarayanan 2002)	Medium
CR/TR-Motion (Liu, Jin, Liao, Yu & Xu 2011)	Low
HPC Migration (Ibrahim et al. 2011)	Medium
DSB Migration (Hines et al. 2009)	Low
HMDC Migration (Hu et al. 2013)	Low
Zap (Osman et al. 2002)	High

5.2 Downtime

When downtime of a VM is considered, if there is less memory to transfer to the destination, it is quite low. This can be done using compression of memory or deduplication (Riteau et al. (2011); Jin et al. (2009); Svard, Tordsson, Hudzia & Elmroth (2011)). It also can be minimized by choosing an optimal termination point for the iterative pre-copy rounds (Bradford et al. 2007). Downtime would be relatively high for the following reasons.

- There's no progress in the pre-copy rounds.
- Memory that is transferred is write-intensive.
- A post-copy method (that is not optimized) is chosen.
- Multiple VMs are migrated at once.
- Non-live migration is chosen as opposed to live-migration.

Migration Scheme	Downtime
IRLM (Deshpande et al. 2012)	High
XBRLE (Svärd, Hudzia, Tordsson & Elmroth 2011)	Medium
CloudNet (Wood et al. 2011)	High
Checkpoint Recovery (Gerofi et al. 2011)	Low
Gang Migration (Deshpande et al. 2011)	High
Shrinker (Riteau et al. 2011)	High
Delta-Pre-copy (Svard, Tordsson, Hudzia & Elmroth 2011)	Low
Internet Suspend/Resume (Kozuch & Satyanarayanan 2002)	High
CR/TR-Motion (Liu, Jin, Liao, Yu & Xu 2011)	Low
HPC Migration (Ibrahim et al. 2011)	Medium
DSB Migration (Hines et al. 2009)	High
HMDC Migration (Hu et al. 2013)	Low
Zap (Osman et al. 2002)	High

5.3 Duration of Performance Degradation

The duration of the application's QoS would be relatively low for the following reasons.

- Allocating resources for VM migration daemon.
- Adapting fast compression algorithms.
- Adapting deduplication.
- A suitable migration pattern is chosen for the application's workload.

Adapting write throttling would result in a higher duration of performance degradation. Post-copy methods also typically result in higher duration of performance degradation because of the frequent page faults. This can be minimized by optimizing the post-copy migration by adapting previously mentioned mechanisms such as pre-paging and active-push. When considering application's workload, a suitable migration pattern must be chosen. For example, in case of write-intensive migration, choosing a pre-copy migration would increase the duration of performance degradation as the rate of page dirtying is high. Additionally, when compared to live-migration mechanisms, non-live migration schemes have a higher duration of performance degradation as a result of service interruption.

Migration Scheme	Duration of QoS Degradation
IRLM (Deshpande et al. 2012)	High
XBRLE (Svärd, Hudzia, Tordsson & Elmroth 2011)	Medium
CloudNet (Wood et al. 2011)	High
Checkpoint Recovery (Gerofi et al. 2011)	Medium
Gang Migration (Deshpande et al. 2011)	High
Shrinker (Riteau et al. 2011)	High
Delta-Pre-copy (Svard, Tordsson, Hudzia & Elmroth 2011)	Medium
Sonic Migration (Koto et al. 2012)	Medium
Internet Suspend/Resume (Kozuch & Satyanarayanan 2002)	High
CR/TR-Motion (Liu, Jin, Liao, Yu & Xu 2011)	Low
HPC Migration (Ibrahim et al. 2011)	Medium
DSB Migration (Hines et al. 2009)	Medium
HMDC Migration (Hu et al. 2013)	Medium
Zap (Osman et al. 2002)	High

5.4 Network Link

Network links used for the migration mechanisms can be categorized as LAN links and WAN links. The majority of existing migration schemes have adapted LAN link as LAN migration does not necessitate moving the VM's storage. This is because in LAN migration, both source and target can access the NAS. Compared to LAN migration, there are quite a few complications within WAN migration due to additional overhead of storage migration, higher rate of dropped packets, higher latencies etc. Below table depicts the type of network link used in each migration scheme.

Migration Scheme	Network Link
IRLM (Deshpande et al. 2012)	LAN
XBRLE (Svärd, Hudzia, Tordsson & Elmroth 2011)	LAN
CloudNet (Wood et al. 2011)	WAN
Checkpoint Recovery (Gerofi et al. 2011)	LAN
Gang Migration (Deshpande et al. 2011)	LAN
Shrinker (Riteau et al. 2011)	WAN
Delta-Pre-copy (Svard, Tordsson, Hudzia & Elmroth 2011)	WAN
Sonic Migration (Koto et al. 2012)	LAN
Internet Suspend/Resume (Kozuch & Satyanarayanan 2002)	LAN
CR/TR-Motion (Liu, Jin, Liao, Yu & Xu 2011)	LAN
HPC Migration (Ibrahim et al. 2011)	LAN
DSB Migration (Hines et al. 2009)	LAN
HMDC Migration (Hu et al. 2013)	LAN
Zap (Osman et al. 2002)	LAN

5.5 Bandwidth Utilization Efficiency

Migrating a VM efficiently is quite a complicated task as there are limited resources available. As the network bandwidth is a resource that is shared among other co-hosted VMs as well, not utilizing it efficiently would lead to negative effects on the co-hosted VMs as well. Among the techniques adapted to utilize the network bandwidth are,

- compression of memory,
- deduplication,
- dynamic rate limiting and,
- free pages filtering.

Some of the migration schemes such as the study presented by Hines et al. (2009) have been optimized to use the bandwidth efficiently using many methods whereas others have only adapted few bandwidth optimization techniques.

Below table depicts whether bandwidth utilization mechanisms have been adapted in the studies considered.

Migration Scheme	Bandwidth Utilization Efficiency
IRLM (Deshpande et al. 2012)	Yes
XBRLE (Svärd, Hudzia, Tordsson & Elmroth 2011)	Yes
CloudNet (Wood et al. 2011)	Yes
Checkpoint Recovery (Gerofi et al. 2011)	Yes
Shrinker (Riteau et al. 2011)	Yes
Delta-Pre-copy (Svard, Tordsson, Hudzia & Elmroth 2011)	Yes
Sonic Migration (Koto et al. 2012)	Yes
Internet Suspend/Resume (Kozuch & Satyanarayanan 2002)	Yes
CR/TR-Motion (Liu, Jin, Liao, Yu & Xu 2011)	Yes
HPC Migration (Ibrahim et al. 2011)	No
DSB Migration (Hines et al. 2009)	Yes
HMDC Migration (Hu et al. 2013)	Yes
Zap (Osman et al. 2002)	No

5.6 Hypervisor Type

This attribute represents the hypervisor tool that is used for the migration. Most of the migration schemes have used *Xen* and *KVM* as their hypervisor.

Below table mentions the hypervisor or the VMMs used by the migration scheme.

Migration Scheme	Hypervisor Type
IRLM (Deshpande et al. 2012)	KVM
XBRLE (Svärd, Hudzia, Tordsson & Elmroth 2011)	KVM
CloudNet (Wood et al. 2011)	Xen
Checkpoint Recovery (Gerofi et al. 2011)	KVM
Gang Migration (Deshpande et al. 2011)	KVM
Shrinker (Riteau et al. 2011)	KVM
Delta-Pre-copy (Svard, Tordsson, Hudzia & Elmroth 2011)	KVM
Sonic Migration (Koto et al. 2012)	Xen
CR/TR-Motion (Liu, Jin, Liao, Yu & Xu 2011)	Xen
HPC Migration (Ibrahim et al. 2011)	KVM
DSB Migration (Hines et al. 2009)	Xen
HMDC Migration (Hu et al. 2013)	Xen

5.7 Migration Granularity

The migrating tool could migrate one VM or several VMs depending on the use case. *Migration Granularity* defines this level of granularity adapted by the migrating scheme. As expected, migrating several VMs at once has added complexity due to the limited resources available. Therefore migrating multiple VMs affect the downtime, performance degradation duration and the performance of other co-hosted VMs.

Below table depicts the granularity level of the migration mechanisms.

Migration Scheme	Migration Granularity
IRLM (Deshpande et al. 2012)	Multiple (Parallel)
XBRLE (Svärd, Hudzia, Tordsson & Elmroth 2011)	Single
CloudNet (Wood et al. 2011)	Single
Checkpoint Recovery (Gerofi et al. 2011)	Single
Gang Migration (Deshpande et al. 2011)	Multiple (Parallel)
Shrinker (Riteau et al. 2011)	Multiple (Sequence)
Delta-Pre-copy (Svard, Tordsson, Hudzia & Elmroth 2011)	Single
Sonic Migration (Koto et al. 2012)	Single
Internet Suspend/Resume (Kozuch & Satyanarayanan 2002)	Single
CR/TR-Motion (Liu, Jin, Liao, Yu & Xu 2011)	Single
HPC Migration (Ibrahim et al. 2011)	Single
DSB Migration (Hines et al. 2009)	Single
HMDC Migration (Hu et al. 2013)	Single
Zap (Osman et al. 2002)	Single

6 Conclusion and Future Work

This paper discussed the notion of how virtualization and VM migration technology has been used for cloud services. It thoroughly analyzed the VM migration process and state-of-the-art VM migration methodologies. Variants of these mechanisms are explained in-detail under different categories and the existing optimization techniques to improve them.

In addition to providing benefits to cloud services, VM migration process also causes side effects. These side effects are dynamic and depends on the migration scheme that has been adapted. When considering live migration, three categories are defined as Post-copy migration, Pre-copy migration and Hybrid migration. Write-intensive and read-intensive applications perform well in post-copy migration and pre-copy migration respectively. However, when considering WAN based migrations, there are issues with respect to migration of storage, limited network bandwidth and redirection of network connection.

Migrating large VM memory and gathering adequate system resources are problems of state-of-the-art migration methods. Adapting optimization strategies like smart-iteration-termination criteria, memory compression and deduplication often lead to additional computational expenses.

Additionally, when considering the optimizations of migration schemes, a lot of focus is on how the migration is conducted but not enough focus is there on how to choose the proper destination host. More research can be conducted on areas such as how to choose the most optimal destination host with respect to the network traffic of a VM and how it contributes to the migration scheme's performance.

References

Ahmad, R. W., Gani, A., Ab. Hamid, S. H., Shiraz, M., Xia, F. & Madani, S. A. (2015), 'Virtual machine migration in cloud data centers: a review, taxonomy, and open research issues', *The Journal of Supercomputing* **71**, 2473–2515.

Al-Kiswany, S., Subhraveti, D., Sarkar, P. & Ripeanu, M. (2011), Vmflock: Virtual machine co-migration for the cloud, *in* 'Proceedings of the 20th international symposium on High performance distributed computing', pp. 159–170.

Alibaba Cloud (2023), 'Alibaba cloud'.
URL: `https://in.alibabacloud.com/?utm_key=se_1007710386&utm_content=se_1007710386&gclid=Cj0KCQiA8aOeBhCWARIsANRFrQFZhZk5PnatY5otgaHs5v4Z0-vFwNnjqDHoWZRGn3uXSJ0W_BnYIWQaAik3EALw_wcB`

Atif, M. & Strazdins, P. (2014), 'Adaptive parallel application resource remapping through the live migration of virtual machines', *Future Generation Computer Systems* **37**, 148–161.

AWS (2023), 'Amazon web services'.
URL: `https://aws.amazon.com/`

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. & Warfield, A. (2003), 'Xen and the art of virtualization', *ACM SIGOPS operating systems review* **37**(5), 164–177.

Boss, G., Malladi, P., Quan, D., Legregni, L. & Hall, H. (2007), 'Cloud computing', *IBM white paper* **321**, 224–231.

Bradford, R., Kotsovinos, E., Feldmann, A. & Schiöberg, H. (2007), Live wide-area migration of virtual machines including local persistent state, *in* 'Proceedings of the 3rd international conference on Virtual execution environments', pp. 169–179.

Bugnion, E., Devine, S., Rosenblum, M., Sugerman, J. & Wang, E. Y. (2012), 'Bringing virtualization to the x86 architecture with the original vmware workstation', *ACM Transactions on Computer Systems (TOCS)* **30**(4), 1–51.

Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I. & Warfield, A. (2005), Live migration of virtual machines, *in* 'Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2', pp. 273–286.

Constantine, P. S., Ramesh, C., Ben, P., Jim, C., Monica, S. L. & Mendel, R. (2002), 'Optimizing the migration of virtual computers', *ACM SIGOPS Operating Systems Review* **36**(SI), 377–390.

Deshpande, U., Kulkarni, U. & Gopalan, K. (2012), Inter-rack live migration of multiple virtual machines, *in* 'Proceedings of the 6th international workshop on Virtualization Technologies in Distributed Computing Date', pp. 19–26.

Deshpande, U., Wang, X. & Gopalan, K. (2011), Live gang migration of virtual machines, *in* 'Proceedings of the 20th international symposium on High performance distributed computing', pp. 135–146.

Fernando, D., Bagdi, H., Hu, Y., Yang, P., Gopalan, K., Kamhoua, C. & Kwiat, K. (2016), Quick eviction of virtual machines through proactive snapshots, *in* '2016 IEEE International Conference on Cluster Computing (CLUSTER)', IEEE, pp. 156–157.

Fernando, D., Terner, J., Gopalan, K. & Yang, P. (2019), Live migration ate my vm: Recovering a virtual machine after failure of post-copy live migration, *in* 'IEEE INFOCOM 2019-IEEE Conference on Computer Communications', IEEE, pp. 343–351.

Fernando, D., Yang, P. & Lu, H. (2020), Sdn-based order-aware live migration of virtual machines, *in* 'IEEE INFOCOM 2020-IEEE conference on computer communications', IEEE, pp. 1818–1827.

GCP (2023), 'Google cloud platform'.
URL: `https://cloud.google.com/`

Gerofi, B., Vass, Z. & Ishikawa, Y. (2011), Utilizing memory content similarity for improving the performance of replicated virtual machines, *in* '2011 Fourth IEEE International Conference on Utility and Cloud Computing', IEEE, pp. 73–80.

Hines, M. R., Deshpande, U. & Gopalan, K. (2009), 'Post-copy live migration of virtual machines', *ACM SIGOPS operating systems review* **43**(3), 14–26.

Hirofuchi, T., Ogawa, H., Nakada, H., Itoh, S. & Sekiguchi, S. (2009), A live storage migration mechanism over wan for relocatable virtual machine services on clouds, *in* '2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid', IEEE, pp. 460–465.

Hu, L., Zhao, J., Xu, G., Ding, Y. & Chu, J. (2013), 'Hmdc: Live virtual machine migration based on hybrid memory copy and delta compression', *Appl. Math* **7**(2L), 639–646.

Ibrahim, K. Z., Hofmeyr, S., Iancu, C. & Roman, E. (2011), Optimized pre-copy live migration for memory intensive applications, *in* 'Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis', pp. 1–11.

Jiang, J. W., Lan, T., Ha, S., Chen, M. & Chiang, M. (2012), Joint vm placement and routing for data center traffic engineering, *in* '2012 Proceedings IEEE INFOCOM', IEEE, pp. 2876–2880.

Jin, H., Deng, L., Wu, S., Shi, X. & Pan, X. (2009), Live virtual machine migration with adaptive, memory compression, *in* '2009 IEEE International Conference on Cluster Computing and Workshops', IEEE, pp. 1–10.

Kivity, A., Kamay, Y., Laor, D., Lublin, U. & Liguori, A. (2007), kvm: the linux virtual machine monitor, *in* 'Proceedings of the Linux symposium', Vol. 1, Dttawa, Dntorio, Canada, pp. 225–230.

Koto, A., Yamada, H., Ohmura, K. & Kono, K. (2012), Towards unobtrusive vm live migration for cloud computing platforms, *in* 'Proceedings of the Asia-Pacific Workshop on Systems', pp. 1–6.

Kozuch, M. & Satyanarayanan, M. (2002), Internet suspend/resume, *in* 'Proceedings fourth IEEE workshop on mobile computing systems and applications', IEEE, pp. 40–46.

Kozuch, M., Satyanarayanan, M., Bressoud, T. & Ke, Y. (2002), 'Efficient state transfer for internet suspend/resume', *Intel Research Pittburgh, Tech. Rep. IRP-TR-02-03* .

Li, H., Xiao, G., Zhang, Y., Gao, P., Lu, Q. & Yao, J. (2021), Adaptive live migration of virtual machines under limited network bandwidth, *in* 'Proceedings of the 17th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments', pp. 98–110.

Liu, H., Jin, H., Liao, X., Yu, C. & Xu, C.-Z. (2011), 'Live virtual machine migration via asynchronous replication and state synchronization', *IEEE Transactions on Parallel and Distributed Systems* **22**(12), 1986–1999.

Liu, H., Xu, C.-Z., Jin, H., Gong, J. & Liao, X. (2011), Performance and energy modeling for live migration of virtual machines, *in* 'Proceedings of the 20th international symposium on High performance distributed computing', pp. 171–182.

Luo, Y., Zhang, B., Wang, X., Wang, Z., Sun, Y. & Chen, H. (2008), Live and incremental whole-system migration of virtual machines using block-bitmap, *in* '2008 IEEE International Conference on Cluster Computing', IEEE, pp. 99–106.

Medina, V. & García, J. M. (2014), 'A survey of migration mechanisms of virtual machines', *ACM Computing Surveys (CSUR)* **46**(3), 1–33.

Microsoft (2023*a*), 'Microsoft azure'.
URL: `https://azure.microsoft.com/en-us`

Microsoft (2023*b*), 'Microsoft hyper-v'.
URL: `https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v,urldate={2023-01-16}`

Milojičić, D. S., Douglis, F., Paindaveine, Y., Wheeler, R. & Zhou, S. (2000), 'Process migration', *ACM Computing Surveys (CSUR)* **32**(3), 241–299.

Oracle (2023), 'Oracle vm virtual box'.
URL: https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.html,urldate={2023-01-16}

Osman, S., Subhraveti, D., Su, G. & Nieh, J. (2002), 'The design and implementation of zap: A system for migrating computing environments', *ACM SIGOPS Operating Systems Review* **36**(SI), 361–376.

Padala, P., Zhu, X., Wang, Z., Singhal, S., Shin, K. G. et al. (2007), 'Performance evaluation of virtualization technologies for server consolidation', *HP Labs Tec. Report* **137**.

Princess, J., Jeba, G., Paulraj, L. & Jebadurai, I. (2018), 'Methods to mitigate attacks during live migration of virtual machines–a survey', *Int. J. Pure Appl. Math* **118**(20), 3663–3670.

QEMU (2023), 'Qemu'.
URL: https://www.qemu.org/,urldate={2023-01-16}

Red Hat, Inc (2023), 'Redhat virtualization'.
URL: https://www.redhat.com/en/technologies/virtualization/enterprise-virtualization,urldate={2023-01-16}

Riteau, P., Morin, C. & Priol, T. (2011), Shrinker: Improving live migration of virtual clusters over wans with distributed data deduplication and content-based addressing, *in* 'Euro-Par 2011 Parallel Processing: 17th International Conference, Euro-Par 2011, Bordeaux, France, August 29-September 2, 2011, Proceedings, Part I 17', Springer, pp. 431–442.

Sapuntzakis, C. P., Chandra, R., Pfaff, B., Chow, J., Lam, M. S. & Rosenblum, M. (2002), 'Optimizing the migration of virtual computers', *ACM SIGOPS Operating Systems Review* **36**(SI), 377–390.

SolarWinds Worldwide, LLC (2023), 'Solarwinds virtualization manager'.
URL:
https://www.solarwinds.com/virtualization-manager/use-cases/virtualization-monitoring?CMP=BIZ-RVW-SWTH-mngmt,urldate={2023-01-16}

Svärd, P., Hudzia, B., Tordsson, J. & Elmroth, E. (2011), Evaluation of delta compression techniques for efficient live migration of large virtual machines, *in* 'Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments', pp. 111–120.

Svard, P., Tordsson, J., Hudzia, B. & Elmroth, E. (2011), High performance live migration through dynamic page transfer reordering and compression, *in* '2011 IEEE Third International Conference on Cloud Computing Technology and Science', IEEE, pp. 542–548.

Uddin, M., Shah, A., Alsaqour, R. & Memon, J. (2013), 'Measuring efficiency of tier level data centers to implement green energy efficient data centers', *Middle-East Journal of Scientific Research* **15**(2), 200–207.

V2 Cloud Solutions, Inc (2023), 'V2 cloud'.
URL: `https://v2cloud.com/?referrer=d42d1d2810c5b902e16313e3647f01686XaeD2bHbIO0zEGN4SQmX7UvHx6KqDeh,urldate={2023-01-16}`

VMware, Inc (2023), 'Vm ware workstation'.
URL: `https://www.vmware.com/products/workstation-pro.html,urldate={2023-01-16}`

Voorsluys, W., Broberg, J., Venugopal, S. & Buyya, R. (2009), Cost of virtual machine live migration in clouds: A performance evaluation, *in* 'IEEE international conference on cloud computing', Springer, pp. 254–265.

Wood, T., Ramakrishnan, K., Shenoy, P. & Van der Merwe, J. (2011), 'Cloudnet: dynamic pooling of cloud resources by live wan migration of virtual machines', *ACM Sigplan Notices* **46**(7), 121–132.

Zhang, F., Liu, G., Fu, X. & Yahyapour, R. (2018), 'A survey on virtual machine migration: Challenges, techniques, and open issues', *IEEE Communications Surveys Tutorials* **20**(2), 1206–1243.

Zhu, L., Chen, J., He, Q., Huang, D. & Wu, S. (2013), Itc-lm: a smart iteration-termination criterion based live virtual machine migration, *in* 'Network and Parallel Computing: 10th IFIP International Conference, NPC 2013, Guiyang, China, September 19-21, 2013. Proceedings 10', Springer, pp. 118–129.

Index