Interim Report

Workload Aware Live Migration of Virtual Machines

B.F. Ilma
2019cs061@stu.ucsc.cmb.ac.lk
Index number: 19000618

Supervisor: Dr. Dinuni K Fernando

May 2023

University of Colombo School of Computing
Colombo, Sri Lanka.

Declaration

The project proposal 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 project proposal is based on the work of Ms. B.F. Ilma under my supervision. The project proposal has been prepared according to the format stipulated and is of acceptable standard.

Supervisor Name : Dr. Dinuni K. Fernando

Signature & Date

Contents

List of Acronyms

CC Cloud Computing

CDCs Cloud Data Centers

DSR Design Science Research

DSB Dynamic Self Ballooning

DT Downtime

LAN Local Area Network

NFS Network File System

SLA Service Level Agreement

TMT Total Migration Time

VM Virtual Machine

WALM Workload Aware Live Migration

WAN Wide Area Network

WWS Writable Working Set

QMP QEMU Machine Protocol

1 Section 1

1.1 Introduction

Live migration of Virtual Machine (VM)s migrates a VM from one physical host to another. When considering the migration of VMs, the efficiency of the process is affected by several factors. One such factor is the type of workload that is running on the VMs.

Workload-aware Live Migration involves migrating VMs by considering their type of workload. Specifically, a VM's workload can be categorized (Fernando et al. 2020) as,

- Network intensive,
- CPU intensive and,
- Memory intensive.

The above workloads can also be categorized as read-intensive and write-intensive according to the way they utilize the VM's memory. From above, network and CPU-intensive workloads are read-intensive, as their memory-reads are greater and their memory-writes are lesser, while the opposite is true for the memory-intensive.

Workload-aware live migration's aim is to migrate the VM as quickly as possible from the source machine in case of an apparent failure. In this case, we focus on minimizing the duration of migration, and providing transparency to end users.

Workload-aware migration dynamically detects the nature of the workload running in the VM and migrates it by choosing the most efficient migrating method (Svärd et al. 2014). This will be done without the system administrator manually choosing a migration method.

When considering live migration, all pre-copy (Clark et al. (2005); Nelson et al. (2005)), post-copy (Hines & Gopalan (2009); Hines et al. (2009)) and hybrid methods (Sahni & Varma (2012)) are considered in this solution. When a VM needs to be migrated, workload-aware live migration first detects the type of workload it runs. This will be done by dynamically reading and analyzing the page dirtying rate, CPU and network usage and other characteristics of the VM. Using these readings, workload-aware live migration would migrate the VM in the most efficient method out of pre-copy, post-copy and hybrid, quickly as possible.

1.2 Background

Cloud Data Centers (CDCs) have adapted virtualization as a means to handle the large number of requests that have arisen as Cloud Computing (CC) got popular. When it comes to virtualization, one of the most important advantages of it is migration. When doing routine maintenance, load balancing, or upon failure detection, a VM may be migrated from one machine to another.

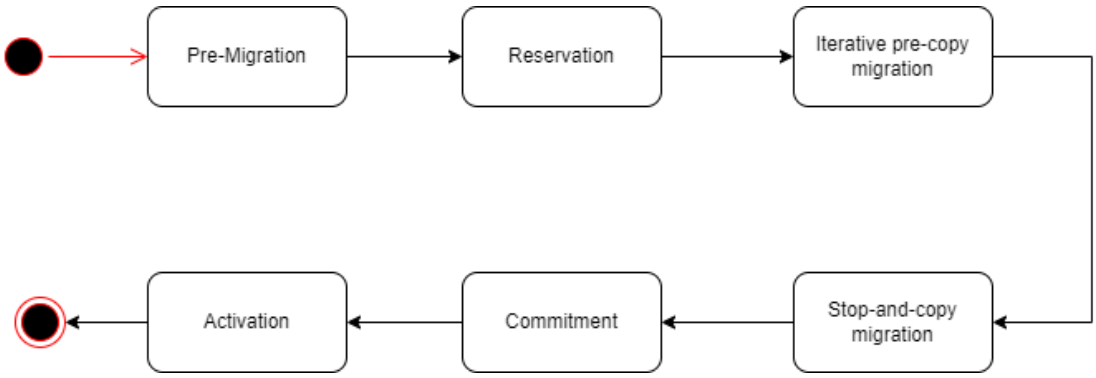

Figure 1: States of Pre-copy VM Migration (Clark et al. 2005)

All migration paradigms can be categorized into different groups. Of these, the most prominent aspect could be taken as the 'Liveliness' of a migration.

When we consider migrating a VM, for real-time services, it is important to do so in a transparent manner to the end-user. The end user should not feel significant service downtime. Hence, migration can be divided into **Live** and **Non-live**. In the latter, the VM is migrated by stopping its execution and transferring its CPU state and its memory all at once. Hence, in non-live migration, the VM would be suspended at the source host and only resumed once the migration is complete and the VM is transferred to the target host.

Migrating a VM while continuing to provide services is called Live migration. In live migration, the VM is migrated without breaking its connection to the end-user so as not to violate the Service Level Agreement (SLA). Live migration is adapted in many CDCs for its benefits such as hardware maintenance with near-zero downtime (Boss et al. 2007), server consolidation (Ala'Anzy & Othman 2019), load balancing, fault tolerance, etc.

There are three primary methods for categorizing live migration by considering the way of migrating the VM's memory pages. Namely they are,

- Pre-copy migration,
- Post-copy migration and,
- Hybrid migration.

Pre-copy migration involves migrating the memory pages before migrating the CPU state of the VM. Migrating the memory pages is done over a round of iterations. Initially, all the memory pages are migrated to the target host. On the next iteration onwards, only the pages that are modified are migrated (as the VM continues to run). At one point, the set of pages that are modified may be small enough that it could be migrated with the CPU state of the VM (stop-and-copy). This point is called the convergence. Pre-copy is mostly adapted to minimize the service degradation of a VM. States of pre-copy migration of VMs are illustrated in figure 1.

As opposed to pre-copy migration, post-copy migrates the CPU state before migrating the memory. This might result in page faults if the target machine

tries to access a memory page which still resides on the source machine. Pages could be fetched as they are faulted at the target machine or before. Transferring pages before they are faulted is called *Active Pushing*. Different optimization mechanisms are adapted to reduce the frequency of page faults occurring within the target machine during post-copy migration. States of post-copy migration of VMs are illustrated in figure 2.

Hybrid migration has been implemented to obtain the advantages of both pre-copy and post-copy methods. It encapsulates features of both pre-copy and post-copy methods. Figure 3 illustrates the states of hybrid migration of VMs. Specifically, hybrid migration could be regarded as a special instance of post-copy migration, albeit with a little improvement. To reduce the page faults occurring within the target machine after the migration of the CPU state, initially, the Writable Working Set (WWS) of the VM is transferred to the target. This is done in a pre-copy manner.

Existing VM live migration paradigms could be categorized into two sets as Local Area Network (LAN) based migrations and Wide Area Network (WAN) based migrations. In LAN based migrations, one does not have to migrate the VM storage (disk), as both source and target machines can access the storage through Network File System (NFS). This, however, becomes a necessity in WAN based migration, which causes it to be more complex than LAN based migration.

Different performance metrics are defined to warrant different aspects related to migration. Even with live migration, there's a time period where the VM cannot provide services due to the migration of the CPU state. This is known as the **Downtime**. **Migration Duration** defines the period of time from the initiation of the migration to its completion. **Network Bandwidth Utilization** which is measured by the network traffic and the migration duration, defines how the bandwidth is utilized.

1.3 Motivation

Servers within CDCs can be subjected to failures any time (Miller 2008). When we detect an an imminent failure of a host, we need to immediately migrate the VMs residing within the physical host. This is a critical issue as failure to quickly migrate the VMs residing in the server would result in service interruption to the end user and violation of Quality of Service.

Different migration methods perform well for different kinds of VM workloads. For example, as Hines et al. (2009) mentions in their study, pre-copying the memory of a VM suits more read-intensive workloads, while post-copying the memory of a VM is preferable for a write-intensive workload. This could be the general case; however, underlying characteristics of the workload of a VM can make the migration inefficient or at the very least cost more time when specific migration methods are used. Shah et al. (2015) analyses the performance of the pre-copy, post-copy and hybrid methods with respect to scientific workloads. The study specifically focuses on different page dirtying rates and memory sizes. The results show that for VMs with low memory running workloads with high page dirty rates, pre-copy migration would provide the minimal migration time. For VMs

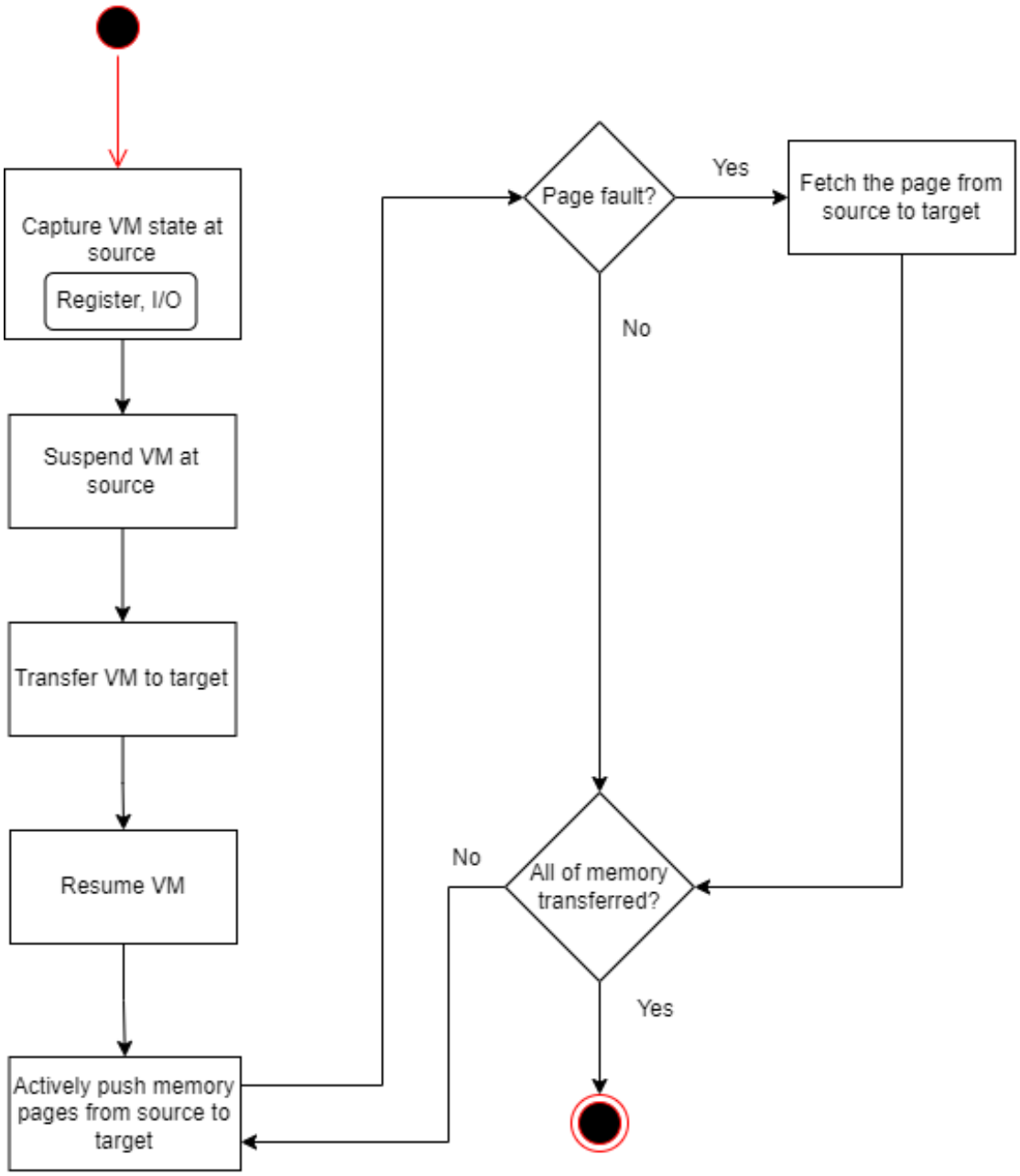

Figure 2: States of Post-copy VM Migration (Ahmad et al. 2015)

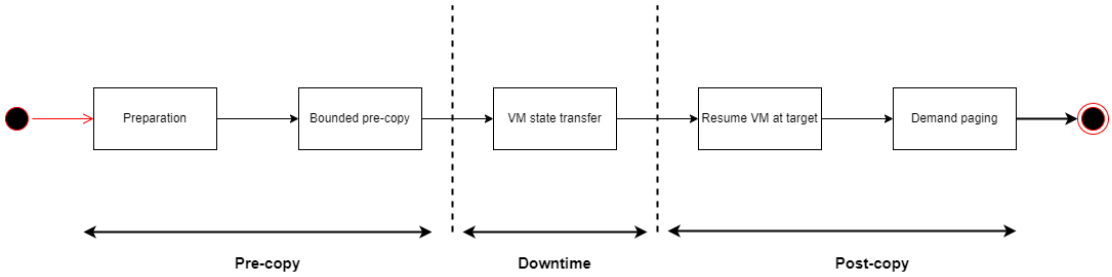

Figure 3: States of Hybrid VM Migration (Ahmad et al. 2015)

with large memory running workloads with low page dirty rate and high page dirty rate, post-copy migration provides the least migration time.

Hence, from the above results, it is clear that there are more aspects to consider (such as the VM memory size) in order to migrate a VM as quickly as possible by choosing the most efficient migration method.

As opposed to Shah et al. (2015)'s study, this research focuses on efficient migration of general workloads and not only within the scientific environment.

Li et al. (2021) mentions the attribute **'success ratio'** in their study, which explains the tendency of migrations to be completed successfully. The success ratio of migrations is often neglected in state-of-the-art migration mechanisms. As such, what it means for a migration to be 'successful', depends on the type of migration method adapted. For example, in the case of pre-copying memory, the migration would be successful if the pre-copy iterations could eventually converge. However, this would not be the case for write-intensive workloads.

Hence, it is obvious that the success of migration depends on the migration method and how well it is suitable for a given type of workload.

2 Section 2

2.1 Literature Review

To understand possible gaps within the state-of-the-art live migration paradigms, a preliminary literature review was conducted on the topic **Live migration of Virtual Machines**.

The foundation for non-live migration can be taken as **Process Migration** by Milojičić et al. (2000). This involves processes being abstracted to remove any dependencies on the host and migrated to another machine. Osman et al. (2002)'s Process Domain (Pod) abstraction method can be taken as an example for this. An excellent example of a non-live migration is Kozuch et al. (2002)'s Internet Suspend/Resume (ISR). This involves suspending the VM's state on the host machine and migrating it over the internet.

2.1.1 LAN Setting

Considering live migration methods under LAN, Ibrahim et al. (2011) presents an optimized pre-copy migration. To lessen the total migration time, the study presents a novel way of terminating the iteration rounds of pre-copy. Clark et al. (2005)'s study of pre-copy mentions that the WWS of a VM is the least ideal set of pages for pre-copying as they are frequently modified. Instead, they should be migrated with the CPU state in the stop-and-copy phase.

Migration duration could be minimized by reducing the content that needs to be migrated over to the target machine. Koto et al. (2012)'s Sonic Migration involves cutting down the memory pages that are not necessary (soft pages) for the VM to function after the migration. **Compression** also reduces the size of

data that needs to be migrated. Svärd et al. (2011)'s presents a new compression algorithm, **XBRLE** which reduces the size of data that needs to be migrated.

Hines et al. (2009) presents the post-copy live migration mechanism to reduce the duration of migration. However, one has to consider the consequences of a migration failure in post-copy. The study mentions that implementing *checkpoints* where the VM's execution state could be sent as a backup back to the source machine would be a solution, albeit increasing the reverse network traffic. Fernando et al. (2019)'s *PostCopyFt* provides this solution of reverse incremental checkpoints. Periodic or Event-based checkpoints are used to send the VM's state back to the source machine so that it can be recovered using its latest consistent checkpoint in case of a failure.

When considering VMs in a machine, they may have dependencies among them. In this case, it is necessary to migrate them together. Deshpande et al. (2011)'s **Gang Migration** migrates multiple VMs parallely. **Deduplication** optimization mechanism is used here where identical memory pages are only transferred once to the target machine (as VMs in the same server might share identical memory pages). Deshpande et al. (2012)'s *Inter-rack Live Migration (IRLM)* provides a method to migrate VMs parallely in a distributed system (VMs in different host machines) which uses distributed deduplication optimization scheme.

This research would be focused on LAN based migration.

2.1.2 WAN Setting

Live migration mechanisms in WAN are rare compared to the focus on LAN based migrations. This is because WAN based migration is quite complex than LAN based migration. Bradford et al. (2007) presents a method in their study where the persistent state of the VM is also migrated in a pre-copy manner. Luo et al. (2008) in their study, uses a bitmap to track dirtied disk blocks in the source. Initially, all disk storage is migrated (by pre-copying) to the target machine while also recording the dirty blocks using the bitmap. Then, this bitmap is migrated along with the CPU state (with the WWS). Then, dirty blocks are fetched by the target machine according to the bitmap. Wood et al. (2011)'s CloudNet presents a novel way of migrating the disk storage of a VM. The disk image is initially copied asynchronously to a remote server. Once the state of the disk is consistent, the migration of the VM's CPU state and memory pages takes place.

2.1.3 Optimization Techniques

Researchers focus on different optimization mechanisms to improve state-of-the-art live migration mechanisms. These optimizations consider one or more of the performance metrics.

When considering the *Downtime* of a migration, it could be minimized by reducing the memory content that needs to be transferred. **Deduplication** could be used to achieve this. Riteau et al. (2011)'s Shrinker only transfers pages with unique hash values (compared to hash values of pages in the target machine) to the target machine. Jin et al. (2009)'s study uses **Data Compression** to compress

memory pages before transfer, and Koto et al. (2012) filters free pages (soft pages) in this regard.

To converge the iterations of pre-copying more efficiently, Zhu et al. (2013) presents a **Smart Iteration-Termination** mechanism.

Liu et al. (2011) presents a novel method of migration via logging the execution state of the VM on the source machine. Then, instead of migrating actual memory pages, this execution trace record is transferred so as to synchronize the VM at the target machine.

Considering *Migration Duration*, it could also be reduced by compression and deduplication of memory pages, as demonstrated by Deshpande et al. (2012). Fernando et al. (2016) presents a novel method of sending snapshots of memory periodically to reduce the time of migration. In this case, one only needs to migrate the pages that have been modified since the last consistent snapshot.

Bandwidth Utilization of a migration could be made more efficient by a number of methods, including compression, filtering free memory pages, and **Dynamic Rate-Liming** (Svärd et al. 2011) and deduplication.

When post-copy migration is considered, the way memory pages are fetched after the migration of the CPU state could be improved so as to lessen the number of page faults. Hines et al. (2009)'s **Adaptive Prepaging** algorithm analyzes the page faults at the target machine and predicts the locality of page access so that pages can be migrated prior to them being faulted. **Dynamic Self Ballooning** prevents migrating unallocated pages by analyzing the memory usage of a VM in real-time and transferring only the pages that are allocated (not the ones that are free).

2.1.4 Related Work

There are several optimization mechanisms for minimizing the duration of migration of a VM including Dynamic Self Ballooning (DSB) (Hines et al. 2009), compression (Deshpande et al. 2011), quick eviction (Fernando et al. 2016) and deduplication (Deshpande et al. (2011); Deshpande et al. (2014)). None of these approaches consider the impact of the VM's workload on the migration process.

Li et al. (2019) addresses the problem of large downtimes and the violation of SLAs with respect to migrating memory-intensive workloads using pre-copy migration. To minimise the negative effects, the study addresses pages which are 'fake dirty' that don't need to be migrated as they are unnecessary and wasteful to be resent. The study explores how these 'fake dirty' pages are generated and then proposes a method to avoid resending them using secure hashes.

Ibrahim et al. (2011) studies the effect of pre-copy migration on scientific workloads such as OpenMP and MPI applications. The study proposes an optimized pre-copy migration method with a novel termination criteria which would prevent inifinite pre-copy iterations.

Fernando et al. (2020) presents an order-aware live migration method **'SO-Live'**, which addresses the significance of minimizing the duration of migration in case of an imminent failure of a server. The study considers VM workloads under the categories of CPU intensive, network intensive, and memory intensive and fo-

cuses on ordering the VMs by analyzing their workloads in the case of migrating multiple VMs. The VM workload characteristics, such as network usage, CPU usage, and memory usage, are dynamically captured so as to categorize them. The study mentions that by minimizing the contention for the finite resources by properly ordering the VMs, the total migration time is significantly reduced. The proposed research extends the workload analysis of this study so as to dynamically choose a migration method according to the category of the workload. In contrast to this study, the proposed research focuses on only migrating a single VM.

Li et al. (2021)'s **'AdaMig'** is an adaptive live migration mechanism. The study addresses problems such as inefficient migration and migration failures with respect to the VM workloads by analyzing the page dirtying rate and the migration speed. A static priority is given to the migration methods (pre-copy is prioritized) and in cases where pre-copy cannot be converged (for write-intensive workloads), they try to make the condition **"migration speed < page dirtying rate"** false by adapting different mechanisms such as CPU throttling and compression. If all of these fail, the VM is migrated via post-copy.

Li et al. (2021)'s study only focuses on pre-copy and post-copy migration methods (excluding hybrid migration) and only involves non-demanding workloads. This research, in contrast, would consider pre-copy, post-copy, and hybrid migration methods, and involve general workloads that can be demanding and cannot be handled via mechanisms such as vCPU throttling without losing transparency. Instead of halting an inefficient migration and choosing a different method, this research focuses on starting the migration with the most efficient method.

2.1.5 Summary

2.2 Research Gap

State-of-the-art live migration paradigms are not always the most optimal for a given type of VM workload. As studies (Clark et al. (2005); Fernando et al. (2019); Sahni & Varma (2012); Svärd et al. (2014); Shah et al. (2015)) prove that the efficiency and the success rate of a migration inherently depend on the type of workload the VM runs, we need to focus on adapting the migration method according to the type of workload.

However, state-of-the-art live migration mechanisms mostly aim to improve an aspect of migration, such as decreasing downtime, migration duration, or performance degradation. There is a lack of emphasis on how the characteristics of the workload impact the migration process and the need to adapt the migration strategy accordingly.

Additionally, there's a lack of migration paradigms where the aspects of the migration are changed dynamically according to the workload. Most existing migration paradigms involve the system administrator manually aborting the inefficient migration process and restarting it in case of a migration failure.

The above reasons indicate the necessity of a migration scheme that enables dynamic selection of a migration method based on the type of workload that runs in the VM. Addressing these concerns, workload-aware live migration focuses on

analyzing how general workloads running in the VM affect the migration process and exploring ways to adapt the migration method accordingly.

2.3 Research Questions

1. How can workload characteristics such as page dirtying rate, CPU usage, network usage, memory usage, etc. be effectively analyzed and classified to determine the most suitable migration method for a given virtual machine?

   This focuses on developing robust and accurate methods for dynamically detecting the type of workload running on a virtual machine. It involves investigating various metrics and techniques to analyze workload characteristics, such as page dirtying rate, CPU usage, network usage, WWS and other relevant factors. The goal is to establish a reliable classification system that can guide the selection of the most efficient migration method.

2. What are the performance implications of different migration methods (pre-copy, hybrid, post-copy) in workload-aware live migration?

   This delves into evaluating the performance of different migration methods when applied in the context of different workloads. It involves conducting experiments and measurements to compare factors such as the total migration time and resource utilization. The goal is to assess the trade-offs and effectiveness of each migration method and identify the optimal approach for different types of workloads.

2.4 Aims & Objectives

2.4.1 Aims

Optimizing the migration of VMs by dynamically considering the characteristics of their workloads while providing minimal total migration time and increased transparency.

2.4.2 Objectives

Research Question	Objectives
1. How can workload characteristics be analyzed and classified to determine the most suitable migration method for a given virtual machine?	• Identify workload metrics that can capture the characteristics of different types of workloads. • Identify the methods to capture the workload metrics dynamically while the VM is running. • Create a classification model that can classify the workloads according to the workload metrics
2. What are the performance implications of different migration methods (pre-copy, post-copy, hybrid) in workload-aware live migration?	• Determine the correlation between migration methods and workload characteristics. • Establish an algorithm that can select the most suitable migration method based on the workload analysis.

Table 1: Research objectives

2.5 Scope & Limitations

2.6 In Scope

The following areas will be covered under the scope of this research.

- Analyzing the nature of VM workloads based on their characteristics.
  - This involves researching and developing methods to analyze and classify the characteristics of different types of workloads, such as network-intensive, CPU-intensive, and memory-intensive workloads.
  - The focus is on identifying key metrics and techniques for workload characterization.
- Determining the most efficient migration methodology for a given workload.
  - Investigating and evaluating pre-copy, hybrid, and post-copy approaches, in the context of workload-aware migration.

  - This involves understanding the advantages, disadvantages, and trade-offs of each method and determining the most efficient approach based on workload characteristics.

- Developing an algorithm for workload-aware live migration.
  - Designing an algorithm for dynamically selecting the most suitable migration method based on workload characteristics.
  - This involves developing an efficient algorithm that can handle real-time workload analysis and adapt migration strategies accordingly.

- Performance evaluation
  - Conducting experiments and performance evaluations to assess the effectiveness of workload-aware live migration.
  - This includes measuring the duration of migration and resource utilization to quantify the benefits and limitations of the proposed methods.

2.7 Out Scope

The following will not be covered under the scope of this research.

1. WAN migrations
   - The research focuses on migration within LAN. This does not involve any specifications or challenges with respect to WAN migration.
2. Multi-tier VM applications
   - Inter-connected VMs that collaborate to provide specific services are not considered in this research. The research is involved in migrating an individual VM and does not involve any complexities or dependencies associated with multi-tier applications.
3. Multiple VM migrations
   - The research involves in migrating individual VMs and does not consider migration of multiple VMs at once.

2.8 Significance of the Project

Workload-aware live migration contributes to the following aspects.

As a result of considering the type of VM workload in the migration process, it selects the most efficient migration method. This optimization mechanism would provide benefits such as reducing the total migration time and consuming less system resources by avoiding unnecessary waste and bottlenecks. The overall aspects of the migration process are improved by Workload-aware live migration.

The study involves dynamically choosing a migration method based on the characteristics of the VM workload. Hence, it provides a way to migrate the VM

as quickly as possible with increasing transparency to the end user by way of minimal service disruption time.

Another significance of this research is the automated decision-making process when it comes to selecting a migration method. Traditionally, this is done with the interference of the system administrators, which can be time consuming and prone to human errors. In workload-aware live migration, this decision is automated based on an analysis of the VM workload, which eliminates the need for manual intervention and the risk of human errors.

Workload-aware live migration would be essential in maintaining the robustness of cloud data centers. The ability to migrate the VM to another physical machine as quickly as possible in case of an apparent failure is essential. By minimizing the duration of migration and providing transparency to the end users, this research contributes to enhancing the overall robustness of the cloud data centers.

This research would pave the way for future research in workload adaptation. This is significant as workload characteristics can evolve over time. The ability to dynamically adapt to the workload would ensure the efficiency of the migration methods and the performance of virtualized environments.

Hence, the advancements provided by this research have the potential to benefit various fields that rely on virtualization, such as Cloud Computing, High-Performance Computing (HPC) and Data Centers.

3 Section 3

3.1 Research Approach and Methodology

The preliminary study of the research can be broken down in to several phases.

The initial phase would encompass studying and implementing the baselines models of the research. This includes vanilla pre-copy, post-copy and hybrid migration methods.

QEMU (*QEMU* n.d.) by default has pre-copy and hybrid migration. Hence, initially, an analysis of how pre-copy and hybrid migration is implemented in QEMU would be conducted along with the pros and cons of using pre-copy migration and hybrid migration. The next steps would include how to implement post-copy migration and hybrid migration (with fixed iterations) in QEMU as well as the strengths and weaknesses of each migration method. A detailed description of QEMU will be given in Section 4.

The next phase is concerned with studying and analysing the workloads that run within the VMs. The first step is to choose synthetic and real-world workloads to study how they can be migrated in different ways. During this phase, experiments would be conducted to gather data by tweaking different parameters of the workloads such as the RAM size, size of the working set, memory usage, CPU usage, network usage etc.

The goal of this phase is to identify the representative workload metrics and study how each of these parameters affect the migration process and how to classify the VM workloads into different categories. Similar experiments would be

conducted for each of the migration methods to analyse how the migration of each kind of workload is affected by the migration method.

From the outcome of the above two phases, we can get a clear idea on what workload metrics need to be analyzed prior to the VM migration in order to classify the VM into a category and choose the most optimal migration method out of pre-copy, post-copy and hybrid.

During the implementation phase, we need to identify and implement a method to dynamically measure the workload metrics considered in the previous phase, prior to migration of a VM.

Once workload metrics can be analyzed dynamically, using this result, Workload Aware Live Migration can migrate the VM using the most optimal migration method for it specifically.

This research would follow the Design Science Research (DSR) methodology (Vom Brocke et al. 2020) where the solution for the workload-aware live migration problem would be empirically investigated. The first two stages of the DSR framework have been completed by now which are the *Problem Identification and Motivation* and *Defining the objectives for a solution.*

The next stage of the DSR framework is to *Design an artifact* where the workload-aware live migration algorithm would embedded. Experiments would then be conducted with the Workload Aware Live Migration (WALM) artifact to demonstrate its functionality.

The final two stages are *Evaluation* and *Communication.* The WALM artifact can be evaluated with respect to the baselines models (vanilla pre-copy, post-copy and hybrid migration) by comparing Total Migration Time (TMT) and Downtime (DT). A detailed description of the evaluation phase is given in Section 4. Finally, the findings would be published to the community.

Figure 4 depicts the steps of the proposed research approach. Steps and tasks are depicted on the left and the output of each stage is depicted on the right.

3.2 Research Design

Figure 5 shows the proposed architecture of WALM. WALM contains a *Resource Tracker* which monitors the VM's resource usage prior to its migration. As shown in Figure 6, the resource tracking module monitors CPU usage, memory usage, RAM and Network usage of a VM. During this period of time, the VM's workload is assumed to be static. That is, the workload could have a dynamic nature, but the nature of the workload just before the migration is taken and considered when choosing the most optimal migration method.

The resource tracker captures the resource usage of VM can be captured as follows.

CPU Usage: The Linux command *top* can be used to measure the CPU usage of the VM's workload prior to the migration. This utility can be used to capture the CPU usage of all processes for a period of time and get the average.

Memory Usage/Page-dirtying rate: The page dirtying rate can be calculated using the *Dirty Bitmap Mode* available in QEMU. The guest VM's memory is

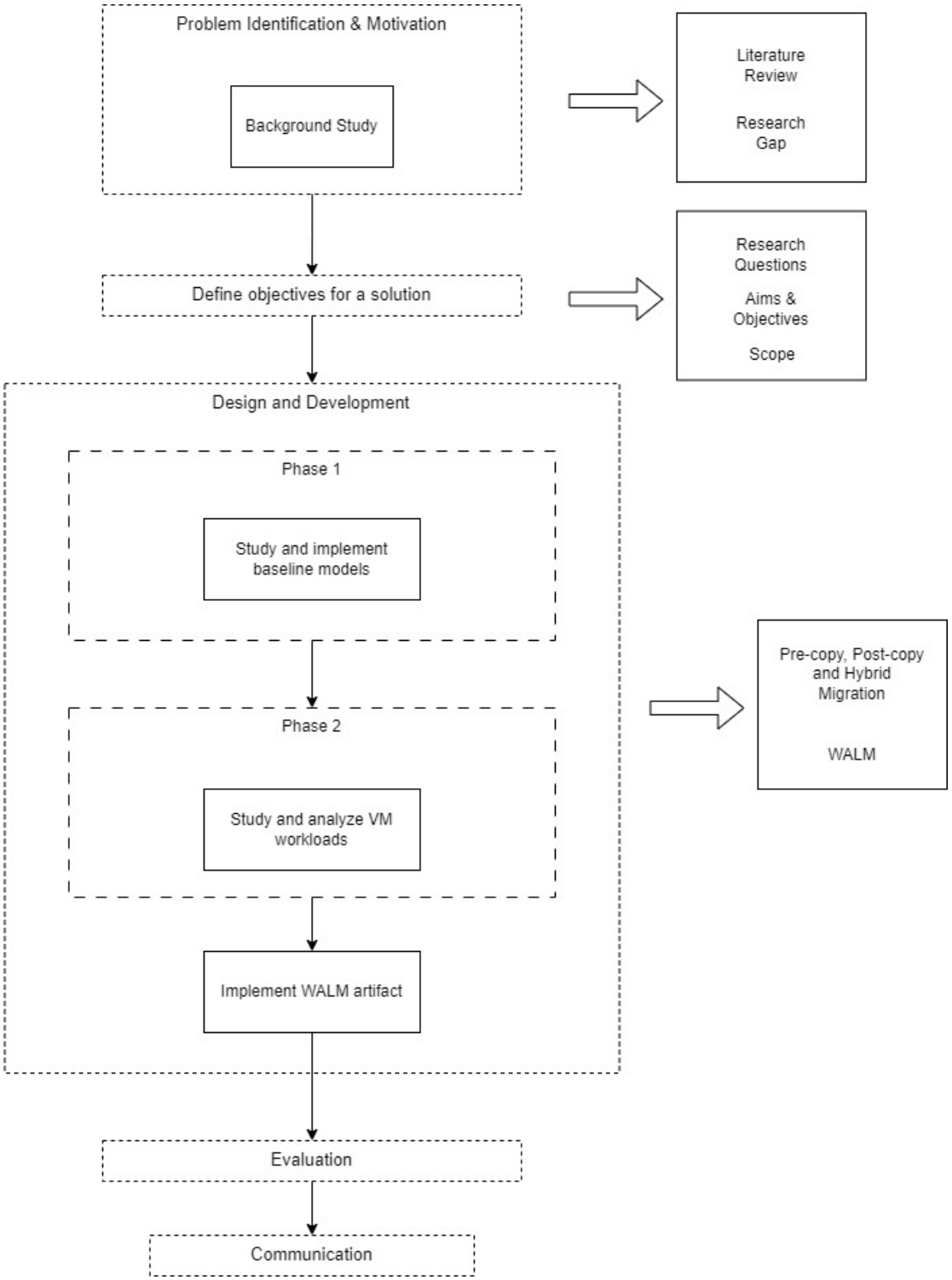

Figure 4: Research Approach

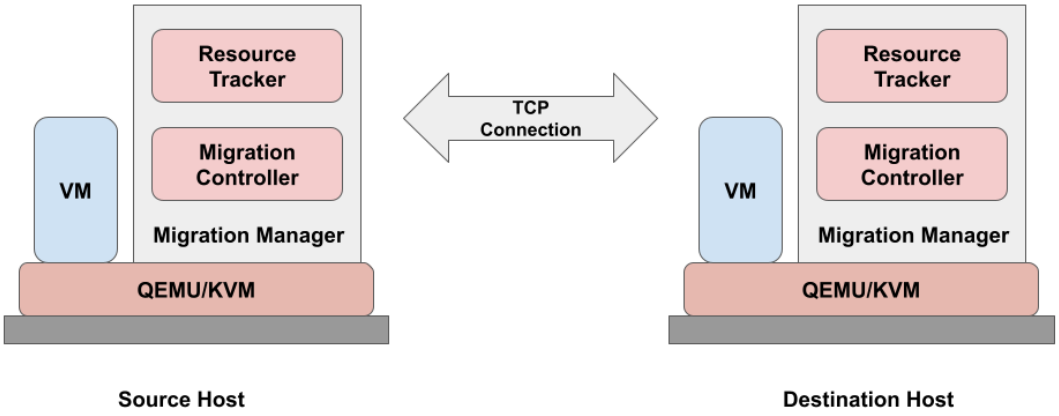

Figure 5: The Architecture of WALM

tracked using a bitmap and it can be queried to calculate and return the dirty-rate of the VM.

The memory usage of a VM can also be captured using the *top* utility in Linux. The average memory usage of all processes running within the VM can be computed here.

RAM size: The RAM size of a running VM can be queried using the QEMU Machine Protocol (QMP) commands.

Network Usage: The network usage of a VM can be measured by taking the total packets sent and received by the tap interface of the VM using the *ifconfig* Linux tool.

When the migration command is given, resource usage is fetched from the resource tracker module to the *Migration Controller* (Figure 7). The migration controller then decides on an optimal migration method based on the evaluation of the resource usage of the VM.

Migration Manager encompasses both the resource tracker and migration controller modules and is considered as the WALM artifact.

The source host and the destination host are both connected via a TCP connection and both runs on top of QEMU/KVM virtualization software.

3.3 Preliminary Results and Discussion

Two machines were set as source and destination hosts bearing the following specifications.

- Product: HP Z620 Workstation
- CPU: Intel(R) Xeon(R) CPU E5-1650 v2 @ 3.50GHz x 12 Cores

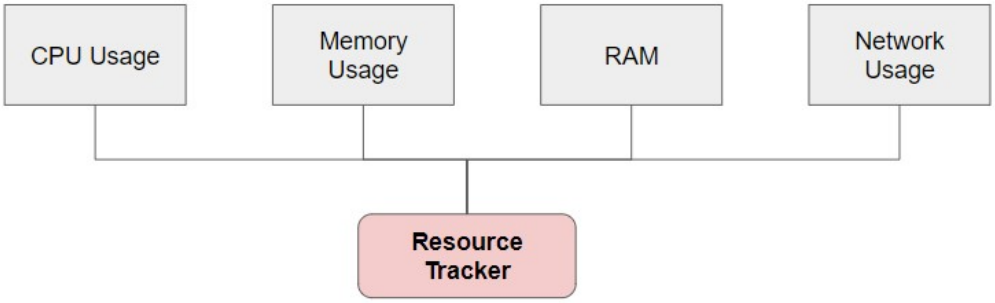

Figure 6: Resource Tracker

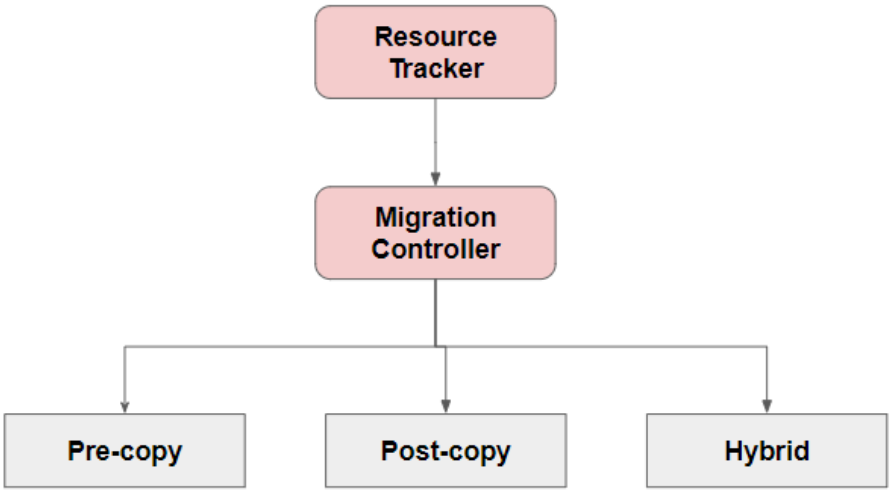

Figure 7: Migration Controller

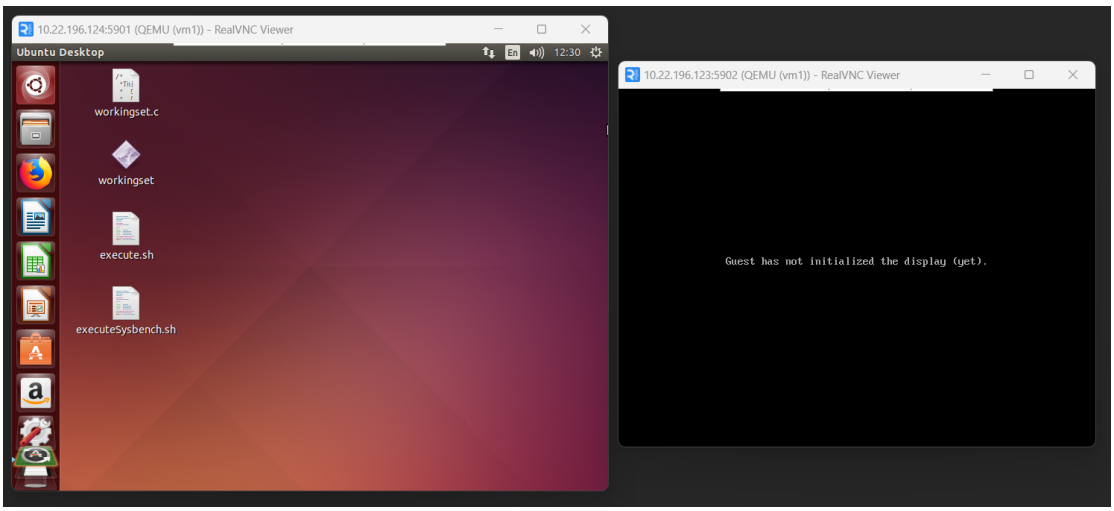

Figure 8: VM running on the source host prior to migration (captured by RealVNC Viewer)

- Memory: 16GB
- Network: Gigabit Ethernet Switch

Using an NFS server, a directory (of the destination) has been shared between the source and destination to be used to store VM hard disks which would not be migrated in LAN migrations. This is explained in Section 4.1.

3.3.1 Baseline Models

Baseline models were studied and implemented using QEMU Emulator version 2.5.0 which was installed in both source and destination hosts.

3.3.1.1 Pre-copy Migration

Pre-copy migration is provided by default in QEMU. To migrate a VM using pre-copy, QEMU loads and saves the state of the guest VM. Hence to do this, we need to run the same command with the same arguments twice in QEMU in the two hosts (source and destination). As shown in Figure 8, the VM runs on the source while the destination waits for the incoming migration stream.

The migration stream is a byte stream transported over a TCP connection.

Migration commands are executed through QMP. Figure 9 shows the scripts used for pre-copy migration.

Once the command is executed, QEMU starts its pre-copy iterations (Figure 10). Once the iterations converge, you can see the VM start working on the destination server (Figure 11).

3.3.1.2 Hybrid Migration

Hybrid migration is also provided by QEMU by default. This feature is provided by QEMU specifically for VMs running workloads which have larger page-dirtying

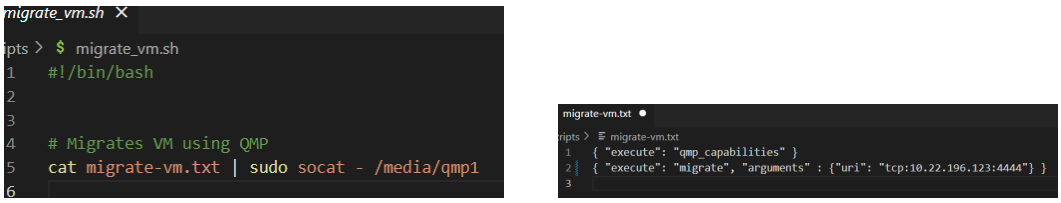

Figure 9: Pre-copy migration command

```
qmp migrate uri tcp:10.22.196.123:4444 has_blk 0 blk 0 has_inc 0 inc 0
 tcp_start_outgoing_migration
tcp_wait_for_connect
qemu_fopen_ops
migrate_fd_connect
migration_thread
Iter 1, pending size(in pages) 266450
Qemu_savevm_State_iterate :: qemu_savevm_state_iterate
iterations 1 pages 266450 time_elapsed 11416
iterations 1 down_pages 0 downtime 0
Iter 2, pending size(in pages) 7052
Qemu_savevm_State_iterate :: qemu_savevm_state_iterate
iterations 2 pages 7052 time_elapsed 221
iterations 2 down_pages 0 downtime 0
Iter 3, pending size(in pages) 3202
migration completion about to call
In migration completion
Migration status active in completion
VM force stop fr post-copy
VM force stop fr pre-copy
qemu_savevm_state_complete_precopy
 ram_save_complete
before migration status completeiterations 3 down_pages 3202 downtime 112 vm_stop 0
New d_time 0.111605 total_time 11.767206
qemu d_time 112 qemu total_time 11767 total_trans_pages 273502
new total_time 11751
PP : prev total MT 112 org TMT 11767
!Entered Postcopy
prev downtime 112 now downtime 112
```

Figure 10: Pre-copy Iterations

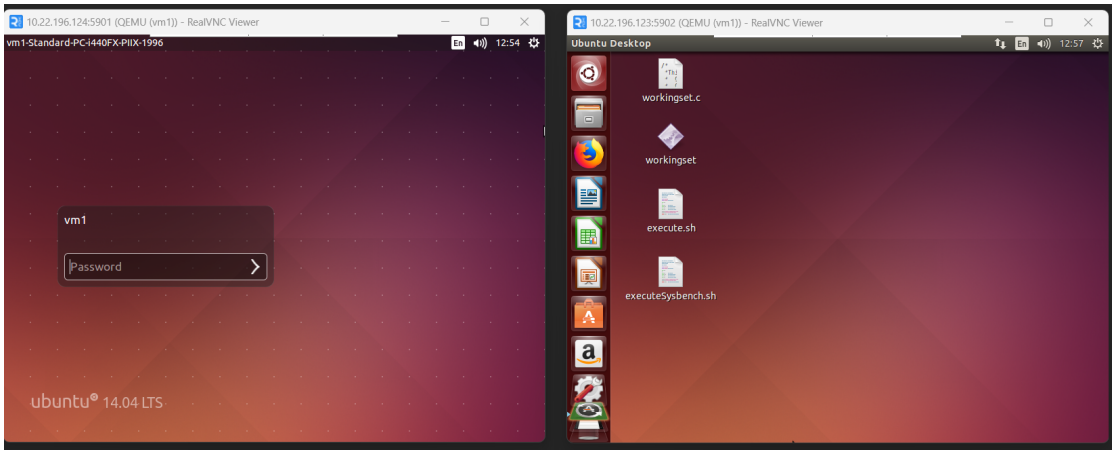

Figure 11: VM running on the destination host after migration (captured by RealVNC Viewer)

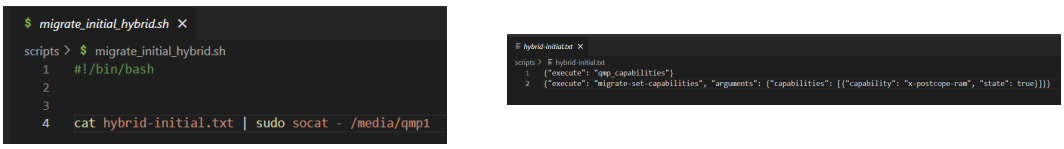

Figure 12: Enable post-copy migration

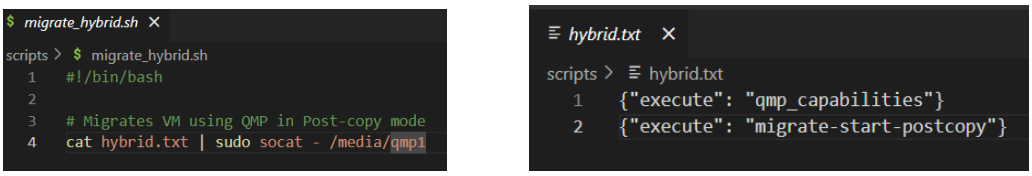

Figure 13: Switching to post-copy migration

rates (with respect to the available bandwidth) so that migration could be completed in finite time.

To perform hybrid migration, post-copy mode must be enabled before migration (Figure 12). Once that is done, migration is started with the pre-copy mode (Figure 9) and after a fixed number of iterations we switch to the post-copy mode (Figure 13).

3.3.1.3 Post-copy Migration

Vanilla post-copy migration is enabled in QEMU by stopping the pre-copy mode at iteration 1 and switching to post-copy mode from there onwards. This modified post-copy migration is identified by the transport mode 'pp' in the QMP command as shown in Figure 14.

3.3.2 Workload Analysis

Synthetic workloads were used for the initial experiments. The experiments would be continued with real-world VM workloads. The experiments were done to study and gather data on how each type of workload performs under different migration schemes.

a. Memory-intensive Workloads

One of the workloads chosen was the *Working set* program where random numbers are written to memory. The size of the working set can be specified when running the program. This program was chosen because it is memory-intensive and the size of the working set can be varied accordingly.

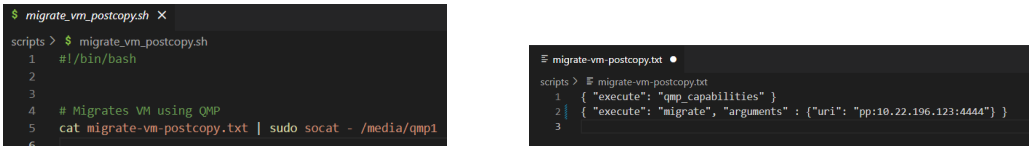

Figure 14: Post-copy migration command

b. CPU-intensive Workloads

Sysbench was also used for CPU benchmarking. Specifically, the tool provides a CPU test to calculate prime numbers up-to a certain threshold. As a result, the memory usage of the VM is zero while the CPU usage is on average around 89%. This test was run in the VM while it was migrated.

Workloads such as *iPerf* or *Netperf* are also CPU intensive. More experiments would be conducted using such workloads to study their performance.

c. Network-intensive Workloads

To study how network-intensive workloads perform under different types of migration, experiments would be continued using such as *iPerf* or *Netperf*.

3.3.2.1 Pre-copy Experiments

The *Working Set* program was run on VMs and migrated using pre-copy method by varying the size of the working set. Readings were taken by performing over 10 migrations for each experiment and taking an average of the TMT and DT along with the CPU Usage of the VM.

Figure 15 depicts the average TMT and DT for varying working sets. The VM was given an 8GB RAM for these experiments. The working set of the workloads were increased from 50% of the RAM upto 90%. The resulting graph shows that the TMT and DT increases with the size of the VM's working set. The working set program also consumes CPU resources. Figure 16 shows the average CPU usage for each of the experiments. The VMs were given a single CPU core. As shown in the graph, CPU consumption percentage increases with the size of the working set.

To study how a computationally intensive workload would perform under pre-copy migration, *Sysbench*'s CPU test (calculating prime numbers upto 5000000) was executed in the VMs during migration. The average CPU usage was approximately 89% of the 1 CPU core allocated to the VMs. In these experiments, the RAM allocated to the VMs were varied to analyze how performance is affected by the VM's RAM size.

As depicted by figure 17, the TMT and DT decreases with the RAM size allocated for the VM.

3.3.2.2 Hybrid Experiments

Sysbench's CPU test was run on VMs while undergoing hybrid migration (without fixed no. of iterations). The VMs were allocated 1 CPU core. In the experiments, as with pre-copy experiments, the RAM size was varied to study how TMT and DT is affected by the VM's RAM size.

Figure 18 depicts the results of the experiment. Note that there's a significant decrease in the DT of hybrid migration compared with pre-copy migration.

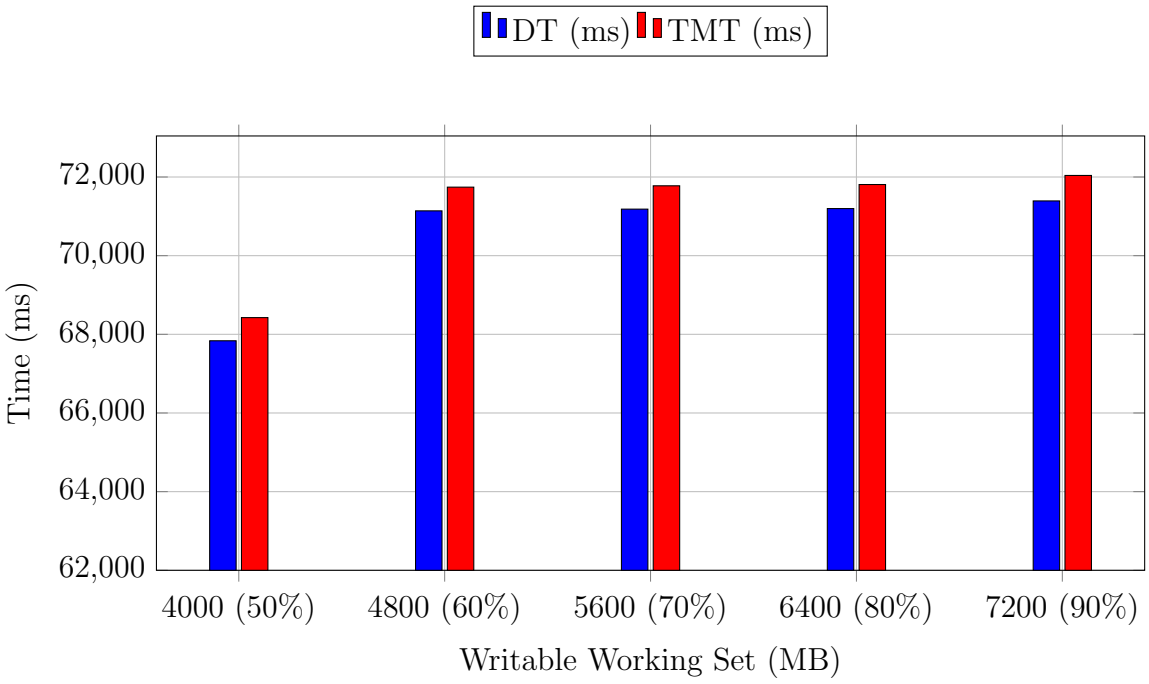

Figure 15: Average TMT and DT for varying Working Set migrated using pre-copy migration

Figure 16: Average CPU Usage for varying Working Set migrated using pre-copy migration

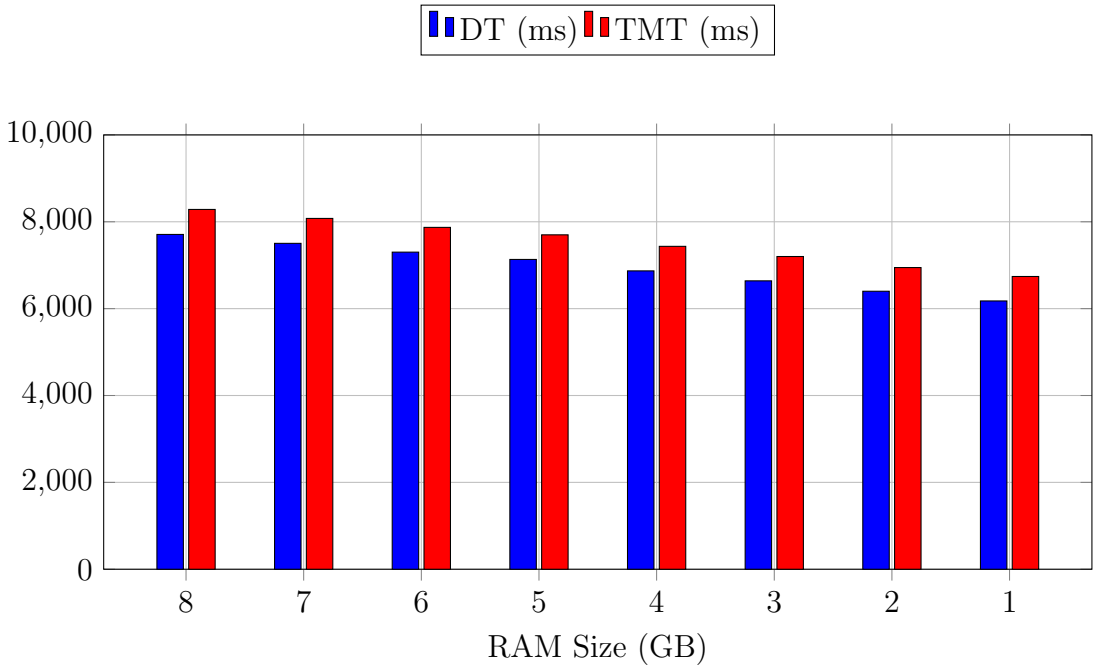

Figure 17: Average TMT and DT for varying RAM size of Sysbench CPU benchmark migrated using pre-copy migration

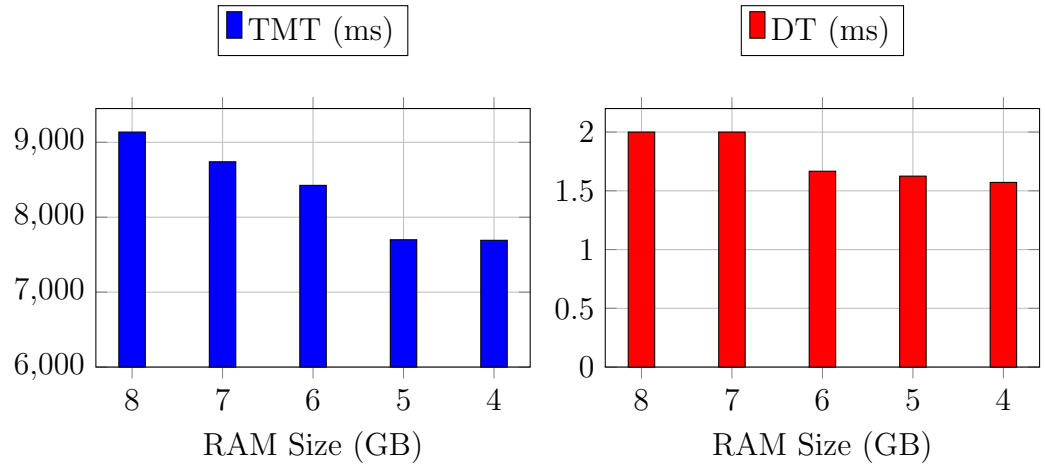

Figure 18: Average TMT and DT for varying RAM size of Sysbench CPU benchmark migrated using hybrid migration

3.3.3 Discussion

From analyzing the results of the experiments above, it is clear that when workloads get more memory-intensive, the TMT and DT increases accordingly (Figure 15) in pre-copy migration.

Comparing the TMTs and DTs of migration of the Working Set workload with the migration of Sysbench workload (in pre-copy), it is clear that CPU intensive workloads perform much well under pre-copy migration as opposed to memory-intensive workloads.

Considering the nature of network-intensive workloads, if the workload traffic is outgoing, it might interfere with the migration traffic whereas if the workload traffic is incoming, it might not interfere with the migration traffic. These points would be considered in the experiments that would be conducted using network-intensive workloads.

For post-copy migration, workloads with higher rate of memory-reads would not perform well since they would end up with a large number of page faults. Hence workloads with a lower rate of memory-reads would perform better.

Considering hybrid migration, since the entire memory footprint is sent in the first iteration of the pre-copy round, it is expected that there would be lesser number of page faults in read-intensive workloads since the page to be read can be present on the destination. Therefore, the performance degradation of workloads with higher rate of memory-reads would be diminished when hybrid migration is adapted.

4 Section 4

4.1 Research Tools Used

4.1.1 QEMU

QEMU is a generic open source emulator and virtualizer used in this research for all virtualization. It is integrated with the KVM hypervisor within the two Linux servers in the environment. QEMU has many in-built features which would be very useful and by default comes with pre-copy migration and hybrid migration (without fixed number of iterations).

All experiments in this research would be done using QEMU Emulator version 2.5.0.

4.1.2 Sysbench

Sysbench is used as a benchmarking tool for Workload analysis. The VM to be migrated runs a Sysbench test (CPU/Memory) while it is being migrated in any one of migration methods, to analyse the performance.

4.1.3 RealVNC Viewer

This tool is used to access the VMs running on the source and destination remotely as captured by Figure 8. This can be used to easily access the VMs from a personal computer.

4.1.4 NFS

Using NFS, a directory is shared between the two servers to store the hard disks of the VMs created. This is so that both the source and destination servers can access the VMs' disks since in LAN migration, the disk is not migrated. The destination server runs an NFS server and exposes a directory to the source server. The source server acts as the NFS client by accessing the shared storage over the network.

4.2 Evaluation

The evaluation of this research consists of the following steps.

4.2.1 Experimental Setup

4.2.1.1 Environment

All experiments would be executed in servers equipped with Intel(R) Xeon(R) CPU E5-1650 v2 @ 3.50GHz x 12 with 16GiB of RAM and running Ubuntu 20.04 LTS as the host OS. The servers are connected via HPE OfficeConnect 1920S Series Switch (JL385A). The servers run QEMU Emulator version 2.5.0 for virtualization.

4.2.1.2 Workload Generation

Workloads would encompass both synthetic and real-world VM workloads that mimic real-world scenarios and are diverse in nature in terms of CPU usage, page-dirtying rate etc.

4.2.2 Experimental Procedure

4.2.2.1 Randomized Trials with Baseline Models

Experiments would be done using vanilla pre-copy, post-copy and hybrid migration methods without any workload awareness to establish a baseline for evaluation.

For each type of workload, baselines models would be randomized and the VMs would be migrated accordingly to monitor and record the TMT and DT of the migration process.

Experiments would be repeated to ensure the reproducability of the results obtained.

4.2.2.2 Workload-Aware Live Migration (WALM)

Workload-Aware Live Migration would be implemented and integrated into QEMU virtualization environment.

4.2.3 Data Analysis

The following metrics would be considered in the data analysis phase.

Total Migration Time (TMT) - Measuring the time taken for the entire migration process using baseline models and using WALM.

Downtime (DT) - Measure the time in which the VM remains unavailable during migration using baseline models and using WALM.

In all experiments, statistical methods would be used to measure and analyze the TMT and DT of the migration process. Any performance improvements or degradations would be calculated by comparing WALM to migration using randomized baseline models (results of the randomized trials VS WALM).

4.3 Project Timeline

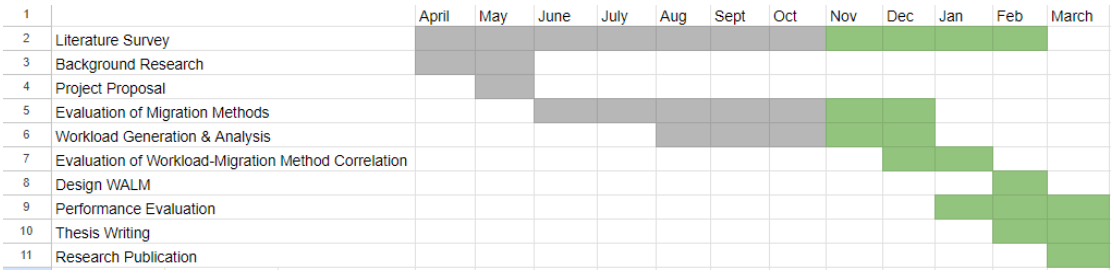

Figure 19: Project Timeline

4.4 Improvements from Feedback

Feedback	Improvement
1. Consider the nature of dynamic workloads.	• Using WALM, nature of the workload would be measured and analysed periodically. When a migration command is given, detecting the nature of the workload is stopped and the most recent reading is used to consider the most optimal migration method.
2. Evaluation plan is not detailed enough.	• A detailed evaluation plan is mentioned under Section 4.2.

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.

Ala'Anzy, M. & Othman, M. (2019), 'Load balancing and server consolidation in cloud computing environments: a meta-study', *IEEE Access* **7**, 141868–141887.

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.

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.

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.

Deshpande, U., You, Y., Chan, D., Bila, N. & Gopalan, K. (2014), Fast server deprovisioning through scatter-gather live migration of virtual machines, *in* '2014 IEEE 7th International Conference on Cloud Computing', IEEE, pp. 376–383.

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.

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

Hines, M. R. & Gopalan, K. (2009), Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning, *in* 'Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments', pp. 51–60.

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.

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.

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

Li, C., Feng, D., Hua, Y. & Qin, L. (2019), 'Efficient live virtual machine migration for memory write-intensive workloads', *Future Generation Computer Systems* **95**, 126–139.

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.

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.

Miller, R. (2008), 'Failure rates in google data centers'.
URL: *https://www.datacenterknowledge.com/archives/2008/05/30/failure-rates-in-google-data-centers*

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

Nelson, M., Lim, B.-H., Hutchins, G. et al. (2005), Fast transparent migration for virtual machines., *in* 'USENIX Annual technical conference, general track', pp. 391–394.

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.

QEMU (n.d.).
URL: *https://www.qemu.org/*

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.

Sahni, S. & Varma, V. (2012), A hybrid approach to live migration of virtual machines, *in* '2012 IEEE international conference on cloud computing in emerging markets (CCEM)', IEEE, pp. 1–5.

Shah, S. A. R., Jaikar, A. H. & Noh, S.-Y. (2015), A performance analysis of precopy, postcopy and hybrid live vm migration algorithms in scientific cloud computing environment, *in* '2015 international conference on high performance computing & simulation (HPCS)', IEEE, pp. 229–236.

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.

Svärd, P., Hudzia, B., Walsh, S., Tordsson, J. & Elmroth, E. (2014), 'The noble art of live vm migration-principles and performance of pre copy, post copy and hybrid migration of demanding workloads', *Technical report, 2014. Tech Report UMINF 14.10. Submitted* .

Vom Brocke, J., Hevner, A. & Maedche, A. (2020), 'Introduction to design science research', *Design science research. Cases* pp. 1–13.

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.

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.