# Enhancing Convergence of Live VM Migration with Dynamic Workloads

Research Proposal
SCS 4224: Final Year Project

Nadeesha Nethmini Epa
Student of University of Colombo School of Computing
Email: 2020cs049@stu.ucsc.cmb.ac.lk

May 3, 2025

Supervisor: Dr. Dinuni K Fernando

# 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 :** D.N.N. Epa
**Registration Number :** 2020/CS/049
**Index Number :** 20000499

---

**Signature & Date**

This is to certify that this project proposal is based on the work of Ms.D.N.N.Epa under my supervision. The project proposal has been prepared according to the format stipulated and is of an acceptable standard.

**Supervisor Name :** Dr. D.K.Fernando

---

**Signature & Date**

# Contents

# List of Figures



# List of Tables

# Acronyms

**CC** Cloud Computing. 1

**CRM** Cloud Resource Manager. 8

**ML** Machine Learning. 6, 7

**OS** Operating Systems. 11, 13

**RTM** Real Time Monitoring. 6

**VM** Virtual Machine. ii, 1–4, 8–11, 13, 14

**VMs** Virtual Machines. 2, 8

**WWS** Writable working set. 3

# 1 Introduction

As the technology grows cloud computing(CC) has become a must for developing reliable and efficient applications. Virtualization is a concept that comes with the use of cloud computing. With the use of virtualization, users were able to use the functionality of multiple hardware resources from one physical machine. As the name suggests Virtualization (Xing & Zhan 2012, Malhotra et al. 2014) defines the process of generating a virtual representation of actual hardware which allows users to place multiple hardware resources inside one physical machine. This virtual representation is known as a Virtual Machine (VM).

VM migration is one of the most important concets used in CC. It defines the idea of transferring a VM running on one resource(Source) to another resource (Destination). Since the failures of a server can happen at any time migration is essential for low-level system maintenance, load balancing, and fault management Choudhary et al. (2017). One of the key terms that need to be considered before the VM migration is identifying the convergence point to complete the migration process. According to the nature of the workload, the convergence point will be changed. For a static workload, it is very easy to identify the convergence point. Because when migrating a static workload there is a point where the same set of pages will get modified. In this instance, the set of pages modified is known as the working set. This instance is identified as the convergence point. However, if the workload is dynamic then additional effort has to be put to handle the dynamic nature because the working set can be changed at any time. Machine learning-based methods and statistical techniques can be defined as two(2) main types of techniques used as solutions to this unpredicted behavior of a dynamic workload.

## 1.1 Motivation

With the improvements of the technology CC has become an important feature for almost all users around the world. The reason behind this is that CC provides facilities for users to create and customize their application needs using the internet. This special feature provides cost-effectiveness and flexibility for users to access their applications.Cloud(GCP 2024), Amazon Web Services (AWS)(AWS 2024), Microsoft Azure(Microsoft 2024), and Oracle(Oracle 2024) are among the top ten(10) cloud service providers. These cloud service providers use VMs to provide scalable and on-demand computing resources to users.

Since a failure of a server can happen at any time there should be a proper way to handle those failures. So, the VM migration was introduced for this process. Over several years, different mechanisms have been implemented to migrate the VMs from one server to another to avoid interruptions during the process due to server failures. At present, most of the applications consist of dynamic workloads. Because of the variability and complexity, dynamic workloads have many challenges (Zhang & Boutaba 2014, Hossain & Song 2016, Cerotti et al. 2012) which makes it difficult to migrate a dynamic workload. One of the main challenges identified

here is that the difficulty in predicting the future workload leads to difficulties in identifying the best migration technology to be used. Because of this, there may be a waste of resources. Hence it is crucial to enhance the performance of live migration techniques in dynamic workloads to perform better.

## 1.2 Background

### 1.2.1 Workload Categorization

As discussed earlier VMs needs to be migrated to avoid the interruptions that occur due to the failures of server nodes. However, the techniques that are used for the migration process depend on the nature of the workload. Workloads can be mainly categorized into two sections static and dynamic . Static workload consists of a constant amount of computing resources over a long period. Because of this nature, it is easy to predict the behavior of the workload by considering the current workload and starting the migration process. While the dynamic workload changes over time it is hard to predict its behavior by just considering the current workload. So different techniques need to handle a dynamic workload. Profiling (Ye et al. 2014, Wu & Wolf 2008, Han et al. 2020) and Machine learning (Wei et al. 2018, Naik 2022) techniques can be introduced as the two main types of techniques used to handle a dynamic workload. More details about these two techniques will be discussed in the section 2.1

### 1.2.2 Live VM Migration

As the name suggests Live VM Migration is responsible for maintaining the failures that occurred during the middle of the execution of an application. In the process of Live VM Migration the VM will be sent to another server which is known as the destination without interrupting the applications already running inside the failed(source) VM. Christopher (2005) has shown several advantages of live migration as

- In live migration the entire operation system and all of its applications are moved at once. This will avoid many challenges encountered by process-level migration methodologies. (Christopher 2005)
- After the migration the host machine does not depend on the source and this solves the problem of residual dependencies.
- Live VM migration has the ability to complete the migration process without providing an interruption to the users.

Live VM migration can be categorized broadly into three(3) main techniques as **Pre-Copy migration, Post-Copy migration, and Hybrid migration**(He 2021, Christopher 2005, Fernando et al. 2020, Hines et al. 2009).

1. **Pre Copy Migration**
   In Pre-Copy(Christopher 2005, Fernando et al. 2020)migration, the memory

transferring will happen via three(3) main steps as **push phase, stop and copy phase, and the pull phase.**

(a) Push phase
During the push phase memory pages will be sent to the destination while the source VM continues its execution. This process executes continuously for several iterations. On the first iteration, each stored memory page will move from the source to the destination. In the second iteration, only the modified memory pages compared to the first iteration will be transferred. Likewise, for the next iteration, only the modified pages compared to the previous iteration will be transferred. If the workload is **static** then the required transferring page count for the coming iterations will become low compared to the first iterations. Then at some point, there is an instance that the same set of pages will become dirty for several iterations. This set of dirty pages will be identified as the **writable working set(WWS)** . For the static workload migration, this point will be considered as the convergence point.
If the workload is **dynamic** then there is no guarantee that there is a point where after that point the dirty page count will be reduced. In there the working set will change repeatedly during the migration. Hence it is really hard to identify a convergence point. So, migrating a dynamic workload needs special types of mechanisms such as profiling (Ye et al. 2014) and machine learning-based methods(Wei et al. 2018). More details about these techniques will be discussed in section 2.1.

(b) Stop and copy phase
After identifying the convergence point the source VM holds its execution and quickly sends the remaining pages, CPU state, and the I/O state to the destination VM. From this point onwards the destination VM starts to be executed. The duration that source VM paused its execution and started resuming in the destination is known as the **downtime** of the VM migration.

(c) Pull phase
While the VM executes in the destination it may notice that some missing pages have not been successfully transferred to the destination. This instance is known as a **page fault**. In this case, the source will re-send the missing pages to the destination.

2. **Post-Copy Migration**
The execution nature of the Pre-Copy technique is used to send the dirty pages to the destination through multiple iterations. But in Post-Copy (Hines et al. 2009, Fernando et al. 2020) migration it uses a mechanism that ensures no duplicates of the same page will be sent to the destination. The steps of this technique are as follows.

(a) First, the VM at the source node paused its execution.Then, a minimal

set of memory pages required for the execution of the destination VM will be moved to the destination.

(b) Then the destination VM starts its execution and the source quickly starts sending the remaining memory pages to the destination. This process is known as **pre-paging**.

(c) Same as the pull-phase in the Pre-Copy if the destination VM may notice that some pages have not been successfully sent to the destination a page fault will occur and the source quickly starts sending the missing pages to the destination. This is known as the **Demand paging**

3. **Hybrid Migration**
Hybrid migration(Sahni & Varma 2012, Altahat et al. 2020) can be known as a combination of both Pre-Copy and Post-Copy techniques. Hybrid migration aims to increase the performance for both read-intensive and write-intensive workloads. The algorithm consists of one(1) Pre-Copy round and the rest of the algorithm consists of the Post-Copy technique. Hybrid algorithm stages can be described as given below.

(a) As the first step the Pre-Copy round executes. This step is used to make a copy of the memory of the VM. During the execution of this first round, the source VM keeps running.

(b) Then the source VM stopped and the CPU state of the source VM will transfer to the destination host.

(c) As the final step the Post-Copy algorithm starts the execution and the remaining dirty pages will copy from the source to the destination. If any page fault occurs the page will be copied to the destination from the source (Demand Paging).

# 2 Literature Review

A literature review was conducted for the topic **Live VM Migration Convergences in a Dynamic Workload** and the report consists of a detailed explanation of live migration techniques, an explanation about the dynamic workload, and the different types of dynamic workload handling methods with their advantages and limitations.

## 2.1 Dynamic workload

A dynamic workload (Hossain & Song 2016, Cerotti et al. 2012) consists of several computational tasks that change the behavior of tasks very frequently over time. Because of its changing behavior, it is hard to predict workload patterns for dynamic workloads. This makes dynamic workload handling a challenging and complex task. Some of the key challenges of dynamic workloads are given below.

1. Dynamic workloads consist of different types of applications that consist of their own and different kinds of resource requirements which is known as the **Heterogeneity** (Zhang & Boutaba 2014).
2. Dynamic workloads have the problem of over-provisioning and under-provisioning. Because of the dynamic nature, it is hard to predict the resource demand hence most of the applications provide resources more than enough to execute the operations. This may lead to a waste of resources. In similar cases, the resources provided may not be enough for the applications to execute which leads to under-provisioning and the failures in application process. This is known as the applications do not meet its **peak demands**. In both cases, part of or the whole cost used for the application process has become a waste (Hossain & Song 2016).

Dynamic workloads can be handled based on the following approaches:

- Mathematical/Machine Learning Techniques
- Statistical Techniques

To deal with dynamic workloads, these techniques require capturing and collecting records of the current behavior of the workload to predict future behaviors. For that process, a well-known technique called **profiling** has been used by several researchers(Wu & Wolf 2008, Ye et al. 2014, Han et al. 2020).

### 2.1.1 Profiling

Profiling is a concept used to capture the behavior of workloads based on their characteristics. There are two profiling techniques runtime profiling and offline profiling. Runtime profiling tries to capture the characteristics of an application when the application starts its execution. Because of that, runtime profiling (Wu & Wolf 2008) is used to deal with dynamic workloads. The captured data will be used to increase application efficiency. CPU usage, memory usage, and network consumption are some of the runtime profiling information.

The studies conducted by Wu & Wolf (2008),Ye et al. (2014), and Han et al. (2020) have used profiling techniques to identify the behavior of a workload. The main idea behind the concept used by those papers is to gather runtime profiling information such as CPU time, memory usage, and service time and understand the workload behavior using the collected information. The study followed by Wu & Wolf (2008) collects real-time information such as task service times, edge utilizations, and task utilizations and then uses a duplication and mapping algorithm that can capture the changes in the workload by using the collected profiling information. So the tasks with high computational demands will be duplicated across multiple cores by the task duplication algorithm. The task mapping algorithm will assign tasks to the processors to minimize the overhead. Also, the study of Ye et al. (2014) collects profile information for various types of workloads such as CPU-intensive, memory-intensive, and network-intensive. Then by analyzing these details, the characteristics and resource demands of different types of work-

loads will be identified. Finally, with the use of these details, they implemented two(2) models as a consolidation planning module and a migration planning module to complete the migration process with minimal effect on the performance. Han et al. (2020) has introduced a combination of profiling systems and refinement frameworks to identify micro-services placement with dynamic workloads.

One of the most important points highlighted by the above studies (Wu & Wolf 2008, Ye et al. 2014, Han et al. 2020) is profiling techniques has the ability to measure the system performance for various workload conditions(CPU intensive, memory intensive, network intensive, etc). So the workload changes can be identified in real time. Also profiling incurs low overhead because profiling can directly monitor specific metrics and uses simple analytical techniques which do not require high computational power.

### 2.1.2 Mathematical/Machine Learning Techniques

Several types of research have been conducted to solve the issue of handling a dynamic workload by using dynamic thresholds (Lin et al. 2011), reinforcement learning (Wei et al. 2018), use of autonomic computing (Sah & Joshi 2014), etc.

A study conducted by Lin et al. (2011) proposed a dynamic resource allocation schema that allocates resources according to the workload changes of the application. Since the paper deals with a dynamic workload the workload can be changed during the application lifetime. So by using a threshold value the optimal instance for the resource allocation will be calculated and then the resource allocation schema will be applied to allocate the resources.

Similar systems can be found in the solution provided by Sah & Joshi (2014), a solution based on autonomic computing. Autonomic computing refers to the term that applications can change themselves according to the changes of the application environment without any external help(Without notifying the user). The algorithm created using autonomic computing, consists of a capacity and utility agent (CUA) and a resource controller. CUA is used to collect data about the resources and the capacity of the VMs. According to the collected data, an initial schedule was implemented, and if the workload changes then the resource controller will change the schedule to allocate resources in a better way.

Enhanced Dynamic Johnson Sequencing is another technique proposed by Banerjee et al. (2023), that uses a combination of ML and RTM to handle the nature of a dynamic workload. It is also known as OptiDJS+. The algorithm provides a schedule based on the dynamic quantum value (minimum execution time + maximum execution time)/2 of the tasks.

Also the framework proposed by Wei et al. (2018) claims a workflow scheduling algorithm using a sequential decision-making approach based on reinforcement learning. The ML approach used in this proposed algorithm is called the Q-learning. This approach aims to find a suitable Infrastructure as a Service (IaaS) provider for a Software as a Service (SaaS) application.

As explained, several instances of using machine learning to handle a dynamic workload exist. However, there are a few important points to be considered when using ML techniques. To provide an accurate prediction, ML models need to be trained properly (Wei et al. 2018). This required a large amount of data. But in this instance acquiring data from a dynamic workload can be challenging since its workload behavior will change frequently(Khelghatdoust et al. 2016). Moreover, there can be situations like the model will overfit (Wei et al. 2018) and this will lead to inaccurate predictions. Also implementing a proper model can be resource-intensive because it requires a high computational power to train and run complex models.

### 2.1.3 Statistical Techniques

Using statistical techniques to predict future workloads has become popular because of its simplicity, less data requirement, and transparency(Devi & Valli 2023, Calheiros et al. 2014). Most of the studies are based on the concept of time series and forecasting.

Calheiros et al. (2014) has conducted a study about workload prediction using **ARIMA** model. ARIMA stands for the Auto-Regressive Integrated Moving Average and it is used to predict the future workload. This paper has introduced an analyzer implemented using the ARIMA model. First, the data that needs to predict the future workload such as CPU usage, memory usage, etc will be fed into the ARIMA model. An important point considered here is these data must be stationary. For that process, a transformation method called differencing has been used. ARIMA model needs three(3) parameters to be fed the number of differences used, the auto-correlation value, and the partial auto-correlation value. The analyzer used in this paper will be responsible for updating the model with new data to increase the prediction accuracy of future workload behaviors. These predicted results have been used to ensure that enough resources are available for users to complete their tasks. With that Calheiros et al. (2014) has been able to increase the quality of service(QOS) by predicting future behavior using the ARIMA model.

The study conducted by Devi & Valli (2023) has used a combination of the ARIMA model and ANN(Artificial Neural Network) to predict the behavior of future workloads. So this hybrid model can be introduced as an enhanced version of Calheiros et al. (2014). ARIMA model was used to deal with the linear components and ANN was used to model the non-linear components. With that, the accuracy of the future workload prediction has been increased. Finally, the study has evaluated the performance of the new method using metrics such as mean absolute error and root mean squared error and the results have proven that the proposed solution works efficiently in a dynamic cloud environment to predict future workloads.

Ganapathi et al. (2010) has proposed a statistic-driven working model that can be used to predict the future behavior of a workload. Mainly the paper tries to predict the resource demands for applications that use MapReduce(Hashem

et al. 2016). Mapreduce tasks consist of several jobs which are known as Hadoop jobs. The proposed solution that named as **Kernel Canonical Correlation Analysis (KCCA)** will predict the execution times, and resource demands for the Handoop jobs. KCCA consist of a special feature called **Feature Vectors** which are constructed using job configuration parameters and input data characteristics. With the use of feature vectors KCCA was able provide accurate predictions about the future workloads.

Nowadays most of the application providers select statistical approaches to predict the future workload behaviors because of its simplicity and easy interpretation. Compared with the machine learning techniques, statistical techniques required smaller datasets (Calheiros et al. 2014, Devi & Valli 2023) and less computational power (Hashem et al. 2016) to train and run the models.

## 2.2 Existing Dynamic Workload Migration Strategies

Naik (2022) has claimed a solution for dynamic workload migration using a technique called **Adaptive Push-Pull**. This algorithm consists of a combination of a push function and a pull function. In the concept of Adaptive Push-Pull the push function is used to distribute the workload among the resources and when a VM is overloaded . The pull function is invoked when the VM is underloaded . For this process, a VM manager and the CRM are used. VM manager will be responsible for managing the workload within the same resource and CRM will be responsible for managing the workload around all the resources. So with the use of this method, the workload will be shared among the resources.

Also similar systems can be found in Lu et al. (2015), an optimal scheduling algorithm called **VHaul** to solve the problem of migrating multi-tier applications. These multi-tier applications consist of a group of co-related VMs that makes the application inherently dynamic. So to create the scheduling algorithm the VMs will be categorized according to their relationships. VMs belonging to the same application will be assigned to the same group. Then the production of resource utilization and migration time are used to decide the impact from the current VM to the next VM to be migrated. Then the migration schedule will be created as the smaller value for the production of resource utilization and migration time means less impact for the next VM to be migrated.

Khelghatdoust et al. (2016) came up with a solution **GLAP**, which is a combination of a gossip-based learning algorithm and Q-learning. GLAP uses continuous monitoring of the resource demands of the VMs to predict the behavior of future workloads.

The table 2.2 shows the comparison of the existing methods.

| | Pros | Cons |
|---|---|---|
| Adaptive Push-Pull | • Provides an adaptive push pull system to dynamically switch between push and pull functions to react to the changes in the workload.<br>• Solves the problem of overloading and underloading in the VMs. | • With a highly dynamic workload CRM and VMM will not be able to handle the problem of overloading. |
| vHaul | • Provide an efficient scheduling algorithm by considering the resource utilization and the migration time. | • Have to calculate the resource utilization and migration time whenever the workload changes. |
| GLAP | • Provides a threshold-free approach to detect future behaviors and migrate a workload without overloading. | • Have to use lot of computation power and resources to train a accurate model |

Table 1: Comaprison for dynamic workload migration techniques

# 3 Research Gap

Migration operation typically triggers due to sudden resource or power failures, maintenance, or loading issues. Difficulty in predicting the future behavior of a workload makes it a complex task to migrate a dynamic workload. Because in dynamic workload CPU, memory and storage will be changing rapidly which makes it difficult to identify the point to start the migration process and identify the correct destination. Also since the next behavior(CPU-intensive,memory-intensive,network-intensive) of the workload is unpredictable it can affect the migration process. The existing studies conducted in the area of dynamic workload migration have been focused on implementing complex mechanisms to dynamically change the schedule of tasks to be migrated according to the changes in the workload. Most of these solutions required information like resource utilization, migration time, and whether the VM is over-loaded or under-loaded. So it takes

some time to calculate and analyze the results before reacting to the changes in the dynamic workload.
Machine learning techniques (Khelghatdoust et al. 2016, Wei et al. 2018) and statistical techniques can be used to predict the future behavior of a workload before the migration. In machine learning techniques, the model must first be trained and then predict future behaviors by identifying patterns in the collected data. However, this process required a long period of training time, a huge set of data requirements, and a high computation power. Compared with the machine learning techniques, statistical techniques required smaller datasets (Calheiros et al. 2014, Devi & Valli 2023) and less computational power (Hashem et al. 2016) to train and run the models. The research conducted by Calheiros et al. (2014),Hashem et al. (2016), and Devi & Valli (2023) has proved that even with the less computation resources and less computation power these statistical techniques were able to predict future workloads accurately.
In the process of VM migration, the entire state of the failed VM should be transferred to the destination. This process required a significant amount of bandwidth. Hence the VM migration process is already identified as a resource-intensive operation. In this situation, if we use a technique that requires a lot of computational power to handle the dynamic behavior of the workload that can affect the performance of the application executed in the VMs negatively. Hence this research aims to address the gap of migrating a dynamic workload using a lightweight solution with enhanced performance impact. With the use of profiling details about CPU usage, memory usage, etc will be calculated and these results will be fed into a statistical model to predict the future behavior of the dynamic workload. Then according to the results, the migration will be scheduled. Furthermore, this research aims to reduce the total migration time and downtime with minimal performance impact on the applications run during the migration process.

# 4 Research Questions

1. How to find the convergence point of a dynamic workload in Live migration?
   This focuses on identifying the convergence point for the pre-copy migration using the data collected using profiling.
2. How can the performance of vanilla pre-copy be improved upon detection of convergence point in terms of total migration time, downtime, and application performance?
   This focuses on improving the performance of the applications by reducing the total migration time, and downtime with minimal impact on the performance.

# 5 Aims and Objectives

## 5.1 Aim

The main aim of the research is to develop and enhance a live migration technique for a dynamic workload that will improve efficiency, and minimize the downtime and total migration time of existing methods.

## 5.2 Objectives

The main objectives of the research are as follows:

1. Design and develop a method to analyze the profiling details and identify the peaks of dynamic workload patterns.
2. Identify the convergence point of a dynamic workload by analyzing the above results.
3. Evaluate the performance of migration using total migration time, downtime, and application performance metrics by incorporating industrial accepted benchmarks such as Sysbench, YCSB (Yahoo cloud serving benchmark), FIO, Memcached, etc.
4. Enhance the pre-copy algorithm to migrate a dynamic workload with minimal total migration time and downtime.

# 6 Scope

## 6.1 In Scope

- **VM live migration with dynamic workloads**: The research will focus on developing a strategy to migrate a dynamic workload using Live VM migration. Mainly focused on the Pre-Copy migration. However, there is a possibility to extend this to use for other Live migration techniques as well.
- **Single VM migration**: The research focused on migrating a dynamic workload in a single VM.
- **Implementing a working prototype**: At the end of the research a working prototype will be developed to find the convergence point and migrate a dynamic workload.

## 6.2 Out Scope

- **QEMU-KVM Hypervisor**: The research only used QEMU-KVM as the hypervisor for implementing and evaluating the working prototype.
- **Non-Ubuntu host OS**: The research will focus on Ubuntu Servers as the host OS.
- **Non-Linux guest OS**: The research will not cover the migration of VMs based on guest OSs such as Windows and macOS.Linux will be considered as the guest OS.

# 7 Preliminary Study

When migrating a dynamic workload it is essential to identify the future behavior of the workload to provide the best resource utilization and minimal downtime. So, for that process profiling can be used to detect the behavior of a workload. Profiling techniques can be used to, collect data about CPU usage, memory consumption, and network consumption of a workload. **mpstat, perf, and free -h** can be introduced as some of the popular profiling tools and their usage is as follows.

- **mpstat**- Monitor the CPU usage over time and provide insights about how the users used the CPU resources
- **perf** - Provide details about the CPU cycles, cache misses, events, and the percentage of time spent on a specific function.
- **free -h** - Monitor memory consumption over time and provide the percentage of memory used and available.

The figure 1 chart displays the data captured using the mpstat and free-h for a while. For the first 60 seconds, the CPU is in the Idle state, and in the period of 60-120 seconds a CPU-intensive workload is executed using the Sysbench benchmark. For the last 60 seconds, a memory-intensive workload is executed.

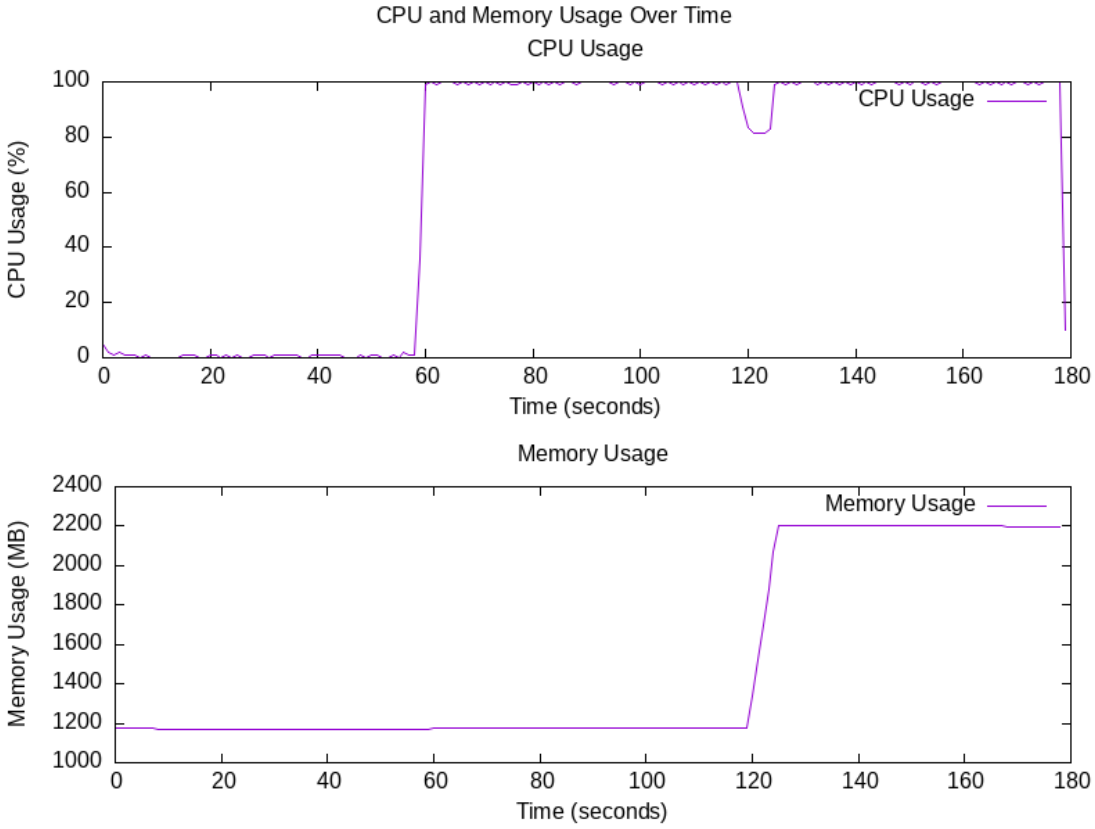
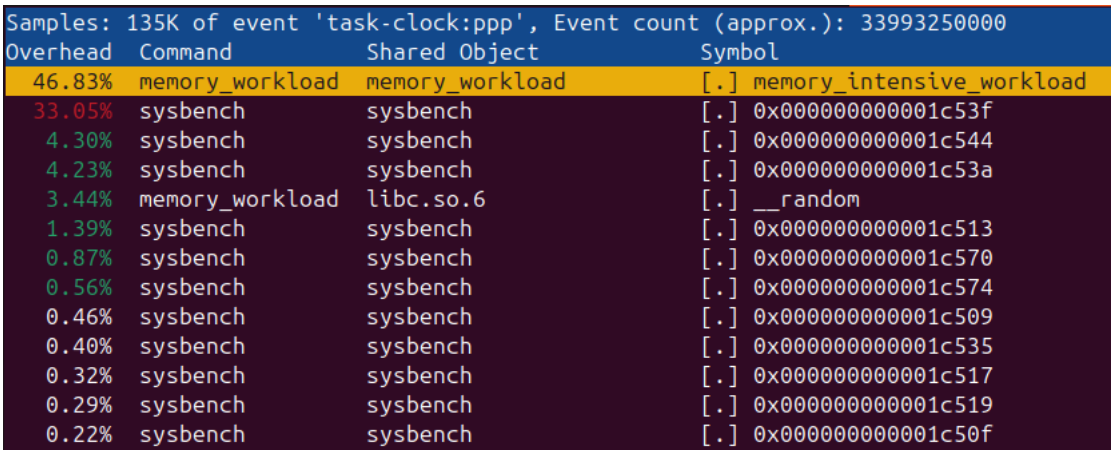



Figure 1: CPU and Memory Usage Over Time

In the same situation, the performance was captured using the Linux perf tool, and the results given are shown in Figure 2. According to the perf report, the

memory-intensive workload has been executed for 46.83% of the time from the whole process. The CPU-intensive workload is displayed as sysbench and it has taken 35.05% from the total time.



```
Samples: 135K of event 'task-clock:ppp', Event count (approx.): 33993250000
Overhead  Command          Shared Object            Symbol
  46.83%  memory_workload  memory_workload          [.] memory_intensive_workload
  33.05%  sysbench         sysbench                 [.] 0x000000000001c53f
   4.30%  sysbench         sysbench                 [.] 0x000000000001c544
   4.23%  sysbench         sysbench                 [.] 0x000000000001c53a
   3.44%  memory_workload  libc.so.6                [.] __random
   1.39%  sysbench         sysbench                 [.] 0x000000000001c513
   0.87%  sysbench         sysbench                 [.] 0x000000000001c570
   0.56%  sysbench         sysbench                 [.] 0x000000000001c574
   0.46%  sysbench         sysbench                 [.] 0x000000000001c509
   0.40%  sysbench         sysbench                 [.] 0x000000000001c535
   0.32%  sysbench         sysbench                 [.] 0x000000000001c517
   0.29%  sysbench         sysbench                 [.] 0x000000000001c519
   0.22%  sysbench         sysbench                 [.] 0x000000000001c50f
```

Figure 2: Performance Captured using Linux Perf Tool

So by monitoring these results the current workload patterns can be captured and these details can be used to predict the future behavior of a workload by sending these details as an input to a future workload prediction model.

# 8 Significance of the Research

This research project aims to enhance the existing live migration techniques for dynamic workloads by finding a new strategy to decide the convergence point of dynamic workload based on the data extracted and analyzed from profiling. This research contributes to the field by expanding the knowledge and understanding of effective live migration processes in virtualized environments. The research aims to optimize the live migration techniques to perform efficiently in the presence of a dynamic workload with minimal downtime and to improve the system performance by ensuring continuous availability of services. With the growing technology, most applications work with dynamic workloads. Hence the final outcome of the research will help to provide a better user experience and improve the service quality for the end users.

# 9 Planned Research Approach

**Data Collection**: As the first step of the research data that needs to predict the future behavior of a dynamic workload will be collected using profiling techniques.

**Test-bed setup**: The test-bed setup consists of two physical servers that are interconnected using a Gigabit Ethernet. QEMU/KVM is chosen as the hypervisor and a Ubuntu Server is used as the host OS. The VMs consist of a Linux-based OS. The following figure provides a high-level view of the test-bed setup.

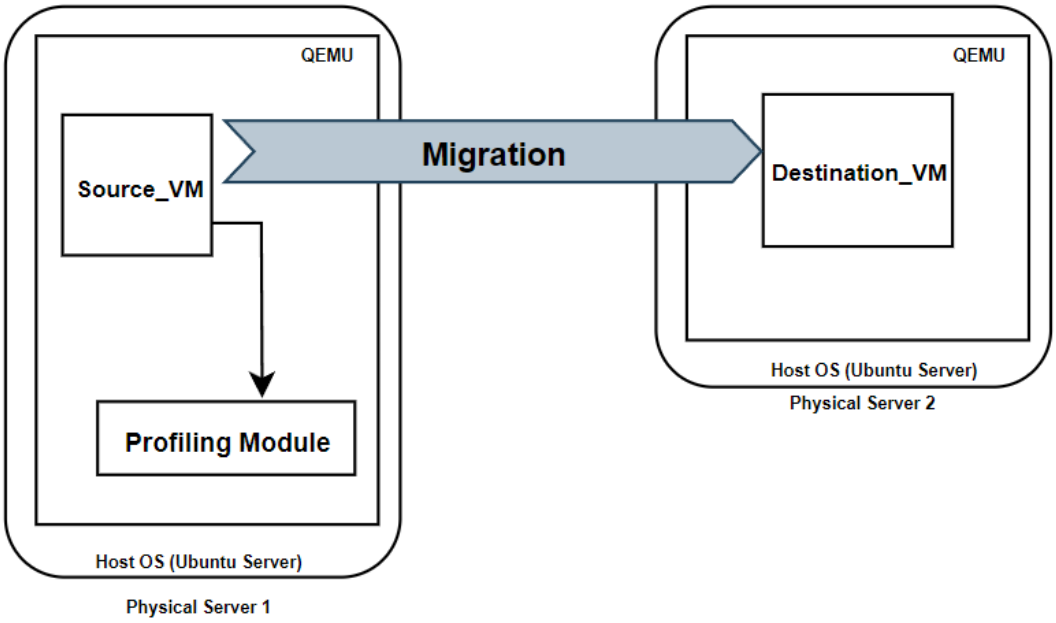



Figure 3: System Architecture

**Analyze the collected data**: The next step is to analyze the collected data and choose a suitable statistical technique to measure future behavior. There are several methods available such as **ARIMA, ETS models**, etc. So, finding the best statistical technique among these techniques and predicting the future behavior of the workload will be covered in this step.

**Design and development of the algorithm**: After identifying the future behavior of the workload an algorithm should be designed to identify the convergence point and start the migration point. The solution mainly focused on migrating a dynamic workload using the pre-copy migration strategy and it is going to be implemented for a single VM. However, the solution can be easily extended to gang migrations.

**Implement the working prototype**: With the use of the implemented algorithm a working prototype will be implemented for the KVM/QEMU platform.

**System Evaluation**: Finally the implemented prototype system will be evaluated through experiments conducted on the test-bed to assess its efficiency. The final result of the research aims to reduce the total migration time and downtime while reducing the performance impact for the applications executed during the migration process.

# 10 Project Timeline

The following chart displays the tentative timeline of the research.

| | June | July | August | September | October | November | December | January | February | March | April |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Literature Survey | | | | | | | | | | | |
| Background research | | | | | | | | | | | |
| Project proposal | | | | | | | | | | | |
| Collect data using profiling tool | | | | | | | | | | | |
| Identify the best statistical approach to predict the future workload | | | | | | | | | | | |
| Desing and implement the solution in KVM/QEMU platform | | | | | | | | | | | |
| Test and analyze the results for pre-copy migration | | | | | | | | | | | |
| Optimize the solution to be used for post - copy migration | | | | | | | | | | | |
| Overall performance evaluation | | | | | | | | | | | |
| Thesis writing | | | | | | | | | | | |
| Research publication | | | | | | | | | | | |

Figure 4: Project Timeline

# References


Altahat, M. A., Agarwal, A., Goel, N. & Kozlowski, J. (2020), 'Dynamic hybrid-copy live virtual machine migration: Analysis and comparison', *Procedia Computer Science* **171**, 1459–1468. Third International Conference on Computing and Network Communications (CoCoNet'19).
**URL:** `https://www.sciencedirect.com/science/article/pii/S1877050920311352`

AWS (2024), 'Amazon web services'.
**URL:** `https://aws.amazon.com/`

Banerjee, P., Roy, S., Modibbo, U. M., Pandey, S. K., Chaudhary, P., Sinha, A. & Singh, N. K. (2023), 'Optidjs+: A next-generation enhanced dynamic johnson sequencing algorithm for efficient resource scheduling in distributed overloading within cloud computing environment', *Electronics* **12**(19), 4123.

Calheiros, R. N., Masoumi, E., Ranjan, R. & Buyya, R. (2014), 'Workload prediction using arima model and its impact on cloud applications' qos', *IEEE transactions on cloud computing* **3**(4), 449–458.

Cerotti, D., Gribaudo, M., Piazzolla, P. & Serazzi, G. (2012), Flexible cpu provisioning in clouds: A new source of performance unpredictability, *in* '2012 Ninth International Conference on Quantitative Evaluation of Systems', IEEE, pp. 230–237.

Choudhary, A., Govil, M. C., Singh, G., Awasthi, L. K., Pilli, E. S. & Kapil, D. (2017), 'A critical survey of live virtual machine migration techniques', *Journal of Cloud Computing* **6**(1), 1–41.

Christopher, C. (2005), Live migration of virtual machines, *in* 'NSDI, 2005'.

Devi, K. L. & Valli, S. (2023), 'Time series-based workload prediction using the statistical hybrid model for the cloud environment', *Computing* **105**(2), 353–374.

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', pp. 1818–1827.

Ganapathi, A., Chen, Y., Fox, A., Katz, R. & Patterson, D. (2010), Statistics-driven workload modeling for the cloud, *in* '2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)', pp. 87–92.

GCP (2024), 'Google cloud platform'.
**URL:** `https://cloud.google.com/`

Han, J., Hong, Y. & Kim, J. (2020), 'Refining microservices placement employing workload profiling over multiple kubernetes clusters', *IEEE access* **8**, 192543–192556.

Hashem, I. A. T., Anuar, N. B., Gani, A., Yaqoob, I., Xia, F. & Khan, S. U. (2016), 'Mapreduce: Review and open challenges', *Scientometrics* **109**, 389–422.

He, T. (2021), 'Migration management in software-defined networking-enabled edge and cloud computing environments', *degree of Doctor of Philosophy, School of Computing and Information Systems, THE UNIVERSITY OF MELBOURNE, ORCID: 0000-0002-5472-7681* .

Hines, M. R., Deshpande, U. & Gopalan, K. (2009), 'Post-copy live migration of virtual machines', *SIGOPS Oper. Syst. Rev.* **43**(3), 14–26.
**URL:** `https://doi.org/10.1145/1618525.1618528`

Hossain, M. A. & Song, B. (2016), 'Efficient resource management for cloud-enabled video surveillance over next generation network', *Mobile Networks and Applications* **21**, 806–821.

Khelghatdoust, M., Gramoli, V. & Sun, D. (2016), Glap: Distributed dynamic workload consolidation through gossip-based learning, *in* '2016 IEEE International Conference on Cluster Computing (CLUSTER)', IEEE, pp. 80–89.

Lin, W., Wang, J. Z., Liang, C. & Qi, D. (2011), 'A threshold-based dynamic resource allocation scheme for cloud computing', *Procedia Engineering* **23**, 695–703.

Lu, H., Xu, C., Cheng, C., Kompella, R. & Xu, D. (2015), vhaul: Towards optimal scheduling of live multi-vm migration for multi-tier applications, *in* '2015 IEEE 8th International Conference on Cloud Computing', IEEE, pp. 453–460.

Malhotra, L., Agarwal, D., Jaiswal, A. et al. (2014), 'Virtualization in cloud computing', *J. Inform. Tech. Softw. Eng* **4**(2), 1–3.

Microsoft (2024), 'Microsoft azure'.
**URL:** `https://azure.microsoft.com/en-us`

Naik, K. J. (2022), 'An adaptive push-pull for disseminating dynamic workload and virtual machine live migration in cloud computing', *International Journal of Grid and High Performance Computing (IJGHPC)* **14**(1), 1–25.

Oracle (2024), 'Oracle — cloud applications and cloud platform'.
**URL:** `https://www.oracle.com/`

Sah, S. K. & Joshi, S. R. (2014), Scalability of efficient and dynamic workload distribution in autonomic cloud computing, *in* '2014 international conference on issues and challenges in intelligent computing techniques (ICICT)', IEEE, pp. 12–18.

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)', pp. 1–5.

Wei, Y., Kudenko, D., Liu, S., Pan, L., Wu, L. & Meng, X. (2018), A reinforcement learning based workflow application scheduling approach in dynamic

cloud environment, *in* 'Collaborative Computing: Networking, Applications and Worksharing: 13th International Conference, CollaborateCom 2017, Edinburgh, UK, December 11–13, 2017, Proceedings 13', Springer, pp. 120–131.

Wu, Q. & Wolf, T. (2008), Dynamic workload profiling and task allocation in packet processing systems, *in* '2008 International Conference on High Performance Switching and Routing', IEEE, pp. 123–130.

Xing, Y. & Zhan, Y. (2012), Virtualization and cloud computing, *in* 'Future Wireless Networks and Information Systems: Volume 1', Springer, pp. 305–312.

Ye, K., Wu, Z., Wang, C., Zhou, B. B., Si, W., Jiang, X. & Zomaya, A. Y. (2014), 'Profiling-based workload consolidation and migration in virtualized data centers', *IEEE Transactions on Parallel and Distributed Systems* **26**(3), 878–890.

Zhang, Q. & Boutaba, R. (2014), Dynamic workload management in heterogeneous cloud computing environments, *in* '2014 IEEE Network Operations and Management Symposium (NOMS)', IEEE, pp. 1–7.