Towards Improving the Reliability of Live migration Operations in OpenStack Cloud

Armstrong Foundjem and Foutse Khomh

SWAT – École Polytechnique de Montréal, Québec, Canada,
{a.foundjem, foutse}@polymtl.ca
http://swat.polymtl.ca/

Abstract. Cloud computing has become commonplace with the help of virtualization as an enabling technology. Virtualization abstracts pools of compute resources and represents them as instances of virtual machines (VMs). End users can consume the resources of these VMs as if they were on a physical machine. Moreover, the running VMs can be migrated from one physical machine to another without disrupting services; a process known as live VM migration. Live VM migration is a powerful tool that system administrators can leverage, for example to balance the loads in a data center or relocate an application to improve its performance and-or reliability. However, if not planned carefully, live migration can fail, which can lead to service outage or significant performance degradation. Hence, it is utterly important to be able to assess and forecast the performance of live migration operations before they are executed. The research community have proposed models and mechanisms to improve the reliability of live migration. Yet, because of the scale, complexity and the dynamic nature of cloud environments, live migration operations still fail. This paper presents RISULM; a novel approach that uses reinforced and supervised learning techniques to predict live migration failures, and to decide about “where" and “when" a VM should be migrated. Our result suggests that RISULM can predict failures of running VMs during the process of live migration with an accuracy of 95%. Moreover, RISULM can also reduce the migration time and downtime of VMs during live migration by 74% and 21% respectively.

Keywords: Cloud Computing, Live Migration, virtualization, Data center, Failure, Prediction

1 Introduction

Nowadays, cloud computing is attracting many organizations thanks to its economical benefits [10]. Both industrial and academic organizations are moving applications to the cloud [2]. Users can lease the services of these applications, ramping up or down the capacity as they need and paying only for what they use. Applications deployed in the cloud typically run in virtual machines (VMs) or containers that are distributed across multiple physical machines hosted in data centers. To respond to changing users' demands, organizations often have to

migrate VMs out of overloaded and–or–overheated servers. VM migration is also necessary during servers' maintenance and–or consolidation. When instances of VMs are moved (i.e., migrated), it is important to keep service downtime to its minimum.

This work aims at improving the reliability of live migration operations. We want to make optimal decisions about ***where*** (the destination Node) and ***when*** (the appropriate time) to live migrate a VM; with the aim to minimize the migration time m_T and the downtime d_T of the migrated VM. To achieve this goal, we have developed RISULM, a ReInforced and SUpervised Learning based approach that allows stakeholders to predict the failure of a VM (if migrated), and identify the time of migration and the destination node that would ensure a minimum down time. Our proposed approach is developed on OpenStack, using KVM and Xen hypervisors. We selected OpenStack because it is open-source, which allows us to modify its source code. Both KVM and Xen support pre- and post-copy migration, and they are the preferred hypervisors for OpenStack [9].

To assess the effectiveness of our proposed approach, we conducted a series of experiments comparing the effectiveness of RISULM with the effectiveness of the Machine Learning based Downtime Optimization approach – MLDO [1], and the Load Balancing Fault Tolerance strategy – LBFT [8]. Specifically, we addressed the following two research questions :

RQ 1: Can we accurately predict live migration failures?
We captured the state of the environment i.e., the CPU%, RAM%, Network-bandwidth utilization of the running virtual machines, on the source and destination Nodes, in both idle and loaded conditions. We used these parameters to train a Random Forest model. The training was performed as changes occurred in the environment. The obtained Random Forest model was able to predict live migration failures with an accuracy of 95%.

RQ2 : Can live migration scheduling be improved using Markov Decision Processes?
We implemented RISULM using Markov decision processes (MDP) to decide *when* and *where* to migrate VMs. We compared the performance of RISULM against the native OpenStack filter, MLDO and LBFT approaches and found that RISULM can reduce VM migration failures by 23.3%, migration time by 74%, and VM downtime by 21%. MLDO could reduce migration failures by 14.6%, migration time by 63% and downtime by 13%, whereas LBFT could only reduce migration failures by up to 10.2%, migration time by up to 55% and downtime by up to 11%.

The remainder of the paper is organized as follows.

Section 2 presents RISULM and the design of our experiments that aimed to assess the effectiveness of RISULM. Section 2.5 presents and discusses the results of our experiments, while Section!3 discusses the threats to the validity of our study. Section 4 discusses the related literature on VM live migration. Section 5 concludes the paper and outlines some avenues for future works.

2 Design of our Study

In this section, we explain the steps followed to implement and evaluate RISULM. We also report the results of our evaluations. For the sake of page limitations, we report only selected results per technology used in this study. However, more details can be found in our technical report[1], which also shows the details of our models.

2.1 Implementation of RISULM

To answer our research questions, we designed and conducted two sets of experiments. In each set of the experiment, we compared the result of RISULM, MLDO, and LBFT against that of OpenStack using the native scheduler. We used a fault injection framework [11] for OpenStack, which we customized, to randomly select one of the compute nodes and inject faults to the nodes, which will make the node to degrade up to the point of shunting down. This way we examine how our model can simulate a real world scenario, when a node becomes faulty or inconsistent in a cluster or a datacenter.

To identify “when" and “where" to migrate a VM, we propose an approach that given information about past VM live migration operations, is able to assess the risk of failure of a future VM migration operation, and identify the destination node that would ensure a minimum down time. The proposes approach is based on a prediction model and a Markov decision processes (MDP).

To select a suitable prediction model, we evaluate the performance of the following machine learning techniques on VM live migration data generated from the OpenStack cluster illustrated on Figure 1: Naive Bayes (NB), K - Nearest Neighbor (KNN with Euclidean distance), and Random Forests (RF). We choose these techniques because they have been used in previous studies from the literature to build predictive models [5], [4],[7], [12].

Each node in the OpenStack cluster from Figure 1 has the following configuration:

- CPU: Intel Xeon Processor E5-2690 (20M Cache, 2.90 GHz, 8.00 GTs Intel QPI)
- RAM: 128GB
- QEMU version 2.6.0
- Libvirt v1.3.2
- Nova (Liberty) 12.0.5, with python-novaclient 2.30.2
- Hard Disk Drive (HDD) SATA : 2TB, western digital RED.
- Ubuntu 14.04

The version of OpenStack installed is Mitaka. To generate VM live migration data, we executed several VMs live migration operations, injecting failures in the different nodes of our clusters. The failures are injected using the framework, which we customized, to randomly select one of the compute nodes and inject faults to the nodes, which will make the node to degrade up to the point of

[1] “https://github.com/foundjem/thesis”

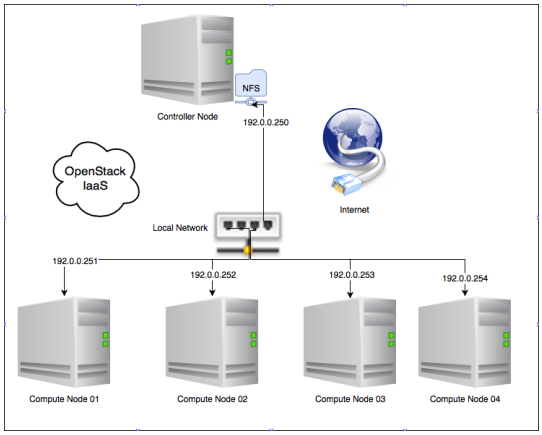


Fig. 1: Our experimental setup and configuration

shunting down. This way we examine how our model can simulate a real world scenario, when a node becomes faulty or inconsistent in a cluster or a datacenter. The configuration of the migrated VMs is described in Table 1.

Table 1: Configuration of the migrated VMs.

Flavor	VCPUs	Disk(in GB)	RAM(in MB)
m1.tiny (t)	1	10	1024
m1.small (s)	1	20	2048
m1.medium (m)	2	40	4096
m1.large (l)	4	80	8192
m1.xlarge (xl)	8	160	16384

We launched[3] 125 VMs (that is twenty five instances for each of the five flavors: Tiny, Small, Medium, Large and xLarge). We executed live migration with all the flavor types of VMs from Table 1. To train and test the machine learning models, we collect the metrics described in Table 2 and conduct a k-fold cross-validation, training the models on 80% of the data and testing them on the remaining 20%. We repeat the process five time and compute average precision and recall values. We compared the performance of NB, KNN, and RF, and found that RF consistently outperforms NB and KNN. Hence, we selected RF to be included in our proposed approach RISULM.

We rely on MDP processes to identify the optimal destination and migration time for a VM live migration. A MDP is defined as a tuple $\langle \mathcal{S}$, $\mathcal{A}$,$\mathcal{P}$, $\mathcal{R}$, $\gamma \rangle$ where :

- $\mathcal{S}$ represents our set of finite states,
- $\mathcal{A}$ represents our set of finite actions,

[3] create VMs and keep them running on the node where they were created

Table 2: Metrics used to predict VMs live migration.

Metrics of Our Study			
Metrics	Source Node	Destination Node	Virtual Machine
RAM%	**H**ost running memory intensive applications will not live migration in OpenStack.	**T**he memory state of the destination machine that will hosting the VM.	**VM_SIZE**, running memory intensive application have high changes of failure.
CPU%	**I**ntensive workload may have scheduling problem and delay live migration.	**I**ntensive workload on the destination, will prolong migration time.	**vCPU**, intensive applications running on a live migrated VM may disrupt it state [2]
Network%	**B**andwidth available during live migration affects downtime and migration time.	**B**andwidth congestion of destination node affects live migration time.	**B**andwidth available for VM, affects downtime and migration time.
Page Dirtying Rate %	**D**uring iterative pre-copy phase, all the memory pages from Node A is copied to Node B whereas the VM is still running on Node A.	**H**owever, If there is a change in the memory pages, they become dirtied and needs to be copied also to Node B .	**P**ages copied until the rate of copying becomes more than rate of dirtying.

- $\mathcal{P}$ represents a transitional probability function moving from state s to state s' when action a is taken,
- $\mathcal{R}$ represent an immediate reward that is obtained when action a is taken,
- γ represent a discounting, which can be finite or infinite. This actually determines our termination criteria.
- π represents a policy that maps a state s to an action a. Our goal using MDP is to find optimal policy that maximizes reward, as we iterate through the policies.

Our implementation of MDP uses policy iteration as shown on algorithm 1, the purpose here is to return the optimal policy, which the MDP needs, to know the best time to migrate a VM and the available destination. The process continues until there are no more changes in the system states, line 11 to 26. The most subtle part of policy iteration is line 13, which evaluate the expected sum of accumulated rewards $\mathcal{R}$ when starting from a given state s_i and acting optimally. We loop from line 14, 16 to 24 and 25 observing if there is a change of policy, in the end, we return the optimal policy, i.e., line 28.

2.2 Evaluation of RISULM

To evaluate RISULM and answer our research questions, we test the following null hypotheses:

- H_0^1 : There is no difference between the average downtime of VMs that are live migrated using the OpenStack scheduler and RISULM.
- H_0^2 : There is no difference between the average migration time of VMs when the live migration is performed using the OpenStack scheduler and RISULM.
- H_0^3 : There is no difference between the average downtime of VMs that are live migrated using the OpenStack scheduler and MLDO.
- H_0^4 : There is no difference between the average migration time of VMs when the live migration is performed using the OpenStack scheduler and MLDO.

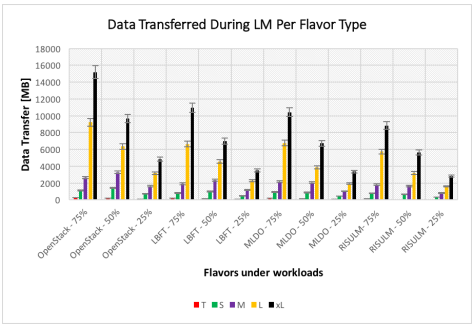
– H_0^5 : There is no difference between the average downtime of VMs that are live migrated using the OpenStack scheduler and LBFT.
– H_0^6 : There is no difference between the average migration time of VMs when the live migration is performed using the OpenStack scheduler and LBFT.

We use the Mann-Whitney U test to accept or refute the null Hypotheses H_0^x, for $x \in \{1, 2, \ldots 6\}$. The Mann-Whitney U test is a nonparametric statistical test used for assessing whether two independent distributions have equally large values.

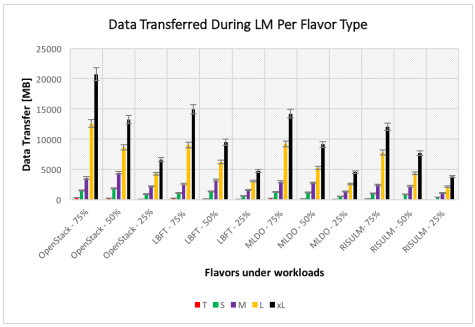
2.3 Evaluation approach

In this section, we discuss how RISULM was evaluated. (i) We compare the performance of RISULM (especially on accuracy on predicting failures of live VMs during migration, the total migration time, and the downtime) against OpenStack native scheduler. (ii) Also, we use MLDO and LBFT and compare their performance against OpenStack scheduler, (iii) we then evaluate the performance of RISULM, MLDO, and LBFT against OpenStack with different hypervisors using both the pre-copy, and post-copy algorithms, as shown in figure 6.
First and foremost, RISULM uses both SL and RL to make decision, firstly, we run an experiment with RISULM (rf-model). Secondly, we run an experiment with the MDP implementation, which enable us to make timely decisions on where and when to migrate live VMs. Thirdly, we combined both random Forests model (RF - model) and the MDP - Model, which forms a hybrid model (RISULM), to understand if the performance of the MDP could improve the performance of RISULM. The output of both models are Boolean ("YES", "NO") and we considered only YES YES; conservatively, we tested our models with the fault injection framework (FF), which simulates disaster on a datacenter.

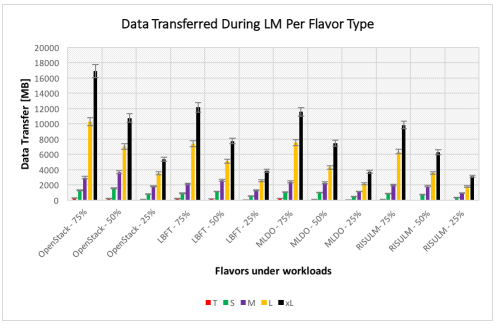
In our evaluation, we use the stress-ng tool [6]. With these tools, we set various categories of workload on the VMs before migration. Base on our initial observations while trying all possible cases of failures and successes in live migration, we did several combinations of workloads and found that, we could categorize the VMs workload such as idle, 25%, 50%, 75%, and 100% load conditions. However, we observe that, when the VMs are loaded above 75% workload they always fail live migration, whereas below 25% of workload they always success live migration under stable network conditions.
Therefore, we consider 75% as full loaded condition, 50% as average loaded conditions and 25% as unloaded condition (idle) of the VM size (VMSIZE). In our experimental setup, all the nodes are connected through a 1Gbps link, to implement the models in this study; OpenStack scheduler, MLDO, LBFT, and RISULM,using KVM and Xen hypervisors with both post-copy and pre-copy algorithms. Values for migration time, downtime and data transfer rates were computed and recorded. Table 4 provide the description of the metrics used to evaluate VM live migration operations.



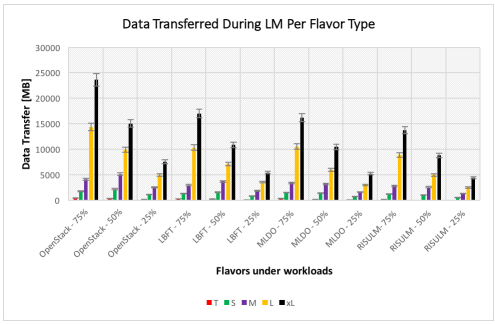
a) *Pre-copy with KVM*



b) *Post-copy with KVM*



c) *Pre-Copy with Xen.*



d) *Post-Copy with Xen.*

Fig. 2: VMs Data Transfer Rate per Models.

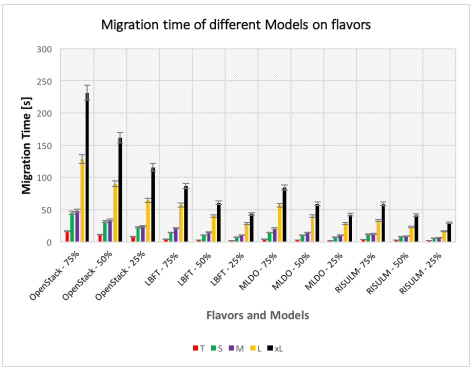
2.4 Results

We observe a statistically significant decrease of migration time and downtime when using the models (RISULM, MLDO, and LBFT) in comparison to OpenStack schedulers. Moreover, the performance of these models vary depending on the choice of the hypervisor used and the algorithm, this is true for all types of flavors and workloads respectively. On average, the KVM had higher accuracy on predicting failures than Xen and post-copy algorithm performs better than pre-copy in terms of optimizing migration time and downtime.

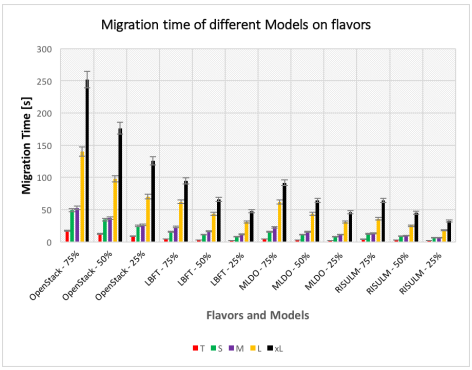
This observation is important to note because depending on the type of application being deployed on the VM during live migration, the choice of the algorithm or hypervisor might affects how likely it will success to migrate, which might also affects the migration time and downtime of the VMs.

We observe the following decreases in migration time and downtime respectively for the following combination of live migration algorithm and hypervisors respectively as shown in table 6.

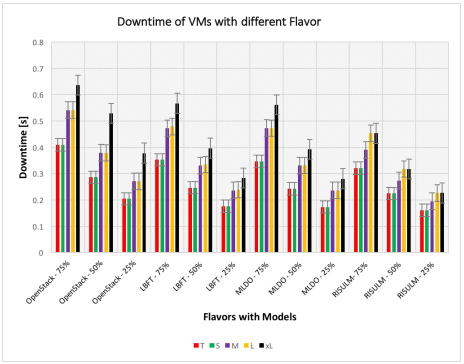
Furthermore, we observe a hike in data transfer that affects the total migration time per flavor type, see figure 2 and 3 as the migration time grows exponentially irrespective of the algorithm used but differs in magnitude depending



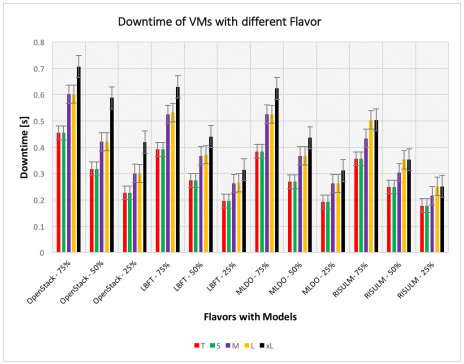
a) *Post-copy with KVM, m_T*



b) *Post-copy with Xen, m_T*



c) *Post-copy with KVM, d_T*

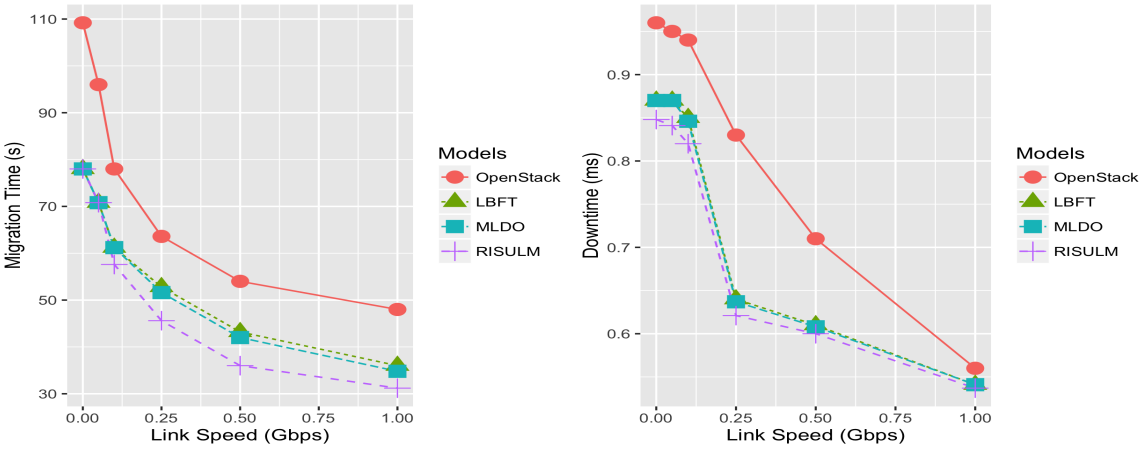


d) *Post-copy with Xen, d_T*

Fig. 3: VMs Migration/Downtime per Model.

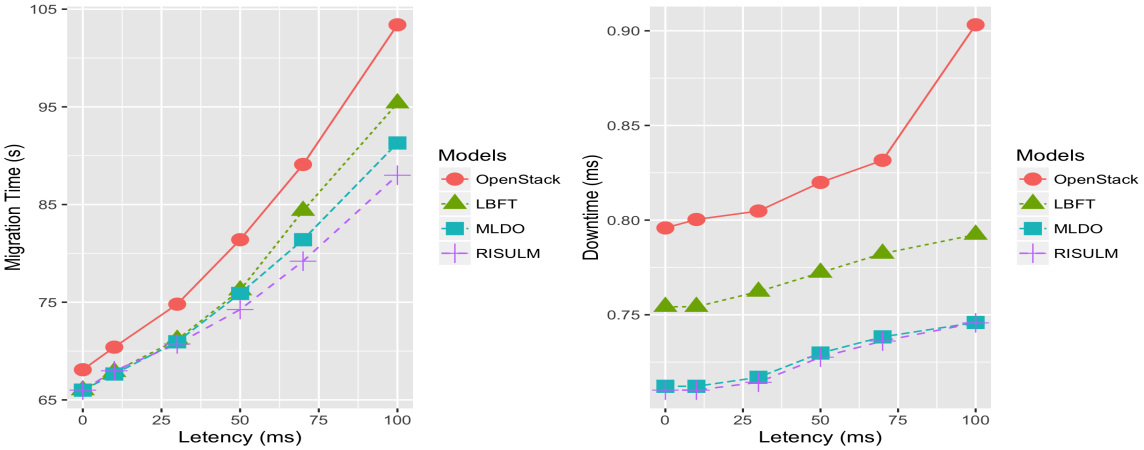
on the hypervisor type, for example KVM in general have a lower migration time than Xen and post-copy has a shorter migration time than pre-copy. On the other hand, we observe a different pattern on the downtime. Most surprisingly is that fact that KVM in both cases of the algorithm records less migration time and downtime. Even though, Xen has the advantages of paravirtualization, which enables VMs to run without an emulator such as QEMU as in the case of KVM. That said, Xen shows better performance than KVM in terms of VMs state after migration to the destination. All the VMs had their states unchanged unlike with KVM, about 37% of the VMs had an altered state, which requires reboot. That said, we included VMs state in failure classifications, hence, Xen performs better than KVM. On the other hand, one possible explanation why KVM was faster than Xen in terms of migration time could be because KVM resides in the Linux kernel itself.

Additionally, we observed the effect network links peed and latency has on both migration time and downtime.



a) *Pre-Copy KVM.* **b)** *Pre-Copy KVM.*

Fig. 4: Effect of Link speed on m_T & d_T on Models.



a) *Post-Copy Xen.* **b)** *Post-Copy Xen.*

Fig. 5: Effect of Latency on m_T & d_T on Models.

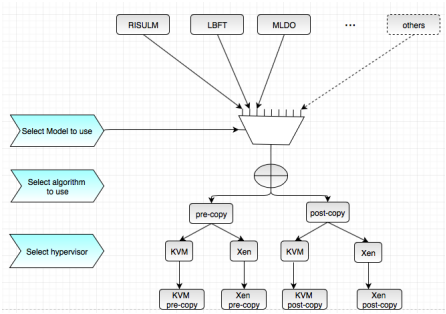


Fig. 6: Implementation of RISULM using different models and hypervisors.

On link speed: on all the flavors types, we vary the link speed and measure the migration time and downtime of the VMs instances. We observe the impact that link speed has on both migration time and downtime, see figure 4.
On latency: we measure the time delay data package running on VMs goes from one node to another using ping (there could be some limitation using ping however, it was not the case for our study) all the four combinations in out study, we use all the models including OpenStack scheduler. We observe the effects latency has on both migration time and downtime, see figure 5.
Our observations suggest that, migration time increases exponentially as latency increases. Initially, between 0 to 10 ms latency, RISULM, MLDO and LFBT were at par but clearly distinct after 3 ms where RISULM started deviating from MLDO and LBFT throughout whereas MLDO moves at part with LBFT up-to 50 ms. This clearly shows that in most applications, RISULM will be optimal in terms of latency. For downtime, we had these observations: As the link speed increases, we observe a shape linear drop of downtime between 13ms to 25 ms, (we had observe an exponential decrease of migration time within this range earlier). and from 25 ms to 1Gbps, the conclusion we make here is that even though both migration time and downtime are directly proportional, their relationship is not necessarily linearly.

The reported p-value and cliff's delta values in table 5 and the other p-values not shown in this paper, show statistically significantly p-values, which implies that, the p-values are significantly different with RISULM showing better performance than MDLO and LBFT models.

2.5 Scalability of RISULM

To assess the scalability of RISULM, we deployed it on AWS and tested it on the migration of EC2 extra large instances; we set up 30, 60 and 150 VMs. We aim at finding out how scalable RISULM could be; that is, when Nodes are added to or reduced from the pool of available Nodes. RISULM turns out to be scalable, which can handle a variable number of Nodes and VMs in a datacenter.

3 Threats to Validity

In this section, we discuss the limitations of our work following common guidelines for empirical studies.

- **Construct validity threats** Relates to the meaningfulness of our measurement result, in other words, this threat is solely due to errors in the measurement of metrics. To compute our metrics, we use *top*, *htop*, $iftop$ and other scripts to capture parameters for our studies such as RAM usage, CPU, Bandwidth, VIRT, dirty page rate, etc. We did not measure the accuracy of this tools. It can be a threat to our captured metrics.
- **Threats to internal validity** Relates to alternative explanations to describe our finding. The implementation of the MDP and ML-Model can be a threat to our study, moreover, our design window (the time frame we set to capture environment parameters) can also be a potential threat because all the Nodes are not running the same application synchronously, even though the time is synchronized.
- **Threats to External validity** Relates to possible generalizing our studies. More validation with different machine learning techniques, possible unsupervised techniques. Different method of ensemble could be used as well to understand the impart of MDP on migration time and downtime.
- **Reliability validity threats** Replication is important in science, this treat addresses how we replicated MLDO and LBFT, Our work can be replicated. To this effect, we attempt to provide necessary details to replicate our study.
- **Conclusion validity threats** Describe relation between the treatment and it outcome. To address this, we were attentive not to violate the assumptions of our null hypothesis.

4 Related Work

Several works on live migration, using different types of hypervisors (such as KVM, QEMU, Xen etc.) exist in the literature. Some of these works have proposed models to predict the performance of live migration operations, and mechanisms to optimize migration time. In this section, we discuss works that are closely related to our research direction.

Biswas et al. [3] investigate performance issues that occur during live migration of VM in OpenStack and observed that VM sizes and workloads have an impact on VM migration times. However, the physical distance between the source and destination nodes did not affected VM migration time and downtime. In this paper, we have included both VM and workload characteristics in our models.

Arif et al. [1] studied live migration in Wide Area Networks (WAN) and observed that, although several approaches have been proposed to reduce downtime during live migration, none have considered the case of VM migrations through a WAN. To help fill this gap in the literature, they designed MLDO; a Machine Learning based Downtime Optimization approach that uses predictive

mechanisms to estimate the downtime of VMs during live migrations over WAN networks. The input of MLDO are system's parameters (e.g., CPU, RAM, Network, and the load conditions of the machines). MLDO has two complementary modules: a *monitoring* and a *processing* module. The monitoring module collects data about changes occurring in the system (both in physical and virtual machines). These data are used by the processing module train classifiers (e.g., the c4.5 algorithm) that are used to generate VM migration decisions. Using MLDO the authors were able to reduce downtime by 15%, during VM migration. In this paper, in addition to minimizing VM downtime, we also aim to reduce the migration time. In our analysis, we will compare the performance of our proposed approach RISULM with that of MLDO.

Li and Wu [8] proposes a Load-Balancing and Fault Tolerance Strategy to help stakeholders decide about when to migrate a VM. Their proposed strategy consists in monitoring resource usages in order to derive threshold values that can help identify busy nodes that should be migrated to balance the workload in the cloud environment.

The authors conducted a series of experiments to assess the effectiveness of their proposed approach, and concluded that LBFT can accurately predict migration time in 90% of cases, saving between 35% ~ 50% of migration cost in comparison to a random strategy. In this paper, we compare the performance of RISULM against the performance of LBFT.

5 Conclusion

In this study, we examine live migration of VMs in OpenStack cloud. We proposed a novel approach, RISULM that uses supervised and reinforcement learning techniques to study live migration. RISULM aims at studying if failures in live migration could be accurately predicted and if the MDP model could improve live migration scheduling in order to accurately determine where and when VMs should be migrated. Moreover, we compared the performance of RISULM against state of the art models from the literature, i.e., MLDO and LBFT, and found that RISULM outperforms both models in terms of accuracy, migration time and downtime.

To simulate a real world scenarios, we used a fault injection framework, which randomly injects faults on targeted Nodes. In other to test the accuracy of our models, we collected real-time data of the system for both models.

Our results suggest that, we can accurately predict failures of VMs with a 95% accuracy. Our results also suggest that MDP model can improve live migration scheduling, reducing the downtime of VMs by approximately 21% and migration time by approximately 74%.

We also implemented our proposed model; RISULM on different hypervisors using both the pre-copy and post-copy algorithm. Results suggest that post-copy algorithms have better migration time and downtime as compare to pre-copy algorithm. Which suggests that depending on the application, the choice of hypervisor and algorithm to use is critically important.

Algorithm 1: Policy_Iteration($\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma$)

Input :
Extract the environmental variables that describe the current state of the systems
$\mathcal{S} \leftarrow \{s_1, s_2, \ldots, s_n\}$
$\mathcal{A} \leftarrow \{a_1, a_2, \ldots, a_n\}$
$\mathcal{P} \leftarrow P(s'|s, a)$
$\mathcal{R} \leftarrow R(s, a, s')$
$\gamma \in]-1, 0[$
Output :
optimal policy π^*
Local variable :
action_array $\pi[S]$
real_array $\mathcal{V}[S]$
$\pi^* \leftarrow \pi'$
repeat
  $sysChange \leftarrow False$
  $\mathcal{V}[S] = \sum_{s' \in S} P(s'|s, \pi[s])(R(s, a, s') + \gamma \mathcal{V}[s'])$
  for *each* $s \in \mathcal{S}$ **do**
    $\mathcal{Q}\_global \leftarrow \mathcal{V}[s]$
    for *each* $a \in \mathcal{A}$ **do**
      $\mathcal{Q}_{s,a} = \sum_{s' \in S} P(s'|s, a)(R(s, a, s') + \gamma \mathcal{V}[s'])$
      if $\mathcal{Q}_{s,a} > \mathcal{Q}_{global}$ **then**
        $\pi[s] \leftarrow a$
        $\mathcal{Q}_{global} \leftarrow \mathcal{Q}_{s,a}$
        $sysChange \leftarrow True$
      else
      end
    end
  end
until $(sysChange = False)$

return π^*

Table 3: p-value and cliff's-δ showing **Downtime** results of significant p-values for **RISULM** against OpenStack with KVM, Xen, using pre-, and post-copy algorithm.

Flavors	Pre-Copy KVM		Post-Copy KVM		Pre-Copy Xen		Post-Copy Xen	
	p-value	Cliff's δ	p-value	Cliff's δ	p-value	Cliff's δ	p-value	Cliff's δ
Tiny	$8.341x10^{-06}$	large	$8.014x10^{-06}$	large	$8.625x10^{-06}$	large	$8.521x10^{-06}$	large
Small	$6.224x10^{-06}$	large	$6.545x10^{-06}$	large	$7.332x10^{-06}$	large	$2.320x10^{-06}$	large
Medium	$6.512x10^{-06}$	large	$2.445x10^{-06}$	large	$4.515x10^{-06}$	large	$3.237x10^{-06}$	large
Large	$5.409x10^{-06}$	large	$2.011x10^{-06}$	large	$4.611x10^{-06}$	large	$3.012x10^{-06}$	large
xLarge	$5.029x10^{-06}$	large	$2.058x10^{-06}$	large	$6.310x10^{-06}$	large	$2.072x10^{-06}$	large

Table 4: Metrics used to evaluate live migration of VMs.

Metrics	Description
Migration Time	Total migration time T_M(sec) it take from Node A to initialize the running VM for live migration process to the time it start running on Node B.
Downtime	During the Stop and copy phase, the time the VM is suspended in Node A and resumes on Node B. For optimal operation we need to keep this time to its minimal.

Table 5: p-value and cliff's-δ showing results of significant p-values for **Migration time**, **RISULM** against OpenStack Scheduler with different hypervisors and algorithms.

Flavors	Pre-Copy KVM		Post-Copy KVM		Pre-Copy Xen		Post-Copy Xen	
	p-value	Cliff's δ	p-value	Cliff's δ	p-value	Cliff's δ	p-value	Cliff's δ
Tiny	$2.101x10^{-06}$	large	$1.250x10^{-08}$	large	$1.140x10^{-05}$	large	$1.240x10^{-06}$	large
Small	$2.123x10^{-06}$	large	$1.206x10^{-08}$	large	$1.310x10^{-05}$	large	$1.480x10^{-06}$	large
Medium	$3.148x10^{-06}$	large	$1.185x10^{-08}$	large	$1.515x10^{-05}$	large	$1.619x10^{-06}$	large
Large	$1.501x10^{-06}$	large	$1.421x10^{-09}$	large	$1.701x10^{-05}$	large	$1.872x10^{-07}$	large
$xLarge$	$1.731x10^{-06}$	large	$1.578x10^{-09}$	large	$1.813x10^{-05}$	large	$1.990x10^{-07}$	large

Table 6: Percentage Reduction on Migration time and Downtime for Different hypervisors and Algorithms.

Model	Migration Time	Downtime	Technology
RISULM	74%	21%	Post-copy KVM
	66%	17%	Pre-copy KVM
	71.2%	19%	Post-copy Xen
	64.5%	15%	Pre-copy Xen
MLDO	62%	13%	Post-copy KVM
	57%	11%	Pre-copy KVM
	60%	11%	Post-copy Xen
	55%	9%	Pre-copy Xen
LBFT	55%	11%	Post-copy KVM
	51%	11%	Pre-copy KVM
	53%	9%	Post-copy Xen
	48.2%	9%	Pre-copy Xen

Bibliography

[1] M. Arif, A. K. Kiani, and J. Qadir. Machine learning based optimized live virtual machine migration over wan links. *Telecommunication Systems*, pages 1–13, 2016.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, Apr. 2010.

[3] M. I. Biswas, G. Parr, S. McClean, P. Morrow, and B. Scotney. An analysis of live migration in openstack using high speed optical network. In *2016 SAI Computing Conference (SAI)*, pages 1267–1272, July 2016.

[4] R. Blaser and P. Fryzlewicz. Random rotation ensembles. *J. Mach. Learn. Res.*, 17(1):126–151, Jan. 2016.

[5] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 161–168, New York, NY, USA, 2006. ACM.

[6] A. Casanovas, J. Alonso, J. Torres, and A. Andrzejak. Work in progress: Building a distributed generic stress tool for server performance and behavior analysis. In *2009 Fifth International Conference on Autonomic and Autonomous Systems*, pages 342–345, April 2009.

[7] T. De Poalo and J. Howard. Predictive modeling in practice: A case study from sprint. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1517–1517, New York, NY, USA, 2014. ACM.

[8] Z. Li and G. Wu. Optimizing vm live migration strategy based on migration time cost modeling. In *Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems*, ANCS '16, pages 99–109, New York, NY, USA, 2016. ACM.

[9] F. OpenStack. Openstack user survey statistics, 2013. `http://www.openstack.org/blog/2013/11/openstack-user-survey-statistics-november-2013/`.

[10] J. Wang, X. Xiao, J. Wang, K. Lu, X. Deng, and A. A. Gumaste. When group-buying meets cloud computing. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016.

[11] Y. Xiaoyong, L. Ying, W. Zhonghai, and L. Tiancheng. Dependability analysis on open stack iaas cloud: Bug anaysis and fault injection. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 18–25, Dec 2014.

[12] C. Xiong, D. Johnson, R. Xu, and J. J. Corso. Random forests for metric learning with implicit pairwise position dependence. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 958–966, New York, NY, USA, 2012. ACM.