Improving Virtual Machine Live Migration via Application-level Workload Analysis

Artur Baruchi and Edson Toshimi Midorikawa
University of Sao Paulo
LAHPC - Sao Paulo, Brazil

Marco A. S. Netto
IBM Research
Sao Paulo, Brazil

***Abstract*—Virtual Machine (VM) live migration is key for implementing resource management policies to optimize metrics such as server utilization, energy consumption, and quality-of-service. A fundamental challenge for VM live migration is its impact on both user and resource provider sides, including service downtime and high network utilization. Several VM live migration studies have been published in the literature. However, they mostly consider only system level metrics such as CPU, memory, and network usage to trigger VM migrations. This paper introduces ALMA, an Application-aware Live Migration Architecture that explores application level information, in addition to the traditional system level metrics, to determine the best time to perform a migration. Based on experiments with three real applications, by considering application characteristics to trigger the VM live migration, we observed a substantial reduction in data transferred over the network of up to 42% and the total live migration time decrease of up to 63%.**

***Keywords*-Live Migration; Cloud Computing; Performance Prediction; Virtualization;**

I. Introduction

Virtual Machine (VM) live migration techniques try to satisfy a given objective, such as cost reduction, through consolidation of several workloads to a few servers, or the performance increase of an application via load balancing. In general, VM migrations happen without the analysis of the VM workload and state. Moreover, live migration techniques and optimizations [1] [2] usually do not consider the state of the data center (*e.g.* live migrations in course and current network traffic). The absence of a control, for knowing when is the right moment for migrating a virtual machine, can lead to resource waste, poor customer experience, and even Service Level Agreement (SLA) penalties.

Our hypothesis is that choosing the right moment to trigger a live migration can lead to significant improvements, such as decreasing the VM downtime or avoiding network congestion. Live migration techniques are very sensitive to memory usage (at least, pre-copy [3], and post-copy [4] algorithms). Hence identifying the current VM resource type utilization (*e.g.* memory, CPU, I/O) can help speed-up a live migration. One key aspect we explore in this paper is that applications may have cycles that utilize more resources than others. For instance, scientific applications can have parallel processes that need to synchronize at time intervals or web servers can be more utilized during the day than during the night.

This paper introduces the Application-aware Live Migration Architecture (ALMA) that supports live migration policies, considers the application-level workload, and carries out a *cycle* identification. The architecture exploits the fact that understanding application characteristics can assist in better live migration decisions. The paper also presents an evaluation of the architecture by investigating the main metrics (VM downtime, total migration time, and network data transfer) in a set of experiments. The experiments also show that knowing the live migration overhead can help evaluate, before hand, whether a VM migration will be worthwhile. Compared to existing work [1], [5]–[12], we couple objective functions (consolidation, load balancing, etc.) with live migration controls, and identify application resource consumption cycles to trigger VM live migrations.

Therefore, the main contributions of this paper are:

- Introduction of an architecture for VM live migration that considers the application-level workload and a cycle identification;
- A method to quantify and predict the application degradation during the live migration and identify application cycles using Fast Fourier Transformation;
- Evaluation of the architecture considering metrics related to quality-of-service and resource management using real applications from different domains and a testbed with real servers.

II. System Architecture

A. Architecture Overview

In an architecture where no Live Migration control exists, once the new VM-to-Host map[1] is computed, it is submitted to hosts without any control and is subject to problems like network congestion [13]. In architectures where there is a Live Migration control it is implemented in the hypervisor layer, and does not interact with the objective function module. Usually, this control is designed to avoid network congestion and does not consider the application's behavior. Another important difference between our proposed architecture and the other two are the evaluated metrics. Most VM live migration architectures make decisions according to system metrics and not according to application characteristics.

Our architecture, the *Application-aware Live Migration Architecture (ALMA)*, computes the objective function to the cur-

[1]VM-to-Host map: selection of Hosts to run a given group of VMs.

rent VM map. The new map is transferred to a module called *Live Migration Control Engine (LMCE)*, which decides when to migrate the VMs. The orchestration of the live migration aims to minimize the network and application overhead and, mainly, unnecessary migrations.

B. Live Migration Control Engine

The Live Migration Control Engine (LMCE) module sits between the physical hosts and the objective function computation module. Once the new Host-to-VM map is computed it is applied by LMCE to analyze which VMs can be migrated according to the application workloads[2] and *cyclic* analyses.

Based on data collected from the VMs, LMCE can decide when and which VMs are the best candidates for migration. This information can be used in future to make cycle identification. If the application presents a cyclic behavior (*e.g.* synchronization barriers, which are network intensive, of a parallel application), LMCE could avoid live migrations during this time interval.

LMCE accepts the configuration of two time constraints. The first one is the maximum time allowed to postpone a live migration. The second one is the live migration cost, since the provider (or the cloud customer) can adjust an acceptable overhead for the application. LMCE analyzes the application resource consumption (*e.g.* memory, CPU, I/O) behaviour over time in order to trigger or postpone a live migration applied by the objective function computation.

C. Migration Cost Prediction

We define overhead as the amount of additional time to finish the workload once the live migration is committed. In order to compute the overhead, it is necessary to normalize the execution time in the hosts involved. Since hosts can have different processor technologies, the time to finish a given workload will be different too. In this paper we considered only processor-related metrics. First, it is necessary to compute the processor ratio between hosts. Once the ratio is computed, the overhead ($O_{A\to B}$) of the live migration from host A to host B is given by:

$$O_{A\to B} = \left[\left(Tm_{A\to B} - t_A\right) + \mathrm{t}_{TB}\right] - \mathrm{T}_B, \text{ where:}$$

$Tm_{A\to B}$: Total execution time of the application workload when migrating from host A to B;

t_A: Application elapsed time executing on host A;

t_{TB}: Application elapsed time executing on host A, but converted to time on host B (using the processing ratio);

T_B: Total execution time of the entire application workload on host B (without live migration occurrences).

The prediction $P_{A\to B}$ of how long a workload will run when migrating a virtual machine from host A to B, given that the workload already runs for a certain time interval in host A is:

$$P_{A\to B} = t_A + \left(\mathrm{T}_B - \mathrm{t}_{TB}\right) + O_{A\to B}$$

This prediction model considers the migration overhead and, more importantly, the hardware differences of the hosts involved in the live migration (the source host and the target host). As observed by Birk *et al.* [14], the Cloud is built on servers of different generations with different capacities and performance, hence this scenario should be part of any live migration prediction strategy and evaluation.

D. Cycle Identification

Many application workloads have a cyclic (or temporal) behaviour pattern [15]. Knowing in advance a likely application workload behavior can be useful for live migration strategies. An application that is about to stress a given resource type (CPU, memory, I/O) can have the migration request postponed to the near future, when its resource consumption is known to be more appropriate for a live migration. The estimation of the cycle size uses the Fast Fourier Transformation (FFT), which is used in other science fields (like physics) to identify cyclic patterns in natural events.

This kind of analysis can be done by storing the application workload history. The collected data is submitted to the FFT, which estimates the cycle size. The cycle is split in two parts, one with propitious live migration moments and the other with the moments that are not good for live migration (by splitting in two we could reduce the search space for the next step). Finally, to obtain in which moment the application is in terms of resource consumption at a given instant, we calculate the module (rest of the division) of the current instant and the size of the cycle (the pseudo code is presented in Figure 1).

Let an application with a cyclic behavior and metric values collected twice an hour, resulting in a total of 48 samples a day. Each sample is composed of workload details of various resources, such as memory, CPU, and I/O usage. Based on these metrics, we classify each sample as suitable or unsuitable to perform live migration (*e.g.* at moments with high paging rate, we classify the workload as an unsuitable moment). The sample is submitted to FFT, which gives us the cycle size. FFT could return a cycle size of 8 hours; *i.e.* every 8 hours the workload is restarted. During the cycle period of 8 hours, we may observe several oscillations between suitable and unsuitable moments for live migration. Hence, knowing the cycle size and how the oscillation between suitable and unsuitable moments will occur inside the cycle, we can estimate which moments to migrate a VM.

For LMCE to work properly and make the right decisions, it is necessary to receive data from three sources:

- **User Information:** Application deadline, which can come from either a cloud service provider or an end user;
- **Objective Function Information:** A new set of Hosts-to-VM maps must be provided;
- **Virtual Machine Application Classification Information:** VMs must send data about the application classification to LMCE. This classification can have two values: *Live Migration (LM)* and *Non Live Migration (NLM)*,

[2]Application workload: the stage in which the application is regarding the type of resource consumption, such as CPU, memory, and I/O.

```
Require: An array C with classification data from a VM for a
    certain time interval. Each classification sample must be chrono-
    logically ordered.

    $CycleSize \leftarrow FFT(C)$          ▷ Find the cycle size using FFT.

    $LMCount \leftarrow 1$
    $NLMCount \leftarrow 1$
    for $i = 1$ to $CycleSize$ do          ▷ Split cycle in two Arrays:
    ArrayLM and ArrayNLM.
        if $C[i] == LM$ then
            $ArrayLM[LMCount] \leftarrow i$
            $LMCount \leftarrow LMCount + 1$
        else
            $ArrayNLM[NLMCount] \leftarrow i$
            $NLMCount \leftarrow NLMCount + 1$
        end if
    end for
    $now \leftarrow CurrentMoment \bmod CycleSize$     ▷ Find in which
    moment inside the cycle we are.

    if $find(now, ArrayNLM)$ then
        $nextLM \leftarrow findNextBigger(now, ArrayLM)$     ▷ Find
    the next moment, greater than now, in ArrayLM.
        $remainingTime \leftarrow nextLM - now$
    else
        $remainingTime \leftarrow 0$               ▷ Inside a LM moment.
    end if
    return $remainingTime$
```

Fig. 1: Algorithm to identify workload cycles.

which present that the moment is right for migration or not, respectively. This data can be sent at different frequencies. The more data available, the better the accuracy of the application classification.

III. Evaluation

This section presents the experiments of ALMA. We used most common metrics of Live Migration (LM) [16] in two sets of experiments. The first set is based in artificial benchmarks, with well defined behaviour and artificial cycles. The second experiment set comprises three real applications, with their own cycle patterns. In addition we present the evaluation of the prediction model.

A. Testbed Configuration

We built a Cloud environment composed of five physical servers and a Network Attached Storage (NAS). We connected all components to a 24-port switch and created three separate networks from each other: a network for live migrations only, one network for NAS data transfer and a network for administrative tasks. We configured ten virtual machines with three configuration profiles. The small configuration has one vCPU and 768MB of memory, the Medium configuration has two vCPUs and 1GB of memory and the Large one with two vCPUs and 2GB of memory. The software configuration comprises OpenSuse Linux 12.1 with 3.1.10 Kernel on physical hosts. Xen 4.1.3 [17] was used as hypervisor and the Virtual Machine was installed with CentOS 5.9 and Kernel 2.6.18. ALMA was implemented in Perl (modules used to cycle calculation) and Python (due to a better API with Xen).

B. Benchmark Experiments

In this experiment set, we compare ALMA against a traditional consolidation (called *SysConsolidation*) which consists in consolidating all VMs, in specific moments during the workload, in two hosts (host_B and host_E). Figure 4 presents the VM placements after the consolidation. We run this test for 10 times, and in a given moment VMs were consolidated. During the tests the workload run from start to finish. After that we submitted consolidation at the same specific moments, but under ALMA control.

TABLE I: Benchmark workload experiments.

Metric-Policy	vm03_A	vm01_D	vm02_C
MigrTime-SysConsol	29.5 ±17.9	31.7 ±21.2	98.9 ±32.6
MigrTime-ALMA	14.0 ±5.2	15.3 ±4.9	37.7 ±0.6
Downtime-SysConsol	16.0 ±5.4	23.4 ±12.5	23.5 ±6.3
Downtime-ALMA	14.5 ±4.4	22.9 ±15.1	16.4 ±6.3

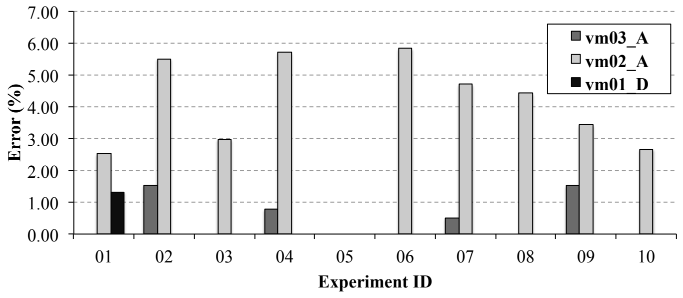


Fig. 2: Cycle Identification Accuracy: Benchmark Exp.

The evaluation of the proposed architecture uses the following metrics:

- **Total Migration Time:** This metric measures the time, in seconds, between the start of the migration submission and the moment that VM is completely released from the source host. This data was collected using the Xen log in debug mode;
- **Downtime Duration:** Time interval, in seconds, in which the VM is unreachable. This metric was collected using ICMP protocol, and the time interval which requests did not receive an answer, we considered it as downtime;
- **Network Data Transfer:** Amount of data, in MB, transferred in the network during the live migration. This data was collected from the switch;
- **Cycle Accuracy Identification:** This metric, in percentage, measured how accurate the Fast Fourier Transformation estimated the cycle size. This metric is the difference between the calculated cycle and the measured one. The closer to zero, the better the accuracy.

The benchmarks used to create artificial cycles are described in Table II. The workload was configured in three VMs (vm02_C, vm03_A and vm01_D, the darker VMs in Figure

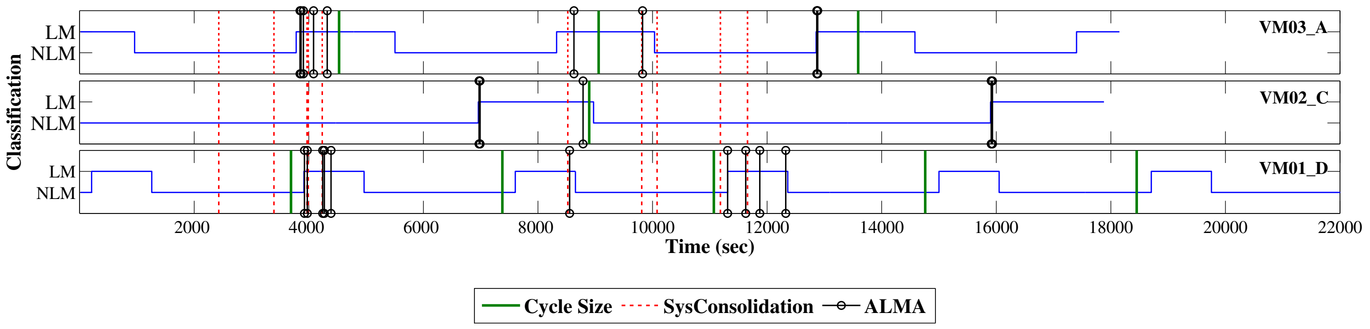


Fig. 3: Evaluation of consolidation moments for benchmark workloads: ALMA consolidations tend to be closer to the beginning or inside the suitable LM moments.

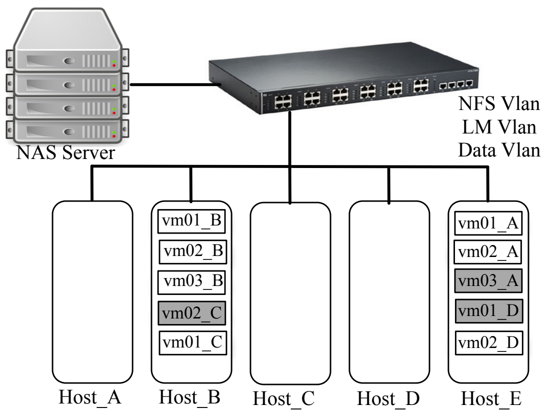


Fig. 4: After Consolidation.

TABLE II: Benchmarks Descriptions.

Benchmark	Workload Type	Description
SPEC [18]	CPU Bound	Used the twolf benchmark, which spends most of the execution time in internal loops doing mathematical calculations.
NAS NPB [19]	Memory Bound	Performs several changes in memory during the execution and consumes a considerable amount of RAM. We used class D of NPB.
IOZone [20]	I/O Bound	Used random read and write in files larger than physical memory (to avoid cache effects). The benchmark performs read and write operations in blocks of 4kb.

4) in order to improve the accuracy of measured metrics. The other VMs were idle during the experiment, but all VMs were migrated to create noise in network.

Figure 3 presents the moments of consolidation using SysConsolidation (in dashed red). The blue line represents the workload of VMs across time and the Classification of the interval. The valleys represent moments for Non Live Migration (NLM) and peaks represent the Live Migration (LM) moments. When the SysConsolidation was submitted, all VMs were migrated simultaneously to consolidate in Host_B and Host_E. When using ALMA, the consolidation was postponed to a more favourable moment to migrate the VM (line in black). This figure shows that ALMA was able to identify and submit the LM in suitable moments to the application workload (during the peaks). The green line in Figure 3 is the cycle size estimated by FFT. The pattern before the green line repeats during the workload, showing that FFT has a good approximation for the workload with benchmarks.

When using ALMA, it was possible to improve the first analysed metric, Total Migration Time (Table I), in 61% (vm02_C). The total migration time experienced a great reduction due to the reduction of the amount of data transferred over the network. In the best case scenario (where vm02_C takes 112 seconds to migrate, against 39 seconds using ALMA) host_C transferred 98% more data in 17 cycles of pre-copy algorithm. When using ALMA, copy occurred in 30 cycles, but the amount of data transferred was substantially reduced. For the second evaluated metric, the Downtime (Table I), there is no improvement. In some cases, the average showed a little improvement, but the standard deviation was virtually the same. The reason for no improvement in this metric is that the network infrastructure created by the VMM is not part of the pre-copy migration algorithm, it is an independent process that takes place just after the LM finishes. It suffers much more influence from the computational resources of the involved hosts than from the LM algorithm.

In the Data Transferred metric, we observed a considerable improvement: SysConsolidation: 10±2 and ALMA 10±2. Using ALMA, the reduction of data transferred over the network was about 42% (about 5GB less data transferred). Finally, the Cycle Accuracy Identification (Figure 2) metric shows the error between the FFT cycle calculation and the size of cycle actually measured. The workload submitted to vm02_C presented the highest error. This is due to the BT benchmark (memory intensive workload) which has a greater fluctuation during its execution (memory sensitive). Nevertheless, this error (about 6%) does not affect the ALMA benefits.

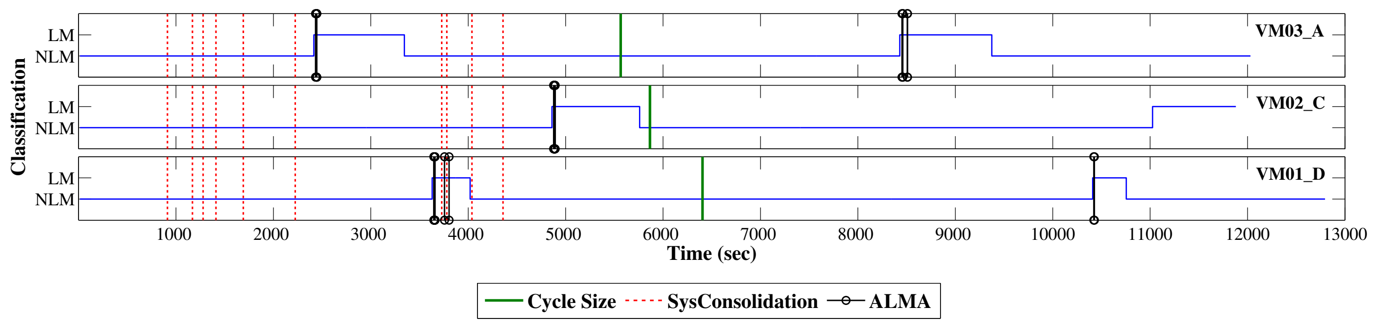


Fig. 5: Evaluation of consolidation moments for application workloads: ALMA consolidations tend to be closer to the beginning or inside the suitable LM moments.

C. *Experiments with Real Applications*

In this experiment set, ALMA is compared using three applications. We choose two scientific applications, with intensive usage of Memory and IO (major part of the workload) and some periods of CPU usage. The third application is a database running the TPC-H workload. The application description is summarized in Table III.

TABLE III: Application Description.

Application	Description	VM
OpenModeller [21]	Biological scientific application. It aims to find the likelihood of a specie to occur given a topography, vegetation and the climate of a region.	vm03_A
BRAMS [22]	Brazilian atmospheric modelling application. Used to weather forecast.	vm02_C
TPC-H [23]	Simulates an decision support system (Business Inteligence). It is composed of 22 queries that access a huge amount of data.	vm01_D

In Figure 5 it is plotted the workload classification (blue line) over the time. As previous test set, lines in red represent the SysConsolidation and lines in black represent consolidation postponed by ALMA. The green line is the cycle size. This figure shows that ALMA was able to identify the cycle pattern of the application and trigger the migration at the proper moment.

TABLE IV: Real application's workload experiments.

Metric-Policy	vm03_A	vm01_D	vm02_C
MigrTime-SysConsol	28.7 ±3.4	27.3 ±9.9	43.2 ±1.6
MigrTime-ALMA	10.8 ±0.4	10.1 ±1.1	36.6 ±0.7
Downtime-SysConsol	19.1 ±10.1	19.0 ±6.9	20.1 ±9.9
Downtime-ALMA	16.8 ±9.0	20.3 ±10.9	22.5 ±11.9

D. *Prediction Model Experiment*

Next, we present the Total Migration Time when using no control over the LM and when using the ALMA architecture (Table IV). We observe an improvement up to 67%

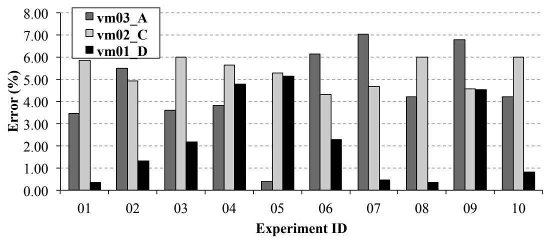


Fig. 6: Cycle Identification Accuracy: Application Exp..

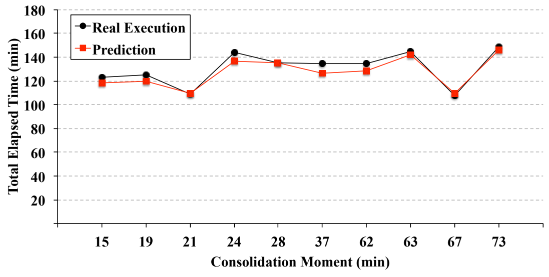
(vm03_A and vm01_D). The improvement of vm02_C was not so significant due to hardware differences of target hosts. The vm02_C machine is consolidated in Host_B, with less memory. So, to reserve the amount of memory for vm02_C, Host_B called the Balloon Driver many times (several calls to Balloon Driver[3] were logged in Xen log file). But, even with these constant calls, we observed an improvement of 15%. As in the previous test set, we did not observe any improvement for Downtime metric for the same reasons (Table IV). The statistical differences are irrelevant and no difference can be observed when using ALMA or SysConsolidation.

The Data Transfer during the LM was improved by 20% when using ALMA to postpone the LM: SysConsolidation 10±2 and ALMA 10±2. We observed a reduction of up to 2.3GB of data when ALMA architecture was used. Finally, the precision of FFT cycle estimation is presented in Figure 6. As can be observed, the accuracy of FFT when dealing with real workloads is deprecated. This is due to the expected fluctuation behaviour of the applications during its execution. But even with an accuracy deprecated, the error is still low (up to 7%).

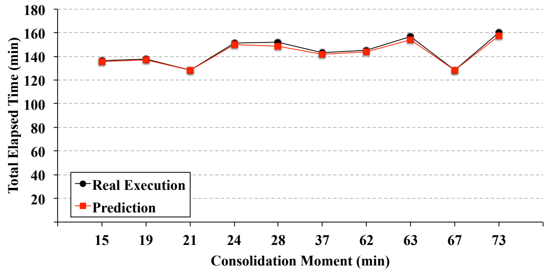
To evaluate the prediction model, we executed the same applications of the second test set. We chose different moments to trigger the LM and calculated the time the workload would take to finish using our prediction model and compared it with the actual measured time.

Th abscissa axis of Figure 7 represents the moment (after the workload started) where the LM was submitted. The ordinate

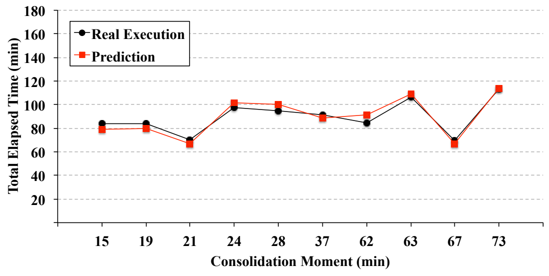
[3]Balloon Driver: It is a mechanism used to force the guest Operating System to give up of some unused memory pages. The unused pages return to the VMM to allocate to other VMs. This mechanisms allows to overcommit the memory available in host between the guests.



(a) Application prediction on vm03._A.



(b) Application prediction on vm02_C.



(c) Application prediction on vm01_D.

Fig. 7: Prediction model experiments.

axis is the elapsed time finish the workload. The prediction model showed a good accuracy, the average error for vm03_A was about 4 minutes (±1.69), 2 minutes (±1.07) for vm02_C and 4 minutes (±2.86) for vm01_D.

We observed that during intensive I/O workloads the prediction accuracy is decreased (vm01_D running TPC workload and vm03_A running the OpenModeller). This is due to the non-deterministic behaviour of I/O operations. The I/O requests issued by the VM can be answered in different times, which will depend on several variables. Memory and CPU workloads have a better behaved profile of execution, improving the accuracy of the prediction.

IV. Concluding Remarks

This paper presented an architecture that allows the coexistence of objective functions and a controlled live migration. The main differences from the existing literature are (1) the use of application-level metrics instead of system-level metrics to evaluate the workload and avoid workloads that harm the live migration process, (2) cycle identification that can postpone or avoid unnecessary migrations and (3) a prediction model that evaluates the migration cost and hardware differences to avoid live migrations that could, potentially, cause an SLA or QoS violation. We also explored the use of Fast Fourier Transformation to identify application cycles to assist the migration decisions. Our main findings are that: (1) when considering the application behaviour for live migration, there are considerable reductions in total live migration time and data transferred over the network and (2) considering hardware differences and migration costs when performing live migrations can improve live migration prediction models and should be considered.

Acknowledgment

This work has been supported and partially funded by FINEP / MCTI, under subcontract no. 03.14.0062.00.

References

[1] Clark *et al.*, "Live migration of virtual machines," in *NSDI*, 2005.
[2] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan, "Live virtual machine migration with adaptive, memory compression," in *Cluster Computing and Workshops*, 2009.
[3] M. M. Theimer, K. A. Lantz, and D. R. Cheriton, "Preemptable remote execution facilities for the v-system," *SIGOPS Oper. Syst. Rev.*, 1985.
[4] M. R. Hines, U. Deshpande, and K. Gopalan, "Post-copy live migration of virtual machines," *SIGOPS Oper. Syst. Rev.*, 2009.
[5] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application performance management in virtualized server environments," in *NOMS*, 2006.
[6] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *IM*, 2007.
[7] W. Voorsluys *et al.*, "Cost of virtual machine live migration in clouds: A performance evaluation," in *Cloud Computing*, 2009.
[8] S. Mehta and A. Neogi, "Recon: A tool to recommend dynamic server consolidation in multi-cluster data centers," in *NOMS*, 2008.
[9] T. C. Ferreto *et al.*, "Server consolidation with migration control for virtualized data centers," *Future Generation Computer Systems*, 2011.
[10] A. Stage and T. Setzer, "Network-aware migration control and scheduling of differentiated virtual machine workloads," in *ICSE Cloud*, 2009.
[11] K. Yang, *et al.*, "An optimized control strategy for load balancing based on live migration of virtual machine," in *ChinaGrid*, 2011.
[12] T. Wood *et al.*, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, 2009.
[13] M. Seki *et al.*, "Selfish virtual machine live migration causes network instability," in *APSITT*, 2012.
[14] R. Birke *et al.*, "State-of-the-practice in data center virtualization: Toward a better understanding of vm usage," in *DSN*, 2013.
[15] A. Khan *et al.*, "Workload characterization and prediction in the cloud: A multiple time series approach," in *NOMS*, 2012.
[16] P. Leelipushpam and J. Sharmila, "Live vm migration techniques in cloud environment: A survey," in *ICT*, 2013.
[17] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, 2003.
[18] N. Mirghafori, M. Jacoby, and D. Patterson, "Truth in spec benchmarks," *SIGARCH Comput. Archit. News*, vol. 23, no. 5, pp. 34–42, Dec. 1995.
[19] D. H. Bailey *et al.*, "The nas parallel benchmarks:summary and preliminary results," in *SC*, 1991.
[20] V. Tarasov, S. Bhanage, E. Zadok, and M. Seltzer, "Benchmarking file system benchmarking: It *is* rocket science," in *HotOS*, 2011.
[21] M. Souza Munoz *et al.*, "openModeller: a generic approach to species potential distribution modelling," *GeoInformatica*, 2011.
[22] S. R. Freitas *et al.*, "The coupled aerosol and tracer transport model to the brazilian developments on the regional atmospheric modeling system (catt-brams) part 1: Model description and evaluation," *Atmospheric Chemistry and Physics*, 2009.
[23] M. Kandaswamy and R. Knighten, "I/O phase characterization of TPC-H query operations," in *IPDS*, 2000.