Exploiting Workload Cycles for Orchestration of Virtual Machine Live Migrations in Clouds

Artur Baruchi[a,*], Edson T. Midorikawa[a], Liria M. Sato[a], Marco A. S. Netto[b]

[a] *University of Sao Paulo, Brazil*
[b] *IBM Research, Brazil*

Abstract

Virtual machine live migration in cloud environments aims at reducing energy costs and increasing resource utilization. However, its potential has not been fully explored because of simultaneous migrations that may cause user application performance degradation and network congestion. Research efforts on live migration orchestration policies still mostly rely on system level metrics. This work introduces an Application-aware Live Migration Architecture (ALMA) that selects suitable moments for migrations using application characterization data. This characterization consists in recognizing resource usage cycles via Fast Fourier Transform. From our experiments, live migration times were reduced by up to 74% for benchmarks and by up to 67% for real applications, when compared to migration policies with no application workload analysis. Network data transfer during the live migration was reduced by up to 62%.

Keywords: Live Migration, Cloud Computing, Virtual Machine, Workload Cycle Recognition, Server Consolidation, Fast Fourier Transform

1. Introduction

Through multiplexing techniques, virtualization allows for different operating systems and workloads to co-exist on the same hardware, without users' perception and interference among themselves. This feature is the core of cloud computing, which is a concept that dates back to the 1960s [1].

*Corresponding author
Email address: artur.baruchi@gmail.com (Artur Baruchi)

Live migration also allows a Virtual Machine (VM) to be moved across physical hosts with minimal interruption. This feature brings several benefits to cloud providers, including policy creation for physical host maintenance and energy consumption reduction via server consolidation [2]. Another benefit is to distribute VM load among physical hosts to meet circumstantial computing demand [3].

Despite the resource usage optimization brought by server consolidation or load balancing policies, they still generate VM performance degradation [4]. This problem comes mainly from live migration algorithms, whose performance is sensitive to VM memory usage. These algorithms can generate large data traffic to migrate VMs across hosts, especially when several VMs are moved simultaneously.

To address this problem, our previous work [5] introduced an Application-aware Live Migration Architecture (ALMA) which determines, before hand, suitable moments to move VMs among physical hosts according to VM workloads. Our hypothesis is that, we can reduce live migration side effects, such as data traffic, by choosing the right moment to trigger the migration process. This hypothesis comes from two observations. The first is live migration algorithms are sensitive to VM memory usage and the second is several industry and scientific workloads follow a resource usage cyclic pattern. Examples of these cycle scenarios are: (i) a Web service with higher access during some periods of day or due to application characteristics that can present long periods of processor usage after I/O access and (ii) parallel applications with processes exchanging synchronization messages.

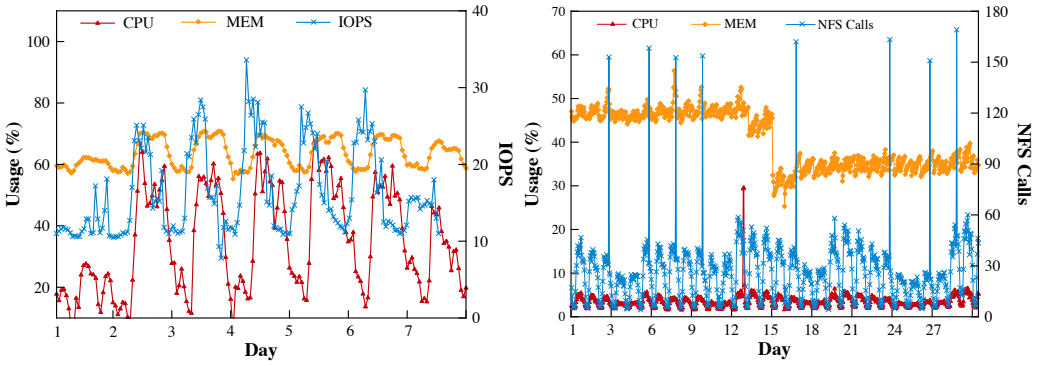
This work presents a detailed description of ALMA algorithms and the characterization process. Moreover, a new set of experiments to evaluate the effectiveness of using application characterization data to trigger migrations and a scalability analysis of our solution is presented. Therefore, we extended our work with the following contributions:

- Characterization and workload cycle recognition using Fast Fourier Transform with benchmarks and real scientific applications (§ 4);
- Live migration orchestration based on workload cycle recognition (§ 5);
- Evaluation of the architecture that implements the migration orchestration on a private cloud environment, including scalability analysis of the architecture with data from up to 1,000 VMs (§ 6).

2. Motivation and Problem Description

The overhead of live migrations comes mainly from algorithms, like pre-copy [6] and post-copy [7] and, at first, it is not the scope of server consolidation policies to deal with migration algorithm issues. As a result, server consolidation is barely used inside the cloud provider data centers [8].

The proposed architecture explores live migration algorithm characteristics and mediates between the consolidation policies and live migration algorithms, reducing the impact created by multiple concurrent migrations. Our architecture is based on the observation that several workloads have cyclic behaviors. Two examples of real workloads with cyclic behavior are presented in Figure 1, which illustrates the resource usage of a production data base from a telco company during a week and the resource usage of a pool of VMs of a big magazine publisher. We can see a cyclic behavior in both graphs and the best periods to perform live migrations is during the valleys of the graph, which consist of low resource usage.

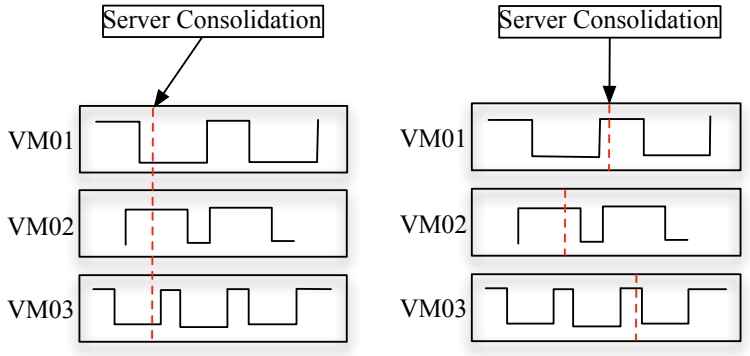


(a) Data Base real workload.

(b) Web real workload.

Figure 1: Real workloads with cyclic behaviors.

Figure 2 shows two scenarios for the timing of triggering migrations: one in which the consolidation triggers live migration just after the definition of the new physical hosts (without cyclic analysis), and the other scenario, where VM live migrations are orchestrated according to the their workload (with cyclic analysis). In the first scenario, live migrations produce more network traffic, since two (VM01 and VM03) out of three VMs are migrated during a not suitable moment. Moreover, if VM03 live migration were



(a) Without cyclic analysis. (b) With cyclic analysis.

Figure 2: Live migration orchestration using cyclic analysis: valleys represent moments where live migrations have potential to congest the network and peaks represent suitable moments for live migration.

postponed, its workload would be in a suitable moment, avoiding network congestion—which is the second scenario.

The main goal of the cyclic analysis is to identify and extract the workload's execution pattern and postpone live migrations with potential to harm the network. In Figure 2b, all live migrations were postponed to a suitable moment to reduce live migration time and network congestion. With our architecture, algorithms for server consolidation and live migration (pre-copy and post-copy) are not modified. Our solution intercepts all pending migrations, and orchestrates them according to the workload cycles. In practice, there should be modification only in the consolidation strategy APIs for concurrent live migrations.

The problems tackled in this paper are therefore: (i) how to detect application resource consumption cycles and (ii) how to use this information to know suitable moments for moving VMs across physical hosts in order to minimize network traffic and migration times.

3. Background

In this section we discuss concepts to understand how our architecture can reduce live migration overhead. These concepts cover pre-copy live migration algorithm and the basics of server consolidation.

3.1. Live Migration

The encapsulation of the operating system execution environment offered by virtualization is fundamental to live migrations [9]. This feature allows for: (1) at any time, a VM to be frozen, (2) all necessary information from restarting the execution to be stored in a file and (3) using this information to restart a VM on any physical host from the breakpoint.

Live migration algorithms can be classified according to the moment in which VM's memory content is copied to its replica on the destination host. The main migration algorithms are: *pre-copy* and *post-copy*. The pre-copy algorithm copies VM memory before it starts its execution in the destination host, whereas the post-copy copies VM memory after it starts its execution.

Research studies use four metrics to compare these algorithms—two are related to performance evaluation and two refer to VM workload overhead caused by live migration:

- **Live migration total time:** time interval between the start of the migration process and the VM execution beginning in destination host;
- **Downtime**: time interval in which VM is not running nor available to the user;
- **Execution time:** execution time of an application, including a possible migration time;
- **Throughput:** amount of data processed in a time interval with and without live migrations.

3.2. Pre-Copy

As pre-copy moves VMs only when their memory is already copied, this algorithm is more robust and widely used in commercial Virtual Machine Monitor (VMM) compared to post-copy—we used this algorithm in this paper. The memory is copied between hosts in several iterations and can be split into five stages [10]:

1. **Resource reservation:** it checks whether destination host has available resources to the VM;
2. **Iterative copy:** the VM's memory is entirely copied to destination host in the first iteration. In next iterations only the memory changed in last iteration is copied (dirty pages);

3. **Stop and copy:** the VM is suspended in source host and the last copy is performed;
4. **Shutdown:** the VM is stopped in source host and all resources allocated to the VM are released;
5. **Activation:** the VM is activated in destination host.

One of the main problems of this algorithm is the possibility of unlimited cycles of memory copy. To avoid this issue, some VMMs impose conditions to stop the copy iterations. Taking Xen VMM [11] as example, the stop conditions are: (i) less than 50 pages marked as dirty since the last iteration; (ii) maximum of 29 iterations; and (iii) amount of data transferred greater than three times of memory assigned to the VM.

Furthermore, this algorithm is sensitive to VM dirty page rate and network throughput. Some studies, such as the one from Strunk [12], formalized the dirty page rate and network throughput dependencies. The author defines an upper and lower limit of the pre-copy algorithm that are presented in Inequalities 1 and 2 containing limits to migration and downtime time:

$$\frac{V_{mem}}{B} \leq T_{mig} \leq \frac{(M+1) * V_{mem}}{B} \tag{1}$$

$$0 \leq T_{down} \leq \frac{(M+1) * V_{mem}}{B} \tag{2}$$

Where:
V_{mem}: amount of memory assigned to VM to be moved;
B: network throughput available;
T_{mig}: live migration duration;
T_{down}: downtime duration;
M: number of times allowed to copy the entire memory in iteration phase.

The lower limits of Inequalities 1 and 2 refer to an idle VM. In this situation, the migration duration is limited only by the network throughput between the two hosts and the downtime duration due to the low dirty page rate. However, the upper limit is related to a high dirty page rate, leading to the occurrence of several copy iterations. The worst case happens when the dirty page rate is higher than network throughput.

There are other factors that influence live migration duration, as observed by Xu et al. [13], such as the number of concurrent migrations. However, the dominant factor to live migration performance is the dirty page rate.

3.3. Server Consolidation

Server consolidation policies consist in choosing VMs, according to a given criterion, and concentrating them into a few physical hosts. Server consolidation is fundamental to help cloud providers reduce energy consumption. The main problem of consolidation policies is to find an optimal combination of VM placement in physical hosts. Another dilemma is how to avoid concurrent migrations to accomplish an objective [14].

The most common policies of consolidation are based on heuristics [15] or linear programming [14], where the former is more explored by researchers due to scalability issues. Consolidation based on heuristics are more flexible and the final solution (usually suboptimal) is obtained faster. Implementations based on linear programming are more efficient when dealing with several restrictions, such as Service Level Agreements (SLAs) and maximum number of concurrent live migrations.

4. Workload Characterization and Cycle Recognition

Workload characterization is the strategy used to collect information about what resources and their consumption by applications under analysis. An application can use several computing resources at the same time, but it is likely that a specific resource is being used more than others. This behavior can be static, meaning that an application can use a given resource more from the beginning to the end or dynamic, when during the application execution, it can use different resources and at various utilization levels. In this work, the workload characterization is defined in time unit to perceive fluctuations in resource usage during the application run time. We characterize the workload every fifteen seconds.

Server consolidation relies on workload characterization to avoid the placement of VMs competing for the same resources in the same physical hosts. There are challenges in cloud workload characterization due to the features of this paradigm, such as dynamic resource usage [16] and multi-tenancy. These factors can produce ambiguous signals to the workload classifier and generate wrong results.

Another common challenge in workload characterization is the data interpretation and gathering from VMs. Virtual Machine Monitors (VMMs) add a second level of indirection in order to isolate VMs located in the same host. This extra level, known as Semantic Gap [17], imposes several challenges to interpret and gather performance metrics.

The majority of traditional workload characterization strategies are computationally expensive and prohibitive to be implemented in a cloud data center running thousands of VMs simultaneously. Recent research findings specialized in VM characterization are difficult to implement in the cloud, because such strategies are mostly based on specific VMM metrics [18] or in VMM's source code modification [19].

A load index is a metric that aims at quantifying the system load in a given moment or during a time interval. In this work we used load indexes related to processor and memory resources, which are enough to identify other types of load, such as I/O.

4.1. Naive Bayes Classifier

The Naive Bayes (NB) classifier is based on Bayes's theorem, which is broadly used in the probability field. The Naive term is due to the assumption the events' probability are independent of one another.

The aim of Bayesian classifiers is to estimate the most probable class of a set of characteristics using probabilities known *a priori*, which are computed used a training data set. The Bayes's theorem requires at least three terms—one conditional probability and two unconditional probabilities—to compute a third conditional probability [20].

The quantitative results of the NB classifier is one of its main features. When submitting data to the classification, the NB classifier returns the most likely class of that data and their probability, allowing the implementation of optimization strategies.

4.2. Cyclic Analysis

Once the workload characterization has been completed as LM (live migration) or NLM (no live migration), cyclic patterns can be extracted, if any, from the characterization data collected over time. Given that possible classes are only LM or NLM, workload cycles can be identified and decomposed.

The extraction process is made by Fast Fourier Transform (FFT) [21]. FFT has $O(n \log n)$ complexity where n is the number of samples used to compute the cycle. FFT allows to convert time (or space) in frequency.

In this work we define a cycle as a recurrent pattern of a workload, which can be composed of several moments, suitable or non-suitable to live migration. Additionally, in a same cycle, there can be several compositions, as presented in Figure 3. The cycle shape depends on the initial instant t_0. If t_0

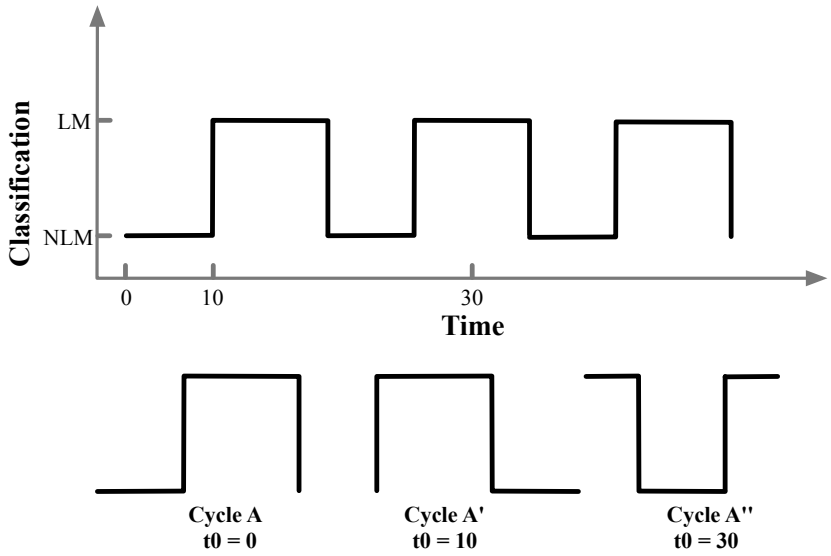


Figure 3: Shapes for the same cycle.

is in 0, the cycle's shape will be like A, if t_0 is in 10 or 30, the cycle's shape will be like A' and A", respectively.

A cycle can be simple or complex. A simple cycle is composed of up to three interleaved intervals, that is, there is a single occurrence of one type of workload (LM or NLM). In Figure 3 the cycle is simple, even in shape A", where we observe three interleaved intervals (LM, NLM and LM). In Figure 4 is presented a complex cycle example, which contains two intervals of NLM and LM. FFT can identify both types of cycles.

5. ALMA - Application Aware Live Migration Architecture

Cyclic analysis, based on workload characterization, can be used to define suitable moments to trigger live migrations. To this end, we propose an architecture, called Application-aware Live Migration Architecture (ALMA) [5], which intermediates all live migration requests from massive migration strategies and the VM monitor.

5.1. Architecture Overview

Efforts to solve macro problems in a data center, such as energy consumption, computational waste and live migration algorithm optimizations do not

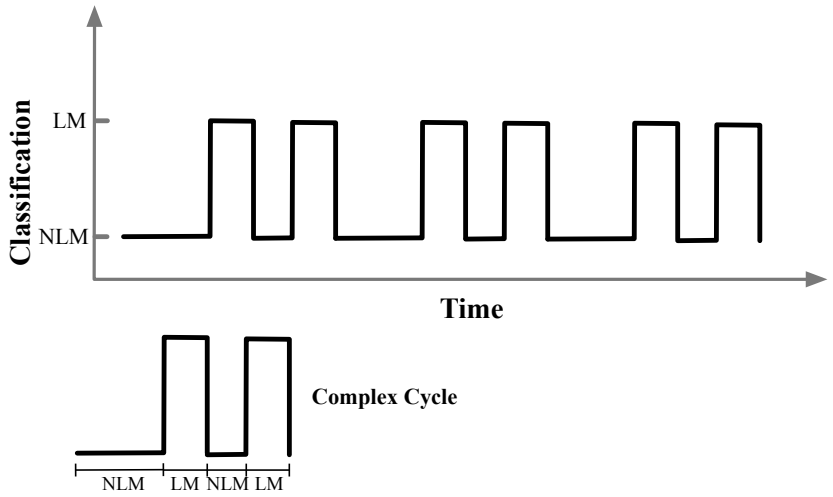
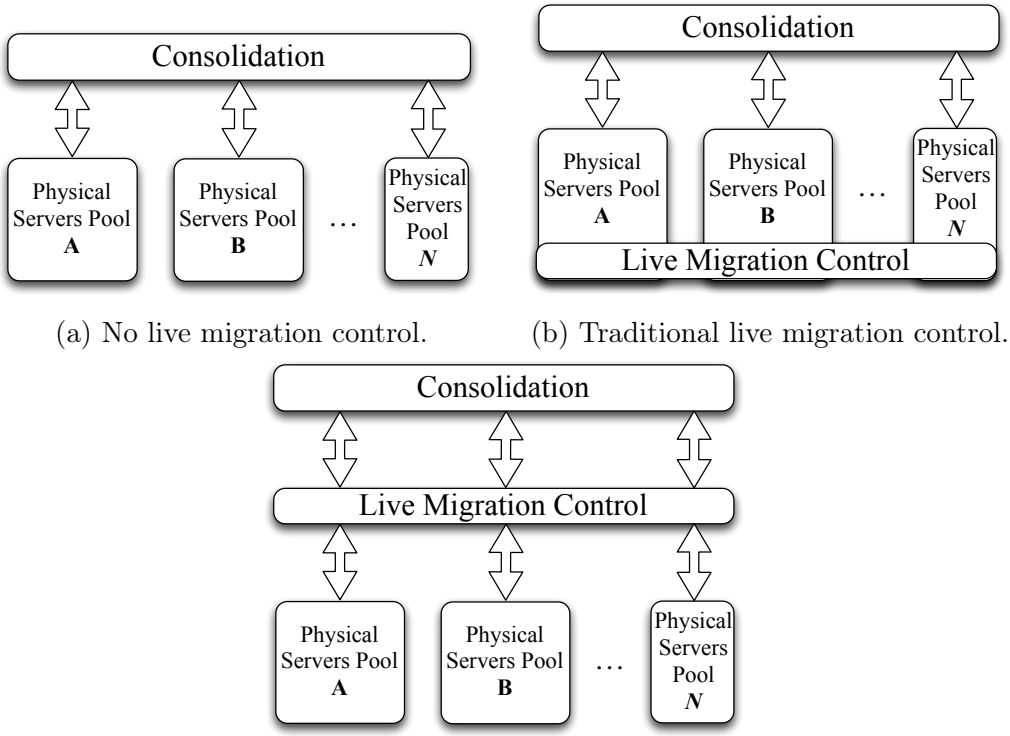


Figure 4: A complex cycle.

address problems related to pre-copy and post-copy algorithm's limitations. Our architecture avoids live migration drawbacks by choosing suitable moments to trigger this operation, which benefits strategies broadly used to solve macro problems.

Figure 5 illustrates ALMA and the other two most common architectures for live migration. Figure 5a presents an architecture with no live migration control—once a VM needs to be moved across hosts, the architecture does so without any concern about network traffic and other ongoing live migrations [22]. The architecture presented in Figure 5b implements a control over the live migrations. However, this control comes from the VM monitor and it orchestrates ongoing live migrations. The orchestration in this architecture considers only one or two metrics, such as available network bandwidth.

Our architecture, presented in Figure 5c, has a different approach. When the live migration plan is created, the architecture receives all live migration submissions and orchestrates the migrations. The main component of ALMA is the Live Migration Control Module (LMCM), which is responsible for migration scheduling based on the VM workload. It can postpone, run immediately or cancel a live migration. The postponement can occur when the VM workload is under a not suitable moment to trigger live migration and then it has to wait for a better moment. However, if the workload is suitable to be moved, ALMA triggers the migration immediately. If the workload is



(a) No live migration control.

(b) Traditional live migration control.

(c) ALMA Proposal.

Figure 5: Architectures for virtual machine live migration.

almost at the end and the live migration cost is higher than keeping the VM running on the current physical server, ALMA can cancel the live migration request.

LMCM accepts parameters from cloud service provider to impose limits, such as the maximum time a VM can wait to be migrated. This could avoid long waiting time due to long cycle periods. On the customer side, there are some parameters that could be implemented, for example the expected time to finish a given workload and with this information ALMA could avoid migrations that harm customer-defined thresholds.

5.2. Algorithms

ALMA implementation is based on two algorithms. The first one finds and extracts cyclic pattern from a workload, while the second computes the waiting time for a suitable moment for live migration. These algorithms are

complementary to one another, since the output from one is used by the other and could be implemented together. However, for a better understanding of the strategy, they will be described separately.

The first algorithm uses the workload classification collected for a given time interval, which is sorted chronologically. Each array position has a VM characterization for the period of the data collected. Moments are represented by the array index and the Fast Fourier Transform computes the cycle size (line 2) based on this array. Thereafter, all analyses occur in the interval of the array that represents the cycle size.

Algorithm 1 Cycle decomposition in two arrays.

Require: An array C with VM workload classification data for a given time interval. The array should be chronologically ordered.

```
1:  function DECOMPOSITION(C)
2:      CycleSize ← FFT(C)                          ▷ Fast Fourier Transform.
3:      LMCount ← 1
4:      NLMCount ← 1
5:      for i do1CycleSize
6:          if C[i] == LM then
7:              ArrayLM[LMCount] ← i
8:              LMCount ← LMCount + 1
9:          else
10:             ArrayNLM[NLMCount] ← i
11:             NLMCount ← NLMCount + 1
12: return ArrayNLM, ArrayLM
```

The part of the array representing an entire cycle is then split into two smaller arrays (line 5 to 13). One array stores only suitable moments for live migration (ArrayLM) and the other stores moments not favorable for live migration (ArrayNLM).

The second algorithm aims to find the instant in which the workload is inside a cycle. We use two known variables: (1) the cycle length time, which is computed in the first algorithm and (2) the workload execution time, which is the elapsed time of the workload execution. The relative time ($M_{relative}$) can be computed as the module between elapse time of the workload ($M_{current}$) and the cycle length time ($CycleSize$, line 2 of Algorithm 2).

After the computation of $M_{relative}$, the next step is to find in which array it is placed (ArrayLM or ArrayNLM). If the relative moment is placed in

ArrayNLM (line 3), we need to find out when the workload will be in suitable moment to be migrated (ArrayLM). In order to compute the remaining time of this period, we look for the first instant longer than the relative moment in ArrayLM (NextLM, in line 4). The difference between both instants (NextLM and $M_{relative}$) is the remaining time ($RemainTime$) to the workload be feasible to be migrated.

Algorithm 2 Identification of live migration moment.

Require: Cycle size and current moment.

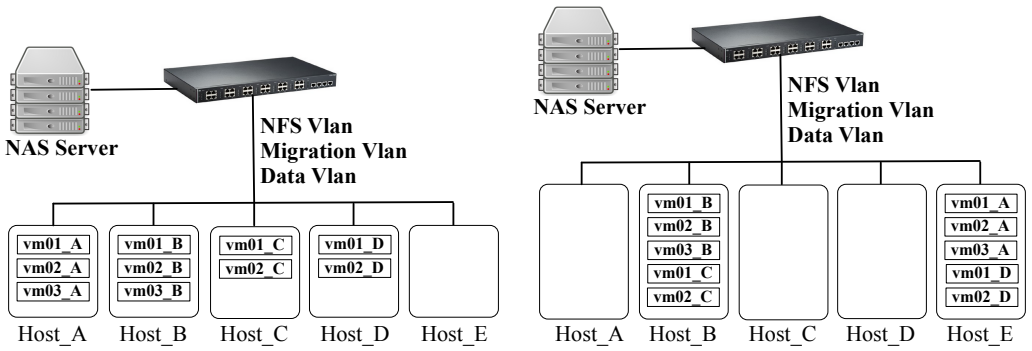
1: **function** POSTPONE($CycleSize, M_{current}$)
2: $M_{relative} \leftarrow M_{current}$ % $CycleSize$
3: **if** $find(M_{relative}, ArrayNLM)$ **then**
4: $NextLM \leftarrow findGreater(M_{relative}, ArrayLM)$
5: $RemainTime \leftarrow NextLM - M_{relative}$
6: **else**
7: $RemainTime \leftarrow 0$
8: **return** $RemainTime$

6. Evaluation

Our previous results showed substantial reduction in network data traffic and live migration time when using workload cycle recognition for live migration [5, 23]. Here we present a more detailed evaluation with more VMs, additional applications, and new insights and discussions.

Metrics evaluated in this work are:

- **Total migration time (secs):** time between the start of a migration submission and the moment the VM is released from the source host;
- **Downtime duration (secs):** time in which the migrated VM is unreachable from network. Data is collected using ICMP;
- **Network data transfer (MB):** amount of data transferred in the network during the live migration;
- **Cycle accuracy identification:** used to show moments where migration actually occurred—not when they are requested.



(a) Before consolidation. (b) After consolidation.

Figure 6: Testbed topology.

We organized our experiments in three parts: (i) workload characterization and cycle recognition analysis; (ii) orchestration analysis with benchmarks and real scientific applications; and (iii) scalability tests to handle data from hundreds of VMs.

6.1. Experiment Setup

We setup a private cloud with five physical hosts, one Network Attached Storage (NAS) and ten VMs equally distributed among physical hosts. We used VMs with three computational resource configurations (Table 1).

Table 1: Virtual Machine configurations used in testbed.

Configuration	**VCPUS**	**Memory (MB)**	**Virtual Machine** (*hostname*)
Small	1	768	vm02_A vm03_A vm01_B vm02_B
Medium	2	1024	vm01_A vm01_C vm01_D vm02_D
Large	2	2048	vm03_B vm02_C

VMs were initially placed in four physical hosts and during the workload execution, they were consolidated into two physical hosts (Figure 6). The consolidation moments were randomly chosen to explore various points in time, having preferences for points where all machines were running workloads in order to stress the consolidation policies.

Table 2: Benchmarks used in testbeds.

Benchmark / Application	**Description**	**Experiment**
TeraSort (MapReduce)	Sort algorithm which uses MapReduce paradigm [24].	Scientific Application Evaluation
BRAMS	Brazilian atmospheric model used for weather forecast [25].	Scientific Application Evaluation
OpenModeller	Scientific application used to mode specimen distribution [26].	Scientific Application and Characterization Evaluation
SPEC CPU 2k	It is a broadly used benchmark to compare computational systems. It has several subprograms which stress the processor [27].	Benchmark and Characterization Evaluation
BT	Part of NASA Parallel Benchmark. This program has a memory footprint of 650 MB with high rate of dirty page [28].	Benchmark Evaluation
IOZONE	Benchmark with high usage of I/O subsystem. To avoid cache usage effect, files are larger than available memory [29].	Benchmark Evaluation
sleep[1]	Linux command used to delay processing for a given time.	Benchmark Evaluation
LAME	LAME is an MP3 codifier used as benchmark [30]. We used input file of 2.3 GB.	Characterization Evaluation

Benchmarks and applications (Table 2) were used to evaluate the characterization strategy and ALMA. OpenModeller, LAME, and SPEC were used with different VM configurations for workload characterization. The ALMA evaluation consists of two experimental scenarios in order to create a controlled scenario. In first scenario we created artificial cycles running benchmarks with a specific behavior in a given order. The evaluation contains the SPEC benchmark as CPU intensive workload, BT as Memory intensive workload, IOZone as I/O intensive workload and, `sleep` command to simulate IDLE periods. The artificial cycles and VMs used are described in Table 3. The second scenario contains BRAMS, OpenModeller, and TeraSort where the former two are scientific applications and the latter represents a typical data-intensive cloud workload, all running simultaneously in different VMs.

[1]Available at: `http://man7.org/linux/man-pages/man3/sleep.3.html`

Table 3: Artificial cycles used to evaluate ALMA.

Virtual Machine	Artificial Cycles
vm03_A	I/O CPU CPU I/O CPU CPU I/O CPU CPU
vm02_C	MEM IDLE CPU MEM IDLE CPU MEM IDLE CPU MEM IDLE CPU
vm02_A	MEM CPU CPU MEM CPU CPU MEM CPU CPU MEM CPU CPU
vm01_C	MEM IDLE CPU MEM IDLE CPU

6.2. Workload Classification and Cycle Recognition

The evaluation of the NB classifier was based on two benchmarks and one scientific application. Benchmarks behavior is more constant during the execution and, due to this characteristic, we can verify the NB classifier precision. On the other hand, the scientific application presents several oscillations during its execution and enables the analysis of the classifier sensibility to peaks of usage and workload changes.

A new subclass of four VM configurations, summarized in Table 4, was used for this experiment. For each configuration, benchmarks (SPEC and LAME) and an application (OpenModeller) were run ten times and during the test load indexes were collected. VMs were installed in a hardware consisting of a 2.66 MHz Intel Core 2 Quad processor, 2 GB of memory, and a 5400 RPM hard disk with nominal throughput of 3 GB/second. As software configuration, the VMM used was Xen 4.1.3 and OpenSuse 12.1 running Kernel 3.1.10 as host OS. All VM images use CentOS 5.9 running Kernel 2.6.18.

All results are summarized in Table 5, which also presents resource usage average, the standard deviation between parenthesis and Naive Bayes classification in last column. Since classification is done during the benchmark and application execution, it is expected to observe some classification oscillation, which we then captured as primary and secondary workload.

In the SPEC characterization, running in configurations C1 and C2, with only one processor available, NB classified as CPU intensive workload, with memory or I/O fluctuations during the benchmark execution. Operations of I/O occurred when SPEC wrote statistics in control files, such as FLOPS and elapsed time. Memory classification was also expected, since the MCF has a high memory usage profile.

For the LAME benchmark, the workload profile is CPU and I/O intensive

Table 4: Virtual Machine configurations.

Configuration ID	Processor (VCPUs)	Memory (GB)
C1	1	1
C2		2
C3	2	1
C4		2

Table 5: Naive Bayes classification summary.

Average Resource Usage				Naive Bayes Characterization	
Benchmark/Application	**Conf. ID**	**CPU (%)**	**MEM (%)**	**CPU (Prim/Sec)**	**MEM (Prim/Sec)**
SPEC	**C1**	96 (±18)	17 (±5)	CPU+I/O	CPU+MEM
	C2	96 (±18)	9 (±2)	CPU+I/O	MEM
	C3	49 (±12)	17 (±5)	IO	I/O+MEM
	C4	49 (±11)	10 (±3)	I/O+MEM	I/O
LAME	**C1**	98 (±15)	7 (±1)	CPU+I/O	I/O+CPU
	C2	98 (±15)	5 (±1)	CPU+I/O	I/O
	C3	50 (±11)	7 (±1)	I/O	I/O
	C4	51 (±11)	5 (±1)	I/O	I/O
OpenModeller	**C1**	100 (±5)	15 (±1)	CPU+I/O	I/O
	C2	99 (±11)	8 (±1)	CPU+I/O	I/O
	C3	51 (±11)	15 (±1)	I/O+MEM	I/O
	C4	51 (±11)	9 (±1)	I/O+MEM	I/O

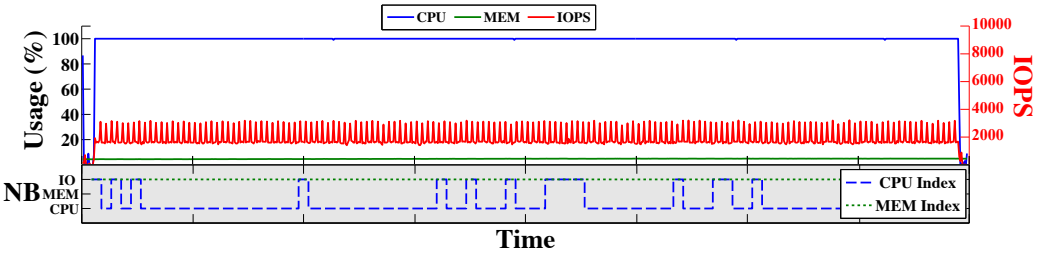


Figure 7: Characterization over time in configuration C3 running LAME.

usage. While an input file is being processed, LAME creates the MP3 file, resulting in high I/O operations, with simultaneous read and write operations. Figure 7 presents the characterization over time for configuration C3 running the workload LAME. The top of the graph contains the VM resource usage (CPU, memory, and I/O per second) and the bottom represents how NB characterized the workload at a given moment.

OpenModeller has a high CPU usage, with some memory access and I/O operations during the application initialization, when it reads the input file and during the finalization, when the benchmark writes the output file. Characterization for C1 and C2 configuration was CPU intensive, which is expected in configurations with only one processor available. The processor usage is more evident for all benchmarks/application for configurations C1 and C2 due to the availability of one processor. When adding a second processor, NB identifies other workload profiles.

The NB's asymptotic complexity is linear which, as previously discussed, is necessary for any characterization strategy in cloud computing. Considering the discretization steps and the probability computation, the complexity is $\Theta(n + k)$, where k is the number of indexes to be discretized and n is the number of classes to be evaluated.

By using the NB characterization we identify when a VM can be moved across physical hosts. Using NB characterization, we can identify the primary workload and, instead of usual classification as CPU, MEM, I/O or IDLE, it is classified as suitable to LM or non-suitable to LM (NLM).

6.3. Orchestration Analysis

This experiment evaluates the orchestration considering suitable moments to trigger live migration. A live migration representation graph illustrates the workload behavior over time and the moments when live consolidation were submitted and the instant when live migration actually occurred.

6.3.1. Benchmark Experiments

Figure 8 presents the migration diagram for the four VMs running benchmarks. Line in blue is the workload behavior over time, where valleys are periods not suitable to trigger live migration (NLM) and peaks are periods where the workloads are suitable to live migration (LM). The workload is executed at the same time across all VMs. Dashed lines in red represent consolidation instants. Lines in black are instants where ALMA actually triggered live migration.

In order to compare the consolidation strategy under control of our architecture and without any surveillance we run two sets of experiments. In the first set, VMs were actually consolidated in instants represented in dashed red lines and we left the workload run to the end. During the second set, our architecture was in place and according to the workload and the cycle analy-

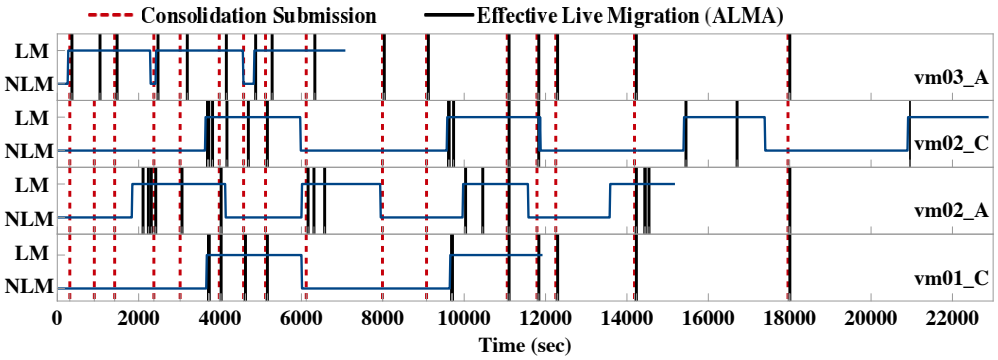


Figure 8: Cycle accuracy identification diagram for benchmarks.

Table 6: Results with four VMs running benchmarks.

Metric	Virtual Machine	Traditional Consolidation	ALMA	Reduction (%)
Downtime (sec)	vm03_A	20.06	20.44	-1.87
	vm02_C	18.63	17.75	4.70
	vm02_A	20.75	23.69	-14.16
	vm01_C	19.25	18.94	1.62
Live Migration Time (sec)	vm03_A	28.81	12.00	58.35
	vm02_C	87.56	42.31	51.68
	vm02_A	43.81	11.13	74.61
	vm01_C	54.31	26.81	50.63
Data Traffic (MB)		11,557.50	9,159.60	21.56

sis, it triggered or postponed live migration to a better instant (represented by the black lines in the figure).

Ideally, when ALMA is in place, live migration (lines in black) should be triggered during the peaks. In this experiment, our architecture was able to migrate VMs at suitable workload moments, thus reducing data transferred and live migration time (Table 6). The reduction in live migration time was up to 74% (vm02_A) and data traffic reduction was up to 21%, representing a reduction of about 2.3 GB. For the downtime metric, with 95% of confidence, it is not possible to infer any improvements when using ALMA or not.

6.3.2. Application Experiments

We used two scientific applications, BRAMS and OpenModeller running in vm02_C and vm03_A, respectively and a typical cloud workload, represented by Hadoop cluster, running in vm01_B, vm02_C and vm01_C. The vm01_B VM was not moved from the physical host because it already was in one of the physical hosts in which the workload was consolidated. That is the reason why metrics of this VM were suppressed from presented results.

Figure 9 shows long suitable periods to live migration, such as the one of the vm01_C. There are also workloads with long periods not suitable to live migration, like vm03_A's workload, and complex cycles as observed in vm02_C. However, even in this scenario, ALMA was able to identify and successfully postpone the live migration to a suitable moment.

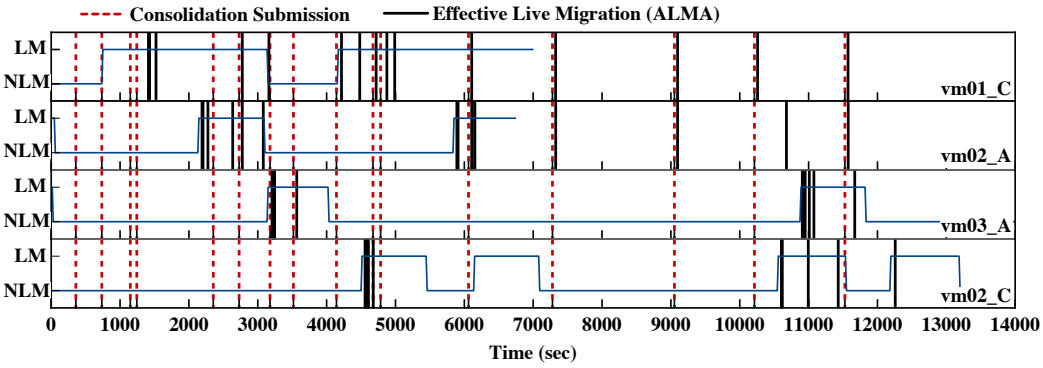


Figure 9: Cycle accuracy identification diagram for applications.

The ALMA accuracy leads to the results presented in Table 7. Reduction in live migration time (up to 67%) and amount of data transferred in network (up to 62%) were significant. This result is due to the Hadoop cluster behavior, which exchanges larges amount of data across the cluster nodes. This application, particularly, was benefited by our architecture, as observed in Table 7.

The application behavior is not known *a priori* (as opposed to the benchmarks evaluation) and it is sensitive to the initial setup, such as command line parameters and the input data. In both experiments, downtime did not show improvements or deterioration. Statistically, with 95% of confidence, it is not possible to determine if using ALMA or not can achieve performance improvements.

The explanation for this is in the TCP behavior and how the hypervisor

Table 7: Results with four VMs running real applications.

Metric	Virtual Machine	Traditional Consolidation	ALMA	Reduction (%)
Downtime (sec)	vm03_A (OpenModeller)	21.80	23.00	-5.50
	vm02_C (BRAMS)	22.60	20.73	8.26
	vm01_C (Hadoop)	19.07	22.33	-17.13
	vm02_A (Hadoop)	12.67	17.20	-35.79
Live Migration Time (sec)	vm03_A (OpenModeller)	31.27	12.73	59.28
	vm02_C (BRAMS)	12.93	10.60	18.04
	vm01_C (Hadoop)	39.20	18.67	52.38
	vm02_A (Hadoop)	38.20	12.40	67.54
	Data Traffic (MB)	14,566.47	5,504.98	62.21

contacts the network devices that a given IP address is hosted by a given physical host. Once a VM is moved across physical hosts, the hypervisor needs to update the ARP table and, to this end, it sends an ICMP packet to the network gateway. This process is not part of the migration algorithm; it is an independent process. Moreover, downtime is sensitive to TCP. Our results corroborate observations from Kikuchi and Matsumoto [10]: when a packet does not arrive to the destiny, sender will try again when retransmission time out (RTO) ends. The RTO is computed using round time trip (RTT) which is, initially, equal to three seconds. Every time a retransmission is needed, RTO value is doubled, increasing downtime.

6.4. Scalability

Scalability is a factor to be considered when dealing with cloud computing environments. Among the main features of such environment we can cite resource elasticity, which enables the user to increase or decrease the amount of available computational resources in response to a given demand [16]. Also, users could add or remove VMs to their environment. These features make cloud computing environments sensitive to solutions with low scalability characteristic.

In this section we present an analysis of scalability of our solution. Despite the fact that previous experiments were run in a real private cloud, performing a scalability test would require a large testbed, which we do not have access to. Therefore, we used traces of the previous executed benchmark experiments to measure the overhead caused by ALMA, specifically the Live Migration Control Module (LMCM) that performs the classification and cycle analysis.

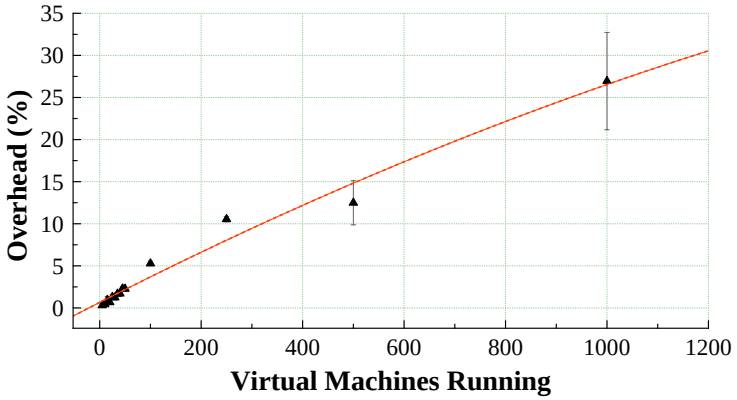


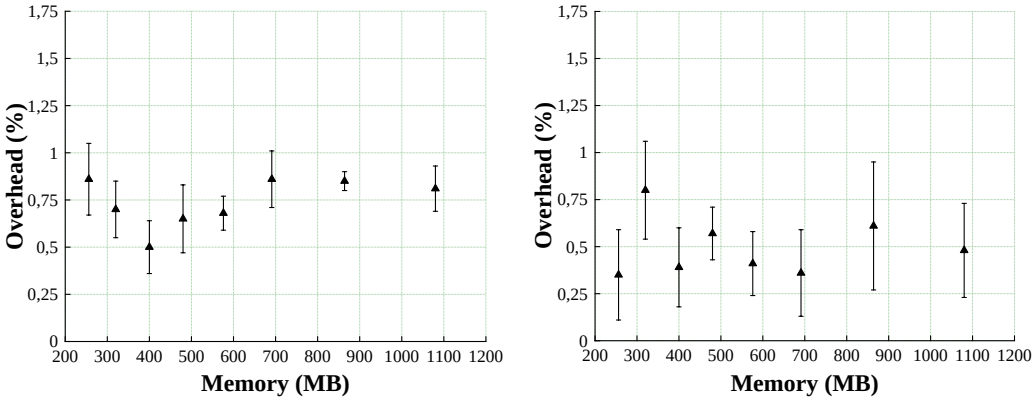
Figure 10: Overhead caused by LMCM with data from up to thousand VMs.

The baseline used to infer the overhead is the Linux kernel compilation with no other processes competing for resources. Next, traces were submitted to the LMCM and the amount of extra time to perform the same compilation was considered as the overhead. For each VM a new process was created inside LMCM and we started the analysis with five VMs and gradually increased to up to thousand VMs. Figure 10 shows the results from ten-run experiments which highlight the fact that the overhead has a linear tendency (line in red) that increases proportionally with the increase of VMs. The overhead has increased, in average, 0.21% for every five VMs added. According to this result, and in this configuration testbed, LMCM saturation would be achieved with 1,800 VMs running and submitting the live migration request at the same time. We considered the saturation when the overhead caused by LMCM reached 100% when compiling the linux kernel.

6.4.1. Data Gathering Overhead in Virtual Machines

The overhead imposed by index data gathering in VMs needs to be considered. As mentioned earlier, data indexes were collected by SNMP version 2. For each SNMP request, a script is executed, which returns a given value according to the index requested. The overhead evaluation conducted in this section is similar to the evaluation that has been conducted to infer the LMCM overhead. We changed the VM configuration during the experiment (processor and memory) and our main objective is to observe and quantify the overhead caused by index data gathering according to the amount of available computational resource.

The first experiment set was conducted using one VM with one processor



(a) One processor. (b) Two processors.

Figure 11: Overhead in virtual machine.

and memory increased from 256 MB to 1,080 MB. Results are presented in Figure 11 and indicate the overhead is about 0.75% and 0.5% for VMs with one and two processors, respectively. The overhead is constant with small fluctuation when varying the memory size. However, it is more sensitive to the available processors.

7. Related Work

The main research efforts on live migration overhead for cloud environments are related to VM workload consolidation techniques [31, 32, 13]. The motivation comes from the substantial cost reduction for service providers that can be achieved using optimized consolidation techniques. Due to the different aspects of existing efforts in this area, we split them into two categories. The first one is on live migration overhead as a factor to trigger live migration. The efforts in second category present strategies to control the live migration to avoid network congestion caused by massive migrations.

7.1. Live Migration with Overhead Constraints

As discussed earlier, consolidation deals with the selection and transfer of VMs to a common physical host in order to reduce power consumption of the source physical hosts. Since this is an NP-hard problem, several solutions based on heuristics are available in literature. Resource reservation was also

considered as a mechanism to optimize migrations [32]. However, most of solutions do not consider live migration overhead in infrastructure and VMs collocated.

Xu et al. [13] present a strategy that takes into account the computational cost of live migration in physical hosts (source and target) that are involved in the migration process and in collocated VMs. This strategy, called iAware, aims to avoid service level agreements violation, preventing that a given live migration harms other VMs. Authors also present a strategy to model the impact of live migration in collocated VMs, which is based on available physical resources and the amount of interruptions generated by VMs. Similar to our work, iAware can be embedded into existing consolidation or load balance strategies. Furthermore, both strategies aim to reduce performance degradation caused by live migration.

Despite the features in common, iAware does not consider the VM workload as it only relies on resource usage of the physical host. Live migration itself has a computational cost, therefore, postponing it to more suitable moment according to the VM workload can reduce overhead in physical host.

Verma et al. [33] present CosMig, which is a model for live migrations, including time estimation to perform them. CosMig is based on processor and memory usage parameters and determines the live migration impact of a VM. Verma et al. also identified that: (1) an effective live migration model must take into account application behavior, (2) only live migration does not improve application performance; other factors can promote performance improvement such as target host computing power and VM memory fragmentation. The main similarity of CosMig and our work is the evaluation of live migration in VM workload. Despite the fact that metrics used to model live migration impact are different, both studies present models to infer live migration impact in workload. A fundamental difference of is how information about live migration impact is used by the proposed strategy. In CosMig, the question asked is related to *"if"* live migration of a given VM will lead to performance gains or not. On the other hand, ALMA asks *"when"* a live migration can be performed in order to avoid infrastructure damage and, consequently, in application.

Finally, Stage and Setzer [34] introduce a live migration scheduling strategy that classifies migrations according to the current workload and identifies the minimal network resources to perform a migration. According to the authors, a migration of a single VM can consume significant network bandwidth during a long period (about 500 Mb/s for ten seconds to migrate a VM run-

ning a web server). The architecture presented by Stage and Setzer has similarities with our work. Like in ALMA, there is a live migration scheduler management, which decides when a VM can be migrated. However, their architecture only observes network parameters (available bandwidth and a live migration time constraint). Also, there is a workload classifier based on the following attributes: (i) predictable: workload is considered predictable when its behavior has a reliable prognosis for a given period; (ii) tendentious: refers to fluctuations of a tendency; and (iii) cyclic: indicates how often a pattern occurs in a given workload. The main difference from our work is that, in Stage and Setzer work, live migration will take place according to the network bandwidth consumption. From the estimative based on workload type and live migration duration threshold, which can be defined by user or service provider, the architecture schedules live migrations in order to meet live migration duration threshold. In addition, the characterization of Stage and Setzer aims to group VMs with similar workload and performs the live migration of these groups. In ALMA, the workload characterization is the main criterion to define the suitable moment to trigger live migrations.

7.2. Live Migration Control Strategies

Beloglazov and Buyya [35] propose a dynamic strategy for VM consolidation that considers suitable moments to perform live migrations. Their goal is to minimize power consumption and maximize quality of service (QoS) delivered by service provider which, according to the authors, composes the trade-off between energy and performance. Their strategy identifies physical hosts overloaded and live migrations intervals are defined in order to keep QoS. In our work, ALMA can postpone a live migration according to the VM workload and in Beloglazov and Buyya study, live migrations can be postponed according to the physical host workload.

Ye et al. [36] present a framework, called VC-Migration, which controls live migrations in a cluster composed of VMs. The VC-Migration has strategies previously configured which decides how many VMs (granularity) will be considered for migration in a given moment. The decision is based on current computational resource usage of physical hosts. The strategies defined by the framework are:

- **Concurrent migration:** this strategy performs the live migration of several VMs, simultaneously, running in the same cluster;

- **Mutual migration:** strategy which is applied when physical hosts involved in live migration process have VMs moved between each other;
- **Homogeneous migration in multi-cluster:** strategy applied when several virtual clusters, with the same number of VMs, are being migrated;
- **Heterogeneous migration in multi-cluster:** same strategy as homogeneous migration, but virtual clusters have different sizes.

The framework chooses the best strategy according to the number of VMs being migrated and network bandwidth consumption. Authors argue that application interdependence, which is common in a cluster environment, reduces the infrastructure impact for network and applications.

8. Concluding Remarks

Live migration algorithms are known to be sensitive to memory usage. However, during an application execution these algorithms can present periods of high memory usage or high processor usage. These periods can float according to the day of the week, period of the year, or even with application input. Therefore, the challenge is to identify workloads with cyclic pattern and, once the cycle is identified, how to postpone live migrations to reduce their overhead.

We proposed and evaluated a migration strategy and architecture using a private cloud running benchmarks and real applications. The architecture was able to reduce up to 74% and 62% in live migration time and data traffic, respectively. The scalability analysis showed a host with 6 GB of memory is capable of handling data of up to 1,800 VMs. Based on evaluation results, our main finding is that using workload cycle recognition it is possible to choose suitable moments to trigger live migration, thus leading to a significant reduction in migration time and data traffic, confirming our hypothesis.

References

[1] D. Parkhill, The Challenge of the Computer Utility, no. p. 246 in The Challenge of the Computer Utility, Addison-Wesley Publishing Company, 1966.

[2] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in: Proceedings of the Conference on Power Aware Computing and Systems, HotPower'08, USENIX Association, 2008.

[3] K. Nuaimi, N. Mohamed, M. Nuaimi, J. Al-Jaroodi, A survey of load balancing in cloud computing: Challenges and algorithms, in: Proceedings of the Second Symposium on Network Cloud Computing and Applications (NCCA'12), 2012.

[4] A. Roytman, A. Kansal, S. Govindan, J. Liu, S. Nath, PACMan: Performance Aware Virtual Machine Consolidation, in: Proceedings of the 10th International Conference on Autonomic Computing (ICAC'13), USENIX, 2013.

[5] A. Baruchi, E. Midorikawa, M. A. S. Netto, Improving virtual machine live migration via application-level workload analysis, in: Proceedings of the 10th International Conference on Network and Service Management (CNSM'14), 2014.

[6] M. M. Theimer, K. A. Lantz, D. R. Cheriton, Preemptable remote execution facilities for the v-system, SIGOPS Oper. Syst. Rev. 19 (5) (1985) 2–12.

[7] M. R. Hines, K. Gopalan, Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning, in: Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '09), ACM, 2009.

[8] R. Birke, A. Podzimek, L. Chen, E. Smirni, State-of-the-practice in data center virtualization: Toward a better understanding of VM usage, in: Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'13), 2013.

[9] J. Smith, R. Nair, Virtual Machines: Versatile Platforms for Systems and Processes (The Morgan Kaufmann Series in Computer Architecture and Design), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[10] S. Kikuchi, Y. Matsumoto, Impact of live migration on multi-tier application performance in clouds, in: Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD'12), 2012.

[11] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, SIGOPS Oper. Syst. Rev. 37 (5) (2003) 164–177.

[12] A. Strunk, Costs of virtual machine live migration: A survey, in: Proceedings of the Eighth IEEE World Congress on Services (SERVICES'12), 2012.

[13] F. Xu, F. Liu, L. Liu, H. Jin, B. Li, B. Li, iAware: Making Live Migration of Virtual Machines Interference-Aware in the Cloud, IEEE Transactions on Computers 99 (2013) 1.

[14] T. C. Ferreto, M. A. S. Netto, R. N. Calheiros, C. A. F. De Rose, Server consolidation with migration control for virtualized data centers, Future Generation Computer Systems 27 (8) (2011) 1027–1034.

[15] E. Feller, C. Morin, A. Esnault, A case for fully decentralized dynamic VM consolidation in clouds, in: Proceedings of the IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom'12), 2012.

[16] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, Above the clouds: A berkeley view of cloud computing, Tech. rep., EECS Department, University of California, Berkeley (Feb 2009).

[17] T. Garfinkel, M. Rosenblum, When virtual is harder than real: Security challenges in virtual machine based computing environments, in: Proceedings of the 10th Conference on Hot Topics in Operating Systems (HoTOS'05), USENIX Association, 2005.

[18] I. Ahmad, Easy and Efficient Disk I/O Workload Characterization in VMware ESX Server, in: Proceedings of the 2007 IEEE 10th International Symposium on Workload Characterization (IISWC '07), IEEE Computer Society, 2007.

[19] J. Du, N. Sehrawat, W. Zwaenepoel, Performance profiling of virtual machines, SIGPLAN Not. 46 (7) (2011) 3–14.

[20] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd Edition, Prentice Hall Press, Upper Saddle River, NJ, USA, 2009.

[21] G. D. Bergland, Numerical analysis: A fast fourier transform algorithm for real-valued series, Communications of the ACM 11 (10) (1968) 703–710.

[22] M. Seki, Y. Koizumi, H. Ohsaki, K. Hato, J. Murayama, M. Imase, Selfish virtual machine live migration causes network instability, in: Proceedings of the 9th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT'12), 2012.

[23] A. Baruchi, E. Toshimi Midorikawa, L. Matsumoto Sato, Reducing virtual machine live migration overhead via workload analysis, Latin America Transactions, IEEE (Revista IEEE America Latina) 13 (4) (2015) 1178–1186.

[24] J. Dean, S. Ghemawat, Mapreduce: Simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.

[25] S. R. Freitas, K. M. Longo, M. A. F. Silva Dias, R. Chatfield, P. Silva Dias, P. Artaxo, M. O. Andreae, G. Grell, L. F. Rodrigues, A. Fazenda, J. Panetta, The coupled aerosol and tracer transport model to the brazilian developments on the regional atmospheric modeling system (catt-brams), Atmospheric Chemistry and Physics 9 (8) (2009) 2843–2861.

[26] M. E. de Souza Muñoz, R. D. Giovanni, M. F. de Siqueira, T. Sutton, P. Brewer, R. S. Pereira, D. A. L. Canhos, V. P. Canhos, openModeller: a generic approach to species' potential distribution modelling, GeoInformatica 15 (1).

[27] S. Sair, M. Charney, Memory Behavior of the SPEC2000 Benchmark Suite, Tech. rep. (2000).

[28] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, S. K. Weeratunga, The nas parallel benchmarks - summary and preliminary results, in: Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing (SC'91), ACM, 1991.

[29] V. Tarasov, S. Bhanage, E. Zadok, M. Seltzer, Benchmarking file system benchmarking: It *is* rocket science, in: Proceedings of the 13th

USENIX Conference on Hot Topics in Operating Systems (HotOS'13), USENIX Association, 2011.

[30] J. Johnston, A. Ferreira, Sum-difference stereo transform coding, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'92), 1992.

[31] K. Ye, Z. Wu, C. Wang, B. B. Zhou, W. Si, X. Jiang, A. Y. Zomaya, Profiling-based workload consolidation and migration in virtualized data centers, IEEE Transactions on Parallel and Distributed Systems 26 (3) (2015) 878–890.

[32] K. Ye, X. Jiang, D. Huang, J. Chen, B. Wang, Live migration of multiple virtual machines with resource reservation in cloud computing environments, in: Proceedings of the International Conference on Cloud Computing (CLOUD11), IEEE, 2011.

[33] A. Verma, G. Kumar, R. Koller, A. Sen, CosMig: Modeling the Impact of Reconfiguration in a Cloud, booktitle = Proceedings of the 2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '11), IEEE Computer Society, 2011.

[34] A. Stage, T. Setzer, Network-aware migration control and scheduling of differentiated virtual machine workloads, in: Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing (CLOUD'09), 2009.

[35] A. Beloglazov, R. Buyya, Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints, IEEE Transactions on Parallel and Distributed Systems 24 (7) (2013) 1366–1379.

[36] K. Ye, X. Jiang, R. Ma, F. Yan, VC-Migration: Live Migration of Virtual Clusters in the Cloud, in: Proceedings of the ACM/IEEE 13th International Conference on Grid Computing (GRID'12), IEEE Computer Society, 2012.