Resilient and Adaptive Networked Systems

Mauro Andreolini*, Sara Casolari°, Marcello Pietri°, Stefania Tosi°

**Department of Mathematical, Physical and Natural Sciences*

°Department of Engineering "Enzo Ferrari"

University of Modena and Reggio Emilia, Modena, Italy

mauro.andreolini@unimore.it, sara.casolari@unimore.it, marcello.pietri@unimore.it, stefania.tosi@unimore.it

Resilient and Adaptive Networked Systems

Modern system infrastructures must accommodate continuously changing demands for different types of workloads and time constraints. In a similar context, adaptive management of virtualized application environments among networked systems is becoming one of the most important strategies to guarantee resilience and performance of available computing resources. This chapter analyzes management algorithms that decide in an adaptive way the transparent reallocation of live sessions of virtual machines in large numbers of networked hosts. We discuss the main challenges and solutions related to the adaptive activation of the migration process, the number and location of virtual machines to migrate.

Keywords: *adaptive models*, *live migration*, *virtualization*

1. Introduction

Most data centers were characterized by high operating costs, inefficiencies, and a multitude of distributed, heterogeneous servers that added complexity in terms of resilience and management. In the last years, enterprises consolidated their systems through virtualization solutions in order to improve data center efficiency and offer the benefit of performance and fault isolation, flexible migration, resource consolidation, and easy creation of specialized environments (Ibrahim et al. 2011). Logically pooling all system resources and centralizing resource management allows the increment of the overall node utilization while lowering management costs.

In this scenario, *live migration* of virtual machines is one of the main building blocks to guarantee resilience and high utilization of networked resources. Live migration is defined as the transfer of running virtual machine instances from one physical server to another with little or zero downtime and without interrupting virtualized services (Chen et al. 2011). There are multiple benefits of live migration among which fault tolerance, load balancing, and consolidation. For instance, to avoid failover of the virtual

machines, it is necessary to live migrate one or more guest running on one physical server to another physical server that guarantees continued and uninterrupted service. Live migration has been supported by the vast majority of hypervisors, such as VMware, Xen, KVM and VirtualBox (Shetty et al. 2012). Moreover, some recent versions (e.g., VMware VMotion) support adaptive migration, although supported by static threshold-based activation mechanisms. This is a step ahead, but we state that full resilience and high resource utilization require the research for *adaptive* management algorithms and supports that are able to continuously tune their behavior as a function of changing operating conditions. Adaptive capacity management requires continuous monitoring services and runtime decision algorithms for deciding when a physical host should migrate a portion of its load, which portion of the load should be moved and where. These problems and solutions represent the focus of this chapter.

The chapter is organized as follows. Section 2 discusses the virtual machine migration process. In Section 3 presents a taxonomy of live migration strategies. Section 4 analyzes a case study showing the differences between a *static* and an *adaptive* selection strategy. Conclusions are drawn in Section 5.

2. Migration of virtual machines

A typical networked architecture consists of a huge set of physical machines (*hosts*), each of them equipped with some virtualization mechanisms, from hardware virtualization up to micro-partitioning, operating system virtualization, software virtualization. These mechanisms allow each machine to host a concurrent execution of several virtual machines (*guest*) each with its own operating system and applications.

To accommodate varying demands for different types of processing, the most modern infrastructures, such as clouds, include adaptive management capabilities and virtual

machine mobility that is, the ability to move transparently virtual machines from one host to another. In this scenario, the decision algorithm orchestrating the live migrations has to select one or more *sender* hosts from which some virtual machines are moved to other destination hosts, namely *receivers*. By migrating a guest from an overloaded host to an unloaded one, it is possible to improve resource utilization and resilience.

The migration algorithms defines three sets: sender hosts, receiver hosts, and migrating guests, where their cardinalities are denoted as S, R, and G, respectively. Let also N be the total number of hosts. The algorithm has to guarantee that $N \geq S + R$, and that the intersection between the set of sender hosts and of receiver hosts is null (Andreolini et al. 2009).

Every virtual machine migration approach shares a common management model made up of four distinct phases, outlined in Figure 1.

Selection of sender hosts. The first action requires the selection of the set of sender hosts that require the migration of some of their guests. The idea is to have a migration algorithm so that the cardinality S of the set of senders is much smaller than the total number of hosts that is, $S << N$.

Selection of guests. Once selected the senders, we have to evaluate how many and which guests it is convenient to migrate. Even for this phase, the goal is to limit the number of guests for each host that should migrate, so that $G < (N - S)$. If this does not occur after the first evaluation, the guest selection can proceed iteratively until the constraint is satisfied. (It is worth to observe that in our experiments, no instance required an iteration.)

Selection of receiver hosts. Once selected the guests that have to migrate, we have to define the set of receiver hosts. In these networked infrastructures, the major risk we

want to avoid is a dynamic migration that tends to overload some receiver hosts so that at the successive checkpoint a receiver may become a sender, and so on. Similar fluctuations devastate performance and resilience.

Assignment of guests. The guests selected for migration are assigned to the receivers through a management algorithm aiming to satisfy some architectural and/or application constraints. For example, a greedy algorithm may begin to assign the most onerous guests to the lowest loaded hosts, and so on until the sender list is completed. Many other possibilities exist.

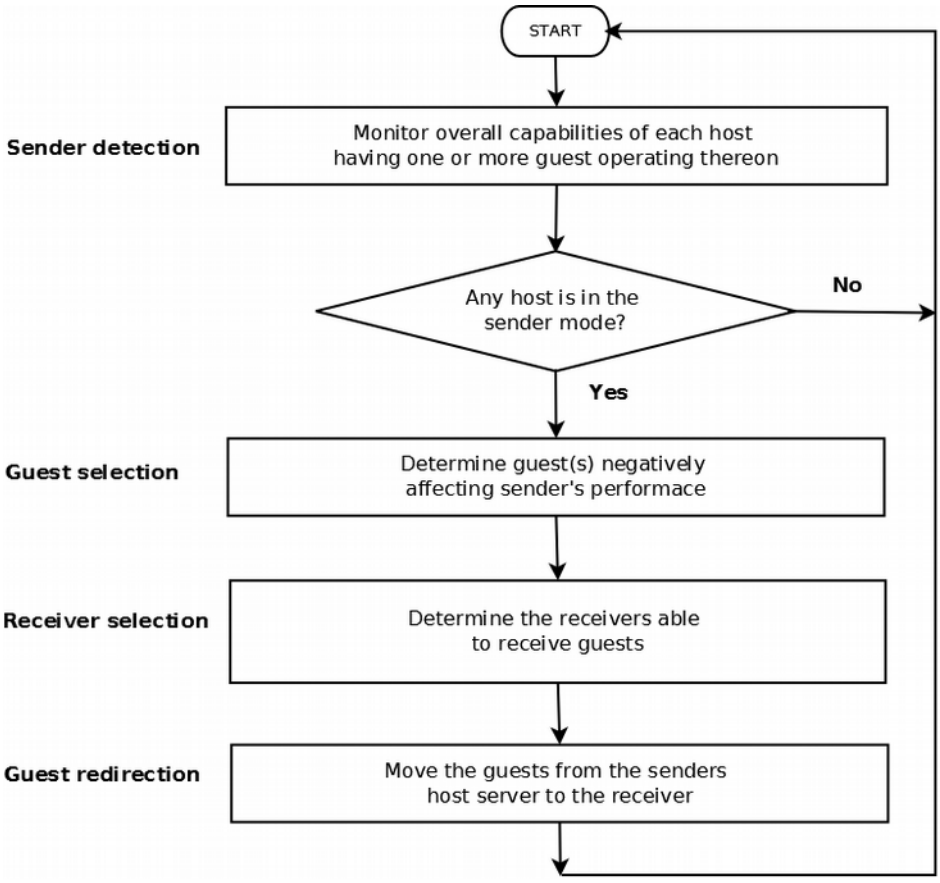


Figure 1: Flow diagram of the virtual machine migration process.

3. A taxonomy of live migration strategies

Existing approaches to live migration of virtual machines can be broadly classified according to different decisions intervening during the four phases of the migration process. These decisions, shown in Figure 2, include the *activation strategy*, *monitoring*, *component selection*, *destination* and *migration mechanism*.

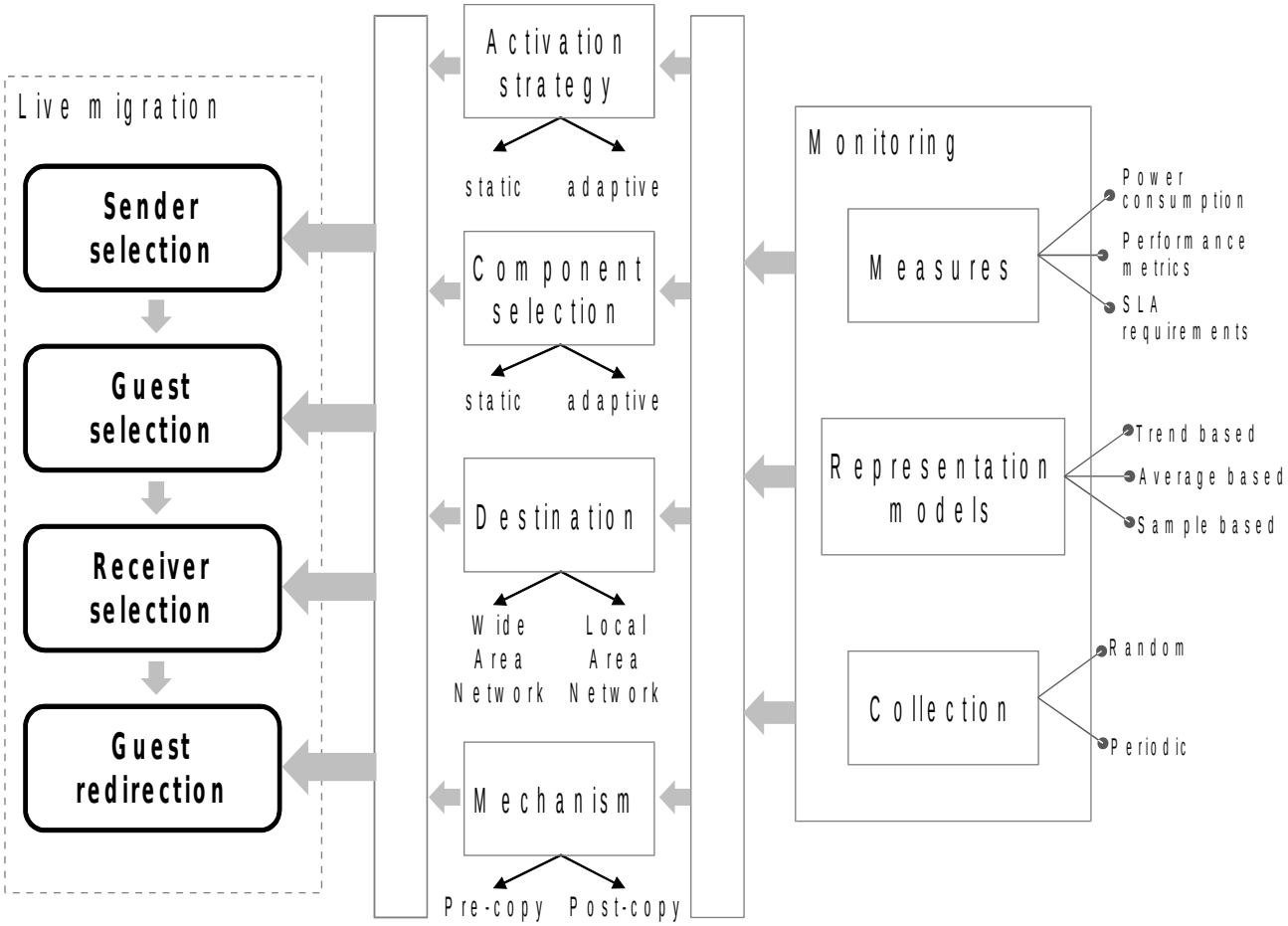


Figure 2. Taxonomy of live migration strategies.

Monitoring. The monitoring system used throughout all the operations is one of the most important component of the live migration mechanism. The first decision regards the performance indicators (*measures*) used to evaluate the load conditions of hosts and virtual machines. Different choices are possible, according to the goal of the migration.

- *Performance*: throughput, utilization of the most relevant hardware resource components (CPU, disk, memory, network) as in (Stage et al. 2009, Gerofi et al. 2010).

- *Power consumption*: CPU voltage at the different power states (Jung et al. 2010).

- *Reliability*: ping response times, service response probes as HTTP response times (Stage et al. 2009).

The next decision regards the *collection* of individual samples through the definition of an appropriate sampling interval. The collection may be periodic (e.g., every second, every minute), or random. This decision may influence the statistical properties of the collected measures and therefore the results of the algorithm applied in the virtual machine migration process, especially at low sampling frequencies.

Once defined a sampling strategy, we define *representation models* that read the sampled time series and produce "reduced views" filtered from outliers and reflecting the behavioral load trend of hardware and software resources. These representation models may be linear (e.g., moving average, ARMA) or nonlinear (e.g., polynomial, spline).

Activation strategy. The activation strategy decides when the live migration mechanism has to start. Current solutions operate in two ways: *statically* or *adaptively*.

Static approaches, such as (Khanna et al. 2006, Wood et al. 2007), set some threshold value and live migration is triggered as soon as the host load overcomes that predefined value. For example, the scheme proposed by (Khanna et al. 2006) classifies a host as a sender or as a receiver depending on whether its load is beyond or below some fixed

thresholds. When the load of a host overcomes the threshold, this solution moves all the selected guests to the physical host having the least available resources sufficient to run them without violating the SLA. If there is no available host, it activates a new physical machine. Static approaches cannot work in a networked system consisting of thousands of hosts where, at a checkpoint, a threshold may signal hundreds of senders and, at the successive checkpoint, the number of senders can become few dozen or, even worse, most servers differ from those of the previous set. Also the decision about which guests is useful to migrate from one server to another is affected by similar problems if we adopt some threshold-based method. Adaptive approaches where no static thresholds are used to activate the live migration process are preferable. In the adaptive scenario, the activation is triggered by adaptive factors, such as *significant changes* in the host load conditions. A significant change is any modification in the statistical behavior of the load that lasts for a considerable number of consecutive measurements (Casolari et al. 2012). Considering different statistical behaviors brings to the detection of different significant changes, like *trend changes* or *state changes*. A significant trend change happens when the trend patterns of the load vary over time in their direction (e.g., upward or downward) or in their distribution (e.g., linear or exponential). For example, Figure 3(a) shows a host load profile (concerning host CPU utilizations) presenting two trend changes. An exponential increasing trend (until sample 156) is followed by an exponential decreasing trend (from sample 157 to sample 336) and then by a horizontal linear trend (from sample 337 to sample 600). A significant state change is a considerable variation of the mean load value that occurs either instantaneously or rapidly with respect to the period of sampling and that lasts for a significant number of consecutive measurements (Casolari et al. 2012). The load profile in Figure 3(b)

presents two significant state changes, one upwards and one downwards. At sample 200, the mean load value passes from ≈0.3 to ≈0.6, and then it returns to ≈0.3 at sample 420.

a) Trend change profile

b) State change profile

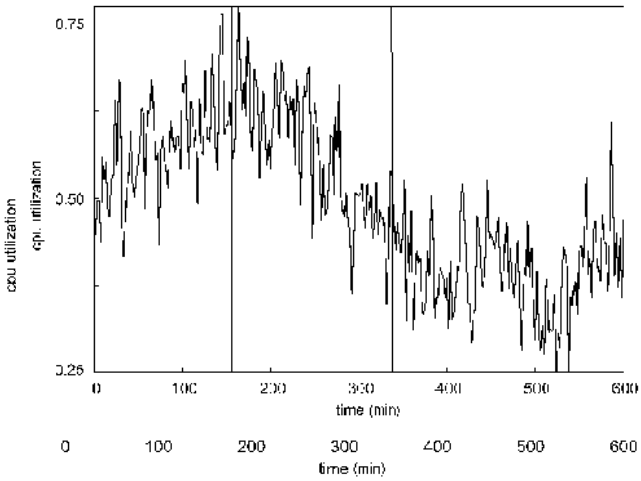


Figure 3. Load profiles of hosts.

The ability of capturing such changing conditions in host and guest load profiles is crucial for an activation strategy to reduce the number of migrations to just the most significant ones.

In the case study described in Section 4, we present the implementation of an approach that adaptively selects as the senders only the hosts subject to significant state changes of their load. On the other hand, (Stage et al. 2009) consider the bandwidth consumption during migration and propose a system that classifies the various loads in order to consolidate more guests on each host based on typical periodic trends, if they exist.

Similar considerations are also valid for the *component selection* which is responsible for choosing hosts and virtual machines that will participate in the live migration process.

Destination. The destination host can be in the same local subnet or, in the most modern networked systems such as the clouds, even located in another geographical

network. Common solutions for the selection of receiver hosts are limited to guests that can migrate only among hosts that are within the same subnet and share common storage (Kamna & Sugandha 2012). This limit prevents adaptive migration of guests among federated datacenters that will be more and more important in the next future for performance and resilience reasons.

Some works (Ramakrishnan et al. 2007, Travostino et al. 2006, Clark et al. 2005) address the problems of guest migration across WANs and aim to reduce downtime during migration. For example, the solution proposed in (Clark et al. 2005) is very efficient because it is able to transfer an entire machine causing a downtime of few hundreds of milliseconds. (Travostino et al. 2006) migrate virtual machines on a WAN area with just 1-2 seconds of application downtime through lightpath (DeFanti et al. 2003). (Ramakrishnan et al. 2007) propose cooperative, context-aware migration mechanisms through existing server virtualization technologies and by proposing dynamic storage replication technologies to facilitate migration across geographically interconnected machines.

Migration mechanism. Current live migration mechanisms are mainly based on the *pre-copy* or *post-copy* schemes.

In the pre-copy scheme, the bulk of the guest's memory state is migrated to a target node even as the guest continues to execute at a source host (Hines & Gopalan, 2009). If a transmitted page is dirtied, it is re-sent to the target in the next round. This iterative copying of dirtied pages continues until either a small, writable working set has been identified, or a preset number of iterations is reached, whichever comes first. This represents the end of the memory transfer phase and the beginning of service downtime.

The guest is then suspended and its processor state plus any remaining dirty pages are sent to a target node. Finally, the guest is restarted and the copy at source is destroyed.

On a high-level, post-copy migration defers the memory transfer phase until after the guest's CPU state has already been transferred to the target and resumed there. Post-copy first transmits all processor state to the target, starts the guest at the target, and then actively pushes memory pages from source to target. Concurrently, any memory pages that are faulted on by the guest at target, and not yet pushed, are demand-paged over the network

4. Case study

In this chapter, we present a case study application of adaptive migration algorithms in an experimental testbed of 30 physical machines hosting 140 virtual machines. The adopted virtualization solution is VMware ESXi 4.0. We collect periodic samples of the CPU utilization through the VMware monitor (VMware) with a frequency of one every twenty seconds. We show an application of a guest selection algorithm that is able to adaptively select the most critical guests for each server on the basis of a load trend-based model instead of traditional approaches based on instantaneous or average load measures.

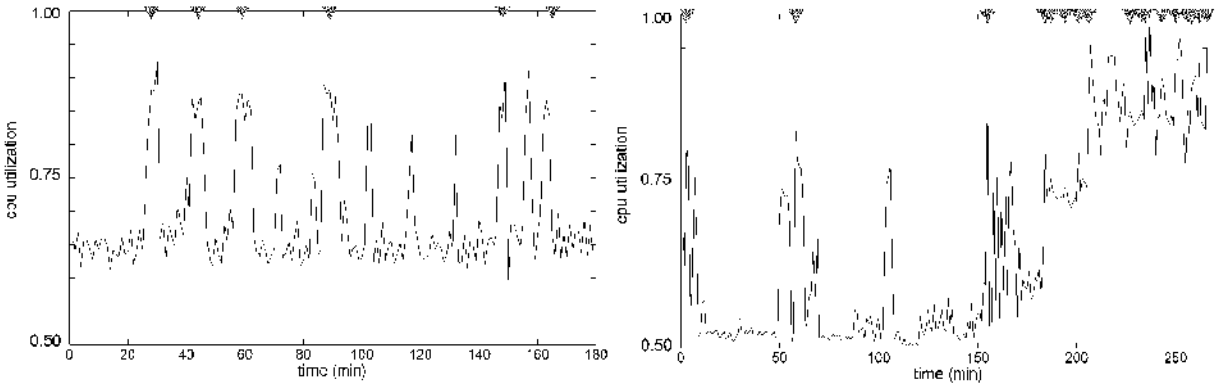
The focus is on the first two phases (selection of the sender host, and selection of guests) that concern open research issues and that represent the core of Section 4.1 and Section 4.2, respectively. The last two phases (selection of receiver hosts, and assignment of guests) are hinted at in Section 4.3.

4.1 Selection of sender hosts – CUSUM *vs* Static models

The identification of the set of sender hosts represents the most critical problem for the adaptive management of a networked architecture characterized by thousands of machines where an abuse of guest migrations would devastate performance and resilience.

We present an adaptive algorithm for sender hosts selection that guarantees high performance and low overheads since it is able to limit the number of migrations to few really necessary instances. The algorithm considers the load profile evaluated through the CUSUM-based stochastic model (Page 1957). The goal is to signal only the hosts subject to *significant state changes* of their load, where we define a state change *significant* if it is intensive and persistent. This is not an easy task when the application context consists of large numbers of hosts subject to: many instantaneous spikes, non-stationary effects, and unpredictable and rapidly changing load. As examples, Figure 4(a) and Figure 4(b) show two typical profiles of the CPU utilization of two hosts in a cloud architecture. The former profile is characterized by a stable load with some spikes but there is no *significant state change* in terms of the previous definition. On the other hand, the latter profile is characterized by some spikes and by two significant state changes around sample 180 and sample 220. A robust detection model should arise no alarm in the former case, and just two alarms in the latter instance. In a similar scenario, it is clear that any detection algorithm that takes into consideration an absolute or average load value as alarm mechanism tends to cause many false alarms. This is the case of threshold-based algorithms (Khanna et al. 2006, Wood et al. 2007) that are widely adopted in several management contexts. Just to give an example, let us set the load threshold to define a sender host to 0.8 of its CPU utilization (done for example in (VMware)). In Figures 4, the small triangles on the top of the two figures denote the checkpoints where the threshold-based detection algorithm signals the host

as a sender. There are 10 signals in the former case and 17 in the latter case instead of the expected 0 and 2. This initial result denotes a clear problem with a critical consequence on performance: we have an excessive number of guest migrations even when not strictly necessary.



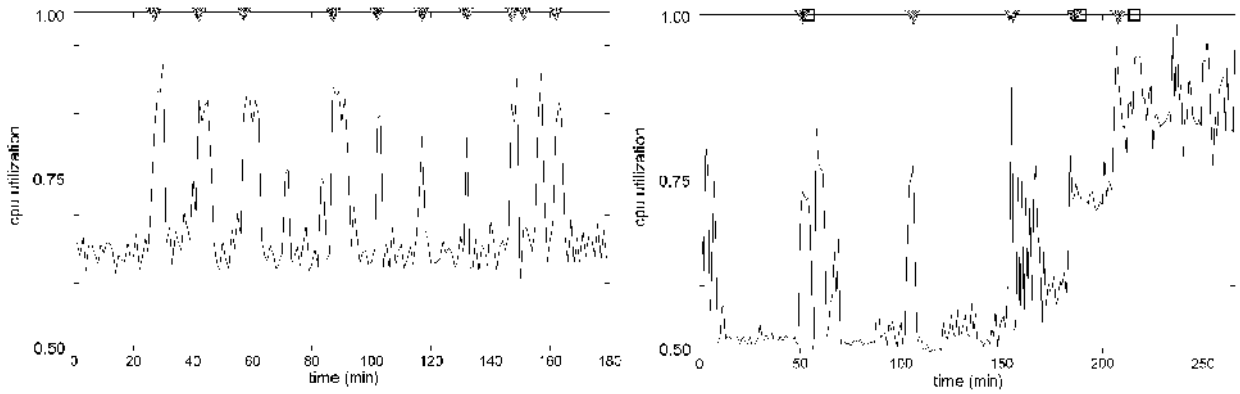
a) Profile 1 b) Profile 2

Figure 4. CPU load in two hosts.

The proposed algorithm adopts a different approach for selecting sender hosts by evaluating the entire load profile of a resource and aiming to detect abrupt and permanent load changes. To this purpose, we consider a stochastic model based on the CUSUM algorithm (Page 1957) that works well even at runtime. We consider both the simpler *Baseline CUSUM* implementation and the more sophisticated *Selective CUSUM* implementation presented in (Andreolini et al. 2009). The Baseline CUSUM model statically uses reference values for its parameters, while the selective CUSUM algorithm adaptively adjusts its parameters according to data characteristics.

In Figure 5, we report the results obtained by using both Baseline and Selective CUSUM for sender host selection. If we compare the triangles plotted in Figures 5 with

those in Figures 4 (referring to a threshold-based algorithm), we can appreciate that the total number of detections is significantly reduced because it passes from 27 to 11. In particular, the Baseline CUSUM is able to avoid detections due to load oscillations around the threshold value. On the other hand, it is unable to address all the issue of unnecessary detections related to short-time spikes, such as those occurring at samples 30, 45, 55 and 90 in Figure 5(a).



a) Profile 1 b) Profile 2

Figure 5. Baseline and Selective CUSUM models.

The three small boxes on the top of Figure 5(b), instead, denote the activations signaled by the Selective CUSUM. This algorithm determines robust and selective detections of the sender hosts because it is able to remove any undesired signal caused by instantaneous spikes in Figure 5(a), and to detect only the most significant state changes at samples 55, 185, 210 in Figure 5(b), actually just one more (at sample 55) than the optimal selection of two signals.

4.2 Selection of guests – Trend-based *vs* Sample-based

When a host is selected as a sender, it is important to determine which of its guests should migrate to another host. As migration is expensive, it is important to rely on adaptive solutions that are able to select few guests that have contributed to the significant load change of their host. For each host, the proposed adaptive solution is based on following three steps:

(1) evaluation of the load of each guest;

(2) sorting of the guests depending on their loads;

(3) choice of the subset of guests that are on top of the list.

The first step is the most critical, because there are several alternatives to denote the load of a guest. Let us consider for example the CPU utilization of five virtual machines (A-E) in Figure 6 obtained by the VMware monitor.

The typical approach of considering the CPU utilization at a given sample as representative of a guest load (e.g., Khanna et al. 2006, Wood et al. 2007) is not a robust choice here because the load profiles of most guests are subject to spikes. For example, if we consider samples 50, 62, 160, 300 and 351, the highest load is shown by the guest B, albeit these values are outliers of the typical load profile of this guest.

Even considering as a representative value of the guest load the average of the past values may bring us to false conclusions. For example, if we observe the guests at sample 260, the heaviest guest would be A followed by E. This choice is certainly preferable to a representation based on absolute values, but it does not take into account an important factor of the load profiles: the load of the guest E is rapidly decreasing while that of the guest A is continuously increasing.

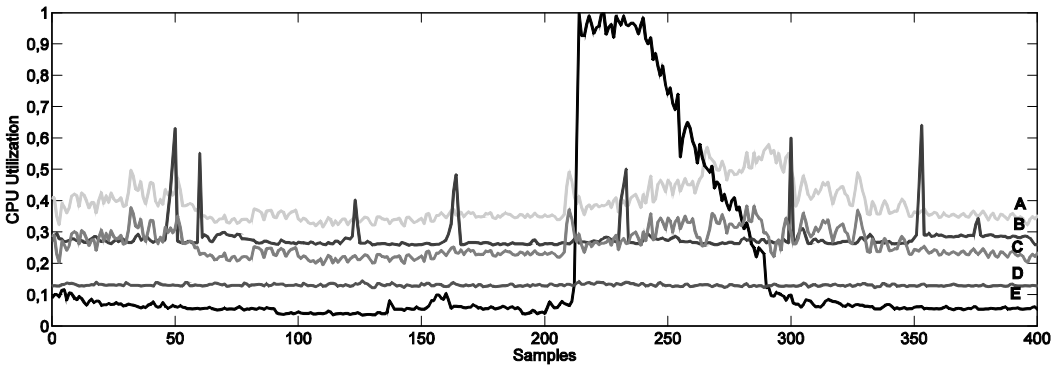


Figure 6. Profiles of guest machines

The idea is that a guest selection model should not consider just absolute or average values, but it should also be able to estimate the *behavioral trend* of the guest profile. The behavioral trend gives a geometric interpretation of the load behavior that adapts itself to the not stationary load and that can be utilized to evaluate whether the load state of a guest is increasing, decreasing, oscillating or stabilizing. Consequently, it is possible to generate a load representation of each guest based on the computation of a weighted linear regression of *trend coefficients* and the actual load value of a server. After having obtained a load representation for each guest, they sort them from the heaviest to the lightest and then select only the guests that contribute to one-third of the total relative load. To give an idea, let us consider two hosts H_1 and H_2 characterized by the following load values: *(0.25, 0.21, 0.14, 0.12, 0.11, 0.10, 0.03, 0.02, 0.01)*, and *(0.41, 0.22, 0.20, 0.10, 0.04, 0.02, 0.01)*, respectively.

In H_1, we select the first two guests because the sum of their relative loads 0.46 exceeds one-third. On the other hand, in H_2 we select just the first guest that alone contributes to more than one-third of the total load.

4.3. Selection of receiver hosts and assignment of guests

Although migration mechanisms are rapidly improving (Kamna & Sugandha 2012, Ibrahim et al. 2011), live migration remains an expensive task that should be applied selectively especially in a cloud context characterized by thousands of physical machines and about one order more of virtual machines. The receiver selection process must be carefully designed to avoid migration loops that could occur between sender and receiver hosts. For example, an overloaded resource consuming guest being moved to a receiver host may trigger a further migration process if the receiver host becomes part of the senders at next activation. The scheme proposed by (Khanna et al. 2006) moves all the selected guests to the physical host having the least available resources sufficient to run them without violating the SLA. If there is no available host, it activates a new physical machine. Next to performance and SLA requirements, also bandwidth consumption should be considered in the selection of receiver hosts. The authors in (Bobroff et al. 2007) introduce prediction techniques and a bin packing heuristic to allocate and place virtual machines while minimizing the number of activated physical machines. They also propose an interesting method for characterizing the gain that a virtual machine can achieve from dynamic migration. (Stage et al. 2009) propose dynamic scheduling models for the assignment of guests to hosts taking into account additional migration control parameters like bandwidth adaptation behavior, minimum and maximum bandwidth usage, iterated pre-copy migration algorithms, and different termination iteration conditions.

Once selected the receiver hosts, we have to assign them guests selected for migration. As we want to spread the migrating load to the largest number of receiver hosts, we want that no receiver should get more than one guest that is, $G = R$. Hence, we have to guarantee that the number of guests we want to migrate is $G < (N - S)$. Typically, this

constraint is immediately satisfied because S is a small number, $S << N$, and typically $G \leq 2S$.

However, if for certain really critical scenarios it results that $G > (N - S)$, we force the choice of just one guest for each sender host. This should guarantee a suitable solution because otherwise we have that $S > R$ that is, the entire platform tends to be overloaded. Similar instances cannot be addressed by an adaptive migration algorithm but they should be solved through the activation of standby machines (Khanna et al. 2006) that typically exist in a networked data center. It is also worth to observe that all our experiments were solved through the method based on the one-third of the total relative load with no further intervention.

5. Conclusions

Adaptive live migrations of virtual machines are an interesting opportunity to allow networked infrastructures to guarantee resilience and accommodate changing demands for different types of processing subject to heterogeneous workloads and time constraints. Nevertheless, there are many open issues about the most convenient choice about when to activate migration, how to select guest machines to be migrated, and the most convenient destinations. These problems are becoming even more severe in modern data centers and cloud architectures characterized by hundreds of thousands of virtual machines. The proposed adaptive algorithms and models are able to identify just the real critical host and guest devices, by considering the load profile of hosts and the load trend behavior of the guest instead of thresholds, instantaneous or average measures that are typically used in literature.

References

Ibrahim K. Z., Hofmeyr, S., Iancu, C., Roman, E., Optimized pre-copy live migration for memory intensive applications, In: Proc. of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, 2011.

Andreolini, M., Casolari, S., Colajanni, M., Messori, M., "Dynamic load management of virtual machines in a cloud architecture", In: Proc of First Int. Conference on Cloud Computing, 2009

Casolari, S., Tosi, S., Lo Presti, F., An adaptive model for online detection of relevant state changes in Internet-based systems, Performance Evaluation, vol. 69, no. 5, 2012.

Page, E. S.: Estimating the point of change in a continuous process. In: Biometrika vol. 44, 1957.

Khanna, G., Beaty, K., Kar, G., Kochut, A.: Application Performance Management in Virtualized Server Environments, In: Proc. of Network Operations and Management Symp., 2006.

Stage, A., Setzer, T.: Network-aware migration control and scheduling of differentiated virtual machine workloads, In: Proc. of 31st Int. Conf. on Software Engineering, 2009.

Clark, C., Fraser, K., Steven, H., Gorm Hansen, J., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live Migration of Virtual Machines, In: Proc. of the 2nd ACM/USENIX Symp. on Networked Systems Design and Implementation, 2005.

Travostino, F., Daspit, P., Gommans, L., Jog, C., de Laat, C., Mambretti, J., Monga, I., Van Oudenaarde, B., Raghunath, S., Wang, P. Y.: Seamless live migration of virtual machines over the MAN/WAN, Future Gener. Computer System, vol. 22, no. 8, 2006.

DeFanti, T., de Laat, C., Mambretti, J., Neggers, K., St. Arnaud, B.: TransLight: a global-scale LambdaGrid for e-science, Communications of the ACM, 2003.

Hines, M. R., Gopalan, K.: Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning, In: Proc. of the ACM SIGPLAN/ SIGOPS Int. Conf. on Virtual execution environments, 2009.

Wood, T., Shenoy, P., Venkataramani, A., Yousif, M.: Black-box and Gray-box Strategies for Virtual Machine Migration, In: Proc. of the 4th USENIX Symp. On Networked Systems Design and Implementation, 2007.

Hermenier, F., Lorca, X., Menaud, J.-M., Muller, G., Lawall, J.: Entropy: a Consolidation Manager for Cluster, In: Proc. of the Int. Conf. on Virtual Execution Environments, 2009.

Chen X., Gao X., Wan H., Wang S., Long X., Application- Transparent Live Migration for virtual machine on network security enhanced hypervisor. Research paper. China Communications, 2011.

Kamna, A., Sugandha, S., A Survey on Infrastructure Platform Issues in Cloud Computing , International Journal of Scientific & Engineering Research, vol. 3, no. 6, 2012.

Ramakrishnan, K. K., Shenoy, P., Van der Merwe, J., Live data center migration across WANs: a robust cooperative context aware approach, In: Proc. of the 2007 SIGCOMM workshop on Internet network management, 2007.

VMware Distributed Power Management Concepts and Use. Paragraph: use this for the first paragraph in a section, or to continue after an extract.

Gerofi, B., Fujita, H. And Ishikawa, Y., An efficient process live migration mechanism for load balanced distributed virtual environments, In: Proc. of 2010 IEEE International Conference on Cluster Computing.

Jung, G., Hiltunen, M. A., Joshi, K. R., Schlichting, R. D. And Pu, C., Mistral: dynamically managing power, performance, and adaptation cost in cloud infrastructures, In: Proc. of 2010 IEEE International Conference on Distributed Computing Systems.

Shetty, J., Anala M. R. And Shobha, G., A survey on techniques of secure live migration of vitual machine, In: International Journal of Computer Applications, Vol. 39, No. 12, Feb. 2012.

Bobroff, N., Kochut, A., Beaty, K., Dynamic Placement of Virtual Machines for Managing SLA Violations, In: Proc. of the 10th IFIP/IEEE International Symp. On Integrated Network Management, 2007.