2013 IEEE 12th International Symposium on Parallel and Distributed Computing

# Analysis of requirements for virtual machine migration in dynamic clouds

Stelios Sotiriadis, Nik Bessis
School of Computing and Mathematics
University of Derby
Derby, United Kingdom
(s.sotiriadis, n.bessis)@derby.ac.uk

Pawel Gepner
Intel Corporation
United Kingdom
pawel.gepner@intel.com

Nicolas Markatos
National Technical University of Athens
Greece and KETAK, Mediterranean
College, Greece
n.markatos@ntua.gr

***Abstract*— Highly dynamic environments like clouds by nature cause a high degree of unpredictability of resource utilization and performance. Failures, latencies and heterogeneity should always be the main concern for affecting the scheduling decisions in distributed infrastructures. As a result, the scheduling efficiency of jobs before their submission is very difficult to be achieved or either forecasted. Even in the cases of the most complex schedulers a comprehensive dynamic view cannot always be predicted. Thus, the rescheduling concept takes advantage of the current scheduling status and performs a dynamic scheduling decision. In this paper we present a discussion of the virtual machine migration strategies that are currently available in distributed systems based on the need of migrating virtualized resources in order to achieve better resource utilization and performance such as improve load balancing, makespan and higher throughput of jobs. We conclude our study with a critical discussion of vital requirements for virtual machine migration.**

***Keywords—Cloud, Virtual Machine, Virtual Machine Migration, Process Migration, Live Migrations***

## I. INTRODUCTION

Cloud computing aim is to facilitate an environment for wider distribution of services where users access resources remotely on a pay on demand model. So, cloud computing is defined as a bespoke service setting where resources (hardware and software) that reside to remote locations are utilized by everyday Internet users. The on-demand services are included in virtualized environments named as virtual machines (VMs). Specifically, the VM term refers to the virtual representation of a part of computational resources along with an operating system. In this context a cloud defines three main roles namely as the service consumer, the service provider and the service creator [1]. Traditionally, the service creator generates a service that is utilized by the consumer and represents the user hardware and software requirements for leasing cloud capacity. This request is hosted in the premises of the service provider.

A cloud service life cycle contains various user requests for services submitted to a cloud service provider. So cloud can be seen as a large-scale dynamic environment that combines distributed computing requirements such as resource unpredictability [31]. So the job submissions of user tasks share similar features. The overall view covers requests for service (jobs) that submitted by the cloud clients to the service cloud providers. Thus, all job submissions are enclosed and executed in clouds, a process that is called sandboxing.

The term virtualization in clouds refers to the deployment of virtual hosts belonging to cloud datacentres instead of utilizing the core physical resources in order to split the computational power of the underlying infrastructure. The fundamental idea is that the actual physical machine (host) generates and orchestrates various VMs (called guest machines) through its operating system. The terms host and guests distinguish the software that is executed in the VMs. In addition, the host machine contains software for creating and controlling the virtual parts that called hypervisor. The last one controls and allows multiple isolated guests to run concurrently within the same host machine.

In general, a cloud could be considered as a highly dynamic environment with a high degree of unpredictability of resource utilization and performance. In such systems the most important features are related with resource management e.g. discovery and scheduling [32]. This work is focusing on the resource scheduling in distributed systems. Failures, latencies and heterogeneity should always be the main concern for affecting the scheduling architecture decision of distributed infrastructures. As a result, the scheduling efficiency of jobs before their submission is very difficult to be achieved or either forecasted. Even in the cases of the most complex schedulers a comprehensive dynamic view cannot always be predicted [30]. Thus, the rescheduling concept takes advantage of the "in the progress system scheduling status" and performs a highly dynamic scheduling decision.

This could be achieved by utilizing a migration technique that is directly related with the virtualization paradigm [2]. The concept of performing migration is based on the movement of a job or a set of job tasks to relevant resources in order to improve load balancing, makespan and higher throughput of jobs [9]. This work presents a state-of-the-art analysis of requirements for dynamic migration in cloud environments. By focusing on related approaches, we present a literature review study in order to analyze the most important migration techniques and to highlight its features. This will lead to the identification of key requirements. Specifically, section 2 presents a discussion of virtualization in clouds. This includes the two generic classifications of migration procedures in the area of scheduling, called process migration and live migration. The rest of the paper is organized as follows. Section 3 details the review of techniques, section 4 emphasizes the most important requirements for VM migration and section 5 concludes with a summary and the future works section.

978-0-7695-5018-3/13 $26.00 © 2013 IEEE
DOI 10.1109/ISPDC.2013.24

## II. VIRTUALIZATION IN CLOUDS

As discussed previously the cloud defines a model in which software is hosted, run and administered in large web data centres and provided as a service [1]. By using a variety of web technologies, both hardware and software Cloud services can be delivered through the Internet in a seamless manner. Several potentials of Clouds exist, depending on how the Cloud is employed and applied to different areas. The main types of Clouds are: a) Infrastructure as a Service (IaaS), b) Platform as a Service (PaaS), and c) Software as a Service (SaaS).

Starting with the IaaS, [3] states that Cloud is provided to the end users as a virtualization capability, and by using a web interface services can be delivered through a reliable access. Secondly, the PaaS offering facilitate deployment of applications without the cost and complexity of buying and managing the underlying hardware and software [3]. It allows developers to create their own Cloud applications using supplier-specific tools and programming languages. PaaS offers rapid development of web-based application at low cost in a public or private manner. Finally, the SaaS, allows existing applications to be run on a cloud supplier's hardware. With SaaS, providers authorize applications to users as a service on demand, through a pay per-use fee.

In any case IaaS, PaaS and SaaS environments could be considered as a virtual cloud environment, where the hypervisor plays a vital role in the whole service management procedure. In general, [4] suggest that two types of hypervisors exist, the Type 1; native or bare metal hypervisor that is executed within the physical computer for hosting guests, and the Type 2; hosted hypervisors that execute guests as applications on an unmodified commodity OS. Examples of Type 1 are the Kernel-based VMs (KVM) and Xen, while examples of Type 2 include the VMWare Server and Workstation, Parallels Workstations and Oracle VM VirtualBox. In any case of Type 1 or Type 2, developers make use of the hypervisor software for developing and deploying their services (hardware or software) relied upon the generic needs of the customers or the company's leasing target, by always aiming to scalability and flexibility of lightweight solutions.

At a first glance, the most common used hypervisors are the Xen and the KVM which both are under the GNU general public licence. [5] compare both solutions and discuss that Xen project has been released earlier in 2003 and has been included in various Linux distributions, while is also the base hypervisor for Citrix Enterprise solution and Amazon EC2. In contrast, KVM, has been released in 2007 [4]; it introduced a new way to manage VMs, that has been proven to be quite efficient while at the same time particularly lightweight as presented in [7]. All these years various studies have compared both hypervisors and authors in [8] suggest that in the case of comparable performance Xen scalability properties outperforming KVM. Nevertheless, the choice for one hypervisor or the other can depend on performance, flexibility of use, and elasticity of requested services as well as strategic considerations [5] of the cloud provider.

In the case of management of the overall VM development, the hypervisor plays an important role as controls the OS and the deployment of applications within the VM. It should be mentioned that the hypervisor is located among the physical host and the VM layers of the layered structure as illustrated in figure 4. There are two basic types of hypervisors, the Type 1: bare-metal and Type 2: hosted. Figure 1 demonstrates the Type 1 hypervisor that is located beneath the host hardware layer [6] and creates VM operating systems (VM-OS).

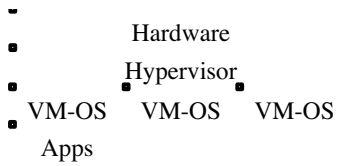


Fig. 1. The Type 1: Bare-Metal Hypervisor Structure

Figure 2 demonstrates the Type 2 hypervisor that is placed as software beneath the OS layer of the hardware [6].

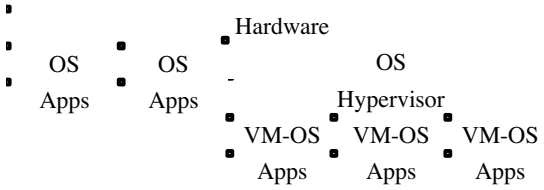


Fig. 2. The Type 2: Hosted Hypervisor Structure

In general a cloud environment could utilize both aforementioned types of hypervisors. The next section presents a detailed discussion of migration techniques with regards to scheduling.

## III. DISCUSSION OF VM MIGRATION TECHNIQUES

Highly dynamic environments by nature cause a high degree of unpredictability of resource utilization and performance. In the distributed system the migration has appeared based on the need of transferring VMs among resources. The idea is simple, by performing migration of a job or a set of job tasks to relevant resources the rescheduling procedure could offer improved load balancing, makespan and higher throughput of jobs [9]. There are two generic classifications of migration procedures in the area of scheduling, called process and live migration.

Process migration is an old studied approach, which includes the procedure of transferring a process from one machine (host resource) to another (remote resource). Live migration, conversely, includes the movement of VMs from one machine to another while processes are still up and running. Process and live migration have advantages and drawbacks however the live migration presents more challenges because of the large amount of data that needs to be transferred. In addition, the live migration is computational expensive solution as it is associated with memory migration.

The following section presents a state-of-the-art discussion of the most well-known techniques, approaches and systems. Specifically, we base on both approaches (process and live migrations) and we present each work and its association to the scheduling concept for dynamic environments.

### *A. Process migration*

This old approach firstly implemented in 1985 for operating systems e.g. MOSIX etc. usually was focused on

migration issues for achieving load balancing [10]. In the cloud computing paradigm process migration could lead to many potential benefits. The most common need for that is the transparency of the setting in which downtime is the major performance factor that makes live migration a more competent method. As downtime is defined the due time in which a status of a process changes from running to suspending and running again. In other words, downtime is the relocation time. Specifically, process migration is also correlated with the problem of residual dependencies. This could lead to unbalanced performance when a process is migrated and is transferred from an operating system.

In any case, process migration was a top research area in 1980s, as many operating systems were evolved to distributed operating systems with the capability of stop and relocate running process tasks and applications [10]. The authors also discuss that the most generic process migration classifications are in the level of operating systems, user-level, object-based and virtualization at the operating system level. The following presents an analysis of the major process migration efforts.

- Operating systems migration: The most widely known solutions were the Accent Amoeba Charlotte, Mosix, Sprite and V. By providing a single image across a cluster of machines [10] they provide a migration mechanism. This is mainly based on the kernel design, which provides transparent execution environment. Although the single management system simplifies the scheduling process, these systems share significant drawbacks. First, when a resource node procedure fails then the remote procedure fails also. Secondly, the approached have shown low flexibility and low heterogeneity [10]. Also it is complex to be implemented as it requires significant improvements on operating systems.
- User-level migration: In contrast with the kernel context theme described above, several systems have been developed to support process migration at the user level. As there is not kernel support these systems have been developed for executing long-running applications on a cluster machine. For that reason the implementation of process migration on these systems e.g. Condor, CoCheck libckpt and MPVM is difficult to be achieved as new services needs to be developed in the operating system. This drawback minimizes the number of application that can be used with such systems [10]. Also, this solution cannot use the inter-process communication set of methods (IPC) for exchanging data among multiple processes.
- Object-based migration: Several systems have been developed that provide migration using object-based approaches including Abacus, Emerald, Globus, Legion and Rover. Specifically, these solutions have been developed as programming languages and middleware tools for achieving migration. [10] suggest that a reduced amount of state that required to be recorded and moved is required. Since new programming paradigms have been utilized, taking advantage of migration of legacy applications is minimal. As a result application needs to be created using new programming languages [10].

The rescheduling concept based on the migration mechanism it appears to provide a good method for improving performance as presented in [11]. Job migration has been presented in [12] with the aim of understanding the impact of migration of parallel jobs in distributed systems. The authors suggest that by using migration an extra ability of moving some or all of the tasks of a job to different resources during execution. This adds flexibility for filling queue holes which otherwise remain empty. The work shows definitely benefits from migration for both gang scheduling and backfilling gang-scheduling. Specifically, the migration mechanism first vacates tasks from node to node and then re-instantiate those to the target set. Finally the authors suggest that migration when combined with backfilling can be beneficial in terms of utilization.

In [13] authors present a totally decentralized load balancer. The model uses the ProActive library for the migration of jobs, and a multicast channel to coordinate the nodes. It improves the decision time in non-centralized environment as offers large stability. However, the method has not good efficiency, the throughput is medium, and the scalability is low. In general this method is based on percentage loading at node.

### *B. Live migration*

Live migration or virtual machine migration provides the ability to transfer VMs from one physical resource (host machine) to another (remote or destination machine). The major advantage of this method is that migration happens without pre-empting execution and without any perceived degradation [14]. Using this way, the migration is strongly isolated and there is no need for name-spaces sharing. Also, downtime is not affected as the whole VM is transferred and interfaces to VMs are clearly defined. However, migration to a wide area environment is a challenging issue. The large amount of data to be transferred makes it tough to be achieved effectively majorly based on the huge amount of network bandwidth in addition to the memory capacity needs. In the following discussion is presented various virtual machine migration methods and tools with the aim of identifying advantages and drawbacks of each technique.

In [15] the authors have shown a system that transfers a computer's state from one machine to another in a sufficient amount of time. Specifically they use an example in which an OS instance is transferred from the work computer to the home computer using the slow DSL link while the user is driving back home. By using four optimization techniques namely copy-on-write, demand paging, ballooning and hashing present a study in which future systems can take advantage for designing capsule migration. The capsule state includes the OS and running application processes. Starting with the copy-on-write solution authors suggest that "by using copy-on-write to capture the updates to disks, the amount of state transferred to update a capsule is proportional to the modification made in the capsule. Then demand paging, based on the user requests capture the part of the capsule which is demanded by the user. Finally, while ballooning minimizes the transferring time by removing all the unnecessary data from the memory, the hashing exploits similarities of capsules for speeding-up the data transfer.

In [16] the authors present the implementation of a system that uses VM technology to provide fast and transparent migration of applications. This work, unlikely to previous attempts, encapsulates the state of a running application. Using

VM, allows the encapsulation of VMs along with the OS in a fully transparent way for the users. Specifically, this happens by migration of the memory in order to minimize the downtime. Initially, the migration process includes selection of VM to migrate and its destination. Then, while the VM is running at the source a pre-copying procedure happened of the memory state. The control of the VM is transferred to the destination in which it is resumed. Finally, any remaining memory states are sending to the relocated machine, whilst removing the dependency of the source machine. Using this method a high transparency comes at the price of performance degradation at the time of the memory state migration.

The authors of [17] suggest the use of VMs for the internet Suspend/Resume project. Particularly, they mimic the opening and closing of a laptop the users are capable of suspending work at one machine and resuming to another. The key for achieving that is the ability of layering VM on a distributed file system. The initial prototype shows that internet Suspend/Resume can be successfully implemented on today's hardware by suspending VM monitor images in the distributed filesystems and makes it available to multiple locations.

In [18] the authors have developed a more advanced version called Coda. Coda clients could accommodate the whole VM in their caches and support "a clean interface to exploit advance knowledge of resume site" [18]. The internet Suspend/Resume project is mainly based on a WAN scenario and the best case travel time at migration is 45 seconds for 100 Mb/s network. In a more realistic bandwidth speed scenario (1 Mb/s) the downtime is approximately 14 minutes, thus making this solution effective in terms of functioning but insufficient in terms of downtime performance.

In [20] the discuss the NomadBIOS, an application which runs on top of the L4 kernel. L4 kernels are a family of microkernels usually used to Unix-based operating systems [19]. NomadBIOS starts an incoming new OS as a new L4 task and it provides it with a virtualized address space, memory and Ethernet address. Similarly, in [21] authors present the NomadLinux, a version of L4 linux. In this case the memory is paged by NomadBIOS so it is easy to migrate a memory state from one to another host. Both solutions minimize downtime by using pre-copy migration, in other words keeping the OS running after migration and by tracking changes send updates to the original site through iterations [20]. Those solutions are considered as host driven migration and benchmarks shows that performance was generally on parallel with VMWare.

The work of [22] presents the Denali hypervisor for hosting internet services. The first Deanli effort didn't support interpositions and was only able to host a specially developed OS called "Ilwaco" [22]. This solution also didn't support virtualization of MMU (Memory Management Unit). The μ-Denali is a more advanced version of the Denali, which includes a stop-and-copy VM migration and support of virtual MMU. Both systems comparing with the traditional process migration systems provide better isolation, however, issues such as security as not yet fully considered.

Authors in [19] present a prototype based on Xen [23]. Xen provides a platform for allowing plenty OS to run on a single machine by deploying a variety of service (web-content, media stream distribution etc.). Their self-migration algorithm is capable of transferring a copy of its entire state to a different machine using pre-copy migration. In this way a viable mechanism for supporting advanced scheduling in clusters and grids is suggested in which the OS is keeping responsive. However, the authors do not present comprehensive benchmark results but it is a more theoretical framework for identifying migration implications.

The work in [24] suggests that important benefits can be gained through virtualization in data centres. Specifically, the authors suggest the Sandpiper, a system that initiates the migration of data centres by monitoring and detecting hotspots. The system is based on two strategies, namely black-box and grey-box. The first one suggests that "all usages must be inferred solely from external observations and without relying on OS-level support inside the VM". The second one uses a light-way daemon to monitor virtual servers' statistics. Through evaluation of a Xen-based prototype, the authors have shown that VM migration is a viable method for eliminating simultaneous hotspots involving multiple resources. Furthermore, their results show that "Sandpiper is able to resolve single server hotspots within 20 seconds and scales well to larger, data centre environments".

Authors in [25] present the Shirako system which deals with issues – architectural and algorithmic – for resource management policy. The system is an on demand leasing of shared resources organized in federated clusters. "The Shirako architecture factors provisioning and placement where provider sites retain control over VM placement, but delegate limited provisioning power to brokers" [25]. In this way the authors show that migration is important way to solve problems among policies by supporting advance reservations. Finally, the system has been extended to support live migration in dynamic environments including utility and grid computing.

In [26], authors present a recent work on migrating VMs between clouds. They highlight that need by suggesting that an emerging requirement for clouds is to enable better service availability. The proposed migration mechanism aims to improve efficiency of migrating storage in a wide area. The great difference of this method with the conventional approaches is that instead migrating one large piece which needs to be transferred from beginning to end, it transfers storage blocks in a planned sequence. Thus, the authors develop a scheduling algorithm to make use of the VM workloads to compute the ordering of chunks. The evaluated results show that the method effectively reduces the extra traffic.

The work presented in [27] describes a system that enables live VM migration for a wide-area that uses local storage and open network connections. Specifically, the system has been developed as part of the Xen facility for live migration, and allows the VM to continue running on the source host during the migration. Using this technique, it guarantees consistency, not service disruption and not high I/O performance overhead. In addition, the authors have shown that migration works well, as it doesn't significantly decrease the performance of services running in the VM. However, storage migration here inherently faces significantly challenges because a much larger size needs to be moved instead of a memory chunk.

To conclude, the work discussed above surveys process, live and storage migration solutions. The ability to live-migration of applications among physical resources is a very

popular solution. This is mainly because of the minimization of downtime needed for migration including all states (memory, networks etc.), thus offering a high quality solution for administrators who for example want to perform maintenance. In addition, load-balancing among different data-centres can be achieved through this techniques in a such way that clouds could cooperate with each other with the aim to provide a high user experience. The following section discusses a summary of the advantages and drawbacks of each technique in order to identify the most important.

## IV. Analysis of Selected Approaches

This section demonstrates the advantages and drawbacks of the approaches discussed in the literature. Specifically, by highlighting their characteristics we conclude to a critical discussion that forms the requirements for VM migration.

In [12] authors present the migration mechanism that first vacates tasks from node to node and then re-instantiate those to the target set. The authors suggest a migration solution that is combined with backfilling scheduling and can be beneficial in terms of utilization. The advantages of this technique are:

- Higher acceptance utilization.
- Smaller slowdowns and wait times for fixed utilization.
- Gang scheduling and backfilling improved in an average opportunity of scheduling tasks in empty wholes in the queue.

The benchmark analysis is based on metrics related to the maximum system utilization, job slowdown, acceptance utilization and waiting times. However, the highlighted disadvantages of this technique are as follows:

- The maximum utilization does not change from a system perspective.
- There is no opportunity for improvements when not enough jobs or not enough holes in the scheduling queue exist.

In [13] authors presents a solution that is based on percentage loading at nodes, that uses the ProActive library for the migration of jobs, and a multicast channel to coordinate the nodes. The advantages of this technique are:

- Good migration forecasting accuracy.
- Large stability.
- Proactive non-centralized mechanism.

The benchmark analysis is based on metrics related to the system utilization and throughput. However, the highlighted disadvantages of this technique are as follows:

- Small resource utilization.
- Medium average throughput.
- Low to average migration efficiency.

In [10] the work presents the Zap, a system to allow process migration of domains called pods. Zap capsules without extensive OS changes inspired virtualization. The advantages of this technique are:

- It does not require significant OS changes.
- Migration happens while preserving open network connections.

The benchmark analysis is based on metrics related to the virtualization cost and virtualization overhead. However, the highlighted disadvantages of this technique are as follows:

- Limited success primarily because of the difficulty of encapsulating the state of a running application.
- Uniform operating system configuration across all participating nodes.

In [15], the work utilizes four different optimization techniques namely copy-on-write, demand paging, ballooning and hashing. The study is visionary where future systems can take advantage for designing capsule migration. The advantages of this technique are:

- Migration of OS instance happened in sufficient amount of time.
- The approach reduces the migration times.

The benchmark analysis is based on metrics related to the migration of propagated software updates and data transferring, migration time for DSL and LAN links. However, the highlighted disadvantages of this technique are as follows:

- Mainly optimized for slow DSL networks.
- The OS execution stopped while migration is taking place.
- Use a specific set of enhancements to reduce the transmitted image size.

In [16] authors present a system that uses VM technology to provide fast and transparent migration of applications. This work, unlikely to previous attempts, encapsulates the state of a running application. Using VM, allows the encapsulation of VMs along with the OS in a fully transparent way for the users. The advantages of this technique are:

- Experiments show that real world memory downtime takes less than a second.
- Full control of migration procedure and control of impact of other VMs running states.

The benchmark analysis is based on metrics related to the CPU reservation, Pre-copy and downtime. However, the highlighted disadvantages of this technique are as follows:

- Significant resources required transferring VMS.
- The system consumes time (significant number of seconds) for VM migration even in fast networks.

In [17] the authors use VMs for the Internet Suspend/Resume project. Particularly, they mimic the opening and closing of a laptop the users are capable of suspending work at one machine and resuming to another. The key for achieving that is the ability of layering VM on a distributed file system.

- Stores VM monitors images in a network.
- The approach makes image accessible for multiple locations.

The benchmark analysis is based on metrics related to suspend, resume times and downtimes. However, the highlighted disadvantages of this technique are as follows:

- Downtime not efficient.
- Portability is considered as limited.

In [18] the work details a second version of [17] in which tasks are split into chunks. In this way the Coda distributed file system will be able to track chunks and store them to regularly visited nodes. The advantages of this technique are:

- More efficient than [17] in terms of task splitting.
- Accommodate whole VMs in caches.

The benchmark analysis is based on metrics related to the migration downtime. However, the highlighted disadvantages of this technique are as follows:

- More appropriate for scenarios associated to wide area networks.
- High downtime migration (Average 45 seconds).

In [20] authors describe the NomadBIOS that starts an incoming new OS as a new L4 task and it provides it with a virtualized address space, memory and Ethernet address for performing the OS migration. The advantages of this technique are:

- Reduces downtime using pre-copy algorithm.
- Further optimization through ARP packet.
- It is considered as a complex method for real-time systems.

The benchmark analysis is based on metrics related to the migration downtime. However, the highlighted disadvantage of this technique is related with issues on migration of block device contents.

The work of [21] presents the NomadLinux. This is a version of L4 Linux in which memory is paged by NomadBIOS so it is easy to migrate a memory state from one to another native Linux host. The advantages of this technique are:

- Downtime performance in parallel to VMWare.
- The approach offers better scalability.

The benchmark analysis is based on metrics related to the migration slowdown and downtime. However, the highlighted disadvantage of this technique is that migration only works for native Linux machines.

The work of [22] presents the μ-Denali that is a more advanced version of the Denali, which includes a stop-and-copy VM migration and support of virtual MMU. The system addresses the problem of support for developing cooperative virtual machine services. The advantages of this technique are:

- Handling physical resource management.
- Device namespace virtualization.
- Virtual hardware event trapping and routing.

The benchmark analysis is based on metrics related to the port table, migration downtime. However, the highlighted disadvantages of this technique are as follows:

- The solution is linked to a centralized resource sharing.
- Non-live stop-and-copy migration, so a high downtime could be observed.

The work of [19] presents a self-migration algorithm that is capable of transferring a copy of its entire state to a different machine using pre-copy migration. In this way a viable mechanism for supporting advanced scheduling in clusters and grids is suggested in which the OS is keeping responsive. The advantages of this technique are:

- Considers security, accounting, performance, flexibility and portability.
- The solution keeps the OS responsive during migration.

However, the highlighted disadvantages of this technique are as follows:

- Self-migration needs to be re-implemented for each type of guest OS.
- Benchmarks have not been presented in a detailed manner.

The work of [24] presents the Sandpiper, which is a system that initiate migration using automation of monitoring tasks and by detecting hotspots determines, a new mapping of physical to virtual resources. The advantages of this technique are:

- Xen live migration for hotspot migration.
- Use of the Distributed Resource Scheduler for load balancing.
- Improved responsiveness of the system.

The benchmark analysis is based on metrics related to the workload and CPU utiliztion. However, the highlighted disadvantages of this techniqueis that it does not support replication services automation.

In [25] authors present the Shirako, which is an architecture that enables flexible factoring of resources in federated clusters by supporting VM migration, based on lease policies using advanced reservations. The advantages of this technique are:

- Leases are dynamic based.
- Uses cryptographic operations to access clusters.
- Support for live VM migration mechanisms.

The benchmark analysis is based on metrics related to the number of migrations to placement policies. However, the highlighted disadvantages of this technique are as follows:

- Solution for a cluster base federation of resources.
- Shirako suspend VM in the case of live-migration cannot happened for reducing the complexity of the problem

In [26] authors describe a storage migration-scheduling algorithm for improving storage and input/output performance during migration. The advantages of this technique are:

- Benefit of scheduling increases when Internet bandwidth decreases.
- Reduces the amount of extra traffic.
- Benefits of scheduling increases, as the image size gets larger.

The benchmark analysis is based on metrics related to the workload migration. However, the highlighted disadvantages of this technique are as follows:

- Degradation in high amount dirty data environments (file servers and mail servers).

In [24] the work details the design, implementation and evaluation of a storage migration system to support transparent live migration. It allows VM to continue running on the source machine during migration to achieve stability and consistency and it does not include additional service disruption compared to memory-only migration.The advantages of this technique are:

- Live migration persistent.
- Consistency of VM in the destination machine.
- VM migrated services are not affected.

The benchmark analysis is based on metrics related to the migration throughput and disk input/output overhead. However, the highlighted disadvantages of this technique are as follows:

- Performance affected as the larger number of migrated size when compared with memory-only migration.
- There is no support for data compression facilities.
- There is no support for batch jobs.

Next, the study is focusing on the summary of requirements for various VM migration cases.

## V. Analysis of Requiremetns for VM Migration

The study correlates the solutions described in section IV. Figure 3 demonstrates the association of literature review approaches with the highlighted requirements. The numbers denote the approach number from the list of references.

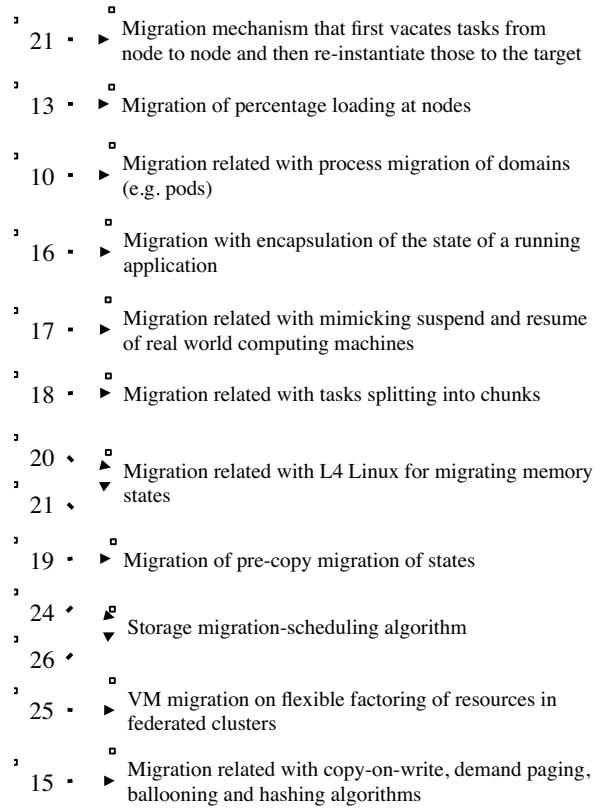


Fig. 3. The analysis of the literature review approaches

The research objectives are discussed as follows:

- Migration mechanisms that first vacates tasks from node to node and then re-instantiate those to the target set offer higher acceptance utilization and smaller slowdowns and wait times for fixed utilization as in [21].
- Migration of percentage loading at nodes offer good migration forecasting accuracy and stability as in [13].
- Process migration of domains (e.g. pods) does not require significant OS changes as in [10].
- Encapsulation of the state of a running application show that real world memory downtime offer optimized performance (migration consumes less than a second) as in [16].
- Replication of suspending and resuming of real world computing machines offers the ability to make image accessible for multiple locations as in [17].
- Tasks splitting into chunks are more efficient than mimicking the solutions of suspend and resume of real world computing machines and could accommodate whole VMs in caches as in [18].
- L4 Linux for migrating memory states offer optimized values for downtimes and offer better scalability as in [20] and [21].
- Stop-and-copy VM migration offers efficient handling of physical resource management as in [22].
- Transferring a copy of its entire state to a different machine using pre-copy migration rely on a solution that keeps the OS responsive during migration as in [19].
- Storage migration-scheduling algorithm reduce the amount of extra traffic thus decrease the Internet bandwidth as in [24] and [26]. For example [24] uses the Distributed Resource Scheduler (DRS) for load balancing.
- Architectures that enable flexible factoring of resources in federated clusters by supporting VM migration offer leasing in dynamic based cases and live VM migrations as in [25].
- Copy-on-write, demand paging, ballooning and hashing offer migration of OS instance in sufficient amount of time. For example the work of [15] describes an approach that approach with four scheduling mechanism to reduce the migration times.

To conclude, the applicability of the VM migration in clouds is particular valuable for service consolidation and isolation scenarios. In particular, the VM migration will be useful in order to organize services from multiple providers to be collaborative. A flexible solution includes the migration VMs among IT infrastructures in order to enhance the agility in improving the quality of service in cases of system overload.

## VI. Conclusion and Future Steps

The virtualization approach expands the cloud capabilities with the aim of achieving a more transparent setting for users. In such settings, one of the most important design issues is the dynamic-ness of the system, thus VM migration could offer the required infrastructure to allow transferring of virtualized parts among clouds. The future directions of this work are related with the interoperable cloud setting namely as inter-cloud. Through an effective VM migration inter-cloud elasticity and scalability will increase its efficiency.

So, a more extended interoperable environment of clouds will offer additional advances along with elasticity and scalability. This includes new services to users in a coordinated workload management setting. Specifically, we aim of enhancing the Inter-Cloud Meta-Scheduling (ICMS) [28] model with a VM migration tactic and perform experiments in the SimIC toolkit [29]. Further to this, the Message Exchanging Optimization (MEO) model [33], [34] will offer the require framework to allow a more sophisticated and light-weighted mechanism for VM migration. The purpose is to develop VM migration in an inter-cloud setting and to allow service distribution to achieve better elasticity and scalability.

## References

[1] S. Sotiriadis, N. Bessis, P. Sant, and C. Maple “From Grids to Clouds: A collective intelligence study for inter-cooperated infrastructure”s, The Fourth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP-2010), IARIA, 25 – 30 October 2010, Florence, Italy, pp.: 142-147.

[2] S. Sotiriadis, N. Bessis, F. Xhafa and N. Antonopoulos, “Cloud Virtual Machine Scheduling: Identifying Issues in Modeling the Cloud Virtual

Machine Dynamic Instantiation", 6th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS-2012), Palermo, July 4-6 2012, pp. 233-240.

[3] The Future Of Cloud Computing, Available at: http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf, Accessed at 24/02/2013

[4] P. Li, and L. W. Toderick, "Cloud in cloud: approaches and implementations". In Proceedings of the 2010 ACM conference on Information technology education (SIGITE '10). ACM, New York, NY, USA, pp. 105-110

[5] D. Cerbelaud, S. Garg and J. Huylebroeck "Opening the clouds: qualitative overview of the state-of-the-art open source VM- based cloud management platforms". In Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware (Middleware '09). Springer-Verlag New York, Inc., New York, NY, USA, , Article 22 , 8 pages

[6] 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". In SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pages 164–177, New York, NY, USA

[7] I. Habib "Virtualization with kvm". Linux J., 2008(166):8

[8] J. N. Matthews, T. Deshane, Z. Shepherd, M. Ben-Yehudah, A. Shah, and B. Rao, "Quantitative comparison of xen and kvm". In Xen Summit, June

[9] K. Kurowski, B. Ludwiczak, J. Nabrzyski, A. Oleksiak, and J. Pukacki "Dynamic grid scheduling with job migration and rescheduling in the GridLab resource management system". Sci. Program. 12, 4 (December 2004), pp. 263-273.

[10] S. Osman, D. Subhraveti, G. Su, and J. Nieh "The design and implementation of Zap: a system for migrating computing environments". SIGOPS Oper. Syst. Rev. 36, SI (December 2002), pp. 361-376.

[11] H. Dail, O Sievert, F Berman, H. Casanova, A. YarKhan, S. Vadhiyar, J. Dongarra, C. Liu, L. Yang, D. Angulo, and I Foster "Scheduling in the Grid application development software project. In Grid resource management", Jarek Nabrzyski, Jennifer M. Schopf, and Jan Weglarz (Eds.). Kluwer Academic Publishers, Norwell, MA, USA pp. 73-98.

[12] Y. Zhang, H. Franke, J. E. Moreira, and A Sivasubramaniam "The Impact of Migration on Parallel Job Scheduling for Distributed Systems". In Proceedings from the 6th International Euro-Par Conference on Parallel Processing (Euro-Par '00), Arndt Bode, Thomas Ludwig, II, Wolfgang Karl, and Roland Wismller (Eds.). Springer-Verlag, London, UK, 242-251.

[13] J. Bustos "Robin hood: An active objects load balancing mechanism, for intranet". In Proc. of Workshop de Sistemas Distribuidos y Paralelismo, Chile, 2003

[14] C. Ward, N. Aravamudan, K. Bhattacharya, K. Cheng, R. Filepp, R. Kearney, B. Peterson, L. Shwartz, C. C. Young "Workload Migration into Clouds Challenges, Experiences, Opportunities," Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on CLOUD 5-10 July 2010, pp.164-171

[15] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum "Optimizing the migration of virtual computers". In Proceedings of the 5th symposium on Operating systems design and implementation, (OSDI '02). ACM, New York, NY, USA, pp. 377-390.

[16] M. Nelson, B.-H. Lim, and G. Hutchins "Fast transparent migration for virtual machines". In Proceedings of the annual conference on USENIX Annual Technical Conference (ATEC '05). USENIX Association, Berkeley, CA, USA, 25-25.

[17] M. Kozuch, and M. Satyanarayanan "Internet suspend/resume". In Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, 2002

[18] M. Kozuch, M. Satyanarayanan, T Bressoud, C Helfrich, and S Sinnamohideen "Seamless Mobile Computing on Fixed Infrastructure". Computer 37, 7 (July 2004), pp.65-72.

[19] J. Gorm H. and E. Jul. 2004 "Self-migration of operating systems". In Proceedings of the 11th workshop on ACM SIGOPS European workshop (EW 11). ACM, New York, NY, USA, Article 23

[20] J. Liedtke "On micro-kernel construction". In Proceedings of the fifteenth ACM Symposium on Operating System Principles, pages 237-250. ACM Press, 1995

[21] H. Hartig, M. Hohmuth, J. Liedtke, and S. Schonberg. "The performance of micro-kernel-based systems". In Proceedings of the sixteenth ACM Symposium on Operating System Principles, pages 66-77. ACM Press, 1997.

[22] A. Whitaker "Building System Services with Virtual Machine Monitors". Ph.D. Dissertation. University of Washington, Seattle, WA, USA. AAI3199790.

[23] 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". In Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03). ACM, New York, NY, USA, pp. 164-177.

[24] T. P. Wood, A. Shenoy, A. Venkataramani and M. Yousif "Black-box and Gray-box Strategies for Virtual Machine Migration". In Proceedings of the Fourth Symposium on Networked System Design and Implementation (NSDI '07), 2007.

[25] L. Grit, D. Irwin, A. Yumerefendi, and J. Chase. "Virtual Machine Hosting for Networked Clusters: Building the Foundations for "Autonomic" Orchestration". In Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing (VTDC '06). IEEE Computer Society, Washington, DC, USA

[26] J. Zheng, T. S Eugene Ng, and K. Sripanidkulchai "Workload-aware live storage migration for clouds". In Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (VEE '11). ACM, New York, NY, USA, pp. 133-144.

[27] R. Bradford, E. Kotsovinos, A. F., and H. Schiberg "Live wide-area migration of virtual machines including local persistent state". In Proceedings of the 3rd international conference on Virtual execution environments (VEE '07). ACM, New York, NY, USA, pp. 169-179.

[28] S. Sotiriadis, N. Bessis, N. and Antonopoulous, N. "SimIC: Designing a new Inter-Cloud Simulation Platform for Integrating Large-scale Resource Management". In proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications (AINA-2013), March 25-28, Barcelona [to appear]

[29] S. Sotiriadis, N. Bessis, P. Kuonen, and N. Antonopoulous "The Inter-cloud Meta-scheduling (ICMS) Framework". In proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications (AINA-2013), March 25-28, Barcelona [to appear]

[30] N. Bessis, S. Sotiriadis, F. Xhafa, F. Pop And V. Cristea "Meta-scheduling Issues in Interoperable HPCs, Grids and Clouds", International Journal of Web and Grid Services, InderScience (SCI JSCR IF 2010: 0,978 – SJR: Q2 [Computer Networks and Communications, Software]), Volume 8, Issue 2, Inderscience, ISSN: 1741-1106, pp.: 153-172.

[31] S. Sotiriadis, N. Bessis, Y. Huang, P. Sant and C. Maple "Defining Minimum Requirements of Inter-collaborated Nodes by Measuring the Weight of Node Interactions", 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2010), 15th-18th February, Krakow, pp.: 291-298.

[32] S. Sotiriadis, N. Bessis, Y. Huang, P. Sant and C. Maple "Towards to Decentralized Grid Agent Models for Continuous Resource Discovery of Interoperable Grid Virtual Organizations", International Workshop on Distributed Information and Applied Collaborative Technologies (DIACT-2010), in conjunction with the 3rd International Conference on the Applications of Digital Information and Web Technologies (ICADIWT-2010), 12th -14th July 2010, Istanbul, pp.: 170-175.

[33] N. Bessis, S. Sotiriadis, F. Pop and V. Cristea "Optimizing the Energy Efficiency of Message Exchanging for Service Distribution in Interoperable Infrastructures", 4th IEEE International Conference on Intelligent Networking and Collaborative Systems (INCoS-2012), September 19-21 2012, Bucharest, Romania, ISBN: 978-0-7695-4808-1, pp. 105-112

[34] N. Bessis, S. Sotiriadis, F. Pop and V. Cristea "Using a Novel Message-Exchanging Optimization (MEO) Model to Reduce Energy Consumption in Distributed Systems", Simulation Modeling Practice and Theory Elsevier, 2013 (in press)