A Formal Approach for Virtual Machine Migration Planning

Saeed Al-Haj and Ehab Al-Shaer

Department of Software and Information Systems
University of North Carolina Charlotte
Charlotte, NC, USA
{salhaj, ealshaer}@uncc.edu

***Abstract*—Cloud computing is an emerging paradigm in information technology. Virtualization is the corner stone for this paradigm in which resources are utilized by running multiple virtual machines (VMs) on a physical host. During the VM's life cycle, the cloud provider may migrate the VM from one host to another host. During the live migration process, some security, capacity, and dependency requirements are subject to violations due to the temporal relationship between migration steps. In this paper, we present a formal approach to plan VM migration; that is to find a sequence of migration steps such that all security, dependency, and performance requirements are met. The migration planning problem is modeled as a Constraints Satisfaction Problem and it is solved using Satisfiability Modulo Theory (SMT) solvers. We provide *VMM-Planner*, a formal framework that provides a VM migration plan to formally verify the given requirements in all intermediate migration steps.**

I. Introduction

Virtualization technology is the solid ground that cloud computing stands on. It provides means to utilize resources by consolidating multiple VMs to run on the same hosting machine. Clouds give their users the ability to scale up/down their resources, to start/stop their VMs at any time, and to use pay-as-you-go scheme for the requested resources. These features keep the cloud in a dynamic state in which the available resources fluctuate among cloud users. VM migration is the main driver that keeps the dynamic state of a cloud usable for both the cloud provider and the cloud customer.

VM migration plays an essential role in clouds' infrastructure management to cope with the dynamic nature of clouds. This nature, even though it is considered as a bright side of clouds, has dark sides. There are many angles to look to the dark sides of clouds; one side can be seen from the angle of low resource utilization caused by the departed VMs. VM migrations can bring the light into these dark sides. These migrations aim to reallocate VMs to different host machines to exploit the wasted resources. Other than low resources utilization, there are several reasons why VM migration is useful. The reasons include, but are not limited to, hardware failure and scheduled maintenance, load balancing, disaster recovery, energy saving and thermal management for datacenters [1], resources compaction for future growth, and security zones management [2].

There is tremendous work that has been done in the area of VM migration. The problem of VM migration has four sub-problems: 1) identifying the hotspots, physical hosts that need to be switched off; 2) selecting which VMs have to be migrated and where to be reallocated; 3) setting the frequency of migration, how frequent the migration process is triggered and when to start it; and 4) planning the migration process. The literature shows a lack of research done in the area of VMs migration planning. The VM migration planning is the first step toward achieving consistent configuration updates in clouds. In other words, providing a correct and safe migration sequence will make configuration updates more accurate.

A successful migration plan decreases both migration time and migration downtime, and incurs minimal disruptions during the live migration process. Some VM migration sequence plans are better than other sequence plans because they consider the VMs' characteristics to determine the migration order [3].

In this work, we are targeting the problem of VM migration planning, finding the sequence in which VMs are migrated. Migrating VMs in an arbitrary sequence may violate some dependency requirements. Generally, migrating one VM that has communication dependency with another VM may incur communication latency that breaks the SLA. For example, the sequence order of migrating a web server workload VM affects the downtime and transfer time for other VMs as observed in the work done in [3]; therefore, it is suggested to migrate web server workload VMs before the other VMs.

It is also important to insure that VMs keep certain requirements during the live migration process. Such requirements include risk and security requirements, dependency requirements, and performance requirements. Risk requirements are defined to reduce the potential of cross-VM attacks [4], by placing VMs with conflicting risk requirements on different hosts. Another example that shows the importance of migration planning is VM migration in over-committed clouds. In over-committed clouds [5], resources are limited and some VM migration scenarios may look infeasible. In such a case, it can take extra migration steps to reach the final (target) placement. Therefore, it is important to insure that the previously mentioned requirements are enforced during the entire migration process.

ISBN 978-3-901882-53-1, 9th CNSM and Workshops ©2013 IFIP

We introduce *VMM-Planner*, a framework that finds a sequence of migration steps in which the current VMs placement converges to the target placement without violating dependency, risk, and capacity requirements. *VMM-Planner* encodes VM migration planning problem into a constraint satisfaction problem in which all requirements, the current cloud status, and the target placement are formally encoded as constraints. Then, the modeled constraints are fed to a satisfiability modulo theory (SMT) solver. The solver finds a satisfiable solution to the given instance, if such solution exists. The provided solution is the VM migration sequence that can be performed to execute the live VM migration process safely.

Our contributions in this paper can be summarized as follows:

- Modeling the problem of VM migration planning as a constraint satisfaction problem,
- Providing an implementation of the framework and integrating it with the SMT solver, and
- Evaluating the model.

The paper is organized as follows: §II motivates and defines the problem and discusses its complexity. The modeling constraints are described in §III. The evaluation of our model is shown in §IV. In §V, the related work is briefly discussed. Finally, §VI concludes the paper and presents the future work.

II. Motivations, Definitions, Problem Statement, and Problem Complexity

A. Motivations

The VM migration is an elementary part in managing dynamic clouds. In most cases, migrating a set of VMs and reallocate them to different hosts are required. The migration planning problem considers the cases in which there is a temporal relationship between VM migration steps. The temporal relationship affects the sequence in which VMs are migrated. One reason that influences the sequence ordering is the bandwidth limit. The optimal case is to migrate all VMs at once, but the bandwidth capacity limits the number of VMs that can be migrated at the same time. Another reason that shows how migration planning is useful is the collocation dependency between VMs. Consider a case in which the application server and the database server are running on two different VMs. The application server VM has frequent data-intensive communication to the database server; thus, it is required to allocate both VMs within a close physical distance from each other for performance purposes. During live migration process, the sequence of migration may cause communication latency if the collocation dependency is not considered which can lead to SLA violations. An example of such requirements is shown in §II-D. Security configurations can also affect the temporal relationship between the migrated VMs. Migrating one VM before the other can drop the communication between both VMs due to the security context of a VM that depends on its old location [6]. The workload characteristics of VMs affect the migration cost in terms of migration downtime and migration transfer time [3]. As observed in the experiments done by [3], a memory-dirty work load VM or a web server work load VM will affect these VMs migrated before them; therefore, it is recommended to give higher priority for memory-dirty workload VMs and web server workload VMs to be migrated before other VMs to reduce the cost of migrations.

The previous examples show how the migration planning can be useful to avoid the cases in which the migration process can incur extra cost or violate some dependency requirements during the migration sequence.

B. Definitions and Problem Statement

Let $\mathcal{PM}$, $|\mathcal{PM}| = m$, represent the set of available physical machines (PMs). Let $\mathcal{VM}$, $|\mathcal{VM}| = n$, represent the set of running virtual machines. VMs are specified by their resources requirements such as CPU, memory, disk space, etc. Also, PMs are specified by their resources capacity. Without loss of generality, we will consider only CPU and memory resources dimensions in this work.

Let $\mathcal{M}$, $\mathcal{M} \subset \mathcal{VM}$, represent the set of virtual machines to be migrated. Let $\mathcal{H}$, $\mathcal{H} \subset \mathcal{PM}$, represent the set of hotspot physical machines that are scheduled to be switched off. Let $\mathcal{D}$ represent the dependency matrix for the running VMs. In $\mathcal{D}$, $D_{x,y} = 1$ means that both VM_x and VM_y have communication interdependency. Let $\mathcal{C}$, $\mathcal{C} \subset \mathcal{VM}$, represent the set of critical VMs; the VMs that cannot be migrated or reallocated after their first allocation assignment.

Let $\mathcal{R}$ represent the risk score for all VMs. R_i is the risk score for VM_i. Risk scores for VMs depend on several factors such as: reachability and connectivity between VMs, the vulnerability of VMs, and the impact value (cost of damage) for VMs. More details about calculating VMs' risk scores can be found in [2]. In this work, the risk constraints require a VM to be placed on a physical machine (PM) that hosts VMs with similar risk score. Risk constraints are also required to be enforced during intermediate migration steps.

Let π_x represent the placement status for all virtual machines at time x, π_0 represents the initial placement, and π_τ represents the target placement. Let σ represent the transition from one placement status to another. For example, $\sigma_1 : \pi_0 \xrightarrow{1} \pi_1$ represents one transition from π_0 to π_1. The transition from one placement status to another placement status reflects the changes due to VM migration. In a single transition, one VM migration (serial) or multiple VM migrations (parallel) can be performed.

Let $\rho(\pi_x)$ represent the safety constraints for the placement status π_x. The safety constraints include capacity, dependency, and risk constraints. At each intermediate migration step, the safety constraints have to be enforced.

The list of all variables and symbols used in this work is shown in Table I.

The *migration planning problem* is defined as follows: given a set of physical machines, a set of VMs allocated on the physical hosts, the set of VMs to be migrated, the set of hotspot PMs, the set of critical VMs, and the target placement of VMs, find a migration plan; a sequence of migration steps, that

converges the initial placement to the target placement with a feasible cost, without violating the capacity constraints of individual hosts, communication interdependency constraints, and security migration constraints. Costs may include data storage and transfer, migration interruptions, migration time, and number of migrations.

Formally, we state the problem as follows:

Given:

$$\mathcal{PM}, \mathcal{VM}, \mathcal{H}, \mathcal{M}, \mathcal{D}, \mathcal{C}, \mathcal{R}, \pi_0, \pi_\tau$$

Find:

$$\sigma_k \mid \sigma_k : \pi_0 \xrightarrow{k} \pi_\tau \wedge \left(\bigwedge_k \rho(\pi_k) \right)$$

The problem is modeled as Constraints Satisfaction Problem (CSP) not as an optimization problem. Our goal is to find a satisfiable migration plan that satisfies all the constraints and the thresholds.

The solution includes the migration steps required to assemble resources and make the space needed for the target assignment.

C. Problem Complexity

The migration planning problem is similar to the N-dimensional Bin Packing problem which is known to be "NP-Hard" problem [7], [8]. Due to the space limitations, we will omit providing a proof in this paper, but it is possible, using reduction techniques from similar problems [7], to show that finding a migration plan with the minimum cost is "NP-Hard".

D. Example

Fig. 1 shows a simple example to migrate a set of VMs. For simplicity, we consider only the memory capacity and dependency requirements as safety requirements in this example. Using the same notations presented in §II-B, The system is described as follows: $\mathcal{VM} = \{VM_1, VM_2, VM_3, VM_4\}$, $\mathcal{PM} = \{PM_1, PM_2, PM_3\}$, $\mathcal{M} = \{VM_1, VM_2\}$, $\mathcal{D} = \{(VM_1, VM_2)\}$, $\mathcal{H} = \{PM_1\}$, and $\mathcal{C} = \{VM_3\}$. Also, the communication dependency threshold is set to *one*, $T_D = 1$, which means that any two VMs having dependency between them are required to be allocated on hosts within one link distance, at most, from each other. PM_1 and PM_2 are within one link distance, PM_2 and PM_3 are also within one link distance.

The initial placement state, π_0, is:

$\pi_0 = \{(VM_1, PM_1), (VM_2, PM_1), (VM_3, PM_2), (VM_4, PM_3)$

The target placement state, π_τ, is:

$\pi_\tau = \{(VM_1, PM_3), (VM_2, PM_2), (VM_3, PM_2), (VM_4, PM_3)\}$

The goal is to find a migration plan that makes π_0 converge to π_τ without violating safety constraints. The *VMM-Planner* provides the following plan:

1) $\sigma_1 : (VM_2, PM_1) \xrightarrow{1} (VM_2, PM_2)$
2) $\sigma_2 : (VM_1, PM_1) \xrightarrow{2} (VM_1, PM_3)$

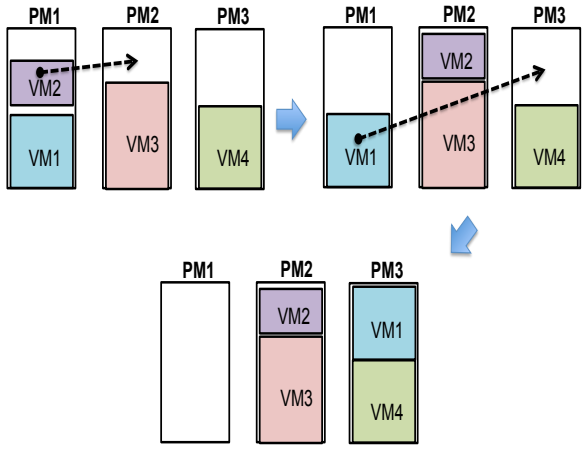

Fig. 1. An Example of VMs Migration.

The previous migration plan maintains the dependency requirements between VM_1 and VM_2 at all intermediate states. Note that migrating VM_1 before VM_2 will break the dependency requirement at the first migration step; PM_1 and PM_3 are within 2 links distance and the threshold is 1. Therefore, this migration sequence is not safe and will be avoided.

This example shows how important the migration sequence is, especially when the size of the problem increases. Building an automated framework to provide migration plans is a great benefit for situations that may look infeasible from the first glance. Another example that shows the importance of VM migration planning is shown in [9], the example considers the link bandwidth limits as constraints for VM migration.

III. Modeling

In this section, we describe the modeling constraints that convert the migration planning problem to a constraint satisfaction problem.

a) Placement Boolean variables: A decision variable is needed to control the placement of virtual machines on physical machines. This decision variable is restricted to be a Boolean variable as follows:

$$a^k_{i,j} \in \{0, 1\} \tag{1}$$

The variable $a^k_{i,j}$ indicates if the VM_i is mapped to the PM_j at the k^{th} migration step.

b) Mutual Exclusion: In a cloud environment, physical servers are utilized by running more than one VM on the same server, but a VM can only run on one PM.

$$\forall k \quad \sum_{j=1}^{m} a^k_{i,j} = 1 \tag{2}$$

Constraint 2 ensures that for every virtual machine VM_i, there is only one decision variable, $a^k_{i,j}$, is set to *one* at every migration step k. Mutual exclusion constraint sets all other decision variables to *zero* once a virtual machine is allocated.

TABLE I
GLOSSARY OF ALL VARIABLES USED IN THE MODEL

Variable	Type	Description
$\mathcal{VM}$	Set	The set of running VMs
$\mathcal{PM}$	Set	The set of physical machines (PMs)
$\mathcal{H}$	Set	The set of "'hotspot" (PMs) that is scheduled to be switched off
$\mathcal{M}$	Set	The set of VMs to be migrated
$\mathcal{R}$	Set	The set of risk scores for all VMs
$\mathcal{C}$	Set	The set of critical VMs
$\mathcal{D}$	Matrix	The dependency matrix between VMs
n	Integer	Number of virtual machines
m	Integer	Number of physical machines
$a^x_{i,j}$	Boolean	Decision variable to place VM_i on PM_j at the x^{th} migration step
c_i^{cpu}, c_i^{mem}	Real	*cpu/memory* capacity for VM_i, respectively
C_j^{cpu}, C_j^{mem}	Real	*cpu/memory* capacity for PM_j, respectively
s_j	Boolean	A variable to indicate the status of PM_j
o_i	Integer	The priority score of VM_i
π_x	Set	The placement status for all VMs at the x^{th} migration step
$\rho(\pi_x)$	Formula	The safety condition for the placement status π_x
η_i	Boolean	A variable to flag critical VMs
$\delta^k_{i,j}$	Boolean	A variable to indicate that VM_i is migrated to PM_j at the $(k+1)^{th}$ migration step
r_i	Enumeration	The risk score for the VM_i
$l_{v,w}$	Integer	The physical distance between PM_v and PM_w
T_D	Integer	A threshold to set the max physical distance between two dependent VMs

c) Capacity limits: The capacity of VMs hosted in a single PM should not exceed the capacity limits of that PM. CPU, memory, disk space, and bandwidth are the main specification dimensions to be used in resources provisioning. Without loss of generality, we consider only CPU and memory dimensions in the current model. Constraint 3 represents the *CPU* constraint ensuring that capacity limitations will not be violated.

$$\forall k, \forall j \in PM \qquad \sum_{i=1}^{n} \left(a^k_{i,j} * c_i^{cpu}\right) < C_j^{cpu}, \qquad i \in \mathcal{VM} \quad (3)$$

The previous constraint limits the total CPU capacity of VMs allocated in a single physical machine not to exceed the CPU capacity of the hosting physical machine. Variables c_i^{cpu} and C_j^{cpu} represent CPU specs for VM_i and PM_j, respectively. Similarly, other constraints such as memory, disk space, and bandwidth can be modeled using the same notations used in constraint 3. The generality of the model allows more than one specification dimension to be considered at the same time.

d) Single migration per VM: Let $\delta^k_{i,j}$ represent the change of placement status for VM_i at two consecutive migration steps, k and $k+1$. Formally:

$$\delta^k_{i,j} \in \{0, 1\} \quad (4)$$

$$(\delta^k_{i,j} = 1) \Leftrightarrow (\bigwedge_{q=0}^{k} \neg a^q_{i,j}) \wedge (\bigwedge_{r=k+1}^{\tau} a^r_{i,j}) \quad (5)$$

The Boolean variable, $\delta^k_{i,j}$, is set to 1 if the VM_i is migrated from a hosting machine to a different host, PM_j, at the $(k+1)^{th}$ migration step. To avoid unnecessary migrations in which a VM is migrated several times, we will limit each VM to be migrated once at most. Constraint 5 ensures that by setting $a^r_{i,j}$ equal to *one* for all steps from (k+1) and onward. Also, this constraint helps to avoid situations in which the allocation of a VM is flipped in a loop between PMs.

e) Dependency Constraints: Migrating VMs may break the communication interdependency between VMs. Dependent VMs are required to be placed within a pre-determined physical distance threshold, T_D. In our model, we define the physical distance between physical hosts as number of links between the pair of machines hosting dependent VMs. Enforcing dependency constraints eliminates a primary factor for performance degradation; it helps bringing dependent VMs to be close to each other. The dependency requirement is given as an input to model, a Boolean dependency matrix, D, is defined to encode the dependency between each pair of VMs.

$$D_{x,y} \in \{0, 1\}, \qquad x, y \in \mathcal{VM} \quad (6)$$

$$\forall x \forall y \in \mathcal{D} \quad a^k_{x,v} \wedge a^k_{y,w} \rightarrow l_{v,w} \leq T_D \quad (7)$$

$l_{v,w}$ is the physical distance between PM_v and PM_w.

Constraint 7 checks the dependency between all VMs at every migration step. In some cases, dependency constraint may migrate VMs that are not scheduled for migration to maintain their dependency distance threshold, T_D.

f) Prioritized Migration: The workload characteristics can affect the total migration overhead of all VMs to be migrated [3]. In this work, we will leverage the observations and the conclusions that are suggested in the work done in [3]. A VM can be characterized by its workload as follows: *memory-dirty* workload, *disk I/O* load, *NET I/O* load, *CPU* load, and *web server* workload. Each workload affects the migration process to a certain extent. For example: a web server workload VM affects both the migration time and the migration downtime of its *co-hosted* VMs.

To model these observations, each VM is assigned a priority score based on its workload characteristics; thus, the sequence of migrations can be prioritized to decrease the downtime for VMs. Then, the VMs will be migrated based on their priority score. The lower the priority score, the higher the migration priority. Let o_x represent the priority score for VM_x. The priority score is given based on the observations provided in [3].

$$\forall x \forall y \in \mathcal{M} \quad o_x < o_y \rightarrow \delta^k_{x,v} \wedge \delta^q_{y,w} \wedge k < q \quad (8)$$

Constraint 8 enforces the migration order of VMs; VMs with the highest priority will be migrated before the other VMs. The prioritized migration constraint provides partial information to the solver about the sequence ordering.

g) Critical VMs: Some VMs require specific settings and dedicated hardware. Such VMs are considered as critical VMs. Critical VMs are *non-migratable* VMs and they are not supposed to be migrated. Let η_i represent a Boolean variable to flag critical VMs.

$$\eta_i \in \{0,1\}, \qquad i \in \mathcal{VM} \tag{9}$$

$$\eta_i \rightarrow \sum_{l=0}^{k-1} \delta_{i,j}^{l} = 0 \tag{10}$$

$\delta_{i,j}^{l}$ represents the count of how many times a VM is migrated. Constraint 10 asserts the count of migrations for a critical VM to be *zero*. The critical VMs constraint can be also used and utilized to fix the allocation of VMs that have specific workload; memory-dirty work load for example.

h) Hotspot PMs constraint: The set of hotspot PMs, $\mathcal{H}$, that is scheduled to be switched off should not host any VM at the final placement, π_τ. Let s_j^k indicate the status of a PM at the k^{th} migration step,

$$s_j^k \in \{0,1\} \tag{11}$$

$$\forall j \in \mathcal{H} \quad s_j^\tau \rightarrow \sum_{i \in \mathcal{VM}} a_{i,j}^\tau = 0 \tag{12}$$

The previous constraint ensures that the hotspot physical hosts are excluded at the final migration step, τ, and they are not hosting any VM.

i) Risk Anti-Collocation: The security of a VM in virtualized environments relies on resource isolation among VMs. There is no complete isolation between VMs hosted on the cloud, it is possible to have cross-VM attack from VMs that are collocated on the same host [4]. To provide better secure isolation during the live migration process, we will enforce that there will be no two VMs with conflicting security/risk requirements are hosted on the same PM. During the migration process and before the final allocation of the migrated VMs, VMs's risk requirements may be violated. A VM can be allocated intermediately on a PM that hosts other VMs with conflicting risk requirements. For more details about risk calculation, refer to [2]. For simplicity, we categorized VMs' risk into three different categories: *high*, *medium*, and *low*. Each PM has a risk tag indicating the type of VMs that it can host according to their risk category.

The following constraint ensures that a VM's risk requirement will be maintained during all intermediately steps.

$$\forall k \quad a_{i,j}^k \wedge a_{x,j}^k \rightarrow r_i = r_x \qquad x,i \in \mathcal{VM} \tag{13}$$

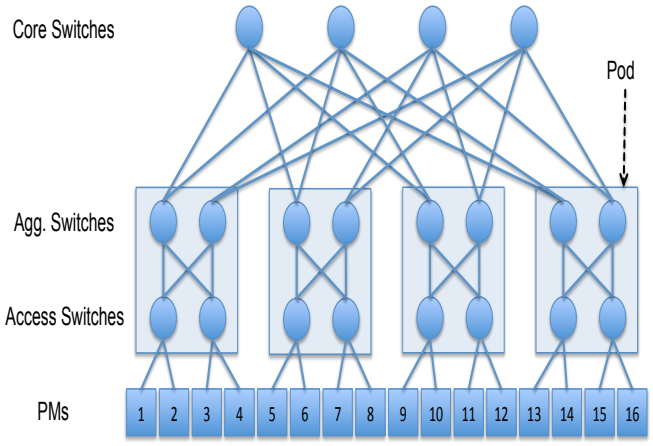

Fig. 2. Fat-Tree Topology (k=4).

where r_i is the risk score for VM_i. In constraint 13, during all migration steps, all VMs that are collocated on the same PM have the same risk score.

VMM-Planner unwraps all the modeled constraints to their valid combinations and gives them to the SMT solver. The solver in its turn assembles the given combinations and finds a satisfiable assignment if there is one. In case there is no satisfiable assignment, the constraints need to be relaxed. The following scenarios can be considered to find a solution in case there was no solution:

- Changing the target placement π_τ: by changing the target placement, it is possible to find a VM migration sequence that leads to π_τ.
- Changing the dependency threshold T_D.
- Allowing VMs to be migrated multiple times.
- Changing the set of critical VMs, $\mathcal{C}$.

The input values such as π_0 and $\mathcal{H}$ are asserted as *satisfied* variables.

The framework can be easily extended to include other constraints to serve different requirements and applications.

IV. Evaluation

A. Experimental Setup

Without loss of generality, we only specify *cpu* and *memory* specification dimensions for the resources regarding the migration process. The datacenter architecture model used in this experiment is fat-tree three-tier topology. We simulated the topology using k-port switches to build 16 PMs datacenters (k=4). Using k-port switches in a fat-tree topology gives $\frac{k^2}{2}$ aggregator switches in total; we have k pods and $\frac{k}{2}$ aggregator switches and access switches in each pod, each pod connects $\frac{k^2}{4}$ PMs with $\frac{k^2}{4}$ core switches [10]. Fig. 2 shows an example of a fat-tree datacenter topology for (k=4). All physical machines have the same specifications, each physical machine has an 8 core CPU and 64 GB memory. All VMs' types used in our simulation are similar to the types used in Amazon EC2 cloud [11].

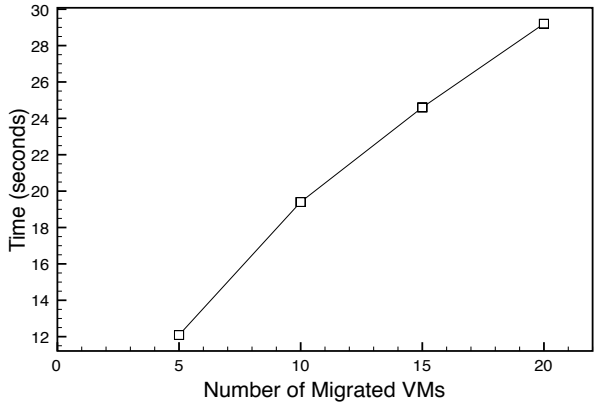

Fig. 3. Running time overhead to find a migration plan.

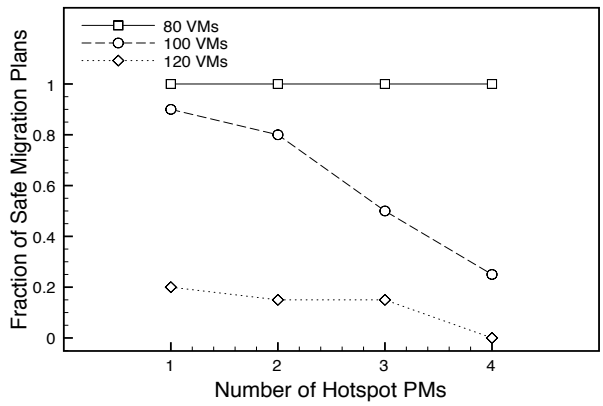

Fig. 4. The effect of number of placed VMs on fraction of violations

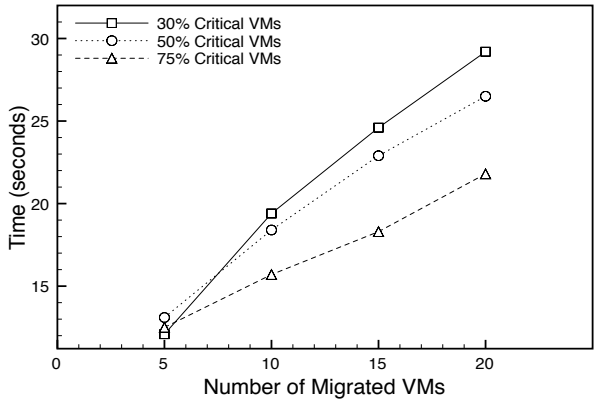

Fig. 5. The effect of percentage of critical VMs on running time overhead

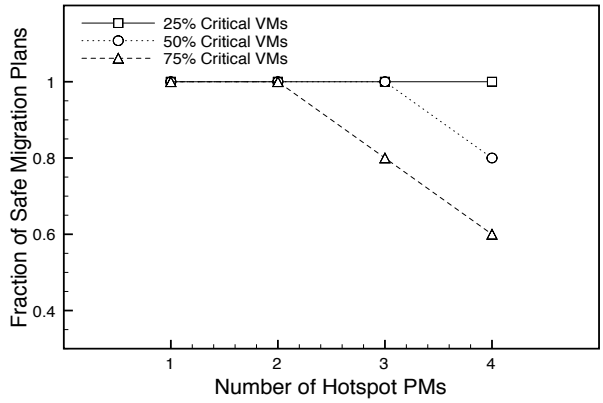

Fig. 6. The effect of the percentage of critical VMs on fraction of violations

All evaluation results were simulated on 3.10 GHz quad core CPU with 16 GB memory; the framework uses only one core because it is not a multi-threaded application. For SMT formalization, we used Yices 1.0.36 SMT solver [12] to encode the presented constraints.

The framework is implemented in C++ and all constraints are asserted by invoking Yices C API [12]. All input data and thresholds for the framework are provided through configuration files, which are then parsed and translated into Yices constraints using API functions provided by the solver.

B. Results

The following set of experiments is conducted to measure the running time overhead to find a migration plan, and to measure the percentage of safety violations for the migrated VMs. These two measures depends on several factors: size of migration set, resource utilization percentage, the percentage of critical VMs, the dependency threshold (T_D), and the dependency percentage between VMs.

The initial placement of all VMs, π_0, is randomly mapped to the available resources. Then, we randomly select the hotspot PMs that will be switched off. The dependency between VMs is randomly assigned.

The impact of migration set size: In this experiment, we evaluate the effect of migration set size on (1) running time overhead to find a satisfiable migration plan, and (2) the fraction of safe migration plans. The fraction of safe migration plans represents the percentage of satisfiable assignments reported by the solver. An unsafe VM migration plan means that some safety requirements are violated. Fig. 3 depicts the time required to find a VM migration plan. In this experiment, we placed 80 VMs evenly on 16 PMs to achieve 5:1 consolidation ratio; then, we select some PMs to be switched off. We set the percentage of critical VMs to be 30% and the dependency threshold (T_D) to be 2. The linear trend is shown in Fig. 3, as the number of migrated VMs increases, the running overhead increases linearly. The effect of total number of VMs placed in the datacenter on finding a successful VM migration plan is shown in Fig. 4. In this experiment, we placed 80, 100, 120 VMs on the datacenter, respectively. Increasing the number of placed VMs means increasing the consolidation ratio, and means less available resources. As shown in Fig. 4, placing more VMs in the datacenter affects finding a satisfiable migration plans. The high resource utilization status means that the system cannot accept more VMs to be allocated.

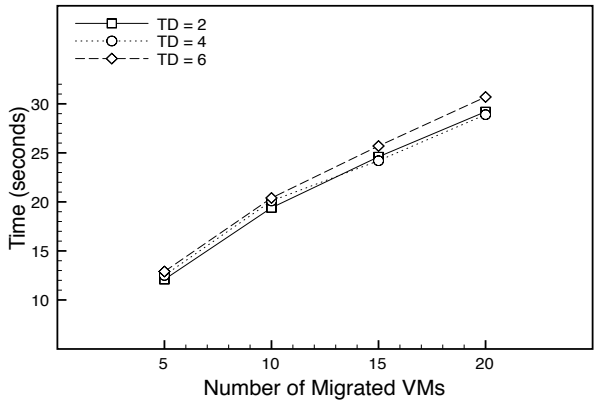

Fig. 7. The effect of dependency threshold (T_D) on running time overhead

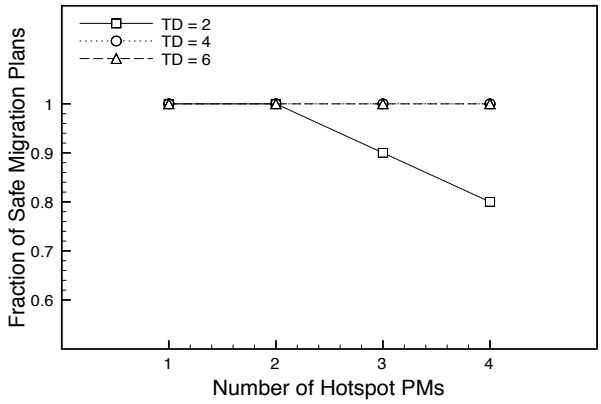

Fig. 8. The effect of dependency threshold (T_D) on fraction of violations

The Impact of critical VMs: The effect of the percentage of critical VMs is evaluated in this set of experiments. A critical VM is a VM that cannot be migrated or reallocated, it has to stay on the same PM all the time. The critical VMs limit the search space for the solver, and adds more restrictions on the migration plan. This in turn, reduces the time required to find a migration plan. Fig. 5 shows that increasing the percentage of critical VMs from 30% to 75% reduces the running time overhead from 31 seconds to 22 seconds. The added restrictions by critical VMs decreases the chances to get a migration plan. This restricts the solver to move the VMs around and assemble the needed resources for the migration plan. Fig. 6 shows that setting 30% of total 80 VMs to be critical VMs does not affect finding satisfiable assignments; while setting the percentage to 75% reduces the percentage of satisfiable migration plans to 60% when switching off 4 PMs (20 VMs to be migrated).

The impact of dependency cost (T_D):

Setting T_D = 2 means that any two dependent VMs have to be accessible to each other using one access switch at most. While setting T_D = 4 means that the dependent VMs are placed in the same pod; otherwise, they can be placed anywhere in the datacenter. In this experiment, we evaluate the effect of changing the dependency threshold (T_D) on the running time overhead and on the fraction of finding a safe migration plan. Fig. 7 depicts that there is no extra overhead added as a result of changing the threshold. On the other hand, changing the dependency threshold affects the percentage of finding a migration plan as shown in Fig. 8. Lowering the dependency threshold (T_D) forces the solver to keep VMs closer to each other during the migration process; in turn, this will limit the search space and decreases the chances of getting a safe migration plan.

V. Related work

There are several works addressed the problem of VMs' migration. The area of VMs migration is driven by different factors such as: resource utilization, energy consumption, cooling and thermal concerns, network and application dependency, etc. Our approach utilizes the existing work that has been done in the area and complements it by providing the VM migration sequence plan.

Piao et al. present a network-aware approach for placing and migrating VMs considering the network condition between PMs [13]. In this approach, VMs are placed and migrated to obtain shorter data access time. Shrivastava et al. propose an approach to migrate VMs that have application dependency between them [14]. They present a greedy algorithm to place VMs on PMs considering the application dependency between VMs and the network topology. Wood et al. [15] propose an approach that uses black-box and grey-box like memory monitoring, CPU utilization, and network utilization to detect hotspot PMs and migrate PMs to suitable hosts. Ma et al. present a memory-encoding approach to decrease the total transferred data, migration time, and migration downtime by identifying the useful memory pages to be transferred [16]. Al Shayeji et al. present an algorithm to save energy in datacenters by migrating VMs from low utilized hosts [17]. Mishra et al. discuss the components of VM migration — when to migrate, which VM to migrate, and where to migrate — and different heuristics to apply VM techniques [18]. Zhang et al. outline the problem of VM migration in over-committed clouds [5]. They introduce an algorithm to minimize the number of VM migrations, to balance VM resource utilization, and to reduce the risk of overload in the cloud. Voorsluys et al present a performance evaluation on the effects of live migration of virtual machines on the performance of applications running on "Xen" VMs [19].

None of these works considers the VM migration sequence planning during the live migration process.

The closest work to our work is done by Ghorbani et al. [9]. They address the problem of determining the order of VM migrations. A heuristic approach is used to assign a migration score for each VM in the migration set. The score for VM_x reflects the number of migrations that would be feasible after the migration of VM_x. The approach considers the bandwidth

limit as a migration constraint. Compared to our approach, we consider several constraints that include: capacity, dependency, security/risk, and VM's workload characteristics.

Li et al. study the live migration features in load balancing scenario [3]. In their experiments, they show the effect of VMs' workload characteristics on the migration time and the migration downtime for different scenarios. We leverage the observations and remarks presented in their work to assign a priority score for each VM based on its workload.

VI. Conclusion

This paper introduced a formal approach for VM migration planning. The framework, *VMM-Planner*, encodes the VM migration planning problem as a Constraint Satisfaction Problem (CSP). The initial VMs placement, the target placement, and the safety conditions are modeled as Boolean constraints. Then, these constraints are given to the SMT solver to find a satisfiable assignment. The satisfied assignment represents a migration sequence the fulfills security, dependency, capacity, and priority requirements. Our analysis and experimental evaluation show that the performance of *VMM-Planner* is adequate to the practical situations. Our approach leverages the existing approaches that identify the VMs to be migrated and where, and complements them by providing the migration sequence plan to execute the migration process.

For future work, we plan to include the security configurations in the migration process. There is some work needs to be done at the area of configuration migration; this can be done by utilizing the migration sequence plan that we have introduced by updating the configurations according the migration sequence plan.

References

[1] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, and C. Hyser, "Renewable and cooling aware workload management for sustainable data centers," in *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '12, (New York, NY, USA), pp. 175–186, ACM, 2012.

[2] S. Al-Haj, E. Al-Shaer, and H. Ramasamy, "Security-aware resource allocation in clouds," in *The 10th International Conference on Services Computing (SCC)*, IEEE, 2013.

[3] X. Li, Q. He, J. Chen, K. Ye, and T. Yin, "Informed live migration strategies of virtual machines for cluster load balancing," in *Proceedings of the 8th IFIP international conference on Network and parallel computing*, NPC'11, (Berlin, Heidelberg), pp. 111–122, Springer-Verlag, 2011.

[4] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-vm side channels and their use to extract private keys," in *Proceedings of the 2012 ACM CCS*, (New York, NY, USA), pp. 305–316, ACM, 2012.

[5] X. Zhang, Z.-Y. Shae, S. Zheng, and H. Jamjoom, "Virtual machine migration in an over-committed cloud," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pp. 196–203, 2012.

[6] Z. Tavakoli, S. Meier, and A. Vensmer, "A framework for security context migration in a firewall secured virtual machine environment," in *Information and Communication Technologies* (R. Szabó and A. Vidács, eds.), vol. 7479 of *Lecture Notes in Computer Science*, pp. 41–51, Springer Berlin Heidelberg, 2012.

[7] S. S. Skiena, *The algorithm design manual*. New York, NY, USA: Springer-Verlag New York, Inc., 1998.

[8] J. Hall, J. Hartline, A. R. Karlin, J. Saia, and J. Wilkes, "On algorithms for efficient data migration," in *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, SODA '01, (Philadelphia, PA, USA), pp. 620–629, Society for Industrial and Applied Mathematics, 2001.

[9] S. Ghorbani and M. Caesar, "Walk the line: consistent network updates with bandwidth guarantees," in *Proceedings of the first workshop on Hot topics in software defined networks*, HotSDN '12, (New York, NY, USA), pp. 67–72, ACM, 2012.

[10] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," INFOCOM'10, (Piscataway, NJ, USA), pp. 1154–1162, IEEE Press, 2010.

[11] "Amazon ec2." http://http://aws.amazon.com/ec2/instance-types.

[12] "Yices smt solver." http://yices.csl.sri.com/index.shtml.

[13] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, pp. 87–92, 2010.

[14] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *INFOCOM, 2011 Proceedings IEEE*, pp. 66–70, 2011.

[15] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proceedings of the 4th USENIX conference on Networked systems design & implementation*, NSDI'07, (Berkeley, CA, USA), pp. 17–17, USENIX Association, 2007.

[16] Y. Ma, H. Wang, J. Dong, Y. Li, and S. Cheng, "Me2: Efficient live migration of virtual machine with memory exploration and encoding," in *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, pp. 610–613, 2012.

[17] M. Al Shayeji and M. Samrajesh, "An energy-aware virtual machine migration algorithm," in *Advances in Computing and Communications (ICACC), 2012 International Conference on*, pp. 242–246, 2012.

[18] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *Communications Magazine, IEEE*, vol. 50, no. 9, pp. 34–40, 2012.

[19] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, (Berlin, Heidelberg), pp. 254–265, Springer-Verlag, 2009.