An Analysis of First Fit Heuristics for the Virtual Machine Relocation Problem

Gastón Keller, Michael Tighe, Hanan Lutfiyya and Michael Bauer
Department of Computer Science
The University of Western Ontario
London, Canada
{gkeller2|mtighe2|hanan|bauer}@csd.uwo.ca

***Abstract*—In recent years, data centres have come to achieve higher utilization of their infrastructure through the use of virtualization and server consolidation (running multiple application servers simultaneously in one physical server). One problem that arises in these consolidated environments is how to deal with stress situations, that is, when the combined demand of the hosted virtual machines (VMs) exceeds the resource capacity of the host. The VM Relocation problem consists of determining which VMs to migrate and to which hosts to migrate them, so as to relieve the stress situations. In this paper, we propose that the order in which VMs and hosts are considered for migration results in better outcomes, depending on the situation and the data centre's business goals. We evaluate and compare a set of First Fit-based relocation policies, which consider VMs and hosts in different order. We present simulation results showing that the policies succeed to different extents depending on the scenario and the metrics observed.**

***Keywords*-dynamic resource management; virtualization; vm relocation; vm migration; greedy heuristics;**

I. Introduction

A data centre can be thought of as a collection of physical servers connected through a high speed, high bandwidth network. The physical servers host applications (or application servers) on behalf of clients who pay rental fees for using the data centre's infrastructure. However, running and maintaining a data centre has costs (infrastructure acquisition, power consumption, etc). Therefore, the business of the data centre relies on maximizing the utilization of its infrastructure while minimizing its expenses [1].

Through the use of systems virtualization [2], data centres are able to increase their resource utilization by consolidating their load. This process involves hosting in a single physical server several application servers, each running within its own virtual machine (VM). The problem of mapping VMs into physical servers (hosts) is referred to in the literature as VM Placement, static server consolidation, or simply resource allocation [3], [4], [5].

The challenge of the VM Placement problem resides in how to maximize resource utilization, while at the same time ensuring individual VMs are allocated enough resources to satisfy their demand. One approach to placing VMs is to assign them to a host based on their expected *average* resource demand. This strategy improves resource utilization by *oversubscribing* the host's resources. In other words, VMs are placed in such a way that the co-located VMs' combined average demand can be accommodated, but the combined peak demand exceeds the host's total capacity. When these *stress situations* occur, that is, when the combined demand of co-located VMs does exceed the resource capacity of the host, one of the VMs has to be migrated to another host, in order to free enough resources in the stressed host to satisfy the resource demand of the remaining VMs. The problem of determining which VM to migrate and to which host to migrate it is known as VM Relocation, or dynamic VM re-allocation [6], [7], [8].

The VM Relocation problem presents similarities with the bin packing problem. [1] There is a set of VMs with varying resource demands (items with their associated weights) that have to be mapped into hosts (packed into bins) in such a way that the overall resource utilization is maximized (the number of bins used is minimized). It has been shown that greedy heuristics for the bin packing problem have near optimal solutions, one such heuristic being *First Fit Decreasing* (FFD) [10], [11]. This heuristic consists of sorting the items in decreasing order by *weight*, and sequentially placing them into the first bin in which they fit. Applied in the context of virtualized data centres, this heuristic would result in minimizing the number of active hosts.

However, while the goal of the bin packing problem is to place a set of items (VMs) in the least number of bins (hosts), the VM Relocation problem may have additional goals for which to strive, such as minimizing the number of Service Level Agreement (SLA) violations or minimizing the number of migrations used. These goals may even take priority over maximizing resource utilization, depending on the situation or the data centre's business goals. For that reason, an heuristic that works well under one situation or business strategy, may not work well under a different situation.

The order in which a heuristic for the VM Relocation problem considers VMs and hosts can affect the final set of migrations produced. Therefore, different heuristics that prioritize VMs and/or hosts based on different criteria may produce better assignments (and achieve better long-term outcomes) when considering their particular goals, associated with the data centre's management strategy.

[1] Though differences exist that make VM Relocation more complex [9].

978-3-901882-48-7

In this work, we evaluate and compare a set of relocation policies, all variations on a *First Fit* heuristic [12], where the difference between policies lies in the order in which VMs and hosts are considered as migration candidates and migration targets, respectively.

The remainder of this paper is organized as follows: Section II discusses related work in the area, Section III describes in detail the VM Relocation problem and the relocation policies designed, Section IV presents experiments and results, which are later discussed in Section V, and Section VI presents conclusions and future work.

II. Related Work

The management of virtualized data centres provides several research challenges. These challenges include the VM Placement and VM Relocation problems (introduced in Section I), the Dynamic Server Consolidation problem (i.e. migrating VMs away from underutilized hosts and into better utilized hosts, so as to increase overall resource utilization in the data centre), and the problem of dynamically provisioning VMs to meet the resource demand of the application servers they host.

The problem of mapping VMs into hosts is referred to in the literature as VM Placement, static server consolidation, or simply resource allocation. Its difficulty resides in finding a mapping that ensures individual VMs are allocated enough resources to satisfy their demand, while at the same time maximizing resource utilization. Bobroff et al. [13] implemented a First Fit Decreasing heuristic that periodically re-calculated the mapping of VMs to hosts, based on the VMs' forecasted demand. Their goal was to minimize the average number of active hosts, while providing probabilistic SLA guarantees for the VMs. Cardosa et al. [3] developed a VM placement algorithm that leveraged the CPU allocation features `min` , `max` , and `shares` present in modern hypervisors to negotiate the tradeoff between VMs' performance (tied to CPU allocation) and overall power consumption. Speitkamp and Bichler [4] proposed a static server consolidation approach that combined data analysis to characterize variations in real-world workload traces, and an LP-relaxation-based heuristic to optimally map VMs into hosts. Stillwell et al. [5] worked on mapping a set of static workloads into hosts, optimizing the VMs' resource allocation for performance and fairness. They proposed and evaluated an extensive set of algorithms, finally identifying a vector (or multi-dimensional) bin packing algorithm that achieved close to optimal solutions to the problem.

The problem of determining which VM to migrate and to which host to migrate it when a stress situation occurs has also been studied. Khanna et al. [6] addressed the VM Relocation problem by developing a mathematical optimization model to select VMs and hosts for migration. They implemented an heuristic that sorted VMs in increasing order by CPU and memory utilization - to minimize migration costs - and sorted hosts in increasing order by residual capacity (i.e. available resources) - to maximize resource utilization. The authors did not consider any additional sorting strategies, nor the impact of their heuristic in the number of migrations issued. Wood et al. [7] implemented in their management system Sandpiper a First Fit Decreasing heuristic that sorted hosts in increasing order by resource utilization. Their goal was not, however, to evaluate the efficiency of their heuristic, but to compare two mechanisms for VM monitoring. Gmach et al. [8] developed a fuzzy logic-based controller in their management system. The controller not only issued migrations when a host became stressed (VM Relocation), but also when a host became underutilized (Dynamic Server Consolidation). The target host for a migration was the least loaded host with enough resources to fit the VM. However, it was not clear how the VMs were selected for migration. Beloglazov and Buyya [14] proposed algorithms and heuristics to deal both with stressed and underutilized hosts. In the case of stressed hosts, their algorithm selected for migration the VMs with the least memory allocation and selected as many VMs as needed to bring the hosts' CPU utilization down to an acceptable level. When hosts were underutilized, all their VMs were selected for migration. The target host selection was based on a Best Fit Decreasing heuristic: the migrating VMs were sorted in decreasing order by CPU utilization and placed in the host that provided the least increase in power consumption due to the allocation of the incoming VM. The use of a Best Fit approach makes target sorting strategies irrelevant at the cost of having to consider every single host in the data centre for each VM that has to be placed.

On the topic of dynamic VM provisioning, Gmach et al. [15] developed and compared four different resource reallocation policies, with the goal of minimizing the number of active hosts while providing Quality of Service to the VMs. The authors concluded that work-conserving policies with dynamically set weights offered the best results. Pokluda et al. [16] focused on dynamic memory management. They developed a policy-based framework that would reallocate memory among the VMs co-located in a host to meet the VMs' changing demand. The system could also trigger VM migrations were the local memory adjustments insufficient to deal with the stress situation.

III. VM Relocation

In this section we describe the VM Relocation problem (Subsection III-A), the assumptions and limitations we work with (Subsection III-B), and the relocation policies designed to test our hypothesis (Subsection III-C).

A. Problem Definition

The VM Relocation problem consists of the following: given a set of *stressed*, *non-stressed* and *suspended* hosts, finding a set of VM *relocations* that will eliminate the *stress situations*.

Hosts (or physical servers) can be in *active* or *suspended* state. Hosts in active state host at least one VM; otherwise, they are suspended to conserve power. Active hosts are further classified as *stressed* or *non-stressed*, as determined by their

resource utilization level. A $stress$ threshold defines the division between categories. [2]

A host is resource stressed (or in a resource stress situation) when the hosted VMs' combined demand for that resource reaches or exceeds the total capacity of the host. In this situation, the host is unable to satisfy the resource demand of the hosted VMs, negatively affecting their performance. A host with enough resources to satisfy the VMs' combined demand may still be considered stressed if the utilization level is so high that the host is unable to accommodate any further increases in the VMs' resource demand.

VMs can be *relocated* in order to release resources in their current host. There are two forms of relocation: *migration* and *replication*. Migration is the transfer of a VM from its current host to a different host. Replication is the instantiation of a (pre-stored) copy of a VM in another host.

The VM Relocation problem is similar to the *m-dimensional bin packing problem*, which is known to be NP-hard [9]; there are n VMs to be placed (items), p hosts (bins), and at least three dimensions to consider: CPU, memory and bandwidth. Additional constraints only add complexity to the problem: the initial placement of the VMs, dependencies between VMs, an upper limit to the number of relocations that can be found to solve the problem (so as to limit the overhead), heterogeneous hosts, etc.

B. Assumptions and Limitations

In order to limit the scope of the VM Relocation problem, we have defined the following assumptions and limitations:

1) **The set of physical servers is homogeneous (in terms of resource capacity and power consumption).**
2) **Hosts' utilization level calculation is based only on CPU utilization.**
3) **The virtual machines are independent from each other (i.e. no dependencies).**

Item 1 proposes that all physical servers be homogeneous in terms of their resource capacity and power consumption (effectively making all hosts equal under these two factors). Therefore, when selecting source or target hosts, resource capacity (as in total number of CPU cores, total memory, etc) and power consumption can be safely ignored in the decision process.

Item 2 proposes that CPU be the only resource considered when determining the utilization level of a host. Therefore, whether a host is stressed or not will depend solely on its CPU utilization level. A host could have all its memory allocated and in use, and still not be considered stressed (i.e. memory stress is not considered a stress situation). Nonetheless, memory and bandwidth needs are still taken into account when trying to place or relocate a VM, although simply as a last minute check that the target host has enough resources to meet the needs of the incoming VM. In this study, both memory and bandwidth will be constant and the same values for all VMs, effectively reducing the number of dimensions in the problem.

Item 3 proposes that there be no dependencies between VMs. Dependencies may impose restrictions of the form *"virtual machine x must to be hosted together with virtual machine y"* or *"virtual machines x and y cannot be hosted in the same physical server (or even rack or cluster)."* Removing these restrictions simplifies the problem by making it easier to select VMs to relocate and target hosts for those relocations.

Although these assumptions and limitations do simplify the VM Relocation problem, the remaining problem is still challenging. Many factors still remain to be considered and prioritized when searching for relocations, such as size (CPU utilization level) of the VMs, utilization level of the source and target hosts before and after relocation, relocation overhead on the source and target hosts, number of concurrent relocations per host, and number of active hosts in the data centre.

C. Policies

The relocation policies are based on a First Fit heuristic [12]. This heuristic consists of taking the items that need to be placed, one at a time, and assigning them to the first bin in which they fit. The relocation policies differ from each other in the order in which they consider the VMs (items) and the target hosts (bins) for their migrations.

In order to distinguish between hosts, we use the active host classification introduced in Section III-A, though refined as follows. Active hosts are classified as *stressed*, *partially utilized* or *underutilized*, based on their resource utilization level. Two thresholds define the division between categories: $stress$ and $minUsage$ (high and low level thresholds, respectively). Since we only consider CPU to calculate the utilization level of a host (as per Item 2 in Section III-B), the thresholds are renamed as $stress_{CPU}$ and $minUsage_{CPU}$. The categories are formalized as follows:

- **Underutilized**: hosts with CPU utilization in the range $[0, minUsage_{CPU}]$;
- **Partially-utilized**: hosts with CPU utilization in the range $(minUsage_{CPU}, stress_{CPU}]$;
- **Stressed**: hosts with CPU utilization in the range $(stress_{CPU}, 1]$.

Suspended hosts form a category of their own, **Suspended**.

The relocation policies have to find sets of VM relocations. Each relocation consists of three elements: a *source host*, a *candidate VM*, and a *target host*. Source hosts are selected from among **Stressed** hosts, candidate VMs are selected from among the VMs hosted in the selected source hosts, and target hosts are selected from among **Suspended**, **Underutilized** and **Partially-utilized** hosts.

As mentioned before, the relocation policies are based on a First Fit heuristic (see Algorithm 1). The first step (line 1) in the heuristic classifies the hosts in their respective categories. The second step (line 2) removes from among the source hosts those hosts that are currently involved in relocations. Once the relocations are completed, the utilization level of the hosts will change and the stress situations may cease to exist. If

[2]Threshold value would depend on the physical servers' resource capacity and the data centre's business objectives.

```
1: classify hosts in categories
2: filter and sort source hosts
3: sort target hosts
4: for each source host do
5:    filter and sort candidate VMs
6:    for each candidate VM do
7:       for each target host do
8:          try to migrate VM to host
9:          if success then
10:            record migration
11:            move on to next source host
12:         end if
13:      end for
14:   end for
15: end for
16: return list of migrations
```

Algorithm 1: First Fit heuristic.

the stress situations persist, they will be addressed in the next cycle. The remaining hosts are then sorted in decreasing order by CPU utilization. This sorting process is common to all policies (except *Random*, introduced below).

The third step (line 3) sorts the target hosts. We defined three different ways of combining the categories from which a target host is to be selected:

1) **Increasing**: sort Partially-utilized and Underutilized hosts in increasing order by CPU utilization. Consider Underutilized, Partially-utilized and Suspended hosts, in that order;
2) **Decreasing**: sort Partially-utilized and Underutilized hosts in decreasing order by CPU utilization. Consider Partially-utilized, Underutilized and Suspended hosts, in that order;
3) **Mixed**: sort Partially-utilized hosts in increasing order by CPU utilization and Underutilized hosts in decreasing order by CPU utilization. Consider Partially-utilized, Underutilized and Suspended hosts, in that order.

The fifth step (line 5) filters and sorts the VMs hosted in the selected source host. Only VMs with equal or greater CPU load than the CPU load by which the host is stressed are considered as migration candidates. If no VM satisfies this criteria, all the VMs are considered. This filtering is done to improve the chances of selecting for migration a VM that will bring the source host's utilization level below the $stress_{CPU}$ threshold. After filtering, the VMs are sorted in one of two ways:

A) **Decreasing**: sort VMs in decreasing order by CPU load; [3]

B) **Increasing**: sort VMs in increasing order by CPU load.

The eighth step (line 8) checks whether the selected target host has enough spare resources to accept the migration of the selected candidate VM. If this check turns positive, the migration is recorded (line 10) and the process moves on to the next source host (line 11). Be it noted that at most one migration is issued for each source host.

[3]Be it noted that CPU load is not the same as CPU utilization. The first term refers to the actual number of CPU shares in use, while the second term refers to the ratio of CPU shares in use over total allocated CPU shares.

TABLE I
VM RELOCATION POLICIES

Policies	VM sorting	Target sorting
FFDI	A	1
FFDD	A	2
FFDM	A	3
FFII	B	1
FFID	B	2
FFIM	B	3

By exhaustively combining the two sorting strategies for migration candidates and the three sorting strategies for target hosts, we obtain six different relocation policies (shown in Table I).

We implemented a seventh policy, *Random*, which randomizes the order in which source hosts, migration candidates and target hosts are considered. This policy was added as a benchmark.

A final note regarding host selection in these policies: a host cannot be selected both as migration source and target. Hosts can be source for one migration at a time, but can be target for several migrations concurrently (as long as they have enough resources to satisfy the resource requirements of all incoming VMs).

IV. EXPERIMENTS

We used the simulation framework DCSim [17], [18] to conduct our experiments. Though simulations do not capture the whole complexity of real-world environments, they allow for replicable testing environments (very important when comparing algorithms), enable large-scale experiments, and significantly reduce the real-time duration of experiments.

Subsection IV-A discusses the configuration of the simulation environment, Subsection IV-B describes the experiments' design, and Subsection IV-C presents the results.

A. Simulator Configuration

DCSim (Data Centre Simulator) is an extensible data centre simulation framework, designed to provide an easy framework for developing and experimenting with data centre management techniques and algorithms [18]. Several components in DCSim have to be extended or implemented to evaluate alternative management policies. We worked with DCSim, version 12.01, in these experiments.

In DCSim, a data centre has a VM Placement Policy to map VMs into hosts at creation time, a VM Consolidation Policy to perform server consolidation, and a VM Relocation Policy to relocate VMs away from stressed hosts.

In our experiments we use a simple VM Consolidation Policy that migrates VMs from underutilized hosts to partially

utilized or higher-loaded underutilized hosts. The use of a VM Consolidation policy was necessary to allow for stress situations to continue occurring throughout the simulation. Any solution to the VM Relocation problem, by definition, attempts to dissipate stress situations by migrating VMs away from stressed hosts. Unless the number of hosts in the set is so small that VMs are simply migrated in circles, the VM Relocation policy will unavoidably spread the load (VMs) across the data centre. Server consolidation policies exist to remedy this situation, and in this particular case, to better allow us to compare the VM Relocation policies introduced in Section III-C.

Policies are also used in DCSim to manage the allocation (and reallocation) of resources among the hosted VMs. Static Resource Managers allocate resources to VMs upon placement in the host and do not alter the allocation at any further point. Dynamic Resource Managers, on the other hand, perform an initial allocation based on the resource request of the VMs and then dynamically adjust the allocation to match the VMs' resource utilization.

In our experiments, we use an oracle-like CPU Manager. This manager is a special kind of dynamic CPU Manager with perfect knowledge about the VMs' resource needs at any given time and can therefore compute a perfect allocation. The use of this CPU Manager allows us to perform resource reallocation in real time to meet the resource needs of the VMs without overprovisioning them. Also, the oracle nature of this manager helps us to prevent ineffective CPU allocation policies from affecting (and obscuring) the results of the VM Relocation policies under study.

B. Design

We designed four sets of seven experiments to evaluate the relocation policies. Every experiment in a set used the same data centre configuration, but a different VM Relocation policy. Each experiment lasted 10 simulation days and was repeated 5 times. Results were averaged.

For the first set of experiments, the data centre was configured with 100 hosts, and was set to run the VM Relocation process every 10 minutes and the VM Consolidation process every 24 hours. The data centre hosted 300 VMs, initially provisioned to meet their peak demand. (When provisioned for peak demand, a host can only fit 3 VMs, so 300 is the maximum number of VMs that can be placed in this data centre.)

The second, third and fourth sets of experiments utilized the same data centre configuration, but hosted 400, 452 [4] and 500 VMs, respectively, which were initially provisioned to meet their average demand.

The hosts in these experiments had 4 CPU cores with 1000 CPU shares each (for a total of 4000 CPU shares per host) and 8GB of memory. Bandwidth and storage were not considered in these experiments. Each host was restricted to at most two concurrent outgoing migrations. The $stress_{CPU}$ threshold was set at 85% and the $minUsage_{CPU}$ threshold at 50%. There were 400 CPU shares reserved by the CPU Manager to cover migration overhead (200 shares * 2 concurrent migrations), thus effectively reducing the CPU shares available for allocation to 3600.

Each VM was attached to a separate instance of one of four Application Models, with an equal number of VMs attached to each type of model. All four Application Models worked in a similar way, but used different workload traces as input. The first two used the ClarkNet HTTP trace and the EPA HTTP trace available from the Internet Traffic Archive [19]. The second two used the Google Cluster Data trace [20]. Google Cluster Data is divided into four job types, from which we extracted the first two as individual workload traces. For each workload trace, we used the total number of incoming requests in 100 second intervals as the workload level, which was normalized to the range [0, 1]. Workload traces that were shorter than the simulation time were looped, and each virtual machine started the trace at a randomly chosen offset time value to differentiate them from each other. The virtual machines were placed in the hosts by the VM Placement Policy in a randomized order, with a different random order for each execution of the simulation.

VMs were limited to a maximum CPU need of 1000 shares (equal to 1 core of the hosts described above), calculated using the formula $100 + 900 * workload$. In the experiment in which VMs' initial allocation was set to *peak*, the allocation was 1000 CPU shares. When it was set to *average*, the allocation was the average workload level of the trace to which the VM was attached. Memory needs for the VMs were fixed at 1GB, and bandwidth and storage were not considered.

In order to compare the relocation policies, we used the following set of metrics provided by DCSim:

- **Number of migrations.** VM migrations that were issued by the VM Relocation and VM Consolidation policies in use during the simulation.
- **Average Active Hosts.** The average number of hosts that were in the active state during the simulation.
- **Host-hours.** The combined active time of each host in the simulation (measured in hours).
- **Host Utilization.** The average utilization of active hosts (hosts in suspended state do not contribute to the average).
- **Kilowatt-hours.** The power consumption of the hosts during the simulation. Suspended hosts are assumed to have zero power consumption. Power consumption of active hosts is calculated as follows:

$$250watts + 250watts * host.utilization \quad (1)$$

- **Dropped Requests.** The average resource demand of the virtual machines that could not be satisfied by the data centre. This unmet demand is an indicator of the Quality of Service achieved and is interpreted as the ratio of lost requests to total requests.

[4] 452 VMs were used instead of 450, so as to have an equal number of VMs associated to each of the four available Application Models.

C. Results

The results of the experiments are presented in Tables II, III, IV and V. Results were averaged over 5 repetitions of each experiment and the standard deviation is shown in square brackets. The first day of simulation time was discarded to eliminate the influence of the initial VM placement and allow the system to stabilize before collecting metrics.

Policy FFDI used consistently the most hosts, consumed the most power, and achieved the lowest host utilization. On the other hand, it also achieved the best service (i.e. lowest percentage of dropped requests) and required the least number of migrations.

Policy FFID behaved opposite to FFDI. It achieved consistently the highest host utilization and used the least hosts; consequently, it also reported the lowest power consumption. However, it paid the price dropping the most requests and triggering the most migrations.

Policies FFII, FFIM and FFDD achieved average results (between FFDI and FFID), not excelling under any particular metric. FFDM followed one step behind in terms of using less hosts, less power and achieving higher utilization, but it performed fewer migrations in doing so.

The Random policy behaved – predictably – random. It did not excel in any category, but it did not trigger the most migrations and it did achieve good service.

V. Discussion

The results presented in the previous section show that no one policy scored best in every metric, which is to be expected when considering conflicting goals, such as minimizing the number of active hosts and minimizing the number of dropped requests.

The experiments also show that the policies succeeded to different extents depending on the scenario and the metrics observed. For example, policy FFDM used consistently more hosts and consumed more power than policies FFII and FFIM. However, in the last experiment, FFDM achieved similar results under both metrics, while issueing 50-60% the number of migrations.

With regard to the sorting strategies used, we can make a few observations. By sorting VMs in decreasing order (policies FFD), the first VMs to be considered for migration (and likely migrated) are high-load VMs. Consequently, source hosts see a significant drop in their CPU utilization (up to 25% if the VM was using one whole core) and more hosts are activated due to the difficulty of finding target hosts that can receive a high-load VM.

On the other hand, by sorting VMs in increasing order (policies FFI), low-load VMs [5] are considered first. These VMs have a smaller impact on the source hosts' utilization (thus keeping utilization high) and its likely easier to find a partially utilized host that can receive them. The downside of this situation is that since source hosts are kept highly utilized and partially utilized hosts see an increase in their utilization with the arrival of migrated VMs, both source and target hosts are at a higher risk of becoming stressed again in the near future, therefore causing more migrations in the long run.

From the viewpoint of the target hosts sorting strategies, an anomaly is observed. Policies FFDI and FFII are the most liberal, looking for suitable target hosts among the lowest utilized hosts first, while policies FFDD and FFID are the most conservative, trying to migrate VMs to highly utilized hosts first. Finally, policies FFDM and FFIM represent a compromise, searching for suitable hosts among the lowest partially utilized hosts first, thus trying to keep an overall high utilization without being too conservative. This observation appears to hold true for the FFD policies and FFID. However, policies FFII and FFIM seem to have switched roles, the former using less hosts and achieving higher utilization than the latter.

Another interesting observation is that the highest host utilization achieved in these experiments was only 72.6% when the stress threshold (and therefore the utilization goal) was set to 85%. A careful analysis of the restrictions and resource allocation policies in place inform us that the maximum host utilization the management policies could hope to achieve is 76.5%. This conclusion follows from two premises. First, that the maximum CPU shares available for allocation in a host is 3600 (as indicated in Section IV-B). Second, that the CPU Manager Oracle allocates CPU to VMs in such a way that the VMs' utilization is 85% of their CPU allocation. Therefore, even if a host has 3600 CPU shares allocated, only 3060 would normally be in use, which is equivalent to a utilization of 76.5%. Therefore, even if utilization could actually increase up to 90% (i.e. using the 3600 CPU shares allocated), it would be completely dependent upon the workload of the hosted VMs; the management policies would not be able to force the utilization higher by placing more VMs into the host.

This work presents a series of shortcomings. Some of these shortcomings were purposefully introduced as assumptions or limitations to limit the scope of the VM Relocation problem (see Section III-B). Other shortcomings were imposed by the simulation framework. [6]

First of all, the cost (or overhead) of VM migrations. Migrations impose an overhead in both source and target hosts. In this experiments, only CPU overhead was accounted for (the CPU Manager considered a fixed overhead of 200 CPU shares per migration in both hosts). Memory and bandwidth overhead was ignored at the hosts, and bandwidth overhead on the networking infrastructure was equally neglected, assuming a dedicated network for migrations.

Second, no delay or cost was associated to the host activation and suspension processes, enabling migrations to suspended hosts to start immediately.

Finally, DCSim provided an advantage that would not hold true in a real-world scenario: the data centre management

[5] Though in most cases, these VMs would have enough load to terminate the stress situation; otherwise, they would have been filtered out.

[6] Several of these shortcomings have been addressed already in a newer version of DCSim.

TABLE II
EXPERIMENT 1 - 300 VMS

Metrics	FFDI	FFDD	FFDM	FFII	FFID	FFIM	Random
# Migrations	480 [17.7]	681 [47]	577 [25.6]	530 [29.2]	1755 [110]	766 [109]	909 [48.2]
Avg. Active Hosts	70.2 [1.2]	64.8 [0.6]	65.6 [0.9]	79.7 [4]	61.5 [0.5]	64.3 [0.5]	72.9 [0.4]
Host-hours	15209 [82.8]	14683 [76.9]	14847 [103]	14765 [319]	13634 [70.4]	14487 [29.8]	17627 [196]
Host Utilization	61.5% [0.3]	63.5% [0.3]	62.8% [0.3]	63.3% [1.2]	67.6% [0.3]	64.3% [0.1]	53.3% [0.5]
Kilowatt-hours	6142.9 [22.1]	6002.7 [20]	6046.5 [27.4]	6029.7 [82.2]	5715.3 [19]	5953.5 [7.2]	6760.6 [50.5]
Dropped Requests	0.6% [0.04]	0.9% [0.04]	0.8% [0.04]	0.6% [0.05]	1.7% [0.06]	0.8% [0.06]	0.7% [0.02]

TABLE III
EXPERIMENT 2 - 400 VMS

Metrics	FFDI	FFDD	FFDM	FFII	FFID	FFIM	Random
# Migrations	478 [74.3]	737 [21.3]	585 [42.2]	814 [120]	2315 [90.7]	826 [59.3]	1015 [116]
Avg. Active Hosts	88.6 [0.6]	84.7 [0.8]	86.4 [0.5]	82.9 [0.4]	78 [0.3]	84.7 [0.5]	93 [0.7]
Host-hours	19318 [561]	18374 [120]	18695 [75]	18029 [178]	16928 [29.4]	18314 [127]	21094 [200]
Host Utilization	65.3% [0.5]	67.5% [0.4]	66.4% [0.2]	68.8% [0.6]	72.5% [0]	67.8% [0.4]	59.2% [0.5]
Kilowatt-hours	7879.6 [40.9]	7696.4 [31.6]	7781.8 [19.8]	7610.8 [46.3]	7300.6 [9]	7684.2 [33.3]	8397.6 [51.9]
Dropped Requests	0.6% [0.02]	0.8% [0.03]	0.7% [0.02]	0.7% [0.02]	1.7% [0.07]	0.7% [0.03]	0.7% [0.02]

TABLE IV
EXPERIMENT 3 - 452 VMS

Metrics	FFDI	FFDD	FFDM	FFII	FFID	FFIM	Random
# Migrations	513 [72]	865 [49.7]	669 [73.5]	960 [125]	2527 [149]	929 [66.4]	652 [52.3]
Avg. Active Hosts	99.6 [0.2]	95.6 [0.7]	97.8 [0.1]	92.8 [0.2]	88.4 [0.3]	95.6 [0.7]	99.3 [0.2]
Host-hours	21845 [570]	20775 [138]	21188 [41.4]	20187 [88.8]	19172 [72.8]	20696 [85.2]	21589 [5.53]
Host Utilization	65.1% [0]	67.4% [0.4]	66.3% [0.1]	69.4% [0.2]	72.3% [0.2]	67.8% [0.2]	65.2% [0]
Kilowatt-hours	8917.2 [1.6]	8699.5 [36.3]	8810.5 [10.3]	8550 [23.5]	8262 [20.6]	8682.7 [21.1]	8917.2 [2]
Dropped Requests	0.6% [0.04]	0.8% [0.04]	0.7% [0.04]	0.8% [0.04]	1.6% [0.06]	0.7% [0.02]	0.6% [0.03]

TABLE V
EXPERIMENT 4 - 500 VMS

Metrics	FFDI	FFDD	FFDM	FFII	FFID	FFIM	Random
# Migrations	1242 [26.8]	1540 [93.9]	832 [28.1]	1665 [70.5]	3114 [249]	1371 [72.6]	1324 [69.3]
Avg. Active Hosts	100 [0]	100 [0]	100 [0]	100 [0]	97.5 [0.6]	100 [0]	100 [0]
Host-hours	23658 [30.3]	23585 [69.1]	23658 [53.5]	23088 [130]	21111 [92.6]	23398 [20.4	23745 [13.9]
Host Utilization	71.4% [0.01]	71.4% [0.02]	71.4% [0.04]	71.4% [0.02]	72.6% [0.28]	71.3% [0.02]	71.4% [0.01]
Kilowatt-hours	9259 [0.6]	9255.4 [1.1]	9255.9 [2.2]	9258.2 [1.2]	9110.4 [25.1]	9254.5 [1.4]	9258.4 [1]
Dropped Requests	1.1% [0.02]	1.2% [0.03]	1.2% [0.05]	1.1% [0.03]	1.7% [0.07]	1.2% [0.03]	1.1% [0.02]

system (and policies) possessed perfect knowledge of the utilization levels of hosts and VMs at all times. In a real system, status information would be sent periodically from hosts to management system, with two consequences: first, the information would not be always up-to-date, and second, the status update messages would introduce an overhead in the networking infrastructure, whose extent would depend on the amount of information transferred and the frequency of such transmissions.

VI. CONCLUSIONS AND FUTURE WORK

The goal of this work was to determine whether changing the order in which a relocation policy considered the candidate VMs and target hosts for migration resulted in better outcomes (in terms of data centre management metrics), depending on the situation or the data centre's business goals.

The experimental results (see Section IV) showed that no one policy scored best in every metric, and that the policies succeeded to different extents depending on the scenario and the metrics observed.

These results demonstrate that one single policy will not

satisfy all goals and that by tweaking the VM and host sorting strategies better trade-offs can be achieved. Most importantly, the results suggest that dynamically switching between policies may offer better overall results.

As future work, we plan to focus on removing the assumptions and limitations introduced in Section III-B, as well as the shortcomings described in Section V, and evaluating the merits of dynamically switching between policies. Another avenue of research could be to look into fuzzy logic-based controllers for the tasks of target host and candidate VM selection.

ACKNOWLEDGEMENTS

We thank the National Sciences and Engineering Research Council of Canada (NSERC) for their support.

REFERENCES

[1] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, pp. 7–18, 2010.

[2] J. E. Smith and R. Nair, "The architecture of virtual machines," *Computer*, vol. 38, no. 5, pp. 32–38, 2005.

[3] M. Cardosa, M. R. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *IM'09: Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Network Management, 2009*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 327–334.

[4] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *Services Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 266 –278, oct.-dec. 2010.

[5] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova, "Resource allocation algorithms for virtualized service hosting platforms," *J. Parallel Distrib. Comput.*, vol. 70, no. 9, pp. 962–974, Sep. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2010.05.006

[6] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application performance management in virtualized server environments," in *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, 2006, pp. 373–381.

[7] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proceedings of the Fourth Symposium on Networked Systems Design and Implementation (NSDI'07)*, Cambridge, MA, USA, Apr. 2007, pp. 229–242.

[8] D. Gmach, J. Rolia, L. Cherkasova, G. Belrose, T. Turicchi, and A. Kemper, "An integrated approach to resource pool management: Policies, efficiency and quality metrics," HP Laboratories Palo Alto, Palo Alto, CA, USA, Tech. Rep. HPL-2008-89, 2008.

[9] C. Hyser, B. Mckee, R. Gardner, and B. J. Watson, "Autonomic virtual machine placement in the data center," HP Laboratories, Palo Alto, CA, USA, Tech. Rep. HPL-2007-189, Dec. 2007.

[10] M. Yue, "A simple proof of the inequality FFD (L) $\leq$ 11/9 OPT (L) + 1, $\forall$ L for the FFD bin-packing algorithm," *Acta Mathematicae Applicatae Sinica (English Series)*, vol. 7, pp. 321–331, 1991, 10.1007/BF02009683. [Online]. Available: http://dx.doi.org/10.1007/BF02009683

[11] G. Dsa, "The tight bound of first fit decreasing bin-packing algorithm is FFD (I)$\leq$11/9 OPT (I)+6/9," in *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, ser. Lecture Notes in Computer Science, B. Chen, M. Paterson, and G. Zhang, Eds. Springer Berlin / Heidelberg, 2007, vol. 4614, pp. 1–11, 10.1007/978-3-540-74450-4_1. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74450-4_1

[12] M. Y. Kao, *Encyclopedia of Algorithms*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[13] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *IM'07: Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management, 2007*, 2007, pp. 119–128.

[14] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, pp. 1–24, 2011. [Online]. Available: http://dx.doi.org/10.1002/cpe.1867

[15] D. Gmach, J. Rolia, and L. Cherkasova, "Satisfying service level objectives in a self-managing resource pool," HP Laboratories Palo Alto, Palo Alto, CA, USA, Tech. Rep. HPL-2009-170, Jul. 2009.

[16] A. Pokluda, G. Keller, and H. Lutfiyya, "Managing dynamic memory allocations in a cloud through golondrina," in *Proceedings of the 4th International DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM'10)*, Oct. 2010.

[17] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, "DCSim: A data centre simulation tool for evaluating dynamic virtualized resource management," in *Proceedings of the 6th International DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM'12)*, Oct. 2012.

[18] DCSim project site. Distributed and Grid Systems (DiGS) Research Group, The University of Western Ontario. [Online]. Available: http://digs.csd.uwo.ca/digs/

[19] (2012, Aug.) The internet traffic archive. [Online]. Available: http://ita.ee.lbl.gov/

[20] (2012, Aug.) Google cluster data. Google Inc. [Online]. Available: http://code.google.com/p/googleclusterdata/