# The Right Tool for the Job: Switching Data Centre Management Strategies at Runtime

Graham Foster*, Gastón Keller*, Michael Tighe*, Hanan Lutfiyya and Michael Bauer
Department of Computer Science
The University of Western Ontario
London, Canada
{dfoste2|gkeller2|mtighe2|hanan|bauer}@csd.uwo.ca

***Abstract*—Applications are shifting into large scale, virtualized data centres that provide resources on a pay-per-usage basis. Data centres must minimize resource consumption while providing enough resources to meet application requirements. To meet highly variable application demands, a dynamic approach to virtual machine (VM) management is required. This involves three basic management operations: (i) placing newly arrived VMs, (ii) migrating (moving) VMs off of highly utilized machines to avoid performance degradation, and (iii) migrating VMs off of underutilized machines so that they may be shut down to save power. We define a management strategy to consist of a set of policies that guide these three operations. We consider the goals of minimizing Service Level Agreement violations and minimizing power consumption. Developing a management strategy to achieve both of these goals is challenging, as the goals are often in conflict. We propose achieving both goals through dynamically switching between two management strategies, each with a single goal, depending on current data centre state. We propose three methods of dynamically switching strategies, and evaluate these methods through simulation. Dynamic strategy switching offers improved results over a single management strategy.**

***Keywords*—*data center management; virtualized infrastructure management; cloud management; energy management; SLA management***

## I. Introduction

Computing today is shifting into large-scale data centres that provide access to computing resources for client applications on a pay-per-usage basis. Data centres face the challenge of minimizing resource consumption while providing sufficient resources to meet application requirements. One approach to resource allocation is to statically allocate enough resources to meet the peak demand of an application. However, the computing resources needed by an application often have high variability [1]. This can lead to a significant over-provisioning, resulting in underutilized resources. Virtualization allows for smaller units of resource to be allocated by using a single physical machine (*host*) to host multiple virtual machines (VMs), each hosting a client application. If resources are still allocated for peak demand, however, then the physical machine may still be highly underutilized. Utilization can be increased by allocating only enough resources to meet average demand. This, however, can result in VMs being forced to compete for resources when demand increases. Since the overall utilization of a host is high, an increase in demand for an application can result in the VM requiring resources that are already in use by another co-located VM, thus leading to a degradation in application performance.

If the VMs are hosting applications with known demands, then a static allocation (placement) of VMs may be applicable. Static allocation, for example, can be modelled as a vector bin packing problem [2], [3] and can typically be solved using linear programming techniques. This solution can accommodate changes in long term workload distribution of the applications being hosted by VMs. However, many applications have highly variable demands and there may be frequent changes in the set of VMs [4], thus necessitating a more dynamic approach. There is work that considers variable demand by periodically re-calculating the mapping of VMs to hosts using linear programming techniques. However, these approaches generally do not scale well [5] or are not responsive enough. For dynamic management, Stillwell et al. [3] have shown that variants of First Fit heuristics for vector bin packing work best for large-scale systems.

Dynamic management can address the utilization problem by taking advantage of the ability to migrate (move) a running VM from one physical host to another (*live migration*). More generally, dynamic management of VMs entails a coordinated use of three operations: (i) *VM Placement (Allocation)*: the placement of a VM on a host machine in response to a VM creation request; (ii) *VM Relocation*: the migration of VMs from a host when the combined requirements of co-located VMs exceed the resources available on the host *(stress situation)*; and (iii) *VM Consolidation*: the migration of VMs from an under-utilized host, so that the machine may be powered off to reduce costs. These operations make use of metrics characterizing the utilization of resources and the behaviour of applications. VM Relocation and VM Consolidation are triggered on regular time intervals. Decisions on when to invoke these operations are based on conditions on one or more metrics, e.g., when a certain threshold is exceeded. The specific conditions, metrics and threshold values vary and can be represented as a policy.

We consider a *dynamic management strategy* to consist of a set of policies, such that there is a policy that governs each of the defined management operations (i.e., a VM Placement policy, a VM Relocation policy, and a VM Consolidation policy). In our current research, we focus on two of the most commonly studied goals in the area: (i) minimizing power consumption; and (ii) minimizing Service Level Agreement (SLA) violations. We consider an SLA to be a set of non-functional requirements, such as a promised condition on a

* All three authors contributed equally to this work.

978-3-901882-50-0 

metric (e.g., response time below a given threshold). These goals are often in conflict. Minimizing power consumption is usually approached by reducing the number of hosts in use (and thus powered on). This is achieved by placing as many VMs on a single host as possible. However, sudden increases in workload are more likely to result in a shortage of resources and therefore lead to a high number of SLA violations. Conversely, minimizing SLA violations typically requires VMs to be spread across more hosts, often each having a significant amount of unused resources available to handle spikes in demand. This, however, results in higher power consumption. Designing a management strategy to achieve both of these goals is therefore difficult, as improving performance towards one goal typically results in degradation of performance towards the other. Design of management strategies often focuses on achieving a single goal, or on prioritizing goals such that a single goal is considered the primary goal and others are considered secondary, e.g., [6], [1], [7], [8].

Within a dynamic environment there may be times when one management strategy is more appropriate than another. We propose an approach to dynamically switch between management strategies where each has a primary focus on a single goal; in this case, one strategy to minimize SLA violations and another to minimize power consumption. By doing so, we aim to achieve better performance in attaining both goals. The main contributions of this paper are three novel methods of dynamically switching between single-goal management strategies, and a method of comparing the performance of strategies that aim to achieve more than one goal.

The remainder of this paper is organized as follows: Section II reviews related work in this area, Sections III and IV describe the management strategies and strategy switching approaches we explored, respectively. Section V presents experiments and evaluation, and finally, Section VI concludes and discusses future work.

## II. Related Work

Several works in the area have approached the problem of placing VMs in a data centre statically. These works assume VMs' service demand to be static, or to be variable but exhibit a periodic cycle, and perform a one-time mapping (i.e., a static allocation) of a set of VMs into a set of empty hosts by different methods. Cardosa et al. [9] relied on a Best Fit heuristic that leveraged the CPU allocation features `min`, `max`, and `shares` present in modern hypervisors. Speitkamp and Bichler [2] proposed a Linear Programming relaxation-based heuristic, combined with data analysis to characterize variations in workload traces. Stillwell et al. [3] proposed and evaluated an extensive set of algorithms, finally settling for a vector bin packing algorithm that achieved close to optimal solutions to the problem.

Other static approaches have considered variable and aperiodic service demand, but have addressed the problem by statically allocating VMs to hosts and periodically re-calculating the complete mapping. Such is the case of Bobroff et al. [1], who used a First Fit Decreasing heuristic for the task. In most cases, these static approaches aimed to minimize the average number of active hosts, or to increase host or data centre utilization, and in that way minimize power consumption.

Research in dynamic management of virtualized data centres focuses on one of the three management operations (i.e., VM Placement, VM Relocation or VM Consolidation), or a combination of them. Most solutions focus on pursuing a single goal, or on pursuing multiple goals, but prioritizing one goal over all others. Wood et al. [7] addressed the VM Relocation problem using a First Fit Decreasing heuristic that spread load across hosts and thus aimed to reduce SLA violations. Keller et al. [10] studied variants of a First Fit heuristic to address the VM Relocation problem, showing that the order in which VMs and hosts are considered for migration impacts data centre performance metrics, such as power consumption and SLA violations.

Several works address VM Relocation and VM Consolidation together. In most cases, they focus primarily on minimizing power consumption and then on minimizing SLA violations. Khanna et al. [6] implemented a First Fit heuristic that migrated the least loaded VMs (in terms of CPU and memory usage) into the highest loaded hosts. Verma et al. [8] relied on a First Fit Decreasing heuristic that placed VMs in the most power efficient servers first. Beloglazov and Buyya [11] proposed a Best Fit Decreasing heuristic that selected for migration the VMs with the smallest memory footprint and placed them in the host that provided the least increase in power consumption. Much of the existing work on dynamic management uses some form of First Fit heuristics, though occasionally alternatives are proposed, such as fuzzy logic-based controllers, e.g., [12].

The main difference between this work and the cited ones is that the solution proposed here focuses on pursuing at the same time two different (and opposing) goals – namely minimizing SLA violations and minimizing power consumption – without prioritizing one goal over the other.

## III. Management Strategies

This section presents three management strategies that are representative of those found in the literature. The strategies presented assume frequent monitoring. Calculations are performed on monitored values over a sliding window of time, referred to as the *monitoring window*.

### A. Terminology

This section presents the terms and metrics used in the description of management strategies.

*SLA Violation:* An SLA violation occurs when resources required by a VM are not available to it, as this situation leads to a degradation in performance. The percentage of required CPU not available in the SLA violation is denoted by $s$.

*Data Centre Utilization:* The overall utilization of the data centre is calculated as the percentage of total CPU capacity in the data centre that is currently in use.

*CPU Shares:* We quantify the capacity of a CPU as *CPU shares*, where each CPU core has a specific number of shares which represents its computing power. In our work, the number of shares assigned to each core is based on its frequency, with 1GHz = 1000 shares.

*Power Efficiency:* For a host, $h$, the power efficiency, $p_h$, is the amount of processing being performed per watt of power. This is measured in CPU-shares-per-watt (cpu/watt). The calculation of the power efficiency of a single host is presented in Equation 1:

$$p_h = \frac{cpuInUse_h}{powerConsumption_h} \quad (1)$$

where $cpuInUse_h$ is the number of CPU shares currently in use across all cores in the host, and $powerConsumption_h$ is the current power consumption in watts of the host. As an active host machine consumes a significant amount of power even when under little or no CPU load (i.e. very low power efficiency) increased host utilization corresponds with increased power efficiency for that host. This metric is used to calculate the power efficiency for the entire data centre, $p_{dc}$, calculated as

$$p_{dc} = \frac{\sum_{h \in hosts} cpuInUse_h}{\sum_{h \in hosts} powerConsumption_h} \quad (2)$$

such that $hosts$ is the collection of all hosts in the data centre.

*Maximum Power Efficiency:* This metric represents the best power efficiency a host can achieve, calculated as the power efficiency of the host at maximum CPU utilization.

*Optimal Power Efficiency:* Optimal Power Efficiency, $p_{dc}^{opt}$, represents the best possible power efficiency achievable at the data centre level, given the current workload and set of host machines available. The best power efficiency would be achieved by placing VMs in such a way that each host is 100% utilized, with the most power efficient hosts being filled first. We first calculate the total CPU-in-use across the data centre. We order the available hosts by maximum power efficiency, and allocate the CPU-in-use to hosts such that each host is allocated 100% of its CPU capacity. We calculate $p_{dc}^{opt}$ to be the power efficiency of the data centre given this allocation.

### B. Host Classification

Each time a management operation takes place, hosts are classified into categories based on their power state: `on`, `suspended` or `off`. Powered on hosts are further classified as *stressed*, *partially-utilized* or *under-utilized*, based on their CPU utilization level. Hosts may transition between these states based either on changes in workload of the hosted VMs, or migrations performed by the management operations. Two threshold values are used for categorization: $stress_{CPU}$ and $minUsage_{CPU}$. Classification is based on the hosts average CPU utilization over the last monitoring window (measurements collected every 2 minutes over a sliding window of size 5). Categories are defined as follows:

- **Stressed**: hosts with average CPU utilization in the range $(stress_{CPU}, 1]$;
- **Partially-utilized**: hosts with average CPU utilization in the range $(minUsage_{CPU}, stress_{CPU}]$;
- **Under-utilized**: hosts with average CPU utilization in the range $[0, minUsage_{CPU}]$;
- **Empty**: hosts that do not currently have any VM assigned to them. Hosts in `suspended` or `off` power state are included in this category.

It should be noted that VM Relocation policies make the determination of whether a host is stressed in a slightly different way based on how the most recent measurements of host utilization are considered. This *stress check* may mark as partially-utilized a host that is considered stressed under this classification, or vice versa. More information on this issue is presented in Section III-C2.

### C. Power and SLA Strategies

Power and SLA are *single-goal* strategies, which means that all management decisions are geared towards achieving a single, primary goal. Single-goal strategies may pursue secondary goals, but always give them lower priority than the primary goal.

In the next subsections, we will describe the VM Placement, VM Relocation and VM Consolidation policies that comprise these two strategies. Much of the existing work on dynamic management uses some form of First Fit heuristics. The work described in Stillwell et al. [3] (for static workloads) and Keller et al. (for dynamic workloads) [10] studied variants of First Fit heuristics to find that they work best in practice. The Power and SLA strategies are based on such heuristics and are representative of other work on dynamic resource management.

The strategies use different values for the $stress_{CPU}$ threshold: the Power strategy uses 95% and the SLA strategy used 85%. The lower threshold for the SLA strategy allows for additional resources to be available for workload variations. Both strategies use the $minUsage_{CPU}$ threshold at 60%.

*1)* **VM Placement:** This management operation runs each time a new VM creation request is received, and selects a host in which to instantiate the VM. The VM Placement policy for the Power strategy (see Algorithm 1) first classifies $hosts$ in their respective categories (line 3): stressed ($z$), partially-utilized ($p$), under-utilized ($u$) and empty ($e$). The policy then sorts each host category (lines 4-5): $p$ and $u$ are sorted in decreasing order first by maximum power efficiency and then by CPU utilization, and $e$ is sorted in decreasing order first by maximum power efficiency and then by power state. This sorting method ensures that the placement focuses on power efficiency over any other considerations. The policy then builds a list of $target$ hosts by concatenating $p'$, $u'$ and $e'$ (line 6). Finally, following a First Fit approach, the policy assigns the VM to the first host in $target$ with enough capacity to host the VM (lines 7-12). The method `hasCapacity(VM)` checks whether the host can meet the resource requirements indicated in the VM creation request (line 8) without the host becoming stressed.

The VM Placement policy for the SLA strategy differs from the Power strategy's policy in the way $p$ and $u$ are sorted: $p$ is sorted in increasing order first by CPU utilization and then by maximum power efficiency and $u$ is sorted in decreasing order first by CPU utilization and then by maximum power

```
1: Input: VM
2: Output: –
3: z, p, u, e = classifyHosts(hosts)
4: p', u' = sortPowerEffThenUtil(p, u)
5: e' = sortPowerEffThenState(e)
6: target = concatenate(p', u', e')
7: for host in target do
8:    if host.hasCapacity(VM) then
9:       host.deploy(VM)
10:      break
11:   end if
12: end for
```

**Algorithm 1**: Power strategy's VM Placement policy.

efficiency. This sorting method ensures that the placement focuses on spreading load across the hosts, leaving spare resources to handle spikes in resource demand, over any other considerations.

*2)* **VM Relocation:** This management operation runs frequently over short intervals of time, so as to detect stress situations as soon as possible. For both strategies, the interval is set to 10 minutes. This operation determines which hosts are experiencing a stress situation and attempts to resolve the situations by migrating one VM from each stressed host to a non-stressed host. The VM Relocation policy for the Power strategy (see Algorithm 2) first classifies $hosts$ in their respective categories (line 1), performing a *stress check* on all hosts to determine whether or not they are stressed. The policy determines that a host is stressed if its CPU utilization has remained above the $stress_{CPU}$ threshold all of the time over the last CPU load monitoring window. The resulting host categories are: stressed ($z$), partially-utilized ($p$), under-utilized ($u$) and empty ($e$). The policy then sorts each host category (line 2-4): $z$ is sorted in decreasing order by CPU utilization, $p$ and $u$ are sorted in decreasing order first by maximum power efficiency and then by CPU utilization, and $e$ is sorted in decreasing order first by maximum power efficiency and then by power state. The policy then builds a list of $target$ hosts by concatenating $p'$, $u'$ and $e'$ (line 6). Following a First Fit heuristic, the policy selects one VM from each host $h$ in $source$ and a corresponding $host$ in $target$ to which to migrate the VM (lines 7-22). For each host $h$ in $source$, the policy filters out the VMs with less CPU load than the CPU load by which $h$ is stressed and sorts the remaining VMs in increasing order by CPU load (line 8). If the list of remaining VMs is empty, all VMs are considered and sorted in decreasing order by CPU load. The method `migrate(h,VM,host)` initiates a migration (line 13).

The VM Relocation policy for the SLA strategy differs from the Power strategy's policy in the way $p$ and $u$ are sorted: $p$ is sorted in increasing order first by CPU utilization and then by maximum power efficiency and $u$ is sorted in decreasing order first by CPU utilization and then by maximum power efficiency. In addition, the policy performs a different stress check: a host is stressed if its last two monitored CPU load values are above the $stress_{CPU}$ threshold or its average CPU utilization over the last CPU load monitoring window exceeds $stress_{CPU}$.

```
1: z, p, u, e = classifyHosts(hosts)
2: z' = sortUtil(z)
3: p', u' = sortPowerEffThenUtil(p, u)
4: e' = sortPowerEffThenState(e)
5: source = z'
6: target = concatenate(p', u', e')
7: for h in source do
8:    vms = filterAndSort(h.vms)
9:    success = FALSE
10:   for VM in vms do
11:      for host in target do
12:         if host.hasCapacity(VM) then
13:            migrate(h, VM, host)
14:            success = TRUE
15:            break
16:         end if
17:      end for
18:      if success then
19:         break
20:      end if
21:   end for
22: end for
```

**Algorithm 2**: Power strategy's VM Relocation policy.

*3)* **VM Consolidation:** This management operation runs less frequently than VM Relocation, given that its purpose is to consolidate the load that VM Placement and VM Relocation have spread across the data centre. The interval is set to 1 hour for the Power strategy and to 4 hours for SLA strategy, aiming in the former case to achieve and sustain a higher degree of consolidation. This operation consolidates load in the data centre by migrating VMs away from under-utilized hosts (and suspending or powering them off) and into partially-utilized hosts. The VM Consolidation policy for the Power strategy (see Algorithm 3) first classifies $hosts$ in their respective categories (line 1): stressed ($z$), partially-utilized ($p$), under-utilized ($u$), and empty ($e$), and powers off $e$ (line 2). The policy then sorts $p$ and $u$ in decreasing order first by maximum power efficiency and then by CPU utilization (line 3) and builds a list of $target$ hosts by concatenating $p'$ and $u'$ (line 4). Afterwards, the policy sorts $u$ again, but this time in increasing order first by power efficiency and then by CPU utilization, and uses that list as $source$ (line 5). Following a First Fit heuristic, the policy attempts to vacate every host $h$ in $source$ by migrating their VMs into $hosts$ in $target$ (lines 6-16). For each host $h$ in $source$, the policy sorts its VMs in decreasing order first by overall resource capacity (memory, number of CPU cores, core capacity) and then by CPU load (line 7). Given that $source$ and $target$ are not disjunct, measures have to be taken to avoid using a host both as source and target for migrations.

The VM Consolidation policy for the SLA strategy differs from the Power strategy's policy in the way $p$ and $u$ are sorted: first, $p$ is sorted in increasing order first by CPU utilization and then by maximum power efficiency and $u$ is sorted in decreasing order first by CPU utilization and then by maximum power efficiency, and then, $u$ is sorted in increasing order by CPU utilization.

```
1:  z, p, u, e = classifyHosts(hosts)
2:  powerOff(e)
3:  p', u' = sortPowerEffThenUtil(p, u)
4:  target = concatenate(p', u')
5:  source = sortPowerEffThenUtil(u)
6:  for h in source do
7:      vms = sort(h.vms)
8:      for VM in vms do
9:          for host in target do
10:             if host.hasCapacity(VM) then
11:                 migrate(h, VM, host)
12:                 break
13:             end if
14:         end for
15:     end for
16: end for
```

**Algorithm 3**: Power strategy's VM Consolidation policy.

### D. Hybrid Strategy

We designed a *dual-goal* strategy as a combination of the Power and SLA strategies; the Hybrid strategy consists of the VM Placement and VM Relocation policies of the SLA strategy and the VM Consolidation policy of the Power strategy. Furthermore, the stress check performed by the VM Relocation policy represents a compromise between the checks of SLA and Power: it determines that a host is stressed only if its average CPU utilization over the last monitoring window exceeds the $stress_{CPU}$ threshold. The thresholds $stress_{CPU}$ and $minUsage_{CPU}$ were set to 90% and 60% respectively.

## IV. Dynamic Strategy Switching

Dynamic Strategy Switching (DSS) refers to changing between strategies at run-time in response to changing data centre state. DSS periodically performs an evaluation of data centre metrics monitored between executions to determine if the strategy currently in use (the *active strategy*) should be changed. In this section, we present three different DSS meta-strategies.

### A. SP-DSS

The SLA-Power Thresholds (SP-DSS) meta-strategy uses the SLA violation ($s$) and power efficiency ratio ($per$) metrics to evaluate whether the active strategy should be switched. The power efficiency ratio is calculated as the ratio of optimal power efficiency ($p_{dc}^{opt}$) to current power efficiency ($p_{dc}$) over the last hour. A strategy switch is triggered when the metric related to the goal of the active strategy (i.e., $s$ for the SLA strategy, $per$ for the Power strategy) is below a normal (i.e. acceptable) threshold ($s_{norm}$ or $per_{norm}$), while the metric related to the inactive strategy exceeds a high threshold ($s_{high}$ or $per_{high}$). See Algorithm 4 for details. Switching strategies in this manner allows the data centre to respond to a situation in which performance in one metric has deteriorated, by activating the strategy that focuses on optimizing it.

### B. Goal-DSS

We define two goals, $s = 0\%$ and $p_{dc} = p_{dc}^{opt}$, to evaluate performance with respect to the $s$ and $p$ metrics. By calculating

```
1: if activeStrategy = Power_Strategy then
2:     if per < per_norm & s > s_high then
3:         Switch to SLA_Strategy
4:     end if
5: else if activeStrategy = SLA_Strategy then
6:     if s < s_norm & per > per_high then
7:         Switch to Power_Strategy
8:     end if
9: end if
```

**Algorithm 4**: SP-DSS Switching Conditions.

the distance to these goals, it is possible to determine towards which goal the system is performing worst and thus switch to the strategy that would improve achievement of that goal. The Distance to Goals (Goal-DSS) meta-strategy is based on this principle. Evaluating whether or not the active strategy should be switched requires the calculation of two metrics that represent the distance to those goals. These are presented in Equations 3 and 4.

$$slaDist = \frac{s}{s_{worst}} \tag{3}$$

where $s$ is the current SLA violation percentage and $s_{worst}$ is an operator-defined parameter that indicates the worst acceptable SLA violation percentage, and

$$powerDist = 1 - \frac{p_{dc} - p_{worst}}{p_{dc}^{opt} - p_{worst}} \tag{4}$$

$$p_{worst} = p_{dc}^{opt} * p_C \tag{5}$$

where $p_{worst}$ is the worst acceptable power efficiency, and $p_C$ is an operator-defined parameter that indicates how large a deviation from the optimal power efficiency is acceptable.

Calculating the distances in this manner is necessary in order to equate values of $s$ with values of $p$, based on the parameters $s_{worst}$ and $p_C$. These two values are considered equivalent in terms of distance to their respective goals. At each iteration of the strategy switching mechanism, the strategy for which the corresponding distance is greater is selected to become active. The switching algorithm is similar to Algorithm 4, with the condition in line 2 being $slaDist > powerDist$ and the condition in line 6 being $powerDist > slaDist$.

### C. Util-DSS

Through experimentation, two key situations in which one strategy had an advantage over the other became apparent. When overall data centre utilization is growing, increasing the stress on host machines, the SLA strategy is more effective as it places greater emphasis on preventing SLA violations. Conversely, when utilization is decreasing or stable, thus increasing the likelihood of hosts becoming underutilized, the Power strategy is more effective as it can quickly make changes to conserve power. Data centre utilization is defined as the percentage of CPU shares in use across the entire data centre.

The Data Centre Utilization Trends (Util-DSS) meta-strategy is designed to exploit this pattern. It uses the rate

of change of overall data centre utilization, $m$, to determine appropriate times to switch strategies. Measurements of the overall data centre utilization are taken at regular intervals. Linear regression over the last $n$ data centre utilization measurements provides the rate of change, $m$, over a window of time. The value $m_{SLA}$ defines a threshold for $m$ over which a switch is made to the SLA strategy. Similarly, the value $m_{Power}$ defines a threshold for $m$ under which the Power strategy is set to be active. The switching algorithm is similar to Algorithm 4, with the condition in line 2 being $m > m_{sla}$ and the condition in line 6 being $m < m_{power}$.

## V. Experiments

This section presents our experimental approach and results.

### A. Strategy Evaluation and Comparison

In order to evaluate the effectiveness of the strategies, two metrics are used: power efficiency ($p$) and SLA violation ($s$). Comparing strategies based only on the use of these two metrics is problematic. If one strategy were to perform well with respect to SLA violations at the expense of power, and another performed well with respect to power at the expense of SLA violations, it is difficult to conclude which strategy is preferable. The decision depends in part upon the relative change in each area as well as the importance placed on each metric by the data centre operators based on their business objectives, the relative costs of power and SLA violations and the potential for lost revenue due to poor application behaviour.

In order to determine whether DSS can offer improved results over a single strategy, we propose a method of evaluating the performance of a strategy based on experimental results. We use the SLA and Power strategies as benchmarks, with their SLA violation and power efficiency results serving as baseline measurements with which to evaluate other strategies. The SLA strategy provides the bounds for the best SLA violation value ($s_{best} = s_{SLA}$) and the worst power efficiency ($p_{worst} = p_{SLA}$), while the Power strategy provides the worst SLA violation ($s_{worst} = s_{Power}$) and best power efficiency ($p_{best} = p_{Power}$). Values from a candidate strategy, $i$, are then normalized using these bounds to produce the normalized vector, $v_i$, represented by $[s_{norm}, p_{norm}]$. The values $s_{norm}$ and $p_{norm}$ are defined in Equation 6.

$$\begin{aligned} s_{norm} &= \frac{(s_i - s_{best})}{(s_{worst} - s_{best})} \\ p_{norm} &= \frac{(p_{best} - p_i)}{(p_{best} - p_{worst})} \\ v_i &= (s_{norm}, p_{norm}) \end{aligned} \tag{6}$$

where $p_{norm}$ is the normalized power efficiency and $s_{norm}$ is the normalized SLA violation.

Note that $p_{best} > p_{worst}$, but $s_{best} < s_{worst}$, so the normalization equations differ to reflect this. Once we have the normalized vector, $v_i$, we calculate its $L^2$-norm, $|v_i|$, and use this as an overall score ($score_i$) for the candidate strategy.

$$score_i = |v_i| = \sqrt{s_{norm}^2 + p_{norm}^2} \tag{7}$$

where a smaller score is considered better, as it represents a smaller distance to the *best* bounds of each metric (defined by $s_{best}$ and $p_{best}$). The SLA and Power strategies always achieve a score of 1 by definition, as they achieve the *best* score in one metric and the *worst* in the other. Scores less than 1 indicate that overall performance of the candidate strategy has improved relative to the baseline strategies.

Note that this score is only valid for a single experiment in which all factors except for the active management strategy remain constant. In our work, we vary the workload pattern experienced by the data centre. As such, the baselines and score must be calculated separately for each workload pattern. The average final score across all experiments can then be used to evaluate the strategy. We use this method to evaluate and compare competing management strategies.

### B. Experimental Setup

We conduct our experimentation by simulation using DCSim [13]. Our simulated data centre consists of 200 host machines, of which there are an equal number of two types: *small* and *large*. The *small* host is modelled after the HP ProLiant DL380G5, with 2 dual-core 3GHz CPUs and 8 GB of memory. The *large* host is modelled after the HP ProLiant DL160G5, with 2 quad-core 2.5GHz CPUs and 16GB of memory. Cores in the *large* host have 2500 CPU shares, and cores in the *small* host have 3000 CPU shares. The power consumption of both hosts is calculated using results from the SPECPower benchmark [14]. The maximum power efficiency of the *large* host (85.84 cpu/watt) is roughly double that of the *small* host (46.51 cpu/watt).

Three VM sizes are created: *small* requires 1 virtual core with at least 1500 CPU shares and 512MB of memory, *medium* requires 1 virtual core with at least 2500 CPU shares and 512MB of memory, and *large* requires 2 virtual cores with at least 2500 CPU shares each and 1GB of memory.

Hosts are modelled to use a work-conserving CPU scheduler, as available in major virtualization technologies. That is, any CPU shares not used by a VM can be used by another. No maximum cap on CPU is set for VMs. In the case of CPU contention, VMs are assigned shares in a round-robin fashion until all shares have been allocated. No dynamic voltage and frequency scaling (DVFS) is considered. Memory is statically allocated and not overcommitted.

During a VM migration, an SLA violation of 10% of CPU utilization is added to migrating VMs, and an additional CPU overhead of 10% of the migrating VMs CPU utilization is added to both the source and target host [11].

Measurements of metrics used by management policies, such as host CPU utilization and SLA violation, are drawn from each host every 2 minutes and evaluated by the policy over a sliding window of 5 measurements.

### C. Workload

A data centre experiences a highly dynamic workload, driven by VM arrivals and departures, as well as dynamic workloads and resource requirements of VMs. We generate random *workload patterns* to evaluate our strategies, where a workload pattern consists of a set of VMs with specific

| Strategy | Param. | Value |
|---|---|---|
| SP-DSS | $per_{norm}$ | 0.004 |
| SP-DSS | $per_{high}$ | 0.006 |
| SP-DSS | $s_{norm}$ | 1.15 |
| SP-DSS | $s_{high}$ | 1.3 |
| Goal-DSS | $s_{worst}$ | 0.01 |
| Goal-DSS | $p_C$ | 0.83 |
| Util-DSS | $m_{SLA}$ | 0.00255 |
| Util-DSS | $m_{power}$ | 0.00255 |

TABLE I. DSS TUNING PARAMETERS

start and stop times, each with dynamic trace-driven resource requirements. Each VM is driven by one of 5 individual traces: the *ClarkNet*, *EPA*, and *SDSC* traces [15], and two different job types from the *Google Cluster Data* trace [16]. The normalized rate of incoming requests, in 100 second intervals, is calculated for each trace. The request rates are used to define the current workload of each VM, with the CPU resource requirements of the VM calculated as a linear function of the current rate. Each VM starts its trace at a randomly selected offset time.

The number of VMs within the data centre is also varied dynamically to simulate the arrival and departure of VMs. A base of 600 VMs is created within the first 40 hours and remain running throughout the entire experiment, to maintain a reasonable minimum level of load. After 2 simulated days, new VMs begin to arrive at a changing rate, and terminate after about 1 day. The arrival rates are generated such that on a fixed interval of once per day, the total number of VMs in the data centre is equal to a randomly generated number uniformly distributed between 600 and 1600. The maximum number of VMs, 1600, was chosen because beyond that point, the SLA strategy is forced to deny admission of some incoming VMs due to insufficient available resources. This continues for 10 simulated days at which point the experiment terminates. Data from the first 2 days of simulation are discarded to allow the simulation to stabilize before recording results.

### D. Strategy Switching Tuning Parameters

Each DSS strategy has some tuning parameters that must be configured to provide the best possible results, as described in Section IV. The first of these is the frequency with which the strategy switching algorithm is run. We evaluated the meta-strategies over multiple frequency values and found 1 hour to be an appropriate frequency for evaluating a strategy switch. Each DSS strategy looks at a set of data centre metrics sampled by a monitor over a certain window size: SP-DSS and Goal-DSS sample every 5 minutes and use a window size of 6 samples; Util-DSS samples every 20 minutes and uses a window size of 6. Util-DSS uses a longer monitoring frequency and window size in order to ignore minor fluctuations in data centre utilization and focus on longer term trends. This helps identify periods of real change in overall utilization, and avoid thrashing between strategies. For the remaining DSS tuning parameters, each combination of values was evaluated over a set of 5 randomly generated workload patterns, and the values that resulted in the best $score$ were chosen. Table I contains the values of the best performing of all parameters defined in Section IV.

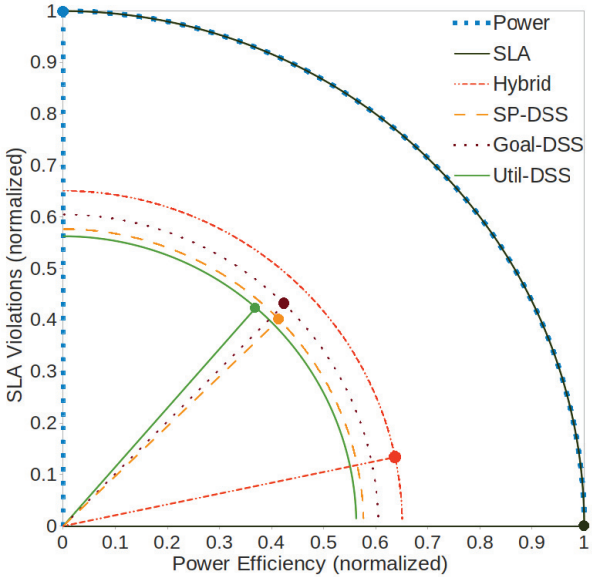


Fig. 1. Strategy Scores

### E. Results

The results of the experiments are presented in Table II. Each management strategy was evaluated with the same set of 100 randomly generated workload patterns. Each experiment was repeated only once per workload pattern, as the simulation is deterministic. Results were averaged across all workload patterns and the standard deviation is shown in square brackets. We report the following metrics: Average Active Host Utilization is the average CPU utilization of powered on hosts; # of Migrations is the number of VM migrations triggered by the management strategies; Power Consumed is the total power consumed by all hosts in kWh; Power Efficiency is $p_{dc}$ over the entire simulation; SLA Violation is $s$ over the entire simulation; and # of Strategy Switches is the number of times that the active strategy was changed. We also report the normalized SLA and power values for each strategy, as well as the $score$. Figure 1 presents a graphical representation of the scores. The benchmark strategies (SLA and Power) both achieve a score of 1, by the definition of the score in Section V-A. The angle of the line from the origin to each point gives an indication of how fairly the strategy behaved towards each goal, with a 45 degree angle representing a perfect balance between SLA and power.

Analysis of Variance was performed on the score results, as well as paired t-tests for each pair of management strategies. The resulting scores for each management strategy were found to be significantly different from each other.

### F. Discussion

All three DSS meta-strategies, as well as Hybrid, achieved better scores than the single-goal SLA and Power strategies. Util-DSS achieved the lowest score, followed by SP-DSS, then Goal-DSS, and finally Hybrid. The meta-strategies improved the score by about 40% when compared to Power and SLA, and by about 7-12% when compared to Hybrid. Util-DSS and SP-DSS each slightly favoured one of the goals, with

| | SLA | Power | Hybrid | SP-DSS | Goal-DSS | Util-DSS |
|---|---|---|---|---|---|---|
| Avg. Active Host Util. | 75% [0.4] | 88% [0.4] | 81% [0.4] | 80% [1] | 81% [1] | 82% [1] |
| # of Migrations | 15818 [2292] | 24378 [3311] | 14643 [1930] | 18608 [3317] | 19448 [2754] | 19580 [3047] |
| Power Consumed (kWh) | 5488 [703] | 4384 [519] | 5049 [679] | 4840 [561] | 4821 [612] | 4778 [583] |
| Power Efficiency | 60.6 [2.4] | 75.2 [2.0] | 65.9 [2.7] | 69.7 [1.8] | 69.0 [2.4] | 69.8 [2.3] |
| SLA Violation | 0.033% [0.01] | 0.474% [0.05] | 0.092% [0.01] | 0.198% [0.04] | 0.222% [0.02] | 0.220% [0.05] |
| # of Strat. Switches | N/A | N/A | N/A | 20 [5] | 56 [4] | 30 [10] |
| $s_{norm}$ | 0.0 | 1.0 | 0.135 [0.01] | 0.360 [0.09] | 0.430 [0.04] | 0.425 [0.09] |
| $p_{norm}$ | 1.0 | 0.0 | 0.636 [0.06] | 0.452 [0.08] | 0.425 [0.05] | 0.373 [0.08] |
| Score | 1.0 | 1.0 | 0.651 [0.05] | 0.588 [0.05] | 0.607 [0.03] | 0.576 [0.041] |

TABLE II. RESULTS PER STRATEGY

Util-DSS favouring power and SP-DSS favouring SLA. Goal-DSS behaved fairly towards both goals. Hybrid, on the other hand, was considerably more skewed towards SLA than power, potentially limiting its usefulness in a practical application. The improved overall performance, as well as the balanced treatment of each goal, may therefore favour the selection of DSS over Hybrid. Among the meta-strategies, Util-DSS showed to be the most effective, though Goal-DSS was the most balanced.

All meta-strategies triggered 31 to 33% more migrations than the Hybrid strategy. While migration overhead was taken into consideration and reflected in the SLA violation and host utilization metrics, further work investigating the effect of migrations on networking should be conducted to determine if this migration count is acceptable. The increase in migration count from Hybrid to DSS is likely a side-effect of switching between strategies with different $stress$ thresholds. The Power strategy efficiently pushes the utilization of a large number of hosts to a high value, just below its $stress$ threshold. A switch to the SLA strategy at this point causes a large number of hosts to be considered stressed, as its $stress$ threshold is below the current utilization achieved by the Power strategy. Thus, a spike in migrations is triggered. This also causes a spike in SLA violations due to migration overhead. It may be possible to introduce a mechanism to mitigate this effect and thus lower the DSS meta- strategy migration count. Such a mechanism may also result in an overall better $score$ for the meta-strategies.

SP-DSS switched strategies the least number of times, followed by Util-DSS and Goal-DSS. This may be an indication that Util-DSS and Goal-DSS performed some strategy switches that did not contribute to improving performance towards the intended goals (possibly exhibiting a thrashing behaviour), and should be investigated.

## VI. CONCLUSIONS AND FUTURE WORK

The development of data centre management strategies that can simultaneously pursue opposing goals, such as maximizing power efficiency and minimizing SLA violations, is a difficult task. In this work, we proposed dynamically switching between two strategies, each designed to achieve a single goal, to better adapt to changing data centre conditions. We developed three *meta-strategies* to perform dynamic strategy switching, and evaluated them through simulation. The meta-strategies improve overall performance by about 40% when compared to either of the single-goal strategies, and by 7-12% when compared to a hybrid strategy designed to pursue both goals simultaneously.

There are several directions for future work. Regarding DSS, the meta-strategies' behaviour when switching between strategies could be improved, so as to avoid spikes in migrations. DSS could also be applied separately to subsets of hosts, such as individual racks or clusters. Finally, threshold values and tuning parameters could be learned rather than fixed.

Networking overhead was not considered in this work. In the future, we intend to develop networking metrics to be used in our evaluations of management policies and strategies. Another topic of interest is the inclusion of constraints and affinity rules to help in determining the placement of VMs on hosts, as discussed by Gulati et al. [17].

Currently, our work relies only on CPU measurements to determine the level of load of a host or VM, with other resources used only as a constraint on placement. In the future, load calculation should take into consideration memory and bandwidth, in addition to CPU (e.g.,[18]).

Future work could incorporate forecasting of VMs' resource demand, such as done by Bobroff et al. [1]. This would have an effect in the detection of stress situations, and in VM and host selection for migration.

Finally, this work essentially assumes a central manager for all decision making. Other architectural models could be explored, such as that presented by Zhu et al. [19].

## ACKNOWLEDGEMENTS

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

## REFERENCES

[1] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *IM Proceedings, 2007 IEEE/IFIP Int. Symp. on*, 2007, pp. 119–128.

[2] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE TSC*, vol. 3, no. 4, pp. 266 –278, 2010.

[3] 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.

[4] A. Kochut and K. Beaty, "On Strategies for Dynamic Resource Management in Virtualized Server Environments," in *MASCOTS Proceedings, 2007 15th Int. Symp. on*, 2007, pp. 193–200.

[5] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing," Microsoft Research, Tech. Rep., 2011.

[6] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application performance management in virtualized server environments," in *NOMS Proceedings, 2006 IEEE/IFIP*, 2006.

[7] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *NSDI Proceedings, 4th Symp. on*, Cambridge, MA, USA, Apr. 2007, pp. 229–242.

[8] A. Verma, P. Ahuja, and A. Neogi, "pmapper: power and migration cost aware application placement in virtualized systems," in *Proceedings of the 9th ACM/IFIP/USENIX Int. Conf. on Middleware*, 2008.

[9] M. Cardosa, M. R. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *IM Proceedings, 2009 IEEE/IFIP Int. Symp. on*, 2009.

[10] G. Keller, M. Tighe, H. Lutfiyya, and M. Bauer, "An analysis of first fit heuristics for the virtual machine relocation problem," in *SVM Proceedings, 6th Int. DMTF Academic Alliance Workshop on*, Oct. 2012.

[11] 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 Computat.: Pract. Exper.*, pp. 1–24, 2011.

[12] 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," in *38th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN)*, June 2008.

[13] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, "DCSim: A data centre simulation tool for evaluating dynamic virtualized resource management," in *SVM Proceedings, 6th Int. DMTF Academic Alliance Workshop on*, Oct. 2012.

[14] (2012, Aug.) Specpower_ssj2008 benchmark. Standard Performance Evaluation Corporation. [Online]. Available: http://www.spec.org/power_ssj2008/

[15] (2012, Aug.) The internet traffic archive. [Online]. Available: http://ita.ee.lbl.gov/

[16] (2012, Aug.) Google cluster data. Google Inc. [Online]. Available: http://code.google.com/p/googleclusterdata/

[17] A. Gulati, G. Shanmuganathan, A. Holler, C. Waldspurger, M. Ji, and X. Zhu, "Vmware distributed resource management: design, implementation, and lessons learned," *VMware Technical Journal*, vol. 1, no. 1, 2012.

[18] E. Arzuaga and D. R. Kaeli, "Quantifying load imbalance on virtualized enterprise servers," in *Perf. Eng. Proceedings, 1st WOSP/SIPEW Int. Conf. on*, 2010.

[19] X. Zhu, D. Young, B. J. Watson, Z. Wang, J. Rolia, S. Singhal, B. McKee, C. Hyser, D. Gmach, R. Gardner, T. Christian, and L. Cherkasova, "1000 islands: Integrated capacity and workload management for the next generation data center," in *Proceedings of the 2008 International Conference on Autonomic Computing (ICAC'08)*, Chicago, IL, USA, Jun. 2008, pp. 172–181.