3-27-2013 12:00 AM

UTIL-DSS: Utilization-Based Dynamic Strategy Switching for Improvement in Data Centre Operation

Graham Foster, *The University of Western Ontario*

Supervisor: Hanan Lutfiyya, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

UTIL-DSS: UTILIZATION-BASED DYNAMIC STRATEGY SWITCHING FOR IMPROVEMENT IN DATA CENTRE OPERATION

(Thesis format: Monograph)

by

Graham Foster

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

Applications are shifting into large scale, virtualized data centres that provide resources on a pay-per-usage basis. With power consumption representing a major operational cost, data centres must prioritize efficiency while still providing enough resources to meet application requirements. To meet variable application demands, a dynamic approach to virtual machine (VM) management is required. This requires: (i) placing newly arrived VMs, (ii) migrating VMs from highly utilized machines to avoid performance degradation, and (iii) migrating VMs from underutilized machines so that they may be deactivated to save power. Here, a management strategy is considered to be a policy-set that guides these three operations. To achieve the conflicting goals of performance and efficiency, I propose and evaluate a system of dynamically switching between two management strategies, each with a single goal, based on trends in data centre workload. Experimentation over a simulated data centre demonstrates the superiority of this approach over single-strategy techniques.

Keywords

Virtualization, Dynamic Resource Management, Data Centre Operation, Data Centre Simulation, Power Efficiency, Resource Management Evaluation.

Co-Authorship Statement

This work is based on a previous work by this author and others. The previous work [26] was an equal collaboration between Graham Foster, Gastón Keller, Michael Tighe, Michael Bauer and Hanan Lutfiyya.

Acknowledgments

I would like to acknowledge the assistance of Hanan Lutfiyya for her support and supervision in the direction of this thesis as well as Gastón Keller, Michael Tighe, Michael Bauer and Hanan Lutfiyya for their support in co-authoring the work [26] upon which this thesis is based.

Table of Contents

List of Tables

List of Figures

List of Equations

List of Algorithms

List of Appendices

1 Introduction

1.1 Background

Computing today is shifting into large-scale data centres that provide access to computing resources for client applications on a pay-per-usage basis. [1] This affords businesses and other organizations the opportunity to simplify their computing needs by offloading the hosting of their systems onto these off-site data centres. Outsourcing their computing requirements means that businesses no longer have to invest in internal IT solutions with all the costs that come along with that in the form of hardware acquisition and maintenance, staffing and concerns over hosting stability and uptime. These applications may range in type from computationally intensive research applications, or simply web/mail servers. Regardless of the type of application, data centres allow clients to rent out their computing resources and pay only for what they need. [1]

However, as data centres grow, they may simultaneously be hosting a wide variety of client applications across many machines. The significant operating costs associated with these large-scale data centres in the form of hardware acquisition and power consumption means that there is great motivation to organize these client applications onto as few, and as efficient machines as possible, while still ensuring that each application receives an adequate amount of computational power. Additionally, the functionality of each application must not be affected by the actions of neighbouring applications as this may result in an error in one application bring down one or more others. To address this and aid in resource management during application hosting, the technique of virtualization is used as it provides a number of features to aid in data centre management [2].

Virtualization involves a piece of software that wraps around an application, behaving, from the point of view of the hosted application, as a standalone physical machine. The virtual machine will have an operating system and mimic the behaviour of the hardware components of a physical machine. In this way, each application may be isolated within a virtual machine, and requests for physical resources (ex. CPU, memory, network, disk space) are handled by the virtual machine, which may in turn translate these requests into

operations run by the true hosting machine. By placing each hosted application within a virtual machine (VM), the functionality of each application can be effectively isolated from collocated VMs and the applications they house, so the behaviour and actions of one application do not interfere with the functionality of others. Additionally, virtualization allows for the precise division of host resources among hosted VMs. In this way virtualization greatly simplifies the problem of allocating client applications to host machines; however the problem remains of determining an effective allocation that balances the desire to conserve power, while ensuring applications have adequate resources.

1.2 Research In Resource Allocation

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 [3]. 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 to host multiple 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 [4], [5] 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 [6], 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 [7] or are not responsive enough. For dynamic management, Stillwell et al. [5] 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.

A dynamic management strategy is considered 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). This work will focus on two of the most commonly studied goals in the area: (i) minimizing power consumption; and (ii) minimizing Service Level Agreement (SLA) violations. A SLA is considered to be a set of nonfunctional requirements, such as a promised condition on a metric (e.g., response time below a given threshold). Failure to meet the terms of the SLA is termed an SLA violation and is typically associated with some monetary cost to the data centre and so it follows that along with the minimization of power costs, minimization of such SLA violations is a major goal of data centre management.

However, the goals of power consumption minimization and SLA violation minimization 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., [8], [3], [9], [10].

1.3 Thesis Focus

Within a dynamic environment there may be times when one management strategy is more appropriate than another. For example, when overall data centre workload is increasing, this trend would likely cause application resource requirements to potentially grow beyond their current allotments, causing a shortage of resources and, in turn, an SLA violation. During these times, this work proposes that extra care should be taken in managing VMs to guard against this. Conversely, when data centre utilization is stable or decreasing, the probability of SLA violations is likely smaller, and so less caution is required in this regard and the goal of conserving power should take precedence. This work proposes an approach to dynamically switch between two 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 and selectively applies each strategy according to changing data centre conditions. By doing so, better performance in attaining both goals may be produced.

The remainder of this paper is organized as follows. Section 2 reviews recent, relevant work in the area. Section 3 outlines the management strategies used in this work as well as the strategy switching meta-strategy which is the main contribution of this work. Section 4 explains the experiments that were run, including the functionality of the simulator, how workloads were simulated, and the method by which performance was evaluated. Section 5 outlines the results of these experiments. Section 6 analyses these

results and explores potential limitations of the experiments. Finally Section 7 concludes the work and outlines avenues of future work that may warrant exploration.

2. Related Work

Solving the problem of efficiently allocating VMs to host machines while ensuring each application has access to adequate computing resources (termed Quality of Service, or QoS) has been the subject of much research. This work may generally be categorized as falling into two categories. Static allocations generally involve determining an allocation of VMs to host machines that attempts to balance the goals of utilization efficiency and high QoS based on the specified resource requirements of each application. These allocations are, in practice, performed once and attempt to consolidate workloads efficiently while reducing the likelihood of SLA violations occurring due to insufficient host resources. Dynamic allocations are similar in that they balance these same two driving forces of power efficiency and high QoS; however dynamic resource management involves periodic monitoring of VMs and hosts during operation and responding to changes in application workload by either migrating VMs from one host to another to alleviate resource contention in an over-utilized, or stressed host, or migrating all VMs away from an under-utilized host so that it may be placed in a low power state to reduce power consumption.

This section is an overview of the major contributions to each allocation category. In general, research into static resource management largely predates research in dynamic resource management and so, this area will be focused on first. Additionally, within each section, work in the area will be presented in a roughly chronological fashion so as to reflect the progress in the field. Section 2.1 will focus on research into static resource management, Section 2.2 will briefly describe what is termed here as semi-static resource management and finally Section 2.3 will centre around recent work in dynamic resource management.

2.1 Static Resource Management

Initial work in this area [11] focused on leveraging the information available in the definition of each client application in addition to recent developments of the virtualization technology itself. Assumed to be known about each application during

placement is some measure of its expected workload. This allows the specification of a min and max value for each VM which will represent the minimum amount of resources that must be allocated to this application to allow an acceptable level of performance and the maximum expected amount of resources this application will ever request. This work assumes a revenue model for the data centre by which increased levels of application performance produce additional revenue from the client. Under this assumption, each application can be said to have a specific profitability, or utility function associated with it, whereby the revenue generated by an application depends on its level of performance. In this way, the allocation of additional resources to the application beyond the minimum, can be associated with additional revenue from the client. The level of profitability seen at different application performance levels was assumed to be specified in the business agreement with the client and potentially different for every application. To facilitate the sharing of resources in their simulation, the CPU is assumed to be capable of being arbitrarily divided among VMs using a 'shares' approach. Furthermore, a data centre composed of a heterogeneous collection of host machines is assumed with machines having potentially different resource capacities and different levels of power efficiency, measured as power consumption per unit of CPU.

Given this information several works [11], [4], [5], have suggested candidate algorithms to generate allocations:

2.1.1 GreedyMax

The GreedyMax (GM) algorithm [11] allocates each VM enough resources to satisfy its max allocation level. In this way the maximum level of revenue is derived from each client application. VMs are then assigned to hosts in order of the power efficiency of the specific host machine.

2.1.2 GreedyMinMax

The GreedyMinMax (GMM) algorithm [11] allows for the possibility that the profitability of each VM relative to the cost it incurs may be greater at the VM's min level than its max level depending on the particulars of that application. To address this, the algorithm generates a list of VMs to be allocated, with each VM being given two

entries, one with its resource requirements specified at its min value and one at its max value. This list is then sorted according to each VM's profitability at the specified resource level from high to low. VMs are allocated from this list to host machines in a first fit manner in this order with the caveat that when a VM is allocated from the list at either its min or max level, the corresponding entry representing its other resource level is also removed from the list, ensuring that each VM is placed only once.

2.1.3 ExpandMinMax

The ExpandMinMax (EMM) algorithm [11] seeks to take advantage of a key situation which GMM fails to address. When placing VMs using the GMM algorithm above, it may be the case that after each VM has been assigned to a node at either its min or max capacity, there may be capacity remaining on a number of hosts. This negatively impacts the data centre's operation as maximising utilization should be a constant goal, and it may be the case that incrementally increasing the resource allocation beyond the min level for a particular VM may increase its profitability as well.

To address this, when placing a VM, EMM first calculates the estimated profitability of each host if the current VM were to be placed there. Profitability is calculated by first setting all VMs located on the host to require their min level of resources, and then increasing the resource allotment incrementally to those VMs that would provide the greatest profit, per unit of capacity. This incremental increase in resource allotment continues until either the host's capacity is reached or each VM is at its max level. Replacing the first-fit method of placement in GMM with this method would allow the selection of resource allocation levels between the min and max levels for each VM.

2.1.4 PowerExpandMinMax

The PowerExpandMinMax (PEMM) algorithm [11] further builds on the EMM algorithm above by considering the cost of additional host activations in the determination of VM placement. This is designed to address a problem whereby EMM would tend to use all the host machines available during its placement process. For example, when placing a new VM, EMM would tend to place it on an entirely vacant host machine whenever possible as it would be able to expand the new VMs resource allocation without

decreasing that of collocated VMs as would be necessary were it to place the new VM on a partially utilized host. This "server sprawl" has a negative impact on power costs and generally leads to low average host utilization.

To address this problem, PEMM performs VM placement in a similar manner to EMM with the exception that when calculating the profitability of placing the VM on a particular host, there is an additional penalty incurred when the placement would require the activation of an otherwise unoccupied host. This penalty reflects the power costs incurred by supporting an additional host. This power cost is then scaled down to the size of the VM and considered to be the proportional power cost for the VM. Note that even when the expected profitability of placing the VM on a new host is negative, the placement may still take place if the profitability of placing the VM on a different, partially utilized host is even lower (more negative), as may be the case in the situation where more profitable VMs need to be compressed to make room for the new VM.

2.1.5 Genetic Algorithm

A genetic algorithm (GA) [5] was also suggested as a means of producing a high-performing VM placement allocation. The idea here is to have each "chromosome" or candidate represent a simple mapping of VMs to hosts. In support of GA, they suggest a mutation operator, which swaps the host assignment of two randomly selected VMs within the chromosome, and a crossover operator where two chromosomes are split at an identical random location and the pieces concatenated together to form two new VM allocation candidates. If an allocation is produced by either of these operations that is not feasible (ie. the combined resource requirements of collocated VMs exceed the capabilities of the host machine), a greedy algorithm is used to pass through the allocation in an arbitrary order and for each overloaded host, simply move VMs to less loaded machines.

To evaluate this technique, the authors of this algorithm suggest a slightly different performance metric than the min/max algorithms listed above. These authors do not consider that different VMs may have different profitabilities at different levels of utilization. Instead, they seek to maximize the minimum resource allocation to each

application, informally referred to as the application's level of happiness. and in so doing, maximize the average happiness of the client applications as a whole while ensuring that no VM trails significantly behind in terms of performance. Happiness here is calculated as the provided resources divided by the requested resources of each VM.

2.1.6 Results

Experimentation of the genetic algorithm above showed it to be generally less effective than other greedy algorithms the authors implemented [5]. These greedy algorithms are very similar to the greedy algorithms listed above by [11]. They found that their greedy algorithm outperformed the genetic algorithm in over 90% of trials, and among these trials, the average happiness of client applications was 32% higher than the average among GA allocations. Although they note that when the number of generations allowed in GA is increased from 100 to 2000, marginal improvements in allocation performance are seen, the execution time of the algorithm increases to be an order of magnitude larger than the greedy policies and so increasing the number of generations further is not feasible. Note that increasing the population size of GA produced no significant improvements.

In general within these works, evaluation between algorithms is performed by comparing the results of experimentation against a theoretical upper bound, common in evaluation of allocations ([11], [4], [5]). This upper bound represents the theoretical best performance attainable by removing the constraint that each client application must be entirely contained within a single server. Essentially, removing this constraint would allow all client VMs to be placed "end-to-end" in each host with any portion of a VM not fitting on a host "spilling over" into the next host. Hosts are filled in order of efficiency and the theoretical upper bound in terms of VM allocation performance is produced.

Among the algorithms suggested by [11], when evaluated in this way, PEMM was shown to be the most effective at approaching this theoretical upper bound. High performance was theorized to have been produced by the algorithm's ability to intelligently balance the profitability of allowing a greater resource allotment to each VM against the costs of requiring a large number of active hosts.

2.2 Semi-static Resource Management

2.2.1 Variable Workloads

Later work in this area, noted that consideration must be given to the fact that different types of client applications may exhibit dramatically different patterns of resource demand over time. In fact, this often unpredictable variability represents a key challenge in determining an effective VM allocation and has been the subject of much research [12], [13], [14], [15]. For example, a hosted intra-office scheduling application may experience significantly higher levels of activity during regular business hours, followed by comparatively lower levels of activity outside of these times. Therefore, it was proposed that cyclic patterns in workload could be leveraged by attempting, where possible, to assign complementary workloads to each host. This would mean that ideally, when one application can be expected to be requiring additional resources due to its high demand, the requirements of a complementary application would be expected to diminish, freeing up additional requirements.

Indeed, with regard to the these complementary workloads, the authors of a work investigating a periodic recalculation of the allocation [12] demonstrate that even when consideration of daily variations in resource requirements are measured using a granularity of one hour, their placement algorithm yields a 31% reduction the the number of host machines required to support a given workload, as compared to their naive placement algorithm. This reduction in required servers translates to a large reduction in power consumption and cooling requirements, two of the largest operational costs of a data centre.

In addition to cyclic variations in workload, the authors also point out that over a larger time scale, variations in resource requirements of client applications are likely to change. These changes are likely to disrupt the overall performance of a given static placement and so they suggest that it may be the case that some form of "periodic reoptimization" be necessary to deal with these changes. They suggest, perhaps as somewhat of a precursor to dynamic management, that there may be need for some form of automated controller to provide adaptive, self-organizing data centre management.

2.2.2 Semi-static Resource Management

Semi-static resource management was an area briefly explored as a means of dealing with variations in workload that may take place over time [12]. Essentially, semi-static resource management would repeat the static allocation process as described above on some regular time interval. Ideally, this interval would be at least as small as the period over which significant variations in application workload could be observed. Such an algorithm would work by first analyzing historical workload data from each application. Using this data, a forecast could be made to predict expected future workloads. Based on these predictions a remapping of VMs to host machines would be performed to ensure additional resources are available if application workload is expected to increase, or to reclaim resources from an application expected to decline for use elsewhere. This process is then regularly repeated over some time interval, *t*. The authors here investigated varying *t* to different values ranging from fifteen minutes, up to ten hours. This periodic recalculation of the static placement yielded significantly better results in the authors' experimentation. In fact, this approach displayed savings in required host activation of up to 50% as compared to static allocation. However, the author's note an inherent limitation to this approach in that when calculating a new static allocation, perhaps after even very little change in application workload, the new allocation may be dramatically different than the previous allocation. Due to the costs in terms of resources as well as performance degradation associated with moving, or migrating, a VM from one host to another, this dramatic change is not ideal. The authors suggest that future work in this area should address these problems by perhaps searching for a means of mitigating the costs associated with relocating VMs from one host to another, or by adapting the allocation itself to either minimize change from a previous allocation, or generate more robust allocations that would delay the need for reallocation.

2.3 Dynamic Resource Management

Dynamic Resource Management addresses many of the limitations of static resource management that semi-static allocations were designed to also address. The idea that variable workloads require that static allocations be modified, and that these modifications themselves should be based on the nature of the variation by providing

more resources to increasing workloads, and reclaiming resources from declining workloads. The benefits of dynamic resource management lie in the fact that it can selectively deal with problems as they arise, eliminating the need for a regular, periodic reallocation cycle, and that solutions can be applied locally to directly address problem areas in the data centre, without interfering with those portions of the allocation that are performing well. These added capabilities create additional challenges such as how these problem areas can be identified in a timely manner, and how best to resolve the problem. Much work over the past several years has centred around these challenges and will be explored here.

Early work in 2007 [9] introduces a management system named Sandpiper which focuses on the detection and resolution of situations where the combined resource requirements of collocated VMs exceed the capabilities of the host, which they term hotspots. This automated resolution of hotspots was designed to replace manual resolution and so be able to react in a much more timely manner, as well as being tolerant of sudden, short-lived spikes in resource demand. For example, it may be the case that a sudden short-lived increase in demand should be tolerated, causing either a momentary degradation in performance, or the dropping of a few requests, rather than go through the potentially expensive process of relocating the VM, just to have demand return to its former level. To accomplish this, Sandpiper only classifies a host as being stressed if there has been a continued lack of available resources for some period of time, and this trend is predicted to continue at least one time period into the future where the prediction is provided using regression over the past several observed utilization levels. Upon classification of a hotspot in this manner, resolution is performed by the migration of the smallest VM, measured in terms of its memory footprint, as this is indicative of how long the migration will take [9]. This process of VM migration is iteratively repeated until the hotspot is resolved. In the case that there is no host with capacity to accept the incoming VM, Sandpiper relocates the VM on the host which displays the highest volume-to-size ratio (VSR), measured as its CPU requirements divided its memory footprint, with a VM on an unstressed host with the lowest VSR.

While Sandpiper was shown to be an effective way to resolve performance problems and therefore ensure consistent QoS, it wasn't until the following year that power considerations were included in dynamic management systems [10]. Motivated by the average power consumption of typical data centres of 100W / sq.ft, and the noted increase in that figure of 15-20% per year [17], a system was suggested which considered several methods of power reduction. This system, termed pMapper, considered power-saving techniques of three varieties:

- **soft** - hypervisor limits access to hardware of certain VMs to reduce cpu load;
- **hard** - dynamic voltage and frequency scaling (DVFS) throttles down the cpu to reduce power consumption;
- **consolidation** - emptying a lightly-utilized host machine by relocating its VMs to other hosts, thereby allowing the emptied host to be placed in a low-power state (standby, sleep, etc.);

Consolidation was additionally motivated by a recent poll showing motivation for consolidation split between the desire to control server sprawl, reducing power and cooling needs of the data centre and reducing total cost of ownership (TCO) costs (hardware acquisition, maintenance, etc.) [18].

The architecture of pMapper takes a slightly different approach to evaluating performance. This system defines an SLA as a set of minimum performance requirements for an application. Specifically, an SLA will require that performance be at least at a specified minimum level, for at least a specified proportion of the time. For example, an SLA may require that an application be allotted at least 95% of its requested resources for at least 98% of the time. In this sense, application performance can be viewed as a constraint rather than as an evaluation metric. This reformulates the VM allocation problem as one of minimizing power consumption subject to the constraint that SLAs must be satisfied. In evaluating pMapper, and employing combinations of the three power-saving techniques listed above, the authors were able to produce up to 25% savings in power compared to static allocations. However, they note that power savings

diminish as total data centre utilization increases due to the fact that the most significant power savings were produced by the consolidation technique, as opposed to the soft and hard power-saving measures, and as each host machine approaches its capacity, the opportunity for consolidating workload onto fewer machines diminishes. The reason for the high performance of consolidation relative to the hard and soft power-saving measures is suggested to be due to the fact that up to 70% of the maximum power consumption of a host machine is present even when the host is idle [19] and so the primary goal of a power-saving management system should be to have as few host machines active as possible at any given time.

Further work in 2011 [20] refined these ideas by expressing the problem of dynamic resource management simply as a combination of VM relocation, as a process to relieve stress situations, and VM consolidation, as a means of saving power and preventing server sprawl. VM relocation can be further divided into two components: VM target selection to select a VM from a stressed host for relocation, and VM placement, to locate a host to receive the VM.

Additionally, the authors suggest a number of potential improvements to the method by which host machines can be identified as being under- or over-stressed. Up to this point, this identification process employed static thresholds, common across all machines, that mark upper and lower bounds on the cpu utilization of the host. For example, if a host's utilization rises above 90% for some period of time, it should be considered stressed and in danger of causing SLA violations and VM relocation should be performed, whereas if its utilization falls below 50% it should be considered underutilized and marked for consolidation. This technique, the authors suggest, could be improved upon by considering the variability in the workload that each host machine is under. For example, if a host's client applications are relatively stable, experiencing very little fluctuation in their workload level, it may be safe to allow a utilization of 95% before considering VM relocation, whereas if the variability in workload of the machine is very high, an upper threshold of 80% may be appropriate for preventing SLA violations. To solve the problem of determining appropriate upper and lower thresholds for each host, the authors suggest the following techniques:

- **Median Absolute Deviation (MAD)** - Setting thresholds that correspond to the deviation in the median workload value, allows the thresholds to be tolerant of momentary spikes in workload and prevents far outliers from skewing this value too greatly.

- **Interquartile Range (IQR)** - Setting lower and upper bounds on cpu utilization that correspond with the 25th and 75th percentile of observed workload levels respectively. This also provides a tolerance to sudden workload spikes as above.

- **Linear Regression (LR)** - Avoiding the use of an upper threshold entirely, linear regression can be used to predict whether future utilization levels will exceed the capacity of the host machine. As this technique is more sensitive to shorter term trends in workload that the previous techniques ignore, there is less importance on providing a buffer between an upper threshold and the full capacity of the machine. By discarding these thresholds, utilization can more closely approach the capacity of the machine. Based on these predictions for high or low utilization going forward, the host can be marked for VM relocation or consolidation as appropriate.

To solve the problem of VM target selection when a host has been marked for VM relocation, the authors suggest the following algorithms [20]:

- **Minimum Migration Time Policy (MMT)** - Selection for migration of the VM that will take the least amount of time to migrate, calculated as the size of the memory footprint divided by the bandwidth of the network connection.

- **Maximum Correlation Policy (MC)** - Based on the idea that VMs with similar workload patterns will be more likely to increase in demand together, thereby magnifying the impact on the host's resources, this algorithm seeks to select for migration that VM whose workload best correlates with the workloads of other collocated VMs, thereby reducing the risk of a simultaneous increase in workload between several VMs, which would be more likely to cause a host to become stressed.

- **Random Choice Policy (RC)** - Use a uniformly distributed random variable to index a VM within the host to be selected for migration.

Finally, to address the problem of host selection to receive migrating VMs, the authors note that the problem can be represented as a bin packing problem with variable bin sizes and prices. In this case, the size of the bins represents the CPU capacity of the host machine and the price of the bin represents the host's power efficiency. This allows for the representation of a heterogeneous collection of host machines as is likely to be the case in a commercial setting. However, the problem of bin packing is NP-hard and so to address it, the authors use a modification on the best fit decreasing algorithm (BFD) whereby VMs are sorted in decreasing order based on their workload, and placed on hosts such that the minimum power consumption increase is produced, essentially utilizing the most efficient host machines first.

Evaluating the overall performance of an experiment using the metrics of frequency of SLA violations and gross power consumption can be difficult as each is measured in different units and vary over different ranges of values. Additionally, they are often considered to be in conflict with one another as improving performance in one area for a given workload can usually correlate with diminished performance in the other. Therefore, to evaluate their proposed algorithms the authors measure simulation performance against a combined energy - SLA metric termed, Energy-SLA Violation (ESV). This metric is calculated simply as the product of the energy consumption of the data centre and the number of SLA violations. In this way, slight increases in frequency of SLA violations can be considered acceptable if accompanied by significant savings in terms of energy consumption and vice versa. To solve the problem of balancing the priority of each metric, the authors additionally suggest a weight value, through which the relative importance of each metric can be adjusted.

From the authors' experimentation, linear regression (LR) proved the most effective way of marking a host machine as stressed, significantly outperforming both MAD and IQR. This suggests that it is more important to react to sudden changes in workload, as LR is able to do, rather than smoothing out the workload pattern as is done by MAD and IQR.

Additionally, MMT was shown to outperform RC and MC suggesting that migration time was a more important factor to minimize than correlation of workloads between VMs.

In a later 2012 work [16], these authors further refine the subject area of dynamic resource management by suggesting several more VM selection algorithms for evaluation. Note that VM placement was performed in these experiments in a similar manner to the authors' previous work, [20], using their modified best fit decreasing algorithm. Their proposed VM selection algorithms are as follows:

- **Minimization of Migrations (MM)** - Select the minimum number of migrations necessary to relieve a stress situation on host. That is, the algorithm selects for migration the VM with the smallest workload big enough to entirely relieve the stress situation. If no single VM has a large enough utilization to relieve the stress situation itself, the largest VM is selected for migration and the algorithm repeats.

- **Highest Potential Growth (HPG)** - Select for migration that VM which is experiencing the lowest current utilization, relative to its expected utilization level. This VM is therefore considered to have the highest potential growth in workload and is removed so as to decrease the likelihood of total host workload increasing in the future.

- **Random Choice (RC)** - As above, use a uniformly distributed random variable to index a VM on the stressed host and select it for migration

Experimentation on a simulated data centre [21] of the above algorithms yielded no significant difference in energy savings between the three proposed algorithms. However, in terms of frequency of SLA violations, MM and RC were found to produce significantly fewer violations than HPG, indicating that potential for future increase in workload should not be an immediate factor in the event of a stressed host being identified. Additionally, when comparing MM against RC, RC was found to require up to 10x more migrations throughout the simulation, leading the authors to suggest MM as

their best-performing selection algorithm. However, although fewer in number, the size of each migration is not addressed.

A 2012 work identified a further difficulty of optimizing VM placement in terms of host utilization and power consumption in that these goals can be considered to be somewhat conflicting. [14] That is to say that actions taken to improve performance in one area often come at the cost of decreased performance in the other. This was seen in the evaluation of a number of different VM relocation policies investigated in this work. These algorithms are all based around the first-fit heuristic, used to approximate a solution to the NP-hard vector bin-packing problem to which VM allocation is related. Variations of this heuristic are widely used to aid in VM placement [20], [16], however sorting techniques vary. In this work, the authors investigate sorting techniques that differ simply in that under a stress situation VMs are either sorted for migration in an increasing or decreasing manner by their CPU workload, and for host selection, host machines are sorted by their CPU availability in either an increasing, decreasing or mixed manner, where mixed refers to sorting partially utilized machines in an increasing manner, and underutilized machines in a decreasing manner. This method of sorting VMs and hosts is a well established technique and is utilized in the above dynamic resource management works [20], [16].

The comparison of the six algorithms produced from the permutations of these sorting methods reveals the conflicting nature of the goals of high performance, and low power consumption. A comparison of two of these algorithms is summarized in Table 1. The algorithms were named according to how they sort both VMs on stressed hosts and target machines. Note the First Fit Increasing-Decreasing (FFID) algorithm, in which VMs on stressed machines are sorted in an increasing manner by CPU and target host machines are sorted in a decreasing manner by CPU utilization, demonstrates a high host utilization, corresponding to high power efficiency, coinciding with a relatively poor request drop rate. However, the First Fit Decreasing-Increasing (FFDI) algorithm, in which VMs are sorted in a decreasing manner and target host machines in an increasing manner by CPU utilization, demonstrates the opposite, a lower host utilization, corresponding to a lower power efficiency rating, with much higher performance, seen in

a 65% lower request drop rate. This highlights the idea that these primary operational goals of a data centre are often at odds with one another and that further research is warranted to investigate the possibility of optimizing VM allocations for both these goals

Metrics	FFDI	FFID
# Migrations	480 [17.7]	1755 [110]
Avg. Active Hosts	70.2 [1.2]	61.5 [0.5]
Host-hours	15209 [82.8]	13634 [70.4]
Host Utilization	61.5% [0.3]	67.6% [0.3]
Kilowatt-hours	6142.9 [22.1]	5715.3 [19]
Dropped Requests	0.6% [0.04]	1.7% [0.06]

Table 1: A comparison of two of the VM relocation policies [14]

2.4 Conclusion

With the advent of virtualization technology, the opportunities for numerous large-scale data centres to provide effective solutions to the IT requirements of their consumers have increased dramatically. As a consequence of this proliferation, data centre operators have had to balance the complex and often conflicting goals of providing a high QoS to their clientele, and ensuring efficient utilization of their resources in order to achieve maximum capacity with minimal operating costs. Initially, work in this area focused on solving the problem of safely assigning client applications to host machines in a single, stable, allocation. This was designed to be a robust allocation that would require few manual adjustments in the future. However, in the past several years, developments in the ability to monitor and update an allocation on-the-fly without significant degradation to QoS, combined with increasing energy pressures and costs have enabled the development of more complex dynamic resource management techniques. These techniques enable allocation of VMs that will more fully utilize each host machine relying on the assurance that any situation in which the requirements of client applications exceeds the host's capabilities can be automatically detected and mitigated in a timely manner. Due to the complex nature of dynamic resource management and mounting pressures as applications of this technology become more popular, research in

this area continues to pursue techniques to improve QoS while minimizing power consumption.

3. Management Strategies

The term management strategy here is defined to represent an instantiation of policies to perform the placement, relocation and consolidation of VMs on host machines. This section presents three management strategies: (i) Power, which is designed to emphasize the reduction of power consumption (Section 3.3), (ii) SLA, which is designed to emphasize the minimization of SLA violations (Section 3.3), and (iii) Hybrid, which is designed to pursue both goals simultaneously (Section 3.4). Finally, this section will introduce a meta-strategy called Utilization-Based Dynamic Strategy Switching, Util-DSS. This meta-strategy dynamically switches between the Power and SLA strategies at runtime based on the monitoring of changing data centre conditions (Section 3.5).

3.1 Terminology

This section presents the terms and metrics used in the description of management strategies. These terms are also defined in Appendix A.

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: The capacity of a CPU is quantified using CPU shares, where each CPU core has a specific number of shares which represents its computing power. In this 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:

$$p_h = \frac{cpuInUse_h}{powerConsumption_h}$$

Equation 1 - Power efficiency of a single host

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 to increased power efficiency for that host. This metric is used to calculate the power efficiency for the entire data centre, p_{dc}, calculated as in Equation 2.

$$p_{dc} = \frac{\sum_{h \in hosts} cpuInUse_h}{\sum_{h \in hosts} powerConsumption_h}$$

Equation 2 - Data centre power efficiency

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

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

Optimal Power Efficiency: Optimal Power Efficiency, pdcopt, represents the best possible power efficiency achievable at the data centre level, given the current workload and set of host machines available. This theoretical upper bound is similar to that proposed in past works [11], [4], [5]. Although unattainable in reality, this value serves as a useful bound against which to measure observed data centre power efficiency. 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. First, the total CPU-in-use across the data centre is calculated. Then the available hosts are ordered by maximum power efficiency, and the CPU-in-use is allocated to the hosts such that each

host is allocated 100% of its CPU capacity. The optimal power efficiency, pdcopt, is calculated as the power efficiency of the data centre given this allocation.

3.2 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 different VM Relocation policies may make the determination of whether a host is stressed in slightly different manners based on how the most recent measurements of host utilization are considered. This *stress check* differs in its determination of a stress level depending on the primary goal of the strategy. An example of the different manners in which this *stress check* is performed are seen in the difference in VM Relocation policies seen in the Power and SLA strategies below.

3.3 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, the VM Placement, VM Relocation and VM Consolidation policies that comprise these two strategies are explained. Much of the existing work on dynamic management uses some form of First Fit heuristics. The work described in [5] (for static workloads) and [14] (for dynamic workloads) studied variants of First Fit heuristics and found that they work best in practice at determining VM allocations. 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 of 60%.

3.3.1 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*), underutilized (*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.

```
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

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 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. This change in sorting allows the SLA strategy to pursue the primary goal of minimization of SLA violations over the secondary goal of maximizing power efficiency.

3.3.2 VM Relocation

This management operation runs frequently over short intervals of time, so as to detect stress situations quickly. 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 performs its *stress check* by classifying a host as 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*), underutilized (*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).

```
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

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 as follows: 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 $stress_{CPU}$.

3.3.3 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. As more frequent consolidation can be considered a more aggressive approach to saving power as it may increase the risk of SLA violations, the VM Consolidation interval is set to 1 hour for the Power strategy and to 4 hours for the SLA strategy. 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 first classifies hosts in their respective categories: stressed (*z*), partially-utilized (*p*), underutilized (*u*), and empty (*e*), and powers off *e*. The policy then sorts *p* and *u* in decreasing order first by maximum power efficiency and then by CPU utilization and builds a list of target hosts by concatenating *p'* and *u'*. 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*. Following a First Fit heuristic, the policy attempts to vacate every host *h* in source by migrating their VMs into hosts in *target*. 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. Given that *source* and *target* are not disjunct, measures are taken to avoid using a host both as a source and target for migrations. The functionality of the Power strategy's consolidation algorithm is outlined in Algorithm 3.

```
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

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 to generate a list of target machines, *u* is sorted in increasing order by CPU utilization. The functionality of the Power strategy's consolidation algorithm is outlined in Algorithm 3.

3.4 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.

This Hybrid strategy is intended to serve as a representative example of how a single strategy may pursue the goals of reducing power consumption and SLA violations simultaneously. To validate the performance of this strategy so as to establish it as a

valid benchmark against which to compare the strategy switching technique (described in Section 3.5), the performance of the Hybrid strategy is compared against an implementation of the Minimization of Migrations algorithm. Through the experimentation performed in [16], this algorithm was found to produce the best compromise among several candidate strategies when trying to pursue both goals simultaneously. The results of this comparison are described in Section 5.1.

3.5 Utilization-Based Dynamic Strategy Switching

Through preliminary 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 generally more effective as it places greater emphasis on preventing SLA violations. Conversely, when utilization is stable or decreasing, thus decreasing the likelihood of stress situations and increasing the likelihood of hosts becoming underutilized, the Power strategy is more effective as it can more quickly perform VM Consolidation to conserve power.

The Utilization-Based Dynamic Strategy Switching (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 outlined in Algorithm 4.

```
1:  util = getDCUtilWindow(dc,n)
2:  m = linearRegression(util)
3:  if activeStrategy = Power_Strategy then
4:     if m > mSLA then
5:        Switch to SLA_Strategy
6:     end if
7:  else if activeStrategy = SLA_Strategy then
8:     if m < mPower then
9:        Switch to Power_Strategy
10:    end if
11: end if
```

Algorithm 4: Util-DSS Switching Conditions

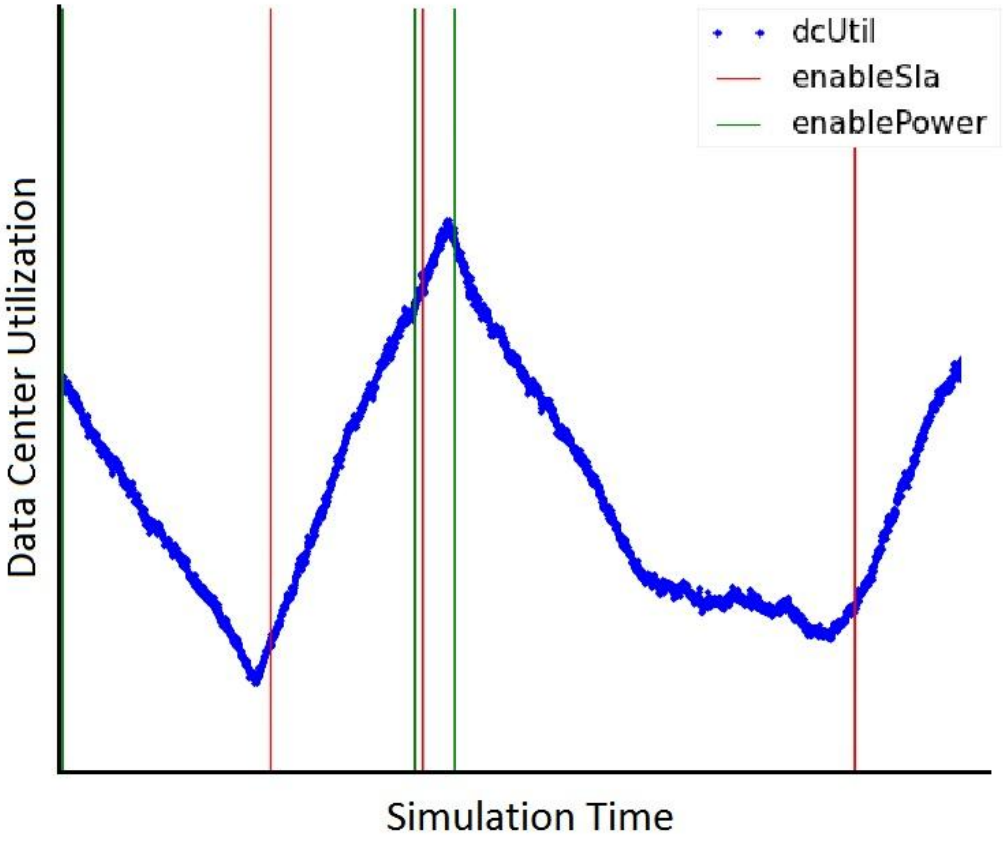
where *getDCUtilWindow(dc)* returns the data centre utilization measurements over the past *n* measurements. No other conditions cause a strategy switch beyond those outlined in Algorithm 4. If the *activeStrategy* is *Power_Strategy*, the only event that could cause a switch to *SLA_Strategy* is an increase of *m* above m_{SLA} and if *activeStrategy* is *SLA_Strategy*, the only event triggering a switch to *Power_Strategy* would be a decrease of *m* below m_{Power}.

The switching thresholds m_{SLA} and m_{Power} were determined using a brute force search across all combinations of threshold values between -0.005 and 0.015 in increments of 0.0005. Threshold values less than -0.005 or greater than 0.015 produced great drop-offs in performance due to the switching thresholds being so far from typically observed values of *m* that no thresholds were ever crossed. In this case, strategy switching would either never occur, or in the specific case of an overly low m_{SLA} threshold and an overly high m_{Power} threshold, a strategy switch would occur every time the switching algorithm was triggered, causing thrashing. Additionally, at this fine granularity of 0.0005 only small changes in performance were seen between adjacent threshold values, and so it is assumed that no threshold values produced by smaller granularity searches would produce significantly better performance than the results of this search. Due to the feasibility of searching the entire population of candidate thresholds down to this fine level of granularity, an exhaustive search over all possible candidates was performed instead of some sort of artificial intelligence threshold-learning technique. This was done because, under the assumptions of acceptable threshold ranges,

and fine search granularity above, any given learning algorithm could only perform, at best, as well as this exhaustive search.

The search for threshold values used a set of 5 simulations, each of a duration of 10 simulated days and each using a different random workload (workload generation is described in Section 4.2). Each candidate pair of threshold values was evaluated against these 5 simulations and ranked by score (scoring technique described in Section 4.4). Threshold candidates generally performed consistently across the 5 simulations. Due to this observed consistency, the selected thresholds, chosen for their high performance during this search, are assumed to be high performing in general and can be used across all randomized workloads under the conditions of these experiments.

Figure 1 below represents the operation of Util-DSS over a segment of simulated time. The blue series represents measured values of overall data centre utilization over time. Red and blue vertical lines represent switches in management strategy to the SLA and Power strategies respectively. Note that when data centre utilization begins to increase, the slope of the line-of-best-fit crosses the m_{SLA} threshold and a switch is made to the SLA strategy, represented by a vertical red line, and when utilization begins to level off or decrease, the slope of the line-of-best-fit falls below m_{Power} and a switch is made back to the Power strategy.


dcUtil
enableSla
enablePower
Data Center Utilization
Simulation Time

4. Experiments

This Chapter outlines the setup of DCSim [22], the simulator used to run the experiments, as well as the implementation of variable workloads. Additionally, this Chapter outlines a novel evaluation technique whereby the performance of multiple strategies can be directly and quantitatively compared using the very different metrics of SLA violations and aggregate power efficiency.

4.1 DCSim

Experimentation is conducted by simulation using DCSim [22]. The 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 8GB 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 [23]. 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 [20].

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.

4.2 Variable Workload

A data centre experiences a highly dynamic workload, driven by VM arrivals and departures, as well as the dynamic workloads and resource requirements of VMs. Here, random *workload patterns* are generated to evaluate our strategies, where a workload pattern consists of a set of VMs with specific start and stop times, each with dynamic trace-driven resource requirements. As resource allocation is naive of application-type, it is likely that hosted VMs in a data centre will embody a wide variety of application types. Therefore, in these simulations, a variety of traces, representing a variety of applications from different real-world sources were used. Each VM is driven by one of 5 individual traces: the *ClarkNet*, *EPA*, and *SDSC* traces [24], and two different traces from the *Google Cluster Data* trace [25]. 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.

4.3 Util-DSS Parameter Selection

The switching thresholds m_{SLA} and m_{Power} used for Util-DSS are arrived at using the method described in Section 3.5. They are each selected to be 0.00255 as this combination produced the best aggregate performance across the five random workloads and are therefore submitted as the best performing switching thresholds. Note that as the value of each threshold is the same, the behaviour of Util-DSS will be simplified to switch to the Power strategy if the rate of change in data centre utilization, m, falls below 0.00255 and switch to the SLA strategy if the rate rises above this value.

The frequency with which the strategy switching method is evaluated was selected to be every one hour following informal, preliminary experimentation over multiple frequency values. Additionally, the monitoring window of Util-DSS, from which measurements of utilization are recorded and considered during strategy switching evaluation, is set to a size of 2 simulated hours in 6, 20-minute intervals. This timing was selected during preliminary experimentation as a balance between being sensitive to changes in workload patterns, but avoiding thrashing between strategies caused by overreacting to minor fluctuations in data centre utilization.

4.4 Strategy Evaluation and Comparison

In order to evaluate the effectiveness of the strategies, two metrics are used: power efficiency (p) and SLA violation (s). However, 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 frequent SLA violations, it is difficult to conclude which strategy is preferable. In practice, this 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 the absence of well-defined business rules governing the relative value of each metric, a method is required to evaluate performance based only on the observed values of these metrics.

In order to determine whether DSS can offer improved results over a single strategy, this work proposes a method of evaluating the performance of a strategy *relative to other strategies' performance* based on the experimental results of each. Using the SLA and Power strategies as benchmarks, their SLA violation and power efficiency results can be used 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 3.

$$s_{norm} = \frac{(s_i - s_{best})}{(s_{worst} - s_{best})}$$

$$p_{norm} = \frac{(p_{best} - p_i)}{(p_{best} - p_{worst})}$$

$$v_i = (s_{norm}, p_{norm})$$

Equation 3: Score Vector Calculation

where pnorm is the normalized power efficiency and snorm is the normalized SLA violation. Note that $p_{best} > p_{worst}$, but $s_{best} < s_{worst}$, so the normalization equations differ to reflect this. Using the normalized vector, v_i , it is possible to calculate its L2-norm, $|v_i|$ (Equation 4), and use this as an overall score ($score_i$) for the candidate strategy.

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

Equation 4: Score Calculation

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 is superior to that of 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 this work, the workload pattern experienced by the data centre is varied from one trial to the next. 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. This is the method used to evaluate and compare candidate management strategies.

5. Results

This Chapter will outline the results of experiments comparing Minimization of Migrations (MM) [16] with the Hybrid strategy, outlined in Section 3.4. This will determine the validity of the Hybrid strategy and in so doing, establish its effectiveness as a competitor with Util-DSS. Following these experiments, this section will outline the results of a comparison between Hybrid Strategy and Util-DSS over a much larger number of trials. The results of these experiments will determine the validity of Util-DSS as an effective data centre management technique.

5.1 Validation of Hybrid Strategy

In order to determine the validity of the Hybrid strategy as an effective management strategy against which a fair comparison of the effectiveness of Util-DSS can be made, experiments were run comparing Hybrid strategy to an implementation of the Minimization of Migrations strategy [16] outlined above. The experiments were run using the same configuration of DCSim outlined above in Sections 4.1 and 4.2. The evaluation was performed with a set of 5 randomly generated workload patterns. The results of this evaluation are presented in Table 2. Reported metrics were averaged across all workload patterns and the standard deviation is presented in square brackets. The following metrics are reported: 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; and SLA Violation is s over the entire simulation. Note that as the purpose of this experiment is to determine the validity of the Hybrid strategy, and the components of the Hybrid strategy are drawn from the Power and SLA strategies, no calculation of overall score is performed as the similarity between Hybrid and the baseline strategies would likely skew the results.

	Minimization of Migrations	Hybrid Strategy
Avg Active Host Util.	68.33 [4.28]	80.59 [0.25]
# of Migrations	503,130 [38,414]	13,150 [1,556]
Power Consumption (kWh)	4,694 [357]	4,754 [524]
Power Efficiency	62.53 [3.36]	335.02 [2.14]
SLA Violation	9.098 [0.120]	0.409 [0.008]

Table 2: Results of comparison between Minimization of Migrations and Hybrid Strategy

5.2 Util-DSS Results

The results of the experiments are presented in Table 3. 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. The following metrics were reported: 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; and SLA Violation is s over the entire simulation. Also reported was the normalized SLA and power values for each strategy, as well as the score. Figure 2 presents a graphical representation of the scores.

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.

	SLA	**Power**	**Hybrid**	**Util-DSS**
Avg. Active Host Util.	75% [0.4]	88% [0.4]	81% [0.4]	82% [1]
# of Migrations	15818 [2292]	24378 [3311]	14643 [1930]	19580 [3047]
Power Consumed (kWh)	5488 [703]	4384 [519]	5049 [679]	4778 [583]
Power Efficiency	60.6 [2.4]	75.2 [2.0]	65.9 [2.7]	69.8 [2.3]
SLA Violation	0.033% [0.01]	0.474% [0.05]	0.092% [0.01]	0.220% [0.05]
s_{norm}	0.0	1.0	0.135 [0.01]	0.425 [0.09]
p_{norm}	1.0	0.0	0.636 [0.06]	0.373 [0.08]
Score	1.0	1.0	0.651 [0.05]	0.576 [0.041]

Table 3: Experimental results comparing Util-DSS and Hybrid management strategies. Result data drawn from previous publication of these experiments by this author and others [26].

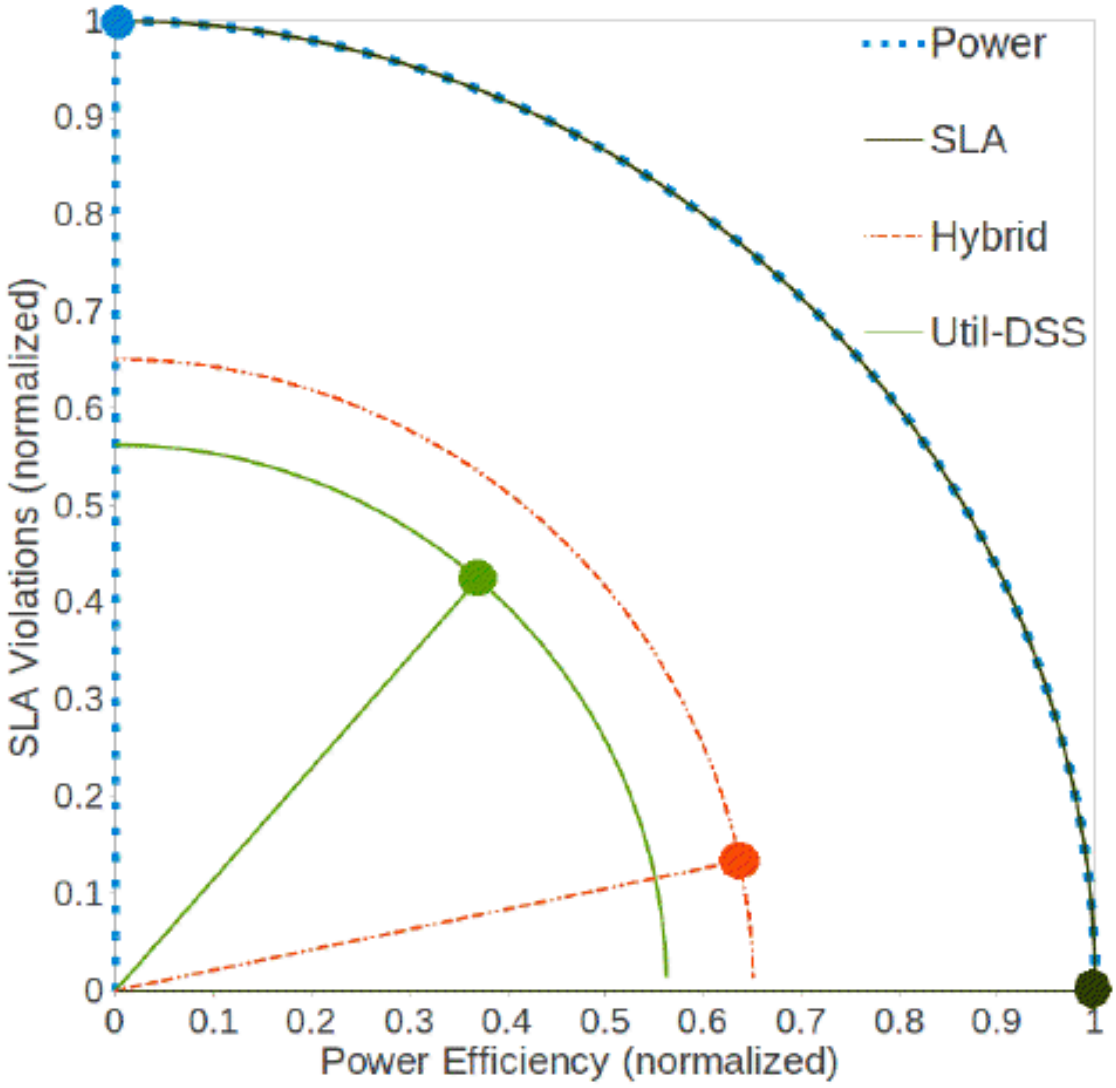


Figure 2: Graphical representation of results of comparison between Hybrid and Util-DSS strategies. SLA and Power strategy results are also displayed for reference, however their results, by definition, form the axes of the graph.

6. Discussion

Both Util-DSS and the Hybrid strategy produced better overall performance than either of the single-goal Power and SLA strategies with an improvement in score of around 40%. Furthermore, when compared against Hybrid, using the scoring method outlined above, Util-DSS outperformed the Hybrid strategy by ~11.5%. This improvement was largely seen in a 271 kWh reduction in power consumption. Although Util-DSS had a higher percentage of SLA violations than Hybrid, the savings in power were more than enough to make up for this.

Additionally, the results from Util-DSS display a near-perfect balance between the Power and SLA metrics. However, as these results are calculated relative to the results of the Power and SLA strategies, the observed balance of Util-DSS relies on the truth of the assumption that Power and SLA strategies are each equally performing strategies that differ only in a separate, but equal degree of preference for their primary goals. In the absence of well defined business rules quantifying the value of performance in each metric, specific statements of balance between metrics are difficult.

A potential drawback of Util-DSS when compared to the Hybrid strategy is the 34% increase in number of migrations. Although the migration count of each strategy falls far below the migration count observed in the Minimization of Migrations algorithm (evaluated in Table 2), it may be the case that in a situation where network bandwidth is highly constrained, this increase in frequency of migrations would deteriorate the performance of Util-DSS. It is likely that this increase in migration count of Util-DSS over Hybrid is due to the aggressiveness of the consolidation brought on by switching to, and operating under, the Power strategy over the course of Util-DSS's operation. This is supported by the much larger number of migrations observed when the Power strategy is run in isolation (See Table 2).

Further work may be needed to address the issue of strategy evaluation beyond the scoring method suggested here. This scoring method is inherently relative as it rests on normalizing metric values within ranges defined by other strategies. Although it builds

on the Energy-SLA Violation (ESV) suggested in [20] by providing a method of equating performance in metrics that range over different values, it is still ignorant of the relative value that performance in each metric correlates with in a real-world data centre. As such, it may be that the balanced performance in each metric displayed by Util-DSS is not, in fact, desirable.

7. Conclusion

The problem of resource management in virtualized data centres is well researched. However, pursuing multiple, conflicting management goals is difficult. Additionally, without any manner of comparing performance between these different metrics, evaluation of management strategies is also difficult. In this work, a technique is proposed to leverage the fact that under certain data centre conditions, a strategy focusing on just one goal may be appropriate, while under different conditions, a different goal should take precedence. Specifically, by monitoring data centre utilization trends, Util-DSS is able to focus on maintaining QoS when this metric is likely to be constrained, and conversely, when utilization is decreasing, the risk of SLA violations is much lower, allowing management to focus more strongly on maximizing power efficiency.

Additionally, this work proposes a novel method of comparing strategies by normalizing the performance in each area between predefined ranges. In doing so, a comparison can be made between strategies that may improve performance in one area while worsening performance in another. In this absence of business rules that define the value in each metric in terms of money, this method provides a direct method of quantifying performance between independent strategies.

Using the proposed scoring technique, Util-DSS was shown to perform about 40% better than the single-goal strategies, Power and SLA. Additionally, when compared against the Hybrid strategy, designed to compromise between the two goals as well as possible, Util-DSS was shown to outperform Hybrid by about 11.5%.

7.1 Future Work

In the simulation of data centre activity, this work only considers the CPU capacity of simulated machines. Although CPU is likely to be the most constrained of computing resources, future work in this area should consider other resources such as memory and bandwidth. Additionally a future characteristic of large-scale data centres is the division of computing resources into a hierarchical structure with racks and clusters of machines. This introduces additional challenges in data centre management with constraints that

certain applications must be collocated in the same rack as other applications if they depend upon each other, or perhaps that they must not be located in the same rack as they serve as backups for one another and should be separated. These constraints add complexity to the problem and must be addressed.

Furthermore, in proposing that different sets of data centre conditions warrant different management strategies, this work proposes that data centre utilization is a good metric to determine switching times. Future work is warranted in investigating other switching conditions. For example, work done by this author and others [26] has investigated switching strategies based on the distance the observed level of SLA violations and power efficiency are from certain predefined goals, or switching strategies when these observed values cross some predefined threshold. Additional work is warranted in evaluating other techniques for strategy switching.

Although the strategies selected for switching in this work were designed such that there was one strategy designed to prioritize each primary management goal. Future work in this area may focus on the selection of either different strategies from those used in this work, or perhaps more than two strategies. Such work would likely require a more sophisticated switching technique to accommodate this.

The switching techniques outlined in this work require the definition of switching thresholds be searched for beforehand, and then held static during the simulation. A useful development in this area would be the online searching for switching thresholds. Essentially, this would require that threshold values be set to some arbitrary starting value and then as the simulation progresses, using some sort of heuristic, a determination is made on the effectiveness of the current switching thresholds and they are adjusted techniques derived from the field of artificial intelligence. This difficult problem would benefit greatly from future work and greatly increase the performance of likely any sort of strategy switching technique.

In general, further work in the area of dynamic resource management for virtualized data centres holds the opportunity to make great advancements in terms of reliability and quality of service as well as greatly reducing the power consumption of such systems.

Given the increasing popularity and ubiquity of cloud computing, advancements in these systems will provide great benefit not just to the academic field, but to the society that benefits from it as well.

References

[1] G. Wang, T.S. Ng, "The impact of virtualization on network performance of amazon ec2 data centre," in INFOCOM proceedings, 2010 IEEE, 2010.

[2] A. Weiss, "Computing in the Clouds," *Computing*, pp 16-25, 2007.

[3] 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, pp. 119-128.

[4] 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.

[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.

[6] 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.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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, pp. 119–128.

[13] X. Zhu, D. Young, B. Watson, Z. Wang, J. Rolia, S. Singhai, B. Mckee, C. Hyser, D. Gmach, R. Gardner, T. Christian, L. Cherkasova, “1000 Islands: Integrated Capacity and Workload Management for the Next Generation Data Center,” in International Conference on Autonomic Computing (ICAC), pp. 172-181. 2008.

[14] G. Keller, M. Tighe, H. Lutfiyya, M. Bauer, “An Analysis of First Fit Heuristics for the Virtual Machine Relocation Problem,” in 6th International DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM), 2012.

[15] 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.

[16] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,” Future Gener. Comput. Syst., vol. 28, no. 5, 2012.

[17] Control power and cooling for data center efficiency HP thermal logic technology. An hp bladesystem innovation primer (June 2006)

[18] J.R. Phelps, “Data Center Conference 2007 Server Consolidation Poll Finds Projects Increasing, Reasons Changing and Outside Assistance Growing,” Gartner, Inc., Jan. 2008.

[19] Fan X, Weber WD, Barroso LA. Power provisioning for a warehouse-sized computer. Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA 2007), ACM New York, NY, USA, 2007; 13-23.

[20] 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.

[21] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F.D. Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience, 41 (1) (2011) 23–50.

[22] 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, Oct. 2012.

[23] (2012, Aug.) Specpower_ssj2008 benchmark. Standard Performance Evaluation Corporation. [Online]. Available: http://www.spec.org/power ssj2008/

[24] (2012, Aug.) The internet traffic archive. [Online]. Available: http://ita.ee.lbl.gov/

[25] (2012, Aug.) Google cluster data. Google Inc. [Online]. Available: http://code.google.com/p/googleclusterdata/

[26] G. Foster, G. Keller, M. Tighe, H. Lutfiyya. M. Bauer, "The Right Tool for the Job: Switching Data Centre Management Strategies at Runtime," IFIP/IEEE International Symposium on Integrated Network Management, 2013.

Appendices

Appendix A - Definition of Terms

VM	Virtual Machine. Software that wraps around client application and behaves as a stand-alone machine to provide isolation and consistent resource provision to the application.
SLA	Service Level Agreement. Contract between data centre and client governing acceptable performance levels of client application.
SLA Violation	Violation of the SLA. This is caused by an under-provisioning of a client application producing a measurable degradation of application performance. In this work this is calculated as the percentage of cpu shares requested that were not provided.
QoS	Quality of Service. Subjective measurement of the frequency with which the SLA is violated. High QoS corresponds with few SLA violations
t	Time interval with which a static allocation would be recreated in [12] as a means of dealing with variable workloads. This was termed Semi-static Resource Management
hotspot	Term used in [9] to describe a host machine with insufficient resources to meet the demands of its hosted VMs.
VSR	Volume to Size Ratio. This metric was used in [9] for VM selection and is calculated as a VMs CPU requirements divided by its memory footprint.
TCO	Total Cost of Ownership. Figure representing the total cost to the data centre of owning and operating hardware (hardware acquisition, maintenance, staffing, power consumption, etc.)
ESV	Energy - SLA Violation. Metric used in [20] as a simple means of comparing energy consumption and SLA violations against each other
Data Centre Utilization	The overall utilization of the data centre calculated as the percentage of total CPU capacity in the data centre that is currently in use.

CPU Shares	Means of quantifying the capacity of a CPU, representing its computing power. In this work 1GHz = 1000 CPU shares
p_h	Power efficiency of a particular host, *h*, measured as the number of CPU shares in use divided by the host's current power consumption
p_{dc}	The sum of all the CPU shares in use across all hosts divided by the total power consumption of those hosts
Maximum Power Efficiency	Metric representing the best power efficiency of a given host calculated as the power efficiency of the host at maximum CPU utilization
pdcopt	Optimal Power Efficiency of the data centre. Theoretical upper bound calculated as the p_{dc} value when the total workload of the data centres is distributed across the host machines filling them in decreasing order of power efficiency to 100% utilization. In practice the VMs may not be able to be split this way across hosts and so this value is a theoretical maximum
$stress_{CPU}$	Upper threshold on host utilization beyond which (subject to the particular rules governing *stress check*) the host machine is considered stressed
$minUsage_{CPU}$	Lower threshold on host utilization below which the host machine is considered underutilized
stress check	Determination of whether a host should be considered stressed based on whether its CPU utilization exceeds $stress_{CPU}$. Different policies may be more or less tolerant of momentary spikes above $stress_{CPU}$ when classifying a host as stressed or not
source	In policies performing migrations such as VM Relocation and VM Consolidation, *source* represents the set of host machines from which migrations will take place.
target	In the VM Placement policies, as well as policies performing migrations such as VM Relocation and VM Consolidation, *target* represents the set of host machines which will receive VMs.
m	The slope of the line of best fit of Data Centre Utilization. This represents the rate of increase or decrease of utilization across the data centre and is used in the determination of whether to switch strategies in the Util-DSS meta strategy
m_{SLA}	Threshold bounding *m* above which a switch to the SLA Strategy should be made if the currently active strategy is the Power

	Strategy
m_{Power}	Threshold bounding m below which a switch to the Power Strategy should be made if the currently active strategy is the SLA Strategy
n	Number of past measurements of Data Centre Utilization that are considered by the Util-DSS meta-strategy in the calculation of linear regression. In these experiments, n was set to 6
workload patterns	The pattern of resource requirements over time that the simulated data centre is under. This is determined by the sum of the workloads of individual VMs, as determined by their traces.
p	Used when evaluating strategies, p represents the total power efficiency of the data centre over the course of the simulation
s	Used when evaluating strategies, s represents the total percentage of resources that were not provided to VMs over the course of the simulation
s_{SLA}	The observed s value after execution of a simulation using the SLA Strategy
s_{Power}	The observed s value after execution of a simulation using the Power Strategy
s_{best}	The better (smaller) value of either s_{SLA} or s_{Power}
s_{worst}	The worse (larger) value of either s_{SLA} or s_{Power}
s_i	The observed s value after execution of a simulation using a given candidate strategy, i
s_{norm}	The s_i value when normalized between s_{best} and s_{worst}
p_{SLA}	The observed p value after execution of a simulation using the SLA Strategy
p_{Power}	The observed p value after execution of a simulation using the Power Strategy
p_{best}	The better (larger) value of either p_{SLA} or p_{Power}
p_{worst}	The worse (smaller) value of either p_{SLA} or p_{Power}
p_i	The observed p value after execution of a simulation using a given candidate strategy, i

p_{norm}	The p_i value when normalized between p_{best} and p_{worst}
v_i	The vector representing the performance of a candidate strategy, i, calculated as (s_{norm}, p_{norm})
$score_i$	The L2-norm of vector v_i calculated as vi This value represents the combined performance of a strategy, relative to the Power and SLA strategy. Strategies with a lower *score* value are considered superior to those with a higher *score* value.

Curriculum Vitae

Name:	Graham Foster
Post-secondary Education and Degrees:	Queen's University Kingston, Ontario, Canada 2007-2011 B.CMP. H. Cog. Sci. The University of Western Ontario London, Ontario, Canada 2011 - Present M.Sc.
Honours and Awards:	Queen's University Entrance Scholarship 2007
Related Work Experience	Teaching Assistant The University of Western Ontario 2011-2012

Publications:

G. Foster, G. Keller, M. Tighe, H. Lutfiyya. M. Bauer, "The Right Tool for the Job: Switching Data Centre Management Strategies at Runtime," IFIP/IEEE International Symposium on Integrated Network Management, 2013.