A Distributed Approach to Dynamic VM Management

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

***Abstract*—Computing today is increasingly moving into large-scale virtualized data centres, offering computing resources in the form of virtual machines (VMs) on a pay-per-usage basis. In order to minimize costs, VMs should be consolidated on as few physical machines (PMs) as possible, switching idle PMs into a power saving mode. It may be necessary to dynamically allocate and reallocate VMs to PMs in order to meet highly dynamic VM resource requirements. The problem of assigning VMs to PMs is known to be NP-Hard. Most solutions focus on a centralized approach, with a single management node making allocation decisions periodically. This approach suffers from poor scalability and the existence of a single point of failure. We present a fully distributed approach to dynamic VM management, and evaluate our approach using a simulation tool. Results indicate that the distributed approach can achieve similar performance to the centralized solution, while eliminating the single point of failure and reducing the network bandwidth required for management.**

I. Introduction

Computing today is increasingly moving into large-scale data centres, offering computing resources on a pay-per-usage basis. An Infrastructure as a Service (Iaas) Cloud allows clients to provision resources with low-level access, on demand. In order for the IaaS cloud provider to minimize costs, it should ensure that its data centre resources are being highly utilized. This allows for both the hosting of additional client workloads, as well as for unused resources to be switched into a power saving mode to reduce costs. However, the provider must also meet the resource requirements of hosted client applications.

Through the use of virtualization, multiple virtual machines (VMs) can run on a single physical machine (host). Since application resource requirements are highly dynamic, a static allocation of resources can still lead to significant resource underutilization. To address this, resources such as CPU can be *oversubscribed*, promising more CPU to a set of VMs than it actually possesses. This can drastically increase the utilization of individual hosts, however, it can also lead to resource contention, and thus to VM performance degradation. A dynamic approach to VM allocation is therefore required. This can be achieved through the use of VM live migration, a process by which a running VM may be migrated (moved) from one host to another with minimal downtime. The problem of finding optimal VM to host allocations is known to be NP-hard [1]. Most work describing dynamic management approaches for large-scale systems makes use of first-fit heuristics [2] [3] [4] to periodically calculate new VM allocations in response to dynamic resource requirements. Existing solutions focus on a centralized architecture, requiring global knowledge of the data centre state. However, given the large scale and highly dynamic nature of the problem, a centralized solution is unlikely to scale to meet realistic demands [5].

We propose a distributed adaptation of a centralized method, using a first-fit heuristic algorithm [6] [3]. The goals of this approach are to achieve similar performance compared to the centralized approach in terms of SLA and power consumption, while spreading management computation across all hosts, and reducing bandwidth usage for management. We evaluate our approach using the DCSim [7] simulation tool.

The remainder of this paper is organized as follows: Section II presents an overview of related work. Section III introduces Dynamic VM Management, and presents a centralized algorithm which we use as the starting point for our work. Section IV presents our Distributed Dynamic VM Management algorithm. The algorithm is evaluated in Section V, and we conclude and discuss future work in Section VI.

II. Related Work

The majority of work in the area focuses on a centralized approach to dynamic VM management, especially work targeted specifically at a data centre providing an IaaS cloud [3] [4] [8]. Several distributed approaches to dynamic resource management in the cloud have been recently proposed. Yazir et al. [9] propose distributing migration decisions to each host, but require hosts to have global knowledge of the state of all hosts, require a performance model of the application, and do not consider SLA performance. Other methods do not consider dynamic resource requirements [5], or still make use of centralized algorithms which can lead to performance problems [10].

Wuhib, Stadler and Spreitzer [11] propose a novel approach to distributed load-balancing in the cloud using a gossip protocol. The work was extended in [12] to consolidate workload. The proposed solutions make use of a demand profiler to estimate resource requirements of modules, as well as control over load-balancing and the starting/stopping of module instances. The target environment is a Platform as a Service (PaaS) cloud, although the authors claim that the approach could be adapted to manage an IaaS cloud. In

contrast, our approach directly targets an IaaS environment, and does not require a demand profiler or control over load-balancing and module instances.

Our work differs from the current literature in that we propose a distributed, decentralized approach to dynamic VM management in an IaaS environment. Several works propose distributed approaches to management in other cloud environments, which do not necessarily translate to IaaS. Other work does not consider dynamic VM resource requirements, minimizing SLA violations or minimizing power consumption. Our approach considers all of these aspects to the problem.

III. Dynamic VM Management

We define *Dynamic Virtual Machine Management* as the dynamic allocation and re-allocation of VMs within a data centre in response to highly variable workloads and VM resource requirements. This is done by a combination of both the intelligent initial placement and allocation of VMs within the data centre, as well as the use of VM live migration to adapt VM allocations to new conditions. The primary goals are to consolidate the set of VMs onto as few physical hosts as possible, while still providing the resources each VM requires to perform up to client expectations. These goals can be expressed as: 1) *Minimize SLA violation*; 2) *Minimize power consumption*.

We define *SLA Violation* as the percentage of incoming requests to hosted client applications that are not completed due to VM resource under-provisioning. Since we are dealing with CPU oversubscription, resource under-provisioning is a result of CPU contention. We assume applications to be running interactive, request-response workloads. When a VM does not have enough resources to meet the current rate of incoming requests to the application running within it, then requests that cannot be processed immediately are dropped. The SLA Violation Percentage, which we refer to simply as *SLA Violation*, is then calculated as the percentage of incoming requests dropped and therefore not completed.

It is also important to minimize the impact of management on the operation of the data centre. Towards this objective, we include secondary goals: 1) *Minimize the number of migrations*; 2) *Minimize management bandwidth usage*.

The minimization of migrations is important to reduce bandwidth consumed by migrations and the performance impact of migration on VMs. We use this set of goals to evaluate the performance of dynamic VM management methods.

Throughout our dynamic VM management algorithms, we quantify CPU resources as *CPU units*, where 1 CPU unit is equivalent to 1MHz processor speed (e.g., a 2.5GHz processor has 2500 CPU units). The term *CPU Utilization* refers to the value CPU_{inUse}/CPU_{total} for a single host.

A. A Centralized Approach

We now present an existing centralized approach employing a first-fit heuristic algorithm [6] [3], which we use as a base for developing and evaluating the distributed solution. There are a number of variations on first-fit algorithms, which provide varying results in terms of SLA, power, and number of migrations [3]. We choose a well-balanced variation which provides reasonable performance in both SLA violations and power consumption [6]. Furthermore, it has been shown in previous work that this approach outperforms static allocation of VMs [3]. The method considers dynamic CPU utilization, and uses static memory requirements as a constraint on VM allocation. Each host periodically (every 5 minutes) sends its state to the central manager, which performs the following operations:

1) VM Relocation: The VM Relocation operation is responsible for relieving hosts that have become stressed (CPU utilization exceeding an upper threshold) by migrating VMs away to other hosts. Stressed hosts are considered *sources* and are sorted in decreasing order by CPU utilization. The remaining hosts are considered *targets* and are sorted by CPU utilization and power efficiency. VMs on stressed hosts are also sorted by CPU use. A first-fit heuristic is then used to select a VM to be migrated from each stressed host, and a target host to which to migrate. Once a VM and target have been identified, a migration is triggered.

2) VM Consolidation: The VM Consolidation operation consolidates VMs on as few hosts as possible by migrating VMs away from under-utilized hosts (and suspending or powering them off) and into partially-utilized hosts. The algorithm is similar to the *VM Relocation* algorithm, except that under-utilized hosts are used as sources and non-stressed hosts are used as potential targets. Sources, targets, and VMs are again sorted, and migrations are determined using a first-fit algorithm.

3) VM Placement: The VM Placement operation runs each time a new VM creation request is received by the data centre, and selects a host in which to instantiate the VM. The algorithm works similarily to VM Relocation.

B. Periodic versus Reactive

The VM Relocation and Consolidation operations are typically triggered on a regular periodic interval. We refer to this method of triggering VM Relocation and VM Consolidation as *Periodic VM Management*. Varying the length of these intervals can affect the performance of the algorithm. We trigger VM Relocation every 10 minutes, and VM Consolidation every hour, based on previous work [6]. VM Relocation can also be triggered in a reactive fashion rather than periodic. We implement *Reactive VM Management* by checking a host for stress every time host state messages are received, and by triggering VM Relocation immediately upon stress detection.

IV. Distributed VM Management

We now present a distributed adaptation of the centralized dynamic VM management approach described in Section III. The goal of developing a distributed approach is to eliminate the requirement of a single, central manager, and to reduce the network bandwidth required for VM management. Furthermore, VM management should be done continuously, rather than on scheduled intervals, to spread migration overhead over

time rather than trigger large bursts of migrations. Decision making is moved into individual hosts, which communicate with each other asynchronously with small messages. An Autonomic Manager within each host performs monitoring, checks for stress or under-utilization situations, and triggers VM Relocation or VM Consolidation operations as required. New VMs can be placed within the data centre by triggering the VM Placement operation on any (powered on) host. Each host is either in the `active` state, in which it is actively hosting VMs, or it is in the `inactive` state, in which it is in a power-saving mode such as `suspended` or `off`. Management operations are initiated with a lightweight *broadcast* message, and each host makes a decision as to whether or not it should participate in the action. The action is then completed with only the set of participating hosts.

The distributed VM management system consists of the following operations:

1) Monitoring: Hosts monitor their resource utilization on a periodic interval, every 5 minutes, and performs a check for stress or under-utilization by comparing average CPU utilization over the last 5 monitoring intervals against threshold values. A host is considered *stressed* when its CPU utilization exceeds an upper threshold, and *under-utilized* when it falls below a lower threshold. These situations trigger VM Relocation and VM Consolidation, respectively.

2) VM Relocation: When a host is *stressed*, it must relocate one of its VMs to another host to relieve the situation. We refer to this process as *eviction*. Each host performs VM relocation itself, first by locating potential target hosts for VM migration, and then by selecting a VM to migrate and a specific target. A stressed host determines the minimum amount of CPU required to be available on another host to evict one of its VMs and relieve the stress situation, and broadcasts a *Resource Request* message to all other hosts. We conserve bandwidth by sending CPU requirements *only*, as it is a good indication of overall load, and is highly contentious. Each host determines if it can accommodate the minimum request, and if so, responds with a *Resource Offer*. After a specified time period, the evicting host selects a VM and a target host using a first-fit heuristic algorithm, similar to the algorithm in the centralized approach (Section III-A1), and performs the migration. If no hosts respond to the *Resource Request*, then the `evicting` host must boot an `inactive` host (`suspended` or `off`), if available. Each host maintains a list of `inactive` hosts, for this purpose. In the event that more than one host triggers a VM Relocation operation simultaneously, it may be the case that both perform their operation with a subset of hosts. In a large data centre, the effect of this is likely to be negligible. Furthermore, if a host fails to evict a VM, and remains stressed, it will retry eviction shortly afterwards.

In order to reduce thrashing between highly utilized hosts, we implement an *relocation freeze*, preventing a host from offering resources for a specified amount of time after the same host evicts a VM. Similarly, if a host offers resources and is chosen as the target, the *relocation freeze* prevents it from evicting a VM for a specified time period. This mechanism helps to reduce unnecessary migrations, and tuning the *relocation freeze* time parameter enables trading a lower migration count for increased SLA violations.

3) VM Consolidation: When a host is *under-utilized*, it is desirable to migrate its VMs to other hosts and shut it down. However, there may be other hosts which are better candidates for shutting down. For example, selecting the host with the lowest utilization increases the probability that a host will successfully shut down. We therefore introduce a *Shutdown Selection* process to select the most appropriate host for shut down.

When a host detects that it is *under-utilized*, it triggers a *Shutdown Selection* operation and broadcasts its intent to shutdown and current CPU utilization to all hosts. Any host that has a lower CPU utilization or worse power efficiency responds, indicating that it is a candidate for shutting down. The original, triggering host selects the candidate host (which includes itself) first by power efficiency, and then by CPU utilization (favouring lower power efficiency and lower CPU utilization). This host is selected as the *winner*, and all hosts are notified of the outcome. The *winner* host then attempts to find migration targets for *all* of its hosted VMs by sending a *Resource Request* message and collecting *Resource Offers*, as in the *eviction* process. It differs, however, in that it will only perform migrations if it finds target hosts for *every* hosted VM. Otherwise, the shutdown is cancelled, as migrating hosts without shutting down will only serve to increase the likelihood of target hosts becoming stressed without gaining any reduction in power consumption. If VMs are successfully migrated, the host powers off.

In order to control the frequency of host shut downs, we add a *shutdown freeze* time during which no host can attempt shut down after a Consolidation operation has taken place, similar to the *relocation freeze*. Controlling the *shutdown freeze* time has the effect of trading power consumption for SLA performance.

4) VM Placement: The VM Placement operation is triggered whenever a new VM arrives at the data centre and must be placed on a host. It is performed in the same manner as VM Relocation, except that the result is a new VM instantiation rather than a migration. Any host can perform the VM Placement operation; no central placement controller is required.

One of our goals in developing a distributed dynamic VM management system was to reduce the amount of bandwidth used for VM management. The centralized system transmits host state data from each host on a regular interval, regardless of whether or not that data is required at the time. Each message contains the resource utilization of each hosted VM (vector $(cpu, memory, bandwidth, storage)$). We assume each individual resource value to be 4 bytes in size, combining for a total of 16 bytes. The distributed system attempts to send data only when required. Many messages contain no payload data, while others contain only a single value, such as the *Resource Request* and *Shutdown Selection* messages. Messages that

contain full resource data contain only a *single* resource vector. In the case of *broadcast* messages, we total each time the message is received, rather than simply considering it as equivalent to a single message.

V. Experiments

In this section, we will evaluate our distributed approach through conducting a set of experiments using an open source simulation tool, DCSim [7] [13]. We compare the distributed approach with two forms of centralized management, namely, *periodic* and *reactive*. We report the following metrics:

Number of Migrations: The number of migrations triggered during the simulation.

Power Consumption: Power consumption is calculated for each host, and the total kilowatt-hours consumed during the simulation are reported.

SLA Violation (SLA V): SLA Violation percentage as defined in Section III. We also include a migration overhead penalty of 10%, applied to VMs during migration [14].

SLA Violation Duration (SLA Duration): The total amount of time that all VMs spent in a state of SLA violation.

Management Bandwidth Usage: The total amount of bandwidth used for VM management messages. Message sizes are calculated as defined in Section IV.

A. Setup

Our simulated data centre consists of 200 host machines, of two types: 4 cores/8 GB RAM, and 8 core/16GB RAM. We define three different VM sizes: 1 vcore (virtual core) with 1500 CPU shares/512MB memory; 1 vcore with 2500 CPU shares/512MB memory; and 2 vcores with 2500 CPU shares each/1GB memory. Hosts are modelled to use a work-conserving CPU scheduler, as available in major virtualization technologies. VMs are assigned CPU shares in a fair-share manner, and memory is statically allocated. We model a set of interactive applications running within the data centre, with each VM running a single application. VMs arrive and depart the system throughout the experiment, and exhibit dynamic resource requirements driven by real workload traces (the *ClarkNet*, *EPA*, and *SDSC* traces [15], and two job types from the *Google Cluster Data* trace [16]). We compute a normalized rate of requests from the traces, with the CPU resource requirements of VMs calculated as a linear function of the current rate. VMs arrive at a changing rate, and terminate after about 1 day. The total number of VMs in the system varies daily, using randomly chosen values uniformly distributed between 600 and 1600. All results are averaged over 10 experiments.

B. Evaluation

We now compare the performance of distributed VM management with that of the centralized approach. The operation and performance of the distributed VM management system can be fine-tuned through the manipulation of the *relocation freeze* and *shutdown freeze* durations discussed in Section IV. We evaluated six combinations of tuning parameters, and have chosen two configurations for comparison. The complete results of the tuning parameter evaluation have been omitted due to space constraints. We compare the centralized algorithms against a distributed configuration with very good SLA performance (*Dist. SLA*) and one with very good power performance (*Dist. Pwr*). We compare against the *Periodic* and *Reactive* versions of the centralized system, as defined in Section III-B. Table I presents the average results over 10 simulations.

	Migs	Power	SLA V.	SLA Dur.	Man BW
Dist. Pwr	11524	5072kWh	0.113%	25.87 days	10.8GB
Dist. SLA	9091	5352kWh	0.043%	11.36 days	11.0GB
Periodic	10261	5056kWh	0.109%	26.58 days	38.0GB
Reactive	12508	5121kWh	0.059%	15.42 days	38.1GB

TABLE I
Distributed vs Centralized Results

The *Reactive* centralized management system provides improved SLA performance when compared to *Periodic*, at the expense of power and migrations. This is due to the fact that it responds immediately to stress situations, triggering VM Relocation as soon as one is detected. *Distributed Power* achieves similar power consumption and SLA violation performance to *Periodic*, at the cost of a slight increase in migrations. It is expected that there should be a trade-off involved in using a distributed version, as it uses only partial knowledge to perform management operations. *Distributed SLA*, on the other hand, achieves reduced SLA violations when compared to all other systems, as well as a greatly reduced number of migrations. It does so at the expense of an increase in power consumption though. Both distributed versions offer about a 71% reduction in management bandwidth usage, falling in line with our goals for distributed VM management.

VI. Conclusions and Future Work

We present a distributed VM management system, adapted from an existing centralized system. The goals of the approach were to replicate the SLA and power performance of the centralized system, while eliminating the central manager and reducing the bandwidth consumed by management. Through evaluation with a simulation tool, we have shown that the distributed approach can in fact achieve these goals, although there is always a trade-off to be made between number of migrations, SLA violation and power. Furthermore, the distributed system can be tuned to favour SLA or power, to suit the requirements of the data centre operator. In all cases, the distributed system provided a reduction in management bandwidth usage. To our knowledge, this is one of the first works to present an entirely distributed solution to this problem.

There are a number of possible directions for future work. Most importantly, broadcast messaging could be replaced by a different communication method or overlay network to attempt to further reduce management bandwidth usage. Hosts could be split into smaller multicast groups, with a new mechanism introduced to communicate between them only when necessary. Furthermore, consideration of the topology of the data centre may drive the design of algorithms as well as impose constraints on VM placement.

REFERENCES

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

[2] 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.

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

[4] 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.

[5] E. Feller, C. Morin, and A. Esnault, "A case for fully decentralized dynamic vm consolidation in clouds," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*. IEEE, 2012, pp. 26–33.

[6] G. Foster, G. Keller, M. Tighe, H. Lutfiyya, and M. Bauer, "The Right Tool for the Job: Switching data centre management strategies at runtime," in *Integrated Network Management (IM), 2013 IFIP/IEEE International Symposium on*, May 2013.

[7] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, "Towards an improved data centre simulation with DCSim," in *SVM Proceedings, 7th Int. DMTF Academic Alliance Workshop on*, Oct. 2013.

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

[9] Y. O. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, and Y. Coady, "Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. Ieee, 2010, pp. 91–98.

[10] F. Quesnel, A. Lèbre, and M. Südholt, "Cooperative and reactive scheduling in large-scale virtualized platforms with dvms," *Concurrency and Computation: Practice and Experience*, 2012.

[11] F. Wuhib, R. Stadler, and M. Spreitzer, "Gossip-based resource management for cloud environments," in *Network and Service Management (CNSM), 2010 International Conference on*. IEEE, 2010, pp. 1–8.

[12] R. Yanggratoke, F. Wuhib, and R. Stadler, "Gossip-based resource allocation for green computing in large clouds," in *Network and Service Management (CNSM), 2011 7th Int. Conf. on*, Oct. 2011.

[13] DCSim on GitHub. [Online]. Available: https://github.com/digs-uwo/dcsim

[14] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency Computat.: Pract. Exper.*, pp. 1–24, 2011.

[15] (2013, May) The internet traffic archive. [Online]. Available: http://ita.ee.lbl.gov/

[16] (2013, May) Google cluster data. Google Inc. [Online]. Available: http://code.google.com/p/googleclusterdata/