NORTHEASTERN UNIVERSITY
Graduate School of Engineering

Dissertation Title: Using Live Virtual Machine migration to Improve Resource Efficiency in Virtualized Data Centers

Author: Emmanuel Arzuaga
Department: Electrical and Computer Engineering

Approved for Dissertation Requirements of the Doctor of Philosophy Degree:

Dissertation Adviser: Prof. David Kaeli

Dissertation Reader: Prof. Peter Desnoyers

Dissertation Reader: Prof. Ningfang Mi

Department Chair: Prof. Ali Abur

Graduate School Notified of Acceptance:

Director of the Graduate School: Dr. Sara Wadia-Fascetti
USING LIVE VIRTUAL MACHINE MIGRATION TO
IMPROVE RESOURCE EFFICIENCY IN VIRTUALIZED
DATA CENTERS

A Dissertation Presented
by
Emmanuel Arzuaga
to
The Department of Electrical and Computer Engineering
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the field of
Computer Engineering
Northeastern University
Boston, Massachusetts
January 2012
Abstract

Efficient resource management is still an open problem in data centers. Although there has been continued improvement in the performance of such large scale systems, the improvements have been mostly at the expense of adding more servers to the system; increasing space requirements and power consumption sometimes without making an effort to better utilize available resources. Modern data centers are growing at a such high pace that space and power consumption are becoming limiting factors. Virtualization has the potential to address these limitations by increasing resource efficiency throughout the data center. Because hardware resources are being shared by different virtual machines (VM), it is important that the placement of a VM in a physical host does not degrade the performance the overall system. In this thesis, we are focused analyzing the benefits that live VM migration can have on virtualized data centers. Specifically, it is our goal to develop a robust VM migration framework that can be used to improve resource efficiency throughout the data center. In this thesis we propose a metric that we can accurately quantify the load of a virtualized enterprise server. We demonstrate how this metric can be used to load balance a entire system. We also consider extension to our framework to consider reducing power consumption.
Acknowledgements

First, I would like to thank my advisor, Prof. David Kaeli, for all his help, advice, and support throughout this process. I would also like to thank all of my friends and colleagues from the Department of Electrical and Computer Engineering at Northeastern University, specially to the Northeastern University Computer Architecture Research Group (NUCAR) from whom I have learned so many things and made my time there very pleasant. Finally but no less, I would like to thank my family for their unconditional love, support, patience, and understanding while I have been pursuing my graduate degrees.
# Contents

Abstract iii

## 1 Introduction

1.1 Modern Data Centers ........................................ 7
1.2 Load Balancing in Data Centers ............................... 8
   1.2.1 Preemptive Load Balancing: Process Migration ........... 11
1.3 Virtualization in Data Centers ............................... 15
   1.3.1 Virtualization and Load Balancing ....................... 17
1.4 Power Consumption in Data Centers ........................... 18
1.5 Scope and Contributions of This Thesis ...................... 20
1.6 Organization of This Thesis ................................ 21

## 2 Related Work

2.1 Server Virtualization .......................................... 23
2.2 VM Migration .................................................... 26
2.3 Dynamic Resource Management ................................ 30
2.4 Energy Savings .................................................. 32
   2.4.1 Physical Servers ......................................... 32
3 The Dynamic Resource Efficiency Manager

3.1 Quantifying The Virtualized Server Load
   3.1.1 Desirable Properties of a Virtualized Server Load Metric
   3.1.2 VM Resource Efficiency
   3.1.3 Measuring the Load of a Physical Host

3.2 DREM Implementation: Physical Testbed

3.3 Benefits of the Dynamic Resource Efficiency Manager

4 Using DREM for Load Balancing

4.1 Load Imbalance Metric based on VSL

4.2 VSL Inductive Balancing Method
   4.2.1 Building the Algorithm

4.3 Experiments

4.4 DREM Results
   4.4.1 \( I_{Metric} \) Analysis
   4.4.2 DREM-DRS Comparison

4.5 Looking further: \( I_{Metric} \) and Power

4.6 Summary

5 Using DREM to Reduce Power Consumption

5.1 Power Savings Metric based on VSL

5.2 VSL Power Savings Method
   5.2.1 Issuing Migrations

5.3 DREM Behavior under Different Workloads
   5.3.1 Workload Characteristics
List of Figures

1.1 Enterprise server application assignments in a data center. (a) Traditional one-to-one assignment. (b) Server consolidation produces a many-to-one assignment. ..................................................... 3
1.2 In a virtualized server system, one or more VMs run in one physical server. .......................................................... 6
1.3 Components of a modern data center. .......................................................... 8
1.4 Process migration flow. ........................................................................ 11
1.5 Total-Freezing process migration scenario. The process is halted while all the state is transferred from the source host to the target host. ... 12
1.6 Pre-Copy process migration scenario. The process continues to execute on the source while the address space is being transferred from the source to the target. ..................................................... 13
1.7 Copy-On-Reference process migration scenario. The process is quickly transferred from the source host to the target host. The state is transferred when requested while executing on the target node. ........ 14
1.8 Live VM migration process. ................................................................. 17
1.9 Projected electricity use for US data centers. Source: 2007 EPA congressional report on server and data center energy efficiency. .......... 19
3.1 The Dynamic Resource Efficiency Manager communicates with the data center virtualization manager to collect system state information. It then issues VM migrations to meet certain system objectives such as load balancing.

3.2 Virtualized Data Center. The system is composed of a set of 6 Virtualized Enterprise Servers (top) sharing a Storage Sub-system (bottom) and two servers for managing the system (middle): the Virtualization Manager Server and the Dynamic Resource Efficiency Manager server. The Virtualized Enterprise Servers are connected to a Watts up? PRO ES power meter that reports power consumption measurements to the DREM server through a USB interface. The Virtualization Manager and the Dynamic Resource Efficiency Manager collaborate together to handle VM migrations across the Virtualized Enterprise Servers.

3.3 Experimental testbed of the Virtualized Data Center. The data center contains 18 VMs running on 6 Virtualized Enterprise Servers sharing two iSCSI LUNs. The DREM service attaches to VMware Virtual Center Server to collect state information and handle VM migrations across the Virtualized Enterprise Servers using the load balancing or the power savings modules.

3.4 VM sizes and their virtual components.

4.1 $C_L$ as a function of throughput in Transactions-per-Minute (TpM). For lower $C_L$ values there are higher TpM values.
4.2 Experimental setup for the load balancing problem, (a) DREM will issue VM migrations following DREM’s VIBM algorithm and VMware’s DRS recommendations; VM initial placements for the different experiments and number of servers used, (b) 6 VM and (c) 8VM experiments use 2 servers, (d) 10 VM experiments use 3 servers, (e) 16 VM experiments use 5 servers, and (f) 18 VM experiments use 6 servers.

4.3 Accuracy of $I_{Metric}$ prediction for a set of DREM migrations.

4.4 Comparing the accuracy of $I_{Metric}$ prediction following different migration recommendations: DRS (a) and DREM (b). Notice how DREM selects the migrations that better reduce $I_{Metric}$.

4.5 A comparison of $I_{Metric}$ values over time per used method for: (a) 6 VM, (b) 8 VM, (c) 10 VM, (d) 16 VM, and (e) 18 VM experiments. DREM obtains the lowest average $I_{Metric}$ for all experiments.

4.6 OLTP Transaction Throughput (TpM) for all VM configurations. DREM migrations produce a higher TpM in the two experiments.

4.7 Power vs Performance in OLTP Transaction Throughput vs total server consumed power (W/TpM) for all VM configurations. DREM migrations produce a better W/TpM ratio in all experiments.

4.8 A comparison of total server power consumption over time per used method for: (a) 6 VM, (b) 8 VM, (c) 10 VM, (d) 16 VM and (e) 18 VM experiments. There is no clear form to select which method provides better power-performance efficiency.

4.9 A comparison of $I_{Metric}$ as a function of the total server power consumption over time per used method for: (a) 6 VM, (b) 8 VM, (c) 10 VM, (d) 16 VM and (e) 18 VM experiments.
5.1 $\mu_L$ as a function of power in Watts (W). For lower $\mu_L$ values there are lower power consumption values. ................. 70

5.2 Experimental setup for the power savings, (a) DREM will issue VM migrations following DREM’s VPSM model; VM initial placement for the different experiments and number of servers used, (b) 6 VM, 8VM and 10 VM experiments use 3 servers and (c) 14 VM, 16 VM and 18 VM experiments use 6 servers. ....................... 74

5.3 A comparison of measured power consumption over time per workload for: (a) 6 VM, (b) 8 VM, (c) 10 VM, (d) 14 VM, (e) 16 VM , and (f) 18 VM experiments. DREM is able to reduce consumed power for all experiments. ............................................. 76

5.4 3 and 6 Servers energy savings for all three workloads. .............. 77

5.5 A comparison of achieved performance, total server power consumption and energy efficiency over time per used workload for the 3 servers configuration: (a) Apache, (b) Email, and (c) OLTP workloads. Note how the performance degradation (left) and the power savings (middle) results align to produce a net scaling back effect that yields comparable energy efficiency values (right). ................................. 78

5.6 A comparison of achieved performance, total server power consumption and energy efficiency over time per used workload for the 3 servers configuration: (a) Apache, (b) Email, and (c) OLTP workloads. Note how the scaling back effect is maintained in this larger working set. . 79
6.1 Experimental setup for the analysis of VSL and related metrics, (a) DREM will issue VM migrations following DREM’s VPSM model; VM initial placement for the different experiments and number of servers used, (b) 10 VM experiments use 3 servers and 18 VM experiments use 6 servers. ................................. 84

6.2 3 Servers 10VM Apache workload metric analysis: (a) $I_{Metric}$ and its components, (b) $E_{Metric}$ vs power, and (c) $I_{Metric}$ vs power. ................................. 85

6.3 6 Servers 18VM Apache workload metric analysis: (a) $I_{Metric}$ and its components, (b) $E_{Metric}$ vs power, and (c) $I_{Metric}$ vs power. ................................. 86

6.4 3 Servers 10VM Email workload metric analysis: (a) $I_{Metric}$ and its components, (b) $E_{Metric}$ vs power, and (c) $I_{Metric}$ vs power. ................................. 88

6.5 6 Servers 18VM Email workload metric analysis: (a) $I_{Metric}$ and its components, (b) $E_{Metric}$ vs power, and (c) $I_{Metric}$ vs power. ................................. 89

6.6 3 Servers 10VM OLTP workload metric analysis: (a) $I_{Metric}$ and its components, (b) $E_{Metric}$ vs power, and (c) $I_{Metric}$ vs power. ................................. 90

6.7 6 Servers 18VM OLTP workload metric analysis: (a) $I_{Metric}$ and its components, (b) $E_{Metric}$ vs power, and (c) $I_{Metric}$ vs power. ................................. 92

6.8 6 Server 18VM VSL values: (a) Apache workload VSL used in this thesis, (b) Apache workload VSL with storage component, (c) Email workload VSL used in this thesis, (d) Email workload VSL with storage component, (e) OLTP workload VSL used in this thesis, and (f) OLTP workload VSL with storage component. ................................. 95
List of Tables

4.1 OLTP Transactions and I/O pattern. ........................................ 54
4.2 VM name and type. VM Small configurations have id 1 thru 7 while
VM Tiny types run from 8 to 18. ............................................. 56
4.3 VM OLTP workload description. ........................................... 60
4.4 $I_{Metric}$ and throughput relation for the different VM configurations.
The average $I_{Metric}$ value is presented along with the average OLTP
throughput in transactions per minute (TpM). A lower value of $I_{Metric}$
yields an increase in transaction throughput. ............................... 60

5.1 VM name and type. VM Small configurations have id 1 thru 9 while
VM Tiny types run from 10 to 18. ........................................... 73
Chapter 1

Introduction

Over the past few decades, advances in computing technology have revolutionized our lifestyle. The Internet opened the door to online services that enable us to communicate with people around the world, share information and provide access to a myriad of applications. For example, current online services range from search engines, online stores (goods, books, auctions), bill paying and banking services, access to government services, and more recently social networks have made a huge impact on how we interact with colleagues, friends and family. All of these services have diversified the type of workloads that run in data centers. Data center systems utilize a range of resources (including CPU, memory, storage and network), which can differ greatly depending in the type of workload being serviced. Even though any workload will use CPU and memory resources, the performance of selected applications may be limited by other resources. For example, streaming applications tend to be network intensive, where performance is only limited by the available network bandwidth. Many database applications are storage intensive applications that consume large amounts of disk space to manage data sets. For these applications, performance
CHAPTER 1. INTRODUCTION

is limited by the rate that data can be read or written to the storage media per unit time. Each application has its own particular needs for resources in order to execute efficiently. It is important for the data center to adaptively service all needs without squandering resources. Data centers provision computing resources to guarantee certain performance levels specified in service-level agreements (SLAs) when resources are constrained. However, most of the time the system is running at much lower load levels, with some components even sitting idle. This can waste computational resources, can lead to underutilized disk space, produce higher power budgets, resulting in inefficient system utility. We need to increase system resource efficiency (i.e., reducing the cost of ownership of large systems without degrading performance) to satisfy the demands for current and future online services.

Advances in microelectronic technology have resulted in the availability of low-cost, highly efficient, server systems. Typically a data center consists of a large pool of servers networked together, replacing high-end mainframes. A good example of this model is the Google Cluster [1]. This change in design provides comparable application performance, while significantly reducing the total cost of ownership of the system. However, changing the hardware systems does not eliminate the cost associated with underutilized resources. Given the added heterogeneity of these systems, the tasks of load balancing and resource management become increasingly challenging.

A commonly used strategy to conserve system resources in data centers is known as server consolidation [2, 3, 4]. An example is shown in Figure 1.1. Usually, data centers perform a one-to-one match between the operating system (OS) and applications running on server nodes (Figure 1.1(a)). Server consolidation, on the other
Figure 1.1: Enterprise server application assignments in a data center. (a) Traditional one-to-one assignment. (b) Server consolidation produces a many-to-one assignment.
CHAPTER 1. INTRODUCTION

hand, attempts to perform a many-to-one application-to-operating system assignment; multiple applications run on one server (Figure 1.1(b)). The main goal of server consolidation is to increase efficiency by maintaining good resource utilization. However, server consolidation can many times lead to performance degradation and application security violations due to applications contending for shared resources. Since all of the applications will be running on the same server and operating system, this approach also raises issues in terms of fault tolerance.

A common methodology that data centers use for effectively sharing resources is load balancing [5, 6, 4, 7]. In traditional load balancing schemes, the system is analyzed at the process or job level. Load balancing attempts to distribute all available resources to all jobs running in the system. The load balancing layer attempts to provide a solution to the job placement or assignment problem such that all resources are being utilized. Given that most server nodes connected by a network, load balancing techniques can employ remote job execution in order to improve resource utilization and relieve system hotspots. While there exist several accepted load balancing algorithms, any approach requires accurate estimation of the system load.

Some load balancing algorithms use global system load metrics, while others are based on per-node system state information. The use of global system parameters allows an algorithm to be fast and straightforward to implement. However, this class of balancing algorithms lacks essential information about workload variability and may not be able to work effectively in a heterogeneous environment. Per-node state measurements require a more complex implementation and may require additional data collection/analysis overhead, but generally yield better performance because they can capture more fine-grained system behavior.
The first step in collecting per-node state information is to decide on the metric to measure and the method to acquire that metric (i.e., the *load metric policy*). The chosen load metric is a critical part of any successful load balancing scheme because it must be measured periodically to capture the state of each node. This metric also needs to be tied to the overall throughput of the system. Since the load on a server captures behavior at a particular instance in time, the load balancing algorithm must decide quickly and accurately on the necessary steps to balance the system. Hence the second step is to select the most appropriate *load distribution policy*. This policy will decide how to distribute the load on the available nodes, and also determine the robustness of the load balancing scheme. For example, a *static or fixed* load distribution policy assigns jobs to nodes using only one placement decision per job. This limits the ability of the system to react quickly to dynamic system behavior. A *dynamic* load distribution policy, on the other hand, can potentially make more than one placement decision per job, and reassign the job to different nodes depending on changes in the overall system load.

Recently, we have witnessed the rapid growth in terms of deployments of *virtualization* in data centers [8, 9, 10]. Virtualization has been proposed to provide application isolation and security. Following this approach, a single physical server runs one or more virtual machines (VMs). The *Virtual Machine Monitor* (VMM), also known as a *hypervisor* is a low-level software layer running in the server. The VMM is in charge of managing the execution of VMs, as well as resource enforcing sharing policies across the physical host. As shown in Figure 1.2, every application running on a server will have its own context; each application is encapsulated in a VM that maintains a separate copy of the OS and state.

Virtualized servers are becoming the preferred path for server consolidation. But
virtualization used in a data center setting can introduce new performance and power challenges. In particular, the VM is an extra layer of software that will run on the server and it cause some degradation in performance if not properly configured. Moreover, multiple VMs running on the same physical hardware may compete for the same resources and negatively impact each other’s performance. Consequently, an appropriate VM placement is a key issue that must be dealt with. Again, this problem is non-trivial due to the variability of the class of workloads being run in typical data center environments. Therefore, a static or fixed VM placement algorithm might not be the best solution for all cases. The VM placement problem can be expressed as a classical load balancing problem. In general, as in the case of load balancing, we need the ability to dynamically move VMs between different physical nodes in the data center. In a virtualized environment, this movement is called VM migration, and is commonly used to tune the system to adapt to a particular workload mix being run
CHAPTER 1. INTRODUCTION

at a particular time. VM migration typically uses a per-node state metric.

In this thesis, we present a new methodology to increase resource efficiency in a data center. We call this methodology the *Dynamic Resource Efficiency Manager (DREM)*. While the resource efficiency manager builds on prior work in this field, it introduces improvements to current techniques. The resource efficiency manager achieves this by focusing primarily on quantifying the load on a virtualized server at a particular time instance, rather than on global system parameters. The resource efficiency manager computes the load of the server system as a function of the VMs currently assigned to it; this focus allows for enhancement of the system efficiency by migrating VMs depending on different efficiency goals, presented in Chapter 3.

In this chapter, we provide an introduction covering some of the background in the area of data center virtualization. This includes a brief overview of data centers and virtualization, investigating improved resource management. Specifically, we focus on issues related to load balancing and power consumption in data centers. We discuss in detail the concepts of process and virtual machine migration, the main issues addressed in this thesis. Finally, we present the major contributions of this thesis, and provide an overview of the remainder of this thesis.

1.1 Modern Data Centers

Modern data centers are complex systems composed of a variety of different subsystems. The typical data center is primarily composed of electronic equipment used for data processing (*computer systems*) and storage (*storage systems*), all interconnected (*network systems*). Data centers may contain some amount of redundancy on some or all of its components (power, storage, servers) for critical applications. Figure 1.3
CHAPTER 1. INTRODUCTION

Figure 1.3: Components of a modern data center.

shows such a system.

Large data centers are comprised of tens of thousands of servers with tens of petabytes of storage, and multiple hundreds of gigabit of network bandwidth. Computing and storage capacities of these data centers are continually increasing. One negative side effect this capacity increase has been a rapid rise in the energy consumption and power density of data centers. Thus, resource efficiency is gaining more importance in data center design.

1.2 Load Balancing in Data Centers

Data centers contain multiple computing and storage resources typically arranged in a physically distributed fashion. All of the system components are interconnected by a network subsystem (see Figure 1.3). The network enables all resources to potentially be available to service a particular service request. Once a request is received, it is important to assign the available resources to service it rather than using already committed resources to service other requests. One approach to accomplish this is the use of sophisticated load balancing algorithms to match service requests to physical resources. The goal of most load balancing algorithms is to distribute the load evenly throughout the data center nodes, resulting in improved resource utilization. The
load balancing algorithm will transfer the load from heavily loaded nodes to lightly loaded ones in an attempt to improve overall performance relative to some system performance metric. Since traditional load balancing algorithms distribute load at a job granularity, the most common system performance metric is job response time.

Designing a proper load balancing algorithm for any data center is not an easy task because of the following challenges:

- **Load metric policy**: Determining how to capture the load of a particular system node is a difficult task. The load metric may contain deterministic parameters (state-based such as the number of processes in a node or resource demands) or non-deterministic parameters (system-based such as architectural characteristics of the node such as processor speed and memory/disk capacity). Selecting a proper load metric given the large number of available parameters is therefore hard and an ongoing research area [5, 11, 12, 13, 14]. Since a load metric that can be easily measured is desirable, deterministic parameters are often preferred. The CPU queue length and utilization are the most commonly used deterministic parameters [5, 7, 12].

- **The prediction of system state**: Predicting system state is a key component of any load balancing system. The selection of a load metric should be done in such a manner that provides accurate measurements of performance-related trends on a per host basis. We need the ability to accurately predict what the performance benefit of a selected assignment will be. This benefit should be computed at a complete system level; we need to make sure that it does not simply benefit a single VM and leave the rest of the system significantly impacted.
• Load distribution policy: The load distribution policy deals with the detection of imbalance and job placement. This can be done in a centralized manner where a manager collects load information and decides the process assignment, or in a distributed manner where each node is in charge of detecting its load status and can request and/or accept a job transfer. We can either assign processes to nodes statically (using scheduling) or dynamically (via process migration). In general, a dynamic approach can yield better system performance and is preferred, though is subject to thrashing if not done carefully. There exist several process migration algorithms. In the next section we discuss a number of these algorithms in more detail.

• Avoidance of migration bounces: A migration bounce occurs if the node load level is detected as capable of accepting jobs, but accepting another job may force it to request migration for the incoming job or another job currently assigned to that node. In general, an accurate prediction of system state should help to anticipate such instances, helping to avoid them.

Load balancing algorithms can be divided in two major groups: non-preemptive (static) and preemptive (dynamic) [6]. Non-preemptive algorithms are also known as distributed job scheduling algorithms and they basically assign a request to a node at the time the request is made. The request will run until completion on the assigned node. Preemptive load balancing algorithms move requests from node to node, and react to changes in the profiled system state. In the event that one node becomes idle, a preemptive load balancing algorithm can redistribute the load and move waiting requests to other nodes that are less heavily loaded. Since these types of algorithms react to changes in system state and can balance load while in the middle of their execution, they have the ability to produce better resource management solutions.
1.2.1 Preemptive Load Balancing: Process Migration

Preemptive load balance algorithms are designed to be aware of dynamic workload behavior. Each system node has a notion of the processes it is servicing. A process can be defined as an instance of a computer program in execution. The information about a process includes the resources that it requires for execution (e.g., allocated memory, registers, data structures and other operating system data) and is referred to as the process state. The necessary information to migrate a process includes the process’ state and address space.

Process migration can be defined as the transfer of a process from its current host (source node) to a target host (destination node). The migration process follows an execution flow similar to the one presented in Figure 1.4. The first step is to select which process to migrate on the source host and quiesce its execution. Then, an essential components migration of the process is performed. Once the essential components are on the destination node, the process can resume execution. After migrating all the process’ components, the migration ends and the process data on
CHAPTER 1. INTRODUCTION

Figure 1.5: Total-Freezing process migration scenario. The process is halted while all the state is transferred from the source host to the target host.

The source node can be discarded. The total migration time can be defined as the essential components migration time plus the post-processing time.

The typical process migration scenarios are:

- **Total-Freezing:** Figure 1.5 depicts this scenario. In this case, the process is completely removed from execution on the source host and then it is transferred to the target host. Once the process has been completely migrated, it resumes execution on the target host. Depending on the address space size of the process, this approach may significantly degrade performance.

- **Pre-Copy:** Figure 1.6 shows how this case works. The process starts to transfer its address space to the target host, but continues to execute on the source node. If there is a dirty reference (e.g., a modified memory address) associated with the current execution, that reference continues to be sent. When the number of dirty
Figure 1.6: Pre-Copy process migration scenario. The process continues to execute on the source while the address space is being transferred from the source to the target. If the number of references is below a predetermined threshold, the process execution is stopped on the source host and the final address space is moved. Execution is then resumed on the target host. This approach does not impact the performance of the process significantly because it hides the associated transfer latency while the process remains running on the source node.

- **Copy-On-Reference:** This strategy is shown in Figure 1.7. This approach transfers essential process state to the target host and resumes execution there. The rest of the process state gets transferred similar to demand paging or copy-on-write caches. Whenever there is a reference to an address that is not on the target, it is requested from the source until all the state is copied to the target. While this scheme provides minimal process execution downtime, it may result in significant performance degradation while the state is being transferred. It
also elongates the time in which the process is executing with residual dependencies.

There exist a large number of process migration mechanisms, but they are mostly variants of these three scenarios. Any successful process migration implementation must have the following characteristics:

- **Transparency**: No process in the system can be capable of detecting the migration. Communication with the migrated process may be delayed but never lost. After migration, the process should be able to communicate through its previously opened I/O channels as if the migration never took place.

- **Low-Impact**: The migration of a process should not significantly degrade the performance of the process or the other processes in the whole system. A typical form of achieving this is to minimize the essential components migration time.
CHAPTER 1. INTRODUCTION

• \textit{Residual Dependencies Free}: There should be no \textit{residual dependency} left on the source node. Once migrated, a process must not depend on state left on the source node. If such dependencies are left, there will be a performance degradation imposed on the source node and if the node fails it might destroy the process. In either case, this violates the low-impact feature.

• \textit{Minimum-cost}: The cost for migration should be kept at a minimum. This includes the time for migrating a process, the time required to identify which process to migrate, and the time to decide where to migrate to.

• \textit{Reliability}: The process migration implementation should be robust enough so that in the event of a failure it can recover with minimal impact on the system.

Developing a process migration mechanism that fully complies with all of the listed features is not a trivial task. Although there has been significant research in this area [15], it remains extremely difficult to enable process migration on UNIX-like operating systems since the kernel is designed as a monolithic centralized system. Working with such a monolithic view, lacks a distributed-system-wide notion that affects the design of a process’s state information and the overall migration methodology.

There have been successful implementations that trade off residual dependencies to obtain full transparency such as MOSIX and Sprite [15]. In these systems, the source node remains in charge of part of the state and the target host forwards any request that relies on the state to the source.

1.3 Virtualization in Data Centers

Virtualization technologies have a rich history, dating back to the 1960s with IBM’s System/360 and 370 [16]. The term \textit{virtualization} is used to describe the abstraction
of the computational resources needed to complete a service request and the underly-
ing physical equipment used to provide the service. A typical example would be the use of virtual memory. In this case, the application that is running on the machine can potentially allocate more memory space than what is physically available. Even though the extra memory needed may be provided by disk or other type of storage, the application just cares about the resources being provided and not on the physical equipment used to provide the extra memory space.

Current commercial virtualization solutions are dominated by VMware’s ESX Server [17], Microsoft’s Hyper-V [18] and Oracle’s xVM [19]. Other popular virtual-ization platforms include Citrix XenSource [20] and Xen; both of these provide open source solutions. There are open source versions of xVM as well.

Server virtualization is currently used to consolidate load [9], enhance the utiliza-
tion of physical systems [21] and provide a level of isolation and security for applications [10]. This has opened the door for the use of VMs in data centers. However, in order to better utilize the resources in these systems, it is necessary to have the ability to dynamically allocate resources to the running VMs. Figure 1.8 shows a VM being migrated from Server 1 to Server 2. The Virtualization Manager is in charge of handling migrations and it selects which VM to migrate based on an imbalance measurement. Notice that while the migration is taking place, the VM is completely functional, hence the term live migration. One of the most common uses of live VM migration is precisely to load balance a virtualized data center. In the following section we discuss in detail load balancing on data centers and the origins of VM migration.
1.3.1 Virtualization and Load Balancing

Virtualized data centers assign resources to the VMs in the system according to the predetermined SLAs. A virtualized enterprise server provides resources to its resident VMs. The data center virtualization manager can perform a preemptive scheme to load balance the system by the use of live VM migration.

The origins of VM migration can be traced back to *process migration* in distributed systems. Enabling live virtual machine migration is a much easier task than migrating processes because VMs have all of their context enclosed within themselves. As mentioned, the VM runs under the VMM, therefore, the virtualization manager just needs to forward the current state of the VM from the source host to the destination host and resume execution on the VMM of the target. The VMM handler on the source can be killed, eliminating residual dependencies. From the VM's point of view, the migration is completely transparent. The advantage of having an enclosed state
has allowed VM migration to be supported in multiple virtualization platforms with relative ease. Examples of such implementations include the Xen hypervisor [22] and VMware VMostion [23, 24]. These follow a migration scheme similar to the pre-copy process migration scenario described previously. The main reason is to reduce the impact on performance and keep the cost of migration at a minimum. If the migration fails, the VM continues to execute on the source, which causes minimal impact on the whole system. Thus, VM migration has all of the characteristics desired in a successful migration mechanism.

1.4 Power Consumption in Data Centers

The continued growth in the number of data center servers has also increased the total energy used by the servers and the power infrastructure needed to support them. This increase in energy use generates a significant amount of added cost that includes energy charges for enterprises and government agencies, increased emissions from electricity generation, load increases on the current power infrastructure to meet demand, and increased cost for the expansion of current and the development of new data centers. According to the U.S. Environment Protection Agency (EPA) congressional report on data center infrastructure [25], in 2006 alone, data centers consumed 61 billion kilowatt-hours (kWh) for a total electricity cost of $4.5 billion. This quantity is expected to grow to more than 100 billion kWh or $7.4 billion of electricity cost. However, the same report identifies possible improvements for up to 80% of infrastructure energy efficiency as shown in Figure 1.4. The "state of the art" scenario includes the following recommendations:
Figure 1.9: Projected electricity use for US data centers. Source: 2007 EPA congressional report on server and data center energy efficiency.

- The use of liquid cooling, improved airflow design, improvement in power distribution schemes.

- The full elimination of legacy servers.

- An aggressive adoption of energy-efficient servers.

- An aggressive consolidation of servers.

- An aggressive consolidation of storage.

- The establishment of power management at different levels including applications, servers and other equipment (networking and storage).

All of the previous improvements may reduce total energy consumption to levels of the year 2000. The main limitation impeding the implementation of these improvements is that some of them imply added cost of equipment that may not be feasible for businesses and government agencies to implement immediately. On the
other hand, the report lays out an effective plan that includes power management at the application level in which the system is power aware and can dynamically reduce or keep power consumption under control. Other recommendations (e.g., server consolidation) suggest that virtualization can be the right path to computer resource efficiency while at the same time obtaining energy efficiency.

As a key element of this thesis we will look at how we can leverage a virtualization infrastructure to achieve better power efficiency in the data center. While workload balance and VM migration can be used to improve performance, they can also be used to drive power efficiency.

1.5 Scope and Contributions of This Thesis

The key contribution of this thesis is the introduction of the Dynamic Resource Efficiency Manager (DREM). The resource efficiency manager provides an automated hardware resource manager for virtualized servers that relies on live VM migrations. The manager can be tuned to issue migrations following different system efficiency metrics. This scheme provides for data centers a much more robust framework for the management of system resources. The benefits of this are numerous and are detailed in Chapter 3.

Other contributions of this thesis include:

- the development of a metric to capture the load of a virtualized enterprise server,
- the development of a system resource efficiency metric for load balancing,
- the implementation of a DREM Managed Virtualized Data Center Testbed,
• an evaluation of DREM for load balancing a data center running enterprise workloads,

• a comparison of DREM to available commercial load balancing solutions that offer similar capabilities such as VMware DRS [26],

• the development and implementation of a system resource efficiency metric for power savings,

• an evaluation of DREM for saving power in a data center running multiple enterprise workloads, and

• an analysis of the system metrics used in this thesis and how the different system resources, specifically CPU and memory usage, contribute to the proposed efficiency metrics.

1.6 Organization of This Thesis

The remainder of this thesis is organized as follows. Chapter 2 provides a summary of existing techniques to load balancing on virtualized data centers. This chapter will also present a summary of techniques to reduce power consumption. Chapter 3 introduces the theoretical underpinnings for DREM. We then move on to develop methods to estimate the load of a virtualized enterprise server. We conclude the chapter with a description of the DREM Virtualized Data Center testbed implementation that is used in our evaluation methodology. Chapter 4 describes the use of DREM to load balance a Virtualized Data Center. We present results on load balancing from our DREM system, as well as an analysis of the performance improvements found when using this method. A comparison of the system to commercial solutions with
comparable behavior is also shown. Chapter 5 describes the use of DREM to save power in the Data Center. We present results in terms of power savings, as well as the power-performance effectiveness, of our approach. An analysis of our new system metrics is provided on Chapter 6. We examine the components of the metric and their behavior during system execution. Finally, Chapter 7 provides a summary of the thesis, discussing specific contributions, and discusses some limitations of the current framework and directions for future work.
Chapter 2

Related Work

Our work implements a hardware resources manager for virtualized servers to achieve specific resource efficiency goals. We are interested in using this framework to evaluate potential improvements in workload throughput and power consumption. In this thesis we propose the use of VM migration to arrive at these goals. There has been a growing body of prior work done in the area of VM migration in virtualized server environments, which is natural due to the growing popularity of VM technology. In this chapter we first present prior work addressing traditional server virtualization technologies, and then prior work on VM migration, dynamic resource management (load balancing) and energy savings.

2.1 Server Virtualization

As discussed in Chapter 1, we can trace virtualized environments back the work of IBM starting in the 1960s [16]. During this period, computers were mostly defined as mainframes that processed programs from different users sharing the system for
different purposes. The first virtualized environments surged in popularity due to design limitations in the Instruction Set Architecture (ISA) and time-sharing Operating Systems (OS) that to this day remain relevant: the dual-mode instruction execution capability. The ISA offers a dual-level hardware organization with a privileged and an unprivileged mode. The privileged mode provides the capability of executing all of the ISA’s instructions while the unprivileged mode restricts the use of certain instructions, comprised mostly of those that handle system resources such as memory management and I/O. OS designers take advantage of this dual-level programming model to, among other things, implement time-sharing systems. In time-sharing systems, the OS administers and distributes resources between users by dividing the time each user has access to a resource. To accomplish this task effectively, OS designers also divide program execution in different levels that commonly include user-level and kernel-level (i.e., OS-level) execution. User-level code executes code in ISA non-privileged mode, while kernel-level code has control of the privileged mode. Resources are then managed by the OS through the use of privileged instructions. When a user-level program needs to access a resource, it makes a request to the OS which will access the resource on its behalf. This level of abstraction allows the OS to transparently share resources among multiple users without causing users to conflict with one another when sharing resources. However, running in non-privileged mode restricts the user’s ability to interact with the hardware and other resources directly.

Virtualization on the other hand, allows the execution of multiple operating systems to execute in a single computer, providing users the appearance that they have complete control over the entire system, and can also support execution of privileged instructions. The approach consists of having the Virtual Machine Monitor (VMM) to control system privileged mode and handle the execution of multiple operating
systems in the same machine. Whenever an OS attempts to use a privileged mode instruction, the VMM will trap on its issue and service it accordingly, giving the illusion to the OS that it is in charge of privileged mode. This implementation enables users to have separate contexts (isolation) of both programming levels, while still sharing the same hardware. Some key advantages of this approach are that we can run different operating systems on the same hardware, provide privileged mode software development and testing environments, and increase overall system reliability and data protection.

Although virtualization has been commonly used on mainframes since the 1960s, current virtualization environments gained popularity only after a viable virtualization platform was made accessible on low-cost commodity hardware environments. Given the ubiquity of x86 servers available in the late 1990s and early 2000s. Elements of current commercial products followed the academic efforts by Mendel Rosenblum’s group at Stanford University [27, 28]. Later in the multi-core era (starting around 2005), interest in the use of virtualization for server consolidation started to gain popularity to the extent that first AMD [29] and then Intel [30] implemented virtualization extensions to their x86 ISA in order to overcome performance limitations imposed by moving gracefully between privilege levels. The rapid rise in popularity and adoption of virtualized x86 multi-core environments has helped motivate the need for the research carried out in this thesis.

Virtualized frameworks can be categorized depending on the level of detail of the virtualization infrastructure. **Full-Virtualized** systems are characterized by implementing all the necessary components such that a regular OS can run transparently on top of the VMM. **Para-Virtualized** environments on the other hand, do not implement all the necessary details to run a regular copy of the OS on the VM and rely
on OS extensions or a modified OS to support virtualized execution. There are clear advantages for using full-virtualization, including flexibility and OS transparency. Typically, para-virtualization requires optimizations in the OS to avoid performance bottlenecks.

Traditional x86 virtualization technologies allow the use of VMs in two different scenarios:

- **Hosted** environments - the VMs run on top of a host OS. The VMM runs in user-level code and interacts with the OS to handle resources on behalf of the different VMs. The VMM shares user space with any other application in the OS. Example of hosted environments would be: VirtualBox [19], Xen [31], KVM/QEMU [32] and VMware Workstation [33].

- **Hostless** environments - the VMs runs inside a thin-layer OS that handles basic functions. The hostless system is designed to be used exclusively for the VMM and its VMs so there is no sharing of user space with other applications. It is the preferred option in commercial grade server consolidation solutions. Examples of such systems include Microsoft Hyper-V [18], Citrix XenServer [20], IBM zVM [34] and VMware ESX server [17].

### 2.2 VM Migration

In chapter 1, we discussed that current VM migration algorithms are based on prior research on process migration. The main advantage of migrating an operating system using virtualization is that the state of the VM can be encapsulated without *residual dependencies*, a task that is very difficult to accomplish working at the process level.
because of OS state dependencies associated with processes such as open file descriptors or network sockets. Using virtualization, all of these structures can be effectively encapsulated inside the VM state, thus avoiding such dependencies.

Most of the initial research on x86 virtualization revolves around a hosted system model. Chen and Nobel’s work discusses what x86 virtualization could bring to the management of the physical system [35]. In their scenario, the host OS would only provide services for local administration and help the VMM handle services requested by the resident VMs. They propose that the host OS could then provide services independently and transparent to the guest OS, such as intrusion prevention and detection, and environment migration. This work presents VM migration as a solution to provide compute mobility to users. Following this approach, users can move VMs based on the particular physical hardware they are presently running on. The authors also consider some of the challenges that need to be addressed before VM migration can be effectively implemented, including the efficient handling of the VM state. The state of a VM includes its address space, which may be large. Therefore, saving and moving a VM state across the network can be a costly operation. The authors also mention ISA dependence factors. A VM must be able to migrate to another x86 family processor without depending on a particular feature present in only select x86 implementations.

The possibility of creating hardware independent environments has recently gained popularity with Kozuch and Satyanarayanan’s work [36]. The authors focus on the benefits of providing users with the capability of having similar behavior for suspend/resume operations available on notebook computers, but without the need to carry the physical equipment. They call this feature Internet suspend resume. A user would simply suspend execution at one starting location and resume execution on a
selected destination. This work was the first to present a prototype that successfully suspended an x86 VM on one physical host and resume it on a different host. As Chen and Nobel predicted, their results show that the state of the VM can be of a significant size, so in order to provide a commercial grade VM migration system, reducing this overhead must be addressed.

One solution that reduces the overhead of migration of VMs was suggested by Sapuntzakis et al. [37]. This work introduces several innovative designs aimed at reducing the size of the state needed for a successful VM migration. The authors refer to the state of a running VM as a \textit{capsule}. They manage to reduce the size of the capsule significantly by using four major optimizations:

- \textit{Copy-on-Write}. When transferring disk data, just keep track of writes to the capsule disk and only transfer those as an incremental update.

- \textit{Memory Ballooning}. Run a ”balloon” program inside the OS. The balloon will request large amounts of memory to force the OS to page out used memory. It will then fill the requested memory with zeros for easy compressing and minimize transfer size.

- \textit{Demand paging}. Instead of sending the entire disk, just send disk pages on demand, taking advantage of OS features that reduce disk fetch latencies.

- \textit{Hashing}. Use collision-resistant hashing to identify and avoid sending data that is already available at the destination node.

All of the state is then compressed using gzip and transferred to the destination node through the network. The authors show that these four optimizations managed to reduce the amount of information by 70% to 90% in disks and 94% to 96% of
memory. This work helped to open the door to the deployment of virtualization and VM migration in commercial data centers. However, there is still the need to suspend the VM, transfer it and then resume it on the destination node. The only missing feature is thus the ability to migrate a VM without suspend/resume so that administrators in a data center can service faulty hardware while still maintaining the system online.

Once VM migration was shown to be feasible, there was also a paradigm shift from hosted to hostless VM environments in which all VMs share a cluster of physical hosts and can be migrated easily from host to host. The definitive adoption of hostless VM environments on data centers arrived with live VM migrations. Live VM migrations enabled the data center with all the tools to move VMs across the system without the need to suspend VMs and hence provide zero-downtime capability, a valuable tool for system administrators to enforce SLAs.

The first work on live VM migration was published by Clark et al. [22]. This work presented a scheme to enable live VM migration in the Xen VMM. Almost simultaneously, Nelson, Lim and Hutchins [23] presented similar work for the VMware VMM, which is the basis for what it is known today as VMware’s VMotion [24]. The creation of these mechanisms have opened the door for most of the research being conducted in this area. Both works present a Pre-Copy live VM migration approach, with the intuition that by reducing VM downtime, performance is less impacted. Hines and Gopalan have recently presented a post-copy (Copy-on-Reference) method for the Xen VMM [38]. Their argument is that the performance gain by transferring memory only once (avoiding the multiple transfer of dirty references that occur in Pre-Copy migrations) can offset the penalty of transferring memory requests when referenced.
CHAPTER 2. RELATED WORK

2.3 Dynamic Resource Management

Dynamic Resource Management has its origins in multiple areas of research such as parallel and distributed computing [39] and autonomic computing [40]. In this thesis we are focused on dynamic resource management on virtualized systems. Most previous work that studied the benefits of VM migration focused on dynamic resource management. However, these techniques are oriented towards individual system resources. Choi et al. [21] proposed an autonomous learning method for threshold-based VM migration. In their work, they focus on dynamically changing migration thresholds. They base their approach on a migration history log that keeps records of previous migrations and proposes new migrations following the observed behavior of the system. They use User-Mode Linux (UML) VMs. Their approach for developing a resource efficiency metric considers the standard deviation of CPU resources. Their results show that their learning approach tends to obtain better results than a pure threshold-based scheme. However, these results are heavily dependent on the metric used.

Park et al. [41] propose a self-managing solution that takes into consideration application service level objectives (SLOs), and an optimization model based on linear programming (LP), specifically a Branch- and-Bound solution to the Binary Multiple Knapsack Problem [42]. They propose an optimal solution to their problem based on the implemented cost function applied to each VM. However, they mention as a limitation the fact that their approach can produce migration cycles in which a heavily used server migrates a VM continuously because it destabilizes the system on each migration (migration bounces).

Kochut and Beaty [43] have developed an analytical model of virtual machine migration that provides estimates of the expected gain in response time due to the
selected migration. The model is based on queueing theory [44] and takes into consideration the characteristics of a virtualized environment such as migration cost and overhead. They present simulations using data center CPU resource utilization traces. They assume that the physical servers follow an M/M/1 queue behavior to derive the response time equations for the system. The migrations are performed by selecting the one that minimizes the response time of the system.

Wood et al. [45] presented a VM memory-sharing-aware placement system. The system relies on memory fingerprinting algorithms to determine the sharing potential among a set of VMs, and based on that information, compute more efficient placements. It also makes use of live VM migration to maintain global memory-sharing benefits as workloads change. Their results show a 17% increase on the amount of VMs that could be resident on their data center while imposing a low overhead and a scalability of up to a thousand servers.

The work most similar to ours is by Khanna et al. [9], which targets application performance management. Their work is focused on server consolidation. The authors define the residual capacity of a server as the unused resources that may be used by a VM being migrated to it. The authors explain that a physical host needs to have a high residual capacity in order to accommodate incoming VMs, therefore their migration metric is based on maximizing the variance of the residual capacities of the physical servers and also considers the cost of the migration.

VMware DRS [26] is a commercial application that performs dynamic resource allocation in VMware’s Virtual Center (VC). It monitors overall system behavior and migrates VMs to balance the load by issuing VMotion commands (live VM migration). As it is a commercial product, there is limited information available on DRS internals. However, available documentation shows that DRS uses some form of per-resource
imbalance using their concept of VM shares and entitlements and produces a metric based on the standard deviation of the per-resource imbalance values of the system.

2.4 Energy Savings

Saving energy has become an important feature in the operation of data centers, especially when considering costs and the availability of power resources. In this section we describe prior research in energy savings in virtualized environment considering both physical and virtualized server systems.

2.4.1 Physical Servers

One proposed approach is to dynamically allocate servers for power and performance [46]. The idea behind this scheme is that since server provisioning is done under maximum load conditions, the data center can operate most of the time with a significant percentage of the servers powered down and whenever system performance levels require more resources to maintain SLAs, servers can be brought back online. However, if this scheme is used exclusively and often, we may wear out system components and reduce the life expectancy of the system.

Instead of powering servers, the work of Felter et al. [47] is centered on the reduction of peak power consumption in server systems. Peak power consumption is reduced by employing dynamic allocation of power among components of the system using real-time workload-guided profiling and power estimation techniques. The main idea is that power consumption can be tied to application execution.

Another approach that follows this application-power relation for multiprocessor systems is to balance consumption using energy-aware CPU scheduling, which can
also be used to avoid overheating on CPU cores [48]. This prior work showed that system performance counters can be used to estimate the power consumed over a period of time, with an error of less than 10%.

As the EPA congressional report suggested, server consolidation has also been proposed to reduce power consumption [3]. The idea is that by consolidating applications, we can reduce the number of servers needed at a particular time and thus save energy. However, this work shows that using this approach may produce a performance-power tradeoff, as applications contend for resources inside a system can degrade performance.

Although this previous research does not specifically target virtualized servers, it shows that power consumption in server systems is application dependent and that power consumption can be effectively reduced dynamically to maintain certain energy budget restrictions.

2.4.2 Virtualized Servers

With the introduction of virtualization technology into data centers, there has also been interest in understanding and managing power consumption on such large-scale systems. Nathuji and Schwan’s work attempts to manage power in virtualized enterprise servers on two levels: 1) the VM layer or soft power scaling and 2) the hardware layer or hard power scaling [49]. In the soft layer, the VMM can limit a VM’s access to resources and reduce usage transparent to the VM, while in the case of the hard layer the VMM can employ system-wide techniques such as dynamic frequency scaling. Stoess, Lang and Bellosa [50] follow a similar multi-layer breakdown of power analysis, where they take advantage of para-virtualization (a technique that allows for virtualization by modifying the VM’s guest OS) to create an energy-aware VM
system.

Other work in the area takes a system-wide approach to manage power versus by focusing on individual servers. A common approach is to pursue an effective VM placement methodology to obtain power savings. Cardosa, Korupolu and Singh [51] use the concept of VM Shares, which are a VMM defined concept. VM shares are the amount of resources a particular VM is entitled to receive at any particular instance. The VMM uses VM shares to establish a fair scheduling policy for VMs sharing system resources. The shares of a VM are used, along with a server power cost, to decide which server is the more energy-efficient to handle that VM.

Live VM migration has also been proposed as a method for reducing power consumption in virtualized clusters. The Magnet system is a scheduling policy to reduce virtualized cluster power consumption by using live VM migration [52]. The idea is to have a subset of the physical hosts powered down to save energy. This work treats VMs requests as two different job blocking problems. The inner job blocking problem is mostly caused by variable workloads that possess heavy resource requirements, and which may lead to node thrashing. The outer job blocking problem on the other hand, arises when a request for a large amount of resources is received. Magnet arranges hosts in a multi-layer ring based on the type of workload making the request. Inner layer rings handle inner job blocking problems and outer layer ones handle the outer job blocking problem. Live VM migration is performed between hosts in the same ring until the requests from a particular layer get serviced. Hosts can be promoted from layer to layer depending their current load and resources available. The major limitations of this work are: 1) the complexity associated with handling not only the workload on the system, but the host rings and 2) Magnet only deals with homogeneous server configurations.
The pMapper is another proposed scheme to manage power in virtualized systems [53]. This work also uses VM migration to dynamically handle power consumption in virtualized clusters. It is focused on reducing power and migration costs, while minimally impacting performance. The main contribution of pMapper is that it established a system-wide model that included performance, power and migration costs. This allows different configurations to handle different workload behavior. Another advantage is that it can be applied to heterogeneous systems. pMapper’s main limitation is that it requires additional specialized equipment such as energy managers and workload arbitrators to manage VM migrations.

We have seen that current implementations lack several components that may improve overall resource efficiency. First, there are no generalized expressions for a multi-resource performance target. Having such a performance target would allow us to better tune workloads based on the physical systems present, and the resource consumption of a particular workload. Another missing component is a resource manager that can improve resource efficiency while considering multiple objectives such as system load and power consumption. The Dynamic Resource Efficiency Manager developed in this thesis is designed to address these needs.
Chapter 3

The Dynamic Resource Efficiency Manager

In this chapter, we present the resource management framework that we have named the *Dynamic Resource Efficiency Manager* (DREM). The resource efficiency manager is intended to be a framework that communicates with the data center virtualization manager. Figure 3.1 illustrates this configuration. The resource efficiency manager collects state information about available resources from the virtualization manager. With the gathered information, the manager can then issue VM migration actions to obtain desired system objective. Figure 3.1 presents load balancing and energy savings specifically, since those are the target system objectives of this thesis.

DREM is designed with the knowledge that a virtualized data center system consists of multiple resource components that interact together to provide service to the applications that run on it. Therefore, we can quantify the usage of each individual resource independent from other available resources. This is of particular interest
in the case of virtualized enterprise servers that naturally share physical processors, memory and IO across applications. In the general load balancing problem, the most commonly used deterministic metric to determine server load is the CPU utilization. This is the case because CPU resources are usually the most heavily used in a computer system. However, given the diversity available in current enterprise workloads, the metrics capturing contention for other resources in the system provide important information for a better characterization of system performance.

In a virtualized server environment, relying only on a CPU utilization metric for load balancing can lead to load balancing decisions that can negatively impact the performance of the system. For example, consider a virtualized enterprise server that contains several VMs each running either CPU, memory or disk intensive workloads. The majority of the CPU resource usage on the server will be due to the CPU-intensive VMs. If we were to quantify the load of this server by restricting our focus to CPU
utilization only, we would be leaving out valuable information when analyzing the memory and disk intensive workloads. Using this limited system perspective might lead the load balancing algorithm to make poor decisions, creating bottlenecks in memory or disk. Thus, we are interested in a resource management system that carefully considers contention for each system resource.

The rest of the chapter is organized as follows. First, we present how we measure VM resource efficiency. Following this, we present how we can measure the load of a physical server based on the VM resource efficiency of the current VMs residing on it: The Virtualized Server Load Metric (VSL) and describe its components. We then present our database testbed which is a full DREM physical system implementation. Finally, we summarize the benefits of DREM.

3.1 Quantifying The Virtualized Server Load

3.1.1 Desirable Properties of a Virtualized Server Load Metric

Virtualization introduces another layer of abstraction on top of a physical server – VMs are only aware of virtual hardware resources, while the VMM manages both virtual and physical hardware resources and is responsible for performing the virtual to physical resource mapping. As a result, it becomes more challenging to balance performance if we rely solely on per-VM utilization of system resources. The VMM can hide the latency associated with the handling of physical resources to the VMs. Working at the VMM level instead, allows us to identify each VM’s virtual and physical resource consumption and enables us to isolate the resource consumption of a particular VM when assessing system performance. This flexibility also allows us
to make predictions on future system state if we were to move one VM to other physical server.

Another desirable property of a virtualized server load metric is that it should be robust enough so that it can be used on heterogeneous systems. Since new systems are being added or replaced constantly in a data center, real-world data centers are rarely homogeneous. Thus in order for a virtualized server metric to apply to a variety of system configurations, we need to quantify the capabilities of, and the load on, a virtualized server in a manner that does not depend on fixed resource units. A unitless metric allows the direct comparison of different servers in the system regardless of their internal components characteristics.

### 3.1.2 VM Resource Efficiency

A virtualized server contains several specialized hardware resources to provide its services. Although each resource can be composed of several components, we are interested in resource metrics that are measured at a coarse grain. We are interested in capturing possible VM application execution inefficiencies, and then determine how we can increase resource efficiency. Therefore, we have identified four critical resources a VM will typically consume. These are: the CPUs, the memory infrastructure, the storage infrastructure and the network bandwidth. We will need to develop a metric that incorporates an efficiency measure of these resources.

For the purpose of defining such a metric, let $R = \{CPU, memory, disk, network\}$ be the set containing our four resources and $resource \in R$ be a resource of interest. Let $VM_{Host}$ be the set of virtual machines currently running on physical server $Host$. We can express the efficiency of a particular VM $v \in VM_{Host}$ in terms of the ratio of $v$’s $resource$ usage over the total capacity of $resource$ that $Host$ provides. In other
words, the VM resource efficiency for VM \( v \) can be expressed as:

\[
VM_{\text{resource}}(v) = \frac{U_{\text{resource}}(v)}{C_{\text{resource}}(\text{Host})}, \quad (v \in VM_{\text{Host}}),
\]

(3.1)

where \( U_{\text{resource}}(v) \) and \( C_{\text{resource}}(\text{Host}) \) are the usage and the host physical capacity for each resource respectively. We can select resources according our desired resource-awareness level of the metric, for example, we can limit the resources to be a subset of \( R \).

Note that this metric measures the impact a particular VM has on the physical server it is running. It is based on profile data available at the system level (i.e., at the VMM level). It also ties VM resource usage (\( \forall v \in VM_{\text{Host}} \)) to the host capacity for that resource (\( C_{\text{resource}}(\text{Host}) \)). This makes the measurement unitless and starts at 0 when all \( U_{\text{resource}}(v) \) values are zero and its maximum depend on the amount of resources considered; it is \(|R|\) when all resources are used.

### 3.1.3 Measuring the Load of a Physical Host

Given \( VM_{\text{resource}}(v) \), we can express the load of a physical system. Let \( S \) be the set of physical servers in data center \( D \). Each server contains a certain amount of the aforementioned resources. There are none, one or several VMs residing in each server, each consuming a fraction of the available resources. Then, The Virtualized Server Load of server \( \text{Host} \) (\( VSL_{\text{Host}} \)) can be expressed as a linear combination of the available resources being consumed by \( VM_{\text{Host}} \):

\[
VSL_{\text{Host}} = \sum_{\text{resource}} W_{\text{resource}} \times \left( \sum_{v \in VM_{\text{Host}}} VM_{\text{resource}}(v) \right),
\]

(3.2)
for all resources being considered.

\(VSL_{Host}\) varies dynamically, depending on the VMs running on the physical system (\(Host\)). This fact makes it suitable to represent a range of application-dependent system characteristics and thus it can be used as a metric for managing resources. \(W_{resource}\) is a weight associated with each \(resource\). The resource weights have been provided in the equation as a means of providing a general metric that can be adjusted to hosts with different characteristics. For example, they can be adjusted to a specific resource quantity so that all nodes in the data center have \textit{normalized} resource values.

### 3.2 DREM Implementation: Physical Testbed

We have built a virtualized data center testbed consisting of six virtualized enterprise servers attached to two different storage systems and managed by two management servers. The virtualized servers are connected to a Watts up? PRO ES power meter. Figure 3.2 shows the system components and their hardware configuration. The DREM server can collect information of the current load present on the system by interacting with the Virtualization Manager. It can also collect power consumption measurements by communicating with the power meter through a USB interface. The system was implemented using VMware infrastructure tools as shown in Figure 3.3. One important reason for choosing the VMware VI API for the implementation of the Dynamic Resource Efficiency Manager prototype is the presence of an available commercial product in this infrastructure that could be used as a baseline to evaluate the effectiveness of the load balancing framework, that is VMware DRS. Even though the current prototype uses this API, there are other hypervisors that provide the
Figure 3.2: Virtualized Data Center. The system is composed of a set of 6 Virtualized Enterprise Servers (top) sharing a Storage Sub-system (bottom) and two servers for managing the system (middle): the Virtualization Manager Server and the Dynamic Resource Efficiency Manager server. The Virtualized Enterprise Servers are connected to a Watts up? PRO ES power meter that reports power consumption measurements to the DREM server through a USB interface. The Virtualization Manager and the Dynamic Resource Efficiency Manager collaborate together to handle VM migrations across the Virtualized Enterprise Servers.

similar capability of extracting the same information. In particular, the Xen hypervisor provides a set of profiling tools and an API [54, 55, 56] that is robust enough to provide the same information to the DREM. Thus, the resource efficiency manager can be reconfigured to use other hypervisors with only minor changes. In this sense, it is an advantage of the DREM over DRS because of the latter is a custom built-in
Figure 3.3: Experimental testbed of the Virtualized Data Center. The data center contains 18 VMs running on 6 Virtualized Enterprise Servers sharing two iSCSI LUNs. The DREM service attaches to VMware Virtual Center Server to collect state information and handle VM migrations across the Virtualized Enterprise Servers using the load balancing or the power savings modules.

There are two storage systems of 1TB capacity each attached to the virtualized servers, both of them presented through iSCSI SAN VMs. One of the iSCSI LUNs is backed by a DELL PowerVault Disk Array attached to a DELL PowerEdge 2950.
server with RedHat Enterprise Linux 4 and VMware Workstation version 6.0. The other VM is backed by an EMC CLARiiON CX300 FC Storage Array attached to a second PowerEdge 2950 server with ESX 4.0 Enterprise Edition. This design choice is selected because the storage infrastructure is located in a different facility and that prohibits any dedicated direct connection to the evaluation servers. Consequently, because not all of the data center components are located in the same facility, the network resource will be stressed to additionally handle all disk I/O. We are not presently not considering the network and storage components of the VSL metric and only concentrate on a subset of resources: \( R' = \{CPU, memory\} \) \((R' \subset R)\) which are the ones directly available on the physical servers. Future versions of the DREM testbed will consider the storage infrastructure.

The resource efficiency manager connects to VMware Virtual Center Server (VC) and calculates the Virtualized Server Load values for each physical host. After calculating VSL for all hosts, it will select migration candidates following the current efficiency objective and issue VMotion calls with the selected VMs.

The experimental evaluations are preformed on the testbed system hosting up to 18 VMs, as shown in Figure 3.3. We configure two different VM sizes (in terms of virtual resources; CPUs and memory). The VM Small configuration consists of two virtual CPUs and 1GB of main memory. The VM Tiny has one virtual CPU and 512MB of main memory. Both VM configurations have 20GB virtual hard disks. Figure 3.4 shows the VM types and sizes. The 18 VM files are evenly stored on both storage systems (9 VMs each).
3.3 Benefits of the Dynamic Resource Efficiency Manager

The Dynamic Resource Efficiency Manager can have immediate tangible benefits to the data center community; we provide a few examples here. First and foremost, the resource efficiency manager provides the means to quantify the current load of a virtualized server in a per-VM resource consumption scheme. This has several advantages. For example, by isolating resource consumption per each VM, we can immediately identify the VMs that are requesting more resources and those that are not. This gives us enough information to prioritize resources to the VMs that have an immediate need. Having per-VM resource consumption information also allow us to perform VM placement that avoids resource conflicts between applications. In this form we can consolidate servers without worry about overloading any physical resources. Other potential uses include VM intrusion detection or overall system anomaly detection. We can detect unusual behavior of a particular VM by analyzing its resource consumption history.

Other benefits provided by DREM are the flexible modular programming model
that enables future extensibility. Users can identify new performance targets they desire to meet and introduce them to the resource efficiency manager as a new module in their system. In other words, new modules enable the customization of the efficiency manager to specific user needs. Finally, all the proposed modules presented in this thesis are designed to be platform independent. This means that the resource efficiency manager can potentially be implemented to serve data centers regardless of their chosen virtualization platform with relative ease. This is a powerful benefit because it does not restrict the data centers’ administrators to a specific virtualization infrastructure in order to fully utilize the resource efficiency manager.
Chapter 4

Using DREM for Load Balancing

In this chapter, we focus on the use of DREM as a solution to the load balancing problem. Therefore, our primary focus is on presenting the load balancing problem in terms of \( VSL \); its behavior and what this means for overall system load values; and how to integrate and effectively use DREM to load balance a virtualized data center. To those ends, we begin the chapter with a discussion on measuring system load imbalance based on measured \( VSL \) values. Then, we show how we can build a load imbalance metric based on the variation in the different \( VSL \) measures from all the physical servers in the data center. We describe the \textit{VSL Inductive Balancing Method} (VIBM), which is the load balancing algorithm used by DREM. We then present experiments to evaluate how DREM, when used to guide VM migrations, can provide for better load balance. Finally, we compare these results with those obtained when issuing migrations using VMware’s DRS, a commercial load balancing solution with similar capabilities.
CHAPTER 4. USING DREM FOR LOAD BALANCING

4.1 Load Imbalance Metric based on VSL

Load balancing has been one of the most important techniques used to improve resource utilization and system performance in parallel and distributed systems. Although previous work has focused on specific balancing algorithms [11, 13, 14], it is difficult to directly apply them to virtualized servers. Additionally, we need to take into consideration the impact that the virtualization layer imposes on system resources. The virtualization layer can significantly change the known behavior of a workload running on a physical system.

It is important for a load balancing algorithm to capture the variation present across the loads of different individual servers. A typical imbalance metric based on the resource utilization of physical servers is the standard deviation of the CPU utilization [13]. The reasoning behind this metric is that if the server loads are evenly distributed (e.g., low variation between server loads), the standard deviation will be small. The smaller this metric value, the higher the probability that the load is balanced in the system. Although CPU utilization is generally a good predictor of system performance, the use of such a metric for load balancing does not take into consideration memory, network or disk I/O performance. In the case of a more complex workload such as VMs running commercial workloads, relying solely on CPU information may lead to undesirable results.

In the previous chapter, we defined a server load metric that is a function of VM resource usage (VSL). This metric considers information specific to each VM when quantifying the load of a particular physical server. Our proposed metric takes into consideration the usage of multiple resources by the VMs resident on the system. Based on this definition, we will build a load imbalance metric. Let us define load set $L$ containing the VSL values corresponding to all physical servers ($\forall$ physical
servers ∈ \(S\)). The desired system imbalance metric can then be defined in terms of the coefficient of variation of \(L\):

\[
C_L = \frac{\sigma_L}{\mu_L} = \frac{\frac{1}{|S|-1} \sum_{\forall \text{Host} \in S} (VSL_{\text{Host}} - \mu_L)^2}{\frac{1}{|S|} \sum_{\forall \text{Host} \in S} VSL_{\text{Host}}}
\]  

(4.1)

As shown, \(C_L\) is defined as the ratio of the standard deviation of the VSL values (\(\sigma_L\)) over the average VSL values (\(\mu_L\)). The coefficient of variation is used in many areas of computer science including queueing theory [44]. This metric captures the dispersion of the values assumed by a variable in a way that is independent of the measurement unit. The higher the \(C_L\), the greater the variation in the measured values. Note that this metric will give us a tighter bound than using only the standard deviation. For example, if we use \(C_L\) to predict future system state after migrating a set of VMs, the metric accounts for the migration set that not only reduces \(\sigma_L\), but that recognizes changes in \(\mu_L\) as well.

However, there are some problems that must be taken into consideration when using the \(C_L\) as an imbalance metric. The most evident problem is in cases where \(\mu_L\) is zero. For these cases, \(C_L\) will be undefined. Similar problems may occur if the VSL returns non-positive values. This is not an issue for our VSL metric, as long as we use positive values in \(W_{\text{resource}}\). Thus, the only scenario we need to worry about is the case where \(\mu_L\) becomes zero. The only time this could happen is when all servers are idle or when the virtual machine monitor (VMM) is not consuming any resources. Although these cases are extremely rare, in order to avoid this problem, we define our imbalance metric as:
CHAPTER 4. USING DREM FOR LOAD BALANCING

Figure 4.1: $C_L$ as a function of throughput in Transactions-per-Minute (TpM). For lower $C_L$ values there are higher TpM values.

$$I_{Metric} : \begin{cases} 
0, \text{ if no VMs are active} \\
C_L, \text{ otherwise}
\end{cases}$$  \hspace{1cm} (4.2)

Figure 4.1 shows the behavior of this metric for different throughput values of a virtualized system running an Online Transaction Processing (OLTP) workload. As seen, this metric captures the variability of $VSL$ across all hosts in $S$ and minimizing this metric produces a load-balanced set of servers. In this thesis the weight values ($W_{resource}$) in Equation 3.2 are assumed to be equal.
4.2 VSL Inductive Balancing Method

In this thesis we are interested in the use of live VM migration for load balancing. Based on $I_{Metric}$, we can start the design of a method for balancing the load in a virtualized data center. The migration criteria is to migrate a VM to a different host if the system is imbalanced. In the case of a data center with $N$ servers and $M$ VMs (where $|M| >> |N|$), the general problem of dynamically allocating VMs to physical servers has been shown to be similar to bin-packing [9] or knapsack [41] problems, both classic NP-Hard problems. We develop a greedy heuristic solution that follows an iterative approach by inductively selecting the VM migrations that yield the greatest improvement of the imbalance metric at its present state. This approach guarantees that we seek migrations that reduce the value of $I_{Metric}$ over the current running interval of the system. We then attempt to reduce $I_{Metric}$ further by performing migrations several times. In order for this approach to success we need a high accuracy in predicting future system state.

4.2.1 Building the Algorithm

Our load balancing problem can be stated as:

"to migrate VM $v_{candidate}$ from physical host $source$ to physical host $target$ such that $I_{Metric}$ is reduced"

The initial step for the load balancing module of the resource efficiency manager is to determine whether the system is unbalanced or not, that is, if $I_{Metric}$ is greater than some threshold value. Whenever it is over threshold, the resource efficiency manager looks for migration candidates. Our first heuristic focuses on the adaptive behavior of the resource efficiency manager. If we want to adaptively adjust the load
of the system, the selected VMs for migration must be identified fast and efficiently. In our iterative approach, we consider that those VMs to be migrated should be selected from the most heavily loaded physical host. In other words, this heuristic consists of giving a higher priority to the most heavily loaded server to offer VMs as a possible $v_{\text{candidate}}$. Therefore, if the resource efficiency manager detects that the system is imbalanced, it selects the physical system with the highest $VSL$ as the source host to select the next migration candidate.

Although the selection of $v_{\text{candidate}}$ might not provide the optimal solution for the problem, it is a reasonable starting point that will find VMs that are impacting system resource efficiency in a fast and effective fashion. This enables the resource efficiency manager to react quickly to an imbalance situation and rapidly attempt to bring the system to balance. This approach also reduces the size of the problem by just concentrating in one subset of all VMs to migrate. Once the node with the largest $VSL$ value is selected as our source node, the rest of the nodes are considered to be the possible target node. The eventual target node is selected based on a prediction of future system state. We evaluate what a predicted $I_{\text{Metric Predicted}}$ value for the system would be if VM $v_{\text{candidate}}$ ($v_{\text{candidate}} \in VM_{\text{source}}$) is transferred to VM$_{\text{target}}$. This prediction is made by adding the contribution of $v_{\text{candidate}}$ to the corresponding target load ($VSL_{\text{target}}$) and subtracting it from the load of the source server ($VSL_{\text{source}}$). The $v_{\text{candidate}}$ migration that provides the greatest improvement to the $C_{L_{\text{candidate}}}$ value will be selected for promotion. If the $I_{\text{Metric Predicted}}$ is below the current system $I_{\text{Metric}}$, the candidate gets promoted, otherwise the system is unchanged.

Now that we have all the necessary pieces in place, we can build an algorithm for load balancing a complete virtualized data center using DREM. We have named the
CHAPTER 4. USING DREM FOR LOAD BALANCING

Algorithm 1: VSL Inductive Balancing Method: VIBM

\[
I_{Metric} \leftarrow C_L
\]
while \( I_{Metric} \geq \text{threshold} \) do

source \( \leftarrow \) host \( \in S \): \( VSL_{host} = \text{Max}(L) \)

for \( v_{candidate} \in VM_{source} \) do

for (target \( \in S \) \( \neq \) source) do

predict benefit of migrating \( v_{candidate} \) to target
compute \( L_{candidate} \)

\[
L_{candidate} : \begin{cases} 
VSL_{target} \leftarrow VSL_{target} + v_{candidate} \\
VSL_{source} \leftarrow VSL_{source} - v_{candidate}
\end{cases}
\]

insert \((v_{candidate}, source, target, L_{candidate})\) to Candidates
end
end
select candidate from Candidates : \( \text{Min}(C_L_{candidate}) \)

\[
I_{Metric}^{\text{PREDICTED}} \leftarrow C_L_{candidate}
\]
if \( I_{Metric}^{\text{PREDICTED}} < I_{Metric} \) then

promote candidate to migrate

\[
I_{Metric} \leftarrow I_{Metric}^{\text{PREDICTED}}
\]
else

do not promote candidate
end
end

load balancing approach the \textit{VSL Inductive Balancing Method} (VIBM). The method is presented in Algorithm 1. After iterating through the algorithm several times and the migration candidates are promoted, DREM issues the corresponding migrations using the promoted candidates.
Table 4.1: OLTP Transactions and I/O pattern.

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Access Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>New-order</td>
<td>read/write</td>
</tr>
<tr>
<td>Payment</td>
<td>read/write</td>
</tr>
<tr>
<td>Delivery</td>
<td>read/write</td>
</tr>
<tr>
<td>Order-status</td>
<td>read</td>
</tr>
<tr>
<td>Stock-level</td>
<td>read</td>
</tr>
</tbody>
</table>

### 4.3 Experiments

Our initial experiments present results from running a simple CPU-intensive microbenchmark and an OLTP workload. The CPU-intensive workload issues simple integer operations similar to the Dhrystone benchmark [57]. It is used to show the accuracy of the VIBM algorithm to predict system state. The OLTP workload is used for performance tests. It is implemented based on the TPC-C specification [58], which models an online wholesale supplier managing orders. Order-entry provides a conceptual model for the benchmark, with the underlying components being typical of any OLTP system. Table 4.1 shows the five transaction types and their read/write characteristics. The transactions operate against a relational database composed of 9 tables. Transactions generate reads, writes, and rollbacks. The application uses primary and secondary key access. Our OLTP workload consists of warehouses, each containing 10 terminals, as outlined in the TPC-C specification. The terminal interface consists of a Linux terminal that displays information about the current transaction being performed. This workload presents a considerable amount of randomness in its behavior that even the same application may change resource consumption over time (burstiness) making it an excellent candidate to evaluate overall system performance.
Figure 4.2: Experimental setup for the load balancing problem, (a) DREM will issue VM migrations following DREM’s VIBM algorithm and VMware’s DRS recommendations; VM initial placements for the different experiments and number of servers used, (b) 6 VM and (c) 8VM experiments use 2 servers, (d) 10 VM experiments use 3 servers, (e) 16 VM experiments use 5 servers, and (f) 18 VM experiments use 6 servers.
CHAPTER 4. USING DREM FOR LOAD BALANCING

The experimental DREM setup for our imbalance metric and VM migration algorithms is shown in Figure 4.2 (a). Our experiments are designed to compare the effectiveness of our approach to improve resource efficiency. We present results for three different migration policies: 1) no migrations, 2) migrations issued by VMware DRS and 3) migrations issued by DREM’s VIBM algorithm. VMware DRS has certain migration settings to set how ambitious the algorithm should be when attempting to balance the system. Our results will be compared to those of DRS when configured with an aggressive migration setting. Running with this configuration, DRS makes recommendations even if the suggested migration only provides a slight improvement in the overall system balance. We want DRS to be as aggressive as possible to provide a very competitive baseline.

DREM is also in charge of monitoring the whole system and proposing load balancing solutions. Our implementation allows the selection of the type of migration scheme that will be performed during the workload analysis between the VIBM algorithm and DRS. In this way, we can conduct accuracy tests between $I_{\text{Metric}}$ and $I_{\text{Metric\_PREDICTED}}$ associated with each migration scheme to understand, evaluate and compare both methods from the same framework.

Table 4.2: VM name and type. VM Small configurations have id 1 thru 7 while VM Tiny types run from 8 to 18.

<table>
<thead>
<tr>
<th>VM Small</th>
<th>VM Tiny</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM1 VM2</td>
<td>VM8 VM9 VM10</td>
</tr>
<tr>
<td>VM3 VM4</td>
<td>VM11 VM12 VM13</td>
</tr>
<tr>
<td>VM5 VM6</td>
<td>VM14 VM15 VM16</td>
</tr>
<tr>
<td>VM7</td>
<td>VM17 VM18</td>
</tr>
</tbody>
</table>

The load balancing experiments include all 18 VMs running on all 6 virtualized
enterprise servers. In order to test the utility of the VIBM algorithm to increase overall system balance, we have partition our testbed data center into smaller sets and conducted tests with 6, 8, 10, 16 and 18 VMs and different number of servers available. The VM types being used are described on table 4.2 while the active servers for each experiment and the VM initial placements are depicted in Figure 4.2 (b)-(f). These scenarios are designed to test what the system balance improvements would be if we added an idle system to our cluster. They will also help us understand overall system behavior as well as to identify possible scalability problems or limitations in our testbed implementation.

The following sections will explain in more detail the experimental evaluation.

4.4 DREM Results

4.4.1 $I_{Metric}$ Analysis

Predicting System State

The first set of experiments evaluates the capacity of the DREM to predict system state. For this purpose we start with a configuration of 6 VMs: 2 small VMs and 4 tiny. For this particular test, we have selected the initial placement shown in Figure 4.2(b). A CPU-intensive workload was run on VMs 1 to 4 with VM5 and VM6 running background processes (just the OS). The VIBM algorithm migrated VM1 to the idle ESX server in the first iteration, VM4 in the second iteration and VM6 in the third. After making those changes, the $I_{Metric}$ of the system was reduced by more than 40%. Figure 4.3 shows the results. As we can see, each migration resulted in an improvement in the overall system $I_{Metric}$ value. We can also see that the actual state is very close to the state predicted by the VIBM algorithm, with differences
ranging from 0-3.6%.

The second test consists of comparing the VIBM migration pattern with that of VMware’s DRS. For this experiment we also use the CPU-intensive workload and the 6 VM system configuration with all VMs running the same workload. We made separate runs of the DREM configured to use DRS and VIBM. Figure 4.4 compares the results of the recommendations provided by VIBM versus those offered by DRS. The VIBM algorithm elects to migrate VM1 to the idle ESX server in the first iteration and then migrates VM4 in the second iteration. After that, the system is well-balanced, reducing the $I_{Metric}$ by more than 60%. DRS, on the other hand, suggests different movements for this workload. DRS only moves VM2 to the idle ESX server which results in an improvement of close to 37% in the $I_{Metric}$. Figure 4.4 also shows the accuracy of $I_{Metric, Predicted}$ when either DRS or DREM is used to find the next migration, and compares this prediction to the actual value after the migration has taken place. As we can see, the predicted values are within 2.8% and 2.9% of the actual values. These results suggest that we can use $I_{Metric}$ to predict future system behavior after a move with a high accuracy.
CHAPTER 4. USING DREM FOR LOAD BALANCING

Figure 4.4: Comparing the accuracy of $I_{Metric}$ prediction following different migration recommendations: DRS (a) and DREM (b). Notice how DREM selects the migrations that better reduce $I_{Metric}$.

4.4.2 DREM-DRS Comparison

Now that we have shown that the $I_{Metric}^{predicted}$ can provide very accurate predictions of future system states, we also want to compare our DREM VIBM algorithm to DRS to see how well it compares to the state-of-the-art. For this purpose, we
focus on the relationship between $I_{Metric}$ and system performance. To study this question, we use the OLTP workload. Two sets of experiments are designed using the initial placement configurations mentioned earlier. The workload parameters used in our OLTP implementations vary based on the size of the VMs used, as shown in Table 4.3.

Table 4.3: VM OLTP workload description.

<table>
<thead>
<tr>
<th>Warehouse Number</th>
<th>Number of Terminals</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM Small</td>
<td>5</td>
</tr>
<tr>
<td>VM Tiny</td>
<td>2</td>
</tr>
</tbody>
</table>

We begin by letting the system warm-up for 25 minutes prior to starting the VM migration routines. After this warm-up period, the system runs for one hour while the VM migration handler is migrating VMs following either DRS or DREM recommendations. These results are also compared to a no-migration scheme.

Table 4.4: $I_{Metric}$ and throughput relation for the different VM configurations. The average $I_{Metric}$ value is presented along with the average OLTP throughput in transactions per minute (TpM). A lower value of $I_{Metric}$ yields an increase in transaction throughput.

<table>
<thead>
<tr>
<th>Migration Method</th>
<th>6 VMs</th>
<th>8 VMs</th>
<th>10 VMs</th>
<th>16 VMs</th>
<th>18 VMs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$I_{Metric}$</td>
<td>TpM</td>
<td>$I_{Metric}$</td>
<td>TpM</td>
<td>$I_{Metric}$</td>
</tr>
<tr>
<td>None</td>
<td>0.99</td>
<td>128.98</td>
<td>0.89</td>
<td>174.63</td>
<td>0.86</td>
</tr>
<tr>
<td>DRS</td>
<td>0.81</td>
<td>133.96</td>
<td>0.71</td>
<td>193.20</td>
<td>0.22</td>
</tr>
<tr>
<td>DREM</td>
<td>0.70</td>
<td>149.10</td>
<td>0.61</td>
<td>203.52</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Table 4.4 shows the average $I_{Metric}$ value and the average transaction throughput obtained for the 6 VM and 8 VM performance tests. As we can see, a high
$I_{Metric}$ value corresponds to a lower transaction throughput and vice-versa for all experiments. From the previous results we can see that by minimizing $I_{Metric}$ we can potentially achieve higher throughput. Figure 4.5 shows the behavior of $I_{Metric}$ for the experiments. We can see that DREM produces the lower $I_{Metric}$ values throughout the workload execution. By observing the results in Table 4.4 and Figure 4.5, we can see that DREM provides a greater reduction in the $I_{Metric}$ and hence obtains better performance.

Figure 4.6 presents results running the OLTP workload. Normalized throughput numbers are presented, computed by recording the number of new order transactions per minute (TpM), divided by the maximum number possible in the ideal case. The ideal case is obtained in the following form: single runs of each VM type are run in a dedicated ESX server without any resource contention; that is, no other VMs are running or using the shared storage space. This TpM amount obtained is linearly scaled to the size of the experimental setup. In the 6 VM experiment, VIBM obtains a 3% increase in performance when compared to DRS. The 8 VM results show a larger difference between the two methods, with a consistent performance advantage for VIBM of 5% over DRS. When we see the results for the 10, 16 and 18 VMs, VIBM performance is better by 8%, 6% and 5% respectively. As we can see in all cases, VIBM produced the best total throughput, obtaining the same transaction throughput as the single run results for the 6 and 8 VM configurations. The main reason for such high performance numbers is that the bottleneck in the system is the large amount of I/O generated to the iSCSI LUNs. In other words, any other latencies are hidden by the latency of the storage system. We can clearly observe this effect in the throughput values of the experiments with a larger number of VMs, where the obtained throughput is less than half of that predicted by the ideal case. This is also
Figure 4.5: A comparison of $I_{\text{Metric}}$ values over time per used method for: (a) 6 VM, (b) 8 VM, (c) 10 VM, (d) 16 VM, and (e) 18 VM experiments. DREM obtains the lowest average $I_{\text{Metric}}$ for all experiments.
CHAPTER 4. USING DREM FOR LOAD BALANCING

Figure 4.6: OLTP Transaction Throughput (TpM) for all VM configurations. DREM migrations produce a higher TpM in the two experiments.

shown in Table 4.4 where the throughput of DREM after 8 VMs and beyond remains basically constant at around 200 TpM. Given that this effect is equal in each scheme and it is also present in the experiments using DRS, the results give us confidence that using VIBM introduces little overhead and is comparable to the overhead that VMware vCenter Server already introduces.

4.5 Looking further: $I_{\text{Metric}}$ and Power

As we have mentioned, the DREM server has the ability to measure the power consumed by the virtualized servers during execution. Although we have not designed DREM’s VIBM algorithm to optimize power consumption in virtualized servers, we decided to see if $I_{\text{Metric}}$ could also be used as a metric to guide the reduction of power consumption in virtualized data centers.
Figure 4.7: Power vs Performance in OLTP Transaction Throughput vs total server consumed power (W/TpM) for all VM configurations. DREM migrations produce a better W/TpM ratio in all experiments.

We have collected power measurements during the experiments for all VM configurations. In our first analysis, we are interested in understanding the relationship between performance and power. As illustrated in Figure 4.7, DREM migrations tend to give a better watt/TpM ratio over all the experiments. These results are encouraging; the $I_{Metric}$ could potentially be used for reducing power consumption. On the other hand, the results may only reflect that the solution that provided the best throughput is also the solution that is most power efficient. Thus we also analyze the power consumption behavior through the VM migration phases. Figure 4.8 shows the total server power consumption during the VM migration experiments. As we can see, the consumed power during the experiments does not seem to be responding to changes in $I_{Metric}$. Furthermore, we present $I_{Metric}$ as a function of power in figure 4.9. This figure clearly shows that there is no direct correspondence between the
Figure 4.8: A comparison of total server power consumption over time per used method for: (a) 6 VM, (b) 8 VM, (c) 10 VM, (d) 16 VM and (e) 18 VM experiments. There is no clear form to select which method provides better power-performance efficiency.
Figure 4.9: A comparison of $I_{\text{Metric}}$ as a function of the total server power consumption over time per used method for: (a) 6 VM, (b) 8 VM, (c) 10 VM, (d) 16 VM and (e) 18 VM experiments.
$I_{Metric}$ and power consumed. From these results, we can conclude that although the VIBM algorithm proved effective in predicting system load, the algorithm working alone is insufficient for guiding power efficiency decisions in a virtualized infrastructure.

4.6 Summary

In this chapter, we examined the ability of VSL to capture overall system load and be used as the base metric utilized by DREM. In particular, we constructed a VM migration policy (VIBM) to better share available system resources between the VMs. VIBM relies on an imbalance metric ($I_{Metric}$) that measures the variability of VSL values throughout the data center. We also presented highly accurate predictions of future system state (i.e., with an error margin of less than 5%) by measuring per-VM resource usage. This is important information for data center administrators, in order to understand the impact that a particular VM migration will have on the whole system ahead of time. We demonstrated that issuing VM migrations that reduce the value of $I_{Metric}$ can result in higher workload throughput, even increasing performance over similar commercial solutions.

As a result of our experiments, we believe that VSL can be an important tool for virtualization infrastructure managers and designers to use in their efforts to improve system resource usage. We believe that VIBM provides an excellent load balancing policy to a virtualized data center. However, we can conclude that using $I_{Metric}$ to handle other resource efficiency targets such as power consumption would not result in an effective strategy.
Chapter 5

Using DREM to Reduce Power Consumption

In the previous chapter, we have defined a load imbalance metric \( I_{\text{Metric}} \) based on VSL. We have also shown that \( I_{\text{Metric}} \) was able to increase system performance but could not improve system power consumption. In this chapter, we focus on the use of DREM as a solution to the power consumption problem. We begin with a presentation of a VSL based metric aimed at improving power consumption throughout the data center. We then show how this metric compares to \( I_{\text{Metric}} \) and what are the factors that differentiate both. Following, we introduce the VSL Power Savings Method (VPSM) which is the power savings model used by DREM. We then present experiments to show how successful are DREM VM migrations for saving power. Finally, we compare the results in terms of performance degradation with those obtained when having the system balanced without issuing migrations, since it is the scenario in which we can obtain the most performance out of our system.
CHAPTER 5. USING DREM TO REDUCE POWER CONSUMPTION

5.1 Power Savings Metric based on VSL

We are interested in the use of live VM migration to improve overall power consumption. As stated previously, the migration criteria applied in DREM is to migrate a VM to a different host if a resource inefficiency is detected. In this particular case, high resource usage is directly proportional to the amount of power the system is consuming. Thus, we monitor variability in the VSL values for each physical host in the system. Similar to the VIBM algorithm, we look at load set $L$ and study the behavior of the collected VSL values. The average VSL value in $L$ should give us broad information about the amount of resources being used in data center $D$. The desired system power efficiency metric can then be defined in terms of the mean of $L$: \[
E_{Metric} = \mu_L = \frac{1}{|D|} \sum_{\forall \text{Host} \in D} VSL_{Host} \quad (5.1)
\]

As shown, $\mu_L$ is defined as the average VSL value in $L$. This metric captures the average resources usage without depending on the measurement unit. The higher $\mu_L$ is, the greater the resource usage and the power consumption of the server systems. Figure 5.1 shows the behavior of this metric for measured power consumption values of a virtualized data center of three active servers and running an OLTP Workload. As seen, this metric captures the average VSL across all hosts in $S$ and minimizing this metric produces a lower power consumption state on the set of servers.

Among all physical hosts, we detect the one with the lower resource efficiency by examining whether or not we can detect resource underprovisioning on the host. That is, the case when:
Figure 5.1: $\mu_L$ as a function of power in Watts (W). For lower $\mu_L$ values there are lower power consumption values.

\[
\left( \sum_{\upsilon \in VM_{Host}} VM_{resource}(\upsilon) \right) \ll 1,
\]  

(5.2)

for any resource. Note that $E_{Metric}$ is the denominator term on equation 4.1, which is the non-zero part of $I_{Metric}$. Thus, an increase in $E_{Metric}$ implies a decrease in $I_{Metric}$ and vice versa. This inverse relation between both metrics gives us an initial explanation as to why the VIBM algorithm does not reduce power consumption by reducing $I_{Metric}$. There are two forms of reducing $I_{Metric}$: 1) by decreasing $\sigma_L$ or 2) increasing $\mu_L$. If we decrease $\sigma_L$, we would at best case leave resource usage at the same level and if we increase $\mu_L$ we would be increasing resource usage. Therefore, both methods of decreasing $I_{Metric}$ should not work to reduce the average server load and consequently power consumption.
5.2 VSL Power Savings Method

5.2.1 Issuing Migrations

We can formulate the VSL Power Savings Method (VPSM) VM migration solution for power savings as:

"to migrate VM $v_{\text{candidate}}$ from physical host source to physical host target such that (5.1) is reduced"

The initial step for the model is to determine whether the system is in an inefficient state or not, that is, if $E_{\text{Metric}}$ is larger than a predetermined value. We then look for the most underprovisioned host or check whether or not (5.2) is valid for any physical host. Whenever this situation is detected, the migration manager searches for possible migration candidates. Our first implementation focuses on the adaptive behavior of the VSL metric. If we want to adaptively adjust the resource usage of the system, the selected VMs for migration must be identified fast and efficiently. In our iterative approach, we consider that those VMs to be migrated should be selected from the most underutilized physical host. In other words, this heuristic consists of giving a higher priority to the least loaded server to offer its VMs as a possible $v_{\text{candidate}}$. Therefore, if the model detects that the system is currently in a less efficient state, it selects the physical system with the lowest VSL as the source host to select the next migration candidate. Although this selection might not provide the optimal solution for the problem, it is a reasonable starting point that will find VMs that are hurting the overall server power consumption in a fast and effective fashion. This enables the model to react quickly to an inefficient VM mapping and rapidly attempt to bring the system to an improved state. This approach also reduces the size of the problem by just concentrating in one subset of all available VMs to migrate.
Once the node with the lowest VSL value is selected as our source node, the rest of the nodes are considered to be the possible target node. The eventual target node is selected based on a prediction of future system state. We evaluate what a predicted $E_{Metric}$ value for the host system would be if VM $v$ ($v \in VM_{source}$) is transferred to $VM_{target}$. This prediction is made by adding the contribution of $VM_{resource}(v_{candidate})$ to the corresponding target system load ($VSL_{target}$) and subtracting it from the load of the source server ($VSL_{source}$). The $v_{candidate}$ migration that provides the greatest improvement to the value of (5.2) for a source without affecting the target’s capacity is selected for promotion. If an improvement in $E_{Metric}$ is detected, $v_{candidate}$ gets promoted and DREM then issues the migration of $v_{candidate}$, otherwise the system is unchanged.

5.3 DREM Behavior under Different Workloads

5.3.1 Workload Characteristics

Enterprise servers run varied types of applications, including databases, web servers and email servers. Applications behave differently in terms of resource usage, so it is important to evaluate our algorithm in varied workload scenarios. We present experiments when running online transaction processing (OLTP), web server and email server workloads. These three workloads present a representative sample of current workloads in enterprise servers.

The implementation of the OLTP workload is the same used for validating DREM’s VIBM algorithm for load balancing. It follows the TPC-C specification [58], which models an online wholesale supplier managing orders. Order-entry provides a conceptual model for the benchmark, with the underlying components being typical of any
OLTP system. The issued transactions operate against a relational database composed of 9 tables defined in the specification. Transactions generate reads, writes, and rollbacks. The application uses primary and secondary key access. Our OLTP workload consists of warehouses, each containing 10 terminals, as outlined in the TPC-C specification. The terminal interface consists of a Linux terminal that displays information about the current transaction being performed.

The web server application we use is the Apache webserver [59]. Apache is one of the most used web servers in the world. We installed version 2.2.12 of the Apache webserver on the VMs. The web page requests were performed from a remote client machine using the ab command line. The following options are used: -c 5 (for 5 simultaneous requests) -k (to perform multiple requests in a session) and -n 11400000 (for 11,400,000 requests).

The email server is a basic smtp server implementation using gnu mailutils [60]. We will be sending text emails with image attachments (total of 300KB per email) to the VMs and measure the amount of emails we can send in the system evaluation time interval.

Table 5.1: VM name and type. VM Small configurations have id 1 thru 9 while VM Tiny types run from 10 to 18.

<table>
<thead>
<tr>
<th>VM Small</th>
<th>VM Tiny</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM1 VM2 VM3</td>
<td>VM10 VM11 VM12</td>
</tr>
<tr>
<td>VM4 VM5 VM6</td>
<td>VM13 VM14 VM15</td>
</tr>
<tr>
<td>VM7 VM8 VM9</td>
<td>VM16 VM17 VM18</td>
</tr>
</tbody>
</table>
CHAPTER 5. USING DREM TO REDUCE POWER CONSUMPTION

Figure 5.2: Experimental setup for the power savings, (a) DREM will issue VM migrations following DREM’s VPSM model; VM initial placement for the different experiments and number of servers used, (b) 6 VM, 8VM and 10 VM experiments use 3 servers and (c) 14 VM, 16 VM and 18 VM experiments use 6 servers.

5.3.2 Testbed System

The experimental setup of our DREM data center testbed for the validation of VPSM is shown in Figure 5.2 (a). In our experimental setup, the storage arrays are shared
between all hosts and VM disk file locations are fixed, in other words, there is no migration of VM storage media. All VMs also share the same network infrastructure. The focus of this work is efficient resource usage, thus we concentrate on the efficient utilization of all available resources rather than considering other server consolidation techniques such as powering down servers. Our experiments are designed to compare the effectiveness of our approach to improve resource efficiency. We present results for the following migration policies: 1) no migrations, 2) migrations issued by our model. Both policies will be compared in terms of the performance obtained by workload execution and consumed power.

The evaluations are preformed on a system that contains up to 18 VMs. As with the VIBM experiments, we configure two different VM sizes (in terms of virtual resources; CPUs and memory), The VM Small configuration consists of two virtual CPUs and 1GB of main memory. The VM Tiny has one virtual CPU and 512MB of main memory. Both VM configurations have 20GB virtual hard disks. A complete description of the VM types is provided in Figure 3.4. Table 4.2 summarizes the VM names used.

In order to test the utility of our model to increase overall system balance, we have conducted a set of tests with 6, 8, and 10 VMs using half of the resources (3 servers), and using all available resources (6 servers) for 14, 16 and 18 VMs. The active servers for each experiment and the VM initial placements are depicted in Figure 5.2 (b) and (c). These scenarios are aimed at testing and understanding what the system power consumption improvements and performance degradation would be against the best case scenario for performance (balanced system with no migrations) throughout different workload sizes.

The following sections will explain in more detail our experiments.
Figure 5.3: A comparison of measured power consumption over time per workload for: (a) 6 VM, (b) 8 VM, (c) 10 VM, (d) 14 VM, (e) 16 VM, and (f) 18 VM experiments. DREM is able to reduce consumed power for all experiments.
First, we would like to understand the potential savings in power consumption that DREM offers. For this purpose, we collect power consumption information throughout the experiments. The experiments include both resources configurations: 3-servers and 6-servers. We begin by letting the system warm-up for 25 minutes prior to starting the VM migration routines. After this warm-up period, the system runs for one hour while the VMs are being migrated following DREM recommendations.

### Power Savings

First, we would like to understand the potential savings in power consumption that DREM offers. For this purpose, we collect power consumption information throughout the experiments. The experiments include both resources configurations: 3-servers and 6-servers. We begin by letting the system warm-up for 25 minutes prior to starting the VM migration routines. After this warm-up period, the system runs for one hour while the VMs are being migrated following DREM recommendations.
Figure 5.5: A comparison of achieved performance, total server power consumption and energy efficiency over time per used workload for the 3 servers configuration: (a) Apache, (b) Email, and (c) OLTP workloads. Note how the performance degradation (left) and the power savings (middle) results align to produce a net scaling back effect that yields comparable energy efficiency values (right).

These results are then compared to a *no-migration* scheme.

Figure 5.3 illustrates power consumption results over the experimental time for all server configurations. We can observe how DREM migrations are able to reduce power consumption over the measured interval for all cases with the Apache workload.
Figure 5.6: A comparison of achieved performance, total server power consumption and energy efficiency over time per used workload for the 3 servers configuration: (a) Apache, (b) Email, and (c) OLTP workloads. Note how the scaling back effect is maintained in this larger working set.

benefiting the most, followed by the OLTP workload and then the Email workload, which was the one that showed the least benefits.

Figure 5.4(a) shows the obtained energy savings for the 3-servers configuration.
We observe savings between 7% and 9% on power consumption in the Apache workload. The email workload shows a 3% savings and the OLTP savings range from 5% to 8%. In the case of the 6-servers configuration (Figure 5.4(b), Apache savings are from 4% to 6%, Email from 2% to 3% and OLTP from 3% to 4%. These results suggest that DREM can find migrations that can provide power savings in the datacenter.

Performance Results

We would also like to analyze DREM in terms of performance. Ideally, we are interested in performing VM migrations that lower power, but maintained performance. Figure 5.5 (left) presents performance results when running the Apache (a), Email (b) and OLTP (c) workloads in the 3-servers configuration. In the 6-VM experiment, DREM obtains a 3%, 5% and 9% decrease in Apache, Email and OLTP performance respectively when compared to the base case. The 8-VM results show a performance drop of 9%, 2% and 10%. The 10-VM experiments show a degradation in performance of 8%, 4% and 11%. Figure 5.6 (left) shows the same information for the 6-servers configuration. When observing the performance results for the 14-VM, 16- and 18-VMs experiments, we notice that the degradation in performance for Apache and Email is much less than in the 3-servers scenarios. Given a larger optimization space, we have a much better chance of distributing workloads while maintaining performance. The OLTP workload is the only exception, with a performance drip of between 5% and 6%. These results also suggest that we can consolidate network-dominated workloads and still reduce power consumption without significant impact on performance. In the case of the OLTP workload, although there is high I/O usage, the CPU and memory usage is higher than the other workloads and thus we pay some degradation for consolidating VMs when the number of servers in small.
Power and Performance

After analyzing power and performance impacts separately, we move to study the combined effects on the system. Figure 5.5 and Figurefig:6servers (right) provide information on the power consumed versus the application throughput. As shown in the figures, we obtain similar results to the base case, with differences of less than 2%. These results suggest that DREM’s impact on performance is compensated by the reduction on power consumption, keeping server efficiency comparable to the original system. Thus DREM is working as an energy savings mechanism that scales down performance in a thoughtful way.

5.4 Summary

In this chapter we presented a power savings VM migration model (VPSM) to enhance DREM capabilities. VPSM provides for energy savings by consolidating VMs to a smaller number of systems, keeping the rest of the servers running in an idle state. It achieved these power savings by relying in a power efficiency metric \( E_{Metric} \) based on VSL. With these results, we demonstrated that VSL is robust enough to be able to provide the grounds to not only a load balancing resource management strategy, but an effective energy savings scheme as well.

Our results suggest that we can obtain power savings of up to 9%. We have also shown that DREM is able to provide these savings without significantly impacting performance. Thus the power/performance ratio remains very close to the base case, which indicates we can perform workload power/performance scaling when we want to save energy.
Chapter 6

Understanding VSL and its derived Metrics

In the previous chapters of this thesis, we have shown how VSL can be effectively used to create load balancing and power savings VM migration policies. In this chapter, we will analyze the components of VSL and their impact on the overall metric. If we can quantify the effect that different components have on the range of workloads studied, we may be able to identify which components benefit a particular workload and identify which resource is the bottleneck in the virtualization infrastructure. This insight can help future work to enhance the Dynamic Resource Efficiency Manager.

6.1 VSL Components Analysis

VSL is composed of different system resources $R = \{CPU, memory, disk, network\}$ and is a function of the resource consumption of the resident VMs present on the physical server. In the implementation of VIBM and VPSM we demonstrated that
VSL could effectively be used to build efficient multiple target efficiency policies such as load balancing and power savings methods.

Let us analyze how VSL would behave in a real scenario. In this thesis we work with a subset \( R' \subseteq R \) of resources: \( R' = \{ CPU, memory \} \), with equally assigned resource weights \( W_{CPU} = W_{memory} \), therefore the maximum VSL value per physical server is 2 and occurs when all resources in the physical server are being utilized. Ideally, we would like to operate the data center at a lower level of usage since a regular load of that size indicates underprovisioning and the need to augment available physical resources. Thus, another effective use of VSL is as a tool to measure and evaluate provisioning in the data center.

### 6.1.1 Description

We evaluate the VSL values measured during the VPSM VM migration experiments, paying particular attention to the CPU and memory components and how they combine to form the VSL value for each physical host. Figure 6.1 illustrates the experimental setup and VM initial placement scenario. The analysis is performed on the 3 Server configuration with 10 VMs and the 6 Server configuration with 18 VMs. The 3 Server analysis allows us to understand better the VM migration pattern observed when using VPSM in a smaller subsystems. Thus we can quantify the impact at a lower scale and identify possible differences due to the size of the workload. The 6 Server configuration will highlight the maximum capacity of our DREM virtualized data center testbed and will provide insight on the scalability of our approach. In the following sections we analyze the observed metric values on each commercial workload execution.
Figure 6.1: Experimental setup for the analysis of VSL and related metrics, (a) DREM will issue VM migrations following DREM’s VPSM model; VM initial placement for the different experiments and number of servers used, (b) 10 VM experiments use 3 servers and 18 VM experiments use 6 servers.

6.1.2 Apache Webserver

The Apache Webserver workload simulates a website servicing web page requests. The web page requests were performed from a remote client machine using the Apache Benchmark tool (ab command line). The workload is generated by running the ab program using the following options: -c 5 (provides 5 simultaneous requests) -k (perform multiple requests in single session) and -n 11400000 (total number of requests). Figure 6.2 shows results for the 10VM 3 Server experiments. Part (a) illustrates the behavior of the different VSL values. There is a reduction over time in the VSL values of the source (VSL3) physical server until reaching a zero value when all VMs have
been migrated and the host has entered an idle state. We can notice and increase in the VSL value of the target servers (VSL1, VSL2) as VMs are migrated into them.

Figure 6.2(b) displays the behavior of the VSL based metrics during the execution of the workload. We can observe how $E_{Metric}$ is reduced over time. We also observe how the reduction in $E_{Metric}$ causes an increase in $I_{Metric}$ which at the initial stage reflected a low value corresponding to a well balanced workload in the system. As the workload execution progresses and DREM issues VM migrations using VPSM,
Figure 6.3: 6 Servers 18VM Apache workload metric analysis: (a) $I_{Metric}$ and its components, (b) $E_{Metric}$ vs power, and (c) $I_{Metric}$ vs power.

$I_{Metric}$ continues to increase as $E_{Metric}$ decreases and continues to reduce power consumption.

Figures 6.2(c) and (d) illustrate the relationship that exists between the metric and consumed power for $E_{Metric}$ and $I_{Metric}$ respectively. As it is shown, $E_{Metric}$ is directly proportional to power consumption, while $I_{Metric}$ is indirectly proportional to power consumption.

Figure 6.3 presents results for the 6 Server DREM testbed configuration using 18 VMs. This set should complement the previous findings and help us understand
the behavior of the system in a broader scale. We can observe an increment in the CPU component of VSL throughout the data center due to the increase in workload size. In this case, the VPSM method identifies servers VSL3 and VSL5 as the source hosts. It is interesting to observe that DREM detects both servers to have the least VSL values and correctly starts issuing VM migrations until both systems are left in a lower power consumption state.

Figures 6.3 (b) provides consistent results for this workload. A reduction in $E_{Metric}$ causes an increase in $I_{Metric}$. We can notice that at time interval 6, the reduction in $\mu_L$, causes a reduction in $\sigma_L$. In this particular instance, we observe a slight decrease in $I_{Metric}$, but this behavior is not representative of the total execution behavior. Figures 6.3 (c) and (d) show the expected behavior: with a reduction in $E_{Metric}$ we obtain reduction in power consumption while a reduction in $I_{Metric}$ causes an increase in power consumption.

### 6.1.3 SMTP Email Server

The email workload consists of a SMTP server linux implementation using gnu mailutils [60]. The sent email size is 300KB and consists of text with an image attachment. Emails are sent to the VMs continuously during the workload execution. Figure 6.4 shows results for the 10VM 3 Server experiments. The behavior of the different VSL values is presented in Figure 6.4 (a). As observed in the Apache workload, there is a reduction over time in the VSL values of the source (VSL3) physical server until all VMs have been migrated and the host has entered an idle state. The VSL value of the target servers (VSL1, VSL2) increase as the other VMs are migrated to them. This workload captures the impact of VM migrations between these servers in the measured VSL values, but the system is able to identify power savings benefits at a
Figure 6.5 presents results for the 6 Servers using 18 VMs. We can observe an increment in the CPU component of VSL values that can be attributed to the increased workload size. In this case, the VPSM method identifies servers VSL5 and VSL6 as the source hosts. The reduction in power consumption is due to moving the
workload out of these two servers and keeping them idle. We observe that contrary to the 10 VMs experiments in which the source host was emptied at time interval 7, DREM manages to empty both source hosts at time interval 3.

Results in Figure 6.5 (b) shows behavior of the VSL based metrics. As expected, a reduction in $E_{\text{Metric}}$ causes an increase in $I_{\text{Metric}}$. We observe that from time interval 4 to 5, there is a slight increase in the value of both $\mu_L$ and $\sigma_L$. This is due to the changes in hardware resource sharing between the resident VMs and the recently migrated ones. Figures 6.5 (c) and (d) confirm that a lower $E_{\text{Metric}}$ value
results in the lower power consumption state while a reduction in $I_{Metric}$ increase the consumed power.

### 6.1.4 OLTP Workload

The OLTP workload follows the TPC-C specification [58], which models an online wholesale supplier managing orders by issuing transactions to a relational database
composed of 9 tables. These transactions generate database reads, writes, and rollbacks. The workload consists of warehouses, each containing 10 terminals. The terminal interface consists of a Linux terminal that displays information about the current transaction.

During the 10 VM OLTP workload results (illustrated in Figure 6.6), there is considerable activity in both CPU and memory VSL components. We can additionally observe how migrations start early in execution (as early as time interval 2), emptying VSL3 and even in the latter stages (time interval 7) VPSM detects power savings benefits to issue migrations to the heaviest loaded server: VSL1. These results clearly demonstrate the adaptive behavior of DREM, reacting to the current state of the system. This behavior is also reflected in Figure 6.6 (b), where starting at time interval 6 we see further reduction of $E_{Metric}$. We can see that DREM is migrating all VMs to VSL1, moving VSL2 to a low power consumption state.

Results for the 6 Servers using 18 VMs are presented in Figure 6.5. The VPSM method identifies servers VSL3 and VSL5 as the source hosts. We observe that similar to the 10 VM experiments at time interval 7, DREM detects potential power savings migrations and issues them. During the last stages DREM is reducing the available physical resources to close to 65% by almost having two servers in idle state and only one servicing most of the VMs.

Results in Figure 6.5 (b) shows behavior of the VSL based metrics. As expected, a reduction in $E_{Metric}$ causes an increase in $I_{Metric}$. We observe that from time interval 4 to 5, there is a large reduction in the value of $\mu_L$. This is an effect of DREM completely emptying VSL3 and having now two servers in low power consumption state. It is interesting to note that the additional migrations that occur after time interval 6 cause a sudden increase in $\sigma_L$ and consequently in $I_{Metric}$ at time interval
9. Figures 6.5 (c) and (d) are consistent with our previous results: $E_{Metric}$ values are directly proportional to the lower power consumption state while being inversely proportional to $I_{Metric}$ values.

### 6.1.5 Discussion

We observe common behavior on all workloads that fully illustrate our problem formulation: lowering $E_{Metric}$ reduces power consumption and degrades performance
(increases the value of $I_{Metric}$).

In general, the Apache and OLTP workloads are more CPU dominant than the email workload. One interesting point is that the memory component dominates the total $VSL$ value for all workloads. In fact, in most cases we see a high memory usage behavior (component close to 1). We can use this information to better understand workload behavior and the needs of applications in a typical virtualized infrastructure, leading to improvements in DREM’s management capabilities.

### 6.2 Potential for Other Resource Efficiency Targets

In this thesis we have presented the design of a new virtualized data center resource manager. We have demonstrated the effectiveness of this manager in load balancing and reducing power consumption when driving live VM migrations. DREM is able to improve its efficiency targets using the $VSL$ metric presented in this work. However, there are still other opportunities to improve resource usage in the data center. In this section we explore the potential on using $VSL$ to address other resource efficiency objectives in the data center. In particular addressing management of the storage infrastructure and achieving multi-objective DREM behavior.

#### 6.2.1 Storage Management

Storage systems account for a large portion of the investment in enterprise data centers making them a vital component of them. The storage infrastructure tends to be several orders of magnitude slower than processor and memory speeds. Therefore, management of storage is a constant problem and a potential source of opportunity
for increasing efficiency in the data center. In order to service the workload’s storage requirements, storage systems employ the use of resource consolidation, fast response time, high availability, and data sharing.

The inclusion of a storage component into VSL could enable DREM to effectively use this information to improve overall data center resource efficiency. Figure 6.8 presents data for the previous 18 VM VSL behavior with the storage usage component added to the metric. As we can see, there is a high use of storage when using VPSM. This is expected as we have observed in the results that memory is a heavily used resource among VMs and consolidation of VMs in a smaller server base would require the need of swapping to share the memory resources. If we integrate this component into VSL, both efficiency target studied could benefit. For example, VIBM could use this extra information to enhance its migration decisions. In the case that there are VMs using heavily one storage system, and another storage array is present and underutilized, perhaps it would make sense to keep these VMs in separate storage arrays to obtain better disk response times.

The storage infrastructure is accounted for a large portion of energy consumption among the various components of a data center[61]. This problem is augmented by the continuous availability of faster disks with higher power needs as well as the increasing replacement of tape backups by disk backups to increase backup performance. With the added disk component into VSL, VPSM could be extended to not only capture expected server power consumption, but be storage-aware as well and issue migrations to extend the idle period of underutilized storage infrastructure to save energy consumption on other component of the data center.
Figure 6.8: 6 Server 18VM VSL values: (a) Apache workload VSL used in this thesis, (b) Apache workload VSL with storage component, (c) Email workload VSL used in this thesis, (d) Email workload VSL with storage component, (e) OLTP workload VSL used in this thesis, and (f) OLTP workload VSL with storage component.
6.2.2 A Power/Performance Unified Solution

Our experiments illustrate the effectiveness of $I_{Metric}$ to capture imbalance through the virtualized data center. This capability enables the inclusion of this metric during load balancing. Similarly, $E_{Metric}$ has been successfully used to issue VM migrations that improve overall server power consumption. Considering the fact that $E_{Metric}$ is a component of $I_{Metric}$, the use of the latter can be extended to be used as a multi-purpose metric for optimizing power/performance. For example, DREM could be adjusted to use the current version of the VIBM module during heavily loaded periods in the day, and whenever we detect a sustained decrease in the load of the system, adaptively switch to DREM’s VPSM module for power savings. In the case where an application can run at lower than peak speeds, we can adjust the module transition phases to account for these applications and keep the data center objective as a combination of both performance and power savings.

6.2.3 Summary

In this chapter, we further examined the ability of VSL to capture the load of a virtualized server by understanding the behavior of its individual components. In particular, we analyzed VSL component behavior when running different workloads. Our results illustrate changes in CPU and memory requirements during workload execution and how these changes are tied to changes in the VSL value. We also observe how changes in VSL values affect our derived metric values. In the case of $I_{Metric}$, we observed how the proposed migrations increase the value of this metric, signaling the loss of balanced resource sharing and thus a drop in achieved performance. Additionally, our results help clarify the power consumption vs performance tradeoff present in computer systems. An improvement in the $I_{Metric}$’s value should
yield higher performance at the expense of greater power consumption. Alternatively, VM migrations that cause a reduction in $E_{Metric}$ prove to be effective at reducing server power consumption in the data center with an associated drop in overall system performance.

The potential use of VSL to help address other efficiency targets in the data center has been also studied. Specifically, we observe how including more resources into the VSL metric can change its measured behavior. This change in behavior can be used to include more system awareness to DREM. We showed that if we included disk information, that both VIBM and VPSM would be more storage-aware when attempting to improve load balance or power consumption. Finally, we also have presented the potential to extend DREM to consider multi-objective functionality. This can be accomplished given the relation between $E_{Metric}$ and $I_{Metric}$. 
Chapter 7

Summary and Conclusions

In this thesis, we have introduced, developed, and analyzed DREM: a novel method to dynamically issue VM migrations for efficient resource management in a virtualized data center with multiple efficiency objectives, specifically load balancing and power savings. Our goal was to provide a method which could exploit the abstractions present in modern virtualized computer systems (e.g., virtualized resources) to expand the scope of adaptive VM resource sharing. The hope is that this framework will ultimately yield substantial benefits for overall system resource management.

7.1 Summary of Research

In this section, we summarize the research contributions of this thesis, with a focus on the novel concepts and ideas that we have introduced, developed, and analyzed.
7.1.1 The Dynamic Resource Efficiency Manager

The major novel contribution of this research is the introduction, development and evaluation of the Dynamic Resource Efficiency Manager. The resource efficiency manager extends previous work, primarily the use of live VM migrations to enhance efficiency in a virtualized data center. Our work surpasses prior work by providing a single management infrastructure that is capable of handling multiple efficiency objectives such as load balancing and power savings.

As part of this thesis we have developed the necessary infrastructure to deploy a DREM-managed virtualized data center. Additionally, we have presented new metrics and terminology that underline the different resource efficiency objectives. For example we defined the concept of the \( VM \text{ Resource Efficiency} (VM_{resource}) \) as the amount of resource usage over the host resource capacity that a particular VM is consuming. This quantity allows us to measure per-VM resource consumption in an effective fashion, and this metric can later be used to evaluate overall system resource usage.

7.1.2 The Virtualized Server Load

In this thesis we introduced a new virtualized server load metric that is based on the current resident \( VM_{Host} \) set resource usage named the \( \text{Virtualized Server Load} \). \( VSL \) is the sum of each VM’s \( VM_{resource} \) (resident on the physical host) for all resources available on the physical system. This metric is the central component of DREM. DREM communicates with the virtualization manager of the system and collects per-VM resource usage information and computes \( VSL \). It then uses this metric to measure the load on each virtualized server and decides whether or not to start migrating VMs.
The Virtualized Server Load is one of the most useful components developed in this thesis, primarily because it can be used to make decisions about the system during application execution. It is possible to measure the VSL of a physical system and quickly understand the provisioning on that host. We demonstrated that the VSL of a possible target host for a VM migration can be accurately predicted by adding the candidate VM resource usage to that host’s current VSL value.

### 7.1.3 The VSL based Imbalance Metric

The first efficiency objective developed in this thesis is a solution for the load balancing problem for a virtualized data center. A metric that measures load imbalance throughout the data center is defined. $I_{Metric}$ captures variability in the VSL metrics for all servers in the data center. If configured for load balancing, whenever DREM detects a high imbalance value, it triggers the load imbalance module named VIBM. VIBM load balances the data center by iteratively finding VM migrations that reduce $I_{Metric}$. We demonstrated that the VIBM module is capable of finding VM migrations that load balance a virtualized data center and improve overall workload performance.

### 7.1.4 The VSL based Power Consumption Metric

Another efficiency objective developed in this thesis is a framework for power savings in the virtualized data center. A VSL-based metric that measures average resource usage is presented for power consumption efficiency. $E_{Metric}$ is the average VSL value for all servers in the data center. Since resource usage in the data center directly determines the amount of power the servers are consuming, this metric can be used to estimate overall consumption. If configured for power savings, whenever
DREM detects a high $E_{Metric}$ value, it triggers the power savings module named VPSM. VPSM finds VM migrations that reduce $E_{Metric}$ in the data center.

We demonstrated that the VPSM module is capable of finding VM migrations that reduce power consumption. We also showed that VPSM slightly impacts performance in order to save power. It accomplishes this task by consolidating VMs in a subset of the available servers and keeping the rest in a lower power consumption state (idle mode).

### 7.1.5 The DREM System Implementation

As part of this thesis, we have constructed a virtualized data center testbed managed by DREM. The system was implemented using VMware infrastructure tools. This approach allow us to use as a baseline for VIBM, our load balancing module an available commercial product in this infrastructure: VMware DRS.

In the VMware DRS – VIBM comparison, we demonstrated that we can generate a VM migration pattern that reduced $I_{Metric}$ the most. Our results also showed that by accurately predicting the imbalance metric to guide VM migrations can result in higher workload throughput, increasing performance by 2-5% over VMware’s DRS.

Although the current prototype uses the VMware API, there are other hypervisors that provide similar capabilities for DREM to extract the necessary information to compute $VSL$. For example, the Xen hypervisor provides the Xenoprof API [54, 55, 56] that is robust enough to provide the same information to the DREM. Thus, a DREM implementation that uses the Xen hypervisor can be achieved with minor changes to its core structure.
7.2 Discussion

In this section, we discuss limitations of the work presented, as well as potential future extensions to DREM. This is helpful in order to give the reader some perspective of the potential future work that could be derived from this thesis.

7.2.1 Limitations

As discussed in Chapter 3, the implementation of the DREM testbed included storage infrastructure that was located in a different facility than the virtualized servers. This limited the scope of the experiments we could perform in the storage layer. In this thesis, we presented our system evaluation methodology without including network and disk traffic information. However, this limitation in our implementation was has no bearing on how DREM could utilize this information for load balancing and power savings. As we stated earlier, a simple modification to have a dedicated connection to the storage infrastructure would solve the saturation of the network traffic and would enable the introduction of both elements to the VSL metric.

7.2.2 DREM Future Extensions

To give the reader a general sense of the potential extensions of this work, we discuss three possible additions that would enhance the functionality provided by DREM. These additions result from the progress in the area of virtualized data centers that is currently experiencing a rapid rate of acceptance and deployment.
CHAPTER 7. SUMMARY AND CONCLUSIONS

Using VSL in the Cloud

Perhaps the most significant contributor to the rapid adoption of virtualized data centers is Cloud Computing services. For example, large Internet-based companies such as Google, Apple, Amazon and Microsoft are deploying cloud infrastructures to provide users with centralized content that could be easily accessed on multiple devices such as computers, smartphones, tablets and home entertainment systems. With multiple data centers located in different geographical parts, cloud computing virtualization infrastructure poses challenges in terms of effectively sharing the available resources. In order to be successful, the VM management techniques must be lightweight and provide mechanisms to move data fast. We believe this to be a fertile area for future study, and one in which the potential benefits of VSL could be exploited, since it has already proven to be well designed to drive multi-objective resource management.

Extend DREM to Handle the Storage Infrastructure

Storage Infrastructures such as Storage Area Networks (SAN) or Network Attached Storage (NAS) have become widely used solutions for high performance enterprise storage. However, these infrastructures are very complex and varied among different vendors. It is important to understand the behavior of the underlying storage infrastructure in order of enhancing performance of the applications that run on them. VSL can capture disk I/O information, and we believe this information could pave the way to extend DREM to manage future storage infrastructure supporting the virtualized data center.
Increased Power Savings

This work provided energy savings by consolidating VMs to a smaller server base and keeping the rest of the servers in an idle state. A natural extension of this work would be to enhance DREM by reducing idle power consumption. Possible solutions include shutting down the servers completely. In this case in particular, we need to account for rapid workload changes that may require availability of hardware resources. Thus performance constraints may limit the use of a complete shutdown. Other possible mechanisms used to reduce idle server power consumption are dynamic voltage and frequency scaling for the internal server components. In this case, we could reduce consumption considerably without having to pay a high cost to bring servers back online to respond to a change in workload behavior.

As we saw in the previous chapter, power savings can also be obtained by better management of storage. The extension of VSL to include disk information could enable DREM to not only capture expected server power consumption, but make decisions accounting for the energy consumption of the storage infrastructure. This effectively would give DREM the potential to save more power consumption by considering other components in the data center.
Bibliography


ACM SIGOPS symposium on Operating systems principles, pages 265–278, New York, NY, USA, 2007. ACM.


