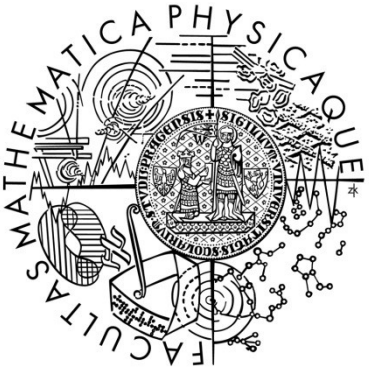
Charles University in Prague

Faculty of Mathematics and Physics

MASTER THESIS



Michael Grafnetter

Dynamic resource balancing in virtualization clusters

Department of Software Engineering

Supervisor of the master thesis: RNDr. Jakub Yaghob, Ph.D.

Study programme: Informatics

Specialization: Software Systems

Prague 2012

I would like to thank my supervisor RNDr. Jakub Yaghob, Ph.D. for providing both consultations and administrative access to the departmental virtualization infrastructure. I would also like to thank my family for tremendous support.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague, August 5, 2011 signature

Název práce: Dynamické vyvažování zátěže ve virtualizačních klastrech

Autor: Michael Grafnetter

Katedra: Katedra softwarového inženýrství

Vedoucí diplomové práce: RNDr. Jakub Yaghob Ph.D.

Abstrakt: Účelem této práce bylo analyzovat problém vyvažování zátěže ve virtualizačních klastrech. Dalším cílem bylo implementovat pilotní verzi aplikace vyvažující zátěž ve virtualizačním klastru postaveném na platformě VMware vSphere Standard. V práci byly také prozkoumány dostupné komerční a open source řešení a prověřena jejich použitelnost a efektivita.

Při návrhu vlastního řešení byla zvolena modifikace hladového algoritmu pro určování, který virtuální stroj má být přemigrován na kterého cílového hostitele. Také byl proveden experiment k zjištění vhodných parametrů pro algoritmus.

Nakonec bylo experimentálně ověřeno, že implementované řešení může být použito k efektivnímu vyvážení zátěže virtualizačních serverů živou migrací virtuálních strojů běžících na těchto hostitelích.

Klíčová slova: virtualizace, vyvažování zátěže, klastr, přidělování prostředků

Title: Dynamic resource balancing in virtualization clusters

Author: Michael Grafnetter

Department: Department of Software Engineering

Supervisor of the master thesis: RNDr. Jakub Yaghob Ph.D.

Abstract: The purpose of this thesis was to analyze the problem of resource load balancing in virtualization clusters. Another aim was to implement a pilot version of resource load balancer for the VMware vSphere Standard-based virtualization cluster. The thesis also inspected available commercial and open source resource load balancers and examined their usability and effectiveness.

While designing the custom solution, a modification of the greedy algorithm has been chosen to be used to determine which virtual machines should be migrated and to select their target hosts. Furthermore, experiments have been conducted to determine some parameters for the algorithm.

Finally, it was experimentally verified that the implemented solution can be applied to effectively balance virtualization server workloads by live migrating virtual machines running on these hosts.

Keywords: virtualization, load balancing, cluster, resource allocation

Contents

Contents **5**

Introduction **1**

Background *1*

Objectives *1*

Outline *2*

Chapter 1: General Concepts **3**

1.1 Virtualization *3*

1.2 Server Virtualization *3*

1.3 Virtualization Cluster *5*

1.4 Live Migration *5*

1.5 Virtualization Infrastructure *5*

1.6 Virtualization at the Department of Software Engineering *6*

Chapter 2: Resource Load Balancing **8**

2.1 Server Load Metric *8*

2.2 System Imbalance Metric *8*

2.3 Reducing System Imbalance *8*

2.4 Other Key Parameters *9*

2.5 Initial Placement *9*

2.6 Affinity Rules *10*

2.7 Power Management *10*

Chapter 3: Commercial and Open-Source Implementations **11**

3.1 Load Balancers in Virtual Infrastructure Management Software *11*

3.2 VMware Distributed Resource Scheduler *11*

3.3 Microsoft Performance and Resource Optimization *18*

3.4 OpenNebula *25*

3.5 Summary *26*

3.6 Feature Requirements *28*

Chapter 4: Solution Approach **29**

4.1 Performance Counters *29*

4.2 Imbalance Metric *29*

4.3 Balancing Algorithm *30*

4.4 Virtualization Platform Selection *31*

4.5 vSphere Plug-in Model *31*

4.6 Application Server *32*

4.7 Libraries Used *32*

4.8 User Interface *33*

4.9 Configuration Options *36*

Chapter 5: Results **37**

5.1 Testing Environment *37*

5.2 Duration of a Live Migration *39*

5.3 Running Live Migrations Concurrently *41*

5.4 Disclaimer *42*

5.6 Internal Structure *43*

5.7 Implementation Issues *44*

Conclusion **46**

Real-World Deployment *46*

Future Work *46*

Bibliography **47**

Attachments **51**

CD *51*

Introduction

Background

Virtualization has undeniably changed the whole IT industry and became a leading technology priority for a large number of companies worldwide [1]. Server virtualization has proven itself to be invaluable also in academia for both educational and research purposes, as it allows, among others, to create very complex network environments on top of much simpler physical infrastructures and to cut costs by server consolidation. A virtual machine (VM) basically encapsulates a complete set of virtual hardware resources, the operating system, and all installed applications into a few files. Therefore, it can be easily moved between virtualization hosts (i.e., physical servers) and even copied to offsite locations, making disaster recovery a straightforward task, so to speak.

While undoubtedly solving many problems, virtualization also brings a couple of new challenges, one of them being workload management. As the virtual machine workload is dynamic in nature and hard to predict in most cases, hardware resources of some virtualization hosts (e.g., processor and memory) become over-utilized in time, while others are under-utilized. This imbalance can be reduced by live migration of VMs between hosts in a virtualization cluster according to their current workload and/or past behavior.

Although some commercial workload management solutions exist, they are mostly targeted at the enterprise and therefore hardly affordable for the academia. Others, though open-source, are highly specialized and thus usable only in very specific scenarios.

Objectives

The purpose of this thesis is to analyze the problem of resource load balancing in virtualization clusters and discuss existing commercial and open source solutions. Based on the discussion, a suitable technique will be selected and implemented in a prototype of the resource load balancer for the VMware vSphere Standard-based virtualization cluster.

Outline

This thesis consists of five chapters and an attachment. Chapters one and two explain the concepts of virtualization and resource load balancing in general. Chapter three provides a comparison of the available commercial and open-source balancers. Chapter four describes the custom load balancer implementation. The actual results are discussed in chapter five. A CD with PDF version of this document is attached.

Chapter 1: General Concepts

In this chapter we will introduce some basic concepts related to the resource load balancing in virtualization clusters, as we will use them in the rest of this thesis.

1.1 Virtualization

The simplest definition of virtualization might be that it is an abstraction layer that decouples physical resources from their users [2]. This abstraction can be applied at several levels within the computing stack:

- Server virtualization
- Desktop virtualization
- Application virtualization
- Storage virtualization
- Network virtualization

The terminology related to virtualization is far from being unified, because much of it originated as marketing terms of commercial virtualization solution vendors rather than originating from academia. Thus, the same basic concept can be called by different names (e.g. Microsoft Live Migration vs. VMware vMotion) and distinctive concepts can be marketed under the same name (e.g. Application Virtualization solutions from Citrix and VMware) [3]. Moreover, as server virtualization is the most common type of virtualization, terms "virtualization" and "server virtualization" are often used interchangeably. Consequently, when using one of these ambiguous terms, we will always clarify what we are meaning by it.

1.2 Server Virtualization

Server virtualization is a method of running multiple independent virtual operating systems on a single physical server. It was first introduced by IBM in the 1960s with System/360. Recently, with increased computing capacity of the low-end machines, similar capabilities are now available for the x86 platform and server virtualization has become the top technology priority at companies worldwide [4].

Typically, a thin software layer called a hypervisor executes on a physical server (virtualization host) and presents an abstraction of the underlying hardware to host multiple virtual machines (VMs). Each of the VMs executes either unmodified (in case of full virtualization) or slightly modified (in case of para-virtualization) version of the operating system (guest OS) [2]. This difference between running

server operating systems with and without virtualization (bare-metal) is illustrated graphically in Figure 1and Figure 2.

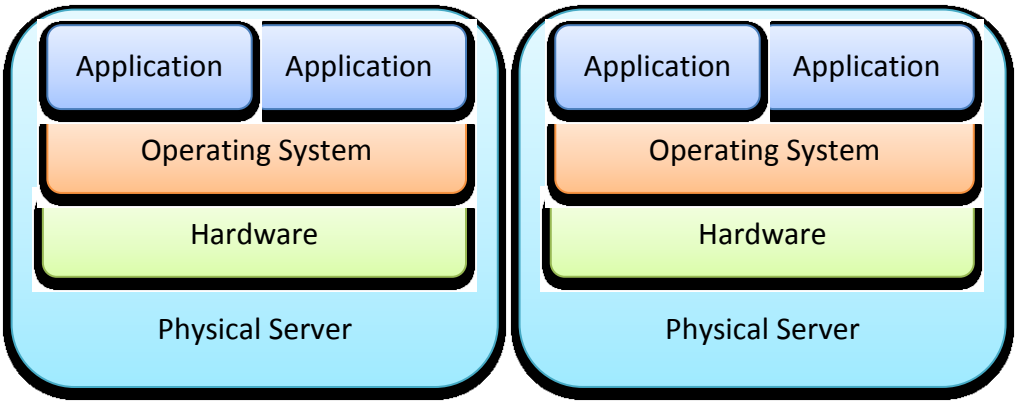


Figure 1: Traditional server concept

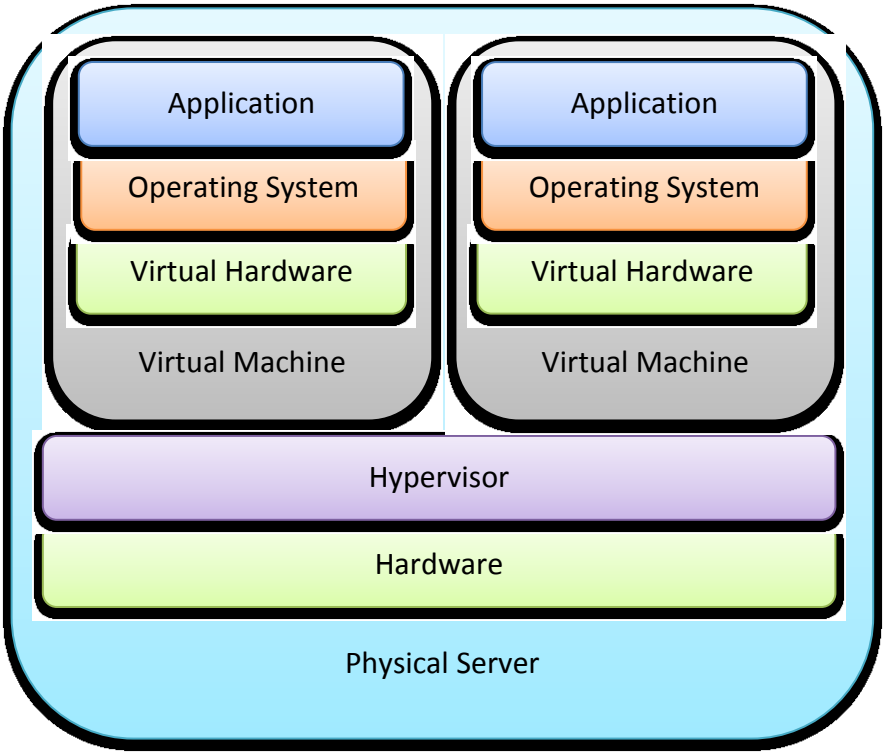


Figure 2: Server Virtualization

Server virtualization provides a way of achieving higher server density to minimize the investment in hardware. However, it does not actually increase total computing power; it decreases it slightly because of overhead. [5]

1.3 Virtualization Cluster

A virtualization cluster is basically a set of tightly coupled virtualization servers (i.e., cluster nodes), that usually share a common CPU architecture, software configuration and some physical resources (e.g. network and storage). Some of the cluster capabilities may include:

- Live migration of VMs between cluster nodes.
- Aggregation of distributed resources (e.g. CPU and memory)
- High availability
- Fault tolerance
- Load balancing

1.4 Live Migration

Live migration is a common feature of virtualization clusters that allows moving virtual machines across distinct physical machines with minimal downtime (~ tens of milliseconds on high-speed networks). It allows a clean separation between hardware and software, and can facilitate fault management, load balancing, and low-level system maintenance [6].

The process of live migration of a virtual machine usually involves these phases [7] [8]:

1. The execution state and active memory of the virtual machine is transferred over the network, while the VM is still running.
2. The virtual machine is suspended.
3. The remaining modified memory pages are duplicated to the destination physical host.
4. The VM is brought online on the destination server.
5. The network switch is pinged to ensure that it is aware of the new physical location of the virtual MAC address.

As a result, both large VM memory size and high VM activity level increase the time it takes to complete a live migration, since more active memory pages have to be transferred between the source and destination cluster nodes.

While Microsoft calls this feature simply as “Live Migration” in its virtualization products, VMware calls it vMotion. To stay consistent, we will call it as “live migration” regardless of vendor.

1.5 Virtualization Infrastructure

A typical virtualization infrastructure consists of these elements:

- Virtualization hosts, e.g. VMware ESXi or Microsoft Hyper-V

- Network infrastructure, e.g. Ethernet switches and routers, both physical and virtual
- Storage area network (SAN), e.g. iSCSI or FibreChannel
- Disk array(s)
- Virtual infrastructure management server(s), e.g. VMware vCenter Server or Microsoft System Center Virtual Machine Manager
- Authentication and authorization server(s) , e.g. Active Directory

Common virtual infrastructure management software features include:

- Centralized control over the entire infrastructure
- Automation of administrative tasks and processes
- Monitoring server performance and health
- Reporting of live and historical data

1.6 Virtualization at the Department of Software Engineering

Server virtualization has been actively used at the Department of Software Engineering at Charles University in Prague for several years, serving both production and educational purposes.

Many core infrastructure servers (e.g. HTTP, SMTP, SNTP, DNS, SVN, etc.) have been migrated from physical to virtual machines and are now requiring just two physical servers to run on.

Virtualization is also heavily used in the "Windows Systems Administration" and "UNIX Administration" courses. During the hands-on labs sessions, students create complex network topologies comprising of hundreds of servers and subnets, so they can learn configuring server operating systems and gain better understanding of how some common network services work. This would be impossible without utilizing virtualization.

As the virtualization infrastructure used for educational purposes consists of four physical servers only, it does not have the capacity to serve hundreds of running VMs. The students are therefore required to power off their own virtual machines when they are not actively using them. As a result, the system is imbalanced most of the time, with one virtualization server running more VMs than the others. Consequently, virtual machines running on this overloaded host are less responsive. This situation would benefit a dynamic load balancer that would migrate some

of these VMs to a less utilized virtualization host. Unfortunately, the departmental virtualization infrastructure is based on VMware vSphere Standard, which lacks any load balancing feature. Upgrading to VMware vSphere Enterprise, which includes VMware Distributed Resource Scheduler (DRS), would increase one-time license costs by $30,080.00 and annual support fees by $5,296.00 and that is unfeasible [9]. These facts have been our main motivation for studying resource load balancing solutions for virtualization clusters.

Chapter 2: Resource Load Balancing

Dynamic live migration of virtual machines can be used to maximize the utilization of system resources in virtualization clusters by constantly balancing server loads, i.e., moving VMs from overloaded physical hosts to those hosts that are underutilized.

Such load balancing can either be done manually by an administrator, or automatically by a load balancer integrated with virtualization infrastructure manager. In this chapter, we will discuss various requirements that are put on such dynamic load balancers.

2.1 Server Load Metric

In order to tell which cluster node is overloaded and which one is underutilized, physical server load needs to be quantified first. A common approach is to measure the relative utilization of its resources (e.g., CPU time, memory utilization, network, disk I/O traffic and other performance counters) [10]. A server load metric is then defined as a weighted sum of these resource utilizations, where weights denote relative importance of the performance counter [11]. Some balancers even use SLA-based metrics, for example server priority level or number of transactions executed per second [12] [13].

2.2 System Imbalance Metric

A system imbalance metric can measure the workload imbalance across the entire cluster. If it is lower than a specified threshold, the system is considered to be balanced. One of the ways the overall system imbalance metric can be defined is the coefficient of variation of individual server loads [14].

2.3 Reducing System Imbalance

The general problem of dynamically allocating VMs to physical servers in order to minimize the system imbalance has been shown to be similar to bin-packing or knapsack problems both classic NP-Hard problems. Nevertheless, highly optimized mixed integer linear programming (MILP) solvers can be employed to effectively load balance smaller virtualization environments [15] [16].

A more common approach involves using a modified version of the greedy algorithm, which is fast and yields satisfactory (though not optimal) results [14] [17].

It works by inductively predicting which VM migration will yield the greatest improvement of the imbalance metric in a particular step.

One of the key aspects is predicting the change in server load metric a migration would cause. Working at the hypervisor level, it is possible to isolate the resource consumption of a particular VM when making predictions.

In many models, future workload is only predicted according the current workload. However, it is possible to monitor the performance metrics and find some regular patterns, like less workload at nights or weekends, and perform proactive migrations based past system behavior. A special case are various lease management systems that schedule batch workloads in advance to maximally utilize hardware resources [18].

2.4 Other Key Parameters

Imbalance Threshold – If the threshold is too high, system will stay imbalanced. If too low, balanced state may never be achieved and thus migrations would never stop, even with stable workloads.

Balancing Frequency – The balancing frequency defines how often the balancer evaluates performance metrics and creates a new migration plan. Typical frequency is 5 minutes [17].

Live Migration Time – The migration process can take up to several minutes, meaning that only a limited number of migrations can be executed in the balancing period. Future migration times may be predicted using historical data. [6]

Migration Overhead - The migration process itself consumes a significant amount of system resources, mainly memory and network I/O. As a result, the decrease in performance caused by frequent migrations can overwhelm the benefits of balancing the system.

Level of Automation – Load balancers can often work in two modes: manual and automatic. In manual mode, administrator has to approve every proposed migration before it is executed.

2.5 Initial Placement

Common feature of load balancers is initial placement of a VM to the least utilized host when the VM is powered on or resumed from suspension. The same algorithm as by load balancing can be used.

2.6 Affinity Rules

Load balancers may support affinity rules that can be used to solve some issues, which may arise when VMs are freely migrated between nodes. Some common business rules are:

VM-VM affinity

A VM-VM affinity rule specifies whether selected individual virtual machines must run on the same host. Typical scenario would be tying a web server and the database server it intensively communicates with together to eliminate network latency

VM-VM anti-affinity

A VM-VM anti-affinity rule specifies whether selected individual virtual machines must be kept on separate hosts. This rule might be used to separate primary and secondary DNS servers. In that case, if a problem occurs with one host, not all DNS servers would be placed at risk.

VM-Host affinity

A VM-Host affinity rule specifies on which hosts a virtual machine can run. As some software vendors still tie their software to physical computers, by forcing the VMs running the software in question to run on specific servers, licensing compliance can be ensured. This type of rule can also be used when a VM utilizes a physical resource that is available only on specific hosts.

2.7 Power Management

With green IT initiatives gaining on popularity, some load balancing solutions have introduced power management features [19]. When virtual machines in a cluster need fewer resources, such as during nights and weekends, some load balancers can consolidate workloads onto fewer servers and powers off the rest to reduce power and cooling costs. When virtual machine resource requirements increase, powered-down hosts are brought back online to ensure service levels are met [3].

Chapter 3: Commercial and Open-Source Implementations

3.1 Load Balancers in Virtual Infrastructure Management Software

According to the latest (June 2010) server virtualization market research conducted by Gartner, the major competitors on the market are VMware and Microsoft [20]. Thus we will focus on their respective solutions in the next sections.

Although both of these companies offer free server virtualization solutions, load balancers are only available in their paid editions [21] [22]. Each company is calling this feature (or set of features) differently:

- VMware Distributed Resource Scheduler (DRS) and Distributed Power Management (DPM)
- Microsoft Performance and Resource Optimization (PRO)

To our knowledge, OpenNebula is the only open source virtual infrastructure manager in production level of development that has at least some limited load balancing capabilities. [18]

3.2 VMware Distributed Resource Scheduler

VMware Distributed Resource Scheduler (DRS), a feature of VMware vCenter Server (virtualization infrastructure management software) with VMware vSphere Enterprise license [9], is considered to be the leading commercially available load balancing solution for virtualization clusters [14]. DRS dynamically balances computing capacity across the cluster and allocates available resources among the virtual machines based on pre-defined rules. When a virtual machine experiences an increased load, VMware DRS automatically allocates additional resources by redistributing virtual machines among the physical servers in the cluster. [23]

VMware DRS continuously collects resource usage information from servers and virtual machines, and periodically generates recommendations to optimize virtual machine allocation (Figure 3). These recommendations can be executed automatically or manually (Figure 4) by performing live migration of virtual machines through VMware VMotion. When a virtual machine is first powered on, VMware DRS either automatically places the virtual machine on the most

appropriate physical server or makes a recommendation (Figure 5) to the administrator.

Both VM-Host and VM-VM affinity and anti-affinity rules are supported by DRS and can be used to fulfill some special performance, availability and licensing requirements. A feature called VMware Distributed Power Management (DPM) can also reduce energy consumption in the datacenter by consolidating workloads and powering off power consuming servers.

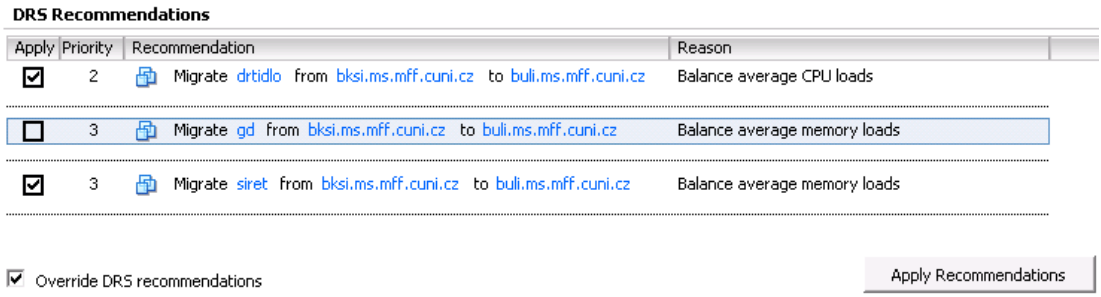


Figure 3: DRS Migration Recommendations

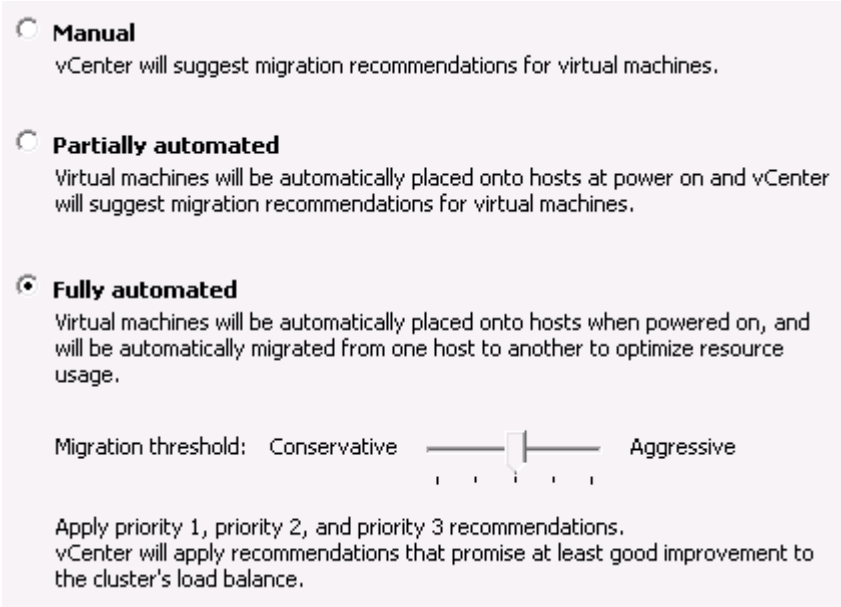


Figure 4: DRS Automation Level Configuration

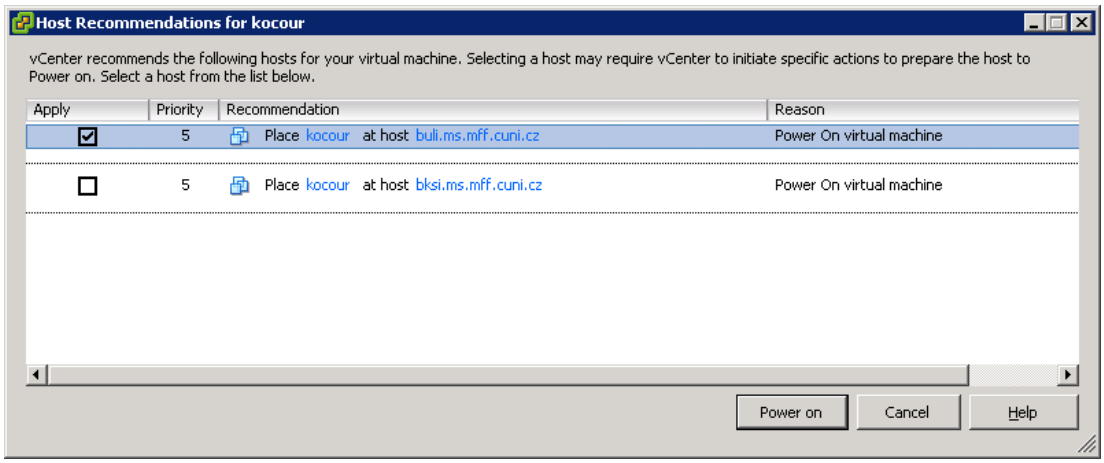


Figure 5: Initial Placement with DRS in Manual Mode

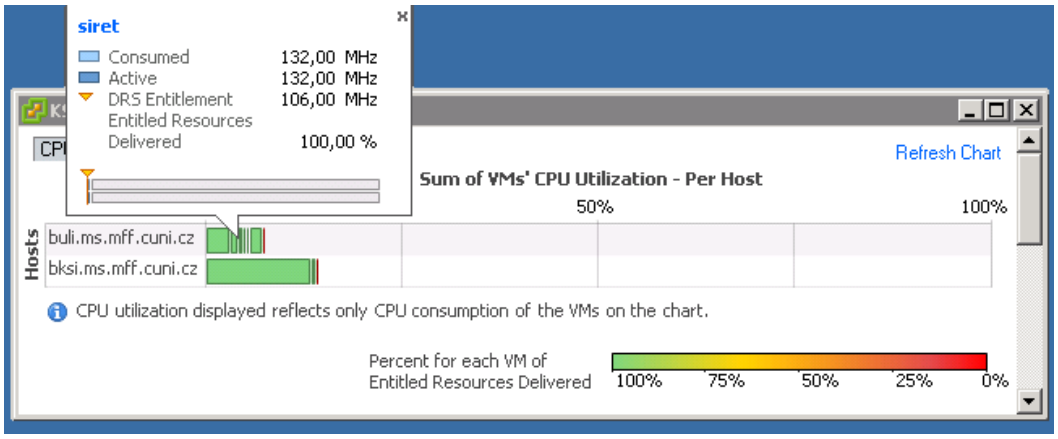


Figure 6: DRS Resource Distribution Chart

Load Balancing Algorithm

VMware DRS evaluates the cluster every 5 minutes. If there's an imbalance in load, it will reorganize the cluster, with the help of vMotion, to create an evenly balanced cluster again [3]. Only limited amount of information about the algorithm is available, mostly from unofficial sources [17] [10] [24] [25].

Figure 7 shows the information about the key DRS parameters displayed in the VMware vSphere Client 4.1 GUI.

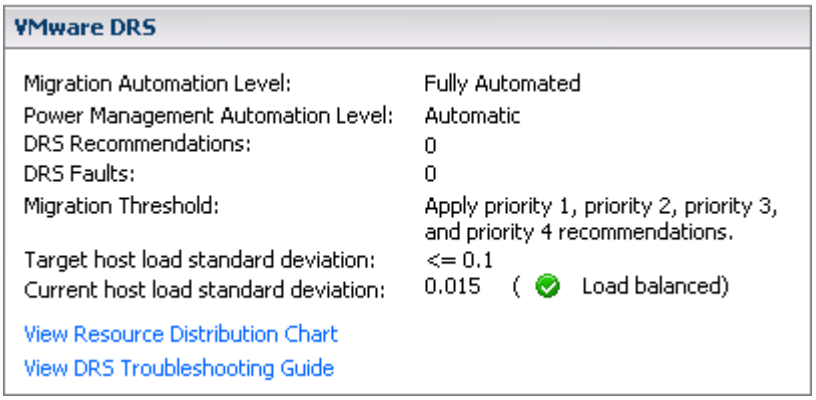


Figure 7: Host Load Standard Deviation

Depending on the Migration Threshold, which can be configured in the Cluster Settings (see Figure 4), and the amount of hosts in the cluster, DRS calculates the "Target host load standard deviation". The exact formula to calculate this value is not publicly available.

Afterwards, the DRS calculates the sum of the resource entitlements of all virtual machines (linear combination of the active CPU and active memory metrics [10]) on a single host and divides that number by the capacity of the host C_j [17]:

$$\forall j \in H: U_j = \frac{\sum_{i \in M} L_i}{C_j}$$

Where L_i is expected load of VM i and U_j is the total load of the host excluding the hypervisor overhead.

The result of all hosts is then used to compute an average and the "Current host load standard deviation". If the environment is imbalanced and the "Current host load standard deviation" exceeds the value of the "Target host load standard deviation" DRS will either recommend migrations or perform migrations automatically depending on the chosen setting (see Figure 4).

The following algorithm is used to form a set of recommendations to correct the imbalanced cluster [17]:

```
while load imbalance metric > threshold {cluster is imbalaced}
begin
      move = GetBestMove
      if move = Nil { No good migration is found }
```

```
            exit
      else
      begin
            Add move to the list of recommendations
            Recompute the "Current host load standard deviation"
      end
end
```

The GetBestMove function is defined as this:

```
function GetBestMove
begin
foreach v in VMs
foreach h Hosts that is not Source Host:
      if h is lightly loaded compared to Source Host:
      if Cost Benefit and Risk Analysis is accepted
      begin
            simulate move v to h
            measure new cluster-wide load imbalance metric as g
      end
GetBestMove := v that gives least cluster-wide imbalance g
end
```

This should result in a migration which gives the most improvement in terms of cluster balance. This is the reason why usually the larger VMs are moved as they will most likely decrease "Current host load standard deviation" the most. If it is not enough to balance the cluster within the given threshold the GetBestMove gets executed again by the procedure which is used to form a set of recommendations.

A constant stream of vMotions is avoided by weighing costs vs. benefits vs. risks. These consist of:

- Costs
    - CPU reserved during migration on the target host
    - Memory consumed by shadow VM during vMotion on the target host

    - VM downtime during the vMotion

- Benefits
    - More resources available on source host due to migration
    - More resources for migrated VM as it moves to a less utilized host
    - Cluster Balance
- Risk Analysis
    - Stable vs. unstable workload of the VM (historic data is used)

Based on these considerations a cost-benefit-risk metric is calculated and if this has an acceptable value the VM will be considered for migration. Again, this metric is proprietary.

Afterwards, every migration recommendation gets a priority rating P_i. This priority rating is based on the Current host load standard deviation σ_i [26]:

$$P_i = 6 - \left\lceil \frac{\sigma_i}{0.1\sqrt{|H|}} \right\rceil$$

This would result in a priority level of 5 for the migration recommendation if the cluster was imbalanced.

There are limits to how many migrations DRS will recommend per interval per ESXi host because there's no advantage to recommending so many migrations that they won't all be completed by the next re-evaluation, by which time demand could have changed. Since ESXi 4.1, the limit on moves per host is dynamic, based on the frequency DRS is invoked and the average migration time observed from previous migrations. [25]

Initial VM Placement

The initial placement of a VM when being powered on is also part of DRS [3]. DRS analyzes the cluster using the algorithm described in the previous section. As no current resource load values for the VM which is being powered on are available, DRS assumes that 100% of the provisioned resources for this VM will be used. If DRS can't guarantee the full 100% of the resources provisioned

for this VM can be used it will vMotion other VMs away so that it can power on this single VM. If however there are not enough resources available it will not power on this VM [17].

Preferential VM-Host Affinity Rules

As mentioned before, DRS supports two kinds of VM-Host affinity rules [27]:

- Must run rules (Mandatory)
- Should run rules (Preferential)

During a DRS invocation, DRS runs the algorithm with preferential rules as mandatory rules and will evaluate the result. If the result contains violations of cluster constraints, such as over-reserving a host or over-utilizing a host leading to 100% CPU or Memory utilization, the preferential rules will be simply dropped and the algorithm is run again. [28]

Planned Functionality

As we have already mentioned, only CPU and memory utilization are used by Distributed Resource Scheduler (DRS) for load balancing VMs across a cluster. The future version of vSphere should introduce a new feature called Storage DRS, which extends the DRS by monitoring disk array space availability, I/O capacity and latency. It will also support affinity and anti-affinity rules to ensure SLAs [27]. The available configuration options are shown on Figure 8.

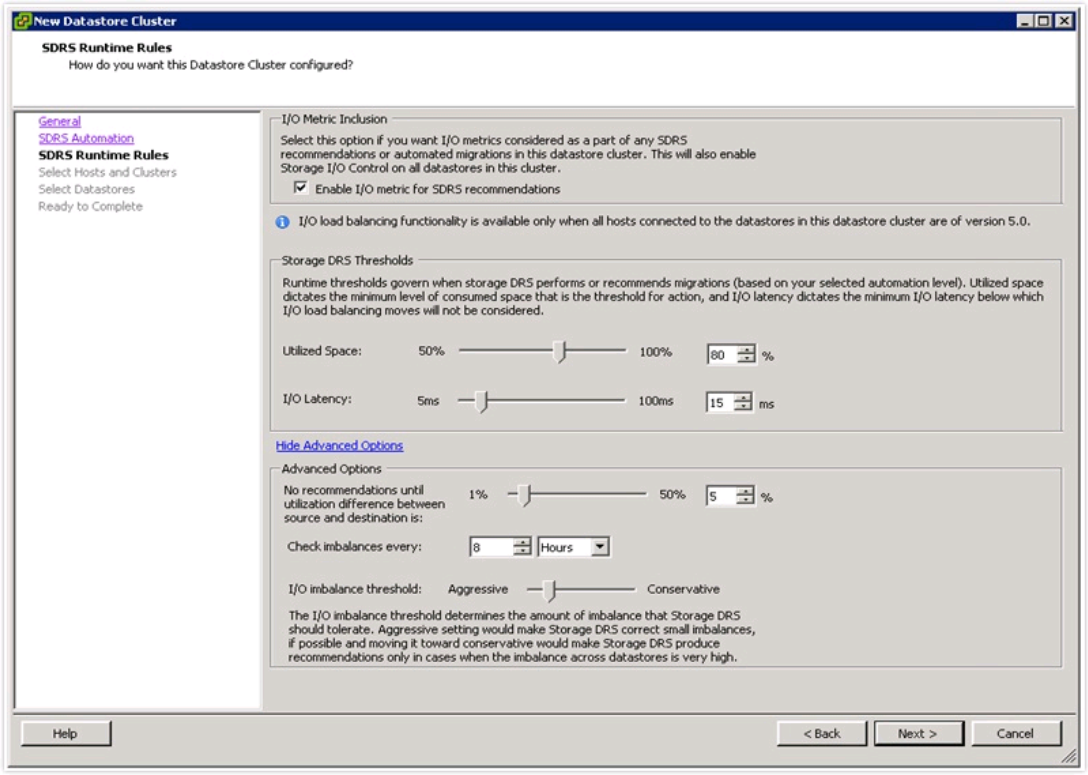


Figure 8: SDRS Runtime Rules

3.3 Microsoft Performance and Resource Optimization

Overview

Performance and Resource Optimization (PRO), a feature of Microsoft System Center Virtual Machine Manager (VMM, virtualization infrastructure management software), uses System Center Operations Manager (SCOM, infrastructure monitoring and reporting software) to monitor the health and availability of the virtual machines and virtual machine hosts that VMM is managing. It ties specific alerts from SCOM to remediation actions in VMM (PRO tips). For example, alerts might occur when specific thresholds are exceeded, such as transactions per second, CPU utilization and e-mail message delivery SLA, or when a hardware failure is detected (e.g. a fan failure). A remediation action often involves migrating virtual machines between physical hosts or altering configuration of virtual machines to improve performance (see Figure 9). VMM administrator can manually approve PRO tips, or the administrator can configure PRO to implement the recommended actions automatically. [29]

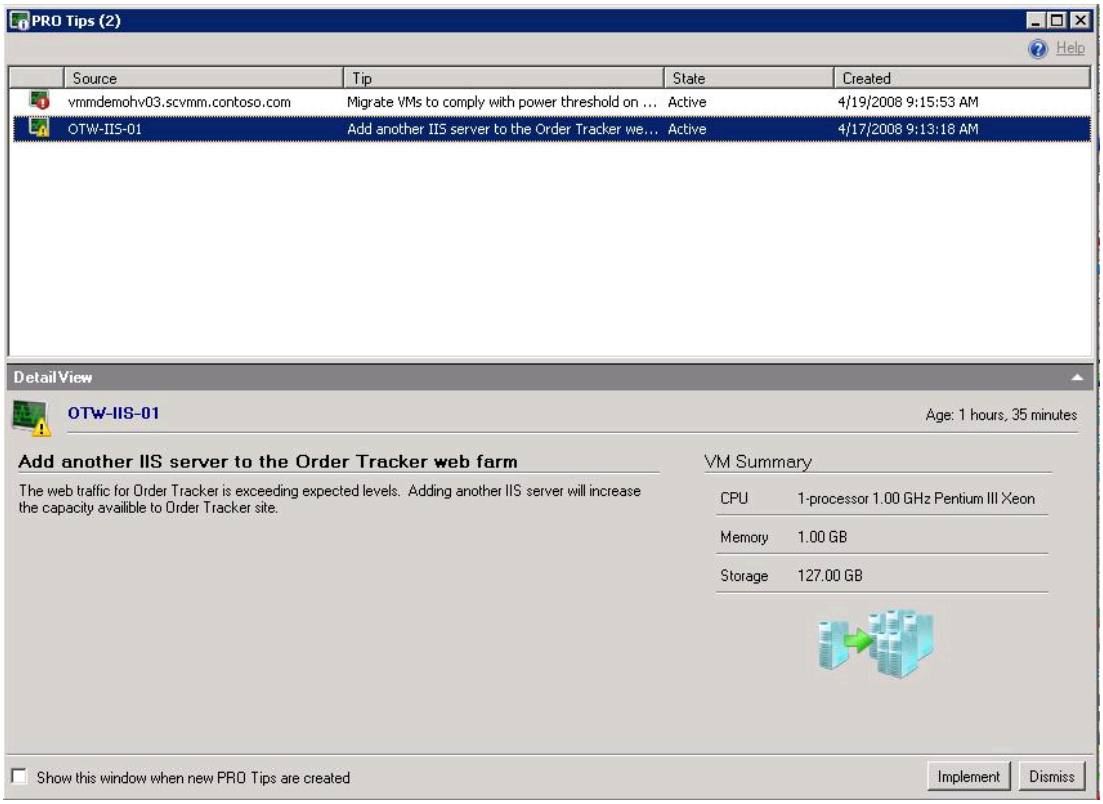


Figure 9: Performance and Resource Optimization Tips

Healthy State Definition

The default settings and thresholds in these management packs reflect a general definition of a healthy state for the hosts and virtual machines that VMM is monitoring. The following tables list the performance thresholds and sampling methods for the PRO monitors for hosts and virtual machine in the VMM management pack [30]:

PRO Monitors for Hosts

Monitor	Threshold	Sampling Interval	Calculation
Memory Utilization	90%	60 sec	Average of past 3 samples
CPU Utilization	75%	60 sec	Average of past 3 samples

PRO Monitors for Virtual Machines

Monitor	Threshold	Sampling Interval	Calculation
Memory Utilization	90%	60 sec	Average of past 3 samples
CPU Utilization	90%	60 sec	Average of past 3 samples

Extensible Architecture

All objects that define a health model (monitors, rules, classes, and so on) are stored in a management pack (MP, see Figure 10). The default VMM MP provides a basic set of PRO monitors that detect situations in which migrating virtual machines or changing a virtual machine configuration can optimize the performance of a host or a virtual machine. Changes to MPs can be done by creating overrides that change various aspects of the objects defined in the MPs or by creating additional MP elements, such as rules, monitors and recovery tasks, to meet the requirements of their virtualized environments. Recovery tasks can execute custom scripts written in Visual Basic Script (VBS) or Windows PowerShell (PS). [31]

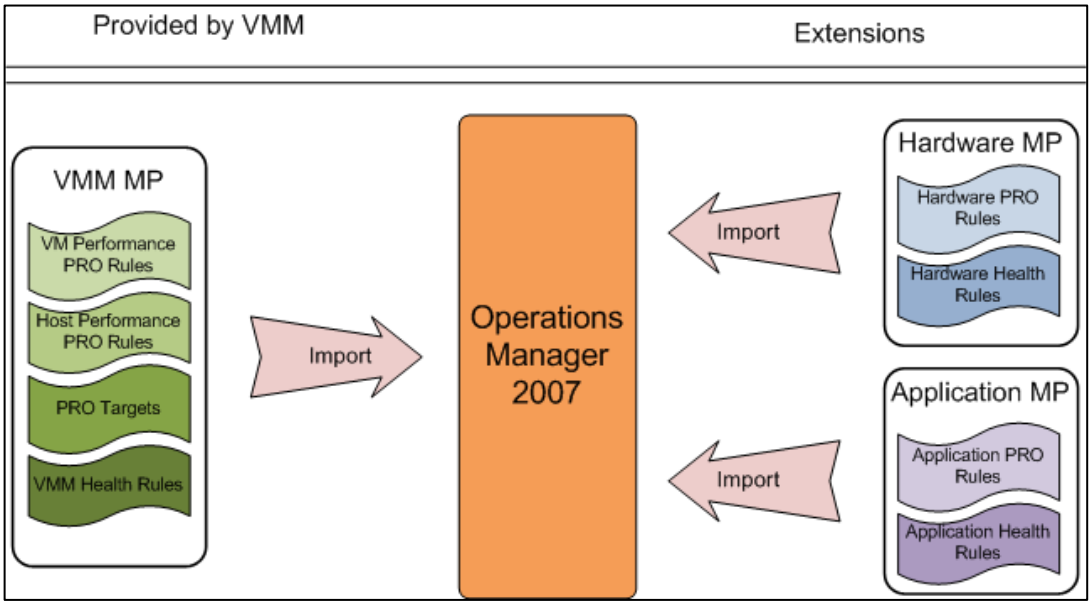


Figure 10: VMM 2007 Extensibility by Management Packs [32]

Intelligent Placement

Intelligent Placement is a capacity planning function that takes key performance metrics together with user settings as input and generates a set of ratings of hosts for a given VM to be started on or migrated to (Figure 11).

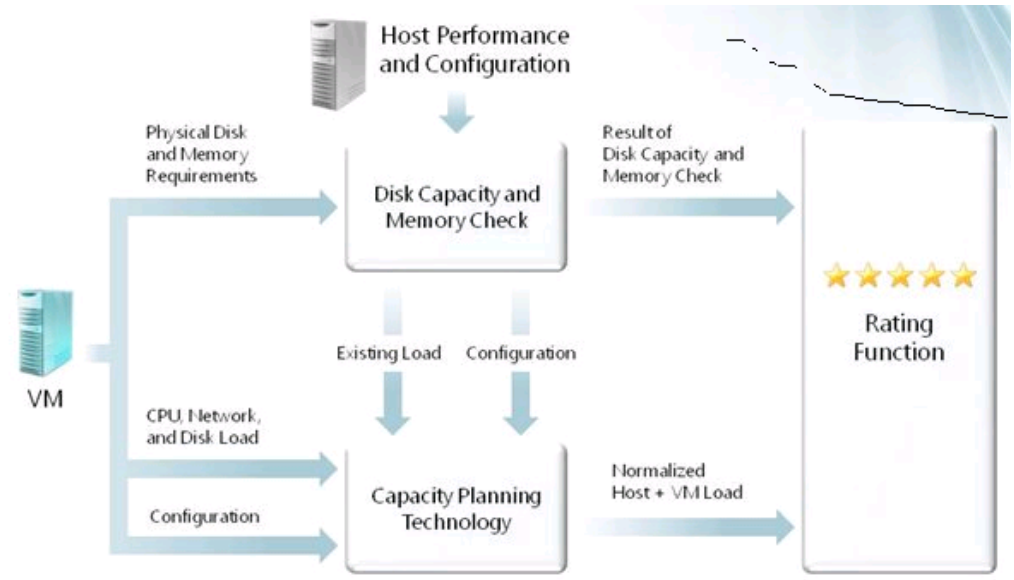


Figure 11: Intelligent Placement Algorithm [11]

The algorithm is used in a number of scenarios:

- When a VM is created in the self-service console, it is automatically place on a host by VMM based on the host ratings.
- When a VM is created in the administration console, it will recommend a host for the administrator to choose.
- When there is a host failure, VMM will use Intelligent Placement to move the VM to the highest rated host.
- When SCOM and PRO tips initiate an alert, VMM will use Intelligent Placement to relocate VMs to the host with the most available resources, i.e., the highest rated host.
- When a VM is moved to a host group, the VM will be automatically placed on a host in that group based on host ratings.

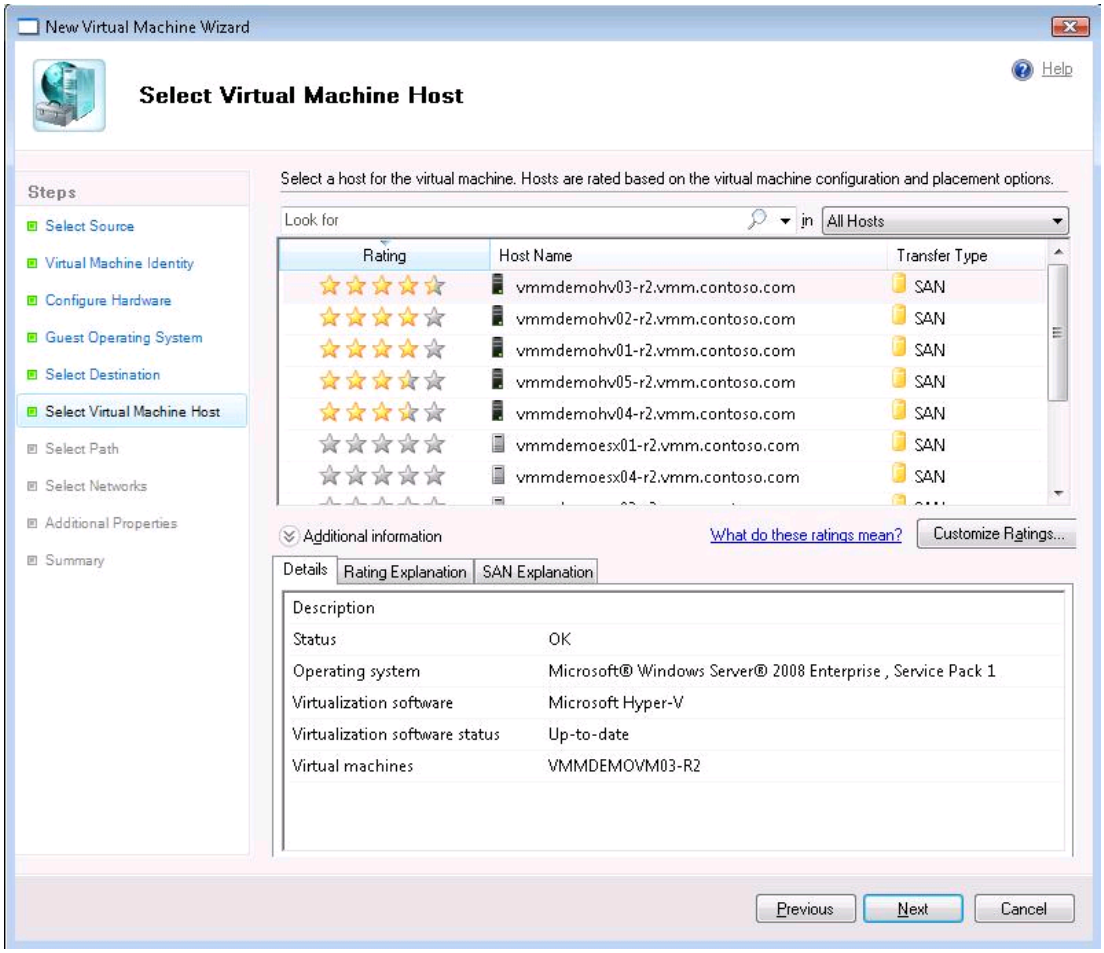


Figure 12: VMM Host Recommendations

The Intelligent Placement algorithm can be configured to work according to two basic models (Figure 13):

- Resource Maximization: VMM will try to place as many VMs on a single host as possible.
- Load Balancing: VMM will locate VMs in an effort to balance the resource utilization across all hosts.

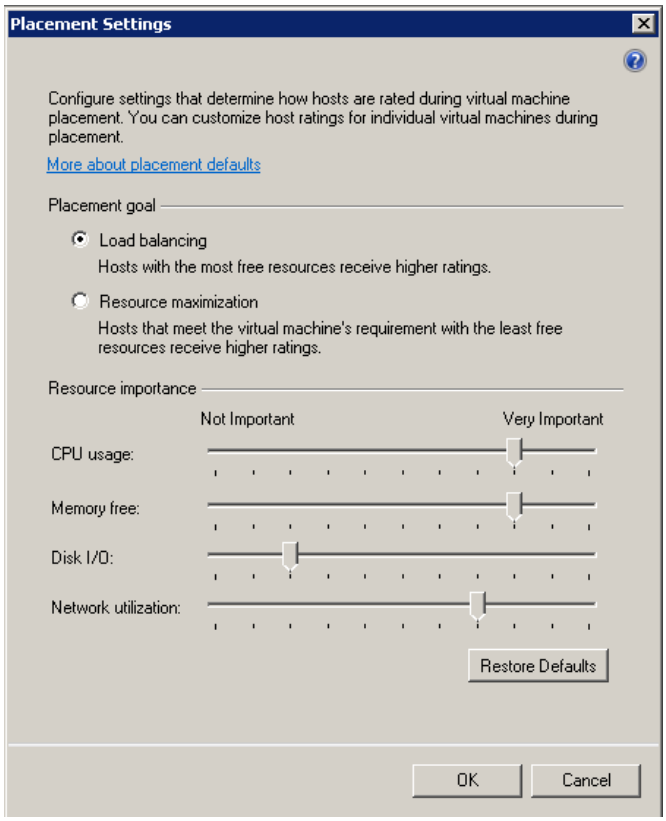


Figure 13: Intelligent Placement Settings

There are 4 resource types that are utilized in the algorithm. The user interface of the VMM provides a slider to allow the user to prioritize these resources when they are evaluated (see Figure 13):

- CPU
- Memory (RAM)
- Disk I/O capacity
- Network capacity

The actual host rating is then computed using this simple formula [11]:

HostRating = FreeCPU * CPUWeight + FreeMemory * MemoryWeight
+ FreeDisk * DiskWeight + FreeNetwork * NetworkWeight

Figure 14 illustrates how the remaining free capacity is calculated.

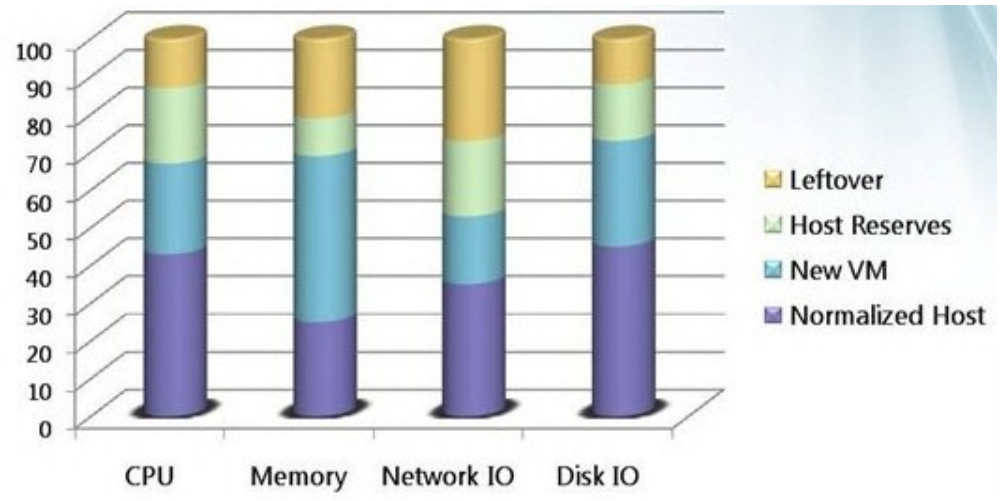


Figure 14: Intelligent Placement Free Resources Calculation

A number of circumstances can cause a host's rating to be zero, i.e., the placement is unsuitable [33]:

- There is not enough RAM available for the VM you want to place on a host.
- There is not enough available storage for a VM, e.g. a Windows Server 2008 Hyper-V cluster does not have an available LUN for the VM.
- The virtual network the VM is configured to use is not available on the host.
- Some advanced VM configuration is not supported by the host, e.g. advanced networking or high availability.

3.4 OpenNebula

OpenNebula is an open source, virtual infrastructure manager that deploys virtualized services on both a local pool of resources and external IaaS clouds. It automates VM setup (preparing disk images, setting up networking, and so on) regardless of the underlying virtualization layer (Xen, KVM, or VMware are currently supported) or external cloud (EC2 or ElasticHosts are currently supported) [18].

Scheduling Policies

The Scheduler module is in charge of the assignment between pending Virtual Machines and known Hosts. It is designed in a generic way, so it is highly modifiable and can be replaced by third-party developments. OpenNebula comes with a match making scheduler that implements the Rank Scheduling Policy for initial placement. The goal of this policy is to prioritize those resources more suitable for the VM.
The match-making algorithm works as follows: [34]

- First those hosts that do not meet the VM requirements and do not have enough resources (available CPU and memory) to run the VM are filtered out.
- The Rank expression is evaluated using the information gathered by the monitor drivers.
- Those resources with a higher rank are used first to allocate VMs.

Several placement heuristics can be implemented by choosing a Rank expression. As each VM has its own Rank and so its own policy, different policies can be applied to different instance types: [34]

- Packing Policy: The target is to minimize the number of cluster nodes in use. Those nodes with more VMs running are used first to reduce VM fragmentation.
- Striping Policy: The target is to maximize the resources available to VMs in a node. Those nodes with less VMs running are used first to spread the VMs in the cluster nodes.
- Load-aware Policy: The target is to maximize the resources available to VMs in a node. Those nodes with more free CPU are used first.

3.5 Summary

Here is a summary of key positive and negative aspects of the analyzed solutions:

VMware DRS

Pros

- **Simplicity**: VMware DRS is an integral part of vCenter Server and works out of the box, without any special configuration.
- **Agentless data collection**: No monitoring agents need to be installed in the managed VMs.
- **Affinity rules**: Complex VM-Host and VM-VM affinity rules can be created.
- **Cross-platform**: Virtually any guest OS is supported, because the monitoring is done at the hypervisor level.

Cons

- **Limited customizability**: Neither metrics nor remediation actions can be implemented by user-provided code.
- **Lack of application-level monitoring**: DRS cannot monitor application-level metrics, e.g. number of transactions processed per second, which are often defined in SLAs.
- **Limited resource type support**: DRS uses only CPU and Memory metrics to decide, which VMs need to be migrated.

Microsoft PRO Tips

Pros

- **Support for heterogeneous environments**: Provides native support for the most common hypervisors, Microsoft Hyper-V, VMware ESXi and Citrix XenServer.
- **Extensibility**: Can be fully extended by management packs containing custom rules, monitors and recovery tasks.
- **Rich resource type support**: Uses CPU, Memory, Storage and Network metrics to decide, which VMs need to be migrated. Even custom resources can be added by management packs.
- **Application-level monitoring**: Management agents installed in the VMs can monitor server software, e.g. web or database server, and provide application-specific metrics.

Cons

- **High complexity**: Before PRO can be used to Live Migrate virtual machines between hosts, Hyper-V, Active Directory (AD), Failover Clustering, Cluster Shared Volumes (CSV), VMM and SCOM need to be configured first. The next step is to integrate VMM with SCOM, which is a relatively complex task [35].
- **SCOM Agents dependency**: SCOM monitoring agents need to be installed on each virtualization host (except VMware ESXi servers), and on each managed VM. This also means that each guest OS has to be explicitly supported by Microsoft.
- **VMware vCenter Server dependency**: VMM cannot manage VMware ESXi hosts directly. It uses vCenter Server as a proxy instead, which introduces additional licensing and configuration burden.

Pricing

It cannot be generally decided, which of these two solutions is more costly, because of fundamental differences in licensing models. While VMware licenses are sold per CPU [9], Microsoft licenses are sold per VM [22].

OpenNebula

Pros:

- **Open source**: The OpenNebula software is fully open-source software distributed and licensed for use under the terms of the Apache License, Version 2.0.
- **Rank expressions**: OpenNebula Scheduler supports user-provided host Rank expressions that direct the VM placement.
- **Support for hybrid clouds**: Xen, KVM, VMware, EC2 and ElasticHosts are currently supported.

Cons

- **No dynamic balancer**: Only initial VM placement feature is supported by the scheduler module.
- **No GUI**: Scheduler configuration is done by manually editing text files.

3.6 Feature Requirements

None of the discussed solutions are suited for our educational environment because of our unique combination of requirements:

- Dynamic load management
- Intelligent placement
- Seamless integration with VMware vSphere Standard
- Simple deployment
- Free solution

Chapter 4: Solution Approach

As none of the existing load balancers discussed in the previous chapter suited our specific needs, we resolved to design and implement our own solution. In this chapter we discuss the most important design decisions made in the process.

4.1 Performance Counters

We decided to use Active CPU and Active Memory as performance counters used by the imbalance metric [10] [36]. We decided not to consider network I/O, because most of the network traffic in our environment does not cross virtualization host boundaries. As there is only one storage array present in our virtualization infrastructure and it is shared by all virtualization hosts, we also decided not to monitor storage I/O, simply because migrating a VM between hosts does not change the total amount of storage array R/W operations.

4.2 Imbalance Metric

One of the key decisions we had to make when designing the load balancer was choosing the right imbalance metric. We decided to use a verified metric proposed in [14] over designing our own.

The Virtualized Server Load (VSL) for a Host ϵ Cluster is then defined as:

$$\sum_{resource \in \{CPU, memory\}} W_{resource} \text{ x } \frac{\sum_{v \in VM_{Host}} v_{resource}\ usage}{Host_{resource} capacity}$$

where $W_{resource}$ is a weight associated with each resource. This generates a load set L containing the VSL values corresponding to all physical hosts. The system imbalance metric is then defined as:

$$C_L = \frac{\sigma_L}{\mu_L}$$

This metric captures the dispersion of the values assumed by a variable in a way that does not depend on the measurement unit. The higher the C_L, the greater is the variation in the measured values.

When all servers are idle or when the virtual machine monitor is not consuming any resources, μ_L becomes zero and C_L will be undefined. Therefore, the final metric was redefined as:

$$I_{Metric}: \begin{cases} 0, if\ no\ VMs\ are\ active \\ C_L, otherwise \end{cases}$$

4.3 Balancing Algorithm

We shortly experimented with an optimization model based on linear programming (LP) [15]. We defined the LP model using Microsoft Solver Foundation Enterprise Edition 3 and used Gurobi Optimizer 4 as the back-end solver. Unfortunately, our model had over 500 binary variables and running times of the Branch-and-Bound method used by Gurobi grow exponentially with the input size. As expected, we did not receive any results and abandoned LP.

We then decided to use the VSL Inductive Balancing Method (VIBM) that follows a greedy approach by inductively selecting the VM migration that will yield the greatest improvement of the imbalance metric at its present state [14]. The method is defined in Algorithm 1.

$I_{Metric} \longleftarrow C_L$
while $I_{Metric} \geq threshold$ **do**
  $src \longleftarrow$ host $\in S : VSL_{host} = \texttt{Max}(L)$
  for $v_{candidate} \in VM_{src}$ **do**
    for $(target \in S) \neq src$ **do**
      predict benefit of migrating $v_{candidate}$ to $target$
      compute $L_{candidate}$

$$L_{candidate}: \begin{cases} VSL_{target} \longleftarrow VSL_{target} + v_{candidate} \\ VSL_{src} \longleftarrow VSL_{src} - v_{candidate} \end{cases}$$

      insert $(v_{candidate}, src, target, L_{candidate})$ to $Candidates$
    end
  end
  select $candidate$ from $Candidates$: $\texttt{Min}(C_{L_{candidate}})$
  $I_{Metric_{PREDICTED}} \longleftarrow C_{L_{candidate}}$
  if $I_{Metric_{PREDICTED}} < I_{Metric}$ **then**
    promote $candidate$ to migrate
    $I_{Metric} \longleftarrow I_{Metric_{PREDICTED}}$
  else
    do not promote $candidate$
  end
end

Algorithm 1: VSL Inductive Balancing Method

4.4 Virtualization Platform Selection

As VMware vSphere Standard is used by the Department of Software Engineering, it was also the primary target of our solution. We wanted the GUI to our load balancer to be as user-friendly as possible, so we decided to tightly integrate it with the VMware vSphere Client, which is a Windows application used to manage the virtualization infrastructure. On the other hand, we plan to extend the balancer to support other virtualization platforms (e.g. Microsoft Hyper-V or Citrix XenServer) in the future, so we tried to design it to be as platform independent as possible by introducing an abstraction layer that resides between the load balancing engine and the virtualization infrastructure management API. Adding support for a new platform would therefore require reimplementing just this thin abstraction layer, together with the GUI integration (Microsoft Management Console in case of Hyper-V). An alternative approach would be to provide a universal web interface for the price of having to use two distinct applications to manage the virtualization infrastructure.

4.5 vSphere Plug-in Model

VMware offers third-party developers and partners the ability to extend the vSphere Client with custom menu selections and toolbar icons that provide access to custom capabilities [37]. There are two methods of using the vSphere Plug-in architecture:

- Script-based – This type plug-in is basically a web application displayed within the vSphere Client GUI using an embedded Internet Explorer window. The web pages can interact with the client GUI through JavaScript calls. Custom buttons and context menus can be configured to open application-specific URLs. When using the script-based method, the following plug-in components must be supplied:
    - A configuration file that describes GUI elements that extend the vSphere Client user interface.
    - A script or application that extends the vCenter Server with capabilities that support the vSphere Client GUI extensions.
    - Extension registration
- C# API – Allows writing C# code that executes within the vSphere client process.

As the C# API has been marked as deprecated by VMware [38], we decided to use script-based model instead. It has proven itself to be a wise decision, because a new version of vSphere has been announced in the meantime and it is not going support the C# API.

4.6 Application Server

VMware vCenter Server 4, the infrastructure management server, is a Java application that installs together with a bundled preconfigured Apache Tomcat Server. Surprisingly, it can only run on top of Microsoft Windows Server.

Because we decided to use the script-based plug-in model, we had to choose an application server for our web-based application to run on. The viable alternatives were:

- Apache Tomcat – By choosing Tomcat as the underlying platform, our application would be able to run side-by-side on the same application server as vCenter Server itself, without needing to install and configure additional server components.
- Microsoft Internet Information Services (IIS) – Although an integral component of Windows Server, IIS has to be installed and configured explicitly. Using an additional application server could possibly introduce some extra security vulnerabilities and which is undesired, especially on a virtualization infrastructure management server.

Although all of the aforementioned arguments speak in favor of Tomcat, we still chose Microsoft IIS, because we have some experience in developing ASP.NET applications, but no experience in developing Java Server Pages.

4.7 Libraries Used

Although vSphere has a well-defined web services API, we rather used the vSphere .NET SDK to interact with the virtualization infrastructure, because it provides a robust object-oriented interface [37].

Another dependency is the Microsoft Automatic Graph Layout (MS AGL) from Microsoft Research, which is a tool for graph layout and viewing. We facilitated it to render virtualization infrastructure topology graphs. Although it is commercial software, it can be obtained through MSDN.

4.8 User Interface

The load balancer extends the vSphere Client GUI by adding these UI elements:

- Virtual machine "**Smart Power On**" toolbar button (Figure 15) and context menu option (Figure 16) – Powers on the selected VM while placing it on the least utilized cluster node.
- **vCenter Service Status** (Figure 17) – Shows the health status of the load balancer. If the load balancer web application cannot be contacted, status is switched from green to red.
- **Balancer cluster view tab** (Figure 18) – Shows detailed information about the proposed migrations and allows executing them. The information includes proposals visualization (Figure 19) and utilization charts. The proposals visualization is an incidence graph representing Host-VM relationships before (red) and after (green) performing migrations.
- **Plug-in Manager** (Figure 20) – Shows the information about the installed plug-ins.

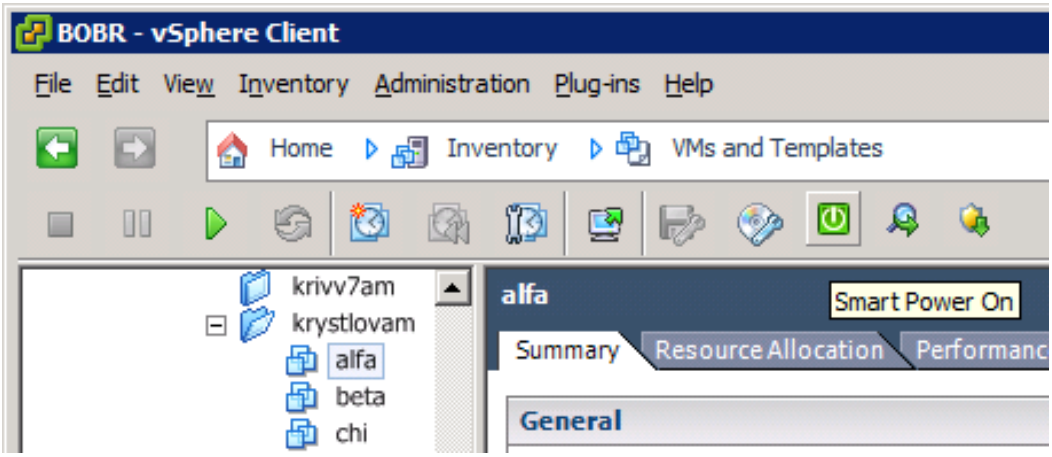


Figure 15: VM Toolbar Button

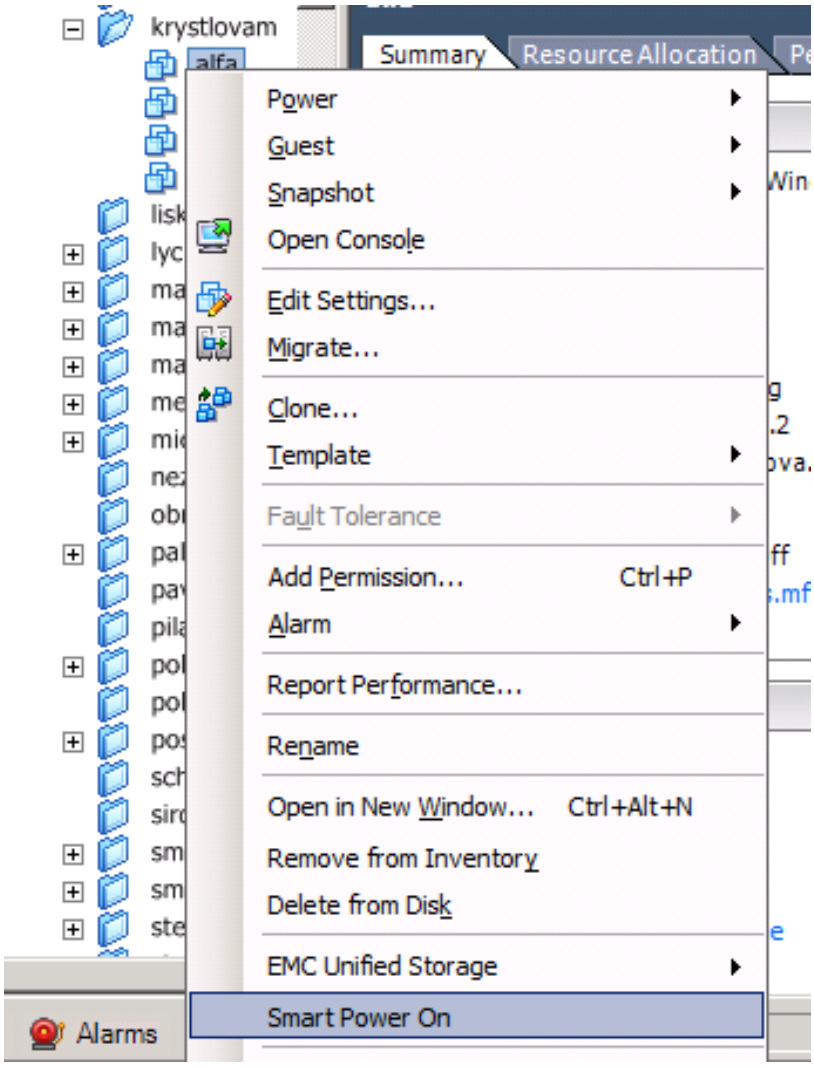


Figure 16: VM Context Menu

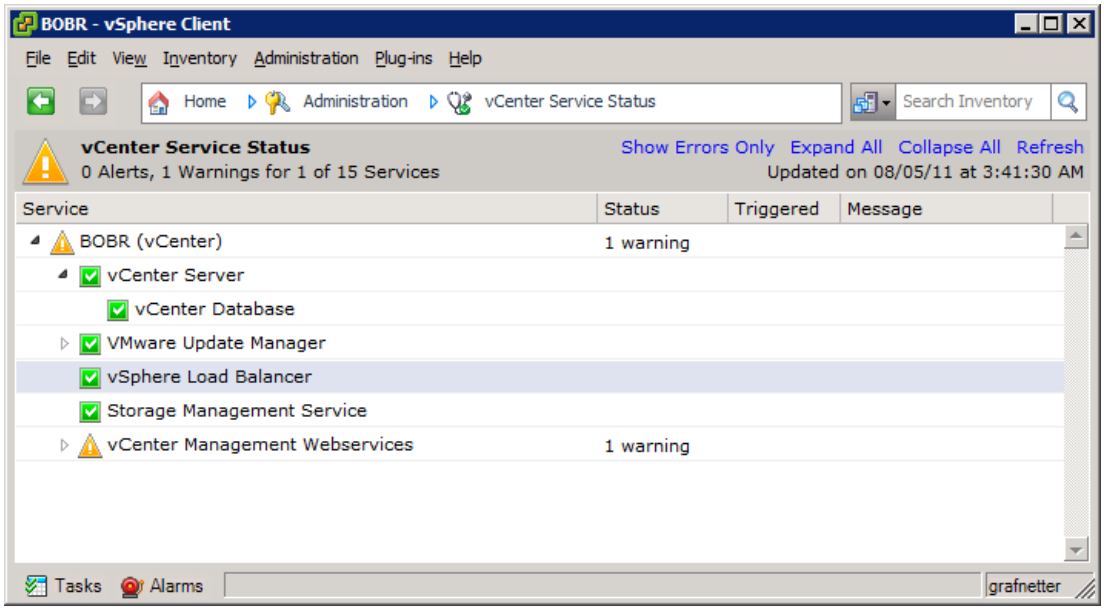


Figure 17: vCenter Service Status

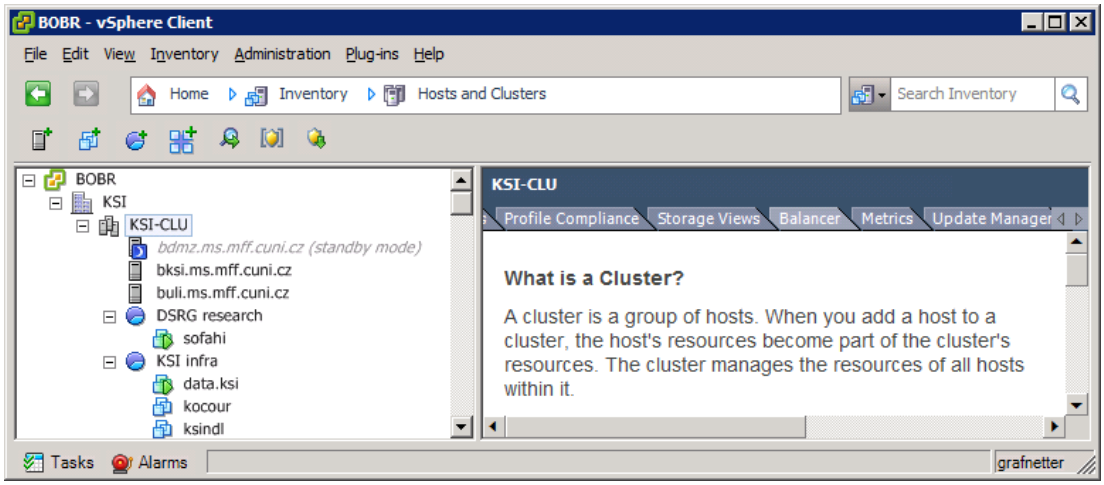


Figure 18: Balancer Cluster View

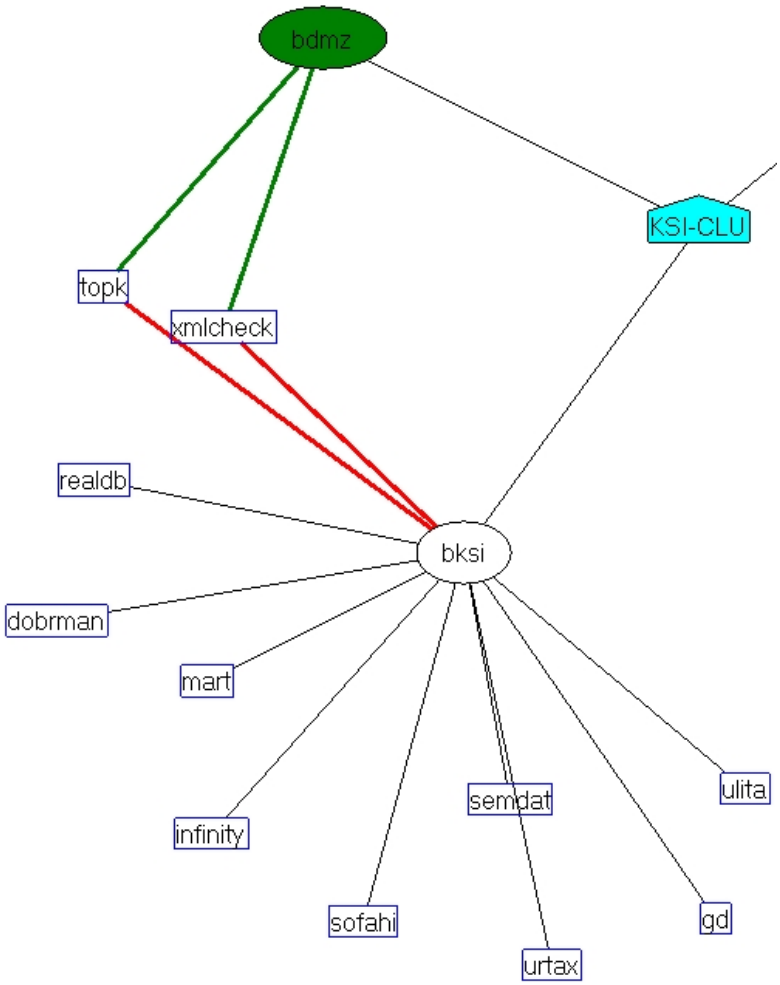


Figure 19: Load Balancer Proposals Visualization

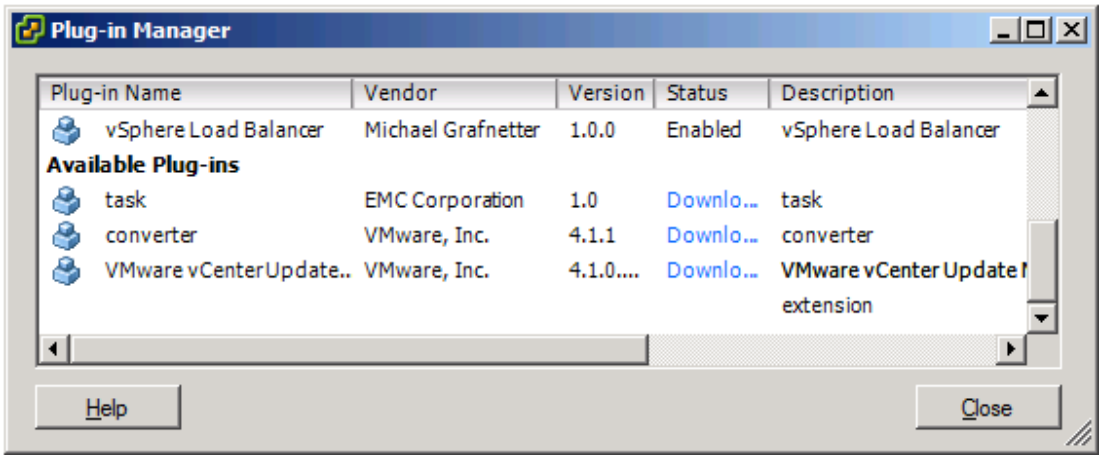


Figure 20: Plug-in Manager

4.9 Configuration Options

These options can be configured in the web.config file:

- Imbalance threshold
- CPU and Memory weights
- Balancing frequency
- List of managed clusters

Chapter 5: Results

5.1 Testing Environment

As stated earlier in this text, all of our experiments were conducted in the Virtlab Datacenter of the Department of Software Engineering at Charles University in Prague. The Datacenter consists of four blade servers named Bogatyr, Gromoboj, Navarin and Potemkin with the following hardware configuration:

Bogatyr and Navarin

Model: Dell PowerEdge M910

CPU: 4x Intel Xeon E7540 (6 cores running at 2.0 GHz with HT enabled)

Memory: 128 GB RAM

Gromoboj

Model: Dell PowerEdge M905

CPU: 4x AMD Opteron 8431 (6 cores running at 2.4 GHz)

Memory: 96 GB RAM

Potemkin

Model: Dell PowerEdge M905

CPU: 4x AMD Opteron 8356 (4 cores running at 2.3 GHz)

Memory: 64 GB RAM

CPU and Live Migration

Unfortunately, these four machines have slightly different hardware configuration, even different CPU architecture (AMD vs. Intel) [3]. As a result, they form 2 mutually exclusive sets (Bogatyr-Navarin and Gromoboj-Potemkin) and virtual machines running on these physical servers can only be live-migrated within the boundaries of their respective sets. The only way to migrate a VM between these sets (e.g. from Bogatyr to Gromoboj) is to shut the VM down first, perform the migration, and turn it on again. Even trying the migration of the VM with its Guest OS being in suspended/hibernated state would result in a kernel panic or the infamous Blue Screen of Death (BSoD) in the case of a Windows OS, caused by the absence of some instructions on the target host's CPU.

These facts bring some limitations to the tests that can be carried out in the testing environment. For example, the workload cannot be balanced across the whole datacenter with VMs turned on. This should also serve as a warning that when datacenter hardware acquisition is planned, even the compatibility of processor architectures has to be taken into account.

Network Configuration

The servers are interconnected through a 1000Mb physical network switch built into the blade enclosure and each host is equipped with one Broadcom NeXtremene II BCM5709 Quad Port Gigabit Ethernet Adapter. One Ethernet port on each of these Network Interface Controllers (NICs) and a separate VLAN on the physical switch are dedicated to vMotion traffic only. This decision has been made to eliminate or at least minimize any interference between the migration traffic and ordinary or management traffic.

Storage

All virtualization hosts are connected to an EMC CLARiiON CX4 Series storage array through 8Gb Fibre Channel switched fabric. All virtual machine files are stored on a shared 8TB VMFS 3.33 partition spanning four 2TB LUNs.

Thin provisioning of virtual disks is employed to make virtual machine provisioning faster and to reduce the storage capacity needed to store the virtual machine files. This approach has one major disadvantage, which is a negative impact on the I/O performance, especially on the Write operations.

After a careful examination of the parameters of all hardware components, it is obvious that the storage array is the sole bottleneck of the entire infrastructure. One solution would be to add a second partition and to distribute the virtual machines between these two partitions. This new partition should of course be located on different spindles than the first one and ideally, also on LUNs served by disparate storage processors. A drawback of this approach would be the impossibility to live migrate virtual machines between these two partitions because the VMware vSphere 4 Standard license owned by the department does not include Storage vMotion feature.

Software

All virtualization hosts are running VMware vSphere Hypervisor ESXi 4.1.0 Update 1 and are centrally managed by a discrete physical server running VMware vCenter Server 4 Standard on top of Microsoft Windows Server 2008 R2 Standard SP1 and Microsoft SQL 2008 R2 Standard.

5.2 Duration of a Live Migration

In chapter two we stated that total duration of migrations proposed by the load balancer should be much shorter than the time period between two executions of the balancer. Therefore, we wanted to know how long a single live migration takes, and more importantly, how many live migrations per host can be run serially between two executions of the migration planner.

We decided to perform a synthetic test by creating a simple Windows PowerShell script that migrates a VM between two hosts 1000 times in a row, saving the migration times into a CSV file. As both Bogatyr and Navarin have the same hardware configuration, these two servers seemed to be the best candidates for the test. As the VMware PowerCLI commandlets communicate with the vCenter Server using Web Services, we executed the script directly on the server hosting VMware vCenter Server to minimize the effect of network latency on the results.

The VM being migrated had these parameters:

CPU:	1 vCPU
Memory:	512 MB
Guest OS:	Windows 7 x64

This VM was under very low workload as well as the other 130 VMs running inside the virtualization infrastructure, which are circumstances expected to be true also during the semester.

Figure 21 shows the measurement results projected onto a histogram plot with bin size of 2 seconds. As we can see, the majority of the migrations took approximately 21 seconds with a relatively low variance, but there were a couple of migrations taking almost twice as long.

The data appear to have a normal distribution. To be sure, we tested the normality of the sample using the Kolmogorov-Smirnov Goodness-of-Fit Test at a significance

level α=0.05 and α=0.01. Both of these tests failed so we had to reject this hypothesis. Nevertheless, we will presume that the total duration of *n* migrations performed serially has asymptotically normal distribution with mean n*20.7, according to the central limit theorem.

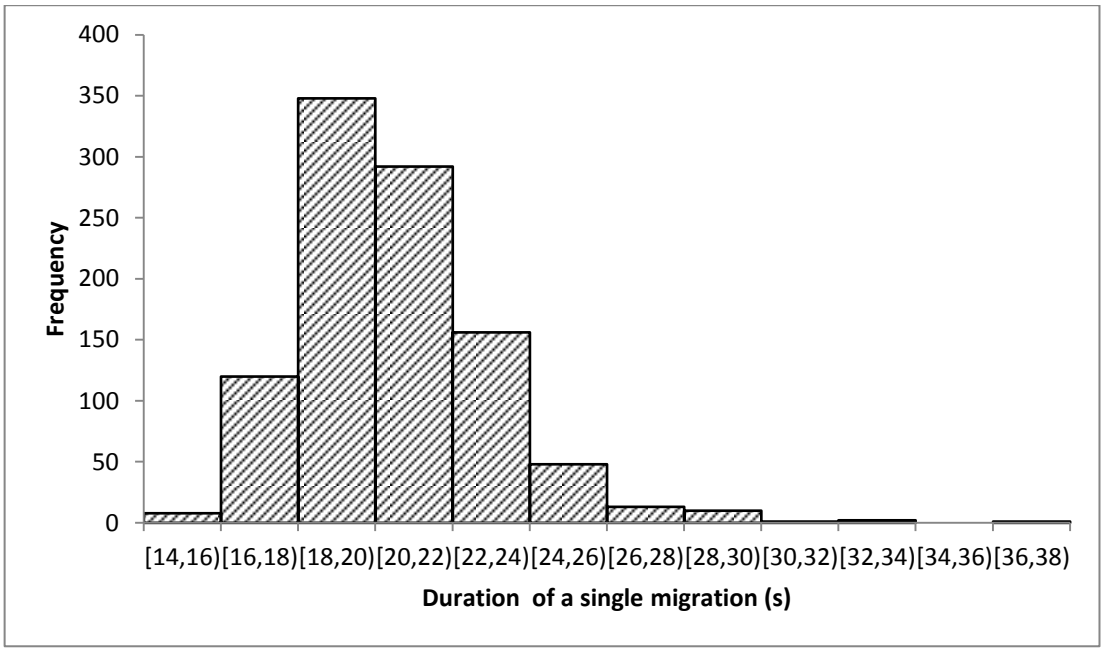


Figure 21: Histogram of Live Migration Test Results

It might appear that 21 seconds is a relatively short time for a live migration, but we cannot forget the fact that we often have to migrate several VMs in a row for the system to become balanced. In the worst-case scenario, with the system being totally unbalanced, all the VMs would be running on a single virtualization host. If we suppose that all hosts have the same hardware configuration and all VMs have equal resource demands, the VMs would have to be evenly distributed to the rest of the hosts to achieve equilibrium. In our testing environment, this would take as much as 35 minutes with 130 VMs in the powered on state.

Another assumption we can make about the results is that during the load balancer's 5-minute period, not more than 14 migrations involving a common host (being source or target) can be executed. For practical reasons, the system should be given at least a minute to stabilize itself. Otherwise, the host performance counters could be affected by the bygone migration. Therefore, the highest applicable number of migrations per period is 10.

5.3 Running Live Migrations Concurrently

While each set of nodes involved in a Microsoft Windows Server 2008 R2 Hyper-V Live Migration (the source and target) supports only one concurrent live migration at a time [39], VMware vSphere 4.1 allows 4 concurrent vMotion operations per host on a 1Gb/s network [40] [24]. Presence of this feature raises a question whether concurrent migrations should be used by the load balancer.

As we previously stated, there are several reasons why only one live migration should be run at a time and we estimated that running two migrations concurrently would have a remarkable impact on the performance.

To verify our expectation, we conducted a similar experiment as the one described in the previous section, but this time we ran several live migrations concurrently. We only measured migration times of the same reference VM as used in the previous experiment, because even if the other VMs had identical configurations, their respective operating systems could have been using different amounts of memory pages, which would negatively affect the results.

The table below shows the statistical summary of experimental results:

	Number of concurrent migrations		
	1	**2**	**3**
Median	20.6	21.8	27.9
Mean	20.7	22.1	30.1
Min	14.6	15.6	16.6
Max	36.8	44.9	53.7
$\mathbf{Q_1}$	18.9	20.0	21.8
$\mathbf{Q_3}$	21.9	24.2	39.7

To help us compare the distributions of these data sets, we created the boxplot shown on Figure 22. Contrary to our expectations, running two migrations concurrently had just a marginal impact on the duration of the migration of the reference virtual machine. In other words, we have registered almost linear speedup with regard to the total migration time when doubling the number of concurrent migrations. The only drawback is a slightly increased variance.

Increasing the number of concurrent migrations to 3 brought further variability, making the duration far less predictable, with a quarter of migrations lasting more than 40 seconds. Moreover, as the successful outcome of the algorithm used to migrate dirty memory pages between hosts is highly dependent on network latency and throughput [3] [6], it is enough to discourage us from running more than two concurrent migrations in production environment with the same configuration as our testing environment. Therefore, we decided not to repeat the experiment for 4 concurrent migrations, which is a hard-coded maximum the platform can handle.

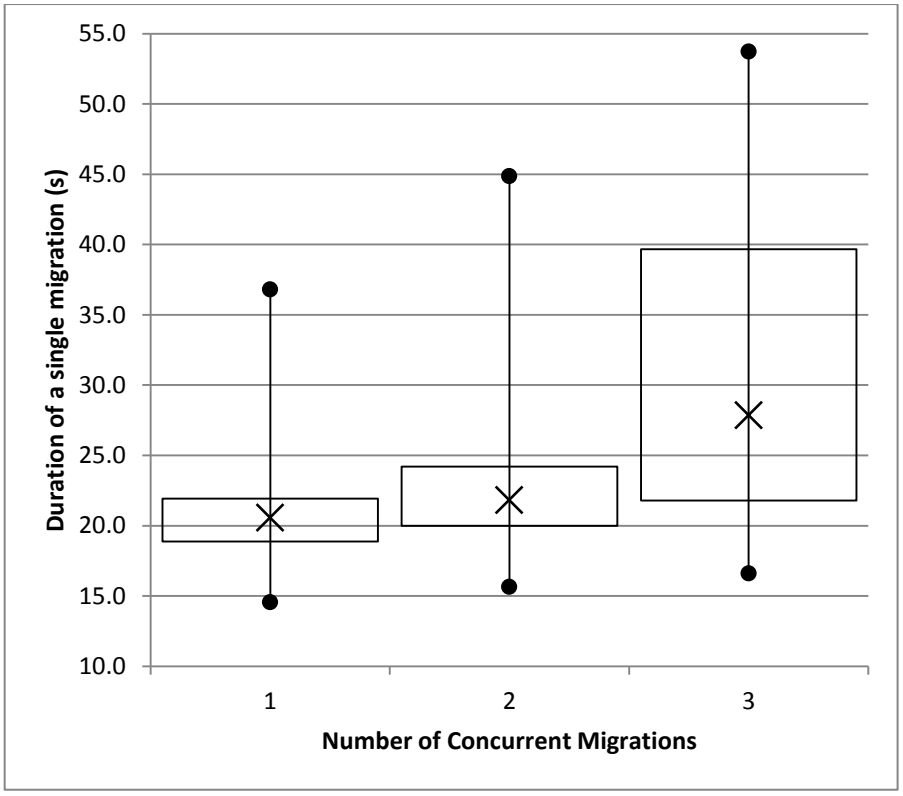


Figure 22: Effect of concurrency on live migration time

5.4 Disclaimer

All the results published in this chapter are heavily dependent on hardware and software configuration of the entire infrastructure. As a result, all our conclusions can only be applied to this particular environment. Furthermore, they might have also been biased by workload originating in the production datacenter, which too is managed by the same vCenter Server and shares some hardware resources with the testing environment, including the Storage Area Network and Ethernet switches. There was no way for us to eliminate or influence these

factors. Consequently, the results can by no means be interpreted as benchmarks of VMware vMotion or of the underlying hardware. Needless to say, the VMware vSphere license strictly disallows the publication of any third-party benchmarks.
All the previous tests were conducted only under very low workloads, as this is the expected behavior in the educational environment. However, other studies of the live migration are available, that have analyzed its behavior under various workloads [6].

5.6 Internal Structure

The load balancer consists of these modules:

Balancer.vSphere.Web

This is the main module containing the actual load balancer ASP.NET web application. The most important files are:

- Web.config – The ASP.NET configuration file, where the balancer parameters can be changed.
- ScriptConfig.xml – Plug-in configuration file. Defines all vSphere client extension points the UI is hooked to.
- health.xml – Contains static information about the health of the application. If accessible through HTTP, the application is considered to be healthy by vCenter Server.
- ClusterInfo.aspx – Implementation of the cluster view tab.
- SmartPowerOn.aspx – Smart Power on action handler.

VIClientPlugin

This library provides only minor helper classes that are facilitated by all ASPX pages, among them are:

  - VIPluginPage – Base class for all command and view pages. Extracts the session information from URL parameters and initializes a VIClient Context.
  - VIClientContext – Class that represents the current vSphere User Session and manages the client-server connection.

Balancer.vSphere.Plugin

Contains only the extension .xml file, that has to be registered by the vCenter Server Extension Manager. Provides plug-in description and web application URL, from which the vSphere Client can download ScriptConfig.xml and health.xml.

Balancer.Visualization

This library is used to render proposals chart from object model using MS AGL.

Balancer.Interfaces

Contains common interfaces and enums, which provide the abstraction layer and are independent of the underlying virtualization technology, e.g. IEntity, ICluser, IHost, IVM, VMState and HostState.

Balancer.vSphere

Contains the layer that communicates with the VMware vSphere. Classes Cluster, Host and VM are adapter classes between vSphere SDK and Balancer.Interfaces.

Balancer.XML

Provides dummy implementations of Balancer.Interfaces that serialize/deserialize the object model to/from XML instead of communicating with an actual virtualization infrastructure. Used for debugging /testing purposes only

Balancer.Service

Implements the load balancing service that runs in the background. It consists of the balancing engine that periodically calculates the imbalance metric and generates migration proposals and of the execution engine that performs queued lived migrations.

The service must be run in context of a user account that has the right to read VM properties and start live migration.

5.7 Implementation Issues

Lack of vSphere script plug-in sample code – We did not find any open-source implementation of a vSphere plug-in to draw some inspiration from.

Microsoft Automatic Graph Layout configuration issues – The GraphRenderer class ignored the rendering algorithm settings, so we had to reverse engineer (decompile) its source code and modify the default behavior. This approach will probably cause compatibility issues when upgrading to a newer version of the AGL library.

Conclusion

In this thesis, we focused on the problem of dynamic resource load balancing in virtualization clusters by using live migration to reduce imbalance. We introduced the basic concepts behind virtualization load balancing and discussed some common approaches to solve this problem, including algorithms and metrics that can be used. We then demonstrated how they are implemented in leading commercial and open source virtualization infrastructure management solutions – VMware vSphere, Microsoft Virtual Machine Manager and OpenNebula – and evaluated their respective strengths and weaknesses in detail. Based on the discussion, we have selected the Virtual Server Load (VSL) as the server load metric and the VSL Inductive Balancing Method (VIBM) as the balancing algorithm. We then implemented a prototype of a resource load balancer for the VMware vSphere Standard-based virtualization cluster. Unlike some other solutions, it is a lightweight plug-in that seamlessly integrates with the infrastructure management client.

Real-World Deployment

Our load balancer is going to be deployed at the Department of Software Engineering to optimize resource usage in the educational virtualization infrastructure. It is expected to increase the overall system responsiveness and possibly delay costly hardware upgrades.

Future Work

Although our solution is well-suited for the educational environment we have targeted, there are a number of areas in which we hope to carry out future work. We will implement additional imbalance metrics and balancing algorithms to compare their effectiveness. We are also planning to add support for additional virtualization platforms, including Microsoft Hyper-V and Citrix XenServer. Finally, we would like to expand our load balancer to enable effective power management.

Bibliography

[1] **McDonald, Mark P.** Leading in Times of Transition: the 2010 CIO Agenda. [Online] Gartner, January 19, 2010. http://blogs.gartner.com/mark_mcdonald/2010/01/19/leading-in-times-of-transition-the-2010-cio-agenda/.

[2] *On Strategies for Dynamic Resource Management in Virtualized Server Environments.* **Kochut, A and Beaty, K.** Washington, DC, USA : IEEE Computer Society, 2007. Proceedings of the 2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems. pp. 193--200. 978-1-4244-1854-1.

[3] **VMware, Inc.** *vSphere 4.1 Datacenter Administration Guide.* 2010.

[4] **Gartner, Inc.** Gartner EXP Worldwide Survey of Nearly 1,600 CIOs Shows IT Budgets in 2010 to be at 2005 Levels. [Online] January 19, 2010. http://www.gartner.com/it/page.jsp?id=1283413.

[5] **Ou, George.** Introduction to server virtualization. *TechRepublic.* [Online] May 22, 2006. http://www.techrepublic.com/article/introduction-to-server-virtualization/6074941.

[6] *Live Migration of Virtual Machines.* **Clark, Christopher, et al., et al.** 2005. In Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI.

[7] **Rajan, J. P.** How Live Migration works in Hyper-V R2. [Online] March 31, 2010. http://blogs.technet.com/b/ranjanajain/archive/2010/03/31/how-live-migration-works-in-hyper-v-r2.aspx.

[8] **VMware, Inc.** VMware vMotion. [Online] 2009. www.vmware.com/files/pdf/VMware-VMotion-DS-EN.pdf.

[9] —. VMware vSphere Pricing. [Online] http://www.vmware.com/products/vsphere/pricing.html.

[10] **Epping, Duncan.** Which Metrics does DRS use. [Online] October 15, 2009. http://www.yellow-bricks.com/2009/10/15/which-metrics-does-drs-use/.

[11] **Microsoft Corp.** Intelligent Placement in SCVMM 2008. [Online] May 14, 2008. http://blogs.technet.com/b/chengw/archive/2008/05/13/intelligent-placement-in-scvmm-2008.aspx.

[12] *SLA-Aware Virtual Resource Management for Cloud Infrastructures.* **Nguyen, Hien, Tran, F. D and Menaud, J.-M.** 2009. Proc. Ninth IEEE Int. Conf. Computer and Information Technology CIT '09. Vol. 1, pp. 357-362.

[13] *SLA-Driven Dynamic Resource Management for Multi-tier Web Applications in a Cloud.* **Iqbal, Waheed, Dailey, Matthew N and Carrera, David.** 2010. Proc. 10th IEEE/ACM Int Cluster, Cloud and Grid Computing (CCGrid) Conf. pp. 832-837.

[14] *Quantifying load imbalance on virtualized enterprise servers.* **Arzuaga, Emmanuel and Kaeli, David R.** 2010. WOSP/SIPEW'10.

[15] *Virtual machine migration in self-managing virtualized server environments.* **Park, Jong-Geun, et al., et al.** Piscataway, NJ, USA : IEEE Press, 2009. Proceedings of the 11th international conference on Advanced Communication Technology - Volume 3. pp. 2077--2083. 978-8-9551-9138-7.

[16] *Dynamic Resource Allocation in Computing Clouds Using Distributed Multiple Criteria Decision Analysis.* **Yazir, Y. O, et al., et al.** 2010. Proc. IEEE 3rd Int Cloud Computing (CLOUD) Conf. pp. 91-98.

[17] **Epping, Duncan and Denneman, Frank.** *VMware vSphere 4.1 HA and DRS technical deepdive.* 1. s.l. : CreateSpace, 2011. Vol. 1. 978-1456301446.

[18] *Virtual infrastructure management in private and hybrid clouds.* **Sotomayor, B, et al., et al.** 5, 2009, Internet Computing, IEEE, Vol. 13, pp. 14-22.

[19] *Maximizing Cloud Providers' Revenues via Energy Aware Allocation Policies.* **Mazzucco, M, Dyachuk, D and Deters, R.** 2010. Proc. IEEE 3rd Int Cloud Computing (CLOUD) Conf. pp. 131-138.

[20] **Dawson, Philip, et al., et al.** Magic Quadrant for x86 Server Virtualization Infrastructure. [Online] June 30, 2011. http://www.gartner.com/technology/media-products/reprints/microsoft/vol2/article8a/article8a.html.

[21] **VMware, Inc.** Compare vSphere 4.1 Editions. [Online] February 3, 2011. http://www.vmware.com/products/vsphere/buy/editions_comparison.html.

[22] **Microsoft Corp.** How to Buy Virtual Machine Manager 2008 R2. [Online] http://www.microsoft.com/systemcenter/en/us/virtual-machine-manager/vmm-pricing-licensing.aspx#Pric.

[23] **VMware, Inc.** Resource Management with VMware DRS. [Online] http://www.vmware.com/pdf/vmware_drs_wp.pdf.

[24] —. Changes to vMotion in vSphere 4.1. [Online] http://kb.vmware.com/kb/1022851.
[25] **Denneman, Frank.** DRS 4.1 Adaptive MaxMovesPerHost. [Online] August 27, 2010. http://frankdenneman.nl/2010/08/drs-4-1-adaptive-maxmovesperhost/.
[26] **VMware, Inc.** Calculating the priority level of a VMware DRS migration recommendation. [Online] http://kb.vmware.com/kb/1007485.
[27] —. VMware Storage Distributed Resource Scheduler (SDRS). [Online] http://www.vmware.com/products/datacenter-virtualization/vsphere/vsphere-storage-drs/features.html.
[28] **Denneman, Frank.** Should or Must VM-Host affinity rules. [Online] December 1, 2010. http://frankdenneman.nl/2010/12/vm-host-affinity-rules-should-or-must/.
[29] **Microsoft Corp.** About Performance and Resource Optimization (PRO). [Online] November 19, 2009. http://technet.microsoft.com/en-us/library/cc917965.aspx.
[30] —. Tuning PRO Performance Thresholds. *Microsoft TechNet.* [Online] http://technet.microsoft.com/en-us/library/ee423768.aspx.
[31] —. Customizing PRO. *Microsoft TechNet.* [Online] October 23, 2009. http://technet.microsoft.com/en-us/library/cc956018.aspx.
[32] —. Authoring PRO-Enabled Management Packs. [Online] 2009. http://download.microsoft.com/download/7/8/7/7871265C-8021-4E2E-8AC9-D5BFED4392B6/Building%20PRO%20Management%20Packs.docx.
[33] **Aidan, Finn.** How Does VMM Place Virtual Machines. [Online] October 12, 2009. http://www.aidanfinn.com/?p=10201.
[34] **OpenNebula.** Scheduler 2.2. [Online] http://opennebula.org/documentation:rel2.2:schg.
[35] **Microsoft Corp.** Configuring Operations Manager Integration with VMM. [Online] January 22, 2011. http://technet.microsoft.com/en-us/library/cc956099.aspx.
[36] **VMware, Inc.** vCenter Performance Counters. [Online] http://communities.vmware.com/docs/DOC-5600.
[37] vSphere SDK for .NET Developer's Guide. [Online] http://www.vmware.com/support/developer/windowstoolkit/wintk40/doc/viwin_devg.pdf.
[38] **VMware, Inc.** *Create and manage vCenter Server plug-ins.* [Online] http://www.ibm.com/developerworks/web/library/wa-vcenter/?ca=drs-.

[39] **Savill, John.** Q: How many concurrent live migrations does each pair of Hyper-V nodes in a cluster support? *Windows IT Pro.* [Online] April 12, 2011. http://www.windowsitpro.com/article/server-virtualization/q-how-many-concurrent-live-migrations-does-each-pair-of-hyper-v-nodes-in-a-cluster-support-.

[40] **VMware, Inc.** *VMware vSphere 4.1 Configuration Maximums.* [Online] http://www.vmware.com/pdf/vsphere4/r41/vsp_41_config_max.pdf.

[41] —. *vCenter Orchestrator Developer's Guide.* [Online] http://www.vmware.com/pdf/vco_410_developers_guide.pdf.

[42] vCenter Performance Counters. [Online] http://communities.vmware.com/docs/DOC-5600.

[43] *vCenter Orchestrator Developer's Guide.* [Online] http://www.vmware.com/pdf/vco_410_developers_guide.pdf.

[44] **Jain, R.** *The Art of Computer Systems Performance Evaluation.* New York : Wiley, 1991.

[45] *Customizing the vSphere Client.* [Online] http://www.vmware.com/support/developer/vc-sdk/vcplugin/vSphere_Plugin_4_1_Technote.pdf.

[46] **VMware, Inc.** Virtualization Basics. [Online] 2010. http://www.vmware.com/virtualization/virtual-machine.html.

Attachments

CD

The attached CD contains this document in a PDF format.

The source codes of the solution we implemented can be downloaded from departmental server at address https://bobr.ms.mff.cuni.cz:1234/balancer.zip.