Virtual Machine Creation in a Distributed Environment Ensuring Balanced Load Among Physical Machines

Project-6c

Name	RollNo	E-Mail	ContactNumber
Bhuma Sruthi	MT2012035	Sruthi.Bhuma@iiitb.org	8892145002
D Surya Pratap Desai	MT2012038	SuryaPratap.Desai@iiitb.org	7259981049
Preethy Varma	MT2012102	Preethy.Varma@iiitb.org	9739813467
Sruti Davis	MT2012140	Sruti.Davis@iiitb.org	9686518156
Tushar Sharma	MT2012153	Tushar.Sharma@iiitb.org	9980757450

Team Leader: Tushar Sharma

Contents

1 Introduction

Virtual Machines

Physical resources can be split into a number of logical slices called as Virtual Machines (VMs).Virtual machines enable us to run different operating systems in a single machine with complete independence.One physical machine can host multiple virtual machines and thus can act as a server system in itself.

Load Balancing

Load balancing is a core and challenging issue in the context of virtualization. It is a computer networking method to distribute workload across multiple computers or a computer cluster, or other resources, to achieve better performance. It is one of the prerequisites to utilize the full resources of parallel and distributed systems. Load balancing mechanisms can be broadly categorized as follows.

1. Centralized or Decentralized
2. Dynamic or Static
3. Periodic or Non-periodic

Load Balancing in Virtual Machine Creation

Load Balancing algorithms are used to handle the requests for the creation of new virtual machines in a data centre or cluster of computers so as to balance the load on the physical machines. Some purposes that a load balancing algorithm aims to achieve are:

1. Reduced task servicing time and task waiting time
2. Equal treatment of all tasks and requests
3. A working load balancing system inspite of failure of one or few nodes
4. Capability of modifying itself in response to changes
5. Creates faster job servicing and improves resource utilization.

The Data Center (or) Cluster Controller uses a virtual machine load balancer to determine which physical machine should be assigned to the next request for creating a new virtual machine. Most common virtual machine load balancing algorithms are

1. Throttled load balancing algorithms
2. Active monitoring load balancing algorithms

2 Project Description

Objective

When virtual machines are created in a networked cluster or data center, it is desirable that they be created in a way that balances the loads on all the physical machines of the cluster or data center.

Motivation

Maintaining load balance is a matter of critical importance for any distributed system. When the physical machines of the network are used for hosting virtual machines which may run varied applications, the issue of load balancing becomes a central concern.Through this project we aim at creating a system where requests to create virtual machines are handled in a way so that the load on the physical machines are balanced to the greatest extent possible.

Problem description

This problem could be thought of as a specific version of general load balancing issue in any distributed or cloud systems.

We consider the scenario of as a cluster environment consisting of the below components:

1. A set of homogeneous physical machines.
2. A central system that receives the requests for creating virtual machines from the external systems.

The central system should monitor the current state of physical machines of the network, where state includes parameters like resource utilization, job execution status etc. Based on current state and capacities of individual machines of the system, the load balancing algorithm should assign the task of creating the virtual machine on one of the selected systems of the network, such that the entire load of the cluster is balanced. The central server requires information about the jobs running in cluster machines, capacities of individual machines and resource requirements of the virtual machine. Once the load balancing algorithm running

on the central server decides one of the physical servers to host the virtual machine, it sends the virtual machine creation request to the selected physical server. Now, the physical server will create the new virtual machine.

3 Literature Study

Alakeel [3] in this paper, proposes dynamic approach to the load balancing algorithm, where the decisions to allocate work are dependent on the current system status. According to the author, the dynamic load balancing algorithm comprises of 3 components: information, transfer and location strategies [4]. Information strategy gives enough information to both the other strategies to make necessary decisions in the process of load balancing such as the current workload at each node. Transfer strategy is responsible for deciding whether it is necessary to transfer a newly arrived task to a remote node. When it has been decided to transfer the task then the location strategy finds a suitable destination node to which the task will be transferred. This paper lists various mechanisms for executing each of these strategies.

Branco et.al [5] talks about the requirement of a load metric to estimate the usage of resources. A good performance index is the one which will be able to estimate the future through current and past load factors. It has been noticed that many load indices are volatile which indicates the instability of the considered load metric. The instability arises due to working loads flotation.

There are four basic steps that nearly all load balancing algorithms have in common as referred in [6] [7]:

1. Monitoring workstation performance (load monitoring)
2. Exchanging this information between workstations (synchronization)
3. Calculating new distributions and making the work movement decision (rebalancing criteria)
4. Actual data movement (job migration)

4 Gap Analysis

VMWare DRS [2] achieves an on-demand resource scheduling scheme for virtual machine cluster via migrating virtual machines among physical machines. DRS guides the initial VM placement as well as semi- or fully-automated load management of VMs using its Distributed Resource Scheduler (DRS) feature. DRS does not enable much customization of the load management algorithm and is based

solely on CPU and memory utilization.What is more, no detail design of DRS is available because of business factors.

XenServer [1] includes a feature called Workload Balancing (WLB). Workload Balancing captures data such as CPU, memory, disk I/O and network I/O on the hosts and virtual machines to guide the initial and ongoing host location for virtual machines.There are two optimization modes for WLB: optimize for performance and optimize for density. Optimizing for performance ensures that minimum performance thresholds are maintained, whereas optimizing for density ensures reduced power consumption by placing VMs on the minimum number of hosts.

Similarly many of the load balancing solutions work with a dynamic methodology involving the migration of the virtual machines among the different machines in the network. Most of the work is concentrated in solution involving migration on a periodic basis.

We propose a static algorithm which assigns the task of creation of a virtual machine to a chosen system. Instead of attempting to load balance the clusters periodically, only the issue of initial VM placement is tackled as part of our project.

5 Architecture

The setup for the target infrastructure is as shown in the figure 1. The system comprises of the following four key components:

1. Client:

   Client in the architecture represents user or application requesting virtual machines. A typical scneario we have targeted is regarding the "host based desktop virtualization", where clients are given access to a virtual machine(VM) hosted on physical machine via a thin client and/or server application.

2. Central Dispatcher:

   A central dispatcher is the system dealing with client's VM Requests. All the reqests for VM creation as requested by the clients, are first recieved by the CD. CD also maintains performance snapshot of all the physical machines.

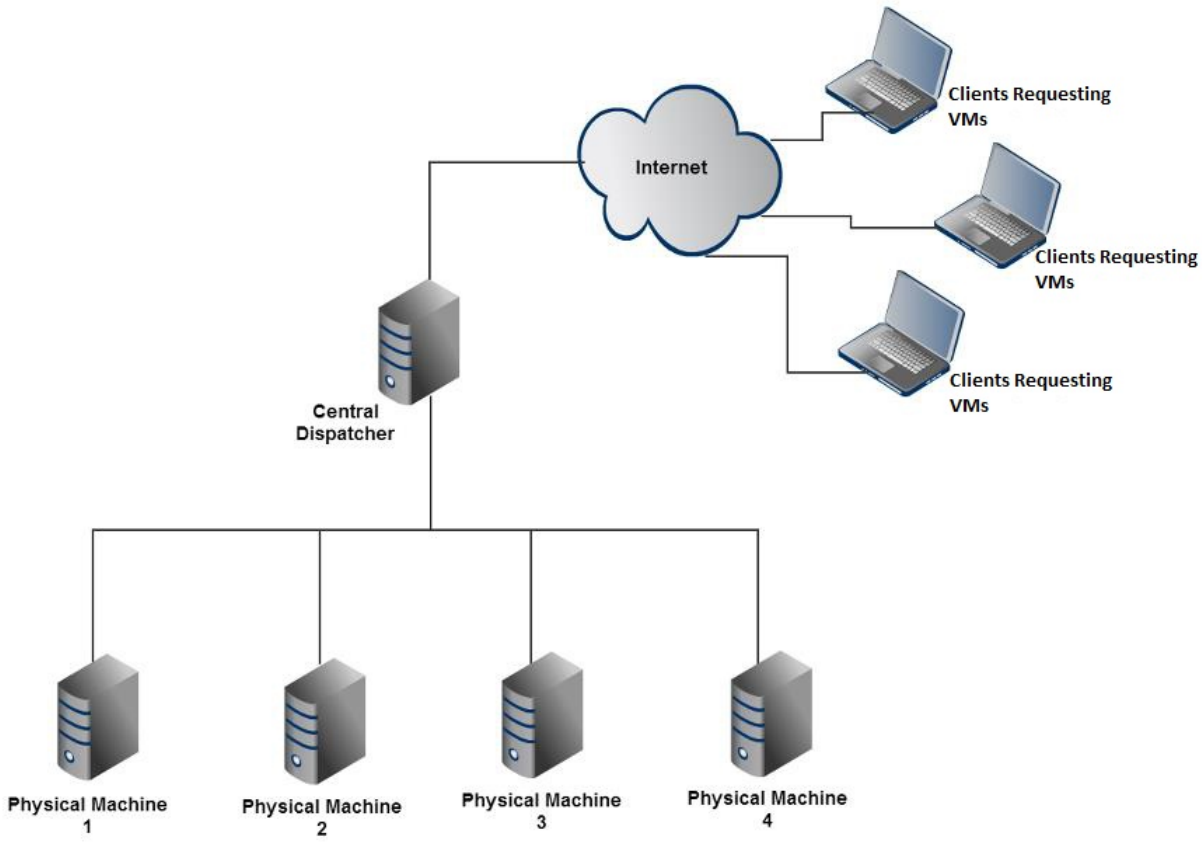

Figure 1: Architecture Diagram of Hardware

3. Physical Machines:

   Physical machines represent the actual computing resources which will be distributed via VM creation. Each physical machine is a high specs server which will be running a host OS(Linux).

4. Virtual Machines:

   Virtual machines represent the emulated computers running on the physical machines. Each VM is connected to a unique client.(A particular physical machine may host more than one VM which is only restricted by its load and computing resourecs.)

Proposed Solution Architecture

1. Metric Reporting Module:

   A statistic reporting module will be developed which will be deployed on each of the physical machines of the cluster. This module collects the indicative metric data of that physical machine. The collected data has to be dispatched to the central dispatcher where it will be stored in the data store.

2. Metric Data Store:

   A data repository will be created on the central dispatcher which will hold the indicative metrics received from the individual physical machines.

3. Metric Reporting Module-Client

   This module will extract the indicative metric data of each physical machine and send it to the Metric Reporting Module of the Server.

4. Metric Reporting Module-Server

   This module will receive the statistical information from the Metric Reporting Module of each client and store in the data store.

5. VM Placement Service:

   The most important part of this module is the following algorithm.

   VM Optimum Placement Algorithm
   This algorithm uses the data store and the VM request parameter to determine the most appropriate physical machine for the VM request.

   VM Placement Service module executes the load balancing algorithm using the data store and outputs the physical machine ID for the VM creation service.

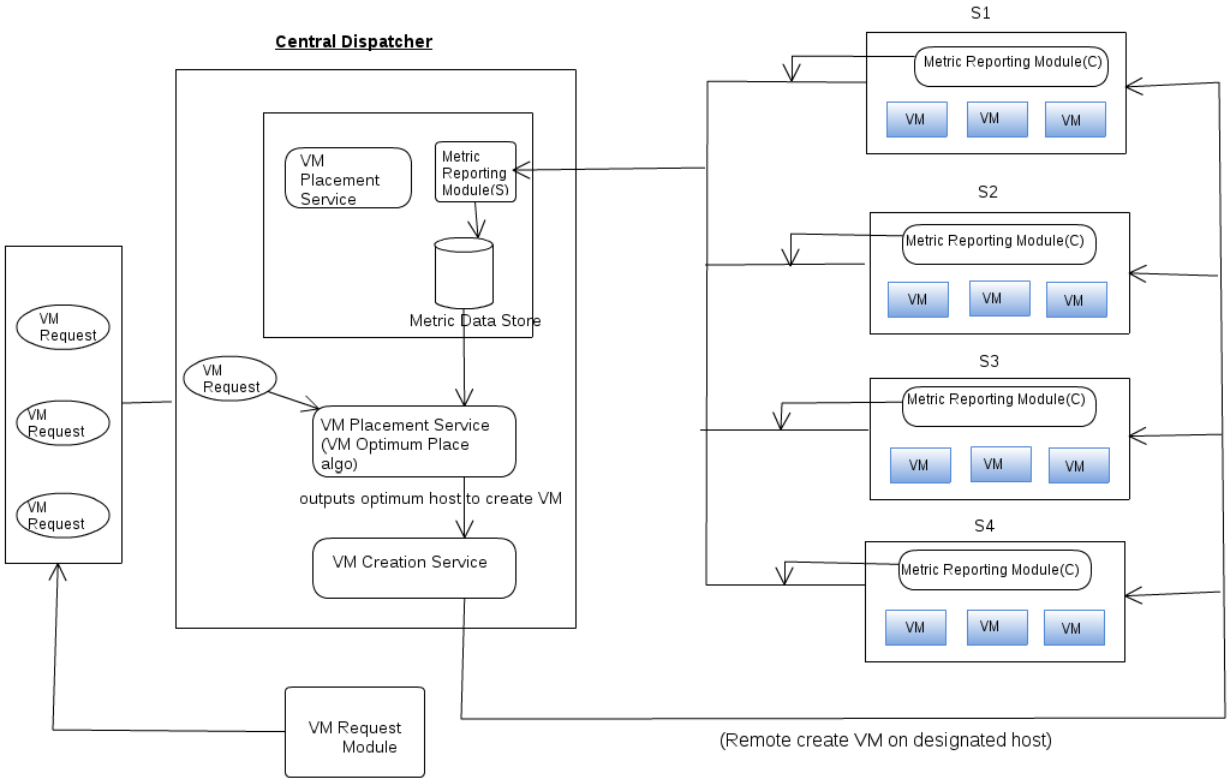

Figure 2: Architecture of Proposed Solution

6. VM Creation Service:

   This module will create the requested virtual machine on the physical machine identified by the VM Placement Service. For this we will be using the 'libvirt', open source libraries which provide an interface to communicate with all popular hypervisor's.

7. VM Request Module:

   This module is used to emulate client requests for VM creation. It will use a simple web page interface to specify a VM by the required parameters and request the central dispatcher to allocate a physical machine for it.

6 Development Plan

1. **Selection of an indicative metric**

   An indicative metric to represent performance snapshot of a physical machine is selected. The various candidates(CPU utilization, Memory Utilization, Disk I/O Stats, Network utilization) have to be analyzed and a final selection regarding one of them or a combination of them is to be made.

2. **Development of the Metric Reporting Module**

   The following components of this module are developed and integrated.

   Metric Data Store

   Metric Reporting Module-Server

   Metric Reporting Module-Client

3. **Development of VM Placement Service**

   An algorithm to select optimum physical machine for VM creation will be developed. The algorithm will take into account the performance metrics of all the physical machines and the requirement parameters of incoming VM request(we are assuming homogenous parameters initially) to decide upon the suitable physical machine.

4. **Development of VM Creation Service**

Once a physical machine is identified by the algorithm, we will create the requested virtual machine on the selected physical machine.

5. **VM Request Module**

   This module as described in the solution architecture will be developed.

6. **Integration**

   Integration of the system modules.

7 Milestones

Deadlines are defined in Table 1.

DATE	*TASK*
January 18	Uploading first draft of project goal document , first wiki update with the team leader chosen
January 28	Final draft of revised goal document, SVN repositories.
January 31	Selection and analysis of the metric to represent performance snapshot of the physical machine.
February 3	Development of VM Optimum placement algorithm.
February 5	Brief presentation of the project architecture and plans.
February 12	Developing a Metric Reporting Module for Client and Server.
February 17	Developing the VM Placement Service.
February 24	Developing the VM Creation Service and integration with the already developed modules.
March 4	Mid term review of the project with a working prototype.
March 25	Implementation of the VM Request Module and complete integration of the project.
April 11	Completion of project testing. Modifications if required.
April 18	Final review of project completion. Submission of necessary documents and reports.

Table 1: Meeting Deadlines

Bibliography

[1] *Xenserver*, `http://www.citrix.com/products/xenserver/overview.html`, [online].

[2] *Implementing microsoft network load balancing in a virtualized environment*, `http://www.vmware.com/files/pdf/implmenting_ms_network_load_balancing.pdf`, 2008, [online].

[3] Ali M. Alakeel, *A guide to dynamic load balancing in distributed computer systems*, IJCSNS Int. Journal of Computer Science and Network Security **10** (2010), 33–44.

[4] Archana B.Saxena and Deepti Sharma, *Analysis of threshold based centralized load balancing policy for heterogenous machines*, Int. Journal of Advanced Information Technology (IJAIT) **1** (2011), 39–53.

[5] Kalinka R. L. J. Castelo Branco et.al, *Piv and wpiv: Two new performance indices for heterogeneous systems evaluation*, Industrial Electronics, IEEE Int. Symp., 2006.

[6] Zaki et.al, *Customized dynamic load balancing for a network of workstations*, Proc. HPDC, 1996.

[7] Shahzad Malik, *Dynamic load balancing in a network of workstations*, `http://www.cs.toronto.edu/~smalik/downloads/paper_515.pdf`, November 2000, [online].