CHAPTER 1

INTRODUCTION

1.1 Introduction

Virtualization is emerging as a key mechanism of scaling the IT infrastructure and enabling enterprises to move from vertical silos of servers to horizontal pools of resources. Server virtualization provides the ability to slice larger, underutilized physical servers into smaller, virtual ones. Although virtualization has been around for more than three decades, it has found its way into the mainstream only recently, as a consequence of the recent developments in virtualization software and improved hardware support. A variety of solutions — both commercial and open source — are now available for commodity systems.

The motivations for enterprises to adopt virtualization technologies include increased flexibility, the ability to quickly re-purpose server capacity to better meet the needs of application workload owners, and to reduce overall costs of ownership. Virtualization services offer interfaces that support the life cycle management (e.g., create, destroy, move, size capacity) of VMs that are provided with access to shares of resource capacity (e.g., cpu, memory, input-output). Furthermore, some virtualization platforms provide the ability to dynamically migrate VMs from one physical machine to another without interrupting application execution. For large enterprises it offers an ideal solution for server and application consolidation.

Unfortunately, the complexity of these virtualized environments presents additional management challenges. In this research, resource allocation and capacity management problems in VM-based environments are looked upon. In such environments there are many different workloads, and a finite number can be hosted by each server. Each workload has capacity requirements that may frequently change based on business needs. VMs currently provide ideal fault isolation. In an enterprise environment, however, they must also provide performance and resource isolation; meaning that rogue services should not impact the performance of other applications that share the same infrastructure [1]. Cost effective capacity management methods are not yet available. Moreover, such capacity management methods critically depend on the characteristics of the resource allocation mechanisms of the underlying VM platform.

While the broader premise is that resource allocation for VMs is, in general, a hard problem, in this research the focus is on CPU scheduling. As a concrete example of the types of challenges involved, Barham, Phrasera and Warfield [2] have analyzed and compared the CPU schedulers in the Xen VMM in the context of traditional workload managers. Workload managers and

similar tools were, until a few years ago, known only to mainframers and users of large Unix environments. These technologies have their own requirements from the underlying resource management mechanisms (e.g., CPU scheduling of VMs). Using Xen and its evolution with different CPU schedulers, the challenges are demonstrated in choosing the appropriate scheduler features and parameters to support desirable application performance, as well as demonstrate the performance impact of these different choices. It has been found that, at least for the popular Xen environment, much work remains to be done in CPU scheduling before effective resource isolation guarantees can be delivered in large-scale deployments.

1.2 Motivation

Virtualization has been a commonly used technology for enterprise data centers, as it helps to manage the resources of the enterprise in an efficient manner. Various virtualization platforms are in use by different companies like VMWare, Hyper-V, Xen Server, KVM, etc. Xen Server is an open source and free virtualization platform and is widely used across many enterprises across the globe. As discussed previously, the challenge of current time is to use the virtualization platform without any complaints on performance issues. However, if the virtualization technology is not used with proper manner, then it brings in various problems and performance issues. Since Xen Server is a widely used virtualization platform, I was motivated to dig in to find out the various performance determining factors through several experiments and discover possible ways to optimize them for the performance tuning of xen server virtualized systems. This led to the motivation of development of a new CPU scheduling algorithm during the course of research which was an improved version of previous scheduling algorithm in the Xen Server environment.

1.3 Problem Statement

Virtualization is a relatively young technology and even though it is adopted in many enterprises, there are still lots of technical challenges that need to be overcome. Though technology has excelled with the virtualization technologies providing near to physical machine performance, performance degradation is still a problem in virtualized systems due to the presence of an extral

layer of hypervisor. Discovering the reason for performance degradation is not always easy and this alone can cause a lot of problem in a production environment. Performance must be tuned for different types of application workloads. Tuning for large files and continuous performance isn't optimal for small files and vice versa, and tuning for high performance of small files isn't best for large files and continuous performance. If one or two VMs are running especially I/O-intensive applications and hammering a disk, other VMs that utilize the same data store may suffer the effects of disk I/O contention. This problem is more difficult to identify than overloaded CPU or memory.

In case of Xen, of the most popular and widely used open source paravirtualized platform, the credit scheduler which is the default scheduler, is used to schedule the allocation of Physical CPUs (PCPU) to the virtual CPUs (VCPUs), on a proportionally fair basis, by providing credits to each VCPUs on the basis of their weights defined. The weights determine how much percentage of PCPU will be provided to the VCPUs of a domain in Xen Server. The credit scheduler is fair to all the running domains as it allocates the share based on the weights predefined. However, in case of varying workloads in different VMs, which is a common scenario in practical usage, it cannot dynamically determine, if a particular virtual machine needs more PCPU share than the other, based upon their respective workloads.

1.4 Objectives

The main objectives of this research work are as follows:

- To research and perform experiments to find out the effect of various parameters and environments that are responsible for the performance of virtual machines in Xen server.
- To develop an improvised CPU scheduling algorithm that improves the performance of the virtual machines in Xen server in varying workload conditions and verify the improved results with experiments.

1.5 Brief overview of the thesis

The first chapter of this thesis gives an introduction about the research and brief glimpse to to the emerging issues in technologies regarding virtualization, describes about the problems in the

subject matter, and clarifies the objectives of the research work. The remaining chapters in this thesis are organized as follows.

Chapter two provides the literature review regarding the subject matter. Various existing studies which were done in the field and relative to the study of our objectives were examined. Chapter 3 gives an introduction to the Virtual Machine Monitors (VMMs) and their different aspects to give an understaning about the Xen Server. In the following chapter 4, the schedulers are introuduced with their scheduling goals and various schedulers that are used by the Xen hypervisor are discussed. Chapter 5 explains the plans and methodologies through which this research work is guided for completion. Chapter 6 provides the results and observations observed from the experiments conducted as described in research metholodologies, and discussions are made based upon the observational results. Finally the conclusion of the research work and works that can be done for the further research in this topic are discussed in Chapter 7.