Elastic VM for Cloud Resources Provisioning Optimization

Wesam Dawoud, Ibrahim Takouna, and Christoph Meinel

Hasso Plattner Institute,
Potsdam University,
Potsdam, Germany
firstname.lastname@hpi.uni-potsdam.de

Abstract. Rapid growth of E-Business and frequent changes in websites contents as well as customers’ interest make it difficult to predict workload surge. To maintain a good quality of service (QoS), system administrators must provision enough resources to cope with workload fluctuations considering that resources over-provisioning reduces business profits while under-provisioning degrades performance. In this paper, we present elastic system architecture for dynamic resources management and applications optimization in virtualized environment. In our architecture, we have implemented three controllers for CPU, Memory, and Application. These controllers run in parallel to guarantee efficient resources allocation and optimize application performance on co-hosted VMs dynamically. We evaluated our architecture with extensive experiments and several setups; the results show that considering online optimization of application, with dynamic CPU and Memory allocation, can reduce service level objectives (SLOs) violation and maintain application performance...

Keywords: virtualization, consolidation, elasticity, application performance, automatic provisioning, optimization, cloud computing.

1 Introduction

Later advance in virtualization technology software, e.g. Xen [2] and VMware [16], enabled cloud computing environment to deliver agile, scalable, elastic, and low cost infrastructures, however, current implementation of elasticity in “Infrastructure as a Service” cloud model considers Virtual Machine (VM) as a scalability unit. In this paper, we developed an automated dynamic resources provisioning architecture to optimized resources provisioning in consolidated virtualized environments (e.g., Cloud computing). Unlike current implementation of elasticity in cloud infrastructure, we replaced the VM (as a coarse-grain scalability unit) with fine-grain resources units (i.e. %CPU as a share, Memory as MB). Our Elastic VM is scaled dynamically in-place to cope with workload fluctuations, furthermore, the hosted application is also tuned after each scaling to maintain predetermined (SLOs). As a use case we implemented our approach
into Xen environment and used Apache web server as an application, our SLO in this paper is to keep the response time of the web requests less than a specified threshold. Nevertheless, our architecture could be extended for any application that has tunable parameters such as Database applications. The key contributions of this work are as follow: First, we have studied Apache application performance under different configuration and different CPU and Memory allocation values. Second, we have developed a dynamic application optimization controller for Apache application to maintain the desired performance. Third, we built CPU and Memory controllers based on [6]. Fourth, we built elastic system architecture that join CPU, Memory, and application optimization controllers for elastic consolidated virtualized environments. Finally, the elastic system architecture has been evaluated with extensive experiments on several synthetic workload and experimental setups, experiments also have included real workload demand requests. Our results show that elastic system architecture can guarantee the best performance for application in terms of throughput and response time. The rest of the paper is organized as follow. Section 2 study the systems and concepts that drive our research. In section 3 we describe our elastic system architecture. Section 4 provides literature review for related work. In section 5, we describe our experimental setup and analyze results.

2 Overview

In this section, we give an overview of systems and concepts that drive our research; we will start with a detailed study of Apache server, then will discuss the complexity of enforcing SLOs into consolidated environments (e.g. clouds), and finally will explain concerns that accompany using feedback control systems in computing systems.

2.1 Apache Server

Apache [1], is structured as a pool of workers processes that handle HTTP requests. Currently, Apache supports two kinds of modules, workers and prefork modules. In our experiments we use Apache with prefork module to handle dynamic requests (e.g., php pages). In prefork mode, requests enter the TCP Accept Queue where they wait for a worker. A worker processes a single request to completion before accepting a new request. Number of worker processes is limited by \( \text{MaxClients} \) parameter.

Figure 1 displays the result of experiments in which Apache is configured with different settings of Memory, traffic rate, and \( \text{MaxClients} \). By monitoring the throughput, we notice that, there is a value of \( \text{MaxClients} \), (e.g. 75), which gives the highest throughput (450 req/sec) for specific Memory settings (512MB). Before this value there is no enough workers to handle requests, and after this value, performance regrades because of one of the following problems: CPU spend much time switching between many process or Memory is full so paging to harddisk consumes most of CPU time. Our heuristic Apache controller job is to find this optimum value dynamically.