Future Generation Computer Systems 27 (2011) 871–879

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Adaptive resource provisioning for read intensive multi-tier applications in the cloud

Waheed Iqbal [a,*], Matthew N. Dailey [a], David Carrera [b], Paul Janecek [a]

[a] Computer Science and Information Management, Asian Institute of Technology, Thailand
[b] Technical University of Catalonia (UPC), Barcelona Supercomputing Center (BSC), Spain

ARTICLE INFO

Article history:
Received 15 June 2010
Received in revised form
21 October 2010
Accepted 25 October 2010
Available online 4 November 2010

Keywords:
Cloud computing
Adaptive resource management
Quality of service
Multi-tier applications
Service-Level Agreement
Scalability

ABSTRACT

A Service-Level Agreement (SLA) provides surety for specific quality attributes to the consumers of services. However, current SLAs offered by cloud infrastructure providers do not address *response time*, which, from the user's point of view, is the most important quality attribute for Web applications. Satisfying a maximum average response time guarantee for Web applications is difficult for two main reasons: first, traffic patterns are highly dynamic and difficult to predict accurately; second, the complex nature of multi-tier Web applications increases the difficulty of identifying bottlenecks and resolving them automatically. This paper proposes a methodology and presents a working prototype system for automatic detection and resolution of bottlenecks in a multi-tier Web application hosted on a cloud in order to satisfy specific maximum response time requirements. It also proposes a method for identifying and retracting over-provisioned resources in multi-tier cloud-hosted Web applications. We demonstrate the feasibility of the approach in an experimental evaluation with a testbed EUCALYPTUS-based cloud and a synthetic workload. Automatic bottleneck detection and resolution under dynamic resource management has the potential to enable cloud infrastructure providers to provide SLAs for Web applications that guarantee specific response time requirements while minimizing resource utilization.

1. Introduction

Cloud providers [1] use the infrastructure as a service model to allow consumers to rent computational and storage resources on demand and according to their usage. Cloud infrastructure providers maximize their profits by fulfilling their obligations to consumers with minimal infrastructure and maximal resource utilization.

Although most cloud infrastructure providers provide Service-Level Agreements (SLAs) for availability or other quality attributes, the most important quality attribute for Web applications from the user's point of view, *response time*, is not addressed by current SLAs. Guaranteeing response time is a difficult problem for two main reasons. First, Web application traffic is highly dynamic and difficult to predict accurately. Second, the complex nature of multi-tier Web applications, in which bottlenecks can occur at multiple points, means response time violations may not be easy to diagnose or remedy. It is also difficult to determine optimal static resource allocation for multi-tier Web applications manually for certain workloads due to the dynamic nature of incoming requests and exponential number of possible allocation strategies. Therefore, if a cloud infrastructure provider is to guarantee a particular maximum response time for any traffic level, it must automatically detect bottleneck tiers and allocate additional resources to those tiers as traffic grows.

In this paper, we take steps toward eliminating this limitation of current cloud-based Web application hosting SLAs. We propose a methodology and present a working prototype system running on a EUCALYPTUS-based [2] cloud that actively monitors the response times for requests to a multi-tier Web application, gathers CPU usage statistics, and uses heuristics to identify the bottlenecks. When bottlenecks are identified, the system dynamically allocates the resources required by the application to resolve the identified bottlenecks and maintain response time requirements. The system furthermore predicts the optimal configuration for the dynamically varying workload and scales down the configuration whenever possible to minimize resource utilization.

The bottleneck resolution method is purely *reactive*. Reactive bottleneck resolution has the benefit of avoiding inaccurate a priori performance models and pre-deployment profiling. In contrast, the scale-down method is necessarily *predictive*, since we must avoid premature release of busy resources. However, the predictive model is built using application performance statistics acquired *while the application is running* under real-world traffic loads, so it

* Corresponding author.
E-mail address: waheed751@gmail.com (W. Iqbal).

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.10.016

neither suffers from the inaccuracy of a priori models nor requires pre-deployment profiling.

In this paper, we describe our prototype, the heuristics we have developed for reactive scale-up of multi-tier Web applications, the predictive models we have developed for scale-down, and an evaluation of the prototype on a testbed cloud. The evaluation uses a specific two-tier Web application consisting of a Web server tier and a database tier. In this context, the resources to be minimized are the number of Web servers in the Web server tier and the number of replicas in the database tier. We find that the system is able to detect bottlenecks, resolve them using adaptive resource allocation, satisfy the SLA, and free up over-provisioned resources as soon as they are not required.

There are a few limitations to this preliminary work. We only address scaling of the Web server tier and a read-only database tier. Our system only performs hardware and virtual resource management for applications. In particular, we do not address software configuration management; for example, we assume that the number of connections from each server in the Web server tier to the database tier is sufficient for the given workload. Additionally, real-world cloud infrastructure providers using our approach to response time-driven SLAs would need to protect themselves with detailed contracts (imagine for example the rogue application owner who purposefully inserts delays in order to force SLA violations). We plan to address some of these limitations in future work.

In the rest of this paper, we provide related work, then we describe our approach, the prototype implementation, and an experimental evaluation of the prototype.

2. Related work

There has been a great deal of research on dynamic resource allocation for physical and virtual machines and clusters of virtual machines [3]. In [4,5], a two-level control loop is proposed to make resource allocation decisions within a single physical machine. This work does not address integrated management of a collection of physical machines. The authors of [6] study the overhead of a dynamic allocation scheme that relies on virtualization as opposed to static resource allocation. None of these techniques provide a technology to dynamically adjust allocation based on SLA objectives in the presence of resource contention.

VMware DRS [7] provides technology to automatically adjust the amount of physical resources available to VMs based on defined policies. This is achieved using the live-migration automation mechanism provided by VMotion. VMware DRS adopts a VM-centric view of the system: policies and priorities are configured on a VM-level.

A approach similar to VMware DRS is proposed in [8], which proposes a dynamic adaptation technique based on rearranging VMs so as to minimize the number of physical machines used. The application awareness is limited to configuring physical machine utilization thresholds based on off-line analysis of application performance as a function of machine utilization. In all of this work, runtime requirements of VMs are taken as given and there is no explicit mechanism to tune resource consumption by any given VM.

Foster et al. [9] address the problem of deploying a cluster of virtual machines with given resource configurations across a set of physical machines. Czajkowski et al. [10] define a Java API permitting developers to monitor and manage a cluster of Java VMs and to define resource allocation policies for such clusters.

Unlike [7,8], our system takes an application-centric approach; the virtual machine is considered only as a container in which an application is deployed. Using knowledge of application workload and performance goals, we can utilize a more versatile set of automation mechanisms than [7–9] or [10].

Network bandwidth allocation issues in the deployment of clusters of virtual machines has also been studied in [11]. The problem there, is to place virtual machines interconnected using virtual networks on physical servers interconnected using a wide area network. VMs may be migrated, but rather than resource scaling, the emphasis is on allocating network bandwidth for the virtual networks. In contrast, our focus is on data center environments, in which network bandwidth is of lesser concern.

There have been several efforts to perform dynamic scaling of Web applications based on workload monitoring. Amazon Auto Scaling [12] allows consumers to scale up or down according to criteria such as average CPU utilization across a group of compute instances. [13] presents the design of an auto-scaling solution based on incoming traffic analysis for Axis2 Web services running on Amazon EC2. [14] presents a statistical machine learning approach to predict system performance and minimize the number of resources required to maintain the performance of an application hosted on a cloud. [15] monitors the CPU and bandwidth usage of virtual machines hosted on an Amazon EC2 cloud, identifies the resource requirements of applications, and dynamically switches between different virtual machine configurations to satisfy the changing workloads. However, none of these solutions address the issues of multi-tier Web applications or database scalability, a crucial step to dynamically manage multi-tier workloads.

Thus far, only a few researchers have addressed the problem of resource provisioning for multi-tier applications. [16] presents an analytical model using queuing networks to capture the behavior of each tier. The model is able to predict the mean response time for a specific workload given several parameters such as the *visit ratio*, *service time*, and *think time*. However, the authors do not apply their approach toward dynamic resource management on clouds. [17] presents a predictive and reactive approach using queuing theory to address dynamic provisioning for multi-tier applications. The predictive approach is to allocate resources to applications on large time scales such as days and hours, while the reactive approach is used for short time scales such as seconds and minutes. This allows the system to overcome the "flash crowd" phenomenon and correct prediction mistakes made by the predictive model. The technique assumes knowledge of the resource demands of each tier. In addition to the queuing model, the authors also provide a simple black-box approach for dynamic provisioning that scales up all replicable tiers when bottlenecks are detected. However, this work does not address database scalability or releasing of application resources when they are not required. In contrast, our system classifies requests as either dynamic or static and uses a black box heuristic technique to scale up and scale down only one tier at a time. Our scale-up system is reactive in resolving bottlenecks and our scale-down system is predictive in releasing resources.

The most recent work in this area [18] presents a technique to model dynamic workloads for multi-tier Web applications using *k*-means clustering. The method uses queuing theory to model the system's reaction to the workload and to identify the number of instances required for an Amazon EC2 cloud to perform well under a given workload. Although this work does model system behavior on a per-tier basis, it does not perform multi-tier dynamic resource provisioning. In particular, database tier scaling is not considered.

In our own recent work [19], we consider single-tier Web applications, use log-based monitoring to identify SLA violations, and use dynamic resource allocation to satisfy SLAs. In [20], we consider multi-tier Web applications and propose an algorithm based on heuristics to identify the bottlenecks. This work uses a simple reactive technique to scale up multi-tier Web applications