SLA-Driven Dynamic Resource Management for Multi-tier Web Applications in a Cloud

Waheed Iqbal, Matthew N. Dailey
Computer Science and Information Management
Asian Institute of Technology, Thailand
Email: waheed.iqbal, mdailey@ait.ac.th

David Carrera
Technical University of Catalonia
Barcelona Supercomputing Center, Spain
Email: dccarrera@ac.upc.edu

Abstract—Current service-level agreements (SLAs) offered by cloud providers do not make guarantees about response time of Web applications hosted on the cloud. Satisfying a maximum average response time guarantee for Web applications is difficult due to unpredictable traffic patterns. The complex nature of multi-tier Web applications increases the difficulty of identifying bottlenecks and resolving them automatically. It may be possible to minimize the probability that tiers (hosted on virtual machines) become bottlenecks by optimizing the placement of the virtual machines in a cloud. This research focuses on enabling clouds to offer multi-tier Web application owners maximum response time guarantees while minimizing resource utilization. We present our basic approach, preliminary experiments, and results on a EUCALYPTUS-based testbed cloud. Our preliminary results show that dynamic bottleneck detection and resolution for multi-tier Web application hosted on the cloud will help to offer SLAs that can offer response time guarantees.

Keywords—cloud computing; service level agreement; multi-tier Web applications; dynamic resource management; application placement; bottleneck detection; bottleneck resolution

I. INTRODUCTION

Many IT giants such as IBM, Sun, Amazon, Google, and Microsoft offer computational and storage resource rental services to consumers. Consumers of these services host applications and store data for business or personal needs. The key features of these services, on-demand resource provisioning and pay-per-use, mean that consumers only need to pay for the resources they actually utilize. In this environment, cloud service providers must maximize their profits by fulfilling their obligations to consumers with minimal infrastructure and maximal resource utilization.

Although most cloud 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. The reason for this is obvious: Web application traffic is highly unpredictable, and response time depends on many factors, so guaranteeing a particular maximum response time for any traffic level would be suicide for the cloud provider unless it had the ability to dynamically and automatically allocate additional resources to the application as traffic grows.

Usually cloud consumers reserve resources without any optimal prediction or estimation, leading to either underutilization or overutilization of reserved resources. Web application owners, in order to ensure the usability of their applications, need to maintain a minimum response time for their end users. Web application owners should be able to host their applications on a cloud by specifying a desired Quality of Service (QoS) in terms of response time. Cloud providers should ensure these QoS requirements using a minimal amount of computational resources.

A typical Web application consists of three tiers: a Web server tier, an application server tier, and a database server tier. Every tier may place a heavy demand on specific resources. For example, a database tier requires high disk throughput, an application server requires a large amount of CPU time, and a Web server tier requires high network bandwidth to handle incoming traffic. The Web application tier depends on one or more of the other tiers; therefore, whenever a tier becomes a bottleneck, the efficiency of dependent tiers also degraded the whole application’s performance starts decreasing dramatically.

Cloud infrastructure builds on virtualization platforms such as Xen, KVM, and VMWare. Virtualization provides flexible mechanisms to dynamically provision more hardware resources to the VMs. To maintain an average response time for an application hosted on a cloud, it is necessary for cloud providers to dynamically detect violations of response time requirements, identify the bottleneck tier(s), and ensure bottleneck resolution by using the best possible solution. Although some tiers of a multi-tier Web application hosted on virtual machines are easy to replicate and scale horizontally with load balancing, some tiers are impossible or difficult to replicate. Examples include a tier uses a database that does not support clustering or synchronization, a tier that uses an expensive software license, a tier that does not allow distributing workload, and a tier that is designed as a singleton. There are other possible solutions for vertical scaling such tiers, such as migrating a virtual machine to another physical host or dynamically increasing
the hardware allocated to the virtual machine.

Virtual machine placement is a very important concept for clouds. A suitable physical host is chosen on the creation or activation time of a virtual machine. Cloud middleware needs to place virtual machines on existing physical machines in such a way as to achieve good utilization of physical resources and maximize revenue. It is possible to reduce the chances of tiers (hosted in virtual machines) to become bottleneck by using an optimal placement model for virtual machines placement in a cloud. Mapping a virtual machine onto physical machine requires a prior knowledge of physical machines capacity and requirements of the virtual machine. It is difficult to determine the exact resource requirements for a virtual machine due to the dynamic workloads. Therefore, it is necessary that system should identify the best place for the virtual machines in the beginning and capable to alter the place of virtual machine dynamically for satisfying the dynamic workload needs.

In the rest of this paper, we describe specific objectives, research methodology, related work, and preliminary experiments.

II. Specific Objectives

This research will focus on enabling clouds to offer multi-tier Web application owners maximum response time guarantees while minimizing resource utilization. Currently, neither the commercial cloud providers nor the existing open source frameworks support maximum response time guarantees. Following are the specific objectives that we will address in this dissertation:

- Design and implement algorithms for bottleneck detection in multi-tier Web applications hosted on a cloud.
- Develop prototype system for automatic bottleneck resolution for multi-tier Web applications hosted on a cloud.
- Propose a model for the application placement problem in the scope of multi-tier applications running in a large heterogeneous cloud.
- Design and implement algorithms to optimize application placement problem in the scope of multi-tier applications under the proposed model.
- Evaluate proposed solutions to the application placement problem by running simulations.
- Evaluate the entire system using benchmark Web applications and realistic workloads.

III. Methodology

We will design algorithms capable of identifying the bottleneck tiers in multi-tier Web applications hosted on a cloud. We can identify the bottleneck tier by profiling the CPU, memory, and I/O resource usage of each tier with a combination of real time monitoring and processing of each tier’s log files. Most of Web application tiers need to configure concurrency level, for example a typical three-tier Web application may need to configure number of concurrent connections in a database tier, number of threads in an application server tier, and number of worker processes in a Web server tier. Incorrect configuration of these parameters may also cause a tier to become a bottleneck without saturating its resources. To address this issue, we will identify heuristics and incorporate them into the proposed algorithms. We also need to minimize the overhead of active profiling of system resources.

We will develop a prototype system based on the proposed algorithms for bottleneck detection. The possible techniques to resolve a bottleneck in a tier are horizontal scaling the tier with load balancing, vertical scaling the tier by migration to another physical machine, and vertical scaling the tier by dynamically increasing the hardware allocated. The prototype system will identify and use the best possible solution to resolve the bottleneck.

If resource utilization of tiers are known, we could reduce the application placement problem to bin-packing. Jungy et al. [1] and Wood et al. [2] has modeled application placement problem and reduce it to the bin-packing. We will formalize the application placement problem with an additional constraint (tiers of an application should not place on the same physical machine) in the scope of multi-tier applications running in a cloud and identify the problem class. After formalizing and identifying the problem class, we will propose a model to obtain the best place for the application tiers in the cloud. The proposed model will receive current state of the cloud as well as the tier types with associated performance goals as inputs and identify the best place for the specific tiers. We will run simulations to evaluate the proposed solution in the scope of multi-tier applications running in a cloud with thousands of physical machines.

Finally, we will integrate bottleneck detection, bottleneck resolution, and application placement problem. Then evaluate our proposed system using available benchmark Web applications with realistic workloads.

IV. Related Work

Andreolini et al. [3] present an algorithm to reallocate the placement of virtual machines in a cloud for better performance and resource utilization. Periodically, their algorithm identifies three sets of machines, senders (the most overutilized physical machines), receivers (the most underutilized physical machines), and guests to migrate (the most overutilized virtual machines hosted on the senders). The algorithm only considers CPU utilization to identify the overutilized and underutilized machines.

Jung et al. [4] use a queuing network model to identify the optimal configuration of a multi-tier application hosted in a virtualized data center. However, they pre-deploy the application then identify the values of required parameters for their model. An online controller ensures that any failure
of a tier is resolved without degradation of application performance. Whenever a failure is detected, they use a simple select-reject technique to identify where to launch the tier. Their select-reject technique iteratively identifies a physical machine that has enough CPU capacity to host the tier and, to maximize fault tolerance, does not host another instance of the same tier.

Sandpiper [2] is a system for identification and mitigation of hotspot (overloaded) virtual machines in a Xen-based virtualized data center. It gathers usage statistics for physical and virtual machines. These profiles are used with prediction techniques to identify the hotspots. Then, it identifies the requirements of the overloaded virtual machines and use a greedy algorithm to migrate virtual machines to the appropriate physical machines.

Bodik et al. [5] present 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. They predict the next 5 minutes workload using linear regression based on the last 15 minutes of traffic. Then, the predicted workload is used as an input to a performance model that estimates the number of servers required to handle the predicted workload. The system adjusts the performance model using machine learning techniques such as online training and change point detection. However, they do not address the issues of multi-tier Web applications.

Urgaonkar et al. [6] present an analytical model using queuing networks to capture the behavior of tiers. The model is able to predict the mean response time for the given workload. To compute the mean response time for a given workload, the model requires several parameters such as the visit ratio, service times, and think time. They processed the log files of tiers to calculate these parameters.

Villela et al. [7] argue that the optimal resource allocation to the application server tier in a multi-tier e-commerce Web application hosting environment will increase the profit of service provider. They identify the request arrival process as a Poisson process by analyzing the traces of a real Web application and formalized the resource allocation problem to maximize the profit of service providers by optimal resource allocation to the application server tier. They present and evaluate three different approximation methods for optimal resource allocation using simulations.

Dubeyb et al. [8] present the initial results of dynamic regression and queuing modeling techniques to obtain the approximate system performance model for multi-tier Web application hosting on virtualized data centers. They use the dayTrader [9] benchmark Web application to evaluate their prototype system on a Xen-based virtualization platform.

Jungy et al. [1] present off-line techniques to generate adaptation policies for multi-tier applications hosted on virtualized data center. The purpose of an adaptation policy is to provide optimal configurations of an application for the given workload. They present a queuing model with optimization techniques to generate optimal system configuration for a multi-tier application. Their model is able to identify the number of replicas for tiers, tiers placement, and CPU allocation for tiers.

V. PRELIMINARY INVESTIGATIONS AND EXPERIMENTS

As previously described in section I that current service-level agreements (SLAs) offered by cloud providers make guarantees about quality attributes such as availability. However, although one of the most important quality attributes from the perspective of the users of a cloud-based Web application is its response time, current SLAs do not guarantee response time. Satisfying a maximum average response time guarantee for Web applications is difficult due to unpredictable traffic patterns, but in [10] we show how it can be accomplished through dynamic resource allocation in a virtual Web farm. We present the design and implementation of a working prototype built on a EUCALYPTUS-based heterogeneous cloud that actively monitors the response time of each virtual machine assigned to the farm and adaptively scales up the application to satisfy a SLA promising a specific average response time. We demonstrate the feasibility of the approach in an experimental evaluation with a testbed cloud and a synthetic workload. However, we only considered a simple one tier Web application.

More recently, we have extended our work and considered a typical Web application consisting on two tiers, a Web server tier and a database tier, hosted on the heterogeneous compute cloud. We propose a solution for automatic identification and resolution of the bottleneck tier by horizontally scaling to satisfy the response time requirements. We assume that a mechanism such as GridScale [11] to ensure consistent reads after updates to the master exists. Our focus, however, is on read-intensive applications. We use heuristics and active profiling of the CPU of virtual machines hosted application tiers for identification of the bottleneck. Our system reads the Web server proxy logs for 60 seconds and clusters the log entries into dynamic content requests and static content requests. Then system calculates the 95th percentile of the average response time and scales the Web server tier in case of static content response time saturation. If the system determines that dynamic content response time is saturated, then it obtains the CPU utilization of the Web server tier. If the CPU utilization of any instance in the Web server tier reaches the saturation point the system scales up the Web server tier; otherwise it scales up the database tier. Before initiating a scale operation, our system ensures that the effect of the last scale operation has been realized. Figure 1 shows a flow diagram for bottleneck detection and resolution in our prototype system.

We evaluate our proposed system on a EUCALYPTUS-based testbed cloud consisting on five heterogeneous machines using a PHP implementation of RUBiS[12], an open
source benchmark Web application for auctions. We use httpperf to generate synthetic workload for the experiments. We generate workload for a specific duration with a required number of user sessions per second. After every five minutes, we step load level by incrementing 5 from load level 5 through load level 60. Each load level represents the number of user sessions per second; each user session makes 6 requests to static resources and 5 requests to dynamic resources including 5 pauses to simulate user think time. Dynamic resources consist on PHP pages that makes read-only database queries. Figure 2 shows the workload generation. We performed two experiments based on this workload and the RUBiS application.

A. Experiment 1: Static allocation

Experiment 1 profiles the system’s behavior with static allocation of resources to the application. In this experiment, two virtual machines are allocated to the application. Therefore each tier hosts on separate virtual machine.

B. Experiment 2: Automatic bottleneck detection and resolution under dynamic allocation

Experiment 2 profiles the system’s behavior using automatic bottleneck detection and resolution under dynamic allocation of resources to the application. In this experiment, we try to satisfy a SLA that enforces one-second maximum average response time requirements for static and dynamic resources of the RUBiS application regardless of load level. Initially, each tier hosts on separate virtual machine and system dynamically add more virtual machines to satisfy the SLA.

In this experiment, we use Nginx [13] as a HTTP load balancer because it offers detailed logging and allows reloading of its configuration file without termination of existing client sessions. We monitor QoS attributes (response time requirements) using Nginx proxy traces.

VI. PRELIMINARY RESULTS

This section describes the results we obtained in Experiment 1 and Experiment 2.

A. Experiment 1: Static allocation

Figure 3 shows the throughput of the system during Experiment 1. After load level 25, we do not observe any growth in the number of served requests because one or both of the tier(s) reaches its saturation point. Although the load level increases with time, the system is unable to serve all requests and it either rejects or queues the remaining requests.

Figure 4 shows the CPU utilization of virtual machines hosting the application tiers during Experiment 1 using static resource allocation. It is clear that both both of the virtual machine never utilized more then 75% of the CPU during this experiment.

Figure 5 shows the 95th percentile of average response time during Experiment 1 using static allocation. From load level 1 to load level 25, we observe a nearly constant
response time, but after load level 25, the arrival rate exceeds the limit of the system processing capability. One of the virtual machines hosting the application tier becomes a bottleneck, then requests begin to spend more time in the queue and request processing time increases. From that point we observe a rapid growth in the response time. After load level 30, however, the queue also becomes saturated, and the system rejects most requests. Therefore, we do not observe further growth in the average response time. Clearly, we cannot provide a SLA guaranteeing a specific response time with an undefined load level for a Web application using static resource allocation.

B. Experiment 2: Automatic bottleneck detection and resolution under dynamic Allocation

This section describes the result of Experiment 2. Figure 6 shows the system throughput of the system over time. We observe linear growth in the system throughput with each load level. Figure 8 shows the CPU utilization of each virtual machine over the experiment. During the load level 25 VM3 is dynamically added to the system and during the load level 30 VM4 is dynamically added to the system for satisfying the response time requirements. Different CPU utilization and the time required to encapsulate the effect of the last scale operation.

Figure 7 shows the system throughput of the system over time. We observe linear growth in the system throughput with each load level. Figure 8 shows the CPU utilization of each virtual machine over the experiment. During the load level 25 VM3 is dynamically added to the system and during the load level 30 VM4 is dynamically added to the system for satisfying the response time requirements. Different CPU utilization
for different VMs reflects the differing processor speeds for the physical nodes in the testbed cloud.

The experiments show that dynamic bottleneck detection and resolution under dynamic resource management on clouds for multi-tier Web applications would allow us to offer SLAs that enforce specific response time requirements.

VII. CONCLUSION AND WORK PLAN

In this paper, we have described the problem of bottleneck detection and resolution for multi-tier Web applications hosted on a cloud and presented our initial investigations. The initial experimental results show that automatic bottleneck detection and resolution will enable us to offer SLAs that defines specific maximum response time requirements.

We are confident to obtain more promising results during this Ph.D. dissertation. We are extending our prototype system to support n-tier Web applications and intend to evaluate the system on different benchmark Web applications with realistic workloads. The work plan for this Ph.D. research is given in Table I.

REFERENCES


