2012 IEEE Fifth International Conference on Cloud Computing

Automatic Resource Scaling Based on Application Service Requirements

Ching-Chi Lin Jan-Jan Wu
Institute of Information Science
Research Center for Information Technology Innovation
Academia Sinica, Taipei, Taiwan
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
Email: {deathsimon, wuj}@iis.sinica.edu.tw

Jeng-An Lin Li-Chung Song Pangfeng Liu
Department of Computer Science
and Information Engineering
Graduate Institute of Networking and Multimedia
National Taiwan University
Taipei, Taiwan
Email: {r99944038,r00922089,pangfeng}@csie.ntu.edu.tw

***Abstract*—Web applications play a major role in various enterprise and cloud services. With the popularity of social networks and with the speed at which information can be disseminate around the globe, online systems need to face ever-growing, unpredictable peak load events.**

***Auto-scaling* technique provides on-demand resources according to workload in cloud computing system. However, most of the existing solutions are subject to some of the following constraints: (1) replying on user-provided scaling metrics and threshold values, (2) employing the simple Majority Vote scaling algorithm, which is ineffective for scaling Web applications, and (3) lack of capability for predicting workload changes.**

In this work, we develop an auto-scaling system, *WebScale*, which is not subject to the aforementioned constraints, for managing resources for Web applications in datacenters. We also compare the efficiency of different scaling algorithms for Web applications, and devise a new method for analyzing the trend of workload changes. The experiment results demonstrate that *WebScale* can keep the response time of Web applications low even when facing sudden load changing.

***Keywords*-Cloud Computing, Web Applications, Resource Provisioning, Auto-Scaling, Trend Analysis, Virtual Machines.**

I. Introduction

Web applications play a major role in various enterprise and cloud services. Many Web applications, such as eBanking, eCommerce and online gaming, face fluctuating loads. Some of the loads are predictable, such as the workload around a holiday for eShopping services.

The basic idea of *auto-scaling* is to estimate the load for short window of time, and then be able to up-scale or down-scale the resources when there is a need for it. *Auto-scaling* not only maintains application service quality but also reduces wasted resources.

Many cloud services, such as Amazon EC2 [1] and Google App [4] Engine, have proposed auto-scaling service. Other softwares such as Scalr and RightScale provide auto-scaling mechanism that can apply to cloud environments. However, most of the existing solutions are subject to some of the following constraints: (1) replying on user-provided scaling metrics and threshold values, (2) employing the simple *Majority Vote* scaling algorithm, which we will show in this paper to be ineffective for scaling Web applications, and (3) lack of capability for predicting workload change, and thus may result in unnecessary scaling actions.

In this paper, we develop an auto-scaling system, *WebScale*, which is not subject to the aforementioned constraints. *WebScale* monitors the behavior of the applications and the system, and based on the collected metrics, decides in real time whether the number of VMs for an application needs to be increased or decreased. A pool of available VMs is provided to facilitate real-time configuring and releasing of resources in an on-demand basis. We devise a *Workload-Based* scaling algorithm, which does not require user-provided metrics and threshold values. We also propose a algorithm to analyze the trend of workload changes.

The main contributions of this work are as follows. (1) We develop a modularized auto scaling system, *WebScale*, for dynamic resource provision in data centers. (2) We propose an algorithm to analyze the *trend* of workload change in a Web application. This trend analysis algorithm can significantly reduce the number of peaks (longer than 2 seconds) in response time caused by workload fluctuating. (3) Our experiment results with workload generated by *httperf* demonstrate that *WebScale* can keep the average response time of Web applications within 2 seconds even when facing sudden load changing.

II. Scaling Algorithms

Majority Vote selects the choice with the most properties among all choices. There are three choices, **scale in**, **scale out**, and **no scale**, for a VM when making decision. Each VM makes their choice according to their current loading, and a final scaling decision will be made by majority vote. *Majority Vote* scales one VM at a time.

Workload-Based Algorithm determines the number of running VMs needed for the business-logic tier and the number of database servers needed for the data-access tier, based on the incoming workload. The latter is the number of requests per second for the business-logic tier, and the number of SQL queries per second for the data-access tier.

978-0-7695-4755-8/12 $26.00 © 2012 IEEE
DOI 10.1109/CLOUD.2012.32

Our workload-based algorithm works as follows. For every fixed time interval, the decision maker asks the load balancer for the current workload. The number of VMs needed to handle current requests is calculated, and scaling decision is made by comparing that number with current number of running VMs.

Trend Analysis Algorithm works as a helper to the scaling algorithms by providing workload trend information to the scaling algorithms to make more ”correct” decision while handling workloads with periodic behaviors. Periodic behavior means that similar behavior of the workload shows up every fixed length of time, thus we can predict the coming workload by historical data. Workload prediction has been studied by some previous works [2], [3].

Instead of accurately predicting the workload value, for auto-scaling, it suffices to only predict the *trend* of workload change. *Trend* is the **direction** of workload changing in a fixed size of time.

Trend Analysis Algorithm works as follow. If a *scale in* decision “conflicts” with the trend, i.e., the decision is *scale in* while the trend is *scale out*, then the decision will be canceled. The rationale is to avoid removing VMs during workload increasing.

III. Experiment Results

Our experiment environment consists of 24 physical servers, each with 4 core X5460 CPU * 2 with hyper-threading, 16 GB memory, and 250 GB disk. The hypervisor is Xen 4.1 and the OS of domain 0 is Gentoo. All the VMs use Gentoo OS. We set up MediaWiki as the application benchmark. We use Httperf as our performance measuring tool.

The workload we used in the experiment is the workload from the log of *Judgegirl*, an online grading system for teaching purpose in department of CSIE, NTU. In the record, each data point represents the load in fifteen minutes. We shrink this length into thirty seconds and the result is shown in Figure 1.

Figure 2 depict the average response time using different scaling algorithms. The red line indicates the SLA requirement, which is 2 seconds here. This result shows that majority vote is not an effective scaling algorithm for web applications with frequently changing workloads. On the other hand, workload-based outperforms majority vote. Furthermore, workload-based with trend analysis has lesser response time bursts than without trend analysis after the first pattern(time 48). This make workload-based with trend analysis performs better.

IV. Conclusion

Auto-scaling technique provides on-demand resources according to workload in cloud computing system. In this work, we implemented an auto-scaling system for 3-tier web applications, *WebScale*. We also conduct experiments to evaluate the performance of different scaling algorithms. The results show that for workloads with periodic behavior, using workload-based algorithm with trend analysis performs the best among all three strategies. Slight sudden workload change will not affect workload-based algorithm with trend analysis.

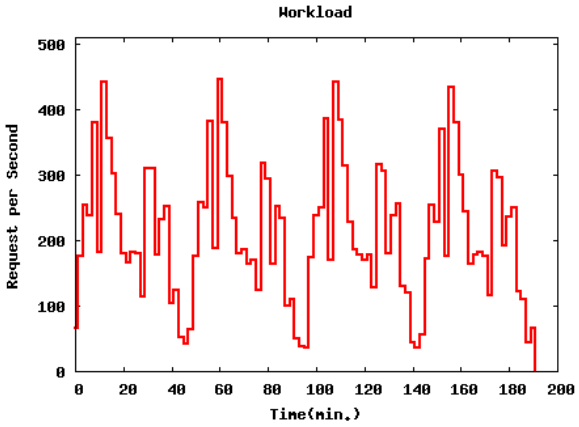

Figure 1. PREDICTABLE workload

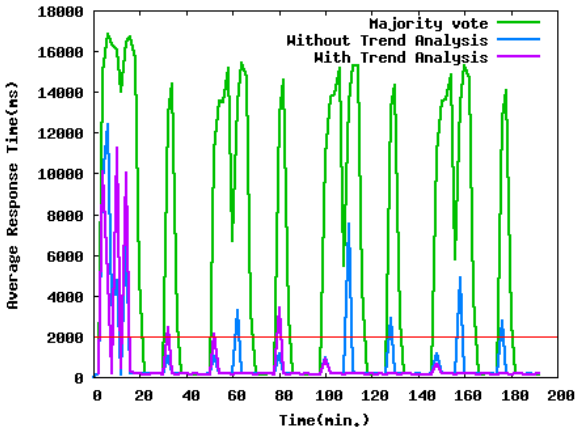

Figure 2. Average response time results under PREDICTABLE workload

References

[1] Amazon elastic compute cloud. http://aws.amazon.com/ec2/.

[2] E. Caron, F. Desprez, and A. Muresan. Forecasting for grid and cloud computing on-demand resources based on pattern matching. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 456–463, December 2010.

[3] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Workload analysis and demand prediction of enterprise data center applications. In *Workload Characterization, 2007. IISWC 2007. IEEE 10th International Symposium on*, pages 171–180, September 2007.

[4] Google app engine. https://developers.google.com/appengine/.