ISSN:2249-5789
Ankita Anoop Rathor et al , International Journal of Computer Science & Communication Networks,Vol 4(6),190-192

Elastic Resource Provisioning for Applications in Cloud Computing

Ankita Anoop Rathor
Dept. Of Computer Engineering, PICT,
Savitribai Phule Pune University, Pune, India
ankitarathor@gmail.com

Prof. Amar Buchade
Dept. Of Computer Engineering, PICT,
Savitribai Phule Pune University, Pune, India
amar.buchade@gmail.com

Abstract

Cloud computing is an emerging technology that is becoming more and more popular. This is because of its elastic nature i.e. users can acquire or release resources on demand and pay only for the resources they use. Those resources are usually in the form of virtual machines (VM's). Elastic provisioning of resources means optimum utilization of resources by avoiding over provisioning and under provisioning of resource. The objective of this paper is to present a study on Bin packing problem for resource provisioning in cloud. We are going to model it as Class constrained bin packing problem(CCBP) where each server is a bin, items are applications. The class constraint is the limit on the number of applications a server can run simultaneously. The goal is to pack the applications onto minimum number of servers and hence increasing the demand satisfaction ratio and saving energy by reducing the number of servers used when the load is less.

.

1. Introduction

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction[8]. The three types of services provided by cloud are IaaS (Infrastructure as a Service),PaaS (Platform as a Service), SaaS (Software as a Service). One of the most important property of cloud computing is elasticity: resources can be scaled up or down as when required. The Amazon EC2 service [1], for example, allows users to buy as many virtual machine (VM) instances as they want and operate them much like physical hardware. Elasticity [9] is defined as the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible. Auto-scaling property will benefit the cloud based applications to acquire or release resources required for their applications automatically. The users are charged only for what they actually use - pay as you go model. To utilize the resource optimally, applications should be capable of dynamically scaling in size to meet demands. For example, If enough resources are provisioned for a Web application by taking into consideration the peak load, in this case if the load remains low then this results into underutilization of resources. On the other hand if the resources are provisioned for average load, then the performance can suffer during peak load. In order to increase the demand satisfaction ratio and reduce the unnecessary costs, resources must be added and removed dynamically as per the requirement. Demand satisfaction ratio is defined as the percentage of demand for an application that is satisfied successfully. The main actors in cloud environment are the provider, the client, and the user. The cloud provider is the owner of the cloud infrastructure, which is rented to the clients. Clients deploy applications in the cloud, which are accessed by users. The provider provides services to the clients, and the clients provide services to their users. Elasticity is essential to the concept of cloud computing being actually used for various purposes [10]. From the perspective of the provider, the elasticity ensures better use of computing resources, providing economies of scale and allowing multiple users to be served simultaneously. From a user perspective, the elasticity has been used mostly to avoid the inadequate provision of resources and consequently the degradation of system performance. Some studies have described the use of elasticity for other purposes, such as, cost reduction [11], and energy savings [12].

2. Literature Survey

The traditional bin packing problem has been studied in the literature. Given n items and n bins, with w_j = weight of item, c= capacity of bin . The problem is to assign items to bins such that bin does not exceed c and no. of bins used is minimum [6].

Some of the most famous approximation algorithms for bin packing problem are[3], Next fit, First fit, Best fit, Worst fit, and Almost any fit algorithms

The two dimensional bin packing problem[13], a set of n rectangular items j E J= {1, . . . , n},each defined by a width, w_j, and a height, h_j and further given an unlimited number of identical rectangular bins of width W and height H, and the objective is to allocate all the items to the minimum number of bins. This bin packing problem cannot be applied in our environment.

The Class Constrained Multiple Knapsack problem (CCMK) aims to maximize the total number of packed items under the restriction that each knapsack has a limited capacity and a bound on the number of different types of items it can hold [14], Unlike CCBP, it does not attempt to minimize the number of knapsacks used. Hence, it does not support green computing when the system load is low.

Application placement in enterprise environments has been studied in [16,17]. They run multiple applications on the same set of servers directly without using VMs or Sand- box. Their approach is suitable when the applications are trustworthy (e.g., enterprise applications). It is not suitable for a cloud environment where applications come from untrusted users .Their decision algorithm has no concern on green computing.

Google AppEngine service provides auto-scaling for web applications. The users are charged by the CPU cycles consumed and not by the no. of application instances. Its internal algorithm used is not disclosed.

There are cloud vendors like Amazon, GoGrid, RackSpace which provides auto-scaling solutions for cloud users [7]. All these solutions allow users to define a set of scaling rules , each rule is composed of one or more conditions and a set of actions to be performed when those conditions are met. Conditions are defined using a set of metrics like cpu usage, memory usage or some threshold.

3. Proposed Work

The proposed architecture is shown in Fig. 1. It consists of a dispatcher, scalar, monitoring system, servers.

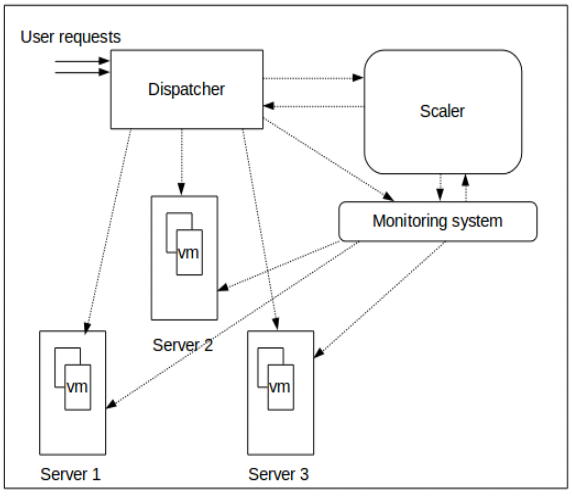

Fig. 1 Overview of System Architecture

3.1. Dispatcher

It is a s/w that determines what pending tasks should be done next and assigns the available resources. It is used to direct incoming requests to an appropriate server based on a set of rules which may include load balancing requests across several servers and content-based routing (i.e. redirecting a request based on the content of that request)[15].

3.2. Scalar

The Scalar does the job of providing on demand resources to the application. It consists of two sub modules:

3.2.1. Load distribution module- The load of an application is mainly the request rate to access that particular application. For each application we need to predict its future resource demand based on request rate and past statistics. Using this information we will decide how to allocate the load among the set of running instances.

3.2.2. Application placement module- It talks about where to place the application. For each application, we need to decide the servers on which its instances run.

3.3. Monitoring system-

It is an important part of such type of system. It gathers different and updated metrics about system and application current state e.g. (memory usage, cpu utilization, response time, pending requests etc.) This information is used to decide the scaling action to be taken e.g. add a VM or remove a VM.

3.4. Servers-

These are the systems on which applications will run. A server consist of Virtual machines on which the application instances run.

The system starts with the user requests, user requests for accessing applications firstly goes to the dispatcher. It contacts with the monitoring system as well as the scaling unit. The application instance is encapsulated inside the VM(virtual m/c). The monitoring system then cheks various application metrics like any instance of the same application is running, request rate, response time, memory required by that application, cpu utilization etc. It also checks the resource metrics like cpu capacity, memory capacity etc. This information is used by the scaling module to distribute the load and to decide where to place the application. Finally dispatcher assigns the resources to the application.

We have framed this system as modified (Class Constrained Bin packing) CCBP problem, where the servers are the bins, items are the applications and the class constraint is on the type of applications. In CCBP problem, we are given a set of bins each having a capacity v and c compartments, and n items of M different classes. We need to fill the bins with items, subject to capacity constraints, such that items of different classes are placed in separate compartments and thus each bin can contain items of at most c distinct classes. The goal is to pack all items in a minimal number of bins. Our goal is to pack applications onto the minimum number of servers thus satisfying more application demand and optimizing the resource usage. When the system is idle, it can be turned off and can save the energy cost.

4. Conclusion

We have discussed the concept of auto-scaling of resources for applications in cloud. Elastic resource provisioning is required to avoid under utilization and over utilization of resources. we modeled this problem as CCBP problem, where the goal is to pack the applications onto minimum number of servers. In future we will discuss the detailed algorithm and its implementation.

5. References

[1] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/.
[2] Google App Engine. http://code.google.com/appengine/.
[3] E.G. Coffman, M.R. Garey, D.S. Johnson, “Approximation algorithms for bin packing: A Survey”
[4] L. Epstein, C. Imreh, and A. Levin, “Class constrained bin packing revisited,” Theor. Comput. Sci., vol. 411, no. 34–36, pp. 3073–3089, 2010.
[5] E. C. Xavier and F. K. Miyazawa, “The class constrained bin packing problem with applications to video-on-demand,” Theor. Comput. Sci., vol. 393, no. 1–3, pp. 240–259, 2008.
[6] Bin packing- www.or.deis.unibo.it/kp/Chapter8.pdf
[7]] E. Caron, L. Rodero-Merino, F. Desprez, and A. Muresan, “Autoscaling, load balancing and monitoring in commercial and opensource clouds,” INRIA, Rapport de recherche RR-7857, Feb. 2012.
[8] National Institute of Standards and Technology, http://csrc.nist.gov/groups/SNS/cloud-computing/
[9] Herbst, Nikolas Roman; Samuel Kounev; Ralf Reussner (2012). "Elasticity in Cloud Computing: What It Is, and What It Is Not". *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013), San Jose, CA, June 24–28*.
[10] D. Owens, “Securing elasticity in the cloud,” ACM Queue, 2010.
[11] Ming Mao, Marty Humphrey, “Auto-scaling to minimize cost and meet application deadline in cloud workflows”, Department of computer science, University of Virginia, Charlottesville, VA 22904.
[12] Zhen Xiao,Qi Chen, Haipeng Luo, “Automatic scaling of Internet applications for cloud computing services”, *IEEE TRANSACTIONS ON COMPUTER*, VOL. 63, NO. 5, MAY 2014.
[13] Andrea Lodi * , Silvano Martello, Michele Monaci, “ Two-dimensional packing problems: A survey”,*European Journal of Operational Research* 141 (2002) 241–252.
[14]] H. Shachnai and T. Tamir, “Tight bounds for online class constrained packing,” Theor. Comput. Sci., vol. 321, no. 1, pp. 103–123, 2004.
[15] http://en.wikipedia.org/wiki/IBM_Websphere_Edge_Components.
[16] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, and A. Tantawi, “Dynamic placement for clustered web applications,” in Proc. Int. World Wide Web Conf. (WWW’06), May 2006, pp. 595–604.
[17] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, “A scalable application placement controller for enterprise data centers,” in Proc. Int. World Wide Web Conf. (WWW’07), May 2007, pp. 331–340.