



International Journal of Advance Engineering and Research Development

Volume 3, Issue 3, March -2016

AUTOMATIC SCALING OF INTERNET APPLICATIONS FOR CLOUD COMPUTING SERVICES

Jyoti Khomane¹, Shilpa Shinde², Varsha Sodnawar³, Prof.S.V.Athawale⁴

^{1,2,3,4}Computer Engineering, AISSMS COE

Abstract — Cloud computing is the latest technology to provide resources from the large data centers. Cloud Computing is made available as a service to the users. Many web applications can get benefit from automatic scaling property of cloud where the no of resources utilization can be scale up and down automatically by cloud service provider. So here present system that provides automatic scaling for internet application in cloud environment. So every application instance encapsulated into the virtual machine and model it as the Class Constraint Bin Packing (CCBP) problem, where each class represents an application and each server is a bin and make use of virtualization technology for fault isolation. Now many business customers require good satisfy response services from cloud. So design and develop semi online color set algorithm that attain good demand satisfaction ratio and also when load becomes low it reducing number of server and save energy. It take help of green computing to adjusting the placement of application instance adaptively and putting ideal machine into the standby mode.

Keywords- Cloud Computing, virtualization, CCBP, green computing, auto scaling etc

I. INTRODUCTION

Cloud computing is a paradigm that based on sharing data and computations over a scalable network of nodes, spanning across end user computers, data centers and web services. A scalable network of such nodes form a cloud. An application based on these clouds is considered as a cloud application. In recent years, most of the software, hardware and networking have extended, specially service-based cloud computing has changed the traditional computer and its centralized storage. This cloud model raise availability and is composed of five essential characteristics ,three delivery models, and four deployment models .

Even though the cloud computing model is sometimes advocated as providing infinite capacity on demand, the capacity of data centers in the real world is finite. The illusion of infinite capacity in the cloud is provided through statistical multiplexing. When a large number of applications experience their peak demand around the same time, the available resources in the cloud can become constrained and some of the demand may not be satisfied. We define the demand satisfaction ratio as the percentage of application demand that is satisfied successfully. The total computing capacity available to an application is limited by the placement of its running instances on the servers. The more instances an application has and the more powerful the underlying servers are, the higher the potential capacity for fulfilling the application demand. On the other hand, when the demand of the applications is low, it is important to preserve energy by reducing the number of servers used. Various studies have found that the cost of electricity is a prime portion of the operation cost of large data centers. At the same time, the average server usage in many Internet data centers is very low.

II. PROBLEM DEFINITION

For web application, there exist multi-dimensional resource demands, such as CPU, memory, network bandwidth, disk I/O bandwidth etc. Among them, we choose CPU and memory as the representative resources to be considered. However, we can handle other types of bottleneck resources as well (e.g., replace memory with disk space for streaming media applications). We cannot handle applications with a larger number of simultaneous bottleneck resources (CPU, memory, disk, network I/O, etc.),

System goals are to maximize the demand satisfaction ratio, minimize the placement change frequency and minimize energy consumption.

III. SYSTEM ARCHITECTURE

In the current work, we will focus on the IaaS clients perspective. A typical scenario could be a user that wants to host a web application, and for this purpose contracts several resources from a known IaaS provider such as Amazon EC2.

From now on we will use the following terminology:

Provider: It refers mainly to the IaaS provider, that offers virtually unlimited resources in the form of VMs. It could also apply to a PaaS provider,

although they sometimes offer limited scaling capacity, that usually cannot be configured by the user.

Client: The client is the user of the IaaS or PaaS service, that uses it for hosting the application. In other words, it is the application owner or provider.

User :It is the final user that accesses the web application. Cloud Computing is essentially powerful computing paradigm to deliver services over the network. The model of Cloud Computing has been distinguished into infrastructure-as-a service (IaaS), Platform-as-a-service (Paas), and software-as-a-service (SaaS)

In Cloud Computing environment, the virtualization technology is the import role to provision the physical resources, such as processors, disk storage, and broadband network. Virtualization refers primarily to platform virtualization, or the abstraction of physical resources for users. In the Cloud, these physical resources is regarded as a pool of resources, these resources thus can be allocated on demand.

Computing at the scale of the Cloud system allows users to access the enormous and elastic resources on-demand. However, a user demand on resources can be various in different time, maintaining sufficient resources to meet peak resource requirements can be costly. On the contrary, if user maintains only minimal computing resources, the resource is not sufficient to handle the peak requirement..We will consider a typical web application with a 3-tier architecture managed by a IaaS user and deployed on a IaaS infrastructure:

Load balancer (LB): It receives all the incoming requests from users and routes them to the application servers.

Business-logic tier (BT): It contains the application servers that execute the application logic.

Storage or persistence tier (ST): It refers to the database system. Each tier can be scaled separately, but we will focus on scaling the business tier(the application servers).

This architecture includes three main components: Front-end load balancer,Virtual cluster monitor system and Auto-provisioning system with an auto-scaling algorithm. Front-end load balancer can balance the requests between different virtual machines that perform the same application in a virtual cluster. Virtual cluster monitor system collects resource usages of all VMs for each virtual cluster that are running on Cloud. Auto-provisioning system can horizontally expand/shrink the number of VMs according to workload of a virtual cluster where VMs belong. If the application running on the virtual cluster consumes most of the resources, auto-scaling can create a new VM that perform the same application and front-end load balancer then balances the requests among VMs in the same virtual cluster.

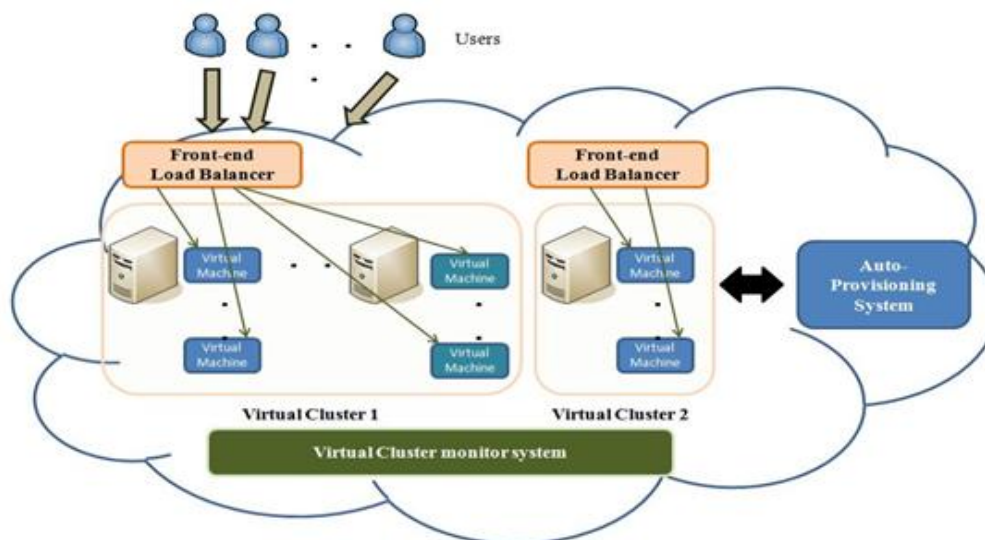


Figure 1. Auto scaling in cloud

In our architecture, Front-end load balancer is utilized to balancing the web application load. Apache HTTP Load-Balancer is utilized as Front-end load balancer to allow incoming HTTP request to be routed into web servers that perform the web application. Since the Apache HTTP Load-Balancer configuration can be updated dynamically, this allows a Cloud system to automatically and dynamically add new web servers for a virtual cluster. The additional web servers enable the virtual cluster to scale and thus provide better response time for incoming HTTP requests.

IV. ALGORITHM

4.1. Class Constrained Bin Packing (CCBP)

A key observation of our work is: not all item movements are equally expensive. Recall that we need to make two decisions in our system: application placement and load distribution. Creating a new application instance on a server is expensive. In the CCBP model, this corresponds to packing the first item of a class into a bin. Adjusting the load distribution, on the other hand, is a much cheaper operation. In the CCBP model, this corresponds to adjusting the number of items of an existing class in a bin. Our algorithm handles the case when all bins are used up. The size of an item represents an amount of load for the corresponding application. By making all items the same unit size, we can represent the item size as a unit of load equal to a specific fraction of the server capacity. This is called the load unit. The capacity v of a bin thus represents how many units of load a server can accommodate. The number of items waiting to be packed from a specific class represents the amount of resource needed by the corresponding application. The resource needs of applications can vary with time. This is modeled as item arrivals and departures: load increases correspond to arrivals of new items, while load decreases correspond to departure of already packed items.

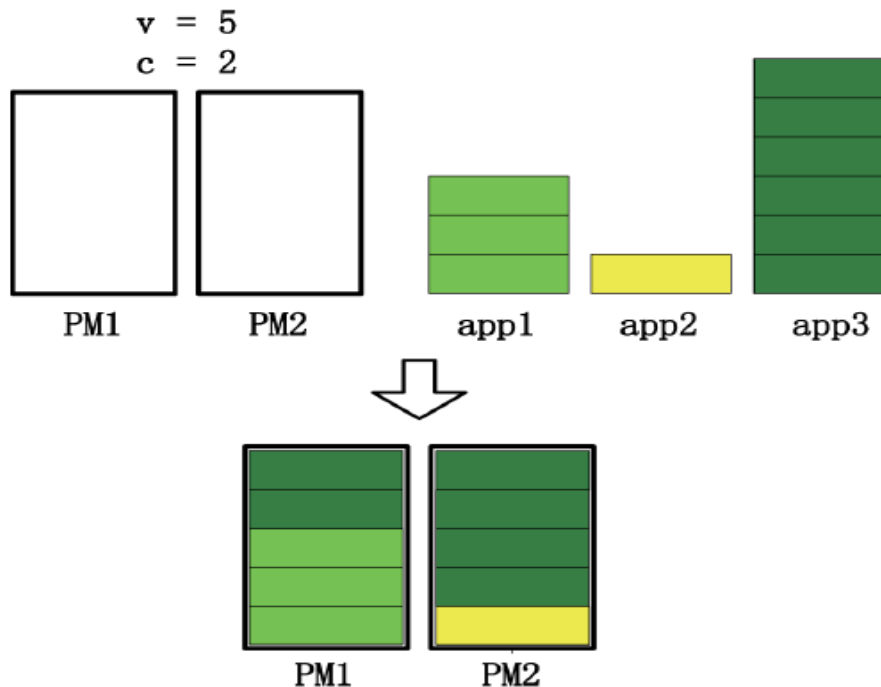


Figure 2. Class constrained bin packing (CCBP)

4.1.1. Application Load Increase

The load increase of an application is modeled as the arrival of items with the corresponding color. A naive algorithm is to always pack the item into the unfilled bin if there is one. If the unfilled bin does not contain that color already, then a new color is added into the bin. This corresponds to the start of a new application instance which is an expensive operation. Instead, our algorithm attempts to make room for the new item in a currently full bin by shifting some of its items into the unfilled bin. Let c_1 be the color of the new item and c_2 be any of the existing colors in the unfilled bin. We

search for a bin which contains items of both colors. Let be such a bin. Then we move an item of color c2 from bin b to the unfilled bin.

This makes room for an item in bin b where we pack the new item. If we cannot find a bin which contains both colors, we see if we can shift the items using a third color as the intermediate. More specifically, we search for two bins:

a) bin b1 contains colors c1 and c3

b) bin b2 contains colors c2 and c3

If we can find such two bins, we proceed as follows:

c) move an item of color c2 from bin b2 to the unfilled bin

d) move an item of color c3 from bin b1 to bin b

e) pack the item in bin b1

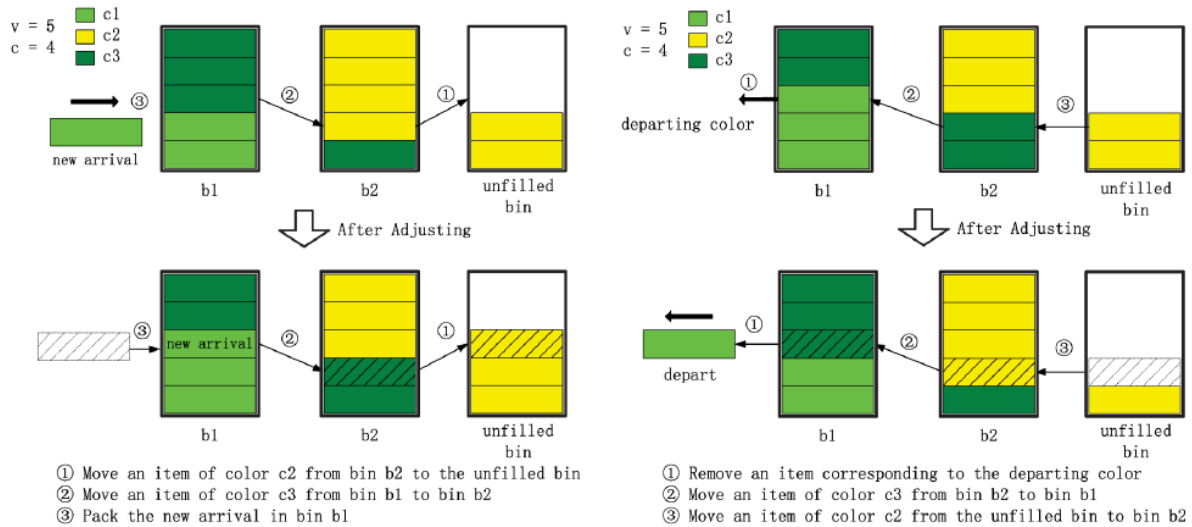


Figure 3. Arrival of a new item (left) and departure of an existing item (right).

4.1.2 Application Load Decrease

The challenge here is to maintain the property that each color set has at most one unfilled bin. Our departure algorithm works as follows. If the color set does not have an unfilled bin, we can remove any item of that color and the resulting bin becomes the unfilled bin. Otherwise, if the unfilled bin contains the departing color, a corresponding item there can be removed directly. In all other cases, we need to remove an item from a currently full bin and then fill the hole with an item moved in from somewhere else. Let c1 be the departing color and be any of the colors in the unfilled bin. We need to find a bin which contains items of both colors. Let be such a bin. We remove the departing item from bin and then move in an item of color from the unfilled bin. More generally, we can find a chain of colors and fill the hole of the departing item by shifting the existing items along the chain. The procedure is similar to the previous case for application load increase. This process for a chain with three colors.

If we cannot find such a chain, we start a new application instance to fill the hole

- remove an item of the departing color from any bin which contains that color.
- select a color c2 in the unfilled bin and add that color into the departing bin.
- move an item of color c2 from the unfilled bin to the departing bin.

If the unfilled bin becomes empty, we can remove it from the color set and shut down the corresponding server since all application instances there receive no load. It might seem counter-intuitive that a decrease in application load can result in the start of a new application instance. However, this is inevitable if we want to consolidate server

V. ADVANTAGES AND CHALLENGES

5.1 Advantages

- It has scale out instances automatically and consistent when demand is increase.
- It covers unnecessary cloud instances automatically and save money when demand is collapse.

- It changes the delicate and impassable instances to maintain higher availability of cloud resources of your application.
- It runs at on demand or spot instance.

5.2 Challenges and Issues

Even though cloud computing is an emerging technology, the research on cloud computing is an early stage. New challenges keep on rising in cloud computing. In this section, we address some of the emerging research challenges in cloud computing that relates to auto scaling.

- The time taken to start the auto scaling is up to 3 minutes.
- The provider cannot differentiate the valid and malicious traffic.
- Badly configured auto scaling will increase the cost of infrastructure and creates unnecessary capacity.
- Auto Scaling mainly focused to reduce the Cost, Energy, High availability and QoS

VI. CONCLUSION

The presenting design and implementation of system which is scale up and scale down number of application instance automatically based on user demand. It develop color set algorithm and also use greedy algorithm to decide the application placement and load distribution. This reaches high satisfaction ratio of an application demand even when the load becomes very high , also saves the energy by reducing number of running instance of virtual machine when the load is low. It use virtualization for fault isolation an maintain good health of cloud application server.

REFERENCES

- [1] Zhen Xiao, Senior Member, IEEE, Qi Chen, and Haipeng Luo, "Automatic Scaling of Internet Applications for Cloud Computing Services", IEEE TRANSACTIONS ON COMPUTERS, VOL. 63, NO. 5, MAY 2014
- [2] Ming Mao and Marty Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud work flows", Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, ISBN: 978-1-4503-0771-0.
- [3] Brian Dougherty, Jules White and Douglas C. Schmidt, "Model driven autoscaling of green cloud computing infrastructure", International Journal of Future Generation Computer Systems, Volume 28 Issue 2, February, 2012, pp371-378.
- [4] Ching-Chi Lin, Jan-Jan Wu, Jeng-An Lin, Li-Chung Song and Pangfeng Liu, "Automatic Resource Scaling Based on Application Service Requirements", Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, ISBN: 978-0-7695-4755-8, pp 941-942.
- [5] Theera Thepparat, Amnart Harnprasarnkit, Douanghatai Thippayawong, Veera Boonjing and Pisit Chanvarasuth, "A Virtualization Approach to AutoScaling Problem", Proceedings of the 2011 Eighth International Conference on Information Technology: New Generations, ISBN: 978-0-7695-4367-3, pp169-173.
- [6] Bruno Ciciani, Diego Didona, Pierangelo Di Sanzo, Roberto Palmieri, Sebastiano Peluso, Francesco Quaglia and Paolo Romano, "Automated Workload Characterization in Cloud-based Transactional Data Grids", Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum, ISBN: 978-0-7695-4676-6, pp 1525-1533.
- [7] Srikumar Venugopal, Han Li and Pradeep Ray, "Auto-scaling Emergency Call Centers use Cloud Resources to Handle Disasters", IEEE Computer Society, 2011, ISBN: 978-1-4577-0103-0 Anshul Gandhi, Mor Harchol-Balter, Ram ghunathan and Michael A. Kozuch, Distributed, "Robust Auto-Scaling Policies for Power Management in Compute Intensive Server Farms", IEEE Computer Society, 2011, ISBN: 978-0-7695-4650-6.
- [8] Nilabja Roy, Abhishek Dubey and Aniruddha Gokhale, "E_cient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting", Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing, ISBN: 978-0-7695-4460-1, pp 500-507.