RESEARCH ARTICLE OPEN ACCESS

Automatic Scaling Of Internet Application For Cloud Computing

C.Muruganandam [1], M.Gayathiri [2]
Assistant Professor in Computer Science [1], Research Scholar [2]
Department of Computer Science,
Rajah Serfoji Government college, Thanjavur.
Tamil Nadu - India

ABSTRACT
An automatic scaling property utilized by many Internet applications in the cloud service provider and get benefit from where their resource usage can be scaled up and down automatically. The present paper proposes a system that provides automatic scaling for Internet application in the cloud environment. The present method encapsulate each application instance inside a Virtual Machine (VM) and use virtualization technology to provide fault isolation and this method is called Class Constraint Bin Packing (CCBP) problem where each server is a bin and each class represents an application. The class constraint reflects the practical limit on the number of applications a server can run simultaneously. The present paper develops an efficient semi-online color set algorithm that achieves good demand satisfaction ratio and saves energy by reducing the number of servers used when the load is low. Experiment results demonstrate that the proposed system can improve the throughput by 180% over an open source implementation of Amazon EC2 and restore the normal QoS five times as fast during flash crowds.

Keywords:-VM, EC

I. INTRODUCTION

One of the often cited benefits of cloud computing service is the resource elasticity: a business. However, the users still need to decide how much resources are necessary and for how long. We believe many Internet applications can benefit from an auto scaling property where their resource usage can be scaled up and down automatically by the cloud service provider. Customer can scale up and down its resource usage as needed without upfront capital investment or long term commitment.

The Amazon EC2 service, for example, allows users to buy as many Virtual Machine (VM) instances as they want and operate them much like physical hardware

A user only needs to upload the application onto a single server in the cloud, and the cloud service will replicate the application onto more or fewer servers as its demand comes and goes. The users are charged only for what they actually use- the so called **"pay –as- you- go"** facility model.

1.1 AUTO SCALING

Auto scaling technique provides on-demand resources convenience supported bound workloads in cloud computing systems. The auto scaling service permits the configuration of capability management policies applied to dynamically decide on deed or emotional resource instances for a given application.

The auto scaling algorithm program includes Run-time scaling and performance-oriented planning algorithm program. Varied auto scaling methods exploitation log traces of Google information center clusters embrace Auto-scaling Demand Index (ADI) metric for auto-scaling strategy.

The ability of a system to dynamically adapt its underlying computing infrastructure resources in response to variable workload changes over time.

1.2 LOAD BALANCING

Load balancing may be a technique to distribute the load across the nodes. The choice to balance load is formed domestically by a node, supported its current utilization. Every node ceaselessly measures its resource utilization of computer hardware, memory, network consumption and space.

Load balancing allows enterprises to manage application or workload demands by allocating resources among multiple computers, networks or servers.

Cloud load balancing involves hosting the distribution of workload traffic and demands that reside over the internet.

1.3 SCALING METHODS

Depending on the particular Cloud environment, elastic scaling can be performed vertically, horizontally, or in a hybrid. Each of them has their advantages and limitations. In this section, we discuss the key factors that need to be considered when making the provisioning plan.

(i) Vertical Scaling

VM Resizing Vertical scaling means removing or adding resources, including CPU, memory, I/O, and network, to or from existing VMs. To dynamically perform these operations, modern hypervisors utilize mechanisms such as CPU sharing and memory ballooning, to support CPU and memory hot-plug.

(ii) Horizontal Scaling

Horizontal scaling is the core of the elasticity feature of Cloud. Most Cloud providers offer standardized VMs of various sizes for customers to choose. Others allow users to customize their VMs with a specific amount of cores, memory, and network bandwidth. Besides, multiple pricing models are co-existing in the current Cloud market, which further increases the complexity of the provisioning problem.

II. APPLICATION SCALING AND CLOUD INFRATRUCTURE

2.1 Application Service Behaviour Prediction

It is critical that the system is able to predict the demand and behaviours of the hosted services, so that it intelligently undertake decisions related to dynamic scaling or de-scaling of services over federated cloud infrastructures. Concrete prediction or forecasting models must be built before the behaviour of a service, in terms of computing, storage, and network bandwidth requirements, can be predicted accurately.

2.2 Scalable Monitoring of System Components

The components that contribute to a federated system may be distributed, existing techniques usually employ centralized approaches to overall system monitoring and management.

Monitoring of system components is required for effecting on-line control through a collection of system performance characteristics. We architecting service monitoring and management services based on decentralized messaging and indexing models [27].

2.3 Application Load Increase

The load increase of an application is modelled 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.

Let c_1 and c_2 be the color of the new item. Let b be such a bin. This makes room for an item in bin b where we pack the new item. More specifically, we search for two bins:

- bin b_1 contains colors c_1 and c_3
- bin b_2 contains colors c_2 and c_3

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

- move an item of color c_2 from bin b_2 to the unfilled bin.
- Move an item of color c_3 from bin b_1 to bin b_2.
- Pack the item in bin b_1.

2.4 Application Load Decrease

The load decrease of an application is modelled as the departure of previously packed items. Note that the departure event here is associated with a specific color, not with a specific item. The algorithm has the freedom to choose which item of that color to remove.

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.

2.5 Virtualization

In cloud providing services of web application the virtualization plays important role for fault isolation. It is one of the key enabling technology for cloud computing the main goal of virtualization is to improve the utilization of instance, enable failure, and easy to dispensation.

The storage virtualization is allows transparent provisioning storage capacity and simplifies data flexibility and management. The server virtualization is using Virtual Machine Monitor (VVM) layer running between operating system and hardware.

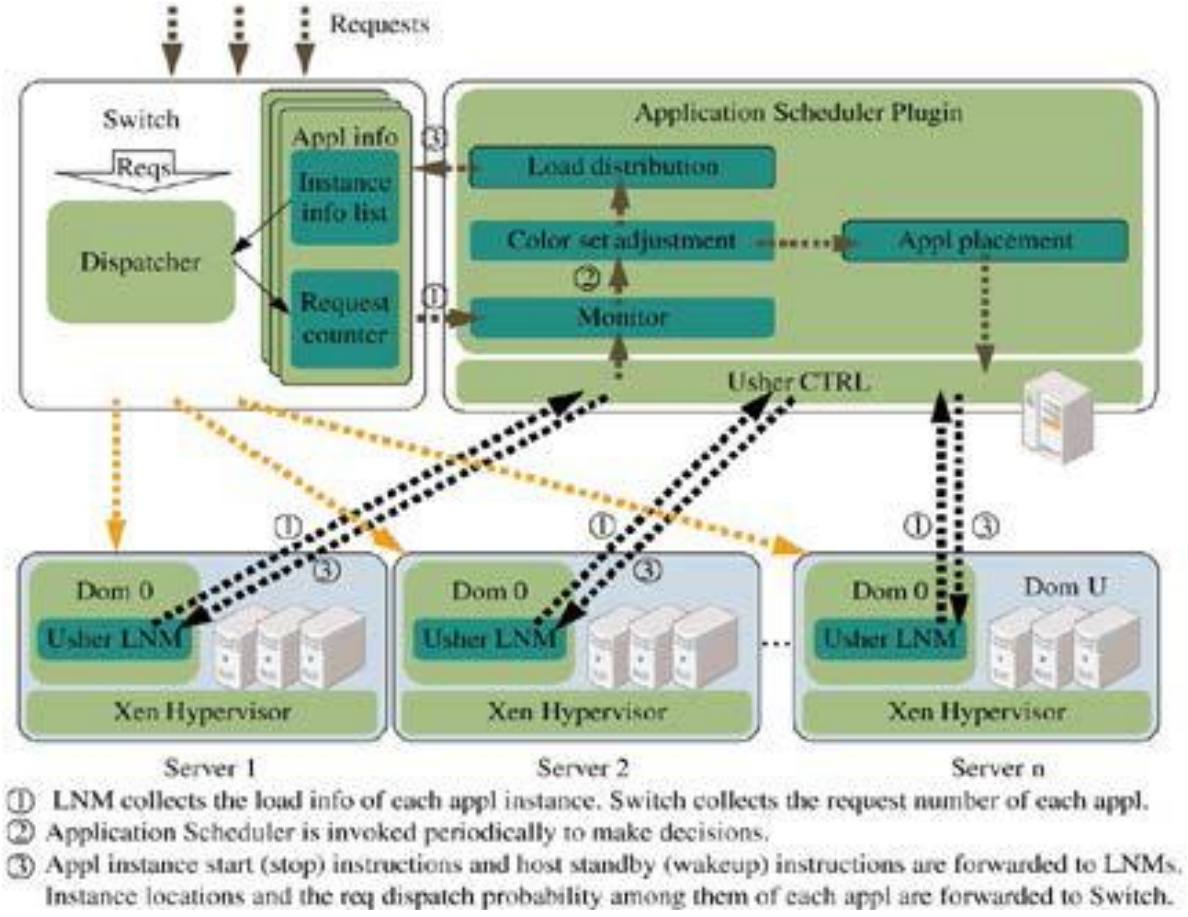
III. METHODOLOGY

3.1 System Architecture

The architecture of our system. We encapsulate each application instance inside a

Virtual Machine (VM). The use of VMs is necessary to provide isolation among untrusted users. Both Amazon EC2 and Microsoft azure use VMs in their cloud computing offering. Each server in the system runs the Xen hypervisor which supports a privileged domain 0 and one or more domain.

The more instances an application has and the more powerful the underlying servers are, the higher the potential capacity for satisfying the application demand.



3.1 System Architecture

3.1.1 Scheduled Procedure in Architecture

The scheduled procedure of our system can be described as follows:

(i) The LNM at each node L7 switch collect the application placement, the resource usage of each instance, and the total request number of each application periodically.

(ii) The Application Scheduler is invoked periodically to make the following decisions:

- **Application Placement**: For each application decide the set of servers its instance run on.
- **Load Distribution**: For each application, predict its future resource demands based on the request rate and past statistics, and then decide how to allocate its load among the set of running instances. The load of an Internet application is largely driven by the rate of user requests.

(iii) The decisions are forwarded to the LNM and the L7switch for execution. The list of action items for each node includes:

- Standby or wake up instructions.
- Application starts and stops.
- The allocation of local resource among the applications.

(iv) After that the scheduler notifies the L7 switch of the new configuration including:
- The list of applications.
- For each application, the location of its running instances and the probability of request distribution among them.

(v) The L7 switch starts processing web requests according to the new configuration. It may seem from the discussion above that the User CTRL is a central point of failure.

(vi) The complicated applications can take along time (several minutes or much longer) to start and finish all the initializations.

(vii) The present method takes the advantage of this feature to bypass the application start process by suspending a fully started and initialized application instance to the disk.

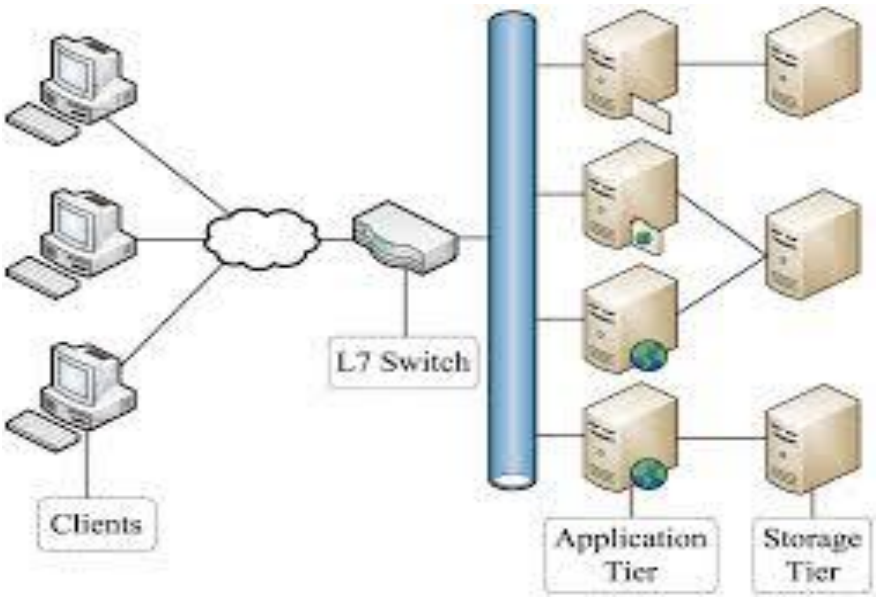
3.1.2 Two-tiered Architecture for Internet Applications

The typical architecture of data center servers for internet applications. It consists of a load balancing switch, a set of application servers, and a set of backend storage servers.

The front end switch is typically a Layer 7 switch [2] which parses application level information in web requests and forwards them to the servers with the corresponding applications running.

Each application can run on multiple server machines and the set of their running instances are often managed by some clustering software such as web Logic [3]. Each server machine can host multiple applications.

The applications store their state information in the backend storage servers. It is important that the applications themselves are stateless so that they can be replicated safely. The storage servers may also become overloaded, but the focus of this work is on the application tier.



3.1.2 Internet Applications for Two-Tier Architecture

3.2 Class constrained Bin Packing Problems

We consider class-constrained bin-packing problems (CCBP), in which we are given a set of bins, each having a capacity v and c compartments, and n items of M different classes and the same (unit) size.

In the class-constrained bin-packing problem multiple knapsack problem (CCMK), we wish to maximize the total number of items packed in m bins, for m>1. The CCBP and CCMK problems model fundamental resource allocation problem in computer and manufacturing systems.

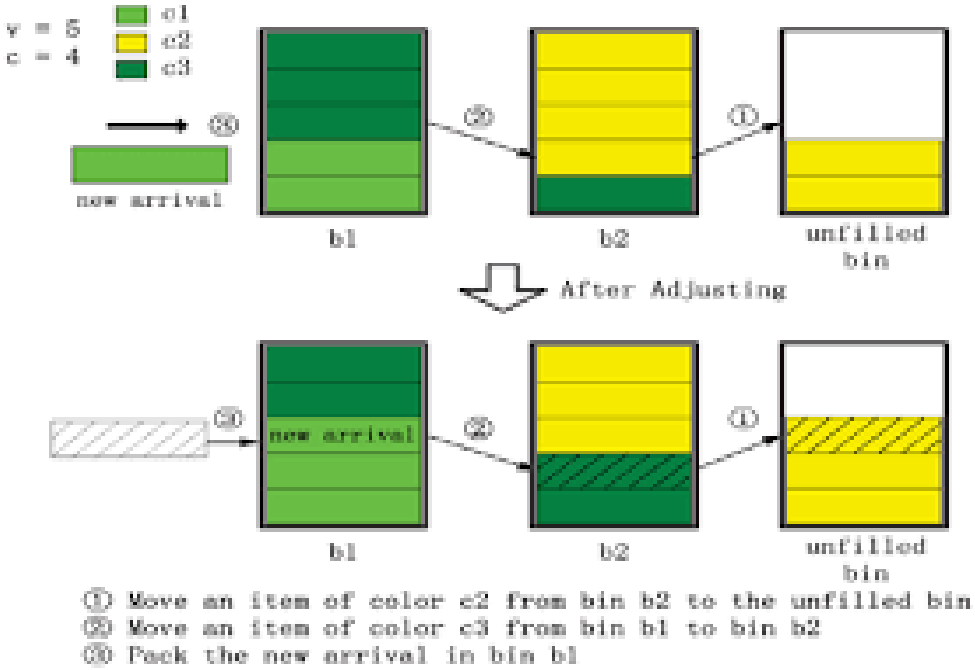
3.3 Modified CCBP Problem

The modified class-constrained Bin packing problem has the arrival and departure of new item.

3.3.1 Modified CCBP Arrival of New Item

Now in modified CCBP mainly focus on the two key point's one is application placement and other one is load distribution. Based on observation develop a semi-online algorithm for CCBP which packs the current item without any knowledge of any sub sequential item in list of input sequence.

Here in this scenario color set algorithm is used for the label each class of item with color and arrange them in to color set as per they arrive input sequence.



3.3.1 Arrival of New Item.

Item from different color set are packed independently. For packing each item in color set use greedy algorithm it is mathematical process which solve multistep problem by decide which next step will provide most obvious benefit.

Here each color set has one unfilled bin so when new item from a specific color set arrives it packed into corresponding unfilled bin. And each item are packed into current bin until the capacity is reached.

Actually here the algorithm attempts to make space for new item in currently full bin by shifting some of item into unfilled bin.

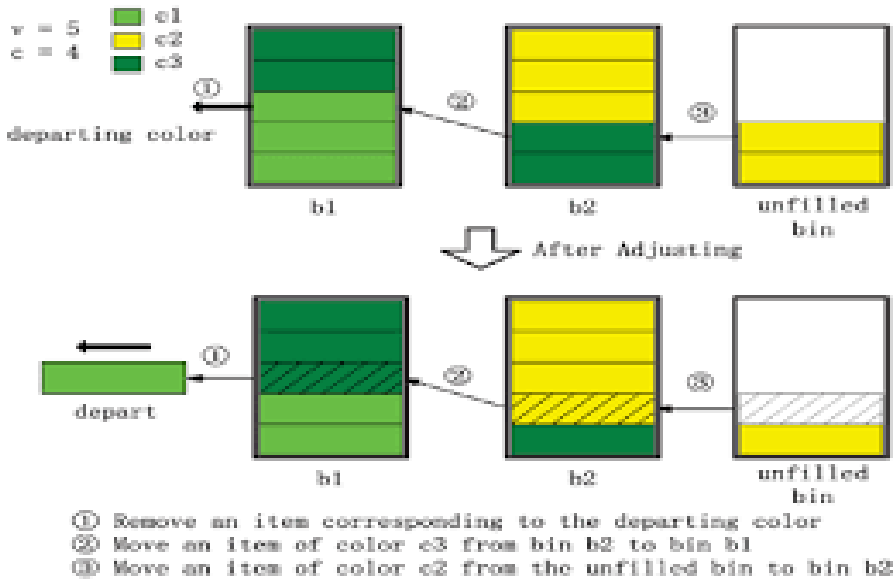
If the application load increase, as the arrival of new item with the corresponding color, if the unfilled bin does not exists that color already the new color is to add in to bin.

3.3.2 Modified CCBP Departure of Item

The departure event is associated with number of specific color, not with specific item. Main advantage is the algorithm has freedom to select which item of particular color is to remove.

And challenge is to maintain property that every color set has one unfilled bin departure working as follows:

- If in the color set does not present unfilled bin then remove any item of that color and the output bin becomes unfilled bin.
- If the unfilled bin contains the departing color, so corresponding item removed directly.



3.3.2 Departure of Existing Item.

And finally last item of particular color leaves that selected color can be removed from its color set this is nothing but the closing down the last instance of an application when load reduce to zero.

And color set become unfilled the challenge is here is to maintain property that there is at most one unfilled color set in the system.

3.4 Existing system

Even though the cloud computing model is sometimes advocated as providing infinite capacity on demand, the capacity of data center in the real world is finite. The illustrations of infinite capacity in the cloud is provided through stastical 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 amount of computing capacity available to an application is limited by the placement of its running instances on the servers.

3.5 Proposed system

In this paper, we present a system that provides automatic scaling for internet applications in the cloud environment. Our contributions include the following:

- We summarize the automatic scaling problem in the cloud environment, and model it as a modified Class Constrained Bin Packing (CCBP) problem where each server is a bin and each class represents an application.
- We creatively support item departure which can effectively avoid the frequent placement changes caused by repacking.
- We support green computing by adjusting the placement of application instances adaptively and putting idle machines into the standby mode.
- Experiments and simulations show that our algorithm is highly efficient and scalable which can achieve high demand satisfaction ratio, low placement change frequency, short request response time, and good energy saving.

IV. RESULT&DISCUSSION

The following table-1 shows the comparative performance of the above discussed protocols.

Cloud Platforms	Load Balancing	Provisioning	Auto Scaling
Amazon Elastic	Good	Good	Good

Compute cloud			
Eucalyptus	Good	Moderate	Bad
Google App Engine	Good	Good(Fixed templates so far)	Manual
Manjarasoft Aneka	Good	Good	Good
GoGrid Cloud Hosting	Good	Moderate	Programming way only

4.1 Scaling-Up and Scaling-down

Metric	Policy of Scaling Up	Policy of Scaling Down
CPU Utilization	CPU Utilization >= 80 % for 5 minutes	CPU Utilization <= 20 % for 5 minutes
Network Out	Network Out >= 2 GB for 5 minutes	Network Out <= 5 MB for 5 minutes
Latency	Latency >= 0.3 second for 5 minutes	Latency <= 0.002 second for 5 minutes
Request Count	Request Counts >= 150 requests for 5 minutes	Request Counts <= 2 requests for 5 minutes

4.1 Comparsison between Scaling-Up and Scaling-Down.

V. CONCLUSION

In Automatic Scaling presented the design and implementation of a system that can scale up and down the number of application instances automatically based on demand. Our system achieves high satisfaction ratio of application demand even when the load is high. It saves energy by reducing the number of running instances when the load is low. There are several directions for future work. In the future, we plan to extend our system to support differentiated services but also consider fairness when allocating the resources across the applications. The CCBP algorithm is gives better performance compare with other algorithm.

REFERENCES

[1] ”Amazon elastic compute cloud(Amazon EC2), http:// aws.amazon.com/ec2/.”

[2] H. Shachnai, T. Tamir, On two class-constraine diversions of the multiple knapsack problem, Algorithmica 29 (2001) 442–467.

[3] J.L. Wolf, P.S. Yu, H. Shachnai, Disk load balancing for video-on-demand systems, ACM Multimedia Systems J. 5 (1997) 358–370.

[4] 4. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications”, In Proc. ACM SIGCOMM (2001), San Diego, CA, August 2001.

[5] IBM WebSphere Software, http://www.ibm.com/software/websphere, November 2007.

[6] HAMILTON, J. Cost of Power in Large-Scale Data Centers [online]. November 2008 Availablefrom:http://perspectives.mvdirona.c

om/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx.

[7] J.Zhu, Z.Jizng, Z.xiao, and X.Li, "Optimizing the performance of virtual machine synchronization for fault tolerance," IEEE Transactions on Computers, Dec.2011.

[8] C.Yeo and R.Buyya. Integrated Risk Analysis for a commercial Computing service. Proc. Of the 21st IEEE International parallel and Distributed Processing Symposium, Long beach, California, USA, March 2007.

[9] Google App Engine, http://appengine.google.com [17 march 2010].

[10] D.Chappell. Introducing the Azure Services Platform. White paper, http://www.microsoft.com/azure [Jan 2009].

[11] Oracle WebLogic Suite. http://www.oracle.com/us/products/middleware/cloud-app-foundation/weblogic/overview/index.html. Accessed on May 10, 2012.

[12] M.R. Garey and D.S. Johnson, "A 71/60 theorem for bin packing," J.Complexity, vol.1, pp. 65-1606, 1985.

[13] Scalr: The Auto Scaling Open source Amazon EC2 Effort. http://www.scalr.net/.Accessed on May 10, 2012.

[14] 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.

[15] C.Chekuri and S.Khanna, "On multidimensional packing problems," SIAM J.Comput., vol.33, no.1, pp.8367-851, 2004.