Application Placement and Load Balancing of Internet Application in Cloud Computing Services

S. R. Mete
MBES College of Engg.
Ambajogai
Maharashtra India.

B. M. Patil
MBES College of Engg.
Ambajogai
Maharashtra India.

ABSTRACT
Automatic scaling is useful in the internet applications in which the cloud service provider provide the services to resource usage and resources scaled up and down automatically. Automatic scaling provides the scaling of internet application in cloud environment. System calculates the fault isolation using virtualization technology. In this system, it uses the Class Constraint Bin Packing (CCBP) algorithm and semi online color set algorithm which saves the energy by reducing server and gives the good satisfaction ratio.

Keywords
Cloud computing, CCBP, auto scaling, fault isolation

1. INTRODUCTION
Cloud computing also On-demand computing is a kind of internet based computing that provides shared processing resources and data to computer and other processing devices on demand. It is a model for On-demand access to a shared pool of configurable computing resources. Cloud computing and storage solution provides various capabilities to store and process their data in third party data centers. Applications are running faster in cloud computing. Architecture involves multiple cloud components communicating with each other.

For security and privacy purpose the user can encrypt the data that is proposed or stored within the cloud to prevent unauthorized access. Some small businesses that don’t have expertise in IT security cloud find that it’s more secure for them to use a public cloud.

One of the key characteristics of cloud computing is the flexibility that it offers and one of the ways that flexibility is offered is through scalability. This refers to the ability of a system to adapt and scale to changes in workload. Cloud technology allows for the automatic provision and de-provision of resource as and when it is necessary, thus ensuring that the level of resource available is as closely matched to current demand as possible.

There is also great choice in the level of security and management required in cloud deployments.

A public cloud is a cloud in which services and infrastructure are hosted off-site by a cloud provider. Also shared across their client base and accessed by these clients via public networks such as the internet. Public clouds offer great economies of scale and redundancy but are more dangerous than private cloud setups due their high levels of accessibility.

Private clouds use pooled services and infrastructure stored and maintained on a private network whether physical or virtual accessible for only one client. The transparent benefits to this are greater levels of security and control. Cost benefits must be sacrificed to some extent though, as the enterprise in question will have to purchase or rent and maintain all the necessary software and hardware.

The final cloud option is a hybrid cloud which combines both public and private cloud elements. A hybrid cloud allows a company to maximize their efficiencies by utilizing the public cloud for non-sensitive operations while using a private setup for sensitive or mission critical operations. Companies can ensure that their computing setup is ideal without paying any more than is necessary.

There are 3 models of cloud computing witch describe the service on offer; Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

1.1 Infrastructure as a Service (IaaS)
In the other cloud computing categories, IaaS refers to the delivery of virtualized computing resource where a service across a network connection. IaaS specifically deals with hardware or computing infrastructure as a service. It provides virtualized server space, storage space, network connections and IP addresses. The resource is pulled from a pool of servers distributed across data centers under the provider’s control. The user is then granted access to this resource for build their own IT platforms. IaaS can provide enterprises with great business benefits.

1.2 Platform as a Service (PaaS)
PaaS is an extension of IaaS which describes a category of cloud computing that provides developers with environments to build applications over the internet. In addition to the fundamental computing resource supplied by the hardware in an IaaS offering, PaaS models include the software and configuration (often known as the solution stack) which required to create the platform where clients can create their applications. PaaS packages can be oriented to meet individual user needs and also they can pick the features of the service that are relevant to them while dismissing those are not. PaaS provides a number of benefits to enterprises including simplifying the development process for geographically split development teams.

1.3 Software as a Service (SaaS)
SaaS describes a software delivery model in which applications are hosted and made available to customers over a network connection. Many people make use of SaaS without realizing it as many web applications are delivered in this way. Gmail, Flickr, Twitter and Facebook are all popular examples of SaaS. Enterprise users also frequently make use of SaaS with many popular accounting, invoicing, sales, communications and CRM systems being delivered this way. Advantages of SaaS are reduced time to benefit, lower costs,

scalability and integration, new releases, easy to use and perform proof of concepts. The popularity of SaaS is steadily increasing because it simplifies deployment and reduces customer acquisition costs.

2. RELATED WORK

On Multi-dimensional Packing Problems; system studies the approximateness of multi-dimensional generalizations of three classical packing problems: multiprocessor scheduling, bin packing, and the knapsack problem [4]. Specifically, system studies the vector scheduling problem i.e. dual problem of the vector bin packing problem, and a class of packing integer programs. The vector scheduling problem is to schedule n d-dimensional tasks on m machines such that the maximum load over all dimensions and all machines is minimized. The vector bin packing problem seeks to minimize the number of bins needed to schedule all n tasks such that the maximum load on any dimension across all bins is bounded by a fixed quantity, say 1. Such problems naturally arise when scheduling tasks that have multiple resource requirements. Finally, packing integer programs capture a core problem that directly relates to both vector scheduling and vector bin packing which means the problem of packing a maximum number of vectors in a single bin of unit height.

Multi-agent learning approach to online distributed resource allocation in computing clusters is traditionally centralized which limits the cluster scale [5]. Effective resource allocation in a network of computing clusters may enable building larger computing infrastructures. The system needs to suggest a MAL algorithm and apply it for optimizing online resource allocation in cluster networks. The learning is distributed to each cluster, using local information only and without access to the global system. Experimental results of our multi-agent learning approach perform reasonably well compared to an optimal solution and better than a centralized shortsighted allocation approach in some cases.

The most interesting feature that Cloud Computing brings, from a Cloud client's point of view, is the on-demand resource provisioning model [6]. This allows Cloud client platforms to be scaled up in order to accommodate more incoming clients and to scale down when the platform has unused resources which can all be done while the platform is running. Therefore the physical resources are used more efficiently and the cloud client saves expenses. It is done by leveraging more direct functionalities that clouds provide among which three of these key functionalities are automatic scaling, load balancing and monitoring. They represent the focus of the current work in detail each of the three topics and present details on their presence and implementation in the commercial, open-source and research worlds. Among the commercial cloud platform providers, the system has focused on Amazon EC2, Microsoft Azure, GoGrid and RackSpace. From the open-source initiatives, system has focused our attention on Nimbus, Eucalyptus and OpenNebula. The system has also detailed attempts in the research world that are relevant to the topic of interest and are not connected to any of the cloud providers. By using the three cloud features of a cloud client, will understand their inner workings and will be aware of the state-of-the-art available in current commercial and open-source cloud platforms.

3. CLASS CONSTRAINT BIN PACKING

Application placement problem is essential to achieving a high demand satisfaction ratio without wasting energy. Class Constrained Bin Packing (CCBP) problem where each server is a bin and each class represents an application which develop an innovative auto scaling algorithm to solve the problem and present a rigorous analysis on the quality of it with provable bounds. It effectively avoids the frequent placement changes caused by repacking which use a fast restart technique based on virtual machine (VM) suspends and resumes that reduces the application start up time dramatically for internet services. In this project efficient and dynamic scaling of internet application over cloud computing while transferring application from arrival to departure CCPB problem occurs. To rectify CCBP problem apply the semi-online algorithm. In early days while transferring application from arrival to departure, the client can send application to only one server. So system uses virtual-machine technique to overcome this problem.

In the traditional bin packing problem, a series of items of different sizes need to be packed into a minimum number of bins. The class constrained version of this problem divides items into classes or colors. Each bin has capacity c and can accommodate items from at most d distinct classes. It is class constrained because the class diversity of items packed into the same bin is constrained. The goal is to pack the items into a minimum number of bins. The system can model our resource allocation as the Class Constrained Bin Packing (CCBP) problem where each server is a bin and each class represents an application. Items from a specific class represent the resource demands of the corresponding application. The class constraint reflects the practical limit on the number of applications a server can run simultaneously. The capacity of a bin represents the amount of resources available at a server for all its applications where the servers are homogeneous with uniform capacity. The number of servers available is finite. This 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; system can represent the item size as a unit of load equal to a specific fraction of the server capacity which is called the load unit. The capacity c 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 which is modeled as item arrivals and departures. That means load increases correspond to arrivals of new items, while load decreases correspond to departure of already packed items. Most online CCBP algorithms do not support item departure. In online class constrained packing has a general discussion on item departure, but does not describe any specific algorithm [3]. Instead they focus on proving the bounds on the approximation ratio for any removal algorithm. In result it does not apply here because they do not repack items after the departure. A main participation of our algorithm is the support for item departure which is essential to maintaining good performance in a cloud computing environment where the resource demands of Internet applications can vary dynamically.

Our algorithm is a part of the family from color set algorithms [3]. The system label each class of items with a color and organize them into color sets as they arrive in the input sequence. The number of distinct colors in a color set is at most dc which ensures that items in a color set can always be packed into the same bin without assaulting the class constraint. All color sets contain exactly dc colors except the last one which may contain less color. Items from different color sets are packed independently. A greedy algorithm is used to pack items within each color set in which the items are packed into the current bin until the capacity is reached. Then the next bin is opened for packing. Thus each color set has at most one unfilled bin. Note that a full bin may contain less than dc colors. When a new item from a specific color set arrives, it is packed into the corresponding unfilled bin. If all bins of that color set are full, then a new bin is opened to adapt the item.

3.1Application 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 then a new color is added into the bin which represents to the start of a new application instance which is an expensive operation. Instead, our algorithm attempts to make space for the new item in a currently full bin by shifting some of its items into the unfilled bin. The length of the chain is bounded by the number of colors in the color set.

The system can adapt the new item by shifting the existing items along the chain. The item movements are suppositional and used only to calculate the new load distribution. No physical movement of any application occurs. Also the chain length is bounded by a constant and does not increase with the numbers of applications or servers in the system. If system cannot find such a chain, the new color has to be added into the unfilled bin which requires starting a new application instance. If the color set has no unfilled bin, then a new bin is allocated. If all bins are used up, then the load increase cannot be satisfied.

3.2 Application Load Decrease

If the load decrease of an application is modeled as the departure of previously packed items in which the departure event here is associated with a specific color not with a specific item. The algorithm has the exemption to choose which item of that color to remove. In this system can maintain the property when each color set has at most one unfilled bin. If the color set does not have an unfilled bin, system 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, it needs to remove an item from a currently full bin and then fill the hole with an item moved in from somewhere else.

If the unfilled bin becomes empty, they 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-non rational that a decrease in application load can result in the start of a new application instance. Using this it can save energy.

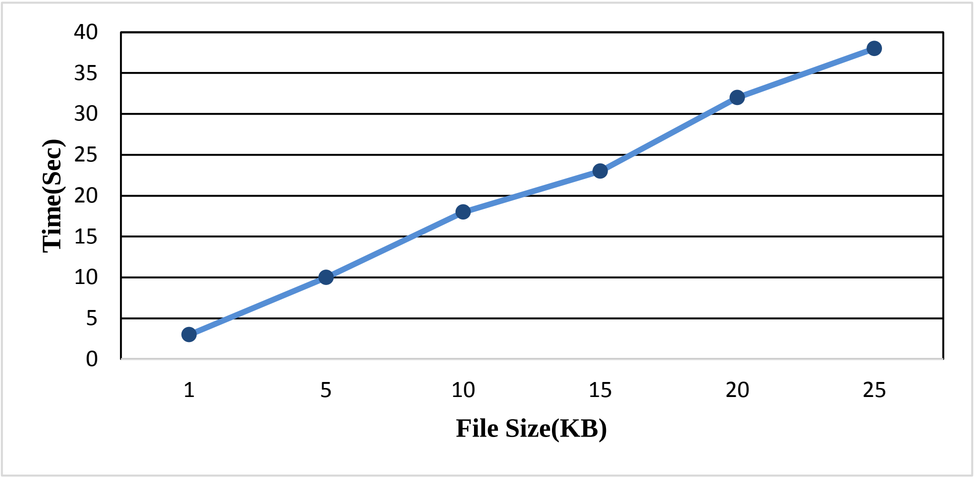

Fig 1 - Uploading File

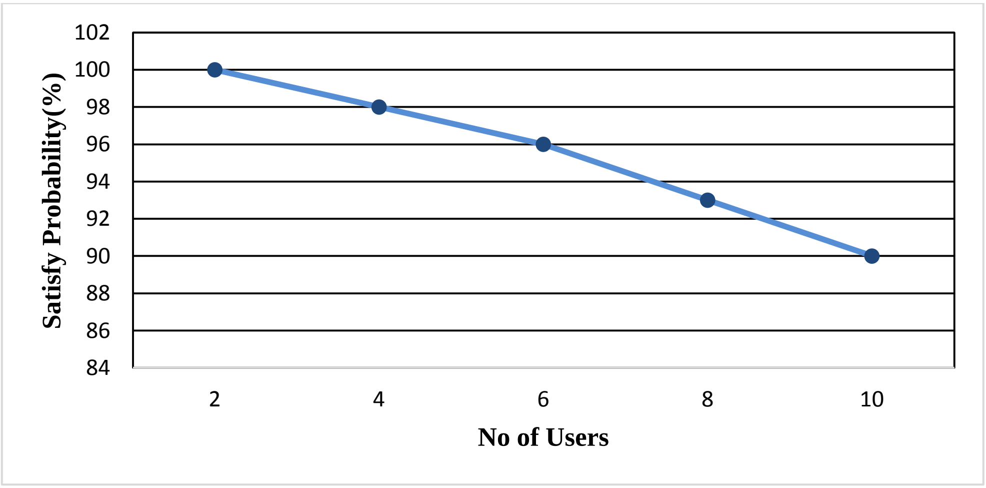

Fig 2 - Performance of the algorithm as the demand increases

4. EVALUATION

In this section, simulation results for our method proposed. For this result, Java language is used.

4.1 Uploading file

The simulation evaluates the uploading file scheme in which we studied the comparison of required time for specific uploading file size. Fig 1 shows the required time as the specific uploading file size changes. From this result, the required time for uploading increases as file size increases.

Also the required time for uploading decreases as file size decreases.

4.2 Performance of the algorithm as the demand increases

The simulation measures performance of the algorithm as the demand increases satisfaction ratio decreases as shown in Fig 2. As no. of users increases satisfy probability decreases. Also if no. of users decreases then satisfy probability increases.

5. CONCLUSION AND FUTURE SCOPE

They developed a color set algorithm to decide the application placement and the load distribution. High satisfaction ratio of application demand even when the load is very high. It saves energy by reducing the number of running instances when the load is low. Resources become tight; they may want to give their premium customers a higher demand satisfaction ratio than other customers. To extend our system to support differentiated services but also consider fairness when allocating the resources across the applications. CCBP works well when the aggregate load of applications in a color set is high. Future work is to develop an efficient algorithm to distribute incoming requests among the set of equivalence classes and to balance the load across those server clusters adaptively.

6. REFERENCES

[1] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/. Accessed on May 10, 2012.

[2] A. Cohen, S. Rangarajan, and H. Slye, “On the performance of tcp splicing for url-aware redirection,” in Proc. 2nd Conf. USENIX Symp. Internet Technol. Syst., 1999, p. 11.

[3] H. Shachnai and T. Tamir, “Tight bounds for online classconstrained packing,” Theor. Comput. Sci., vol. 321, no. 1, pp. 103–123, 2004.

[4] C. Chekuri and S. Khanna, “On multidimensional packing problems,” SIAM J. Comput., vol. 33, no. 1, pp. 837–851, 2004.

[5] C. Zhang, V. Lesser, and P. Shenoy, “A multi-agent learning approach to online distributed resource allocation,” in Proc. Int. Joint Conf. Artif. Intell. (IJCAI’09), 2009, pp. 361–366.

[6] 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.

[7] Google App Engine. http://code.google.com/appengine/. Accessed on May 10, 2012.

[8] M. Armbrust et al., “Above the clouds: A Berkeley view of cloud computing,” EECS Depart., Univ. California, Berkeley, CA, Tech. Rep. UCB/EECS-2009–28, Feb. 2009.

[9] L. Siegele, “Let it rise: A special report on corporate IT,” in The Economist, London, U.K.: London Economist Newspaper, Oct. 2008, vol. 389, pp. 3–16.

[10] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, “Managing energy and server resources in hosting centers,” in Proc. ACM Symp. Oper. Syst. Princ. (SOSP’01), Oct. 2001, pp. 103–116.

[11] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the art of virtualization,” in Proc. ACM Symp. Oper. Syst. Princ. (SOSP’03), Oct. 2003, pp. 164–177.

[12] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker, “Usher: An extensible framework for managing clusters of virtual machines,” in Proc. Large Install. Syst. Admin. Conf. (LISA’07), Nov. 2007, pp. 1–15.

[13] J. Zhu, Z. Jiang, Z. Xiao, and X. Li, “Optimizing the performance of virtual machine synchronization for fault tolerance,” IEEE Trans. Comput., vol. 60, no. 12, pp. 1718–1729, Dec. 2011.

[14] RUBiS: Rice University Bidding System. http://rubis.ow2.org/. Accessed on May 10, 2012.

[15] Linux Documentation. Available online at: http://www.kernel.org/doc/documentation/power/states.txt. Accessed on May 10, 2012.