ISSN:0975-9646

Automatic Load Balancing of Applications Using Class Constrained Bin Packing Algorithm: A Review

Akshay Karale[1], Shradha Katte[2],Prashant Yadav,[3] Chetan Godha[4]

[1,2,3,4]*Department of Computer Engineering,*
P.E.S. Modern College Of Engineering ,Pune-05,
University Of Pune, Maharashtra ,India

Abstract :**There are many applications that can benefit from automatic load balancing property. For that application instance is encapsulated inside the virtual machine(VM) and model it as the Class Constraint Bin Packing (CCBP) problem. Where each class represents an application and each server as a bin. Here our aim is to consider CCBP algorithm to achieve good demand satisfaction ratio and proper utilization of the servers. It also helps to reduce the number of servers and saves the energy. Here the proper security and authentication is provided while uploading and downloading application onto and from the server. Also the green computing is considered to put unused server in standby mode.**

Keywords- **Cloud computing, Virtual machine(VM), Automatic load balancing, CCBP etc.**

I. INTRODUCTION

Here we consider the virtual machine instance to carry out different operations. While uploading the files onto server user needs to decide the resources that are essential for that application. So here we provide a system that can carry out automatic load balancing of the application. The user only needs to do is to upload that application onto a server depending upon the demand of that application its instances get created on several virtual machines. For performing that operation we consider the effective load balancing algorithm called as Class Constrained Bin Packing (CCBP) algorithm.[1]

The green computing is considered for putting the unused bins into the standby mode. It also provides good demand satisfaction ratio and considers the proper utilization of the servers resources and as demand goes up and down we create the application instances onto several or fever servers.[1]

We build a system which supports our auto load balancing algorithm. We compare the overall performance of our system with the traditional system with the help of time line chart and graphs.

II. PROBLEM DEFINITION

For any kind of application we consider multiple kind of demands of the resources such as memory , hardware , CPU and different components. So here we consider mainly the CPU and memory for server resource utilization [1,2].

For those applications, memory is typically the determining factor on how many applications a server can run simultaneously, while CPU is the target resource we need to allocate among the application instances. Here depending upon the utilization of the CPU and the corresponding memory resources of server , we consider them as a prime factors for automatic load balancing of the appplication onto a specific bin[1].

The CPU capacity of server is the maximum number of application instances which can run on server simultaneously according to memory.Then the automatic load balancing problem is similar to the Class Constrained Bin Packing (CCBP) problem when we label each application as a class and treat the CPU demands of all classes as the items which need to be packed into bins. The difference is that CCBP problem does not have the "Minimize the placement change frequency" goal. Therefore, in order to solve problem, we modified the CCBP model to support the minimize the placement change frequency goal and also considered for that of departure of application from the specific bin.
[2]

III. LOAD BALANCING USING CCBP

In 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 the items into classes or colors. Each bin has capacity V and can accommodate items from at most C 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.We can model our resource allocation as the Class Constrained Bin Packing (CCBP) problem where each server is a bin and each class represents the specific application while considering load. 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[1].

For general applications memory is the basic resource that is to be considerd.The capacity of a bin represents the amount of resources available at a server for all its applications.[2] 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.[1] 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. 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.[1,2]

IV. ALGORITHM DETAILS

Our algorithm belongs to the family of the color set algorithm , but with significant modification to adapt to our problem. We 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 C, the maximum number of distinct classes in a bin. This ensures that items in a color set can always be packed into the same bin without violating the class constraint. The packing is still subject to the capacity constraint of the bin. All color sets contain exactly C colors except the last one which may contain fewer colors.[2]

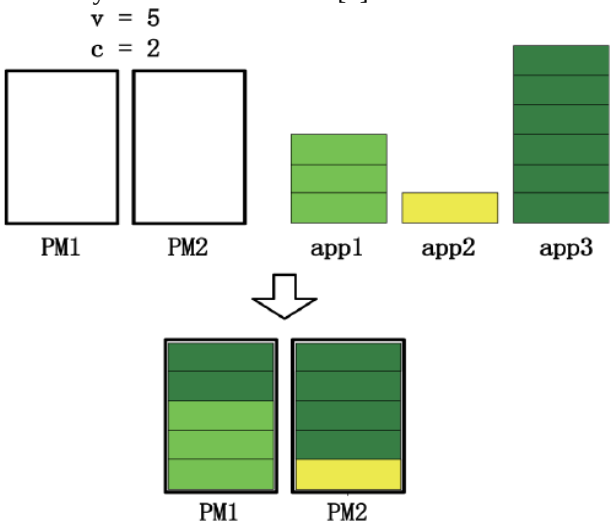

Fig.1 Class Constrained Bin Packaging [1]

A. Arrival Of The Application

The arrival of items with the corresponding color models the load increase of an application . To always pack the item into the unfilled bin for that we consider the naive algorithm. If the unfilled bin does not contain that color already, then a new color is added into the bin. This concurs to the start of a new application instance which is an expensive operation. As a substitute, 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 be the color of the new item and be any of the existing colors in the unfilled bin. We search for a bin which contains items of both colors. Then an item of color is moved from bin to the unfilled bin,which makes room for an item in bin where we pack the new item. If we are unable to find a bin which contains both colors, we see if we can shift the items using a third color as the intermediate.[1]

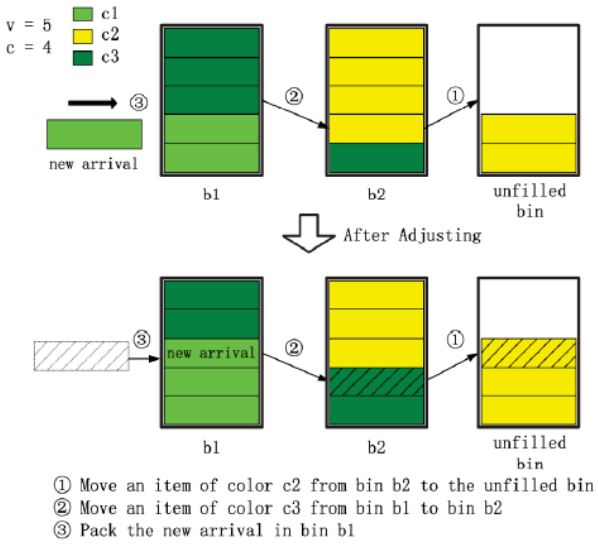

Fig.2 Arrival Of The New Item[1]

B. Departure Of The Application

The departure of previously packed items models the load decrease of an application. Here it is associated with a specific color, not with a specific item. To choose which item of that color is to be removed is given to the algorithm. Here we have to maintain the property that each color set has at most one unfilled bin.[1]

Our departure algorithm works as 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.[1,2]

Let 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. 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.[1,2] 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 bin move an item of color from the unfilled bin to departing bin. If the unfilled bin becomes empty, we can then remove it from the color set and shut down the corresponding server since all application instances there receive no load. It might look that a decrease in application load can result in the start of a new application instance.[1]

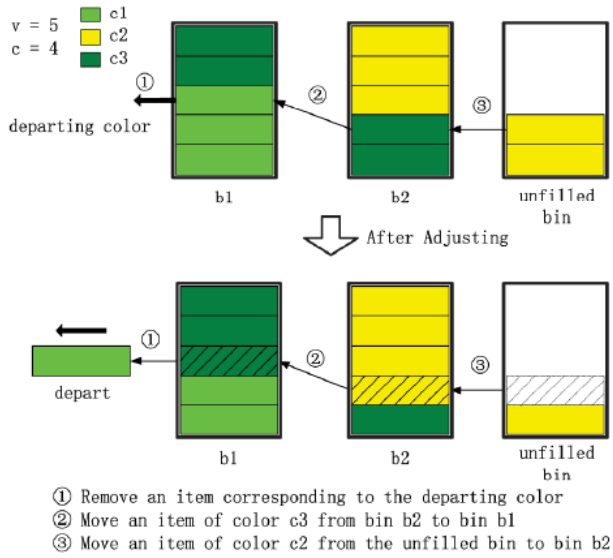

Fig.3 Departure Of The Existing Item [1]

C. Horizontal Scalability

Horizontal scalability is the ability to connect multiple hardware or software entities, such as servers, so that they work as a single logical unit. It means adding more individual units of resource doing the same job. In the case of servers, you could increase the speed or availability of the logical unit by adding more servers. Instead of one server, one can have two to ten, or more of the same server doing the same work. Horizontal scalability is also referred to as scaling out.[3]

Fig.4 Horizontal Scalability[3]

D. Vertical Scalability

Vertical scalability is the ability to increase the capacity of existing hardware or the software by adding more number resources. Adding processing power to a server to make it faster. It can be achieved through the addition of extra hardware such as hard drives, servers, CPU's etc. Vertical scalability provides more shared resources for the operating system and applications.[3]

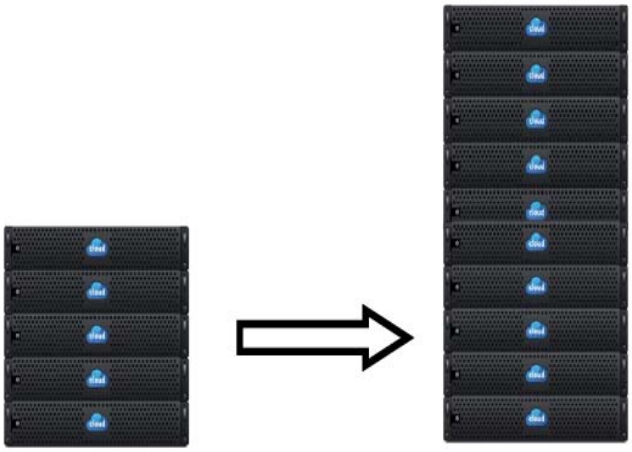

Fig.5 Vertical Scalability[3]

IV. CONCLUSION

We presented the design of a system that can scale up and down the number of application instance. For that we have a color set algorithm i.e CCBP algorithm to decide the application placement and the load distribution. Our result depend upon the size of the bin without violating the class constraints. Results into automatic load balancing of the applications.

REFERENCES

[1] Zhen Xiao, Senior Member, IEEE, Qi Chen, and Haipeng Luo as The Automatic Scaling of internet applications for cloud computing these SERVICES .YEAR-2014

[2] Sushil Deshmukh,Sweta Kale, Automatic Scaling Of web applications for cloud computing services.year-2014

[3] M.Krishanth,L.Arockiam And G.Justy Mirobi Research Scholar , The Department Of computer science ,St. Joseph's college , Auto Scaling of internet appications in a cloud comuting.year-2014

[4] Sasipriy,Ms.Kavitha,Department of computer science and engineering Sri Eshwar College of engineering,coimbatore ,A survey on Automatic Scaling of internet appications in a cloud environment.year-2014

[5] Oracle WebLogic Suite. http://www.oracle.com/us/products/middle/in ware/cloud-app-foundation/weblogic/overview/index.html Year- 2015

[6] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker, "Usher: An using extensible framework for managing clusters of virtual machines,"in the Proc. Large Install. Syst. Admin. Conf. (LISA'07).Year-2007

[7] J. Zhu, Z. Jiang, Z. Xiao, and X. Li, "Optimizing the performance the virtual machine synchronization for fault tolerance," IEEE Trans.Volu Comput., vol. 60, no. 12, pp. 1718–1729.Year- 2011

[8] L. Epstein, C. Imreh, and A. Levin, "Class constrained bin of packing revisited," Theor. Comput. Sci., vol. 411, no. 34–36, pp. 3073–3089. Year-2010.