International Journal of Engineering Science Invention (IJESI)
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 8 Issue 06 Series. II || June 2019 || PP 11-19

A Resource and Energy Consumption Aware Bin Packing Approach for Cloud Environment

Sanjit Kumar Acharya[1], Brojo Kishore Mishra[2]
[1] Department of Computer Science and Engineering, Roland Institute of Technology, India
[2]Department of Computer Science and Engineering, GIET University, India
Corresponding Author: Sanjit Kumar Acharya

Abstract*:* *Energy consumption management and resource management are two vital managing components of cloud data centers. From last two decades most of cloud data centers suffering from these two, mostly energy consumption and it become a serious issue now-a-days. After introducing the concept like virtualization and dynamic scheduling, customer required items are able to pack in efficient way, which leads to reduction in energy consumption in cloud data centers. In this work, an approach that uses an appropriate virtualization technology to dynamically allocate data center resources based on the demands of the application and availability of resources which support green computing by optimizing the number of active servers used. This is a modified existing on-line bin packing approach and develops a practical, efficient algorithm. The resources available are adjusted to each Virtual Machine both within and across physical servers. Experimental simulation results demonstrate that proposed approach achieves good performance compared to the other existing bin packing algorithms.*
In this work, an attempt has been made to achieve the resource utilization and also minimizes the energy consumption. The simulation results show case overhead of the existing work performance.
Keywords – ***bin packing,*** *cloud computing, data center, dynamic scheduling, virtualization*

Date of Submission: 22-06-2019 Date of acceptance: 08-07-2019

I. INTRODUCTION

Cloud computing is a networked based computing environment in which large groups of remote servers are networked to allow the centralized data storage and online access to computer services or resources. Amazon EC2 is a cloud service provider which provides networked virtual machines (VM) for lease i.e. it allows users to buy many VM instances as they want and operate them much like physical hardware. This is named as Infrastructure as a Service (IaaS), is broadly accepted in the industry.

The load applications changes dynamically within the VMs depending upon the requirements, because any number of resources can be requested variably by the application. For some applications, how much resource it will demand it is very difficult to forecast.

Although server consolidation has the effect of absorbing load fluctuation, the sum of the peak resource demand of VMs sharing a physical machine (PM) can be much higher than the mean. Thus, it is often inefficient to over-provision resources based on the peak demand because doing so will leave the resources under-utilized most of the time.

According to Live migration [1] a Virtual Machine (VM) can be migrated from one server to another without effecting the application running inside it. In this way VM layout can be used for adjusting the load balancing or for energy saving. For example, to reduce the load, VMs running on a server can be migrated to another server when the resource utilization of the server becomes too high.

Similarly, to save power, the VMs can be merged to a fraction of the servers, when the average server utilization in the system becomes too low and the idle servers can be allowed to sleep or to wake up [2][3]. Between the sleep and active states a negligible milliseconds can be considered, if Novel hardware assisted servers can be accomplished. [4].

When the utilization of server goes above some predefined threshold point it is called as "hot spot"[5]. An algorithm called black box/gray box (BG) periodically detects hot spots in the system and takes a decision that which VMs should be moved from the overloaded server to relieve the hot spots. It does not consolidate VMs at valley load for green computing purpose, since its main focus is overload avoidance. Nor does VectorDot [6] perform green computing.

A well- known packing problem which is one of the oldest and most thoroughly studied problems in computer science and combinatorial optimization is the bin packing problem which is a combinatorial NP-hard problem.

The bin packing optimization problem is used for putting a group of objects into a finite number of bins of capcity Vm. In this way minimum number of bins can be used. Here we focus on a number of problems which have many important applications in areas such as resource allocation, multi-processor scheduling, packet routing, stock cutting, paged computer memory systems, storage, loading trucks with weight capacity, creating file backup in removable media and technology mapping in FPGA implementation of custom hardware and many others.

This work explains the theoretical problems, which includes the classical bin packing problem for general and restricted inputs. We present this problem as well as its implication in data server storage. The classical bin packing problem models in the following way where a group of items of different size requires to be stored in the data center on a hard disk. Here we assume that the hard disk are of same size and items are stored without losing their generality. Here we further assume that the hard drives are of same size since purchasing the hard drives is in bulks which reduces the cost instead of buying them individually. We further assume that the size of a hard drive is 1(assuming that all sizes were scaled), and the hard drive represents a bin that needs to receive items to be stored. The aim is to optimize the number of bins or hard disks for storing the files. Some of the inputs that needs to be considered here is the parametric case, where sizes of items are much smaller than the size of recipients. Other generalizations are variable-sized bin packing where bins of several sizes are available.

Here we implement the approach of online bin packing algorithm with variable item size, which is capable to support both overload avoidance and also supports green computing. Here live migration concept is used to allocate data center resources dynamically based on the application demands and support green computing by optimizing the number of servers used.

II. LITERATURE REVIEW

Consolidation of resources and controlling resource utilizations has been researched in recent times. A framework for adaptive control of virtualized resources in utility computing environments is presented in [7]. In the past, cluster resource management has posed similar problems to that of consolidation of virtualized resources. For instance, [8] proposes a two level resource distribution and scheduling scheme, and [9] proposes a cluster level feedback control scheme for performance optimization. Power aware load balancing and migration of activity within a processor, among multicore processors and servers has also been studied in [10,11,12]. However, there has been little work on joint power and performance aware schemes for multi-dimensional resource allocation. Specifically, characterizing power-performance characteristics of consolidated workloads with respect to multiple resources has not been adequately addressed.

III. RESOURCE AWARE LIVE MIGRATION

The process of shifting the running instance of a virtual machine or application between different physical machines without disturbing the client or application is known as Live Migration. For the purpose of load balancing or saving energy this is used. The server provides various resources and when the utilization of resources becomes too high, some of the virtual Machines can be shifted to reduce the load of server. Similarly, when the average server utilization in the system becomes too low, the VMs can be merged to the servers, so that idle servers can be put to sleep in this way power utilization can be reduced. To save power. Memory, storage, and network connectivity of the virtual machine are transferred from the original guest machine to the destination. There are number of ways for shifting the contents of a VM's memory from one physical machine to another. However balancing the requirements for minimizing both down time and total time migration should be taken into consideration while shifting the live instance of the VM. The former is the period during which the service is unavailable due to there being no currently executing instance of the VM this period will be directly visible to clients of the VM as service interruption. The latter is the duration between when migration is initiated and when the original VM may be finally discarded and, hence, the source host may potentially be taken down for maintenance, upgrade or repair

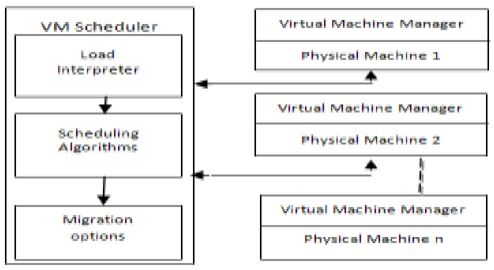


Figure 1: System Architecture

There are two modules in the VM Scheduler. The number of resources that can be demanded in the near future can be predicted by the load predictor. The Scheduling algorithm minimized the virtual machine's layout so that the resource demands are handled and resource wasted is minimized. Depending upon the estimates, every PM has been checked for the enough number of resources as per the required demands. If so, the resource allocation is being done by the VMM locally. Otherwise, the algorithm executes overload avoidance mechanism by sending away some of the VMs from the excess overloaded PMs. The underutilized PMS are also taken care by the scheduling algorithm for the VMs so that some of the PMs can be made to sit idle to save energy. The outcome of the scheduling algorithm is a VM migration list which is then send to the VMM for the used PMs for execution.

IV. PROBLEM DEFINITION

Online Bin packing algorithm is a resource allocation problem in cloud environment. In cloud when the application requests come, it needs to be process as soon as possible. So the resource demand should be handled in such a way that optimum number of servers are kept in active mode.

In the bin packing problem the servers which is taken as physical machine are packed with the virtual machine which is taken as items. Virtual machines are created according to the resource demand. When users send requests, the requests are consolidated and as per that variable sized virtual machines are created. The major issue here is that how the virtual machines allocate so that minimum numbers of server can be used. For the allocation of items inside the physical machine an algorithm is used which is called bin packing. In this algorithm, the size of the item is compared with the capacity of the physical machine and placed inside that. If it is not possible then a new physical machines is allocated. The entire thing is called Offline bin packing. Here a major problem arises when the application completes its execution (deadline is completed). Hence it is discarded form the bin, so the capacity of the bin is increase. In such case it may be possible that it can able to contain the items of another bin. Hence a new approach is rise, which is called online bin packing.

In online bin packing approach, it can overcome the problem of off-line. In each time interval when some virtual machines complete its deadlines and leave the corresponding physical machine, it checks all the used bin whether any migration is possible or not. If any bin can capable to contain another bin, one bin will move to the bin having minimum capacity. So that the bin having minimum capacity will utilize in an optimum way and other bin will in inactive mode. In this way minimum number of servers will remain active as compare to off-line bin packing.

V. ALGORITHM DETAILS

In this section, it is considered that virtual machine as item and the physical machine is bin. The size of bin is fixed i.e. 1 and the variable sized items need to pack. The key idea of VISBP (Variable Item Size Bin Packing) algorithm is to implement such algorithm where the minimum numbers of physical machines are used with accordance of migration should be less. Therefore the algorithm supports the concept of virtualization as well as live migration concept. For the better performance and according to their sizes we can divide items into four types. So that to minimize the amount of wasted space, the item types can be combined restrictedly inside the bin. The model of a framework can be designed using the following primitive functions and a set of definitions. We divide items into several types based on their sizes.

T-item (tiny): size $\epsilon(0, .33]$
S-item (small): size $\epsilon(0.33, 0.50]$
L-item (large): size $\epsilon(0.50, 0.66]$
B-item(big): size $\epsilon(0.66, 1]$

Table 1: Symbols and Notation

$BIN_i^{(binld)}$	Bin ld of ith bin
$BIN_j^{(binSize)}$	Bin capacity of ith bin
$BIN_j^{(mode)}$	**If** the bin is active then mode of the ith bin is True otherwise False
$BIN_i^{(binItem)}$	Bin ltem contains the item which are inserted to the ith bin
$ITEM_j^{(Size)}$	Size of the jth item
$ITEM_j^{(deadline)}$	Deadline of the jth item
$ITEM_j^{(arrival)}$	Arrival time of the jth item
$ITEM_j^{(itemId)}$	Item ld of the jth item

VI. ALGORITHM

Initially capacity of the bins are initialized to 1, the bin id sets to zero and the mode of the bin is false as the bin is at inactive state. The bin ltem which contains the items inside the bin is set to null. Likewise the items also need to initialize such as item id, item arrival time (same as running time slot) and item deadline.

```
$ITEM\_Initialization\ (slot\ s)$
==========================================
$Begin$
        $setVMList \leftarrow 0, j \leftarrow 0$
        $noVM \leftarrow rand(min, max);$
        $while(j \le noVM)$
                $ITEM_j^{(Size)} = rand(min, max);$
                $ITEM_j^{<deadline)} = rand\ (min,\ max);$
                $ITEM_j^{<arrival)} = s;$
                $j \leftarrow j+1;$
                $VMList.add\ (ITEM_i);$
        $Endwhile;$
$End;$
```

```
$BIN\_Initialization\ (slot\ s)$
==========================================
$Begin$
        $set\ PMList \leftarrow 0, i \leftarrow 0$
        $while(i \le noPM)$
                $BIN_i^{(binId)} \leftarrow 0$
                $BIN_i^{(binSize)} \leftarrow 1$
                $BIN_i^{(mode)} \leftarrow false$
                $BIN_i^{(inItem)} \leftarrow null$
                $i \leftarrow i+1;$
                $BINList.add\ (BIN_i);$
        $End\ while;$
$End;$
```

```
VISBP $Algorithm\ Main\ Function$
==============================================
$Begin:$
     $Timer\ t \leftarrow 0;$
       $PMList = BIN\_Initialization();$
       $while(true)$

              $VMList = ITEM\_Initialization(t);$
              $if(\ t \ne 0\ )\ then$
                     $VM\_Migration(t);$
              $Endif;$

              $VM\_Placement();$
              $Wait(\ s);$
              _
              $Wait\ for\ period\ of\ s$
              _
              $t \leftarrow t+1;$
              $Dead\_Line\_Function(\ t\ );$
       $Endwhile$
$end;$
```

```
$VM\_Placement(vm[], PM[]) Algorithm$
==============================
$Begin$
$VMList[] = ITEM\_Generator();$
$set\ i \leftarrow 0,\ b \leftarrow 0;$
$while(\ each\ ITEM_i\ :VMList[\ ])$
        $if\ (BIN_b^{binSize} > ITEM_i^{size})\ then$
                $BIN_b^{binSize} = BIN_b^{binSize} - ITEM_i^{size}$
        $//\ the\ capacity\ of\ the\ be\ reduce\ due\ to\ the\ placement\ of\ item insert\ the\ item\ inside\ BIN\ b$
        $else$

                $while(BIN_b^{binSize} < ITEM_i^{size})$
                        $b \leftarrow b + l\ \ //----\ search\ for\ the\ bin\ inside\ which\ ITEM;\ can\ be\ fit$
                $Endwhile;$
                $BIN_b^{(bin\ Size)} = BIN_b^{(bin\ Size)} - ITEM_j^{(size)};$
                $//\text{--}the\ capacity\ of\ the\ be\ reduce\ due\ to\ the\ placement\ of\ the\ item$
        $//---insert\ the\ item\ inside\ BIN_b;$
                $b = 0;$
      $endif;$
      $i \leftarrow i + 1;$
$Endwhile;$
$End;$
```

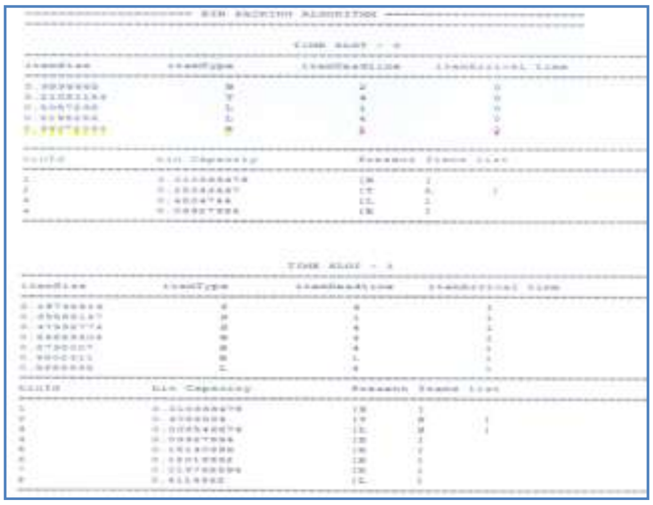
```
$DeadLine\_Function\ (BIN\ PMList[], int\ totalBin)$
==================================
$The\ items\ which\ reach\ its\ deadline\ automatically\ removed\ from\ the\ bin.$
$begin$
$set\ i \leftarrow 0, b \leftarrow 0$
$while(b < totalBin)$
        $while(i < BIN_b^{(bin\ Item)})$
          $set\ temp \leftarrow 0;$
          $temp = BIN_b(ITEM_j^{(arrival)}) + BIN_b(ITEM_i^{(deadline)});$
          $if\ (temp == running\ time\ slot)$
             $BIN_b^{(binSize)} = BIN_b^{(binSize)} + BIN_b(\ ITEM_i^{(Size)});$
            $BIN_b(remove(i));\ //\ remove\ the\ ith\ item\ which\ is\ present\ inside\ the\ both\ bin\ form\ the\ bth\ bin$
            $if\ (BIN_b^{(binS,ze)} == 1)\ //if\ the\ bin\ having\ a\ single\ item\ ,\ after\ removal\ it is\ set\ to\ inactive\ mode$

        $BIN_b^{(mode)}{}_{=false}.$
        $Endif;$
        $t \leftarrow -1;$
    $Endif;$
   $i \leftarrow i+1;$
        $Endwhile;$
$b \leftarrow b + 1$
$Endwhile;$
$end;$
```

```
VM _Migration(BIN PMList[],int totalBin)
=========================================================
If there is possible any bin can contain another bin, then the migration takes place.
Begin
set $j \leftarrow 0; a \leftarrow 1$;
while$\left( j < totalBin \right)$
  while$\left( a < totalBin \right)$
     if $\left( BIN_j^{(mode)} == true \ \&\& BIN_a^{(mode)} == true \right)$
          sort$\left( bin, totalBin \right)$; // sort the bin according to their capacity in ascending order
          if $\left( BIN_j^{(binSize)} >= \left( 1 - BIN_a^{(binSize)} \right) \right)$ // if possible Move the items of $BIN_a$ into the $BIN_j$
                                                  migration of the items from $BIN_a$ to BINj .
             $BIN_j^{(binSize)} = BIN_j^{(binSize)} - \left( 1 - BIN_a^{(binSize)} \right)$;
                   // capacity of the bin in addition of new items due to migration
             $BIN_a^{(binSlzc)} = 1$; // free the bin from which items are migrated
             $BIN_a^{(mode)} = false$.
           clearBIN(); // delete the items from the bin
          Endif
        Endif
    Endwhile;
Endwhile;
End;
```

VII. RESULT DISCUSSION

The implemented algorithm shows the following output, where the time slot represents the time interval which the algorithm is invoked periodically. In each time slot numbers of variable sized items are generated. Using Deadline_Function() the items which are completed are removed . If possible then migration takes place.



```
[illegible]
```

Figure 2: Sample Output of Simulation (Time slot 0 to 1)

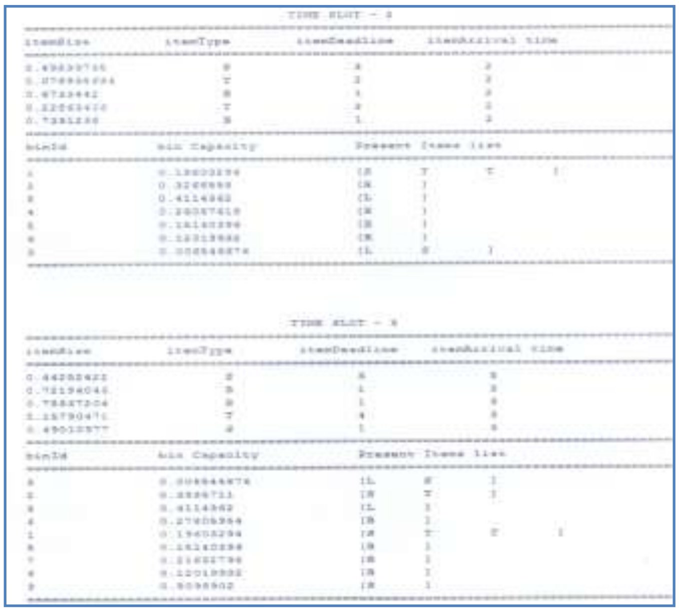


Figure 3: Sample Output of Simulation (Time slot 2 to 3)

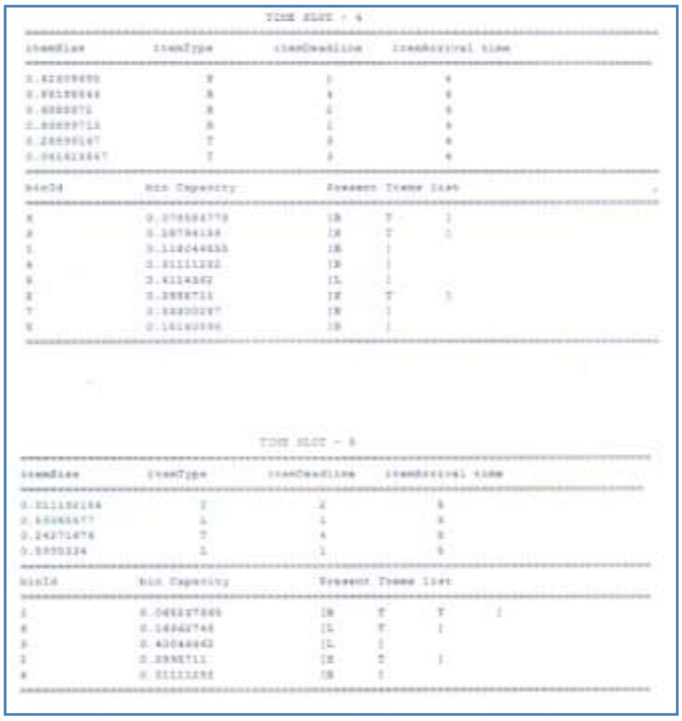


Figure 4: Sample Output of Simulation(Time slot 4 to 5)

VIII. RESULT ANALYSIS

Here in the following graph the comparison between the number of virtual machines (VM) in offline bin packing and (VISBP) Variable Sized Bin Packing and the active physical machine (APMs) in traced. It shows that the number of VMs is more in the VISBP as compared to the offline bin packing.

Figure 5 shows that for all the algorithms used the average number of APMs increase linearly, with the system size. The graph demonstrates that our VISBP algorithm achieves good green computing effect.

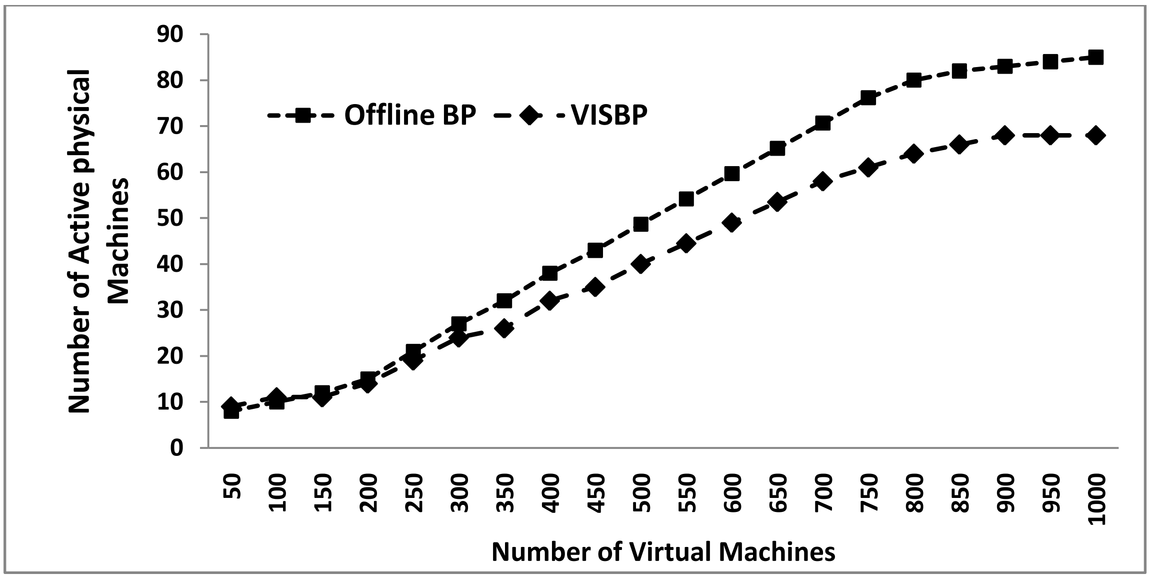


Figure 5: No. of APMs vs. no. of VMs

The below graph compares the number of APMs with the virtual machines and their time slots. In every time slots the number of virtual machine in offline bin packing and VISBP is increasing 500.

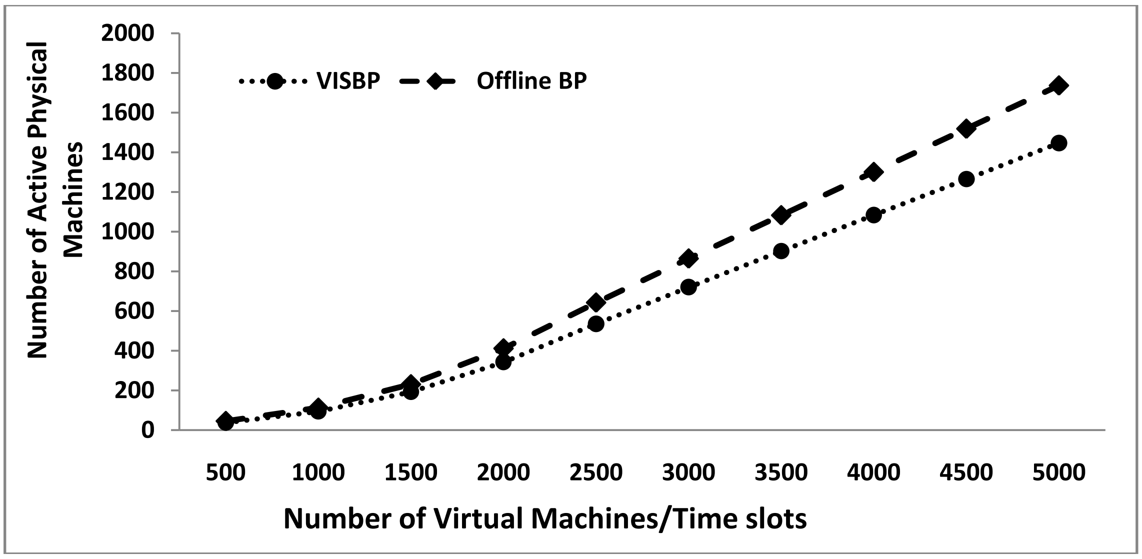


Figure 6: No. of virtual machines/time slot vs. no. APMs

The average number of hot spots in the system can be compared as pert the figure 6 when the utilization is above a predefined hot threshold a PM is considered a hot spot. Here 90% hot threshold is used. With the system size the number of hot spots increases roughly linearly for all the four algorithms.

Some closed bins might have sufficient space for the items but still according to VISBP algorithm it used only unfilled or unused bins to receive an item. This feature assumes that the bins are having some spare space for absorbing fluctuation of item size. This feature does not supported by the offline bin packing algorithm.

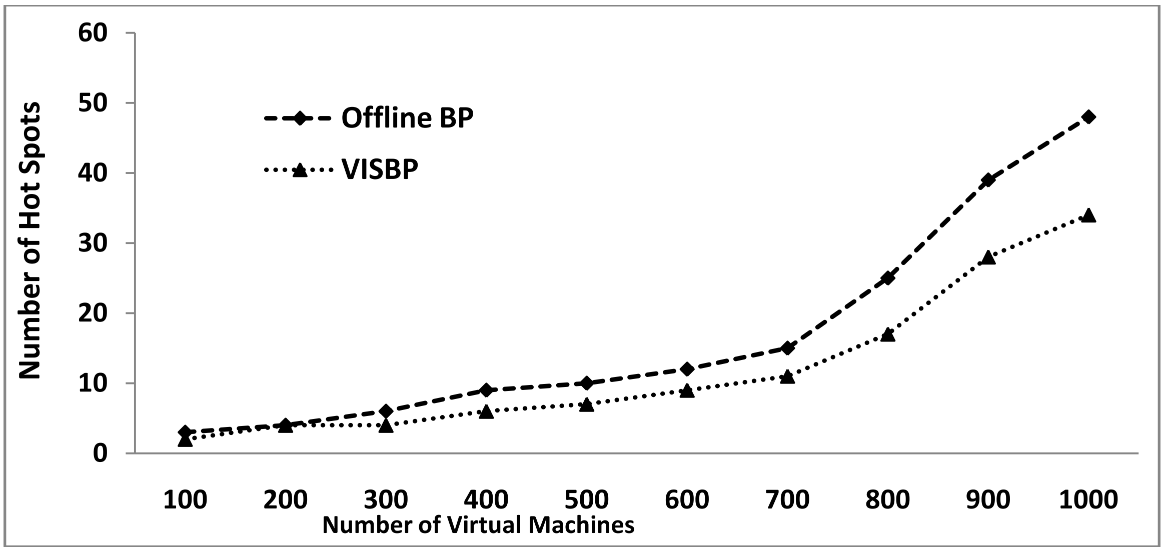


Figure 7: No. of VMs vs. no. of hotspots)

IX. CONCLUSION

We have presented the design, implementation, and evaluation of a practical online bin packing algorithm called the Variable Item Sized Bin Packing (VISBP) which can allocate data center resources dynamically through live VM migration. In this approach we have implemented an algorithm where minimum numbers of physical machines are used. We have used the concept of live migration for optimum use of data centers.

The implemented algorithm is compared to the existing off-line algorithm, results that the number of active physical machines is less, as compare to the offline algorithm. The number of hotspot detected in VISBP algorithm is less as compare to the offline. VISBP excels in hot spots mitigation and load balance. Variable item size bin packing is one of the algorithm in which optimum number of servers are used during the resource allocation in cloud.

As future works, plan can be made to support green computing by using the energy equation. Calculating the energy consumption in both case, online and offline bin packing and make comparison between them. The algorithm can be make more efficient which can give more optimum result by using a class constraint factor. It can also be develop for the heterogeneous sized physical machine. As in real time scenario it is not possible that all the physical machines are having same specification and capacity.

REFERENCES

[1]. Amazon ec2. http://aws.amazon.com/ec2/.
[2]. Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta, "Somniloquy: augmenting network interfaces to reduce PC energy usage," in Proc. of the 6th USENIX symposium on Networked systems design and implementation (NSDI'09), 2009.
[3]. Y. Agarwal, S. Savage, and R. Gupta, "Sleepserver: a software-only approach for reducing the energy consumption of PCs within enterprise environments," in Proceedings of the USENIX conference on USENIX annual technical conference, 2010.
[4]. D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in Proc. of the 14th international conference on Architectural support for programming languages and operating systems (ASPLOS'09), 2009.
[5]. T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and Gray-box strategies for virtual machine migration," in Proc.of the 4th Symposium on Networked Systems Design and Implementation(NSDI'07), 2007.
[6]. A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration arid load balancing in data centers," in Proc. of the 20th ACM/IEEE conference on Supercomputing (SC'08), 2008.
[7]. PADALA, P., et al. Adaptive control of virtualized resources in utility computing environments. In European Conference on Computer Systems (2007).
[8]. SHEN, K., et al. Integrated resource management for cluster-based internet services. SIGOPS Oper. Syst. Rev. 36, SI (2002), 225-238.
[9]. WANG, X., AND CHEN, M. Cluster-level feedback power control for performance optimization. In HPCA (2008).
[10]. HEO, S., BARR, K., AND ASANOVI´C, K. Reducing power density through activity migration. In ISLPED (2003).
[11]. GOMAA, M., et al. Heat-and-run: leveraging smt and cmp to manage power density through the operating system. SIGOPS Oper. Syst. Rev. 38, 5 (2004).
[12]. RANGANATHAN, P., et al. Ensemble-level power management for dense blade servers. In ISCA (2006).

Sanjit Kumar Acharya" A Resource and Energy Consumption Aware Bin Packing Approach for Cloud Environment" International Journal of Engineering Science Invention (IJESI), Vol. 08, No. 06, 2019, PP 11-19