Intelligent Automation And Soft Computing, 2020
Copyright © 2020, TSI® Press
Vol. 26, no. 1, 13–25
https://doi.org/10.31209/2018.100000047

# Application Centric Virtual Machine Placements to Minimize Bandwidth Utilization in Datacenters

**Muhammad Abdullah[1], Saad Ahmad Khan[1], Mamdouh Alenez[2], Khaled Almustafa[3], Waheed Iqbal[1]**

[1]Punjab University College of Information and Technology, University of the Punjab, Lahore, Pakistan
[2]College of Computer and Information Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia
[3]College of Engineering, Prince Sultan University, Riyadh 11586, Saudi Arabia

**ABSTRACT**
An efficient placement of virtual machines (VMs) in a cloud datacenter is important to maximize the utilization of infrastructure. Most of the existing work maximises the number of VMs to place on a minimum number of physical machines (PMs) to reduce energy consumption. Recently, big data applications become popular which are mostly hosted on cloud datacenters. Big data applications are deployed on multiple VMs and considered data and communication intensive applications. These applications can consume most of the datacenter bandwidth if VMs do not place close to each other. In this paper, we investigate the use of different VM placement methods to decrease the usage of bandwidth in different sizes of datacenters. We implemented and evaluated five different VM placement algorithms. Our comprehensive set of experiments show a significant decrease in bandwidth ranging from 65% to 78% can be achieved using the extended implementations of the knapsack and first fit VM placement methods.

**KEY WORDS**: Bandwidth Allocation, Datacenters, Resources Optimization, VM Placement.

## 1 INTRODUCTION

NOWADAYS, cloud computing becomes popular and attracted a large user base mainly due to its easy in use, pay-per-use, and dynamic resource provisioning features (Iqbal et al., 2011, 2016). The users can easily demand compute, storage, and network services from the Infrastructure-as-a-Service (IaaS) cloud. A public cloud maintains multiple datacenters in different regions to entertain the maximum number of users. However, management of IaaS cloud is concerned to efficiently utilize their datacenters to use minimal resources to serve a large user base. This will significantly reduce operational cost and maximizes the revenue.

Cloud computing is leveraging the benefits of hardware level virtualization which is achieved using hypervisors. A hypervisor enables a physical machine to host multiple virtual machines by providing a high level of isolation and efficient resource allocation to the VMs. There are various virtualization platforms available, some of the most commonly used are Xen, VMware, and Hyper-V. Every VM contains its own operating system and necessary binaries to run the VMs on physical machines. A user of VM gets all the features and feel of a real physical machine.

Big data, Internet of Things, and multi-tier applications are most commonly hosted on the cloud. These applications are deployed on multiple VMs and intensively share data among them to perform computation and user tasks. These type of applications in a datacenter consume significant bandwidth, however, intelligent placement of VMs may reduce the usage of bandwidth (Meng et al., 2010). In principle, it is possible to minimize the usage of bandwidth by placing coordinating VMs to same physical machine or closer to each other. However, this might introduce trade-off with minimizing the number of physical machines.

VM placement in a datacenter is a complex problem mainly due to dynamic resource demand of VMs and a limited number of physical machines and their resource

CONTACT Muhammad Abdullah muhammad.abdullah@pucit.edu.pk
© 2020 TSI® Press

capacities (Alahmadi et al., 2014; Singh et al., 2008a; Mi et al., 2010). Figure 1 shows the VM placement problem where VMs of different sizes can be deployed on PMs. There is an exponential number of possibilities to deploy the given VMs at any time interval on a set of PMs. Identifying the optimal placement is challenging as the VM placement problem is NPhard. A polynomial time solution for VM placement is not known. Therefore, most of the research to address this problem proposed and evaluated using heuristic and approximation based approaches. There are various optimization and constraints can be used to study VM placement problem. Most of the existing work to address this problem was focused to minimize energy consumption while placing the maximum number of VMs on a minimum number of PMs. However, to address the need for minimizing bandwidth utilization in datacenter for applications hosted on multiple VMs are not well studied.

In this paper, we evaluate traditional methods for VM placement and also introduced two approaches namely application level knapsack and application level first fit to specifically help to minimize the datacenters bandwidth usage. We assumed dynamic demand of VMs and used five different methods to place them on different sizes of datacenters. We performed experiments to evaluate different parameters like resource wastage, bandwidth utilization, placement time, and number of PMs used to deploy the VMs. Where resource wastage refers to the sum of unallocated resources including CPU, memory, and disk of all PMs. The exact contributions of this paper are:

1. Implement and present the traditional VM placement algorithms named round robin, knapsack, and first fit.
2. Modify knapsack and first fit algorithms to place VMs to minimize the overall bandwidth of datacenters.
3. Study the impact of these five algorithms on bandwidth, placement time, and resource wastage using hetrogenous demand of VMs on different sizes of datacenters.

Rest of the paper is organized as follows. We discussed related work in Section 2. VM placement optimization problem is discussed in Section 3. We provide details about VM placement algorithms in Section 4. Detailed experimental evaluations are given in Section 5. In Section 6, conclusions are drawn and future work is discussed.

## 2 RELATED WORK

THERE have been various contributions to address VM placement problem in datacenters. For example, many of the authors (Ghribi et al., 2013; Alahmadi et al., 2014; Bobroff et al., 2007; Singh et al., 2008a; Mi et al., 2010) propose and evaluate VM placement methods only considering CPU resources. There have been also some work considering multiple resources to place virtual machines. For example, authors in (Song et al., 2014; Dong et al., 2015; Chen et al., 2011; Wood et al., 2009; Singh et al., 2008b; Ferdaus et al., 2014; Gao et al., 2013; Tao et al., 2016) use CPU, I/O, and network resources to schedule VMs. However, application specific scheduling to minimize overall bandwidth utilization in the datacenter is not addressed in any of the above work.

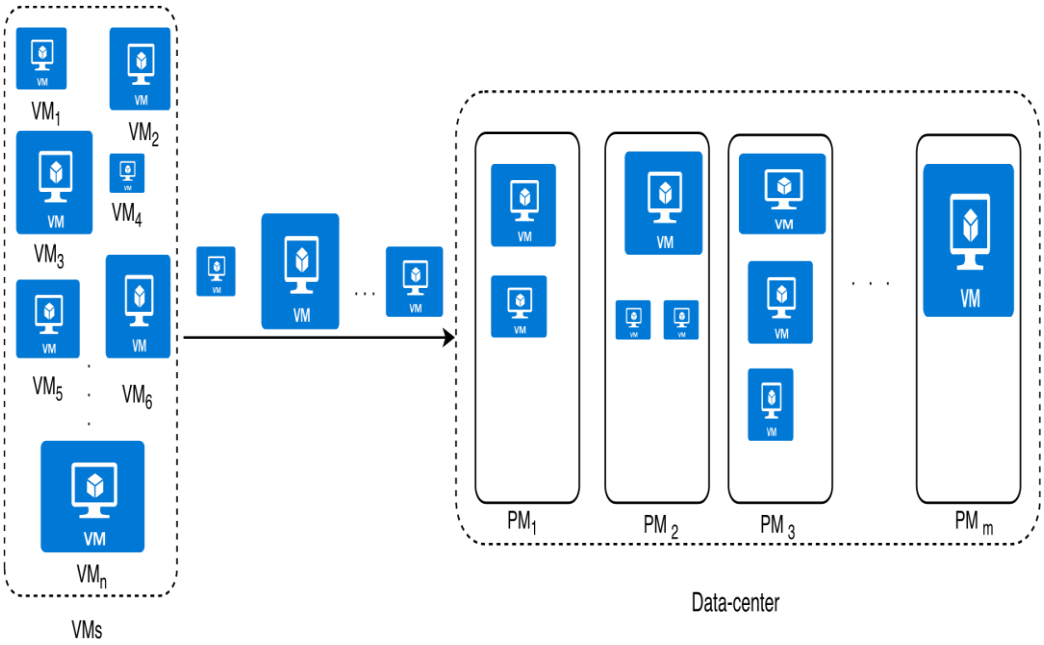


**Figure 1. VM placement problem. Different sizes of VMs can be placed on PMs with exponential possibilities.**

A comprehensive survey of VM placement algorithms is performed by (Masdari et al., 2016). Authors in this paper survey 51 different papers addressed VM placement problem. These papers were classified in 18 different techniques. These techniques are considering some of the metrics i.e. minimize the number of PMs, reduce the power consumption, increase the performance etc. for placing the VMs. Most of the techniques use heuristics to place the VMs efficiently. They also discussed the limitations and advantages of the currently used placement algorithms. However, the paper discussed bandwidth optimization are fewer and none of them addressed the overall bandwidth minimization in datacenters using heterogeneous demands of VMs. Bhuvan et al. (2007) discussed the application placement problem as NP-hard and proposed and evaluated different approximation algorithm to address the application placement problem efficiently.

Wang et al. (2011) addressed to reduce the number of physical servers used to place VMs. They reduced the VM placement problem to stochastic bin packing problem by assuming that at a given time interval the bandwidth demand for the VMs is known. As a solution for stochastic bin packing problem, they presented an online packing algorithm. Their work assumes homogeneous demand for each VM, however, diversity in demand of VMs has not been addressed in this work. Another approach for VM placement had been presented by Zheng et al. (2016). In this approach, authors used biogeography-based optimization methods and according to them, these optimization methods were firstly used for VM placement where their goal is to minimize the power consumption and resource wastage of PMs. They also migrate the VMs to minimize the resource wastage of PMs. Recently Camati et al. (2014) presented a new approach to minimize energy consumption and VM migration overhead by addressing the VM placement problem. Their solution is based on multidimensional knapsack problem.

Xu & Fortes (2010) proposed and evaluated a VM placement algorithm to minimize total resource wastage and power consumption simultaneously in datacenters using a fuzzy logic based solution to address the multi-objective optimization problem. Thiruvenkadam & Karthikeyani (2014) the VM scheduling algorithm for minimizing the server overload issues in datacenters and also to minimize the power consumption. They represent live VM migration without affecting QoS to address the problem. (A. K. Paul et al., 2014) discussed the VM placement problem in the datacenter to minimize the energy and cost. The authors represent this problem as an optimization problem and model it as a bin packing problem. Their simulation result shows the minimization of cost and energy in the datacenters. Ilkhechi et al. (2015) presented an algorithm to maximize the number of VMs to deploy on PMs. They consider the closeness of VMs on the network as satisfaction which reduces the network bandwidth. However, they do not consider other VM demands like CPU, memory, and disk to place the VMs.

A recent work by Shen et al. (2016) proposed a scheme named AppBag (Applicationaware Bandwidth Guarantee) which consider one step-ahead information to schedule VMs to optimize bandwidth within datacenter. Their approach place and migrate the VMs to minimize the datacenter network traffic. The proposed approached minimized the migration overhead by only considering VMs which are intensively communicating with each other. Chen et al. (2016) had addressed the challenge of fair allocation of network bandwidth in DCNs environment. The proposed solution aims to place maximum VMs by effectively utilizing the available bandwidth.

A most recent work by Shabeera et al. (2017) presented a new approach for VM placement to minimize the bandwidth utilization in datacenters. They also proved the VM and data placement problem to be NP-Hard by reducing Knapsack problem (NP-complete) to it in polynomial time. The proposed method is heuristic algorithm based on Ant Colony Optimization. However, the assumed homogeneous demands of the VMs and also consider all VMs belong to one application.

None of the work discussed above considered heterogeneous VMs demanding different resources to place them in the datacenter to minimize the overall bandwidth utilization. Here, in this paper, we address this problem in the datacenters by extending standard knapsack implementation and compare it with existing state of the art approaches used nowadays in the real datacenters.

## 3 VM PLACEMENT OPTIMIZATION MODEL

IN this section, we explain the VM placement optimization model. We aim to minimize overall bandwidth utilization in the datacenter by effectively placing VMs. Following are the important parameters in our optimization model:

- $P = \{p_1,p_2,...,p_m\}$ is a set of available physical machines
- $\forall p_i = \{c_1,c_2,...,c_k\}$is a set of available resources capacity, where $p_i \in P$
- $A = \{a_1,a_2,...,a_q\}$ is a set of applications
- $\forall a_i = \{v_1,v_2,...,v_t\}$is a set of virtual machines belongs to application, where $a_i \in A$
- $V = \{v_1,v_2,...,v_n\}$ is a set of all virtual machines of all applications
- $\forall v_i = \{r_1,r_2,...,r_k\}$ is a set of resource demands, where $v_i \in V$
- $x_{i,j} \in \{0,1\}$,where $1 \leq i \leq m, 1 \leq j \leq n$
- $x_{i,j} = 1 \Leftrightarrow v_j$ is placed on $p_i$
- $x_{i,j} = 0 \Leftrightarrow v_j$ is not placed on any $p_i$

- $w_{j,l}$ the $r_l$ resource of $v_j$
- $C_{i,l}$ the available resource capacity $c_l$ of physical machine $p_i$
- $r_{j,b}$ the bandwidth resource capacity of virtual machine $v_j$

Our proposed VM placement optimization model contains of two objective functions and two constraints. We explain them using the following equations:

$$\text{Maximize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j} \,.\, v_j \tag{1}$$

$$\text{Minimize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j} . v_j \;\; r_{j,b} \tag{2}$$

$$\text{Subject to} \quad \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{k} x_{i,j} \,.\, w_{j,l} \;\; \leq C_{i,l} \tag{3}$$

$$\text{and} \quad \forall \text{j} \sum_{i=1}^{m} x_{i,j} \;\; \leq 1 \text{ where } 1 \;\; \leq \text{j} \;\; \leq \text{n} \tag{4}$$

Equation 1 is used to achieve the objective to place maximum number of VMs on the available physical machines. Where $v_j$ represents the $j^{th}$ VM and $x_{i,j}$ becomes 1 whenever $v_j$ is placed on a PM $p_i$ otherwise it remains 0. Equation 1 ensures that the sum of the placed VMs is maximum, where $m$ represents the total number of PMs and $n$ represents the total number of VMs in the datacenter. Equation 2 shows our objective to minimize the overall bandwidth utilization of datacenter. Where $r_{j,b}$ shows the $j^{th}$ VM's bandwidth resource. Equation 2 ensures the sum of bandwidth resource of all placed VMs is minimum.

Equation 3 enforce a constraint that ensures that all the VMs placed on a PM does not demand more than available capacity. In this equation $x_{i,j}$ is a binary variable which initializes to 1 if $j^{th}$ VM placed on $i^{th}$ PM otherwise remains 0, $w_{j,l}$ shows the $l^{th}$ resource of $j^{th}$ VM. Equation 3 ensures that the sum of $l^{th}$ resource of all placed VMs on $i^{th}$ PM is less or equal from available capacity of $l^{th}$ resource of $i^{th}$ PM $C_{i,l}$. Where $m$ is total number of PMs, $n$ is total number of VMs and $k$ is total number of resources (e.g CPU, memory, disk, bandwidth). Equation 4 shows the constraint to ensure that each VM is placed on only one PM. This equation ensures that the sum of $x_{i,j}$ for all PM is less or equal to 1 for a specific VM $j$ which means that $j^{th}$ VM is only placed on one PM and if the sum of $x_{i,j}$ equals to 0 then the $j^{th}$ VM is unplaced.

In this paper, we achieve these objectives by greedily placing the VMs of the same application on the minimum number of PMs. Our approach assumes that multiple VMs belongs to an application ideally be placed on a single PM. It would reduce the bandwidth utilization significantly, however, practically it may not be possible as the applications may need to spawn a large number of VMs which cannot be deployed on a single PM. Therefore, our approach tries to minimize the number of physical machines for placing VMs belonging to an application.

## 4 VM PLACEMENT ALGORITHMS

IN this section, we explain traditional methods for VM placement and also modify explain two approaches namely application level knapsack and application level first fit which we modified to minimize the bandwidth in datacenets.

### 4.1 Round Robin

Round robin algorithm places VMs in a circular order to the PMs. It assumes the list of PMs as a circular list and schedules the incoming VMs to the list of PMS in circular order. For example, if we have a list of four VMs (v1, v2, v3, and v4) and two PMs (p1 and p2) then this method will place v1, v3 to p1 and v2, v4 to p2 provided that VMs resource demand is less than PMs capacity. This algorithm is also placed VMs in linear time, however, it may use more PMs comparing to first fit method. Round-robin VM placement algorithm is available in many open-source cloud middlewares including OpenNebula, EUCALYPTUS, and OpenStack.

### 4.2 First Fit

First fit VM placement algorithm identifies the most appropriate PM meets the demand of the VM by linearly searching the list of available PMs. We implemented this approach as explained in Kuo et al. (2015). This method is a linear time algorithm to schedule a set of VMs. It has the ability to use the minimum number of PMs as it tries to exhaust the PMs in order.

### 4.3 Multi-Dimensional Knapsack

Algorithm 1 explains our proposed knapsack approach for VMs placement. We implement the traditional knapsack greedy method to place VMs by considering the capacity of PMs as knapsack dimensions. The proposed method is a variant of the implementation used in Camati et al. (2014). In our implementation, we assume that a set of VMs request arrives at a given time interval and we need to deploy them on the PMs. We also consider that resources can be allocated in specific units. For example, CPU resources can be provisioned as at least 1 core to the VMs. Therefore, we define the minimum allocation unit for memory, CPU, and disk resources. We use *cpu allocation unit*, *memory allocation unit*, and *disk allocation unit* variables as CPU, memory, and disk minimum resource limits in the Algorithm. Our knapsack implementation yield O($m \times n \times c$ *units* $\times m$ *units* $\times d$ *units*) time complexity. Where $m$ is the number of PMs, $n$ is the number of VMs to place, c

units is the total CPU units, m units is the total memory units, and d units is the total disk units of a PM. This algorithm maximizes the number of VMs placed and minimize the usage of PMs, however, it takes more time to place VMs as compare to first fit and round robin.

### 4.4 Application Centric Placement with Knapsack to Minimize Bandwidth

Algorithm 2 shows our approach for application centric placement of VMs using the knapsack method. In this method we use the knapsack implementation explained in Algorithm 1. The proposed algorithm takes the list of applications and the list of PMs as input and iterates for each application to greedily place all its VMs to a minimal number of physical machines using knapsack implementation. This algorithm place application specific VMs to completely free PMs which ensures to use a minimal number of PMs used to host the application centric VMs. It would minimize the overall bandwidth utilization in the datacenter significantly. However, if no completely free PM has remained left then it will place remaining applications specific VMs to the free capacities of PMs. This algorithm takes more time to schedule all VMs comparing to knapsack.

### 4.5 Application Centeric Placement with First Fit to Minimize Bandwidth

Algorithm 3 shows our approach for application centric placement of VMs using first fit. This proposed algorithm also takes the list of applications and the list of PMs as input and iterates for each application to place all its VMs to minimize the number of PMs same as Algorithm 2 but use first-fit of knapsack implementation. This algorithm also minimizes the overall bandwidth utilization in datacenter significantly. This approach takes less time as compared to the Algorithm 2.

## 5 EXPERIMENTAL EVALUATION

IN this section, we explain the experiments performed to evaluate the proposed methods to place VMs. We performed simulations using a tool written in Java to evaluate the proposed methods. The source code of this simulation tool is available at Abdullah (2017). We used a core i3 machine with 3.0 GHz CPU, and 8 GB Memory to perform all the simulations.

### 5.1 Experimental Datacenter Scenarios

We emulate four type of datacenter deployment scenarios namely micro, small, medium, and large. In each of these deployments, we variate the number of PMs, applications, and VMs. We used 10, 100, 500, 1000 PMs in micro, small, medium, and large datacenters respectively. For each of the deployment scenario, we conducted experiments three times to place VMs using our proposed algorithms. In each of the iteration we keep the number of PMs and applications same, however, we assign a random number of VMs to each application. Each VM can demand from 2 to 5 CPU units, 3 to 7 memory units, and 5 to 25 disk units. We assume a simple one vCPU mapping to one physical CPU. Each VM of the application can also demand a specific bandwidth with other VMs of the same application. We randomly assign bandwidth utilization between 0 to 50 KB per second for each VM with other VMs of the same application. We generate the number of VMs and their demands in the beginning of each iteration and then used all five algorithms independently to place them on the specific datacenter deployment scenario.

**Algorithm 1** VM Placement Using Multi-dimensional Knapsack

```
1: procedure Knapsack(V,P)
2:     for each p in P do
3:             cpu, memory, disk ← getFreeCapacity(p)
4:         c_units = cpu/cpu_allocation_unit
5:         m_units = memory/memory_allocation_unit
6:         d_units = disk/disk_allocation_unit
7:         K[V.size, c_units,m_units,d_units]        // Initialize a multi-dimension array
8:         i=0
9:         for each v in V do
10:            for c in c_units, and m in m_units, and d in d_units do
11:                if (v.cpu > c) || (v. memory > m) || (v.disk > d) then
12:                    K[i,c,m,d]=K[i-1,c,m,d]
13:                else
14:                    K[i,c,m,d]=Max(K[i-1,c,m,d] , K[i-1,c-v.cpu,m-v.memory,v.disk]+1)
15:                end if
16:            end for
```

```
17:                 i++
18:             end for
19:             for j= V.size to 0 do
20:                 if K[j,c,m,d] ≠ K[j-1,c,m,d] then
21:                     V(j).PM=P                    // V(j) is allocated on physical machine p
22:                     c = c – v.cpu
23:                     m = m – v.memory
24:                     d = d – v.disk
25:                     V = V – {v}
26:                 end if
27:             end for
28:     end for
29: end procedure
```

**Algorithm 2** Application Level Placement Using Multi-dimensional Knapsack

```
1: procedure ApplicationLevelKnapsack(A, P)
2:         for each app in A do
3:          P'← getFreePhysicalMachines(P)
4:         if P'.size > 0 then
5:             Knapsack ( P', app.vm_s )        // Call Knapsack to place the VMs of the app using
                                                    completely free PMs
6:         Else
7:             Knapsack(P, app.vm_s )           // Call Knapsack to place the VMs of the app using free
                                                    capacity of all PMs
8:         end if
9:       end for
10: end procedure
```

**Algorithm 3** Application Level Placement using First Fit

```
1: procedure ApplicationLevelFirstFit(A, P)
2:         for each app in A do
3:           P'← getFreePhysicalMachines(P)
4:         if P'.size > 0 then
5:             Use First Fit Algorithm to place the VMs of the app
               using completely free PMs P'
6:         Else
7:             Use First Fit Algorithm to place the VMs of the app
               using free capacity of all PMs P
8:         end if
9:       end for
10: end procedure
```

Most of the cloud datacenters use homogeneous PMs (Crago et al., 2011), therefore, the PMs we used in our experimental evaluation are homogeneous. Each PM consisting of a 16 CPU units, 20 memory units, and 100 disk units. Where each CPU unit is considered as 1 core, each memory unit is considered as 512 MB and each disk unit is considered as 100 GB. Table 1 summarizes the experimental datacenter scenarios and shows the number of PMs use, the number of applications to deploy, the total number of VMs used to deploy the applications with each iteration.

### *5.2 Evaluation Metrics*

To evaluate the effectiveness of the proposed algorithms, we calculate bandwidth utilization, placement time, resource wastage, the percentage of VMs unplaced, and percentage of PMs used. We explain each of these in the following subsections.

#### *5.2.1* Bandwidth Utilization

We calculate the total bandwidth utilization of the datacenters after placing the VMs using the proposed algorithms. A VM placement algorithm that minimizes the overall bandwidth utilization of datacenter is considered the best.

#### *5.2.2* Placement Time

Placement time represents the time taken to place the VMs in the datacenter using the proposed algorithms. A VM placement algorithm with minimum placement time is considered the best as it will take less time compared to others to take the decisions to assign VMs to the PMs.

#### *5.2.3* Resource Wastage

This metric represents the portion of computing resources left unallocated at each PM which cannot be used to place any VM. Ideally we want 100% resource of PMs to be used, however, due to heterogeneous resource requirements of the VMs, we observe resource wastage on PMs. The lower waste of CPU, memory, and disk is desirable because these reflect that the algorithm is capable to minimize the number of PMs to host the maximum number of VMs. Therefore, a VM placement algorithm with minimum resource wastage is considered the best.

#### *5.2.4* Percentage of VMs Unplaced

The VMs that are not placed on any of the PM is considered as an un-placed VM. This can happen when the placement algorithm does not use the PMs efficiently or the number of VMs to allocate on PMs are higher than the actual capacity of the available PMs. A minimum percentage of un-placed VMs is considered better because it indicates that algorithm is capable to lace maximum VMs.

#### *5.2.5* Percentage of PMs Used

A PM is considered as free if none of the VM is placed on it. However, a PM is considered as used if at least one of the VM is placed on it. We calculate the percentage of total PMs used to host the VMs in the datacenter. An algorithm gives the minimum percentage of PMs used is considered the best as it uses less number of PMs to host the given set of VMs.

**Table 1. Experimental datacenters scenarios for VM placement methods**

| Deployment | # of PMs | # of Applications | Iteration # | # of VMs |
|---|---|---|---|---|
| Micro | 10 | 15 | 1 | 38 |
| | | | 2 | 36 |
| | | | 3 | 40 |
| Small | 100 | 150 | 1 | 386 |
| | | | 2 | 369 |
| | | | 3 | 373 |
| Medium | 500 | 700 | 1 | 1807 |
| | | | 2 | 1686 |
| | | | 3 | 1738 |
| Large | 1000 | 1500 | 1 | 3753 |
| | | | 2 | 3808 |
| | | | 3 | 3810 |

### 5.3 Experimental Results

The algorithms explained in Section 4 are evaluated using four type of datacenter deployment scenarios and dynamic resource demands of VMs. We used round robin as baseline method to compare the performance of other VM placement algorithms.

Figure 2-a shows the overall bandwidth utilization in all of the deployment scenarios against each placement algorithm. We show the average bandwidth utilization of all three iterations with error bars showing the standard deviation. The bandwidth utilization is significantly minimized using application level knapsack and application level first fit methods. Application level knapsack has reduced 78.01%, 71.89%, 76.46%, and 71.43% overall bandwidth utilization in micro, small, medium and large deployment scenarios respectively as compared to the baseline round-robin method. The application level first fit algorithm also minimized 77.24%, 66.15%, 71.75%, and 65.43% overall bandwidth utilization for micro, small, medium and large deployments respectively when comparing to the round robin method.

Figure 2-b shows the total time taken by each placement method for the different datacenter deployment scenarios. Application level knapsack method takes more time as compared to our baseline, however, application level first fit takes almost equal time as compared to the baseline method. Our baseline method takes only 24ms time to place the VMs for the large deployment scenarios. Whereas first fit algorithm takes maximum 136 ms, knapsack takes 8.46 minutes, application level knapsack algorithm takes 2.66 hours and application level first fit algorithm takes 52.8ms time to place VMs in the large deployment scenario.

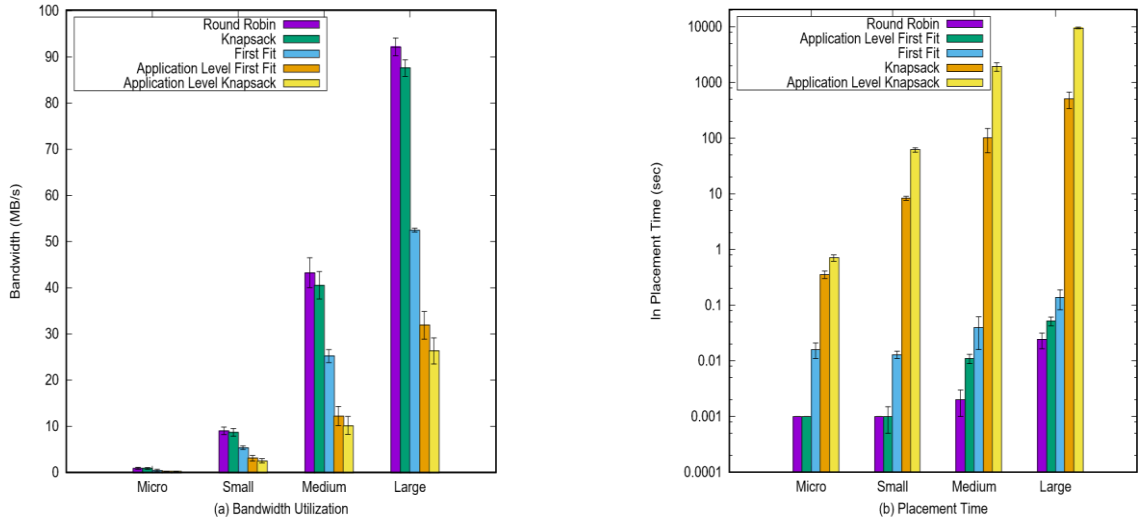


**Figure 2. Bandwidth utilization and placement time using VM placement methods on different datacenter deployment scenarios.**

Figure 3 shows a comparisons of all the algorithm in each of the datacenter deployment scenarios. We show the average CPU, memory, and disk wastage of all three iterations with error bars. Figure 3-a, Figure 3-b, Figure 3-c, and Figure 3-d show the resource waste using micro, small, medium, and large deployment scenarios respectively. In all of the deployment scenarios knapsack outperforms all other methods to minimizing the resource wastage. In average, knapsack saved 9.07%, 9.14%, 12.33%, and 8.51% CPU resources in micro, small, medium, and large type of datacenter deployment scenarios respectively when comparing to baseline method. Similarly, it also saves 9.15%, 9.24%, 12.27%, and 8.39% memory resources in micro, small, medium, and large type of datacenter deployment scenarios respectively when comparing to baseline method. However, disk wastage is not significantly minimization by any of the placement algorithm.

Knapsack algorithm performed best to minimize the resource wastage mainly due to the fact it identifies the maximum number of possible VMs to be placed on one PM. This leads to using a minimum number of PMs which also helps to minimize the overall resource wastage in the datacenter. However, the wort performing algorithms in resource wastage are First Fit and Round Robin. Both of these algorithms, exhaust all PMs iteratively and does not consider minimizing the number of PMs which leads to higher resource wastage.

Figure 4 shows the percentage of total VMs left without placing (VMs unplaced) and percentage of total PMs used (PMs used) in each of the deployment scenarios. Figure 4-a, Figure 4-b, Figure 4-c, and Figure 4-d show the percentage of VMs unplaced and percentage of PMs used during the micro, small, medium, and large deployment scenarios respectively. The lower the VMs unplaced and lower the PMs used

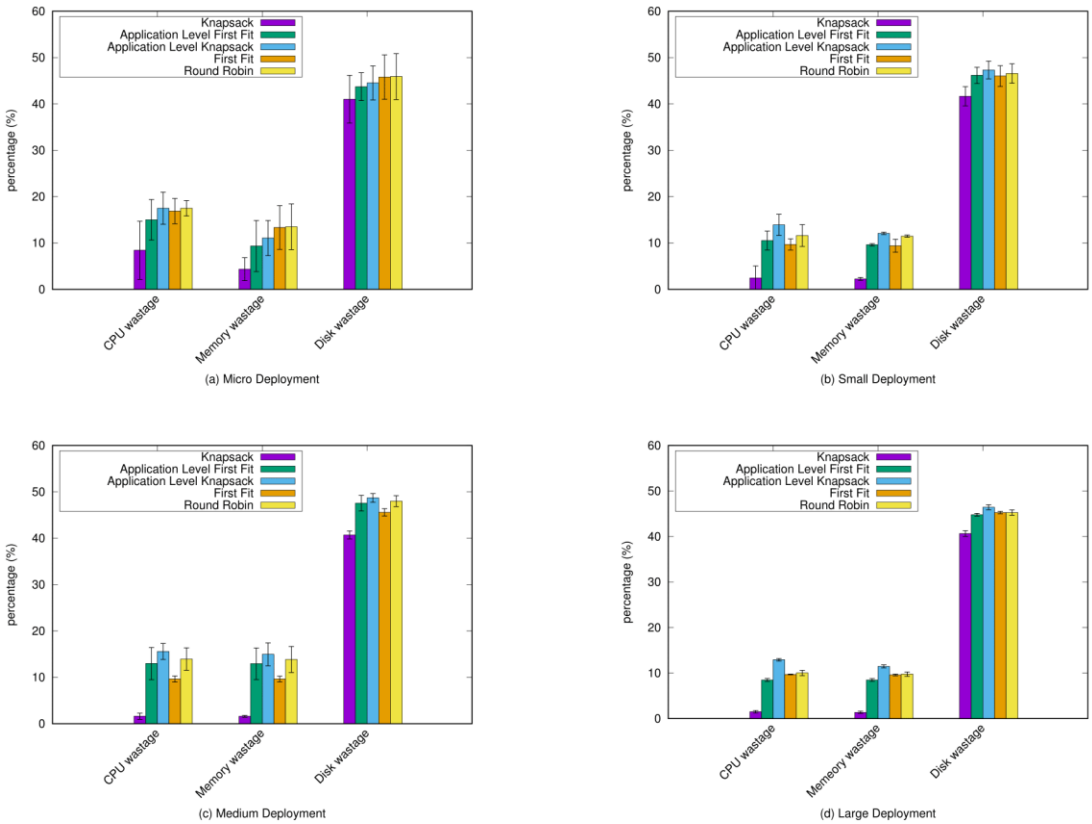


**Figure 3. Resource wastages using VM placement methods on different datacenter deployment scenarios.**

is better. In most of the deployment scenarios 100% of the available PMs used by the round robin, application level knapsack, and application level first fit. However, knapsack saves 3.66%, 11.46%, and 3.93% PMs in small, medium, and large datacenter deployment scenarios. The first fit also saved 3.86% PMs in medium datacenter deployment scenarios. Knapsack also outperforms to minimize the number of VMs unplaced in all of the deployment scenarios. It placed all the VMs in every deployment scenario except micro where knapsack left 1.71% VMs unplaced. All other four placement algorithm approximately left the same number of VMs unplaced in all of the deployment scenarios.

Knapsack algorithm fully utilizes a PM before start using another PM to place the given set of VMs, therefore, it performs the best to minimize the use of PMs. However, Round Robin iteratively use all PMs to place VMs and perform worst to minimize the use of PMs.

Application level Knapsack and application level First Fit allocate all VMs belonging to the same application on a minimum number of PMs which helps to minimize the PMs in some of the deployment scenarios.

Table 2 summarizes the experimental evaluation and shows CPU waste, memory waste, disk waste, and bandwidth minimization comparing to the baseline round robin algorithm for each of the deployment scenarios. The negative values in the table indicate better than round robin and positive values indicate worse than round robin performance for the different measurements. For example, application level knapsack minimized 78.01% bandwidth comparing to the round robin. Knapsack is very effective to minimize the resource waste, however, does not effectively minimize the bandwidth usage in the datacenters. Both application level knapsack and application level first fit outperforms in minimizing the bandwidth in all type of deployment scenarios. However, they will waste more resources comparing to knapsack which can overcome by using additional PMs.

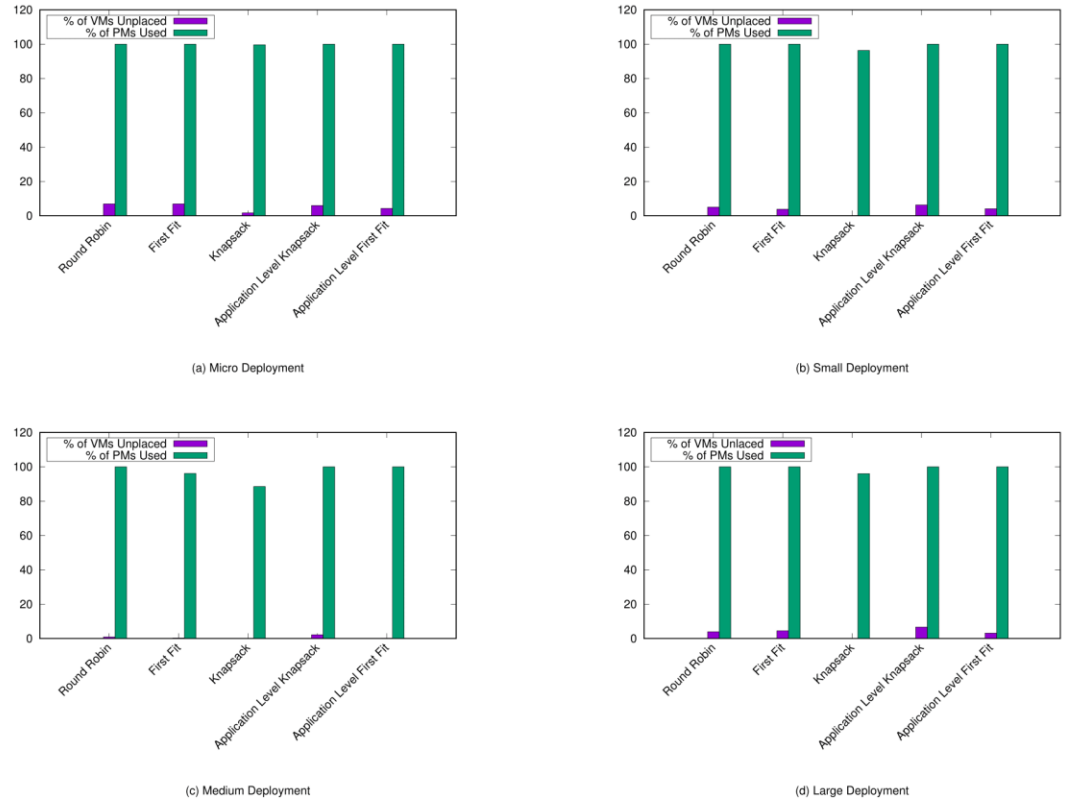


**Figure 4. Percentage of unplaced VMs and used PMs using VM placement methods on different datacenter deployment scenarios.**

## 6 CONCLUSION AND FUTURE WORK

VM placement is a challenging and complex problem in cloud datacenters. Many different optimization and constraints can be studied by addressing this problem. In this paper, we study the use of traditional implementations of addressing this problem to minimize the overall bandwidth utilization in datacenters. We extended knapsack and first fit methods to specifically address to significantly minimize the bandwidth usage in datacenters. Our experimental evaluation shows that the application level knapsack and application level first fit can help reducing bandwidth usage from 65% to 78%. However, these algorithms introduce overhead to the standard implementation of knapsack as these waste more computing resources of PMs. This trade-off of minimizing bandwidth and using more resources in the datacenter is interesting as the IaaS cloud management can identify their objectives and can use the appropriate VM placement method to achieve them. We believe our work explained in this paper will help to decide the VM placement algorithm to cloud providers. We plan to extend this work to incorporate VM migration in application level knapsack and application level first fit to reduce the overall wastage of resources. We also plan to implement both of these algorithms in OpenNebula and evaluate it on a real private cloud using different big data and multi-tier applications. We also intend to explore using Linear Programming to solve the proposed optimization problem.

## 7 REFERENCES

A. K. Paul, S. K. A., Sahoo, B., & Turuk, A. K. (2014). Application of greedy algorithms to virtual machine distribution across data centers, (pp. 1–6).

M. Abdullah, M. (2017). Application level VM placement simulation. https://github.com/abdullahsunny/VM-Placement-for-Application-centric-Data-centers-Simulation.

A. Alahmadi, Alnowiser, A., Zhu, M. M., Che, D., & Ghodous, P. (2014). Enhanced first-fit decreasing algorithm for energy-aware job scheduling in cloud. In *2014 International Conference on Computational Science and Computational Intelligence* (pp. 69–74). volume 2.

U. Bhuvan, Rosenberg, A. L., & Shenoy, P. (2007). Application placement on a cluster of servers. *International Journal of Foundations of Computer Science*, *18*, 1023–1041.

N. Bobroff, Kochut, A., & Beaty, K. (2007). Dynamic placement of virtual machines for managing sla violations. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management* (pp. 119–128).