Virtual Machine Placement optimization supporting performance SLAs

Ankit Anand*
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi, India
ankit.s.anand@gmail.com

J Lakshmi , S K Nandy
Supercomputer Education and Research Centre
Indian Institute of Science, Bangalore, India
jlakshmi@serc.iisc.in, nandy@serc.iisc.ernet.in

***Abstract*—Cloud computing model separates usage from ownership in terms of control on resource provisioning. Resources in the cloud are projected as a service and are realized using various service models like IaaS, PaaS and SaaS. In IaaS model, end users get to use a VM whose capacity they can specify but not the placement on a specific host or with which other VMs it can be co-hosted. Typically, the placement decisions happen based on the goals like minimizing the number of physical hosts to support a given set of VMs by satisfying each VMs capacity requirement. However, the role of the VMM usage to support I/O specific workloads inside a VM can make this capacity requirement incomplete. I/O workloads inside VMs require substantial VMM CPU cycles to support their performance. As a result placement algorithms need to include the VMM's usage on a per VM basis. Secondly, cloud centers encounter situations wherein change in existing VM's capacity or launching of new VMs need to be considered during different placement intervals. Usually, this change is handled by migrating existing VMs to meet the goal of optimal placement. We argue that VM migration is not a trivial task and does include loss of performance during migration. We quantify this migration overhead based on the VM's workload type and include the same in placement problem. One of the goals of the placement algorithm is to reduce the VM's migration prospects, thereby reducing chances of performance loss during migration. This paper evaluates the existing ILP and First Fit Decreasing (FFD) algorithms to consider these constraints to arrive at placement decisions. We observe that ILP algorithm yields optimal results but needs long computing time even with parallel version. However, FFD heuristics are much faster and scalable algorithms that generate a sub-optimal solution, as compared to ILP, but in time-scales that are useful in real-time decision making. We also observe that including VM migration overheads in the placement algorithm results in a marginal increase in the number of physical hosts but a significant, of about 84 percent reduction in VM migration.**

Keywords*—*Virtual machine Placement,Service Level Agreements, Vector Packing problem, Virtual Machine Migration

I. Introduction

Today, most of the enterprise workloads are either dynamic where resource requirements vary continuously over time of the day or are resource centric which utilise one resource more than the other resources (CPU intensive v/s I/O intensive applications). In such scenarios, there is overall low utilization of physical resources if applications are hosted on dedicated servers. So, most of the enterprises are shifting towards cloud based solutions [1] where these applications are hosted in Virtual Machines(VMs). Virtualization is a key enabler of cloud computing where a number of VMs share a physical resource. It not only provides software isolation to these applications but also leads to overall better utilization of server resources.

One of the important characteristics of this Cloud Computing model is abstraction of resources at the user level. This abstraction leads to separation of usage of resources from ownership, management and control of resources. While an end user can demand a fixed capacity of resources for a VM, he has no control on which specific host this VM would be located or what kind of co-resident VMs would be with this VM. A typical aim in undertaking resource provisioning at cloud provider's end is to maximize resource utilization by placing all VMs over a minimal set of physical hosts while meeting capacity requirement for a VM. The capacity requirement for a particular VM is specified by a user in terms of CPU cycles, memory, disk and network bandwidth used by application. But as discussed by Menon et. al. [2] and others [3] and [4], when applications are executing in virtualized scenarios, virtualization overheads manifest as extra CPU usage by hypervisor (or virtual machine monitor [5]) and this extra usage depends on virtualization technology used (paravirtualization or full virtualization). This extra CPU requirement by hypervisor, which is substantial in case of I/O applications, makes the actual capacity requirement incomplete. Not considering this overhead in placement decisions leads to bottleneck situations that are currently resolved using VM migration which further contribute to loss in application performance. Hence, to honour performance Service Level Agreements (SLAs), there is a need to capture this VMM or hypervisor CPU overhead explicitly along-with the VM's resource requirements in the placement problem.

Also, the Virtual Machine Placement Problem (VMPP) is dynamic in nature i.e after the initial placement, not only new VMs with varying resource requirements arrive and existing VMs leave but also the resource requirements for VMs change continuously. So, after doing initial placement, scheduling decisions need to be taken periodically at each scheduling cycle. And VMPP algorithms need to be invoked at each scheduling cycle catering to these varying requirements of workloads. The existing technologies handle these elastic needs by migration of VMs from one host to other. Also, the VM migration is proposed as a solution to meet SLAs in case of changing resource requirements [6]. However, migration in itself is a costly operation. It not only degrades the performance of

*The work was done while the author was at Indian Institute of Science, Bangalore

applications for a certain time (during migration) but also introduces additional network and other overheads while migrating [7]. This migration overhead depends on type of application and resources used by it, for example, applications having more memory and I/O footprint would have higher migration overheads compared to pure CPU intensive applications. To minimize the number of migrations and loss in performance due to the same, the provisioning algorithms must be made migration aware by considering the previous placement of VMs and aim for reducing both the number of migrations and the number of hosts used.

There have been attempts in the past to include this migration overhead in placement decisions but how this migration overhead should be quantified in terms of all resources still needs to be explored. This paper attempts to address that problem by weighted summation of resource requirements. Also, most of papers consider placement by considering CPU and memory as resources while ignoring disk and network requirements which can be significant in I/O applications. Here, we discuss algorithms in general framework where all resources are considered.

We propose that these two factors of extra CPU overhead and migration overhead (quantified in terms of resource requirements) should be included in VM placement problem so that scheduling decisions have minimum effect on performance of applications along with main aim of minimizing the number of hosts. We evaluate the above modifications in both traditional Integer Linear Programming (ILP) approach and FFD heuristics. We observe that ILP solvers yield optimal solution but need long computing time even with a parallel shared memory version. However, FFD heuristics are much faster and scalable which yield sub-optimal solutions which are good enough to be used in practice and in time scales so as to aid real time decision making. Also, we compare different FFD heuristics and provide reasoning, why one yields better results than others. We observe that, making the placement algorithms migration aware in all these heuristics leads to less than 2 percent increase in number of hosts but reduces the number of migrations by more than 84 percent.

The rest of the paper is organized as follows: Section II presents the related work. Section III presents performance SLAs and VM placement issues with typical use-case results for I/O workloads. Section IV gives the integer programming formulation with virtualization and migration overhead considered and section V compares the different heuristics. Section VI shows the experimental setup and results. Finally, in section VII, we conclude with a discussion on future work.

II. Related Work

The problem of server consolidation is well explored in literature. It is a classical problem of bin packing where different sized items are to be packed in bins of fixed capacity and one aims to minimize the number of bins used. Here, the problem translates into a vector packing problem where VMs are items to be inserted, physical machines are bins and resources like CPU, memory, bandwidth etc. form different dimensions of vector.

The problem is posed classically as an Integer Linear Programming problem where objective function is to reduce the number of physical hosts used and hosts' capacities and other restrictions are posed as constraints for the same. In [8], Gupta et. al. have given a two stage heuristic algorithm for solving the server consolidation problem with item-item (two conflicting VMs cannot be placed together) and bin-item (a VM cannot be placed on a particular host) incompatibility constraints. Agarwal et. al. in [9] deal with the same problem by an approach of group genetic algorithm and shows how a genetic algorithm can prove to be better than other approaches. But it has been observed that beyond a few hundred VMs, genetic algorithms and ILP based approaches have very high time complexity in worst cases even with parallel versions. Hence, these are not seen as potential solutions in real world scenarios affecting scheduling decisions.

To reduce time complexity, algorithms based on greedy heuristics like First Fit, Best Fit etc. have been implemented. These give close to optimal solutions in one dimensional case [10]. Here, items are arranged according to sizes and then, packed into bins based on algorithm used. In case of multi-dimensional problems, it is not clear what combination of vector should be used to generate a scalar(size) so that one-dimensional algorithms can be used. Panigrahy et. al. [11] in their report on heuristics for vector bin packing explore different combinations of vectors to generate the scalar(size) and propose the new concept of geometric heuristics. But it does not justify why one heuristic outperforms others and given a workload, which heuristic should be applied.

Another way of transforming a vector to scalar was proposed by Woods et. al. in automated monitoring and provisioning system - Sandpiper [12]. They primarily use a volume metric for detecting hotspots and enable VM migration. The shortcomings of the same are shown in a recent work by Mishra et. al. [13]. This work explains shortcomings in existing technologies and also presents a new approach based on vector algebra for VM placement. We evaluated their approach of resource imbalance vectors and observe that the same is outperformed by Euclidean distance/Norm-2 approach in many cases, giving specific counter examples for the same.

Recently, Wu et al. in [14] have given a simulated annealing based algorithm to tackle this problem where initial solution is obtained by applying FFD based heuristics and then with some compromise over time, significant improvements can be obtained. A recent paper by Lee et. al. of Topology Aware Resource Allocation (TARA) [15], brings out a new dimension where by network topology related information and application's requirements are used to allocate data intensive workloads. They do so by a naive genetic approach that is different than discussed in [9].

A few recent works have also explored the migration overhead problem. The problem of migration control is described in [16]. They put a constraint where a static workload would be never migrated. In this case, one may end up getting less packing efficiency. We point out that such a migration control should be dependent on type of workload and migration cost involved for migrating that VM.

Another paper by Schaffrath et. al. [17] deals with the migration problem in CloudNets and deals with migration overhead for ILP settings specifically for migration of Virtual Networks. We deal with the same problem in general in Clouds

and propose a quantification of migration overhead along with hypervisor CPU usage included in the VM's resource vector, in both FFD heuristics and ILP.

A paper by Hermenier et. al. [18] uses a concept of entropy for minimizing the number of hosts used. It first chooses all solutions with minimum number of hosts and then selects the one with minimum migration overhead. Also, it focuses on only two resources CPU and memory using a constraint propagation model, how the same has to be extended to other resources and in FFD heuristics is not clear from the same.

Also, Verma et. al. have discussed a tool pMapper [19] where they do consider migration overhead while doing placement. They model and base migration cost based on decrease in throughput of application and hence loss of revenue due to migration is considered for migration cost. We quantify the migration overhead as weighted sum of resource requirements of application which are more closely related to VM placement decisions. Also, our approach does not require migration experiments repeatedly to calculate migration cost which may change with changing resource requirements of applications.

There are many prevalent datacenter management tools like OpenNebula, Eucalyptus etc.. They have diverse strategies for VM placement. For instance, the OpenNebula [1] implements a rank scheduling policy for VMs where by hosts are ranked using a formula on available monitoring information. The pre-defined policies exist for both Packing(Minimum number of hosts)and load balancing. The metrics used by these are number of running VMs or Free CPU available on that host. These are coarse grained and high level metrics because the type and resource requirements are more important than merely the number of running VMs. Also, a decision should not be taken on a single resource like Free CPU. OpenNebula also provides an option to user to define its own ranking policy. But it is not possible to obtain SLA performance guarantees for the applications without using complete resource vector. On the other hand, Eucalyptus follows first fit or next fit based algorithms for VM provisioning on hosts.

To the best of authors' knowledge, none of these solutions consider the virtualization overhead and workload specific migration awareness in a general framework of VMPP. The contributions of this paper in this context are as follows:
1. Including architecture specific virtualization overhead in problem definition to honour performance SLAs.
2. Considering previous placements of VMs and quantification of migration overhead based on type of workload.
3. Placement using resource vector including VMM CPU cycles along-with the VM's resource requirement, rather than just CPU or Memory.

III. Performance SLAs and VM Placement issues

In this section, firstly, we present a case study showing the impact of virtualization overhead on performance SLAs of applications. As discussed in [20] and [3], in case of I/O applications virtualization overhead is manifested as extra CPU usage by the hypervisor apart from usual guest usage. We show that not considering this overhead in problem domain leads to severe degradation of applications' performance and therefore SLA violations.

We present two different scenarios, in Case-1 virtualization overhead is considered in resource provisioning while in Case-2, no overhead is considered. We use KVM [2] as a hypervisor for our experiments. Since in KVM each VM is treated as a separate process, we take sum of CPU used by guest and CPU usage of hypervisor as total CPU requirement in Case 1. Only the guest CPU is considered in Case-2 where no virtualization overhead is considered. The other resources considered are memory, disk I/O bandwidth and network bandwidth.

In order to validate above scenarios, we first show the difference in resource requirements in two cases by analysing resources used in case of web-server application. We hosted apache web-server application in virtualized and non-virtualized environments. The machine specifications for the two cases are shown in Table I. To simulate workloads, we used httperf which is commonly used benchmarking tool for generating representative workloads. Figure 1 plots the total CPU used for different request rates for the two cases discussed above. We analyze that the CPU requirement in virtualized case is significantly high compared to non-virtualized scenario. Further breaking down the CPU requirement in different modules in figure 2, we observe that extra CPU requirement is due to CPU used by different modules in hypervisor in case of virtualized environment.

TABLE I: Machine Specifications

Hw-Sw	Physical Machine	Virtual Machine
Processor	AMD 2.4 Ghz 12 cores	AMD 2.4GHz (Req. based)
Memory	16GB	Req. based
OS	OpenSuse 12.1	Opensuse 11.4
Network B/W	1Gbps	Best Effort
Disk I/O B/W	7000 KB/s	Best Effort

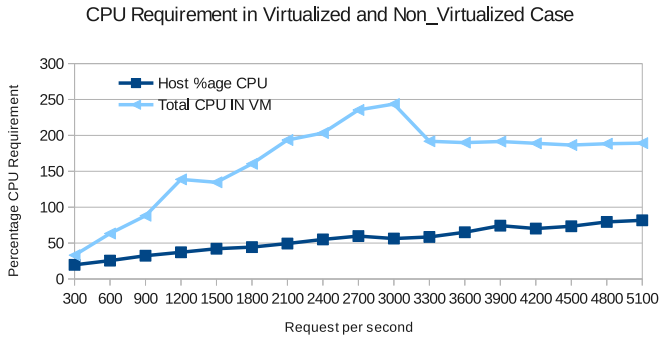

Fig. 1: CPU Usage in Virtualized and Non-Virtualized Settings

Next, we analyze the impact of not considering this CPU overhead in scheduling decisions. For our experiments, we hosted 5 VMs with four different applications as shown in Table II. The table also shows the metric for performance evaluation, how the workload is generated and the CPU requirement for each application. The other resources like memory, Network I/O and Disk I/O do not effect the scheduling decision in this case and their actual requirements are

[1] http://opennebula.org/documentation:archives:rel3.2:schg

[2] http://www.linux-kvm.org/

TABLE II: Types of applications

Type of appl. (Workload intensity)	Guest CPU (core)	Hypervisor CPU (core)	Performance Metric	Load generation method
web-server (3000 rq/s)	1 CPU	2 CPUs	response time (ms)	httperf [21]
smtp mail-server(6000 req/s)	0.5 CPU	0.5 CPU	delivery time of mail(s)	smtp-source [3]
Prime numbers (Till 100000)	8 CPUs	0 CPU	Task completion time(s)	sysbench
Matrix Multiplication-I(1024*1024)	1 CPU	0 CPU	Task completion time(s)	size of matrix
Matrix Multiplication-II(1024*1024)	-do-	-do-	-do-	-do-

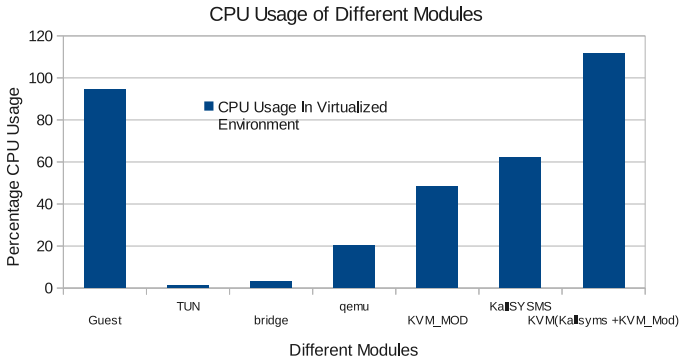

Fig. 2: CPU Usage Breakdown for Saturation Points

shown in Table IV and Table V . We observe from figure 1 that each web-server VM requires only 1 CPU (less than 100 percent, assuming 1 Core per 100 percent) if no overhead is considered while there is requirement of 3 CPUs in virtualized environment. The scheduler aims on consolidating workloads to minimum number of physical machines and packs all VMs on a single host in no overhead case, while it needs two hosts in other case. We analyze the effect on performance of web-server application in two Cases. Figure 3, plots request rate with response time which is most common metric for defining SLAs in case of web-servers. Since only 1 CPU is given to web-server VM while not considering virtualization overhead, we observe that response time begins to rise sharply at around 1000 requests/second. This is in contrast to other scenario where we can achieve reasonable response time even at 3000 requests per second. Hence, if SLA is such that there should be less than 100 ms response time for request rates till 3000 req/s, we would be achieving the SLAs for only 30 percent of requested rate.

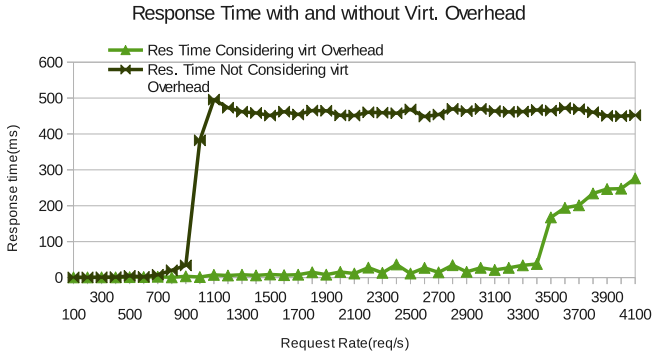

Fig. 3: Virtualization impact on application performance

Hence, this study clearly points out that while moving applications, particularly I/O workloads, to the cloud, we must consider the resource requirements in virtualized settings which are significantly high compared to host usage and effect performance SLAs of application drastically. This extra CPU is CPU used by hypervisor and can be monitored only through host.

Quantification of Migration Overhead: On the other hand, owing to dynamic nature of problem and to cater to elastic workloads, VMPP algorithm needs to be invoked at every scheduling cycle. The new solution in order to minimize the number of hosts may lead to migration of VMs from one host to another which itself is expensive process and incurs additional overheads.

We propose the quantification of this migration overhead based on the type of workload in terms of its resource requirements. We argue that this overhead is more for applications having high I/O (network and Disk) and memory footprint compared to pure CPU intensive applications. For instance a web server (Network B/W Intensive) or Disk I/O application would have more effect on performance during migration. This argument stems from the fact that in case of applications that have network or disk associated accesses, apart from transfer of memory, the connections causing changes to the memory content need to be quiesced before migration can complete. Apart from this, since most of these applications are live and communicating directly with user, these should have minimum downtime [7] and given least priority in migration. Secondly, the applications having high memory footprint need to transfer whole memory stack from one machine to another and have high migration time and effect on network resources compared to ones having low memory footprint. Therefore, we give high weights to Network and Disk requiremnts of applications compared to CPU resources. We quantify the total migration overhead(α) by weighted addition of different resource requirements (r_i) for each VM as shown in equation 1. The relative weights(w_i) used are 0.8 for network bandwidth resources, 0.6 for disk I/O resources, 0.4 for memory and 0.1 for CPU resource. Also, resource requirements are normalized to 1 with respect to physical machine's capacity.

$$\alpha = \sum_{i=1}^{i=d} r_i * w_i \tag{1}$$

where, d is number of resources to be considered.

The actual values of these weights have to be arrived experimentally by undertaking extensive migration tests, which is left for future work but the relative ratio for the same should remain the same as discussed above. We modify the VMPP by including this migration overhead and then, minimizing both the number of hosts and the migration overhead. The actual problem definition and algorithms to solve the same are described in next two sections.

TABLE III: Notations

$V_{i,j}$	binary variable,1 if VM j is placed on host i
m	No of Hosts
P_i	binary Variable,1 if host i is used for any VM
n	No of Previously hosted VMs
α_j	Migration Coefficient of VM j
t	No of new VMs to be allocated
R_i	Capacity of R resource in Host i
r_j	Requirement of r resource for VM j
$V_{i,j}^{prev}$	values of $V_{i,j}$ according to previous allocations
$cpu_j^{Hypervisor}$	CPU used by hypervisor for VM j
cpu_j^{VM}	CPU used by VM j

IV. Integer Programming Formulation

The problem of VM placement is formulated traditionally in the form of an ILP. Here, in this section, we give an ILP formulation of VMPP along with modifications to address issues of migration and virtualization overhead discussed in previous sections. The problem definition is shown below along with description of notations in Table III.

Minimize:

$$F_{obj} = \sum_{i=1}^{i=m} P_i - \sum_{i=1}^{i=m} \sum_{j=1}^{j=n} \alpha_j * V_{ij}^{prev} * V_{i,j} \quad (2)$$

Constraints:

$$\sum_{i=1}^{i=m} V_{ij} = 1 \; \forall j \quad (3)$$

$$P_i * CPU_i \geq \sum_{j=1}^{j=n+t} V_{ij} * \left(cpu_j^{Hypervisor} + cpu_j^{vm} \right) \; \forall i \quad (4)$$

$$P_i * R_i \geq \sum_{j=1}^{j=n+t} V_{ij} * r_j \; \forall i \quad (5)$$

Here, the objective function (eqn. 2) of problem consists of two parts, the first part focuses on minimizing the number of hosts while the second part relates to minimum migration overhead. Here, α_j is calculated by weighted addition of resource requirements as described in previous section. So, eqn. 2 would minimize the objective function if both $V_{i,j}^{prev}$ and $V_{i,j}$ are one. Hence, objective function evaluates the trade off between using minimum number of hosts and migration overhead. If two allocations have same number of hosts used, the objective function would choose an allocation with minimum migration overhead. Here, one may point out that how good is to compare the number of hosts with our proposed migration overhead but the point to note is that weights for each resource is user defined parameter and can be adjusted maintaining the relative ratio between different resource requirements same. This trade off between extra host used and migration overhead can be adjusted by varying weights parameters to suit cloud provider needs.

In constraint set, eqn. (3) describes the fact that a VM is allocated to only one host. Eqn. (4) shows the second modification to the problem, where virtualization overhead is considered. We have added CPU used by hypervisor separately and CPU used by VM in this case. This CPU used by hypervisor can be monitored in the host by using Linux Perf. Finally, eqns. (5) puts the capacity constraints on different resources. Also, it is assumed that maximum disk I/O bandwidth and other resource capacities are not effected much in virtualized settings and if they differ significantly, the values should be considered in virtualized settings as well. We used lpsolve [4] package for solving the exact ILP in serial form which uses branch and bound method for solving the same. To further reduce time in big instances, we used scip multi-threaded package [5] and ran the algorithm upto 12 pthreads. According to a classical result by Lensra [22], any ILP with n variables and m constraints can be decided in $O(c^{n^3} - m^d)$ time. In our case, for p virtual machines and q physical hosts, we have $(p * q + q)$ variables and $(p + 4q)$ constraints which gives a highly exponential time. The results for the ILP formulation for small problem instances is shown in section VI . The next section captures the same problem definition in different FFD heuristics to reduce the exponential time.

V. FFD vector heuristics

The above Integer Linear Program provides an easy formulation of the problem and all the constraints and requirements are expressed directly capturing the problem domain naturally. But solving ILP is an NP hard problem and so can't be used in practice for real time solutions. However, we can do some modifications to the same by having a linear programming relaxation of the original ILP. These may give some early results for small cases but as problem size grows, these algorithms are not suitable for obtaining near optimal solutions within the scheduling cycle [11]. Therefore, FFD heuristics might be a better trade off.

The theoretical bounds of 11/9 of optimal number of bins are shown for one dimensional case in [10] for these heuristics. In average cases, FFD performs reasonably well and that too in small duration of time such that solving a new problem with large number of VMs in every scheduling cycle is feasible.

The first problem in this case is how to incorporate the notion of a vector in FFD. Different heuristics for the same were suggested in [11] and [13]. We choose 3 different FFD based methods viz. Dot Product, Euclidean Distance and Resource Imbalance Vector method as suggested in the literature and then modify those to our problem settings. Firstly, we analyse these methods and explain what these methods are capturing in different scenarios. The three strategies differ in the ordering of VM placement in each case. Before proceeding to strategies, we define some acronyms, RRV is Resource Requirement Vector defined for all VMs, RUV is Resource Utilization Vector defining currently used resources of a host and RCV is Residual Capacity Vector defining remaining capacity for all used hosts.

In Dot Product approach, we take the unit vector corresponding to RCV of currently used host and take its dot product with unit vector of all unallocated valid VMs (VMs which satisfy resource capacity constraints) and the VM having maximum dot product is chosen for allocation. The intuition behind the same is that a large Dot product value means small

[4] http://lpsolve.sourceforge.net/5.5/

[5] http://scip.zib.de/

angle between the RCV of host and RRV of VM. This in turn means better alignment of VM vector with the host vector, thus, making VM a good choice to be placed on that host. The issue with this approach is that although it captures the direction sense pretty well, it does not capture the remaining capacity in absolute sense.

The second approach of Resource Imbalance Vector (RIV) was suggested by Mishra et. al. in [13]. They take the projection of RUV on normalized resource diagonal i.e (1,1,1..1) vector and subtract it from RUV to get RIV of that physical machine(fig 4). Similarly, we get RIV for every VM. Then, by adding these two RIVs, we choose the one with minimum magnitude. The intuition behind this is that complementary workloads across diagonal will be placed on same machine. This leads to better host utilization. We argue that in a number of cases, there would be many such vectors which would balance the RIVs exactly but not all are good candidates for placement. As shown in figure 4, all the vectors a,b,c,d of the VMs give the same total RIV but only c is the candidate which actually captures the remaining capacity exactly and should be chosen for placement.

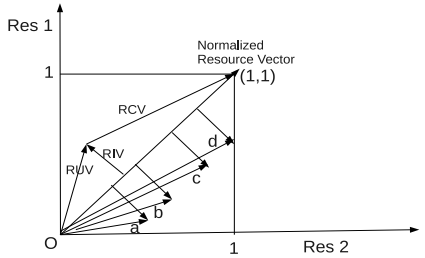

Fig. 4: Resource vectors in 2-D plane

Thirdly, we look at the euclidean distance strategy where we will calculate euclidean distance between RCV of PM and RRV of VMs and choose the VM having minimum Euclidean distance with the host. Euclidean distance method inspite of its simplicity captures both the direction and magnitude aspects correctly. It is able to capture the direction sense of a vector because with increase in angle between two vectors, the euclidean distance also increases. Also, most of the workloads in real world are specific resource centric i.e they use one resource more than the others. So, the solution should not only capture its alignment with host capacity as done in previous two approaches but a good fit overall (across most of dimensions).

Our basic strategy is underlined in Algorithm 1. First, we normalize all capacities and resource requirements in all dimensions to 1. We initialize a host and to place the next VM, we consider all the valid VMs (which meet resource requirements) and find the most appropriate VM according to different heuristics based on what we call 'the score'. The key contribution of our algorithm is that we change (increase in Dot product or decrease in RIV/Euclidean distance) the score of a VM if the VM is getting placed on the previously allocated host. This increases the chances of a VM being placed on the same host again compared to other VMs. This change of score is quantified in terms of migration overhead α_j as described in previous section. The issue of meeting performance SLAs is dealt in this case, by adding CPU usage of hypervisor as described in section III . Analysing our algorithm's complexity, in the worst case, we select 1 VM in every pass and in total, we make number of passes equal to number of VMs, so to place n VMs ,we will take $O(n^2)$ time which is significantly less compared to exponential time of ILP. The calculation of score is a constant in all algorithms whose value depends on heuristic used.

```
for All dimensions do
    Normalize all capacities and requirements to 1;
end
Initialize a host;
while All VMs Not Placed do
    Initialize pointer to start of VM list;
    while Pointer not At the end of VM list do
        if VM can be packed into current Host then
            calculate score by RIV or Dot product or
            Euclidean distance ;
            if VM placed on same host in Previous
            Mapping then
                Update score of each VM by adding
                migration overhead based on heuristic,;
            end
        end
        Goto next VM in the list ;
    end
    if No VM can be placed but remaining unplaced
    VMs then
        Initialize and add a new host;
        Goto statement 4 ;
    end
    Find VM with minimum score in RIV and
    Euclidean distance and maximum score in Dot
    product;
    Place that VM over host and remove it from VM
    list;
    Update host's current capacity;
end
```

Algorithm 1: FFD previous mapping aware heuristic

Also, since all of these are heuristic solutions for the problem, the counter examples for all of these can be found. But the choice of heuristic should depend on the nature of workload and problem context. In the next section, we state our experimental setup and results.

VI. Experimental Setup and Results

In this section, we evaluate different heuristics and ILP in a setup consisting of four different types of workloads as described in Table II. In our work, we consider four dimensions of resources, Total CPU usage of VM and hypervisor, memory requirements, disk I/O bandwidth and network bandwidth. We assume sufficient storage (30 GB) for all applications although same can be added as an extra dimension without any modification.

Simulation Strategy To select a good mix of workloads, we choose a uniform random distribution across these four types of workloads means expectedly we will get equal number of VMs of each type. Then for each type of application (VM) i.e webserver, mail-server etc., we generate different request rates and monitor resource requirements (shown in Table IV and Table V) for the same by experiments such that performance SLAs are met. For instance, the performance SLA for webserver VMs is response time less than 100ms.

For our experiments, we choose ten different request rates for web-servers and ten different delivery rates for mail servers and use a normal distribution over the same to emulate real world scenario. Similarly, five different CPU workloads and four different memory intensive workloads are chosen. The request rates for each type of VM are normally distributed and we refer Table IV and V to get actual resource requirements corresponding to a particular request rate. The same resource requirement would be given by Resource Monitoring Unit for a running VM in each scheduling cycle in real world scenario. The normal distribution ensures that we have less number of VMs having very high or very low request rate but a significant number having moderate request rates in each and every case of web-server , mail-server, CPU based load etc.

The reason for choosing these kind of workloads is these are most commonly used in clouds and each of them uses a different resource, thus providing a good mix of workloads. Also, these are similar to SPECvirt [6] workloads for benchmarking virtual servers.

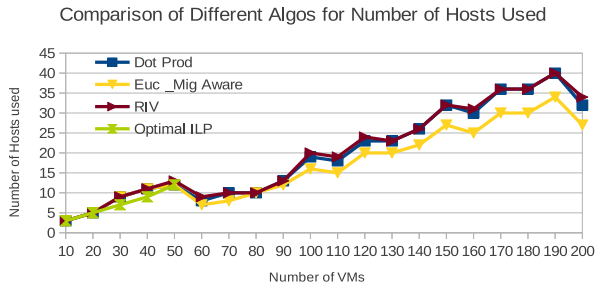

Fig. 5: Number of hosts for FFD heuristics

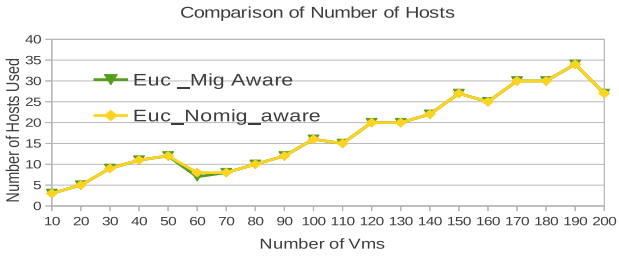

Fig. 6: Number of Hosts with and w/o Migration

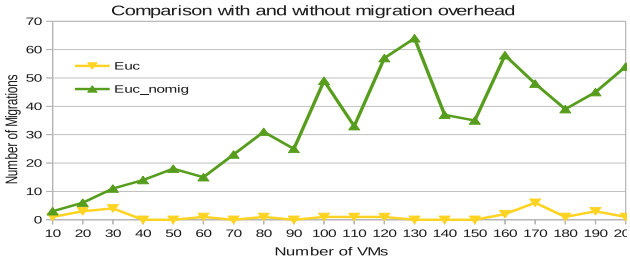

Fig. 7: Number of Migrations for Euclidean Strategy

We tested the above three algorithms, varying the total number of VMs from 10 to 200. The results presented here, are simulation results. This is due to lack of access to a large cloud data center. However, the resource requirements of each VM are derived experimentally for individual instance. In figure 5, the number of hosts needed to place all VMs is plotted against number of VMs. Due to exponential running time, ILP based optimal solutions are available till only 60 VMs beyond which it was taking many hours even with twelve parallel threads. As we see, all the heuristics are near optimal for small instances, but for bigger problem instances Euclidean distance strategy starts to outperform others by around 10 percent. Also, we show number of hosts used both with and without migration overheads (for euclidean distance) in figure 6. We observe less than 2 percent increase in number of hosts when migration overheads are considered. This points out that multiple solutions are available in most of the cases and by making our algorithm migration aware, we are able to choose those solutions which take minimum migration overhead. In figure 7, we plot number of migrated VMs against number of VMs with and without consideration of migration overheads for euclidean distance strategy. We observe that our strategy reduces the migrations by more than 84 percent in almost all cases.

From above results, we clearly observe that we are able to meet SLA performance guarantees by including overhead aspect in problem definition. Also, by modifying problem statement, we are making problem aware of previous assignments. Hence, we get a mapping with minimum migration overhead without increasing the number of hosts used. This approach ensures performance SLAs are met where actual requirements are taken care of by considering virtualization overhead and there is minimal loss in performance due to migrations. In the next section, we conclude the discussion along with directions of our future work.

VII. Conclusion and Future Work

In this work, we enumerate modifications to VMPP to include VM specific virtualization overhead and VM migration overheads based on resource requirement vector so as to ensure performance SLA guarantees for the applications hosted inside the VM. We show that this approach significantly reduces the number of migrations along with obtaining a solution near to optimal number of hosts. We also compare various heuristics and show that Euclidean distance approach captures the resource vector quite well. In future, we would like to test these algorithms in real world elastic scenarios and compare the same with existing algorithms. We would also like to evaluate effect of other architectural considerations like cache interference while taking placement decisions. Also, we would like to explore and define a formal Packing metric in this multi-dimensional resource case to evaluate fragmentation effects in different algorithms.

[6]http://www.spec.org/

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672

[2] A. Menon, J. R. Santos, Y. Turner, G. J. Janakiraman, and W. Zwaenepoel, "Diagnosing performance overheads in the xen virtual machine environment," in *Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, ser. VEE '05. New York, NY, USA: ACM, 2005, pp. 13–23. [Online]. Available: http://doi.acm.org/10.1145/1064979.1064984

[3] M. Dhingra, J. Lakshmi, and S. Nandy, "Resource usage monitoring in clouds," in *Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on*, sept. 2012, pp. 184 –191.

[4] J. Lakshmi, "System virtualization in the multi-core era, a qos perspective," Dissertation, Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore, 2010.

TABLE IV: Resource requirements for web-server having response time less than 100 milli-seconds

Request rate	Disk I/O	Memory(GB)	Network B/W(Mbps)	Hypervisor's CPU(percent)	VM CPU(percent)
100	0.3552	0.45038	10.3959	9.69	4.96
200	85.8906	0.45023	20.7628	17.19	11.1
300	7.20042	0.44788	31.1061	23.56	11.71
400	11.3644	0.44781	41.5110	28.43	14.33
500	33.0323	0.44857	51.7151	30.34	16.8
600	30.4944	0.45174	61.9890	26.11	17.6
700	46.1489	0.47109	76.8173	27.54	21.02
800	74.372	0.53614	83.9530	30.02	21.94
900	71.3733	0.54164	88.2553	30.17	22.62

TABLE V: Resource requirements for mail-server having delivery time less than 5 seconds

Delivery rate	Disk I/O	Memory(GB)	Network B/W (Mbps)	Hypervisor's CPU(percent)	VM CPU(percent)
600	648.052	0.5079	0.3710	2.08	7.73
1200	1373.38	0.5558	0.7496	3.84	5.96
1800	1944.1	0.60919	1.1145	5.23	9.16
2400	2472.13	0.63842	1.4393	7.55	13.04
3000	3685.53	0.63096	1.8411	12.54	16.98
3600	3685.68	0.6771	2.2419	10.68	16.17
4200	4223.03	0.69432	2.5875	12.19	19.96
4800	4542.14	0.69722	2.7729	14.29	22.64
5400	5031.67	0.72077	2.9412	15.09	28.53
6000	5674.32	0.71144	3.6031	16.74	25.73

[5] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," *Commun. ACM*, vol. 17, no. 7, pp. 412–421, Jul. 1974. [Online]. Available: http://doi.acm.org/10.1145/361011.361073

[6] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, 2007, pp. 119–128.

[7] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *Communications Magazine, IEEE*, vol. 50, no. 9, pp. 34–40, 2012.

[8] R. Gupta, S. Bose, S. Sundarrajan, M. Chebiyam, and A. Chakrabarti, "A two stage heuristic algorithm for solving the server consolidation problem with item-item and bin-item incompatibility constraints," in *Services Computing, 2008. SCC '08. IEEE International Conference on*, vol. 2, july 2008, pp. 39 –46.

[9] S. Agrawal, S. K. Bose, and S. Sundarrajan, "Grouping genetic algorithm for solving the serverconsolidation problem with conflicts," in *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, ser. GEC '09. New York, NY, USA: ACM, 2009, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/1543834.1543836

[10] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson, "Approximation algorithms for np-hard problems," D. S. Hochbaum, Ed. Boston, MA, USA: PWS Publishing Co., 1997, ch. Approximation algorithms for bin packing: a survey, pp. 46–93. [Online]. Available: http://dl.acm.org/citation.cfm?id=241938.241940

[11] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing," Technical report,Microsoft Research, Tech. Rep., 2011.

[12] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proceedings of the 4th USENIX conference on Networked systems design & implementation*, ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 17–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973430.1973447

[13] M. Mishra and A. Sahoo, "On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, 2011, pp. 275–282.

[14] Y. Wu, M. Tang, and W. Fraser, "A simulated annealing algorithm for energy efficient virtual machine placement," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, 2012, pp. 1245–1250.

[15] G. Lee, N. Tolia, P. Ranganathan, and R. H. Katz, "Topology-aware resource allocation for data-intensive workloads," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 120–124, Jan. 2011. [Online]. Available: http://doi.acm.org/10.1145/1925861.1925881

[16] T. C. Ferreto, M. A. Netto, R. N. Calheiros, and C. A. De Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1027–1034, 2011.

[17] G. Schaffrath, S. Schmid, and A. Feldmann, "Optimizing long-lived cloudnets with migrations," in *Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on*, 2012, pp. 99–106.

[18] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: a consolidation manager for clusters," in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, ser. VEE '09. New York, NY, USA: ACM, 2009, pp. 41–50. [Online]. Available: http://doi.acm.org/10.1145/1508293.1508300

[19] A. Verma, P. Ahuja, and A. Neogi, "pmapper: power and migration cost aware application placement in virtualized systems," in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, ser. Middleware '08. New York, NY, USA: Springer-Verlag New York, Inc., 2008, pp. 243–264. [Online]. Available: http://dl.acm.org/citation.cfm?id=1496950.1496966

[20] A. Anand, M. Dhingra, J. Lakshmi, and S. Nandy, "Resource usage monitoring for kvm based virtual machines," in *Advanced Computing and Communications (ADCOM), 2012 18th Annual International Conference on*, 2012, pp. 66–70.

[21] D. Mosberger and T. Jin, "httperf-a tool for measuring web server performance," *SIGMETRICS Perform. Eval. Rev.*, vol. 26, no. 3, pp. 31–37, Dec. 1998. [Online]. Available: http://doi.acm.org/10.1145/306225.306235

[22] S. Kratsch, "On polynomial kernels for sparse integer linear programs," *CoRR*, vol. abs/1302.3494, 2013.