2012 IEEE Fifth International Conference on Cloud Computing

Risk Aware Provisioning and Resource Aggregation based Consolidation of Virtual Machines

Kishaloy Halder, Umesh Bellur and Purushottam Kulkarni
Department of Computer Science and Engineering
Indian Institute of Technology Bombay, Mumbai
e-mail: {kishaloy, umesh, puru}@cse.iitb.ac.in

***Abstract*—Server consolidation has emerged as an important technique to save on energy costs in virtualized data centers. The issue of instantiation of a given set of Virtual Machines (VMs) on a set of Physical Machines (PMs) can be thought of as consisting of a provisioning step where we determine the amount of resources to be allocated to a VM and a placement step which decides which VMs can be placed together on a physical machines thereby allocating VMs to PMs. In this paper, we introduce a provisioning scheme which takes into account acceptable intensity of violation of provisioned resources. In addition we identify a serious shortcoming of existing placement schemes that correct in our correlation aware placement scheme. We consider correlation among aggregated resource demands of VMs while finding the VM-PM mapping. Experimental results reveal that our approach leads to a significant amount of reduction in the number of servers (up to 32% in our settings) required to host 1000 VMs and thus enables us to turn off unnecessary servers. It achieves this by packing VMs more tightly by correlating resource requirements across the entire set of VMs to be placed. We present a comprehensive set of experimental results comparing our scheme with the existing provisioning and placement schemes.**

***Keywords*-consolidation; acceptable intensity of violation; provisioning; correlation;**

I. Introduction

Data-center provisioning is increasingly leaning towards virtualization-based server consolidation techniques. Several applications, hosted in VMs, display time-varying resource requirements. Server consolidation aims at co-hosting VMs and multiplexing resource requirements while meeting performance requirement of individual applications—increasing resource and power-usage efficiency. Consolidation techniques can be *static*— as part of the initial placement plan when a set of VMs is hosted in the data center or *dynamic*—a continuous/periodic process adjusting consolidation levels to workload variations. As part of this work, we focus on the problem of static server consolidation—what is the initial placement of VMs yielding large resource and power usage efficiency?

Central to the problem of static server consolidation are the problems of: (i) *Resource provisioning*, amount of resources to be allocated to each VM, and (ii) *Placement planning*— on which physical machine should each VM be placed?

Existing methodologies commonly evaluate the goodness of resource allocation using the metric of violation count—number of time the resource demand exceeded provisioned capacity. Studies have shown that it is also crucial to consider the extent of violation [12]. Shortfall in the amount of resources to the total amount of resources required has adverse effects on the application behavior. For example, in case of batch processing tasks (like parallel Map-Reduce jobs [6]), shortfall of resources at any point of time leads to additional resource demand in the future; in case of web-request serving scenario shortfall of resources may translate to number of dropped requests. As part of this work, we use the metric of total shortfall in resources and not just number of time when resources fell short to guide provisioning. We consider the amount of resources in shortfall to the total amount of resource required. In this work, we propose *risk*, an aggregate resource unavailability metric. Risk represents the total unavailable resource requirement during a period of time, as opposed to number of time this resource was unavailable. We use the risk metric to determine resource provisioning levels of each VM.

There exists a basic trade-off between the notion of packing efficiency (how tightly VMs have been packed onto physical machines) and performance degradation due to resource unavailability (multiplexed virtual machines cumulatively require more resources than available). Several efforts [13], [10], [1], [3] have studied this trade-off by exploiting the correlation of time-varying resource demands VMs. VMs with positive correlations, indicate that their time-varying resource demand patterns coincide. VMs whose peak and average resource demands do not coincide largely are negatively correlated. Negatively correlated VMs are good candidates for consolidation on a physical machine, as cumulative resource demands over a period of time are expected to be lower than co-hosting correlated VMs.

Existing correlation aware placement schemes [8], [13] take a narrow view of correlation between VMs. They either consider correlation between pairs of VMs [8] or a small set of VMs or divide VMs into correlated sets and uses these inputs for placement. These techniques statically estimate correlations factors. An iterative correlation method would update correlation factors for each VM's placement decision. Updated correlations on such VM aggregates follow changing available capacity for placement. Instead of relying on static inputs and single-shot placement methodology, we explore the effectives of an interactive placement strategy that update correlations based on placed-VM aggregate to process with the placement plan. Our evaluation of such a technique results in significant reduction (up to 32% in our settings) in the number of active servers required to host all the VMs in the dataset compared to approaches relying on static correlation estimates.

Our contribution towards exploiting extent of resource unavailability and using an interactive placement methodology are:

- We present a resource unavailability scoring model considering extent of violation.
- We propose an iterative placement algorithm to place VMs on physical machines considering correlation on aggregated resource demands.
- We present a comparative evaluation between our and other existing placement algorithms.
- We identify factors in resource requirements patterns that govern effectiveness of placement algorithms.

The rest of the paper is organized as follows. A problem statement is presented in Section 2. Related work is discussed in Section 3. Section 4 presents our resource unavailability scoring model and the placement algorithm is presented in section 5. Section 6 presents our evaluation of the placement techniques and Section 7 concludes this work.

II. Problem Statement

In our work, we assume that the resource utilization traces of a each application are known. It is also assumed that future resource demands will be similar to the historical resource demand. Though this may sound quite optimistic there are several workload forecasting methodologies in place [11], [8] which can be applied in order to estimate the future resource demands from the historical resource demands. Workload forecasting is entirely orthogonal to our problem scope. For our problem, we assume that each VMs represents an application or a component of an application. Further, the "risk" an application (or each its components) is willing to accept (it's service-level agreement) is assumed to be known. Note that deciding the exact level of tolerable risk is a separate question, in particular, translating application-level notions of SLA to resource-level requirements. We assume such client-centric SLA utility functions are processed by a resource manager and appropriate resource unavailability risk levels are estimated for each application. As part of this work, we also concern ourselves only the CPU resource as the resource dictating provisioning and placement decisions.

Given a set of VMs, a time-series of resource utilization levels for each and set of physical machines (an their resource capacities), the consolidation problem of concern is to provision resources for each virtualization and determine the host on which each VM will execute. The aim is to place virtual machines in as few physical machines as possible simultaneously not violating the resource "risk" limits of each application.

III. Related Work

Management of service-level agreements (SLAs) and SLA driven resource allocation has received considerable attention within the research community [1], [5], [2], [10], [8], [3]. In this section, we analyze the important contributions of current literature and identify their shortcomings.

Chaisiri et al. [4] propose an algorithm to find out the optimal placement plan which minimizes the cost spending in the reservation and on-demand payment plans offered by cloud providers. The cost depends on the prices charged by each provider for each resource. They view the provisioning of resources in three phases: reservation, utilization and on-demand. The algorithm uses stochastic integer programming and works in two stages: the first stage calculates the number of VMs provisioned (demand) in the reservation phase and the second stage calculates the number of VMs allocated in both utilization and on-demand phases. Using the set of all possible demands and prices, an objective function and a set of constraints is developed considering the cost of resources charged by each provider in each phase. They consider all possible combinations of VMs to be put together by formulating the placement problem as an Integer Linear Programming Problem (ILP). The ILP's complexity being a NP-hard problem does not scale well to larger input sizes.

Verma et al. [13] propose a 90^{th} percentile based provisioning approach. They propose two algorithms, (i) Correlation Based Placement (CBP) in which each VM is sized at 90^{th} percentile of its peak resource demand. For each VM to be placed, it is first checked whether it has a positive correlation with any of the VMs that are already placed in that particular machine. 2) Peak Clustering Based Placement (PCP): with PCP approach each VM is provisioned with the 90th percentile utilization value and a peak buffer of capacity equal to the maximum of peak size of all the VMs with considerably low correlation among their peaks of resource demand is kept reserved for all those VMs. CBP has an obvious disadvantage of ending up using many servers when there are many correlated VMs or applications. Although PCP fixes up this problem but still provisioning resource for each VM individually and presence of a peak buffer leave much scope for resource wastage.

Meng et al. propose that VMs should be provisioned jointly [8]. They have shown that if we put un-correlated VMs together capping each of them at their peak utilization value and provision resource for the aggregated utilization profile then that will lead to significant savings by packing VMs more densely however keeping the SLA violation metrics low. The joint provisioning approach definitely looks lucrative because it eliminates the concern of provisioning each VM individually as it can be formally proven that if the aggregated utilization profile abides by the joint-VM SLA constraint then all the VMs will also abide by the single-VM SLA constraints individually. However, this doesnt hold in case of all SLA models. The SLA constraint discussed by them is simple in the sense that it only considers how many times the violation did occur. If we incorporate the intensity of the violation i.e., the extent by which the violation had occurred into the SLA constraint model then the theorem doesnt hold. Another significant drawback of this approach is that once a pair of un-correlated VMs are formed we do not consider the correlation with some other already formed pair. Thus this joint pair-wise provisioning still leaves scope for resource wastage.

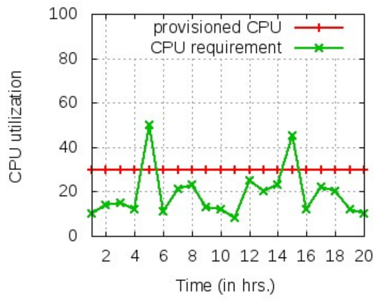
Abrahao et al. propose an SLA-driven approach for capacity management in [1]. They have described two SLA models namely normal and surge which allow customer to pay for extra capacity than that of normally required. They have also developed a penalty-reward based cost model i.e., when customers expectations in terms of SLA conditions are met rewards are gained and when SLA conditions are violated e.g., increased response time etc then penalty is incurred. In their framework they use a workload forecaster in order to predict future workloads and that information is fed into a capacity Manager. It takes a decision by computing all the potential rewards and penalties and sends that decision in terms of new IDC configuration to the virtualization layer. They assume that the fraction of the total resources available in the physical server, provisioned to a particular VM is inversely proportional to the average service time of that VM. This seems to be a reasonable assumption but it doesnt take into account the potential rewards gained by considering correlation among workloads.

We have incorporated the notion of user-defined acceptable level of resource unavailability in provisioning step of VMs. We have also incorporated a broader aspect of correlation among VMs by considering correlation on aggregated resource demand.

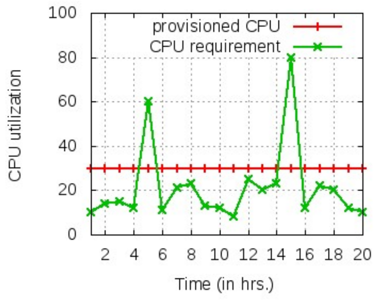
IV. Resource Unavailability Scoring Metrics

Resource unavailability implies the provisioned capacity to be not enough for a particular resource utilization profile. Resource utilization values are sampled at regular time intervals and this time-series of utilization values are taken as a resource utilization profile. In statistics, the notion of risk is often modeled as *the expected value of some outcome seen as undesirable*. A violation metric or resource unavailability metric combines the probabilities of various possible events and an assessment of the corresponding adverse effects into a single value.

Whenever a VM is provisioned at a value less than it's peak-utilization value, there is always a chance that resource requirements exceed the available provisioned capacity. The degree of violation is equivalent to the inadequacy of the resource under consideration. As mentioned earlier, for the purposes of this paper, we consider only CPU as the resource of interest. The proposed technique can be extended to cover other resource types such as memory allocations and I/O bandwidth and techniques to combine them as single metric is separate direction that we do not explore.



(a) CPU utilization profile 1



(b) CPU utilization profile 2

Figure 1. Different cases of Provisioning

A. Definitions of Violation metrics

Violation Count (V_c): Ratio of the number of violations in a time period to the total number of sampled resource requirement instances.

Violation Intensity (V_i): Ratio of the total amount of resource that is shortfall to the total amount of resource required to complete a set of tasks. Given a resource requirement time series and a provisioned capacity for that resource, V_i is the ratio of the area under the utilization curve that is not covered by the provisioned capacity to the total area under the utilization curve.

In Figure 1 two CPU utilization profiles are shown. Along the x-axis is the time scale and along y-axis are the CPU utilization values of an application. 30% is the provisioned CPU capacity in both cases. Now the area under CPU utilization curve under the provisioned capacity line signifies that much of resource is always guaranteed to be provided to the VM whereas the area under the CPU utilization curve that resides outside the provisioned capacity line signifies that much resource may not be available when required. If there are resources available beyond provisioned capacity then only Hypervisor will provide that much of resource. The amount of resources that lies above the provisioned capacity line indicates that much of resources have to provided in the future resulting in further demand and possibly starvation of resources in the future if none of the pending computations has to be dropped which typically happens in case of batch processing. Where there is a deadline to finish computational tasks (e.g., serving web-requests) the amount of resources that lies above the provisioned capacity line indicates that a proportionate number of requests may be dropped due to lack of computational resources.

Illustration: Now from Figure 1(a) it is found that there are 2 out of 20 points where violation occurred. Similarly for Figure 1(b) as well there are 2 out of 20 points where violation took place. So in both the cases,

$$V_c = \frac{2}{10} * 100\% = 20\%$$

But clearly intensity of the violation in case of Figure 1(b) was quite higher than that of Figure 1(a). Still V_c is showing the same score of resource unavailability in both the cases where Figure 1(b) looks definitely more "risky" than that of Figure 1(a).
Now if we go by the notion of V_i,

- **case 1a:** $V_i = 9.51\%$
- **case 1b:** $V_i = 19.37\%$

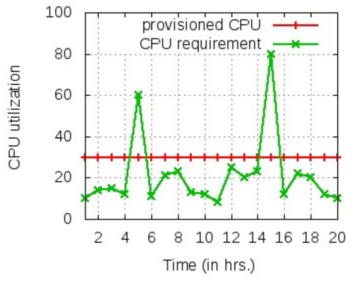
We can say that V_i gives an idea about the severity in case of Figure 1(b) by incorporating the intensity of the violation into the computation. In case of a batch processing scenario this translates to the fact that 9.51% and 19.37% of the total resources required will have to provided sometime later. This either will increase resource utilization levels at some other time intervals or will result in increased task completion time (as resources needed will be provisioned much later). In case of a web-server scenario, the resource unavailability or risk of 9.51% and 19.37% can translate to a proportional number of dropped requests. If number of requests dropped factors into a client's SLA, using the resource unavailability notion of risk is beneficial.

In our work, we formally define *risk* as violation intensity (V_i)—the cumulative resource unavailability over period of time. Further, we assume that the administrator has prior knowledge about the maximum acceptable risk per application. However, our placement algorithm is independent of the sizing methodology. Any suitable sizing methodology could be used as an input to the consolidation algorithm.

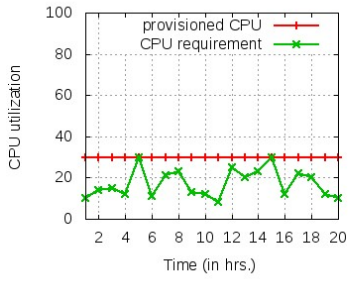
B. Risk-Aware Provisioning of Virtual Machines

In our technique, the acceptable level of risk (aggregate resource unavailability) for each VM is user-defined. In order to achieve the desired provisioning point for a given value of risk, a binary search based technique can search within the minimum and maximum utilization values to be alloted. A simple scheme that increases or decreases the provision resources in log-step based on whether the risk level is met, will determine the least required provisioned capacity. With d discrete time instances of historical resource utilizations known and n the maximum utilization level, the complexity of the above search is $O(d \log n)$.

Illustration: An example is as shown in Figure 2. Given the CPU utilization profile shown in Figure 2(a) we have to size the VM's risk level (V_i) is 20%.



(a) CPU utilization profile



(b) sized CPU utilization profile with Vi = 20%

Figure 2. Illustration of risk-aware provisioning for a VM.

In order to do this we first try with the lower most value of CPU as the provisioned capacity and find out the score of V_i. If the score is found greater than that of the acceptable value then the size is increased or the other way round as we typically do in binary search. In Figure 2(b) the sized CPU utilization profile is shown. Instead of binary search some other searching algorithm can also be used e.g., regula falsi method [7].

V. Placement of VMs using Correlation on Aggregated Resource Demand

Correlation between resource demands of different VMs is a well-studied area in the literature [8], [13], [10], [5]. Correlation is primarily considered to find out resource demand profiles that are mostly complementary to each other i.e., when one profile exhibits high requirement then the other one has as minimal requirement as possible. Pair-wise joint provisioning does the same thing for pairs of VMs [8]. But these pairs in turns can be highly correlated or un-correlated to each other. So, while placing a particular VM it is crucial to consider whether it is highly correlated with the aggregated resource demand as determined by all placed VMs on a given physical machine or not. Our algorithm is based on the idea that a VM could be placed only if it exhibits minimal correlation with the aggregated resource demand of already placed VMs.

The algorithm is given in Algorithm 1. The algorithm takes the profiles of all VMs after the risk aware sizing in $\{V\}$. Whereas all the capacities of the physical servers are in $\{C\}$. Here we intend to find the number of physical servers required to host all the VMs. Hence, we assume that it will never be the case that all the VMs can not

Algorithm 1 Resource Aggregation based Consolidation Algorithm

```
Input: 1. C[]        //$C[i]$ denotes capacity of $i^{th}$ PM
    and $C[i] \geq C[j]$ if $i < j$
       2. V[][]      //$V[i][t]$ denotes utilization value
    of $i^{th}$ VM at $t^{th}$ sample point
Output: $A[]$        //$A[i]$ denotes the PM index where
    $i^{th}$ VM is placed
 1: Initialize $A[i] \leftarrow$ invalid $\forall i$
 2: $M \leftarrow$ computeCorrelationMatrix (V[][])
 3: currentPM $\leftarrow 0$
 4: repeat
 5:     clear $U$   //$U[t]$ denotes resources consumed at $t^{th}$
        sample point
 6:     Choose i, j with minimum $M[i][j]$ and $A[i] = A[j] =$ invalid
 7:     $U[t] \leftarrow V[i][t] + V[j][t] \quad \forall t$
 8:     $A[i] \leftarrow$ currentPM, $A[j] \leftarrow$ currentPM
 9:     while $U[t] < C[currentPM] \quad \forall t$ do
10:         $j \leftarrow i$ s.t correlation($U, V[i]$) is minimum and
            $A[j]$ is invalid
11:         $U[t] \leftarrow U[t] + V[j][t] \quad \forall t$
12:     end while
13:     currentPM $\leftarrow$ currentPM $+1$
14: until every VM is placed
```

be hosted within the given set of physical servers. We first compute a matrix called correlation matrix ($M[][]$) where $M[i][j] =$ Pearson's correlation co-efficient [9] between profiles of the i^{th} and j^{th} VM in $\{V\}$. Pearson's correlation coefficient between a pair of time-series $\{x_1, x_2, ..., x_N\}$ and $\{y_1, y_2, ..., y_N\}$ is defined as follows,

$$r_{xy} = \frac{N\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{N\sum x_i^2 - (\sum x_i)^2}\sqrt{N\sum y_i^2 - (\sum y_i)^2}}$$

We use a subroutine called *computeCorrelationMatrix()* to cumpute the correlation matrix. We take an FFD (First-Fit-Decreasing) approach to fill up the servers. We first take up the largest possible server and try to fill it up to the point where the aggregated resource utilization exceeds the available capacity of the physical server at some points along time axis.

For each physical server we first consult the correlation matrix to find out a pair of VMs that is least correlated (step 6). Both of them are placed on a particular server and resource utilization of that server becomes the aggregation of the resource demands of the two (steps 7-8). After that we try to find out some other VM from $\{V\}$ such that correlation co-efficient of this VM and the aggregated profile of the already placed VMs is minimum across all unplaced VMs (step 10). We continue to place VMs on that server until it gets exhausted i.e., at some points the aggregated resource demand exceeds capacity of that server (steps 9-11). We keep on doing this until all VMs are placed (steps 4-14).

It overcomes the shortcoming of pairwise joint provisioning [8] in the sense that while placing a particular VM it always checks the correlation between already placed VMs and the yet to be placed VM. In pairwise joint provisioning pairs of least correlated VMs are formed but the fact that this joint profiles can mutually be correlated to each other is not considered. It overcomes the shortcoming of CBP [13] in the sense that unless there is room in a server it continues to place VMs in that particular server unlike CBP where even if there is room a VM is not placed if it exhibits correlation beyond some threshold with any of the already placed VMs. Although PCP [13] seems to resolve this issue but still it leaves much scope for wastage of resource by enveloping the resource demand profile, provisioning each of the VMs on an individual basis and keeping a common peak buffer with capacity equalizing the maximum peak across all VMs placed within that server. Whereas our approach exploits the potential multiplexing of resources by doing a joint provisioning on an aggregation basis. Apart from these advantages our algorithm unlike PCP and CBP [13] leverages the hassle of tuning numerous parameters depending on the resource demand pattern.

VI. Experimental Setup and Evaluation

Workload characterization in the literature [8], [13], [10], [5] reveals that the resource demands of VMs in an enterprise setup closely resembles normal distribution with long tails. Most of the time a particular VM does not require the peak amount of resource. To incorporate this typical nature of resource utilization pattern we have taken a data-set consisting of several time series each following normal distribution. First of all we are interested to know how does our algorithm perform compared to other existing ones. We are also interested to know what factors or parameters of the resource demand time series data-set can effect the efficiency of the algorithms. In order to investigate these we have used the following framework.

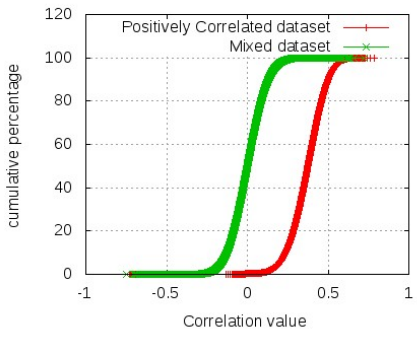
We have taken 1000 time-series each of which is normally distributed corresponding to CPU utilization traces of 1000 VMs. We have taken pairwise joint provisioning [8] and CBP [13] as baseline cases to compare against.

A. Effect of correlation among resource demands of VMs

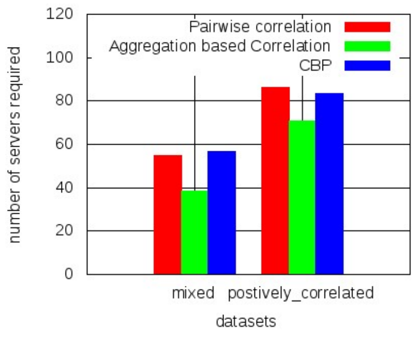
Correlation on aggregated resource demand has an advantage over the others is that it considers a broader view of the correlation between existing resource demands of the VMs. We now address the question how does correlation among different VMs can effect solutions given by our algorithm as well as other existing algorithms.

To answer the above question, we use two different types of datasets,

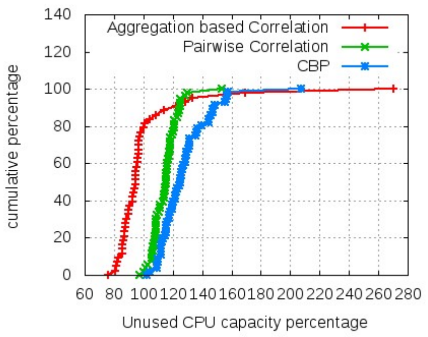
- (1) **Mixed:** The profile dataset consists of a mixed distribution of positively correlated, negatively correlated and uncorrelated VMs.
- (2) **Positively Correlated:** The data consists of a majority of positively correlated VMs.



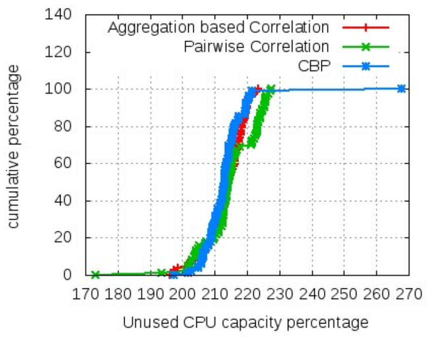
(a) CDF of correlation values in mixed dataset



(b) Number of servers taken by different algorithms



(c) CDF of unused CPU capacity across servers for mixed dataset



(d) CDF of unused CPU capacity across servers for positively correlated dataset

Figure 3. Effect of correlation among VMs on consolidation

The Cumulative Distribution Functions (CDF) of the correlation values of all possible VM pairs are given in the Figure 3(a). We applied all the three algorithms on these datasets and have reported consolidation efficiency in terms of number of physical hosts (or interchangeably referred as servers) required by each of the algorithms and unused CPU capacity in each physical server. Unused CPU capacity left in a particular server is measured by taking an integration of unused amount of CPU capacity over the time-period.

In Figure 3(b) the average number of servers required in each of the algorithms are plotted. In case of the mixed dataset, aggregation based correlation took around 38 servers on an average to place 1000 VMs whereas Pairwise joint provisioning and CBP took 55 physical machines and 56 physical machines, respectively, indicating a reduction in number of servers by 32%. In case of positively correlated dataset each of the algorithm required comparatively higher number of servers to place all the VMs. Our algorithm took 71 servers whereas pairwise and CBP took 86 and 86 servers respectively. So in this case as well our algorithm leads to reduction in number of servers by $15-17\%$. These experiments show that our algorithm does better packing of VMs than the others on all of the three types of profile datasets. In general, the advantage is more prominent when profiles are of mixed nature i.e., it has both correlated, un-correlated and negatively correlated VMs which are indeed the most common scenario in a real enterprise setup [13], [8].

In Figure 3(c) and Figure 3(d) CDFs of unused CPU in case of mixed and positively correlated dataset are plotted. We can see that in case of our algorithm i.e., aggregation based correlation, CPU wasted is quite less than that of the other algorithms in case of the mixed dataset. As correlation among different VMs increases each of the three CDFs shifts towards the higher values and the difference between them also reduces.

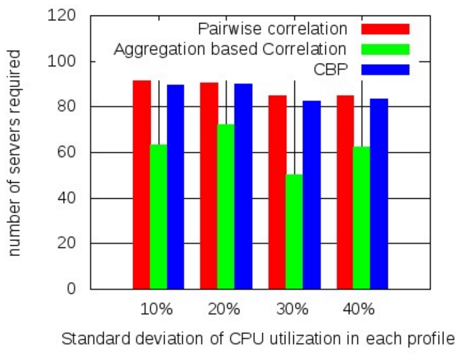


Figure 4. Effect of varying burstiness on consolidation

We have also studied the effect of burstiness of utilization values on consolidation performance of each of the algorithms. We have measured burstiness of a utilization profile by standard deviation of utilization values of a particular VM over time. Basically standard deviation is the primary variable that in turns controls correlation across different VMs. More the burstiness of the utilization profile more is the chance of multiplexing of resources. In our experiment we varied the standard deviation of the CPU resource requirements for each profiles.

From Figure 4 we can say that consolidation efficiency of all the algorithms improves as standard deviation increases if there is a mixed distribution of positively, negatively and un-correlated VMs in the dataset.

B. *Effect of varying acceptable level of resource unavailability*

We have used the notion of acceptable level of resource unavailability in order to size the VMs before placements. We have studied what effect does acceptable level of resource unavailability have on the consolidation performance of our algorithm as well as on other baseline cases. Here we have taken V_i as the resource unavailability scoring metric. In Figure 5 number of active servers is

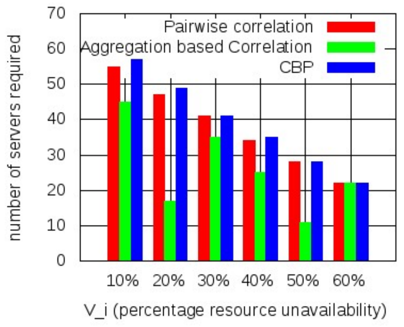


Figure 5. Effect of varying acceptable resource unavailability

plotted along y-axis for all the three algorithms for different acceptable resource unavailability scores. Apart from a few undulations we can observe that the number of servers required to hold all the VMs decreases as acceptable resource unavailability score keeps on increasing. When the acceptable resource unavailability score reaches 60% all the three algorithm converge to the same number of servers as per expectation.

Table I
EFFECT OF VARYING ACCEPTABLE RESOURCE UNAVAILABILITY ON UNUSED CPU CAPACITY

	90th percentile of Unused CPU capacity		
Vi	CBP	Pair-wise	Aggregation based
10%	150%	120%	110%
20%	135%	110%	122%
30%	120%	108%	95%
40%	115%	105%	100%
50%	110%	92%	100%
60%	90%	85%	70%

Table I summarizes the effect of varying acceptable resource unavailability on unused CPU capacity. The 90^{th} percentile values in the CDFs of unused CPU capacity across all the physical servers are presented in this table. We observe that as the acceptable resource unavailability (V_i) is varied from $10\% - 60\%$ the unused CPU capacity reduces. This is due to the fact that as we increase the acceptable resource unavailability, undulation in the utilization profile reduces.

C. *Complexity of Correlation on aggregation based placement algorithm*

If there are n number of VMs in a dataset and there are d number of sample points in each time series then time complexity of all our algorithm and pairwise joint provisioning algorithm is $O(n^3)$ whereas CBP takes $O(n^2d)$ time. In case of pairwise joint provisioning computation of correlation matrix takes $O(n^2d)$ time and repetitively finding the minimum entry in the matrix takes $O(n^3)$ time. In case of CBP there is no need of computation of correlation matrix, so complexity is only $O(n^2d)$. Our algorithm takes $O(n^2d)$ time to populate the correlation matrix, $O(n^3)$ time to find out the minimum entry, and $O(n^2d)$ time to find out a particular profile that has minimum correlation with the aggregated profile. The increase in complexity is due to repetitive finding of the minimum entry in the correlation matrix that ensures that least correlated VMs are chosen whenever we start placing VMs on some PM. In our algorithm finding minimum entry from correlation matrix is done whenever one PM is exhausted. So the complexity becomes $O(n^2 * (n/k))$ where k is average number of VMs placed in one PM. But as VM packing density is typically quite high the actual complexity is closer to $O(n^2)$.

VII. CONCLUSION

In this work, we discussed shortcomings of existing approaches for measuring goodness of resource provisioning. We proposed "risk", a goodness measuring metric incorporating the intensity of violations (degree of resource unavailability). Shortcomings of the existing placement frameworks were addressed and a novel iterative placement algorithm was proposed. The iterative placement algorithm considered already placed VMs to compute correlations between "used" aggregate resource levels and application resource requirements. Experimental evaluation revealed that our algorithm leads to significant reduction in the number of active physical hosts compared to that of the existing placement algorithms. We also discussed in detail effects of correlation among VMs, burstiness of utilization profiles of individual VMs and acceptable resource unavailability on consolidation efficiency of all the algorithms. In case of a mixed dataset consisting of positively, negatively and un-correlated VMs our algorithm resulted in 32% reduction in the number of active servers required compared to CBP and pairwise joint VM provisioning. Whereas in case of a dataset consisting mostly of positively correlated VMs the reduction goes down to $15\% - 17\%$. Future work would include translation of application-level SLAs to resource-level risk metrics. Currently our framework takes into account only CPU requirement of VMs. We are planning to expand our ideas to multiple dimensions in the future. We are also looking for linear programming based fractional solutions as an alternative approach.

Acknowledgement
This project is supported in part by a grant (grant no. 09IBM001 at IIT Bombay) from IBM under the CAS program. Part of this work was done through a summer internship (May-July 2011) at IBM India Software Lab, Pune.

References

[1] Bruno Abrahao, Virgilio Almeida, Almeida Jussara, and et. al. Self-adaptive SLA-Driven Capacity Management for Internet Services. In *Tenth IEEE/IFIP, Network Operations and Management Symposium, 2006.*

[2] Norman Bobroff, Andrzej Kochut, and Kirk Beaty. Dynamic Placement of Virtual Machines for Managing SLA Violations. In *Integrated Network Management, 2007.*

[3] David Breitgand and Amir Epstein. SLA-aware Placement of Multi-Virtual Machine Elastic Services in Compute Clouds. *IFIP/IEEE International Symposium on Integrated Network Management*, 2011.

[4] Sivadon. Chaisiri, Bu-Sung Lee, and Dusit Niyato. Optimal virtual machine placement across multiple cloud providers. In *IEEE Asia-Pacific Services Computing Conference, 2009.*

[5] Ming Chen, Hui Zhang, Ya-Yunn Su, and Xiaorui Wang et. al. Effective VM sizing in Virtualized Data Centers. In *Twelfth IFIP/IEEE International Symposium on Integrated Network Management*, 2011.

[6] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. In *Sixth Symposium on Operating System Design and Implementatio*, 2004.

[7] M. Dowell and P. Jarratt. A modified regula falsi method for computing the root of an equation. *BIT Numerical Mathematics*, 11:pp 168–174, 1971.

[8] Xiaoqiao Meng, Canturk Isci, Jeffrey Kephart, Li Zhang, and et. al. Efficient resource provisioning in compute clouds via VM multiplexing. In *Seventh International Conference on Autonomic Computing*, 2010.

[9] Joseph Lee Rodgers and W. Alan Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):pp. 59–66, 1988.

[10] Jerry Rolia, Ludmila Cherkasova, Martin Arlitt, and Artur Andrzejak. A capacity management service for resource pools. In *Fifth International Workshop on Software and Performance*, 2005.

[11] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale. Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting. In *IEEE CLOUD*, 2011.

[12] Vanish Talwar, Klara Nahrstedt, and Dejan Milojicic. Modeling remote desktop systems in utility environments with application to qos management. In *Integrated Network Management, 2009.*

[13] Akshat Verma et.al. Server workload analysis for power minimization using consolidation. In *USENIX Annual Technical Conference*, 2009.