Portfolio Theory-Based Resource Assignment in a Cloud Computing System

Inkwon Hwang and Massoud Pedram
University of Southern California
Los Angeles, CA, USA
{inkwonhw, pedram}@usc.edu

***Abstract*— The focus of this paper is on energy-aware resource management in a cloud computing system. Much of the existing work assumes that the resource requirements for various applications are known and given as scalar values. However, it is very difficult to know the exact resource requirements, and thus, it is more appropriate to treat resource requirements for applications as random variables with known characteristics. For a desired quality of service, the required total resource amount can then be estimated as a function of the means and standard deviations of these random variables. Inspired by the modern portfolio theory, this paper presents algorithms that minimize the total amount of estimated resource in the system. A source of difficulty is that some of the aforesaid random variables may be correlated with each other. The proposed algorithms effectively deal with correlated applications. Experimental results show that, in spite of its simplicity and scalability, the proposed solution outperforms the well-known heuristics i.e., *first fit decreasing (FFD)* and *best fit decreasing (BFD*) by an average of 10% while having a low execution time.**

Keywords- Cloud computing; portfolio effect; bin-packing; resource allocation

I. INTRODUCTION

Cloud computing systems, which are typically housed in facilities called data centers, are composed of a large number (say, thousands) of servers, each server consuming 100's of Watt. This means that the power consumption of the servers plus cooling and air conditioning units in a typical datacenter can easily exceed 1MW. With 10 cents per KWhr of electrical energy consumed, the electrical energy cost alone will be in the order of thousands of dollars per day. Thus, there is a growing need for energy-aware resource management in cloud computing systems. Considering that much of the time, server machines in a data center are under-utilized, efficient resource management can be quite effective in reducing the electrical energy cost of the cloud computing systems.

Energy aware resource management problem has been the subject of many previous studies. A key approach has been to adopt a performance model and allocate resources so as to maximize the performance. For example, in [1], Gandi *et al.* present a performance model based on the queuing theory and allocate power to minimize the average response time of the tasks. Quality of the results obtained by such an approach is strongly dependent on the accuracy of the performance model. The issue is that it is difficult to come up with an accurate (assumption-free) performance model under realistic usage scenarios.

Another approach for energy-aware resource management is based on the control theory. For example, in [2], Raghavendra *et al.* present five power management controllers that utilize feedback control loops to minimize energy while meeting some performance targets. This approach is quite practical but the challenge is to determine control parameters, which are supposed to be customized for target systems. Yet another approach starts by assuming that resource requirements of applications are known and assigns resources to the applications. The approach makes the further assumption that there are no performance violations as long as enough resources are allocated to each application. For example, in [3], Srikantaiah *et al.* investigate the problem of application consolidation to minimize energy consumption in a cloud computing system. They assume that resource requirements for each application are pre-known, and thus formulate the problem as a multi-dimensional bin packing problem. Stillwell *et al.*, who study resource allocation problem for HPC applications [4], also assume that resource requirements are known a priori. Wilcox *et al.*, which rely on a probabilistic resource requirement model [5], simply calculate the amount of required resource from the given probability density function (pdf).

In this paper, we assume resource requirements for each application are given as random variables with known means and standard deviations. We believe random variable resource requirement model is more realistic and useful than deterministic model; this is because resource requirements are estimated from historical data and profiling, which are subject to noise and uncertainties and show variability. If the resource requirements are modeled as random variables, we can reduce the total amount of required resource by applying principles of the modern portfolio theory.

There can be tens of thousands of virtual machines and thousands of physical machines in a cloud computing system. Hence, scalability of any proposed resource management solution is a must. In this paper, we present a portfolio-based hierarchical resource management solution, which is scalable and reduce the energy cost of the cloud computing system.

The rest of the paper is composed as follow: In the section II, we introduce the concept of portfolio effect and our problem statements. Main algorithms and detailed explanation about the proposed scheme is explained in section III. In section IV, the simulation results are shown and discussed. Finally, we summarize and conclude in section V.

II. PORTFOLIO BASED RESOURCE ALLOCATION

A. Estimation of the required resource

We assume that the amount of a required resource for each application is specified as a random variable (*RV*). This amount is estimated based on the application characteristics and computing needs as well as the target quality of service (*QoS*) level. If the cumulative distribution function (*cdf*) of this *RV* is known, we can estimate the amount of required resource from the *cdf* based on our target *QoS*. However, such detailed information may not be available in many cases. Without knowledge of the *cdf*, and based on only the *mean* and *standard deviation* of *RVs*, the amount of required resource can be estimated by *Cantelli's inequality* [6], which is the one-tailed variant of *Chebyshev's inequality*:

$$P\{X \geq \mu_X + \beta\sigma_X\} \leq {}^{1}/_{(1+\beta^2)} \quad ,\beta > 0 \tag{1}$$

According to the *Cantelli's inequality*, the amount of resource to achieve a target *QoS* of $\beta^2/(1+\beta^2)$ may be calculated as:

$$\mu_X + \beta\sigma_X \tag{2}$$

In this study, our target *QoS* is 95% resource satisfaction i.e., 95% of the time the resource allocation meets the resource requirement of the application. This is achieved with β = 4.4. However, the above equation tends to overbook the resource because the *Cantelli's inequality* does not give a tight bound. If we know more information about the *VM* such as the *cdf*, we can assign fewer resources while meeting the same *QoS*. For example, if we are told that *RV X* is normally distributed, β can be set to as low as 1.7, which is much smaller than what the *Cantelli's inequality* gives.

According to the *central limit theorem* (*CLT*), the mean of a sufficiently large number of independent random variables, each with finite mean and variance, will be approximately normally distributed [6]. *CLT* holds even for weakly dependent *RVs*. Hence, we can use smaller β (=1.7) if the *RV* is sum of large number of weakly dependent *RVs*.

B. Review of the modern portfolio theory

Modern portfolio theory (MPT) is a theory of finance, which attempts to maximize a portfolio's expected return for a given amount of portfolio risk, or equivalently minimize risk for a given level of expected return, by carefully choosing the proportions of various assets. The MPT models an asset's *return* as a normally distributed function, defines *risk* as the standard deviation of return, and models a portfolio as a weighted combination of assets, so that the return of a portfolio is the weighted combination of the assets' returns. By combining different assets whose returns are not perfectly positively correlated, MPT seeks to reduce the total variance of the portfolio return [7].

MPT reduces risk of portfolio through the ***portfolio effect***, which may be stated as follows: the risk of a portfolio is always less than or equal to sum of each asset's risk (3). Let *Y* denote a portfolio composed of assets X_i. Then,

$$\mu_Y = \sum_i \mu_{X_i}, \quad {\sigma_Y}^2 = \sum_i \sum_j \rho_{ij}\sigma_{X_i}\sigma_{X_j} \leq \left(\sum_i \sigma_{X_i}\right)^2$$
$$where\ Y = \sum_i X_i \ \text{and}\ \rho_{ij} is\ the\ correlation\ coefficient\ between\ X_i\ and\ X_j (-1 \leq \rho_{ij} \leq 1) \tag{3}$$

The degree of risk reduction is a function of a correlation coefficient (ρ_{ij})—the smaller ρ_{ij} is, the lower the risk is (cf. Figure 1.) In other words, one has to avoid from putting highly positively correlated assets into the same portfolio.

In this study we apply the *portfolio effect* to resource assignment problem in a cloud computing system in order to reduce the standard deviation of the required resource. From (2), a reduction in the standard deviation also reduces the amount of required resource. The goal is then to do a *VM* to *PM* assignment such that the sum of standard deviations for all *PMs* is minimized. Note that this is a reasonable problem formulation since by minimizing the total amount of required resource for the given *QoS* level, fewer *PMs* can be utilized; hence, energy cost can be reduced.

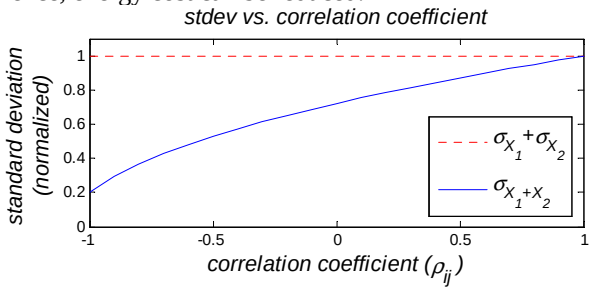

Figure 1. Effect of correlations between portfolio's assets on a portfolio's risk (standard deviation)

We illustrate the *portfolio effect* applied to resource management by the following example (cf. Figure 2.) Consider four *applications* and two *physical machines* (PM_1 and PM_2.). For simplicity sake, we assume applications are uncorrelated. The case on the left assigns the 1st and 2nd applications to PM_1 and the other two to PM_2. Once a collection of VMs are assigned to a *PM*, the standard deviation of the required resource for that *PM* is computed as the square root of the sum of variances of assigned *applications*. The total standard deviation of the left case is 31.6 ($= \sqrt{15^2+5^2} + \sqrt{15^2+5^2}$.) On the other hand, the case on the right assigns the 1st and 4th applications to PM_1 and the other two to PM_2. In this case, the total standard deviation is 28.3 ($= \sqrt{15^2+15^2} + \sqrt{5^2+5^2}$), which is smaller. As discussed before, the second case will require less resource to meet the same *QoS* level, and hence, it is more desirable.

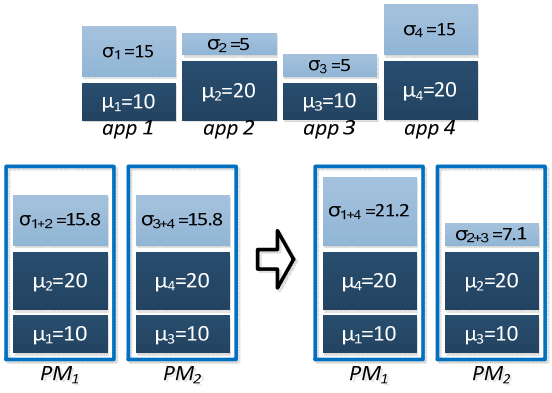

Figure 2. Comparison of different resource allocation cases

From the above discussion, we conclude the following:

1. It is desirable to assign *applications* that are least (positively) correlated to the same *PM*.
2. *Applications* must be well deployed on *PMs* to maximize the portfolio effect.

It is well known that the performance drops dramatically when CPUs are almost fully utilized (close to 100% utilization) [8]. Hence, the target CPU utilization level should be appreciably lower than 100%, e.g. 80%. We assume that the resource capacity of the CPUs is set based on this target CPU utilization level.

Our target system is a *virtualized* cloud computing system. This is not because the proposed solution is only applicable to virtualized systems; Instead it is because the proposed solution can easily be implemented and applied under virtualized systems—live migration [9], which decreases the performance overhead of virtual machine migration, and performance isolation [10], which makes effective resource management easier, are supported in virtualized environments.

C. Problem statement - bin-packing optimization problem

There are *M* physical machines (*PMs*) and *N* virtual machines (*VMs.*) We assume that the amount of resource required by *VMs* are random variables (*RVs*) with known means and standard deviations. In this paper, we only consider CPU resource. The work may be extended to deal with multiple resource type. We leave it as the future work.

Our objective is to find the optimal assignment (e_{nm}) of *VMs* to *PMs* so as to minimize the total amount of assigned resource while meeting the target *QoS*.

To provide a precise problem formulation, we need to give some definitions and notation.

X_n : CPU resource required by the n^{th} *VM* ($n = 1,2,\ldots,N$.) This is a *RV* (similar to an asset) with mean and variance of μ_{X_n} and $\sigma_{X_n}{}^2$

ρ_{ij} : Correlation coefficient between X_i and X_j

e_{nm} : Assignment variable. It is 1 if the n^{th} *VM* is assigned to the m^{th} *PM*, and 0 otherwise

$Y_m = \sum_{n=1}^{N} e_{nm} \cdot X_n$: Total CPU resource demand on the m^{th} *PM*. This is a *RV* (similar to a portfolio)

r_m : CPU resource capacity of the m^{th} *PM* ($m = 1,2,\ldots,M$)

$f_m = \begin{cases} 1 & if\ Y_m > 0 \\ 0 & otherwise \end{cases}$

The *minimum resource assignment* (MRA) problem may be formulated as follows.

$$Find\ e_{nm}\ to\ minimize\ \sum_m f_m \cdot r_m$$
$$s.t.\ \begin{cases} \forall n, m & e_{nm} \in \{0,1\} \\ \forall n & \sum_{m=1}^{M} e_{nm} = 1 \\ \forall m & P\{Y_m \le r_m\} \ge 0.95 \end{cases} \quad (4)$$

If *PMs* are homogenous, the objective is simply to minimize the number of active *PMs* ($\sum_m f_m$.) However, our target system is comprised of heterogeneous *PMs*, thus our objective function is the sum of total required resource for all *active PMs* (those to which at least one *VM* has been assigned.) Note that the above formulation assumes that resource capacity of a *PM* will be fully utilized as soon as the *PM* becomes active. It implicitly drives a solution to have fewer active *PMs* as possible. As shown in (2), the last condition of (4) can be re-written as follow:

$$\forall m\ \ \mu_{Y_m} + \beta\sigma_{Y_m} \le r_m$$
$$where\ \beta = \begin{cases} 1.7 & if\ Y_m\ is\ distributed \\ & normally \\ 4.4 & otherwise \end{cases} \quad (5)$$

The above problem is variation of the *bin-packing optimization* problem, which is known to be a NP-hard problem [6]. Several heuristic algorithms have been presented in the literature to solve this problem, including the *first fit decreasing (FFD)* and *best fit decreasing (BFD)* algorithms. The solution should be scalable because there can be tens of thousands of *VM*s and thousands *of PMs* in a cloud system [11]. In this paper, we present a hierarchical resource management solution. The algorithms used as parts of this solution are explained in section III, and we will show that the proposed solution outperforms *FFD* and *BFD* in terms of our objective function (4) in section IV. The improvement is possible because the proposed solution effectively maximizes the portfolio effect by considering correlation between required resources of applications.

The proposed solution is composed of a *cloud-level resource manager* and *cluster-level resource managers.* We introduce additional definitions and notation.

S_c : Set of indices *m* of the *PMs* that belong to the c^{th} cluster ($c = 1, 2,\ldots,C$)

g_{nc} : Assignment variable. It is 1 if the n^{th} VM is assigned to the c^{th} cluster; otherwise 0 ($c = 1,2,\ldots,C$)

$Z_c = \sum_{n=1}^{N} g_{nc} \cdot X_n$: Total CPU resource demand on the c^{th} cluster

$t_c = \sum_{m \in S_c} r_m$: CPU resource capacity of the c^{th} cluster

A *cloud-level resource manager* deploys *VMs* to clusters. A cluster can be defined as a group of *PMs*, which are connected to each other through a network switch and are sharing their power supply. Now we have a new optimization function. The summation of the means is independent of the assignment variables (f_{nc}), so the new objective is to minimize the sum of standard deviations of the clusters.

$$Min \sum_{c=1}^{C} \left(\mu_{Z_c} + \beta\sigma_{Z_c}\right) \equiv Min \sum_{c=1}^{C} \sigma_{Z_c}$$
$$s.t.\ \begin{cases} \forall n, c & f_{nc} \in \{0,1\} \\ \forall n & \sum_{c=1}^{C} g_{nc} = 1 \\ \forall c & \mu_{Z_c} + k\sigma_{Z_c} \le t_c \end{cases} \quad (6)$$

After the *cloud-level resource manager* deploys *VMs* to clusters, *cluster-level resource managers* deploy *VMs* to *PMs.* Note that *cluster-level resource managers* are independent of each other, i.e., they work in parallel. Resource allocation problem of *cluster-level resource managers* is the same as problem (4), but its size is much smaller.

III. Hierarchical Resource Management Solution

In this section we introduce main idea and algorithms for the hierarchical resource management solution. The proposed solution is composed of two resource managers, *cloud-level* and *cluster-level resource manager*. First the

cloud-level resource manager assigns *VMs* to *clusters*, and then the *cluster-level resource manager* allocates *VMs* to the *PMs* in the cluster. Modern data centers consist of many clusters, and the number of *PMs* in a cluster is bounded because of capacity of network switches and power supply capacity limitations: larger data centers or cloud computing systems have more clusters, but typically not bigger clusters. The key advantage of the proposed solution is that it converts a large problem into number of small independent problems. The size of small problems is bounded, and these problems can be solved in parallel, thus, there is an opportunity to apply more sophisticated and elaborate solution approaches to these problems, something which is not possible for the original (flat) problem because of its size.

A. Cloud-level Resource Manager

The *cloud-level resource manager* assigns *VMs* to clusters. Some *VMs* may be correlated with each other. For example, multiple *VMs* may be spawned off by the same application and *VMs* may correspond to different tiers of a multi-tiered application, etc. We can imagine two typical cases: all *VMs* are uncorrelated or some *VM*s are correlated each other. Because the uncorrelated case is simpler, we analyze and solve this case first in order to get some useful intuition, and then present another algorithm for the other case where some *VMs* are correlated each other.

1) Case1: uncorrelated ($\rho_{ij} = 0 \; if \; i \neq j$)

In this section we present the *Portfolio-based Resource Allocation algorithm I (PBRA$_{uncorr}$)*, which is for the case that resource requirement of *VMs* (X_n) is uncorrelated. Main idea of the algorithm is based on the following *proposition*.

Proposition: Let us say we have *N* balls with weights σ_n *(n = 1, 2, … , N)* and *K* bins. The size of the k^{th} bin is c_k and total size of all bins is the same as the number of balls ($\sum_{k=1}^{K} c_k = N$.) The balls are sorted by their weight in non-increasing order ($\sigma_i \geq \sigma_j \; for \; i < j$.) The bins are also sorted by their size (c_k) in non-increasing order ($c_i \geq c_j \; for \; i < j$.) Set S_k is the set of balls that are put into the k^{th} bin. Cost is defined as:

$$cost := \sum_{k=1}^{K} \sqrt{\sum_{\sigma_n \in S_k} {\sigma_n}^2} \tag{7}$$

The cost is minimized if the bin of bigger size contains the heavier (or the same weight) balls:

$$\begin{gathered} for \; \sigma_a \in S_i \; and \; \sigma_b \in S_j \quad (i < j) \\ \sigma_a \geq \sigma_b \end{gathered} \tag{8}$$

Proof: We first think about a simple case that there are only two bins (K = 2.) It will be generalized later. The initial deployment of balls is:

$S_1 = \{\sigma_1, \sigma_2, \dots, \sigma_{c_1}\} \; and \; S_2 = \{\sigma_{c_1+1}, \sigma_{c_1+2}, \dots, \sigma_N\}$

The cost of the initial deployment is:

$cost = \sqrt{\sum_{n=1}^{c_1} {\sigma_n}^2} + \sqrt{\sum_{n=c_1+1}^{N} {\sigma_n}^2}$

Now, each set is split into two subsets.

$S_1 = S_A \cup S_B$ and $S_2 = S_C \cup S_D$

Assume that the size of set S_A is the same as that of S_C. The cost can be rewritten as follow:

$cost = \sqrt{\mathrm{v}_A + \mathrm{v}_B} + \sqrt{\mathrm{v}_C + \mathrm{v}_D}, where \; \mathrm{v}_* = \sum_{\sigma_n \in S_*} {\sigma_n}^2$

Note that there is couple of inequalities between sets:

$\mathrm{v}_A \geq \mathrm{v}_C \; and \; \mathrm{v}_B \geq \mathrm{v}_D$

This is because any member of S_A is greater than members of S_C and any member of S_B is greater than members of S_D. In addition, the size of S_A is equal to that of S_C and the size of S_B is greater than or equal to that of S_D.

If set S_A is swapped for set S_C, the new cost is:

$cost' = \sqrt{\mathrm{v}_C + \mathrm{v}_B} + \sqrt{\mathrm{v}_A + \mathrm{v}_D}$

For easy comparison of the cost, we check if the difference of cost squared is positive or negative.

$$\begin{aligned} & cost'^2 - cost^2 \\ & = 2\left(\sqrt{(\mathrm{v}_C + \mathrm{v}_B)(\mathrm{v}_A + \mathrm{v}_D)} - \sqrt{(\mathrm{v}_A + \mathrm{v}_B)(\mathrm{v}_C + \mathrm{v}_D)}\right) \\ & (\mathrm{v}_C + \mathrm{v}_B)(\mathrm{v}_A + \mathrm{v}_D) - (\mathrm{v}_A + \mathrm{v}_B)(\mathrm{v}_C + \mathrm{v}_D) \\ & = (\mathrm{v}_A - \mathrm{v}_C)(\mathrm{v}_B - \mathrm{v}_D) \geq 0 \end{aligned}$$

Hence, $cost'^2 - cost^2 \geq 0$ and $cost' \geq cost$ and this means the initial deployment (8) is the optimal solution in terms of minimum cost.

This result can be generalized. For the case of more than two bins (K > 2), we can convert any swaps among a number of bins into a sequence of swaps between two bins. Hence, the proposition is true for the general case. □

Note that the above proposition is not perfectly fit to our problem. Capacity of bins is defined as the number of balls it can have, so the capacity is independent of the types of balls. However, in our problem, the number of *VMs* that can be assigned to a cluster depends on the resource requirements of the *VMs*. Nevertheless, simulation results in section IV show that the proposed algorithm, which is based on the above proposition, produces high quality results.

Pseudo code of *Portfolio-based Resource Allocation Algorithm I (PBRA$_{uncorr}$)* is shown below. Its main structure is similar to the *First Fit Decreasing (FFD)* algorithm.

Portfolio-based Resource Allocation Algorithm I (PBRA$_{uncorr}$)

Inputs: t_c , μ_{X_n} , σ_{X_n} , $and \; \rho_{ij}$

Output: g_{nc}

```
1:   sort clusters C by t_c in non-increasing order
2:   sort VMs X_n by σ_{X_n} in non-increasing order
3:   for all clusters C do
4:     for all unassigned VMs X_n do
5:       g_nc = 1          // assign X_n to C
6:       resource = Σ_i g_ic · μ_{X_i} + β · Σ_i Σ_j g_ic · g_jc · ρ_ij · σ_{X_i} · σ_{X_j}
7:       if resource > t_c then
8:         g_nc = 0        // cancel the assignment
9:       end if
10:    end for
11:  end for
```

Figure 3. Portfolio-based Resource Allocation Algorithm I (uncorrleated)

At the beginning, we sort clusters and *VMs* in non-increasing order (lines 1 and 2 in Figure 3.) Note that VMs are sorted by their standard deviations, not by their means. For each cluster the algorithm pre-assigns the *VM* with largest standard deviation among all unassigned *VMs* (lines 4 and 5.) It calculates the total amount of resource that the cluster is supposed to provide (line 6.) If it is greater than the capacity of the cluster, the assignment is canceled (line 7 and 8.) We repeat the above steps until either all VMs are

assigned or all clusters are full. Note that the equation in line 6 can be simplified as:

$$\sum_i g_{ic} \cdot \mu_{X_i} + \beta \cdot \sum_i g_{ic} \cdot {\sigma_{X_i}}^2 \quad (9)$$

This is because correlation coefficient ρ_{ij} is zero *for* $i \neq j$, that is, X_i's are uncorrelated.

2) Case2: correlated

Dealing with the correlated case requires very high amount of computing resources because complexity of correlation calculation is square of the number of *VMs*. Finding the optimal solution takes huge amount of time and makes the scheme is non-scalable. Hence, we present a heuristic approach to solve the problem.

The main idea is that a *VM* is assigned to a cluster one at a time and the best *VM* is selected in a greedy manner. Suppose there are *N VMs* in a cluster ($X = \sum_{n=1}^{N} X_n$) and we are going to assign a *VM* to the cluster. If a *VM* is deployed to the cluster, variance of the cluster becomes:

$${\sigma_{X+X_{N+1}}}^2 = {\sigma_X}^2 + 2\left(\sum_{n=1}^{N} \rho_{nN} \cdot \sigma_{X_n}\right)\sigma_{X_{N+1}} + {\sigma_{X_{N+1}}}^2$$
$$penalty := \sum_{n=1}^{N} \rho_{nN} \cdot \sigma_{X_n} \quad (10)$$

If there are a few candidates X_{N+1} for whom the standard deviations $\sigma_{X_{N+1}}$ are similar to each other, the *VM* with least *penalty* (10) is the best choice that gives rise to the smallest increase in standard deviation.

Portfolio-based Resource Allocation algorithm II (PBRA$_{corr}$) is nearly identical to *PBRA$_{uncorr}$* except for a few lines (lines 5 through 8 in Figure 4.) It considers the first **H** *VMs* as candidates for allocation. Because the list of *VMs* is sorted by standard deviations of the *VMs*, standard deviations of the candidates are similar to each other. Hence, *PBRA$_{corr}$* subsumes the correlated case by choosing the *VM* with least *penalty* (10) among the candidates.

Determining proper **H** is important for good performance of the algorithm. If **H** is 1, *PBRA$_{corr}$* becomes almost identical to *PBRA$_{uncorr}$*. On the other hand, *PBRA$_{corr}$* is very different from *PBRA$_{uncorr}$* if **H** is equal to *N*. Hence, it is important to choose a proper value for **H**. Intuitively, a large value for **H** is reasonable if many *VMs* are correlated. However, it is not so simple, and it will be discussed at the next section IV.

B. Cluster-level Resource Manager

The *cluster-level resource manager* deploys *VMs* that are given by the *cloud-level resource manager* on *PMs* of the cluster. Its job is conceptually equivalent to the job of the *cloud-level resource manager*: assign *VMs* to clusters/*PMs*. Thus, we use the same algorithms (*PBRA$_{corr}$* and *PBRA$_{uncorr}$*) for implementation of the *cluster-level resource manager:* the only difference is *clusters* in the algorithms are replaced by *PMs*. However, more elaborate algorithms e.g., minimum bin slack (MBS) heuristic [12] may also be used. Because the size of problems given to the *cluster-level resource manager* is bounded, we have more liberty to choose more complex (but yielding better results) algorithms.

The algorithms implicitly reduce the energy cost by utilizing the minimum number of *PMs*. The algorithms start from sorting its *PMs* by their capacity in non-increasing order, and assigning as many as *VMs* to each *PM*. Hence, the algorithms effectively solve (4).

There may be no feasible solution. If the manager cannot deploy some *VMs*, it lets the *cluster-level resource manager* know which *VMs* are not assigned. The *cluster-level resource manager* will reassign the *VMs* to other clusters. These steps are repeated until either all *VMs* are assigned or all clusters are full.

Portfolio-based Resource Allocation Algorithm II (PBRA$_{corr}$)

Inputs: t_c , μ_{X_n} , σ_{X_n} , $and\ \rho_{ij}$

Output: g_{nc}

```
1:  sort clusters C by t_c in non-increasing order
2:  sort VMs X_n by σ_{X_n} in non-increasing order
3:  for all clusters C do
4:    while true then
5:      for the first H unassigned VMs X_n do
6:          find X_n of which penalty is minimal (10)
7:          where penalty = Σ_i g_ic · ρ_in · σ_{X_i}
8:      end for
9:      g_nc = 1          // assign X_n to C
10:     resource = Σ_i g_ic · μ_{X_i} + β · Σ_i Σ_j g_ic · g_jc · ρ_ij · σ_{X_i} · σ_{X_j}
11:     if resource > t_c then
12:       g_nc = 0        // cancel the assignment
13:       break           // break while loop
14:     end if
15:   end while
16: end for
```

Figure 4. Portfolio-based Resource Allocation Algorithm II (corrleated)

IV. Simulation Results

A. Simulation setup

For the simulation we needed the following data: i) capacity of *PMs*, ii) list of *PMs* in clusters, iii) means and standard deviations for resource requirements of *VMs*, and iv) correlation coefficients among various *VMs*. We generate the data randomly as follows:

- Number of *VMs (N)*, *PMs (M)*, and clusters *(C)*
- Resource requirements of *VMs*: mean and standard deviation of μ_{X_n}, σ_{X_n} $(\mu_{\mu_{X_n}}, \sigma_{\mu_{X_n}}, \mu_{\sigma_{X_n}}, and\ \sigma_{\sigma_{X_n}})$
- Capacity of *PMs*: mean and standard deviation of total resource amount
- C_{ratio} : (# of correlated *VMs)* / (# of total *VMs)*
- C_{size}: # of correlated *VMs* in a group

It is randomly decided which *PMs* are placed in which clusters. Because we have heterogeneous *VMs*, their means (μ_{X_i}) and standard deviations (σ_{X_i}) are randomly generated from the given information ($\mu_{\mu_{X_n}}, \sigma_{\mu_{X_n}}, \mu_{\sigma_{X_n}}, and\ \sigma_{\sigma_{X_n}}$.) A correlation matrix is created based on C_{ratio} and C_{size}. Note that $C_{ratio} = 0$ means all *VMs* are uncorrelated whereas $C_{ratio} = 0.5$ means half of *VMs* are correlated. Furthermore, if, for example, we have 10 correlated *VMs* and $C_{size} = 5$, there is going to be two groups, and each of the two groups has five *VMs* in it. *VMs* in a group are correlated only with the *VMs* in the same group.

Making a valid correlation coefficient matrix $[\rho_{ij}]$ is important. We use the *hypersphere decomposition* [13] method, which is a relatively simple method for generating a valid correlation matrix.

B. Comparing algorithms

To assess the quality of solutions generated by the proposed algorithms, we compare our solution with some other well-known algorithms:

- *SA*–use simulated annealing [14] algorithm. It does not guarantee to find the global optimal point, but it finds a near-optimal solution given a slow enough cooling schedule. This method may generate different solutions each time, so we run the SA six times and pick the best result. SA results may be treated as the result to beat.
- *random*–do assignments randomly. If a solution is worse than *random*, it means quality of the solution is quite poor. Lower gap between results of *SA* and *random* means there is less gain to be had from any optimization. We run *random* algorithm ten times and report the average of these runs.
- *FFD* and *BFD*–use *first fit decreasing* and *best fit decreasing* algorithms. Both are well-known and commonly used heuristics for solving the bin-packing problem. *FFD* sorts X_n by its resource requirements $(\mu_X + \beta\sigma_X)$ in a non-increasing order, and assigns X_n to the first cluster that is available. *BFD* is similar to *FFD*, but it assigns X_n to the cluster with the minimum remaining capacity among the cluster of which capacity is greater than $(\mu_X + \beta\sigma_X)$. None of these two heuristics takes advantage of the portfolio effect. More precisely, they treat the resource requirements of *VMs* not as *RVs* but as constant values equal to $\mu_X + \beta\sigma_X$. They simply add or subtract these constant values in order to calculate the required resource. This is equivalent to assuming that all *VMs* are fully correlated ($\forall i, j\ \rho_{ij} = 1$), which is a very conservative approach.
- FFD_{pf} *and* BFD_{pf} – They are identical to *FFD* and *BFD* except that they consider the portfolio effect.

C. Cloud-level Resource Manager

The objective of the *cloud-level resource manager* is to minimize the sum of standard deviations of clusters (6), which will be called *cost* in this section. Lower cost means better quality of the solution. There are 300 *VMs (N = 300)*, which is relatively small number of *VMs*. This is because *SA* requires huge amount of time for large problem sizes. We first investigate the quality of the algorithms by comparing with the near optimal (*SA's results)* for small problem size, and then compare results of the algorithms except *SA* for larger number of *VMs* in section D.

Figure 5. depicts normalized costs of the algorithms. For a fair comparison, we generate eight different test cases based on the same data and run simulations. Note that even based on the same data, the results can be different because the data is randomly generated in each case (see IV-A.) Both $PBRA_{corr}$ and $PBRA_{uncorr}$ produce excellent results that are even better than the results of *SA (*Figure 5. *a.*) Note that results of $PBRA_{corr}$ and $PBRA_{uncorr}$ are almost identical when *VMs* are uncorrelated ($C_{ratio} = 0$.) This is because *penalty* (10) is always zero for **H** candidate *VMs*. When half of the *VMs* are correlated ($C_{ratio} = 0.5$), $PBRA_{corr}$ produces better result (less cost) than $PBRA_{uncorr}$, and is still better than *SA* for six test cases (Figure 5. *b.*)

Quality of the algorithms can be affected by C_{ratio} and C_{size}, thus we run multiple simulations for different combinations of C_{ratio} and C_{size}. Comparison among the algorithms under different C_{size} values is shown in Figure 6. C_{ratio} is 0.9 for all cases, which means 270 *VMs* out of 300 *VMs* are correlated. For larger C_{size}, the cost difference between *random* and *SA* decreases, which means cost reduction from optimization decreases. This is because if C_{size} is sufficiently enough, we can easily put positively correlated *VM*s into different clusters; hence the cost will decrease a lot. On the other hand, if too many *VMs* are correlated with each other, it is difficult to avoid assigning correlated *VMs* to the same *PM*. Hence, there are small improvements obtained from the algorithms.

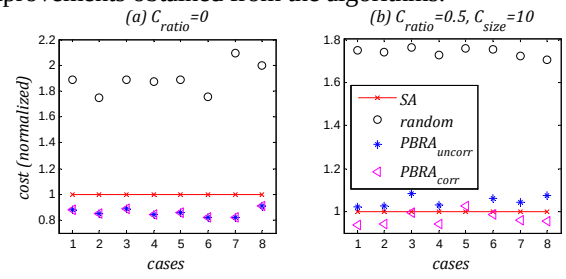

Figure 5. Comparison among algorithms (N = 300, H=30)

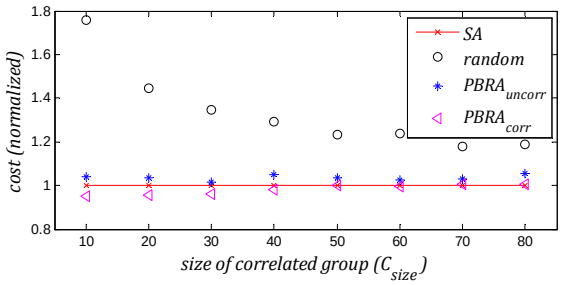

Figure 6. Size of correlated group vs. cost (C_{ratio}= 0.9, N=300, H =30)

If C_{size} is small enough, the quality of $PBRA_{corr}$ and $PBRA_{uncorr}$ is not influenced much by C_{ratio} (Figure 7.) $PBRA_{corr}$ produces better results than $PBRA_{uncorr}$ and cost of both algorithms increases a little bit for higher C_{ratio}. This shows that the proposed algorithms produce high quality solutions if size of correlated group is small enough. The statement is also valid even if there are many correlated *VMs*.

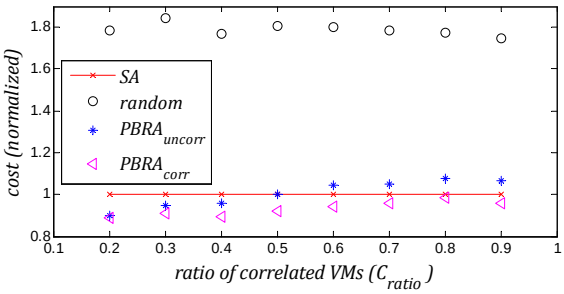

Figure 7. Ratio of correlated *VMs* vs. cost (C_{size}= 10, N=300, and H=30)

As mentioned at the previous section, choosing a proper **H** is important for higher quality of $PBRA_{corr}$ (Figure 4.) If

H is 1, $PBRA_{corr}$ is nearly identical to $PBRA_{uncorr}$. On the other hand, $PBRA_{corr}$ becomes very different from $PBRA_{uncorr}$ when **H** is *N* (the number of *VMs*.) Note that there are two reasons for cost reduction: the first one comes from the *proposition* presented in section *III.A* and the second is due to avoiding the assignment of highly correlated *VMs* to the same cluster. The best **H** minimizing the cost is decided depending on which source of cost reduction is dominant: when the first source (*proposition*) is dominant, smaller **H** is better choice and vice versa.

The results of $PBRA_{corr}$ for variety of **H** values are reported in Figure 8. It is expected that the best **H** is not the same under different patterns of correlation: How many *VMs* are correlated and how big the group of correlation is? We categorize correlation patterns into four cases and run simulations for each case:

- Mid $C_{ratio}(0.5)$, small $C_{size}(10)$ – Figure 8. a
- Mid $C_{ratio}(0.5)$, large $C_{size}(100)$ – Figure 8. b
- High $C_{ratio}(0.9)$, small $C_{size}(10)$ – Figure 8. c
- High $C_{ratio}(0.9)$, large $C_{size}(100)$ – Figure 8. d

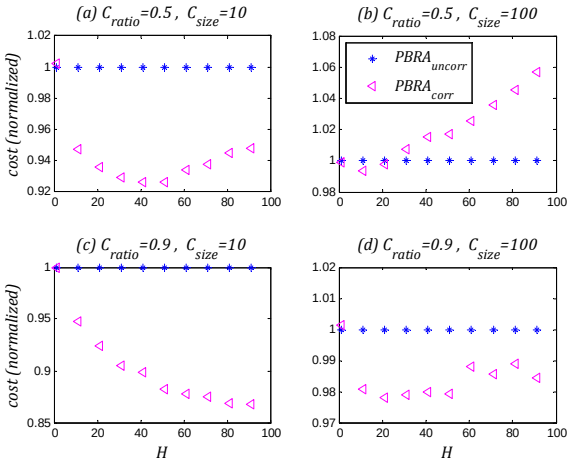

Figure 8. H vs. cost for four cases (N=300)

C_{ratio} = 0.5 means that half of *VMs* are correlated; hence there are two sources of cost reduction. Depending on which source is dominant, the best **H** value is chosen. For small C_{size} (Figure 8. a), it is easier to avoid putting highly correlated *VMs* into the same cluster. In addition, bigger **H** means more candidate *VMs* are investigated; hence, cost saving from the second source becomes greater. However, at the same time, cost saving from the first source (*proposition*) keeps decreasing for bigger **H**. Thus, the best **H** is around 50 (Figure 8. a.) How about bigger C_{size} with the same C_{ratio}? (Figure 8. b) As shown in Figure 6. , the amount of cost saving from putting correlated *VMs* in separate clusters becomes smaller for bigger C_{size}. Thus, cost saving from the first source is dominant. Because cost saving from the second source is rather small, the best **H** is around 10. When C_{ratio} is close to 1.0, that is most of all *VMs* are correlated, it is highly probable that the second source of cost saving is dominant (Figure 8. c and d) When C_{size} is small (Figure 8. c), there is more cost saving as **H** increases, and amount of cost saving of this case is largest among the four cases. When both C_{ratio} and C_{size} are large, $PBRA_{corr}$ always produces better result, but there is neither big difference in cost nor a clear relationship between **H** and cost saving (Figure 8. d.)

Based on the above discussion, we get some insight for choosing the proper **H**. First of all, relatively bigger **H** is promising for the case of small C_{size} (Figure 8. a and b.) For the last case (Figure 8. d), it does not matter if we choose any value of **H**, but higher **H** increases the computational overhead. Hence, we need to pick as small **H** as possible. Consequently, small **H** is good for the case of large C_{size} (Figure 8. b and d.)

D. Overall performance comparisons

We have shown that the proposed algorithms are simple and work well for cluster-level resource allocation. Their results are very close to or even better than the *SA* results. What *cluster-level resource managers* do is very similar to what the *cloud-level resource manager* does: assign *VMs* to *PMs*. Hence, we apply the same algorithms ($PBRA_{uncorr}$ and $PBRA_{corr}$) for *cluster-level resource managers*. However, the high quality of each level does not necessarily guarantee the high quality of the overall solution. In this section, we will compare the quality of the final solution generated by different algorithms and verify that the proposed algorithms produce better quality solutions. In addition to this, we run the simulation for a large number of *VMs*, and show if the proposed scheme is scalable or not.

Comparison among the algorithms with different numbers of *VMs* is reported in Figure 9. It is seen that costs of *FFD* and *BFD* are much larger than those of all others. This is because *FFD* and *BFD* do not consider the portfolio effect and thus tend to overbook the resource. A reason for this big difference is that σ_{X_n} is set to be larger than μ_{X_n} in this study, which magnifies the portfolio effect. Nevertheless, the experiment suggests that the portfolio effect has to be considered. Figure 9. (a-2 and b-2) plots the same results as Figure 9. (a-1 and b-1) except that results of *BFD* and *FFD* are excluded for better visual clarity. Because $PBRA_{uncorr}$ and $PBRA_{corr}$ are nearly the same for uncorrelated *VMs*, their results look almost identical. The difference in cost among the four algorithms is very small (less than 2 %) (Figure 9. (a-2).) Interestingly, $PBRA_{corr}$ produces the best result for the other case where many *VMs* are correlated (Figure 9. b-2.) This result shows $PBRA_{corr}$ outperforms all other heuristics when many *VMs* are correlated (up to 10% cost reduction compared to *FFD*.)

One of the most important features of a solution for clouding computing is scalability. Hence, the relationship between execution time of the algorithms and problem size is very important. Execution time of the proposed algorithm is defined as follow:

$$T_{exe} = T_{cloud} + \max(T_{cluster})$$

where T_{cloud} and $T_{cluster}$ are execution time of the *cloud-level resource manager* and *cluster-level resource managers*, respectively. The *cluster-level resource managers* are running in parallel; hence, their longest execution time is used. As shown in Figure 10. , the trends of two cases (correlated and uncorrelated) are very similar: execution time

of *FFD* and *BFD* is the smallest while the execution time of $PBRA_{corr}$ is the largest. Execution times of the other three algorithms (FFD_{pf}, BFD_{pf}, and $PBRA_{uncorr}$) are similar. Note that *y* axis is plotted with a logarithmic scale; thus, the plots clearly show the proposed algorithms are scalable. In addition to this, the execution time of $PBRA_{corr}$ becomes close to those of the other portfolio-based algorithms (FFD_{pf}, BFD_{pf}, and $PBRA_{uncorr}$) as the number of *VMs* increases.

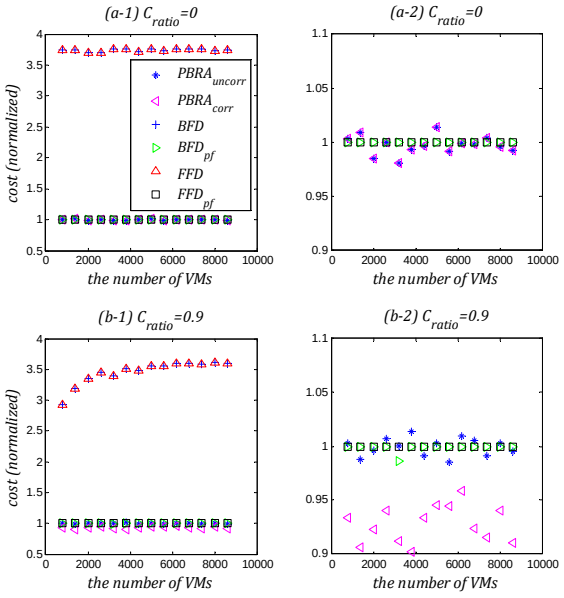

Figure 9. Quality comparison among algorithms (C_{ratio}= 0 and 0.9)

The results show that the proposed algorithms ($PBRA_{corr}$ and $PBRA_{uncorr}$) outperform the well-known heuristic algorithms (*FFD* and *BFD.)* $PBRA_{corr}$ produces the best results among the algorithms for correlated *VMs*. The important fact is that the proposed scheme is distributed whereas other algorithms are centralized. Because the problem size of *cluster-level resource manager* is much smaller, we have the opportunity to apply more sophisticated algorithms with less concern about their computational complexity. Hence, the fact that solution of the proposed scheme is better than the existing heuristics is meaningful even when the difference between qualities of solutions is not so big.

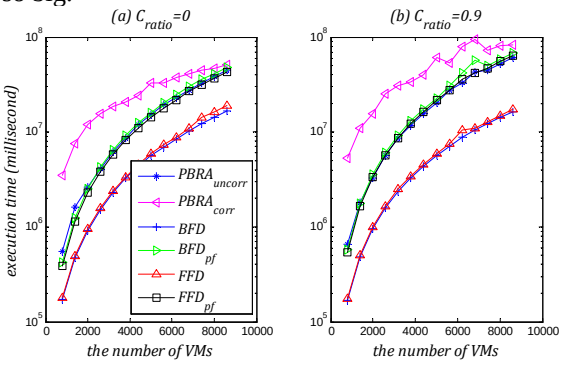

Figure 10. Running time comparison among algorithms (C_{ratio}= 0 and 0.9)

V. Conclusions

With increasing energy cost of cloud computing systems, necessity of energy aware resource management techniques has been growing. This paper proposed a hierarchical resource management scheme which is scalable and produces high quality solutions. Resource requirements were modeled as random variables and correlation among the RVs were considered. The proposed solution outperforms well-known heuristic algorithms when *VMs* are correlated with each other. The solution achieves up to 10% cost reduction compared to *FFD* and *BFD*.

Acknowledgment

This work is sponsored by a grant from the National Science Foundation.

References

[1] A. Gandhi, *et al.* Optimal power allocation in server farms. in *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*.

[2] R. Raghavendra, *et al.* No "power" struggles: coordinated multi-level power management for the data center. in *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*.

[3] S. Srikantaiah, *et al.* Energy aware consolidation for cloud computing. in *Proceedings of the 2008 conference on Power aware computing and systems*.

[4] M. Stillwell, *et al.* Resource Allocation Using Virtual Clusters. in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*.

[5] D. Wilcox, *et al.* Probabilistic Virtual Machine Assignment. in *CLOUD COMPUTING 2010, The First International Conference on Cloud Computing, GRIDs, and Virtualization.*

[6] P. Billingsley, *Probability and Measure*2012: John Wiley & Sons.

[7] A. Rudd and H.K. Clasing, *Modern portfolio theory: the principles of investment management*1988: Andrew Rudd.

[8] M. Pedram and I. Hwang. Power and Performance Modeling in a Virtualized Server System. in *Proceedings of the 2010 39th International Conference on Parallel Processing Workshops*.

[9] C. Clark, *et al.* Live migration of virtual machines. in *Proceedings of the 2nd conference on Symposium on Networked Systems Design \& Implementation - Volume 2*.

[10] P. Barham, *et al.*, *Xen and the art of virtualization*, in *Proceedings of the nineteenth ACM symposium on Operating systems principles*ACM: Bolton Landing, NY, USA.

[11] B. Botelho. Virtual machines per server: A viable metric for hardware selection? . 2008; Available from: http://itknowledgeexchange.techtarget.com/server-farm/virtual-machines-per-server-a-viable-metric-for-hardware-selection/.

[12] J.N.D. Gupta and J.C. Ho, A new heuristic algorithm for the one-dimensional bin-packing problem. *Production Planning and Control*, 1999. **10**(6): p. 598-603.

[13] R. Rebonato and P. Jäckel, The most general methodology for creating a valid correlation matrix for risk management and option pricing purposes. *Journal of Risk*, 2000.

[14] S. Kirkpatrick, *et al.*, Optimization by Simulated Annealing. *Science*, 1983. **220**(13): p. 10