Application-aware Workload Consolidation to Minimize both Energy Consumption and Network Load in Cloud Environments

Nikos Tziritas[1], Cheng-Zhong Xu[1,4], Thanasis Loukopoulos[2], Samee Ullah Khan[3], Zhibin Yu[1]
[1]Chinese Academy of Sciences, China,{nikolaos, cz.xu, zb.yu}@siat.ac.cn
[2]Technological Educational Institute of Lamia, luke@teilam.gr
[3]North Dakota State University, USA samee.khan@ndsu.edu
[4]Wayne State University, USA, czxu@wayne.edu

Abstract— **In this paper we tackle the problem of virtual machine (VM) placement onto physical servers to jointly optimize two objective functions. The first objective is to minimize the total energy spent within a cloud due to the servers that are commissioned to satisfy the computational demands of VMs. The second objective is to minimize the total network overhead incurred due to: (a) communicational dependencies between VMs, and (b) the VM migrations performed for the transition from an old assignment scheme to a new one. We study different methodologies for solving the aforementioned problem. The first approach is based on VM packing algorithms that optimize the above objective functions separately, reaching a single solution. The other approach is to tackle simultaneously the two optimization targets and define a set of non-dominating solutions. Performance evaluation using simulation experiments reveals interesting trade-offs between energy consumption and network load.**

Keywords- workload consolidation; virtual machine placement, network load minimization, Pareto optimization

I. INTRODUCTION

In the last few years, the cloud computing paradigm has received considerable attention from academia and industry because of the promised opportunities. Nevertheless, the rising energy cost has become a burden for cloud environments. By reducing the energy spent within a cloud system, a cloud provider can decrease the Total Cost of Ownership (TCO) and subsequently increase the Return on Investment (ROI) of cloud infrastructures. For the above reason, energy reduction has become an ardent desire for cloud providers. Such reduction can be achieved by leveraging virtualization technology, which is one of the most prominent features of cloud environments to increase system utilization. Part of ongoing research focuses on workload consolidation to a minimum number of physical machines and switching off idle machines. It has been shown that the aforementioned technique is a key factor to drastically reduce the energy spent within a cloud system.

However, any workload consolidation (if not done judiciously) may adversely affect the performance of running applications, due to the cost of VM migrations. Specifically, [4] reports that the performance of a running application is likely to be negatively impacted during the migration process because of the overhead caused by successive iterations of memory pre-copying. Another reason is that VM migrations incur extra network load on their own [10]. Thus, extra care must be taken regarding which VMs to migrate where so as not to overburden the network with extra communication overhead due to the communication dependencies between virtual machines [18].

In this work, we aim to solve the following problem: *Assuming that a set of VMs is already deployed within a cloud, reassign the VMs to physical servers so as to both minimize the total energy consumption due to server computations and the total network overhead incurred by: **(a)** the communicational dependencies between virtual machines, and **(b)** the migrations performed.*

In this paper, we discuss various optimization algorithms to tackle the aforementioned problem that fall into two main categories. In the first category we propose algorithms that attempt to optimize one of the two criteria either this being energy consumption or network overhead. In the second category, we tackle the problem as Pareto optimization [15] giving an algorithm that identifies a set of non-dominating solutions. We would like to note that the algorithm in the second category uses as components the single function optimization algorithms of the first category. For this reason, we experimented with various VM packing algorithms, especially with the ones that aim at minimizing energy consumption. The contributions of our work include the following: **(a)** we tackle the optimization problem of jointly minimizing the energy consumption and the total network overhead incurred in a cloud environment; **(b)** we propose algorithms that define either single solutions or a gamma of non-dominating solutions and **(c)** we evaluate the performance of the proposed algorithms through simulation experiments.

This paper is structured as follows. Section II reports the related work. In Section III we describe the system model and formulate the optimization problem. The proposed algorithms are presented in Section IV and evaluated in Section V. Finally, Section VI concludes the paper.

II. RELATED WORK

Large data center operators (e.g., Google) try to find ways to decrease the operational power of data centers. Ref [6] claims that the above can be achieved by keeping system utilization at a maximum level within a set of active machines operating at the lowest power levels. In [1], the authors adopt the aforementioned policy and propose energy-aware heuristics that consolidate VMs onto the minimum number of servers. At the same time, the authors

also explore solutions to minimize the SLA (Service Level Agreement) violations experienced within a cloud due to the workload consolidation strategies. The authors propose a workload consolidation algorithm that modifies the Best Fit Decreasing (BFD) approach [16]. The proposed algorithm is named Modified Best Fit Decreasing (MBFD) and it works as follows. It sorts all VMs in decreasing order of their CPU demands, and allocates each VM to a host that provides the least increase of power consumption due to this allocation. The same problem is also tackled in [12] using Limited Lookahead Control (LLC). A Kalman filter is employed to estimate the number of future requests to predict the future state of the system and perform necessary reallocations. Ref [2] proposes adaptive heuristics to address the aforementioned problem in an online manner. The authors also conduct a competitive analysis and prove competitive ratios of optimal online deterministic algorithms for the single VM migration and dynamic VM consolidation problems. Gandhi et al. [8] address the problem of allocating an available power budget among servers in a virtualized heterogeneous server farm, while minimizing the mean response time. Mertzios et al. [17] have considered the problem of minimizing the power consumption of a set of machines which can be measured by the amount of time the machines are switched on and processing jobs. Specifically, the above work tries to consolidate jobs whose processing times overlap, such that to minimize the busy time of machines. Another work on workload consolidation is that of [7], where the authors model the problem as an instance of the multi-dimensional bin-packing (MDBP) problem and design a distributed algorithm based on the Ant Colony Optimization. Last, Curino et al. [5] formalize the problem of consolidating multiple databases on fewer machines. They analyze the load characteristics of multiple dedicated database servers and pack their workloads into fewer physical machines, while they achieve near-zero performance degradation.

Network load may play a crucial role in performance degradation of communication-intensive applications [18]. Sonnek *et al.* [18] considers VM migrations for minimizing the total communication overhead and improving in that way the application performance. In the above work, a *distributed bartering algorithm* was proposed that allows VMs to negotiate the placement on a physical server that is closer to the data required. The above problem is also tackled in [19], where the authors propose a fully distributed algorithm that results always in optimal placements in tree-structured networks. The aforementioned algorithm is based on minimum-cut maximum-flow techniques. Ref [3] considers the online problem of minimizing the total network load between a service represented by one or more VMs and the clients requesting the corresponding service. The authors solve the problem by proposing an online strategy that migrates VMs towards the center of their communicational gravity. Their strategy is also accompanied with its competitive analysis.

In the following text, we mention the most relevant works we found in the literature in regards to the mechanisms of live virtual machine migrations and their effects on clouds. We must note that such works are complementary to our work, since our algorithm could adopt such mechanisms to implement virtual machine migrations in and of themselves. Specifically, the authors in [9] study the problem of live virtual machine migrations across a Gigabit LAN. They facilitate the use of post-copy with adaptive pre-paging to eliminate all duplicate page transmissions. They also eliminate the transfer of free memory pages in both migration schemes through a dynamic self-ballooning mechanism. Ref. [11] takes advantage of the Remote Direct Memory Access (RDMA) to improve the efficiency of VM migration. Particularly, their methodology reduces the migration overhead: up to 80% on total migration time and up to 77% on application observed time. In [14] the authors combine the virtual machine migration with the process migration for running high performance computing applications on clusters and grids. They presented a novel approach, by which it is possible to migrate groups among different clusters. Another work focused on live virtual machine migrations is that of [13] which provides a synchronization algorithm to orchestrate the migrating VM states. They showed through experimental evaluation that their algorithm can drastically reduce migration overheads compared with pre-copy algorithm. Finally, Ref. [20] evaluated the effects of live migration of virtual machines performance on the performance of applications running inside Xen VMs. The authors of the paper showed that, in most cases, migration overhead is acceptable but cannot be disregarded, especially in systems where availability and responsiveness are governed by strict Service Level Agreements.

To the best of our knowledge, none of the above papers consider the problem of both minimizing the energy consumption and the total network load incurred within a cloud due to **(a)** VM communicational dependencies and **(b)** VM migrations. We compare our solutions with BFD and MBFD, since these algorithms are more closely to our work.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. System Model

We assume that our network is organized in a three-level hierarchy. The first level comprises of a number of clusters, with each of them consisting of a number of racks (2^{nd} level) that each of them contains a number of physical servers (3^{rd} level). The communication between physical servers takes place through three different ways based on servers' location. Specifically: **(a)** servers within a rack communicate with each other through Ethernet; **(b)** servers that are located on the same cluster but on different racks communicate also through Ethernet; **(c)** the communication between two servers that are located on different clusters (geographically dispersed) is effected through one or more bottleneck-links connecting the corresponding clusters (inter-cluster communication). An example of networked clusters is given in Figure 1. Although it is not depicted in the figure, servers

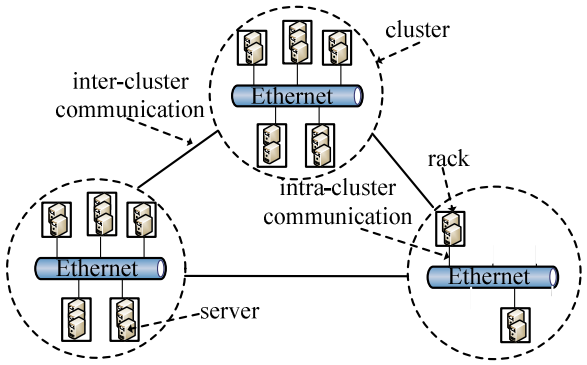

Figure 1. Network model

belonging to the same rack communicate with each other through Ethernet.

Let s_x and S be the x^{th} physical server and the total number of servers within the cloud, respectively. Variable d_{xv} captures the cost of exchanging one abstract data unit between s_x and s_v ($d_{xx} = 0$). Let J be the number of jobs comprising an application to be executed within the cloud, and j_i define the i^{th} such job. The communication dependencies between jobs, under a time interval $[t_1, t_2]$, are encoded by a $J \times J$ matrix denoted by W, with $w_{xy}^{t_1,t_2}$ representing the amount of data (in bytes) exchanged between j_x and j_y ($w_{xy}^{t_1,t_2} = w_{yx}^{t_1,t_2}$). The number of virtual machines is defined by V, while the k^{th} virtual machine within the system is denoted by v_k. To capture the communication dependencies between two virtual machines v_k and v_m, we sum the data exchanged between the set of jobs executed on v_i and the set of jobs executed on v_j. The total data exchanged between v_k and v_m, under a time interval $[t_1, t_2]$, is encoded by an $V \times V$ matrix denoted by $c_{mk}^{t_1,t_2}$ ($c_{mk}^{t_1,t_2} = c_{km}^{t_1,t_2}$).

B. Problem Formulation

To provide a rigorous mathematical formulation of the problem we need to introduce the following. The placement of VMs onto physical servers is captured by an $S \times V$ matrix denoted by F, with F^{old} and F^{new} defining an old and new placement, respectively. f_{kx} denotes whether v_k is hosted by s_x or otherwise. Specifically, f_{kx} equals one if v_k is hosted by s_x, otherwise f_{kx} equals zero. The computing capacity of a server s_x is represented by CPU_x, while the computing capacity required from a virtual machine v_k to be executed correctly for a specified point in time t is defined by $Req\_CPU_k(t)$. Let $U_x(F, t)$ represent the utilization of s_x as a function of a placement F and time, which is calculated through Eq. 1. Eq. 2 states that the total utilization of a server cannot be greater than one. Let also $p_{\max}^x$ denote the maximum power consumed by a server s_x when the latter is fully utilized. Given a placement F and a point in time t, then the power consumption of a server s_x is represented by $P_x(F, t)$. Recent studies [1], [2] show that in average an idle server consumes approximately 70% of the power consumed by the respective server when it is fully utilized. The above is captured by Eq. 3, which models the power consumption of a server s_x. The total energy consumed by all of the servers within the system for a given placement F and a time interval $[t_1, t_2]$, is captured by Eq. 4. Eq. 5 records the total network load incurred within the system under a placement F and a time interval $[t_1, t_2]$. Last, the implementation cost for the transition from a placement F^{old} into a placement F^{new} is given by Eq. 6.

$$U_x(F,t) = \sum_{k=1}^{V} f_{xk} \times (\mathrm{Re}q\_CPU_k(t) / CPU_x) \quad \text{Eq. 1}$$

$$\sum_{k=1}^{V} f_{xk} \times (\mathrm{Re}q\_CPU_k / CPU_x) \le 1 \quad \text{Eq. 2}$$

$$P_x(F,t) = 0.7P_{\max}^x + 0.3P_{\max}^x \times U_x(F,t) \quad \text{Eq. 3}$$

$$Energy(F,t_1,t_2) = \sum_{x=1}^{S} \int_{t1}^{t2} P_x((F,t)) \quad \text{Eq. 4}$$

$$comm(F,t_1,t_2) = \sum_{m=1}^{V}\sum_{k=1}^{V}\sum_{x=1}^{S}\sum_{y=1}^{S} c_{mk}^{t_1,t_2} f_{mx} f_{ky} d_{xy} \quad \text{Eq. 5}$$

$$\mathrm{Impl}(F^{old}, F^{new}) = \sum_{k=1}^{V}\sum_{x=1}^{S}\sum_{y=1}^{S} S(v_k) f_{kx}^{old} f_{ky}^{new} d_{xy} \quad \text{Eq. 6}$$

The problem is formally stated as: *Given a number of virtual machines with communicational inter-dependencies due to inter-communicating jobs hosted on them, try to find a feasible placement F* of these VMs onto physical machines such that to minimize:* ***(a)*** *The energy spent due to the active physical servers;* ***(b)*** *The total network overhead due to the communication inter-dependencies between VMs and the VM migrations performed to implement the transition from the old placement into the new one.*

IV. ALGORITHMS

In this section, we illustrate the proposed algorithms starting from the ones using single criterion optimization and proceeding with the rest.

A. Low Perturbation Bin Packing Algorithm (LPBP)

This algorithm follows a lazy approach on improving VM placement by performing only the most beneficial migrations. The rationale of the algorithm is that by proclaiming a slow transition between the existing VM-server assignment scheme and a new one, the number of migrations is kept low.

The algorithm keeps the servers in a list sorted in descending order according to energy consumption and works as follows. First, it attempts to offload the most energy-consuming server, by migrating some (even all) of its VMs to the least energy-consuming server(s). In doing so, it considers VMs in a decreasing order of their computing requirements. In case a feasible (no computing capacity violations) assignment scheme that completely offloads the server is reached, the relevant migrations are performed. Otherwise, no migration is done, the server under consideration is removed from the server list and the algorithm proceeds with the next one, until the list is empty.

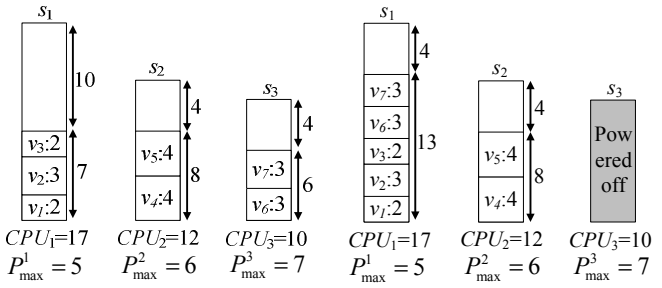

Figure 2. Initial placement

Figure 3. LPBP's placement

We illustrate the functionality of the algorithm through an example. Consider the initial placement of seven virtual machines onto three servers shown by Figure 2. Rectangle size in the example represents server computational capacity and VM computational needs. Let F^{init} represent the placement shown in Figure 2. Applying Eq.3, server power consumptions are calculated as follows: $P_1(F^{\text{init}}, t) = 4.12$, $P_2(F^{\text{init}}, t) = 5.4$ and $P_3(F^{\text{init}}, t) = 6.16$, giving a total energy spent per abstract time unit of 15.68.

Figure 3 shows the resulting placement after applying LPBP (F^{LPBP}). Recall, that LPBP offloads most energy-consuming server (s_3) towards the least one(s_1). Therefore, it will attempt to migrate both v_6 and v_7 from s_3 towards s_1. Since these migrations do not violate the computational capacity constraint at s_1 they will be performed and s_3 will be powered off. Next, LPBP proceeds with the second most energy-consuming server and attempts to migrate v_5 and v_4 from s_2 to s_1. However, this cannot be done due to the computational constraint at s_1 and the final placement is reached. Server energy consumption in the resulting placement is: $P_1(F^{\text{LPBP}}, t) = 4.65$, $P_2(F^{\text{LPBP}}, t) = 5.4$, $P_3(F^{\text{LPBP}}, t) = 0$, giving a total energy spent per abstract time unit of 10.05.

In the sequel, we present LPBP', which works in a little bit different way compared to LPBP. Specifically, LPBP' orders energy-consuming servers in a decreasing manner according to the ratio between their power consumption and computing capacity. Recall that LPBP performs server ordering based only on their energy consumption.

B. Power- and Computing capacity- Aware Best Fit Decreasing (PCA-BFD)

BFD is a well-known bin packing algorithm, which offers good performance [16]. The algorithm works as follows. Initially, it considers all servers as empty/unused. Then it assigns each VM to the used server of maximum free computing capacity. The VMs are considered in a decreasing order according to computing capacity requirements. If the chosen VM doesn't fit in any of the used servers (or no such exist), it is placed to the previously unused server of minimum computing capacity.

To illustrate the behavior of BFD consider the example of Figure 2. First, BFD selects the most demanding VM to place (v_4) and places it to the unused server of minimum capacity (s_3). Next, v_5 is considered and since s_3 has enough remaining capacity it is hosted there. Similarly, v_7, v_6, and v_2 will be hosted at s_2 since they don't fit in s_3 and s_2 is the next unused server that must be filled. Then assuming v_3 and v_1 are considered in this order, v_3 will be placed at s_2 (the non-empty server of maximum remaining computing capacity) and subsequently, v_1 at s_3, resulting in the placement of Figure 4 whereby: $P_1(F^{\text{BFD}}, t) = 0$, $P_2(F^{\text{BFD}}, t) = 5.85$, $P_3(F^{\text{BFD}}, t) = 7$. Consequently, BFD results in an assignment where the total energy spent per time unit is 12.85, inferior to LPBP.

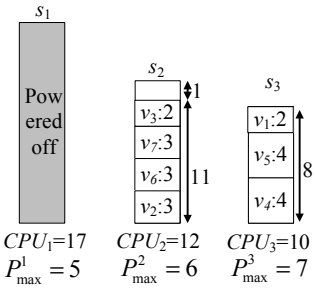

Figure 4. BFD's placement

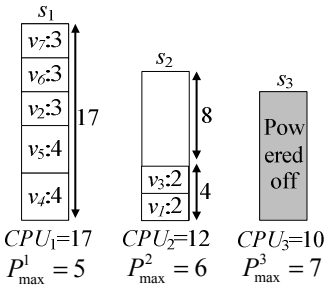

Figure 5. PCA-BFD's placement

Notice, that the primary reason for the performance difference between LPBP and BFD in the example, is the decision of BFD to choose the unused server of smallest capacity (s_3), even if it is more energy consuming than the rest. Therefore, we propose a modified algorithm called Energy Aware Best Fit Decreasing (PCA-BFD). PCA-BFD works in the same way as BFD, with the only difference being that the former opens servers in an increasing order according to the ratio $p^x_{\max}/\text{CPU}_x$. Revisiting the previous example, PCA-BFD will first fill s_1 placing v_2, v_4, v_5, v_6, and v_7.Then it will consider s_2 and assign on it v_3 and v_1 reaching a final placement whereby energy consumption is as follows: $P_1(F^{\text{PCA-BFD}}, t) = 5$, $P_2(F^{\text{PCA-BFD}}, t) = 4.8$, $P_3(F^{\text{PCA-BFD}}, t) = 0$, for a total of 9.8.

C. Communication-Aware Migration Algorithm (CAM)

Unlike the previous algorithms, CAM tries to remap VMs onto servers such that to minimize the total network load, instead of minimizing the energy consumption. The remapping phase takes place in an iterative way. Specifically, the algorithm considers the migration of each VM towards the set of servers hosting VMs that communicate with the respective VM. For each v_k there are at most $deg(v_k)$ candidate destinations, with $deg(v_k)$ denoting the number of adjacent VMs to v_k. Note that we have exactly $deg(v_k)$ destinations if and only if there are no co-locations among the VMs that are adjacent to v_k. For each candidate *feasible* migration of a VM v_k, CAM calculates its benefit in terms of the network load reduction achieved within the system. By saying feasible migration, we mean that the destination server has sufficient computing capacity to cover the computing demands of the VM to be migrated. We must also note that when calculating the migration benefit, CAM takes also into account the network load incurred due to the corresponding migration. To calculate the migration benefit of a VM v_k, we first need to declare the following. **(a)** Given a time interval $[t_1, t_2]$ and a

placement f, then the total network load incurred by v_k within the system is given by Eq. 7. **(b)** Given a migration of v_k ($f^{old} \rightarrow f^{new}$), then the network load incurred by the respective migration is given by Eq. 8. Therefore, the benefit of migrating a VM v_k ($f^{old} \rightarrow f^{new}$) is given by (see Eq. 9) subtracting from the total network load incurred by v_k when v_k is hosted by the old server (F^{old} mapping): **(i)** the total network load incurred by v_k when v_k is hosted by the new server (F^{new} mapping), and **(ii)** the network load incurred due to the corresponding migration. After calculating all the potential migration benefits of a given VM, we choose the one that maximizes Eq. 9. However, before deciding to implement a migration, we first need to calculate the benefit of migrating each VM to its respective best candidate server. In the sequel, we choose to migrate the VM that yields the largest migration benefit. The above procedure takes place in an iterative way, until there is no feasible beneficial migration.

$$L(v_k, F, t_1, t_2) = \sum_{m=1}^{V} \sum_{x=1}^{S} \sum_{y=1}^{S} c_{km}^{t_1,t_2} f_{kx} f_{my} d_{xy} \quad \text{Eq. 7}$$

$$\mathrm{MC}(v_k, F^{old}, F^{new}) = \sum_{x=1}^{S} \sum_{y=1}^{S} S(v_k) f_{kx}^{old} f_{ky}^{new} d_{xy} \quad \text{Eq. 8}$$

$$B(v_k, F^{old}, F^{new}, t_1, t_2) = L(v_k, F^{old}, t_1, t_2) - L(v_k, F^{new}, t_1, t_2) - MC(v_k, F^{old}, F^{new}) \quad \text{Eq. 9}$$

To illustrate the behavior of CAM we extend the example given in Figure 2, by introducing the communicational dependencies between VMs for a given time interval [t_1, t_2]. The aforementioned dependencies are shown in Figure 6. For simplicity, we make the following assumptions: **(a)** the computational demands of a VM also reflect the data needed to be transferred when migrating the respective VM. And **(b)** s_1, s_2, and s_3 are located on the same cluster, and the cost of exchanging one data unit with each other is equal to one. Therefore, according to the placement shown in Figure 2, the total network load incurred within the system equals 18.

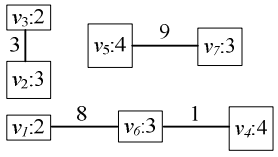

Figure 6. VM comm. dependencies.

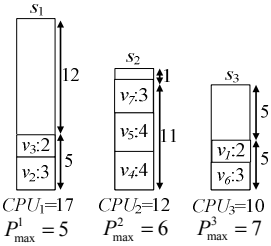

Figure 7. PCA-BFD's placement.

At first iteration, CAM will find that the best migration benefit is achieved when moving v_1 from s_1 to s_3. The aforementioned benefit according to Eq. 9 is 8-0-2=6. At next iteration, CAM decides to migrate v_3 from s_3 to s_2, with the migration benefit being equal to 9-0-3=6. In the sequel, the procedure stops because there is no VM migration that can improve the total network load. The final placement is shown in Figure 7, with the total network load being equal to one. It must be stressed that the aforementioned placement (called F^{CAM}) is the best placement regarding the network load compared to the placements produced by LPBP (10), BFD (10), PCA-BFD (11). However, F^{CAM} is defeated by all the aforementioned placements (F^{init}, F^{LPBP}, F^{BFD}, $F^{PCA\text{-}BFD}$) in regards to the energy consumption. Specifically, we have that $P_1(F^{CAM}, t) = 3.94$, $P_2(F^{CAM}, t) = 5.85$, $P_3(F^{CAM}, t) = 5.95$. Therefore, the total energy consumption is equal to 15.74.

D. Communication- and Power-Aware Migration Algorithm(CPAM)

In this section, we tackle the problem simultaneously in both dimensions. That is, we try simultaneously to minimize the energy consumption and the network load. To do so, we introduce an algorithm that builds gradually a two-dimensional solution-tree in a Pareto-efficient way. We must note that Pareto efficient solutions are the solutions that cannot be defeated simultaneously in all dimensions by any other solution. Nodes belonging to the intermediate levels of the solution-tree are named *intermediate nodes* and they represent partial solutions (partial assignments). Each node that belongs to a final leaf of the solution-tree is called *leaf node* and it represents a different complete solution (complete assignment). Each node (either intermediate or final) contains information about both the energy consumption and the network load incurred within the system when applying the corresponding (either partial or complete, respectively) solution.

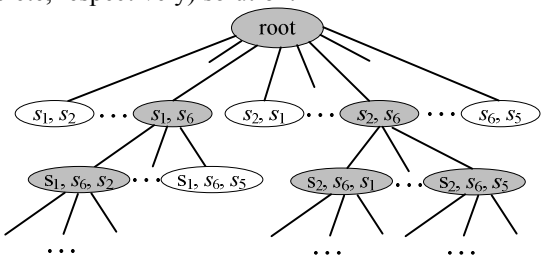

Figure 8. Expanding nodes in the solution-tree.

The process of building the solution-tree is split into H iterations, with H representing the height of the solution tree. Initially, we construct the root node belonging to the first level of the tree. We must note that the root node is only a symbolic node that does not represent any solution and does not carry any information. At first iteration, we construct the partial solutions belonging to the second level of the solution-tree, by considering all server pairs. At each iteration, we expand only the best candidate nodes by adding a new server to each node that is to be expanded. The process of evaluating the quality of a node is explained in the following paragraphs. The above discrimination in regards to the expansion of nodes is attributed to the fact that we are not able to explore all the possible solutions because of the prohibitive execution cost. The above procedure repeats itself, until there is no partial solution to be expanded. We must note that a node located at i^{th} level of the solution-tree contains i servers (except the root note that is only a symbolic node). To obtain a solution (partial or complete) for a node at i^{th} level, we do the following. We

apply LPBP on the servers belonging to the corresponding node. In the sequel, we apply CAM on top of the placement generated by LPBP. Afterwards, we record for the respective node the energy and the total network load incurred due to the resultant placement.

An example of building a solution-tree with six servers is shown in Figure 8. As we can see, the second level of the tree comprises of all server pairs. In the sequel, we expand only the best candidate pairs that are colored in grey, by adding a new server for each of them. As a consequence, the 3rd level of the tree is consisted of server triplets. Repeating the procedure, we expand the triplets into quadruplets, and so on. At last iteration, we construct nodes that are consisted of server sixplets that represent complete solutions.

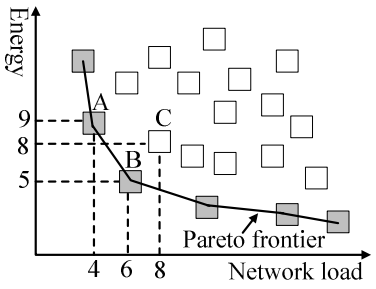

Figure 9. Pareto frontier

As stated in the preceding text, because the execution cost of exploring all the solutions is prohibitive, we resort to expanding only the best candidate nodes. We evaluate the quality of a node/solution across two metrics: the total network load and the energy consumption incurred within the system when applying the corresponding solution. Therefore, any solution (either partial or complete) is presented by a point in a two-dimensional space. Each point reflects two values (one for each dimension). For example, assume a partial solution that is represented by the point A (4, 9) that is pictured in Figure 9. Assume also that the two dimensions represent the network load and the energy incurred within the system when applying a specific solution. Then, the above partial solution (point A) incurs within the system four abstract units of network load and nine abstract units of energy. Best candidate nodes are considered the nodes where the solutions they represent belong to the Pareto frontier. We must note that Pareto frontier is the set of points (solutions) that are not dominated by other points. Put it otherwise, a point belonging to the Pareto frontier cannot be defeated simultaneously in both dimensions by any other point. The above is illustrated in Figure 9, where the points belonging to the Pareto frontier are colored grey, while the rest of them are colored white. As we can see, point C does not belong to the Pareto frontier, because it is defeated in both dimensions by point B. Specifically the solution represented by point B achieves better energy consumption and network load compared to the solution represented by point C. As stated earlier, points belonging to the Pareto frontier cannot be defeated simultaneously in both dimensions by any other point. Consider for example points A and B that both belong to the Pareto frontier. As observed, point A defeats point B in the dimension of network load, while it is defeated by point B in the dimension of energy.

V. EVALUATION

A. Experimental Setup

The presented algorithms were evaluated through simulations for different network topologies. For each experiment, the number of clusters varies between 10 and 15, with each cluster consisting of a varying number of racks. Specifically, the number of racks for each server is defined to be between 10 and 15. Each rack has been decided to have a number of physical servers that varies between 5 and 10. The average number of physical servers amounts to 1070. A total of 20 different network topologies were generated as follows. To simulate a geographical distribution of the clusters, we placed the clusters in a 70x70 2D plane. Clusters were assumed to be in range of each other if their Euclidean distance was less than 30. The cost of transferring one abstract data unit between virtual machines that are hosted on the same physical server is zero. The cost of transferring one abstract data unit between two servers that are located on the same rack has been fixed at one. The cost of transferring one abstract data unit between two servers that are located on the same cluster but on different racks has been decided to be 2. The cost of transferring one abstract data unit between two servers located on different clusters is equal to the summation of the costs of the inter-cluster communication links participating to the shortest path between the corresponding clusters. We must note that the cost of transferring one abstract data unit over an inter-cluster communication link varies uniformly between 5 and 10.

The maximum computing capacity of each physical server varies between 50 and 400 abstract computing units. The computing capacity required from a virtual machine to be executed correctly is chosen randomly between 10 and 50 abstract computing units. Each virtual machine communicated with a varying number of randomly selected virtual machines. The above number of virtual machines varies in a uniform distribution between 5 and 20. The data exchanged between a pair of virtual machines is decided to be uniformly distributed between 1 and 100 KB per time unit. While the data size of a virtual machine is chosen to be between 500 and 2000 MB.

In the following experiments we discuss the performance of LPBP, LPBP', PCA-BFD, CAM, and CPAM. We compare the performance of the above algorithms against the most well-known bin packing algorithms (BFD [16] and MBFD [1]) found in the literature. As discussed in the related work section, MBFD is power-aware bin packing algorithm. As a reference, we also include results obtained for a naïve algorithm (RAND) which randomly places a new virtual machine as long as there is a physical server with enough computing capacity to host it. In each experiment,

the quality of the solution is measured in terms of total network load and total energy consumption for the placement produced by the proposed algorithms. We also record the total number virtual machine migrations performed by our algorithms to reach the final placement.

B. Results

In the first experiment, we keep the number of VMs fixed such that the ratio between the total required VMs' computing capacity and the total computing capacity of physical servers is roughly 80%. Because the number of VMs is dependent on how many servers do exist in each topology, we report the average number of VMs, which is equal to 5080.

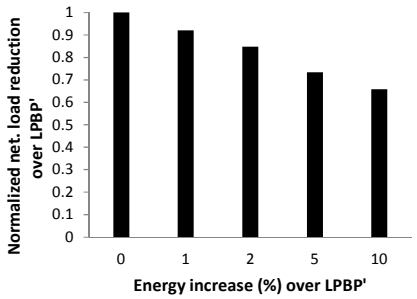

Figure 10. CPAM normalized net. load reduction when increasing energy.

Recall that CPAM results in more than one solution, with each solution belonging to the Pareto frontier. Among the solutions derived when running CPAM, we chose to show the ones when sacrificing 1%, 2%, 5%, and 10% more energy against LPBP'. Specifically, in the "y" axis we show the network load reduction (in percentage units) of CPAM against LPBP, while in "x" axis we show how much energy (in percentage units) we must sacrifice compared to LPBP' to obtain the corresponding network load reduction. The behavior of CPAM is depicted in Figure 10. As we can see, by sacrificing 1%, 2%, 5%, and 10% energy (over LPBP') we can result in a placement that is almost 8%, 15%, 26%, and 34%, respectively, better regarding the total network load against LPBP'. For the next experiments we show the performance of CPAM when sacrificing 2% energy against LPBP' (called CPAM(2)).

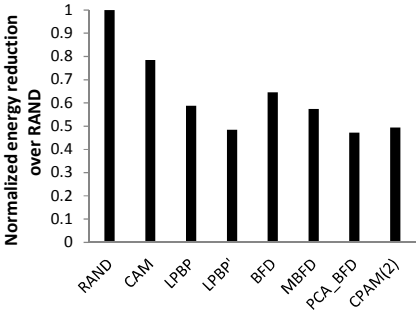

Figure 11. Normalized energy reduction against RAND.

In Figure 11, we show the normalized energy reduction (in percentage units) achieved by the proposed algorithms over the initial placement produced by RAND. As we can see the best performance is achieved by PCA BFD, with LPBP' and CPAM(2) following closely. MBFD, LPBP, and BFD do not perform so well as the above algorithms. This is attributed to the fact that when ordering the servers, they do not take into account the ratio between their power consumption and computing capacity. Another (expected) remark is that CAM is remarkably inferior compared to all of the aforementioned algorithms. Even though, both BFD, and CAM do not take into account the power consumption of the servers when deciding for the placement of VMs, they perform always better compared to RAND. The above is attributed to the fact both BFD and CAM have packing capa bilities, while RAND does not have. Specifically, BFD directly minimizes the number of opened servers, by packing VMs onto as few servers as possible. On the other hand, CAM indirectly minimizes the number of opened servers, by packing VMs communicating heavily on the same servers. We must note that the observed superiority of BFD against CAM is justified by the fact that BFD's packing ability is strongest against that of CAM.

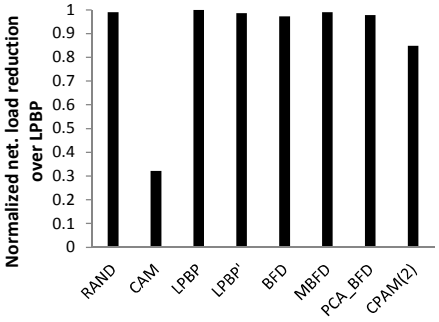

Figure 12. Normalized net. load reduction against LPBP.

In Figure 12, we demonstrate the normalized network load reduction (in percentage units) achieved by the proposed algorithms over LPBP, which is the algorithm with the worst performance. We must say that, besides CPAM(2) and CAM, all of the proposed algorithms do not take into account the network load incurred between VMs when making the placement decisions. By delving into the raw data, we noticed that also BFD, MBFD, LPBP', RAND, and PCA-BFD had the worst performance in some cases. Therefore, we claim that the fact that LPBP achieves the worst performance in average is incidental. We must note that we performed extra experiments to corroborate the above. As we can see, CAM results always in superior placements (up to 70% better performance against LPBP) compared to the rest algorithms, with CPAM(2) being the next best candidate. The reason that CPAM(2) is not so good as CAM is explained by the fact that CPAM(2) sacrifices only two percentage units of energy to result in the corresponding network load reduction against LPBP'. Therefore, by sacrificing more energy, CPAM could result in better placements regarding the total network load.

The number of migrations performed by the proposed algorithms is demonstrated in Figure 13. Because RAND produces the initial placement for the rest algorithms, it does not perform any migration. Therefore, we exclude RAND from Figure 13. As observed, the worst behavior is achieved

by BFD, MBFD and PCA BFD. The above was expected since the aforementioned algorithms result in new VM placements without taking into account the old ones. LPBP performs the smallest number of VM migrations, with LPBP' following closely. The above is attributed to the natu-

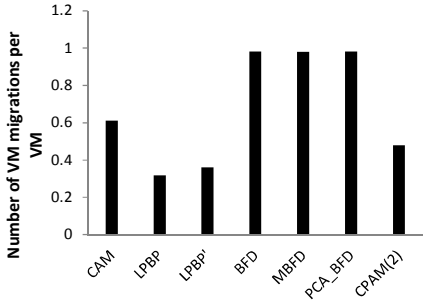

Figure 13. Number of VM migrations performed.

re of the aforementioned algorithms to gradually improve the initial placement. We also observe that CPAM(2) and CAM do not perform as many VM migrations as BFD, MBFD, and PCA BFD. The above explanation is attributed to the fact that CPAM(2) is partly based on LPBP', while CAM performs a gradual transition from an old placement to a new one.

Last, we must note that we conducted more experiments by varying the ratio between the total required VMs' computing capacity and the total computing capacity of physical servers. We noticed that the trend of the algorithms remained the same. Therefore, we choose not to show them due to the monotonicity of the results.

VI. CONCLUSIONS

In this work, we tackled the problem of jointly optimizing the energy consumption and the network load incurred within clouds. Even though, the VM consolidation problem has been addressed in many papers in the literature, this is the first time that the above problem is tackled with simultaneously minimizing the total network load incurred due to **(a)** VM communicational dependencies and **(b)** VM migrations. Through our experimental evaluation, we showed that our single-solution heuristics achieved good performance compared to the ones existing in the literature. It is also shown that Pareto optimization is crucial to such kind of problems where multiple objectives must be optimized in a simultaneous manner. We also showed that when optimizing only one of the two objectives, then we resulted in poor performance regarding the objective we did not take into consideration. Another remark that we must make is that Pareto-efficient algorithms, as CPAM, give the liberty to the final decision maker to choose among various solutions, as opposed to algorithms that result only in one solution.

ACKNOWLEDGEMENTS

This work is partially supported by the ZTE Corporation and Chinese Government Fund.

Samee U. Khan's work was partly supported by the Young International Scientist Fellowship of the Chinese Academy of Sciences, (Grant No. 2011Y2GA01).

Nikos Tziritas's work was partly supported by the Postdoctoral Fellowship of the Chinese Academy of Sciences.

REFERENCES

[1] A. Beloglazov, J. Abawajy, R. Buyya, "Energy-aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing," *In Journal Future Generation Computing Systems*, Vol 28 (5), pp. 755-768, 2012.

[2] A. Beloglazov, R. Buyya, "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers," *Concurrency and Computation: Practice and Experience*, Vol. 24, pp. 1397-1420, 2012.

[3] M. Bienkowski, A. Feldmann, D. Jurka, W. Kellerer, G. Schaffrath, S. Schmid, J. Widmer, "Competitive Analysis for Service Migration in VNets," *ACM SIGCOMM workshop on Virtualized Infrastructure Systems and Architectures (VISA)*, 2010.

[4] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, "Live Migration of Virtual Machines," *In Proc. of NSDI*, 2005.

[5] C. Curino, E. P. C. Jones, S. Madden, H. Balakrishnan, "Workload-Aware Database Monitoring and Consolidation," *ACM International Conference on Managing of Data (SIGMOD)*, 2011.

[6] Environmental Protection Agency (EPA): Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431 (2007).

[7] E. Feller, L. Rilling, C. Morin, "Energy-Aware Ant Colony Based Workload Placement in Clouds," *IEEE/ACM International Conference on Grid Computing (GRID)*, 2011.

[8] A. Gandhi, M. Harchol-Balter, R. Das, C. Lefurgy, "Optimal power allocation in server farms," *ACM International Joint Conference on Measurement and Modeling of Computer Systems*, 2009.

[9] M. R. Hines, K. Gopalan, "Post-copy Based Live Virtual Machine Migration Using Adaptive Pre-paging and Dynamic Self-ballooning," *Proc. ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, 2009.

[10] Q. Huang, F. Gao, R. Wang, Z. Qi, "Power Consumption of Virtual Machine Live Migration in Clouds," *In Proc. of 3rd International Conference on Communications and Mobile Computing*, 2011.

[11] W. Huang, Q. Gao, J. Liu, D. K. Panda, "High Performance Virtual Machine Migration with RDMA over Modern Interconnects," *IEEE International Conference on Cluster Computing*, 2007.

[12] D. Kusic, J.O. Kephart, J.E. Hanson, N. Kandasamy, G. Jiang, Power and performancee management of virtualized computing environments via lookahead control, *Cluster Computing*, Vol. 12 (1), 2009.

[13] H. Liu, J. Jin, X. Liao, L. Hu, C. Yu, "Live Migration of Virtual Machine Based on Full System Trace and Replay," *ACM International Symposium on High Performance Distributed Computing*, 2009.

[14] T. Maoz, A. Barak, L. Amar, "Combining Virtual Machine Migration with Process Migration for HPC on Multi-Clusters and Grids," *IEEE International Conference on Cluster Computing*, 2008.

[15] R. T. Marler, J. S. Arora, "Survey of Multi-Objective Optimization Methods for Engineering," *Springer Structural and Multidisciplinary Optimization*, Vol. 26 (6), pp. 269-395, 2004.

[16] S. Martello, P. Toth, "Knapsack Problems – Algorithms and Computer Implementations," *John Wiley & Sons*, 1990.

[17] G. B. Mertzios, M. Shalom, A. Voloshin, P. W. H. Wong, S. Zaks, "Optimizing Busy Time on Parallel Machines," *International Parallel and Distributed Processing Symposium (IPDPS)*, 2012.

[18] J. Sonnek, J. Greensky, R. Reutiman, A. Chandra, "Starling: Minimizing Communication Overhead in Virtualized Computing Platforms Using Decentralized Affinity-Aware Migration," *International Conference on Parallel Processing (ICPP)*, 2010.

[19] N. Tziritas, S. U. Khan, C.-Z. Xu, J. Hong, "An Optimal Fully Distributed Algorithm to Minimize the Resource Consumption of Cloud Applications", *IEEE International Conference on Parallel and Distributed Systems*, 2012.

[20] W. Voorsluys, J. Broberg, S. Venugopal, R. Buya, "Cost of Virtual Machine Migration in Clouds: A Performance Evaluation," *International Conference on Cloud Computing*, 2009.