A Communication-Aware Energy-Efficient Graph-Coloring Algorithm for VM Placement in Clouds

Nikos Tziritas
Cloud Computing Department
Shenzhen Institutes of Advanced Technology
Shenzhen, China
e-mail: nikolaos@siat.ac.cn

Thanasis Loukopoulos
Dept. of Computer Science and Biomedical Informatics
University of Thessaly
Lamia, Greece
e-mail: luke@dib.uth.gr

Samee U. Khan
Electr. and Computer Eng. & Div. of Computer and Network Systems
North Dakota State Univ. & National Science Foundation
Fargo & Alexandria, USA
e-mail: samee.khan@ndsu.edu

Cheng-Zhong Xu
Cloud Computing Department
Shenzhen Institutes of Advanced Technology
Shenzhen, China
e-mail: cz.xu@siat.ac.cn

Albert Y. Zomaya
School of Information Technology
University of Sydney
Sydney, Australia
e-mail: albert.zomaya@sydney.edu.au

***Abstract*— The problem of virtual machine (VM) consolidation has received a lot of attention over the past years. Most of the proposed techniques tackling the VM consolidation problem focus on energy consumption ignoring the network traffic incurred within the system. Redundant network traffic may be crucial for the sustainability of cloud computing systems. To tackle the aforementioned problem, we propose a communication-aware graph-coloring algorithm placing the VMs in the underlying system in an energy efficient manner. Specifically, the proposed algorithm considers migrating VMs in batches resulting in energy efficient placements against approaches tackling single VM migrations. We have conducted an experimental evaluation to show the superiority of the proposed algorithm against state-of-the-art algorithms in terms of network overhead reduction, while keeping the energy consumption at low levels.**

Keywords-component; VM consolidation, communication-aware algorithms, energy-efficient algorithms

I. INTRODUCTION

During the last years, the cloud computing paradigm has received wide attention from academia and industry due to several promising characteristics, such as elasticity and on-demand computing [1], [20], [21], and [40]. Nevertheless, the rising energy cost has become a considerable burden to successfully execute cloud based technologies [4], [7], [8], [10], and [12]. By reducing the energy expenditures within a cloud computing environment, a cloud provider can decrease the Total Cost of Ownership (TCO) and subsequently increase the Return on Investment (ROI). Because of the above fact, reduction in energy consumption has become an ardent desire for the cloud providers. Such a reduction can be achieved by leveraging on the virtualization technologies that is one of the most prominent features of cloud computing to increase system utilization. A part of the ongoing research focuses on the workload consolidation that minimizes the number of physical machines by decommissioning the idle machines. Such a simple technique can drastically reduce the energy expenditures within a cloud environment [1] , [22].

Workload consolidation (if not done judiciously) may adversely affect the performance of executing applications, due to the overhead incurred by the virtual machine (VM) migrations. Specifically, Ref. [6] reports that the performance of an 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 the VM migrations incur additional network traffic, causing bottlenecks and out-of-order message relying [5] and [17]. Therefore, extra care must be taken regarding which VMs are to be migrated, so as not to overburden the network with extra communicational overhead, predominantly due to the communication dependencies between the VMs [30] and [29]. Because of the above, we propose an innovative graph-coloring algorithm that performs VM migration such that to minimize network traffic while keeping the energy consumption at low levels. Specifically, our primary aim is to develop competitive solutions for the following problem: *Assuming that a set of VMs is already deployed within a cloud, reassign the VMs to physical servers so as to minimize both the aggregate energy consumption within the system, and the total network overhead incurred due to the: (i) communicational dependencies between VMs and (ii) VM migrations.*

The main contribution of our work is summarized as follows.

- We adopt a server power consumption model, whereby a VM may consume proportionally

different amounts of energy when executed on different servers. Therefore, the energy consumption is dependent on both the type of VM and the server it is executed on.

- The proposed algorithm employs a graph coloring technique as well as the knapsack approach to migrate VMs from one server towards another one in an iterative fashion. The proposed algorithm aims at optimizing the network overhead due to the VM communication inter-dependencies. At the same time, the proposed algorithm keeps the energy consumption at low levels.
- We conducted experiments to show the behavior of the proposed algorithms compared to the state-of-the-art algorithms found in the literature. Specifically, the proposed algorithm achieved energy consumption close to state-of-the-art algorithms in the literature.

The remainder of the 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 graph-coloring algorithm is presented in Section IV and evaluated in Section V. Finally, Section VI concludes the paper.

II. Related Work

Large data center operators, such as Google and Amazon are actively pursuing solutions to reduce the energy expenditures of their data centers. Ref. [9] claims that such a reduction can be achieved by keeping system utilization at a maximum level decommissioning idle servers. Authors in [27] propose a workload consolidation algorithm called Modified Best Fit Decreasing (MBFD) that modifies the Best Fit Decreasing (BFD) approach. MBFD sorts all of the VMs in decreasing order as per the computing capacity, and assigns each of the VM to a host providing the least increase in the power consumption due to such an assignment. Ref. proposes Adaptive heuristics are proposed in [3] to address the VM consolidation problem in an online setting. Gandhi *et al.* [11] address the problem of allocating an available power budget among servers in a virtualized heterogeneous server farm, while minimizing the mean response time. Bin packing algorithms are proposed in [36] to reduce energy consumption in cloud systems. The proposed algorithms (named LPBP and LPBP') perform VM migrations in a low perturbation way to keep the number of VM migrations as low as possible. Authors also propose a bin packing algorithm (called PCA_BFD) which is aware of power and computing capacity when assigning VMs onto servers.

Mertzios *et al.* [28] have considered the interval scheduling problem to minimize the power consumption of a set of machines. In the above problem, authors attempt to consolidate jobs whose processing times overlap to minimize the busy time of machines. Curino *et al.* [7] addresses the problem of consolidating multiple databases on fewer servers. Authors perform an analysis about the load characteristics of multiple dedicated database servers and consolidate the corresponding workloads into fewer physical machines, while achieving negligible performance degradation. In [14] a resource consolidation framework is proposed for multiple virtual clusters. The authors treat the VMs as "moldable" during consolidation such that to allocate the VMs into a fewer number of servers. The main difference between our work and [14] is the system model since authors assume virtual clusters, with each VM being assigned to a specific virtual cluster, while each node can host at most one VM of each virtual cluster. Authors of [39] and [41] address the problem of scheduling precedence-constrained parallel tasks such that to minimize both execution time of the workload as well as energy consumption.

Redundant communication load may lead to considerable performance degradation of communication-intensive applications [29]. Sonnek *et al.* [29] perform VM migrations to reduce the total communication overhead, resulting in that way in an improved application performance. Authors solve the aforementioned problem by proposing a distributed bartering algorithm (called DBA) which places the VMs closer to the required data. Similarly, authors in [31] and [32], propose an optimal fully distributed algorithm for tree-structured networks leveraging on the minimum-cut maximum-flow techniques. In [37], authors address the online problem of migrating services such that to minimize the total network overhead between them. The proposed algorithm is accompanied with a rigorous analysis about its competitive ratio. As can be seen from the above, the work reported in [29], seems to be the closest against our work. Authors In [36] propose a communication aware algorithm (called CAM) to reduce network traffic. The proposed algorithm examines only single VM migrations against our solution performing batch of VM migrations. In [33], [34], and [35] authors propose algorithms to co-locate heavy-communicating agents as well as to place them close to data such that to minimize the total network traffic.

In terms of the below discussion, we report the most relevant works we found in the literature regarding the mechanisms of live VM migrations (i.e., the process of migrating a running VM between two different physical servers without disconnecting the applications executing on the respective VM) and the corresponding effects on cloud computing systems. Specifically, authors in [16] address the problem of live VM migration across a Gigabit LAN. The proposed solution facilitates the use of post-copy with adaptive pre-paging to eliminate any duplicate page transmission. Ref. [18] leverages on the Remote Direct Memory Access (RDMA) to optimize the time efficiency of VM migration. In terms of [26], authors propose a methodology taking into account both VM migration and process migration for the execution of high performance computing applications on clusters and grids. Last, the work presented in [24] proposes an algorithm that synchronizes the migrating VM states. Authors demonstrated through experimental evaluation that their approach can drastically reduce the migration overheads against a pre-copy algorithm. We must note that such works are complementary to our work, as our algorithms could adopt such mechanisms to implement the VM migrations without disrupting the functioning of the migrating VMs.

III. System Model and Problem Formulation

A. System Model

We assume that the network is organized as a three-level hierarchical structure. The first level is composed of an interconnected set of data centers, with each of them consisting of a number of racks (the second level). These racks contain a number of physical servers (the third level level). The communication between the physical servers takes place through three different methods based on locale. Specifically: (i) servers within a rack communicate with each other through the edge layer switch; (ii) servers that are located within the same cluster but on different racks communicate through the core layer switch; (iii) the communication between two servers that are at different data centers (geographically distributed) is more than often effected by the bottleneck-links.

To capture the communication dependencies between processes (or VMs) and nodes, we extend the number of processes by N virtual ones, with each of them representing a node. Therefore, the data exchanged between a virtual process and a real process represents the communicational demands between the real process and the node represented by the virtual process. Specifically, p_i is a real process iff $0 < i \leq P$, and a virtual one iff $P < i \leq P+N$. By doing so, we are able to extend the matrix encoding the communicational dependencies between processes to an $(P+N)\times(P+N)$ matrix that includes the communication dependencies between processes and nodes (besides the ones between processes). For simplicity, let the virtual process representing n_x be denoted by p_{n_x}.

Let s_x and S be the x-th physical server and the total number of servers within a cloud, respectively. The variable d_{xv} captures the "cost" of exchanging one abstract data unit (e.g., byte) between s_x and s_v, where $d_{xx} = 0$. Let J be the number of jobs comprising an application to be executed within a cloud, and j_i identifies the i-th such job. The communication dependencies between jobs, within 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, where $w_{xy}^{t_1,t_2} = w_{yx}^{t_1,t_2}$. The number of VMs is denoted by V, while the k-th VM within the system is denoted by v_k. To capture the communication dependencies between two VMs v_k and v_m, we aggregate the data exchanged between the set of jobs executed on v_i and the set of jobs executed on v_i. The total data exchanged between v_k and v_m, within a time interval $[t_1, t_2]$, is encoded by an $V\times V$ matrix, denoted by $c_{mk}^{t_1,t_2}$, where $c_{mk}^{t_1,t_2} = c_{km}^{t_1,t_2}$. The z-th switch within the system is defined by sw_z, while SW denotes the total number of switches. The binary variable r_{mk}^{z} is equal to one, if the communication between v_m and v_k must pass through sw_z; otherwise, r_{mk}^{z} equates to zero. Because r_{mk}^{z} depends on the VM placement, we must modify the notion to $r_{mk}^{z}(F)$, where F represents a placement.

B. Problem Formulation

To provide a rigorous mathematical formulation of the problem we must introduce the following terminologies. 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. The variable f_{kx} denotes whether v_k is hosted by s_x or otherwise. More specifically, f_{kx} is equal to one, if v_k is hosted by s_x, otherwise f_{kx} equates to zero. The computing capacity of a server s_x is represented by $C(s_x)$, while the computing capacity required from a v_k to be executed correctly for a specified point in time t is defined by $REQ(v_k,t)$. Let $US_x(F,t)$ represent the utilization of s_x as a function of a placement F and time t, which is to be calculated using Eq. 1. The variable $USW_x(F,t)$ denotes the utilization of the z-th switch as a function of F and t, is calculated using Eq. 2. Eq. 3 and Eq. 4 state that the total utilization of a server and a switch, respectively, cannot be greater than one. Let $PS_{\max}^{x}$ and $PSW_{\max}^{j}$ denote the maximum instantaneous power consumed by a server s_x and a switch sw_j, respectively. Given a placement F and a point in time t, then the power consumption of a server s_x is represented by $PS_x(F, t)$. Recent studies (e.g., [23]) have shown that different VMs consume different amounts of power. Specifically, Koller *et. al.* [23] have shown that the power consumption does not only depend on the VM but also on the server it is executed. Therefore, in this paper the model for the server power consumption is built according to the aforementioned findings, which may be captured by Eq. 5 and Eq. 6. Particularly, Eq. 5 models the power consumption of a specific VM v_k running on s_x. a_{xk} and b_{xk} are VM and server dependent parameters that are predicted through linear regression [23]. Regarding Eq. 6, it captures the total power consumption of s_x. In terms of Eq. 7, ξ_j and γ_j are network switch dependent parameters that can be easily obtained through measurements [25], [15]. The total energy consumed by all of the servers and switches within the cloud for a given F and within a time interval $[t_1, t_2]$, is captured by Eq. 8. Now, Eq. 9 records the total network load incurred within the cloud with F and within $[t_1, t_2]$. The implementation network overhead for the transition from F^{old} to F^{new} is given by Eq. 10.

$$US_{xk}(F,t) = f_{xk} \times REQ(s_k,t) / C(s_x) \tag{1}$$

$$USW_j(F,t) = \sum_{k=1}^{V}\sum_{m=1}^{V} r_{mk}^{j}(F) \times c_{mk}^{t} / C(sw_j) \tag{2}$$

$$\sum_{k=1}^{V} f_{xk} \times REQ(s_k) / C(s_x) \leq 1 \tag{3}$$

$$\sum_{k=1}^{V}\sum_{m=1}^{V} r_{mk}^{j}(F) \times c_{mk}^{t} / C(sw_j) \leq 1 \tag{4}$$

$$PS_{xk}(F,t) = f_{xk} a_{xk} PS_{\max}^{x} + f_{xk} \beta_{xk} PS_{\max}^{x} \times US_{xk}(F,t) \tag{5}$$

$$PS_x(F,t) = \max\{f_{x1}a_{x1}, \ldots, f_{xV}a_{xV}\} \times PS_{max}^{x} + \sum_{k=1}^{V} \beta_{xk} PS_{max}^{x}\, US_{xk}(F,t) \tag{6}$$

$$PSW_j(F,t) = \xi_j PSW_{\max}^{j} + \gamma_j PSW_{\max}^{j} \times USW_j(F,t) \tag{7}$$

$$En(F,t_1,t_2)=\sum_{x=1}^{S}\int_{t1}^{t2}PS_x(F,t)dt+\sum_{j=1}^{SW}\int_{t1}^{t2}PSW_j(F,t)dt \quad (8)$$

$$comm(F,t_1,t_2)=\sum_{m=1}^{V}\sum_{k=1}^{V}\sum_{x=1}^{S}\sum_{y=1}^{S}c_{mk}^{t_1,t_2}f_{mx}f_{ky}d_{xy} \quad (9)$$

$$\mathrm{Impl}(F^{old},F^{new})=\sum_{k=1}^{V}\sum_{x=1}^{S}\sum_{y=1}^{S}S(v_k)f_{kx}^{old}f_{ky}^{new}d_{xy} \quad (10)$$

Based on the above discussion, the problem can be formally stated as: *Given a number of VMs with communicational inter-dependencies due to inter-communicating jobs that are being hosted by the VMs, find a feasible placement F* of the VMs onto physical machines to minimize:* (i) *energy spent due to the active physical servers and* (ii) *total network overhead due to the communication inter-dependencies between the VMs and the VM migrations that are performed to implement the transition from* F^{old} *to* F^{new}.

IV. Graph Coloring Algorithm (GCA)

In this section, we present an energy-efficient graph coloring algorithm for minimizing the network overhead. The algorithm consists of two procedures: (i) the VMs exchange procedure called the VXP and (i) the server pair selection procedure called the SPSP. The VXP procedure concerns the part where two servers are chosen to exchange VMs with each other to reduce the communication overhead incurred within the network. Such an overhead is due to the communicational dependencies between the VMs hosted on the chosen pair of servers. The SPSP procedure invokes the VXP for each pair of the servers within the network until there are no possible exchanges of the VMs that can further reduce the total network overhead.

A. VMs Exchange Procedure (VXP)

Initially, given a pair of servers, we construct the VM communication graph that depicts the communication among the VMs belonging to the servers in question. The vertices of the graph have a one-to-one correspondence with the VMs belonging to the chosen pair of servers. Moreover, each vertex has a weight equaling to the amount of computing resources required from the corresponding VM to be correctly executed. Each edge of the VM communication graph represents the communicational dependencies between the corresponding VMs. For example, given the network and application graph pictured in Fig. 1 and a server pair (s_2, s_5), we construct the VM communication graph, as shown in Fig. 2.

Next, the VM communication graph is transformed into a coloring graph. Specifically, the weight of each of the edge of the VM communication graph is multiplied by the number of hops separating the servers of a chosen server pair. The above is due to the fact that the merger of two vertices must reflect the actual benefit of a combined hosting of the respective VMs. Moreover, the VM communication graph is extended by adding two vertices, with these vertices corresponding to the servers of the chosen pair. The aforementioned vertices have weight that is equal to zero and are colored randomly through a 2-color scheme (e.g. gray, black). Note that the rest of the vertices (VM vertices) remain uncolored for the time being. To take into account the VM migration cost, for each of the VM vertex, we add two extra edges towards the two server vertices. Because such an edge represents the migration cost of the corresponding VM, the weight is calculated in the following way. If the VM represented by the incident VM vertex to such an edge is not hosted on the server represented by the incident server vertex, then the weight is equal to zero. Otherwise, the weight is equal to the migration cost of the corresponding VM from the server of the chosen pair hosting it towards the server of the chosen pair. Fig. 3 shows an illustrative example of the aforementioned scenario. Specifically, the weight of each edge pictured in Fig. 2, is multiplied by the number of hops (three in this case) separating the servers of the chosen pair (referring to s_2 and s_5). For clarity, the edges between (s_2, s_5) and (v_1, v_4) are omitted. The weight of the edge between (s_2, v_2) equals to the size of v_2 (that being two), multiplied by the number of hops between (s_2, s_5), giving a total value of six. Similarly, the weight of the edge (s_5, v_3) equals a value of twelve (4×3). In general, the VMs hosted on the servers other than the ones included in the chosen server pair, may generate communication overload. In Fig. 1 and Fig. 2, we can see such a scenario, by considering the server pair (s_2, s_5), where v_5, v_6, and v_7 generate communicational loads towards the v_1, v_2, and v_3, respectively. Such an external (to the server pair) load must be incorporated within the graph coloring model. The external load can be viewed as another form of server related cost within the model, as was the case with the cost of the VM migration. Consider for instance the migration of v_3 from s_5 to s_2. Aside from the migration cost of twelve to transfer v_3 from s_5 to s_2, there will also be a change to the network overhead in terms of the external communication load directed to/from v_3. For instance, the load generated by (v_3, v_7) will not incur a network overhead of five, but rather a cost of ten, as the hop distance between v_3 and v_7 will increase from one to two. To incorporate the above case in the coloring graph, it suffices to augment: (i) the weight of the edge incident to v_3 and s_5 by the network overhead incurred if v_3 moved to s_2 and (ii) the weight of the edge incident to v_3 and s_2 by the network overhead incurred if v_3 stayed at s_5. Repeating the process for all of the external loads of v_3 results in: (i) the weight between v_3 and s_5 being augmented by ten (v_7's load) plus eight (v_8's load) for a total of 18 and (ii) the weight between v_3 and s_2 being augmented by five (v_7's load) plus four (v_8's load) for a total of nine. Fig. 4 illustrates the augmented coloring graph. As indicated previously, to avoid cluttering, only the edges between s_2, s_5 and v_2, v_3 are depicted in the figure.

Recall that the merger of two VM vertices must reflect the actual benefit of collectively hosting the respective VMs.

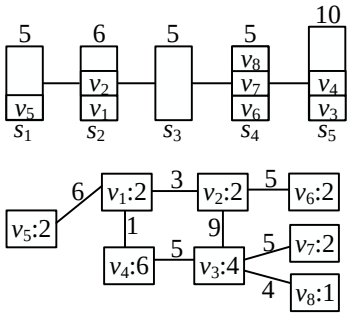

Fig. 1. Network and application graph.

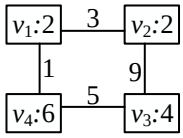

Fig. 2. The VM communication graph.

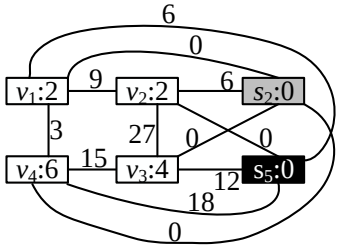

Fig. 3. The coloring graph.

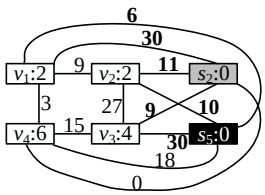

Fig. 4. The augmented coloring graph

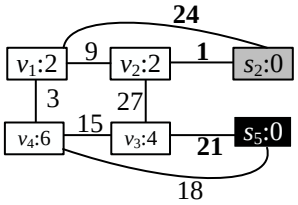

Fig. 5. The final coloring graph.

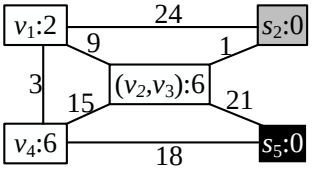

Fig. 6. Merging Step 1

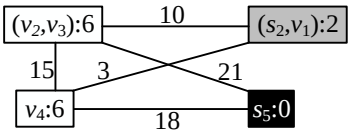

Fig. 7. Merging Step 2

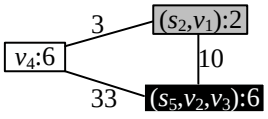

Fig. 8. Merging Step 3

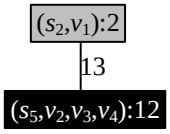

Fig. 9. Merging Step 4

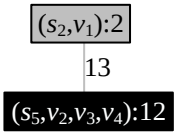

Fig. 10. Merging Step 5

Similarly, the merger of a VM vertex with a server vertex must also reflect the actual benefit when the respective server hosts the respective VM. Reverting to the example scenario, if v_2 and v_3 are placed together, then the communication load will be alleviated to, corresponding to the actual benefit of 27. However, the same is not true when considering the edges involving a server vertex. For instance, if v_3 and s_5 are merged, then the actual benefit will not be 30, but rather benefit would be the network overhead difference between placing v_3 at s_5 and at s_2. Accordingly, for each of the VM vertex we must perform the following: (i) we remove from the edges connecting the vertex (in question) the server vertices with the lowest weight and (ii) we change the weight of the rest of the edges to be equal to the difference of the weight of the edges. Note that in case the edges in question have equal weights, then we remove on random one of the two edges, and we set the weight of the rest of the edges equal to zero. The aforementioned changes are reflected in Fig. 5.

After constructing the final coloring graph, we proceed to the step of merging the vertices of the graph. Such a step is iterative in nature. In each of the iteration, the edge with the highest weight is selected and the vertices that it connects with are merged. Specifically, if the incident vertices to the aforementioned edge: (i) contain only VMs, then the merging benefit is obtained from placing the respective VMs on the same server or (ii) contain a server and VMs, then the merging benefit is derived from placing the respective VMs at the respective servers. The new vertex has the cumulative weight of the previous vertices and the remaining edges. If the new vertex has more than one edge towards another vertex, then such edges are compacted into one edge. If any of the vertices belonging to the merged vertex is colored, then the merged vertex will also be colored the same. In the case when the two vertices to be merged are colored with different colors, the merging is not performed and the respective edge becomes grey. The aforementioned scenario is captured in the example depicted in Fig. 6 through Fig. 10. Specifically, we choose to merge v_2 and v_3 in Fig. 6. The above is performed because the edge incident to v_2 and v_3 has the largest weight in the graph shown in Fig. 5. In Fig. 7, v_1 is merged with s_2, with the new vertex (s_1, v_1) being colored gray due to the fact that the uncolored vertex v_1 is merged with the gray-colored vertex s_2. Subsequently, the uncolored vertex (v_2, v_3) is merged with the black-colored vertex s_5 and the new vertex is colored black. Thereafter, in Fig. 9, v_4 is merged with (s_5, v_2, v_3) and the new vertex is colored black. In Fig. 10, there are two colored vertices that cannot be merged. Therefore, the edge connecting the two vertices is colored grey. Lastly, because there is no other edge to be considered, the algorithm terminates with the actual benefit (in terms of network overhead) being equal to fourteen (9×3-3×3+2-3×2).

Each time when the vertices are merged, the GCA attempts to find if a feasible vertex coloring exists within the new graph. To realize such feasibility checks the GCA

solves the knapsack twice. The first time knapsack is run for the one server of the chosen server pair and next for the rest (i.e., s_2 and s_5 according to the chosen server pair in our running example). The knapsack candidate objects represent the vertices of the coloring graph (the size of each object being the weight of the vertex). Having obtained a knapsack solution for s_2 (grey node) the algorithm checks if the remaining objects fit in s_5. If so, then the algorithm accepts the merging of the respective vertices and proceeds with the next iteration. Otherwise, the algorithm attempts to find a valid placement by solving knapsack for s_5 (the black node) and checking whether the remaining objects fit at s_2. If after trying both of the knapsack solutions the VXP is unable to find a valid placement involving all of the objects, it backtracks to the graph state before merging, marking the edge under consideration as grey.

The VXP continues in the same fashion until either all of the remaining agent-vertices are colored, indicating migrations; or edges are colored in grey, indicating that no migrations are performed.

B. Server Pair Selection Procedure

The SPSP iterates through all of the server pairs by applying the VXP. If during an iteration the VXP reduces the total network overhead, then the process reiterates. Otherwise, the VXP terminates with the final VM placement. We must note that if the VM communicational dependencies do not change frequently, then the VM migration overheads instead of fully contributing into the coloring graph; however, the migrations may contribute in a small percentage. Such a percentage contribution is captured by a constant called the MIG_OVERHEAD. For example, when the MIG_OVERHEAD is equal to 0.1, then it means that the overhead of the VM migrations contributes by a 10% when constructing the coloring graph. For the VXP to successfully optimize (within a server pair), the VM placement adaptations are required for the contribution of the overhead of the VM migrations.

V. Evaluation and Discussion Results

The presented algorithms were evaluated through simulations for various network topologies. the number of clusters varied between five and ten, with each cluster consisting of a varying number of racks. Specifically, the number of racks for each cluster was defined to be between ten and twenty. Each of the rack had been decided to have a number of physical servers varying between twenty and forty. The average number of physical servers amounted to 3,375.

A total of 20 different network topologies were generated for each of two setups as follows. To simulate a geographical distribution of the clouds, we placed the clusters in a 80×80 (measured in meters) 2D place. Clusters were assumed to be within the range of each other, if their Euclidean distance was less than 30 meters. The network overhead when transferring one abstract data unit between VMs that are hosted on the same physical server was considered to be equal to zero. The cost of transferring one abstract data unit between two servers that are located on the same rack was fixed at one. The cost of transferring one abstract data unit between two servers located on the same cluster but on different racks was decided to be equal to two. The cost of transferring one abstract data unit between two servers located on different clusters was assumed to be equal to two plus the summation of the inter-cluster communication links participating to the shortest path between the corresponding clusters.

The maximum computing capacity of each physical server varied between 50 and 400 abstract computing units. The computing capacity required from a virtual machine to be executed correctly was chosen randomly between 10 and 50 abstract computing units. Each of the VM communicated with a varying number of randomly selected VMs. The above number of virtual machines varied in a uniform distribution between 5 and 20. The data exchanged between a pair of virtual machines was decided to be uniformly distributed between 1 KB and 100 KB per time unit. While the data size of a VM is chosen to be between 500 and 2000 MB. The server power consumption was assumed to be between 200 and 400 watts [13]. On the other hand, the power consumption of networking components was decided according to [15]. VMs have different power consumption based on the servers they are hosted according to [23].

In the following simulations, we discuss the performance of GCA against state-of-the-art algorithms. the LPBP, LPBP', PCA-BFD, CAM, GCA, CPAM, CPAMI, and CPAM-GCA. We compare the performance of the above mentioned algorithms with the most well-known bin packing algorithms (MBFD [1], BFD [27], pMaP [38], LPBP/LPBP' and PCA_BFD [36]) found in the literature. As discussed in the related work section, the MBFD is power-aware bin packing algorithm. In terms of the network overhead, we compared DBA [29] and CAM [36] with our proposed algorithms. As a reference, we also included results obtained with a naïve algorithm (RAND) that randomly places a new VM as long as there is a physical server with enough computing capacity. In each of the simulations, the quality of the solution is measured in terms of the total network load and total energy consumption for the placement produced by the proposed algorithms. We also recorded the total number VM migrations performed by our algorithms to reach the final placement.

A. Experiments

In Fig. 11, we show the normalized energy reduction (in percentage units) achieved by the proposed algorithms over the initial placement produced by the RAND. As observed, the best performance was achieved by LPBP'. The LPBP, pMaP, MBFD, PCA_BFD, and GCA algorithms have slightly worst performance compared to the LPBP'. At the other extreme, BFD, CAM and DBA record the worst performances. The same performance features of the algorithms roughly hold when the capacity ratio changes to the MID and LARGE. As we can observe, the algorithms perform better for the case when the capacity ratio is set to LARGE.

In Fig. 12, we demonstrate the normalized network overhead reduction (in percentage units) achieved by the proposed algorithms over the RAND. We must report that,

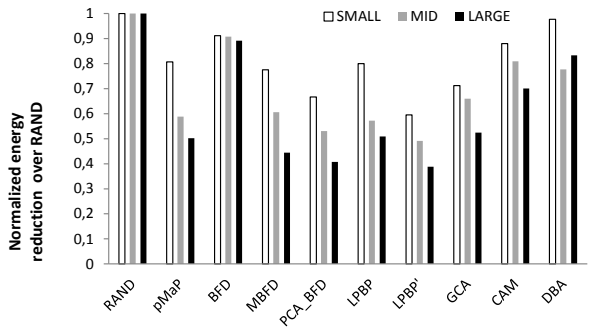

Fig. 11. Normalized energy reduction compared to the RAND.

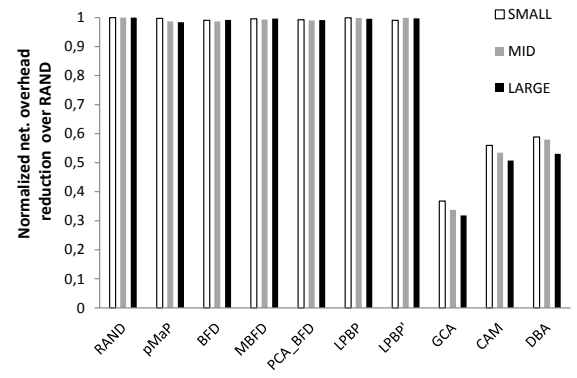

Fig. 12. Normalized network overhead reduction against RAND

besides the GCA, DBA, CAM and CPAM variations, all of the proposed algorithms do not take into account the network load incurred between the VMs when making the placement decisions. By delving into the raw data, we noticed that the BFD, MBFD, pMaP, LPBP variants, RAND, and PCA-BFD algorithms recorded the worst performance in some cases. We must note that we performed extra experiments to corroborate the above. As observed, the results of the GCA were always superior (up to 65% better performance against the bin packing algorithms) compared to the rest algorithms, with the CAM, CPAM, and DBA variations as the next best candidates. We must note that GCA achieved up to 18% better results compared to the CAM. The aforementioned phenomenon is justified by the fact that the CAM migrates the VMs by choosing one VM at a time. On the other extreme, the GCA may migrate a batch of VMs. Therefore, the CAM may not identify beneficial VM migrations due to the interdependencies between the VMs. In that way, the VMs are locked-in to the corresponding host servers. Lastly, we also observed that there were small ad-hoc differences when the capacity ratio varied.

The experimental results are summarized as follows. In terms of energy consumption, LPBP' achieved the best performance, with GCA following closely. At the other extreme, regarding network overhead, the superiority of GCA is clearly evident, with CAM following closely.

VI. Conclusions and Future Work

In this work, we addressed the problem of optimizing network traffic incurred within the system due to the communicational dependencies between VMs. The aforementioned optimization takes place in an energy-efficient manner as evidenced by the experimental evaluation.

We have identified the following three directions for future work: (i) SLA violations could be considered as another dimension in our problem; (ii) tackling the problem in an online fashion by proposing online solutions; and (iii) considering the creation and deletion of VM replicas.

Acknowledgment

Samee U. Khan's work was supported (while serving at) the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, P. Merle, "Elasticity in Cloud Computing: State of the Art and Research Challenges," *IEEE Transactions on Services Computing*, Vol. 11 (2), pp. 430-447, 2018.

[2] 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.

[3] 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.

[4] K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, M. Iqbal, C.-Z. Xu, and A. Y. Zomaya, "Quantitative Comparisons of the State of the Art Data Center Architectures," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 12, pp. 1771-1783, 2013.

[5] K. Cheng, Y. Bai, Y. Zhao, Y. Ma, D. Lu, Y. Peng, M. Zhou, "HV2M: A Novel Approach to Boost inter-VM Network Performance for Xen-based HVMs," Journal of Systems and Software, vol. 114, pp. 54-68, 2016.

[6] 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.

[7] 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.

[8] H. Duan, C. Chen, G. Min, Y. Wu, "Energy-aware Scheduling of Virtual Machines in Heterogeneous Cloud Computing Systems," *Future Generation Computer Systems*, Vol. 74, pp. 142-150, 2017.

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

[10] K. Gai, M. Qiu, H. Zhao, "Energy-aware Task Assignment for Mobile Cyber-enabled Applications in Heterogeneous Cloud Computing," *Journal of Parallel and Distributed Computing*, Vol. 111, pp. 126-135, 2018.

[11] 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.

[12] P. A. Garcia, J. M. M. Fernandez, J. L. A. Rodrigo, R. Buyya, "Proactive Power and Thermal Aware Optimizations for Energy-Efficient Cloud Computing," 2017.

[13] J. Guerra, W. Belluomini, J. Glider, K. Gupta, H. Pucha, "Energy Proportionality for Storage: Impact and Feasibility," *ACM SIGOPS*, 2010.

[14] L. He, D. Zou, Z. Zhang, C. Chen, H. Jin, S. A. Jarvis, "Developing Resource Consolidation Frameworks for Moldable Virtual Machines in Clouds," *Future Generation Computer Systems*, vol. 32, pp. 69-81, 2014.

[15] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, N. Mckeown, "ElasticTree: Saving Energy in Data Center Networks", *USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, 2010

[16] 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.

[17] 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.

[18] 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.

[19] A. Kansal, F. Zhao, N. Kothari, A.A. Bhattacharya, "Virtual Machine Power Metering and Provisioning," *International Symposium on Cloud Computing*, 2010.

[20] A. N. Khan, M. L. M. Kiah, S. U. Khan, and S. A. Madani, "Towards Secure Mobile Cloud Computing: A Survey," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1278-1299, 2013.

[21] A. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A Survey of Mobile Cloud Computing Application Models," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 393-413, 2014.

[22] D. Kliazovich, P. Bouvry, F. Granelli, N. L. S. da Fonseca, "Energy Consumption Optimization in Cloud Data Centers," *Wiley, Cloud Services, Networking and Management*, 2015.

[23] R. Koller, A. Verma, A. Neogi, "WattApp: An Application Aware Power Meter for Shared Data Centers," *International Conference on Autonomic Computing*, 2010.

[24] 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.

[25] P. Mahadevan, P. Sharma, S. Banerjee, P. Ranganathan, "A Power Benchmarking Framework for Network Devices," *NETWORKING*, 2009

[26] 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.

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

[28] G. B. Mertzios, M. Shalom, A. Voloshin, P. W. H. Wong, S. Zaks, "Optimizing Busy Time on Parallel Machines," *Theoretical Computer Science*, Vol. 562, pp. 524-541, 2015.

[29] 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.

[30] N. Tziritas, M. Koziri, A. Bachtsevani, T. Loukopoulos, G. Stamoulis, S. U. Khan, C.Z. Xu, "Data Replication and Virtual Machine Migrations to Mitigate Network Overhead in Edge Computing Systems," IEEE Transactions on Sustainable Computing, vol. 2 (4), pp. 320-332, 2017.

[31] 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.

[32] N. Tziritas, S. U. Khan, C.-Z. Xu, T. Loukopoulos, S. Lalis, "On Minimizing the Resource Consumption of Cloud Applications using Process Migrations," *Journal of Parallel and Distributed Computing*, vol. 73 (12), pp. 1690-1704, 2013.

[33] N. Tziritas, S. U. Khan, T. Loukopoulos, S. Lalis, C.-Z. Xu, P. Lampsas, "Single and Group Agent Migration: Algorithms, Bounds, and Optimality Issues," *IEEE Transactions on Computers*, vol. 63 (12), pp. 3143-3161, 2014.

[34] N. Tziritas, T. Loukopoulos, S. Lalis, P. Lampsas, "GRAL: A Grouping Algorithm to Optimize Application Placement in Wireless Embedded Systems," *International Symposium on Parallel and Distributed Processing (IPDPS)*, 2011.

[35] N. Tziritas, T. Loukopoulos, S. Lalis, P. Lampsas, "On Deploying Tree Structured Agent Applications in Networked Embedded Systems," *International European Conference on Parallel Processing (Euro-Par)*, 2010.

[36] N. Tziritas, C. Z. Xu, T. Loukopoulos, S. U. Khan, Z. Yu, "Application-aware Workload Consolidation to Minimize both Energy Consumption and Network Load in Cloud Environments," *IEEE International Conference on Parallel Processing (ICPP)*, 2013.

[37] N. Tziritas, S. U. Khan, T. Loukopoulos, S. Lalis, C.-Z. Xu, K. Li, A. Zomaya, "Online Inter-Datacenter Service Migrations," IEEE Transactions on Cloud Computing, DOI: 10.1109/TCC.2017.2680439.

[38] S. Verma, P. Ahuja, A. Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," *Middleware*, 2008.

[39] L. Wang, G. von Laszewski, J. Dayal, F. Wang, "Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS,"*IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2010.

[40] C. Yang, M. Yu, F. Hu, Y. Jiang, Y. Li, "Utilizing Cloud Computing to Address Big Geospatial Data Challenges," *Computers, Environments and Urban Systems*, Vol. 61, pp. 120-128, 2017.

[41] Z. Zong, A. Manzanares, B. Stinar, X. Qin, "Energy-Aware Duplication Strategies for Scheduling Precedence-Constrained Parallel Tasks on Homogeneous Clusters," *IEEE Transactions on Computers*, Vol. 60 (3), pp. 360-374, 2011.