Exploiting CPU-Load and Data Correlations in Multi-Objective VM Placement for Geo-Distributed Data Centers

Ali Pahlevan Pablo Garcia del Valle David Atienza
Embedded Systems Laboratory (ESL), EPFL
Email: {ali.pahlevan, pablo.garciadelvalle, david.atienza}@epfl.ch

Abstract—**Cloud computing has been proposed as a new paradigm to deliver services over the internet. The proliferation of cloud services and increasing users' demands for computing resources have led to the appearance of geo-distributed data centers (DCs). These DCs host heterogeneous applications with changing characteristics, like the CPU-load correlation, that provides significant potential for energy savings when the utilization peaks of two virtual machines (VMs) do not occur at the same time, or the amount of data exchanged between VMs, that directly impacts performance, i.e. response time.**

This paper presents a two-phase multi-objective VM placement, clustering and allocation algorithm, along with a dynamic migration technique, for geo-distributed DCs coupled with renewable and battery energy sources. It exploits the holistic knowledge of VMs characteristics, CPU-load and data correlations, to tackle the challenges of operational cost optimization and energy-performance trade-off. Experimental results demonstrate that the proposed method provides up to 55% operational cost savings, 15% energy consumption, and 12% performance (response time) improvements when compared to state-of-the-art schemes.

I. Introduction

Ever increasing demands for computing and growing number of clusters and servers in data centers (DCs) have ramped up power consumption world-wide. In this context [1], DC providers tend to use geo-distributed DCs to reduce costs, which are multiple DCs built in different geographical locations, and connected through the network. They are coupled with renewable energy sources to mitigate the carbon emission rate and their dependency on energy from the grid.

Among the energy reduction techniques, virtual machine (VM) consolidation [2] is one of the widely used methods which packs VMs into the minimal number of active servers. The process of placing a set of VMs on a server requires not only that the total size of VMs' load does not exceed the servers' capacity [3], but also analyze other factors that influence how suitable they are for co-location. In particular, two key factors to consider are data and CPU-load correlations. Data correlation refers to the dependency between each two VMs due to the amount of data that they need to exchange [4], and CPU-load correlation indicates if their CPU utilizations coincide during a certain time interval [5].

Based on the user's demands, virtualized DCs or cloud computing host heterogeneous services and applications that lead to very different computation and communication patterns [6]. For instance, scale-out applications (e.g. web search, MapReduce, etc.) feature different characteristics compared to high-performance computing (HPC) services. They present higher variability in CPU utilization (fast-changing loads) due to their dependency on the number of clients/queries. Additionally, they provide high data correlation, e.g., a query often requires parallel communication within VMs to return the most relevant results in web search.

This work has been partially supported by the YINS RTD project (no. 20NA21_150939), funded by Nano-Tera.ch with Swiss Confederation Financing and scientifically evaluated by SNSF, and the EC FP7 GreenDataNet STREP project (Agreement No. 609000). We would like to thank Maurizio Rossi and Davide Brunelli for the discussions about the design of the renewable energy controller.

However, data correlation is an important aspect missing from many previous works. In the multiple DCs problem, [4], [7], [8] take into account communication among VMs in single-path traffic flows to minimize network traffic and response time. Nonetheless, in practice, two VMs regularly exchange information in both directions with different amounts (bidirectional data correlation), and these amounts change at runtime depending on real-time information.

Regarding CPU-load correlation, the authors of [5] demonstrate that, having detailed information about the applications characteristics, as opposed to using stationary CPU-load values for the VMs (e.g. peak or average values), gives the opportunity to further reduce the energy consumption of a single DC. Therefore, efficient DC management is a challenging problem, since these correlation constraints indicate opposed goals: highly data-correlated VMs should be clustered together, while highly CPU-load correlated VMs should be placed apart.

Moreover, the complexity of these factors increases dramatically in geo-distributed DCs, where we need to consider inter-DC VMs migration and price diversities while maximizing the renewable and battery energy utilizations. These emerging modern DCs require innovative approaches for the operational cost (the cost of the energy from the grid) optimization and balancing of energy and performance.

To the best of our knowledge, this work is the first to propose a multi-objective VM placement for green geo-distributed DCs exploiting CPU-load and bidirectional data correlations in one problem. Compared to previous studies, the contributions of this work are as follows:

- We jointly incorporate CPU-load and data correlations to address the energy-performance trade-off. We consider bidirectional data correlations which change at runtime.
- We propose a two-phase controller along with a migration technique that splits the complex VM placement problem into clustering and allocation phases.
- We define and formulate this problem for geo-distributed DCs connected through a network topology to address the energy-performance trade-off and operational cost minimization while maximizing the use of renewable and

battery energies. Our experimental results show that, by applying our proposed method, up to 55% operational cost savings, 15% energy consumption reduction, and 12% performance improvement can be obtained compared to state-of-the-art approaches.

- We optimize the whole problem to find the best solution (unrestricted problem) based on load and renewable forecast information. Therefore, we are able to adopt a low-complexity rule-based green controller to compensate the difference between real and forecasted information.

The remainder of this paper is organized as follows. Sect. II reviews related work. In Sect. III, we describe the network and latency model used for geo-distributed DCs. In Sect. IV, we introduce the proposed optimization scheme. Sect. V presents our experimental results, followed by conclusions in Sect. VI.

II. Related Work

We classify previous studies on scheduling for single and geo-distributed DCs according to different objectives.

A. Energy-aware VM placement

When deciding to place a set of VMs on a server, many works only check that the total size of VMs' load does not exceed the server's capacity [3]. Hence, various server consolidation solutions are proposed based on per-VM workload characteristics, i.e., the peak, off-peak, and average utilization of VMs [2]; whereas, there are a few studies [5], [9], [10] to consider also other attributes of the VMs, like the CPU-load correlation, to achieve further energy savings. In [10], Meng et al. proposed a VM sizing technique that pairs two uncorrelated VMs into a super-VM by predicting the workloads. However, once the super-VMs are formed, this solution does not consider dynamic changes, which limits further energy savings. In [5], a CPU-load correlation-aware solution is proposed for a single DC based on the First-Fit-Decreasing heuristic to separate CPU-load correlated VMs. They also exploit server's dynamic voltage and frequency scaling (DVFS) techniques to achieve further energy savings. This approach cannot be used for online management at multiple DCs scale due to its high computational overhead. Moreover, these approaches do not consider operational costs and performance optimization.

B. Network-aware VM placement

To provide better network resource usage and, thereby, improve the performance of service applications, certain algorithms, [4], [8], take into account the communication among VMs. However, previous works assume that data dependencies are given in the form of a directed acyclic graph (DAG). Differently, in practice, there are often cyclic communication scenarios, where two VMs regularly exchange information in both directions. As a result, the authors in [6] proposed two heuristic algorithms to address bidirectional data communication under time-varying traffic demands. The first one, 2PCCRS, can be applied only if the network topology is a tree. The second one, GH, has more freedom during VM placement and is applicable for different types of network topology. However, both approaches neglect the main providers' objectives, including operational costs and energy consumption.

C. Operational costs

The use of geo-distributed DCs allows designers to minimize the electricity cost by exploiting dynamic workload allocation based on the renewable sources and temporal and regional diversities of electricity price [11], [12], [13]. However, data transfer among VMs is an important aspect missing from these problem formulations which directly affects the response time and user experience. In addition, an energy-efficient management is missing from these works based on existing CPU-load correlation to achieve more energy and cost savings. Authors in [14] presented a workload assignment and migration technique to minimize the costs of energy consumed by IT and cooling equipment considering the fluctuations of electricity price and the variability of the DCs' Power Usage Effectiveness (PUE). Zhao et al. [15] addressed the problem of dynamic pricing by designing an efficient online job scheduling and server provisioning in each DC to maximize the time-average overall profit of the cloud provider with respect to delay constraints. In [16], the authors addressed the same problem targeting energy costs and the delay based on the DC distance. Gu et al. [17] presented an optimization problem, which is formulated as a mixed-integer linear programming problem and then solved by a computation-efficient heuristic algorithm to minimize electricity cost via DC resizing. However, without the consideration of the characteristics of the workload, these research works are sub-optimal to minimize operational cost, energy consumption, and response time.

To the best of our knowledge, jointly CPU-load and data correlation effects on VM placement have not been previously considered for green geo-distributed DCs to optimize operational costs, energy consumption and response time.

III. Network and Latency Model

Wide-area data transmission is the major contributor to the network costs [18]. Therefore, in order to model it accurately, our algorithm considers intra-DC local links with bandwidth (B_L) (to access the network-attached storage), and inter-DC connections, modeled as a full mesh backbone network topology with bandwidth (B_{bb}). The global links are modeled in the presence of bit error rates (BERs) and their probabilities (P_{BER}) associated to the data transmission, the speed of light, and distance between DCs.

To compute the total latency for both migrating a set of VMs (according to VMs size) at time slot T and data communication during the time interval of $(T, T+1)$ from multiple DCs to a specific DC, we take into account two parts: 1) local and global latency for the i^{th} source DC, i.e. L_l^i and $L_g^{i,j}$ respectively, to transmit information through the local and global networks to the j^{th} destination DC, and 2) local latency for the j^{th} destination DC (L_l^j) to transmit data collected from other DCs to its storage. Equation 1 represents the total (worst-case) latency for the j^{th} destination DC (L_t^j) as the summation of the maximum latency between source DCs for transmitting the corresponding data through their dedicated local and global links, and local latency inside destination DC. N_{DC} is the total number of DCs.

$$L_t^j = max_i(L_l^i + L_g^{i,j}) + L_l^j \qquad i = 1 \; to \; N_{DC} \; and \; i \neq j \quad (1)$$

Local latency of the i^{th} source DC is dependent on the volume of data $(Vol^{i,j})$ ready to be transferred to the j^{th} destination DC and its local bandwidth (B_L^i). Therefore, each source DC local latency is calculated as:

$$L_l^i = (Vol^{i,j})/B_L^i \quad (2)$$

The local latency of the j^{th} destination DC is related to the total volume of data received from the multiple source DCs and its local bandwidth (B_L^j), computed as:

$$L_l^j = \sum_{i=1, i\neq j}^{N_{DC}} Vol^{i,j}/B_L^j \quad (3)$$

The global latency includes propagation latency as a primary source and data latency with respect to the volume of data being transmitted. Propagation latency is a function of how long the data takes to travel at the speed of light (S_l) from source to destination (distance: $Dist_{i,j}$). Data latency ($L_e^{i,j}$) is a function of the effective bandwidth $(B_e(t))$ and the $(BER(t))$ (corrupted data must be resent). Hence, the global latency is calculated as:

$$L_g^{i,j} = Dist_{i,j}/S_l + L_e^{i,j} \quad (4)$$

To calculate the data latency (L_e) in the presence of transmission errors, first we calculate the effective bandwidth and, then, we fragment the transmission into the necessary number of time steps. Algorithm 1 describes this process analytically.

Algorithm 1 Global Data Latency (L_e) w.r.t BER

```
1:  while true do
2:      $B_e(t) = (1 - BER(t)) \cdot B_{bb}, \quad BER(t) \propto P_{BER(t)}$
3:      if $Vol^{i,j} \leq B_e(t)$ then
4:          $L_e = L_e + Vol^{i,j}/B_e(t)$
5:          Break
6:      else
7:          $Vol^{i,j} = Vol^{i,j} - B_e(t)$
8:          $L_e = L_e + 1$
9:      end if
10: end while
```

IV. Proposed Optimization Method

In this section, we first define the problem of two-phase VM placement. Then, the proposed algorithm is presented.

A. Problem definition

The problem consists of VMs clustering for DCs (global dispatching controller), and allocating clusters to the servers (local controller). At each time slot T, first the global controller receives the VMs' loads from the previous time interval $[T-1, T)$, data communications, renewable forecast, available battery energy and grid price from each DC; all of them are non-stationary parameters that change dynamically. Then, we cluster the VMs (available VMs in the system and newly arrived), for each DC. After clustering, at local level, distributed in each DC, the VMs are allocated to the minimal number of servers. During the time interval of $[T, T+1)$, the local green controllers in each DC compensate the difference between real and forecasted load and renewable information.

B. The proposed VM placement algorithm

While optimal VM placement is an NP-complete problem, we propose a two-phase algorithm with low computational overhead that can be applied in real-time.

1) Global phase - VMs clustering

We split this phase into three different steps. First, at time slot T, all the VMs available in the system are represented as points in a two dimensional plane (2D plane). Based on the data and CPU-load correlation properties, as highly data-correlated VMs should be clustered together while highly CPU-load correlated VMs should be placed apart, a function is defined to calculate attraction and repulsion forces between each two VMs. Equation 5 calculates the force from i^{th} to j^{th} VM ($F_t^{i,j}$) as a function of attraction force ($F_a^{i,j}$) based on the data correlation ($Corr_{data}^{i,j}$) normalized as $[-1, 0)$, and repulsion force ($F_r^{i,j}$) based on the CPU-load correlation ($Corr_{cpu}^{i,j}$) normalized as $(0, 1]$. The attraction force from i^{th} to j^{th} VM is different from j^{th} to i^{th} VM due to the consideration of bidirectional data correlation and calculated as amount of data two VMs exchange. The repulsion force is computed as a worst-case peak CPU utilization when the peaks of two VMs coincide during the last time slot. α denotes a weighting factor for energy and performance trade-off calculation.

$$\begin{cases} F_a^{i,j} = Corr_{data}^{i,j} \\ F_r^{i,j} = Corr_{cpu}^{i,j} \end{cases} \Rightarrow \; F_t^{i,j} = \alpha \; . \; F_a^{i,j} + (1-\alpha) \; . \; F_r^{i,j} \quad (5)$$

Initially, at time slot 0, all the points are distributed in the 2D plane. Then, the resultant forces in the X (F_x^i), and Y (F_y^i) directions are calculated amongst points ($\theta_{j,i}$ is the angle) and, as a result, the points are remapped in the 2D plane with new coordinates $(Loc_x^i(k), Loc_y^i(k))$ at each iteration k as follows:

$$\begin{cases} F_x^i = \sum_{j=1, j\neq i}^{N_{vm}} F_t^{j,i} . \cos(\theta_{j,i}) \\ F_y^i = \sum_{j=1, j\neq i}^{N_{vm}} F_t^{j,i} . \sin(\theta_{j,i}) \\ Loc_x^i(k) = Loc_x^i(k-1) \; + \; 0.5 \; . \; F_x^i(k) \; . \; t^2 \\ Loc_y^i(k) = Loc_y^i(k-1) \; + \; 0.5 \; . \; F_y^i(k) \; . \; t^2 \end{cases} \quad (6)$$

where N_{vm} and t denote the number of VMs (points) available in the system and time period of displacement, respectively, since: $displacement = \; 1/2 \; acceleration \; . \; time^2$.

The process is iterated until the cost function ($Cost^{AR}$) of the current iteration k, Eq. 7, yields a lower value than that calculated in the previous one $k-1$. We also fix a maximum number of iterations to avoid a convergence time overhead.

$$Cost_k^{AR} = \sum_{i=1}^{N_{vm}} \sum_{j=1, j\neq i}^{N_{vm}} F_t^{i,j} . (d_k^{i,j} - d_{k-1}^{i,j}) \quad (7)$$

where $d_k^{i,j}$ depicts the distance between i^{th} and j^{th} points at iteration k. This function demonstrates if there is either an attraction force between each pair of points ($F_t^{i,j} < 0$), and they are attracted to each other ($d_k^{i,j} - d_{k-1}^{i,j} < 0$), or a repulsion force ($F_t^{i,j} > 0$), and they separate away. The final location of all the VMs becomes the initial position for the next time slot.

In the second step, we first define a capacity cap (in Joules) per each DC (cluster) to minimize the operational cost, computed according to the available battery energy, renewable energy forecast, grid price and DCs power consumed during the last previous time slot; i.e., last-value predictor.

Then, we utilize a modified version of the k-means algorithm to cluster VMs with respect to each cluster capacity cap, VMs load, and the distance between two VMs obtained from the repulsion and attraction phase in the 2D plane. In the modified k-means, the initial centroid of each cluster is calculated based on the last position of points available in that cluster in the previous time slot. In this step, we do not consider network latency.

At the last step, we revise the modified k-means output to meet the hard time constraint for migrating VMs across DCs based on their size as described in Algorithm 2. The output of the modified k-means creates two queues per cluster (DC): *outgoing* and *incoming*. The first one contains the candidates to be migrated outside, to another DC, sorted in descending order according to their distances from the corresponding cluster's centroid (Q_{out}). The second one contains the candidates to be migrated to this DC sorted in ascending order (Q_{in}).

Algorithm 2 Migration Step - Modified K-means Output Revision

```
Input: Outgoing and incoming queues
Output: VMs migration actions
 1: $Q^i_{out} \leftarrow$ Sort $i^{th}$ DC outgoing queue based on VMs distances from its centroid (Descending order)
 2: $Q^i_{in} \leftarrow$ Sort $i^{th}$ DC incoming queue based on VMs distances from its centroid (Ascending order)
 3: $i \leftarrow 1$    Initial DC
 4: while ($Q_{in}$ and $Q_{out}$ are not $NULL$ for all DCs) & (Latency constraint is not violated for all connections) do
 5:     if $R_i < Cap_i$ then
 6:         $VM$ = Head ($Q^i_{in}$)
 7:         $j \leftarrow$ Current DC of $VM$
 8:         if $L^i_t <$ Latency constraint then
 9:             Migrate $VM$ from $j^{th}$ DC to $i^{th}$ DC
10:             Update $i^{th}$ and $j^{th}$ DCs' load ($R_i$ and $R_j$)
11:         end if
12:         Erase $VM$ from $Q^i_{in}$ and $Q^j_{out}$
13:     else if $R_i \geq Cap_i$ then
14:         $VM$ = Head ($Q^i_{out}$)
15:         $j \leftarrow$ Destination DC of $VM$
16:         if $L^j_t <$ Latency constraint then
17:             Migrate $VM$ from $i^{th}$ DC to $j^{th}$ DC
18:             Update $i^{th}$ and $j^{th}$ DCs' load ($R_i$ and $R_j$)
19:             Erase $VM$ from $Q^i_{out}$ and $Q^j_{in}$
20:             $i \leftarrow j$    Move to destination DC
21:         else
22:             Erase $VM$ from $Q^i_{out}$ and $Q^j_{in}$
23:         end if
24:     end if
25: end while
```

The algorithm first selects one DC (i^{th} DC) and checks if its previous load (R_i) is less than its capacity cap (Cap_i). Then, it selects the first VM from the head of the incoming queue of the cluster (Head(Q^i_{in})). If this VM can be migrated in less than *latency constraint* time, the migration is executed; otherwise, we erase it from the queue and select the next VM. We repeat and update the DC's load until there is either no VM to accept or the load of the DC becomes more than the cap (lines 5∼12). In this later case (lines 13∼24), we select the VM from the head of the outgoing queue of the current cluster (Head(Q^i_{out})) which has the maximum distance to the centroid. If this VM can be migrated, we check the current load of the destination cluster and repeat this process there. Otherwise, we select the next one in the cluster. This algorithm iterates until violating the latency constraint for all DCs or there is no action to do. Unallocated VMs that have been available in the system will stay in their previous DC, and unallocated new VMs are assigned to the DCs determined from the modified k-means step without the consideration of the network latency constraint. In this case, we have tried to find the best solution for migrating the appropriate VMs when the number of migrations is bounded. We also prevent network bottlenecks made by one DC when the other DCs need to migrate their VMs to the same destination DC.

2) Local phase - VMs allocation

At local phase, the VMs of each cluster are allocated to servers of their corresponding DC, and the optimal frequency for each server is computed. We use only CPU-load correlation to allocate VMs to the minimum number of servers, since data correlation (and migrations) mainly contribute to inter-DC network bottlenecks [7], [18]. Hence, we base our implementation on the best algorithm [5] for VMs allocation.

3) Green controller

The proposed VM placement algorithm reduces the dependency on grid energy based on the load and renewable forecast. Therefore, we require a low-complexity green controller to compensate the difference between real and forecast information with respect to the current electricity price of DCs.

After allocating all the VMs to servers at time slot T, the green controller inside each DC manages the energy sources during the time interval of $[T, T+1)$ based on the real renewable energy and DC energy consumption.

When the available renewable energy is more than the DC energy consumption, we use this free energy for the DC and the excess energy is stored in the battery bank. Otherwise, during the high price period, we use the whole renewable energy for the DC's load and, for the remaining load, we discharge the battery considering its depth of discharge (DoD). During the low price periods, we charge the battery by grid energy and we do not use it for the DC.

V. Experimental Results

A. Setup

We consider three different DCs located in Europe: Lisbon (DC1), Zurich (DC2) and Helsinki (DC3), along with their distances (for the network model), time zone and two-level real electricity price scenario. Each DC contains 10 rooms and, each room, has 150, 100 and 50 servers for DC1, DC2 and DC3, respectively. Table I summarizes the number of servers, PV module size and lithium-ion battery capacity (with 50% of DoD, keeping the remaining capacity in case of outage) per DC. We target an Intel Xeon E5410 server consisting of 8 cores and two frequency levels (2.0GHz and 2.3GHz), and use the power model in [19]. For cooling power consumption, we use a time-varying PUE model, as in [20]. The DCs are connected through 100 Gb/s full duplex peer-to-peer optical fiber links, and the intranet uses 10 Gb/s full-duplex links. Global links experience a BER that is chosen randomly from the following distribution: 54% probability of 10^{-6}, 20% of 10^{-5}, 15% of 10^{-4}, 10% of 10^{-3}, and 1% of 10^{-2}.

In order to simulate a realistic scenario, DC VMs and energy demand, we sampled the VMs' utilization of a real DC every 5 seconds for one day, and extended it to 7 days by adding

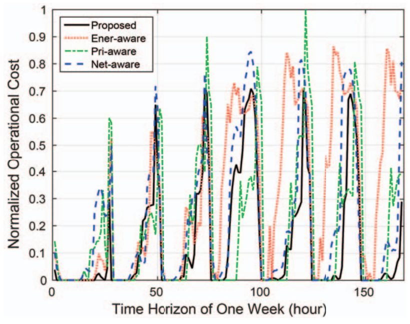

Fig. 1. Normalized operational cost for time horizon of one week.

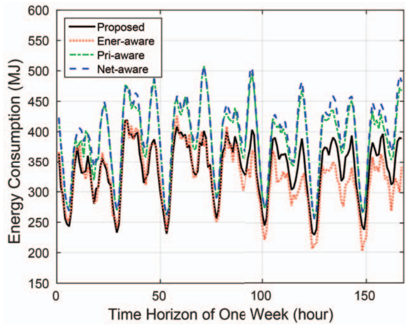

Fig. 2. Energy consumed by DCs for time horizon of one week.

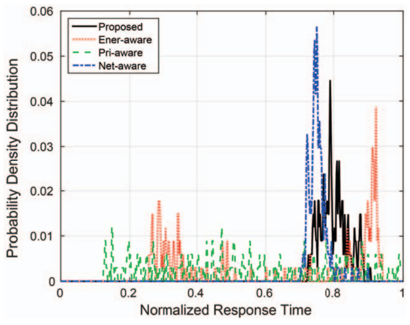

Fig. 3. Probability distribution of normalized response time in one week.

statistical variance with the same mean as the original traces. For renewable forecast, we implemented the algorithm in [21].

Arrival and life-time of each VM, given in time slots, are generated by poisson and exponential distributions, respectively. Data correlation between each pair of VMs is generated by a log-normal distribution with the mean of 10 MB and uniform variance selection in the range of $[1,4]$ [22]. For migration, the size of the VMs are in the range of 2, 4, and 8 GB according to the distribution of 60%, 30%, and 10%.

Finally, the global and local controllers are invoked every one hour, and the green online controller in each DC is invoked every 5 seconds. We also take into account a hard time constrain (latency constraint in Algorithm 2) for migrating the VMs across DCs through the network. A value of 98% for the quality of service (QoS) guarantees that the migration of VMs will take less than the 2% of the time slot.

B. Results

We compared our algorithm against three state-of-the-art approaches that are the best in their class to optimize operational costs, energy consumption and performance, respectively:

- Cost-aware approach (*Pri-aware*) [17].
- Energy-aware VM allocation (*Ener-aware*) [5].
- Network-aware VM placement (*Net-aware*) [6].
- *Proposed:* the proposed multi-objective VM placement.

All the mentioned methods are used jointly with the same local green controller to manage battery and renewable energy.

1) Operational cost

Figure 1 shows the operational cost normalized by the worst-case value among the mentioned methods in the time horizon of one week: 55, 25 and 35% cost savings for the proposed method compared to Ener-aware, Pri-aware and Net-aware, respectively. Proposed clusters the VMs by specifying a load cap for different DCs based on the grid price and available renewable and battery energy. It outperforms the other algorithms when a local energy-aware VM allocation is utilized to further reduce DCs' dependency on grid energy. Differently, Ener-aware uses CPU-load correlation to reduce energy consumption and cost in each DC locally but, globally, it cannot efficiently cluster and dispatch VMs for right DCs based on available renewable energy, battery status and grid price. In Pri-aware, the VMs are packed and placed onto DCs and servers with the lowest current grid price, but it neglects to maximize free energies usage. Finally, the Net-aware approach provides load balancing across DCs which in turn leads to better exploiting free energies (renewable and battery) compared to Ener-aware and Pri-aware. However, this algorithm does not consider the electricity price diversities and neglects to utilize an energy-efficient management to reduce its dependency on the grid.

TABLE I
DCs NUMBER OF SERVERS AND ENERGY SOURCES SPECIFICATION.

DC	Number of Servers	PV Capacity (KWp)	Battery Capacity (KWh)
*DC*1	1500	150	960
*DC*2	1000	100	720
*DC*3	500	50	480

2) Energy consumption

Figure 2 shows the hourly energy consumed by the DCs for one week. The total energy consumption is 57, 55, 65 and 67 GJ for the Proposed, Ener-aware, Pri-aware and Net-aware methods, respectively. The results show 12 and 15% energy improvements for our proposed algorithm compared to Pri-aware and Net-aware due to the consideration of the CPU-load correlation between VMs, that places highly CPU-load correlated VMs apart, in different DCs and servers. This favors consolidation and leads to power savings by lowering the number of active severs and their operating frequency. On the other hand, the Ener-aware approach first uses the FFD clustering heuristic, placing VMs into the first DC in which its load capacity fits, and then packs the VMs into the minimal number of active servers based on the CPU-load correlation. Hence, the DC local controller finds a better mapping of VMs to servers when most of the VMs are in the same DC. Our algorithm, however, tries to find the best VMs clusters per each DC based on the CPU-load and data correlations and determined DCs' capacity cap. Although these correlations indicate opposed goals for energy and performance, Ener-aware only results in 3% energy improvement compared to our multi-objective algorithm, while significantly degrading operational costs and performance (shown in the next section).

3) Performance

In this context, performance is defined as the response time of the VMs; i.e., the amount of time they have to wait for data from other VMs in the network. Figure 3 shows the

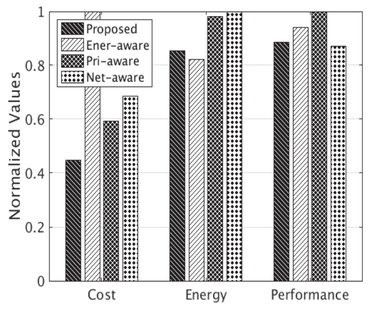

Fig. 4. Total cost, energy and performance.

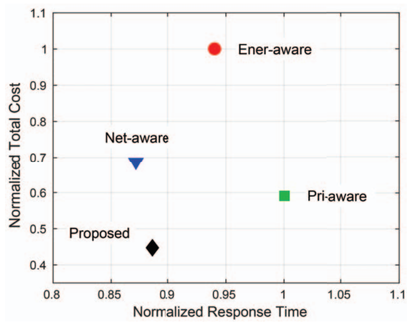

Fig. 5. Cost-Performance trade-off.

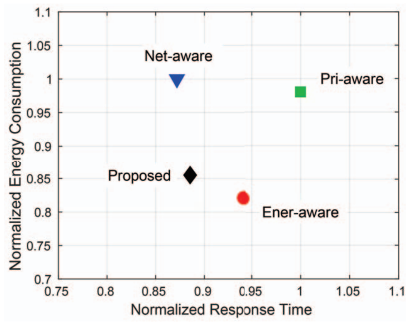

Fig. 6. Energy-Performance trade-off.

probability density distribution of the response time in one week. Note that the response time results are normalized with respect to the worst-case value among the methods. As a result, Proposed and Net-aware encompass a range of response time with higher average and lower variance compared to Ener-aware and Pri-aware methods. The goal of Net-aware is to balance the network across DCs, which in turn leads to better worst-case and higher average response time (both for times of high and low data demands between VMs). However, when compared to Proposed, Net-aware only achieves 2% performance improvement. Ener-aware and Pri-aware tend to place the VMs on a lower number of DCs, which leads to unbalanced network traffic with bigger fluctuations and, accordingly, lower average response time. However, since DCs providers typically consider worst-case response time in their Service Level Agreements (SLA contracts), the proposed algorithm results in up to 12% performance improvement compared to state-of-the-art approaches.

4) Trade-offs discussion

The experimental results confirm that, by having a holistic approach, we can obtain better trade-offs in the problem of VM placement. Figures 4, 5 and 6 summarize the benefits of Proposed: In the first place, Fig. 4 depicts the totals, showing up to 55, 15 and 12% improvements for operational cost, energy consumption and performance, respectively. Then, Fig. 5 shows the cost-performance trade-off, with Proposed providing 25 and 12% improvements for cost and response time, respectively, compared to Pri-aware. In comparison with Net-aware, it achieves 35% cost savings while it leads to only 2% performance degradation. Finally, Fig. 6 exhibits the energy-performance trade-off: our algorithm results in 6% performance improvement with a 3% energy overhead compared to Ener-aware; and it provides 15% energy savings and 2% performance degradation compared to Net-aware.

VI. Conclusion

In this paper, we proposed a novel method to tackle the challenges of operational cost optimization and energy-performance trade-off on resource-constrained green geo-distributed DCs. We introduced a two-phase multi-objective VM placement algorithm along with a dynamic migration technique that exploit the holistic knowledge of VMs characteristics. The first phase, i.e. global controller, clusters VMs for each DC considering time-varying VMs CPU-load and data correlations and the status of DC energy sources. The second phase, i.e. local controller, allocates the VMs of each DC cluster to servers exploiting CPU-load correlation. Finally, experimental results showed that, using our proposed method, up to 55, 15 and 12% improvements can be obtained for operational cost, energy consumption and performance, respectively, compared to state-of-the-art approaches.

References

[1] Y. Zhang and et al., "Greenware: Greening cloud-scale data centers to maximize the use of renewable energy," in *Middleware*. Springer, 2011.
[2] E. Pakbaznia and et al., "Minimizing data center cooling and server power costs," in *ISLPED*, 2009.
[3] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE TPDS*, 2013.
[4] D. de Oliveira and et al., "A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds," *JGC*, 2012.
[5] J. Kim and et al., "Correlation-aware virtual machine allocation for energy-efficient datacenters," in *DATE*, 2013.
[6] O. Biran and et al., "A stable network-aware vm placement for cloud systems," in *CCGRID*, 2012.
[7] F. Researc, "The future of data center wide-area networking," 2010.
[8] S. Pandey and et al., "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *AINA*, 2010.
[9] A. Verma and et al., "Server workload analysis for power minimization using consolidation," in *USENIX*, 2009.
[10] X. Meng and et al., "Efficient resource provisioning in compute clouds via vm multiplexing," in *ICAC*, 2010.
[11] L. Rao and et al., "Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment," in *INFOCOM*, 2010.
[12] J. Li and et al., "Towards optimal electric demand management for internet data centers," *IEEE TSG*, 2012.
[13] X. Xiang and et al., "Greening geo-distributed data centers by joint optimization of request routing and virtual machine scheduling," in *UCC*, 2014.
[14] K. Le and et al., "Reducing electricity cost through virtual machine placement in high performance computing clouds," in *ACM SC*, 2011.
[15] J. Zhao and et al., "Dynamic pricing and profit maximization for the cloud with geo-distributed data centers," in *INFOCOM*, 2014.
[16] P. X. Gao and et al., "It's not easy being green," in *SIGCOMM*, 2012.
[17] L. Gu and et al., "Joint optimization of vm placement and request distribution for electricity cost cut in geo-distributed data centers," in *ICNC*, 2015.
[18] A. Amokrane and et al., "Greenhead: virtual data center embedding across distributed infrastructures," *IEEE TCC*, 2013.
[19] M. Pedram and et al., "Power and performance modeling in a virtualized server system," in *ICPPW*, 2010.
[20] J. Kim and et al., "Free cooling-aware dynamic power management for green datacenters," in *HPCS*, 2012.
[21] C. Bergonzini and et al., "Comparison of energy intake prediction algorithms for systems powered by photovoltaic harvesters," *MEJ*, 2010.
[22] D. Dias and L. Costa, "Online traffic-aware virtual machine placement in data center networks," in *GIIS*, 2012.