Energy Cost Optimization for Geographically Distributed Heterogeneous Data Centers

Eric Jonardi*, Mark A. Oxley*, Sudeep Pasricha*†, Anthony A. Maciejewski*, Howard Jay Siegel*†

*Department of Electrical and Computer Engineering
†Department of Computer Science
Colorado State University, Fort Collins, Colorado 80523, USA
{eric.jonardi,mark.oxley,sudeep,aam,hj}@colostate.edu

***Abstract*—The proliferation of distributed data centers has recently motivated researchers to study energy cost minimization at the geo-distributed level. Researchers have been using models for time-of-use (TOU) electricity pricing and renewable energy sources to help reduce energy costs when performing geographical workload distribution, but have made oversimplifying assumptions at the data center level. Important considerations such as the thermal, power, and co-location interference effects within each data center have a large impact on the performance of workload management techniques. By designing three techniques that possess varying amounts of knowledge of such information, we compare and quantify the benefits of considering detailed models at the data center level, and demonstrate that our best heuristic can on average achieve a cost reduction of 37% compared to state of the art prior work.**

***Keywords*—geo-distributed data centers; workload management; power-aware computing; memory interference.**

I. Introduction

The strong success and extensive growth of cloud computing has resulted in data center operators geographically distributing their data center locations (e.g., Google [1]). Distributing data centers geographically offers benefits to the clients (e.g., low latency due to shorter communication distances). However, a strong motivating factor for data center operators to geographically distribute their data centers is to reduce operating expenditures by exploiting time-of-use (TOU) electricity pricing [2], and reducing electricity costs is now a focus of data center management.

Relocating workload among geo-distributed data centers offers several benefits. First, workloads can be shifted to locations in different times zones to concentrate workload in the regions with the lowest electricity prices at that time. Second, an opportunistic distribution of the workload among data centers during periods of peak demand can allow individual nodes to run in slower but possibly more energy-efficient performance states (P-states), further reducing electricity costs. Due to the ever-increasing electricity consumption of data centers, the use of on-site renewable energy sources (e.g., solar and wind) has grown in recent years. Adding on-site renewable power can provide additional opportunities for geographical load distribution (GLD) techniques to reduce electricity costs.

The goal of our research is to design techniques for geographical load distribution that will minimize energy cost for executing incoming workloads. We use detailed models of power, temperature, and co-location interference at each data center to provide more accurate information to the geo-distributed workload manager. This work applies to environments where there is information about the history of the types of tasks being executed (e.g., DigitalGlobe, Google, DoD). By considering TOU pricing and renewable power models at each data center, we design three new workload management techniques that assume varying degrees of co-location interference knowledge to distribute or migrate the workload to low-cost data centers at regular time intervals, while ensuring all of the workload completes. We compare to the state-of-the-art method [3], and show that our best heuristic can, on average, achieve a cost reduction of 37% comparatively. The contributions of this work are as follows:

- A new hierarchical framework for the GLD problem that considers cost-minimization workload management at both the geo-distributed and local heterogeneous data center level;
- A data center model that considers heterogeneous compute node types, P-states, node temperatures, cooling power, renewable power sources, and co-location interference;
- The design of three novel heuristics which possess varying degrees of co-location interference prediction knowledge to demonstrate and motivate the use of detailed models in workload management decisions.

II. Related Work

Workload distribution for geo-distributed data centers has been studied in [3, 4, 5, 6, 7, 8, 9]. Knowledge of TOU pricing is typically used to either minimize electricity costs across all geo-distributed data centers (e.g., [3, 4, 6, 7, 8, 9]), or to maximize profits when a revenue model is included for computing (e.g., [5]). A quality of service (QoS) constraint of some form is recognized in most of the aforementioned works, typically as a queuing delay constraint [4, 6, 8]. Others incorporate QoS violations into the cost function, where a monetary penalty is associated with violating queuing delay [5], latency [7], or migration [3] service level agreements (SLAs). The modeling detail varies significantly, some works include dynamic voltage and frequency scaling (DVFS) in decision making [6], some include power consumption of the cooling system in addition to the computing system [7, 8], others consider real-world TOU pricing data [5, 6], and one considers renewable energy sources at each data center location [3]. Our research includes all aforementioned modeling aspects to assist in workload management decisions: DVFS to exploit the power/performance tradeoffs of P-states, cooling system power to include thermal awareness and reduce cooling

This research was supported by NSF grants CCF-1302693 and CCF-1252500. This research used the CSU ISTeC Cray System supported by NSF Grant CNS-0923386. We also thank Hewlett Packard for donating servers.

cost, TOU pricing data from an actual electric company, and renewable power sources at each data center. To the best of our knowledge, our work is the first to encompass all of these aspects within the GLD problem. In addition, unlike any prior work in GLD, we consider co-location interference as part of our load distribution techniques; a phenomena that occurs when multiple cores within the same multicore processor are executing applications simultaneously and compete for shared resources (e.g., last-level cache or DRAM).

Similar to [3], our study considers a renewable energy source at each geo-distributed data center, a cooling system at each data center, and migration penalties associated with moving already-assigned workloads to different data centers. We differ significantly from [3] by including TOU electricity pricing traces, consideration of DVFS P-state decisions in our management techniques, and integrating interference caused by the co-location of multiple tasks to cores that share resources.

III. System Model

A. Geo-distributed Level

The goal of the geo-distributed resource manager (GDRM) is to minimize the total monetary cost of the system while servicing all requests. We divide time evenly into intervals called *epochs*. As an example, in this work an epoch is an hour of time (T^e), thus a 24-epoch period is a full day.

We assume that the beginning of each epoch is a steady-state scheduling problem where we assign *execution rates* of a set of I task types to D data centers. A task type $i \in I$ is characterized by its arrival rate AR_i, and its estimated computational speeds on each of the heterogeneous compute nodes in all P-states. The assignment problem at the geo-distributed level is to map execution rates for each task type i to each data center $d \in D$ such that total energy *cost* across all data centers is minimized, and the execution rates of all task types meet or exceed their arrival rates. For each epoch τ, we assign a desired data center execution rate $ER^{DC}_{d,i}$ for each task type i to each data center d such that the total execution rate for all task types exceed (or equal) the corresponding arrival rate, AR_i, thus ensuring the workload is completed. That is,

$$\sum_{d=1}^{D} ER^{DC}_{d,i}(\tau) \geq AR_i(\tau), \quad \forall i \in I. \tag{1}$$

B. Data Center Level

1) Overview: Each data center d houses NN_d compute nodes that are arranged in hot aisle/cold aisle fashion (Fig. 1), and a cooling system comprised of NCR_d computer room air conditioning (CRAC) units. A compute node n is of a heterogeneous compute node type, where node types vary in their execution speeds, power consumption characteristics, and number of cores, i.e., they are heterogeneous. Cores within a compute node are homogeneous, and each core is DVFS-enabled to allow independent configuration of its P-states. The number of cores in node n is NCN_n, and NT_k is the compute node type to which core k belongs.

2) Core Execution Rates: At each data center d, the sum of execution rates of all cores that are assigned to execute task type i must exceed or equal $ER^{DC}_{d,i}(\tau)$. We assume that we know the estimated computational speed (ECS) of any task of type i on a core of node type n in P-state p, ECS(i, n, p).

The execution rate of task type i on core k, $ER^{core}_{i,k}$, is the product of the assigned desired fraction of time core k spends executing tasks of type i, $DF_{i,k}(\tau)$, and the execution speed that core executes tasks of type i in P-state $PS_{i,k}(\tau)$. That is, the execution rate of task type i on core k is

$$ER^{core}_{i,k}(\tau) = DF_{i,k}(\tau) \cdot ECS(i, NT_k, PS_{i,k}(\tau)). \tag{2}$$

At the data center level, we assign $DF_{i,k}(\tau)$ and $PS_{i,k}(\tau)$ such that power is minimized (see Section III-B3), and the execution rates of all task types on cores in data center d meets or exceeds the execution rate assigned by the GDRM, ensuring that the arriving workload is fully executed. That is,

$$\sum_{n=1}^{NN_d} \sum_{k=1}^{NCN_n} ER^{core}_{i,k} \geq ER^{DC}_{i,d} \quad \forall i \in I, \forall d \in D. \tag{3}$$

3) Power Model: The power consumption of a compute node consists of the overhead ("idle") power consumption and dynamic power consumed by cores executing tasks. We define O_n as the overhead power consumption of compute node n. Let APC(i, NT_k, $PS_{i,k}(\tau)$) be the average power consumed by core k in a node of type NT_k when executing tasks of type i in P-state $PS_{i,k}(\tau)$ during epoch τ. The power consumption of node n during epoch τ, $PN_n(\tau)$, is

$$PN_n(\tau) = O_n + \sum_{k=1}^{NCN_n} \sum_{i=1}^{I} APC(i, NT_k, PS_{i,k}(\tau)) \cdot DF_{i,k}(\tau). \tag{4}$$

The power consumed by a CRAC unit, $PCR_{d,c}(\tau)$, is a function of the heat removed at that CRAC unit and the Coefficient of Performance (CoP) of the CRAC unit [10], calculated using Eq. 5 of [11].

4) Renewable Energy Model: Solar energy E^{solar}_d and wind energy E^{wind}_d (both kWh) are calculated for each data center d as an average per epoch τ (Eqs. 5 and 6 are from [12]). A^{solar}_d is the total active area of all solar panels, and A^{wind}_d is the total swept rotor area of all wind turbines. The solar-to-electricity and wind-to-electricity conversion efficiency are given by α and β, respectively. Lastly, $s_d(\tau)$ is the average solar irradiance, $v_d(t)$ is the wind speed, and $\rho_d(\tau)$ is the air density, as averaged for data center d during epoch τ. The total renewable energy, $R_d(\tau)$, available at data center d during epoch τ is the sum of the wind and solar energy available at that time. We use these models with historical data to predict the renewable power available at each data center, given by

$$E^{solar}_d(\tau) = \alpha \cdot A^{solar}_d \cdot s_d(\tau) \cdot T^e, \tag{5}$$

$$E^{wind}_d(\tau) = \beta \cdot \frac{1}{2} \cdot A^{wind}_d \cdot \rho_d(\tau) \cdot v_d(\tau)^3 \cdot T^e, \tag{6}$$

$$R_d(\tau) = E^{solar}_d(\tau) + E^{wind}_d(\tau). \tag{7}$$

5) Thermal Model: Using the notion of thermal influence indices [13] that were derived using computational fluid dynamics simulations, we can calculate the steady-state temperatures at compute nodes and CRAC units in each data center. Because we assume the same physical layout for each of the data centers (Fig. 1), we derive these thermal influence indices for one data center, and assume they are the same for all other data centers.

The outlet temperature of each compute node is a function of the inlet temperature, the power consumed, and the air flow

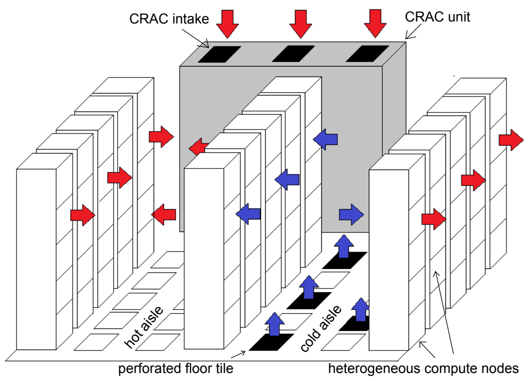

Fig. 1. Data center in hot aisle/cold aisle configuration [11].

rate of the node. The inlet temperature of each compute node is a function of the outlet temperatures of each CRAC unit and the outlet temperatures of all compute nodes [11]. Lastly, for all nodes the inlet temperature of each node is constrained to be less than or equal to the red line temperature (maximum allowable node temperature).

6) System Electricity Cost: The electricity price at data center d during epoch τ is defined as $E_d^{price}(\tau)$. Let Eff_d be the approximation of power overhead in data center d due to the inefficiencies of power supply units and uninterruptable power supplies. The total electricity cost for data center d during epoch τ, $PC_d(\tau)$, is defined as

$$PC_d(\tau) = E_d^{price}(\tau) \cdot \left[\left(\sum_{c=1}^{NCR_d} PCR_{d,c}(\tau) + \sum_{n=1}^{NN_d} PN_n(\tau) \right) \cdot Eff_d - R_d(\tau) \right]. \tag{8}$$

7) Node Activation/Deactivation Cost: At each data center, the number of nodes of each node type that are in use changes frequently between epochs. Inactive nodes are placed in a sleep state, but entering and exiting this sleep state takes a non-negligible amount of time. Each node that is active is considered to be active for the entire epoch, which requires that any node transitioning to/from a sleep state do so during the epoch following/prior the current epoch, respectively.

For each data center d, let $N_{d,j}^{start}(\tau)$ be the number of nodes of type j that are inactive during epoch τ and active during epoch $\tau+1$, and let $N_{d,j}^{stop}(\tau)$ be the number of nodes of type j that are active during epoch $\tau-1$ and inactive during epoch τ. Let P_j^S, P_j^D, and P_j^{Sleep} be the average static power, average peak dynamic power, and average sleep power for node type j, respectively, with the average CPU utilization of node type j defined as ϕ_j^E. Let the coefficient to approximate $\underline{C}$RAC $\underline{u}$nit $\underline{p}$ower at data center d be CUP_d. We assume each data center contains the same number of nodes, however each data center is heterogeneous in the sense that the number of nodes belonging to each node type among data centers varies. Let J_d be the set of node types in data center d. Let T^S be the time required for a node to transition to/from a sleep state. Recall that T^e is the duration of an epoch. The node $\underline{a}$ssignment $\underline{c}$ost AC_d for data center d during epoch τ is calculated as

$$\begin{aligned} AC_d(\tau) = &\sum_{j \in J_d} E_d^{price}(\tau) \cdot Eff_d \cdot \left(1 + \frac{1}{CUP_d}\right) \cdot \frac{T^S}{T^e} \\ &\cdot \left(\phi_j^E P_j^D + P_j^S - P_j^{Sleep}\right) \cdot \left(N_{d,j}^{start}(\tau) + N_{d,j}^{stop}(\tau)\right). \end{aligned} \tag{9}$$

8) Co-Location Interference Model: Tasks competing for shared memory in multicore processors can cause severe performance degradation, especially when competing tasks are memory-intensive [14]. The memory-intensity of a task refers to the ratio of last-level cache misses to the total number of instructions executed [15]. We adapt a linear regression model from [15] that uses a set of features (i.e., inputs) based on the current applications assigned to a multicore processor to predict the execution time of a target application i on core k. These features are $A_{i,k}$, the number of applications co-located on that multicore processor, $B_{i,k}$, the base execution time, $C_{i,k}$, the clock frequency, D_k, the average memory intensity of all applications on that multicore processor, and $E_{i,k}$, the memory intensity of application i on core k.

In a linear model, the output is a linear combination of all features and their calculated coefficients. We classify the task types into memory-intensity classes on each of the node types, and calculate the coefficients for each memory-intensity class using the linear regression model. If we denote u, v, w, x, and y as the linear model coefficients for feature symbols A, B, C, D, and E, respectively, plus the constant term z, the equation for $\underline{c}$o-located $\underline{e}$xecution $\underline{t}$ime of a task type i of memory-intensity class m on core k ($CET_{i,k}(\tau)$) is

$$\begin{aligned} CET_{i,k}(\tau) = &\, u_{m,k} \cdot A_{i,k} + v_{m,k} \cdot B_{i,k} + w_{m,k} \cdot C_{i,k} \\ &+ x_{m,k} \cdot D_k + y_{m,k} \cdot E_{i,k} + z_{m,k}\,. \end{aligned} \tag{10}$$

The execution rate is the reciprocal of the execution time. Therefore the $\underline{c}$o-located $\underline{e}$xecution $\underline{r}$ate for task type i on core k, $CER_{i,k}^{core}(\tau)$, is $1/CET_{i,k}(\tau)$. The total execution rate for task type i in epoch τ is therefore given by

$$CER_i(\tau) = \sum_{d=1}^{D} \sum_{k=1}^{NC_d} CER_{i,k}^{core}(\tau). \tag{11}$$

To allocate tasks to cores when considering co-location interference, some of our techniques use knowledge of $CER_{i,k}^{core}$ to judge actual execution rates more accurately than techniques that do not consider co-location interference. When considering co-location the execution rate constraint becomes

$$\sum_{k=1}^{NC_d} CER_{i,k}^{core}(\tau) \geq ER_{d,i}^{DC}(\tau), \quad \forall i \in I, \forall d \in D. \tag{12}$$

IV. Heuristic Descriptions

A. Problem Statement

The GDRM allocates the incoming workload to specific nodes within each data center. The GLD problem is NP-hard [3], and therefore we propose three heuristics for GDRM (FDLD-TAO, FDLD-CL, and GALD-CL), with each having different levels of detail of the system available to it. The system as a whole is under-subscribed, i.e., all tasks must be completed without dropping. The objective of a GDRM is to minimize monetary electricity cost of the geo-distributed system (the sum of Eq. 8 across all data centers) while ensuring the workload is completed (Eqs. 1 and 12).

B. Force Directed Load Distribution Heuristics

$\underline{F}$orce-$\underline{d}$irected $\underline{l}$oad $\underline{d}$istribution (FDLD) is a variation of force-directed scheduling [16]. We adapt the FDLD proposed in [3], designated FDLD-SO, to our rate-based allocation environment, and propose two new FDLD heuristics (FDLD-TAO and FDLD-CL) that each possess different amounts of co-location interference information to solve this problem.

In FDLD-SO, to account for co-location interference performance degradation and let the FDLD technique meet the execution rate constraint at a given data center (Eq. 12), we give the technique simple over-provisioning (FDLD-SO) to compensate for performance degradation due to co-location. This technique over-provisions all task types equally by scaling estimated task execution rates by the factor ϕ^C. The FDLD-TAO technique improves upon FDLD-SO by using task aware over-provisioning to estimate co-location effects for each task type by a factor specific to each task type i, ϕ_i^C. For both FDLD-SO and FDLD-TAO, the degree of over-provisioning (ϕ^C and ϕ_i^C, respectively) is determined empirically. Lastly, the FDLD-CL heuristic uses the co-location models given in Sec. III-B8 to account for co-location effects when calculating task execution rates. All versions of FDLD consider a system implementation where the computing time of each core in a node is evenly divided among its assigned tasks.

The fundamental operation of all FDLD variants is described in Algorithm 1. To generate the initial solution, every node in every data center in every epoch is assigned to execute all task types (step 1). Each iteration of the FDLD removes one instance of one task type from a single node, selecting the task to remove that would result in the lowest total system force, F^S (steps 3-20). F^S is the sum of the execution rate forces ($F^{ER}(\tau)$) and cost forces ($F^C(\tau)$) across all epochs.

The execution rate force F^{ER} is the ratio of task execution rate (calculated during steps 6-11) to task arrival rate. Task execution rate is a function of the P-state of the node the task is executing on, but the FDLD is not designed to make DVFS decisions to set the execution rates of task types, and therefore an average execution rate must be determined for all task types using the average node utilization factor ϕ_j^E for each node type j. Let $ER_{j,i}(P_{MAX})$ and $ER_{j,i}(P_0)$ be the execution rates of task type i running on a single core of a node of type j in the highest numbered P-state and lowest numbered P-state, respectively. Therefore, the equivalent single core execution rate $R_{j,i}$ of task type i on node type j is

$$R_{j,i} = ER_{j,i}(P_{MAX}) + [ER_{j,i}(P_0) - ER_{j,i}(P_{MAX})]\,\phi_j^E. \tag{13}$$

Let $N_{d,j}$ be the number of nodes of type j in data center d. Let $W_{d,j,m}(\tau)$ be the set of instances of task type i placed on node m of node type j in data center d during epoch τ. Let $Q_{d,j,i}(\tau)$ be the equivalent number of nodes of type j running task type i in data center d during epoch τ, given by

$$Q_{d,j,i}(\tau) = \sum_{m=1}^{N_{d,j}} \begin{cases} \frac{1}{|W_{d,j,m(\tau)}|} & \text{if } i \in W_{d,j,m}(\tau) \\ 0 & \text{else} \end{cases} \tag{14}$$

Let K_j be the number of cores in a node of type j. The average estimated execution rate $ER_{j,i}^E(\tau)$ of task type i on machine type j during epoch τ, when using either the FDLD-SO or FDLD-TAO versions, is given by

$$ER_{j,i}^E(\tau) = \sum_{d=1}^{D} \sum_{j \in J_d} K_j \cdot R_{j,i} \cdot F \cdot Q_{d,j,i}(\tau) \tag{15}$$

subject to the constraint

$$ER_{j,i}^E(\tau) \geq AR_i(\tau) \quad \forall i \in I. \tag{16}$$

To compensate for performance degradation due to co-location effects, node over-provisioning is accomplished by the factor F. F is replaced by either ϕ^C or ϕ_i^C in Eq. 15 when using either FDLD-SO or FLDB-TAO, respectively.

Algorithm 1 Pseudo-code for FDLD heuristics

```
1. allocate an instance of each task type to every node in
   every data center in every epoch
2. while
3.   for each node with tasks still allocated to it
4.     for each task type on the node
5.       temporarily remove task type from node
6.       if FDLD-CL
7.         estimate execution rates using Eq. 11 ($CER_i$)
8.       else if FDLD-TAO
9.         estimate execution rates using Eq. 15 and $\phi_i^C$
10.      else if FDLD-SO
11.        calculate execution rates using Eq. 15 and $\phi^C$
12.      estimate power costs using Eq. 18
13.      calculate $F^S$ from $F^{ER}$ and $F^C$
14.      if execution rate constraints are not violated (Eq.
         12 for FDLD-CL, Eq. 16 FDLD-SO & FDLD-TAO)
15.        add to set of possible task removal operations
16.      restore task type to node
17.  if set of possible task removal operations is empty
18.    break
19.  else
20.    choose and implement the task type removal
       operation that would result in the lowest $F^S$
21. end while
22. calculate final execution rates ($CER_i(\tau)$, $\forall i \in I$, $\forall \tau \in N^\tau$)
23. calculate final cost from sum of power costs and allocation
    costs ($PC_d(\tau)$ and $AC_d(\tau)$, $\forall d \in D$, $\forall \tau \in N^\tau$)
```

The execution rate force F^{ER} is calculated using

$$F^{ER}(\tau) = \sum_{i \in I} e^{\left(\frac{Z}{AR_i(\tau)} - 1\right)} - 1. \tag{17}$$

When considering the FDLD-CL heuristic, the term Z is replaced by $CER_i(\tau)$, and is replaced by $ER_{j,i}^E(\tau)$ when using either FDLD-SO or FDLD-TAO. Observe that $F^{ER}(\tau)$ decreases to zero as the ratio of Z to $AR_i(\tau)$ decreases to one.

Recall that $R_d(\tau)$ is the renewable power available at data center d during epoch τ. For all FDLD variants, let $PC_d^E(\tau)$ be the estimated power cost at data center d during epoch τ, calculated as

$$PC_d^E(\tau) = E_d^{price}(\tau) \cdot \left(\sum_{j \in J^d} \sum_{m=1}^{N_{d,j}} P_{d,j,m}^E \cdot \left(1 + \frac{1}{CUP_d}\right) \cdot Eff_d - R_d(\tau) \right) \tag{18}$$

where

$$P_{d,j,m}^E = \begin{cases} P_j^{Sleep} & \text{if } |W_{d,j,m}| = 0 \\ \phi^j P_j^D + P_j^S & \text{else} \end{cases}. \tag{19}$$

Let $C_d^{actual}(\tau)$ be sum of power ($PC_d^E(\tau)$) and allocation ($AC_d(\tau)$) costs incurred at data center d during epoch τ. Let $C_d^{max}(\tau)$ be the maximum real power cost possible at data center d, calculated using

$$C_d^{max}(\tau) = E_d^{price}(\tau) \cdot \sum_{j \in J_d} N_{d,j} \cdot \left[\phi^j P_j^D + P_j^S\right]. \tag{20}$$

The cost force F^C can then be calculated with

$$F^C(\tau) = \sum_{d=1}^{D} e^{\left(\frac{C_d^{actual}(\tau)}{C_d^{max}(\tau)}\right)} - 1. \tag{21}$$

Observe that the value of F^C goes to zero as the ratio of $C_d^{actual}(\tau)$ to $C_d^{max}(\tau)$ decreases to zero.

Let N^{τ} be the total number of epochs being considered. The total system force across all epochs, F^S, is calculated as

$$F^S = \sum_{\tau=1}^{N^{\tau}} F^{ER}(\tau) + F^C(\tau). \tag{22}$$

C. Genetic Algorithm Heuristic

We also designed a third heuristic; a genetic algorithm load distribution with full co-location awareness (GALD-CL). The GALD-CL heuristic (Algorithm 2) has two parts: a genetic algorithm based GDRM and a greedy heuristic serving as the fitness function of the genetic algorithm. The GALD-CL assigns fractions of the global task arrival rate to each of the data centers in the simulation (step 3), with the arrival rates of each task type i at each data center d acting as the genes of the chromosomes. Using the task arrival rates assigned to each data center by the genetic algorithm at the geo-distributed level, the local greedy heuristic assigns tasks types to execute on specific nodes (steps 5-15). If the greedy heuristic finds that the task arrival rate assigned to a data center exceeds the capacity of that data center (step 16), the global arrival rates are adjusted slightly and the chromosome is evaluated once again (steps 5-15), with further adjustments made to the global allocations within the chromosome until a valid solution can be reached. The greedy heuristic has full knowledge of the entire system model, including the co-location models and task-node power models, allowing it to make better placement decisions.

The GALD-CL heuristic addresses two potential shortcomings of the FDLD variants. First, the nature of the FDLD variants prevents them from making of DVFS decisions. The greedy heuristic in the GALD-CL approach chooses the most efficient P-state for each task type on each node type [11]. Second, the FDLD variants are susceptible to becoming trapped in local minima. The genetic algorithm portion of the GALD-CL approach intrinsically enables escape from local minima, allowing a more complete search of the solution space.

Algorithm 2 Pseudo-code for GALD-CL heuristic

```
1. create an initial population of chromosomes
2. while within time limit do
3.   perform selection, crossover and mutation to create new
     chromosomes
4.   for each new chromosome, evaluate:
5.     for each data center
6.       find most efficient P-state for all task type/node type
         pairs
7.       sort all task type/node type pairs by efficiency
8.       while power constraint not violated do
9.         choose first task type/node type pair
10.        assign 100% desired fraction of time for selected
           task type to a single core from selected node type
11.        remove core from future consideration
12.        if no cores within selected node type available
13.          remove task type/node type pair from use
14.        set CRAC outlet temperatures to hottest
           temperatures such that thermal constraints are met
15.      end while
16.    if solution is invalid
17.      modify chromosome, return to step 5
18.  trim population (with elitism)
19. end while
20. take final allocation from the best chromosome
21. calculate final execution rates ($CER_i(\tau)$, $\forall i \in I$, $\forall \tau \in N^{\tau}$)
22. calculate final cost from sum of power costs and allocation
    costs ($PC_d(\tau)$ and $AC_d(\tau)$, $\forall d \in D$, $\forall \tau \in N^{\tau}$)
```

TABLE I. Node Processor Types Used in Experiments

Intel processor	# cores	L3 cache	frequency range
Xeon E3-1225v3	4	8MB	0.8 - 3.20 GHz
Xeon E5649	6	12MB	1.60 - 2.53 GHz
Xeon E5-2697v2	12	30MB	1.20 - 2.70 GHz

V. Simulation Results

A. Experimental Setup

Experiments were conducted for groups of four, eight, and sixteen data centers. The site locations for data centers were selected so that each group would have a fairly even east coast to west coast distribution to better exploit TOU pricing and renewable power. Each data center consists of 4,320 nodes arranged in four aisles, and is heterogeneous within itself, having nodes from either two or three of the node types given in Table I, with most locations having three nodes types and per-node core counts that range from 4-12 cores depending on the mix of node types.

The electricity prices used during experiments were taken directly from Pacific Gas and Electric (PG&E) Schedule E-19 [17]. Each data center had an installed renewable power generating capacity equivalent to 20% of the maximum power consumption of the location during the month with the highest generated power for that location. Renewable power data was obtained from [18], where each location uses either wind power, solar power, or a combination of the two.

Sleep power for all nodes is calculated as a fixed percentage of static power for each node type, assumed to be 16% based on a recent study of node power states [19]. The average node utilization factor used during FDLD allocations, ϕ^E, is set as 0.75. The coefficient to approximate CRAC unit power at data center d (CUP_d) was determined empirically by simulating workloads of multiple levels at each data center location, and its value ranged between 1.43 and 2.08 for the different configurations. The time of each epoch τ was set to be one hour. The time required to transition a node to or from a sleep state, T^S, was assumed to be five minutes.

The GALD-CL heuristic was limited to a run time of one hour for each epoch it was solving for, to mimic the representative time of each epoch. The FDLD heuristics for four, eight, and sixteen locations completed on average in one, four, and thirteen minutes per epoch simulated, respectively.

Each of five task types is representative of a different benchmark from the PARSEC benchmark suite. Task execution times and co-located performance data were obtained from running the benchmark applications on the nodes listed in Table I [15]. Synthetic task arrival rates were constructed that follow a sinusoidal pattern, peaking during business hours and declining during the evening and until the next morning.

B. Monetary Cost Comparison of Heuristics

Our first set of experiments compared the cost associated with using the four heuristics described in Section V. These experiments used a data center group consisting of four locations and estimated costs over a 24-hour period (Fig. 2). It can be observed that the FDLD-CL technique, using the co-location models, performs the best of the FDLD variants for provisioning the minimum number of nodes necessary to meet execution rate requirements. The FDLD-SO technique performed the worst, severely over-provisioning nodes. The GALD-CL heuristic outperformed all other approaches. While not performing as well as well as the GALD-CL, the FDLD variants do have the advantage of reaching a solution more quickly, which may be beneficial in some cases.

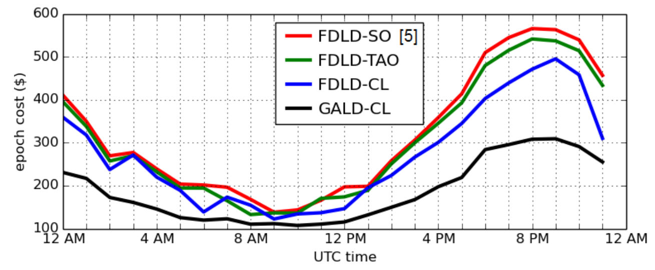

Fig. 2. System costs across twenty four epoch period for each heuristic, four locations.

C. Workload Type Analysis

The experiment in Section V-B used a workload that was a mix of memory-intensive and CPU-intensive tasks types. Fig. 3 shows experiments for the FDLD-CL and GALD-CL heuristics for a group of four data centers where two additional workload types have been added: one where all of the tasks are highly memory-intensive (using data from *canneal*, *cg*, *ua*, *sp*, and *lu* benchmarks), and one where the tasks are highly CPU-intensive (using data from *fluidanimate*, *blackscholes*, *bodytrack*, *ep*, and *swaptions* benchmarks). The composition of data center workloads can vary greatly and can impact the resource requirements, and these experiments show that the techniques presented in this work will perform well for a variety of workload types.

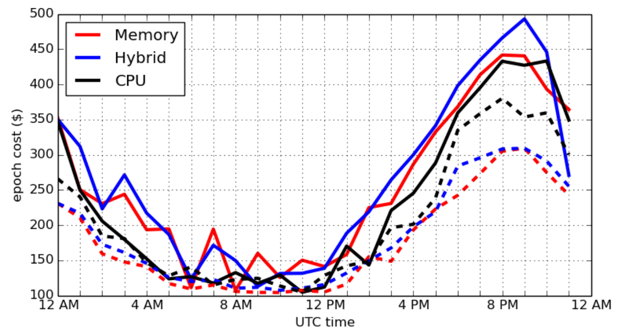

Fig. 3. System costs across twenty four epoch period for different workload types, for a group of four locations. FDLD-CL shown as solid line, and GALD-CL shown as dashed line.

D. Scalability Analysis

Additional experiments were conducted using groups of eight and sixteen separate data centers. For each of the data center group sizes, the average performance improvement of each technique over the FDLD-SO method is given in Table II. It should be noted that as the number of data centers in the group grows larger, the time for the FDLD variants to reach a solution increases, and the number of GALD-CL generations that can take place within the time limit decreases. As previously mentioned, the increase in the runtime of the FDLD heuristics was very manageable as the number of data centers in the group increased.

TABLE II. MONETARY COST REDUCTION COMPARED TO FDLD-SO [3]

Heuristic	4 data centers	8 data centers	16 data centers
FDLD-TAO	5.2%	4.6%	5.7%
FDLD-CL	14.2%	15.7%	18.8%
GALD-CL	39.7%	39.2%	36.8%

VI. Conclusions

We proposed three workload allocation heuristics for workload allocation across geographically distributed data centers. In this work, we explored adding different levels of knowledge of the system, particularly co-location interference, to geographical workload distribution algorithms. We demonstrated that including additional information about the co-location interference in the decision process of the heuristics resulted in a lower energy cost by reducing or eliminating node over-provisioning while still meeting all required workload execution rates. Our FDLD-CL and GALD-CL heuristics resulted on average in 10% and 37%, respectively, lower total cost than the prior work (represented by the FDLD-SO heuristic) [3]. In systems where the workload profile changes rapidly and therefore requires short epochs (a few minutes), we recommend FDLD-CL. When the workload profile is not changing rapidly and workload distribution decisions are given more time (an hour), GALD-CL is a more suitable technique.

References

[1] "Data center locations," http://www.google.com/about/datacenters/inside/locations/index.html.
[2] Y. Li *et al.*, "Operating cost reduction for distributed data centers," in *CCGRID '13*, May 2013, pp. 589–596.
[3] H. Goudarzi and M. Pedram, "Geographical load balancing for online service applications in distributed datacenters," in *CLOUD '13*, June 2013, pp. 351–358.
[4] L. Gu *et al.*, "Joint optimization of VM placement and request distribution for electricity cost cut in geo-distributed data centers," in *ICNC '15*, Feb. 2015, pp. 717–721.
[5] J. Zhao *et al.*, "Dynamic pricing and profit maximization for the cloud with geo-distributed data centers," in *INFOCOM '14*, Apr. 2014, pp. 118–126.
[6] L. Gu *et al.*, "Optimal task placement with QoS constraints in geo-distributed data centers using DVFS," *IEEE Trans. Comp.*, vol. 64, no. 7, pp. 2049–2059, June 2015.
[7] H. Xu, C. Feng, and B. Li, "Temperature aware workload management in geo-distributed data centers," *IEEE Trans. Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1743–1753, May 2015.
[8] M. Polverini *et al.*, "Thermal-aware scheduling of batch jobs in geographically distributed data centers," *IEEE Trans. Cloud Comp.*, vol. 2, no. 1, pp. 71–84, Apr. 2014.
[9] D. Mehta, B. O'Sullivan, and H. Simonis, "Energy cost management for geographically distributed data centres under time-variable demands and energy prices," in *UCC '13*, Dec. 2013, pp. 26–33.
[10] J. Moore *et al.*, "Making scheduling "cool": Temperature-aware workload placement in data centers," in *ATEC '05*, Apr. 2005, pp. 61–75.
[11] M. A. Oxley *et al.*, "Thermal, power, and co-location aware resource allocation in heterogeneous high performance computing systems," in *IGCC '14*, Nov. 2014, 10 pp.
[12] X. Deng *et al.*, "Eco-aware online power management and load scheduling for green cloud datacenters," *IEEE Sys. Journal*, no. 99, pp. 1–10, 2014.
[13] H. Bhagwat *et al.*, "Fast and accurate evaluation of cooling in data centers," *J. Electronic Packaging*, vol. 137, no. 1, Mar. 2015, 9 pp.
[14] S. Govindan *et al.*, "Cuanta: Quantifying effects of shared on-chip resource interference for consolidated virtual machines," in *SOCC '11*, Oct. 2011, pp. 1–14.
[15] D. Dauwe *et al.*, "A methodology for co-location aware application performance modeling in multicore computing," in *APDCM '15*, May 2015, pp. 434–443.
[16] P. Paulin and J. Knight, "Force-directed scheduling for the behavioral synthesis of asics," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 8, no. 6, pp. 661–679, Jun 1989.
[17] Pacific Gas and Electric Company, "Electric schedule e-19," Apr. 2015, http://www.pge.com/tariffs/tm2/pdf/ELEC_SCHEDS_E-19.pdf.
[18] NREL, "National solar radiation database," https://mapsbeta.nrel.gov/nsrdb-viewer/, April 2015.
[19] C. Isci *et al.*, "Agile, efficient virtualization power management with low-latency server power states," in *ISCA '13*, June 2013, pp. 96–107.