Minimizing Energy Costs for Geographically Distributed Heterogeneous Data Centers

Ninad Hogade, *Student Member, IEEE*, Sudeep Pasricha, *Senior Member, IEEE*, Howard Jay Siegel, *Fellow, IEEE*, Anthony A. Maciejewski, *Fellow, IEEE*, Mark A. Oxley, *Member, IEEE*, and Eric Jonardi, *Member, IEEE*

Abstract—The recent proliferation and associated high electricity costs of distributed data centers have motivated researchers to study energy-cost minimization at the geo-distributed level. The development of time-of-use (TOU) electricity pricing models and renewable energy source models has provided the means for researchers to reduce these high energy costs through intelligent geographical workload distribution. However, neglecting important considerations such as data center cooling power, interference effects from task co-location in servers, net-metering, and peak demand pricing of electricity has led to sub-optimal results in prior work because these factors have a significant impact on energy costs and performance. We propose a set of workload management techniques that take a holistic approach to the energy minimization problem for geo-distributed data centers. Our approach considers detailed data center cooling power, co-location interference, TOU electricity pricing, renewable energy, net metering, and peak demand pricing distribution models. We demonstrate the value of utilizing such information by comparing against geo-distributed workload management techniques that possess varying amounts of system information. Our simulation results indicate that our best proposed technique is able to achieve a 61 percent (on average) cost reduction compared to state-of-the-art prior work.

Index Terms—Geo-distributed data centers, workload management, memory interference, peak shaving, net metering

1 Introduction

The success of cloud computing has resulted in data center operators expanding and geographically distributing their data center locations, e.g., Google [1] and Amazon [2]. Distributing data centers geographically offers several benefits to the clients such as low latency due to shorter communication distances and service resiliency. 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 [3]. Electricity prices are not constant, but rather follow a TOU pricing model where the cost of electricity varies based on the time of day as demonstrated in Fig. 1. Electricity prices are higher when total electrical grid demand is high, and fall during periods when electrical grid demand is low [4], [5]. Beyond the TOU electricity costs, most utility providers also charge a flat-rate (peak demand) fee based on the highest (peak) power consumed at any instant during a given billing period, e.g., month [6], [7]. Reducing electricity costs has been a major focus of data center management, and has continued to grow in importance as the annual electricity expenditure for powering data centers has, in some cases, surpassed the costs of purchasing the equipment itself [8].

Relocating workloads among geo-distributed data centers is one effective approach to curb electricity expenditures. Workloads can be migrated to data centers located in different times zones with the goal of concentrating the workload in regions with the lowest TOU electricity and peak demand pricing available at that time. The allocation of workloads within a data center can also reduce electricity cost by exploiting dynamic voltage and frequency scaling (DVFS) and any heterogeneity across compute nodes (e.g., different power and performance characteristics).

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 rapidly in recent years. Several data center operators have already built or announced plans to build "green" data centers, i.e., data centers, that are completely (or at least partially) operated with the help of renewable energy. For example, a portion of Apple's data centers in North Carolina are powered by a 60 MW solar plant [10]. McGraw-Hill operates a data center using a 14 MW solar array [11]. Similar to these examples, major global data center providers, e.g., Microsoft [12], Google [13], and Facebook [14], have invested in green energy facilities. Some data center providers have begun to use locations with large amounts of renewable energy available to reduce energy costs, or even exploit net metering. *Net metering* is a billing mechanism that gives renewable energy customers credit on their utility bills for the excess clean energy they sell back to the grid [15]. Adding on-site renewable power

- *N. Hogade and A.A. Maciejewski are with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523. E-mail: {ninad.hogade, aam}@colostate.edu.*
- *S. Pasricha and H.J. Siegel are with the Department of Electrical and Computer Engineering and the Department of Computer Science, Colorado State University, Fort Collins, CO 80523. E-mail: {sudeep, hj}@colostate.edu.*
- *M.A. Oxley is with Numerica Corporation, Fort Collins, CO 80528. E-mail: moxley07@gmail.com.*
- *E. Jonardi is with Rockwell Automation, Milwaukee, WI 53204. E-mail: eric.jonardi@gmail.com.*

Manuscript received 3 Dec. 2017; accepted 13 Mar. 2018. Date of publication 3 Apr. 2018; date of current version 6 Dec. 2018.
(Corresponding author: Ninad Sanjay Hogade.)
Recommended for acceptance by P. Bouvry.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSUSC.2018.2822674

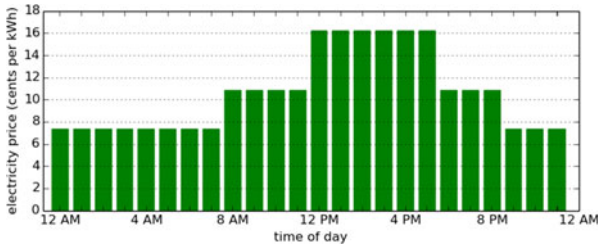


Fig. 1. TOU electricity pricing, PG&E Schedule E-19 [9].

and exploiting net metering can reduce electricity costs [16], peak grid power costs [17], or both [18], [19]. This can provide additional opportunities for geographical load distribution (GLD) techniques to reduce overall electricity costs.

The goal of our research is to design techniques for geographical load distribution that will minimize the energy cost for executing incoming workloads considering many aspects of the overall system. We use detailed models for data center cooling power, co-location interference, TOU electricity pricing, renewable energy, net metering, and peak demand pricing distribution to provide the most-accurate information possible to the geo-distributed workload manager. Co-location interference is a phenomena that occurs when multiple cores within the same multicore processor are executing tasks simultaneously and compete for shared resources, e.g., last-level cache or DRAM. Our work is highly useful for environments where historical execution information about the types of tasks executed is readily available. Examples of such environments exist in industry, e.g., commercial companies (DigitalGlobe, Google), military computing installations (Department of Defense), and government labs (National Center for Atmospheric Research).

By considering data center cooling power, co-location interference, TOU electricity pricing, renewable energy, net metering, and peak demand pricing distribution models, we design three new workload management techniques. These techniques assume varying degrees of co-location interference characteristics to distribute or migrate the workload to low-cost data centers at regular time intervals, while ensuring that all of the workload is completed. We compare these techniques to a state-of-the-art method [20], and our previous work [21], and show that our best proposed resource management heuristic can, on average, achieve a cost reduction of 61 percent. Key contributions in our work can be summarized as follows:

- a hierarchical framework for the GLD problem that considers cost-minimization oriented workload management at both the geo-distributed and local heterogeneous data center level;
- a new detailed data center model that exploits net metering and peak shaving, i.e., peak power reduction, by considering information about heterogeneous compute node-types, P-states, compute node temperatures, cooling power, TOU and peak demand pricing, renewable power sources, and performance degradation caused by co-location interference;
- the design of three resource management heuristics that possess varying degrees of co-location interference prediction information to demonstrate and motivate the use of detailed models in workload management decisions.

The rest of the paper is organized as follows. In Section 2, we review relevant prior work. Our system model is characterized in Section 3. Sections 4 and 5 describe our specific problem in detail and the heuristics we propose to solve it. The simulation environment is discussed in Section 6. Lastly, we analyze and evaluate the results of our approach in Sections 7 and 8, respectively.

2 Related Work

There have been many recent efforts proposing methods to minimize electricity costs across geo-distributed data centers, with the fundamental decisions of the optimization problem relying on a TOU electricity pricing model [22]. These models either use data analytics for pricing models or make predictions of electricity costs. Electricity costs are often much higher during peak hours of the day (typically 8 A.M. to 5 P.M.). The TOU electricity cost models, sometimes in combination with a model for revenue generated from computation of a workload, motivates the use of optimization techniques to minimize energy cost or, if provided a revenue model, to maximize total profit.

Workload distribution for geo-distributed data centers has been studied in prior work such as [20], [21], [23], [24], [25], [26], [27], [28]. Information about TOU pricing is typically used to either minimize electricity costs across all geo-distributed data centers, e.g., [20], [21], [23], [25], [26], [27], [28], or to maximize profits when a revenue model is included for computational work performed, e.g., [24]. A quality of service (QoS) constraint of some form is recognized in most of the aforementioned works. Typically, this is incorporated into the model as a queuing delay constraint [23], [25], [27]. Other works incorporate QoS violations into a cost function, where a monetary penalty is associated with violating service level agreements (SLAs) due to excessive queuing delay [24], latency [26], or migration penalties [20]. The detail of each model varies significantly among approaches in the prior work. Some works include DVFS in decision making [25], some include power consumption of the cooling system in addition to the computing system [26], [27], and others consider real-world TOU pricing data [24], [25].

Our research considers all of the aforementioned modeling aspects to assist in workload management decisions: (a) DVFS to exploit the power/performance trade-offs of P-states; (b) cooling system power and thermal properties to reduce cooling cost; (c) TOU and peak demand pricing data to reduce monetary electricity cost. To the best of our knowledge, our work is the first to integrate all of these aspects within the GLD problem. In addition, unlike any prior work in GLD, we consider performance degradation caused by co-location interference as part of our load distribution techniques.

With respect to energy usage, some studies have considered renewable energy for energy optimization of geo-distributed data centers [20], [21], [29], [30]. Similar to [19], our work uses information about TOU electricity pricing, peak demand pricing and peak shaving, renewable energy usage, and net metering. However, reference [19] proposes a solution for a single data center rather than for a group of geo-distributed data centers. Moreover, it does not consider heterogeneous compute node-types, different performance states of cores, compute node temperatures, cooling power, or co-location interference. Our work considers all these factors at a geo-distributed level. Similar to our work, reference [29] proposes a workload scheduling technique that uses both peak shaving and renewable energy prediction

models. However, it does not consider net metering and detailed cooling power and co-location interference models. Similar to [20] and our prior work in [21], 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 [20] by including TOU electricity pricing traces, considering DVFS P-state decisions, and integrating interference caused by the co-location of multiple tasks to cores that share resources in our management techniques. We extend our prior work significantly from [21] by including a peak demand price model and also exploiting peak shaving and net metering.

3 System Model

3.1 Overview

We propose a hierarchical framework for a geo-distributed resource manager (GDRM) that consists of a high-level manager to distribute incoming workload requests and migrate already-allocated requests to geographically distributed data centers. The goal of the GDRM is to minimize the total energy cost of the system while servicing all requests. Each data center has its own local workload management system that takes the workload assigned to it by the GDRM and maps requests to compute cores within the data center. We first describe the system model at the geo-distributed level and then provide further details into the models of components at the data center level. We provide a list of abbreviations and notations in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSUSC.2018.2822674.

3.2 Geo-Distributed Level Model

We consider a rate-based workload management scheme, where workload arrival rate can be predicted over the decision interval called an *epoch* [31], [32]. In our work, an epoch length T^e is one hour, and a 24-epoch period represents a full day. Over the course of an epoch, the workload arrival rates can be reasonably approximated as constant, e.g., the Argonne National Lab Intrepid log shows mostly-constant arrival rates over large intervals of time [33].

We assume that the beginning of each epoch represents a steady-state scheduling problem where we assign execution rates, i.e., reciprocal of the execution time, of a set of I workload task-types to D data centers. A task-type $i \in I$ is characterized by its arrival rate AR_i, and the estimated time required to complete a task of task-type i on each of the heterogeneous compute nodes in each P-state. The assignment problem at the geo-distributed level is to assign 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, with the constraint that the execution rates of all task-types meet their arrival rates, i.e., all tasks complete without being dropped or unexecuted. To fulfill this constraint, for each epoch τ, we assign a maximum 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 can be completed. That is

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

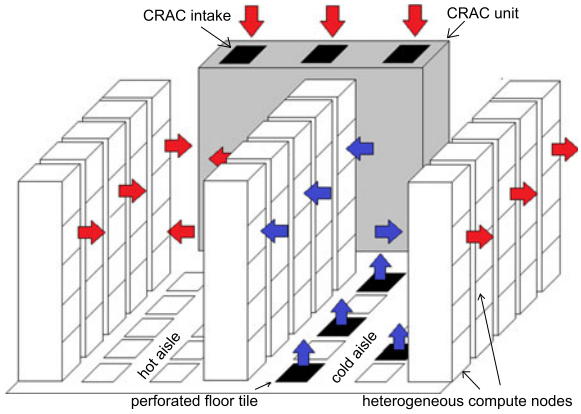


Fig. 2. Data center in hot aisle/cold aisle configuration [34].

3.3 Data Center Level

3.3.1 Organization of Each Data Center

Each data center d houses NN_d number of nodes that are arranged in hot aisle/cold aisle fashion (Fig. 2), and a cooling system comprised of NCR_d number of computer room air conditioning (CRAC) units. Heterogeneity exists among compute nodes, where nodes vary in their execution speeds, power consumption characteristics, and number of cores. 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 node type to which core k belongs.

3.3.2 Compute Core Execution Rates

Recall that our GDRM determines the distribution of tasks of each task-type among all data centers. 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, $ECS(i,n,p)$, determined using historical, experimental, or analytical techniques [35], [36].

For epoch τ, we assign a desired fraction $DF_{i,k}(\tau)$ of time each core k will spend executing tasks of type i and the P-state $PS_{i,k}(\tau)$ each core k is configured when executing tasks of type i. We assume tasks will run serially until completion. That is, a core sharing its time among multiple tasks implies that a scheduler will assign different tasks to execute on the core in such a manner that, over a long period of time (i.e., steady-state), the amount of time a core k spends executing a task of type i would equal its assigned $DF_{i,k}(\tau)$ value. The core execution rate $ER^{core}_{i,k}$ of tasks of 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 3.3.3), and the execution rates of all task-types on cores in all data centers equal the arrival rate ensuring that the arriving workload is fully executed. That is

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

3.3.3 Compute Node Power Model

The power consumption of a compute node consists of the static *overhead* power consumption (equal to the amount of

power consumed when the system is idle) and the additional dynamic power consumed when cores are executing tasks. We define O_n as the overhead power consumption of compute node n (a constant value independent of the workload resulting from system components such as storage, network interfaces). 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

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

3.3.4 Cooling Power Model

The heat generated by compute nodes is removed by the CRAC units. The airflow within the data center causes heat generated from nodes to propagate to other nearby nodes, thereby increasing the inflow temperature of those nodes. Using the notion of thermal influence indices [37] 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. 2), we use thermal influence indices derived for one data center layout based on an average workload that would be executed by the data center.

The outlet temperature of each compute node is a function of the inlet temperature, the power consumed, and the air flow 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 of the same data center [34]. The ASHRAE guidelines have designated the inlets of IT equipment as the common measurement point for temperature compliance [38], and therefore we consider thermal constraints at the inlets of compute nodes. 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).

The power consumed by a CRAC unit is a function of the heat removed at that CRAC unit and the Coefficient of Performance (CoP) of the CRAC unit [39]. Let ρ_d be the density of air and C_d be the specific heat capacity of air at data center d. Let $TC^{in}_{d,c}(\tau)$, $TC^{out}_{d,c}(\tau)$, and $AFC_{d,c}(\tau)$ be the inlet temperature, outlet temperature, and air flow rate, respectively, of CRAC unit c in data center d during epoch τ. Then the power consumed by this CRAC unit, $PCR_{d,c}(\tau)$, can be calculated as [39]

$$PCR_{d,c}(\tau) = \frac{\rho_d \cdot C_d \cdot AFC_{d,c}(\tau) \cdot \left(TC^{in}_{d,c}(\tau) - TC^{out}_{d,c}(\tau)\right)}{CoP_d\left(TC^{out}_{d,c}(\tau)\right)}. \quad (5)$$

3.3.5 Node Activation/Deactivation Power Overhead

At each data center, the number of nodes of each node-type that are in-use frequently changes among epochs. Inactive nodes are placed in a sleep state, but entering and exiting this sleep state takes some time due to the actions required in both hardware and software to transition the system between states. 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/before the current epoch, respectively.

For each data center d, let $N^{trans}_{d,j}(\tau)$ be the number of nodes of type j that activate or deactivate during epoch τ. Let P^{Sleep}_j be the average sleep power for node-type j. Let P^D_j be the average peak dynamic power for node-type j. It is calculated by averaging over all task-types the peak power for each task-type i executing on node-type j. The average node utilization of node-type j defined as μ_j. Let CoP of the CRAC unit at data center d be CoP_d. Without loss of generality, we assume that 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 activation/deactivation power ADP_d for data center d during epoch τ is then calculated as

$$ADP_d(\tau) = \sum_{j \in J_d} \left[\left(\mu_j P^D_j - P^{Sleep}_j\right) \cdot N^{trans}_{d,j}(\tau)\right] \cdot \left(1 + \frac{1}{CoP_d}\right) \cdot \frac{T^S}{T^e}. \quad (6)$$

3.3.6 Renewable Power Model

Each data center is equipped with and partially powered by a renewable energy source. Every location can have either solar power, wind power, or some combination of both. Solar power P^{solar}_d and wind power P^{wind}_d (both have units of kW) are calculated as [40] for each data center d as an average per epoch τ. The total renewable power, $PR_d(\tau)$, available at data center d during epoch τ is the sum of the wind and solar power available at that time. We use these models with historical data [41] to predict the renewable power available at each data center

$$PR_d(\tau) = P^{solar}_d(\tau) + P^{wind}_d(\tau). \quad (7)$$

3.3.7 Overall Data Center Power Model

Let Eff_d be an approximation of the power overhead coefficient in data center d due to the inefficiencies of power supply units. Eff_d is always greater than or equal to 1. The total non-renewable power consumed throughout data center d during epoch τ, $PD_d(\tau)$, is calculated as

$$PD_d(\tau) = \left(\sum_{c=1}^{NCR_d} PCR_{d,c}(\tau) + \sum_{n=1}^{NN_d} PN_n(\tau) + ADP_d(\tau)\right) \cdot Eff_d - PR_d(\tau). \quad (8)$$

For epoch τ, $PD_d(\tau)$ can be negative if the renewable power available at the data center, $PR_d(\tau)$, is greater than the cooling and computing power.

3.3.8 Net Metering Model

Net metering allows data center operators to sell back the excess renewable power generated on-site to the utility company. When the excess power is added into the grid, utility companies pay a fraction of the retail price. This fraction is called the net metering factor, α.

3.3.9 Peak Demand Model

Most utility providers charge a flat-rate (peak demand) fee based on the highest (peak) power consumed at any instant

during a given billing period, e.g., month. The peak demand price per kW at data center d is denoted as P_d^{price}. We define $Pc_d^{peak}(\tau)$ as the highest grid power consumed since the beginning of the current month, including the current epoch τ. We define $Pp_d^{peak}(\tau)$ as the highest grid power consumption since the beginning of the current month until the start of the current epoch τ. The peak power increase at data center d during epoch τ, $\Delta_d^{peak}(\tau)$, is then defined as

$$\Delta_d^{peak}(\tau) = Pc_d^{peak}(\tau) - Pp_d^{peak}(\tau), \tag{9}$$

if $Pc_d^{peak}(\tau) \geq Pp_d^{peak}(\tau)$ else it is equal to 0. The peak power increase $\Delta_d^{peak}(\tau)$ is calculated in each epoch τ and summed over all epochs in a billing period to calculate total peak power.

3.3.10 System Electricity Cost

The electricity price per kWh at data center d during epoch τ is defined as $E_d^{price}(\tau)$. Data center operators can use net metering if total non-renewable power consumed throughout the data center, $PD_d(\tau)$, is negative. For such conditions, the total power/electricity cost $PC_d(\tau)$ for data center d during epoch τ can be defined as

$$PC_d(\tau) = E_d^{price}(\tau) \cdot PD_d(\tau) + P_d^{price} \cdot \Delta_d^{peak}(\tau), \tag{10a}$$

if $PD_d(\tau)$ is positive. If $PD_d(\tau)$ is negative then

$$PC_d(\tau) = E_d^{price}(\tau) \cdot \alpha \cdot PD_d(\tau) + P_d^{price} \cdot \Delta_d^{peak}(\tau), \tag{10b}$$

where $\alpha = 0$ if net metering is not available. The first term in Equation (10) represents the TOU electricity cost and the second term represents the peak demand cost.

3.3.11 Co-Location Interference Model

Tasks competing for shared memory in multicore processors can cause severe performance degradation, especially when competing tasks are memory-intensive [42]. The memory-intensity of a task refers to the ratio of last-level cache misses to the total number of instructions executed. We employ a linear regression model from [43] that combines a set of disparate features (i.e., inputs that are correlated with task execution time) based on the current tasks assigned to a multicore processor to predict the execution time of a target task i on core k in the presence of performance degradation due to interference from task co-location. These features are, the number of applications co-located on that multicore processor, the base execution time, the clock frequency, the average memory intensity of all applications on that multicore processor, and the memory intensity of application i on core k.

In our 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 to determine a co-located execution rate for task-type i on core k, $CER_{i,k}^{core}(\tau)$. The total execution rate for task-type i in epoch τ is therefore given by

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

TABLE 1
Node Processor Types Used in Experiments

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

Because co-location interference degrades the execution rate of task-types, some tasks may be unable to finish if task-types are allocated to cores based on Equation (2) that does not consider degradation effects. To allocate tasks to cores when considering co-location interference, some of our techniques use information about $CER_{i,k}^{core}$ to judge actual execution rates more accurately than techniques that do not consider co-location interference. When considering co-location at a data center d, the data center execution rate constraint becomes

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

The linear regression model was trained using execution time data that was collected by executing benchmarks from the PARSEC [44] and NAS parallel [45] benchmark suites on a set of server class multicore processors that define the nodes used in our study (see Table 1 in Section 6 for details about each node). This model for execution time prediction under co-location interference is derived from real workloads and machines, and results in a mean prediction error of approximately 7 percent.

4 Problem Formulation

We consider a scenario with multiple data centers sharing a single workload. The system is assumed to be under-subscribed in the sense that the system is expected to have enough computation resources to complete the workload without requiring that any tasks be dropped. Though the system is under-subscribed, individual data centers may be executing at full capacity. The tasks originate off-site from the data centers, and we make the simplifying assumptions that the transmission time and cost from a task origin to a data center is equivalent for all data centers. The objective of a GDRM is to minimize monetary electricity cost of the geo-distributed system (the sum of Equation (10) across all data centers) while ensuring that the workload is completed according to the constraints defined by Equations (1) and (12).

The problem is especially challenging when considering the variable amount of renewable power available at each data center, the heterogeneity of compute nodes within a data center, and the additional constraint that the entire workload must complete without dropping any tasks. Having information about TOU electricity pricing, peak demand pricing, a prediction of the amount of renewable power, net metering policy at each data center, the incoming workload, and the execution speeds of task-types on the heterogeneous compute nodes allows our GDRM to make intelligent decisions for allocating the workload, as described next in Section 5.

5 Heuristics Descriptions

5.1 Overview

The GDRM allocates the incoming workload not only to individual data centers, but also to specific nodes within

each data center. The GLD problem is NP-hard [20], and therefore we propose three resource management heuristics for GDRM, with each having different levels of detail of the system model available to it.

5.2 Force Directed Load Distribution Heuristics

Force-directed load distribution (FDLD) is a variation of force-directed scheduling [46], a technique often used for optimizing semiconductor logic synthesis. FDLD is an iterative heuristic that selectively performs operations to minimize system forces until all constraints are met. We adapt the FDLD approach proposed in [20] to the rate-based allocation environment we have outlined in Section 3, and enhance it to propose two new FDLD based heuristics to solve our problem.

Our baseline FDLD heuristic is the one proposed in [20], which we enhance with simple over-provisioning (FDLD-SO) to compensate for performance degradation due to co-location. This allows the FDLD heuristic to meet the execution rate constraint at a given data center. This heuristic over-provisions all task-types equally by scaling estimated task execution rates by a factor ϕ^C. Our first new heuristic improves upon FDLD-SO by using task aware over-provisioning (FDLD-TAO) 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 through simulation studies to provide values that give the system the best possible performance. Lastly, our second new heuristic uses the co-location (FDLD-CL) models given in Section 3.3.11 to account for co-location effects when calculating task execution rates.

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, resulting in the lowest total system force F^S (steps 4-23). After getting the final allocation solution from the heuristic, we calculate the final execution rates and the system electricity cost by summing the power costs across all data centers. The rest of this section presents the derivation of the total system force F^S, Equation (15) (used in steps 10 and 12 in Algorithm 1), and Equation (16) (used in step 15 in Algorithm 1).

As per Equation (2), the task execution rate is a function of the P-state of the node the task is executing at, but FDLD is not designed to make DVFS decisions to set the execution rates of task-types, and therefore we assume that it is going to make the decisions based on the node utilization. An average execution rate must be determined for all task-types using the average node utilization factor μ_j 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

$$R_{j,i} = ER_{j,i}(P_{MAX}) + \left[ER_{j,i}(P_0) - ER_{j,i}(P_{MAX})\right]\mu_j. \quad (13)$$

The single core execution rate of the application is equal to the minimum execution rate executing at the highest numbered P-state (first term in Equation (13)) plus a performance factor (second term in Equation (13)). The node utilization will affect the single core execution rate such that the execution rate will be proportional to the node utilization, e.g., $R_{j,i}$ will be $ER_{j,i}(P_0)$ for $\mu_j = 1$ and $ER_{j,i}(P_{MAX})$ for $\mu_j = 0$.

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. operations-remaining = true
3. **while** operations-remaining
4. **for** each node with tasks still allocated to it
5. **for** each task-type on the node
6. temporarily remove task-type from node
7. **if** FDLD-CL
8. estimate execution rates using Equation (11) (CER_i)
9. **else if** FDLD-TAO
10. estimate execution rates using Equation (15) and ϕ_i^C
11. **else if** FDLD-SO
12. calculate execution rates using Equation (15) and ϕ^C
13. calculate estimated power costs $PC_d^E(\tau)$
14. calculate F^S from F^{ER} and F^C
15. **if** execution rate constraints are not violated (Equation (12) for FDLD-CL, Equation (16) for FDLD-SO and FDLD-TAO)
16. add to set of possible task removal operations
17. restore task-type to node
18. **end for**
19. **end for**
20. **if** set of possible task removal operations is empty
21. operations-remaining = false
22. **else**
23. choose and implement the task-type removal operation that would result in the lowest F^S
24. **end while**
25. return final allocation solution

Let $NN_{d,j}$ be the number of nodes of type j in data center d. Let $S_{d,j,n}(\tau)$ be the set of instances of different task-types placed on a node n of node-type j in data center d during epoch τ. An instance of a task-type is a task that belongs to the specific task-type. 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. We assume that the compute time of each core on a node is evenly divided among its assigned tasks. Therefore the equivalent number of nodes $Q_{d,j,i}(\tau)$ represent the compute time allocated to task-type i on node-type j for data center d, given by

$$Q_{d,j,i}(\tau) = \sum_{n=1}^{NN_{d,j}} \begin{cases} \frac{1}{\left|S_{d,j,n(\tau)}\right|} & \text{if } i \in S_{d,j,n}(\tau) \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

For example, for a data center with twelve nodes and three task-types assigned, each task-type would get the equivalent of four nodes out of twelve, i.e., $\frac{1}{3}$ of the available compute resources. This will be assigned such that each task-type gets $\frac{1}{3}$ of the execution time of each of the twelve nodes.

To compensate for performance degradation due to co-location effects, node over-provisioning is accomplished by the factor ϕ. 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 node-type j during epoch τ, when using either the FDLD-SO or FDLD-TAO versions, is given by

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

subject to the constraint

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

Where ϕ is replaced by either ϕ^C or ϕ^C_i in Equation (15) when using either FDLD-SO or FDLD-TAO, respectively.

Let Z be the term that is replaced by $CER_i(\tau)$ when considering the FDLD-CL heuristic and is replaced by $ER^{E}_{j,i}(\tau)$ when using either FDLD-SO or FDLD-TAO. The execution rate force F^{ER} is calculated using

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

Observe that $F^{ER}(\tau)$ will decrease to zero as the ratio of Z to $AR_i(\tau)$ decreases to 1.

Let $P^{E}_{d,j,n}$ be the estimated average power for node n of node-type j in data center d, calculated as

$$P^{E}_{d,j,n} = \begin{cases} P^{Sleep}_j & \text{if } \left|S_{d,j,n}(\tau)\right| = 0 \\ \mu_j P^{D}_j + P^{S}_j & \text{otherwise.} \end{cases} \tag{18}$$

For all FDLD variants, let $PD^{E}_{d}(\tau)$ be the estimated non-renewable power consumed at data center d during epoch τ, calculated as

$$PD^{E}_{d}(\tau) = \left[\sum_{j \in J_d} \sum_{n=1}^{NN_{d,j}} P^{E}_{d,j,n} \cdot \left(1 + \frac{1}{CoP_d} \right) + ADP_d(\tau) \right] \cdot Eff_d - PR_d(\tau), \tag{19}$$

For all FDLD variants, let $PC^{E}_{d}(\tau)$ be the estimated power cost at data center d during epoch τ. $PC^{E}_{d}(\tau)$ is calculated as shown in Equation (10), where $PD_d(\tau)$ is replaced by its estimated version, i.e., $PD^{E}_{d}(\tau)$.

Let $PC^{max}_{d}(\tau)$ be the maximum real power cost possible at data center d, calculated using

$$PC^{max}_{d}(\tau) = E^{price}_{d}(\tau) \cdot \sum_{j \in J_d} NN_{d,j} \cdot \left(\mu_j P^{D}_j + P^{S}_j \right) + P^{price}_{d} \cdot \Delta^{peak}_{d}(\tau). \tag{20}$$

The cost force F^C can then be calculated with

$$F^{C}(\tau) = \sum_{d=1}^{D} \left[e^{\left(\frac{PC^{E}_{d}(\tau)}{PC^{max}_{d}(\tau)} \right)} - 1 \right]. \tag{21}$$

Observe that the value of F^C goes to zero as the ratio of $PC^{E}_{d}(\tau)$ to $PC^{max}_{d}(\tau)$ 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

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

5.3 Genetic Algorithm Heuristic

We designed a third heuristic: a genetic algorithm load distribution with co-location awareness (GALD-CL). The Genitor style [47] GALD-CL heuristic has two parts: a genetic algorithm based GDRM (Algorithm 2) and a local data center level greedy heuristic (Algorithm 3) that is used to calculate the fitness value of the genetic algorithm.

The initial population for GALD-CL (Algorithm 2) is generated by randomly partitioning the global arrival rate AR_i for each task-type i in the epoch across all of the data centers (step 1). Each data center d gets a desired fraction of arrival rate $DAR_{d,i}$ for each task-type i such that

$$\sum_{d=1}^{D} DAR_{d,i}(\tau) = AR_i(\tau), \quad \forall i \in I. \tag{23}$$

Each chromosome is a matrix of I x D genes, where each gene is a $DAR_{d,i}$ and d pair, representing the desired arrival rate of each task-type i at each data center d. We perform selection, crossover, and mutation that alter existing chromosomes to generate new offspring chromosomes (step 3). We use a roulette wheel selector, a two-point crossover, and a simple real range mutator [48]. After the generation of two offspring, for each new chromosome, a local greedy heuristic discussed in the next paragraph is used to perform allocations at the data center level (steps 4-6). After the greedy heuristic, the fitness value (total energy cost including CRAC energy) for each chromosome is calculated (step 7). The population is trimmed to its original size by eliminating the least-fit (highest cost) chromosomes (step 8). After reaching the time limit, the algorithm ends and the final allocation of arrival rates is obtained from the best (lowest cost) chromosome.

Algorithm 2. Pseudo-Code for GALD-CL Heuristic

1. create an initial population of chromosomes
2. **while** within time limit
3. perform selection, crossover and mutation to create new chromosomes
4. **for** each new chromosome
5. use greedy heuristic to perform allocations at data center level
6. **end for**
7. find the fitness values (total energy costs) of the population
8. trim population to its original size by eliminating the least-fit chromosomes
9. **end while**
10. return best chromosome from population and use final allocation from it

Local Greedy Heuristic. This greedy approach (Algorithm 3) is similar in concept to Min-min in [49], [50] and is used to assign the desired arrival rate $DAR_{d,i}$ of each task-type i to cores in data center d. For this heuristic, cores are dedicated to a single particular task-type, i.e., a core cannot execute more than one task-type at a time. Our greedy heuristic iteratively assigns task-types to cores to find the most efficient mapping, where we define efficiency $EFF_{i,k}$ for a task mapping of type i on a node of type NT_k in P-state $PS_{i,k}(\tau)$ as

$$EFF_{i,k}(\tau) = \frac{ECS(i, NT_k, PS_{i,k}(\tau))}{APC(i, NT_k, PS_{i,k}(\tau))}. \tag{24}$$

For each data center, we start this greedy heuristic by finding the task-type/node-type pair with the highest $EFF_{i,k}(\tau)$ value (Equation (24)) among all possible pairs (step 2). Our heuristic uses the most efficient P-state for a given task-type/node-type pair. All I x J_d task-type to node-type pairs

are then sorted by their efficiency in descending order to create a list (step 3). At each iteration of the heuristic, the first (most efficient) pair is selected (step 5). Then the heuristic checks if any unassigned cores within the selected node-type exist (step 6). If so, we assign fraction of the $DAR_{d,i}$ for the selected task-type to a core in the selected node-type (based on $ECS(i,n,p)$) (step 7) and that core is removed from consideration (step 8). The CRAC outlet temperatures are set to the red-line temperature, and then the outlet temperatures of all CRAC units are iteratively decreased by one degree until the thermal constraint (the inlet temperature of the node, with the selected core, should be less than the red-line temperature) is met (step 9). After the assignment, the execution rate $ER_{d,i}^{DC}$ is updated (step 10). If the $ER_{d,i}^{DC}$ for the selected task-type exceeds its $DAR_{d,i}$ (step 11), all task-type/node-type pairs with that task-type are removed from the list (step 12). The heuristic uses Equation (11) to estimate core execution rates with constraint defined by Equation (12). If no unassigned cores within the selected node-type exist (step 13), that task-type/node-type pair is removed from consideration (list) (step 14). This iterative part (steps 4-15) of the heuristic stops when there are no more task-type/node-type pairs to consider.

Algorithm 3. Pseudo-Code for Local Greedy Heuristic

1. **for** each data center
2. find most efficient task-type/node-type pair among all possible pairs
3. sort all task-type/node-type pairs by efficiency in descending order
4. **while** sorted list is not empty
5. select first task-type/node-type pair
6. **if** unassigned core within selected node-type exist
7. assign fraction of $DAR_{d,i}$ for the selected task-type to a core from the selected node-type
8. remove that core from future consideration (assigned)
9. set CRAC outlet temperatures to highest temperatures such that node inlet temperature is less than red-line temperature
10. update $ER_{d,i}^{DC}$
11. **if** $ER_{d,i}^{DC}$ for the selected task-type exceeds its $DAR_{d,i}$
12. remove all task-type/node-type pairs with that task-type
13. **else** //no more cores from node-type exist
14. remove that task-type/node-type pair from the list
15. **end while**
16. **end for**
17. **for** each data center
18. **for** each task-type
19. **if** $DAR_{d,i}$ exceeds the data center assignment $ER_{d,i}^{DC}$ //solution invalid
20. adjust $DAR_{d,i}$ value by reducing it by 10 percent for data center where solution is invalid
21. normalize $DAR_{d,i}$ values to modify chromosome until valid solution reached, return to step 1
22. **end for**
23. **end for**

After mapping the desired arrival rates $DAR_{d,i}$ of each task-type i to cores in each data center d, if the $DAR_{d,i}$ exceeds the data center assignment $ER_{d,i}^{DC}$ (checked for each task-type in each data center), the solution is invalid (step 19). If so, we adjust $DAR_{d,i}$ value by reducing it by 10 percent for the data center at which the solution is invalid (step 20). To calculate normalized arrival rate value $DAR_{d,i}^{norm}$, we then normalize the $DAR_{d,i}$ values such that the sum of $DAR_{d,i}^{norm}$ values for each task-type matches the global arrival rate AR_i

$$DAR_{d,i}^{norm}(\tau) = DAR_{d,i}(\tau) \cdot \frac{AR_i(\tau)}{\sum_{d=1}^{D} DAR_{d,i}(\tau)}. \quad (25)$$

We then replace $DAR_{d,i}$ with $DAR_{d,i}^{norm}$ for all data centers d with the given task-type i to modify the chromosome until a valid solution can be reached (step 21), i.e., $ER_{d,i}^{DC}$ exceeds (or equals) $DAR_{d,i}$.

In summary, the GALD-CL heuristic addresses two potential shortcomings of the FDLD variants. First, the nature of the decision making within the FDLD variants prevents them from making any kind of DVFS decisions, therefore a single P-state is chosen for each node-type regardless of the tasks executing on the node. The greedy heuristic within the GALD-CL approach chooses the most efficient P-state for each task-type on each node-type [34]. 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.

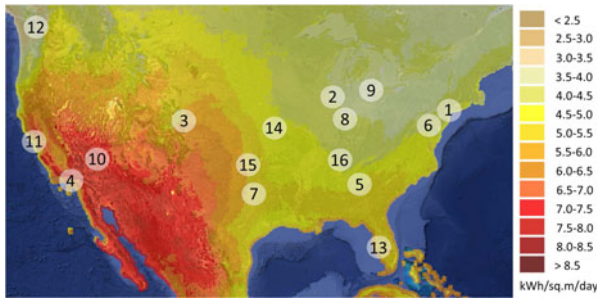


Fig. 3. Location of simulated data centers overlaid on solar irradiance intensity map (average annual direct normal irradiance); wind data collected but not shown [41].

6 Simulation Environment

Experiments were conducted for three geo-distributed data center configurations containing four, eight, and sixteen data centers. Locations of the data centers in the three configurations were selected from major cities around the continental United States to provide a variety of wind and solar conditions among sites and at different times of the day (see Fig. 3).

Experiments for the configuration with four data centers used locations one through four from Fig. 3, while experiments using configuration with eight and sixteen data centers used locations one through eight and one through sixteen, respectively. The sites of each configuration were selected so that each configuration would have a fairly even east coast to west coast distribution to better exploit TOU pricing, peak demand pricing, net metering, 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 1, with most locations having three node-types and per-node core counts that range from 4-12 cores. For CRAC units, the red-line temperature was set to 30° C, which is on the high end of ASHRAE's temperature guidelines [38].

TABLE 2
Peak Demand Prices Used in Experiments

data center location	peak demand price ($/kW)	data center location	peak demand price ($/kW)
New York	11.04	Detroit	14.54
Chicago	3.82	Las Vegas	8.25
Denver	6.75	San Francisco	13.01
LA	8.91	Seattle	3.29
Atlanta	8.11	Tampa	10.25
Baltimore	3.84	Kansas City	6.39
Dallas	11.88	Oklahoma City	6.20
Indianapolis	10.57	Nashville	5.09

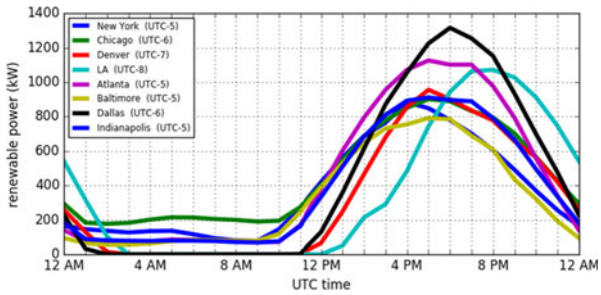


Fig. 4. Renewable power available at eight locations.

Nodes placed in a sleep state by a heuristic are considered to be in the Advanced Configuration and Power Interface (ACPI) node sleep state S3, where RAM remains powered on. Sleep state S3, also commonly referred to as *suspend* or *standby*, allows greatly reduced power consumption while still possessing a small latency to return to an active operating state. Sleep power for all nodes is calculated as a fixed percentage of static power for each node-type, assumed to be 16 percent based on a study of node power states [51]. The average node utilization factor used during FDLD allocations, μ_j, is set as 0.75. The Coefficient of Performance of the CRAC unit was determined empirically by simulating workloads with different memory intensity classes at each data center location, and its value ranges between 1.43 and 2.08 for 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 conservatively assumed to be five minutes.

The electricity prices used during experiments, as shown in Fig. 1, were taken directly from Pacific Gas and Electric (PG&E) Schedule E-19, which is for commercial locations consuming between 500 kW and 1 MW [9]. The peak demand prices per kW are given in Table 2.

We assume that each data center has peak renewable power generating capacity equivalent to its maximum power consumption [10], [11], [52]. Renewable power at each location was either wind power, solar power, or a combination of the two. For example, a location with high average solar irradiance but low average wind speed would be restricted to having solar power only. Solar and wind data was obtained from the National Solar Radiation Database [41]. An example of the renewable power available at different locations is given in Fig. 4.

In net metering, data centers send excess power back to the grid and utility companies pay back the customer a fraction of retail price α. In most cases, the net metering factor α is 1; in very few cases, it is less than 1; and in some cases, it is 0, i.e., net metering is not available at that location [15], [53].

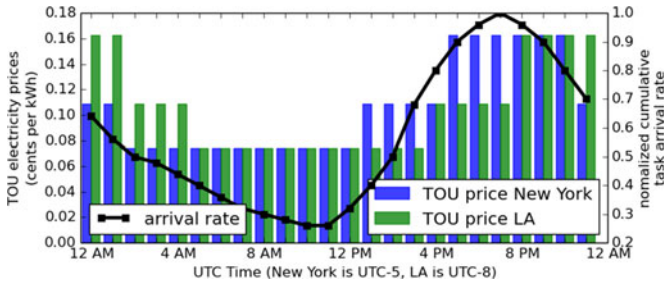


Fig. 5. Baseline task arrival rate and TOU prices at two sites (New York, Los Angeles) over 24 hours.

Each task-type used in our experiment is representative of a different benchmark from the PARSEC [44] and NAS parallel [45] benchmark suites. Task execution times and co-located performance data for the task of the different memory intensity classes were obtained from running the benchmark applications on the nodes listed in Table 1 [43]. Synthetic task arrival patterns were constructed and the baseline pattern is shown in Fig. 5. For reference, the figure also shows TOU prices for an east coast and a west coast site.

7 Experiments

7.1 Cost Comparison of Heuristics

Our first set of experiments analyzes the total system energy cost for each heuristic described in Section 5 with and without peak shaving and net metering. For each heuristic, we evaluate four variants: (1) without both peak shaving and net metering, (2) with net metering only, (3) with peak shaving only, and (4) with both peak shaving and net metering. Heuristic variants that are referred to as "without peak shaving" do not include the peak demand pricing factor in their objective functions but consider it while calculating the total monthly electricity cost at the end of the billing period. These experiments use a data center configuration consisting of eight locations and a workload that was a hybrid mix of memory-intensive and CPU-intensive task-types (discussed in Section 7.2). The system energy costs are estimated over a duration of one day. The results are shown in Fig. 6a.

For each individual heuristic, considering both net metering and peak shaving produced the best results, while experiments without these produced worse results. This validates our consideration of peak shaving and net metering during geo-distributed data center workload management, to more effectively minimize energy costs. It can also be observed that the FDLD-CL heuristic, using the co-location models, performs the best among the FDLD variants. The FDLD-SO heuristic performed the worst, severely over-provisioning nodes and resulting in high operating costs. The GALD-CL heuristic with net metering and peak shaving outperformed all other approaches. This heuristic has complete information about the entire system model, including the co-location models and task-node power (DVFS) models, allowing it to make better placement decisions. With additional execution time and a larger population of chromosomes even better solutions can be found.

The decision option related to DVFS P-states gives the GALD-CL heuristic a strong advantage over the FDLD variants but it also takes longer to execute. The GALD-CL heuristic was limited to a run time of approximately one hour per epoch simulated. Alternatively, the FDLD heuristics completed in approximately six minutes per epoch

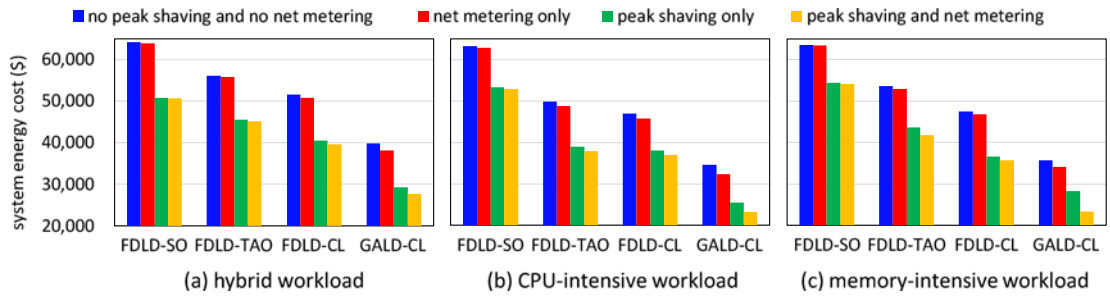


Fig. 6. System energy costs for each heuristic over a day for (a) hybrid, (b) CPU-intensive, and (c) memory-intensive workloads, for eight data center locations.

simulated. While not performing as well as the GALD-CL, the FDLD variants have the advantage of reaching a solution more quickly, which may be beneficial in some cases.

7.2 Workload Type Analysis

The previous experiment used a workload that was a hybrid mix of memory-intensive and CPU-intensive task-types. Fig. 6 shows experiments for all of the FDLD and GALD-CL heuristics for a group of eight data centers where two additional workload types were evaluated: 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) [44], [45]. The composition of data center workloads can vary greatly and can impact the resource requirements, and these experiments show that the techniques presented in our work will perform well for a variety of workload types. Observe that the CPU-intensive workload typically costs the least as it experiences the least co-location degradation and therefore requires fewer nodes. However, the memory-intensive tasks cause performance degradation because they compete for shared memory in multicore processors.

7.3 Scalability Analysis

7.3.1 Data Center Scalability Analysis

In this experiment, we analyze heuristic performance for additional problem sizes. Simulations running hybrid workloads were conducted for four and sixteen data center configurations in addition to the previously discussed eight data center configuration. For each configuration, the average performance improvement of each heuristic over the FDLD-SO heuristic with no peak shaving and no net metering is given in Table 3. The GALD-CL heuristic was limited to a run time of approximately one hour. FDLD heuristics for four, eight, and sixteen locations completed on average in two, six, and eighteen minutes per epoch simulated, respectively. These experiments confirm that all FDLD heuristics can perform well for smaller and larger problem sizes but the GALD-CL heuristic consistently performs the best for all problem sizes.

TABLE 3
Energy Cost Reduction Comparison

	heuristic	no PS and no NM	NM only	PS only	PS and NM
4 data centers	FDLD-SO	0.0%	0.2%	23.4%	23.6%
	FDLD-TAO	12.3%	14.5%	33.4%	34.0%
	FDLD-CL	15.0%	15.7%	36.7%	37.9%
	GALD-CL	49.5%	58.2%	67.3%	75.8%
8 data centers	FDLD-SO	0.0%	0.3%	20.8%	21.1%
	FDLD-TAO	16.9%	16.8%	31.5%	32.5%
	FDLD-CL	19.6%	20.8%	36.9%	38.3%
	GALD-CL	40.9%	45.7%	54.5%	60.8%
16 data centers	FDLD-SO	0.0%	0.4%	21.7%	24.5%
	FDLD-TAO	11.1%	12.2%	30.9%	32.9%
	FDLD-CL	14.2%	16.1%	33.4%	36.6%
	GALD-CL	33.1%	36.4%	44.0%	46.6%

PS = peak shaving, NM = net metering.

7.3.2 GA Run Time Scalability Analysis

Table 3 shows similar energy cost reduction results for all FDLD variants in the cases of the data center configurations containing four, eight, and sixteen data centers running hybrid workloads. But for GALD-CL, we notice that the energy cost reduction decreases with the increasing number of data centers. Here, as the number of data centers in the group grows larger, the problem size increases and the number of GALD-CL generations that can take place within the time limit (one hour by default) decreases, which decreases the performance of GALD-CL.

To better understand how the GALD-CL solution quality is impacted by the heuristic's run time, we increase the GALD-CL run time in proportion to the increase in number of data centers. We execute GALD-CL for about one hour for four data centers, about two hours for eight data centers, and about four hours for sixteen data centers. The results from Table 4 show that GALD-CL is capable of performing well for larger problem size, when given more time. For comparison, the Table 4 also includes results for GALD-CL executed for about one hour for a group of four, eight, and sixteen data centers. It should be noted that, even when allowing the GALD-CL to execute for one hour, it still provides the system a significant energy cost reduction in comparison to the all FDLD heuristics as shown in Table 3.

TABLE 4
Impact of GALD-CL Run Time

	GALD-CL epoch	no PS and no NM	NM only	PS only	PS and NM
4 data centers	1 hour	49.5%	58.2%	67.3%	75.8%
8 data centers	1 hour	40.9%	45.7%	54.5%	60.8%
	2 hours	46.7%	55.2%	62.5%	72.2%
16 data centers	1 hour	33.1%	36.4%	44.0%	46.6%
	4 hours	40.1%	49.4%	61.6%	69.9%

PS = peak shaving, NM = net metering.

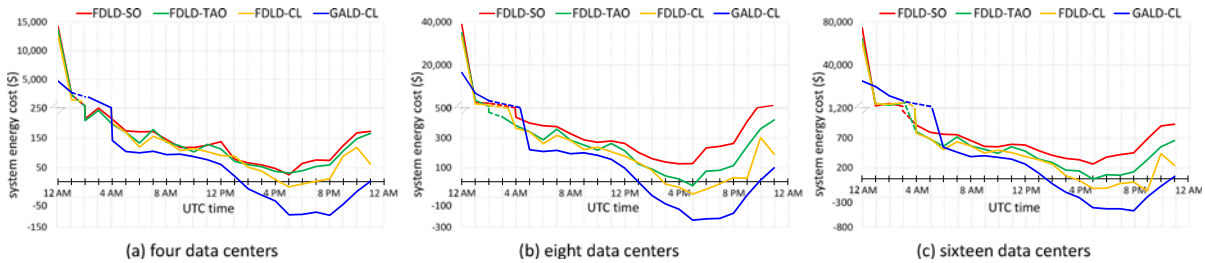


Fig. 7. System energy costs for each heuristic over a day for epoch-based analysis, for a configuration with (a) four, (b) eight, and (c) 16 data center locations running the hybrid workloads.

7.3.3 Epoch-Based Analysis

For most of our experiments, we analyzed the total system cost for each heuristic over one day. Fig. 7 shows a more detailed view of the system operating cost at one-hour intervals over the course of a day for four, eight, and sixteen data centers executing a hybrid workload. The four resource management heuristics in this study consider both peak shaving and net metering.

Net metering causes the plots to go into the negative region in certain epochs, which represents the case when the system earns money by selling excess renewable power back to the utility companies. The operating cost for each heuristic is very high during the first epoch because the period for which the results are shown represents the first day of the month where the initial peak demand cost is added. This effect would not be present for other days of the month. After a few epochs, the performance of the FDLD-CL came close to the GALD-CL, but was not able to surpass its performance.

7.4 Sensitivity Analysis

7.4.1 Net Metering Factor

Renewable power generation changes throughout the year. The amount of renewable power generated at a data center also depends on its location. As discussed in Section 6, data centers generate different amounts of renewable power at each location. Different states have different net metering laws and utility companies from those states have different energy buy-back rates for net metering. In this section, we analyze the impact of net metering (with no peak shaving) and study how the net metering factor impacts energy costs and the behavior of the heuristics. For these experiments, we consider a configuration with eight data centers executing a hybrid workload. The experiments analyze the total system cost for each heuristic utilizing only net metering. Peak shaving is ignored because this study focuses only on the sensitivity analysis of the net metering factor.

Net metering factor values for data centers were randomly sampled (with uniform distribution) from three value ranges, where buy-back costs for renewable energy are low ($0 \leq \alpha 1 < 0.4$), medium ($0.4 \leq \alpha 2 < 0.7$), and high ($0.7 \leq \alpha 3 \leq 1$). Furthermore, we consider five simulation runs for each set of values and plot standard deviations as shown in Fig. 8. In the figure, the system energy costs decrease, but not dramatically as the net metering factor values increase. However, we can observe that GALD-CL is able to exploit net metering better than the other heuristics as α values increase.

7.4.2 Peak Demand Price

As discussed in Section 6, the set of data centers we are considering for these experiments are heterogeneous and therefore consume different amounts of peak power. Utilities at different locations have different peak prices as shown in Table 2. In this section, we analyze the impact of peak shaving (with no net metering) and study the impact of peak demand price. For these experiments, we consider eight data centers executing a hybrid workload. The experiments analyze the total system cost for each heuristic utilizing only peak shaving. Net metering is not considered because this study focuses only on the sensitivity analysis of the peak demand price.

Peak demand pricing values for data centers were randomly sampled (with uniform distribution) from three value ranges, where these values are low ($3 \leq p1 < 7$), medium ($7 \leq p2 < 11$), and high ($11 \leq p3 \leq 15$). Furthermore, we consider five simulation runs for each set of values and plot standard deviations as shown in Fig. 9. As expected, the system energy costs increase as the peak demand pricing values increase. We observe a significant cost change across $p1$, $p2$, and $p3$ because peak demand cost is one of the two major components of the total system cost (see Equation (10)) and peak demand pricing has a major impact on the peak demand cost component.

The wide standard deviation for peak shaving shows that the system is highly sensitive to the peak demand price. By

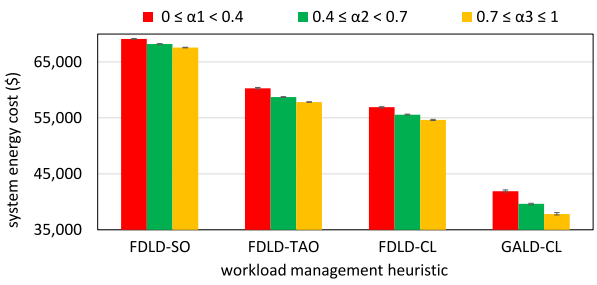


Fig. 8. System energy costs for each heuristic over a day for net metering factor sensitivity analysis, for eight data center locations running a hybrid workload.

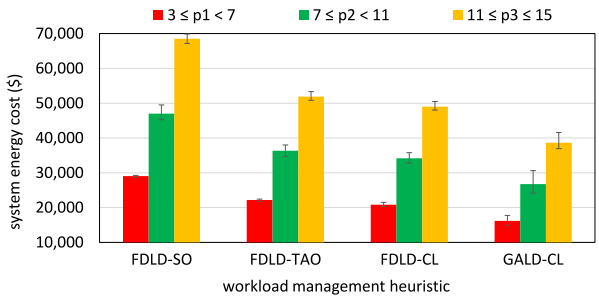


Fig. 9. System energy costs for each heuristic over a day for peak demand price sensitivity analysis, for eight data center locations running a hybrid workload.

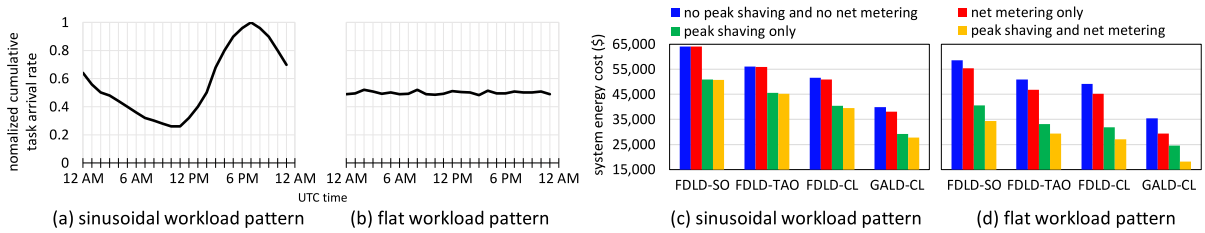


Fig. 10. Comparison of (a) sinusoidal and (b) flat workload arrival rate patterns. Comparison of system energy costs among heuristics over a day for (c) sinusoidal and (d) flat workload arrival rate patterns for eight data center locations running a hybrid workload.

comparing both net metering factor and peak demand price sensitivity analyses, we can observe that peak shaving has a bigger impact on system cost than net metering. The major reason behind this is realistic assumptions of renewable power generation capabilities at each data center. If we consider self-sustainable green data centers [10], [11], [52] and the trend of increasing on-site renewable (solar/wind) power farms, more renewable power will be available for net metering which will have a more significant impact on the system.

7.5 Task Arrival Rate Pattern Analysis

All of our experiments so far have assumed a sinusoidal task arrival rate pattern as shown in Fig. 5. This kind of pattern exists in environments where workload traffic depends on user/consumer interaction and follows their demand during the day, e.g., Netflix [54], Facebook [55]. However, for the environments where continuous computation is needed and the workload pattern is non user/consumer interaction specific, the task arrival rate pattern is usually flat (nearly-constant). Examples of such environments exist in military computing installations (Department of Defense), government research labs (National Center for Atmospheric Research), etc. We conducted a set of simulations to analyze the impact of varying the task arrival pattern. We consider a configuration with eight data centers executing a hybrid workload with both sinusoidal and flat arrival patterns.

Recall that each task-type is characterized by its arrival rate and the estimated time required to complete the task on each of the heterogeneous compute nodes in all P-states. The four GDRM heuristics map execution rates to minimize total energy cost across all data centers with the constraint that the execution rates of all task-types meet their arrival rates (see Equation (1)). Therefore the assignment of execution rates alters with the change in arrival rates, which further affects the system cost. The results shown in Figs. 10c and 10d indicate that the geo-distributed system responds differently for sinusoidal and flat arrival rate patterns.

Overall system energy cost is higher for the sinusoidal workload arrival pattern because it produces higher peak data center power, which further increases the peak demand cost, than the flat workload arrival pattern. Peak shaving significantly reduces the energy costs for both workload patterns. The sinusoidal workload arrival pattern roughly aligns with the pattern of the renewable power generation (see Fig. 4), allowing data centers to utilize all of the available renewable power.

The flat (nearly-constant) workload arrival rate pattern does not align with the pattern of the renewable power generation. This leaves data centers with excess renewable energy in the second half of the day, which allows heuristics to exploit net metering heavily and produce more cost savings. Thus, heuristics that consider net metering perform better for the flat arrival rate pattern as compared to the sinusoidal arrival rate pattern.

8 Conclusions

We studied the problem of minimizing the energy costs for geographically distributed heterogeneous data centers with the constraint that all tasks complete without being dropped. Renewable energy, peak demand, and co-location interference at data centers have a significant impact on energy consumption. We capture these effects by including net metering, peak shaving, and co-location models in our workload distribution techniques. We analyzed several techniques that possess varying degrees of co-location interference prediction information: (1) a force-directed scheduling technique, FDLD-TAO, that uses task aware over-provisioning to estimate co-location effects for each task-type, (2) a force-directed scheduling technique, FDLD-CL, that uses co-location models when calculating task execution rates, and (3) a genetic algorithm combined with a local search technique, GALD-CL, that has information about the co-location models and DVFS P-states.

The primary contributions of our research are to provide analyses of the aspects of the problems and solutions associated with renewable energy, peak demand, and co-location aware resource management in heterogeneous computing systems. We used new peak demand and net metering models that directly consider real-world peak demand prices and net metering policies in calculation of the system electricity cost, and a new co-location interference model created from a linear regression technique using data from real servers. We demonstrated that including this additional information in the decision process of the resource management heuristics resulted in a lower energy cost. This is achieved by reducing or eliminating node over-provisioning while still meeting all required workload execution rates. Ignoring interference effects altogether can be especially detrimental to overall performance and energy overheads when task execution times deviate far from those expected due to interference. We also demonstrated the importance of net metering and peak shaving by illustrating the impact of the awareness of these factors on reducing operating costs for geographically distributed data centers.

We compared our techniques across various workload profiles, performing a scalability assessment, examining sensitivity to renewable power availability and peak demand pricing parameters, and testing system behavior for different task arrival patterns. Our proposed FDLD-CL and GALD-CL heuristics resulted in 37 and 61 percent lower operational costs on average than an approach from prior work (represented by the FDLD-SO heuristic) [20]. However, to

implement our approach in a real system, it needs to be used with some form of a workload prediction technique, e.g., [56], [57]. Additionally, due to scalability issues of GALD-CL, we recommend using FDLD-CL with a workload prediction technique in systems where the workload profile changes rapidly and therefore requires short epochs, e.g., a few minutes. When the workload profile is not changing rapidly and the workload distribution decisions are provided more time, e.g., an hour, using GALD-CL with a workload prediction technique is a more suitable heuristic.

Acknowledgments

The authors thank Dr. Greg Pfister and Daniel Dauwe for their valuable comments on this work. This work is supported by the US National Science Foundation (NSF) under grant CCF-1302693. A preliminary version of portions of this work appeared in [21]. They expanded and significantly improved upon their prior work by considering peak shaving, net metering, and co-location interference models in our load distribution techniques. They also performed several new experiments and provided more in-depth analyses of the proposed resource management heuristics. The result is notable improvements in reducing operational costs for geo-distributed data centers.

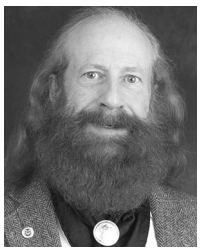
References

[1] Data center locations. [Online]. Available: http://www.google.com/about/datacenters/inside/locations/index.html, Accessed on: Aug. 1, 2017.
[2] Global infrastructure. [Online]. Available: http://aws.amazon.com/about-aws/global-infrastructure/, Accessed on: Aug. 1, 2017.
[3] Y. Li, H. Wang, J. Dong, J. Li, and S. Cheng, "Operating cost reduction for distributed internet data centers," in *Proc. 13th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, May 2013, pp. 589–596.
[4] What is time-of-use pricing and why is it important? [Online]. Available: http://www.energy-exchange.net/time-of-use-pricing/, Accessed on: Aug. 1, 2017.
[5] Dynamic pricing. [Online]. Available: http://whatis.techtarget.com/definition/dynamic-pricing. Accessed on: Aug. 1, 2017.
[6] Demand charges. [Online]. Available: http://www.stem.com/resources/learning/, Accessed on: Aug. 1, 2017.
[7] Understanding peak demand charges. [Online]. Available: https://energysmart.enernoc.com/understanding-peak-demand-charges, Accessed on: Aug. 1, 2017.
[8] C. Hsu, "Rack PDU for green data centers," in *Data Center Handbook*, H. Geng, Ed. Hoboken, NJ, USA: Wiley, Nov. 2014, ch. 29.
[9] Pacific Gas and Electric Company, "Electric schedule e-19." [Online]. Available: http://www.pge.com/tariffs/tm2/pdf/ELEC_SCHEDS_E-19.pdf, Accessed on: Apr. 15, 2015.
[10] Apple to build a 3rd massive solar panel farm in North Carolina. [Online]. Available: https://gigaom.com/2014/07/08/apple-to-build-a-3rd-massive-solar-panel-farm-in-north-carolina/, Accessed on: Aug. 1, 2017.
[11] Solar energy project at McGraw-Hill site recently completed. [Online]. Available: http://www.nj.com/mercer/index.ssf/2012/01/solar_energy_project_at_mcgraw.html/, Accessed on: Aug. 1, 2017.
[12] Green power: Accelerating the transition to a clean energy future. [Online]. Available: https://www.microsoft.com/about/csr/environment/renewable_energy/, Accessed on: Aug. 1, 2017.
[13] Achieving our 100% renewable energy purchasing goal and going beyond. [Online]. Available: https://static.googleusercontent.com/media/www.google.com/en//green/pdf/achieving-100-renewable-energy-purchasing-goal.pdf, Accessed on: Aug. 1, 2017.
[14] Facebook's Altoona, Iowa data center to be completely wind-powered. [Online]. Available: https://www.slashgear.com/facebooks-altoona-iowa-data-center-to-be-completely-wind-powered-13305335/, Accessed on: Aug. 1, 2017.
[15] Net metering. [Online]. Available: http://freeingthegrid.org/, Accessed on: Aug. 1, 2017.
[16] I. Goiri, R. Beauchea, K. Le, T. D. Nguyen, M. E. Haque, J. Guitart, J. Torres, and R. Bianchini, "GreenSlot: Scheduling energy consumption in green datacenters," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, Nov. 2011, pp. 1–11.
[17] C. Ren, D. Wang, B. Urgaonkar, and A. Sivasubramaniam, "Carbon-aware energy capacity planning for datacenters," in *Proc. IEEE 20th Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst.*, Aug. 2012, pp. 391–400.
[18] I. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "GreenHadoop: Leveraging green energy in data-processing frameworks," in *Proc. 7th ACM Eur. Conf. Comput. Syst.*, Apr. 2012, pp. 57–70.
[19] I. Goiri, W. Katsak, K. Le, T. D. Nguyen, and R. Bianchini, "Parasol and GreenSwitch: Managing datacenters powered by renewable energy," in *Proc. 18th Int. Conf. Archit. Support Program. Languages Operating Syst.*, Mar. 2013, pp. 51–64.
[20] H. Goudarzi and M. Pedram, "Geographical load balancing for online service applications in distributed datacenters," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, Jun. 2013, pp. 351–358.
[21] E. Jonardi, M. A. Oxley, S. Pasricha, A. A. Maciejewski, and H. J. Siegel, "Energy cost optimization for geographically distributed heterogeneous data centers," in *Proc. 6th Int. Green Sustainable Comput. Conf.*, Dec. 2015, pp. 1–6.
[22] A. Wierman, Z. Liu, I. Liu, and H. Mohsenian-Rad, "Opportunities and challenges for data center demand response," in *Proc. Int. Green Comput. Conf.*, Nov. 2014, pp. 1–10.
[23] L. Gu, D. Zeng, S. Guo, and B. Ye, "Joint optimization of VM placement and request distribution for electricity cost cut in geo-distributed data centers," in *Proc. Int. Conf. Comput. Netw. Commun.*, Feb. 2015, pp. 717–721.
[24] J. Zhao, H. Li, C. Wu, Z. Li, Z. Zhang, and F. C. M. Lau, "Dynamic pricing and profit maximization for the cloud with geo-distributed data centers," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 118–126.
[25] L. Gu, D. Zeng, A. Barnawi, S. Guo, and I. Stojmenovic, "Optimal task placement with QoS constraints in geo-distributed data centers using DVFS," *IEEE Trans. Comput.*, vol. 64, no. 7, pp. 2049–2059, Jul. 2015.
[26] H. Xu, C. Feng, and B. Li, "Temperature aware workload managementin geo-distributed data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1743–1753, Jun. 2015.
[27] M. Polverini, A. Cianfrani, S. Ren, and A. V. Vasilakos, "Thermal-aware scheduling of batch jobs in geographically distributed data centers," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 71–84, Jan. 2014.
[28] D. Mehta, B. O'Sullivan, and H. Simonis, "Energy cost management for geographically distributed data centres under time-variable demands and energy prices," in *Proc. IEEE/ACM 6th Int. Conf. Utility Cloud Comput.*, Dec. 2013, pp. 26–33.
[29] Z. Abbasi, M. Pore, and S. K. Gupta, "Impact of workload and renewable prediction on the value of geographical workload management," in *Proc. 2nd Int. Workshop Energy Efficient Data Centers*, May 2013, pp. 1–15.
[30] C. Chen, B. He, and X. Tang, "Green-aware workload scheduling in geographically distributed data centers," in *Proc. 4th IEEE Int. Conf. Cloud Comput. Technol. Sci. Proc.*, Dec. 2012, pp. 82–89.
[31] P. Bodik, R. Griffith, C. Sutton, A. Fox, M. I. Jordan, and D. A. Patterson, "Automatic exploration of datacenter performance regimes," in *Proc. 1st Workshop Automated Control Datacenters Clouds*, Jun. 2009, pp. 1–6.
[32] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proc. 5th USENIX Symp. Netw. Syst. Des. Implementation*, Apr. 2008, pp. 337–350.
[33] D. G. Feitelson, D. Tsafrir, and D. Krakov, "Experience with using the parallel workloads archive," *J. Parallel Distrib. Comput.*, vol. 74, no. 10, pp. 2967–2982, Oct. 2014.
[34] M. A. Oxley, E. Jonardi, S. Pasricha, A. A. Maciejewski, H. J. Siegel, P. J. Burns, and G. A. Koenig, "Rate-based thermal, power, and co-location aware resource management for heterogeneous data centers," *J. Parallel Distrib. Comput.*, vol. 112, no. 2, pp. 126–139, Feb. 2018.
[35] V. Shestak, J. Smith, A. A. Maciejewski, and H. J. Siegel, "Stochastic robustness metric and its use for static resource allocations," *J. Parallel Distrib. Comput.*, vol. 68, no. 8, pp. 1157–1173, Aug. 2008.
[36] M. A. Iverson, F. Ozguner, and L. Potter, "Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment," *IEEE Trans. Comput.*, vol. 48, no. 12, pp. 1374–1379, Dec. 1999.

[37] H. Bhagwat, U. Singh, A. Deodhar, A. Singh, and A. Sivasubramaniam, "Fast and accurate evaluation of cooling in data centers," *J. Electron. Packag.*, vol. 137, no. 1, pp. 1–9, Mar. 2015.
[38] ASHRAE Technical Committee 9.9, "Thermal guidelines for data processing environments-expanded data center classes and usage guidance,"American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Inc Tech. Rep., 2011, [Online]. Available: http://ecoinfo.cnrs.fr/IMG/pdf/ashrae_2011_thermal_guidelines_data_center.pdf
[39] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, "Making scheduling "Cool": Temperature-aware workload placement in data centers," in *Proc. USENIX Annu. Tech. Conf.*, Apr. 2005, pp. 61–75.
[40] X. Deng, D. Wu, J. Shen, and J. He, "Eco-aware online power management and load scheduling for green cloud datacenters," *IEEE Syst. J.*, vol. 10, no. 1, pp. 78–87, Mar. 2016.
[41] NREL, "National solar radiation database." [Online]. Available: https://mapsbeta.nrel.gov/nsrdb-viewer/, Accessed on: Apr. 15, 2015.
[42] S. Govindan, J. Liu, A. Kansal, and A. Sivasubramaniam, "Cuanta: Quantifying effects of shared on-chip resource interference for consolidated virtual machines," in *Proc. 2nd ACM Symp. Cloud Comput.*, Oct. 2011, Art. no. 22.
[43] D. Dauwe, E. Jonardi, R. D. Friese, S. Pasricha, A. A. Maciejewski, D. A. Bader, and H. J. Siegel, "HPC node performance and energy modeling with the co-location of applications," *J. Supercomput.*, vol. 72, no. 12, pp. 4771–4809, Dec. 2016.
[44] PARSEC benchmark suite. [Online]. Available: http://parsec.cs.princeton.edu/index.htm/, Accessed on: Aug. 1, 2017.
[45] NAS parallel benchmarks. [Online]. Available: https://www.nas.nasa.gov/publications/npb.html, Accessed on: Aug. 1, 2017.
[46] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of ASICs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 8, no. 6, pp. 661–679, Jun. 1989.
[47] D. Whitley, "The GENITOR algorithm and selective pressure: Why rank-based allocation of reproductive trials is best," in *Proc. 3rd Int. Conf. Genetic Algorithms*, Jun. 1989, pp. 116–121.
[48] Pyevolve: A complete genetic algorithm framework. [Online]. Available: http://pyevolve.sourceforge.net/0_6rc1/index.html, Accessed on: Aug. 1, 2017.
[49] T. D. Braun, H. J. Siegel, N. Beck, L. Bölöni, R. F. Freund, D. Hensgen, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *J. Parallel Distrib. Comput.*, vol. 61, no. 6, pp. 810–837, Jun. 2001.
[50] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *J. Parallel Distrib. Comput.*, vol. 59, no. 2, pp. 107–131, Nov. 1999.
[51] C. Isci, S. McIntosh, J. Kephart, R. Das, J. Hanson, S. Piper, R. Wolford, T. Brey, R. Kantner, A. Ng, J. Norris, A. Traore, and M. Frissora, "Agile, efficient virtualization power management with low-latency server power states," in *Proc. 40th Annu. Int. Symp. Comput. Archit.*, Jun. 2013, pp. 96–107.
[52] G. Cook, T. Dowdall, D. Pomerantz, and Y. Wang, "Clicking clean: How companies are creating the green internet," Greenpeace Inc., Washington, DC, Apr. 2014, [Online]. Available: https://www.greenpeace.org/usa/wp-content/uploads/legacy/Global/usa/planet3/PDFs/clickingclean.pdf
[53] Net metering policies. [Online]. Available: http://www.dsireusa.org/, Accessed on: Aug. 1, 2017.
[54] Scryer: Netflix's predictive auto scaling engine. [Online]. Available: https://medium.com/netflix-techblog/scryer-netflixs-predictive-auto-scaling-engine-a3f8fc922270, Accessed on: Aug. 1, 2017.
[55] Making Facebook's software infrastructure more energy efficient with Autoscale. [Online]. Available: https://code.facebook.com/posts/816473015039157/making-facebook-s-software-infrastructure-more-energy-efficient-with-autoscale/, Accessed on: Aug. 1, 2017.
[56] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Oct. 2015.
[57] J. Xue, F. Yan, R. Birke, L. Y. Chen, T. Scherer, and E. Smirni, "PRACTISE: Robust prediction of data center time series," in *Proc. 11th Int. Conf. Netw. Service Manage.*, Nov. 2015, pp. 126–134.

Ninad Hogade received the BE degree in electronics engineering from the Vishwakarma Institute of Technology, Pune, India. He is currently working toward the graduate degree in electrical engineering at Colorado State University, Fort Collins, Colorado. His research interests include energy aware scheduling of high performance computing systems and data centers. He is a student member of the IEEE.

Sudeep Pasricha received the BE degree in electronics and communications from the Delhi Institute of Technology, in 2000, and the MS and PhD degrees in computer science from the University of California, Irvine, in 2005 and 2008, respectively. He is currently a Monfort and Rockwell-Anderson professor in the Department of Electrical and Computer Engineering and also the Department of Computer Science, Colorado State University. He is a senior member of the IEEE and ACM. More information about him can be found at: http://www.engr.colostate.edu/~sudeep.



Howard Jay Siegel received the BS degree from MIT, and the MSE and PhD degrees from Princeton University. He is a professor Emeritus with Colorado State University. From 2001 to 2017, he was the Abell Endowed chair distinguished professor of electrical and computer engineering there, where he was also a professor of computer science. From 1976 to 2001, he was a professor with Purdue University. He is a fellow of the IEEE and ACM. More information about him can be found at: http://www.engr.colostate.edu/~hj.

Anthony A. Maciejewski received the BSEE, MS, and PhD degrees from the Ohio State University, in 1982, 1984, and 1987, respectively. From 1988 to 2001, he was a professor of electrical and computer engineering with Purdue University, West Lafayette. He is currently a professor and department head of electrical and computer engineering at Colorado State University. He is a fellow of the IEEE. A complete vita is available at: http://www.engr.colostate.edu/~aam.

Mark A. Oxley received the BS degree in computer engineering from the University of Wyoming, and the PhD degree in electrical engineering from Colorado State University, in 2016. His research interests include energy-aware, thermal-aware, and robust resource management techniques. He is currently a research scientist for Numerica Corporation in Fort Collins, Colorado. He is a member of the IEEE.

Eric Jonardi received the BS degree in electrical engineering from Marquette University, in 2013 and the MS degree in electrical engineering from Colorado State University, in 2015. He is currently working as a hardware development engineer for Rockwell Automation. He is a member of the IEEE.