Power and Thermal-Aware Workload Allocation in Heterogeneous Data Centers

Abdulla M. Al-Qawasmeh, Sudeep Pasricha, *Member*, *IEEE*, Anthony A. Maciejewski, *Fellow*, *IEEE*, and Howard Jay Siegel, *Fellow*, *IEEE*

Abstract—Many of today's data centers experience physical limitations on the power needed to run the data center. The first problem that we study is maximizing the performance (quantified by the reward collected for completing tasks by their individual deadlines) of a data center that is subject to total power consumption (of compute nodes and CRAC units) and thermal constraints. The second problem that we study is how to minimize the power consumption in a data center while guaranteeing that the overall performance does not drop below a specified threshold. For both problems, we develop novel optimization techniques for assigning the performance states of cores at the data center level to optimize the operation of the data center. The resource allocation (assignment) techniques in this paper are thermal aware as they consider effects of performance state assignments on temperature and power consumption by the CRAC units. Our simulation studies show that in some cases our assignment technique achieves about 17% average improvement in the reward collected, and about 9% reduction in power consumption compared to an assignment technique that only considers putting a core in the performance state with the highest performance or turning the core off.

Index Terms—Thermal-aware, performance states, data center, CRAC, heterogeneous computing

1 Introduction

Over the last decade, the power consumption of data centers has been increasing at a rapid rate. In a report by the EPA [18], it is estimated that the power consumed by servers and data centers has more than doubled between the years 2000 and 2006. In 2006, it is estimated that the power consumed by servers and data centers was 61 billion kWh, which is equal to 1.5% of the total U.S. electricity consumption that year. This amounts to $\$4.5$ billion in annual electricity costs, equivalent to the power consumption costs of 5.8 million average U.S. households. Motivated by the need to reduce the power consumption of data centers, many researchers have been investigating methods to increase the energy efficiency in computing at different levels: chip, server, rack, and data center (e.g., [2], [7], [8], [10], [14], [21], [24], [28], [29], [32], [36], [40], [45]).

In some cases, there are physical limitations on the amount of power available for data centers. For example, Morgan Stanley is no longer able physically to get the power needed to run a new data center in Manhattan [11]. In a survey of data centers [19], 31% identify power availability as a key factor limiting server deployment. The EPA report also indicates that about 50% of the power consumed in data centers is due to the infrastructure for power delivery and cooling. Therefore, minimizing the power consumed by the cooling infrastructure, can lead to significant overall power savings.

This paper studies two problems. In *Problem 1*, we maximize the performance of a data center that is subject to both a total power consumption constraint (P_{const}) and thermal constraints. The power consumption of the data center is the sum of the power consumption of both Computer Room Air Conditioning (*CRAC*) units and compute nodes. We quantify the performance of the data center as the total reward collected from completing tasks in a workload by their individual deadlines. We consider a data center in the steady state where task flow rates, temperatures at compute nodes and CRAC units, and the power consumption of compute nodes become constant; we assume that the steady state may take on different values during subsequent time intervals. Therefore, the performance is equivalently quantified by the total rate at which reward is collected (the total *reward rate*). In *Problem 2*, we minimize the power consumption of a data center while guaranteeing that the total reward rate does not drop below a specified threshold (R_{const}[1]).

Performance states (*P-states*) of cores provide a trade-off between the power consumed by a core and its performance [23]. *Lower P-states consume more power and provide better performance.* The relationship between the performance and power consumption of the P-states is non-linear. In most cases, the lowest P-state (P0) is not the P-state with the best trade-off between performance and power consumption [30], [41].

P-state assignments in data centers are mainly done at the compute node level. In cases where the workload fluctuates, the P-state of one or more cores is increased when the node's utilization drops below a specified threshold (e.g., [17], [35],

- *A.M. Al-Qawasmeh, S. Pasricha, A.A. Maciejewski, and H.J. Siegel are with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523. E-mail: abdulla@arabellaconsulting.com, sudeep, aam, hj@colostate.edu.*
- *H.J. Siegel is also with the Department of Computer Science, Colorado State University, Fort Collins, CO 80523.*

Manuscript received 20 Sept. 2012; revised 31 Mar. 2013; accepted 12 May 2013. Date of publication 21 May 2013; date of current version 16 Jan. 2015. Recommended for acceptance by D. Bader.
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/TC.2013.116

1. Appendix C, which can be found on the Computer Society Digital Library at https://doi.ieeecomputersociety.org/10.1109/TC.2013.116, provides a list of notations used in this paper.

[40]. However, in a power or performance constrained data center where the workload assigned to a core is constant, the utilization of each core (that is not turned off) in a specific P-state will be close to 100% to avoid idle time. This is because we want to execute as many tasks as possible to obtain the maximum performance for a given power consumption. In this paper, we show that our technique of assigning the P-states when considering the whole data center increases the overall performance (in terms of increased reward or reduced power consumption) of the data center.

The power consumed by compute nodes in the data center is dissipated as heat that is removed by the CRAC units. Our approach of assigning tasks and P-states to cores is *thermal aware* as it considers the temperature evolution effects of P-state assignments, which in turn affects the power consumed by the CRAC units. For both problems studied in this paper, we show how each assignment can be expressed as an exact optimization problem. The P-state assignment part of the problem introduces integer constraints. The integer constraints make each assignment problem not scalable with respect to the number of cores in the data center. A simple relaxation of the integer constraints may introduce additional binary constraints that make each assignment problem not scalable. To address this, we propose novel and scalable assignment techniques for both problems. Each technique involves solving a set of scalable optimization problems. Our techniques are compared against a technique that only considers putting a core in the lowest P-state or turning off the core. We show that using our techniques results in notable performance improvements.

In summary, we make the following novel contributions. Our first contribution is that we model data centers that are power or reward constrained. The second contribution is that we express each assignment problem at the data center level as an optimization problem that includes P-state assignment. The decisions of this optimization problem are: the P-states of cores, the desired number of tasks per unit time allocated to a core, and the outlet CRAC temperatures. The known solution techniques to both optimization problems are not scalable due to integer constraints imposed by the P-states. The third contribution is a scalable assignment technique to find near-optimal solutions for each problem. The fourth contribution is a dynamic scheduler that assigns tasks to cores such that the actual rate of task execution is as close as possible to the desired rate. Finally, the fifth contribution of this research is showing, through simulations, the performance gains of applying our techniques to solve Problems 1 and 2 as opposed to current techniques.

The rest of this paper is organized as follows. Section 2 discusses related work. The data center model is described in Section 3. In Section 4, the thermal constraints in the data center are described. The assignment problems and our solution to the problems are given in Section 5. Section 6 describes the simulation set up. Simulation results are shown in Section 7. Conclusions are given in Section 8.

2 Related Work

In [40], a control system for minimizing the power consumption in blade server enclosures is proposed. The power consumption of the blade server is minimized using three techniques: blade server consolidation, adjusting the speeds of the fans, and assigning P-states to processors. The P-state assignment is based on a simple utilization-based technique. A processor is assigned a P-state so that the utilization is never higher than 80%. However, as discussed in the introduction, this technique is not effective in a power or performance constrained data center because the utilization of each core should be close to 100%. The other two techniques (blade server consolidation and adjusting the speeds of the fans) can be used in future work in combination with our assignment technique to reduce the power consumption.

In [36], it is shown that using an integrated approach to managing the cooling and computational resources in a data center is more efficient than if the two resources were managed independently. Their technique is similar to ours in that it trades-off power consumption with QoS (reward). They trade-off power by deciding the amount of compute resources will be turned on at a compute node. In this paper, we extend that work in two directions. First, we consider a power constrained data center and a reward constrained data center. Second, we show how assigning P-states at the data center level results in improved performance.

The P-state assignment problem for optimizing some objective in a computer system has been studied widely (e.g., [7], [8], [13], [41], [45], [46]). The primary difference between our work and these studies is that our work considers the power consumed by the CRAC units in addition to the power consumed by compute nodes.

The thermal-aware scheduling problem has been previously researched (e.g., [2], [14], [32], [34]). However, unlike our study, none of these papers consider P-state assignment.

Many other techniques to increase the energy efficiency of data centers exist. For example, the Open Compute Project started by Facebook proposes the following two techniques: 1) using a 480 V electrical distribution system to reduce energy loss and 2) reusing hot aisle air in the winter to heat offices. Another example proposed by the Sustainable Ecosystems Research Group at HP is to use water evaporation for cooling instead of using compressors. Many of these techniques can be used in conjunction with our technique to obtain further performance gains.

3 Data Center Model

3.1 Overview

In this section, we give a detailed description of the workload and the different components of the data center. Data centers are typically arranged in a hot-aisle/cold-aisle configuration, as depicted in Fig. 1. CRAC units draw hot air from the top and deliver cold air through the perforated floor tiles in the cold aisles. Compute nodes draw air from the cold aisles. The power consumption of compute nodes causes the temperature inside of the compute nodes to rise. The hot air inside a compute node is expelled into the hot aisle. Due to complex air flow patterns in a data center, some of the hot air exiting a compute node re-circulates into another compute node.

3.2 Workload

We assume that we have a set of T known task types. The arrival rate of tasks of type i is given by λ_i. The types of tasks and their arrival rates in real-world data centers may change over time (e.g., depending on the time of day). This may cause

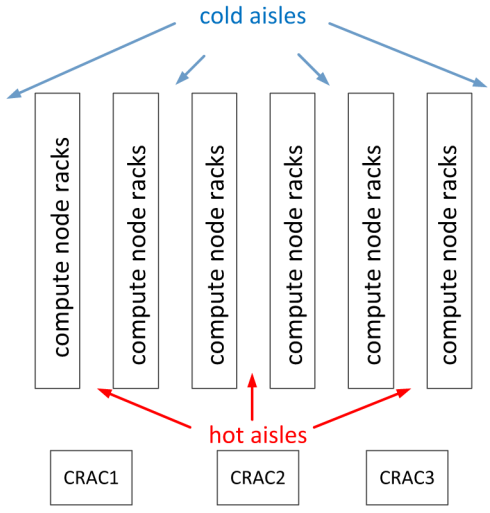


Fig. 1. A hot-aisle/cold-aisle data center layout.

the data center to operate at less than capacity at times and be overloaded at other times. In this paper, we assume that historic data can be used to identify the types of tasks that the data center will be executing and estimate the arrival rates of each of the task types in different time intervals in the future (e.g., [26]). In each of these time intervals, the arrival rate of tasks of each type remains constant. When the arrival rates of tasks change (i.e., we enter a new time interval), the assignment problems in this paper will be solved again for the new arrival rates.

A reward r_i is collected for completing a task of type i by the task's individual deadline. The deadline of a task of type i is given by: $\text{deadline} = \text{arrival time} + d_i$. The value of d_i would be specified by the user. In addition, we assume tasks can be dropped if their deadlines cannot be satisfied. The goal of Problem 1 in this paper is to maximize the total reward that is collected given a constraint on the total power consumption of the data center (the power consumption of compute nodes and CRAC units). The goal of Problem 2 in this paper is to minimize the total power consumption given a constraint on the total reward collected.

3.3 Compute Nodes

Let the number of compute nodes in the data center be NCN. Each compute node has a number of identical cores. Each compute node j belongs to a specific compute node type NT_j. Compute nodes with the same type are identical (i.e., they have the same number and type of cores, and the same power consumption characteristics). The characteristic of compute nodes (e.g., number of cores, the power consumption properties, and/or the computational performance of cores) differs across compute node types. The total number of cores in the data center is $NCORES$. We use a global index for cores. Let CT_k be the type of the compute node to which core k belongs. Because all cores within a compute node are identical, we also refer to CT_k as being the type of core k.

The power consumption of a compute node is the sum of its base power consumption and the power consumption of its cores. The base power consumption is used to model non-compute devices (e.g., disks, fans). The power consumed by the non-compute devices is not affected by the utilization of the cores [33]. Let B_j be the base power consumption of a compute node of type j. We assume that compute nodes are not turned off during the execution of a workload. Therefore, the base power consumption will always be incurred even if the compute node is not executing any tasks.

Each core of type j in the data center can be put in one of η_j P-states. P-state 0 is the P-state with the highest clock frequency and highest power consumption. Each consecutive P-state has a lower clock frequency and lower power consumption. We also consider the case where we can turn off the core.

We model the case where the core is turned off by adding one additional P-state to the available P-states of a core. The turned-off P-state will be the highest P-state (e.g., for cores with P-states from P0 to P5, the turned-off state will be P6). The power consumption of a core of type j running in P-state k is $\pi_{j,k}$.

In some cases, the power consumption of a core is also a function of the task type that it executes. For example, I/O intensive tasks usually consume less power than other tasks [33]. In this work, we assume that the power consumption of a core is dependent on its type and P-state alone. However, it is possible to extend our model to capture the effect of a task type (e.g., I/O or compute intensive task types) on core power consumption. A third index would have to be added to π to represent the effect of a task type on the power consumption of a core. For example, for an I/O bound task type 1 the power consumption of a core of type j running in P-state k is $\pi_{j,k,1}$ which will be less than the power consumption of the same core when it runs a CPU bound task type 2 (i.e., $\pi_{j,k,1} < \pi_{j,k,2}$). Introducing this third index will increase the size of our assignment problems. However, the same assignment techniques will still be applicable.

Let PS_k be the assigned P-state of core k. Let $cores_j$ be the set of indices of cores that belong to compute node j. The power consumption of compute node j, PCN_j, is given by:

$$\text{PCN}_j = B_{\text{NT}_j} + \sum_{k \in \text{cores}_j} \pi_{\text{NT}_j,\text{PS}_k}. \quad (1)$$

The first term refers to the baseline power for a compute node j of type NT_j and the second term refers to the active operational power that depends on the P-state.

3.4 Estimated Computational Speed

We assume that the estimated computational speed (ECS) of a task of type i on a core of type j running in P-state k, ECS(i, j, k), is known. ECS(i, j, k) represents the number of tasks of type i that can be completed per time unit on a core of type j when running in P-state k. The estimated computational speed is equal to the reciprocal of the estimated time to compute (ETC) [4], [5]. The assumption of ETC information is a common practice in resource allocation research (e.g., [9], [15], [22], [25], [27], [37], [42]). The ETC values for a given system can be obtained from user supplied information, experimental data, or task profiling and analytical benchmarking [3], [22], [27], [42]. Obviously, when the core is turned-off, the ECS of a task of any type is 0, i.e., $\text{ECS}(i, j, \eta_j - 1) = 0$ for all i and j.

3.5 Computer Room AC Units

We assume that the number of CRAC units in a data center is $NCRAC$. CRAC units are used to remove the heat generated by the compute nodes. The power consumed by a CRAC unit is

equal to the ratio of the amount of heat removed at that CRAC unit to the *Coefficient of Performance* (*CoP*) of that CRAC unit [32].

The amount of heat removed by a CRAC unit i depends on the inlet air temperature, $TCRAC_i^{\text{in}}$ (which is directly affected by the heat generated by compute nodes), and the assigned outlet air temperature, $TCRAC_i^{\text{out}}$ (which is the temperature of the cool air to be generated by the CRAC). Let ρ be the density of air, Cp be the specific heat capacity of air, and $FCRAC_i$ be the air flow rate at CRAC unit i. The amount of heat removed per unit time at CRAC unit i is equal to [39]:

$$\rho \cdot \text{Cp} \cdot \text{FCRAC}_i \cdot (\text{TCRAC}_i^{\text{in}} - \text{TCRAC}_i^{\text{out}}). \quad (2)$$

The CoP of a CRAC unit is a function of its outlet temperature [32]. The power consumed by CRAC unit i, $PCRAC_j$, is given by [32]

$$\text{PCRAC}_i = \frac{\rho \cdot \text{Cp} \cdot \text{FCRAC}_i \cdot \left(\text{TCRAC}_i^{\text{in}} - \text{TCRAC}_i^{\text{out}}\right)}{\text{CoP}\left(\text{TCRAC}_i^{\text{out}}\right)}. \quad (3)$$

When the inlet air temperature of a CRAC unit is less than or equal to the assigned outlet temperature (there is no heat to be removed) the power consumption is 0.

4 Thermal Constraints

Due to the law of energy conservation, the power consumed at a compute node will be dissipated as heat causing an increase in the temperature of the air going through the compute node. To maintain the reliability of the CRACs and compute nodes, CRAC units must remove the heat generated by the compute nodes so that the inlet air temperature of the CRACs and compute nodes are kept at or below a redline temperature. Let TCN_i^{in} and TCN_i^{out} be the inlet and outlet air temperatures at compute node i, respectively. Let FCN_i be the air flow rate at compute node i. The outlet air temperature of compute node i is given by [39]

$$\text{TCN}_i^{\text{out}} = \text{TCN}_i^{\text{in}} + \left(\frac{\text{PCN}_j}{\rho \cdot \text{Cp} \cdot \text{FCN}_i}\right). \quad (4)$$

Air flow patterns in data centers are complex. The inlet temperatures of CRAC units and compute nodes are affected by the outlet temperatures of other CRAC units and compute nodes [39]. Let

$$T^{\text{out}} = [\text{TCRAC}_1^{\text{out}}, \ldots, \text{TCRAC}_{\text{NCRAC}}^{\text{out}}, \text{TCN}_1^{\text{out}}, \ldots, \text{TCN}_{\text{NCN}}^{\text{out}}]^T,$$

and $T^{\text{in}} = [\text{TCRAC}_1^{\text{in}}, \ldots, \text{TCRAC}_{\text{NCRAC}}^{\text{in}}, \text{TCN}_1^{\text{in}}, \ldots, \text{TCN}_{\text{NCN}}^{\text{in}}]^T$. Using the "Abstract Heat Flow Model" proposed in [39], we can compute each element of T^{in} as a linear combination of the elements of T^{out}, i.e.,

$$T^{\text{in}} = AT^{\text{out}}. \quad (5)$$

The values in matrix A can be estimated using sensor measurements [39]. Let T^{redline} be the vector of redline temperature constraints on inlet air temperatures, which gives the constraint

$$T^{\text{in}} \leq T^{\text{redline}}. \quad (6)$$

The inequality is element-wise (i.e., element i in vector T^{in} must be less than or equal to element i in vector T^{redline})

5 Assignment Problems

5.1 Overview

Given a workload composed of a set of tasks arriving at different times, the goal of assignment Problem 1 is to maximize the total reward rate subject to a constraint on the maximum total power consumption. The goal of assignment Problem 2 is to minimize the power consumption of the data center subject to a constraint on the minimum reward rate. Both problems are subject to thermal constraints (i.e., the redline temperatures at the inlets must not be exceeded). The decisions that both assignment problems must make are: 1) the P-states of cores, 2) the task to core assignments, and 3) the outlet temperatures of the CRAC units.

Because we assume the arrival rates of task types remain relatively constant during the steady state, the decision variables will remain constant. When the arrival rates change, then we will have a new time interval and another optimization problem that we will have to solve to obtain the new values of the decision variables.

Temperature evolution in the data center is in orders of minutes, while the execution of a task is in orders of seconds or milliseconds. To make workload assignments tractable, previous research (e.g., [2], [36]) has used a *two-stage assignment* approach. The first stage manages the power and the thermal evolution in the data center, while the second stage performs workload balancing. In this paper, we apply the two-stage assignment approach for both of our assignment problems. In the first stage, our approach assigns the P-states of cores, the desired execution rate of task types on cores, and the outlet temperatures of CRAC units. The first stage guarantees that the thermal constraints and the power constraint for Problem 1 or the reward constraint for Problem 2 are not violated. In the second stage, our approach implements a dynamic scheduler that assigns tasks to cores so that the actual execution rate of each task type on each core approaches the desired execution rate set by the first step. The dynamic scheduler can also make the decision to drop a task. The two-stage assignment is depicted in Fig. 2.

In Section 5.2, we focus on the first-stage assignment problem for solving Problem 1. The difference between the first-stage assignment for Problem 1 and Problem 2 is shown in Section 5.3. In Section 5.4, we propose a dynamic scheduler to assign incoming tasks to cores.

5.2 First-Stage Assignment: Problem 1

5.2.1 Overview

In Section 5.2.2, we formulate the assignment problem as an exact *mixed integer nonlinear program* (MINLP). Because the solution techniques for solving the exact MINLP are not scalable, we propose a scalable technique to find near-optimal solutions. The technique divides Stage 1 into three steps. The

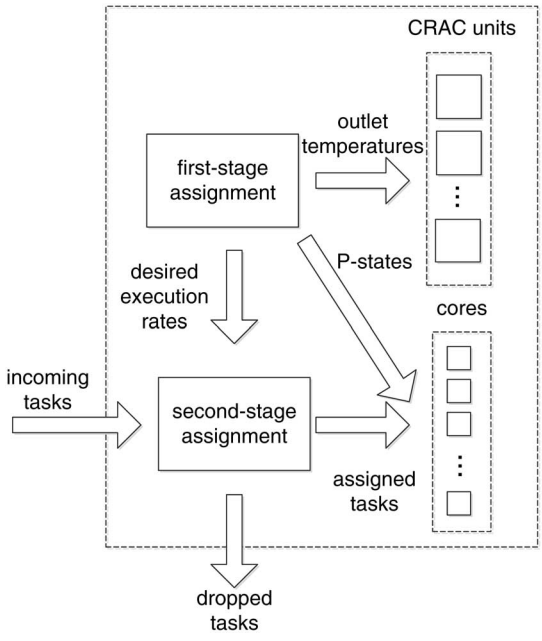


Fig. 2. The assignment problem in the data center. The first stage assigns the outlet temperatures of CRAC units, the P-states of cores, and the desired execution rate of task types on cores. The second stage assigns the incoming tasks to cores based on the desired execution rate set by the first step or drops tasks that cannot make their deadline.

first step assigns power budgets to compute nodes, CRAC outlet temperatures, and the fraction of time each core spends running tasks of each type. The second step converts the power budget assignment into a P-state assignment for each core. Finally, Step 3 uses the exact P-state assignment in Step 2 to assigns the desired execution rate of each task type on each compute node.

5.2.2 Problem Formulation

The decisions made at the first stage are: the outlet temperature of each CRAC unit ($\mathrm{TCRAC}_i^{\mathrm{out}}$), the P-state of each core (PS_k), and the desired rate of executing tasks of each type on each core. The desired rates are organized in a matrix ER. Entry ER(i, k) represents the desired execution rate of tasks of type i on core k. Once a P-state of a core is assigned, we assume that it is not changed. Therefore, the first stage assignment is considered as static assignment.

The following equation shows the assignment problem for Problem 1:

$$\text{maximize} \sum_{i=1}^{T} \left(r_i \sum_{k=1}^{\mathrm{NCORES}} \mathrm{ER}(i,k) \right), \tag{7}$$

subject to:

1. $\sum_{i=1}^{T} \mathrm{ER}(i,k)/\mathrm{ECS}(i,\mathrm{CT}_k,\mathrm{PS}_k) \leq 1,$
   $k = 1, \ldots, \mathrm{NCORES}.$
2. $\mathrm{ER}(i,k)(d_i - (1/\mathrm{ECS}(i,\mathrm{CT}_k\mathrm{PS}_k)) \geq 0$
   $i = 1, \ldots, T$ and $k = 1, \ldots, \mathrm{NCORES}$
3. $\sum_{k=1}^{\mathrm{NCORES}} \mathrm{ER}(i,k) \leq \lambda_i, \quad i = 1, \ldots, T.$
4. $\sum_{j=1}^{\mathrm{NCN}} \mathrm{PCN}_j + \sum_{i=1}^{\mathrm{NCRAC}} \mathrm{PCRAC}_i \leq P_{\mathrm{const}}.$
5. $T^{\mathrm{in}} \leq T^{\mathrm{redline}}.$

The objective function is the total reward rate. The first constraint guarantees that the desired execution rate of task types on a core will not exceed the core's ability to complete the tasks. When the estimated execution time of a task of type i on a core of type j running in P-state k (i.e., $1/\mathrm{ECS}(i, j, k)$) is greater than d_i, no task of type i can make its deadline on core k even if its execution starts immediately after its arrival. Therefore, if $1/\mathrm{ECS}(i,j,k) > d_i$, then Constraint 2 guarantees that $\mathrm{ER}(i,k) = 0$ to avoid executing tasks of type i on core k. Constraint 3 guarantees that the sum of the desired execution rate of a task type on all cores does not exceed its arrival rate. The power constraint is guaranteed by Constraint 4. Finally, Constraint 5 guarantees the thermal constraints.

Note that there are two cases where ECS values can be 0. First, when PS_k is the turned-off P-state, the ECS of any task type on core k is 0. Second, a core type may not be able to execute certain task types (for example, due to certain required software not being installed on the corresponding node type). When an ECS value is 0, 1/ECS will not be defined. However, we can solve this issue by assuming that the ECS value is a "small enough" positive number.

The problem in Equation 7 is a MINLP for the following two reasons. First, the above problem contains integer constraints due to the requirement that the P-states be integers. Second, the measured CoP of the CRAC units at the HP Labs Utility Data Center as a function of CRAC output temperature, τ, is given by [32]

$$\mathrm{CoP}(\tau) = 0.0068\tau^2 + 0.0008\tau + 0.458. \tag{8}$$

For this CoP, the power consumption of the CRAC units (Equation 3) is nonlinear (and non-convex), which makes constraint 4 a nonlinear (and non-convex) constraint.

MINLPs belong to the class of NP-hard problems. Finding the optimal solution of such problems is computationally infeasible for large problem sizes. For example, consider a compute node that has 32 cores and that each core can be put in one of 5 P-states. This gives us $5^{32} \approx 2.3 \times 10^{22}$ P-state assignment combinations.

In the following subsections, we show how the Stage 1 assignment is divided into three steps to relax the integer P-state constraints. In the first step, instead of assigning P-states to cores, we assign power consumption to cores. This makes the assignment problem simpler. The decision variables in the first step are the power consumption of each compute node, the outlet temperature of each CRAC unit, and the fraction of time each compute node spends on tasks of each type. The second step converts the compute node power assignment into a P-state assignment. The third step assigns the desired execution rate of each task type on each core for the P-state assignment obtained from the second step.

5.2.3 Step 1 Assignment

Relaxing the integer P-state constraint means that we allow a core to be assigned a continuous P-state value rather than a discrete one. Therefore, we have to define core power consumption and ECS functions for continuous P-states. Another equivalent assignment problem is to assume that cores can be assigned a continuous power value between zero and the power consumption in P-state 0. We use this equivalent

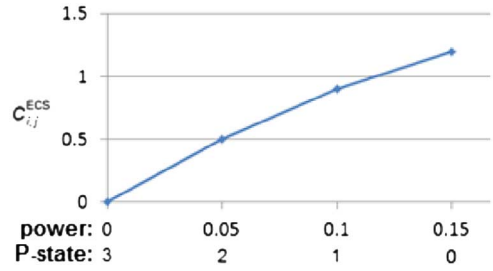


Fig. 3. An example $C_{i,j}^{\mathrm{ECS}}$ function.

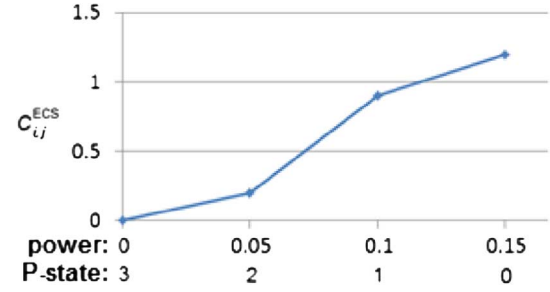


Fig. 4. An example of a non-concave $C_{i,j}^{\mathrm{ECS}}$ function.

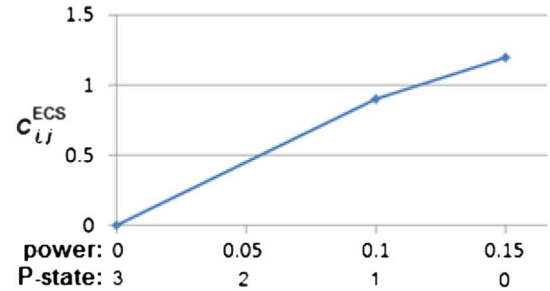


Fig. 5. An illustration of the calculation of the $C_{i,j}^{\mathrm{ECS}}$ function in Fig. 4 when the "bad" P-state is ignored.

assignment problem because it makes the representation of the assignment problem easier (we relate power directly with performance), and it eliminates the need to define power consumption functions for continuous P-states.

For the relaxed problem, the ECS of a task of type i on core k is a continuous function of the power consumption of the core. Let $PCORE_k$ be the power assigned to core k. Let $C_{i,j}^{\mathrm{ECS}}(\mathrm{PCORE_k})$ be the ECS for a task of type i running on a core of type j as a continuous function of PCORE_k. To minimize the difference between the integer solution and the relaxed solution of the Step 1 assignment, we select $C_{i,j}^{\mathrm{ECS}}$ so that it goes through the points

$$(\pi_{j,0}\mathrm{ECS}(i,j,0)),\ldots,(\pi_{j,\eta_j-1},\mathrm{ECS}(i,j,\eta_j-1)).$$

Each of these points is the power consumption of a P-state and the ECS at that P-state. Intuitively, one can view the value of $C_{i,j}^{\mathrm{ECS}}$ when PCORE_k is not equal to the power consumption of any P-state, is to assume that the core can switch between a P-state with power consumption lower than PCORE_k and a P-state with a power consumption higher than PCORE_k such that the average power consumption is equal to. Therefore, we chose to represent $C_{i,j}^{\mathrm{ECS}}$ using a piecewise linear function.

For example, assume a core of type j with four P-states. The power consumption of P-states 0, 1, 2, and 3 is 0.15, 0.1, 0.05, and 0 Watts (W), respectively. The ECS PCORE_k values for task type i for each of the four P-states are 1.2, 0.9, 0.5, and 0, respectively. The $C_{i,j}^{\mathrm{ECS}}$ function is a linear piecewise function that goes through the points (0, 0), (0.05, 0.5), (0.1, 0.9), and (0.15, 1.2). This function is shown in Fig. 3.

Because for both Problems 1 and 2 a higher ECS value will result in a better solution, if the $C_{i,j}^{\mathrm{ECS}}$ function is concave, then the computational expense of the optimization can be greatly reduced by representing the $C_{i,j}^{\mathrm{ECS}}$ function with linear constraints. The $C_{i,j}^{\mathrm{ECS}}$ function, however, is not guaranteed to be concave. In that case, an equivalent representation is only achieved by introducing additional binary constraints. The introduction of the binary constraints would make the Stage 1 optimization problem computationally infeasible for a large number of cores. For instance, consider the example shown in Fig. 4 where the number of P-states is four (i.e., $\eta_j = 4$). The ECS values for the P-states 0, 1, 2, and 3 are 1.2, 0.9, 0.2, and 0, respectively. This $C_{i,j}^{\mathrm{ECS}}$ function is not a concave function.

The non-concavity of an $C_{i,j}^{\mathrm{ECS}}$ function is caused by a P-state that has an ECS to power consumption ratio that is less than its next lower P-state. We call this P-state a "bad" P-state.For the $C_{i,j}^{\mathrm{ECS}}$ function in Fig. 4, P-state 2 is a "bad" P-state because the ratio of its ECS to its power consumption is 4, where the ratio of P-state 1's ECS to its power consumption is 9. If we ignore P-state 2 (the "bad" P-state), then the $C_{i,j}^{\mathrm{ECS}}$ function will be concave. This case is shown in Fig. 5. We ignore "bad" P-states (i.e., do not assign a core a "bad" P-state) so that we can reduce the computational complexity of the Step 1 assignment.

In general, when the "bad" P-states are not ignored, the Step 1 assignment will still avoid "bad" P-states. For example, consider the case where a compute node of type j has two cores. Assume that the compute node can assign a maximum of 0.1 W total power to its cores. Assume that there is only one task type and it has a reward of 1, i.e., $r_i = 1$. If the function in Fig. 4 is the ECS of that compute node, then the optimal solution in this case would be to put one of the cores in P-state 1 (i.e., assign 0.1 W power to it) and the other in P-state 3 (i.e., assign 0 W power to it). This will result in a total reward rate of 0.45, which is the same as when "bad" P-states are ignored. It should be noted that the optimal value when the "bad" P-states are ignored is never better than the optimal value when the "bad" P-states are *not* ignored.

Let $DF(i,k)$ be the desired fraction of time that core k spends executing tasks of type i. The desired execution rate of task type i on core k, $\mathrm{ER}(i,k)$, is equal to

$$\mathrm{DF}(i,k)\mathrm{C}_{i,\mathrm{CT}_k}^{\mathrm{ECS}}(\mathrm{PCORE}_k).$$

The Step 1 (relaxed) optimization problem is obtained from Equation 7 by replacing $\mathrm{ER}(i,k)$ with

$$\mathrm{DF}(i,k)\mathrm{C}_{i,\mathrm{CT}_k}^{\mathrm{ECS}}(\mathrm{PCORE}_k),$$

and $\mathrm{ECS}(i,\mathrm{CT}_k,\mathrm{PS}_k)$ with $C_{i,\mathrm{CT}_k}^{\mathrm{ECS}}(\mathrm{PCORE}_k)$. The effect of Constraint 2 is captured by considering a P-state k of core type j a "bad" P-state for task type i if $d_i < 1/\mathrm{ECS}(i,j,\mathrm{k})$.

Equation 9 and its constraints represent the Step 1 optimization problem.

$$\text{maximize} \sum_{i=1}^{T} \left(\sum_{k=1}^{\text{NCORES}} r_i \text{DF}(i,k) C_{i,k}^{\text{ECS}}(\text{PCORE}_k) \right), \quad (9)$$

subject to:

1. $\sum_{i=1}^{T} \text{DF}(i, \text{k} \leq 1)$, $k = 1, \ldots, \text{NCORES}$.
2. $\sum_{k=1}^{\text{NCORES}} \text{DF}(i, \text{k}) C_{i,k}^{\text{ECS}}(\text{PCORE}_k) \leq \lambda_i, \quad i = 1, \ldots, T$.
3. $\sum_{j=1}^{\text{NCN}} \text{PCN}_j + \sum_{i=1}^{\text{NCRAC}} \text{PCRAC}_i \leq P_{\text{const}}$.
4. $T^{\text{in}} \leq T^{\text{redline}}$.

Because all the cores within a compute node are homogenous, we can reduce the time to find a solution for Equation 9 by assuming that all cores within a compute node will be assigned the same value of DF for each task type and the same power consumption (PCORE_k). This reduces the size of matrix DF and the number of ECS functions to be equal to $T \cdot \text{NCN}$.

The problem in Equation 9 is an NLP. To avoid locally optimal solutions that have $\sum_{i=1}^{T} \text{DF}(i, \text{k}) \leq 1$ (i.e., core k is not fully utilized), we substitute the inequality in Constraint 1 with an equality. Even with this change, a solution to Equation 9 may be locally (and not globally) optimal. Different locally optimal solutions may be obtained from different initial starting points. Therefore, we try multiple random starting points. Details on how we decided on the number of random start points are in Section 7 (simulation results).

5.2.4 Step 2 Assignment

The purpose of the Step 2 assignment is to convert the power assigned to a core k into a P-state. The solution to Step 1 guarantees that the power and thermal constraints are satisfied. Therefore, the power consumption at any compute node should be kept at or below the power consumption that resulted from the Step 1 assignment. We design the following heuristic to convert the PCORE_k values into a P-state assignment:

1. For each core k, assign it the highest possible P-state that results in a power consumption greater than or equal to PCORE_k.
2. For each compute node j
   While the power consumption calculated by Equation 1 is greater than the power consumption that resulted from Step 1
   Increase the P-state of the core with the smallest P-state the next non-bad P-states.

Because the $C_{i,j}^{\text{ECS}}$ functions are concave, the ECS to power consumption ratio of a P-state will always be lower than or equal to that of a higher P-state. Therefore, in Step 2 of the procedure above we increase the P-state of the core with the lowest P-state. In cases where there are multiple task types assigned to a core, we only ignore the bad P-states of the task type that gives the most reward rate on that core.

5.2.5 Step 3 Assignment

In Step 3, we solve Equation 7 to determine the optimal desired execution rate of task types on cores (i.e., the optimal ER matrix) using the outlet temperature of CRACs that is determined in Step 1 and the discrete P-state assignment determined in Step 2, which make the solution to Equation 7 a simple *linear program* (*LP*).

5.3 First-Stage Assignment: Problem 2

The exact formulation of the first-stage assignment for Problem 2 is similar to the formulation of Problem 1 (given in Equation 7) except that the objective function for Problem 2 is to minimize power consumption subject to a constraint on the minimum total reward rate. The exact formulation of Problem 2 is also a MINLP. Therefore, we propose a three-step approach for solving Problem 2. Similar to the first step of Problem 1, the first step of Problem 2 uses the $C_{i,j}^{\text{ECS}}$ functions to relax the integer P-state assignment constraints.

Step 2 of the first-stage assignment for Problem 2 converts the power consumption assignment of cores into a discrete P-state assignment. To guarantee that the total reward rate constraint is not violated, the conversion at a specific compute node j must guarantee that the cores in their assigned P-states will collectively be capable of executing task types at a rate that is greater than or equal to the total desired execution rate that is set by Step 1 for compute node j.

A simple way to guarantee that the total reward rate constraint is not violated at a compute node j is to assign each core in compute node j to the highest possible P-state that results in a power consumption greater than or equal to PCORE_k. This guarantees the total reward rate constraint because we assume that $\text{ECS}(i,j,p) \geq \text{ECS}(i,j,p+1)$.

One may be able to improve this simple P-state assignment (i.e., reduce the power consumption) by incrementing the P-state of some cores. However, incrementing the P-state of any core will require reassigning the desired execution rate of task types among the compute node cores so that the total reward rate constraint is satisfied. We show how a mixed integer program can be used to find the optimal P-state assignment that satisfies the total reward rate constraints. Because finding the optimal solution of a mixed integer program is an NP-hard problem, we design a heuristic procedure that can be used to find a near-optimal solution.

Let the reassigned desired execution rates of task types on cores be arranged in matrix RER. Entry $\text{RER}(i,k)$ represents the reassigned desired execution rate of task type i on core k. For cores in compute node j, the following mixed integer program finds the optimal P-state assignment and reassigned desired execution rates for cores that belong to compute node j:

$$\text{minimize} \sum_{k \in \text{cores}_j} \pi_{\text{NT}_j, \text{PS}_k}, \quad k \in \text{cores}_j, \quad (10)$$

subject to:

1. $\sum_{i=1}^{T} \text{RER}(i, \text{k}) / \text{ECS}(i, \text{CT}_k \text{PS}_k) \leq 1$, $k \in \text{cores}_j$
2. $\text{RER}(i,k)(m_i - (1/\text{ECS}(i, \text{CT}_k, \text{PS}_k)) \geq 0$
   $i = 1, \ldots, T$ and $k \in \text{cores}_j$.
3. $\sum_{k \in \text{cores}_j} \text{RER}(i,k) \geq \sum_{k \in \text{cores}_j} \text{ER}(i,k)$, $i = 1, \ldots, T$.

Constraints 1 and 2 are similar to Constraints 1 and 2 in Equation 7. Constraint 3 guarantees that the total available execution rate for each task type at compute node j is greater than or equal to the total desired execution rate set by Step 1. The problem in Equation 10 contains integer constraints due

to the P-state assignment that makes it computationally intractable for many of today's compute nodes that contain a large number of cores. However, for a fixed P-state assignment, the problem in Equation 10 is a LP feasibility problem. We use the LP feasibility problem to test whether a specific P-state assignment satisfies the total reward rate constraints.

We designed the following procedure to convert the power consumption of cores into a P-state assignment:

1. For each core k, assign it the highest possible P-state that results in a power consumption greater than or equal to PCORE_k.
2. For each compute node j
   a. Until a feasible solution exists for Problem 10
      Increase the P-state of the core with the smallest P-state to the next non-bad P-state.

The output of Step 2 is a P-state assignment and the matrix RER (which now becomes the new ER). Similar to Step 2 of Problem 1, Step 2 of Problem 2 avoids "bad" P-states.

The P-state assignment of Step 2 may violate the thermal constraints. Therefore, Step 3 solves the exact optimization problem for Problem 2. Because the P-state and desired execution rates are determined, the exact optimization problem becomes an NLP (due to the power consumption of CRAC units).

5.4 Second Stage Assignment

The second stage assignment for both Problems 1 and 2 is the same. The dynamic scheduler at the second step keeps track of the actual execution rate of each task type on each core in matrix AER. The goal of the dynamic scheduler is to make the ratio of $\mathrm{AER}(i,k)/TC(i,k)$ as close as possible to 1 for each task type i and core k.

For each incoming task t, the dynamic scheduler maps t to a core that can complete it before its deadline and has the minimum $\mathrm{AER}(i,k)/TC(i,k)$ value that is less than or equal to 1. If no such core exists, then the dynamic scheduler drops t.

6 Simulation Setup

6.1 Overview

We conducted simulation studies to evaluate the effectiveness of our assignment technique. In this section, we show how the parameters of the simulations were generated.

Real-world data centers can vary widely in the number and type of compute nodes, the number and type of CRAC units, and the arrival rate and type of task types. For illustration purposes, we set up our simulations with eight task types, two compute node types, and three CRAC units.

6.2 Compute Nodes

In our simulations, we used a varying number of compute nodes. Each compute node belongs to one of two compute node types based on two 7U servers listed in the SPECpower_ssj2008 results [38]. The first compute node type is based on the HP ProLiant DL785 G5 server. This server has eight AMD Opteron 8381 HE processors with four cores in each processor. The second compute node type is based on the NEC Express5800/A1080a-S server. This server has four Intel Xeon X7560 processors. Each processor has eight cores. Table 1 lists the parameters of both node types. Details on how the values of the parameters were obtained are in Appendix A.

TABLE 1
Parameters of the Two Node Types Used in Simulations

node type	1	2
base power consumption (kW)	0.353	0.418
number of cores	32	32
number of P-states	4	4
power consumption of P-state 0 (kW)	0.01375	0.01625
clock frequencies of P-states (MHz)	2500, 2100, 1700, 800	2666, 2200, 1700, 1000
air flow rate (m^3/s)	0.07	0.0828

We used a uniform random variable to assign a node type to compute nodes. Each compute node type has an equal probability of being assigned to a compute node.

6.3 ECS Matrices

The estimated computation speed values are arranged in a three-dimensional ECS matrix. These dimensions represent task types, node types, and P-states. In a real world environment, the ECS values can be based on user supplied information, experimental data, or task profiling and analytical benchmarking (e.g., [3], [20], [22], [27], [31], [43], [44]). The following paragraphs discuss how we generated synthetic ECS data for the purposes of our simulations.

In our simulations, we have eight task types, two compute node types, and four P-states (not including the turned-off P-state). First, we generate a two dimensional ECS matrix for P-state 0. The columns represent the node types and the rows represent the task types. The ratio of the performance of node type 1 to node type 2 is 0.6 (this is based on the number of server side Java operations per second each node type can perform [38]). Therefore, we assumed that the average ECS over all task types for node types 1 and 2 is 0.6 and 1, respectively. Let $NTYPES$ be the number of compute node types in the data center. The easiness of a task type i, $\underline{\mathrm{TE}}_i$, is assumed to be equal to the sum of the ECS values over all node types for P-state 0, i.e., $\underline{\mathrm{TE}}_i \sum_{j=1}^{\mathrm{NTYPE}} \mathrm{ECS}(i,j,0)$. Without loss of generality, we assume that the easiness for task type i is half that of task type $i+1$. Let $rand[a,b]$ be a uniform random variable in the interval $[a,b]$. Entry (i,j) in the two-dimensional ECS for P-state 0 is the product of the average ECS for task type i, the average ECS for node type j, and a variation factor $\mathrm{rand}[1-V_{\mathrm{ECS}}, 1+V_{\mathrm{ECS}}]$. The variation factor is used so that the affinity between task types and node types differs across the types. The value of $\underline{V}_{\mathrm{ECS}}$ that we used is 0.1.

Let $f_{j,k}$ be the clock frequency of a core of type j running in P-state k. In many cases, the ECS of task types on cores is not exactly proportional to clock frequency. For example, reducing the clock frequency will have less impact on the ECS of a task that is I/O bound versus a task that is CPU bound. Therefore, we use a parameter V_{prop} so that the ECS of a task type on a core type is not exactly proportional to the clock frequency of the P-states. The ECS is extended in the third dimension by using

$$\mathrm{ECS}(i,j,k) = \mathrm{ECS}(i,j,0) \cdot \frac{f_{j,k}}{f_{j,0}} \cdot \mathrm{rand}\,[1-V_{\mathrm{prop}}, 1+V_{\mathrm{prop}}]. \quad (11)$$

We used two different values for V_{prop} in different sets of simulations, i.e., 0.1 and 0.3.

Using Equation 11 may result in a P-state that has a higher ECS value for a specific task type and a specific core type than a lower P-state. To prevent this case, if an entry (i, j, k) is higher than entry $(i, j, k-1)$, then we generate a random number $\text{rand}[1 - V_{\text{prop}}, 1 + V_{\text{prop}}]$ for $\text{ECS}(i, j, k)$ until it is less than $\text{ECS}(I, j, k-1)$. We start by generating the ECS for P-state 1 then P-state 2 and so on.

6.4 Task Types

The number of task types in all of our simulations is assumed to be eight. The reward for completing a task of type i by its deadline is assumed to be equal to the reciprocal of its easiness, i.e.,

$$r_i = 1/\text{TE}_i. \tag{12}$$

We also have conducted simulations to show how different reward values will affect the performance of our assignment techniques (see Section 7.2.5).

Now we show how the d_i values that are used to calculate the deadline of individual tasks are generated. Let $MinECS_i$ and $MaxECS_i$ be the minimum and maximum ECS values for task type i over all core types and all P-states except the turned-off P-state. MinECS_i is given by

$$\text{MinECS}_i = \min[\text{ECS}(i, j, \eta_j - 2)] 1 \leq j \leq \text{NTYPES}. \tag{13}$$

MaxECS_i is given by

$$\text{MaxECS}_i = \max[\text{ECS}(i, j, 0)] 1 \leq j \leq \text{NTYPES}. \tag{14}$$

The value of d_i is given by

$$d_i = 1.5\text{rand}[1/\text{MaxECS}_i, 1/\text{MinECS}_i]. \tag{15}$$

We used Equation 15 to compute d_i because it guarantees that there is at least one core type that can make the deadline of a task of type i. There is also a chance of generating a task type such that some of its tasks' deadlines can be met by all core types running at their lowest frequency.

The last parameter that needs to be generated for a task type is its arrival rate, λ_i. Let $SumECS_i$ be the ECS obtained for a task type i if all cores in the data center are used equally by every task type and all cores are running in P-state 0. The value of SumECS_i is given by

$$\text{SumECS}_i = \sum_{k=1}^{\text{NCORES}} \text{ECS}(i, CT_k, 0)/T. \tag{16}$$

Our goal is to assign arrival rates for task types such that the data center can complete all the arriving tasks when running at full capacity (i.e., all cores in P-state 0) but would be oversubscribed if there is a power constraint (i.e., there is not enough power to run all cores in P-state 0). This is not simple to achieve. Therefore, we use SumECS_i to approximate the arrival rates. In addition, to introduce some randomness in the assigned arrival rate of task type i, we use a parameter, V_{arrival}. Once the arrival rate for a task type is assigned, it remains constant. The arrival rate of task type i is given by

$$\lambda_i = \text{SumECS}_i \cdot \text{rand}[-V_{\text{arrival}}, 1 + V_{\text{arrival}}]. \tag{17}$$

The value of V_{arrival} that we used is 0.3.

6.5 Cross Interference Coefficients

In [39], for two compute nodes i and j, the *cross interference coefficient*, $\alpha_{i,j}$, is the percentage of air recirculated from compute node i to compute node j. Computational Fluid Dynamics (CFD) simulations were used in [39] to obtain cross interference coefficients for a small data center (ten racks with five compute nodes in each rack, and one CRAC unit). The time consumed for a single run of a CFD simulation was about an hour with a CFD simulation required for each of the 50 compute nodes [39]. In our simulations, we use 150 compute nodes and three CRAC units. The amount of time to run the CFD simulations for each data center in our simulations is prohibitive. In Appendix B, we show how an LP feasibility problem can be used to generate the cross interference coefficients. Our goal is not to propose a method of calculating the cross interference coefficients for a real data center. Rather, our goal is to generate cross interference coefficients for simulation studies that are based on realistic information about the air flows in data centers.

6.6 Power and Thermal Constraints

To set a reasonable power constraint in our simulations for Problem 1, we need to find the minimum and maximum power consumption of the data center. The minimum power consumption occurs when all cores in the data center are turned off. The maximum power consumption occurs when all cores are running in P-state 0. The minimum and maximum power consumption of the data center can be found using the NLP problem below solved for the two extreme values of PCN_j. The solution for this problem will provide the power consumption bounds of the data center. The decision variables are the outlet temperatures of CRAC units. Because it is an NLP problem, our solution to the problem will not necessarily provide the global minimum. Therefore, the solutions are considered an upper bound of the minimum and maximum power consumption of the data center.

$$\text{minimize}\left[\sum_{j=1}^{\text{NCN}} \text{PCN}_j + \sum_{i=1}^{\text{NCRAC}} \text{PCRAC}_i\right], \tag{18}$$

subject to

$$T^{\text{in}} \leq T^{\text{redline}}.$$

Let $P_{\min}$ and $P_{\max}$ be the upper bounds on the minimum and maximum power consumption of the data center, respectively. Let Φ be a "power multiplier" that takes values in the interval [0,1]. The power multiplier allows us to select a power constraint that is between the minimum and the maximum power consumption bounds. The power constraint, P_{const}, is given by

$$P_{\text{const}} = P_{\min} + (P_{\max} - P_{\min}) \cdot \Phi. \tag{19}$$

The redline inlet air temperature was set at 25° Celsius for compute nodes and 40° Celsius for CRAC units.

6.7 Total Reward Constraint

To set a reasonable total reward rate constraint for Problem 2, we need to find the maximum possible total reward rate that occurs when all cores are running in P-state 0. Let $FRAC(i, j)$

be the average fraction of a core in compute node j that is used to execute task of type i. The effective number of cores at compute node j that are used to execute tasks of type i, $E(i,j)$, is equal $|\text{cores}_j|\text{FRAC}(i,j)$. The maximum reward rate can be found using the following LP:

$$\text{maximize} \sum_{i=1}^{T} \sum_{j=1}^{\text{NCM}} r_i \text{ECS}(i,j,0) E(i,j). \quad (20)$$

Subject to

1. $\sum_{j=1}^{\text{NCN}} \text{ECS}(i,j,0) E(i,j) \leq \lambda_i, 1 \leq i \leq T$,
2. $\sum_{i=1}^{T} E(i,j) \leq |\text{cores}_j|, 1 \leq j \leq \text{NCN}$,
3. $T^{\text{in}} \leq T^{\text{redline}}$.

6.8 CRAC units

In our simulations, we assumed that there are 3 homogeneous CRAC units. The CoP for each CRAC unit is given by Equation 8. The air flow rate of each CRAC unit is set so that the sum of the air flow rates of the compute nodes is equal to the sum of the flow rates of the CRAC units. The layout of the data center is given in Fig. 1.

7 Simulation Results

7.1 Comparison Overview

One may choose to run all cores in the data center in P-state 0 without considering the power consumption implications. Although this approach is simple and will result in the highest reward rate, it may violate the power constraint and result in a lower reward rate per power consumption. We show this in the next subsection.

We performed simulations for the first-stage assignment problem and compared our technique with a technique that only considers putting a core in P-state 0 or turning off the core. The authors in [36] show how the fraction of the "computational resources" at a compute node can be used to compute the power consumption of a compute node and the QoS obtained from that compute node. Our techniques solve different assignment problems than the technique in [36] and our techniques consider P-state assignments. We adapt the technique in [36] by relating the effective number of cores used at a compute node to the reward rate obtained from that compute node and the total power consumed at that node as described by Equations 21 and 22. We compare our techniques with the one adapted from [36].

The power consumption of compute node j is given by

$$\text{PCN}_j = B_i + \pi_{\text{NT}_j,0} \sum_{i=1}^{T} E(i,j). \quad (21)$$

The reward rate for a task of type i at compute node j is equal to $r_i ECS(i,j,0)E(i,j)$ The comparison technique for solving Problem 1 is given by

$$\text{maximize} \sum_{i=1}^{T} \sum_{j=1}^{\text{NCN}} r_i \text{ECS}(i,j,0) E(i,j). \quad (22)$$

Subject to

1. $\sum_{j=1}^{NCN} ECS(i,j,0)E(i,j) \leq \lambda_i, 1 \leq i \leq T$,
2. $\sum_{i=1}^{T} E(i,j) \leq |\text{cores}_j|, 1 \leq j \leq \text{NCN}$,
3. $\sum_{j=1}^{\text{NCN}} \text{PCN}_j + \sum_{i=1}^{\text{NCRAC}} \text{PCRAC}_i \leq P_{\text{const}}$,
4. $T^{\text{in}} \leq T^{\text{redline}}$.

Constraint 1 guarantees that the execution rate for a task type is not higher than its arrival rate. The effective number of cores used at a compute node must not exceed the total number of cores at that compute node. This is guaranteed by Constraint 2. Constraints 3 and 4 are the power and thermal constraints, respectively. The deadline constraints can be dealt with by setting $E(i,j) = 0$ whenever $d_i < (1/\text{ECS}(i, \text{NT}_j, 0))$ Equation 22 is an NLP problem due to the power consumption of CRAC units. The comparison technique for solving Problem 2 is similar to the comparison technique for solving Problem 1 except that the objective is to minimize power consumption while guaranteeing that the total reward rate does not drop below R_{const}.

7.2 Results

7.2.1 Overview

We have conducted simulations to compare our technique against the one described in Equation 22. We illustrate the effect of the following three parameters on the performance: 1) static power consumption of cores, 2) the variation of the ECS values from being proportional to the clock frequency of cores, and 3) the reward and power constraints. Note that the static power consumption is part of the total power consumption of a *core* and is different than the base power consumption of a *compute node*. The static power consumption is part of the second term in Equation 1. The total power consumption of a compute node is equal to the sum of its base power consumption and the sum of the static and dynamic power consumption of its cores. The results in this section show the percentage increase in the reward rate or the percentage decrease in the power consumption that our technique achieves in comparison to the one described in Equation 22.

7.2.2 Random Starting Points

Because the Step 1 assignments of both Problems 1 and 2 are NLPs, their solutions may be locally optimal. The quality of the locally optimal solution is affected by the starting point of the NLP optimization. To determine an appropriate number of starting points to use, we have conducted 20 simulations, each using 100 randomly generated starting points. For each simulation, we determined the number of random starting points needed to obtain a solution that is within 1% of the best solution. The upper limit of a 95% confidence interval of the number of solutions was 11.45, so we used 11 random starting points for our simulations and one starting point that is the solution of Equation 22.

The Problem in Equation 22 is a NLP due to the power consumption of the CRAC units. Therefore, the solutions to the problems may be locally optimal. A brute force discretized optimization of a problem that has three CRAC units,

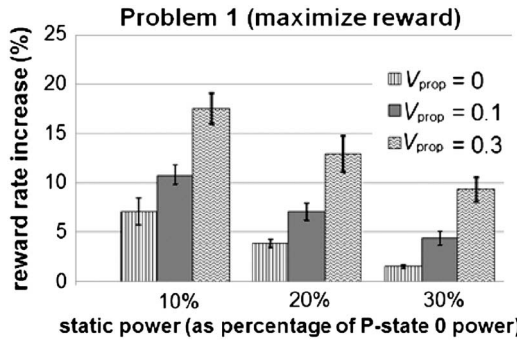


Fig. 6. This figure shows the results for Problem 1 (maximizing reward rate). The average percentage improvement obtained by using the three-step assignment given in Section 5.2 versus the assignment that is based on [36] (given in Equation 22) is shown. A 95% confidence interval is shown for each average percentage improvement. The static power consumption of P-state 0 as a percentage of the total processing core power consumption is increased from 10% to 30%. Each group of columns compares the results when the value of V_{prop} is 0, 0.1, and 0.3. The number of compute nodes, task types, and CRAC units for each simulation is 150, eight, and three, respectively.

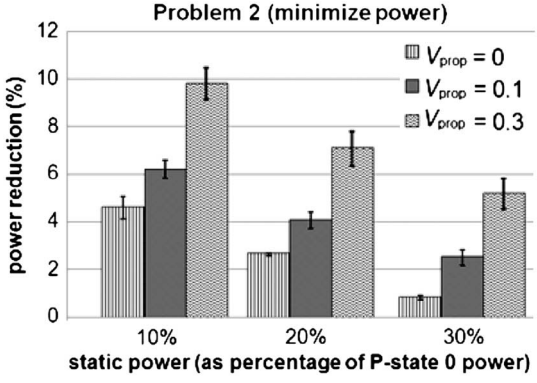


Fig. 7. This figure shows the results for Problem 2 (minimizing power consumption). The average percentage improvement obtained by using the three-step assignment given in Section 5.3 versus the assignment that is based on [36] (given in Equation 22) is shown. A 95% confidence interval is shown for each average percentage improvement. The static power consumption of P-state 0 as a percentage of the total processing power consumption is increased from 10% to 30%. Each group of columns compares the results when the value of V_{prop} is 0,0.1, and 0.3. The number of compute nodes, task types, and CRAC units for each simulation is 150, eight, and three, respectively.

150 compute nodes, and eight task types, is computationally intractable. However, tests on smaller problems, i.e., two CRAC units, 40 compute nodes, and eight task types, have shown no improvement. Therefore, we only use a single starting point to find the solution to Equation 22.

7.2.3 Main Results for Problems 1 and 2

Figs. 6 and 7 show the percentage increase in the reward rate for Problem 1 and the percentage reduction in power consumption for Problem 2 that our technique achieves. Each bar in Figs. 6 and 7 represents the average of 20 simulations. Error bars are added to show a 95% confidence interval around the average.

Both figures show that as the static power consumption of P-state 0 increases, the relative performance of our technique decreases. Because P-state 0 runs at a higher voltage and frequency, the percentage of dynamic power consumption is usually higher than that of the other P-states. Therefore, the static power as a percentage of the overall power consumption for the other P-states will be higher compared to that of P-state 0. The static power consumption is not related to the frequency, so the higher P-states will have a lower performance (in terms of clock frequency) to power consumption ratio compared to P-state 0. When the performance to power consumption ratio of P-state 0 is the highest among all the P-states, the assignment technique of Equation 22 will perform as well as our technique. The static power percentages for all the P-states of each node type are shown in Table 2. The static power consumption of the P-states in each node type is calculated using the static power of P-state 0. For our simulations, we assumed three different static power consumption percentages for P-state 0 (10, 20, and 30%). Refer to Appendix A for details about the calculation of the static power.

TABLE 2
Static Power Consumption of P-States of Cores in Each Node Type as a Percentage

	Node type 1 (P-state 1, 2, 3)	Node type 2 (P-state 1, 2, 3)
P-state 0 static power =10%	12.3, 15.6, 31.0%	12.5, 16.6, 27.5%
P-state 0 static power =20%	24.0, 29.3, 50.2%	24.4, 31.0, 46.0%
P-state 0 static power =30%	35.1, 41.5, 63.4%	35.6, 43.5, 59.4%

Figs. 6 and 7 also show that the relative performance of our technique increases as V_{prop} increases from 0 to 0.3. For a given core type, a higher value of V_{prop} gives a higher affinity of P-states to task types (i.e., some P-states will be better suited for specific task types than others). Therefore, more reward rate per power consumption can be obtained by matching task types with their better suited P-states.

The reason that our technique achieves higher increase in reward rate for Problem 1 compared to the decrease in power consumption for Problem 2 is that there is a lower bound on the minimum power.

The lower bound occurs when all cores are turned off and all compute nodes are only consuming the base power. However, the minimum reward rate of the data center is zero, which also happens when all cores are turned off. Because the minimum power is greater than zero, Problem 2 leaves less opportunity for improvement than Problem 1. If we do not consider the minimum power consumption of the data center for both our technique and the technique in Equation 22, then percentage power reduction that our technique achieves over the technique in Equation 22 will be on average 1.58 times higher on average.

The simple approach of running all cores in the data center at P-state 0 will result in a violation of the power constraint for Problem 1 and higher power consumption for Problem 2. For example, when P-state 0 static power is 10% of its total power consumption and V_{prop} is 0.1, the simple approach resulted in a violation of the power constraint by 42% for Problem 1 and a power consumption of 75% higher than our approach for Problem 2.

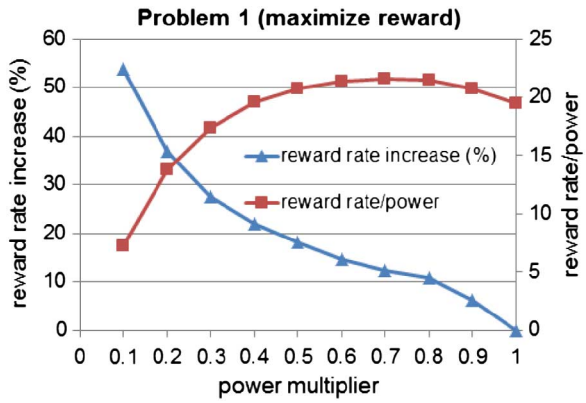


Fig. 8. This figure shows the percentage increase in reward rate that our approach achieves over the technique in Equation 22 and the reward rate per power consumption for our technique for Problem 1 (maximizing reward rate). The power multiplier is increased from 0.1 to 1 with a step of 0.1. The static power of P-state 0 is 10% and V_{prop} is 0.3. Each point in the figure represents a simulation case for one data center. The number of compute nodes, task types, and CRAC units for each simulation is 150, eight, and three, respectively.

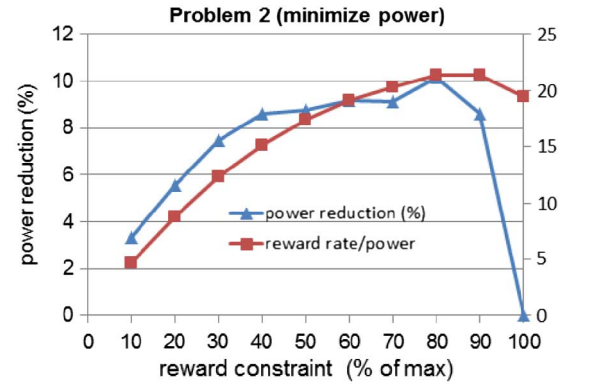


Fig. 9. This figure shows the percentage reduction in power consumption that our approach achieves over the technique in Equation 22 and the reward rate per power consumption for our technique for Problem 2 (minimizing power). The reward rate as a percentage of the maximum possible reward rate is increases from 10% to 100%. The static power of P-state 0 is 10% and V_{prop} is 0.3. Each point in the figure represents a simulation case for one data center. The number of compute nodes, task types, and CRAC units for each simulation is 150, eight, and three, respectively.

7.2.4 Effect of Power and Reward Constraints

We also have conducted simulations to show the effect of increasing the power and reward constraints on the performance of our techniques. For these simulations, the static power consumption of P-state 0 was set to 10% of its total power consumption and V_{prop} was set to 0.3. These simulations are shown in Figs. 8 and 9.

As the power constraint gets tighter (i.e., the power multiplier value gets lower) the relative performance of our technique improves. This is because when power is scarce, managing it intelligently can lead to substantial performance gains. As the power constraint gets looser, our technique will start assigning lower P-states to take advantage of the available power. Therefore, the performance of our technique will be closer to the performance of the technique in Equation 22 until they are both equal when the power multiplier is equal to 1.

As shown in Fig. 9, when the reward constraint is low, the relative performance of our technique is low. This is because there is a lower bound on the minimum power consumption of the data center. However, as the reward constraint increases, the power needed to satisfy the reward rate constraint becomes higher and more power savings can be obtained using our technique. Even when the reward rate constraint is at 90% of the maximum possible, our technique achieves 8.6% improvement. When the reward rate constraint is 100% of the maximum possible, both our technique and the one in Equation 22 will run all cores in the data center at P-state 0 to satisfy the constraint. Therefore, our approach will have no reward rate improvements.

Figs. 8 and 9 also show the reward rate per power consumption ratio for different power and reward rate constraints. When the power multiplier is low (i.e., tighter power constraint) or the reward rate multiplier is low (i.e., looser reward rate constraint) solutions to Problem 1 and Problem 2 are both driven to consume less power and collect less reward. Because of the minimum power consumption of the data center is not zero, the ratio of the reward rate to power consumption *decreases* more with the reduction in reward rate than it does *increase* with the reduction in total power consumption. This explains the low reward rate to power consumption ratio when the power multiplier is low or the reward rate multiplier is low.

There are two reasons that cause the ratio of reward rate to power consumption to decline for a higher value of a power constraint for Problem 1 or a higher value of a reward rate constraint for Problem 2. The first reason is that our technique will run the cores at lower P-states that are not power efficient so that all the power that is available is used (in Problem 1) or the reward rate constraint is satisfied (in Problem 2). The second reason is that our technique will assign tasks of types that have low reward rates because all the higher reward rate tasks are fully assigned. The simple approach of running all cores in the data center at P-state 0 will be equivalent to the case where the reward rate constraint is 100% of the maximum and the case where the power multiplier is 1. The simple approach will result in a reward rate per power consumption ratio equal to 19.43 that is less than the highest ratio which is equal to 21.6 for Problem 1 (Fig. 8) and 21.4 for Problem 2 (Fig. 9).

7.2.5 Effect of Reward

In all the results discussed previously, the reward of a task type was assumed to be equal to the reciprocal of its easiness (see Equation 12). We have conducted two sets of simulations for Problem 1 to show the effect of different task type reward values on the performance results of our technique. The following are the common parameters between both sets of simulations (which are identical to those from Fig. 6):

1. The number of task types, CRAC units, and compute nodes is eight, three, and 150, respectively.
2. The static power percentage for P-state 0 is 10%.
3. The power multiplier is 0.5.
4. Each set of simulations has 20 cases.

In the first set of simulations, we calculated eight reward values using Equation 12. The reward value of a task type was assigned randomly with no replacement from the set of eight reward values. The average increase in reward rate that our technique achieved compared to the one in Equation 22 was

TABLE 3
Execution Times for Step 1 of Problem 1 as the Number of Compute Nodes Varies

number of nodes	number of task types	average execution time (seconds)
30	8	42
90	8	679
150	8	2771
210	8	3688
270	8	9575

Table 4
Execution Times for Step 1 of Problem 1 as the Number of Compute Nodes and the Number of Task Types Vary

number of nodes	number of task types	average execution time (seconds)
30	2	10
90	5	181
150	8	2771
210	11	12242
270	14	134842

13%. This reward rate increase is less than the reward rate increase for the same static power (10%) and V_{prop} (0.3) in Fig. 6 (in that case it was 18.5%). The reason for this lower reward rate increase is that when we assign reward values randomly, we will have some tasks that have faster average execution rates and have more reward. Both heuristics (ours and the one based on Equation 22) execute the easier tasks that have the most reward. Even though our heuristic executes more tasks than the one based on Equation 22, the difference in total reward rate is not as large because the extra tasks are more difficult and have a smaller reward rate. This effect was even more pronounced in our second set of simulations were we assigned task types a reward value that was equal to its easiness (i.e., easier tasks will have more reward).

7.2.6 Scalability Analysis

In our approach, the step that consumes the most time is Step 1. We conducted a scalability analysis for the execution time of Step 1 for Problem 1 (the execution time for Problem 2 was similar to Problem 1). Table 3 shows the execution times in seconds for different numbers of compute nodes. The number of task types remains fixed at eight task types. In Table 4, we have increased the number of task types as the number of compute nodes was increased. The execution times for each case in both Tables 3 and 4 are averaged across five simulation runs. All the simulations were run on a laptop computer with a Core i7 processor running at a clock frequency of 2.8 GHz. The number of CRAC units was fixed at three CRAC units.

Tables 3 and 4 show that the execution time is sensitive to both the number of task types and the number of compute nodes. Recall that the task type arrival rate is a function of the number and type of compute nodes (which determines the total number of cores) and the number of task types (T), as shown in Equation 16. As T increases, the arrival rate of each task type decreases (Equation 16). However, the total workload remains relatively constant.

Because the calculation of Step 1 is done offline (i.e., the assignment decisions are made before tasks arrive) based on estimated task arrival rates obtained from historical data for each task type, it is feasible to execute it for a longer time compared to online techniques. Furthermore, if Step 1 is parallelized (for example, by calculating the solutions to multiple starting points in parallel), then the time consumed by it can be significantly reduced.

7.2.7 Unexecuted Workload

Because of the power constraint in Problem 1 and the reduction of power consumption in Problem 2, a portion of the workload will not be executed. The task types that are not executed are the ones with low reward per power consumption ratio. In many cases, these task types were the same for our technique and the technique in Equation 22. However, there are some cases were this was not the case. This is because our technique considers higher P-states that may have a better reward per power consumption for different task type than the technique in Equation 22.

Both our technique and the technique in Equation 22 execute more tasks for Problem 1 compared to Problem 2. This is because in Problem 1 the goal is to execute as many tasks as possible to maximize the reward. However, in Problem 2 the goal is to minimize the power consumption which will result in less tasks being executed because we are not concerned with collecting reward at a rate higher than the reward rate constraint.

7.2.8 Summary

Our results show an average improvement over the comparison technique of up to 17% for Problem 1 (maximizing reward) and up to 9% for Problem 2 (minimizing power). Higher percentage increase in reward rate can be achieved for data centers with tighter power constraints. Because today's data centers are large, these improvements can mean hundreds of thousands of dollars in additional revenue or power savings. For example, the average cost of electricity in the U.S. for the industrial sector is $\$0.0688/\text{kWh}$ [16]. If we achieve 9% power savings in a data center that has an average power consumption of 5 MW, then that will result in about $\$271,395$ in annual savings.

8 Conclusion

In this paper, we study two assignment problems. The first problem maximizes the reward collected for completing tasks by their deadlines with a constraint on the maximum total power consumption. The second problem minimizes power consumption with a constraint on the minimum reward rate. We show how the P-states can be assigned at the data center level and divide each assignment problem into two stages. The first stage assigns the P-states of cores, the desired number of tasks per unit time allocated to a core, and the outlet CRAC temperatures. The second stage assigns individual tasks as they arrive at the data center to cores so that the actual number of tasks per unit time allocated to a core approaches the desired number set by the first stage.

We formulate the first-stage assignment as a MINLP. Because the MINLP is not scalable with respect to the number of cores, we propose a multi-step, scalable assignment technique. At the second stage, we propose a dynamic scheduler to assign tasks entering the data center to cores.

In many data centers where static and dynamic core power consumptions are considered, P-state 0 is not the P-state with the highest performance to power consumption ratio. Therefore, using the assignment techniques in this paper will result in a better total reward over a technique that chooses between putting a core in P-state 0 or turning it off.

We conducted simulations to show the effectiveness of our technique over a technique based on [36] which did not consider multiple P-states. In some cases, our technique achieved 17% average improvement for the problem of maximizing reward and 9% average improvement for the problem of minimizing the power consumption. In a large data center, these improvements can mean hundreds of thousands of dollars in additional revenue from additional productivity (Problem 1) or power savings (Problem 2).

In our work, we assume that there is always enough memory to run all the tasks assigned to the cores in a specific compute node. One way this work can be extended is to take into account memory limitations.

Acknowledgments

The authors thank Mark Oxley and Ryan Friese for their valuable comments on this work. This research was supported by the NSF under Grant CNS-0905399, and by the Colorado State University George T. Abell Endowment. A preliminary version of portions of this material appeared in the Heterogeneity in Computing Workshop 2012.

References

[1] *AMD Family 10h Server and Workstation Processor Power and Thermal Data Sheet*. Publication # 43374, Revision 3.19, June 2010.

[2] Z. Abbasi, G. Varsamopoulos, and E.K.S. Gupta, "Thermal Aware Server Provisioning and Workload Distribution for Internet Data Centers," *Proc. 19th ACM Int'l Symp. High Performance Distributed Computing (HPDC'10)*, pp. 130-141, 2010.

[3] S. Ali, T.D. Braun, H.J. Siegel, A.A. Maciejewski, N. Beck, L. Bölöni, M. Maheswaran, A.I. Reuther, J.P. Robertson, M.D. Theys, and B. Yao, "Characterizing Resource Allocation Heuristics for Heterogeneous Computing Systems," *Advances in Computers, Parallel, Distributed, and Pervasive Computing*, vol. 63, pp. 91-128, 2005.

[4] A. M. Al-Qawasmeh, A. A. Maciejewski, and H. J. Siegel, "Characterizing Heterogeneous Computing Envinronments Using Singular Value Decomposition," *Proc. 19th Heterogeneity in Computing Workshop (HCW 2010), 24th Int'l Parallel and Distributed Processing Symp., Workshops and PhD Forum (IPDPSW 2010)*, pp. 1-9, Apr. 2010.

[5] A.M. Al-Qawasmeh, A.A. Maciejewski, and H.J. Siegel, "Characterizing Task-Machine Affinity in Heterogeneous Computing Environments," *Proc. 20th Heterogeneity in Computing Workshop (HCW 2011), 25th Int'l Parallel and Distributed Processing Symp., Workshops and PhD Forum (IPDPSW 2011)*, pp. 34-44, Apr. 2011.

[6] A.M. Al-Qawasmeh, A.A. Maciejewski, H. Wang, J. Smith, H.J. Siegel, and J. Potter, "Statistical Measures for Quantifying Task and Machine Heterogeneities," *J. Supercomputing, Special Issue on Advances in Parallel and Distributed Computing*, vol. 57, no. 1, pp. 34-50, July 2011.

[7] J. Apodaca, D. Young, L. Briceno, J. Smith, S. Pasricha, A.A. Maciejewski, H.J. Siegel, S. Bahirat, B. Khemka, A. Ramirez, and Y. Zou, "Stochastically Robust Static Resource Allocation for Energy Minimization with a Makespan Constraint in a Heterogeneous Computing Environment," *Proc. 9th ACS/IEEE Int'l Conf. Computer Systems and Applications (AICCSA'11)*, p. 10, Dec. 2011.

[8] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems," *Proc. 22nd IEEE Real-Time Systems Symp. (RTSS'01)*, pp. 95-105, Dec. 2001.

[9] H. Barada, S. M. Sait, and N. Baig, "Task Matching and Scheduling in Heterogeneous Systems Using Simulated Evolution," *Proc. 10th Heterogeneous Computing Workshop (HCW 2001), 15th IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS 2001)*, pp. 875-882, Apr. 2001.

[10] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755-768, May 2012.

[11] D. Brown and C. Reams, "Toward Energy-Efficient Computing," *Communications of the ACM*, vol. 53, no. 3, p. 14, Mar. 2010.

[12] J.A. Butts and G.S. Sohi, "A Static Power Model for Architects," *Proc. 33rd Ann. ACM/IEEE Int'l Symp. Microarchitecture (MICRO 33)*, pp. 191-201, Dec. 2000.

[13] K. W. Cameron, R. Ge, and X. Feng, "High-Performance, Power-Aware Distributed Computing for Scientific Applications," *Computer*, vol. 26, no. 11, pp. 40-47, Nov. 2005.

[14] J. Choi, S. Govindan, J. Jeong, B. Urgaonkar, and A. Sivasubramaniam, "Power Consumption Prediction and Power-Aware Packing in Consolidated Environments," *IEEE Trans. Computers*, vol. 59, no. 12, pp. 1640-1654, Dec. 2010.

[15] M.K. Dhodhi, I. Ahmad, and A. Yatama, "An Integrated Technique for Task Matching and Scheduling onto Distributed Heterogeneous Computing Systems," *Parallel and Distributed Computing*, vol. 62, no. 9, pp. 1338-1361, Sep. 2002.

[16] Energy Information Administration. http://www.eia.gov/electricity/data.cfm, last accessed, 2012.

[17] E. N. Elnozahy, M. Kistler, and R. Rajamony, "Energy-Efficient Server Clusters," *Proc. Second Int'l Workshop Power-Aware Computer Systems*, pp. 179-197, 2002.

[18] Environmental Protection Agency. *Report to Congress on Server and Data Center Energy Efficiency*, http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf, Last accessed, 2011.

[19] D. Filani, J. He, S. Gao, M. Rajappa, A. Kumar, P. Shah, and R. Nagappan, "Dynamic Data Center Power Management: Trends, Issues, and Solutions," *Intel Technology J.*, vol. 12, no. 1, pp. 59-67, 2008.

[20] R.F. Freund and H.J. Siegel, "Heterogeneous Processing," *Computer*, vol. 26, pp. 13-17, June 1993.

[21] S.K. Grag, C.S. Yeo, A. Anadsivam, and R. Buyya, "Environment-Conscious Scheduling of HPC Applications on Distributed Cloud-Oriented Data Centers," *Parallel and Distributed Computing*, vol. 71, no. 6, pp. 732-749, June 2011.

[22] A. Ghafoor and J. Yang, "A Distributed Heterogeneous Supercomputing Management System," *Computer*, vol. 26, no. 6, pp. 78-86, June 1993.

[23] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., and Toshiba Corporation Std. (Apr. 2010). *Advanced Configuration and Power Interface Specification*, Rev. 4.0a, http://www.acpi.info/DOWNLOADS/ACPIspec40a.pdf, last accessed, 2012.

[24] J.-W. Jang, M. Jeon, H.-S. Kim, H. Jo, J.-S. Kim, and S. Maeng, "Energy Reductions in Consolidated Servers through Memory-Aware Virtual Machine Scheduling," *IEEE Trans. Computers*, vol. 60, no. 4, Apr. 2011.

[25] M. Kafil and I. Ahmad, "Optimal Task Assignment in Heterogeneous Distributed Computing Systems," *IEEE Concurrency*, vol. 6, no. 3, pp. 42-51, July 1998.

[26] X. Kang, H. Zhang, G. Jiang, H. Chen, X. Meng, and K. Yoshihira, "Understanding Internet Video Sharing Site Workload: A View from Data Center Design," *J. Visual Comm. Image Representation*, vol. 21, no. 2, pp. 129-138, Feb. 2010.

[27] A. Khokhar, V.K. Prasanna, M.E. Shaaban, and C. Wang, "Heterogeneous Computing: Challenges and Opportunities," *Computer*, vol. 26, no. 6, pp. 18-27, June 1993.

[28] J.-K. Kim, H.J. Siegel, A.A. Maciejewski, and R. Eigenmann, "Dynamic Resource Management in Energy Constrained Heterogeneous Computing Systems Using Voltage Scalling," *IEEE Trans. Parallel and Distributed Systems, Special Issue on Power-Aware Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1445-1457, Nov. 2008.

[29] Y. Lin and L. He, "Dual-VDD Interconnect with Chip-Level Time Slack Allocation for FPGA Power Reduction," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2023-2034, Oct. 2006.

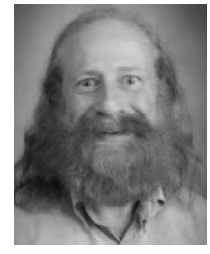
[30] J.R. Lorch and A.J. Smith, "Improving Voltage Scaling Algorithms With PACE," *ACM SIGMETRICS Performance Evaluation Rev.*, vol. 29, no. 1, pp. 50-61, June 2001.

[31] M. Maheswaran, T.D. Braun, and H.J. Siegel, "Heterogeneous Distributed Computing," *Encyclopedia of Electrical and Electronics Engineering*. New York: John Wiley & Sons, vol. 8, pp. 679-690, 1999.
[32] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making Scheduling 'Cool': Temperature-Aware Workload Placement in Data Centers," *Proc. USENIX Ann. Technical Conf. (ATEC'05)*, 10 pp., 2005.
[33] T. Mukherjee, G. Varsamopoulos, S.K.S. Gupta, and S. Rungta, "Measurement-Based Power Profiling of Data Center Equipment," *Proc. IEEE Int'l Conf. Cluster Computing*, pp. 476-477, Sep. 2007.
[34] E. Pakbaznia, M. Ghasemazar, and M. Pedram, "Temperature-Aware Dynamic Resource Provisioning in a Power-Optimized Datacenter," *Proc. Conf. Design, Automation and Test in Europe 2010*, pp. 124-129, 2010.
[35] V. Pallipadi and A. Starikovsky, "The On-Demand Governor," *Proc. 2006 Linux Symp.*, pp. 215-229, 2006.
[36] L. Parolini, N. Tolia, B. Sinopoli, and B. H. Krogh, "A Cyber-Physical Systems Approach to Energy Management in Data Centers," *Proc. 1st ACM/IEEE Int'l Conf. Cyber-Physical Systems*, pp. 168-177, 2010.
[37] H. Singh and A. Youssef, "Mapping and Scheduling Heterogeneous Task Graphs using Genetic Algorithms," *Proc. 5th IEEE Heterogeneous Computing Workshop (HCW'96)*, pp. 86-97, 1996.
[38] Standard Performance Evaluation Corporation (SPEC). *SPECpower_ssj2008*, http://www.spec.org/power_ssj2008, last accessed, 2011.
[39] Q. Tang, T. Mukherjee, S.K.S. Gupta, and P. Cayton, "Sensor-Based Fast Thermal Evaluation Model for Energy Efficient High-Performance Datacenters," *Proc. 4th Int'l Conf. Intelligent Sensing and Information Processing (ICISIP 2006)*, pp. 203-208, Dec. 2006.
[40] N. Tolia, Z. Wang, P. Ranganathan, C. Bash, and M. Marwah, "Unified Thermal and Power Management in Server Enclosures," *Proc. ASME/Pacific Rim Technical Conf. Exhibition on Packaging and Integration of Electronic and Photonic Systems, MEMS, and NEMS (InterPACK)*, 10 pp., July 2009.
[41] C. Xian, Y.-H. Lu, and Z. Li, "Dynamic Voltage Scaling for Multitasking Real-Time Systems with Uncertain Execution Time," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 8, pp. 1467-1478, Aug. 2008.
[42] D. Xu, K. Nahrstedt, and D. Wichadakul, "QoS and Contention-Aware Multi-Resource Reservation," *Cluster Computing*, vol. 4, no. 2, pp. 95-107, Apr. 2001.
[43] J. Yang, I. Ahmad, and A. Ghafoor, "Estimation of Execution Times on Heterogeneous Supercomputer Architectures," *Proc. Int'l Conf. Parallel Processing*, vol. 1, pp. 219-225, Aug. 1993.
[44] J. Yang, A. Khokhar, S. Sheikh, and A. Ghafoor, "Estimating Execution Time for Parallel Tasks in Heterogeneous Processing (HP) Environment," *Proc. Heterogeneous Computing Workshop*, pp. 23-28, Apr. 1994.
[45] B.D. Young, J. Apodaca, L.D. Briceño, J. Smith, S. Pasricha, A.A. Maciejewski, H.J. Siegel, B. Khemka, S. Bahirat, A. Ramirez, and Y. Zou, "Deadline and Energy Constrained Dynamic Resource Allocation in a Heterogeneous Computing Environment," *J. Supercomputing*, vol. 63, no. 2, pp. 326-347, Feb. 2013.
[46] H. Yu, B. Veeravalli, and Y. Ha, "Dynamic Scheduling of Imprecise-Computation Tasks in Maximizing QoS under Energy Constraints for Embedded Systems," *Proc. 2008 Asia and South Pacific Design Automation Conf. (ASPDAC'08)*, pp. 452-455, Mar. 2008.

Abdulla M. Al-Qawasmeh received the BS degree in computer science and information systems from Jordan University of Science and Technology, Irbid, Jordan, in 2005, the MS degree in computer science from the University of Houston Clear Lake, in 2008, and the PhD degree in computer engineering from Colorado State University. He currently holds the position of senior software developer at Service Logix and Lead Software Developer at step2compliance. His work is focused on Web and cloud-based applications. His research interests include robust, power, and energy-efficient scheduling techniques in heterogeneous computing systems and data centers and the characterization of heterogeneous computing systems.

Sudeep Pasricha (M'02) received the BE degree in electronics and communication engineering from Delhi Institute of Technology, Delhi, India, in 2000, and the MS and the PhD degrees in computer science from the University of California, Irvine, in 2005 and 2008, respectively. He is currently an assistant professor of Electrical and Computer Engineering at Colorado State University, Fort Collins, with a joint appointment in the Department of Computer Science. His research interests are in the areas of energy efficiency and fault tolerant design for high performance computing, embedded systems, and mobile computing. He is currently an advisory board member of ACM SIGDA, information director of the *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, editor of the ACM SIGDA E-news, organizing committee member of the NOCS, RTCSA, AICCSA, and ICECS conferences and ICCAD CADathalon, and technical program committee member of the DAC, DATE, CODES+ISSS, ISQED, NOCS, NoCArc, VLSI Design, and GLSVLSI conferences and the SIGDA DAC PhD Forum. He holds an affiliate faculty member position at the Center for Embedded Computer Systems, University of California, Irvine. He was the recipient of the AFOSR Young Investigator Award in 2012, and Best Paper Awards at the IEEE AICCSA 2011, IEEE ISQED 2010, and ACM/IEEE ASPDAC 2006 conferences.

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 at Purdue University, West Lafayette. He is currently a professor and head of the Department of Electrical and Computer Engineering at Colorado State University. He is a fellow of the IEEE, with research interests that include robotics and high performance computing. His complete vita is available at: http://www.engr.colostate.edu/~aam.



Howard Jay Siegel received the BS degree from the Massachusetts Institute of Technology (MIT), and the MA, MSE, and PhD degrees from Princeton University. He was appointed the Abell Endowed chair distinguished professor of Electrical and Computer Engineering at Colorado State University (CSU) in 2001, where he is also a professor of computer science. From 1976 to 2001, he was a professor at Purdue University. He is a fellow of the IEEE and a fellow of the ACM. He has coauthored over 400 technical papers. His research interests include robust computing systems, resource allocation in computing systems, heterogeneous parallel and distributed computing and communications, parallel algorithms, and parallel machine interconnection networks. He was a co-editor-in-chief of the *Journal of Parallel and Distributed Computing*, and was on the Editorial Boards of both the *IEEE Transactions on Parallel and Distributed Systems* and the *IEEE Transactions on Computers*. He has been an international keynote speaker and tutorial lecturer and has consulted for industry and government. Homepage: www.engr.colostate.edu/~hj.