Thermal-Aware Performance Optimization in Power Constrained Heterogeneous Data Centers

Abdulla M. Al-Qawasmeh[1], Sudeep Pasricha[1], Anthony A. Maciejewski[1], and Howard Jay Siegel[1,2]
[1]Department of Electrical and Computer Engineering
[2]Department of Computer Science
Colorado State University
Fort Collins, Colorado, USA
e-mails: {Abdulla.Al-Qawasmeh, Sudeep, aam, hj}@colostate.edu

***Abstract*—The power consumption of data centers has been increasing at a rapid rate over the past few years. Many of these data centers experience physical limitations on the power needed to run the data center. This paper attempts to maximize the performance of a data center that is subject to total power consumption and thermal constraints. We consider a power model for a data center that includes power consumed in both Computer Room Air Conditioning (CRAC) units and compute nodes. Our approach quantifies the performance of the data center as the total reward collected from completing tasks in a workload by their individual deadlines. We develop novel optimization techniques for assigning the performance states of compute cores at the data center level to increase the performance of the data center. The assignment problem in this paper is thermal aware as it considers the temperature evolution effects of performance state assignments, which in turn affects the power consumed by the CRAC units. Our simulation studies show that in some cases the assignment technique used in this paper achieves about 10% average improvement in the performance of a data center over an assignment problem that only considers putting a compute core in the performance state with the highest performance or turning the core off.**

Keywords-Thermal-Aware; Performance States; Data Center; CRAC; heterogeneous computing

I. INTRODUCTION

Over the last few years, the power consumption of data centers has been increasing at a rapid rate. In a report by the EPA [14], 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 bg illion 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, 20, 22, 26, 30, 33]).

In some cases, there are physical limitations on the amount of power that is 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 [10]. In a survey of data centers [15], 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, while guaranteeing that the thermal constraints are not violated, can lead to significant overall power savings.

This paper attempts to maximize the performance of a data center that is subject to a total power consumption constraint ($\underline{P}_{const}$) 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.

Performance states (P-states) of cores within a processor provide a tradeoff between the power consumed by a core and its performance [17]. 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 many cases, the lowest P-state (P0) is not the P-state with the best tradeoff between performance and power consumption [21, 31].

P-state assignments in data centers are mainly done at the compute node level. The P-state of one or more cores is increased when the node's utilization drops below a specified threshold (e.g., [13, 25, 30]). However, for a power constrained data center, the utilization is often close to 100% because the data center is often oversubscribed (i.e., the power needed to execute all tasks is more than the power available). As opposed to most of the previous work, in this paper, we assign the P-states by considering the whole data center and show how our technique increases the performance 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. We first show how the assignment problem in a data center can be expressed as an exact optimization problem. The P-state assignment part of the problem introduces integer constraints. The integer constraints make the assignment problem not scalable with

978-0-7695-4676-6/12 $26.00 © 2012 IEEE
DOI 10.1109/IPDPSW.2012.19

respect to the number of cores in the data center. A simple relaxation of the integer constraints will introduce additional binary and nonlinear constraints that make the assignment problem not scalable. Therefore, we propose a novel and scalable assignment technique that involves solving a set of scalable optimization problems. We compare our technique to a technique based on [26]. That technique only considers putting a core in P-state 0 or turning it off. We show that using our technique results in notable performance improvements.

In summary, we make the following novel contributions. Our first contribution is that we consider a power constrained data center. With the power constraint, the data center becomes oversubscribed. Therefore, power optimization using P-state assignment at the individual server level (as done in many previous works) will not be effective and a holistic data center level P-state assignment approach is needed. The second contribution is that we express the assignment problem at the data center level as an optimization problem. The decisions of this optimization problem are: the P-states of cores, the desired rate of executing tasks on cores, and the outlet CRAC temperatures. The optimization problem is not scalable due to integer constraints imposed by the P-states. We show that a simple relaxation of the integer constraints makes the problem not scalable. So the third contribution is that we propose a scalable assignment technique. This technique consists of three stages. The first stage finds the CRAC outlet temperatures and the desired power consumption of each core. The second stage converts the desired power consumption at a core into a P-state. The third stage finds the desired rate of executing task types on cores. The forth contribution is that we propose a dynamic scheduler that schedules tasks as they come into the data center to cores such that the actual rate of executing task types on cores is as close as possible to the desired rate. Finally, the fifth contribution of this research is that we conduct simulations that show the performance gains of applying our technique in a power constrained data center.

The rest of this paper is organized as follows. Section II discusses related work. The data center model is described in Section III. In Section IV, the thermal constraints in the data center are given. The assignment problem and our solution to the problem are given in Section V. Section VI describes the simulation set up. Simulation results are shown in Section VII. In Section VIII, we list a number of future directions for this work. Conclusions are given in Section IX.

II. Related Work

In [30], 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 such that the utilization is never higher than 80%. However, as discussed in the introduction, this technique is not effective in a power constrained data center because the utilization will 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 [26], 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. In this paper, we extend that work in two directions. First, we consider a power constraint on the data center operation. 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, 31, 33, 34]). 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 compute nodes' power consumption.

There are many papers that deal with the thermal-aware scheduling problem (e.g., [2, 22, 24]). 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 480V 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.

III. Data Center Model

A. Overview

Data centers are usually arranged in a hot-aisle/cold-aisle configuration. This is depicted in Figure 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 aisle. The power consumption at compute nodes causes the temperature of the air going through the node to rise. The hot air exits a compute node 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. This model is based on one used in the literature (e.g., [2, 24, 29]) to model real world systems (e.g., the data center that is managed by the Sustainable Ecosystems Research Group at HP in Fort Collins, CO). In this section, we give a detailed description of the workload and the different components of the data center.

B. Workload

We assume that we have a set of known $\underline{T}$ task types. The arrival rate of tasks of type i is given by $\underline{\lambda}_i$. A reward $\underline{r}_i$ is collected for completing a task of type i by its individual deadline. The deadline of a task of type i is given by: deadline = arrival time + m_i. The value of m_i would be specified by the user. In addition, we assume tasks can be

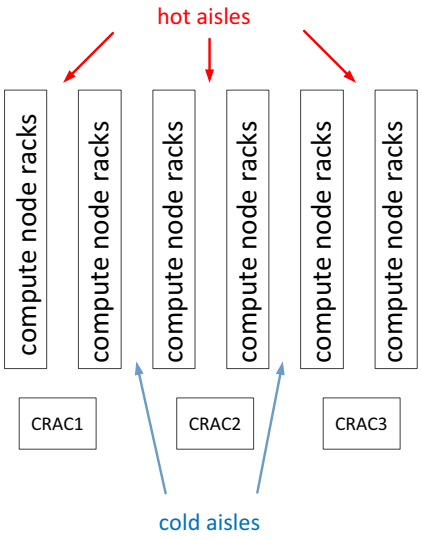

Figure 1. A hot-aisle/cold-aisle data center layout.

dropped if their deadlines cannot be satisfied. The goal of this paper is to maximize the total reward that is collected given a constraint on the total power consumption (the power consumption of compute nodes and CRAC units) of the data center.

C. Compute Nodes

Let the number of compute nodes in the data center be NCN. Each compute node has a number of identical cores. Further, 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 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 nodes 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 its cores' power consumption. The base power consumption is used to model non-compute devices (e.g., disks, fans). Because we are considering an oversubscribed system, we are not considering the case where compute nodes can be turned off (only individual cores can be turned off) so the non-compute devices will consume power regardless of whether the compute node is executing tasks or not. Further, the power consumed by the non-compute devices is not affected by the utilization of the compute cores [23]. Let B_j be the base power consumption of a compute node of type j.

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 the core can be turned off. 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-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 [23]. 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 (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.

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:

$$PCN_j = B_{NT_j} + \sum_{k \in cores_j} \pi_{NT_j,PS_k} . \quad (1)$$

The first term refers to the baseline power and the second term refers to the active operational power that depends on the P-state.

D. Estimated Computational Speed

We assume that the estimated computational speed (ECS) of a task of type i on a compute 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., [6, 9, 12, 16, 18, 19, 27, 32]). The ETC values for a given system can be obtained from user supplied information, experimental data, or task profiling and analytical benchmarking [3, 16, 19, 32]. Obviously, when the core is turned off, the ECS of a task of any type is 0, i.e., ECS(i,j, η_j−1) = 0 for all i and j.

E. Computer Room AC Units

We assume that the number of CRAC units in a data center is NCRAC. These 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 [22].

The amount of heat removed by a CRAC unit i depends on the inlet air temperature, $TCRAC_i^{in}$ (which is directly affected by the heat generated by compute nodes), and the assigned outlet air temperature, $TCRAC_i^{out}$ (which is the temperature of the cool air to be generated by the CRAC). Let ρ be the density of air. Let Cp be the specific heat capacity of air. Let $FCRAC_i$ be the air flow rate at CRAC unit i. Then the amount of heat removed at CRAC unit i is equal to [29]:

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

The CoP of a CRAC unit is a function of its outlet temperature [22]. The power consumed by CRAC unit i, $\underline{\mathrm{PCRAC}_i}$, is given by [22]

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

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.

IV. 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 $\underline{\mathrm{TCN}_i^{\mathrm{in}}}$ and $\underline{\mathrm{TCN}_i^{\mathrm{out}}}$ be the inlet and outlet air temperatures at compute node i, respectively. Let $\underline{\mathrm{FCN}_i}$ be the air flow rate at compute node i. The outlet air temperature of compute node i is given by [29]

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

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 [29]. Let

$\boldsymbol{T}^{\mathrm{out}} = \left[\mathrm{TCRAC}_1^{\mathrm{out}}, \ldots, \mathrm{TCRAC}_{\mathrm{NCRAC}}^{\mathrm{out}}, \mathrm{TCN}_1^{\mathrm{out}}, \ldots, \mathrm{TCN}_{\mathrm{NCN}}^{\mathrm{out}}\right]^{\mathrm{T}}$.

Let

$\boldsymbol{T}^{\mathrm{in}} = \left[\mathrm{TCRAC}_1^{\mathrm{in}}, \ldots, \mathrm{TCRAC}_{\mathrm{NCRAC}}^{\mathrm{in}}, \mathrm{TCN}_1^{\mathrm{in}}, \ldots, \mathrm{TCN}_{\mathrm{NCN}}^{\mathrm{in}}\right]^{\mathrm{T}}$.

Using the Abstract Heat Flow Model proposed in [29], we can compute each element of $\boldsymbol{T}^{\mathrm{in}}$ as a linear combination of the elements of $\boldsymbol{T}^{\mathrm{out}}$, i.e.,

$$\boldsymbol{T}^{\mathrm{in}} = \mathrm{A}\boldsymbol{T}^{\mathrm{out}}. \quad (5)$$

The values in matrix A can be estimated using sensor measurements [29]. Let $\boldsymbol{T}^{\mathrm{redline}}$ be the vector of redline temperature constraints on inlet air temperatures– i.e.,

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

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

V. Assignment Problem

A. Overview

Given a workload comprised of a set of tasks arriving at different times, the goal of our assignment problem is to maximize the total reward collected for completing tasks by their individual deadlines. The assignment must guarantee the thermal (i.e., guarantee the redline temperatures at the inlets are not exceeded) and power constraints of the data center. The decisions that the assignment problem must make are: 1) the P-states of cores, 2) the task to core assignment, and 3) the CRACs outlet temperatures.

Temperature evolution in the data center is in orders of minutes, while the execution of a task is in orders of seconds or milliseconds. Therefore, to make the assignment problem tractable, previous research (e.g., [2, 26]) has used a two-step assignment approach. The first step manages the power and the thermal evolution in the data center, while the second step performs workload balancing. In this paper, we apply the $\underline{\text{two-step}}$ $\underline{\text{assignment}}$ approach for a power constrained data center. In the first step, 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 step guarantees that the power and thermal constraints in the data center are not violated. In the second step, our approach implements a dynamic scheduler that sends tasks, as they come into the data center, to cores so that the actual execution rate of each task type on each core matches, as much as possible, the desired execution rate set by the first step. The dynamic scheduler can also make the decision to drop a task. The two-step assignment is depicted in Figure 2.

In Subsection B, we discuss the first-step assignment problem. We formulate the assignment problem as an exact $\underline{\text{mixed}}$ $\underline{\text{integer}}$ $\underline{\text{nonlinear}}$ $\underline{\text{program}}$ (MINLP). Because the MINLP is not scalable with the number of cores in the data center, we propose scalable techniques to find near-optimal solutions. We propose a dynamic scheduler to assign incoming tasks to cores in Subsection C.

B. First Step Assignment

1) Problem Formulation

At the first step, our approach is concerned with the long-term steady state total reward. In the steady state, maximizing the total reward is equivalent to maximizing the total reward rate (total reward per unit time). The decisions made (decision variables) at the first step 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 $\underline{\mathrm{TC}}$. Entry $\mathrm{TC}(i, k)$ 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.

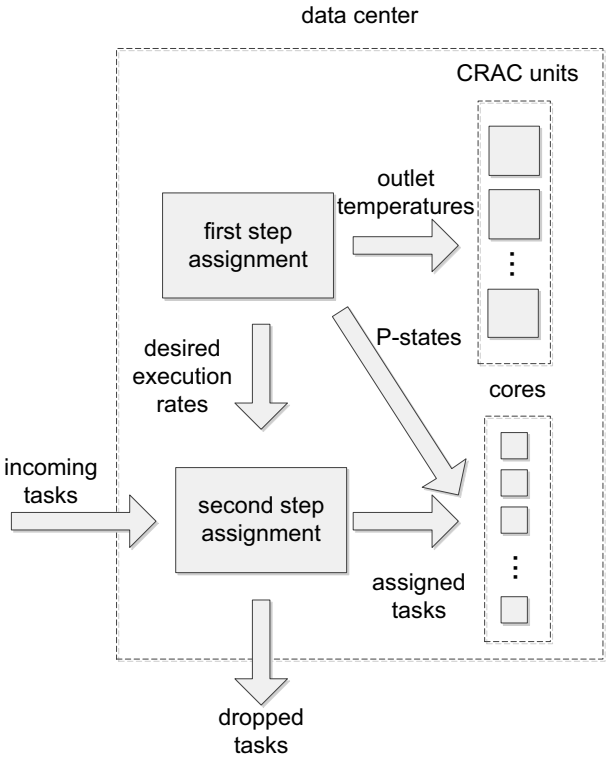

Figure 2. The assignment problem in the data center. The first step assigns the outlet temperatures of CRAC units, the P-states of cores, and the desired execution rate of task types on cores. The second step assigns the incoming tasks to cores based on the desired execution rate set by the first step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

The total reward rate (the optimization objective at the first step) is equal to

$$\sum_{i=1}^{T}\left(r_i \sum_{k=1}^{\text{NCORES}} \text{TC}(i,k)\right). \qquad (7)$$

This objective is subject to the following constraints:

1. $\sum_{i=1}^{T} \text{TC}(i,k)/\text{ECS}(i, \text{CT}_k, \text{PS}_k) \le 1$, $k = 1,\ldots,$ NCORES.
2. $\text{TC}(i,k)\,(m_i - (1/\text{ECS}(i, \text{CT}_k, \text{PS}_k)) \ge 0$ $i = 1,\ldots, T$ and $k = 1,\ldots,$ NCORES
3. $\sum_{k=1}^{\text{NCORES}} \text{TC}(i,k) \le \lambda_i, \quad i = 1,\ldots, T.$
4. $\sum_{j=1}^{\text{NCN}} \text{PCN}_j + \sum_{i=1}^{\text{NCRAC}} \text{PCRAC}_i \le P_{\text{const}}.$
5. $\boldsymbol{T}^{\text{in}} \le \boldsymbol{T}^{\text{redline}}.$

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., $1/\text{ECS}(i, j, k)$) is greater than m_i, no task of type i can make its deadline on core k even if its execution starts immediately after its arrival. Therefore, if $1/\text{ECS}(i, j, k) > m_i$, then Constraint 2 guarantees that $\text{TC}(i, k) = 0$ 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 [22]

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

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. One technique to find near-optimal solutions to MINLPs is to relax the integer constraints (i.e., allow integer variables to take continuous values). The relaxation makes the problem simpler to solve. Then, after solving the continuous version of the problem, we can find an integer solution that is close to the continuous solution (for example, by rounding each continuous variable to the closest integer value).

In the following subsections, we show how the integer P-state constraints are relaxed. However, as we show in the next subsection, the relaxation of the integer P-state constraints introduces additional nonlinear constraints. To avoid introducing additional nonlinear constraints, we divide the assignment problem at the first step into three stages. In the first stage, instead of assigning P-states to cores, we assign power consumption to cores. The decision variables in the first stage are the power consumption of each core and the outlet temperature of each CRAC unit. The second stage finds the closest integer P-state assignment from the cores power consumption assignment. The third stage finds the optimal assignment of the desired execution rate of each task type on each core for the integer P-states assignment obtained from the second stage.

2) Stage 1 Assignment

Relaxing the integer P-state constraint means that we allow a core to be assigned a continuous P-state value. 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 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 function of the power consumption of the core. Let $\underline{\text{PCORE}}_k$ be the power assigned to core k. Assume for a task of type i running on a core of type j the ECS as a function of PCORE_k is given by a monotonically increasing function $\underline{C}_{i,j}^{\text{ECS}}(\text{PCORE}_k)$ (in general, the more power a specific core consumes the faster it executes). Substituting $C_{i,j}^{\text{ECS}}(\text{PCORE}_k)$ for $\text{ECS}(i, j, \text{PS}_k)$ in Constraints 1 and 2 in Equation 7 results in additional nonlinear constraints. However, Constraints 1 and 2 relate power consumption at a core to the reward rate that is obtained from that core. Therefore, we can avoid the additional nonlinear constraints by defining the aggregate (over all task types) reward rate of a core of type j, $\underline{\text{ARR}}_j$, as a function of the power consumed at a core of type j. Now the only nonlinear constraint is Constraint 4. Its nonlinearity is due to the power consumption of CRAC units. The problem in Equation 7 is simplified to the following NLP

$$\text{maximize} \sum_{k=1}^{\text{NCORES}} \text{ARR}_{\mathbf{CT}_k}(\text{PCORE}_k). \qquad (9)$$

Subject to

1. $\sum_{j=1}^{\text{NCN}} \text{PCN}_j + \sum_{i=1}^{\text{NCRAC}} \text{PCRAC}_i \le P_{\text{const}}$.
2. $\boldsymbol{T}^{\text{in}} \le \boldsymbol{T}^{\text{redline}}$.

Now we describe how the ARR_j functions are calculated. First, we show how the reward rate for executing tasks of type i on a core of type j as a function of the power consumption of the core, $\underline{\text{RR}}_{i,j}$, is calculated. Then we show how these functions are aggregated to obtain ARR_j.

To minimize the difference between the integer solution and the relaxed solution of the Stage 1 problem, we select $\text{RR}_{i,j}$ so that it goes through the points

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

Each of these points is the power consumption of a P-state and the reward rate obtained at that P-state. An intuitive value of $\text{RR}_{i,j}$ when PCORE_k is not equal to any P-state power consumption, is to assume that the compute core can switch between a P-state with power consumption lower than PCORE_k and a P-state with a higher power consumption than PCORE_k consumption such that the average power consumption is equal to PCORE_k. Therefore, we chose to represent $\text{RR}_{i,j}$ using a piecewise linear function.

For example, assume a core of type j with 4 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 values for task type i for each of the 4 P-states are 1.2, 0.9, 0.5, and 0 W, respectively. Assume the reward of completing a task of type i by its deadline is 1, i.e., $r_i = 1$. The $\text{RR}_{i,j}$ 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 Figure 3.

The above calculation of $\text{RR}_{i,j}$ does not consider the deadline constraints. For example, if m_i was equal to 1.5, then no core of type j running in P-state 2 will ever be able to make the deadline of any task of type i and the reward rate at that power consumption will be 0. The $\text{RR}_{i,j}$ function for this example is shown in Figure 4.

To calculate ARR_j, we average the $\text{RR}_{i,j}$ functions over the "best" ψ% of task types for core type j, where ψ is a parameter set by the user. To calculate the "best" ψ% task types for core type j, we compute the average ratio of reward rate to power consumption (i.e., $\text{RR}_{i,j}(\text{PCORE}_k)/\text{PCORE}_k$) over all P-states, except the last (turned off) P-state, for each task type i. Then, we choose the ψ% task types with the highest average ratio of reward rate to power consumption. In some cases, ties may occur, i.e., some task types may have the same average ratio of reward rate to power consumption. We break the ties arbitrarily.

Because the Stage 1 problem is a maximization problem, if the ARR_j functions are concave, then the computational expense of the optimization can be greatly reduced. However, the ARR_j functions are not guaranteed to be concave. Therefore, the ARR_j functions would have to be modeled by binary variables. This would make Stage 1 optimization problem computationally infeasible for a large number of cores. For instance, consider the example shown in Figure 4. Assume that there is only one task type in the data center. The ARR_j will be equal to the function shown in Figure 4. This function is not a concave function.

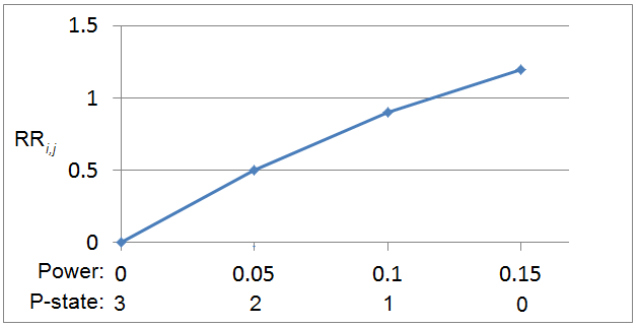

Figure 3. An example $\text{RR}_{i,j}$ function.

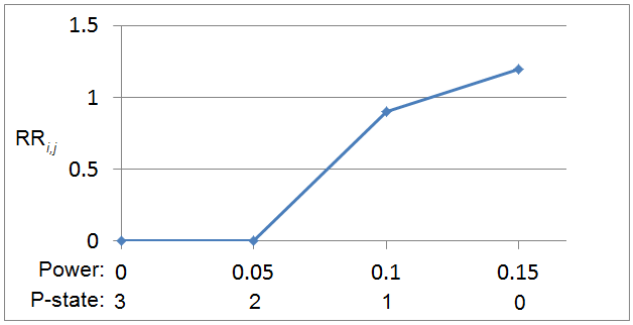

Figure 4. An example that illustrates the calculation of $\text{RR}_{i,j}$ when a P-state cannot make the deadline of a task type.

The non-concavity of an ARR_j function is caused by a P-state that has an aggregate reward rate to power consumption ratio that is less than its next lower P-state. We call this P-state a "bad" P-state. For example, assume ARR_j is equal to the function in Figure 4. P-state 2 is a "bad" P-state because the ratio of its aggregate reward rate to power consumption is 0, where the ratio of P-state 1's aggregate reward rate to power consumption is 9. If we ignore P-state 2 (the "bad" P-state), then the ARR_j function will be concave. This case is shown in Figure 5.

In general, the solution to the Stage 1 problem, when the "bad" P-states are **not** ignored, will avoid "bad" P-states. For example, consider the case where a compute node of type *j* has 2 cores. Assume that the compute node can assign a maximum of 0.1 W total power to its cores. If the function in Figure 4 is the aggregate reward rate function 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 aggregate reward rate of 0.45, which is the same answer 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.

The above procedure requires the solution of an NLP, which may result in a local optimal solution. However, if the CRAC units' outlet temperatures are fixed, then the Stage 1 problem (Equation 9) becomes an LP instead of an NLP. In addition, the outlet temperatures of the CRAC units usually have a granularity of 1 degree. Therefore, a discretized search for the optimal set of CRAC units' outlet temperatures can be performed for a small number of CRAC units. However, the number of combinations of CRAC units' outlet temperatures increases exponentially with the number of CRAC units.

One method to reduce the number of combinations is to consider a multi-step method where the first step is a coarse-grained search for the entire range of possible outlet temperatures. Every subsequent step searches around the best set of CRAC units' outlet temperatures found in the previous step, however, with a finer granularity.

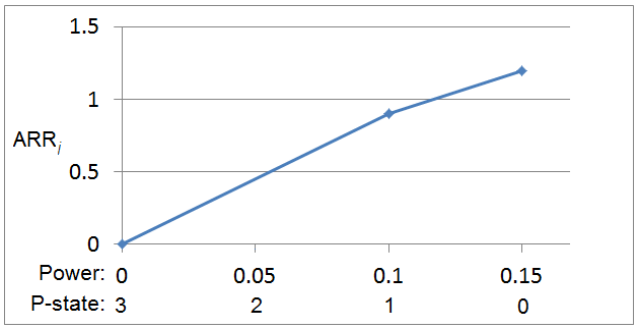

Figure 5. An example that illustrates the calculation of ARR_j when the "bad" P-states are ignored.

3) Stage 2 Assignment

The purpose of the Stage 2 assignment is to convert the power assigned to a core *k* ($PCORE_k$) into a P-state. The solution to Stage 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 Stage 1 assignment. The following procedure converts the $PCORE_k$ values into a P-state assignment:

1. For each compute 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. Calculate the power consumption by using Equation 1.
   b. Repeat Step *c* until the power consumption calculated by Equation 1 is less than the power consumption that resulted from the Stage 1 assignment
   c. Increment the P-state of the core with the smallest P-state by 1.

Because the ARR_j functions are concave, the aggregate reward rate to power consumption of a P-state will always be lower than or equal to that of a higher P-state. Therefore, in Step 2.*c* of the procedure above we increment the P-state of the core with the lowest P-state.

4) Stage 3 Assignment

Now that the values of two of the three decision variables have been found, the optimization problem in Equation 7 becomes an LP. In Stage 3, we solve this LP to find the optimal desired execution rate of task types on cores (i.e., the optimal TC matrix). The desired execution rates are only optimal for the fixed P-states and CRAC units' outlet temperatures assignment that were determined by the previous two stages.

C. Second Step Assignment

The dynamic scheduler at the second step keeps track of the actual execution rate of each task type on each core in matrix ATC. The goal of the dynamic scheduler is to make the ratio of $ATC(i, k)/TC(i, k)$ 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 has the minimum $ATC(i,k)/TC(i,k)$ value (unless $ATC(i,k)/TC(i,k) \geq 1$) and can complete *t* before its deadline. If no such core exists, then the dynamic scheduler drops *t*.

VI. Simulation Setup

A. 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.

B. Compute Nodes

In our simulations, we used a varying number of compute nodes. Each compute node belongs to one of two node types based on two 7U servers listed in the results of

SPECpower_ssj2008 benchmarks. The first node type is based on the HP ProLiant DL785 G5 server. This server has 8 AMD Opteron 8381 HE processors with 4 cores in each processor. The second node type is an NEC Express5800/A1080a-S server. This server has 4 Intel Xeon X7560 processors. Each processor has 8 cores. Table I lists the parameters of both node types. Details on how the values of the parameters were obtained are in Appendix A.

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

C. ECS Matrices

The estimated time to compute values are arranged in a three dimensional ECS matrix. These dimensions represent task types, compute node types, and P-states. In our simulations, we have 8 task types, 2 compute node types, and 4 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 [28]). Therefore, we assumed that the average ECS over all task types for node types 1 and 2 is 0.6 and 1, respectively. We assume that the average ECS over all node types for task type i is half that of task type $i + 1$. Let $\underline{\text{rand}}[a, b]$ 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 $\text{rand}[1 - V_{\text{ECS}}, 1+V_{\text{ECS}}]$. The variation factor is used so that there is affinity between task types and node types. The value of V_{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. The ECS is extended in the third dimension by using equation 10.

$$\text{ECS}(i, j, k) = \text{ECS}(i, j, 0) \cdot \frac{f_{j,k}}{f_{j,0}} \cdot \text{rand}\left[1 - V_{\text{prop}}, 1+V_{\text{prop}}\right]. \quad (10)$$

The variation factor $\text{rand}\left[1 - V_{\text{prop}}, 1+V_{\text{prop}}\right]$ is used so that the ECS of a task type on a core type is not exactly proportional to the clock frequency of the P-states. We used two different values for V_{prop} in different sets of simulations. These values are 0.1 and 0.3.

TABLE I. 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

Equation 10 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 entry (i, j, k) is higher than entry $(i, j, k - 1)$, then we regenerate a random number $\text{rand}\left[1 - V_{\text{prop}}, 1+V_{\text{prop}}\right]$ and recomputed $\text{ECS}(i, j, k)$ 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. We note that the requirement that $\text{ECS}(i, j, k) \leq \text{ECS}(i, j, k - 1)$ is just to make the ECS more realistic and is not a requirement of our assignment technique described in sections V.*B* and V.*C*.

D. Task Types

The number of task types in all of our simulations is assumed to be 8. The reward for completing a task of type i by its deadline is assumed to be equal to the reciprocal of the average ECS of that task type over all compute node types, i.e.,

$$r_i = 1/(\text{average ECS of task type } i). \quad (11)$$

The above reward value is used for the purposes of our simulations. We note that the reward value is not required to be equal to the reciprocal of task easiness for our assignment technique to work.

Now we show how the m_i values that are used to calculate the deadline of individual tasks are generated. Let $\underline{\text{MinECS}}_i$ and $\underline{\text{MaxECS}}_i$ 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. Let $\underline{\text{NTYPES}}$ be the number of compute node types in the data center. MinECS_i is given by

$$\text{MinECS}_i = \min\left\{\text{ECS}\left(i, j, \eta_j - 2\right) : 1 \leq j \leq \text{NTYPES}\right\}. \quad (12)$$

MaxECS_i is given by

$$\text{MaxECS}_i = \max\{\text{ECS}(i, j, 0) : 1 \leq j \leq \text{NTYPES}\}. \quad (13)$$

The value of m_i is given by

$$m_i = 1.5\ \text{rand}[1/\text{MaxECS}_i,\ 1/\text{MinECS}_i]. \quad (14)$$

We used Equation 14 to compute m_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 we need to generate for a task type is its arrival rate, λ_i. Let $\underline{\text{SumECS}}_i$ 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

$$\text{SumECS}_i = \sum_{k=1}^{\text{NCORES}} \text{ECS}(i, \text{CT}_k, 0)/T. \quad (15)$$

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

$$\lambda_i = \text{SumECS}_i \cdot \text{rand}[1 - V_{\text{arrival}}, 1 + V_{\text{arrival}}]. \quad (16)$$

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

E. Cross Interference Coefficients

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 [29]. In [29], Computational Fluid Dynamics (CFD) simulations are used to obtain cross interference coefficients for a small data center (10 racks with 5 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 [29]. In our simulations, we use 150 compute nodes and 3 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.

F. Power and Thermal Constraints

To set a reasonable power constraint for our simulations, 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 following NLP problem:

$$\text{Minimize} \sum_{j=1}^{\text{NCN}} \text{PCN}_j + \sum_{i=1}^{\text{NCRAC}} \text{PCRAC}_i . \quad (17)$$

Subject to

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

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 CRAC units' outlet temperatures. 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.

Let $P_{\min}$ and $P_{\max}$ be the upper bounds on the minimum and maximum power consumption of the data center, respectively. In our simulations, the power constraint, P_{const}, is given by

$$P_{\text{const}} = (P_{\min} + P_{\max})/2 \quad (18)$$

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

G. 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 Figure 1.

VII. Simulation Results

A. Comparison Overview

To show the benefit of using the assignment technique proposed in this paper, we performed simulations for the first step assignment problem and compared our technique with a technique that only considers putting a core in P-state 0 or turning it off. The authors in [26] 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. In our case, instead of QoS we are concerned with reward rate and so we have implemented an adaptation of the technique in [26] to solve our problem.

Let $\text{FRAC}(i, j)$ be the fraction of compute resource (i.e., the number of cores) used at compute node j used to execute tasks of type i. The power consumption of compute node j is given by

$$\text{PCN}_j = B_i + \pi_{\text{NT}_j,0} \sum_{i=1}^{T} \text{FRAC}(i,j) . \quad (19)$$

The reward rate for a task of type i at compute node j is given by $r_i \text{ECS}(i,j,0) \left| \text{cores}_j \right| \text{FRAC}(i,j)$ so that the assignment problem is given by

$$\text{maximize} \sum_{i=1}^{T} \sum_{j=1}^{\text{NCN}} r_i \text{ECS}(i,j,0) \left| \text{cores}_j \right| \text{FRAC}(i,j). \quad (21)$$

Subject to

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

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

Constraint 1 guarantees that the execution rate for a task type is not higher than its arrival rate. For a compute node, the sum of the fractions over all task types must not exceed 1. 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 FRAC(i, j)=0 whenever $m_i < (1/\text{ECS}(i, \text{NT}_j, 0))$. The problem in Equation 21 is, in general, an NLP problem due to the power consumption of CRAC units. The total number of cores used at compute node j is equal to

$$\left|\text{cores}_j\right| \sum_{i=1}^{T} \text{FRAC}(i, j). \tag{22}$$

This value may not be integer. Therefore, after solving the problem in Equation 21, the fractions FRAC(i, j) for compute node j are all reduced by the same percentage so that the value of Equation 22 is an integer number.

B. Results

We have conducted three sets of simulations. Each set of simulations consists of 25 simulation runs. Each simulation run is for a data center consisting of 3 CRAC units and 150 compute nodes. For each data center, we obtain the total reward rate by solving the three-stage assignment problem given in Section V.*B,* once by calculating the aggregate reward rate functions using the "best" 25% of task types (i.e., $\psi = 25$) and another time using $\psi = 50$, and also compute the result for using the best of the two. We compare the total reward rate of the three-stage assignment against the total reward obtained by solving the problem in Equation 21. (see Figure 6).

Each bar in Figure 6 is the average percentage improvement of our technique over the technique based on [26]. We show a 95% confidence interval for each result. In the first two sets of simulations, we assumed that the static power consumption of P-state 0 for each core type is 30% of the total core power consumption in P-state 0 compared to 20% for the last set of simulations (see Appendix A for a detailed description of the static and dynamic power calculation). For the last two sets of simulations, the value of V_{prop}, that is used to generate the ECS matrices, is 0.3 compared to 0.1 for the first set of simulations.

Three interesting observations can be made by examining the results. First, when P-state 0 static power consumption is 20% versus 30%, the percentage improvement is higher. This can be explained as follows. Because P-state 0 runs at a higher voltage and frequency, the percentage of dynamic power consumption is usually higher than the other P-states. Therefore, the other P-states will have higher percentage of static power consumption compared to 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 in Equation 21 will, in general, perform better than the three-stage technique given in Section V.*B*.

The second observation is that when V_{prop} is 0.3 versus 0.1, the percentage improvement is higher. This is because, for a given core type, the higher the value of V_{prop}, the higher the 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 can be obtained by matching task types with their better suited P-states.

Finally, the third observation is that for different values of ψ we get different solutions. Many factors affect the best choice of ψ. The main factors are the arrival rates of task types, the power constraint, and the affinity of task types to machines. As the results show, the percentage improvement is slightly higher when ψ is equal to 50 versus 25 (although with significant overlap between the confidence intervals). However, for some instances, a ψ value of 25 was better. We explain the slightly higher percentage improvement for when ψ is equal to 50 as follows. When the "best" 25% of the task types are chosen to calculate the aggregate reward rate function (used at Stage 1), the aggregate reward rate obtained at Stage 1 will be higher than the total reward rate obtained at Stage 3. This is because the arrival rate of the "best" task types will not be enough to keep the cores busy. Therefore, Stage 3 of the assignment will assign other task types to cores which will result in a lower reward rate. When ψ is 50 this problem is mitigated.

Both our Stage 1 problem (Equation 9) and the problem described in Equation 21 are NLPs 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 3 CRAC units, 150 compute nodes, and 8 task types, is computationally expensive. However, tests on smaller problems, i.e., 2 CRAC units, 40 compute nodes, and 8 task types, have shown no improvement.

Our results show an average improvement of up to 10% when using our technique over the one given in Equation 21. Higher percentage improvements may be experienced for data centers with different setups. Because today's data centers are large, this improvement can mean millions of dollars in additional revenue or power savings.

VIII. Future Work

In this paper, we studied the problem of maximizing the total reward rate in a data center that is subject to a total power consumption constraint. In data centers that must provide stringent workload performance guarantees and where power constraints are not active (restricting), minimizing the overall power consumption may be a more relevant problem. Therefore, one future work extension is to study the problem of minimizing the power consumption subject to a total reward rate constraint.

There are also many parameters that can be changed in the simulation setup (e.g., static power consumption, V_{prop}, the easiness of task types, the performance of core types). Our simulations consider a few of them. In future work, we will conduct more simulation studies to examine the impact

of some of these parameters on the performance of our approach.

IX. Conclusions

In a power constrained data center, delegating the P-state assignment to a server level controller is not effective. In this paper, we show how the P-state assignment can be included at the data center level. We divide the assignment problem into two steps. We formulate the first step assignment as a MINLP. Because the MINLP is not scalable with respect to the number of cores, we propose a multi-stage, scalable assignment technique. At the second step, we propose a dynamic scheduler to assign tasks entering the data center to cores.

We conducted simulations to show the effectiveness of our technique over a technique based on [26]. In some cases, our technique achieved a 10% improvement over the compared technique. In today's large data centers, this improvement can mean millions of dollars in additional revenue or power savings.

In many data centers, P-state 0 is not the P-state with the highest performance to power consumption ratio. Therefore, using the assignment technique in this paper will result in a better total reward over a technique that choses between putting a core in P-state 0 and turning it off. However, if P-state 0 is the P-state with the highest performance to power consumption ratio, then we show a technique that choses between putting a core in P-state 0 and turning it off that is, in general, more effective.

Acknowledgments

The authors thank Jerry Potter, Greg Pfister, Jay Smith, Ryan Friese, Nishit Kapadia, and Mark Oxley for their valuable comments on this work. This research was supported by the NSF under grant number CNS-0905399, and by the Colorado State University George T. Abell Endowment.

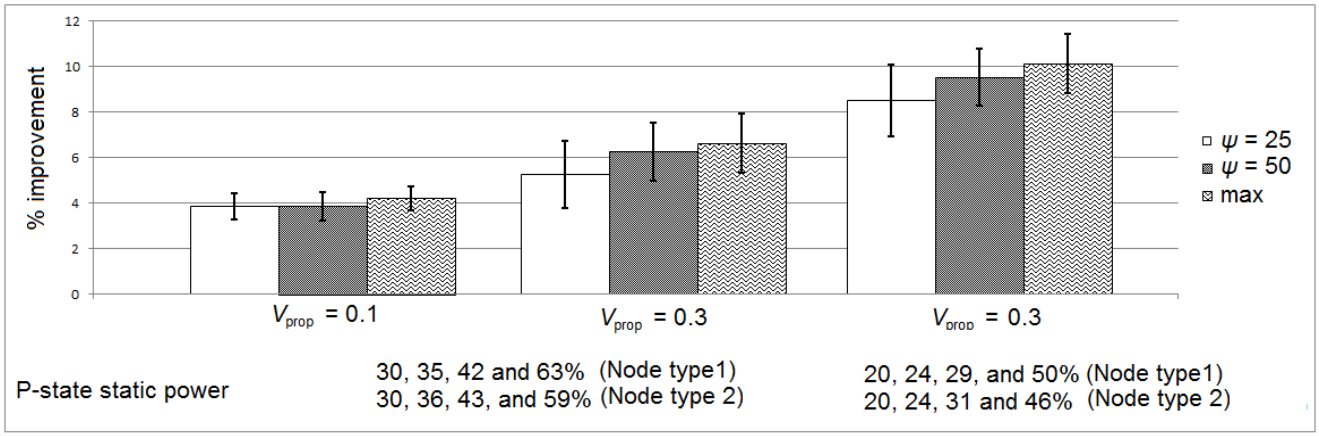

Figure 6. This figure shows the average percentage improvement of the total reward rate obtained by using the three-stage assignment given in section V.*B* versus the assignment problem that is based on [26] (given in Equation 21). A 95% confidence interval is shown for each average percentage improvement. In the first two column groups, the static power consumption of P-state 0 is 30% of the total core power consumption in P-state 0 compared to 20% for the last column group. The static power consumption percentage for the other P-states for each node type is also shown. In the last two column groups, the value of V_{prop} is 0.3 compared to 0.1 for the first column group.

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," *19th ACM International Symposium on High Performance Distributed Computing (HPDC'10)*, 2010, pp. 130-141.

[3] S. Ali, T. D. Braun, H. J. Siegel, A. A. Maciejewski, N. Beck, L. Boloni, 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*, Vol. 63: Parallel, Distributed, and Pervasive Computing, pp. 91–128, 2005.

[4] A. M. Al-Qawasmeh, A. A. Maciejewski, and H. J. Siegel, "Characterizing heterogeneous computing envinronments using singular value decomposition," 19th Heterogeneity in Computing Workshop (HCW 2010), *24th International Parallel and Distributed Processing Symposium, Workshops and PhD Forum (IPDPSW 2010)*, Apr. 2010, pp. 1-9.

[5] A. M. Al-Qawasmeh, A. A. Maciejewski, and H. J. Siegel, "Characterizing task-machine affinity in heterogeneous computing envinronments," 20th Heterogeneity in Computing Workshop (HCW 2011), *25th International Parallel and Distributed Processing Symposium, Workshops and PhD Forum (IPDPSW 2011)*, Apr. 2011, pp. 34-44.

[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," *The Journal of 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," *9th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA '11)*, Dec. 2011, 10 pp.

[8] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Dynamic and aggressive scheduling techniques for power-aware real-time

systems," *22nd IEEE Real-Time Systems Symposium (RTSS '01)*, Dec. 2001, pp. 95-105.

[9] H. Barada, S. M. Sait, and N. Baig, "Task matching and scheduling in heterogeneous systems using simulated evolution," 10th Heterogeneous Computing Workshop (HCW 2001), *15th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2001)*, Apr. 2001, pp. 875-882.

[10] D. Brown and C. Reams, "Toward energy-efficient computing," *Communications of the ACM*, Vol. 53, No. 3, Mar. 2010, 14 pp.

[11] J. A. Butts and G. S. Sohi, "A static power model for architects," *33rd annual ACM/IEEE international symposium on Microarchitecture (MICRO 33)*, Dec. 2000, pp. 191-201.

[12] 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, Sep. 2002, pp. 1338-1361.

[13] E. N. Elnozahy, M. Kistler, and R. Rajamony, "Energy-efficient server clusters," *Second International Workshop on Power-Aware Computer Systems, 2002*, pp. 179-197.

[14] Environmental Protection Agency. *Report to Congress on Server and Data Center Energy Efficiency*, 2007. http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf. Last accessed 12/6/2011.

[15] 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 Journal*, Vol. 12, No. 1, 2008, pp. 59-67.

[16] A. Ghafoor and J. Yang, "A distributed heterogeneous supercomputing management system," *IEEE Computer*, Vol. 26, No. 6, June 1993, pp. 78–86.

[17] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., and Toshiba Corporation Std., *Advanced Configuration and Power Interface Specification*, Rev. 4.0a, Apr. 2010, http://www.acpi.info/DOWNLOADS/ACPIspec40a.pdf. Last accessed 1/3/2012.

[18] M. Kafil and I. Ahmad, "Optimal task assignment in heterogeneous distributed computing systems," *IEEE Concurrency*, Vol. 6, No. 3, July 1998, pp. 42–51.

[19] A. Khokhar, V. K. Prasanna, M. E. Shaaban, and C. Wang, "Heterogeneous computing: Challenges and opportunities," *IEEE Computer*, Vol. 26, No. 6, June 1993, pp. 18–27.

[20] Y. Lin and L. He, "Dual-Vdd interconnect with chip-level time slack allocation for FPGA power reduction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, No. 10, Oct. 2006, pp. 2023 – 2034.

[21] J. R. Lorch and A. J. Smith, "Improving voltage scaling algorithims with PACE," *ACM SIGMETRICS Performance Evaluation Review*, Vol. 29, No. 1, June 2001, pp.50 -61.

[22] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling "cool": Temperature-aware workload placement in data centers," *USENIX Annual Technical Conference (ATEC '05)*, 2005, 14 pp.

[23] T. Mukherjee, G. Varsamopoulos, S. K. S. Gupta, and S. Rungta, "Measurement-based Power Profiling of Data Center Equipment," IEEE Internatinoal Conference on Cluster Computing, Sep. 2007, pp. 476 – 477.

[24] E. Pakbaznia, M. Ghasemazar, and M. Pedram, "Temperature-aware dynamic resource provisioning in a power-optimized datacenter," *The Conference on Design, Automation and Test in Europe 2010*, 2010, pp. 124-129.

[25] V. Pallipadi and A. Starikovsky, "The ondemand governor". *The 2006 Linux Symposium*, 2006, pp. 215-229.

[26] L. Parolini, N. Tolia, B. Sinopoli, and B. H. Krogh, "A cyber-physical systems approach to energy management in data centers," *1st ACM/IEEE International Conference on Cyber-Physical Systems*, 2010, pp. 168-177.

[27] H. Singh and A. Youssef, "Mapping and scheduling heterogeneous task graphs using genetic algorithms," *5th IEEE Heterogeneous Computing Workshop (HCW '96)*, 1996, pp. 86-97.

[28] Standard Performance Evaluation Corporation (SPEC), SPECpower_ssj2008, http://www.spec.org/power_ssj2008. Accessed Dec. 12 2011.

[29] Q. Tang, T. Mukherjee, S. K. S. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters," *4th International Conference on Intelligent Sensing and Information Processing (ICISIP 2006)*, Dec. 2006, pp. 203-208.

[30] N. Tolia, Z. Wang, P. Ranganathan, C. Bash, and M. Marwah, "Unified thermal and power management in server enclosures," *InterPACK '09*, July 2009, 10 pp.

[31] C. Xian, Y.-H. Lu, and Z. Li, "Dynamic voltage scaling for multitasking real-time systems with uncertain execution time," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 27, No. 8, Aug. 2008, pp. 1467-1478.

[32] D. Xu, K. Nahrstedt, and D. Wichadakul, "QoS and contention-aware multi-resource reservation," *Cluster Computing*, Vol. 4, No. 2, Apr. 2001, pp. 95–107.

[33] D. Young, J. Apodaca, L. Briceno, J. Smith, S. Pasricha, A. A. Maciejewski, H. J. Siegel, B. Khemka, S. Bahirat, A. Ramirez, and Y. Zou, "Energy-constrained dynamic resource allocation in a heterogeneous computing environement," *4th International Workshop on Parallel Programming Models and Systems Software for High-End Computing (P2S2)*, Sep. 2011, pp. 298-307.

[34] H. Yu, B. Veeravalli, and Y. Ha, "Dynamic scheduling of imprecise-computation tasks in maximizing QoS under energy constraints for embedded systems," *2008 Asia and South Pacific Design Automation Conference (ASPDAC '08)*, Mar. 2008, pp. 452-455.

APPENDIX A

In this appendix, we show how the parameters for two compute node types used in the simulations (Table I) are obtained. The first compute node type is based on the HP ProLiant DL785 G5 server. This server has 8 AMD Opteron 8381 HE processors with 4 cores in each processor. We assumed that the power consumption of the processor is 0.055 kW. This is based on the thermal design power (TDP) numbers found in the AMD data sheet [1]. At 100% utilization, the power consumption of the server was 0.793 kW. To obtain the base power consumption (B_1) of this server, we subtracted the total power consumed by the 8 processors from the power consumption at 100%. The base power consumption is equal to 0.353 kW.

The AMD Opteron processor has 4 P-states. The frequencies of P-states 0, 1, 2, and 3 are 2500, 2100, 1700, and 800 MHz, respectively. The supply voltages of the P-states are 1.325, 1.25, 1.175, and 1.025 V, respectively. To obtain P-state 0 power consumption of an individual core, the total power consumption of the processor is divided by the number of cores. Therefore, the power consumption of P-state 0 is 0.01375 kW. The power consumption of a core is due to static and dynamic power consumption. In [11], the authors show that the static power consumption for a core is equal to a constant multiplied by the supply voltage. Let β_j be this constant for core type j. The dynamic power is calculated by the standard CMOS dynamic power dissipation

formula. If S_j is the number of transistor switches per clock cycle for core type j, CL_j is the capacitive load for core type j, $f_{j,k}$ is the clock frequency of a core of type j running in P-state k, and $V_{j,k}$ is the supply voltage of a core of type j running in P-state k, then the dynamic power consumption of core type j running in P-state k is equal to $S_j \cdot CL_j \cdot f_{j,k} \cdot V_{j,k}^2$. We assume that $S_j \cdot CL_j$ is a constant and is not dependent on the P-state. Let $SC_j = S_j \cdot CL_j$. The total core power consumption, $\pi_{j,k}$, is given by

$$\pi_{j,k}=SC_j \cdot f_{j,k} \cdot V_{j,k}^2+\beta_j \cdot V_{j,k}. \quad (23)$$

In our simulations, we assume that the static power consumption as a percentage of the total power consumption for P-state 0 is known. Therefore, SC and β in Equation 23 can be calculated for each core type and the power consumption of the other P-states can be calculated by substituting SC, β, the frequency of each P-state, and the supply voltage of each P-state.

The air flow rate at node type 1 is assumed to be $0.07 m^3/s$. This will guarantee that the maximum increase in temperature of the air going through a node of type 1 will be 9.4° Celsius or 17° Fahrenheit. We assume that the air density is equal to 1.205 kg/m^3 and the specific heat capacity of air is 1 (in reality, the density of air and its specific heat capacity depend on multiple factors such as pressure and temperature).

The second node type is an NEC Express5800/A1080a-S server. This server has 4 Intel Xeon X7560 processors. Each processor has 8 cores. The frequency of P-state 0 is 2666 MHz. We assumed there are 4 P-states and the frequencies of P-states 1, 2, and 3 are 2200, 1700, and 1000 MHz. The supply voltage for P-states 0 is assumed to be 1.35 V (this is based on Intel Xeon E7540 processor that has the same feature size). The voltages of P-states 1, 2, and 3 are assumed to be 1.268, 1.18, 1.056 V, respectively. The calculation of the other parameters of the second node type is similar to calculation of the parameters of the first node type.

APPENDIX B

In this appendix, we describe how an LP feasibility problem can be used to generate the cross interference coefficients. As opposed to [29], in our simulations, the data centers contain more than one CRAC unit. Therefore, the cross interference coefficients must include the CRAC units in addition to the compute nodes. We assume that the first NCRAC i and j indices in $\alpha_{i,j}$ are CRAC units. If $i \le$ NCRAC , then $\alpha_{i,j}$ is the percentage of air flow generated from CRAC unit i, otherwise, $\alpha_{i,j}$ is the percentage of the air flow generated from compute node i – NCRAC. Similarly, if $i \le$ NCRAC, then $\alpha_{i,j}$ is the air flow recirculated into CRAC unit j, otherwise, $\alpha_{i,j}$ is the air flow recirculated into compute node j – NCRAC.

The exit coefficient EC_i is the percentage of air flow of compute node i that goes into CRAC units [29]. Recirculation coefficient RC_i is the percentage of the air flow at the inlet of compute node i that is recirculated from the outlet of other compute nodes [29]. The position of a compute node within a rack affects its EC and RC. Compute nodes at the bottom of a rack will have a low RC (most if their inlet air comes from the perforated tiles) and a low EC (most of their air is recirculated into compute nodes). Compute nodes at the top of the rack have a high EC and a high RC. Compute nodes in [29] are labeled A-E. Node A is at the bottom of the rack and node B is at the top of the rack. Table II shows the ranges of EC and RC for each node label based on the CFD simulations in [29].

TABLE II. THE RANGES OF EC AND RC FOR DIFFERENT COMPUTE NODE LABELS.

Label	EC range	RC range
A	30-40%	0-10%
B	30-40%	0-20%
C	40-50%	10-30%
D	70-80%	30-70%
E	80-90%	40-80%

Because we have more than one CRAC unit in our simulations, for each compute node, we will have an exit coefficient per CRAC unit. Let $EC_{i,j}$ be the exit coefficients of compute node i for CRAC unit j.

Usually a data center is arranged in a hot aisle/cold aisle fashion. Figure 1 shows the data center layout that we assumed for our simulations. Assuming that CRAC unit i faces hot-aisle i, the EC for a compute node who's outlet air goes into hot-aisle i will have a greater EC to CRAC unit i than to any other CRAC unit. Let $MinEC_l$ and $MaxEC_l$ be the minimum and maximum EC for a compute node with label l (as shown in Table II). Let $MinRC_l$ and $MaxRC_l$ be the minimum and maximum RC for a compute node with label l (as shown in Table II). Let L_j be the label of compute node j. Let HA_j be the hot-aisle that compute node j belongs to (i.e., compute node j's outlet air goes into hot-aisle HA_j). For simplicity, we assume that $M(i, j)$ is the percentage of the EC of a compute node in hot aisle i that goes to CRAC unit j. Let F be the vector of air flow rates $F = [FCRAC_1, \ldots, FCRAC_{NCRAC}, FCN_1, \ldots, FCN_{NCN}]^T$. Let $\boldsymbol{\alpha}$ be the matrix of the cross interference coefficients.

The LP feasibility problem for generating the cross interference coefficients is given by

Find $\boldsymbol{\alpha}$

Subject to

1. $\sum_{i=1}^{NCRAC+NCN} \alpha_{i,j}=1$, $1 \le j \le$ NCRAC+NCN.
2. $\sum_{j=1}^{NCRAC+NCN} \alpha_{i,j}F_i = F_j$, $1 \le i \le$ NCRAC+NCN.
3. $MinEC_{L_i} M(HA_i, j) \le \alpha_{i+NCRAC,j}$, $1 \le i \le$ NCN and $1 \le j \le$ NCRAC.
4. $\alpha_{i+NCRAC,j} \le MaxEC_{L_i} M(HA_i, j)$, $1 \le i \le$ NCN and $1 \le j \le$ NCRAC.
5. $MinRC_{L_j} \le \sum_{i=1}^{NCN} \alpha_{i+NCRAC,j+NCRAC} \le MinRC_{L_j}$, $1 \le j \le$ NCN.

The sum of the percentages of the air flows generated from a CRAC unit or a compute node must equal to 1. This is guaranteed by Constraint 1. Constraint 2 guarantees that the sum of the air flows at the inlet of a CRAC unit or a compute node is equal to its air flow rate. Constraints 3 and 4 guarantee that the sum of the exit coefficients at a compute node is within the range shown in Table II, and that the percentages set by matrix M are satisfied. The range of the recirculation coefficients for each node is guaranteed by Constraint 5.