J. Parallel Distrib. Comput. ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Rate-based thermal, power, and co-location aware resource management for heterogeneous data centers

Mark A. Oxley[a,*], Eric Jonardi[a], Sudeep Pasricha[a], Anthony A. Maciejewski[a], Howard Jay Siegel[a], Patrick J. Burns[b], Gregory A. Koenig[c]

[a] *Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523, USA*
[b] *Vice President for Information Technology, Colorado State University, Fort Collins, CO 80523, USA*
[c] *Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA*

HIGHLIGHTS

- Derivation of a new detailed model of a heterogeneous data center.
- Design of resource management methods for co-location, power, and temperature.
- Analyses of resource management methods using several facility configurations.
- Sensitivity analysis under a range of constraint values and workload environments.

ARTICLE INFO

Article history:
Received 22 August 2016
Received in revised form 2 February 2017
Accepted 29 April 2017
Available online xxxx

Keywords:
Heterogeneous computing
Resource management
Thermal-aware computing
Power-aware computing
DVFS
Memory interference

ABSTRACT

Today's data centers contain large numbers of compute nodes that require substantial power, and therefore require a large amount of cooling resources to operate at a reliable temperature. The high power consumption of the computing and cooling systems produces extraordinary electricity costs, requiring some data center operators to be constrained by a specified electricity budget. In addition, the processors within these systems contain a large number of cores with shared resources (e.g., last-level cache), heavily affecting the performance of tasks that are co-located on cores and contend for these resources. This problem is only exacerbated as processors move to the many-core realm. These issues lead to interesting performance-power tradeoffs; by considering resource management in a holistic fashion, the performance of the computing system can be maximized while satisfying power and temperature constraints. In this work, the performance of the system is quantified as the total reward earned from completing tasks by their individual deadlines. By designing three resource allocation techniques, we perform a rigorous analysis on thermal, power, and co-location aware resource management using two different facility configurations, three different workload environments, and a sensitivity analysis of the power and thermal constraints.

© 2017 Elsevier Inc. All rights reserved.

1. Introduction

Faster execution time has long been the highest priority design objective for data centers. The great success achieved thus far in optimizing performance of these systems has come at the cost of increased power consumption, leading to increased ownership costs from powering and cooling these systems. Furthermore, the increased demand for performance from these systems has led to chip designers increasing the number of cores in their multicore processors, and led data center administrators to include these high-performance multicore processors in their facilities. The increased number of processing chips in data centers also has increased the power consumption of computing and cooling. The increased number of cores per chip has led to greater interference between cores competing for shared resources. The push to exaflop-capable systems will require even more on-chip parallelism (and sharing nodes among multiple tasks), thereby reducing performance because of this interference and requiring larger systems to meet needs, thus exacerbating power consumption.

Motivated by the need to reduce electricity costs and encourage green computing, the Green500 list aims to switch the paradigm

* Corresponding author.
E-mail addresses: mark.oxley@colostate.edu (M.A. Oxley), eric.jonardi@colostate.edu (E. Jonardi), sudeep@colostate.edu (S. Pasricha), aam@colostate.edu (A.A. Maciejewski), hj@colostate.edu (H.J. Siegel), patrick.burns@colostate.edu (P.J. Burns), koenig@ornl.gov (G.A. Koenig).

http://dx.doi.org/10.1016/j.jpdc.2017.04.015
0743-7315/© 2017 Elsevier Inc. All rights reserved.

of performance as a primary design objective to metrics such as energy efficiency and performance-per-watt as higher priority design objectives. The DGX SATURNV supercomputer currently tops the Green500 list with a performance-to-power ratio of 9.46 GFLOPS per Watt [15]. A linear extrapolation of the DGX SATURNV system to exascale results in a power consumption of 105 MW, or approximately $33 million per year in electricity in the United States. The Defense Advanced Research Projects Agency (DARPA) has set the target for an exaflop capable system to consume no more than 20 MW [26], consuming only approximately one-fifth of the power of today's most power-efficient supercomputer.

Designing new power-aware and thermal-aware resource management techniques is one method to reduce the power consumption of large-scale systems and to alleviate the amount of cooling to further reduce power consumption and total cost of ownership. Examples of power-aware and thermal-aware resource management include exploiting different power and performance characteristics of heterogeneity across compute nodes, configuring dynamic voltage and frequency scaling (DVFS) in cores, and setting computer room air conditioning (CRAC) thermostats to higher temperatures during operation. These decisions must ensure the temperatures of the compute nodes do not exceed the red-line threshold, defined as the maximum allowable equipment intake temperature [2].

The prevalence of multicore processors in modern enterprise data center facilities has given rise to heavy contention for shared resources among the cores, such as at the last-level cache and DRAM. This has led to performance degradation when running tasks on cores that share these resources, with a significant impact on the execution time of those tasks [18]. The performance degradation can range from negligible to severe, and the amount of degradation is correlated with attributes such as how many tasks are co-located (i.e., running on cores within the same processor), the memory intensity of the co-located tasks (defined as the ratio of last-level cache misses to the total number of instructions executed), and clock frequency of the core. With knowledge of these workload characteristics, it is possible to predict the execution time degradation of tasks under co-location interference effects [12]. Typically, performance degradation is more severe when memory intensive tasks are co-located, because they try to access shared memory simultaneously more often than compute intensive tasks. Therefore, to mitigate interference, it is intuitive to co-locate memory intensive tasks with compute intensive tasks.

We consider an enterprise data center with workload arrival rates that can be predicted over a decision interval (epoch) [6,10], where the task arrival rates, temperatures at compute nodes and CRAC units, and the power consumption of the computing system and CRAC units remain virtually invariant over that interval of time. That is, a day split into epochs, and over the course of an epoch the workload arrival rates can be reasonably approximated as constant, e.g., the Argonne National Lab Intrepid log that shows mostly-constant arrival rates over large intervals of time [14]. The performance of the data center is measured by the reward collected from completing tasks by their individual deadlines, where reward represents the worth of completing that task to the system. In a rate-based resource management framework, maximizing reward is equivalent to maximizing the *reward rate*. The goal of our resource management techniques is to maximize reward rate. Our techniques mitigate the impact of co-location interference by maximizing a reward rate objective function that considers co-location interference. The problem we solve is to maximize the reward rate earned by the system while obeying red-line temperature thresholds and a power constraint on the whole facility (both compute and cooling power). By taking a holistic approach to the control of such a facility, we maximize the reward rate earned while ensuring that the compute nodes do not exceed their red-line temperatures and the total power consumed by the compute nodes and CRAC units do not exceed a given power constraint. Though it is not typical for today's resource managers (e.g., SLURM [38], MOAB [30]) to consider DVFS performance states (P-states) for its allocations, and we assume an oversubscribed environment (i.e., the system is not capable of executing all incoming tasks) where it would be common to execute the workload in the fastest P-state, we believe the consideration of DVFS by our resource manager provides greater control of the power consumption of compute nodes and the intelligent selection of P-states allows for greater performance gains within that power budget.

To solve this optimization problem, we design a new greedy heuristic, develop a genetic algorithm (GA) based approach, and improve a non-linear programming (NLP) approach from our previous work [1] to consider the effect of co-locating tasks on multicore processors. We perform extensive analyses of these techniques on different workload environments, data center facility cold-aisle isolation configurations, and data center facility sizes (large and small).

In summary, we make the following novel contributions:

- Derivation of a new detailed model of a heterogeneous data center that considers the power consumption and performance of the compute nodes and cooling system, thermal constraints, DVFS, and co-location interference.
- Design of resource management techniques based on a greedy heuristic, a GA with local search, and an adaptation of a previously proposed NLP approach with consideration of co-location while ensuring that power consumption and thermal constraints are obeyed.
- In-depth simulation-based analyses of resource management techniques under several important physical facility configurations with different system sizes and cold-aisle isolation capabilities.
- Sensitivity analysis of our resource management techniques under a range of values for the power and thermal constraints across different cold-aisle isolation configurations and workload environments.

The rest of the paper is organized as follows. We discuss related work in Section 2. In Section 3, we explain our models for compute nodes, CRAC units, and workload, as well as how we consider co-location interference. Section 4 describes our proposed resource allocation techniques. Our simulation and evaluation setup and results are in Sections 5 and 6, respectively. In Section 7, we conclude and discuss ideas for future work.

2. Related work

2.1. Overview

Data centers use a large amount of power, resulting in high electricity costs and a large carbon footprint. In an attempt to encourage green data center design by reducing the power consumption in data centers, the authors of [20] have identified four factors that can drive green data center design: (1) improved physical design to reduce heat recirculation and hot-spots, (2) using highly energy-efficient computing and cooling systems, (3) using sustainable energy sources such as solar power, and (4) using intelligent computing resource management strategies. In this study, we focus on designing and analyzing green resource management techniques. Temperature, power, and performance of the compute nodes are all tightly coupled, and thus we take a holistic approach to data center resource management that considers heterogeneous compute nodes, spatial temperature knowledge, P-states, CRAC outlet temperatures, and interference caused by co-location on multicore processors.

2.2. Thermal-aware scheduling

While larger data centers such as Google's have small power usage effectiveness (PUE) and therefore only 6% of the power consumption is due to cooling and power distribution [17], a survey conducted in 2014 found an average data center PUE of 1.7 that implies approximately 40% of data center power consumption is due to cooling and power distribution [39]. Also, our collaboration with Oak Ridge National Labs (ORNL) has demonstrated that large computing systems (e.g., Titan) have significant cooling power costs. A holistic approach to management of both the cooling and computing components is vital in reducing electricity costs. The power and temperature relationship between the compute nodes and cooling infrastructure is highly correlated. That is, reducing the power of compute nodes also reduces the temperatures of those nodes, therefore easing the load on the cooling infrastructure and reducing cooling costs. However, thermal-aware and power-aware computing decisions are often conflicting. Power-aware computing techniques often attempt to concentrate the workload to a few active nodes, allowing those nodes without a workload to switch into an idle state or be deactivated. Intuitively, this leads to high heat dissipation in a small area, creating thermal hotspots and increasing the local cooling power for just a small number of the nodes. Thermal-aware scheduling techniques reduce cooling power by minimizing hotspots and total heat generated [9].

Both minimizing hotspots and total heat generated is examined in [44], where virtual machines are dynamically migrated based on temperatures and compute node loads to try to satisfy the conflicting objectives of reducing cooling costs by spreading the workload, and reducing compute node power costs by consolidating the workload. The fundamental idea of their migration algorithm is to detect overheating compute nodes (through onboard sensors), and migrate virtual machines from an overheated compute node to the coolest one in the facility. By placing a temperature cap for detecting overheating, and then migrating loads from hot nodes to the coolest ones, hotspots are managed and heat is spread more evenly throughout the room. But neither DVFS nor heat recirculation is considered, as we do.

The research in [19] takes an interesting approach in its thermal-aware workload management by considering multiple facilities, each located in different geographical locations. Each facility has intermittent and unreliable renewable energy sources attached (e.g., wind or solar), making it sensible to migrate "time insensitive" batch workloads to facilities with high renewable energy generation at a given point in time. The computing facilities also are equipped with a thermal energy storage (TES) system. The intuition behind the two kinds of thermal energy storage systems in today's computing facilities are to either (1) excessively cool the data center when an abundance of renewable energy is available so it remains at a reasonable temperature when no renewable energy is available, or (2) attach a dedicated thermal storage unit that uses renewable energy to chill liquid for use later. That paper considers the latter, and models the thermal storage unit similarly to a battery with a capacity, charge, and discharge rate. The problem is formulated as an economics problem to minimize the operating cost (energy and "bandwidth" cost for migration) to complete the workload. Unlike [19], our research considers heterogeneity of compute nodes, DVFS, and CRAC unit control in our resource management techniques.

There have been many recent works that consider thermal-aware resource management (e.g., [19,23,25,27,33,35,40,41,43,44]). Some do not consider heterogeneity in their work [19,43,44] or recirculation of heat in the room [19,33,41,44]. Others do not consider DVFS control [19,25,27,33,35,40,41,44] or CRAC unit control [19,35,41,44], leaving the thermal-aware allocation choices limited to only turning nodes on/off or basing the choices on CPU utilization. Lastly, some thermal-aware works only examine thermal and power metrics, while not considering performance metrics of any sort [44]. Today's systems typically require abiding by performance constraints to ensure either deadlines or SLAs are not violated.

We consider a similar environment and problem to solve as our previous work in [1]. The goal of [1] is to maximize the reward rate earned in a data center in a power and thermal constrained environment. We improve significantly upon that work by considering the effects of co-location interference, designing new techniques to mitigate the effects of co-location interference, and performing analyses across a broader range of different workload environments and facility configurations.

2.3. Co-location interference

Multicore processors are prominent in today's data centers, and the number of cores per processor is increasing. In such processors, many (if not all) cores share resources such as the last-level cache, causing contention and performance degradation for tasks executing on those cores because of the thrashing in shared caches or competition for main memory bandwidth. The degradation experienced is dependent on the characteristics of the co-located tasks, and ranges from negligible to severe [18]. The problem of co-location interference between tasks on multicore processors is fairly new; thus, most recent works are still related to measuring or modeling of the phenomena, with very few on the actual design of resource management techniques that use such models to mitigate interference effects. The research in [24] proposes a general methodology for measuring task interference in multicore processors, and tests it in a real data center. In [46], an empirical approach to predict performance degradation is proposed. Focusing on shared cache and memory bandwidth as the two shared resources that affect performance the most, functions that predict a task's pressure on those subsystems are found. Both linear models and neural network models are created using empirical data in [12]. A set of benchmarks from both the PARSEC benchmark suite and NAS benchmark suite are co-located on cores of multicore processors in numerous configurations to collect data for correlated features such as last-level cache misses, memory intensities, and execution times. These models allow predictions to be made for performance degradation when multiple tasks are co-located on a multicore processor.

To mitigate co-location interference among virtual machines allocated to the same multicore processor, the algorithm in [13] aims to leverage information from tasks the system has seen in the past to classify and then schedule tasks without violating QoS guarantees. In [5], it is argued that consolidation of several tasks onto a compute node (e.g., through virtualization) increases utilization but results in degrading task performance significantly. Therefore, a technique that increases node utilization while still maintaining SLAs is proposed and tested. A weighting scheme is used to determine the pairings for critical (strict SLA agreements) and non-critical workloads onto compute nodes.

Our paper does not focus on modeling co-location interference effects as [12,18,24,46] do. We focus on the design of resource management techniques that use predictive models to avoid interference effects, such as [5,13]. However, our work also considers power and temperature as constraints in the decision matrix of our resource management techniques.

3. System model

3.1. Overview

Our model of a data center and workload builds on the model proposed in [1]. We assume the facility is configured in a hot

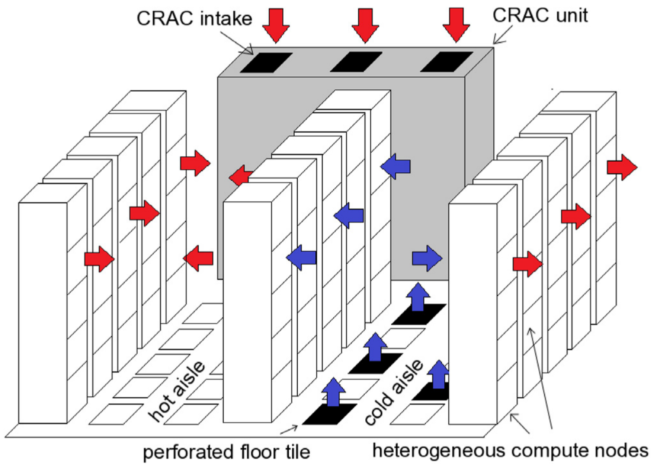


Fig. 1. Data center in hot aisle/cold aisle configuration.

aisle/cold aisle fashion (Fig. 1). In such a configuration, cold air is supplied from the CRAC units to a cold aisle through perforated floor tiles that face the inlets of the compute nodes. The compute nodes consume power and expel hot air through the opposite end to a hot aisle. The CRAC units draw the hot air from the hot aisles to cool. If the cold aisles are isolated using containment technology (e.g., using plastic curtains or commercial Plexiglass systems), the recirculation of heat among compute nodes in a data center may or may not be important to consider. Some facilities have isolated aisles (e.g., the research computing facility at Hewlett Packard in Fort Collins, CO) and some have non-isolated aisles (e.g., the data center at Colorado State University). In our work, we compare both and examine the benefits of isolation.

3.2. Compute nodes

The *number of compute nodes* in the data center is NN, and each compute node j belongs to a compute *node type* $NT(j)$. We assume a heterogeneous system with the *number of compute node-types* equal to NNT, and compute nodes that belong to a specific compute node-type are identical and contain the same number of cores, power characteristics, and performance characteristics. Cores within a compute node type are homogeneous, and we assume the cores can be independently assigned performance states (P-states) that provide a tradeoff between power and performance [21]. P-states are discrete voltage and clock frequency pairings, with lower-numbered P-states consuming more power but increasing the execution speed of tasks. We model a case where a core is deactivated by adding one additional P-state to the available P-states of a core, where the deactivated state is the highest-numbered P-state. The total *number of cores* in the data center is NC, and $CT(k)$ is the *type of the compute node to which core* k belongs. If core k is in node j, then $CT(k) = NT(j)$. We provide a table of notations (Table 14) in the appendices.

3.3. Workload

We assume that we have a set of T known *task types*. We consider a data center in the steady-state where it is assumed that for a given epoch interval, the arrival rates of tasks of type i, denoted λ_i are known (or can be predicted [6,10]. A *reward* r_i is obtained for completing a task of type i by its individual *deadline* d_i, relative to its arrival time.

The system we consider is heterogeneous, i.e., the power and performance characteristics of the compute node-types are different. Therefore, tasks of the same type can have different execution rates on different node-types and P-states. We assume that we know the *estimated computational speed* (ECS) of any task of type i on a core of type j in P-state k, ECS(i, j, k) (measured in number of tasks per second). In many computing environments, such as those in national labs (e.g., NCAR, ORNL) and industry (e.g., DigitalGlobe), the tasks executed are from a known set of tasks and therefore their arrival rates can be predicted and their execution times pre-characterized using historical, experimental, or analytical techniques [22,28,37]; in Section 5, we discuss the values used for our simulations using benchmark data collected from server class machines.

We assign a *desired fraction* of time each core k will spend executing tasks of type i, denoted $DF(i, k)$, and the *P- state* each core k is configured when executing tasks of type i, denoted $PS(i, k)$. We assume tasks will run serially until completion. That is, a core sharing its time among multiple tasks implies that an online scheduler using information from solutions provided by our rate-based resource manager will assign different tasks to execute on the core in such a manner that, over a long period of time (i.e., steady-state), the amount of time a core k spends executing a task of type i would equal its assigned $DF(i, k)$ value. Our goal is to maximize the reward we can obtain and meet the power and thermal constraints of the system. Given $DF(i, k)$ and $PS(i, k)$, we calculate the *execution rate* of tasks of type i on core k, $ER(i, k)$, as

$$ER(i, k) = DF(i, k) \cdot ECS(i, CT(k), PS(i, k)). \qquad (1)$$

3.4. Power model

In an oversubscribed environment such as the one we consider, non-CPU components are always active and thus their power dissipation can be assumed to be static (i.e., not time-varying). We consider the static power consumption of a compute node (e.g., from main memory, disks, fans) in addition to the dynamic power consumption of the CPU cores. We assume that CPU cores are able to change P-states over time depending on what task-type is currently being executed, and that the time associated with switching P-states is negligible in comparison to the execution time of tasks. The power consumed by cores is a function of the task-type being executed and the P-state in which the core is executing the task. Let $S(j)$ be the *static power consumption* of compute node j, let APC$(i, NT(j), PS(i, k))$ be the *average power consumed* by a core k in a node of type $NT(j)$ executing tasks of type i in P-state $PS(i, k)$, and $NCN(j)$ be the *set of cores in node j*. We calculate the power consumption of node j, $PN(j)$, as

$$PN(j) = S(j) + \sum_{k \in NCN(j)} \sum_{i=1}^{T} APC(i, NT(j), PS(i, k)) \cdot DF(i, k). \qquad (2)$$

We assume a constant static power $S(j)$, but in cases where a core may not be active 100% of the time (i.e., the sum of $DF(i, k)$ values across all T task types is less than 100%), that core would not consume its full power according to Eq. (2).

The power consumed at a CRAC unit is a function of the heat removed at that CRAC unit in addition to the Coefficient of Performance (CoP) of the CRAC unit [31]. Let NCR be the total *number of CRAC units* in the data center, $TC^{in}(i)$ be the *inlet temperature of CRAC unit* i, $TC^{out}(i)$ be the *outlet temperature of CRAC unit* i, ρ be the *density of air*, C be the *specific heat capacity of air*, and $AFC(i)$ be the *air flow rate of CRAC unit* i. The *power consumed by CRAC unit* i, $PC(i)$, is calculated as [31]

$$PC(i) = \frac{\rho \cdot C \cdot AFC(i) \cdot (TC^{in}(i) - TC^{out}(i))}{CoP(TC^{out}(i))}. \qquad (3)$$

3.5. Thermal model

To calculate the temperatures at compute nodes and CRAC units, we use the concept of thermal influence indices that characterizes the causal relationship between heat sources and sinks from [4]. That is, the thermal influence indices are used to help estimate the recirculation of air between compute nodes and CRAC units, similar to the recirculation models introduced in [31] and [42]. Let $TN^{in}(j)$ be the *inlet temperature at compute node j* and $TN^{out}(j)$ be the *outlet temperature at compute node j*. The outlet temperature at compute node j is a function of the inlet temperature, the power consumed, and the *air flow rate of the node AFN*(j), calculated as

$$TN^{out}(j) = TN^{in}(j) + PN(j)/(\rho \cdot C \cdot AFN(j)) . \tag{4}$$

Let $\mathbf{TC^{out}}$ and $\mathbf{TC^{in}}$ be the vectors of outlet and inlet temperatures of CRAC units. Also, let $\mathbf{A^{CRAC}}$ be a matrix of thermal influence indices where each element $\mathbf{A^{CRAC}}[\mathbf{i}, \mathbf{y}]$ represents the percentage of heat transferred from CRAC unit i to CRAC unit or node y, and let $\mathbf{A^{Node}}$ be a matrix of indices where each element $\mathbf{A^{Node}}[\mathbf{j}, \mathbf{y}]$ represents the percentage of heat transferred from node j to CRAC unit or node y. The $\mathbf{A^{CRAC}}$ matrix intuitively represents how much of the heat generated by a CRAC unit (a row of $\mathbf{A^{CRAC}}$) is transferred to a CRAC unit or node (a column of $\mathbf{A^{CRAC}}$). Similarly, the $\mathbf{A^{Node}}$ matrix intuitively represents how much of the heat generated by a node (a row of $\mathbf{A^{Node}}$) is transferred to a CRAC unit or node (a column of $\mathbf{A^{Node}}$). We derive the thermal influence indices from computational fluid dynamics (CFD) simulations using the physical layout of the data center we study (see Section 5). We can then calculate the inlet temperatures of any CRAC unit or node y as [4]

$$T^{in}(y) = \sum_{i=1}^{NCR} \mathbf{A^{CRAC}}[\mathbf{i}, \mathbf{y}] \cdot TC^{out}(i) + \sum_{i=j}^{NN} \mathbf{A^{Node}}[\mathbf{j}, \mathbf{y}] \cdot TN^{out}(j). \tag{5}$$

The ASHRAE guidelines have designated the inlets of IT equipment as the common measurement point for temperature compliance [2], and therefore we consider thermal constraints at the inlets of compute nodes. If we let $\mathbf{T^{redline}}$ be the vector of red-line temperatures, the thermal constraint is the element-wise inequality

$$\mathbf{T^{in}} \le \mathbf{T^{redline}}. \tag{6}$$

3.6. Co-location interference

Tasks competing for shared memory in multicore processors can cause severe performance degradation, especially when competing tasks are memory intensive [18]. Analysis of performance counters in computing systems (e.g., as in [11]) when executing a task allows for determination of the memory intensity of that task. The memory intensity of a task refers to the ratio of last-level cache misses to the total number of instructions executed [12]. We employ a linear regression model from [12] that combines a set of disparate features (i.e., inputs that are correlated with task execution time) based on the current tasks assigned to a multicore processor to predict the execution time of a target task i on a target core k. A description of the features we use, in addition to the symbols representing each feature, is given in Table 1. In a linear model, the output is a linear combination of all features and their calculated coefficients, where the coefficients are used to combine the disparate features together into one measure. We classify the task-types into memory intensity classes to calculate these coefficients for the different memory intensity classes using the linear regression model. If we denote u, v, w, x, and y as the coefficients for feature symbols $A, B, C, D,$ and E (from Table 1) for a task-type of memory intensity class m on core k, then with z as the constant, the equation for co-located execution time of a task-type i on core k ($CET(i, k)$) is

$$\begin{aligned} CET(i, k) = {} & u \cdot A(k) + v \cdot B(i, k) + w \cdot C(i, k) \\ & + x \cdot D(k) + y \cdot E(i, k) + z. \end{aligned} \tag{7}$$

The coefficients of the model for the various features in Eq. (7) (e.g., u, v, w) were calculated using the linear regression model [12] for task types of memory intensity class m on a core k. The execution rate is the reciprocal of the execution time. Therefore, the co-located execution rate for task-type i on core k, $CER(i, k)$, is $1/CET(i, k)$. To assist our resource management techniques in estimating actual reward rate under the effects of co-location interference, we introduce an objective function called *reward rate*, calculated as

$$RR = \sum_{i=1}^{T} \left(r_i \cdot min \left(\sum_{k=1}^{NC} CER(i, k), \lambda_i \right) \right). \tag{8}$$

Eq. (8) states that the reward rate for a given task-type i is the product of the reward earned and its co-located execution rate. The *min* function in Eq. (8) enforces the constraint that if a task is assigned for execution at a faster rate than its arrival rate λ_i, additional reward is not earned. One goal of our proposed resource management techniques is to maximize the *RR* objective function for a given epoch in the rate-based problem that we study. Using Eq. (8) allows our techniques to evaluate co-location interference effects on reward rate earned over an epoch interval.

We also present an objective named *naïve estimated reward rate*, denoted *NERR*. *NERR* uses the execution rate (*ER*) rather than co-located execution rate (*CER*) values, and is therefore naïve to co-located interference effects. *NERR* is a false reward rate, however, because co-location interference exists. We use *NERR* as an objective in co-location unaware versions of the techniques we propose to demonstrate the value of considering co-location interference effects. *NERR* is calculated as

$$NERR = \sum_{i=1}^{T} \left(r_i \cdot min \left(\sum_{k=1}^{NC} ER(i, k), \lambda_i \right) \right). \tag{9}$$

Reward is only earned when tasks are completed by their deadline. We have no way of guaranteeing deadlines are met in our rate-based model, but we can help minimize the number of deadlines that would be missed by eliminating any task-type/core-type/P-state combinations from consideration that would result in a missed deadline even if a task of type i starts immediately after its arrival, i.e., we eliminate combinations from consideration that violate

$$\frac{1}{ECS(i, CT(k), PS(i, k))} \ge d_i. \tag{10}$$

The goal of this study is to maximize true reward rate when subject to power consumption and thermal constraints. *RR* is a better estimate of reward rate than *NERR* because co-location interference exists. The total power consumed by both CRAC units and compute nodes must be less than the power constraint, denoted as ϕ. The thermal constraint is defined in Eq. (6). In the next section, we describe several approaches to solve this problem.

4. Heuristics

4.1. Non-linear programming approach

We adapt a power and thermal-aware approach from [1], and improve it to include the effects of co-location by using our *RR* objective to estimate the reward rate instead of *NERR*. The problem in [1] is formulated as a non-linear program and then solved using

ARTICLE IN PRESS

Table 1
Description of features for predicting execution time.

Name	Symbol	Description
Number of co-located tasks	$A(k)$	The number of tasks co-located with target task on core k
Base execution time	$B(i, k)$	Base execution time of target task i when executing alone on core k in P-state $PS(i, k)$
Frequency of target core	$C(i, k)$	Clock frequency of target core k when running target task i in P-state $PS(i, k)$
Average memory intensity	$D(k)$	Average memory intensity of all tasks on the multicore processor that contains core k
Target memory intensity	$E(i, k)$	Memory intensity of target task i on core k

approximations, heuristics, and linear programming. The complexity of common interior point algorithms (i.e., the type of algorithms used in many non-linear programming solvers) is $\mathcal{O}(n^{3.5}L)$, where n is the number of variables and L is the input length of the problem [32]. We refer the reader to [1] for the full details of this technique, but give the problem formulation in Appendix A in addition to a brief summary below.

The NLP technique is divided into three steps to solve the formulated mixed integer NLP for maximizing naïve estimated reward rate (using Eq. (9)) while obeying the power and thermal constraints. (1) The first step relaxes the integer constraint on P-states (by assuming continuous P-states) and solves for the CRAC outlet temperatures and power consumption of compute cores such that reward rate is maximized and the power and thermal constraints are met. (2) Using the core power consumption values obtained from the first step, the second step uses a simple heuristic to round the continuous power consumption values of cores to discrete P-states, while maintaining the power and thermal constraints. (3) Lastly, a linear program is solved to find the desired fraction of time values that every core spends executing a given task so that reward rate is maximized, assuming the CRAC outlet temperatures from the first step and the discrete P-state assignments from the second step. A non-feasible solution can sometimes be returned by the NLP if the power or thermal constraints are set to extreme values. In that case, the local search technique from the GA (Alg. 4) is performed on the resulting solution to set the CRAC outlet temperatures such that the thermal constraint is met (but possibly not the power constraint).

We improve upon this approach in two ways. First, we incorporate the knowledge of the memory intensity of tasks by considering the APC matrix (because a core consumes a different amount of power based on task-type) instead of assuming all task-types consume the same amount of power in a given P-state. Second, we incorporate knowledge of co-location interference in this algorithm by maximizing our *RR* measure, which is a better estimate of reward rate than *NERR*. Pseudo-code for our NLP technique is given in Alg. 1.

Algorithm 1 Pseudo-code for our NLP technique

```
1. Step 1:
2.     relax integer constraints on P-states
3.     solve for CRAC outlet temperatures and power consumption of
        cores to maximize $RR$ and obey power/thermal constraints
4. Step 2:
5.     use the solved continuous power consumption
        of cores to round to discrete P-states
6. Step 3:
7.     solve linear program to determine $DF(i, k)$ values, assuming the CRAC
        outlet temperatures determined in Step 1 and P-states found in
        Step 2
8.     if power/thermal constraints violated, apply local search from GA
```

4.2. Greedy heuristic

We designed a two-phase greedy approach similar in concept to "Min–min" in [7,29] to assign task-types to cores. For this heuristic, cores are dedicated to a single particular task-type (i.e., 100% of the core's time is dedicated to tasks of that type). Our greedy heuristic (see Alg. 2) iteratively assigns task-types to cores to find the most efficient mapping, where we define efficiency for a task mapping of type i on a node of type j in P-state k, $EFF\,(i, j, k)$, as

$$EFF(i, j, k) = ECS(i, j, k)/APC(i, j, k). \qquad (11)$$

We start by finding the P-states with the highest EFF values for all task-types and node-types (line 1); all other P-states are not considered. All $T \times NNT$ task-type to node-type pairings are then sorted by their efficiency in descending order (line 2). At each iteration of the heuristic, the first pairing (i.e., most efficient) is selected and the heuristic checks if at least one core within the chosen node-type remains unmapped. If so, that core is assigned a 100% desired fraction of time for that task-type (line 6). After making an assignment, that core is removed from consideration (line 7). If the new mapping results in the execution rate of the task-type exceeding its arrival rate (line 8), that core is unmapped (line 9), returned to the pool of available cores (line 10), and that task-type/node-type pair is removed from future consideration (line 11). The heuristic uses the CER value to estimate execution rate (line 8), thus making the decision to provision task-type execution rates as long as the CER value does not exceed the arrival rate. The CRAC outlet temperatures are set to the red-line temperature, and then the outlet temperatures of all CRAC units are iteratively decreased by one degree until the thermal constraints are met (line 12). If no unmapped cores within the selected node-type exist, the current task-type/node-type pair is removed from consideration (line 14). The algorithm repeats using the next pairing until the power constraint is violated, or there are no more task-type/node-type pairings to consider (line 3).

4.3. Genetic algorithm

4.3.1. Overview

We also designed a GA to solve our resource allocation optimization problem (e.g., as in [7]), based on the Genitor GA [45]. Our GA in this study operates on a population of 200 chromosomes (empirically determined). Each chromosome represents one possible solution, i.e., a complete resource allocation. Each chromosome consists of a matrix of $T \times NC$ genes, where each gene is a pair of $DF(i, k)$ and $PS(i, k)$ values representing the desired fraction of time and P-state of a task-type/core combination. The initial population is generated by assigning random desired fractions of time and P-states to each gene within the chromosome, and then normalizing the desired fractions of time so that cores cannot spend greater than 100% of their time executing tasks. That is, if a core k is spending greater than 100% of the time executing tasks, we normalize the $DF(i, k)$ values on that core. Normalization is

ARTICLE IN PRESS

Algorithm 2 Pseudo-code for our greedy heuristic

```
1. get most efficient P-state for each task-type/node-type pair
2. sort these task-type/node-type pairs by efficiency
3. while power constraint not violated and at
   least one task-type/node-type pair remains
4.     choose first task-type/node-type pair
5.     if unmapped core within selected node-type exists
6.         assign 100% desired fraction of time for selected
           task-type to a core from selected node-type
7.         remove core from future consideration (mapped)
8.         if execution rate of task-type exceeds its arrival rate
9.             unmap core running that task-type
10.            return unmapped core to pool of available cores
11.            remove task-type/node-type pair
               from future consideration
12.        set CRAC outlet temperatures to hottest temperatures
           such that thermal constraints are met
13.    else
14.        remove task-type/node-type pair from consideration
15. end while
```

Algorithm 3 Pseudo-code for our GA heuristic

```
1. create an initial population of chromosomes
2. evaluate and rank population
3. while time limit (e.g., epoch interval) not surpassed
4.     perform crossover to generate two offspring chromosomes
5.     perform mutation on offspring chromosomes
6.     normalize $DF(i, k)$ values in offspring chromosomes
7.     perform local search on offspring to meet constraints
8.     evaluate and rank population
9.     trim population to original size
10. end while
11. return best chromosome from population
```

performed by summing the desired fractions of time all task-types spend executing on a given core k, denoted $SumDF(k)$, and dividing the desired fraction of time values for all T task-types by $SumDF(k)$, forcing their sum to equal 100%. Normalization is not performed on a core if the value of $SumDF(k)$ is less than 100%.

One chromosome of the initial population is from the greedy heuristic to seed the GA and assist the GA by providing better genetic material than randomly generated chromosomes. After the initial population generation, the chromosomes in the population are evaluated and ranked by reward rate (RR). We then perform crossover and mutation that alter existing solutions to generate offspring chromosomes. After the offspring are generated, local search is performed on the offspring to meet the power consumption and thermal constraints. The population is then evaluated and ranked, and subsequently the population is trimmed to its original size by eliminating the least-fit chromosomes. By using RR instead of $NERR$ to evaluate chromosomes, we consider co-location interference effects by effectively penalizing solutions that have low reward rate due to co-location interference. The pseudo-code for our GA heuristic is provided in Alg. 3.

4.3.2. *Crossover and mutation*

Crossover starts by selecting two parent chromosomes using a linear bias [45] and generates two points, x and y, such that $x < y \leq NC$. All genes for cores ranging from x to y are swapped among parent chromosomes to create two offspring solutions. This operation spans over all task-types for the cores ranging from x to y.

Mutation is probabilistically performed on offspring chromosomes to introduce perturbations in the genes, allowing a broader search. Offspring chromosomes have a probability p_m of being mutated (empirically set to 0.1). If a chromosome is selected for mutation, each gene has a probability p_g of being mutated (empirically set to 0.05). If a gene is selected for mutation, the desired fraction of time and P-state for its task-type on this core are set to random values. Then, the chromosome is normalized so that the desired fractions of time for all cores sum to 100%. Unlike the greedy heuristic that dedicates a core to one task-type (i.e., 100% desired fraction to one task-type), the GA can have cores assigned to execute multiple task-types.

4.3.3. *Local search*

We perform a local search on (possibly mutated) offspring chromosomes that sets CRAC outlet temperatures and P-states such that the power consumption and thermal constraints are met. The pseudo-code for our local search is given in Alg. 4. The step described in line 3 is performed by setting the CRAC outlet temperatures to the red-line temperature, and then the outlet temperatures of all CRAC units are iteratively decreased by one degree until the thermal constraints of the nodes are met. The steps performed in lines 4 to 7 have a two-fold effect to reduce power consumption. First, increasing the number of the P-state assignment (i.e., reducing the power and frequency) directly reduces the power consumption of the hottest node. Second, the CRAC units may be able to run with a higher outlet temperature, reducing the overall power required to maintain red-line temperatures. In our environment, our local search typically ensures that the power and thermal constraints are met, but the resource allocation may have a poor reward rate. If the power and thermal constraints are difficult to meet, there is a limit to how long the search proceeds (*maxIter*) before allowing the GA to continue. We set the value of *maxIter* to 2000 in this study. We examine the effects of our resource management techniques under different power and thermal constraint values in Section 6.4.

5. Evaluation setup

5.1. *Overview*

In this section, we provide the simulation parameters we used for the platform configuration and workload. Many of our simulation parameters (e.g., power values, workload execution times, coefficients for the co-location interference model) were obtained from executing benchmarks on actual servers in our lab. These values can be found in Appendix B (for example, data representing the power consumption of a core when executing PARSEC benchmarks on our lab servers at the **lowest** frequency P-state can be found in Table 11). Also, error values for the thermal and co-location execution time models can be found in Appendix B.

5.2. *Platform*

In our simulations, we consider two heterogeneous platforms of different sizes. The *small* platform (see Fig. 2(a)) consists of one CRAC unit, and 30 compute node cabinets of size 42U with 36 nodes per cabinet (typically some cabinet slots are left empty, e.g., due to power and cooling requirements). This gives a total

Algorithm 4 Pseudo-code for our local search technique

```
1. repeat for maxIter or until power and thermal constraints met
2.     set CRAC outlet temperatures to hottest temperatures
       such that thermal constraints are met
3.     find node with highest temperature (node $j$)
4.     while node $j$ is hottest node and not all core/task-type
       combinations in node $j$ are assigned maximum numbered P-state
5.         choose random core/task-type
           combination from node $j$
6.         increase P-state assignment by 1
           if not already in maximum P-state
7.     end while
```

ARTICLE IN PRESS

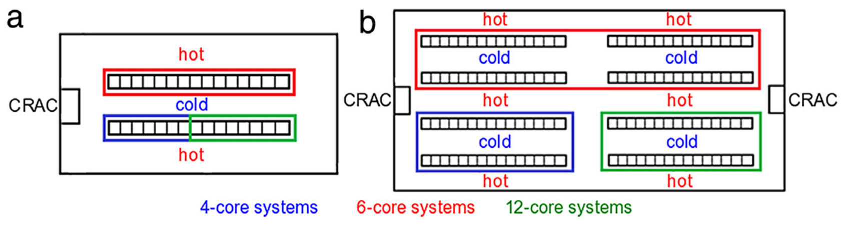


Fig. 2. (a) Small data center platform configuration, and (b) large data center platform configuration.

of 1080 compute nodes, where each node is one of three node types (see Table 2). The *large* platform (see Fig. 2(b)) consists of two CRAC units, and 120 compute node cabinets with 36 nodes per cabinet (a total of 4320 compute nodes), where each node is one of the three node types. These platforms represented the physical layout of the facility we considered when performing CFD simulations to calculate the thermal influence matrices ($\mathbf{A}^{\mathbf{CRAC}}$ and $\mathbf{A}^{\mathbf{Node}}$). The evaluation of the accuracy for the technique used to calculate the $\mathbf{A}^{\mathbf{CRAC}}$ and $\mathbf{A}^{\mathbf{Node}}$ matrices was performed in [4]. The power characteristics of our different node-types were obtained from power measurements of three server class machines when executing different PARSEC benchmarks across all P-states (see Table 2), and the workload characteristics were also obtained from executing the benchmarks on the node-types listed in Table 2. For a given P-state on a given server type, a power and execution time measurement was obtained by executing the desired PARSEC benchmark serially on a core. The power constraint was set to 900 kW for the large platform, and 230 kW for the small platform. The red-line temperature was set to 30°C, which is on the high end of ASHRAE's temperature guidelines [2].

In this study, we assume that the CRAC units are homogeneous. The CoP for a CRAC unit is a function of its outlet temperature, τ, given by $CoP(\tau) = 0.0068\tau^2 + 0.0008\tau + 0.458$ [31]. We used that particular CoP function for simulations, but it could be easily changed depending on the particular CRAC unit considered. The air flow rate of each CRAC unit is set to 26.1 m^3/s, which is the air flow rate of a CRAC unit with capacity to cool approximately 350 kW [3].

5.3. Workload

The task-types are from the PARSEC benchmark suite [36], and corresponding ECS matrix entries for each benchmark on each node-type in each P-state are obtained from taking the reciprocals of the execution times of those benchmarks on each of the machines and P-states. We chose to represent the workload in this study based on the PARSEC benchmark suite because of the diverse set of tasks it offers. For example, benchmarks we consider are associated with fluid dynamics, option pricing, body tracking of a person, simulated annealing optimization, and pricing a portfolio of swap options. In all, the following benchmarks were used: *canneal*, *cg*, *ua*, *sp*, *lu*, *fluidanimate*, *blackscholes*, *bodytrack*, *ep*, and *swaptions*. With regards to their memory intensity class, *canneal*, *cg*, and *ua* were classified as "heavy", *sp*, *lu*, and *fluidanimate* were classified as "medium", *blackscholes* and *bodytrack* were classified as "light", and *ep* and *swaptions* were classified as "very-light".

We split the benchmarks into three workload environments based on their memory intensities to be representative of three different types of real-world computing environments. The *memory intensive* workload consists of *canneal*, *cg*, *ua*, *sp*, and *lu* to represent a database style workload. The *hybrid* workload includes *ua*, *sp*, *lu*, *fluidanimate*, and *blackscholes* to form an all-purpose group of both CPU and memory intensive tasks. Finally, the *CPU intensive* environment is formed of *fluidanimate*, *blackscholes*, *bodytrack*, *ep*, and *swaptions* to emulate scientific computing.

The reward for task types are generated based on a uniform random variable in the range [0.5, 1]. The d_i and arrival rate λ_i values are generated using the same techniques as in [1].

6. Simulation results

6.1. Overview

A primary contribution of this research is to provide extensive analyses of our co-location interference and thermal-aware techniques for maximizing reward under a range of physical facility configurations and workload environments. Large-scale computing systems fall into several different categories that execute a diverse set of workloads, making it important to simulate a variety of environments so a system administrator can choose resource management techniques that are most applicable to a specific environment. For example, it is common in supercomputing environments to run scientific tasks that are compute intensive, with less co-location interference effects and therefore less of a need to consider them in resource management. However, in a data processing environment the tasks access memory often (memory intensive), and ignoring co-location interference effects can cause severe performance issues. The analyses we present are also useful for physical data center facility design (compute node placement optimization), where researchers can determine potential hotspots and optimal use of space for node and CRAC placement with an "average" workload.

The bar graphs discussed in the rest of this section represent the averages of 48 trials, with each trial varying in the deadline, reward, and arrival rate values of the task-types. The ECS and power values are obtained from experimental data of the PARSEC benchmarks (see Section 5.3) on three heterogeneous server class machines. The error bars are the 95% confidence intervals around the mean of those 48 trials.

6.2. Workload experiments

In our first experiment, we examine the effects that the different workload environments have on the performance of our resource management techniques. Fig. 3 shows a comparison of the greedy, GA, and NLP techniques across the three different workload environments: *CPU intensive*, *hybrid*, and *memory intensive*. In this figure, we also compare each of the co-location aware techniques (non-hashed bars) with versions that are co-location unaware (hashed bars). That is, we compare the GA and NLP techniques when using either the naïve estimated reward rate (*NERR*) or reward rate (*RR*) as their objective functions, and greedy uses either the *ER* values or *CER* values to estimate execution rates (line 8 of Alg. 2). The NLP technique that uses *RR* as its objective (co-location unaware) is similar to [1]. The reward rate is normalized by the total reward rate that the system is capable of obtaining, i.e., when all task-types are allocated to execute at rates equal to their respective arrival rates. To make a fair comparison between the NLP and GA techniques, the GA was terminated at the same time as it took the NLP to finish (approximately four hours). All techniques were able to meet the power and thermal constraints.

We can see in Fig. 3(a) that for all workload types, the NLP technique works best, GA is second best, and greedy last. Though none of the proposed techniques are guaranteed to find the global

ARTICLE IN PRESS

Table 2
Node-types used in simulations.

Node-type	Lenovo TS140	HP Z600	HP Z820
Processor (Xeon)	E3-1225v3	E5649	E5-2697v2
Architecture	22 nm	32 nm	22 nm
Number of cores	4	6	12
Number of P-states	16	9	16
Case fans	2 x 80 mm	2 x 92 mm, 1 x 80 mm	3 x 92 mm
Air flow rate (m^3/s)	0.0284	0.0519	0.0566

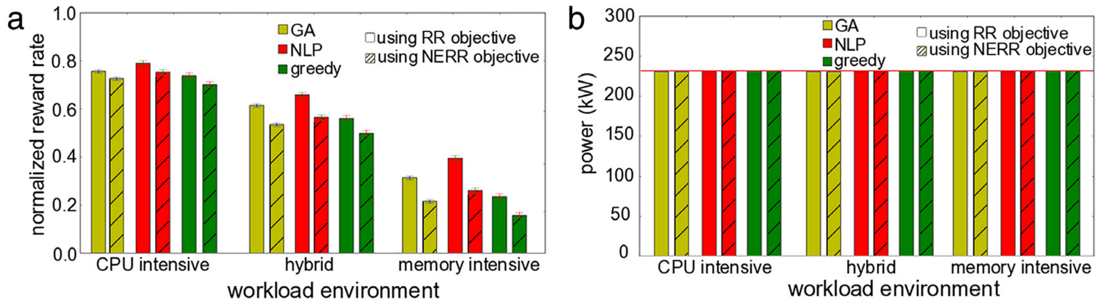


Fig. 3. (a) Normalized reward rate comparison between different workload environments on the 1080 node system, and (b) power consumption comparison in relation to power budget constraint (red line). Solid bars represent the GA, NLP, or greedy technique when using *RR* as the objective (co-location aware), and the hashed bars represent those techniques when using *NERR* as the objective (co-location unaware). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

optimum (within reasonable time limits), the NLP technique is guaranteed to find a local optimum that is better than the solutions found by the GA and greedy techniques. We can also see, within the same power budget (demonstrated in Fig. 3(b)), that all techniques earn significantly less reward as the memory intensity of the workload increases (moving from CPU intensive to hybrid, and hybrid to memory intensive). Intuitively, this makes sense as the co-location interference effects become exacerbated as the memory intensity increases. The NLP technique achieves approximately half of the reward on the memory intensive workload environment than on the CPU intensive environment. Comparing the co-location interference aware (*RR*) with those that are unaware (*NERR*), we can see the benefits of considering co-location interference, and those benefits become more significant as the memory intensity of the workload increases. Fig. 3(b) demonstrates that all techniques were able to meet the power budget exactly, and in conjunction with Fig. 3(a), we can observe that the more sophisticated techniques are able to achieve more reward rate within that power budget.

Another observation to note is the difference in performance of algorithms within a given workload type. For example, we can see in Fig. 3(a) that the dropoff in performance between the NLP and greedy techniques for the CPU intensive workload is much less than for the memory intensive workload, because it becomes more important to intelligently mitigate co-location interference effects as the memory intensity of the workload increases. Depending on the requirements of the system administrator and the mapping event interval (epoch length), it may be more worthwhile to employ a simple greedy heuristic when the workload environment experiences little co-location interference because it is much faster to make decisions than the GA and NLP. The performance difference between the NLP, GA, and greedy heuristics become more pronounced and extremely significant when executing the memory intensive workload environment, with the greedy performing half as well as the NLP in that case. In summary, for the CPU intensive workload, NLP outperforms GA by 4% and GA outperforms the greedy heuristic by 3% when they are considering the same objective function (*RR* or *NERR*). However, when the workload is memory intensive, NLP earns 30% more reward rate than the GA when using *RR* as the objective, and 24% more reward rate than GA when using *NERR* as the objective. Similarly, for the CPU intensive workload, GA is able to achieve 1% more reward rate than the greedy heuristic when using *RR* as the objective, and 1% more reward rate than greedy when using *NERR* as the objective. For the memory intensive workload, the GA is able to earn 29% more reward rate than the greedy heuristic when using *RR* as the objective, and 23% more reward rate than the greedy heuristic when using *NERR* as the objective.

6.3. Isolation experiments

The isolation of the cold-aisles in a data center provides many benefits at a low upfront cost to purchase the isolation system. Isolating cold aisles can be done using several different methods (e.g., hanging plastic curtains, installing custom Plexiglass walls and ceiling), all with the same goal of minimizing the heat recirculation among nodes. Isolating the cold aisles has previously been shown to have the intuitive result of significantly reducing node inlet temperatures when CRACs are set to the same outlet temperatures or higher [16], because the nodes' inlets are located in the cold aisle and experience little heat recirculation from the hot aisles. In our thermal-aware resource management problem, reducing the inlet temperatures of the compute nodes allows the CRAC units to run at higher temperatures, reducing the cooling power required and allowing more power budget to be allocated to the compute nodes for executing tasks. Because the cold aisles are isolated, the node inlets do not experience a significant amount of the extra heat that is generated due to the greater amount of power being used by the compute nodes.

To quantify the impact of isolation, we experiment with our resource management techniques after calculating an additional set of the thermal influence indices (i.e., the $\mathbf{A}^{\mathbf{CRAC}}[\mathbf{i},\mathbf{y}]$ and $\mathbf{A}^{\mathbf{Node}}[\mathbf{j},\mathbf{y}]$ values). These coefficients were calculated by placing isolation curtains around the cold aisles in the mesh of the computational fluid dynamic (CFD) simulations. Fig. 4 shows a comparison between using the isolated and non-isolated aisles for the CPU intensive and memory intensive workload environments. In Fig. 4(a), the normalized reward rate is shown, and in Fig. 4(b), the power consumption is shown, with hashed portion of the bars showing the power consumed by the CRAC unit and the solid portion of the bars showing the power consumed by the compute nodes.

ARTICLE IN PRESS

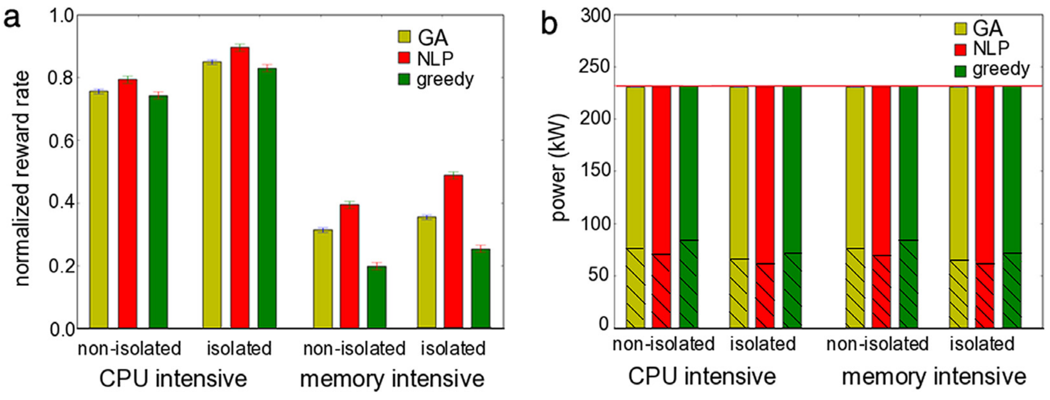


Fig. 4. (a) Normalized reward rate comparison between isolated and non-isolated cold aisle configurations on the 1080 node system, and (b) power consumption comparison in relation to power budget constraint (red line). Hashed portion of bars show power consumed by the CRAC unit, and solid portion of bars show the power consumed by the compute nodes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In Fig. 4(a), we observe that the reward earned is more in an isolated configuration for all techniques, following similar trends as the non-isolated configuration. Fig. 4(b) gives the reason why the isolated configuration can earn more reward: under the same power constraint, the techniques when using the isolated configuration are able to use less power for cooling (hashed portion of the bars) and more power for computing (solid portion of the bars). In summary, for the CPU intensive workload, isolation of the aisles provides all techniques with the ability to achieve approximately 12% greater reward rate, and approximately 20% greater reward rate for the memory intensive workload. With the cold aisles isolated, the NLP is to obtain approximately 5% more reward rate than the GA, and the GA earns 3% more reward rate than the greedy heuristic when considering the CPU intensive workload. For the memory intensive workload, NLP outperforms GA by 28% and the GA outperforms greedy by about 52%. If the reward earned is analogous to some form of revenue (e.g., in a cloud computing environment), the initial cost of isolating the aisles could be worth the price over time.

6.4. *Power and thermal constraint sensitivity analysis*

Our third set of experiments examines the effects of varying the power and thermal constraints on the normalized reward that can be earned in both isolated and non-isolated configurations. The values of both of these constraints have a significant effect on the ratio of power consumed by cooling and computing, and the reward that can be earned by the system. Such studies are valuable to the system administrator or data center owner to answer "what-if" questions, such as "what-if" the system is allowed 20% more power, or "what-if" the maximum allowable (red line) temperature is reduced to increase reliability?

The results of our temperature and power constraint sensitivity analysis are shown in Figs. 5 and 6. Fig. 5 shows the normalized reward rate and associated total power consumption results of our resource management techniques when varying the thermal constraint (red-line temperature) from 24°C to 32°C with the power constraint set to 230 kW. Fig. 6 shows the normalized reward rate and total power consumption of our resource management techniques when varying the power budget constraint from 200 kW to 300 kW in 10 kW increments with the thermal constraint set to 30°C.

Note in Fig. 5 that if the red-line temperature becomes difficult to meet (less than 26°C for the isolated configuration and less than 28°C for the non-isolated configuration), none of the techniques are able to meet the power constraint, and also earn no reward. This is due to the thermal-aware aspects of the techniques, where the CRAC outlet temperatures are set such that the thermal constraints are met. When the CRAC outlet temperatures are set low for the compute node temperatures to be under red line, the power consumption of the CRAC units becomes very high (see Eq. (3)), and the system is unable to meet the power budget even with most (if not all) of the compute cores deactivated (either assigned zero desired fraction of time, or running in the highest-numbered P-state). With all cores deactivated, a compute node still consumes static power ($S(j)$) and would still require cooling. The amount of reward earned decreases non-linearly with the red-line temperature threshold. This is because when the CRAC units have to run at a cooler temperature to ensure that red-line temperatures are not violated, a quadratically greater amount of the power budget must be used for cooling. This non-linear relationship between a CRAC unit's outlet temperature and its power consumption is captured by the quadratic term in the CoP equation (see Section 5.2). The non-isolated configuration shows the techniques earning less reward, and also violating the power constraint at lower red-line temperature values, but overall similar trends as the isolated configuration. However, in the isolated configuration, we see that the greedy technique does not use all available power when the red-line threshold is set to 32°C. This is because assigning all core/task-type combinations to execute in their most power-efficient P-state, in combination with the small amount of cooling power required to maintain a 32°C threshold, does not use all of the allotted power budget. The GA and NLP are able to assign core/task-types to execute in faster P-states to consume that additional power.

Fig. 6 shows that all techniques are effective at using all of the power budget and not violating the power budget. The reward earned varies almost linearly with the power budget, and the relationship between power budget and reward rate is not as exaggerated as in Fig. 5. Giving the system a greater power budget to operate within does result in more reward rate earned, because nodes can operate in faster P-states and execute more tasks. However, as evidenced by Fig. 5, it may be more worthwhile to operate the compute nodes and CRAC units at a slightly higher temperature to realize significant gains in reward rate earned. If reward rate is analogous to a revenue generated for computing, analyses such as these would give important insights into the most profitable operating point.

6.5. *Scalability analysis*

As data centers become larger, it is important to address the scalability of resource allocation techniques on larger system complexities. Figs. 3 and 4 showed that NLP was able to achieve the best results for a 1080 node platform size across three different workload environments and both non-isolated and isolated cold-aisle

ARTICLE IN PRESS

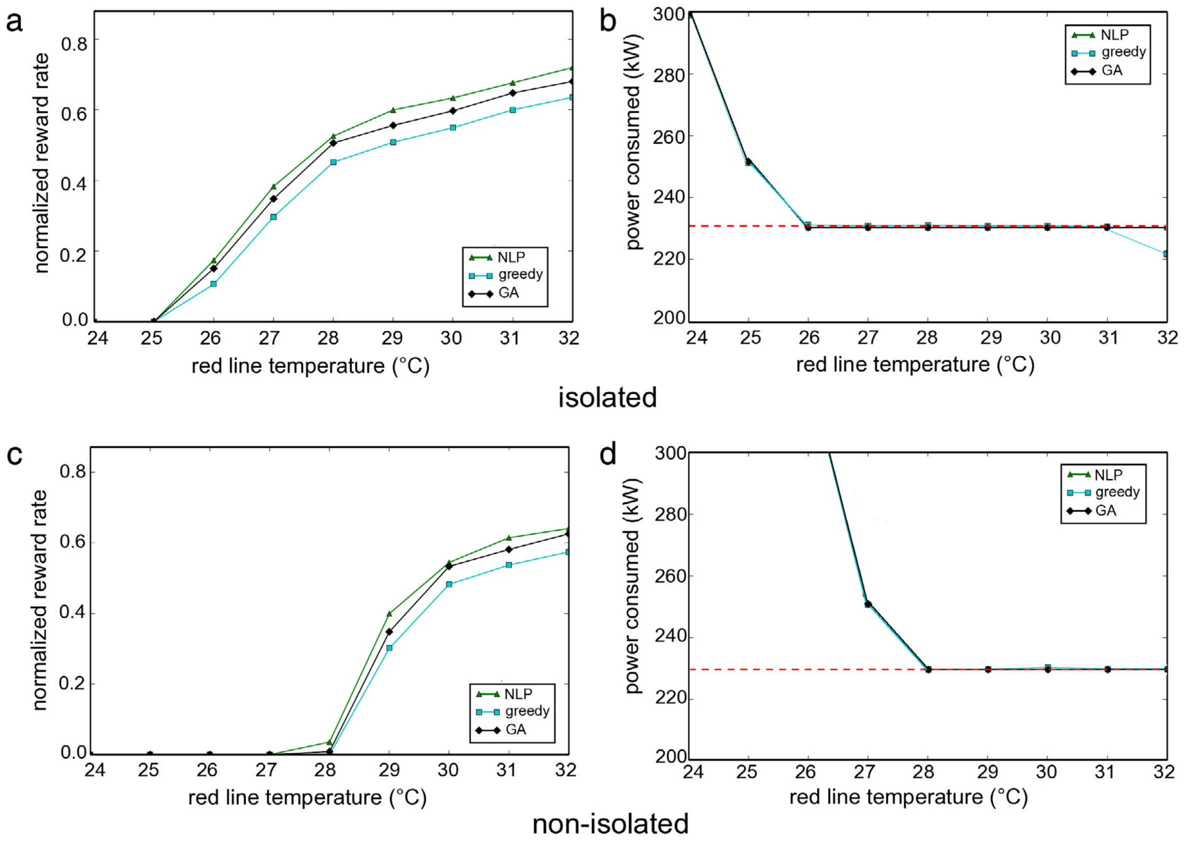


Fig. 5. Sensitivity analysis of our greedy, GA, and NLP resource management techniques for both isolated and non-isolated data center configurations using the 1080 node system on the thermal (red line) constraint. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

configurations. However, the primary drawback associated with the NLP technique is its poor scalability. GAs hold an advantage in that they are able to provide a solution within any given algorithm runtime bounds, though typically better results are obtained the longer they run. Fig. 7(a) and (b) show a comparison of greedy and GA on the small (1080 nodes) and large (4,320 node) platforms, respectively. We recorded the performance of GA at many different heuristic execution times to examine the benefits of running it for different time intervals. The NLP is excluded, as it was unable to finish within two weeks. Simulations were performed on a laptop with an Intel i7-4700HQ CPU and 8GB of RAM within a virtual machine using an Ubuntu Linux operating system. The simulation framework was written in C++.

In summary, for a small platform and problem size, the better results obtained using the NLP approach motivates its use to generate resource allocations that optimize reward rate. However, as the platform size becomes larger, the NLP technique becomes intractable, and GA offers a solution in a reasonable amount of time.

6.6. *Epoch size interval considerations*

The length of the epoch interval has a large impact on the choice of a resource management technique. For a computing environment that has long running and predictable tasks (e.g., supercomputing environments dedicated to climate analysis or molecular biology simulations), the task arrival rates would not change significantly in short periods of time and thus not frequently require new resource allocations. Setting a longer epoch interval time and performing resource management using the longer running techniques (GA or NLP) lends itself to such environments. In an environment where the types or arrival rates of the workload changes rapidly, resource allocation decisions would have to be fast, motivating the use of the greedy technique. Over time, however, our GA and NLP heuristics are able to provide better solutions, and the GA can be terminated at any point, e.g., at the start of a new epoch when new execution rates and P-states need to be found.

7. Conclusions

We studied the problem of maximizing the reward collected for completing tasks by their deadlines subject to power and thermal constraints for heterogeneous data centers. Co-location interference can have a significant impact on the execution speeds of tasks (and thus total reward). We capture these effects with our performance metric that considers co-location. We analyzed several resource management techniques: a greedy technique based on assigning tasks to machines in order of their efficiency (most performance per unit of power), a GA combined with a local search technique that maximizes the reward rate and ensures the power and thermal constraints are met, and an NLP technique building on our prior work to maximize reward rate instead of naïve estimated reward rate that ignores the impact of co-location interference.

The primary contributions of this research were to provide in-depth analyses of the problems and solutions associated with co-location and thermal-aware resource management in heterogeneous computing systems. We used a new thermal model that directly considers CRAC units in its calculation of the thermal influence coefficients, and a new co-location interference model created from a linear regression technique using data from our lab servers. Because the amount of co-location interference can vary

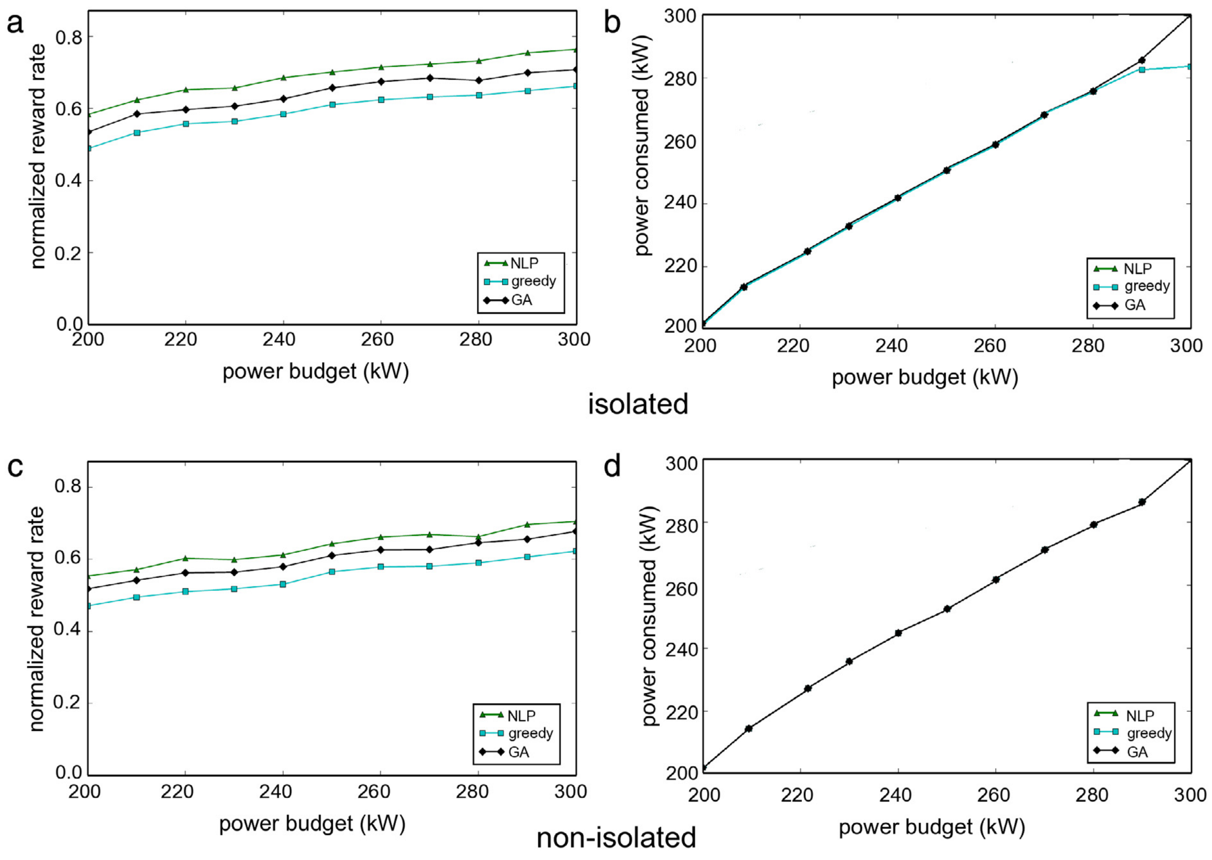


Fig. 6. Sensitivity analysis of our greedy, GA, and NLP resource management techniques for both isolated and non-isolated data center configurations using the 1080 node system on the power constraint.

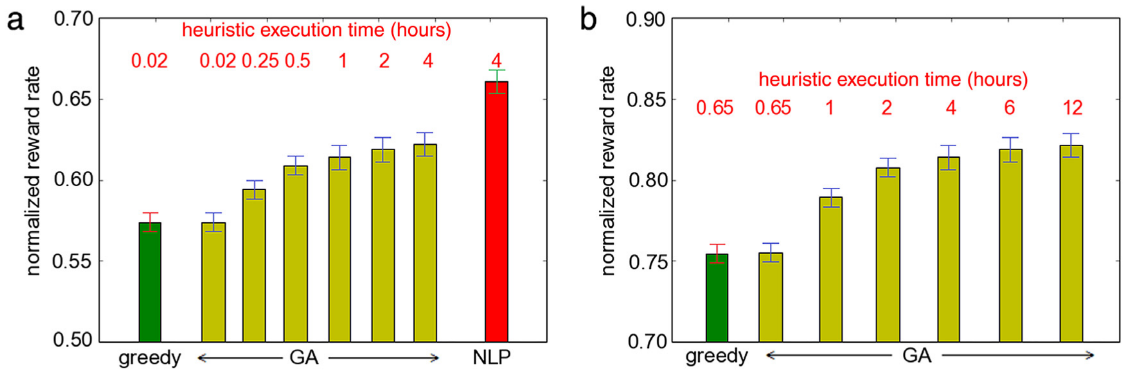


Fig. 7. Comparison of normalized reward rate on the (a) small (1080 nodes) platform, and (b) large (4320 nodes) platform when using the hybrid workload. NLP excluded in (b) because it was unable to finish within two weeks. The power constraint was set to 230 kW for the small platform, and 900 kW for the large platform. The red-line temperature was set to 30°C. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

greatly between workloads, we classified task-types into three different workload environments of varying memory intensities. We found that our greedy technique can perform almost as well as the more complex GA and NLP techniques when interference between tasks is small, but the complex techniques become significantly better at heavy interference.

Isolation of the cold-aisles has a significant impact on the thermal profile of the data center. We quantify the usefulness of isolation by comparing the reward rate our resource management techniques are able to earn in both configurations. We then show a power and thermal constraint sensitivity analysis to answer some "what-if" questions that would help system administrators and facility operators in deciding what power budget or thermal constraints would be ideal.

We have several directions we wish to pursue relating to this research. First is examining the co-location and thermal-aware resource management problem in a real-time dynamic environment instead of rate-based allocations in epoch intervals. The complexity associated with predicting co-location interference and thermal profiles fast enough to make real-time decisions is a daunting task. Second, we wish to account for uncertainties in task execution times that can vary due to changes in the data inputs or network load (e.g., such as techniques in [8,28,37]), and design co-location and thermal-aware resource management techniques that are robust against such uncertainties to more closely match

ARTICLE IN PRESS

real-world problems. In high-performance computing (HPC) environments, it is typical to execute large, parallel workloads. Extending this work to consider tasks that span multiple nodes introduces considerations such as data sharing and communication latencies that would be interesting to analyze and mitigate. Finally, classifying unknown tasks into task types as they arrive would be an intriguing addition to this work, as would multi-objective Pareto front analyses to determine efficient performance per energy operating points.

Acknowledgments

The authors thank Tim Hansen and Daniel Duawe for their valuable comments on this work. This research was supported by NSF grants CNS-0905399, CCF-1302693, and CCF-1252500. This research used the CSU ISTeC Cray System supported by NSF Grant CNS-0923386. We also thank Hewlett Packard for donating servers for this work.

A preliminary version of portions of this material appear in the 2014 International Green Computing Conference [34]. This paper extends [34] in the following ways: (a) enhances our greedy heuristic by considering co-location interference in its decision making and an unmapping feature that unmaps task types on overprovisioned cores, (b) integrates a new model to predict execution times of tasks under co-location interference with the model coefficients obtained by executing benchmarks on server class machines, (c) analyzes our techniques using a different thermal model that directly includes CRAC units, (d) performs in-depth analysis of resource management techniques under several important data center configurations (e.g., isolated and non-isolated cold-aisle configurations), and workload environments of differing memory-intensities, and (e) presents sensitivity analysis of the power and thermal constraints under the different data center configurations and workload environments.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.jpdc.2017.04.015.

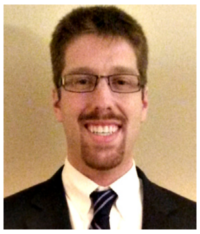
References

[1] A.M. Al-Qawasmeh, S. Pasricha, A.A. Maciejewski, H.J. Siegel, Power and thermal-aware workload allocation in heterogeneous data centers, IEEE Trans. Comput. 64 (2) (2015) 477–491.
[2] ASHRAE Technical Commitee, Thermal Guidelines for Data Processing Environments—Expanded Data Center Classes and Usage Guidance, Tech. rep., American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Inc., 2011.
[3] BASX Solutions, CRAC / CRAH Units. http://www.basxsolutions.com/--crac-crah-units (Accessed 15 October 2015).
[4] H. Bhagwat, U. Singh, A. Deodhar, A. Singh, A. Vasan, A. Sivasubramaniam, Fast and accurate evaluation of cooling in data centers, J. Electron. Packag. 137 (1) (2015) 9 pp.
[5] S. Blagodurov, D. Gmach, M. Arlitt, Y. Chen, C. Hyser, A. Fedorova, Maximizing server utilization while meeting critical SLAs via weight-based collocation management, in: 2013 Int'l Symp. Integrated Network Management (IM '13), 2013, pp. 277–285.
[6] P. Bodik, R. Griffith, C. Sutton, A. Fox, M.I. Jordan, D.A. Patterson, Automatic exploration of datacenter performance regimes, in: 1st Workshop on Automated Control for Datacenters and Clouds, 2009, pp. 1–6.
[7] T.D. Braun, H.J. Siegel, N. Beck, L. Bölöni, R.F. Freund, D. Hensgen, M. Maheswaran, A.I. Reuther, J.P. Robertson, M.D. Theys, B. Yao, A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, J. Parallel Distrib. Comput. 61 (6) (2001) 810–837.
[8] L. Briceño, H.J. Siegel, A.A. Maciejewski, M. Oltikar, J. Brateman, J. White, J. Martin, K. Knapp, Heuristics for robust resource allocation of satellite weather data processing onto a heterogeneous parallel system, IEEE Trans. Parallel Distrib. Syst. 22 (11) (2011) 1780–1787.
[9] M. Chaudhry, T. Ling, A. Manzoor, S. Hussain, J. Kim, Thermal-aware scheduling in green data centers, ACM Comput. Surv. 47 (3) (2015) 48 pp.
[10] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, F. Zhao, Energy-aware server provisioning and load dispatching for connection-intensive internet services, in: 5th USENIX Symp. Networked Systems Design and Implementation, 2008, pp. 337–350.
[11] G.T. Chetsa, L. Lefévre, J. Pierson, P. Stolf, G.D. Costa, Exploiting performance counters to predict and improve energy performance of HPC systems, Future Gener. Comput. Syst. 38 (1) (2014) 287–298.
[12] D. Dauwe, E. Jonardi, R. Friese, S. Pasricha, A.A. Maciejewski, D.A. Bader, H.J. Siegel, A methodology for co-location aware application performance modeling in multicore computing, in: 17th Workshop on Advances on Parallel and Distributed Computational Models (APDCM '15), 2015, pp. 434–443.
[13] C. Delimitrou, C. Kozyrakis, Quality-of-service-aware scheduling in heterogeneous datacenters with paragon, IEEE Micro. 34 (3) (2014) 17–30.
[14] D.G. Feitelson, Parallel workload archive. http://www.cs.huji.ac.il/labs/parallel/workload/l_anl_int/index.html, 2011 (Accessed 17 September 2015).
[15] W. Feng, The Green500 list—November 2016. https://www.top500.org/green500/list/2016/11/, 2016 (Accessed 8 January 2017).
[16] S. Gondipalli, S. Bhopte, B. Sammakia, M. Iyengar, R. Schmidt, Effect of isolating cold aisles on rack inlet temperature, in: 11th Conf. Thermal and Thermomechanical Phenomena in Electronic Systems (ITHERM '08), 2008, pp. 1247–1254.
[17] Google. Efficiency: How we do it. http://www.google.com/about/datacenters/efficiency/internal, 2015 (Accessed 20 November 2015).
[18] S. Govindan, J. Liu, A. Kansal, A. Sivasubramaniam, Cuanta: quantifying effects of shared on-chip resource interference for consolidated virtual machines, in: 2nd ACM Symp. on Cloud Comp. (SOCC '11), 2011 14 pp.
[19] Y. Guo, Y. Gong, Y. Fang, P.P. Khargonekar, X. Geng, Energy and network aware workload management for sustainable data centers with thermal storage, IEEE Trans. Parallel Distrib. Syst. 25 (8) (2014) 2030–2042.
[20] S.K.S. Gupta, A. Banerjee, Z. Abbasi, G. Varsamopoulos, M. Jonas, J. Ferguson, R. Gilbert, T. Mukherjee, GDCSim - A simulator for green data center design and analysis, ACM Trans. Model. Comput. Simul. 24 (1) (2014) 37 pp.
[21] Hewlett-Packard, Intel, Microsoft, Phoenix Technologies, and Toshiba. Advanced configuration and power interface specification, rev. 5.0, 2011.
[22] M.A. Iverson, F. Özgüner, L. Potter, Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment, IEEE Trans. Comput. 48 (12) (1999) 1374–1379.
[23] Z. Jiang, W. Huang, I. You, Z. Qian, S. Lu, Thermal-aware task placement with dynamic thermal model in an established datacenter, in: 8th Int'l Conf. Innovative Mobile and Internet Services in Ubiquitous Comp. (IMIS '14), 2014, pp. 1–8.
[24] M. Kambadur, T. Mosely, R. Hank, M.A. Kim, Measuring interference between live datacenter applications, in: Int'l Conf. High Performance Comp., Networking, Storage, and Analysis (SC '12), 2012 12 pp.
[25] F. Kaplan, J. Meng, A.K. Coskun, Optimizing communication and cooling costs in data centers via intelligent job allocation, in: 4th Int'l Green Comp. Conf. (IGCC '13), 2013 10 pp.
[26] P. Kogge, et al., Exascale computing study: technology challenges in achieving exascale systems, DARPA, 2008.
[27] E.K. Lee, H. Viswanathan, D. Pompili, VMAP: Proactive Thermal-aware Virtual Machine Allocation in HPC Cloud Datacenters, in: 19th Int'l Conf. on High Performance Comp. (HiPC '12), 2012 10 pp.
[28] Y.A. Li, J.K. Antonio, H.J. Siegel, M. Tan, D.W. Watson, Determining the execution time distribution for a data parallel program in a heterogeneous computing environment, J. Parallel Distrib. Comput. 44 (1) (1997) 33–52.
[29] M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, R.F. Freund, Dynamic mapping of a class of independent tasks onto heterogeneous computing systems, J. Parallel Distrib. Comput. 59 (2) (1999) 107–131.
[30] Moab HPC Suite. http://www.adaptivecomputing.com/products/hpc-products/moab-hpc-basic-edition/, 2015 (Accessed 20 November 2015).
[31] J. Moore, J. Chase, P. Ranganathan, R. Sharma, Making scheduling "cool": temperature-aware workload placement in data centers, in: USENIX Annual Technical Conf. (ATEC '05), 2005, pp. 61–75.
[32] A. Nemirovski, Interior point polynomial time methods in convex programming. http://www2.isye.gatech.ed/~nemirovs/Lect_IPM.pdf, 2004 (Accessed 11 March 2016).
[33] S. Nesmachnow, C. Perfumo, I. Goiri, Controlling datacenter power consumption while maintaining temperature and QoS levels, in: 3rd Int'l Conf. Cloud Networking (CloudNet '14), 2014, pp. 242–247.
[34] M.A. Oxley, E. Jonardi, S. Pasricha, A.A. Maciejewski, G.A. Koenig, H.J. Siegel, Thermal, power, and co-location aware resource allocation in heterogeneous high performance computing systems, in: 5th Int'l Green Comp. Conf. (IGCC '14), 2014 10 pp.
[35] M. Polverini, A. Cianfrani, S. Ren, A.V. Vasilakos, Thermal-aware scheduling of batch jobs in geographically distributed data centers, IEEE Trans. Cloud Comput. 2 (1) (2014) 71–84.

ARTICLE IN PRESS

[36] Princeton University. The parsec Benchmark Suite. http://parsec.cs.princeton.edu/overview.htm, 2009 (Accessed 7 December 2014).
[37] V. Shestak, J. Smith, A.A. Maciejewski, H.J. Siegel, Stochastic robustness metric and its use for static resource allocations, J. Parallel Distrib. Comput. 68 (8) (2008) 1157–1173.
[38] SLURM Workload Manager. http://slurm.schedmd.com/, 2015 (Accessed 20 November 2015).
[39] M. Stansberry, 2014 data center industry survey. https://journal.uptimeinstitute.com/2014-data-center-industry-survey/, 2014.
[40] H. Sun, P. Stolf, J. Pierson, G.D. Costa, Energy-efficient and thermal-aware resource management for heterogeneous datacenters, Sustainable Computing: Informatics and Systems 4 (4) (2014a) 292–306.
[41] H. Sun, P. Stolf, J.M. Pierson, G.D. Costa, Multi-objective scheduling for heterogeneous server systems with machine placement, in: 14th Int'l Symp. Cluster, Cloud, and Grid Comp. (CCGRID '14), 2014b, pp. 334–343.
[42] Q. Tang, T. Mukherjee, S.K. Gupta, P. Cayton, Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters, in: 4th Int'l Conf. on Intelligent Sensing and Information Processing (ICISIP '06), 2006, pp. 203–208.
[43] O. Tuncer, K. Vaidyanathan, K. Gross, A.K. Coskun, Coolbudget: data center power budgeting with workload and cooling asymmetry awareness, in: 32nd Int'l Conf. Computer Design (ICCD '14), 2014, pp. 497–500.
[44] V. Villebonnet, D.D. Costa, Thermal-Aware cloud middleware to reduce cooling needs, in: 23rd IEEE Int'l Conf. Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '14), 2014, pp. 115–120.
[45] D. Whitley, The GENITOR algorithm and selective pressure: Why rank-based allocation of reproductive trials is best, in: 3rd Int'l Conf. on Genetic Algorithms, 1989, pp. 116–121.
[46] J. Zhao, X. Feng, H. Cui, Y. Yan, J. Xue, W. Yang, An empirical model for predicting cross-core performance interference on multicore processors, in: 22nd Int'l Conf. Parallel Architectures and Compilation Techniques (PACT '13), 2013, pp. 201–212.

Mark A. Oxley is a Ph.D. graduate from Colorado State University in 2016. He received his B.S. degree in Computer Engineering from the University of Wyoming. His research interests include energy-aware, thermal-aware, and robust resource management techniques. He is currently a Research Scientist at Numerica Corporation in Fort Collins, CO.



Eric Jonardi obtained his M.S. from Colorado State University in 2015. He received his B.S degree in Electrical Engineering from Marquette University in 2013.

Sudeep Pasricha received his B.E. degree in Electronics and Communications from Delhi Institute of Technology in 2000, and his M.S. and Ph.D. degrees in Computer Science from University of California, Irvine, in 2005 and 2008. He is currently an Associate Professor in the Department of Electrical and Computer Engineering and also the Department of Computer Science at Colorado State University. He is a Senior Member of the IEEE and ACM. Homepage: http://www.engr.colostate.edu/_sudeep.

Anthony A. Maciejewski received the B.S.E.E., M.S., and Ph.D. degrees from The Ohio State University in 1982, 1984, and 1987. From 1988 to 2001, he was a professor of Electrical and Computer Engineering at Purdue University, West Lafayette. He is currently a Professor and Department Head of Electrical and Computer Engineering at Colorado State University. He is a Fellow of the IEEE. A complete vita is available at: http://www.engr.colostate.edu/_aam.

Howard Jay (”H.J.”) Siegel was appointed the Abell Endowed Chair Distinguished Professor of Electrical and Computer Engineering at Colorado State University in 2001, where he is also a Professor of Computer Science. From 1976 to 2001, he was a professor at Purdue University. He is an IEEE Fellow and an ACM Fellow. He received B.S. degrees from MIT, and the M.S.E. and Ph.D. degrees from Princeton University. Homepage: http://www.engr.colostate.edu/_hj.

Patrick J. Burns received his B.S. in Mechanical Engineering at Tulane University and his M.S. and Ph.D. at the University of California at Berkley in Mechanical Engineering with an emphasis of heat transfer and energy engineering. He is currently the Vice President for IT and Dean of Libraries, and a professor of Mechanical Engineering. He is responsible for operations of the main and redundant data centers at Colorado State University.

Gregory A. Koenig is an R & D Staff member at Oak Ridge National Laboratory where his work involves developing scalable system software and tools for ultrascale-class parallel computers. He holds three B.S. degrees (computer science, 1993; electrical engineering technology, 1995; mathematics, 1996) from Indiana University-Purdue University Fort Wayne, as well as an M.S. (2003) and Ph.D. (2007) in computer science from the University of Illinois at Urbana–Champaign.