J. Parallel Distrib. Comput. 96 (2016) 181–193

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Communication and cooling aware job allocation in data centers for communication-intensive workloads☆

Jie Meng[a], Eduard Llamosí[b], Fulya Kaplan[a,*], Chulian Zhang[b], Jiayi Sheng[a], Martin Herbordt[a], Gunar Schirner[b], Ayse K. Coskun[a]

[a] Electrical and Computer Engineering Department, Boston University, Boston, MA, United States
[b] Electrical and Computer Engineering Department, Northeastern University, Boston, MA, United States

HIGHLIGHTS

- We jointly optimize the cooling and communication costs via job allocation.
- Our joint allocation strategy saves 16.4% cooling energy on average.
- We design a framework to extract the communication patterns of HPC applications.
- Combining joint allocation with task mapping reduces communication costs by 20.9%.

ARTICLE INFO

Article history:
Received 30 March 2015
Received in revised form
2 May 2016
Accepted 23 May 2016
Available online 8 June 2016

Keywords:
High-performance computing
Communication Pattern
Job allocation
Task mapping
Optimization
Cooling management

ABSTRACT

Energy consumption is an increasingly important concern in data centers. Today, nearly half of the energy in data centers is consumed by the cooling infrastructure. Existing policies on thermally-aware workload allocation do not consider applications that include many tasks (or threads) running on a large set of nodes with significant communication among the tasks. Such jobs, however, constitute most of the cycles in high performance computing (HPC) domain, and have started to appear in other data centers as well. Job allocation strongly affects the performance of such communication-intensive applications. Communication-aware job allocation methods exist, but they focus solely on performance and do not consider cooling energy. This paper proposes a novel job allocation methodology to jointly minimize communication cost and cooling energy consumption in data centers. We formulate and solve the joint optimization problem using binary quadratic programming. Our joint optimization algorithm reduces cooling energy by 16.4% on average with only a 2.66% average increase in application running time compared to solely performance-aware allocations. To further optimize the communication cost, we develop a Charm++ based framework that extracts the communication behavior of applications. We then integrate our job allocation policy with recursive coordinate bisection (RCB) based task mapping method to place highly-communicating tasks in close proximity. Experimental results show that task mapping further decreases the communication cost by up to 20.9% compared to assuming all-to-all communication, a popular assumption in much of the prior work.

1. Introduction

Over the last decade, energy consumption of data centers has become a first class concern on par with their computational capacity and may be the most formidable challenge in achieving continued performance scaling. World-wide data center energy requirements doubled from 2007 to 2011, and further increased by 63% in just the next year [49]. Cooling costs have increased in parallel and now constitute 25%–50% of the overall energy cost of a data center [40]. A lot of effort has been given to decrease data center Power Usage Effectiveness (PUE, the ratio of the total data center facility power to the power delivered to computing equipment) recently. According to a recent report, the average PUE reduced from 2.5 in 2007 to 1.89 in 2011, reaching 1.65 in 2013 [42]. However, even a PUE of 1.65 means that 40% of the

☆ This work has been partially funded by the Massachusetts Green High-Performance Computing Center (MGHPCC), Sandia National Laboratories, and by NSF award #CNS-1205593. The authors thank Dr. Vitus Leung and Dr. Gilbert Hendry at Sandia National Laboratories for their valuable feedback.

* Corresponding author.
E-mail address: fkaplan3@bu.edu (F. Kaplan).

http://dx.doi.org/10.1016/j.jpdc.2016.05.016
0743-7315/© 2016 Elsevier Inc. All rights reserved.

total power is consumed by non-IT equipment (including cooling), which indicates significant room for improvement. Recently, free cooling strategies have been proposed to achieve PUE values in the range of 1.1–1.3, however, free cooling may not be a feasible option for many data centers due to the geographical location, seasonal changes, and the required infrastructure costs.

The temperature of the computer room air conditioner (CRAC) is a fundamental parameter in determining the amount of cooling power consumed in current data centers. Therefore, one key factor in efficient cooling is to keep the CRAC supply temperature as high as possible [34]. A number of techniques focus on decreasing the inlet temperatures of data center nodes through thermally-aware job allocation so that a higher CRAC supply temperature is sufficient to cool the data center [34,44].

Performance is another first-order constraint in data center design and management. Communication-intensive parallelized applications constitute most of HPC workloads, and have started to occupy more cycles in other data centers following the efforts in designing parallelized large-scale applications in many domains. These applications run on multiple nodes with intensive communication between the threads, with job running times in the range of minutes, hours, or even days. In particular, the performance of a communication-intensive application is sensitive to the specific allocation and mapping onto the compute nodes. In other words, communication cost has a significant impact on system performance [28,33,8], with long communication distances and communication resource contention increasing the overall delay. Leung et al. have shown a 2X slow down for two communication-intensive jobs when hand-placing them such that their communication paths overlap significantly [28].

Most of the existing performance-aware job allocation algorithms focus on minimizing the average number of communication hops between communicating nodes (e.g., [33,8,10,43]). For data centers running transactional enterprise loads, performance-aware job allocation algorithms focus on satisfying the response time constraints imposed by service level agreements (e.g., [38]). However, the workload and performance models of such algorithms do not apply to parallel HPC applications as they do not consider the impact of communication among the tasks.

Existing algorithms for job allocation in HPC data centers address cooling efficiency and performance separately [33,8,34,44]. Treating these problems separately, however, may incur significant inefficiencies. For example, allocating jobs solely to minimize cooling cost may map communicating tasks of an HPC application to distant nodes. Increased communication distance increases communication delay which, in turn, increases the running time. This may in fact cause a larger cooling energy consumption. Thus, a solution that considers both performance and cooling energy for HPC applications running in data centers is truly essential for improving energy efficiency. In this paper, we formulate and solve a joint optimization problem that simultaneously considers the cooling power and communication latency for parallel HPC applications with highly communicating threads. Our specific contributions are as follows:

- We propose a two-level job allocation policy. The first level policy allocates the jobs to the nodes of a data center to jointly optimize the communication cost of communication-intensive applications and the cooling energy. We formulate and solve the first level joint optimization problem using binary quadratic programming.
- The second level policy maps the tasks of a job onto specific nodes. It takes communication patterns of applications into account to further reduce communication cost by minimizing the distance between tasks with frequent communication. Our task mapping technique uses a modified version of the recursive coordinate bisection (RCB) algorithm [21].
- We expand the Charm++ run-time environment [22] to automatically extract communication patterns from HPC applications during simulation. Our framework-instrumented approach does not require modification of the application code.
- Our joint optimization algorithm reduces the cooling energy by 16.4% on average compared to the performance-aware job allocation policy, while achieving similar performance. Solely optimizing for cooling energy results in 19.6% higher communication cost in comparison to the performance-aware allocation. Our task mapping policy decreases communication cost by up to 20.9% compared to a generic all-to-all pattern.

The rest of the paper starts with a discussion of related work. Section 3 presents the experimental methodology for performance simulation, communication pattern extraction, as well as data center temperature and cooling power modeling. We introduce the proposed job allocation and task mapping strategies in Section 4. We provide experimental evaluation in Section 5 and conclude in Section 6.

2. Related work

This section first reviews data center job allocation and task mapping algorithms for optimizing the performance of HPC applications. It then discusses existing approaches for reducing data center cooling energy costs. The section also highlights distinguishing features of our work.

2.1. Job allocation for optimizing performance of communication-intensive jobs

Most of the performance-aware job allocation algorithms designed for HPC data centers and supercomputers focus on minimizing the average number of communication hops among the nodes that run the application. Some of the algorithms allocate only a *contiguous* (i.e., adjacent) sets of processors to each job (e.g., [10,43,13]), as contiguous node allocation provides significant reduction in execution time for communication-intensive parallel programs. For example, Bhattacharya et al. use a look-ahead method that analyzes the job queue and design an algorithm to detect free sub-mesh area for efficient allocation [10]. Contiguous allocation, however, may result in external fragmentation (i.e., available nodes that are separated from each other cannot be utilized) and reduce the achievable system utilization.

A number of recent communication-aware job allocation techniques allow *discontiguous* allocation of processors. Mache et al. present the *MC* allocation strategy for mesh-connected parallel computers. Their method yields compact allocations by containing the jobs in the smallest rectangular area possible [33]. Motivated by the MC allocation strategy, Bender et al. propose an MC1x1 processor-allocation algorithm, in which the first sub-mesh is a 1X1 *shell* and subsequent sub-meshes grow in the same way as in MC [8]. Walker et al. discuss fast job allocation algorithms for mesh-connected clusters [50]. Their method is based on space-filling curves, where processors are ordered according to a given curve. None of these approaches consider the impact of job allocation decisions on the data center cooling power.

2.2. Task mapping for communication-intensive applications

Task mapping techniques consider intra-application communication patterns while assigning specific tasks or threads of an HPC application to nodes. Hoefler et al. present several mapping algorithms including a greedy algorithm, a recursive bisection mapping algorithm, and a new mapping strategy based on graph similarity [21]. They demonstrate that the proposed algorithms achieve significant reduction of congestion for various network topologies and communication patterns. Bhatelé et al. introduce a framework based on heuristic models to map an application's communication graph to either 2D or 3D mesh-based topologies [9]. Wu et al. propose a hierarchical task mapping for cosmological simulations, taking both the inter-node and intra-node mappings into account [52]. Chung et al. also propose a hierarchical mapping strategy that achieves $O(n \log n)$ complexity by grouping heavy-communicating processes together [14].

Existing performance-aware job allocation and task mapping strategies focus on improving the performance and reducing the communication overhead, without considering the potential impact of job allocation on the data center energy consumption or temperature. Next, we review the related work on cooling energy management.

2.3. Cooling energy management

A number of thermal modeling and management techniques at the data center level have been proposed to decrease data center cooling costs. Simpler thermal models either use a linear formula to compute steady state server temperatures [26] or a first order RC-network to model transient server temperature [51]. These models take thermal resistance, thermal capacitance, server power consumption and the ambient temperature as input. The disadvantage of these models is that they do not take the *recirculation* phenomena into account. Some work evaluate temperature using computational fluid dynamics (CFD) simulations, which provide high accuracy at the cost of long simulation times [34]. Recent research provides more accurate thermal models for data centers based on the layouts and thermal sensor data from real-world data centers. Heath et al. introduce a data center temperature emulation suite named Mercury that estimates temperatures based on the data center layout, hardware, and component utilizations [19]. Tang et al. propose a linear model to compute data center temperatures and cooling power fast and accurately [44]. Their model considers *recirculation* effect and is validated by comparing against CFD simulations.

Workload allocation and scheduling have been used for managing data center temperatures [44,34,6]. Moore et al. present two temperature-aware workload placement algorithms: the first one assigns power inversely proportional to the server's inlet temperature and the second one minimizes the heat recirculating within the data center [34]. Tang et al. solve an optimization problem for minimizing the peak node inlet temperatures through job allocation [44]. Other work proposes spatio-temporal job scheduling with dynamic thermostat setting [6].

Another group of work propose techniques that target minimizing data center power consumption through dynamic voltage–frequency scaling (DVFS) and consolidation. Sarood et al. present algorithms to prevent thermal hot spots using DVFS and frequency-aware load balancing [39]. Pakbaznia et al. propose the Minimum Total Data Center Power algorithm to minimize the total server and cooling power by turning off part of the servers and applying DVFS [35]. Other works such as PowerTrade and TACOMA reduce the total power by trading-off idle power and cooling power [4,1]. These algorithms dynamically decide on how many and which servers to keep in active/idle states depending on the load and cooling demands [4,1]. Sansottera et al. propose the Greedy Least Power (GLP) algorithm to minimize the power consumption while satisfying response time constraints imposed by service level agreements [38]. Other research focuses on optimizing performance under temperature and power constraints [5,12]. Al-Qawasmeh et al. assign performance states to computing cores to maximize the number of tasks completed by their deadline [5], while Chen et al. implement dynamic resource allocation and workload migration in virtualized data centers [12].

Recently, data center management techniques using renewable energy or free cooling has been gaining attention [53,17,30,29,27]. The main focus of these techniques is to select the most efficient source of energy to minimize the energy consumption. They use various approaches such as predicting the IT demands and allocating the IT resources based on varying power supply [30], dynamically choosing the source of energy among the renewable energy, batteries or the grid [17], coordinating liquid cooling and free cooling for power optimization [29], and distributing the workload in geo-distributed data centers with cost and energy-efficiency considerations [53,27].

The existing techniques on cooling energy management generally target transactional data center loads such as web applications and short requests [1,4,38] rather than communication-intensive parallel HPC applications. They use performance metrics such as average response time and QoS [1], user estimated job turnaround times [6], estimated computing time based on the P-state [5], and service level agreements (SLAs) [30,27,12]. However, *none* of these works on cooling-aware management consider the performance impact of the location of communicating tasks as part of their placement or scheduling algorithms.

2.4. Distinguishing aspects from prior work

Our work differentiates from prior research as our job allocation policy simultaneously optimizes the application performance (in terms of the communication cost) and the cooling energy of HPC data centers. We demonstrate that for jobs involving intensive communication between the nodes, application performance becomes an important factor in determining the total cooling energy consumed.

In comparison to the heuristic solver we proposed in our recent work [24], we formulate and solve the joint optimization problem using binary quadratic programming method, which provides a faster and scalable solution to the joint optimization problem. Additional contributions include the integration of task mapping with our job allocation policy to further reduce the communication cost of communication-intensive applications, and considering a more accurate communication model. To automatically obtain communication patterns from real HPC applications, we extend Charm++'s performance analysis tool, *Projections*. Our work is the first to perform task mapping based on real communication patterns along with job allocation optimization for minimizing both the communication cost and cooling energy.

3. Experimental methodology

The target system in this work is a small data center with two rows of industry standard racks arranged in the layout shown in Fig. 1. In this arrangement, rack inlets where the cool air is supplied are facing the outer aisles forming cold aisles at the sides. Rack outlets, where the hot air exits, are facing each other forming a hot aisle in between the two rows. Each row is composed of five racks and each rack has four compute *nodes*. In our experiments, we assume that each *node* includes 10 servers and each server has two processors. This layout corresponds to a total of 800 processors across the two rows of the data center. The proposed data center setup has been commonly used in recent work and is representative of the organization in today's data centers [38].

Our simulation infrastructure is composed of two main parts: performance and cooling. Performance simulation setup is used to determine the communication delays between communicating nodes in the data center. Cooling energy simulation involves server power model and data center thermal model. Using the server power model and the maximum node inlet temperature estimates, we compute the CRAC unit cooling power.

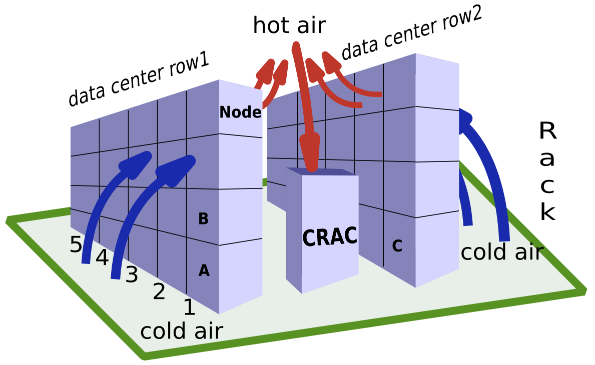


Fig. 1. Layout of the target data center.

3.1. Performance simulation infrastructure

In this paper, we target communication-intensive parallel applications that use high-level communication middleware such as Charm++ and message passing interface (MPI). For such workloads, the communication overhead in the data center is one of the major performance bottlenecks [33]. We evaluate our job allocation algorithms for data centers with 3D-mesh networks. Mesh-connected networks for message passing are commonly used in many data centers, especially in HPC data centers and supercomputers. Some examples include Cplant, a commodity-based supercomputer developed at Sandia National Laboratories [28], IBM Blue gene/l supercomputer [3], and Cray XT4 distributed memory multiprocessor by Google [2]. As for supercomputers, torus network (a form of 3D-mesh) is the first choice since it has no network edges and provides high bandwidth nearest-neighbor connectivity [3]. Torus network is also applicable to many scientific and data-intensive applications; therefore, it requires no extra network-specific changes to programs.

The simulation infrastructure used in this paper is a cluster-level simulator, SST/Macro [20], which is designed by Sandia National Laboratories. SST/Macro performs coarse-grained simulation of HPC applications. It is configurable to simulate data centers with various sizes and network topologies. The simulation results of SST/Macro have been validated using the performance data from real-world data centers.

In our implementation on SST/macro, we use two skeleton applications to extract the communication time between each pair of nodes within the data center: a *test* application to send and receive messages between each pair of nodes in the data center and a *noise* application to generate random congestion on the network. We conduct simulations for each pair of nodes within the data center in presence of randomized network congestion, and then collect the average communication latencies. For example, for the 3D mesh network, we observe that the communication latency for a pair of nodes is linearly correlated with the total X–Y–Z (3D-Manhattan) distance between the nodes as expected. The simulation results are used for constructing the communication cost matrix, H, of the data center. The H matrix, which contains the relative communication latencies for each pair of nodes, is used in the optimization formulation in Section 4.2.

3.2. Workloads and extraction of communication patterns

We use two categories of workloads in this paper: (1) generic communication-intensive jobs with various sizes, where each job requests a number of nodes to run on; (2) jobs that are based on real-life HPC workloads. We use (1) for evaluating our policies for a large number of dynamic scenarios and (2) to investigate the benefits of communication pattern-awareness in our proposed technique.

To obtain realistic workloads in category (2), we study eight Charm++ applications that are actively used at multiple HPC centers (or closely resemble such workloads). We selected the following applications (unless otherwise noted, they are obtained from Charm++ web site [48]): An implementation of the Fast Fourier Transform (*FFT*). *CharmLU*, a linear solver operating on sparse matrices, is representative of applications operating on sparse data sets. *NAMD* is a widely used parallel molecular dynamics simulator for large molecular systems [37]. We investigate two variants. *NAMD-noPME* omits the long-range force computation and uses only range-limited electrostatics resulting in primarily local communication. Conversely, *NAMD-PME* includes the long-range force computation, which results in additional communication when transforming into vector space as well as the addition of a FFT stage. *Barnes-Hut* [7] and *ChaNGa* (Charm N-body Gravity solver) [16] are celestial body simulations with a tree-based implementation. *OpenAtom* [11], a quantum chemistry simulation framework, has a number of interdependent heterogeneous phases including sparse and dense FFTs, as well as non-square matrix multiplications. Finally, the adaptive mesh refinement *AMR* adaptively refines computation through repartition and data migration.

We use an extended version of Charm++'s performance analysis tool, *Projections* [23], to measure an application's communication volume (bytes) and number of messages between each pair of processing elements (PEs). The correctness of this extension has been validated through the use synthetic benchmarks with controllable payloads as well as manually instrumenting NAMD at application level [25].

Fig. 2 shows the communication volumes between PE pairs for the eight applications (source on x-axis, destination on y-axis) which is visualized by color, with red indicating a higher volume.

Overall, communication among neighboring PEs is most common among all applications (visible as a diagonal line with some spread). It is most noticeable in *OpenAtom*. *CharmLU* has mostly local, but irregular communication. In addition, PE-0 acts as a coordinator sending more messages to all other PEs. *Barnes-Hut* has a distinct pattern with source PE index being two times of the destination PE index. In addition, it has an all-to-all communication for PEs-0 through 17. *ChaNGa* uses Barnes-Hut for much of its computation, but has drastically different communication. There is much local communication, but also more communication between complementary PEs. *NAMD-noPME* mostly exhibits local neighbor communication. Including long-range electrostatics, *NAMD-PME* significantly increases communication with mostly an all-to-all background, but also increases the distance and volume of neighbor communication. *AMR* is dominated by local communication, but with a wider range compared to the other applications. Similarly to *CharmLU*, PE-0 acts as a coordinator.

Most applications exhibit communication patterns that significantly differ from the commonly assumed *all-to-all* pattern, motivating pattern-aware task mapping optimization (i.e., placing highly communicating tasks in close proximity to each other in the cluster). One exception is FFT (Fig. 2(g)), which shows a uniform all-to-all communication from the matrix transposition operations, yielding limited optimization potential.

3.3. Server power model

A server's power consumption is the dominant factor in determining its temperature and the required cooling energy to maintain operation within safe temperatures. The power consumption of the servers varies with their activity. We assume that a server consumes 300 W during computationally-intensive phases, 230 W during communication-intensive phases, and 100 W when idle, based on reported power consumption values of several data center servers [38,31]. Power consumption is

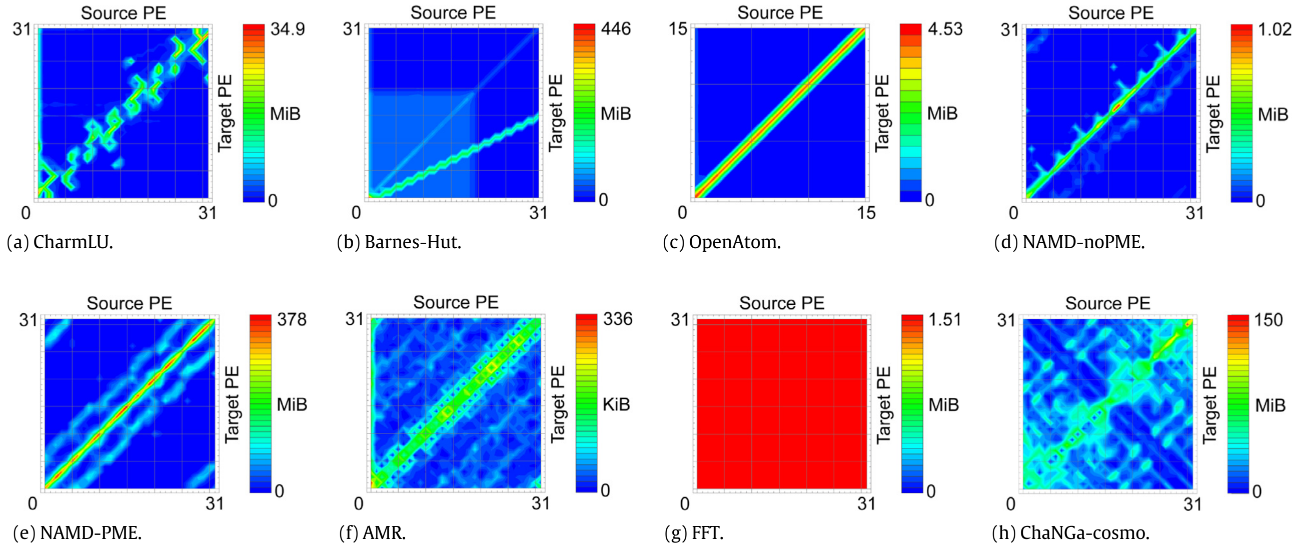


Fig. 2. Communication patterns in total Mebibytes (MiB).

lower during the communication phases due to the lower CPU activity and the time spent in I/O. We target large communication-intensive jobs and assume that each job fully occupies a set of nodes (i.e., all of the servers in a node would be assigned to the same job). Therefore, for each node (each containing 10 servers), the total power of the node is 3000 W, 2300 W, or 1000 W, depending on the activity and regardless of the application.

In this work, we specifically focus on data centers running communication-intensive parallel workloads, where the applications highly utilize the nodes. Therefore, we assume the same node power consumption for each application. We observe that these jobs spend a large percentage of time in communication, and the ratio of the communication time to total running time typically varies between 20% and 40% [41,15]. We select the ratio of communication time to running time as 30% in this work. Thus, 30% of the workload running time is spent in the communication phase, representing an average case. Based on the allocation decision, the communication latency of the application changes. We reflect this change by adjusting the communication portion of the running time. We compute the node's total power consumption while executing a job as the weighted sum of the computation and communication power.

Our experimental choices are based on real-life scenarios in today's large scale computing clusters. Without loss of generality, the proposed techniques are applicable to data centers with different server power levels and applications with different communication to computation ratios. We discuss how the results vary as the power and communication activities change in Section 5.

3.4. Cooling energy model

We use a typical data center layout [38] as shown in Fig. 1. In this layout, cold air enters the room from the floor tiles into rack inlets and gets hotter within the racks. Hot air exits from the back to the center aisle and leaves the room from the ceiling. This hot aisle/cold aisle arrangement avoids mixing cold supply and exhaust air.

In order to compute the cooling energy consumption of the data center during job allocation, we use the model proposed and validated by Tang et al. [45]. Their model combines a linear, low-complexity heat recirculation model with a linear power model. This model is more practical than most existing models as it requires only one set of CFD simulations to characterize the data center. Once we have the measured data center parameters, the vector of inlet temperatures, T_{in}, for all the nodes is computed by the following linear equation:

$$T_{in} = T_{sup} + DP \qquad (1)$$

$$D = [(K - A^T K)^{-1} - K^{-1}] \qquad (2)$$

where T_{sup} is the CRAC unit supply temperature vector, D represents the heat distribution matrix and P is the node power vector. K is the thermodynamic constant matrix, which is calculated as follows:

$$K = diag(K_i), K_i = \rho f_i c_p \qquad (3)$$

where $\rho = 1.19$ kg/m^3 is the density of air, $f_i = 0.2454$ m^3/s is the flow rate of node i (assumed fixed for all nodes), and $c_p = 1005$ J/kg K is the specific heat of air [44].

Matrix A is the heat cross-interference coefficient matrix representing the recirculation phenomena of a data center. It represents the fraction of output heat from each node that is recirculated to the inlet of the other nodes. It is an $N \times N$ matrix for a system with N nodes with each term a_{ij} representing the fraction of heat at node i recirculating back into node j. Matrix A just needs to be obtained once for a particular data center through CFD simulations or sensor measurements [45].

The heat cross-interference coefficient matrix has been used by many others to carry out large scale thermal simulations rapidly and accurately [35,39,38]. We utilize the heat cross-interference coefficients for the data center given in recent work [38]. Fig. 3 shows the cross-interference coefficient matrix for the 40-node system in a 3D map.

Data centers with different layouts and heat flow characteristics have different A matrices. However, the equations to calculate the inlet temperatures are independent of the data center. This linear thermal model using the cross-interference matrix [44] is so far one of the most practical models available for data center thermal simulations.

After deriving the heat cross-interference of the target data center, we are able to model the impact of job allocation on the cooling energy cost. The power values of the nodes are used in Eq. (1) to calculate the inlet temperatures resulting from different allocation schemes.

We perform node-level allocation in this paper, which is a reasonable hierarchical level for data centers that run long jobs with large number of threads. Assume that a job has size n, which corresponds to the total number of nodes the job requires. x_i is an

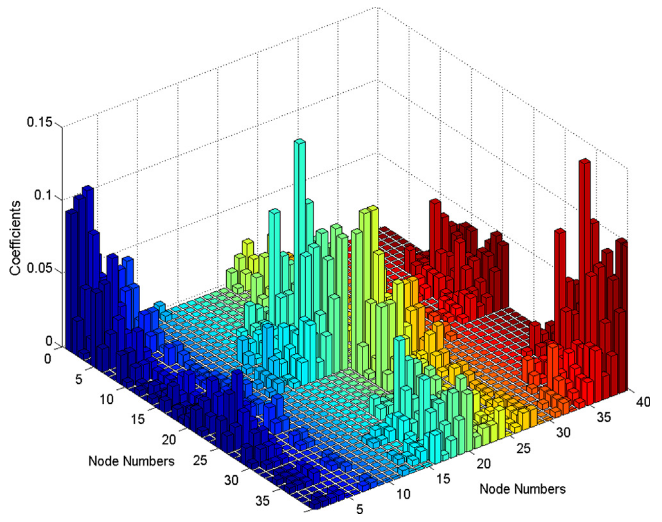


Fig. 3. Cross-interference coefficient matrix *A*.

integer {1,0} variable showing whether node i is assigned a job or not. The power consumption of a node i can be expressed with a linear model as follows:

$$P_i = P_{idle,i} + x_i P_{active,i} \tag{4}$$

where $P_{idle,i}$ is the node idle power and $P_{active,i}$ is the power consumed when running a task. $P_{active,i}$ is dependent on the communication to computation ratio of the job, as discussed earlier. We focus on the steady-state power consumption and use a constant $P_{active,i}$ during job execution, as the server power fluctuations are limited compared to the total node power. Also, the average power consumption determines the cooling need because of the large thermal time constants involved in data center cooling.

After computing the inlet temperatures, we use a cooling power model to estimate the power consumed by the cooling unit, which depends on the efficiency of the CRAC unit. A common CRAC unit efficiency metric is the coefficient of performance (CoP). CoP is the ratio of the heat removed from the system to the energy spent on cooling: CoP $= P_c/P_{AC}$, where P_c is the total computing power (sum of the values in the P vector) and P_{AC} is the cooling power. CoP increases with higher CRAC supply temperature (T_{sup}). In this work, we use the CRAC unit CoP model given by [34] as follows:

$$CoP(T_{sup}) = 0.0068{T_{sup}}^2 + 0.0008T_{sup} + 0.458 \tag{5}$$

where T_{sup} is in Celsius. The upper limit on how much the supply temperature (T_{sup}) can be increased depends on the difference between redline temperature (T_{red}), which is the highest allowed temperature at the node inlets, and maximum node inlet temperature ($T_{in,max}$). In other words, we can use this temperature *slack* to increase the supply temperature and operate at higher CRAC efficiency without violating the temperature constraints. A new supply temperature can be computed by adding this difference to T_{sup}, and the cooling cost is calculated as follows:

$$T_{sup'} = T_{sup} + T_{red} - T_{in,max} \tag{6}$$

$$P_{AC} = \frac{P_c}{CoP(T_{sup'})}. \tag{7}$$

This model provides fast results and it is able to capture the effect of recirculation. The accuracy of the temperature model is verified in prior work [45].

4. Proposed joint optimization methodology

In this section, we present our optimization algorithm for minimizing the communication cost and the cooling energy cost of data centers through job allocation and task mapping. Our goal in allocation optimization is to simultaneously reduce the communication cost for each job and the cooling energy spent in the data center. We solve this problem using binary quadratic programming. The final part of this section discusses the integration of job allocation with task mapping. After we assign a set of nodes to each job, we allocate each task of the job to a specific node so as to reduce the communication cost, taking the job's communication pattern into account.

4.1. Formulation of the joint optimization problem

We formulate the job allocation problem to reduce both communication cost and cooling energy cost at the same time. Moreover, we provide an optimization algorithm with adjustable weighting factors between communication and cooling energy cost. In this way, we enable optimizing the overall cost of data centers for various preferences of performance or cooling energy savings.

The objective of our joint optimization problem is to allocate a job on a set of idle nodes on the data center such that both the communication delay of the application and the required cooling power are jointly minimized. The formulation of the job allocation problem with this joint goal is shown in Eq. (8):

$$\begin{aligned} \underset{X_{job}}{\text{minimize}} \quad & \alpha \cdot Cost_{comm}(X_{job}) + \beta \cdot Cost_{cool}(X_{job}) \\ \text{subject to} \quad & E \cdot X_{job} = n \end{aligned} \tag{8}$$

where $X_{job} = \{x_1, x_2, \ldots, x_N\}$ is an $N \times 1$ vector that represents the job allocation decision. N is the number of total nodes within the data center and x_i are the integer variables denoting whether a node is busy or idle. $N = 40$ in our target data center, as described in Section 3.

When a new job arrives, we solve the optimization problem to determine the idle nodes the job will be allocated to. Thus, after allocating each new job, a set of nodes switch states from idle to busy, and we modify the corresponding x_i values from 0 to 1. Similarly, when a job finishes execution, the x_i variables are updated back to 0. Our algorithm is also able to allocate jobs onto a partially utilized data center. If a new job arrives while some other jobs are running, we resolve the optimization problem by considering the busy nodes in our constraints. In this work, we do not consider workload migration, following the current common trends in HPC data centers.

In Eq. (8), E is a $1 \times N$ vector with all of its elements set to 1. n is the total number of nodes required by a job. In other words, we can consider each job as having n high-level tasks, where each task is allocated on a node. The linear constraint $E \cdot X_{job} = n$ means that the job requires n nodes in the data center.

$Cost_{comm}(X_{job})$ and $Cost_{cool}(X_{job})$ represent the communication cost of the job and the cooling cost of the data center, respectively. α and β are the corresponding weight factors for the communication cost and the cooling cost. α and β can be adjusted to adapt to optimization requirements in different data centers. A larger ratio of α/β indicates that reducing the communication cost is more significant compared to decreasing the cooling cost. For example, when $\alpha = 1$ and $\beta = 0$, the joint optimization problem is converted to the job allocation problem that only considers the communication cost.

In our work, we assign equal weights to the cooling and communication costs. However, we apply scaling factors to both parts of the goal function such that when we set $\alpha = \beta = 0.5$, the two cost terms will contribute similarly to the goal. That is, the cost function in Eq. (8) is updated as follows: $\alpha' \cdot Cost_{comm}(X_{job}) + \beta' \cdot Cost_{cool}(X_{job})$, where $\alpha' = \alpha \cdot (scaling\ factor_{comm})$ and $\beta' = \beta \cdot (scaling\ factor_{cool})$. The scaling factors are determined empirically.

4.2. Solving the joint optimization problem

In order to solve the joint optimization problem, we need to formulate both the communication cost and the cooling cost. The communication cost of each job arriving at the cluster can be expressed as in Eq. (9):

$$Cost_{comm}(X_{job}) = \frac{X_{job}^T \cdot H \cdot X_{job}}{n} \tag{9}$$

where H is an $N \times N$ matrix, whose elements represent the communication delay between each pair of nodes within the data center. Thus, Eq. (9) calculates the total communication cost among all the nodes that are assigned to the current job. The total cost is then normalized to the job size, n. This normalization step is added because it is intuitive to have a higher overall communication delay for a larger job; however, we would like to give similar importance to all the jobs in the optimization irrespective of their sizes.

The communication cost matrix H is determined by the data center's network topology. We construct the H matrix based on average delays due to the number of network hops, which are constant. We consider average traffic congestion, as our goal is to optimize the steady-state costs. We leave the analysis on the impact of time-variant congestion on the performance variations to future work. We use SST/Macro to extract the H matrix, as discussed in Section 3.1. Other methods of constructing this matrix for mesh topologies include computing the pairwise L1 (Manhattan) distances across the communication nodes [28]. Our simulation approach is capable of generating the H matrix for various data center network topologies.

The cooling energy cost of the data center after the allocation of a new job can be formulated as in Eq. (10), which is derived from the cooling energy model presented in Section 3.4:

$$Cost_{cool}(X_{job}) = \frac{sum\{D \cdot (P_{active} \circ X_{job}) + T_{sup} + D \cdot P_{idle}\}}{U} \tag{10}$$

where T_{sup} is the CRAC unit supply temperature vector and D is the $N \times N$ heat distribution matrix. P_{idle} and P_{active} are $N \times 1$ vectors representing the idle and dynamic power for the nodes, respectively. U is the number of currently active nodes.

As described in Section 3.4, the cooling energy of the data center is a function of the inlet temperatures of the nodes within the data center. $D \cdot (P_{active} \circ X_{job})$ is the portion of the inlet temperature vector that is determined by the job allocation decisions. Here, '$\circ$' represents element-wise multiplication of matrices (i.e., Hadamard product). $T_{sup} + D \cdot P_{idle}$ is the constant part of the inlet temperature formulation. In Eq. (10), we normalize the summation term by the total number of currently active nodes in the data center, U. This normalization prevents potential biases of the system utilization on the results during each iteration of the optimization. In other words, our approach minimizes the cooling cost after each job's arrival regardless of the current load on the data center.

The cooling energy of the data center is determined by the *maximum* node inlet temperature, as described in Section 3.4. When we are performing evaluations in Section 5, we calculate the cooling power required by the data center using the maximum inlet temperature across all of the data center's nodes. However, in our formulation of the optimization problem in the binary quadratic model (Eq. (10)), we compute cost as a function of the *average* inlet temperature. In other words, instead of minimizing the maximum inlet temperature across all the data center's nodes, our optimization algorithm minimizes their average inlet temperature normalized to the data center utilization. Using the average inlet temperature provides substantial advantages in simplifying the problem's solution; however, it can introduce suboptimal solutions.

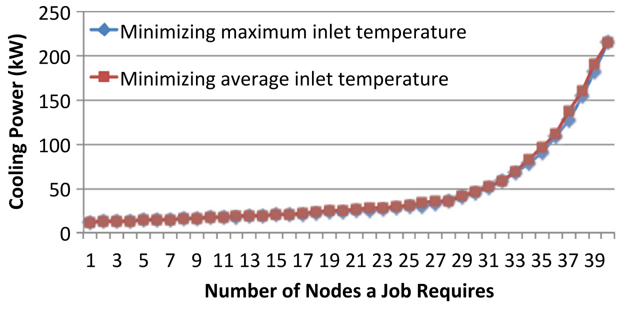


Fig. 4. Cooling power comparison of minimizing the maximum versus average inlet temperature.

In order to justify our choice of using the average inlet temperature in the formulation, we compare the cooling cost resulting from minimizing the maximum inlet temperature against the cost obtained from minimizing the average inlet temperature. As shown in Fig. 4, we repeat this comparison for different numbers of active nodes ranging from $n = 1$ to $n = 40$. When the utilization of the data center is less than 93%, the cooling energy difference between the two solutions is limited to 1.2% on average. The maximum difference is 9.9 kW, but such differences only appear when utilization exceeds 93%, as higher utilizations create larger maximum inlet temperatures. This error margin is acceptable as it does not affect the optimization results for the vast majority of our experiments. In addition, most data centers run at medium utilization levels. Even most HPC data centers and supercomputers largely operate below 90% utilization.

By integrating the formulations of the communication cost and the cooling energy cost from Eqs. (9) and (10) into the formulation of our joint optimization problem in Eq. (8), we obtain the following equations. As the constants in the goal function do not affect the optimization decisions, we simplify the equations and only use the quadratic part of the communication cost function and the linear part of the cooling cost function while computing the total cost.

$$\begin{aligned} \underset{X_{job}}{\text{minimize}} \quad & \frac{\alpha}{n} \cdot X_{job} \cdot H \cdot X_{job}^T \\ & + \frac{\beta}{U} \cdot sum\{D \cdot (P_{active} \circ X_{job})\} \\ \text{subject to} \quad & E \cdot X_{job} = n. \end{aligned} \tag{11}$$

Thus, we express the joint optimization problem as a binary quadratic programming problem, which is a combinatorial optimization problem. It is an NP-hard problem; however, in practice, it can be efficiently solved using well-known discrete optimization techniques such as the branch and bound algorithm [47]. In our experiments, we solve the optimization problem using the TOMLAB/CPLEX solver, which provides a Matlab interface to solve binary quadratic programming problems. For the joint optimization of cooling and performance, we set $\alpha = \beta = 0.5$.

Note that the problem is solved iteratively for each job arriving at the data center. In this way, our optimization formulation is suitable for handling dynamic job arrivals in real-life settings. When a job completes execution, the X_{job} vector is adjusted to revert the states of that job's nodes back into idle and the power vector P is modified.

4.3. Combining task mapping with job allocation

We integrate task mapping with the job allocation optimization to further reduce the communication cost based on the communication patterns of the applications. After the job allocation optimizer decides on the nodes for a job, the goal of task mapping is to decide which task of a job should be allocated on each specific node.

We perform task mapping using the patterns extracted for each HPC application (see Section 3.2). We assume the communication patterns observed at the PE-level in Section 3.2 are similar to the patterns at the node-level. Recall that our work focuses on node-level allocation and mapping; thus, each task fully occupies each of its nodes.

In our implementation, we apply a task mapping algorithm based on the recursive bisection mapping (RCB) algorithm proposed by Hoefler et al. [21]. In the RCB algorithm, the logical communication pattern of an application is represented using a weighted graph and the physical data center nodes with a certain network topology is presented using a separate graph. RCB algorithm follows a *divide and conquer* approach by recursively allocating the subsets of tasks to subsets of nodes. The algorithm starts with two graphs corresponding to all tasks to be mapped and all nodes in the system. At each step, the algorithm calls a graph bipartitioning algorithm which divides both graphs into two disjoint sub-graphs with minimum weighted edge-cuts. The recursion continues until a node graph is restricted to a single node. Prior work demonstrates successful application of the RCB algorithm in determining task mappings on mesh-based network topologies [36,21].

We modify the RCB algorithm proposed in [21] by adding an exhaustive search step to find the optimal solution of task mapping when the number of the nodes in the recursion is less than or equal to eight. The exhaustive search is expensive when the size of the job is large. However, when the number of the nodes to map the tasks is less than or equal to eight, the performance overhead of applying the search is negligible. We observe that adding this feature into the RCB algorithm reduces the communication cost considerably, without affecting its practical performance.

Note that the relationship between the communication cost model (H matrix) and the task-level communication patterns is orthogonal. The H matrix represents the impact of network hops on the communication delay and the communication pattern accounts for the intensity of communication. Thus, while evaluating the resulting communication cost with and without task mapping, we include both H matrix and communication patterns in our simulations. We calculate the communication cost as the 'weighted' sum of H matrix entries for the allocated tasks. The communication patterns shown in Fig. 2 determine the weights.

5. Experimental results

In this section, we present the experimental results for three different allocation strategies: cooling-aware, performance-aware, and our joint optimization technique. First, using a single-row of the modeled data center, we demonstrate the different job allocation decisions of the three strategies and how they affect the data center's performance and cooling energy consumption. Then, we experiment with multiple-row allocation and evaluate all three policies under dynamically changing workload. Finally, we demonstrate how the integration of task mapping and job allocation impacts the communication cost.

5.1. Job allocation in a single row

Fig. 5 presents the allocation decisions made by the cooling-aware, performance-aware, and our joint optimization algorithms

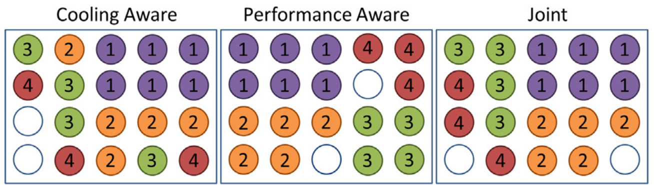


Fig. 5. Single row allocation comparing the three policies.

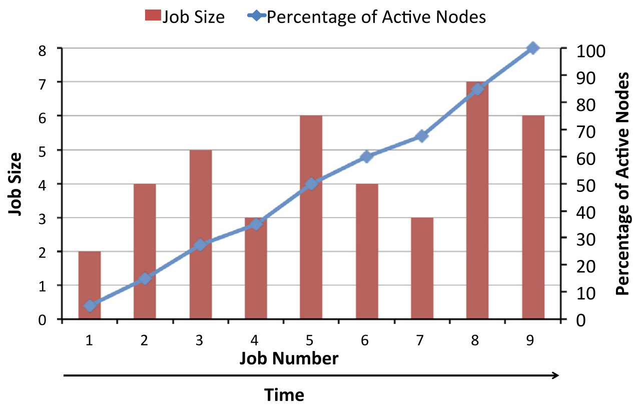


Fig. 6. Multiple row allocation: percentage of active nodes and job sizes (number of nodes a job requires).

in the single-row case, where four jobs are to be allocated sequentially. Each job requires six, five, four, and three nodes to run on. Numbered circles represent the busy nodes while the empty circles imply free nodes. The numbers indicate the arrival order of the jobs (e.g., job 1 arrives first and runs on six nodes).

We see that the cooling-aware policy assigns the jobs to the nodes starting from the right side of the data center and avoids the nodes that contribute more to heat recirculation. On the other hand, the performance-aware policy starts from an arbitrary node of the data center and tries to confine the allocated nodes for each job to the smallest area possible. The joint policy prefers the cooling-efficient nodes first and also assigns the tasks of a job in close proximity to each other.

5.2. Job allocation in multiple rows of the data center

In order to evaluate the job allocation policies across both rows of the data center, we use a sequence of nine jobs. Fig. 6 shows the percentage of the active nodes and the size of each job in terms of the number of nodes required. The maximum node inlet temperature changes between 23.7 °C (idle, 0% utilized) and 41.2 °C (100% utilized) within the data center. In Fig. 7, cooling power over time is given for the three allocation policies. We see that the joint policy closely follows the cooling-aware one and all policies converge at the 100% utilization point, reaching the same cooling power demands. However, performance-aware allocation causes a higher maximum inlet temperature even under medium utilization, resulting in a steeper cooling power curve.

We present the communication cost of each job in Fig. 8 and observe that our policy achieves performance levels that are very similar to the performance-aware policy. For some small-sized jobs such as 1 and 6, all policies result in very similar allocation decisions with an equal job communication cost. We see the benefits of better allocation decisions more clearly for larger jobs such as 5 and 8. We observe that, when the system utilization of the data center grows, the degree of freedom decreases and the last job is allocated to the remaining idle nodes, which explains the trend for job 9.

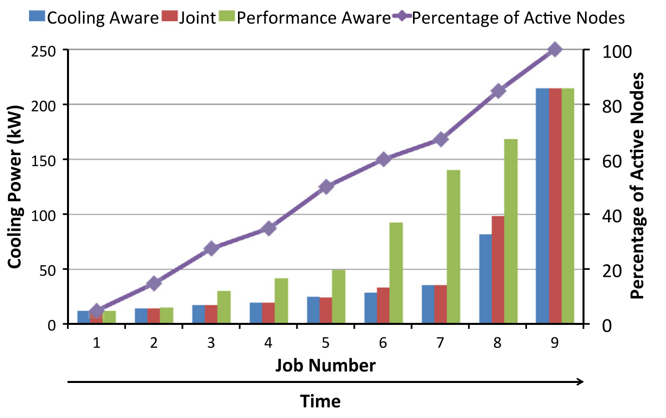


Fig. 7. Multiple row allocation: cooling power comparison among the three policies.

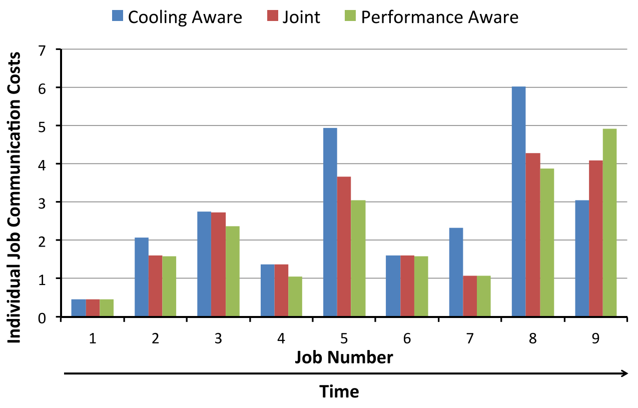


Fig. 8. Multiple row allocation: communication cost comparison among the three policies.

In comparison to the cooling-aware policy, our joint policy results in only 4% higher average cooling power while the exclusively performance-aware policy causes a 70% increase. On the other hand, our joint policy follows the performance-aware policy closely, bringing only 5% increase in the average communication cost per job. The cooling-aware policy increases the communication cost by 23%. Our policy sacrifices performance slightly compared to performance-aware allocation for substantial improvement in cooling efficiency.

Note that our results for the single and multiple-row allocation do not consider the impact of communication cost on the job execution time. In fact, larger communication costs may change the power-performance characteristics of the jobs, and thus, affect the total energy. Next, we investigate such interactions between performance, cooling power, and energy in detail.

5.3. Dynamic job allocation

To evaluate our policy for a dynamically changing workload scenario similar to real-life cases, we generate a job queue with job arrival times following an exponential distribution as in prior work [18]. We randomly generate the job sizes and running times, where each job occupies 1–16 nodes and runs for between 1 and 20 min. We use a job arrival rate of 20 jobs/hour.

In the dynamic job allocation experiment, when a job finishes execution, we update the data center status by freeing that job's nodes. We also adjust the power and running times of the jobs according to the communication latency to have a realistic model. We use the randomly generated job running times as the default running time for each job and assume a 30% communication to total running time ratio (as discussed in Section 3.3). After each job

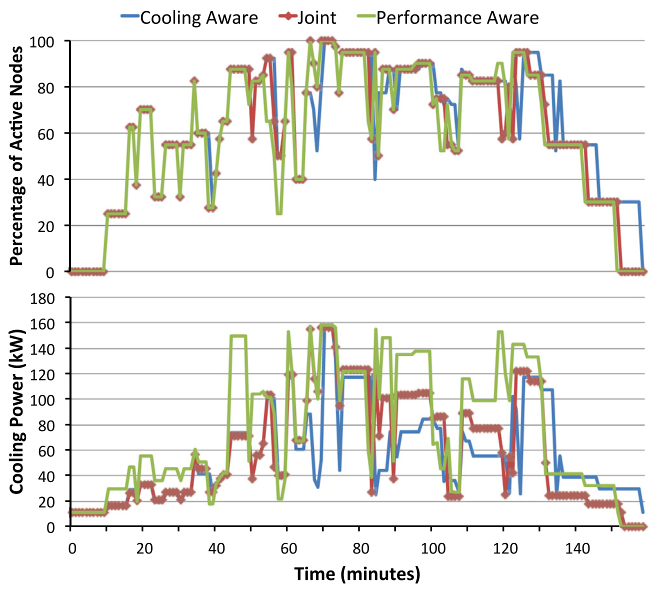


Fig. 9. Dynamic allocation: percentage of the active nodes in the data center (top) and the cooling power over time (bottom).

allocation, the resulting communication cost for that job is compared to the communication cost of the *best possible* (ideal) allocation for a job of that size. Using this relative delay compared to the best-case, we scale the communication portion of the job's running time. The adjustment in the job's running time is also reflected in the average power consumption computation as an adjustment on the communication power. Thus, there is a connection between the performance delay and the communication power, which results in different average power values depending on the allocation decision.

The allocation follows a first-come first-serve policy and if a sufficient number of free nodes are not available, we wait for some of the currently running jobs to finish. We simulate a total time of 2 h (40 jobs total) recording at each time step the maximum inlet temperature and the communication cost for each job. At each time step, the currently available node list, power values of active nodes, and the finishing time of the jobs are updated as needed. We repeat the same experiment 10 times (2 h each) using queues with the same arrival rates to increase the confidence of the results.

Fig. 9 shows an example trace of the system utilization (i.e., the percentage of active nodes) and the corresponding cooling power over time for all three allocation policies. We observe that the performance-aware and joint policies result in a similar utilization of around 60% on average. However, the cooling-aware policy causes a slightly higher utilization of 63% and results in a 7-min delay in the overall execution time of the job queue because of the higher communication latency it causes. The total execution time of the queue is similar among the performance-aware and joint policies.

Cooling power for our joint policy closely follows the power consumption of the cooling-aware policy. The simulated average cooling power for the 2-hour period is 56.4 kW and 80.9 kW for cooling-aware and performance-aware policies, respectively. Our joint policy achieves 65.4 kW average cooling power, which corresponds to 19.2% cooling power savings compared to the performance-aware allocation. The cooling energy consumption values of the data center are 148.6 kWh, 156.8 kWh, and 198.5 kWh for the cooling-aware, joint and performance-aware allocation policies, respectively.

We present the communication cost, running time increase, cooling power, and cooling energy of each of the 10 queues of the dynamic allocation experiment in Figs. 10–13. Fig. 10 compares

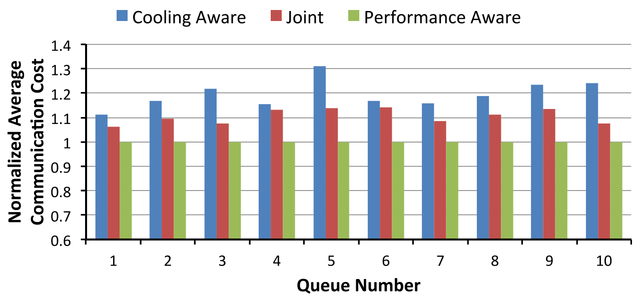


Fig. 10. Average communication cost of all jobs for each iteration of the experiment, normalized to the performance-aware case.

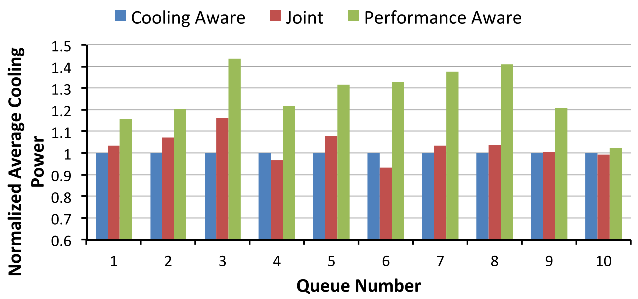


Fig. 12. Average cooling power for the 10 iterations of the experiment, normalized to the cooling-aware case.

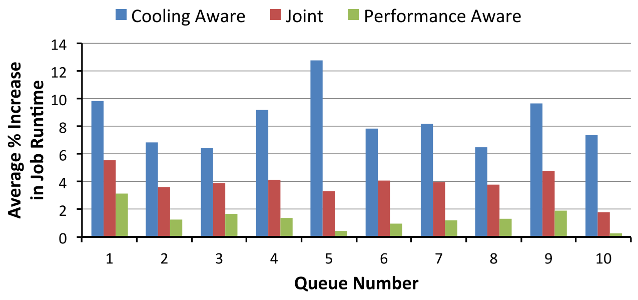


Fig. 11. Average increase in job running times for the 10 experiment iterations w.r.t. the ideal case.

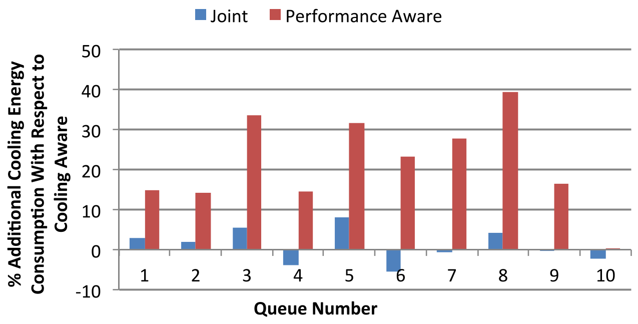


Fig. 13. Cooling energy consumption of the joint and performance-aware policies for the 10 iterations of the experiment, with respect to the cooling-aware case.

the average communication cost of all 40 jobs for each iteration. The results are normalized to the performance-aware case and we observe the benefits of using the joint policy in comparison to the cooling-aware policy for all the queues. On average, across all queues, the cooling-aware policy has 19.6% higher communication cost compared to the performance-aware policy. Our joint policy sacrifices 10.5% communication cost compared to the performance-aware policy for achieving higher cooling efficiency.

We present the average increase in job execution time for each policy in comparison to the ideal job running times in Fig. 11. We see that even the performance-aware policy results in a slight increase in job running times. This is because the job running times in the ideal case do not consider the utilization of the data center and the availability of the nodes. In other words, depending on the current utilization of the data center and the locations of the available nodes, performance-aware policy may not always guarantee an ideal allocation for each job. Our joint allocation policy increases the job running time by only 3.89% compared to the ideal case, and by 2.66% compared to the performance-aware case, on average. However, the cooling-aware policy results in considerably longer job running times, resulting in an up to a 12% increase. Intuitively, a lower communication to computation ratio will result in a lower increase in job execution times; however our joint policy can be applied to any scenario where jobs spend a non-negligible portion of their time in communication.

Fig. 12 compares the average cooling power for the three policies across all queues. As expected, we see that our joint policy results in an average cooling power close to the power consumption achieved by the cooling-aware policy. In comparison to solely performance-aware policy, joint allocation reduces cooling power by 18% on average, which corresponds to saving 16.4% cooling energy. An interesting observation is that for some cases, the joint policy achieves lower average cooling power than the cooling-aware policy such as in queues 4 and 6. This is because with the cooling-aware allocation, some jobs may take a longer time to run in comparison to the joint policy. When a large job is running for an extended period of time, it keeps the nodes active for a longer time. This results in higher total computing power and cooling power in comparison to the joint policy. Also, the following job in the queue may have to be allocated in a less efficient way as a fewer number of nodes are available.

Next, we present our analysis on cooling energy, taking into account the total time it takes to finish all the jobs in each queue. Fig. 13 compares the cooling energy consumption of joint and performance-aware policies with respect to the cooling-aware policy for each of the queues. Bars above zero means that additional cooling energy is consumed compared to the cooling-aware policy. We observe significant cooling energy benefits for the joint policy compared to the performance-aware case, which results in up to 40% additional cooling energy consumption. Moreover, our policy achieves up to 5% cooling energy savings compared to the cooling-aware case for queues 4, 6, 7, 9, and 10. These results are in parallel with the average cooling power results of Fig. 12 for the queues 4 and 6. For queues 7, 9, and 10, cooling energy benefits come from the shorter total execution time achieved by the joint policy. The joint policy results in only 1% higher cooling energy on average in comparison to cooling-aware optimization, while performance-aware policy results in 21.6% higher cooling energy consumption.

Note that one can apply our policy to arbitrary topologies once they extract the H matrix using the procedure described in Section 3.1. For fat tree topology for example, the H matrix will be more uniform. Thus, in the case of a fat tree, cooling cost would likely dominate, while for mesh topology both cooling and communication costs dominate. While our policy is applicable to arbitrary topologies, it is mainly designed for data centers with variable communication delays among different nodes (e.g., mesh).

The overhead of our joint optimization algorithm is negligible compared to the long running times of the jobs. We have measured the time spent running the allocation algorithm on Intel Xeon E5504 processor: for a data center with 40 nodes, the computation time to make a job allocation decision in our Matlab-based implementation is 0.0029 s. In order to test the scalability of our job allocation algorithm, we have measured the time spent on allocating the same jobs on data centers with node numbers ranging from 40 to 1000. Fig. 14 shows that as the size of data

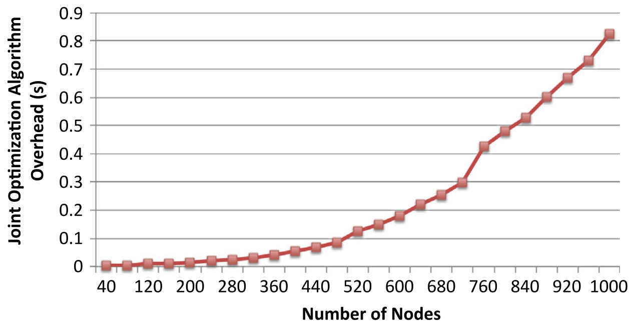


Fig. 14. The overhead of our joint optimization algorithm for data centers with nodes ranges from 40 to 1000.

center increases, the computation time of our joint optimization algorithm increases polynomially. For a data center with 1000 nodes, such as Cielo from Los Alamos National Laboratory [32] (number 26 on the 2013 Top500 list [46]), the execution time of our joint algorithm is 0.82 s.

We also compare our joint optimization algorithm with the heuristic approach proposed in our recent work [24]. We observe that for the same job queues used in this paper, our joint optimization algorithm achieves similar cooling cost to that of the heuristic algorithm proposed in [24] and provides up to 4.2% lower communication cost. The main advantage of the joint optimization algorithm is the considerably shorter computation time required to make the job allocation decision in comparison to our heuristic method [24]. For example, in our target data center with 40 nodes, the computation time for the job allocation is less than 0.01 s, which is much smaller than the several seconds overhead of the heuristic for the same data center.

5.4. *Task mapping*

In this section, we present the results of the RCB-based task mapping algorithm. We apply the task mapping algorithm after the joint policy decides on which nodes to allocate a job. Our baseline is an in-order allocation algorithm, which allocates the tasks of a job starting from the top left of the data center, traversing the assigned nodes from left to right and from top to bottom.

Fig. 15 compares the communication cost resulting from our RCB-based task mapping algorithm against the cost of the in-order algorithm for job sizes $n = 4, 8$ and 16, respectively, using the communication patterns provided in Section 3.2. In this experiment, one job is allocated on a fully idle data center at a time. For $n = 4$, we do not observe much benefit from RCB task mapping as the communication distance between the allocated nodes is already small. For FFT, which is dominated by all-to-all communication, we again do not observe benefits, as the communication cost is dominated by the job allocation rather than task mapping. RCB-based task mapping achieves a significant communication cost reduction for CharmLU, reaching up to 20.9% for a job size of 16. CharmLU has a lot of local communication; thus, placing the highly communicating tasks close to each other makes a substantial difference in the communication delay.

We next demonstrate the results for a dynamic allocation scenario, where the RCB-based task mapping is integrated into our joint allocation policy. We generate job queues in the same way as discussed earlier in this section, but this time, for each job, we also assign a communication pattern randomly selected among the 8 patterns in Section 3.2. We confine the job sizes to $n = 4, 8$ and 16. We perform the experiment on one of the rows of the data center, with a total of 20 nodes. Fig. 16 shows the reduction in the job communication cost in comparison to the baseline in-order policy for each of the 40 jobs in the queue. RCB policy achieves

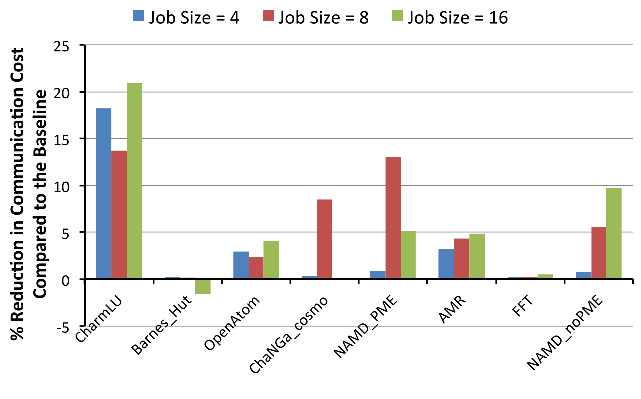


Fig. 15. Percentage reduction in the communication cost achieved by RCB-based mapping compared to in-order allocation for all the patterns and various job sizes.

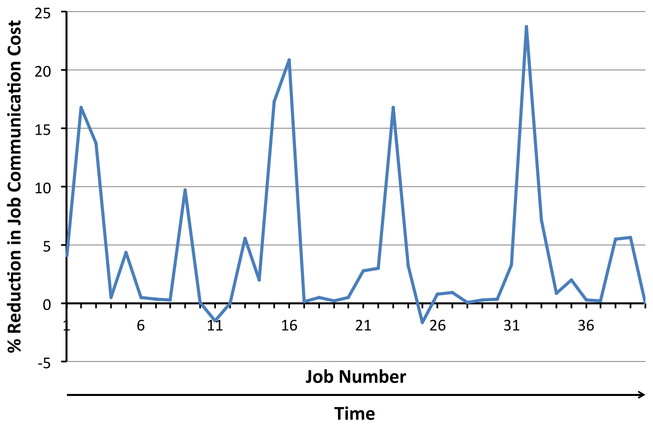


Fig. 16. Percentage reduction in job communication cost using RCB-based task mapping for the dynamic allocation scenario.

a higher reduction in communication cost (reaching up to a 25% decrease) for the CharmLU pattern, which corresponds to jobs 2, 3, 9, 16, 23 and 32. Similarly, RCB achieves a high cost reduction for NAMD_PME for job sizes larger than 4, which corresponds to job 15.

On the other hand, we do not see any improvement for FFT and Barnes_Hut patterns, which correspond to the data points with a zero value on the graph. This behavior is consistent with the results presented in Fig. 15. As discussed in Section 4, RCB does not apply exhaustive search for job sizes larger than 8. Thus, it may not always give the optimal communication cost, which explains the negative reduction (i.e., increase) numbers for jobs 11 and 25. We achieve an additional 4.3% reduction in communication cost on average across all the jobs for the dynamic scenario.

6. Conclusion

In this paper, we have proposed an optimization algorithm to both deliver the desired performance and to reduce the cooling energy through job allocation in data centers running communication-intensive jobs. We have provided a solution for the joint optimization problem based on binary quadratic programming. The experimental results demonstrate that, in comparison to performance-aware job allocation, our joint optimization algorithm reduces the cooling power by 18% on average with only a 2.66% increase in application running time. Our joint policy also results in comparable cooling energy to that of the cooling-aware allocation, while it achieves 16.4% average cooling energy reduction in comparison to the performance-aware allocation.

In order to further improve the performance of data centers running multi-threaded applications with heavy communication among their tasks, we have integrated an RCB-based task mapping algorithm into our job allocation methodology. In this way, our technique is able to consider application-specific communication patterns during task mapping. We have developed a set of tools in Charm++ to extract communication patterns from a set of real-life HPC applications. Our experiments show that by integrating task mapping into our joint optimization algorithm, we decrease the communication cost further by up to 20.9% in comparison to assuming all-to-all communication patterns.

Our future work will focus on more advanced aspects related to job allocation and HPC application performance. Some of them include the performance impact of allocation when running multiple jobs with different characteristics together. In such case, not only the network distance between communicating tasks, but also the network congestion caused by neighboring applications becomes an important contributor to performance. Another aspect is that the communication pattern or communication intensity of an application may change at runtime, which may affect the performance significantly. In order to model such effects in an accurate manner, we are currently looking into development of a more detailed simulation framework. Our simulation framework combines the job scheduling and allocation/task mapping components with a detailed network model including MPI patterns.

References

[1] Z. Abbasi, G. Varsamopoulos, S.K.S. Gupta, Tacoma: Server and workload management in Internet data centers considering cooling-computing power trade-off and energy proportionality, ACM Trans. Archit. Code Optim. 9 (2) (2012) 11:1–11:37. http://dx.doi.org/10.1145/2207222.2207227, URL http://doi.acm.org/10.1145/2207222.2207227.

[2] D. Abts, Cray xt4 and seastar 3-d torus interconnect, in: D.A. Padua (Ed.), Encyclopedia of Parallel Computing, Springer, 2011, pp. 470–477. URL http://dblp.uni-trier.de/db/reference/parallel/parallel2011.html#Abts11.

[3] N.R. Adiga, M.A. Blumrich, D. Chen, P. Coteus, A. Gara, M.E. Giampapa, P. Heidelberger, S. Singh, B.D. Steinmacher-Burow, T. Takken, M. Tsao, P. Vranas, Blue gene/l torus interconnection network, IBM J. Res. Dev. 49 (2) (2005) 265–276. http://dx.doi.org/10.1147/rd.492.0265.

[4] F. Ahmad, T. Vijaykumar, Joint optimization of idle and cooling power in data centers while maintaining response time, in: ACM Sigplan Notices, Vol. 45, 2010, pp. 243–256.

[5] A. Al-Qawasmeh, S. Pasricha, A. Maciejewski, H. Siegel, Thermal-aware performance optimization in power constrained heterogenous data centers, in: Parallel and Distributed Processing Symposium Workshops PhD Forum, IPDPSW, 2012 IEEE 26th International, 2012, pp. 27–40. http://dx.doi.org/10.1109/IPDPSW.2012.19.

[6] A. Banerjee, T. Mukherjee, G. Varsamopoulos, S.K. Gupta, Integrating cooling awareness with thermal aware workload placement for {HPC} data centers, Sustain. Comput.: Inform. Syst. 1 (2) (2011) 134–150. http://dx.doi.org/10.1016/j.suscom.2011.02.003, special Issue on Selected Papers from the 2010 International Green Computing Conference. URL http://www.sciencedirect.com/science/article/pii/S2210537911000229.

[7] J. Barnes, P. Hut, A hierarchical $O(n \log n)$ force-calculation algorithm, Nature 324 (4) (1986) 446–449.

[8] M.A. Bender, D.P. Bunde, E.D. Demaine, S.P. Fekete, V.J. Leung, H. Meijer, C.A. Phillips, Communication-aware processor allocation for supercomputers: Finding point sets of small average distance, Algorithmica 50 (2) (2008) 279–298. http://dx.doi.org/10.1007/s00453-007-9037-2.

[9] A. Bhatelé, L. Kale, Heuristic-based techniques for mapping irregular communication graphs to mesh topologies, in: IEEE International Conference on High Performance Computing and Communications, HPCC, 2011, pp. 765–771.

[10] S. Bhattacharya, W.-T. Tsai, Lookahead processor allocation in mesh-connected massively parallel multicomputer, in: International Parallel Processing Symposium, 1994, pp. 868–875.

[11] E. Bohm, A. Bhatelé, L. Kale, M. Tuckerman, S. Kumar, J. Gunnels, G. Martyna, Fine grained parallelization of the car-parrinello ab initio molecular dynamics method on blue gene/l, IBM J. Res. Dev. 52 (1–2) (2008) 159–175.

[12] Y. Chen, D. Gmach, C. Hyser, Z. Wang, C. Bash, C. Hoover, S. Singhal, Integrated management of application performance, power and cooling in data centers, in: Network Operations and Management Symposium, NOMS, 2010 IEEE, 2010, pp. 615–622. http://dx.doi.org/10.1109/NOMS.2010.5488433.

[13] P.-J. Chuang, N.-F. Tzeng, An efficient submesh allocation strategy for mesh computer systems, in: IEEE International Conference on Distributed Computing Systems, 1991, pp. 256–263.

[14] I.-H. Chung, C.-R. Lee, J. Zhou, Y.-C. Chung, Hierarchical mapping for hpc applications, in: IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, IPDPSW, 2011, pp. 1815–1823.

[15] M. Crovella, R. Bianchini, T. Leblanc, E. Markatos, R. Wisniewski, Using communication-to-computation ratio in parallel program design and performance prediction, in: IEEE Symposium on Parallel and Distributed Processing, 1992, pp. 238–245.

[16] F. Gioachin, S. Chakravorty, C. Mendes, L. Kale, T. Quinn, Cosmological simulations on supercomputers, in: International Conference on High Performance Computing, Networking, Storage and Analysis, 2006.

[17] I.n. Goiri, W. Katsak, K. Le, T.D. Nguyen, R. Bianchini, Parasol and greenswitch: Managing datacenters powered by renewable energy, in: Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS'13, ACM, New York, NY, USA, 2013, pp. 51–64. http://dx.doi.org/10.1145/2451116.2451123, URL http://doi.acm.org/10.1145/2451116.2451123.

[18] T.J. Hacker, K. Mahadik, Flexible resource allocation for reliable virtual cluster computing systems, in: SC, 2011, pp. 48–48.

[19] T. Heath, A.P. Centeno, P. George, L. Ramo, Y. Jaluria, R. Bianchini, Mercury and freon: temperature emulation and management for server systems, in: ASPLOS, 2006, pp. 106–116.

[20] G. Hendry, A. Rodrigues, SST: A simulator for exascale co-design, in: ASCR/ASC Exascale Research Conference, 2012.

[21] T. Hoefler, M. Snir, Generic topology mapping strategies for large-scale parallel architectures, in: Proceedings of the International Conference on Supercomputing, 2011, pp. 75–84.

[22] L.V. Kalé, S. Krishnan, Charm++: A portable concurrent object oriented system based on c++, in: Proceedings of the Eighth Annual Conference on Object-oriented Programming Systems, Languages, and Applications, OOPSLA'93, ACM, New York, NY, USA, 1993, pp. 91–108. http://dx.doi.org/10.1145/165854.165874, URL http://doi.acm.org/10.1145/165854.165874.

[23] L. Kalé, A. Sinha, Projections: A preliminary performance tool for Charm, in: Parallel Systems Fair, International Parallel Processing Symposium, Newport Beach, CA, 1993, pp. 108–114.

[24] F. Kaplan, J. Meng, A.K. Coskun, Optimizing communication and cooling costs in HPC data centers via intelligent job allocation, in: Proceedings of the International Green Computing Conference, IGCC, 2013.

[25] M. Khan, M. Herbordt, Communication requirements for FPGA-centric molecular dynamics, in: Symposium on Application Accelerators for High Performance Computing, 2012.

[26] J. Kim, M. Ruggiero, D. Atienza, Free cooling-aware dynamic power management for green datacenters, in: International Conference on High Performance Computing and Simulation, HPCS, 2012, pp. 140–146.

[27] K. Le, R. Bianchini, M. Martonosi, T. Nguyen, Cost-and energy-aware load distribution across data centers, Proceedings of HotPower.

[28] V. Leung, E. Arkin, M. Bender, D. Bunde, J. Johnston, A. Lal, J. Mitchell, C. Phillips, S. Seiden, Processor allocation on cplant: achieving general processor locality using one-dimensional allocation strategies, in: IEEE International Conference on Distributed Computing Systems, 2002, pp. 296–304.

[29] L. Li, W. Zheng, X. Wang, X. Wang, Coordinating liquid and free air cooling with workload allocation for data center power minimization, in: 11th International Conference on Autonomic Computing, (ICAC 14), USENIX Association, Philadelphia, PA, 2014, pp. 249–259. URL https://www.usenix.org/conference/icac14/technical-sessions/presentation/li_li.

[30] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, C. Hyser, Renewable and cooling aware workload management for sustainable data centers, in: Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS'12, ACM, New York, NY, USA, 2012, pp. 175–186. http://dx.doi.org/10.1145/2254756.2254779, URL http://doi.acm.org/10.1145/2254756.2254779.

[31] C. Lively, X. Wu, V. Taylor, S. Moore, H.-C. Chang, K. Cameron, Energy and performance characteristics of different parallel implementations of scientific applications on multicore systems, J. High Perform. Comput. Appl., Vol. 25, no. 3.

[32] Los Alamos National Laboratory, High-performance computing: Cielo supercomputer, http://www.lanl.gov/orgs/hpc/cielo/.

[33] J. Mache, V. Lo, K. Windisch, Minimizing message-passing contention in fragmentation-free processor allocation, in: International Conference on Parallel and Distributed Computing Systems, 1997, pp. 120–124.

[34] J. Moore, J. Chase, P. Ranganathan, R. Sharma, Making scheduling "cool": temperature-aware workload placement in data centers, in: Proceedings of the annual conference on USENIX Annual Technical Conference, 2005, pp. 5–5.

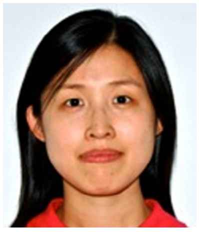
[35] E. Pakbaznia, M. Pedram, Minimizing data center cooling and server power costs, in: International Symposium on Low Power Electronics and Design, 2009, pp. 145–150.

[36] F. Pellegrini, Static mapping by dual recursive bipartitioning of process architecture graphs, in: Scalable High-Performance Computing Conference, 1994., Proceedings of the, 1994, pp. 486–493.

[37] J. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. Skeel, L. Kale, K. Schulten, Scalable molecular dynamics with NAMD, J. Comput. Chem. 26 (2005) 1781–1802.

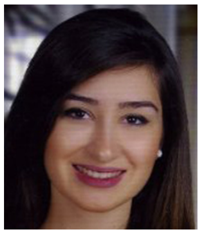
[38] A. Sansottera, P. Cremonesi, Cooling-aware workload placement with performance constraints, Perform. Eval. 68 (11) (2011) 1232–1246.

[39] O. Sarood, P. Miller, E. Totoni, L.V. Kale, Cool load balancing for high performance computing data centers, IEEE Trans. Comput. 61 (12) (2012) 1752–1764.
[40] R. Sawyer, Calculating total power requirements for data centers, White Paper, American Power Conversion.
[41] M. Sayeed, H. Bae, Y. Zheng, B. Armstrong, R. Eigenmann, F. Saied, Measuring high-performance computing with real applications, Comput. Sci. Eng. 10 (4) (2008) 60–70.
[42] M. Stansberry, Uptime institute 2013 data center industry survey, Tech. rep, Uptime Institute, 2013.
[43] V. Subramani, R. Kettimuthu, S. Srinivasan, J. Johnston, P. Sadayappan, Selective buddy allocation for scheduling parallel jobs on clusters, in: IEEE International Conference on Cluster Computing, CLUSTER, 2002, p. 107–116.
[44] Q. Tang, S.K.S. Gupta, G. Varsamopoulos, Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach, IEEE Trans. Parallel Distrib. Syst. 19 (11) (2008) 1458–1472.
[45] Q. Tang, T. Mukherjee, S. Gupta, P. Cayton, Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters, in: International Conference on Intelligent Sensing and Information Processing, ICISIP., 2006, pp. 203–208.
[46] Top 500 List, Top 500 list—November 2013, http://www.top500.org/lists/2013/11/.
[47] H. Trinh, Q. Fan, P. Gabbur, S. Pankanti, Hand tracking by binary quadratic programming and its application to retail activity recognition, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2012, pp. 1902–1909.
[48] UIUC Parallel Programming Laboratory, Charm++: MiniApps, http://www.charmplusplus.org/benchmarks.
[49] A. Venkatraman, Global census shows datacentre power demand grew 63% in 2012, October 2012. http://tinyurl.com/mhq2up8.
[50] P. Walker, D. Bunde, V. Leung, Faster high-quality processor allocation, in: Proceedings of the 11th LCI International Conference on High-Performance Clustered Computing, 2010.
[51] L. Wang, G. von Laszewski, J. Dayal, X. He, A. Younge, T. Furlani, Towards thermal aware workload scheduling in a data center, in: International Symposium on Pervasive Systems, Algorithms, and Networks, ISPAN, 2009, pp. 116–122.
[52] J. Wu, Z. Lan, X. Xiong, N.Y. Gnedin, A.V. Kravtsov, Hierarchical task mapping of cell-based AMR cosmology simulations, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC, 2012, pp. 75:1–75:10.
[53] H. Xu, C. Feng, B. Li, Temperature aware workload management in geo-distributed datacenters, in: Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS'13, ACM, New York, NY, USA, 2013, pp. 373–374. http://dx.doi.org/10.1145/2465529.2465539, URL http://doi.acm.org/10.1145/2465529.2465539.



Jie Meng received her Ph.D. degree in Electrical Engineering from Boston University in 2013. She is currently a software engineer at CGG. Her research interests include modeling and simulations for energy-efficient high performance computing systems.

Eduard Llamosí (SM'13) received the M.S. degree in Computer Science and Information Technology from the Technical University of Catalonia, Barcelona, Spain (2014), having developed his Master's Thesis as a visiting student at Northeastern University. At present, he is a Software Engineer at Zhilabs, Barcelona, Spain. His research interests include massively parallel systems, big data technologies and reliable software design.



Fulya Kaplan is currently a Ph.D. candidate in the Department of Electrical and Computer Engineering, Boston University, Boston, M.A. She received her B.S. degree in Electrical and Electronics Engineering from the Middle East Technical University, Turkey, in 2011. Her research interests are temperature and energy management in multicore processors and data centers. She is a member of the IEEE.

Chulian Zhang received his Bachelor degree in Electrical Engineering from Beijing University of Posts and Telecommunications, Beijing, China (2011). At present, he is pursing Master degree of Computer Engineering in Northeastern University, Boston, USA. His research interest includes Computer Architecture and GPGPU.

Jiayi Sheng received his B.S. from School of Microelectronics of Fudan University. He currently is a Ph.D. candidate in the Department of Electrical and Computer Engineering at Boston University. His research focus includes Computer Architecture and High Performance Computing.

Martin Herbordt received the Ph.D. in Computer Science from the University of Massachusetts. He is currently a professor in the Department of Electrical and Computer Engineering at Boston University with research interests in computer architecture and high performance computing.

Gunar Schirner (S'04–M'08) holds Ph.D. (2008) and MS (2005) degrees in Electrical and Computer Engineering from the University of California, Irvine. He is currently an associate professor in Electrical and Computer Engineering at Northeastern University. His research interests include communication systems analysis and modeling, embedded system modeling, system-level design, and the synthesis of embedded software.

Ayse K. Coskun (M06) received the M.S. and Ph.D. degrees in Computer Science and Engineering from the University of California, San Diego. She is currently an associate professor at the Department of Electrical and Computer Engineering, Boston University (BU). She was with Sun Microsystems (now Oracle), San Diego, prior to her current position at BU. Her current research interests include energy-efficient computing, 3-D stack architectures, computer architecture, and embedded systems and software.