Optimizing Communication and Cooling Costs in HPC Data Centers via Intelligent Job Allocation

Fulya Kaplan, Jie Meng, and Ayse K. Coskun
Electrical and Computer Engineering Department, Boston University, Boston, MA
{fkaplan3, jiemeng, acoskun}@bu.edu

***Abstract*—Nearly half of the energy in the computing clusters today is consumed by the cooling infrastructure. It is possible to reduce the cooling cost by allowing the data center temperatures to rise; however, component reliability constraints impose thermal thresholds as failure rates are exponentially dependent on the processor temperatures. Existing thermally-aware job allocation policies optimize the cooling costs by minimizing the peak inlet temperatures of the server nodes. An important constraint in high performance computing (HPC) data centers, however, is performance. Specifically, HPC data centers run multi-threaded applications with significant communication among the threads. Thus, performance of such applications is strongly affected by the job allocation decisions. This paper proposes a novel job allocation methodology to jointly minimize communication cost of an HPC application while also reducing the cooling energy. The proposed method also considers temperature-dependent hardware reliability as part of the optimization.**

I. Introduction

The computational capacities of the data centers have been growing over the last decade. In tandem, the electricity consumed for the computational and cooling power has also increased. According to a recent report, the electricity used by the data centers worldwide increased by 56% from 2005 to 2010, while the increase in the US was by 36% [1]. In 2010, the worldwide data center electricity consumption accounted for 1.3% of the total electricity use [1]. This number translates to an energy cost reaching up to millions of dollars and cooling costs reach close to half of the overall energy cost [2]. Thus, one of the main challenges in maintaining an energy-efficient data center is to design efficient cooling methods.

Some of the prior work addresses the cooling challenges with various cooling-aware job allocation policies (e.g., [3], [4], [5]). One of the key factors in efficient cooling is to keep the computer room air conditioner (CRAC) supply temperature as high as possible [3], while delivering sufficient cooling to the computational nodes. Thus, a number of techniques focus on decreasing the node inlet temperatures through workload management. Recirculation of the heat from the servers back in the data center has been shown to be a significant contributor to inlet temperature; thus, some policies focus on reducing the recirculation effect [3], [4]. Other approaches dynamically control the number of active servers and the voltage/frequency settings to reduce the total power [5].

Another critical aspect in data center management is performance. In HPC clusters, highly parallel scientific, financial, or other applications run on multiple nodes for long durations in the range of minutes, hours or days. The threads of these applications communicate with each other through communication infrastructures such as the message passing interface (MPI). The running time of a communication-intensive application is highly dependent on the location of the individual computing units that are communicating with each other. Prior work has demonstrated that the communication cost of communication-intensive applications has a significant impact on system performance in HPC data centers [6]. Existing performance-aware job allocation algorithms focus on minimizing the average number of communication hops between the communicating nodes. Some researchers propose contiguous job allocation schemes, for which the allocated nodes are adjacent [7], [8]. Other schemes allow discontiguous allocation to avoid fragmentation of available processors [9], [10]. There are also algorithms that minimize the power consumption while satisfying the response time constraints imposed by service level agreements [11]; however, such algorithms focus on transactional enterprise loads rather than HPC applications with intensive communication.

We observe that existing algorithms for job allocation in HPC data centers address cooling efficiency and performance separately. How to optimize the performance and cooling energy tradeoffs achieved by these policies is currently an open question. To the best of our knowledge, our work is the first to propose a policy that reduces both cooling power and communication latency in an HPC data center. We target HPC data centers that run multi-threaded workloads with heavy communication among the threads. Our main goal is to deliver both the desired performance and reduce cooling energy as much as possible. Our specific contributions are as follows:

- We introduce a job allocation methodology to jointly optimize the communication cost of HPC applications and the cooling energy in a data center.
- We evaluate our joint allocation policy and compare against other cooling-aware and performance-aware job allocation methods under both static (known) and dynamically changing workloads. We observe that the policies only targeting cooling efficiency may cause higher communication latency, which would then lead to high temperatures in data centers lasting for an extended period of time due to the longer job running times (i.e., longer active time of the data center nodes). On the other hand, performance-aware job allocation policies which do not consider the thermal impact may result in higher cooling energy. Our policy combines performance-awareness with cooling-awareness to solve this problem.
- Under dynamically changing workloads, we show that our policy achieves 40% average cooling power savings compared to the baseline policies when running HPC applications that spend 20% of their time in communication.

- We also provide a mechanism in our policy to enable the administrators or users to impose target reliability constraints. Our policy is able to keep the temperature-related hardware reliability over a desired value.

The rest of the paper starts with a discussion of the related work. Section III presents the performance, temperature, cooling power, and reliability models. We introduce the performance and cooling optimization problems separately and then describe our joint allocation strategy in Section IV. We also explain how to optimize allocation with reliability constraints in the same section. We provide the experimental evaluation in Section V and conclude in Section VI.

II. Related Work

This section reviews the related work on data center job allocation algorithms for optimizing performance and cooling energy costs. In addition, we discuss the related research on reliability of computer systems and how it affects the job allocation decisions.

A. Performance-Aware Job Allocation

Performance-aware job allocation algorithms for data centers and supercomputers typically focus on minimizing the average number of communication hops between processors on which a job is running. Some of the job allocation algorithms allocate only *contiguous* (i.e., touching, in close proximity) sets of processors to each job (e.g., [7], [8], [12]), as contiguous node allocation provide significant reduction in execution time for communication-intensive parallel programs. For example, Bhattacharya et al. propose a heuristic for job allocation in a mesh-connected parallel processor. They use a lookahead idea by analyzing the queue of waiting jobs and propose an algorithm to detect free submesh area for efficient allocation [7]. However, such contiguous allocation algorithms may result in external fragmentation of available processors (i.e., available nodes that are separated from each other cannot be utilized) and reduce the achievable system utilization.

A number of recent proposed algorithms on communication-aware job allocation allow *discontiguous* allocation of processors (i.e., the processors given to a job do not need to be next to each other). For example, 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 [9]. Motivated by the MC allocation strategy [9], 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 [10]. Walker et al. discuss fast algorithms to allocate processors to compute jobs in mesh-connected clusters [13], where they introduce a curve-based allocation (processors are ordered according to some curve) and propose several buddy-system strategies (uses a data structure to organize free processors). Existing performance-aware job allocation strategies solely target the performance and communication costs without considering the potential impact of job allocation on the cooling costs.

B. Thermal Modeling and Cooling Energy Management

As thermal management and reducing the cooling costs are among the dominant concerns for today's data centers, a number of thermal modeling and management techniques at data center level have been proposed recently. Jungsoo et al. use the linear formula with parameters ambient room temperature, thermal resistance between die and air, and server power to find server temperatures. However, their model does not consider the effect of recirculation on temperature [14]. Wang et al. use another model in which they compute the temperature of a node as a combination of an RC-thermal model and a task-temperature profile [15]. However, this model assumes that the thermal map of the data center is available through input ambient sensors and on-board sensors. Moore et al. carry out CFD simulation to conduct thermal evaluation, which can not be used for online real-time datacenter thermal management [3]. Heath et al. introduce a data center temperature emulation suite called Mercury that emulates temperatures based on the data center layout, hardware, and component utilizations [16]. Despite its advantages of accuracy and efficiency, Mercury has not been validated for large data center systems. Tang et al. propose a linear model to compute data center temperatures and cooling energy costs, and solve an optimization problem for minimizing the peak node inlet temperature (MPIT) through job assignment [4]. They use both genetic algorithm and sequential quadratic programming to solve the problem.

Moore et al. present two temperature-aware workload placement algorithms: the first one assigns workloads to servers based on location-aware discretization heuristics and the second one minimizes the heat recirculating within the data centers [3]. Pakbaznia et al. propose Minimum Total Data Center Power (MTDP) algorithm to minimize the total of server and cooling power in a data center by turning off some of the servers and chassis and deciding on the voltage/frequency setting for the servers [5]. However, these techniques focus on reducing temperature and cooling cost of data centers without considering the impact of the workload allocations on application performance. Sansottera et al. proposes the Greedy Least Power (GLP) algorithm to minimize the power consumption while satisfying response time constraints imposed by service level agreements [11]. However, their main focus is not HPC applications with intensive communication, so their model does not include the communication latency during allocation.

C. Reliability-Aware Workload Management

Recent work on reliability-aware job allocation and management techniques mainly targets multi-chip multicore systems. Wang et al. present a technique to optimize the lifetime of a multicore system by utilizing a sequential quadratic programming method while maintaining a given aggregate processor speed [17]. Meng et al. propose a workload allocation policy to optimize the reliability of multi-chip multicore systems considering the series/parallel connections of processors and servers in a data center [18]. These methods perform reliability-aware task allocation in the multi-chip multicore systems; however, they do not investigate the temperature-dependent reliability at the data center level. To improve the reliability of HPC clusters, Hacker et al. propose scheduling policies and corresponding analytical resource prediction models for estimating the size of the virtual cluster system needed to provide a reliable service for a realistic HPC workload. They model virtual cluster reliability as a Weibull distribution based on the kernel logs of a cluster [19].

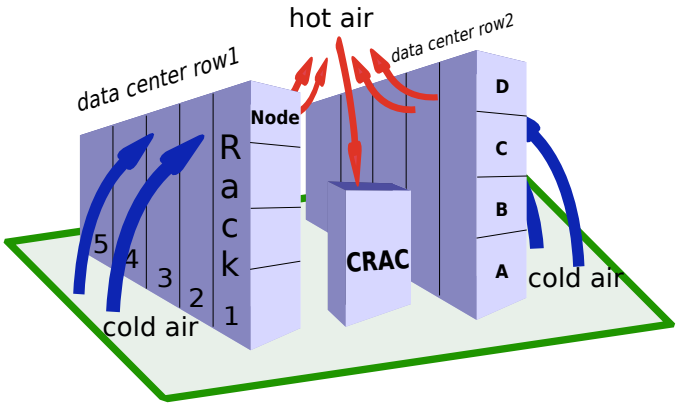


Fig. 1: Layout of the data center.

D. Distinguishing Aspects from Prior Work

Our work differentiates from prior research as our job allocation policy optimizes both the application performance (in terms of the communication cost) and the cooling energy cost of HPC data centers. Our policy confines the communicating nodes of a job in close proximity, but it also selects the most cooling-efficient locations possible. We implement and show the impact of prior performance-aware (MC1X1) and cooling energy-aware job allocation policies. We demonstrate that for jobs involving intensive communication between the nodes, application performance becomes an important factor in determining the cooling power. We also include processor reliability as a constraint in our policy.

III. Experimental Methodology

In this paper, we model a small size data center with 2 rows of industry standard racks arranged in a layout shown in Figure 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 5 racks and each rack has 4 compute *nodes*. In our experiments, we assume that each *node* includes 10 servers and each server has 2 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 widely used in prior work and is representative of today's data center configurations [11]. Our methodology and proposed policy are scalable to larger data centers.

A. Workload and Performance Model

In this paper, we target communication-intensive parallel applications that use high-level message passing interfaces such as MPI. For such workloads, the communication overhead inherent in the data center is one of the major performance bottlenecks [9]. We evaluate our job allocation algorithms for mesh-connected HPC data centers and supercomputing systems. Mesh-connected networks for message passing are widely used in many experimental and commercial distributed memory parallel computers, such as IBM BlueGene/L and Cplant, a commodity-based supercomputer developed at Sandia National Laboratories.

We specify our workloads as jobs that require a number of nodes in the data centers. The performance metric for our evaluation is the average pairwise L1 distance (Manhattan distance) across all the communicating nodes of a job running on the mesh-connected parallel system [10]. We employ L1 distance as our metric as it has been demonstrated to correlate with application running time [6]. We call the total L1 distance among the nodes of a job the *communication cost* and formulate it as follows:

$$CC_{job} = \frac{1}{n} \sum_{(s,t)\in(S,T)} [w_x(s,t) + w_y(s,t)] \quad (1)$$

where n is the job size (the number of nodes a job requires), (s,t) is the pair of source and destination nodes of a message and (S,T) is the set of all the source and destination node pairs for all the messages. In this work, we assume $n > 1$ for all jobs. $w_x(s,t)$ and $w_y(s,t)$ represent the distance between s to t along the x-axis and y-axis, respectively, as shown in Figure 2(b). Division by n provides the normalization with respect to job size; thus, CC_{job} gives the communication cost of a job per node.

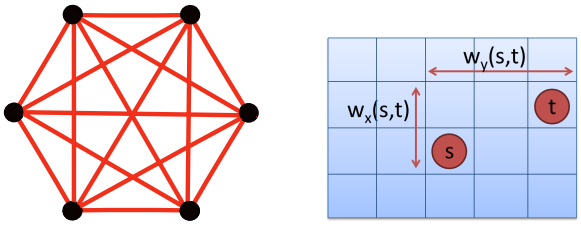
We assume *all-to-all* communication pattern for our workloads. In *all-to-all* pattern, each processor communicates with all the other processors running the same job, as shown in Figure 2(a). *All-to-all* is a common communication pattern in HPC routines such as Fast-Fourier-Transform, which is part of several applications including molecular dynamics, quantum chemistry, and digital signal processing [20].

For our experiments, we set the one-hop distance within a data center row as 1 and the distance between nodes of different rows as 10. The reason for the larger distance among rows is that nodes placed at different rows communicate through a larger number of switches and longer interconnects on the communication path [21]. Thus, this effect should be included in the communication latency calculation.

In our simulations, we reflect the effect of job communication cost on job runtime. We assume that the application spends a certain percentage of time on communication, denoted as C% [22]. As a baseline for a job's communication cost, we take the minimum CC_{job} that can be accomplished for a given job size. For example, for a job of size 4, the minimum achievable communication cost is 4 using Equation 1. We then define the ratio of the current CC_{job} to the best case CC_{job} as the latency factor, L_f. We calculate the actual job runtime by scaling the communication portion of the runtime by L_f.

B. Cooling Energy Model

We use a typical data center layout validated by prior work [23] (See Figure 1). In this layout, racks and perforated vent tiles are placed on a raised floor. Cold air enters the room



(a) All-to-all pattern (b) Distance metrics

Fig. 2: Communication pattern and distance measure.

from the floor tiles, goes into the rack inlets from the sides, and gets hotter as it moves through the racks. Hot air exits the rack from the back into the center aisle and the exhaust air exists the room from the ceiling to be cooled again. This set-up is called hot aisle/cold aisle arrangement which avoids mixing cold supply air with exhaust air.

To be able to compare different job allocation strategies, we need a fast and accurate data center thermal model. We use the model proposed and validated by Tang et al [24]. Their model combines a linear, low complexity heat recirculation model with a linear power model. This model is more practical than most other existing models as it requires a set of computational fluid dynamics (CFD) simulations only once to characterize the data center. Once we have the measured data center specific parameters, the vector of inlet temperatures, $\boldsymbol{T_{in}}$, for all the nodes are computed using the following linear equation:

$$\boldsymbol{T_{in}} = \boldsymbol{T_{sup}} + \boldsymbol{DP} \tag{2}$$

$$\boldsymbol{D} = [(\boldsymbol{K} - \boldsymbol{A^T K})^{-1} - \boldsymbol{K^{-1}}] \tag{3}$$

where T_{sup} is the CRAC unit supply temperature vector, **D** is the heat distribution matrix and **P** is the node power vector. **K** is the thermodynamic constant matrix and **A** is the heat cross-interference coefficient matrix representing the recirculation phenomena. **K** is calculated as:

$$\boldsymbol{K} = diag(K_i) \tag{4}$$

$$K_i = \rho f_i c_p \tag{5}$$

where ρ=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/KgK is the specific heat of air [4].

Matrix **A** represents the fraction of output heat from each node that is recirculated to the inlet of 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. It has been shown that elements of matrix **A** mostly depend on the data center layout rather than the power consumption of the nodes or the supply temperature [11]. Therefore, this matrix is obtained once for a data center. The matrix is calculated through CFD simulations in prior work [24]. If one has input ambient sensors mounted already, **A** can be obtained using sensor measurements instead of CFD simulations and following the same procedure in [24]. We use the coefficients for the data center given in [11]. We insert the colormap plot given in [11] into Matlab to extract the coefficient value corresponding to each data point in the image. We map RGB values to indexes, which preserve the relationship between coefficients relative to each other. Next, we perform calibration by scaling the matrix according to the given temperature graph in [11]. Figure 3 shows the matrix **A** for the 40-node system in a 3-D plot. For a data center with different layout and heat flow characteristics, matrix **A** differs. However, the equations to calculate the inlet temperatures are independent of the data center; in other words, the model applies to data centers in general.

For a node *j*, $\sum_{i=1}^{n} a_{ij}$ is called the recirculation coefficient (RC) of node *j* and is a measure of the total recirculation effect of that node [24]. On the other hand, for a node *i*, the value of $(1 - \sum_{j=1}^{n} a_{ij})$ is called the exit coefficient (EC) of

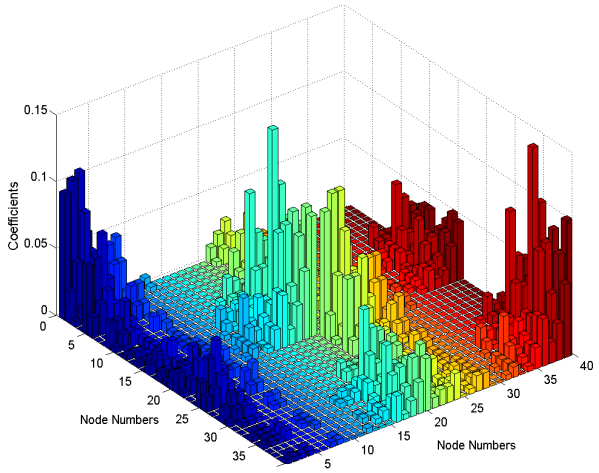


Fig. 3: Cross-interference coefficient matrix for our system.

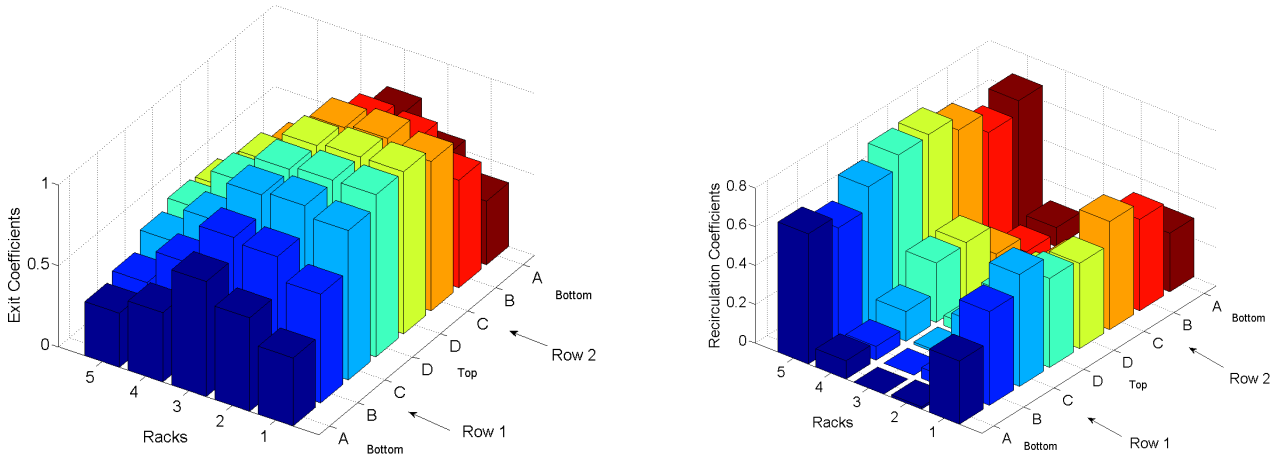
node *i*. EC is a measure of the heat at node *i*'s outlet returning back to the cooling system without recirculating back to other nodes. EC and RC for our system are given in Figure 4. As seen from Figure 4(a), the nodes at the bottom of the racks and at the ends of the aisles have lower EC values, meaning they contribute more to the recirculation effect. Moreover, according to Figure 4(b), the nodes at the top and at the ends of the aisle have higher RC values, which means they are more affected (or victimized) by the recirculated heat. The asymmetry between EC and RC values of right and left end of the aisles is due to asymmetries in the heat flow within the data center. Our policy considers the specific thermal behavior of the given data center and takes the differences in EC and RC into account during cooling optimization.

Next, we describe how the node powers are calculated based on job allocation. These power values are used in Equation (2) 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 HPC data centers. Once a job is assigned to a node of multiple servers, server and core level workload allocation will follow. Assume a given task of size *n*, which corresponds to the total number of nodes a task requires, x_i is an integer variable showing whether node *i* is assigned a job or not and it is either 1 or 0, respectively. Power consumption of node *i* can be expressed with a linear model as follows:

$$P_i = P_{idle} + x_i P_{util} \tag{6}$$

where P_{idle} is the node idle power and P_{util} is the power consumed by a node when running a task. We use 1000 W for P_{idle} and 2500 W for P_{util}. Therefore, for a node (composed of 10 servers) running a job, the total power is 3500 W. These numbers are in line with the server power values given in [11].

We adjust the total node power according to the actual runtime and percentage of time spent in communication. During communication intensive phase, power consumption will be lower due to the time spent waiting for messages. We assume 2 different power levels, 3500 W for computation phase and 2700 W during communication phase. These numbers are in line with the values in [25]. We set the total power of a node as the weighted sum of the computation power and communication power. Communication level (C%) or the power levels corresponding to computation/communication phases depend



(a) Exit Coefficients

(b) Recirculation Coefficients

Fig. 4: Exit and recirculation coefficients for our system.

on the workload and power characteristics of the system. However, the policy is applicable to systems with different power and communication levels.

Once we get the node inlet temperatures using Equation (2), we need a cooling power model to measure the power consumed by the cooling unit at various temperatures. This power depends on the efficiency of the CRAC unit. One of the most common metrics used for CRAC unit efficiency is the coefficient of performance (CoP). CoP is defined as the ratio of the heat removed from the system to the energy spent on cooling and has the following formula:

$$CoP = \frac{P_c}{P_{AC}} \tag{7}$$

where P_c is the total computing power (sum of the values in **P** vector) and P_{AC} is the cooling power. CoP increases with higher CRAC suppy temperature (T_{sup}). In this work, we use the CRAC unit CoP model given by [3] as follows:

$$CoP(T_{sup}) = 0.0068\,{T_{sup}}^2 + 0.0008\,T_{sup} + 0.458 \tag{8}$$

where T_{sup} is in Celcius. The upper limit on how much we can increase supply temperature (T_{sup}) 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 is found by adding this difference to T_{sup} and cooling cost is calculated as follows:

$$T_{sup}\prime = T_{sup} + T_{red} - T_{in,max} \tag{9}$$

$$P_{AC} = \frac{P_c}{CoP(T_{sup}\prime)} \tag{10}$$

C. Temperature Model

Node inlet temperatures are sufficient for computing the cooling cost; however, we need to calculate the junction temperature to estimate processor reliability. We calculate the junction temperature in two steps using a linear model. Firstly, we calculate the heat sink temperature as follows:

$$T_{HS} = T_{HS,ref} + (T_{in} - T_{in,ref})x \tag{11}$$

where T_{HS} is the heat sink temperature, $T_{HS,ref}$ is the reference heat sink temperature, T_{in} is the node inlet temperature and $T_{in,ref}$ is the reference inlet temperature. T_{HS} corresponds to the typical heat sink temperature at reference inlet temperature, $T_{in,ref}$. For example, we take $T_{in,ref}$ as the supply temperature $T_{sup} = 15°C$ and $T_{HS,ref} = 45°C$, meaning that when the inlet temperature is 15°C, we observe 45°C on the heat sink. x is a scaling factor determining the effect of T_{in} deviation from $T_{in,ref}$ on the heat sink temperature. For example, $x = 0$ corresponds to the case for which, heat sink temperature stays constant with changing inlet temperature. We take x as 0.6 as suggested in prior work [26]. Secondly, we calculate the server junction temperature T_j as follows:

$$T_j = T_a + P \times R_{ja} \tag{12}$$

where T_a is the ambient temperature and is equal to the heat sink temperature, P is the processor power and R_{ja} is the junction to ambient thermal resistance and is typically 0.1 °C/W for a high quality heat sink. Our previous server power value of 350 W included the total power for 2 processors, memory, interconnects etc. To calculate the junction temperature of a single processor, we use 120 W for processor active power.

D. Data Center Reliability Model

Hardware reliability is an important issue in data centers. High component temperatures lead to higher failure rates, which in turn impact both maintainability of the data center and the job performance. Without precautionary mechanisms such as failure prediction or redundant failover nodes, node failures may result in job termination, which impacts the performance of the HPC applications. Thus, our job allocation policy considers the processor hardware reliability as a constraint. Our main focus in this work is the processor reliability. We assume all the processors are of homogeneous hardware, so we assume the default failure rate is the same across processors.

We consider three main temperature-induced failure mechanisms, *Electromigration (EM)*, *Time Dependent Dielectric Breakdown (TDDB)*, and *Negative Bias Temperature Instability (NBTI)* [27], [28], [29]. They all have the general exponential failure rate formula given as follows:

$$\lambda = \lambda^0 \times e^{\frac{-E_a}{kT}} \tag{13}$$

where E_a is the activation energy for the failure mechanism, k is the Boltzmann's constant ($8.62 \cdot 10^5$), T is the temperature, and λ^0 is a material-dependent constant. $E_{a_{EM}} = 0.7eV$ for Al alloys [27]. We set $E_{a_{TDDB}} = 0.75eV$ [27]. NBTI activation energy represents $E_{a_{NBTI}} \times 1/n$, where n is the measured time exponent. We use $E_{a_{NBTI}} = 0.15eV$ and $n = 0.25$, giving the product $0.6eV$ [27], [28].

Using the sum of failure rates model, failure rate for a component is the sum of failure rates for each failure mechanism [29]. The resulting component mean-time-to-failure (MTTF) and reliability are calculated as follows:

$$\lambda_c = \lambda_{EM} + \lambda_{TDDB} + \lambda_{NBTI} \tag{14}$$

$$R(t) = e^{-\lambda_c t} \tag{15}$$

$$MTTF_c = 1/\lambda_c \tag{16}$$

where λ_c, $MTTF_c$ and R(t) are component failure rate, MTTF, and reliability at time t, respectively. We set the failure rate constants such that per-processor MTTF is 5 years at 60°C.

IV. Optimization Methodology

In this section, we first formulate and solve the cooling energy optimization and communication cost optimization problems individually. For cooling cost minimization, we use the Minimize Peak Inlet Temperature (MPIT) algorithm [4]; for communication cost minimization, we deploy the MC1X1 algorithm [10]. We then propose a job allocation algorithm, which takes both cooling efficiency and communication latency into consideration. We also discuss how reliability constraints can be included in the job allocation optimization.

A. Performance-aware Job Allocation

The objective of performance-aware (i.e., communication cost-aware) job allocation is to assign a job to a set of available nodes on a target system such that the average number of communication hops between the nodes is minimized. The target system in this paper is a mesh-connected HPC cluster, as discussed in Section III. We formulate the performance-aware job allocation problem in Equation (17).

$$\begin{aligned} \underset{X_{job}}{\text{minimize}} \quad & CC_{job}(X_{job}) \\ \text{subject to} \quad & \sum_{i=1}^{N} x_i = n \qquad x_i \in \{0,1\} \end{aligned} \tag{17}$$

where N=40 is the number of total nodes within the data center and n is the total number of nodes required by a job. X_{job} is a vector described as $X_{job} = \{x_1, x_2, ..., x_N\}$, where x_i $(i = 1, \ldots, N)$ represents whether node i is assigned the *current job* or not. It shows the selected nodes to allocate the *current job*, so n of its elements are 1 and the rest is 0. CC_{job} represents the communication cost of a job running on the target system as introduced in Section III-A. Based on Equation (1), $CC_{job}(X_{job})$ can be formulated as:

$$CC_{job}(X_{job}) = \frac{1}{n} \sum_{(x_i, x_j) \in X_{job}} (x_i \cdot x_j)[w_x(x_i, x_j) + w_y(x_i, x_j)] \tag{18}$$

where n is the number of nodes a job requires and (x_i, x_j) $(i, j = 1, \ldots, m)$ stands for a pair of source and destination nodes that a message is passing through. We use the MC1X1 algorithm [10] to minimize the communication cost, as it aims at minimizing the pairwise L1 distance across the communication nodes and provides acceptable results for *all-to-all* communication pattern [6].

MC1X1 allocation algorithm tries to confine the allocated jobs into the smallest possible area. A rectangular-shaped area, in which all the assigned nodes are ideally confined, is called a *shell*. The node located at the center of the shell is called the shell center. For an incoming job, MC1X1 traverses the data center layout and finds shells of different centers and sizes among the available (idle) nodes. During this traversal, MC1X1 records a score for each node, where the score is the size of the smallest possible shell centered at that node. The decision of which node to select as a shell center depends on its score. A lower score indicates a smaller shell area, leading to a more compact allocation with lower communication cost.

B. Cooling-aware Job Allocation Policy

Maximum cooling energy saving is achieved when the maximum inlet temperature $\{T_{in}\}$ in the data center is minimized [4]. Therefore, a cooling-aware allocation policy assigns jobs to nodes so that the resulting max$\{T_{in}\}$ will be minimum; this algorithm is called as the Minimize Peak Inlet Temperature (MPIT) algorithm in prior work [4]. We formulate the optimization problem of allocating a job to an idle data center with minimal cooling energy as follows:

$$\begin{aligned} \underset{X_{dcenter}}{\text{minimize}} \quad & max\{T_{in}(X_{dcenter})\} \\ \text{subject to} \quad & \sum_{i=1}^{N} x_i = n_{dcenter} \qquad x_i \in \{0,1\} \end{aligned} \tag{19}$$

where $X_{dcenter}$ is a vector described as $X_{dcenter} = \{x_1, x_2, ..., x_N\}$, where x_i $(i = 1, \ldots, N)$ represents whether node i is assigned any job or not. Vector $X_{dcenter}$ shows all of the busy nodes in the data center corresponding to *currently* and *previously* allocated jobs. $n_{dcenter}$ is the sum of the sizes of all jobs running on the data center. Rest of the parameters are defined the same as in Equation (17). T_{in} represents the inlet temperature of a system which is defined in Equation (20).

$$T_{in}(X_{dcenter}) = T_{sup} + D \cdot P_{idle} + D \cdot X_{dcenter} \cdot P_{util} \tag{20}$$

where T_{sup} is the CRAC unit supply temperature, D is heat distribution matrix. P_{idle} and P_{util} are the idle and dynamic power for the nodes. Note that, in order to allocate a second job to a busy data center, we use additional constraints to represent the currently busy nodes. For example, if nodes 1, 2 and 3 are busy at the time of allocation, we add the constraints x_1=1, x_2=1, x_3=1 to solve the problem.

As described in Section III-B, cooling cost is highly dependent on the CRAC supply temperature T_{sup}. If we can increase T_{sup} as much as possible without causing the nodes to exceed the redline temperature, we can save power. The maximum allowed T_{sup} increase, therefore, is limited by the maximum inlet temperature max$\{T_{in}\}$. We use the Matlab function *fminimax* to solve the optimization problem. The function returns a real number solution x_{real} and we use the discretization algorithm suggested in [4] to convert it to the nearest integer solution x_{int} which obeys the constraints.

```
while(job queue ~= empty)
    do
    n= jobsize (jobno)
    [possible_sc]=MPIT(n, X_dcenter, P)
    for i ∈ {all possible_sc}
        possible_X_dcenter(i)=MC1X1_revised(possible_sc(i), X_dcenter)
        temp(i)=find_max_Tin (possible_X_dcenter(i))
        possible_X_job(i)= possible_X_dcenter(i) - X_dcenter
        CC_job(i)=find_CC_job(possible_X_job(i))
    end
    sort (temp)
    for j ∈ {min(temp)}
        selected=find(possible_X_dcenter with min(CC_job(j)))
    end
    X_dcenter=selected
    update (P)
    record (Tin_max, CC_job, P_ac)
    jobno++
    end
```

Fig. 5: Joint optimization algorithm.

C. Joint Optimization Policy for Data Center Job Allocation

This section provides the details of our optimization policy, which jointly optimizes cooling energy and communication cost of applications running in an HPC data center. Cooling-aware and performance-aware policies optimize cooling power and communication latency independently, which means that the resulting allocations may not be successful when both objectives are considered simultaneously. Cooling-aware job allocation is mostly affected by the layout of the data center as the recirculation effect changes depending on the location of the active nodes. In most cases, cooling-aware policy allocates jobs to the nodes located far from each other. For example, for a job of size 4, cooling-efficient allocation distributes the job equally among the data center rows in order to minimize the peak inlet temperature. This causes very high communication latency for cooling-aware policy. On the other hand, performance-aware MC1X1 policy confines the nodes of each allocated job into the smallest possible *shell*. It follows a regular pattern to allocate the jobs in the data center and arbitrarily breaks ties. It does not care about whether an allocation results in high temperature as long as the allocated nodes are within the smallest shell possible, potentially causing inefficient cooling.

We design a heuristic algorithm combining both policies. Our algorithm first considers cooling-aware job allocation solution, and then uses the resulting nodes as candidates for shell centers to apply the performance-aware job allocation policy. Then, we break the ties of possible performance-aware job allocations by selecting the allocation with minimal peak inlet temperature.

When a job arrives, our algorithm first checks which nodes the cooling-aware policy would allocate the job to. These nodes are called as *possible shell centers*. Then, we feed the locations of these *possible shell centers* to the MC1X1 algorithm to minimize communication cost. We modify the MC1X1 algorithm to make it open a shell centered at a given input node (possible shell center) accordingly. In MC1X1, opening a shell centered at a node refers to finding the smallest square-shaped area to include all nodes of a job. Starting from the smallest shell (1 square unit), the number of available nodes in the shell are checked. If the size of the job is larger than the available nodes, shell is expanded. When the available node count is met, policy examines whether there are multiple allocation options within the shell area. For example, assume that we have a shell with 9 nodes, 3 of whom are busy, and we will assign a job of size 4 to the rest. In this case, we choose the 4 nodes with minimum communication cost possible. The resulting selection is the possible allocation corresponding to that *possible shell center*. For each *possible shell center*, revised MC1X1 algorithm gives an allocation vector, *possible_X_dcenter*. Among those vectors, we select the most cooling efficient one (i.e., resulting in smallest peak inlet temperature). For example, assume that for a job i of size 3, cooling-aware policy assigns the job to nodes 1, 4, and 5. We open shells centered at those nodes and select the one with the smallest inlet temperature. For the cases where two or more allocations result in the same inlet temperature but different communication costs, we find and choose the smallest communication cost allocation.

Figure 5 illustrates the flow of the joint optimization algorithm. MPIT and MC1X1_revised are the cooling-aware and revised performance-aware algorithms, respectively. *X_dcenter* and *P* are the vectors holding the current busy nodes information and the power values. *Possible_sc* is the possible shell center and *CC_job* stands for the job communication cost. *Possible_X_dcenter* is the vector of busy nodes that will result from the possible allocation. *Possible_X_job* vector shows which nodes will be assigned to the job. Note that the joint policy is scalable to larger data centers. The only parameters to change for a different data center are the cross-interference coefficient matrix and the power values for the nodes.

If the user or the administrator wants to add a minimum MTTF constraint to the joint policy, we check what the resulting MTTF value for each processor would be before every allocation decision. To compute these MTTF estimates, we first compute the resulting inlet temperatures for that allocation using Equation (2). Next we compute junction temperatures as described in Section III-C. Finally, we compute processor MTTF as explained in Section III-D. If the current allocation is expected to result in an MTTF value lower than the given threshold for any processor, we stall the allocation and wait for some of the existing jobs to finish.

V. Experimental Results

In this section, we present the experimental results for the three different allocation strategies: cooling-aware, performance-aware and our joint optimization technique. We first demonstrate the job allocation decision of each strategy on a single row of the data center. We then experiment with multiple-row allocation for our target data center with 40 nodes. We also compare our joint allocation policy against the cooling-aware and communication-aware policies under dynamically changing workload.

Single-row Job Allocation

In this test case, there are four jobs to be allocated sequentially, which have sizes of $4, 5, 6$ and 3 nodes, respectively. Figure 6 illustrates how each policy assigns the jobs to the

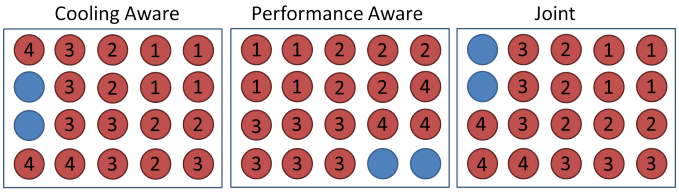


Fig. 6: Allocation scheme for the three policies.

nodes. Red and blue colors respectively represent busy and free nodes. The numbers in the circles show which jobs are running on the nodes. Cooling-aware policy assigns jobs to the nodes located at the right side of the data center and avoids the nodes that are high recirculation contributors. This result is in parallel with previous the characteristics of our data center as shown in Figure 4. Communication-aware policy, on the other hand, tries to confine the allocated nodes to the smallest area possible. Therefore, the resulting allocation for each job is more compact. Our joint allocation policy finds the cooling-efficient areas and assigns the jobs to the nodes as close to each other as possible without causing notable temperature increases. Joint policy does not always result in the same minimum inlet temperature as the cooling-aware policy, but follows closely.

Table I shows the percentage of active nodes, maximum inlet temperatures, individual job communication cost (CC), and cooling power (P_{ac}) for all the three allocation schemes. As we can see in Table I, performance-aware policy gives the lowest job communication cost (CC) for each job; however, it reaches the high inlet temperatures very fast. Cooling-aware policy keeps the temperatures low, but results in very high communication latency for all the jobs. As expected, our joint policy's performance is in between the two policies.

Multiple-row Job Allocation

In order to evaluate the job allocations across the multiple rows of the data center, we use a job sequence that is similar to the sequence in the previous experiment. Figure 7 shows the percentage of the active nodes and the size of each job in terms of number of nodes required. Figure 8 shows the cooling power over time for the three allocation policies. Joint policy follows the cooling-aware policy closely and all policies converge at the 100% utilization point. However, performance-aware allocation reaches high cooling power values much faster than the joint policy.

We present the communication cost of each job in Figure 9 and observe that cooling-aware assignment results in high communication cost. The reason is that the cooling-aware assignment distributes the jobs across different rows to minimize inlet

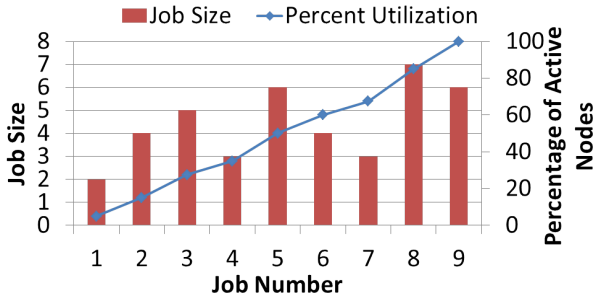


Fig. 7: Job sizes and percentage of active nodes for multiple-row allocation.

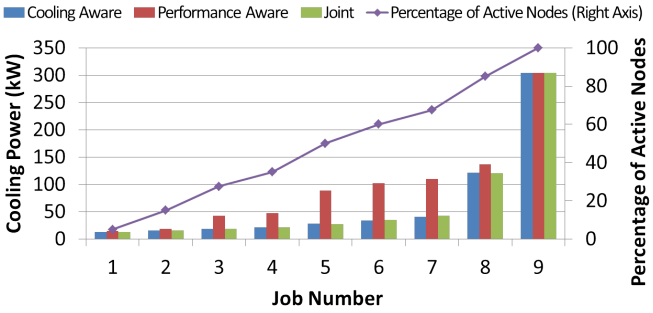


Fig. 8: Cooling power for multiple-row allocation.

temperature. As a result, communication cost is significantly affected by the distance between the communicating nodes. Joint policy resolves this issue by sacrificing some cooling efficiency. It assigns the job within a row in the most cooling-efficient way possible, and alternates the rows as more jobs arrive. However, if the number of available nodes in a row is not sufficient to service an incoming job, joint allocation also results in high communication cost. An example is seen for jobs 8 and 9 in Figure 9, where the jobs are allocated across the two rows.

We observe that our joint policy reduces the average cooling power by 30.8% compared to the performance-aware policy while increasing the power by only 0.5% compared to the cooling-aware policy. On the other hand, in comparison to the cooling-aware policy resulting in 2.45 times larger average communication cost compared to the performance-aware policy, our joint policy causes only 0.69 times larger cost. This is expected as our joint policy sacrifices some performance for improving cooling efficiency, and vice versa.

Note that our results for the single and multiple-row allocation do not consider the change in application execution time as the communication cost changes. In other words, larger communication costs may change the power-performance characteristics of jobs, hence, also affect the cooling power. Next, we investigate such interactions between performance and cooling power in detail.

Dynamic Job Allocation

In this section, we present the results for a dynamically changing workload scenario and compares our joint policy with the baseline policies. We generate a job queue with arrival time following an exponential distribution, which has been widely used in data center workload models [19]. We use an arrival rate of 15 jobs/hour. In this experiment, we update the data center status as some of the jobs finish executing.

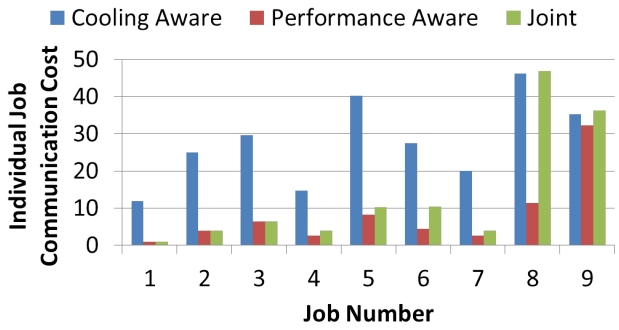


Fig. 9: Individual job communication costs for multiple-row allocation.

TABLE I: Simulation results for the single-row job allocation.

Policy		Perf-aware			Cooling-aware			Joint-opt		
Job	**Util**	**CC**	$MaxT_{in}$	$P_{ac}(kW)$	**CC**	$MaxT_{in}$	$P_{ac}(kW)$	**CC**	$MaxT_{in}$	$P_{ac}(kW)$
Job1	20%	4.0	25.0	9.4	4.0	19.9	6.3	4.0	19.9	6.3
Job2	45%	6.4	25.1	13.4	9.6	20.5	9.4	8.0	20.3	9.2
Job3	75%	8.3	32.1	35.8	13.3	23.3	15.6	14.7	23.3	15.6
Job4	90%	2.7	32.1	40.4	5.3	28.1	27.0	2.7	28.5	28.0

We adjust the power and runtime of the jobs according to the communication latency to have a realistic model. The allocation is based on a first-come-first-serve policy and when there are no available nodes, we wait for other jobs to finish. We simulate a total time of 4 hours and use the last 3 hours of the simulation in which 41 jobs arrive. We record the maximum inlet temperature at each time step and cooling cost for each job. At each time step, the current available node list, power values of active nodes and the finishing time of the jobs are updated according to the model in Section III-B. We set the communication level for all the applications, C%, as 20%.

Figure 10 shows the percentage of active nodes over time and the cooling power for all three allocation policies. An important observation is that, in the dynamic case, the active node percentage is higher for the cooling-aware policy. This is because cooling-aware allocation results in high communication latency, which means that C% part of the application is running slower and thus results in longer runtime. Therefore, not only the nodes dissipate power for longer time, but also the next job is allocated in a less efficient way due to more limited allocation freedom. On the other hand, joint optimization policy manages to overcome this problem by following a pattern similar to the MC1X1 algorithm. For example, during the time between the black dashed lines (70-90 minutes), cooling-aware case has almost 100% of its nodes active, while for performance-aware and joint allocation cases, a job finishes after 75 minutes and some nodes are freed. This performance effect translates into changes in the cooling cost, as shown in the bottom plot in Figure 10. Cooling power for our joint policy closely follows the cooling-aware policy from time 0 to 80 minutes. However, when cooling-aware policy starts losing its efficiency because of the performance overheads, joint policy starts following the performance-aware policy (see Figure 10). These results show that for a data center running HPC applications with intensive communication, even a cooling-aware policy may result in inefficient cooling if it does not take into account the communication latency. The average cooling power for the 3-hour period is 53.1 kW for the cooling-aware policy while it is 53.3 kW and 32.2 kW for performance-aware and joint policies, respectively. This corresponds to close to 40% cooling power savings in comparison to both cooling-aware and performance-aware policies. We also evaluate the energy consumption of the data center for different allocation schemes and observe 170.7 kWh, 163.3 kWh, 98.4 kWh for cooling-aware, performance-aware and joint allocation policies, respectively.

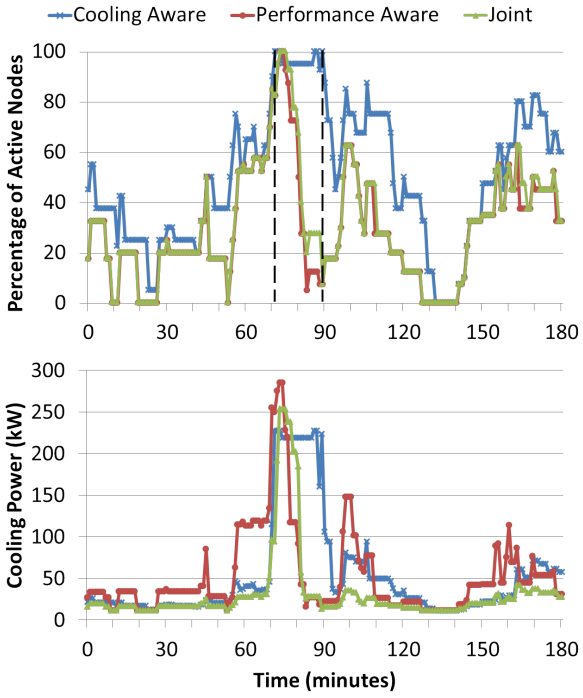


Fig. 10: Percentage of the active nodes and the cooling power traces for dynamic allocation.

Figure 11 compares the communication cost for each job allocation policy. It shows that the frequency of the occurrence of communication costs for the total number of jobs allocated. For the performance-aware policy, data points are confined to the lower communication cost area and while for cooling-aware policy it is distributed across the spectrum. For the performance-aware policy, all the jobs have communication costs lower than 30, while 97.6% of the jobs have $CC < 30$ for the joint policy.

We repeat the same experiments with a higher communication level per application of $C = 30\%$. We observe the average cooling power as 74.2 kW, 50.8 kW and 32.1 kW, while the corresponding energy consumptions are 238.96 kWh, 154.96 kWh, 98.3 kWh for cooling-aware, performance-aware and joint allocation schemes, respectively. This corresponds to 56.7% cooling power saving compared to the cooling-aware policy and 36.8% compared to the performance-aware policy.

In order to include reliability awareness during job allocation, we set a minimum MTTF constraint of 4 years and achieve an average cooling power of 20 kW without total runtime change. Even though the allocation stalls in order to meet the MTTF constraint (i.e., waits for other jobs to finish so that temperatures decrease), total runtime of the job set is not affected under the given job arrival rate. When we increase the arrival rate to 25 jobs/hour and compare the results with and without reliability constraint, we observe a 63% increase in the total runtime.

Note that our runtime job allocation policy has low overhead. We measure the time spent on running the allocation algorithm for each job for the dynamic queue of 41 jobs. We have observed that the time each job allocation decision takes is less than 1 second in our Matlab-based implementation.

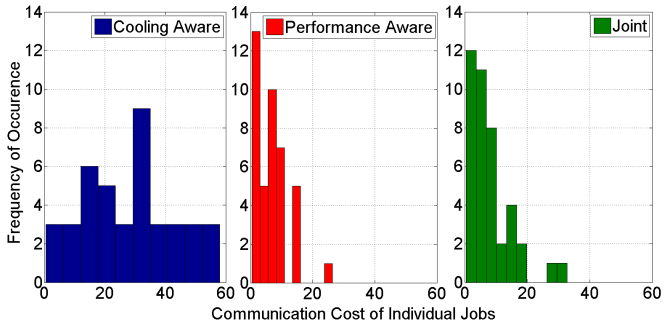


Fig. 11: Histogram of the communication cost for the dynamic allocation experiment.

VI. Conclusion

In this paper, we have proposed a joint job allocation policy to optimize both cooling power and communication latency in HPC data centers. Our policy first uses the MPIT algorithm to find the most cooling-efficient nodes to allocate a job. It then applies the modified MC1X1 algorithm to allocate the job on cooling-efficient nodes while keeping the average L1 distance at a minimum. We showed that for static allocation, our joint policy reduces the average cooling power by 30.8% compared to the performance-aware policy while it increases the power by only 0.5% compared to the cooling-aware policy. We demonstrated that for dynamically changing workloads, solely using a cooling-aware policy does not give the minimum cooling power due to the resulting high communication latency. We validated our joint policy under dynamically changing workloads and observed that, for HPC applications with 20% communication, our policy decreases the cooling power by 40% in comparison to cooling-aware and performance-aware policies. At the same time, we achieve comparable performance to that of the performance-aware policy. Cooling power savings increase for HPC applications that have a higher frequency of communication.

Acknowledgements

This work has been partially funded by NSF grant CNS-1149703 and Sandia National Laboratories.

References

[1] J. Koomey, "Growth in data center electricity use 2005 to 2010." http://www.analyticspress.com/datacenters.html, August 1 2011.

[2] R. Sawyer, "Calculating total power requirements for data centers," *White Paper, American Power Conversion*, 2004.

[3] Moore *et al.*, "Making scheduling "cool": temperature-aware workload placement in data centers," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, pp. 5–5, 2005.

[4] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, pp. 1458–1472, Nov. 2008.

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

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

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

[8] V. Subramani, R. Kettimuthu, S. Srinivasan, J. Johnston, and P. Sadayappan, "Selective buddy allocation for scheduling parallel jobs on clusters," in *IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 107–, 2002.

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

[10] M. A. Bender, D. P. Bunde, E. D. Demaine, S. P. Fekete, V. J. Leung, H. Meijer, and C. A. Phillips, "Communication-aware processor allocation for supercomputers: Finding point sets of small average distance," *Algorithmica*, vol. 50, pp. 279–298, Jan. 2008.

[11] A. Sansottera and P. Cremonesi, "Cooling-aware workload placement with performance constraints," *Performance Evaluation*, vol. 68, pp. 1232–1246, nov 2011.

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

[13] P. W. D. Bunde and V. Leung, "Faster high-quality processor allocation," in *Proceedings of the 11th LCI International Conference on High-Performance Clustered Computing*, 2010.

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

[15] L. Wang, G. von Laszewski, J. Dayal, X. He, A. Younge, and T. Furlani, "Towards thermal aware workload scheduling in a data center," in *International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN)*, pp. 116 –122, dec. 2009.

[16] T. Heath *et al.*, "Mercury and freon: temperature emulation and management for server systems," in *ASPLOS*, pp. 106–116, 2006.

[17] S. Wang and J.-J. Chen, "Thermal-aware lifetime reliability in multicore systems," in *International Symposium on Quality Electronic Design (ISQED)*, pp. 399–405, 2010.

[18] J. Meng, F. Kaplan, M. Hsieh, and A. K. Coskun, "Topology-aware reliability optimization for multiprocessor systems," in *International Conference on VLSI and System-on-Chip*, pp. 243–246, 2012.

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

[20] S. Kumar, Y. Sabharwal, R. Garg, and P. Heidelberger, "Optimization of all-to-all communication on the blue gene/l supercomputer," in *International Conference on Parallel Processing*, pp. 320 –329, 2008.

[21] "Belden incorporation, data center design guidelines." http://www.belden.com/pdfs/techbull/datacenterguide.pdf, 2007.

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

[23] P. Rad, K. Karki, and T. Webb, "High-efficiency cooling through computational fluid dynamics," *Dell Power Solutions*, February 2008.

[24] Q. Tang, T. Mukherjee, S. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters," in *International Conference on Intelligent Sensing and Information Processing, ICISIP.*, pp. 203 –208, 15 2006-dec. 18 2006.

[25] C. Lively *et al.*, "Energy and performance characteristics of different parallel implementations of scientific applications on multicore systems," *J. High Perform. Comput. Appl.*, vol. 25, Aug 2011.

[26] E. Walsh *et al.*, "From chip to cooling tower data center modeling: Part ii influence of chip temperature control philosophy," in *IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, pp. 1 –7, june 2010.

[27] JEDEC, "Failure mechanisms and models for semiconductor devices, technical report jep122c." http://www.jedec.org, March 2006.

[28] M. Alam, H. Kufluoglu, D. Varghese, and S. Mahapatra, "A comprehensive model for pmos nbti degradation: Recent progress," *Microelectronics Reliability*, vol. 47, no. 6, pp. 853 – 862, 2007.

[29] J. Srinivasan *et al.*, "Ramp: A model for reliability aware microprocessor design," Tech. Rep. IBM-RC23048(W0312-122), Dec. 2003.