Energy-Efficient, Thermal-Aware Task Scheduling for Homogeneous, High Performance Computing Data Centers: A Cyber-Physical Approach

Qinghui Tang, *Member, IEEE,* Sandeep K. S. Gupta, *Senior Member, IEEE,* and Georgios Varsamopoulos, *Member, IEEE*

***Abstract*—High Performance Computing data centers have been rapidly growing, both in number and size. Thermal management of data centers can address dominant problems associated with cooling such as the *recirculation* of hot air from the equipment *outlets* to their *inlets*, and the appearance of *hot spots*. In this paper, we show through formalization that minimizing the peak inlet temperature allows for the lowest cooling power needs. Using a low-complexity, linear heat recirculation model, we define the problem of minimizing the peak inlet temperature within a data center through task assignment (MPIT-TA), consequently leading to minimal cooling requirement. We also provide two methods to solve the formulation, XInt-GA, which uses a genetic algorithm and XInt-SQP, which uses sequential quadratic programming. Results from small-scale data center simulations show that solving the formulation leads to an inlet temperature distribution that, compared to other approaches, is 2 °C to 5 °C lower and achieves about 20%-30% cooling energy savings at common data center utilization rates. Moreover, our algorithms consistently outperform MinHR, a recirculation-reducing task placement algorithm in the literature.**

***Index Terms*—Performance Analysis and Design; Energy-aware systems; Modeling techniques; Measurement, evaluation, modeling, simulation of multiple-processor systems.**

I. Introduction

Server farms and high-performance computing clusters are housed in data centers that are limited by power and cooling capacity. Current trends in data center growth show a rapid increase in both computing and storage capacity; IBM and Intel have announced integration process plans for the next 10 years, at scales below 45 nm. Moreover, technological trends increase both the operating frequency and the power density which, for data centers, is expected to reach 60 KW/m^2 by the year 2010. Collectively, these factors yield an exponential increase in power needs of data centers that is not sustainable. According to a recent report [2], in 2006, data centers in the U.S. used 59 billion KWh of electricity, costing the US $4.1 billion and 864 million metric tons of carbon dioxide (CO_2) in emission; this accounted for 2% of the total USA energy budget, while it is projected that it will reach 3% by the year 2010. The increasing cost of power and recent legislative efforts [3] press for greener data centers.

For a large scale data center, the annual energy cost can be up to millions of dollars, and the cooling cost is at least half of the total energy cost [4]–[6]. Although recently built data centers exhibit better cooling efficiency (roughly 1:1.5 for IBM's BlueGene/L and 1:2.5 for TACC's Ranger), cooling energy consumption is still a significant portion of the total data center energy consumption. A key challenge towards optimizing the operation of a data center is to minimize the cooling requirement and, as a result, improve its overall energy efficiency. In this paper, we investigate data centers from a holistic, cyber-physical aspect that encompasses both their computing and cooling equipment; we explore the potential of reducing the operational cost of data centers by making cooling more efficient—and thus more economical—through *thermal-aware task management*. Heat recirculation plays a significant role in a data center's energy efficiency [7], [8]; as computing devices in a data center emit heat by running tasks, the cooling system must supply cold air to their air inlets at a temperature below their *redline* temperature, i.e. the maximum allowed operational temperature specified by the device manufacturer. However, the *recirculation* of hot air from the air outlets of the IT equipment back into their air inlets (see Fig. 1) increases the inlet temperatures and can cause the appearance of hot spots [7], [8]. Heat recirculation forces data center operators to operate their *computer room air conditioners* (CRACs) to supply cold air at a *much* lower temperature than the redline (although in the ideal case of no recirculation it could be equal to the redline temperature). Lowering CRAC's output temperature forces it to operate at a worse *coefficient of performance*, i.e. the ratio of the removed heat over the energy required to do so, which considerably increases the cooling cost.

The most elaborate way to describe recirculation is through a *computational fluid dynamics* (CFD) model. Similarly, data center *thermal maps*, parameterized with respect to CPU utilization, are models that, given the utilization layout, predict the temperature distribution in a data center [9]. However, CFD models and thermal maps are computationally expensive to produce. Particularly, CFD models are not suitable solutions for on-line decisions on task placement.

In efforts to characterize the recirculation with simpler terms, HP Labs and Duke University [8], [10]–[12] introduced

This work is supported in part by grants from Intel Corporation, Science Foundation Arizona and National Science Foundation (CNS#0649868).

A preliminary version of this paper appeared in IEEE Cluster 2007 [1].

Qinghui Tang is with Texas Instruments, Austin, TX. E-mail: jeffrey.tang@ti.com.

Sandeep K. S. Gupta and Georgios Varsamopoulos are with the Impact Laboratory, School of Computing and Informatics, Ira A. Fulton School of Engineering, Arizona State University, Tempe, AZ 85287-8809. E-mail: sandeep.gupta@asu.edu, georgios.varsamopoulos@asu.edu. http://impact.asu.edu/.

the the Supply Heat Index (SHI) and Recirculation Heat Index (RHI) [8] which are scalar, dimensionless metrics that are used to classify the thermal efficiency of a data center, and the Heat Recirculation Factor (HRF) metric that is used by MinHR [12], a thermal-aware workload placement algorithm. The idea behind thermal-aware management is to predict the thermal effects of a task placement, thus selecting a placement with best effects.

Although characterizations such as HRF can achieve savings in cooling cost, they fail to accurately capture the phenomenon of heat recirculation and can hardly be used as process models to predict the thermal effects of a task placement. In this paper, we modify our previous work on thermal interference in biomedical sensor networks [13], and we propose a new approach that has lower complexity (and granularity) than CFD models yet effectively characterizes the heat recirculation among all pairs of equipment.

In this paper, we assume a homogeneous data center, i.e. of identical equipment, and combine a linear, low complexity heat recirculation[1] model [14] and a linear power model [15], both from previous work, to develop an exact process model of inlet temperatures and formulate the problem of ***how to distribute an incoming task among the servers in order to maximize the supply temperature while respecting the redline temperatures and thus minimize the cooling requirement*** (the properly formulated problem is named MPIT-TA). This paper's contribution can be summarized as follows:

- power profiling and modeling of compute equipment, to verify a linear power model with respect to CPU utilization;
- a linear, low-complexity process model to predict the equipment inlet temperatures for a data center given a server utilization vector;
- mathematically formalizing the problem of minimizing the data center cooling cost as the problem of minimizing the maximal (peak) inlet temperature through task assignment (MPIT-TA);
- provision of two methods, XInt-GA and XInt-SQP, to solve the aforementioned problem.

Due to the homogeneity of the data center, the placement of a task has little effect on the computational performance of the task. Hence, the algorithm does not deal with compromising the performance to achieve better thermal effects. Actually, performance of the data center may in fact improve because this algorithm, and any recirculation-reducing algorithm, can diminish the hot spots that may otherwise cause some servers to throttle down. Irrespective of the performance issues, simulations on a small-scale data center show that the XInt algorithms can result in up to 35% in cooling cost savings.

The rest of the paper is organized as follows: Section II presents the system model and formulates the MPIT-TA problem. Section III describes the XInt algorithms. Section IV briefly describes previous similar approaches that are used for comparison. Section V describes the power profiling and methodology used on equipment from Arizona State University's (ASU) Fulton High Performance Computing Initiative (HPCI) center to validate the linear power model assumed by MPIT-TA. Section VI describes the simulation setup and results. Section VII gives a categorization of thermal management techniques. Section VIII describes extensions to MPIT-TA. Lastly, Section IX concludes the paper with comments on the results and with discussion about future work.

[1]It is important to distinguish between *reduction of air recirculation* and *reduction of heat recirculation*. The method presented in this paper does not attempt to alter the *air recirculation*, i.e. the flow pattern of recirculated air, to achieve the cooling cost benefits; in fact, it relies on a steady, predictable air flow pattern to compute a way to have a lesser amount of *heat recirculated* by that air flow into the equipment inlets.

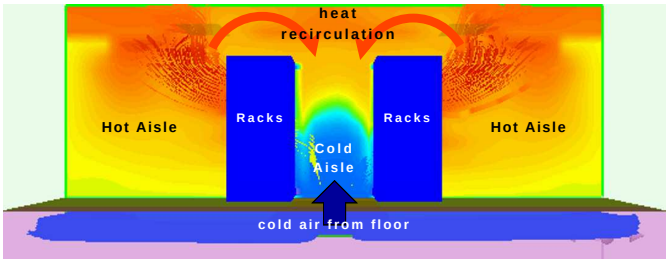


Fig. 1. Demonstration of heat recirculation: heated air in the hot aisle loops around the equipment to enter the air inlets.

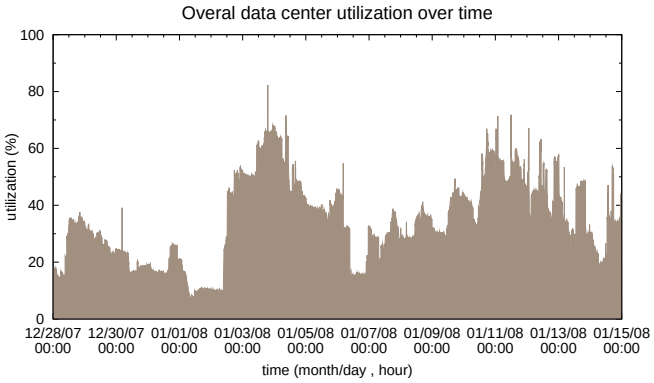


Fig. 2. Overall utilization for the ASU Fulton HPCI data center over the course of about three weeks. The data center is mostly underutilized.

II. System Model and Problem Formulation

A. Preliminaries

A typical data center is laid out with a hot-aisle/cold-aisle arrangement by installing the racks and perforated floor tiles on the raised floor. The racks are typically 42U-size racks, each fitting 42 1U systems or 6 chassis of blade systems. All servers are connected to a high-speed network (e.g. Gigabit Ethernet or InfiniBand). Typical network configuration for a system is a star topology over a central switch. The CRACs (a.k.a Heating Ventilation Air Conditioners – HVACs) deliver cold air under the elevated floor through the perforated tiles. The cold air enters the racks from their front side (i.e. the air inlets), picks up heat from the circuitry, exits the racks into the hot aisles, and gets extracted from the room by the air conditioner intakes, normally positioned above the hot aisles.

In typical High Performance Computing (HPC) scenarios, the servers perform a task for hours or even days. For example, a Spice circuit simulation task of a new VLSI design may run the simulation in parallel on hundreds of servers for several

days. Moreover, data centers tend to be under-utilized [16], [17]; Fig. 2 shows the overall utilization of the ASU Fulton HPCI data center for the course of about two and a half weeks.

In data centers, when there is a change in the distribution of power consumption, the temperature distribution reaches a new steady state in about 10 to 20 minutes [18]. Therefore, we assume the data center stays in a certain utilization rate long enough for the temperature distribution under this utilization rate to reach a steady thermal state that can be analytically expressed. This is especially true for HPC data centers where tasks take days to finish. This assumption allows us to analyze data centers in terms of both power and energy consumption.

For simplicity, we assume a *homogeneous* hardware environment. All nodes (chassis) contain the same number of server blades, which have the same power consumption and computing capability. Also, in the problem formulation to follow, we assume the basic scenario of assigning a single, multi-processor task to an idle data center. This basic scenario is the building block for extended formulations (e.g. to handle multiple tasks or operate on a partly used data center), as shown in Section VIII.

Generally, recirculation is not uniform; it demonstrates uneven patterns. Therefore, the placement of a task in the data center has a varied effect on the heat recirculation, and consequently a varied effect on the cooling cost. Using this linear thermal model, we show that the cold air supply and the heat recirculation are the two factors that determine the inlet temperatures—and thus the maximum inlet temperature. Our cooling cost savings rely on the idea that, if the temperature rise due to heat recirculation can be lowered by appropriately placing a task in a data center, there will be more room to supply "warmer" cold air, which effectively will allow the cooling system to work in a more energy-efficient mode.

The model that connects the CPU utilization (i.e. the task placement) to the inlet temperatures and consequently to the cooling cost is developed by the following steps:

Section II-B: we provide a relation of the supplied cold air temperature to the power consumed in the data center by showing the effects of the *coefficient of performance* to the energy needs of the data center;

Section II-C: we express the inlet temperatures in terms of power consumed, using an abstract heat recirculation model from our previous work [14].

Section II-D: we express the inlet temperatures in terms of the task power profile and placement;

Section II-E: lastly, we express the maximum allowed supplied cold air temperature in terms of the maximum inlet temperature, which is in turn expressed as a function of the task placement.

B. Importance of coefficient of performance to the total consumed power

The total power of a data center is composed of the total computing power—from both computing and networking devices—and the total cooling power. Incidental energy consumption such as that for lighting are considered to be negligible. The total *computing power*, referred to as **computing cost**, P_{C}, is presented as

$$P_{\mathrm{C}} = \sum_{i=1}^{n} P_i, \tag{1}$$

where P_i is the power expenditure at chassis i, in Watts. Similarly, the **cooling cost** is defined as [12]

$$P_{\mathrm{AC}} = \frac{P_{\mathrm{C}}}{\mathrm{CoP}\,(T_{sup})}, \tag{2}$$

where CoP is the **coefficient of performance** of the CRAC set to supply cold air at T_{sup} temperature, and it characterizes the efficiency of a CRAC, i.e. it is defined as the ratio of the amount of heat removed by the cooling device to the energy consumed by the cooling device performing the removal. For example, a ratio of two indicates that to remove heat at the rate of 1000 W, the rate of work performed by the cooling device is 500 W.

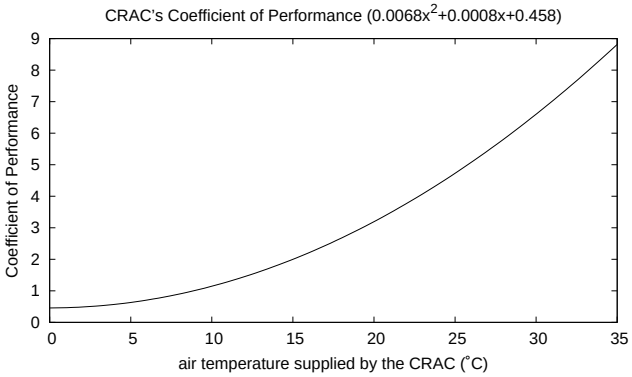


Fig. 3. Coefficient of performance curve for the chilled-water CRAC units at the HP Labs Utility Data Center. (source: [12])

We use the CoP model (shown in Fig. 3) for a water-chilled CRAC unit in a HP utility data center [12]:

$$\mathrm{CoP}\,(T) = (0.0068T^2 + 0.0008T + 0.458),$$

where T is the temperature of the supplied cold air. Note that the change of CoP is not linear and normally increases with the supplied air temperature. We can observe that operating the cooling system at a higher temperature is saving energy. Intuitively, to provide colder air, the CRAC has to work harder and consume more energy to remove more heat from the heated air. Therefore, we can minimize P_{AC} by maximizing the supplied cold air temperature T_{sup} while satisfying the constraints of redline threshold. The total power needed to operate a data center is expressed as

$$P_{\mathrm{Total}} = P_{\mathrm{AC}} + P_{\mathrm{C}} = \left(1 + \frac{1}{\mathrm{CoP}(T_{sup})}\right) \sum_{i=1}^{n} P_i,$$

C. Consumed computation power affects inlet temperatures

A data center is abstracted to consist of n **nodes** (chassis). Each node i consists of m **servers** (blades). Each node i draws air with inlet temperature T_{in}^i, and dissipates hotter air with average outlet temperature T_{out}^i. The outlet temperature of a node comes from the combined activity of the servers in that node, while the inlet temperature comes from the combination of cold air supplied from the CRAC and hot air recirculated

TABLE I
SYMBOLS AND DEFINITIONS.

Symbol	Definition
P_{Total}	Total power consumption for the data center
P_{AC}	Power needed for cooling
P_{C}	Power needed for computing
$\text{CoP}(T)$	coefficient of performance at supplied air temperature T
n	The number of computing nodes
m	the number of servers (blades) in node i
q	the number of incoming tasks
C_{tot}	the number of servers (blades) the task requires
a	the power consumption of a server at node i running task k
b	idle power consumption of node i's power unit
c_p	Specific heat of air (typical value: 1005 J Kg^{-1} K^{-1})
ρ	Density of air (typical value: 1.19 Kg/m^3)
c_i	The number of tasks assigned to server i
P_i	Power consumption of node i
T_{sup}	air temperature as supplied from the cooling unit
T_{red}	manufacturer's redline temperature (typical value 25 ℃)
T^i_{in}	Inlet air temperature of node i
T^i_{out}	Outlet air temperature of node i
f_i	Flow rate of node i (typical value 520 CFM = 0.2454 m^3/s)
Q_i	Heat rate at node i (in watts)
K_i	the thermodynamic constant, short for $\rho f_i c_p$
$\mathbf{A}$	The heat cross-interference coefficient matrix
$\mathbf{K}$	thermodynamic constant matrix $\mathbf{K} = \text{diag}(K_i), i = 0 \ldots n$
$\mathbf{D}$	distribution matrix, concise for $[(\mathbf{K} - \mathbf{A}^\mathsf{T}\mathbf{K})^{-1} - \mathbf{K}^{-1}]$
$\mathbf{b}$	the vector $\{b\}_n$ of idle consumption b
$\mathbf{c}$	the vector $\{c_i\}_n$ of task assignment/placement
$\mathbf{p}$	the vector $\{P_i\}_n$ of per-chassis power values P_i
$\mathbf{t}_{in}$	the vector $\{T^i_{in}\}_n$
$\mathbf{t}_{out}$	the vector $\{T^i_{out}\}_n$
b_{l}	Idle power consumption of a blade
$b_{\circ}$	power consumption of an empty chassis (empty of blades)
$P_{\text{s}}(u)$	power consumption of a single blade at u utilization
$P_{\text{c}}(m,u)$	power consumption of a chassis with m servers at u utilization each
redline temperature	the manufacturer-specified maximum air temperature permitted to enter the air inlet of an equipment (see T_{red})
heat recirculation	the phenomenon of heated air from an equipment's air outlets entering the inlets
power profile	an analytical power consumption model derived from experimental measurements

from the node outlets. Contemporary data centers are cooled by using conventional air-cooled technology.

According to the *law of energy conservation*, the **heat rate**, i.e. the amount of heat or energy carried by an air flow per unit time is

$$Q = \rho f c_p T,$$

where ρ, f, c_p, T are thermo-physical values: ρ is the air density (g/m^3), f is the air flow rate (m^3/s), c_p is the *specific heat*[2] of the air (Jg^{-1}K^{-1}), and T is the air temperature (K). Due to the unique location of each node in the data center, we assume that the air flow rate differs for each node. We will denote as f_i the air flow of the node i.

Considering that the power drawn by a computing device is dissipated as heat, the steady-state relationship between power consumption of a node and the inlet/outlet temperature can be written as:

$$P_i = \rho f_i c_p \left(T^i_{out} - T^i_{in}\right),$$
$$T^i_{out} = T^i_{in} + K_i P_i, \text{ where } K_i = \rho f_i c_p.$$

In other words, the power consumption of node i will cause air passing through the node i to experience an energy increase of P_i, and a temperature rise from T^i_{in} to T^i_{out}.

According to the abstract heat model of the data center, as described in previous work [14], the recirculation of heat can be described by a cross-interference coefficient matrix $\mathbf{A}_{n\times n} = \{\alpha_{ij}\}$, which denotes how much of its outlet heat each node contributes to the inlet of every other node. That is, the matrix element α_{ij} denotes that α_{ij} heat rate output from node i recirculates into node j:

$$Q^j_{in} = \sum_{i=1}^{n} a_{ij} Q^i_{out} + Q_{sup} + P_j, \text{ for } j = 1 \ldots n, \text{ or}$$
$$\mathbf{q}_{in} = \mathbf{A}^\mathsf{T}\mathbf{q}_{out} + \mathbf{q}_{sup} + \mathbf{p} \text{ (in vector format).}$$

Note that $\sum_{i=0}^{n} \alpha_{ij} \leq 1$, but $\sum_{j=0}^{n} \alpha_{ij} \lneqq 1$.

We can shift from the power space to the temperature space by applying the thermodynamic constants to the equation above. Let the thermodynamic constants K_i be organized into a diagonal matrix $\mathbf{K}_{n\times n} = \text{diag}(K_1, K_2, \ldots K_n)$. The vector of inlet temperatures $\mathbf{t}_{in}$ can now be expressed as [14]

$$\mathbf{t}_{in} = \mathbf{t}_{sup} + \left[(\mathbf{K} - \mathbf{A}^\mathsf{T}\mathbf{K})^{-1} - \mathbf{K}^{-1}\right]\mathbf{p}.$$

For brevity, we define $\mathbf{D} \equiv \left[(\mathbf{K} - \mathbf{A}^\mathsf{T}\mathbf{K})^{-1} - \mathbf{K}^{-1}\right]$, and refer to it as the *heat distribution matrix*. The difference between matrices $\mathbf{A}$ and $\mathbf{D}$ is that the latter converts power supplied by $\mathbf{p}$ into the temperature domain. The equation is then formulated as

$$\mathbf{t}_{in} = \mathbf{t}_{sup} + \mathbf{D}\mathbf{p}, \quad \mathbf{D} = \left[(\mathbf{K} - \mathbf{A}^\mathsf{T}\mathbf{K})^{-1} - \mathbf{K}^{-1}\right], \tag{3}$$

which means that each inlet temperature rises above the supply temperature due to heat from recirculation. We can see that *the row in the product* $\mathbf{Dp}$ *with maximum value determines the row in* $\mathbf{t}_{in}$ *with the maximum value*.

The next subsection links the power dissipation to the served tasks by introducing a **power profile** that maps a task to its power needs with respect to the running node.

D. Task placement affects the inlet temperatures

The data center is given a task of "size" C_{tot} to run. For simplicity, we assume that the size of the task means the number of processors required. A task of size 20 means the task requires 20 servers. For multiprocessor, multi-core systems, we can easily divide the task size by the number of cores per server to yield the task's requirement in servers. Each node contains m servers. A scheduler dispatches the task to n nodes, each node will run a "sub-task" of size c_i. Of course the scheduling results should satisfy the constraints

$$\sum\nolimits_{i=1}^{n} c_i - C_{\text{tot}} = 0, \text{ and } c_i \leqslant m.$$

The organization of our data center abstraction conforms with the modern organization of data centers into chassis and blades. A blade does not have its own power unit; it relies

[2]Energy that is stored in one unit of mass per each degree of temperature.

on the power unit of the chassis it is in. If the node expends b power when idle, and each blade expends a power when running a specific task, the power consumption of a chassis with m blades is modeled as

$$P = b + m\,a.$$

This is a *linear* power model with respect to CPU utilization. Such power models are quite accurate given their simplicity [14], [19], [20]. Section V explains in detail how to experimentally obtain the parameters a and b.

In HPC data centers, a task usually runs in parallel on many processors. When a task runs on c_i servers on a node i, the power needs of that task is $c_i a$. Thus, when a node i runs a task on c_i blades, its power consumption is

$$P_i = b + c_i a. \tag{4}$$

We construct a vector $\mathbf{c}$ of the values c_i in the data center; the power vector $\mathbf{p}$ is then expressed as

$$\mathbf{p} = \mathbf{b} + \mathbf{c}a, \quad \text{where } \mathbf{b} \equiv \begin{bmatrix} b & b & \dots & b \end{bmatrix}^{\mathsf{T}}. \tag{5}$$

Applying Eq. 5 to Eq. 3, we get:

$$\mathbf{t}_{in}(\mathbf{c}) = \mathbf{t}_{sup} + \mathbf{D}\,\mathbf{b} + \mathbf{D}\,\mathbf{c}\,a. \tag{6}$$

In the above equation, vector $\mathbf{c}$ is the target parameter to alter. Considering the analysis in Section II-C, it is now evident that *the inlet temperature vector depends on the placement of jobs within a data center*. The next subsection provides the dependency of the supplied cold air temperature on the maximum inlet temperature.

E. Inlet temperatures versus supplied cold air temperature

Suppose that, from Eq. 6 the difference between the peak inlet temperature and the red-line temperature T_{red} is

$$\Delta = T_{red} - \max_i\{T^i_{in}\}.$$

As mentioned earlier, CRACs supply cold air at a temperature well below the redline value. We can adjust the supplied cold air temperature to a higher value T'_{sup}:

$$T'_{sup} = T_{sup} + \Delta = T_{sup} - \max_i\{T^i_{in}\} + T_{red}, \tag{7}$$

which is the point at which one of the inlet temperatures reaches the redline. Thus, maximizing the supplied cold air temperature T'_{sup} is equal to the problem of minimizing $\max_i\{T^i_{in}\}$ given T_{sup}.

Also, assuming no heat transfer from outside sources, the only reason that a server may have a higher inlet temperature than that of others is because it suffers from more heat recirculation. Supplying cold air at T'_{sup} instead of T_{sup} results in the following cooling cost savings:

$$P_{\text{AC-savings}} = \frac{P_{\text{c}}}{\text{CoP}\,(T_{sup})} - \frac{P_{\text{c}}}{\text{CoP}\,\left(T'_{sup}\right)}.$$

Based on Eq. 6, the problem of assigning a single task in an idle data center can be defined as:

Minimize the Peak Inlet Temperature through Task Assignment (MPIT-TA) Problem: Given

- a data center of n chassis, each chassis having m servers with power characteristics a, b;
- a task demanding C_{tot} servers[3];
- the heat distribution matrix $\mathbf{D}$;

find a task placement vector $\mathbf{c}$ to:

$$\begin{aligned} &\text{minimize } \max_i\{T^i_{in}\} \\ \text{such that: } &C_{\text{tot}} - \sum_{j=1}^{n} c_j = 0, \\ &\mathbf{t}_{in} = \mathbf{t}_{sup} + \mathbf{D}\mathbf{b} + \mathbf{D}\mathbf{c}a, \\ &m \geqslant c_j \geqslant 0, j = 1 \dots n. \end{aligned} \tag{8}$$

That is, MPIT-TA partitions the task C_{tot} into a subtask vector $\mathbf{c}=\{c_1, c_2, \dots c_n\}$ that minimizes the peak inlet temperature.

F. Section Summary

In this section, we combined a linear power model and a linear recirculation model to formulate the MPIT-TA problem. We also related the maximum air inlet temperature ($\max_i\{T^i_{in}\}$) to the supplied cold air temperature (T_{sup}) and showed that minimizing the peak inlet temperature allows for minimization of cooling cost. In the following section, we present two methods to solve MPIT-TA.

III. Solving the MPIT-TA formulation

In the previous section, we formulated the MPIT-TA problem as a minimization problem of linear equations and constraints. In this section we provide two approaches for solving the problem: (a) a Genetic Algorithm (XInt-GA), and (b) Sequential Quadratic Programming (XInt-SQP). The GA approach is provided for two reasons: (i) the nature of the problem and solutions fits the idea of genes, i.e. the task placement vector $\mathbf{c}$ can be easily viewed as a gene sequence (i.e. a genome), and (ii) there are many software packages that efficiently implement genetic algorithms. The minimax SQP approach is provided as an independent approach to GA, thus the two approaches can be used to validate each other.

Both approaches were implemented in MATLAB. The XInt-GA algorithm was implemented using the Genetic Algorithm and Direct Search Toolbox, while the XInt-SQP algorithm was implemented using the `fminimax` function of the Optimization Toolbox, internally using SQP with quasi-Newton method.

A. The XInt-GA solution

We describe how to use a *genetic algorithm* (GA) optimization approach [21] to find a near-optimal scheduling result. In short, a genetic algorithm is an iterative approach which is given a pool of *genomes* (i.e. feasible solutions), and, by mixing solutions (i.e. *mating*) and inserting random alterations in them (i.e. *mutating*), creates new solutions and discards inefficient ones based on a *fitness function* (i.e. a metric), effectively exploring the solution space to reach a near-optimal solution. Thanks to its mating and mutation phases, a GA can escape from local optima.

[3] In multi-processor, multi-core systems, if a task specifies the number of cores it needs, we can simply divide the number of cores requested by the number of cores per server

Algorithm 1 XInt-GA: Minimizing the maximum inlet temperature using a GA approach

```
1: function BFS(C_tot, n, m)                          ▷ BFS: Basic Feasible Solution
2:     for i ← 1 to n do
3:         BFS[i] ← min{C_tot, m}
4:         C_tot ← C_tot − BFS[i]
5:     end for
6:     return BFS
7: end function
8:
9: procedure XINT-GA(C_tot, n, m, T_sup, D, a, b)
10:     CurGen ← a pool of BFS(C_tot, n, m) solutions
11:     for i ← 1 to MaxGen do
12:         SelSubs ← a select a subset from CurGen using roulette wheel
13:         MuSubs ← mutation of solutions in SelSubs
14:         MaSubs ← mating of solutions in SelSubs
15:         Apply the fitness function F() (Eq. 9) on CurGen, MuSubs, MaSubs
16:         CurGen ← all fit solutions, i.e. ones with low peak inlet temperature
17:     end for
18:     FinalSolution ← the solution within CurGen with best fitness
19: end procedure
```

A feasible solution is a solution $\mathbf{c}$ for which

$$0 \leqslant c_i \leqslant m, \forall\ i = 1 \ldots n,$$

that is, every per-chassis server assignment must not exceed the available servers. Of course, for a feasible solution to exist, the number of available servers must be more than or equal to the requested number of servers, that is,

$$mn \geqslant C_{\text{tot}}.$$

To construct a feasible solution, we need to fill up the assignment vector $\mathbf{c}$ with values m until we add up to C_{tot}. Function BFS in Algorithm 1 implements this approach.

To apply a GA approach to find a near-optimal solution, we need to define the gene and the fitness function: (a) a gene is any element in the assignment vector $\mathbf{c}$; (b) the fitness function is the resulting peak inlet temperature of that assignment, i.e. we define the fitness function F of the solution as

$$F(\mathbf{c}) = \max_i \left\{ T_{in}^i(\mathbf{c}) \right\} \quad \text{(from Eq. 6)}. \tag{9}$$

The genetic algorithm starts with a pool of feasible solutions, a.k.a. the *individuals*. The initial individuals are not required to be good, i.e. fit, solutions. Specifically, we use the BFS function to compute the basic feasible solution and assign the same value to all individuals:

$$\{\text{BFS}(C_{\text{tot}}, n, m), \text{BFS}(C_{\text{tot}}, n, m), \ldots, \text{BFS}(C_{\text{tot}}, n, m)\}$$

The algorithm repeats the following two steps:

- a step of mating individuals together, i.e. combining genes, to produce new individuals, i.e. new solutions, and
- a step of mutating individuals, i.e. randomizing the genes within an individual.

Three parameters that a genetic algorithm takes are the percentages of: a) individuals that are selected for mating, b) the population that is replaced by the new generation, and c) individuals that undergo mutation.

Also, the exact mating process is generally a parameter to the genetic algorithm. Ideally, offspring solutions should exhibit some genes from one parent and the rest of the genes from the other. However, this may generate invalid solutions as the genes may not add up to exactly C_{tot}. For this reason, we use a customized mating process, during which we randomly select several pairs of solutions, exchange a subset of two task assignments and obtain two new solutions. For example, for the two individuals $\mathbf{c}_1 = [5\ 0\ 4\ 0]$ and $\mathbf{c}_2 = [0\ 5\ 2\ 2]$, a possible mating would be to swap the first two elements and produce the offspring individuals $\mathbf{c}_3 = [0\ 5\ 4\ 0]$ and $\mathbf{c}_4 = [5\ 0\ 2\ 2]$.

To meet the percentage of replaced population the best fit offspring solutions are replicated to match the population needs. To find the solutions that are to be replaced by the new generation, we apply a probability-based *roulette wheel* selection process [21].

Algorithm 2 XInt-SQP: Minimizing the maximum inlet temperature using an SQP approach

```
1: function DISCRETIZETOCLOSESTINTEGERS(s,m)
2:     z ← floor(s);
3:     pos ← POSITIONOFFIRSTHIGHESTVALUE(s)
4:     while (‖z‖_1 < ‖s‖_1) ∧ (pos≠error) do
5:         if z(pos)+1 ⩽ m then
6:             z(pos) ← z(pos)+1;
7:         end if
8:         pos ← NEXTHIGHESTVALUEPOSITION(s,pos)
9:     end while
10:    return z
11: end function
12:
13: procedure XINT-SQP(C_tot, n, m, T_sup, D, a, b)
14:     RealSolution ← fminimax(C_tot, n, m, T_sup, D, a, b)
15:     FinalSolution ← DISCRETIZETOCLOSESTINTEGERS(RealSolution,m)
16: end procedure
```

B. The XInt-SQP method

Sequential Quadratic Programming is one of the most popular methods used to solve minimax formulations in the real number domain; there are numerous software packages that implement SQP platforms. Since this problem is defined on the integer domain, we use a two-step approach on solving it in the real domain using the `fminimax` function of MATLAB's Optimization Toolbox, and finding a close integer (and feasible) solution that respects the constraints. The approach followed is described in Algorithm 2: the problem is formatted into the parameters of `fminimax` and the function is called; the solution returned is then passed to the DISCRETIZETOCLOSESTINTEGERS, which finds an integer vector that is close to the real solution of `fminimax` but respects the row sums not to exceed the value m.

IV. OTHER APPROACHES

This section describes the algorithms used in the simulation section for comparison with the XInt algorithms. The algorithms used are divided into two groups: (a) the non thermal-aware algorithms and (b) the thermal-aware algorithm MinHR. We use these algorithms as reference to show the improvement achieved by the XInt algorithms.

A. Non thermal-aware algorithms

Most data centers do not run temperature-aware, power-aware or thermal-aware placement algorithms; instead, the

scheduler's placement policy is hard-coded with respect to the numbering of the chassis. Although, ideally, the XInt algorithms should be compared to a completely thermally oblivious placement, they are so arbitrary that there is no typical scheme that can be used as reference. Therefore, basic non hard-coded algorithms are used, such as the ones used by Moore et al. [12] to compare with MinHR. These placement algorithms are based on observation and intuition instead of taking into account the heat recirculation phenomenon. Moreover, although two of them, i.e. UOP and MCE, use temperature readings as input, they do not use some metric of their effect on the recirculation, that is why we do not regard them as thermal-aware. Their definition, as presented in here, first appeared in [15].

1) Uniform Outlet Profile (UOP): This scheme is similar to the ONEPASSANALOG algorithm [12]. Based on the inlet temperature of each computing node, the algorithm will assign more tasks to nodes with low inlet temperatures, and fewer tasks to nodes with high inlet temperatures. The objective is to achieve a uniform outlet temperature distribution.

2) Minimal Computing Energy (MCE): MCE minimizes the number of powered-on chassis and servers to concentrate computing power costs on those active servers and processors and turns off all other idle blades. For the homogeneous data center used in our study, the computing nodes with the lowest inlet temperature will be assigned tasks first, as per the COOLESTINLETS algorithm [12].

3) Uniform Task (UT): With this scheme, all nodes are assigned the same amount of tasks: $c_i = C_{\text{tot}}/n, \forall i$. This is similar to the UNIFORMWORKLOAD algorithm [12].

B. Minimize Heat Recirculation (MinHR) algorithm

The work in this paper was partially motivated by MinHR [12], a recirculation-reducing approach. MinHR is based on calculating the *Heat Recirculation Factor* (HRF) for each chassis[4], and assigning tasks according to the ratio of each chassis's HRF to the sum of all HRFs. In other words, MinHR assigns fewer tasks to chassis that cause higher recirculation, and has the same underlying principle as the MPIT-TA formulation to achieve cooling cost savings.

1) MinHR problem description: For its computations, MinHR requires the knowledge of a series of reference heat recirculation parameters, called Heat Recirculation Factors, each one describing a chassis' "contribution" to the heat recirculation. The HRFs are computed as follows: given a reference application that generates a given heat load Q_{ref}, which is the summed power consumption (heat dissipation) of all nodes at the reference state, the recirculated heat within this reference scenario is defined [12] as

$$Q_{ref} = \sum_{i=1}^{n} \rho f_i c_p (T^i_{out} - T^i_{in}), \text{ and}$$
$$\delta Q_{ref} = \sum_{i=1}^{n} \rho f_i c_p (T^i_{in} - T_{sup}),$$

while the HRF for a chassis j is defined [12] as

$$HRF_j = \frac{\text{the change in total heat dissipation}}{\text{the change in total heat recirculation}} = \frac{Q_j - Q_{ref}}{\delta Q_j - \delta Q_{ref}},$$

[4]the original term used in [12] is *pod*

Algorithm 3 Modified MinHR algorithm based on HRF

1: run XInt-GA and sum the computing power into P_c
2: run the original MinHR with P_c as input, and
3: calculate the power assignment according to Eq. 10.
4: **while** $\exists i$ such that $(\text{PwrAsgn}_i < \text{IdlePower}_i)$ or $(\text{PwrAsgn}_i > \text{FullPower}_i)$ **do**
5: $\quad \text{PwrVctr}_i = \frac{HRF_i}{\sum_{j=1}^{n} HRF_j} P_{\text{C}}$
6: $\quad \text{PwrAsgn}_i = \max\{\text{IdlePower}_i, \min\{\text{PwrVctr}_i, \text{FullPower}_i\}\}$
7: $\quad$ remove the chassis i from pool of available chassis.
8: $\quad P_{\text{C}} = P_{\text{C}} - \text{PwrAsgn}_i$
9: $\quad$ re-calculate the power assignment with the new P_{C} and chassis pool.
10: **end while**
11: Convert the PwrAsgn vector to a task placement vector $\mathbf{c}$ using the formula $c_i = (\text{PwrAsgn}_i - b)/a$.

where Q_j and δQ_j are the total amount of heat and the recirculated heat generated by the j^{th} chassis, respectively. The HRFs can be obtained through a series of profiling steps [12]. Although the reverse of the fraction above would make more sense as a proper HRF, by this definition, this metric can be directly used in distributing an incoming task's power according to the following formula:

$$\text{PwrVctr}_j = \frac{HRF_j}{\sum_{i=1}^{n} HRF_i} P_{\text{C}}, \quad \text{where } P_{\text{C}} = \sum_{i=1}^{n} P_i \text{ (Eq. 1)}, \quad (10)$$

i.e., given a total power P_{total} and the set of HRFs for the chassis, distribute the power (i.e. calculate a "power assignment") of an incoming task to each chassis j according to Eq. 10. A small HRF value indicates that a chassis is a strong recirculation contributor, so it will be assigned less workload.

2) MinHR-modified: A direct comparison with MinHR is not possible for two reasons: (a) MinHR cannot be directly applied to the problem defined in Section II-E, because it is designed to allocate power assignments and not tasks; (b) Eq. 10 may calculate a power assignment to a server that exceeds its maximum capacity. For a proper and fair comparison of MinHR with XInt-GA and XInt-SQP, we propose a *MinHR-modified* (MinHR-m) whose total (computing) power to be assigned is taken from the resulting total (computing) power of XInt-GA's solution. Also, if the assigned power consumption for some chassis is larger than its peak power consumption or less than its idle power consumption, then this server' power consumption is set to the maximum or minimum possible and the difference from the one calculated in Eq. 10 is added or subtracted from P_{total}, respectively; the procedure is repeated on the remaining available chassis until all power is assigned. The pseudocode for MinHR-m is given by Algorithm 3.

V. POWER PROFILING

As mentioned in Section II-D, the formulation relies on a linear power model with respect to utilization. This section presents an overview of power models and the power profiling process performed at the ASU Fulton HPCI data center. The results confirm that a linear utilization-power model is a valid assumption, at least for the system measured at the data center, i.e. the Dell PowerEdge 1855. Variation in other component utilization, such as disk or memory I/O rate, in these systems does not significantly vary the power consumption.

A. Power Models

To our knowledge, most of the algebraic power models that express power in terms of computer system component usage are linear with respect to CPU utilization (as opposed to other computer components such as hard disk, memory or NIC). Linear correlation between component utilization and power consumption have been assumed in [19], where thermal predictions are performed based on component power. Similar linearity is assumed in chassis power consumption in [14], where thermal evaluation of the data center is performed based on power estimation at each chassis.

B. Processor Utilization and Power Consumption Correlation

In linear power models, the server power has two parts: i) fixed power consumption, which is the idle server power, and ii) power consumption which varies linearly with the server's CPU utilization. Parameters of the linear models are obtained through calibrations based on measurements in real systems. Different mechanisms (such as performance counters for processors [19]) have been used for this purpose. For data centers, however, prediction of chassis level power consumption is required [14], as the servers are powered through the chassis, and the heat dissipation depends on the aggregate power profile of the chassis.

If $b_\circ$ is the base power consumption to run the chassis power unit, m is the number of servers in the chassis, and b_I is the idle server power consumption, then the fixed idle power consumption of the chassis (b) is given as

$$b = b_\circ + m b_\mathrm{I}.$$

Due to the linearity of the power consumption with the CPU utilization, the power consumption $P_\mathrm{s}(u)$ at a server due to processor utilization u can be characterized as

$$P_\mathrm{s}(u) = (b_\mathrm{T} - b_\mathrm{I})\, u = a\, u,$$

where b_I is the power consumption of the server in idle state (0% CPU utilization), and b_T is the power consumption of the fully utilized server (100% CPU utilization). Note here that the total power consumption for a single server has to account for the idle power consumption b_I, thereby giving the total power consumption $P(u)$ for a single server as

$$P(u) = b_\mathrm{I} + (b_\mathrm{T} - b_\mathrm{I})\, u = b_\mathrm{I} + a\, u.$$

The total chassis power consumption can be given as

$$P_\mathrm{c}(m,u) = b_\circ + m P(u) = b + m P_\mathrm{s}(u),$$

where u is the CPU utilization at each server in the chassis.

C. Instrumentation

We performed power measurements of Dell PowerEdge 1855 blade systems using the DUALCOM power meter from CyberSwitching Inc. The blade servers are powered from the chassis. The power meter is connected between the chassis and its power supply to measure the current drawn by the chassis (as shown in Figure 4). We used the SNMP-based *CSTools* utilities supplied by CyberSwitching to retrieve the information from the power meter. The product of the current drawn (in amperes) with the supply voltage (208 V) of the ASU data center gives the power measurements (in watts). Table I further provides the details of the Dell PowerEdge 1855 model used for the experiments.

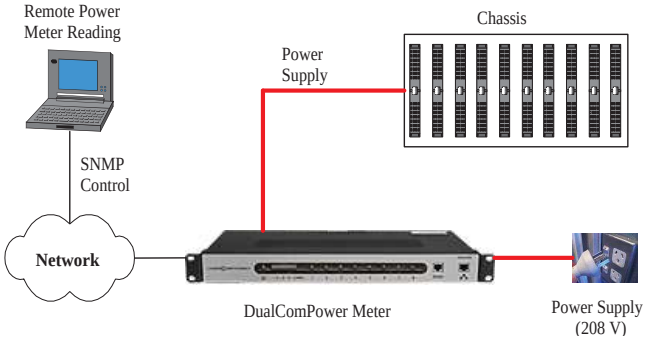


Fig. 4. Experimental Set-up for Power Measurement

D. Power Profiling Methodology

The power consumed by the chassis is an aggregation of the base chassis power consumption and the power consumption of the servers in the chassis. We obtained the base chassis power consumption ($b_\circ$ in Eq. 11) by taking power measurements for the empty chassis (i.e. with no servers in the chassis). The increase in chassis power consumption after the inclusion of a single idle server in the chassis gives the idle server power consumption b_I. In order to obtain the $P_\mathrm{s}(u)$ values in Eq. 11, we ran different applications on the server in the chassis. The power consumption of a single server, $P_s(u)$, is obtained by measuring the power consumption increase when running a single-threaded application on the server with respect to an idle (unutilized) server.

We perform the power-profiling of the servers in three steps:

1) *empty chassis* power measurement, to derive $b_\circ$;
2) *chassis power* measurement with a single blade server, to obtain $P(u)$;
3) *full chassis* power measurement, to obtain $P_\mathrm{c}(m,u)$.

In the following, we elaborate on the profiling steps:

1) Empty Chassis Power Consumption: first, to determine the constant power requirement of the chassis, we perform the power measurements of the empty chassis for Dell PowerEdge 1855 server. This constant chassis power consumption along with the power consumption of the idle servers constitute the base power consumption for the chassis.

2) Single Server in a Chassis: in the second step, we perform the power measurements of the chassis with a single server in the chassis. These set of experiments were designed to observe the power consumption of a single server with different CPU utilization and disk I/O. We varied these parameters using Gamut (Generic Application eMUlaTor) (version 0.7.0) [22], a multi-threaded application that selectively utilizes parts of a single machine or networked servers. We employed the CPU and I/O modes of Gamut, and we configured it to run in both CPU and disk I/O intensive modes.

TABLE II
TARGET EQUIPMENT USED IN THE EXPERIMENTS

Model	Chassis	Processor	Disk	Memory	O/S
Dell PowerEdge 1855	7U Modular Chassis BMX v.1.3	2 Intel Xeon(R) uni-core @ 2.33GHz	Maxtor Ultra 320 SCSI 146GB @ 10K RPM	4GB Fully Buffered DIMM Memory	Linux 2.4

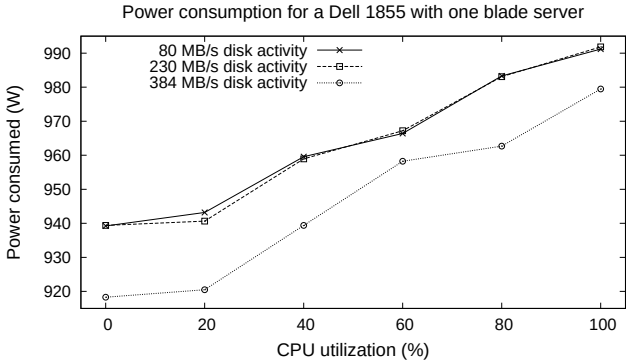


Fig. 5. Chassis Power Consumption for Single Server in Chassis.

3) Full Chassis Power Consumption: we performed the last set of experiments with a chassis full of servers. Dell PowerEdge 1855 chassis has 10 server slots. The objective of this experiment is to verify the chassis power consumption variation with multiple servers in it. We executed the same tasks (as in the second set of experiments) at the same time in all the servers in the chassis. Table II gives the breakdown of configurations used for the experiments.

E. Profiling Results

We observe that the empty chassis power consumption for PowerEdge 1855 is 820W. Fig. 5 shows the variation in power consumption for PowerEdge 1855 system: high disk I/O forces the CPU to idle cycles, leading to less power consumption; whereas a higher CPU utilization leads to a higher power consumption.

F. Analysis

In this section, we analyze the results in further detail. This analysis is required to identify the correlation between the task assigned to the servers in the chassis and the chassis' power consumption.

1) Linearity with CPU Utilization: We commence by analyzing power consumption of the single server in the chassis. When the server is idle (i.e. utilization is 0%), the total power consumption of the chassis is 940W. The difference of the empty chassis power consumption from this value gives the idle server power consumption as $b_{\mathrm{I}} = 940\,\mathrm{W} - 820\,\mathrm{W} = 120\,\mathrm{W}$. Similarly, at 100% CPU utilization for single server in the chassis, the total power consumption is $b_{\mathrm{T}} = 970\,\mathrm{W} - 820\,\mathrm{W} = 170\,\mathrm{W}$. Eq. 11 can be re-written as

$$P(u) = 120 + 50u.$$

The results for different CPU utilization can be verified from Figure 5. For example, at 50% CPU utilization, the chassis power consumption with a single server is $P_{\mathrm{s}}(0.5) + b_{\circ}$ (Replacing u in Eq. 11), which is $50 \times 0.5 + b_{\mathrm{I}} + b_{\circ} = 25 + 120 + 820 = 375\,\mathrm{W}$, which verifies the experimental results.

2) Linearity with Number of Servers in Chassis: we verify the total power consumption estimated by Eq. 11 with the experimental results. Parameter b in Eq. 11 has two parts: i) power required to run the chassis (i.e. empty chassis power consumption), and ii) power drawn by all the idle servers in the chassis. Based on the empty chassis and idle server power measurements, b for PowerEdge 1855 can be given as $b = 820 + 10 \times 120 = 2020\,\mathrm{W}$. Therefore, the parameters supplied to Eq. 4 and to the MPIT-TA formulation are: ***a*=50 W** and ***b*=2020 W**.

VI. SIMULATION AND RESULTS

The performance of the XInt algorithms has been tested using a simulated small-scale data center of 7U blade-server equipment. They were compared to the algorithms described in Section IV, both in terms of temperature distribution achieved, i.e. maximal inlet temperature, and in terms of cooling energy.

A. Simulation Setup

We used Flovent [23], a CFD simulation software to obtain the thermal distribution for the various scheduling algorithms. We simulated a small scale data center with physical dimensions 9.6 m × 8.4 m × 3.6 m (see Figure 6), which has two rows of industry standard 42U racks arranged in a typical cold aisle and hot aisle layout. The cold air is supplied by one computer room air conditioner, with the flow rate 8 m^3/s. The cold air rises from raised floor plenum through vent tiles, and exhausted hot air returns to the air conditioner through ceiling vent tiles. There are 10 racks and each rack is equipped with 5 chassis (marked from bottom to top as A, B, C, D and E), with every chassis having 10 servers of two processors each. This data center has 1000 processors. The total power consumption of the whole data center is 101 KW at idle state and 126 KW at full utilization rate.

Figure 7 shows the inlet temperature distribution when all the servers are idle. Obviously, the chassis located at the lower part of the rack (A and B) obtain plenty of cold air from the floor vents and have a lower inlet temperature, where chassis located at the upper part (E) of the rack experience a highest inlet temperature due to the insufficient supply of cold air.

B. Comparison with respect to temperatures

We simulated the MCE, XInt-GA and MinHR-m algorithms for an incoming task with a load of 50% of the data center's computing capacity. Fig. 8 shows the power consumption distribution for MCE, XInt-GA and MinHR-m (the graph

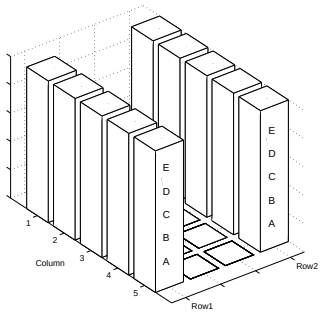


Fig. 6. Two-row data center used in our simulation study, each rack has five blade server chassis, marked from bottom to top as A, B, C , D and E.

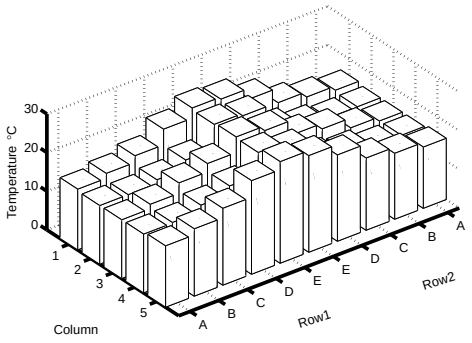
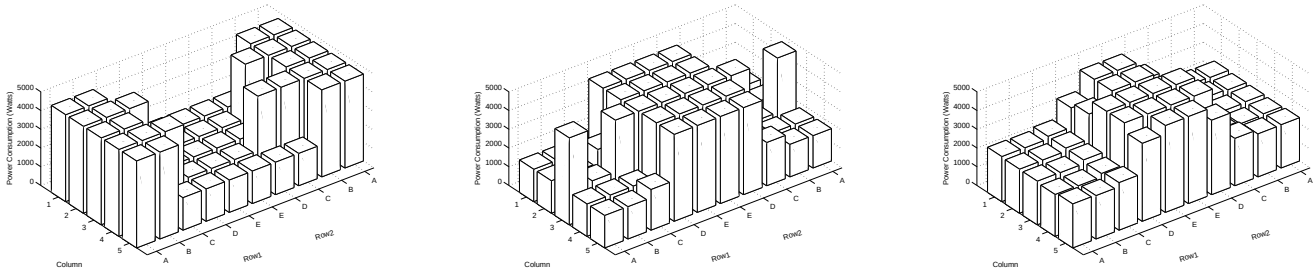
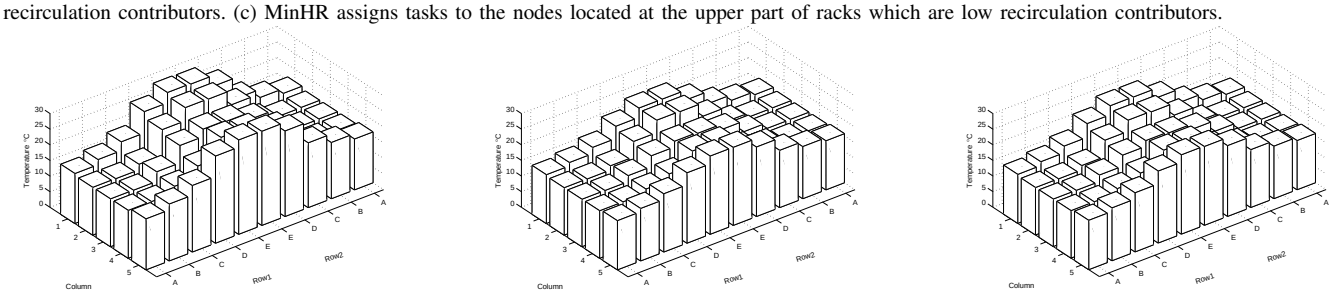


Fig. 7. Inlet temperature distribution at **idle**: chassis locate at the lower part of the rack obtain plenty of cold air from floor vent and have a low inlet temperatures



(a) MCE: power distribution (b) XInt-GA: power distribution (c) MinHR: power distribution

Fig. 8. Power distributions yielded by the placements of a task requiring 50% of the data center: (a) MCE assigns tasks to the nodes located at the lower part of racks which have relatively low inlet temperatures. (b) XInt-GA assigns tasks to the nodes located at the upper part of racks which are low recirculation contributors. (c) MinHR assigns tasks to the nodes located at the upper part of racks which are low recirculation contributors.



(a) MCE: inlet temperatures (b) XInt-GA: inlet temperatures (c) MinHR: inlet temperatures

Fig. 9. Inlet temperatures achieved by the algorithms for the same task placements above and for a T_{sup} of 15 °C. Inlet temperature distribution of (a) MCE, peak temperature is 30.5 °C; (b) XInt, peak temperature is 25.6 °C; (c) MinHR, peak temperature is 26.3 °C.

for XInt-GA approximates the distribution for XInt-SQP). The graphs also imply the task placement distribution. MCE assigns tasks to nodes with the lowest inlet temperature, which are located at the lower part of racks in our studied model, while XInt-GA and XInt-SQP, due to the MPIT-TA formulation, tend to place tasks at the top chassis. This is because the heat distribution matrix suggests that the lower chassis are larger recirculation contributors than the upper chassis. Similarly, MinHR-m also tends to place tasks at the top chassis because the HRF metric captures that the recirculation contributors are the lower equipment.

Placing tasks at the least contributing chassis has a direct impact on the inlet temperatures. Fig. 9 shows the resulting inlet temperature distributions for MCE, XInt-GA and MinHR-m. The peak inlet temperatures observed are 30.5 °C, 25.6 °C and 26.3 °C, respectively, with respect to a T_{sup} of 15 °C.

C. Comparing with respect to cooling cost

The roughly 5 °C temperature difference between MCE and XInt algorithms will result in significant difference in demand for the cooling capability. This is shown in Fig. 12 which shows the maximum supplied cold air temperature possible for the algorithms UOP, MCE, UT, XInt-GA, XInt-SQP and MinHR-m. Both XInt algorithms always exhibit the highest supply temperature, leading to the lowest cooling cost.

Computation of the cooling cost is performed using Eq. 2, where T_{sup} is set to such a value so that the peak inlet temperature matches the redline temperature. This T_{sup} is computed as follows: the output of any algorithm is a task placement; with an arbitrary initial $T_{sup} = T_{init}$, for example 0 °C, we calculate the $\max_i\{T_{in}^i\}$, then T_{sup} is adjusted using Eq. 7 (i.e. the T_{sup} is set to the resulting T'_{sup}). Effectively, the compared algorithms, due to differences in task placement, yield different T_{sup} values, thus resulting in different cooling costs.

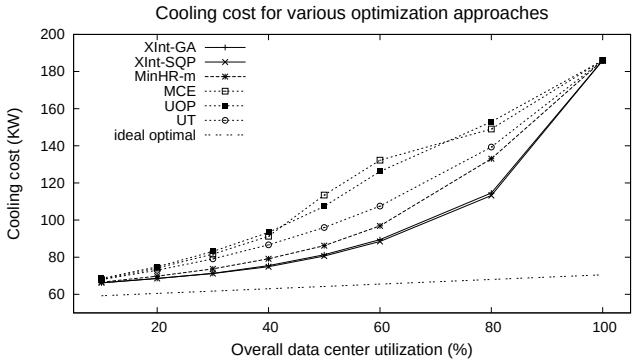


Fig. 10. XInt-SQP (closely followed by XInt-GA) achieves the minimal cooling cost, MCE's cooling cost is the worst one. The ideal optimal is achieved when there is no recirculation at all, computed by Eq. 11.

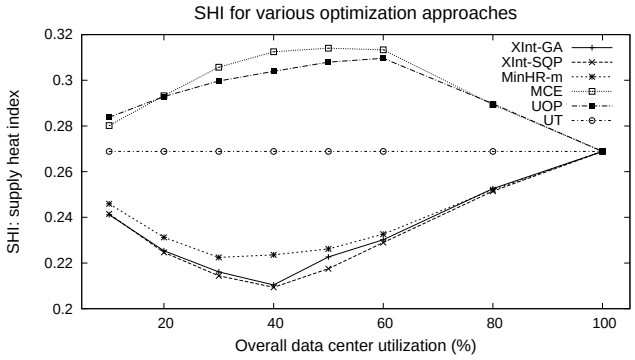


Fig. 11. Supply Heat Index of the compared algorithms: the result is consistent with Figure 10, XInt-SQP and XInt-GA have the minimal SHI since it is the recirculation minimized algorithm.

Fig. 10 shows cooling cost comparison for the four aforementioned algorithms. We observe that XInt-GA and XInt-SQP consistently have the minimal cooling cost. At 50% utilization rate, XInt-GA and XInt-SQP can save 24% to 35% power compared to UT and UOP. In addition, the performance of MCE has the worst cooling cost for most of the utilization rates. Fig. 10 also shows the ideal optimal for the cooling cost. The ideal optimal scenario assumes existence of no heat recirculation, while the supplied cold air and all the inlet temperatures are at the redline temperature of 25 °C:

$$P_{\text{AC-opt}} = P_{\text{C}}/\text{CoP}(25), \quad \text{from Eq. 2.} \tag{11}$$

MinHR-m outperforms all the naive algorithms in terms of cooling cost, but is slightly worse than the XInt algorithms.

D. Evaluation using Heat Indexes

The *Supply Heat Index* (SHI) and *Return Heat Index* (RHI) [8] are dimensionless (unitless), scalar metrics that measure the recirculation of heat in a data center. SHI is defined as

$$SHI = \frac{\text{Enthalpy rise due to infiltration in cold aisles}}{\text{Total enthalpy rise at the rack exhausts}}$$

and RHI is defined as $1-$SHI. These dimensionless metrics try to capture the "badness" of heat recirculation into one scalar value: the lower the SHI value, the better the energy efficiency.

Fig. 11 shows the SHI of the simulated scheduling algorithms. Better energy-efficient algorithms exhibit lower SHI values. The XInt algorithms achieve the lowest SHIs. It is also interesting that MCE, which expresses the idea of placing jobs at the lowest, coolest rows, has the largest SHI value.

E. Discussion

Intuitively, MCE and the traditional thermal engineer approach should provide a reasonably good thermal environment, because they place tasks in the locations with the lowest temperatures. However, this approach ignores the fact that the nodes with low inlet temperatures can be significant recirculation contributors and can cause the inlet temperatures of other nodes to rise, thus demanding extra cooling.

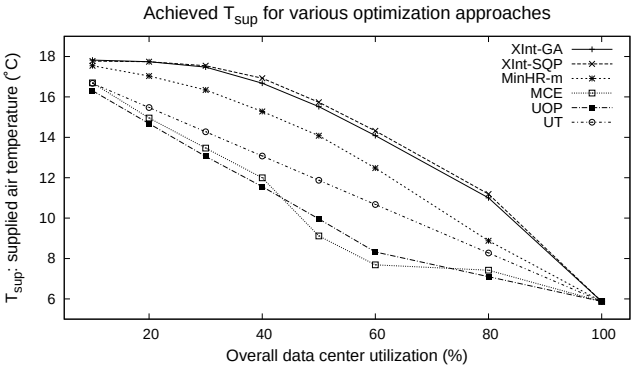


Fig. 12. Highest achievable supply air temperature for each placement algorithm. We can see that all algorithms match at 0% and 100% utilization.

The MPIT-TA solutions (i.e. the XInt algorithms) persistently achieve the best cooling cost, because they assign a task to nodes that only cause minimal recirculation. Consequently, minimizing recirculation equals to minimizing maximum inlet temperature. Hence, assigning a task to the nodes with the lowest inlet temperature does not necessarily result in a good temperature distribution. We have to assign a task based on the global information, more specifically, based on the global heat recirculation information. The difference between XInt-GA and XInt-SQP can be attributed to difference in the exploration of the solution space by each algorithm.

The performance difference between MinHR and the XInt algorithms can be explained as follows. First, the goal of original MinHR is to minimize the total amount of recirculated heat. However, this is not adequate for minimizing the cooling cost requirement; we also need to distribute the recirculated heat among all server nodes as evenly as possible to minimize the peak inlet temperature.

Secondly, the metric HRF used in MinHR-m characterizes an aggregated effect of recirculation (the total amount of recirculation from one node to all the other nodes), it does not show where the recirculated heat goes; whereas our cross interference matrix $\mathbf{A}$ shows the multi-point to multi-point recirculation among all server nodes (the amount of recirculation from any single node to another node).

Thirdly, MinHR-m distributes power consumption with re-

spect to the ratio of the heat produced over the heat recirculated. This is based on observation and intuition, and is not necessarily optimal. Instead, our work *mathematically formalized* the minimization of peak inlet temperature as the objective function of Eq. 8.

Finally, and most importantly, the MPIT-TA is a *task-oriented placement* problem formulation whereas MinHR-m's formulation is a *power-oriented workload placement* problem which has no capacity of placing tasks per se. In reality, a data center administrator does not receive the workload in terms of Watts but in terms of how many resources/processors are required. For example, it is not the case that an administrator receives a task submission indicating it needs 100 KW to run the task and figures out how to distribute the power consumption among server nodes. Instead, an administrator receives an incoming task that may need 300 servers and the problem is which 300 servers to select. Therefore, we believe our task-oriented approach is more applicable in data centers.

VII. Related Work

There exist *two steps* toward improving the thermal management at the data center level. The *first step* is from the infrastructure design and planning perspective. The *second* step, which is the focus of this work, is to improve and optimize the cooling cost, especially the temperature distribution, during the operation of a data center. Our contributions fall under the second category.

A. Improving the computation power efficiency

At **chip level**, the work on multi-core thermal management [24] tries to achieve thermal management through changing voltage or migrating process among multiple cores. The power-aware distributed computing for scientific applications project [25] is focusing on improving energy efficiency of large distributed and parallel computing systems by dynamically changing processor voltage without significantly affecting the system performance. Another similar work [26] discussed the problem of minimizing execution time while satisfying energy constraints and time constraints based on a voltage and frequency scalable cluster.

At **chassis level**, the work in [16] is based on the observation that typically data centers are under-utilized and proposes dynamically redistributing the power assignment to avoid inefficient over-provisioning in the cooling and power delivery. They suggested using some typical power control mechanisms such as voltage and frequency scaling.

At **data center level**, some research has also been conducted to reduce computing power cost [27], [28].

B. Reducing the cooling power requirements

Researchers at HP Labs and Duke University have published a series of work [9]–[12], [22] on smart cooling techniques for data centers. They have developed online measurement and control techniques to improve their energy efficiency.

MinHR [12] is a heat recirculation minimizing algorithm based on calculating the *Heat Recirculation Factor* (HRF) for each *pod*, i.e. usually a chassis or a rack, it assigns fewer tasks to pods that cause higher recirculation, while assigning more tasks to pods that cause less recirculation. Basically, it is a **power-oriented workload placement** algorithm instead of a **task placement** algorithm.

OnePassAnalog and Zone Based Discretization (ZBD) are proposed in work [12] to intuitively assign tasks inversely proportional to the server's inlet temperature. We believe they are similar to MCE, an intuition based algorithm that cannot guarantee the best energy efficiency.

The XInt algorithms presented in this paper falls in this category. It uses a more sophisticated yet fast-computable inlet temperature model to predict the temperature distribution at the inlets of a task placement.

Algorithms in this category do not compromise the performance of the servers in the data center to lower the temperatures. On the contrary, they may free servers from throttling by eliminating hot spots. In fact, among the above algorithms, only the MPIT-TA solutions predict to lower the standard deviation (i.e. the unevenness) of the inlet temperatures thus reducing the intensity of hot spots.

VIII. Formulation Extensions

In this section we provide extensions to the MPIT-TA formulation to place multiple tasks issued at the same time as well as placing those tasks in a partly utilized data center Both formalizations below are similar to the basic MPIT-TA formalization on page 5; they are expressed in a minimax formulation, and they can be solved in a similar manner, without a considerable increase the computational complexity.

A. Placing multilple tasks in an idle data center

The MPIT-TA formulation (page 5) can be extended to handle multiple tasks. For example, if we have to place two tasks, the problem would be to find two placement vectors $\mathbf{c}_1$, and $\mathbf{c}_2$ such that the *sum* vector $\mathbf{c}_1+\mathbf{c}_2$ minimizes the peak inlet temperature, i.e. $\mathbf{t}_{in}(\mathbf{c})=\mathbf{t}_{sup}+\mathbf{Db}+\mathbf{D}(\mathbf{c_1}+\mathbf{c_2})a$, while making sure that the sum doesn't use more than m servers per chassis. In general, having to place q tasks, the placement vectors can form a matrix; the problem is then defined as follows:

MPIT-TA for multiple tasks in an idle data center: Given:

- a data center of n chassis, each chassis having m servers with power characteristics a, b;
- q tasks, each task k demanding $c(k)$ servers;
- the heat distribution matrix $\mathbf{D}$;

find a task placement table $\mathbf{C}$ to:

$$\begin{aligned} &\text{minimize } \max_i\{T^i_{in}\} \\ &\text{such that: } c(k)-\sum_{j=1}^{n} c_{jk}=0,\ k=1\dots q, \\ &\mathbf{t}_{in}=\mathbf{t}_{sup}+\mathbf{Db}+\mathbf{D}\odot\mathbf{C}a, \\ &\sum_{i=1}^{q} c'_{ij}=c(j),\ j=1\dots n, \\ &m\geqslant\sum_{j=1}^{q} c'_{ij}\geqslant 0,\ i=1\dots n. \end{aligned} \tag{12}$$

where $\odot$ is the *row-wise dot product* of two matrices, yielding a vector[5]. Effectively, the multiple task vectors can be accu-

[5] In other terms, $A\odot B\equiv \mathrm{diag}\{AB^T\}$, i.e. the main diagonal of AB^T.

mulated into a single task vector, and compute the objective function as if the problem had to assign a single "supertask".

B. Placing multilple tasks in an partly-used data center

If non-preemptable tasks are already running, with a placement matrix $\mathbf{C}_\circ$, then the concatenation of the two matrices $\mathbf{C}_\circ$ and $\mathbf{C}$ is the overall placement vector:

MPIT-TA for multiple tasks in a partly utilized data center: Given:

- a data center of n chassis, each chassis having m servers with power characteristics a, b;
- p tasks already running on the servers, with $\mathbf{C}_\circ$ their $n \times p$ placement table;
- q tasks, each task k demanding $c(k)$ servers;
- the heat distribution matrix $\mathbf{D}$;

find a task placement table $\mathbf{C}$ in order to :

$$\text{minimize } \max_i\{T_{in}^i\} \tag{13}$$

$$\text{such that: } c(k) - \sum_{j=1}^{n} c_{jk} = 0, \;\; k = 1 \ldots q,$$

$$\mathbf{t}_{in} = \mathbf{t}_{sup} + \mathbf{Db} + \mathbf{D} \ominus [\mathbf{C}_\circ | \mathbf{C}] a, \tag{14}$$

$$\sum_{i=1}^{k} c'_{ij} = c(j), \; j = 1 \ldots n,$$

$$m \geqslant \sum_{j=1}^{p+q} c'_{ij} \geqslant 0, \; i = 1 \ldots n.$$

Eq. 14 can be re-written as: $\mathbf{t}_{in}$=$\mathbf{t}_{sup}$+$\mathbf{Db}$+$\mathbf{D}\ominus\mathbf{C}_\circ a + \mathbf{D}\ominus\mathbf{C}a$. In this form, we can view the $\mathbf{D}\ominus\mathbf{C}_\circ a$ term as part of the idle consumption, as we can only alter the matrix C.

IX. Conclusions

To improve the energy efficiency and, potentially, the performance of a data center, a peak inlet temperature minimization problem was formulated, MPIT-TA, which, when solved, allows for maximizing the supply temperature and thus minimizing the cooling energy needs. Two solution approaches to MPIT-TA were developed, XInt-GA and XInt-SQP, both of which find very good solutions, achieving of up to 30% cost savings for a simulated, small-scale data center, and outperforming other compared algorithms. The characteristics of this approach are:

- cooling-oriented thermal-aware placement,
- no performance compromise, under homogeneity of equipment
- accurate, low-complexity heat recirculation model.

Future work will focus on increasing the applicability of the MPIT-TA/XInt framework by reducing its dependence on simulation and measurements. Work-in-progress tries to obtain a power and recirculation model from built-in sensor and from O/S statistics logs, with the objective to waive the requirement for extensive simulation and manual effort in obtaining the model. Preliminary data from on-board sensors at the ASU Fulton HPCI center show a strong, near-linear correlation between utilization and outlet temperature that can be used to develop a model: Fig. 13 shows selected samples of utilization, inlet and outlet temperature graphs from blade server chassis.

X. Acknowledgments

We thank Dan Stanzione for granting access to the ASU Fulton HPCI Facility and its logs, Michael Jonas for processing the Fulton HPCI data, Tridib Mukherjee for performing and documenting the power measurements, and Ayan Banerjee for assisting with the simulations. We also thank Sanjay Rungta from Intel Corporation for his collaboration in power-profiling the equipment. Lastly, we thank the anonymous reviewers for their insightful comments and suggestions.

References

[1] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Thermal-aware task scheduling for data centers through minimizing heat recirculation," in *IEEE Cluster*, Sept. 2007.

[2] R. Mullins, "HP service helps keep data centers cool," IDG News Service, Tech. Rep., July 2007. http://www.pcworld.com/article/id, 135052/article.html.

[3] U. S. Congress, "H.R. 5646 [109th]: To study and promote the use of energy efficient computer servers in the united states," 2006. http://www.govtrack.us/congress/bill.xpd?bill=h109-5646.

[4] C. D. Patel, C. E. Bash, R. K. Sharma, A. Beitelmal, and R. J. Friedrich, "Smart cooling of datacenters," in *IPACK'03: The PacificRim/ASME International Electronics Packaging Technical Conference and Exhibition*, Kauai, HI, July 2003.

[5] R. F. Sullivan, "Alternating cold and hot aisles provides more reliable cooling for server farms," White Paper, Uptime Institute, 2000.

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

[7] C. Bash and G. Forman, "HPL-2007-62 cool job allocation: Measuring the power savings of placing jobs at cooling-efficient locations in the data center," HP Laboratories Palo Alto, Tech. Rep. HPL-2007-62, Aug. 2007.

[8] R. K. Sharma, C. E. Bash, and C. D. Patel, "Dimensionless parameters for evaluation of thermal design and performance of large scale data centers," in *American Institute of Aeronautics and Astronautics (AIAA)*, 2002, p. 3091.

[9] J. Moore, J. Chase, and P. Ranganathan, "Weatherman: Automated, online, and predictive thermal mapping and management for data centers," in *3rd IEEE Int'l Conf. Autonomic Computing*, June 2006.

[10] C. D. Patel, R. Sharma, C. E. Bash, and A. Beitelmal, "Thermal considerations in cooling large scale high compute density data centers," in *ITherm*, San Diego, CA, June 2002, pp. 767–776.

[11] M. H. Beitelmal and C. D. Patel, "Thermo-fluids provisioning of a high performance high density data center," Hewlett Packard Laboratories, Tech. Rep. HPL-2004-146, September 2004. http://www.hpl.hp.com/techreports/2004/HPL-2004-146.html.

[12] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling "cool": Temperature-aware resource assignment in data centers," in *2005 Usenix Annual Technical Conference*, April 2005.

[13] Q. Tang, N. Tummala, S. K. S. Gupta, and L. Schwiebert, "Communication scheduling to minimize thermal effects of implanted biosensor networks in homogeneous tissue," *IEEE Tran. Biomedical Eng.*, vol. 52, no. 7, pp. 1285–1294, July 2005.

[14] Q. Tang, T. Mukherjee, S. K. S. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters," in *ICISIP 2006*, Dec 2006.

[15] Q. Tang, S. K. S. Gupta, D. Stanzione, and P. Cayton, "Thermal-aware task scheduling to minimize energy usage of blade server based datacenters," in *IEEE DASC'06*, Oct 2006.

[16] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level power management for dense blade servers," in *ISCA'06*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 66–77.

[17] IT@Intel, "Energy-efficient performance for the data center,", 2006. http://www.intel.com/it/pdf/energy-efficient-perf-for-the-data-center.pdf

[18] J. Moore, R. Sharma, R. Shih, J. Chase, C. Patel, and P. Ranganathan, "Going beyond cpus: The potential of temperature-aware data center architectures," in *First Workshop on Temperature-Aware Computer Systems*, June 2004.

[19] T. Heath, A. P. Centeno, P. George, L. Ramos, and Y. Jaluria, "Mercury and Freon: temperature emulation and management for server systems," in *ASPLOS-XII*. New York, NY, USA: ACM Press, 2006, pp. 106–116.

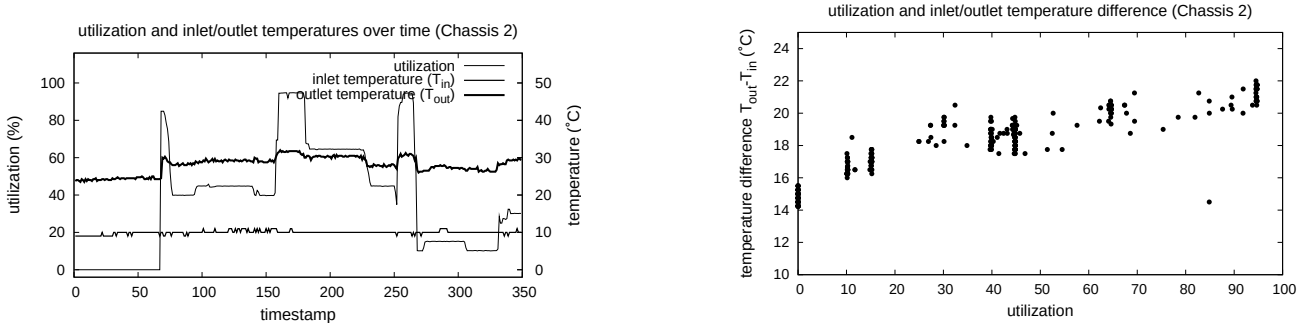


Fig. 13. Example graphs: **(left)**: utilization, inlet and outlet temperatures over time; **(right)**: outlet temperature vs. utilization generated from left.

[20] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level power management for dense blade servers," in *IEEE ISCA'06*, Boston, MA, May 2006, pp. 66–77.

[21] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. Springer-Verlag New York, Inc., 1994.

[22] J. Moore, J. Chase, K. Farkas, and P. Ranganathan, "Data center workload monitoring, analysis, and emulation," in *Eighth Workshop on Computer Architecture Evaluation using Commercial Workloads*, February 2005.

[23] "Flovent CFD simulation software." http://www.flomerics.com/

[24] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," *SIGARCH Comput. Archit. News*, vol. 34, no. 2, pp. 78–88, 2006.

[25] R. Ge, X. Feng, and K. W. Cameron, "High-performance, power-aware distributed computing for scientific applications," *IEEE Computer*, pp. 40–47, November 2005.

[26] R. Springer, D. K. Lowenthal, B. Rountree, and V. W. Freeh, "Minimizing execution time in MPI programs on an energy-constrained, power-scalable cluster," in *ACM SIGPLAN PPoPP '06*. New York, NY, USA: ACM Press, 2006, pp. 230–238.

[27] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," in *ACM SIGMETRICS Conference*. New York, NY, USA: ACM Press, 2005, pp. 303–314.

[28] T. Heath, B. Diniz, E. V. Carrera, W. M. Jr., and R. Bianchini, "Energy conservation in heterogeneous server clusters," in *ACM SIGPLAN PPoPP'05*. New York, NY, USA: ACM Press, 2005, pp. 186–195.

Sandeep Kumar S. Gupta is a Professor with the School of Computing and Informatics, Arizona State University, Tempe, USA. He received the B.Tech degree in Computer Science and Engineering (CSE) from Institute of Technology, Banaras Hindu University, Varanasi, India, M.Tech. degree in CSE from Indian Institute of Technology, Kanpur, and M.S. and Ph.D. degree in Computer and Information Science from Ohio State University, Columbus, OH. His current research focuses on **dependable, criticality-aware, adaptive distributed systems with emphasis on wireless sensor networks, thermal and power-aware computing and communication, and pervasive healthcare**. He has co-authored the book "***Fundamentals of Mobile and Pervasive Computing***," McGraw Hill, and is currently on the editorial board of IEEE Communication Letters and a co-guest editor for various IEEE journals. He is a member of the ACM and a senior member of the IEEE. Dr. Gupta heads the IMPACT (Intelligent Mobile and Pervasive Applications and Computing Technologies) Lab at Arizona State University. For information about his recent research projects and publications please visit http://impact.asu.edu.

Qinghui Tang received the B.S. degree in Telecommunication Engineering and the M.S. degree in Information Engineering from Beijing University of Posts & Telecommunications, China; and the PhD degree in Electrical Engineering from Arizona State University, Tempe, Arizona. He is currently with Texas Instruments as electrical design engineer. Dr. Tang's research interests include thermal management of computing systems. His publication list is available at http://impact.asu.edu/~tang/.

Georgios Varsamopoulos received the B.S. degree in computer and information engineering from University of Patras, Greece, the M.S. degree in computer science from Colorado State University, Fort Collins, Colorado, and the PhD degree in computer science from Arizona State University, Tempe, Arizona. He is currently a Research Faculty member with the School of Computing and Informatics at Arizona State University, Tempe, Arizona. Dr. Varsamopoulos' research interests include parallel and distributed computing, mobile and pervasive computing, algorithm analysis, and combinatorial analysis and optimization. His publication list is available at http://impact.asu.edu/~george/. Dr Varsamopoulos is a member of the ACM and the IEEE.