Spatio-Temporal Thermal-Aware Job Scheduling to Minimize Energy Consumption in Virtualized Heterogeneous Data Centers [1]

Tridib Mukherjee, Ayan Banerjee, Georgios Varsamopoulos, Sandeep K. S. Gupta *

Impact Lab
School of Computing and Informatics
Arizona State University
Tempe, Arizona, USA

Sanjay Rungta
Intel Corporation
Chandler, Arizona, USA

Abstract

Job scheduling in data centers can be considered from a *cyber-physical* point of view, as it affects the data center's *computing performance* (i.e. the cyber aspect) and *energy efficiency* (the physical aspect). Driven by the growing needs to green contemporary data centers, this paper uses recent technological advances in data center virtualization and proposes *cyber-physical*, *spatio-temporal* (i.e. start time and servers assigned), *thermal-aware* job scheduling algorithms that minimize the energy consumption of the data center under performance constraints (i.e. deadlines). Savings are possible by being able to temporally "spread" the workload, assign it to energy-efficient computing equipment, and further reduce the heat recirculation and therefore the load on the cooling systems. This paper provides three categories of thermal-aware energy-saving scheduling techniques: a) FCFS-Backfill-XInt and FCFS-Backfill-LRH, thermal-aware job placement enhancements to the popular *first-come first-serve with back-filling* (FCFS-backfill) scheduling policy; b) EDF-LRH, an online earliest-deadline-first scheduling algorithm with thermal-aware placement; and c) an offline genetic algorithm for SCheduling to minimize thermal cross-INTerference (SCINT), which is suited for batch scheduling of backlogs. Simulation results, based on real job logs from the ASU Fulton HPC data center, show that the thermal-aware enhancements to FCFS-backfill achieve up to 25% savings compared to FCFS-backfill with *first-fit* placement, depending on the intensity of the incoming workload, while SCINT achieves up to 60% savings. The performance of EDF-LRH nears that of the offline SCINT for low loads, and it degrades to the performance of FCFS-backfill for high loads. However, EDF-LRH requires milliseconds of operation, which is significantly faster than SCINT, the latter requiring up to hours of runtime depending upon the number and size of submitted jobs. Similarly, FCFS-Backfill-LRH is much faster than FCFS-Backfill-XInt, but it achieves only part of FCFS-Backfill-XInt's savings.

Key words: job scheduling, job placement, energy-efficiency, thermal-aware algorithms, heuristic algorithms, virtualized data centers

* Corresponding author.
Email address: sandeep.gupta@asu.edu (Sandeep K. S. Gupta).
URL: http://impact.asu.edu/ (Sandeep K. S. Gupta).

[1] This work was funded in parts by NSF (CNS#0649868 and CSR#0834797), SFAz and Intel.

1. Introduction

Reducing the energy consumption of data centers and making greener has been the interest of academic research, industry and state [1–8]. 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) emissions; this accounted for 2% of the total USA energy budget, while it is projected that it will reach 3% by the year 2010 [9]. A large portion of this energy (up to 50%) is consumed for cooling the data center. Various energy efficiency techniques are being developed at various levels; at chip level (low-voltage ICs [10]), at chassis level (ensemble-level power management [11]), at data center level (energy-efficient design [12], capturing and reusing the produced heat [13]). This paper focuses on energy-efficient thermal-aware job scheduling for high-performance computing (HPC) data centers. HPC data centers are oriented toward computation-intensive applications, e.g. simulations or large data set processing, that require many processors and may run for long periods of time.

Thermal-aware job scheduling describes any scheduling effort that has some degree of knowledge of the thermal impact (i.e. the effect on the heat and temperature distribution in the data center room) of a schedule to the data center. Thermal-aware job scheduling is a cyber-physical approach to scheduling which aims to produce schedules that yield minimal cooling needs and therefore are energy-efficient. It is cyber-physical because it takes into consideration both the cyber (computing) performance and the physical (energy, cooling) performance. Most of the previous work on thermal-aware job scheduling has focused on the spatial aspect of job scheduling, i.e. the server assignment of jobs [1–4]. Spatial thermal-aware job scheduling tries to address the effects of heat recirculation, a common phenomenon which is responsible for much of the energy inefficiency in contemporary data centers. Algorithms such as MinHR [2] and XInt [3] perform thermal-aware spatial job scheduling and manage to reduce the heat recirculation; they can effectively reduce the Supply Heat Index [14], a metric that describes the energy efficiency of a data center, by more than 15% for job sizes of about half the data center's capacity [3].

This paper investigates the challenges and benefits of extending the previous work to the temporal dimension, that is, the challenges and benefits of spatio-temporal thermal-aware job scheduling. Usually, HPC data centers contain equipment of various models and manufacturers, with various computing performance and power characteristics. A job is usually submitted with a runtime estimate, which is largely overestimated. The work in this paper leverages the equipment heterogeneity and slack in execution estimates, and shows that thermal-aware spatio-temporal job scheduling can yield a synergistic effect of energy efficiency benefits. For example, one result in this paper shows that spatial-only thermal optimization on an unaltered time schedule has an improvement of about 5% in energy savings, while a thermally optimized spatio-temporal job schedule for that sample can yield over double the savings. The synergy derives from the temporal smoothing of the workload, yielding a large pool of available servers at any time

for thermally efficient spatial placement of the jobs.

A real-world example that can benefit from spatio-temporal thermal-aware job scheduling is the *weekend effect*, where a burst of jobs is submitted on a Friday night and results are collected on Monday morning, causes inefficiency in data centers; using the common *first-come first-serve with back-filling* (FCFS-backfill) policy, a scheduler will cause a period of high and cooling-expensive data center utilization, followed by an energy-wasting period of idle servers. The weekend effect can be addressed with spatio-temporal thermal-aware job scheduling which is aware of the execution times and distributes the workload over the entire weekend thus achieving energy savings.

1.1. *Overview of the challenges and contributions*

In computing systems, there is a general rule of energy-latency trade-off. Results presented in this paper suggest that this trade-off exists in spatio-temporal job scheduling, where the throughput and turn-around time are sacrificed to gain energy efficiency. In addition to the energy-latency trade-off, there is a trade-off between the scheduler's runtime and the energy efficiency of the resulting schedule. These trade-offs are the major challenges in designing spatio-temporal thermal-aware algorithms; their optimal points are affected by:

– *job and overall performance requirements*: data centers operate for several years, while equipment is partially updated or added on an annual or semi-annual basis. The resulting equipment heterogeneity causes the execution time of a job to depend upon the server assignment. A spatio-temporal thermal-aware job scheduler must be able to estimate the execution time and decide if it can meet the job's requirements. Moreover, there are overall performance objectives that need to be met, such as throughput and average turn-around time. Generally, the job performance requirements and overall data center performance requirements give an upper bound on the energy savings that can be achieved. Specifically, in this paper, we use the slack between the user-estimated execution time and the actual execution time of the job to determine how much delay can be induced to improve energy efficiency without violating the user's expectation of the job's turn-around time.

– *decision-making speed*: the scheduler must be able to decide on a job in reasonable time. At the ASU's Fulton High Performance Computing Initiative (HPCI) data center examined in this paper, there are up to a few hundred daily submissions of jobs that last from a few minutes (short-running) up to several days (long-running). In this kind of HPC environment, it is desirable to have energy-efficient schedules, but it is still required not to affect the turn-around time of short-running jobs and jobs with little slack. The decision-making speed requirement sets an upper bound on the quality (i.e. energy efficiency) of a schedule that can be computed.

Virtualization is being widely used in data centers, mainly for two reasons: a) exploiting hardware transparency provided by virtualization to run software developed for a specific platform on any platform; and b) consolidating a data center's infrastructure on less equipment and consequently reducing the energy consumption. In this paper,

we show that the spatial flexibility for job assignment provided by (a) can reap additional energy savings. Further, *first-come first-serve* with *back-filling* (FCFS-backfill) is a widely used job scheduling policy in contemporary data centers [15]. One aspect of this paper is to improve the energy-efficiency of the FCFS-backfill policy with thermal-aware spatial scheduling. Specifically, the paper addresses the following research questions:

(i) ***What are the energy savings of an FCFS-based temporal scheduling enhanced with optimal thermal-aware spatial scheduling?*** Finding an optimal solution to spatial scheduling would require exhaustive search; therefore, to answer this question, we enhance the FCFS-backfill temporal scheduling with the XInt heuristic for spatial scheduling. We chose XInt because it computes the best known near-optimal solutions [3]. Compared to a reference FCFS-backfill with *first-fit* placement (FCFS-Backfill-FF) algorithm, we observe energy savings of up to 10% for the tested job trace scenarios.

(ii) ***What are the potential energy savings of an optimal thermal-aware spatio-temporal scheduling?*** Again, as finding an optimal solution requires an exhaustive search, to answer this, we:

  (a) Use a heat recirculation model and formulate a non-linear optimization problem to minimize data center energy consumption by thermal-aware scheduling of the jobs.

  (b) Develop the SCINT heuristic, which uses a genetic algorithm technique to solve the aforementioned problem in an offline manner.

  Results show that SCINT produces a schedule that is the more energy-efficient compared to schedules from all the examined online algorithms, and achieves up to 12.5% savings with respect to FCFS-Backfill-FF.

(iii) ***What are fast approximations of FCFS-based thermal placement and of SCINT?*** Both FCFS-Backfill-XInt and SCINT take several minutes to compute the schedule for submitted jobs, which is impractical. These algorithms explore the solution space based on a genetic approach and estimate the temperature layout and energy consumption of each examined solution (i.e. schedule). To reduce the computation needed, we develop the FCFS-Backfill-LRH and EDF-LRH algorithms as the fast counterparts of FCFS-Backfill-XInt and SCINT, respectively. The fast algorithms solve a simpler optimization problem called *least recirculated heat* (LRH), whose implementation is based on pre-calculated server "rankings", and take only a fraction of a second to run, but sacrifice energy efficiency compared to SCINT and XInt. Moreover, the EDF-LRH algorithm uses the deadline information to spread the workload in a temporal manner and allow its LRH part to choose energy-efficient servers.

The intuition is that energy efficiency can be achieved by spreading the jobs in a temporal manner, thus allowing to choose "thermally smart" server assignments for the jobs in the spatial manner. In fact, these two techniques work synergistically to reap additional energy savings. However, energy savings by spatial spreading is not an option, as HPC jobs are generally not "malleable" with respect to the number of processors—they are submitted with a specific

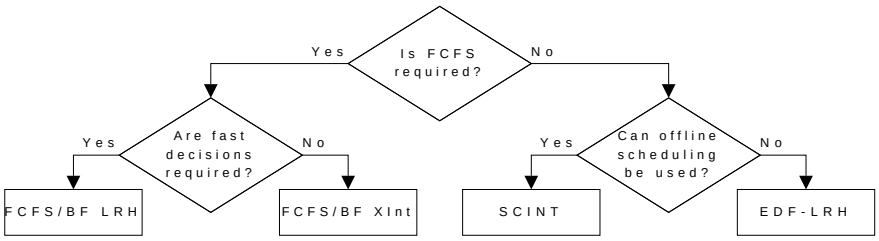

Fig. 1. A decision tree demonstrating the complementary nature of the proposed spatio-temporal scheduling algorithms.

number of processors requested. Typically HPC data center utilization is between 60% and 80% [3]. The algorithms presented in this paper smooth the workload, resulting in servers which are idle for longer durations of time and hence can be shut down with out negatively affecting the performance. Thus, the algorithms can leverage energy savings from *dynamic power-scaling* technologies, such as *energy-proportional computing* [16]. Results on shutting down idle servers show that extra energy savings can be achieved—savings exceeded 60% when using SCINT in one scenario with sparse workload.

The paper examines several scheduling heuristics with different characteristics. Choosing the appropriate scheduling technique depends on various performance and practical criteria, such as speed of decision making and whether FCFS has to be used. According to the results in this paper, we propose the decision tree in Figure 1 to be used as a guide in selecting the appropriate heuristic.

The rest of the paper is organized as follows. Section 2 presents the related work on data center scheduling, followed by the problem formulation in Section 3. The SCINT algorithm is presented in Section 4. Thermal-aware variants of FCFS-backfill algorithm and the EDF-LRH algorithm are presented in Section 5 and 6, respectively. Section 7 provides results on the energy consumption of the proposed algorithms for three example job trace scenarios. Finally, Section 8 concludes the paper.

2. Related Work

Recent economic and technological trends push for use of *virtualization* techniques to reduce data center operation costs. Both CentOS and Solaris operating systems offer *container virtualization* (or *O/S virtualization*), a type of *full virtualization* technology that allows a wide selection of hardware configurations to be supported in a data center. Another virtualization technology is *para-virtualization*, in which the availability of hardware resources is virtualized. The most prominent para-virtualization software is the `xen` virtual machine monitor. This paper assumes the existence of a container virtualization layer in the data center, so that any submitted job can be executed on any available server.

Job scheduling, in its general form, is NP-hard [17]. Online job schedulers are needed to make quick decisions, to ensure low overhead. For that reason, heuristic algorithms and policies are implemented. The *first-come first-serve* (FCFS) policy is extensively used, often augmented with *back-filling* to increase throughput [15]. With respect

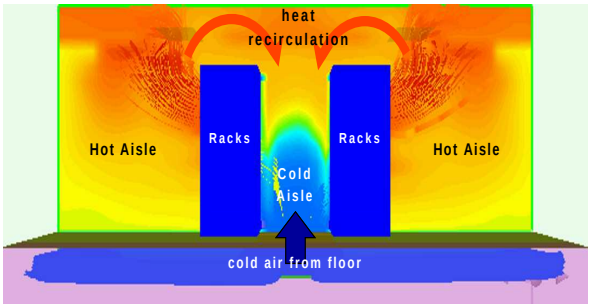

Fig. 2. Demonstration of heat recirculation: heated air in the hot aisle loops around the equipment to enter the air inlets.

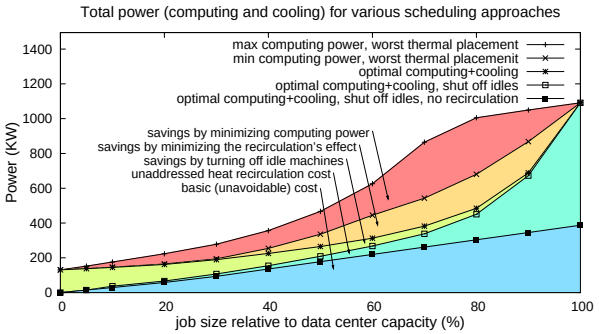

Fig. 3. Data center operation cost (in kilowatts) for various "savings modes". Savings are based on heat recirculation data obtained by FloVENT simulation of the ASU Fulton HPCI data center.

to cost savings, some research has included economical models to computing schedules [18]. Most of the schedulers implement the *first fit* policy in job placement, mainly for reasons of low overhead.

In data centers, the majority of energy inefficiency is attributed to *idle running servers* and to *heat recirculation* [2] (Figure 2). Solutions such as low-voltage ICs [10], ensemble-level power management [11] and energy-efficient design of data centers [12] which try to address these causes of inefficiency by reducing the inherent heat generation. Power control schemes, such as Freon-EC (an extension to the Freon power-aware management software which adds power control [4, 5]), or using energy-proportional systems [16] can help in addressing the energy inefficiency due to idle servers. To address heat recirculation, *energy-efficient thermal-aware* spatial scheduling algorithms have been proposed, such as MinHR and XInt [1–3]. Spatial scheduling can avoid or even prevent excessive heat conditions, while it can greatly reduce the cooling costs, which account for a large portion (about one third) of a data center's utility.

To demonstrate the magnitude of savings achieved by thermal-aware placement, and the thermal-aware placement's complementary relation to power control schemes, we provide Figure 3, which was produced using XInt and numerical results from previous spatial scheduling work on thermal-aware placement [3]: the top two lines in the figure represent the most energy-consuming schedules, and they were produced using a variant of XInt that maximizes the thermal inefficiency of the data center; the third line is the XInt curve as obtained in the previous work [3]; the fourth line represents the combined use of XInt and turning off idle servers, and it was obtained by removing the power consumption of all un-assigned servers from the XInt line; the bottom line represents the power consumption without heat recirculation. The figure shows that explicit power control and thermal-aware job placement are mutually complementary, with the former showing the most savings at low data center utilization rates and the latter at moderate to high (but not full) data center utilization rates. The figure also shows that power-aware yet heat-oblivious approaches that minimize only the computing power (second line from the top) do not save as much as thermal-aware approaches do.

The aforementioned thermal-aware job scheduling algorithms try to optimize the spatial scheduling (i.e. placement)

of the jobs to achieve energy savings. This paper extends the previous work to perform thermal-aware spatio-temporal scheduling.

3. System model and the SCINT problem definition

This section presents an analytical energy consumption model of a data center with respect to the job it executes, and the optimal spatio-temporal scheduling problem formulation, to be solved by the SCINT algorithm (Section 4).

3.1. *Typical layout and operation of an HPC data center*

Contemporary *high-performance computing* (HPC) data centers use raised floors and lowered ceilings for cooling air circulation, with the computing equipment organized in rows of 42U racks arranged in an aisle-based layout, with alternating cold aisles and hot aisles (Figure 2). The computing equipment is usually in blade-server form, organized in 7U chassis. Often, in data centers, server racks are provided with chiller doors, which cool down the hot air coming out of the blade servers before it enters the data center room [19]. The cooling of the data center room is done by the *computer room air conditioning* (CRAC), also known as the *heating and ventilation air conditioner* (HVAC). They supply cool air into the data center through the raised floor vents.

The state-of-the-art in commercially available data center management software follows a conventional job queuing and issuing paradigm that focuses on optimizing *performance policy metrics*, those usually being throughput and turn-around time. The user front-end of a data center is the *submission* interface, i.e. the interface of the *scheduler*, which decides when and where (i.e. what servers) the jobs to be run at. A *job submission* usually provides: a) the executable, b) the input data, c) the number of servers it requires and the *estimated runtime*, and d) other constraints such as a priority, and specific *computing node* preferences. A *computing node* is a chassis containing multiple blade servers. A data center houses one or more *compute clusters*, i.e. a collection of servers that are managed by the same resource manager.

Virtualized data centers consolidate clusters with different equipment into a single pool of equipment. This paper uses such capabilities of a virtualized data center to minimize the data center energy consumption through thermal-aware job management in the heterogeneous resource clusters. The differences in the job run times in different clusters have to be considered in order to ensure that the jobs are completed within their respective reservation times. There are three major steps in developing such an energy-efficient scheduling algorithm: i) determination of the data center energy consumption; ii) formulation of the optimization problem to minimize energy consumption while meeting job deadlines; and iii) designing heuristics to solve the formulated problem.

Table 1
Scalar Symbols and Definitions.

Symbol	Definition
n	The number of computing nodes
s_i	the total number of servers (blades) in node i
s_i^a	the available number of servers (blades) in node i
p	the number of jobs
c_k^{tot}	the number of servers (blades) job k requires
c_{ikm}	the number of servers at node i allotted to job k during time-slot m
α_{ik}	the power consumption of a server at node i running job k
ω_i	idle power consumption of node i's power unit
t_{ik}	time taken to perform job k in a server in node i
b_k	time slot when job k starts executing for the first time
f_k	time slot when job k finishes execution
a_k	time slot when job k arrives to the job queue
d_k	time slot for the deadline of the job k
L	number of time-slots in the observed time
τ	duration of each time-slot
P_{im}	Power consumption of node i during time-slot m [a]
T_m^{sup}	air temperature as supplied from the cooling unit during time-slot m
T^{red}	manufacturer's red-line inlet temperature
T_i^{in}	Inlet air temperature of node i
T_i^{out}	Outlet air temperature of node i
E^{total}	total energy consumption in the data center including, computing energy, cooling energy, and the migration energy over the observed time period
E^{comp}	computing energy over the observed time period
E^{AC}	cooling energy over the observed time period
E^{mig}	migration energy over the observed time period

[a] We assume a macroscopic power model, where the power consumption is averaged out for a job over time.

Table 2
Vector and Matrix Symbols and Definitions.

Symbol	Definition
$\boldsymbol{w}$	the vector $\{\omega_i\}_n$
$\boldsymbol{A}$	the matrix $\{\alpha_{ik}\}_{n\times p}$
$\boldsymbol{p}_m$	the vector $\{P_{im}\}_n$
$\boldsymbol{t}_m^{in}$	the vector $\{T_i^{in}\}_n$
$\boldsymbol{t}_m^{sup}$	the vector $\{T_m^{sup}\}_n$
$\boldsymbol{s}$	the vector $\{s_i\}_n$
$\boldsymbol{s}^a$	the vector $\{s_i^a\}_n$
$\boldsymbol{a}$	the vector $\{a_k\}_p$
$\boldsymbol{b}$	the vector $\{b_k\}_p$
$\boldsymbol{d}$	the vector $\{d_k\}_p$
$\boldsymbol{c}^{tot}$	the vector $\{c_k^{tot}\}_p$
$\mathbf{D}$	heat distribution coefficient matrix (see [20])
$\boldsymbol{t}$	the matrix $\{t_{ik}\}_{n\times p}$
$\boldsymbol{C}$	$n \times p \times L$ 3D allocation matrix representing the number of allocated processors during each node for each job during each time-slot (see Figure 4)
$\boldsymbol{C}_m$	$n \times p$ matrix representing the number of allocated processors at each node for each job during time-slot m
$\boldsymbol{C}_k$	$n \times L$ matrix representing the number of allocated processors at each node for job k during each time-slot
$\boldsymbol{C}_i$	$p \times L$ matrix representing the number of allocated processors at node i for each job during each time-slot
$\boldsymbol{C}_{km}$	vector of size n representing the number of allocated processors at each node for job k during time-slot m
$\boldsymbol{C}_{ik}$	vector of size L representing the number of allocated processors at node i for job k during each time-slot
$\boldsymbol{C}_{im}$	vector of size n representing the number of allocated processors at node i for each job during time-slot m

3.2. *Data center energy consumption*

For any observed time period, the data center energy consumption consists of both the cooling and the computing energy consumption. The observed time is divided into L time slots, each time slot m ($m = 1 \ldots L$) is of equal duration τ. This fragmentation discretizes the time, so that the problem can be solved by the genetic algorithm SCINT. The length of the time slots is a design choice, depending on the job submission frequency, job execution times, job migration times, the time to reach a thermal steady-state in the data center, and how much time is available to spend

on finding a schedule.

The efficiency of the cooling units is characterized by the coefficient of performance (CoP), which is the ratio of the heat removed over the work required to remove that heat. The data center cooling energy cost can be expressed as [2]:

$$E^{AC} = \sum_{m=1}^{L} \frac{P_m^{comp}}{\mathrm{CoP}\left(T_m^{sup}\right)} \tau, \tag{1}$$

where $\mathrm{CoP}(T_m^{sup})$ is the CoP for the cooling system to achieve a cool air temperature T_m^{sup} for time slot m. Note that CoP is typically super-linear with respect to the parameter T^{sup}, thus cooling efficiency improves with higher temperatures.

Further, the jobs may be reallocated to different servers dynamically in the virtualized data center. As a result, there would be energy expended to migrate the jobs. In fact, as jobs have various execution and arrival times, it is possible to observe cases in which placements that are optimal when they are decided, are later sub-optimal when some of the jobs finish or new ones start. In those cases, migration can serve as the mechanism for re-assigning the jobs to achieve more thermally efficient placements. The total energy consumption in the data center is the sum of the cooling energy, computing energy, and migration energy:

$$E^{total} = E^{comp} + E^{AC} + E^{mig}. \tag{2}$$

Using the CoP formulation from Eq. 1, we get

$$E^{total} = \sum_{m=1}^{L} \left(1 + \frac{1}{\mathrm{CoP}(T_m^{sup})}\right) P_m^{comp} \tau + E^{mig}. \tag{3}$$

This equation shows the relation between the coefficient of performance and the total cost.

3.3. *Effect of Heat Recirculation and Computing Power Distribution on the Total Energy Consumption*

To keep electronic devices working reliably, the thermal environment has to satisfy the constraint that the inlet temperatures of electronic devices should be kept under the red-line temperature T^{red} (with nominal values in the range of 25–35 °C). Ideally, any air that reaches the air inlets should come directly from the cooling system, thus the supplied cool air temperature could be set as high as the redline. However, due to the air recirculation phenomenon among the air outlets and inlets of the equipment, heat is also recirculated, thus causing the inlet temperatures to be higher than the supplied cool air temperature. Hence, the cooling system has to run at a lower temperature to keep all equipment under the redline, thus spending more energy according to the CoP.

According to the abstract heat model of the data center, from previous work [21], the vector of inlet temperatures $\boldsymbol{t}_m^{in} = \{T_i^{in}\}_m,\ i = 1, \ldots, n$ (where n is the number of nodes), for any time-slot m can be expressed as [20]:

$$\boldsymbol{t}_m^{in} = \boldsymbol{t}_m^{sup} + \mathbf{D}\boldsymbol{p}_m, \tag{4}$$

where $\boldsymbol{t}_m^{sup}$ is the vector $\{T_m^{sup}\}_n$, $\mathbf{D}$ is the *heat distribution* matrix (obtained from the Abstract Heat Recirculation Model [20]), $\boldsymbol{p}_m$ is the power consumption vector, i.e. $\boldsymbol{p}_m$=$\{P_{im}\}$, where P_{im} is the power of node i in time-slot m, and $\mathbf{D}\boldsymbol{p}_m$ is the vector of *recirculated heat* to each node i in time-slot m. In short, this equation expresses the inlet temperatures as a linear system with respect to the power vector. This means that each inlet temperature is above the supply temperature due to the excess recirculated heat. Note that *the row in the product* $\mathbf{D}\boldsymbol{p}_m$ *with maximum value determines the row in* $\boldsymbol{t}_m^{in}$ *with the maximum value*.

As the inlet temperatures have to be less than or equal to the redline, it follows that

$$\{T_i^{in}\}_m \leqslant T^{red},\ \forall i = 1, \ldots, n$$
$$\text{or } \max_i\{\boldsymbol{t}_m^{in}\} \leqslant T^{red}.$$

Applying Eq. 4 to the above inequality, we have:

$$T_m^{sup} + \max_i\{\mathbf{D}\boldsymbol{p}_m\} \leqslant T^{red} \ \Rightarrow T_m^{sup} \leqslant T^{red} - \max_i\{\mathbf{D}\boldsymbol{p}_m\}. \tag{5}$$

If we use the maximum possible T_m^{sup}, i.e. $T^{red} - \max\{\mathbf{D}\boldsymbol{p}_m\}$, then Eq. 3 expresses the E^{total} with the minimum cooling energy:

$$E^{total} = \sum_{m=1}^{L}\left(1 + \frac{1}{\text{CoP}(T^{red} - \max_i\{\mathbf{D}\boldsymbol{p}_m\})}\right)\sum_i \boldsymbol{p}_m\tau + E^{mig}. \tag{6}$$

This equation shows how $\boldsymbol{p}_m$ affects the total energy consumption.

Some assumptions behind this formulation are: (i) other heat sources in the room, such as lighting, are negligible compared to that of computing equipment; (ii) heat transfer through conduction is negligible compared to that through convection; (iii) the flow of air is steady and indifferent to the operating temperature.

3.4. *The effect of job scheduling on the total energy consumption*

This subsection provides a quantitative analytical model that maps job execution to the energy consumption of the data center. In blade-based systems, if a node (chassis) expends ω power at idle CPU, and each blade expends α power, the power consumption of a chassis with s blades is modeled as:

$$P = \omega + s\alpha.$$

Introducing a job-dependent and equipment-dependent aspect to the equation above, we denote the power consumption of a job k on a server in a node i as α_{ik}. It should be noted that for a given i and k, α_{ik} is same for any m. Since a node contains many servers, a node may run many jobs. In a high-performance computing data center, jobs usually require many servers. Therefore, if a job k runs on c_{ikm} servers on a node i at a time-slot m, then the power needs of that job at that time-slot on node i is $c_{ikm}\alpha_{ik}$. From the above, when a node i runs various jobs, where each job runs on c_{ikm} blades of that node, the power consumption of that node is

$$P_{im} = \omega_i + \sum_k c_{ikm}\alpha_{ik}. \tag{7}$$

Taking it a step further, we can construct a matrix $\mathbf{C}_m$ of the values c_{ik}, and a matrix $\mathbf{A}$ of the values α_{ik}, for a given time-slot m. If the vector $\boldsymbol{w}$ contains the values ω_i, the power vector $\boldsymbol{p}_m$ is expressed as:

$$\boldsymbol{p}_m = \boldsymbol{w} + \mathbf{C_m} \ominus \mathbf{A}, \tag{8}$$

where $\ominus$ is the *row-wise dot product* of two matrices, resulting into a vector[2]. In the above equation, values $\boldsymbol{w}$ and $\mathbf{A}$ are not adjustable; they are also independent of m. On the other hand, matrix $\mathbf{C}_m$ is adjustable and may vary over m depending on new job placements, old job completions, and running job migrations. The job scheduling decision involves the determination of the $\mathbf{C}_m$ matrices for any m in the observed time. The migration of the jobs is determined by the difference in the allocation matrices for consecutive time-slots (i.e. consecutive $\boldsymbol{C}_m$ matrices). Further, the differences in consecutive $\boldsymbol{C}_m$s include a job migration twice, because both the source and the destination nodes would change their respective c values. Therefore, for any job k, if P_k^{mig} and t_k^{mig} denote the power consumption and time required to migrate job, respectively, then the migration energy consumption can be calculated as:

$$E^{mig} = \frac{1}{2}\sum_{m=1}^{L-1}\sum_k\sum_i \left|c_{i,k,m+1} - c_{i,k,m}\right| P_k^{mig} t_k^{mig}. \tag{9}$$

We assume that the value of τ, i.e. the time-slot duration, is large enough to incorporate any job migration. Applying Eq. 8 and 9 to Eq. 6, we get that:

$$E^{total} = \sum_{m=1}^{L}\left(1 + \frac{1}{\mathrm{CoP}\left(T^{red} - \max_i\{\mathbf{D}(\boldsymbol{w}+\mathbf{C_m}\ominus\mathbf{A})\}\right)}\right)\sum_i\{\boldsymbol{w}+\mathbf{C_m}\ominus\mathbf{A}\}\tau + \tfrac{1}{2}\sum_{m=1}^{L-1}\sum_k\sum_i\left|c_{i,k,m+1} - c_{i,k,m}\right| P_k^{mig} t_k^{mig}. \tag{10}$$

It is now clear that *the total energy consumption depends on the scheduling of jobs in the data center*. Before formulating the energy-minimizing optimization problem for the SCINT algorithm, it is important to characterize the job turn-around time. The scheduling decisions have to ensure that the job turn-around time is within the deadline of the jobs. In the next subsection, the job turnaround times are characterized.

3.5. *Job submission, deadline, execution and turn-around time*

This subsection provides a derivation of the job turn-around time from the job scheduling decision. For that purpose, the set of matrices $\{\mathbf{C}_m\}$ is combined into a single three dimensional matrix $\mathbf{C}$ (Figure 4) of dimension $n \times p \times L$. The matrix $\mathbf{C}$ is referred to as the allocation matrix for the jobs in the observed time. The horizontal plane of the 3D matrix $\mathbf{C}$ in Figure 4 shows $\mathbf{C}_k$, which is a $n \times L$ matrix representing the allocation of a job k at every node at every time-slot. Similarly, we can represent the $p \times L$ matrix

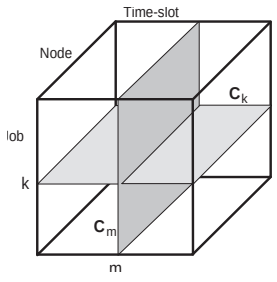

Fig. 4. 3D matrix $\mathbf{C}$ representing the number of processors allocated in every node for every job at every time-slot.

[2] Another way of expressing the $\ominus$ operator is $A \ominus B \equiv \mathrm{diag}\{AB^T\}$, i.e. the main diagonal of AB^T.

denoting the allocation of a every job on processor i at every time by $\mathbf{C}_i$. We follow a similar subscripting scheme to denote the vectors. For example, the vector representing the allocation of job k on every node in time-slot m is denoted as $\mathbf{C}_{km}$, the vector representing the allocation of every job on node i in time-slot m is denoted as $\mathbf{C}_{im}$, and the vector representing the allocation of job k on node i at every time-slot is denoted as $\mathbf{C}_{ik}$.

The execution of job k begins at the first time-slot m such that there is at least one non-zero element in the m-th column of $\mathbf{C}_k$. To determine the start time of any job k, we define a unary matrix operation $lz : \mathbf{C}_k \rightarrow \mathbb{Z}^+$, which calculates the number of leading columns with all zeros. $lz(\mathbf{C}_k)$ returns the number of time-slots (i.e. columns in $\mathbf{C}_k$) before the job k is allocated to any processor. The time-slot at which the job k begins execution is therefore given by $b_k = lz(\mathbf{C}_k) + 1$. If the job k arrives (i.e. is submitted to the queue) at the time-slot a_k, then the scheduling decision making should ensure that $b_k \geqslant a_k$ so that the job starts executing after its arrival. Similar to lz, another matrix operation $tz : \mathbf{C}_k \rightarrow \mathbb{Z}^+$ is defined to calculate the number of trailing columns in $\mathbf{C}_k$ with all zeros. This can determine the time-slot when job k finishes as $f_k = L - tz(\mathbf{C}_k)$, where L is the total number of time-slots in the observed time. This finish time of the job should be less than its deadline. If d_k denotes the time-slot for the deadline of job k, then the scheduling decision should ensure $f_k \leqslant d_k$. The difference between the start and finish times of job k (i.e. $f_k - b_k$) is the run time of the job. The turn-around time of job k is given by $f_k - a_k$.

The execution time of a job k at a node i can be calculated from the vector $\mathbf{C}_{ik}$. To this effect, we define vector operation, $nz : \mathbf{C}_{ik} \rightarrow \mathbb{Z}^+$, which calculates the number of non-zero elements in $\mathbf{C}_{ik}$. Consequently, the number of time-slots for which job k is allocated to node i is given by $nz(\mathbf{C}_{ik})$. This derives the execution time of job k on node i after multiplication with the time-slot duration τ and the subtraction by the migration time for the job. The migration time depends on the number of times migration has occurred. This number is given by $\frac{1}{2}\sum_{m=1}^{L-1}\left|c_{ik(m+1)} - c_{ikm}\right|$ (Section 3.4). Multiplication of this number with the average time for migration of job k generates the total migration time.

3.6. *Problem Formulation for SCINT*

The allocation matrix $\mathbf{C}$ is subject to the rules i) $\sum_{k=0}^{p} c_{ikm} \leqslant s_i \ \forall i$ and $\forall m$, which means that the assigned processors (cores) per chassis should be less than or equal to the available ones; ii) $\sum_{i=0}^{n} c_{ikm} = c_k^{tot} \ \forall k$ and $\forall m$ such that $b_k \leqslant m \leqslant f_k$, which means that the sum of assigned processors should be equal to the ones required by each job; and iii) $nz(\mathbf{C}_{ik})\tau - \frac{1}{2}\sum_{m=1}^{L-1}\left|c_{ik(m+1)} - c_{ikm}\right| t_k^{mig} \leqslant t_{ik}$, which means the execution of each job at each processor should not exceed the *estimated execution times*[3] . The problem then translates to:

[3] The execution times are estimated based on historical execution logs, for the equipment that logs are available, as well as on speed-up and slow-down factors obtained by *a-priori* comparative benchmarking of the computing equipment.

Problem 1 *Given:*

- *a data center of n nodes, each node i having: s_i processors (cores), and an idle chassis power consumption of ω_i;*
- *p jobs, each demanding c_k^{tot} processors (cores);*
- *a composite job-server power profile matrix $\mathbf{A} \equiv \{\alpha_{ik}\}_{n\times p}$;*
- *the heat distribution matrix $\mathbf{D}$;*

find a job allocation matrix $\mathbf{C} = \{c_{ikm}\}_{n\times p\times L}$, and:

minimize

$$E^{total} \equiv \sum_{m=1}^{L} \left(1+\frac{1}{CoP(T^{red}-\max_i\{\mathbf{D}(\boldsymbol{w}+\mathbf{C_m}\ominus\mathbf{A})\})}\right) \sum_i \boldsymbol{p}_m\tau + \frac{1}{2}\sum_{m=1}^{L-1}\sum_k\sum_i \left|\mathbf{C}_{m+1}-\mathbf{C}_m\right| P_k^{mig} t_k^{mig}$$

subject to

$$\sum_{k=0}^{p} c_{ikm} \leqslant s_i, \forall i = 1\ldots n, \forall m = 1\ldots L \qquad \{\textbf{\textit{Capacity constraints}}\}$$

$$\sum_{i=0}^{n} c_{ikm} = c_k^{tot}, \forall k = 1\ldots p, f_k \leqslant m \leqslant l_k \qquad \{\textbf{\textit{Server requirements}}\}$$

$$nz(\mathbf{C}_{ik})\tau - \frac{1}{2}\sum_{m=1}^{L-1}\left|c_{ik(m+1)}-c_{ikm}\right| t_k^{mig} \leqslant t_{ik}, \forall k = 1\ldots p, \forall i = 1\ldots n \qquad \{\textbf{\textit{Execution time constraints}}\}$$

$$f_k \leqslant d_k; f_k = L - tz(\mathbf{C}_k), \forall k = 1\ldots p \qquad \{\textbf{\textit{Deadline constraints}}\}$$

$$b_k \geqslant a_k; b_k = lz(\mathbf{C}_k)+1, \forall k = 1\ldots p \qquad \{\textbf{\textit{Arrival time constraints}}\}$$

4. The Offline SCINT Algorithm

The problem, as defined in the previous section, is a non-linear optimization problem. We solve it using a Genetic Algorithm (GA) based heuristic [22], which randomly searches through the feasible space. We briefly describe how to use a GA to find a near-optimal scheduling result. In short, a genetic algorithm is an iterative approach which is given an initial pool of *individuals* (i.e. feasible solutions). It then creates new solutions by mixing existing individuals (i.e. mating) and inserting random alterations in them (i.e. mutating). Thus the solution space is effectively explored to reach a near-optimal solution. Inefficient solutions are discarded based on a fitness function (i.e. a metric). A GA can escape from local optima because of the mating and mutation phases.

A feasible solution is the $n \times p \times L$ allocation matrix $\mathbf{C}$ for which the capacity constraint, server requirement, execution time constraint, job deadline and arrival time constraints in Problem 1 hold. A feasible solution is constructed as thus: a subset of the submitted jobs is selected in the order of arrival, and the allocation vector $\mathbf{C}_{km}$ is filled up with server assignments for the selected jobs such that at each time slot m the server assignments are exactly $\boldsymbol{c}_k^{tot}$ k for each job, and each job assignment occupies as many time slots as specified by the runtime estimation. The process is repeated until all jobs are assigned. The nodes are selected such that it does not violate the execution time constraint. Further, for the job k, the start time m is chosen such that the deadline constraints are also satisfied. Function BFS in Algorithm 1 implements this approach.

Solving Problem 1 using the GA approach we use E^{total} (Equation 10) as the fitness metric. The genetic algorithm starts with a pool of feasible solutions; specifically, we use the BFS function to compute the basic feasible solution and assign the same value to all individuals:

Algorithm 1 SCINT: Minimizing the Data center energy consumption

procedure SCINT($\boldsymbol{c}^{tot}$, n, p, L, $\boldsymbol{s}^a$, $\boldsymbol{a}$, $\boldsymbol{d}$, $\boldsymbol{t}$, T^{red}, $\mathbf{D}$, ω, A)
  CurGen ← a pool of BFS($\boldsymbol{c}^{tot}$, n, L, p, $\boldsymbol{s}^a$, $\boldsymbol{a}$, $\boldsymbol{d}$, $\boldsymbol{t}$) solutions
  for $x \leftarrow 1$ to *MaxGen* **do**
    SelSubs ← select subset from *CurGen* using roulette wheel
    MuSubs ← call MUTATE for mutation solutions in *SelSubs*.
    MaSubs ← call RECOMBINE for mating solutions in *SelSubs*.
    Apply the objective function E^{total} (Equation10) on *CurGen*, *MuSubs*, *MaSubs*.
    CurGen ← all fit solutions, i.e. ones with low total energy consumption.
  end for
  FinalSolution ← the solution within *CurGen* with best fitness.
end procedure

function MUTATE(*SelSubs*)
  for each randomly selected solution in *SelSubs* **do**
    for each job in a random pair of jobs in the solution **do**
      Select two random chassis ($c1$ and $c2$).
      Determine available cores, s^a_{c1}, in $c1$.
      if no. of cores in $c2$ for the job exceeds s^a_{c1} **then**
        Free s^a_{c1} cores for the job from $c2$.
        Allocate s^a_{c1} cores in $c1$ for the job.
      else
        Free all the cores for the job from $c2$.
        Allocate the required no. of cores in $c1$.
      end if
      If the deadline not met then penalize the new solution.
    end for
  end for
  return the new solution space.
end function

function BFS($\boldsymbol{c}^{tot}$, n, L, p, $\boldsymbol{s}^a$, $\boldsymbol{a}$, $\boldsymbol{d}$, $\boldsymbol{t}$)
  for $m \leftarrow 1$ to L **do**
    create set of jobs $OverlapJobs(m)$ that overlap in time slot m and can also fit the data center capacity.
    for $k \leftarrow 1$ to $OverlapJobs(m)$ **do**
      if $m \geqslant a_k$ & $m + max_i(t_{ik}) \leqslant d_k$ **then**
        for $i \leftarrow 1$ to n **do** BFS[i,m] = $min\{\boldsymbol{c}^{tot}, \boldsymbol{s}^a\}$
        end for
      end if
    end for
  end for
  return BFS
end function

function RECOMBINE(*SelSubs*)
  Select two random solutions, $S1$ and $S2$, in *SelSubs*.
  for each randomly selected job **do**
    Swap the start times and placements of the job in $S1$ and $S2$.
    for each of the two new solutions **do**
      if the capacity constraint is violated for any chassis **then**
        the excess no. of required cores are allocated in the chassis having largest number of available cores.
      end if
      If the deadline constraints are not met then penalize the new solution's fitness.
    end for
  end for
  return the new solution space.
end function

$$\left\{\text{BFS}(\boldsymbol{c}^{tot}, n, L, p, \boldsymbol{s}^a, \boldsymbol{a}, \boldsymbol{d}, \boldsymbol{t}), \text{BFS}(\boldsymbol{c}^{tot}, n, L, p, \boldsymbol{s}^a, \boldsymbol{a}, \boldsymbol{d}, \boldsymbol{t}), \ldots, \text{BFS}(\boldsymbol{c}^{tot}, n, L, p, \boldsymbol{s}^a, \boldsymbol{a}, \boldsymbol{d}, \boldsymbol{t})\right\}$$

The algorithm repeats the following three steps:

(i) a step of selection from the current pool of solutions using Roulette Wheel Selection algorithm;

(ii) a step of mating individuals (solutions) together, i.e. combining genes, to produce new solutions (procedure Recombine in Algorithm 1); and

(iii) a step of mutating individuals, i.e. randomizing the genes within an individual (see Mutate in Algorithm 1).

Three input parameters of a genetic algorithm 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. It should be noted that the deadline (i.e. the reservation time) may be underestimated by the user during job submission. As a result, the deadline constraint is not considered as a hard constraint, because for underestimated jobs it may not be possible to meet the deadlines even if the fastest processors are allocated. We handle this problem as follows: when an individual exceeds the deadline constraints d_k in the formulation, then the fitness of the solution is increased by a penalty value; the penalty is defined as the total energy the data center spends in the observed time running at 100% utilization. Thus if there exists a solution that does not satisfy the deadline constraint then the probability of selection of this solution is greatly decreased. The algorithm will always return a schedule even if that violates the deadline constraints for some

Algorithm 2 FCFS-Backfill-LRH

```
procedure INITIALIZATION( )
    Group the nodes with respect their power specifications
    Sort the groups with respect to nodes' computing efficiency
    (i.e. MIPS/watt).
    Perform victim ranking and victor ranking, according to Eq. 11
end procedure

procedure UPONJOBCOMPLETION( )
    Dispatch the next job in this group's queue using LRH.
    Remove the job from the queue.
end procedure

procedure LRH( )
    Place job on the available node(s) with the lowest victor ranking.
end procedure

procedure UPONJOBARRIVAL( )
    if job comes with node restrictions then
        Insert the job in the queue of the node group that it specifies.
    else
        Insert the job in the first (most energy efficient) group queue.
    end if
    for each node group, from the most efficient to the least one do
        Backfill jobs in the queue.
        If required number of nodes in this group are available,
        dispatch the first job in this group's queue using LRH.
        Remove the job from the queue.
    end for
end procedure
```

jobs.

Note that SCINT is a batch scheduling algorithm based on the overall knowledge of the job arrivals in the observed time, jobs' respective deadlines, and execution times in different servers. In addition to its *offline* nature, SCINT requires a lot of time to run. For an example input submission log of 40 jobs over three days, SCINT takes about 40 minutes to run. However, SCINT can be used to provide a lower bound on the energy consumption for other online heuristics, which we propose in the following section. These algorithms have online time-wise nature with thermal-aware placement, which are faster than SCINT—with a corresponding degradation in optimality. We propose two types of online algorithms: i) FCFS-backfill-based approach; and ii) EDF-based approach, which tries to approximate the scheduling behavior of SCINT.

5. FCFS-backfill based online approaches

FCFS-backfill is the most common scheduling algorithm used in the current data centers. We enhance this popular scheduling with two thermal-aware placement policies. In this paper, we propose two variants of FCFS-backfill: i) FCFS-Backfill-LRH, a low-complexity thermal-aware job placement extension to FCFS-backfill; and ii) FCFS-Backfill-XInt, a high-complexity thermal aware heuristic that minimizes the total power (cooling+computing) requirements. Both these algorithms are event driven in nature where FCFS-backfill is used to find the temporal schedule upon each job arrival. The algorithms use the procedures UponJobArrival and UponJobCompletion to specify the actions when a job arrives and finishes, respectively. The remaining functions and procedures in the algorithms depend on the specific spatial scheduling (i.e. job placement) algorithms used. Lastly, we need to mention FCFS-Backfill-FF, that is FCFS-backfill with *First-Fit* placement. A first-fit placement usually uses a hard-coded order of servers and it does not consider any thermal effects. Since FCFS-Backfill-FF is the prominent FCFS variant used in data centers, it is selected to be the basic approach to which all other algorithms are compared to.

5.1. *FCFS-Backfill-LRH*

In this algorithm, each node is ranked based on the total amount of recirculated heat, which originates from it. As per Equation 4, the recirculated heat reaching a node depends on its current power consumption and the heat recirculation coefficients to that node. The LRH algorithm pre-computes *victor rankings* for the servers as follows:

$$r_i = \sum_j v_j d_{ij} p_i^{\max}, \quad \text{where } v_j = \sum_i d_{ij} p_i^{\max}, \tag{11}$$

where p_i^{max} is the maximum power consumption at node i. Note that we use p_i^{max} instead of the current power consumption p_i so that r_i is statically computable; and consequently ensuring that LRH is of low complexity. The ranks v_j act as weight factors to the importance of the heat interference. The Initialization function in Algorithm 2 performs the pre-computation of the victor ranking. After a job is temporally scheduled using the FCFS-backfill scheduling algorithm, the job is assigned to the available (i.e. idle) servers with the minimal r_i (the victor ranking of server i) values (procedure LRH in Algorithm 2). Thus, FCFS-Backfill-LRH greedily exhausts the nodes with minimum recirculated heat.

5.2. *FCFS-Backfill-XInt*

This heuristic uses the XInt placement algorithm [3, 20] for spatial scheduling (procedure XInt in Algorithm 3). To the best of our knowledge, XInt is the best available thermal-aware job placement algorithm, which minimizes the energy consumption for homogeneous data centers [20]. XInt finds the number of servers to be assigned at each node for the set of scheduled jobs such that the peak inlet temperature is minimized. Effectively, XInt tries to minimize the $max\{\mathbf{t}_m^{in}\}$. FCFS-Backfill-XInt heuristic uses the XInt placement in conjunction with the popular FCFS-backfill algorithm to perform the spatio-temporal scheduling in heterogeneous data centers. Similar to SCINT, XInt also uses GA to find job allocation results. However, unlike SCINT, XInt deals with 2D ($n \times p$) matrices reducing the search space. Consequently, XInt is much faster than SCINT. However, as shown in the simulation results (Section 7.5), the run-time of FCFS-Backfill-XInt is higher than FCFS-Backfill-LRH.

6. EDF based online approach

One property of FCFS-backfill is that it maintains the burstiness of job arrivals subject to the the data center's capacity. The consequence is that FCFS-backfill can cause periods of high data center utilization, thus running at low energy efficiency state, when jobs could be spread out over time and still meet their deadlines. To counter this effect, we propose an online earliest-deadline-first with LRH placement. The EDF-LRH algorithm tries to execute as few jobs at the same time as possible. While a job is running, it queues up any incoming jobs and orders them in earliest deadline first order. If some job is projected to finish after its deadline, then it is scheduled to execute at an earlier

Algorithm 3 FCFS-Backfill-XInt

```
function BFS(c^tot, n, s^a)
    for i ← 1 to n do
        BFS[i] ← min{c^tot, s^a_i}
        c^tot ← c^tot − BFS[i]
    end for
    return BFS
end function

procedure UponJobArrival( )
    if job comes with node restrictions then
        Insert the job in the queue of the node group that it specifies.
    else
        Insert the job in the first (most energy efficient) group queue.
    end if
    for each node group, from the most efficient to the least one do
        Backfill jobs in the queue.
        If required no. of nodes are available, dispatch the first job
        from the queue using XInt placement.
        Remove the job from the queue.
    end for
end procedure

procedure UponJobCompletion( )
    Dispatch the next job in the queue using XInt placement.
    Remove the job from the queue.
end procedure

procedure XInt(c^tot, n, s^a, t^sup_m, D, p_m)
    Same as procedure SCINT in Algorithm 1.
▷ The BFS, Mutate, and Recombine functions are different in XInt.
end procedure

function Mutate( SelSubs )
    for each randomly selected solution in SelSubs do
        Randomly select a job in the solution.
        Select two random chassis (c1 and c2).
        Determine the no. of available servers, s^a_c1.
        if no. of servers allocated in c2 for the job exceeds s^a_c1 then
            Free the s^a_c1 servers for the job from c2.
            Allocate s^a_c1 servers in c1 for the job.
        else
            Free all the servers for the job from c2.
            Allocate the required no. of servers in c1 for the job.
        end if
    end for
return the new solution space.
end function

function Recombine( SelSubs )
    Select two random solutions from SelSubs.
    Select a random job in both the solutions.
    Swap the chassis allocation of the job in two solutions.
    if in any chassis the capacity constraint is violated then
        put the excess number of required servers in the chassis
        having the largest no. of available servers.
    end if
return the new solution space.
end function
```

time, thus having the data center run multiple jobs at the same time. Algorithm 4 gives an event-driven pseudo-code of this approach. It uses the procedures, UponJobArrival and UponJobCompletion, when a job arrives and finishes, respectively (Algorithm 4). The LRH procedure performs the job placement using the nodes' victor ranking (Section 5.1).

7. Performance comparisons

This section presents the performance comparison results of SCINT, EDF-LRH, FCFS-Backfill-LRH and FCFS-Backfill-XInt with the widely used FCFS-Backfill-FF algorithm. The goal is to capture the energy savings and the impact on the throughput and turnaround times in the proposed online and off-line algorithms. Since we do not have permissions to re-configure any live data center, the performance comparison is performed based on the simulations using job traces and physical dimensions from the ASU Fulton HPCI data center. Performing the simulations on the proposed formulation and algorithms requires three pieces of information:

(i) a set of jobs and their power profile values $\boldsymbol{w}$ and $\mathbf{A}$.

(ii) a data center setting simulator, including values such as size of room, air flow, specific heat capacity of air etc, in order to produce the heat distribution matrix $\mathbf{D}$.

Algorithm 4 EDF-LRH: Earliest Deadline First with Least Recirculated Heat placement

```
procedure INITIALIZATION( )
    Group the nodes with respect to their power specifications
    Sort the groups with respect to node computing efficiency
    (i.e. MIPS/watt).
    Perform victim ranking and victor ranking, according to
    Eq. 11.
end procedure

procedure LRH( )
    Place job to the available node(s) with the lowest victor
    ranking.
end procedure

procedure UPONJOBCOMPLETION( )
    Dispatch the next job in this group's queue using LRH
    placement.
    Remove the job from the queue.
end procedure

procedure UPONJOBARRIVAL( )
    if job comes with node restrictions then
        Insert the job in the queue of the node group that it specifies.
    else
        Insert the job in the first (most energy efficient) group queue.
    end if
    for each node group, from the most efficient to the least one do
        Sort the jobs queued in this group in Earliest Deadline First order.
        for each job in the queue do
            if job's finish estimation > deadline then
                (1) Insert the job in an "opening" which has enough free
                servers for enough time. Continue with next job.
                (2) If Step 1 fails, push-fit the job at an earlier "slot" if shifting
                jobs further down in time still make the deadline.
                Continue with next job.
                (3) If Step 2 fails, add the job to the queue in the next group.
            end if
        end for

        if required nodes in this group are idle then
            Dispatch the job in this group's queue using LRH placement.
            Remove the job from the queue.
        end if
    end for
end procedure
```

(iii) the arrival times (a_k), the deadlines (d_k), the estimated execution times (t_{ik}) of the jobs, and the number of processors required (c_k^{tot}).

The performance comparison section is organized as follows: Subsection 7.1 provides details on the simulated environment. Subsection 7.2 shows the method followed and the data obtained in power profiling various jobs at the ASU data center equipment. Subsection 7.3 discusses the job profiles in terms of job arrivals, reservation times, and execution times from the ASU data center job traces, used for the simulation. Subsection 7.4 provides a discussion on performing the simulations. Lastly, Subsection 7.5 elaborates on the results obtained.

7.1. *Simulation Setup*

We used FloVENT [23], a CFD simulation software to conduct thermal simulations to obtain heat distribution matrix used by the XInt and SCINT algorithms. Based on the ASU HPCI data center physical layout, we created a data center simulation model with physical dimensions 9.6 m × 8.4 m × 3.6 m, 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 ten racks and each rack is equipped with five 7U (12.25-inch) chassis (marked from bottom to top as A, B, C, D and E). There are two different types of computing equipment in the data center. Among the total fifty chassis, there are thirty Dell PowerEdge 1955 chassis (i.e. three racks) and twenty Dell PowerEdge 1855 chassis.

Table 3
Power Consumption Parameters

	ω	α
PowerEdge 1855	1820	72
PowerEdge 1955	2420	175

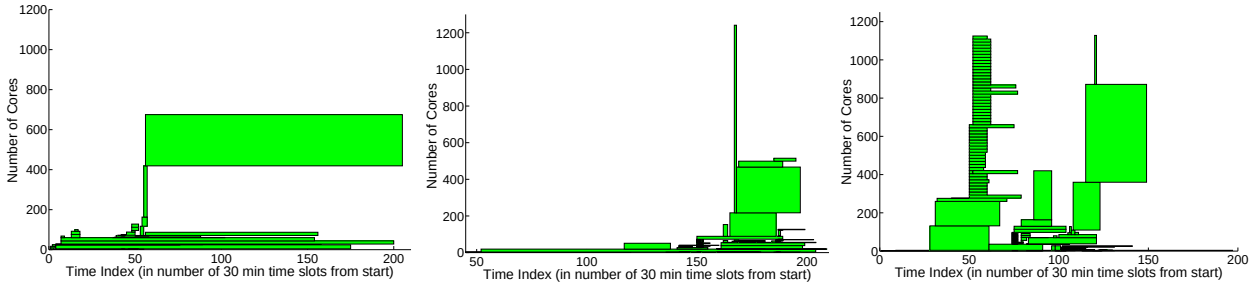

Fig. 5. Two-dimensional Gantt-like plots of three submission log snapshots of the ASU HPC data center used as simulation scenarios. Each job is depicted as a rectangle with the arrival time as the abscissa of its lower left corner, the estimated execution time as its length, and the requested numbers of processors as its height.

7.2. *Equipment Power Consumption*

We performed power measurements of Dell Power Edge 1855 and 1955 blade servers using the DUALCOM [24] power meter from CyberSwitching Inc. Using the power measurements on the blade systems and performing linear regressions on the data, we provide the idle chassis power consumption (ω) and the single fully-utilized server power consumption (α) values for the simulation runs [3, 25], as given in Table 3. The simulations assume that the jobs are CPU-intensive. The estimated power consumption of the resulting linear function has an error of 0.4–9% from the actual measurements.

7.3. *Data center job profile*

We used the ASU data center job traces of around one and half year (from Feb 2007 to Jul 2008) for the simulation. The job traces provide: i) the job arrival times (i.e. the a_k), ii) their corresponding deadlines (i.e. the d_k), iii) the number of processors required (i.e the c_k^{tot}), and iv) the job start and finish times using the FCFS-backfill scheduling.

Figure 5 shows the distribution of the job arrival, and their deadlines. We further estimate the job execution times (i.e. $t_{ik}s$) on the data center equipment based on the actual execution times (calculated by taking a difference between the finish times and the corresponding start times) in the ASU job traces.

To estimate the execution times for the other type of node, we simply multiply the estimated execution time for one equipment with the average gain in execution time in the other. This gain is calculated from the execution time measurements from the Standard Performance Evaluation Corporation (SPEC) [26, 27] in the Dell PowerEdge 1855

and 1955 servers. From the estimates, we deduce a speed-up of 2.5 on the Dell PowerEdge 1955 for the jobs initially running on a Dell PowerEdge 1855.

7.4. *Performing the simulations*

We obtain job scheduling of SCINT first and compare total energy consumption over an observed time-period with the FCFS-backfill scheduling algorithm with both the first-fit placement and the XInt placement. Using FloVENT simulations, we can produce the heat distribution matrix $\boldsymbol{D}$, as shown in a gray-scale-coded format in Figure 6. After this step, all other calculations have been performed using MATLAB 2007b. The period of each time-slice (τ) is assumed to be 30 minutes in the simulation. This however is a design parameter which should be large enough so that the thermal steady state is reached within each time-slice in case the T^{sup} is changed. Based on our experiments in the ASU data center, it takes around 8 minutes to reach the thermal steady-state. Further, τ can not be too large because it increases the "internal fragmentation" of time slots, where the servers remain idle between the end of their job and the end of the time slot. We chose τ as 30 minutes as the job deadlines are provided as a multiple of 30 minutes. Further it provides a reasonable time after the thermal steady-state is reached and at the same time ensures low internal fragmentation.

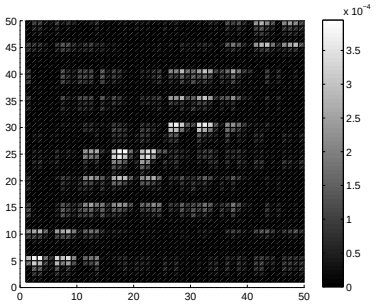

Fig. 6. The matrix $\boldsymbol{D}$ of heat distribution coefficients used in the simulations. The average temperature error using this matrix is 0.38 ℃.

We implemented SCINT, EDF-LRH, FCFS-Backfill-LRH, FCFS-Backfill-XInt, and FCFS-Backfill-FF algorithms and executed them for different job traces with various data center utilization. Based on the scheduling and placement results the throughput, turnaround time, and energy consumption are calculated. All the simulations are repeated for two different cases – i) idle servers kept running, and ii) idle servers turned off. For the second case, an exact solution would require 2^n different heat distribution matrices to account for all power combinations; for that reason we use the same heat distribution matrix, leaving the verification of whether it is a good approximation as future work. For the cases of SCINT and EDF-LRH, which require a knowledge of deadline, the slack times were used as deadline indicators.

7.5. *Simulation Results*

Figure 7 shows the power consumption for FCFS-Backfill-FF, EDF-LRH, FCFS-Backfill-XInt and SCINT scheduling, for three selected submission snapshots (scenarios), when having idle servers on and when having idle servers off. The three scenarios were selected to represent all ranges of load (light, medium and high). For all the cases, the

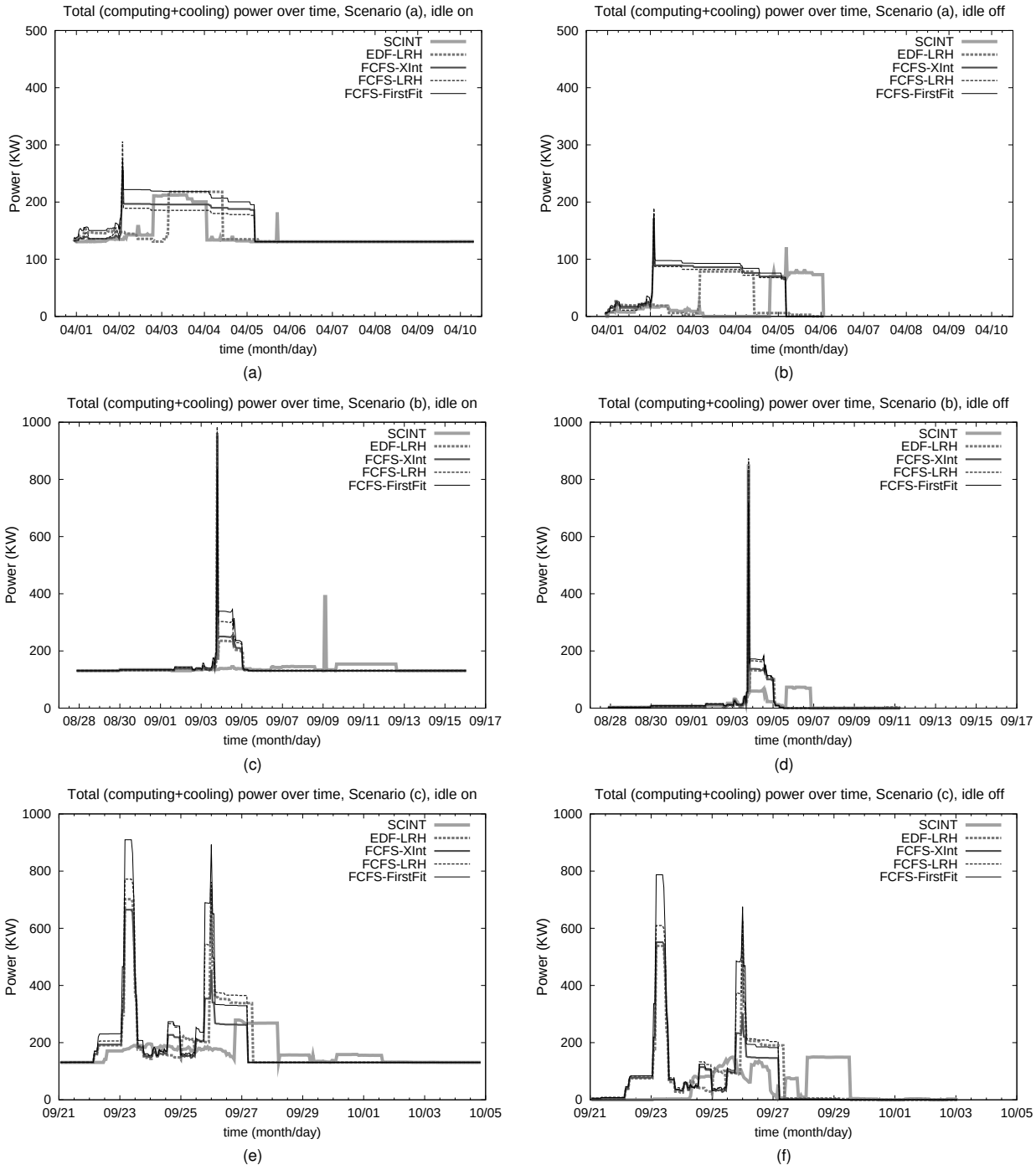

Fig. 7. Power comparison of the simulated schemes for the three scenarios. The plots in the left column correspond to an idle-on policy, while t he plots in the right column correspond to an idle-off policy.

SCINT algorithm distributes the execution of the jobs over time while meeting the respective reservations (if the job reservations are not underestimated). We will refer to the cases where idle servers are kept on as *"idle-on" cases*, and the cases where idle servers are turned off as *"idle-off" cases*. This is unlike the FCFS-backfill scheduling

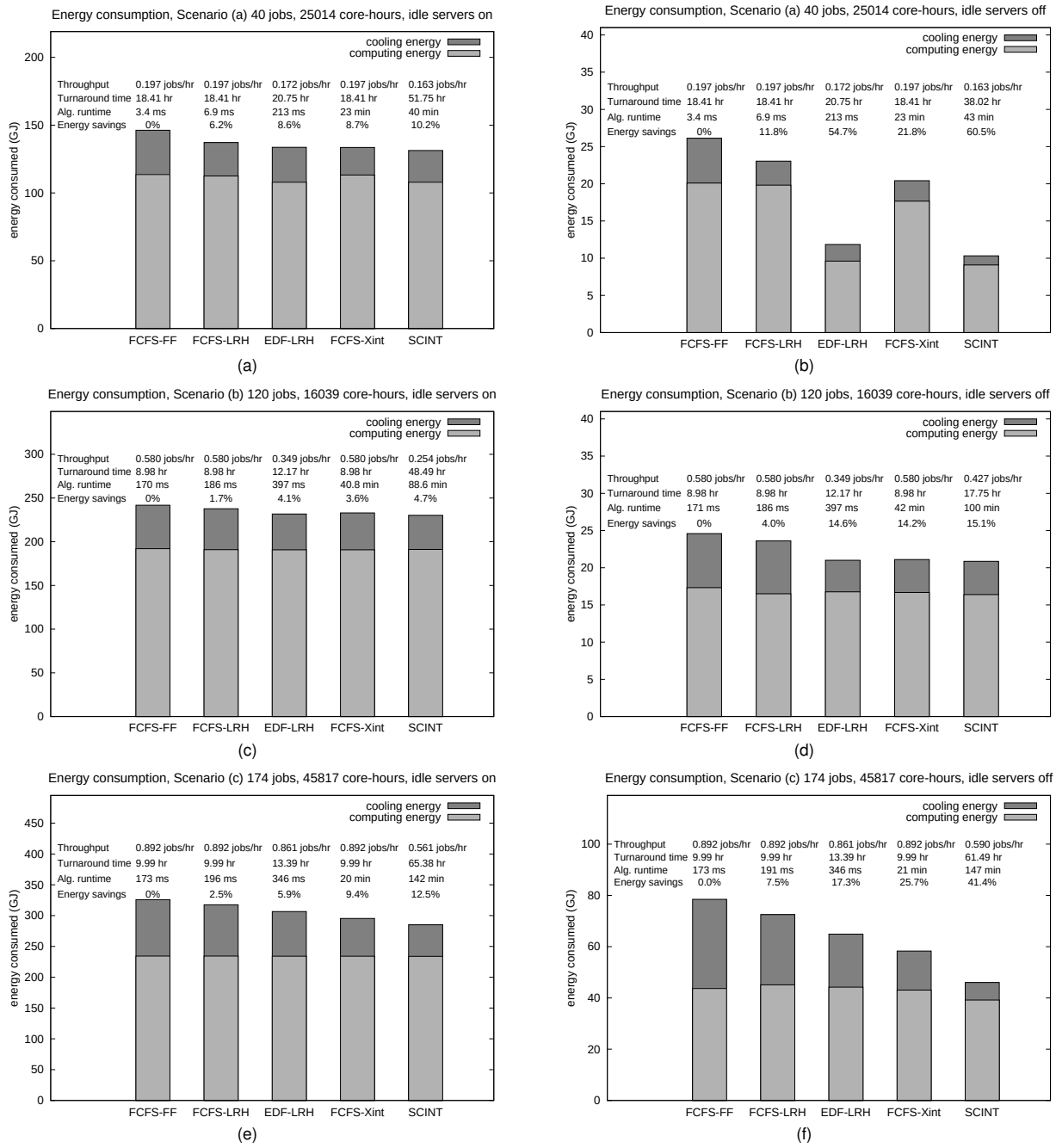

Fig. 8. Energy comparison of the simulated schemes for the three scenarios. The plots correspond in respective positions to the plots of Figure 7.

policy used in the data center, which enables job execution as soon as they arrive if the queue is empty and the data center is lightly loaded. In the "idle-on" case (Figure 8a), the total energy consumption using SCINT, EDF-LRH, FCFS-Backfill-FF, FCFS-Backfill-LRH and XInt placement is 131.3 GJ, 133.9 GJ, 146.2 GJ, 139.3 GJ and 139.3 GJ, respectively. However, in the "idle-off" case, the energy consumption reduces to 10.3,GJ, 11.8 GJ, 26.1 GJ, 24.4 GJ, and 24.6 GJ, respectively. Notice that the savings exceed 80% for any approach. The savings is achieved by: (i) the

reduction in the computing energy (since the idle servers' power usage is removed), and (ii) the reduction in the cooling requirement (since there are no heat recirculation from the idle servers). The throughput and turnaround time for all the variants of FCFS are same. However, these are compromised by SCINT and EDF-LRH. Figure 8 shows the results. The following subsections describe the aforementioned compromise for different data center utilization.

7.5.1. *Performance of FCFS-Backfill-XInt*

In the "idle-on" case, FCFS-Backfill-XInt placement leads to the energy savings ranging from 3.6% to 9.4%, when the idle servers are on, and from 5.8% to 25.7% when the idle servers are turned off (Figure 8). XInt places jobs such that the heat re-circulation is minimized, thereby requiring reduced cooling provisioning (Section 3.3). Results show that when idle servers are kept on, FCFS-Backfill-FF requires 29% of computing energy for cooling, whereas FCFS-Backfill-XInt placement requires 23% of computing energy for cooling.

However, in the "idle-off" case, FCFS-Backfill-XInt placement requires 13% of the computing energy for cooling, whereas FCFS-Backfill-FF requires 23% of the computing energy for cooling. Since the idle servers are turned off, the recirculation effect of the idle servers are removed, leading to the reduction in the cooling requirements for both the XInt placement and the first-fit placement.

7.5.2. *Performance of FCFS-Backfill-LRH*

In the "idle-on" case, FCFS-Backfill-LRH placement leads to the energy savings of 6.9 GJ (4.7%) over FCFS-Backfill-FF (Figure 8a), the same with FCFS-Backfill-XInt. This is because LRH is an approximation to the XInt algorithm. However, when the idle servers turn-off, the percentage of energy savings increases, mainly because LRH chooses the servers with the minimum cumulative heat recirculation. The advantage of XInt over LRH becomes bigger when the workload increases (Figures 8c, 8e).

In the "idle-off" case, FCFS-Backfill-XInt placement requires 13% of the computing energy for cooling, whereas FCFS-Backfill-FF requires 23% of the computing energy for cooling. Since the idle servers are turned off, the re-circulation effect of the idle servers are removed, leading to the reduction in the cooling requirements for both the XInt placement and the first-fit placement.

7.5.3. *Performance of SCINT*

In the "idle-on" case, SCINT achieves an energy savings of 14.9 GJ (10%) over FCFS-Backfill-FF (Figure 8a). This savings is achieved by SCINT through – i) the reduction in the peak utilization by temporal job distribution, and ii) the reduction in the peak inlet temperature by re-circulation minimized spatial job allocation (Section 3.3), similar to XInt. The temporal job distribution leaves more idle servers in the data center at an instance so that the lower power servers can be utilized to save computing energy, and re-circulation minimized placement can be more effective to reduce cooling requirement. The distribution of data center load over time can be verified in Figure 10.

In the "idle-off" case, the total energy consumption for SCINT over FCFS-Backfill-FF is 2.5 GJ (10%) (Figure 8, second column). SCINT non-intuitively distributes job over time thereby reducing the data center idle periods (Figures 7b, 7d, 7f). This job distribution however leads to the reduction in the peak utilization. As a result, the total amount of idle servers accumulated over time remains same. Therefore, the savings in the computing energy would not be affected even if the jobs are distributed.

Overall, in the "idle-on" case, only 21% of the computing energy needs to be expended for cooling in case of SCINT, and 25% in the case of FCFS-Backfill-LRH and FCFS-Backfill-XInt, whereas FCFS-Backfill-FF requires 29% of the computing power to be expended for cooling. In the "idle-off" case, SCINT requires 15% of the computing energy for cooling, while similar percentages hold for FCFS-Backfill-LRH and FCFS-Backfill-XInt, whereas FCFS-Backfill-FF requires 23% of the computing energy.

7.5.4. *Performance of EDF-LRH*

The performance of EDF-LRH is a hybrid between SCINT and FCFS-Backfill-LRH. At low workloads, the algorithm can easily spread the load over time (Figures 7(a) and 7(b)), thus achieving the energy-saving benefits that SCINT does (Figures 8a, 8b). Higher workloads, in synergistic effect with the online nature, prohibit the algorithm to approximate SCINT; the algorithm's performance is similar to that of FCFS-Backfill-LRH (Figures 8c through 8f).

7.6. *Processing Complexity of the algorithms*

SCINT is an omniscient scheduling algorithm that tries to schedule the jobs in time and space in order to reduce energy consumption. FCFS-Backfill-XInt placement assumes the FCFS-backfill scheduling in time (an instantaneous process) and makes the only decision of placing the jobs through XInt algorithm. Similar to SCINT, XInt also uses GA to find a 2D (*server* $\times$ *job*) job allocation results. However, unlike XInt, SCINT obtains a 3D (*server* $\times$ *job* $\times$ *time*) allocation matrix. Thus the search space of SCINT is much larger than the search space of FCFS-Backfill-XInt placement. Hence SCINT has to have a larger population size and larger number of generations than XInt to make a fair comparison (especially since the mutation and recombination processes are the same). The time complexity of GA based algorithms largely depend on the population size and the stopping condition. In our implementation the stopping condition was given as a bound on the maximum number of generations. The GA in SCINT was run for 300 generations each generation having a population size of 1000 individuals. The GA for XInt was run for 150 generations each generation having a population size of 400 individuals.

The algorithms were implemented on MATLAB 2007b running on a system with E8200 Core(TM)2 Duo CPU @ 2.66 GHz with 3.25 GB of RAM. The time taken by SCINT to come up with a scheduling decision is about 2.5 hours at maximum (Figure 8f). It is interesting to state that this version of MATLAB reached 100% core utilization, but

used only one core. FCFS-Backfill-XInt placement is run any time a new job arrives and a placement decision needs to be made. In the worst case scenario for our experiment (a maximum of 18 jobs arriving at a time) XInt required 42 minutes on the same machine environment (Figure 8d).

Algorithms based on LRH approach are considerably faster. FCFS-Backfill-LRH takes about the same time to run as first-fit, while EDF-LRH takes about twice that time.

7.7. *Results Summary*

To summarize the results, SCINT can achieve much higher energy savings when there is ample time for it to run. Given that the energy savings is significant, SCINT can be used as an off-line batch scheduling algorithm for long running jobs (e.g. scientific applications, high performance computing applications) where, the time to compute the schedule is insignificant in terms of jobs' execution times. For a more on-line requirement (short running jobs, e.g. web-services), if high-end equipment are not available to run SCINT faster (in seconds or minutes), FCFS-Backfill-XInt placement can be used as a reasonable alternative. Lastly in cases where the overhead of the scheduler must be extremely small, FCFS-Backfill-LRH or EDF-LRH can be used which have considerable savings over FCFS-Backfill-FF and also have extremely low runtimes.

8. Conclusions

In this paper, we investigated spatio-temporal thermal-aware job scheduling in order to improve the temperature distribution within data centers, and consequently the energy efficiency and the reliability of a data center's operation. We analytically expressed the effect of job scheduling on the data center energy needs and formulated a non-linear optimization problem. We proposed both off-line (SCINT) and on-line heuristics for the problem. The online heuristics include the variants of FCFS-backfill scheduling with XInt (FCFS-Backfill-XInt) and *least recirculated heat* (FCFS-Backfill-LRH) placement, and *earliest deadline first* with least recirculated heat (EDF-LRH) placement.

Simulation results show that there are considerable energy savings when the proposed algorithms are used. SCINT, being offline, distributes the job execution over time while meeting the user-estimated job execution times. This reduces the peak data center utilization leaving more choices for job assignment. Results show that SCINT can saves up to 40% energy, in certain scenarios, compared to FCFS-Backfill-FF, FCFS with Back-filling and first-fit placement. However SCINT is an offline algorithm requiring minutes to hours of operation depending on the average data center utilization. The online EDF-LRH requires only milliseconds of operation approximating the behavior of SCINT (by spreading the jobs in time) for low data center utilization. The behavior (and consequently the energy-savings) of EDF-LRH however degrades to that of FCFS-Backfill-LRH for high data center utilization since the online job spreading becomes more and more similar to FCFS-backfill with increase in utilization. Thus, SCINT is more suitable as an off-

line batch scheduling algorithm for long running jobs where the time to compute the schedule is insignificant in terms of jobs' execution times. EDF-LRH is a reasonable on-line alternative of SCINT for short running jobs with mostly low data center utilization, e.g. web-based applications.

8.1. *Future Work*

The paper assessed the energy savings of using thermal-aware scheduling and optionally turning the idle servers off; the results showed considerable energy reduction. The results on turning idle servers off were based on the assumption that the heat recirculation does not significantly alter with turning servers off, which is left as future work.

Further, the paper investigated thermal-aware spatio-temporal job scheduling for HPC data centers. In such data centers, submitted jobs require tens or hundreds of processors (cores) and take hours or days to finish. In future, the formulation needs to be extensible to other types of data centers, e.g. transaction-oriented data centers, and new forms of equipment, e.g. chiller doors. Extending this work to transaction-oriented data centers is non-trivial, as job scheduling has different requirements. In transaction-oriented data centers, jobs do not specify any estimates. Moreover, a thermally optimal placement of a single job has infinitesimal impact on the overall energy efficiency. Also, computing the optimal placement for each separate job (ie. transaction request) is impractical. For transaction-oriented data centers, work in progress is considering the jobs altogether as a single perpetual job that alters size in time, and compute the spatial scheduling of that "aggregate" job.

References

[1] J. Moore, J. Chase, and P. Ranganathan, "Weatherman: Automated, online and predictive thermal mapping and management for data centers," in *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*, June 2006, pp. 155–164.

[2] 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, pp. 61–75.

[3] 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 Transactions on Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1458–1472, 2008.

[4] 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: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*. New York, NY, USA: ACM Press, 2006, pp. 106–116.

[5] L. Ramos and R. Bianchini, "C-oracle: Predictive thermal management for data centers," in *IEEE 14th International Symposium on High Performance Computer Architecture (HPCA2008).*, Feb. 2008, pp. 111–122.

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

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

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

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

[10] Intel Corp., "Dual-Core Intel Xeon Processor LV and ULV Datasheet," Sept. 2006.

[11] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level power management for dense blade servers," in *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 66–77.

[12] R. Sullivan and K. G. Brill, "Cooling techniques that meet "24 by forever" demands of your data center," Uptime Institute, Inc., Tech. Rep., Jan. 2006.

[13] T. Brunschwiler, B. Smith, E. Ruetscheo, and B. Michel, "Toward zero-emission data centers through direct reuse of thermal energy," *IBM Journal of Research and Development*, vol. 53, no. 3, pp. 11:1–11:13, 2009.

[14] 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 *Proceedings of the American Institute of Aeronautics and Astronautics (AIAA)*, 2002, p. 3091.
[15] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Backfilling using system-generated predictions rather than user runtime estimates," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 18, no. 6, pp. 789–803, June 2007.
[16] L. A. Barroso and U. Hölzle, "The case of energy-proportional computing," *IEEE Computer*, vol. 40, no. 12, Dec. 2007.
[17] M. R. Gary and D. S. Johnson, *A Guide to the Theory of NP-Completeness*. Freeman, 1979.
[18] J. L. W. Jennifer Burge, Partha Ranganathan, "Cost-aware scheduling for heterogeneous enterprise machines," in *Workshop on Green Computing (GreenCom), Proceedings of the IEEE Cluster conference*, Sept. 2007.
[19] S. R. LaPlante, N. Aubry, L. Rosa, P. Levesque, B. S. Aboumrad, D. Porter, C. Cavanaugh, and J. Johnston, "Liquid cooling of a high density computer cluster," [online], 2006, http://www.electronics-cooling.com/articles/2006/2006_nov_a1.php.
[20] 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.
[21] Q. Tang, T. Mukherjee, S. K. S. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters," in *Int'l Conf. Intelligent Sensing & Info. Proc. (ICISIP2006)*, Dec 2006, pp. 203–208.
[22] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. Springer-Verlag New York, Inc., 1994.
[23] u. Flomerics Ltd, "Flovent version 2.1," Hampton Court, Surrey, KT8 9HH, England, 1999. [Online]. Available: http://www.flomerics.com/
[24] Cyber Switching, "DUALCOM user manual," [online], http://www.cyberswitching.com/pdf/DualcomManual.pdf.
[25] T. Mukherjee, G. Varsamopoulos, S. K. S. Gupta, and S. Rungta, "Measurement-based power profiling of data center equipment," in *Proc. IEEE Conference on Clustered and Grid Computing (Cluster 2007), Workshop on Green Computing (GreenCom'07)*, Austin, TX, Sept. 2007.
[26] "Standard Performance Evaluation Corporation – CINT2000 Result: Dell PowerEdge 1955." [Online]. Available: http://www.spec.org/cpu2000/results/res2006q3/cpu2000-20060626-06297.html
[27] "Standard Performance Evaluation Corporation – CINT2000 Result: Dell PowerEdge 1855." [Online]. Available: http://www.spec.org/cpu2000/results/res2005q3/cpu2000-20050902-04544.html