Thermal Prediction for Efficient Energy Management of Clouds Using Machine Learning

Shashikant Ilager, Kotagiri Ramamohanarao, and Rajkumar Buyya, *Fellow, IEEE*

Abstract—Thermal management in the hyper-scale cloud data centers is a critical problem. Increased host temperature creates hotspots which significantly increases cooling cost and affects reliability. Accurate prediction of host temperature is crucial for managing the resources effectively. Temperature estimation is a non-trivial problem due to thermal variations in the data center. Existing solutions for temperature estimation are inefficient due to their computational complexity and lack of accurate prediction. However, data-driven machine learning methods for temperature prediction is a promising approach. In this regard, we collect and study data from a private cloud and show the presence of thermal variations. We investigate several machine learning models to accurately predict the host temperature. Specifically, we propose a gradient boosting machine learning model for temperature prediction. The experiment results show that our model accurately predicts the temperature with the average RMSE value of 0.05 or an average prediction error of 2.38 °C, which is 6 °C less as compared to an existing theoretical model. In addition, we propose a dynamic scheduling algorithm to minimize the peak temperature of hosts. The results show that our algorithm reduces the peak temperature by 6.5 °C and consumes 34.5 percent less energy as compared to the baseline algorithm.

Index Terms—Cloud computing, machine learning, energy efficiency in a data center, datacenter cooling, hotspots

1 Introduction

The transition from ownership-based on-premise IT infrastructure to subscription-based Cloud has been tremendous in the past decade due to the vast advantages that cloud computing offers [1]. This rapid proliferation of cloud has resulted in a massive number of hyper-scale data centers that generate an exorbitant amount of heat and consume a large amount of electrical energy. According to [2], around 2 percent of global electricity is spent on data centers, and almost 50 percent of this energy is spent on cooling systems [3].

Modern cloud data centers' rack-mounted servers can consume up to 1,000 watts of power each and attain peak temperature as high as 100 °C [4]. The power consumed by a host is dissipated as heat to the ambient environment, and the cooling system is equipped to remove this heat and keep the host's temperature below the threshold. Increased host temperature is a bottleneck for the normal operation of a data center as it escalates the cooling cost. It also creates hotspots that severely affect the reliability of the system due to cascading failures caused by silicon component damage. The report from Uptime Institute [5] shows that the failure rate of equipment doubles for every 10 °C increase above 21 °C. Hence, thermal management becomes a crucial process inside the data center Resource Management System (RMS).

Therefore, to minimize the risk of peak temperature repercussions, and reduce a significant amount of energy consumption, ideally, we need accurate predictions of thermal dissipation and power consumption of hosts based on workload level. In addition, a scheduler that efficiently schedules the workloads with these predictions using certain scheduling policies. However, accurate prediction of a host temperature in a steady-state data center is a non-trivial problem [6], [7]. This is extremely challenging due to complex and discrepant thermal behavior associated with computing and cooling systems. Such variations in a data center are usually enforced by CPU frequency throttling mechanisms guided by Thermal Design Power (TDP), attributes associated with hosts such as its physical location, distance from the cooling source, and also thermodynamic effects like heat recirculation [6], [7]. Hence, the estimation of the host temperature in the presence of such discrepancies is vital to efficient thermal management. Sensors are deployed on both the CPU and rack level to sense the CPU and ambient temperature, respectively. These sensors are useful to read the current thermal status. However, predicting future temperature based on the change in workload level is equally necessary for critically important RMS tasks such as resource provisioning, scheduling, and setting the cooling system parameters.

Existing approaches to predict the temperature are inaccurate, complex, or computationally expensive. The widely used theoretical analytical models [6], [7], [8], [9], [10] that are built based on mathematical relations between different cyber-physical components lack the scalability and accurate prediction of the actual temperature. In addition, theoretical models fail to consider several variables that contribute towards temperature behavior and they need to be changed for different data centers. Computational Fluid Dynamics (CFD) models are also predominantly used [11], [12] for

• *The authors are with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia. E-mail: shashikant.ilager@gmail.com, {kotagiri, rbuyya}@unimelb.edu.au.*

Manuscript received 18 June 2019; revised 6 Nov. 2020; accepted 12 Nov. 2020. Date of publication 26 Nov. 2020; date of current version 11 Dec. 2020. (Corresponding author: Shashikant Ilager.) Recommended for acceptance by K. W Cameron. Digital Object Identifier no. 10.1109/TPDS.2020.3040800

accurate predictions, but their high complexity requires a large number of computing cycles. Building these CFD models and executing them can take hours or days, based on individual data center complexity [13]. The CFD models are useful in initial design and calibration of data center layout and cooling settings, however, it is infeasible for the realtime tasks (e.g., scheduling in large scale clouds) that are dynamic and require quick online decisions. Moreover, CFD simulation requires both computational (e.g, the layout of the Data Center, open tiles) and physical parameters, and changes to these parameters need expensive retraining of the models [14]. However, our approach is fast and cost-effective as it solely relies on the physical sensor data that are readily available on any rack-mounted servers and implicitly captures variations. Hence, data-driven methods using machine learning techniques is a promising approach to predict the host temperature quickly and accurately.

Machine learning (ML) techniques have become pervasive in modern digital society mainly in computer vision and natural language processing applications. With the advancement in machine learning algorithms and the availability of sophisticated tools, applying these ML techniques to optimize large scale computing systems is a propitious avenue [15], [16], [17], [18]. Recently, Google has reported a list of their efforts in this direction [19], where they optimize several of their large scale computing systems using ML to reduce cost, energy and increase the performance. Data-driven temperature predictions are highly suitable as they are built from actual measurements and they capture the important variations that are induced by different factors in data center environments. Furthermore, recent works have explored ML techniques to predict the data center host temperature [6], [20]. However, these works are applied to HPC data centers or similar infrastructure that relies on both application and physical level features to train the models. In addition, they are application-specific temperature estimations. Nevertheless, the presence of the virtualization layer in Infrastructure clouds prohibits this application-specific approach due to an isolated execution environment provided to users. Moreover, getting access to the application features is impractical in clouds because of privacy and security agreements between users and cloud providers. Consequently, we present a host temperature prediction model that completely relies on features that can be directly accessed from physical hosts and independent of the application counters.

In this regard, we collect and study data from our University's private research cloud. We propose a data-driven approach to build temperature prediction models based on this collected data. We use this data to build the ML-based models that can be used to predict the temperature of hosts during runtime. Accordingly, we investigated several ML algorithms including variants of regression models, a neural network model namely Multilayer Perceptron (MLP), and ensemble learning models. Based on the experimental results, the ensemble-based learning, gradient boosting method, specifically, XGBoost [21] is chosen for temperature prediction. The proposed prediction model has high accuracy with an average prediction error of 2.5 °C and Root Mean Square Error (RMSE) of 0.05. Furthermore, guided by these prediction models, we propose a dynamic scheduling algorithm to minimize the peak temperature of hosts in a data center. The scheduling algorithm is evaluated based on real-world workload traces and it is capable of circumventing potential hotspots and significantly reduces the total energy consumption of a data center. The results have demonstrated the feasibility of our proposed prediction models and scheduling algorithm in data center RMS.

In summary, the key contributions of our work are:

- We collect physical-host level measurements from a real-world data center and show the thermal and energy consumption variations between hosts under similar resource consumption and cooling settings.
- We build machine learning-based temperature prediction models using fine-grained measurements from the collected data.
- We show the accuracy and the feasibility of proposed prediction models with extensive empirical evaluation.
- We propose a dynamic workload scheduling algorithm guided by the prediction methods to reduce the peak temperature of the data center that minimizes the total energy consumption under rigid thermal constraints.

The remainder of the paper is organized as follows: The motivations for this work and thermal implications in the cloud are explained in Section 2. Section 3 proposes a thermal prediction framework and explores different ML algorithms. Section 4 describes the gradient boosting based prediction model. The feasibility of the prediction model is evaluated against a theoretical model in Section 5. Section 6 presents a dynamic scheduling algorithm. The analysis of scheduling algorithm results is done in Section 7 and the feature set analysis is described in Section 8. The relevant literature for this work is discussed in Section 9. Finally, Section 10 concludes the paper and also points out future research directions.

2 Motivation: Intricacies in Cloud Data Centers' Thermal Management

Thermal management is a critical component in cloud data center operations. The presence of multi-tenant users and their heterogeneous workloads exhibit non-coherent behavior with respect to the thermal and power consumption of hosts in a cloud data center. Reducing even one degree of temperature in cooling saves millions of dollars over the year in large scale data centers [17]. In addition, most data centers and servers are already equipped with monitoring infrastructure, that has several sensors to read the workload, power, and thermal parameters. Using this data to predict the temperature is cost-effective and feasible. Thereby, to analyze the complex relationships between different parameters that influence the host temperature, we collected data from a private cloud and studied it for intrinsic information. This data includes resource usage and sensor data of power, thermal, and fan speed readings of hosts. The detailed information about the data and collection method is described in Section 3.2.

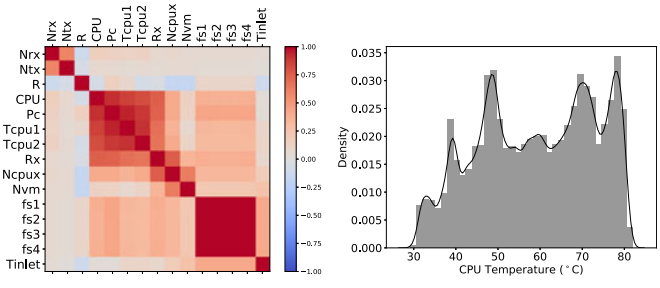
The correlation between different parameters (Table 1) and temperature distribution in the data center can be

TABLE 1
Definition of Features Collected

Features	Definition
CPU	CPU Load (%)
R	RAM- Random Access Memory (MB)
R_x	RAM in usage (MB)
N_{CPU}	Number of CPU cores
N_{CPUx}	Number of CPU cores in use
N_{Rx}	Network inbound traffic (Kbps)
N_{Tx}	Network outbound traffic (Kbps)
P_c	Power consumed by host (watts)
T_{cpu1}	CPU 1 temperature (°C)
T_{cpu2}	CPU 2 temperature (°C)
fs_1	fan1 speed (RPM)
fs_2	fan2 speed (RPM)
fs_3	fan3 speed (RPM)
fs_4	fan4 speed (RPM)
T_{in}	Inlet temperature (°C)
N_{vm}	Number of VMs running on host

observed in Figs. 1a and 1b. These figures are drawn from the data recorded on 75 hosts over a 90 days period. The logging interval was 10 minutes (i.e., $75 \times 90 \times 24 \times 6$ records). The correlation plot in Fig. 1a is based on the standard pairwise Pearson correlation coefficient represented as a heat map. Here, the correlation value ranges from -1 to 1, where the value is close to 1 for highly correlated features, 0 for no correlation, and -1 for the negative correlation. For better illustration, the values are represented as color shades as shown in the figure. In addition, the correlation matrix is clustered based on pairwise euclidean distance to enhance interpretability. It is evident that the CPU temperature of a host is highly influenced by power consumption and CPU load. However, factors like memory usage and machine fan speeds also have some degree of interdependence with it. Additionally, inlet temperature has a positive correlation with fan speeds and the number of VMs running on a host.

The high number of hosts operating at a peak CPU temperature can be observed from Fig. 1b. The figure represents a histogram of the temperature distribution of all hosts. Thereby each bin on the x axis represents a quantized CPU temperature and the y axis the corresponding probability density value. CPU temperature of hosts can reach more than 80 °C and the occurrence of such conditions are numerous which is evidenced by high-density value on the y axis for the respective bin. In addition, hosts exhibit inconsistent



(a) Correlation between all features (b) CPU temperature distribution

Fig. 1. Feature set correlation and temperature distribution.

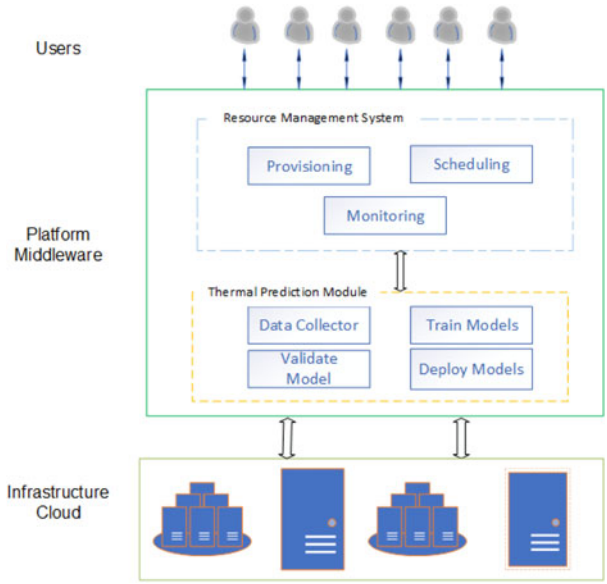


Fig. 2. System model.

thermal behavior based on several factors. This non-linear behavior of hosts presents a severe challenge in temperature estimation. A single theoretical mathematical model, applied even for homogeneous nodes, fails to accurately predict the temperature. Two homogeneous nodes at a similar CPU load observe different CPU temperatures. For instance, at a CPU load of 50 percent of the different hosts in our data set, CPU temperature varies up to 14 °C. Furthermore, with similar cooling settings, inlet temperature also varies up to 9 °C between hosts. These temperature variations are caused by factors like physical attributes such as the host's location, thermodynamic effects, heat recirculation, and thermal throttling mechanisms induced by the operating system based on workload behaviors [6]. Therefore, a temperature estimation model should consider the non-linear composite relationship between hosts.

Motivated by these factors, we try to rely on data-driven prediction approaches compared to existing rigid analytical and expensive CFD based methods. We use the collected data to build the prediction models to accurately estimate the host temperature. Furthermore, guided by these prediction models, we propose a simple dynamic scheduling algorithm to minimize the peak temperature in the data center.

3 System Model and Data-Driven Temperature Prediction

In this section, we describe the system model and discuss methods and approaches for cloud data center temperature prediction. We use these methods to further optimize our prediction model in Section 4.

3.1 System Model

A system model for predictive thermal management in the cloud data center is shown in Fig. 2. A Resource Management System (RMS) interacts with both, the users and the thermal prediction module, to efficiently manage the underlying resources of the cloud infrastructure. The prediction module consists of four main components, i.e., data

TABLE 2
Description of the Feature Set Variations in the Dataset (Aggregated From All the Hosts)

	$CPU(\%)$	R_x	N_{Rx}	N_{Tx}	N_{vm}	N_{CPUx}	P_c	fs_2	fs_1	fs_3	fs_4	T_{cpu1}	T_{cpu2}	T_{in}
Min	0	3974	0	0	0	0	55.86	5636	5686	5688	5645	29.14	25.46	13.33
Max	64.74	514614	583123.08	463888.76	21	101	380.53	13469	13524	13468	13454	82	75.96	18.05
Mean	18.09	307384.48	2849.00	1354.164	9	54	222.73	9484	9501	9490	9480	59.50	50.78	25.75

collector, training the suitable model, validating the performance of the model, and finally deploying it for runtime usage. RMS in a data center can use these deployed models to efficiently manage the resources and reduce the cost. The important elements of the framework are discussed in the following subsections.

3.2 Data Collection

An ML-based prediction model is as good as the data it has been used to train. In the data center domain, training data can include application and physical level features to train the model [6]. The application features include instruction count, number of CPU cycles, cache metrics (read, write and miss), etc. Accordingly, physical features include host-level resource usage (CPU, RAM, I/O, etc.) and several sensor readings (power, CPU temperature, fan speeds). Relying on both of these features is feasible in bare metal HPC data centers where administrators have exclusive access to the application and physical features. However, in the case of Infrastructure as Service (IaaS) clouds, resources are virtualized and provisioned as VMs or containers, thus, giving users exclusive isolated access to the application execution environment. The presence of a hypervisor or container-based virtualization in IaaS clouds restricts access to application-specific features. Moreover, a diverse set of users in the cloud have a different type of workloads exhibiting different application behaviors which impede cloud RMS to rely on application-specific features. As a consequence, to predict host temperature, the RMS is required to monitor fine-grained resource usage and physical features of the host system that can be directly accessed. In this regard, we show that this data is adequate to predict the host temperature accurately.

The Melbourne Research Cloud (MRC)[1] provides Virtual Machines (VM) to students and researchers. The representative data is collected from a subset of machines from MRC. This computing infrastructure provides computing facilities to students and researchers as a virtual machine (VM). We collect data from a subset of the total machines in this cloud. A brief summary of this data is presented in Table 3. It includes logs of 75 physical hosts having an average number of 650 VMs. The data is recorded for a period of 3 months and the log interval is set to 10 minutes. The total count of resources includes 9,600 CPU cores and 38,692 GB of memory. After data filtration and cleaning, the final dataset contains 984,712 tuples, each host approximately having around 13,000 tuples. Each tuple contains 16 features including resource and usage metrics, power, thermal, and fan speed sensors measurements. The details of these features are given in Table 1. As each host is equipped with two distinct CPUs, two temperature measurements are reported per machine. In addition, each system has four separate fans installed to provide cooling. The reason to collect data for an extended period is to capture all the dynamics and variations of parameters to train the model effectively. This is only possible when host resources have experienced different usage levels over time. A model built over such data allows accurate prediction in dynamic workload conditions. An overview of variations of all parameters is depicted in Table 2 (N_{CPU} and R are not included as they represent constant resource capacity).

To collect this data, we run a collectd[2] daemon on every host in the data center, which is a standard open-source application that collects system and application performance counters periodically through system interfaces such as IPMI and sensors. These metrics are accessed through network API's and stored in a centralized server in the CSV format. We used several bash and python scripts to pre-process the data. Specifically, python pandas[3] package to clean and sanitize the data. All invalid measurements (e.g., NaN) were removed. For the broader use of this data to the research community and for the sake of reproducibility, we will publish the data and scripts used in this work.

3.3 Prediction Algorithms

The choice of regression-based algorithms for our problem is natural since we aim to estimate the numerical output variable i.e., temperature. In the search for suitable prediction mechanisms, we have explored different ML algorithms including different regression techniques, such as Linear Regression (LR), Bayesian Regression (BR), Lasso Linear Regression (LLR), Stochastic Gradient Descent regression (SGD), an Artificial Neural Network (ANN) model called Multilayer Perceptron (MLP), and an ensemble learning technique called gradient boosting, specifically, eXtreme Gradient Boosting (XGBoost).

Since each host in our cluster has two CPUs that are jointly controlled by the same operating system (which may dynamically move workloads between them), we always regard the maximum of the respective two CPU temperature measurements as the systems' effective CPU temperature. We aim to build a model for each host to accurately capture its thermal behavior properties. For that reason, instead of solely predicting CPU temperature, we predict the host ambient temperature (T_{amb}) which is a combination of inlet temperature and CPU temperature [22]. The reason to consider ambient temperature instead of CPU temperature is manifold. First, by combining the inlet and CPU

1. https://docs.cloud.unimelb.edu.au/

2. https://collectd.org/
3. https://pandas.pydata.org/

TABLE 3
Private Cloud Data Collected for This Work

#Hosts	#VMs	Total CPU Cores	Total Memory	Collection Period	Collection Interval
75	650	9600	38692 GB	90 days	10 Minute

temperature, it is feasible to capture thermal variations that are induced by both the inlet and CPU temperature (cause of these variations are discussed in Section 2). Second, at a data center level, cooling settings knobs are adjusted based on host ambient temperature rather than individual CPU temperature [13]. In addition, resource management systems in the data center consider host-level ambient temperature as a threshold parameter whereas operating system level resource management techniques rely on CPU temperature.

Therefore, to build the prediction model for individual hosts, we parse the data set and partition it based on host IDs. For each individual host, the feature set consists of a variable number of tuples, with each tuple having these features (CPU, R, R_x, N_{CPU}, N_{CPUx}, N_{Rx}, N_{Tx}, N_{vm}, P_c, $fs_1 - fs_4$, T_{amb}). Note that, we have excluded inlet and CPU temperatures from the list, as we have combined these as ambient temperature (T_{amb}) which is our target prediction variable.

We used sci-kit learn package [23] to implement all the algorithms. For XGBoost, we used a standard python package[4] available on Github. The parameters for each of the algorithms are set to their default settings in our implementation. For MLP, it follows a standard 3 layers architecture, with the number of neurons at a hidden layer set to 5 and a single output neuron, and 'ReLu' as the activation function.

To avoid the overfitting of the models, we adopt k-fold cross-validation where the value of k is set to 10. Furthermore, to evaluate the goodness of fit for different models, we use the Root Mean Square Error (RMSE) metric which is a standard evaluation metric in regression-based problems [24]. The RMSE is defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(y_i - \hat{y_i}\right)^2}. \tag{1}$$

In Equation (1), y_i is the observed value, $\hat{y_i}$ is the predicted output variable, and n is the total number of predictions. The value of RMSE represents the standard deviation of the residuals or prediction errors. The prediction models attempt to minimize an expectation of loss, thus, lower RMSE values are preferred.

The performance of different algorithms is shown in Fig. 3. These results are an average of all the hosts' prediction model results. In Fig. 3, we can observe that XGBoost has a very low RMSE value, indicating that, the residuals or prediction errors are less and its predictions are more accurate. We observed that MLP has a high error value compared to other algorithms. In addition, different regression variants have performed almost similar to each other. As the gradient boosting method XGBoost results are promising, we focus more on this algorithm to explore it further, optimize, and adapt it for further scheduling as explained in Section 6.

4. https://github.com/dmlc/xgboost

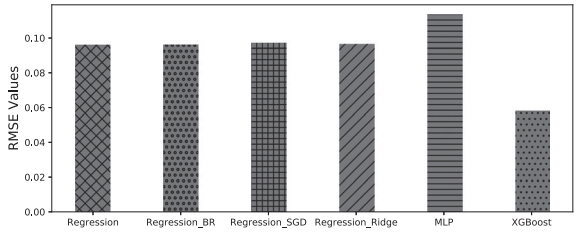


Fig. 3. Average prediction error between different models.

4 Learning With Extreme Gradient Boosting

Boosting is an ensemble-based machine learning method that builds strong learners based on weak learners. Gradient boosting is an ensemble of weak learners, usually decision trees. XGBoost (eXtreme Gradient Boosting) is a scalable, fast and efficient gradient boosting variant for tree boosting proposed by Chen *et al.* [21]. It incorporates many advanced techniques to increase the performance, such as parallelism, cache optimization with better data structure, and out of core computation using block compression and block sharing techniques which is essential to prevent the memory overflow in training large data sets on constrained resource environments. Accordingly, the impact of boosting techniques including XGBoost is evidenced by its dominant adoption in many Kaggle competitions and also in large scale production systems [25], [26], [27].

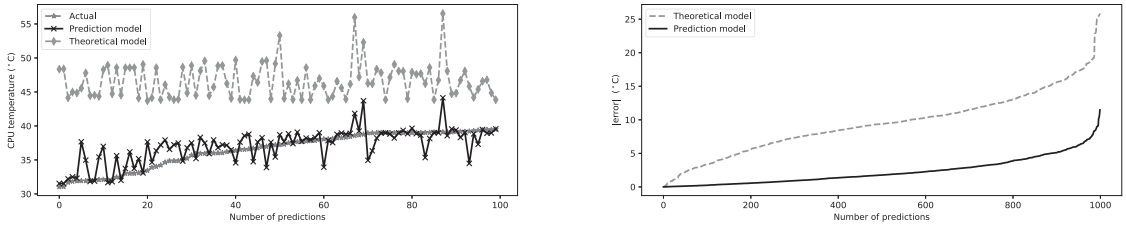
The XGBoost algorithm is an ensemble of K Classification or Regression Trees (CART) [21]. This can be used for both classification and regression purpose. The model is trained by using an additive strategy. For a dataset with n instances and m features, the ensemble model uses k additive functions to estimate the output. Here, x being a set of input features, $x = \{x_1, x_2, \ldots x_m\}$ and y is the target prediction variable

$$\hat{y_i} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \qquad f_k \in F. \tag{2}$$

In the Equation (2), F is space of all the regression trees, i. e, $F = \{f(x) = w_{q(x)}\}$, and $\left(q : \mathbb{R}^m \rightarrow T, w \in \mathbb{R}^T\right)$. Here, q is the structure of each tree which maps to corresponding leaf index. T represents total number of leaves in the tree. each f_k represents an independent tree with structure q and leaf weights w. To learn the set of functions used in the model, XGBoost minimizes the following regularized objective

$$\zeta(\phi) = \sum_i l(\hat{y_i}, y_i) + \sum_k \Omega(f_k), \quad where \quad \Omega(f) = \gamma T = 1\frac{2}{\lambda} \parallel w \parallel^2 . \tag{3}$$

In Equation (3), the first term l is the differentiable convex loss function that calculates the difference between predicted value $\hat{y_i}$, observed value y_i. Ω penalizes the complexity of the model to control overfitting. Thereby, T is the number of nodes in the tree and w is assigned values for each leaf node of the tree. This regularized objective



(a) Temperature estimation compared to actual values (b) Rank order of prediction errors

Fig. 4. Comparison of prediction and theoretical model.

function attempts to select a model based on simple predictive functions.

We use the grid search technique to find the optimal parameters to further enhance the performance of the model. Here, the γ parameter is used to decide the minimum loss reduction required to make a further partition on a leaf node of the tree. Subsample ratio decides the amount of sampling selected from training data to grow the trees. Accordingly, the optimal values for γ are 0.5, the learning rate is 0.1, maximum depth of the tree is 4, minimum child weight is 4, and the subsample ratio is 1, and rest of the parameters are set to default. With these settings, the best RMSE value achieved is 0.05. It is important to note that the prediction based temperature estimation is feasible for any data center given the historical data collected from the individual data center.

5 Evaluating the Prediction Model With Theoretical Model

To evaluate the feasibility of our temperature prediction models, we compare the prediction results to extensively used theoretical analytical model [7], [8], [9]. Here, the temperature estimation is based on the RC model which is formulated from analytical methods. The temperature of a host (T) is calculated based on the following equation:

$$T = PR + T_{in} + (T_{initial} - PR - T_{in}) \times e^{-\frac{t}{RC}}. \quad (4)$$

In Equation (4), P is the dynamic power of host, R and C are thermal resistance (k/w) and heat capacity (j/k) of the host respectively. $T_{initial}$ is the initial temperature of the CPU. Since analytical models estimate CPU temperature, we also predict CPU temperature to compare the results instead of ambient temperature.

To compare the results, we randomly select 1,000 tuples from our whole dataset and analyze the result between prediction and theoretical models. For the theoretical model, the value of P and T_{in} are directly used from our test data set. The value of thermal resistance (R) and heat capacity (C) is set as 0.34 K/w and 340 J/K respectively and $T_{initital}$ is set to 318 K [9].

The performance of the two models in temperature estimation can be observed in Fig. 4. For the sake of visibility, Fig. 4a includes 100 tuples of data. As the figure suggests, our proposed model based on XGBoost's estimation is very close to the actual values, whereas the theoretical model has a large variation from the actual values. Fig. 4b, represents a rank order of the absolute errors (from actual temperature) of two models in °C. The theoretical model deviates as far as 25 °C from the actual values. In this test, the average error of the theoretical model is 9.33 °C and our prediction model is 2.38 °C. These results reflect the feasibility of using prediction models over theoretical models for temperature estimation. It is important to note that, the prediction models need to be trained for different data centers separately with well-calibrated data that have enough data points to cover all temperature and load conditions in order to predict temperature accurately. Nevertheless, in the absence of such a facility, it is still feasible to use theoretical analytical models that rely on a minimum number of simple parameters.

6 Dynamic Scheduling Guided by Prediction Models

Applications of temperature predictions are numerous. It can be used to change the cooling settings such as supply air temperature to save the cooling cost [22]. It is also useful in identifying the thermal anomalies which increase the risk of failures and injects performance bottlenecks. Moreover, one foremost usage would be in a data center resource management system's tasks such as resource provisioning and scheduling.

With the given historical host's data, predictive models are trained and deployed for runtime inference. A scheduling algorithm invokes a deployed prediction model to accurately predict the host temperature. The input to the prediction model is a set of host features. In our model, the features can be easily collected from the host's onboard sensors. These features are accessed from the host's system interface through HTTP APIs. The complexity to retrieve this input feature set information is $O(1)$. The latency of this operation depends on the data center's local network capabilities. Moreover, the models need to be retrained only when changes are introduced to the data center environment, like, the addition of new hosts or change in the physical location of hosts. Considering the fact that such changes are not so frequent in a data center, the cost of building and using such predictive models in resource management tasks like scheduling is highly feasible.

In this regard, we propose dynamic scheduling of VMs in a cloud data center based on the temperature prediction model we have proposed. Here, we intend to reduce the peak temperature of the system while consolidating VMs on fewest hosts as possible for each scheduling interval which is a preferred way to reduce the energy in a cloud data center [28]. In this problem, n physical hosts in data center hosting m VMs at timestep t, the objective is to reduce the number of active hosts in a data center at $t+1$ by consolidating the VMs based on workload level. This

consolidation process inside the data center is critical and carried out regularly to reduce overall data center energy [29], [30]. This procedure mainly includes three steps. First, identifying under loaded hosts from which we can potentially migrate VMs and shut down the machine. Also finding overloaded hosts and migrate VMs from them to reduce the risk of Service Level Agreements (SLA) violation, here, SLA is providing requested resources to VMs without degrading their performance. Second, selecting VMs for migration from the over-and underloaded hosts identified in previous step, and finally, identifying new target hosts to schedule the selected VMs. The scheduling for consolidation process allows hosts to experience high load and potentially reach the threshold temperature which is useful in evaluating our prediction models effectively. Therefore, The objective of our problem is defined as follows:

$$\begin{aligned} \text{minimize} \quad & T^{peak} = \sum_{t=0}^{T} \sum_{j=1}^{m} \sum_{i=1}^{n} \delta_{ji}^{t} T_i^t \\ \text{subject to} \quad & u(h_i) \leq U_{max}, \\ & T_i^t < T_{red} \\ & \sum_{j=0}^{m} VM_{ji}(R_{cpu}, R_{mem}) \leq h_i(R_{cpu}, R_{mem}) \\ & \delta_{ji}^{t} = \{0, 1\} \\ & \sum_{i=1}^{n} \delta_{ji}^{t} = 1. \end{aligned} \tag{5}$$

The objective function in Equation (5) minimizes the peak temperature of the hosts while scheduling VMs dynamically in all the time steps $t = \{0, \ldots \quad T\}$. Here, list of VMs that are to be scheduled are represented with the index j where $j = \{1, \ldots \quad m\}$, and list of candidate hosts as i, where $i = \{1, \ldots \quad n\}$. The T_i^t indicates temperature of host i at time t. The constraints ensure that potential thermal and CPU thresholds are not violated due to increased workload allocation. They also assure the capacity constraints, i. e, a host is considered as suitable only if enough resources are available for VM (R_{cpu}, R_{mem}). Here, δ_{ji}^t is a binary with the value 1 if the VM_j is allocated to $host_i$ at time interval t, otherwise, 0. The summation of δ_{ji}^t is equal to 1, indicating that VM_j is allocated to at most 1 host at time t. The objective function in Equation (5) is executed at each scheduling interval to decide the target host for the VMs to be migrated. Finding an optimal solution for the above equation is an NP-hard problem and it is infeasible for on-line dynamic scheduling. Accordingly, to achieve the stated objective and provide a near-optimal approximate solution within a reasonable amount of time, we propose a simple Thermal-Aware heuristic Scheduling (TAS) algorithm that minimizes the peak temperature of data center hosts.

To dynamically consolidate the workloads (VMs) based on current usage level, our proposed greedy heuristic scheduling Algorithm 1 is executed for every scheduling interval. The input to the algorithm is a list of VMs that are needed to schedule. These are identified based on overload and underload condition. To identify overloaded hosts, we use CPU (U_{max}) and temperature threshold (T_{red}) together. In addition, if all the VMs from a host can be migrated to current active hosts, the host is considered as an underloaded host. The VMs that are to be migrated from overloaded hosts are selected based on their minimum migration time, which is the ratio between their memory usage and available bandwidth [28]. The output is scheduling maps representing target hosts for those VMs. For each VM to be migrated (line 2), Algorithm 1 tries to allocate a new target host from the active list. In this process, algorithm initializes necessary objects (lines 3-5) and the prediction model is invoked to predict the accurate temperature of a host (line 7). The VM is allocated to a host that has the lowest temperature among active hosts (lines 8-11). This ensures the reduction of peak temperature in the data center and also avoids potential hotspots resulting in lower cooling cost. Moreover, this algorithm also assures the constraints listed in Equation (5) are met (line 10), so that added workload will not create a potential hotspot by violating threshold temperature (T_{red}). In addition, resource requirements of VM (VM(R_x)) are satisfied, and the CPU utilization threshold is within the limit (U_{max}). If no suitable host is found in the process, a new idle or inactive host is allocated (line 16) from the available resource pool.

Algorithm 1. Thermal Aware Dynamic Scheduling to Minimize Peak Temperature

Input: $VMList$- List of VMs to be scheduled
Output: Scheduling Maps

```
1: for t ← 0 to T do
2:     for all vm in VMList do
3:         allocatedHost ← ∅
4:         hostList ← Get list of active hosts
5:         minTemperature ← maxValue
6:         for all host in hostList do
7:             T̂_i ← Predict temperature by invoking prediction
               model
8:             if (T̂_i < minTemperature) then
9:                 minTemperature ← T̂_i
10:                if (T̂_i < T_red and u(h_i) ≤ U_max and vm(R_x) <
                   host(R_x)) then
11:                    allocatedHost ← host
12:                end if
13:            end if
14:        end for
15:        if allocatedHost == ∅ then
16:            allocatedHost ← Get a new host from inactive hosts
               list
17:        end if
18:    end for
19: end for
```

The Algorithm 1 has a worst-case complexity of $\mathcal{O}(VN)$, which is a polynomial-time complexity. Here, $|V|$ is the number of VMs to be migrated, and $|N|$ is a number of hosts in a data center.

7 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithm coupled with our prediction model and compare and analyze the results with baseline algorithms.

TABLE 4
VM Configurations

Name	Core	RAM
VM1 (uom.general.1c4g)	1	4 GB
VM2 (uom.general.2c8g)	2	8 GB
VM3 (uom.general.4c16g)	4	16 GB
VM4 (uom.general.8c32g)	8	32 GB

7.1 Experimental Setup

We evaluated the proposed thermal aware dynamic scheduling algorithm through CloudSim toolkit [31]. We extended CloudSim to incorporate the thermal elements and implement Algorithm 1. We used a real-world dataset from Bitbrain [32], which has traces of resource consumption metrics of business-critical workload hosted on Bitbrain's infrastructure. This data includes logs of over 1,000 VMs workloads hosted on two types of machines. We have chosen this data set as it represents real-world cloud Infrastructure usage patterns and the metrics in this data set are similar to the features we have collected in our data set (Table 1). This is useful to construct precise input vectors for prediction models.

The total experiment period is set to 24 hours and the scheduling interval to 10 minutes, which is similar to our data collection interval. Note that, in the algorithm, prediction models are invoked in many places. The prediction is required to identify the host with the lowest temperature, to determine a host overloaded condition, and also to ensure thermal constraints by predicting their future time step temperature.

To depict the experiments in a real-world setting, we model host configurations similar to the hosts in our data center, i.e., DELL C6320 machines. This machine has an Intel Xeon E5-2600 processor with dual CPUs (32 cores each) and 512 GB RAM. The VMs are configured based on the VM flavours in our research cloud.[5] We choose four VM types from general flavors, configuration of these VMs are presented in Table 4. The number of hosts in the data center configuration is 75, similar to the number of hosts in our private cloud collected data, and the number of VMs is set to 750, which is the maximum number possible on these hosts based on their maximum resource requirements. The workload is generated to these VMs according to Bitbrain's dataset.

The CPU threshold (U_{max}) is set to 0.9. According to the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) [4] guidelines, the safe operable temperature threshold for data center hosts is in-between 95 to 105 °C. This threshold is a combined value of CPU temperature and inlet temperature together. Accordingly we set temperature threshold (T_{red}) to 105 °C.

The new target machines for VMs to be scheduled are found based on Algorithm 1. This requires predicting the temperature of hosts in the data center. If the $host_i$ temperature is predicted ($\hat{T}_i$) at the beginning of timestep $t+1$ then the input to prediction model is a single vector consisting of a set of features (CPU, P_c, $fs_1 - fs_4$, N_{CPU}, N_{CPUx}, R, R_x, N_{Rx}, N_{Tx}, N_{vm}) representing its resource and usage metrics along with the power and fan speed measurements. The resource usage metrics are easily gathered from host utilization levels based on its currently hosted VMs' workload level. To estimate the power $\hat{P}_i$, we use SPECpower benchmark [33], which provides accurate power consumption (in watts) for our modeled host (DELL C6320) based on CPU utilization. We estimate fan speeds from simple regression using remaining features to simplify the problem.

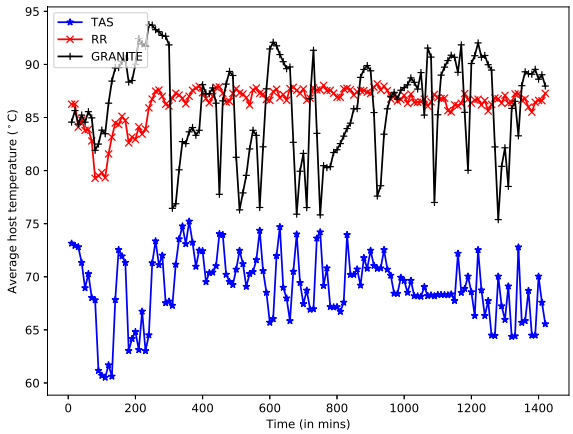


Fig. 5. Average temperature in each scheduling interval (total experiment time of 24 hours, with scheduling interval of 10 minute).

We export the trained models as serialized python objects and expose them to our scheduling algorithm by hosting on HTTP Flask application.[6] The CloudSim scheduling entities invoke the prediction model through REST APIs by giving feature vector and host ID as input, the HTTP application returns predicted temperature for the associated host.

7.2 Analysis of Results

We compare the results with two baseline algorithms as shown below.

- Round Robin (RR) - This algorithm tries to distribute the workload equally among all hosts by placing VMs on hosts in a circular fashion. The similar constraints are applied as in Algorithm 1. We show that the notion of equal distribution of workloads fails to minimize the peak temperature and thermal variations in a data center.
- GRANITE- This is a thermal-aware VM scheduling algorithm proposed in [34] that minimizes computing and cooling energy holistically. We choose this particular algorithm, because, similar to us, it also addresses the thermal-aware dynamic VM scheduling problem.

We use our prediction models to estimate the temperature in both RR and GRANITE algorithms. For GRANITE, the required parameters are set similar to their algorithm in [34] including overload and underload detection methods. The comparison of the average temperature from all hosts in each scheduling interval by all three algorithms is shown in Fig. 5. Our Thermal-Aware Scheduling (TAS) has the lowest average temperature compared to RR and GRANITE. The

5. https://docs.cloud.unimelb.edu.au/guides/allocations/

6. http://flask.pocoo.org

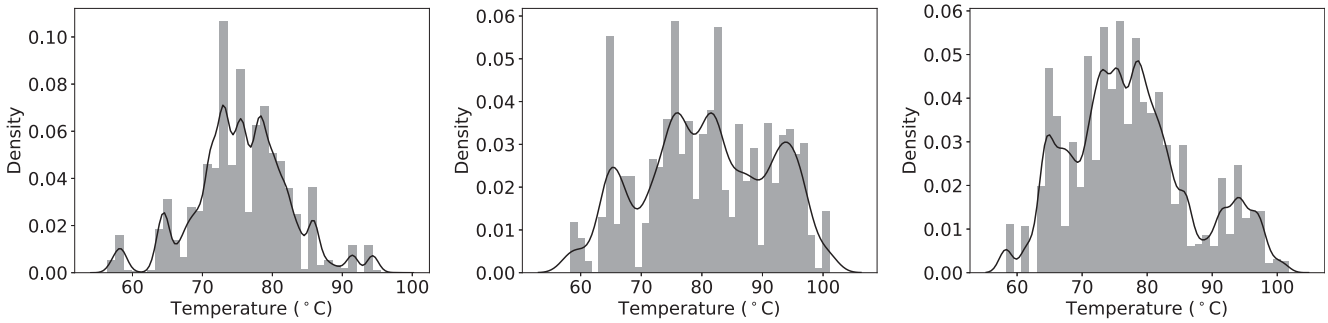


Fig. 6. Temperature distribution analysis due to scheduling (aggregated from all hosts in experimented period).

RR algorithms' equal workload distribution policy results in less variation in average temperature. However, this will not help to reduce the peak temperature in the data center irrespective of its intuitive equal distribution behavior as it doesn't consider the thermal behavior of individual hosts and its decisions are completely thermal agnostic. The GRANITE policy has a high average temperature and large variations between scheduling intervals due to its inherent dynamic threshold policies. To further analyze the distribution of temperature due to two scheduling approaches, we draw a histogram with Kernel Density Estimation (KDE) by collecting temperature data from all the hosts in each scheduling interval as shown in Fig. 6. Most of the hosts in the data center operate around 70 to 80 °C in TAS (Fig. 6a), well below the threshold due to its expected peak temperature minimizing objective. However, the RR approach results in more thermal variations with sustained high temperatures (Fig. 6b). The GRANITE also has significant distributions around the peak temperature (Fig. 6c). This temperature distribution is effectively summarized using the Cumulative Distribution Function (CDF) between three approaches (Fig. 7). As we can see in Fig. 7, TAS reaches the probability density value of 1 well below 100 °C, indicating most of the hosts operate in reduced temperature value. RR and GRANITE has a peak temperature of more than 100 °C with high cumulative probability. In addition, as depicted in Fig. 7, the average and standard deviation of temperature in TAS ($\mu = 75.65$, $\sigma = 6.82$) is lesser compared to the other two approaches ($\mu = 80.69$, $\sigma = 10.49$ for RR and $\mu = 77.36$, $\sigma = 9.34$ for Granite), this is also evidenced by Fig. 5.

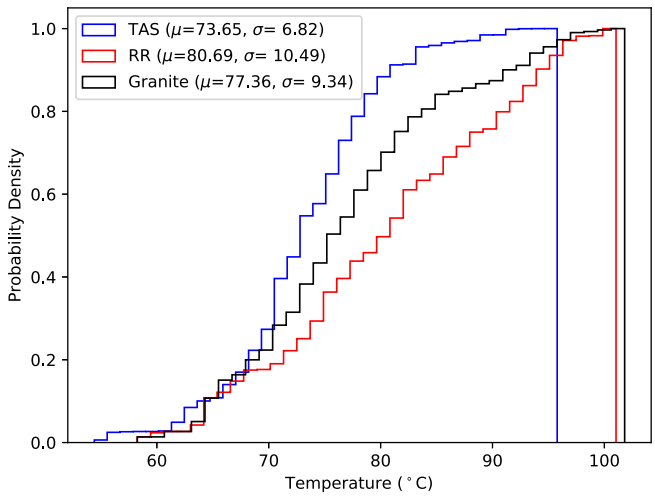


Fig. 7. CDF between TAS, RR, and GRANITE.

Further results of the experiments are depicted in Table 5. The total energy consumption by TAS, RR, and GRANITE is 172.20, 391.57, and 263.20 kWh, respectively (the total energy is a combination of cooling and computing energy calculated as in [10]). Therefore, RR and GRANITE have 56 and 34.5 percent more energy consumption than TAS, respectively. This is because RR and GRANITE distribute workload into more hosts resulting in a high number of active hosts. In this experimented period, RR and GRANITE had 18 and 11 average number of active hosts while the TAS algorithm resulted in 4 active hosts. Furthermore, although RR distributes workload among many hosts, its thermal agnostic nature had a peak temperature of 101.44 °C, GRANITE had peak temperature of 101.80 °C and TAS had attained a maximum of 95.5 °C during the experimentation period which is 6.5 °C lower than the latter approaches. This demonstrates that the accurate prediction of host temperature with an effective scheduling strategy can reduce the peak temperature and also save a significant amount of energy in the data center.

7.3 Evaluating Performance Overhead

It is important to estimate the overhead of dynamic scheduling caused due to migration and workload consolidation. In the context of scheduling in the cloud, the expected performance is usually defined using Service Level Agreements (SLAs). In our approach, the scheduling is at a higher VM level, hence, we represent the SLA metrics using the VM level features. In this regard, we consider the following metrics [28], [34]:

Number of VM Migrations. Virtual machines may experience degraded performance during migration. Hence, the

TABLE 5
Scheduling Results Compared With RR and GRANITE Algorithm

Algorithm	Peak Temperature (°C)	Total Energy (kwh)	Active Hosts
TAS	95	172.20	4
RR	101.44	391.57	18
GRANITE	101.81	263.20	11

number of migrations should be minimized to reduce the overhead and avoid SLA violations.

$SLA_{violation}$. Due to oversubscription and consolidation, hosts may reach full utilization level (100 percent), in such cases, the VMs on such host experiences degraded performance. This is expressed using SLA violation Time per Active Host (SLA_{TAH}) metric as shown in Fig. 9c. Furthermore, the consolidation of VMs comes with performance overhead caused due to live VM migration [35], this Performance Degradation due to Migration (PDM) is defined as in Equation (7)

$$SLA_{TAH} = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{max}}{T_{active}} \tag{6}$$

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{Aj} - C_{Rj}}{C_{Rj}} \tag{7}$$

$$SLA_{violation} = SLA_{TAH} \times PDM. \tag{8}$$

Here, N is total number of hosts, T_{max} is amount of time $Host_i$ has experienced 100 percent of utilization and T_{active} is total active time of $Host_i$. M is the total number of VMs. The C_{Aj} is the total amount of CPU capacity allocated and C_{Rj} is the total amount of CPU capacity requested by VM_j while in migration during its lifetime, this captures the under allocation of VMs during live migration. The overall SLA violation of cloud infrastructure ($SLA_{violation}$) can be defined by combining both SLA_{TAH} and PDM metrics as shown in Equation (8).

The results of overhead metrics for different algorithms are shown in Fig. 9. As shown in Fig. 9a, the number of migrations is 10,417 and 18,117 for GRANITE and TAS, respectively. The RR has zero migrations. It is expected as RR distributes workload equally among the required number of hosts from the initial step and is not concerned about dynamic optimizations in runtime. For the PDM metric (Fig. 9b), GRANITE and TAS have 0.0037 and 0.0064 percent, respectively. This is because to TAS has a higher number of migrations compared to GRANITE. As TAS continuously tries to minimize the peak temperature among active hosts based on workload level, it performs aggressive consolidation in each scheduling interval. However, the proactive approach of TAS trying to reduce the host peak of temperature also results in reduced CPU overload of hosts. This is evidenced as the TAS has a lower value of SLA_{TAH} metric (0.34 percent) compared to the GRANITE (0.53 percent). Furthermore, for the overall $SLA_{violation}$ metric (Fig. 9d), TAS has increased value (0.22×10^{-6}) compared to GRANITE (0.20×10^{-6}). This little increased value is due to the higher PDM value of TAS. However, TAS significantly outperforms both GRANITE and RR in reducing peak temperature and energy efficiency with this negligible overhead.

7.4 Dealing With False Predictions

In our scheduling experiments, we observed that a few of the temperature predictions have resulted in some large number which is beyond the boundaries of the expected value. A further close study into such cases has revealed that this happens with particularly three hosts which were almost idle in the data collection period of 3 months having a CPU load less than 1 percent, which means the models trained for these hosts have limited variations in their feature set. As the trained models did not have any instance close to the instance of prediction, prediction results in an extreme variant value. Such a false prediction in runtime results in an incorrect scheduling decision that affects the normal behavior of the system. In this regard, the scheduling process should consider such adverse edge cases. To tackle this problem, we set minimum and maximum bound for expected prediction value based on our observations in the dataset. For any prediction beyond these boundaries, we pass the input vector to all remaining hosts' models and take an average of predicted value as a final prediction value. In this way, we try to avoid the bias influenced by a particular host and also get a reasonably good prediction result. In the case of a huge number of hosts, subsets of hosts can be used for this.

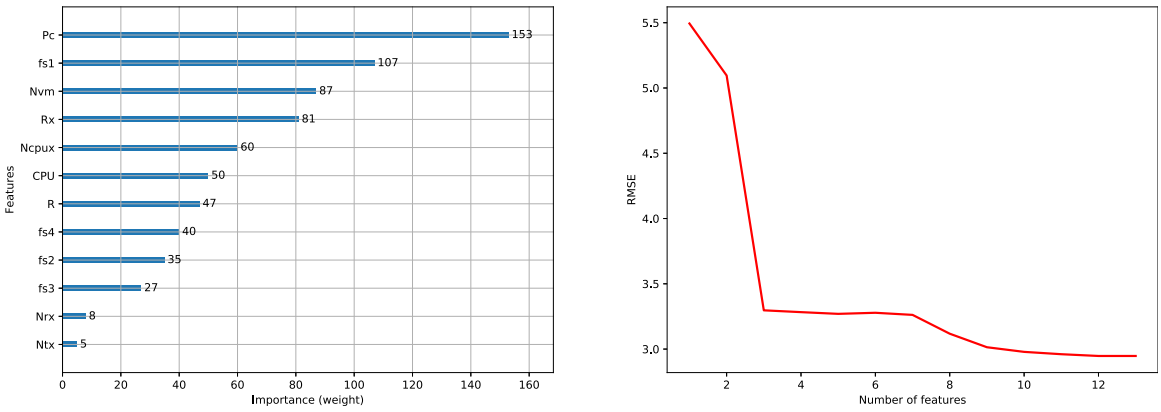
This also suggests that, to effectively use the prediction models, the training data should have a distribution of values of all hosts covering all possible ranges of values. Deploying such models in a real-world data center requires good coverage of data to handle all possible operating points of the data center so that when ML models are trained they will not be overfitted for a skewed range of data and thus perform poorly.

7.5 Assumptions and Applicability

The scheduling algorithm and prediction models proposed in this paper have the following assumptions and applicabilities. The scheduling algorithm is applicable for workloads that run in VMs for a long period without any interruptions (such as web and enterprise applications). Our policy tries to monitor the utilisation level of such workloads and consolidate them at regular intervals for energy efficiency while minimising the data center's peak temperature. The workload independent performance metrics in Section 7.3 indirectly captures the overhead of the scheduling algorithm. For other types of workloads such as tasks with predefined completion time, this algorithm is not directly applicable. In addition, the models trained from the particular data center should only be used in that data center. This is required to capture the unique characteristics and configuration of a data center that influences temperature variations in it. They include data center physical rack-layout, air circulation pattern, and server heat dissipation rate that directly affects the sensor readings and thus ambient temperature of server [7], [34], [36]. Hence, it is essential to train prediction models with data collected from a individual data center to capture its characteristics. However, our proposed techniques are still applicable in building such models. Therefore, the scheduling algorithm and prediction models are only suitable for a specific workloads, in a particular data center.

8 Feature Set Analysis

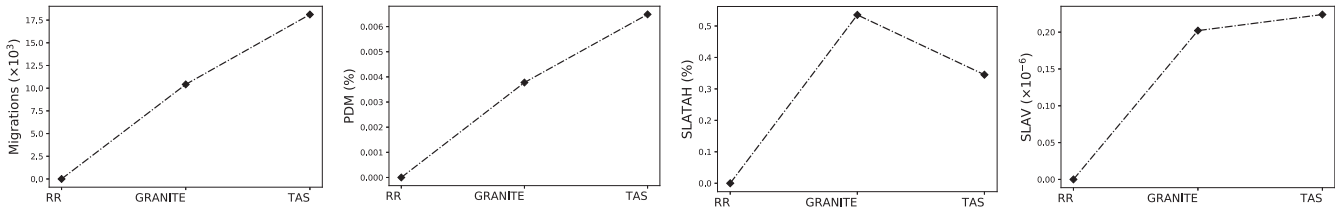
We carried out a feature analysis to identify the importance of each feature towards the model performance.



(a) Feature importance (weight)- number of times a particular feature occurs in the trees

(b) Feature threshold analysis

Fig. 8. Feature analysis.



(a) Number of VM migrations

(b) The PDM metric

(c) The SLATAH metric

(d) The $SLA_{violation}$ metric

Fig. 9. Performance overhead metrics.

This analysis can also be used in the feature selection process to remove the redundant features, reduce the computational cost, and increase the performance. Fig. 8a shows the importance of each feature in the constructed XGBoost model. Here, the weight metric associated with each feature corresponds to its respective number of occurrences in the constructed tree which indirectly notifies its importance. Based on the results, host power (P_c), fanspeed1 (fs_1) and number of VMs (N_vm) are the most important features towards accurate prediction. It is important to note that, though we have 4 fan speeds, the model intuitively selects one fan speed with more weight, this is since all four fans operate almost at the same rpm, which is observed in our data set. The least important feature is network metrics (N_{rx}, N_{tx}) along with the remaining three fan speed readings. The crucial observation is that the model gives high importance to power instead of CPU load, indicating, the high correlation between temperature and power. The number of cores (NC) is not included in the tree as it has constant value across hosts introducing no variation in the data.

The performance of temperature prediction with different thresholds can be observed in Fig. 8b. We start with the most important feature and recursively add more features according to their importance to the model. The y axis indicates RMSE value and the x axis shows a number of features. The first three features (P_c, fs_1, N_{vm}) significantly contribute to prediction accuracy and the accuracy gain is little as we add more features to the model. Therefore, based on the required accuracy or RMSE value, we can select top n features to effectively train the model with less complexity.

9 Related Work

Thermal management using theoretical analytical models has been studied by many researchers in the recent past [7], [8], [22], [37]. These models based on mathematical relationships to estimate the temperature are not accurate enough when compared to the actual values. Moreover, [7], [37] uses analytical models and targets HPC systems where jobs have specific completion time, while our work target the virtualized cloud datacenters with long-running applications that need dynamic scheduling and migration in realtime. Furthermore, some of the studies have also explored using CFD models [11]. Computational Fluid Dynamics (CFD) models provide an accurate thermal measurement, however, their massive computational demand hinders their adoption in realtime online tasks such as scheduling. Researchers are audaciously exploring data-driven ML algorithms to optimize the computing system efficiency [15], [19]. With the help of ML techniques, Google data centers are able to reduce up to 40 percent of their cooling costs [17].

Many researchers in recent years study thermal and energy management inside the data center using machine learning techniques. The vast applications have been used for finding an optimal setting or configurations of systems

to achieve energy efficiency [38]. However, ML techniques specific to temperature prediction are studied by Zhang *et al.* [36] where they proposed the Gaussian process-based host temperature prediction model in HPC data centers. They used a two-node Intel Xeon Phi cluster to run the HPC test applications and collect the training data. In addition, they also proposed a greedy algorithm for application placement to minimize the thermal variations across the system. In an extended work [6], they enhanced their solution to include more efficient models such as lasso linear and Multilayer Perceptron (MLP). The results have shown that predictive models are accurate and perform well in data center resource management aspects. Imes *et al.* [38] explored different ML classifiers to configure the different hardware counters to achieve energy efficiency for a given application. They tested 15 different classifiers including Support Vector Machine (SVM), K-Nearest Neighbours (KNN), and Random Forest (RF), etc. This work only considers energy as an optimization metric ignoring the thermal aspect. Moreover, these works are specific to HPC data centers where temperature estimation is done for application-specific which requires access to application counters. Nevertheless, our proposed solution is for Infrastructure clouds, where such an approach is not feasible due to limited access to application counters enforced by the isolated virtualized environment. Thus, we rely on features that completely surpass application counters and only consider host-level resource usage and hardware counters and yet achieve a high prediction accuracy.

Furthermore, Ignacio *et al.* [39] showed the thermal anomaly detection technique using Artificial Neural Networks (ANNs). They specifically use Self Organising Maps (SOM) to detect abnormal behavior in the data center from a previously trained reliable performance. They evaluated their solution using traces of anomalies from a real data center. Moore *et al.* [13] proposed Weatherman, a predictive thermal mapping framework for data centers. They studied the effect of workload distribution on cooling settings and temperature in the data center. These models are designed to find the thermal anomalies and manage the workload at a data center level without giving any attention to accurate temperature prediction.

In addition to thermal management, many others applied ML techniques for scheduling in distributed systems to optimize the parameters such as energy, performance, and cost. Among many existing ML approaches, Reinforcement Learning (RL) is widely used for this purpose [18], [40], [41]. Orheab *et al.* [40] studied the RL approach for scheduling in distributed systems. They used the Q-learning algorithm to train the model that learns optimal scheduling configurations. In addition, they proposed a platform that provides scheduling as a service for better execution time and efficiency. Cheng *et al.* proposed the DRL cloud, which provides an RL framework for provisioning and task scheduling in the cloud to increase energy efficiency and reduce the task execution time. Similarly, [41] *et al.* studied deep RL based resource management in distributed systems. Learning to schedule is prominent with RL based methods due to the fact that RL models keep improving in runtime [42] which is convenient for scheduling. However, our work is different from these works in a way that, the primary objective of our problem is to estimate the data center host temperature accurately to facilitate the resource management system tasks. In this regard, our work acts as complementary to these solutions where such thermal prediction models can be adopted by these ML-based scheduling frameworks to further enhance their efficiency.

10 Conclusion and Future Work

Estimating the temperature in the data center is a complex and non-trivial problem. Existing approaches for temperature prediction are inaccurate and computationally expensive. Optimal thermal management with accurate temperature prediction can reduce the operational cost of a data center and increase reliability. Data-driven temperature estimation of hosts in a data center can give us a more accurate prediction than simple mathematical models as we were able to take into consideration CPU and inlet airflow temperature variations through measurements. Our study which is based on physical host-level data collected from our University's private cloud has shown a large thermal variation present between hosts including CPU and inlet temperature. To accurately predict the host temperature, we explored several machine learning algorithms. Based on the results, we found a gradient boosting based XGBoost model for temperature prediction is the best. Our extensive empirical evaluation has achieved high prediction accuracy with the average RMSE value of 0.05. In other words, our prediction model has an average error of 2.38 °C. Compared to an existing theoretical model, it reduces the prediction error of 7 °C.

Guided by these prediction models, we proposed a dynamic scheduling algorithm for cloud workloads to minimize the peak temperature. The proposed algorithm is able to save up to 34.5 percent more of energy and reduce up to 6.5 °C of average peak temperature compared to the best baseline algorithm. It is important to note that, though the models built for one data center are optimized for its own (as each data center's physical environment and parameters vastly change), the methodology presented in this work is generic and can be applied to any cloud data center given the sufficient amount of data collected from the respective data centers.

In the future, we plan to explore more sophisticated models to achieve better accuracy and performance. We also intend to extend the work for heterogeneous nodes like GPUs or FPGAs. Another interesting direction is to consider parameters related to weather and predictions and their effect on cooling and scheduling long jobs.

Acknowledgment

The authors would like to thank Bernard Meade and Justin Mammarella at Research Platform Services, The University of Melbourne for their support and providing access to the infrastructure cloud and data. This work was supported in part by a Discovery Project research grant funded by the ARC (the Australian Research Council). They would also like to thank the editor-in chief, associate editor, and anonymous reviewers for their thoughtful suggestions on improving the article.

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[2] A. Shehabi *et al.*, "United States data center energy usage report," Lawrence Berkeley Nat. Lab., Berkeley, CA , USA, 2016.

[3] C. D. Patel, C. E. Bash, and A. H. Beitelmal, "Smart cooling of data centers," U.S. Patent 6 574 104, Jun. 3, 2003.

[4] ASHRAE, "American society of heating, refrigerating and air-conditioning engineers," 2018. [Online]. Available: http://tc0909.ashraetcs.org/

[5] U. PLC, "A guide to ensuring your ups batteries do not fail from ups systems," 2018. [Online]. Available: http://www.upssystems.co.uk/knowledge-base/the-it-professionals-guide-to-standby-power/part-8-how-to-ensure-your-batteries-dont-fail/

[6] K. Zhang *et al.*, "Machine learning-based temperature prediction for runtime thermal management across system components," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 2, pp. 405–419, Feb. 2018.

[7] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 11, pp. 1458–1472, Nov. 2008.

[8] H. Sun, P. Stolf, and J.-M. Pierson, "Spatio-temporal thermal-aware scheduling for homogeneous high-performance computing datacenters," *Future Gener. Comput. Syst.*, vol. 71, pp. 157–170, 2017.

[9] S. Zhang and K. S. Chatha, "Approximation algorithm for the temperature aware scheduling problem," in *Proc. Int. Conf. Comput.-Aided Des.*, 2007, pp. 281–288.

[10] S. Ilager, K. Ramamohanarao, and R. Buyya, "ETAS: Energy and thermal-aware dynamic virtual machine consolidation in cloud data center with proactive hotspot mitigation," *Concurrency Comput., Practice Experience*, vol. 31, no. 17, 2019, Art. no. e5221.

[11] J. Choi, Y. Kim, A. Sivasubramaniam, J. Srebric, Q. Wang, and J. Lee, "A CFD-based tool for studying temperature in rack-mounted servers," *IEEE Trans. Comput.*, vol. 57, no. 8, pp. 1129–1142, Aug. 2008.

[12] A. Almoli, A. Thompson, N. Kapur, J. Summers, H. Thompson, and G. Hannah, "Computational fluid dynamic investigation of liquid rack cooling in data centres," *Appl. Energy*, vol. 89, no. 1, pp. 150–155, 2012.

[13] J. Moore, J. S. Chase, and P. Ranganathan, "Weatherman: Automated, online and predictive thermal mapping and management for data centers," in *Proc. IEEE Int. Conf. Autonomic Comput.*, 2006, pp. 155–164.

[14] M. Zapater, J. L. Risco-Mart ín, P. Arroba, J. L. Ayala, J. M. Moya, and R. Hermida, "Runtime data center temperature prediction using grammatical evolution techniques," *Appl. Soft Comput.*, vol. 49, pp. 94–107, 2016.

[15] G. Fox *et al.*, "Learning everywhere: Pervasive machine learning for effective high-performance computation," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops*, 2019, pp. 422–429.

[16] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proc. ACM Special Interest Group Data Commun.*, 2019, pp. 270–288.

[17] J. Gao, "Machine learning applications for data center optimization," Google White Paper, 2014.

[18] M. Cheng, J. Li, and S. Nazarian, "DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2018, pp. 129–134.

[19] D. Jeff, "ML for system, system for ML, keynote talk in Workshop on ML for Systems, NIPS," 2018. [Online]. Available: http://mlforsystems.org/

[20] Y. Luo, X. Wang, S. Ogrenci-Memik, G. Memik, K. Yoshii, and P. Beckman, "Minimizing thermal variation in heterogeneous HPC systems with FPGA nodes," in *Proc. IEEE 36th Int. Conf. Comput. Des.*, 2018, pp. 537–544.

[21] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794.

[22] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, "Making scheduling "cool": Temperature-aware workload placement in data centers," in *Proc. USENIX Annu. Tech. Conf.*, 2005, pp. 61–75.

[23] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, no. Oct., pp. 2825–2830, 2011.

[24] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 161–168.

[25] R. Bekkerman, "The present and the future of the KDD cup competition: An outsider's perspective," 2015. [Online]. Available: https://www.linkedin.com/pulse/present-future-kdd-cup-competition-outsiders-ron-bekkerman

[26] A. Mangal and N. Kumar, "Using big data to enhance the bosch production line performance: A Kaggle challenge," in *Proc. IEEE Int. Conf. Big Data*, 2016, pp. 2029–2035.

[27] R. Bianchini *et al.*, "Toward ML-centric cloud platforms," *Commun. ACM*, vol. 63, no. 2, pp. 50–59, 2020.

[28] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

[29] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *Proc. USENIX Annu. Tech. Conf.*, 2009, pp. 28–28.

[30] M. Xu, A. V. Dastjerdi, and R. Buyya, "Energy efficient scheduling of cloud application components with brownout," *IEEE Trans. Sustain. Comput.*, vol. 1, no. 2, pp. 40–53, Jul.–Dec. 2016.

[31] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Practice Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[32] S. Shen, V. van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *Proc. 15th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2015, pp. 465–474.

[33] SPEC, "Standard performance evaluation corporation," 2018. [Online]. Available: https://www.spec.org/benchmarks.html

[34] X. Li, P. Garraghan, X. Jiang, Z. Wu, and J. Xu, "Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1317–1331, Jun. 2018.

[35] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in *Proc. IEEE Int. Conf. Cloud Comput.*, 2009, pp. 254–265.

[36] K. Zhang, S. Ogrenci-Memik, G. Memik, K. Yoshii, R. Sankaran, and P. Beckman, "Minimizing thermal variation across system components," in *Proc. Int. Parallel Distrib. Process. Symp.*, 2015, pp. 1139–1148.

[37] T. Cao, W. Huang, Y. He, and M. Kondo, "Cooling-aware job scheduling and node allocation for overprovisioned HPC systems," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2017, pp. 728–737.

[38] C. Imes, S. Hofmeyr, and H. Hoffmann, "Energy-efficient application resource scheduling using machine learning classifiers," in *Proc. 47th Int. Conf. Parallel Process.*, 2018, pp. 45:1–45:11.

[39] I. Aransay, M. Z. Sancho, P. A. García, and J. M. M. Fernández, "Self-organizing maps for detecting abnormal thermal behavior in data centers," in *Proc. 8th IEEE Int. Conf. Cloud Comput.*, 2015, pp. 138–145.

[40] J. P. D. Comput, A. Iulian, F. Pop, and I. Raicu, "New scheduling approach using reinforcement learning for heterogeneous distributed systems," *J. Parallel Distrib. Comput.*, vol. 117, pp. 292–302, 2018.

[41] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. 15th ACM Workshop Hot Topics Netw.*, 2016, pp. 50–56.

[42] R. S. Sutton *et al.*, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1998.