2017 IEEE 24th International Conference on High Performance Computing (HiPC)

Support for Power Efficient Proactive Cooling Mechanisms

Bilge Acun*, Eun Kyung Lee†, Yoonho Park†, Laxmikant V. Kale*
*University of Illinois at Urbana-Champaign
{acun2, kale}@illinois.edu
†IBM T.J. Watson Research Center
{eunkyung.lee, yoonho}@us.ibm.com

***Abstract*—Increasing scale of data centers and the density of server nodes pose significant challenges in producing power and energy efficient cooling infrastructures. Current fan based air cooling systems have significant inefficiencies in their operation causing oscillations in fan power consumption and temperature variations among cores. In this paper, we identify the cause these problems and propose proactive cooling mechanisms to mitigate the power peaks and temperature variations.**

An accurate temperature prediction model lies behind the basis of our solutions. We use a neural network-based modeling approach for predicting core temperatures of different workloads, under different core frequencies, fan speed levels, and ambient temperature. The model provides guidance for our proactive cooling mechanisms. We propose a preemptive and decoupled fan control mechanism that can remove the power peaks in fan power consumption and reduce the maximum cooling power by 53.3% on average as well as energy consumption by 22.4%. Moreover, through our decoupled fan control method and thermal-aware load balancing algorithm, we show that temperature variations in large scale platforms can be reduced from 25° C to 2° C, making cooling systems more efficient with negligible performance overhead.

I. Introduction

As a 20 MW power limit for an exascale system set by Department of Energy, reducing the power consumption of high performance data centers has become an important challenge [1]. Moreover, some supercomputing facilities, such as Oak Ridge National Laboratory that hosts the largest supercomputer in the United States – Titan, is charged based on its maximum power usage [2]. These has increased the importance of making optimizations for power-constrained systems, improving the performance under a strict power budget and similar power focused research [3], [4], [5] compared to prior focus on energy optimizations.

Up to half of the data center power consumption can be cooling costs [6]. Current fan based air cooling systems have significant inefficiencies in their operation. In this paper, we identify and tackle two of them. First, fans have a reactive behavior to the temperature changes in the cores that cause oscillations in fan speeds and hence power peaks. Second, fans in the server nodes do not take into account temperature variations that can happen among cores and act based on maximum core temperature within the node. We analyze temperature variations within a chip and across chips using more than 1,000 cores in two different architectures and show that there can be up to 25° C difference running a balanced benchmark. To mitigate these problems, we propose novel proactive cooling mechanisms that remove the oscillations in fan speed, reduce temperature variations and make cooling systems more power and energy efficient.

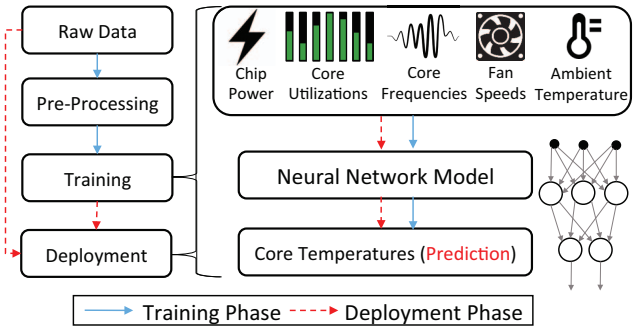

Figure 1: Neural network-based prediction approach.

Our proactive cooling mechanisms require having an accurate temperature prediction model. Accurately predicting core temperatures is difficult due to multiple factors including sophisticated workloads, complex thermodynamics within the data center, physical layout of each core/node and inherent manufacturing related CMOS process variations. We propose a learning-based temperature prediction modeling approach that can capture complex parameters causing on-chip temperature variations using a neural network (NN). Figure 1 shows our approach which consists of four steps: (1) monitor and collect core temperatures data under different utilization rates, core frequencies, fan speed, ambient temperatures (2) pre-process data, (3) train the neural network model, and (4) deploy the neural network to provide core temperature predictions. We construct the neural network model for each chip independently, which provides more accurate temperature predictions than existing model-based approaches that assume all instances of a given hardware component behave similarly [7], [8].

The main contributions of this paper are the following:

- We propose a simple yet powerful neural network-based temperature prediction model that can predict steady-state core temperatures accurately under different core utilization levels, frequency levels, and fan speeds.
- We propose a proactive fan control mechanism, *precooling*, that removes the power oscillations and can reduce the maximum fan power by 45.6% on average as well as energy consumption by 9.4% by predicting core temperatures.
- We analyze inter-chip and intra-chip temperature variations in two different architectures using 1,000 cores with 25° C variation. We show that decoupling the fans reduces the variation down to 10° C, the power by additional 7.7%,

0-7695-6326-0/17/$31.00 ©2017 IEEE
DOI 10.1109/HiPC.2017.00020

and the energy by 13%. We show that the remaining 10° C is intra-chip variation and can be reduced to 2° C via thermal-aware load balancing.

II. Motivation and Observations

In this section, we share some observations about the cooling fans in the server nodes that set the basis of our motivation in this paper. Cooling fans are used to reduce the temperature of hardware components in the servers by drawing cool air from outside into the server node and moving heat away from the heat sinks. The particular server nodes we study in this paper are IBM POWER8 nodes with four individual fans at the edge of the case as illustrated in Figure 2. These four fans are responsible for cooling the two chips, memory buffers and GPUs in the server node. If the temperature of any of these hardware components hits their threshold ($73°C$, $79°C$, and $74°C$ respectively), the fans increase their speed rapidly in a *reactive* way. If all of the components are under their threshold, fan speed is calculated based on the ambient temperature. Analyzing the fans, we observe three critical behaviors: (1) synchronous control, (2) cubic polynomial exponential increase in powe, and (3) oscillations in speed.

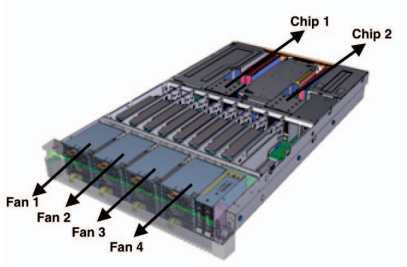

Figure 2: POWER8 server node architecture illustration.

Synchronous Control: The four individual fans in the node operate synchronously, i.e., they have the same speed in terms of Revolutions Per Minute (RPMs). However, because of the physical layout Fans 1, 2 have more cooling effect on Chip 1 and Fans 3, 4 have more effect on Chip 2. Figure 3 shows the temperature variations among the cores within a compute node. There is an $8°C$ difference between state temperatures of the cores with maximum being $72°C$ while running a balanced benchmark. The fans keep the maximum temperature of the cores right under the threshold of $73°C$, however Chip 1 is over-cooled $8°C$ with maximum temperature of $64°C$. Synchronous control of the fans can result in even larger with an imbalanced workload. For example, when the workload is running on Chip 2 and Chip 1 is left idle, the temperature difference among the cores increases up to $25°C$. Trying to cool down the active chip, the idle chip is over-cooled.

In summary, "hot" cores can cause the cooling system to activate unnecessarily and result in over-cooling surrounding hardware. "Hot" chips can have the same effect, and we analyze the temperature variations at larger scale in Section IV. Low core temperatures have the potential benefit of reducing fault rates [9]. In this paper, we consider the cores under a certain threshold (i.e., fan kick-off threshold) to be equal in terms of reliability and focus our evaluation on power and energy consumption.

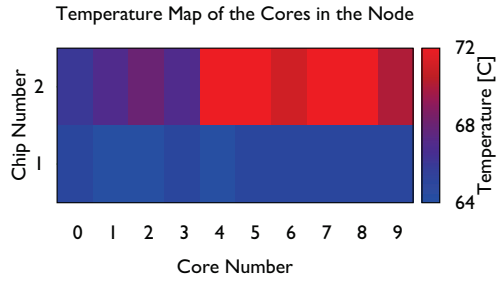

Figure 3: There is an 8°C steady-state temperature variation among the cores within a node running a balanced benchmark, LeanMD.

Cubic Polynomial Growth in Power: Power consumption of the fans grows cubic polynomially with respect to fan speed [10]. Figure 4 shows the total power of the four fans at different fan speed levels. We use simple curve fitting to calculate the model function parameters shown in the plot. Total power can peak at over 200W. Even the bursts with short duration can cause significant power peaks. We show examples of such scenarios next.

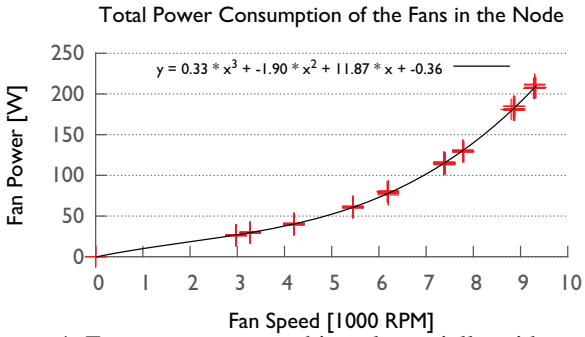

Figure 4: Fan power grows cubic polynomially with respect to fan speed.

Oscillations in Fan Speed: The *reactive* behavior of the fans result in oscillations in the fan speed and power. Figure 5 shows this behavior over time under different CPU utilization levels with a synthetic workload generator which starts at around 300 seconds. The lower utilization levels shows lower peaks in fan, whereas higher levels cause bigger jumps. In the 100% CPU utilization case, the total fan power peaks to 81W from idle power of 29W, then settles at 40W which is half of the maximum fan power. This behavior can also differ based on the application and number of threads as it can be seen in Figure 6. DGEMM has the least peaks since its running on 20 threads, whereas the other benchmarks are using all 160 SMT threads on the node. kNeighbor and LeanMD benchmarks are compute intensive and fans are able to stabilize after a single peak. On the other hand,

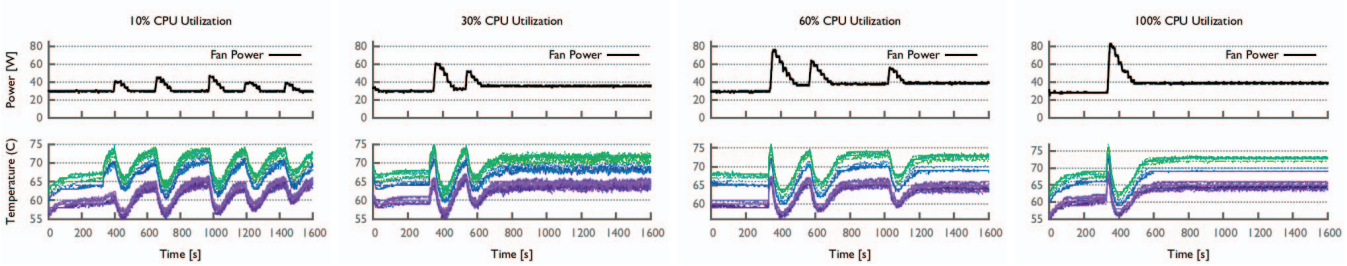

Figure 5: Plots show fan power (top) and core temperature (bottom) behavior from 10% to 100% CPU utilization levels (from left to right). Purple and green lines in the bottom plots represent cores in Chip-1 and Chip-2 respectively.

Stencil3D is a memory intensive benchmark and fans do not seem to be able stabilize even after 15 minutes of the application run.

This oscillatory behavior has multiple downsides. First, maximum power consumption is an important cost metric for large data centers and since the fans settle much lower than their maximum power, the budgeted power will not be fully used. Second, oscillations can cause fan wear and reduce hardware lifetime. Finally, making power/temperature predictions and creating power/temperature profiles for applications becomes much harder.

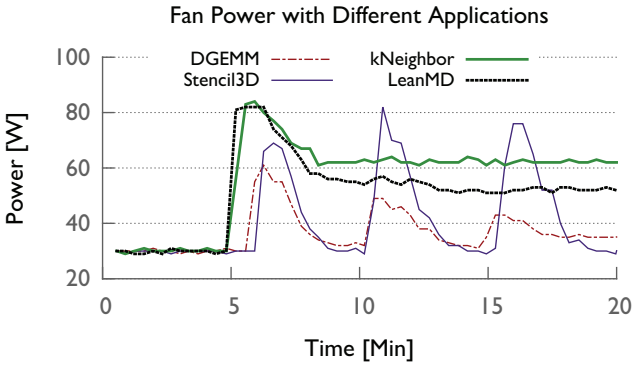

Figure 6: Fan behavior under different applications.

III. Neural Network-Based Temperature Prediction Model

In this section, we describe our experimental setup and provide the details of our temperature prediction model and its validation. Our model is a proof-of-concept prototype that can do fine-grained temperature predictions for our proactive cooling mechanisms, which is described later in Section IV.

A. Experimental Setup

We use an IBM cluster with POWER8 processors running at 3.5 GHz nominal clock speed in our experiments. Each node in the cluster has two sockets. Each socket has 10 physical cores. Each core has 8 hardware threads. The On-Chip Controller (OCC) provides temperature and power data. The Baseboard Management Controller (BMC) provides fan speed data and fan speed control. The MATLAB Neural Network Toolbox [11] is used for the modeling.

Benchmarks: We use a set of benchmarks that exhibit different characteristics for training and performing our experiments. These consist of compute intensive applications *DGEMM and LeanMD*, a memory intensive application *Stencil3D*, a communication intensive application *kNeighbor*, and *lookbusy* synthetic workload generator. *Stencil3D* is a 7-point stencil application based on a 3D grid using Jacobi kernel. *DGEMM* is a naive 3-loop double precision matrix multiply code. *LeanMD* is the mini-app version of the molecular dynamics application NAMD [12]. It simulates the behavior of atoms based on the Lennard-Jones potential in a 3D space consisting of atoms which are divided into cells with short and long-range force calculations. *kNeighbor* is a micro-benchmark with a near-neighbor communication pattern where each object exchanges fixed-sized messages with a fixed set of fourteen neighbors in every iteration. *lookbusy* generates random instructions for Linux systems and is useful for analyzing different scenarios easily as custom custom CPU utilization levels can be specified for each core [13].

B. Model Description and Validation

The computational neural network is inspired by biological neural networks to predict or approximate functions that can depend on a large number of inputs. There are two phases in using a neural network: training and deployment. During the training phase, neurons in each layer are adjusted iteratively using training data. Then, the trained neurons are used to predict the new output in the deployment phase. Figure 1 shows our neural network design. First, input data is pre-processed to obtain steady-state temperatures, i.e. when the heat extraction is same as heat generation, since we are not interested in predicting transition state temperatures, for example when the application has just started. Second, pre-processed input data is fed to the input layer of the neural network for training. After the training phase, the neural network is deployed to provide core temperature predictions. The information flow of the training phase and the deployment phase is shown in Figure 1 as the solid blue line and dotted red line, respectively.

In a neural network, each connection between neighboring layers has a weight to scale data and a bias that allows

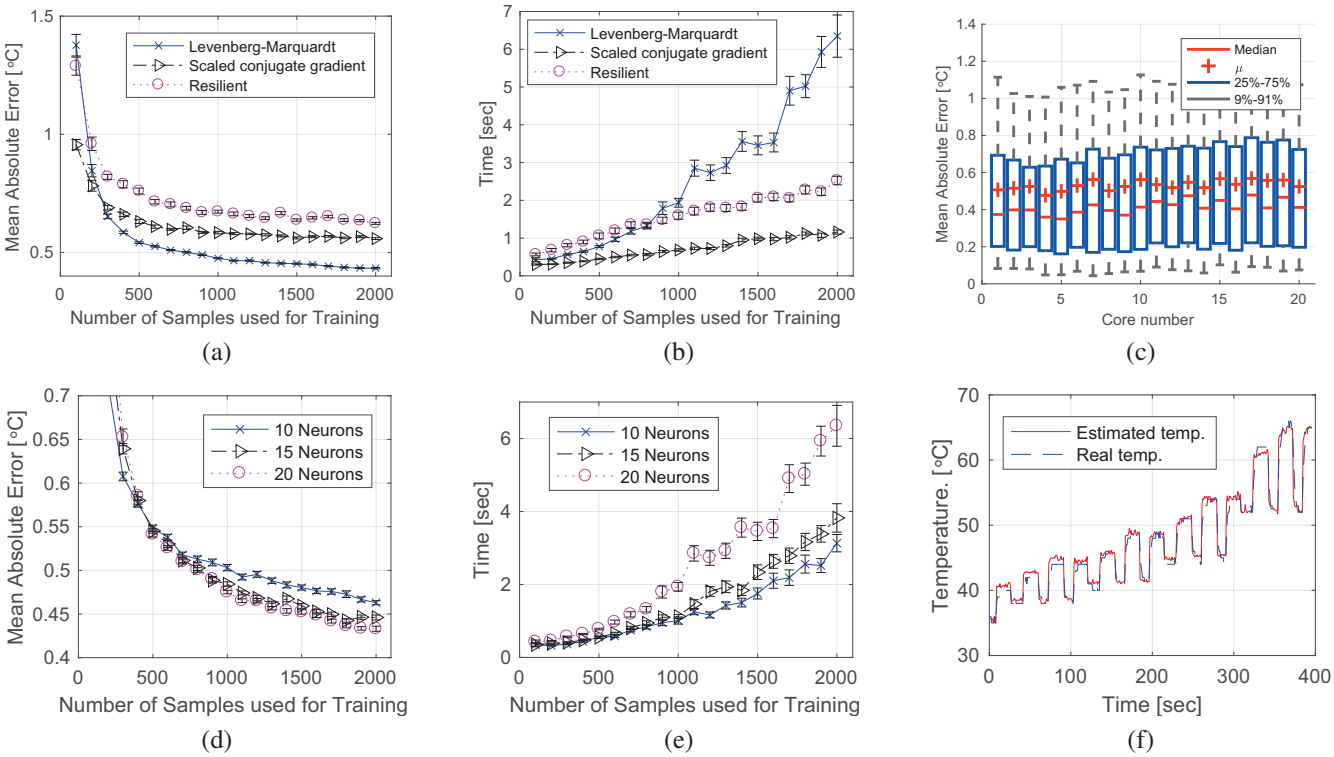

Figure 7: (a) Mean absolute error using different back-propagation algorithms; (b) Training time using different back-propagation algorithms; (c) Mean absolute error per core using Levenberg-Marquardt algorithm; (d) Mean absolute error using different number of neurons; (e) Training time using different number of neurons; and (f) Model validation while increasing CPU frequency. Plots are shown with 95% confidence interval.

shifting of the activation function. Data (with 9 data points) from the input layer are inserted as inputs to the next consecutive layers (hidden layers). Then, the hidden layers sum the data fed to them, scale (weight) the data, and process it until the data reaches the last layer that outputs the predicted core temperature. 9 input data points includes chip and memory power (1-2), core utilization (3), core frequency (4), fan speeds (5-8), and ambient temperature (9). The 10 output data points are core temperature as there are 10 temperature sensors per socket. The weights and biases of one neural network are updated as follows in the training phase:

$$w_{ij}(k+1) = w_{ij}(k) - \eta \frac{\delta e_k}{\delta w_{ij}} \quad (1)$$

η is the learning rate parameter, which determines the rate of learning. w_{ij} represents the scalar value of weight on the connection from layer i to j. e_k represents the error of NN at the kth iteration. $\delta e_k / \delta w_{ij}$ determines the weighted search direction for this iterative method.

The weights and biases of the network are updated only after the entire training set has been applied to the network. 90% of the randomly selected data is used as a training set and 10% is used as a validation set. An entire data set was collected for different utilization rates (idle-100%), CPU frequencies (2 - 3.5 GHz), and fixed fan speeds (3300 - 5800 RPM) with the auto control mechanism disabled. The gradients calculated for each training set are added together to determine the change in weights, and biases. The weights and biases are updated in the direction of the negative gradient of the performance function.

We try three commonly used back-propagation algorithms: *Levenberg-Marquardt* [14], *Scaled conjugate gradient* [15] and *Resilient* [16]. Figure 7 (a) shows the Mean Absolute Error (MAE) of using different back propagation algorithms. MAE is a good indicator of predictor performance and tends to decrease with the number of training samples. Our neural network becomes more knowledgeable about the relationship between temperature changes and resource allocation as it processes more training samples. The Levenberg-Marquardt algorithm performs the best in terms of the accuracy (showing minimum mean error) but has more computational overhead (showing higher execution time for training) with larger number of neurons. The time elapsed for training is shown in the Figure 7 (b) and (e). Scaled conjugate gradient performs the best in terms of computational overhead showing less time for training than the other algorithms. Figure 7 (c) shows MAE is consistent with all of the cores. We choose the Levenberg-Marquardt back-propagation algorithm because our highest priority is

model accuracy. However, scaled conjugate gradient, or resilient algorithms can be used if learning overhead is a constraint. As Figure 7 (d) shows, since using more than 15 neurons does not improve model accuracy significantly, we use 15 neurons for the hidden layer. We verified the accuracy of neural network approach by repeating the experiment and comparing the predicted core temperature over time with the actual core temperature as shown in Figure 7 (f). Our neural network will be able to make accurate predictions despite slight hardware differences, variable heat, and air circulation patterns (thermodynamic phenomena) of different nodes and, furthermore, regions inside a data center.

IV. Proactive Cooling Control

Having an accurate temperature prediction model enables us to implement proactive cooling mechanisms that mitigate the problems described earlier in Section II. These problems include temperature variations and reactive and oscillatory behavior in fan speed. In this section, we propose a preemptive fan control mechanism and compare it with two other mechanisms including the existing reactive control mechanism. Below are the fan control mechanisms we compare:

1) Reactive Fan Control (Default Mode): This is the existing method used in the server nodes. As we described in Section II, fan speed is determined by core and ambient temperatures. If any core exceeds the temperature threshold of 73°C, the fans increase their speed to cool down the chips. The floor fan speed, i.e. when all components are under their threshold, is determined by the ambient temperature. This approach consumes minimal power if the components are under the threshold. However, once the cores hit the threshold, the fans react rapidly. This results in peaks in power consumption and in certain scenarios such as compute-intensive workloads, the fan power can triple. The fans can also be triggered by other components such as GPUs or memory hitting their threshold. In our experiments, CPU core temperatures are the sole reason for fans to trigger.

2) Constant Fan Control: As the name implies, this technique uses a constant fan speed regardless of the processor utilization or the benchmark running. This constant speed is sufficient to cool down the most compute-intensive benchmark, therefore it guarantees the components will operate under the temperature threshold. This constant fan speed can be determined by running a synthetic benchmark which fully utilizes the CPU or the most compute-intensive benchmark that is projected to run, and measuring the fan speed that can keep core temperatures under the threshold. Since this is the worst case, any other benchmark will be guaranteed to operate under the temperature limit. There are advantages and disadvantages with this technique. The advantages are the removal of the oscillatory behavior, having a predictable fan power, and a potential reduction in the maximum fan power. The disadvantage is the increased overall power and energy consumption because the cores will be over-cooled when they are idle or running a non-intensive benchmark.

3) Preemptive Fan Control: This is our proposed model-guided fan control mechanism that preemptively cools the processor before the cores hit the threshold. We also refer to it as *precooling*. (The idea is conceptually similar to *prefetching* where the data is brought from memory ahead of time.) In *precooling*, the processor is cooled before the application starts or the fan speed is adjusted for different phases of the application, i.e. before a computationally intensive phase. *Precooling* speed is determined from the temperature prediction model using some hints about the application profile such as power, utilization etc. as we described in Section III-B. The model predicts the steady-state temperatures under different fan speeds. Then, the desired speed can be set via the job scheduler of the cluster or the runtime system of the application.

In Figure 8, we compare the fan power and the core temperatures under these three cooling control mechanisms using the LeanMD benchmark. At the start when the cores are idle, the reactive and preemptive control mechanisms have the same speed and temperatures, whereas cores under constant fan control are much cooler since fan runs at a higher speed even when the processor is idle. Before the application starts at 100s, preemptive control starts the cooling process and the fan speed is set to the stable level of the benchmark using the temperature prediction model. In reactive mechanism, the fans start after the maximum core temperature hit the threshold, which is around 15 seconds after the application start. Only after 300 seconds (5 minutes) from the application start, the fans gradually lower their speed and stabilize at 4900 RPM and 51W. Preemptive and reactive mechanisms converge to the same temperature and power level at around 400s. On the other hand, constant fan control shows 5°C lower maximum core temperature. Since we determine the constant speed for the worst case benchmark in our benchmark set (which is kNeighbor), it causes over-cooling for the LeanMD benchmark.

Table I: Peak fan power and energy consumption of different fan control mechanisms and benchmarks.

	Reactive	Preemptive	Constant
DGEMM	61W, 19.4kJ	38W, 17.7kJ	63W, 31.3kJ
Stencil3D	81W, 21.0kJ	42W, 18.7kJ	63W, 31.3kJ
kNeighbor	84W, 29.7kJ	63W, 27.7kJ	63W, 31.3kJ
LeanMD	83W, 26.8kJ	51W, 23.7kJ	63W, 31.3kJ

Table I shows the maximum power and total energy consumption of the three mechanisms in a 500-second window where the application starts at 100s as shown in Figure 8. Overall, the preemptive mechanism reduces the peak fan power by 40.5% on average (48% with DGEMM, 50% Stencil3D, 25% with kNeighbor, and 39% with LeandMD) compared to the reactive mechanism. It also reduces the en-

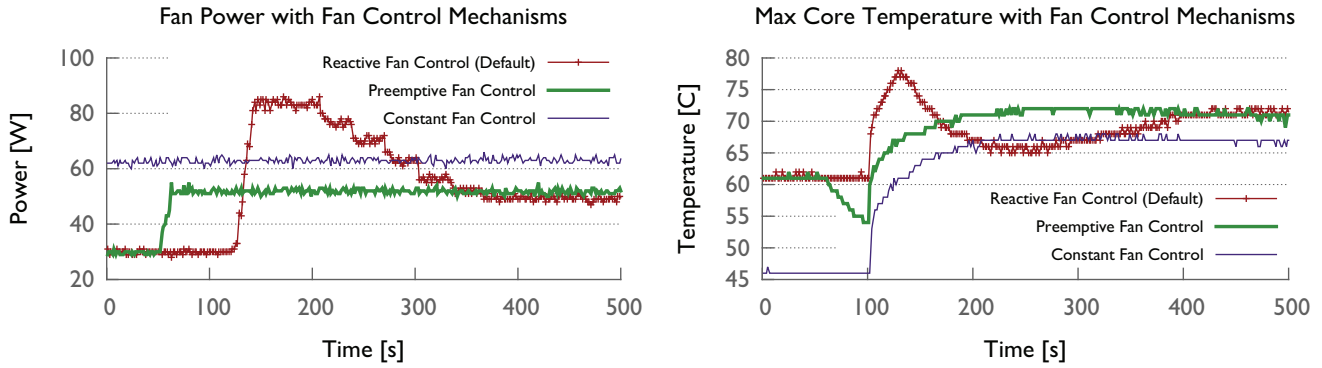

Figure 8: Power and temperature comparison of the three fan control mechanisms with *LeanMD* benchmark starting at 100s. (Preemptive cooling has been started 50s ahead of time for clarification of the plots and the idea. It can achieve a similar effect if triggered within few seconds of the application start as well.)

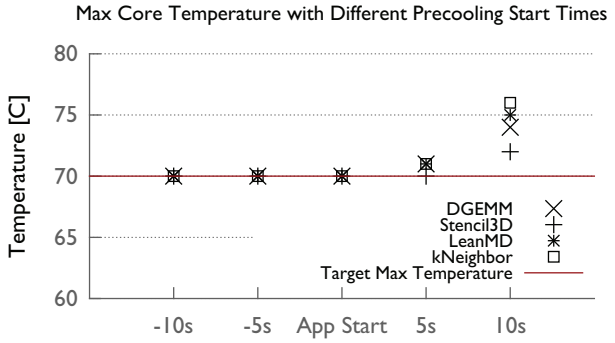

Figure 9: Precooling should start within a few seconds of application start at the latest.

ergy consumption by 9.4% on average (8.7%,10.9%, 6.7%, 11.5% with the four benchmarks respectively). Constant fan control is able to reduce the peak fan power at most by 25%, but it increases the energy consumption by 61% which is quite significant and undesirable. In summary, preemptive cooling mechanism provides the minimum power and energy in all cases. We have also observed reduction in chip's maximum power consumption due to the reduction in the peak temperature of the cores in preemptive and constant fan control. However, because the reduction is not significant (just a few watts), we are not including it in our results.

When to start *precooling?*: A critical factor in *precooling* is to determine when precooling should start. Is it before the application starts, when the application starts, or even after the application starts? How many seconds before or after the application starts? When the CPU is transitioning from a low-utilization state to high-utilization state, it is important to have sufficient time to cool the cores enough. This is not as critical when the CPU is transitioning from a high-utilization state to low-utilization state, since the fans are going to be set to a lower level and delay in setting the new level will not harm the cores but only cause some minimal energy loss compared to the optimal case. Figure 9 shows that precooling needs to start within a few seconds after the application start at the latest, otherwise the target threshold will be exceed. The window can be extended by a few seconds further if the application is not as intense and the temperatures are not increasing rapidly.

Decoupling the Fans: So far, we have shown how preemptive fan control can remove the fan oscillations and power peaks on one node. Next, we analyze how fans behave in large scale. In particular, we look into temperature variations and investigate decoupling the fans as a solution.

After showing the benefits on one node, we look at the fan behavior on 90 nodes and how much preemptive cooling can help at larger scale. Figure 10 shows the maximum and stable fan power of 90 nodes running LeanMD benchmark. The difference between maximum and stable power is the savings that preemptive cooling can provide for each node. The reason that each node has a different fan power despite running the same benchmark is because of the temperature variations across the nodes.

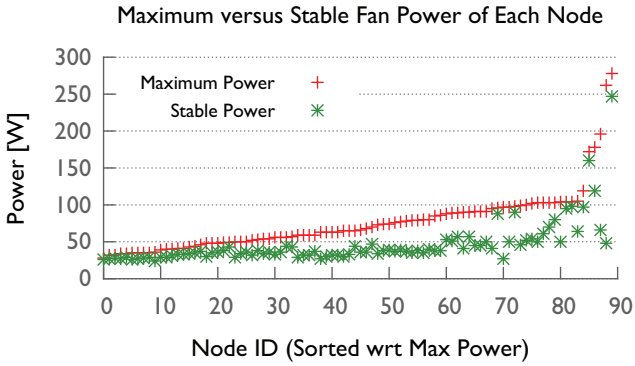

Figure 10: The difference between the maximum and the stable fan gives the power reduction that preemptive cooling can achieve with LeanMD.

The temperature variation is partly caused by the synchronous fan behavior. The four fans in the server node change their speed synchronously, however each of the

Table II: Fan Power Reduction in Large Scale

Optimization \ Benchmarks	DGEMM	Stencil3D	kNeighbor	LeanMD	Average
Reactive Fan Control	5868 W	13433 W	6769 W	6770 W	8210 W
Preemptive Fan Control	3893 W	8526 W	4381 W	4224 W	5256 W
Preemptive and Decoupled Fan Control	3179 W	7972 W	3765 W	3569 W	4621 W
Total Power Reduction (%)	45.8	59.3	55.6	52.7	53.3

Table III: Decoupling the left two and the right two fans reduces chip-to-chip temperature variations, power, and energy consumption.

LeanMD	**Synchronous**	**Decoupled**
Fan Speeds (RPM)	4900, 4900	3900, 4900
Max Temp Chip 1, 2 (C)	63, 71	70, 71
Temp Variation (C)	8	5
Power (W)	51	44
Energy (kJ)	23.7	20.6

individual fans has a different cooling effect on the two chips because of the physical layout, which can be seen in earlier Figure 2. The fans are more effective in cooling closer chips. If the fans are decoupled, i.e. controlled independently, temperature variations can be reduced, and the fans could consume less power and energy. To evaluate the potential benefits of decoupling the fans, we first analyze temperature variations in large scale. Figure 11 shows temperature distribution of 1,800 cores in two different platforms: Cori and Minsky. Cori has Intel Haswell generation processors with 36 cores per node, and Minsky has IBM POWER8 processors with 20 cores per node. Our goal is not to compare the cooling efficiency of one platform to another. We are simply showing the significance of the temperature variation problem and that it is not unique to a specific platform. The rest of our experiments are solely done on Minsky. The maximum core-to-core temperature difference is 22 °C for Cori and 25 °C for Minsky. We can not include the distribution of other benchmarks due to space restrictions however, all of the benchmarks show very similar temperature distribution. While this may sound surprising, it is because all of the benchmarks use the cores in a balanced manner, they all exceed the fan kick-off threshold and the fans stabilize the core temperatures optimally just below the threshold. Therefore, it is expected to see a similar distribution.

The 25 °C variation on Minsky can be partially prevented by decoupling the fans. We have observed that the left two fans affect Chip 1 and the right two fans affect Chip 2 most significantly. The effect on the cross/diagonal chips are minimal. Therefore, we decouple the control of the left two fans and right two fans from each other and evaluate potential power reductions. Figure 12-left shows the new temperature distribution of the cores after fans are decoupled. The maximum temperature variation is reduced from 25 °C to 12 °C with the lowest temperature of 64°C. The remaining variation is intra-chip variation that fan decoupling cannot resolve. Figure 12-right shows the distribution of intra-chip temperature variation. While the majority of the chips have less than 6°C temperature difference among their cores, the range can go up to 10°C. We analyze ways to mitigate the intra-chip temperature variation, and the potential benefits in the next section (V).

Table III shows various node parameters when the fans are decoupled. First, the left two fans are now running at 1,000 RPM less, and this results in 7 W reduction in the stable fan power. Power reduction also results in 13% less energy consumption. The maximum core temperature in Chip 1 is increased, but it is still kept under the threshold. In-node temperature variation reduces from 8 to 5°C. The remaining 5°C temperature variation is intra-chip variation that decoupling cannot mitigate.

To summarize, Table II shows the cumulative power consumption of the 90 nodes under reactive control, preemptive control, and decoupled control. On average, preemptive and decoupled fan control methods can reduce the maximum power by 53.3% (45.6%, 7.7% respectively) and energy by 22.4% (9.4%, 13% respectively).

Application and Safety of Model-Based Fan Control: *Precooling* speed can be set via job scheduler or application runtime. As we show earlier, in the most cases fans are able to find a stable level after one or a few peaks. A job scheduler approach would eliminate that initial peak for most scenarios and set the speed back to optimal idle speed at the end of the application. On the other hand, runtime systems can do more fine-grained optimizations for applications with different phases. The runtime system can detect the start of a different phase and determine a new optimal fan speed. The phase could be either transitioning from non-CPU-intensive to high-intensive or vice versa. However, one needs to consider that machine learning models can be error prone for different reasons such as noise, estimation bias, and variance (commonly known as over-fitting and under-fitting). Therefore, using a learning mechanism for a safety-critical feature such as temperature control requires fail-safe mechanisms. One approach is to set the maximum threshold lower than hardware controller's level to have a safety margin in case there are 2-3°C errors in the prediction. The system can still continue running while continuing to train the model to make better predictions in the future. If the

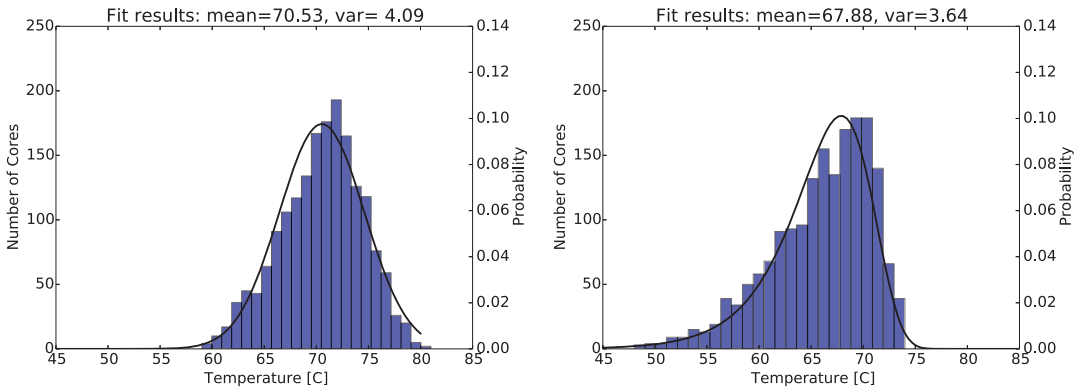

Figure 11: Steady-state temperature distribution of 1,800 cores running DGEMM kernel in two different architectures: Cori (left) with Intel Xeon Haswell processors and Minsky (right) with IBM POWER8 processors.

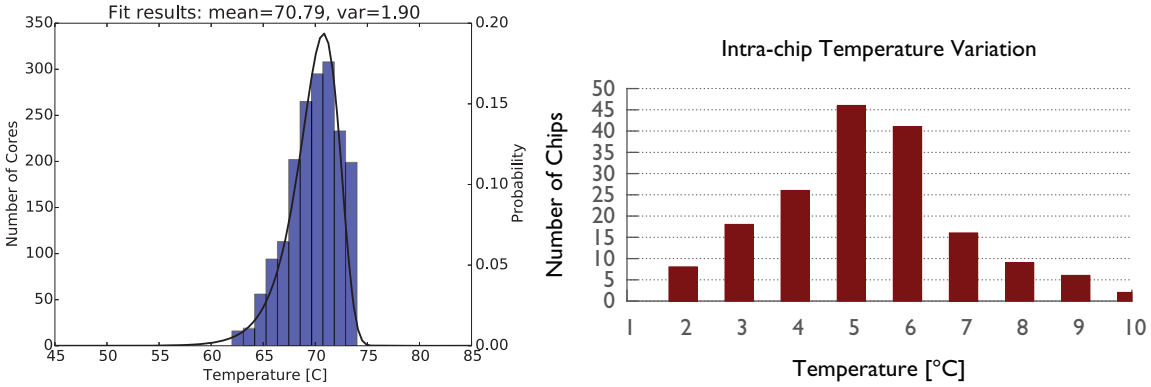

Figure 12: The left plot shows steady-state temperature distribution of the DGEMM benchmark with decoupled fans on Minsky (Notice the range change in the y axis). Independent fan control can not remove temperature variations fully mostly because of the intra-chip variations. The right plot shows the distribution of intra-chip temperature variation among cores.

error is more than that, then hardware controllers can take over the control overriding the model-guided mechanism.

V. Mitigating Intra-Chip Temperature Variation

In this section, we investigate methods to solve intra-chip temperature variation problem that preemptive and independent fan control can not mitigate fully. Dynamic Voltage and Frequency Scaling (DVFS) and load balancing are two potential solutions:

1) Temperature-Aware Frequency Control: DVFS is a common technique used in power and energy optimizations. It can be used to mitigate the temperature variations by reducing the frequency of the high temperature cores. DVFS can be done in core-level or chip-level. While the core-level one allows finer-grained control, chip-level DVFS is ineffective in mitigating core level temperature variations. Yet, many commercial processor architectures only support chip-level DVFS since core-level DVFS requires having more complex and costly independent voltage regulators in hardware. Unlike many others, POWER8 chips can support per-core DVFS [17], however it is not a production feature. For Intel processors, only Haswell generation supports per-core DVFS and the support has been discontinued for future generations [18]. We conclude, since core-level DVFS is not currently supported in many processors yet, this is not a viable solution.

2) Temperature-Aware Load Balancing: Load balancing can be used to eliminate within chip temperature variations by moving the work units away from the hot cores. We implemented a new load balancing algorithm using Charm++ 's load balancing framework. In Charm++ , the workload is represented as C++ objects which can migrate from processor to processor [19]. The runtime system moves objects from high-temperature (above average) cores to low-temperature (below average) cores in order to create temperature balance. The neural network model predicts core temperatures of potential object migration, and allows us to determine the best workload distribution. Since the neural network uses CPU utilization as input, we need to approximate the load of each task in terms of CPU utilization. We do this by calculating the ratio of a task's execution time and the total execution time of the tasks on the same processing core from historical information stored in Charm++ 's runtime database.

Figure 13 shows how core temperatures change over time

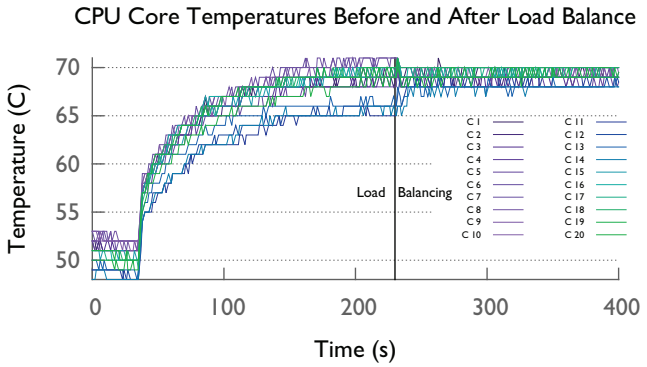

Figure 13: Load balancing reduces within chip temperature variations from 5°C to 2°C. (Inter-chip variation is mitigated by decoupled fans.)

with load balancing triggered around 220s. It can be triggered earlier as well, it's done after the cores heat up only to show the difference between before and after load balancing temperatures clearly in the plot. Load balancing reduces the temperature variation from 5°C to 2°C, and the maximum core temperature from 71°C to 70°C. Around 5% of the total objects (or task units) needed to be migrated to achieve the balance. This results in a 7% performance overhead since the applications balance is distorted in order to create the temperature balance. Doing temperature balancing can allow the preemptive fan speed to be set at a higher level. In this particular case, reduction of 1°C in the maximum temperature only gives minimal benefit in terms of the fan power.

As we show earlier in Figure 12, within-chip temperature variation can peak up to 10°C . Load balancing may be beneficial only if the temperature difference is greater than or equal to 6°C. That still represents a significant portion, 41%, of the chips. For those chips, the average difference between the maximum core temperature and the average core temperature is 3°C. 3°C reduction in maximum temperature in those chips reduces the peak fan power further by 3.3% and hence the average reduction in fan power becomes 56.6% compared to reactive fan control. This strategy comes at a cost of performance overhead of up to 10%. Given its marginal benefit and its performance overhead, this load balancing method may not be advisable if the CPU is at peak load like in our case. If the CPU is not fully utilized to their capacity (i.e. when the application application behavior creates the imbalance), such thermal-aware mechanisms could be more useful. For example, when running at large scale, some applications use less cores than the number of physical cores in the node because of running out of memory. For a second example, some applications require to launch on power of two number of processes for performance reasons and some cores may have to be left idle. We have considered the worst-case scenario in our experiments.

VI. Related Work

The neural network modeling approach has been gaining popularity in multiple areas of data center resource management. It has been used to do power and energy modeling of HPC kernels with different code variants [20], to predict user demand from the usage history to be able to turn servers on or off [21] and to build an online thermal mapping system for data centers at the level of one or more servers [22], [23]. Their prediction approach is coarse-grained, i.e. at the level of one or more servers, and cannot predict individual core temperatures. Another study uses neural networks to predict the core and Network-on-Chip component temperatures of the chip, and the predictions are used for energy efficient data exchange [24]. However, they heavily rely on the simulated data to evaluate their model, and their neural network models are not thoroughly validated in the literature. Other machine learning models have also been used to predict data center temperatures [25]. Although neural networks have shown many advantages over statistical methods, due to their capability in handling nonlinear behavior, other temperature prediction models can also be used to do proactive cooling mechanisms proposed in this paper.

Recent research supports the existence of thermal and manufacturing variations in supercomputing systems [25], [26], [27]. Thermal-aware job scheduling strategies have been proposed to cope with this problem. However, since variations can be observed even within one node, workload scheduling strategies would not be sufficient to address the temperature variations especially for large-scale HPC applications. Moreover, HPC applications may have communication patterns that allow them to benefit from topology-aware mapping strategies and job topology requirements can conflict with thermal-aware job scheduling techniques.

A forward-looking fan controller, where the system uses system utilization information to predict temperatures and controls the fan to dampen power peaks has been patented [28]. However, the effectiveness of the mechanism has not been published.

VII. Conclusion

In this paper, we first analyze inefficiencies in air-cooling systems such as oscillations in fan speed and temperature variations. We show how proactive cooling mechanisms can be used to mitigate these inefficiencies and reduce cooling fan power by 53.3% and energy by 22.4% without any performance overhead. These solutions cannot be used in production without an accurate temperature prediction model. Yet thermal-aware methods are often applied greedily. In this work, as a proof-of-concept model, we use a neural-network based approach that can predict steady-state core temperatures under different workloads, frequencies, and fan speed levels.

REFERENCES

[1] ASCAC Subcommittee, "Top ten exascale research challenges," *US Department Of Energy Report*, 2014.

[2] T. Patki, N. Bates, G. Ghatikar, A. Clausen, S. Klingert, G. Abdulla, and M. Sheikhalishahi, "Supercomputing centers and electricity service providers: A geographically distributed perspective on demand management in europe and the united states," in *International Conference on High Performance Computing*. Springer, 2016, pp. 243–260.

[3] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, "Exploring hardware overprovisioning in power-constrained, high performance computing," in *Proceedings of the 27th international ACM conference on International conference on supercomputing*. ACM, 2013, pp. 173–182.

[4] O. Sarood, A. Langer, A. Gupta, and L. Kale, "Maximizing throughput of overprovisioned hpc data centers under a strict power budget," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2014, pp. 807–818.

[5] A. Marathe, P. E. Bailey, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, "A run-time system for power-constrained hpc applications," in *International Conference on High Performance Computing*. Springer International Publishing, 2015, pp. 394–408.

[6] L. Wang, S. U. Khan, and J. Dayal, "Thermal aware workload placement with task-temperature profiles in a data center," *The Journal of Supercomputing*, vol. 61, no. 3, pp. 780–803, 2012.

[7] W. Huang, C. Lefurgy, W. Kuk, A. Buyuktosunoglu, M. Floyd, K. Rajamani, M. Allen-Ware, and B. Brock, "Accurate fine-grained processor power proxies," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, 2012, pp. 224–234.

[8] E. K. Lee, H. Viswanathan, and D. Pompili, "Vmap: Proactive thermal-aware virtual machine allocation in hpc cloud datacenters," in *High Performance Computing (HiPC), 2012 19th International Conference on*, 2012, pp. 1–10.

[9] O. Sarood, E. Meneses, and L. V. Kale, "A 'Cool' Way of Improving the Reliability of HPC Machines," in *Proceedings of The International Conference for High Performance Computing, Networking, Storage and Analysis*, Denver, CO, USA, November 2013.

[10] Z. Wang, C. Bash, N. Tolia, M. Marwah, X. Zhu, and P. Ranganathan, "Optimal fan speed control for thermal management of servers," *Proc. IPAC*, pp. 1–10, 2009.

[11] H. Demuth and M. Beale, "Neural Network Toolbox for Use with MATLAB," MathWorks, 2017, http://www.mathworks.com/help/nnet/.

[12] J. C. Phillips, G. Zheng, S. Kumar, and L. V. Kalé, "NAMD: Biomolecular simulation on thousands of processors," in *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, 2002, pp. 1–18.

[13] D. Carraway, "lookbusy – a synthetic load generator," 2017, https://www.devin.com/lookbusy/.

[14] J. J. More, "The levenberg-marquardt algorithm: Implementation and theory," *Numerical Analysis, ed. G. A. Watson, Lecture Notes in Mathematics 630, Springer Verlag*, pp. 105–116, 1977.

[15] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *NEURAL NETWORKS*, vol. 6, no. 4, pp. 525–533, 1993.

[16] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the rprop algorithm," in *Neural Networks, 1993., IEEE International Conference on*, 1993, pp. 586–591.

[17] Z. Toprak-Deniz, M. Sperling, J. Bulzacchelli, G. Still, R. Kruse, S. Kim, D. Boerstler, T. Gloekler, R. Robertazzi *et al.*, "5.2 distributed system of digitally controlled microregulators enabling per-core dvfs for the power8 tm microprocessor," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, Feb 2014, pp. 98–99.

[18] E. A. Burton, G. Schrom, F. Paillet, J. Douglas, W. J. Lambert, K. Radhakrishnan, and M. J. Hill, "Fivrfully integrated voltage regulators on 4th generation intel® core socs," in *Applied Power Electronics Conference and Exposition (APEC), 2014 Twenty-Ninth Annual IEEE*. IEEE, 2014, pp. 432–439.

[19] B. Acun, A. Gupta, N. Jain, A. Langer, H. Menon, E. Mikida, X. Ni, M. Robson, Y. Sun, E. Totoni *et al.*, "Parallel programming with migratable objects: charm++ in practice," in *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2014, pp. 647–658.

[20] A. Tiwari, M. A. Laurenzano, L. Carrington, and A. Snavely, "Modeling power and energy usage of hpc kernels," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE, 2012, pp. 990–998.

[21] T. V. T. Duy, Y. Sato, and Y. Inoguchi, "Performance evaluation of a green scheduling algorithm for energy savings in cloud computing," in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 1–8.

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

[23] L. Wang, G. von Laszewski, F. Huang, J. Dayal, T. Frulani, and G. Fox, "Task scheduling with ann-based temperature prediction in a data center: a simulation-based study," *Engineering with Computers*, vol. 27, no. 4, pp. 381–391, 2011.

[24] S. Aswath Narayana, "An artificial neural networks based temperature prediction framework for network-on-chip based multicore platform," Master's thesis, Rochester Institute of Technology, 2016.

[25] K. Zhang, S. Ogrenci-Memik, G. Memik, K. Yoshii, R. Sankaran, and P. Beckman, "Minimizing thermal variation across system components," in *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*. IEEE, 2015, pp. 1139–1148.

[26] Y. Inadomi, T. Patki, K. Inoue, M. Aoyagi, B. Rountree, M. Schulz, D. Lowenthal, Y. Wada, K. Fukazawa *et al.*, "Analyzing and mitigating the impact of manufacturing variability in power-constrained supercomputing," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015, p. 78.

[27] B. Acun, P. Miller, and L. V. Kale, "Variation among processors under turbo boost in hpc systems," in *Proceedings of the 2016 International Conference on Supercomputing*. ACM, 2016, p. 6.

[28] D. Tobias, "Forward-looking fan control using system operation information," Feb. 7 2006, US Patent 6,996,441.