PredictNcool: Leakage Aware Thermal Management for 3D Memories Using a Lightweight Temperature Predictor

LOKESH SIDDHU and PREETI RANJAN PANDA, Department of Computer Science and Engineering, Indian Institute of Technology Delhi

Recent research on mitigating thermal problems in 3D memories has covered reactive strategies that reduce memory power consumption, and thereby, performance, when the memory temperature reaches the maximum operating limit. Such techniques could benefit from temperature prediction and avoid unnecessary invocations and state transitions of the thermal management strategy. We develop an accurate steady state temperature predictor for thermal management of 3D memories. We utilize the symmetries in the floorplan, along with other design insights, to reduce the predictor's model parameters, making it lightweight and suitable for runtime thermal management. Using the temperature prediction, we introduce *PredictNcool*, a proactive thermal management strategy to reduce application runtime and memory energy. We compare *PredictNcool* with two recent thermal management strategies and our experiments show that the proposed optimization results in performance improvements of 28% and 5%, and memory subsystem energy reductions of 38% and 12% (on average).

CCS Concepts: • **Hardware → Temperature control**; *3D integrated circuits*; *Temperature simulation and estimation*;

Additional Key Words and Phrases: 3D memories, thermal management, energy efficiency

ACM Reference format:
Lokesh Siddhu and Preeti Ranjan Panda. 2019. PredictNcool: Leakage Aware Thermal Management for 3D Memories Using a Lightweight Temperature Predictor. *ACM Trans. Embed. Comput. Syst.* 18, 5s, Article 64 (October 2019), 22 pages.
https://doi.org/10.1145/3358208

1 INTRODUCTION

benefits such as smaller form factor and lower interconnect power. Such memories offer several benefits, other than high bandwidth, such as smaller form factor and lower interconnect power. Current trends show that the conventional DRAM process scaling, which has been a critical enabler for higher performance, is slowing down. To meet the bandwidth requirements of modern applications, researchers have proposed stacking 2D memories to realize a 3D memory. In addition to higher bandwidth, such memories also offer other benefits such as smaller form factor and lower interconnect power. However, they suffer from thermal issues arising out of higher power density.

This article appears as part of the ESWEEK-TECS special issue and was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2019.
Authors' address: L. Siddhu and P. R. Panda, Department of Computer Science and Engineering, Indian Institute of Technology Delhi, Hauz Khas, New Delhi, Delhi, 110016; emails: {siddhulokesh, panda}@cse.iitd.ac.in.

1539-9087/2019/10-ART64 $15.00
https://doi.org/10.1145/3358208

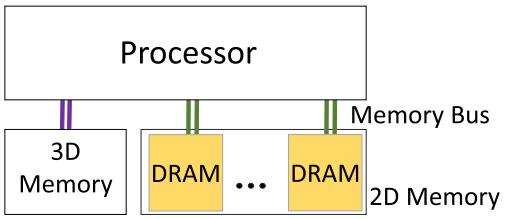

Fig. 1. Processor with off-chip 3D and 2D memories.

3D memories frequently need to be stalled to avoid overheating. Obtaining higher performance and lower energy under thermal constraints is a significant challenge for 3D memories [27]. To meet the temperature constraints, researchers have proposed various policies to reduce dynamic and static power consumption. Dynamic power is dissipated when there is switching in the memory, whereas static/leakage power is consumed whenever the memory is turned ON. Moreover, at high temperatures, memory static power increases, creating a positive feedback loop [37], and is comparable to the dynamic power [18]. Thus, the thermal management policies for memories need to be leakage-aware.

In this paper, we consider architectures containing both 2D and 3D memories as illustrated in Figure 1. Such architectures utilize the 2D memory to alleviate the high cost per bit and thermal issues of 3D memories [14]. Traditionally, the dynamic thermal management (DTM) strategies for these architectures have been reactive [26], [31]. They reduce memory power consumption when the 3D memory temperature is nearing the limit of maximum operating temperature. The DTM policies have multiple thermal emergency levels/states and implement stringent power reduction techniques at higher temperatures. However, such a reactive approach results in frequent invocation of cooling periods or changing of DTM level. On the other hand, a proactive thermal management approach can predict the future temperature, invoke the appropriate DTM level, and minimize the number of cooling periods. However, a major challenge in deploying a proactive approach is the design of an accurate runtime temperature predictor.

In this work, we first design a low-overhead, accurate runtime temperature predictor using linear regression for 3D memories, and then use this predictor to develop a proactive thermal management strategy called PredictNcool. PredictNcool maximizes the application performance and minimizes the memory energy consumption under thermal constraints for 3D memories. An accurate temperature predictor for 3D memories would require modeling a large number of parameters due to many layers within a 3D memory. However, we utilize the floorplan symmetry and other design insights to significantly reduce the model complexity and design a novel linear steady-state temperature predictor for 3D memories. The proposed predictor is able to accurately estimate steady state temperatures even if some memory channels are turned off to avoid memory heating.

Using symmetry and other insights derived from the thermal behaviour, we were able to reduce the number of model parameters significantly (by a factor of 362); this reduces training time, prediction time, and prediction data, and makes it tenable for runtime temperature prediction (with a low prediction time of 12 μs) for 3D memory architectures. Our experimental results show that the proposed temperature predictor is highly accurate (average error of 0.39°C, between 45°C and 100°C). Experimental evaluation of PredictNcool using SPEC CPU2006 [16] workloads demonstrates that the designed steady-state temperature predictor can be effectively utilized to reduce both the memory subsystem energy by 38% and 12%, and the application runtime by 28% and 5% in comparison with state-of-the-art DTM strategies [26] and [31] respectively. In summary, this paper makes the following key contributions:

(1) We develop a lightweight and accurate steady-state temperature predictor for 3D memories which can be used at runtime. To the best of our knowledge, this is the first work exploiting various designer insights about the layout/floorplan to reduce model parameters of a machine learning-based predictor, which consequently reduces the required training data and also the prediction time.
(2) Using the proposed predictor, we develop a proactive DTM policy for 3D memory architectures resulting in lower memory energy consumption and improved application performance. To the best of our knowledge, this is the first proactive DTM policy proposal for 3D memory architectures.

The rest of this paper is organized as follows. Section 2 discusses the related prior work. We explain the motivation for temperature prediction for 3D memories in Section 3. Our proposal for designing a temperature predictor and a proactive DTM strategy is explained in Section 4. Section 5 demonstrates experimental results on various standard benchmarks. We conclude the paper and indicate future directions in Section 6.

2 RELATED WORK

Thermal-aware design of processors and memories [1, 3, 4, 12, 17, 25, 34–36] has been a topic of research since the early/mid-2000s [5]. Thermal management strategies can be divided into static (design-time approach) and dynamic (runtime approach). Static methods rely on floorplanning [11] based on temperature profiling at the design-time. Dynamic thermal management (DTM) policies could be further classified into reactive (based on current temperature) or proactive (additionally, utilizes the predicted temperature).

2.1 Reactive Thermal Management

DTM strategies reduce heating by either re-mapping memory accesses to colder regions or by decreasing the access rate of function units through throttling or dynamic voltage/frequency scaling (DVFS) [15, 29, 33]. DTM schemes incur performance and energy overheads, and limiting such overheads is an active area of research [23, 26, 31, 41]. Recent works have addressed the thermal management of 3D processors [23, 24, 41]. Liao et al. [24] propose a combined approach of task scheduling and DVFS. For 3D Networks-on-Chip, Zheng et al. [41] suggest adjusting the bus frequency and routing strategy, and Lee et al. [23] propose using DVFS to limit the maximum temperature. However, relatively few efforts have addressed thermal issues in 3D memories. Lo et al. [26] discuss a thermal-aware page allocation strategy where allocation requests to a hot channel are reallocated to the coldest channel. FastCool [31] is a leakage aware DTM policy which migrates data of the hot channels from 3D to 2D memory and then switches the hot channels OFF to reduce leakage power. However, these strategies are reactive in nature and incur frequent cooling periods.

2.2 Proactive Thermal Management

Thermal modeling (transient and steady state) for planar/2D and 3D architectures has received significant attention from academia and industry [6, 13, 32, 40]. To reduce the runtime of thermal simulations, researchers have proposed training temperature prediction models using machine learning techniques. Kumar et al. [22] suggest a linear neural network (NN) model for MPSoCs which predicts transient temperatures based on the current temperature and power. The NN is implemented on a dedicated hardware and used at runtime. Juan et al. [21] present a design-time transient temperature predictor for Chip MultiProcessors (CMPs) using a linear regression (LR) model. Subsequently, Juan et al. [20] show that for 3D CMPs the steady-state temperature can be

accurately predicted at design time using a linear model (weighted summation) of leakage power even in the presence of process variation.

A runtime proactive DTM scheme for Multi-Processor System on Chips (MPSoCs) was suggested by Coskun et al. [10]. They use an autoregressive moving average of temperature to predict transient temperature. Ayoub et al. [2] propose an RC model based transient temperature prediction for improving the energy efficiency of memories. Cochran et al. [9] reduce the temperature for processors using thermal-aware DVFS for each workload phase, which is detected using performance counters. Zhang et al. [38] study various machine learning models (and select Gaussian process model) to predict the transient temperature of an HPC system.

2.3 Limitations of Prior Work

Prior works on temperature management [26, 31] of 3D memories do not use temperature prediction, which could help reduce both performance and energy overheads of DTM schemes. Among the existing proactive approaches, prior art in the temperature prediction models do not take into account the symmetries in the floorplan (chip geometry). Many of the predictors use power traces obtained using current measurements, which are difficult to obtain at fine granularity at runtime. Further, most of the proactive DTMs [2, 10, 21, 22, 38, 38] rely on transient temperature prediction which requires information about the current temperature and the access/power profile, thereby making the predictor complex. Moreover, effective use of the steady-state predictor (which needs only the access profile) could make the predictor/DTM lightweight. Additionally, many thermal management policies do not consider the effect of the thermal-leakage power loop, and consequently suffer from frequent heating and cooling phases. Especially, in case of 3D memories, there is a significant leakage power dissipation at high temperature [31]. Based on these limitations of prior work, we propose PredictNcool, a proactive DTM strategy for 3D memories which utilizes a novel leakage aware symmetry-based linear steady-state temperature predictor (using memory access counts) to reduce both energy and performance overheads of DTM.

3 BACKGROUND AND MOTIVATION

In this section, we first present some background on the thermal characteristics of 3D memories, establish the applicability of linear models for steady state temperature prediction, and motivate the benefit of a proactive thermal management policy compared to a reactive policy.

Thermal Characteristics of 3D Memories: Figure 2 shows a typical 1 GB, 3D stacked memory with 8 memory layers (128 MB each) [19]. Each layer of the 3D memory is divided into 16 partitions (or *ranks*). A memory *channel* is a vertical stack of ranks. A heat sink is placed on top of the structure to improve heat dissipation. The layers closer to the heat sink have a lower temperature as they have better heat dissipation capability than lower layers which are further away. Since the primary heat dissipation path is through the heat sink, the bottom layer has the highest temperature. Within a layer, under uniform power dissipation, the central partitions/ranks get heated from all the sides and exhibit the highest temperatures.

Thermally dependent leakage power and linear steady state temperature prediction: Linear steady-state temperature predictors estimate temperature as a weighted sum of power dissipation/accesses. Prior works have experimentally established that a linear approximation of leakage power works well for estimation of 3D processor temperature within the typical range of operating temperature [20]. We performed experiments to study the possibility of using a linear predictor for 3D memory, considering both dynamic and leakage power (memory power modeling was done using CACTI-3DD [8] and is similar to [31]). Figure 3 shows the actual and estimated temperatures for Rank 0 (left bottom corner) on the bottom layer using linear prediction (details of the formulation are given in Section 5). The correlation coefficient is 0.998, and the average error is 0.36°C.

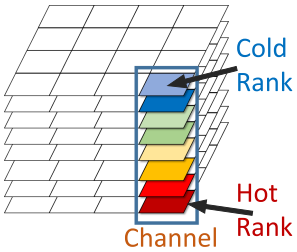

Fig. 2. 8-layer, 16-channel 3D memory. Bottom layer are hottest.

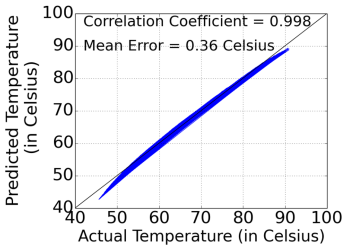

Fig. 3. Linear steady state temperature prediction for a 3D memory.

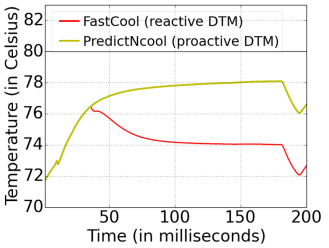

Fig. 4. Reactive versus proactive DTM strategy for bwaves.

The other ranks showed similar trends), which motivates the argument that linear predictors can be used to predict steady-state temperature for 3D memories in the operating temperature range. We also implemented non-linear predictors such as neural networks (with one hidden layer of size 50) to estimate the 3D-memory steady-state temperature profile, and observed an average error of 0.35°C, leading to an average error difference of only 0.01°C compared to linear predictor; thus, we use linear prediction owing to its simplicity.

Proactive thermal management: We compare the effect of using a recent reactive DTM policy (FastCool) [31] versus our proposed proactive DTM policy (PredictNcool) on the performance of the system. For this purpose, we use *bwaves*, a SPEC CPU2006 benchmark (more experimental details are given in Section 5). Once the temperature crosses a threshold (76.4°C), FastCool triggers DTM to reduce the temperature of the 3D memory. However, as shown in Figure 4, there was no need to trigger DTM as the temperature would never cross the allowed limit of 80°C [31]. If a steady state temperature predictor could provide this information up front, a proactive strategy would never trigger DTM and thus prevent performance reduction due to the unnecessary invocation of DTM - this motivates the need for using a proactive DTM policy. With this motivation, we present our proposed approach for steady-state temperature estimation and PredictNcool strategy in the next section.

4 OUR PROPOSAL

In this section, we discuss the design of a temperature predictor for 3D memories that effectively uses various designer insights to reduce the complexity of the model. We use this predictor in a proactive DTM strategy named PredictNcool, which improves both the energy consumption and performance. We also present an energy model to analyze the energy consumption of a memory subsystem with both 2D and 3D memories.

4.1 Design of a Temperature Predictor

In this section, we first discuss the design of a simple temperature predictor, and subsequently, demonstrate various designer insights for a 3D memory and systematically incorporate them in the predictor to reduce the model size.

4.1.1 A Leakage Aware, Steady-state Temperature Predictor. Power dissipation and steady-state temperature in 3D memories depend on the dynamic activity (access rate) and the static power consumption (when the memory channels are ON). Figure 5 shows the high-level view of a simple steady-state temperature predictor. Such a model can estimate the temperature even in the presence of leakage aware DTM strategies [31] applied at channel level granularity. We use a linear regression based predictor as it results in high accuracy in the operating range, as motivated in

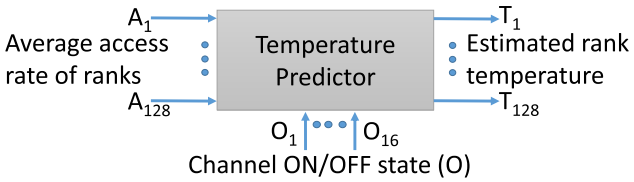

Fig. 5. Steady-state temperature predictor for 3D memory.

Section 3. The rank temperature (T) is estimated as a weighted sum of rank access rates (A) and channel ON/OFF states (O). O is 1 if the channel is turned ON, else 0. We estimate the temperature at rank level granularity, as most DRAM architectures use rank as the minimum controllable unit for power reduction.

For an 8 layer memory with 16 (= 4 × 4) ranks per layer (Figure 2), the temperature for a rank in position (p, q, r) is given by:

$$T_{(p,q,r)} = \underbrace{\sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{8} W_{(i,j,k)(p,q,r)} \times A_{(i,j,k)}}_{\text{weighted rank accesses}} + \underbrace{\sum_{i=1}^{4}\sum_{j=1}^{4} L_{(i,j)(p,q,r)} \times O_{(i,j)}}_{\text{weighted channel ON/OFF}} + \underbrace{C_{(p,q,r)}}_{\text{constant term}}$$

Subscripts (i, j, k) indicate the rank location (similar to Cartesian coordinates); i and j indicate the 2D (x-y) position in a layer and k represents the layer number. For example, the left corner rank of the bottom layer is located at (i = 1, j = 1, k = 1) and the top right corner rank of the eighth layer has coordinates (4,4,8). In the above equation, the rank temperature is predicted for the location (p, q, r) and for each location/rank in the 3D memory, fitting coefficients (W, L, C) need to be learned. Since the weights L correspond to channel turn ON/OFF status, the L values do not use k. Thus, from the above equation, 145 (= 4 × 4 × 8 + 4 × 4 + 1) parameters are learned per rank and a total of 18432 (= 145 × 128) parameters are learned for the 128 ranks.

To learn the fitting coefficients, we initially create an offline database of steady-state rank temperatures (obtained using multiple HotSpot [39] simulations) collected for inputs that exercise various random access rates for different ranks. Each pair of input (A, O) and output (T) is referred to as a data point. The coefficient learning process consists of a training phase and a test phase. The training phase determines the coefficients (W, L, C) to minimize error. To implement the training algorithm, we used an off-the-shelf machine learning library, *scikit* [30] (sklearn.linear_model.LinearRegression), which minimizes the root mean square error. Next, in the test phase, the learned coefficients are used to estimate $T_{(p,q,r)}$; high accuracy on the test database is expected for a well-trained model. We use 70% of our input database for training and rest 30% for testing and perform ten-fold cross-validation (similar to [21]) so that the training results generalize to an independent data set. As mentioned above, we use a learning-based approach to determine the weights. Alternatively, we also considered obtaining the weights using the resistance/conductance matrix used internally by HotSpot. However, we observed that HotSpot's resistance matrix does not include modeling for temperature dependent leakage power and uses iterative methods to obtain steady state temperatures in such cases. Hence, we use a learning-based framework to obtain the weights (similar observation was made in [22]). Moreover, HotSpot does not utilize the symmetries in the floorplan which are proposed and used in this work to significantly bring down the prediction time.

We study below the effect of increasing database size on the coefficients learned by the predictor. At 2000 (2K) data points per rank, we found the linear predictor accurately estimated the steady-state temperatures with an average absolute error of 0.371°C (Table 1). Figure 6(a) shows

	$i=1$	$i=2$	$i=3$	$i=4$
$j=4$	-0.046	-0.269	0.221	-0.092
$j=3$	-0.135	-0.003	0.068	0.121
$j=2$	0.782	0.487	0.125	0.160
$j=1$	**2.685**	0.532	0.117	0.245

(a) 2K data points

$i=1$	$i=2$	$i=3$	$i=4$
0.101	0.087	0.068	0.060
0.189	0.136	0.090	0.068
0.664	0.323	0.133	0.082
2.504	0.665	0.186	0.092

(b) 5M data points

Fig. 6. Bottom layer weights for predicting steady state temperature of the left bottom rank corner.

Table 1. Properties of Learned Weights with Varying Database Size

Data Points[1]	Mean Absolute Test Error	Positive Weights	Decreasing Weights	Symmetry
2K	0.37°C	✗	✗	✗
20K	0.36°C	✓	✗	✗
5 Million	0.36°C	✓	✓	✗
8 Million	0.36°C	✓	✓	Almost

[1]Each data point is obtained using a HotSpot steady-state thermal simulation.

the learned weights (W) for the bottom layer when the temperature for the corner left bottom rank (shown in bold) is predicted[1]. Coefficients (L, C) are not shown in Figure 6(a) for brevity. We observe that most of the weights are positive numbers; however, some of them are negative (e.g., at location (1,4,1) the value is -0.046). We also observe that the learned weights do not follow any ordering as we move away from the location (1,1,1) for which prediction is being done. However, these observations do not match our prior understanding of the thermal behavior. As we increase the database size, we expect the following properties to hold true for the learned coefficients/weights.

Insight 1: Positive cross-coupling between access rate and temperature. Since the weights represent the cross-coupling between the access rate and temperature, an access to any rank cannot lead to a reduction in temperature. We have: $W_{(i,j,k)(p,q,r)} \geq 0$.

Insight 2: Larger the distance, lower the weight. As we go further away from location (p, q, r) for which the temperature is being predicted, the cross-coupling decreases. Hence, weights should decrease as we move away from the location (p, q, r). We have:

$$\begin{aligned}
\text{if } p < i, W_{(i,j,k)(p,q,r)} &> W_{(i+1,j,k)(p,q,r)} \\
\text{else } W_{(i,j,k)(p,q,r)} &< W_{(i+1,j,k)(p,q,r)} \\
\text{if } q < j, W_{(i,j,k)(p,q,r)} &> W_{(i,j+1,k)(p,q,r)} \\
\text{else } W_{(i,j,k)(p,q,r)} &< W_{(i,j+1,k)(p,q,r)} \\
\text{if } r < k, W_{(i,j,k)(p,q,r)} &> W_{(i,j,k+1)(p,q,r)} \\
\text{else } W_{(i,j,k)(p,q,r)} &< W_{(i,j,k+1)(p,q,r)}
\end{aligned}$$

Figure 6(b) shows the bottom layer learned access rate weights (W) for 5 Million data points. We observe that the temperature predictor learns the above two properties after sufficiently

[1]The figures showing the weights assume access rate A to be in MBps.

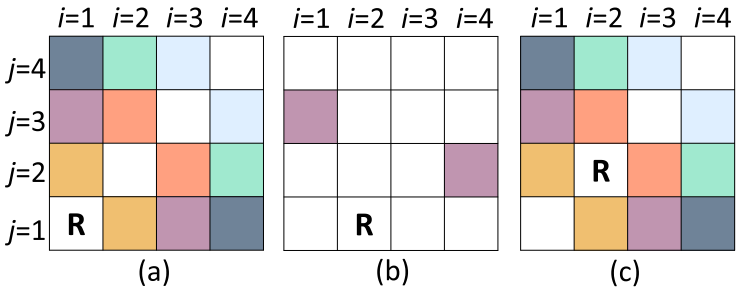

Fig. 7. Symmetric ranks with respect to a rank R are shown in the same color (except white - no symmetry).

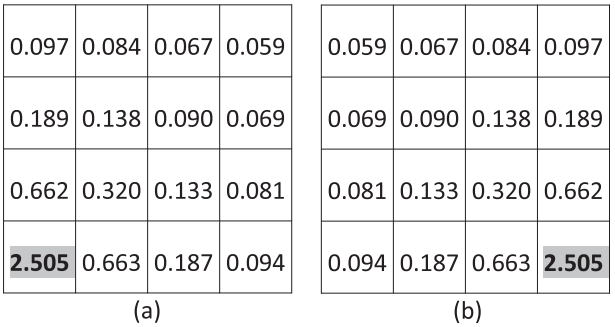

Fig. 8. Bottom layer weights using 8M datapoints - (a) for ranks (1,1,1), (b) for rank (4,1,1).

large training data is provided. Even though the other layers are not shown, they also follow the conditions mentioned above. Although we traditionally stop the training process upon achieving high training accuracy (low training error), Table 1 shows that this may not ensure that system properties are learned by the predictor. Also, extensive training data is required for the learned weights to capture the above properties. Due to symmetry in the floorplan of 3D memories and the nature of cross-coupling, various symmetries exist in the fitting coefficients (W, L, C).

Symmetry 1: Effect of symmetrically located ranks with respect to a given rank. While predicting the temperature for a specific rank R, various weights have the same value due to their symmetrical location with respect to R. For example, in Figure 7(a), ranks (1,2,1) and (2,1,1) are located symmetrically with respect to (1,1,1). Thus, while estimating the temperature for the rank (1,1,1), the weights of ranks (1,2,1) and (2,1,1) should be equal. These weights are multiplied by the respective access rates to determine the effect on temperature.

$$W_{(1,2,1)(1,1,1)} = W_{(2,1,1)(1,1,1)}$$

Figure 7(a) also highlights other symmetrically located ranks with respect to rank (1,1,1). Symmetric ranks are shown in the same color. In general, we observe a transpose symmetry in the weights:

$$W_{(i,j,k)(1,1,1)} = W_{(j,i,k)(1,1,1)} \text{(transpose symmetry)}$$

Figures 7(b) and 7(c) show equal weights (color coded) while predicting the temperatures for locations (2,1,1) and (2,2,1).

Symmetry 2: Temperature prediction for symmetrically located ranks in a layer. Consider, as an example, the base layer weights (for 8 Million data points) of the corner locations (1,1,1) and (4,1,1), shown in Figure 8. We observe that the weights for the location (4,1,1) can be easily derived from rank (1,1,1) based on symmetry. Similarly, other corner ranks (1,4,1) and (4,4,1) could also reuse

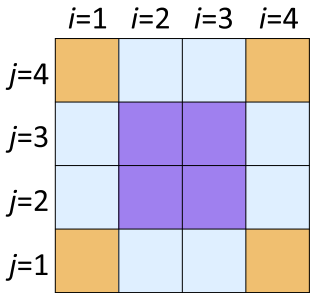

Fig. 9. Symmetrically located ranks in a layer.

0.084	0.082	0.067	0.059
0.120	0.113	0.111	0.078
0.297	0.451	0.246	0.116
0.662	**2.040**	0.547	0.165

Fig. 10. Bottom layer weights for rank (2,1,1).

the weights of the rank (1,1,1). Figure 9, shows the general picture of locations at which weights can be re-used. Essentially, at each layer, we need to learn weights for only three locations {(1,1), (2,1), (2,2)}; the remaining locations could reuse the weights due to symmetry.

Symmetry 3: Symmetry in cross-coupling. Figures 8(a) and 10 show the base layer weights of ranks (1,1,1) and (2,1,1). Due to symmetry, the cross-coupling/weight (= 0.663) of (2,1,1) when predicting temperature at location (1,1,1) is expected to equal the weight (= 0.662) of (1,1,1) when predicting temperature at location (2,1,1). We have: $W_{(2,1,1)(1,1,1)} = W_{(1,1,1)(2,1,1)}$. Essentially, cross-coupling (weight) seen from either end is symmetrical.

$$W_{(i,j,k)(p,q,r)} = W_{(p,q,r)(i,j,k)}$$

As shown in Table 1, a large amount of training data is required for automatically learning the symmetry properties. However, even after using 8M data points (which required ~72 days to generate the training and test database using HotSpot's steady state simulations) we noticed that machine learning algorithms could learn symmetry up to two decimal places. Having shown that the coefficients exhibit symmetry, we explain how to incorporate symmetry in the learning process and reduce the model parameters in the next section.

4.1.2 Using Symmetry in the Learning Process. Symmetry can be easily used in the learning process to reduce the model parameters. Inputs with equal weights could be added together, and a single weight could be learned. For example, for rank (1,1,1), we have:

$$T_{(1,1,1)} = W_{(1,2,1)(1,1,1)} * A_{(1,2,1)} + W_{(2,1,1)(1,1,1)} * A_{(2,1,1)} + \text{other terms}$$

As discussed previously, as per *symmetry 1*,

$$W_{(1,2,1)(1,1,1)} = W_{(2,1,1)(1,1,1)} = W$$

$$\text{Hence, } T_{(1,1,1)} = W * (A_{(1,2,1)} + A_{(2,1,1)}) + \text{other terms}$$

The above equation, which is valid under symmetry, allows us to reduce the model size. Moreover, the symmetries discussed under *symmetry 1*, can be incorporated in the learning process using the above steps. Also, as per *symmetry 2*, coefficients for only three rank locations ((1,1), (2,1), (2,2)) per layer need to be trained. Coefficients for the remaining ranks can be easily derived from these values. Hence, symmetries discussed under *symmetry 2* could be easily included in the training process.

In cases where weights for certain input features are known (as in *symmetry 3*) before training a model, they can be easily dropped from the training process. For example, while training a temperature prediction model for rank (2,1,1) we could reuse some of the weights obtained during training a temperature prediction model for location (1,1,1). More specifically, the weight $W_{(2,1,1)(1,1,1)}$

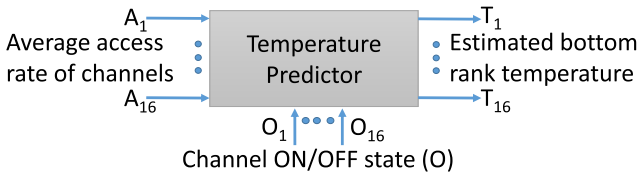

Fig. 11. Base layer steady state temperature prediction for a 3D memory.

Table 2. Summary of the Steady State Temperature Predictors for 3D Memories

Type of input	Predicted temperature	Data points	Mean absolute error	Model size	Remarks
Rank Level	For all ranks	8M	0.36°C	18432	Training using raw data
Channel Level	For base layer ranks	520K	0.39°C	528	Training using raw data
Channel Level	For base layer ranks	320K	0.39°C	51	Symmetry enforced explicitly

could be reused while training a model for (2,1,1) because, as per *symmetry 3*, $W_{(2,1,1)(1,1,1)}$ and $W_{(1,1,1)(2,1,1)}$ are equal.

Using the steps described above, we incorporated the above symmetries in the learning process and decreased the number of unique model parameters to ~2000, while still achieving an average error of 0.36°C. Thus, designer insights helped us to reduce the model size without loss of accuracy.

4.1.3 Moving from Rank Level to Channel Level Inputs and Maximum Temperature Prediction. In this section, we discuss insights regarding the location of the hottest ranrks and moving from rank level to channel level inputs for temperature prediction of 3D memory.

Insight 3: Lowest layer has the maximum temperature.

Many of the DTM strategies [26, 31] operate if the maximum temperature crosses a certain threshold. Thus, many strategies need the maximum temperature rather than the full thermal profile. We modify the developed predictor to behave as a predictor for maximum steady-state temperature. As mentioned in Section 3, in a 3D memory, the bottom layers have the highest temperatures because they are farthest from the heat sink. Thus, a predictor of maximum temperature needs to estimate temperatures only for Layer 1 (bottom layer) instead of the full thermal profile.

In our experiments and as shown in previous work [14], we observe that modern memory controllers usually attempt to distribute accesses uniformly across ranks in a channel, although different channels might have different access counts. This motivates us to build a channel level prediction method (instead of rank-level).

Using the above observations, we build a steady-state predictor shown in Figure 11. Using the estimated base layer temperatures, we can find the maximum 3D memory temperature and use it to improve energy efficiency of DTM strategies (discussed in Section 4.2). Moreover, such a predictor learns 33 (= 16 + 16 + 1) parameters per rank and a total of 528 (= 33 × 16) parameters are learned.

We trained such a model and observed that the predictor also followed the previously discussed symmetry properties. Next, we incorporated the symmetry properties into the learning process to reduce the model parameters and training data. We observe in Table 2 that moving to channel level access rate inputs introduces only a small error in prediction. In summary, using the design insights and symmetry, we are able to reduce the model parameters by 362X (= 18432/51), reducing training and prediction time (and data). The proposed predictor uses the channel access rate

and the ON/OFF state to estimate the temperature, and does not need re-training for every new workload/application.

Although the above-mentioned steady-state predictor works well for homogeneous 3D memories, extending it to asymmetric layouts is challenging. Asymmetries in the floorplan could occur due to process variation or possibly because of logic layers present in the memory stacks (amongst other causes). Addressing the above limitations would require further research.

4.2 Design of PredictNcool (PNC)

In this section, we discuss the design of PredictNcool (PNC), a proactive DTM policy for 3D memories. A simple proactive DTM strategy that runs every epoch of T time units, could be to reduce power consumption/performance only if the predicted maximum steady-state temperature (T_{SS}) is greater than the threshold (THR, such as 80°C). Although this policy works well initially, after a few tens of milliseconds, T_{SS} varies very slowly, as it depends upon the long-time average of the (entire) access trace which makes the predictor very sluggish and unable to respond to recent changes in the access rates.

To remove this sluggishness, we use the steady-state temperature prediction (T_{K_SS}) based on the access rates in the last K epochs (rather than the entire access trace). This makes the predictor respond to recent changes and also allows the DTM policy to act fast. However, using T_{K_SS} prediction alone might result in undesirable behavior when the current temperature (TC) is low; a short burst of activity might result in predicting a high value of T_{K_SS}. However, the DTM policy need not be invoked when TC is low. Therefore, we utilize T_{K_SS} prediction only when the current temperature crosses a certain threshold (TH, such as 78°C).

Based on the above observations, we reduce performance (or temperature) only when 3D memory gets heated either in the:

- Distant Future: This happens when the predicted T_{SS} is above a threshold, i.e., $T_{SS} > THR$, OR
- Near Future: This happens when the current temperature is high (above a threshold) and estimated T_{K_SS} is also above a threshold, i.e., $TC > TH$ AND $T_{K\_SS} > THR$

Combining,

$$(T_{SS} > THR) \text{ OR } (TC > TH \text{ AND } T_{K\_SS} > THR) \quad (1)$$

If the above condition is satisfied, the 3D memory will likely get heated past the threshold, and therefore, to reduce the temperature, we implement the FastCool [31] thermal management policy. As the temperature rises, it turns OFF more channels (leading to higher performance penalty) and moves to higher thermal emergency levels to avoid overheating (lines 3–4 in Algorithm 1). If it is estimated that memory will not cross the thermal threshold in the future (condition 1 is false), then PNC maintains the last DTM level until the memory cools down below THR_{COOL}. Then, we bring down the DTM level which helps in reducing the performance overhead (lines 5–6 in Algorithm 1). Note that even although PredictNcool uses the same DTM levels as FastCool, it additionally utilizes temperature prediction to decide on state transitions, thereby helping it lower application runtime and memory energy consumption.

Algorithm 1 outlines the complete strategy for PredictNcool to maximize performance and minimize the energy consumption under thermal constraints. Similar to previous work [26, 31], we assume that the maximum temperature constraint such as 80°C is specified. The algorithm runs every epoch of duration T (1 millisecond in our experiments) and regulates the transient temperature. PredictNcool can effectively take decisions based on the estimated near and distant future temperatures. Although many previous proactive DTM strategies (for cores) [2, 10, 38]

ALGORITHM 1: Algorithm for PredictNcool

Input: A: Mean Access Rate
Input: A_K: Mean Access Rate for last K epochs
Input: O: Channel ON/OFF State
Input: TC: Current Temperature

```
1 T_SS ← Predict_Steady_State_Temperature (A, O)
2 T_K_SS ← Predict_Steady_State_Temperature (A_K, O)
3 if (T_SS > THR) OR (TC > TH AND T_K_SS > THR) then
                                                  /* Likely to get heated */
4 |   Reduce temperature using chosen DTM policy
  else
                                                /* Unlikely to get heated */
5 |   if TC < THR_COOL then
                                                      /* Low temperature */
6 |   |   Reduce the thermal emergency level and improve performance
  |   end
  end
```

used transient temperature prediction (which is more complex than steady-state prediction), PredictNcool can utilize a maximum steady-state temperature predictor. Thermal predictions used by PredictNcool take into account both dynamic accesses and temperature-dependent static/leakage power. Also, PredictNcool is not based on any specific DTM policy and could be integrated with any reactive DTM strategy. Presently, we have used FastCool as it a recent DTM policy for 3D memories, and we experimentally show that it can benefit in terms of reducing both energy and performance overheads from proactiveness (see Section 5). Moreover, unlike reactive strategies which gradually increase the DTM level (step-by-step) in reaction to rising temperatures, PredictNCool can predict if temperatures would exceed the threshold and if so, activates the appropriate DTM level avoiding unnecessary intermediate transitions. The temperature thresholds (THR, TH, and THR_{COOL}) used in PredictNcool are determined through experimentation.

4.3 Memory System Energy Modeling

In this section, we present an energy model for a 2D + 3D memory system, which we later use to quantify the energy consumed by various DTM strategies. The energy modeling takes into account the effect of temperature-dependent static (or leakage) energy consumption in 3D memories and data migration (if any) between 2D and 3D memories.

In the memory system, we consider three components: 2D memory, 3D memory, and the bus. The memory system energy (E) can be divided into dynamic (E_D) and static (E_S). The dynamic energy (E_D) is consumed during regular data access (E_A) to/from memory and also, during data migration (E_M). Thus:

$$E = E_D + E_S \tag{2}$$

$$E_D = E_A + E_M \tag{3}$$

Access Energy: Given the access count (A) and energy-per-access (EA) for a component, the access energy = $A \times EA$. Hence, the total access energy is given by:

$$E_A = A_{3D} \times EA_{3D} + A_{2D} \times EA_{2D} + A_{BUS} \times EA_{BUS} \tag{4}$$

with the parameters defined in Table 3.

Table 3. Parameter Definitions for Energy Modeling

Symbol	Description
T	Time for an epoch
T_E	Execution time[1]
C	Cache line size
S	Size of migrated data[1]
3D Memory Parameters	
A_{3D}	Number of access[1]
EA_{3D}	Energy-per-access
P_{3D}	Memory power[1]
2D DRAM Parameters	
A_{2D}	Number of access[1]
EA_{2D}	Energy-per-access[1]
P_{S_2D}	Static power (per channel)
N_{2D}	2D memory channels
LR_{A_2D}	Leakage reduction factor (active mode)[1]
LR_{I_2D}	Leakage reduction factor (idle mode)[1]
T_{A_2D}	Time spent in active mode[1]
T_{I_2D}	Time spent in idle mode[1]
Bus Parameters	
A_{BUS}	Number of access[1]
EA_{BUS}	Energy-per-access
P_{S_BUS}	Static power (per channel)

[1] *Runtime* – Values are determined at runtime.

Migration Energy: If S is the size of migrated data and C is the cache line size, then migration energy = $S \times EA/C$ (memory access granularity is one cache line). Thus, the total migration energy is:

$$E_M = S \times EA_{3D}/C + S \times EA_{2D}/C + S \times EA_{BUS}/C \tag{5}$$

Static Energy: This is the energy dissipated when a component is powered ON but idle. If T_E represents the execution time, then the static energy = $P_S \times T_E$. We use this equation for different components (discussed below). The total static energy (E_S) is:

$$E_S = E_{S\_BUS} + E_{S\_2D} + E_{S\_3D} \tag{6}$$

Bus: The bus static power (P_{S_BUS}) is usually specified for a single 2D memory channel. Hence, for N_{2D} channels (2D memory) we have:

$$E_{S\_BUS} = P_{S\_BUS} \times N_{2D} \times T_E \tag{7}$$

2D Memory: Since memories consume significant static power, various prior works suggested techniques to reduce it. We model this reduction using a leakage reduction factor (LR). Most strategies reduce static energy by differing amounts in active and idle modes of operation. Thus:

E_{S_2D} = static energy when memory is active + static energy when memory is idle

$$E_{S\_2D} = P_{S\_2D} \times N_{2D} \times T_{A\_2D} \times LR_{A\_2D} + P_{S\_2D} \times N_{2D} \times T_{I\_2D} \times LR_{I\_2D} \tag{8}$$

3D Memory: Since static power (P_{S_3D}), especially for 3D memories, changes dynamically with temperature, we use thermal simulation to obtain individual rank static power at each epoch. Then,

Table 4. Core Architecture Parameters

Parameter	Value
Number of Cores	16
Core Model	2.1 GHz, 2.1 V, 45 nm, out-of-order, 3 way decode, 84 entry ROB, 32 entry LSQ
L1 I/D Cache	64 KB@2 ns, 2-way/64B-block
L2 Cache	Private, 512 KB@6 ns, 16-way/64B-block
3D Memory Configuration	1 GB, 16 channels, 8 ranks, 1 bank (per rank), closed page policy, 15 ns (latency), 8 GBps (per channel bandwidth)
2D Memory Configuration	2 GB, 2 channels, 2 ranks, 8 banks (per rank), closed page policy, 45 ns (latency), 12.8 GBps (per channel bandwidth)

we sum the individual rank static power values ($P_{S\_3D_i}$) to obtain the total static power of the 3D memory. Thus, static energy for a 3D memory (E_{S_3D}) with R ranks is:

$$E_{S\_3D} = P_{S\_3D} \times T_E = \left(\sum_{i=1}^{R} P_{S\_3D_i} \right) \times T_E \tag{9}$$

Using Equations (2) to (9), we can estimate the energy consumption of the memory system with both 2D and 3D memories. The model discussed above takes into account: (i) temperature dependent leakage energy consumption in 3D memories, (ii) data migration between 2D and 3D memories, (iii) leakage management in 2D memories, and (iv) varying number of 2D memory channels.

5 EXPERIMENTAL RESULTS AND ANALYSIS

5.1 Experimental Setup

Our trace-based framework makes use of Sniper 6.1 [7] (a performance simulator) and HotSpot 6.0 [39] (a thermal simulator) to obtain the temperature behavior of different workloads. Our DTM strategy and temperature prediction are integrated in the experimental framework. Workloads are executed using Sniper to obtain access trace for the 3D memory with an epoch time of 1 ms. This access trace is converted to a power trace using energy-per-access values determined using CACTI-3DD [8]. The power trace is subsequently used by HotSpot to obtain the temperature trace. In our experiments, we use models similar to state-of-the-art work [31] for modeling temperature dependent leakage power in the HotSpot thermal simulation, and we add the leakage power and the dynamic power to obtain the total 3D memory power (at every epoch of 1 ms). Moreover, we validated the above modeling done in HotSpot using ANSYS Icepak thermal simulation and observed that epoch time of 1 ms could capture the thermal behavior reliably. More details of modeling and validation are given in Section 5.1.1.

The core and memory architectural parameters are shown in Table 4 (similar to [29, 31]). Energy parameters for 2D and 3D memory determined are from CACTI-3DD [8] (values similar to prior art). The energy per access values are $\{EA_{3D}, EA_{2D}\} = \{20.55, 58.57\}$ (in nJ). To obtain P_{S_2D}, we found an 8MB bank consumes 14 mW of static power at 45°C (using CACTI-3DD). We assume that when the 2D memory is serving requests, it is fully turned ON ($LR_{A\_2D} = 1$, active state) and during the idle periods, we can bring down leakage significantly ($LR_{I\_2D} = 0.9$, using the self-refresh mode) [28]. The bus parameters are $\{EA_{BUS}, P_{S\_BUS}\} = \{10.28$ nJ, 0.106 mW$\}$. The temperature thresholds $\{THR, TH, THR\_COOL\} = \{84, 76.9, 74\}$ (in Celsius) are experimentally determined. Also, we take the average access rate of the last five epochs ($K = 5$) when estimating T_{K_SS}. Configuration parameters for HotSpot are shown in Table 5 (values same as [26]). We classify SPEC CPU2006

Table 5. HotSpot Configuration Values

Parameter	Value
Chip thickness	0.1 mm
Silicon thermal conductivity	100 W/mK
Silicon specific heat	1750 kJ/m^{3}K
Spreader thickness	1 mm
Spreader thermal conductivity	400 W/mK
DRAM thickness	0.02 mm
DRAM thermal conductivity	100 W/mK
Heatsink thickness	6.9 mm
Heatsink resistance	0.1 K/W

Table 6. Workloads and Benchmark Details

Benchmark Details and Type	Workload Name
povray(x16) – Compute	*povr*
perlbench(x16) – Compute	*perl*
bwaves(x16) – Mixed	*bwav*
GemsFDTD(x16) – Mixed	*Gems*
soplex(x16) – Memory	*sopl*
lbm(x16) – Memory	*lbm*
povray(x4), perlbench(x4), soplex(x4), lbm(x4)	*ppsl*
povray(x4), perlbench(x4), bwaves(x4), GemsFDTD(x4)	*ppbg*
bwaves(x4), GemsFDTD(x4), soplex(x4), lbm(x4)	*bgsl*
perlbench(x4), lbm(x8), GemsFDTD(x4)	*pllg*

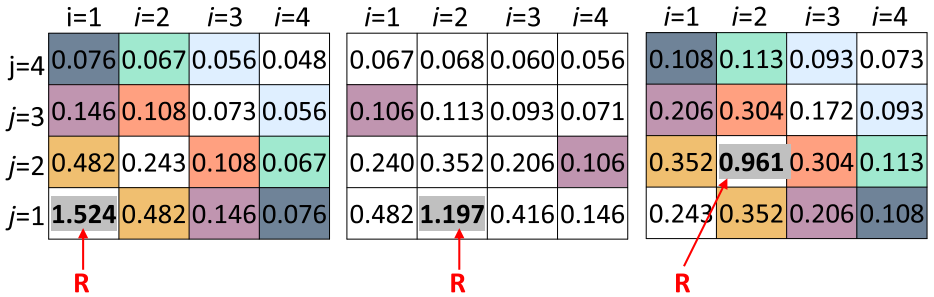

Fig. 12. Final weights learned for a rank R using the insights and symmetries discussed.

benchmarks into compute-intensive, mixed, and memory-intensive, and show results for two representative benchmarks from each category, building ten workloads (Table 6). Sixteen instances of each representative benchmark form the first 6 workloads, and mixes of the representative benchmarks are used to get the next 4 workloads. Simulations are run until the completion of at least 1 billion instructions for each benchmark. The benchmarks that complete early are restarted.

5.1.1 HotSpot Validation. In this section, we discuss modeling the temperature dependent leakage power (determined using CACTI-3DD) in HotSpot thermal simulation, and validate HotSpot steady-state and transient simulations using Icepak.

Modeling. In this work, we use a 1 GB, eight-layer, 16 channel, 50 nm, 3D memory, with the rank size being 8MB (= 1 GB/(8 × 16)). We modeled the above memory configuration in CACTI-3DD and obtained the energy per access (= 20.55 nJ, used for dynamic power calculation) and leakage power values. We also varied the temperature in the CACTI-3DD configuration file to obtain the temperature dependent leakage rank power variation (see Figure 15). We also show that the dynamic power (plotted for activity factor equal to 1) remains constant with temperature. Next, using non-linear exponential fitting, we obtained an equation for the temperature dependent leakage power variation. We add the leakage and dynamic power to obtain the total 3D memory power (at every epoch of 1 ms) in HotSpot.

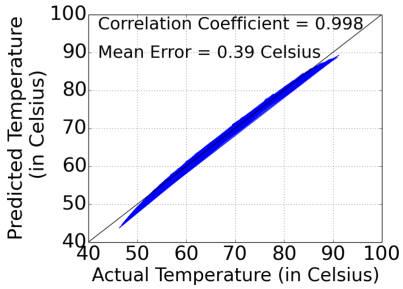

Fig. 13. Linear temperature predictor based on symmetry. Trend similar to Figure 3.

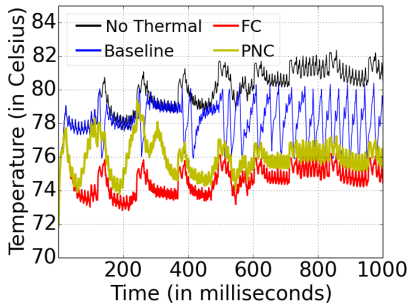

Fig. 14. *sopl* temperature trace.

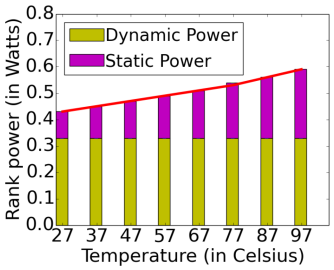

Fig. 15. Rank power versus temperature.

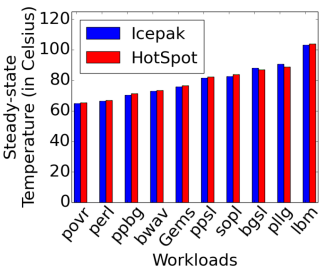

Fig. 16. Steady-state temperature for various workloads.

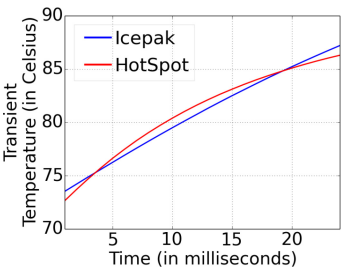

Fig. 17. Transient temperature (with no thermal limit).

Validation. We validated the HotSpot extension with temperature dependent leakage power modeling using ANSYS Icepak. We use the same floorplan, thermal conductivity values (see Table 5: HotSpot Configuration Values), temperature-dependent leakage power function (as discussed above), and power input (derived from the access trace) for both HotSpot and Icepak.

Steady state Temperature Validation: As mentioned in Section 5.1, we use the workload's access trace using Sniper 6.1 and energy per access values (from CACTI-3DD) to obtain the power trace. Using the power traces, we compare (HotSpot vs. Icepak) steady state temperature values (with no thermal limit) obtained for each benchmark. In our simulations, HotSpot gives an error range of −1.76 to 1.03°C with an average absolute error of 0.89°C with respect to Icepak (see Figure 16). We conclude that HotSpot gives accurate steady state temperature even in the presence of temperature dependent leakage power. Also, many benchmarks, starting from *ppsl* to *lbm*, reach temperatures higher than 80°C, motivating a need for thermal management in 3D memories.

Transient Temperature Validation: In the following experiment, we use a 3D memory with an average power dissipation of 1.792 Watt per rank (derived from *bgsl* workload) with no thermal limit. For brevity, the temperature trace is shown only for the most heated rank (see Figure 17); however, other ranks also follow similar trends in both HotSpot and Icepak. In our simulations, HotSpot gives an error range of −0.9 to 0.9°C with an average absolute error of 0.56°C for transient simulations (with an epoch time of 1 ms). As can be seen from Figure 17, HotSpot gives accurate transient temperature even in the presence of temperature dependent leakage power. We also observe that a sampling interval of 1 ms captures the transient thermal behavior reliably for 3D memories.

Table 7. Comparison of Proposed and Simple Steady-state Temperature Predictors

Type and parameter count	Mean Absolute Error	Prediction Time	Training Database Creation Time	Model Training Time	RAM usage for training
Simple, 18432	0.36°C	442 μs	72 days	468 seconds	60 GB
Proposed, 51	0.39°C	12 μs	2 days	4 seconds	0.3 GB

5.2 Results: Temperature Predictor

5.2.1 Characteristics of Proposed Predictor. As discussed in Section 4.1.3 and shown in Table 7, the temperature predictor built using our proposed techniques is very accurate, with a mean absolute error of 0.39°C (validated using HotSpot). Moreover, in the operating range (75°C to 80°C) of the DTM, we see an average error of 0.32°C and a maximum error of 1.5°C. Figure 13 shows the scatter plot of actual and predicted temperature for our proposed predictor which uses various insights and symmetry. The scatter plot shows that our predictor follows a similar trend to the predictor trained using raw data (see Figure 3). The weights learned for different locations are shown in Figure 12 and we observe that the different insights discussed in Section 4.1.3 are confirmed.

We measured the prediction time for our predictor to be 12 μs using simulation (on Sniper 6.1), in comparison to the simple predictor (Table 7) that has 18432 parameters with a prediction time of 442 μs. The proposed design insights reduce the model parameters by 362X, which in turn reduce the prediction time by 36X (=442/12). Training of these predictors is performed offline. The training process requires a database of steady-state temperatures for various access rates and channel ON/OFF conditions. The training database was generated using HotSpot simulations run on an Intel Xeon(R) CPU E5-1620, 3.70 GHz, Caches: 256 KB L1, 1 MB L2, 10 MB L3, and 64 GB RAM. It required ~72 days to generate the training database with 8 Million data points. However, for designing a predictor using our proposed techniques, we could generate the required database in ~2 days, resulting in a reduction of training database generation time by 36X (=72/2). Training the simple predictor using 8M data points required 60 GB of RAM and 468 seconds of training time, whereas we trained the proposed predictor within 4 seconds and needed 300 MB RAM, significantly reducing both training time and resources.

A simple linear predictor for our system requires a large number of parameters (18432) and is consequently slow (prediction time is 442 μs). Moreover, integrating such a predictor makes the DTM policy sluggish as epoch times have to be greater than 4.42 ms to maintain a prediction time overhead of under 10%. Also, the DTM would be unable to manage changes in temperature (~1°C per millisecond from our experiments). Hence, it is difficult to use the simple linear predictor for runtime thermal management of 3D memories.

5.2.2 Temperature Predictor Scalability. We analyze the proposed predictor's scalability by increasing the number of layers (and thus, memory size), keeping the number of channels fixed. We see that the number of parameters (and thus, the prediction time) for the proposed linear predictor remains constant (see Table 8), as they depend only on the number of channels (fixed in this experiment).

We observe a very low mean absolute error for four and eight layer memories. Also, the error increases with the number of layers; this is expected since leakage currents increase as the number of layers increases, and leakage currents have a non-linear (exponential) dependence on temperature. However, linear estimation works well for a smaller number of layers (in the operating range of 45°C to 100°C). Also, 12 layer memories require larger training data so that reasonable (lower error) linear approximation could be learned, which increases the training time and RAM usage.

Table 8. Proposed Predictor Characteristics for Various 3D Memory Capacities

Memory	Prediction			Training		
Layers and Size	Number of parameters	Mean Absolute Error	Prediction Time	Database Generation Time	Training Time	RAM usage during training
4 layer, 0.5 GB	51	0.13°C	12 μs	2 days	3.9 s	300 MB
8 layer, 1 GB	51	0.39°C	12 μs	2 days	4 s	300 MB
12 layer, 1.5 GB	51	2.4°C	12 μs	2.5 days	5.1 s	350 MB

We observe that in such cases to reduce error, we could use a linear predictor for smaller temperature ranges or learn weights for a non-linear predictor. In either case, the proposed insights, which are based on floorplan symmetry, could help in reducing the predictor complexity. We plan to explore non-linear symmetry based prediction in the future.

We analyzed the symmetry based predictor design for an eight-layer 3D memory having a 4×4 rectangular (instead of square) layer layout. We notice that all symmetries (except symmetry 1) and insights discussed in this work still hold (symmetry 2 has a minor modification). We found that the proposed predictor requires 76 parameters, whereas the simple linear predictor has 18432 parameters.

We also investigated the effect of increasing the number of processing cores (system size), keeping the memory configuration fixed; this causes a rise in memory access rates and the memory temperature. We observed that the maximum bandwidth limits the memory access rate. An increase in the core count has minimal or no impact on the training and prediction time, which is determined by the number of memory channels.

5.3 Results: PredictNcool DTM Policy

5.3.1 Runtime Comparison. Figure 18 shows the normalized runtime of the proposed strategy PredictNcool (PNC) with respect to FastCool (FC) [31] and the baseline [26]. Workloads are more memory intensive and result in higher memory temperature as we move from left (compute-intensive) to right (memory-intensive). In the baseline work, the authors assume that only 3D memory is present and redirect page allocation request for a hot channel to the coldest channel. If the coldest channel is already full, pages in the cold channels with lower access rates are moved to hot channels to accommodate the newly reallocated page within the cold channel. Such movement (internal migrations) has both energy and performance overheads. Further, the baseline strategy operates on workloads with a temperature higher than 75°C [26]. However, the reallocation does not reduce power (dynamic or leakage), causing frequent stalls at the memory for forced cooling.

Arguing that leakage and dynamic power are comparable for 3D memories at high temperatures, FC attempts to reduce temperature (and power) by turning OFF hot channels and migrating data to 2D memory. The FC strategy operates on workloads when temperatures are greater than 76.4°C, and as the temperature rises to 80°C, it moves to higher thermal emergency states, closing up to 8 memory channels and moving large data to 2D memory. Although FC achieves lower average runtime than the baseline due to the leakage power reduction and the use of the 2D memory, it does not anticipate the future trends and uses the current temperature to initiate the DTM. Also, as the temperature is a slowly varying function (time constant is in milliseconds), it takes time to respond to changes in the power/access profile. Thus, dynamic thermal management (DTM) strategies trigger at temperatures below the thermal limit. For the baseline and FastCool policies DTM activation starts at 75°C and 76.4°C respectively. Such early triggering in reactive DTMs can be delayed using temperature prediction.

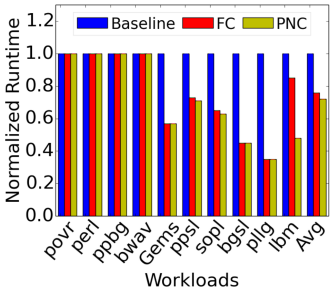

Fig. 18. Runtime comparison.

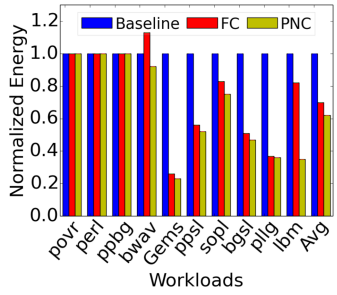

Fig. 19. Energy comparison.

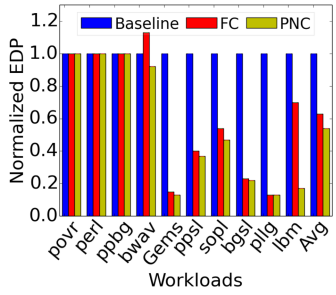

Fig. 20. EDP comparison.

PredictNcool (PNC) uses temperature prediction to decide on activating the FC policy. Based on the workload access profile and the current state of channel ON/OFF, we estimate if the memory would overheat in the future. We prevent transitioning to higher thermal emergency states if the prediction indicates the memory would remain within the thermal limit. Thus, on an average, PNC closes fewer 3D memory channels and migrates a lower volume of data to 2D memory. Hence, due to better utilization of the memory subsystem, PNC improves application performance by 28% and 5% in comparison with the baseline and FC, respectively. As shown in Figure 16, *ppsl*, *sopl*, *bgsl*, and *pllg* have similar steady-state temperatures, and exhibit similar trends in performance improvement and energy reduction. A more detailed analysis can be done using the temperature-time traces (see Section 5.3.2). Note that the application runtime in case of PNC includes the prediction time overheads (12 μs per epoch).

As both 2D and 3D memories operate in parallel, we notice a limited reduction in average application runtime of PNC in comparison to FC. Even though FC frequently transfers more data (when the 3D memory is heated) to 2D memory than PNC, in many cases, 2D memory has sufficient bandwidth to handle the migrated accesses and does not become a bottleneck (except in case of *lbm*, which is a highly memory intensive workload and thus, has significant migrated accesses). Hence, for both PNC and FC, even during phases where the 3D memory is heated, the 2D memory supports accesses and applications can make progress during this time.

5.3.2 Temperature Comparison. We use the temperature-time trace of *sopl* workload to illustrate the operation of various DTMs in more detail. We observe in Figure 14 that the temperature rises above 80°C if there is no thermal management. The baseline strategy can manage temperature initially and does not allow it to rise rapidly (with respect to no thermal constraint). However, from around 600 ms, the baseline policy, being page allocation based, is unable to regulate the rise in temperature and frequently stalls for cooling, causing runtime overheads. FC aggressively closes 3D memory channels and migrates data to 2D memory, achieving lower temperatures. It maintains the temperature in the range of 74 to 76°C. PredictNcool closes a lower number of channels in comparison to FC, thereby utilizing the temperature headroom better. This is because it is able to predict future temperature trends and identifies that closing 4 channels would prevent the temperature from rising above 80°C. However, FC, being reactive, closes additional channels when the temperature crosses 76.9°C.

5.3.3 Energy and EDP Comparison. As mentioned earlier, FC closes 3D memory channels aggressively, and thus, increases the (i) data transferred to 2D memory, (ii) migration energy, and (iii) 2D memory power. Consider the temperature-time trace of *bwaves* from Figure 4. As motivated earlier in Section 3, FC sometimes lowers the temperature by unnecessarily turning OFF channels in the 3D memory, causing the migration of data from 3D to 2D memory. For *bwaves*, such a

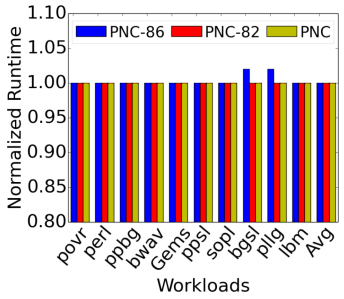

Fig. 21. Runtime comparison (for varying *THR*).

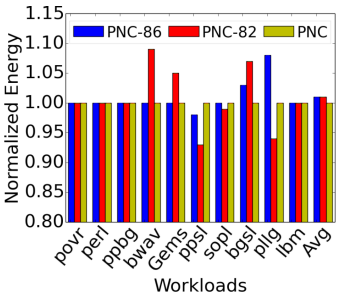

Fig. 22. Energy comparison (for varying *THR*).

migration causes 2D memory to turn ON unnecessarily, adversely impacting the energy consumption of FC (increasing it by 13%). However, PNC avoids such data movement and is more energy efficient.

Since the baseline strategy runs for longer (average 28% slower than PNC) and is unable to reduce the 3D memory static power (which is significant at high temperature), it consumes considerable energy. The internal page migrations in the baseline strategy also consume considerable energy. Figure 19 shows the memory energy (normalized) of PNC and FastCool (FC) with respect to the baseline. We observe that PNC decreases memory subsystem energy by an average of 38% and 12% in comparison with the baseline and FC respectively. PNC improves energy efficiency as it reduces both migration energy and 2D memory (static and dynamic) energy. We report the energy-delay product (EDP) for various DTM strategies in Figure 20 and notice that PNC reduces EDP by an average of 46% and 14% in comparison to the baseline and FC respectively.

5.3.4 Sensitivity Analysis. In this section, we discuss the details regarding the selection of various thresholds (such as *THR*, *TH*, and *THR_COOL*) for PNC. As mentioned previously, the thresholds are experimentally determined, and in our work, we use {*THR*, *TH*, *THR_COOL*} = {84, 76.9, 74} (in Celsius). To select a threshold value, we vary it experimentally, and the threshold value resulting in the maximum performance and minimum energy is chosen.

For brevity, we discuss only the selection of *THR*. This threshold value is compared with the estimated steady-state temperatures to decide on activating of the DTM strategy (Algorithm 1). We show results for three values: 82°C, 84°C, 86°C; the results are normalized with respect to *THR* = 84°C. As we observe in Figures 21 and 22, no single *THR* value gives best results for all benchmarks. Hence, we selected the *THR* (= 84°C) as the default value as it provides the minimum average runtime and energy consumption across all benchmarks.

Lower *THR* values cause a premature triggering of the DTM strategy. For example, in the case of *bwaves*, for *THR* = 82°C, we notice an early triggering of DTM, causing unnecessary data transfers to the 2D memory, increasing memory energy. Higher *THR* values cause delayed triggering of the DTM strategy and could make the DTM response sluggish. Late triggering of the DTM policy can cause the 3D memory to stall for cooling increasing the application runtime in some instances. Specifically, in case of *pllg* (for *THR* = 86°C), we notice that the temperature rises quickly, and PNC is late to respond causing higher static power dissipation and stalls in the 3D memory (increasing the application runtime).

6 CONCLUSION

In this paper, we developed a lightweight, steady state temperature predictor and used it for proactive temperature management of 3D memories. We utilized various design insights such as

floorplan symmetry to reduce the model parameters of the predictor, thereby enabling thermal prediction at runtime. We also presented a model to analyze the energy consumption of a memory subsystem with both 2D and 3D memories. We demonstrated the effectiveness of the proposed thermal management policy (PredictNcool) using SPEC CPU2006 benchmarks running on a 2D + 3D memory platform. Our results show performance improvements of 28% and 5%, and memory subsystem energy reductions of 38% and 12% over two recent thermal management strategies.

Though we demonstrated the predictor model size reduction using a linear predictor and for a 3D memory layout, we believe that the techniques discussed in this paper are generic. In the future, we plan to explore the effect on model size reduction for non-linear predictors, especially for memories with a large number of layers. We would like to extend the temperature predictor to accommodate asymmetries in the floorplan (due to process variation, memories with logic layers, etc.). We plan to analyze the applicability of these techniques for developing a transient temperature predictor (specifically, in the DTM operating temperature range). Also, we would like to explore ambient temperature aware DTM for 3D architectures.

ACKNOWLEDGMENTS

We are grateful to Rajesh Kedia of IIT Delhi for his reviews and valuable suggestions on this work. His detailed feedback helped improve the write-up of various sections (especially, the motivation). We would also like to thank Hameedah Sultan of IIT Delhi for her useful suggestions and help in validating HotSpot using Icepak. Further, we would like to acknowledge the efforts of the anonymous reviewers and their valuable comments. We thank MeitY for providing financial assistance to Lokesh Siddhu under the Visvesvaraya Ph.D. fellowship.

REFERENCES

[1] T. Adegbija and A. Gordon-Ross. 2018. TaPT: Temperature-aware dynamic cache optimization for embedded systems. *Computers* (2018).
[2] R. Ayoub, K. R. Indukuri, and T. S. Rosing. 2010. Energy efficient proactive thermal management in memory subsystem. In *ISLPED*.
[3] R. Ayoub and A. Orailoglu. 2010. Performance and energy efficient cache migration approach for thermal management in embedded systems. In *GLSVLSI*.
[4] P. Bogdan, P. P. Pande, H. Amrouch, M. Shafique, and J. Henkel. 2016. Power and thermal management in massive multicore chips: Theoretical foundation meets architectural innovation and resource allocation. In *CASES*.
[5] D. Brooks and M. Martonosi. 2001. Dynamic thermal management for high-performance microprocessors. In *HPCA*.
[6] D. Calvo, P. González, L. Díaz, H. Posadas, P. Sánchez, E. Villar, A. Acquaviva, and E. Macii. 2011. A multi-processing systems-on-chip native simulation framework for power and thermal-aware design. *JOLPE* (2011).
[7] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout. 2014. An evaluation of high-level mechanistic core models. *TACO* (2014).
[8] K. Chen et al. 2012. CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory. In *DATE*.
[9] R. Cochran and S. Reda. 2013. Thermal prediction and adaptive control through workload phase detection. *TODAES* (2013).
[10] A. K. Coskun, T. S. Rosing, and K. C Gross. 2008. Proactive temperature management in MPSoCs. In *ISLPED*.
[11] D. Cuesta, J. Ayala, J. Hidalgo, M. Poncino, A. Acquaviva, and E. Macii. 2010. Thermal-aware floorplanning exploration for 3D multi-core architectures. In *GLSVLSI*.
[12] K. Dev, A. N. Nowroz, and S. Reda. 2013. Power mapping and modeling of multi-core processors. In *ISLPED*.
[13] A. Fourmigue, G. Beltrame, and G. Nicolescu. 2014. Efficient transient thermal simulation of 3D ICs with liquid-cooling and through silicon vias. In *DATE*.
[14] M. H. Hajkazemi et al. 2017. Heterogeneous HMC+DDRx memory management for performance-temperature tradeoffs. *JETCS* (Sept. 2017).
[15] F. Hameed, M. A. A. Faruque, and J. Henkel. 2011. Dynamic thermal management in 3D multi-core architecture through run-time adaptation. In *DATE*.
[16] J. L. Henning. 2006. SPEC CPU2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News* (2006).

[17] H. Homayoun, A. Gupta, A. Veidenbaum, A. Sasan, F. Kurdahi, and N. Dutt. 2010. RELOCATE: Register file local access pattern redistribution mechanism for power and thermal management in out-of-order embedded processor. In *HiPEAC*.
[18] H. Huang, K. G. Shin, C. Lefurgy, and T. Keller. 2005. Improving energy efficiency by making DRAM less randomly accessed. In *ISLPED*.
[19] J. Jeddeloh and B. Keeth. 2012. Hybrid memory cube new DRAM architecture increases density and performance. In *VLSIT*.
[20] D. Juan et al. 2014. Statistical peak temperature prediction and thermal yield improvement for 3D chip multiprocessors. *TODAES* (2014).
[21] D. Juan, H. Zhou, D. Marculescu, and X. Li. 2012. A learning-based autoregressive model for fast transient thermal analysis of chip-multiprocessors. In *ASPDAC*.
[22] P. Kumar and D. Atienza. 2010. Neural network based on-chip thermal simulator. In *ISCAS*.
[23] D. Lee, S. Das, J. R. Doppa, P. P. Pande, and K. Chakrabarty. 2018. Performance and thermal tradeoffs for energy-efficient monolithic 3D network-on-chip. *TODAES* (2018).
[24] C. H. Liao, C. H. P. Wen, and K. Chakrabarty. 2015. An online thermal-constrained task scheduler for 3D multi-core processors. In *DATE'15*.
[25] W. Liu, L. Yang, W. Jiang, L. Feng, N. Guan, W. Zhang, and N. Dutt. 2018. Thermal-aware task mapping on dynamically reconfigurable network-on-chip based multiprocessor system-on-chip. *TC* (2018).
[26] W. Lo et al. 2016. Thermal-aware dynamic page allocation policy by future access patterns for Hybrid Memory Cube (HMC). In *DATE*.
[27] G. L. Loi et al. 2006. A thermally-aware performance analysis of vertically integrated (3D) processor-memory hierarchy. In *DAC*.
[28] Y. Lu et al. 2016. Rank-aware dynamic migrations and adaptive demotions for DRAM power management. *TC* (2016).
[29] J. Meng et al. 2012. Optimizing energy efficiency of 3-D multicore systems with stacked DRAM under power and thermal constraints. In *DAC*.
[30] F. Pedregosa et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[31] L. Siddhu and P. R. Panda. 2019. FastCool: Leakage aware dynamic thermal management of 3D memories. In *DATE*.
[32] G. Singla, G. Kaur, A. K. Unver, and U. Y. Ogras. 2015. Predictive dynamic thermal and power management for heterogeneous mobile platforms. In *DATE*.
[33] F. Sironi, M. Maggio, R. Cattaneo, G. F. D. Nero, D. Sciuto, and M. D. Santambrogio. 2013. ThermOS: System support for dynamic thermal management of chip multi-processors. In *PACT*.
[34] I. G. Thakkar, S. Pasricha, et al. 2018. LIBRA: Thermal and process variation aware reliability management in photonic networks-on-chip. *TMSCS* (2018).
[35] S. Xydis, G. Palermo, and C. Silvano. 2013. Thermal-aware datapath merging for coarse-grained reconfigurable processors. In *DATE*.
[36] Y. Ye, Z. Wang, P. Yang, J. Xu, X. Wu, X. Wang, M. Nikdast, Z. Wang, and L. H. K. Duong. 2014. System-level modeling and analysis of thermal effects in WDM-based optical networks-on-chip. *TCAD* (2014).
[37] M. Zapater et al. 2013. Leakage and temperature aware server control for improving energy efficiency in data centers. In *DATE*.
[38] K. Zhang, A. Guliani, S. Ogrenci-Memik, G. Memik, K. Yoshii, R. Sankaran, and P. Beckman. 2018. Machine learning-based temperature prediction for runtime thermal management across system components. *TPDS* (2018).
[39] R. Zhang, M. R. Stan, and K. Skadron. 2015. HotSpot 6.0: Validation, acceleration and extension. (2015).
[40] Z. Zhao, A. Gerstlauer, and L. K. John. 2017. Source-level performance, energy, reliability, power and thermal (PERPT) simulation. *TCAD* (2017).
[41] J. Zheng, N. Wu, L. Zhou, Y. Ye, and K. Sun. 2016. DFSB-based thermal management scheme for 3D NoC-bus architectures. *TVLSI* (2016).

Received April 2019; revised June 2019; accepted July 2019