Thermal Management for 3D-Stacked Systems via Unified Core-Memory Power Regulation

YIXIAN SHEN, LEO SCHREUDERS, ANUJ PATHANIA, and ANDY D. PIMENTEL, University of Amsterdam, The Netherlands

3D-stacked processor-memory systems stack memory (DRAM banks) directly on top of logic (CPU cores) using chiplet-on-chiplet packaging technology to provide the next-level computing performance in embedded platforms. Stacking, however, severely increases the system's power density without any accompanying increase in the heat dissipation capacity. Consequently, 3D-stacked processor-memory systems suffer more severe thermal issues than their non-stacked counterparts. Nevertheless, 3D-stacked processor-memory systems do inherit power (thermal) management knobs from their non-stacked predecessors - namely Dynamic Voltage and Frequency Scaling (DVFS) for cores and Low Power Mode (LPM) for memory banks. In the context of 3D-stacked processor-memory systems, DVFS and LPM are performance- and power-wise deeply intertwined. Their non-unified independent use on 3D-stacked processor-memory systems results in suboptimal thermal management. The unified use of DVFS and LPM for thermal management for 3D-stacked processor-memory systems remains unexplored. The lack of implementation of LPM in thermal simulators for 3D-stacked processor-memory systems hinders real-world representative evaluation for a unified approach.

We extend the state-of-the-art interval thermal simulator for 3D-stacked processor-memory systems *CoMeT* with an LPM power management knob for memory banks. We also propose a learning-based thermal management technique for 3D-stacked processor-memory systems that employ DVFS and LPM in a unified manner. Detailed interval thermal simulations with the extended *CoMeT* framework show a 10.15% average response time improvement with the *PARSEC* and *SPLASH-2* benchmark suites, along with widely-used Deep Neural Network (DNN) workloads against a state-of-the-art thermal management technique for 2.5D processor-memory systems (ported directly to 3D-stacked processor-memory systems) that also proposes unified use of DVFS and LPM.

CCS Concepts: • **Computing methodologies → Reinforcement learning algorithms**; • **Hardware → 3D-stacked architectures**; • **Computer systems organization → Embedded systems;**

Additional Key Words and Phrases: Low power design, resource-constrained edge computing, embedded systems, 3D-stacked processors, AI for systems, chiplet-on-chiplet stacking

ACM Reference format:
Yixian Shen, Leo Schreuders, Anuj Pathania, and Andy D. Pimentel. 2023. Thermal Management for 3D-Stacked Systems via Unified Core-Memory Power Regulation. *ACM Trans. Embedd. Comput. Syst.* 22, 5s, Article 120 (September 2023), 26 pages.
https://doi.org/10.1145/3608040

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), 2023.

Authors' address: Y. Shen, L. Schreuders, A. Pathania, and A. D. Pimentel, University of Amsterdam, Science Park 900, Amsterdam, The Netherlands, 1098XH; emails: y.shen@uva.nl, leoschreuders@hotmail.com, {a.pathania, a.d.pimentel}@uva.nl.

1539-9087/2023/09-ART120 $15.00
https://doi.org/10.1145/3608040

1 INTRODUCTION

Lack of memory bandwidth continues to stifle the performance of high-end embedded applications. 3D-stacked processor-memory systems [22, 28] resolve the memory bandwidth bottleneck by stacking main memory DRAM banks on the processing cores using chiplet-on-chiplet packaging technology. The technology allows for numerous shorter vertical interconnects between the memory and logic layers. Therefore, the system's transistor density can increase without technology scaling.

However, stacking several silicon layers on top of each other increases the volume of the 3D-stacked processor-memory systems while only minimally increasing its surface area over non-stacked processor-memory systems [46]. Therefore, the power density of 3D-stacked processor-memory systems is significantly higher than their non-stacked counterparts, while their heat dissipation capabilities remain comparable [16, 33]. Furthermore, higher bandwidth allows cores and memory to operate more actively in 3D-stacked processor-memory systems resulting in significantly more heat than in non-stacked systems [27]. Consequently, the 3D-stacked processor-memory systems suffer from more severe thermal issues that force them to operate at much lower frequencies than non-stacked systems. This thermally sustainable execution in 3D-stacked processor-memory systems eradicates most performance gains over non-stacked systems.

Figure 1 shows abstract three-dimensional heat conduction in 3D-stacked processor-memory systems. 3D-stacked processor-memory systems stack multiple memory layers over a core layer [28]. Most 3D-stacked processor-memory system designs [22] place the heat sink with the core layer because the logic is responsible for most of the heat in the system. Another heat sink on the other side is infeasible as the chip must embed with the motherboard (PCB) on one side. The capacity of the PCB as a secondary heat sink is limited. The memory layers absorb heat from the core layer while producing heat. Therefore, while the thermal hotspots occur in the cores in non-stacked processor-memory systems, the hotspots in 3D-stacked processor-memory systems are in the memory layers. Consequently, thermal management of 3D-stacked processor-memory systems is quintessential to their thermally-safe operation.

3D-stacked processor-memory systems inherit thermal management knobs for core and memory layers. 3D-stacked processor-memory systems can use Dynamic Voltage Frequency Scaling (DVFS) to scale the frequency of their processing cores in the core layer. DVFS allows the cores to reduce their heat generation at the cost of their peak performance and vice versa [8–10, 14, 18, 20, 23, 26, 37, 39, 40, 50]. 3D-stacked processor-memory systems can also toggle their memory banks between Low Power Mode (LPM) and Normal Power Mode (NPM) to trade off their power consumption with performance [30, 45, 47]. Enabling LPM mode for memory banks significantly reduces their heat generation, but this reduction comes at the cost of an increase in memory access latency. Though DVFS and LPM target different parts of 3D-stacked processor-memory systems for power (thermal) management, they are performance- and power-wise intertwined [45]. The effect on the performance of DVFS and LPM for an application is subject to the application's memory and compute intensity. The performance of a memory-intensive application is more sensitive to LPM, while the performance of a compute-intensive application is more sensitive to DVFS. However, the thermals of an application on 3D-stacked processor-memory systems are inherently subject to the combined power consumption of both cores and memory. Therefore, a smart application-aware thermal manager can save power using non-sensitive knobs such as DVFS for memory-intensive applications and LPM for compute-intensive applications without significantly deteriorating applications' performance. The manager can then use the saved power to boost performance using the sensitive knob within a given thermal envelope. This observation makes a case for a unified DVFS and LPM thermal management strategy for 3D-stacked processor-memory systems. We further strengthen the case for such a strategy with a motivational example.

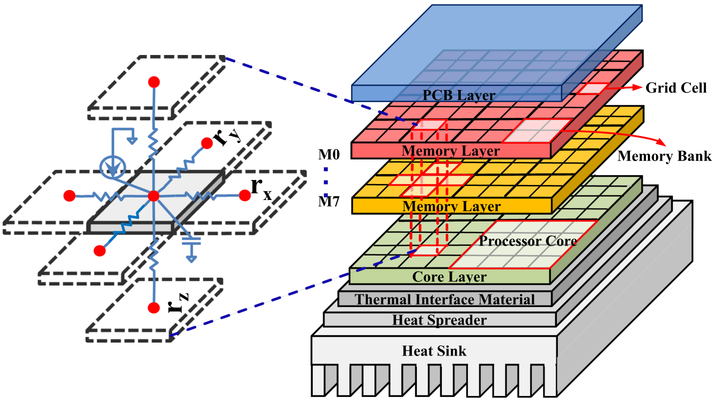


Fig. 1. An abstract representation of heat conduction on 3D-stacked processor-memory systems [36].

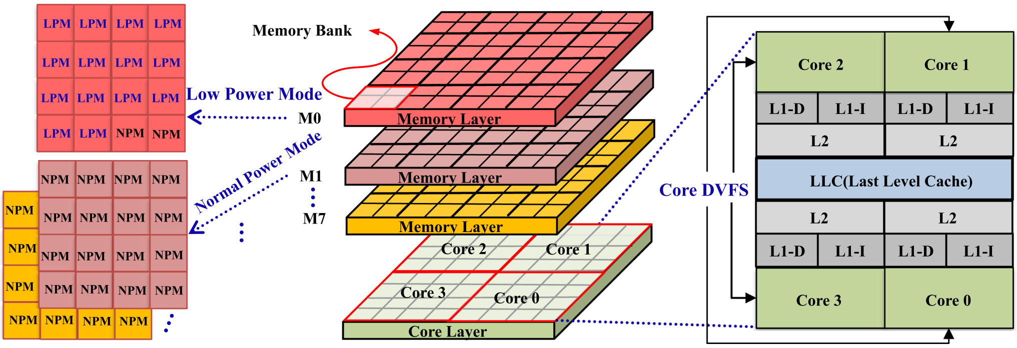
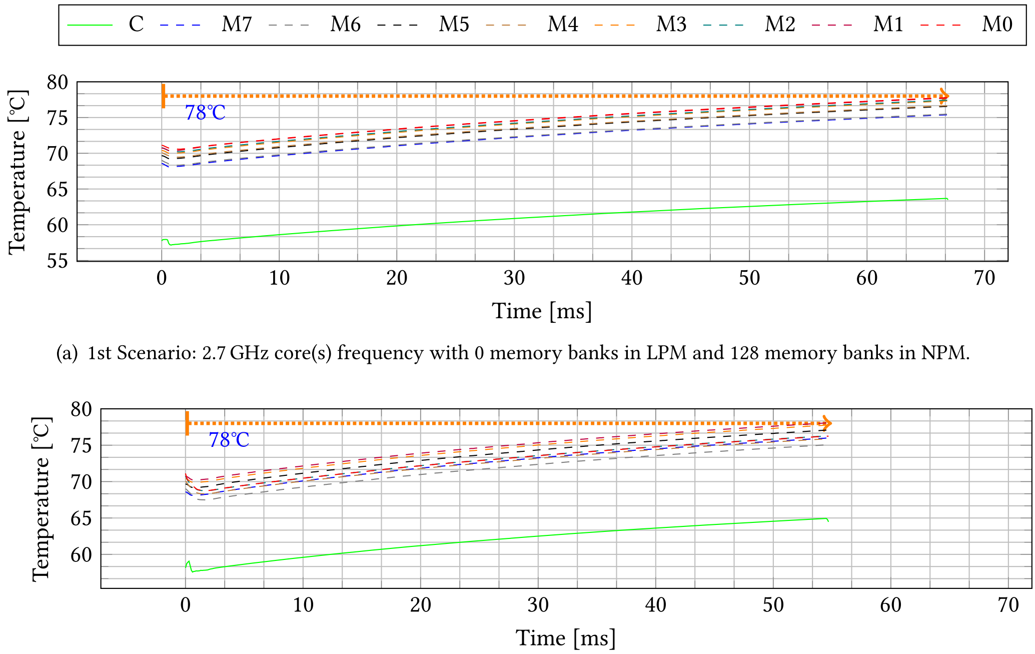


Fig. 2. An abstraction of a 3D-stacked processor-memory system with DVFS and LPM.

Motivational Example: Figure 2 illustrates a 3D-stacked processor-memory system consisting of one quad-core processor layer and eight memory layers. Each memory layer contains 16 memory banks. We select the *streamcluster* benchmark (from the *PARSEC* [2] benchmark suite) executing in a four-threaded configuration to evaluate the performance implications of synergistically employing LPM and DVFS under a thermal threshold of 78 ℃. We present two scenarios.

In the first scenario, we ran the *streamcluster* benchmark at a constant core(s) frequency of 2.7 GHz and all memory banks in NPM. 2.7 GHz is the highest core(s) frequency in which the *streamcluster* benchmark reaches the thermal threshold of 78 ℃ in our 3D-stacked processor-memory system but does not cross it in the steady state, as shown in Figure 3(a). Running at a frequency higher than 2.7 GHz will violate the thermal threshold. We observe a response time of 670.96 ms in the first scenario. We know that *streamcluster* is a benchmark involving significant computations. Consequently, the effect of LPM on its performance is limited. Therefore, in the second scenario, we turn the first fourteen memory banks of memory layer *M0* into LPM, as shown in Figure 2. We transfer the power saved from the memory layers using LPM to the core layer using DVFS to run *streamcluster* at a higher core(s) frequency of 4 GHz. Figure 3(b) shows that *streamcluster* continues to remain below the thermal threshold of 78 ℃ in the steady state. However, being a compute-intensive benchmark, *streamcluster* benefits more from the higher core(s) frequency than its penalization due to the use of LPM. We observe a response time of 547.95 ms in



(a) 1st Scenario: 2.7 GHz core(s) frequency with 0 memory banks in LPM and 128 memory banks in NPM.

(b) 2nd Scenario: 4 GHz core(s) frequency with 14 memory banks in LPM and 114 memory banks in NPM.

Fig. 3. Layer-wised peak temperature for different layers in a 3D-stacked processor-memory system running a four-threaded instance of *streamcluster* benchmark with and without LPM and varying core(s) frequency.

the second scenario. The performance of the *streamcluster* in the second scenario (Figure 3(b)) is 18.33% better than the first scenario (Figure 3(a)).

In practice, it remains a challenge to identify the correct combination of DVFS and LPM based on application characteristics that will minimize the application's response time on a 3D processor-memory system. Recent works [40] have shown machine learning-based thermal management for multi-/many-core processor-memory systems to be an effective strategy. Therefore, in this work, we propose a reinforcement-learning-based technique that unifies DVFS and LPM in one technique to identify near-optimal settings for both knobs synergistically for maximizing an application's performance on 3D processor-memory systems.

Our Novel Engineering Contributions: Several simulators come equipped with power-performance modeling for core DVFS. However, no simulator capable of modeling the thermals of 3D-stacked processor-memory systems comes with an LPM implementation for memory banks. Past research evaluates LPM-based thermal management by offline retrospective manipulation of power and performance traces collected from various simulators [45]. Such evaluation may be an incomplete representation of real-world execution. Therefore, we extend the *CoMeT* [46] interval thermal simulator to model the impact of LPM on application performance and power consumption in a 3D-stacked processor-memory system. We have merged the code for our extensions with the main branch of the open-source *CoMeT* repository at *GitHub*.

Our Novel Scientific Contributions: We propose the first technique for thermal management of 3D-stacked processor-memory systems that synergistically employ core DVFS and memory bank LPM in a unified manner. The technique uses a learning-based algorithm to determine the frequency of every active core and the mode of every memory bank to minimize the response of an application running on a 3D-stacked processor-memory system under a given thermal threshold.

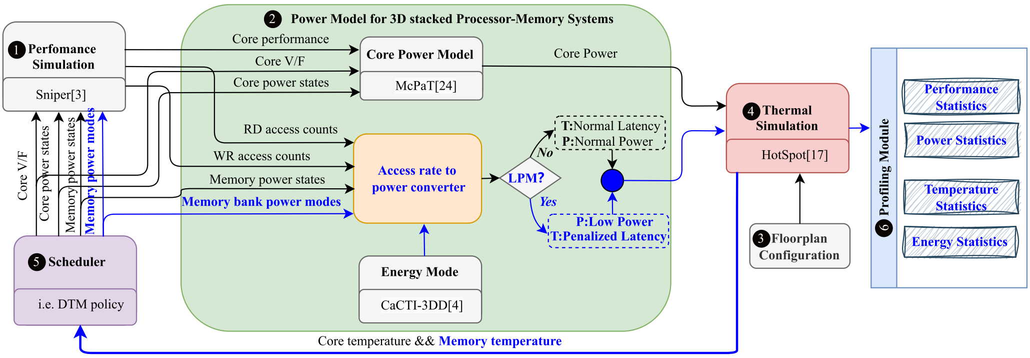


Fig. 4. Flow in simulated 3D-stacked processor-memory systems [46] with extensions in blue.

We compare the proposed technique with state-of-the-art thermal management techniques for 3D-stacked and 2.5D processor-memory systems via detailed interval thermal simulations.

2 LOW POWER MODE FOR 3D-STACKED PROCESSOR-MEMORY SYSTEMS

2D memories rarely suffer from thermal issues, given their low power density and slow operating frequencies. However, the emergence of high-capacity, high-bandwidth, high-density 3D-stacked memories significantly increased the power density of main memories. Consequently, the thermal management of high-density memories became integral to their operation [44]. In contrast to cores, static (leakage) power consumption dominates the total power consumption for memory banks. Therefore, unlike in cores, DVFS that reduces the active (dynamic) power consumption was ineffective for the thermal management of memories. Instead, Low Power Mode (LPM) for main memory that reduces both active and static power consumption of memory banks became the power management knob of choice for thermal management of high-density memories. LPM eliminates the need for data migration or replication by preserving the memory data in situ. However, cores cannot access the data in the memory banks when they are in LPM mode. Therefore, any memory access to memory banks in LPM requires the memory bank to toggle to Normal Power Mode (NPM). Transitioning to the NPM from LPM necessitates a resynchronization process. During this period, the data stored within the memory bank becomes temporarily inaccessible. We model this resynchronization duration based on the findings presented in [30, 45]. As such, the total time associated with every memory access event in the LPM comprises both the resynchronization duration and the standard memory access time. Note that once a memory bank has transitioned to the NPM due to a memory access event, it immediately switches back to LPM to save power. This design choice prioritizes saving power at the cost of increased overheads due to mode toggling.

3D-stacked processor-memory systems inherit the LPM power management knob from their predecessor systems. However, none of the few thermal simulators capable of modeling the thermals of 3D-stacked processor-memory systems provides an implementation for LPM. Therefore, we extend the recently introduced state-of-art interval thermal simulator *CoMeT* [46] with the necessary power-performance modeling associated with LPM. Figure 4 shows the extended *CoMeT* tool flow. The parts in blue represent the new extensions.

The ❶ Performance Simulator (*Sniper* [3]) simulates a workload and tracks access counts to internal components such as execution units, caches, and register files. *CoMeT*[46] extends the performance simulator to count memory access for every memory bank based on the memory addresses. Our extensions keep count of both NPM and LPM memory accesses. *Sniper* uses memory access latencies to model performance. Our extensions make the latencies a function of memory

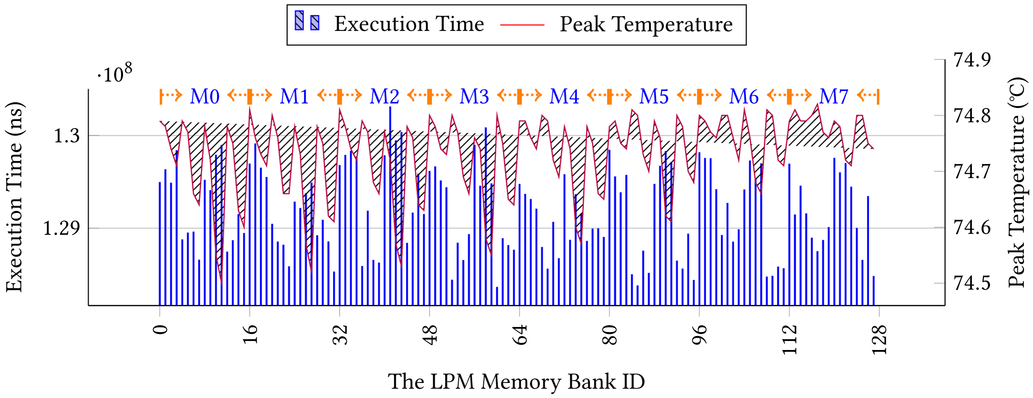


Fig. 5. The execution time and peak temperature with varying LPM memory banks for *streamcluster*.

bank power modes. We introduce the concept of heterogeneous latencies. We make memory access latency for a memory bank in LPM higher than for a bank in NPM.

CoMeT uses *McPat* [24] and *CACTI-3DD* [4] to model the power consumption of cores and memory banks, respectively. Our extensions target the *CACTI-3DD* integration in *CoMeT*. The ❷ Power Model receives the accumulated access counts at every simulation interval. It needs to be aware of the power mode of each memory bank, as banks in LPM consume a fraction of the power compared to those in NPM, with different fractions accounting for leakage and dynamic power. These fractions are determined based on technical specifications, as outlined in [30]. The power model generates a response corresponding to the power consumption based on the different power modes of each memory bank and the energy per access for read (RD) and write (WR) operations.

The role of the ❹ Thermal Simulator in the *CoMeT* tool is to calculate the temperature of each thermal component at each interval using the power consumption data received from the ❷ Power Model and ❸ Floorplan Configuration. The ❹ Thermal Simulator does not need to know the power modes of the memory banks. It inherently models memory banks in LPM to produce less heat due to their lower power consumption. We extend the scheduling API in CoMeT, *SchedAPI*, to allow the scheduler to read memory bank temperature. We also add extensions to the API that allow the scheduler to toggle a memory bank between LPM and NPM. Therefore, based on the temperature feedback from the ❹ Thermal Simulator, the ❺ Scheduler can switch the power mode of a memory bank for thermal management. Finally, the ❻ Profile Module offers statistical data that reflects the impact of LPM on power, temperature, and energy consumption at each interval, as well as the overall performance data. We gain insights into how the applications behave in LPM per their power, energy, and thermals through profiling. A scheduler can use this information to make informed decisions about LPM to achieve the desired results.

3 EXTRA-FUNCTIONAL ANALYSIS OF DRAM ACCESS FOR LOW POWER MODE

We conduct an empirical extra-functional analysis to characterize changes in performance and peak temperature of 3D processor-memory systems by the number and location of memory banks in LPM. Figure 5 shows the variation in *streamcluster* performance by turning different memory banks from NPM into LPM one at a time. We observe a non-uniform distribution that indicates putting different memory banks in LPM leads to different performance (up to 1.5%).

Figure 5 also illustrates the corresponding observed peak temperature. As with the performance results, there is a significant difference in peak temperature between distinct memory banks in LPM (up to 0.32 °C). Figure 6 charts the large variation in memory access count in memory banks

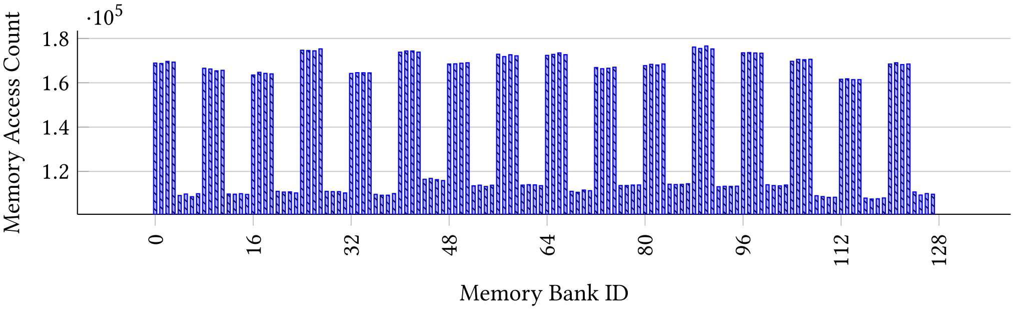
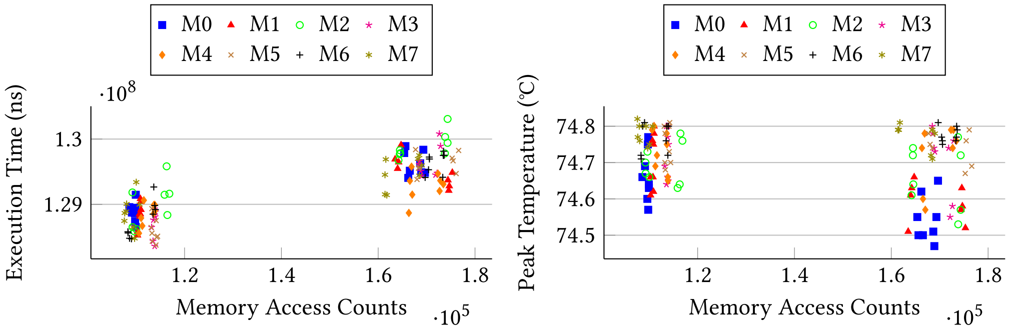


Fig. 6. The access count of individual memory banks in a 3D-stacked system for *streamcluster*.



(a) Performance

(b) Temperature

Fig. 7. Execution time and peak temperature for *streamcluster* with one memory bank in LPM against the memory access count of the memory bank that is in LPM.

that can explain these results. Figure 7 combines the data from Figure 5 and Figure 6 to plot the execution time and peak temperature for *streamcluster* with one memory bank in LPM against the memory access count of the memory bank that is in LPM. The figure shows a positive correlation between execution time and memory access count. This correlation indicates memory banks with a higher memory access count lead to a higher performance penalty when they go into LPM. However, the figure also shows no correlation between observed peak temperature and memory access count. Nevertheless, we still observe turning off memory banks in higher memory layers (near the PCB) leads to a bigger decrease in peak temperature regardless of memory access counts. Therefore, the spatial location of the memory bank also plays a role in the peak temperature observation.

Based on these observations, we draw the following conclusions. We conclude that memory banks located in the upper memory layer (close to the PCB layer) with fewer memory access counts tend to provide better performance and thermal benefits when placed in LPM. Conversely, memory banks located in the lower memory layer with high memory access count tend to result in higher performance penalties and fewer thermal benefits when placed in LPM. This observation suggests that a careful selection of the memory banks for LPM based on their location and memory access count is quintessential to better performance-aware thermal management.

4 THERMAL MANAGEMENT DESIGN

Thermally managing the 3D-stacked processor-memory systems by intelligently switching the memory bank into LPM is already a complex problem, as discussed in Section 3. However, in combination with core DVFS, achieving a unified thermal management strategy becomes even more daunting. We aim to delve deeper into the design space of this unified thermal management for 3D-stacked processor-memory systems. Let the system has N cores and M memory banks. Each core has G DVFS settings. Each memory bank has K actions (i.e., LPM and NPM). The action design space A is $G^N \cdot K^M$. We then consider the state space S, which has I features, including core IPC, core utilization, memory read/write count, etc. Each state feature has H values, and the total design space is $A \cdot S$, which is $G^N \cdot K^M \cdot H^I$. Faced with such a colossal design space, selecting appropriate DVFS levels and memory modes for efficient thermal management is computationally expensive. It is challenging to find one-size-fits-all approaches due to the complexity of the problem and the lack of feasible solutions in polynomial time for a run-time thermal management policy.

We propose a DQN (Deep Q-network) [34]-based algorithm, *3QUTM*, for performing unified thermal management on 3D-stacked processor-memory systems. *3QUTM* coordinates DVFS and LPM and continuously learns the application performance, the interdependent impact of one subsystem on another (i.e., the overcompensation effect), and environmental changes using reinforcement learning. *3QUTM* selects actions based on the anticipated maximum future rewards. The rewards are determined by evaluating the current state and selecting the optimal action that yields the highest future reward. *3QUTM* plays a crucial role in periodically adjusting frequency/voltage strategies for the core and LPM for the memory bank to improve performance while mitigating thermal stress. *3QUTM* leverages a neural network to manage the high-dimensional action and state spaces $A \cdot S$. We use a prioritized experience replay buffer and the sampling copies from past data to minimize the adaptation time to environmental changes and improve the convergence speed. *3QUTM* consistently achieves high IPC in 3D-stacked systems with high compute intensity while effectively balancing the overhead in systems with high memory intensity. *3QUTM* learns the interlinked effects from both subsystems to adapt to the application and environmental changes.

Architecture model: We employ 3D-stacked processor-memory systems with one layer of homogeneous processing cores and multiple layers of main memory banks, as illustrated in Figure 1. The core layer features cores with identical micro-architectures sharing the same memory address space. The core layer stacks directly above the heat sink. Multiple layers of memory banks stack above the core layer, with the topmost memory layer embedded in the secondary minor heat sink (PCB). Communication between the cores and memory banks occurs through Through-Silicon Vias (TSVs). The 3D processor-memory system operates in a thermally constrained environment, and its temperature must remain below a threshold, denoted by T_{DTM}, to prevent hardware-controlled Dynamic Thermal Management (DTM) from being triggered. The 3D processor-memory system executes multi-threaded multi-program shared-memory benchmarks using a one-thread-per-core model.

4.1 Problem Formulation

We formulate the optimization problem by incorporating the observations from Section 3. Specifically, we define the system throughput $\mathbb{T}$ and the total power consumption of the system as P_{tot}. The overall objective is to maximize system throughput $\mathbb{T}$ while minimizing total power consumption P_{tot}. We define the optimization problem as follows:

$$\max_{\pi} \frac{1}{t_D} \sum_{t=t_0}^{t_{D-1}} \left\{ \zeta \mathbb{T}_t + \frac{\Phi}{P_{tot_t}} \right\} \qquad (1)$$
$$s.t.\ T_{peak} \leq T_{cr}$$

Maximizing the average value of Equation (1) across t_D (the duration of the optimization problem execution) leads to maximum performance $max\ \mathbb{T}_{t_D}$ while minimizing power consumption *min* P_{tot} under thermal limitations. Here, T_{peak} and T_{cr} denote the peak and critical temperatures of 3D-stacked systems, respectively. ζ and Φ are trade-off coefficients for balancing performance and power consumption. Each decision epoch seeks to maximize long-term rewards (a central concept in reinforcement learning), as realized through the scheduling policy π, represented by decision variables such as core DVFS settings and LPM knobs. This reward-driven approach lends itself naturally to the framework of a Markov Decision Process (MDP), thereby bridging our optimization problem with the principles of reinforcement learning. The Bellman optimality equation shown in Equation (2) substantiates this connection. We assume an action set A and a state set S. We have $\forall a \in A$ and $\forall s \in S$ for each action and state, respectively.

$$V_{\pi^*}(s_t) = \max_{\pi} \left\{ r(s_t, s_{t+1}) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) V_{\pi^*}(s_{t+1}) \right\} \tag{2}$$

where $V_{\pi^*}(s_t)$ represents the value function of state s_t for an optimal scheduling policy π^*, $P(s_{t+1}|s_t, a_t)$ is the probability of transitioning from state s_t to s_{t+1} by taking action a_t, $r(s_t, s_{t+1})$ represents the immediate reward received when the agent acts a_t at state s_t after observing s_{t+1}, and γ is the discount factor that determines the importance of future rewards relative to immediate rewards. A value of $\gamma = 1$ indicates that the future reward is as important as the immediate reward, while $\gamma = 0$ indicates that the agent does not consider future rewards.

Finding an optimal scheduling policy π^* is essential to maximize future rewards. However, accurately modeling the transitions between the peak temperature of a 3D-stacked system and the employed DVFS and LPM policy can be practically challenging due to the difficulty of estimating transition probabilities. As a solution, we adopt a model-free approach. One such approach is Q-learning [48], which can converge to the Bellman equation without requiring probabilistic modeling of state transitions. Q-learning aims to identify a scheduling policy that maximizes the long-term reward by transforming the value function $V(s_t)$ into the Q-function $Q(s_t, a_t)$.

$$Q_{\pi^*}(s_t, a_t) = \max_{\pi} \mathcal{E}(r(s_t, s_{t+1}) + Q_{\pi^*}(s_{t+1}, a_{t+1})) \tag{3}$$

where $Q_{\pi^*}(s_t, a_t)$ represents the cumulative expected reward for a given action a_t at a specific state s_t, and $r(s_t, s_{t+1})$ denotes the immediate reward received after taking action a_t at state s_t following the resulting state s_{t+1}. However, Q-learning is not a perfect approach to solving our problem. Firstly, it only supports finite discrete states and action sets, making it incapable of handling continuous values such as power consumption. Secondly, even after discretizing our continuous attributes based on statistical distribution, the resulting design space remains vast, requiring extensive storage space and rendering convergence difficult. Consequently, it is impractical for our problem with large state and action spaces. We adopt a Deep Q-network (DQN) to address the limitations of Q-learning. We propose a lightweight Q-network to approximate the Q-function.

4.2 State, Action and Reward Design

4.2.1 State. We select features for defining the state of 3D-stacked systems, namely IPC (Instructions Per Cycle), frequency, core utilization, peak temperature, the average temperature of the hottest memory layer, total power consumption, and memory access count for their significant relevance to the performance, power consumption, and thermal management of these systems. We define the system state of 3D-stacked systems presented in Table 1 by utilizing these internal features. Many of the features in Table 1 are execution-specific. Therefore, *3QUTM* is an application-(thread-)specific solution. Some features are discrete to provide a balance between model complexity and accuracy. T_{hotL}, T_{peak}, and P_{tot} are retained as continuous variables, demonstrating a prioritization of thermal management over performance. For efficient training, the features s^{IPC}

Table 1. State-related Features and its Quantized Values for s^{IPC}, s^{U}, and s^{MAC} for 3D-stacked Systems

State	Description	Discrete values
s^{IPC}	Billion Instructions per Cycle per Core	0,1
s^{f}	Frequency per Core	1.0, 1.2, . . . , 4.0
s^{U}	Utilization per Core	0,1,2,3,4,5,6
$s^{T_{peak}}$	Peak temperature of 3D-stacked Systems	-
$s^{T_{hotL}}$	Average Temperature of the Hottest Memory Layer	-
$s^{P_{tot}}$	Total Power Consumption	-
s^{MAC}	Memory Access Count per Memory Bank	0,1,2,3,4

and s^{U} are discrete. To this end, we employ clustering, a flexible and data-driven approach capable of discerning intricate patterns potentially missed by simpler thresholding techniques. We conduct basic experiments on all PARSEC-2 benchmarks to obtain and quantize the feature values. We use K-means clustering [15] to group s^{IPC} into two categories – computer-intensive thread or memory-intensive thread. For the discrete s^{MAC} and s^{U} attributes, we employ DBSCAN [42] to determine the optimal number of clusters, considering the variation in densities and shapes for these two state features as revealed by the profiling data. Figure 8 visualizes the $|s^{MAC}|$ using the *swaptions* benchmark. Assuming N cores and M memory banks in the 3D-stacked systems, we represent the system state using a tuple, $s_t = \{s_t^{IPC_0}, \ldots, s_t^{IPC_{N-1}}, s_t^{f_0}, \ldots, s_t^{f_{N-1}}, s_t^{U_0}, \ldots, s_t^{U_{N-1}}, s_t^{T_{peak}}, s_t^{T_{hotL}}, s_t^{P_{tot}}, s_t^{MAC_0}, \ldots, s_t^{MAC_{M-1}}\}$.

4.2.2 Action. We define the set of actions as the clock frequency of cores and the number and location of the LPM memory bank. For the V/F action set, we set the minimum operating frequency to 1.0 GHz, with a step of 0.2 GHz for dynamic frequency adjustment. The maximum frequency is 4.0 GHz. However, exploring the entire search space for selecting the number of LPM memory bank locations is prohibitively expensive (i.e., $N = 4$, $M = 128$, $16^4 \cdot 2^{128} = 2^{144}$ combinations). We exploit the observation in Section 3 that transitioning the memory banks located in higher memory layers with lower access counts to LPM results in minimal performance penalties while increasing thermal headroom. The core is the main heat source in a 3D-stacked processor-memory system. The primary heat sink convects away most of the heat. Still, a significant amount of heat also flows towards the secondary heat sink (PCB) for a less efficient convected dissipation. Consequently, the layer closest to the PCB layer has a high chance of violating the thermal threshold. Under these considerations, we perform the LPM actions as per:

- a^{LPM_0}: The action entails transitioning the hottest memory bank alongside its neighboring six memory banks (i.e., front, back, left, right, upper, bottom) into LPM.
- a^{LPM_1}: Place all the memory banks in the hottest layer into LPM.
- $a^{LPM_{2-6}}$: This action category entails transitioning a set number m of memory banks into LPM, where $m \in \{1,2,4,8,16\}$. These memory banks are from the first six layers, with each selected memory bank showing memory access count attribute $|s^{MAC}| = 0$.
- $a^{LPM_{7-10}}$: Analogous to the prior category, this group of actions also places m memory banks into LPM (where $m \in \{1,2,4,8\}$). However, we based the selection on $|s^{MAC}| = 1$.
- $a^{LPM_{11-14}}$: Similarly, this action places m memory banks (where $m \in \{1,2,4,8\}$) into LPM, selecting from memory banks from the first six layers when $|s^{MAC}| = 2$.
- $a^{LPM_{15-19}}$: This category of actions targets memory banks from the first six layers with $|s^{MAC}| = 3$, placing m (where $m \in \{1,2,4,8,16\}$) into LPM.

- $a^{LPM_{20-24}}$: The final category of actions operates on memory banks with $|s^{MAC}| = 4$ in the first six layers, transitioning m (where $m \in \{1,2,4,8,16\}$) into LPM.

LPM actions entail transitioning a specific quantity (m) of memory banks into a low-power state based on a memory access count attribute s^{MAC}. As illustrated in Figure 7(b), we opted to implement LPM actions for the first six layers due to the performance trade-off. While switching off memory banks in these layers results in minimal thermal benefits, it concurrently imposes performance penalties. When selecting memory banks to put into LPM using the actions $a^{LPM_{2-15}}$, we follow a top-down approach based on the memory bank layers. This selection means we prioritize selecting memory banks from higher layers before moving on to lower layers. All non-selected memory banks remain in NPM. We determine the LPM state of a memory bank per epoch, and thus a memory bank placed in LPM during the previous epoch may or may not remain in LPM for the next epoch, contingent on the new LPM action chosen by the agent. This strategy has a significantly condensed action space, leading to $25 \cdot 2^{16}$ possible actions for a 4-core 3D-stacked system. Similarly, the strategy reduces the action space from 2^{192} to $25 \cdot 2^{64}$ for a 16-core 3D-stacked system.

4.2.3 Reward. We propose a dedicated reward function for the *3QUTM* architecture, which considers several key performance factors, including core throughput, the peak temperature of 3D-stacked systems, the compound effect of LPM memory banks, and power consumption. The reward function is defined as follows:

$$r_{t+1}(s_t, s_{t+1}) = \zeta r_{0_{t+1}} + \phi r_{1_{t+1}} + \varphi r_{2_{t+1}} + \frac{\Phi}{\left|s_{t+1}^{P_{tot}}\right|} \tag{4}$$

where $r_{t+1}(s_t, s_t + 1)$ represents the immediate reward after the next state s_{t+1} is observed and is capable of capturing all the relevant information of the next state (i.e., core throughput, $s_{t+1}^{T_{peak}}$ and $s_{t+1}^{T_{hotL}}$), and $|s_{t+1}^{P_{tot}}|$ is the total power consumption (memory and core) of 3D-stacked systems in the next decision epoch. It is inevitable to increase power consumption in pursuit of higher performance. However, to save power, we introduce subtle rewards to minimize the urge. We define three criterion functions, r_0, r_1, and r_2, to represent performance, temperature, and the impact of LPM memory banks, respectively. Here, T_{cr} denotes the thermal threshold, and T_{op} represents the expected baseline temperature of the system. Temperatures below these thresholds indicate a higher potential for performance boosting. The coefficients ζ, ϕ, φ, and Φ represent the significance of each reward sub-function. These coefficients are all between 0 and 1, where their sum equals 1. We have assigned a larger value to ζ to underscore the importance of performance. We presume ϕ and φ to be equal, acknowledging their significant impact on thermal headroom. Lastly, Φ is the smallest among all four coefficients to minimize power consumption. We established the parameters ζ, ϕ, φ, and Φ at values of 0.4, 0.25, 0.25, and 0.1, respectively. This deliberate calibration strategically oriented the agent towards comprehending the intricate balance between performance and energy consumption, a correlation illustrated in Figure 7(a). We reward shape based on the transient system temperature and performance metrics. The r_0 sub-function is as follows:

$$r_{0_{t+1}} = \begin{cases} -5 & |s_{t+1}^{T_{peak}}| > T_{cr} \\ \frac{1}{N} \sum_{i=0}^{N-1} U_{t+1_i} \cdot IPC_{t+1_i} \cdot \frac{|s_{t+1}^{f_i}|}{|s^{fmax}|} & |s_{t+1}^{T_{peak}}| \leq T_{cr} \& |s_{t+1}^{T_{hotL}}| > T_{op} \\ \frac{3}{2} \frac{1}{N} \cdot \sum_{i=0}^{N-1} U_{t+1_i} \cdot IPC_{t+1_i} \cdot \frac{|s_{t+1}^{f_i}|}{|s^{fmax}|} & |s_{t+1}^{T_{hotL}}| \leq T_{op} \& |s_{t+1}^{T_{peak}}| \leq T_{cr} \end{cases} \tag{5}$$

wherein IPC_{t+1_i} represents the number of instructions that can execute in a single clock cycle per core. U_{t+1_i} represents the percentage of time that the CPU core is busy executing instructions. The product of these two values estimates the number of instructions executed per unit of time. Finally, multiplying this value by the clock frequency of the CPU core $|s_{t+1}^{f_i}|$ gives an estimate of the core throughput in instructions at the t+1 decision epoch. We normalized to the maximum available frequency for that core $|s^{f_{max}}|$ and averaged the throughputs across all the cores. The value of the sub-reward function r_0 links closely to the average system throughput. Its design promotes high throughput by including an empirically-determined amplification factor of $\frac{3}{2}$ under certain conditions. This factor is applied when the system maintains considerable thermal headroom, i.e., when $|s_{t+1}^{T_{peak}}| < T_{op}$. This design choice assumes that a higher IPC generally leads to improved system performance. Therefore, a positive reward from the r_0 sub-function signifies that the actions contribute to a performance-boosting state. The reward sub-function r_1 is as follows:

$$r_{1_{t+1}} = \begin{cases} -5 & |s_{t+1}^{T_{peak}|} > T_{cr} \\ exp(T_{amb} - |s_{t+1}^{T_{peak}}|) & |s_{t+1}^{T_{peak}}| \leq T_{cr} \& |s_{t+1}^{T_{hotL}}| > T_{op} \\ 2 \cdot exp(T_{amb} - |s_{t+1}^{T_{peak}}|) & |s_{t+1}^{T_{hotL}}| \leq T_{op} \& |s_{t+1}^{T_{peak}}| \leq T_{cr} \end{cases} \tag{6}$$

where T_{amb} is the ambient temperature, and $|s_{t+1}^{T_{peak}}|$ is the peak temperature of 3D-stacked systems measured in decision epoch t+1. Temperature variations often follow an exponential law [17]. Therefore, we design the value of X to tie closely to temperature. We use an exponential form to capture the temperature gap in our reward function. Increasing the temperature can lead to higher performance, provided the system has sufficient thermal headroom. Therefore, we associate it with a higher reward. Empirically, we set this reward multiplier to two for performance enhancement. However, as the temperature climbs and nears the thermal threshold, we decrease the reward gradually. This reduction is due to the expanding gap between T_{amb} and $|s_{t+1}^{T_{peak}}|$. The decreasing reward encourages the agent to moderate the rate of temperature increase to maintain system stability. In other words, we try to prevent the temperature from rising too quickly or reaching the thermal threshold too soon. If the system temperature continues to rise and crosses the thermal threshold, we assign a high penalty to protect the system from potential damage. This penalty serves as a warning to the agents to take corrective actions and prevent the temperature from reaching a critical level. The reward sub-function r_2, is:

$$r_{2_{t+1}} = \begin{cases} \frac{-1}{10\Psi_0} + \frac{-1}{8\Psi_1} + \frac{-1}{6\Psi_2} + \frac{-1}{4\Psi_3} + \frac{-1}{2\Psi_4} & |s_{t+1}^{T_{peak}}| > T_{cr} \\ \frac{1}{\Psi_0} + \frac{1}{\Psi_1} + \frac{1}{\Psi_2} + \frac{1}{\Psi_3} + \frac{1}{\Psi_4} & \sum_{i=0}^{i=N-1}(|s_{t+1}^{IPC_i}| = 1) > \sum_{i=0}^{i=N-1}(|s_{t+1}^{IPC_i}| = 0) \& |s_{t+1}^{T_{peak}}| \leq T_{cr} \\ \frac{1}{2\Psi_0} + \frac{1}{4\Psi_1} + \frac{1}{6\Psi_2} + \frac{1}{8\Psi_3} + \frac{1}{10\Psi_4} & \sum_{i=0}^{i=N-1}(|s_{t+1}^{IPC_i}| = 1) \leq \sum_{i=0}^{i=N-1}(|s_{t+1}^{IPC_i}| = 0) \& |s_{t+1}^{T_{peak}}| \leq T_{cr} \end{cases} \tag{7}$$

where we define $\Psi_k = \sum_{i=0}^{i=M-1}\{|s_{t+1}^{MAC_i}| = k\}$ as the sum of all instances where the number of LPM banks in memory access feature $|s_{t+1}^{MAC}|$ is k, as shown in Figure 8. $\sum(|s_{t+1}^{IPC_i}|)$ represents the sum of the quantized values of IPC attributes. The r_2 sub-reward function reflects the cumulative effect of LPM actions, with the understanding that frequently accessed memory banks switching to LPM would incur more substantial penalties. The coefficient attributed to each term within the reward function signifies its relative significance. Specifically, in scenarios where compute-intensive threads dominate the system, we grant higher rewards. The rationale for utilizing the reciprocal of Ψ (assigning the corresponding term in r_2 to 0 if Ψ equals 0) to represent the reward is to ensure that the r_2 reward falls within a similar range to the other sub-reward functions. A larger Ψ value indicates a higher penalty resulting in a lesser reward. When memory-intensive threads dominate the system, we vary the coefficient for the reward. LPM memory banks that are accessed less frequently are given a relatively higher reward, whereas those that are accessed more often are given a lesser reward. However, symmetrical coefficients penalize the agent during

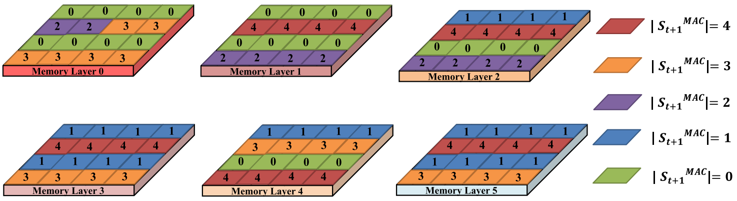


Fig. 8. The value of state feature s_{t+1}^{MAC} when we run the *swaptions* benchmark with 4 core and 128 memory bank. Based on the observation in Section 3, we concentrate the action space in the top six layers.

a thermal emergency. This reward structure ensures a balance between system performance and energy efficiency while factoring in the critical aspects of thermal management.

4.3 Optimal Design, Adaptive Control and Training Techniques for DQNs

Design. Reducing the action space by integrating memory behavior profiling and domain knowledge is an effective strategy. Nevertheless, managing an expanding action space, particularly in the context of increasing cores, poses a substantial challenge. A conventional DQN algorithm generates a distinct Q-value for each action during a single forward pass, with each state-action pair as input. In scenarios where the action space is relatively small, the one-hot encoded representation of Q-values, characterized by one output neuron per action, is a suitable mechanism for network learning. This method facilitates the training process by weight updating to enable the action associated with the highest Q-value. However, this approach becomes untenable when dealing with a large action space. For instance, a 16-core system has an action space of $25 \cdot 2^{64}$, rendering it entirely impractical to store these output neurons. Moreover, selecting the action with the highest Q-value for each training step becomes exceedingly complex.

To further effectively address the surging complexity and scalability of the action space, which is due to the diverse combinations of DVFS settings for each core and LPM choice for memory, we employ a composite approach that incorporates parameterized actions [31] and action embeddings [5]. We treat each core's DVFS setting and the LPM memory choice as separate embedding matrices. These matrices serve as lookup tables, where each row corresponds to a unique DVFS setting (for cores) or a unique memory choice (for memory), and each column corresponds to a dimension $|\mathbb{E}|$ in the embedding space, as illustrated in Figure 9. The size of the action embedding matrix equates to $(N + 1) \times |\mathbb{E}|$, where N is the number of core DVFS settings plus LPM memory choices, and $|\mathbb{E}|$ is the embedding dimension. This embedding dimension, $\mathbb{E}$, is a hyperparameter whose value is experimentally determined via sensitivity analysis, as shown in Table 2. It is evident from the table that an embedding dimension of 16 provides a desirable trade-off between storage overhead, accumulated reward, and episode length.

Take a 16-core 3D-stacked system with 16 DVFS settings per core and 25 LPM knobs. The size of each core's action embedding matrix would then be 16 × 16. Analogously, with 25 LPM knobs, the memory's action embedding matrix would be 25 × 16. The state space dimension is 179 and the action space dimension is 17*16 = 272, the input dimension of the network is 451 - a concatenation of the current state of the 3D-stacked systems and the action embeddings. We select the corresponding DVFS and LPM embeddings from the respective lookup tables for each core at every time step. The concatenated state-action pair (s_t, a_t) is then forwarded through the Q-network, generating a predicted action embedding and the corresponding Q-value. Our Q-network architecture comprises two output layers - the predicted action embedding layer and the corresponding Q-value layer. The initial state of the embedding matrix is determined by small random values, following a Gaussian distribution with a mean of 0 and a small standard deviation (e.g., 0.01). This

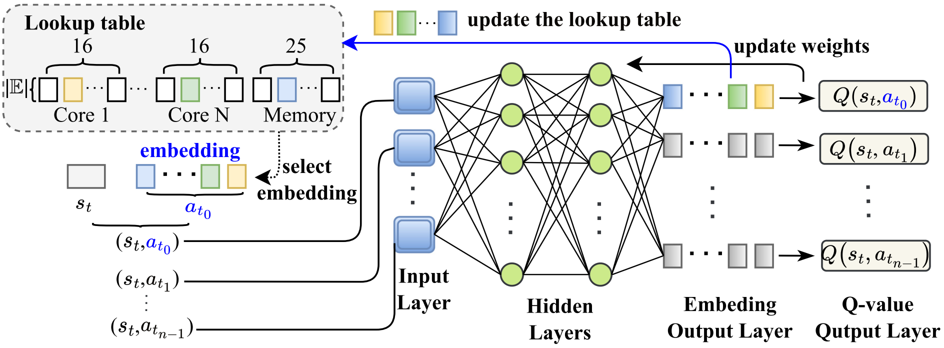


Fig. 9. An abstract diagram depicting the design of the Q-network.

Table 2. The Sensitivity Analysis of Embedding Dimension 𝔼

Embedding Dimension 𝔼	4	8	12	16	20	24	28
Accumulated Reward	374.09	761.94	929.31	1632.25	1652.79	1677.36	1699.66
Episode Length	1730	1850	1950	2010	2200	2390	2650

update process occurs via back-propagation, guided by our dedicated loss function, $L(\theta)$, allowing the action embedding matrices and their corresponding lookup table to be updated jointly with the Q-network.

$$L(\theta) = \frac{1}{2}(Q_{tar} - Q(s_t, a_t; \theta))^2 + \lambda_e \cdot [1 - cosine(\mathbb{E}(a_t), \mathbb{E}(a_p)) \cdot |(Q_{tar} - Q(s_t, a_t; \theta)|] \tag{8}$$

The first term represents the Temporal Difference (TD) error for the Q-values, signifying the gap between the predicted and actual Q-values. The second term corresponds to the cosine similarity between the predicted action embedding $\mathbb{E}(a_p)$ and the actual action embedding $\mathbb{E}(a_t)$. The cosine similarity ranges between -1 and 1, with one indicating that two embedding vectors are identical. Therefore, subtracting the cosine similarity from 1 allows us to treat this term as a form of divergence that we target to minimize. This effect gives rise to a symbiotic relationship between the two components of the loss function. Large TD errors result in a more substantial penalty for the divergence in action embeddings. Contrarily, a diminishing TD error, indicative of improved Q-value predictions, reduces the emphasis on refining action embeddings. Factor λ_e is a hyperparameter that modulates the relative significance of the action embedding loss in contrast to the Q-learning loss. This balance is crucial for harmonizing the simultaneous learning of action embeddings and Q-values. Initially, setting λ_e to 0.01 provides the Q-network with a priority focus on mastering Q-values. As the training progresses, we gradually elevate λ_e in increments of 0.001 until it reaches 1, shifting the focus towards refining action embeddings. This strategy ensures that the model captures the immediate and long-term influences of actions on the system state.

We employ a lightweight fully-connected neural network with few hidden layers to predict the action embedding and Q-value. The activation function used is the *tanh* [11] function. This choice is motivated by its gradual and continuous slope and the potential for capturing the relationship between the input experience tuple and the output Q-values. In addition, it reduces the vanishing gradient problem. We perform L2 regularization to enhance the generalization and engage the *Adam* [21] optimizer to lower memory requirements and improve convergence during training.

We propagate the current state through the Q-network to generate a corresponding action embedding upon model convergence. Subsequently, we compute the cosine similarity between this

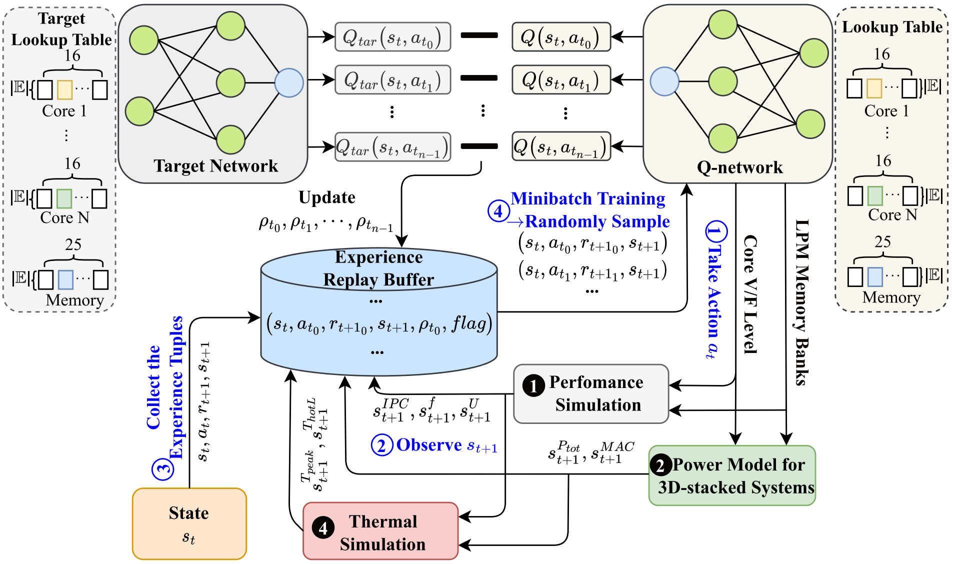

Target Lookup Table
16
Core 1
Core N
25
Memory
Target Network
$Q_{tar}(s_t, a_{t_0})$
$Q_{tar}(s_t, a_{t_1})$
$Q_{tar}(s_t, a_{t_{n-1}})$
$Q(s_t, a_{t_0})$
$Q(s_t, a_{t_1})$
$Q(s_t, a_{t_{n-1}})$
Q-network
Lookup Table
Update
$\rho_{t_0}, \rho_{t_1}, \cdots, \rho_{t_{n-1}}$
④ Minibatch Training →Randomly Sample
$(s_t, a_{t_0}, r_{t+1_0}, s_{t+1})$
$(s_t, a_{t_1}, r_{t+1_1}, s_{t+1})$
Experience Replay Buffer
$(s_t, a_{t_0}, r_{t+1_0}, s_{t+1}, \rho_{t_0}, flag)$
① Take Action a_t
Core V/F Level
LPM Memory Banks
③ Collect the Experience Tuples
$s_t, a_t, r_{t+1}, s_{t+1}$
❶ Perfomance Simulation
$s^{IPC}_{t+1}, s^{f}_{t+1}, s^{U}_{t+1}$
② Observe s_{t+1}
$s^{T_{peak}}_{t+1}, s^{T_{hotL}}_{t+1}$
$s^{P_{tot}}_{t+1}, s^{MAC}_{t+1}$
❷ Power Model for 3D-stacked Systems
State
s_t
❹ Thermal Simulation

Fig. 10. An abstraction depicting the training of 3$QUTM$ on a 3D-stacked processor-memory system.

resulting action embedding and all stored action embeddings in the corresponding action embedding matrix. We then choose the action (i.e., the DVFS setting and LPM operation) corresponding to the action embedding exhibiting the highest cosine similarity for thermal scheduling. The computation of cosine similarity serves as a proxy for traditional Q-value calculations. This proxy emerges as a robust mechanism for action selection, particularly adept at managing the challenges presented by a high-dimensional action space.

However, the run-time overhead is also critical for schedulers. Executing a *tanh* function involves computationally expensive exponentiation and a multiplication operation. Therefore, we perform a piece-wise linear variant of *tanh* [35] (*tanh**). *tanh** has a similar shape to *tanh*, but it is computationally cheaper because it only involves simple linear operations and does not require any expensive transcendental functions. We evaluate these two activation functions, and the *tanh** bears a 3.75% loss of accuracy for a 12.22% reduction in run-time inference time.

Understanding the interaction between cores and memory is crucial within our architecture. Our Q-network structure inherently captures this detail despite using separate lookup tables for each core's and memory's actions. Shared network layers facilitate comprehension of the nuanced correlations between actions across cores and memory, given the process of forward-propagation and subsequent back-propagation. Moreover, temporal learning offers the agent insight into the repercussions of specific actions taken by cores and memory over time while revealing the underlying inter-dependencies. The cumulative effect of all actions is subtly underscored by the global reward signal post-action execution, reinforcing the overall learning process. Therefore, regardless of the independence of action embeddings for each core and memory, our approach indirectly incorporates these interrelationships within the learning framework.

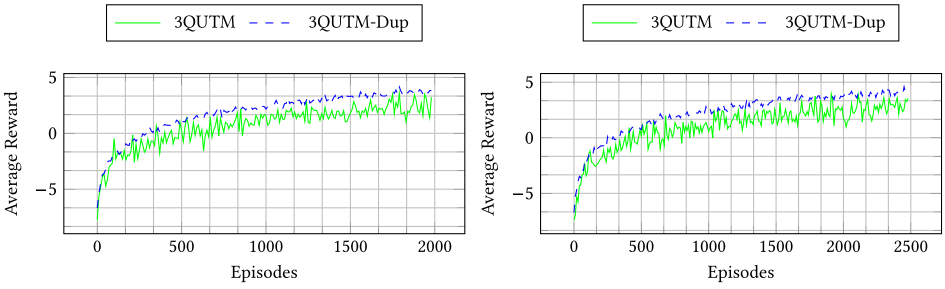
Training algorithm. Figure 10 provides an overview of 3$QUTM$. We use all the benchmarks in the PARSEC-2 suite to generate experience tuples and use them to populate the replay buffer. We conduct training under two distinct scenarios: the first scenario engages homogeneous workloads, where similar benchmarks concurrently deploy on the 3D-stacked system during each training episode. The second scenario introduces heterogeneous workloads wherein different benchmarks arrive in the system as per a Poisson distribution.

ALGORITHM 1: 3QUTM Algorithm

```
Configuration: $T_{cr}, T_{op}, T_{amb}, s^{f}_{max}$, Coefficient of reward function $\zeta, \phi, \varphi, \Phi$, Discount factor $\gamma$, Core size
    $N$, Memory bank size $M$.
Initialization: Reply memory buffer RB to capacity $|RB|$, Neural network $\theta$, Target network $\theta^* = \theta$, Mini-
    batch size $|MB|$, Exploration rate $\varepsilon^{t_0} = 1$, $\varepsilon_{min} = 0.001$, Decay rate $\upsilon = 0.01$, $\lambda_e = 0.01$, Learning rate
    $\alpha_{t_0} = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, Priority $\forall\rho = 0.001$.
1:  $s_{t_0} = \{s_{t_0}^{IPC_0}, \dots, s_{t_0}^{IPC_{N-1}}, s_{t_0}^{f_0}, \dots, s_{t_0}^{f_{N-1}}, s_{t_0}^{U_0}, \dots, s_{t_0}^{U_{N-1}}, s_{t_0}^{T_{peak}}, s_{t_0}^{T_{hotL}}, s_{t_0}^{P_{tot}}, s_{t_0}^{MAC_0}, \dots, s_{t_0}^{MAC_{M-1}}\}$.
2:  for each $t = t_0, t_1 \dots, t_{D-1}$ do
3:      /* $\varepsilon - greedy$ approach
4:      Obtain action $a_t$
5:      Record the state transition $s_t \rightarrow s_{t+1}$
6:      Calculate the reward $r_{t+1}(s_t, s_{t+1})$
7:      Add experience tuple $(s_t, a_t, r_{t+1}, s_{t+1}, \rho_t, flag)$ in $RB$
8:      if $r_{t+1} \leq -5$ or $r_{t+1} \geq 3$ then
9:          /* Duplicate the informative experience tuples
10:         Add copy of experience tuple $(s_t, a_t, r_{t+1}, s_{t+1}, \rho_t, flag)$ into $RB$
11:     if $num((s_t, a_t, r_{t+1}, s_{t+1}, \rho_t, flag)) \in |RB|) > |MB|$ then
12:         Sample random minibatch $|MB|$ experience tuples from $RB$.
13:         /* Freeze $\theta$ parameter during $\delta$ iterations
14:         for each $(s_t, a_t, r_{t+1}, s_{t+1}, \rho_t, flag) \in$ MB do
15:             $Q_{tar} = r_{t+1} + \gamma \underset{a_{t+1}}{max}\{s_{t+1}, a_{t+1}; \theta^*\}$
16:             $\rho_t = |Q_{tar} - Q(s_t, a_t; \theta)|(1 - \lambda_e \cdot cosine(\mathbb{E}(a_t), \mathbb{E}(a_p))) + \eta$ ▹ Update priority of the experience tuple $\rho_t$
17:         /* Perform gradient decent (i.e, Adam optimizer)
18:         $L(\theta) = \frac{1}{2}(Q_{tar} - Q(s, a; \theta))^2 + \lambda_e \cdot [1 - cosine(\mathbb{E}(a_t), \mathbb{E}(a_p))] \cdot |Q_{tar} - Q(s_t, a_t; \theta|$
19:         $\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta)$
20:     $s_t = s_{t+1}$                                   ▹ Environment transitions to next state $s_{t+1}$
21:     $\theta^* = \theta$, $\delta$ initialization                 ▹ Update target network every $\delta$ iterations
22:     $\varepsilon_t = \varepsilon_{min} + (\varepsilon_{t_0} - \varepsilon_{min}) \cdot exp(-\upsilon \cdot t)$                    ▹ $\varepsilon$ decay
23:     $\alpha_t = \alpha_{t_0} * (1 - \beta_2^t)^2/(1 - \beta_1^t)$          ▹ Every minibatch update, learning rate update
```

Algorithm 1 provides a pseudo-code for implementing the training algorithm for 3$QUTM$. The algorithm invokes every scheduling epoch to ensure the system continues to learn and adjust according to the changing states and rewards. During the execution of 3$QUTM$, we initialize the Q-network θ with random weights. In the first training epoch, we obtain the initial state from the profiling module. We set ε to 1, indicating that the agents take a uniform random approach over the V/F levels and varying LPM memory banks. We also observe s_{t+1} and assess the immediate reward $r(s_t, s_{t+1})$, as shown in Lines 4–6. Afterward, we collect the experience tuple $(s_t, a_t, r_{t+1}, s_{t+1})$ and store it in the experience replay buffer *RB*. We duplicate these experience tuples in the replay buffer to emphasize their significance if the reward $r > 3$ indicates the high-performance potential or $r \leq -5$ indicates an increased likelihood of the thermal crisis. When the size of the experience tuples exceeds the minibatch size $|MB|$, we uniformly and randomly select $|MB|$ experience tuples from *RM* to train the DQN model.

To predict the Q-value in DQN, we use both the Q-network θ and a separate target network θ^*. The reason for this is that the target Q-value Q_{tar} is calculated recursively based on the Bellman equation, which depends on the current weights of the Q-network θ. Since these weights update during training, the target Q-value also changes, leading to a moving target problem that can hinder learning. To address this issue, we use the target network θ^*, an identical copy of the Q-network but with fixed weights for a specified number of iterations δ (e.g., $\delta = 1000$), as shown in Line 21. The target network weights periodically update after a fixed number of training iterations by copying the parameters from the Q-network.



(a) Convergence on a 4-core 3D-stacked System (b) Convergence on a 16-core 3D-stacked System

Fig. 11. The learning rates with and without duplicate experience tuples on 3D-stacked systems.

We utilize the *Adam* [21] optimizer to minimize the difference between the predicted $Q(s, a; \theta)$ and target Q_{tar} and iterate the process of updating the weights of the Q-network θ until it reaches convergence, as shown in Lines 17–19. The balance between exploration and exploitation is crucial to achieving the optimal cumulative reward in the training process. We achieve this balance by updating the exploration rate ε over time, as shown in Line 22. The learning rate α is updated for every minibatch in order to improve the efficiency of experience tuple processing, especially in the case of dynamic changes in the environment. This approach leads to faster training and better adaptation capabilities, as detailed in Line 23. We attain a high level of accuracy in predicting the Q-value after training predefined D iterations by utilizing a neural network architecture consisting of three hidden layers, each containing 64, 32, and 16 neurons, respectively.

Prioritized Experience Replay Buffer. We propose a priority scheme [41] for experience tuples to improve the sample efficiency of neural network agents. That is, experience tuples with a high priority, indicating that they are representative, have priority when selecting for training. We leverage the absolute value of the temporal difference (TD) error to quantify the priority, which is the difference between the predicted Q-value $Q(s_t, a_t; \theta)$ and the target Q-value Q_{tar} for a given state-action pair (s_t, a_t). The priority function for an experience tuple i is formulated as follows:

$$\rho_i = |Q_{tar} - Q(s_t, a_t; \theta)|(1 - \lambda_e \cdot cosine(\mathbb{E}(a_t), \mathbb{E}(a_p)) + \eta \tag{9}$$

where η is a small positive constant (i.e., $\eta = 0.001$) to ensure no experience tuple has zero priority. To implement a prioritized experience replay buffer, we use a deque data structure, where each experience tuple consists of state s_t, action a_t, reward r_{t+1}, a boolean variable called the done flag, and a priority value. The done flag indicates whether the current state is a terminal state. Figure 11 displays the advantages associated with duplicative experience tuples. We utilized a t-test to compare average rewards with the p-values of our tests (0.0000006 and 0.000001) in Figure 11, providing evidence that *3QUTM-Dup* offers increased stability and facilitates faster convergence.

Fine-Tuning for Increased Adaptability. When reconfiguring the floorplan of 3D-stacked systems, such as altering the number of cores in a core layer from 4 to 16, the state space undergoes a change. We adapt the network architecture by adding neurons to the input layer and incorporating an additional hidden layer to accommodate this larger input size (i.e., from an input dimension of 223 to 451). However, it is not required to train the Q-network θ from scratch. We leverage pre-existing weights from counterparts of the 4-core system. Specifically, the state space dimension increases from 143 to 179. We retain the weights for the original 143 neurons while the weights of the additional 36 neurons initialize randomly. We construct a four-layer hidden network with a neuron configuration of 128, 64, 32, and 16 per layer. The weights of the newly introduced first layer, consisting of 128 neurons, are initialized randomly. However, the weights of the subsequent

Table 3. Core and Memory Parameters for the Simulated 3D-stacked System

Core Parameters	
Number of Cores	4/16, 1 layer
Core Model	x86, 4.0 GHz, 22 nm, out-of-order
L1 I/D cache	32/32 KB, 4/4-way,64 B-block
L2 cache	private,512 KB, 8-way, 64 B-block
L3 cache	512 KB, 16-way, 64 B-block
Memory Parameters	
3D-stacked Memory	8 GB, 8 layers, 16 channels, 128 banks
Memory Bandwidth	25.6 GB/s

layers directly transfer from the corresponding layers of the pre-trained 4-core system. We load the weights for the memory action embedding matrix and replicate the core action matrix four times to populate the expanded core action embedding matrix. The new Q-network θ can adapt to the new state and action space more efficiently and achieve faster convergence. This strategy of reusing the Q-network θ for a transformed state and action space is especially advantageous as it bypasses the need for extensive training sessions and accelerates the adaptation of the Q-network θ to the new state and action space.

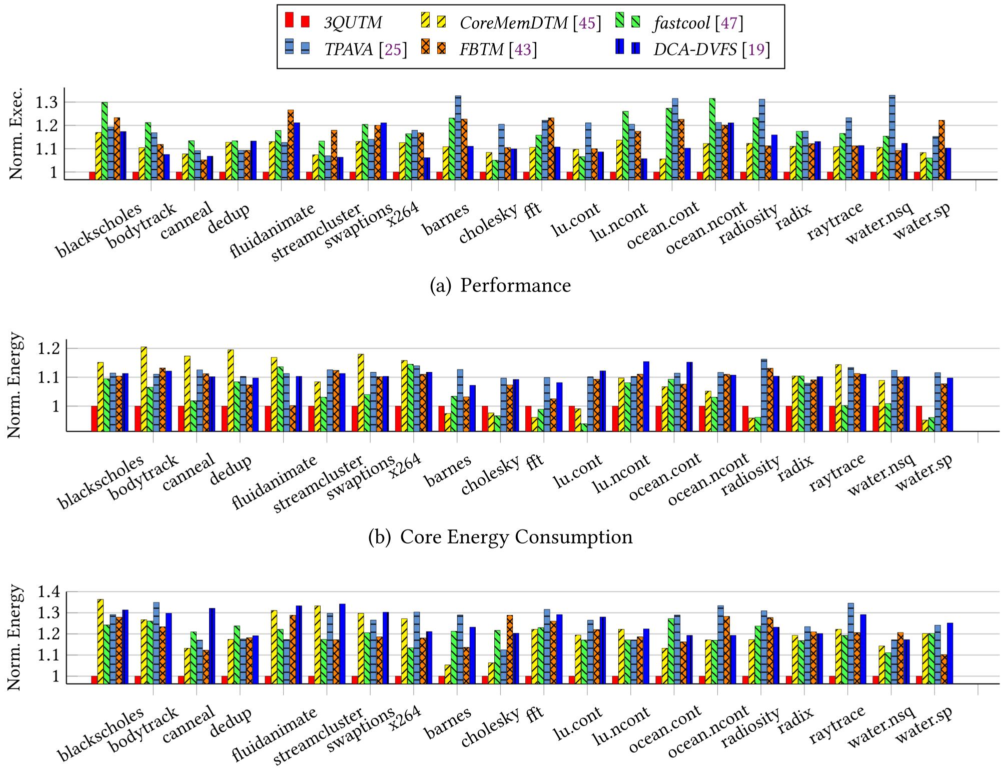
5 EVALUATION

We validate our work using interval thermal simulations via the state-of-the-art *CoMeT* simulator [46] with LPM extensions. *CoMeT* integrates *Sniper* [3], *McPat* [24] with 22nm FinFET and *CACTI* [4], and *HotSpot* [17] into one toolchain for simulating thermals of 3D-stacked systems. Table 3 lists the parameters for the simulated system. For the 4-core 3D-stacked system, the area of each core is 11.65 mm^2, while for the 16-core 3D-stacked system, the area of each core is 2.89 mm^2. Additionally, the area of each memory bank is 2.89 mm^2. We use multi-threaded multi-program workloads from *PARSEC 2.1* [2] (with *sim-medium* inputs) and *SPLASH-2* [49] (with *large* inputs) as the workload. We use all the benchmarks in the benchmark suites that successfully execute to completion using *CoMeT*. We set the $T_{critcal}$, $T_{operate}$, and T_{amb} to 75 ℃, 70 ℃, and 45 ℃, respectively.

Baselines: We compare our work with state-of-art thermal management on 3D-stacked architecture. *CoreMemDTM* is a recent thermal management technique for 2.5D processor-memory systems that synergistically employs DVFS and LPM. We port *CoreMemDTM* [45] to 3D-stacked processor-memory systems by implementing it in *CoMeT*. *CoreMemDTM* is the state-of-the-art technique closest to *3QUTM* in principle. In addition, we port the advanced thermal management technique for cores *TPAVA* [25] and 3D-stacked memory *fastcool* [47] to 3D-stacked processor-memory systems. Furthermore, we consider *FBTM* [43] and *DCA-DVFS* [19], methods originally devised for jointly managing temperature in 3D-stacked caches and cores, as comparative baselines. We tailored the fuzzy logic rules of *FBTM* to govern the 3D-stacked processor-memory systems. Similarly, we adopted a dynamic programming in *DCA-DVFS* to work with 3D-stacked processor-memory systems. The assessment criteria mainly focus on performance, energy consumption, and peak temperature.

5.1 Performance and Energy Improvement on a 4-core 3D-Stacked System

We evaluated our *3QUTM* against five state-of-the-art thermal management policies. *3QUTM* trains using the *PARSEC* benchmarks and then work as a scheduler to perform core DVFS and LPM for the memory bank in tandem in each interval epoch (i.e., 1ms) across *PARSEC* and *SPLASH-2*



(a) Performance

(b) Core Energy Consumption

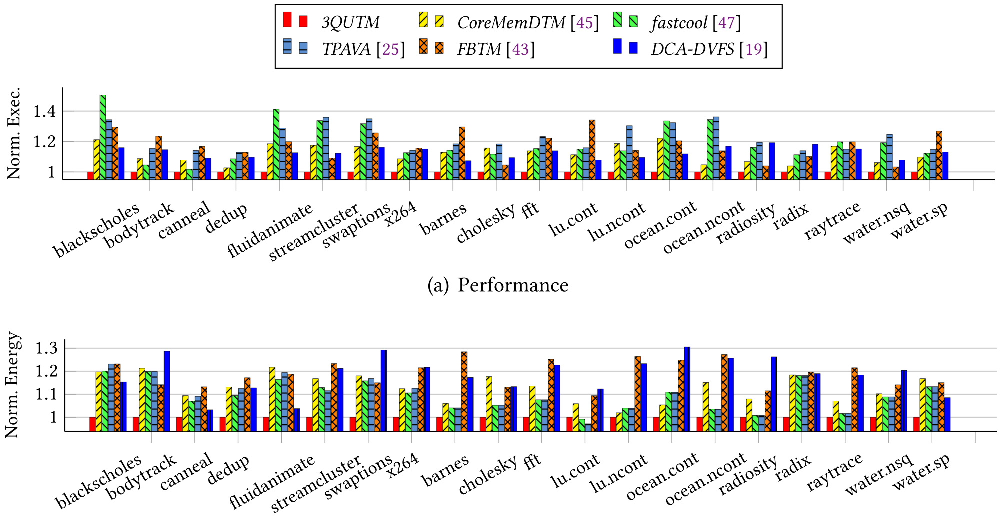
(c) Memory Energy Consumption

Fig. 12. Comparison with state-of-the-art techniques across the *PARSEC* and *SPLASH-2* benchmarks suite on a 4-core 3D-stacked processor-memory system.

benchmarks. The first eight benchmarks are from *PARSEC* with *simmedium* input. The rest are from *SPLASH-2* with *large* input.

The speedup of *3QUTM* over *CoreMemDTM* for high compute intensity benchmarks such as *blacksholes*, *streamcluster*, and *x264* is prominently evident in Figure 12(a). The *3QUTM* over *CoreMemDTM* achieves the highest speedup of 16.96% with *blacksholes*. This speedup is because *blacksholes* is a compute-intensive benchmark that *3QUTM* can learn the performance metric, and its execution time is very short, implying less opportunity for it to overheat. As a result, the chance of triggering the high penalty LPM bank is correspondingly less. The average speedup over *CoreMemDTM* is 11.45%. The average speedup over *fastcool*, *TPAVA*, *FBTM*, and *DCA-DVFS* is 17.89%, 12.33%, 16.36%, and 12.45%, respectively. Our well-trained model inferred the unified thermal management policy on *SPLASH-2* benchmarks. In Figure 12(a), *3QUTM* achieved a maximum speedup of 13.69% against *CoreMemDTM* with an average speedup of 8.83%. Moreover, the average speedup over *fastcool*, *TPAVA*, *FBTM*, and *DCA-DVFS* is 15.32%, 19.35%, 15.13% and 11.67%, respectively.

Figure 12(b) and (c) exhibit core and memory energy consumption, respectively. *3QUTM* obtained an average energy saving of 35.92% compared to *CoreMemDTM* on the *PARSEC* benchmarks. We observe the highest energy saving of 42.87% (core:15.12%, memory:36.33%) on the *blackscholes*, attributed to *3QUTM* learning the memory access pattern and reducing the dynamic and leakage power with LPM and quicker execution time. The *barnes* and *cholesky* benchmarks did not



(a) Performance

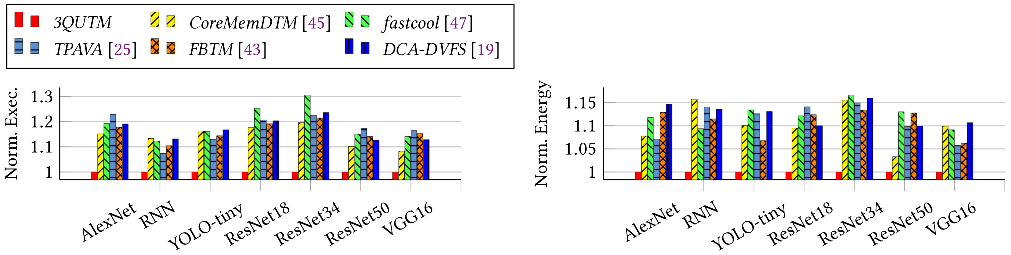
(b) Total Energy Consumption

Fig. 13. Comparison with state-of-the-art techniques across the *PARSEC* and *SPLASH-2* benchmarks suite on a 16-core 3D-stacked processor-memory system.

meet expectations with energy savings of −0.03% and −0.01%, respectively. This observation is because *3QUTM* prioritized performance over energy consumption due to less heat generation. It continuously infers the action that leads to high throughput, making the core run at the full frequency. With sufficient thermal headroom, *3QUTM* made fewer memory banks switch to LPM, thereby trading off performance over energy consumption. *3QUTM* achieved an average of energy savings of 13.22% (*CoreMemDTM*), 16.63% (*fastcool*), 20.76% (*TPAVA*), 17.39% (*FBTM*), and 17.61% (*DCA-DVFS*).

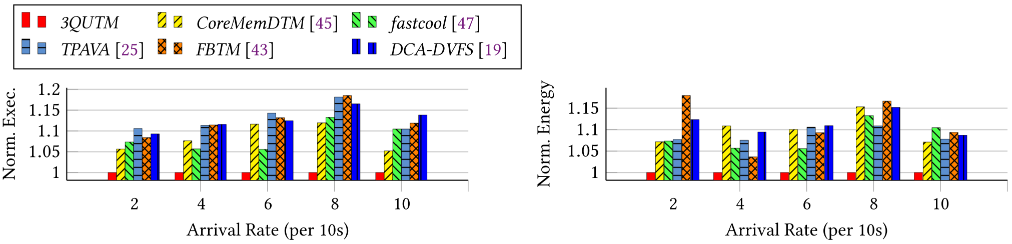
5.2 Performance and Energy Improvement on a 16-core 3D-Stacked System

We evaluated *3QUTM* on a 16-core system, observing modifications in the state space which necessitated fine-tuning of the Q-network. To accomplish this, we utilized a portion of the weights of the Q-network (as presented in Section 5.1) to retrain the model. Notably, the fine-tuned Q-network demonstrated rapid convergence, with results, as shown in Figure 13(a), indicating a considerable speedup over the five other thermal management policies. The *3QUTM* delivered the highest speedup relative to *CoreMemDTM* on the *fluidanimate* benchmark, achieving a speedup of 18.50%, with an average speedup of 11.07%. *3QUTM* performs exceptionally well on high compute intensity benchmarks. Compared with *fastcool*, the *3QUTM* achieved a speedup of 20.05%. We credit this to *3QTM*'s ability to mitigate thermal crises by transitioning the memory bank into LPM without necessitating the deactivation of the memory bank or offloading of data from 3D-stacked memory to 2D DRAM. Furthermore, *3QUTM* surpassed *TPAVA* in performance terms by an average of 18.71%. We attribute this advantage to *3QUTM*'s strategic use of LPM for memory banks, which provides thermal headroom for performance enhancement without the frequent triggering of thermal throttling. On the *SPLASH-2* benchmark suite, *3QUTM* excelled over *CoreMemDTM* by 10.39% in performance on average. More broadly, *3QUTM* outstripped *fastcool*, *TPAVA*, *FBTM*, and *DCA-DVFS* by average margins of 19.72%, 12.54%, 16.90%, and 12.49%, respectively. Notably, as shown in Figure 13(b), *3QUTM* achieved energy savings across all benchmarks.



(a) Performance (b) Total Energy Consumption

Fig. 14. Comparison with state-of-the-art techniques across 7 DNN popular workloads in single-program mode on a 16-core processor-memory system.

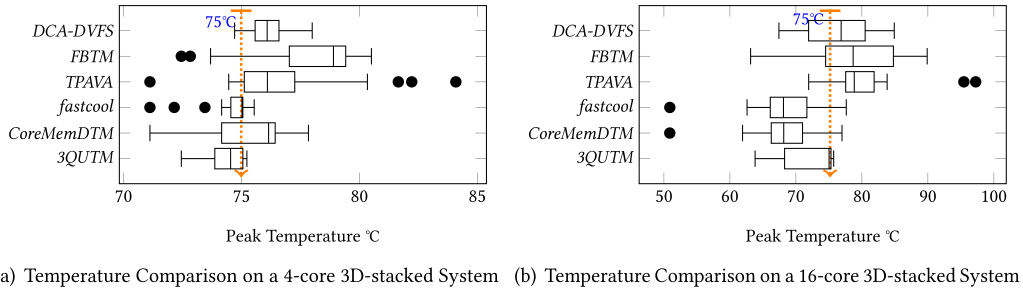


(a) Performance (b) Total Energy Consumption

Fig. 15. Comparison with state-of-the-art techniques across 7 DNN popular workloads executing in multi-program mode on a 16-core processor-memory system.

5.3 Performance and Energy Consumption for DNN Workloads in Inference on a 3D-stacked Processor-memory System

Our evaluation consisted of two experimental groups. The first group entailed homogeneous workloads, specifically threads from a singular DNN workload [38] within the 3D-stacked system. As Figure 14 elucidates, *3QUTM* delivers the highest speedup, especially notable in high compute intensity workloads like *ResNet34*. *3QUTM* achieved an impressive 18.97% and 16.91% speedup over *FBTM* and *DCA-DVFS*, respectively, in terms of performance. *3QUTM* demonstrated 10.85% and 12.55% energy savings compared to *FBTM* and *DCA-DVFS*, respectively. The second experiment group dealt with heterogeneous workloads. We developed 21 DNN workloads, each of which occurred three times. DNN workloads were introduced in 3D-stacked systems at varying rates following a Poisson distribution. *3QUTM* consistently outperforms the other five policies under all load scenarios. The speedup gains with *3QUTM* are most pronounced in medium-loaded systems (with an arrival rate per 10s of 8 DNN workloads). Under these conditions, *3QUTM* achieved performance improvements of up to 11.95%, 13.27%, 18.16%, 18.51%, and 16.72% over *CoreMemDTM*, *fastcool*, *TPAVA*, *FBTM*, and *DCA-DVFS*, respectively. Energy savings with *3QUTM* against the other five policies are also noteworthy, as depicted in Figure 15. The uniqueness of *3QUTM* lies in its ability to learn the intricate interaction between memory and core actions, understanding their intertwined effects on thermal management. This feature distinguishes *3QUTM* from approaches like *FBTM*, which is application-agnostic and cannot account for this interaction, and *DCA-DVFS*, a dynamic programming approach, which, while effective in thermal control, does not exhibit the same learning capacity. The success of *3QUTM* underlines the efficacy of machine learning in efficiently managing thermal conditions and achieving superior performance and energy efficiency.



(a) Temperature Comparison on a 4-core 3D-stacked System (b) Temperature Comparison on a 16-core 3D-stacked System

Fig. 16. Peak temperature comparison between different version of 3D-stacked processor-memory system.

Table 4. Overheads for Varisized Scheduling Epoch on Varisized Version of 3D Processor-memory Systems

Epoch size	1 *ms*		2 *ms*		5 *ms*		10 *ms*	
	4-core	16-core	4-core	16-core	4-core	16-core	4-core	16-core
Overhead	4.52%	6.91%	3.96%	5.11%	1.59%	3.17%	0.79%	1.72%
T_{peak}	75.23℃	75.09℃	75.51℃	75.36℃	76.71℃	76.16℃	77.33℃	76.96℃

5.4 Peak Temperature Analysis

Figure 16 illustrates the comparison of peak temperature between *3QUTM* and the baselines. Figure 16(a) shows that *3QUTM* controls the temperature below the thermal threshold, except for the *ocean.ncont* benchmark, where the peak temperature of *3QUTM* slightly exceeds the thermal threshold at 75.23 ℃. This exception is because *3QUTM* learns unified thermal management at intervals, and it may encounter difficulty in cases where abnormal spikes in power occur within the decision epoch. Additionally, *3QUTM* experiences high fluctuations in power consumption throughout its execution. Similarly, Figure 16(b) demonstrates the performance of *3QUTM* in terms of temperature control on a 16-core 3D-stacked system. The fuzzy control in *FBTM* is less effective in managing complex thermal environments due to its lack of ability to learn from the complex interaction between memory and core actions. Similarly, *DCA-DVFS*, a design-time approach, faces limitations in dynamically managing the thermal environment in an open system.

5.5 Sensitivity Analysis and Overhead Assessment of Scheduling Epoch Size

We performed a sensitivity analysis targeting the scheduling epoch size and its corresponding run-time overhead of *3QUTM*, as presented in Table 4. We translate the well-trained Q-network from *Python* to *C++* and compile it into binary code for execution. We ensured our experimental setup matched the configuration, as outlined in Table 3. Our findings illustrate that increasing the scheduling epoch size from 1 ms to 10 ms reduces the overhead for both 4-core and 16-core systems due to infrequent invocation. However, this reduction in overhead comes at the cost of less control over the peak temperature. This pattern underscores the trade-off between decision frequency (and associated overhead) and the system's peak temperature control. For example, employing a decision epoch of 1 ms on a 4-core system results in *3QUTM* incurring an overhead of 4.52% while sustaining a peak temperature of 75.23℃. In the same conditions, but on a 16-core system, the overhead sees a minor increase to 6.91%. This increment comes from the heightened computational and communication demands of managing several cores. These results demonstrate that *3QUTM* can work across systems with varying core counts while balancing overhead and control.

6 RELATED WORK

Thermal management is a vital area of research for processor-memory systems. There are several approaches for managing the temperature of cores in the literature. These approaches encompass DVFS [10, 37] and task scheduling [8, 9, 14, 18, 20, 23, 26, 39, 40, 50]. Stijn et al. [10] proposed a 3D fine-DVFS algorithm that restricts a core's frequency and voltage to prevent it from negatively affecting neighboring cores. Noltsis et al. [37] used a PID controller to monitor the chip's temperature and adjust DVFS based on temperature differences. Fazal et al. [14] distributed active tasks close to the heat sink to reduce switching activity. Additionally, several researchers have employed innovative learning-based approaches to the thermal management problem. Martin et al. [40] devised a network-based task migration scheduler to determine the best time and place to migrate tasks for optimal performance. Arman et al. [18] developed a reinforcement learning-based DTM policy that considers fan speed, DVFS, and task allocation to optimize performance. Despite the notable progress made in thermal management for processors, these works did not consider memory.

3D-stacked memory (HMC, HBM) has gained significant popularity. Consequently, thermal management for 3D-stacked memory has become a growing area of research. Typical thermal management methods for 3D-stacked memory include page allocation [16, 27, 32], data migration [1, 44, 47], and power regulations [29, 33]. Authors of [16, 32] proposed a memory mapping algorithm that takes both the physical location of the memory and its temperature into account. Lo et al. considered access patterns to map frequently accessed pages to memory banks where the heat dissipates more quickly. Mohammad et al. [13] developed a thermal-aware bandwidth allocation policy. Lokesh et al. [44, 47] migrated data of hot memory channels from 3D memory to off-chip 2D memory and turned off those memory channels to save leakage and dynamic power.

There is limited research on the coordination of thermal management for processor cores and memory. Chen et al. [6, 7] pointed out that synergistically controlling the voltage-frequency levels of cores and DRAMs could achieve higher thermal efficiency than controlling cores only. Jawad et al. [12] proposed a multi-domain power management technique to improve the energy efficiency of mobile SoCs, but they did not jointly perform thermal management for cores and memory. Lokesh et al. [45] attempted to integrate core DVFS and memory LPM for thermal management, but their proposed solution, *CoreMemDTM*, has some limitations. *CoreMemDTM* employs trace simulations to retroactively model the impact on application performance acquired under NPM execution. They also use two different instances of HotSpot [17] for thermal modeling, which limits their ability to model heat transfer between core and memory. Their setup limits their ability to model the power-thermal-performance impact of LPM realistically. Therefore, a unified thermal management solution for 3D-stacked processor-memory systems is still missing.

7 CONCLUSION

We developed extensions for the *CoMeT* simulator to support Low Power Mode (LPM) in 3D-stacked processor-memory systems. This extension allows us to propose a novel thermal management scheduler, *3QUTM*, for 3D-stacked processor-memory systems. *3QUTM* is a Deep Q-Network (DQN)-based learning scheduler that intelligently performs scheduling policy at every decision epoch. The scheduler learns the performance metric and memory access pattern for optimal thermal management. To evaluate the efficacy of *3QUTM*, we conducted experiments using all the benchmarks from *PARSEC* and *SPLASH-2*. We substantiated our approach on popular DNN workloads, leading to an approximate performance gain of 12.37% and energy savings of around 10%. The results indicate that *3QUTM* can effectively perform thermal management, achieving a 10.15% performance speedup and a 13.22% energy savings compared to the state-of-the-art technique

CoreMemDTM. We also verified that *3QUTM* adapts to environmental changes by fine-tuning the Q-network. Overall, our proposed approach, *3QUTM*, is a unified core and memory power regulation scheduler that offers a promising solution to the challenges posed by 3D-stacked processor-memory systems, highlighting the potential for reduced power consumption and enhanced performance in 3D-stacked processor-memory systems.

REFERENCES

[1] Majed Valad Beigi and Gokhan Memik. 2016. TAPAS: Temperature-aware adaptive placement for 3D stacked hybrid caches. In *Proceedings of the Second International Symposium on Memory Systems*. 415–426. https://doi.org/10.1145/2989081.2989085

[2] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*. 72–81. https://doi.org/10.1145/1454115.1454128

[3] Trevor E. Carlson, Wim Heirman, and Lieven Eeckhout. 2011. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12.

[4] Ke Chen, Sheng Li, Naveen Muralimanohar, Jung Ho Ahn, Jay B. Brockman, and Norman P. Jouppi. 2012. CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 33–38. https://doi.org/10.1109/DATE.2012.6176428

[5] Yu Chen, Yingfeng Chen, Zhipeng Hu, Tianpei Yang, Changjie Fan, Yang Yu, and Jianye Hao. 2019. Learning action-transferable policy with action embedding. *arXiv preprint arXiv:1909.02291* (2019).

[6] Yi-Jung Chen, Chia-Lin Yang, Ping-Sheng Lin, and Yi-Chang Lu. 2015. Thermal/performance characterization of CMPs with 3D-stacked DRAMs under synergistic voltage-frequency control of cores and DRAMs. In *Proceedings of the 2015 Conference on Research in Adaptive and Convergent Systems*. 430–436.

[7] Yi-Jung Chen, Chia-Lin Yang, Pin-Sheng Lin, and Yi-Chang Lu. 2016. Opportunities of synergistically adjusting voltage-frequency levels of cores and drams in cmps with 3d-stacked drams for efficient thermal control. *ACM SIGAPP Applied Computing Review* 16, 1 (2016), 26–35.

[8] Yuanqing Cheng, Lei Zhang, Yinhe Han, and Xiaowei Li. 2013. Thermal-constrained task allocation for interconnect energy reduction in 3-D homogeneous MPSoCs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 2 (2013), 239–249. https://doi.org/10.1109/TVLSI.2011.2182067

[9] Ayse Kivilcim Coskun, Tajana Simunic Rosing, Keith A. Whisnant, and Kenny C. Gross. 2008. Temperature-aware MPSoC scheduling for reducing hot spots and gradients. In *2008 Asia and South Pacific Design Automation Conference*. IEEE, 49–54. https://doi.org/10.1109/ASPDAC.2008.4484002

[10] Stijn Eyerman and Lieven Eeckhout. 2011. Fine-grained DVFS using on-chip regulators. *ACM Transactions on Architecture and Code Optimization (TACO)* 8, 1 (2011), 1–24. https://doi.org/10.1145/1952998.1952999

[11] Engui Fan. 2000. Extended tanh-function method and its applications to nonlinear equations. *Physics Letters A* 277, 4-5 (2000), 212–218.

[12] Jawad Haj-Yahya, Mohammed Alser, Jeremie Kim, A. Giray Yağlıkçı, Nandita Vijaykumar, Efraim Rotem, and Onur Mutlu. 2020. SysScale: Exploiting multi-domain dynamic voltage and frequency scaling for energy efficient mobile processors. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 227–240. https://doi.org/10.48550/arXiv.2005.07613

[13] Mohammad Hossein Hajkazemi, Mohammad Khavari Tavana, Tinoosh Mohsenin, and Houman Homayoun. 2017. Heterogeneous HMC+ DDRx memory management for performance-temperature tradeoffs. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 14, 1 (2017), 1–21. https://doi.org/10.1145/3106233

[14] Fazal Hameed, Mohammad Abdullah Al Faruque, and Jörg Henkel. 2011. Dynamic thermal management in 3D multicore architecture through run-time adaptation. In *2011 Design, Automation & Test in Europe*. IEEE, 1–6. https://doi.org/10.1109/DATE.2011.5763053

[15] Greg Hamerly and Charles Elkan. 2003. Learning the k in k-means. *Advances in Neural Information Processing Systems* 16 (2003). https://proceedings.neurips.cc/paper/2003/hash/234833147b97bb6aed53a8f4f1c7a7d8-Abstract.html

[16] Ang-Chih Hsieh and TingTing Hwang. 2013. Thermal-aware memory mapping in 3D designs. *ACM Transactions on Embedded Computing Systems (TECS)* 13, 1 (2013), 1–22. https://doi.org/10.1145/2512457

[17] Wei Huang, Shougata Ghosh, Sivakumar Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R Stan. 2006. HotSpot: A compact thermal modeling methodology for early-stage VLSI design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 14, 5 (2006), 501–513. https://doi.org/10.1109/TVLSI.2006.876103

[18] Arman Iranfar, Federico Terraneo, Gabor Csordas, Marina Zapater, William Fornaciari, and David Atienza. 2020. Dynamic thermal management with proactive fan speed control through reinforcement learning. In *2020 Design,*

Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 418–423. https://doi.org/10.23919/DATE48585.2020.9116510

[19] Kyungsu Kang, Jongpil Jung, Sungjoo Yoo, and Chong-Min Kyung. 2011. Maximizing throughput of temperature-constrained multi-core systems with 3D-stacked cache memory. In *2011 12th International Symposium on Quality Electronic Design*. IEEE, 1–6.

[20] Young Geun Kim, Jeong In Kim, Seung Hun Choi, Seon Young Kim, and Sung Woo Chung. 2019. Temperature-aware adaptive VM allocation in heterogeneous data centers. In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 1–6. https://doi.org/10.1109/ISLPED.2019.8824825

[21] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). https://arxiv.org/abs/1412.6980

[22] Sumeet S. Kumar, Amir Zjajo, and Rene van Leuken. 2017. Fighting dark silicon: Toward realizing efficient thermal-aware 3-D stacked multiprocessors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25, 4 (2017), 1549–1562. https://doi.org/10.1109/TVLSI.2016.2642587

[23] Jie Li, Yuhui Deng, Yi Zhou, Zhen Zhang, Geyong Min, and Xiao Qin. 2022. Towards thermal-aware workload distribution in cloud data centers based on failure models. *IEEE Trans. Comput.* (2022). https://doi.org/10.1109/TC.2022.3158476

[24] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual Ieee/acm International Symposium on Microarchitecture*. 469–480. https://doi.org/10.1145/1669112.1669172

[25] Chien-Hui Liao, Charles H.-P. Wen, and Krishnendu Chakrabarty. 2015. An online thermal-constrained task scheduler for 3D multi-core processors. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 351–356. https://ieeexplore.ieee.org/abstract/document/7092413

[26] Wei Liu, Andrea Calimera, Alberto Macii, Enrico Macii, Alberto Nannarelli, and Massimo Poncino. 2013. Layout-driven post-placement techniques for temperature reduction and thermal gradient minimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 3 (2013), 406–418. https://doi.org/10.1109/TCAD.2012.2228267

[27] Wei-Hen Lo, Kai-zen Liang, and TingTing Hwang. 2016. Thermal-aware dynamic page allocation policy by future access patterns for hybrid memory cube (HMC). In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1084–1089. https://ieeexplore.ieee.org/document/7459470

[28] G. L. Loi, B. Agrawal, N. Srivastava, Sheng-Chih Lin, T. Sherwood, and K. Banerjee. 2006. A thermally-aware performance analysis of vertically integrated (3-D) processor-memory hierarchy. In *2006 43rd ACM/IEEE Design Automation Conference*. 991–996. https://doi.org/10.1145/1146909.1147160

[29] Tiantao Lu, Caleb Serafy, Zhiyuan Yang, and Ankur Srivastava. 2016. Voltage noise induced DRAM soft error reduction technique for 3D-CPUs. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*. 82–87. https://doi.org/10.1145/2934583.2934589

[30] Yanchao Lu, Donghong Wu, Bingsheng He, Xueyan Tang, Jianliang Xu, and Minyi Guo. 2015. Rank-aware dynamic migrations and adaptive demotions for DRAM power management. *IEEE Trans. Comput.* 65, 1 (2015), 187–202.

[31] Warwick Masson, Pravesh Ranchod, and George Konidaris. 2016. Reinforcement learning with parameterized actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[32] Jie Meng and Ayse K Coskun. 2012. Analysis and runtime management of 3D systems with stacked DRAM for boosting energy efficiency. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 611–616. https://doi.org/10.1109/DATE.2012.6176545

[33] Jie Meng, Katsutoshi Kawakami, and Ayse K Coskun. 2012. Optimizing energy efficiency of 3-D multicore systems with stacked DRAM under power and thermal constraints. In *Proceedings of the 49th Annual Design Automation Conference*. 648–655. https://doi.org/10.1145/2228360.2228477

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533. https://www.nature.com/articles/nature14236/?source=post_page

[35] Ashkan Hosseinzadeh Namin, Karl Leboeuf, Roberto Muscedere, Huapeng Wu, and Majid Ahmadi. 2009. Efficient hardware implementation of the hyperbolic tangent sigmoid function. In *2009 IEEE International Symposium on Circuits and Systems*. IEEE, 2117–2120.

[36] Sobhan Niknam, Yixian Shen, Anuj Pathania, and Andy D. Pimentel. 2023. 3D-TTP:Efficient transient temperature-aware power budgeting for 3D-stacked processor-memory systems. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI) 2023*.

[37] Michail Noltsis, Nikolaos Zambelis, Francky Catthoor, and Dimitrios Soudris. 2019. A closed-loop controller to ensure performance and temperature constraints for dynamic applications. *ACM Transactions on Embedded Computing Systems (TECS)* 18, 5 (2019), 1–24. https://doi.org/10.1145/3343030

[38] Shailja Pandey and Preeti Ranjan Panda. 2022. NeuroMap: Efficient task mapping of deep neural networks for dynamic thermal management in high-bandwidth memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 11 (2022), 3602–3613.

[39] Amir M Rahmani, Muhammad Shafique, Axel Jantsch, Pasi Liljeberg, et al. 2018. adBoost: Thermal aware performance boosting through dark silicon patterning. *IEEE Trans. Comput.* 67, 8 (2018), 1062–1077. https://doi.org/10.1109/TC.2018.2805683

[40] Martin Rapp, Anuj Pathania, Tulika Mitra, and Jörg Henkel. 2020. Neural network-based performance prediction for task migration on s-nuca many-cores. *IEEE Trans. Comput.* 70, 10 (2020), 1691–1704. https://doi.org/10.1109/TC.2020.3023022

[41] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).

[42] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)* 42, 3 (2017), 1–21. https://doi.org/10.1145/3068335

[43] Lili Shen, Ning Wu, and Gaizhen Yan. 2020. Fuzzy-based thermal management scheme for 3D chip multicores with stacked caches. *Electronics* 9, 2 (2020), 346.

[44] Lokesh Siddhu, Rajesh Kedia, and Preeti Ranjan Panda. 2020. Leakage-aware dynamic thermal management of 3D memories. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 26, 2 (2020), 1–31. https://doi.org/10.1145/3419468

[45] Lokesh Siddhu, Rajesh Kedia, and Preeti Ranjan Panda. 2022. CoreMemDTM: Integrated processor core and 3D memory dynamic thermal management for improved performance. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1377–1382. https://dl.acm.org/doi/abs/10.5555/3539845.3540166

[46] Lokesh Siddhu, Rajesh Kedia, Shailja Pandey, Martin Rapp, Anuj Pathania, Jörg Henkel, and Preeti Ranjan Panda. 2022. CoMeT: An integrated interval thermal simulation toolchain for 2D, 2.5 D, and 3D processor-memory systems. *ACM Transactions on Architecture and Code Optimization (TACO)* 19, 3 (2022), 1–25. https://doi.org/10.1145/3532185

[47] Lokesh Siddhu and Preeti Ranjan Panda. 2019. FastCool: Leakage aware dynamic thermal management of 3D memories. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 272–275. https://doi.org/10.23919/DATE.2019.8715091

[48] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8 (1992), 279–292. https://link.springer.com/article/10.1007/BF00992698#Abs1

[49] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. 1995. The SPLASH-2 programs: Characterization and methodological considerations. *ACM SIGARCH Computer Architecture News* 23, 2 (1995), 24–36. https://doi.org/10.1145/225830.223990

[50] Xiuyi Zhou, Jun Yang, Yi Xu, Youtao Zhang, and Jianhua Zhao. 2009. Thermal-aware task scheduling for 3D multicore processors. *IEEE Transactions on Parallel and Distributed Systems* 21, 1 (2009), 60–71. https://doi.org/10.1109/TPDS.2009.27

Received 9 March 2023; revised 2 June 2023; accepted 13 July 2023