Temperature-Aware DRAM Cache Management - Relaxing Thermal Constraints in 3D Systems

Minxuan Zhou, Andreas Prodromou, Rui Wang*, Hailong Yang, Depei Qian, and Dean Tullsen

Abstract— High bandwidth 3D-stacked Dynamic Random Access Memory (DRAM) has been proposed to address the memory wall in modern systems, especially when it is used as a large last-level cache (LLC). However, stacking DRAM directly on top of the processor significantly impedes the efficiency of cooling, potentially causing thermal issues both in the processor and DRAM. Dynamic thermal management (DTM) based on DRAM temperature can be heavily intrusive because the normal working temperature for DRAM is lower than the processor temperature limit. This work shows that in many cases it is better to disable hot portions of the cache rather than apply DTM and slow down the processor. Three temperature-aware cache management mechanisms are proposed to decrease the performance impact of DTM on 3D systems. Our experiments show these techniques can improve the performance of DRAM-targeted DTM by 26.1% on average which make 3D systems more practical for future high-performance computing.

I. INTRODUCTION

3D-stacked DRAM technology has been proposed to address the memory wall issues in modern memory-intensive applications because 3D-stacked DRAM provides higher bandwidth than traditional off-chip DDRx DRAM [1], [2]. High bandwidth memory can significantly accelerate emerging applications including graph processing [3], neural network [4], and high-performance computing applications [5]. 3D stacking technologies enable shorter interconnect wire lengths which can significantly reduce the circuit delay and system power dissipation [6], [7]. Therefore, several commercialized products have adopted 3D DRAM technologies. For example, Intel’s Knights Landing architecture has adopted 8 to 16 GB of stacked "near" 3D MCDRAM, a memory architecture similar to Hybrid Memory Cubes [8]. Other examples of HPC-focused architectures that already feature 3D-stacked DRAM include Nvidia’s Tesla P100 [9] and AMD’s FirePro S9300 X2 [10].

Currently, 3D-stacked DRAM is not yet capable of entirely replacing off-chip memory due to its limited capacity and high cost. For instance, MICRON’s HMC [1] provides up to 8 GB capacity per stack and the newest HBM2 [2] provides 8 GB per stack [11]. Due to its capacity limitations, 3D-stacked and off-chip memory technologies often co-exist in a system. Such a configuration is termed “Hybrid Memory Architecture” (HMA). Researchers have proposed the use of the high-bandwidth stacked memory as a large Last Level Cache (LLC) [12]–[15], or as “Part of Memory” (PoM) [16]–[18], where memory addresses can be mapped to off-chip or on-stack DRAM.

Stacking DRAM directly on the processor provides a significant benefit in bandwidth due to vertical through-silicon-vias (TSVs). Building a 3D system is challenging, especially due to thermal concerns. Our thermal simulations on 3D systems show that both the processor and DRAM layers can exceed their respective nominal temperature thresholds for a significant part of execution time with conventional cooling systems. In particular, 3D structures prevent the power-heavy processor from dissipating heat to cooling systems, which raises the temperature of the whole package. Furthermore, even though DRAM itself does not consume enough power to seriously exacerbate the thermal issues, the high temperature of the processor significantly impacts the temperature and operation of the 3D-stacked DRAM. As a result, recent commercial solutions packing the processor and high-bandwidth 3D-stacked DRAM are mostly 2.5D [2], which puts the DRAM stack by the side of the processor instead of above it, sacrificing both latency and throughput and giving up much of the advantage of 3D stacking. Increased die temperature reduces data retention time in DRAM, requiring higher refresh rate to avoid data corruption [19], [20]. For DDR3 DRAM, the normal operating temperature is 85°C [21], [22]. Beyond the normal operating temperature, capacitors discharge at a faster rate and the default memory refresh rate is no longer sufficient to guarantee data correctness. DRAM memories are designed to address the issue by increasing their refresh rate during thermal emergencies [1], [2], which in turn wastes significantly more energy [23], [24].

Therefore, addressing thermal issues becomes critical to make 3D-stacked DRAM feasible in the future. Conventional processors maintain operating temperatures via dynamic thermal management (DTM) solutions such as dynamic voltage-frequency scaling (DVFS) [25]–[27]. DTM mechanisms typically affect performance, even in 2D systems. In 3D systems, the desired operating temperature of DRAM is typically much lower than that of the processor (85°C v.s. 105°C) [28]. Due to disparate temperature sensitivities between DRAM and CPU logic, more aggressive DTM is required to guarantee the temperature limits on all layers of the stack. Our results show that such an aggressive DTM introduces significant performance degradation – from 24% to 52%. We further find that temperature violations on DRAM layers happen more often than the processor, even though both peak and average temperatures are higher on the CPU. Thus, conventional CPU-targeted DTM is not sufficient to protect the memory, while
Our proposed mechanisms are guided by three basic observations: First, the highly-skewed temperature distribution that challenges 3D systems is primarily driven by processor components with varying power density (“hot spots”), combined with their physical distance from the cooling system. Second, the hottest DRAM layers that will first exceed nominal temperature are closer to the processor, while DRAM layers closer to the cooling system remain under the operational temperatures. Third, the lower limit on DRAM temperature is not the result of being more prone to permanent damage at a high temperature, but rather the loss of retention time [29]. Thus, we can let some DRAM areas reach higher temperatures if we are not storing data in them.

Based on these observations, we identify an opportunity to alleviate performance degradation caused by overly aggressive DTM on 3D systems: when faced with a DRAM-based thermal emergency, the system can avoid throttling the CPU. Instead, memory accesses originally addressed to a hot region can be redirected to available cool parts of memory. In other words, we propose mechanisms that trade off LLC capacity for increased CPU frequency, to improve the overall system performance. Our quantitative exploration demonstrates that temporarily disabling regions of our 3D-stacked LLC is often the better decision. In this work we propose three mechanisms that implement our proposal, varying in efficiency, complexity, granularity, and overhead. Compared to a “DRAM-safe”, aggressive DTM solution, our mechanisms improve performance by 21.3%-26.1% on average. Compared to the upper bound – a hypothetical system that ignores thermal violations and never throttles the CPUs – our mechanisms come within 4.5% of the upper bound.

II. MOTIVATION

A. Temperature Distribution in 3D Systems

Figure 1 illustrates the temperature distribution in our 3D system. The highest temperatures are observed in the processor layer and the temperature variations are larger in bottom layers (hotter) than upper layers (cooler). This is due to the different heat dissipation characteristics of the processor and DRAM components. The processor is made of silicon and generates more heat than DRAM, which is made of memory cells that are less power dense. The higher temperature components are closer to the processor layer, while DRAM layers further down the stack remain below 100°C.

B. Dynamic Thermal Management

We assume our system uses a heat sink (which includes a fan) for cooling, located on top of the highest-stacked die. No other active cooling systems are deployed, such as porous silicon or liquid cooling [32]-[35], since they can be cost-prohibitive for 3D implementations. When cooling solutions fail to promptly cool down the whole chip, modern processors employ DTM (e.g., DVFS) to address thermal emergencies. Typically, the system’s temperature is monitored periodically and once it exceeds a predefined threshold, DTM throttles the processor until temperature drops to safe levels. Due to the tight thermal coupling in 3D systems, DTM often throttles the CPU beyond memory requirements, reducing performance.

Figure 2 shows the horizontal and vertical steady-state temperature distribution of our system when running gcc without any thermal management. The variant power density of a core’s components produces uneven horizontal temperature distribution at the base of our 3D system. DRAM has a much smaller power density, thus temperature distribution on DRAM layers is primarily influenced by the underlying hot spots from the processor layer and the temperature variations are larger in bottom layers (hotter) than upper layers (cooler).

Furthermore, temperature drops as we move higher in the 3D stack and closer to cooling. The temperature distribution shows large parts of a 3D system will not exceed the temperature limitation during runtime. This observation provides an opportunity to improve the system performance by utilizing the biased temperature distribution in hot 3D systems. We should note that the peak temperatures (not shown in the steady-state results) we observe during execution of gcc are 120°C on the processor layer and 100°C on the lowest DRAM layer. Thus, without any intervention, the processor and memory layers exceed their normal operating temperatures (by up to 15°C).
coupling between the CPU and memory parts in a 3D system, DTM in a stacked system needs to consider the limitations of all layers as well as the interactions between them. In this work, we assume the DTM throttles via conventional DVFS, dropping one DVFS step each time the threshold is reached.

![Fig. 3. Run-time temperature traces of CPU and DRAM with CPU-safe DTM and a trigger of 104 degrees, running gcc.](image)

1) CPU Temperature Management: We first quantify the problem of CPU-targeted DTM: We execute the gcc benchmark once again. We configure DTM (DVFS) to maintain the temperature across the processor layer below 105°C ($T_{\text{max}}$ of several commercial processors [28]) and measure the peak temperature of our stacked DRAM during execution. Because temperature reacts slowly, DTM typically triggers at a temperature threshold slightly lower than its maximum operating temperature. In this motivating experiment, we set the trigger threshold at 104°C. Figure 3 presents the resulting temperature trace. We first observe that DTM successfully controls the processor layer temperature. Peak DRAM temperature however, exceeds its operating temperature of 85°C for a significant fraction of execution time and rises as high as 90°C. During our experiment, we measure 12% of all memory accesses serviced by hot (unreliable) DRAM regions.

2) DRAM Temperature Management: As a result, DTM in a 3D stacked system needs to consider the temperature requirements of the stacked DRAM. We expand the prior experiment using a similar DTM mechanism where the trigger temperature is set by DRAM temperature. Our experiments show that a trigger temperature set to 83°C keeps all DRAM parts below 85°C. However, DRAM-targeted DTM triggers much more often than CPU-targeted DTM. Our experiments show the DRAM-targeted DTM causes an average 42% performance loss while CPU-targeted DTM only reduces the performance of the system by 12% on average (compared with the system without DTM). Therefore, traditional DTM, even when adjusted for DRAM, is not effective for 3D systems.

III. TEMPERATURE-AWARE CACHE MANAGEMENT

By exploiting the observations in Section II, we can tolerate DRAM temperature violations while allowing the processor to run unthrottled, as long as we are not storing data in hot DRAM regions. We propose a temperature-aware mechanism which enables the system to work even when some parts of the DRAM are disabled.

A. DRAM Cache Organization

Our DRAM cache is organized as a Hybrid Memory Cube (HMC) [1]. Based on HMC specifications, we use 4 DRAM and 1 logic layers, for a total LLC capacity of 4GB. HMC is organized in 32 vaults that “slice” the 4 stacked memory chips vertically. A vault controller located on the logic layer manages traffic to and from all blocks, pages, and sets stored in its vault. We later exploit the independence of vault controllers to optimize our proposed mechanisms.

In this work we assume a page-based, tag-and-data (TAD) 3D-stacked DRAM cache organization, managed using the Unison cache mechanism [12]. We implement a 4-way set associative LRU cache with four pages in each set. Page size is set at 2KB, resulting in 8KB cache sets, thus each set can be stored in a single DRAM row buffer, leading to improved access latency in the case of continuous accesses to the same set. Each page holds 31 data blocks; the 1 remaining block is used as tag store. We further implement Unison’s block-based footprint miss predictor, allowing our cache to load only a subset of cache blocks, which are predicted to be accessed in the near future, inside a page when experiencing a miss. Figure 4 illustrates the simplified model of our LLC organization.

B. Temperature-safe Cache Operation

We introduce two temperature-safe methods to temporarily deactivate hot banks in DRAM cache: DIS and REM.

![Fig. 5. Temperature-safe bank management.](image)

1) Cache Disabling (DIS): Cache disabling is a straightforward solution to guarantee the absence of useful data in hot banks. DIS mechanism temporarily disables DRAM banks...
that exceed the thermal threshold. All accesses to cache blocks of the disabled banks are forwarded directly to the off-chip memory, as illustrated in Figure 5(a). Since the disabled bank will no longer serve any memory operation, data retention is not an issue. Refresh can be entirely disabled in the bank to reduce power consumption without any reliability issue.

2) Cache Remapping (REM): Disabling an entire (e.g. 16MB in HMC [1]) bank will have a significant impact on cache miss rate, particularly if frequently accessed cache blocks become unavailable. Our remapping (REM) mechanism remaps cache blocks to cool DRAM areas that can reliably serve requests. While the cache still loses capacity, cache blocks can still find spots in the LLC.

The process of remapping is shown in Figure 5(b). After remapping, the system will redirect the memory request of the original set to the target set, and look up the requested page by using the tag. In this situation, the number of pages the cache set might serve will be doubled. The cache needs to distinguish the tags of two accesses originally mapped to different sets. As a result, the tag of each cache page is longer than that in a conventional cache not supporting remapping. Extra tag bits denote the original physical location in 3D DRAM. New accesses to data that was previously in the hot bank will miss cache, loading the data into the new cool bank.

When one or both of the cache sets being combined are lightly accessed, the impact on cache miss rate should be minimal. If both are heavily utilized, we will see the impact of increased pressure on that combined cache page. Since it’s possible that all banks are hot in the system, the REM mechanism also includes the disabled status to ensure all data are in temperature-safe places. Specifically, if a hot bank cannot find another bank to remap, it is disabled and accesses off-stack memory for future data requests.

C. Exploration on Cache Remapping

Unlike the DIS mechanism, which is straightforward, there are several opportunities to optimize the REM policies.

1) Target Selection: A key policy decision is the selection of a target for the remapping. A straightforward target is the coolest bank in the system, which has a low probability of becoming hot in the future. This has two benefits. First, it minimizes the risk of remapping itself triggering a thermal event. Second, it reduces the likelihood of a complex remapping event where an oversubscribed bank is deactivated causing two banks’ mappings to be adjusted.

Another potential target is a bank holding data which will become unavailable. With remapping limited to two level, REM-G never ran out of banks, and REM-L reverted to DIS only rarely—so we do not consider higher levels of remapping.

D. Recovery

Since the temperature of any bank will vary based on CPU activities, especially when DTM is used to decrease system temperature, we must also be able to recover the usage of deactivated banks. For the DIS mechanism, it is quite straightforward. If we find that the temperature of a disabled bank is lower than a pre-defined recovery threshold, it will unmark the bank as disabled and enable the processor to start accessing cache sets in this bank again. This condition is also checked at each temperature sensing interval. We set the recovery threshold a little lower than the trigger threshold (85°C) to avoid heavily vacillating between the two states.

For REM, if the temperature of one bank decreases under the recovery threshold, a recovery mechanism is invoked to enable cache sets again for future access. Three situations could happen when a previously hot bank becomes cool: 1. if the bank has not been combined with other banks, the cache sets in the bank will be directly enabled; 2. If the bank is remapped to another bank, we will change the remap table and remap the bank back to its original physical location. In this situation, the dirty blocks that belong to the remapped sets need to be
moved to the original locations. 3. If there was another bank remapped to the bank, we re-enable both of the original two bank mappings that use this bank.

E. Write-back optimization

In order to disable or remap a bank, all dirty cache blocks should be written back before any memory access to these data can be served. The process may lead to a significant performance loss if the number of dirty cache blocks becomes large. Figure 6 shows the average write-back overhead when a global remapping (REM-G) is triggered in various workloads. Our results show the latency of write-back operations would range from tens of micro-seconds to over 1 millisecond. We explore an optimization mechanism to reduce the number of dirty cache blocks, thereby reducing the system-blocked time when triggering cache management.

Since temperature rises relatively slowly, we can typically identify potential hot pages early. We add an extra temperature threshold, which is slightly lower than the hot threshold (85°C). Then, each bank in the system is in one of three states based on two thresholds: cool, dangerous, and hot. The main goal of managing data in dangerous banks is to reduce the significant overhead of write-back operations introduced by thermal-aware cache management mechanisms once the bank becomes hot. The simplest thing we can do is to treat dangerous banks as write-through. This will prevent adding to the number of dirty blocks, but not necessarily decrease it.

More proactively, we can write back blocks on dangerous banks. Doing so immediately, however, just introduces the same overhead, stalling the cache or vault controllers. Instead, we leverage existing accesses to trigger write-back operations in dangerous banks. We can trigger data write-back exclusively on write accesses, or both read and write accesses. We can further write back not just the cache block currently being accessed, but also other cache data residing in the same DRAM row as the accessed block, such as a cache page or a cache set. We bound our write back granularity to a DRAM row, since all its data are contained in a single (currently opened) DRAM row buffer. In this way, we avoid introducing additional overhead from issuing new row activation memory commands.

During the course of our study we explored various other combinations and options, spanning large ranges of three variables: Trigger temperature, access trigger (only writes or both writes and reads) and write-back granularities. We present results of three different flavors of write back optimizations in Section VI: Writes-Block essentially just turns the dangerous bank into a write-through cache. Both-Page and Both-Set trigger on both read and write accesses (row buffer reads), and vary in how aggressively they select blocks for write-back.

We should note none of the proposed operations on dangerous banks changes the already-presented thermal-aware cache management mechanisms. If a hot bank is recovered to serve cache accesses again and the temperature decreases to the "dangerous range", we resume the write-back reduction policies. When the temperature decreases below the dangerous threshold, we can just resume the default cache operations.

IV. MECHANISM IMPLEMENTATION

A. Mechanism Overview

Our temperature-aware cache management methods are designed to complement DTM, not replace it. In fact, the two techniques are quite synergistic – our methods enable DTM to ignore temperature violations that will cause undue throttling of the CPU. In conventional DTM solutions, the system checks the temperature periodically and makes decision in each sensing interval. If the temperature exceeds the predefined trigger threshold, the system throttles to reduce its power consumption. The system recovers its performance when temperature drops below the safe threshold. Our mechanism is checked with the baseline DTM mechanism periodically with a same interval (1 millisecond used in this work).

As shown in Figure 7, the trigger temperature is lower than the critical limit. As analyzed before, the temperature limit is usually the DRAM temperature limit and CPU is still safe at this point. By adopting our methods in the existing DTM, the system can use a higher temperature threshold for throttling events while maintaining system reliability. Specifically, when the temperature reaches the original (DRAM-based) trigger point, the system needs to start protecting the DRAM. Instead of throttling the system, we deactivate the unreliable DRAM area. This allows system temperature to continue increasing to the new trigger temperature, which is set to protect CPU reliability. Only then will DTM kick in to cool down the system. For completeness, the figure shows the temperature decreasing to the lower DRAM trigger, at which point we restore the full capability of the DRAM cache. Overall, this approach allows the system to maintain high frequency over a much wider range of execution scenarios.

B. Temperature Sensing

Our proposed temperature-aware mechanisms operate based on a thermal representation of each memory die. For our exploration we assume the existence of temperature sensors near each bank, which provides accurate thermal representation at
the bank-level granularity. Our assumed implementation is not yet cost-effective, especially for 3D-stacked architectures. However, researchers are already proposing solutions to increase the number of sensors [36], [37]. Prior works also propose models that can extrapolate accurate, fine-grained temperature sensing using the sensors already existing on the CPU layer in combination with the few sensors deployed on DRAM chips [38]–[40]. Finally, according to the HMC specification, there is unused space in the logic layer that could be used for temperature sensing. There are several recent works [41], [42] assuming a per-bank temperature sensing capability which utilizes the temperature sensors with small area overhead [43]. Furthermore, several temperature sensing techniques have been proposed to accurately sense temperature in 3D ICs with low-cost [44]–[46]. Regardless of our assumption for ideal sensing capabilities, all our mechanisms can be easily adapted to operate based on extrapolated thermal models. In other words, the thermal representation is merely an input to our mechanism. The exact method of acquiring these measurements is not central to this work.

C. Bookkeeping – Bank Control Table

![Fig. 8. Local(left) and global (right) bank control table.](image)

Operating at bank-level granularity, our mechanisms require bookkeeping structures to keep track of state changes that lead to triggering of thermal-related actions. Particularly, we propose the addition of a “Bank Control Table”. With the exception of the disable and dangerous (single-bit) flags needed by all three mechanisms, bookkeeping requirements vary between our three proposed solutions. Furthermore, the placement of our BCT structure can affect performance; if placed right after the memory controller and before vault controllers it serializes accesses that could be completed in parallel. Centralized structures can also increase crossbar traffic and power consumption. Decentralized bookkeeping structures are often preferred in memory-related proposals. Figure 8 illustrates the two possible locations to store our BCT.

Our simplest mechanism, DIS, only requires the two common flags (disable, dangerous), which are updated periodically on each sampling interval, depending on the choice of thermal thresholds set in our system. DIS requires minimal bookkeeping overhead: 512 bits are sufficient for our assumed 256 banks, distributed evenly across 32 vault controllers.

REM schemes require additional state information to keep track of remapped and merged banks. The remap field holds the physical bank location of this bank and is initialized to its bank’s ID. In case a bank is remapped, REM schemes update the field to reflect the change. When a remapped bank is accessed, the controller transparently redirects requests to the correct bank based on this field. We further add a combine flag, which uses a single bit to signal that some remote bank has been remapped to this bank. Using the combine flag, we reduce the complexity of remapping management by eliminating the need for reverse searches.

Local remapping (REM-L) requires 3 bits to identify the 8 banks of a single vault in the remap field. Each vault controller maintains its own table since global bank-state information is unnecessary. REM-L requires a total of 224 bytes for BCT, evenly distributed across controllers (7 bytes each). Global remapping on the other hand requires 8-bit fields for bank IDs, leading to a total BCT overhead of 384 bytes. Compared to REM-L, REM-G incurs higher overhead for bookkeeping. Furthermore, a global BCT is conceptually centralized and must be accessed prior to any memory request. With REM-L, a portion of the remap table can be stored near each vault controller. The benefit of doing so is similar to the local DIS table, which can distribute and parallelize accesses to the table.

D. Controllers

To implement our proposed mechanism, a small amount of extra logic needs to be added in the DRAM cache controller and vault controllers. On an L2 cache miss, the on-chip memory controller sends a memory request to the DRAM cache controller to access the data. Since remap requires a longer page tag, the DRAM cache controller will generate a new tag combining the original tag and set number. The set number is used for the cache controller to find the data in the DRAM. The cache controller first extracts the bank and vault number to get the status of the corresponding bank. Following operations depend on what mechanism the system uses.

Thermal-aware cache management operations related to each mechanism are checked every 1ms, which is the temperature sensing and baseline DTM checking interval used in this work. Based on the updated temperature, the corresponding controller (vault controller for local BCT or cache controller for global BCT) updates the BCT based on the mechanism and different temperature thresholds. For each cache access, if the dangerous flag of the target bank is set, the cache controller may initiate one or more write-backs depending on which write-back optimization is used.

E. Mechanism Processing Flow

1) Thermal-aware Cache Management: Thermal-aware cache management operations are checked every 1ms, which is the temperature sampling interval used in this work. Based on the updated temperature, the corresponding controller updates the bank control table based on the mechanism and different temperature thresholds. In DIS, each vault controller identifies
banks that are hot and not disabled. The vault controller notifies the cache controller to write back all valid data in the bank and the vault controller sets the disabled flag. If a disabled bank becomes cool, the vault controller only needs to unset the disabled bit in the table.

In REM-G, if a bank is hot and it has neither been combined nor disabled, the cache controller selects the coolest available bank and updates the remap table. If REM-G fails to find a target bank, the disabled bit in the global REM table will be set by the cache controller, which then writes back dirty cache blocks in the hot bank. The remapping information will not be changed in that case. At the same time, if one disabled bank becomes cool again, the cache controller would change the remapping information in the REM table based on the recovery mechanism. REM-L is similar to REM-G in this scenario except that each vault controller handles these operations.

2) Cache Access: In DIS, each vault controller can access a local (per-vault) DIS table to control the mechanism. Once a memory request comes to a vault controller, it will first check whether the corresponding bank is disabled. If it is disabled, the cache controller directly accesses the data in the off-chip memory. In REM-G, the DRAM cache controller needs to access the global REM table on every memory request, and check whether the corresponding bank is disabled or remapped to another bank. If the bank is disabled, we directly access off-chip memory for the requested data. If the bank is remapped to another bank, the cache controller changes the address bits which indicate the bank index. It then sends the request to the corresponding vault controller of the remapped bank. REM-L is similar to REM-G except all operations are processed by vault controllers.

3) Write-back on Access: For each cache access, if the dangerous flag of the target bank is set, the cache controller may initiate one or more write-backs. If the cache access causes a cache miss, the cache controller would only issue extra commands when the set-level writeback is used. If there is a cache hit, the cache controller would write back dirty cache blocks before the activated DRAM row is replaced.

4) Protection for Hot Banks: To protect the data in the hot bank during the disabling and remapping process, we set the trigger temperature lower than the critical DRAM temperature limit. We empirically choose the trigger temperature such that the temperature of a hot bank will not reach the critical temperature limit before finishing the disabling or remapping operation.

F. Discussion on DRAM Cache Technologies

Besides the Unison cache, more DRAM cache managing techniques can be found in the literature, such as Alloy cache [13]. Our proposed mechanisms are largely orthogonal to the choice of DRAM cache design. Our sole requirement is the existence of translation from memory address to DRAM location (bank-level). For different DRAM cache organization, the cache controller still needs to extract a DRAM vault and bank address from each memory request. Thus, the BCT should be the same. Since we store the full tag for each cache block, the remapping mechanism will also be effective if we only change the address bits indicating the physical DRAM location. For writeback optimization, different cache organizations will support different granularities of write-back on access.

Furthermore, several DRAM caches, including Unison cache, deploy a prefetching mechanism to reduce the overhead of cache miss [12], [47]. In our design, prefetching techniques will remain operational, even during the short temporary increase of memory traffic caused when our mechanisms react to a thermal emergency, and may in fact help hide those delays. The coexistence of prefetchers alongside our mechanisms does not require additional management logic.

V. METHODOLOGY

A. Simulation Infrastructure

We address the experimental needs of our work using a combination of Sniper [48], Ramulator [49], Cacti [50], McPat [51], and Hotspot [52]. During our experiments, control is transferred between these 5 tools, based on a series of carefully synchronized, event-based software interrupts, which form the core of our modifications.

Sniper is a multi-core simulator based on the interval core model and the Graphite simulation infrastructure [53]. We extend Sniper and integrate it with Ramulator: a cycle-accurate DRAM simulator. We configure Ramulator according to our stacked DRAM cache and off-chip memory configurations and we extend it to handle the stacked portion of DRAM as Unison cache. The Sniper-Ramulator combination is sufficient for our performance measurements. We interrupt Sniper on each L2 cache miss, at which point control is transferred to the Ramulator module. Following this control transfer, the internal LLC controller performs necessary transformations dependent on our bookkeeping structures, and finally services the request as a typical DRAM controller. Table I presents the configuration of systems tested in our experiments. In the 3D setting, the aggregated bandwidth provided by 32 vault channels is 512GB/s. In the 2.5D setting, the HMC chip is connected with the host processor by four 16-lane, full-duplex serialized links, which provide up to 320GB/s bandwidth. This data comes from the official HMC specification [1].

We incorporate McPat and Cacti for all our power consumption measurements. We use Cacti to estimate area overhead and power consumption of our SRAM-based bookkeeping structures. Similarly, we use McPat for power estimation for
logic dies (processor and DRAM logic layer) using 22nm technology. The DRAM power consumption is obtained by the published industry DDR4 specification sheets [54]. We model four vertically stacked DRAM layers and one processor layer with 8 cores. Prior to running any experiment, we calibrated our McPat configuration such that results match those of prior work [55].

We utilize HotSpot to acquire runtime temperature traces of the system. HotSpot periodically interrupts simulations and calculates the transient temperature based on prior thermal traces and the new interval’s power consumption measurements. We configure HotSpot intervals to 1ms of simulated time. Table II presents our HotSpot configuration in detail including thermal interface material (layers of material residing between stacked dies for heat dissipation), TSV thermal characteristics and the assumed cooling system. The material characteristics of thermal interface material, TSV, DRAM and Silicon are from a validated previous work [56].

<table>
<thead>
<tr>
<th>Thermal interface material</th>
<th>Specific Heat Capacity</th>
<th>Resistivity</th>
<th>Thickness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4 + 10^6 J/m² K</td>
<td>0.25 mΩ/W</td>
<td>0.02 mm</td>
</tr>
<tr>
<td>TSV</td>
<td>4 + 10^6 J/m² K</td>
<td>0.0058 mΩ/W</td>
<td>0.02 mm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processor and DRAM Silicon</th>
<th>Specific Heat Capacity</th>
<th>Resistivity</th>
<th>Thickness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.75 + 10^6 J/m² K</td>
<td>0.01 mΩ/W</td>
<td>0.15 mm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Heat Sink Specification</th>
<th>Convection Capacitance</th>
<th>Convection Resistance</th>
<th>Thermal Conductivity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>144.4 J/K</td>
<td>0.1 K/W</td>
<td>400 W/mK</td>
</tr>
</tbody>
</table>

TABLE II
THERMAL PARAMETERS

B. Workloads and Baseline Processor

For our simulations, we use 10-billion instruction-long benchmark phases from the SPEC2006 benchmark suite [57]. Most SPEC2006 benchmarks have very small memory footprints, making it challenging to stress-test a 4GB DRAM cache. We address this issue following two approaches: First, we simulate 8 copies of the same benchmark running simultaneously on our 8-core system, essentially multiplying its memory footprint 8-fold. We find that various benchmarks exceed our LLC capacity using this model. Secondly, we present cache size sensitivity experiments, going as far as eliminating the stacked LLC entirely, providing a fair baseline comparison. Our processor layer configuration is presented in Table I.

C. Baseline DTM

The DVFS technology used in this work has five voltage-frequency settings. Frequency decreases by 20% when the V/f setting is decreased by one step. The minimum voltage is 70% of the maximum one which is based on published data of commercial processors [58]. When the peak temperature of one layer exceeds the trigger temperature, the system will decrease the V/f setting by one step unless it’s already the lowest one. In our experiments, the lowest V/f is enough to decrease the temperature of the system without further throttling. When the temperature drops below the trigger temperature, the system increases the V/f setting by one step. Since temperature of both processor and DRAM can affect DTM, we distinguish different DTM policies based on which temperature threshold is used. For example, CPU-105 means the DTM triggers when any part of the processor becomes hotter than 105°C. At the same time, DRAM-85 means DTM triggers when any part of DRAM (not including the DRAM logic layer) becomes hotter than 85°C. In simulation, the latency of changing V/f setting is assumed to be instantaneous – this favors the DTM baseline, which changes frequency more frequently.

VI. EXPERIMENTAL RESULTS

A. Baseline DTM

To understand the effectiveness and cost of a conventional DTM, we examine a sweep of DTM settings, all using the same basic algorithm described in Section V. The variables are (1) whether we are using CPU temperature or DRAM temperature as our trigger, and the trigger temperature. Figure 9 shows the result of this exploration, averaged across all applications. The performance result is shown as average IPC of eight cores in the system, normalized to average IPC of the system without any thermal reactive mechanism. We should note that the IPC results are all scaled to a common cycle time at the end of simulation, which means the IPC results shown can be used to compare results across runs with different DVFS behavior. Hot access rate indicates what percentage of memory requests access data stored in a DRAM area whose temperature is larger than 85°C. For DRAM-targeted DTM (where we monitor DRAM peak temperature, and use that to drive DTM), the range of trigger temperatures is from 80°C to 90°C. The result shows the highest trigger temperature for DRAM-targeted DTM to avoid any hot access in any application is 83°C. At 84°C, the hot access rate is low, which is about 0.1%, but not zero. However, the DRAM-Safe (DRAM83) mechanism suffers 37.2% performance loss.

For CPU-targeted DTM, the tested range of trigger temperatures is from 95°C to 105°C. In this work, the $T_{max}$ of the processor is assumed to be 105°C. We can drive the trigger temperature pretty close to $T_{max}$ without violating CPU limits. In that case, however, many DRAM accesses are unsafe (12.6% and 11.7% in CPU-105 and CPU-104 respectively). We can set the CPU trigger temperature lower, but even at 95°C and 96°C,
the unsafe DRAM accesses are 0.1% and 0.8% while sacrificing 25.7% and 24.0% performance, respectively. We would have to set the CPU trigger lower still to completely eliminate unsafe DRAM accesses, incurring greater performance loss. Based on our experiment, setting CPU trigger to 88°C can eliminate DRAM temperature violations entirely, but incurs a performance loss of 42%.

**B. Overall Results**

Based on the results of the baseline DTM, we use DRAM83 as a comparison point and relabel it as our DRAM-Safe baseline. We will also label CPU104 as CPU-Safe, but we must keep in mind that CPU-Safe is safe for the CPU but not for memory. It serves as an upper bound for reasonable performance. We will also consider IDEAL, which is the full performance with no DTM throttling whatsoever – also unsafe, and a higher upper bound. For more comprehensive comparison, a no-LLC baseline and 2.5D baseline are added in this experiment. Specifically, no-LLC baseline removes 3D-stacked DRAM cache from the tested system and any L2 cache miss will access off-chip memory. The 2.5D baseline still utilizes stacked DRAM as LLC but puts it off-chip with a corresponding higher bus delay. Both no-LLC and 2.5D baseline do not cause DTM to ever be invoked in our simulations.

We will combine our three DRAM management techniques with CPU-Safe, attempting to eliminate the unsafe accesses to DRAM of CPU-Safe without the severe performance loss of DRAM-Safe. Figure 10 shows the performance result of the eight configurations on all applications. Based on the results, all of the three proposed mechanisms outperform the DRAM-Safe approach. Specifically, DIS, REM-L, and REM-G show an average performance improvement over DRAM-Safe of 21.3%, 23.4% and 26.1% respectively. Our mechanisms introduce only a small overhead relative to the CPU-Safe DTM upper bound, being within 8.1%, 6.6%, and 4.5% respectively. Based on the results, our proposed cache management methods can still outperform two no-DTM baseline system. REM-G achieves 11.3% and 22.9% average performance improvement over the 2.5D baseline and the no-LLC baseline, respectively. The latter result was run to verify that our gains are not coming simply because our workloads make poor use of the LLC.

Figure 11 shows the power consumption of different mechanisms used in this work. Compared to the ideal solution without any DTM, the normalized power consumption of DIS, REM-L and REM-G are 0.9, 0.94, and 0.97 respectively. We also calculate the energy consumption based on the power and execution time, our proposed mechanisms consume 7.1%,
9.8%, and 11.0% more energy than the ideal architecture. Such energy consumption overhead mainly comes from the extra memory operations introduced by our mechanisms. As a comparison, DRAM-safe mechanism consumes 9.5% more energy than the ideal architecture. Such results show that our mechanisms provide a comparable energy-efficiency with DRAM-safe mechanism, while significantly increasing the performance. The results also show that these mechanisms control the temperature by effectively limiting the power consumption of the whole system. The extra power is consumed by the memory chip, which contributes much less to the temperature than CPUs.

C. Memory Access Latency

![Image](https://via.placeholder.com/150)

Fig. 15. The results of IPC and miss rate with various CPU-target DTM trigger temperature in gcc(left) and bzip2(right). For each threshold, we test four configurations (from left to right): DIS, REM-L, REM-G, and CPU-Safe.

We further investigate the average memory access latency (AMAL) of three methods and compare them with that of the CPU-Safe baseline. Figure 12 shows the normalized AMAL for each method. The average increase in AMAL of DIS, REM-L, and REM-G are \(2.17\times\), \(1.86\times\), and \(1.50\times\) respectively. To explain such AMAL increase, the cache miss rate results are shown in Figure 13. The geometric means of cache miss rate over all workloads are 21.4%, 17.6%, and 11.2% for DIS, REM-L, and REM-G. Furthermore, Figure 14 shows the frequency of management events for all methods. We record the number of block misses, page misses, and accesses to disabled pages. The results show DIS causes the most block misses because we disable all hot banks. REM-G never disables a bank, which means it can always find a cool bank to remap to. However, remapping cache sets leads to a significant increase in page misses. Since Unison cache employs a footprint predictor to load partial data of a page during a page miss, the negative effect of the page miss is reduced. These results show the three methods have different memory behaviors because of different management policies. Overall, REM-G outperforms the other two methods primarily because of lower cache miss rate resulting from increased flexibility in remapping. The reason why REM-G can always find a target bank for remapping is that the temperature distribution is uneven in not only the vertical but also the horizontal direction. In that case, REM-G remaps the hot banks to the cool banks unless the number of the hot banks in the memory system is larger than that of the cool banks.

D. Temperature-sensitivity Experiment

The choice of trigger temperature is a less obvious one, when combined with our proposed cache management methods, compared to the conventional case. A higher trigger temperature will minimize throttling events in the CPU, but also maximize the number of DRAM banks that are deactivated. More deactivated DRAM banks will minimize LLC capacity and LLC hit rate. Lower trigger temperatures, then, will increase throttling but reduce average memory access time.

Figure 15 shows the results of temperature sensitivity experiments on a memory-moderate workload (gcc) and a memory-intensive workload (bzip2). Both IPC and cache miss rate results for REM-G are better than REM-L and DIS. REM-L also has a higher IPC and lower cache miss rate than DIS. However, in decreasing the trigger temperature for DTM, the gaps for IPC and cache miss rate between different methods shrink because there are fewer banks being deactivated due to the CPU’s lower overall temperature. Overall, though, these results confirm that, despite a drop in LLC miss rate, we achieve the highest throughput with aggressive DTM temperature triggers, sacrificing some memory access latency in exchange for unfettered CPU performance.

E. Write-back Optimization

![Image](https://via.placeholder.com/150)

Fig. 16. The average write-back latency reduction during each cache management in REM-G.

![Image](https://via.placeholder.com/150)

Fig. 17. The overall performance improvement of our most effective write-back mitigation mechanism, T80-Both-Page.

The previous results show that the three proposed mechanisms have already improved the performance over the baseline DRAM-Safe system significantly. However, our results indicate we are still losing performance due to high write-back activity, so there is still an opportunity. Because REM-G shows the best overall performance based on previous experiments, we use it as the baseline method in this section. We now introduce the notion of a dangerous page, as described in Section III-E.

We test three dangerous temperature thresholds: 80°C, 82°C, and 84°C. For each temperature threshold, we test three mechanisms: Write-Block, Both-Page, and Both-Set. Write-block essentially turns the page into a write-through cache for future accesses. The other two potentially trigger multiple writebacks after any LLC access. Figure 16 shows the average write-back latency reduction. Because of the space limitation, we show results for four different workloads which show
different patterns of behavior, each representative of a subset of the full suite. All mechanisms only eliminate a small portion of the latency in bzip2, but reduce over 40% in gcc. Furthermore, the latency reduction of milc is sensitive to access type while bwaves is sensitive to temperature threshold.

We then look into the overall IPC improvement of write-back mechanisms. Figure 17 shows the best performance improvement provided by the tested nine mechanisms, which is T80-Both-Page, compared with REM-G, as well as ideal REM-G which assumes all write-back latencies are zero. On average, we see that the additional gains due to write-back mitigation are not insignificant, but not large; however, in general we achieve half to a third of the upper bound available gains. We found that choosing the best of our policies for each benchmark actually does quite a bit better than this, but that implies a more complex controller.

F. Cache Size Sensitivity Experiments

The pressure on the LLC varies by workload and is also impacted by our choice of an eight-core CPU for our experiments. However, increased cache pressure (which can change the tradeoff between cache size reduction and CPU throttling that is key to this research) can come from several sources, including more intensive workloads, more cores, more threads per core, etc. To capture the effects of an LLC under greater pressure, we examine a variety of LLC sizes.

Specifically, we scale the size of each bank, while keeping the basic HMC structure which consists of 32 vaults and each vault having 8 banks, to provide maximum consistency with our other results. Figure 18 shows the IPC and cache miss rate of one compute-intensive workload (gcc) and one memory-intensive workload (bzip2) on different configurations – other results are similar. For the compute-intensive workload, small cache size significantly increases the LLC miss rate and therefore decrease the IPC for all configurations; however the incremental loss in cache miss rate due to REM-G stays fairly consistent even with small caches. Because we are only impacting the LLC and not the L1 and L2 caches, the impact of the higher LLC miss rates is still somewhat muted. Overall, this means that we continue to see the effectiveness of REM-G even in the presence of higher LLC pressure.

G. Comparison with Thermal-aware Refresh Management

Increasing the refresh rate of DRAM is the traditional method to protect DRAM from the faster data loss caused by high temperature. However, increasing the frequency of refresh operations introduces large overhead in terms of performance and energy consumption in DRAM [23], [59]. Moreover, the negative impact caused by refresh operations may become more severe if the capacity of DRAM increases. Finally, there is no existing technique in the literature that applies different refresh rates in different DRAM areas experiencing different thermal situations. Thus, the refresh rate of the whole DRAM chip would be increased if any part of the memory becomes hot. We then compare our proposed mechanisms with the system which doubles the refresh rate when the temperature exceeds 85 °C.

The energy overhead of increasing refresh rate is estimated based on the execution time of the high temperature phase and the energy consumption of a refresh operation based on published data [54], [59]. For these experiments, we estimate the memory access latency overhead caused by increasing refresh rate after each simulation interval. The estimation is based on the temperature and memory access patterns during the last interval and the expected rate of conflict between access and refresh. The frequency of checking the temperature is the same, and the energy consumption includes the energy consumed by both the logic and the DRAM layers. Figure 19 shows the performance and memory energy consumption of increasing refresh rate compared to REM-G with T80-Both-Page write-back optimization. Because of more frequent refresh operations, thermal-aware DRAM refresh mechanism consumes 7.4% more memory energy in average over all tested applications. Furthermore, enabling a double refresh rate does not provide a better performance than our proposed mechanisms. The results show that our best mechanism provides a slight (1.5%) speedup over the thermal-aware DRAM refresh mechanism while saving a considerable amount of energy.

H. Discussion about Multi-threaded Workloads

This work focuses on multi-program workloads, and analysis of multi-threaded (parallel) workloads is left for future work. However, we expect these solutions will have similar or even better performance when running a multi-threaded workload. In particular, in this work we have modeled a homogeneous multi-programmed workload. We do this because it stresses our experiments in two ways — (1) it tends to heat each core similarly, which reduces our opportunity to remap (a more diverse workload, and resulting thermal pattern, would allow
and effectively helps to (2) increase the overall performance of the CPU and DRAM, and avoids CPU throttling due to high temperature. By applying these proposals to a 3D stacked system, we can expect to see improved performance and lower thermal stress compared to a traditional DRAM system.

VII. RELATED WORK

Several studies in the literature focus on thermal issues in 3D stacked systems. Eckert, et al. [61] explore the thermal feasibility of processing in memory (PIM) based on 3D-stacked DRAM, and show that PIM is thermally feasible even with low-end fanless cooling solutions. However, that work focuses on processors with very low power consumption, while our work explores the impact of 3D thermals in the context of state-of-the-art high-performance processors. Zhu et al. [62] investigate the thermal issues existing in processing in die-stacking memory and show that the host CPU dominates the thermals of a system with a die-stacked PIM accelerator. Unlike their work, this paper assumes a DRAM stack placed directly on the top of the host CPU, which introduces more critical thermal challenges in die-stacking systems. Coskun, et al. [32] utilize microchannel-based liquid cooling technology to control the thermal issues in a 3D stacked architecture and propose a controller to adjust the liquid flow rate to minimize pump energy consumption. Their work is therefore orthogonal to conventional DTM mechanisms, which likely still need to be deployed.

Several works have examined temperature-aware management in memory systems [21], [34], [56], [63]–[68]. Liu, et al. [21] propose three hardware and software schemes to reduce peak temperature on a traditional DRAM chip. Kang, et al. [63] apply runtime cache tuning with per-core DVFS to maximize the performance of chip multiprocessors with 3D-stacked last-level cache memory without thermal-violation constraint. Meng, et al. [56] propose a runtime optimization policy to maximize performance while maintaining power and thermal constraints. TAPAS [64] is a low-cost temperature-aware adaptive block placement and migration policy to reduce access to hot banks for hybrid LLC consisting of STT-RAM and SRAM. There are also several papers focusing on controlling the temperature in systems with die-stacking memory using pipeline control [68], thread migration [65], page allocation [66], and adaptive DVFS [67]. All these previous works focus on peak temperature reduction and cannot completely remove the DTM in the current system. On the contrary, our proposed mechanisms are triggered upon thermal emergencies to improve the performance of conventional DTM and guarantee that DRAM serves memory requests reliably. Furthermore, our methods can be triggered when these temperature-reduction proposals fail to limit the temperature.

Furthermore, disabling parts of the cache was first introduced to reduce cache energy consumption by Albonesi [69]; but the motivation and the cache organization are completely different, necessitating quite different approaches.

VIII. CONCLUSION

This paper explores thermal issues in a 3D system with stacked-DRAM as a large last-level cache combined with a CPU logic layer. This configuration maximizes the throughput and effectiveness of the stacked DRAM for high performance systems, but presents thermal challenges – not just from the cooling standpoint, but also due to the mix of temperature sensitivities between CPU and DRAM. The thermal coupling between the layers, and the different temperature limits for CPU and DRAM make conventional DTM ineffective for controlling temperature in 3D systems. Thus, we propose three DRAM cache management mechanisms, which work with conventional DTM. These allow DTM to only throttle the system for CPU thermal events, instead sacrificing LLC space to eliminate DRAM-induced CPU throttling, enabling the system to maintain close to full performance even in the face of high utilization and high thermal activity. The results show that the proposed mechanisms can improve the performance over DRAM-Safe DTM by up to 26.1% on average which makes 3D systems more practical for future high-performance computing.

IX. ACKNOWLEDGMENT

Rui Wang is support by National Key RD Program of China under grant No.2018YFB0203901. Depei Qian is supported by NSFC under grant No.61732002.

REFERENCES


