Adaptive Bandwidth Management for Performance-Temperature Trade-offs in Heterogeneous HMC+DDRx Memory

Mohammad Hossein Hajkazemi, Michael Chorney, Reyhaneh Jabbarvand Behrouz, Mohammad Khavari Tavana, Houman Homayoun
George Mason University, Fairfax, VA, USA
Email: {mhajkaze, mchorney, rjabbarv, mkhavari, hhomayou}@gmu.edu

ABSTRACT
High fabrication cost per bit and thermal issues are the main reasons that prevent architects from using 3D-DRAM alone as the main memory. In this paper, we address this issue by proposing a heterogeneous memory system that combines a DDRx DRAM with an emerging 3D hybrid memory cube (HMC) technology. Bandwidth and temperature management are the challenging issues for such a memory architecture. To address these challenges, first we introduce a memory page allocation policy for the heterogeneous memory system to maximize performance. Then, using the proposed policy, we introduce a temperature-aware algorithm that adaptively distributes the requested bandwidth between HMC and DDRx DRAM to reduce the thermal hotspot while maintaining high performance. The results show that the proposed memory page allocation policy can utilize the memory bandwidth close to 99% of the ideal bandwidth utilization. Moreover our temperate-aware bandwidth adaptation reduces the average steady-state temperature of the HMC hotspot across various workloads by 4.5^{o}K while incurring 2.5% performance overhead.

Categories and Subject Descriptors
B.3.1 [**Memory Structures**]: Semiconductor Memories–*Dynamic memory (DRAM).*

Keywords
Heterogeneous Memories, Bandwidth, HMC, Temperature

1. INTRODUCTION
3D-integration is a recent technology that addresses the memory wall problem [13][14]. With 3D-integration, different layers of dies are stacked using fast through-silicon TSV interconnects (TSV) with a latency as low as few picoseconds. Therefore, by exploiting 3D-integration, we are able to stack multiple layers of DRAM resulting in shorter memory access latency to potentially address the memory wall problem. Moreover, stacking DRAM gives us the opportunity to have parallel accesses to DRAM banks, which results in higher maximum achievable bandwidth.

Compared to the conventional DRAM architecture (2D), 3D-DRAM results in better performance. Hybrid memory cube (HMC) is an emerging 3D memory interface and design introduced by Micron to address the inefficiency of DDRx DRAMs [14]. HMC stacks up to eight layers of standard DRAM building blocks on a memory controller. However, 3D-integration used in HMC imposes a drastic power density as highlighted in 2013 report by Rambus [20]. Higher power density causes many temperature-related problems including extra cooling costs, reliability, wear-out, and leakage power issues [7]. For example, more stacked layers increases the heat resistivity of the entire chip package that results in higher peak and steady-state temperature. It also complicates the chip packaging process which makes the design more vulnerable to various failure mechanisms [21].

Beside the thermal issues, fabrication cost is another challenge, which could limit the application of HMC. As the capacity of each HMC cube is limited to 2~4GB [9], several cubes need to be chained together to build a larger capacity required. In terms of cost and design feasibility this may not be a practical option. Therefore, conventional 2D, DDRx DRAM, is indispensable in order to maintain high capacity requirement of DRAM to achieve high performance and avoid the thermal and cost challenges associated with the new 3D technology.

A heterogeneous memory system that combines 2D- and 3D-DRAM can exploit the high capacity, low cost and low thermal footprint of 2D, and high bandwidth and low access latency of 3D, simultaneously. However the challenge is how to manage the two substantially different designs effectively to exploit their benefits. [10] attempts to address this issue, however it does not model HMC, and instead, it studies a generic 3D-stacked DRAM. Moreover, despite proposing a policy to achieve higher QoS, the thermal challenge of 3D memory is not addressed in [10].

In this paper we introduce a heterogeneous HMC+DDRx memory system. The focus of this paper is to address both performance and temperature challenges associated with the proposed memory architecture, simultaneously, by introducing performance-temperature aware memory management mechanisms. Over-utilization of either HMC or DDRx DRAM results in bandwidth congestion and incurs a large performance loss. Furthermore, utilizing the HMC to maximize the performance benefits can lead to thermal hotspots, which in turn can severely affects performance, due to thermal emergency response such as throttling. In order to utilize both HMC and DDRx DRAM efficiently, our memory management mechanism allocates the memory pages in an interleaving manner considering the system temperature and performance. To the best of our knowledge this is the first paper to simultaneously address the performance and temperature challenges in a heterogeneous HMC+DDRx DRAM memory subsystem.The main contributions of this work are as follows:

- We show that a heterogeneous HMC+DDRx, is an alternative for conventional DDRx and plain HMC memory system, which

GLSVLSI '15, May 20 - 22, 2015, Pittsburgh, PA, USA.
Copyright 2015 ACM 978-1-4503-3474-7/15/05…$15.00.
http://dx.doi.org/10.1145/2742060.2742070

addresses the performance challenge and thermal issues of 3D-integartion, while maintaining high performance.

- We show that in heterogeneous DDRx+HMC, the average memory access latency changes substantially across various bandwidth allocation, therefore suggests the need for a bandwidth allocation policy to minimize the latency. We propose a run-time memory page allocation policy to efficiently utilize the bandwidth.
- We introduce a dynamic temperature-aware policy that utilizes our proposed heterogeneous DRAM based on the operating temperature of the HMC and the current phase of the workload. As a result, by allocating bandwidth to HMC and DDRx DRAM dynamically, we reduce the steady-state temperature.

The rest of this paper is organized as follows. Section 2 describes our heterogeneous memory system. Section 3 introduces our heterogeneous memory management. Section 4 presents the framework and Section 5 shows the results. Section 6 introduces some related works. Finally, Section 7 concludes the paper.

2. HETEROGENEOUS HMC+DDRx

Prior research [15][16] has shown that 3D-DRAM provides significant advantages in terms of performance while enabling energy-efficient computing. 3D-DRAMs including HMC offer much higher bandwidth compared to DDRx technologies. They are also more power efficient [22][14][12]. This is achieved by having more parallel accesses to the DRAM enabled by short and fast interconnect. However, in terms of cost per pit and relative power density (and temperature footprint) DDRx is a better technology [22][14]. While the HMC cost might even go further down in future, as DRAM is a very cost-sensitive market DDRx will not going away any time soon [12]. Therefore, a memory system consisting of both HMC and DDRx interfaces can address power, performance, temperature, and cost challenges.

Our studied architecture in this work is shown in Fig.1. In our heterogeneous memory system, HMC is combined with a conventional DDRx DRAM to exploit the high memory bandwidth and the low memory latency of the HMC as well as the high capacity and the low cost of the DDRx DRAM. The memory management we employ for the proposed heterogeneous DRAM integrates the OS virtual to physical address translation so that the heterogeneous memory is transparent to the CMP (chip multi-processor) and the cores see a unified address space.

As Fig. 1 illustrates, the cores' memory requests are pushed to the memory request distributer (MRD). Decoding the coming request, MRD transfers the request to the corresponding memory controller (i.e., either HMC or DDRx memory controller). Each controller has its own queue for memory requests. By generating appropriate DRAM commands, the memory controller services the requests in the queue and accesses the DRAM cells. Then, depending on the request type (i.e., read/write) the data is either written to or read from the memory and sent back to the core through the memory controller's read queue. As shown in Fig. 1, our proposed heterogeneous DRAM has two distinct memory channels: one connecting to HMC using two high-speed links, and the other connecting to DDRx DRAM. Without loss of generality, similar to [1][10], we assume in this paper that The HMC and DDRx DRAM employ two and one memory controllers, respectively.

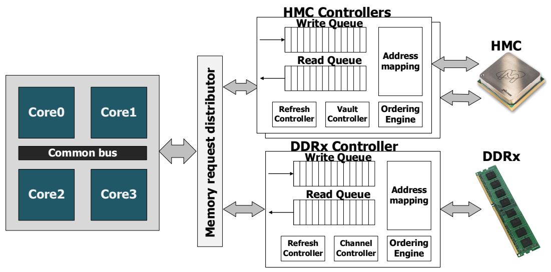


Figure 1. Studied architecture employing heterogeneous DRAM.

The main question for our proposed heterogeneous memory system is how to manage each memory component including the HMC and the DDRx DRAM to gain the best performance while addressing the bandwidth, capacity, and temperature challenges. The key to answer this question is to understand the application's behavior in terms of memory access pattern and utilization. For instance, the more requests the HMC receives in burst, the more its bandwidth is utilized. However, utilizing the HMC aggressively, results in longer memory latency if the application has a large number of memory requests that are coming in burst. On the other hand, applications with a large number of memory requests cause more dynamic power dissipation and thus, higher average temperature. Therefore, a dynamic bandwidth and temperature adaptation is required.

3. HMC+DDRX MANAGEMENT

In this section, we explain how to allocate the application requested memory bandwidth between the HMC and the DDRx DRAM.

3.1 Bandwidth Allocation Policy

Memory access latency is a function of memory bandwidth utilization [1] [10]. As the bandwidth utilization increases, the memory access latency becomes longer, mainly due to congestion in the memory controller and links. While there are several solutions to mitigate this problem [17], above certain bandwidth utilization, due to queuing effect the memory access latency increases significantly [1][10]. In Fig. 2 we investigate this phenomenon for both the DDRx DRAM and the HMC independently. As shown in Fig. 2 we increase the bandwidth utilization of HMC and DDRx DRAM by allocating more number of memory requests for each type of workloads. The x-axis illustrates the memory request portion that each DRAM receives form the entire accesses. For example, in Fig. 2(a) 10/90 means that while 10% of requests are serviced by HMC, 90% of requests are serviced by DDRx DRAM. It is important to note that in Fig. 2(a) and 2(b), we show the average memory latency from HMC and DDRx DRAM perspective, respectively. We categorize application into three groups; the memory-intensive applications, the memory-non-intensive applications and a mixture of them. The workloads are classified based on their LLC misses per 1K instructions (MPKI) which varies from 0.0005 to 24 for our studied benchmarks. We refer to a benchmark as memory-intensive if its MPKI is greater than twelve and non-intensive if MPKI is less than

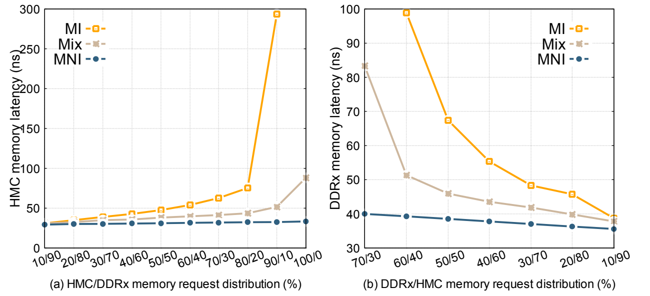


Figure 2. Memory access latency of (a) HMC and (b) DDRx DRAM as a function of memory request allocation.

Table 1. Inaccuracy of proposed bandwidth allocation policy.

Workload	(+/-) inaccuracy %	Workload	(+/-) inaccuracy %
MI1	1.53	Mix3	5.53
MI2	0.33	Mix4	0.74
MI3	0.33	MNI1	2.23
MI4	0.23	MNI2	0.51
Mix1	1.99	MNI3	1.02
Mix2	0.99	MNI4	0.38
Average		1.32	

one. For simplicity, we refer to memory-intensive, memory-non-intensive and mixture applications throughout the paper as MI, MNI and Mix applications. Workloads used in Fig. 2 are representatives of their categories.

As Fig. 2(b) shows, for the DDRx DRAM, the MI workload has the highest rise in memory access latency when request allocation increases from 10% to 60% for DDRx DRAM. For bandwidth above 70%, due to queuing effect the memory access latency for MI workload becomes so large that we could not show it in the figure (for instance with 90% utilization this is 8891ns). In Mix and MNI workloads the memory latency is affected much less as the bandwidth utilization increases. The results show that for MNI workloads, the memory access latency is somewhat linear while for Mix applications it grows exponentially, but at much slower rate compared to MI workloads. We show the results for HMC in Fig. 2(a). As shown, when the memory request allocation is between 10% and 80%, the latency is almost linear across all groups of workloads. For larger bandwidth utilizations, except for MNI, in Mix and MI workloads, HMC latency increases exponentially, however at much slower rate compared to DDRx DRAM (Fig. 2(b)). It is notable that, generally, the memory latency increases in DDRx DRAM more quickly compared to HMC, since HMC has a higher memory bandwidth and faster interconnects, TSVs.

Motivated by the observation from Fig. 2, we introduce a bandwidth allocation policy to effectively utilize both HMC and DDRx DRAM, so that we gain the minimum average memory access latency for any given workload. In this technique, we allocate new memory pages in an interleaving scheme between HMC and the DDRx DRAM, to achieve the minimum average access latency for the entire system. The minimum access latency is achieved at a specific bandwidth utilization of each DRAM which is different across various workloads. We refer to this point as **O**ptimum **B**andwidth **U**tilization or OBU in brief. For instance, for a given workload OBU of 70% means that to achieve the minimum access latency we need to allocate the requests to HMC and DDRx DRAM by 70% and 30% respectively. To satisfy this goal, out of each ten new consecutive writes (page faults), we assign the first seven access (pages) to the first seven free blocks of HMC and the remaining to the three free blocks of the DDRx DRAM. This necessitate a mechanism (using a simple counter) to determine the DRAMs' turn. This helps meeting the OBU for the new incoming write accesses. Nonetheless, since not all the accesses are new writes (i.e., the requested data already resides in the DRAM), and the access pattern to the previously allocated memory blocks may not be uniform, the target bandwidth allocation might not be satisfied. However, our experimental results show that our memory allocation policy can satisfy the target bandwidth, indicating that the access pattern is somewhat uniform. Table 1 reports the average inaccuracy of our allocation technique from the OBU for all other target bandwidths (0 to 100 in step of 10), which indicates how accurate it meets the target bandwidth. As reported in Table 1, the average inaccuracy of the proposed allocation policy is 1.32%, i.e., close to 99% of the ideal bandwidth utilization.

The proposed interleaving memory page allocation policy is shown in Fig. 3. As it shows, upon generating a new request by the CMP, the corresponding core accesses its own TLB and then page table to check whether the address is available in the main memory or not. If so, using MRD, the correspondent DRAM is accessed to read/write the data. Otherwise a page fault occurs, and *bandwidth manager* transfers the page which contains the data from the hard disk to the proper DRAM (i.e., HMC or DDRx). In order to do so, with the help of OS, the bandwidth manager checks whether any of the DRAMs (i.e., HMC and DDRx) has a free page. If any of the DRAMs is full, bandwidth manager accesses the other one that is not full, otherwise it employs page replacement policies to bring the new page to the heterogeneous DRAM. Moreover, bandwidth manager needs to know about DRAM's turn to accommodate the new page in the proper DRAM. This is done with the help of the distribution factor variable that stores the OBU. We will discuss *Temperature-aware distribution factor regulator* in section 3.2.

As discussed, every workload type have different OBU and the interleaving policy results in the minimum memory latency only if the proper bandwidth utilization is set. Therefore, it is important to detect the type of workload, whether it is an intensive, mix or non-intensive, to set the proper OBU. Our studies on workload memory access pattern show that, although the memory access pattern may change through different phases of a program, consistent with prior

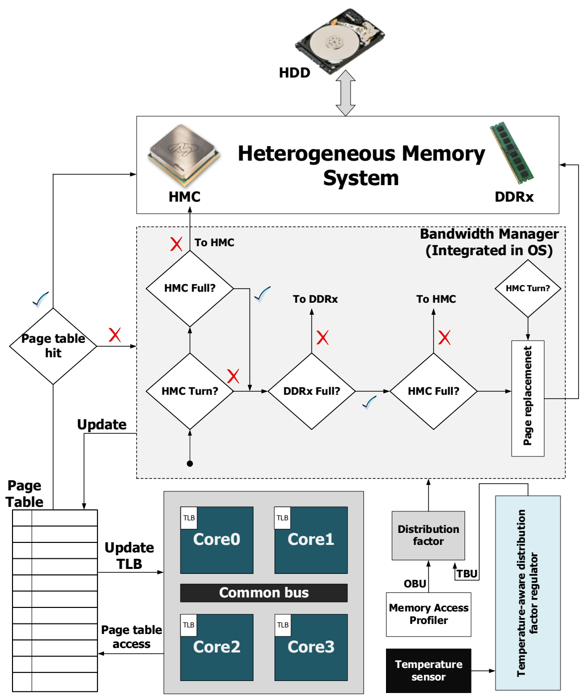


Figure 3. Bandwidth- and temperature-aware memory management.

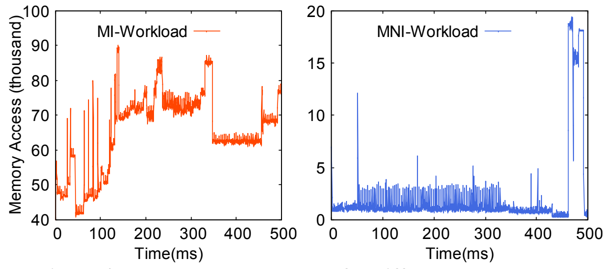


Figure 4. Memory access pattern for different workloads.

work [17], the average intensity of memory requests in a given phase is deterministic and highly predictable. Fig. 4 illustrates the memory access pattern for two representatives of MI and MNI workloads. The samples are collected every 1 million cycles.

As shown in Fig. 4, MNI applications can be clearly distinct from MI workloads, as the number of memory requests in this class of workload remains almost consistently small throughout the 500M cycles studied intervals. Therefore by profiling memory access pattern we can decide the workload type and the relevant OBU accordingly. As Fig. 3 depicts, the *memory access profiler* provides the proper OBU for the bandwidth manager. This can be done every 10 milliseconds, as most operating systems performs context switching at this rate and therefore the memory access pattern will change. After all, as soon as the new page resides in the memory, the corresponding TLB and the page table need to be updated.

Since bandwidth allocation policy brings the memory blocks in page granularity, and given that we use the same page size as homogeneous memory system does, our memory management does not affect the data locality. Applying our memory allocation policy, we estimate the average memory access latency of the proposed heterogeneous memory system when running different workloads using equation 1:

$$L_T = (P * L_{HMC}) + ((1 - P) * L_{DDRx}) \qquad (1)$$

where L_T is the total latency, P is the HMC desired allocated bandwidth, $L_{HMC}$is the HMC latency and L_{DDRx} is the DDRx DRAM latency.

Fig. 5 presents the total memory access latency for two groups of workloads and for various target bandwidths. It is noteworthy that memory access latency (Y axis) for MI and MNI is in the range of microseconds and nanoseconds, respectively. We observe such a high difference in memory latency (us vs. ns) only when the queuing effect occurs in MI workloads. Moreover, as Mix workload behavior is somewhat close to MI workload behavior, due to space limitation, in Fig. 5 we only report the results for the first two studied workloads in MI and MNI categories.

As Fig. 5 shows, different types of workloads have different OBU to achieve minimum average memory access latency. In MI workloads the average memory latency is more sensitive to the bandwidth allocation than the other workload. In Fig. 5(a), for MI workloads, miss utilization of the heterogeneous memory system results in a large performance loss. For example, for the first workload, if the HMC bandwidth allocation is less than 50% or more than 80%, the memory access latency is becoming large in a microsecond range (note that 50% and 80% of HMC allocation means 50% and 20% of DDRx bandwidth allocation). This occurs for the second workload as well, if the HMC bandwidth allocation is less than 30% or equal to 100%. This large penalty is due to the queuing effect. It is important to note that as the simulations for both workloads took so long, we were not able to report the memory access latency for 10% and 20% of HMC bandwidth allocation, in Fig. 5 (a). This shows that the performance loss is even more, compared to 30% of HMC bandwidth allocation. Our observation shows that allocating 60% of the entire bandwidth to HMC gives the best performance for all MI workloads. Therefore, the OBU is set to 60% for this class of workloads. Since Mix workloads show the same behavior as MI workloads do, we set OBU to 60% as well for this class of workloads.

As Fig. 5 (b) presents, in MNI workload, the performance penalty due to DRAMs miss utilization is very small compared to MI workload. Unlike MI and Mix workloads in which we observed the queuing effect, in memory-non-intensive workloads, as we allocate higher bandwidth to HMC we gain a better performance up to the point where we reach to 90% of the entire bandwidth. If we allocate the entire bandwidth to HMC we lose a small performance. Therefore, we can set the OBU at 90% for this class of workloads. Our observation shows that the average memory access latency of our heterogeneous memory system at the OBU for MI, Mix and MNI applications are 64ns, 44ns and 33ns respectively. It is worth mentioning that the workloads that are not presented in these figures have somewhat similar behavior and the illustrated workloads can be representative of its corresponding workload category.

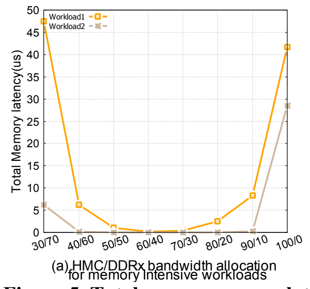
3.2 Temperature-aware Policy

In this section, we propose our algorithm that reduces the steady sate temperature while maintaining the high-performance benefit of bandwidth allocation policy presented earlier. Fig. 6 shows the steady state temperature in HMC as a function of bandwidth allocation, for different types of workloads. As Fig. 6 shows, for the MI and the Mix workloads allocating higher bandwidth to HMC from 10% to 100% results in 25^{o}K and 43^{o}K steady state temperature increase. For workloads with high memory requests (MI) a sharp rise in temperature is observed when higher bandwidth is allocated. As shown, for MNI workload higher bandwidth allocation does not affect the temperature, mainly due to the fact that these workloads do not generate significant memory accesses and therefore they have small power dissipation.

While higher DRAM bandwidth allocation is desired, it comes with a large temperature rise. Such a large thermal rise is not tolerable as it can affect the performance, reliability and the cooling cost of the design [7][21]. Therefore, we need a smart mechanism to dynamically adapt DRAM bandwidth allocation to manage the temperature.

3.2.1 Temperature-aware Bandwidth Allocation

Bandwidth allocation of the heterogeneous DRAM affects DRAM



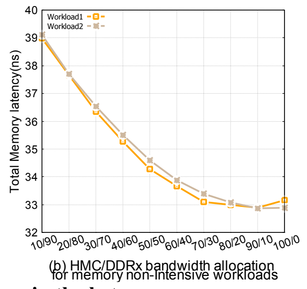


Figure 5. Total memory access latency in the heterogeneous memory system, as a function of HMC/DDRx bandwidth allocation for (a) MI and (b) MNI workloads.

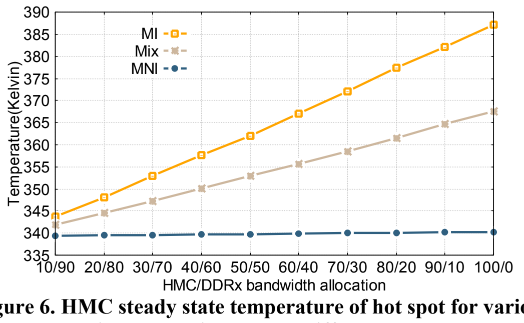


Figure 6. HMC steady state temperature of hot spot for various bandwidth allocations across different workloads.

power dissipation. The power and therefore the temperature of HMC are highly decided by its bandwidth allocation. As indicated in Fig. 6, for MI and Mix workloads there is a large gap in steady state temperature. Motivated by this observation, we propose our **d**ynamic **t**emperature-aware **b**andwidth **a**llocation technique (DTBA in brief) to reduce the steady-state temperature of HMC while maintaining high performance benefit.

In DTBA, first we define two operating temperature regions, namely normal and hot. As long as the HMC operates in the normal region it can be utilized to gain the highest performance using the bandwidth allocation policy. However, whenever HMC enters the hot region we allocate lower bandwidth to it while dedicating higher bandwidth to the DDRx DRAM at the same time to compensate for possible performance loss. This results in lowering HMC power consumption and therefore reduces steady state temperature. We implement DTBA using the proposed memory allocation technique explained in Section 3.1 (see Fig. 3).

As presented in Fig. 6, MNI workloads' temperatures are almost bandwidth insensitive. Therefore these workloads do not require a thermal-aware adaptation and we can simply use the bandwidth allocation technique to manage their bandwidth utilization.

As Fig. 3 shows, our temperature-aware algorithm works as follows. We profile the memory accesses to detect the running workload type. Then, based on the workload type, we set the OBU using bandwidth allocation policy. The temperature sensor on HMC monitors the temperature periodically. If the HMC temperature rises into the hot region, the distribution factor variable is over-written with the new bandwidth referred to as **T**emperature-aware **B**andwidth **U**tilization (TBU). This is done by temperature-aware distributer factor regulator. Otherwise, we continue with the previous bandwidth allocation based on the OBU provided by the memory access profiler. Our temperature-sampling interval is set at 1 millisecond [5][8].

Note that bandwidth allocation policy and DTBA work cooperatively to find the target bandwidth that delivers the highest performance while maintaining the HMC operation below the hot region. As shown in Fig. 5, although allocating 90% of the entire bandwidth to HMC gives the highest performance for MNI workloads, it hurts performance significantly for MI and Mix workloads. Therefore, starting with 60% of bandwidth allocation is an optimal choice as it provides a good performance across all workloads.

4. METHODOLOGY

We use a quad-core CMP architecture with a total of 3GBs of DRAM including 1 GB HMC and 2 GB of DDRx as our target system. For the DDRx DRAM we model a Micron DDR3 SDRAM [3]. Table 2 summarizes the detailed parameters of CMP architecture and heterogeneous memory system modeled in this work.

Table 2. CMP and heterogeneous memory system parameters.

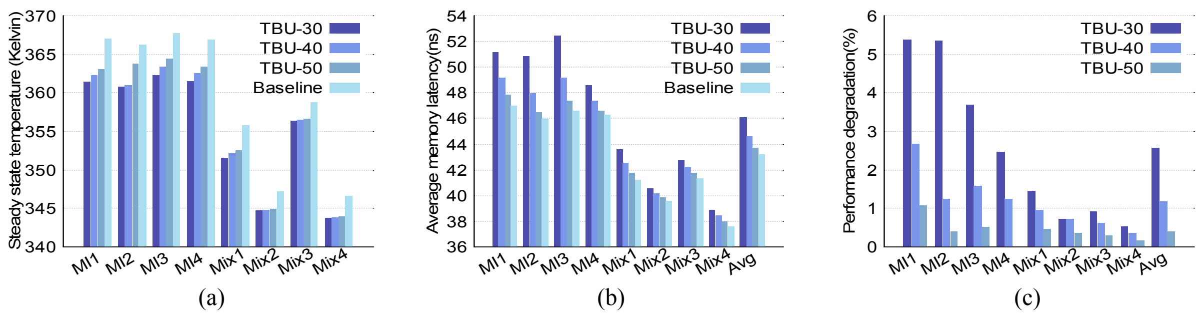
Core	
Core Clock	3GHz
Issue, Commit width	4
INT & FP Instruction queue	32 entries
ROB size, INT Reg, FP Reg	128
L1 cache	64KB, 8-way, 2 cycle
L2 cache	512KB, 20 cycle
HMC and DDRx DRAM	
DRAM Clock	800MHz
Column Access Strobe (t_{CAS})	10 (DDRx), 6 (HMC)
Row Access Strobe (t_{RAS})	24 (DDRx), 24 (HMC)
Row Buffer Policy	Close page
Page Size	4 KB

We integrate SMTSIM simulator [2] and DRAMsim2 [3] for architecture simulation. We use SPEC2000, SPEC2006, NAS [18] and Olden [19] benchmarks to create the 12 studied workloads. We modify DRAMSim2 memory simulator extensively to model the proposed heterogeneous memory. Moreover, DRAMsim2 is extended with a profiler that periodically monitors workload to predict whether it is memory-intensive in the current program phase. It is also equipped with a power profiler to generate the memory system power trace.

To calculate the memory controller power consumption, we use the results reported in [14]. As [14] presents, in an HMC the average power dissipation of the memory controller is 1.8 of the DRAM layers. We employ HotSpot [5] to measure the HMC temperature. DRAMSim2 receives the transient temperature (running temperature) from HotSpot [5] periodically, i.e., every 1 millisecond. We assume that the power dissipation is distributed evenly across all eight DRAM layers, as well as within each layers. We assume the area of the HMC layers including DRAM and controller layers to be 68 mm^2 which is adopted from [6]. We consider the thickness of HMC dies and heat-sink to be 0.05 mm and 0.01 mm, respectively. Other thermal specifications are borrowed from [5]. Similar to [6], since the HMC and CMP are integrated using a PCB, we consider an inexpensive heat-sink for the HMC.

5. DTBA RESULTS

This section evaluates DTBA when applied in the proposed heterogeneous DRAM. To find how effective DTBA can optimize temperature and performance simultaneously, we compare it with a performance-optimized (bandwidth allocation) baseline where the bandwidth adaptation is performed to minimize average DRAM access latency and therefore maximize performance. Hence, the OBU is set to 60%, based on the results discussed in Section 3.1. In order to have a better understating of DTBA impact on temperature, we consider different TBU for the hot region



(a) (b) (c)

Figure 7. (a) Steady-state temperature of HMC, (b) Average latency of the entire DRAM and (c) Performance degradation for different workloads when different TBU is applied.

discussed in Section 3.2. Fig. 7(a) shows the steady state temperature of DTBA. Note that since MNI workloads are not temperature sensitive as discussed earlier, only results for MI and Mix workload are presented.

As shown in Fig. 7(a), TBU=30% configuration achieves the highest temperature reduction. The largest thermal reduction is 5.5°K which is observed in MI4 workload. TBU=40% and TBU=50% results have slightly lower thermal reduction. Moreover, it is important to note that as memory-intensive workloads are more temperature sensitive, temperature results are more sensitive to the TBU compared to Mix workloads. Since DTBA trade-offs temperature with performance, it comes with a small performance penalty compared to the bandwidth allocation policy, which is only optimized for performance. This performance loss is due to the longer memory latency. Fig. 7(b) and 7(c) show the DTBA performance loss for different workloads in terms of memory latency and IPC.

As shown in Fig. 7(b), the average memory access latency increases when DTBA is applied, compared to bandwidth allocation policy. Similar to Fig. 7(a), since there is a negligible performance loss for memory-non-intensive workloads, we do not report the results. As Fig. 7(b) depicts, for all workloads, configurations with more temperature reduction, result in larger memory latency. The largest increase in average memory latency is observed in MI3 workload. Note that, this is the same workload with highest temperature reduction benefit.

As Fig. 7(c) reports, the average performance loss is around 2.5% in the worst case (TBU=30%). The loss in performance is more noticeable in MI workloads. This is consistent with the thermal improvement we show in Fig. 7(a) in which we achieve higher temperature reduction for MI workloads.

6. RELATED WORK

3D stacking can be used in many ways including logic on logic stacking [4], memory on logic stacking [13] and memory on memory stacking [14] to address some of the major challenges microprocessor industry is facing. 3D-DRAM stacking can potentially resolves the memory wall problem and delivers lower power consumption for the memory subsystem.

Although thermal management in 2D deigns for both core and DRAM has been a challenge for architects, introducing 3D stacking even exacerbates the problem. Therefore, many studies have focused to address this issue, especially for stacked memory. These studies either propose static methods at design level [11] or dynamic techniques at runtime [7][13] to reduce the transient or steady state temperature. For instance, [7] proposes a dynamic power and temperature management for a 3D design with stacked cache. Monitoring the runtime application behavior, [13] attempts to choose the best voltage-frequency setting to achieve the maximum throughput while maintaining the power and temperature constraints in 3D multicore system with a stacked DRAM. In a recent work Zhao [8] proposes a migration technique to reduce temperature in a multicore architecture with stacked DRAM. Migrating threads between cores according to their temperature, is the key of their work to reduce the steady state temperature of the system.

[10] proposes a heterogeneous memory management which exploits a stacked DRAM alongside a 2D DRAM. However, unlike to our work, their research does not investigate the thermal characteristics of the design and onl focuses on the quality of service of applications, which also needs the programmer intervention. Another recent work has been on thermal mitigation in hybrid memory cubes (HMC) [6] that tries to reduce the number of read/write burst by compressing data in the logic layer (memory controller). This scheme is orthogonal to ours when used in HMC.

7. CONCLUSION

This paper proposes an adaptive bandwidth allocation and a temperature-aware memory management to exploit the high bandwidth and low latency of 3D hybrid memory cube (HMC) and high capacity and low temperature of the DDRx DRAM. The bandwidth allocation memory management policy profiles workload at run-time and based on memory access pattern allocates DRAM and HMC bandwidth accordingly, to reduce memory bandwidth congestion. While this ensures high performance, it causes significant thermal rise in HMC. To address this challenge, the temperature-aware policy monitors run-time temperature of HMC to adapt the bandwidth. Temperature-aware policy reduces the temperature while maintaining the high-performance benefit of bandwidth allocation technique. This is all done based on application memory access patterns and at run-time. Simulation results show that the bandwidth allocation memory management can utilize the memory bandwidth close to 99% of the ideal bandwidth utilization. Combined with the thermal-aware policy, our proposed memory management reduces steady-state temperature by 4.5°K, on average, across different workloads while maintaining the performance benefits of bandwidth-adaptive technique.

8. REFERENCES

[1] Dong, X., et al. "Simple but effective heterogeneous main memory with on-chip memory controller support" IEEE/ACM SC'2010.

[2] Tullsen, D. M. "Simulation and Modeling of a Simultaneous Multithreading Processor" CMG, Part 2(of 2), pp. 819-828, 1996.

[3] Rosenfeld, P., et al. "DRAMSim2: A Cycle Accurate Memory System Simulator" Computer Architecture Letters, 2011.

[4] Vasileios K., et al. "Enabling Dynamic Heterogeneity Through Core-on-Core Stacking" Proceedings of the 51st Annual Design Automation Conference on Design Automation Conference. ACM, 2014.

[5] Skadron, K., et al. "Temperature-aware microarchitecture," ISCA 2003.

[6] Khurshid, et al. "Data compression for thermal mitigation in the Hybrid Memory Cube," ICCD 2013.

[7] Kang, K., et al. "Temperature-Aware Runtime Power Management for Chip-Multiprocessors with 3-D Stacked Cache" ISQED 2014.

[8] Zhao, D., et al. "Temperature aware thread migration in 3D architecture with stacked DRAM" ISQED 2013.

[9] Hybrid Memory Cube Specification 1.1. 2014.http://hybridmemorycube.org/files/SiteDownloads/HMC%20Rev%201_%20Specification.pdf

[10] Tran L., et al. "Heterogeneous memory management for 3D-DRAM and external DRAM with QoS" ASP-DAC, 2013.

[11] Puttaswamy, K., et al. "Thermal Herding: Microarchitecture Techniques for Controlling Hotspots in 3D-Integrated Processors" HPCA 2007.

[12] Elsasser, W., 5 Emerging DRAM Interfaces You Should Know for Your Next Design, Cadence white paper, 2013.

[13] Meng, J., et al. "Optimizing energy efficiency of 3-D multicore systems with stacked DRAM under power and thermal constraints", DAC 2012.

[14] Jeddeloh, J., et al. "Hybrid Memory Cube – New DRAM Architecture Increases Density and Performance", VLSIT 2012.

[15] Wu, Q., et al., "Impacts of though-DRAM vias in 3D processor-DRAM integrated systems" IEEE 3DIC 2009.

[16] Kang, U., et al. "8 Gb 3-D DDR3 DRAM using through-silicon-via technology" Solid-State Circuits, IEEE Journal of 45, 2010.

[17] Kim, Y., et al. "ATLAS: A scalable and high-performance scheduling algorithm for multiple memory controllers" HPCA 2010.

[18] Bailey, D. H., et al. "The NAS parallel benchmarks" International Journal of High Performance Computing Applications 5.3 (1991): 63-73.

[19] Rogers, A., et al. "Supporting dynamic data structures on distributed-memory machines" ACM TOPLAS 17.2 (1995): 233-263.

[20] Li M., 3D Packaging for Memory Application, Rambus, 2013. http://www.avsusergroups.org/joint_pdfs/2013_6Li.pdf

[21] Srinivasan, J., et al. "Lifetime reliability: Toward an architectural solution" Micro, IEEE 25.3 (2005): 70-80.

[22] Pawlowski, J. T. "Hybrid Memory Cube: Breakthrough DRAM Performance with a Fundamentally Re-Architected DRAM Subsystem" In Proceedings of the 23rd Hot Chips Symposium, 2011.