Data Compression for Thermal Mitigation in the Hybrid Memory Cube

Mushfique Junayed Khurshid
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, Wisconsin 53706
Email: khurshid@wisc.edu

Mikko Lipasti
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, Wisconsin 53706
Email: mikko@engr.wisc.edu

***Abstract*—Main memory performance is becoming an increasingly important factor contributing to overall system performance, especially due to the so-called memory wall. The Hybrid Memory Cube (HMC) is an attempt to overcome this memory wall, consisting of DRAM dies stacked on top of each other, with a logic die at the bottom - all interconnected with highly dense low latency through silicon vias (TSVs). But modelling the Hybrid Memory Cube in HotSpot has indicated that this cube has a natural temperature variation, with the hottest layers at the bottom and the cooler layers at the top. High temperatures and variations within a DRAM can result in reduced performance and efficiency, especially when dynamic thermal management (DTM) schemes are used to throttle DRAM bandwidth whenever temperature gets too high. Hence this paper attempts to reduce the maximum temperature and also this variation, by using data compression - where the compression is performed on the on chip memory controller, and the compressed blocks are read/written using fewer bursts in the Hybrid Memory Cube, hence reducing power dissipation. Compressed blocks are stored only in the hotter banks of the cube to mitigate the thermal gradient in the cube. Maximum temperature was reduced by as much as $6\,^{\circ}\mathrm{C}$, and since the HMC spent lesser time throttling when DTM schemes were used, a maximum of 14.2% speed up was observed, at an average of 2.8%.**

I. Introduction

The Hybrid Memory Cube (HMC), implemented by Micron [1] is a memory architecture, which has DRAM dies stacked on top of each other, with a logic die at the bottom, all interconnected via low latency through silicon vias - an architecture targetting the memory wall problem. A thermal model of the HMC was constructed using HotSpot [4] to observe the variation in temperature within the cube. Initial simulations show that the cube has an inherent temperature variation within itself, as can be observed in Figure 1. Details of the HotSpot model are in Section IV. High temperatures as well as temperature variation in the DRAM can have detrimental effects on reliability and performance of the DRAM. When the maximum temperature in the DRAM reaches a critical level, memory traffic is required to be throttled down to prevent DRAM from reaching damaging temperatures. Whenever DRAM is throttled, it further exacerbates the memory wall problem and affects performance. Hence any reduction in temperature within the cube would help minimize the time DRAM is in "throttled mode", resulting in improved performance. Furthermore, when large temperature variations exists, overall throttling of the DRAM can result in inefficient usage of the DRAM, since even cooler memory banks are also throttled down as demonstrated by Liu et al. [5]. Hence, in this paper, an attempt is made to mitigate the maximum temperature in order to minimize throttling of DRAM as much as possible. Data compression is proposed to solve this temperature problem within the HMC. No compression/decompression is performed in the logic die, as that will further worsen the temperature problem, but is moved to the on chip memory controller (processor side) instead, which communicates with the logic die of the HMC. When compressed blocks are read from/written to the HMC banks, fewer bursts are required, thereby reducing energy consumption. These energy savings are used to reduce maximum temperature within the cube. Compression is only used for hotter parts within the cube, to reduce temperature variation in the cube as much as possible. This technique is evaluated using SPEC2006 benchmarks, and its effects on DRAM performance and temperature is observed, in the context of two DTM schemes applied to the HMC.

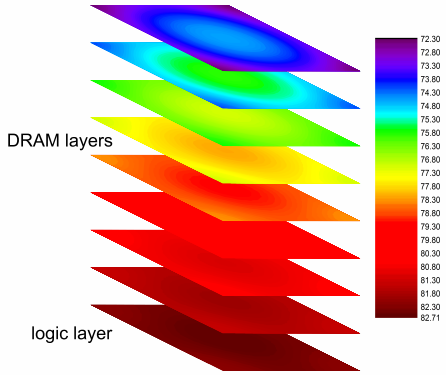

Fig. 1. HMC temperature distribution under maximum power dissipation conditions (a snapshot in midst of execution)

II. Background

A. The Hybrid Memory Cube (HMC)

The Hybrid Memory Cube (HMC) [1] is a heterogenous 3D stack of DRAM dies on top of a logic die. The DRAM layers are of size 1 Gb, divided into 16 partitions. Each partition has 2 banks (also called arrays), and its own data and control through silicon vias (TSVs).Figure 2 shows the structure of

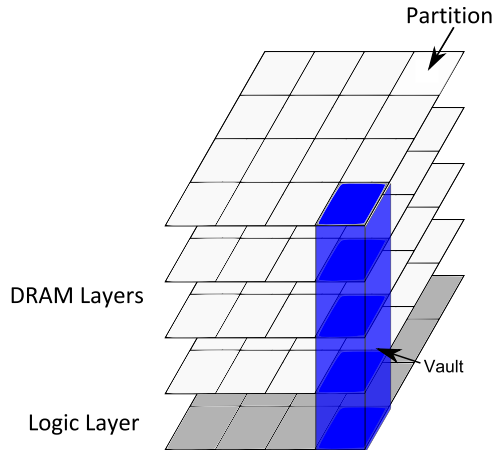

Fig. 2. Hybrid Memory Cube structure (512MB)

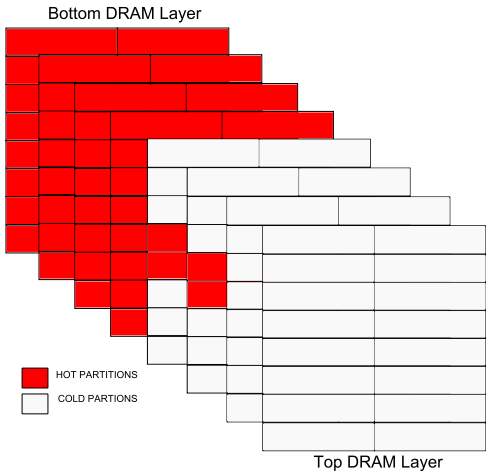

Fig. 3. Hot and Cold Partitions in DRAM layers

the HMC of $512MB$ in size, hence 4 $1Gb$ DRAM layers, and a logic layer. We can see that each layer is divided into 16 partitions. A vertical stack of partitions is called a vault. In this paper, the experiments involve a 1GB HMC.

B. Dynamic Thermal Management

In mobile computing environments, various design requirements limit the cooling facilities that can be provided to the platforms. Increase in memory intensive applications, along with the increasing memory speeds and capacities, can result DRAM chips to exceed the maximum operating temperature (85 °C). In order to entertain this problem, dynamic thermal management techniques are proposed, the basic idea of which, is to reduce memory traffic whenever temperatures are about to reach high temperatures. Bandwidth throttling [6] is one such technique, where there are four different levels of temperature ranges close to the temperature threshold, where the maximum bandwidth is reduced in each level by the memory controller. Lin et al. [7] improves on that by proposing two techniques - adaptive core gating and coordinated Dynamic Voltage and Frequency Scaling (DVFS), where he basically clock gates cores in each of the four different levels or reduces frequency and voltage of cores in each of the levels respectively. Iyer et al. [8] present two such techniques for memory thermal management for platforms based on Intel Centrino Duo Mobile Technology - namely Thermal Sensor on DIMM (TS-on-DIMM) and Delta Temperature in Serial Presence Detect (DT-in-SPD) [8]. TS-on-DIMM places an actual thermal sensor on the DRAM, while DT-in-SPD implements a prediction-based throttling technique, and in both cases, controller throttles down the traffic whenever it detects a violation over the previously set threshold temperature. DT-in-SPD starts throttling at a temperature of 70 °C, while TS-on-DIMM kicks in at 79 °C.

III. Data Compression for Thermal Mitigation

The HMC requires an on chip memory controller, whose function would be including and not limited to - address mapping, communication with the cache and also packetizing the memory requests to interface with the links to the off chip memory, very similar to the BOB Controller in Buffer-On-Board memory sytems [9]. The way data compression can be used for thermal mitigation in the HMC, is if the compression/decompression of the data is moved onto the on chip memory controller, minimizing power dissipated in the HMC as much as possible. The energy savings targetted in this case is the energy saved while reading/writing cache blocks from/to the HMC. Uncompressed data would require 8 bursts (assuming 1 burst = 64 bits of data), while compressed data would require fewer bursts, resulting in energy being saved. Hence, as less energy would be dissipated, the maximum temperature within the HMC would be decreased. One issue that might be argued about this technique, is that compression/decompression might add delay and reduce performance. But the simple compression technique used by Alameldeen [10] requires 5 CPU cycles to perform, which can be more than offset by the bus cycles saved while transferring the compressed block to the HMC.

A. Hot/Cold Partitions

As mentioned earlier, compression would take place on the on chip memory controller. But not all cache blocks would go through compression. Apart from reducing temperature which would reduce time HMC spends throttling, the target is also to reduce variation in temperature within the cube, as that would result in throttling of memory banks be more efficient. Adding on to that, in Figure 1, it was observed that the temperature was highest in the lower layers of the HMC (closer to the logic die) and also at the centre of each DRAM layer. So, it was decided that to best reduce variation in temperature, compressed cache blocks can be stored in only the hotter banks. Figure 3 demonstrates how each partition in each layer was labelled as hot or cold. In total 72 partitions are labelled to be hot. The on chip memory controller, attempts to compress/decompress cache blocks which are mapped only to the hot partitions. It should be noted that even though Figure 2 implies a 4×4 floorplan for the layers, the partitions in each DRAM die in the HMC are actually arranged in an 8×2 fashion (refer to Section IV-B).

B. Segmented Compression

In the baseline HMC simulator, it is assumed that 8 bursts are required to read/write a 64B cache block. But as we know, compression can result in variable sized blocks - some blocks

may even be bigger than 64B. Hence in this proposal, the on chip memory controller, when it sees a write, attempts to compress the block. And it dispatches the compressed block only if it provides any benefit. Otherwise, it does not compress the block, and the full 64B cache block is stored in the HMC. When the HMC receives the compressed block, the HMC would require fewer bursts, hence saving energy. The logic die must maintain a record of whether the cache blocks in question are compressed or not and how many bursts are needed to read that block. Only hot partitions would go through compression, and there are 72 hot partitions in total. Each partition is of 8MB in size - having 131072 partitions. So we have $(72 \times 131072 =) 9437184$ cache blocks that might contain compressed data. If this record would consist of 1 bit for every cache block, the record would be of size 1.125MB. So, in the experiments performed, two cases of this record are explored - 1 bit per cache block and 2 bits per cache block.

1) 1 bit per cache block: When 1 record bit per cache block is used, each cache block is compressed in 32B segments. Hence, if a cache block cannot be compressed to less than 32B, it is left uncompressed, and the record bit would be 0. If it can be compressed to less than 32B, trailing zeroes are added to fill the entire 32B segment, and the record bit would be 1. So if the record bit for a particular cache block would be 1, the logic die would read 4 bursts of data, while for 0, 8 bursts would be needed. For writing a cache block, if the data size is 32B (as sent from the on chip memory controller after compression), 4 bursts of write would be needed, as well as the record bit to be appropriately updated.

2) 2 bits per cache block: In case of 2 record bits per cache block, there are 4 different number of bursts of read/write to encode. Hence it was decided to compress the cache block in segment sizes of 16B (2,4,6 and 8 bursts). In case of read, the record bits are accessed to decide number of bursts, while for writes, the size of the data block dictates the number of bursts, requiring an eventual update of the record bits.

C. Compression Record

As seen in the previous section, atleast 1.125MB of compression record in the HMC side is needed. For the 2 bit per cache block option, 2.25MB worth of memory would be needed. A coarser granularity for ECC calculation [11] can be used to provide space for this required record. Usually, DRAM banks allocate 8 bits of ECC for every 64 bits of data. But if ECC is calculated at a granularity of 128 bits instead, ECC only requires 9 bits - resulting in 7 bits being saved. 7 bits for every 128 bits of data is more than enough for this proposal, which requires maximum 2 bits for every 512 bits of data. This method would be ideal to keep the compression record, as this has a zero overhead for compression record to be stored. This would result in reduction of bit-error coverage, but as mentioned by Gharachorloo et al. [12], this can be offset by the fact that newer DRAM devices have reduced soft-error rates, and also by techniques like periodic DRAM scrubbing.

In both the 1-bit-per-cache-block and 2-bits-per-cache-block cases, the logic die must never read/write less than 2 bursts, hence the logic die can proceed to perform the first two bursts, while in parallel accessing the record bits/determining number of bursts required based on data size. Hence access of this additional data structure should not have any detrimental effect on performance whatsoever, since the latency is hidden.

D. Compression Technique

The low power compression technique, mimicking the one proposed by Alameldeen [10], called Frequent Pattern Compression, is used in this paper. The 64B cache line is compressed on a word-by-word (32 bit) basis, potentially compressing a 32 bit word into a 3 bit *Prefix* that encodes the data pattern, and a variable sized *Data* portion. Table I shows the specific patterns and their corresponding Prefixes and Data sizes. As can be observed from the table, this FPC compression tries to take advantage of certain data types/values that may otherwise unnecessarily occupy the entire 32 bits of a word. The prefixes basically occupy the beginning of the cache block in order to expedite decompression. This compression is simple and hence consumes low power and also provides comparable compression ratios when compared to other more complex compression algorithms [10].

Prefix	Pattern Encoded	Data Size
000	Zero	0 bits (no data stored)
001	4-bit sign-extended	4 bits
010	One byte sign-extended	8 bits
011	halfword sign-extended	16 bits
100	halfword padded with a zero halfword	The nonzero halfword (16 bits)
101	Two halfwords, each a byte sign-extended	The two bytes (16 bits)
110	word consisting of repeated bytes	8 bits
111	Uncompressed word	Original Word (32 bits)

TABLE I: Frequent Pattern Compression [10]

IV. Simulation Environment

The time taken to overheat DRAM is in the order of tens to one hundred seconds. To study the effect of this data compression technique in the context of dynamic thermal management (DTM) schemes would require thousands of seconds of simulation, which is an infeasible simulation timelength. Hence this study involves a two-level simulation environment as shown in Figure 4.

The first level is a cycle-accurate architectural simulator of the Memory side of the system. For this, the Buffer-on-Board Memory System Simulator Suite (BOBSim) [9], configured for the HMC context, is used. BOBSim is fed with the workload (multiple memory reference traces), producing a power and required bandwidth trace of the DRAM structure. This trace is then fed into the second level. The second level is a thermal model simulator for the HMC. The thermal model simulator used is Hotspot [4], configured to simulate the HMC. As mentioned earlier, studying DTM schemes requires hundreds of seconds of simulation. Hence, there is a need to have two runs in this second level. In the first run, HotSpot is fed with the output trace from the first level, and the initial temperature throughout the cube is set at $60\,^{\circ}\mathrm{C}$. The first run produces a steady state temperature distribution within the HMC. This steady state temperature distribution is then fed into HotSpot again as the initial temperature distribution within the cube, for the second run, along with the power and bandwidth trace - thus simulating hundreds of seconds of simulation of the trace. The second run produces the transient temperature variation

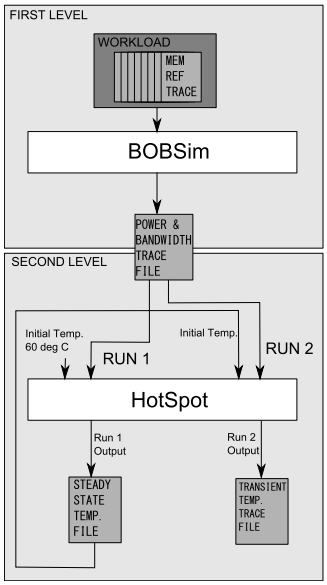

Fig. 4. Two Level Simulator

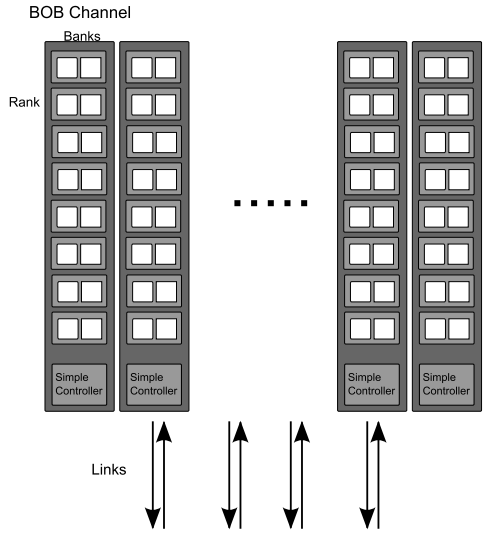

Fig. 5. HMC architecture as simulated using BOBSim

trace within the cube. Dynamic thermal management(DTM) of HMC is simulated in the second level of the two-level simulator, using the bandwidth trace, more details of which is in Section IV-B1.

A. BOBSim Modelling

The Buffer-on-Board Memory System Simulator Suite (BOBSim) [9] actually provides architectural simulation infrastructure of a memory system like IBM Power 795 memory system, Intel SMI/SMB memory system, AMD G3MX memory system, which are mostly used in server environments. In BOBSim there is a BOB Controller which resides on the CPU side, sending memory requests over fast links to the Simple Controllers residing on the memory side (off chip), which are then responsible for managing the specific commands for every request and sending the data back. So this is exactly analogous to the Hybrid Memory Cube, which is why this simulator was chosen for our experiments. But the parameters in BOBSim need to be adjusted, to configure it to simulate an HMC.

1) Architecture: In Figure 5, it can be seen how the overall architecture of BOBSim is configured to suit an HMC. As in Figure 2, it is observable that each cube has 16 vaults which can be accessed simultaneously. Each vault has 8 partitions (for a 1GB HMC), and each partition, has two arrays [1]. So the BOBSim model has 16 BOB Channels (vaults), each having 8 ranks (partitions). And each rank has two banks (arrays) as in Figure 5. The Simple Controllers represent the logic layer of the HMC. 4 32-bit Links(16-bit transmit/16-bit receive) [1] connect the HMC to the CPU die in BOBSim.

2) Energy and Timing values: CACTI-3DD [13] is a framework for architecture-level modeling of 3D die-stacked DRAM technology. It has options to simulate coarse grained rank level stacking as well as fine grained bank level stacking (which is similar to the Hybrid Memory Cube). Hence, CACTI-3DD [13] was used to model a 1GB 3D die stacked DRAM (fine grained) to get estimates for DRAM timing values and energy values, which in turn was fed into the BOBSim suite to complete the Hybrid Memory Cube architecture-level simulator. Table II shows the timing values and energy values that were fed into the BOBSim suite taken from CACTI-3DD. Pawlowski [14] mentions that the worst-case total power consumption by a $512MB$ HMC cube (ie. 4 DRAM Layers and 1 Logic layer) is $11.08W$. This power is not evenly distributed between the logic layer and the DRAM layers though. From [1] we realize that the DRAM layers dissipate energy at $3.7pJ/bit$ while the logic layer dissipate energy at $6.78pJ/bit$. From this ratio, and the fact that total power dissipation is $11.08W$, it was found that $7.168W$ is dissipated on the logic layer alone, while the rest is dissipated in the DRAM layers. Hence, the maximum power dissipation in a Simple Controller (SC) in the logic die is found to be $(7.168/16 =)0.448$W [1]. For the experiments performed in this paper, this maximum value is scaled based on the bandwidth used by the Simple Controller. The equation used is as follows:

$$Power_{SC} = 0.448 \times (1 - P_{static}) + 0.448 \times P_{static} \times \frac{BW_{used}}{BW_{max}}$$

P_{static} is the portion of the power assumed to be static. This value is assumed to be 0.2 for our experiments.

Timing Parameters	Value (ns)
tRCD (Row to column command delay)	4.36
tRAS (Row access strobe latency)	8.22
tRC (Row cycle)	13.29
tCAS (Column access strobe latency)	5.74
tRRD (Row activation to row activation delay)	1.07
Energy Parameters	**Value (nJ)**
Activation Energy	2.12678
Read Energy (8 bursts)	7.35017
Write Energy (8 bursts)	7.35106
Precharge Energy	2.21939

TABLE II: DRAM timing and energy values taken from CACTI-3DD

B. HotSpot Modelling

HotSpot provides the facilities to simulate 3D ICs. To successfully simulate HMC in HotSpot, the high level floorplan

of each of the layers of the HMC and the power dissipation trace in each functional block are required. Figure 2, which is adapted from [1], implies a 4×4 floorplan for the layers, but in [1] we also see a rectangular shaped partition. In that case a 4×4 floorplan would not make the HMC a "cube". Moreover, in [14] an image of the floorplan suggests an 8×2 arrangement. Based on these two evidence, it was concluded that the HMC has an 8×2 floorplan. The area of a layer is given to be $68mm^2$ - each side being $8.23mm$. For the floorplan, it was assumed that each partition is of dimensions $8.23mm \times 4.12mm$. This partition is arranged in an 8×2 manner. A more detailed floorplan of each partition is available in [1], but for our experiments, such details were deemed unnecessary. The power values of each of these partitions were obtained from the first level (BOBSim) of our two level simulator as discussed previously.

1) Dynamic Thermal Management: DTM schemes like TS-on-DIMM and DT-in-SPD [8] starts throttling at $70\,^{\circ}\mathrm{C}$ and $79\,^{\circ}\mathrm{C}$ respectively. In the experiments performed, two kinds of DTM schemes were used based on these two threshold temperatures. Similar to the DTM scheme in [7], these two schemes are also designed to have four different thermal emergency levels of temperature ranges near their respective thresholds where maximum allowed memory bandwidth is reduced each time a level is crossed. The Table III summarizes the temperature ranges of the two schemes used in this project along with their corresponding memory bandwidth restrictions. HotSpot was modified to simulate DTM and DRAM throttling. HotSpot is fed the trace file generated by BOBSim that contains the required bandwidth and power values for every epoch of 100000 CPU clock cycles. If the temperature falls within any of the thermal emergency levels, HotSpot enforces the corresponding bandwidth limit. The length of that epoch and the power values are scaled in proportion to the ratio of the required memory bandwidth and the maximum allowed memory bandwidth in that epoch. This provides a reasonable estimate of the effect of DTM schemes in total execution time of the workloads.

C. Workloads

SPEC2006 benchmarks were used in these experiments. SimPoint [15] was used to get the best sample of 1 billion instructions from each benchmark, and then Pin [16] was used to simulate the cache structure as in Section IV-D, and execute the simpoint to generate the private (Last Level Cache) LLC misses. As discussed in Section IV-D, only a portion of the address space of these benchmarks is allocated to memory in the HMC being simulated. Hence, from the generated memory trace, the 128MB address space with the most memory references are taken into consideration, and that is used as the input trace for the simulation of one core. The HMC Simulator (BOBSim) is fed with 8 of these traces to simulate 8 cores as in Figure 6. 20 workloads are used in these experiments, each workload consisting of 8 randomly chosen benchmarks from SPEC2006 suite. It should be noted that multiple instances of the same benchmark might exist in the same workload.

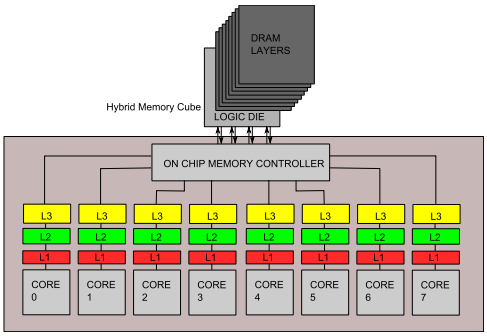

Fig. 6. Simulation Configuration

Workload #	SPEC2006 benchmarks
0	dealII cactusADM GemsFDTD gromacs leslie3d libquantum soplex leslie3d
1	zeusmp omnetpp soplex gcc h264ref gamess libquantum gobmk
2	hmmer tonto omnetpp omnetpp gcc tonto tonto hmmer
3	leslie3d wrf xalancbmk cactusADM zeusmp namd povray cactusADM
4	tonto povray bwaves gromacs milc wrf wrf bwaves
5	mcf h264ref povray gamess tonto libquantum gobmk bzip2
6	dealII gromacs perlbench bzip2 libquantum tonto omnetpp xalancbmk
7	hmmer bwaves perlbench h264ref hmmer omnetpp h264ref calculix
8	namd soplex tonto milc gromacs tonto soplex perlbench
9	tonto gromacs lbm GemsFDTD namd gobmk dealII GemsFDTD
10	perlbench libquantum milc hmmer cactusADM hmmer GemsFDTD xalancbmk
11	milc leslie3d namd dealII hmmer xalancbmk dealII h264ref
12	lbm gamess libquantum bwaves bzip2 leslie3d libquantum xalancbmk
13	povray xalancbmk hmmer wrf gobmk hmmer xalancbmk gamess
14	soplex h264ref xalancbmk povray calculix tonto tonto calculix
15	gobmk omnetpp leslie3d xalancbmk h264ref zeusmp hmmer h264ref
16	bwaves leslie3d mcf wrf gcc lbm omnetpp bwaves
17	cactusADM GemsFDTD hmmer gobmk gcc hmmer bwaves zeusmp
18	bzip2 mcf gobmk povray h264ref perlbench calculix libquantum
19	omnetpp soplex leslie3d leslie3d leslie3d GemsFDTD povray povray

TABLE IV: Workloads

D. System Configuration

An 8 core system is assumed in the experiments performed, with each core having private data cache(L1) and instruction cache(L1) (both of size 32KB), L2 and L3 caches (2MB and 16MB respectively), connected to the on chip memory controller, as can be seen in Figure 6. The on chip memory controller connects the cores to the Hybrid Memory Cube. The HMC is only 1 GB in size, so it is assumed that other memory modules exist in the system, but within the 1GB of the HMC that is taken into consideration, 128 MB is allocated to each core.

V. Experimental Results

In this section, the results of various experiments performed to evaluate the data compression technique is discussed. Overall, three sets of simulations are performed -

- *No DTM*: This is to evaluate the effectiveness of the data compression technique to reduce maximum temperature and temperature variation alone.
- *DTM Scheme 1*: This scheme has a threshold of $70\,^{\circ}\mathrm{C}$ as discussed in IV-B1. The effectiveness of the data compression technique to improve performance when DTM throttles DRAM is evaluated.
- *DTM Scheme 2*: This scheme has a threshold of $79\,^{\circ}\mathrm{C}$.

Emergency Level	L1	L2	L3	L4	L5
Scheme 1(°C)	(-,70)	[70.0, 71.0)	[71.0, 71.5)	[71.5, 72.0)	[72.0, -)
Scheme 2(°C)	(-,79)	[79.0, 80.0)	[80.0, 80.5)	[80.5, 81.0)	[81.0, -)
Memory BW	No Limit	19.2GB/s	12.8GB/s	6.4GB/s	off

TABLE III: Dynamic Thermal Management schemes used

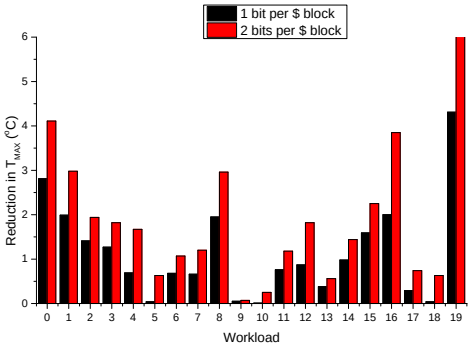

Fig. 7. Reduction in T_{MAX}

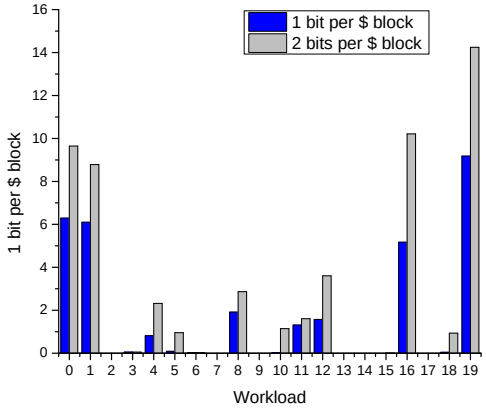

Fig. 9. DTM Scheme 1: % Speed Up

20 Workloads as presented in Section IV-C are used in these three sets of experiments.

For all the workloads, the *reduction* in maximum temperature within the cube for both 1 bit per cache line and 2 bits per cache line compression compared to baseline (no compression) are plotted. In the simulations involving dynamic thermal management (DTM), in case of both schemes, the portion of execution time spent in each of the emergency levels of throttling mode is presented (refer to IV-B1). Percentage speed up as compared to the baseline (no compression) is plotted as the performance metric.

1) No DTM: Figure 7 shows how T_{MAX} in HMC varied with all the workloads. It can be seen that there is a definite reduction in maximum temperature, which can result in minimizing the time spent in throttling mode when DTM is used, and also acts as a cushion to prevent DRAM from reaching harmful temperatures. A maximum temperature reduction of as much as around 6°C is observed. It can be seen that the Compression technique has a cooling effect on the HMC. Increasing the granularity of segmentation for compression, by having 2 record bits per cache block, cools the HMC even further.

The majority of the workloads undergo a reduced temperature owing to the effect of the proposed data compression technique. But, the data compression technique had negligible impact on some workloads, like workloads 9 and 10. This is because majority of the benchmarks in these workloads consist of benchmarks which have very low percentage of cache blocks that are compressible using this technique.

2) DTM Scheme 1: Figure 8 is a stacked column chart showing the percentage of execution that the DRAM spends throttling in different thermal emergency levels. It should be noted the column chart does not show the time spent in normal operation (ie. emergency level L1). Hence the heights of the columns are indicative of how much time the DRAM spends throttling (in all levels) as a percentage of total execution time.

It can be observed that in all cases, the data compression technique has an effect of reducing the total time DRAM spends throttling. As the number of record bits per cache block are increased to 2 bits, the total throttling time is always further decreased. A maximum of 20% of throttling time is decreased by this data compression technique. It should also be highlighted that in many cases, even though the total throttling time is reduced by data compression, the decrease does not seem that significant, as in workloads 0 and 12. If the breakdown of the throttling time is observed, it can be realised that with data compression, the throttling time in higher temperature emergency levels are decreased while throttling time in lower temperature emergency levels are increased. This is the result of reduced temperature in the HMC, due to data compression. This has an overall effect of increased total available memory bandwidth and hence lesser execution time.

Figure 9 displays the percentage speed up as a result of the data compression technique (both 1 and 2 record bits per cache block). The speed up follows a similar trend as in the case of reduction in maximum temperature, as workloads are altered, with a maximum speed up of as high as 14.2% and 9.2% for data compression with 2 and 1 record bit(s) per cache block respectively. The average speed up was 2.8% and 1.6% for data compression with 2 and 1 record bit(s) per cache block respectively. It should be highlighted, as seen in Figure 8, that in case of many of the workloads (2, 7, 9, 13, 14, 17), the HMC never even crosses the threshold temperature to undergo throttling in the first place. If we count these workloads out, the average speed up comes out to be 4% and 2.3% for data compression with 2 and 1 record bit(s) per cache block respectively. Hence we can observe that significant performance benefits are gained by using this technique, when HMC undergoes throttling.

3) DTM Scheme 2: Figure 10 represents the portions of time spent by the HMC in different thermal emergency levels.

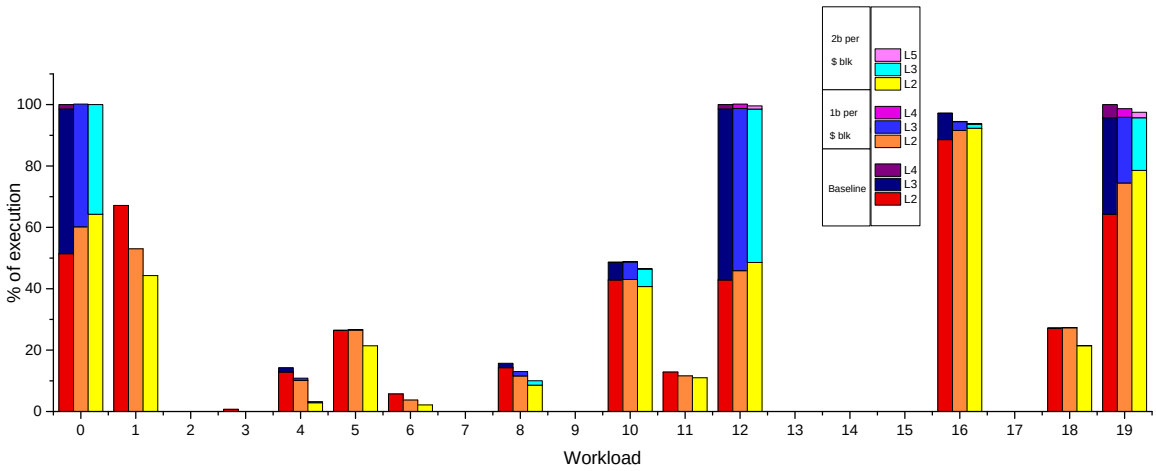

Fig. 8. DTM Scheme 1: % of execution spent throttling in various thermal emergency levels

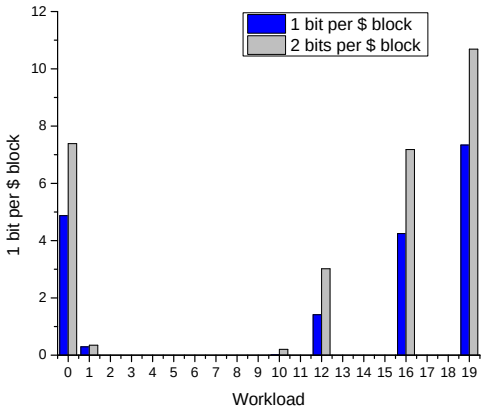

Fig. 11. DTM Scheme 2: % Speed Up

The effect of data compression can be seen to be same as in case of DTM Scheme 1, except for the fact that a lot less workloads stress the HMC enough to actually cause it to reach throttling mode. Figure 11 is the chart showing the speed up for each workloads when subject to data compression. A maximum speed up of 10.7% and 7.3%, with an average of 1.4% and 0.9% for data compression with 2 and 1 record bit(s) per cache block is observed respectively. With a threshold of 79°C, a lot less number of workloads cross into throttling modes, hence resulting in lower average execution time reduction. When averaged over workloads that reached throttling mode, average speed up is noted to be 3.6% and 2.3% for data compression using 2 and 1 record bit(s) per cache block respectively.

VI. Conclusion

This paper appreciates the need for thermal mitigation in the Hybrid Memory Cube, especially in the light of its cubic structure, and proposes using a simple low power data compression technique to reduce maximum temperature and temperature variation within the Hybrid Memory Cube. The data compression/decompression is done in the on chip memory controller, resulting in reduced power consumption in the Off Chip Hybrid Memory Cube as lesser bursts are required to read/write compressed cache blocks. This proposal is evaluated using a two-level simulator consisting of architectural simulation using BOBSim and thermal model simulation using HotSpot, in the context of using two dynamic thermal management (DTM) Schemes. Maximum temperature was reduced by as much as 6 °C. Due to reduced time DRAM spends throttling because of this reduced temperature, an average of 2.8% and a maximum of 14.2% speed up is gained. It can be observed that data compression on the processor side, can cause reduced temperature as well as increased performance in the Hybrid Memory Cube.

References

[1] J. Jeddeloh and B. Keeth, "Hybrid memory cube new dram architecture increases density and performance," in *VLSI Technology (VLSIT), 2012 Symposium on*, june 2012, pp. 87 –88.

[2] A. Fawibe, J. Sherman, K. Kavi, M. Ignatowski, and D. Mayhew, "New memory organizations for 3d dram and pcms," in *Proceedings of the 25th international conference on Architecture of Computing Systems*, ser. ARCS'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 200–211. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28293-5_17

[3] G. Loi, B. Agrawal, N. Srivastava, S.-C. Lin, T. Sherwood, and K. Banerjee, "A thermally-aware performance analysis of vertically integrated (3-d) processor-memory hierarchy," in *Design Automation Conference, 2006 43rd ACM/IEEE*, 0-0 2006, pp. 991 –996.

[4] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam, "Compact thermal modeling for temperature-aware design," in *Proceedings of the 41st annual Design Automation Conference*, ser. DAC '04. New York, NY, USA: ACM, 2004, pp. 878–883. [Online]. Available: http://doi.acm.org/10.1145/996566.996800

[5] S. Liu, B. Leung, A. Neckar, S. O. Memik, G. Memik, and N. Hardavellas, "Hardware/software techniques for dram thermal management," in *17th International Conference on High-Performance Computer Architecture (HPCA-17 2011), February 12-16 2011, San Antonio, Texas, USA*. IEEE Computer Society, 2011, pp. 515–525.

[6] K. Man, "Bensley fb-dimm performance/thermal management," in *Intel Developer Forum*, 2006.

[7] J. Lin, H. Zheng, Z. Zhu, H. David, and Z. Zhang, "Thermal modeling and management of dram memory systems," in *Proceedings of the 34th annual international symposium on Computer architecture*, ser. ISCA '07. New York, NY, USA: ACM, 2007, pp. 312–322. [Online]. Available: http://doi.acm.org/10.1145/1250662.1250701

[8] J. Iyer, C. L. Hall, J. Shi, and Y. Huang, "System memory power and thermal management in platforms built on Intel Centrino Duo mobile technology," vol. 10, no. 2, pp. 123–132, May 2006. [Online]. Available: http://developer.intel.com/technology/itj/2006/volume10issue02/art04_Memory_Power_Management/p01_abstract.htm

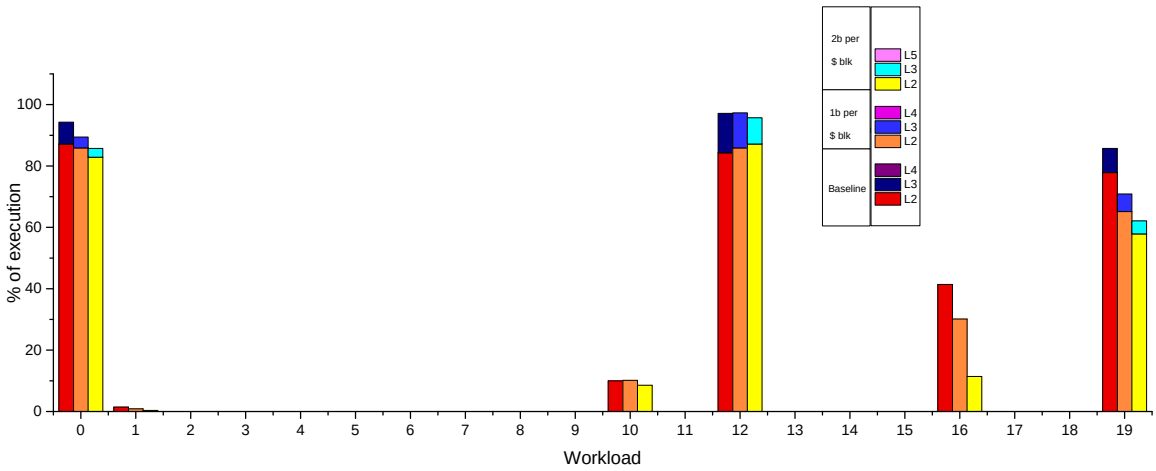

Fig. 10. DTM Scheme 2: % of execution spent throttling in various thermal emergency levels

[9] E. Cooper-Balis, P. Rosenfeld, and B. Jacob, "Buffer-on-board memory systems," in *Computer Architecture (ISCA), 2012 39th Annual International Symposium on*, 2012, pp. 392–403.

[10] A. R. Alameldeen, "Using compression to improve chip multiprocessor performance," Ph.D. dissertation, University of Wisconsin-Madison, Madison,WI,USA, 2006.

[11] A. Nowatzyk, G. Aybay, M. Browne, E. Kelly, D. Lee, and M. Parkin, "The s3.mp scalable shared memory multiprocessor," in *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, vol. 1, 1994, pp. 144–153.

[12] K. Gharachorloo, L. A. Barroso, and A. Nowatzyk, "Efficient ecc-based directory implementations for scalable multiprocessors," 2000.

[13] K. Chen, S. Li, N. Muralimanohar, J.-H. Ahn, J. Brockman, and N. Jouppi, "Cacti-3dd: Architecture-level modeling for 3d die-stacked dram main memory," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, 2012, pp. 33–38.

[14] J. Pawlowski, "Hybrid memory cube (hmc)," in *Proceedings of Hot Chips*, vol. 23, 2011.

[15] G. Hamerly, E. Perelman, J. Lau, and B. Calder, "Simpoint 3.0: Faster and more flexible program analysis," in *Journal of Instruction Level Parallelism*, 2005.

[16] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: building customized program analysis tools with dynamic instrumentation," in *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*, ser. PLDI '05. New York, NY, USA: ACM, 2005, pp. 190–200. [Online]. Available: http://doi.acm.org/10.1145/1065010.1065034