Near Data Processing: Impact and Optimization of 3D Memory System Architecture on the Uncore

Syed Minhaj Hassan, Sudhakar Yalamanchili & Saibal Mukhopadhyay
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, Georgia
minhaj@gatech.edu, sudha@gatech.edu & saibal@ece.gatech.edu

ABSTRACT

A promising recent development that can provide continued scaling of performance is the ability to stack multiple DRAM layers on a multi-core processor die. This paper analyzes the interaction between the interconnection network and the memory hierarchy in such systems, and its impact on system performance. We explore the design considerations of a 3D system with DRAM-on-processor stacking and note that full advantages of 3D can only be achieved by configuring the memory with high number of channels. This significantly increases memory level parallelism which results in decreasing the traffic per DRAM bank, reducing their queuing delays, but increasing it on the interconnection network, making remote accesses expensive. To reduce the latency and traffic on the network, we propose restructuring the memory hierarchy to a memory-side cache organization and also explore the effects of various address translations and OS page allocation strategies. Our results indicate that a carefully designed 3D memory system can already improve performance by 25-35% without looking towards new sophisticated techniques.

Categories and Subject Descriptors

B.3.1 [**Hardware-Dynamic memory**]: Computer systems organization Multicore architectures

Keywords

3D memory system, Near data computing, Interconnection network, Address mapping, HMC

1. INTRODUCTION

3D packaging has emerged as a vehicle for scaling system densities and performance due to i) increased inter-tier bandwidth, ii) reduced inter-tier latencies, and iii) ability to integrate dies from different process technologies as a means of customization and hence performance improvement. Moving forward, continued scaling of Through Silicon Vias (TSVs) is increasing the inter-die bandwidth and reducing inter-die latency. Large numbers of fine grained TSVs across a 2D cross section can support a large number of memory channels. Hence, the memory bandwidth can be increased significantly. e.g. a Micron die stacked DRAM increases bandwidth up to 128GB/s as opposed to 21.34GB/s and 10.66GB/s of DDR4-2667 and DDR3-1333 respectively [17]. It is important to understand features of architectural organizations that can make use of this technology capacity.

This paper addresses the problem of improving execution performance in 3D chip multiprocessors with DRAM-on-processor stacking. The specific problem of interest is the role of the interconnection network in limiting the memory bandwidth utilization. We seek to understand how best to organize the memory hierarchy so as to maximize the bandwidth and latency advantages of 3D technology. The work addresses two key issues, that is, 1) the extreme parallelism of a 3D memory system makes address translations a key determinant of locality and parallelism, which can be used to reshape the memory traffic maximizing performance, and 2) reduced 3D DRAM delay leads to refactoring of the memory latency path, which increases pressure on the interconnection network between the memory and the cache hierarchy and requires architectural modifications that either reduce the network latency and traffic or adapt to this re-factored memory-latency path.

We first discuss the importance of increasing the number of memory channels in a 3D system and show that 2D network latency becomes a more critical problem in such a system. We then propose a memory-side cache organization in which distributed L2 banks are placed next to the DRAM channels which reduces the traffic in the network, simultaneously distributing it to various memory channels. With coordinated address space mappings across the caches and DRAM channels, this organization increases TSV utilization and eliminates unnecessary traffic on the network, with net improvements in performance. We also evaluate refinements of the address space mapping to distribute requests across different DRAM channels. Our load distribution mechanisms significantly reduces queuing delays in the MC queues, thus resulting in reduced round trip memory latency and improved performance. Lastly, we explore the impact of OS page allocation in keeping the network traffic minimal. Our combined approach indicates that a carefully designed 3D memory system can already improve performance by 25-35% without the overhead of any sophisticated technique.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

This paper makes the following contributions.

1. Make a case for high number of narrow memory channels in order to improve performance of a 3D system.
2. Deconstruct a memory-access latency path into network and DRAM latency and evaluate their relative importance for 3D bandwidth utilization.
3. Observe and exploit the potential of various levels of address translations in such a massively parallel system.
4. Propose a memory-side cache organization that when coupled with appropriate address translation mechanisms significantly reduces the negative impact of 2D intra-die network and improves utilization of the 3D TSVs and consequently memory bandwidth and IPC.

The remainder of the paper is organized as follows. Section 2 provides a brief background for 2D vs. 3D systems, describes it challenges and trends, and summarizes the experiments performed. Section 3 determines the number of channels used in the baseline system while section 4 analyzes the impact of bandwidth and parallelism in the 3D system. Section 5 proposes optimizations with various address translations mechanisms and memory system organizations. Finally, the last two sections describe the results and future work, respectively.

2. BACKGROUND

One of the most pressing challenges in the chip industry, the lack of pin bandwidth, can be resolved by die-stacking technology which allows stacking two or more dies, even of different technologies, to be stacked into a 3D package. In this section, we will briefly describe the advantages and challenges of die stacking specially in the context of processor-to-memory stacking.

2.1 2D vs. 2.5D vs. 3D Stacked Memory

A typical integrated circuit (IC) with a ball grid array (BGA) package consists of a single die placed on top of a package substrate as shown in Fig 1 a). The connections between the die and the substrate, called bumps or more recently micro-bumps, has a pitch around 30-50um while the connections between the package substrate and the printed circuit board (PCB), ball interconnect, is around 400-600um. Although the pitch of micro-bumps can be reduced further, PCB designers are facing numerous challenges lowering the ball pitch down resulting in a very slow increase in the pin count of modern chips. Furthermore, these ball interconnects are connected to long wires in the pcb with non-linear coupling and impedance mismatch effects that keep the speed of these wires low. The combined problem has hindered the rapid increase of pin bandwidth, mainly affecting off-chip memory bandwidth and power delivery mechanisms in modern chip multi-processors (CMPs).

A promising solution to this problem is to introduce another silicon layer, a silicon interposer, between the package substrate and the die, and have multiple dies placed together on a single substrate as shown in Fig 1 b). The interconnections between the two dies will have a very small pitch and much lesser distance, increasing both the number of connections and their speed. The speed and thus bandwidth is further increased by the fact that the connection between the two dies passes through a much faster silicon layer than through the slow metallic wires of a PCB. Lesser distance also means smaller wires with lower electrical loads and thus smaller drivers to drive these wires, all of which decreases the overall power dissipation. Another advantage of wires in the faster silicon layer is their higher signal integrity, leading to removal of sophisticated signal preserving techniques like delay locked loops, on-die termination etc., again improving power efficiency. However, the area and thus cost of the silicon interposer and the package substrate is high.

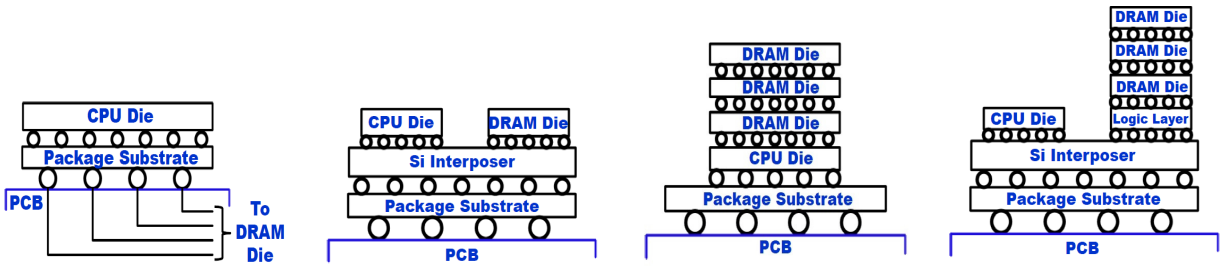
Another, even better solution to the problem is to stack multiple dies together and make a true 3D package as shown in Fig 1 c). Stacking is enabled by etching holes called through-silicon vias (TSVs) in the bulk silicon portion of the die [3] [31]. TSVs have the advantage that two dies of completely different technologies can be stacked and later bonded using various bonding processes [11]. 3D configuration has several important advantages, such as, reduced distance between the dies, low electrical load on the TSVs, an increase in the number of interfaces between the dies, and a reduction in overall chip area and thus cost. A hybrid of the three technologies can also be used as shown in Figure 1 d). The hybrid configuration has only similar structures like multiple DRAM dies stacked in a 3D configuration with a lower probability of defects due to technology mismatch thus achieving significantly higher yield.

Any organization of various components of a system can be stacked. The most straight forward choice is to stack multiple DRAM dies and place them next to a processor die as a 2.5D structure [28] or put it directly on top of the processor die as a 'true 3D structure' [10]. In this work, we will analyze the later case [9] of true 3D stacking, which will simply be referred to here as a 3D system.

2.2 Challenges and Trends

3D processor memory systems have been gaining significant attention over the last decade but their rapid adoption is slow mainly because of the following four challenges. 1) TSV packaging, 2) thermal management, 3) yield concerns, and 4) DRAM capacity. However, recent advancements in packaging [29], microfluidics [26], and silicon-on-interposer [19] technologies indicate rapid progress towards solving these challenges leading to the proposal of many interesting new processor-memory architectures such as PIM [30], PoM [18], and NDC [13]. Increasing the capacity of DRAM, however, is still a problem specially with small number of dies allowed to stack to meet the thermal constraints. Increasing the chip size is also not desirable due to lower yield with bigger chips. This has led to the designers using stacked DRAM as either a large last level cache (LLC) [14] or part of a multi-level main memory [4]. Consequently, the focus has been on the management of DRAM caches or the OS- or hardware-based page swapping between the fast and the slow memory. However, 4GB of stacked DRAM is coming soon [25]. This opens the door to a wider range of management strategies and goals.

In this paper, we focus on the impact of high bandwidth and proximity of the memory on the performance of the whole memory hierarchy, that is, the network, the cache, and the main memory itself. We assume that programs fit in the stacked DRAM and paging events being relatively infrequent for the purposes of performance analysis.



(a) 2D (b) SI Design (c) 3D (d) 2.5D

Figure 1: Various packaging options

2.3 Summarizing the Experiments and Motivation Behind Them

A 3D system provides a large number of connections between the DRAM and the processor layer which can be used to increase the DRAM bandwidth. However, the new challenge is to efficiently manage this bandwidth, that is, to understand how to best utilize these connections to improve performance and reduce power. To achieve this goal, this paper evaluates and re-organizes the memory hierarchy incorporating state-of-the-art techniques in the 3D system.

First, we determine the number of memory channels in a 3D system. We show that having large number of narrow channels is better as compared to a few wide channels. We settled with 16 channels for our baseline system. The baseline system consists of a distributed banked L2 cache as the last-level shared memory with directory-based coherence protocol in which each bank acts as a home directory for part of the global address space. We analyze this system consisting of a large number of channels and LLC banks and point out the high parallelism available in the DRAM, efficient management of which can regulate the power and performance of the overall system. We identify that this massive parallelism reduces the load on individual DRAM banks and channels, decreasing their queuing delay, making the memory-access latency small, and putting more pressure on the interconnection network connecting these channels. This results in the network latency becoming comparable to that of DRAM latency, an observation that does not hold true in conventional 2D systems. Furthermore, we identify that address translation mechanisms at various levels of the memory hierarchy (LLC or DRAM) can severely impact the management of this increased parallelism and used it to regulate locality vs parallelism tradeoffs in the system. We further show that a careful co-ordination between addressing schemes at various levels is beneficial to the overall system performance. Lastly, we also touch upon the importance of traffic distribution to all the channels in order to utilize maximum bandwidth without violating the conventional principles of spatial locality and DRAM hit rate; and use address translation schemes to provide this distribution.

With these observations in mind, we reorganize the memory hierarchy into a banked memory-side cache organization that reduces the network traffic latency. The re-organization is made possible by the co-ordination of address mapping at the LLC and the MC level. To reduce the network traffic further, we implemented locality based OS page allocation strategies that tries to keep the data close to the requesting cores as much as possible and analyzes their impact on the overall system performance.

3. DETERMINING THE NUMBER OF MEMORY CHANNELS

A basic consideration in arranging a 3D memory system is to determine the number of memory channels in the system, e.g., if we assume 1024 data I/Os, should they be arranged as one 1024-bit wide channel or sixteen 32-bit wide memory channels distributed evenly across the die. Note that the bandwidth of the overall system remains the same in both the cases. The following two subsections briefly explain why having multiple channels is a better choice for future memory systems.

3.1 Larger BL and Smaller Transaction Wastes Memory Bandwidth - All Hits Case

The internal speed of DRAM technology is not scaling fast enough, therefore keeping the fundamental latencies of activate, precharge and CAS operations constant, around 13-18ns across various generations. Similarly, the row cycle time still hovers, around 50-60ns. The DDR bus speeds, however, maintained an upward trend from early 400MHz DDR2, to 2133MHz, and even 3200MHz LPDDR4. Such bus scaling has been maintained by over-fetching the data internally, and then delivering it to the DDR bus in a burst mode, which has increased the minimum burst length (BL) parameter from two-to-four in early DRAMs, to eight-to-sixteen in modern DDRs. Assuming fixed bus widths, larger burst lengths result in higher minimum transaction size (e.g., the minimum transaction size of x32 LPDDR4 with burst length of 16 is 64B). On the other hand, cache line sizes are not increasing at a fast rate, and with the advent of on-chip accelerators, the size of transactions being requested from the memory is decreasing as well. As a result, part of the fetched data will be wasted, e.g., for a 32-byte transaction with the above mentioned LPDDR4 technology (min tx. size of 64B), half of the memory bandwidth will be lost even with all page hits. In such a case, reducing the bus width to x16 (min tx. size of 32B) will remove the bottleneck, allowing the memory to have two separate channels that can operate independently. It should also be noted that the problem will worsen in the case of writes, in which masking the other half of the bytes will require reading the data first. We would also like to note here that this is not a very uncommon case and streams with stride greater than 32 will face this issue.

3.2 Increased Frequency Reduces DRAM Efficiency - All Misses Case

As mentioned earlier, the DRAM bus frequency has been increasing rapidly with modern DDRs. A problem with high frequency is that a constant row cycle time (t_{ras}) of 50-

60ns translates into a larger number of cycles. However, the number of cycles in which the bus is actually transferring data (the data cycles) for a given transaction size remain constant. Thus, in the case of consecutive page misses, in which the next transaction can be sent after a minimum of t_{ras} cycles, only a few of the bus cycles will be transferring data. For example, in the case of x1024 bus with $t_{ras} = 60ns@2133MHz$, every subsequent miss transaction can be sent only after 128 cycles, and the DRAM transfer rate is 128B per data cycle. Even for a transaction size of 128B, all cycles except the first cycle will be wasted. If the burst length is greater than one, the DRAM will fetch redundant data which will be discarded in the memory controller (MC). Hence, the DRAM in the all page miss case has a maximum efficiency of $1/128 < 1\%$ with a parallelism of only one. With an x32 bus (32 data cycles for 128B tx.) and 32 parallel channels, the DRAM efficiency will jump to 32/128=25% without wasting any data burst cycles, and allows for a maximum parallelism of 32.

It should be noted that reducing the bus width increases the latency in the bus (one data cycle vs. 32 data cycles), which may decrease the system performance. However, the time on the bus is a relatively small portion of the overall DRAM latency, e.g., 32 bus cycles @2133MHz = 15ns of the 100 or more nano seconds taken by the memory in general. The rest of the time is consumed by activate, precharge, and CAS operations, along with the queuing delays. Hence, increasing the cycles on the bus impacts the overall system performance only by a small amount, even with a large number of DRAM banks sharing the bus. Again, this case can also occur frequently with interference among multiple threads and processes. Not only that, all other cases, which occur between these two extremes, will face similar concerns.

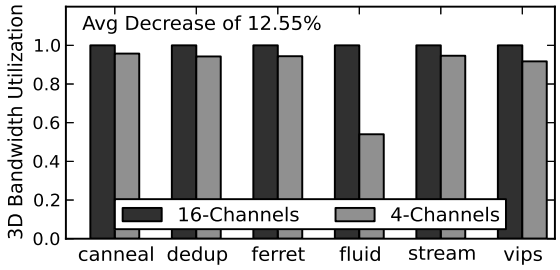


Figure 2: 3D bandwidth utilization with various number of channels

We performed simulations with four and sixteen channels having bus width of 256 and 64 bits, respectively. We increased the number of ranks by 4x in the 4-channel configuration to reduce the impact of the queuing delay. The rest of the system is configured similar to the baseline model discussed in section 6.1. Fig 2 shows the 3D bandwidth utilization of the two configurations. It can be seen that all benchmarks improve their performance by increasing the number of channels. Overall, reducing the number of channels decreases 3D bandwidth utilization on average by 12.5%.

3.3 Cost of a Channel

The previous analysis shows that increasing the number of channels is extremely beneficial for performance reasons. However, a problem with increasing the number of channels is the reduction in DRAM density. Conventionally, DRAM is considered a commodity device which maximizes its density to reduce cost. However, increasing the number of channels and banks (aka parallelism) reduces the size of internal DRAM arrays and sub-arrays, increasing the area taken by decoding, sensing and other peripheral logic attached to it. All channels operate independently with all the peripheral circuitry, including the memory controller schedulers, replicated with each channel, increasing its cost. We fix our design with 16 channels, a number inspired by hybrid-memory cube (HMC's) internal DRAM structure [12].

3.4 Current Standards

In this section, we briefly point out how modern DRAM standards are coping up with the above mentioned challenges. LPDDR4 [23] supports both very high speed and a burst length of 16. However, it keeps the bus width small (x16, x32) increasing the number of channels. DDR4 [21], which again has very high speed, introduces the concept of bank groups [1]. The idea is to keep the burst length of eight (4 DRAM cycles) but does not allow another read/write operation to the same bank group for eight DRAM cycles. The other bank groups, on the other hand, can accept read/write command after only four cycles, as is the case with conventional DDRs. (This is achieved by having two separate t_{CCD}. t_{CCD_L} for same bank group and t_{CCD_S} for different bank groups). The problem with this approach, however, is that consecutive hits to the same bank group can only operate at 50% bus utilization. Wide I/O [20] has low speed and thus very large bus width with only one-to-two channels supported. However, Wide I/O 2 [24] has increased the number of channels to four-to-eight allowing the bus width to reduce (with larger prefetch) and increasing the frequency. At this rate, it is very likely that Wide I/On will support even more channels. HBM [22] again supports eight channels and HMC's internal structure already has 16 channels, with each die partitioned into 16 sub-dies (one for each channel) similar to our baseline design.

4. ANALYZING THE BASELINE SYSTEM

The previous section suggests a large number of channels for a baseline 3D system increasing its parallelism. In this section, we first briefly describe our baseline 3D model and analyze its performance identifying the bottlenecks. We show that because of high parallelism and reduced MC queuing delays, the network latency becomes a bigger problem in a 3D system. Later, we highlight the impact of address mapping at various levels of the memory hierarchy that efficiently utilizes the parallelism available.

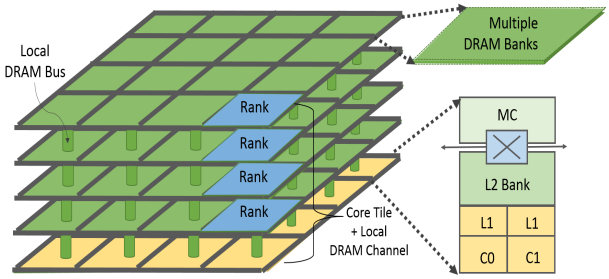


Figure 3: Full system model

4.1 The Baseline System - Overview

Our baseline system consists of a 3D processor-memory architecture, in which multiple DRAM dies are stacked on top of a processor die as shown in Figure 3. The processor die consists of 4x4 tiles, each consisting of two cores + two private L1 caches, a bank of globally shared L2 (divided into 16 banks, one per tile), a router connecting the tiles, and an MC controlling the DRAM dies/layers above. Similar to the HMC organization [12], each of the DRAM layers is divided into 4x4 blocks. One block from each layer combines with the respective block from each of the other layers, creating a DRAM vault. Each vault acts as a multi-layered DRAM with its own memory controller. Hence, there are 16 vaults controlled by 16 MCs in a four-to-eight layered cube. Thus, our baseline 3D system consists of 16 DRAM channels (each vault act as a separate DRAM channel), each consisting of four-to-eight ranks (each sub-layer in a vault can be treated as a rank since all share a common bus), and two banks (each sub-layer is divided into two banks that operates in parallel as is the case with conventional DRAMs).

Next, we discuss the impact of such a system on the overall memory-acces latency path. All simulations in this section uses the baseline 3D system (unless otherwise stated), more details of which can be found in section 6.1.

4.2 Reduced MC Queuing Delay

As discussed earlier, the fundamental RAS/CAS latencies in conventional DRAMs remained roughly the same over the years, since the time to charge and discharge the DRAM capacitors did not change significantly across technology generations, a trend that is persisting with 3D as well. Only, a very small reduction in RAS/CAS latencies will be seen because of the sharing of some peripheral logic. On the other hand, bandwidth and parallelism is increasing, which leads to reduced memory traffic per channel, and thus reduced queuing delays in the MC. Since the queuing delay is a major component of the DRAM latency, the overall round-trip time decreases. Figure 4 plots the latency of reads in DRAM with a 4-channel 2D and a 16-channel 3D system with the same number of cores and caches. Note that the RAS/CAS latencies in both the systems are kept the same. On average, the DRAM latency of the 2D system is 2.1 times higher than that of the 3D system. The main contributors of this increase in latency is the increased memory interference in the MCs (a result of higher memory traffic) and an increased queuing delay.

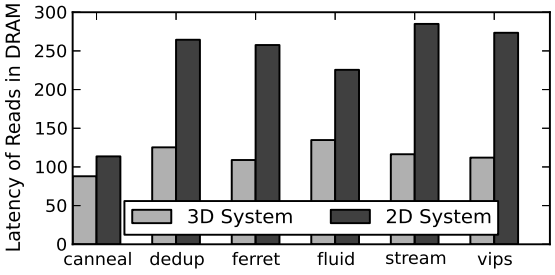


Figure 4: 2D vs. 3D system DRAM latency (cycles)

4.3 Comparable Network Latency

Due to reduced queuing delays, network latency in a 3D system becomes comparable to that of DRAM latency. This behavior is shown in Figure 5, which plots the latency distribution of DRAM bound read requests for various PARSEC applications. Note that in the figure, $L1 - L2$ and $L2 - DRAM$ correspond to the latency in the network, and DRAM-only corresponds to the latency in the DRAM. The impact of DRAM latency on the overall system is further reduced by the fact that the ratio of the memory traffic that reaches the DRAM is generally around a quarter to one third of the total number of requests that travel the network (e.g., 25% for vips). The remaining misses are satisfied by remote caches and related coherency traffic, which only travel in the intra-die links.

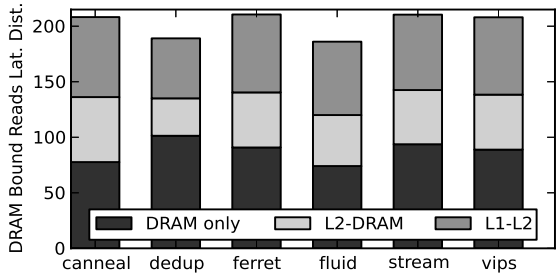


Figure 5: Latency of requests from DRAM (cycles)

To further explore this point, we use a parametric model in which various DRAM timing parameters are normalized relative to the CAS latency. The model is used to illustrate the behavior of DRAM bound reads as a function of CAS latency as shown in Figure 6 (left). The impact on the global IPC is illustrated in Figure 7. From the two figures, it can be concluded that although increasing the latency of DRAM accesses (CAS) increases the overall latency of DRAM-bound read requests, the average latency of all L1 misses (Fig 6 right), and hence the IPC, does not decrease at the same rate. On the other hand, the latency of the 2D network affects all requests (memory bound and coherence included), and therefore contributes more towards the overall latency penalty.

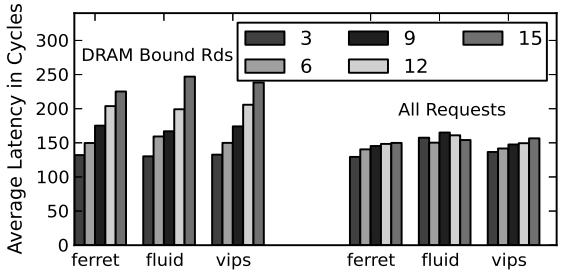


Figure 6: DRAM-bound read latency vs. CAS latency. CAS in DRAM cycles – Request in CPU cycles

4.4 Impact of Address Translations

Our baseline system has multiple L2 banks distributed across the tiles with directory-based coherence protocol. In such a system, part of the address space is assigned to each L2 bank, and the corresponding bank is treated as a home directory for the assigned addresses [7] [15]. Similarly, in systems with multiple DRAM channels, the address space is distributed among the channels, or in this case, among the DRAM vaults (Figure 8). The address translation mecha-

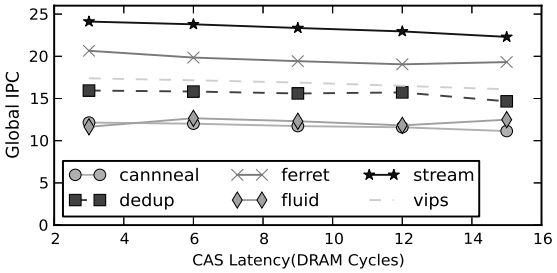


Figure 7: CAS latency vs. IPC

nisms that determine the home L2 bank and the corresponding DRAM vault for a given physical address (Figure 9) dictate the amount of parallelism and the number of hops in the network. For DRAMs, the interleaving granularity of address mapping functions determines whether the memory traffic is distributed among all memory channels at a particular instant of time (low-order interleaving), or localized to one channel (high-order interleaving); see Figure 9. For L2 banks, the address assignment policy is governed by the principle of locality, such as the commonly used first touch policy that keeps the data local as much as possible. The design goals at both levels, however, are conflicting, and thus the interaction of these address translations brings interesting behaviors that has not been discussed in the past, which becomes even more important in the case of 3D systems where we have more channels and more L2 banks.

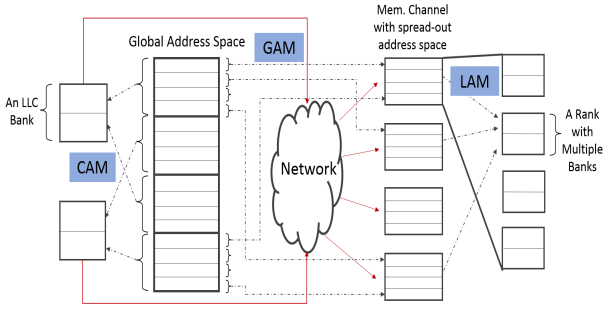


Figure 8: Global address space distribution in cache banks and DRAM channels/banks

The most straight forward translation known as high order interleaving (HOI) maps large blocks of continuous physical addresses to same L2 bank or DRAM vault. On the other hand, Low order interleaving (LOI) distributes cache line size blocks to different banks while page interleaving PgI distributes the address space at the granularity of OS or DRAM page sized blocks. Since all interleavings can happen at both the L2 bank and the MC level, we need to distinguish between whether one is applied at the L2 bank or the MC level. We termed the address translation mechanism at the L2 bank level as CAM (cache address mapping), and the address translations at the MC level as GAM (global address mapping), respectively. Address translations within a DRAM that decides a particular rank and bank are called LAM (local address mapping); see Figure 8. Hassan and Yalamanchili [6] provides more details of various address mapping functions.

4.4.1 *CAM and GAM under Multi-level Cache Hierarchy*

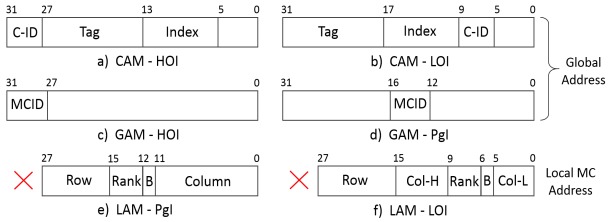


Figure 9: Various address space mappings with CAM, GAM, and LAM

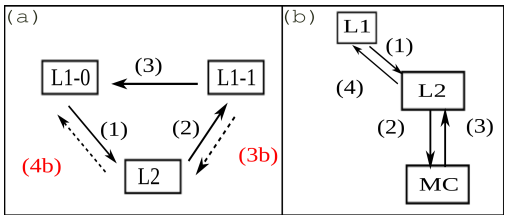


Figure 10: Messages generated to fetch from a) a remote cache and b) the DRAM

In a system with private L1 caches, and a distributed banked L2 cache, an L1 cache miss travels through multiple hops in the form of various data- and coherence-related messages in order to get serviced. The important messages among them are indicated in Figure 10. When an L1 encounters a miss, the home L2 bank is consulted for the updated copy (shown in Figure 10 (msg. 1, fig. a)), which can be local or remote based on the address. If the L2 bank experiences a miss, it sends the message to either another L1 that holds the updated copy (msg. 2, fig. a), or to the corresponding MC (msg. 2, fig. b). The MC generally sends the reply back via the L2 (msg. 2/4, fig. b), while the L1 either returns the copy directly to the requesting L1 (msg. 3, fig. a), or sends it to the home L2 first, which in turn sends it back to the L1 (msg. 3b/4b, fig. a). We use the former case of L1-L1 traffic, in which the L2 also requires an acknowledgment to update its state machine (not shown in the figure). The same thing also happens with other requests like invalidations etc., that do not transfer data, but update the cache state machines.

The above discussion indicates that each L1 miss results into multiple messages that travel along the network in both horizontal and vertical directions. The hop count of these messages can be reduced by careful co-ordination of CAM and GAM schemes. The result is a modification in the organization of the memory hierarchy explained in the next section.

5. AN IMPROVED 3D MEMORY SYSTEM

This section builds on the analysis done in the previous section and proposes a re-organization of the memory system that reduces the network traffic, made possible by a careful co-ordination of CAM and GAM. We further describe the importance of traffic distribution among different memory channels that maximizes parallelism while trying to keep it to the local banks or MCs to reduce the traffic in the network.

5.1 Same Address Mapping and Memory Side Cache

In this section, we propose the use of the same mapping

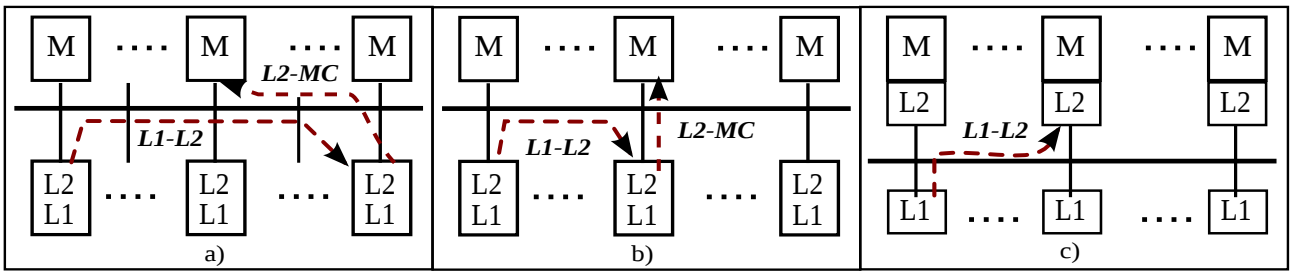


Figure 11: a) The baseline system, b) same CAM and GAM organization, and c) mem-side cache organization

function for both CAM and GAM, which is possible only in the case of 3D systems that has comparable number of L2 banks and memory channels. We show the usefulness of the policy by noting the fact that the use of same function for CAM and GAM keeps the L2-to-MC traffic local, that is, only in the vertical direction. The L1 requests have to already travel horizontally in the 2D network (by 2D network we mean links in the horizontal direction) in order to reach the corresponding L2 bank.

The proposed scheme is described in Figure 11. Figure 11 a) represents a 3D system which consists of different functions for CAM and GAM. An L1 miss first travels to its home L2 bank, which is located on a remote tile to determine the current state of the cache line. If the line is not present in the L2 or any of the remote L1s, it forwards the request to the MC, which again is located in a remote DRAM vault, incurring an additional latency in the network for the L2-MC traffic that could have been avoided by using the same function for both CAM and GAM. Figure 11 b) represents the scenario, in which the L2-MC traffic remains local. Furthermore, since there is no remote traffic between the L2 bank and the MC, providing a direct connection between them (as shown in Figure 11 c)) will remove the serialization latency of an L2 miss destined to the corresponding MC through the router (i.e., a miss has to be converted back and forth from/to flits while traveling through the router, and a direct link will remove this breakdown). It also reduces the load on the corresponding router and the NIs. As shown in Figure 5, this latency is 20-25% of the overall memory latency of the DRAM bound reads, and will be removed altogether.

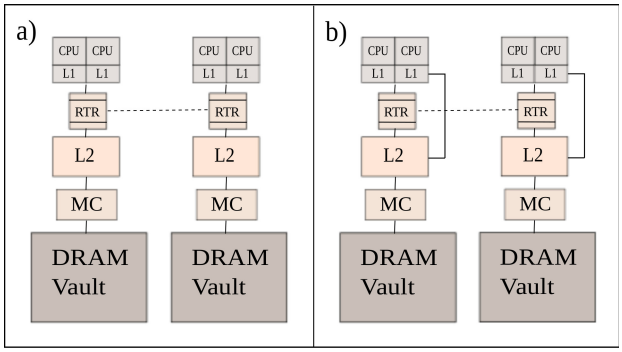


Figure 12: Memory-side cache organization a) w/o b) with L1-L2 links

The result is a memory organization, shown in Figure 12, in which the L2 is placed closed to the memory rather than the L1, making it similar to the memory-side cache organization. However, the cache is now banked into smaller units with no direct paths among these banks. Two configurations of the organization can be explored, one in which the link between the L1 and the local L2 cache bank is maintained (Figure 12 b)), and the other in which it is removed (Figure 12 a)). Although, removing the link will reduce performance by increasing the latency of requests destined to the local L2 bank, the reduction is not significant. On the other hand, the organization with no L1-L2 link is highly attractive, as it is extremely modular and scalable, with each component being designed separately in its own die.

5.2 DRAM Traffic Distribution - Neighbor Mapping

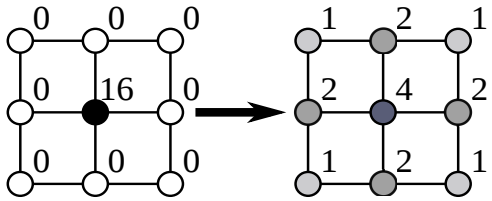


Figure 13: Distribution of pages to neighboring nodes in neighbor mapping scheme

Memory-side cache organization can potentially lead to congestion at a particular DRAM at a particular instant in time, as can be seen with the result of fluidanimate in Figure 15. For such cases, it may be better to distribute the traffic into different MCs, thus not using same mapping function. To explore this point, we defined an address space mapping that distributes pages mapped to an L2 bank or DRAM vault to neighboring banks or vaults. The idea here is to reduce congestion at one MC or L2 bank and increase memory level parallelism by spreading references in a local area that does not increase average hop count significantly. The goal is to better balance parallelism in memory references while keeping DRAM row buffer locality intact. We will refer to this as neighbor-page mapping or simply neighbor mapping. Figure 13 shows how neighbor mapping is performed. Each value in the figure represents the number of pages out of 16 consecutive pages of a particular bank or vault that are mapped to the neighboring banks or vaults. We are not giving details of which particular page is mapped to which immediate neighbor but it is decided based on four bits of the address and is consistent across all banks or vaults, making it a one-to-one function.

5.3 Keeping Data Local

From earlier discussions, it is clear that traffic on the network is a major component of the memory latency path. However, our baseline model distributes the traffic evenly across all the distributed L2 banks and MCs. In this section, we leverage the virtual-to-physical page allocation strategies

in order to maximize keeping the data to the local MC and the corresponding L2 bank. This reduces remote L1-L2 accesses, making a large number of them local, and resulting in an overall decrease in the average number of hops. Since, page allocation happens at the OS level, it is completely orthogonal to CAM and GAM. The challenge lies in the placement of the shared data which is accessed by multiple tiles simultaneously and at different program phases.

5.3.1 First-Touch Policy

A commonly used scheme that tackles this issue is the first touch policy, where pages are mapped to the banks or MCs whose corresponding cores access it for the first time. This will make all subsequent accesses of the data by that core local; making private only data to remain almost always local. Subsequent accesses for the shared data that happens at the other cores access it as a remote memory.

5.3.2 Sharing vs. Replication

We have not explored data replication in multiple L2 banks. Our address space is distributed among all the L2 banks . This can potentially lead to higher remote L2 accesses. Data replication could have been used to reduce these accesses However, we would like to point out that data replication at multiple L2 banks would have complicated the support for the memory-side cache organization, that is, although the data would have been replicated in the L2 banks, it would be in a single location in DRAM, resulting in remote L2-MC accesses. This would again increase the network traffic and its latency. The choice boils down to reducing remote L1-L2 traffic with a decreased L2 capacity (due to data replication) vs removing the remote L2-MC traffic completely with an increased cache capacity. We plan to extend the simulations to find the best tradeoff in future.

6. RESULTS OF THE IMPROVED MEMORY SYSTEM

This section compares the results of the improved memory system organization with that of the baseline system.

6.1 Simulation Environment

We have used the Manifold multicore simulation infrastructure [27] as our simulation environment. Our system simulator is organized as follows. The front-end is a multicore emulator called Qsim [8] that boots a Linux kernel and executes multi-threaded applications from SPLASH and PARSEC benchmark suites, generating x86 based instruction stream for each thread. These instructions are fed into a multi-core processor timing model. We ran 32 core simulations with 1 thread per core. The cores are fast forwarded until all of them start running. As explained earlier, two cores are concentrated in one tile. Loads and stores are sent to the two-level cache hierarchy. We used the mcp [2] model for cache simulations that uses multicore MESI protocol for coherence, as explained in 4.4.1. Iris [27] is used as the network timing model that models a two-stage pipeline router architecture with flit level flow control, as explained in [5]. Each router connects to two network Interfaces (NIs), one each on the cache and the MC side. The NI converts flits to/from cache level requests in its separate injection and ejection queues. The memory model is constructed using the open source DRAMSim2 memory simulator [16]. 16 vaults correspond to 16 instances of DRAMSim2. A single DRAM vault consists of 4 ranks, (equivalent to DRAM layers), and 2 banks per layer of 32MB each. Thus the total DRAM capacity of the system is 4GB (each vault = 256MB, each sub-layer per vault = 64MB). TSV latency across different layers is kept constant, which is equivalent to the DRAM bus speed. Configuration parameters for various system elements are shown in Table 1. DRAM timing parameters used are given in Table 2.

Components	Various Parameters & Values
Processor	Out-of-order, 6 stage pipeline, 2GHz, 2-wide issue/commit, 64-entry ROB, LSQ
L1 cache per core (8KB)	32 sets, 4-way, 64B lines, 8 MSHRs, LRU replacement, 2-cyc hit, 5-cyc lookup
L2 cache per tile (256KB)	256 sets, 16-way, 64B lines, 32 MSHRs, LRU replacement, 10-cyc hit, 20-cyc lookup
Network	4x4 torus, request reply, flit-size - 128 bits Pkt-size - 3 flits w/o data, 6 filts with data, baseline x-y routing (2VCs per virtual net.)
Router	6-port, 5 flits IB, 4VCs/port round robin SA, FCFS VCA
Memory controller	rank and bank round robin, close page, Addr-map - chan:row:col:bank:rank
DRAM config. per vault	64M/die/MC, 1-channel, 4-rank, 2-banks, 8KB row, and 64 bit bus @ 1333MHz

Table 1: System configuration

Parameter	Value - cycles (ns)
t_{CLK}: Clock cycle time	1 (1.5 ns)
t_{RP}: Row precharge time	9 (13.5 ns)
t_{RCD}: RAS to CAS delay	9 (13.5 ns)
t_{RC}: Row cycle time	33 (49.5 ns)
t_{RAS}: Row active time	24 (36 ns)
t_{CAS}: Column access latency	9 (13.5 ns)
t_{WR}: Write recovery time	9 (13.5 ns)
t_{WTR}: Write to read latency	1 (1.5 ns)
t_{RRD}: Row to row active delay	4 (6 ns)
t_{CCD}: Column to column delay	4 (6 ns)
t_{RTP}: Read to precharge delay	5 (7.5 ns)
t_{RFC}: Refresh period	60 (90 ns)
BL: Burst length	8
Refresh Count	8192

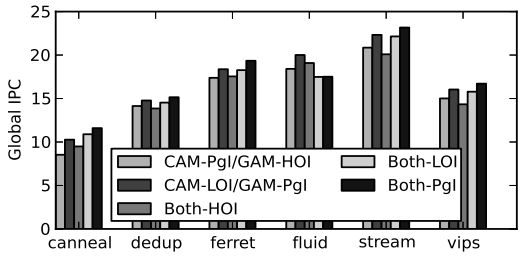
Table 2: DRAM timing parameters [24]

The processor, the cache, and the network execute at 2GHz and the MC (DRAMSim2) executes at 1GHz. We track the Global Instructions Per Cycle (IPC) - the total number of instructions executed across all threads divided by the total number of cycles, which in these experiments is set to be 100M (∼1200-2500 million instructions).

6.2 Address Mapping - Results

Figure 14 a) shows the Global IPC values for PARSEC applications with various address mapping functions. In the last three cases, both the shared L2 banks and the DRAM vaults use the same address mapping, which are either high-order interleaving (HOI), low-order interleaving (LOI), or page interleaving (PgI). The first two bars correspond to the use of a different mapping functions for both CAM and GAM.

We can see that the configurations that have different CAM and GAM do not perform well. The average increase of Both-PgI from CAM-LOI/GAM-PgI and CAM-PgI/GAM-HOI is 6% and 14%, respectively. This can be explained with Table 14 b), which illustrates latency values for different components in vips for all the 5 mapping functions.



(a) Global IPC

Latencies (cycles)	CAM-PgI GAM-HOI	CAM-LOI GAM-PgI	Both-HOI	Both-LOI	Both-PgI
Avg. Hops	11.65	11.94	10.61	10.70	10.68
All Req.	188.25	153.31	215.50	155.39	147.04
Req. From DRAM	313.70	208.03	297.67	221.40	189.00
L2-DRAM (Round Trip)	262.03	138.38	238.04	153.14	135.79
DRAM	217.79	88.87	203.26	118.62	105.50

(b) Latency distribution of vips

Figure 14: Various address mapping schemes

The average number of hops, (Row 1 of Table 14 b), have increased significantly in the first 2 cases. This in turn increases the overall network traffic, and hence reduces performance. Among the three cases with same address mapping for the L2 banks and the DRAM vaults, PgI performs the best. Table 14 b) illustrates the loss in performance for HOI and LOI, which is 14% and 8%, respectively, in comparison with Both-PgI. HOI will direct most of its requests in a particular time frame to a particular L2 cache and DRAM vault. This will put that DRAM under high load, resulting in an increased queuing delay, or DRAM latency (last row in the table). Furthermore, it creates network hot spots around that particular node, causing an increase in average latency in the network. LOI destroys any locality present in the memory reference stream, thus increasing latency within the DRAM. It also disperses the traffic across the network, putting higher load on it that results in an increase in the network latency. PgI, which balances locality, average hops and the DRAM load distribution, performs the best. This means that not only should we provide same mapping to both the L2 banks and the MCs, but also try to distribute the traffic among various DRAM channels, without increasing the average number of hops per request. All subsequent results use PgI for both GAM and CAM, unless otherwise stated.

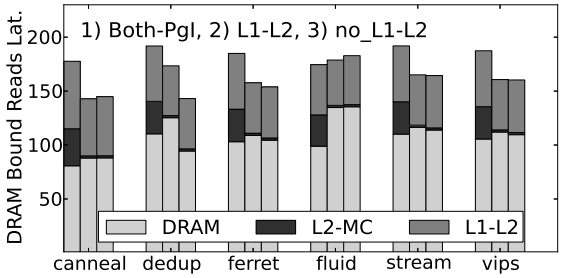


Figure 15: Latency distribution of mem-side cache organization

6.3 Effect of Mem-Side Caching

Figure 15 plots the latency distribution of DRAM bound reads with various memory organizations. The first bar, (Both-PgI), represents the case, where both CAM and GAM have been assigned the same mapping functions. The 2nd and 3rd bar, (L1-L2 and no_L1-L2) represents our memory-side cache organization, with and without the L1-L2 link, respectively. It can be seen that the organizations with a direct L2-MC connection almost removes the L2-MC latency, (It is equal to two cycles, one cycle both ways). Thus, even with higher load on individual DRAM vaults, the overall latency is reduced by 11.7%. fluidanimate is an exception, in which the increase in DRAM latency is high enough to surpass the advantages of reduced L2-MC latency. Furthermore, it should be noted that removing the L1-L2 link does not significantly increase the latency of DRAM bound reads except dedup. We use the configuration without the L1-L2 link and call it as memory-side cache organization.

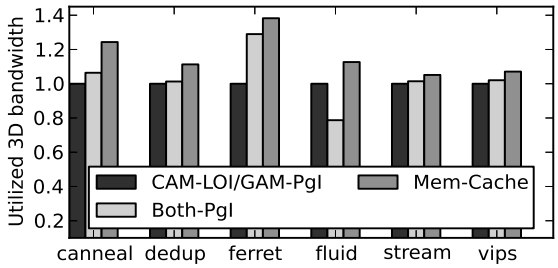


Figure 16: 3D bandwidth utilization normalized to baseline

Figure 16 presents the memory bandwidth utilization, (bandwidth across the DRAM channels), of various schemes normalized to the baseline CAM-LOI/GAM-PgI case. It can be observed that in both cases (that is, applying same mapping for CAM and GAM and the memory-side cache organization) average 3D bandwidth utilization has increased. The increase of Mem-Cache and Both-PgI from baseline is 16.4% and 3.1%, respectively. Small decrease in the case of Both-PgI with fluidanimate can be attributed to the increased DRAM latency (Figure 15), which results in the reduction of its 3D bandwidth utilization.

6.4 Neighbor Mapping - Results

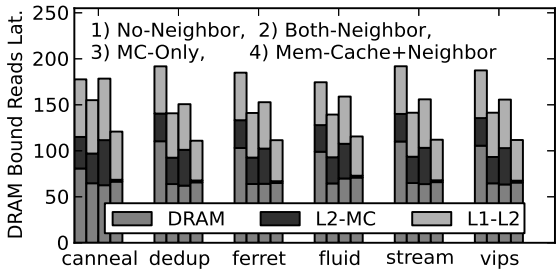


Figure 17: DRAM bound reads latency distribution with neighbor mapping

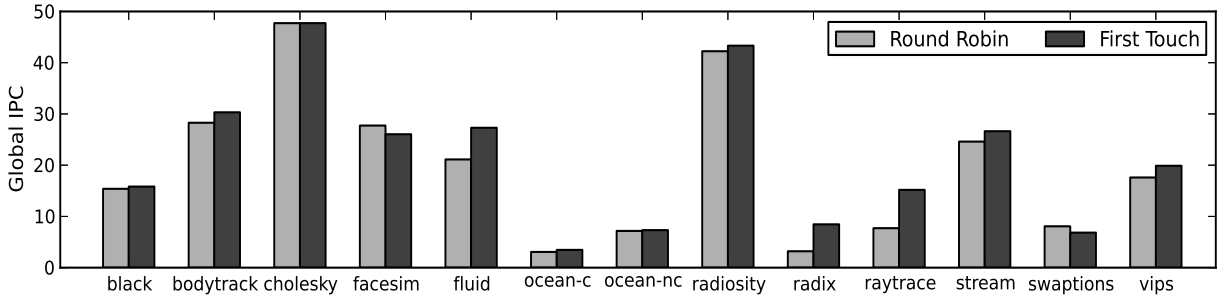


Figure 18: Global IPC with and w/o first touch policy

Figure 17 shows the latency distribution of reads destined to DRAM for neighbor mapping vs the case in which no neighbor mapping is used. The first bar represents the case without any neighbor mapping for both CAM and GAM. The second bar represents the case where neighbor mapping is used for both CAM and GAM. The 3rd bar represents the cases when neighbor mapping is performed only for GAM. All these cases performed neighbor mapping without the memory-side cache organization. The last bar added neighbor mapping for both CAM and GAM on top of the Mem-Cache organization.

In all neighbor mapping cases, the queuing latency within the DRAM is reduced. This indicates a good distribution of traffic into neighboring DRAM vaults that reduces load on one specific DRAM. However, the network latency (latency between L1-L2 and L2-MC) has increased significantly for the MC-only neighbor mapping case. This indicates high load on the router that first sends request to L2 and then immediately sends them to neighboring MCs making them a hot spot which resulted in large increase in latency. Finally, the case with neighbor mapping on top of Mem-Cache organization further reduces the latency with no L2-MC traffic. The IPC increase (not shown) averaged across all applications with neighbor mapping on top of Mem-Cache organization from the case with same CAM and GAM mapping is 11.3%, while the increase from Mem-Cache only without any neighbor mapping is 3.5%. Recall that this is in addition to the cases that do not have same CAM and GAM. We conclude that if the distribution of load among different vaults is required, it is better to provide this distribution at the L2 bank level and keep CAM and GAM the same, thus removing any L2-MC traffic in the horizontal direction.

6.5 First Touch Policy - Results

Fig 18 presents the global IPC of various benchmarks with and without the first-touch policy. The round-robin policy sequentially assigns pages, which with pgI means one page for each MC in a round-robin manner. It can be seen that first-touch improves IPC for most of the benchmarks. The average improvement over all the benchmarks is 9.6%. This improvement is attributed to the decrease in the average number of hops traveled per response, which is reduced by 5.8% (not shown).

7. CONCLUDING REMARKS AND FUTURE WORK

This paper addressed the problem of TSV bandwidth utilization of a 3D stacked processor memory system. First, we showed why increasing the number of channels to utilize the high bandwidth is a desirable choice for 3D systems. Then, we explored the impact of network latency and DRAM access time on a 3D system. We pointed out the fact that the 2D network latency is a bigger problem in these systems specially with higher number of DRAM channels. We proposed a new memory subsystem organization that places L2 cache banks next to DRAM with an interconnection network between only the L1 and the L2. We further explored distribution of traffic across different L2 banks and DRAM channels and showed that keeping the L2-MC traffic local using co-ordinated address translation mechanisms improve performance significantly.

In this paper, we only looked towards reducing network and DRAM latency by memory reorganization and traffic distribution. We have not considered changing topology or improving various policies at different stages of the memory hierarchy to suite the need of reduced DRAM access time. This will be explored as a key next step to this work.

8. ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation under grant CNS 0855110, Sandia National Laboratories and Interconnection Focus Center. We also acknowledge the detailed and constructive comments of the reviewers.

9. REFERENCES

[1] G. Allan. Ddr4 bank groups in embedded applications. May 2013.

[2] J. G. Beu, M. C. Rosier, and T. M. Conte. Manager-client pairing: a framework for implementing coherence hierarchies. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44 '11, pages 226–236, New York, NY, USA, 2011. ACM.

[3] E. Beyne. 3d system integration technologies. In *VLSI Technology, Systems, and Applications, 2006 International Symposium on*, pages 1 –9, april 2006.

[4] C. C. Chou, A. Jaleel, and M. K. Qureshi. Cameo: A two-level memory organization with capacity of main memory and flexibility of hardware-managed cache. In *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, pages 1–12. IEEE, 2014.

[5] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[6] S. Hassan, D. Choudhary, M. Rasquinha, and S. Yalamanchili. Regulating locality vs. parallelism tradeoffs in multiple memory controller environments. In *Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on*, pages 187 –188, oct. 2011.

[7] C. J.Lira and A.GonzÃąlez. Analysis of non-uniform cache architecture policies for chip-multiprocessors using the parsec benchmark suite. In *In Proceedings of the 2nd Workshop on Managed Many-Core Systems*, MMCS'09, Washington D.C, (USA), March 2009.

[8] Kersey, Chad. *QSim - QEMU-based Emulation Library for Microarchitecture Simulation.*

[9] C. Liu, I. Ganusov, M. Burtscher, and S. Tiwari. Bridging the processor-memory performance gap with 3d ic technology. *Design Test of Computers, IEEE*, 22(6):556 – 564, nov.-dec. 2005.

[10] G. H. Loh. 3d-stacked memory architectures for multi-core processors. In *ACM SIGARCH computer architecture news*, volume 36, pages 453–464. IEEE Computer Society, 2008.

[11] M. Motoyoshi. Through-silicon via (tsv). *Proceedings of the IEEE*, 97(1):43–48, Jan 2009.

[12] J. Pawlowski. Hybrid memory cube: Breakthrough dram performance with a fundamentally re-architected dram subsystem. In *Hot Chips*, 2011.

[13] S. Pugsley, J. Jestes, R. Balasubramonian, V. Srinivasan, A. Buyuktosunoglu, A. Davis, and F. Li. Comparing implementations of near-data computing with in-memory mapreduce workloads. *Micro, IEEE*, 34(4):44–52, July 2014.

[14] M. K. Qureshi and G. H. Loh. Fundamental latency trade-off in architecting dram caches: Outperforming impractical sram-tags with a simple and practical design. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 235–246. IEEE Computer Society, 2012.

[15] A. Ros, M. Acacio, and J. Garcia. Dico-cmp: Efficient cache coherency in tiled cmp architectures. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1 –11, april 2008.

[16] P. Rosenfeld, E. Cooper-Balis, and B. Jacob. Dramsim2: A cycle accurate memory system simulator. *IEEE Comput. Archit. Lett.*, 10(1):16–19, Jan. 2011.

[17] G. Sandhu. Dram scaling & bandwidth challenges. In *NSF Workshop on Emerging Technologies for Interconnects (WETI)*, Washington, DC, USA, Feb 2012.

[18] J. Sim, A. R. Alameldeen, Z. Chishti, C. Wilkerson, and H. Kim. Transparent hardware management of stacked dram as part of memory. In *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, pages 13–24. IEEE, 2014.

[19] Y. H. Son, O. Seongil, H. Yang, D. Jung, J. H. Ahn, J. Kim, J. Kim, and J. W. Lee. Microbank: Architecting through-silicon interposer-based main memory systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 1059–1070, Piscataway, NJ, USA, 2014. IEEE Press.

[20] J. Standard. Wide i/o single data rate (wide i/o sdr) (jesd229). Dec. 2011.

[21] J. Standard. Ddr4 sdram standard (ddr4) (jesd79-4a). Nov. 2013.

[22] J. Standard. High bandwidth memory (hbm) dram (jesd235). Oct. 2013.

[23] J. Standard. Low power double data rate 4 (lpddr4) (jesd209-4). August 2014.

[24] J. Standard. Wide i/o 2 (wideio2) (jesd229-2). August 2014.

[25] M. Walton. Hbm explained: Can stacked memory give amd the edge it needs? May 2015.

[26] Z. Wan, H. Xiao, Y. Joshi, and S. Yalamanchili. Co-design of multicore architectures and microfluidic cooling for 3d stacked ics. *Microelectronics Journal*, 45(12):1814–1821, 2014.

[27] J. Wang, J. Beu, R. Bheda, T. Conte, Z. Dong, C. Kersey, M. Rasquinha, G. Riley, W. Song, H. Xiao, P. Xu, and S. Yalamanchili. Manifold: A parallel simulation framework for multicore systems. In IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), March 2014.

[28] G. H. L. Yasuko Eckert, Nuwan Jayasena. Thermal feasibility of die-stacked processing in memory. December 2014.

[29] R. Yu. Foundry tsv enablement for 2.5d/3d chip stacking. August 2012.

[30] D. Zhang, N. Jayasena, A. Lyashevsky, J. L. Greathouse, L. Xu, and M. Ignatowski. Top-pim: throughput-oriented programmable processing in memory. In *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*, pages 85–98. ACM, 2014.

[31] D. H. K. et al, "3d-maps: 3d massively parallel processor with stacked memory," in *Solid-State Circuits Conference (ISSCC), 2012 IEEE International*, feb. 2012.