Another Trip to the Wall: How Much Will Stacked DRAM Benefit HPC?

Milan Radulovic
Barcelona Supercomputing Center (BSC)
Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain
milan.radulovic@bsc.es

Darko Zivanovic
BSC & UPC
Barcelona, Spain
darko.zivanovic@bsc.es

Daniel Ruiz
BSC
Barcelona, Spain
daniel.ruiz@bsc.es

Bronis R. de Supinski
Lawrence Livermore National Laboratory
Livermore, California
bronis@llnl.gov

Sally A. McKee
Chalmers University of Technology
Gothenburg, Sweden
mckee@chalmers.se

Petar Radojković
BSC, Barcelona, Spain
petar.radojkovic@bsc.es

Eduard Ayguadé
BSC & UPC, Barcelona, Spain
eduard@ac.upc.edu

ABSTRACT

First defined two decades ago, the memory wall remains a fundamental limitation to system performance. Recent innovations in 3D-stacking technology enable DRAM devices with much higher bandwidths than traditional DIMMs. The first such products will soon hit the market, and some of the publicity claims that they will break through the memory wall. Here we summarize our analysis and expectations of how such 3D-stacked DRAMs will affect the memory wall for a set of representative HPC applications. We conclude that although 3D-stacked DRAM is a major technological innovation, it cannot eliminate the memory wall.

CCS Concepts

•Hardware → Dynamic memory; •Computer systems organization → *Distributed architectures;*

Keywords

Memory wall, DRAM, bandwidth, latency, hybrid memory cube (HMC), high bandwidth memory (HBM), HPC

1. INTRODUCTION

In 1995, Wulf and McKee published a four-page note entitled "Hitting the Memory Wall: Implications of the Obvious" in the (unrefereed) ACM SIGARCH *Computing Architecture News* [27]. The motivation was simple: at the time, researchers were so focused on improving cache designs and developing other latency-tolerance techniques that the computer architecture community largely ignored main memory systems. The article projected the performance impact of the increasing speed gap between processors and memory. The study predicted that if the trends held, even with cache hit rates above 99%, relative memory latencies would soon be so large that the processor would essentially always be waiting for memory — which amounts to "hitting the wall".

This article was not the first to point out impending problems: Ousterhout had published "Why Aren't Operating Systems Getting Faster As Fast as Hardware?" [22] five years earlier. At the end of 1995, McCalpin demonstrated that current shared-memory High-Performance Computing (HPC) systems could typically sustain only 5–10% of the memory bandwidth needed to keep the floating-point pipelines busy [20], and in 1996 Burger, Goodman, and Kägi pointed out impending pin bandwidth limitations [3]. The memory wall note seemed to strike a nerve where the others did not, though, and it inspired a considerable backlash.

One set of arguments maintained that latency-tolerance techniques like out of order execution, wider instruction issue, and speculative techniques such as hardware prefetching would bridge the processor-memory performance gap. Even in combination, though, such approaches can mask the latency of only so many memory requests — the exact numbers depend on the sizes of the hardware structures. The cost and complexity of implementing larger and larger structures proved prohibitive, and although latency tolerance pushed the memory wall back, it did not save us.

Another set of arguments maintained that new memory technologies like Intelligent RAM (IRAM) [24] and Rambus Direct DRAM (RDRAM) [4] would eliminate the memory wall. In spite of years of hype, embedded DRAM (i.e., IRAM, or eDRAM) did not appear in commercial products for another five years, and then it was only cost-effective in special-purpose platforms like game consoles [6, 19]. eDRAM would not appear in general purpose processor chips for another decade [10, 18]. Rambus modified the DRAM interface and introduced narrow, high-speed channels that supported a peak bandwidth of 1.6 Gbytes/s — a significant improvement over other memory devices at the time — but taking full advantage of RDRAM capabilities required memory controller modifications that would introduce both design and verification costs. Intel released the Pentium 4 with all RDRAM (but without an optimized memory controller), but these systems came at higher cost and offered little or no performance gain. Subsequent products [12, 13] moved to DDR (double data rate) devices [16]. Rambus and others continue

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MEMSYS '15, October 05-08, 2015, Washington DC, DC, USA
© 2015 ACM. ISBN 978-1-4503-3604-8/15/10...$15.00
DOI: http://dx.doi.org/10.1145/2818950.2818955

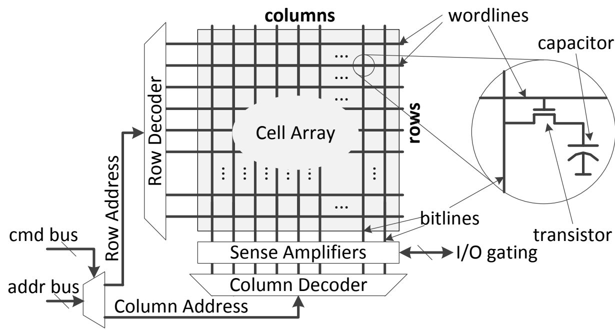

Figure 1: DRAM bank structure (from Cui et al. [5])

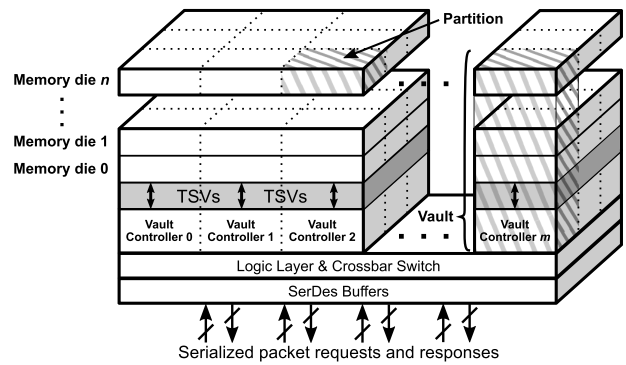

Figure 2: The internal structure of a Hybrid Memory Cube (HMC)

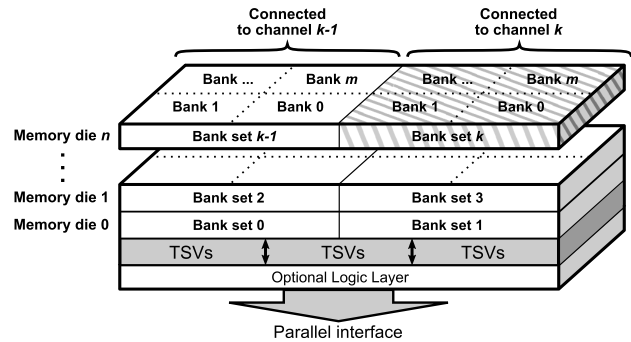

Figure 3: The internal structure of High Bandwidth Memory (HBM)

to deliver high-performance DRAM technology, but they have not (yet) removed the memory wall — latency remains limited by the speed of the underlying storage technology, and high bandwidth does not necessarily reduce latency.

Technological evolutions and revolutions notwithstanding, the memory wall has imposed a fundamental limitation to system performance for 20 years. 3D-stacking technology now enables DRAM devices that support much higher bandwidths than traditional DIMMs, and the first commercial products will soon appear [11, 17]. Some of the publicity surrounding these promising new devices suggests that they will break through the wall.

Here we summarize our analysis and expectations of how 3D-stacked DRAMs will affect the memory wall for a particular set of HPC applications. Recall that the memory wall was defined in terms of latency, not bandwidth. Higher bandwidth *may* lower average latency, provided that our applications offer sufficient memory-level parallelism (MLP) [9] and that CPU architectures can exploit it. But higher bandwidth by itself cannot guarantee better performance. How well we can exploit available bandwidth ultimately depends on the inherent MLP in our targeted workloads.

2. BACKGROUND

At the time of the original memory wall article, asynchronous fast-page mode DRAM was the industry standard. Variations on that technology delivered small performance boosts in specific scenarios. The next real innovation came when Rambus made DRAM transactions synchronous and allowed direct control of all DRAM resources concurrently with data transfer operations, enabling pipelined accesses over shared channels. Virtually all modern memory devices adopt a similar interface.

Figure 1 shows a typical DRAM bank organization. Each bank contains a 2D array of storage cells, row/column decoders, sense amplifiers, and peripheral circuits. To access data stored in the array, the memory controller sends an activate command to open the specified row, i.e., to load it into the row buffer (bank of sense amplifiers). The memory controller then sends one or more read commands to trigger read bursts to the open row. When the memory controller finishes with the row, it sends a precharge command to write the values back to the storage array and prepare for the next row activate command. Multiple banks can be accessed in parallel. Banks are organized into ranks, with DIMMs holding one rank per side. Channels connect ranks to the memory controller.

High-end CPUs usually increase memory channels and I/Os with each DDR generation, which increases memory system power consumption. Package size limits this costly approach. A promising alternative is 3D-stacked DRAM. These devices comprise several DRAM dies that are vertically connected with *through silicon vias* (TSVs). We summarize two 3D-stacked DRAM products that will soon be available: the Hybrid Memory Cube and High Bandwidth Memory.

2.1 Hybrid Memory Cube

The Hybrid Memory Cube (HMC) [11] claims to deliver up to 15× the bandwidth of module at 70% less energy and 90% less space. Figure 2 illustrates the HMC internal structure. HMC is composed of stacked DRAM dies (dies 0 to n) connected with TSVs and microbumps. Each die is divided into *partitions* vertically grouped into *vaults*. Each vault operates independently through a dedicated *vault controller* resembling the memory controller in DIMM-based systems. Finally, each vault is divided into banks much as in traditional DIMMs. The HMC includes a logic layer that redirects requests between off-chip serial interfaces and vault controllers. This logic layer also enables in-memory operations.

A high-speed serial interface connects the HMC to a CPU. The interface has two or four links, with each having four-, eight-, or 16-lane full-duplex serial lines. Lanes support data rates of up to 30 Gb/s, which means that per-device bandwidth can reach 480 GB/s (4 links×16 lanes×2×30 Gbit/s).

2.2 High Bandwidth Memory

High Bandwidth Memory (HBM) [17] is another emerging JEDEC standard. Figure 3 shows its internal structure. Like HMC, HBM consists of several 3D-stacked DRAM dies connected with TSVs. The HBM memory specification allows an optional logic base layer that can redistribute signals and implement logic functions. Each die is divided into banks that are grouped and attached to channels. The channels are independent: each accesses an independent set of banks with independent clocks and memory arrays.

Each HBM stack provides up to eight 128-bit wide channels. A 1024-bit parallel interface merges the channels. Maximum per-channel bandwidth is 32 GB/s, which implies 256 GB/s per device (eight channels×32 GB/s). As with DIMMs, a system can use independent HBM devices to deliver larger overall bandwidth.

2.3 Comparison

Both HMC and HBM are based on 3D-stacked DRAM, which increases package density to enable higher per-chip capacity. To reduce channel latency, the memory chiplets can be placed on a silicon interposer instead of a printed circuit board (PCB). TSV links shorten interconnection paths and reduce connectivity impedance. Thus, data can be moved at higher rates with lower energy-per-bit. HMC and HMB support a logic layer at the bottom of the DRAM stack; this could support in-memory computation that reduces the amount of data transferred between memory and CPU. Both devices target networking, a domain that traditionally requires high memory bandwidth. HBM also targets GPUs, while HMC targets High-Performance Computing (HPC).

HBM has a DIMM-like system organization: the memory controller is associated with the CPU, and a point-to-point parallel interface links it to the main memory. In contrast, HMC changes the system organization by placing the controller in the memory itself. Unlike DIMM-based architectures, each HMC device can directly connect to four devices via independent serial links. Device chains can thus provide an extended, high capacity memory, or even support a network of CPUs, GPUs and HMCs. In a network-like HMC system, remote HMC accesses that require multi-hop routing may have significantly higher latency, requiring an asynchronous interface. This implies higher variability in memory access times and lower timing determinism in the overall system.

A crossbar switch located in the HMC logic base routes memory requests through a network of HMC devices (see Figure 2). The CPU communicates with the HMC using high-level memory requests: it need not be aware of data location (device, vault, bank, row, and column) or memory-device timing parameters. Since the memory controller (i.e., vault controller) resides within the memory device, it can interact with the memory array more efficiently.

3. LATENCY VS. BANDWIDTH

Although memory latency and bandwidth are often described as independent concepts, they are inherently interrelated. Nonetheless, we examine the likely impact of 3D devices on them, in turn.

3.1 Memory latency

When analyzing memory latency, we distinguish between single-access latency in an idle system and latency in a system with many concurrent memory accesses.

Idle-system memory latency includes time spent in the CPU (load/store queues, cache memory, and on-chip memory controller), memory channel, and main memory. 3D-stacked DRAM does not change the time that memory requests spend in the CPU. DRAM technology determines time spent accessing the storage array, so that time is unlikely to decrease significantly. Placing 3D-stacked memory devices on a silicon interposer instead of a PCB reduces time spent in the memory channel. However, higher memory system complexity could *increase* memory latency. For example, HMC memory systems include serial links, serializing/deserializing logic, and more complex memory controllers on both the CPU and memory sides. Multi-hop memory accesses require routing and multiple channel access. Overall, 3D-stacked DRAM barely reduces request time in an idle system. In fact, the first memory system implementations with 3D-stacked DRAM devices still locate them on the PCB [2, 8], so idle-system memory latency will probably increase.

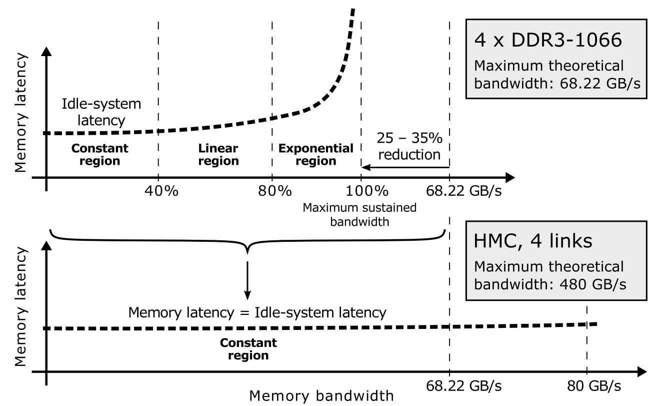

Figure 4: Bandwidth-latency curves of DDR3 and HMC systems

Full-system memory latency considers shared-resource contention among concurrent memory requests. Figure 4 shows the impact of such contention on memory latency [14]. The x axis shows application memory bandwidth, and the y axis shows the corresponding latency. This bandwidth-latency curve has three regions that are limited by the maximum sustainable bandwidth (which is 65-75% of the maximum theoretical bandwidth). First, when application bandwidth utilization is low, memory latency is constant and equals idle-system latency. This region has few concurrent memory requests and negligible contention for shared hardware resources. Jacob [14] observes that memory latency is constant until application bandwidth reaches approximately 40% of the maximum sustainable bandwidth. After this point, increases in bandwidth needs also increase contention for shared resources, which, in turn, increases memory latency. Memory latency increases linearly with application bandwidth usage in the region between 40% and 80% of sustainable bandwidth. Further increases in application bandwidth needs cause severe collisions among concurrent memory requests, and thus memory latency increases exponentially. Queueing theory also explains this dependency: each of the lines in Figure 4 shows *mean system response time as a function of request arrival rate* [15].

Figure 4 compares bandwidth-latency curves of a conventional and an emerging memory system to characterize the impact of 3D-stacked DRAMs on full-system memory latency. For the conventional memory system, we analyze a four-channel DDR3 memory running at 1066 MHz frequency. The maximum theoretical system bandwidth is 68.2 GB/s (four channels×frequency×bytes-per-transfer). For the emerging memory system, we analyze an HMC device. Note that our conclusions also apply to other 3D-stacked DRAMs. Lack of access to real hardware and to details of on-CPU memory controllers and in-memory logic (controller and request routing) complicates estimating the HMC bandwidth-latency curve. For purposes of our analysis, we feel that estimating that the constant-latency region of the curve will reach at least 25% of the maximum theoretical bandwidth, i.e., 80 GB/s, suffices.

Figure 4 illustrates the latency curve transition from conventional DDR3 to the HMC. The constant-latency region of the HMC (up to 80 GB/s) exceeds the maximum theoretical bandwidth of the conventional system (68.2 GB/s). Thus, it covers all three regions of the DDR3 system — constant, linear, and exponential. If an application is in the constant-latency region of the DDR3 system (i.e., when memory latency corresponds to idle-system latency), upgrading the memory to the HMC will not reduce memory latency, nor will it improve overall performance. If the application is in the linear or

exponential regions in the DDR3 system, a significant portion of its memory latency comes from collisions between concurrent memory requests. In this case, the bandwidth upgrade may reduce contention, which could reduce memory latency and improve performance.

3.2 Memory bandwidth

We also analyze whether high-bandwidth memories will increase effective memory bandwidth – what applications actually use. According to Little's Law [15], effective application bandwidth is directly proportional to the number of outstanding memory requests and inversely proportional to memory latency [1]:

$$effective\ bandwidth \propto \frac{outstanding\ memory\ requests}{memory\ latency} \quad (1)$$

Properties of the application (such as the portion of memory accesses in the overall instruction mix and data and control dependencies) and the CPU (such as core count, out-of-order issue, speculative execution, branch prediction, and prefetching) determine the number of outstanding memory requests. 3D-stacked DRAM will not change these parameters, and thus we expect that the number of outstanding memory requests to remain roughly the same. Therefore, effective application bandwidth with emerging 3D-stacked DRAM systems will increase only if memory latency is reduced.

3.3 Summary

3D-stacked DRAM devices will significantly increase available memory bandwidth. How well applications will exploit that higher bandwidth, though, ultimately depends on the workload's memory-level parallelism (MLP). For bandwidth-hungry applications that are in the linear or exponential regions of the bandwidth-latency curve, the bandwidth upgrade will reduce contention among concurrent memory requests, reduce memory latency, and improve performance. Lower memory latency will also increase effective application bandwidth. However, 3D-stacked DRAM cannot reduce idle-system memory latency. Memory latency, and thus effective bandwidth and overall performance, will not improve for applications for which lack of MLP limits effective bandwidth.

4. EXPERIMENTAL ENVIRONMENT

We conduct a preliminary evaluation of our analysis for a set of HPC applications running on a production system. This section describes our hardware platform and applications along with the methodology we use in the study.

4.1 Hardware platform

We conduct all experiments on a dual-socket Sandy Bridge-EP E5-2620 server. Each socket contains six cores operating at 2.3 GHz. We execute experiments with and without the supported two-way hyperthreading. When hyperthreading is disabled at the operating system level the platform appears to have 12 virtual CPUs (two sockets $\times$ six cores). With hyperthreading enabled, the OS sees twice as many. We fully utilize the hardware platform in all the experiments, i.e., we execute either 12 or 24 application processes. Each Sandy Bridge processor accesses main memory through four channels, and each channel connects to am 8 GB DDR3 DIMM, which gives 64 GB total server memory. We use a single server (not a large-scale HPC cluster), because memory bandwidth measurements require root privileges (see Section 4.3).

[1]Instead of equality as originally used in Little's Law, we use $proportional\ to$ ("$\propto$") to avoid converting between units that quantify memory bandwidth (GB/s) and memory latency (CPU cycles or nanoseconds).

We initially set the memory frequency to 1066 MHz, which makes the theoretical maximum memory bandwidth 68.2 GB/s. In order to analyze the performance impact (improvement) of higher memory bandwidth, we then increase the frequency to 1333 MHz (through the BIOS setup at boot time), which increases bandwidth by 25%. This change has no impact on memory latency, though: memory operation latencies are still limited by the DRAM technology. When memory frequency increases (i.e., the duration of a memory cycle decreases), memory commands take more cycles, so memory operation latencies remain practically the same.

4.2 HPC applications

We analyze a set of large-scale production HPC applications and two widely-used HPC benchmarks, Linpack and STREAM.

The **Unified European Application Benchmark Suite (UEABS)** [23] is comprised of programs that are representive of production applications running on large-scale Tier-0 and Tier-1 HPC systems in Europe. We choose four applications: ALYA, GROMACS, NAMD, and Quantum Espresso. We could not run the remaining applications because the input dataset sizes exceed the main memory capacity of our hardware platform.

The **High-Performance Linpack** [25] benchmark is used to rank the supercomputers on the TOP500 list [1]. The benchmark solves a dense system of linear equations; its computational intensity stresses the CPUs and GPUs.

The **STREAM** [21] benchmark measures a system's sustainable memory bandwidth. STREAM performs copy, scale, sum, and triad operations on long arrays (i.e., exceeding cache memory size). In order to stress the memory system, STREAM code is structured to minimize data reuse from the CPU caches.

4.3 Methodology

Memory bandwidth: We calculate the *maximum theoretical bandwidth* of the system based on the specification of the hardware platform under study — by multiplying memory frequency (1066 MHz or 1333 MHz) by the memory data bus width and number of memory channels. We measure *sustainable memory bandwidth* with the STREAM benchmark (a common approach [26]). We measure effective bandwidth via the Intel Performance Counter Monitor (PCM) library, which provides routines to access the memory controller performance counters[2]. In all experiments, we report total memory bandwidth, i.e., read and write memory traffic of all four channels.

Performance: All applications under study report their performance in their output files. For Quantum Espresso, ALYA, GROMACS, NAMD, and Linpack, performance corresponds to the number of elements that are processed in a time unit, which is directly proportional to the number of floating point operations per second. Performance of the STREAM benchmark corresponds to sustainable memory bandwidth.

5. RESULTS

First, we analyze the system without hyperthreading running with a memory frequency of 1066 MHz. The maximum theoretical memory bandwidth of this configuration is 68.2 GB/s, and the maximum sustainable memory bandwidth is 54.1 GB/s. Figure 5 shows the application memory bandwidth relative to the maximum sustainable bandwidth. STREAM, Quantum espresso (QE), and ALYA use a significant portion of the maximum sustainable bandwidth, and thus we expect that increasing available memory bandwidth

[2]Access to memory controller (uncore) performance counters requires root privileges.

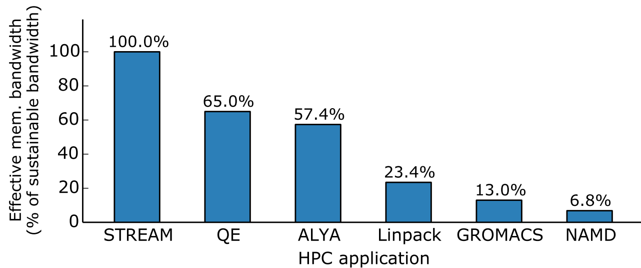

Figure 5: DDR3-1066: Portion of the maximum sustainable bandwidth (STREAM) that HPC applications actually use

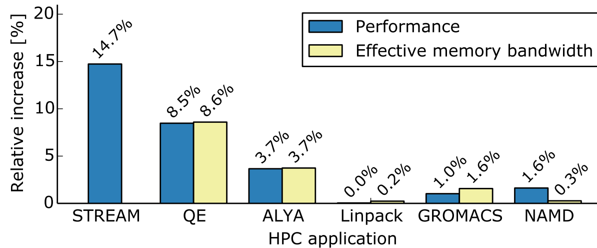

Figure 6: Performance improvement and effective memory bandwidth increase due to 25% memory bandwidth increment

will improve performance for these applications by increasing their effective memory bandwidth. On the other hand, Linpack, GROMACS, and NAMD use a small portion of the sustainable memory bandwidth — 23.4%, 13%, and 6.8%, respectively. We expect no significant performance improvements for these applications when available memory bandwidth increases.

In Figure 6, we quantify the impact of a 25% memory bandwidth increase on application performance. Performance improves by 14.7%, 8.5%, and 3.7% for STREAM, QE, and ALYA, respectively. This improvement clearly correlates with the memory bandwidth that the applications use in the baseline system configuration with DDR3-1066 memory. For Linpack, GROMACS, and NAMD, performance improves negligibly if at all — from 0% (Linpack) to 1.6% (NAMD). Effective memory bandwidth follows the same trend.

We repeat the experiments with hyperthreading enabled in order to understand the impact on effective memory bandwidth[3]. For all applications but Linpack, the results change insignificantly from those in Figures 5 and 6. Linpack changes memory behavior with hyperthreading: it becomes bandwidth-hungry — in the configuration with 1066 MHz memory, it uses 54% of the maximum sustainable bandwidth. When memory frequency increases to 1333 MHz, against our expectations, performance does not improve, nor does effective memory bandwidth increase.

To understand why Linpack does not benefit from the 25% bandwidth increase, we use a *roofline model* [26] to correlate the application performance with its *operational intensity*, or the number of the floating point operations (FLOPS) that it executes per byte of data transferred between the CPU and main memory. Figure 7 shows roofline models for our two systems, with 1066 MHz and 1333 MHz main memory, and the position of the Linpack benchmark in each of them. The x axis of the figure shows the application

[3]The OS views the platform as 24 virtual CPUs: 2 sockets×6 cores×2, so the workloads comprise 24 application processes.

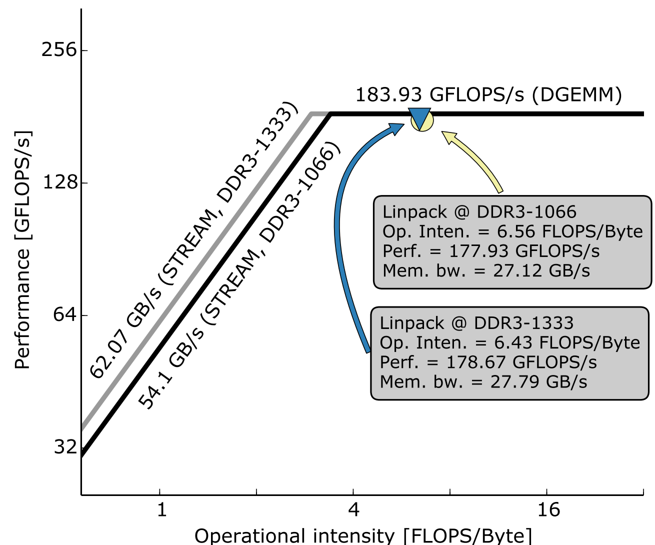

Figure 7: Position of Linpack application on platform roofline model, with DDR3-1066 and DDR3-1333 memory configurations

operational intensity, and the y axis shows application performance in GFLOPS/s. The sloped line shows the maximum sustainable memory bandwidth (as measured by the STREAM benchmark); this determines the upper performance bound for applications with low operational intensity. The horizontal line shows maximum sustainable performance in GFLOPS/s (determined by the DGEMM routine [7]).

Our roofline model shows that even though Linpack uses a significant portion of the available bandwidth, it touches the horizontal line of the chart, meaning that it is clearly limited by the GFLOPS/s that the CPUs can sustain. Increasing the memory frequency from 1066 MHz to 1333 MHz increases sustainable memory bandwidth and raises the inclined part of the roofline chart. However, it has no impact on the horizontal GFLOPS/s performance limit, and therefore does not improve Linpack performance.

To summarize, even when increasing available memory bandwidth mitigates collisions in the memory system, other parts of the system (processing units or interconnect) can still limit system performance. This result emphasizes the importance of building balanced computer systems that properly exploit the benefits of novel high-bandwidth memory solutions.

6. LOOKING FORWARD

How well applications will exploit the higher bandwidth provided by emerging 3D-stacked DRAMs ultimately depends on the workload's memory-level parallelism. For high-MLP applications, the bandwidth upgrade will reduce contention among concurrent memory requests, reduce memory latency, and improve performance. However, 3D-stacked DRAMs cannot reduce idle-system memory latency. Thus, they will not improve the performance of applications with limited MLP.

Further, we are unlikely simply to replace conventional DIMMs with the 3D-stacked DRAMs. Higher prices will prevent memories composed only of 3D devices and may even limit adoption. Instead, we are likely to see main memories that include both 3D devices and conventional DIMMs. Thus, in the best case, the system will still be bandwidth-limited, and it will often be latency-limited. So in contrast to the publicity surrounding 3D DRAMs, they are unlikely to break through the memory wall — at best, they move it.

Building balanced, high-performance systems will require us to design CPUs and memory controllers that can exploit the new devices for high-MLP application domains. The logic layers in the HMC and HBM offer interesting possibilities for in-memory processing and sophisticated memory controller functionality. 3D-stacked DRAM is certainly an interesting technological innovation. Finding a way to use this innovation to build high-performance systems, however, will take time — and the extent of its adoption will likely come down to cost.

7. ACKNOWLEDGMENTS

This work was supported by the Collaboration Agreement between Samsung Co., Ltd. and BSC, Ministry of Economy and Competitiveness of Spain (contract TIN2012-34557), Generalitat de Catalunya (2014-SGR-1272 and 2014-SGR-1051), Severo Ochoa Program (SEV-2011-00067) of the Spanish Government and European Union's 7th Framework Programme under project Mont-Blanc (grant agreement 288777 and 610402). Darko Zivanovic holds the Severo Ochoa grant (SVP-2014-068501) of the Ministry of Economy and Competitiveness of Spain. The authors thank Mario Nemirovsky and Damian Roca from BSC for their technical support.

References

[1] TOP500 List. http://www.top500.org/, Nov. 2014.

[2] Arira Design. Hybrid Memory Cube Evaluation & Development Board. http://www.ariradesign.com/hmc-board, 2013.

[3] D. Burger, J. R. Goodman, and A. Kägi. Memory Bandwidth Limitations of Future Microprocessors. In *Proc. ACM/IEEE International Symposium on Computer Architecture*, pages 78–89, May 1996.

[4] R. Crisp. Direct RAMbus technology: the new main memory standard. *IEEE Micro*, 17(6):18–28, Nov. 1997.

[5] Z. Cui, S. A. McKee, Z. Zha, Y. Bao, and M. Chen. DTail: a Flexible Approach to DRAM Refresh Management. In *Proc. International Conference on Supercomputing*, pages 43–52, June 2014.

[6] K. Diefendorff. Sony's Emotionally Charged Chip. *Microprocessor Report*, 13(5):1,6–11, Apr. 1999.

[7] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. S. Duff. A Set of Level 3 Basic Linear Algebra Subprograms. *ACM Transactions on Mathematical Software*, 16(1):1–17, Mar. 1990.

[8] Y. Durand, P. Carpenter, S. Adami, A. Bilas, D. Dutoit, A. Farcy, G. Gaydadjiev, J. Goodacre, M. Katevenis, M. Marazakis, E. Matus, I. Mavroidis, and J. Thomson. EUROSERVER: Energy Efficient Node for European Microservers. In *Proc. Euromicro Conference on Digital Systems Design*, pages 206–213, Aug. 2014.

[9] A. Glew. MLP yes! ILP no! *International Conference on Architectural Support for Programming Languages and Operating Systems, Wild and Crazy Ideas Session*, Oct. 1998.

[10] P. Hammarlund. 4th Generation Intel® Core™ Processor, codenamed Haswell. *Hot Chips 25*, Aug. 2013.

[11] Hybrid Memory Cube Consortium. Hybrid Memory Cube Specification 2.0. www.hybridmemorycube.org/specification-v2-download-form/, Nov. 2014.

[12] Intel Corporation. Intel® Pentium® 4 Processor and Intel® E7205 Chipset Design Guide. Dec. 2002.

[13] Intel Corporation. Intel® 875P Chipset Datasheet. Feb. 2004.

[14] B. L. Jacob. The memory system: You can't avoid it, you can't ignore it, you can't fake it. *Synthesis Lectures on Computer Architecture*, 4(1):1–77, 2009.

[15] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, Apr. 1991.

[16] JEDEC Solid State Technology Association. Double Data Rate (DDR) SDRAM Standard. www.jedec.org/standards-documents/docs/jesd-79f, Feb. 2008.

[17] JEDEC Solid State Technology Association. High Bandwidth Memory (HBM) DRAM. www.jedec.org/standards-documents/docs/jesd235, Oct. 2013.

[18] R. Kalla, B. Sinharoy, W. Starke, and M. Floyd. Power7: IBM's Next-Generation Server Processor. *IEEE Micro*, 30(2):7–15, Mar. 2010.

[19] A. Mandapati. 2001: A Graphics Odyssey. *Microprocessor Report*, 16(1):7–10, Jan. 2002.

[20] J. D. McCalpin. Memory Bandwidth and Machine Balance in Current High Performance Computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pages 19–25, Dec. 1995.

[21] J. D. McCalpin. STREAM: Sustainable Memory Bandwidth in High Performance Computers. https://www.cs.virginia.edu/stream/ref.html, 1997.

[22] J. K. Ousterhout. Why Aren't Operating Systems Getting Faster As Fast as Hardware? *USENIX Summer Conference*, pages 247–256, June 1990.

[23] Partnership for Advanced Computing in Europe (PRACE). Unified european applications benchmark suite. www.prace-ri.eu/ueabs/, 2013.

[24] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick. A Case for Intelligent RAM. *IEEE Micro*, 17(2):34–44, Mar. 1997.

[25] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary. HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. http://www.netlib.org/benchmark/hpl/, Sept. 2008.

[26] S. W. Williams, A. Waterman, and D. A. Patterson. Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures. *EECS Technical Report UCB/EECS-2008-134*, Oct. 2008.

[27] W. A. Wulf and S. A. McKee. Hitting the memory wall: Implications of the obvious. *ACM SIGARCH Computer Architecture News*, 23(1):20–24, Mar. 1995.