Designing Efficient Heterogeneous Memory Architectures

The authors' model of energy, bandwidth, and latency for DRAM technologies enables exploration of memory hierarchies that combine heterogeneous memory technologies with different attributes. Analysis shows that the gap between on- and off-package DRAM technologies is narrower than that found between cache layers in traditional memory hierarchies. Thus, heterogeneous memory caches must achieve high hit rates or risk degrading both system energy and bandwidth efficiency.

Evgeny Bolotin
David Nellans
Oreste Villa
Mike O'Connor
Alex Ramirez
Stephen W. Keckler
Nvidia

Recent packaging technologies that enable DRAM chips to be stacked inside the processor package or on top of the processor chip can lower DRAM energy-per-bit costs, provide wider interfaces, and deliver substantially higher memory bandwidth. However, these technologies are limited in capacity and come at a higher price than traditional off-package memories, so system designers must balance price, performance, and capacity tradeoffs. The most obvious means to achieve this balance is to employ both on- and off-package memory in a heterogeneous memory architecture. However, designers must then decide whether to deploy the on-package memory as an additional cache hierarchy level (controlled by hardware or software) or as a memory peer to the off-package DRAM in a nonuniform memory access (NUMA) configuration.

Figure 1 shows a generic memory hierarchy with increasing capacities, access latencies, and energy characteristics. Bandwidth and cost per bit decrease with the increase in distance from the computing core. Table 1 summarizes a memory hierarchy actual bandwidth, latency, and capacity values for an example quad-core CPU at 3 GHz. The gap in bandwidth and capacity between on-chip last-level caches and external DRAMs is usually higher than 10×. Tiers in the memory hierarchy can be explicitly managed by software (for example, CPU registers managed by the compiler or file-system buffers managed by the OS) or transparently managed by hardware (for example, static RAM [SRAM] caches). Hardware caches provide a graceful mechanism to improve latency and bandwidth without adding complexity in the software needed to detect and exploit locality. However, hardware caches introduce area and energy overheads due to tag and data management. With the introduction of on-package memories, the question must be reevaluated, yet again, to determine if the memory hierarchy has sufficient room for another tier of caching between these new heterogeneous memory technologies.

This article presents a model and analysis of energy, bandwidth, and latency for current

Published by the IEEE Computer Society

0272-1732/15/$31.00 © 2015 IEEE

and emerging DRAM technologies that enable an exploration of memory hierarchies combining heterogeneous memory technologies with different attributes. We will show that the gap between on- and off-package DRAM technologies is narrower that what is found between cache layers in legacy memory systems. Because of these small differentials, the overheads of hardware caching are magnified. Our results show that unless these caches can achieve high hit rates, they may degrade the system's energy efficiency. We also show that the overhead of data movement between the on-package cache and off-package memory can drastically reduce any potential bandwidth advantage. Finally, we argue that caching data in on-package memory provides little to no latency advantage. Our analytical model provides insight into the technology and application performance characteristics necessary to justify designing heterogeneous memory architectures.

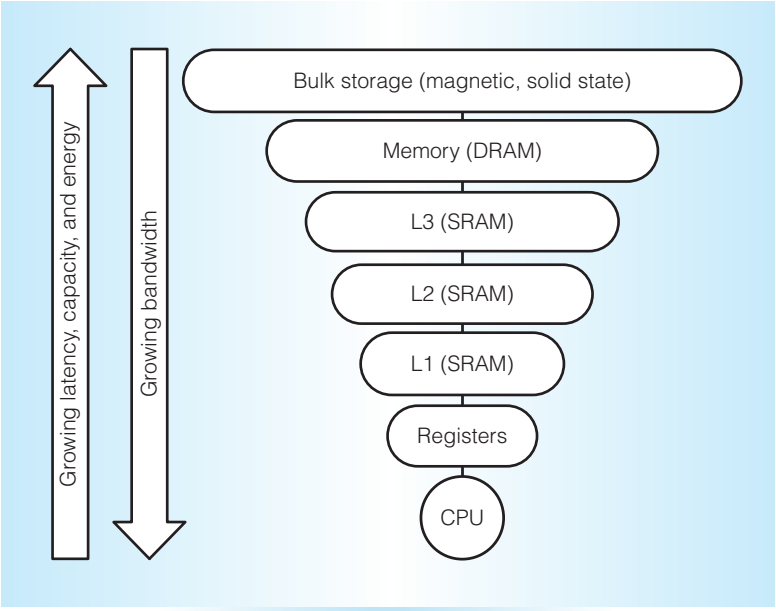


Figure 1. Capacity, latency, and bandwidth properties of a generic memory hierarchy. Traditional memory tier capacities, access latencies, and energy characteristics grow with the increase in distance from the computing core. The bandwidth and cost per bit decrease with additional levels of the hierarchy.

Heterogeneous memory design challenges

To better understand the current and future heterogeneous memory design challenges, we present historical DRAM technology trends and describe two specific state-of-the-art use cases taken from high-performance computing and mobile systems on a chip. We then define the two most common options for heterogeneous memory organization and discuss their architectural and efficiency challenges.

DRAM technology trends

DRAM has seen steady capacity and bandwidth improvements over successive generations for several decades. Broadly generalizing, each new DRAM technology generation has enabled a 4× increase in capacity, with a 2× increase in bandwidth and energy efficiency, whereas DRAM latency has remained relatively constant. Several DRAM technologies have been developed for specialized markets that make different tradeoffs, such as dropping multirank support and optimizing for improved energy efficiency (low-power DDR [LPDDR]) or bandwidth (graphics GDDR [GDDR]). LPDDR memory devices offer 30 to 50 percent energy-efficiency improvements over commodity DDR memories, whereas GDDR memory devices can provide 3 to 4× the bandwidth.

Different contemporary DRAM technologies have not provided a sufficient gap in capabilities to justify using one DRAM technology to cache another. However, stacked DRAM technologies, in which DRAMs are stacked with through-silicon-vias on or adjacent to the processor chip could change this. 3D stacking enables significant increases in bandwidth and energy efficiency but limits on the available footprint for memory stacks, as well as cost, will likely prevent the stacked DRAM from completely replacing commodity off-package DRAM.

Stacked DRAM systems can vary widely in relative bandwidth compared to a non-stacked DRAM system. In this article, we explore a spectrum of these ratios and show two specific use cases. In the first, a high-performance computing node supports 1 Tbyte per second of high-bandwidth memory (HBM) stacked DRAM and 120 Gbytes per second (GBps) of DDR4, providing a factor of 8.3× in bandwidth between the two classes of DRAM. This HBM memory is also roughly 3× more energy efficient than the

Table 1. A typical example of a memory hierarchy with bandwidth, latency, and capacity values for quad-core desktop CPU at 3 GHz.

Memory hierarchy	Aggregate bandwidth	Latency (cycles)	Capacity
DRAM (DDR4)	25 Gbytes per second (GBps)	121	Up to 32 Gbytes
L3 (on-chip)	0.3 Tbytes per second (TBps)	33	8 Mbytes
L2 (on-chip)	0.7 TBps	16	1,024 Kbytes
L1 (on-chip)	1 TBps	4	128 Kbytes
Registers	2 TBps	1	22 Kbytes

DDR4. In the second use case, a mobile system on a chip supports 51 GBps of stacked WideIO2 memory and 24 GBps of LPDDR4 memory, providing a bandwidth ratio of only 2.1×. The energy-efficiency gap is smaller as well, at roughly 1.8×. The latencies of these different memories are all similar, with current stacked DRAM standards providing minimal latency advantages over off-package devices. These parameters are considerably different from those of SRAM-based caches, which typically have a many-fold advantage in bandwidth, power, latency, and energy compared to DRAM (often 10× or more). This decreased differentiation warrants deeper investigation into when heterogeneous DRAM caching is appropriate.

Heterogeneous memory architectures

Figure 2 shows the two most common memory architectures that introduce an additional memory tier beyond the last-level on-chip SRAM cache. In Figure 2a, the stacked DRAM is architecturally beside the existing memory, resulting in a flat NUMA. In this organization, memory accesses are explicitly directed toward either the stacked or the bulk memory as determined by the physical address. In Figure 2b, the stacked DRAM is placed in front of the main memory. All memory requests will be serviced first by the stacked DRAM, and only cache misses will be routed to the main memory.

The NUMA organization provides the highest potential bandwidth, because both memories can be accessed in parallel. It also provides the lowest latency and energy because requests are never serialized between the two memories. However, this approach depends on the OS to allocate memory pages appropriately.[1] Furthermore, if the working set changes over time, the OS must perform page migration to maintain optimal performance.[2,3]

In the cache organization, the software is not aware of the two tiers of memory in the system. Hardware logic is responsible for performing the requested memory operation, regardless of which physical location the data may reside in. The replacement policy automatically detects and manages data locality. However, cache organizations are limited in bandwidth, unless they introduce some form of prediction to save unnecessary lookups and fills. They also pay additional latency and energy penalties if lookups are serialized.[4]

Cache implementations can also pay significant tag area and energy overheads. Reducing the tag management overhead while maintaining high hit rates is an active research area.[5–7] With on-package memories reaching multigigabyte capacities, mixed organizations that can expose the cache footprint to the OS as part of memory are also an area of active research.[4,8] Despite the overheads and design complexity inherent in caching solutions, computer architects often gravitate toward them because they do not require changes to the system software, and because the data placement in NUMA organizations is largely an unsolved problem.[1] However, our analytical model shows that the shrinking energy and bandwidth gaps between the two memory technologies might not always provide the expected advantages when adding a layer in the memory hierarchy, and that NUMA organizations might provide better efficiency in some scenarios.

Heterogeneous memory efficiency

To evaluate hierarchical DRAM architectures, we developed a parameterized, analytical energy and bandwidth model for memory-based caches. We examined and

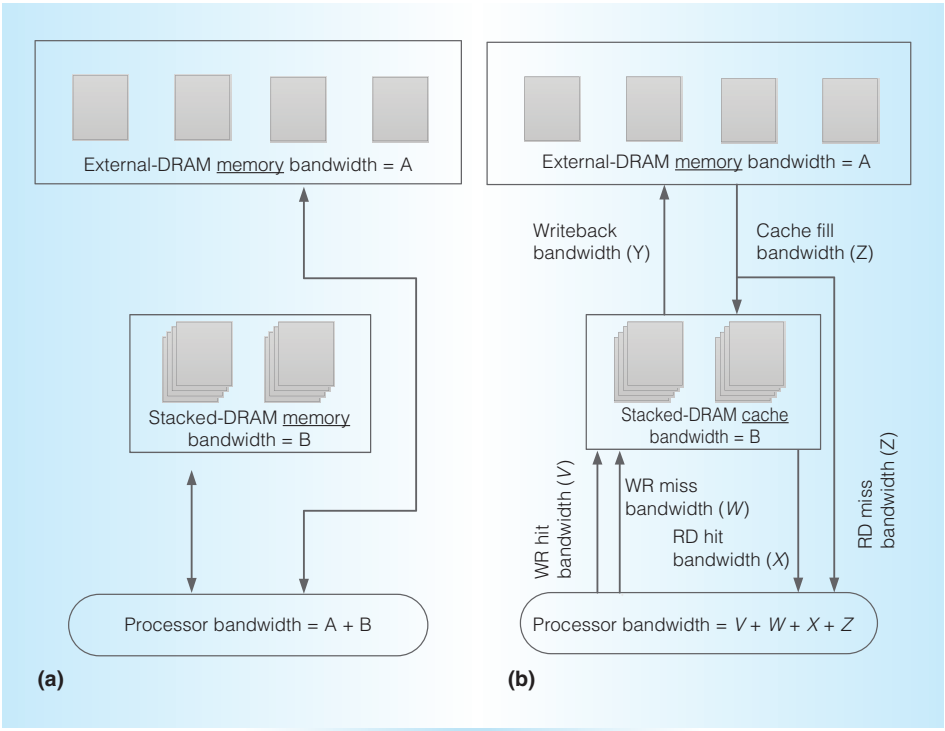


Figure 2. Two logical organizations of an additional stacked-DRAM memory layer. (a) OS-managed flat nonuniform memory access (NUMA) organization, where the stacked DRAM is architecturally beside the external memory. (b) Hardware-managed cache organization caching external DRAM, where stacked DRAM is placed in front of the external memory.

modeled flat NUMA and cache-like organizations (see Figure 2). In the NUMA organization, we assumed that the data is efficiently placed within the memories for optimal bandwidth efficiency. This idealized NUMA organization serves as an oracular memory system that can provide the sum of both external memory bandwidth and stacked memory bandwidth $A + B$ while not incurring any overheads due to page migration.

For the DRAM cache organization, we modeled a generic write-back cache architecture, wherein entries in the cache are allocated on read-miss and a modified entry is written to memory only when being replaced in the cache. On a write-miss, no data is fetched from the memory, and the cache merges multiple-writes for full and partial line writes. This design wastes neither memory bandwidth nor energy for write-streaming workloads with low hit rates.

Energy efficiency model

Our model focuses on data and tag access energy and assumes that cache control energy costs are insignificant. First, all cache accesses require a tag read and tag matching regardless of hit or miss. Second, there is a cache data access in case of a hit and a memory access and cache write to insert the new data in the case of a miss. This cache fill can result in a loss of efficiency if the cache line is not reused enough times to amortize the allocation cost. Third, when new data is allocated, a resident block must be evicted. If the block is not modified, no additional energy is required. However, if the data is modified, the cache must write it back to memory, requiring an extra cache read and a memory write. We do not consider other potential energy costs such as coherence traffic or leakage.

The inputs to our model are the memory and cache energy costs per data and tag access (E_{memory}, $E_{\text{cache-data}}$, and $E_{\text{cache-tag}}$, accordingly). Additional inputs are the cache hit rates, $P_{\text{read-hit}}$ and $P_{\text{write-hit}}$, and the probability for write, P_{write}. We define the energy savings from adding the caching layer (E_{savings}) as

$$
\begin{aligned}
E_{savings} = (&P_{read-hit} \times (1 - P_{write}) \\
&+ P_{write-hit} \times P_{write}) \\
&\times (E_{memory} - E_{hit}) \\
&+ (1 - P_{read-hit}) \\
&\times (1 - P_{write}) \\
&\times (E_{memory} - E_{read-miss}) \\
&+ (1 - P_{write-hit}) \times P_{write} \\
&\times (E_{memory} - E_{write-miss}), \qquad (1)
\end{aligned}
$$

where E_{hit}, $E_{read-miss}$, and $E_{write-miss}$ are the costs of hit and miss calculated below:

$$E_{hit} = E_{cache-data} + E_{cache-tag}. \qquad (2)$$

We developed a statistical model for the probability of a line being dirty (P_{dirty}), through the observation that a clean line in the cache becomes dirty when a write either modifies it (on hit) or replaces it and modifies it (on miss), given by ($P_{write} \times (1 - P_{dirty})$). Similarly, a dirty line becomes clean when a read misses the cache and replaces a dirty line that is written to memory: ($(1 - P_{write}) \times P_{read\text{-}miss} \times P_{dirty}$). In a steady state, these two opposite forces are equal, leading to

$$P_{dirty} = \frac{P_{write}}{P_{write} + P_{read-miss} - P_{write} \times P_{read-miss}} \qquad (3)$$

In the case where the hit rate is 0 percent, $P_{dirty} = P_{write}$ in steady state; when the hit rate is 100 percent, all lines will eventually become dirty because there are no read misses to force write-backs to memory to clean them (we make a simplifying assumption that all cache lines will be eventually written). Accounting for dirty line evictions, the cost of a cache miss becomes

$$
\begin{aligned}
E_{read-miss} = &2 \times E_{cache-tag} \\
&+ E_{memory} \\
&+ E_{cache-data} + P_{dirty} \\
&\times E_{victim}, \\
E_{write-miss} = &2 \times E_{cache-tag} \\
&+ E_{cache-data} + P_{dirty} \\
&\times (E_{victim} - E_{cache-tag}),
\end{aligned}
$$

and

$$
\begin{aligned}
E_{victim} = &E_{cache-tag} \\
&+ E_{cache-data} + E_{memory}.
\end{aligned}
$$

$E_{read-miss}$ accounts for the miss tag check, the memory access to get the data, and the cache tag and data writes, plus the probability of evicting a dirty line. For $E_{write-miss}$, we assume that the entire cache line will be replaced, so we do not need to read data from memory before writing the new cache line. Our approach for evaluating the energy efficiency is generic and can be leveraged to support a range of cache microarchitectures, assuming the correct set of input parameters is used. For example, in the case of recently proposed cache bypassing and hit-prediction mechanisms[5,6] that steer part of the traffic away from the cache (according to its prediction $P_{bypass-prediction}$), the improved hit and miss rate values should be substituted accordingly in Equations 1 and 3. The total energy gain in Equation 1 would need to be multiplied by the probability that the bandwidth is steered toward cache ($1 - P_{bypass-prediction}$) as well.

Energy efficiency analysis

Using our analytical model, Figure 3 summarizes the energy savings for a generic cache-memory pair as a function of the hit rate, for relative energy ratios between the two layers of memory ranging from 1 to 10. The savings are shown as a percentage of the main memory energy to show the energy improvement or loss when adding a caching layer. We analyze a typical scenario wherein write requests constitute 30 percent of memory accesses, equal hit rate for reads and writes, and tag access energy constitutes a modest 10 percent of the cache data access energy.

Figure 3 shows that as the energy gap between the memory and the cache shrinks, the relative cost of caching overheads increases, leading to diminishing (or negative) energy savings. Moreover, a system with a narrow energy gap must achieve substantially higher hit rates to reach positive energy savings than a pairing with a large energy gap. A cache that is 10× more energy efficient than the memory saves energy starting at an 18 percent hit rate, whereas an energy gap of 1.8× must achieve a 78 percent hit rate to break even on energy. The relative difference in energy savings is surprisingly small for energy ratios larger than 4× (the incremental difference is less than 5 percent), but it deteriorates dramatically for ratios lower

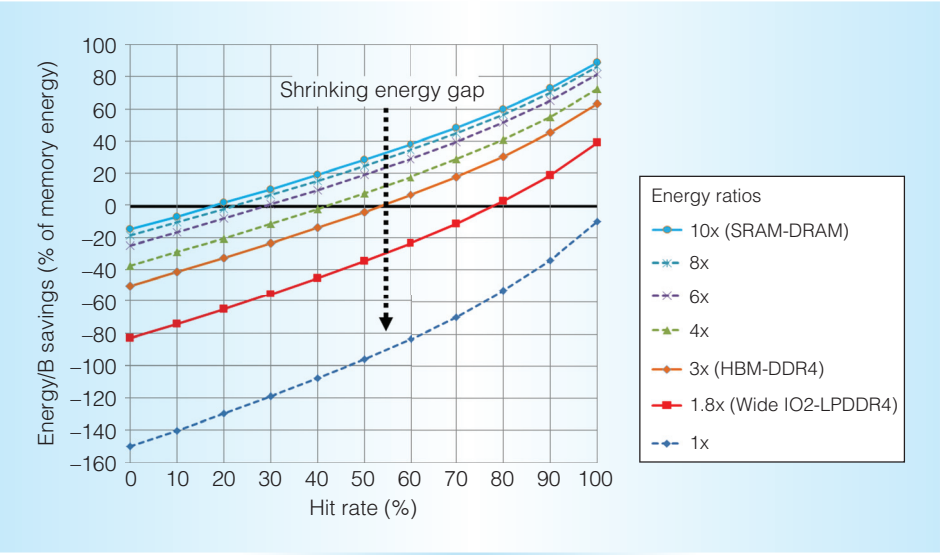


Figure 3. Cache energy savings normalized to memory energy for shrinking memory and cache energy ratios (for a typical scenario of 30 percent writes). As the energy gap between the memory and the cache shrinks, the relative cost of caching overheads increases, leading to diminished or even negative energy savings. Such systems must achieve substantially higher hit rates to reach positive energy savings.

than 4×. With this observation, we derive an energy-ratio rule of thumb of 4:1, as a design guideline (instead of the former 10×) for when it might be appropriate to build cache-memory pairings.

Although SRAM-DRAM technology pairings are attractive from an energy-efficiency viewpoint, an HBM-DDR4 pairing is borderline energy efficient and depends highly on the achieved cache hit rate. The combination of WideIO2-LPDRR4 will be energy negative unless hit rates above 80 percent are achieved. If tag access overheads of the caching implementation bloat from 10 to 30 percent, the HBM-DDR4 pairing will save energy only at greater than or equal to 70 percent hit rate; the WideIO2-LPDDR4 pair needs a hit rate above 90 percent to break even. Moreover, applying a perfect hit rate for writes does not strongly affect the results, because it decreases the break-even hit rate by only 10 percent for both technology pairs. We conclude that on-package DRAM organized as a cache is likely to be attractive only when both high hit rates can be achieved and tag overheads are minimized. Otherwise, system architects could find a NUMA organization more attractive for optimizing energy efficiency.

Bandwidth efficiency model

Our bandwidth model quantifies the relative bandwidth efficiency of a cache-based organization (Figure 2b) compared to an ideal NUMA organization (Figure 2a). We assume that the memory subsystem is fully loaded and serves as a performance bottleneck for the underlying processor. The model accounts for the bandwidth overheads associated with the management of data movement within the cache but ignores tag bandwidth overhead.

Using the notation introduced in Figure 2, we developed expressions for the variables *X, Y, Z, V,* and *W* as a function of memory and cache bandwidth (*A* and *B,* respectively), $P_{\text{read-hit}}$, $P_{\text{write-hit}}$, P_{write}, and P_{dirty}. Equations 4 and 5 are derived from the definition of a hit rate:

$$P_{\text{read-hit}} = \frac{X}{X+Z} \tag{4}$$

$$P_{\text{write-hit}} = \frac{V}{V+W}. \tag{5}$$

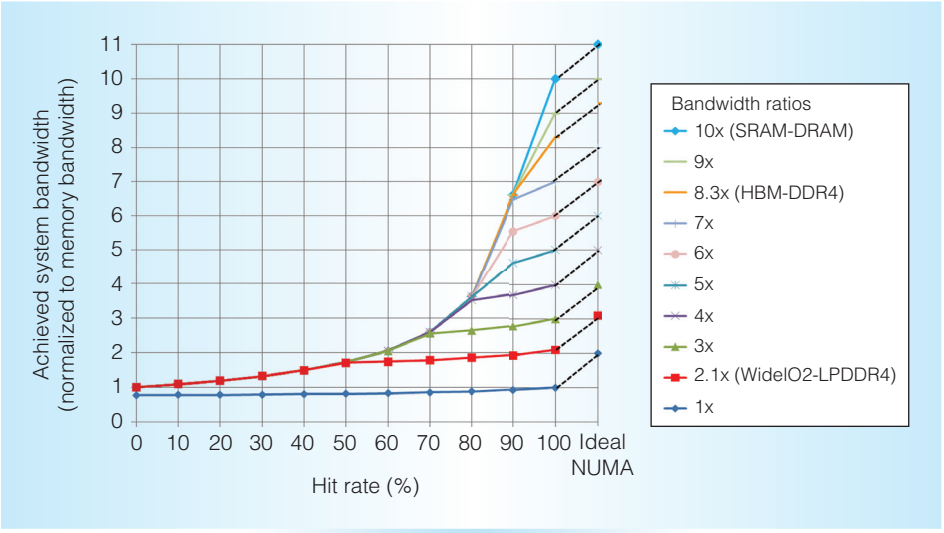


Figure 4. Achieved system bandwidth normalized to memory bandwidth for a range of memory-to-cache bandwidth ratios (for a typical scenario of 30 percent writes). The cache-based solutions can never achieve more than the cache bandwidth. Appealing bandwidth efficiency is achieved with relatively high hit rates and only in systems with sufficiently large bandwidth ratio between the two memory layers.

The probability of a write operation is defined as

$$P_{\text{write}} = \frac{V + W}{V + W + X + Z}. \quad (6)$$

Write-back bandwidth Y is defined as the fill bandwidth multiplied by the probability that a replaced line is dirty:

$$Y = (Z + W) \times P_{\text{dirty}}. \quad (7)$$

Finally, we define two boundary conditions. In Equation 8, the consumed cache bandwidth is less than or equal to the available cache data bandwidth, and in Equation 9, the consumed memory bandwidth is less than or equal to the overall available memory bandwidth:

$$V + W + X + Y + Z \leq B \quad (8)$$

$$Y + Z \leq A. \quad (9)$$

In a bandwidth-constrained scenario, Equations 8 and 9 become mutually exclusive, where either the cache or the memory serves as the bandwidth bottleneck. By solving Equations 4 through 7, and imposing the equality for either a cache (Equation 8) or a memory (Equation 9), we can identify the actual bandwidth limiter and compute the bandwidth achieved by the processor to be $V + W + X + Z$. Our approach for evaluating the bandwidth efficiency is generic and can easily be extended to evaluate other cache microarchitectural innovations. To model the recently proposed cache bypassing and hit-prediction mechanisms that steer a portion of the bandwidth away from the cache directly into the memory, the improved hit-rate values should be used in Equations 4 and 5. In addition, the direct processor-to-memory read and write bandwidth components should be added according to the probability of bandwidth steering ($P_{\text{bypass-prediction}}$) in Figure 2b and Equation 9.

Bandwidth efficiency analysis

Figure 4 shows the bandwidth achieved by various cache pairings relative to the bandwidth of a memory-only system. We show results for several cache-to-memory bandwidth ratios for a system with 30 percent write requests and an equal hit rate for reads and writes. The points on the far right of the graph are the idealized NUMA bandwidths achievable if both memories were fully utilized.

Figure 4 shows that cache-based solutions can never achieve more than the cache bandwidth (without using advanced bypassing

techniques) because cache misses are always filled into the cache and consume its bandwidth. We also see that total bandwidth grows with hit rate, and the maximum bandwidth is only available in an idealized NUMA organization. For low (1 to 3×) and medium (4 to 7×) bandwidth ratios, the achieved bandwidth quickly flattens out, yields diminishing returns at different points, and does not improve substantially at higher hit rates. At this point, the cache becomes a bandwidth bottleneck, and the memory can fully supply the miss bandwidth demand. For larger ratios (8 to 10×), bandwidth improvements continue scaling as hit rates approach 100 percent.

We conclude that cache-based heterogeneous memory organizations are appealing only in systems that achieve sufficiently high hit rates. Otherwise, the bandwidth resources might not be sufficiently utilized. If high cache hit rates can be achieved, the bandwidth ratio between two layers must still be sufficiently large; otherwise, cache-based organizations will be capped at low resource utilization. Therefore, when the bandwidth ratio is relatively low, system architects might want to consider a NUMA memory organization rather than a cache organization. For example, if the cache hit rate is higher than 70 percent, the cache-based solution can achieve bandwidth utilization higher than 70 percent (when compared to idealized NUMA) for cache-memory pairs only when bandwidth ratios are larger than 4×, leading to a bandwidth-ratio rule of thumb of 4:1. These results indicate that SRAM-DRAM and HBM-DDR4 cache-memory technology pairs are attractive for obtaining bandwidth efficiency, whereas WideIO2-LPDRR4 combination is not; combined with a low energy efficiency, this combination is unlikely to make sense in a cache-like organization.

Latency efficiency

Typically, on-chip SRAM has a large (at least 10×) latency advantage over off-chip DRAM. As a result, SRAM caches improve effective memory system latency substantially, in addition to improving energy efficiency and bandwidth. However, when considering caches comprised of technologies built from similar DRAM technologies, this latency advantage all but disappears. As we described earlier, DRAM latency is dominated by the core DRAM cycle time, which is largely unchanged between technologies. When organizing DRAM memories as a cache, the best latency achievable will be the unloaded latency to one of the memories. If a cache implementation requires lookups in both memories, the average latency to memory will effectively increase owing to tag lookup and potential misses in the cache. Recent work on hit prediction has shown it is possible to efficiently predict where your data will be (in cache or memory) with high accuracy, thus limiting latency increases when using DRAM caches. Unfortunately, even with these techniques, there is little to no improvement in effective latency, unlike traditional SRAM-based caches.

The bandwidth and energy advantages of emerging stacked DRAM memory technologies offer new opportunities and challenges to memory system architects. In this article, we showed that cache architectures can be feasible when a high cache hit rate is achieved and the energy and bandwidth ratios between stacked and on-package DRAM are at least 4×. A smaller ratio of those metrics could motivate the use of the on-package or stacked memory as a peer memory with external DRAM in a NUMA architecture. Such an architecture eliminates the hardware overhead and complexity of caching approaches, which could make a NUMA architecture more attractive in many applications. However, effective use of these memories will likely require innovations in data placement and migration to simultaneously exploit the total bandwidth available in the system and provide greater bandwidth and lower energy to frequently accessed data structures. MICRO

References

1. N. Agarwal et al., "Page Placement Strategies for GPUs within Heterogeneous Memory Systems," *Proc. 20th Int'l Conf. Architectural Support for Programming Languages and Operation Systems*, 2015, pp. 607–618.

2. N. Agarwal et al., "Unlocking Bandwidth for GPUs in CC-NUMA Systems," *Proc. 21st Int'l Symp. High Performance Computer Architecture* (HPCA), 2015, pp. 354–365.
3. M. Dashti et al., "Traffic Management: A Holistic Approach to Memory Placement on NUMA Systems," *Proc. 18th Int'l Conf. Architectural Support for Programming Languages and Operation Systems*, 2013, pp. 381–394.
4. 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," *Proc. Int'l Symp. Microarchitecture*, 2014; doi:10.1109/MICRO.2014.63.
5. D. Jevdjic, S. Volos, and B. Falsafi, "Die-Stacked DRAM Caches for Servers Hit Ratio, Latency, or Bandwidth? Have It All with Footprint Cache," *Proc. 40th Ann. Int'l Symp. Computer Architecture*, 2013, pp. 404–415.
6. D. Jevdjic et al., "Unison Cache: A Scalable and Effective Die-Stacked DRAM Cache," *Proc. 47th Ann. IEEE/ACM Int'l Symp. Microarchitecture*, 2014, pp. 25–37.
7. N. Gulur et al., "Bi-Modal DRAM Cache: Improving Hit Rate, Hit Latency and Bandwidth," *Proc. 47th Ann. IEEE/ACM Int'l Symp. Microarchitecture*, 2014, pp. 38–50.
8. J. Sim et al., "Transparent Hardware Management of Stacked DRAM as Part of Memory," *Proc. 47th Ann. IEEE/ACM Int'l Symp. Microarchitecture*, 2014, pp. 13–24.

Evgeny Bolotin is a senior research scientist in the architecture research group at Nvidia. His research interests include computer systems architecture, memory systems, and energy-efficient architectures. Bolotin has a PhD in electrical engineering from the Technion–Israel Institute of Technology. He is a member of IEEE. Contact him at ebolotin@nvidia.com.

David Nellans is a senior research scientist in the architecture research group at Nvidia. His research interests include OS and computer architecture interaction, nonvolatile memory, and storage systems. Nellans has a PhD in computer science from the University of Utah. Contact him at dnellans @nvidia.com.

Oreste Villa is a senior research scientist in the architecture research group at Nvidia. His research interests include system-level computer architecture, high-performance computing, and distributed runtimes. Villa has a PhD in computer science from Politecnico di Milano. Contact him at ovilla @nvidia.com.

Mike O'Connor is a senior manager and research scientist at Nvidia. His research interests include memory systems for GPUs and heterogeneous processors. O'Connor has an MS in electrical and computer engineering from the University of Texas at Austin. He is a senior member of IEEE and a member of the ACM. Contact him at moconnor@nvidia.com.

Alex Ramirez is a principal research scientist in the architecture research group at Nvidia. His research interests include energy-efficient supercomputing, heterogeneous multicore architectures, hardware support for programming models, and simulation techniques. Ramirez has a PhD in computer science from the Universitat Politècnica de Catalunya. He is a member of the ACM. Contact him at aramirez@nvidia .com.

Stephen W. Keckler is the senior director of architecture research at Nvidia and an adjunct professor of computer science at the University of Texas at Austin. His research interests include parallel computer architectures, memory systems, and accelerator architectures. Keckler has a PhD in computer science from the Massachusetts Institute of Technology. He is a fellow of IEEE and the ACM. Contact him at skeckler @nvidia.com.

cn *Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.*