Performance Potential of Mixed Data Management Modes for Heterogeneous Memory Systems

T. Chad Effler
Oak Ridge National Laboratory
Oak Ridge, Tennessee
Email: efflertc@ornl.gov

Michael R. Jantz
University of Tennessee
Knoxville, TN
Email: mrjantz@utk.edu

Terry Jones
Oak Ridge National Laboratory
Oak Ridge, Tennessee
Email: trjones@ornl.gov

Abstract—**Many high-performance systems now include different types of memory devices within the same compute platform to meet strict performance and cost constraints. Such heterogeneous memory systems often include an upper-level tier with better performance, but limited capacity, and lower-level tiers with higher capacity, but less bandwidth and longer latencies for reads and writes. To utilize the different memory layers efficiently, current systems rely on hardware-directed, memory-side caching or they provide facilities in the operating system (OS) that allow applications to make their own data-tier assignments. Since these data management options each come with their own set of trade-offs, many systems also include mixed data management configurations that allow applications to employ hardware- and software-directed management simultaneously, but for different portions of their address space.**

Despite the opportunity to address limitations of stand-alone data management options, such mixed management modes are under-utilized in practice, and have not been evaluated in prior studies of complex memory hardware. In this work, we develop custom program profiling, configurations, and policies to study the potential of mixed data management modes to outperform hardware- or software-based management schemes alone. Our experiments, conducted on an Intel® Knights Landing platform with high-bandwidth memory, demonstrate that the mixed data management mode achieves the same or better performance than the best stand-alone option for five memory intensive benchmark applications (run separately and in isolation), resulting in an average speedup compared to the best stand-alone policy of over 10%, on average.

I. Introduction

Memory technologies with different performance and capabilities than conventional SDRAM have emerged. Many high-performance computing platforms now package conventional memory DIMMs together with devices containing high bandwidth, but limited capacity, "die-stacked" DRAM, such as HBM [1], [2] or MCDRAM [3], or alongside large capacity, non-volatile, memory modules, such as Intel®'s Optane™ DCPMMs.[1] These multi-layer memory architectures have disrupted the traditional notion of memory as a single block of volatile storage with uniform performance. Assigning data with high amounts of reuse to the correct tier can improve performance dramatically for applications that are too large to fit within the fastest memory tier.

Propelled by this shifting architectural landscape, system designers and researchers have developed new strategies to allocate and move data efficiently across the memory hierarchy. One common approach is to exercise the faster, smaller capacity tier(s) as a hardware-managed cache. For example, Intel®'s Cascade Lake platform includes a "memory-mode" option, which applies this approach with DDR4 as a large direct-mapped cache to Optane DC memory [4]. While hardware-managed caching provides some immediate advantages, such as software-transparency and backwards compatibility, it is inflexible, often less efficient, and reduces the system's available capacity.

Alternatively, software-based data tiering uses either the OS by itself, or the OS in conjunction with the application to assign data into different memory tiers, with facilities to allow migrations of data between tiers as needed. Some multi-tier memory systems also provide APIs that allow applications to control the placement of their data objects through the use of source code annotations [5], [6]. These finer-grained controls permit the user to coordinate data-tier assignments with knowledge of data allocation and usage patterns originating from the application itself, potentially exposing powerful efficiencies.

While software-based data tiering can enable significant speedups, it has some severe drawbacks compared to hardware-based caching. Specifically, this approach requires experts with knowledge of application data usage to update, build, and distribute annotated binaries. Researchers have recently developed custom compiler and runtime tools that can reduce or eliminate much of these burdens [7]–[11], but these approaches still require offline program profiling, analysis, and recompilation in order to be effective. Another significant limitation, which has not been addressed in practice or in research, is that all of these software-based approaches employ static tier assignments or only migrate data in very infrequent intervals. Hardware-based strategies are typically much more adaptive because there is no need to synchronize low-level data movement with the upper-level software.

As both hardware- and software-directed data management have their own advantages, many heterogeneous memory platforms now include *mixed* or combined HW/SW data management modes. Such mixed modes allow applications to employ both hardware- and software-based data management simultaneously, but for different portions of their address space. For example, hybrid mode on the Intel® Knights Landing (KNL) architecture allows applications to allocate and

[1]DCPMM stands for Data Center Persistent Memory Module.

use $\frac{1}{4}$ or $\frac{1}{2}$[2] of the capacity of the MCDRAM tier directly, while the remaining capacity is managed as a hardware-directed cache [12]. In this scheme, all data objects that are not directly allocated to or cannot fit within the software-reserved portion of MCDRAM are always accessed through the MCDRAM cache. Alternatively, the mixed mode on the Intel® Cascade Lake platform uses the entire conventional (DDR SDRAM) memory tier as a cache for data on the storage class (Optane DCPMM) memory tier, but also allows applications to withhold data objects from the in-memory cache [4].

While such mixed data management modes provide an opportunity to address the limitations of either hardware- or software-directed data management alone, the task of designing policies and strategies that use mixed mode approaches presents some significant research challenges. In particular, the system or application designer must determine which sets of data will access the faster, smaller memory tiers through hardware-directed caching, and which sets will be software-managed. The best configuration depends on a variety of parameters, including the performance, capacity, and caching scheme of each memory hardware tier, as well as the data allocation and usage patterns of each application. However, the community currently lacks tools for understanding and evaluating how these factors impact the effectiveness of mixed data management modes.

This work examines the performance potential of mixed HW/SW data management for heterogeneous memory systems. It presents a custom framework, based on the Simplified Interface to Complex Memory (SICM) runtime system [13], for evaluating application performance with mixed HW/SW data management configurations. Our developed framework also adopts and extends the MemBrain approach [10], [11] (which has been previously integrated into SICM), and uses it to automatically partition application data into different sets for hardware- and software-based data management. We deploy our framework Intel® KNL platform with two memory hardware tiers (MCDRAM and DDR). Using this platform, we evaluate a simple mixed HW/SW data management configuration with a set of five memory intensive applications from the CORAL-2 [14] and SPEC CPU 2017 benchmark suites [15]. Additionally, we develop and present a custom profiling tool with potential to increase the performance of mixed HW/SW data management by automatically identifying sets of application data that can be managed effectively with hardware-directed caching.

This study makes the following important contributions:

1) We find that even a simple mixed HW/SW management configuration achieves the same or better performance than the best standalone hardware- or software-based management scheme for all of our benchmark applications on our experimental platform. On average, the mixed HW/SW management approach outperforms the best standalone approach by more than 10%.
2) We demonstrate a custom profiling tool that records the distribution of accesses to the pages allocated by each program allocation site. Sites that allocate small groups of intensely accessed pages are better candidates for management with hardware-based caching, while sites that generate uniform bandwidth to a large set of pages are better managed in software.

The rest of this paper is organized as follows. Section II presents recent research in hardware- and software-directed data management for multi-tier memory systems, and contrasts these approaches with this work. Section III describes the tools and framework we have adopted as the basis of this work. Section IV describes the extensions that this work makes to the adopted software framework and tools. Section V describes our experimental platform and methodology and also presents and analyzes our results. Section VI discusses several directions for future work, and Section VII concludes the paper.

II. Related Work

A. Hardware-Managed DRAM Caches

Architecting DRAM as a large, hardware-managed cache imposes some unpalatable design choices for hybrid memory systems. In particular, the faster memory tier must either be implemented as a tagless direct mapped cache or it requires logic and storage for associative tags. These issues become even more problematic as the capacity of the hardware-managed tier(s) increases, and so scalability of this technique is a concern.

Some works have proposed architectural strategies to address these issues, for example, by: collocating tags and data in DRAM to increase efficiency [16], [17], keeping track of cache contents in TLBs and page tables to reduce metadata traffic [18]–[20], or swapping data lines out of the cache to preserve capacity [21], [22]. Mittal and Vetter provide a (2016) survey of this research [23]. In contrast to these works, this research does not propose any architectural modifications or techniques. Our approach leverages existing architectural capabilities to implement a framework that supports mixed HW/SW data management.

B. Software-Directed Data Management

Research in software-directed data management has primarily focused on building tools and techniques to facilitate the assignment of data to memory tiers. Some prior works integrate coarse-grained architectural profiling with page-level management in the OS [24]–[26]. Other projects allow applications to tag and profile certain program data, and then use classification heuristics to assign data to the appropriate tier [27]–[30].

While these efforts demonstrate that application guidance can be useful for certain usage scenarios, they require manual source code modifications or expensive online detection to attach recommendations to data objects. To address this limitation, some frameworks, including the SICM and MemBrain projects adopted in this work, employ static and lightweight

[2]The exact amount is configurable at boot time.

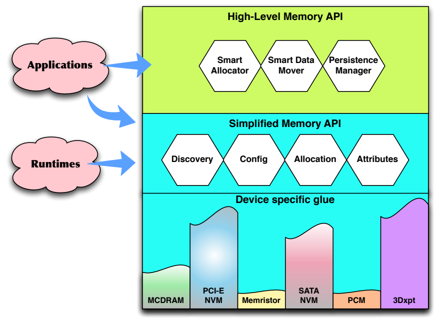


Fig. 1: SICM overview [13]. The high level provides a portable interface for applications, while the low-level implements efficient data management for complex memories.

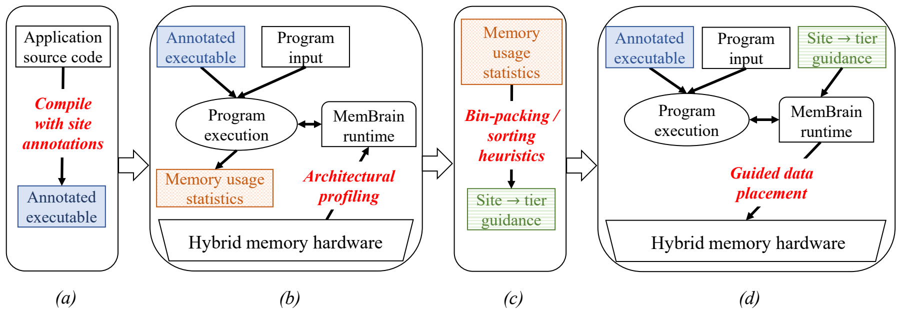


Fig. 2: Application guidance workflow [10]. (a) Compile executable with source code annotations at each allocation site, (b) Profile memory usage of each site in a separate program run using architectural sampling, (c) Employ bin-packing/sorting heuristics to assign data-tier recommendations to each site, (d) Apply data-tiering recommendations during subsequent program executions

runtime tools to attach memory usage guidance to program data *automatically* [10], [11], [31]. However, all of these previous works, employ software-based data management alone with static, or mostly static tier recommendations. Some studies [8], [9], [31] have implemented adaptive placement policies, but these approaches only migrate a small amount of data at relatively infrequent intervals to manage overheads. In contrast to all of these previous works, this work integrates high-level application guidance with mixed HW/SW data management. Hence, our framework enables applications to bind data with stable and uniform usage to a specific memory tier (using software guidance) with the ability to transparently move application data with bursty or non-uniform usage into and out of a large, hardware-managed DRAM cache.

III. Adopted Runtime Framework

Our new tools extend the Simplified Interface to Complex Memory (SICM) project, which is a unified software framework, developed as part of the DOE Exascale Computing Project [13], for automatically adapting applications to a variety of memory devices and configurations. Our work builds on a recent extension of SICM that enables applications to guide their own data management across heterogeneous memory tiers through the use of low-overhead profiling and compiler-assisted allocation site annotations [11]. In this section, we provide an overview of these existing tools.

A. Simplified Interface to Complex Memory

The U.S. Department of Energy (DOE) is working towards achieving new levels of scientific discovery through ever-increasingly powerful supercomputers [32], [33]. Short-term plans call for achieving exaFLOP performance by the year 2021. To make these computing environments viable, the DOE has initiated a large effort titled the Exascale Computing Project (ECP) [34], [35]. The project includes multiple thrust areas to deal with the hardware and software challenges of the most complex and high-performance supercomputers. For such systems, the DOE spends hundreds of millions of dollars to achieve the highest performance possible from available hardware.

The *Simplified Interface to Complex Memory* (SICM), one of the ECP projects, seeks to deliver a simple and unified interface to the emerging complex memory hierarchies on exascale nodes [13]. To achieve this goal, SICM is split into two separate interfaces: the low-level and the high-level, as shown in Figure 1. The high-level interface delivers an API that allows applications to allocate, migrate, and persist their data without detailed knowledge of the underlying memory hardware. To implement these operations efficiently, the high-level API invokes the low-level interface, which interacts directly with device-specific services in the OS.

B. Portable Application Guidance for Complex Memory Systems

The SICM project was recently extended with new tools and allocation strategies to implement a portable guidance-based approach for assigning program data to heterogeneous memory tiers [10], [11]. The approach, which they call MemBrain, automates the use of data-tier guidance by associating profiles of memory behavior (i.e., bandwidth and capacity) with program *allocation sites*. Each allocation site corresponds to the source code file name and line number of an instruction that allocates program data (e.g., `malloc` or `new`) and may optionally include part or all of the call path leading up to the instruction. A separate analysis pass converts the profiles into tier recommendations for each site prior to guided execution. Figure 2 presents an overview of this approach. A full description of SICM with the extended MemBrain toolset is available in [11].

IV. Combined Hardware and Software Data Management with SICM+MemBrain

This work extends the SICM+MemBrain framework described in the previous section with two new capabilities:

1) New experimental configurations for systems with combined HW/SW data management modes, and

2) A custom profiling tool for estimating the cache efficiency of data associated with each allocation site.

A. Experimental Configurations for Combined HW/SW Data Management

To evaluate the potential of mixed HW/SW data management, we deployed the SICM+MemBrain toolset on an Intel® Knights Landing platform booted into *hybrid* memory mode. In this configuration, the KNL reserves half of its (16 GB) high bandwidth (MCDRAM) tier for use as a software-managed address space, while the other half is used as a hardware-managed cache [12]. In this way, data allocated directly to MCDRAM by software does not interfere with data in the MCDRAM cache. Next, we updated the SICM+MemBrain toolset to consider the physical addresses corresponding to the software-managed memory as the upper (high-performance) tier. We then applied the MemBrain approach, described in Figure 2, to assign the data allocated at each allocation site into the hardware- or software-managed sets during each experimental run. For these experiments, we employ the greedy *hotset* approach [31] to select application data with the highest bandwidth per unit capacity to be allocated directly to the non-cached MCDRAM tier.

B. Custom Profiling for Estimating Efficiency of In-memory Caching

While our new experimental configurations are useful for evaluating existing guidance-based approaches on combined HW/SW data management platforms, they do not consider the cache efficiency of the data associated with each allocation site. For example, the hotset approach may assign allocation sites with a relatively large number of cold pages to the uncached tier because the site contains a small number of very hot pages. To estimate the cache efficiency of the data associated with each allocation site, we extended the memory usage profiling capabilities in SICM to measure the rate of usage on each page associated with each allocation site. Our offline profiling tool maintains a map of pages for each site, and maps each memory access sample to each page. The tool prints all of the accessed pages, along with their associated allocation sites, and number of memory access samples to a file on disk at the end of the run.

V. Evaluation

A. Experimental Setup

The profiling and evaluations for this work were collected on an Intel Knights Landing(KNL) platform equipped with an Intel Xeon Phi with 64 cores, 256 hyper-threads, running at 1.40GHz. The machine has 16GBs of MCDRAM and 96GBs of DDR4. The MCDRAM can operate in one of three different modes, as a memory-side direct mapped cache, a software-managed mode, or in a mixed mode exercising 8GB of the MCDRAM as a hardware-directed cache and the other 8GB accessible through software-managed mode. We installed Debian 9.12 with Linux Kernel version 5.2.

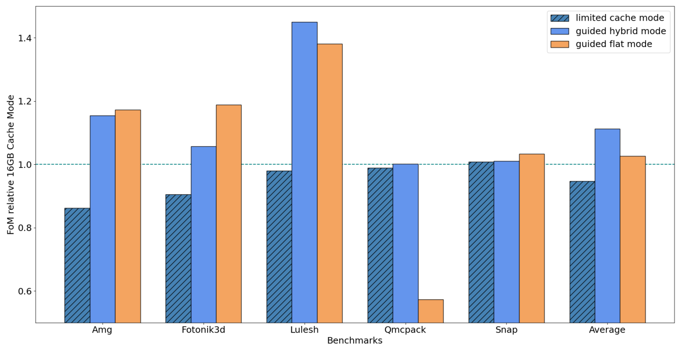


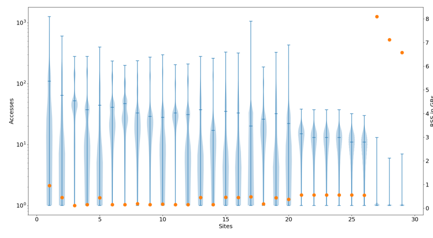
Fig. 3: Figure of merit for the limited (8 GB) cache mode, hybrid Mode, and flat mode shown relative to the full cache mode (16 GB) configuration (higher is better).

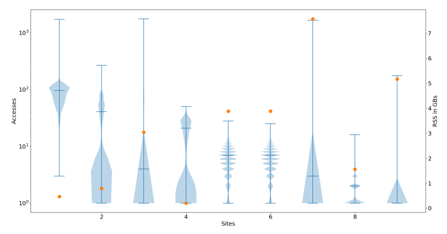
For this evaluation, we selected applications from the SPEC CPU 2017 [15] and CORAL2 [14] benchmark suites with data usage patterns which are known to stress memory bandwidth and processor cache performance. Specifically, we employ three proxy applications (LULESH, AMG, SNAP) and one full scale application (QMCPACK) from the CORAL2 benchmark, as well as the multi-threaded (6xx) version of the fotonik3d application from SPEC CPU 2017. All applications use OpenMP and are configured to use 256 software threads (one for each hardware thread on our KNL platform). For the CORAL2 benchmarks, we employ the 'large' input size from [11] which requires between 80 and 90 GB of capacity for each workload. To test fotonik3d with larger capacity requirements than the default *ref* input (which requires less than 16GB of memory capacity), we constructed a custom input file for a problem size of $N_x = 480$, $N_y = 1880$, and $N_z = 480$ and 812 time steps. Running fotonik3d with this input requires about 40GB of memory capacity with the default cache mode configuration on our KNL platform.

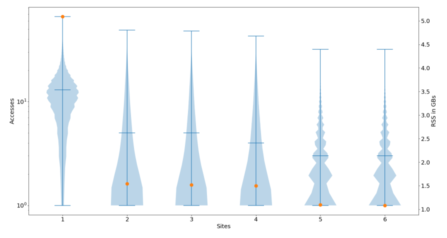
B. Performance Potential of Application Guidance with Combined HW/SW Data Management Modes

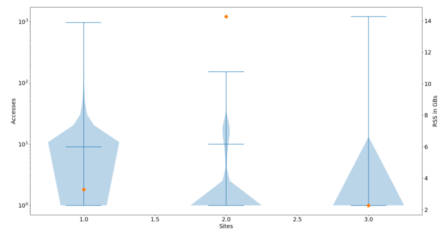
Our first set of experiments aims to evaluate the performance potential of existing application guidance-based approaches with combined HW/SW management modes. For these experiments, we ran each of the selected applications on the KNL platform in mixed HW/SW mode with the guidance configurations described in Section IV-A. For comparison, we ran each application with three additional configurations:

1) A full cache mode configuration, which uses the entire 16GB of MCDRAM as a hardware-directed cache,
2) A limited cache mode configuration, which only accesses MCDRAM through hardware-directed caching, but uses the hybrid mode option on the KNL to limit the MCDRAM cache size only 8GB of capacity, and
3) A fully software-directed flat mode configuration that uses the hotset approach to assign application data to the MCDRAM tier with the full 16GB capacity.


(a) LULESH


(b) AMG


(c) SNAP


(d) QMCPACK

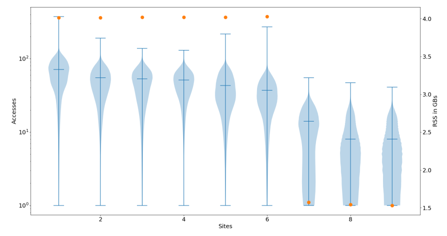

(e) fotonik3d

Fig. 4: Distribution of memory accesses and peak resident set size (RSS) of data allocated at distinct allocation sites. The violin plots, plotted on the left y-axis (log scale), show the sampled memory accesses per page for each site. The peak RSS of each site (in GB) is plotted as an orange marker using the right y-axis.

Figure 3 presents the figure of merit (FoM)(throughput) of each application with the limited cache mode, guided hybrid mode, and guided flat mode configurations relative to the full cache mode configuration. All results in this figure report the median FoM of three application runs (higher is better).

The results show that hybrid mode performs at least as well as the full cache mode in all cases, and is significantly faster than full cache mode for both LULESH and AMG. Additionally, while the fully guided flat mode performs as well or better than cache mode for four of the five benchmarks, it exhibits much worse performance for QMCPACK. Hence, hybrid mode achieves the best performance of the four configurations we tested, and outperforms the next closest configuration (i.e., guided flat mode) by more than 10%, on average.

C. *Identifying Cache Efficient Program Data*

Our next study aims to identify program allocation sites that are likely to generate cache efficient program data and distinguish these from sites that generate data with poor cache utilization. For this analysis, we employ the profiling tool described in Section IV-B to generate violin plots of the distribution of accesses to each page allocated at selected profiling allocation sites. Specifically, for each workload, we use kernel density estimation to show the distribution of accesses to pages within each site (e.g., a wider violin plot indicates more pages with similar access counts)

Figure 4 shows the violin plots for each workload. For presentation purposes, we display only those sites that allocate at least 16MB of data corresponding to at least 1% of sampled memory accesses. For each site, the violin plot is shown vertically in blue with a logarithmic scale of accesses on the left y-axis. The median and both extrema of each sites distribution are also indicated with a darker blue line. We also plot the peak RSS of that site, represented by an orange dot, in GBs along the right y-axis. Sites are sorted along the x-axis from hottest to coldest by accesses per byte.

The plots show that this collection of programs and allocation sites generate a wide range of capacity and usage distributions. Additionally, we find that the data generated by certain allocation sites are likely to utilize memory caches more efficiently than others. Consider sites 2 and 4 for the AMG workload shown in Figure 4b. These sites show a clear bimodal distribution and these would be potential candidate sites to be excluded from the software managed upper-tier and instead could be supported by the hardware memory-side cache since most of their pages have a low number of accesses with just a few pages contributing a large portion of that sites total accesses. On the other hand, site 1 is better suited for the software-managed flat mode of the KNL's hybrid mode since it has a low overall peak RSS and most of its pages have a uniform number of accesses across them. We can see sites exhibiting similar behavior across the other workloads. For instance, site 2 of QMCPACK (4d) shows a similar bimodal distribution as sites 2 and 4 from AMG. We also see a potential reason for fotonik3d (4e) performing best with the guided flat mode since most of its hottest sites have a uniform access distribution across their pages.

VI. Future Work

The development of profiling tools for estimating the cache efficiency of allocation sites for in-memory caches and the analysis of this data are important steps towards understanding how to leverage combined HW/SW data management modes. The next direct step to take would be to evaluate the performance of this model using the guidance data we have gathered. We can extend the hot allocation site identification we discussed in Section IV-A to identify hot sites that should be managed using the software-directed approach but we can exclude sites that we expect to have good in-memory cache performance based on access distribution to potentially improve overall workload throughput even more.

It would also be interesting to explore how these tools can be used on other memory platforms that support mixed HW/SW data management, such as Intel®'s Cascade Lake platform with conventional DRAM as an in-memory cache for non-volatile Optane DC memory devices.

VII. Conclusion

In this work we demonstrate that augmenting a combined HW/SW memory management strategy with application guidance obtains a 10% improvement on average over configurations that use hardware-directed caching or software-directed data placement alone. This work presented new extensions to the SICM+Membrain framework to allow the profiling of of data usage rates of physical pages associated with each allocation site. Lastly, this work presents additional analysis that shows the potential of this approach for guiding data placement on complex memory systems with mixed HW/SW data management modes.

Acknowledgements

We thank the anonymous reviewers for their thoughtful comments and feedback. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration, with additional contributions from the National Science Foundation under CNS-1943305.

References

[1] AMD, "High bandwidth memory (hbm) reinventing memory technology," https://www.amd.com/Documents/High-Bandwidth-Memory-HBM.pdf, 2016.

[2] R. Smith, "Nvidia volta unveiled: Gv100 gpu and tesla v100 accelerator announced," https://www.anandtech.com/show/11367/nvidia-volta-unveiled-gv100-gpu-and-tesla-v100-accelerator-announced, May 2017.

[3] A. Sodani, "Knights landing (knl): 2nd generation intel® xeon phi processor," in *Hot Chips 27 Symposium (HCS), 2015 IEEE*. IEEE, 2015, pp. 1–24.

[4] I. Cutress and B. Tallis, "Intel launches optane dimms up to 512gb: Apache pass is here!" https://www.anandtech.com/show/12828/intel-launches-optane-dimms-up-to-512gb-apache-pass-is-here, 2016.

[5] C. Cantalupo, V. Venkatesan, J. Hammond, K. Czurlyo, and S. D. Hammond, "memkind: An extensible heap memory manager for heterogeneous memory platforms and mixed memory policies." Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2015.

[6] NVIDIA, "Gp100 pascal whitepaper," https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf, 2016.

[7] H. Servat, A. J. Peña, G. Llort, E. Mercadal, H. Hoppe, and J. Labarta, "Automating the application data placement in hybrid memory systems," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2017.

[8] K. Wu, Y. Huang, and D. Li, "Unimem: Runtime data managementon non-volatile memory-based heterogeneous main memory," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. New York, NY, USA: ACM, 2017, pp. 58:1–58:14. [Online]. Available: http://doi.acm.org/10.1145/3126908.3126923

[9] M. Laghari, N. Ahmad, and D. Unat, "Phase-based data placement scheme for heterogeneous memory systems," in *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 2018, pp. 189–196.

[10] M. B. Olson, T. Zhou, M. R. Jantz, K. A. Doshi, M. G. Lopez, and O. Hernandez, "Membrain: Automated application guidance for hybrid memory systems," in *2018 IEEE International Conference on Networking, Architecture and Storage (NAS)*. IEEE, 2018, pp. 1–10.

[11] M. B. Olson, B. Kammerdiener, M. R. Jantz, K. A. Doshi, and T. Jones, "Portable application guidance for complex memory systems," in *Proceedings of the International Symposium on Memory Systems*, 2019, pp. 156–166.

[12] Intel, "Intel xeon phi x200 processor - memory modes and cluster modes: Configuration and use cases," https://software.intel.com/en-us/articles/intel-xeon-phi-x200-processor-memory-modes-and-cluster-modes-configuration-and-use-cases, December 2015.

[13] SICM, "Simplified Interface to Complex Memory - Exascale Computing Project 2019." https://www.exascaleproject.org/project/sicm-simplified-interface-complex-memory/, 2019.

[14] LLNL, "Coral benchmark codes," https://asc.llnl.gov/CORAL-benchmarks, June 2014.

[15] SPEC, "Spec cpu 2017," June 2017. [Online]. Available: https://www.spec.org/cpu2017/

[16] G. H. Loh and M. D. Hill, "Efficiently enabling conventional block sizes for very large die-stacked dram caches," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2011, pp. 454–464.

[17] J. Meza, J. Chang, H. Yoon, O. Mutlu, and P. Ranganathan, "Enabling efficient and scalable hybrid memories using fine-granularity dram cache management," *IEEE Computer Architecture Letters*, vol. 11, no. 2, pp. 61–64, 2012.

[18] Y. Lee, J. Kim, H. Jang, H. Yang, J. Kim, J. Jeong, and J. W. Lee, "A fully associative, tagless dram cache," in *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ser. ISCA '15. New York, NY, USA: ACM, 2015, pp. 211–222. [Online]. Available: http://doi.acm.org/10.1145/2749469.2750383

[19] H. Jang, Y. Lee, J. Kim, Y. Kim, J. Kim, J. Jeong, and J. W. Lee, "Efficient footprint caching for tagless dram caches," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, March 2016, pp. 237–248.

[20] V. Young, P. J. Nair, and M. K. Qureshi, "Dice: Compressing dram caches for bandwidth and capacity," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, pp. 627–638.

[21] 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 *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2014, pp. 1–12.

[22] J. Sim, A. R. Alameldeen, Z. Chishti, C. Wilkerson, and H. Kim, "Transparent hardware management of stacked dram as part of memory," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-47. Washington, DC, USA: IEEE Computer Society, 2014, pp. 13–24. [Online]. Available: http://dx.doi.org/10.1109/MICRO.2014.56

[23] S. Mittal and J. S. Vetter, "A survey of techniques for architecting dram caches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1852–1863, June 2016.

[24] M. Meswani, S. Blagodurov, D. Roberts, J. Slice, M. Ignatowski, and G. Loh, "Heterogeneous memory architectures: A hw/sw approach for mixing die-stacked and off-package memories," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*, Feb 2015, pp. 126–136.

[25] Y. Li, S. Ghose, J. Choi, J. Sun, H. Wang, and O. Mutlu, "Utility-based hybrid memory management," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2017.

[26] N. Agarwal and T. F. Wenisch, "Thermostat: Application-transparent page management for two-tiered main memory," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '17. New York, NY, USA: ACM, 2017, pp. 631–644. [Online]. Available: http://doi.acm.org/10.1145/3037697.3037706

[27] N. Agarwal, D. Nellans, M. Stephenson, M. O'Connor, and S. W. Keckler, "Page placement strategies for gpus within heterogeneous

memory systems," *SIGPLAN Not.*, vol. 50, no. 4, pp. 607–618, Mar. 2015. [Online]. Available: http://doi.acm.org/10.1145/2775054.2694381

[28] S. R. Dulloor, A. Roy, Z. Zhao, N. Sundaram, N. Satish, R. Sankaran, J. Jackson, and K. Schwan, "Data tiering in heterogeneous memory systems," in *Proceedings of the Eleventh European Conference on Computer Systems*. ACM, 2016, p. 15.

[29] I. B. Peng, R. Gioiosa, G. Kestor, P. Cicotti, E. Laure, and S. Markidis, "Rthms: A tool for data placement on hybrid memory system," in *Proceedings of the 2017 ACM SIGPLAN International Symposium on Memory Management*, ser. ISMM 2017. New York, NY, USA: ACM, 2017, pp. 82–91. [Online]. Available: http://doi.acm.org/10.1145/3092255.3092273

[30] S. Akram, J. B. Sartor, K. S. McKinley, and L. Eeckhout, "Write-rationing garbage collection for hybrid memories," in *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI 2018. New York, NY, USA: ACM, 2018, pp. 62–77. [Online]. Available: http://doi.acm.org/10.1145/3192366.3192392

[31] T. C. Effler, A. P. Howard, T. Zhou, M. R. Jantz, K. A. Doshi, and P. A. Kulkarni, "On automated feedback-driven data placement in multi-tiered memory," in *International Conference on Architecture of Computing Systems*. Springer, 2018, pp. 181–194.

[32] "U.s. department of energy and intel to deliver first exascale supercomputer." https://www.anl.gov/article/us-department-of-energy-and-intel-to-deliver-first-exascale-supercomputer, 2019.

[33] "U.s. department of energy and cray to deliver record-setting frontier supercomputer at ornl." https://www.ornl.gov/news/us-department-energy-and-cray-deliver-record-setting-frontier-supercomputer-ornl, 2019.

[34] B. Obama, "Executive order - creating a national strategic computing initiative." https://obamawhitehouse.archives.gov/the-press-office/2015/07/29/executive-order-creating-national-strategic-computing-initiative, 2015.

[35] D. Kothe, S. Lee, and I. Qualters, "Exascale computing in the united states," *Computing in Science & Engineering*, vol. 21, no. 1, pp. 17–29, 2018.