Toward Rethinking the Management Techniques in Emerging Memory Systems

Maryam Babaie[1], Ayaz Akram[2], and Jason Lowe-Power[3]
[1,2,3]*Computer Science Department, University of California-Davis, California, USA*
{[1]mbabaie, [2]yazakram, [3]jlowepower}@ucdavis.edu

***Abstract*— The increasing growth of applications' memory demands has led the CPU vendors to deploy diverse memory technologies either within the same package such as heterogenous memory systems, or in disaggregated form through local or remote memory nodes. As these new memory technologies emerge, the conventional memory managements should be reconsidered to better meet the applications memory requirements. However, there is not a suitable model available in the community to accurately study these new systems. In this work we describe our contribution toward a cycle-level analysis model of heterogenous memories in gem5 simulator. We believe this work enables the community to perform a design space exploration for the next generation of memory systems.**

1. Introduction

Today's computing systems must meet the large processing power and memory capacity demands of a broad range of applications including ML, AI, graph analytics, etc. To provide both high-performance and high-capacity memories to fulfill these applications' memory requirements, vendors have moved to heterogeneous memory systems where the processing units are provided with multiple memory technologies. Moreover, new interconnect technologies such as compute express link (CXL) are becoming mainstream to expand the capacity of local memories by attached devices. For instance, Intel's Sapphire Rapids will provide HBM, DDR, 3DXPoint, and CXL support within the same package. Another trend in this regard is disaggregated memory systems where they can provide multiple memory technologies in their local/remote memory nodes. As these new memory systems emerge, we need to rethink the memory management conventions. However, there is a lack of modeling support in the community to study these memory technologies. Thus, in this work we explain our contribution to high-fidelity heterogenous memory system modeling.

Systems like Intel's Knights Landing, Cascade Lake, and Sapphire Rapids can be configured to use high bandwidth memory as a cache to main memory. Anticipated disaggregation of memory resources also necessitates using local DRAMs as a cache to a large pool of remote memory. Thus, we developed a cycle-level DRAM cache controller model for gem5 [1]. The protocol is inspired by the actual hardware providing DRAM cache, such as Intel's Cascade Lake where DRAM cache is designed as a direct-mapped, insert-on-miss, and write-back cache. This model can be configured in two modes. First, as a *unified DRAM cache and memory controller*, where a DRAM cache and a main memory are connected through a shared bus. Second, as a *disaggregated DRAM cache controller* where a main memory is managed separately and connected to the DRAM cache through a port or a configurable link. Figure 1 shows these configurations. *Our controller model implements the timing and micro-architectural details enforced by the memory technologies employed as the cache and the backing store.* This enables a cycle-level full system analysis of the emerging memory systems, which is not possible by prior works.

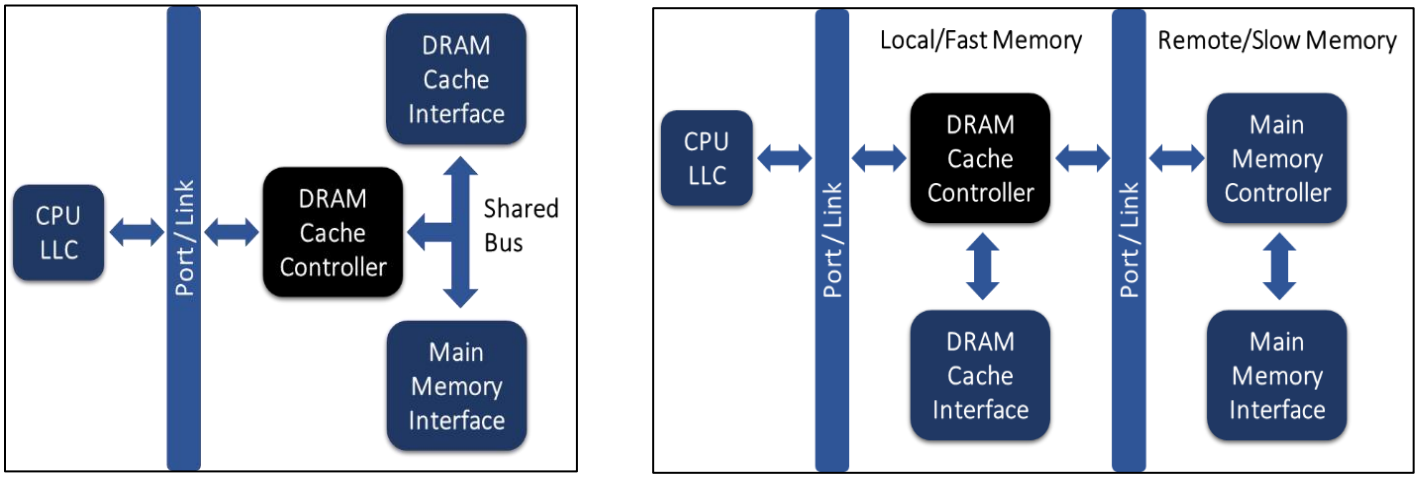

Fig. 1: Unified DRAM cache and main memory configuration (left), and disaggregated DRAM cache and main memory configuration (left).

2. Evaluation

We verified our DRAM cache model against the performance of Intel's Cascade Lake studied by Hildebrand et. al [2]. Here we show performance of the unified DRAM cache controller (UDCC) which aligns well with the Cascade Lake's architecture. First, we compare the effective bandwidth (observed by the last level cache (LLC)) for gem5's default memory controller (DMC) and UDCC. We used a synthetic read traffic pattern such that lead to 100% hit ratio. Figure 2 (top) compares these bandwidth numbers to the theoretical peak of DDR4 (act as DRAM cache) and shows the similarity in observed bandwidth (controllers' scheduling policy explains the slight differences). Moreover, we calculated the access amplification values for UDCC by dividing the effective bandwidth by the sum of the average bandwidth of DRAM cache and backing store devices for a particular run. The comparison of the two access amplification values is shown in Figure 2 (bottom). Our results match the actual hardware in all cases with one exception, write misses. In actual hardware, on a write miss in DRAM cache, the block is first allocated by reading it from backing store and then writing it into DRAM cache. The actual data is then written into the DRAM cache. We merge these two writes; thus, our model leads to one less access than the actual hardware for a write miss.
The disaggregated DRAM cache configuration also matched the expected results in our tests.

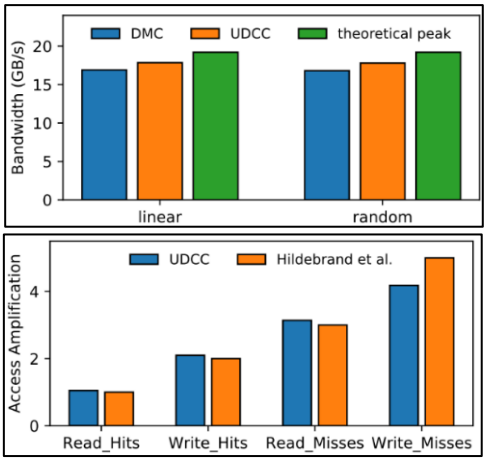

Fig. 2: Comparison of observed bandwidth by LLC in DMC vs. UDCC (top), and comparison of access amplification of our model vs Intel's Cascade Lake (bottom).

3. Case Studies

3.1 HPC Applications performance

In this case study, we performed an evaluation of DRAM caches for real-world applications, i.e., NPB [3] and GAPBS [4] in full system simulation for a system modeling Intel's Cascade Lake memory system. Due to limited space, here we only describe the NPB results. We ran all workloads in three different configurations. The first two configurations (NVRAM, DRAM) model a system without a DRAM cache and the main memory as NVRAM or DRAM. The third configuration (DCache_64MB) uses a 64MB DRAM cache and NVRAM as main memory. Figure 3 (top) shows a million instructions per second (MIPS) values for NPB in three configurations. In most cases, DCache_64MB performs the worst, with the most prominent performance degradation for lu.C and bt.C. The only exception is is.C, where DCache_64MB performs better than NVRAM. The performance of DCache_64MB correlates with the DRAM cache misses per thousand instructions (MPKI) values shown in Figure 3 (bottom). For example, is.C shows the smallest and lu.C shows the largest MPKI values.

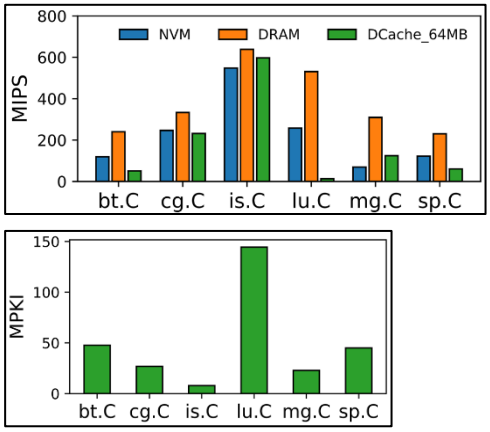

Fig. 3: Performance of NPB in full system simulation using our DRAM cache model.

3.2 Effect of Link Latency

In this case study we analyzed the performance of a DRAM cache while backed up by either DDR4, or NVRAM, configured in the disaggregated mode of our model. We applied different latencies to the link connecting the DRAM cache (local memory) to its backing store (remote memory). Injecting a synthetic read pattern with hit ratio of 0% (all miss-clean), we observed that sensitivity of NVRAM case to the link latency is lower than DDR4 case. Moreover, for the higher latencies the performance of the both cases become close. We believe the high response time that the DRAM cache shows on average, contributes to the tolerance of link latency. Table 1 summarizes our results for this study.

Table 1: Performance of DRAM cache controller while link latency changes.

Remote Mem.	Link Latency (µs)	Total BW (GB/s)	Avg. Resp.Time (µs)
DDR4	**No Link**	**6.31**	**1.484**
	0.2	5.86	1.599
	1	5.61	1.833
	2.5	5.20	2.985
	5	3.04	5.346
NVM	**No Link**	**6.03**	**2.489**
	0.2	6.03	2.487
	1	6.03	2.491
	2.5	4.86	3.269
	5	2.99	5.460

4. Conclusion

In this work we described our contribution to the heterogenous memory systems modeling. The model we described here, will be included in coming releases of gem5 for the community access. As the disaggregation and heterogeneity will play an important role in future memory systems, we believe having this model will enable performing researches that was not possible before.
We plan to include more results and studies for the presentation in the workshop.

5. References

[1] J. Lowe-Power et al., “The gem5 simulator: Version 20.0+,” 2020

[2] M. Hildebrand, J. T. Angeles, J. Lowe-Power, and V. Akella, “A case against hardware managed dram caches for nvram based systems,” in 2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE, 2021, pp. 194–204.

[3] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber et al., “The nas parallel benchmarks,” The International Journal of Supercomputing Applications, vol. 5, no. 3, pp. 63–73, 1991.

[4] S. Beamer, K. Asanovic, and D. Patterson, “The gap benchmark suite,” ´ arXiv preprint arXiv:1508.03619, 2015.