POSTER: Coordinated Memory Scheduling for CXL Memory Systems

Yongho Lee
Sungkyunkwan University
Gyeonggi-do, Suwon-si, Republic of Korea
jhyn205@skku.edu

Junbum Park
Sungkyunkwan University
Gyeonggi-do, Suwon-si, Republic of Korea
jbrara.park@skku.edu

Sungbin Jang
Sungkyunkwan University
Gyeonggi-do, Suwon-si, Republic of Korea
sunbi3361@skku.edu

Osang Kwon
Sungkyunkwan University
Gyeonggi-do, Suwon-si, Republic of Korea
osang915@skku.edu

Minkyu Choi
Sungkyunkwan University
Gyeonggi-do, Suwon-si, Republic of Korea
cmk3071@skku.edu

Seokin Hong
Sungkyunkwan University
Gyeonggi-do, Suwon-si, Republic of Korea
seokin@skku.edu

Abstract

CXL Memory Modules (CMMs) expand memory capacity but split scheduling across a host and an on-device DRAM controller. The host understands application-level objectives, while the CMM sees local queue and row-buffer state. Neither order preservation nor independent device reordering can satisfy both. **HARMONY** closes this semantic gap with priority-tagged requests and device-pressure feedback. It reduces p95 SLO violations by up to 20.9%, raises weighted speedup by 13.5%, and lowers maximum slowdown by 30.8%, approaching an ideal unified scheduler with 0.15% CMM-controller area overhead.

CCS Concepts

• **Software and its engineering → Memory management**.

Keywords

Compute Express Link (CXL), Disaggregated memory, CXL memory module, Memory scheduling

ACM Reference Format:
Yongho Lee, Junbum Park, Sungbin Jang, Osang Kwon, Minkyu Choi, and Seokin Hong. 2026. POSTER: Coordinated Memory Scheduling for CXL Memory Systems. In *International Conference on Parallel Architectures and Compilation Techniques (PACT '26), October 19-22, 2026, Chicago, IL, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3838684.3846849

1 Introduction

CXL [1] makes memory capacity scalable beyond host-attached DRAM (HAM), but a CMM contains its own memory controller [10, 20]. This creates two scheduling domains: the host knows request criticality and application fairness [4, 6], whereas the CMM knows its queue occupancy and DRAM row-buffer locality. Preserving host order leaves locality unrealized, while locality-aware CMM scheduling can invalidate the host policy that established the order.

This work is licensed under a Creative Commons Attribution 4.0 International License.

PACT '26, Chicago, IL, USA

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2915-7/2026/10
https://doi.org/10.1145/3838684.3846849

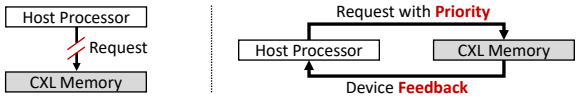


Figure 1: CXL interaction models. HARMONY augments conventional requests with host priority and CMM feedback.

We call this information asymmetry the *semantic gap*. Commercial CMM characterization reveals limited row-buffer reuse [10, 20], leaving substantial opportunity for locality-aware scheduling.

Neither host-order preservation nor independent device reordering jointly captures application-level priorities and device-local conditions. Effective scheduling therefore requires lightweight exchange of host intent and CMM state.

Conventional CXL requests carry no scheduling context. HARMONY instead makes the interface bidirectional: host priority travels with each request, while the CMM returns runtime pressure. This allows both sides to coordinate without centralizing control. Figure 1 summarizes this shift to cooperative scheduling.

2 HARMONY Overview

Figure 2 illustrates HARMONY. A host-side *Priority Estimator* (PE) maps host policy and device feedback to priority levels carried with each memory request. A device-side *Feedback Scheduler* (FS) performs priority-aware scheduling and returns a pressure signal. Together, they form a lightweight bidirectional control loop: the host conveys request importance, while the CMM schedules accordingly using local queue and DRAM state. Device feedback lets the host adapt subsequent priorities to CMM pressure, coordinating both scheduling domains without exposing internal state.

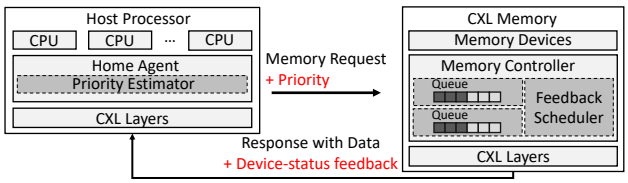


Figure 2: HARMONY framework with host and CMM.

3 Cooperative Scheduling Design

Priority Estimator. The PE sits in the host home agent and first applies a policy-based classification. For an SLO policy [6], it identifies requests from latency-sensitive applications; for a fairness policy [18], it prioritizes requests according to the host's fairness objective. The PE then adjusts the initial choice with the pressure reported by the CMM. Under light pressure, a latency-sensitive request may receive an Urgent tag, while persistent severe pressure progressively reduces the tag to drain congestion. This adaptation lets static system intent and dynamic CMM conditions jointly determine the priority of future requests.

Feedback Scheduler. The FS places incoming requests into four priority queues. It uses row-hit-aware weighted round-robin across the High, Medium, and Low queues, then compares the selected candidate with the Urgent queue using strict priority to protect latency-critical requests. When row-hit requests exist, WRR selects among the eligible queues and serves the oldest request; otherwise, it selects among non-empty queues. This structure preserves host-defined priorities while allowing the CMM to exploit local DRAM row-buffer locality.

CXL integration. HARMONY uses no new message type and no sideband traffic. The prototype carries a 2-bit priority hint in a reserved M2S field and uses the existing DevLoad field in the S2M response for device pressure [1]. This design enables policy-aware host-CMM coordination with only a minimal 2-bit extension to the existing CXL interface.

4 Evaluation Methodology

We evaluate HARMONY using an extended Ramulator 2.0 [11] model calibrated to real-hardware measurements of 181 ns HAM and 276 ns CMM access latency, with a 45 ns one-way CXL-link latency. Our primary configuration places the hottest 20% in HAM and the remaining 80% in CMM [9, 12].

We use SPEC CPU 2006/2017 [16, 17], DLRM [14], XSBench [21], YCSB server A [2], LLaMA 3.3 [5], and GenomicsBench [19]. Each core executes a 200M-instruction warm-up followed by a 1B-instruction measurement interval. We compare four scheduler organizations. *Baseline* uses FCFS at the host and FR-FCFS [15] at the CMM. *Separated* applies host policy while the CMM independently uses FR-FCFS. *Integrated* is an idealized unified scheduler with visibility into both host policy and device DRAM state. *HARMONY* coordinates the separated controllers through priority and feedback exchange.

For SLO evaluation, two high-priority cores contend with six best-effort cores, with p95/p99 targets set to four times uncontended latency. For fairness, eight applications share memory, and we report weighted speedup (WS; higher is better) [3] and maximum slowdown (MS; lower is better) [7, 8, 13].

Table 1: Overall results. p95 violation is normalized to Integrated; lower is better for p95 and MS.

Scheme	p95 violation (norm.) ↓	WS ↑	MS ↓
Baseline	1.273	5.19	3.21
Separated	1.257	5.32	3.11
HARMONY	**1.007**	**5.89**	**2.22**
Integrated	1.000	6.00	2.10

5 Results

SLO attainment. As summarized in Table 1, HARMONY achieves a normalized p95 violation ratio of 1.007, nearly matching the Integrated upper bound of 1.000. In contrast, Baseline and Separated remain at 1.273 and 1.257, respectively. Relative to Baseline, HARMONY reduces p95 SLO violations by 20.9%. HARMONY also improves the performance of high-priority cores by 4.9% on average. The small gap to Integrated shows that bidirectional priority and feedback exchange preserves host scheduling intent, whereas Separated shows that host-side classification alone is insufficient without CMM-visible priorities.

Fairness. HARMONY achieves a WS of 5.89 and an MS of 2.22, compared with 5.19 and 3.21 for Baseline. This corresponds to a 13.5% improvement in weighted speedup and a 30.8% reduction in maximum slowdown. Separated provides marginal improvement (WS 5.32, MS 3.11), whereas HARMONY approaches Integrated (WS 6.00, MS 2.10). The improvement in both metrics shows that priority information lets the CMM preserve host-level fairness objectives while exploiting local scheduling state. These results demonstrate that cooperation across the CXL boundary is necessary to preserve scheduling objectives established at the host.

Overhead. HARMONY adds approximately 10,000 gates at the host and 60,000 gates at the CMM, totaling roughly 70,000 gates. The device cost is about 0.15% of a typical CXL memory-controller logic budget, while the host cost is negligible relative to a server CPU. The benefit therefore comes from a small metadata channel and modest queueing logic, rather than centralized scheduling. This low implementation cost recovers most of the benefit of Integrated while retaining independent host and CMM schedulers.

6 Conclusion

Independent optimization is the wrong abstraction for CXL memory scheduling. HARMONY turns the host-CMM boundary into a lightweight control loop: the host communicates scheduling intent through request priorities, while the device reports local pressure. This cooperation preserves host-level scheduling objectives while allowing the CMM to exploit its local queue and DRAM state. Our results show that lightweight information exchange can closely approach unified scheduling without centralizing control or sacrificing CMM autonomy.

Acknowledgments

This work was partly supported by K-CHIPS (Korea Collaborative & High-tech Initiative for Prospective Semiconductor Research) (RS-2026-25521189, 25%) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea) and the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (RS-2026-25490403, 25%) and the IITP (Institute of Information & Communications Technology Planning & Evaluation)-ITRC (Information Technology Research Center) grant funded by the Korean government (MSIT) (IITP-2026-RS-2021-II212052, 30%) and the BK21 FOUR Project (20%). Yongho Lee, Sungbin Jang, Osang Kwon, Minkyu Choi, and Seokin Hong are affiliated with the Department of Electrical and Computer Engineering. Junbum Park is affiliated with the Department of Semiconductor and Display Engineering. Seokin Hong is the corresponding author.

References

[1] Compute Express Link Consortium 2023. *CXL 3.1 Specification*. Compute Express Link Consortium. https://computeexpresslink.org/cxl-specification/

[2] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking cloud serving systems with YCSB. In *Proceedings of the 1st ACM symposium on Cloud computing*. 143–154.

[3] Stijn Eyerman and Lieven Eeckhout. 2008. System-level performance metrics for multiprogram workloads. *IEEE Micro* 28, 3 (2008), 42–53.

[4] Saugata Ghose, Hyodong Lee, and José F Martínez. 2013. Improving memory scheduling via processor-side load criticality information. In *Proceedings of the 40th annual international symposium on computer architecture*. 84–95.

[5] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[6] Ravi Iyer, Li Zhao, Fei Guo, Ramesh Illikkal, Srihari Makineni, Don Newell, Yan Solihin, Lisa Hsu, and Steve Reinhardt. 2007. QoS policies and architecture for cache/memory in CMP platforms. *ACM SIGMETRICS Performance Evaluation Review* 35, 1 (2007), 25–36.

[7] Yoongu Kim, Dongsu Han, Onur Mutlu, and Mor Harchol-Balter. 2010. ATLAS: A scalable and high-performance scheduling algorithm for multiple memory controllers. In *HPCA-16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*. IEEE, 1–12.

[8] Yoongu Kim, Michael Papamichael, Onur Mutlu, and Mor Harchol-Balter. 2010. Thread cluster memory scheduling: Exploiting differences in memory access behavior. In *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 65–76.

[9] Huaicheng Li, Daniel S Berger, Lisa Hsu, Daniel Ernst, Pantea Zardoshti, Stanko Novakovic, Monish Shah, Samir Rajadnya, Scott Lee, Ishwar Agarwal, et al. 2023. Pond: CXL-based memory pooling systems for cloud platforms. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. 574–587.

[10] Jinshu Liu, Hamid Hadian, Yuyue Wang, Daniel S Berger, Marie Nguyen, Xun Jian, Sam H Noh, and Huaicheng Li. 2025. Systematic CXL memory characterization and performance analysis at scale. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. 1203–1217.

[11] Haocong Luo, Yahya Can Tuğrul, F. Nisa Bostancı, Ataberk Olgun, A. Giray Yağlıkçı, and Onur Mutlu. 2024. Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator. *IEEE Computer Architecture Letters* 23, 1 (2024), 112–116.

[12] Hasan Al Maruf, Hao Wang, Abhishek Dhanotia, Johannes Weiner, Niket Agarwal, Pallab Bhattacharya, Chris Petersen, Mosharaf Chowdhury, Shobhit Kanaujia, and Prakash Chauhan. 2023. TPP: Transparent page placement for CXL-enabled tiered-memory. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*. 742–755.

[13] Onur Mutlu and Thomas Moscibroda. 2007. Stall-time fair memory access scheduling for chip multiprocessors. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 146–160.

[14] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. 2019. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091* (2019).

[15] Scott Rixner, William J Dally, Ujval J Kapasi, Peter Mattson, and John D Owens. 2000. Memory access scheduling. In *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 128–138.

[16] Standard Performance Evaluation Corporation (SPEC) 2006. *SPEC CPU2006 Benchmark Suite*. Standard Performance Evaluation Corporation (SPEC). https://www.spec.org/cpu2006/ Accessed: 2025-02-25.

[17] Standard Performance Evaluation Corporation (SPEC) 2017. *SPEC CPU 2017 Benchmark Suite*. Standard Performance Evaluation Corporation (SPEC). https://www.spec.org/cpu2017/ Accessed: 2025-02-16.

[18] Lavanya Subramanian, Donghyuk Lee, Vivek Seshadri, Harsha Rastogi, and Onur Mutlu. 2016. BLISS: Balancing performance, fairness and complexity in memory access scheduling. *IEEE Transactions on Parallel and Distributed Systems* 27, 10 (2016), 3071–3087.

[19] Arun Subramaniyan, Yufeng Gu, Timothy Dunn, Somnath Paul, Md Vasimuddin, Sanchit Misra, David Blaauw, Satish Narayanasamy, and Reetuparna Das. 2021. Genomicsbench: A benchmark suite for genomics. In *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 1–12.

[20] Yan Sun, Yifan Yuan, Zeduo Yu, Reese Kuper, Chihun Song, Jinghan Huang, Houxiang Ji, Siddharth Agarwal, Jiaqi Lou, Ipoom Jeong, et al. 2023. Demystifying CXL memory with genuine cxl-ready systems and devices. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*. 105–121.

[21] John R Tramm, Andrew R Siegel, Tanzima Islam, and Martin Schulz. 2014. XS-Bench - The Development and Verification of a Performance Abstraction for Monte Carlo Reactor Analysis. In *PHYSOR 2014 - The Role of Reactor Physics toward a Sustainable Future*. Kyoto, Japan.