POSTER: Balancing Cost and Performance in Heterogeneous Memory Systems for LLM Inference

Soojin Hwang
sjhwang@etri.re.kr
ETRI
Daejeon, Republic of Korea

Jungwoo Kim
jungwkim@stanford.edu
Stanford University
California, United States

Sanghyeon Lee
leesh6796@casys.kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Hongbeen Kim
hbkim@casys.kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Jaehyuk Huh
jhhuh@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Abstract

Large language models (LLMs) require both high bandwidth for compute-intensive kernels and large capacity for model parameters and the dynamically growing KV cache. Asymmetric memory architectures offer a promising balance between cost and performance for LLM inference by combining bandwidth- and capacity-centric memory modules, but require efficient runtime mapping of computation and data across heterogeneous resources. We propose a lightweight runtime mapping algorithm based on an analytical performance model, considering the memory capacity and intensity of different kernels. The asymmetric memory system with the proposed mapping algorithm achieves substantial performance improvements over conventional memory systems.

CCS Concepts

• **Computer systems organization → Heterogeneous (hybrid) systems**; **Neural networks**.

Keywords

Large Language Models (LLMs), Heterogeneous memory

1 Memory Demands of LLM Inference

Large language models (LLMs) impose fundamentally conflicting demands on memory systems. They require high memory bandwidth to sustain compute-intensive kernels, while also demanding large capacity to store model parameters and the dynamically growing key–value (KV) cache. Conventional approaches typically optimize only one dimension, leading to either excessive cost and energy consumption or significant performance degradation. Multi-GPU systems with bandwidth-centric memory (e.g., HBM) address the bandwidth bottleneck but incur substantial cost and energy overhead. In contrast, a recent alternative HW-based solution integrates capacity-centric memory with a custom accelerator chip [5]. This approach is more energy- and cost-efficient but suffer from limited bandwidth, resulting in low performance. A natural alternative is hierarchical heterogeneous memory, but it remains ineffective for LLM workloads due to their large working sets and low temporal locality, which force frequent data migration between memory tiers and introduce substantial latency and bandwidth overheads.

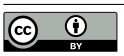
2 Asymmetric Memory

A promising approach to satisfy the two conflicting requirements of LLM inference, *large capacity* and *high bandwidth*, is a heterogeneous memory configured as an *asymmetric memory* architecture. This architecture combines a bandwidth-centric and a capacity-centric memory with computation unit attached for each memory in a single system. It avoids the limitations of the hierarchical memory by enabling direct computation on both memory types, pursuing the balance between performance and cost/energy efficiency.

Figure 1 presents a performance-energy efficiency pareto curve of the three memory configuration with various batch sizes (B) and sequence lengths (S): Capacity-centric memory (`LPDDR-only`), multiple bandwidth-centric memory (`8-HBM`), and an asymmetric memory (`Asymm-Oracle`), normalized to `B16-S512` configuration of LPDDR-only. In this view, the asymmetric memory emerges as a viable compromise between throughput (x-axis) and energy efficiency (y-axis). Rather than fully committing to bandwidth-centric memory, adding a small amount of HBM to a primarily LPDDR-based system can yield substantial gains where memory bandwidth is the bottleneck. Therefore, the asymmetric memory offers a promising direction for finding a sweet spot between performance and sustainable cost/energy budgets. However, to reach the optimal performance of asymmetric memory, **computation and data should be mapped to memory dynamically in runtime**.

3 Dynamic Mapping Algorithm

At runtime, the system must determine a near-optimal mapping of LLM kernels and data across bandwidth-centric and capacity-centric memory that adapts to evolving LLM workloads, balances heterogeneous resources, and minimizes migration under strict latency constraints. This problem is challenging for several reasons. First, the KV cache grows with each generated token and shrinks with the termination of each request in a batch, altering both memory footprint and system bottlenecks — making optimal mapping keep changing. Second, kernel characteristics are heterogeneous: Kernels of attention mechanism are bandwidth-bound, while other weight-activation computing layers exhibit different



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/10.1145/3838684.3846837

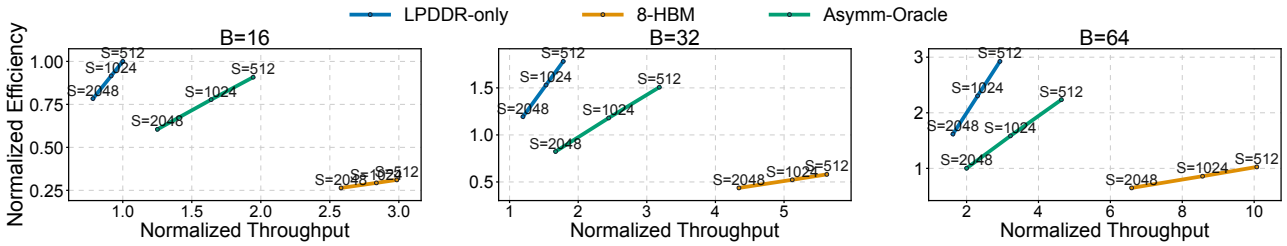


Figure 1: The performance and energy efficiency of memory configurations normalized to B16–S512 of LPDDR-only, showing that asymmetric memory achieves optimal balance.

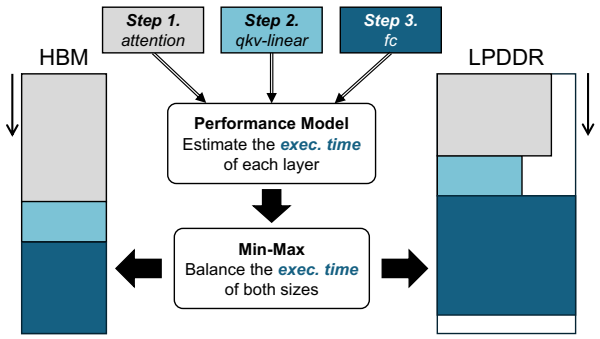


Figure 2: Overview of the dynamic mapping algorithm.

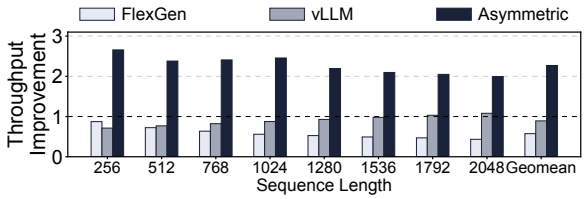


Figure 3: The throughput improvement over the baseline for GH200 GPU with FlexGen [6] and vLLM [3] in comparison with the asymmetric memory, for OPT-66B (FP16) with a batch size of 16.

compute–memory trade-offs, making coarse-grained mapping inefficient. Third, the mapping space is combinatorial at fine granularity, rendering exhaustive search impractical. Finally, dynamic remapping may incur non-trivial cost, as data migration over limited interconnect bandwidth can offset performance gains.

To address this challenges, we propose a lightweight runtime mapping decision algorithm that determines the mapping of kernels to memory devices at each decoding iteration. Rather than relying on expensive profiling or exhaustive search over a large combinatorial space, the algorithm uses a simple analytical performance model based on kernel footprint and arithmetic intensity to estimate execution time on each memory type. It then computes a near-optimal mapping that balances execution across devices while respecting capacity constraints, as visualized in Figure 2, with minimal changes of mapping between iterations. This approach enables fast, per-iteration adaptation to changes in KV cache size incurred by changes in batch composition and sequence length of each request. When mapping decisions change, only the necessary pages are migrated in the background, overlapping with computation and avoiding pipeline stalls. This design minimizes migration overhead while preserving the flexibility of dynamic mapping.

4 Results

We model the accelerator-based asymmetric memory system with a cycle-level performance simulator, referring the capacity and bandwidth of existing heterogeneous memory system (3TB/s HBM3 with 96GB capacity, 544GB/s LPDDR5 with 512GB capacity) [4], connected with a modern CXL interface (96GB/s bandwidth) [1]. We adopt the configuration of Google Cloud TPU for accelerator cores [2], adding more dot product units to saturate the memory bandwidth. We assign 8 cores for each chip, and employ the capacity-centric memory system in the prior work [5] as a baseline and a normalization point of the performance evaluation. Both baseline and asymmetric memory configurations consist of two accelerator chips with identical compute capability. In addition to the simulated baseline, we use FlexGen and vLLM on an NVIDIA GH200 single-GPU system [3, 4, 6]. We assume that all requests in a batch have the same sequence length, as one of the GPU baseline [6] only allows a scenario with a fixed sequence length in a batch.

Performance. Figure 3 shows the single-token generation throughput improvement of GH200 with FlexGen and vLLM in comparison with the asymmetric memory with the proposed mapping algorithm over the baseline, using OPT-66B (FP16) model with a fixed batch size of 16. The GPU system with FlexGen reports only 0.57× throughput improvement and vLLM shows only 0.89× throughput improvement compared to the baseline on average. On the other hand, the asymmetric memory system achieves 2.3× higher throughput over the baseline, 2.5× improvement over the single-GPU system with vLLM. The performance gap is attributed to the hierarchical memory model of GPU system, where data migration acts as a bottleneck.

Effectiveness of Mapping Algorithm. Compared to the ideal (i.e. oracle) mapping of asymmetric memory, the mapping policy adds 2.4% overheads for GPT3-175B (INT8) with the batch size of 16, and 6.6% overheads for Chinchilla-70B (INT8) with the batch size of 32 on average. These results validate the robustness of the proposed mapping algorithm with varying model sizes.

Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) funded by the Ministry of Science and ICT, Korea under Grant RS-2023-00216370.

References

[1] 2020. CXL 2.0 Specification. https://www.computeexpresslink.org/download-the-specification.

[2] Google. 2018. Cloud TPU. https://cloud.google.com/tpu/docs/system-architecture-tpu-vm.

[3] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP'23)*.

[4] NVIDIA. 2023. Grace Hopper Superchip. https://www.nvidia.com/en-us/data-center/grace-hopper-superchip/.

[5] Sang-Soo Park, KyungSoo Kim, Jinin So, Jin Jung, Jonggeon Lee, Kyoungwan Woo, Nayeon Kim, Younghyun Lee, Hyungyo Kim, Yongsuk Kwon, et al. 2024. An LPDDR-based CXL-PNM Platform for TCO-efficient Inference of Transformer-based Large Language Models. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 970–982.

[6] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*.