BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en

Vol. 31 No. 2. 2026

UDC 004.253
DOI: 10.62660/bcstu/2.2026.48

ISSN: 2306-4412
E-ISSN: 2708-6070

Rethinking memory hierarchy policies in DRAM+HBM systems

Maksym Shcherba*
Postgraduate Student
Ivan Franko National University of Lviv
79007, 1 Universytetska Str., Lviv, Ukraine
https://orcid.org/0009-0008-0332-9254

Petro Tarnavskyi
Postgraduate Student
Ivan Franko National University of Lviv
79007, 1 Universytetska Str., Lviv, Ukraine
https://orcid.org/0000-0003-3265-8168

Abstract. Efficient management of heterogeneous memory systems integrating high-bandwidth memory (HBM) is critical for overall performance. Conventional approaches often assume that dynamic random-access memory (DRAM) is the fastest tier, which may not hold true in HBM-equipped configurations. This study aimed to evaluate the performance impact of reconfiguring memory hierarchy policies by shifting the preference from DRAM to HBM in DRAM+HBM systems. The research employed a modified Ambix framework, adapting migration policies to designate HBM as the fast tier and tuning internal parameters to better suit the DRAM+HBM architecture. Performance was assessed using HPCG and AMG benchmarks on a dual-socket server. It was established that placing all data in HBM yielded performance gains of up to 24% over DRAM-only execution. The analysis demonstrated that standard Linux NUMA balancing could cause up to 4% performance degradation because it incorrectly promoted memory pages to the slower DRAM tier. Investigations revealed that while default Ambix configurations might reduce performance by up to 23%, favouring HBM improved results by 1.4% to 22% over DRAM-preferred policies. Furthermore, internal parameters were tuned to reduce page table scanning frequency and increase migration limits per cycle. These modifications achieved a 3-7% performance gain for the HPCG benchmark compared to the baseline system. For the AMG benchmark, experimental results showed variations from 1.5% degradation to 1.4% improvement, depending on the DRAM:HBM capacity ratios. These findings can enable system architects to optimise memory tiering in next-generation servers without requiring kernel or application modification

Keywords: heterogeneous memory systems; memory management; dynamic random-access memory; high bandwidth memory; memory hierarchy

INTRODUCTION

The ongoing evolution of high-performance computing is increasingly characterised by the integration of diverse memory technologies, such as high-bandwidth memory (HBM), alongside conventional dynamic random-access memory (DRAM). This development offers substantial potential for performance gains, particularly for data-intensive applications that require high throughput. However, a notable consideration in these environments is that many existing system management policies were originally designed for traditional architectures where DRAM was the undisputed fast tier. When these established mechanisms are applied to mixed DRAM and HBM configurations, they may not fully exploit the specific performance characteristics

Article's History: Received: 08.01.2025; Revised: 17.04.2026; Accepted: 18.05.2026; Published: 26.06.2026

Suggested Citation:
Shcherba, M., & Tarnavskyi, P. (2026). Rethinking memory hierarchy policies in DRAM+HBM systems. *Bulletin of Cherkasy State Technological University*, 31(2), 48-56. doi: 10.62660/bcstu/2.2026.48.

*Corresponding author (Maksym.Shcherba@lnu.edu.ua)

of the hardware, leading to instances of sub-optimal resource utilisation. Consequently, there is a clear incentive to investigate refined management strategies that can more effectively align data placement with the actual bandwidth and latency profiles of modern heterogeneous memory systems.

The emergence of cache-coherent interconnects such as Compute Express Link (CXL) has recently attracted significant attention due to their capabilities in hardware heterogeneity management and resource disaggregation. D. Gouk *et al.* (2022) proposed DirectCXL, a system that connects a host processor complex directly to remote memory resources over the CXL memory protocol. Their research established that by bypassing data copies between host and remote memory, it is possible to expose the true performance of disaggregated resources to the user. Whilst such technologies facilitate hardware expansion, they also demand more sophisticated monitoring frameworks to handle the resulting complexity. S. Park *et al.* (2022) addressed this by introducing DAOS, an open-source system for data access-aware memory management. By utilising practical trade-offs between monitoring overhead and accuracy, DAOS provides a runtime system that auto-tunes management schemes for user-defined objectives, demonstrating that fine-grained access monitoring is essential for efficient placement decisions.

As memory systems scale to multi-terabyte capacities, the challenges of profiling and migration become even more pronounced. J. Ren *et al.* (2024) developed MTM, a framework for multi-tiered large memory that connects profiling overhead control with high-quality monitoring mechanisms. Their work indicates that a universal page migration policy, combined with huge page awareness, can significantly improve performance across realistic big-data workloads. Building upon these migration efficiencies, L. Xiang *et al.* (2024) introduced Nomad, which challenges the traditional exclusive tiering model. By implementing non-exclusive tiering through transactional page migration and page shadowing, Nomad mitigates memory thrashing and removes data movement from the critical execution path, significantly improving performance under memory pressure compared to standard transparent placement strategies. Beyond general page-level management, some investigators have identified the importance of specific system structures. S. Kannan *et al.* (2021) noted that most existing systems overlook kernel-level structures. Through the development of the KLOCs mechanism, they established that tiering kernel-level objects, such as inodes and socket buffers, significantly improves the efficiency of heterogeneous systems.

Alternative approaches focus on the granularity of management and the timing of data placement. The problem of flexible object-level management was addressed by B. Kammerdiener *et al.* (2023), who developed the bkmalloc method. They concluded that leveraging object-level information at the programming language level allows for more accurate data migration decisions between tiers compared to transparent OS-level mechanisms. The shortcomings of such transparent strategies were further detailed by D. Moura *et al.* (2022), who demonstrated that the standard AutoNUMA balancer yielded modest results for graph analytics due to a lack of temporal locality in data access patterns. Their characterisation showed that an object-level tiering approach could reduce execution times by an average of 21% by more accurately capturing application-specific behaviours than standard kernel heuristics. Similarly, M. B. Olson *et al.* (2022) focused on developing an online application guidance system, concluding that dynamically assigning application data based on live usage patterns achieves comparable performance to offline profiling methods while significantly outperforming standard unguided approaches. These developments suggest a trend toward more autonomous and context-aware systems that reduce the burden on the programmer whilst improving hardware efficiency.

Despite this significant number of developments, a critical gap remains in the current literature. Most analysed solutions are based on the assumption that DRAM is the fast tier compared to slower tiers like non-volatile memory or CXL expansion devices. Aspects of system functioning where DRAM coexists with higher-speed HBM, as well as questions of migration parameter calibration for such configurations, remain insufficiently studied. Existing policies are often motivated by capacity-driven assumptions, whereas for DRAM+HBM systems, the optimal tier designation may differ significantly for certain workloads. Furthermore, online page placement and migration without the necessity for offline profiling, pre-runs, or application recompilation are discussed only to a limited extent. Therefore, this work aimed to experimentally evaluate online tiered-memory management in DRAM+HBM systems without the requirement for offline preconditions. By investigating these dynamics, the study sought to determine how software-defined hierarchy policies could be reconciled with the physical bandwidth advantages of HBM, whilst identifying the efficiency thresholds where active management might become counter-productive.

MATERIALS AND METHODS

A modified memory management strategy based on the Ambix framework (Marques, 2021; Póvoas *et al.*, 2025) was proposed, and its performance was evaluated experimentally. The selection of Ambix was justified by its implementation as a Linux kernel module, which enabled the deployment of new policies without requiring modifications to the operating system kernel or application source code. The approach involved reconfiguring the memory hierarchy to designate HBM as the primary fast tier, whilst DRAM served as the secondary tier for less frequently accessed data. The research was based on a hypothesis that directly contradicted the standard assumptions of the Linux AutoNUMA

mechanism, where DRAM was viewed by default as the fastest hierarchy tier-a premise typically valid for DRAM+NVM or DRAM+CXL systems. Shifting priorities in favour of HBM was driven by its significant bandwidth advantage, which was a critical performance factor for data-intensive high-performance workloads. To optimise system performance, the page table scanning frequency was reduced to lower monitoring overhead, while per-cycle migration limits were increased to facilitate the rapid promotion of active data to the HBM tier. The experimental environment comprised a dual-socket server equipped with an Intel Xeon CPU Max 9462 processor featuring 32 cores per socket. The platform was configured with 256 GB of DRAM and 64 GB of HBM. All benchmark executions were restricted to a single CPU socket. The software stack utilised Rocky Linux 9.5 with Linux kernel version 6.1.

Performance evaluation was conducted using the HPCG (Dongarra *et al.*, 2016) and AMG Problem 2 (Yang *et al.*, 2017) benchmarks. For each run, it was calculated the standard performance metrics reported by the benchmarks as the Figure of Merit (FoM): HPCG performance in GFLOP/s and AMG performance as its reported FoM, which represents the rate of useful sparse-matrix work (i.e., effective matrix operations per second). To verify that the selected test scenarios were indeed bandwidth-bounded, a comparative analysis of results was performed in configurations where the entire memory volume was allocated exclusively in DRAM or exclusively in HBM. These tools provided standardised metrics to analyse system behaviour under bandwidth-intensive conditions.

To investigate the impact of resource availability, a memory offlining mechanism was employed to create controlled DRAM:HBM capacity ratios of 2:1, 1:1, and 1:2. Memory offlining was performed using the Linux memory hotplug interface via sysfs, which allows selected memory blocks to be disabled to enforce the desired ratios. The resulting data were recorded across six distinct system configurations, enabling a comparison between the proposed solution and standard kernel mechanisms, specifically the AutoNUMA balancing system (Huang, 2019). This configuration ensured that the benchmarks occupied nearly all available memory within a single socket, providing a comprehensive understanding of how different policies influenced computational efficiency.

RESULTS AND DISCUSSION

Initial performance comparisons demonstrated that allocating the entire memory volume to HBM yielded substantial throughput improvements over configurations utilising only DRAM. As shown in Figure 1, data placement within HBM provided a 24% performance gain for the HPCG benchmark and a 22% gain for the AMG benchmark. The observed performance gap between DRAM-only and HBM-only configurations validated the bandwidth-intensive nature of the workloads.

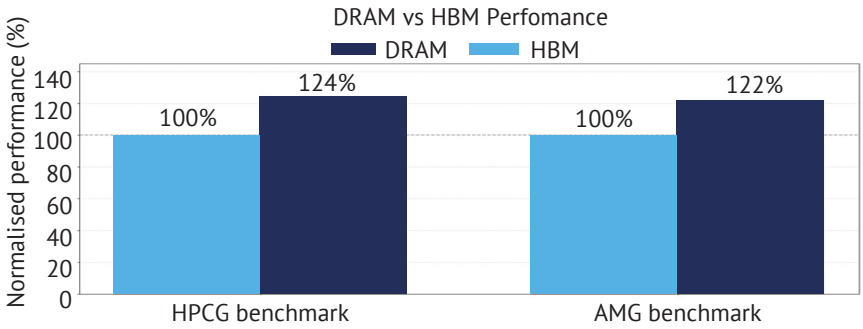

Figure 1. Performance comparison of DRAM and HBM memory across benchmarks
Source: created by the authors

Assessments across varying DRAM:HBM capacity ratios indicated that standard operating system mechanisms could lead to sub-optimal resource utilisation. For the HPCG workload, the *No balancing* baseline yielded performance between 33.62 and 34.93 GFLOP/s, as summarised in Table 1. The activation of NUMA balancing resulted in a performance degradation of up to 4% relative to the baseline – a decrease attributed to the misalignment of page promotion directions, where the kernel incorrectly identified the slower DRAM tier as the fast tier and promoted pages accordingly.

Table 1. HPCG benchmark performance (GFLOP/s) across different DRAM:HBM capacity ratios and memory management policies

Configuration	Performance (GFLOP/s)		
	DRAM:HBM = 2:1	DRAM:HBM = 1:1	DRAM:HBM = 1:2
No balancing	33.62	33.85	34.93
NUMA balancing	32.44	32.40	33.97
Ambix (DRAM-priority)	29.06	29.63	30.87

Continued Table 1.

Configuration	Performance (GFLOP/s)		
	DRAM:HBM = 2:1	DRAM:HBM = 1:1	DRAM:HBM = 1:2
Ambix (HBM-priority)	31.14	30.05	31.42
Ambix (DRAM-priority, tuned)	29.42	31.46	34.19
Ambix (HBM-priority, tuned)	35.86	35.38	36.05

Source: created by the authors

Performance metrics for the HPCG benchmark, illustrated in Figure 2, revealed significant variations based on the management policy employed. The original Ambix (DRAM-priority) configuration resulted in a performance decline of 12-14% relative to the baseline. It was observed that implementing individual modifications to this framework yielded incremental improvements, though neither was sufficient to outperform the baseline independently. Reconfiguring the hierarchy to prioritise HBM – Ambix (HBM-priority) – partially mitigated the loss, reducing degradation to 7-11%. Similarly, maintaining the DRAM-priority while optimising internal framework logic – Ambix (DRAM-priority, tuned) – reduced the performance penalty to 2-12%.

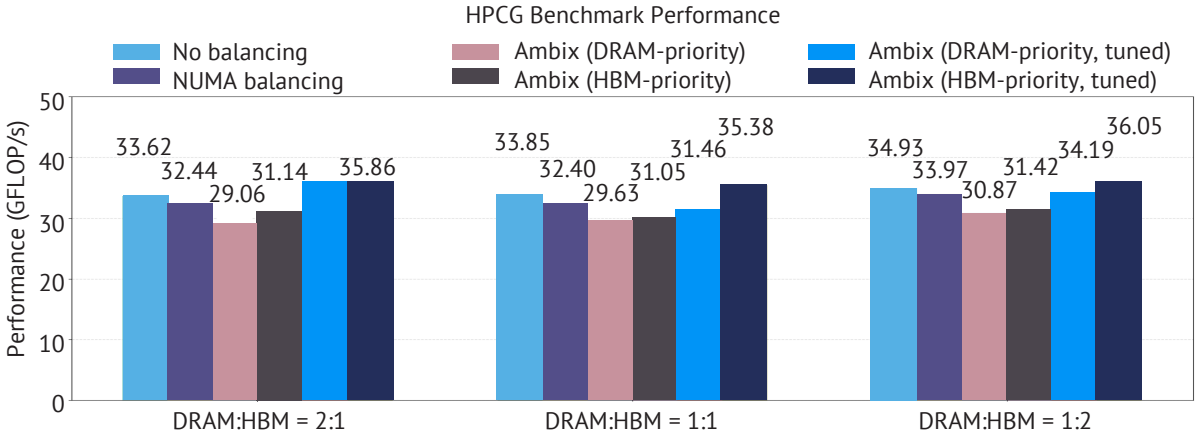

Figure 2. Comparison of HPCG benchmark performance (GFLOP/s) across evaluated memory management policies

Source: created by the authors

Net performance gains were only realised when both modifications were applied simultaneously. Under the *Ambix (HBM-priority, tuned)* framework, which combined an HBM-fast hierarchy with optimised scan frequencies and migration limits, performance improvements of 7%, 5%, and 3% were achieved for the 2:1, 1:1, and 1:2 DRAM:HBM ratios, respectively. These findings indicated that while each structural change provided a marginal benefit, the synergy between a hardware-aligned hierarchy and tuned migration parameters was essential to surpass the performance of the baseline.

Furthermore, the results suggested that the effectiveness of proactive memory placement diminished as the availability of HBM increased. At lower HBM capacities, efficient data placement was critical to maximise throughput, whereas higher HBM availability allowed a larger portion of the hot data to reside in the fast tier without intervention, thereby diminishing the relative impact of managed migration. Performance metrics for the AMG benchmark were summarised in Table 2, where the baseline provided a FoM ranging from $1.25 \cdot 10^9$ to $1.36 \cdot 10^9$. In contrast to the results observed for HPCG, the activation of *NUMA balancing* yielded performance identical to the baseline, indicating that the default kernel mechanism neither improved nor degraded throughput for this specific workload. However, the broader results for managed configurations demonstrated a trend consistent with the overhead typically associated with active memory management in tiered systems.

Table 2. AMG benchmark performance (FoM) across different DRAM:HBM capacity ratios and memory management policies

Configuration	Figure of Merit		
	DRAM:HBM = 2:1	DRAM:HBM = 1:1	DRAM:HBM = 1:2
No balancing	$1.25 \cdot 10^9$	$1.26 \cdot 10^9$	$1.36 \cdot 10^9$
NUMA balancing	$1.25 \cdot 10^9$	$1.26 \cdot 10^9$	$1.36 \cdot 10^9$
Ambix (DRAM-priority)	$0.99 \cdot 10^9$	$1.00 \cdot 10^9$	$1.04 \cdot 10^9$

Continued Table 2.

Configuration	Figure of Merit		
	DRAM:HBM = 2:1	DRAM:HBM = 1:1	DRAM:HBM = 1:2
Ambix (HBM-priority)	$1.06 \cdot 10^9$	$1.07 \cdot 10^9$	$1.10 \cdot 10^9$
Ambix (DRAM-priority, tuned)	$1.19 \cdot 10^9$	$1.21 \cdot 10^9$	$1.29 \cdot 10^9$
Ambix (HBM-priority, tuned)	$1.27 \cdot 10^9$	$1.27 \cdot 10^9$	$1.34 \cdot 10^9$

Notes: the Figure of Merit is the performance metric reported by the AMG benchmark
Source: created by the authors

The introduction of managed placement via the Ambix (DRAM-priority) policy resulted in a significant performance reduction of 21-23% relative to the baseline. Incremental improvements were observed when modifications were applied individually to the framework; these trends were visualised for comparative analysis in Figure 3 to highlight the relative impact of each change. Prioritising the high-bandwidth tier in the Ambix (HBM-priority) configuration reduced the performance penalty to 15-19% compared to the baseline. Similarly, maintaining the original hierarchy while applying parameter optimisations in the Ambix (DRAM-priority, tuned) configuration further limited the degradation to 4-5%.

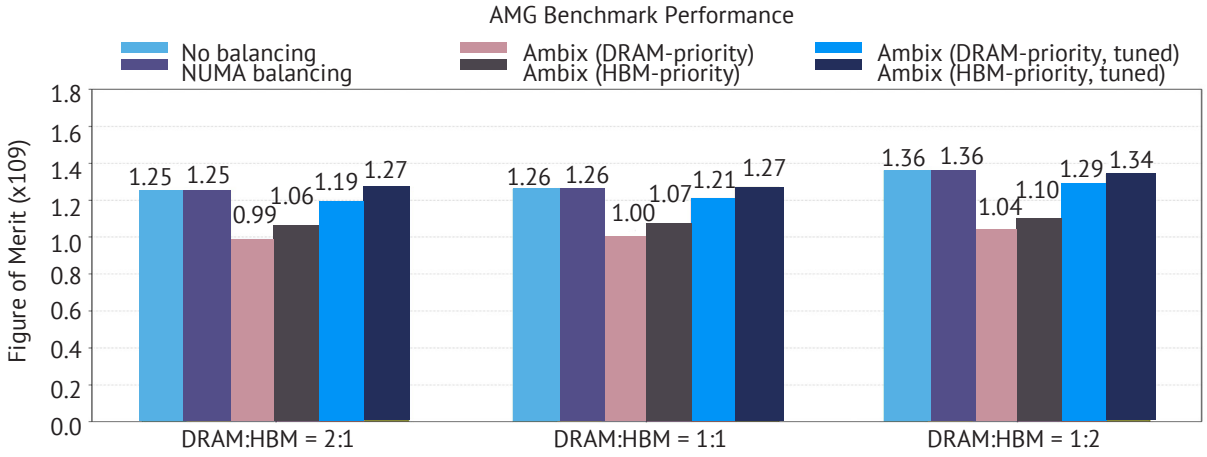

Figure 3. Comparison of AMG benchmark performance (Figure of Merit) across evaluated memory management policies
Notes: the Figure of Merit is the performance metric reported by the AMG benchmark
Source: created by the authors

As illustrated in Figure 3, the highest efficiency across all evaluated configurations was achieved through the simultaneous application of hierarchy reconfiguration and parameter tuning. Under the Ambix (HBM-priority, tuned) policy, the system showed a 1.4% improvement over the baseline at a 2:1 DRAM:HBM capacity ratio, although a 1.5% degradation was experienced at the 1:2 ratio. This slight decline at higher HBM capacities was attributed to the variable nature of the AMG memory footprint. At the onset of the workload, the active dataset could fit entirely within the HBM tier. In such instances, the computational overhead of the management framework outweighed the benefits of page migration, rendering active management counter-productive when HBM resources were sufficiently abundant to accommodate the workload without intervention. In summary, the evaluation across both benchmarks underscored that effective tiered memory management in DRAM+HBM systems required a synergistic approach. While individual modifications – such as aligning the memory hierarchy with hardware bandwidth in Ambix (HBM-priority) or optimising migration logic in Ambix (DRAM-priority, tuned) – yielded incremental benefits, neither was sufficient to consistently surpass the No balancing baseline. Net performance improvements were only realised through the unified Ambix (HBM-priority, tuned) policy, which reconciled the software-defined hierarchy with the physical memory topology. Furthermore, the results highlighted a critical efficiency threshold: as the availability of HBM increased relative to the workload footprint, the computational overhead of the management framework began to outweigh the benefits of proactive migration. This phenomenon was particularly evident in the AMG results, suggesting that the utility of managed memory placement was most pronounced in resource-constrained scenarios where the high-bandwidth tier could not accommodate the entire active dataset without intervention.

The observed performance dynamics in DRAM+HBM systems indicated a fundamental departure from the capacity-driven paradigms that dominated tiered memory research over the last decade. While conventional

heterogeneous memory management had focused on expanding DRAM capacity through slower tiers like CXL, the integration of HBM inverted the performance hierarchy. In this bandwidth-driven context, the secondary tier was significantly faster than the primary capacity tier. The recorded 4% performance degradation under the NUMA balancing configuration underscored the failure of legacy locality heuristics in such environments. As highlighted by H. Al Maruf *et al.* (2023), the Linux kernel was originally designed for homogeneous architectures and traditionally prioritised memory nodes based on relative NUMA distances, which fail to reflect the actual bandwidth capabilities of heterogeneous tiers. Consequently, if a thread was perceived as local to the DRAM node, standard mechanisms inadvertently promoted pages to the slower tier, a finding that aligned with the research of J. Kim *et al.* (2021) regarding multi-tiered systems. This study confirmed that tier-agnostic policies, which prioritised proximity over throughput, were fundamentally ill-equipped for asymmetric bandwidth hierarchies.

The limitations of such heuristics extended beyond standard kernel mechanisms, manifesting even more severely in specialised management frameworks, as evidenced by the significant performance reduction of 21-23% observed in the Ambix (DRAM-priority) configuration. This provided empirical evidence that software assumptions must align with hardware realities, yet merely reconfiguring the hierarchy to prioritise the high-bandwidth tier (Ambix (HBM-priority)) proved insufficient, as it only reduced the performance penalty to 15-19%. This residual gap suggests that a correct hierarchy is only the first requirement; the second essential pillar is the precise calibration of management logic. Similar conclusions were reported by M. Kim *et al.* (2025) for tiered-memory management under co-located workloads, where correct tier preference must be complemented by adaptive runtime control. The sensitivity of tiered systems to such logic was validated by the work of T.D. Doudali *et al.* (2021), whose Cori framework demonstrated that sub-optimal migration frequencies could lead to a 10-100% performance degradation. These findings explain why surpassing the baseline in this study required a synergy between an HBM-priority hierarchy and carefully tuned migration limits. As J.D. McCalpin (2023) established, sustained bandwidth in Xeon Max processors is already constrained by hardware limiters such as insufficient memory concurrency, making the system exceptionally vulnerable to software overheads that arise from uncalibrated management policies. The most pervasive of these overheads originated from the methodological challenges associated with kernel-level monitoring remained a primary friction point. A. Raybuck *et al.* (2021) discovered in their work on HeMem that page table scanning incurred prohibitive costs due to TLB shootdowns and lock contention. While this study utilised a kernel-integrated approach, the performance gains achieved by reducing scanning frequency in the Ambix (HBM-priority, tuned) configuration served as a necessary concession to these inherent costs. This strategy was consistent with the research of T. Lee *et al.* (2023), who established in their work on MEMTIS that fixed-rate scanning mechanisms faced a steep overhead-accuracy trade-off, with CPU costs reaching 72.85% to maintain high monitoring precision. They demonstrated that an effective management layer required dynamic page classification via multiple access thresholds, which enabled a significant reduction in sampling intensity for colder memory regions to mitigate system-wide overhead. Such optimisations were crucial, as the sensitivity of the HBM-DRAM gap made management overhead even more critical than in latency-heavy NVM systems. Recent work such as FlexMem (Xu *et al.*, 2024) supports this point: by adapting profiling and demotion decisions, it avoids scanning with fixed ratio and reduces unnecessary page migrations in tiered memory.

The tangible impact of these overheads was most clearly manifested in the identification of an efficiency threshold, illustrated by the AMG benchmark results. The transition from a 1.4% improvement at a 2:1 ratio to a 1.5% degradation at a 1:2 ratio suggested that active memory management possessed a point of diminishing returns. When HBM resources were sufficiently abundant to accommodate the active dataset at the onset of a workload, the computational overhead associated with page table scanning outweighed any potential performance benefits. Ultimately, the comparative analysis suggested that while industry efforts were largely focused on masking the slowness of expansion tiers, HBM required a paradigm that exploited primary tier bandwidth while minimising management interference. As explored by P. Duraisamy *et al.* (2023) in the TMTS framework, hardware-guided tiering often scaled better than software-only scanning. This study validated that an HBM-priority policy acted as a necessary proxy for bandwidth-priority, forcing data into the wider pipe even when latency signals were ambiguous. The synergy between hierarchy prioritisation and parameter tuning was therefore a fundamental requirement for unlocking the performance potential of next-generation heterogeneous memory systems.

CONCLUSIONS

This study demonstrated that effective utilisation of heterogeneous memory systems, particularly for DRAM+HBM configurations, required a well-defined and carefully considered memory hierarchy. The primary outcome of this research was the confirmation that reconsidering the memory hierarchy, by treating HBM as a top-tier memory in a multi-level system, could significantly enhance performance for bandwidth-dependent workloads. This finding challenged the default assumptions embedded in most modern operating

systems, where DRAM was typically regarded as the fastest memory and thus prioritised for page promotion. The proposed modification, based on the Ambix system, achieved improved efficiency through two key mechanisms: preferential use of HBM and fine-tuning of internal page migration parameters. Experimental evaluations using standard benchmarks revealed consistent performance improvements across most tested configurations. The most substantial gains were observed in scenarios with limited HBM capacity, where intelligent data placement became critical.

In contrast, when HBM capacity was sufficient to store the full active data used in a benchmark, the benefits of controlled page migration lessened or might even have become negative due to overhead costs. This highlighted the importance of adaptive approaches that considered memory utilisation and workload characteristics. In systems with limited DRAM capacity where HBM was added as a supplementary but also constrained memory tier, the proposed approach significantly improved performance by optimising data placement and memory bandwidth usage. Overall, the results indicated that an appropriate memory policy could improve system efficiency without requiring modifications to application code or operating system kernel-an advantage that facilitated deployment especially in large-scale systems. Future work could focus on evaluating and adapting alternative heterogeneous-memory managers to clarify the practical differences between HBM- and DRAM-prioritisation strategies. In addition, a promising research direction is the design and implementation of a workload-aware runtime memory manager that dynamically determines the most suitable memory tier for each page and adapts monitoring intensity based on lightweight indicators such as bandwidth demand, CPU load, and working-set residency across all memory tiers.

ACKNOWLEDGEMENTS

We are grateful to Jakub Schmiegel for his valuable technical assistance and to Łukasz Kucharski for insightful comments that significantly improved the manuscript. We also thank both for their constructive discussions regarding the results.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Dongarra, J., Heroux, M.A., & Luszczek, P. (2016). High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems. *The International Journal of High Performance Computing Applications*, 30(1), 3-10. doi: 10.1177/10943420155.

[2] Doudali, T.D., Zahka, D., & Gavrilovska, A. (2021). Cori: Dancing to the right beat of periodic data movements over hybrid memory systems. *2021 IEEE international parallel and distributed processing symposium (IPDPS)* (pp. 350-359). Piscataway: IEEE. doi: 10.1109/IPDPS49936.2021.00043.

[3] Duraisamy, P., *et al.* (2023). Towards an adaptable systems architecture for memory tiering at warehouse-scale. In *Proceedings of the 28th ACM international conference on architectural support for programming languages and operating systems* (pp. 727-741). New York: Association for Computing Machinery. doi: 10.1145/3582016.3582031.

[4] Gouk, D., Lee, S., Kwon, M., & Jung, M. (2022). Direct access, high-performance memory disaggregation with DirectCXL. In *2022 USENIX annual technical conference (USENIX ATC 22)* (pp. 287-294). Carlsbad: USENIX Association.

[5] Al Maruf, H., Wang, H., Dhanotia, A., Weiner, J., Agarwal, N., Bhattacharya, P., Petersen, C., Chowdhury, M., Kanaujia, S., & Chauhan, P. (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* (pp. 742-755). New York: Association for Computing Machinery. doi: 10.1145/3582016.3582063.

[6] Huang, Y. (2019). *Autonuma: Optimize memory placement in memory tiering system*. Retrieved from https://lkml.org/lkml/2021/2/4/289.

[7] Kammerdiener, B., McMichael, J.Z., Jantz, M.R., Doshi, K.A., & Jones, T. (2023). Flexible and effective object tiering for heterogeneous memory systems. In *Proceedings of the 2023 ACM SIGPLAN international symposium on memory management* (pp. 163-175). New York: Association for Computing Machinery. doi: 10.1145/3591195.3595277.

[8] Kannan, S., Ren, Y., & Bhattacharjee, A. (2021). KLOCs: Kernel-level object contexts for heterogeneous memory systems. In *Proceedings of the 26th ACM international conference on architectural support for programming languages and operating systems* (pp. 65-78). New York: Association for Computing Machinery. doi: 10.1145/3445814.3446745.

[9] Kim M., Han, S., Park, G., & Kim, D. (2025). MTAT: Adaptive fast memory management for co-located latency-critical workloads in tiered memory system. In *Proceedings of the 26th international middleware conference* (pp. 86-98). New York: Association for Computing Machinery. doi: 10.1145/3721462.3770767.

[10] Kim, J., Choe, W., & Ahn, J. (2021). Exploring the design space of page management for multi-tiered memory systems. In *2021 USENIX annual technical conference (USENIX ATC 21)* (pp. 715-728). Berkeley: USENIX Association.

[11] Lee, T., Monga, S., Min, C., & Eom, Y.I. (2023). MEMTIS: Efficient memory tiering with dynamic page classification and page size determination. In *Proceedings of the 29th ACM symposium on operating systems principles* (pp. 17-34). doi: 10.1145/3600006.3613167.

[12] Marques, M.S. (2021). *Ambix: Rethinking Linux's page management to support the new Intel Optane DC persistent memory*. (Master's thesis, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal).

[13] McCalpin, J.D. (2023). Bandwidth limits in the Intel Xeon Max (sapphire rapids with HBM) processors. In *High performance computing: ISC high performance 2023 international workshops* (pp. 403-413). Berlin: Springer-Verlag. doi: 10.1007/978-3-031-40843-4_30.

[14] Moura, D., Mosse, D., & Petrucci, V. (2022). Performance characterization of AutoNUMA memory tiering on graph analytics. In *2022 IEEE international symposium on workload characterization (IISWC)* (pp. 171-184). IEEE. doi: 10.1109/iiswc55918.2022.00024.

[15] Olson, M.B., Kammerdiener, B., Jantz, M.R., Doshi, K.A., & Jones, T. (2022). Online application guidance for heterogeneous memory systems. *ACM Transactions on Architecture and Code Optimization*, 19(3), article number 45. doi: 10.1145/3533855.

[16] Park, S., Bhowmik, M., & Uta, A. (2022). DAOS: Data access-aware operating system. In *Proceedings of the 31st international symposium on high-performance parallel and distributed computing* (pp. 4-15). New York: Association for Computing Machinery. doi: 10.1145/3502181.3531466.

[17] Póvoas, J., Barreto, J., Chomiski, B., Gonçalves, A., Karabeinikau, F., Maciejewski, M., Schmiegel, J., & Storozhuk, K. (2025). Better memory tiering, right from the first placement. In *Proceedings of the 16th ACM/SPEC international conference on performance engineering* (pp. 253-265). New York: Association for Computing Machinery. doi: 10.1145/3676151.3719378.

[18] Raybuck, A., Stamler, T., Zhang, W., Erez, M., & Peter, S. (2021). HeMem: Scalable tiered memory management for big data applications and real NVM. In *Proceedings of the ACM SIGOPS 28th symposium on operating systems principles* (pp. 392-407). New York: Association for Computing Machinery. doi: 10.1145/3477132.3483550.

[19] Ren, J., Xu, D., Ryu, J., Shin, K., Kim, D., & Li, D. (2024). MTM: Rethinking memory profiling and migration for multi-tiered large memory. In *Proceedings of the nineteenth european conference on computer systems* (pp. 803-817). New York: Association for Computing Machinery. doi: 10.1145/3627703.3650075.

[20] Xiang, L., Lin, Z., Deng, W., Lu, H., Rao, J., Yuan, Y., & Wang, R. (2024). Nomad: Non-exclusive memory tiering via transactional page migration. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)* (pp. 19-35). Santa Clara: USENIX Association.

[21] Xu, D., Ryu, J., Shin, K., Su, P., & Li, D. (2024). FlexMem: Adaptive page profiling and migration for tiered memory. In *Proceedings of the 2024 USENIX annual technical conference (USENIX ATC 24)* (pp. 817-833). Santa Clara: USENIX Association.

[22] Yang, U., Falgout, R., & Park, J. (2017). *Algebraic multigrid benchmark*. Retrieved from https://www.osti.gov/biblio/1389816.

Переосмислення політик ієрархії пам’яті в системах DRAM+HBM

Максим Щерба
Аспірант
Львівський національний університет імені Івана Франка
79007, вул. Університетська, 1, м. Львів, Україна
https://orcid.org/0009-0008-0332-9254
Петро Тарнавський
Аспірант
Львівський національний університет імені Івана Франка
79007, вул. Університетська, 1, м. Львів, Україна
https://orcid.org/0000-0003-3265-8168

Анотація. Ефективне управління різнорідними системами пам’яті, що містять пам’ять із високою пропускною здатністю (HBM), є критично важливим для загальної продуктивності. Традиційні підходи часто припускають, що динамічна пам’ять з довільним доступом (DRAM) є найшвидшим рівнем, що може не справджуватися в конфігураціях з HBM. Метою цього дослідження було оцінювання впливу на продуктивність переналаштування політики ієрархії пам’яті шляхом зміни переваги з DRAM на HBM у системах DRAM + HBM. У дослідженні використано модифіковану систему Ambix, у якій адаптовано політики міграції для призначення HBM швидким рівнем та налаштовано внутрішні параметри для кращої відповідності архітектурі DRAM + HBM. Оцінювання проводилося за допомогою тестів HPCG та AMG на двопроцесорному сервері. Встановлено, що розміщення всіх даних у пам’яті HBM забезпечило приріст продуктивності до 24 % порівняно з використанням лише DRAM. Аналіз продемонстрував, що стандартне балансування NUMA в Linux могло спричинити деградацію продуктивності до 4 %, оскільки воно некоректно переміщувало сторінки пам’яті до повільнішого рівня DRAM. Дослідження виявило, що хоча стандартні конфігурації Ambix могли знижувати продуктивність до 23 %, надання переваги HBM покращило результати на величину від 1,4 % до 22 % порівняно з політиками, орієнтованими на DRAM. Крім того, налаштовано внутрішні параметри, що дозволило зменшити частоту сканування таблиць сторінок та збільшити ліміти міграції за цикл. Ці модифікації забезпечили приріст продуктивності від 3 % до 7 % для тесту HPCG порівняно з базовою системою. Для тесту AMG експериментальні результати показали варіювання від деградації на 1,5 % до покращення на 1,4 %, залежно від співвідношення обсягів DRAM:HBM. Отримані результати дозволяють системним архітекторам оптимізувати ієрархію пам’яті в серверах наступного покоління без необхідності внесення змін до ядра ОС або коду програм

Ключові слова: різнорідні системи пам’яті; управління пам’яттю; динамічна пам’ять з довільним доступом; пам’ять з високою пропускною здатністю; ієрархія пам’яті