AGDM: An Adaptive Granularity Data Migration Strategy for Hybrid Memory Systems

Zhouxuan Peng, Dan Feng*, Jianxi Chen*, Jing Hu, Chuang Huang
Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System of MoE,
School of Computer Science & Technology, Huazhong University of Science & Technology, Wuhan, China
Email:{hustpzx, dfeng, chenjx, hujingreal, huangchuang}@hust.edu.cn

***Abstract*—Hybrid memory systems show strong potential to satisfy the growing memory demands of modern applications by combining different memory technologies. Due to the different performance characteristics of hybrid memories, a data migration strategy that migrates hot data to a faster memory is critical to the overall performance. Prior works have focused on identifying hot data and migration decisions. However, we find that the fix-sized global migration granularity in existing data migration schemes results in suboptimal performance on most workloads. The key observation is that the optimal migration granularity varies with access patterns. This paper proposes AGDM, an access-pattern-aware Adaptive Granularity Data Migration strategy for hybrid memory systems. AGDM tracks memory access patterns in runtime and accordingly adopts the most appropriate migration mode and granularity. The novel remapping-migration decoupled metadata organization enables AGDM to set local optimal granularities for memory regions with different access patterns. Our evaluation shows that, compared to the state-of-the-art scheme, AGDM gets an average performance improvement of 20.06% with 29.98% energy savings.**

I. INTRODUCTION

Nowadays, building hybrid memory systems with different memory technologies has become mainstream to meet the ever-increasing memory requirements for modern applications. For example, combining 3D-Stacked DRAM [1, 2] and off-chip DRAM provides higher bandwidth, or combining DRAM and Non-Volatile Memory (e.g., PCM [3], 3D-XPoint [4]) provides larger capacity. In general, a hybrid memory system consists of two types of memory technologies: one has the limited capacity with better performance (higher bandwidth or lower latency), and the other has larger capacity but relatively poor performance. In order not to be limited by specific memory technology in the following discussion, we call the former Near Memory (NM) and the latter Far Memory (FM) as in previous works [5, 6].

The hybrid memory is preferred to be configured in a parallel architecture for larger available capacity and higher aggregate bandwidth. In this architecture, both NM and FM are treated as main memory to build a flat address space. However, using both memories equally results in performance degradation due to the performance flaws of FM and its large share of capacity. Therefore, recent works propose various data migration strategies that dynamically identify and migrate hot pages into NM to improve the overall performance. While how to identify hot data and when to trigger migration are widely studied, another important dimension is rarely explored–How much to migrate?

* Corresponding author: Dan Feng, Jianxi Chen

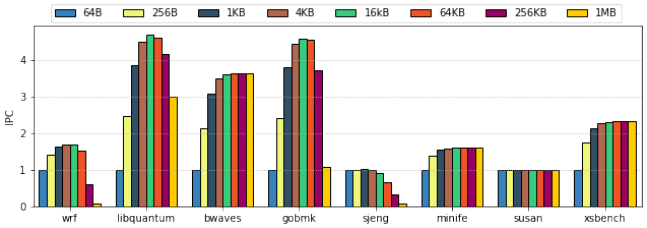

Fig. 1. The migration granularity sensitivity tests on POM

The amount of data migrated in a single migration, called migration granularity, strongly impacts performance. For example, a large migration granularity performs better on workloads with streaming behaviors since it benefits from prefetching effect and amortizing migration overhead. However, large migration granularity results in unnecessary migration traffic and performance degradation when applied to workloads with random and fine-grained access behavior. Therefore, it is important to choose an appropriate migration granularity. A common solution in previous works is to set a default granularity that achieves the best geometric mean performance for all workloads [6, 7]. However, a detailed migration granularity sensitivity experiment on POM [7] shows that this simplification often leads to suboptimal performance on most workloads. As shown in Figure 1, we can draw three conclusions from the results. First, the default 2KB migration granularity of POM is not optimal on most workloads. Second, migration granularity strongly impacts performance. For *libquantum*, the speedup at 16KB migration granularity is 4.69x that of 64B. Third, the optimal migration granularity that achieves the best overall performance varies from workload to workload.

A naive improvement is to set granularity individually for each workload, which requires plenty of sensitivity analysis. However, the optimal granularity varies with the memory access patterns. A real-world workload usually has multiple access patterns and exhibits different behaviors at different running stages. Furthermore, memory regions may exhibit different access patterns due to concurrent access from various applications on a real server. Therefore, a fixed global migration granularity is almost impossible to achieve optimal performance for workloads.

An advanced approach should dynamically set migration granularity for memory regions according to the current access

978-3-9819263-7-8/DATE23/© 2023 EDAA

pattern. To implement the access-pattern-aware adaptive migration granularity, there are two main challenges: i) Monitor access patterns of memory regions in runtime, ii) Set local migration granularity for memory regions with different access patterns. As a solution, we propose AGDM, an access-pattern-aware Adaptive Granularity Data Migration strategy for hybrid memory systems. The contributions of this work are the following:

- We experimentally explore the impact of migration granularity on performance and find that fixed global migration granularity in existing schemes results in suboptimal performance.
- We propose AGDM, to the best of our knowledge, the first access-pattern-aware adaptive granularity scheme for hybrid memory systems. AGDM tackles the two challenges with: i) a simple yet efficient runtime access pattern monitor, and ii) a two-level metadata organization that decouples migration from remapping.
- We propose three migration modes to deal with different access patterns. The migration granularity of AGDM can be continuous or discontinuous to maximize the prefetching effect.
- Our evaluation shows that AGDM improves performance by 20.06% and saves energy by 29.98% on average, compared to the state-of-the-art scheme.

II. CHALLENGES OF AGDM

Memory access patterns are challenging to quantify and rapidly change with the application running. Moreover, memory regions may exhibit different access patterns, requiring different migration granularities. Therefore, the complexity of access patterns brings the following challenges to AGDM.

Challenge 1: How to monitor access pattern? A naive method to understand the access pattern of a workload is offline profiling. However, real-world workloads usually have multiple access patterns, exhibiting different behaviors at different running stages. Offline profiling is costly and inefficient when dealing with variable access patterns. Another common method is application hints. Programmers can mark access patterns by specific instructions during programming. However, the access pattern hints of individual applications may be useless since hundreds of services and applications run on a real server simultaneously. The access patterns visible on memory may be inconsistent with application hints due to plenty of concurrent access.

Challenge 2: How to set migration granularity individually for memory regions with different access patterns? In existing schemes, the migration granularity is a global and one-for-all-workloads parameter. Moreover, the migration granularity is often bound to the remapping structure, which records the remappings of migrated pages and provides address translation. In this case, a migration granularity adjustment will require rebuilding the entire remapping table. Even if it is possible to add an additional field to the remapping table for migration granularity, the global migration granularity can not satisfy all regions with different access patterns. To support the memory-region-customized migration granularity, region-level access pattern tracking and novel metadata organization are required.

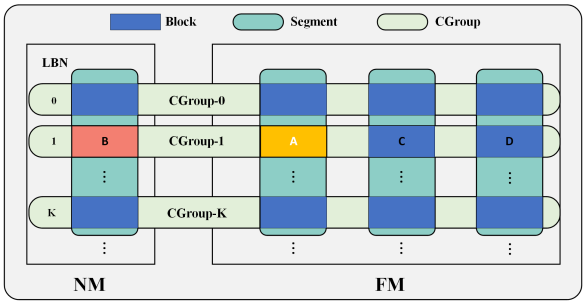

Fig. 2. An example of decoupled two-level organization

III. AGDM DESIGN

This work aims to provide an access-pattern-aware adaptive granularity data migration strategy for hybrid memory systems. To overcome challenges brought by the complexity of access patterns, we provide AGDM, which can set migration granularity individually for memory regions with different access patterns. AGDM consists of three components: i) remapping-migration decoupled two-level metadata organization, ii) runtime access pattern monitor(RAPM), and iii) migration decision logic. The novel metadata organization enables memory-region-customized access pattern tracking and migration granularity adjustment. Below, we will introduce each component in detail.

A. Remapping-Migration Decoupled Metadata Organization

AGDM decouples migration from the remapping structure to support variable and memory-region-customized migration granularity. Specifically, AGDM manages hybrid memory at a basic block-level remapping granularity and migrates data within a larger segment-level memory region. Based on the experience gained from the sensitivity analysis of POM [7], AGDM sets the block and segment size to 256B and 16KB, respectively. An example of the decoupled two-level organization is shown in Figure 2.

Block-level remapping A block is the basic granularity at which AGDM manages the hybrid memory. Theoretically, a hot (frequently accessed) block of FM can be migrated to any position of NM. However, maintaining such a full-associative remapping structure requires a huge metadata space. Therefore, AGDM adopts the congruence group (CGroup) concept from previous works [5, 8]. As shown in Figure 2, a CGroup contains one NM block and several FM blocks. All blocks within a CGroup have the same logical block number(LBN) offset with respect to the NM size. If a FM block becomes hot, it will be migrated to the NM block. In this case, the CGroup follows a fixed-remapping manner. However, increasing NM blocks of a CGroup can support remappings of any associativity. An advantage of fixed-remapping is that it requires less metadata than others, while AGDM requires extra metadata for migration.

Segment-level migration AGDM abolishes the global migration granularity since memory regions may have different access patterns. Instead, AGDM divides the memory logical address space into segments. Each segment consists of a fixed

number of logically continuous blocks. As shown in Figure 2, the CGroups and segments are orthogonal in memory logical space. Therefore, adjusting the migration granularity of a segment does not require changing the remapping structure (e.g., changing the block size). AGDM tracks access patterns at the segment level and sets an appropriate migration granularity for each segment. The migration granularity of a segment can be any integer multiple of a block. Moreover, the decoupled metadata organization enables discontinuous block migration within a segment, which allows AGDM to support flexible migration modes.

Although migration granularity is variable at the segment level, the migration itself is triggered at the block level. All blocks within a CGroup fairly compete for the sole NM block. The block with the most accesses will be the winner, called *token block*. The token block changes over time due to variations in access patterns. AGDM sets a token counter for electing the token block. If the token block is accessed, the token counter is increased by 1. Otherwise, the token counter is decreased by 1. When the token counter drops to 0, the current token block is replaced by the last block which decreases the token counter. The token counter ensures that the token block has more access than other blocks. Migration is triggered when a new token block is generated. The token-based migration trigger policy has two advantages: i) It does not rely on thresholds to determine whether a block should be migrated like in previous works [7, 9]. Instead, the token block is elected through competition between blocks. ii) It avoids blindly migrating blocks that may be rarely accessed, compared to cache-like migration policies [6, 8].

B. Runtime Access Pattern Monitor

As discussed in Section II, it is not easy to define and quantify a specific access pattern rigorously. However, the goal of AGDM is to guide migration granularity adjustment based on access patterns. Therefore, the problem can be simplified to finding useful access patterns for guiding granularity adjustment. There are two intuitive rules: i) Large migration granularity would benefit from streaming and/or scoped access patterns due to the prefetching effect, ii) Small granularity would be suitable for random and/or discrete access patterns since it avoids unnecessary traffic and long tail latency. The *runtime access pattern monitor (RAPM)* simply classify the access behavior of a segment into two types: *scoped* and *discrete*. The access pattern type is determined according to the aggregation degree of the accessed blocks. For an opening segment (being accessed), RAPM uses a 64-bit bitmap to record which blocks are accessed. Then, the number of accessed blocks ($Access_N$) and the access distance ($Access_D$, the maximum LBN absolute value of accessed blocks) are extracted from the bitmap. RAPM uses the quotient of $Access_N$ divided by $Access_D$ to represent the aggregation degree. If the aggregation degree exceeds a threshold, the segment is under scoped access pattern. Otherwise, the segment is under a discrete access pattern. The threshold is empirically set as 0.8. Considering concurrent access behaviors in hybrid memory, RAPM maintains a bitmap of the opening segment for each memory channel. The current opening segment is closed and the bitmap will be reset when another segment in the same channel is accessed.

C. Migration Granularity Decision

Corresponding to the access pattern classification of RAPM, AGDM enables two unique migration modes: i) *fixed-length adaptive mode*, and ii) *spatial footprint prediction mode*. The former deals with scoped access patterns, and the latter is designed for potentially recurring discrete access patterns.

Fixed-length adaptive mode: A large migration granularity that covers all possibly accessed blocks maximizes prefetching benefits for a scoped access pattern. Therefore, AGDM sets a fixed-length and continuous-space migration granularity for segments with scoped access patterns. Specifically, if the $Access_N$ is greater than $Access_D$, it indicates that the segment has a streaming access behavior. Thus, AGDM sets $Access_N$ as the migration granularity. Otherwise, if the $Access_N$ is less than $Access_D$, but the aggregation degree exceeds the threshold, all accessed blocks within the segment are tightly clustered in a range. In this case, AGDM sets $Access_D$ as the migration granularity.

Spatial footprint prediction mode: In general, a small migration granularity is better for discrete access patterns. However, small granularity results in more software overhead, and it loses the prefetching benefits. Therefore, AGDM enables a spatial footprint prediction (SFP) mode for potentially recurring discrete access patterns. The mechanism of SFP is that the two parameters, PC and request address, have a high correlation with the access patterns [10]. If a pair of PC and request address leads to an access pattern, then it is likely that a similar access pattern will repeat when the same PC and request address occurs. When a segment is accessed, RAPM records accessed blocks in a bitmap (also called spatial footprint). Then AGDM stores the bitmap into a Spatial Footprint History Table (SFHT). The XOR of the first accessed block address and the instruction PC is used as the index of SFHT. AGDM tries to find a corresponding spatial footprint in SFHT for a segment with the scoped access pattern. If a matching entry is found, AGDM migrates blocks accordingly. Otherwise, the migration logic will switch to *transparent* mode and migrates at block level like in previous works. The default migration mode of any segment is *transparent*.

D. Metadata Optimization

The flexible migration granularity choices come at the price of more metadata overhead. AGDM has three metadata components: block-level remapping table, segment information cache, and SFHT. We adopt optimizations on each component to shrink the metadata space and reduce metadata access overhead.

Remapping table: AGDM adopts fixed-remapping in the CGroup so that each remapping entry only requires two fields: tokenTag(5 bits) and tokenCntr(3 bits). In the default configuration(capacity ratio of NM:FM=1:8), the tokenTag can mark the original position of the token block. The 5-bits tokenTag supports FM capacity expansion up to 31x the NM capacity, which is enough to verify the capacity scalability of AGDM.

The 3-bits saturation token counter can quickly respond to the changes in the token block. Although each CGroup only requires 1B for remapping, the remapping table is too large to fit on-chip. However, putting the remapping table in NM incurs extra memory access for each request since the remapping table is located in the critical path. Therefore, AGDM enables a small on-chip remapping buffer to cache a few recently used remapping entries. The remapping buffer fetches cacheline-sized (64B) remapping entries at once and adopts the LRU replacement policy. The fetched metadata cacheline contains precisely all the remapping metadata of a segment, which helps non-transparent-mode migration.

Segment information cache: The migration mode, granularity, and spatial footprint of a segment are useful for migration. However, AGDM does not keep them for all segments due to the timeliness of access patterns and the considerable metadata overhead. Instead, AGDM only stores migration mode and granularity of recently accessed segments into an on-chip hardware structure, segment information cache (segCache). The segCache is a 4-way set-associative LRU cache. When the current opening segment is closed (triggered by an access to another segment), AGDM derives the migration mode and granularity of the segment from the recorded spatial footprint. Then if the segment already exists in segCache, the corresponding entry is updated. Otherwise, a new entry of the segment is added based on LRU. Each segCache entry is only 8B in size.

Spatial footprint history table: The SFHT records the spatial footprints of accessed segments. Each SFHT entry contains an 8B bitmap and a 4B index. To accelerate the querying, SFHT is a hash table whose hash function is to take the index modulo the capacity of SFHT. When a collision occurs, SFHT replaces the old entry with the new one. AGDM keeps 10K entries for SFHT and stores it in NM. The random replacement policy is adopted when SFHT is full. When the spatial pattern of a segment has only very few discrete burst accesses (e.g., $Access_D < 5$), it will not be recorded into SFHT.

Overall, the metadata memory space required by AGDM is only about 1/256 of the NM capacity. The 120KB of SFHT is negligible relative to the GB-level memory capacity. According to sensitivity tests in Section IV-D, the remapping buffer and segCache are set to 1KB in size, considering both hit ratios and hardware costs.

E. Put It Together

The workflow of AGDM is shown in Figure 3. The hardware structures required by AGDM are marked in blue, and the logic processes are marked in yellow. All structures on the upper layer are integrated into the on-chip hardware-based hybrid memory controller (HMC), which is responsible for address translation, data migration, and physical-logical isolation. The access path and migration logic of AGDM are as follows:

When HMC receives a memory request, ① Look up the remapping buffer. If the corresponding remapping entry is not found, ② Fetch a new metadata cacheline from the remapping table. The evicted remapping entries are flushed back to NM. The actual data address is known after obtaining the remapping entry. Then, ③ Check if the token block has changed. If the

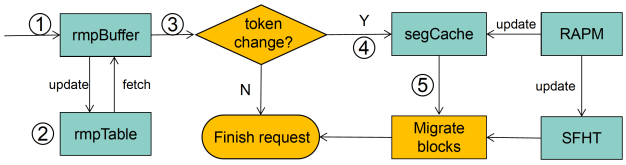

Fig. 3. The workflow of AGDM

token does not change, HMC forwards the request to the correct address and finishes it. Otherwise, the request triggers migration and activates the migration logic. ④ Look up the segCache. If the accessed segment already exists in segCache, ⑤ HMC performs migration according to the segCache entry. The SFHT is queried for a matching entry if it is in SFP mode. If there is no matching entry in segCache or SFHT, HMC performs migration at block granularity in transparent mode. Besides, the RAPM works independently of the migration logic. RAPM updates the spatial footprint of the opening segment at each request. When the current opening segment is closed, RAPM updates segCache and SFHT, respectively.

IV. EVALUATION

A. Methodology

Simulator. We simulate AGDM in the full-system simulator GEM5 [11]. We adopt the same memory configuration and parameters as HYBRID2 [6], which builds the hybrid memory system with 3D-Stacked DRAM and DDR4 DRAM. Table I summarizes the simulated configuration. The energy evaluation model is from [12].

Workloads. We simulated 20 workloads from four widely used benchmark suites: SPEC2006 [13], Mantevo [14], Mibench [15], and CORAL [16], whose characteristics are presented in Table II. We simulated 200 million instructions per application and four applications in a workload. Our workloads run in a multi-programmed simulation mode, where each application instance is run on a single core with warmed-up caches.

Configurations. We compare AGDM to a baseline configuration without NM and four state-of-the-art hardware-based schemes.

- POM [7]: Adopts 2KB migration granularity and a threshold-based migration policy that periodically adjusts the global threshold by sampling different areas.

TABLE I
SIMULATION CONFIGURATION

Cores	4 @3.5GHz(each), X86 ISA, out-of-order
L1(I/D)	32KB, 4-way associative, 64B cacheline
L2	256KB(private), 8-way associative, 64B cacheline
LLC	8MB(shared), 16-way associative, MESI, 64B cacheline
Near Memory	HBM2 2GHz, 2GB, 8 128-bit channels, 8 banks tCAS-tRCD-tRP:7-7-7 RD/WR+I/O energy: 6.4pJ/bit ACT/PRE energy:15nJ
Far Memory	DDR4-3200, 16GB, 2 64-bit channels, 8 banks tCAS-tRCD-tRP: 22-22-22 RD/WR+I/O energy: 33pJ/bit ACT/PRE energy:15nJ

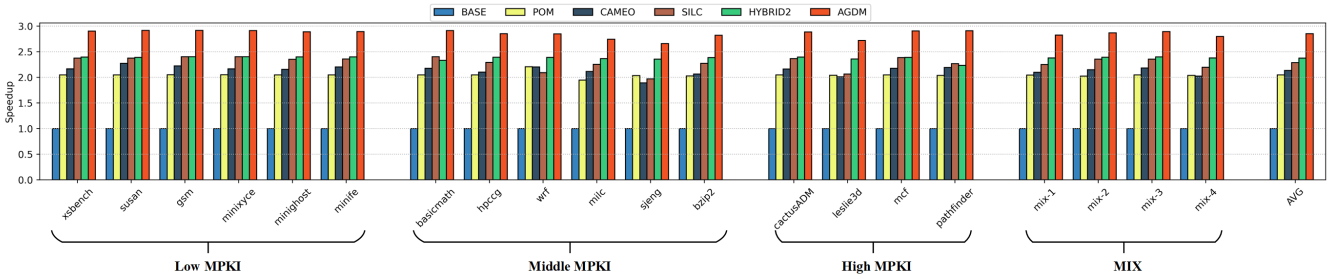

Fig. 4. Performance comparison with other schemes

- CAMEO [8]: Adopts 64B cacheline-sized migration granularity and combines data with the remapping entry to hide the metadata query latency.
- SILC [5]: Adopts set-associative remapping with 2KB page and supports 64B sub-block migration granularity between mapped pages.
- HYBRID2 [6]: Takes part of NM as a cache for FM and uses the rest of NM and the whole FM to build a flat address space. It uses 2KB page granularity for remapping and migrates FM block at 256B granularity.

All configurations are verified under capacity ratios of 1:4, 1:8, and 1:16 (NM:FM), respectively. Due to space limitation, we only show the experimental results of 1:8.

TABLE II
WORKLOAD CHARACTERISTICS.

Suite	Workload (×4)	MPKI (single)	Suite	Workload (×4)	MPKI (single)
CORAL	xsbench	2.10	Mantevo	pathfinder	91.45
MiBench	susan	0.02	SPEC	wrf	12.62
	gsm	3.22		milc	11.64
	basicmath	13.89		sjeng	13.90
Mantevo	minixyce	0.01		bzip2	15.25
	minighost	0.78		cactusADM	21.32
	minife	6.82		leslie3d	31.09
	hpccg	10.64		mcf	49.56
Mix-1	gsm leslie3d hpccg cactusADM	N/A	Mix-3	minighost susan libquantum mcf	N/A
Mix-2	minixyce milc basicmath xsbench	N/A	Mix-4	minife bzip2 wrf sjeng	N/A

B. Performance Analysis

Figure 4 shows the performance improvement of AGDM against other schemes. All performance results are normalized to the baseline configuration without NM. Except for mixed workloads, workloads are classified according to MPKI. Overall, AGDM achieves on average 39.02%, 33.18%, 24.45%, and 20.06% better speedup than POM [7], CAMEO [8], SILC [5], and HYBRID2 [6], respectively.

POM obtains an average 2.04x speedup than the baseline by intelligently selecting hot pages and placing them in NM. However, the threshold-based decision and long threshold adjustment period (10K LLC misses) make POM respond slowly to access pattern changes. CAMEO adopts a cache-like migration decision with 64B migration granularity. Once the accessed block is not in NM, it will be migrated immediately. Although CAMEO achieves up to 99.08% of NM utilization, the software overhead caused by its small migration granularity cannot be ignored. Besides, the unique hardware structure "LEAD" of CAMEO leads to a 14B invalid read on every access, severely wasting bandwidth and energy. SILC enables the on-demand migration by its sub-block interleaving. Only those accessed 64B sub-blocks of a 2KB remapped page in the FM need to be migrated to NM. However, the locking-page feature and long threshold adjustment epoch (1 million memory access) of SILC results in a slow response to access pattern changes. Although SILC successfully limits the migration traffic, it also misses the opportunity to migrate hot data into NM, resulting in performance degradation. Finally, HYBRID2 takes part of NM as a cache of the flat address space and uses 256B cacheline-sized migration granularity. The tag metadata of its small NM cache is totally loaded on chip, requiring 512KB capacity. Since the fixed migrating granularity cannot accommodate all workloads with different access patterns, these schemes show different performance comparisons in Figure 4.

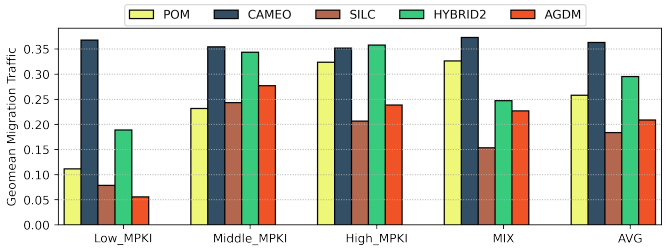

Fig. 5. Geomean of migration traffic

Compared to them, AGDM can match the appropriate migrating mode and granularity for the access pattern, regardless of the workload. The token-based migration decision of AGDM avoids migrating data blindly as cache-like policies while identifying hot data faster than threshold-based or epoch-based policies. The on-chip metadata structures ensure low cost for metadata operations. Therefore, AGDM achieves the best performance on all evaluated workloads.

C. Traffic and Energy Consumption

The two unique migration modes of AGDM benefit from migrating blocks to be accessed in advance. However, they also raise the concern of over-migration. Figure 5 shows the geometric mean of migration traffic percentage in total traffic. First, SILC causes the lowest migration traffic of 18.37% due to its on-demand migration policy. On the contrary, CAMEO

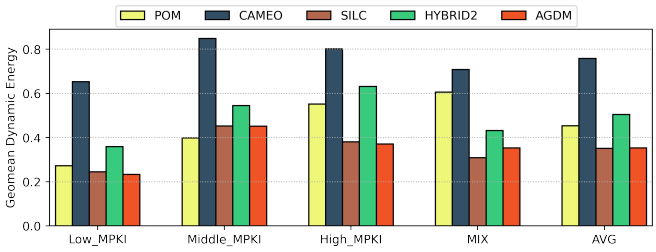

Fig. 6. Geomean of dynamic memory energy

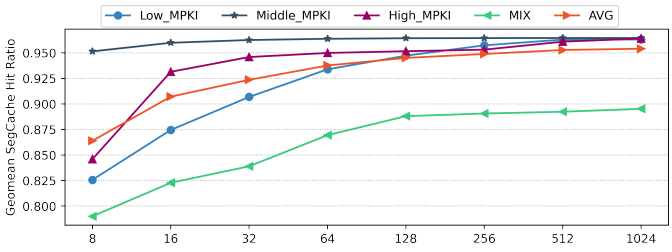

Fig. 7. Geomean segCache hit ratio

leads to the highest migration traffic of 36.29% since it blindly migrates every accessed FM block. HYBRID2 has an average 29.51% migration traffic since it needs to evict NM pages to make room for caching, except for regular migration traffic. The migration traffic percentage of POM varies widely on different workloads due to its 2KB migration granularity. The large migration granularity benefits from workloads with scoped access patterns. However, it is unsuitable for workloads with fine-grained random access. Overall, POM leads to an average 25.82% migration traffic. Finally, AGDM only generates 20.89% migration traffic on average, which is slightly higher than SILC and lower than other schemes. The migration traffic of AGDM is controlled for two reasons: i) The token-based migration decision avoids migrating blocks blindly as CAMEO, ii) AGDM chooses a conservative threshold to determine the scoped access pattern and discards short spatial footprints.

Figure 6 shows the geometric mean of dynamic memory system energy consumption normalized to the baseline. As we know in the analysis of the migrating traffic, CAMEO consumes the most dynamic energy, on average 75.85% of the baseline. SILC has the lowest energy consumption of 35.13%, which benefits from its on-demand migration. Although the migration traffic of AGDM is higher than SILC, AGDM has a slightly higher NM utilization than SILC. Thus, AGDM consumes almost the same energy as SILC and 29.98% less energy than HYBRID2. Overall, the distribution of dynamic memory energy consumption is consistent with the distribution of migration traffic.

D. Sensitivity Analysis

We next perform sensitivity studies on two on-chip metadata structures: segCache and remapping buffer. They are critical components for AGDM, and their hit ratios directly impact performance. We vary the capacity of the remapping buffer and segCache to seek a compromise between hardware overhead and the hit ratio. Due to space limitation and their experimental methods being almost the same, we only show the results of segCache in Figure 7. We vary the segCache capacity from 8 entries to 1024 entries. The growth trend of the hit ratio shows a turning point at 128. When the capacity exceeds 128, the hit ratio increases very slowly. Therefore, AGDM sets the default remapping buffer capacity as 128 entries, which achieves an average hit ratio of 94.57%. Similarly, the remapping buffer achieves an average hit ratio of 91.24% with 16 cachelines. A larger capacity can only bring a slight increase in the hit ratio.

V. CONCLUSION

In this work, we propose AGDM, an access-pattern-aware adaptive granularity data migration strategy for hybrid memory systems. AGDM tackles the challenges of adaptive granularity through novel remapping-migration decoupled metadata organization and runtime access pattern monitor. We use four widely-used benchmark suites that contain a variety of workloads to demonstrate the adaptability of AGDM. The evaluation shows that, compared to HYBRID2, one of the state-of-the-art schemes, AGDM improves performance by 20.06% and saves dynamic memory energy consumption by 29.98%.

VI. ACKNOWLEDGE

This work was supported by the National Natural Science Foundation of China No. 61821003 and No. 61832007. This work was also supported by Engineer Research Center of data storage systems and Technology, Ministry of Education, China. We thank Zuoning Chen for her long-term help and support for our laboratory.

REFERENCES

[1] JEDEC Standard, "High bandwidth memory (hbm) dram." Jesd235, 2013
[2] J. Jeddeloh and B. Keeth, "Hybrid memory cube new DRAM architecture increases density and performance," in *VLSIT*, 2012, pp. 87-88.
[3] M. K. Qureshi et al., "Phase change memory: From devices to systems." *Synthesis Lectures on Computer Architecture*, 2011, 6(4), 1-134.
[4] F. T. Hady et al., "Platform Storage Performance With 3D XPoint Technology," in *Proceedings of the IEEE*, vol. 105, no. 9, 2017.
[5] J. H. Ryoo et al., "SILC-FM: Subblocked InterLeaved Cache-Like Flat Memory Organization," in *HPCA*, 2017, pp. 349-360.
[6] E. Vasilakis et al., "Hybrid2: Combining Caching and Migration in Hybrid Memory Systems," in *HPCA*, 2020, pp. 649-662.
[7] J. Sim et al., "Transparent Hardware Management of Stacked DRAM as Part of Memory," in *MICRO*, 2014, pp. 13-24.
[8] C. C. Chou et al., "CAMEO: A Two-Level Memory Organization with Capacity of Main Memory and Flexibility of Hardware-Managed Cache," in *MICRO*, 2014, pp. 1-12.
[9] Y. Li et al., "Utility-Based Hybrid Memory Management," in *CLUSTER*, 2017, pp. 152-165.
[10] C. F. Chen et al., "Accurate and complexity-effective spatial pattern prediction," in *HPCA*, 2004, pp. 276-287.
[11] B. Nathan et al., "The gem5 simulator." *ACM SIGARCH computer architecture news* 39.2, 2011: 1-7.
[12] C. W. Smullen et al., "Relaxing non-volatility for fast and energy-efficient STT-RAM caches," in *HPCA*, 2011, pp. 50-61.
[13] L. H. John, "SPEC CPU2006 benchmark descriptions." *ACM SIGARCH Computer Architecture News* 34.4, 2006: 1-17.
[14] A. H. Michael et al., "Improving performance via mini-applications." No. SAND2009-5574. *Sandia National Laboratories (SNL)*, 2009.
[15] M. R. Guthaus et al., "MiBench: A free, commercially representative embedded benchmark suite," *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization*. WWC-4 (Cat. No.01EX538), 2001, pp. 3-14.
[16] J. R. Tramm et al., "XSBench-the development and verification of a performance abstraction for Monte Carlo reactor analysis." *The Role of Reactor Physics toward a Sustainable Future (PHYSOR)*, 2014.