Merchandiser: Data Placement on Heterogeneous Memory for Task-Parallel HPC Applications with Load-Balance Awareness

Zhen Xie[1,2], Dong Li[1]

University of California, Merced[1]

Argonne National Laboratory[2]

Motivation and Introduction

- **HM raises a data placement problem**
  - Because of small capacity of fast memory and relatively worse performance of slow memory.
  - Memory pages must be allocated and migrated between fast and slow memories.
  - To make sure that most of memory accesses can happen in fast memory for high performance
- **Many solutions to address the data placement problem on HM uses a profiling-guided optimization (PGO) approach**
  - These solutions identify frequently accessed memory pages (``hot pages'') by periodically sampling memory pages and tracking memory accesses to them
  - Hot pages are then migrated to fast memory for better performance.

Research questions:

- The PGO on HM cannot work well for task-parallel applications
  - Because they lack a view of ``finishing all tasks fast'' for high performance.
  - They migrate and place hot pages into fast memory, but do not consider which task accesses those memory pages.

Methodology

- Introduce a load balance-aware data placement system for HM, named Merchandiser, to address the problem
  - Merchandiser introduces task semantics during memory profiling. This means Merchandiser associates memory accesses with tasks during profiling, instead of being application-agnostic.
  - Using limited task semantics, Merchandiser effectively sets up coordination among tasks on the usage of HM.
  - Merchandiser uses historical, fine-grained profiling results of the task to guide data placement for the subsequent executions of the same task with new inputs.
- Performance modeling to predict execution time of the task:
  - The novelty of our performance modeling lies in the modeling of performance correlation between different data placements of the task.
  - Performance modeling takes the performance of a data placement as input, and then predicts the performance of another data placement.
- Greedy heuristic algorithm:
  - Decide how to allocate the fast memory space among tasks to maximize performance benefit of all tasks

Evaluation Results

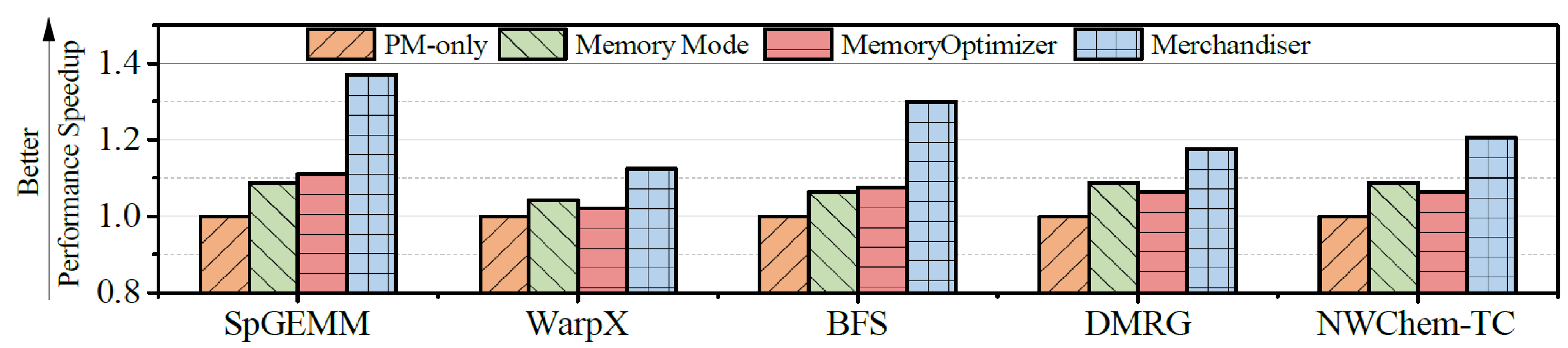

Figure 1: Performance of Memory Mode, MemoryOptimizer, and Merchandiser, compared to the PM-only execution.

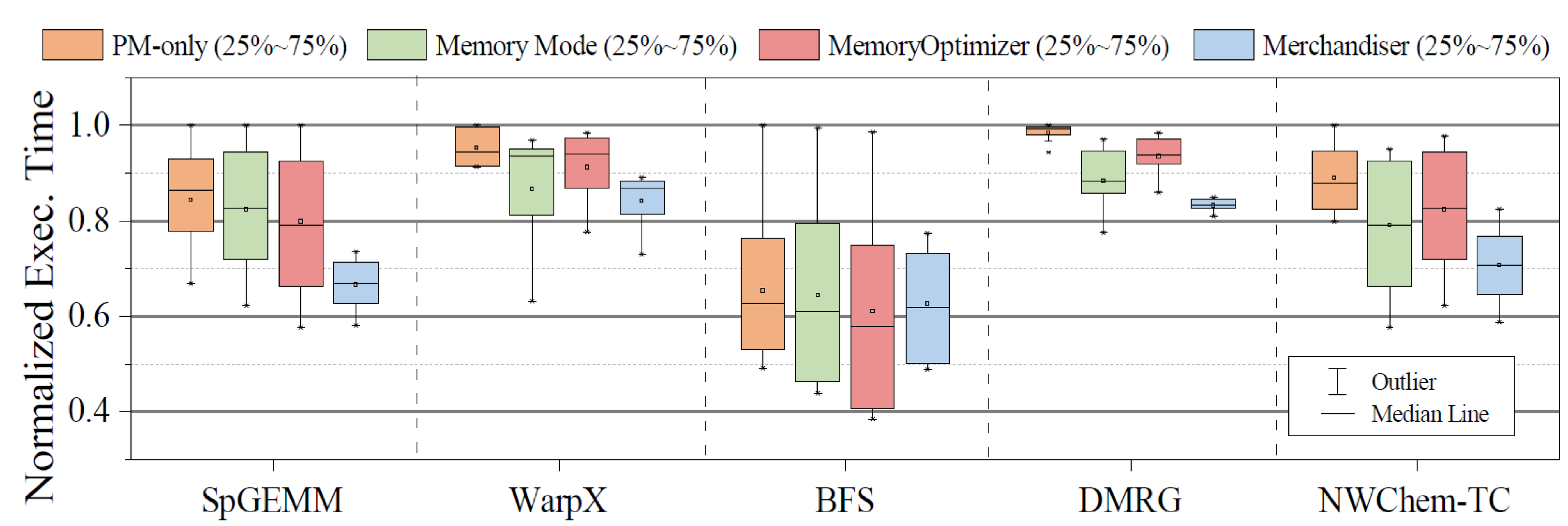

Figure 2: Task execution time and their variance

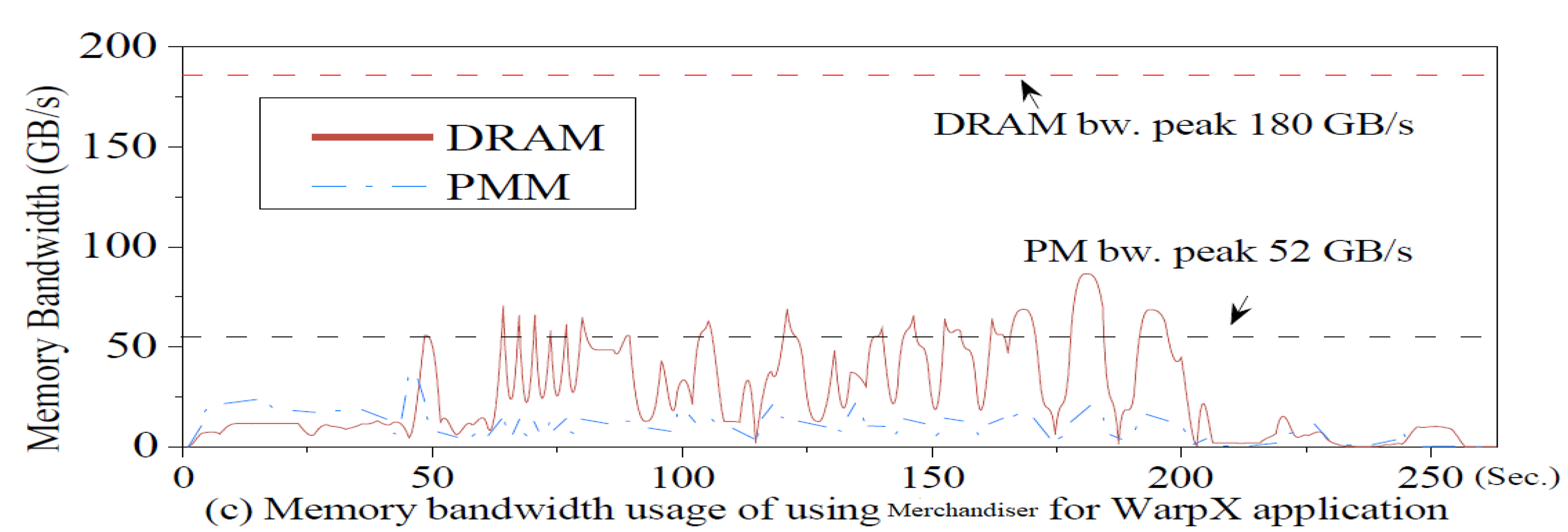

Figure 3: Memory bandwidth usage

Observation 1:

Figure1 shows overall performance normalized to PM-only (Optane-only). Merchandiser introduces 23.6%, 17.1%, and 15.4% performance improvement on average (up to 37.8%, 26.0%, and 23.2%) over PM-only, Memory Mode, and MemoryOptimizer respectively.

For applications with large load imbalance, such as BFS and NWChem-TC, the number of pages being migrated among tasks can vary by up to 21.4 times.

Observation 2:

Compared with Memory Mode and MemoryOptimizer, Merchandiser reduces A.C.V (average coefficient of variation of execution time across threads/processes) by 51.6% and 42.7% on average respectively.

Observation 3:

Compared with Memory Mode, Merchandiser increases average DRAM bandwidth usage from 5.98 GB/s to 24.31 GB/s, indicating the usage of fast memory is improved;

Meanwhile, the average PM bandwidth usage is reduced from 13.74 GB/s to 9.97 GB/s, indicating the effectiveness of page migration in Merchandiser

Conclusion

- The traditional wisdom “migrating frequently accessed pages to fast memory leads to better performance” is not necessarily correct.
- We introducing task semantics during memory profiling and migration to address the limitation of traditional wisdom.
- We introduce a load balance-aware data placement system for HM.

References

[1] Zhen Xie, Jie Liu, Jiajia Li, and Dong Li. 2023. Merchandiser: Data Placement on Heterogeneous Memory for Task-Parallel HPC Applications with Load-Balance Awareness. In Proceedings of the 28th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming.