Optimizing Memory Latency in Hardware Disaggregated Memory Systems

Amit Puri, John Jose, Tamarpalli Venkatesh

Indian Institute of Technology Guwahati, Assam, India

1 Paper Summary

Traditional server that are deployed in data centers face severe memory underutilization due to inconsistant memory allocation. The memory resources gets stranded within different servers in the form of small fragments, making them unusable. Hardware memory disaggregation is a solid alternative of traditional server architecture to overcome its limitations. It decouples the server's memory into seperate resource pools connected through high-speed network interfaces. Server nodes (compute nodes) have a small amount of local on-chip memory and mostly rely on remote memory from memory pools for application requirement that can be allocated on-demand, improving its utilization.

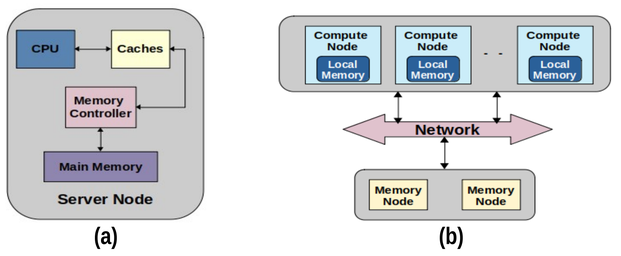

Figure 1: (a) Traditional Server Design vs (b) Disaggregated Memory Design

However, presence of network increase the memory access latency at remote memory pools (or nodes), significantly impacting the system performance. Disaggregated system require a series of system optimizations to reduce memory cost. Currently, there are no commercialy available disaggregated memory systems and many of the system level details are not clear. In our disseration, we work on these research gaps to propose a practical solution for scalable memory disaggregation. We also propose a unique cost-effective hot-page migration mechanism to significantly improve the memory access latency and hence the system performance. We build a scalable disaggregated memory simulator to evaluate our designs.

2 Disaggregated Memory System Design

Memory-semantic fabrics like CXL support coherence access to remote memory. The compute nodes can directly access a cache block in remote memory from on LLC miss with a latency of around 170ns-250ns. A remote memory controller is an addressable hardware module similar to DRAM controller, which is connected to on-chip bus and forwards memory requests belonging to remote memory to the network. A similar memory controller is present at memory nodes to send response.

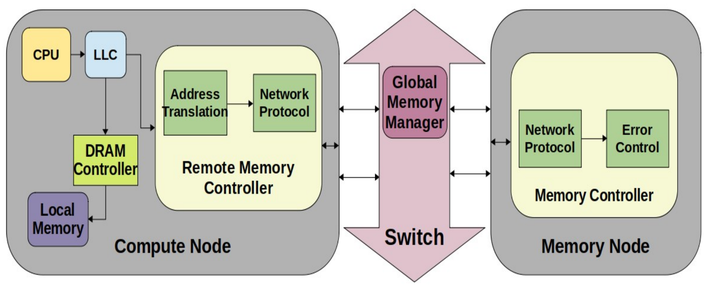

Figure 2: Overview of Disaggregated Memory System

3 Remote Memory Organization

Remote memory address space can be organized in multiple ways:

- **Shared :** Transparent global address space **|** Easy to spread workload across nodes but significant coherency traffic **|** Bottleneck in remote page allocation.
- **Distributed :** Exclusive access to each node **|** Lesser coherency traffic **|** Remory memory can be allocated in larger chunks **|** Use another layer of address translation

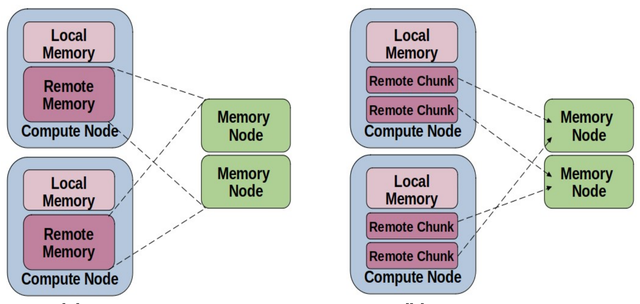

Figure 3: Remote Memory Organization (a) Shared (b) Distributed

Shared memory space is improtant when workloads did not fit into local memory. Most data-centric workloads can easily get compute resources with in single node. With memory being moved to seperate pools on disaggregation, it is better to use distributed approaach. An address map for allocated remote memory chunks in remote memory controller.

4 Remote Memory Allocation and Pool Selection

- Multiple compute nodes with different memory access patterns and footprints use same remote memory pools, while global memory manager allocates remote memory to them.
- If memory pool selection is such that the memory requests are not balanced among memory pools, the network will face congestion and memory pools will face contention in its queues.
- **Random** or **Round-Robin** pool Selection does not distribute memory requests equally and has large variation in total memory requests among the pools. Due to which it faces tail latency.

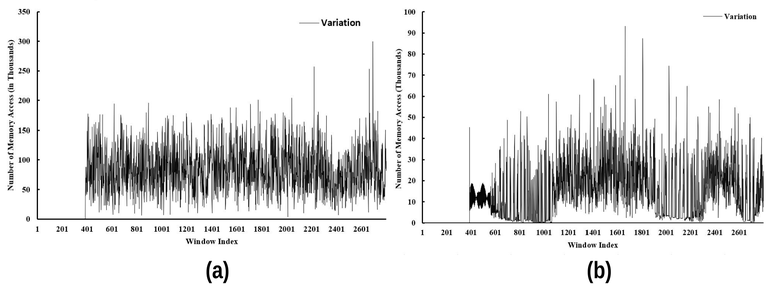

Figure 4: Variation in queue allocation (a) Random (b) Round-Robin

5 Proposed Pool Selection Policies

- **Smart-Idle Pool Selection :** Monitors memory access traffic to each pool before allocating new chunk. The idle memory pool is selected for memory allocation.
- **Uniform-Load Partition :** Divide compute nodes into sets with each set having same memory request rate. Each set is mapped to a memory pool.

Trace-based Simulation

- Pintool for instrumentation and multi-core cache modeling.
- Multiple main memory traces collected, one for each node.
- Traces parsed in parallelfor network and remote memory simulation.
- DRAMSim2 for memory simulation.
- Multiple DRAMSim2 instances each for local memory and remote memory units

Network:

- NIC Node: 100Gbps/10ns (De)-Packetization
- Switch: 400Gbps/5ns Processing Delay

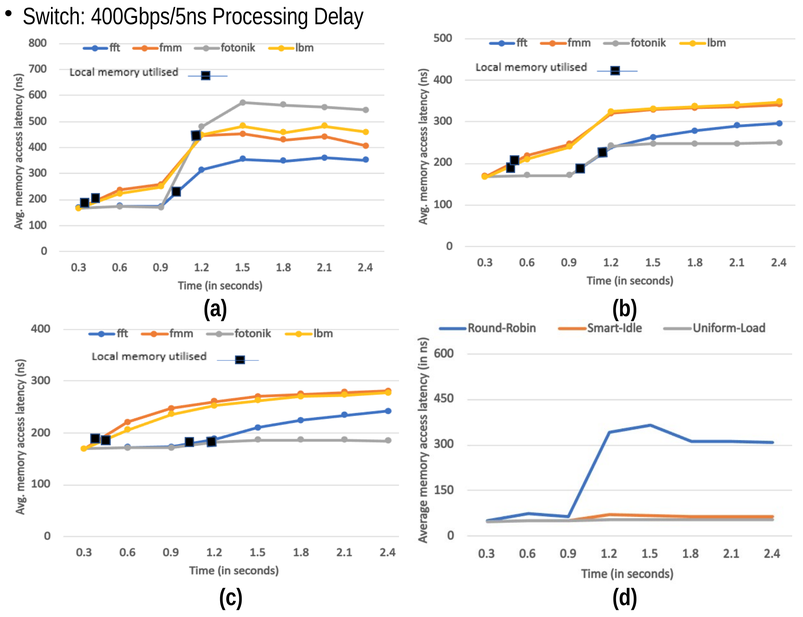

Figure 5: Avergae Memory Access Latency (a) Round-Robin (b) Smart-Idle (c) Uniform-Load Partition | (d) Average Remote Memory Latency

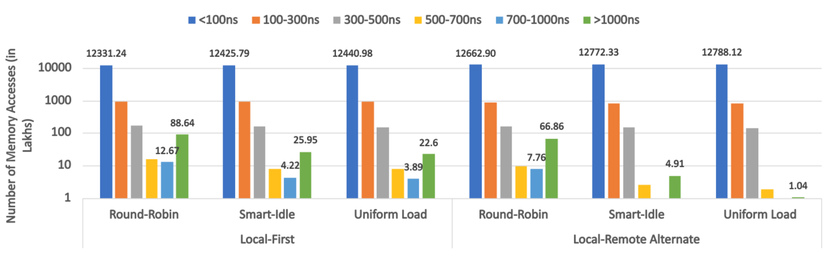

Figure 6: Impact on Tail Latency

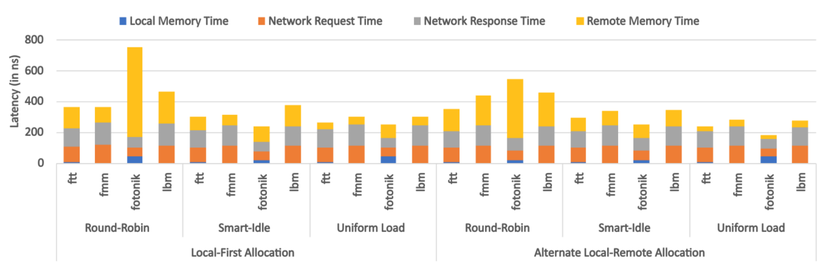

Figure 6: Latency Distribution (Local/Network/Remote)

6 Cost Effective Hot-Page Migration (*CosMo*)

Remote memory latency can be reduced by Hot-Page migration from remote to local memory and using locality of memory accesses in those pages. However, it has multiple issues:

- Multi-tiered Memory Management makes difficult to track hot-pages
- Page migration require page-table updates and introduces long CPU stalls for TLB-shootdown (4-13µs based on number of cores)
- Lastly, accessing page consumes memory and network bandwidth and starves the subsequent block accesses to other pages on their critical path, introducing slow-downs.

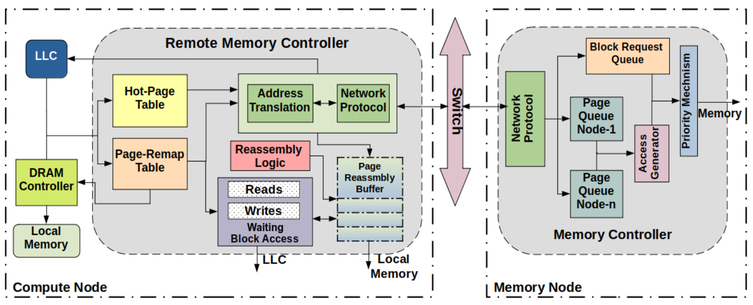

Figure 7: Proposed architecture for Hot-Page migration

Hot Page Table

Page Address	Access Counter	First Access Time
Entry-1		
-		
Entry-n		

(a)

Page Remap Table

Old Address	New Address	Is Local
R1	L1	1
R2	L2	0
-	-	-

(b)

Figure 8: Hardware Structures (a) Hot-Page Tracker (b) Page Remap Table

Hot Page Tracker :

- Track hot pages in multi-tiered memory system
- Training based migration thresholds based on Access Count and Reuse Frequency

Page Remap Table:

- Stores the new local physical addresses of migrated pages.
- Perform page-table updates in batches.
- Memory access to these pages gets new address from this table.

6 Cont..

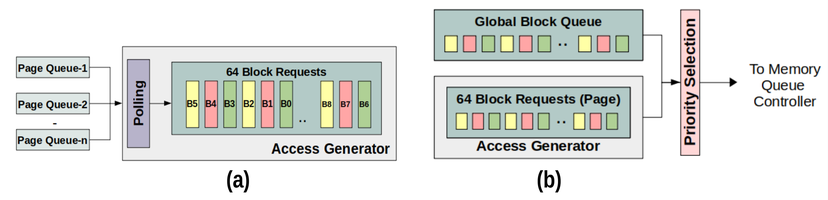

Figure 9: (a) Access Generator for Page Memory Request (b) Bandiwtdh Allocation Using Priority Selection

Access Generator :

- Generate block level accesses for accessing identified hot pages
- Selects between multiple Page-queues of different nodes, to equally partition bandwidth

Bandwidth Allocation

- Selects between regular block accesses and those belonging to pages for equal bandiwtdh partition.

Mechanism eliminates starvation to subsequent block accesses..!

7 Methodology and Results

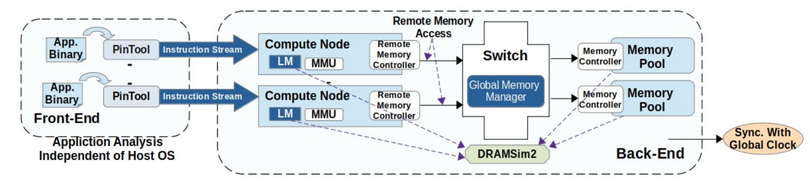

Figure 9: Cycle-level simulation for Multi-node Simulation with OOO computing cores

Memory : 1200x2MHz DDR4 DRAM (19.4Gbps)

Switch : 100/400Gbps, 4MB Port buffer, 5/15ns processing/switching

NIC (Nodes) : 40/100Gbps, 1MB buffer, 10/30ns for (De)-Packetization/processing

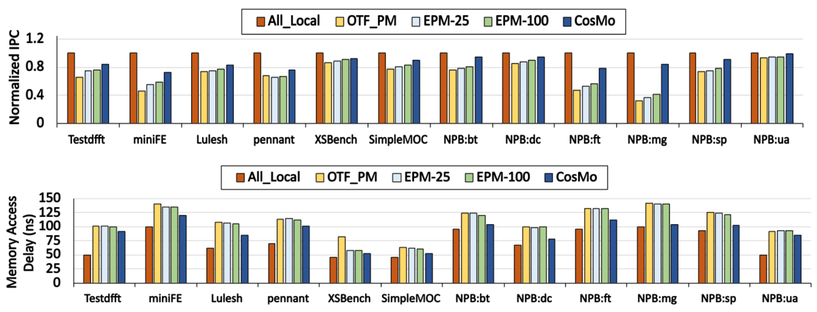

Figure 10: Normalized IPC and Memory Latency in different page migration schemes compared to *CosMo*

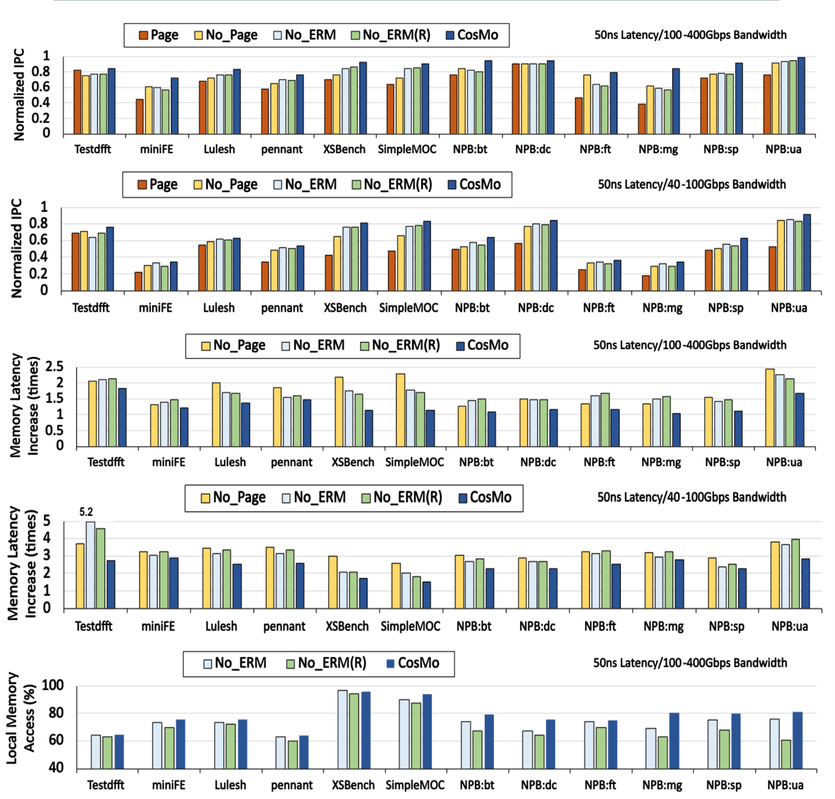

Figure 11: Normalized IPC, Memory Cost Increase and Local Hit Ratio in different data movement schemes compared to *CosMo*

8 Related Work

Komareddy et al. [1][2] explored memory allocation policies for shared memory approach for NVM based disaggregated pools. Even though remote memory pages can directly be allocated and do not actually face the issue of imbalanced memory requests, the coherency is the bigger issue, which increase the waiting time for memory accesses. Hot page migration has been explored in DRAM-NVM hybrid memory systems [3][4][5] in the past which have cnetralized memory management, making it easy to track pages. This does not apply to multi-tiered disaggregated memory management. Further, the interconnect is not the issue in these systems. The solutions are not applicable to hardware disaggregation. Page migration have also been used in the software disaggregated system [6], these systems only allow remote memory access at page granularity and free memory in other servers to swap out cold pages rather to slow disk. These design does not translate to hardware disaggregation.

9 References

[1] Vamsee Reddy Kommareddy, Amro Awad, Clayton Hughes, and Simon David Hammond. 2018. Exploring Allocation Policies in Disaggregated Non-Volatile Memories. In Proceedings of the MCHPC'18, doi.: 10.1145/3286475.3286480

[2] V. R. Kommareddy, C. Hughes, S. Hammond and A. Awad, "Investigating Fairness in Disaggregated Non-Volatile Memories," 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Miami, FL, USA, 2019, pp. 104-110, doi: 10.1109/ISVLSI.2019.00028.

[3] T. Repantis, C.D. Antonopoulos, V. Kalogeraki, and T.S. Papatheodorou. 2004. Dynamic page migration in software DSM systems. In 2004 IEEE International Conference on Cluster Computing (IEEE Cat. No.04EX935). 494–.doi: 10.1109/CLUSTR.2004.1392659

[4] Yujuan Tan, Baiping Wang, Zhichao Yan, Witawas Srisa-an, Xianzhang Chen, and Duo Liu. 2020. APMigration: Improving Performance of Hybrid Memory Performance via An Adaptive Page Migration Method. IEEE Transactions on Parallel and Distributed Systems 31, 2 (2020), 266–278.doi: 10.1109/TPDS.2019.2933521

[5]Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, and Kang G. Shin. 2017. Efficient Memory Disaggregation with INFINISWAP. In Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation (Boston, MA, USA) (NSDI'17). USENIX Association, USA, 649–667.