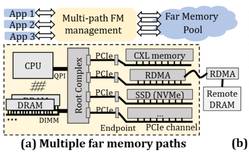
Boosting Data Center Performance via Intelligently Managed Multi-backend Disaggregated Memory

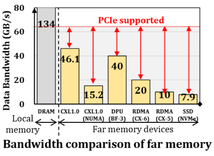
Jing Wang, Hanzhang Yang, Chao Li*, Yiming Zhuansun, Wang Yuan, Cheng Xu, Xiaofeng Hou, Minyi Guo, Yang Hu, Yaqian Zhao

Shanghai Jiao Tong University, Tsinghua University, IEIT SYSTEMS Co., Ltd

Introduction



(a) Multiple far memory paths



(b) Bandwidth comparison of far memory

Figure 1: (a) Multi-backend far memory. (b)Bandwidth comparison of various far memory technologies.

In recent years, disaggregated memory (DM) architecture stands out as a way to enhance data center capabilities by offering highly flexible memory expansion. In this case, a memory-hungry monolithic server can access a PCI Express (PCIe) based secondary memory device (i.e., far memory) with low data access latency.

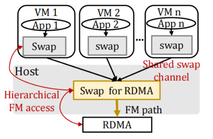
Unfortunately, prior works mostly limit their designs on a **single far memory (FM) device** based on PCIe, which cannot meet the needs of high data/task throughput in today's data center. There is a wide gap between the data bandwidth that a single FM device could support (from 7.9 to 46 GB/s) and the maximum available bandwidth that modern PCIe protocol could provide, as shown in Figure 1.

Incorporating multiple FM devices and oversubscribing the PCIe subsystem allows one to push the limit of server data throughput. This approach, which we call **multi-backend disaggregated memory**, unfortunately cannot be implemented solely based on current data transfer protocols and device drivers. Existing FM management schemes are blind to the possible multiple physical FM channels: they logically only support one data exchange path to FM devices due to original swap mechanism limitation.

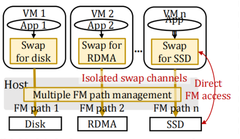
We introduce **xDM**, a novel FM management system engineered to support high-performance data analytic workloads running on the new multi-backend disaggregated memory architecture.

Challenges and Motivations

- **Far Memory Usage Bottleneck**: It is impractical to directly support multi-path far memory management in the OS. A naive VM-based FM system has limitations due to the hierarchical swap architecture and considerable data swap overhead. Removing FM usage bottleneck requires expanding data swap channels and tapping into light-weight swapping.
- **Far Memory Usage Effectiveness**: Most of the prior works follow a simple idea: by offloading part of data to far memory based on workload behaviors, the local DRAM can retain more latency-sensitive tasks. Applications often exhibit more complex performance variation, which requires a detailed analysis of page behaviors. These works unfortunately overlook the importance of several key system parameters.



(a) Single far memory access path



(b) Multiple far memory access paths

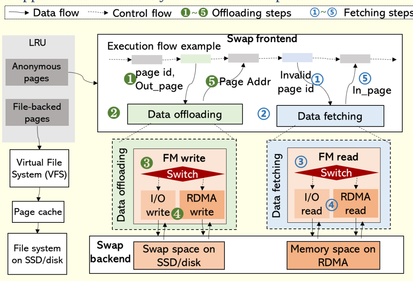
Design

xDM is the first multi-path FM management system that not only allows ***simultaneous*** access of multiple FM devices but also provides ***dynamic and implicit*** adjustment of FM paths based on program behaviors.

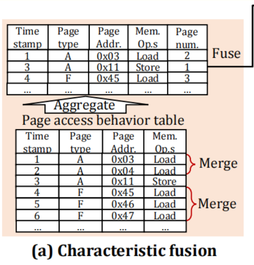
(1) Dynamic FM Switching Mechanism:

Swichable FM Swapper: The switchable swapper includes a modified swap frontend plus a variety of adaptive FM swap backends. Leveraging the data swap interfaces provided by Frontswap, we redirect page swapping to our customized FM read and write functions.

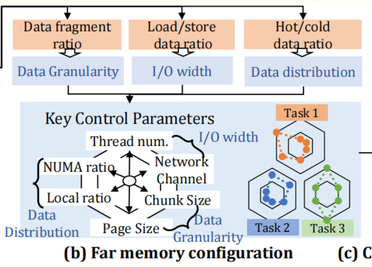
Implicit FM Switching Strategy: We design a far memory switching strategy to choose FM backends based on the page distribution statistics and application's sensitivity to different FM paths.



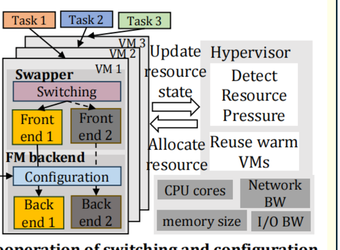
(2) Smart FM Configuration Console: Data characteristic Fusion and FM Parameter Adjustment



(a) Characteristic fusion



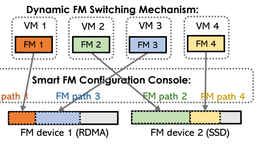
(b) Far memory configuration



(c) Cooperation of switching and configuration

xDM's adaptive FM parameter configuration. The characteristic fusion module provides a synthesis of information from page access traces. xDM can configure and fine-tune multi-backend FM from various aspects.

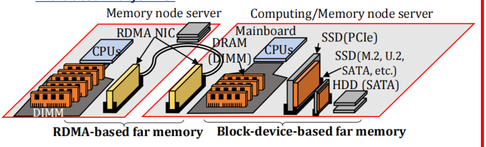
Conclusion



- In this work, we design and implement xDM, a novel **multi-backend far memory system** with high bandwidth utilization and application performance.
- By turning the conventional swap mechanism into a **switchable data swap module**, we successfully realize simultaneous multi-path FM access.
- Based on a rich synthesis of application **page data**, we tailor the far memory path configurations to the needs of various applications.
- Our work shows up to **3.9x data swap** performance speedup, **2.8x data throughput** increase, and **5.1x data** center task serving **throughput** improvement compared with state-of-the-art works.
- Our design provides **a flexible solution** to scale out far memory access paths and an efficient way to manage them on monolithic servers. We expect that our design can improve the execution performance and **memory usage effectiveness** of memory-hungry tasks in cloud and near-edge micro data centers.

Evaluation

Evaluated Platforms:



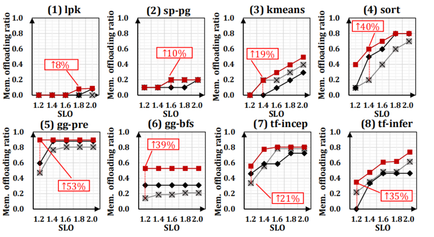
RDMA-based far memory **Block-device-based far memory**

We show the swap performance speedup (Sp.) of our xDM compared with baselines on different backends. We categorize application swap features into two types: swap-sensitive (S, average Sp.≤ 1.5x) and swap-friendly (F, average Sp. > 1.5x).

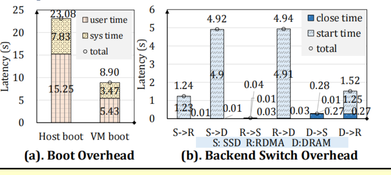
Evalauted Workload	stream	lpk	kmeans	sort	s-pg	gg-pre	gg-bfs	lg-bfs
Swap Feature	S	S	S	S	S	F	S	F
Sp. on DRAM	**1.32×**	1.18×	**1.64×**	1.05×	**1.44×**	**2.24×**	**1.29×**	2.00×
Sp. on SSD	1.01×	**1.52×**	0.88×	0.86×	1.01×	1.02×	1.18×	1.40×
Sp. on RDMA	1.25×	1.09×	1.40×	**1.40×**	1.37×	2.06×	1.19×	**2.24×**
Average Speedup	1.19×	1.26×	1.31×	1.11×	1.28×	1.77×	1.22×	1.88×

Evalauted Workload	lg-comp	lg-mis	tf-infer	tf-incep	clip	tf-tc	chat-int	bert
Swap Feature	F	F	F	F	S	F	F	S
Sp. on DRAM	**2.43×**	**2.17×**	1.88×	1.72×	0.82×	1.28×	1.15×	1.03×
Sp. on SSD	1.52×	1.36×	1.51×	1.34×	0.91×	2.16×	1.92×	**1.75×**
Sp. on RDMA	2.22×	2.07×	**2.70×**	**2.53×**	**2.46×**	**2.55×**	**3.89×**	1.10×
Average Speedup	2.05×	1.86×	2.03×	1.86×	1.40×	2.00×	2.32×	1.29×

Our design shows **larger memory offloading ratios** than baselines on evaluated workloads under different SLOs.



The acceptable **overhead** of virtualization and backend switching.



(a). Boot Overhead (b). Backend Switch Overhead

For more details, please see our full paper in SC'24.

Acknowledgement

We sincerely thank all the anonymous reviewers for their valuable comments. This work is supported by the National Key R&D Program of China (No. 2022YFB4501702), and the National Natural Science Foundation of China (No. 62122053). The corresponding author is Chao Li. Thanks for the support of Emerging Parallel Computing Center (EPCC).