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

jing618@sjtu.edu.cn

SHANGHAI JIAO TONG UNIVERSITY

SC24 Atlanta, GA | hpc creates.

"

1. Background
2. Motivation
3. System Design
4. Experimental Result
5. Conclusion and Future Work

"

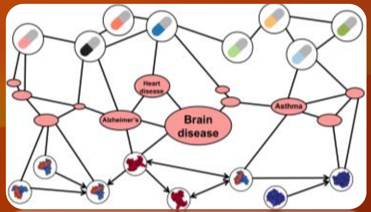
“

1. Background
2. Motivation
3. System Design
4. Experimental Result
5. Conclusion and Future Work

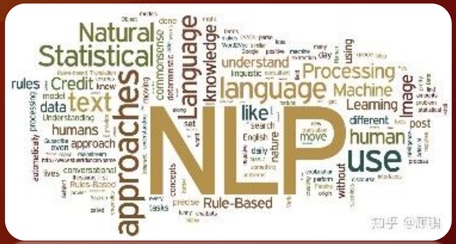
”

1. Background: Growing Application Data

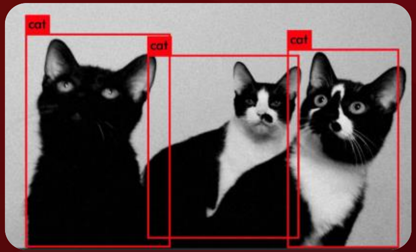
Social Network



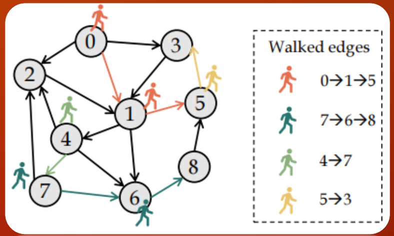
Drug Detection



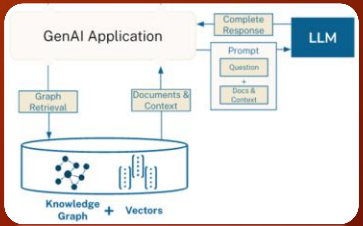
Natural Language Processing



Computer Vision



Graph Netural Network



Knowledge Retrieval

Speech Recognition

AI Generated Content

Data Scale of Graph Processing:

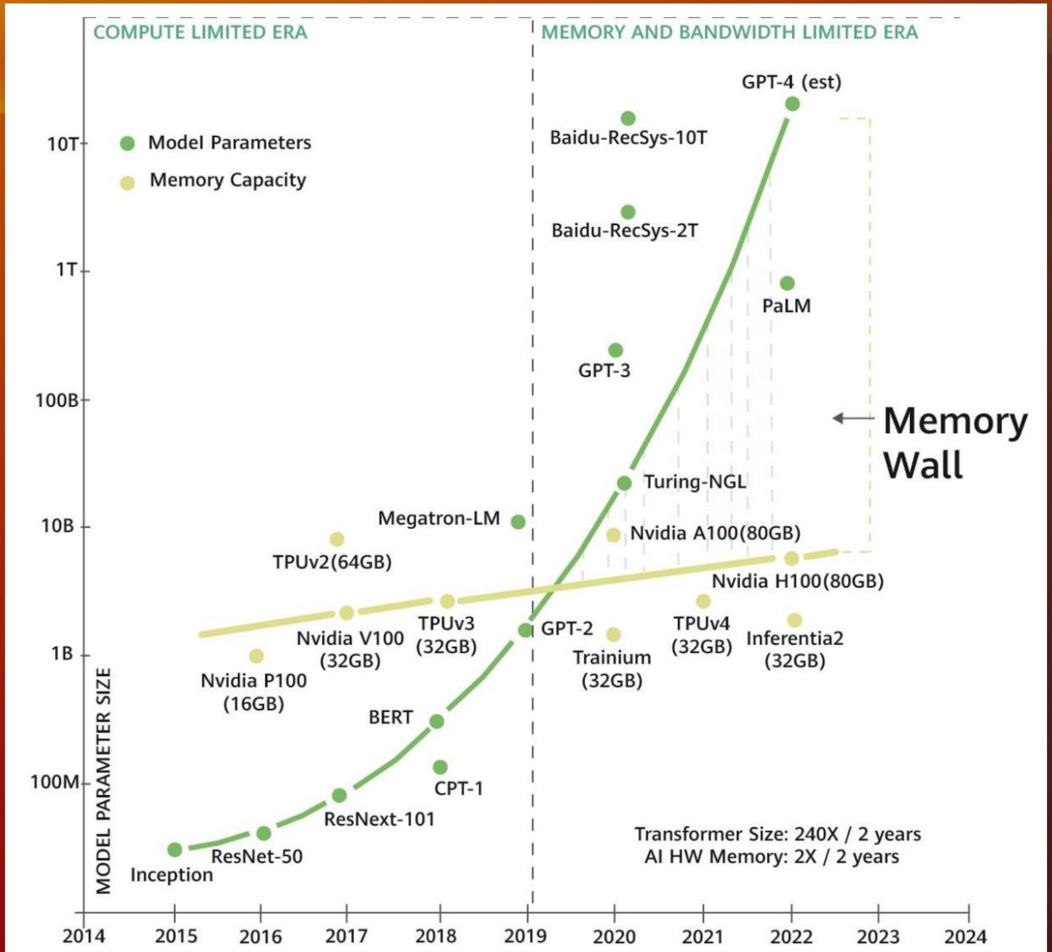
- Tens of Billion of Vertices
- Hundreds of Billion of Edges

Data Scale of AI training/inference:

- Billions of Model Parameters
- Trillions of Tokens

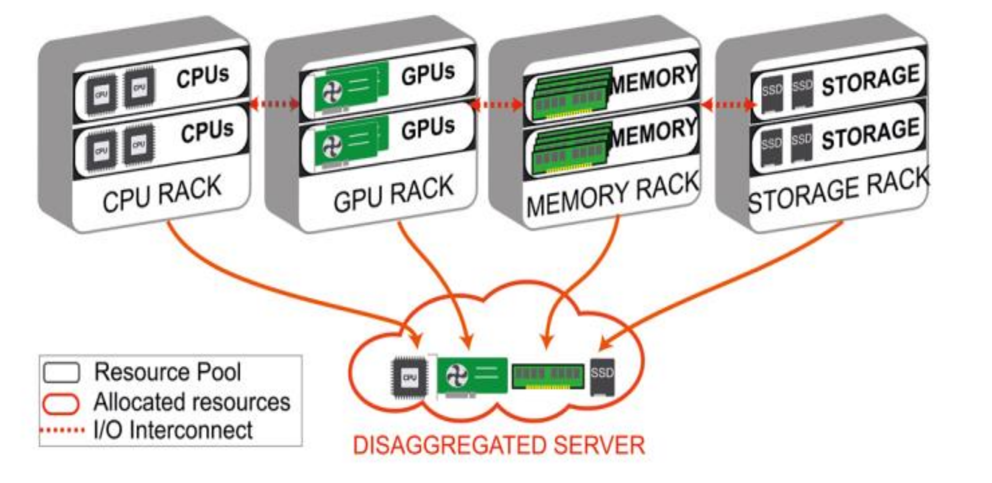
Data centers necessitate large memory capacity and efficient data management.

1. Background: Disaggregated Architecture



--From Write Paper of Elastic Memory System in Huawei Cloud in 2024

Calling for Elastic and Intelligent memory system

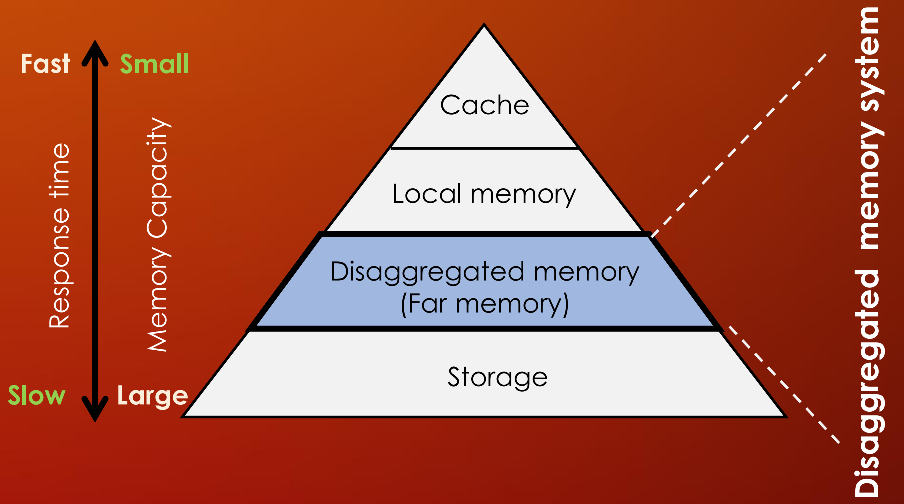


Disaggregated architecture

- Large capacity
- Flexibility
- Scalability

Disaggregated architecture can provides large memory pools.

1. Background: Disaggregated Memory Related Works



SSD-based far memory system:
- Zswap, ASPLOS'19
- Kona, ASPLOS'21
- TMO, ASPLOS'22
- BAM, ASPLOS'23
- CachedAttention, ATC'24

RDMA-based far memory system:
- AIFM, OSDI'20
- ThymesisFlow, MICRO'20
- Fastswap, Eurosys'20
- Sherman, SIGMOD'22
- Memliner, OSDI'22
- Canvas, NSDI'23
- Unimem, ATC'24

CXL-based far memory system:
- BEACON, MICRO'22
- Pond, ASPLOS'23
- ReCXL, ISCA'24

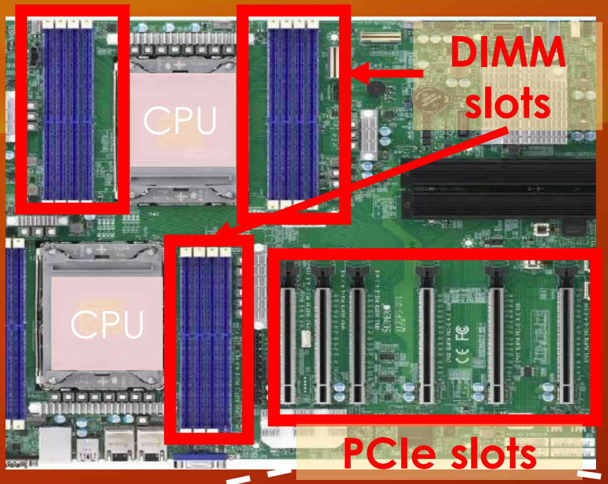
While the addition of far memory (FM) could relieve a server's memory pressure, it unfortunately cannot meet the needs of high data/task throughput in today's data center.

“

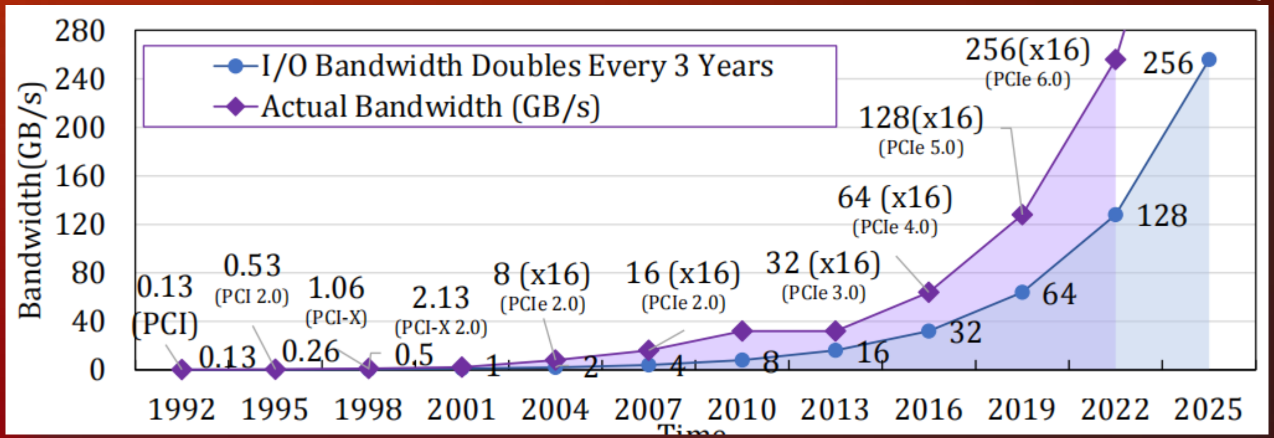
1. Background
2. Motivation
3. System Design
4. Experimental Result
5. Conclusion and Future Work

”

2. Motivation: Memory Extension Ways

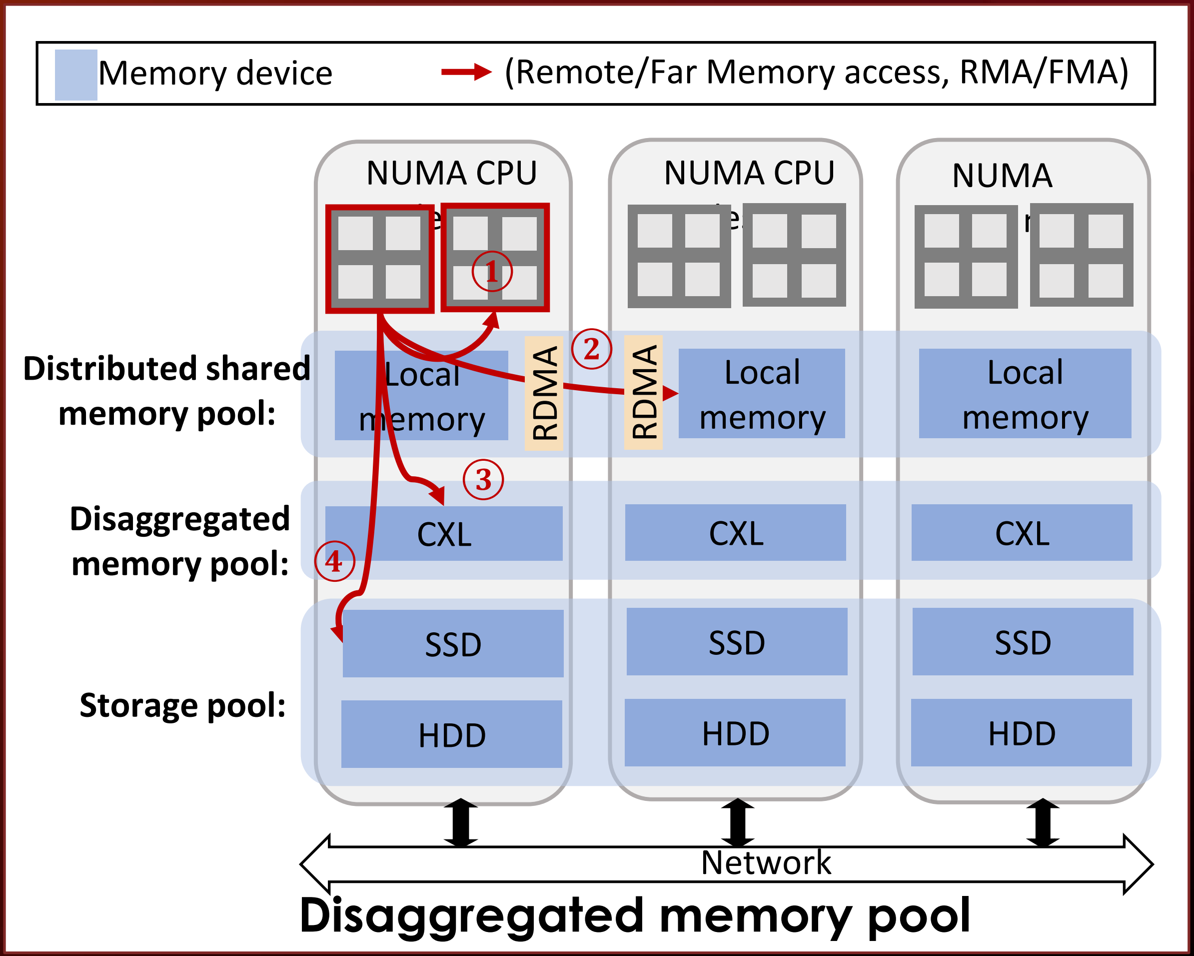


Limited On-chip **DIMM** slots (for memory devices) and **PCIe** slots (for accelerators, network, storages, and new memory devices, etc.)



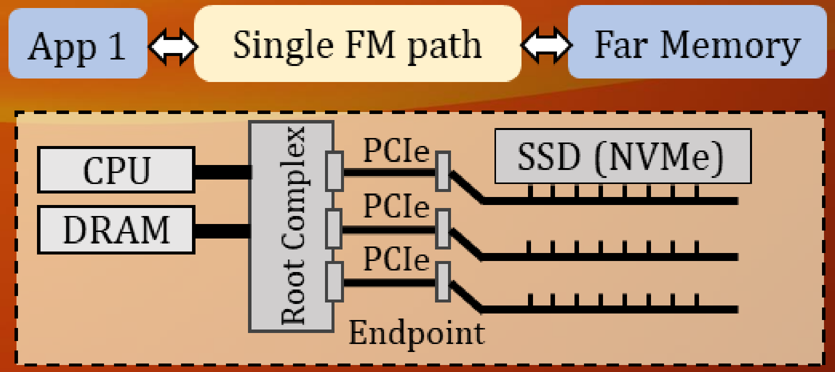
https://www.theverge.com/2022/1/12/22879732/pcie-6-0-final-specification-bandwidth-speeds.

PCIe bandwidth grows faster than estimated I/O trends

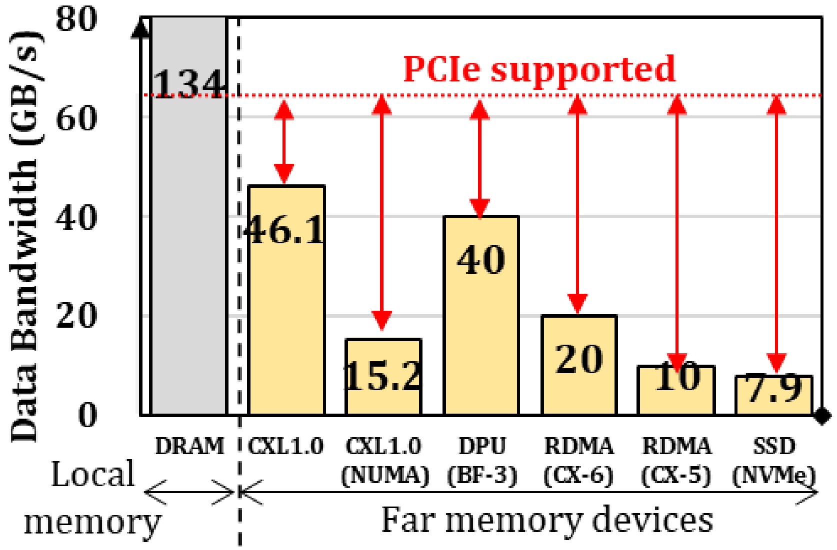


Memory extension ways (Far memory paths)

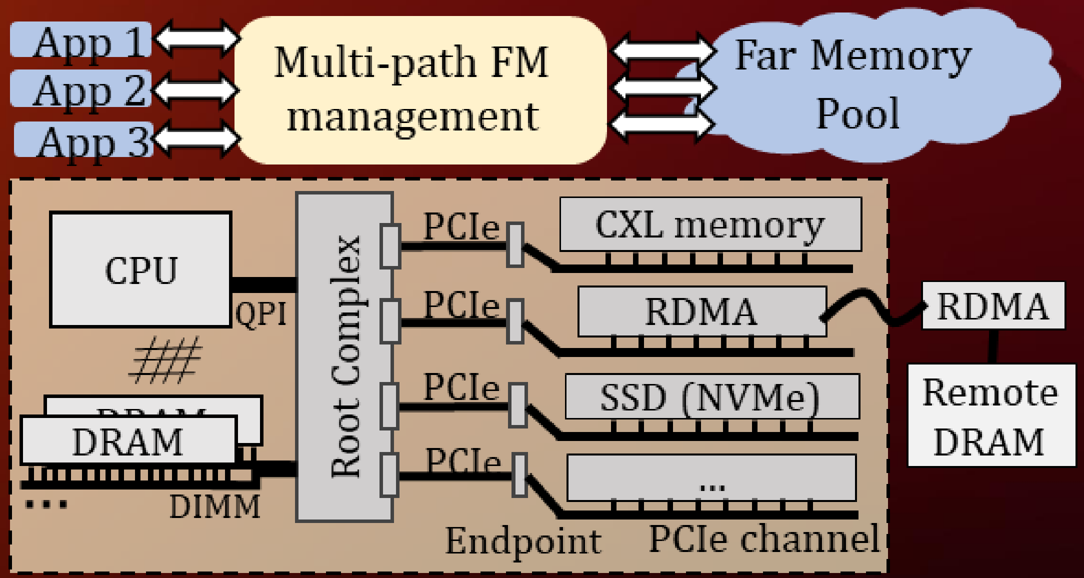
2. Motivation: From Single to Multiple Channels



Prior works limit their designs on a single FM device.



Single far memory device could become a crucial bottleneck due to data bandwidth limitation.



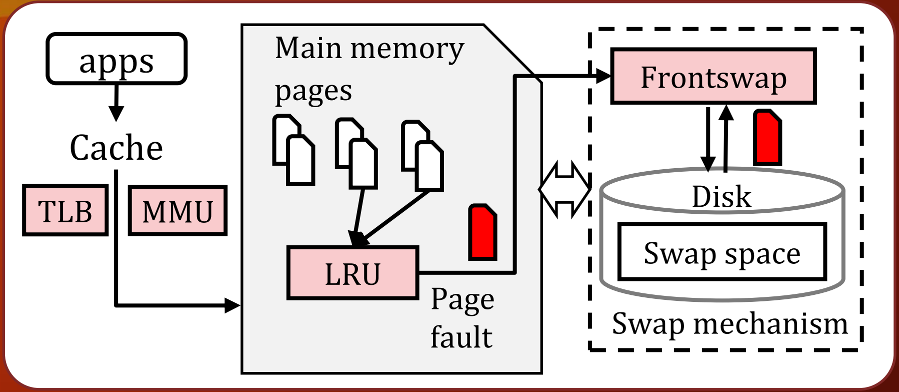
Incorporating multiple FM devices and oversubscribing the PCIe subsystem

Design multi-backend disaggregated memory system

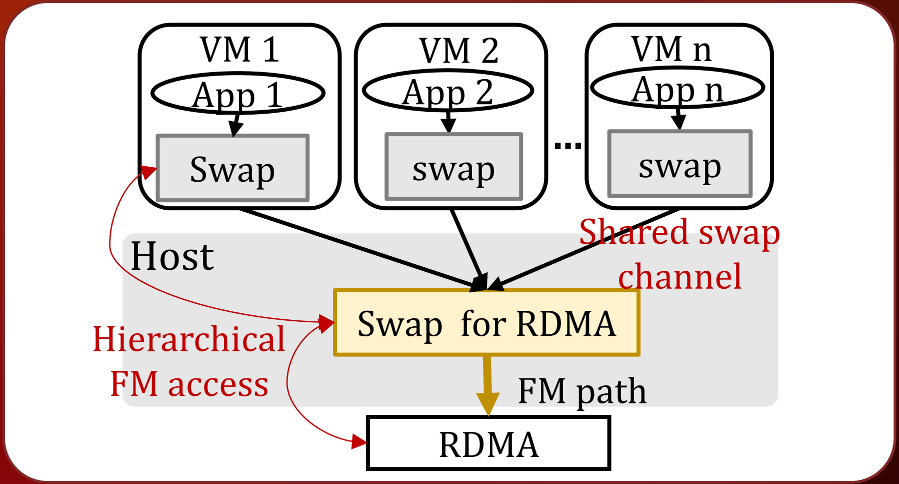
2. Motivation: Design Challenge of Multiple backends

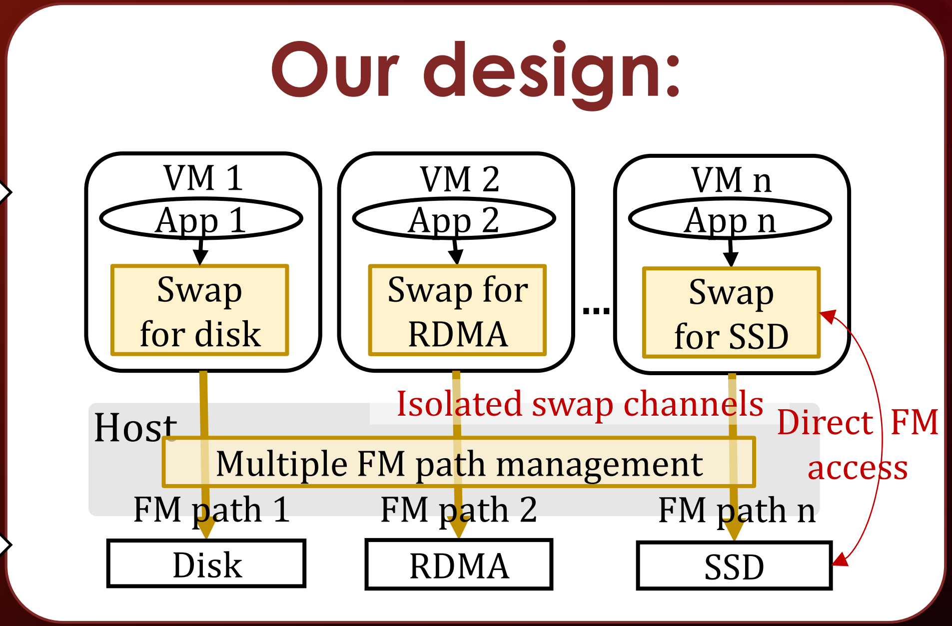
(1). Far Memory Usage Bottleneck: Existing FM management schemes are blind to the possible multiple physical FM channels: logically only support one data exchange path to FM devices.

- Existing works rely on OS-level page swap design that **cannot** allow pages to be swapped to/from **multiple backends**.



- Existing technics with virtual machines (VMs) still use a **hierarchical** data swap mechanism with the host operating system (OS) **involved**.



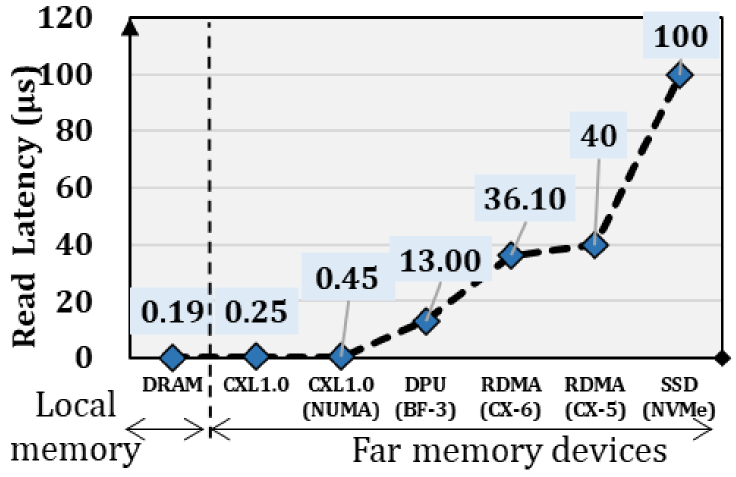


Design Opportunities:

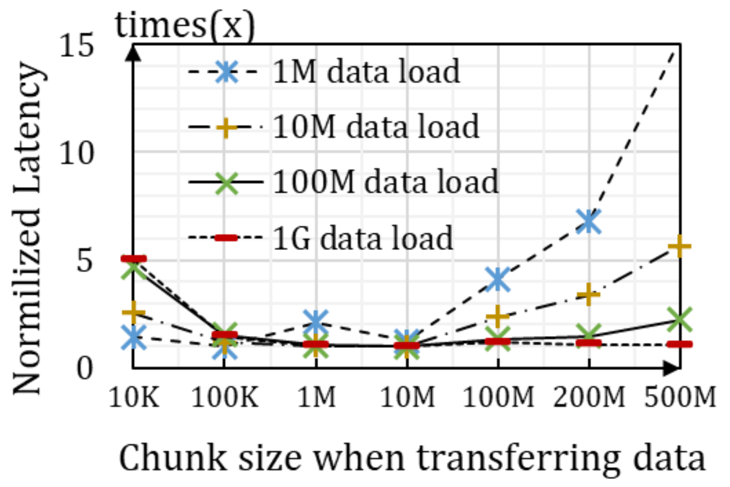
- Allow direct memory/storage access
- Support isolated swap channels
- Support parallel far memory access paths

2. Motivation: Design Challenge of Multiple backends

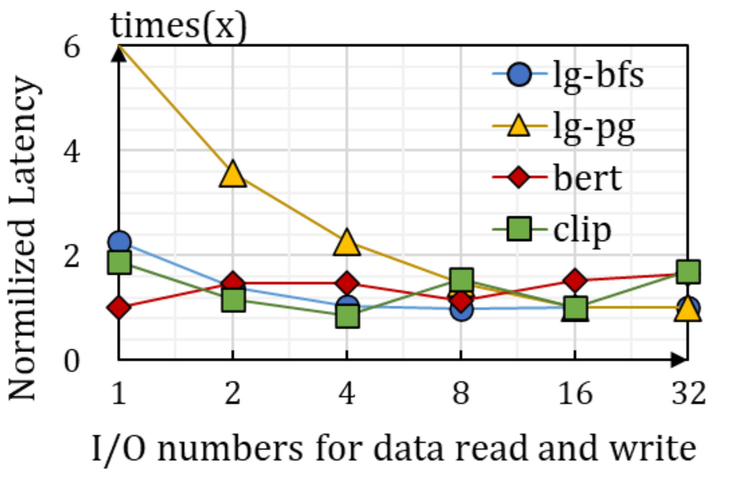
(2). Far Memory Usage Effectiveness: Most of the prior works follow a simple idea: by offloading part of data to far memory based on workload behaviors, ignoring more complex far memory configurations.



Backend differences



Data granularity differences



I/O width differences

Design Opportunities:

- **Resource awareness**: Design a system that can choose proper backends for each application
  -> Lower cost
- **Application awareness**: Providing a system that support multiple-dimensional parameter configuration
  -> Higher performance

“

1. Background
2. Motivation
3. System Design
4. Experimental Result
5. Conclusion and Future Work

”

3. XDM System Design

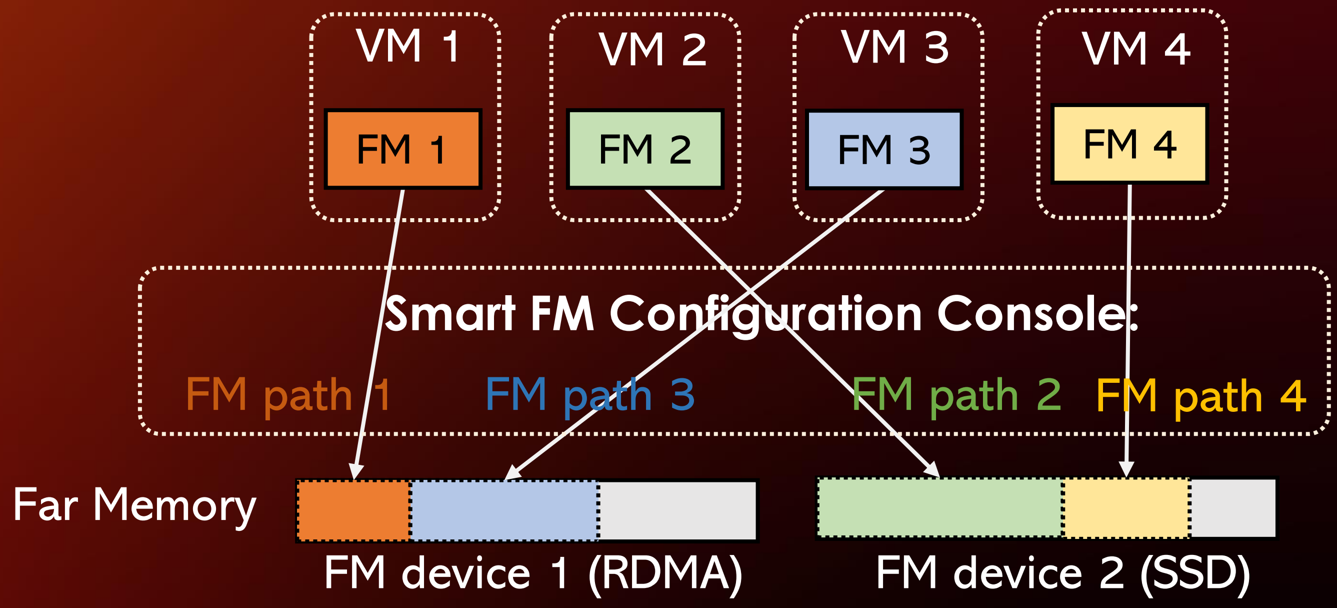
Design philosophy:

(1). Make it dynamic and implicit.
In this work, we aim to make the system **dynamic**: each instance can evaluate task preferences during runtime and implicitly select the **optimal FM path** without the need of user intervention.

(2). Make it versatile and smart.
It is important to leverage a rich set of application **page data** and adjust system settings based on **multi-dimensional** system information, including data distribution, data granularity, as well as I/O characteristics.

We Propose xDM,
an Intelligently Managed
Multi-backend Disaggregated Memory System.

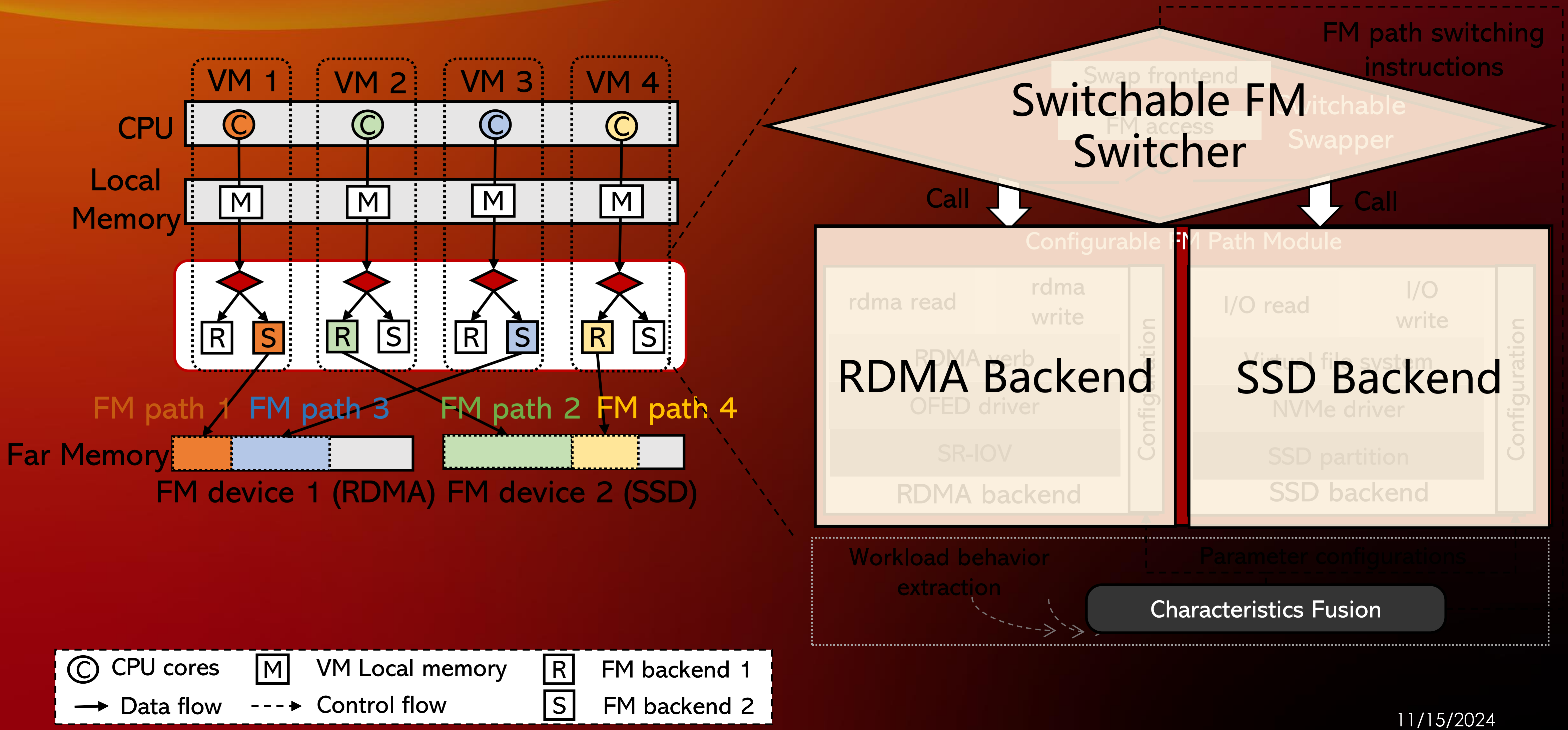
Dynamic FM Switching Mechanism:



3. XDM System Design

3.1 Dynamic FM Switching Mechanism:

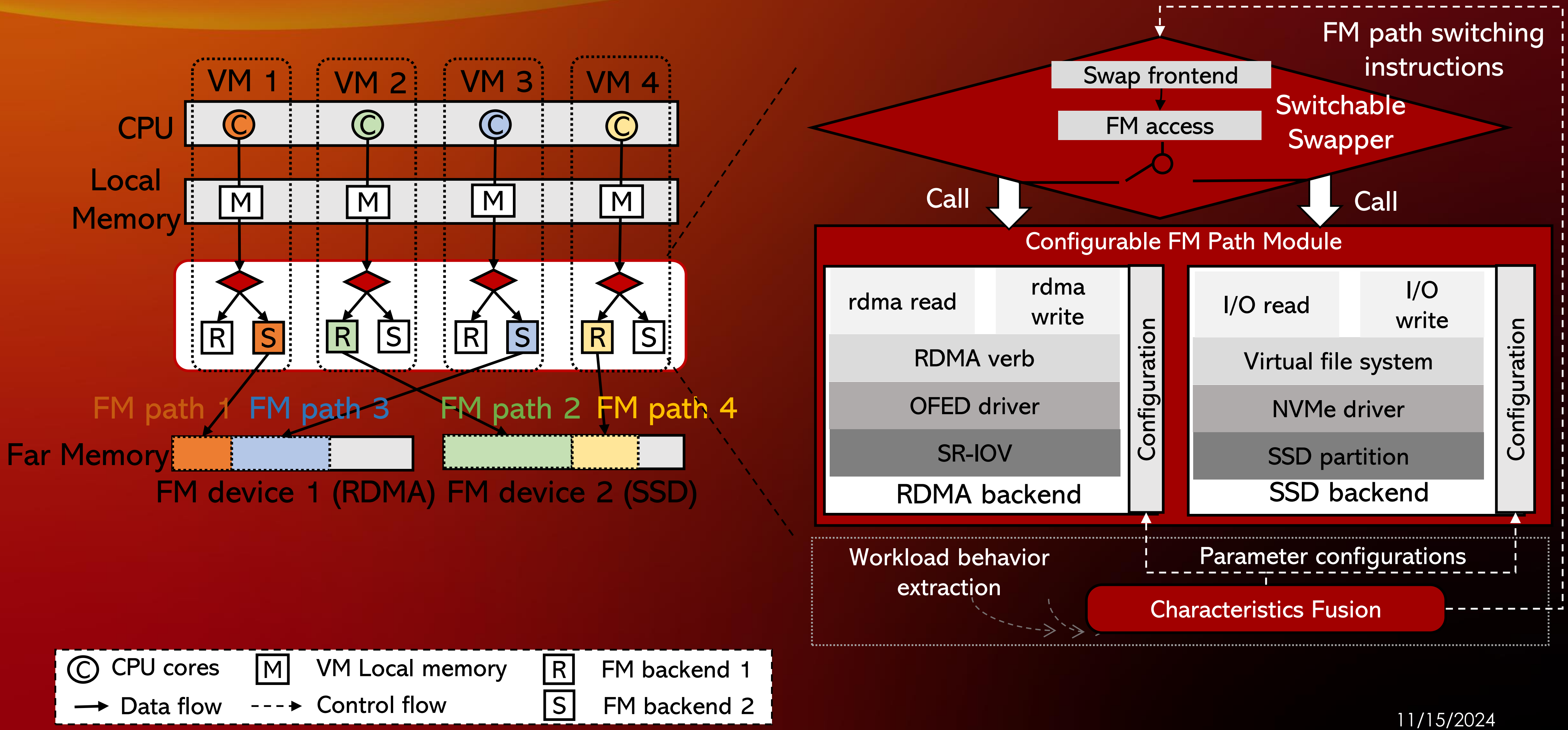
(1). Low-overhead, switchable FM swapper: a modified swap frontend plus a variety of adaptive FM swap backends



3. XDM System Design

3.1 Dynamic FM Switching Mechanism:

(1). Low-overhead, switchable FM swapper: a modified swap frontend plus a variety of adaptive FM swap backends

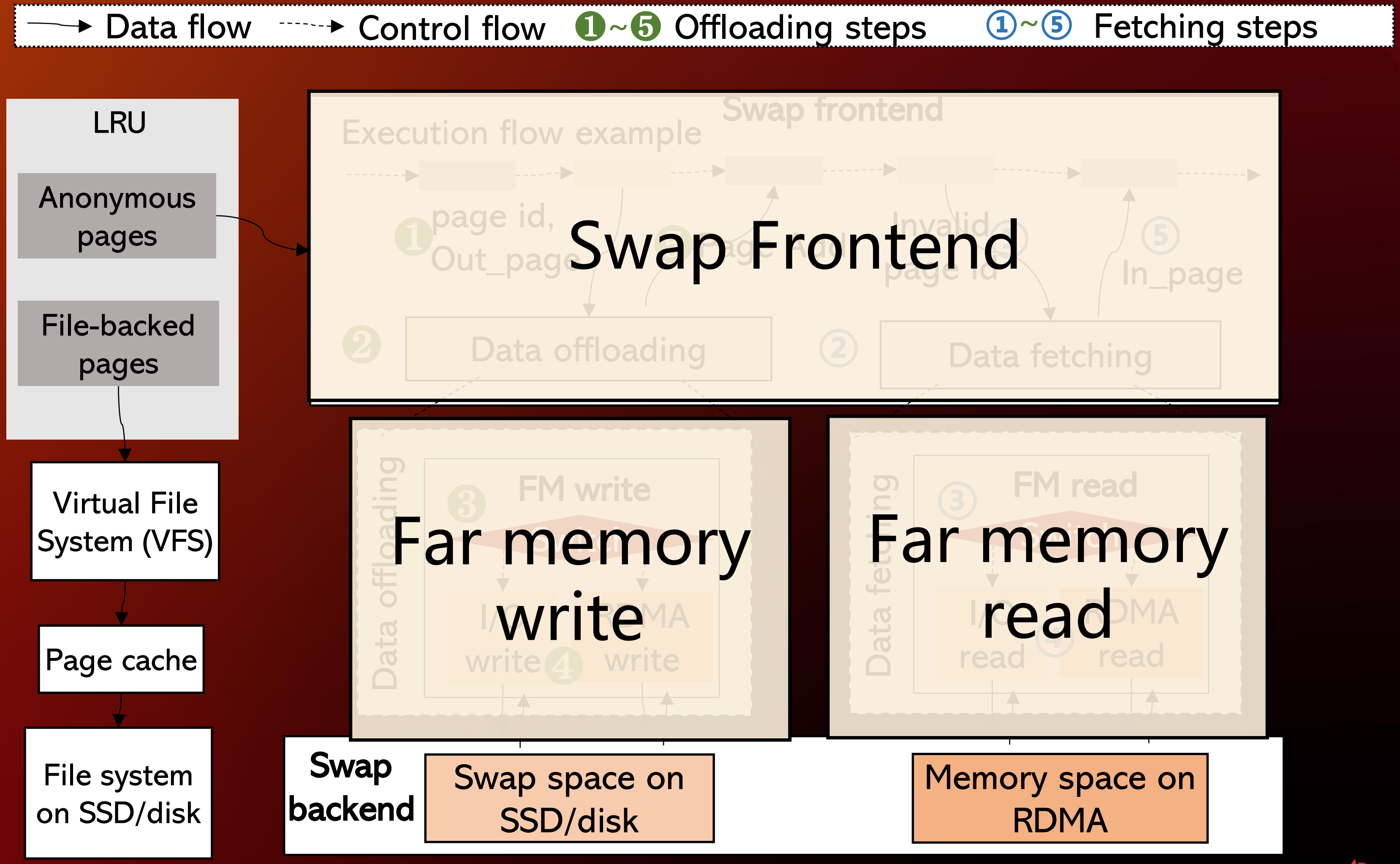


3. XDM System Design

3.1 Dynamic FM Switching Mechanism:

(1). Low-overhead, switchable FM swapper: a modified swap frontend plus a variety of adaptive FM swap backends

- **Swap Frontend:**
  - Out_Page
  - In_page
- **Swap Backend:**
  - Data offloading : far memory write
  - Data fetching: far memory read

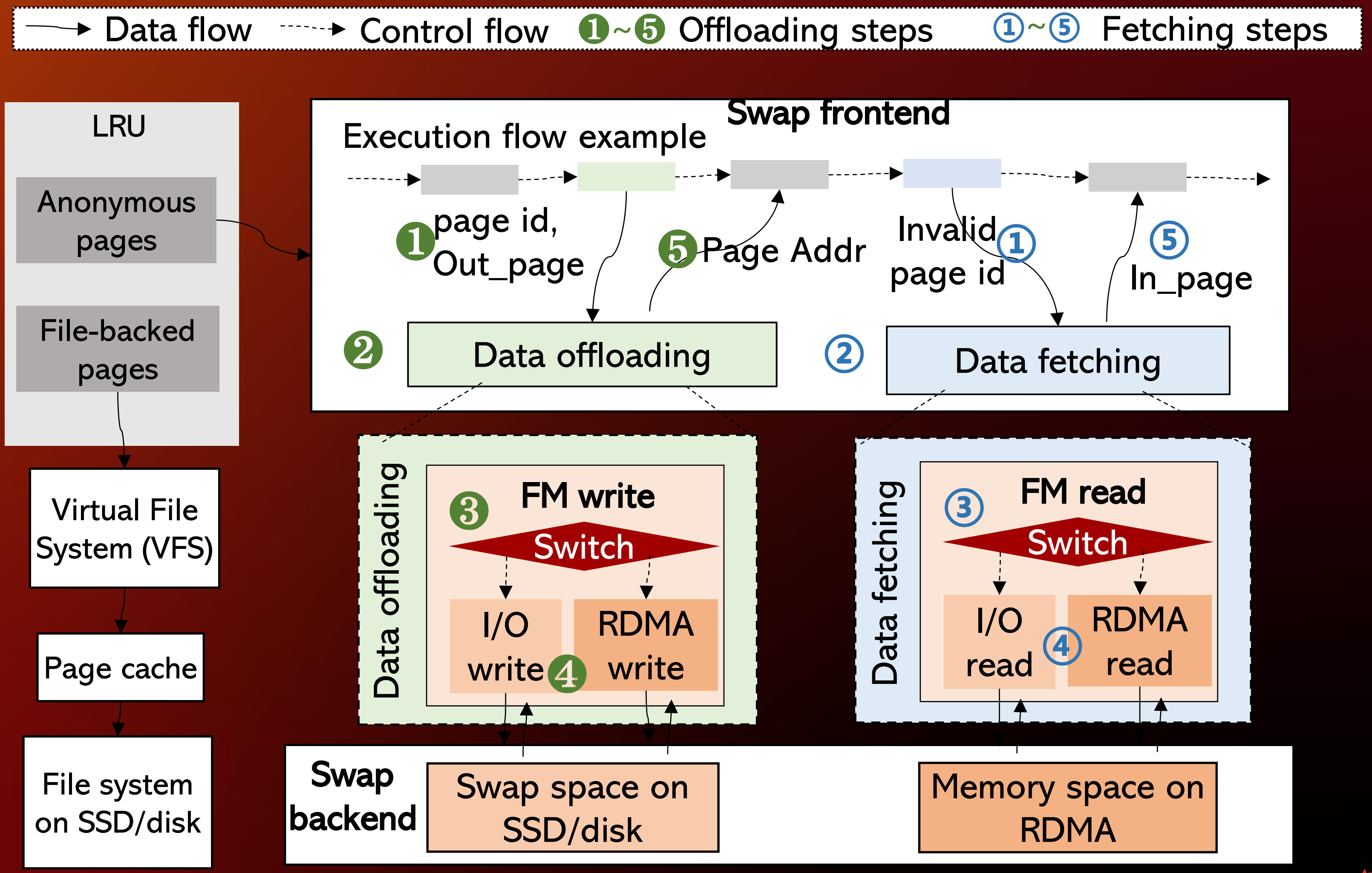


3. XDM System Design

3.1 Dynamic FM Switching Mechanism:

(1). Low-overhead, switchable FM swapper: a modified swap frontend plus a variety of adaptive FM swap backends

- **Swap Frontend:**
  - Out_Page
  - In_page
- **Swap Backend:**
  - Data offloading : far memory write
  - Data fetching: far memory read

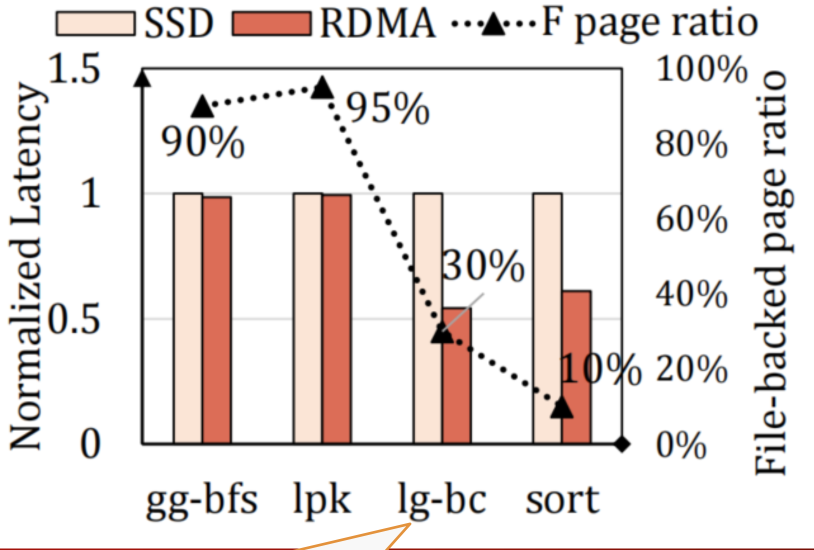


3. XDM System Design

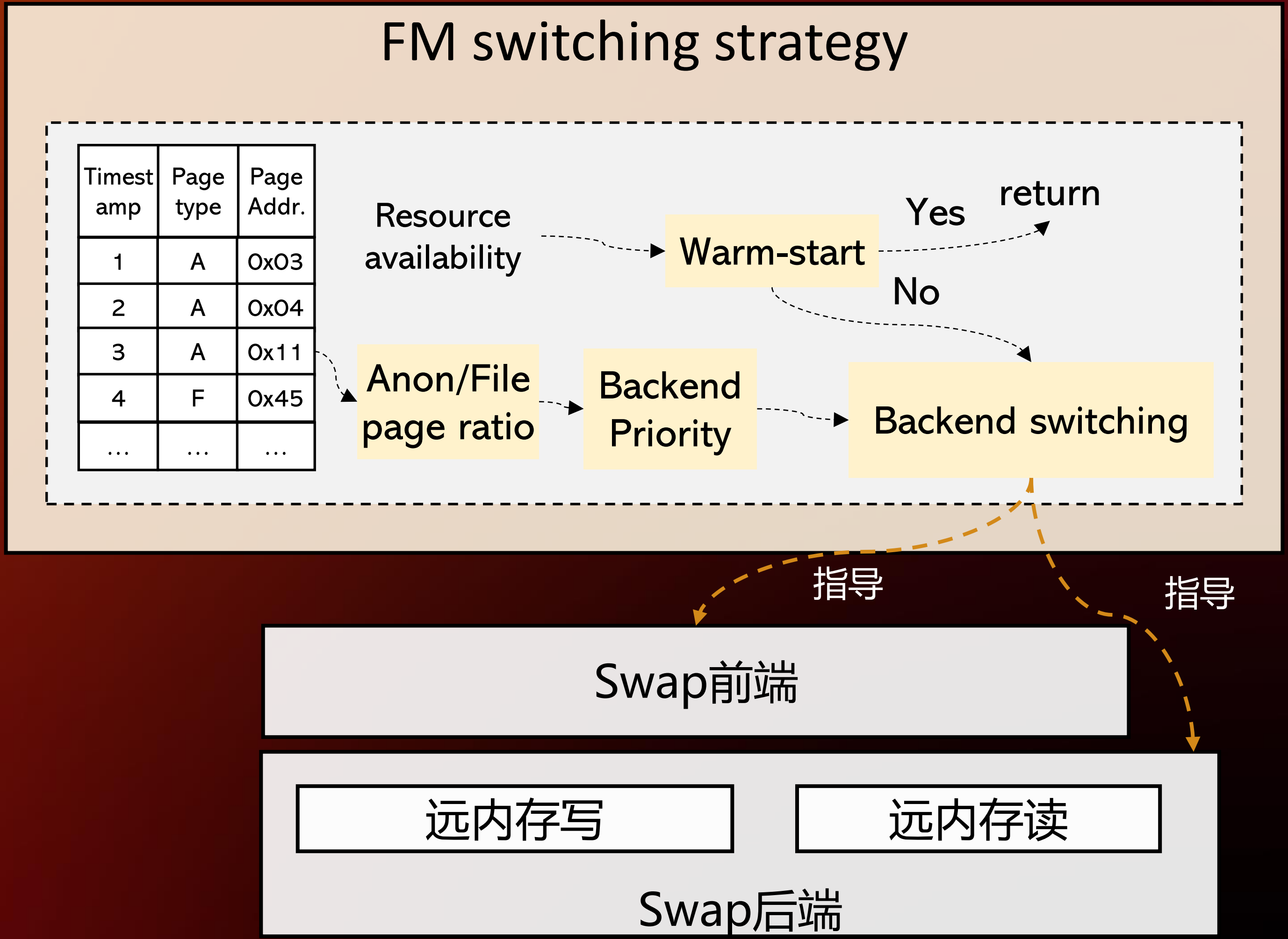
3.1 Dynamic FM Switching Mechanism:

(1). Low-overhead, switchable FM swapper

(2). Efficient, implicit FM switching strategy



Workloads with more file-backed (anonymous) pages prefer SSD (RDMA) backends.



3. XDM System Design

“

The above design includes the basic implements of multi-backend disaggregated memory system,

i.e. enabling **Dynamic FM Switching Mechanism**.

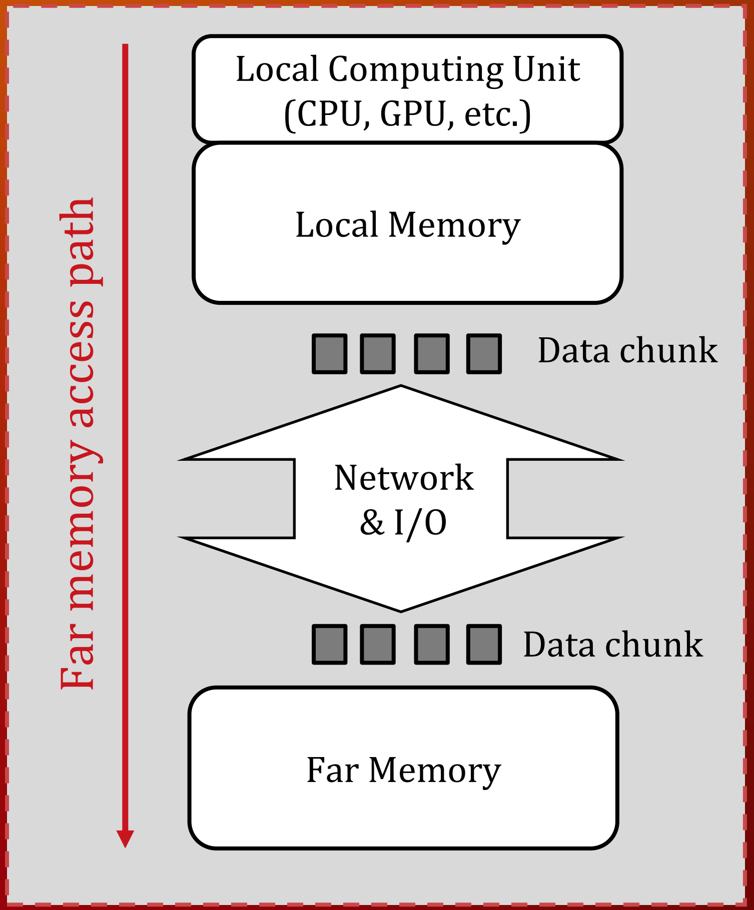
The following configuration design of the far memory access path is to make the best backend usage effectiveness,

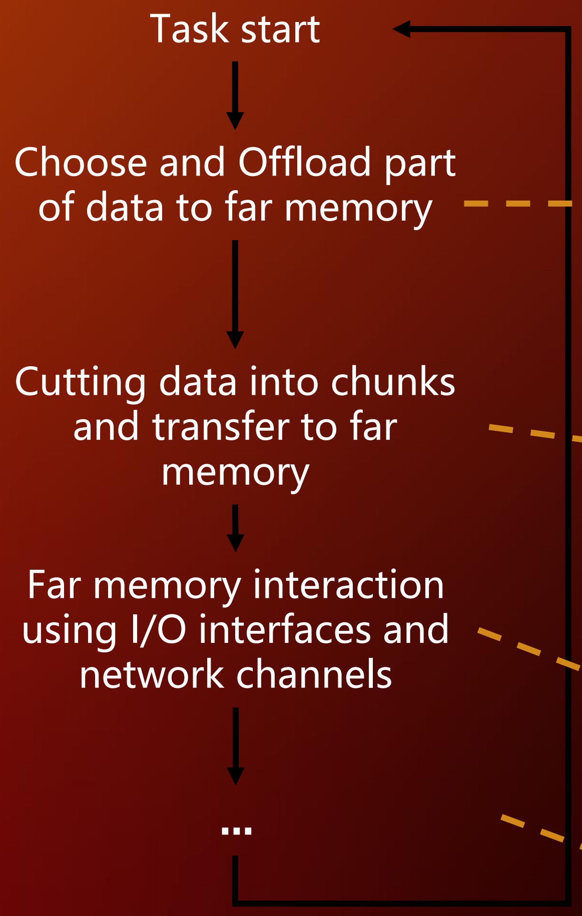
i.e. making **Smart FM Configuration Console.**

”

3. XDM System Design

3.2 Smart FM Configuration Console





Fine-grained data offloading parameters:

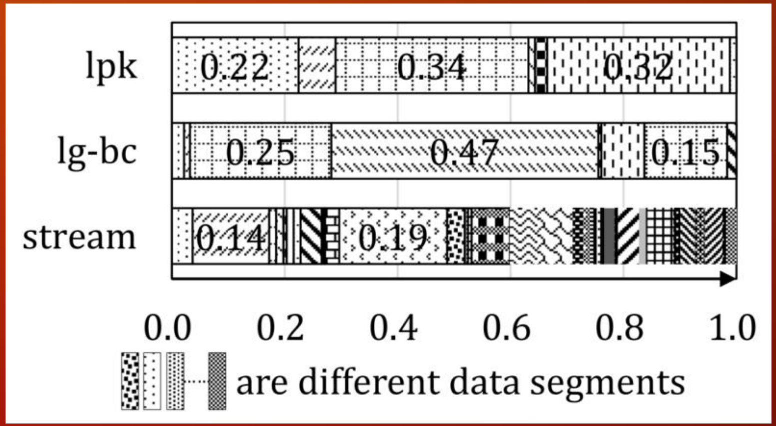
- **Data distribution between local and far memory**
  - **Local ratio**
  - **NUMA ratio**
- **Data granularity when transferring to far memory**
  - **Page size**
  - **Chunk size**
- **Available data bandwidth of far memory paths**
  - **Queue number**
  - **I/O thread**
- **...**

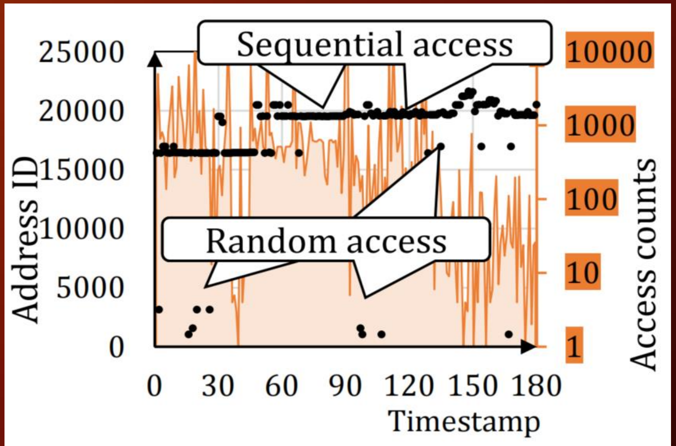
3. XDM System Design

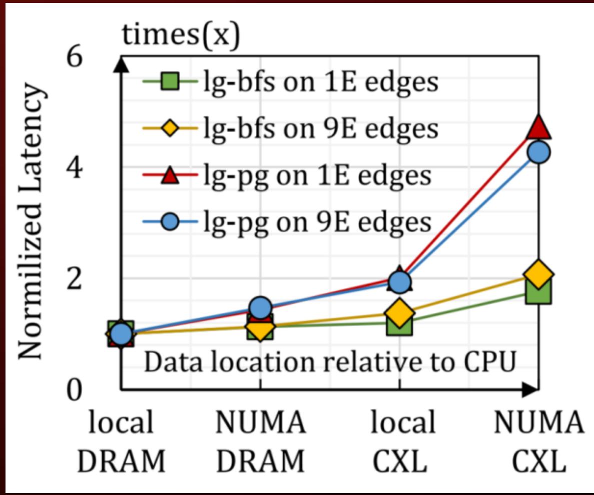
3.2 Smart FM Configuration Console

(1). Data Characteristic Fusion: perceive task characteristics using page-based transparent approach

By analyzing **page** data, we find that data granularity, I/O width, and data distribution significantly impact the performance and resource usage.







Feature 1: Data segments distribution（influenced by data fragment ratios）

Feature 2: Sequential and random page access（influenced by data load/store ratio）

Feature 3: Data distribution（influenced by hot/cold data ratio）

3. XDM System Design

3.2 Smart FM Configuration Console

(1). Data Characteristic Fusion: perceive task characteristics using page-based transparent approach

By analyzing **page** data, we find that data granularity, I/O width, and data distribution significantly impact the performance and resource usage.

Page access behavior

Times tamp	Page type	Page Addr.	Mem. Op.s
1	A	0x03	Load
2	A	0x04	Load
3	A	0x11	Store
4	F	0x45	Load
5	F	0x46	Load
6	F	0x47	Load
…	…	…	…

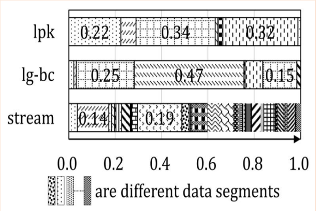
Merge (rows 1–2)

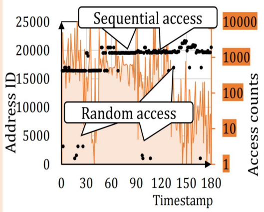
Merge (rows 4–6)

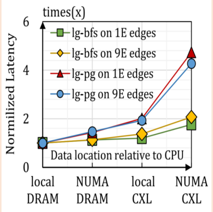
Aggregate

Times tamp	Page type	Page Addr.	Mem. Op.s	Page num.
1	A	0x03	Load	2
3	A	0x11	Store	1
4	F	0x45	Load	3
…	…	…	…	…

Fuse







Data fragment ratio	Load/store data ratio	Hot/cold data ratio
⇩	⇩	⇩
Data Granularity	I/O width	Data distribution

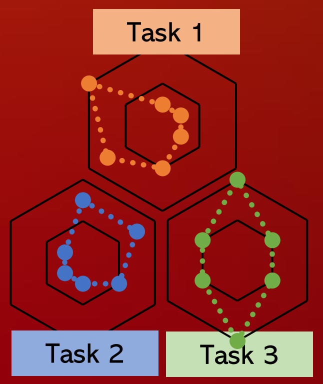
3. XDM System Design

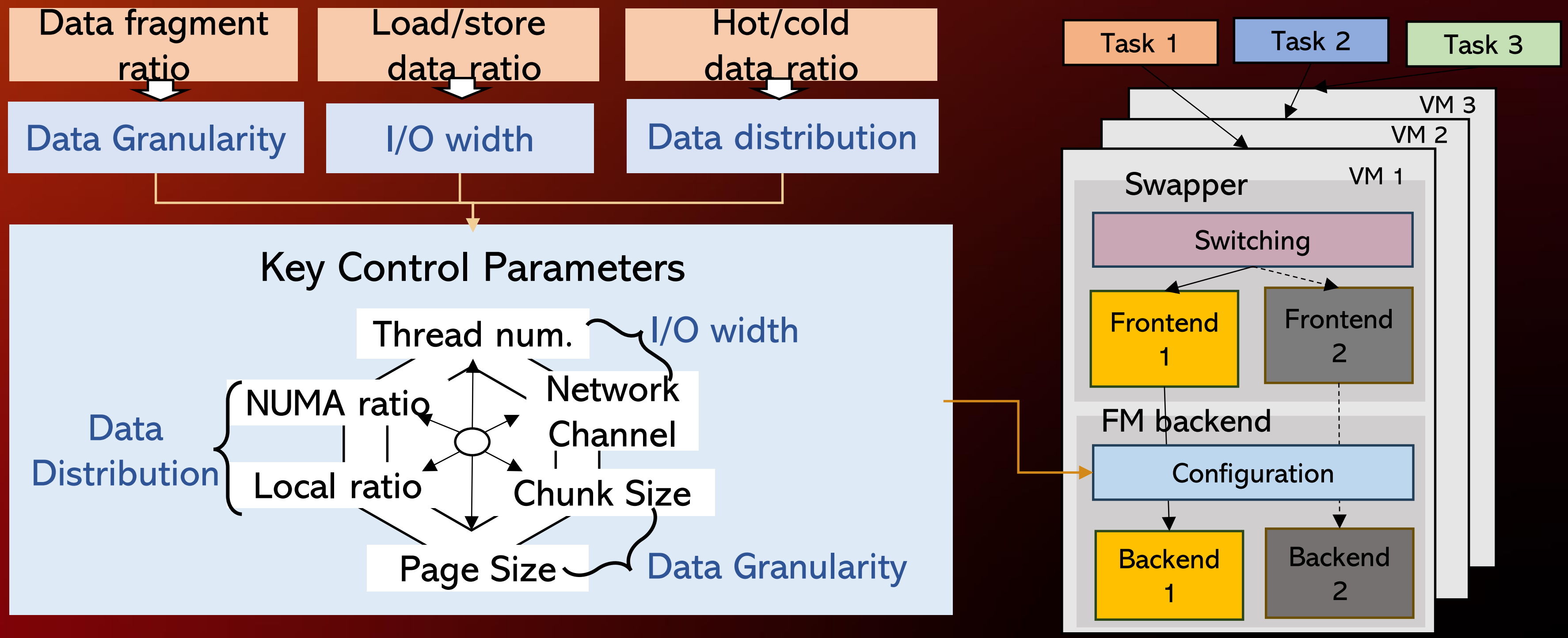
3.2 Smart FM Configuration Console

(1). Data Characteristic Fusion

(2). FM Parameter Adjustment: Parameter configuration shares similar data feature extraction ideas but different implementation methods on far memory backends:

- ***Data granularity:*** size of data units transferred via RDMA (i.e. *chunk size*) or data blocks on SSD (i.e. *page size*).
- ***I/O width:*** assigning *CPU cores* related to I/O channels on SSD and *network channels* of RDMA.
- ***Data distribution:*** adaptively setting *far memory ratio* and *NUMA memory nodes*.





3. XDM System Design

Workflow:

i). Far memory initialization

ii). Offline preparation

iii). VM allocation and warm start

iv). FM path selection and switching

v). FM parameter configuring

Algorithm 1: Multi-backend FM System Workflow.

Input: Application set: A, online VM set: OVs, free VM set: FVs
Result: All applications have been efficiently dispatched

```
for a in A do
    f_a = page_feature_extraction(a)
    b_a = backend_selection(f_a, system_pressure)
    List p_a = parameter_optimization(f_a)
    for Online_VM in OVs do
        if Online_VM.backend = b_a AND Online_VM.accept(a)
         then
            dispatch a → Online_VM
            Online_VM.OptParameters(p_a)
            break
    if no available online VM then
        for Free_VM in FVs do
            if Free_VM.backend = b_a AND Free_VM.accept(a)
             then
                dispatch a → Free_VM
                Free_VM.OptParameters( p_a )
                break
    else if no available idle VM with b_a then
        Free_VM← SelectVM(FVs)
        Free_VM.SwitchBackend(b_a)
        Free_VM.OptParameters( p_a )
        dispatch a → Free_VM
    else if no available idle VM AND host resource is available
     then
        Free_VM← CreateVM(b_a, system_pressure)
        Free_VM.OptParameters(p_a)
        dispatch a → Free_VM
        add Free_VM→ Vs
```

“

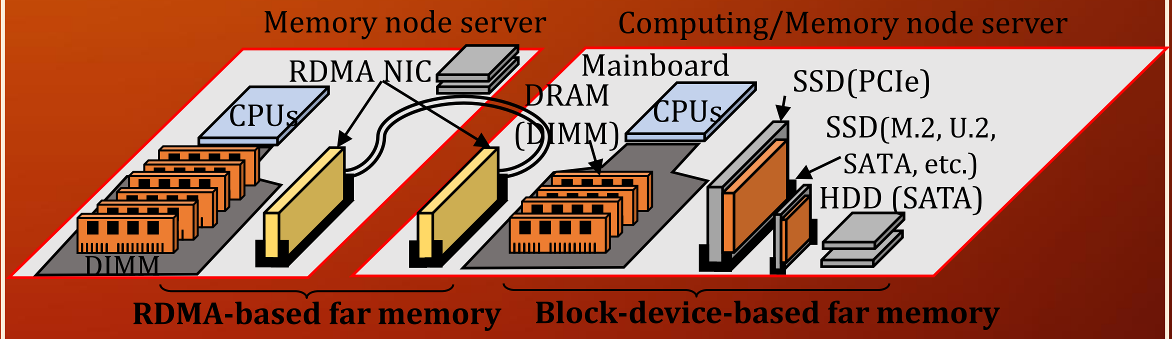
1. Background
2. Motivation
3. System Design
4. Experimental Result
5. Conclusion and Future Work

”

4. Experimental Result

Hardware and Software Testbed

Physical machine:



Conf.	Tools
Memory limtation	Cgroup2
RDMA driver	OFED v4.3.0,RoCE
RDMA far memory	Fastswap
CXL far memory	NUMA simulation
SSD far memory	VFS I/O

Evaluated 17 types of workloads

HPC workloads, graph workloads, AI workloads

Type	Abbr.	Algorithm Description	Max Mem.
HPC workloads	stream	Stream [52] for memory bandwidth	4G
	lpk	Linpack [53] for floating-point computing	4G
	kmeans	K-means clustering on sklearn [48]	4G
	sort	Quicksort [53] on c++ std	8G
	sp-pg	Page rank on Spark [2]	10G
Graph workloads	gg-pre	Graph preprocess on GridGraph [47]	16G
	gg-bfs	Breadth-first search on GridGraph [47]	16G
	lg-bfs	Breadth-first search on Ligra [1]	16G
	lg-bc	Betweenness centrality [1]	16G
	lg-comp	Connected components [1]	16G
	lg-mis	Multiple importance sampling [1]	16G
AI workloads	tf-incep	Resnet inception on Tensorflow [45]	1G
	tf-infer	Resnet inference on Tensorflow [45]	1G
	tf-tc	CNN inference on text classification [46]	10G
	bert	Inference on Bert [7]	1.5G
	clip	Inference on Clip [6]	1.7G
	chat-int	Inference on ChatGLM [5] (int4)	14G

4. Experimental Result

Baseline configurations:

Related works	Far memory	Max BW	FM size
Linux swap [42]	disk	2 GB/s	2T
TMO [37]	SSD	7.9 GB/s	1T
Fastswap [27]	RDMA	10 GB/s	256G
XMemPod [40]	DRAM or RDMA	10 GB/s	1T
xDM-SSD	multiple SSD	32 GB/s	1T
xDM-RDMA	multiple RDMA	32 GB/s	256G
xDM-Hetero	RDMA and SSD	32 GB/s	1.3T

Tunable FM parameters in our system:

Parameter	Offline Conf.	Online Conf.	Scale
Total CPU core	Yes	No	$\leq$ Total CPU cores
Local memory size	Yes	No	$\leq$ Server memory size
NUMA memory	Yes	No	Different NUMA nodes
Far memory ratio	Yes	Yes	$0 \sim 0.9$
Page size	Yes	Yes	4K $\sim$ 2M on average
Network channel	Yes	Yes	$\leq$ Total I/O channels

4. Experimental Result

Functional Comparison:

Related works	to Block Device	to RDMA	Hybrid	Multi-path
Linux zswap [42]	✓	×	×	×
Fastswap [27]	×	✓	×	×
TMO [37]	✓	×	✓	×
XMemPod [40]	✓	✓	✓	×
Pond [31]	✓	×	×	×
xDM (Ours)	✓	✓	✓	✓

TABLE I: Single-path vs. multi-path far memory systems.

Our System can support parallel multi-path far memory access.

Related works	Data Ratio on FM	Data Ratio on NUMA	Data Granularity	I/O Width
Linux zswap [42]	✓	×	×	×
Fastswap [27]	✓	×	×	×
TMO [37]	✓	×	×	×
XMemPod [40]	✓	×	×	×
Pond [31]	✓	✓	×	×
xDM (Ours)	✓	✓	✓	✓

TABLE II: Comparison of key tuning knobs of far memory configuration used in related works.

Our system add more dimensions of system parameter analysis and configuration.

4. Experimental Result

Evalauted Workload	stream	lpk	kmeans	sort	s-pg	gg-pre	gg-bfs	lg-bfs	lg-bc	lg-comp	lg-mis	tf-infer	tf-incep	clip	tf-tc	chat-int	bert
Swap Feature	S	S	S	S	S	F	S	F	F	F	F	F	F	S	F	F	S
Sp. on DRAM	**1.32×**	1.18×	**1.64×**	1.05×	**1.44×**	**2.24×**	**1.29×**	2.00×	2.16×	**2.43×**	**2.17×**	1.88×	1.72×	0.82×	1.28×	1.15×	1.03×
Sp. on SSD	1.01×	**1.52×**	0.88×	0.86×	1.01×	1.02×	1.18×	1.40×	1.42×	1.52×	1.36×	1.51×	1.34×	0.91×	2.16×	1.92×	**1.75×**
Sp. on RDMA	1.25×	1.09×	1.40×	**1.40×**	1.37×	2.06×	1.19×	**2.24×**	**2.26×**	2.22×	2.07×	**2.70×**	**2.53×**	**2.46×**	**2.55×**	**3.89×**	1.10×
Average Speedup	1.19×	1.26×	1.31×	1.11×	1.28×	1.77×	1.22×	1.88×	1.95×	2.05×	1.86×	2.03×	1.86×	1.40×	2.00×	**2.32×**	1.29×

TABLE VI: The swap performance speedup (Sp.) of our xDM compared ... lines include Linux swap [42] on SSD backend, Fastswap [27] on RDMA and D... ation swap features into two types: swap-sensitive (S, average Sp.≤ 1.5×) and swap-friendly (F, average Sp.≥ 1.5×).

Under same Latency, swap performance speedup is **~3.9x**

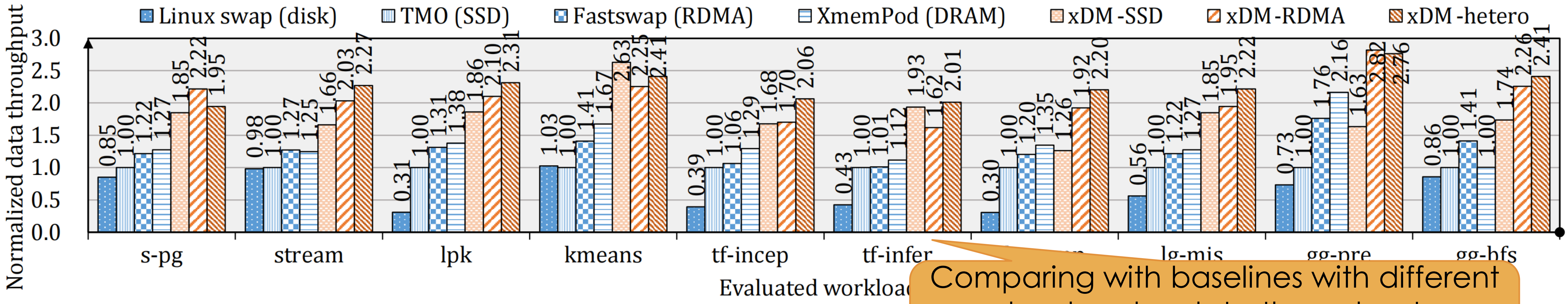
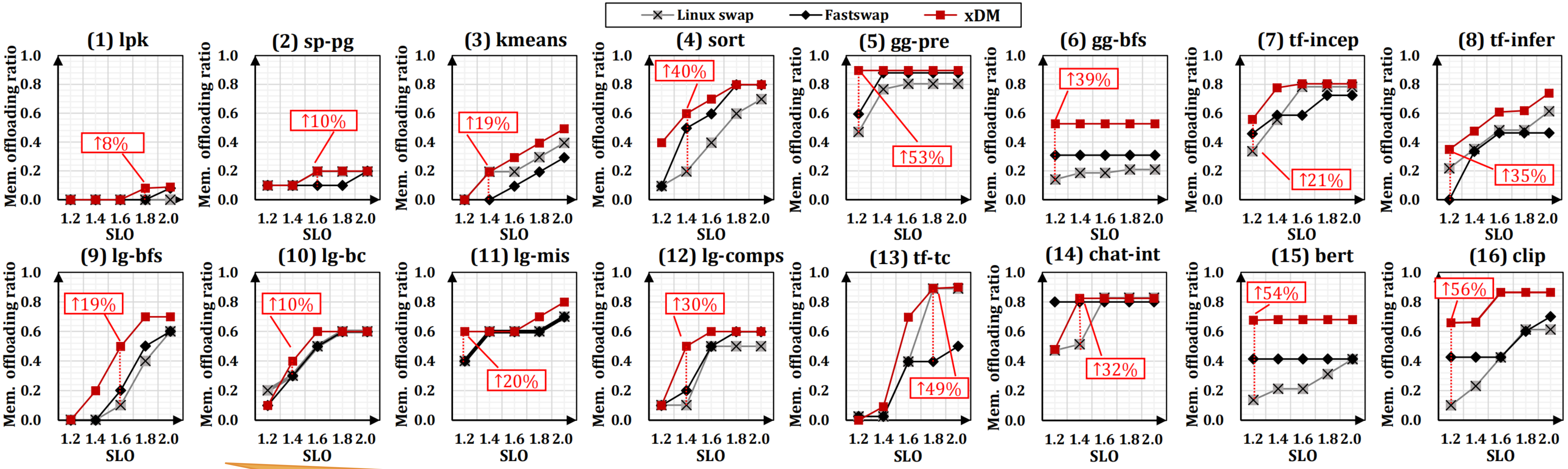


Fig. 14: Our design shows larger data throughput than baselines on evalu... nds.

Comparing with baselines with different backends, data throughput improvement is up to **2.8x**

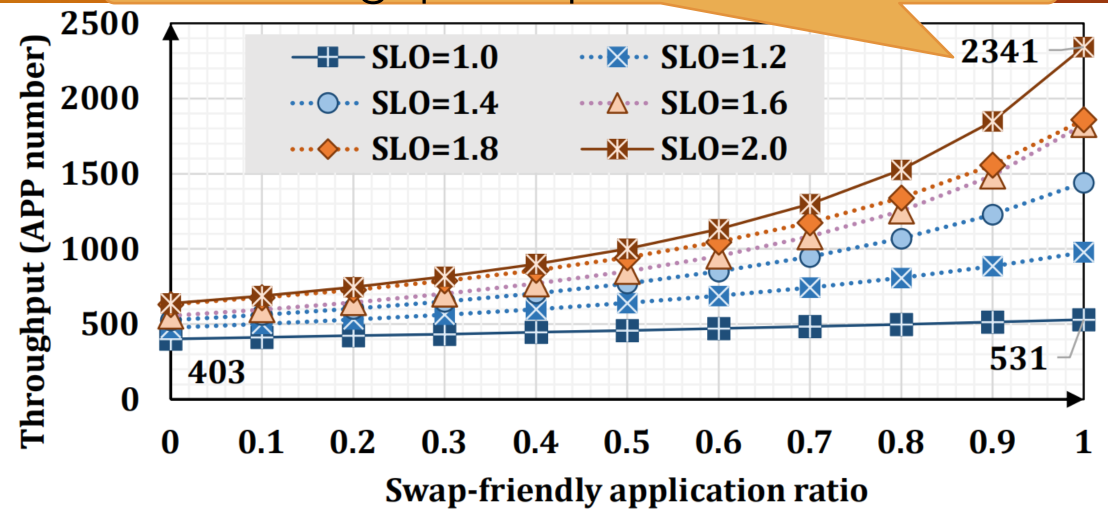
4. Experimental Result

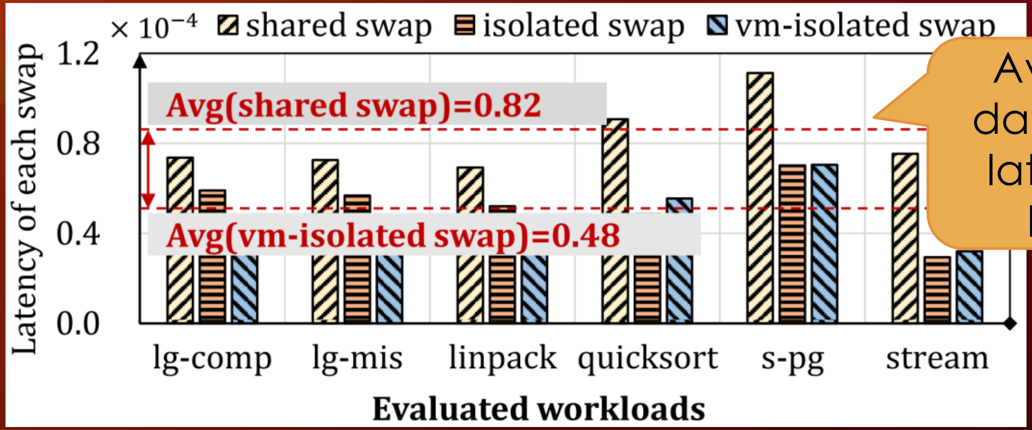


Under same latency (SLO), Our system can have larger offloadable data ratio, saving up to **5x** local memory resource.

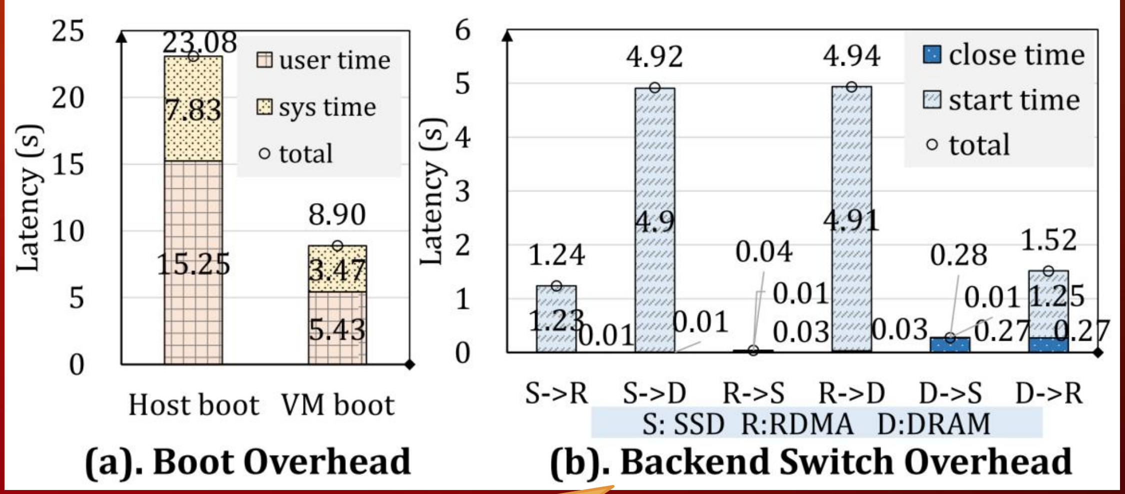
4. Experimental Result

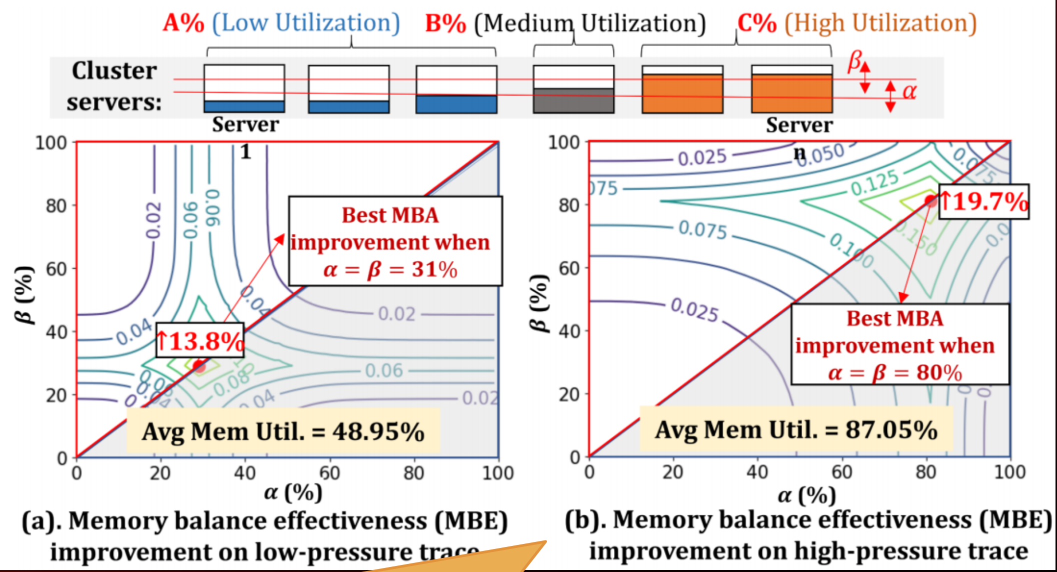
Task throughput improvement is ~**5.1x**





Average data swap latency is lower





Overheads reduction of backend switching is **2.6x**

Simulated memory effectiveness improvement is **19.7%**

“

1. Background
2. Motivation
3. System Design
4. Experimental Result
5. Conclusion and Future Work

”

5. Conclusion and Future Work

Take away message:

- 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 **fusion of application page data**, we tailor the far memory path configurations to the needs of various applications.
- Our design provides a **flexible solution** to scale out far memory access paths and an efficient way to manage them on monolithic servers.

Available on Github: https://github.com/linqinluli/Multi-backend-DM

Future works:

- Data compression on far memory
- Hardware-aid data temperature detection
- High performance data caching and indexing design
- Multi-path GPU far memory system

“

THANK YOU!

QUESTION AND ANSWERING

Jing Wang, contact me at jing618@sjtu.edu.cn

Shanghai Jiao Tong University

”