SHANGHAI JIAO TONG UNIVERSITY

HyFarM: Task Orchestration on Hybrid Far Memory for High Performance Per Bit

Jing Wang, Chao Li, Junyi Mei, Hao He, Taolei Wang, Pengyu Wang, Lu Zhang, Minyi Guo, Hanqing Wu, Dongbai Chen, Xiangwen Liu

Shanghai Jiao Tong University*，*Shanghai Qi Zhi Institute, Alibaba Cloud

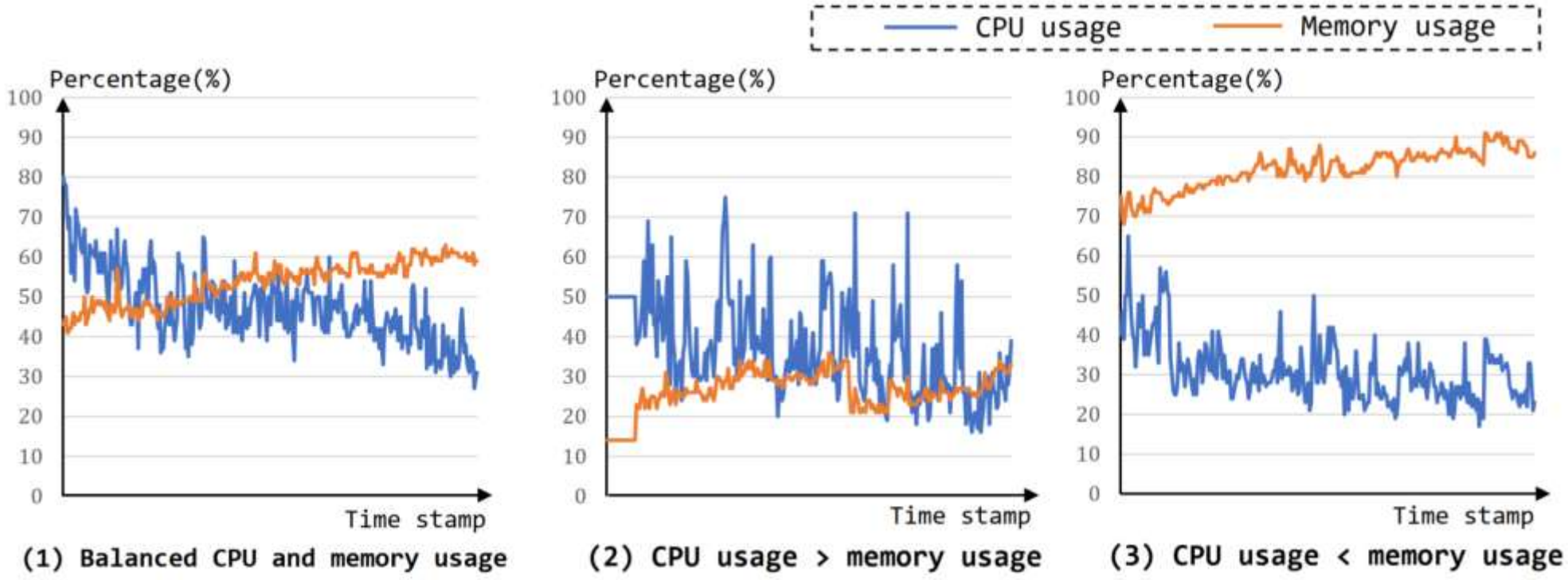
jing618@sjtu.edu.cn

Outline

- **Background**
- Challenge and Motivation
- HyFarM Design
- Evaluation
- Conclusion

Background: Cluster Utilization

- **Resource utilization imbalance in today's data center (Alibaba trace data)**



On some machines, the idle CPU and memory resource stays meaningless for a long period of time.

[1] Analyzing alibaba's co-located datacenter workloads, IEEE Big Data'18

Background: Disaggregated Architecture

Rack-scale Far Memory Expansion:

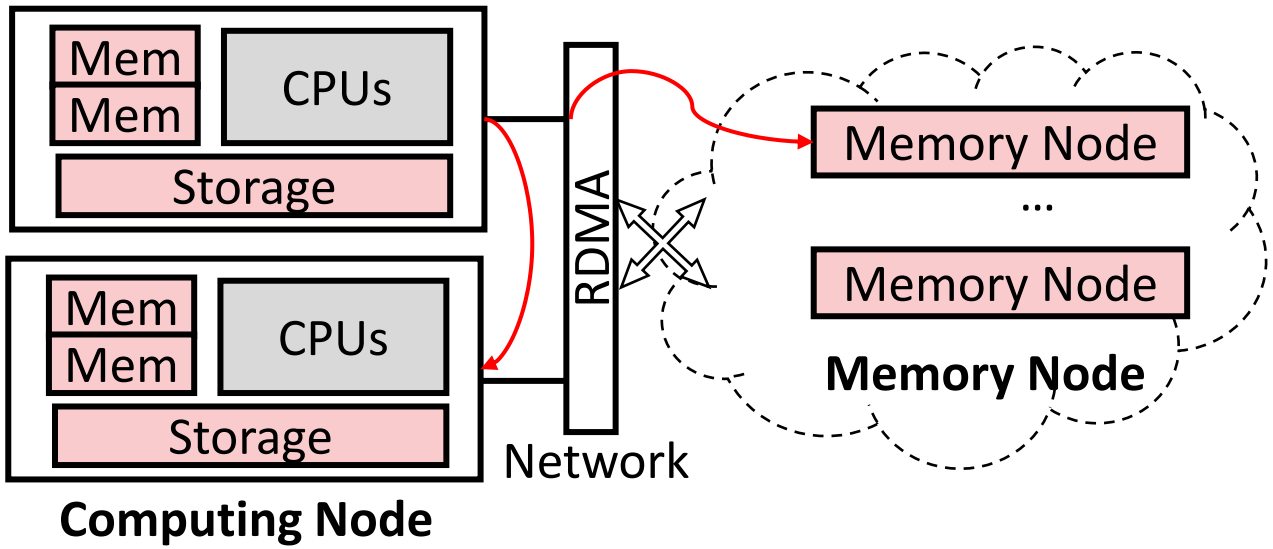
- Build a memory pool with large enough memory capacity (as an memory node).
- Memory node holds oversubscribed memory capacity of the computing nodes in the cluster.
- Data-intensive applications can borrow underutilized memory from remote nodes.
- Improve task performance and memory efficiency, save cost for data centers.

Opportunities:

- Large memory capacity
- Scalability and elasticity

Problems:

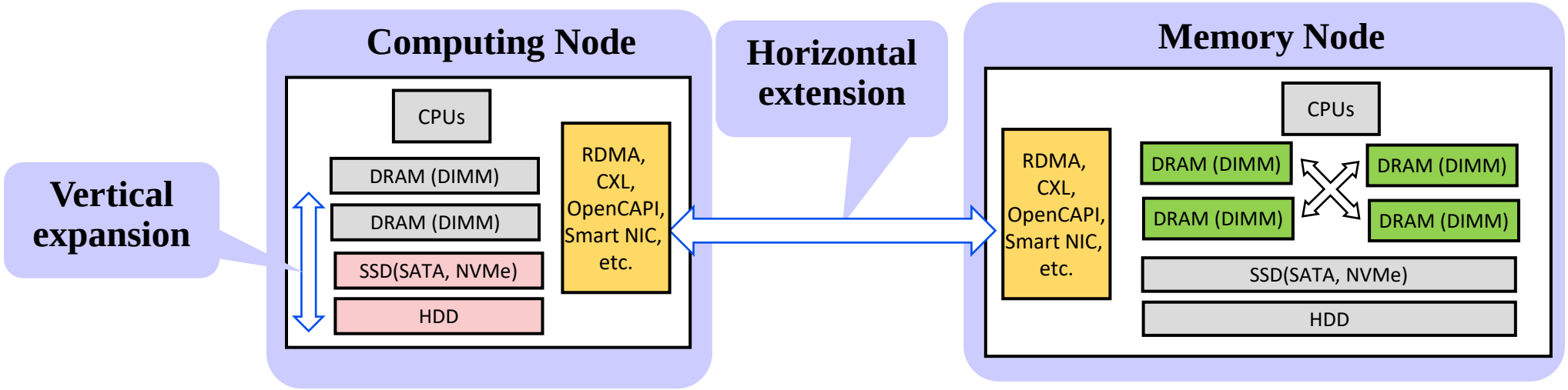
- Task Performance on far memory
- Memory resource orchestration



[1] Disaggregated Memory for Expansion and Sharing in Blade Servers, ISCA'09

Background: Hybrid Far Memory

- **There are two main destinations of memory expansion ways for applications.**



Vertical far memory

→ Higher I/O latency
→ Less Data transfer bandwidth
→ Fixed capacity of local storage
→ Swap cold data passively

Horizontal far memory

→ Fast Remote memory access
→ Large Data transfer bandwidth
→ Flexible memory capacity
→ Offloading data passively or actively

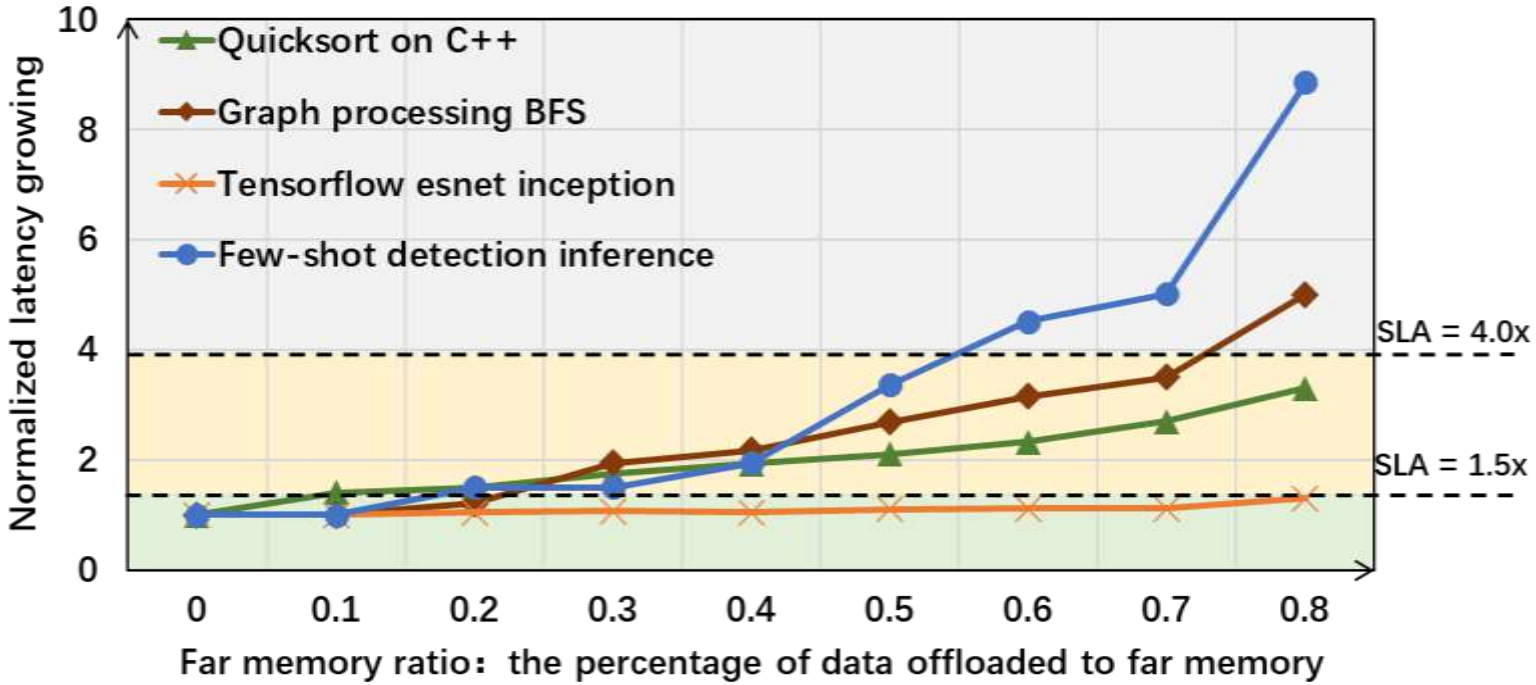
Combining horizontal and vertical FM would provide a better design trade-off.

[1] Tmo: transparent memory offloading in datacenters, ASPLOS'22
[2] Clio: A hardware-software co-designed disaggregated memory system , ASPLOS'22

Background: Far memory Sensitivity

Far memory sensitivity of tasks:

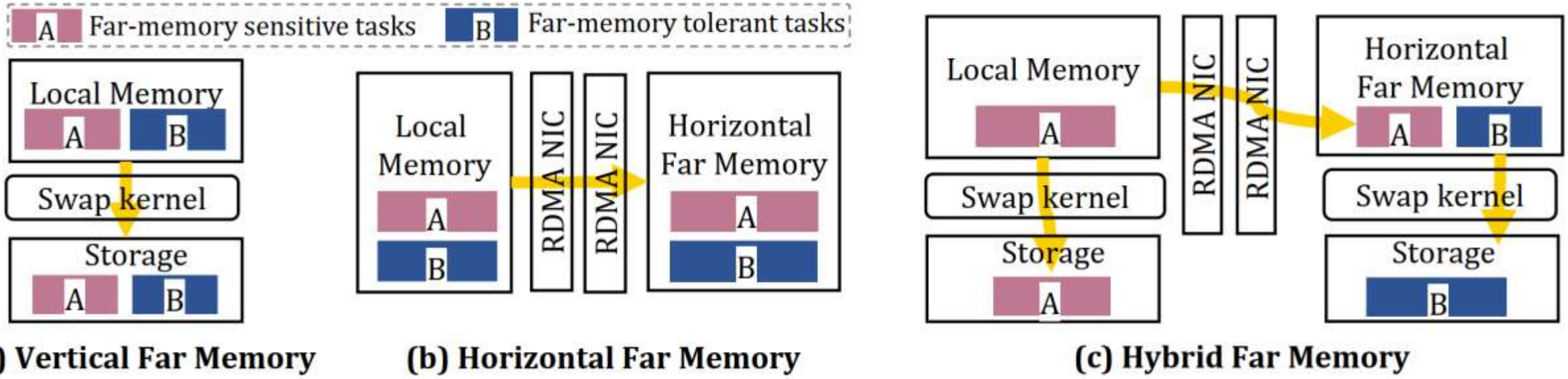
- FM-sensitive tasks slow down obviously when adding a little far memory usage ratio .
- FM-tolerant tasks have tolerable performance downgradation on large amount of far memory.



Analyzing tasks sensitiveness is essential when allocating far memory resources.

Background: Task placement on Hybrid Far Memory

It is important to keep more critical data to be retained in the local main memory.



(a) Vertical Far Memory **(b) Horizontal Far Memory** **(c) Hybrid Far Memory**

Existing strategies blindly squeeze the memory usage of the existing tasks:

- *Last-in-first-cap(LIFC): C*apping the memory footprint of the most recent scheduled tasks
- *Last-in-first-cap(FIFC):* Capping the memory footprint of the earliest tasks

Data center needs to understand task memory access patterns and adapt quickly to the environment.

[1] Can far memory improve job throughput?, EuroSys'20
[2] Software-defined far memory in warehouse-scale computers, ASPLOS'19

Outline

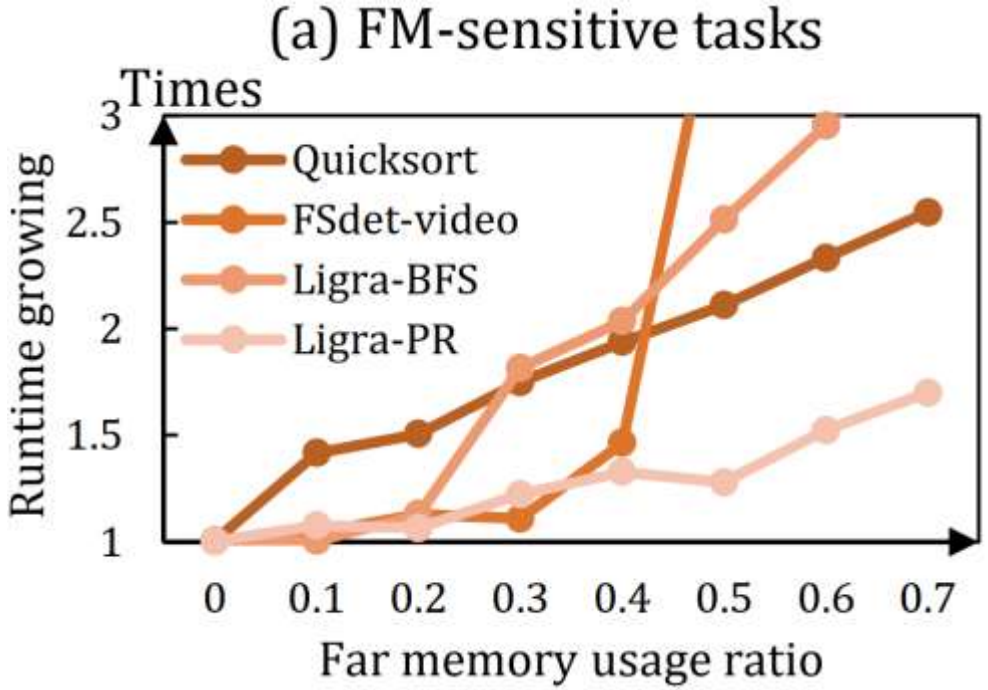
- Background
- **Challenge and Motivation**
- HyFarM Design
- Evaluation
- Conclusion

Challenge: Different Sensitivity

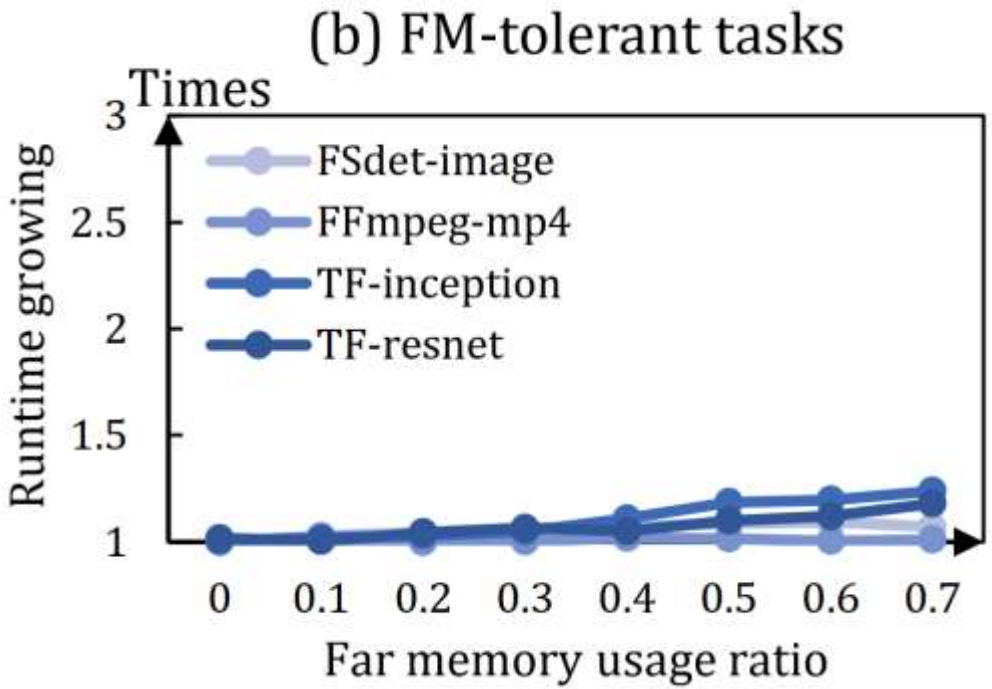
We run applications on far memory and analyze the behavior of task durations.

- **Observation 1:**

Depending on the application type and dataset size, some tasks are more sensitive to FM access.



Latency grows significantly when limiting local memory size.



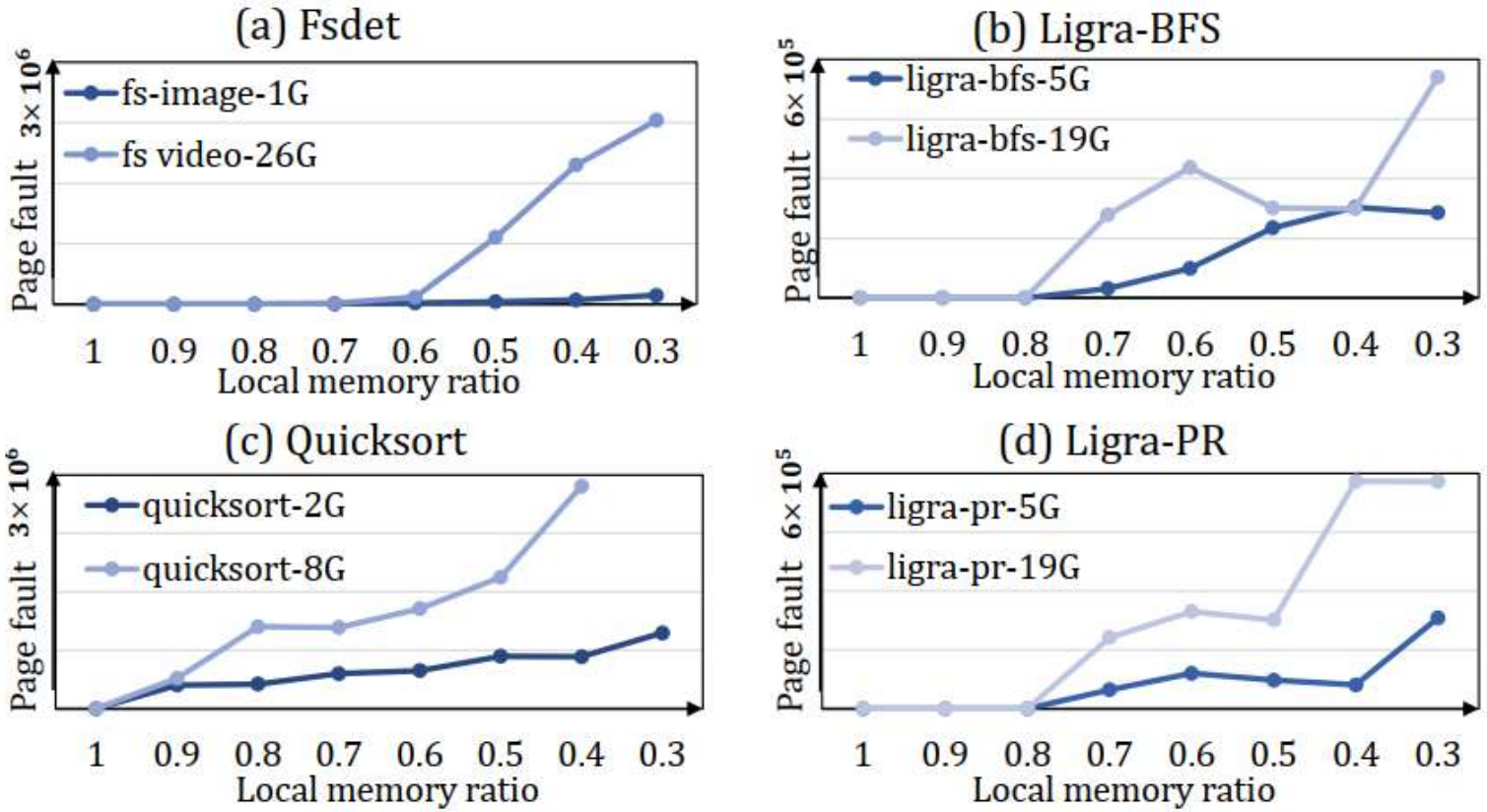
Most of data can be offloaded to FM with small performance change.

Challenge: Different Sensitivity

We collect page fault behavior of FM-sensitive tasks on far memory.

- **Observation 2:**

  FM sensitiveness of tasks can be presented by detecting page fault behavior.
  Task dataset size also affect the FM sensitiveness of tasks.

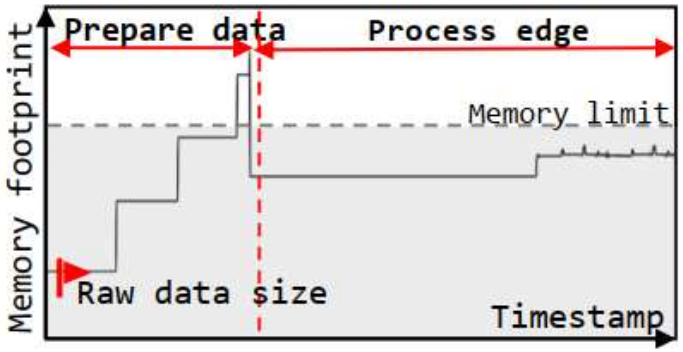


Challenge: Changeable Sensitivity

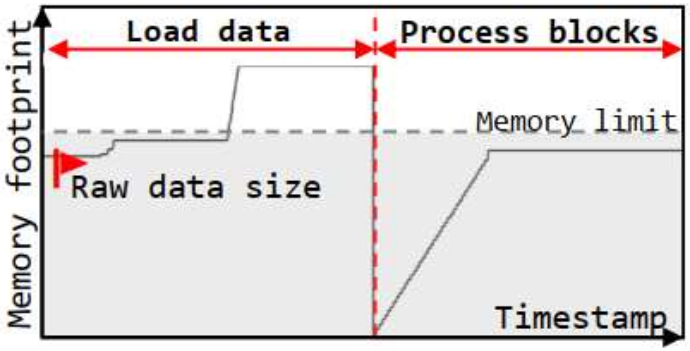
- **Observation 3:** A task may have different sensitivity as its phase changes.

  → it can be problematic to make static FM scheduling decisions considering the dynamicity of applications.

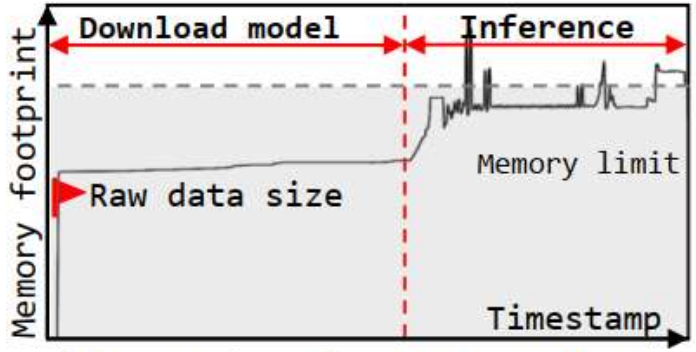
Changed Sensitivity



(a) Pagerank on Ligra

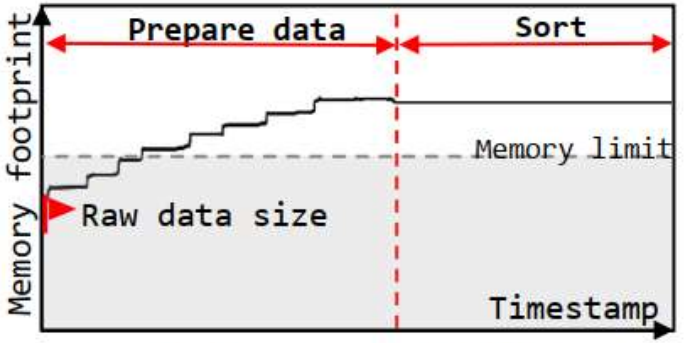


(b) BFS on Gridgraph

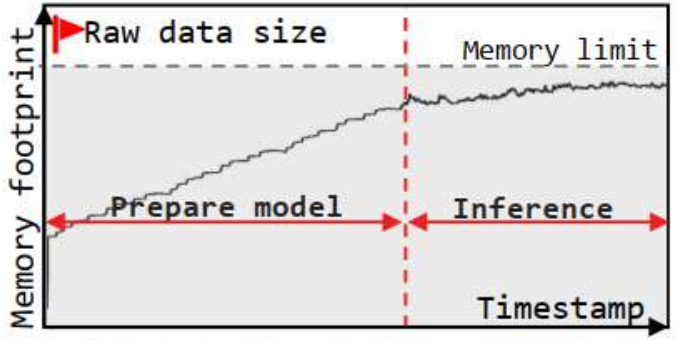


(c) Few-short detection on FsDet

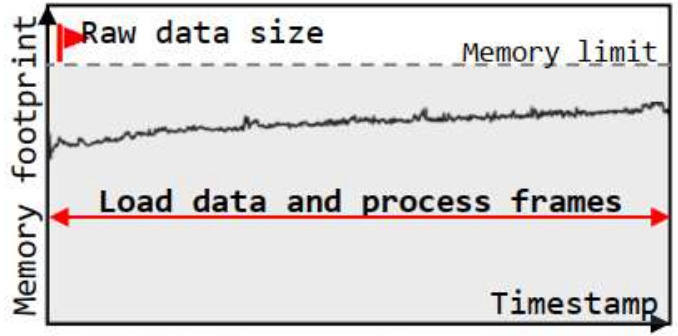
Unchanged Sensitivity



(d) Quicksort on c++ std library

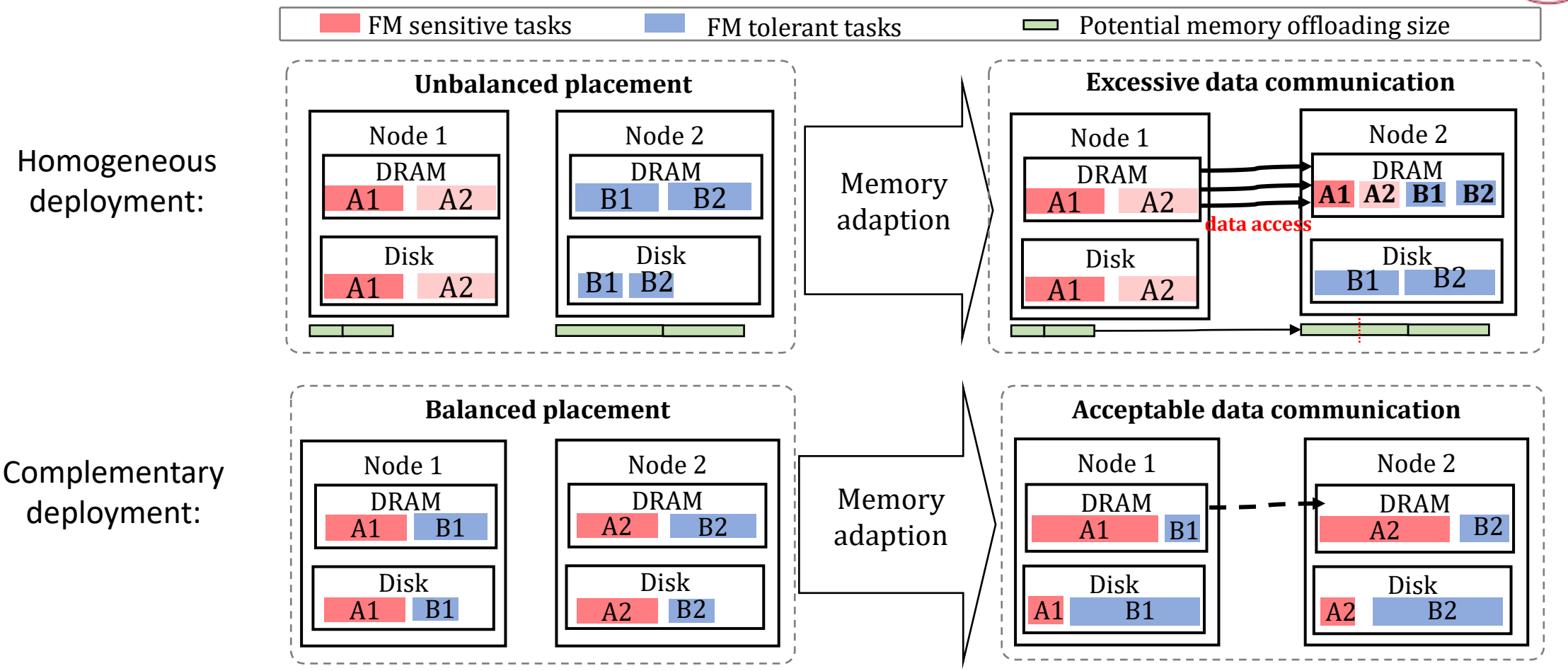


(e) CNN inference on Tensorflow



(f) Video processing on FFmpeg

Challenge: Concurrent Tasks on Hybrid FM



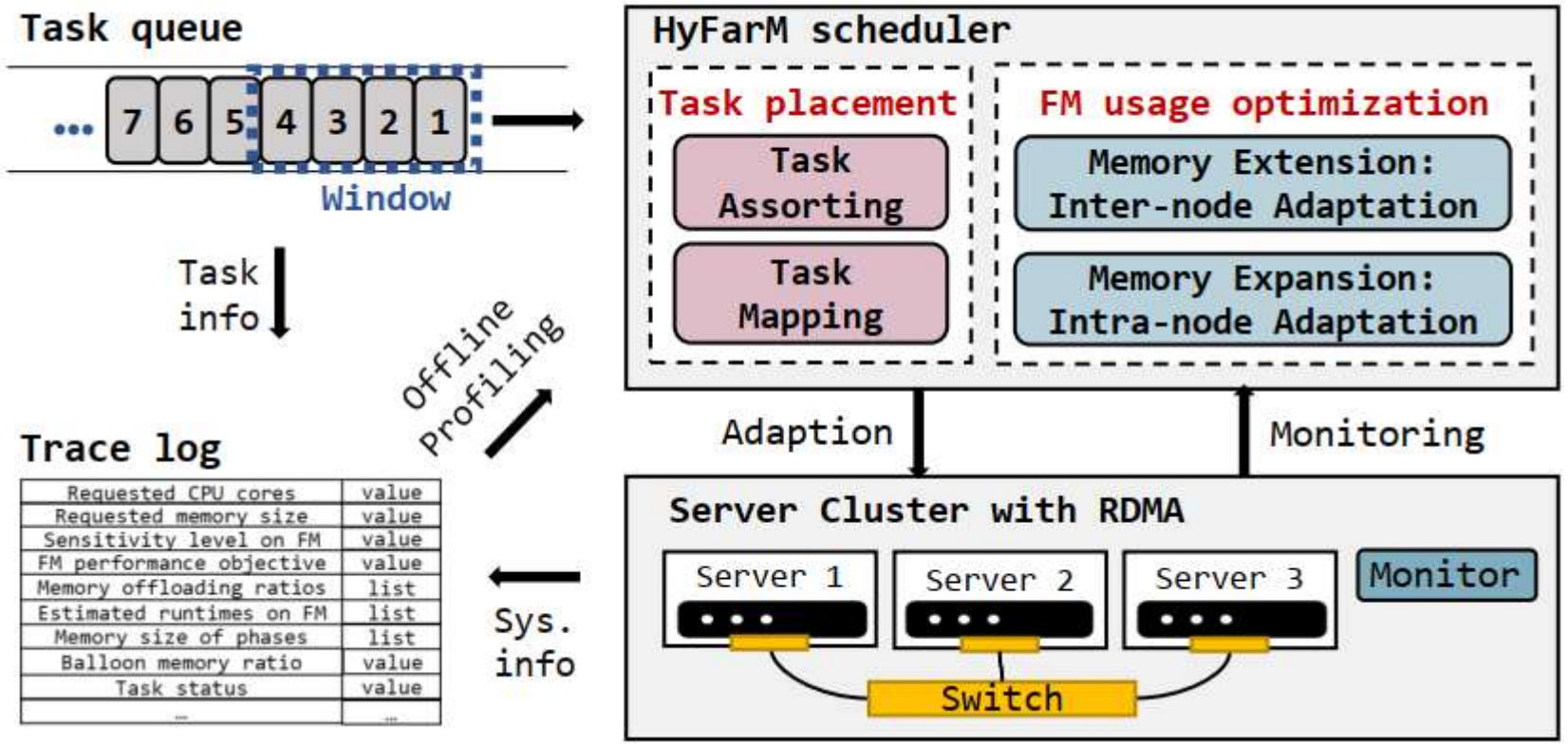
- Smart collocation of FM-sensitive tasks and FM-tolerant tasks is critical.
- FM-tolerant tasks can conveniently spare memory space for local high-priority tasks, without generating excessive network I/Os.

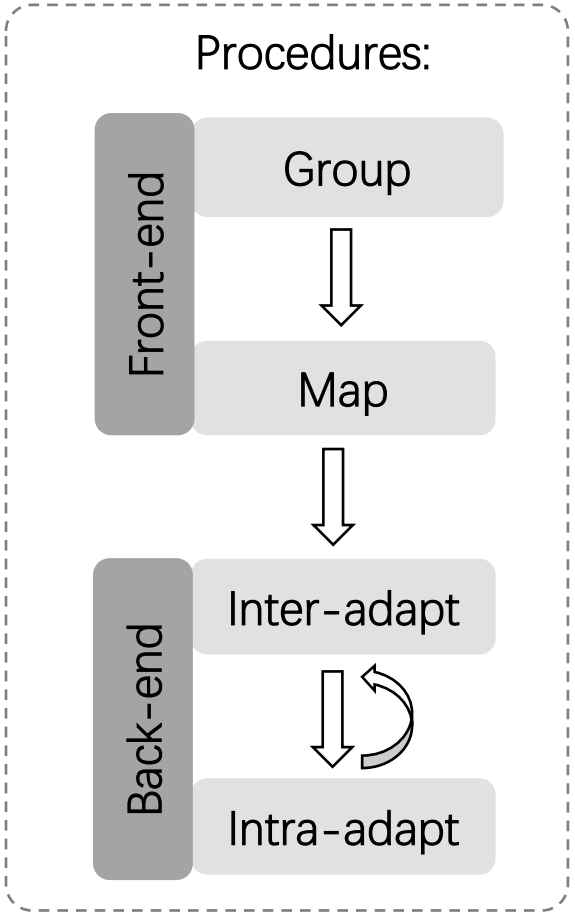
Outline

- Background
- Challenge and Motivation
- **HyFarM Design**
- Evaluation
- Conclusion

HyFarM Design

HyFarM mainly consists of two parts:
The **front-end:** ***FM-oriented task placement***
The **back-end:** ***hybrid FM usage optimization***

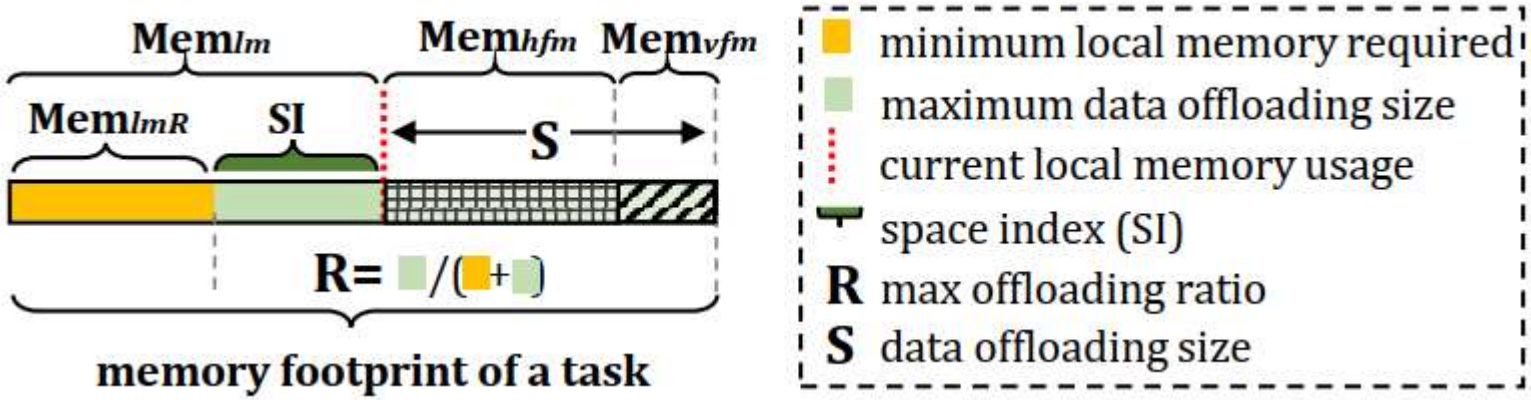




HyFarM frontend: FM-oriented task placement

(1) FM-oriented Task Grouping

- *Task Profiling*
  - We mainly consider two factors: max offloading ratio (**R**) and data offloading size (**S**).
  - We use space index (**SI**) to refer to the capability of an entity to spare its memory.



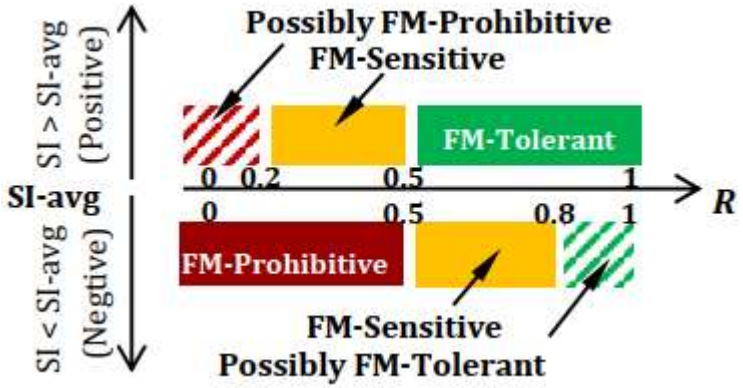
$$SI = R * Mem_{total} - S$$

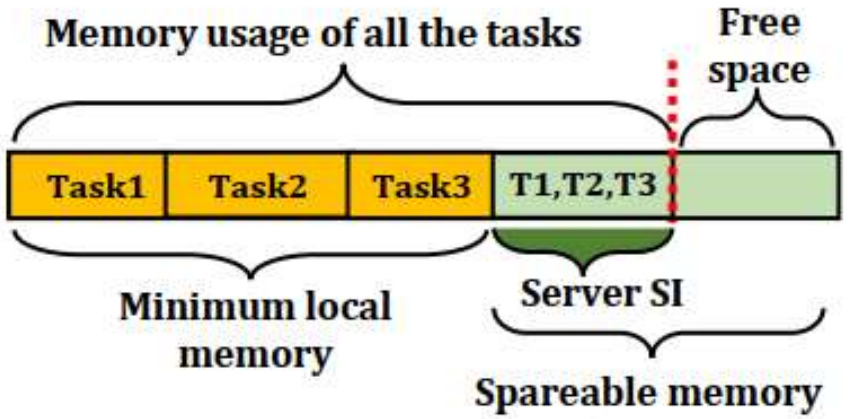
HyFarM frontend: FM-oriented task placement

(1) FM-oriented Task Grouping

- *Task Profiling*
  - We mainly consider two factors: max offloading ratio (**R**) and data offloading size (**S**).
  - We use space index (**SI**) to refer to the capability of an entity to spare its memory.
- *Task Assorting*
  - We divide tasks into three groups: FM-prohibitive, FM-tolerant, and FM-sensitive tasks.

$$SI_{offset} = SI_i - SI_{avg}.$$

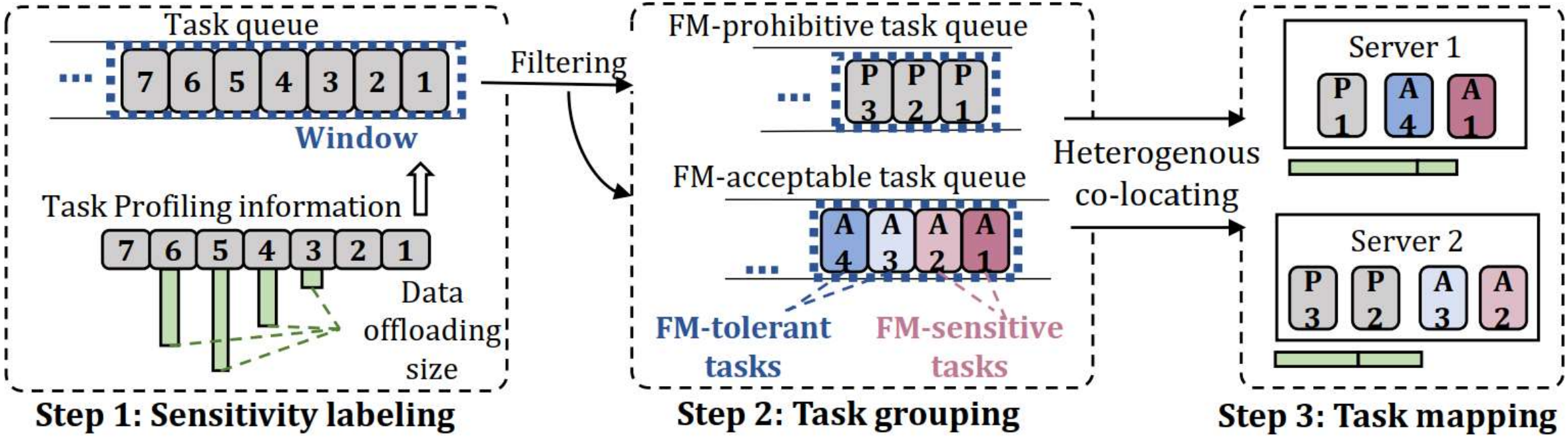




HyFarM frontend: FM-oriented task placement

(2) FM-oriented Task Mapping:

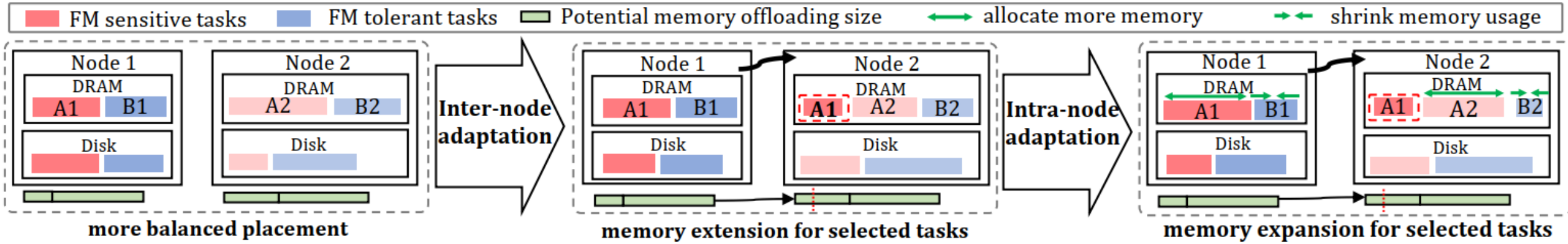
- Guarantee the minimum required local memory of the assigned tasks
- Provide extra local memory in a best-effort way
- Carefully co-locate tasks from two complementary groups



HyFarM Back-end: hybrid FM usage optimization

(1) Inter-node Adaption→ Memory Extension

- Use server SI to maintain balance between servers.
- Far memory providers spare their memory space to far memory borrowers.
- Identify to-be-shrunk and to-be-extended tasks according to tasks SIs.



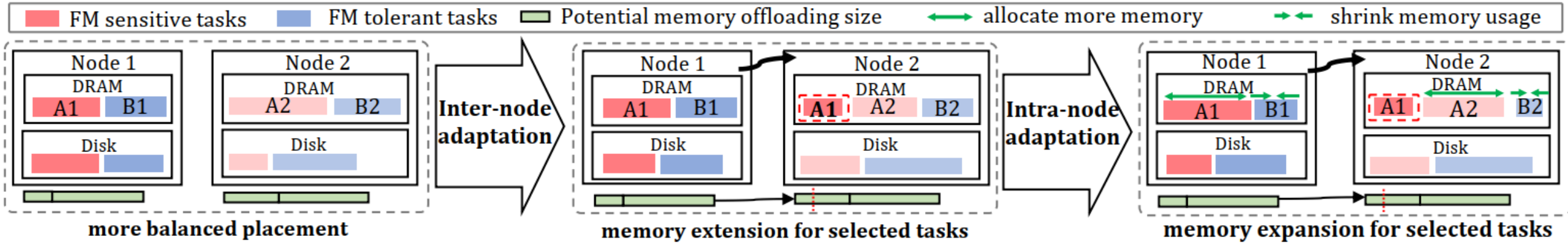
HyFarM Back-end: hybrid FM usage optimization

(1) Inter-node Adaption→ Memory Extension

- Use server SI to maintain balance between servers.
- Far memory providers spare their memory space to far memory borrowers.
- Identify to-be-shrunk and to-be-extended tasks according to tasks **SI**s.

(2) Intra-node Adaption→ Memory Expansion

- FM-tolerant tasks sacrifice to the least local memory ratio for overall efficiency.
- Assign extra local memory space to FM-sensitive tasks according to ratio of task **SI**s.
- Recall this step to keep balance when task phase changes or task finishes.



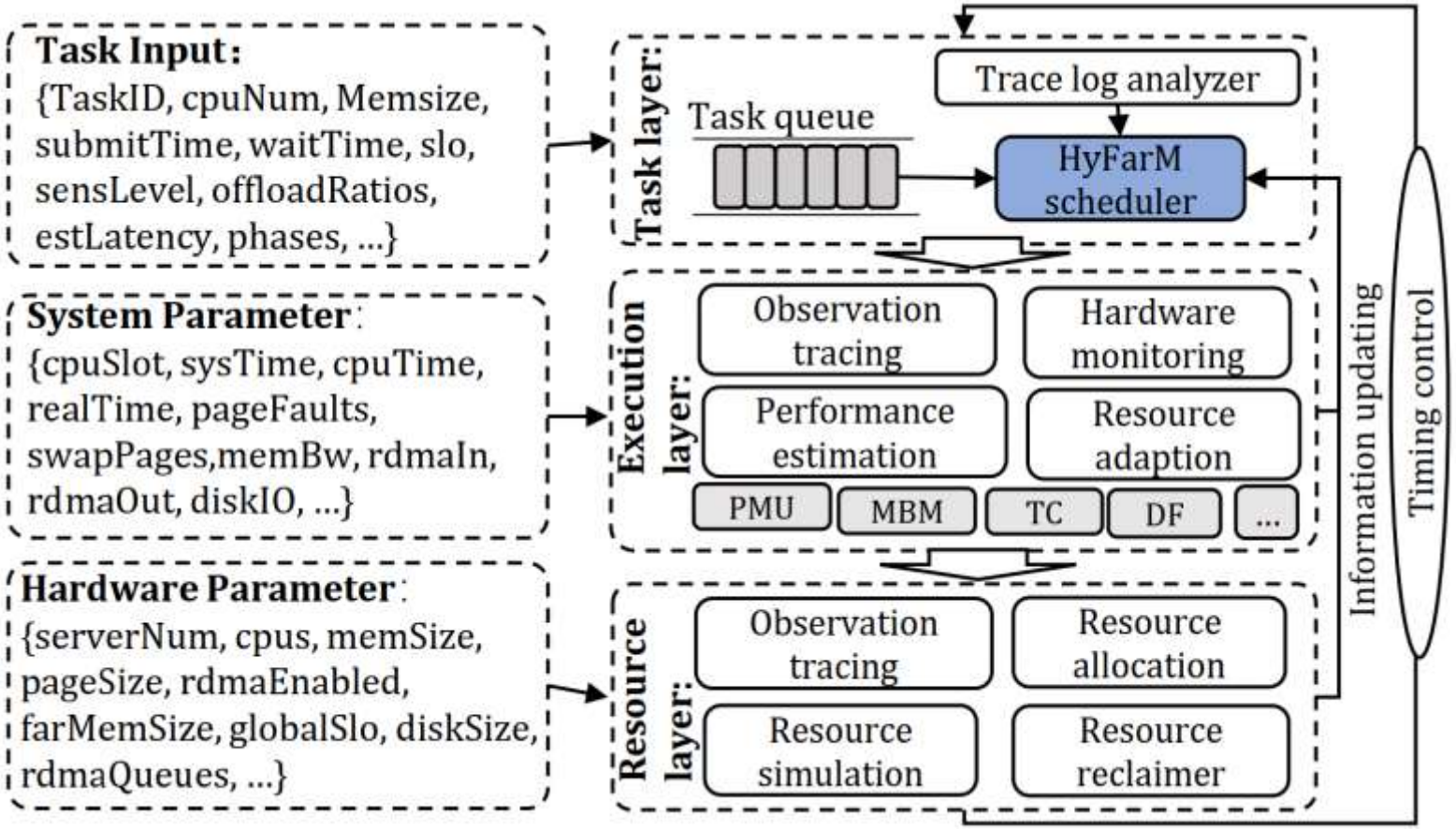
Overall Simulation Methodology

(1) Profile.

- Far memory ratio
- Performance trends
- Task phases

(2) Far Memory Allocation.

- Pre-allocate server id
- Inter-node adaption
- Intra-node adaption
- Simulate results



Simulation Workflow

We provide such interfaces :

- **Group:** collects and calculates the attribute values (labels) of each task.
- **Map:** adopts bin-packing-based algorithms
- for solving the task placement problem.
- **Inter-adapt:** marks servers as either FM borrower or provider.
- **Intra-adapt:** adjusts the to be-allocated memory size of each task inside the server.

Algorithm 1 Pseudocode of HyFarM.

```
1:  HyFarM_manager(tasks, cluster) {
2:        tasks.group(tasks, cluster)
3:        For tasks in cluster.task_pending_list  do
4:        task.server_id = map(task, cluster)
5:        task.server_id.inter_adapt(cluster)
6:        while (is_task_finished or is_task_phase_changed) do
7:              task.local_server.update_SI()
8:              if (changed_SI > Threshold) then:
9:                    inter_adapt(cluster)
10:             else: intra_adapt(task.local_server)
11:       server_list = sort_server_by_SI(cluster)
12:       for (each server_i in server_list) do
13:             if (server_i.SI >= task.least_local_memory) then:
14:                   server_i.add_task(task,task.local_mem) }
15: intra_adapt(server){ // adjust memory inside servers
16:       victim_tasks = estimate_best_ratio(tasks)
17:       task.local_server.reset_task(victim_tasks)
18: inter_adapt(cluster) { // adjust memory across servers
19:       provider = find_largest_SI_server(cluster.server_list)
20:       borrower = find_least_SI_server(cluster.server_list)
21:       provider .add_task(borrower_task)
22:       intra_adapt(provider) }
```

Outline

- Background
- Challenge and Motivation
- HyFarM Design
- **Evaluation**
- Conclusion

Evaluation: Experimental Environment

- **Testbed:**

Memory Type	Hardware	Bandwidth	Size	Environment
Local main memory	DRAM	200 Gpbs	128G	Linux OS
Horizontal far memory	RDMA	40 Gbps	<32G	Fastswap
Vertical far memory	Disk	2.5 Gbps	1T	Linux swap

- **Applications:**

Abbr	Algorithm Description	Framework	Mem. Footprint
qs	quicksort	C++ std [1]	2G~10G
bfs	breadth first search	Ligra [39]	5G~20G
pr	pagerank	Ligra [39]	5G~20G
mp4	mp4 format transcode	FFmpeg [4]	6G~30G
mkv	mkv format transcode	FFmpeg [4]	8G~30G
tf-i	tensorflow inception	Tensorflow [10]	4G~20G
tf-r	tensorflow resnet inference	Tensorflow [10]	4G~20G
fs-i	few-shot detection on images	FsDet [48]	1G~10G
fs-v	few-shot detection on videos	FsDet [48]	10G~30G

Evaluation: Experimental Environment

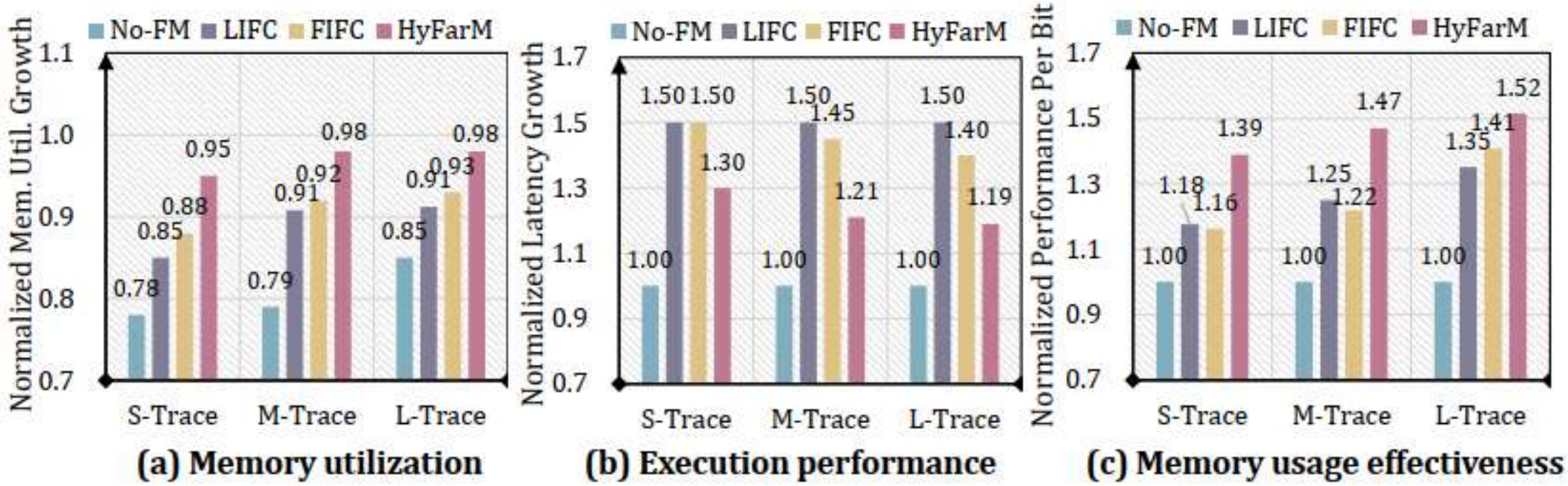
- **Workload Traces and Cluster Configurations**

Workload Traces	S-Trace	M-Trace	L-Trace
Task number	200	500	2000
Task characteristics	latency: 10∼6000s; dataset: 1G∼30G		
FM-sensntive task ratio	10%, 30%, 50%, 70%, 90%		
FM-prohibitive task ratio	10%, 20%, 30%, 40%, 50%		
Window size	10	20	50
Server memory	DRAM memory with 256G		
hFM size	32G at maximum		

- **Baseline methods**

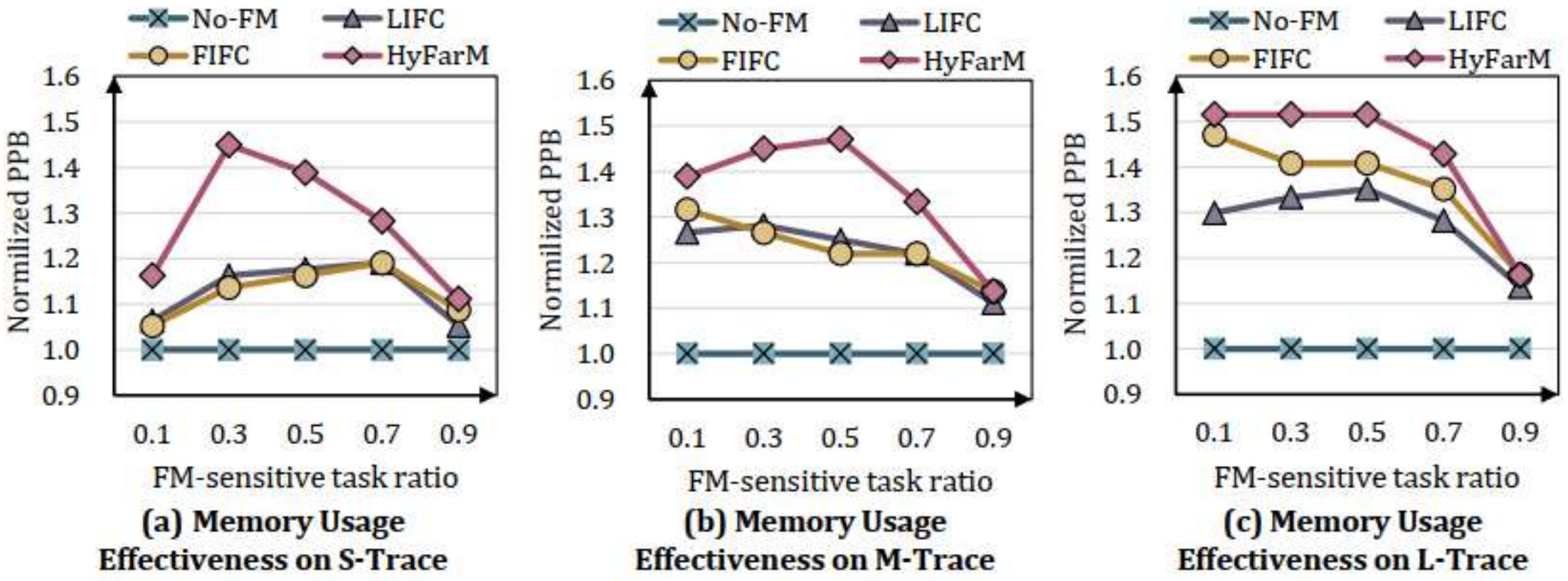
Evaluation	Reference	Memory Allocation Method
non-FM	CQsim [35], without FM	Kill tasks upon resource shortage
LIFC	Zswap [31], VFM only	Last-in-first-cap in proportional
FIFC	CFM [12], HFM only	First-in-first-cap in proportional
HyFarM	ours, HFM+VFM	SI-based HyFarM strategy

Evaluation: Overall Benefits of HyFarM



- Our design shows high **memory utilization** than non-FM, LIFC and FIFC baselines.
- We observe a moderate increase in average task **execution duration** of up to 30%.
- We improves the **PPB** by up to 52%, 17.6%, 20.5% compared with non-FM, LIFC, FIFC baselines.
- HyFarM performs better under larger task set.

Evaluation: Impact of Workload Composition



(a) Memory Usage Effectiveness on S-Trace

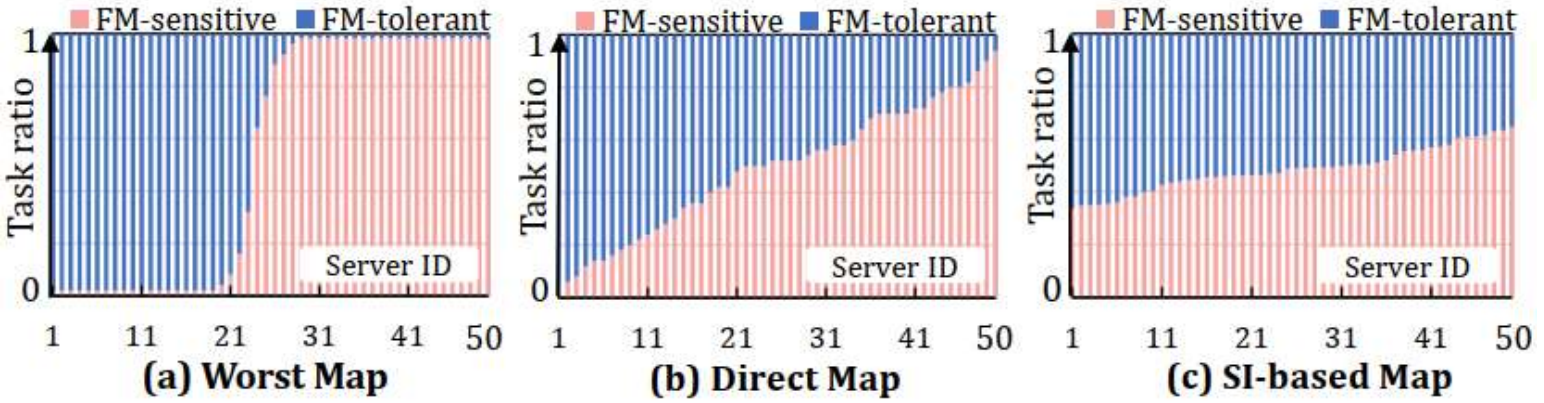
(b) Memory Usage Effectiveness on M-Trace

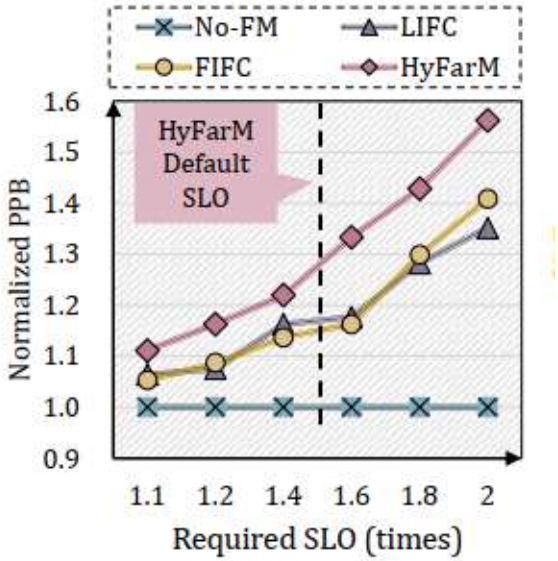
(c) Memory Usage Effectiveness on L-Trace

- Our system has better results when the task set becomes large.
- Our method offers the best results when FM-sensitive tasks account for half of the overall tasks.

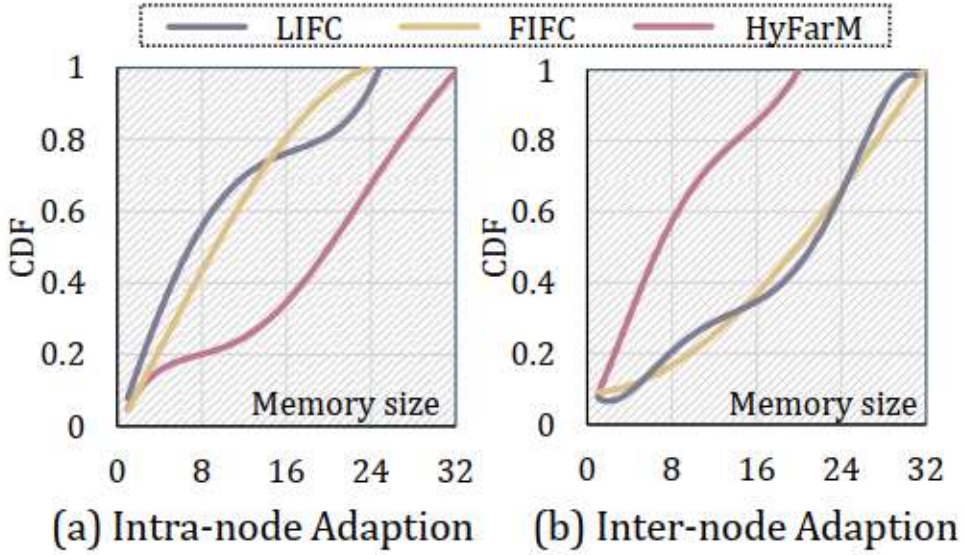
Evaluation: A Deeper Look at HyFarM

- Impact of Sensitivity-aware Mapping



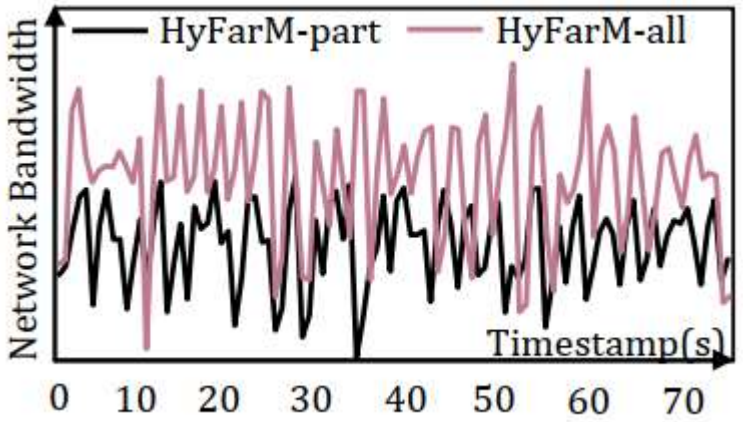


- Latency-tolerance Tasks

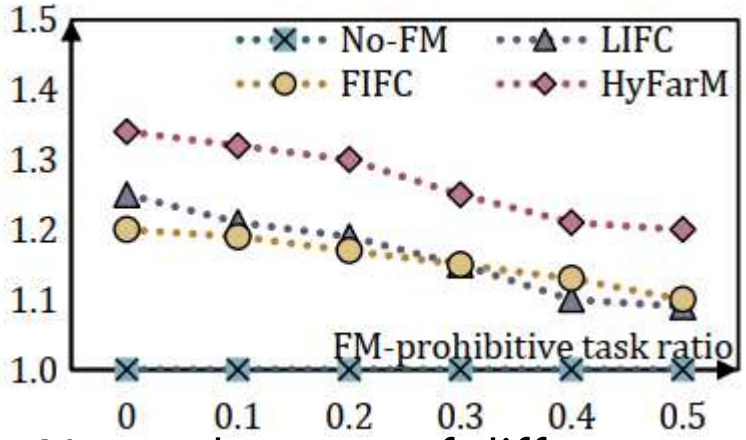


- Joint Node Adaptation

Evaluation: External Influences



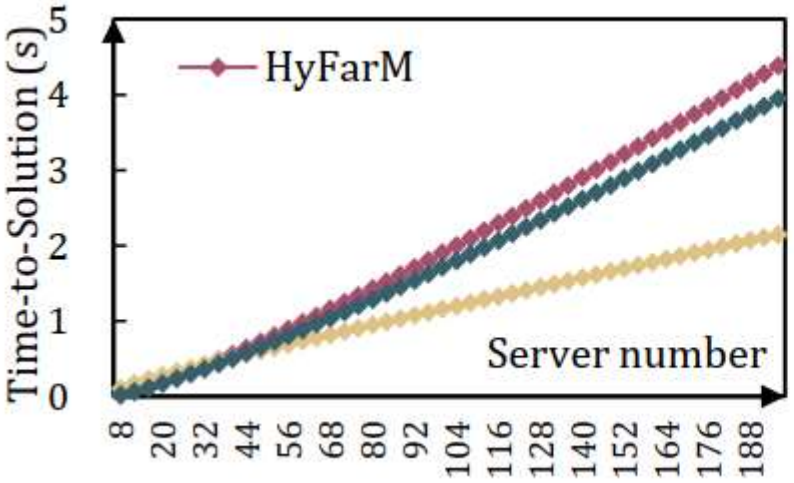
- Network usage on different FM-prohibitive Tasks



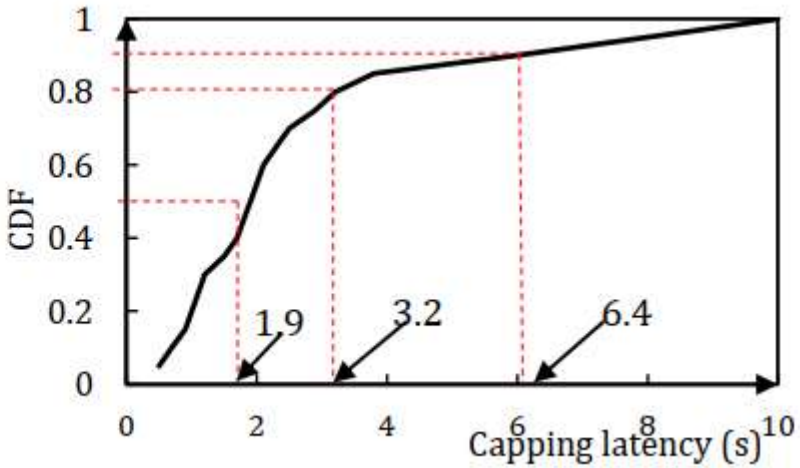
Network usage of different ratio of FM-prohibitive Tasks

- We see increased bandwidth consumption if the FM-prohibitive tasks are few by comparing cases with or without FM-prohibitive tasks.
- Even if over half of tasks are FM-prohibitive, we can still yield 10% throughput improvement than the best baseline method.

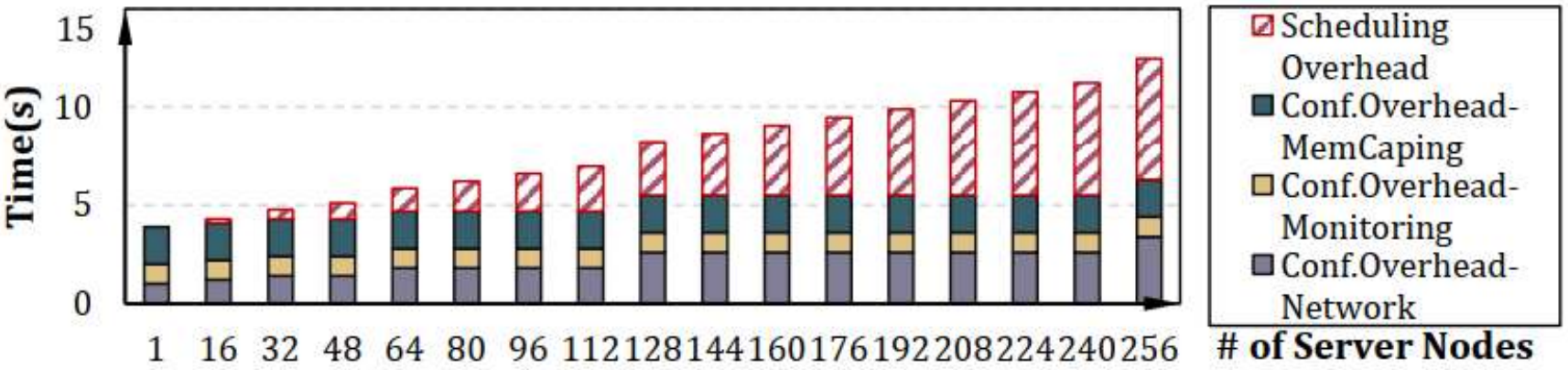
Evaluation: Overhead Analysis



- Task scheduling overhead



- Memory capping overhead



- The scheduling overhead and configuration overheads are positively correlated with system scale.
- The total control overhead is less than 14 seconds which is acceptable for HPC scheduling.

Conclusion

In this work, we explore the potential of **hybrid far memory management**.

- We take the first step to explore **sensitivity-aware** task orchestration in a hybrid FM environment.
- Collocating FM-sensitive and FM-tolerant tasks **in complementary way** brings better memory efficiency.
- Combining both memory **expansion and extension** can greatly improve memory utilization and performance per bit.
- We will continue to improve our design for real-world environments and adapt to new disaggragrated memory devices in the future work.

Contact us at Email: {jing618, meijunyi, he-hao, sjtuwtl, wpybtw, luzhang}@sjtu.edu.cn,
{lichao, guo-my}@cs.sjtu.edu.cn, {hanqin.wuhq, dongbai.cdb, vicki.liuxw}@alibaba-inc.com