OdinFS: Scaling PM Performance with Opportunistic Delegation

Diyu Zhou*, Yuchen Qian, Vishal Gupta, Zhifei Yang,
Changwoo Min, Sanidhya Kashyap

EPFL

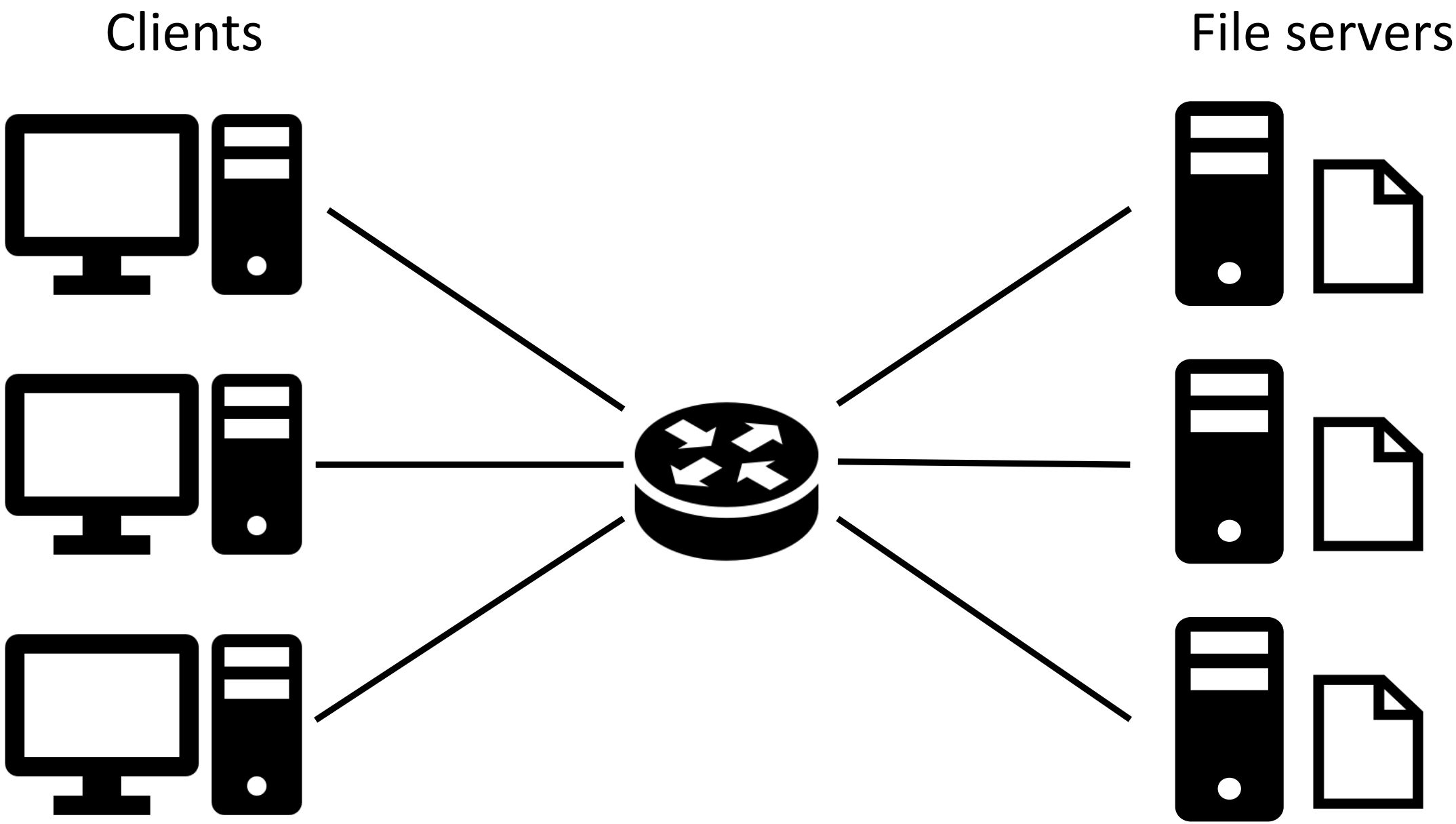
*Looking for a faculty job

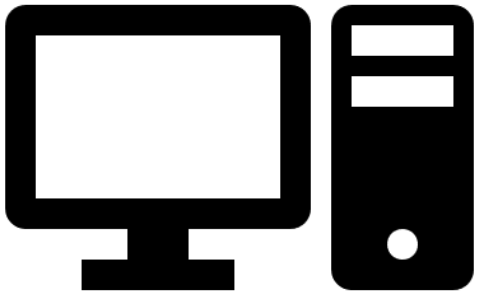
VIRGINIA TECH™

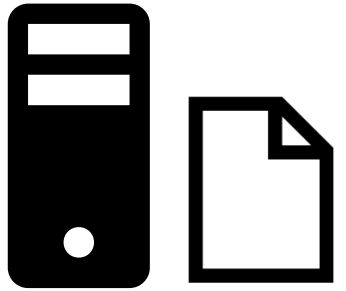
A short time ago, in Silicon Valley…

This is Mei

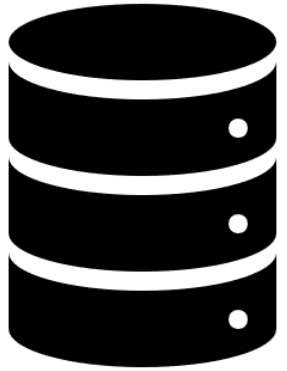
Mei builds a fileserver service for a startup company

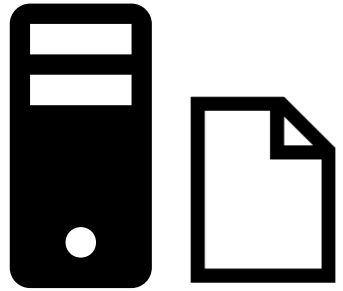

Clients
File servers

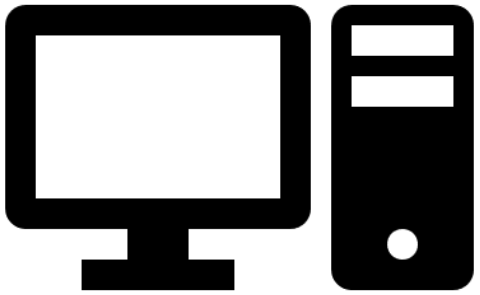


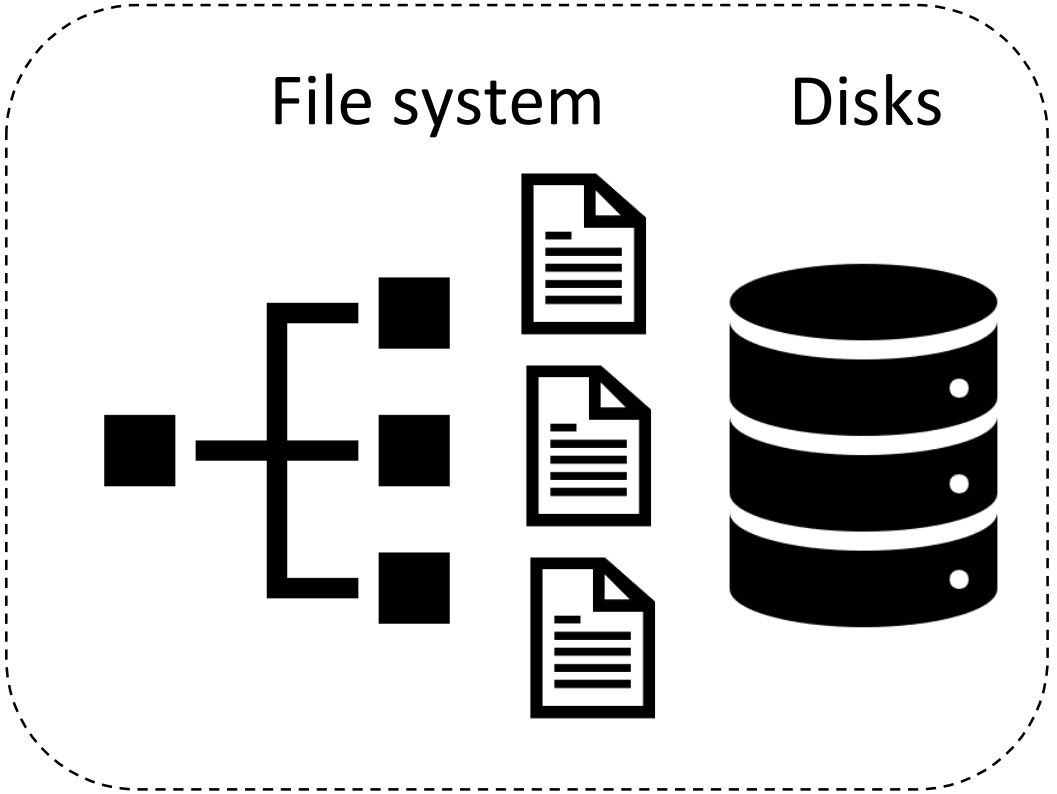


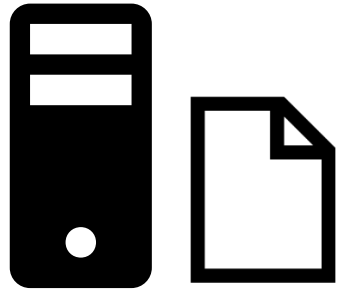
Disks

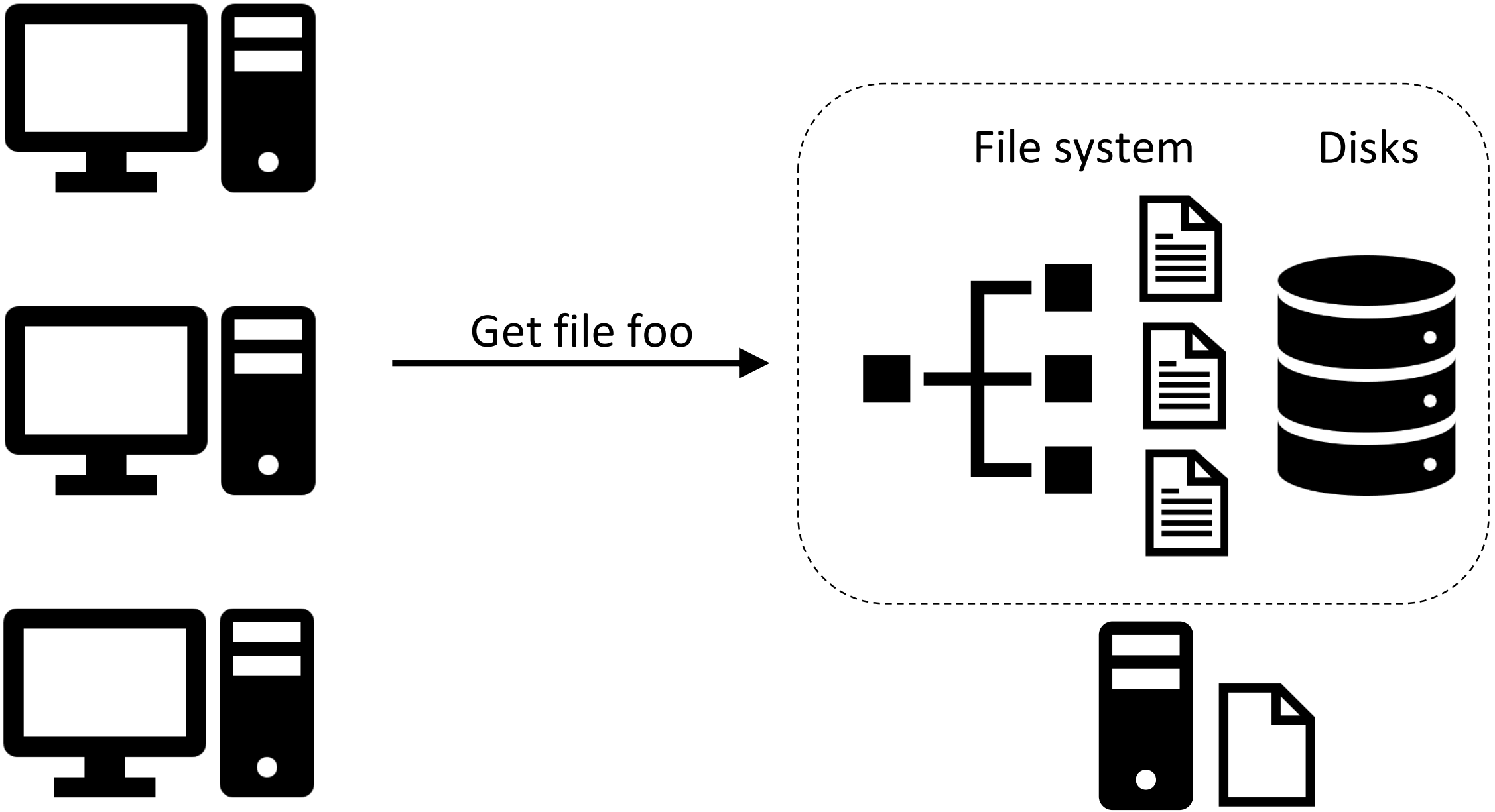


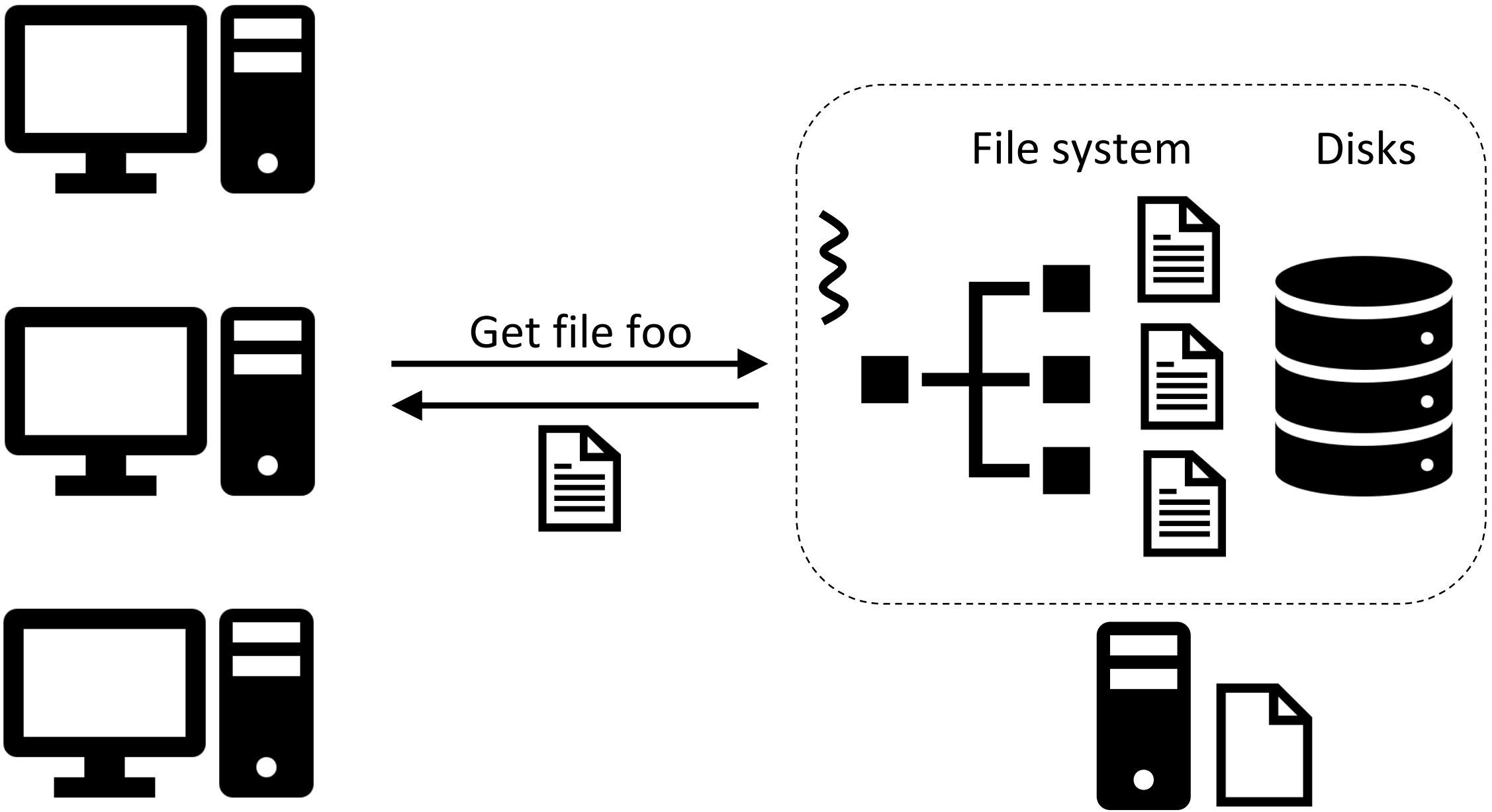


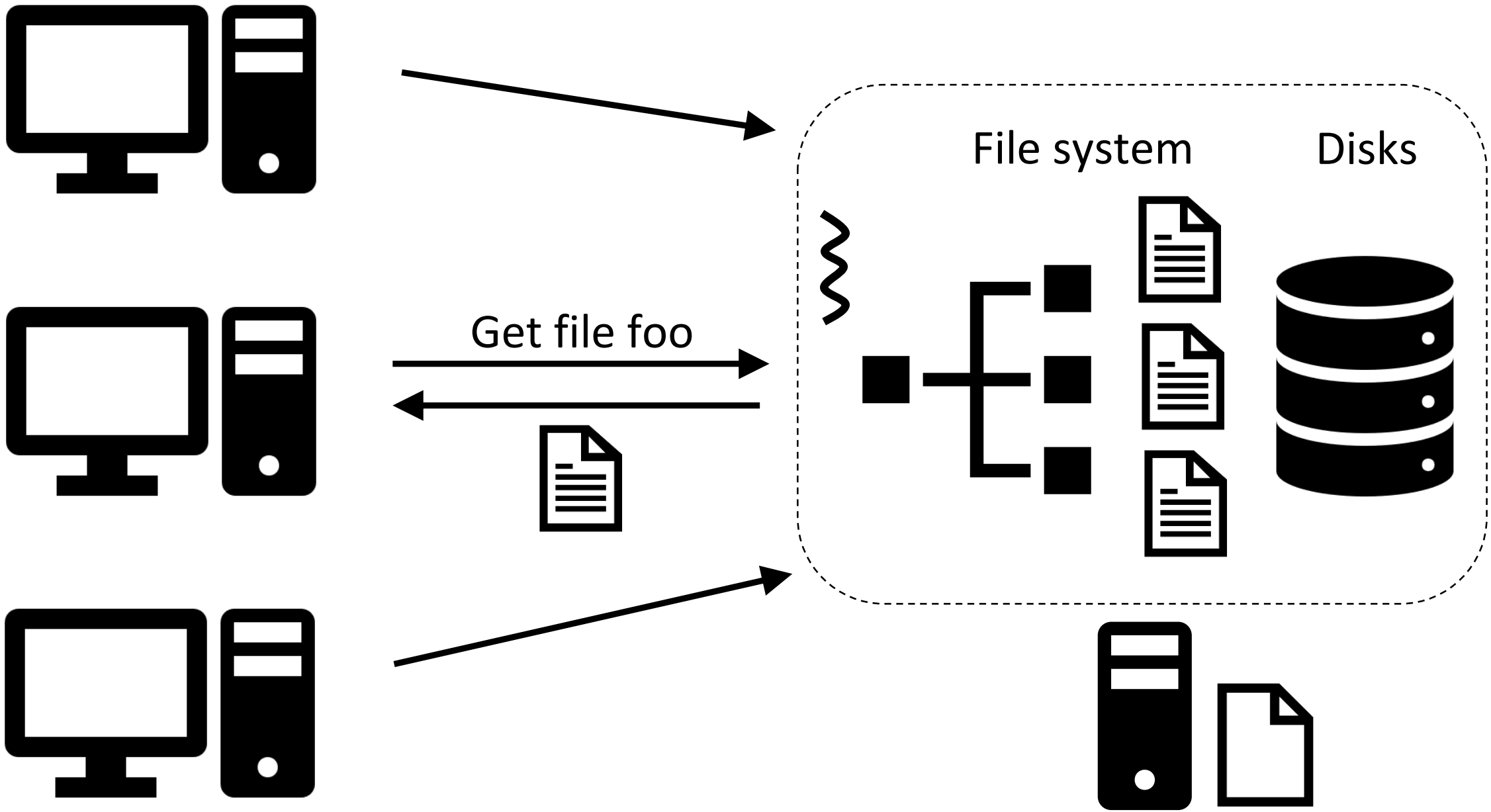


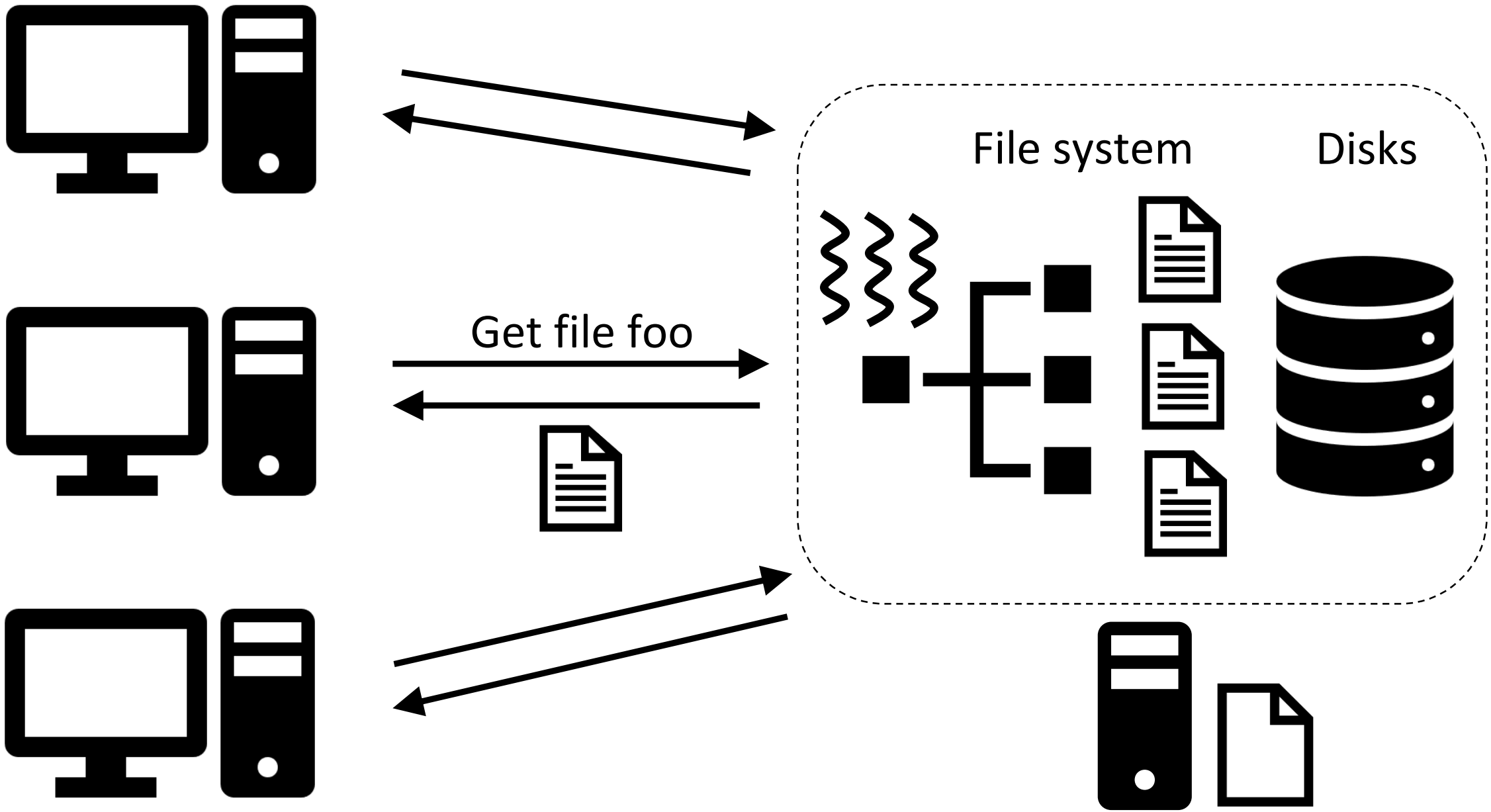

File system
Disks




File system
Disks
Get file foo


File system
Disks
Get file foo


File system
Disks
Get file foo


File system
Disks
Get file foo

The file server is too slow

† Intel Optane: Faster Access to More Data

The file server is too slow

Disks are the performance bottleneck!

- Low speed; e.g., 10s-100s of µs access latency

† Intel Optane: Faster Access to More Data

The file server is too slow

Disks are the performance bottleneck!

- Low speed; e.g., 10s-100s of µs access latency
- Performance collapses with concurrent access†

† Intel Optane: Faster Access to More Data

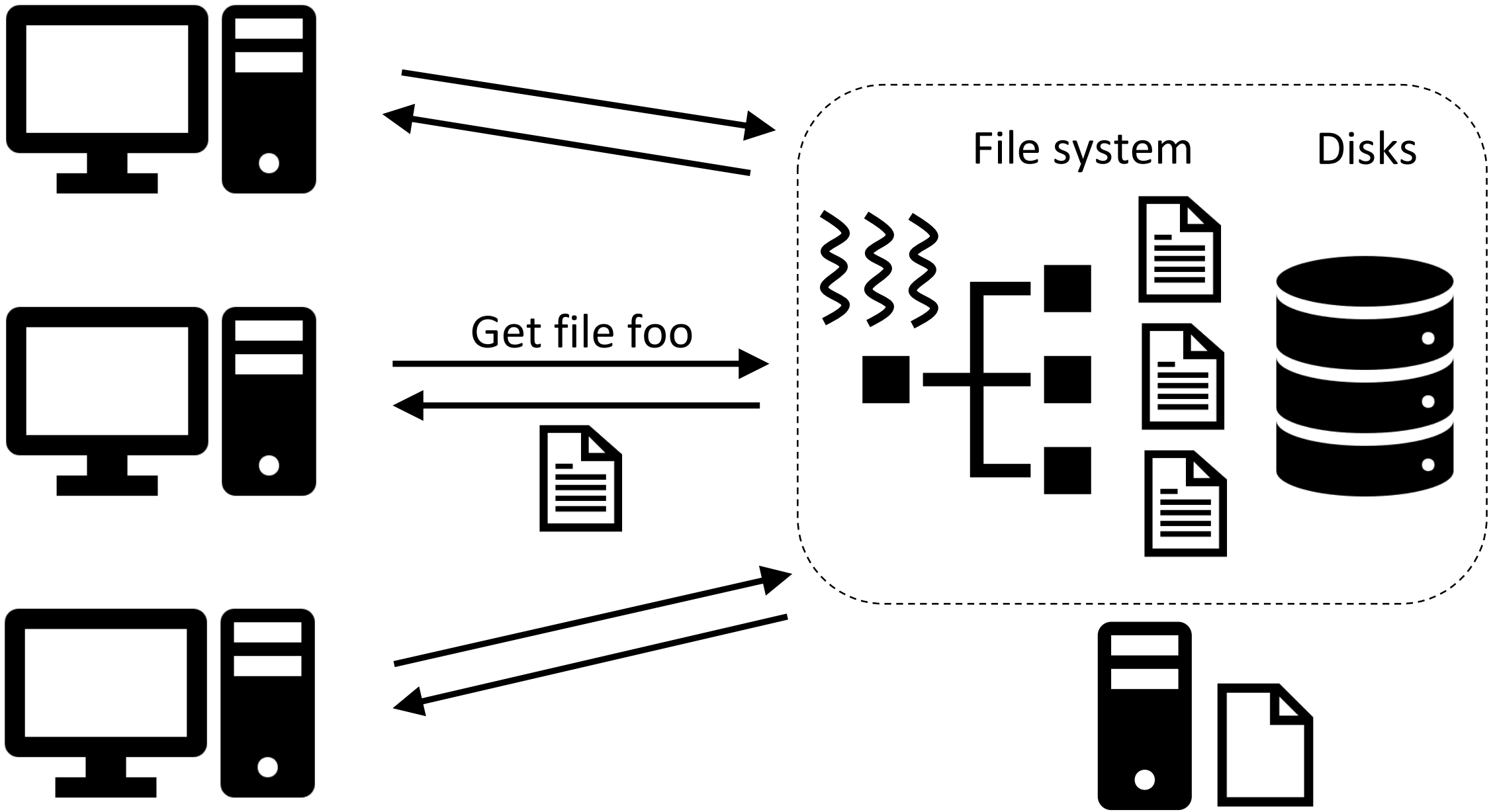
Persistent memory (PM) can solve the problem!

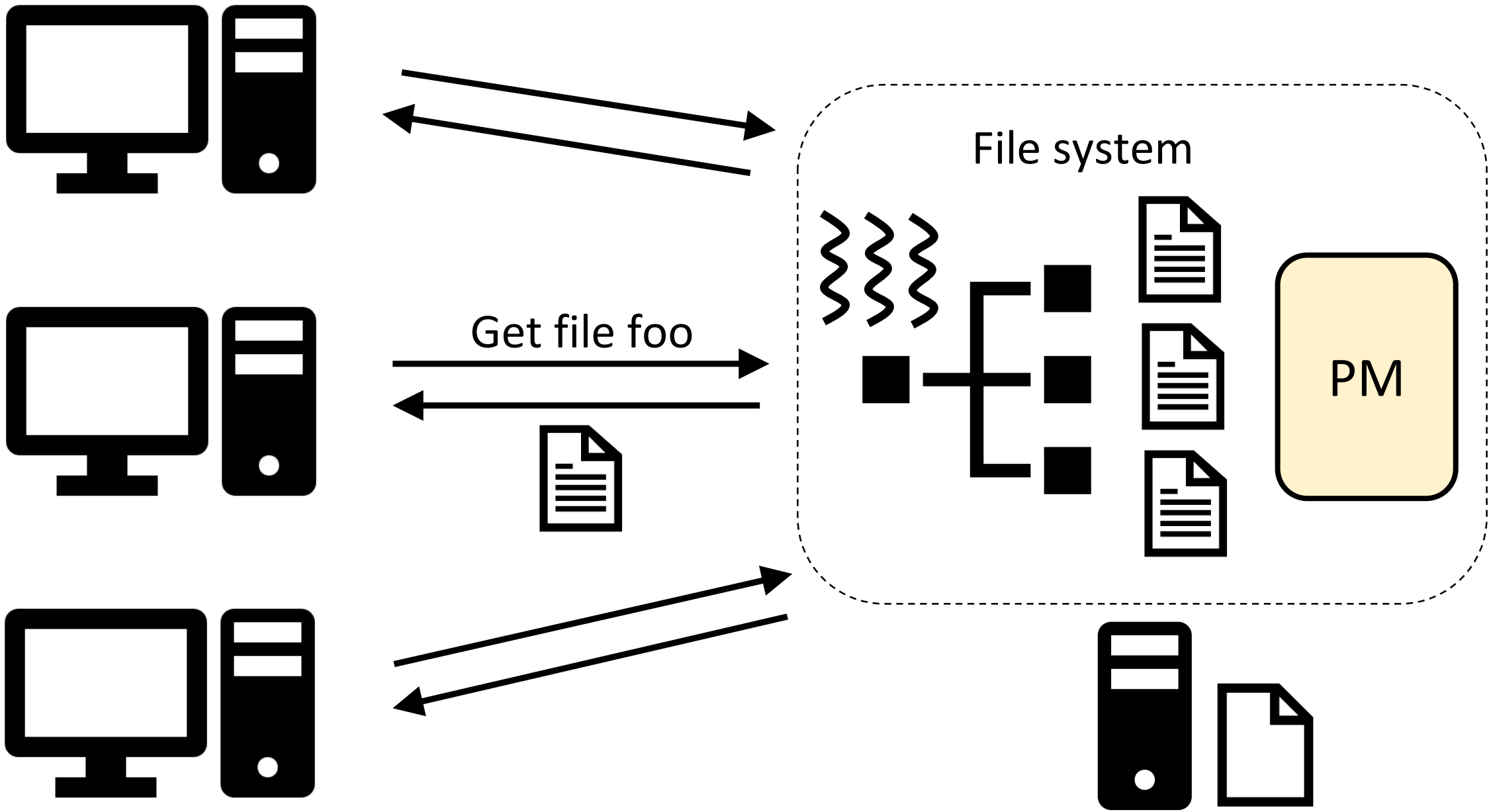
† Intel Optane: Faster Access to More Data

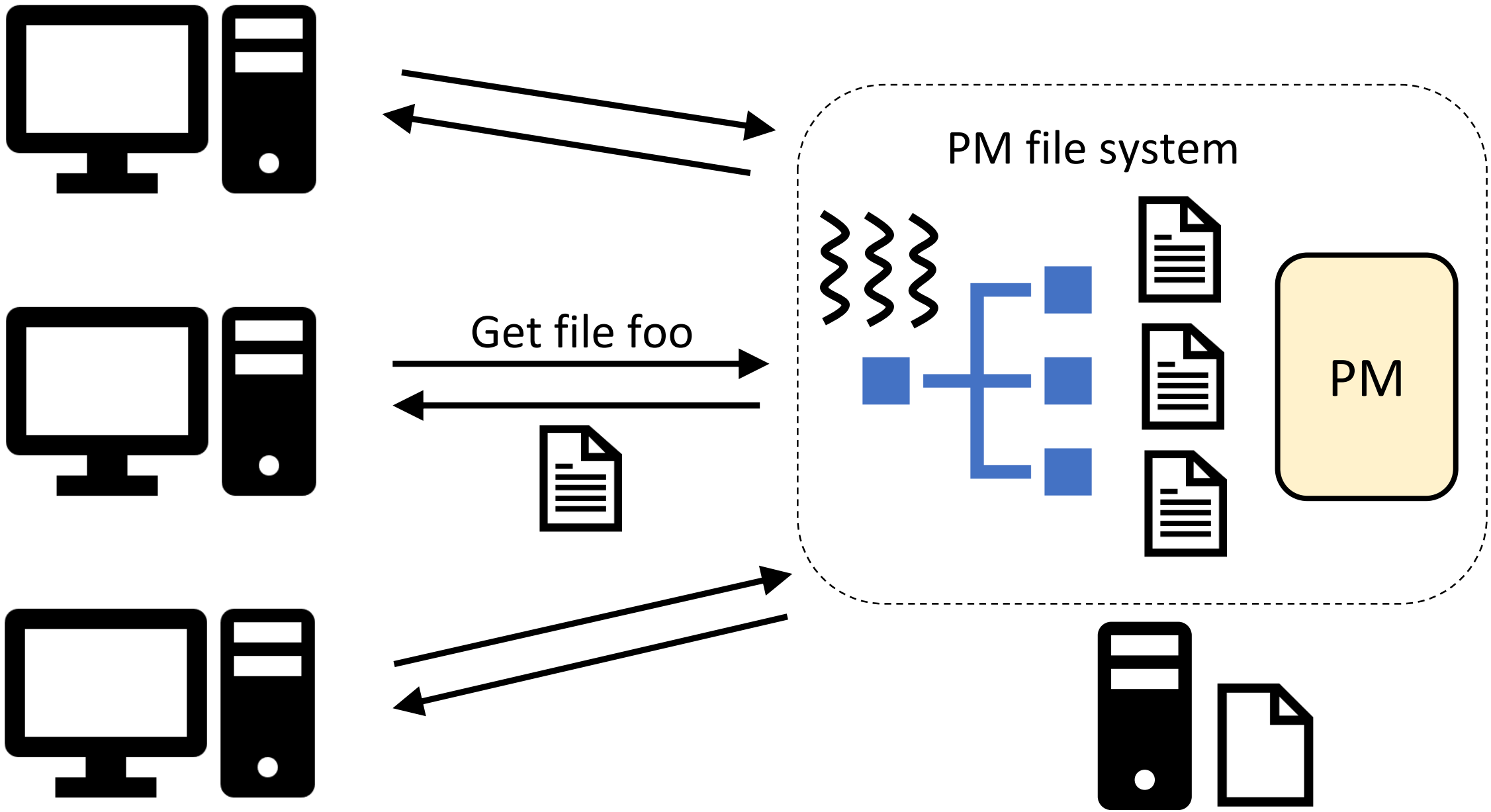
Persistent memory (PM) can solve the problem!

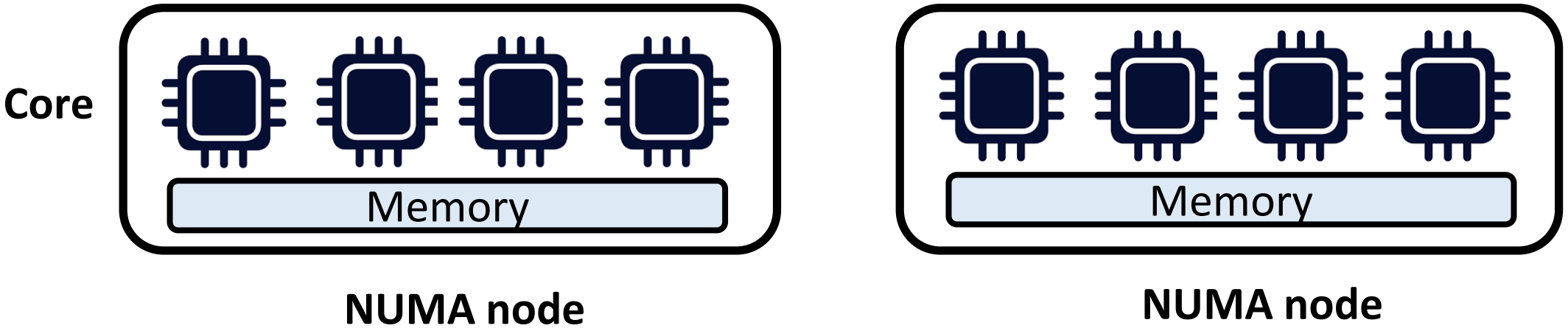
- High speed: e.g., 100s of ns access latency
- Preserves performance with concurrent access[†]

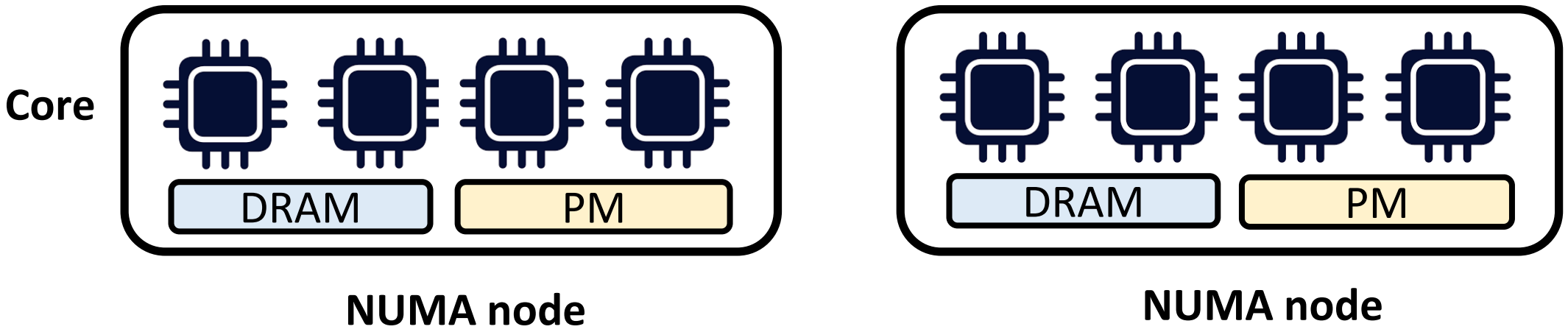
[†] Intel Optane: Faster Access to More Data

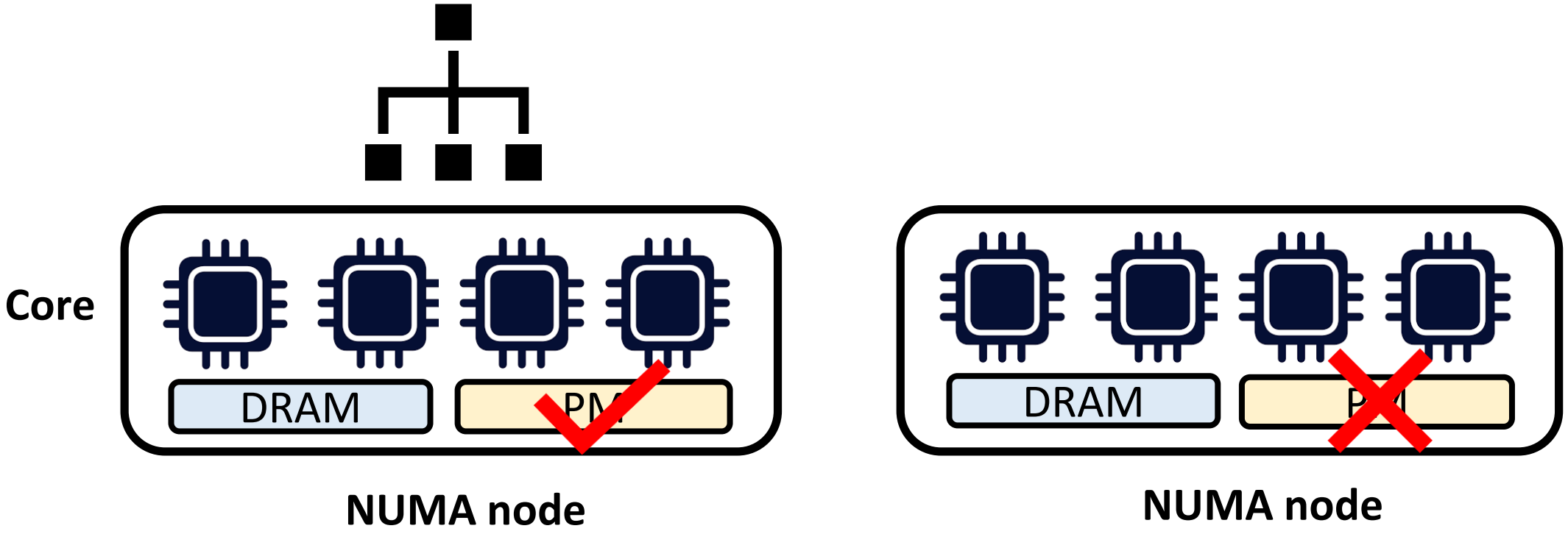

File system
Disks
Get file foo

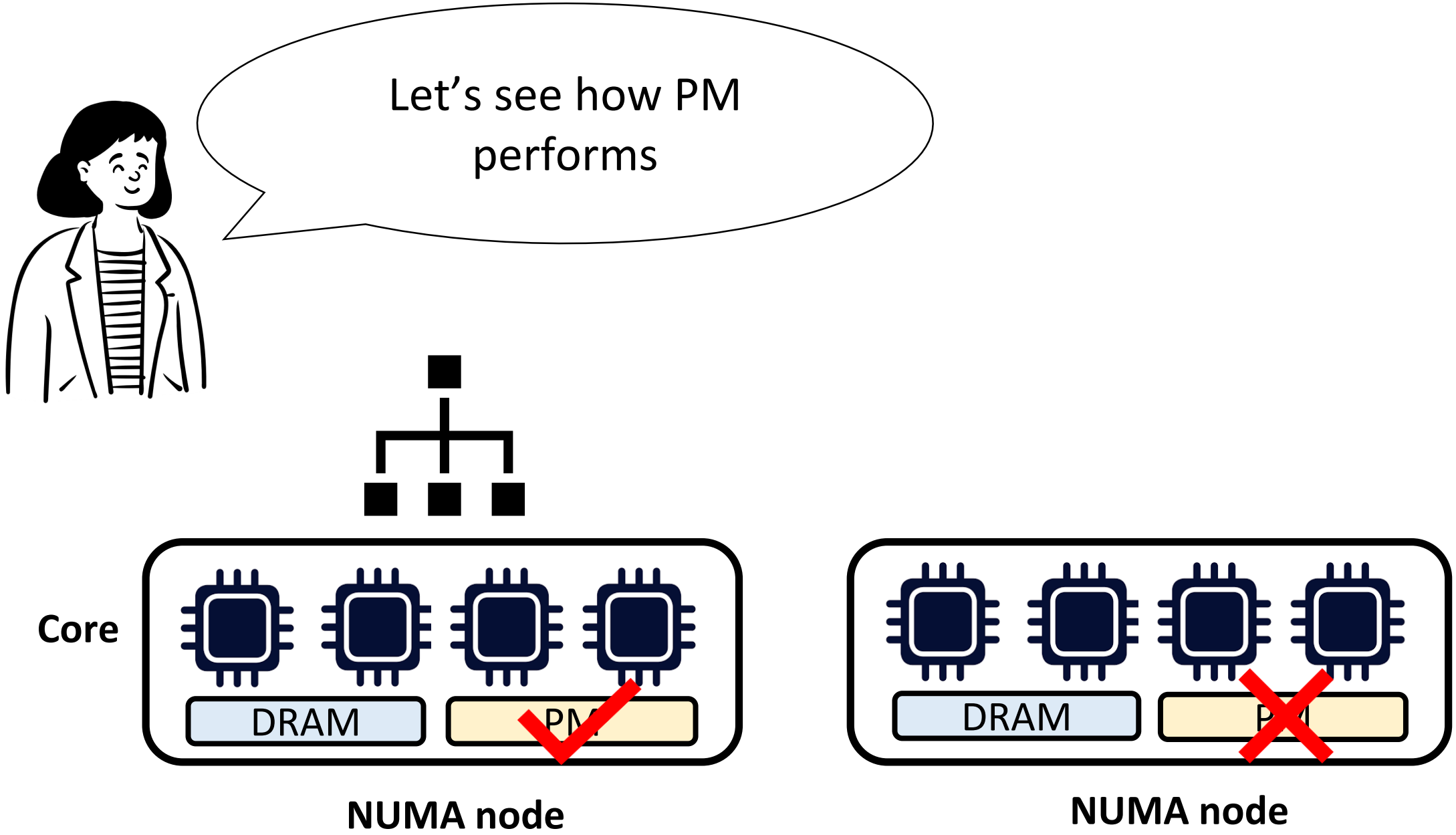

File system
Get file foo
PM


PM file system
PM
Get file foo

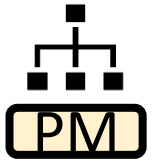

Core
Memory
Memory
NUMA node
NUMA node


Core
DRAM
PM
NUMA node
DRAM
PM
NUMA node


Core
DRAM
NUMA node
DRAM
NUMA node


Let’s see how PM performs
Core
DRAM
PM
NUMA node
DRAM
NUMA node

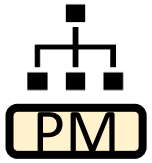
PM performance on a single NUMA node



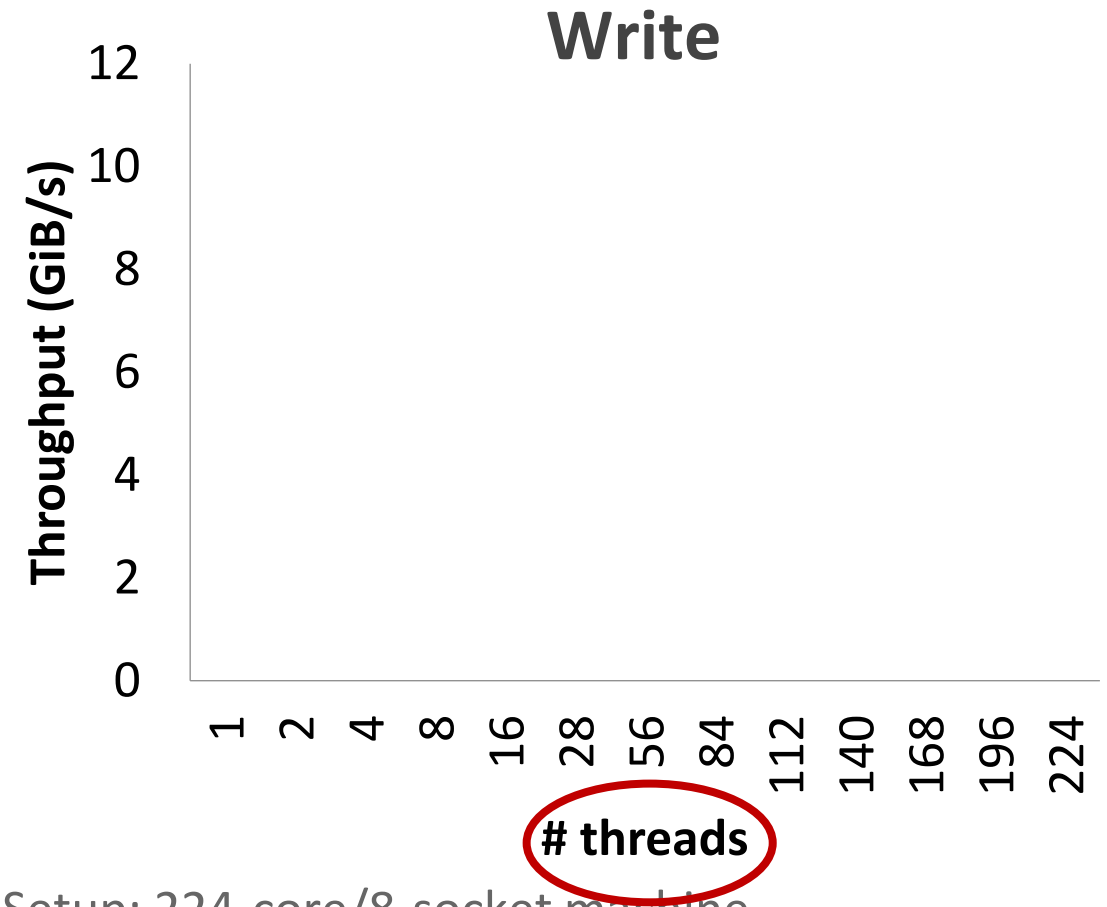
Workload: FIO: each thread writes/reads 2MB data in a private file

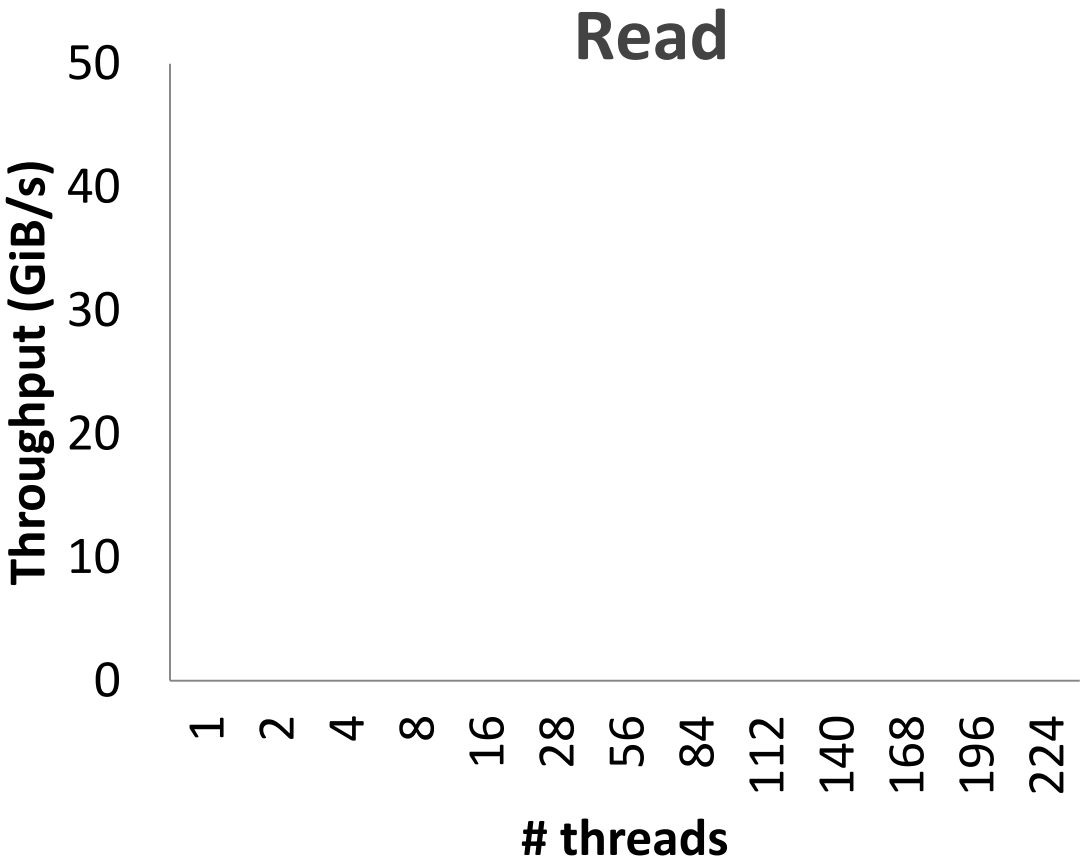
Setup: 224-core/8-socket machine

PM performance on a single NUMA node



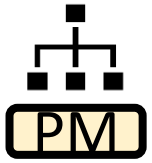
Workload: FIO: each thread writes/reads 2MB data in a private file



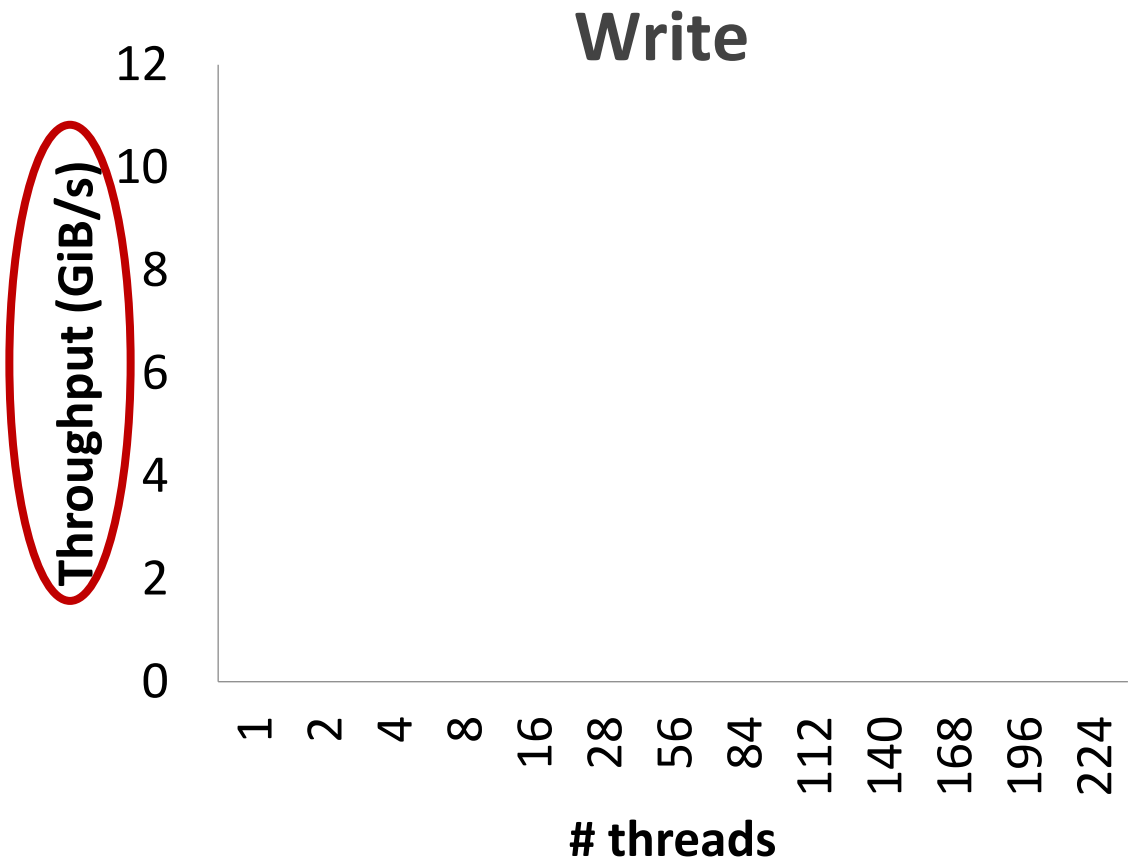


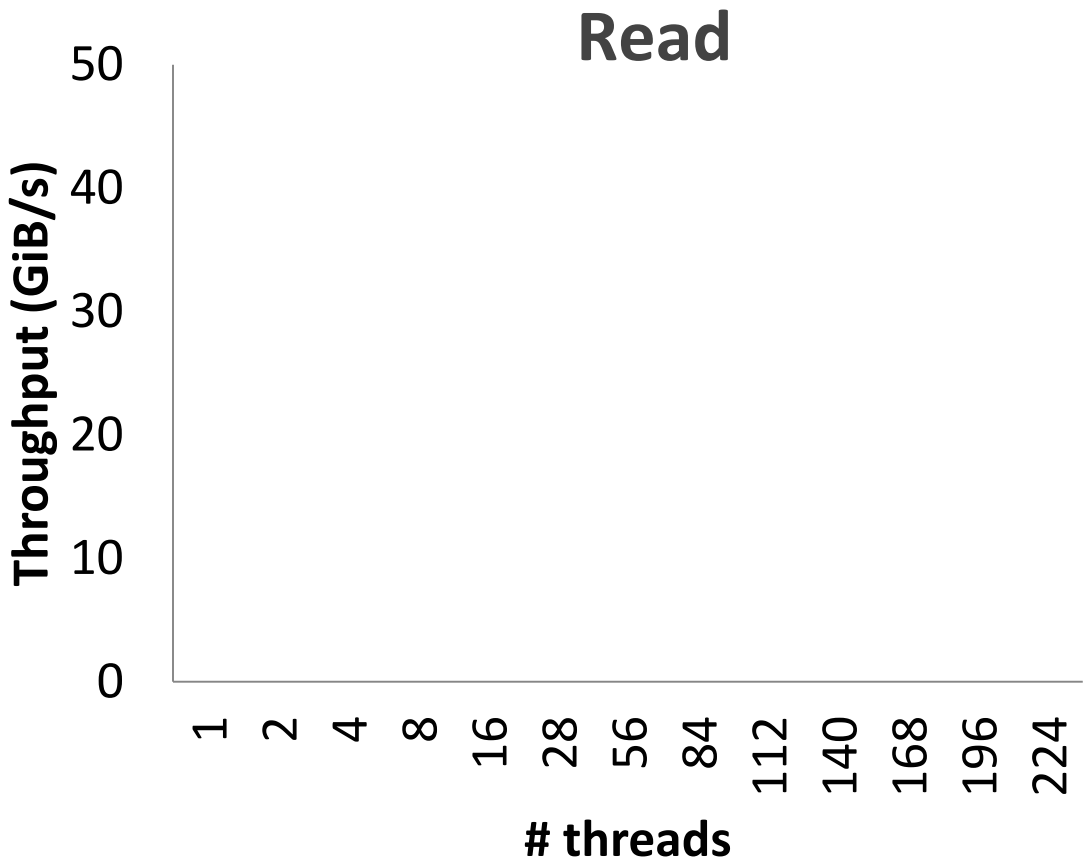
Setup: 224-core/8-socket machine

PM performance on a single NUMA node



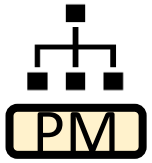
Workload: FIO: each thread writes/reads 2MB data in a private file



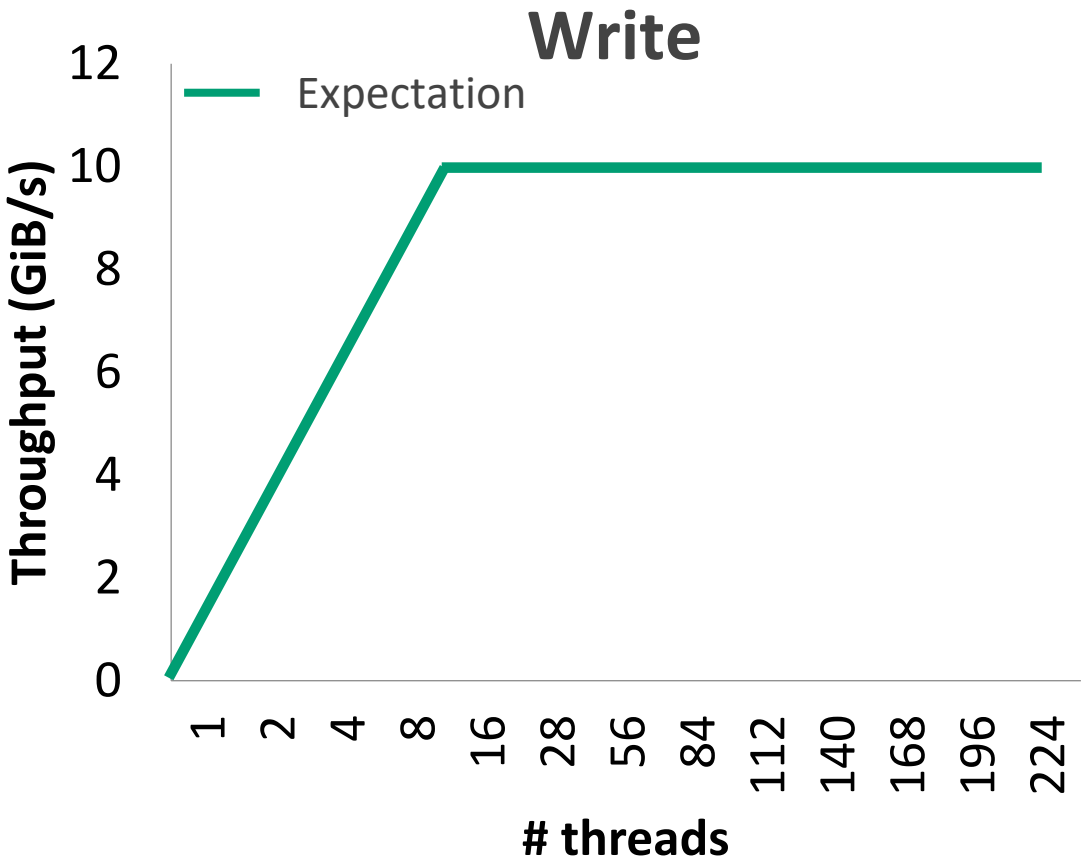


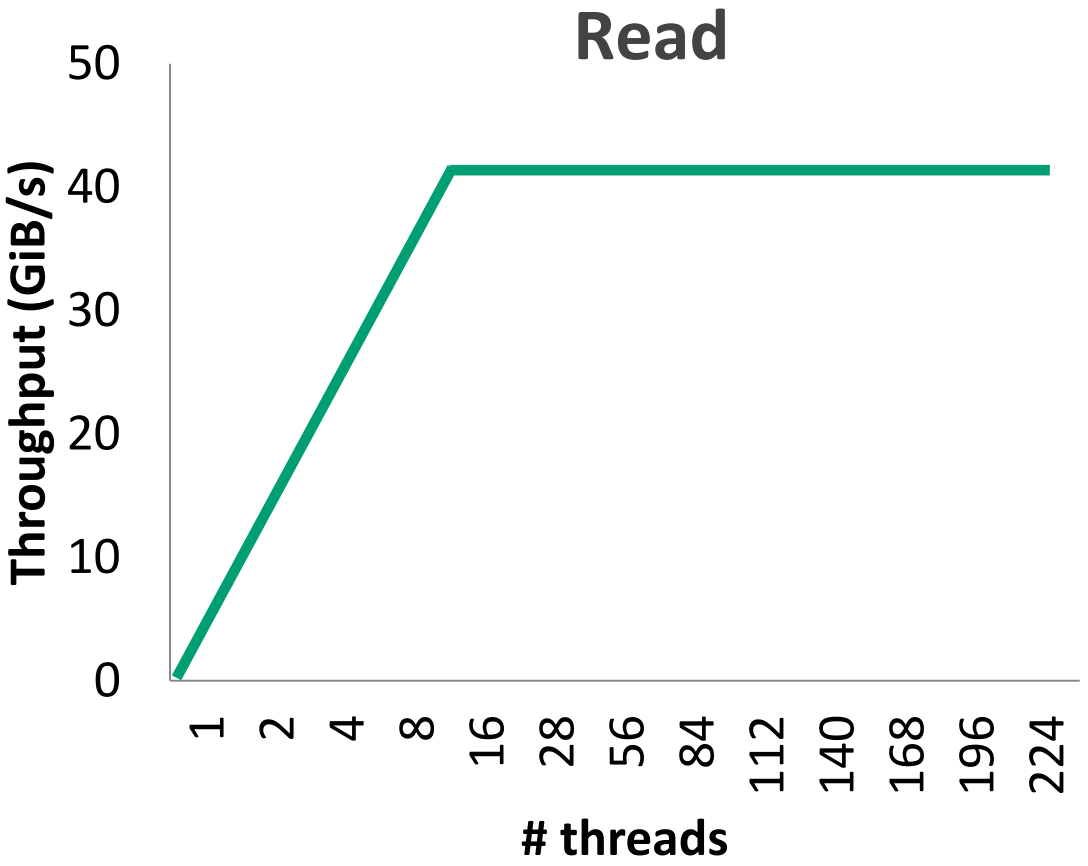
Setup: 224-core/8-socket machine

PM performance on a single NUMA node



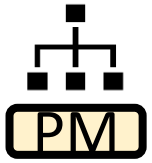
Workload: FIO: each thread writes/reads 2MB data in a private file



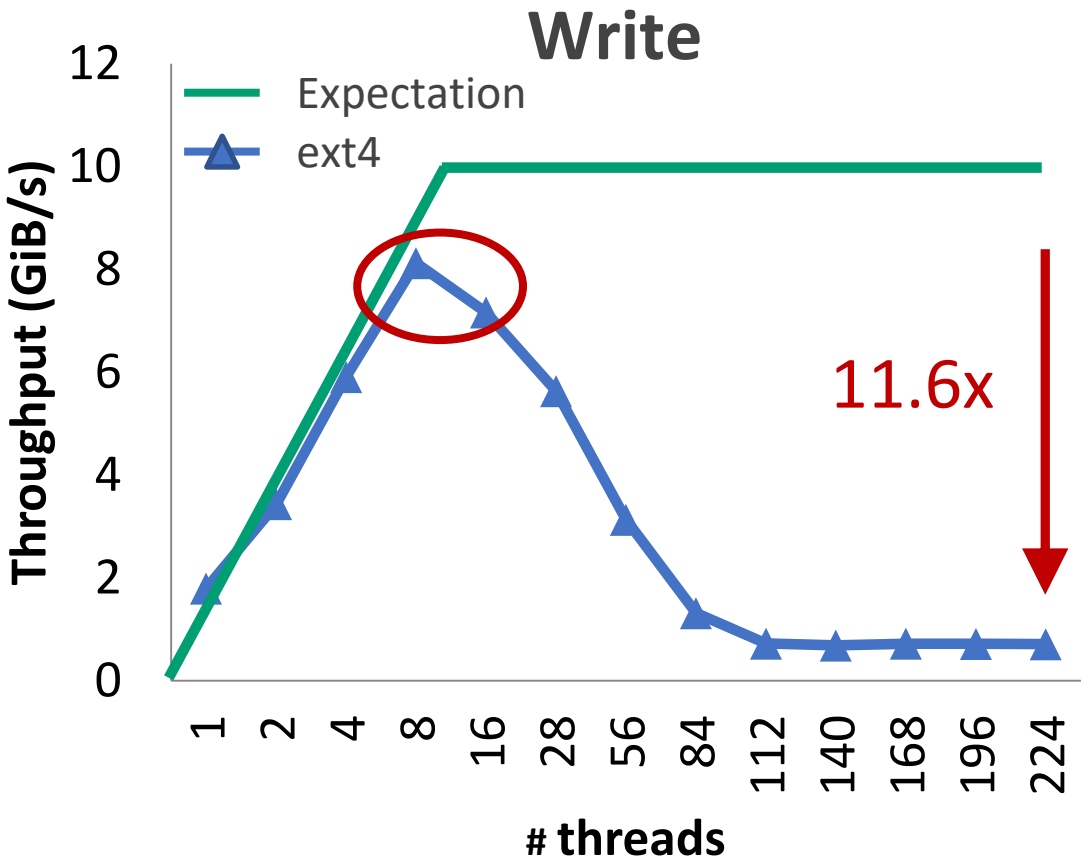


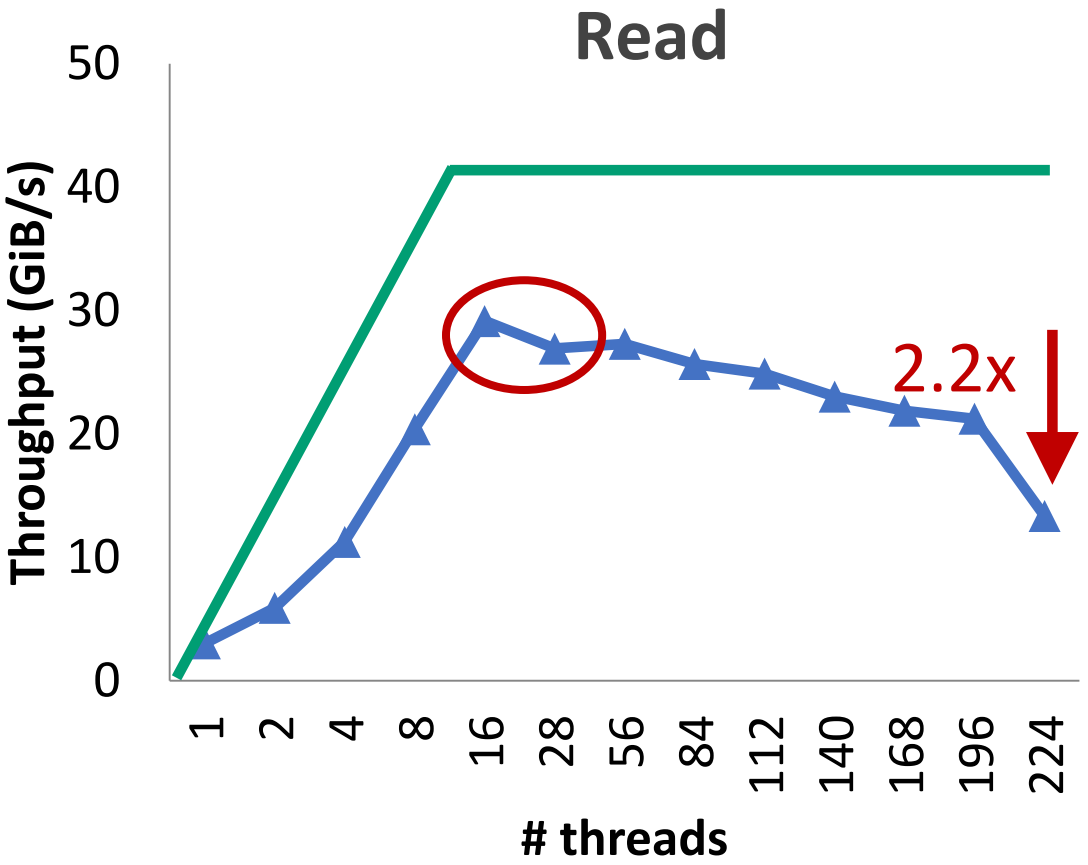
Setup: 224-core/8-socket machine

PM performance on a single NUMA node



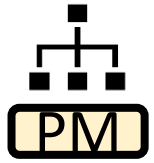
Workload: FIO: each thread writes/reads 2MB data in a private file





Setup: 224-core/8-socket machine

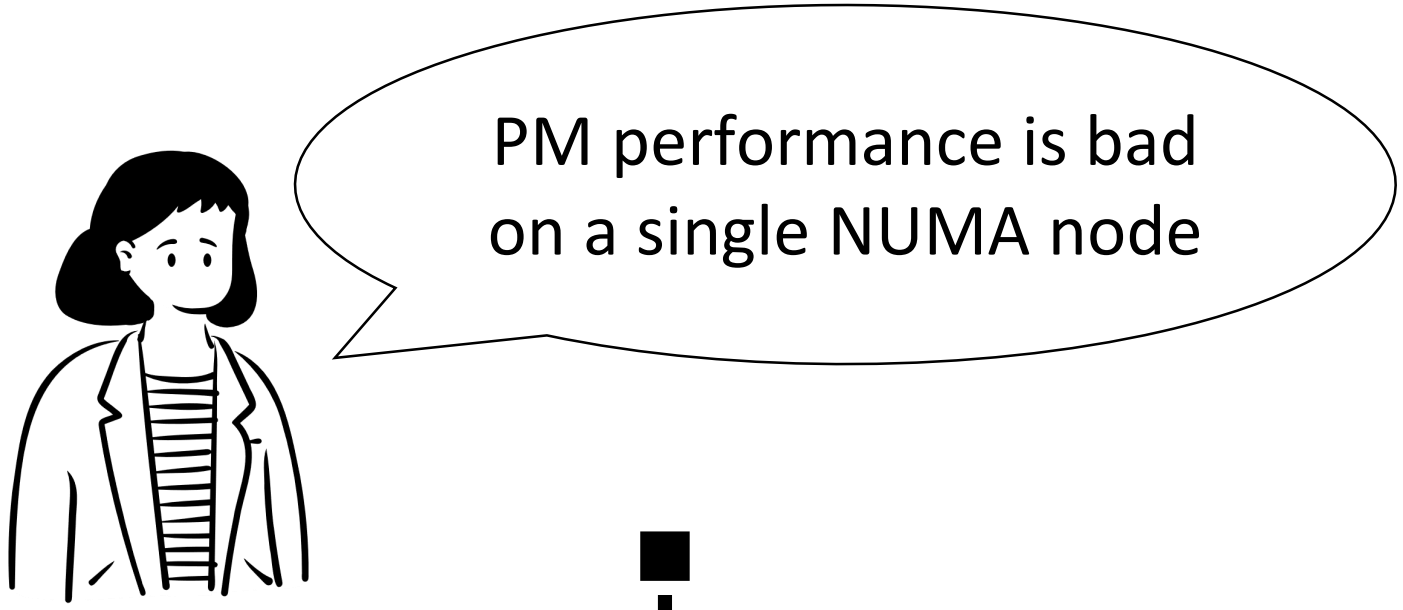
PM performance on a single NUMA node

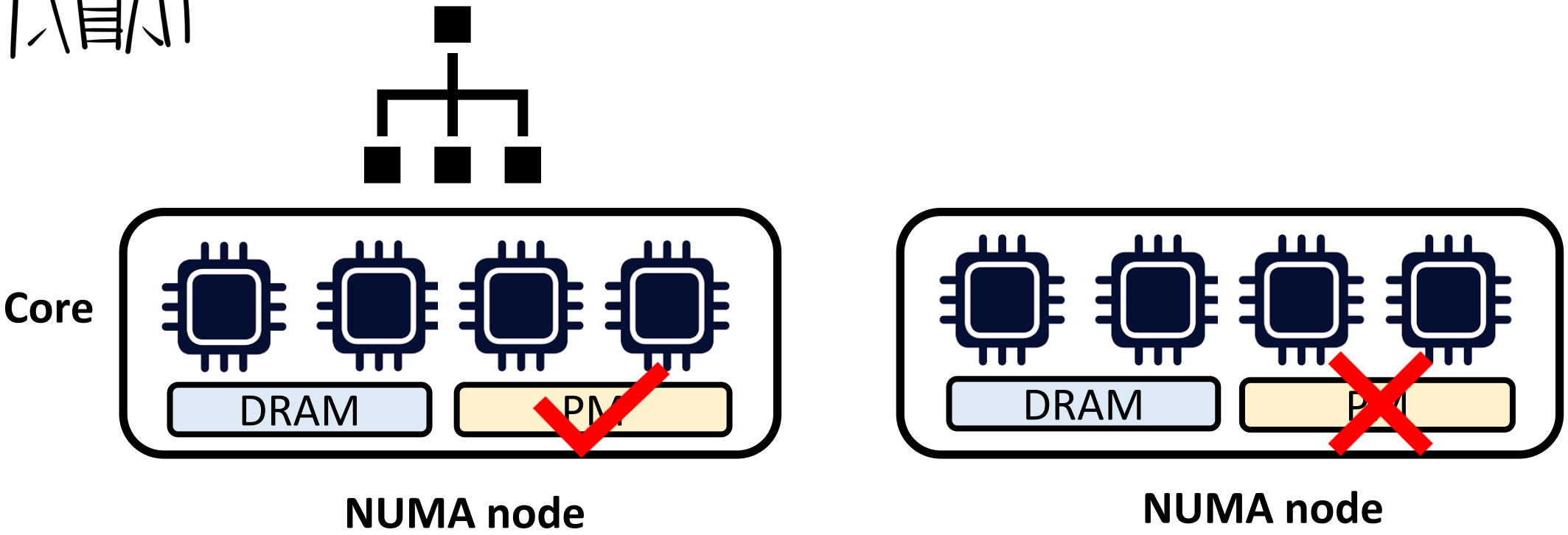


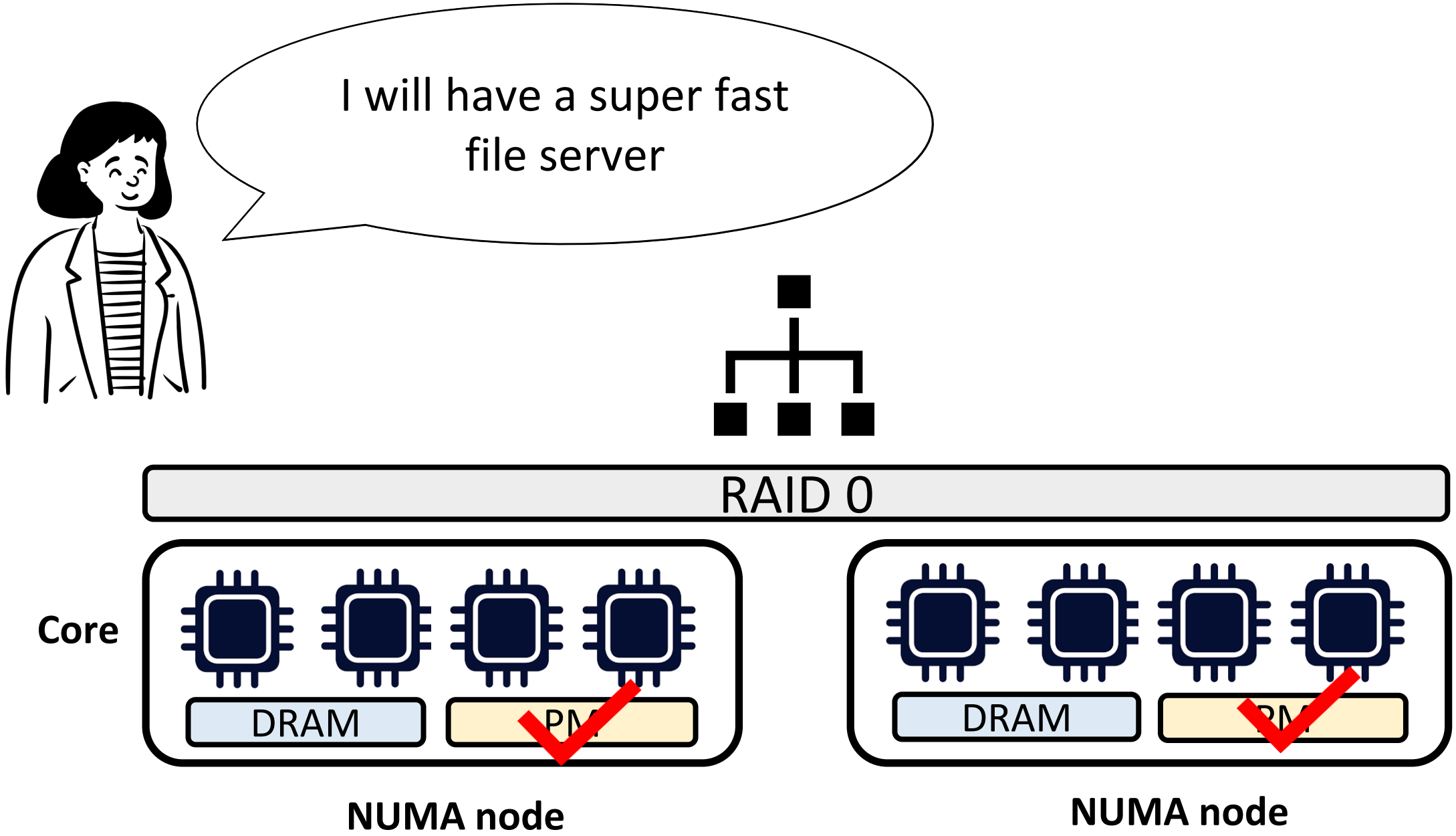
Workload: FIO: each thread writes/reads 2MB data in a private file

Excessive concurrent access
→ PM performance collapse

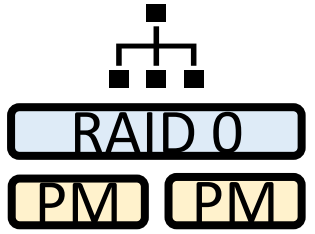
Setup: 224-core/8-socket machine


PM performance is bad
on a single NUMA node

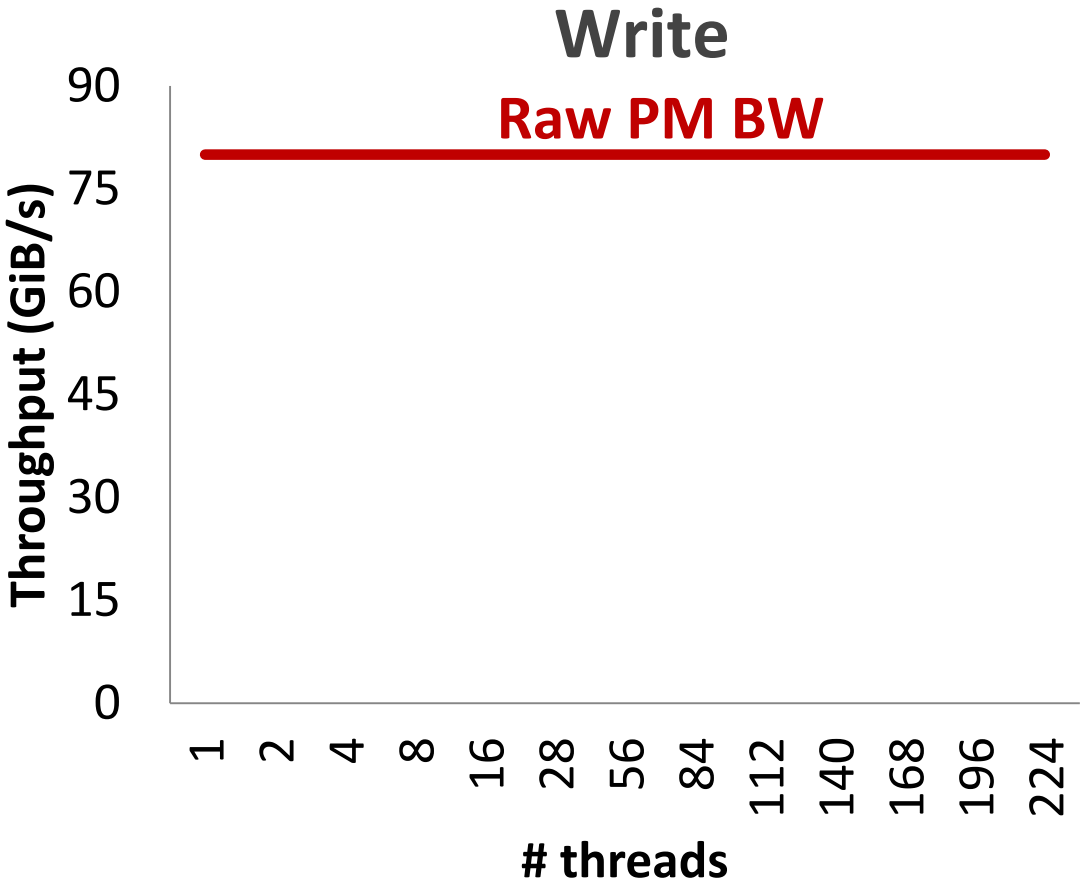

Core
DRAM
PM
NUMA node
DRAM
NUMA node

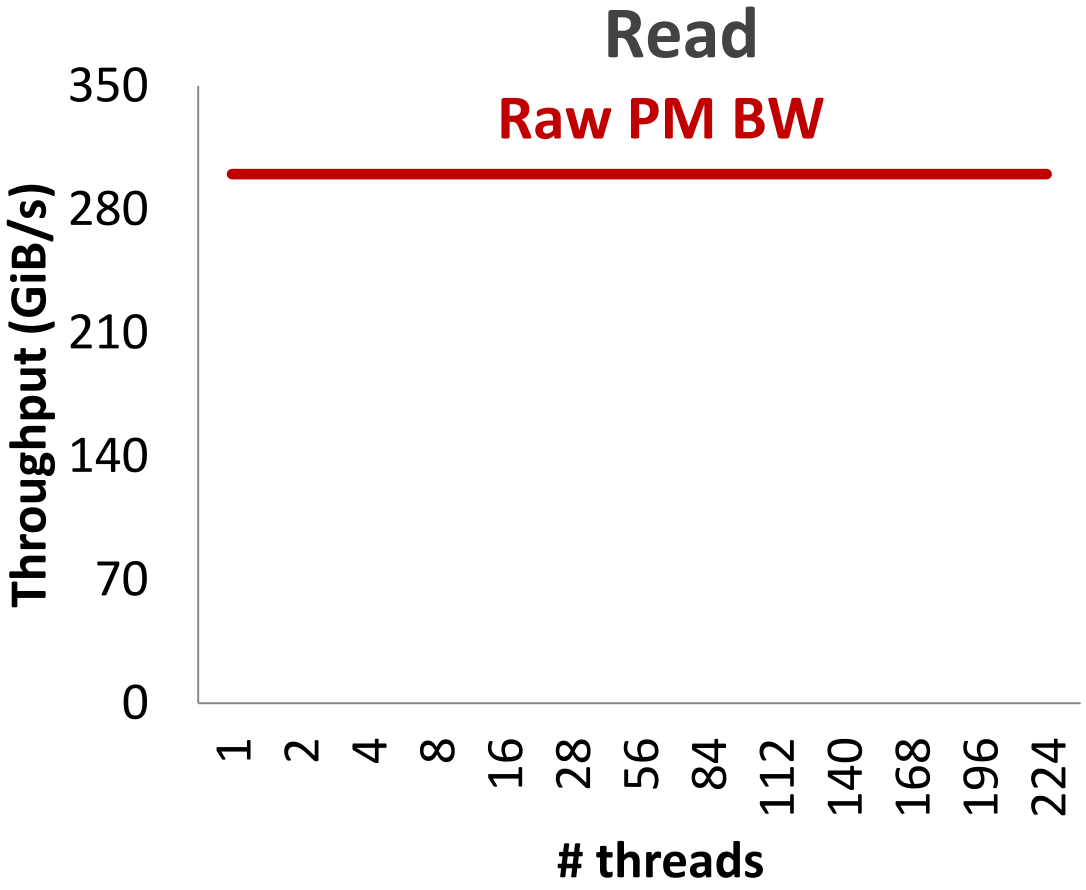

I will have a super fast file server
RAID 0
Core
DRAM
DRAM
NUMA node
NUMA node

PM performance on multiple NUMA nodes



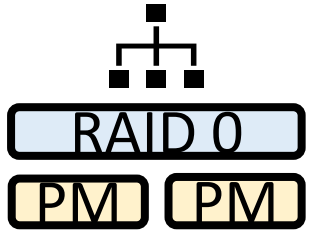
Workload: FIO: each thread writes/reads 2MB data in a private file



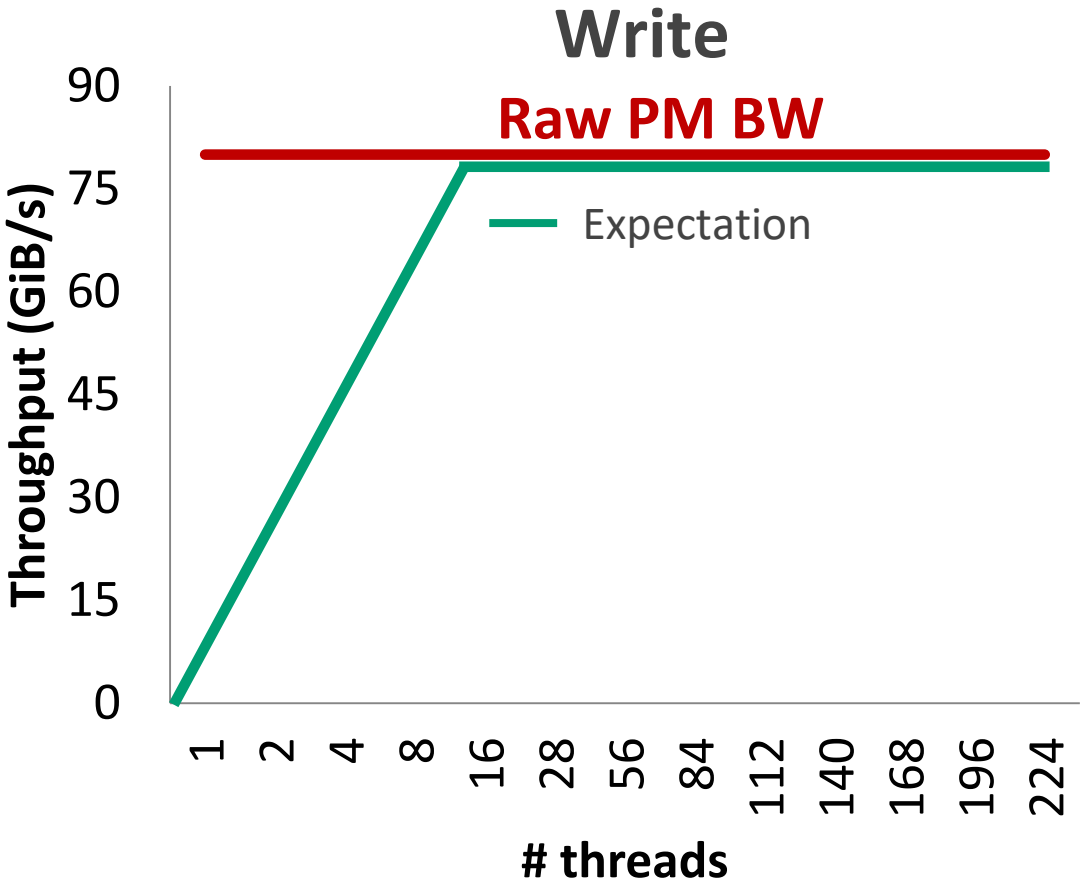


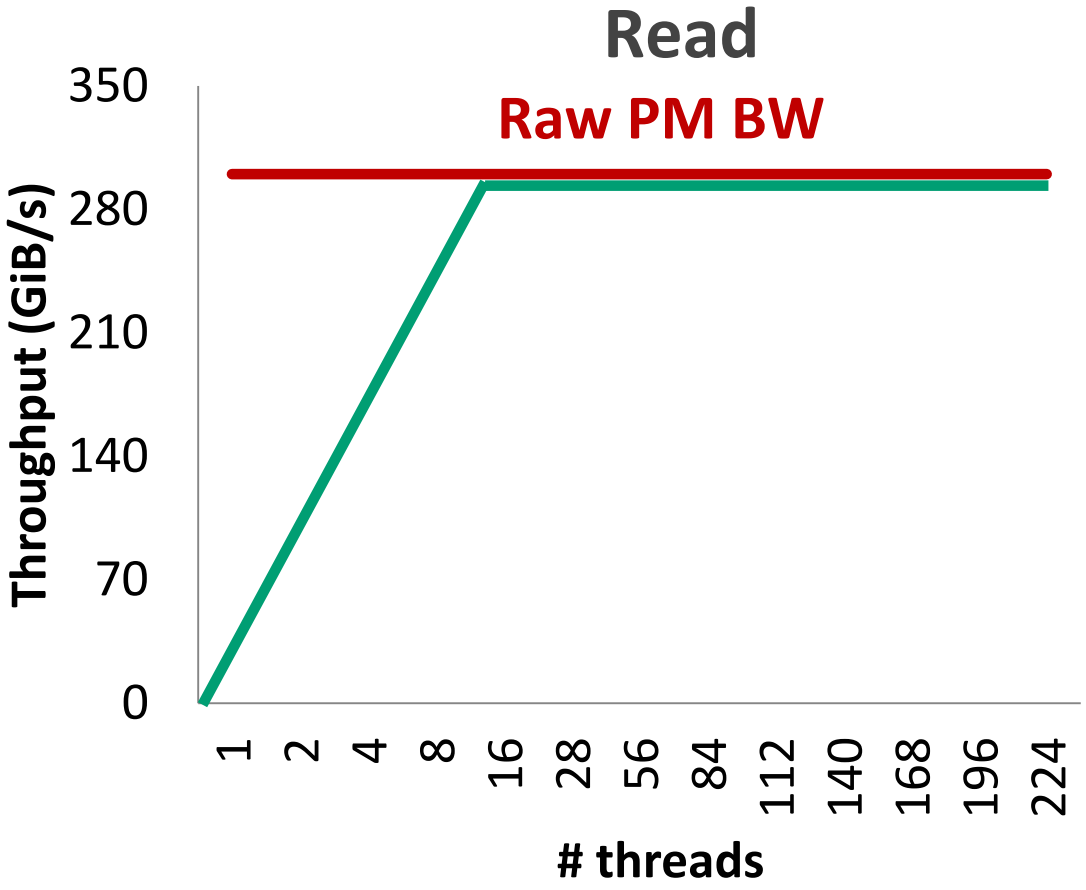
Setup: 224-core/8-socket machine

PM performance on multiple NUMA nodes



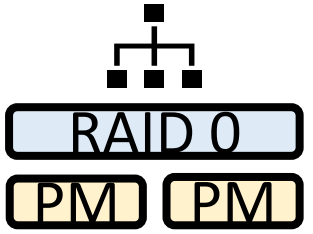
Workload: FIO: each thread writes/reads 2MB data in a private file



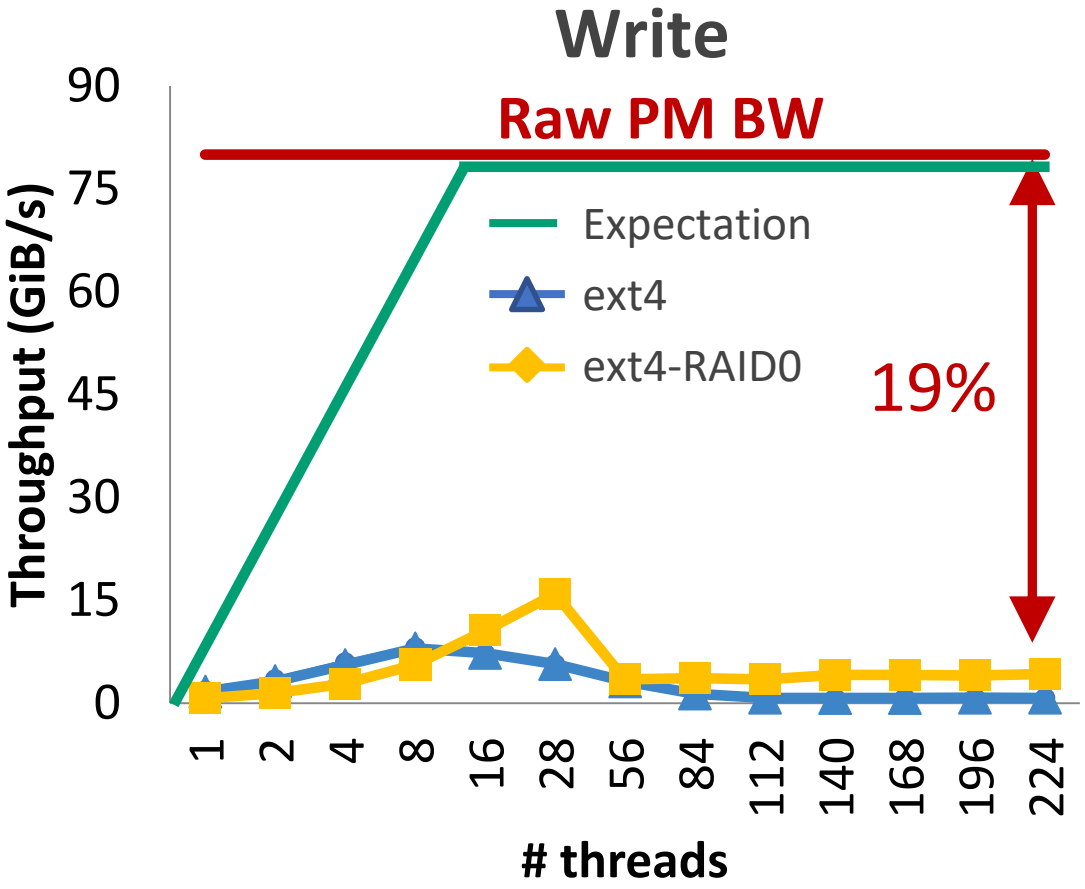


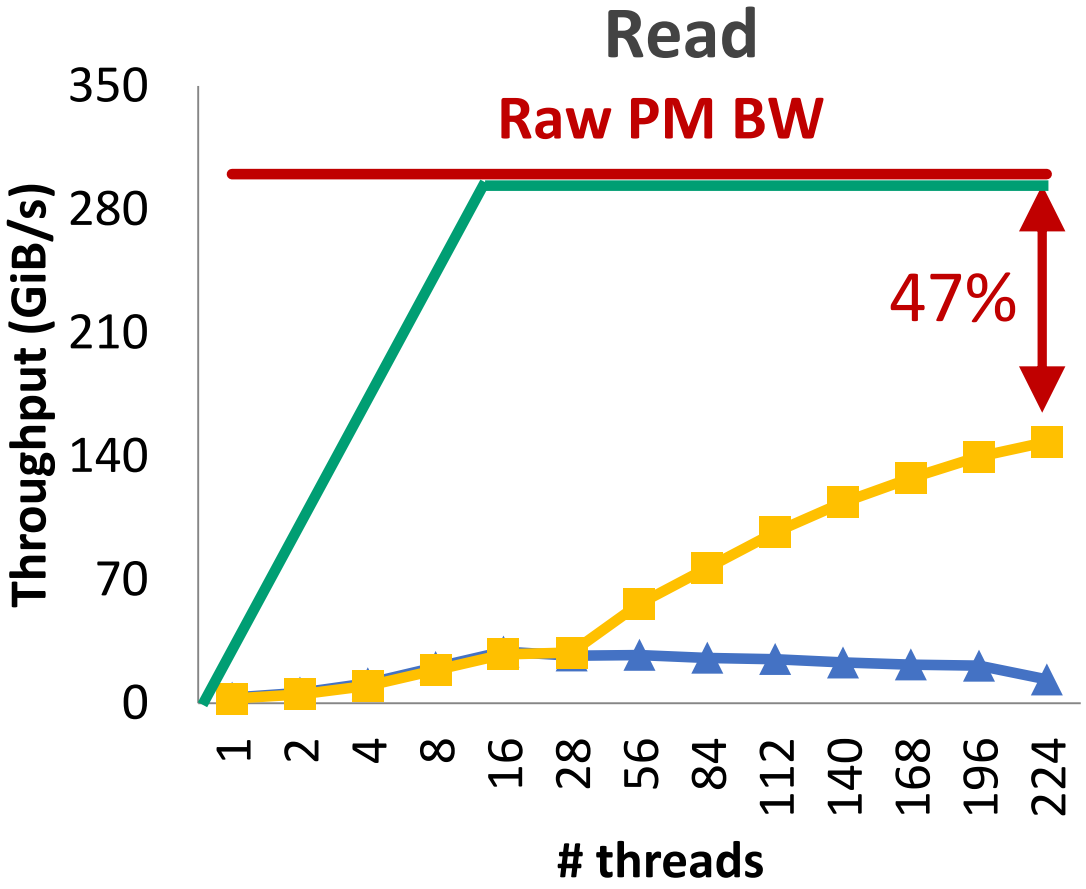
Setup: 224-core/8-socket machine

PM performance on multiple NUMA nodes



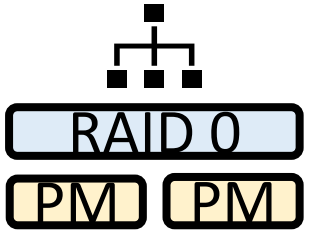
Workload: FIO: each thread writes/reads 2MB data in a private file



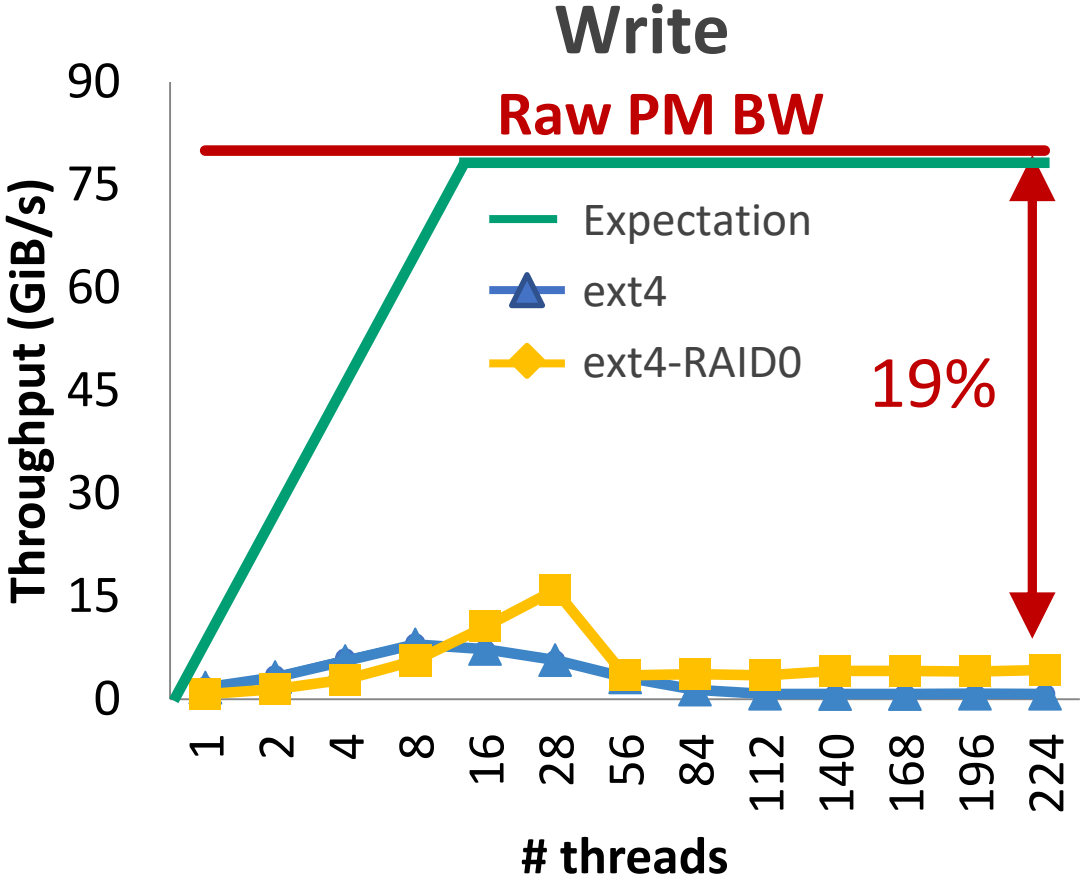


Setup: 224-core/8-socket machine

PM performance on multiple NUMA nodes



Workload: FIO: each thread writes/reads 2MB data in a private file



Inefficient remote PM access

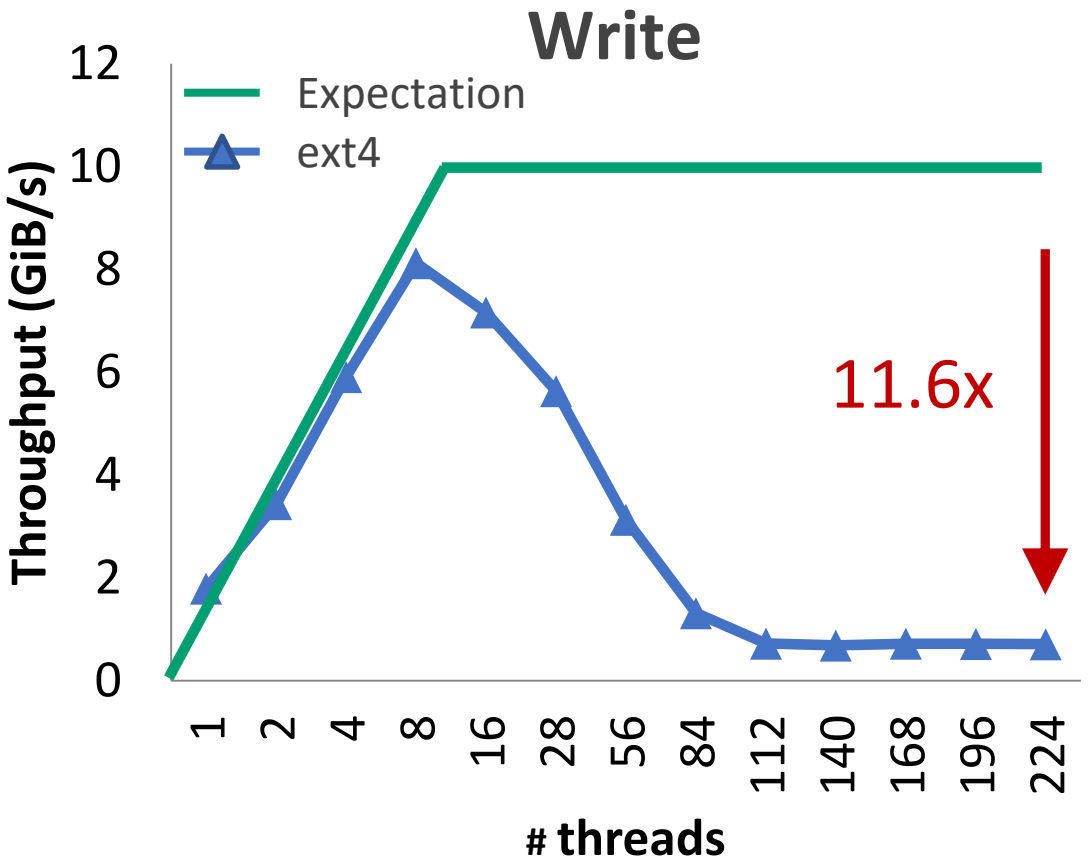
Setup: 224-core/8-socket machine

Design goal of OdinFS:

Maximize PM performance

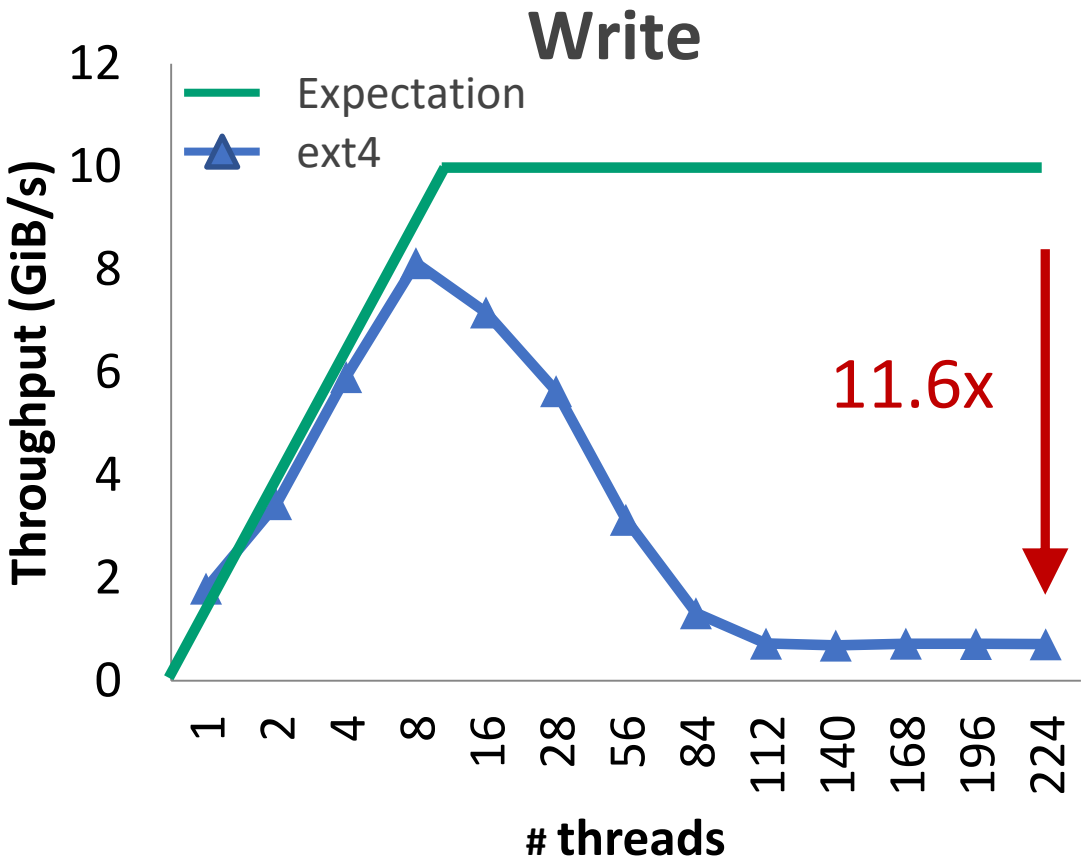
Maintain PM performance with concurrent access

Single NUMA node: Why PM performance collapse



Excessive concurrent access → PM performance collapse

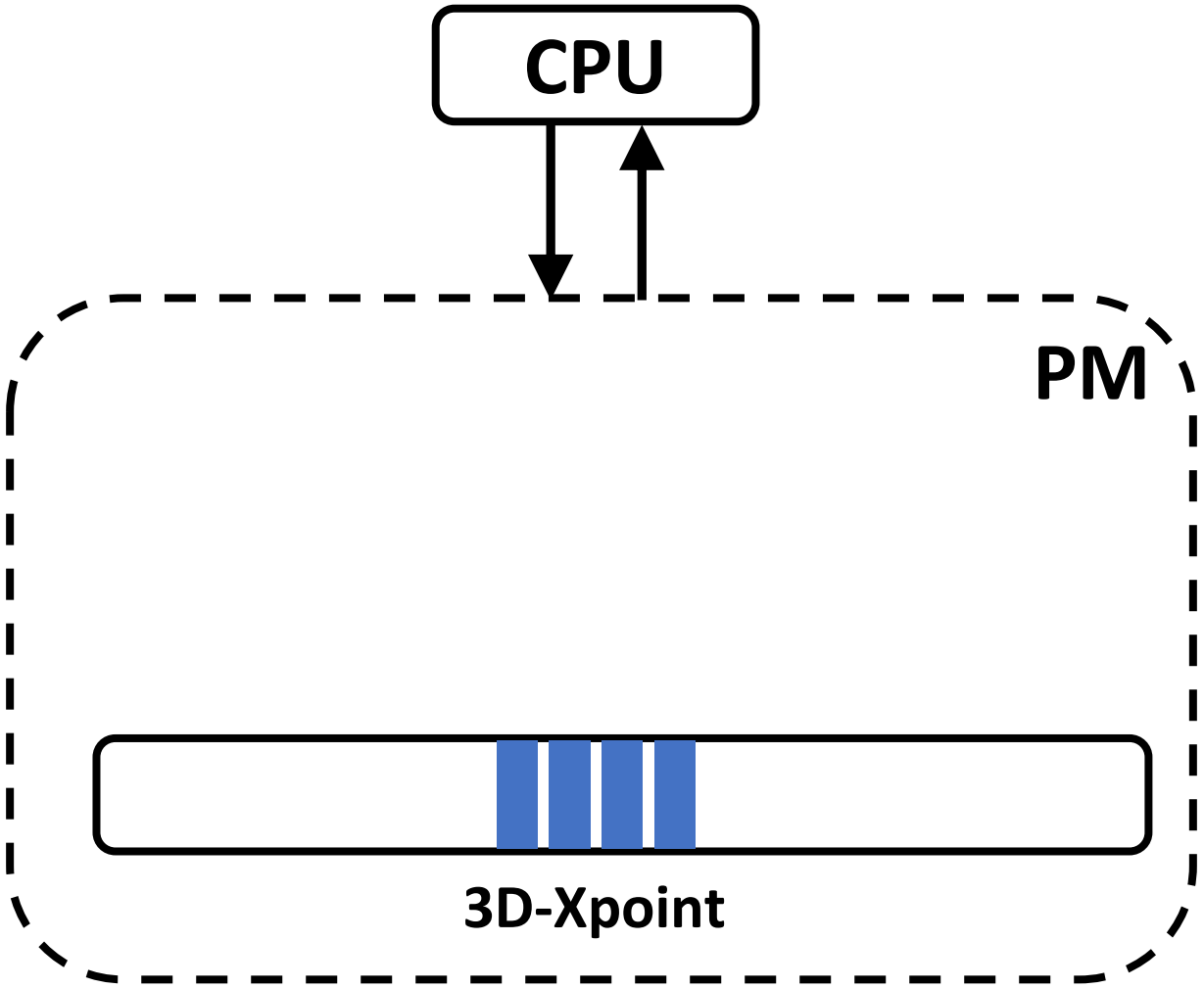
Single NUMA node: Why PM performance collapse



Excessive concurrent access → PM performance collapse

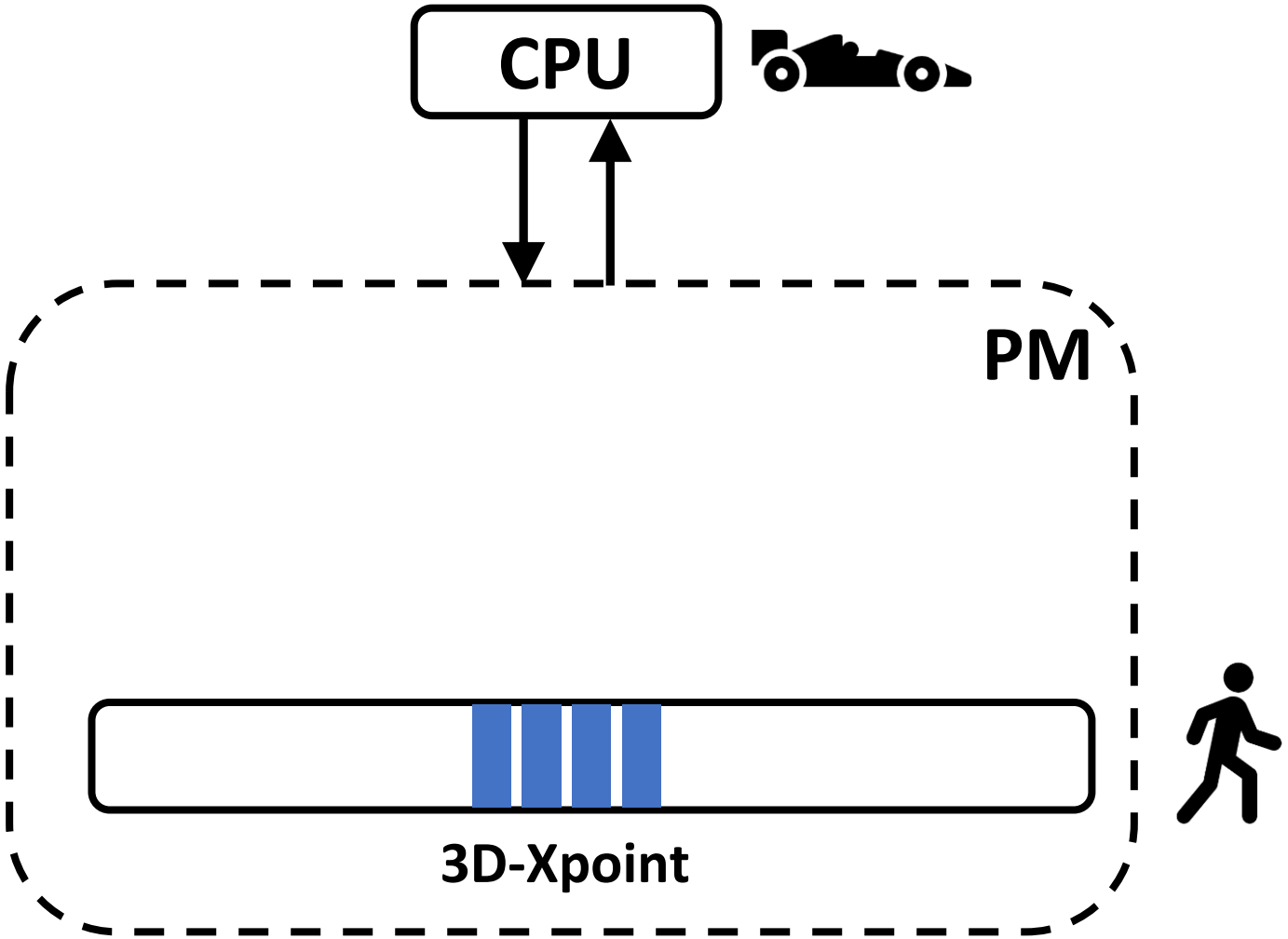
On-DIMM cache thrashing

Caching and prefetching in PM



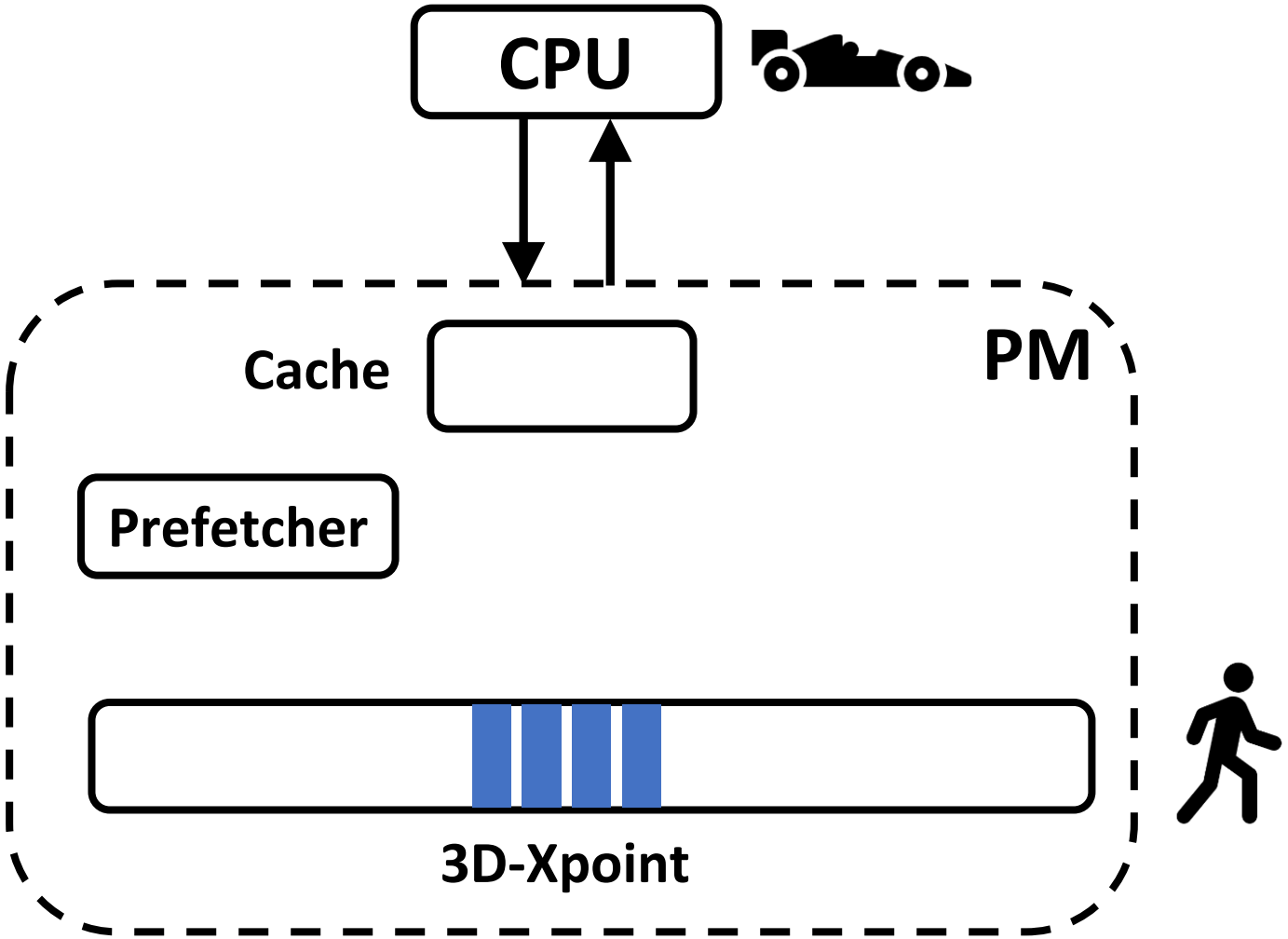
Hide performance gap
with caching & prefetching

Caching and prefetching in PM



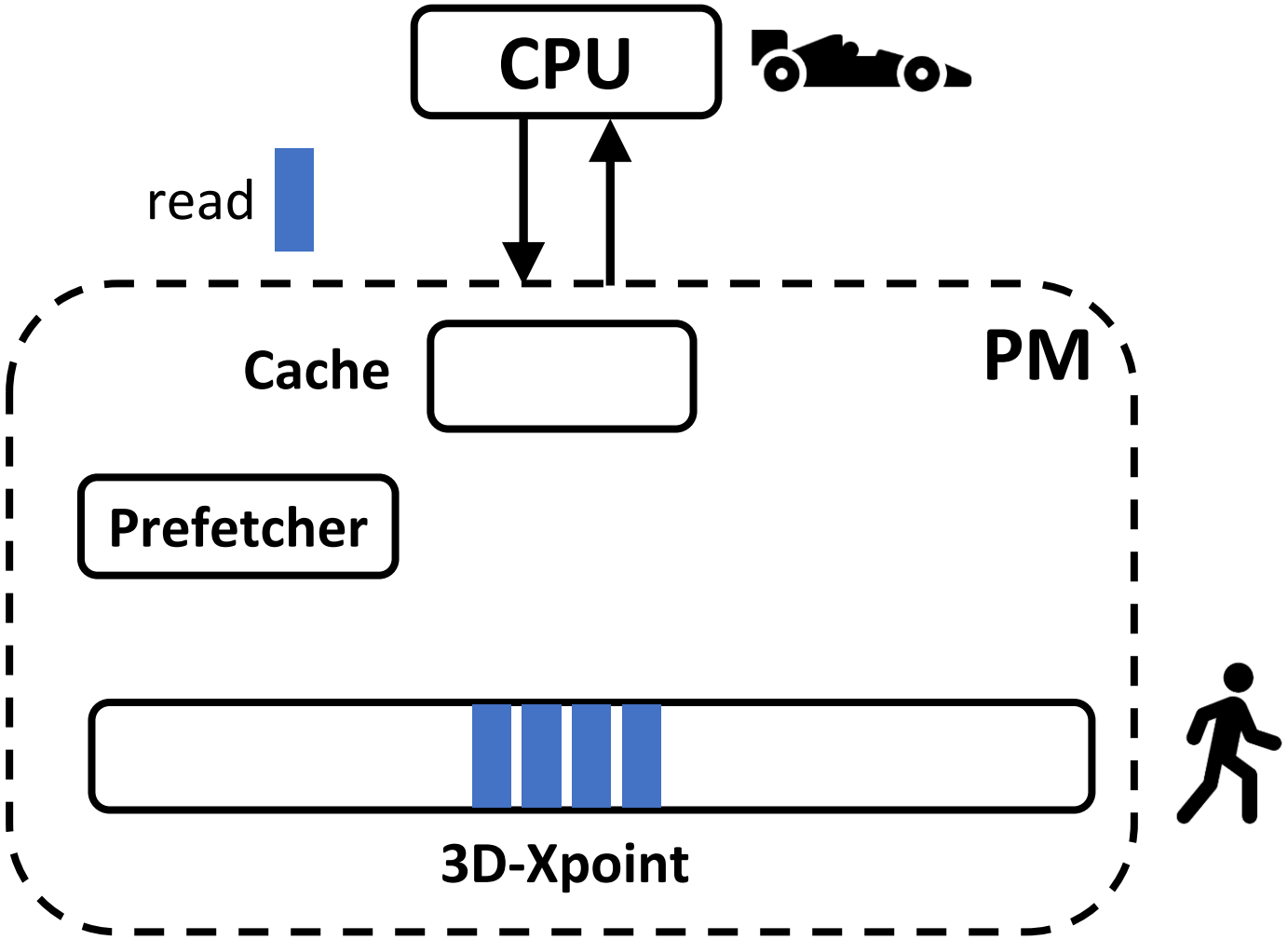
Hide performance gap
with caching & prefetching

Caching and prefetching in PM



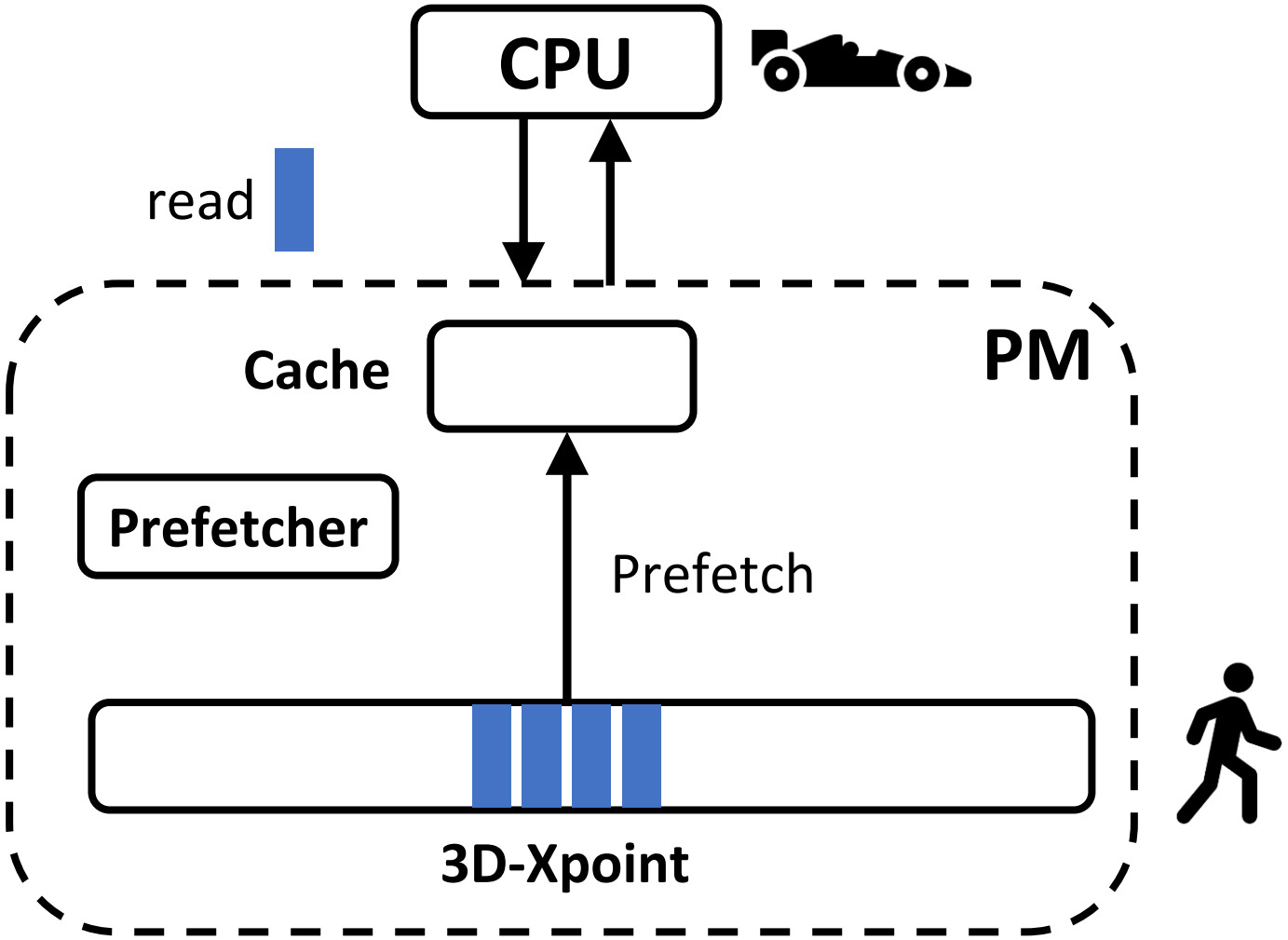
Hide performance gap
with caching & prefetching

Caching and prefetching in PM



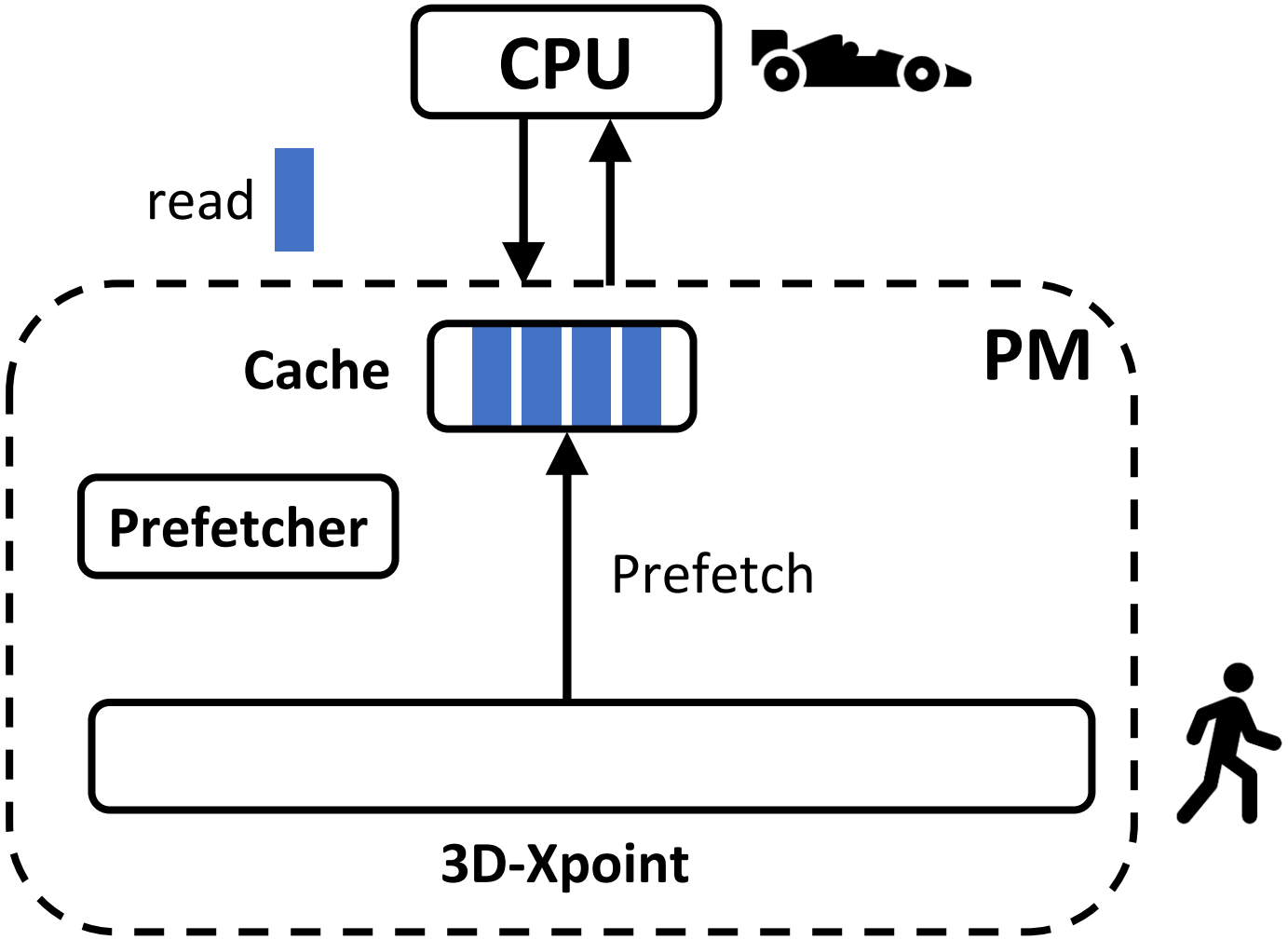
Hide performance gap
with caching & prefetching

Caching and prefetching in PM



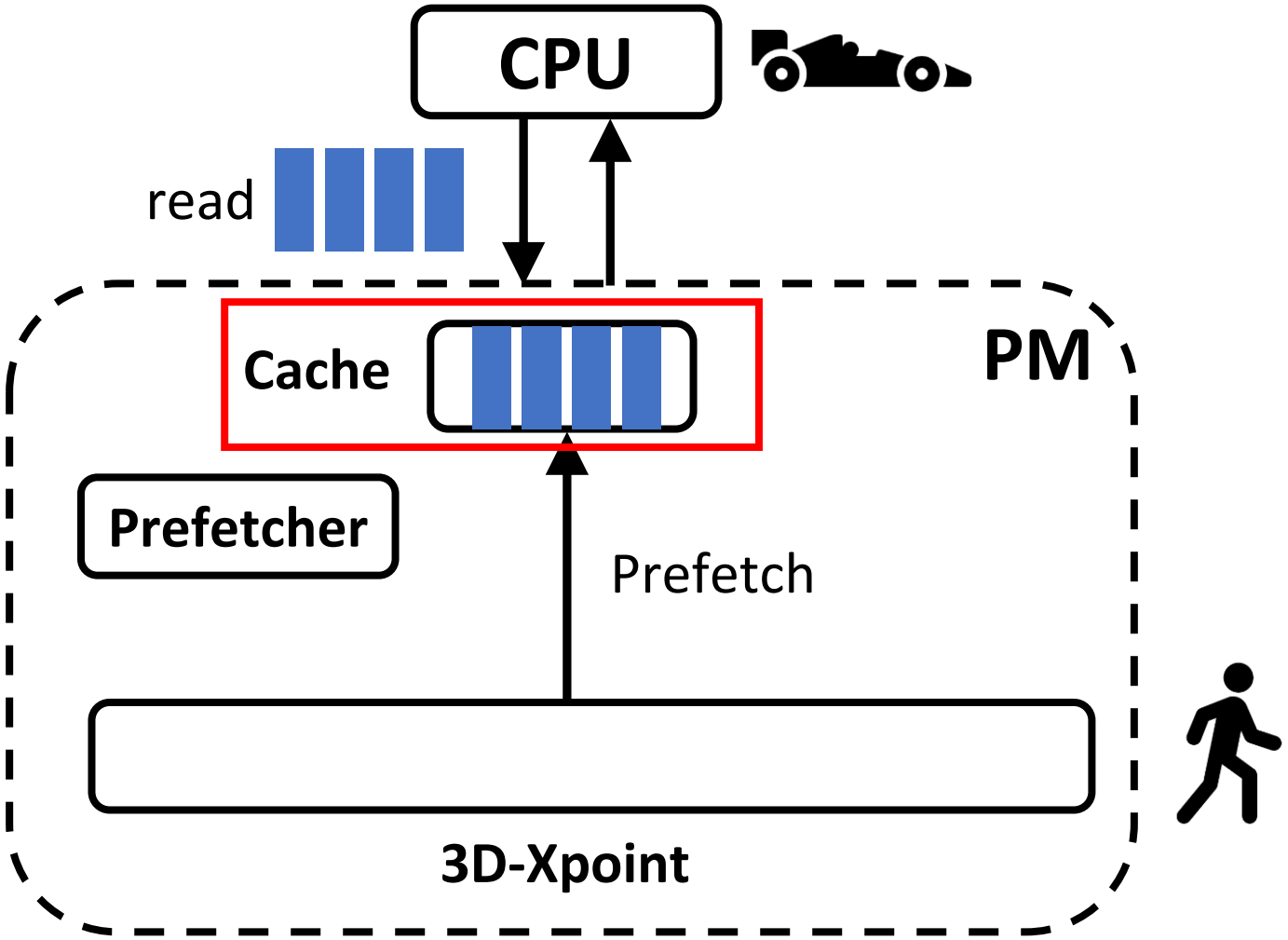
Hide performance gap
with caching & prefetching

Caching and prefetching in PM



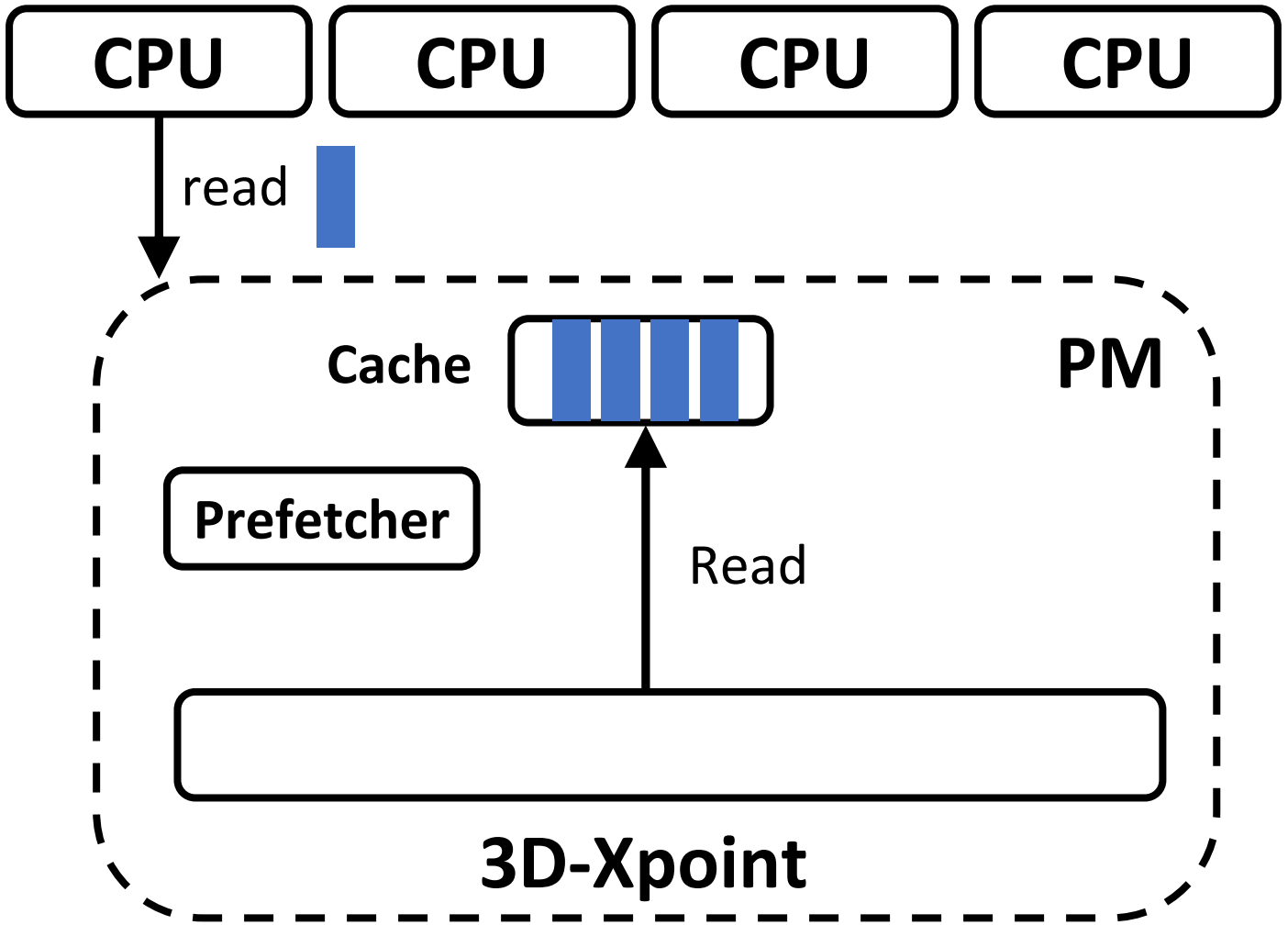
Hide performance gap
with caching & prefetching

Caching and prefetching in PM



Hide performance gap
with caching & prefetching

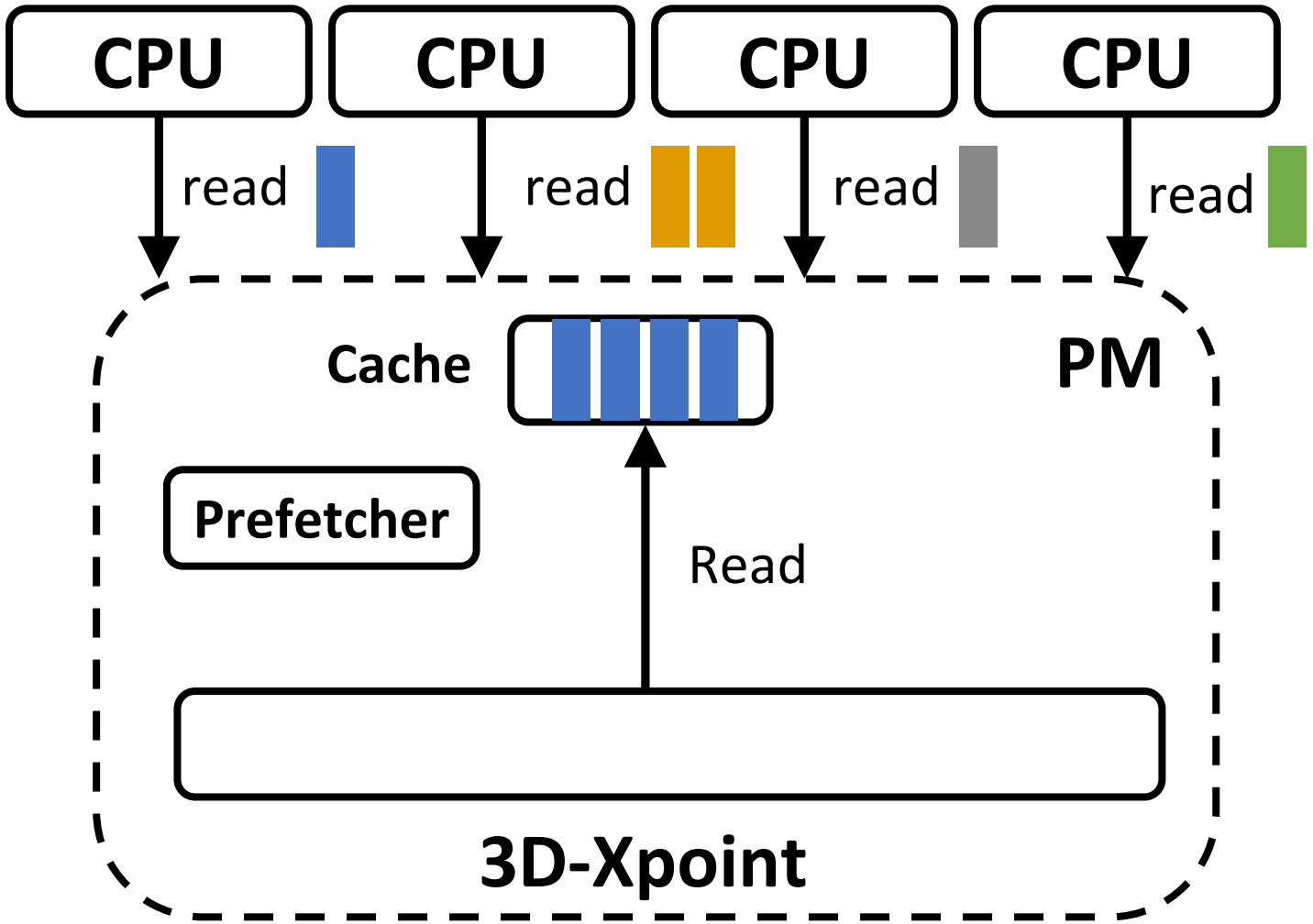
PM cache thrashing due to concurrent access



Hide performance gap
with caching & prefetching

Excessive concurrent access
→ PM cache thrashing

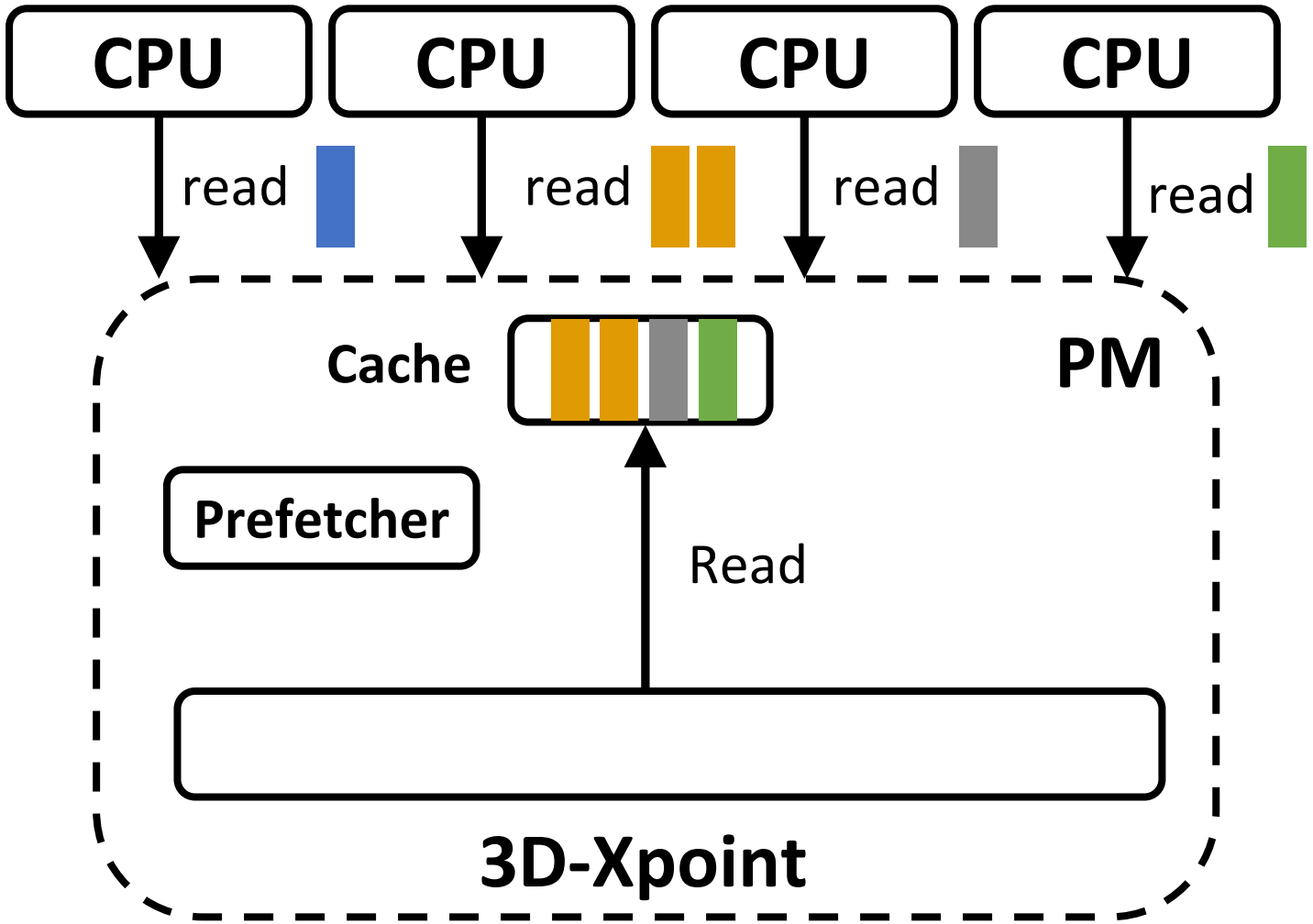
PM cache thrashing due to concurrent access



Hide performance gap with caching & prefetching

Excessive concurrent access → PM cache thrashing

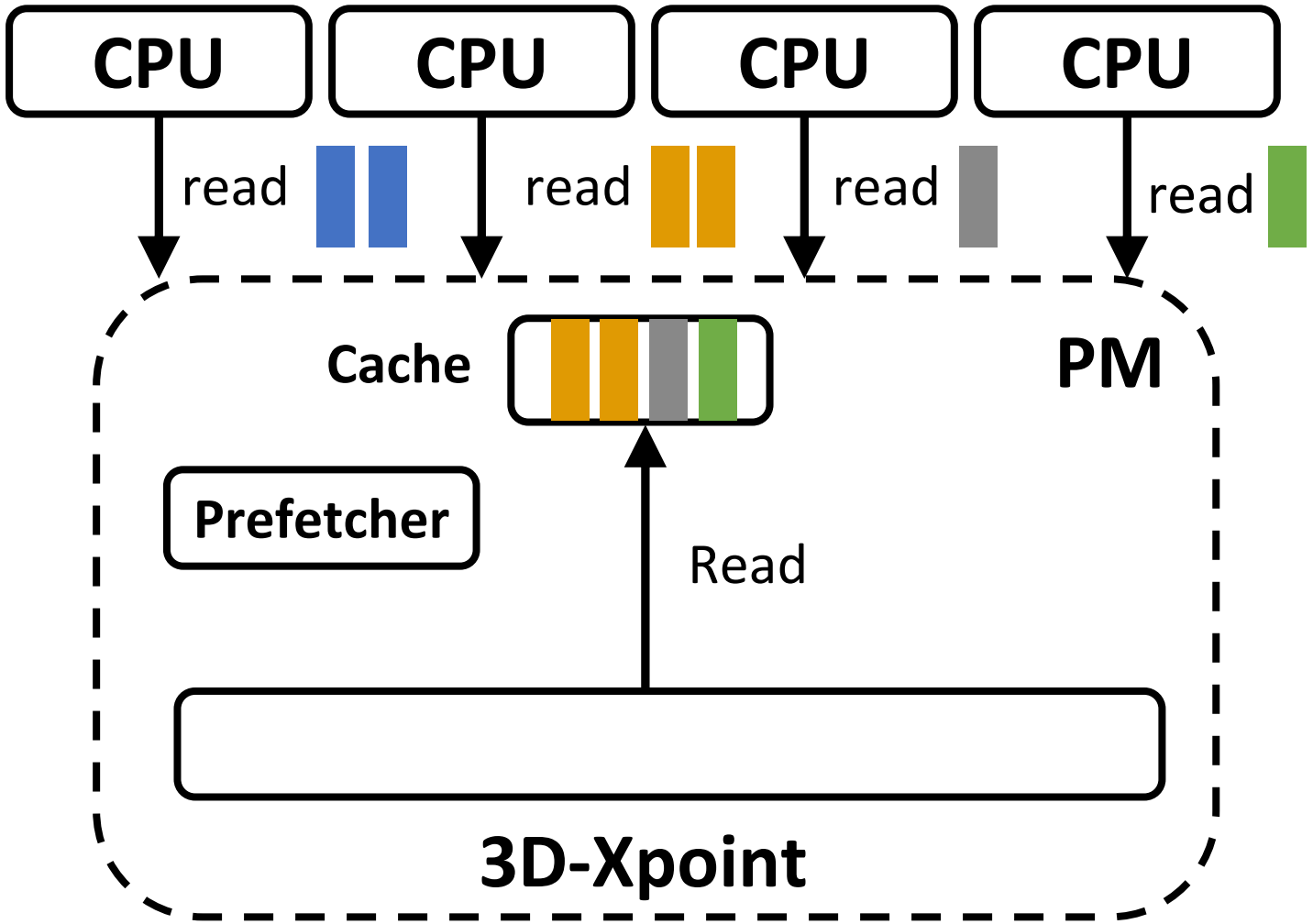
PM cache thrashing due to concurrent access



Hide performance gap with caching & prefetching

Excessive concurrent access → PM cache thrashing

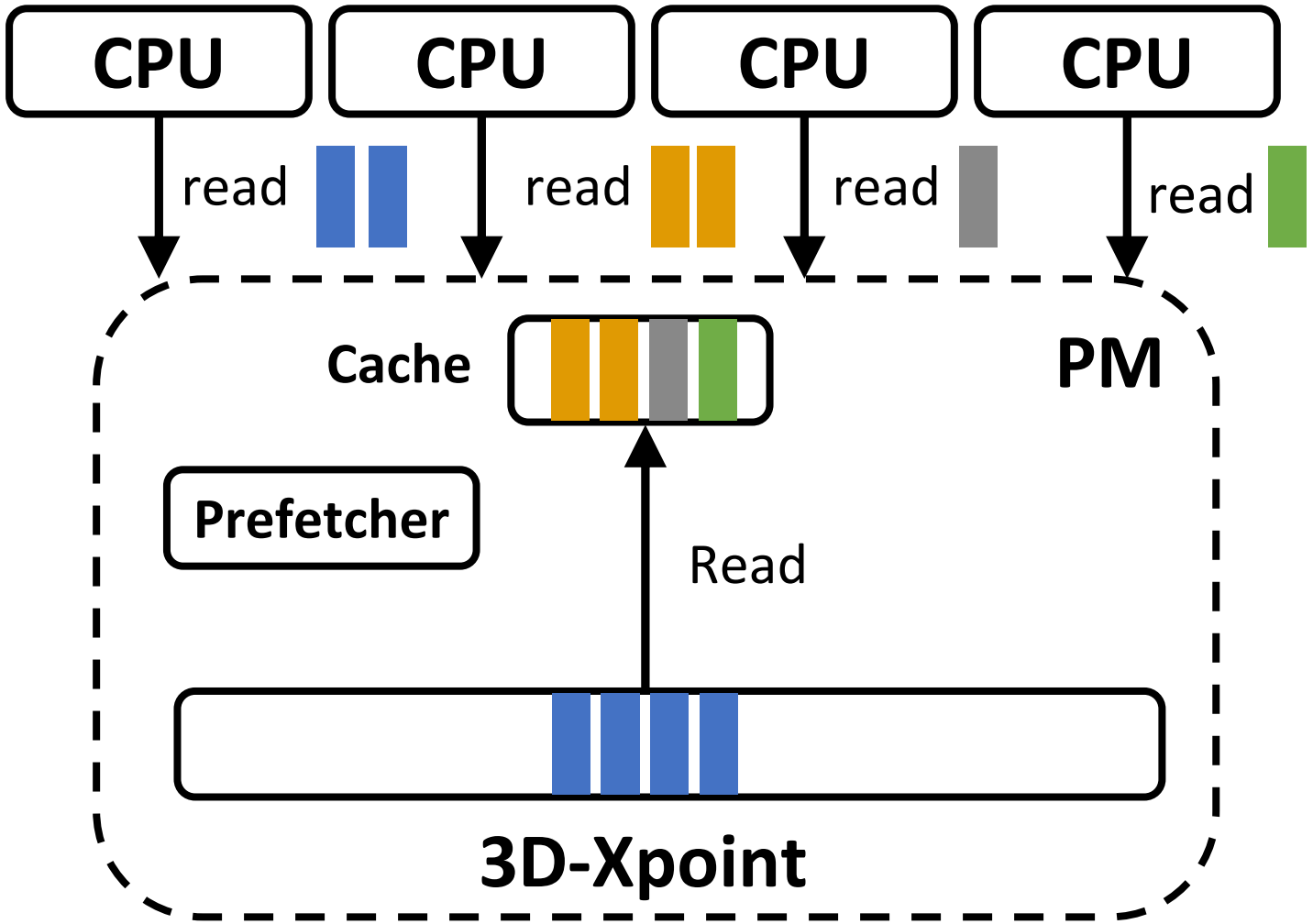
PM cache thrashing due to concurrent access



Hide performance gap with caching & prefetching

Excessive concurrent access → PM cache thrashing

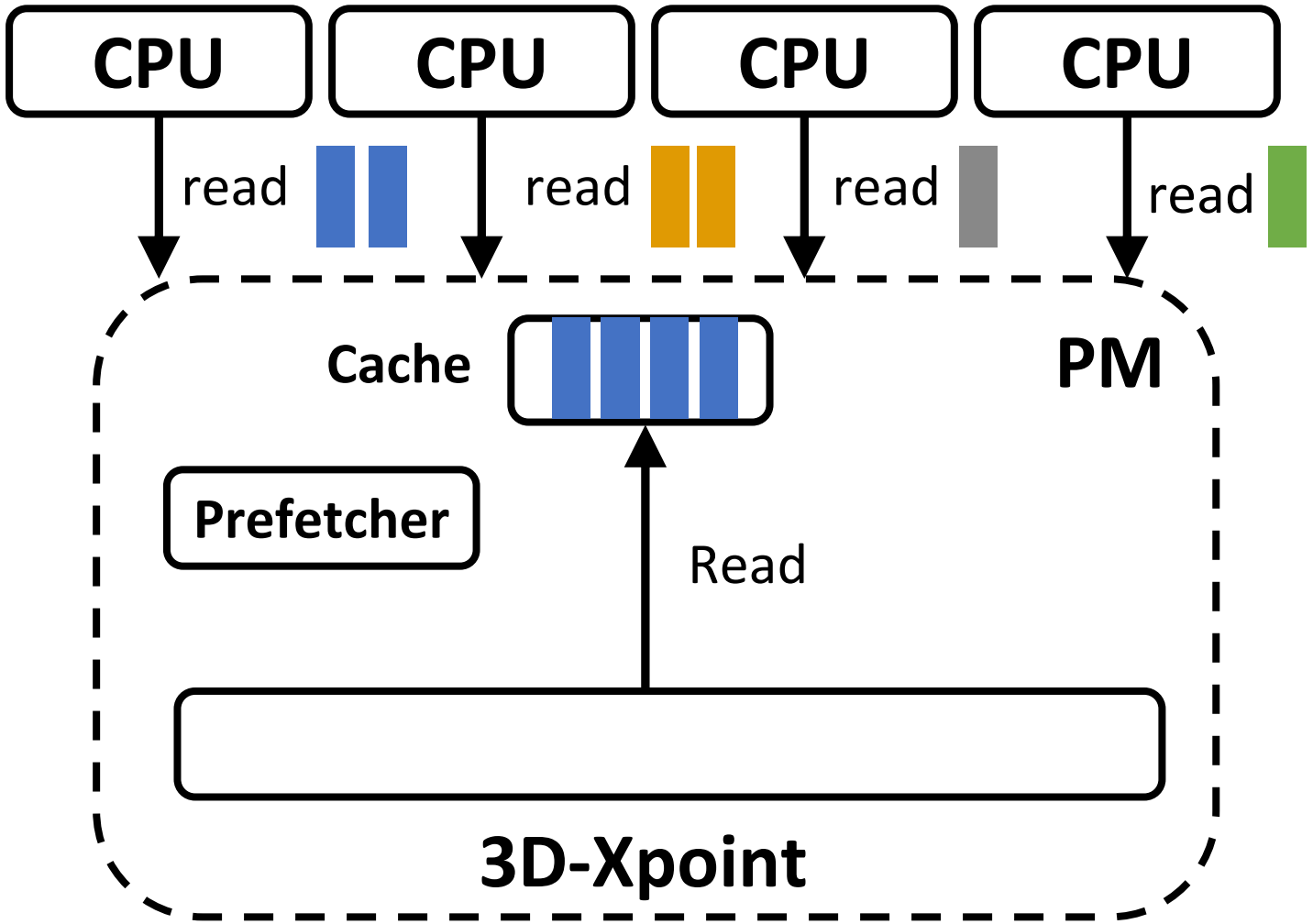
PM cache thrashing due to concurrent access



Hide performance gap with caching & prefetching

Excessive concurrent access → PM cache thrashing

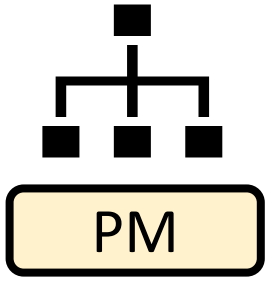
PM cache thrashing due to concurrent access



Hide performance gap with caching & prefetching

Excessive concurrent access → PM cache thrashing

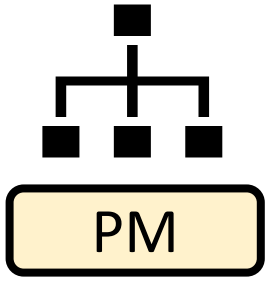
PM performance analysis



Issue: Excessive concurrent access → PM performance collapse

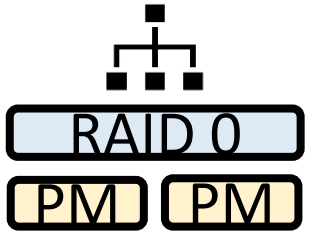
Insight: Control concurrent PM access for maximal performance

PM performance analysis



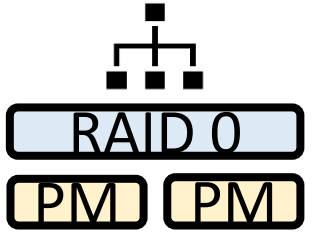
Issue: Excessive concurrent access → PM performance collapse

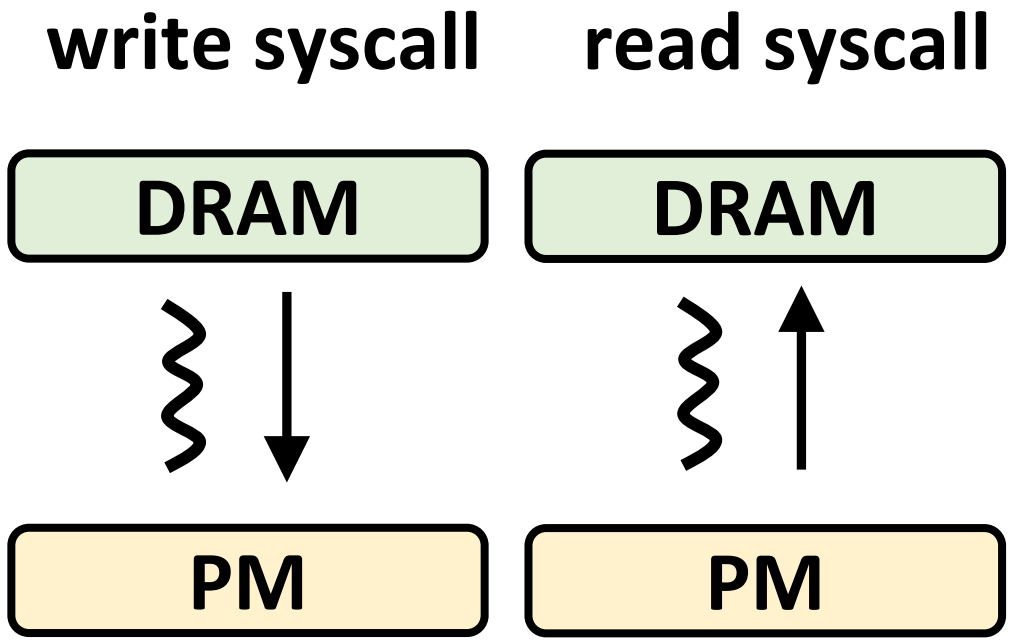
Insight: Control concurrent PM access for maximal performance



Issue: Inefficient remote PM access

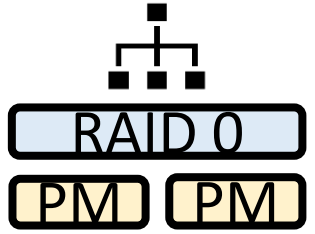
I/O operations in a PM file system

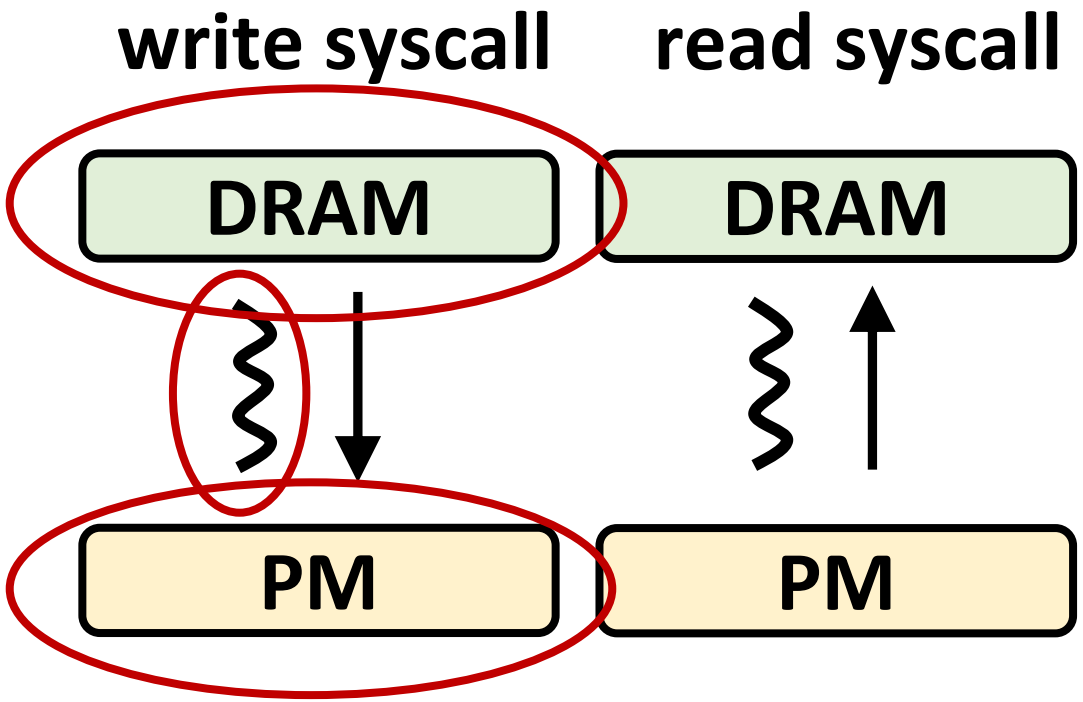




Data transfer between DRAM and PM

I/O operations in a PM file system

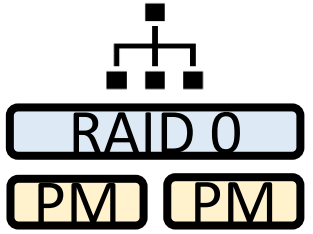


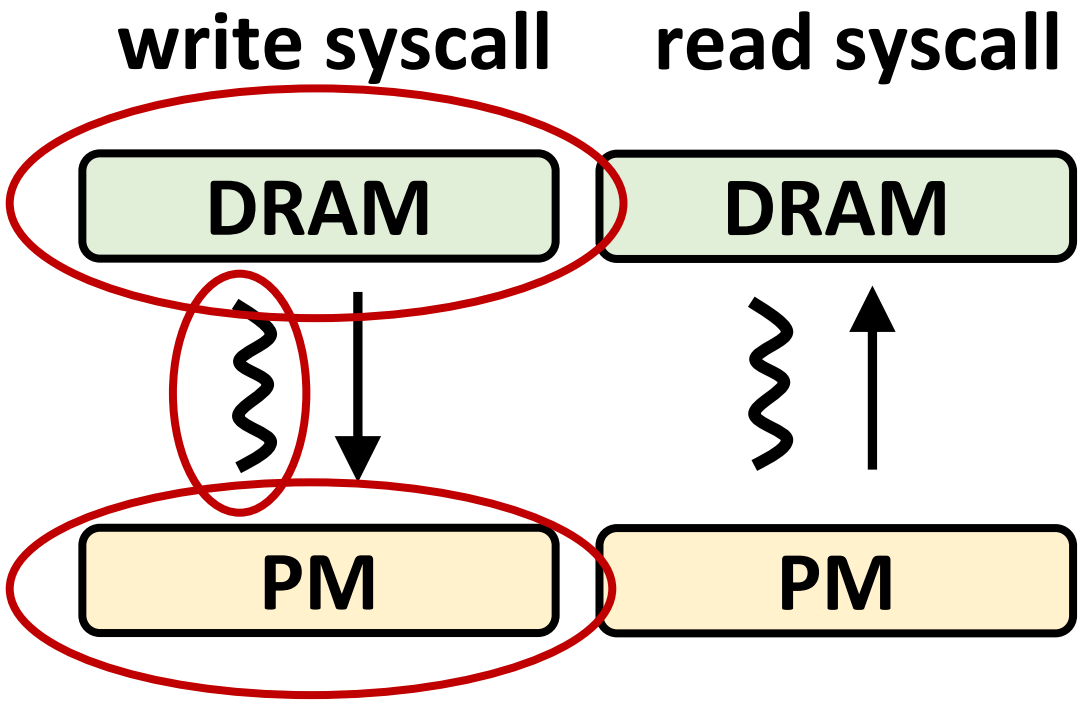


Data transfer between DRAM and PM

Components: DRAM, PM, and thread

I/O operations in a PM file system



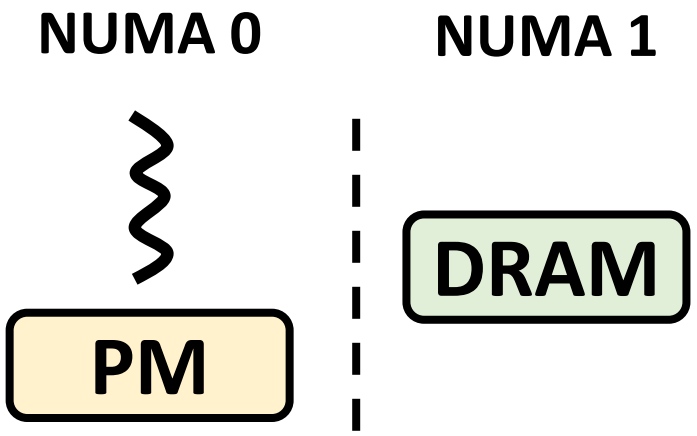


Data transfer between DRAM and PM

Components: DRAM, PM, and thread

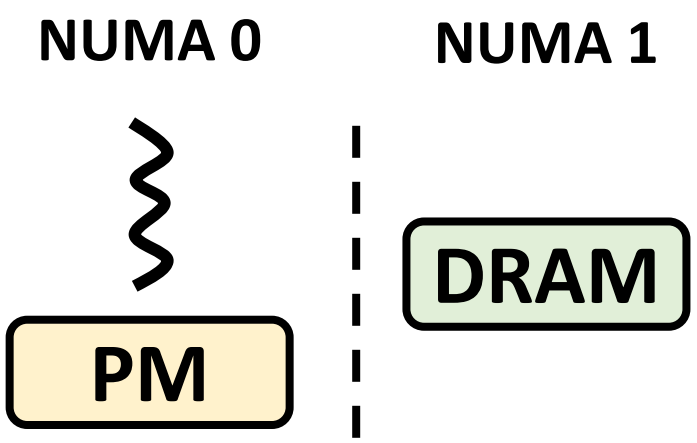
How does the NUMA placement of the components affect performance?

NUMA placement setup

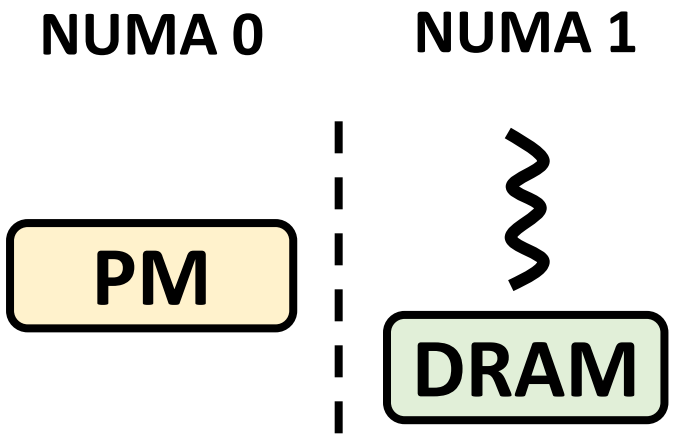


PM-local

NUMA placement setup

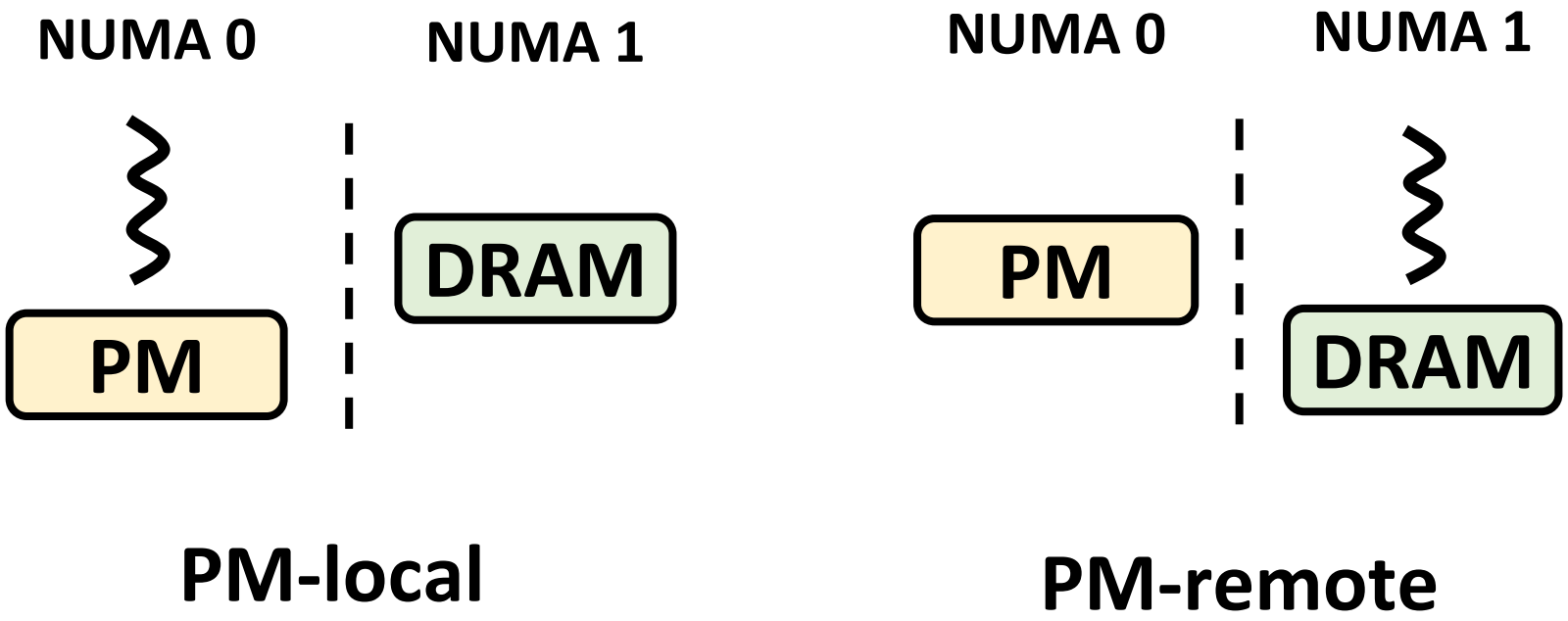


PM-local



PM-remote

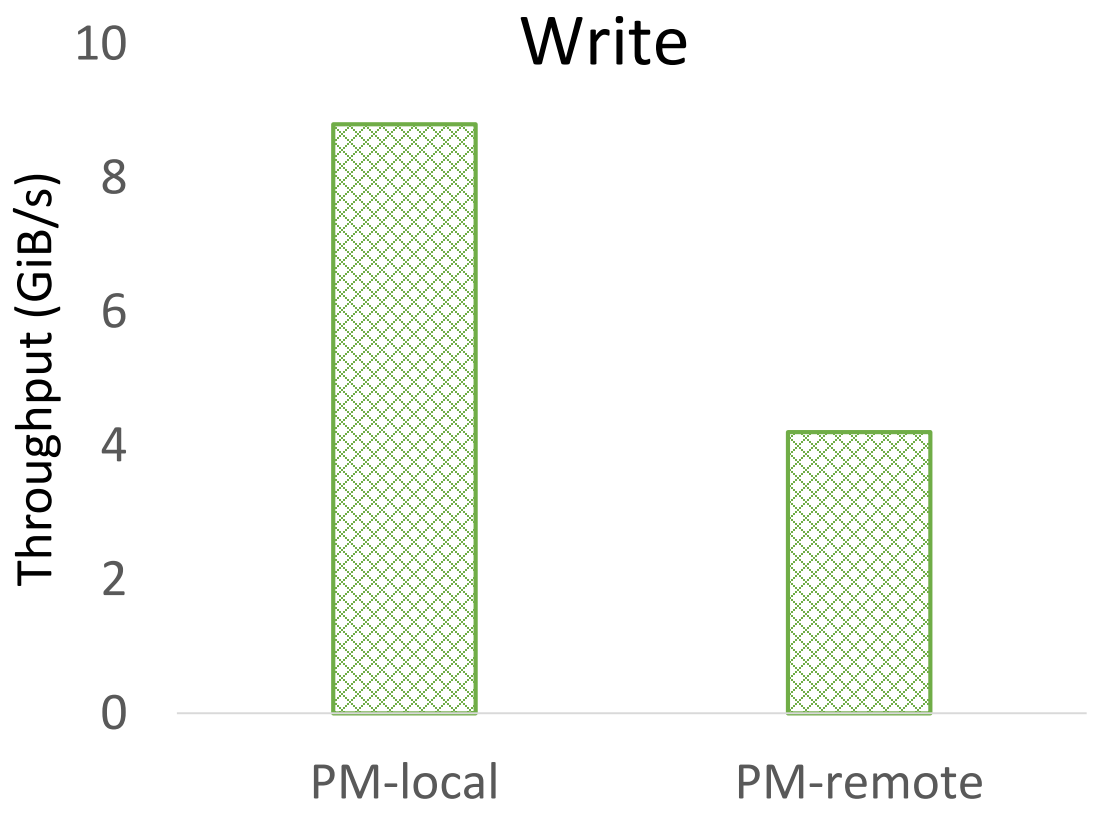
NUMA placement setup

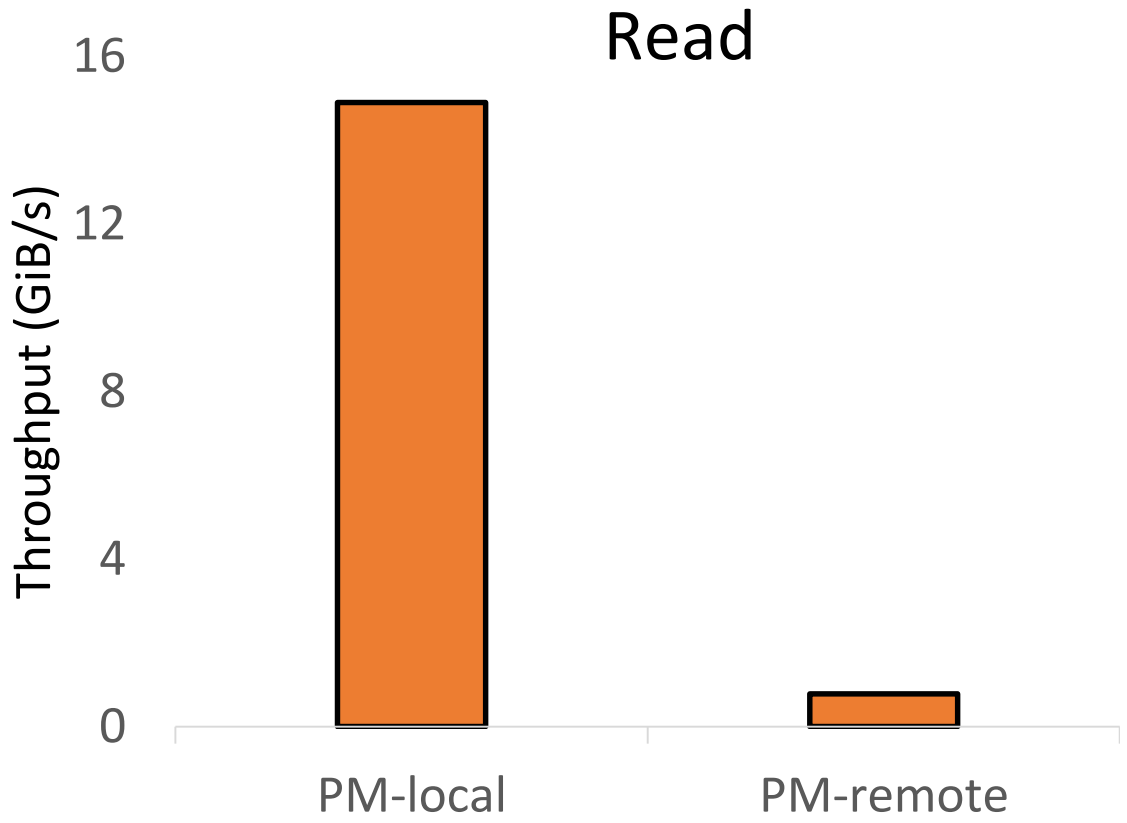


Same Task: copying data between PM in NUMA 0 and DRAM in NUMA 1

Local PM access outperforms remote PM access

Workload: FIO: each thread writes/reads 2MB data in a private file

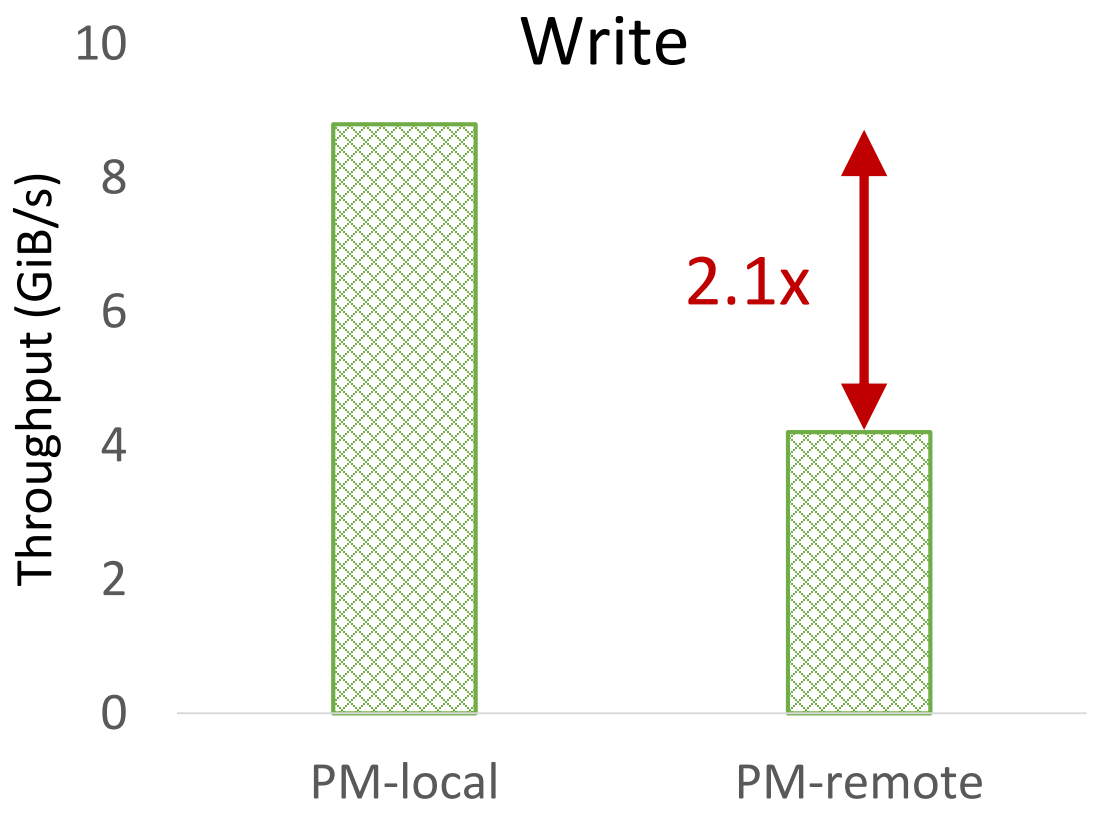


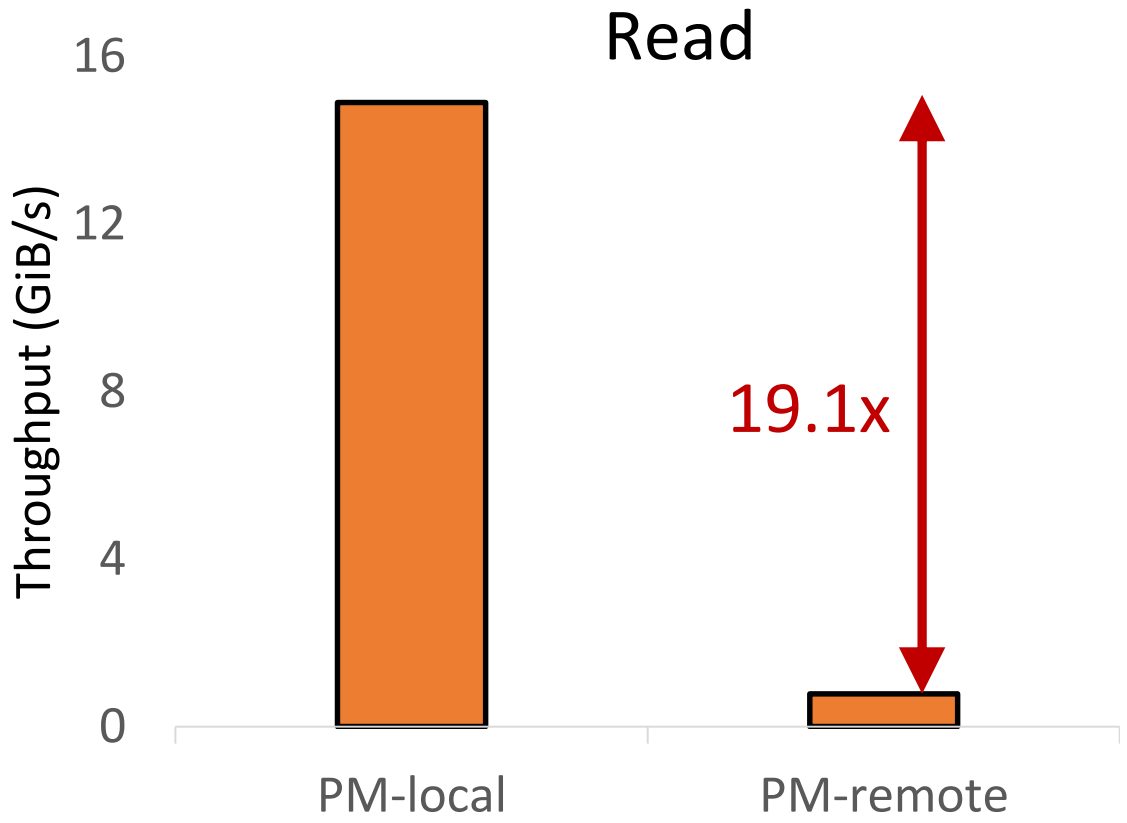


Same Task: copying data between PM in NUMA 0 and DRAM in NUMA 1

Local PM access outperforms remote PM access

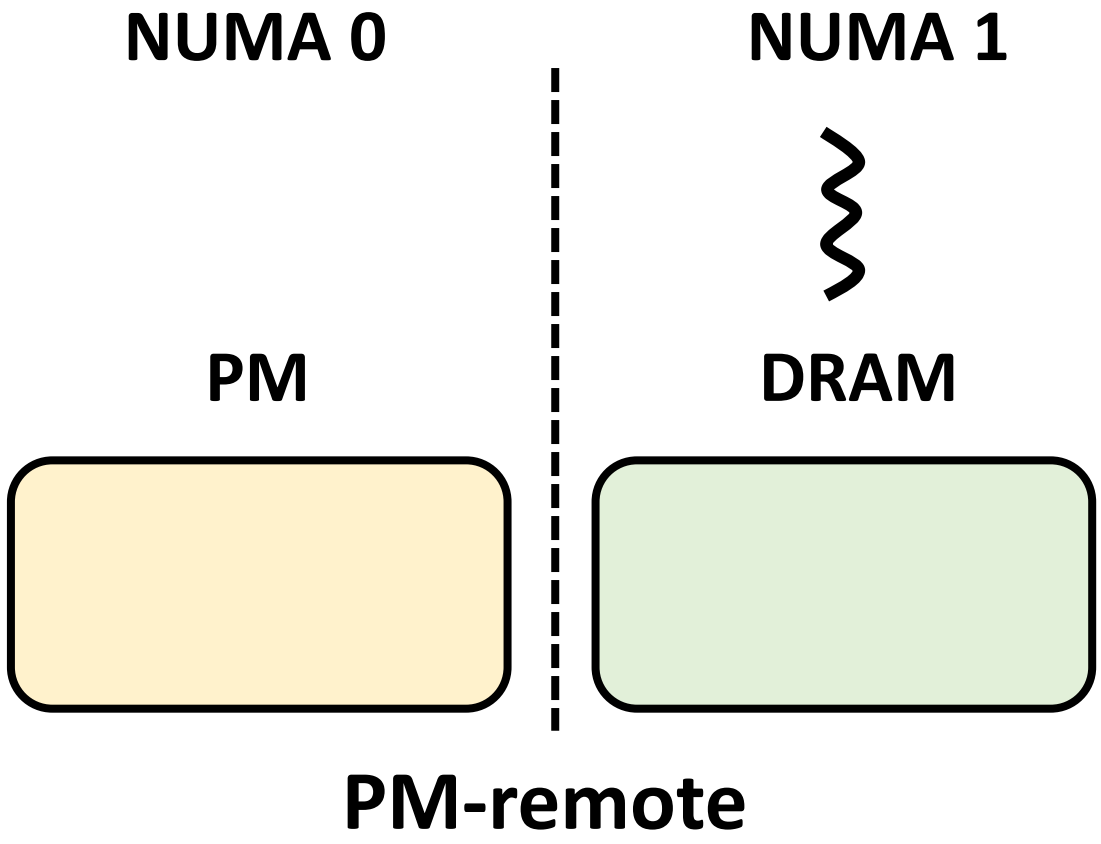
Workload: FIO: each thread writes/reads 2MB data in a private file





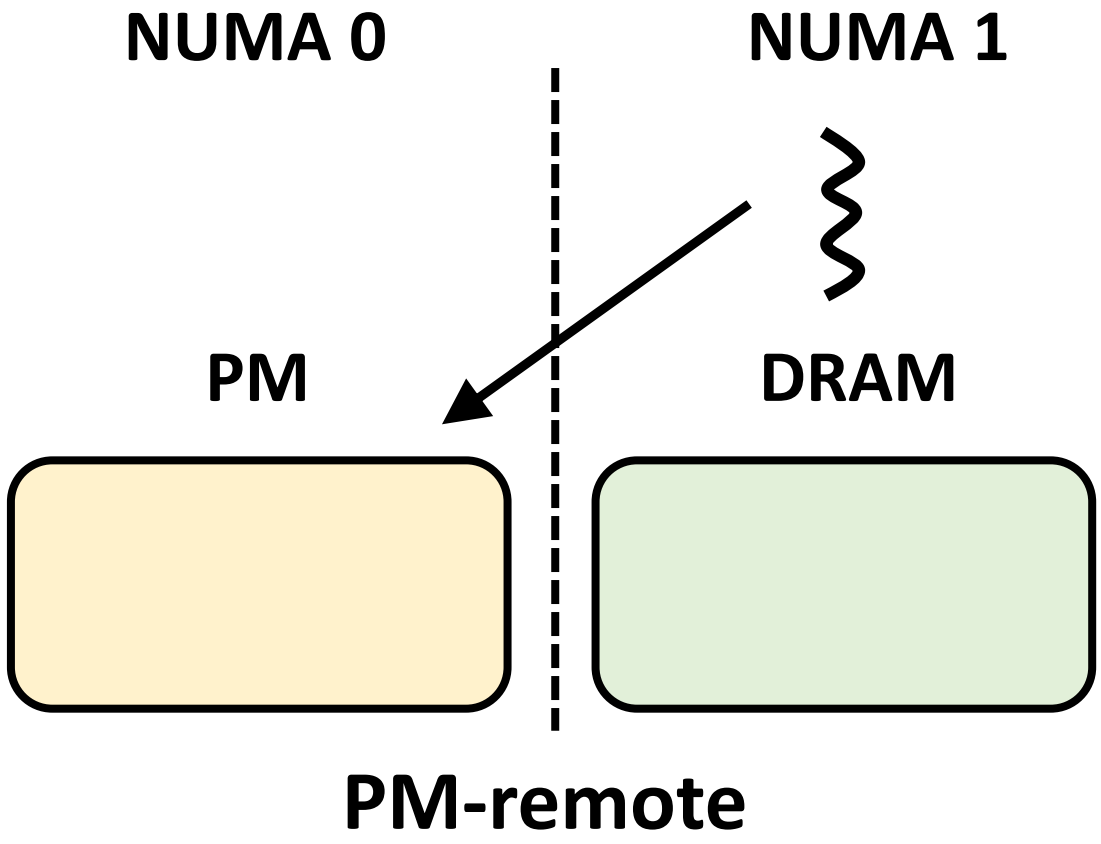
Same Task: copying data between PM in NUMA 0 and DRAM in NUMA 1

Directory coherence → slow remote PM access



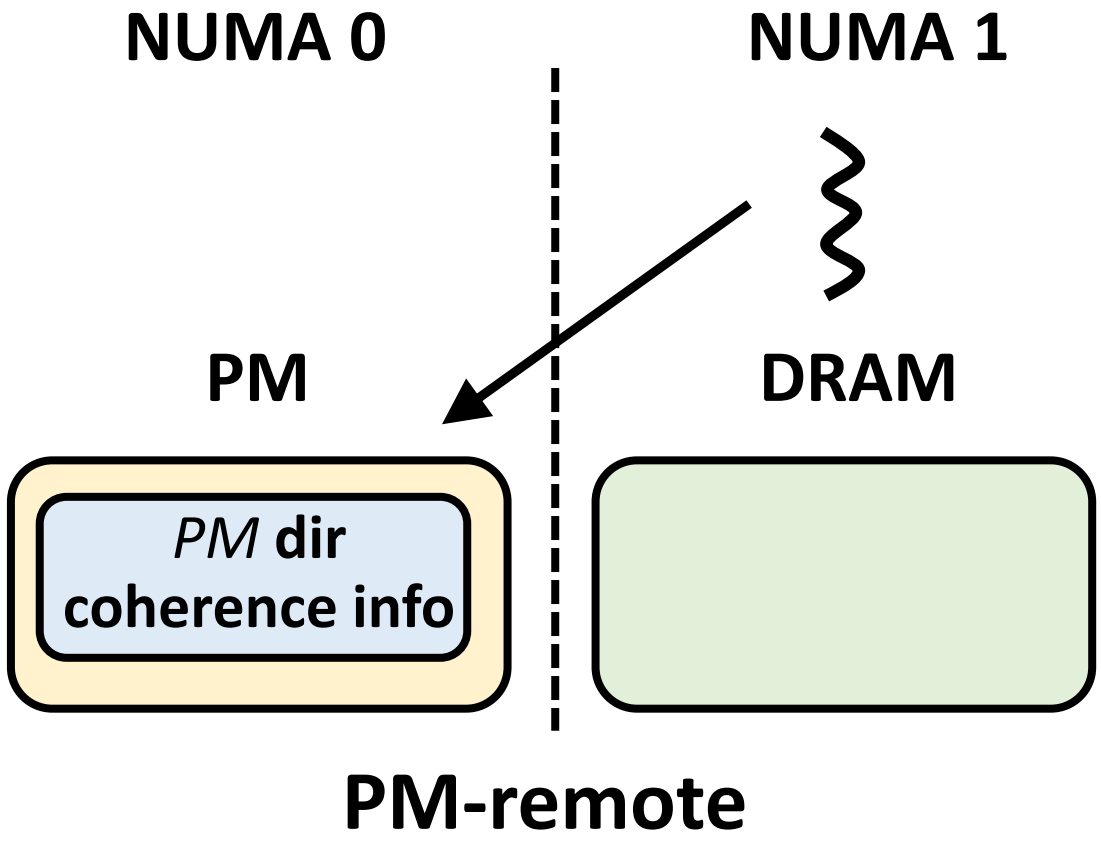
Directory coherence info maintained in memory

Directory coherence → slow remote PM access



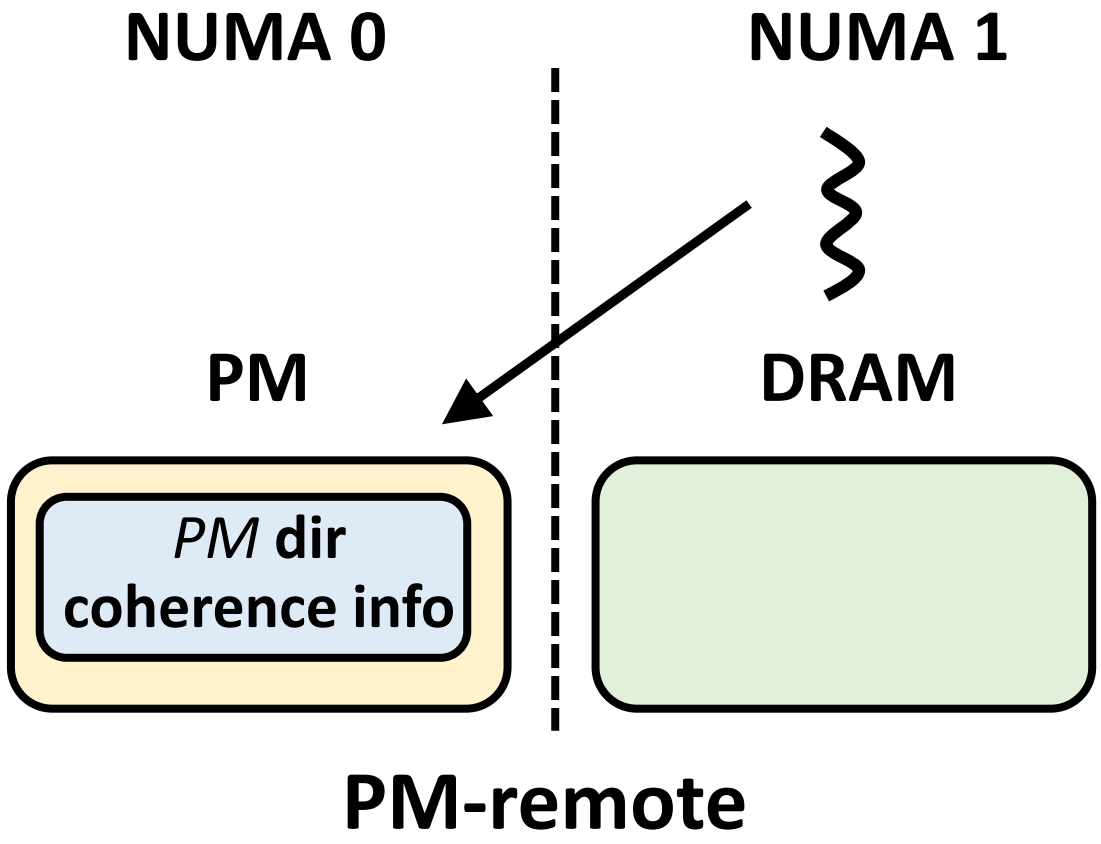
Directory coherence info maintained in memory

Directory coherence → slow remote PM access



Directory coherence info maintained in memory

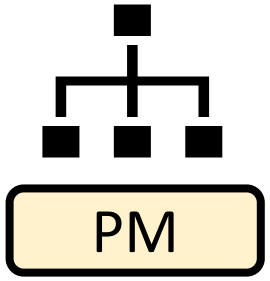
Directory coherence → slow remote PM access



Directory coherence info maintained in memory

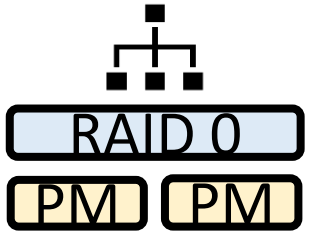
Slower PM performance
→ Slower remote PM access

PM performance analysis



Issue: Excessive concurrent access → PM performance collapse

Insight: Control concurrent PM access for maximal performance



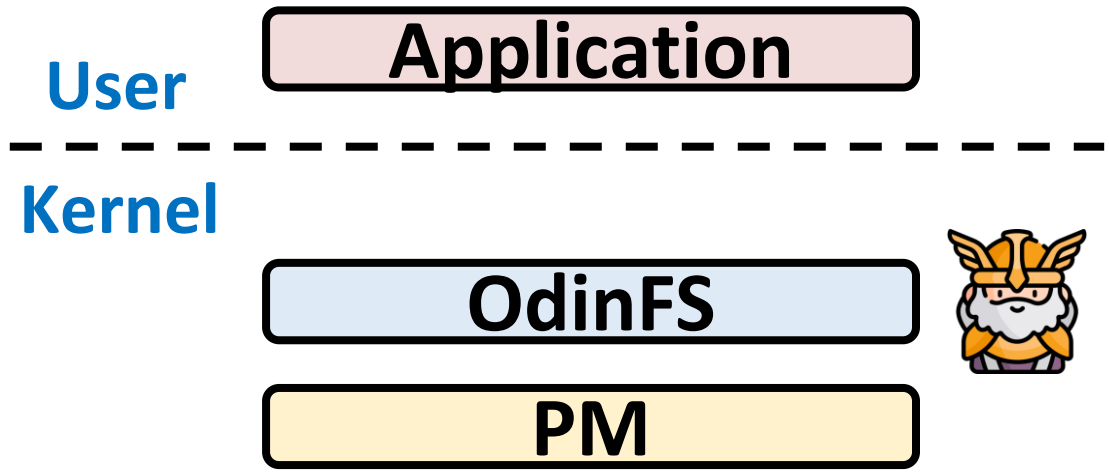
Issue: Inefficient remote PM access

Insight: Perform localized PM access to avoid PM NUMA impact

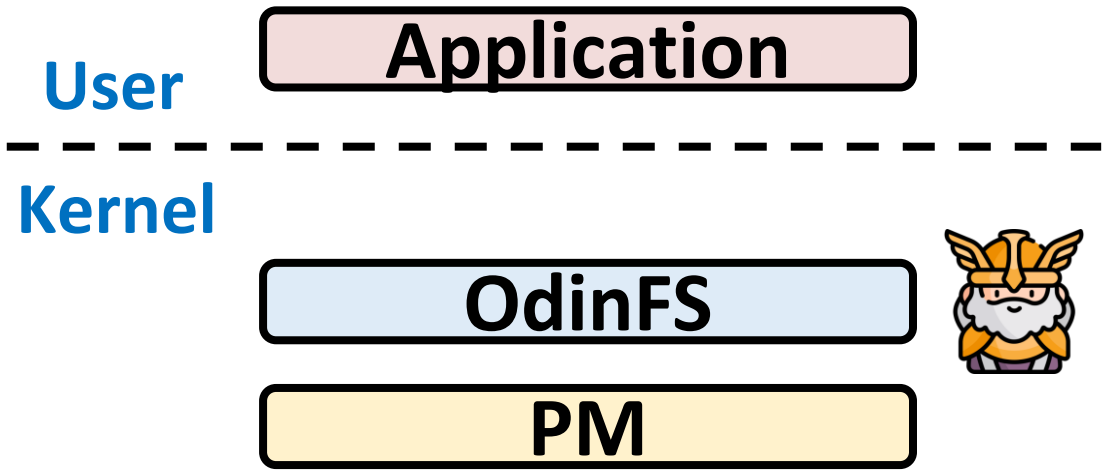
OdinFS

File system maximizes PM performance via opportunistic delegation

What is OdinFS?

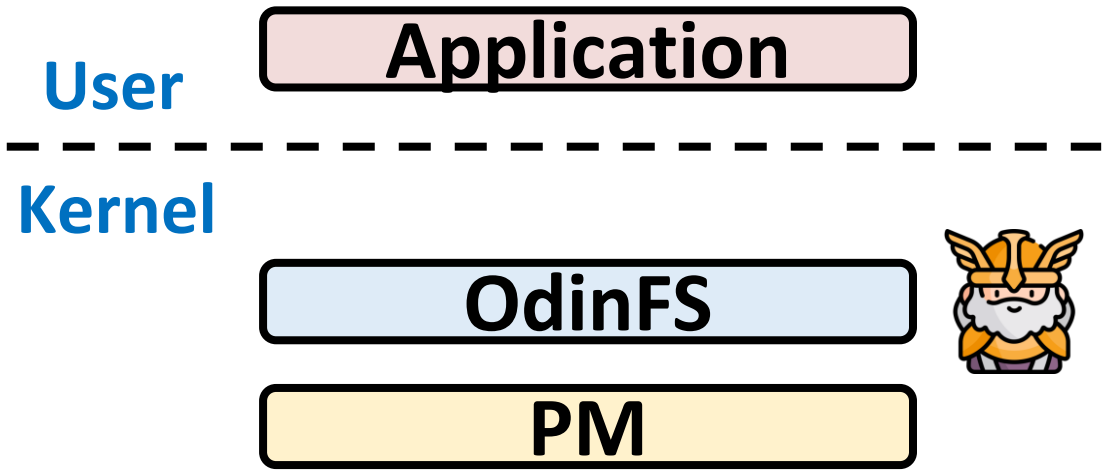


What is OdinFS?



In-kernel file system

What is OdinFS?



In-kernel file system

Fully POSIX compliant

OdinFS design overview

Limit concurrent access

➞ Preserves maximal PM performance within a NUMA node

Always localized PM access

➞ Minimizes PM NUMA impact and efficient use of remote PM

OdinFS design overview

Limit concurrent access

→ Preserves maximal PM performance within a NUMA node

Always localized PM access

→ Minimizes PM NUMA impact and efficient use of remote PM

Efficient use of the aggregated PM bandwidth

→ Applications on a single NUMA node can benefit

OdinFS design overview

Limit concurrent access

→ Preserves maximal PM performance within a NUMA node

Always localized PM access

→ Minimizes PM NUMA impact and efficient use of remote PM

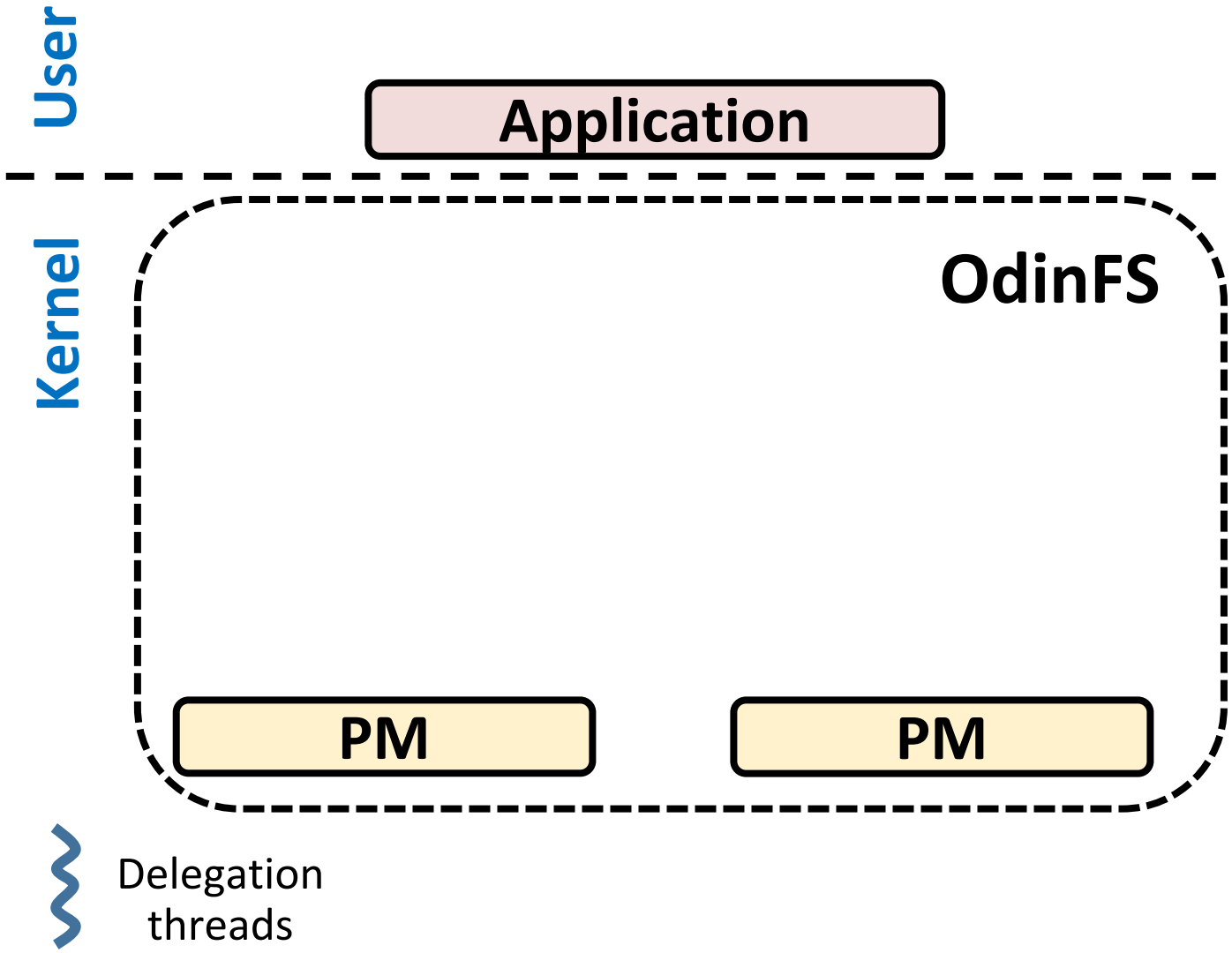
Efficient use of the aggregated PM bandwidth

→ Applications on a single NUMA node can benefit

Key insight: **Decouple** PM access from application threads to achieve the above goals **simultaneously**

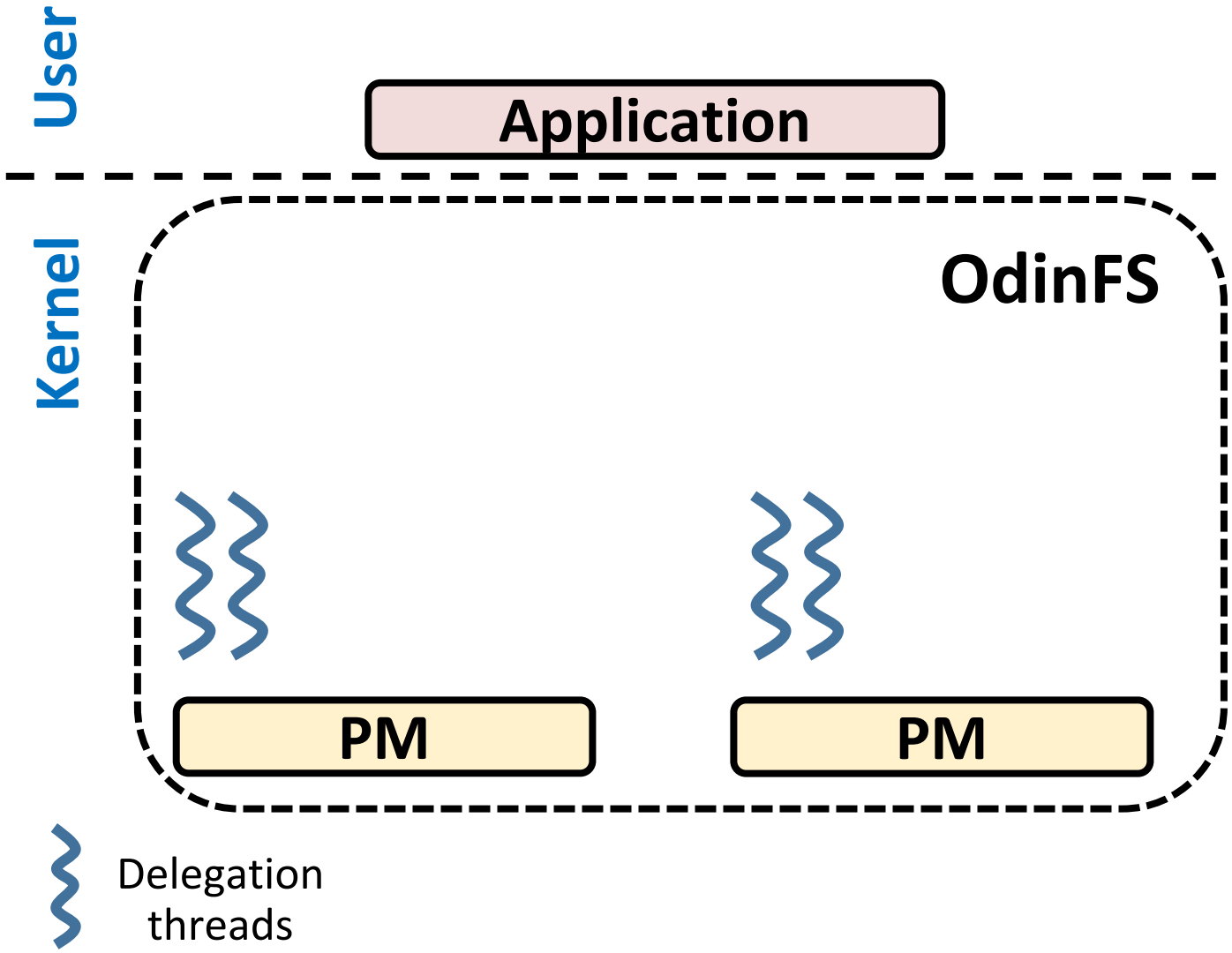
Decouple PM access from application threads

Decouple PM access from application threads



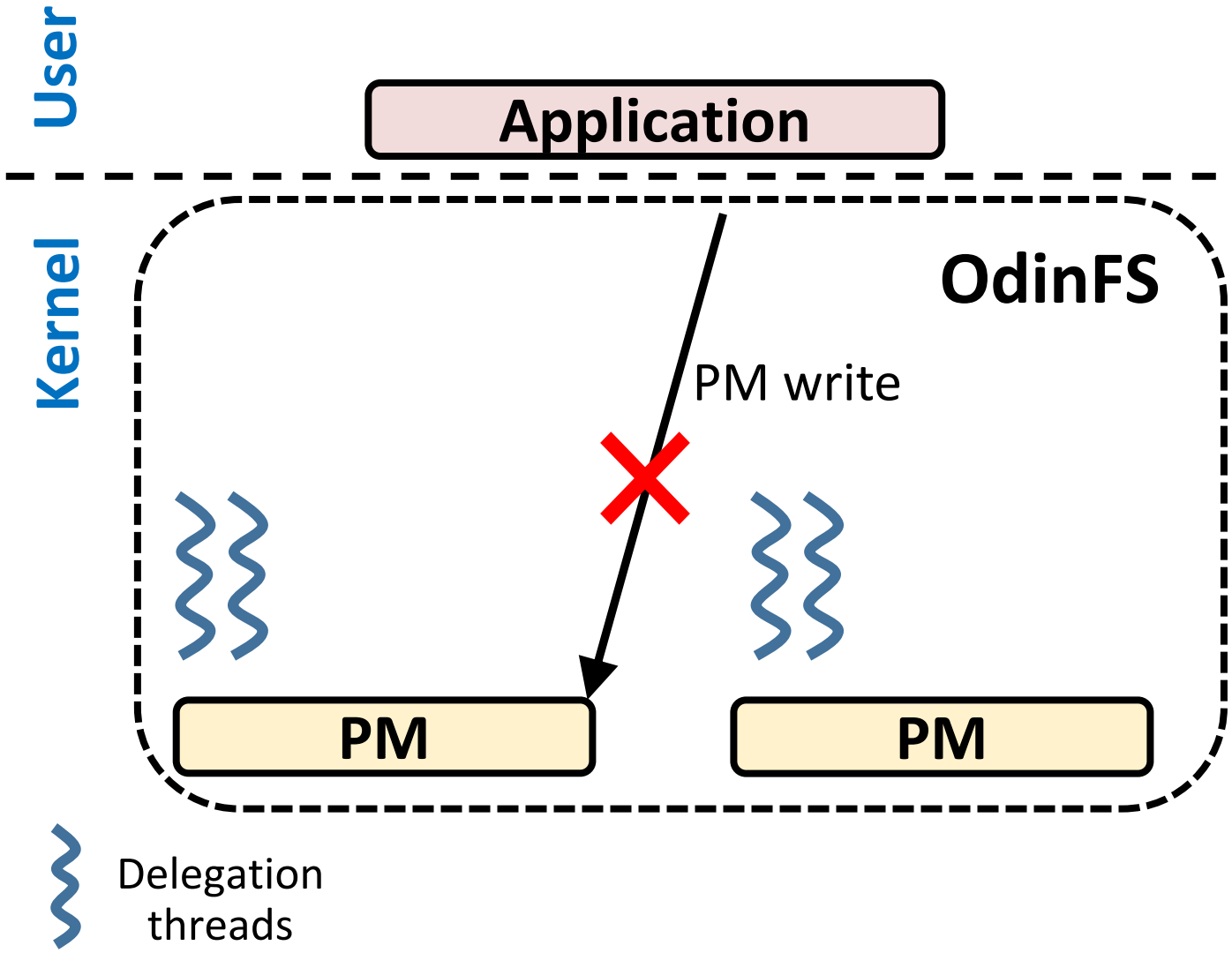
Decouple PM access from application threads

Decouple PM access from application threads



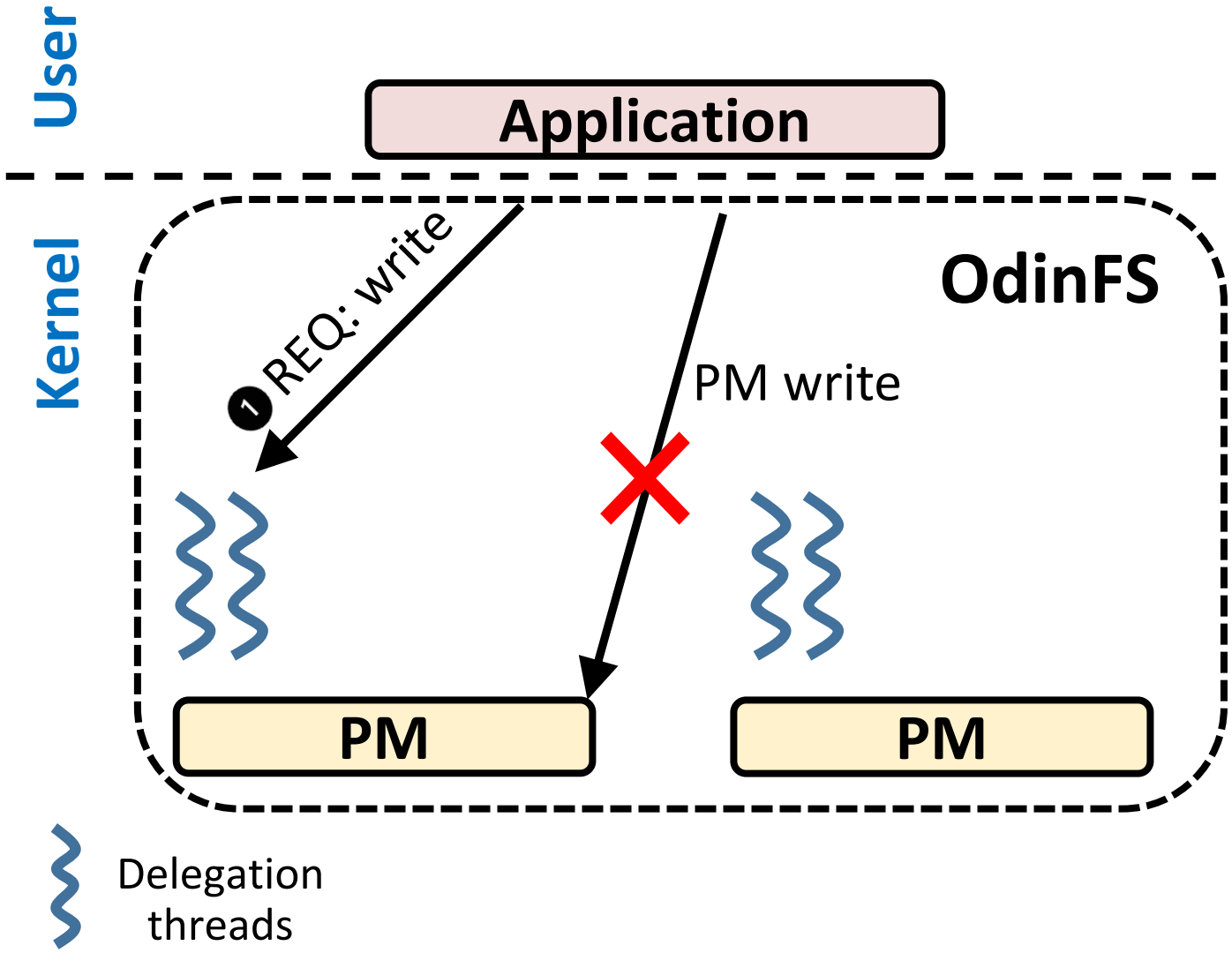
Decouple PM access from application threads

Decouple PM access from application threads



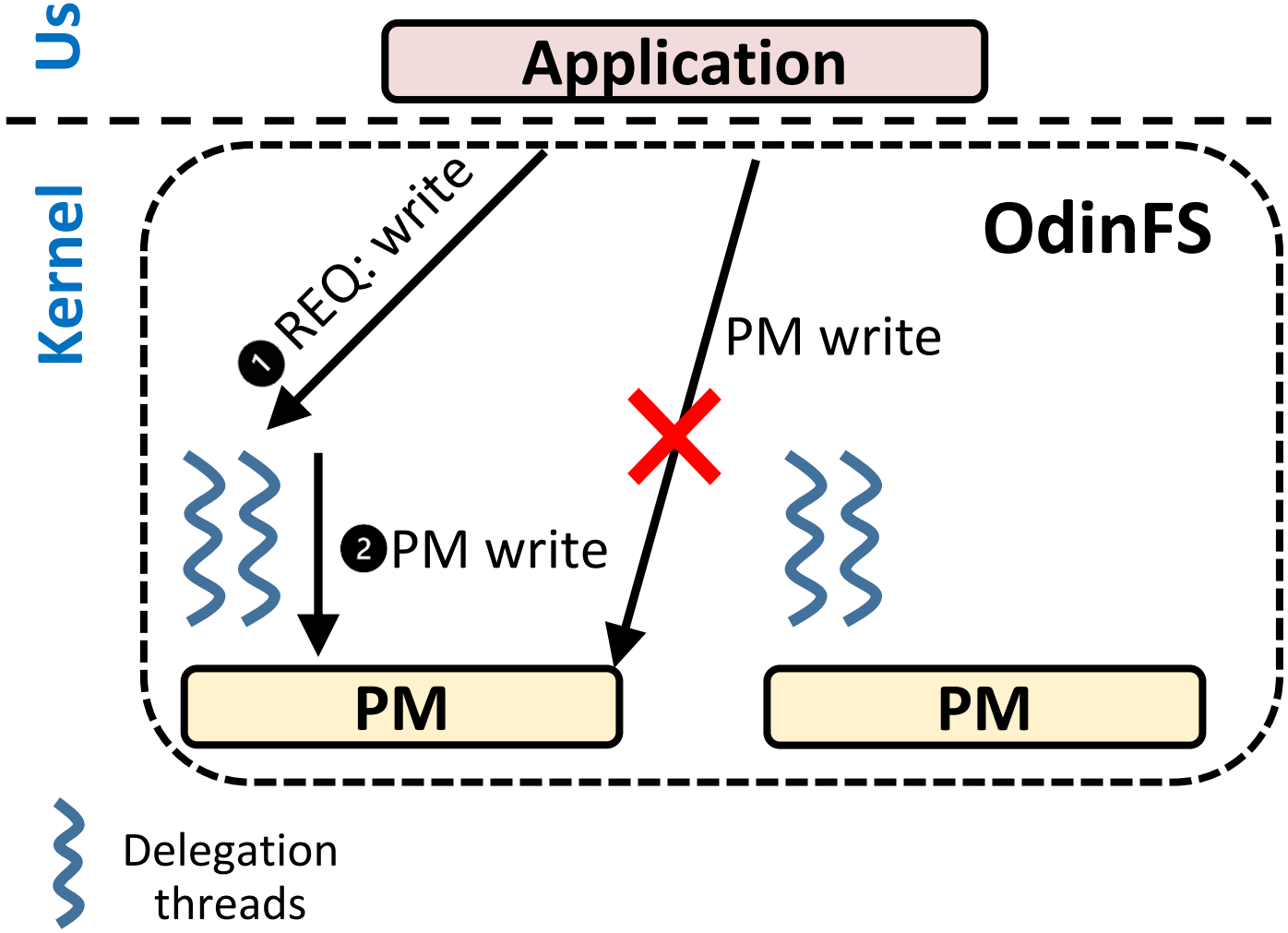
Decouple PM access from application threads

Decouple PM access from application threads



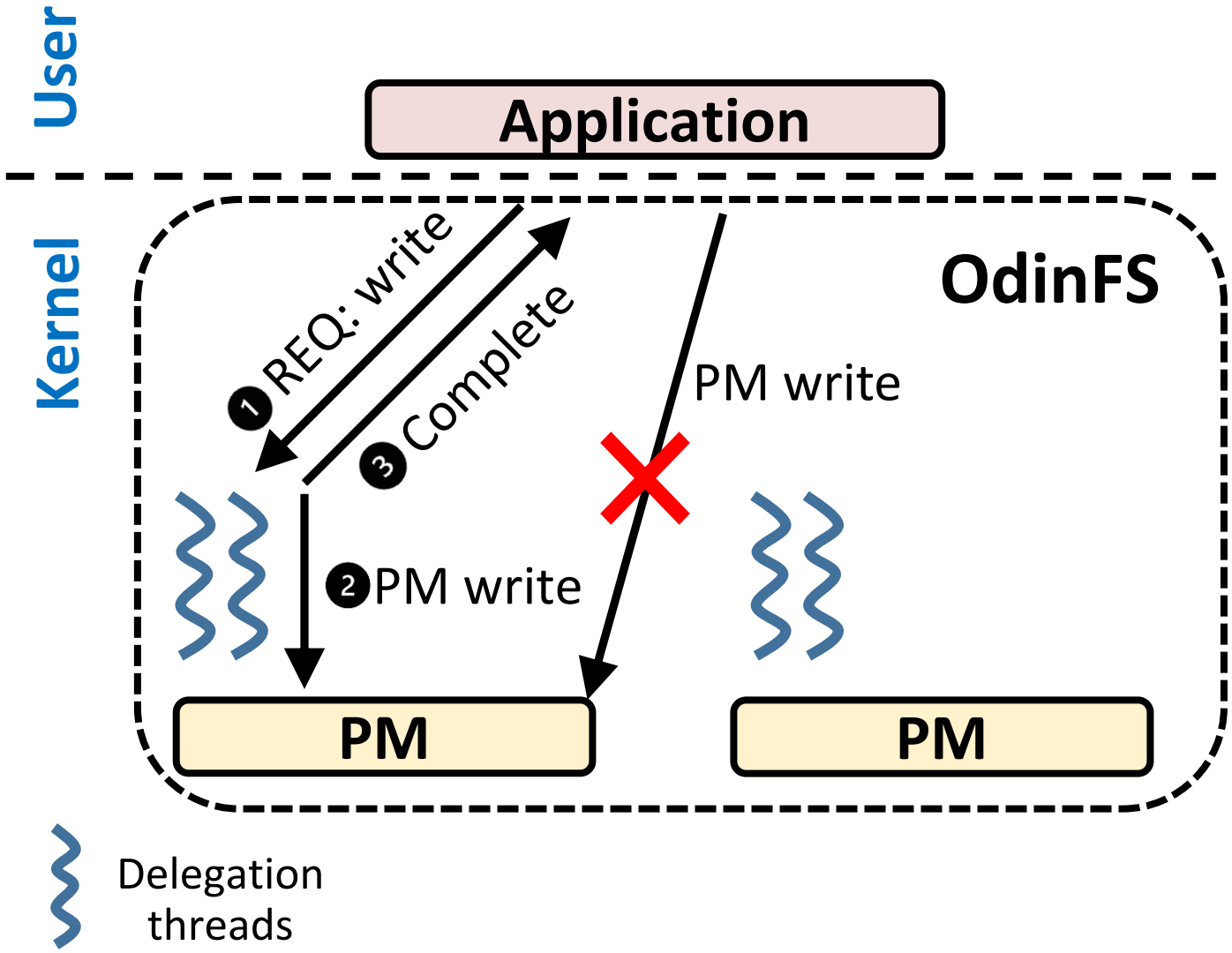
Decouple PM access from application threads

Decouple PM access from application threads

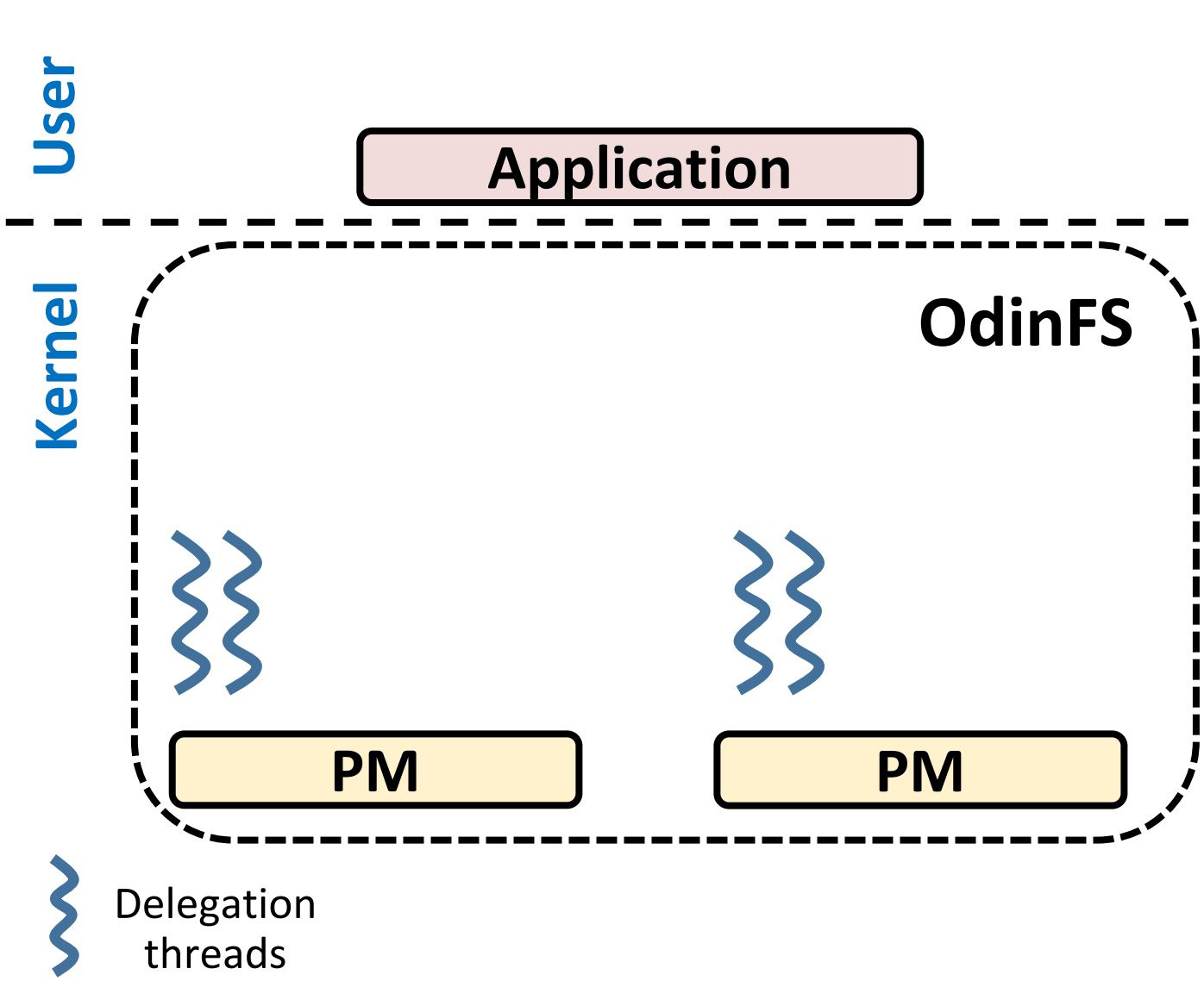


Decouple PM access from application threads

Decouple PM access from application threads



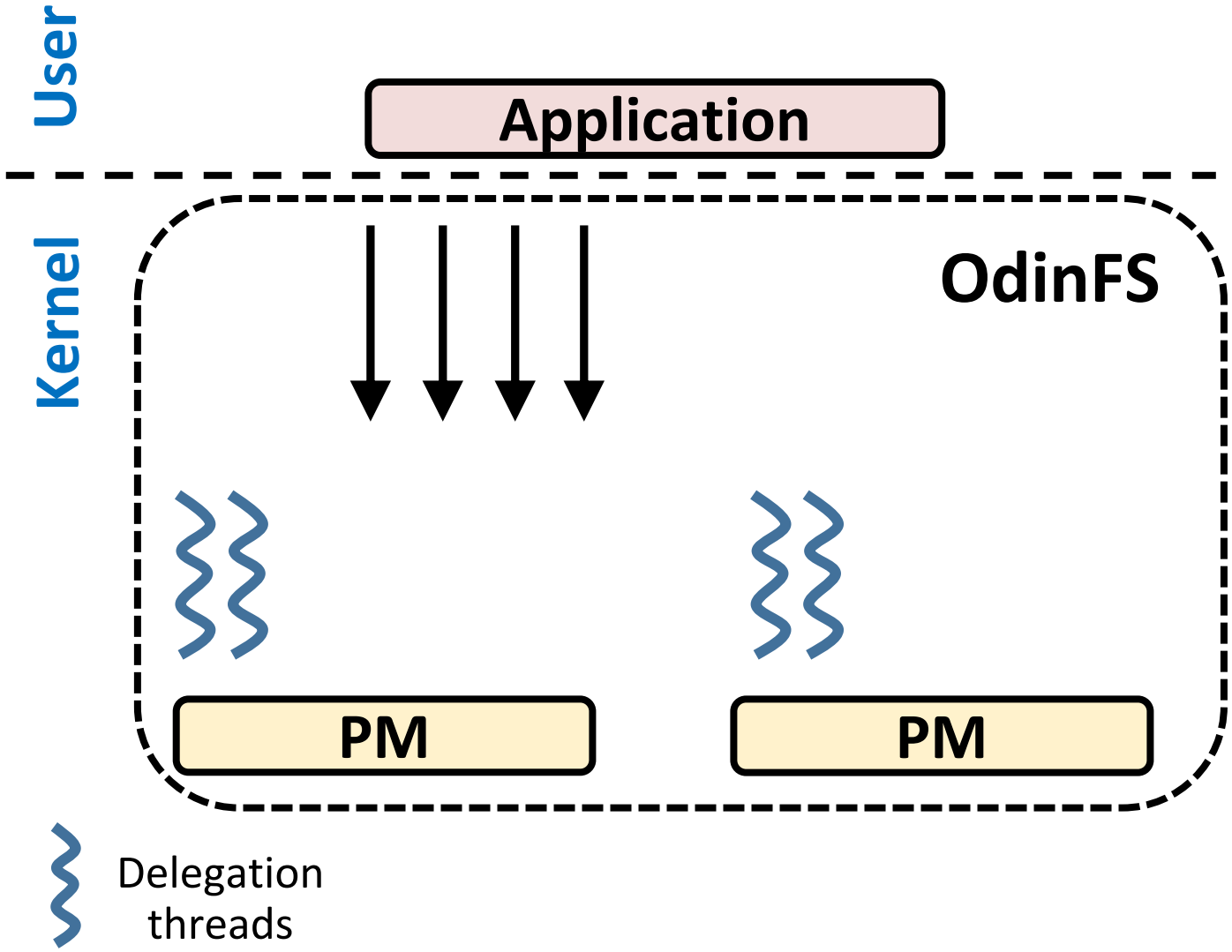
Delegation enables controlled and localized access



Decouple PM access from application threads

Fixed number of delegation threads **limit** concurrent access

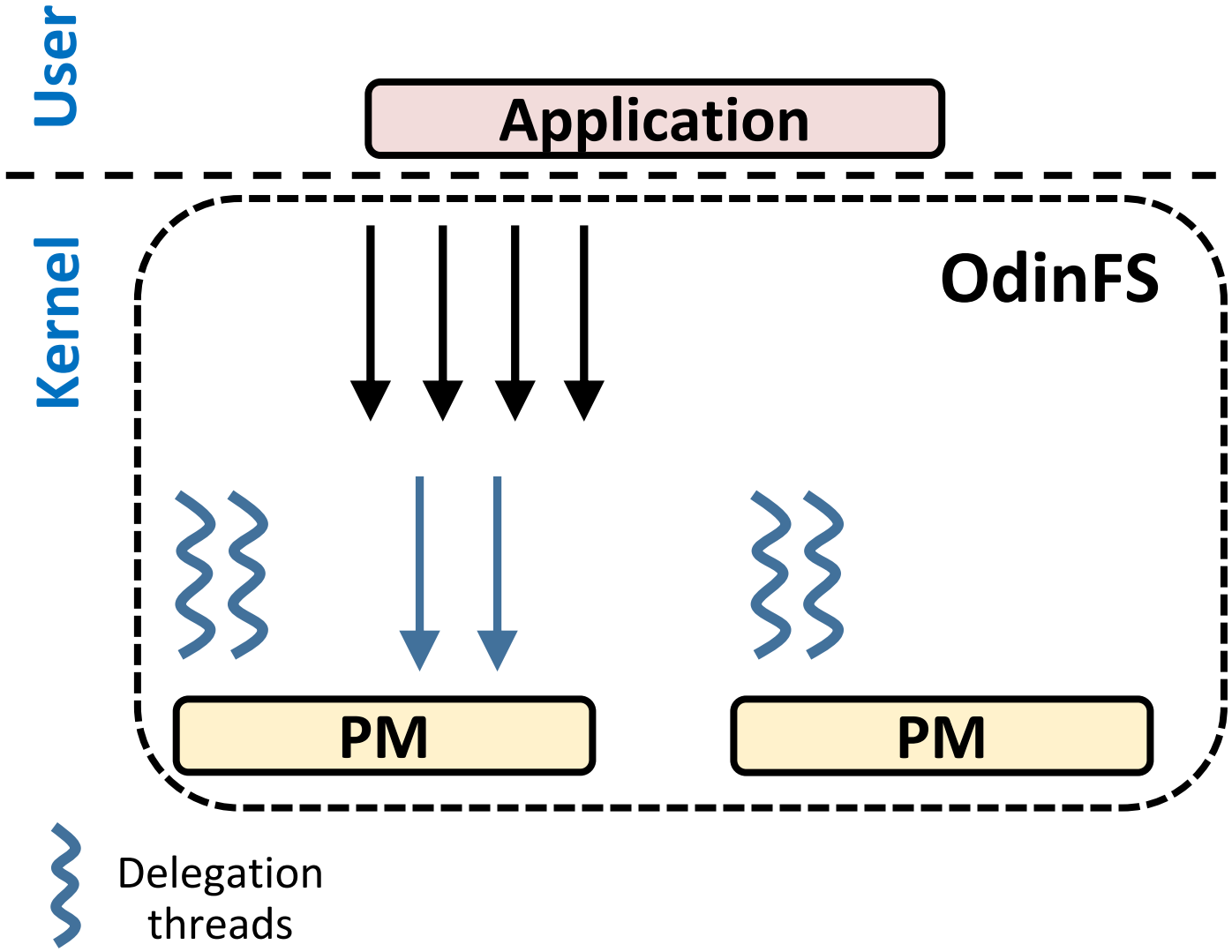
Delegation enables controlled and localized access



Decouple PM access from application threads

Fixed number of delegation threads **limit** concurrent access

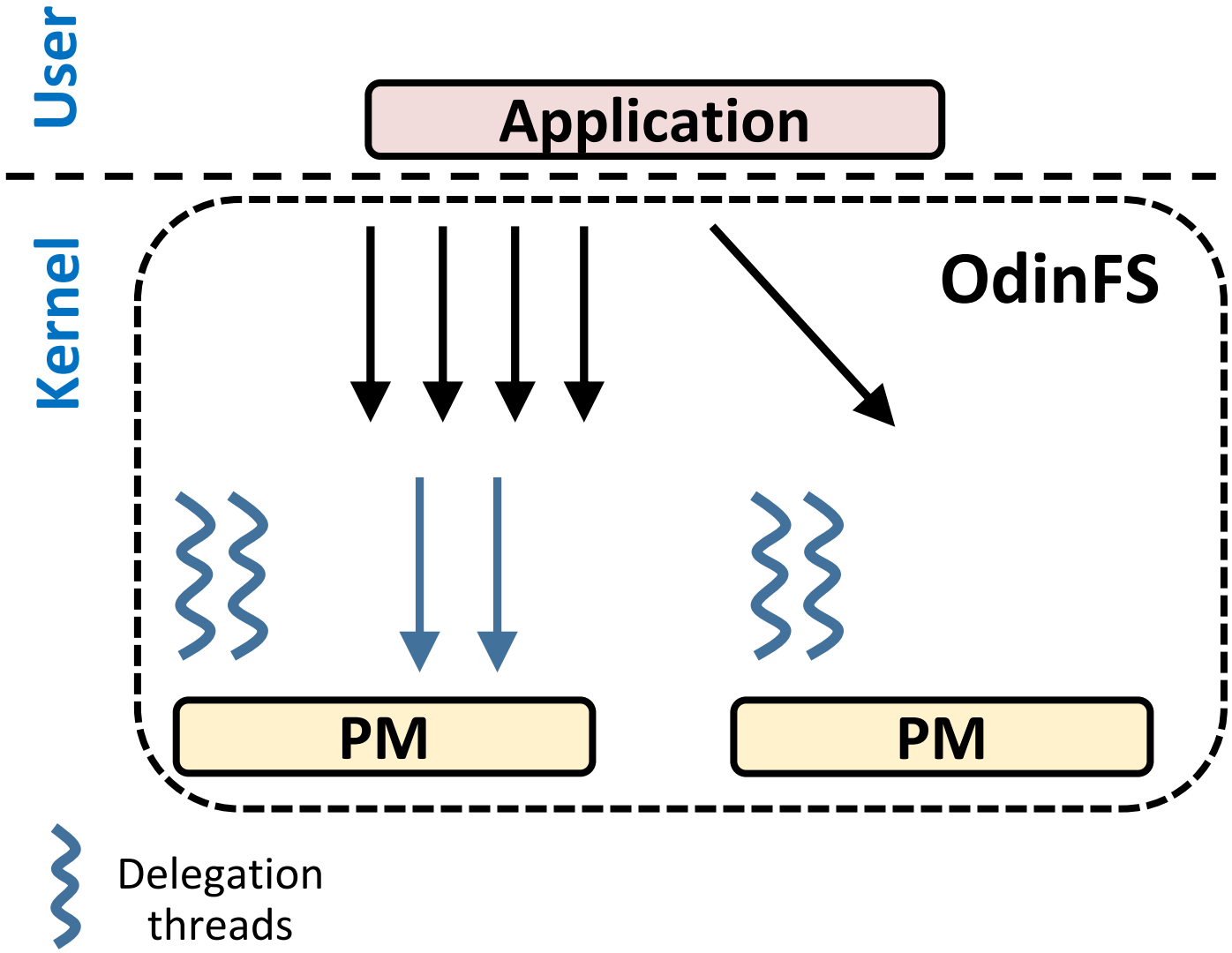
Delegation enables controlled and localized access



Decouple PM access from application threads

Fixed number of delegation threads **limit** concurrent access

Delegation enables controlled and localized access

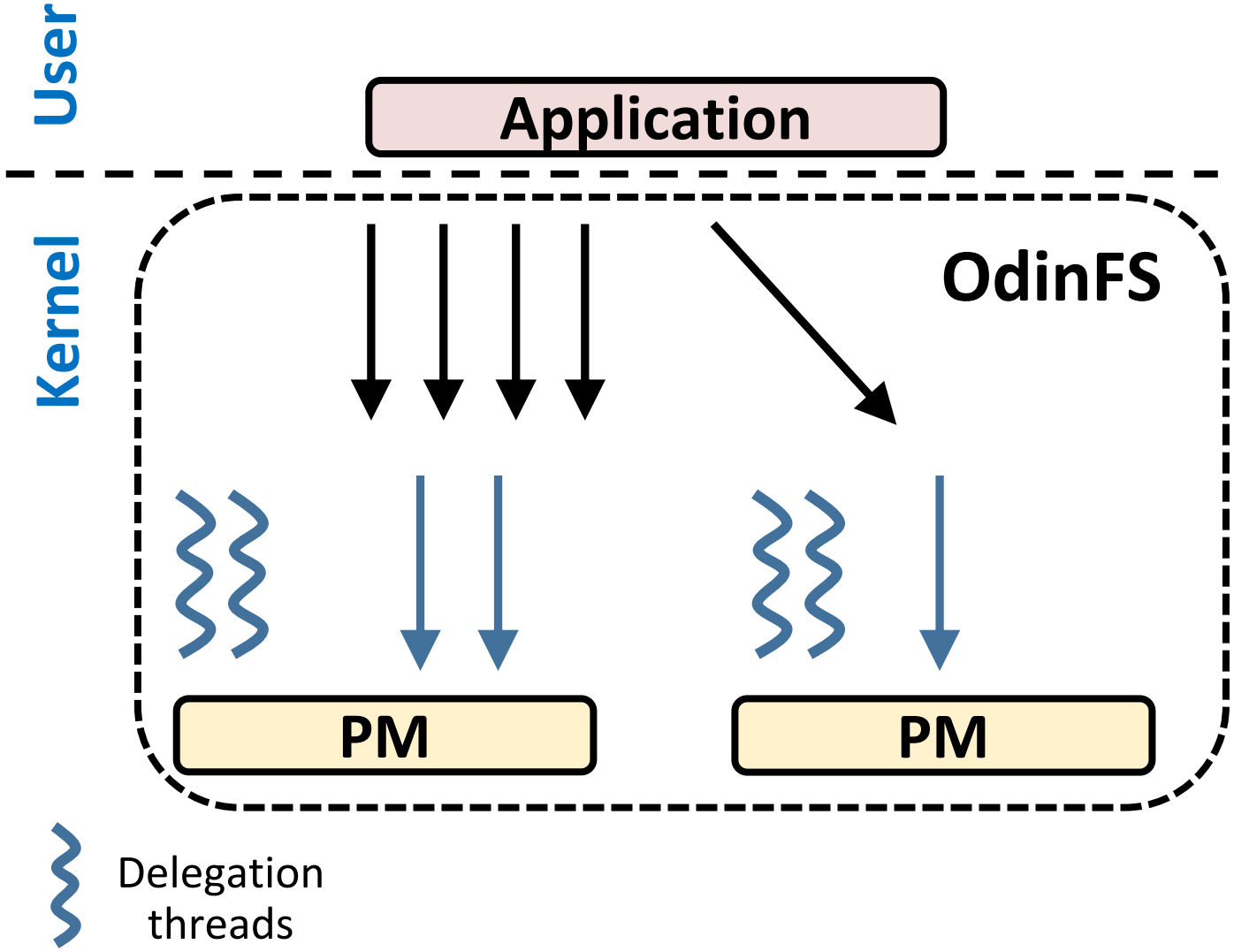


Decouple PM access from application threads

Fixed number of delegation threads **limit** concurrent access

Delegation threads always perform **localized** PM access

Delegation enables controlled and localized access

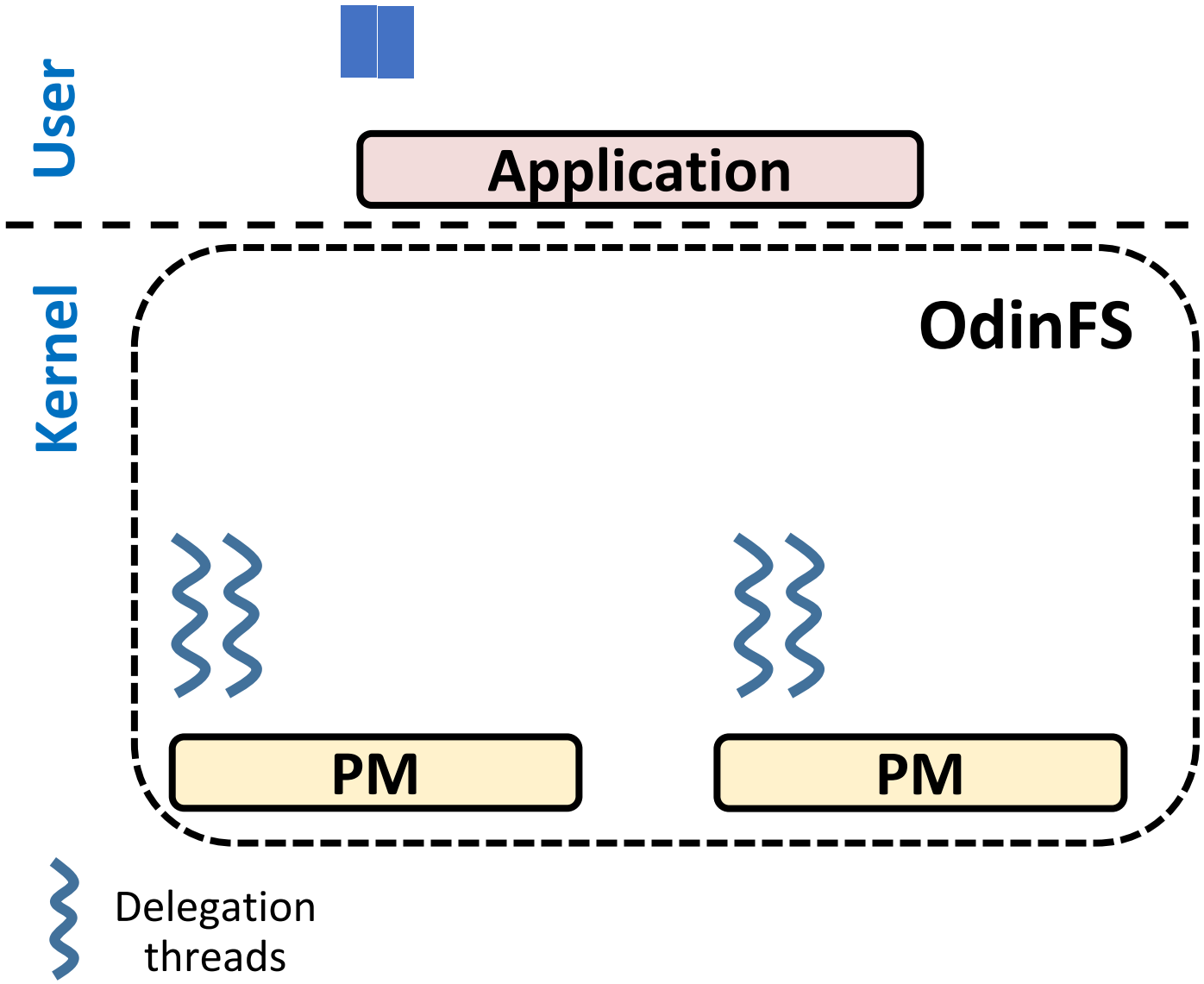


Decouple PM access from application threads

Fixed number of delegation threads **limit** concurrent access

Delegation threads always perform **localized** PM access

Efficiently utilize aggregated PM bandwidth



Decouple PM access from application threads

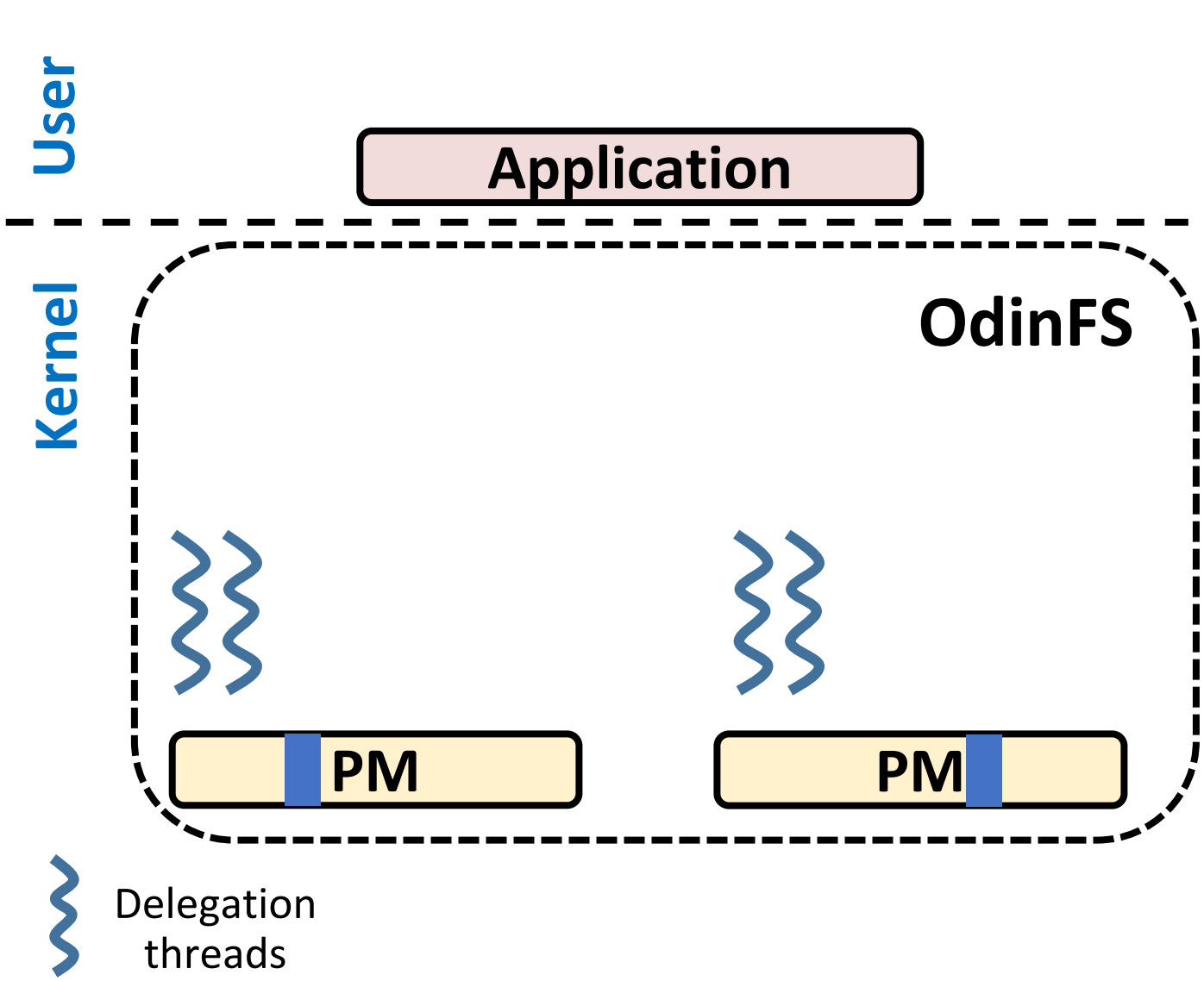
Fixed number of delegation threads **limit** concurrent access

Delegation threads always perform **localized** PM access

Utilize **aggregated** bandwidth

– Stripe file across NUMA nodes

Efficiently utilize aggregated PM bandwidth



Decouple PM access from application threads

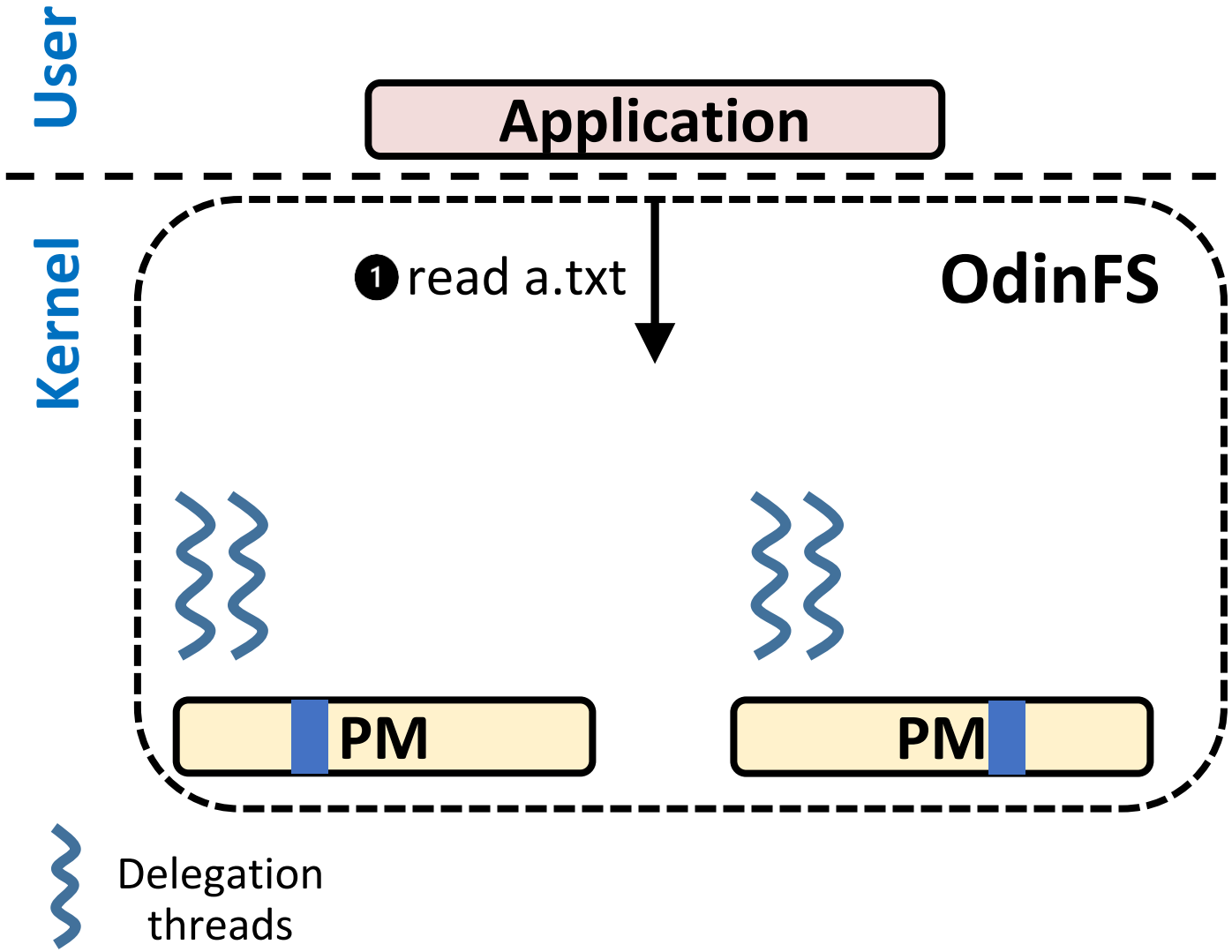
Fixed number of delegation threads **limit** concurrent access

Delegation threads always perform **localized** PM access

Utilize **aggregated** bandwidth

– Stripe file across NUMA nodes

Efficiently utilize aggregated PM bandwidth



Decouple PM access from application threads

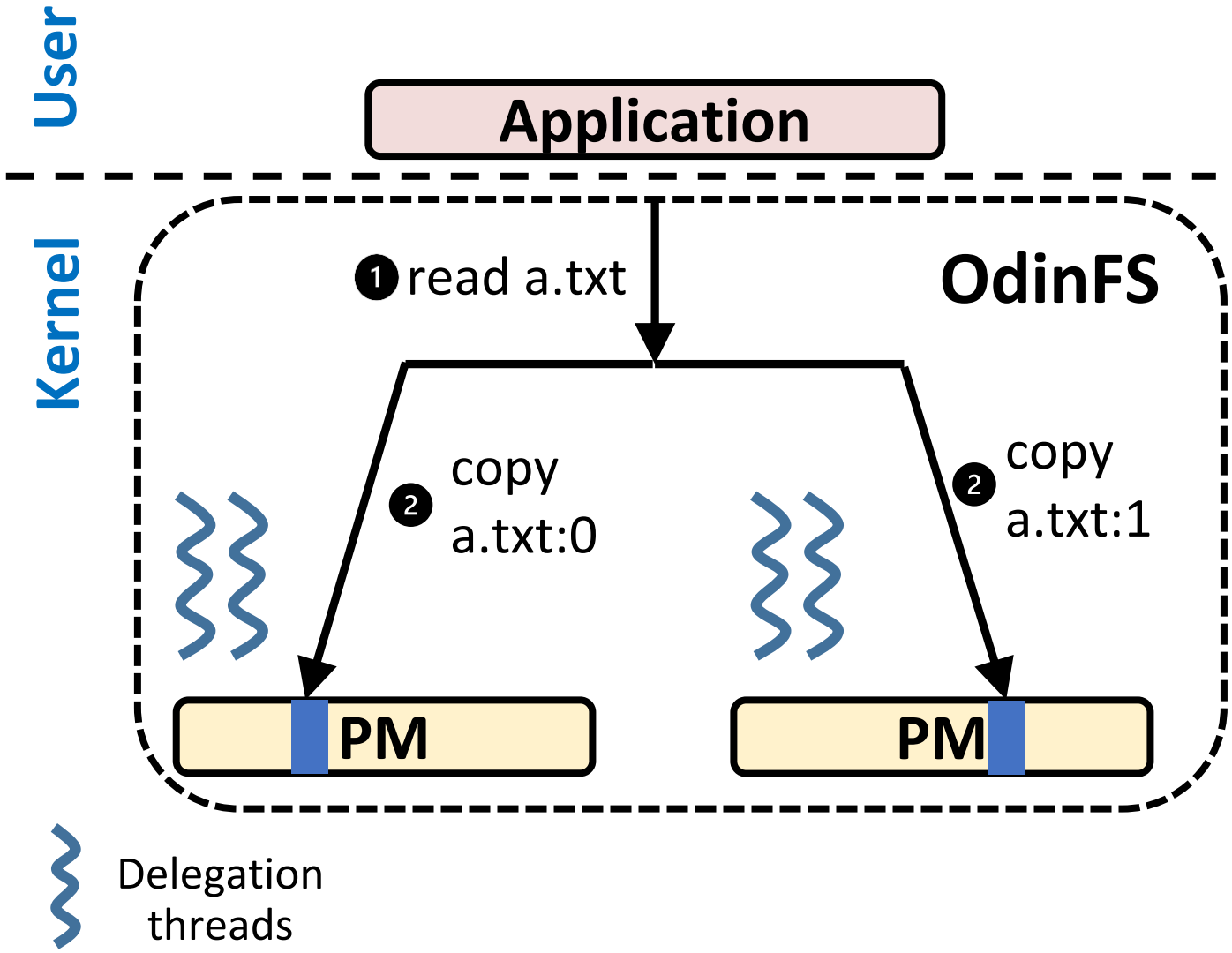
Fixed number of delegation threads **limit** concurrent access

Delegation threads always perform **localized** PM access

Utilize **aggregated** bandwidth

- Stripe file across NUMA nodes
- Transparent parallel PM access across NUMA nodes

Efficiently utilize aggregated PM bandwidth



Decouple PM access from application threads

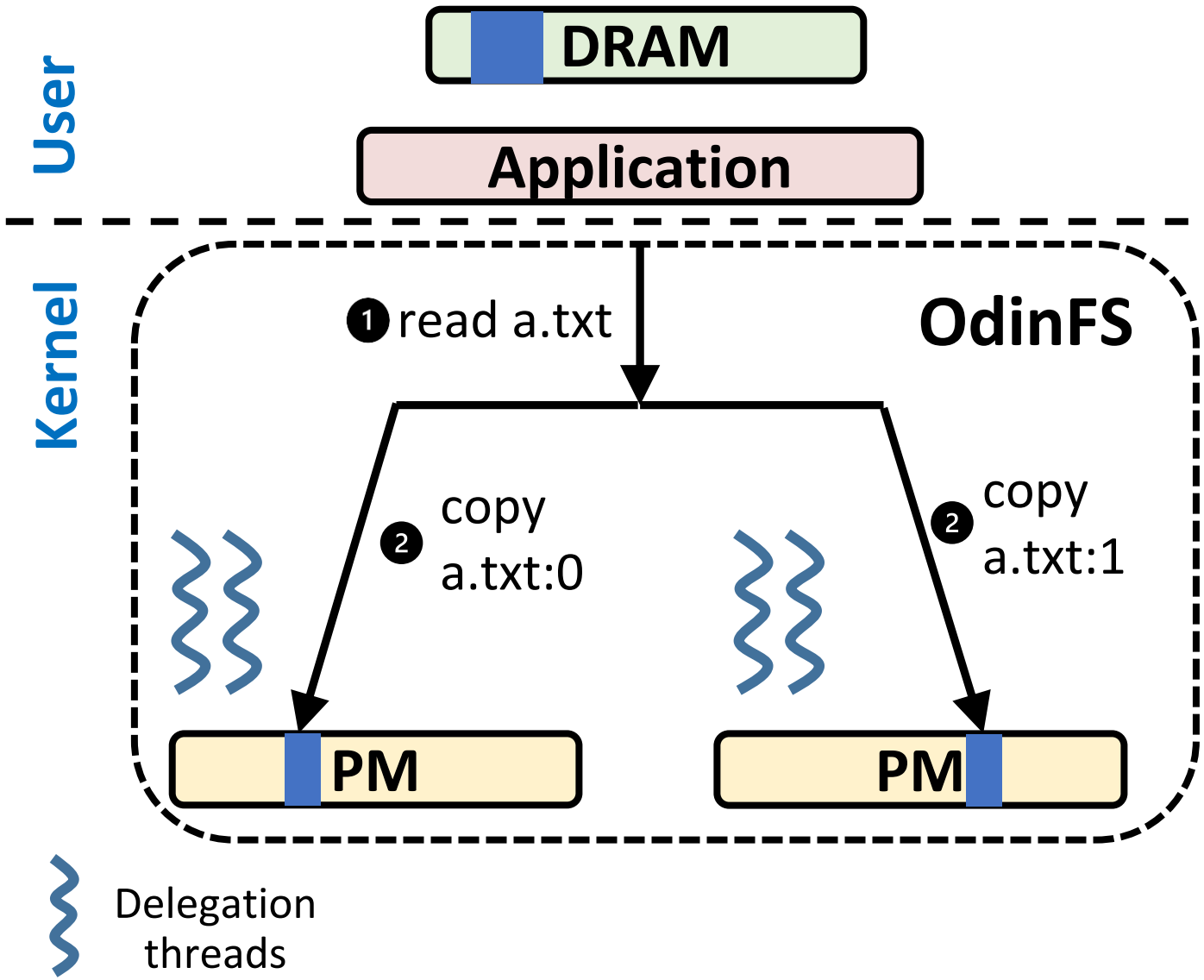
Fixed number of delegation threads **limit** concurrent access

Delegation threads always perform **localized** PM access

Utilize **aggregated** bandwidth

- Stripe file across NUMA nodes
- Transparent parallel PM access across NUMA nodes

Efficiently utilize aggregated PM bandwidth



Decouple PM access from application threads

Fixed number of delegation threads **limit** concurrent access

Delegation threads always perform **localized** PM access

Utilize **aggregated** bandwidth

- Stripe file across NUMA nodes
- Transparent parallel PM access across NUMA nodes

OdinFS: Other design aspects

Highly scalable PM file system

- **Maximize concurrent accesses** with range locks
- **Minimize synchronization overhead** with scalable data structures
- **Ensure crash consistency** despite concurrent access

OdinFS: Other design aspects

Highly scalable PM file system

- **Maximize concurrent accesses** with range locks
- **Minimize synchronization overhead** with scalable data structures
- **Ensure crash consistency** despite concurrent access

Minimize delegation overhead

- **Opportunistic delegation** e.g., do not delegate small PM access
- **Adaptive spinning and parking** → Avoid wasting CPU cycles

Performance Evaluation

Does OdinFS improve I/O performance

- Setup: 224-core eight-socket machine

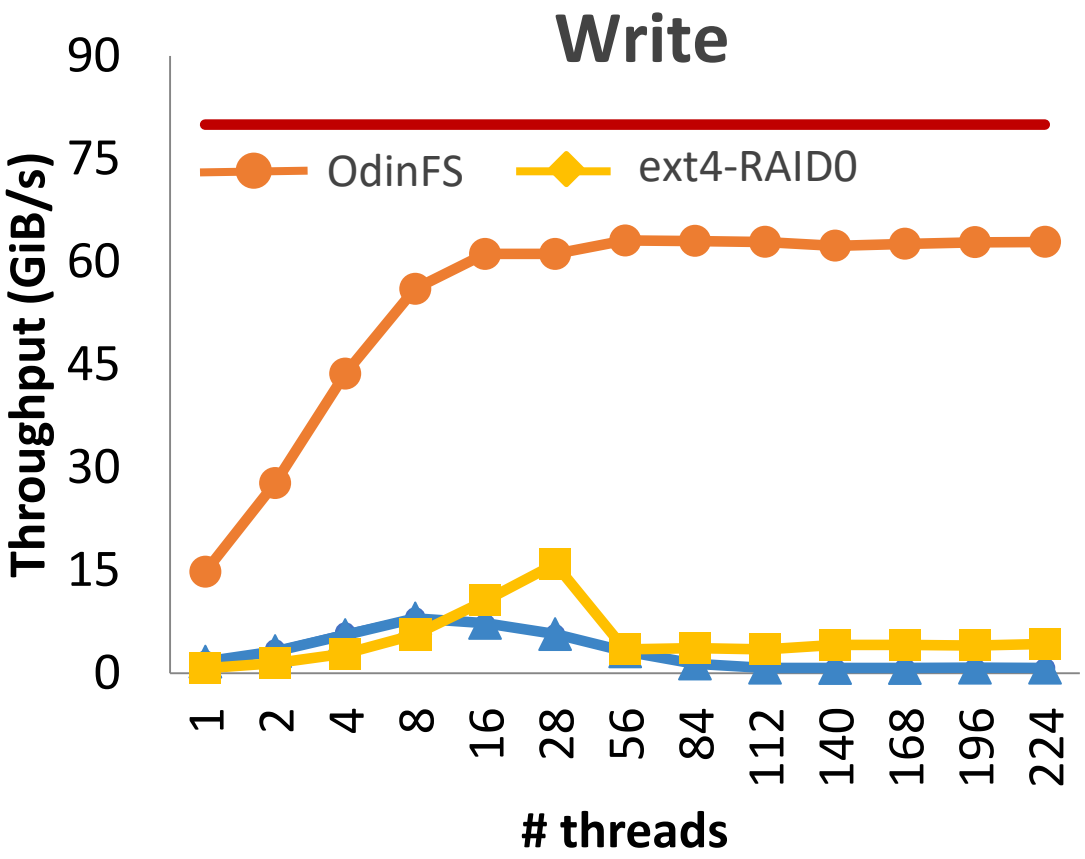
Performance Evaluation

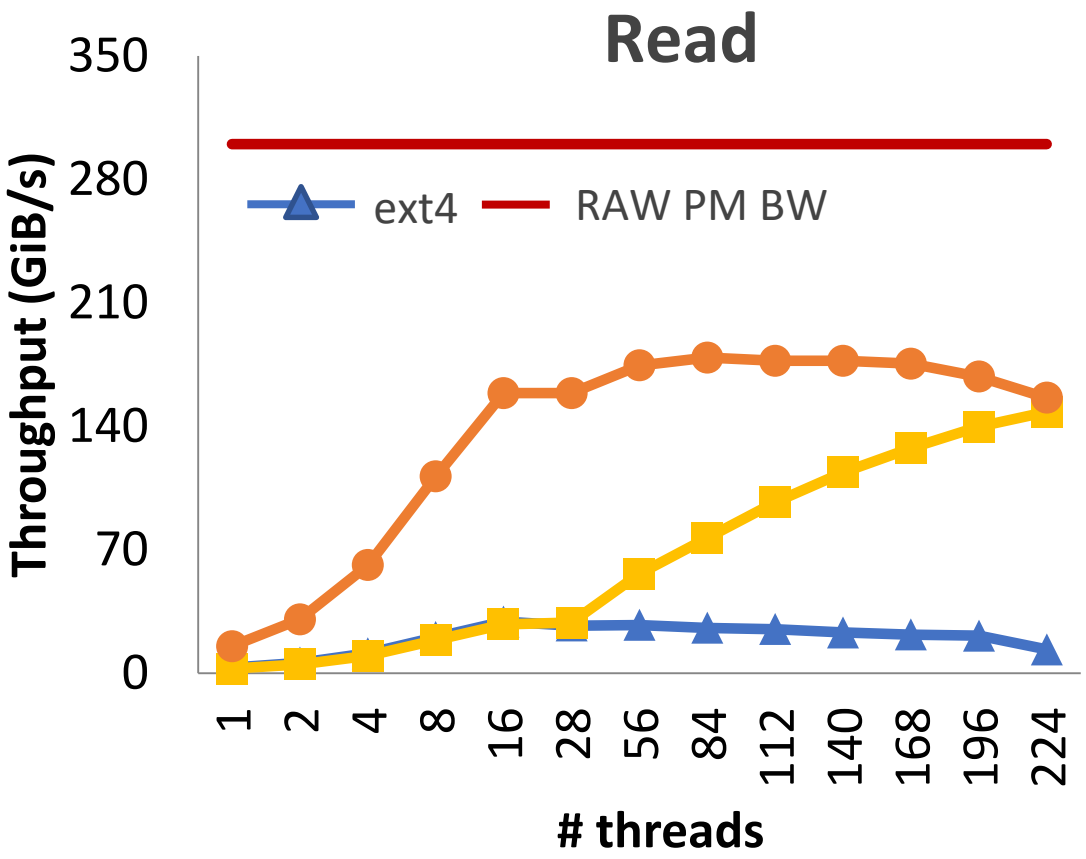
Does OdinFS improve I/O performance

- Setup: 224-core eight-socket machine
- Microbenchmark (FIO)
- Marcobenchmark (Filebench)

Microbenchmark: FIO

Workload: FIO: each thread writes/reads 2MB data in a private file

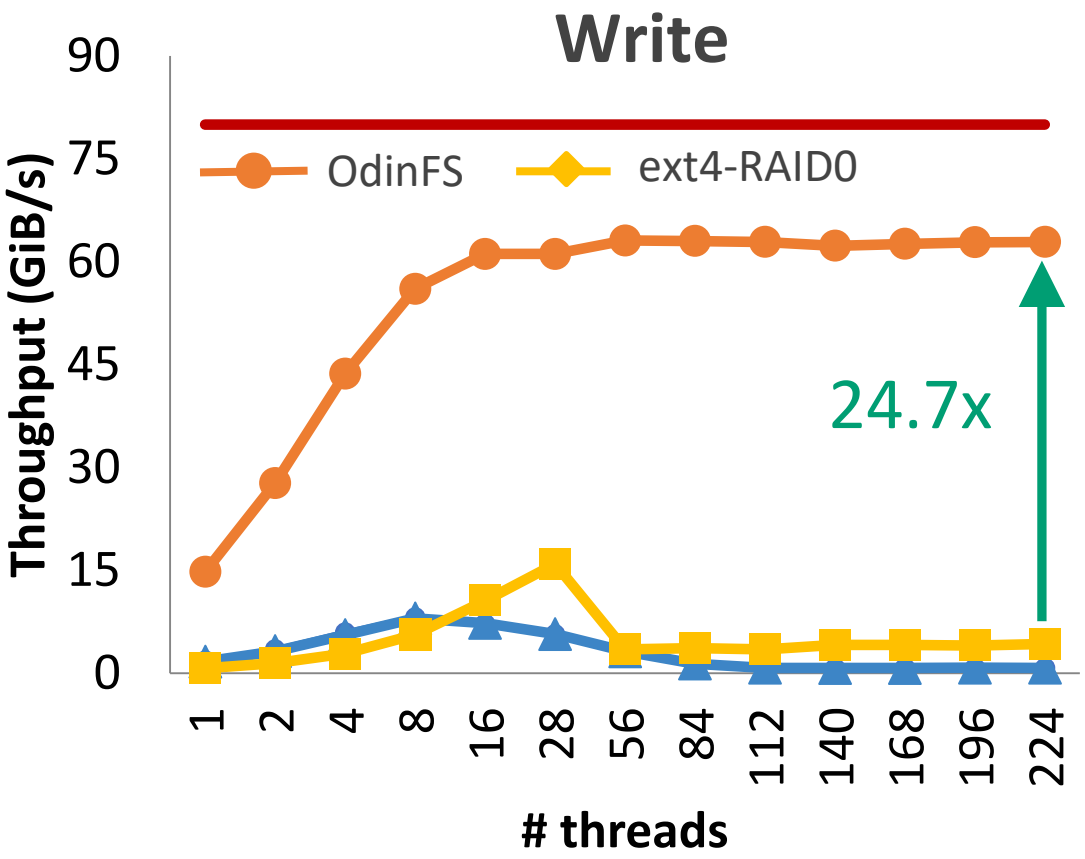


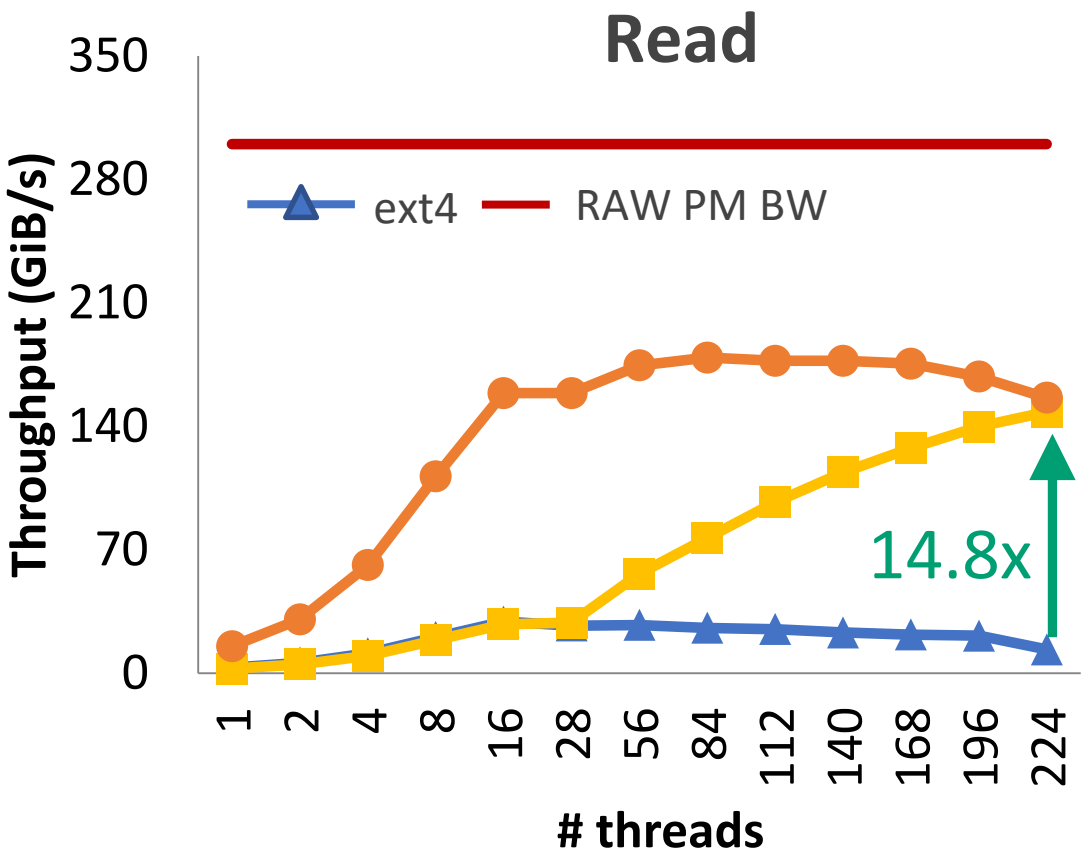


Setup: 224-core/8-socket machine

Microbenchmark: FIO

Workload: FIO: each thread writes/reads 2MB data in a private file

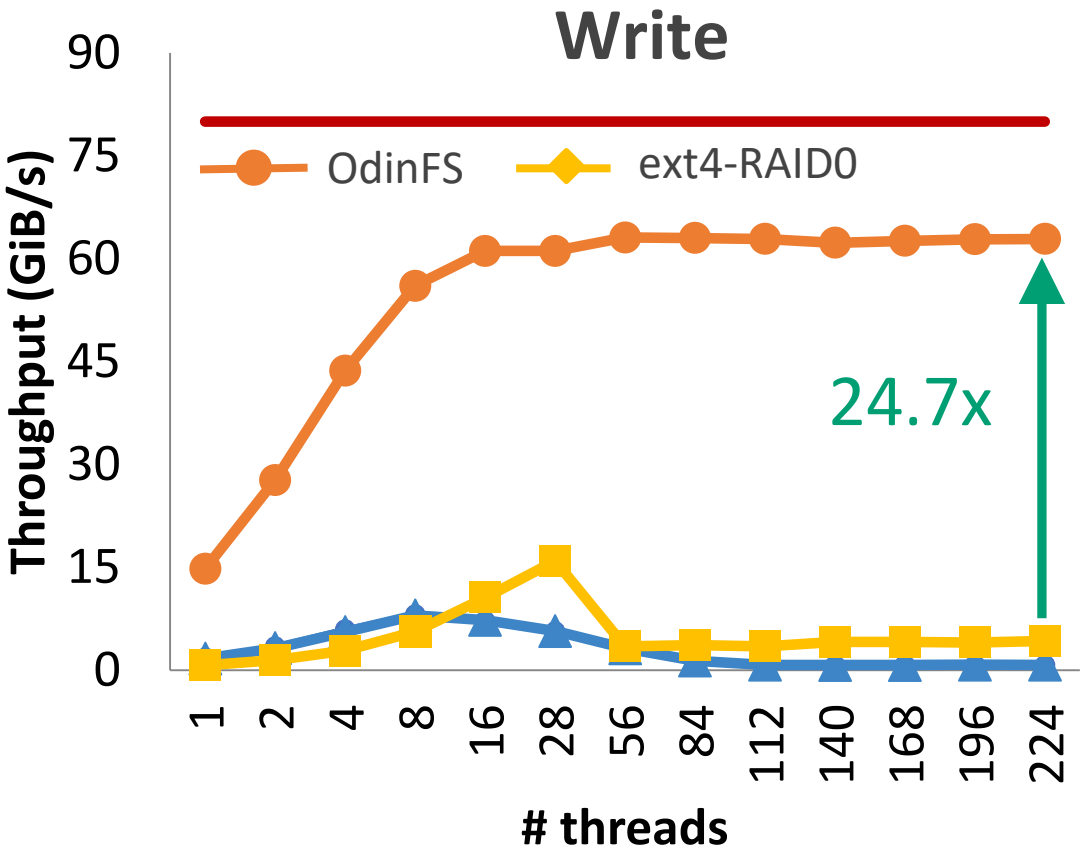




Setup: 224-core/8-socket machine

Microbenchmark: FIO

Workload: FIO: each thread writes/reads 2MB data in a private file



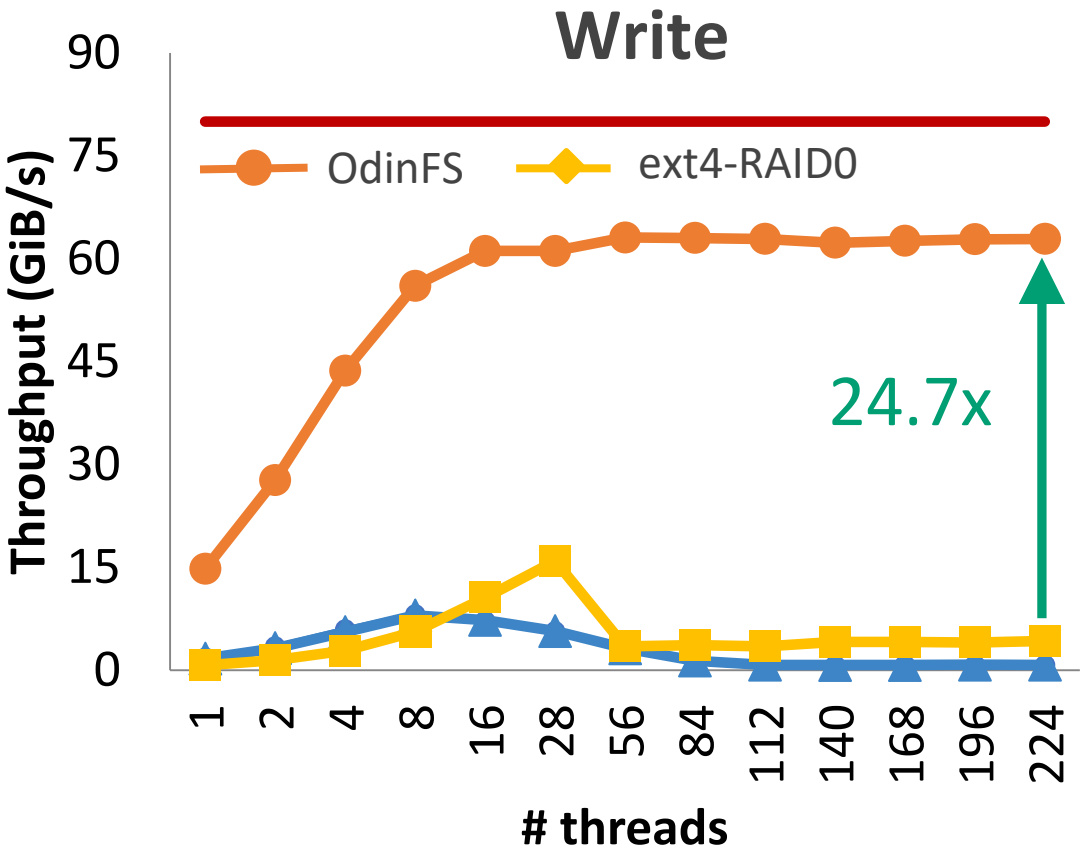
Controlled concurrent access

→ Maximize PM performance within one NUMA node

Setup: 224-core/8-socket machine

Microbenchmark: FIO

Workload: FIO: each thread writes/reads 2MB data in a private file



Controlled concurrent access

→ Maximize PM performance within one NUMA node

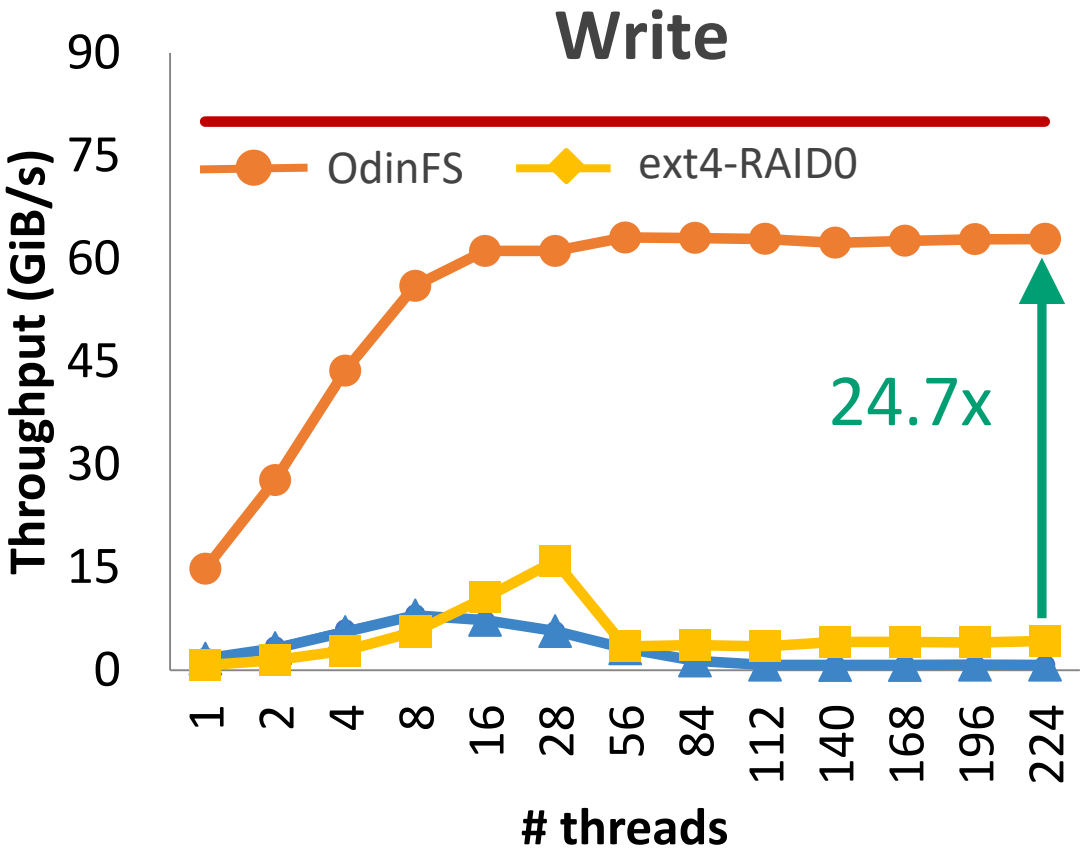
Localized PM access

→ Minimize PM NUMA impact across NUMA nodes

Setup: 224-core/8-socket machine

Microbenchmark: FIO

Workload: FIO: each thread writes/reads 2MB data in a private file



Setup: 224-core/8-socket machine

Controlled concurrent access

→ Maximize PM performance within one NUMA node

Localized PM access

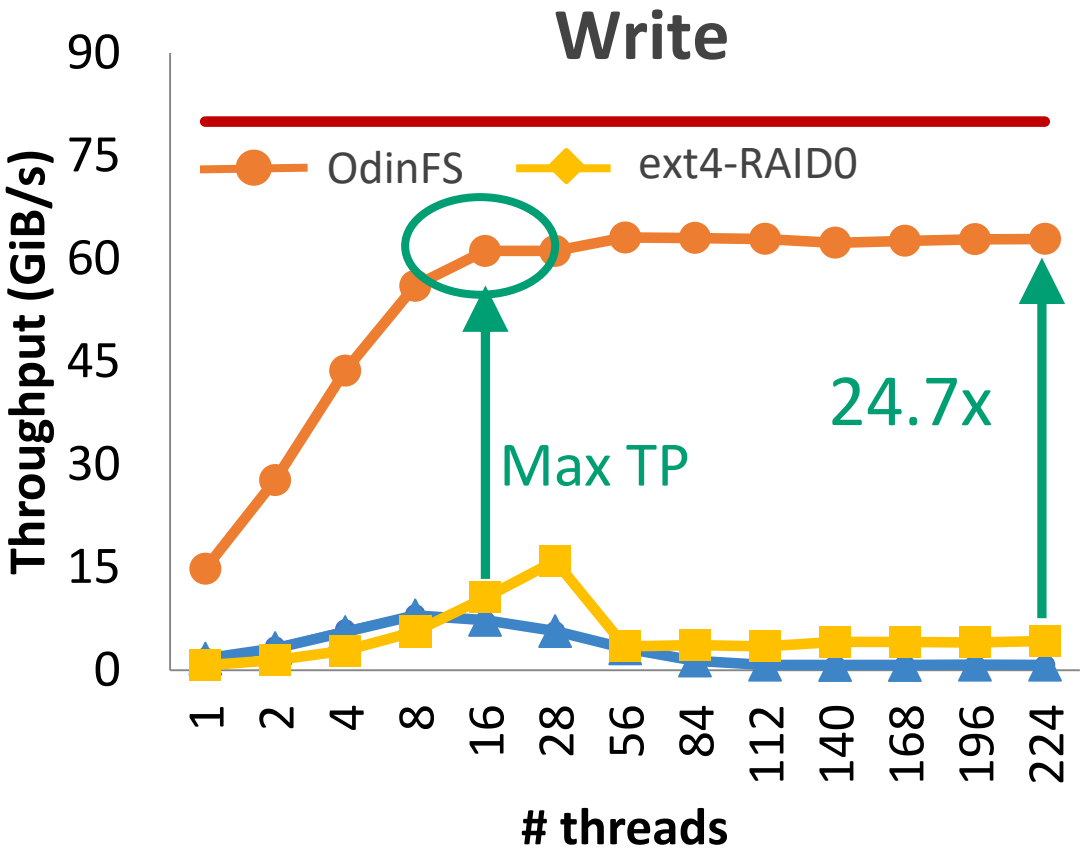
→ Minimize PM NUMA impact across NUMA nodes

Parallel PM access

→ Efficiently utilize aggregated PM bandwidth

Microbenchmark: FIO

Workload: FIO: each thread writes/reads 2MB data in a private file



Setup: 224-core/8-socket machine

Controlled concurrent access

→ Maximize PM performance within one NUMA node

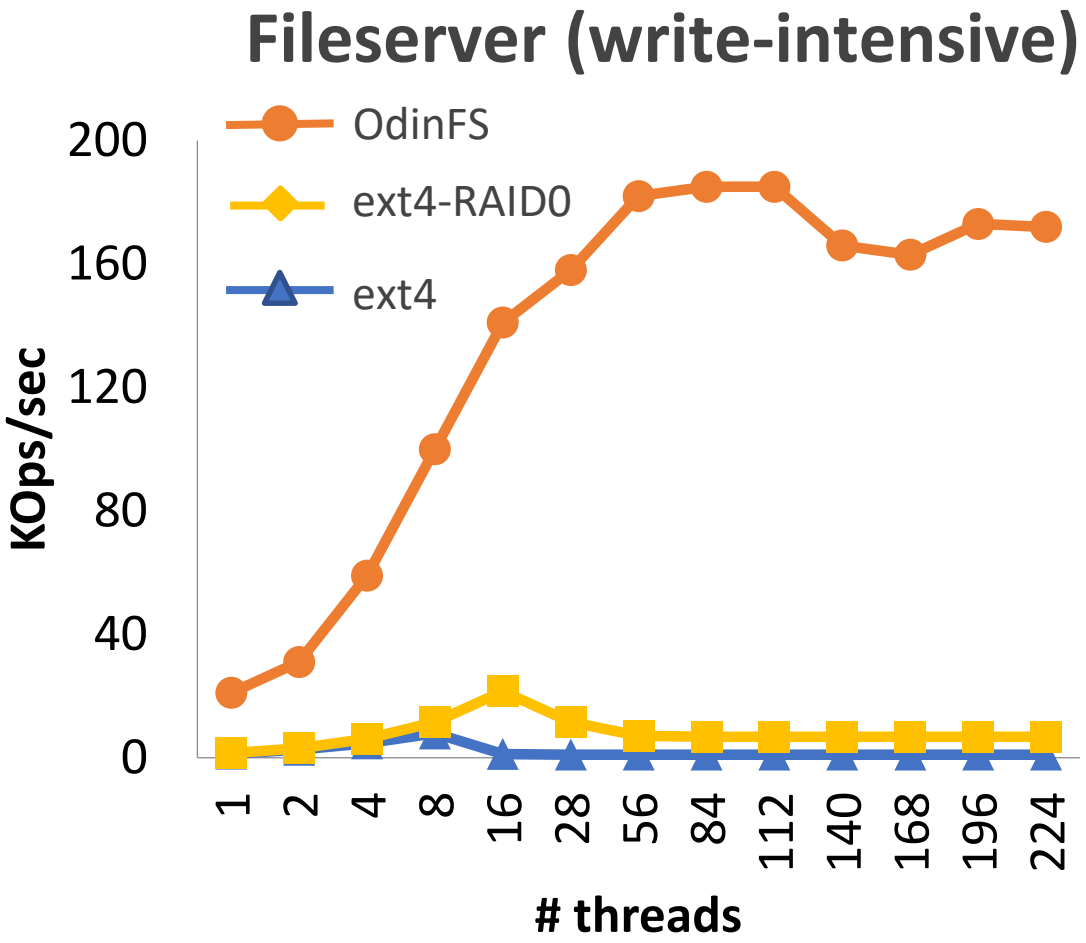
Localized PM access

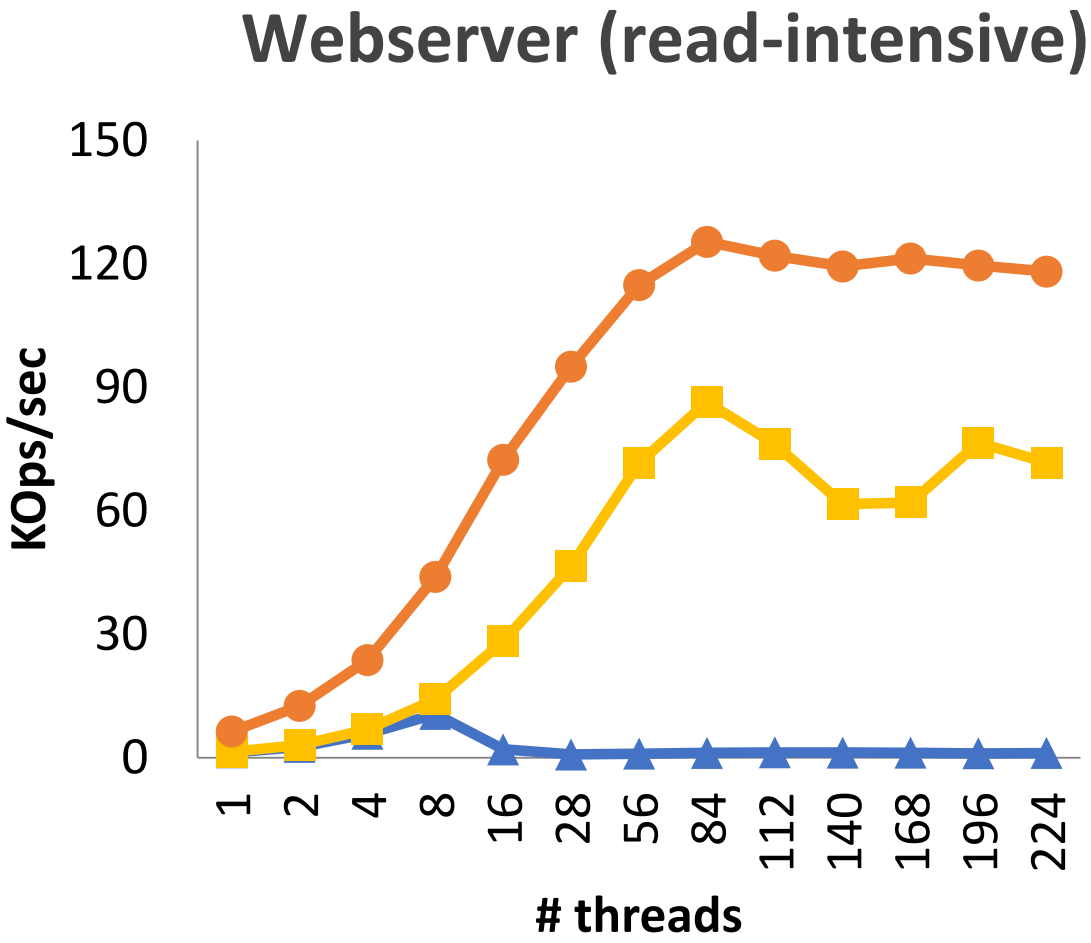
→ Minimize PM NUMA impact across NUMA nodes

Parallel PM access

→ Efficiently utilize aggregated PM bandwidth

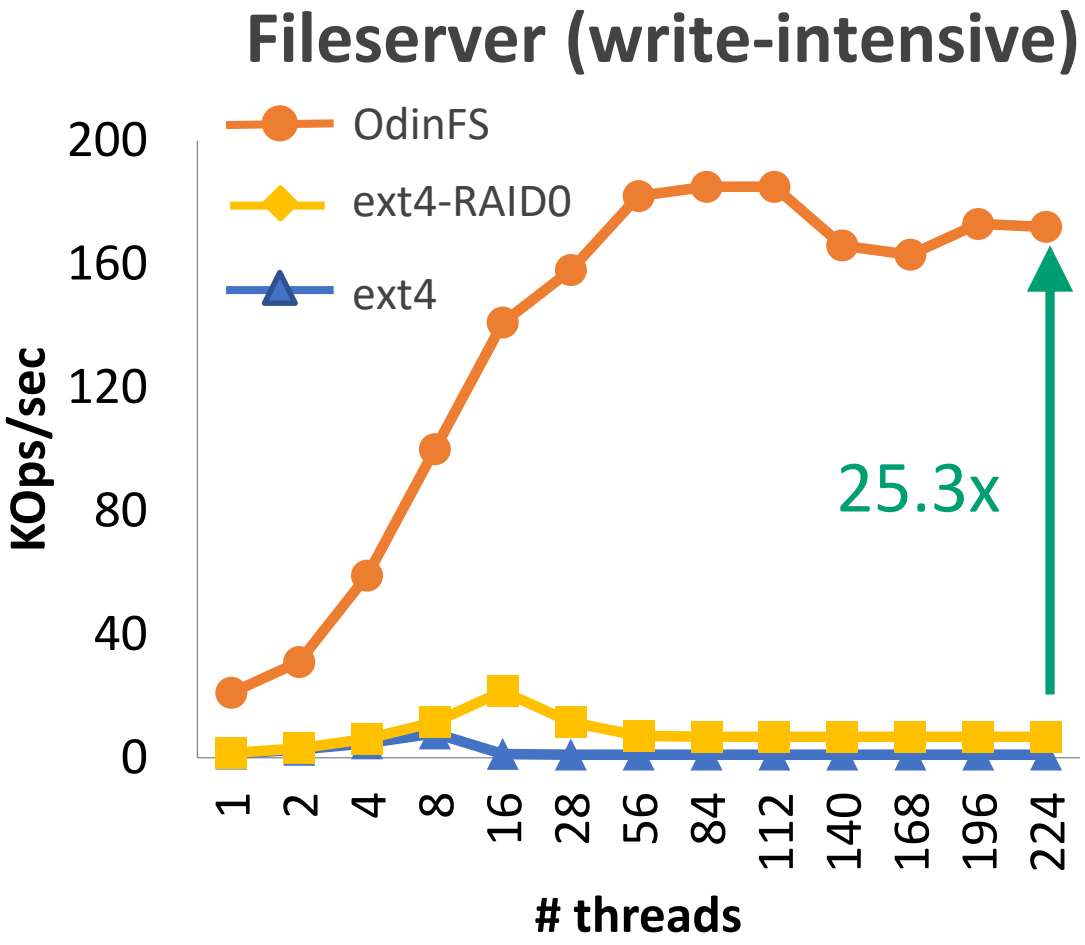
Macrobenchmark: Filebench

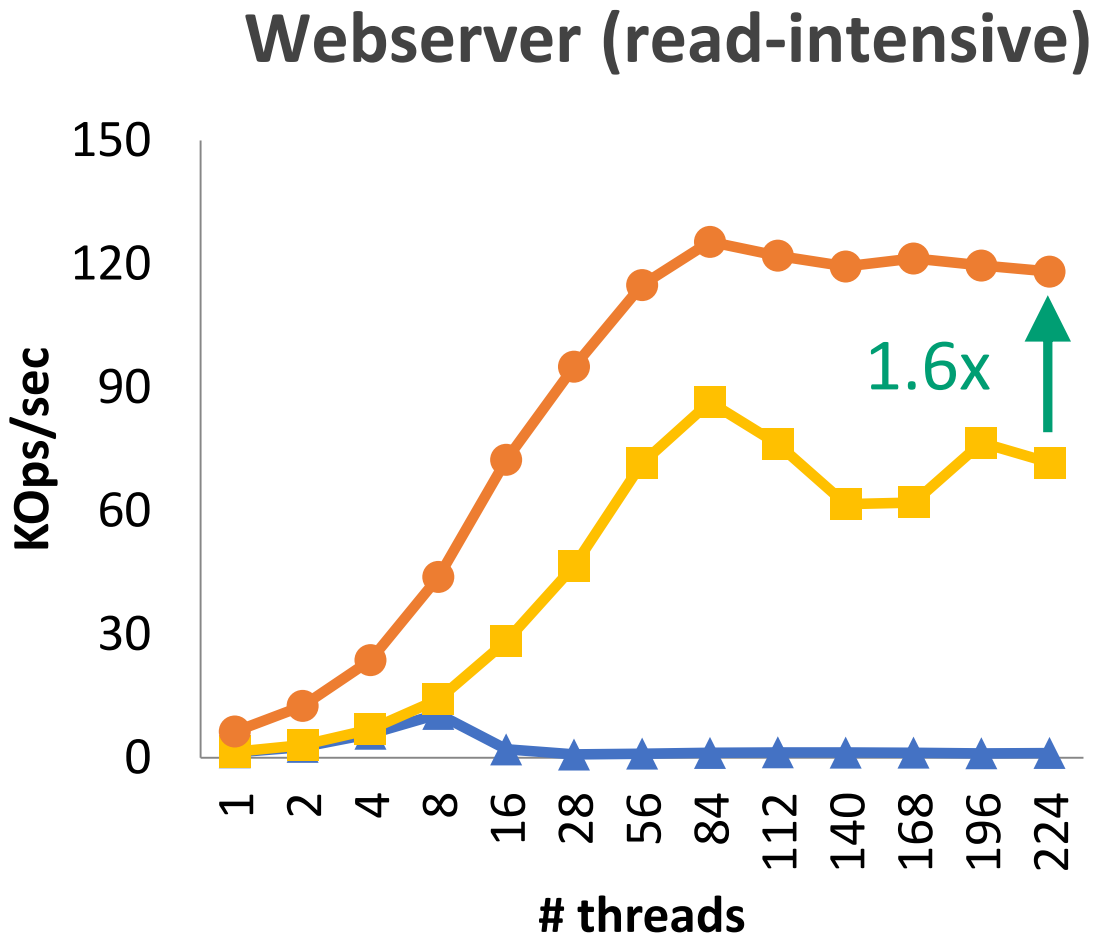




Setup: 224-core/8-socket machine

Macrobenchmark: Filebench





Setup: 224-core/8-socket machine

Conclusion

Existing file systems cannot utilize PM efficiently

- **Uncontrolled** concurrent access
- **Inefficient** remote PM access
- Cannot efficiently leverage the **aggregated PM bandwidth**

Conclusion

Existing file systems cannot utilize PM efficiently

- **Uncontrolled** concurrent access
- **Inefficient** remote PM access
- Cannot efficiently leverage the **aggregated PM bandwidth**

OdinFS: decouple PM access to maximize and scale performance

- **Controlled, localized,** and **parallel PM access**

Conclusion

Existing file systems cannot utilize PM efficiently

- **Uncontrolled** concurrent access
- **Inefficient** remote PM access
- Cannot efficiently leverage the **aggregated PM bandwidth**

OdinFS: decouple PM access to maximize and scale performance

- **Controlled, localized,** and **parallel PM access**
- **Publicly available:** https://github.com/rs3lab/Odinfs

Conclusion

Existing file systems cannot utilize PM efficiently

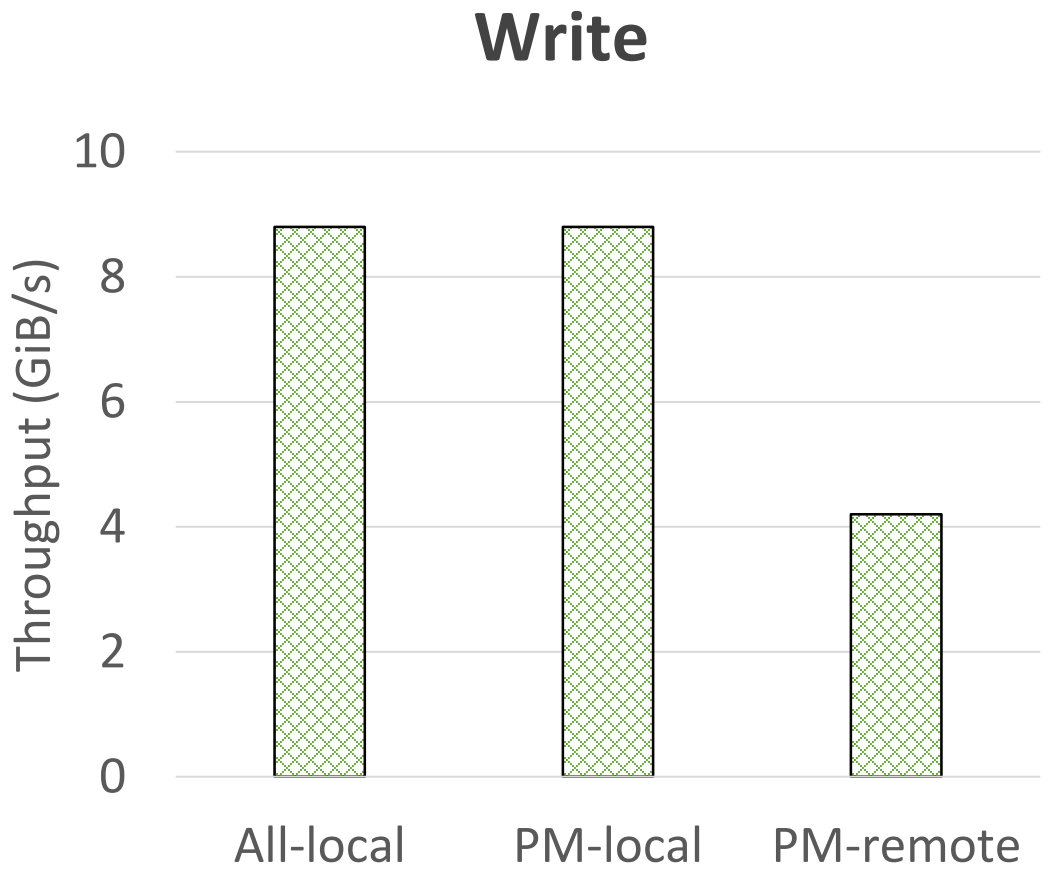
- **Uncontrolled** concurrent access
- **Inefficient** remote PM access
- Cannot efficiently leverage the **aggregated PM bandwidth**

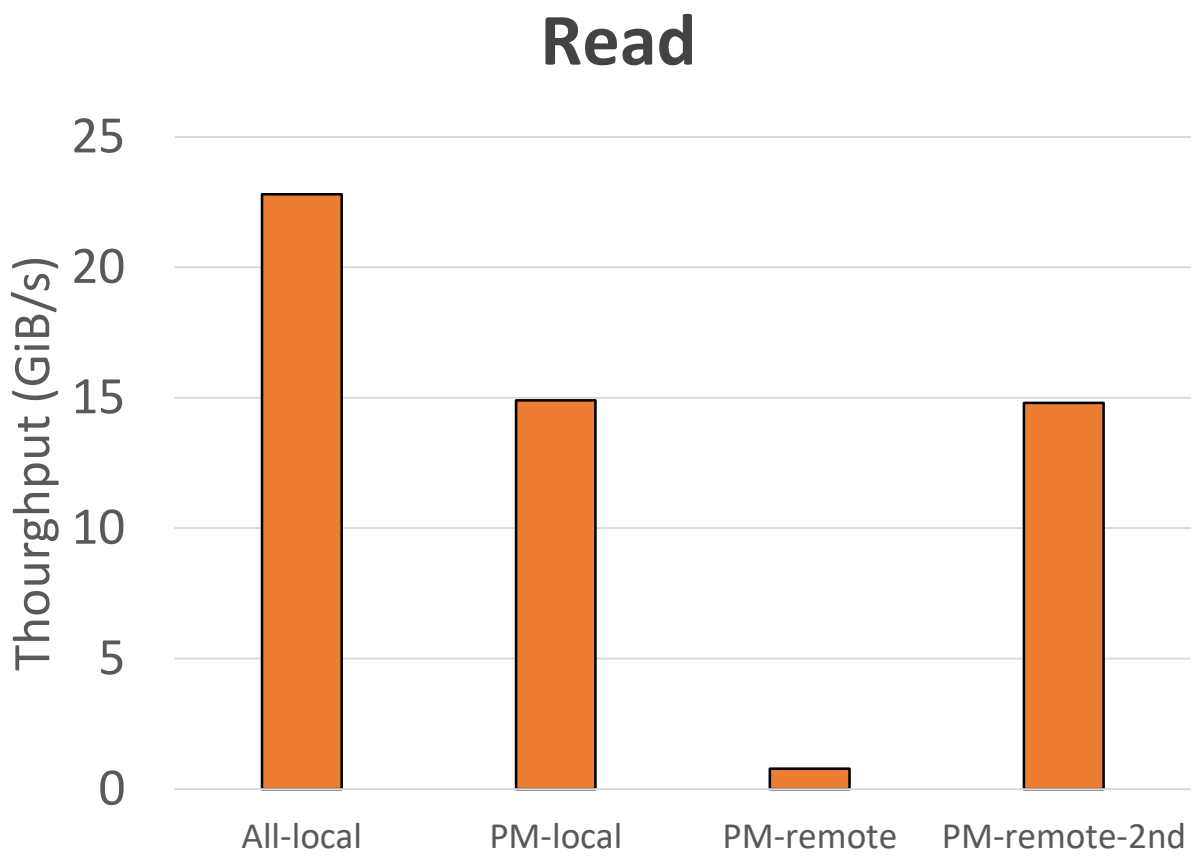
OdinFS: decouple PM access to maximize and scale performance

- **Controlled, localized,** and **parallel PM access**
- **Publicly available:** https://github.com/rs3lab/Odinfs

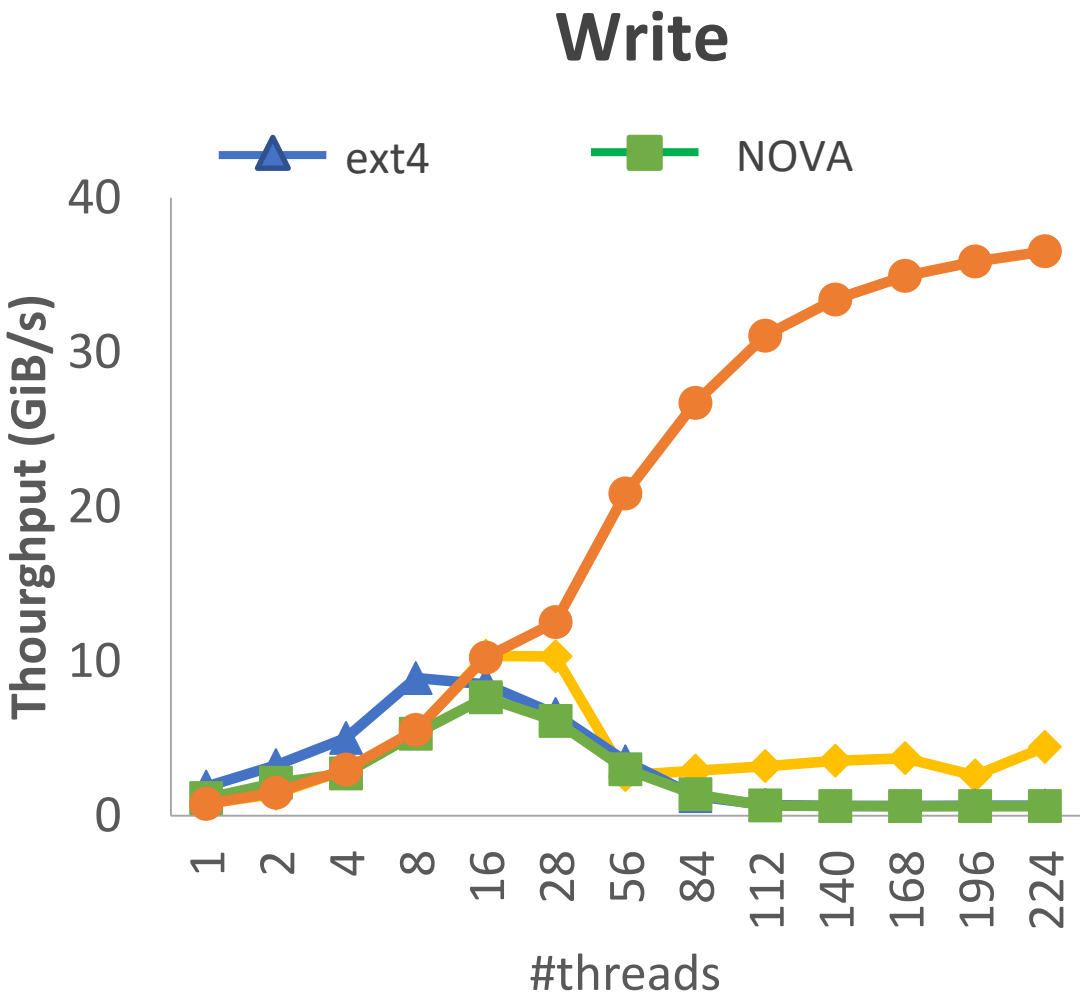
Thank you!

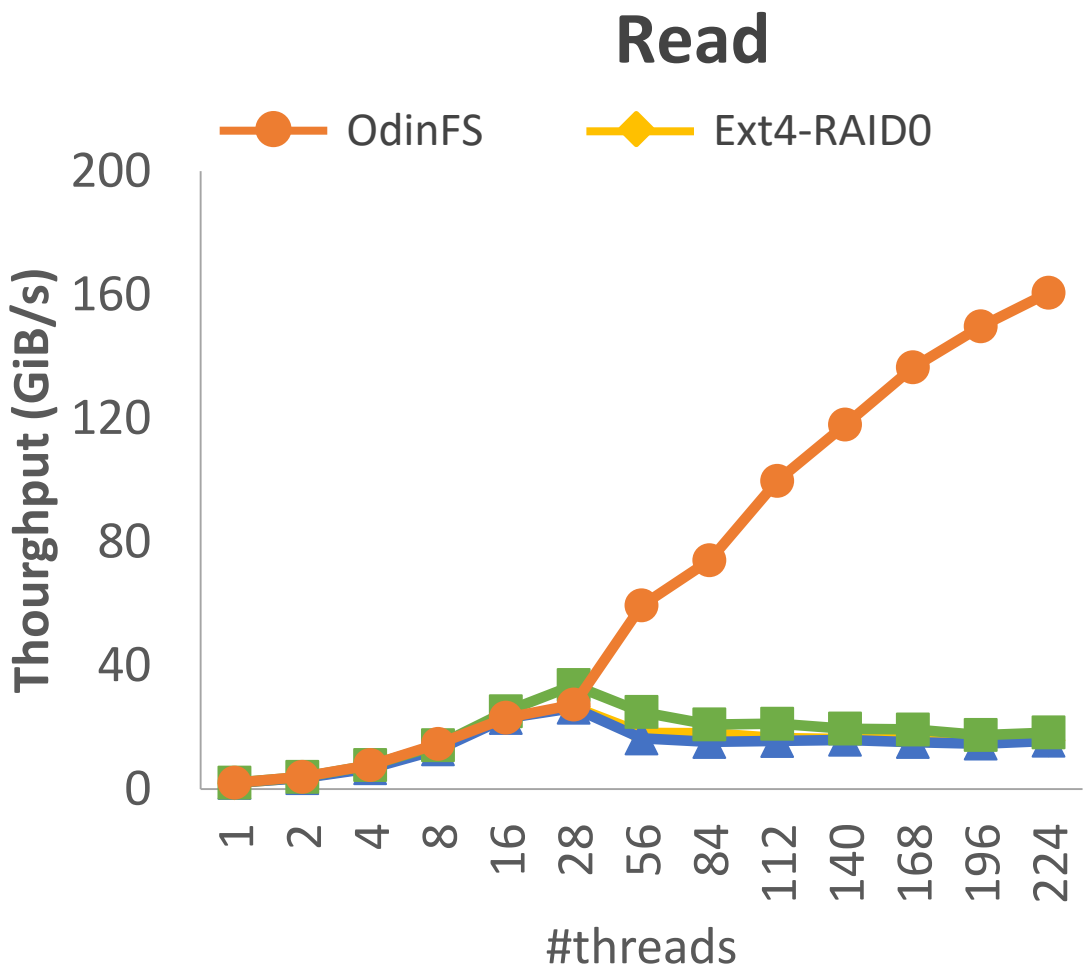
PM performance under NUMA setup



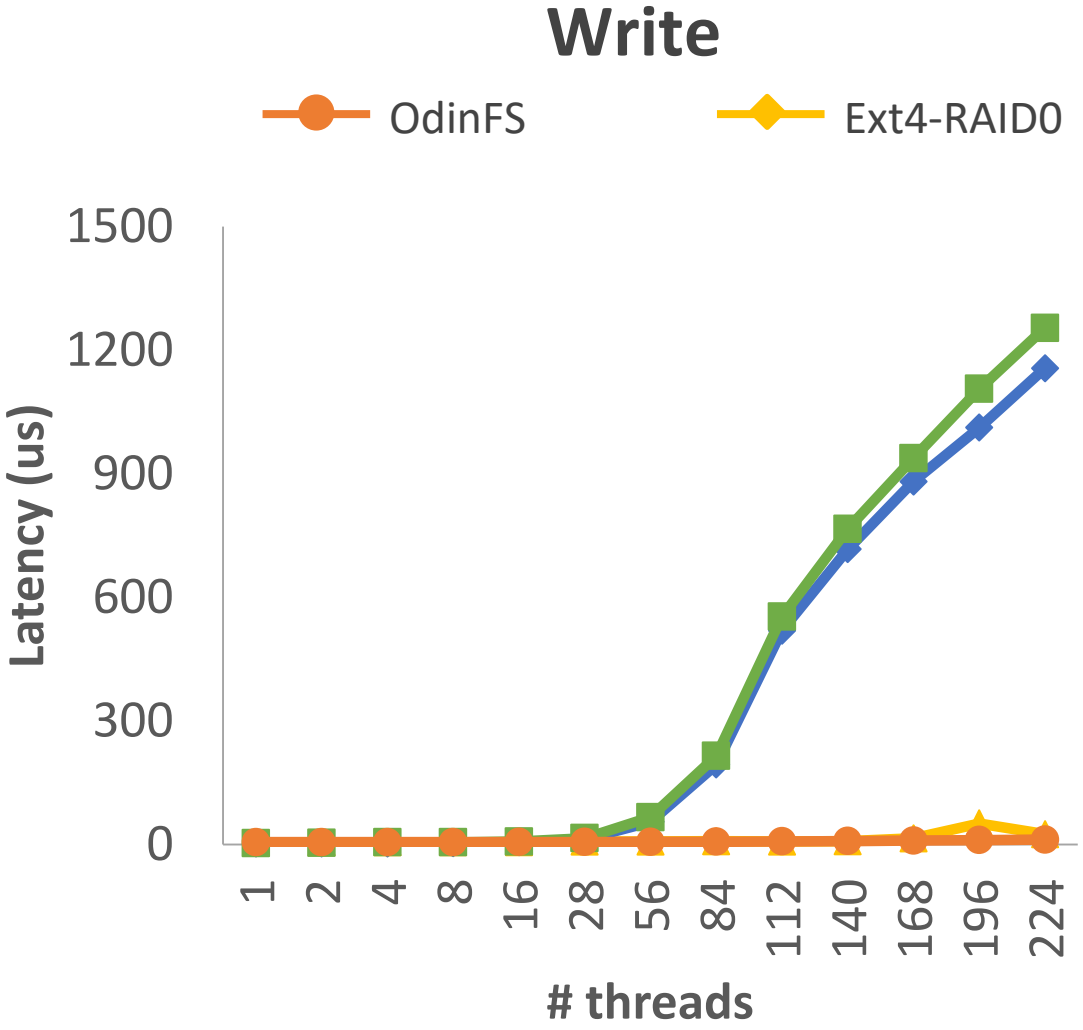


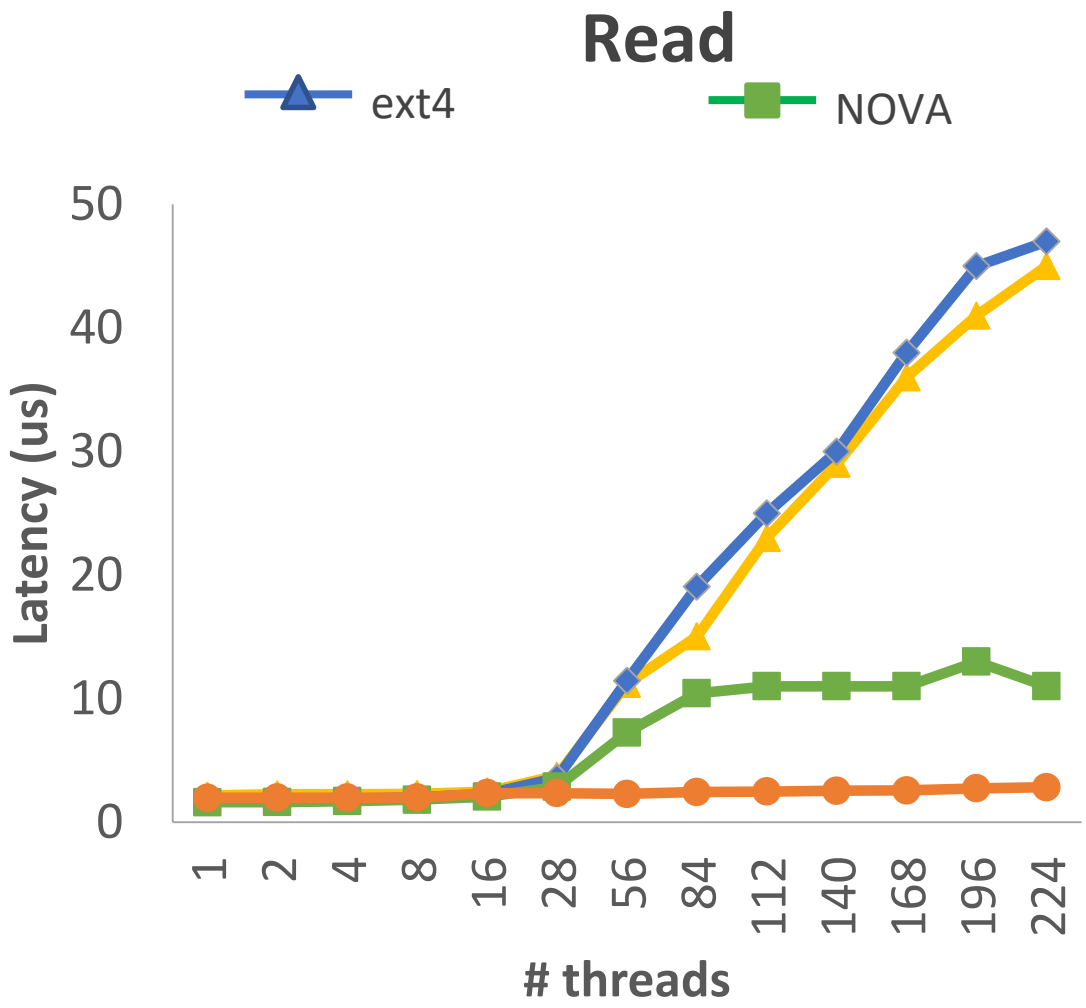
FIO: 4K access size



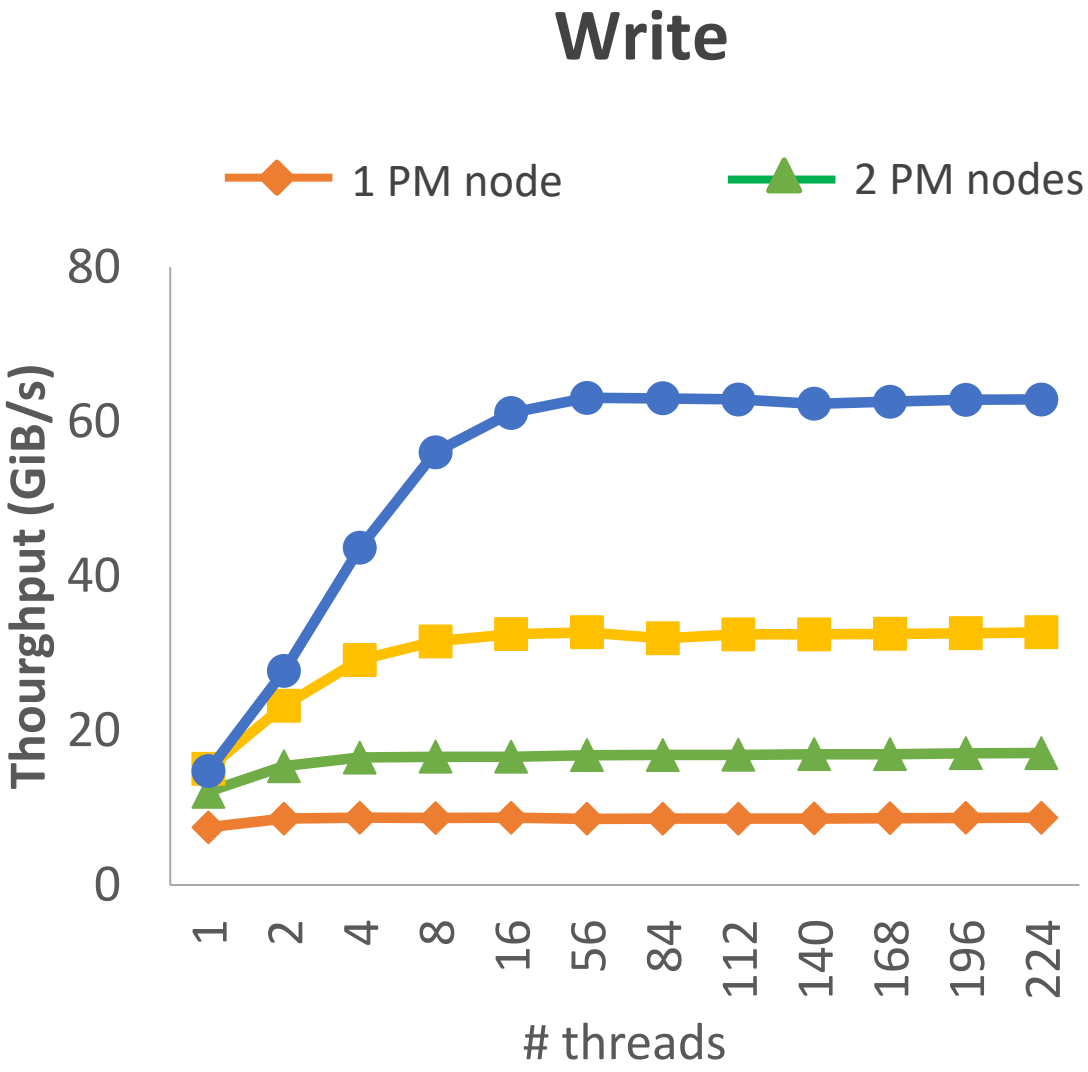


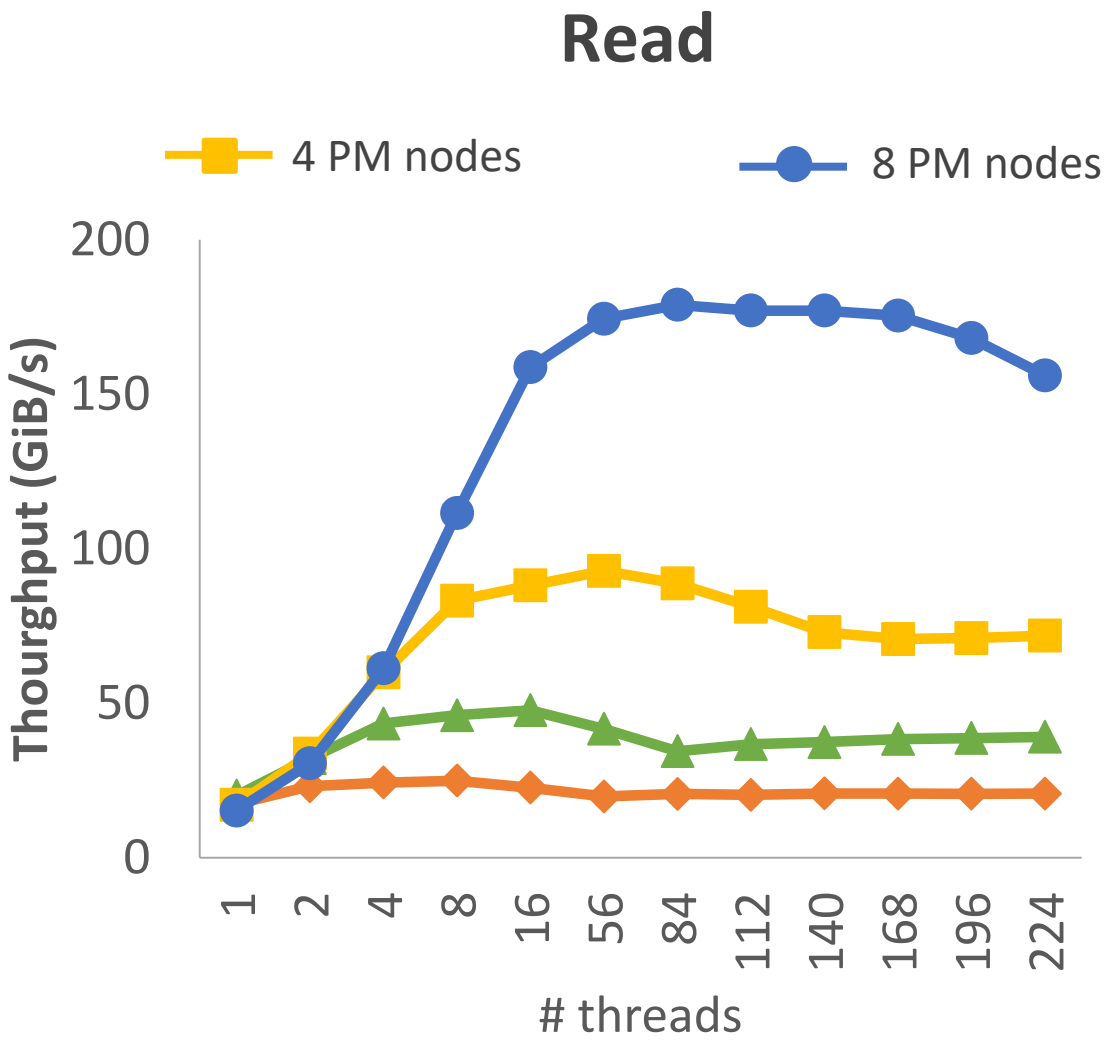
FIO: Latency, 4K access size





OdinFS with different number of NUMA nodes





PM NUMA Impact: Explanation