Effectively Prefetching Remote Memory with Leap

Hasan Al Maruf and *Mosharaf Chowdhury*

UNIVERSITY OF MICHIGAN

Memory-Intensive Applications

powergraph

NumPy

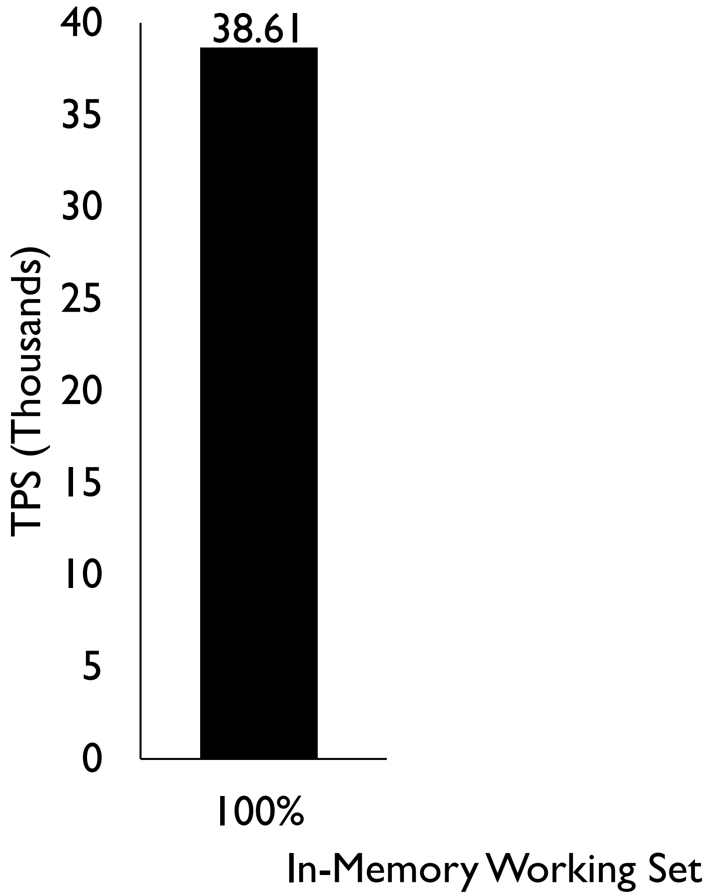
APACHE Spark™

VOLTDB

redis

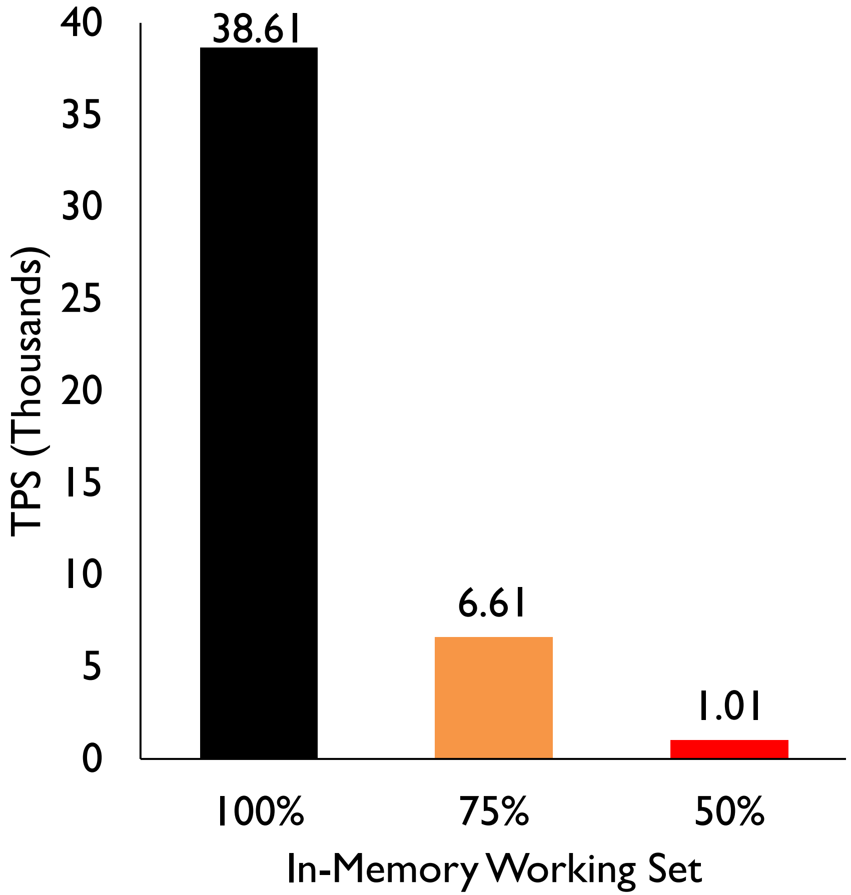
MEMCACHED

Perform Great!



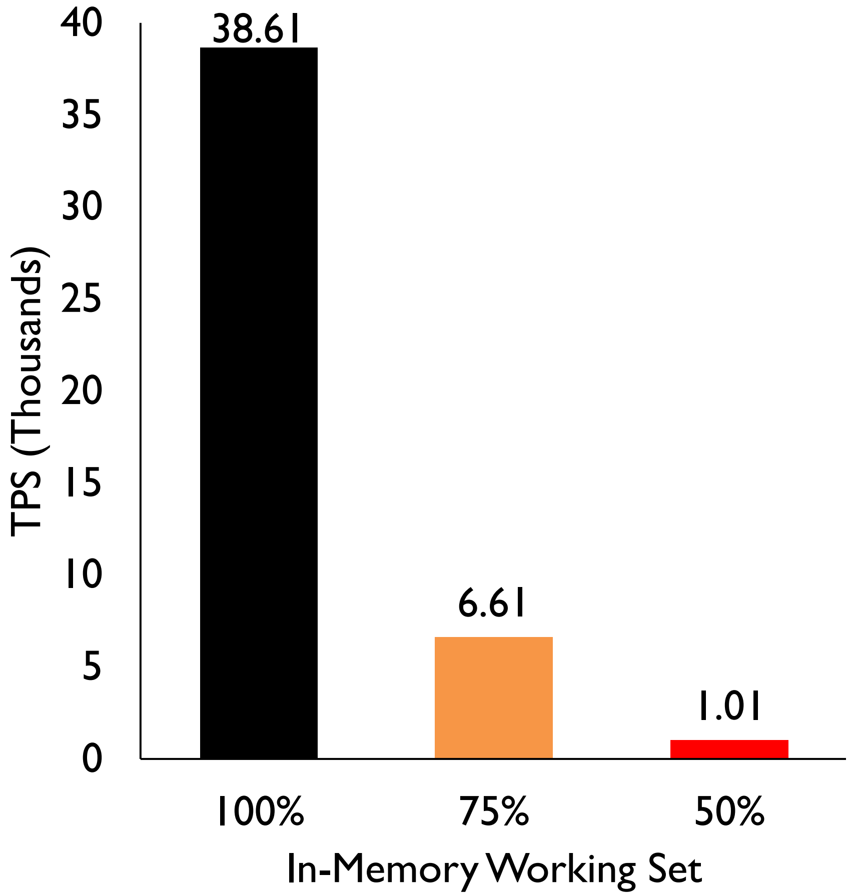
TPC-C on VoltDB

Perform Great Until Memory Runs Out

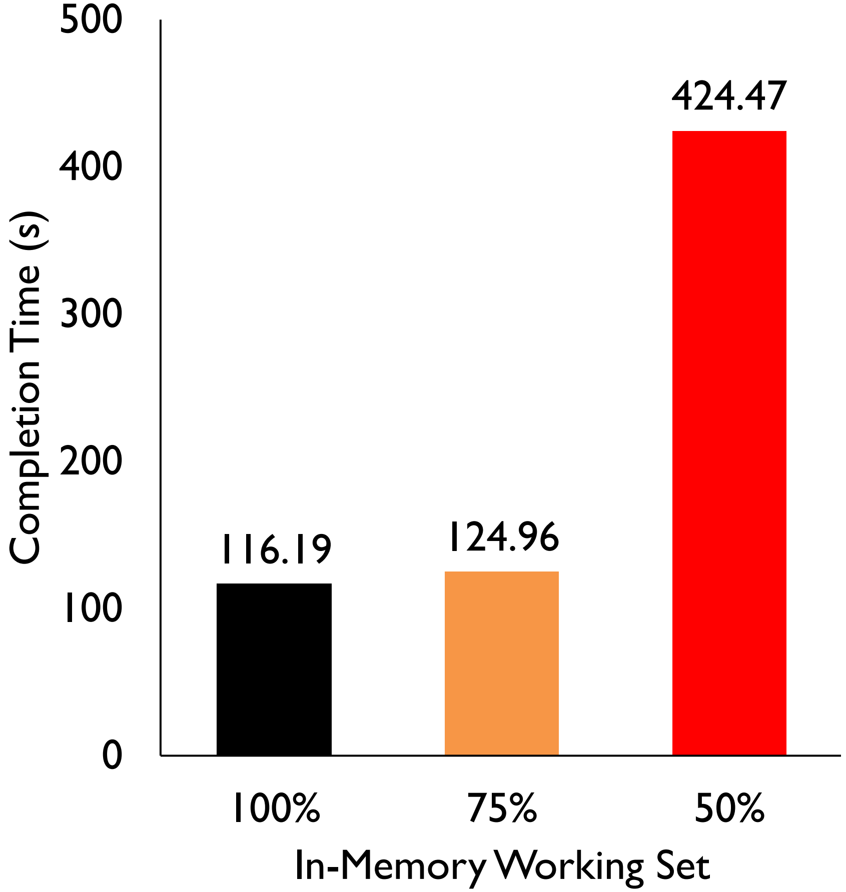


TPC-C on VoltDB

Perform Great Until Memory Runs Out

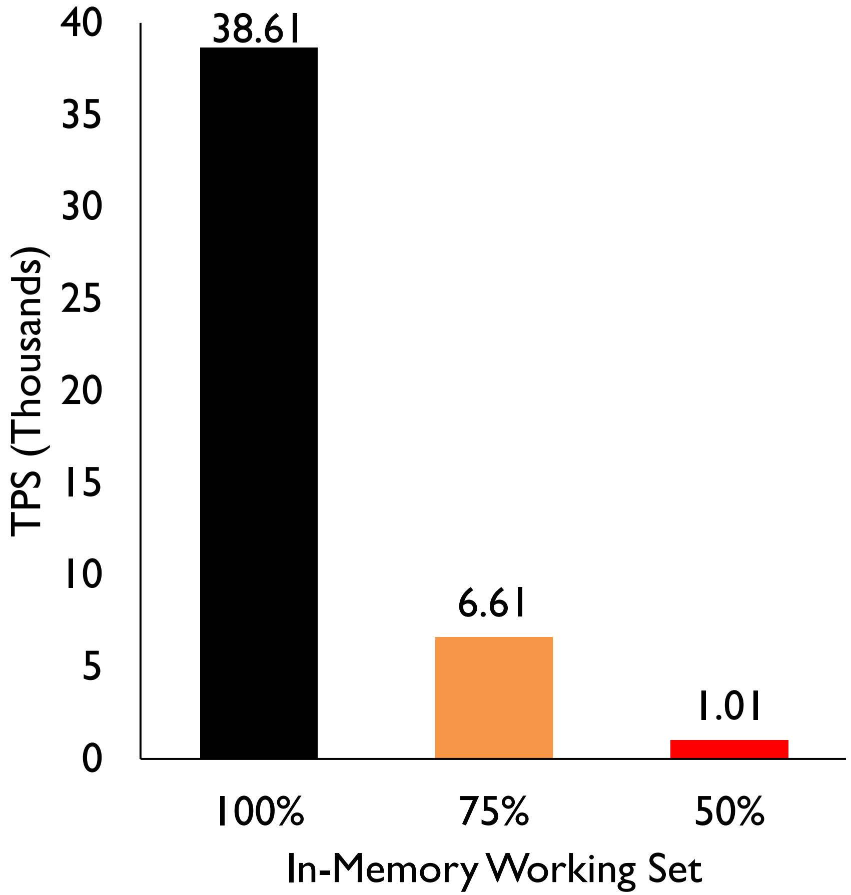


TPC-C on VoltDB



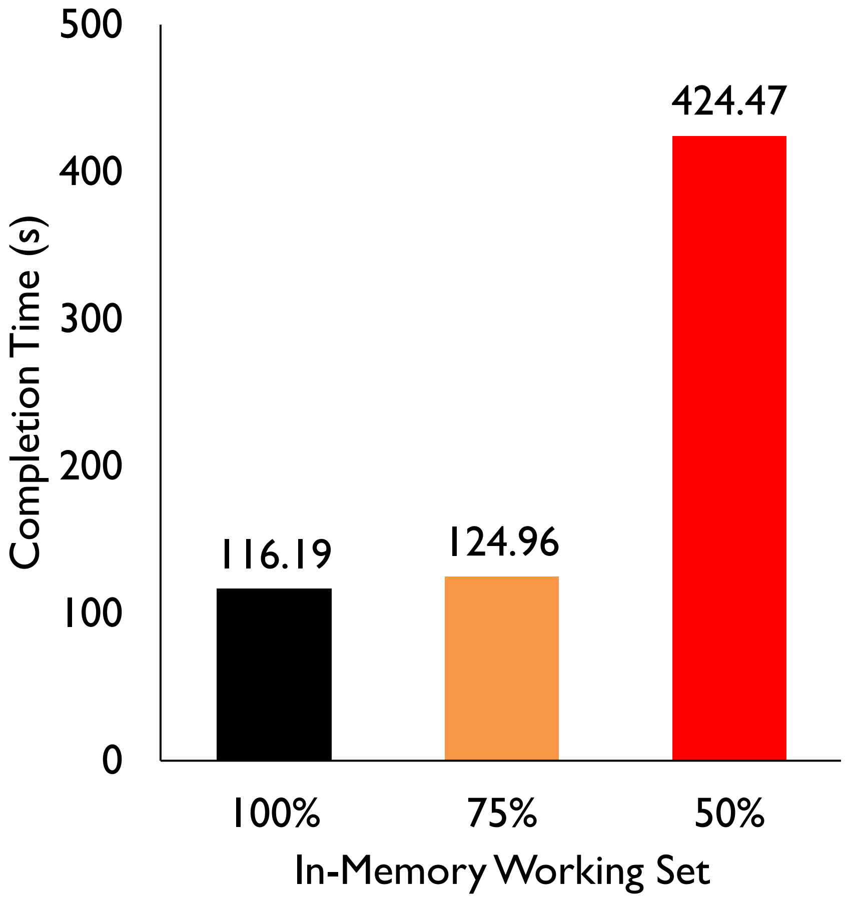
PageRank on PowerGraph

50% Less Memory Causes Slowdown of …



TPC-C on VoltDB

38X



PageRank on PowerGraph

4X

Between a Rock and a Hard Place

Underallocation

Leads to severe performance loss

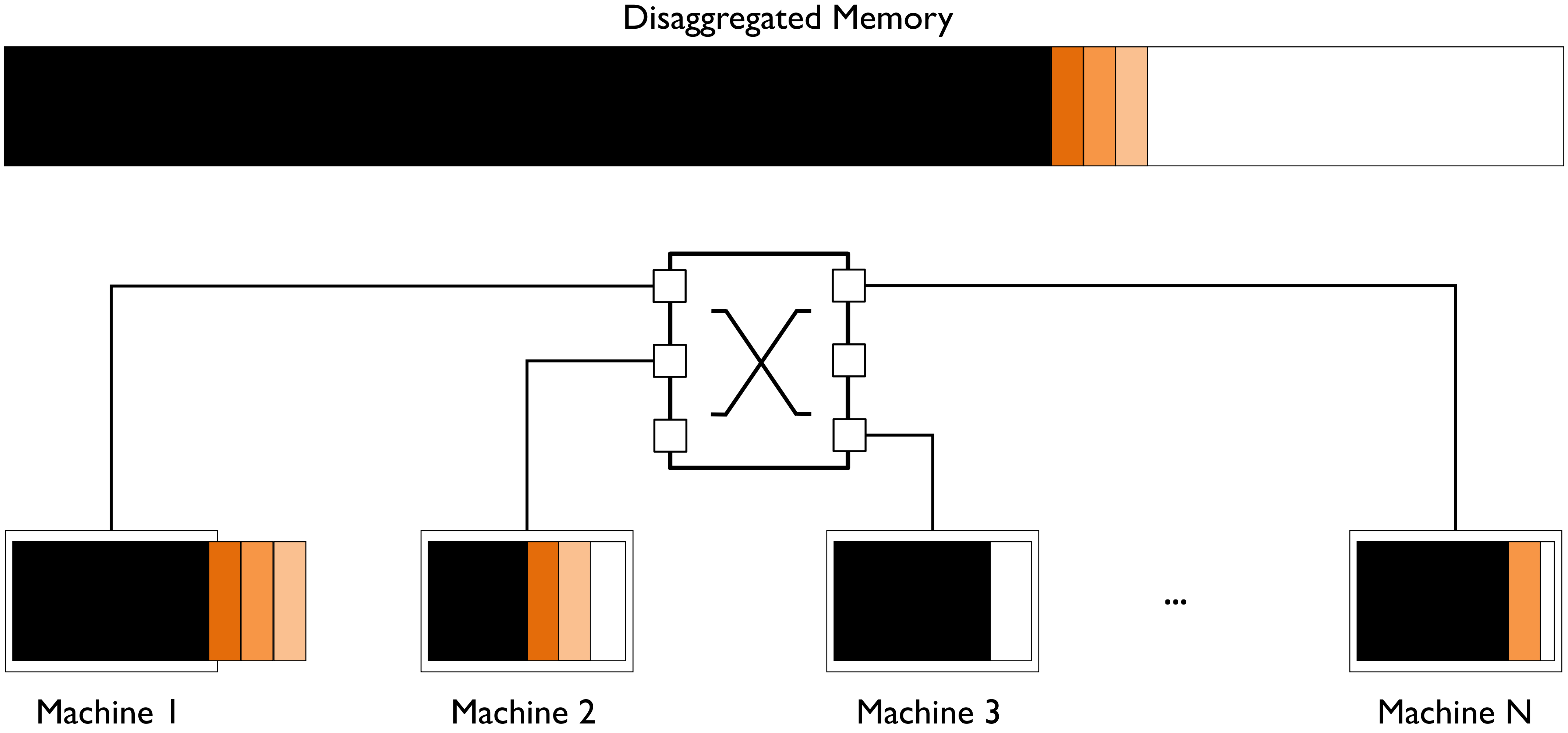
vs.

Overallocation

Leads to underutilization

30-40% in Google, Alibaba, and Facebook

Memory Disaggregation



Disaggregated Memory

Machine 1

Machine 2

Machine 3

...

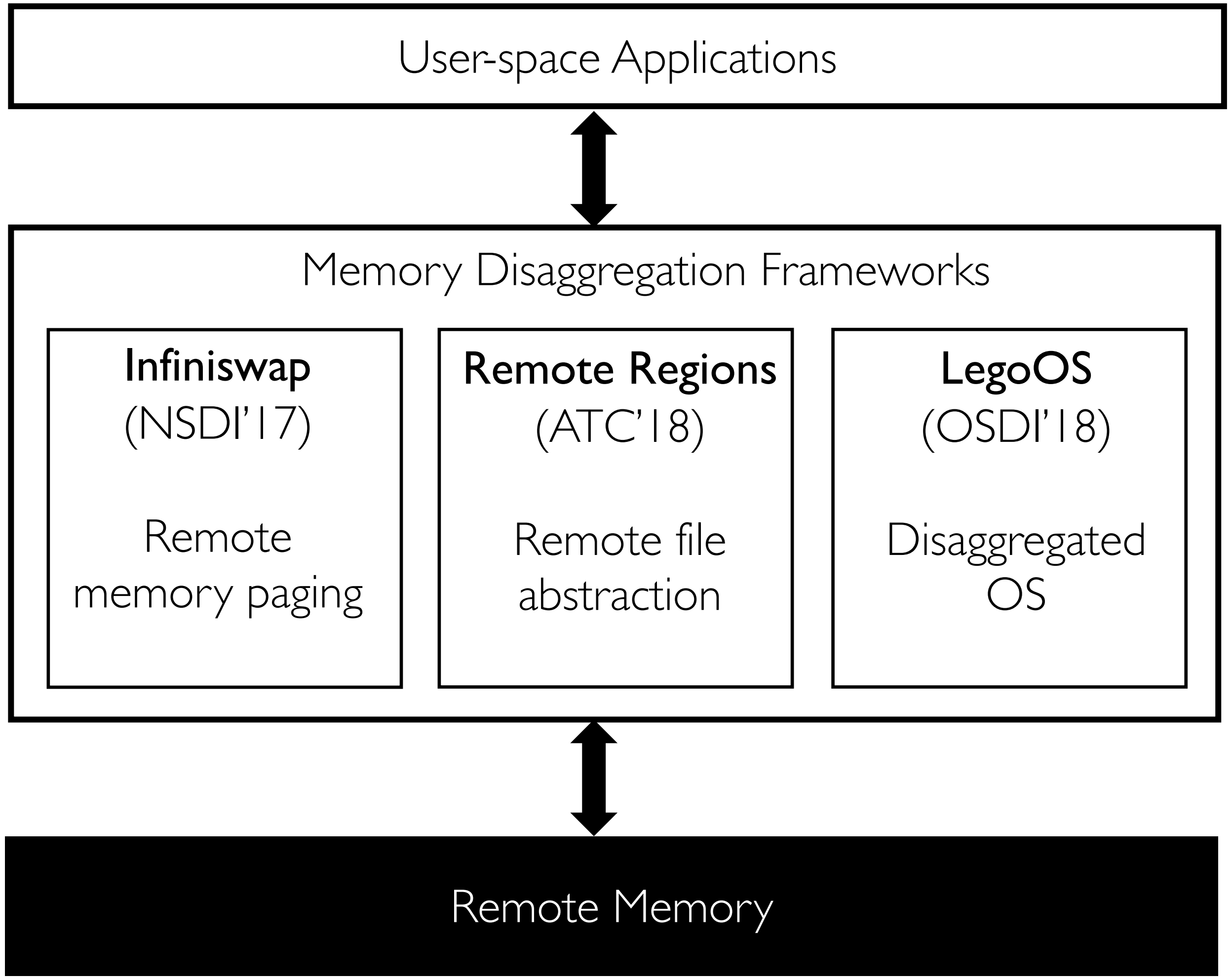
Machine N

■ Used Memory

□ Free Memory

🟧 Remote Memory

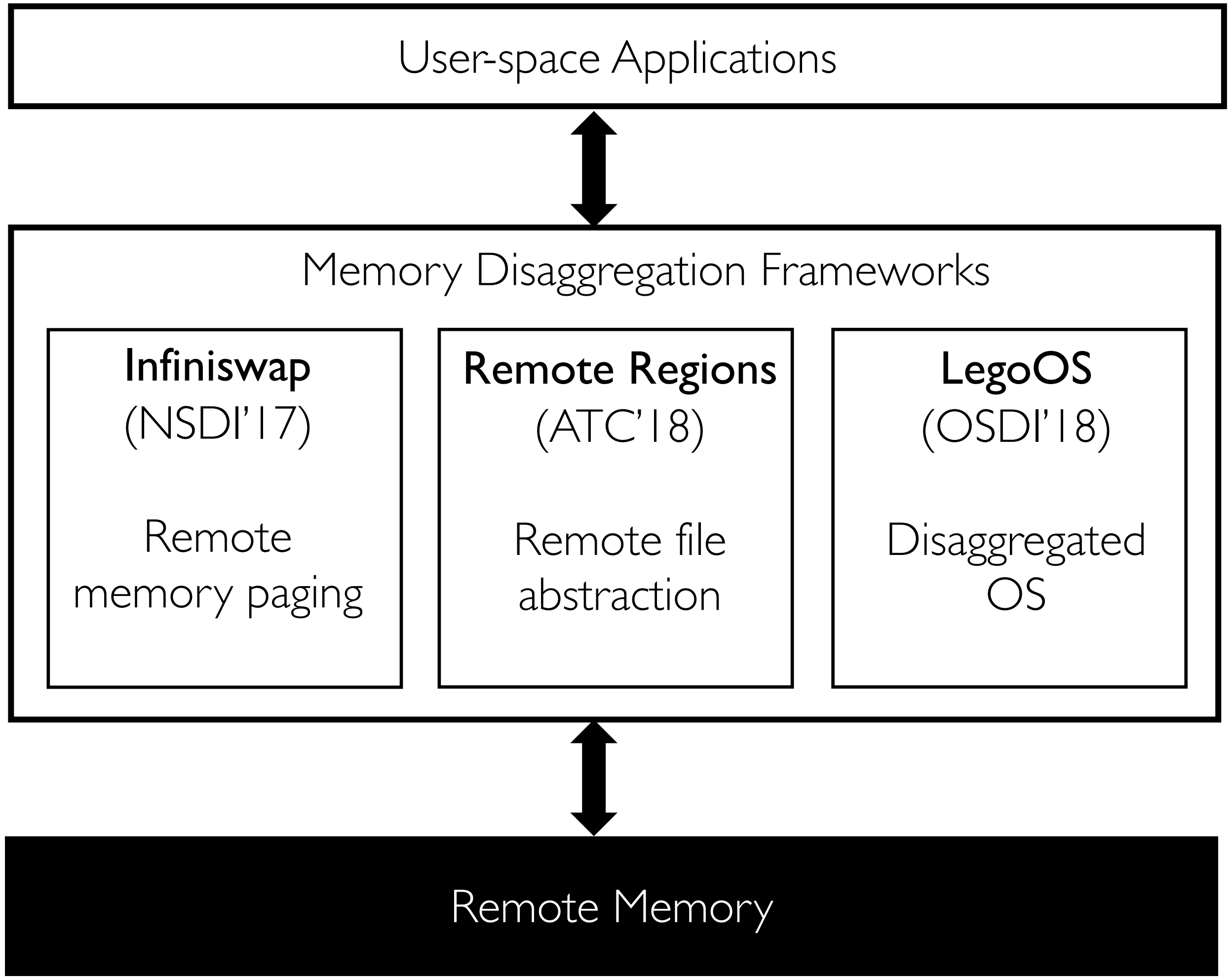
Remote Memory Access



4KB page access latency
local vs. remote

100 ns vs. **4 µs**

Remote Memory Access



4KB page access latency
local vs. remote

100 ns vs. **4 µs**

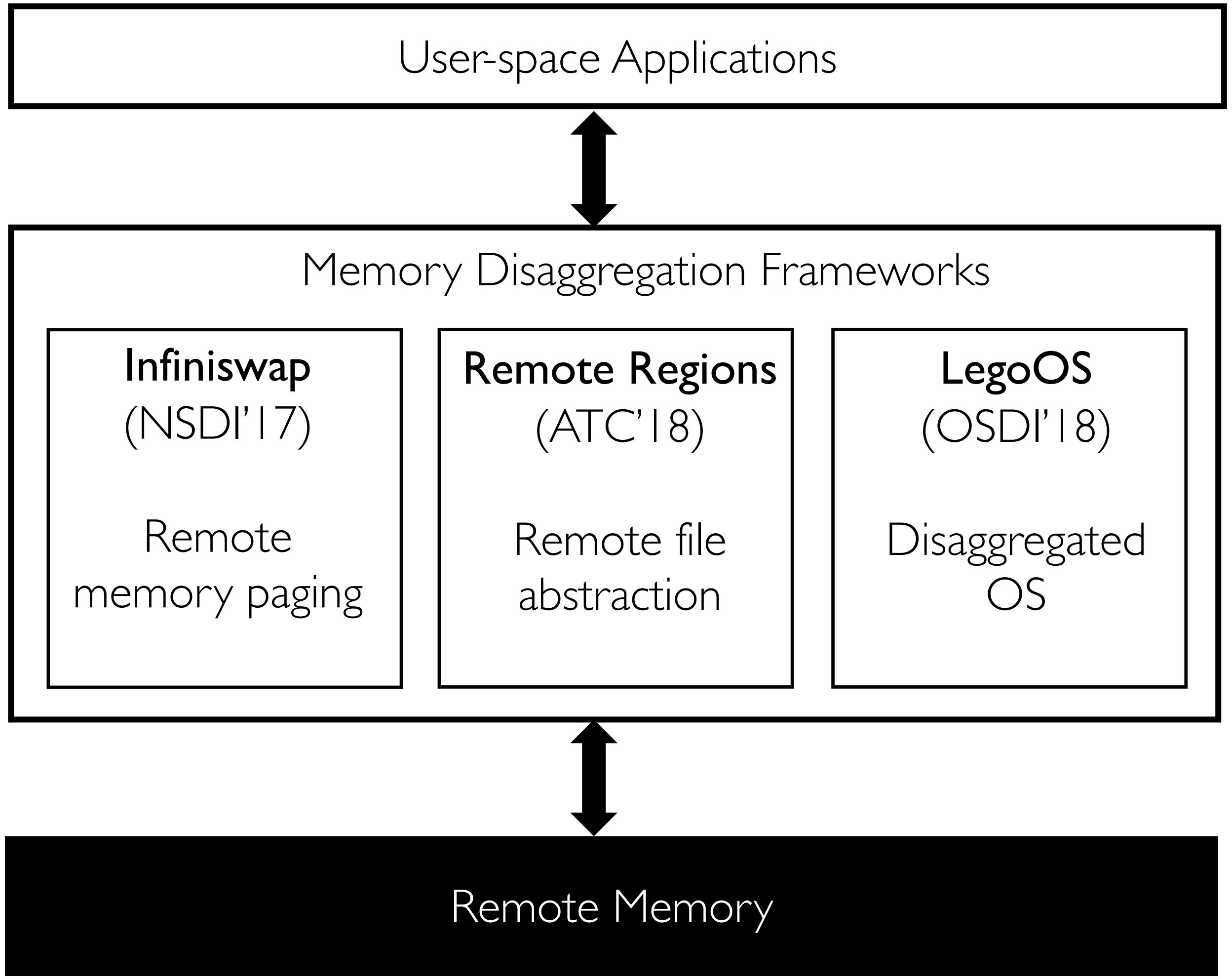
Latency requirement for preferable performance[1]

3 µs

Existing frameworks can't achieve!

[1] P. X. Gao et al. "Network requirements for resource disaggregation" OSDI'16.

Remote Memory Access



data path overhead

variation in network latency

4KB page access latency
local vs. remote

100 ns vs. **4 µs**

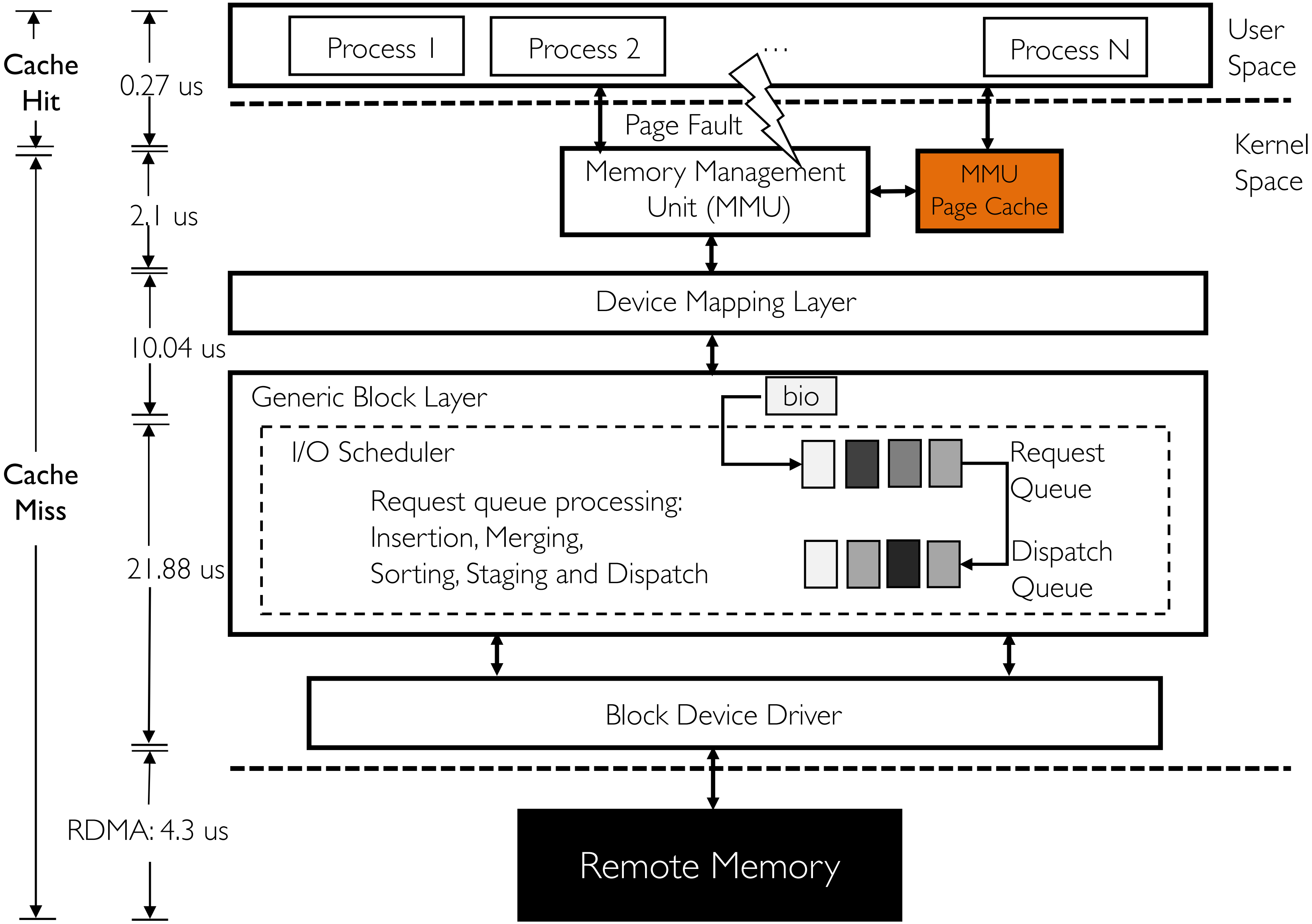
Latency requirement for preferable performance[1]

3 µs

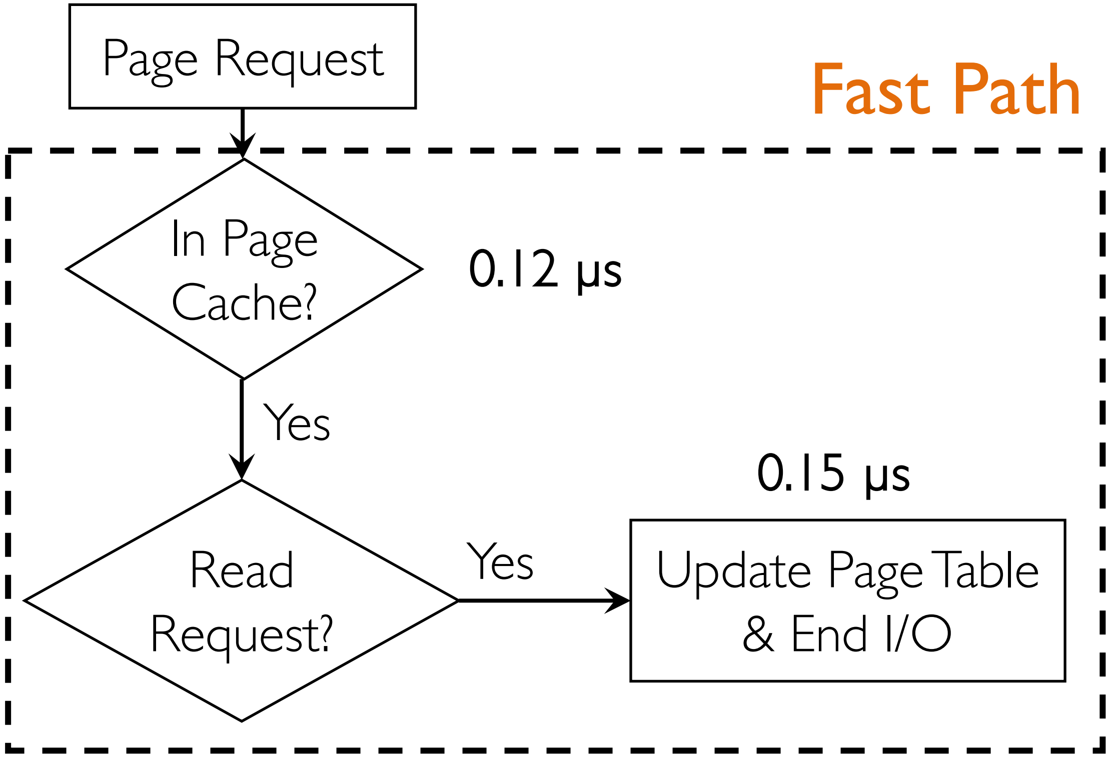
Existing frameworks can't achieve!

[1] P. X. Gao et al. "Network requirements for resource disaggregation" OSDI'16.

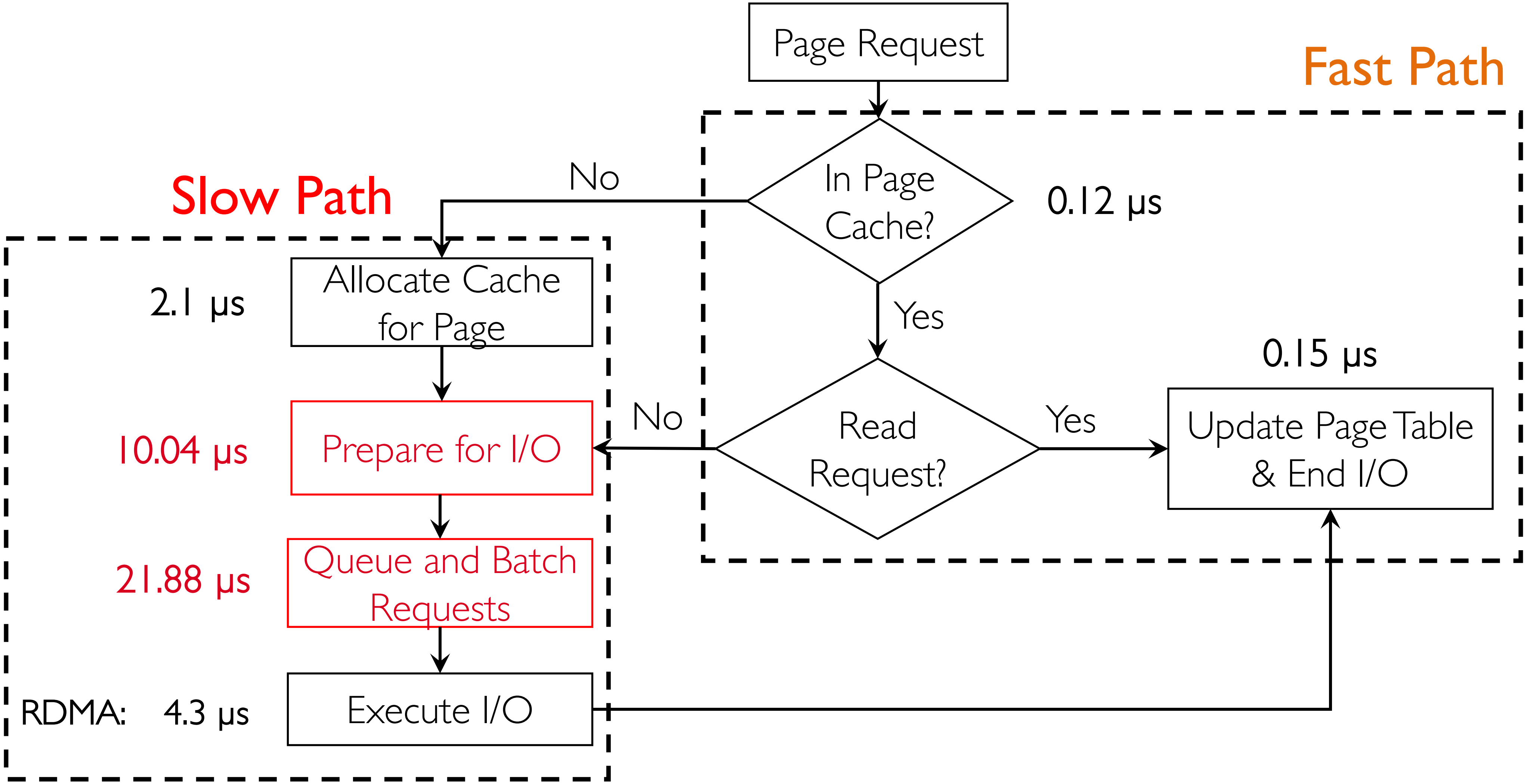
Life of a Page



Where Does the Time Go?



Where Does the Time Go?



Design Goal

1. Increase cache hit
    - faster path serves more page faults

2. Reduce the latency of the slow path
    - remove unnecessary block-layer operations for RDMA

Leap

Online remote memory prefetcher

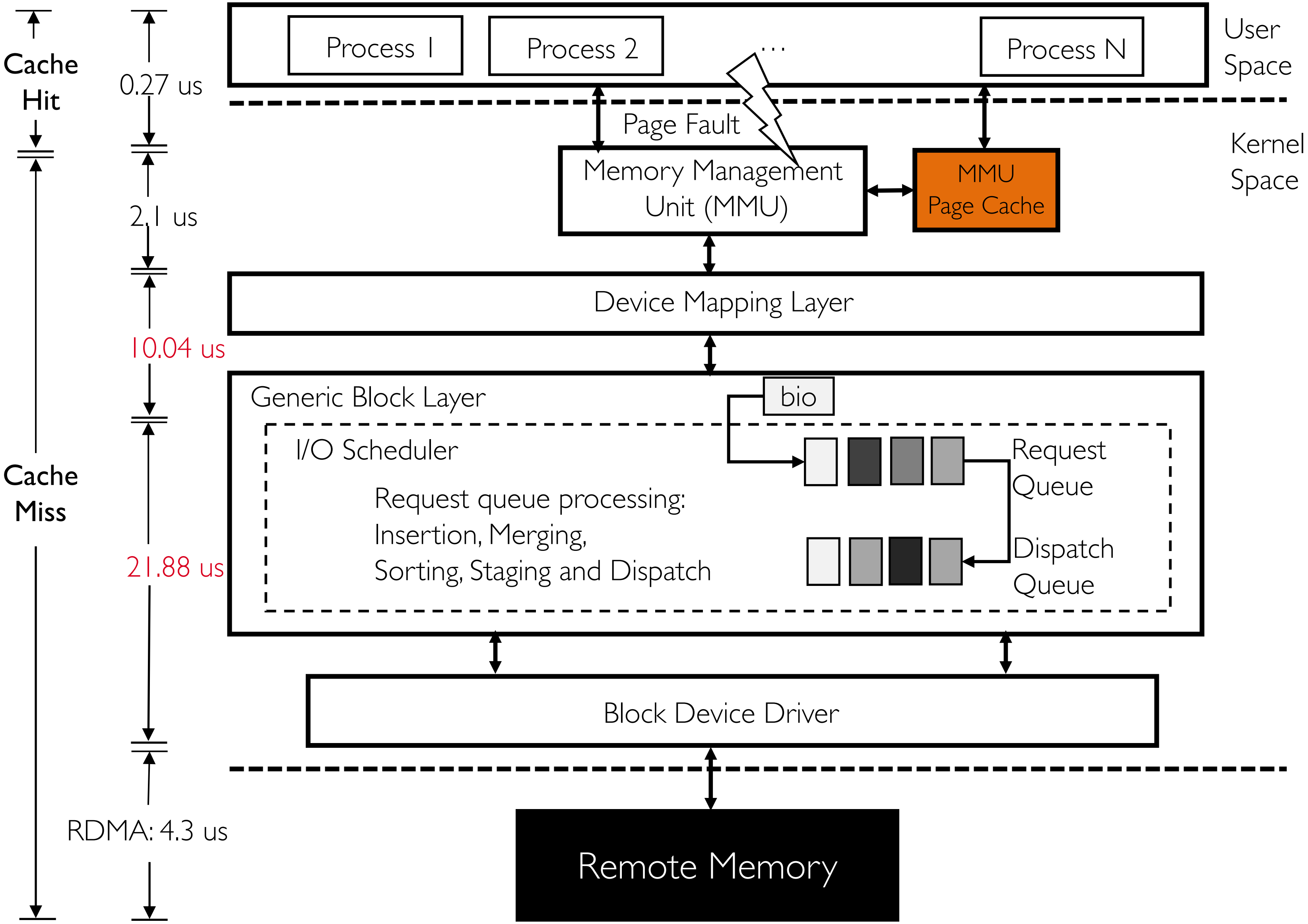
Identifies memory access patterns to prefetch pages in a

- fast,
- cache-efficient, and
- resilient manner

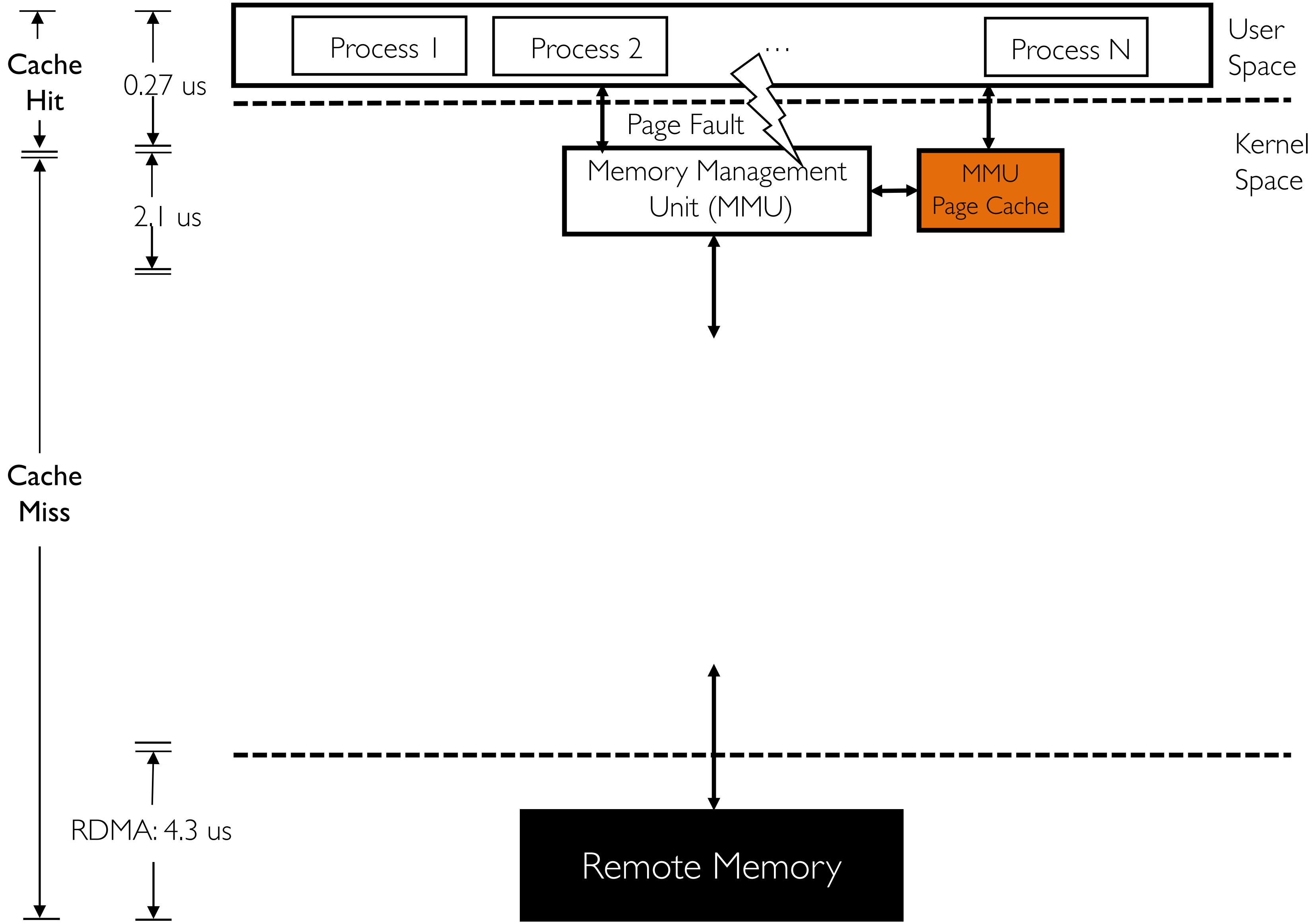
without modifying any

- applications, or
- hardware

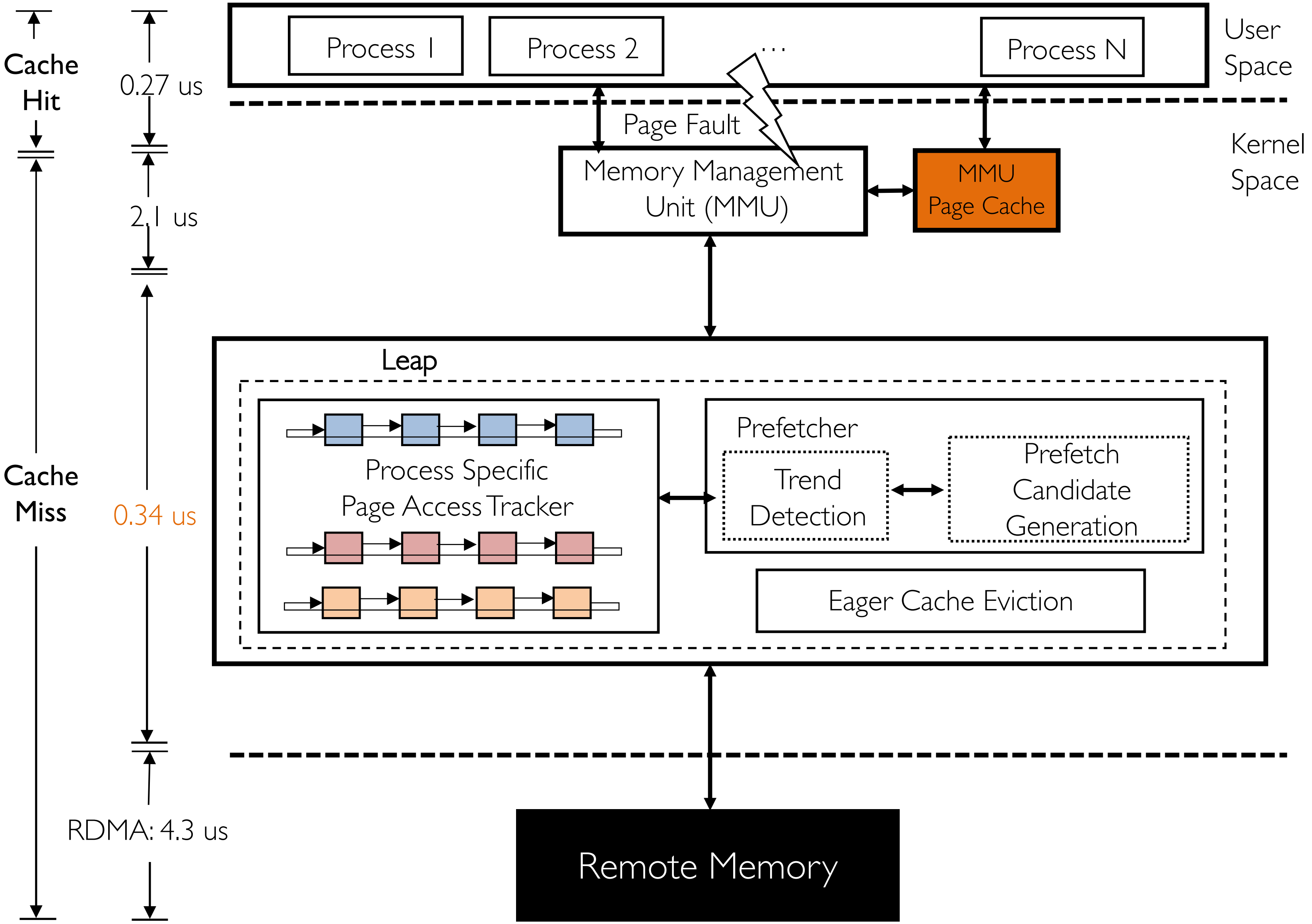
Life of a Page



Life of a Page w/ Leap



Life of a Page w/ Leap



Prefetching in Linux

Reads ahead pages sequentially

Based only on the last page access {

- *too aggressive on seq*: cache pollution
- *too conservative off seq*: brings nothing

Does not distinguish between processes

Cannot detect thread-level access irregularities

Prefetching Techniques

Approach	Low Computational Complexity	Low Memory Overhead	Unmodified Application	HW/SW Independence	Temporal Locality	Spatial Locality	Low Cache Pollution

Prefetching Techniques

Approach	Low Computational Complexity	Low Memory Overhead	Unmodified Application	HW/SW Independence	Temporal Locality	Spatial Locality	Low Cache Pollution
Next N-Line	Yes	Yes	Yes	Yes	No	Yes	No
Stride	Yes	Yes	Yes	Yes	No	Yes	No

Prefetching Techniques

Approach	Low Computational Complexity	Low Memory Overhead	Unmodified Application	HW/SW Independence	Temporal Locality	Spatial Locality	Low Cache Pollution
Next N-Line	Yes	Yes	Yes	Yes	No	Yes	No
Stride	Yes	Yes	Yes	Yes	No	Yes	No
Instruction Prefetch	No	No	No	No	Yes	Yes	No

Prefetching Techniques

Approach	Low Computational Complexity	Low Memory Overhead	Unmodified Application	HW/SW Independence	Temporal Locality	Spatial Locality	Low Cache Pollution
Next N-Line	Yes	Yes	Yes	Yes	No	Yes	No
Stride	Yes	Yes	Yes	Yes	No	Yes	No
Instruction Prefetch	No	No	No	No	Yes	Yes	No
Linux Read-Ahead	Yes	Yes	Yes	Yes	Yes	Yes	No

Prefetching Techniques

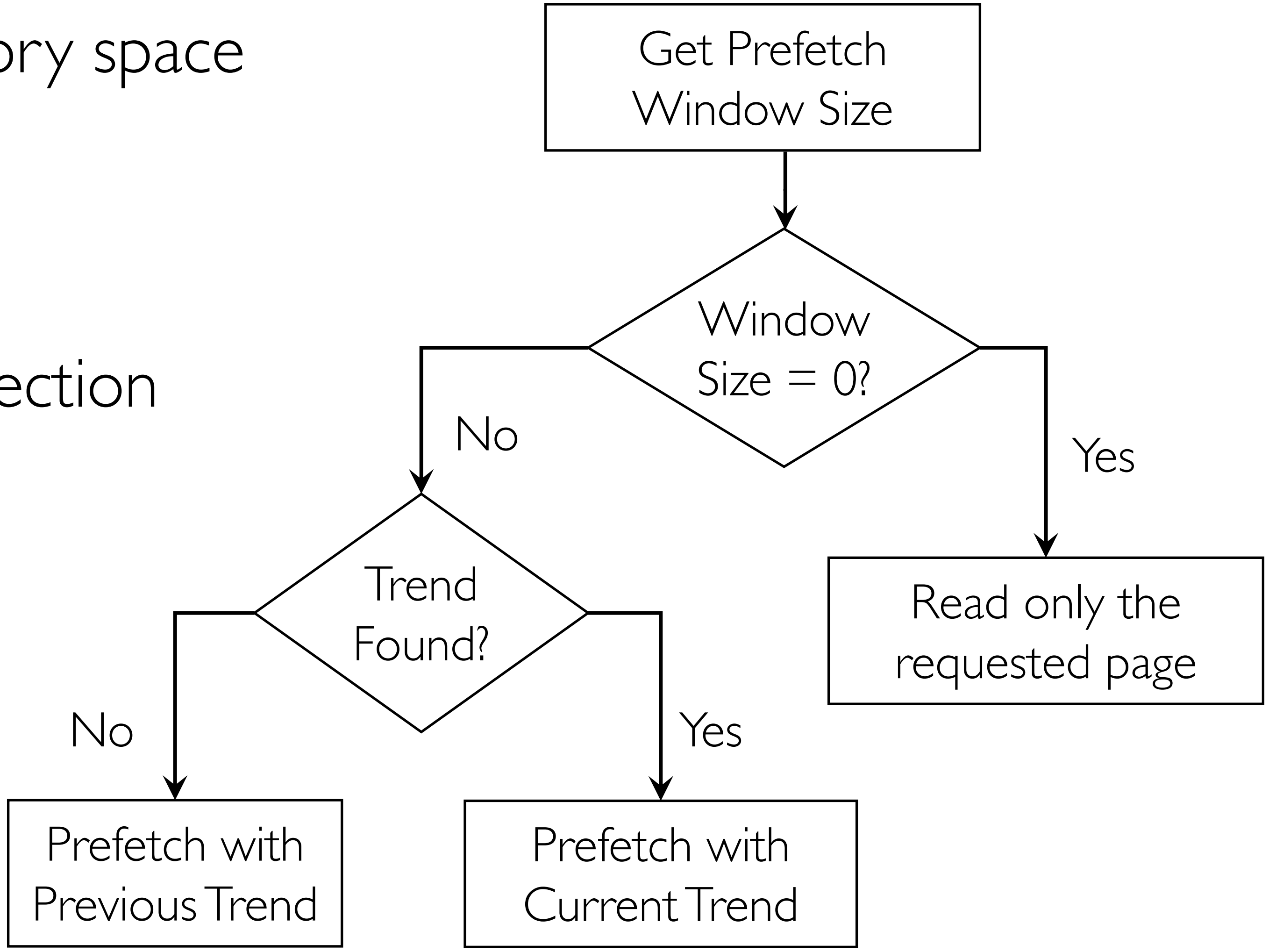
Approach	Low Computational Complexity	Low Memory Overhead	Unmodified Application	HW/SW Independence	Temporal Locality	Spatial Locality	Low Cache Pollution
Next N-Line	Yes	Yes	Yes	Yes	No	Yes	No
Stride	Yes	Yes	Yes	Yes	No	Yes	No
Instruction Prefetch	No	No	No	No	Yes	Yes	No
Linux Read-Ahead	Yes	Yes	Yes	Yes	Yes	Yes	No
Leap	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

Leap Prefetcher

Linear-time and constant memory space

Two main components:

- Trend detection
- Prefetch window size detection

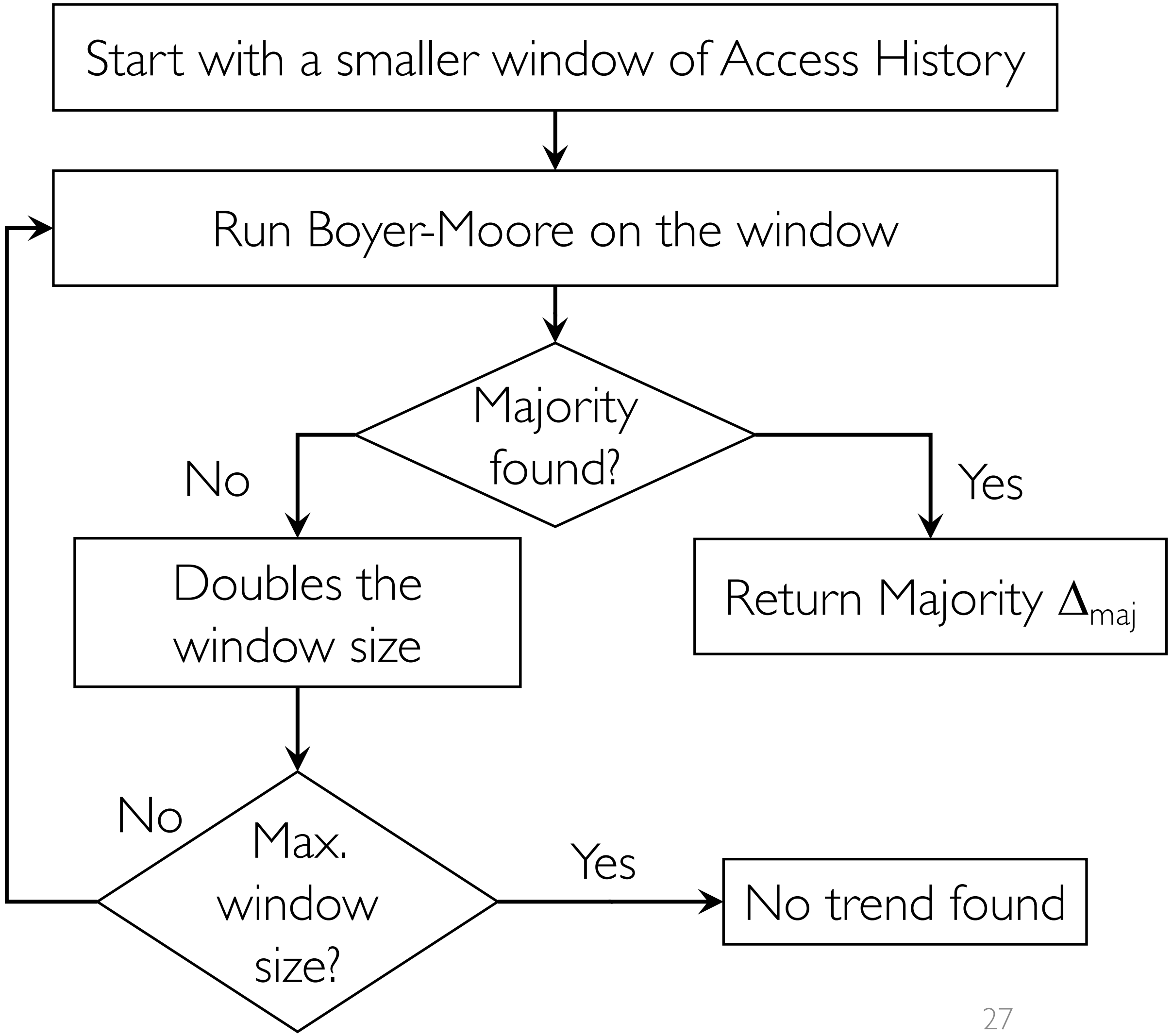

Get Prefetch Window Size
Window Size = 0?
No
Yes
Read only the requested page
Trend Found?
No
Yes
Prefetch with Previous Trend
Prefetch with Current Trend

Trend Detection

Flexible to short term irregularity

Identifies the majority element in access history

Regular trends can be found within recent accesses


Start with a smaller window of Access History

Run Boyer-Moore on the window

Majority found?

No

Yes

Doubles the window size

Return Majority Δ_{maj}

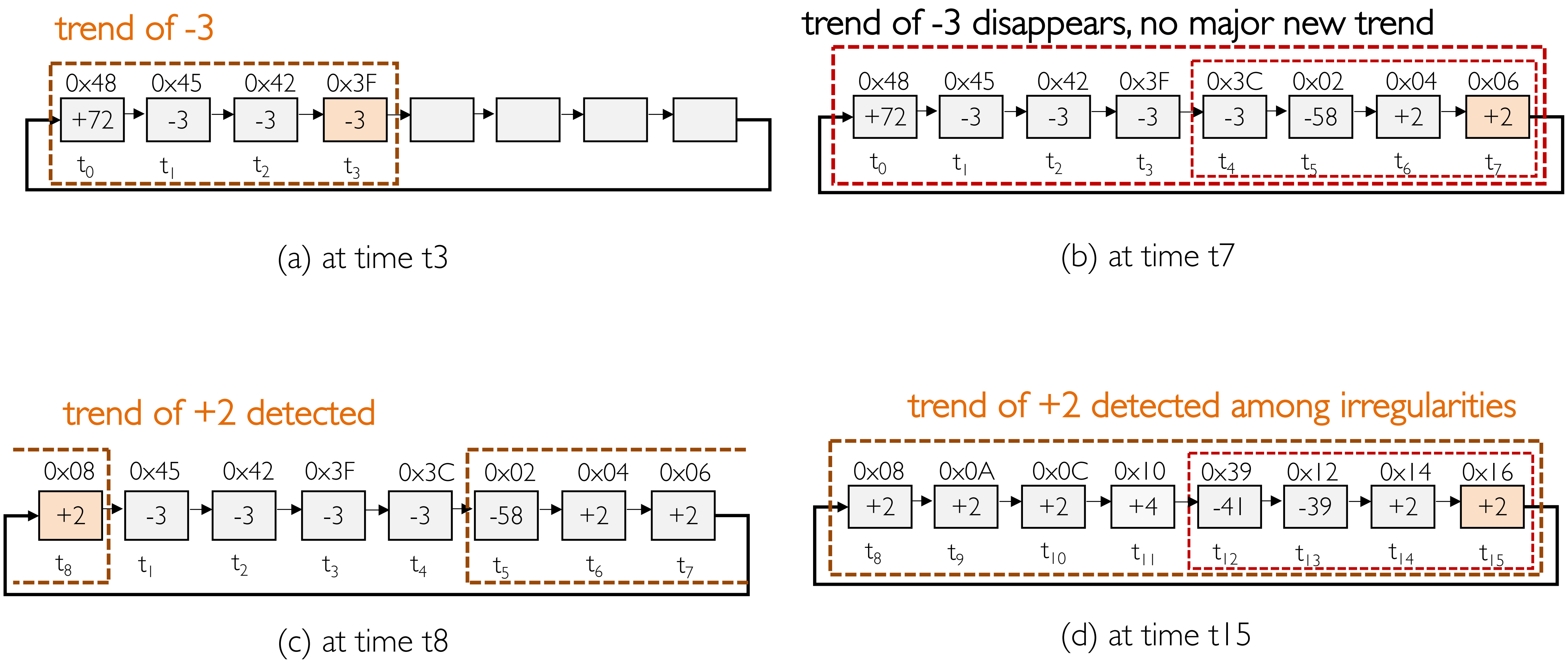
Max. window size?

No

Yes

No trend found

Trend Detection Example



trend of -3

(a) at time t3

trend of -3 disappears, no major new trend

(b) at time t7

trend of +2 detected

(c) at time t8

trend of +2 detected among irregularities

(d) at time t15

Prefetch Window Size Detection

Cache hit indicates prefetch utilization

High cache hit: increase prefetch window aggressively

No cache hit

- *trend availability: increase prefetch window gradually*
- *no trend: decrease prefetch window gradually*

Gradual slow down helps during sudden changes

Evaluation

Deploy and evaluate over 56 Gbps InfiniBand network

Memory Disaggregation Frameworks

- *Disaggregated VMM:* **Infiniswap**
- *Disaggregated VFS:* **Remote Regions**

powergraph

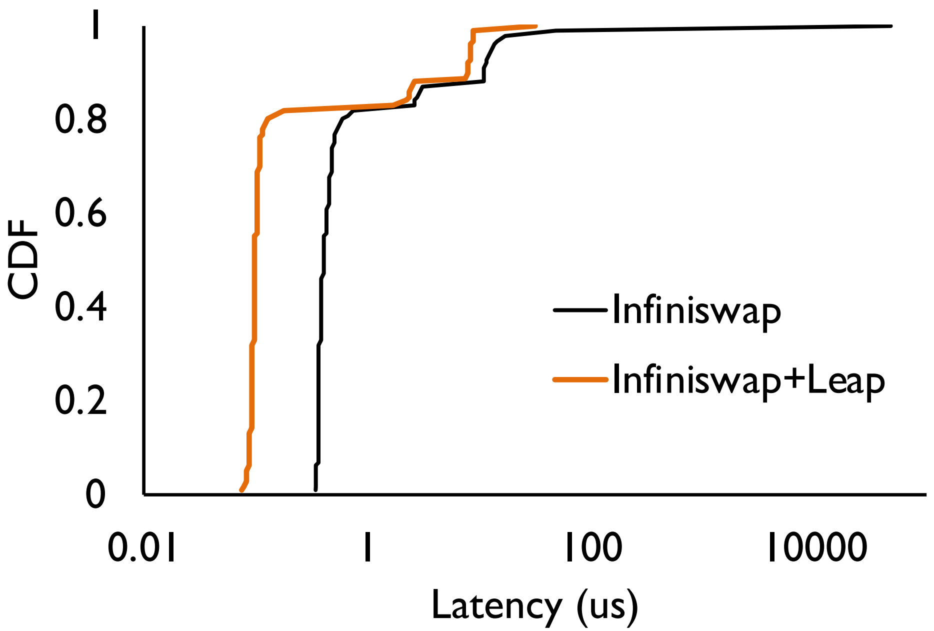
NumPy

VOLTDB

MEMCACHED

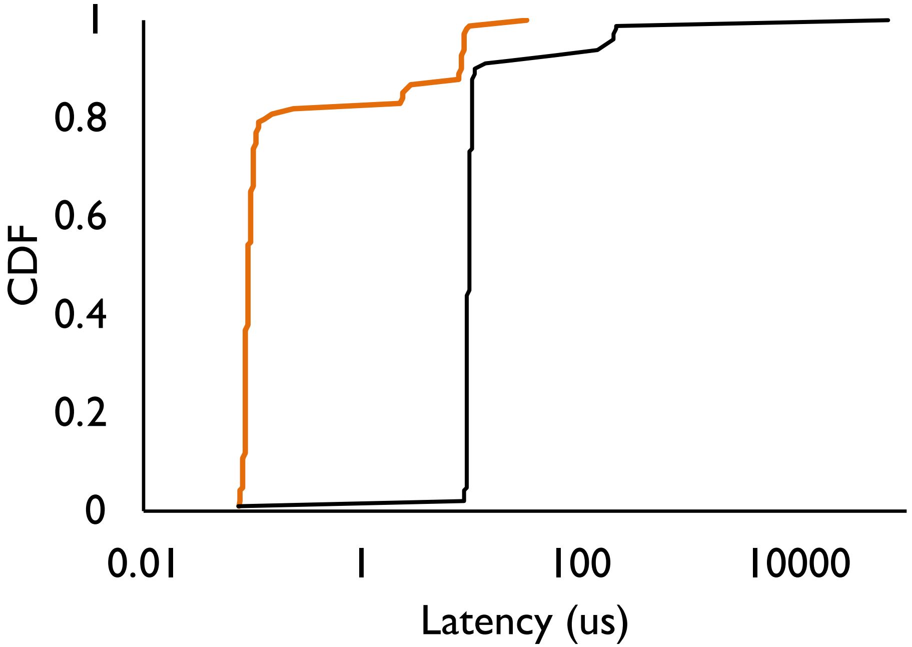
Lowers Remote Page Access Latency by…

Sequential Access



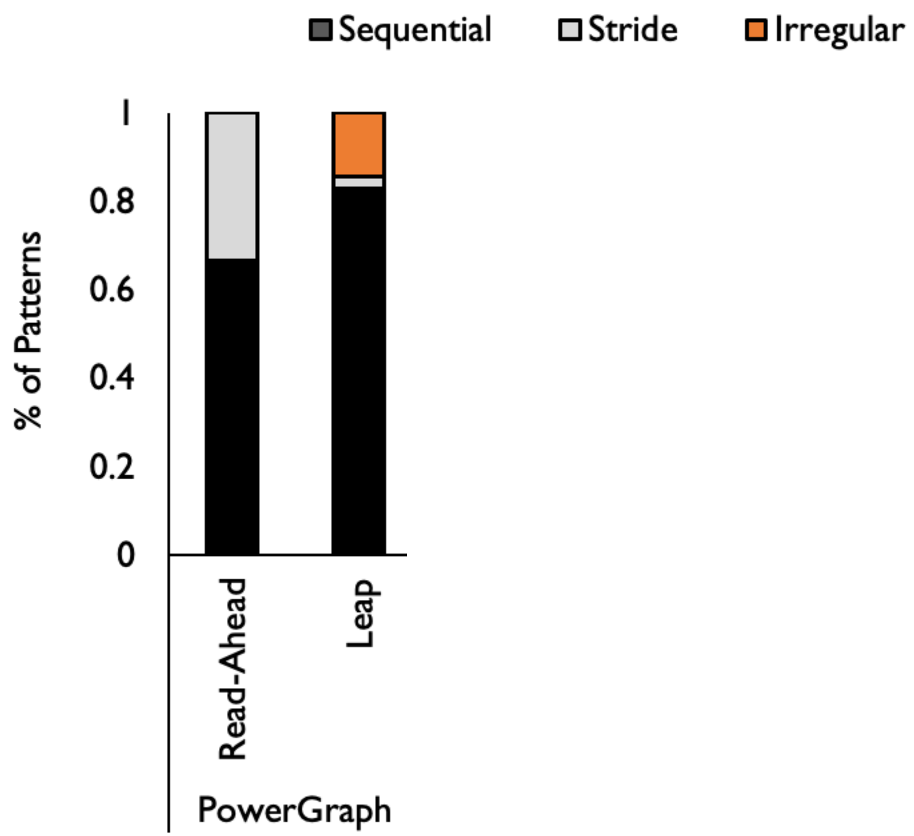
4X

Stride Access



104X

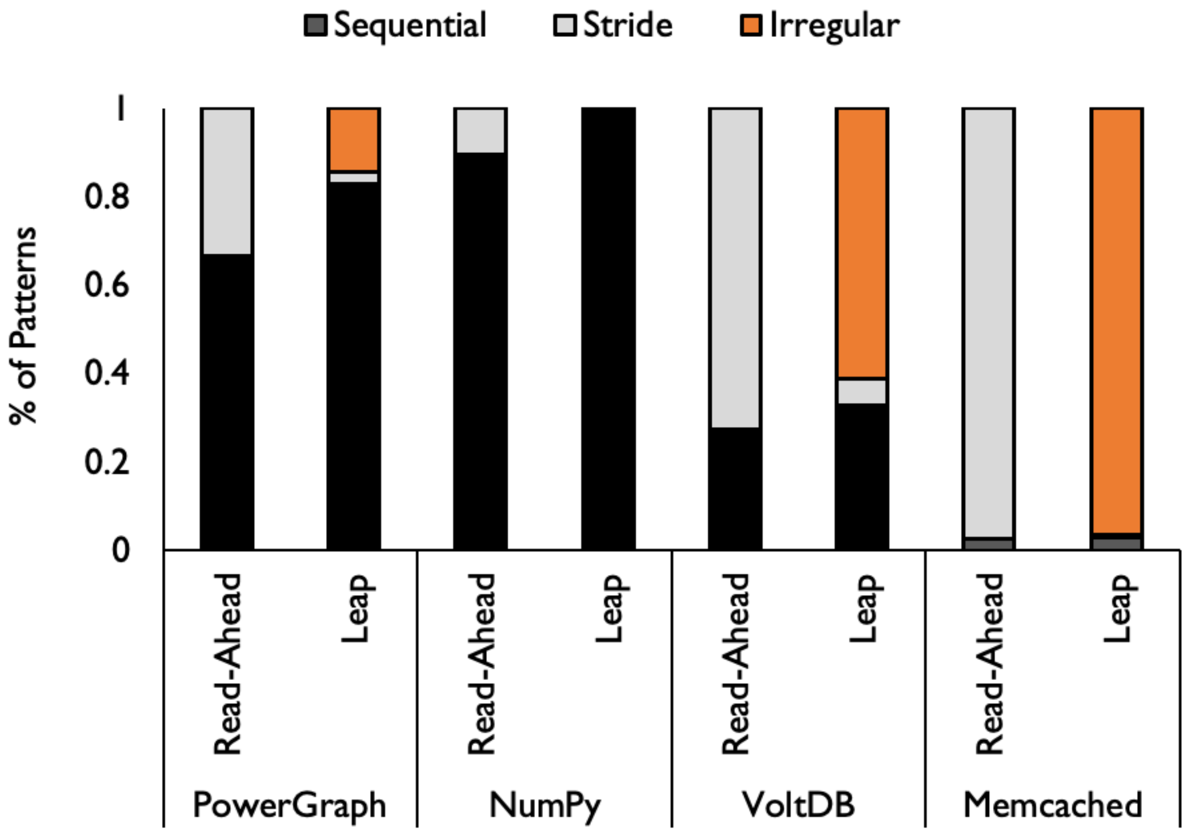
Efficient Pattern Detection



Detects **29.70%** more sequential accesses

Detects most of the irregularity

Efficient Pattern Detection



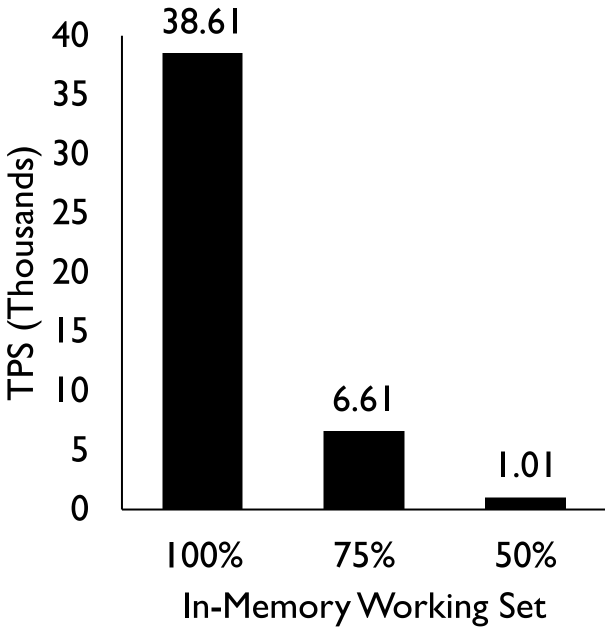
Detects **29.70%** more sequential accesses

Detects most of the irregularity

During irregularities, doing nothing helps the most

Perform Great Even After Memory Runs Out

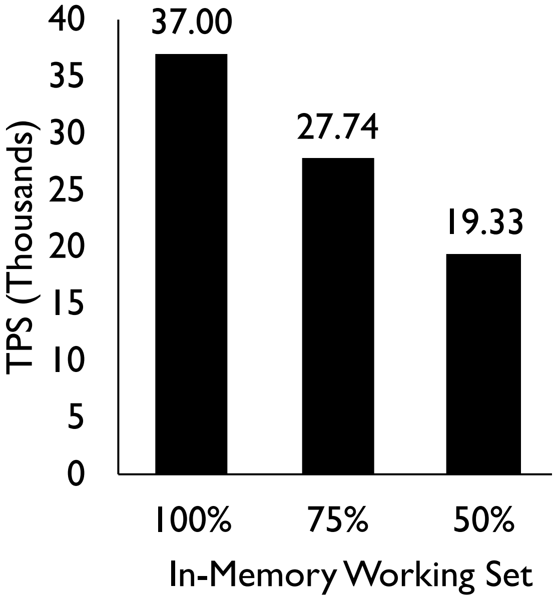
Disk



TPC-C on VoltDB

38X

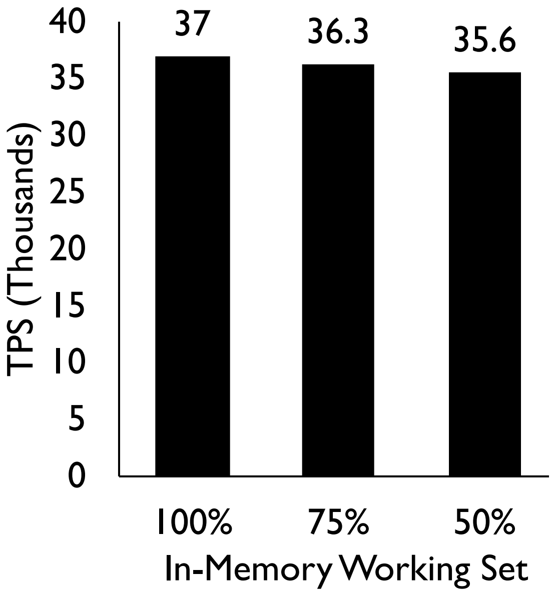
Infiniswap



TPC-C on VoltDB

2X

Infiniswap + Leap

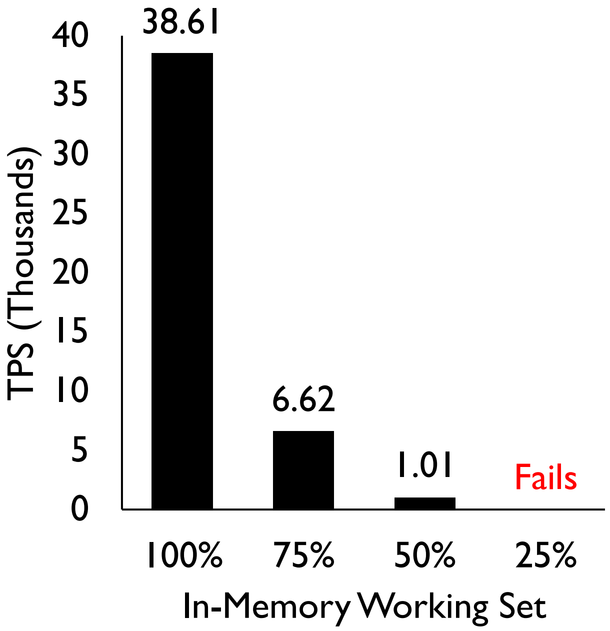


TPC-C on VoltDB

≈1X

Perform Great Even After Memory Runs Out

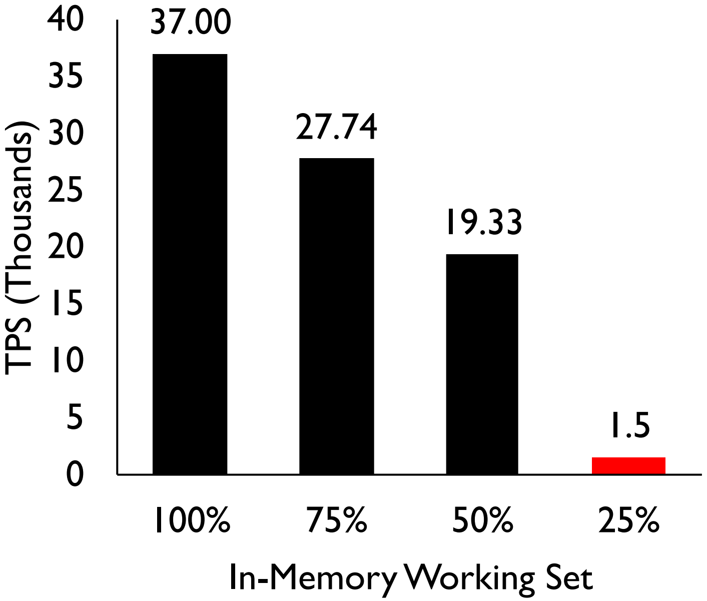
Disk



TPC-C on VoltDB

Fails

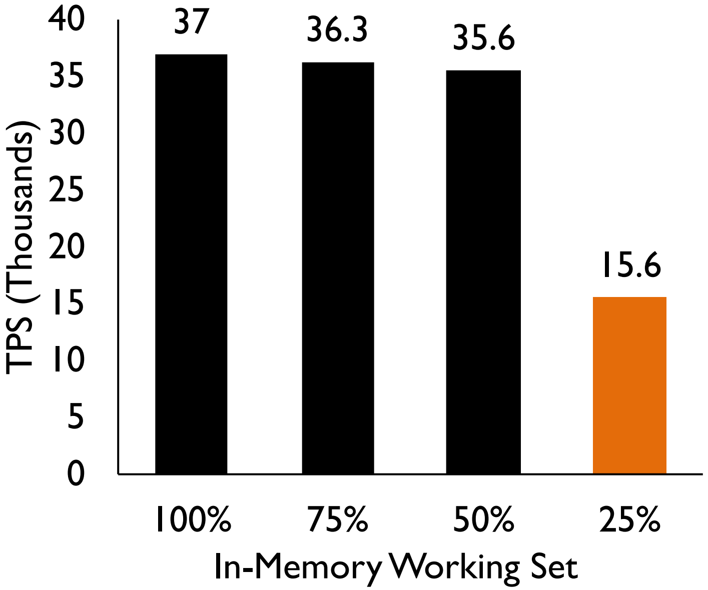
Infiniswap



TPC-C on VoltDB

24X

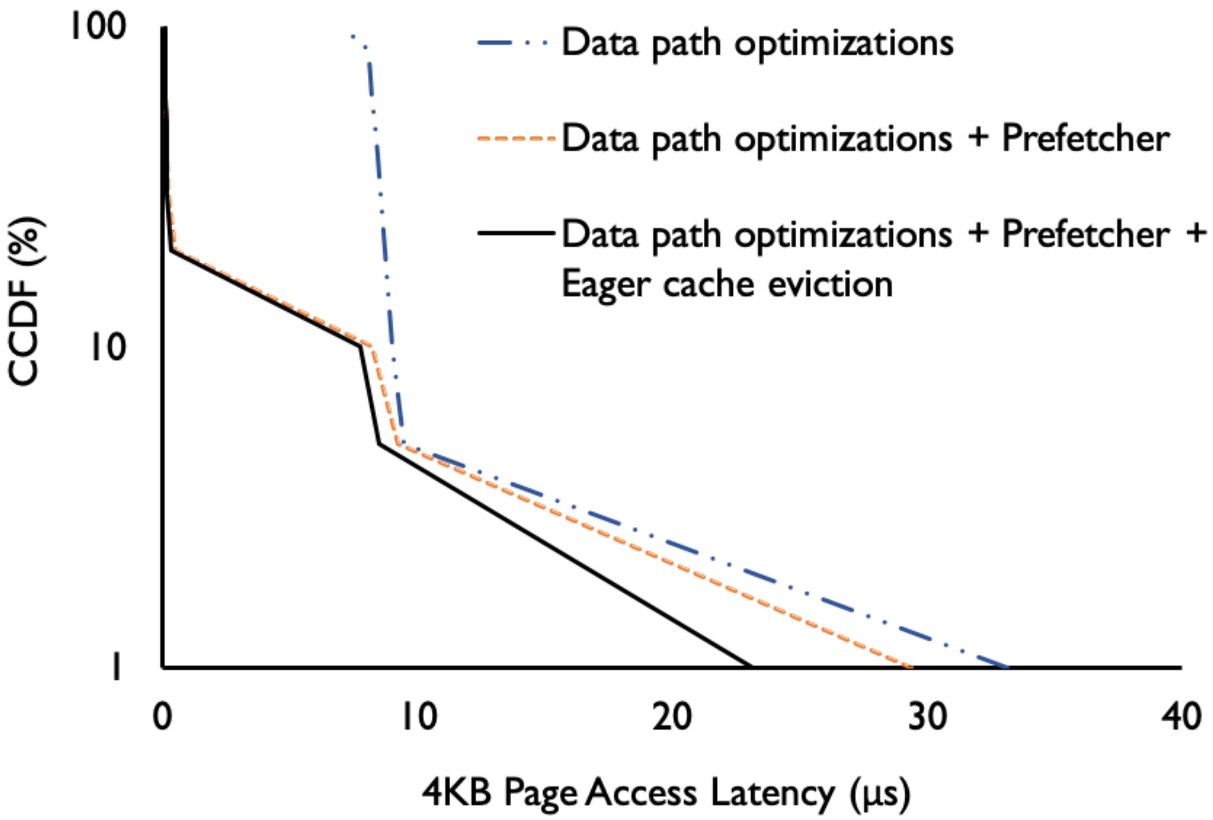
Infiniswap + Leap



TPC-C on VoltDB

2.4X

Benefit Breakdown of Leap's Components



Data path optimizations: single-µs latency till 95th percentile

Prefetcher: sub-µs latency till 85th percentile

Eager cache eviction: improves the 99th percentile latency by 22%

Future Work

1. Thread-specific prefetching for multiple concurrent streams
   - memory is managed at the process level
   - this requires significant changes in virtual memory subsystem

2. Optimized remote I/O interface
   - load balancing,
   - fault-tolerance,
   - data locality, and
   - application-specific isolation in remote memory

Leap

Lightweight and efficient data path for remote memory

source code available at https://github.com/SymbioticLab/leap

Online prefetcher with a leaner data path and eager cache eviction policy to improve

- cache hit,
- remote I/O latency, and
- application-level performance

without modifying any

- application, or
- hardware

Thank You!

source code available at https://github.com/SymbioticLab/leap