Rethinking File Mapping in Persistent Memory

Ian Neal[1], Gefei Zuo[1], Eric Shiple[1], Tanvir Ahmed Khan[1],
Youngjin Kwon[3], Simon Peter[2], Baris Kasikci[1]

[1] UNIVERSITY OF MICHIGAN

[2] The University of Texas at Austin Computer Science

[3] KAIST School of Computing

EFES

Good News! PM has arrived!

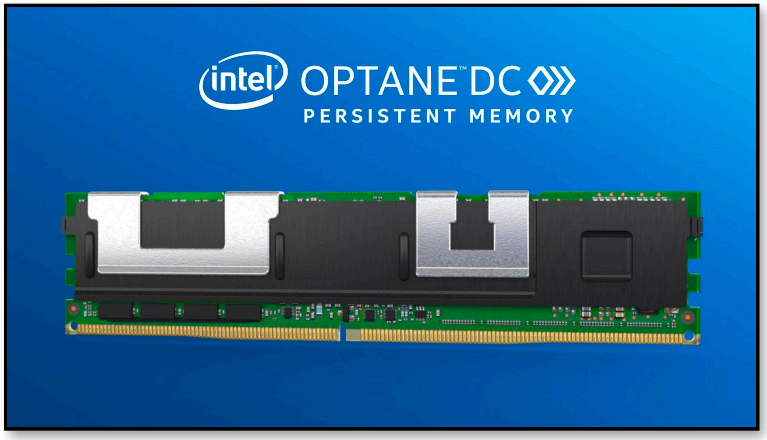


Image Source: https://www.intel.com/content/www/us/en/architecture-and-technology/optane-dc-persistent-memory.html

Good News! PM has arrived!

- Great, fast new storage technology called persistent main memory (PM)
  - AKA non-volatile memory (NVM)

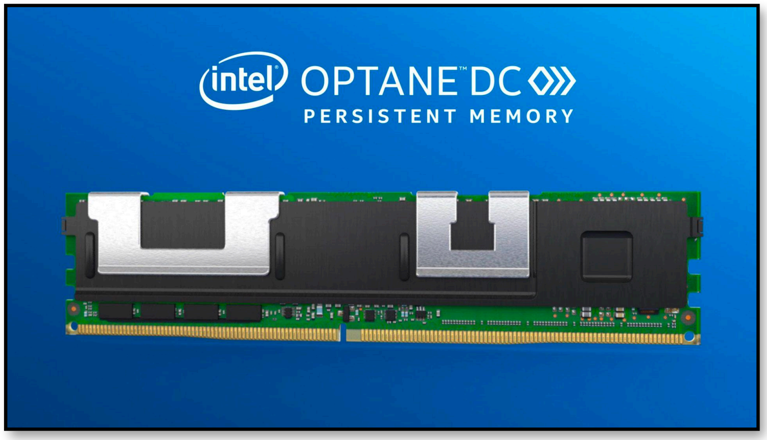


Image Source: https://www.intel.com/content/www/us/en/architecture-and-technology/optane-dc-persistent-memory.html

Good News! PM has arrived!

- Great, fast new storage technology called persistent main memory (PM)
  - AKA non-volatile memory (NVM)
- 30−40x faster than SSDs

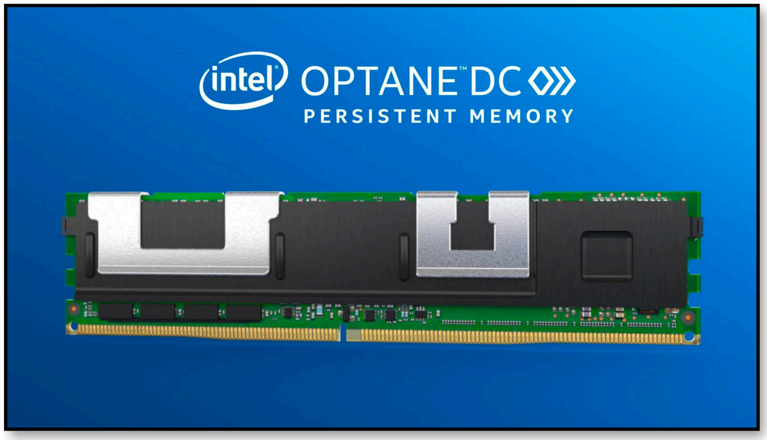


Image Source: https://www.intel.com/content/www/us/en/architecture-and-technology/optane-dc-persistent-memory.html

Good News! PM has arrived!

- Great, fast new storage technology called persistent main memory (PM)
  - AKA non-volatile memory (NVM)
- 30−40x faster than SSDs
- File system IO performance has not been able to keep up with PM performance. Why?

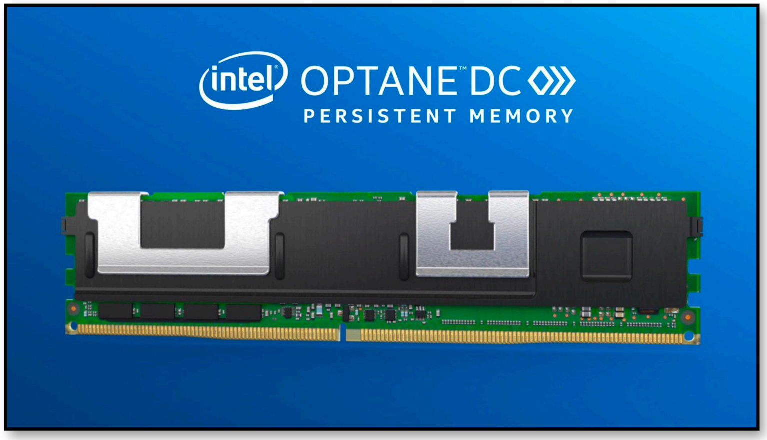


Image Source: https://www.intel.com/content/www/us/en/architecture-and-technology/optane-dc-persistent-memory.html

Good News! PM has arrived!

- Great, fast new storage technology called persistent main memory (PM)
  - AKA non-volatile memory (NVM)
- 30−40x faster than SSDs
- File system IO performance has not been able to keep up with PM performance. Why?
- ***No rigorous analysis of IO path performance!***

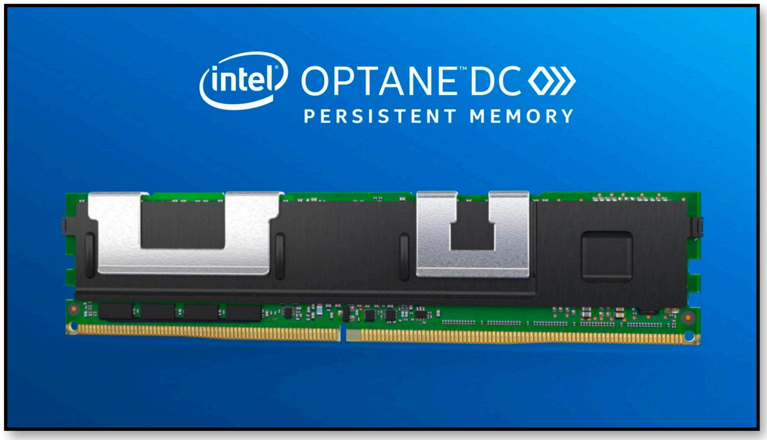
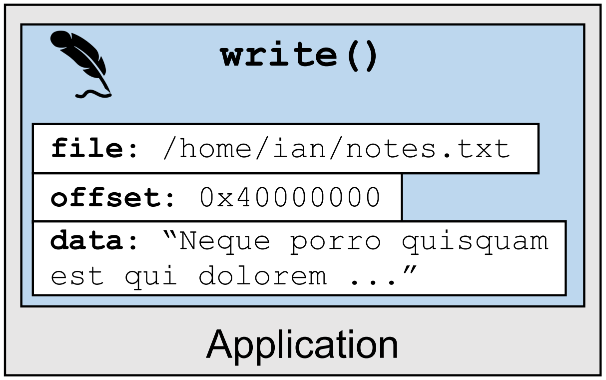


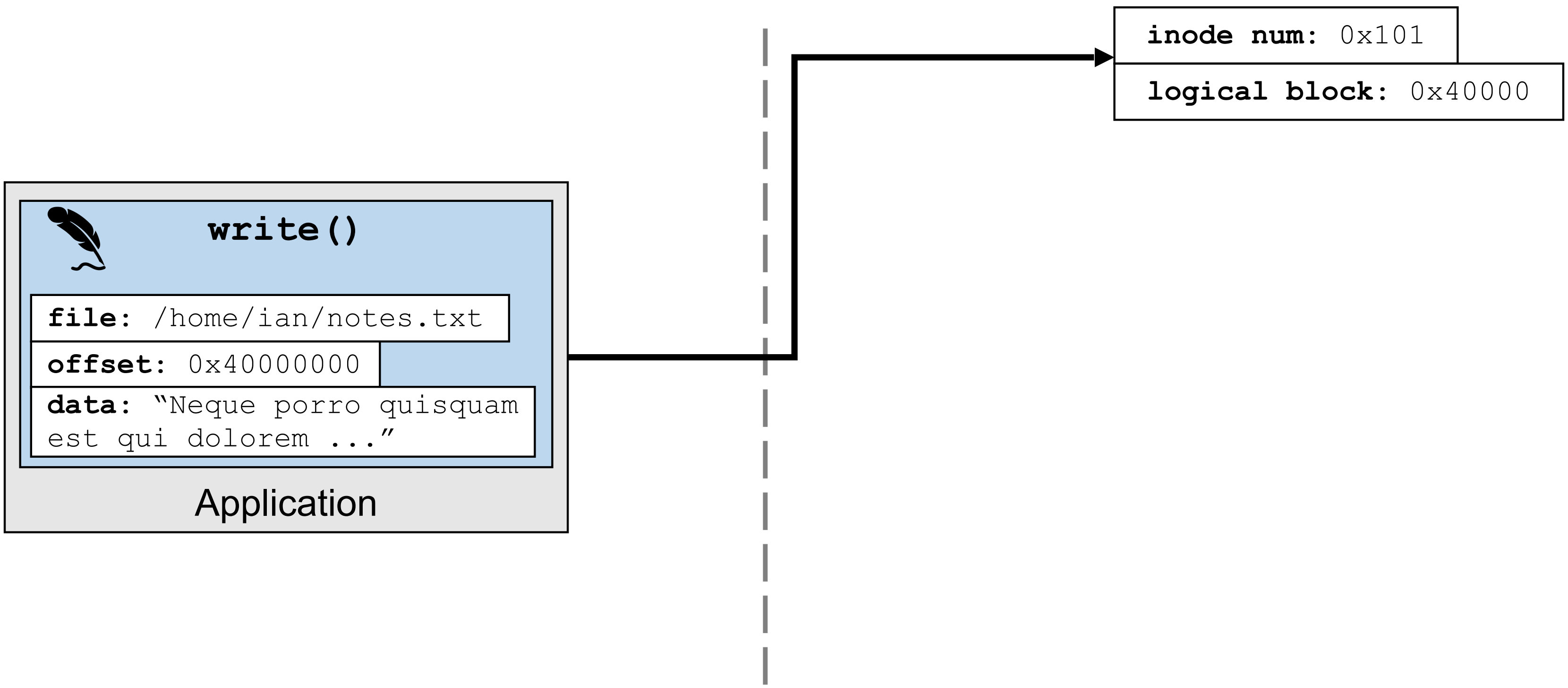
Image Source: https://www.intel.com/content/www/us/en/architecture-and-technology/optane-dc-persistent-memory.html

File IO Before PM

File IO Before PM

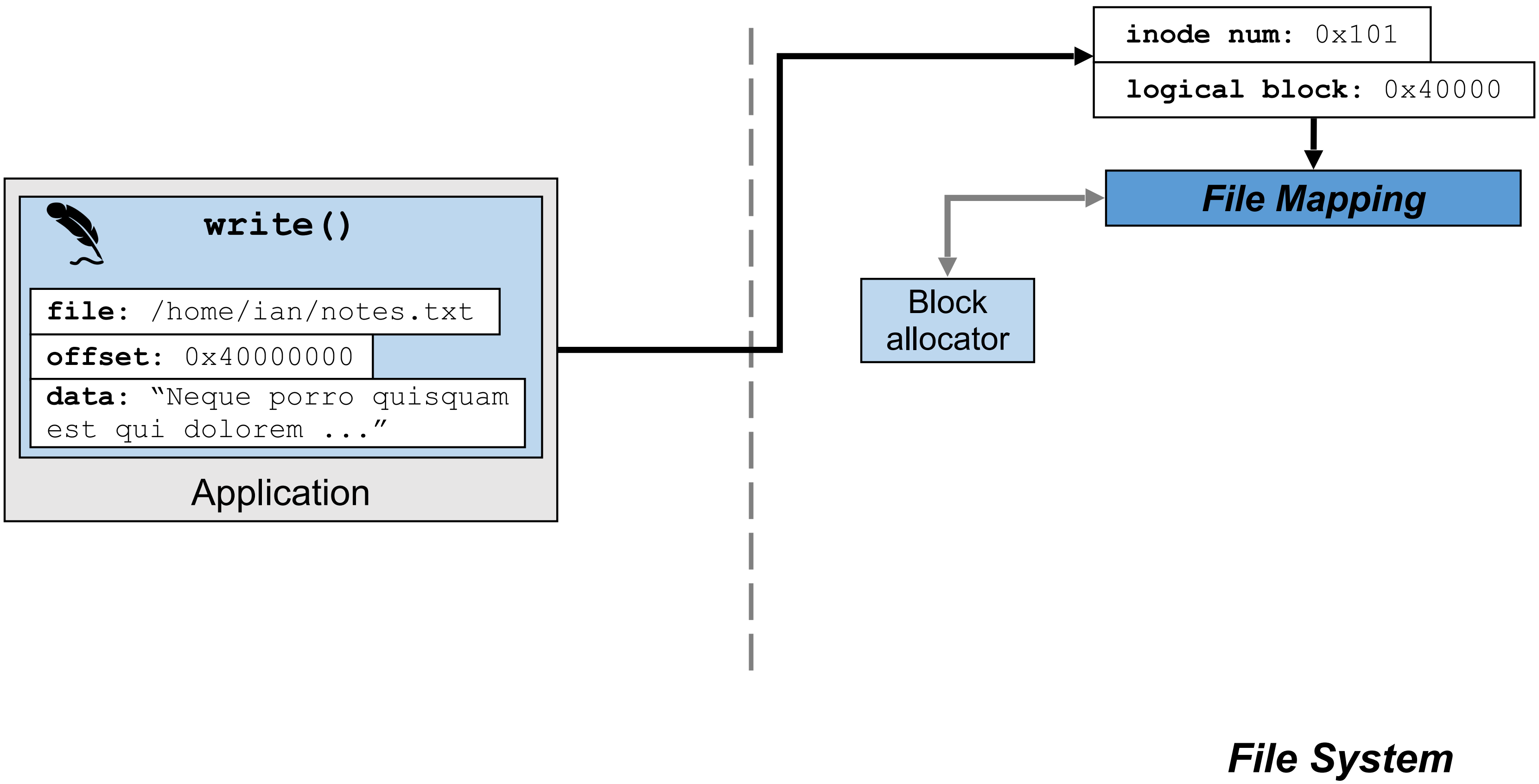


File IO Before PM

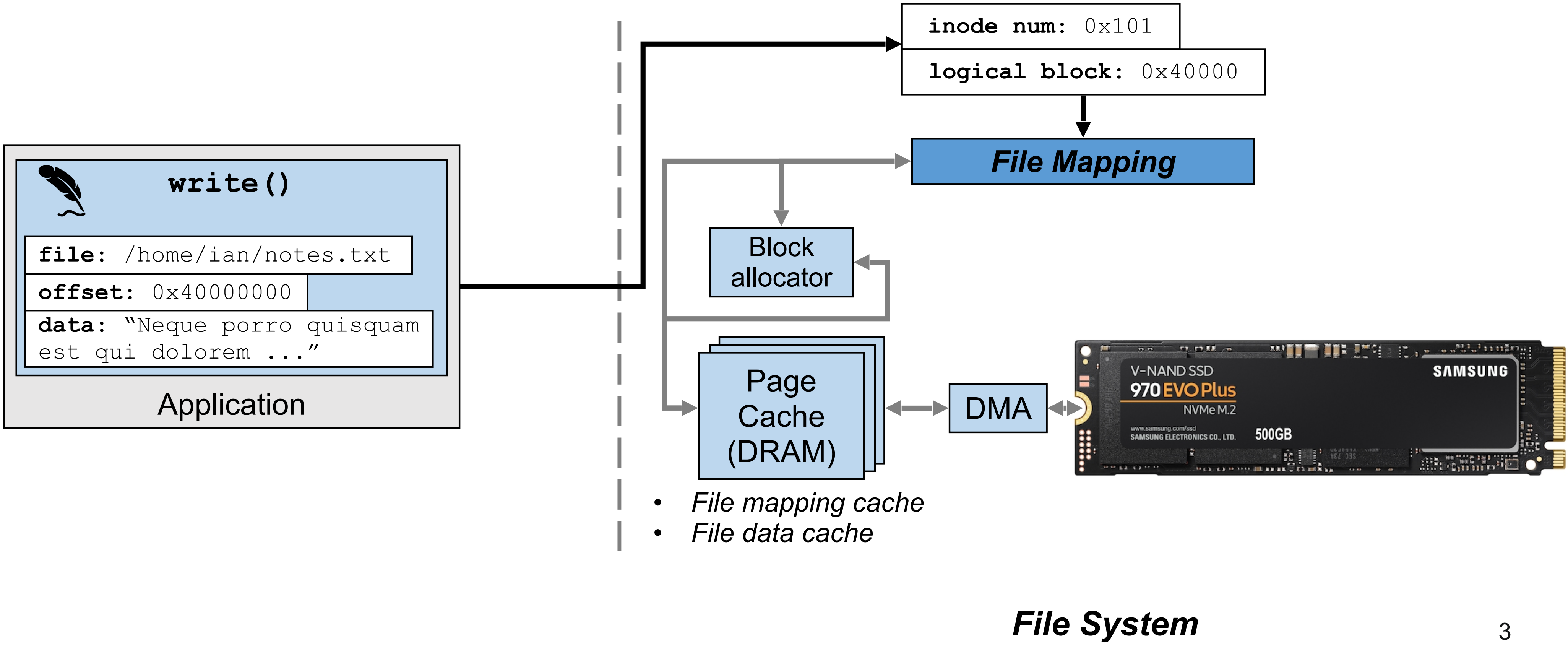


File System

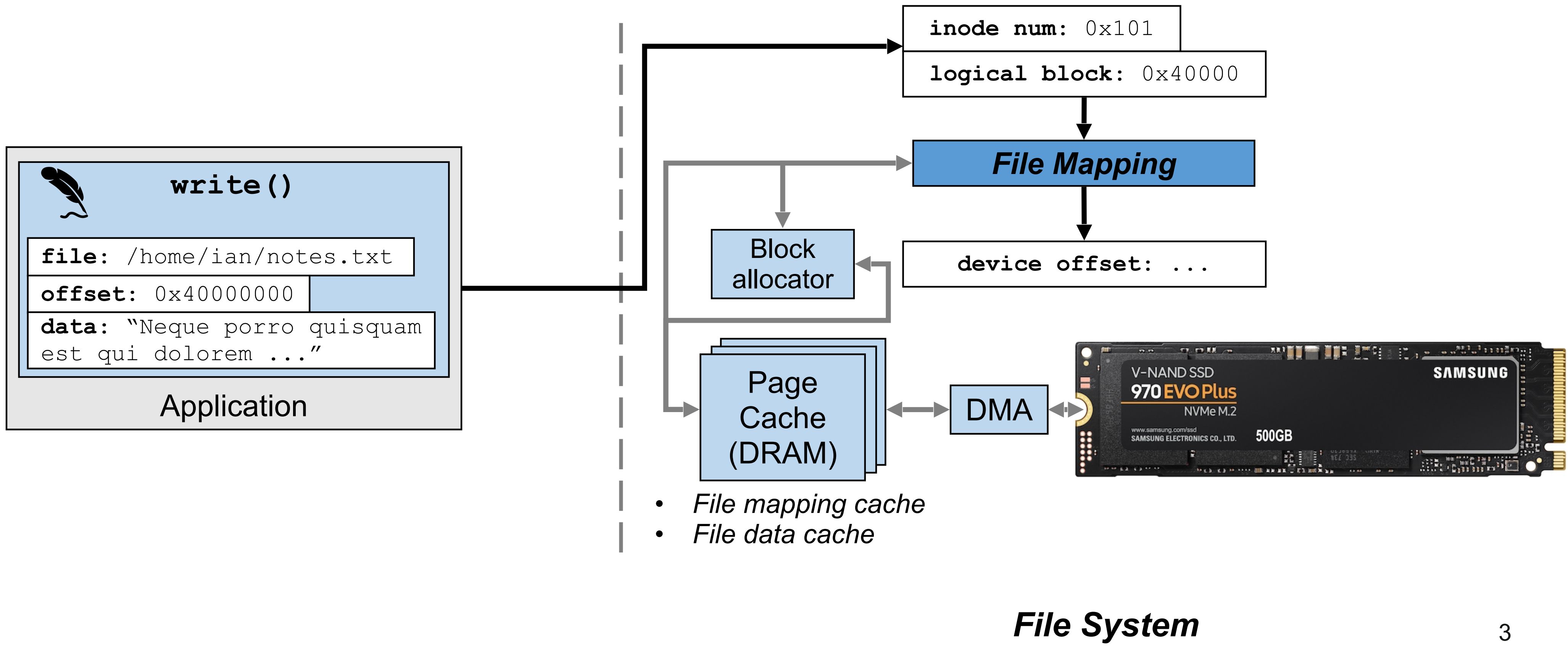
File IO Before PM



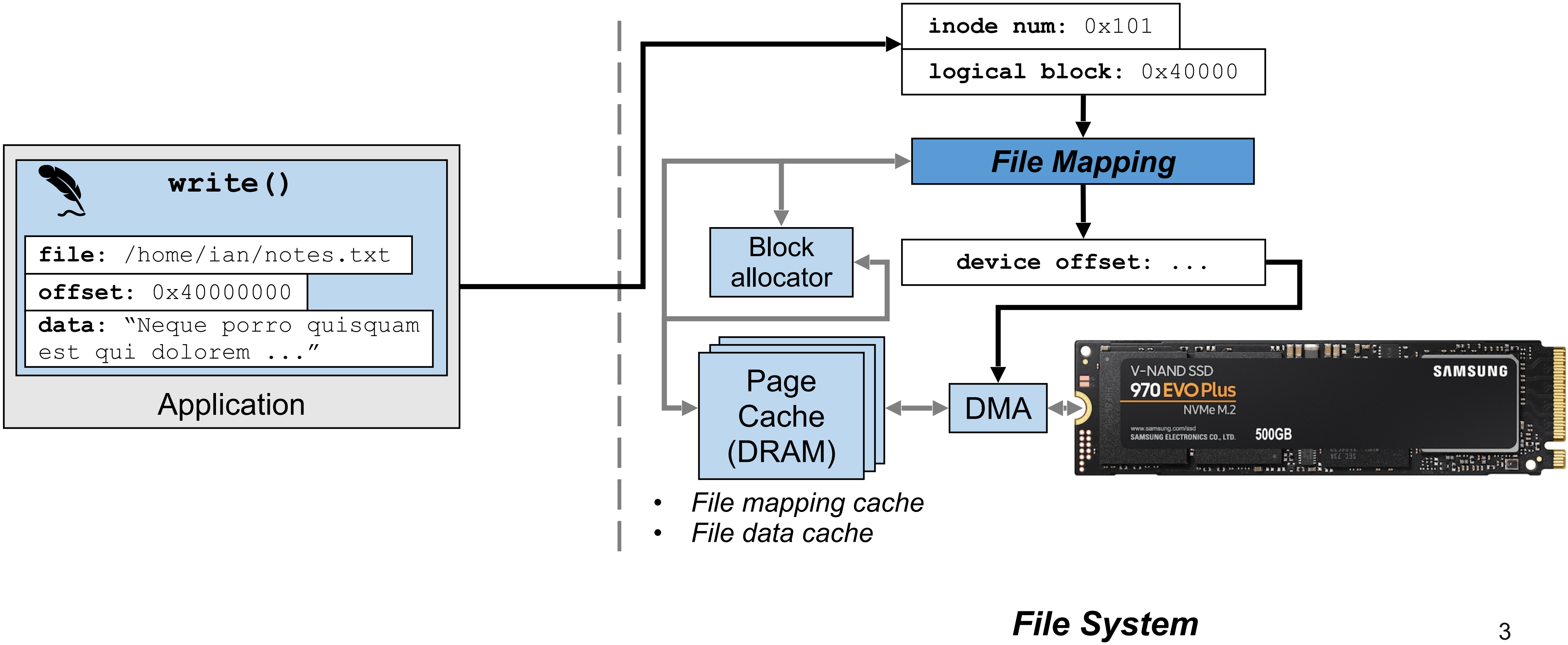
File IO Before PM



File IO Before PM

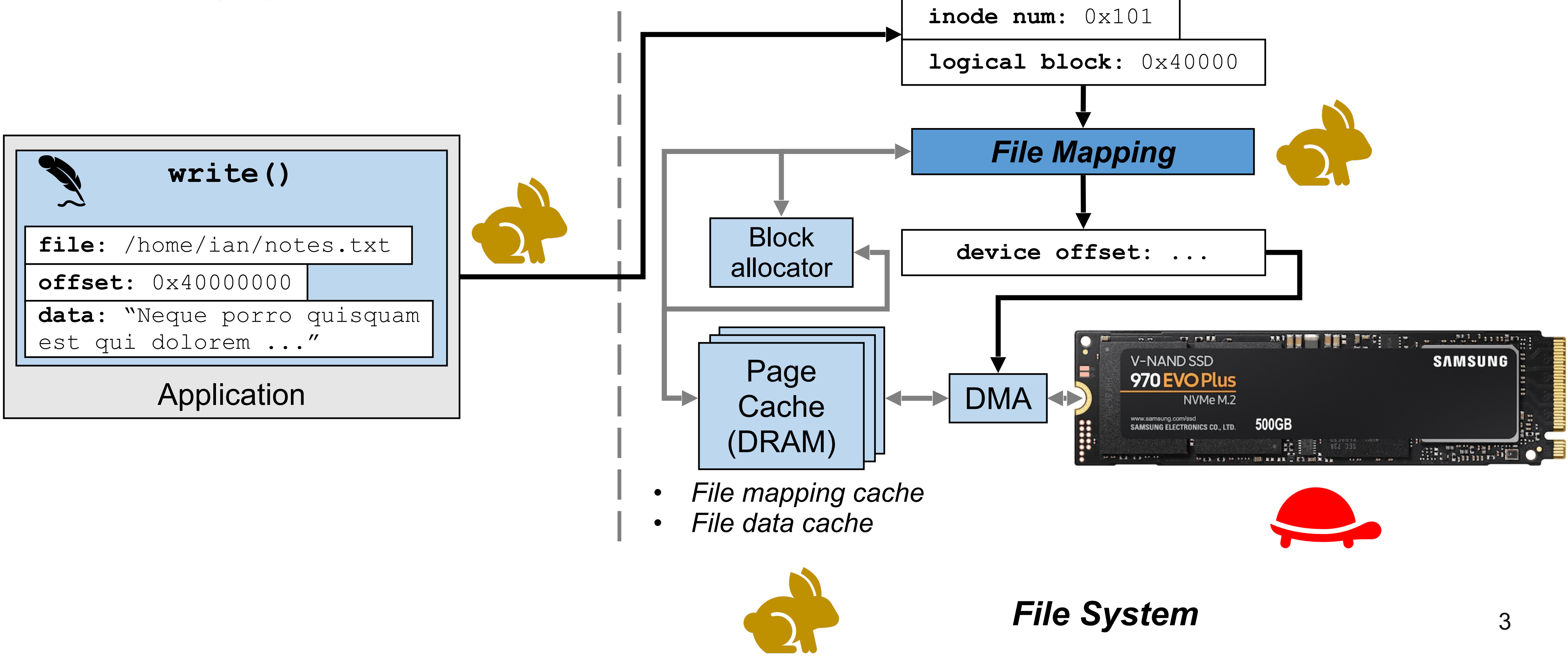


File IO Before PM

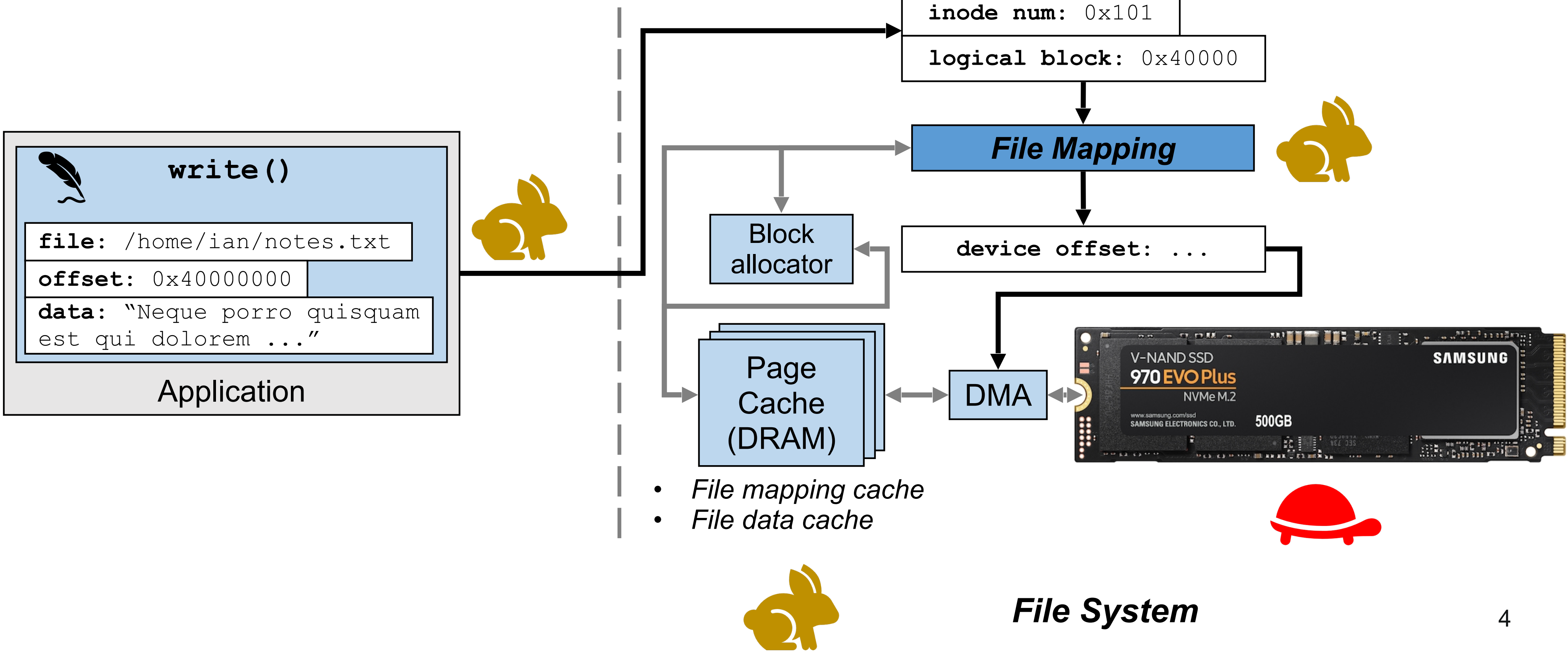


File IO Before PM

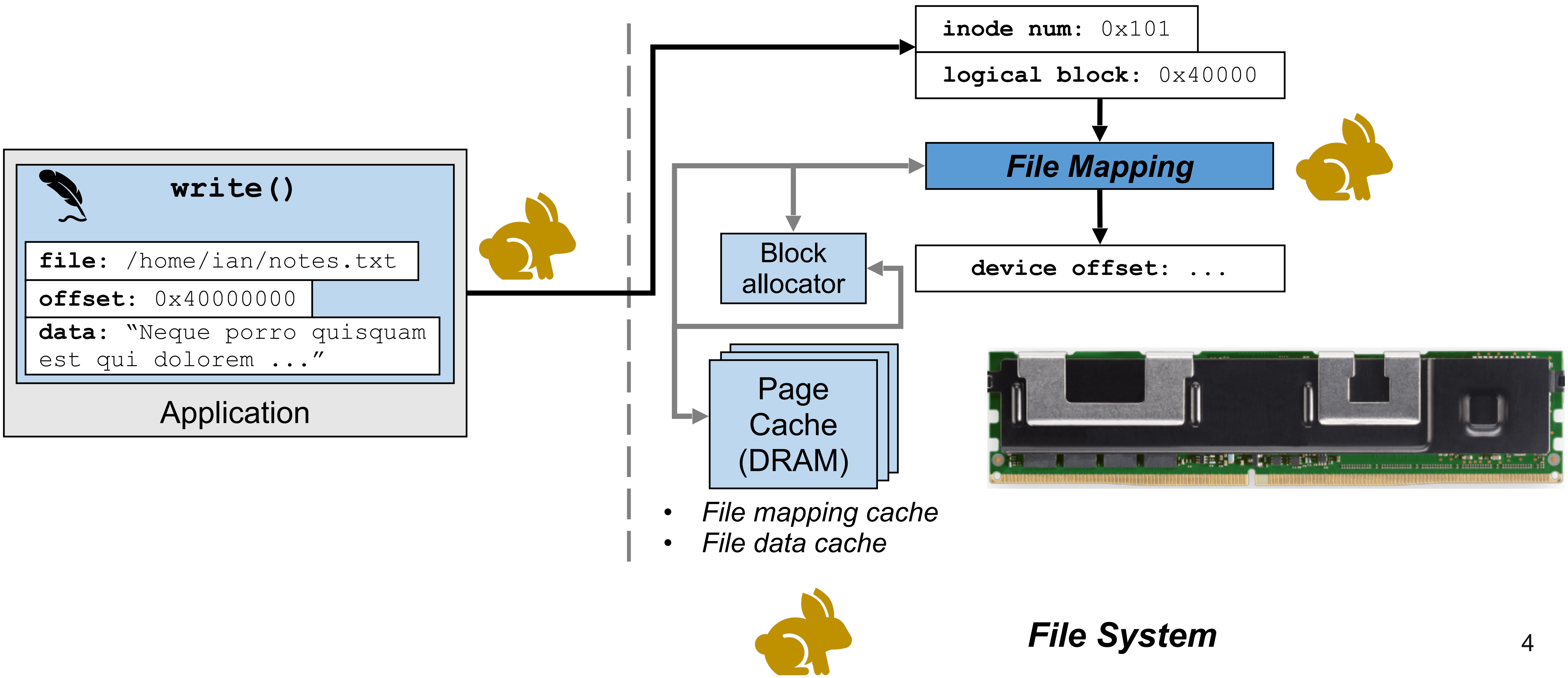
- When devices were slow, FS software overheads were negligible



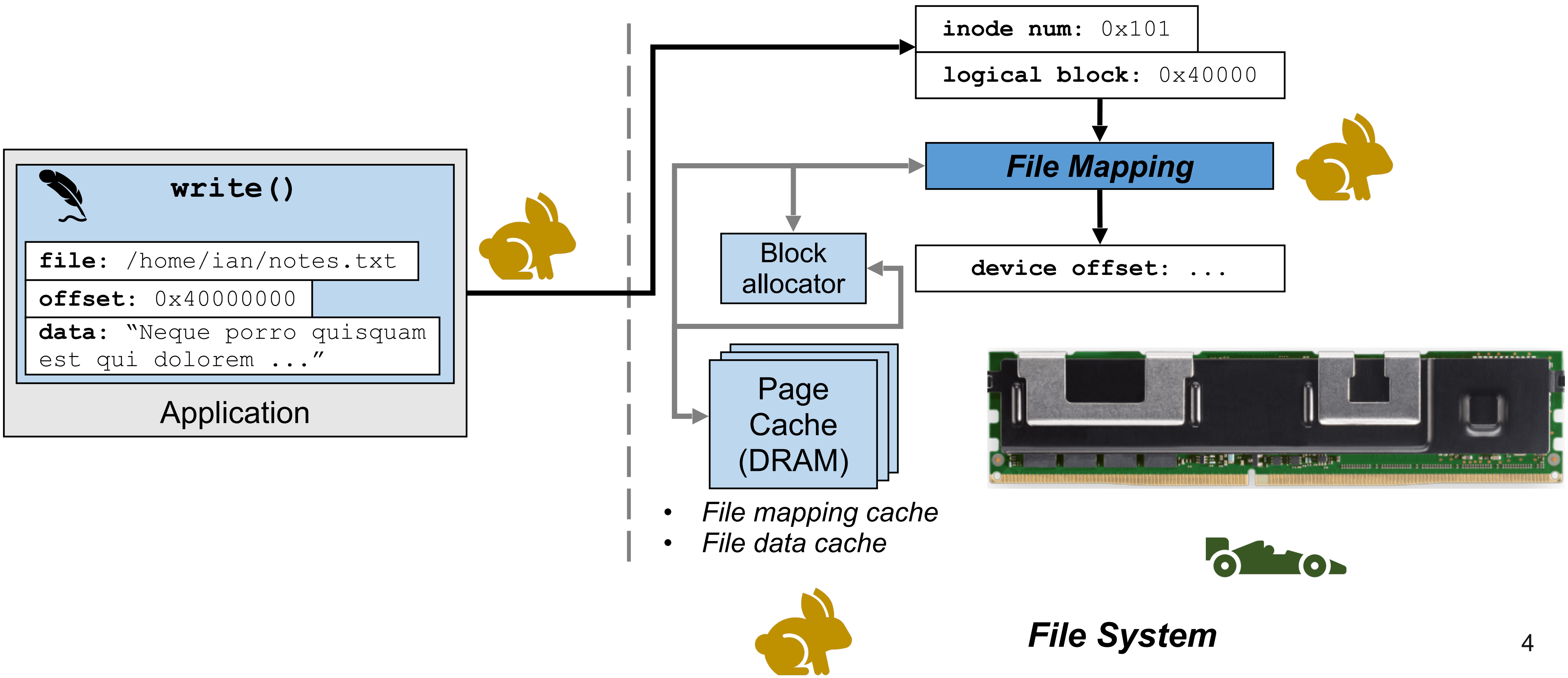
File IO After PM



File IO After PM

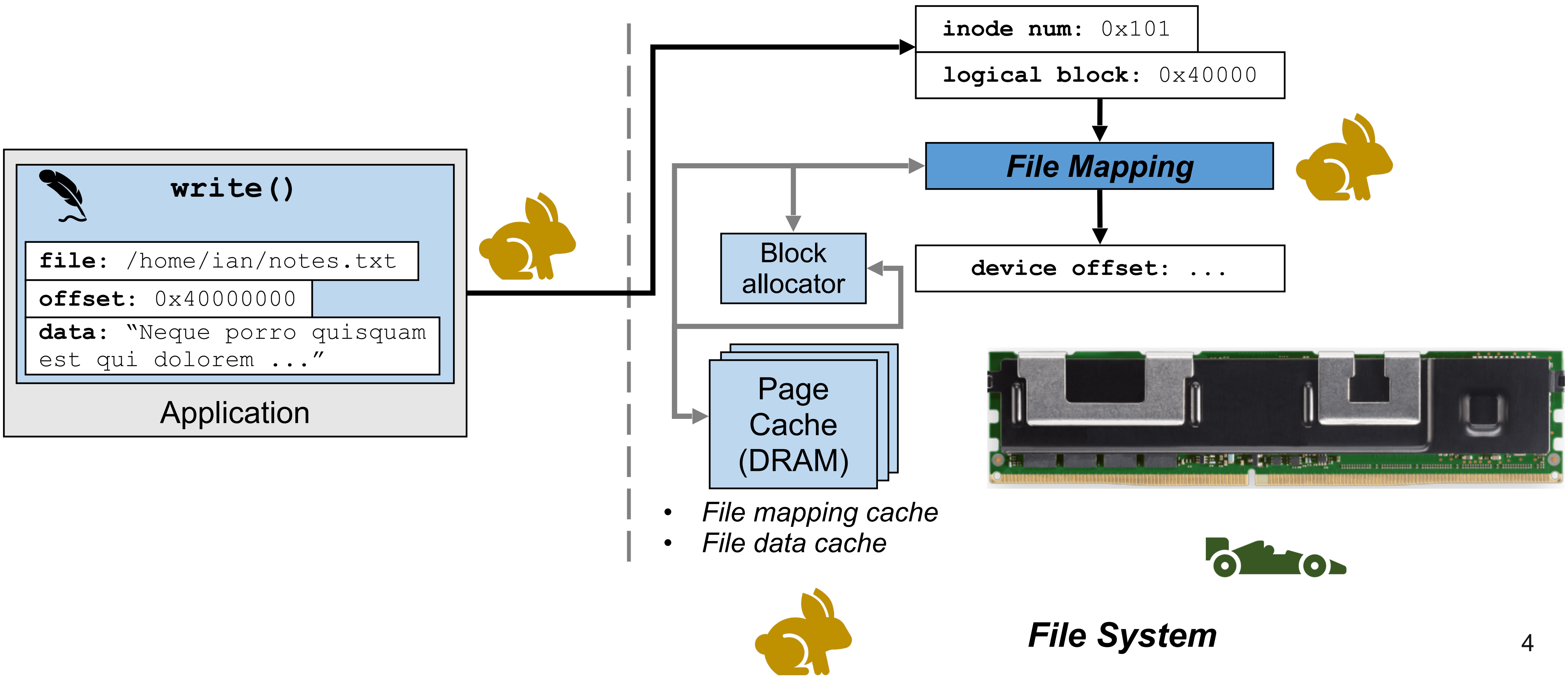


File IO After PM



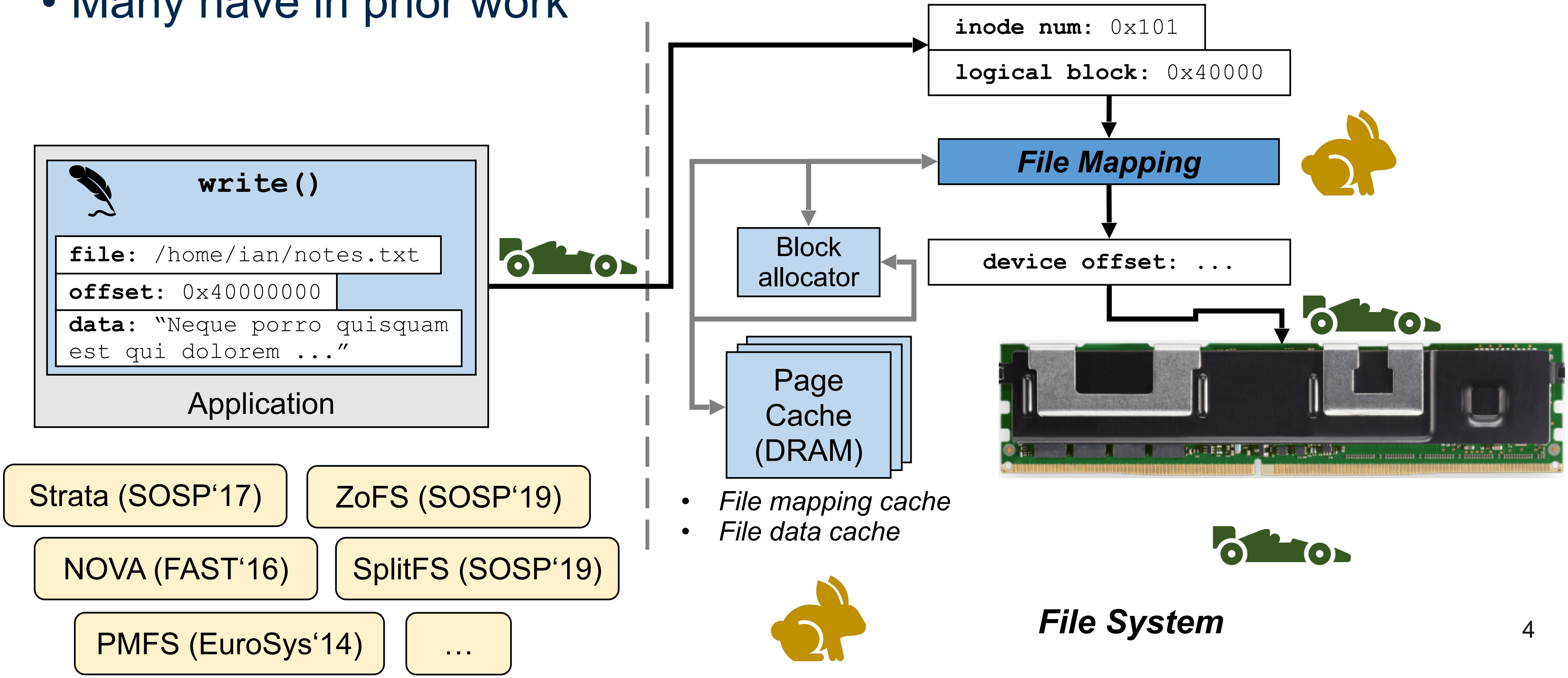
File IO After PM

- FS components must be optimized for PM



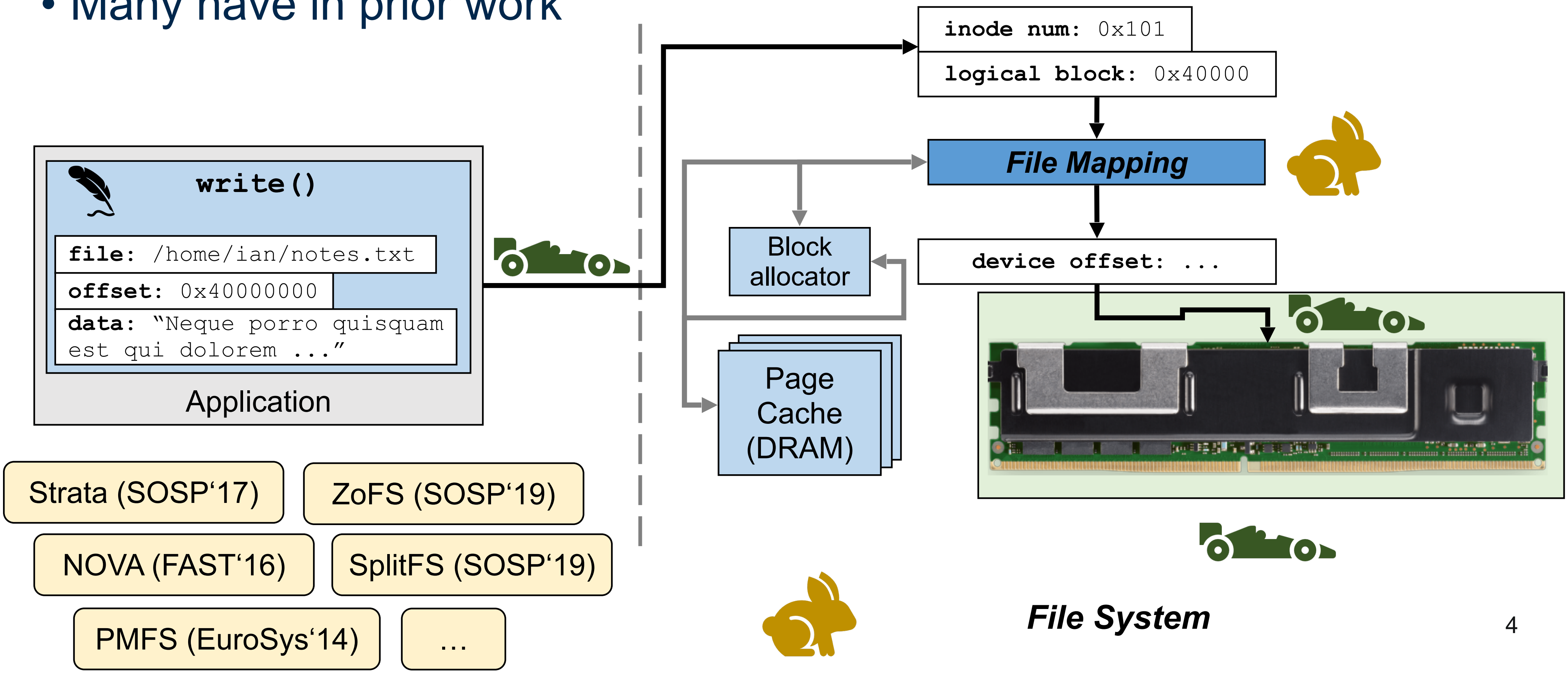
File IO After PM

- FS components must be optimized for PM
- Many have in prior work



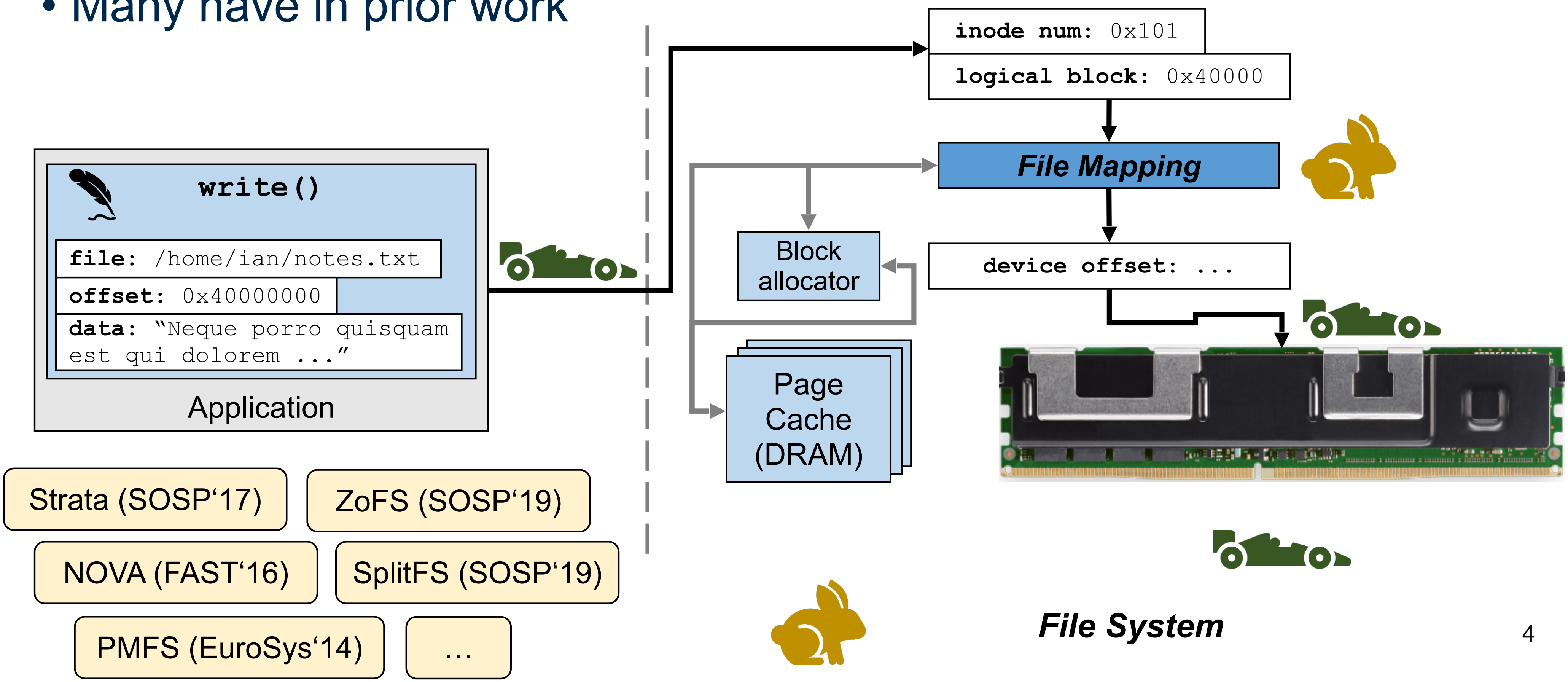
File IO After PM

- FS components must be optimized for PM
- Many have in prior work



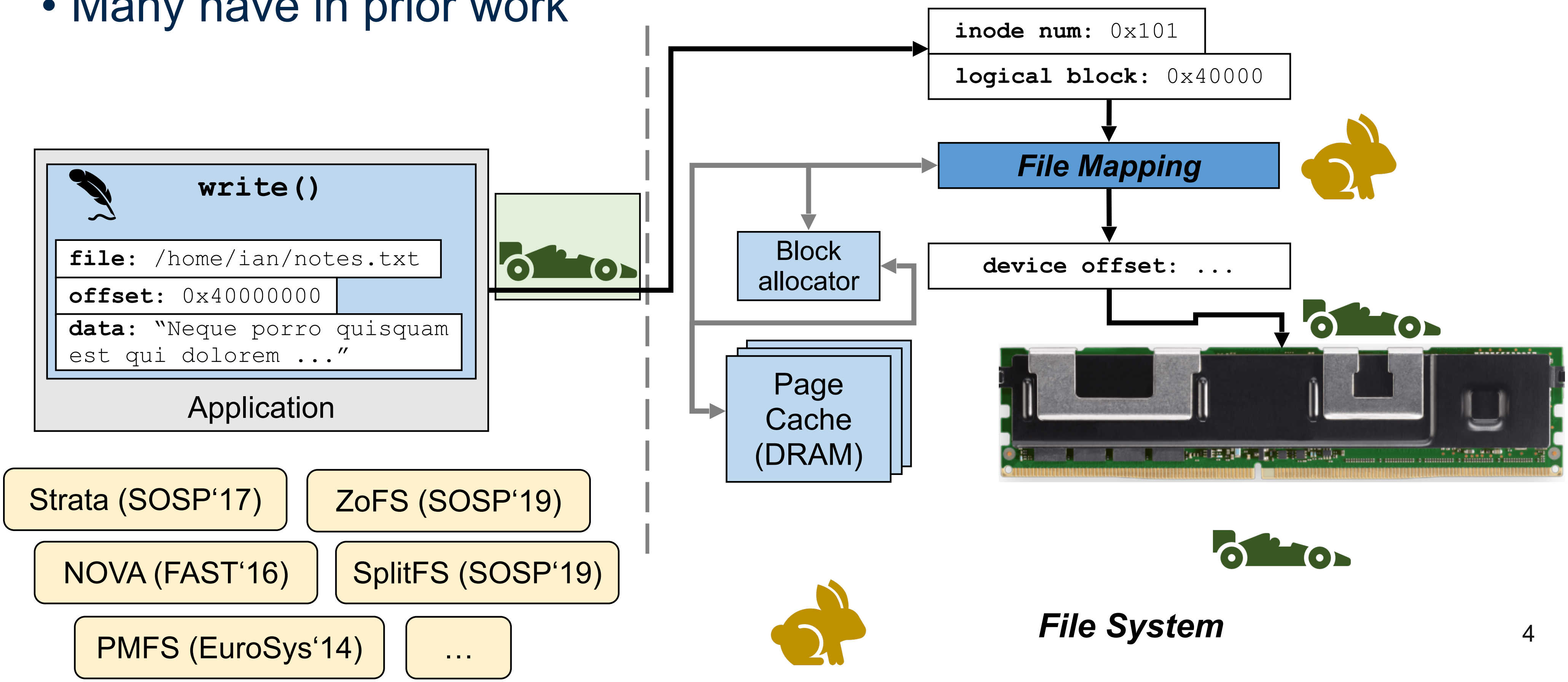
File IO After PM

- FS components must be optimized for PM
- Many have in prior work



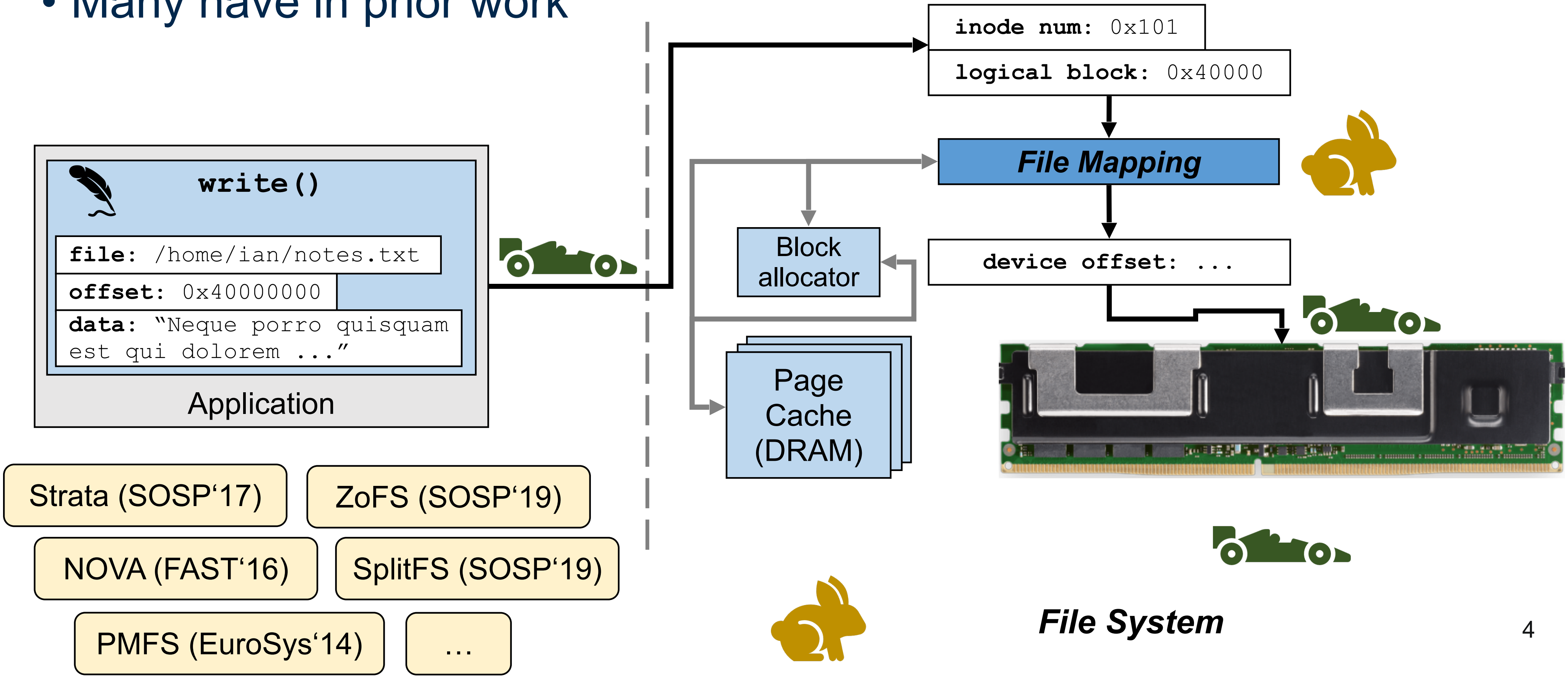
File IO After PM

- FS components must be optimized for PM
- Many have in prior work



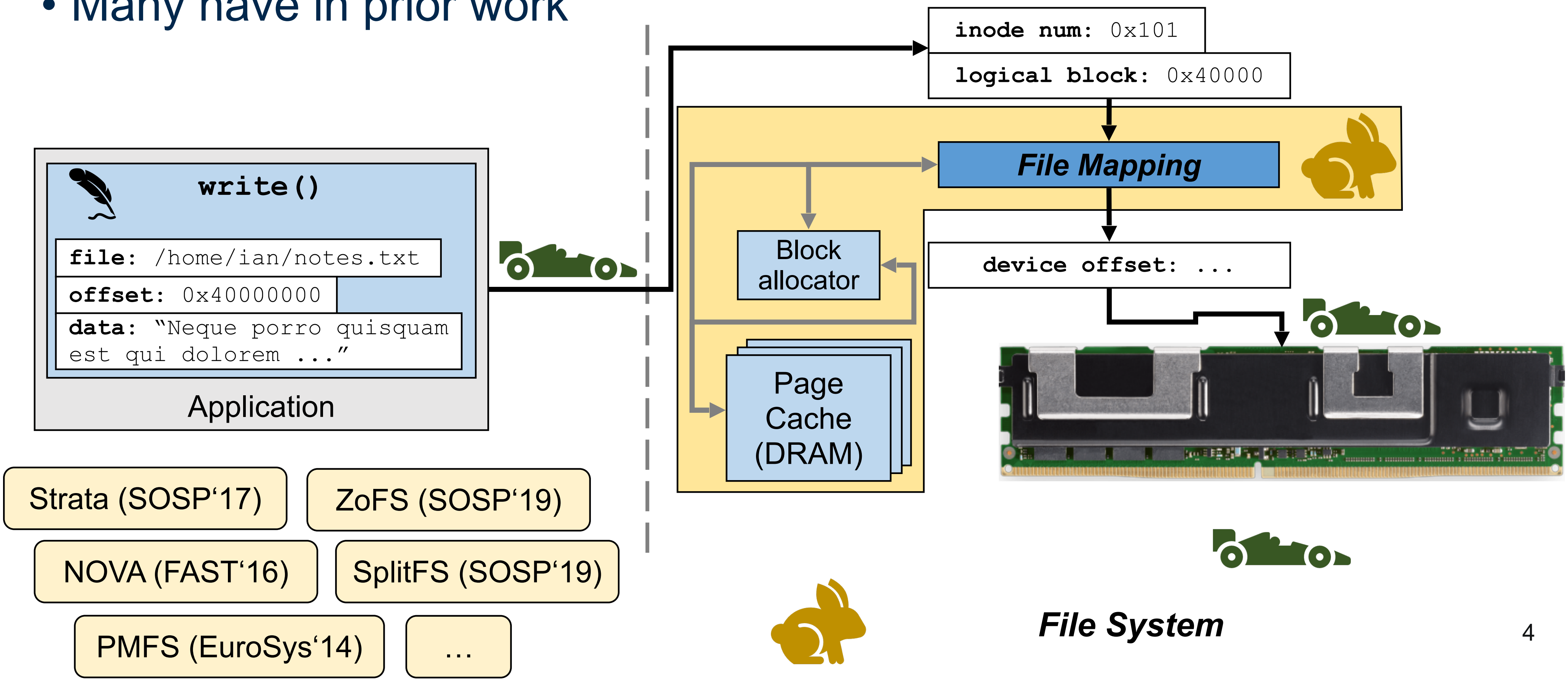
File IO After PM

- FS components must be optimized for PM
- Many have in prior work



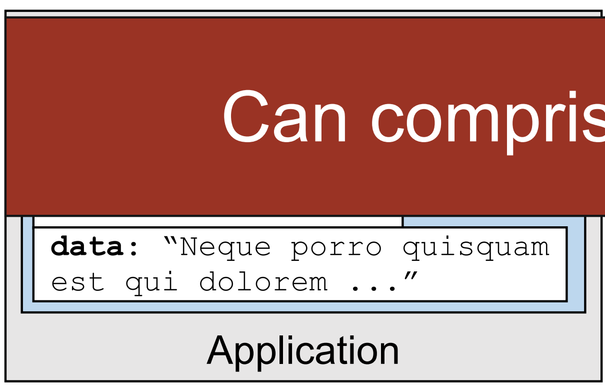
File IO After PM

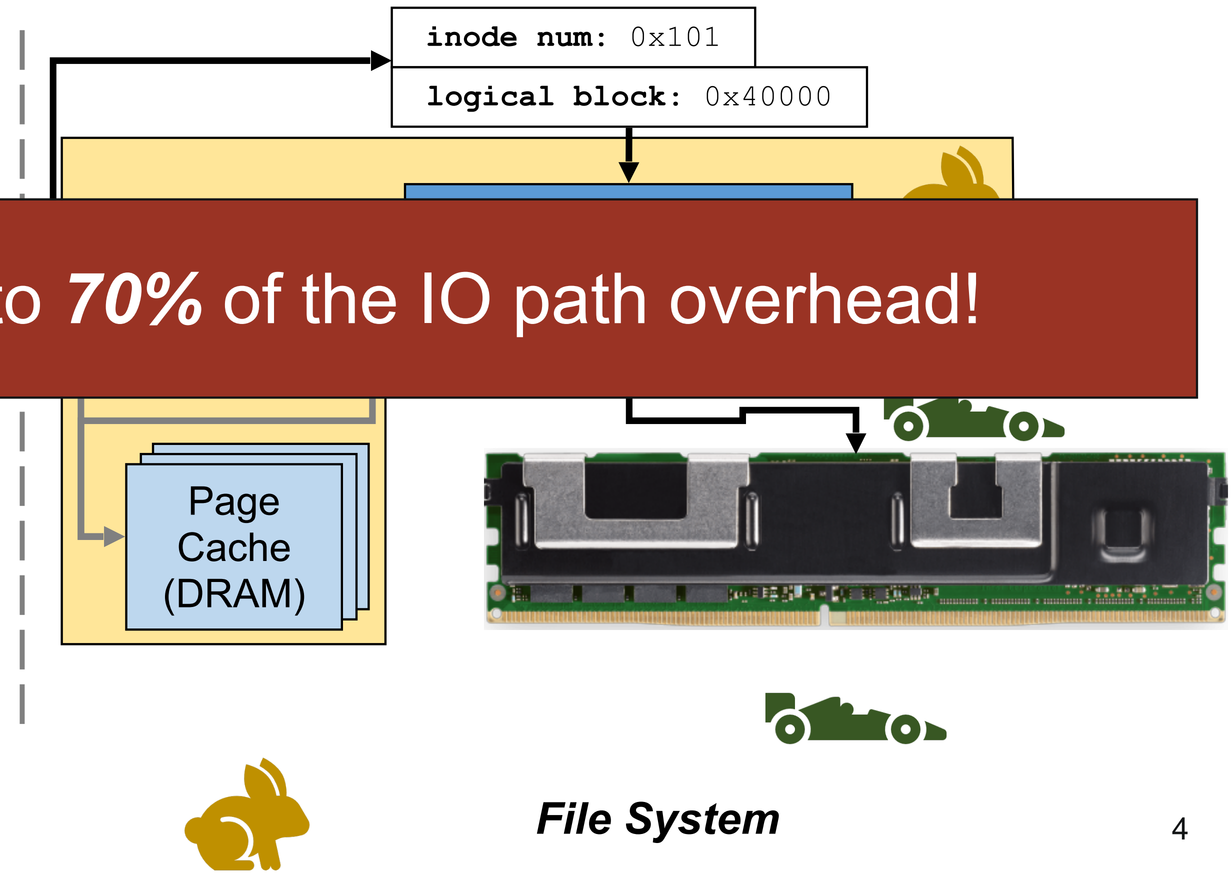
- FS components must be optimized for PM
- Many have in prior work



File IO After PM

- FS components must be optimized for PM
- Many have in prior work





Can comprise up to *70%* of the IO path overhead!

Strata (SOSP'17) | ZoFS (SOSP'19) | NOVA (FAST'16) | SplitFS (SOSP'19) | PMFS (EuroSys'14) | …

Our Contributions

Our Contributions

- Rigorously analyze file mapping in PM

Our Contributions

- Rigorously analyze file mapping in PM
- Optimize legacy PM file mapping structures

Our Contributions

- Rigorously analyze file mapping in PM
- Optimize legacy PM file mapping structures
- Design new PM-optimized file mapping approaches

Our Contributions

- Rigorously analyze file mapping in PM
- Optimize legacy PM file mapping structures
- Design new PM-optimized file mapping approaches
- Evaluate end-to-end performance on real workloads

Our Contributions

- Rigorously analyze file mapping in PM
- Optimize legacy PM file mapping structures
- Design new PM-optimized file mapping approaches
- Evaluate end-to-end performance on real workloads

Analysis Overview

Analysis Overview

How is file mapping affected by…	Design Question
Page caching?	Is page caching necessary?
File size?	Specialize for different file sizes?
IO size?	Optimize for sequential access?
Space utilization?	Make file mapping structure elastic?
Concurrency?	Is ensuring isolation important?
Locality?	Optimize for specific workloads?
Fragmentation?	Make robust against file system aging?
Storage structures?	Can we reuse PM storage structures?
Real workloads?	Are mapping optimizations impactful?

Analysis Overview

How is file mapping affected by…	Design Question
Page caching?	**Is page caching necessary?**
File size?	Specialize for different file sizes?
IO size?	Optimize for sequential access?
Space utilization?	Make file mapping structure elastic?
Concurrency?	Is ensuring isolation important?
Locality?	Optimize for specific workloads?
Fragmentation?	Make robust against file system aging?
Storage structures?	Can we reuse PM storage structures?
Real workloads?	**Are mapping optimizations impactful?**

Analysis Setup

Analysis Setup

- Evaluation setup:
  - Implemented in Strata (SOSP’17)
  - Baseline mapping structure: Extent trees in the page cache (Strata default)
  - Evaluated on 256 GB Intel Optane DC NVDIMMs

Analysis Setup

- Evaluation setup:
  - Implemented in Strata (SOSP'17)
  - Baseline mapping structure: Extent trees in the page cache (Strata default)
  - Evaluated on 256 GB Intel Optane DC NVDIMMs

- Analysis performed using YCSB on LevelDB
  - YCSB: Popular key-value store workload
  - LevelDB: Popular key-value store (used in original Strata evaluation)

Analysis Setup

- Evaluation setup:
  - Implemented in Strata (SOSP'17)
  - Baseline mapping structure: Extent trees in the page cache (Strata default)
  - Evaluated on 256 GB Intel Optane DC NVDIMMs

- Analysis performed using YCSB on LevelDB
  - YCSB: Popular key-value store workload
  - LevelDB: Popular key-value store (used in original Strata evaluation)

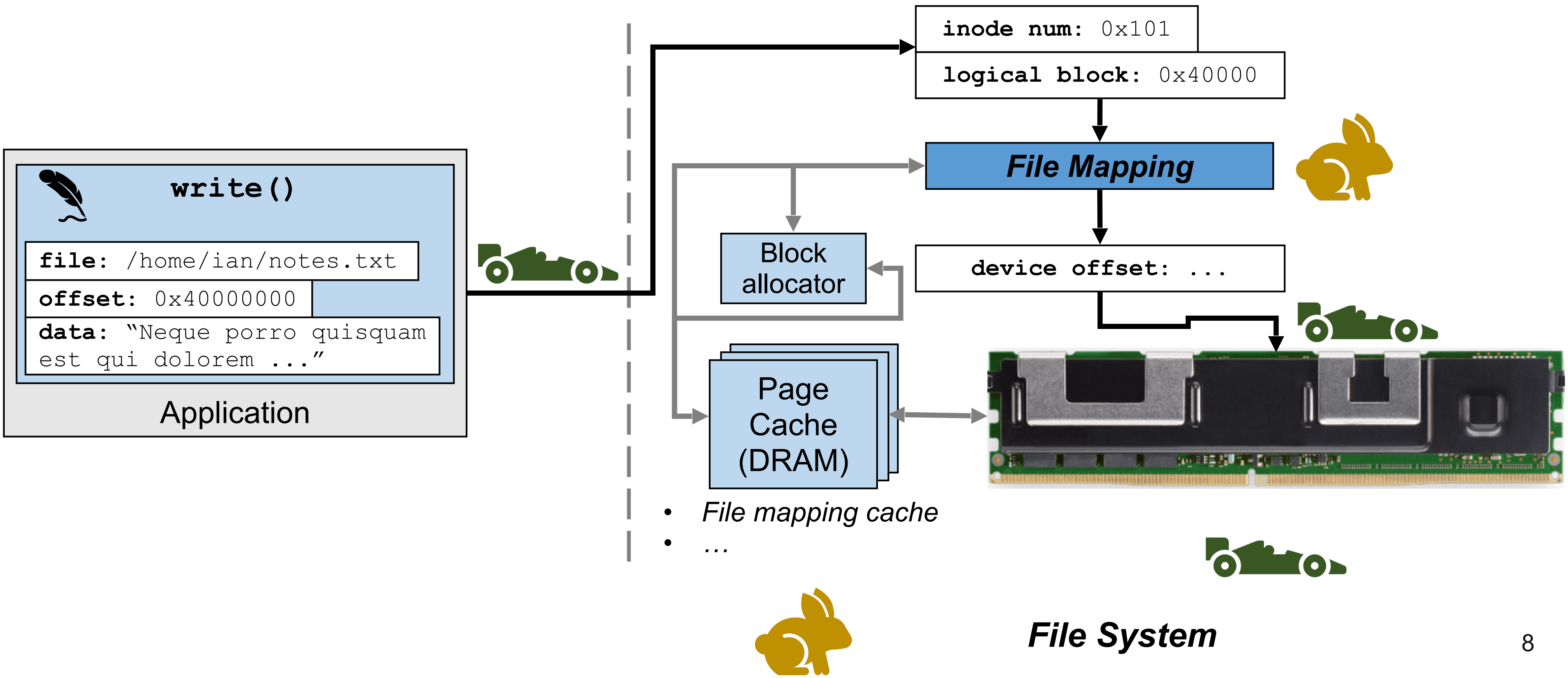
Analysis Setup

- Evaluation setup:
  - Implemented in Strata (SOSP'17)
  - Baseline mapping structure: Extent trees in the page cache (Strata default)
  - Evaluated on 256 GB Intel Optane DC NVDIMMs

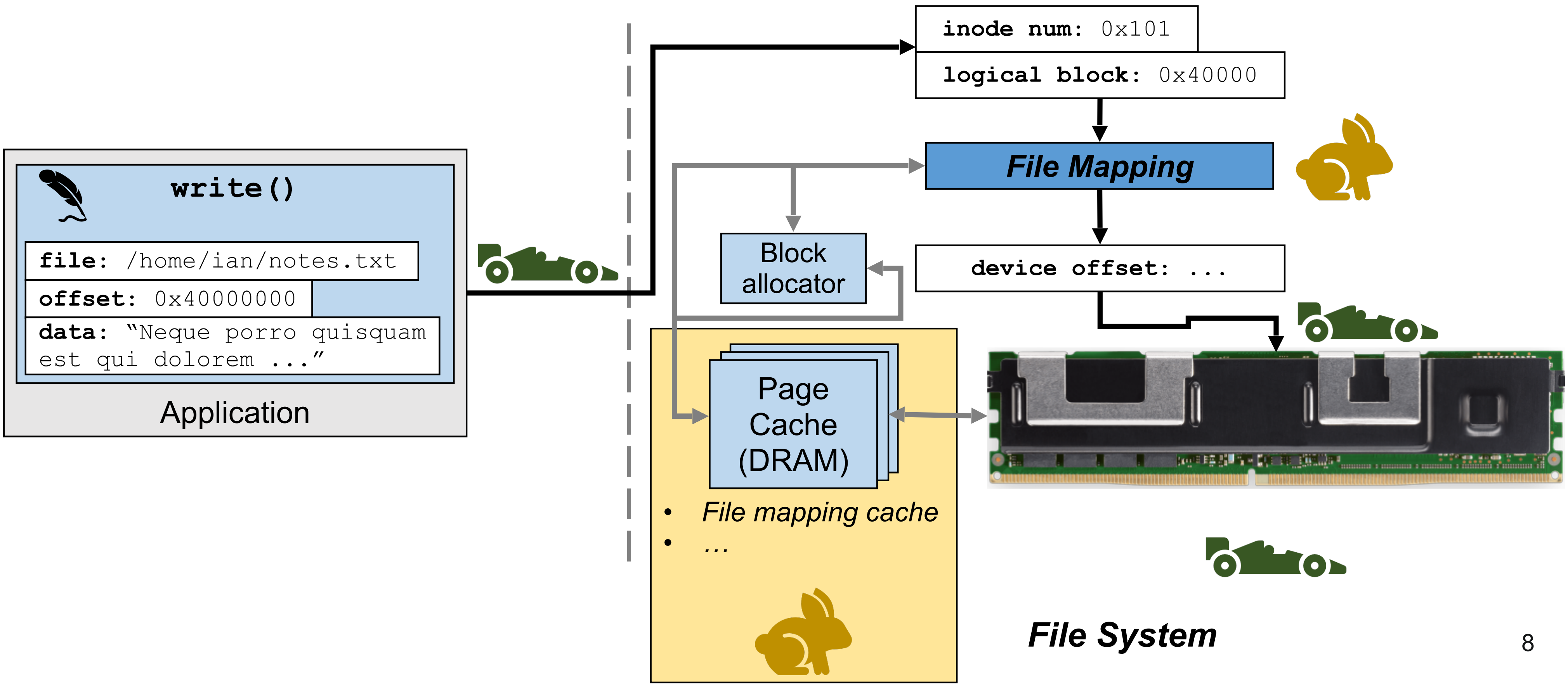
- Analysis performed using YCSB on LevelDB
  - YCSB: Popular key-value store workload
  - LevelDB: Popular key-value store (used in original Strata evaluation)

- Evaluate on Filebench
  - fileserver (1:2 read/write ratio)
  - webproxy (5:1 read/write ratio)

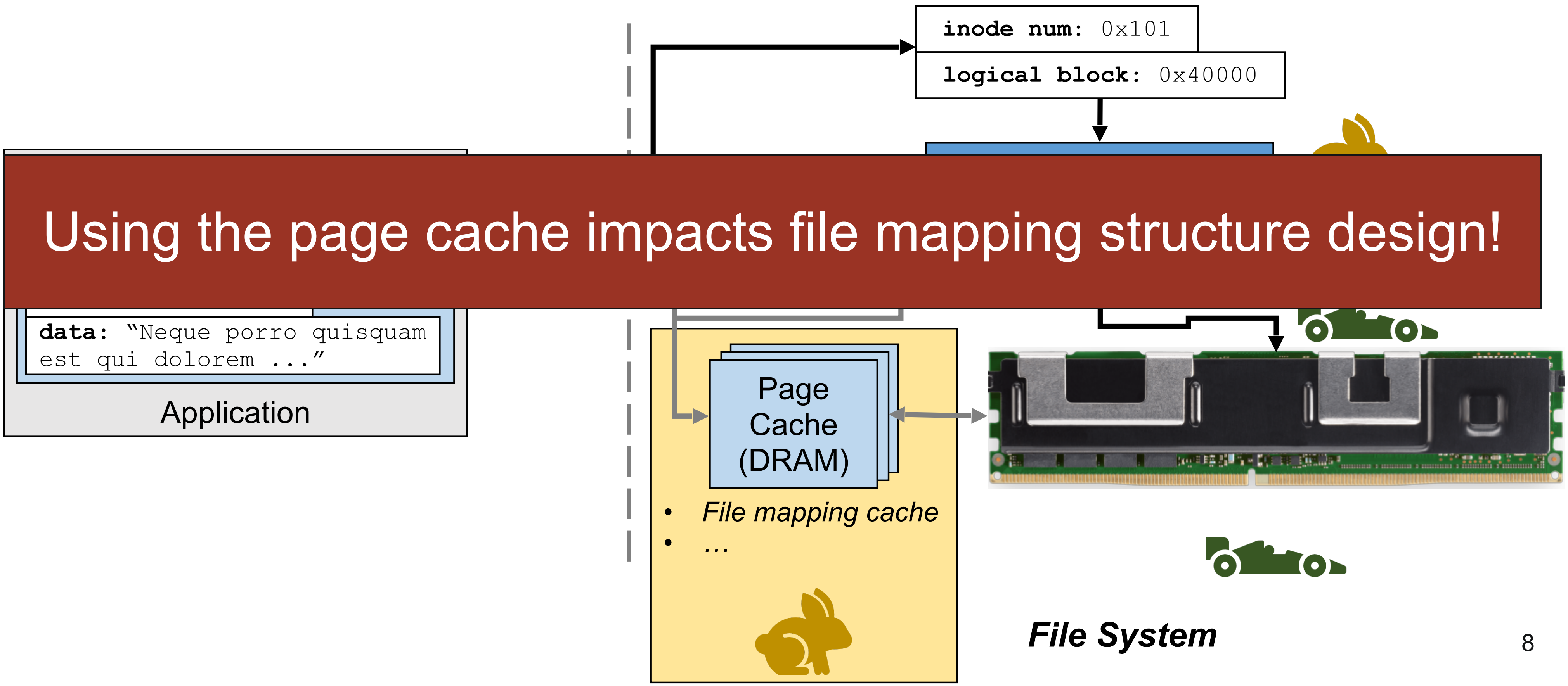
Is Page Caching Necessary?



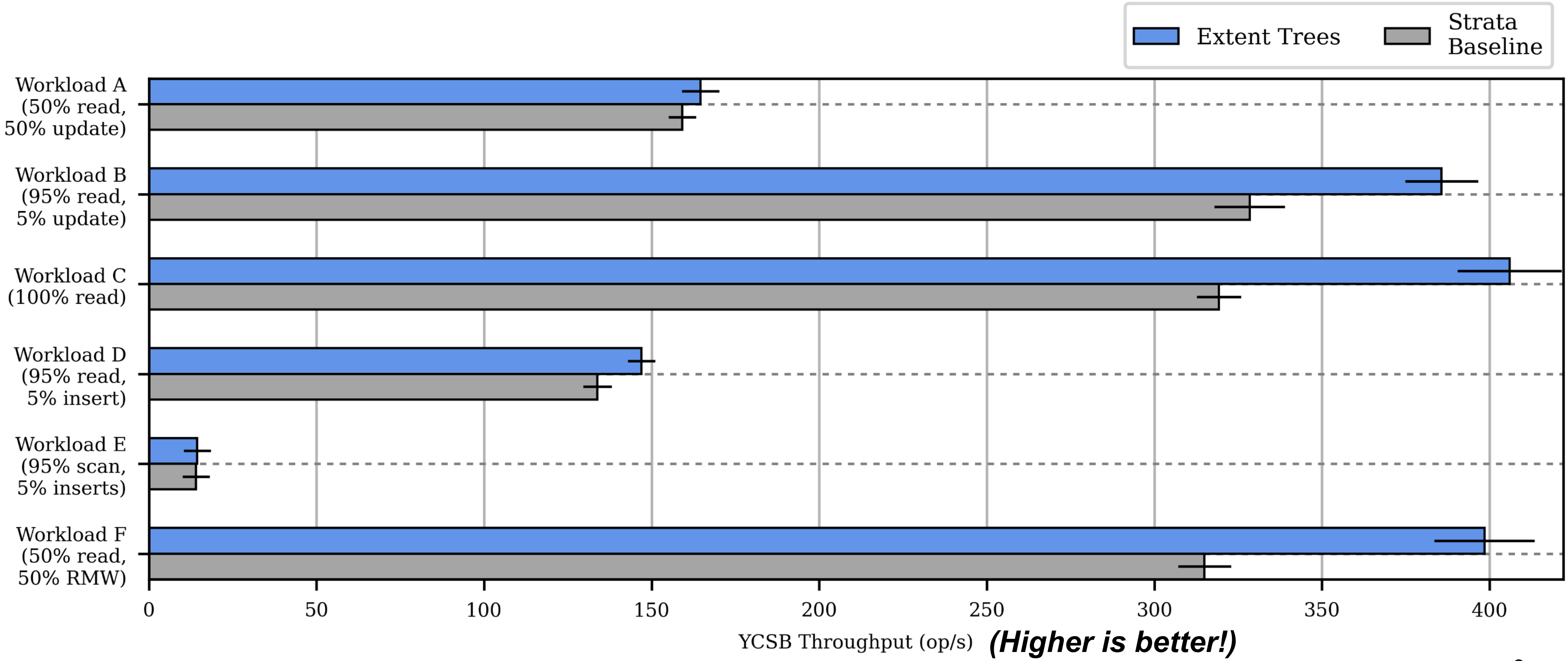
Is Page Caching Necessary?



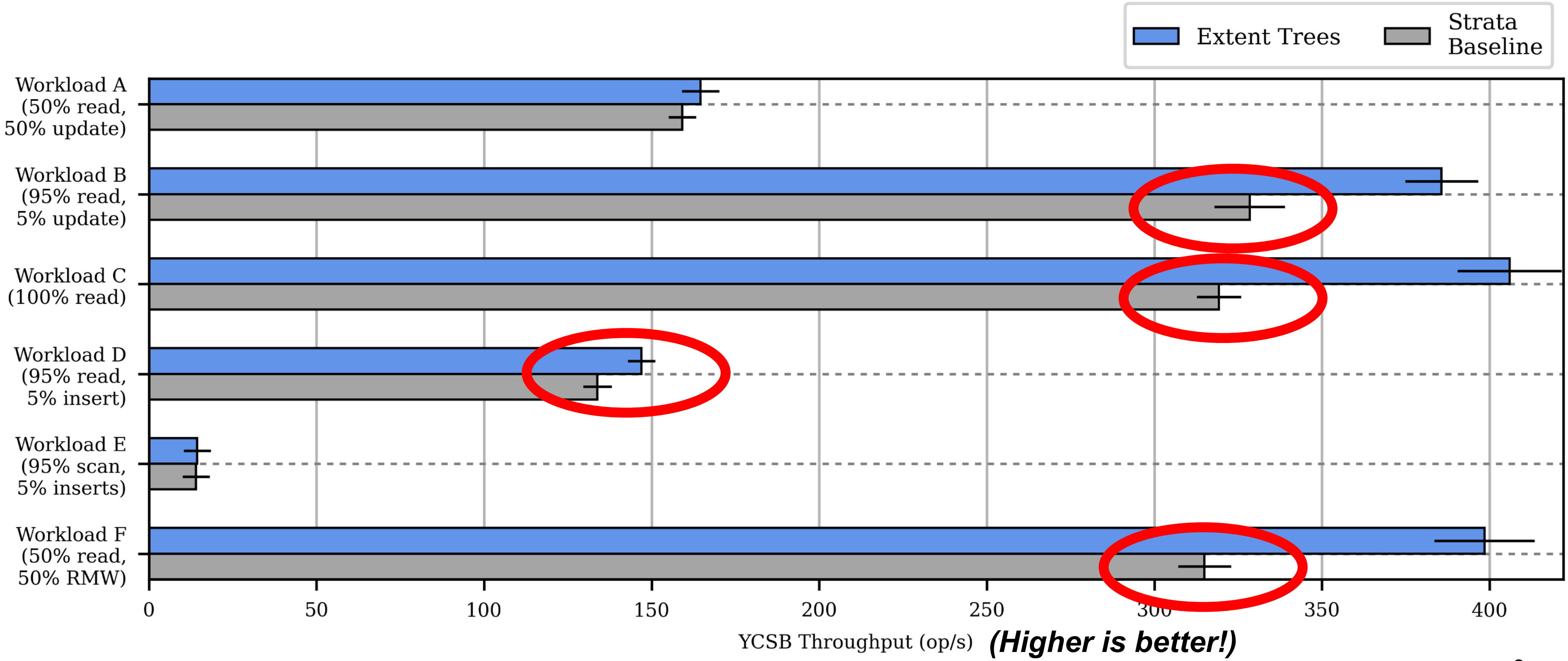
Is Page Caching Necessary?



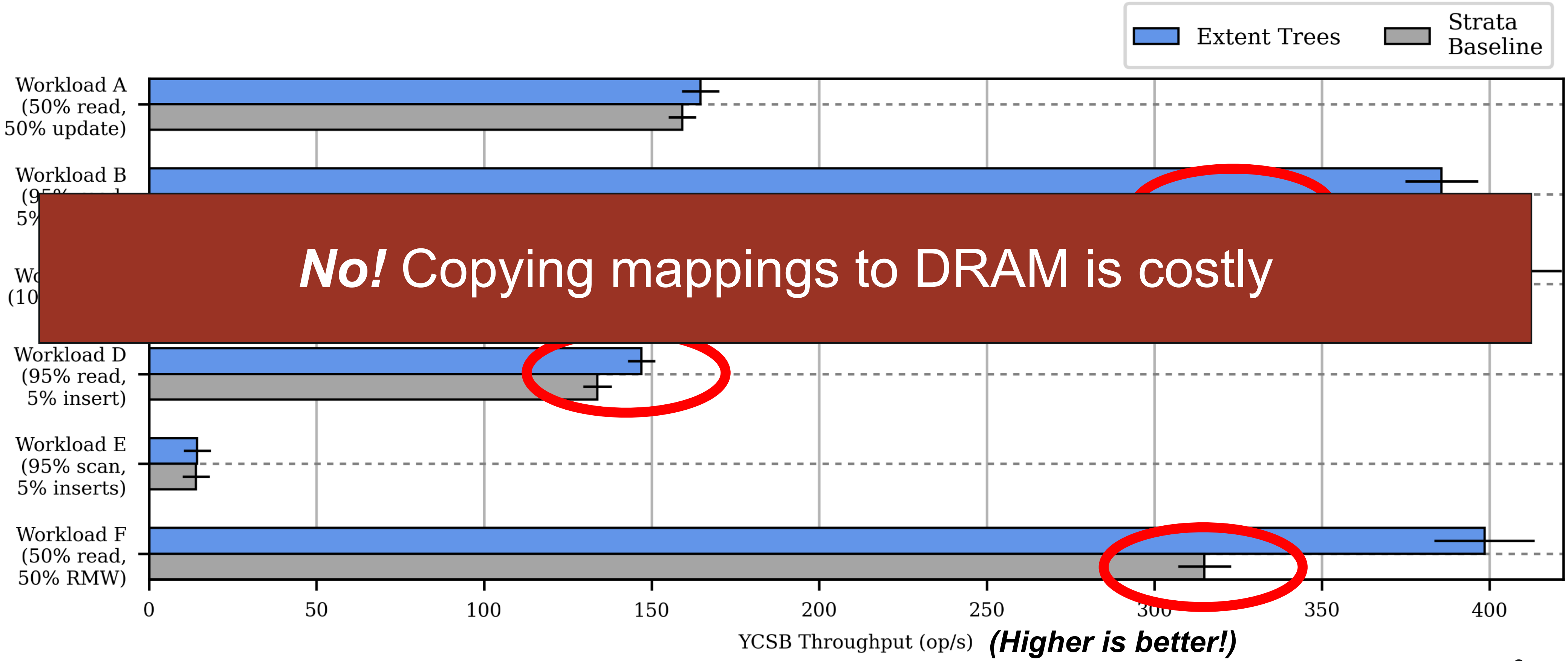
Is Page Caching Necessary?



Is Page Caching Necessary?



Is Page Caching Necessary?



No! Copying mappings to DRAM is costly

Contributions

- Rigorously analyze file mapping in PM
- Optimize legacy PM file mapping structures
- Design new PM-optimized file mapping approaches
- Evaluate end-to-end performance on real workloads

Analysis Overview

How is file mapping affected by…	Design Question
Page caching?	Is page caching necessary?
File size?	**Specialize for different file sizes?**
IO size?	**Optimize for sequential access?**
Space utilization?	**Make file mapping structure elastic?**
Concurrency?	**Is ensuring isolation important?**
Locality?	**Optimize for specific workloads?**
Fragmentation?	**Make robust against file system aging?**
Storage structures?	**Can we reuse PM storage structures?**
Real workloads?	Are mapping optimizations impactful?

Analysis Highlights

Analysis Highlights

- Analyze 4 different file mapping approaches optimized for PM

- **Optimize legacy PM file mapping structures**
  - Extent trees (Strata, ext4-DAX)
  - Radix trees (page cache mapping, NOVA)

Analysis Highlights

- Analyze 4 different file mapping approaches optimized for PM
- **Optimize legacy PM file mapping structures**
  - Extent trees (Strata, ext4-DAX)
  - Radix trees (page cache mapping, NOVA)
- Legacy structures suffered performance degradation on large files, update operations expensive

Analysis Highlights

- Analyze 4 different file mapping approaches optimized for PM
- **Optimize legacy PM file mapping structures**
  - Extent trees (Strata, ext4-DAX)
  - Radix trees (page cache mapping, NOVA)
- Legacy structures suffered performance degradation on large files, update operations expensive

Analysis Highlights

- Analyze 4 different file mapping approaches optimized for PM
- **Optimize legacy PM file mapping structures**
  - Extent trees (Strata, ext4-DAX)
  - Radix trees (page cache mapping, NOVA)
- Legacy structures suffered performance degradation on large files, update operations expensive
- **Design new PM-optimized file mapping approaches**
  - Cuckoo hashing
  - HashFS

HashFS

HashFS

- Hash table structure (linear probing)
  - Possible due to no page cache + PM byte addressability
    - Makes sparse, random updates efficient

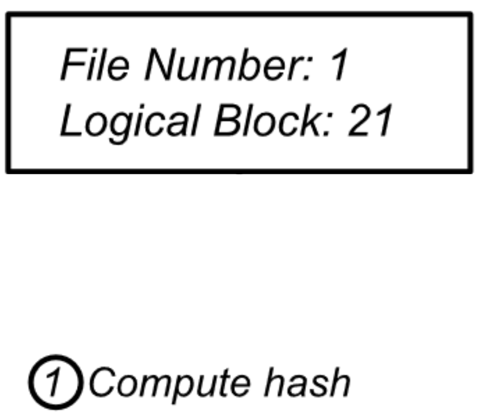
HashFS

- Hash table structure (linear probing)
  - Possible due to no page cache + PM byte addressability
    - Makes sparse, random updates efficient

File Number: 1
Logical Block: 21

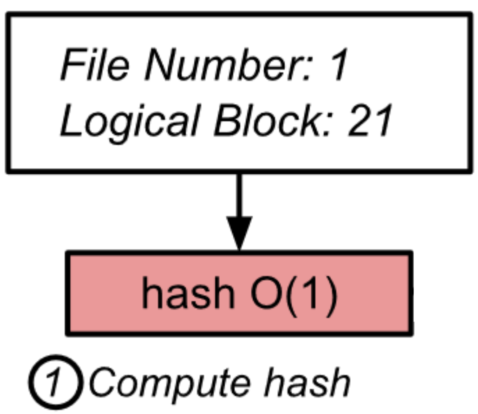
HashFS

- Hash table structure (linear probing)
  - Possible due to no page cache + PM byte addressability
    - Makes sparse, random updates efficient



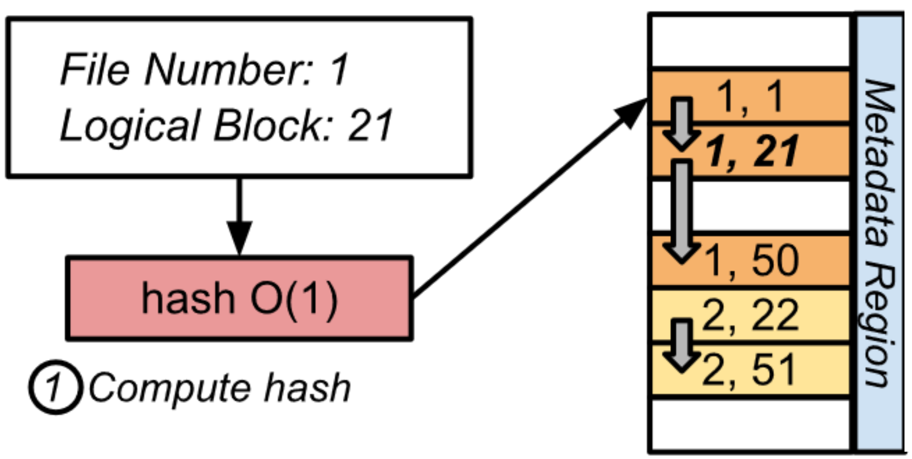
HashFS

- Hash table structure (linear probing)
  - Possible due to no page cache + PM byte addressability
    - Makes sparse, random updates efficient



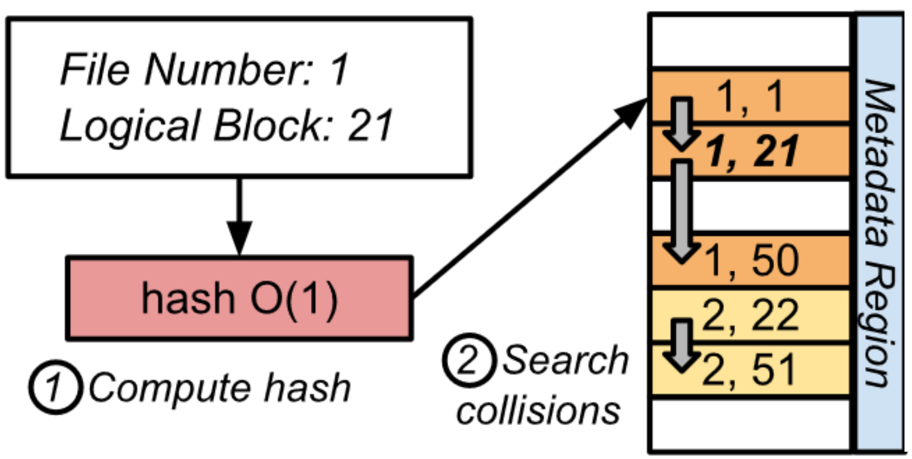
HashFS

- Hash table structure (linear probing)
  - Possible due to no page cache + PM byte addressability
    - Makes sparse, random updates efficient



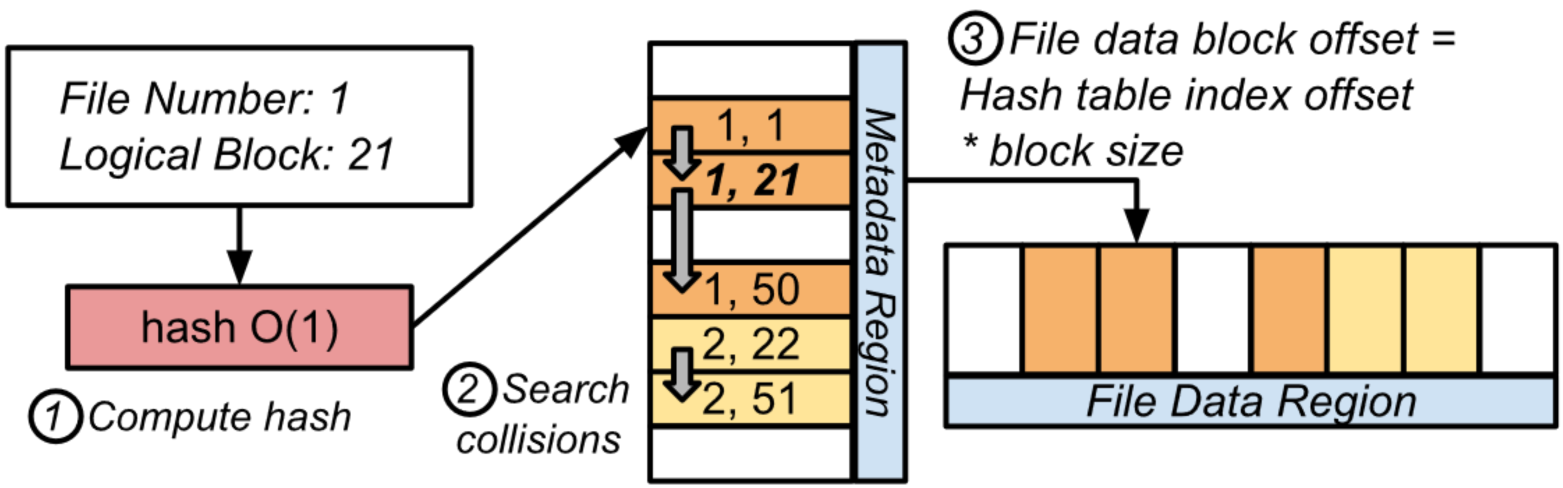
HashFS

- Hash table structure (linear probing)
  - Possible due to no page cache + PM byte addressability
    - Makes sparse, random updates efficient



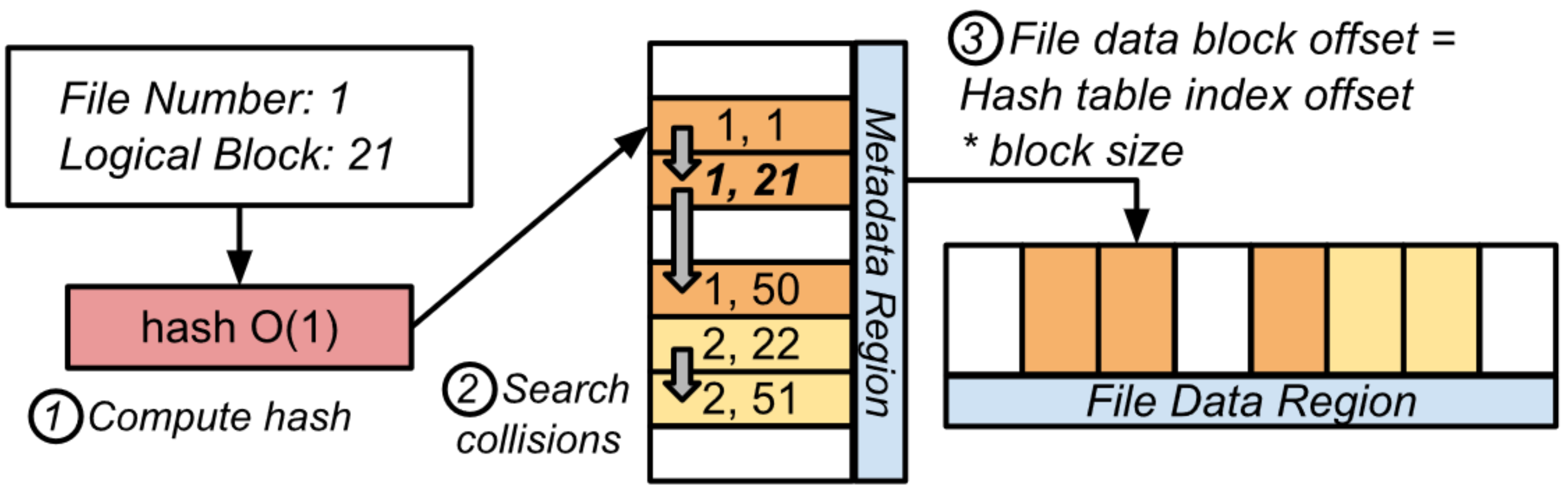
HashFS

- Hash table structure (linear probing)
  - Possible due to no page cache + PM byte addressability
    - Makes sparse, random updates efficient

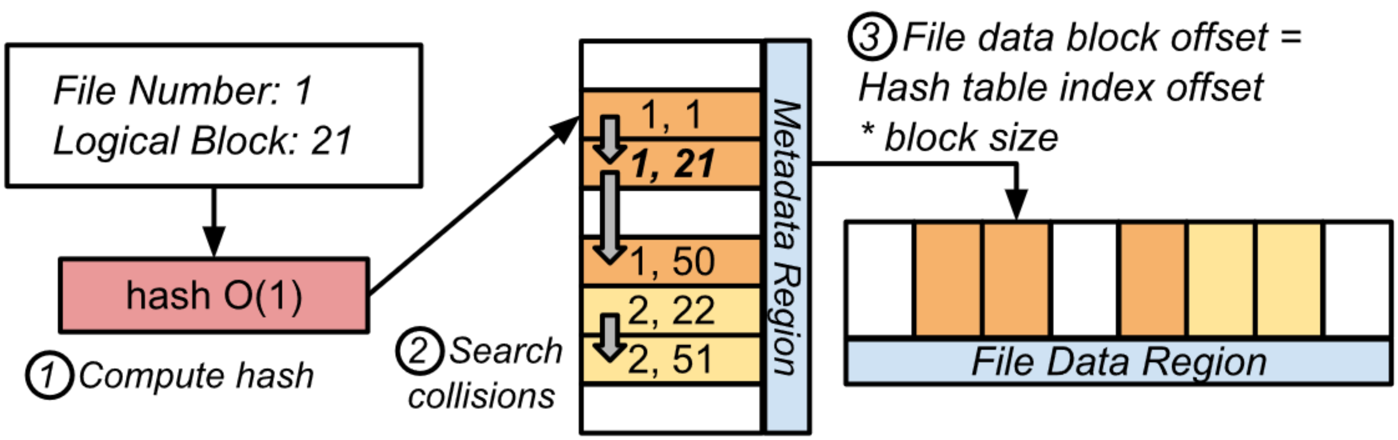


HashFS

- Hash table structure (linear probing)
  - Possible due to no page cache + PM byte addressability
    - Makes sparse, random updates efficient
- Combined block-allocation and file-mapping scheme
  - Insert into hash table implicitly allocates block at corresponding offset
  - **Bypasses expensive block allocator management** (cf. our paper)



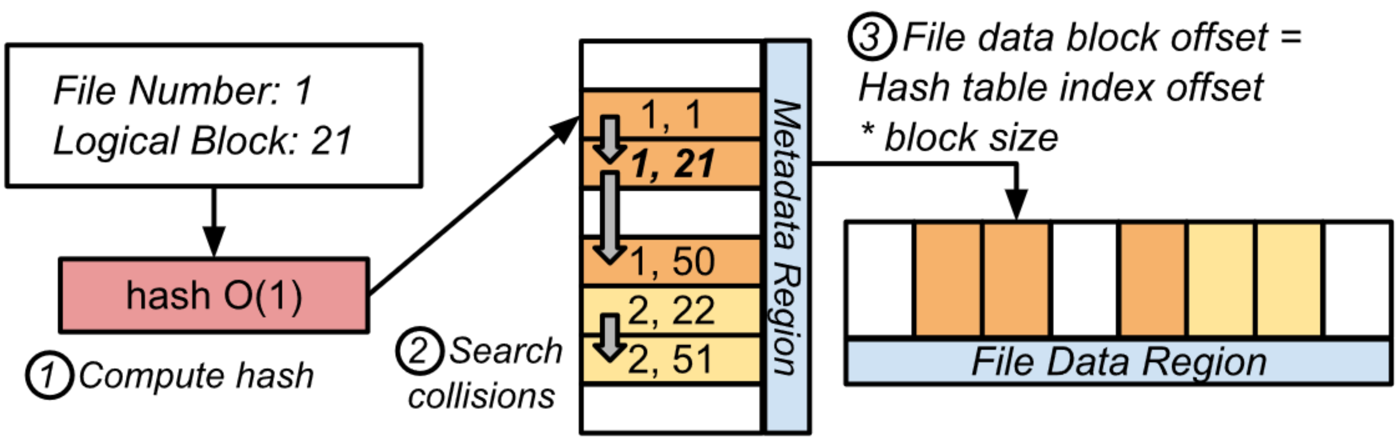
HashFS (cont.)



File Number: 1
Logical Block: 21
hash O(1)
1 Compute hash
2 Search collisions
1, 1
1, 21
1, 50
2, 22
2, 51
Metadata Region
3 File data block offset = Hash table index offset * block size
File Data Region

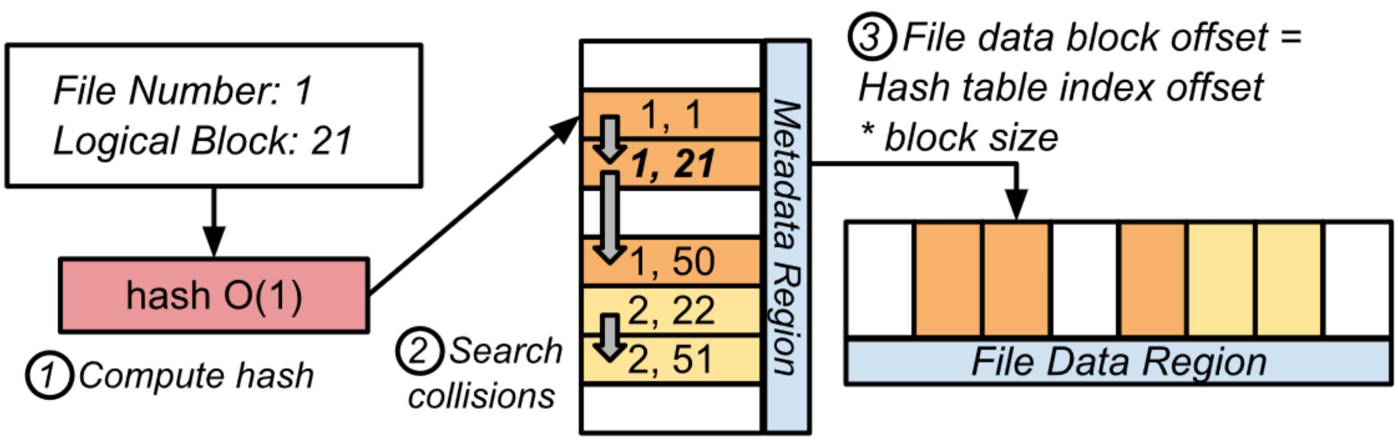
HashFS (cont.)

- Need to avoid all resizing (incurs high update latency)
  - Must also be a global structure (one structure for all files)
  - Statically allocate max size at creation



HashFS (cont.)

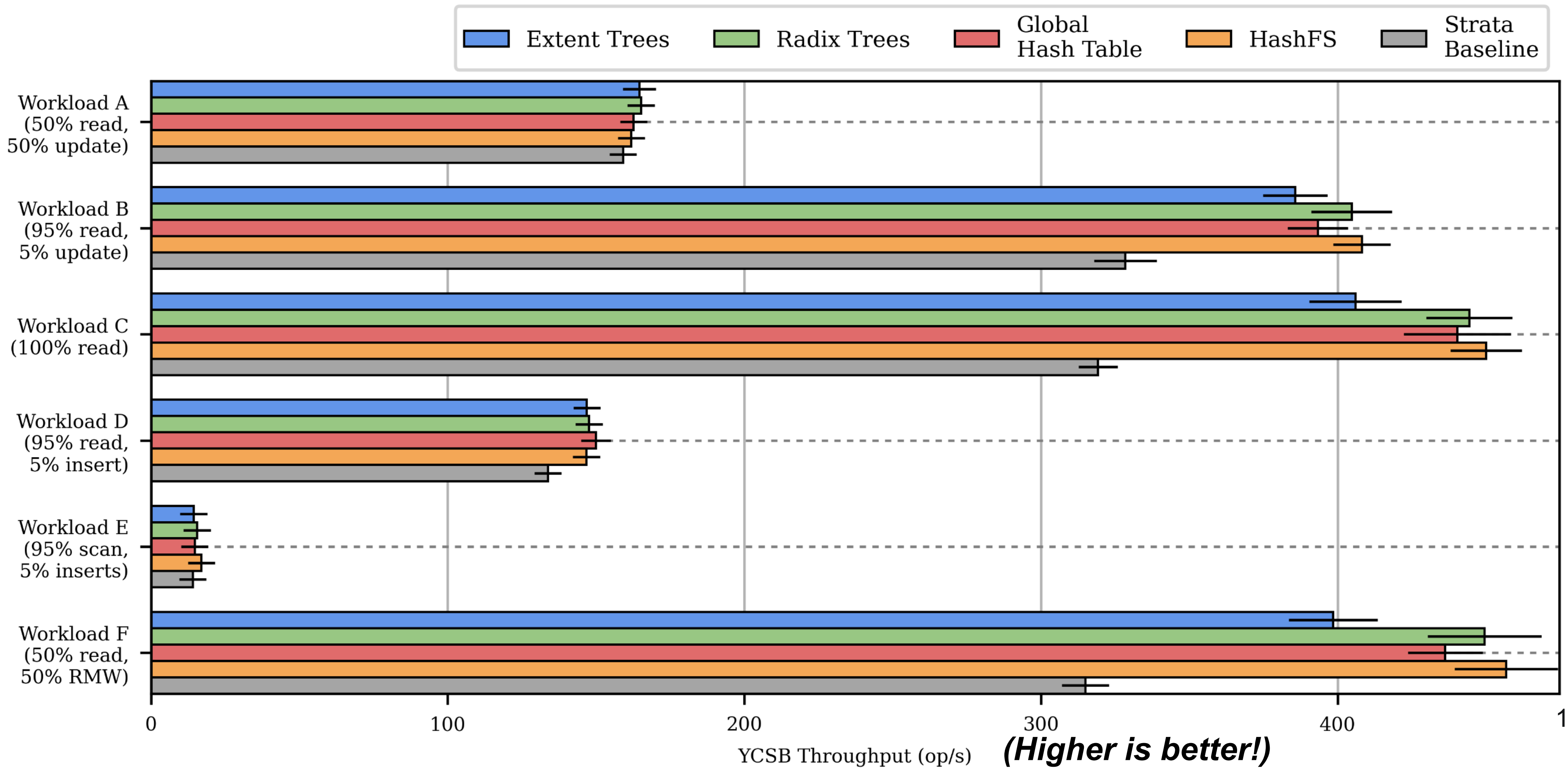
- Need to avoid all resizing (incurs high update latency)
  - Must also be a global structure (one structure for all files)
  - Statically allocate max size at creation
- FS optimization: use SIMD for large IO operations
  - Many file system workloads perform large IO operations
  - For efficiency, mapping structures must return ranges of mappings
  - **Perform hash table operations in parallel**



Contributions

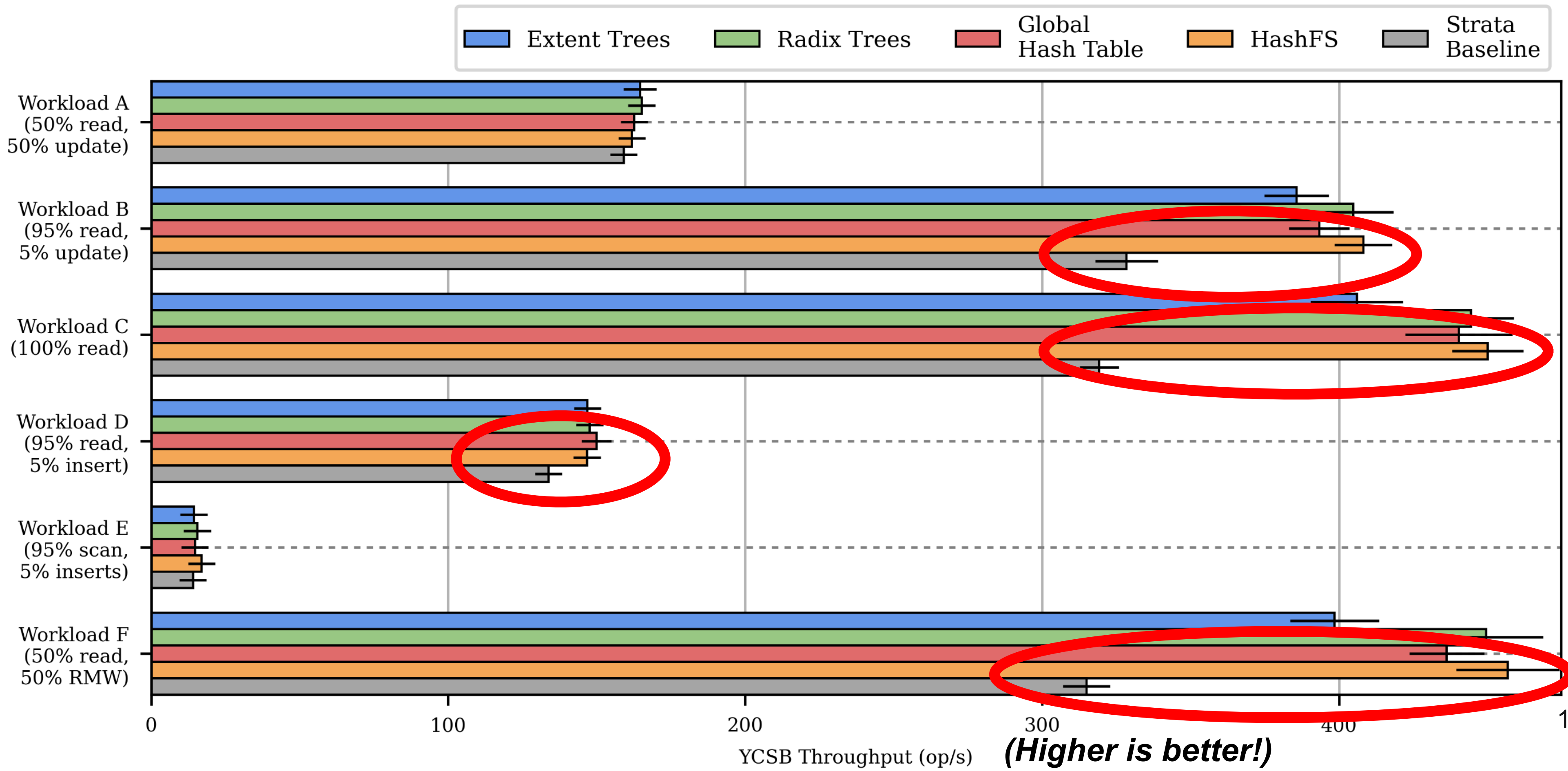
- Rigorously analyze file mapping in PM
- Optimize legacy PM file mapping structures
- Design new PM-optimized file mapping approaches
- Evaluate end-to-end performance on real workloads

Are File Mapping Optimizations Impactful?



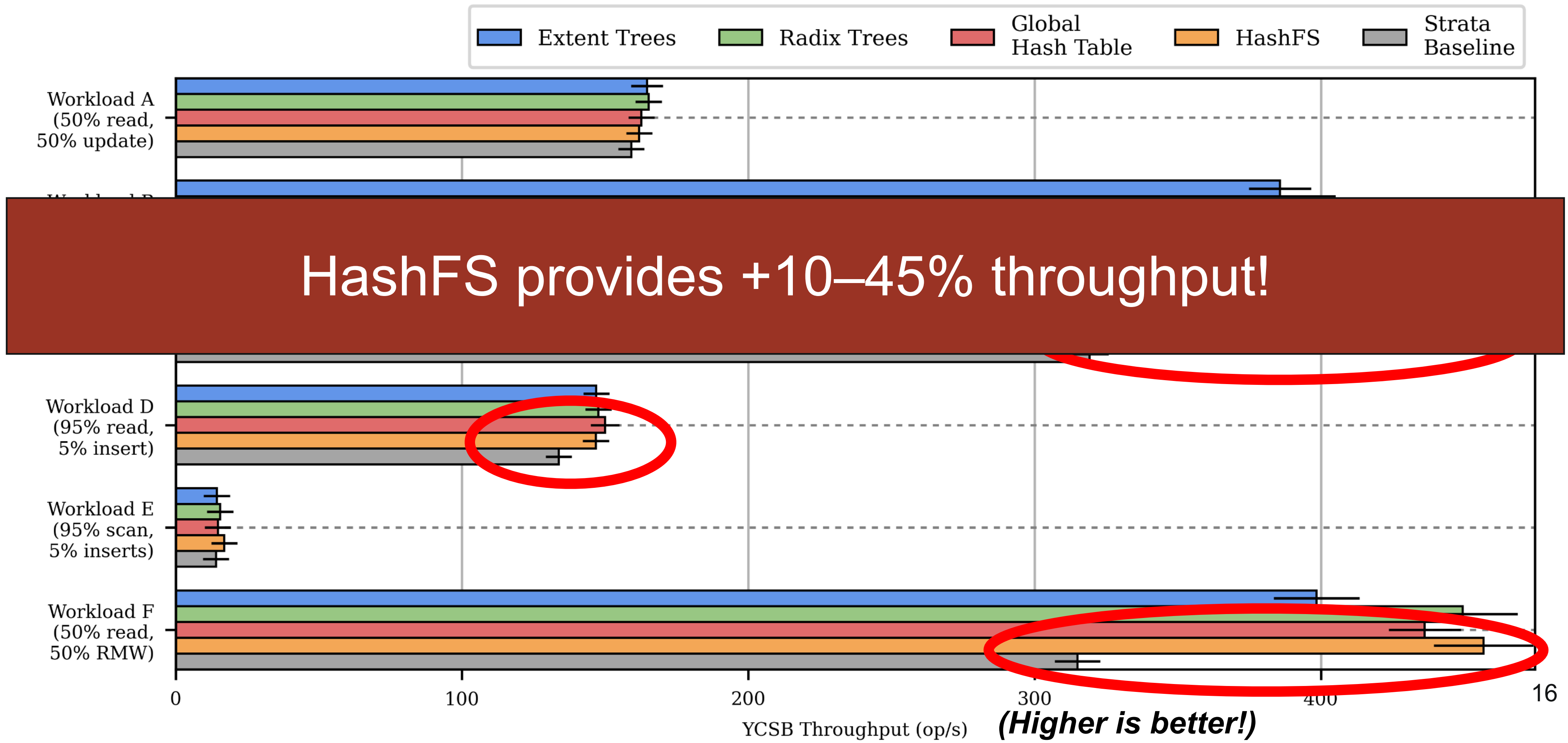
(Higher is better!)

Are File Mapping Optimizations Impactful?

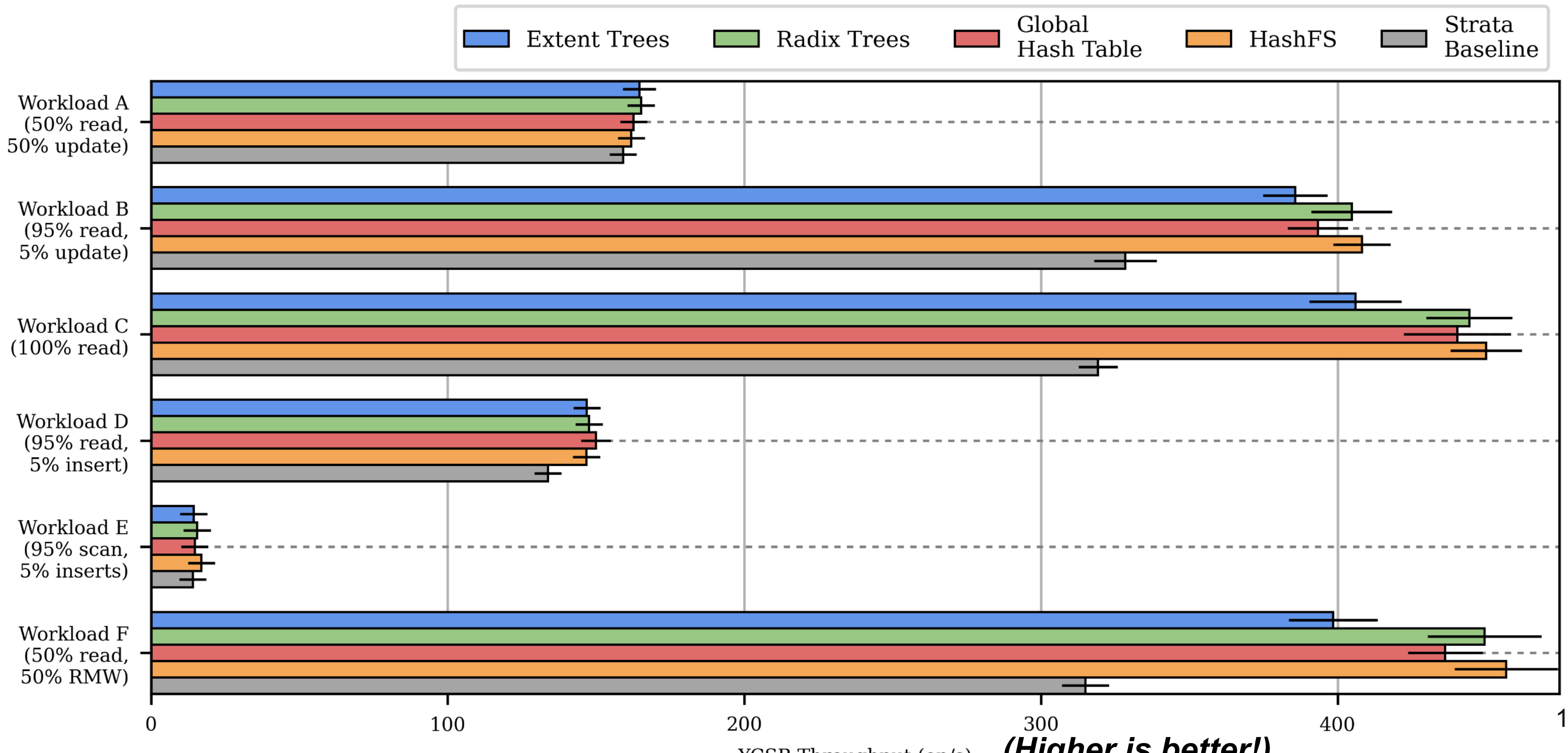


(Higher is better!)

Are File Mapping Optimizations Impactful?

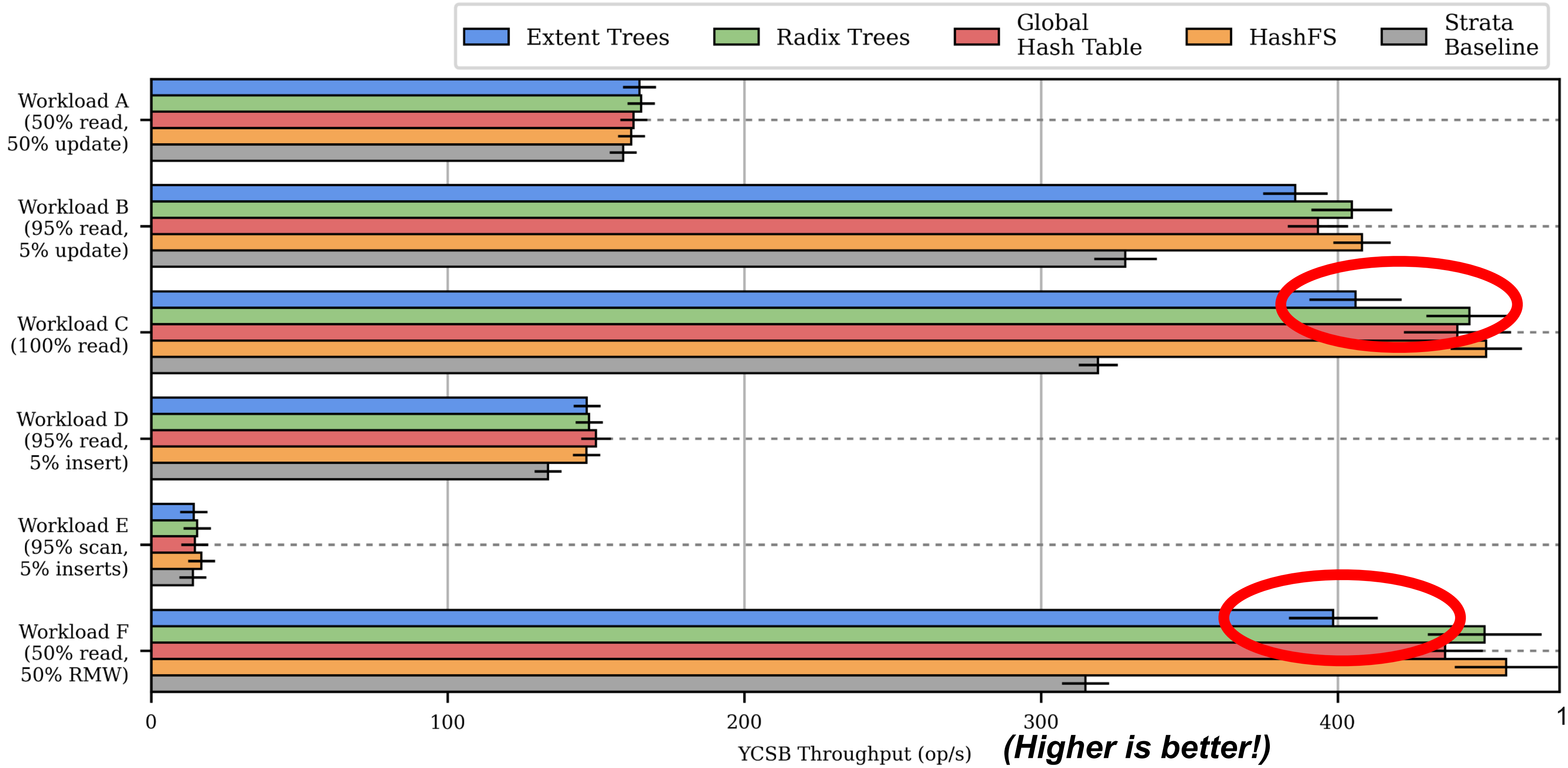


Are File Mapping Optimizations Impactful?



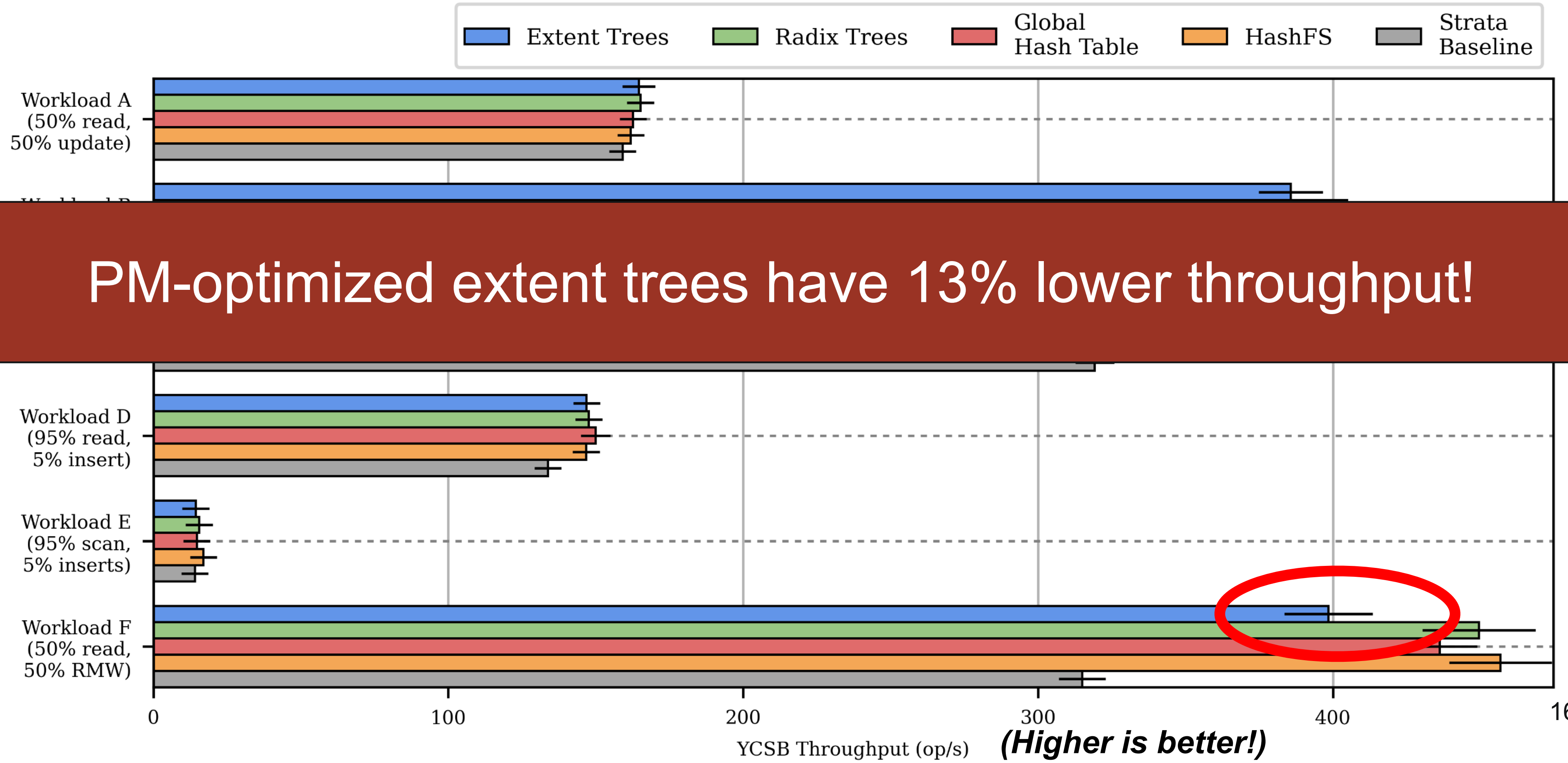
(Higher is better!)

Are File Mapping Optimizations Impactful?



(Higher is better!)

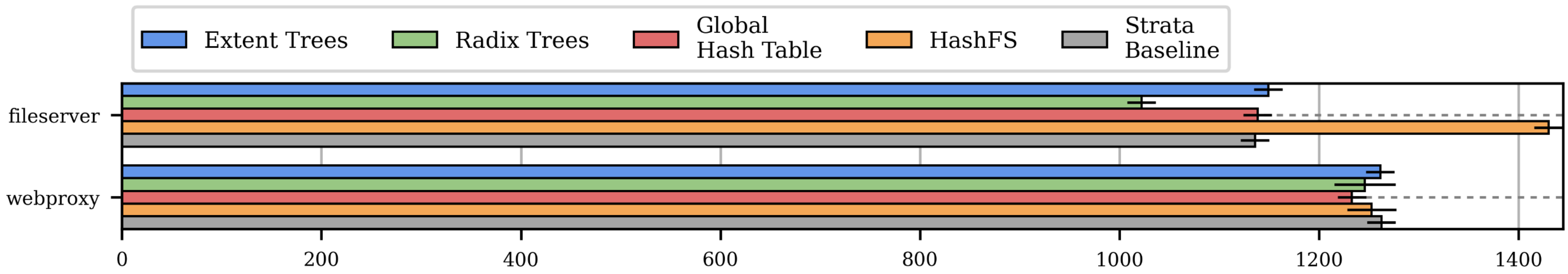
Are File Mapping Optimizations Impactful?



PM-optimized extent trees have 13% lower throughput!

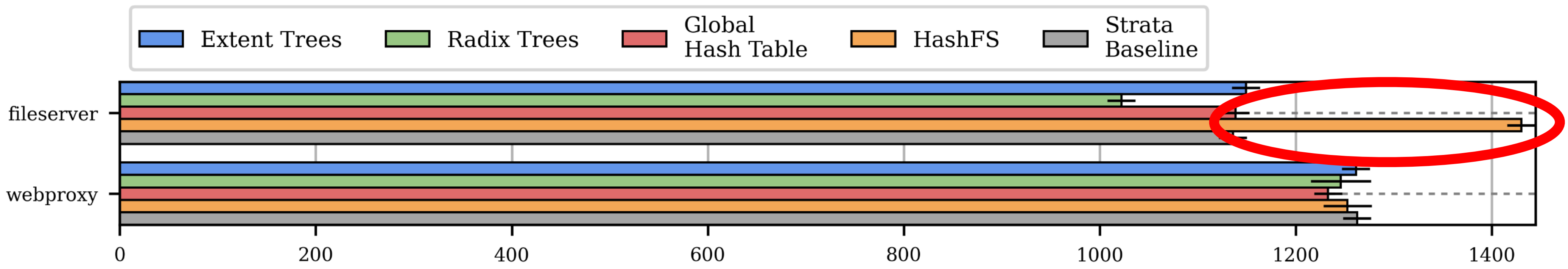
(Higher is better!)

Are File Mapping Optimizations Impactful?



(Higher is better!)

Are File Mapping Optimizations Impactful?



(Higher is better!)

HashFS provides +26% throughput!

Conclusions

Conclusions

- We must re-examine file mapping specifically for PM file systems!

Conclusions

- We must re-examine file mapping specifically for PM file systems!

- A rigorous analysis yields insights into performance-shortfalls of existing mapping approaches

Conclusions

- We must re-examine file mapping specifically for PM file systems!
- A rigorous analysis yields insights into performance-shortfalls of existing mapping approaches
- We design two new, global file mapping approaches (cuckoo hashing, HashFS)

Conclusions

- We must re-examine file mapping specifically for PM file systems!
- A rigorous analysis yields insights into performance-shortfalls of existing mapping approaches
- We design two new, global file mapping approaches (cuckoo hashing, HashFS)
- HashFS (our new PM-optimized file mapping approach) outperforms the state-of-the-art by up to 45% in real workloads

Thank you!

Corresponding Author: **Ian Neal**

iangneal@umich.edu

https://about.iangneal.io