De-indirection for Flash-based SSDs with Nameless Writes

Yiying Zhang, Leo Prasath Aruraj, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau

Indirection: Too Much of a Good Thing?

All problems in computer science can be solved by another level of indirection.*

Indirection

- Mapping between different objects
- Flexibility, simplicity, modularity

Excess indirection

- Redundant levels of indirection in a system
- Space and performance cost

Indirection in Flash-based SSDs

- File offset -> logical address -> physical address
- Hides erase-before-program and wear leveling

* Usually attributed to Butler Lampson

De-indirection with Nameless Writes

…but that usually will create another problem.**

De-indirection

- Remove excess indirection
- The Turtles project [1]

New interface: Nameless Write

- Write without a name (logical address)
- Device allocates and returns physical address
- File system stores physical address

Advantages

- Reduces space and performance cost
- Device maintains critical controls

** Original quote by David Wheeler

Nameless Write Interfaces

Nameless Write

- Writes only data and no logical address

Nameless Overwrite

- Writes data and old physical address

Physical Read

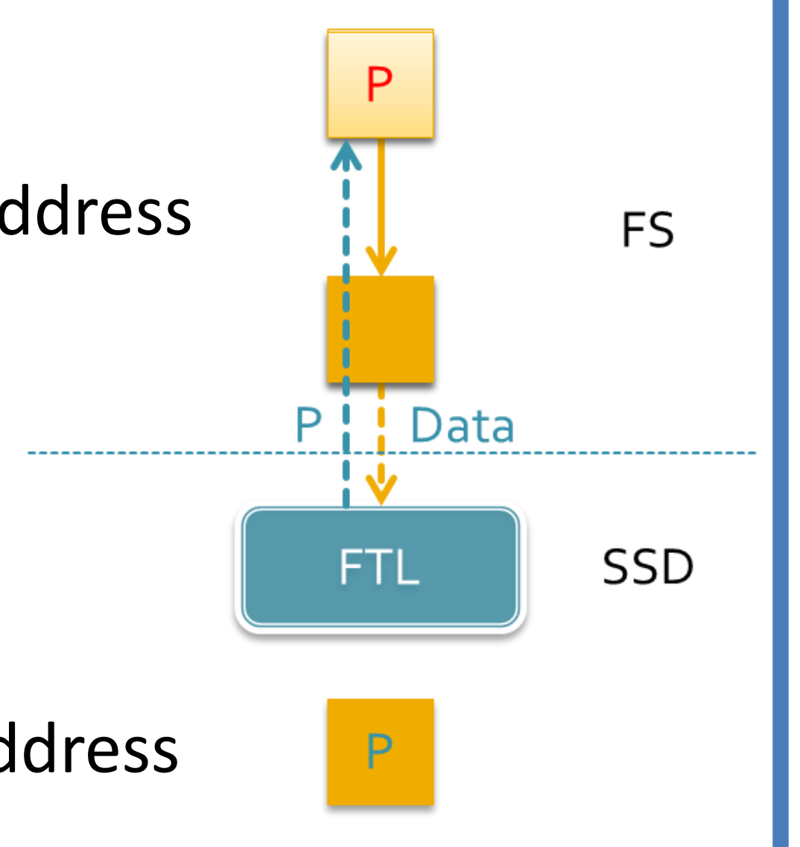
- Reads using physical address

Free/Trim

- Invalidates block at physical address

Virtual Write

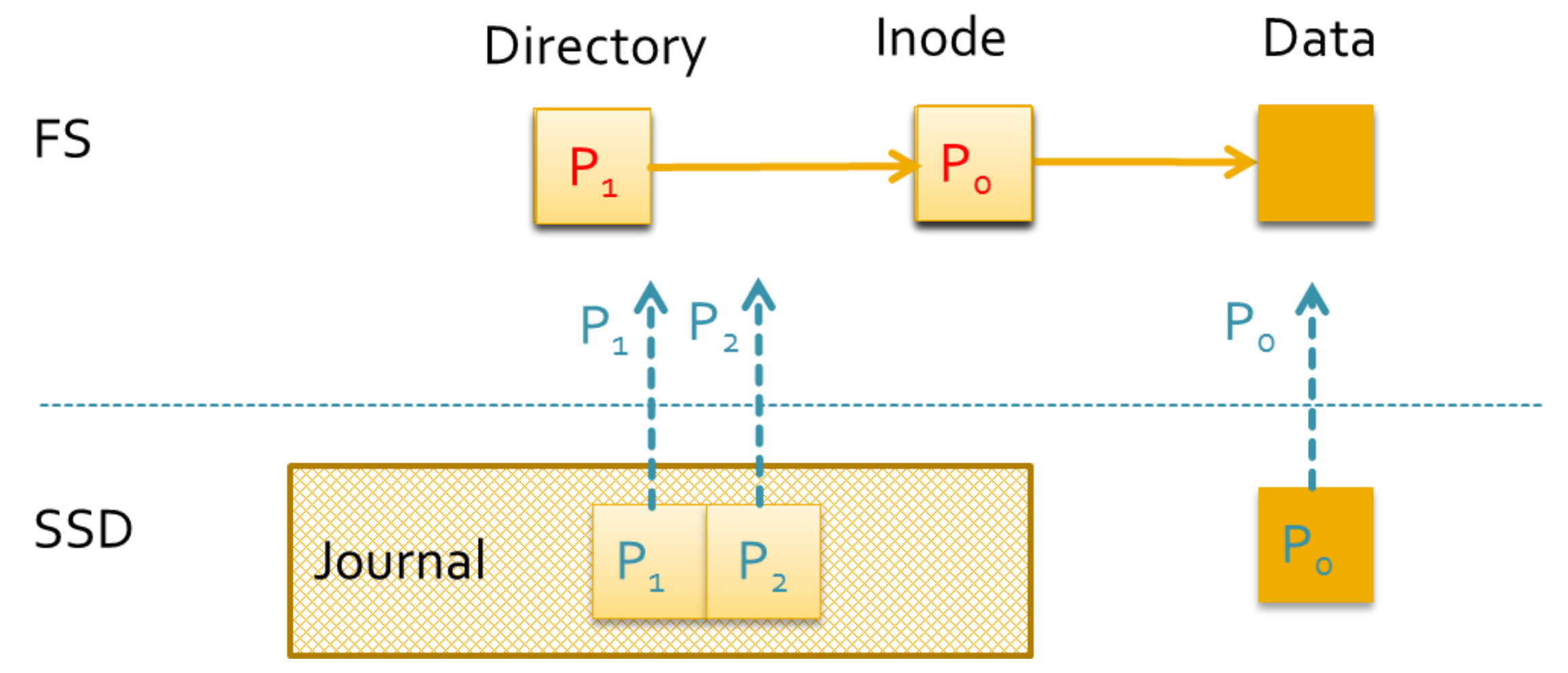
Virtual Read



Segmented Address Space

Problem: Recursive updates

- Writes propagate to reflect physical address
- Ordering needs to be enforced
- Multiple metadata writes for a data write



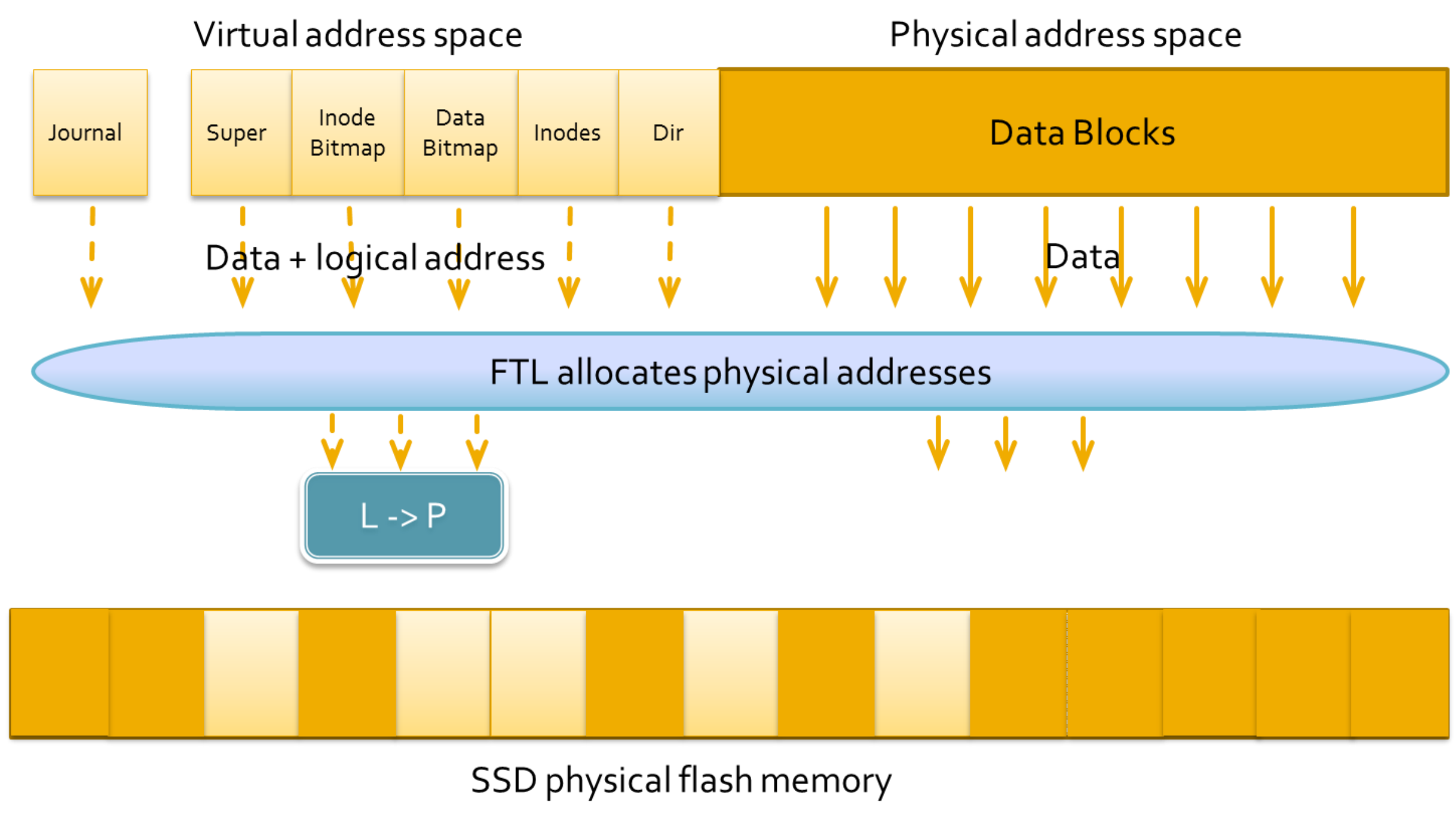
Solution: Two segments of address space

Physical address space

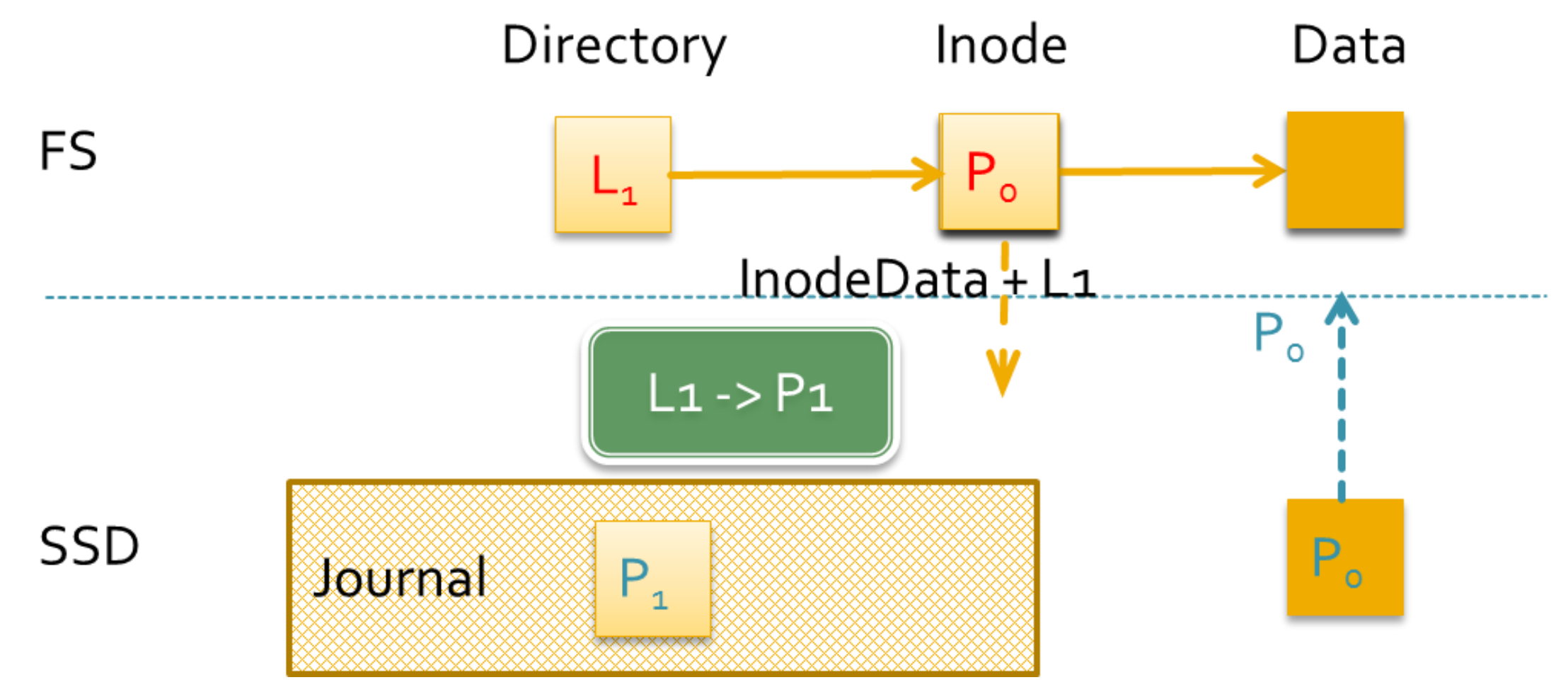
- Nameless write, physical read
- Contain data blocks

Virtual address space

- Traditional (virtual) read/write
- Small indirection table in device
- Contain metadata blocks (typically ~1% [2])



- One level of ordering writes
- Reduce additional metadata writes



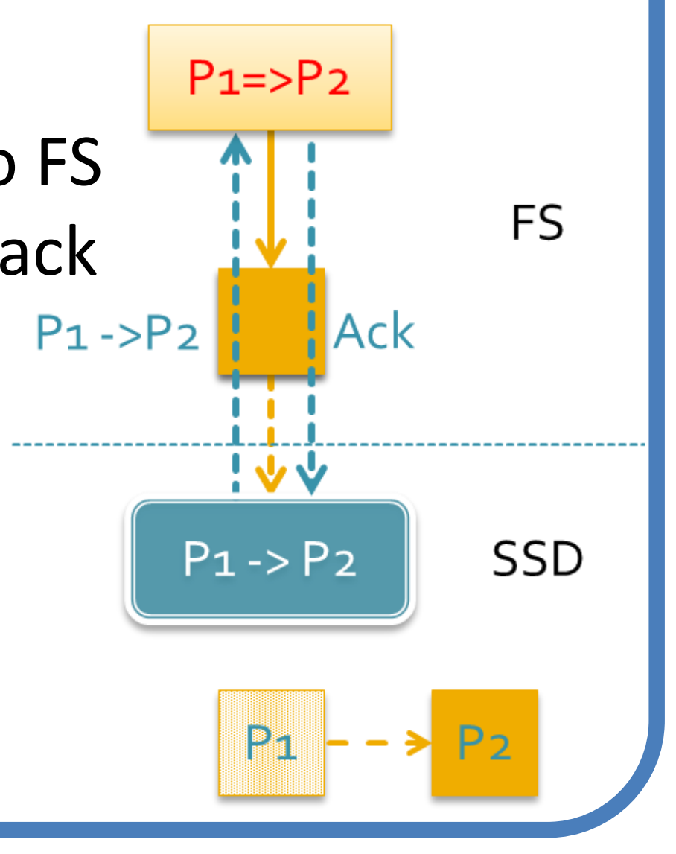
Migration Callbacks

Problem

- SSDs migrate physical pages because of wear leveling
- FS needs to be informed about physical address change

Solution: Migration Callbacks

- Device sends migration callbacks to FS
- Small remapping table during callback
- Reads and overwrites remapped
- FS acknowledges device
- Device removes remapping entries



Associated Metadata

Problem

- Locating metadata structures efficiently
- During callbacks and recovery
- Naive approach: traversing all metadata

Solution: Associated Metadata

- Small amount of metadata used to locate metadata
- E.g. Inode number, inode gen number, block offset
- Send with nameless writes and migration callbacks
- Stored adjacent to data pages on device

Building Nameless-Writing Device and Ext3

Nameless-writing SSD

- Nameless write interfaces support
- Flexible allocation
- Small indirection table
- Control of garbage collection and wear leveling

Nameless-writing ext3

- Ordered journaling mode
- Segmented address space
- Nameless write and physical read
- Free/trim
- Callback

Evaluation

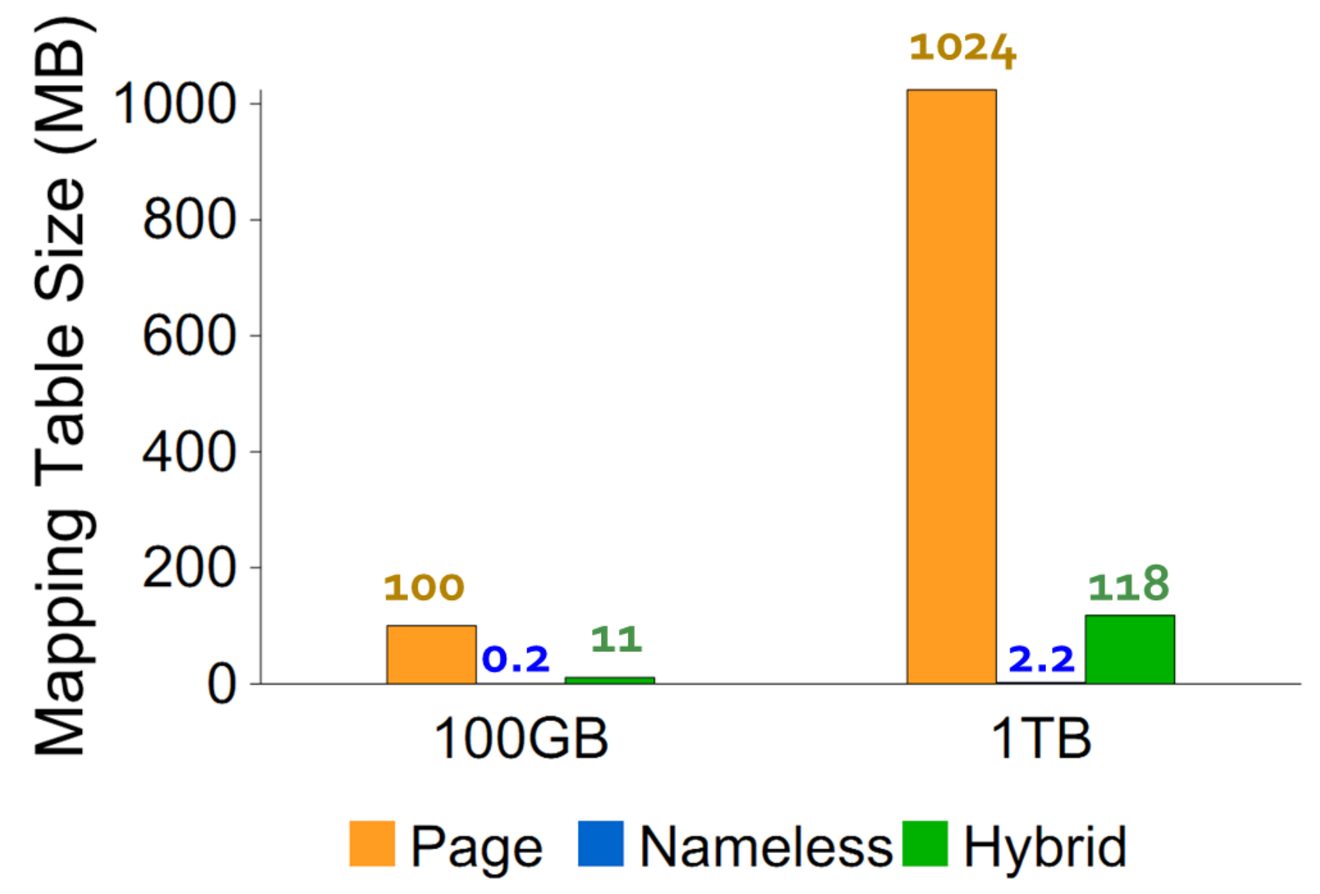
SSD emulator

- Linux pseudo block device
- Data stored in RAM

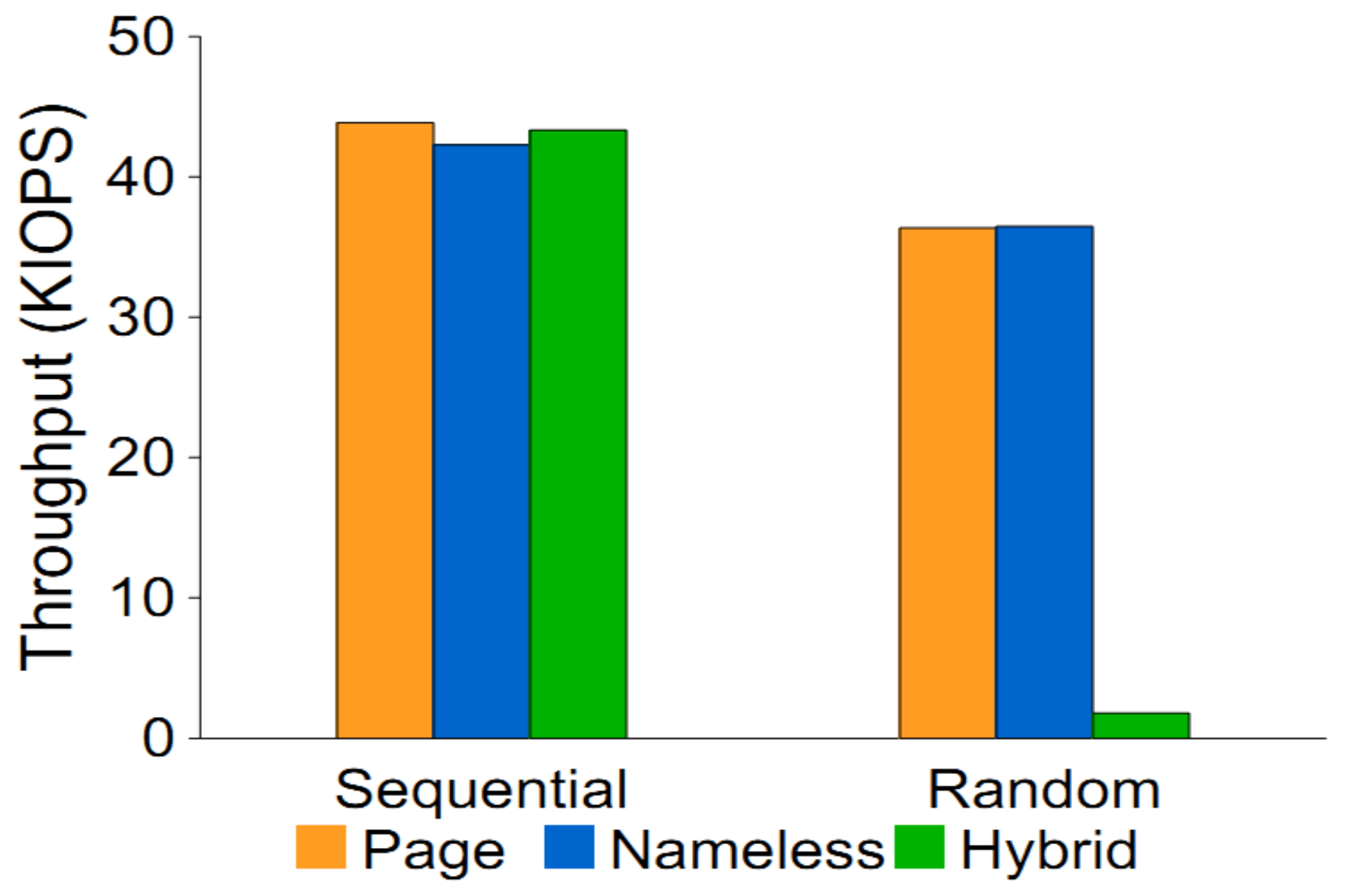
FTLs studied

- Page-level mapping: Performance upper bound
- Hybrid mapping: Models real SSDs
- Nameless-writing

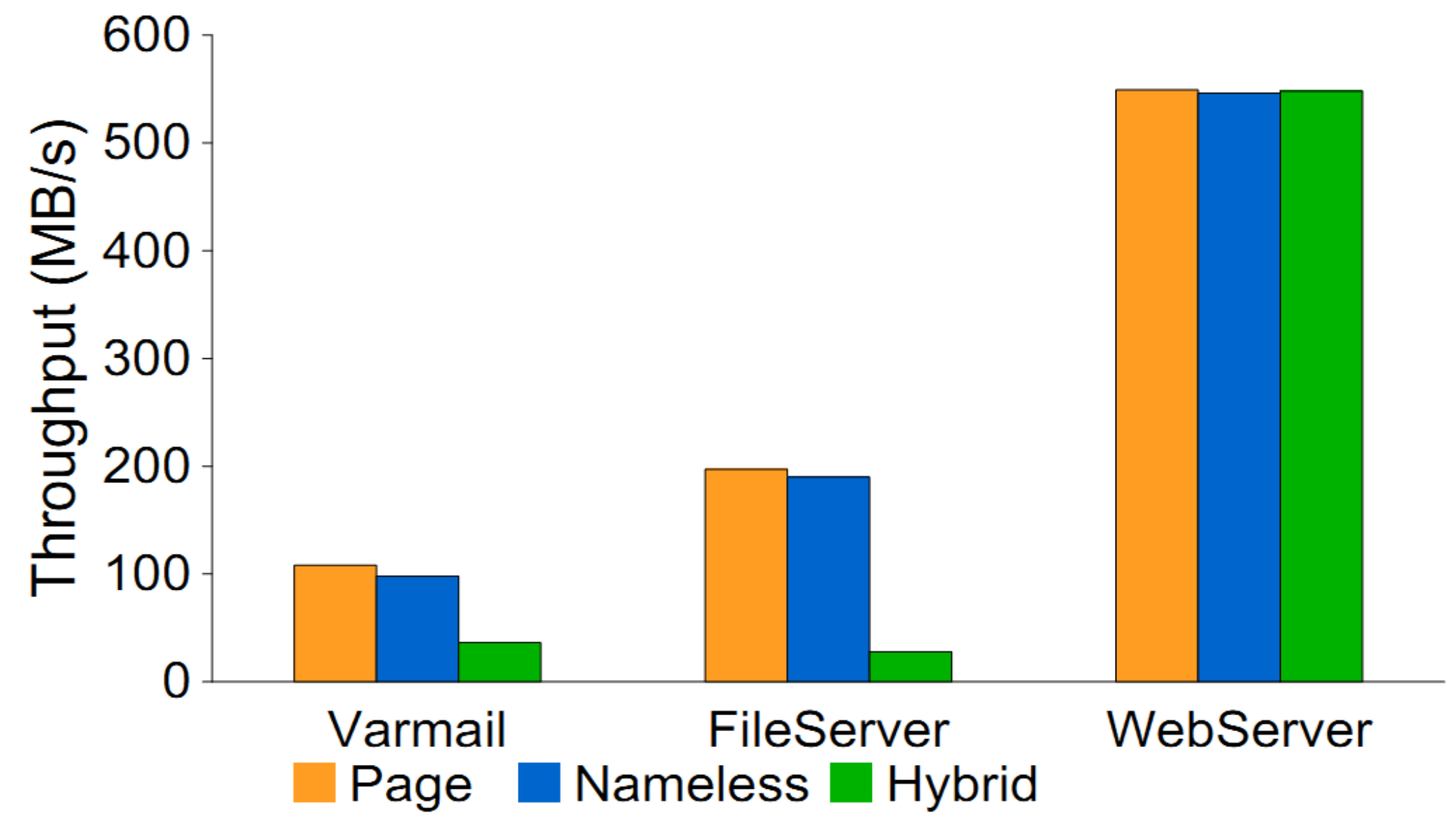
Mapping Table Space Cost



Micro-benchmarks



Macro-benchmarks



References

[1] M. Ben-Yehuda,M. D. Day, Z. Dubitzky,M. Factor, N. Har'El, A. Gordon, A. Liguori, O. Wasserman, and B.-A. Yassour. The Turtles Project: Design and Implementation of Nested Virtualization. OSDI '10. Vancouver, Canada, December 2010.

[2] N. Agrawal, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Generating Realistic Impressions for File-System Benchmarking. FAST '09, San Francisco, California, February 2009.