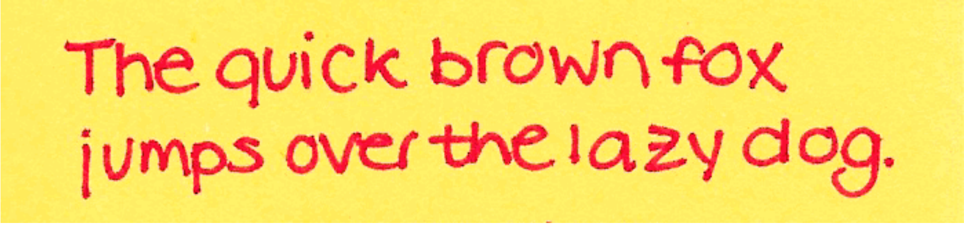
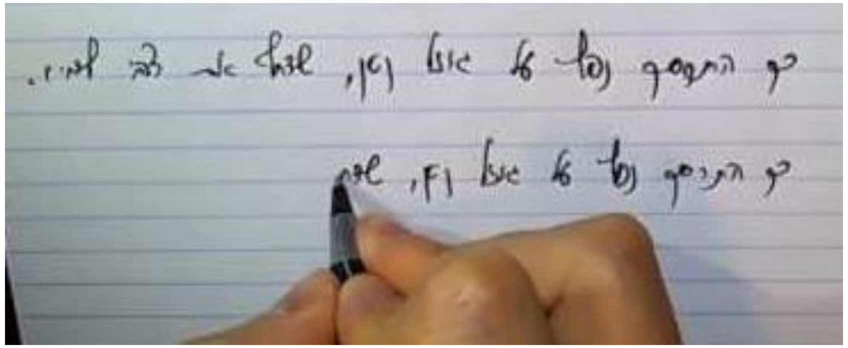
Who moved my cheese?
How storage systems deal with changes in their media

Gala Yadgar

TECHNION | The Henry and Marilyn Taub
Faculty of Computer Science

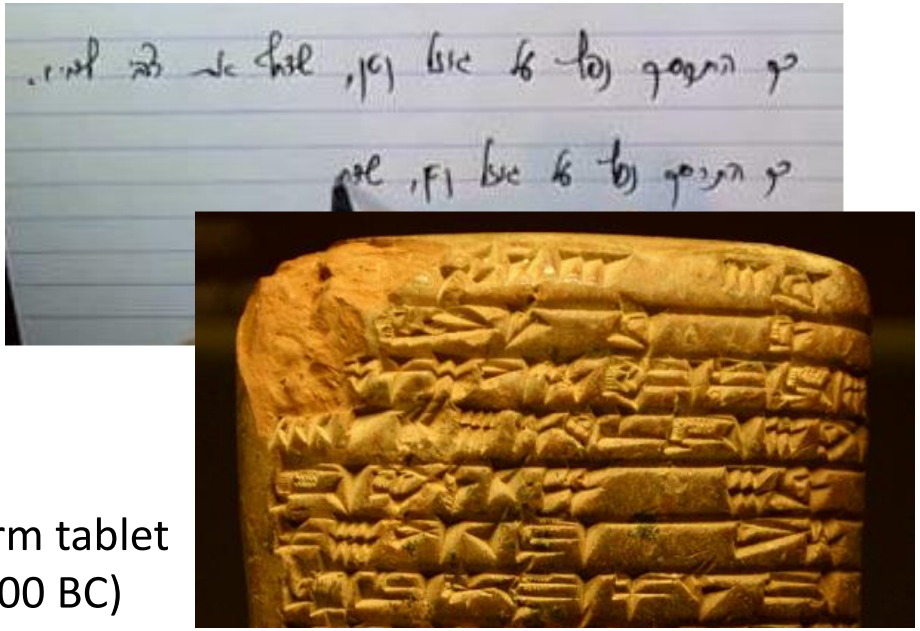
“Why is Hebrew written backwards?”





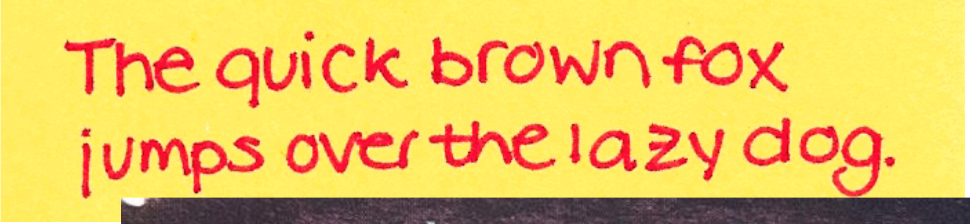
“Why is Hebrew written backwards?”

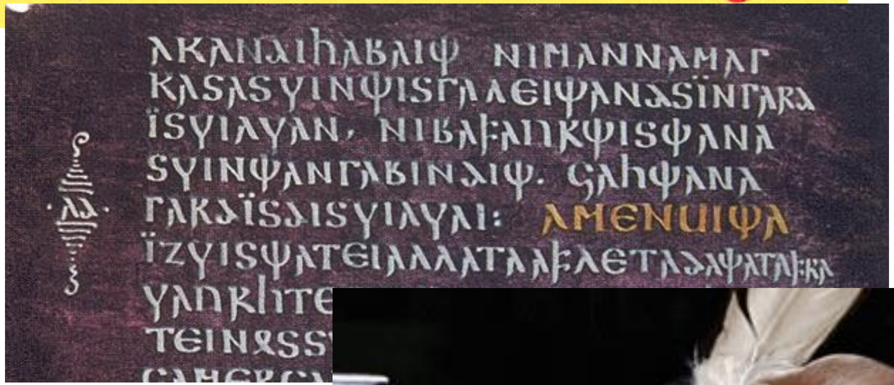
Codex Argenteus
(~500 AC)



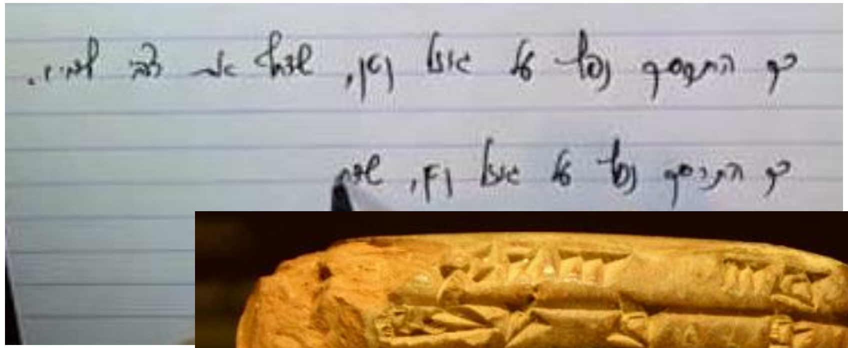
Cuneiform tablet
(~2000 BC)

“Why is Hebrew written backwards?”





Codex Argenteus
(~500 AC)

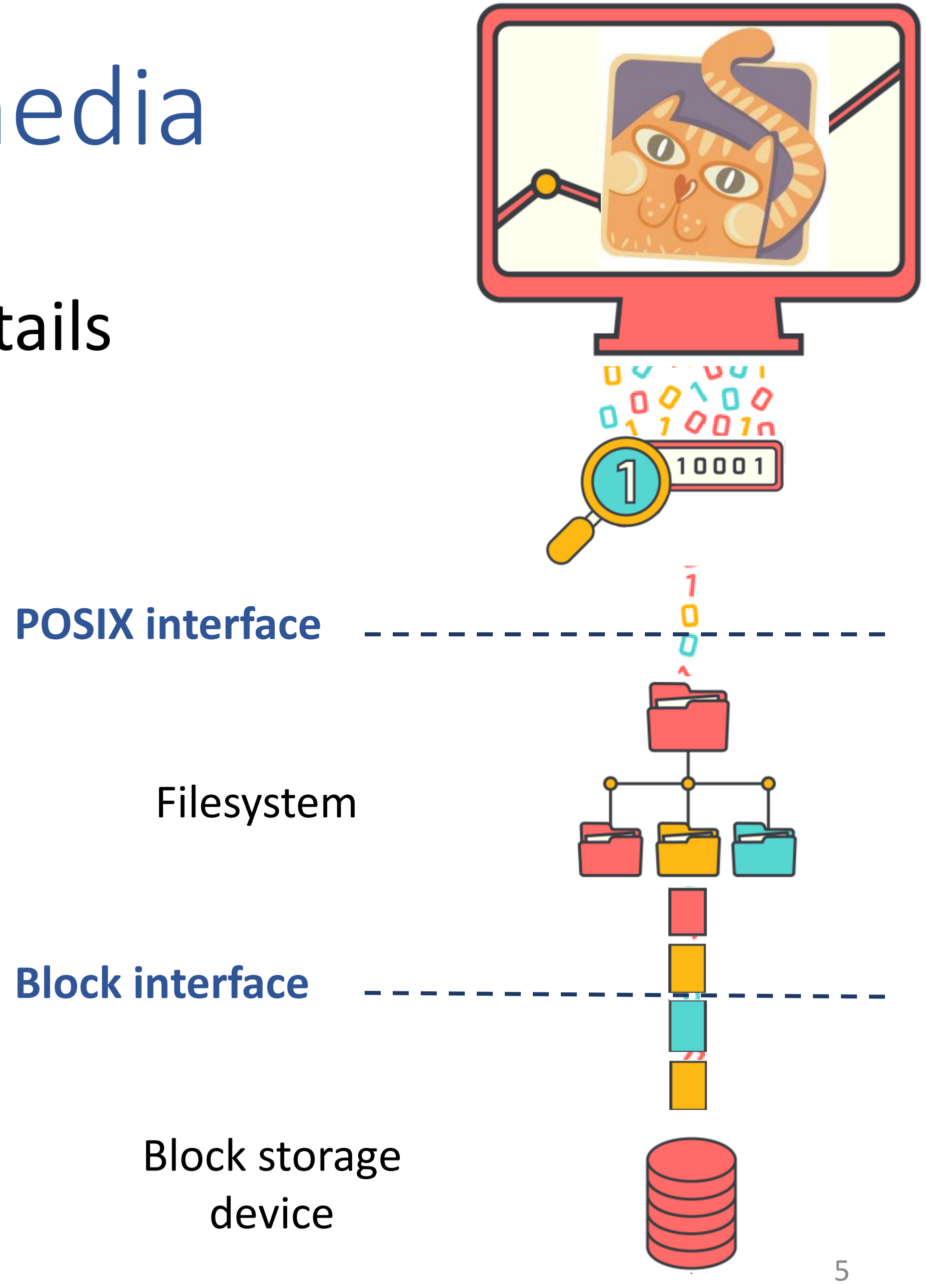


Cuneiform tablet
(~2000 BC)

It’s the storage media!

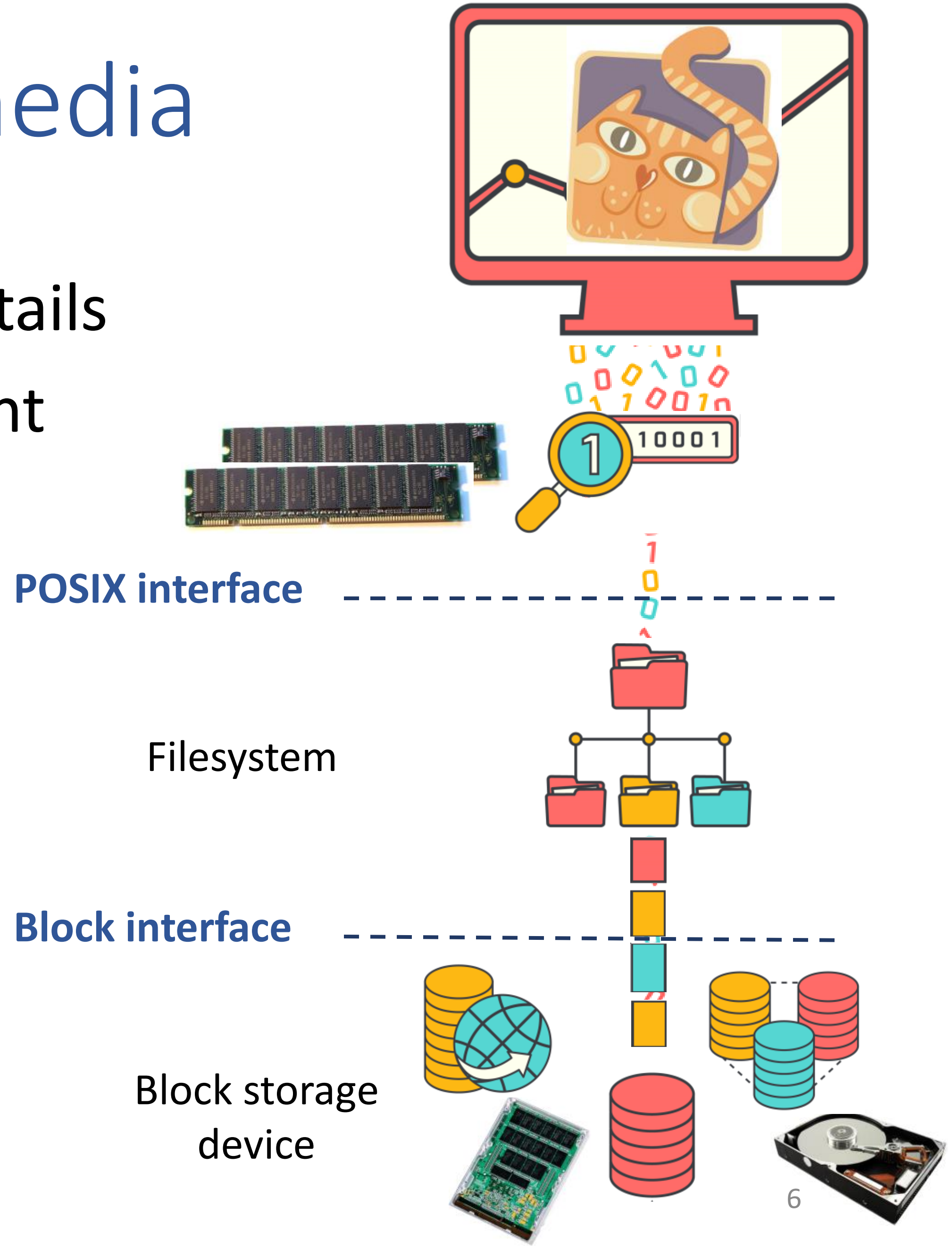
Storage systems and their media

- Abstraction layers hide complexity and details



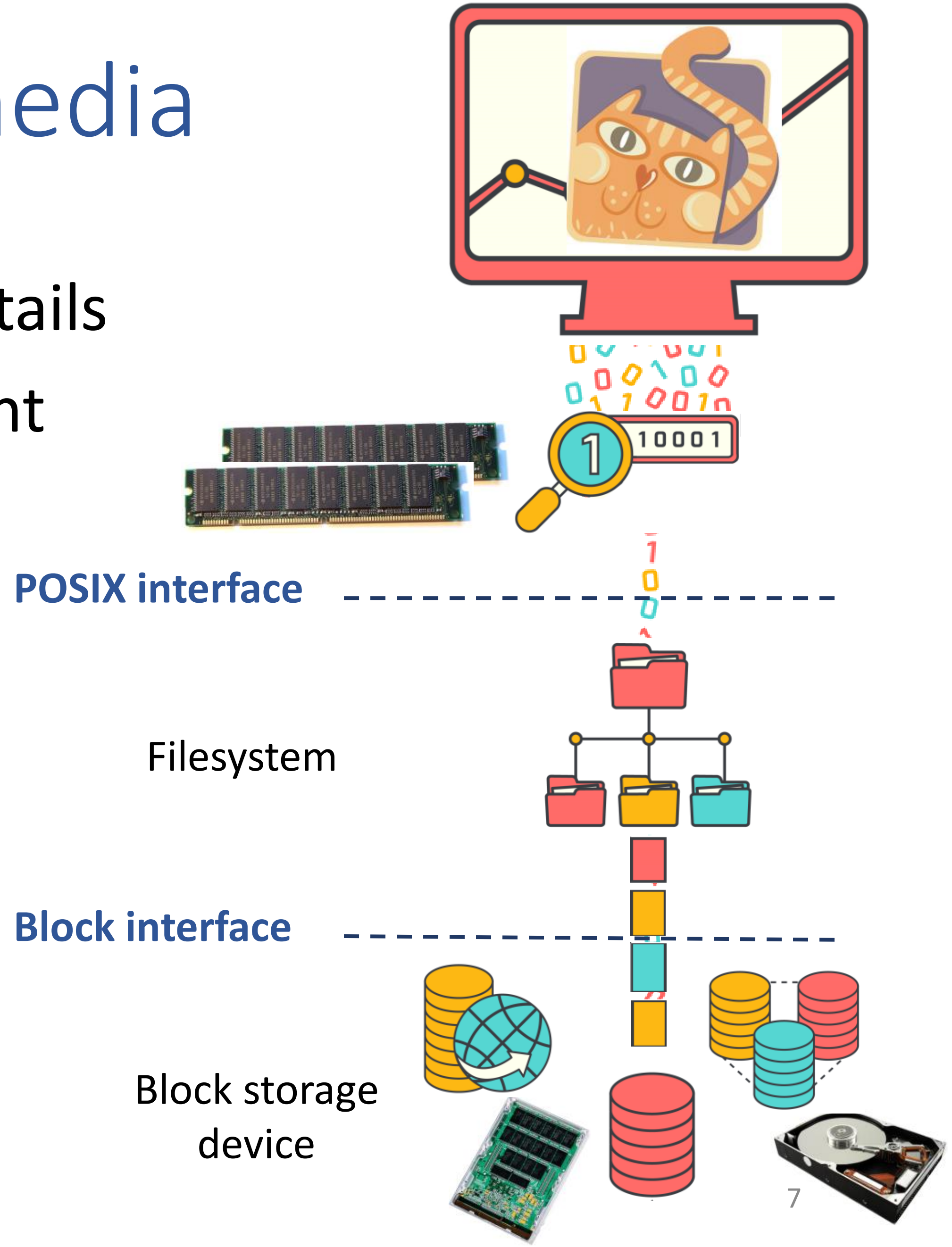
Storage systems and their media

- Abstraction layers hide complexity and details
- Add the new media as a direct replacement
  - Backwards compatibility
  - Little development effort
  - Quick adoption



Storage systems and their media

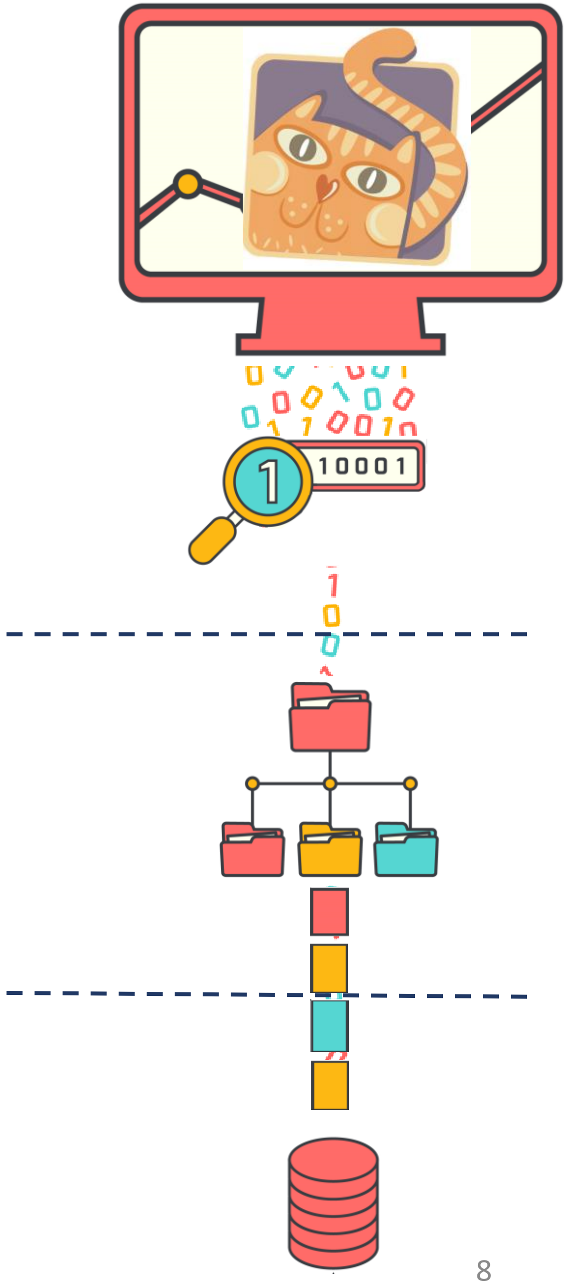
- Abstraction layers hide complexity and details
- Add the new media as a direct replacement
  - Backwards compatibility
  - Little development effort
  - Quick adoption
- It works! But…
  - Performance not as expected
  - Unpredictable user experience
- Notable examples from recent past
  - HDD → RAID → SSD → NAS/cloud storage



This talk

- How the storage-systems community addresses NVM
  - ⚠ Non-exhaustive, non-prioritized list of examples*
  - ⚠ Grossly over-simplified
    - Focus on insights and the adaptation process

- System model and characterization
- Unexpected bottlenecks
- Performance isolation / QoS



* Figures taken from respective cited papers and author presentations

What is NVM, exactly?

Non-volatile memory typically refers to storage in semiconductor memory chips, which store data in floating-gate memory cells consisting of floating-gate MOSFETs (metal–oxide–semiconductor field-effect transistors), including flash memory storage such as NAND flash and solid-state drives (SSD).

Other examples of non-volatile memory include read-only memory (ROM), EPROM (erasable programmable ROM) and EEPROM (electrically erasable programmable ROM), ferroelectric RAM, most types of computer data storage devices (e.g. disk storage, hard disk drives, optical discs, floppy disks, and magnetic tape), and early computer storage methods such as punched tape and cards.[1]

WIKIPEDIA
The Free Encyclopedia

What is NVM, exactly?

Non-volatile memory typically refers to storage in semiconductor memory chips, which store data in floating-gate memory cells consisting of floating-gate MOSFETs (metal–oxide–semiconductor field-effect transistors), including flash memory storage such as NAND flash and solid-state drives (SSD).

Other examples of non-volatile memory include read-only memory (ROM), EPROM (erasable programmable ROM) and EEPROM (electrically erasable programmable ROM), ferroelectric RAM, most types of computer data storage devices (e.g. disk storage, hard disk drives, optical discs, floppy disks, and magnetic tape), and early computer storage methods such as punched tape and cards.[1]

Wikipedia
The Free Encyclopedia

- For many purposes, *fast* and *non-volatile* is NVM
  - Write-ahead logging
  - Metadata or index

What is NVM, exactly?

Non-volatile memory typically refers to storage in semiconductor memory chips, which store data in floating-gate memory cells consisting of floating-gate MOSFETs (metal–oxide–semiconductor field-effect transistors), including flash memory storage such as NAND flash and solid-state drives (SSD).

Other examples of non-volatile memory include read-only memory (ROM), EPROM (erasable programmable ROM) and EEPROM (electrically erasable programmable ROM), ferroelectric RAM, most types of computer data storage devices (e.g. disk storage, hard disk drives, optical discs, floppy disks, and magnetic tape), and early computer storage methods such as punched tape and cards.[1]

WIKIPEDIA
The Free Encyclopedia

- For many purposes, *fast* and *non-volatile* is NVM
  - Write-ahead logging
  - Metadata or index
- For our purpose, *byte-addresable* and *non-DRAM*

What is NVM, exactly?

- Intel Optane persistent memory

- For our purpose, *byte-addresable* and *non-DRAM*

What is NVM, exactly?

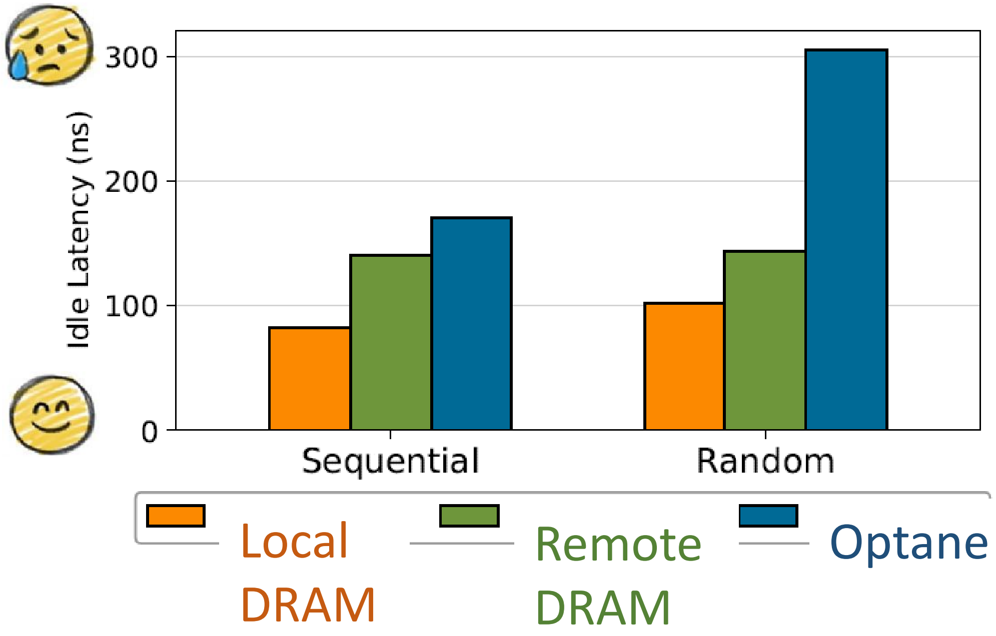
- Intel Optane persistent memory

- [Samsung's CMM-H (CXL Memory Module – Hybrid)]

- For our purpose, *byte-addresable* and *non-DRAM*

Characterization (Optane)

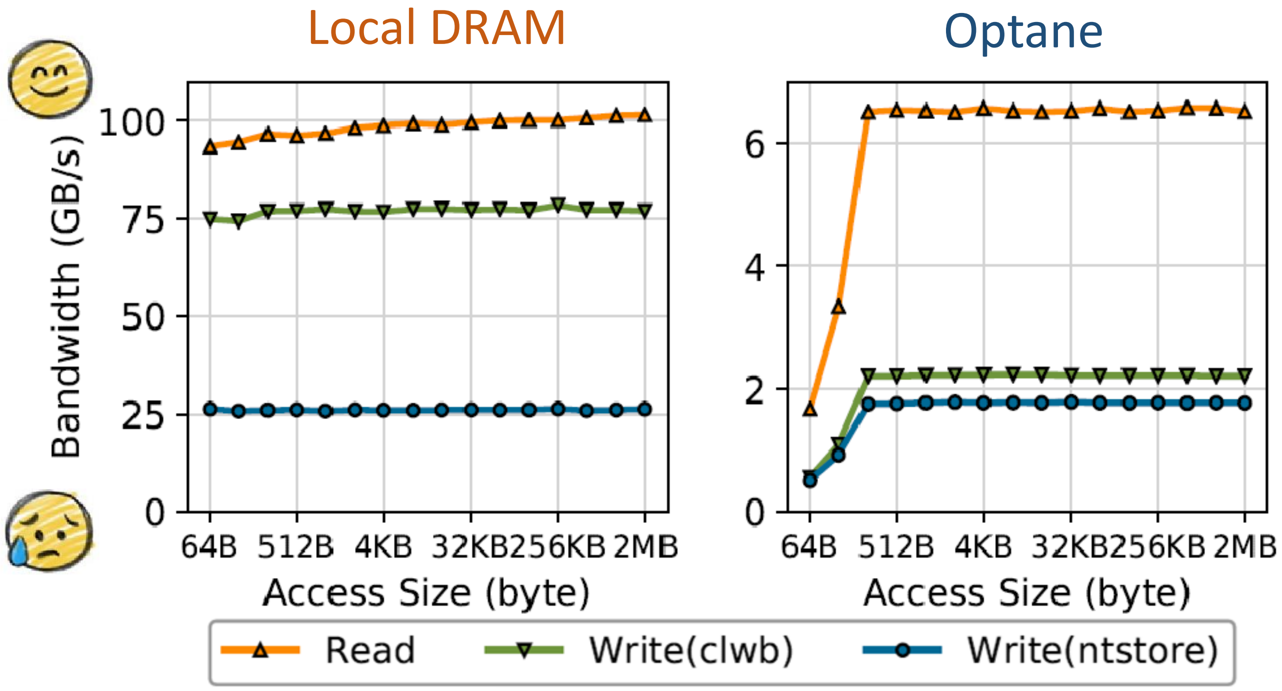
- Faster than HDD and SSD but slower than DRAM
- Does not behave like DRAM
  - Random ≠ Sequential



J. Izraelevitz et al. [Basic Performance Measurements of the Intel Optane DC Persistent Memory Module](), arXiv:1903.05714v3 2019

Characterization (Optane)

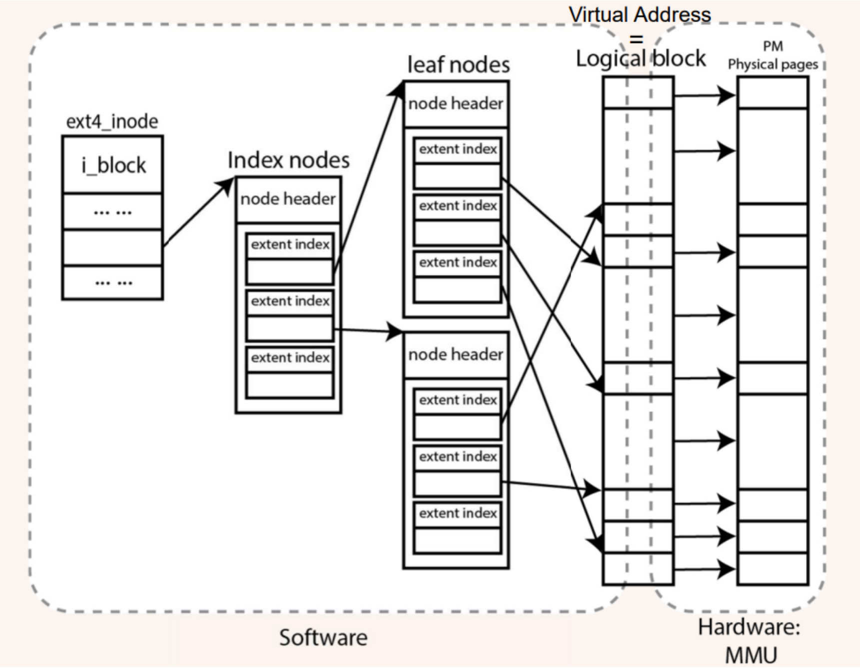
- Faster than HDD and SSD but slower than DRAM
- Does not behave like DRAM
  - Random ≠ Sequential
  - Reads ≠ Writes
  - Small ≠ Large
  - Much lower bandwidth
  - (Much more interference)



J. Izraelevitz et al. Basic Performance Measurements of the Intel Optane DC Persistent Memory Module, arXiv:1903.05714v3 2019

Rethinking file-system indexing

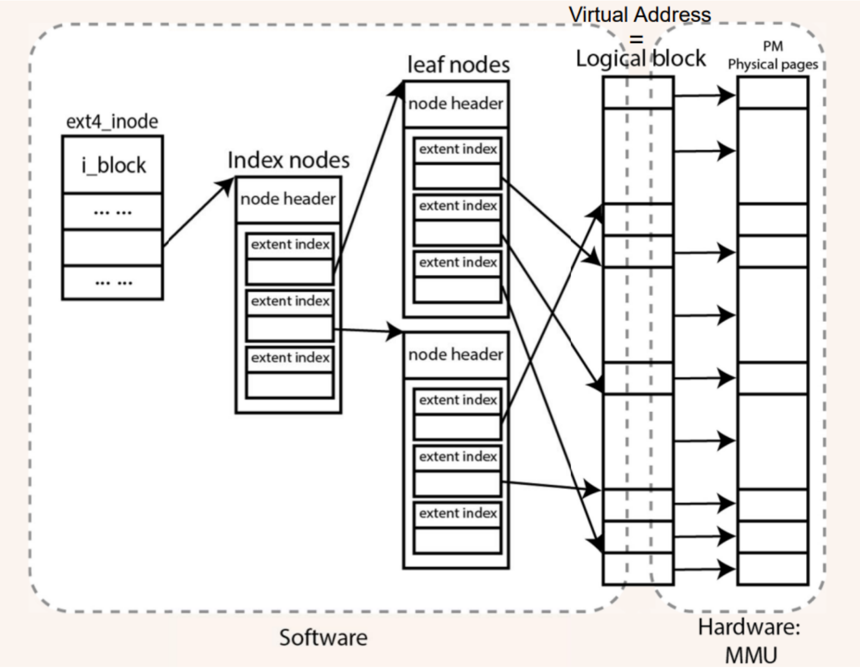
- Bottleneck: *tree-based inode index* incurs high overhead with NVM
  - Up to 63% of the time spend on FS operations (e.g., file append)



R. Li et al. ctFS: Replacing File Indexing with Hardware Memory Translation through Contiguous File Allocation for Persistent Memory, USENIX FAST 2022

Rethinking file-system indexing

- Bottleneck: *tree-based inode index* incurs high overhead with NVM
  - Up to 63% of the time spend on FS operations (e.g., file append)
- Insights:
  - Use hardware-based translation
  - Don't care about physical contiguity
  - Persist small updates quickly

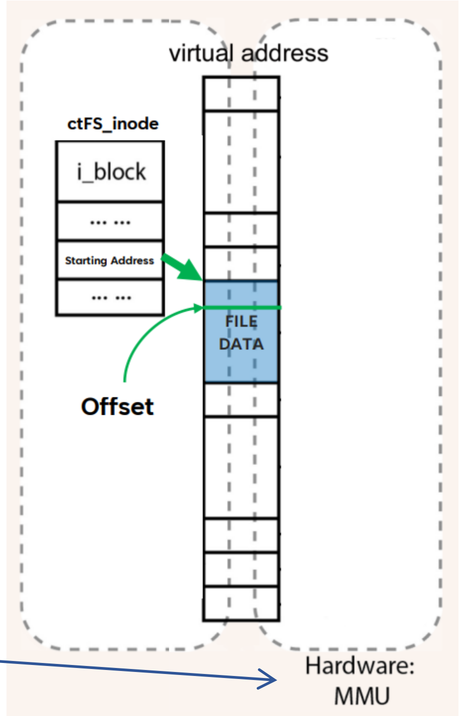


R. Li et al. [ctFS: Replacing File Indexing with Hardware Memory Translation through Contiguous File Allocation for Persistent Memory](), USENIX FAST 2022

Rethinking file-system indexing: ctFS

- Use hardware-based translation
- Don't care about physical contiguity

inode only records start of segment



Kernel module populates DRAM page table

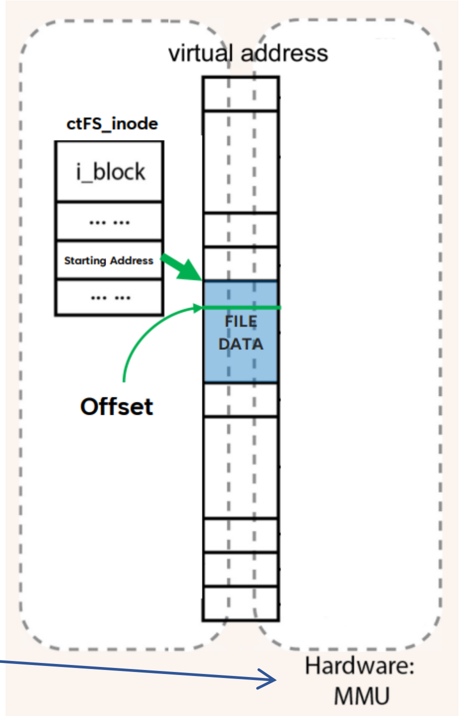
R. Li et al. [ctFS: Replacing File Indexing with Hardware Memory Translation through Contiguous File Allocation for Persistent Memory](), USENIX FAST 2022

Rethinking file-system indexing: ctFS

- Use hardware-based translation
- Don’t care about physical contiguity
- Persist small updates quickly

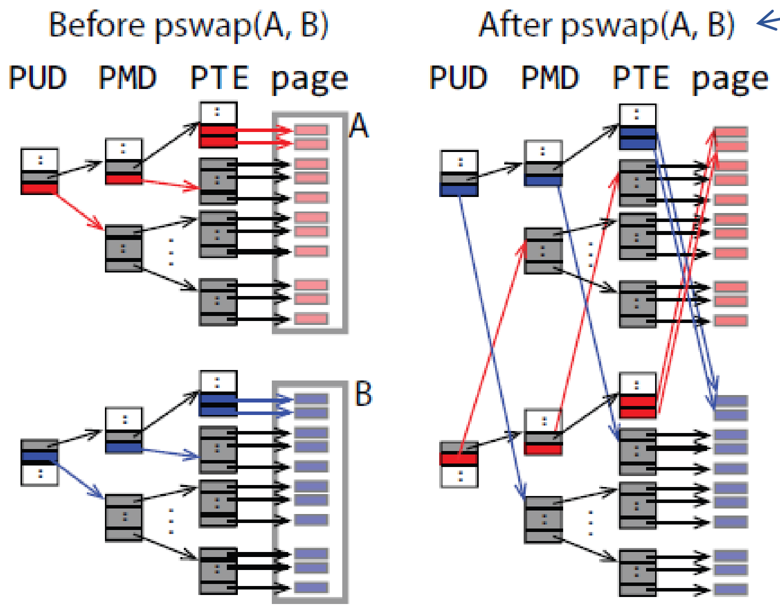
inode only records start of segment

Kernel module populates DRAM page table



New syscall: atomic pswap ()

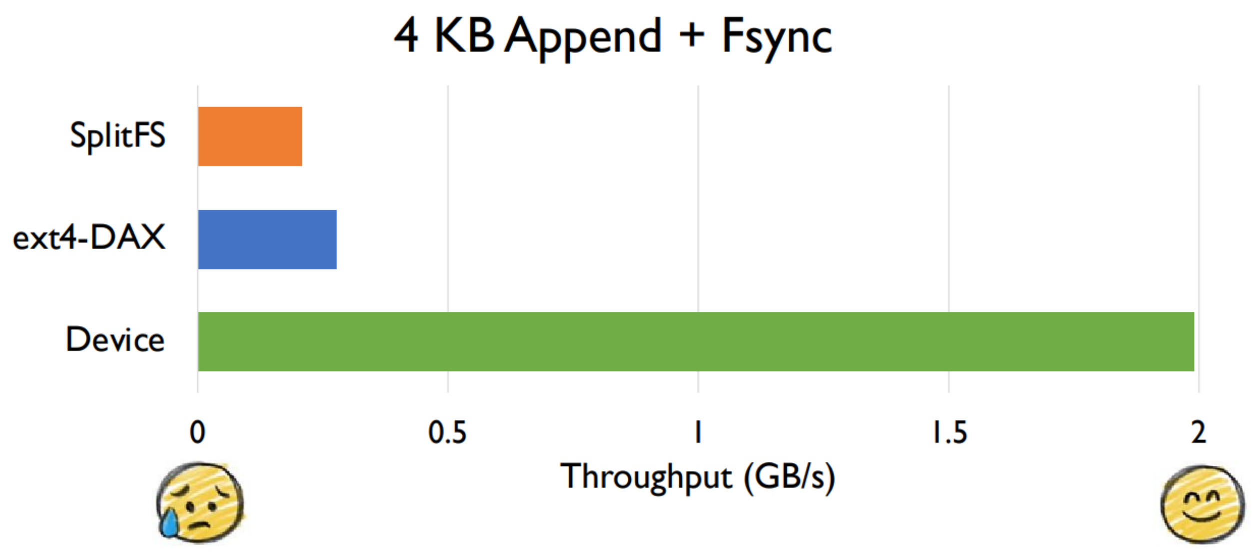
Hierarchical partitions



R. Li et al. [ctFS: Replacing File Indexing with Hardware Memory Translation through Contiguous File Allocation for Persistent Memory], USENIX FAST 2022

Rethinking kernel-space vs. user-space

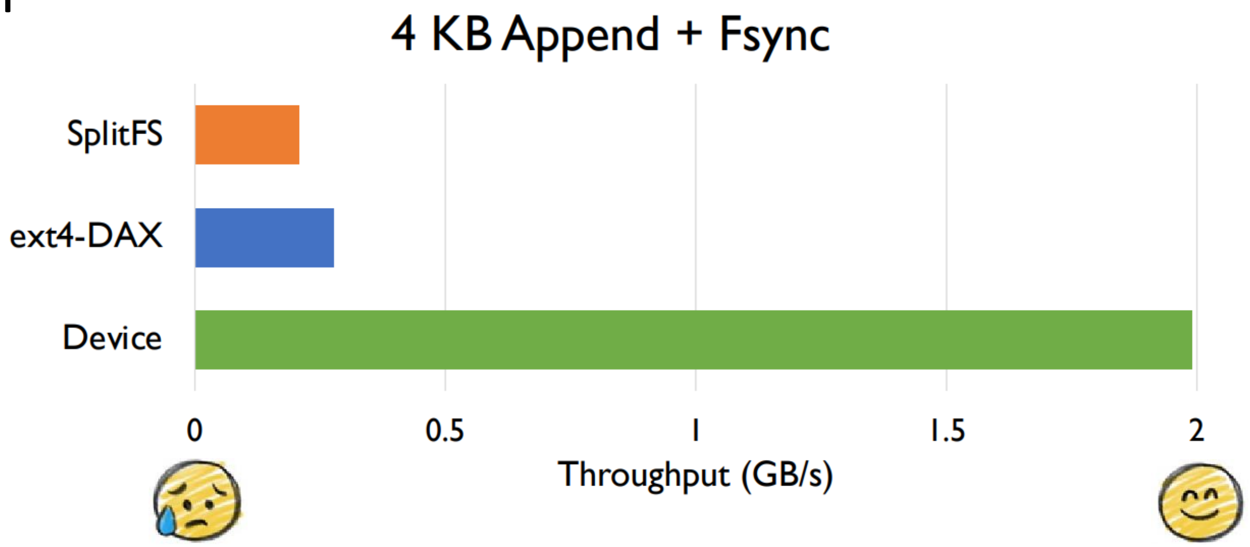
- **Bottleneck**: metadata handling in *kernel space* is inefficient



S. Zhong et al. [MadFS: Per-File Virtualization for Userspace Persistent Memory Filesystems] USENIX FAST 2023

Rethinking kernel-space vs. user-space

- Bottleneck: metadata handling in *kernel space* is inefficient
- Insights:
  - Some metadata can be safely handled in user space
  - Rethink data/metadata separation

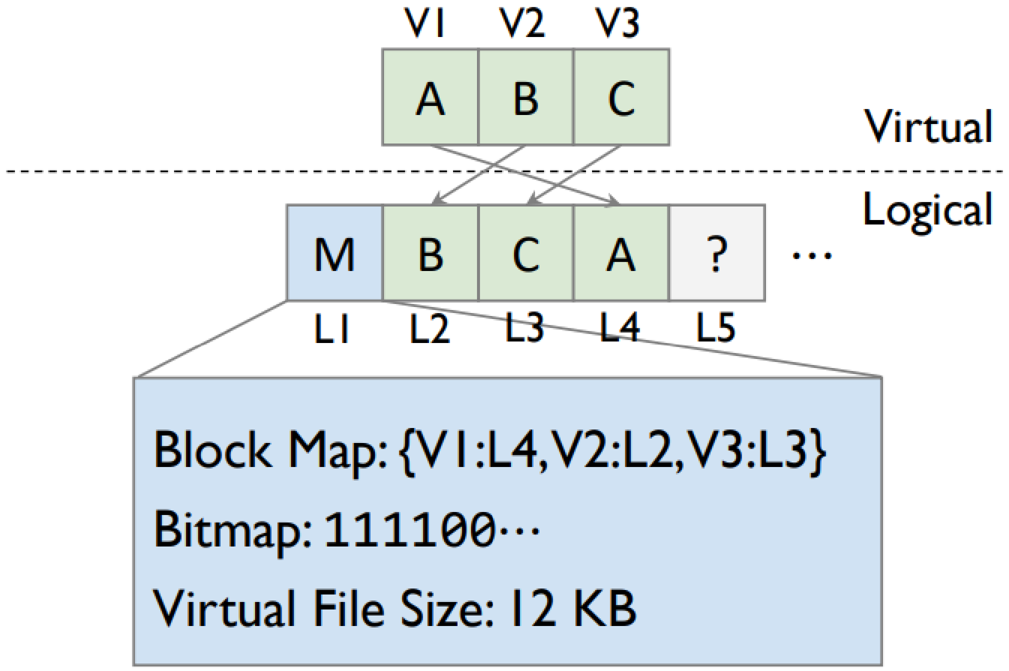


S. Zhong et al. [MadFS: Per-File Virtualization for Userspace Persistent Memory Filesystems]() USENIX FAST 2023

Rethinking kernel-space vs. user-space: MadFS

- Some metadata can be safely handled in user space
- Rethink data/metadata separation

- Embedded metadata
  - "Virtual" file block map and size

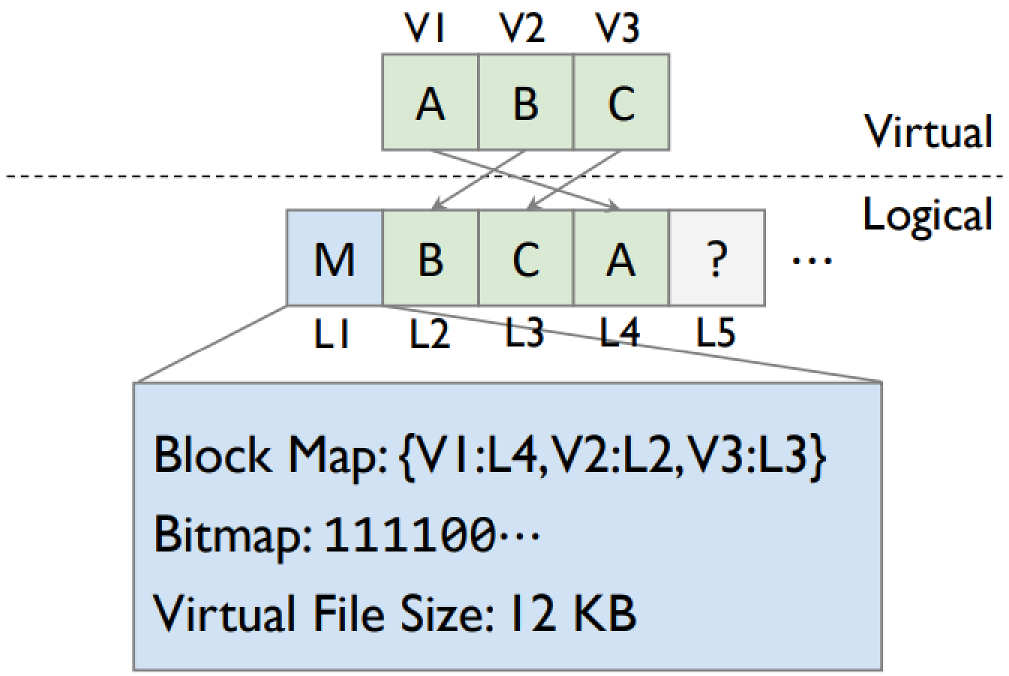


S. Zhong et al. [MadFS: Per-File Virtualization for Userspace Persistent Memory Filesystems]() USENIX FAST 2023

Rethinking kernel-space vs. user-space: MadFS

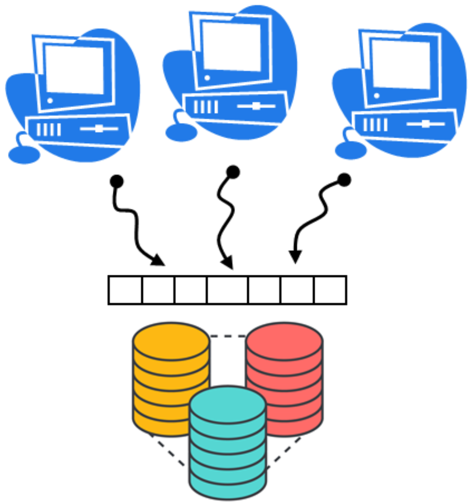
- Some metadata can be safely handled in user space
- Rethink data/metadata separation

- Embedded metadata
  - "Virtual" file block map and size
- Kernel-managed metadata
  - Logical-to-physical mapping
  - File permissions
  - Directory structures



S. Zhong et al. [MadFS: Per-File Virtualization for Userspace Persistent Memory Filesystems] USENIX FAST 2023

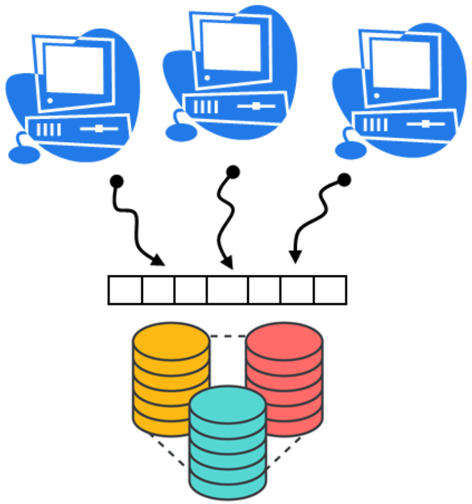
Rethinking shared caches



- DRAM:
  - More space allocation = higher hit rate = better performance
  - Higher bandwidth = more usage = more interference
  - Tenant A is too slow → throttle noisy neighbor B with max bandwidth

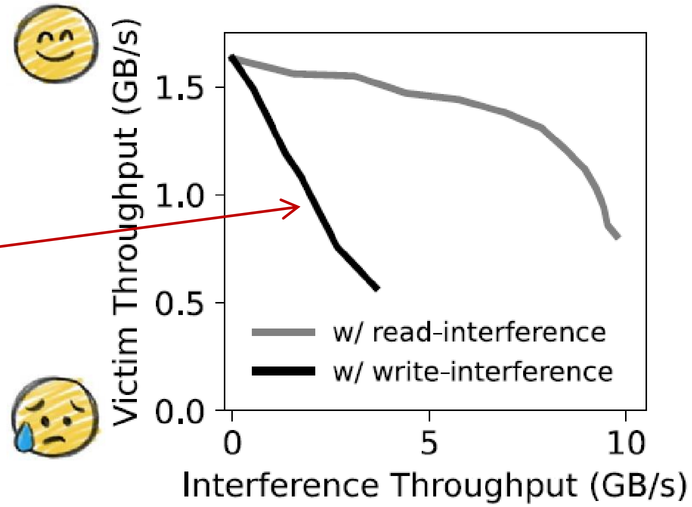
K. Wu at al. [NyxCache: Flexible and Efficient Multi-tenant Persistent Memory Caching] USENIX FAST 2022

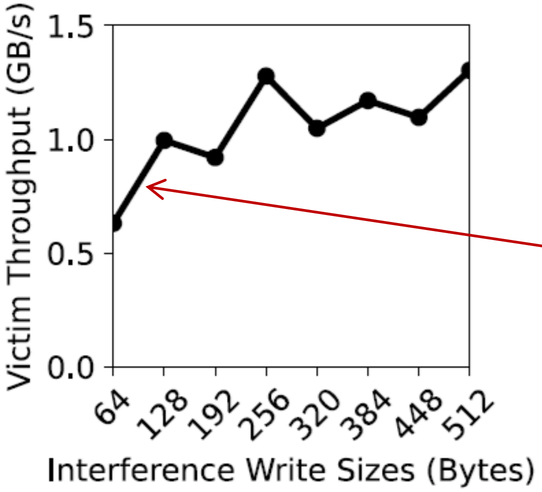
Rethinking shared caches



- DRAM:
  - More space allocation = higher hit rate = better performance
  - Higher bandwidth = more usage = more interference
  - Tenant A is too slow → throttle noisy neighbor B with max bandwidth
- NVM:

Writes interfere more than reads





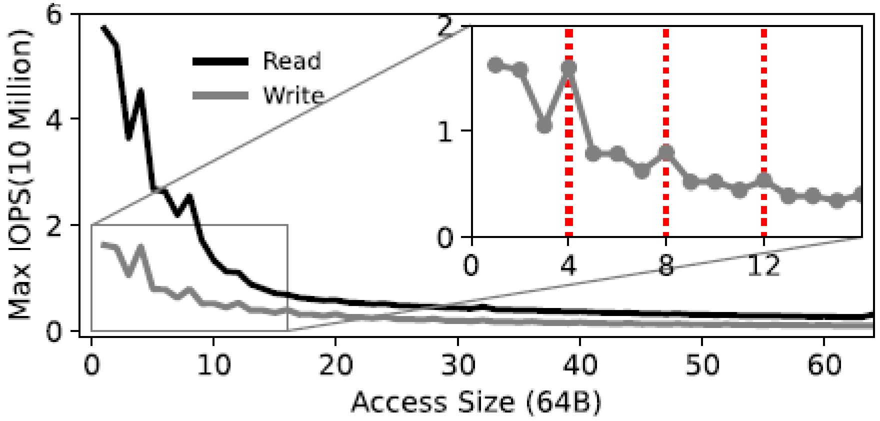
Small requests interfere more than large ones

K. Wu at al. [NyxCache: Flexible and Efficient Multi-tenant Persistent Memory Caching]() USENIX FAST 2022

Rethinking shared caches: NyxCache

- Profile max performance of different access types

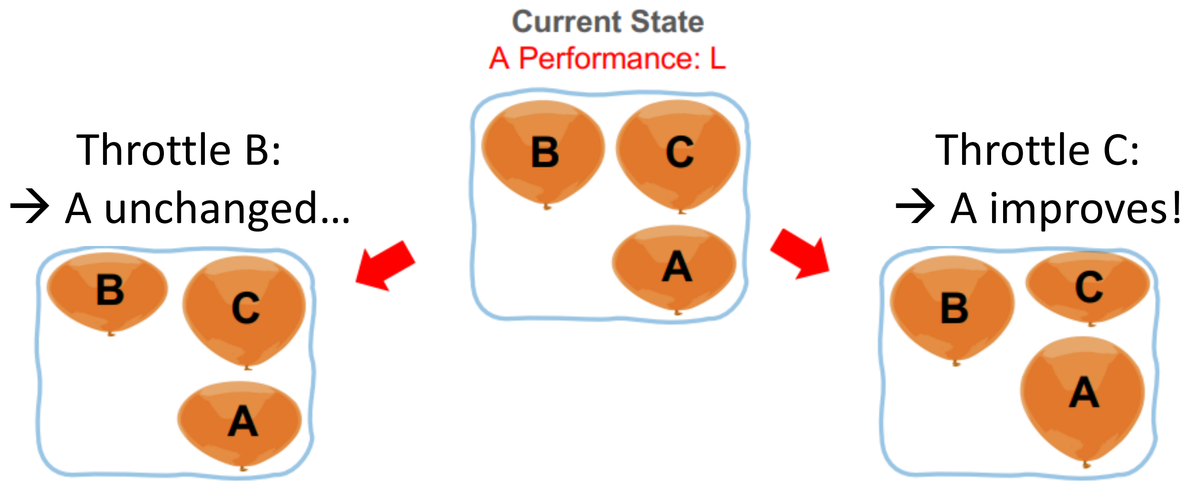
Max IOPS for random reads /writes



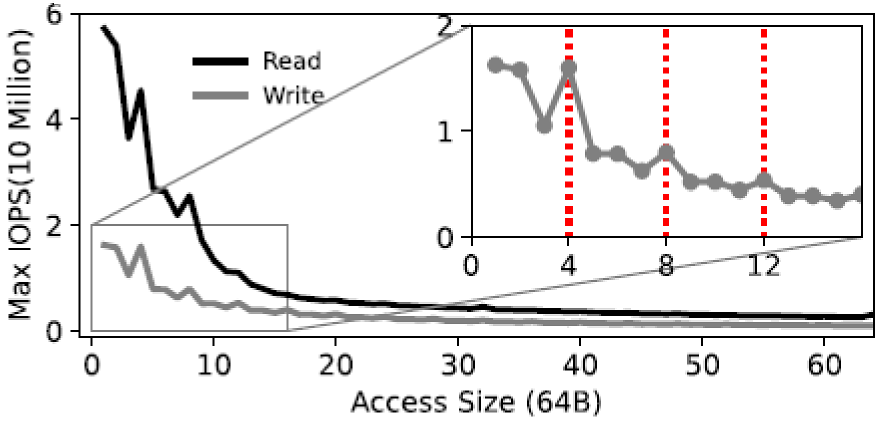
K. Wu at al. NyxCache: Flexible and Efficient Multi-tenant Persistent Memory Caching USENIX FAST 2022

Rethinking shared caches: NyxCache

- Profile max performance of different access types
- Monitor individual tenants and their interaction



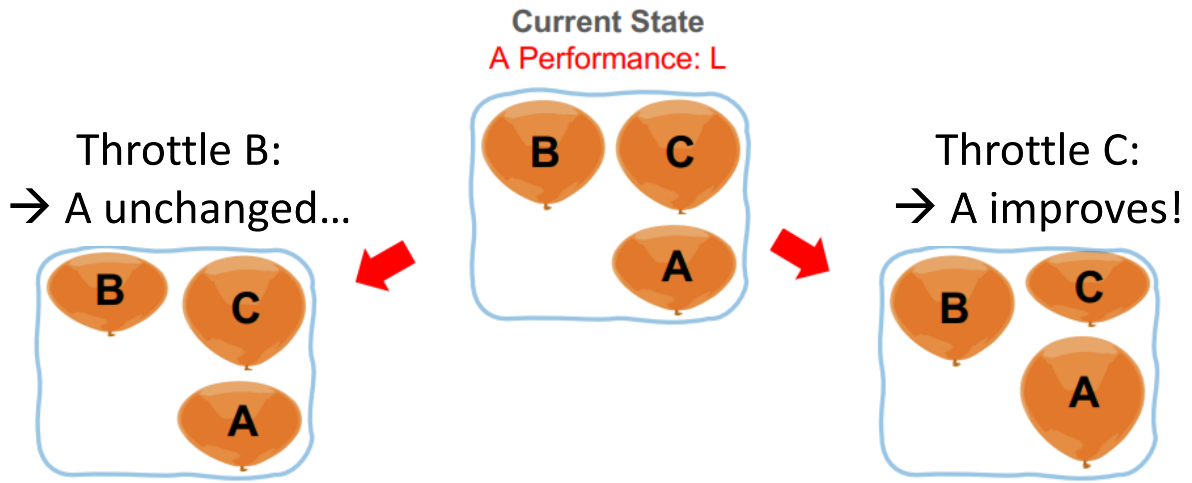
Max IOPS for random reads /writes



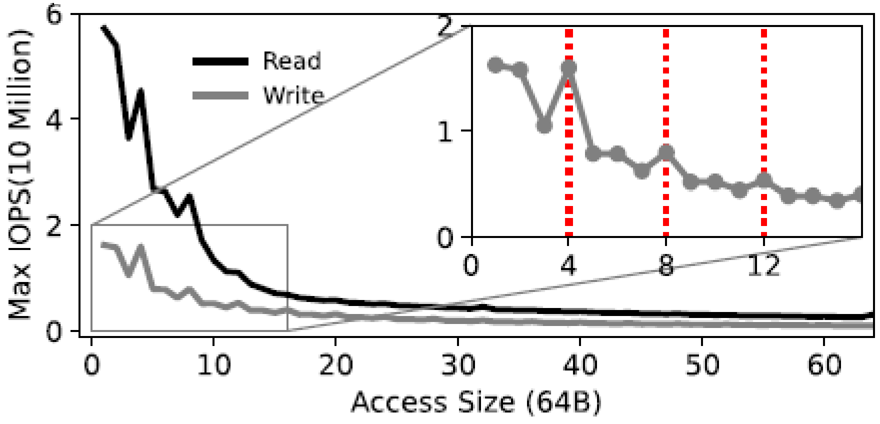
K. Wu at al. [NyxCache: Flexible and Efficient Multi-tenant Persistent Memory Caching] USENIX FAST 2022

Rethinking shared caches: NyxCache

- Profile max performance of different access types
- Monitor individual tenants and their interaction
- Library throttles tenant using most resources / causing most interference

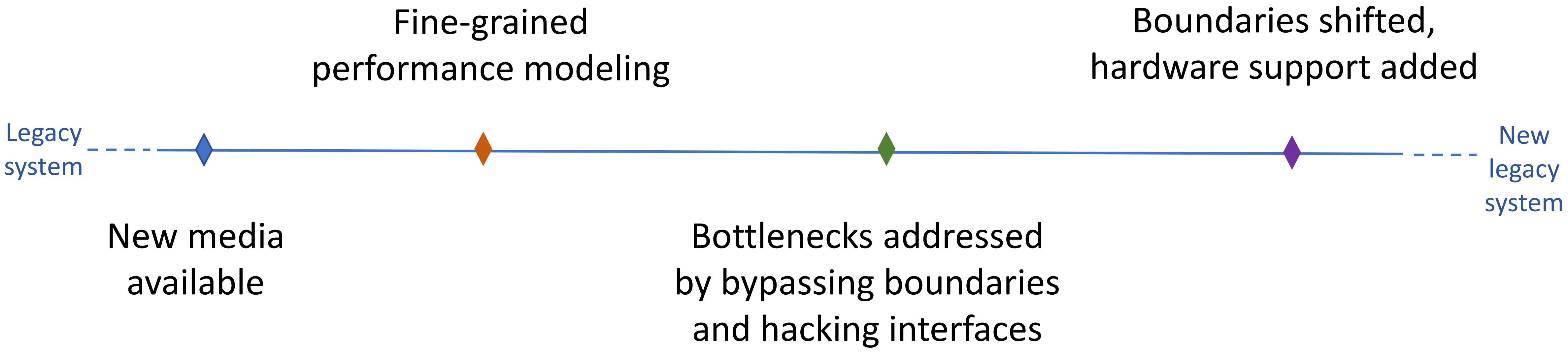


Max IOPS for random reads /writes

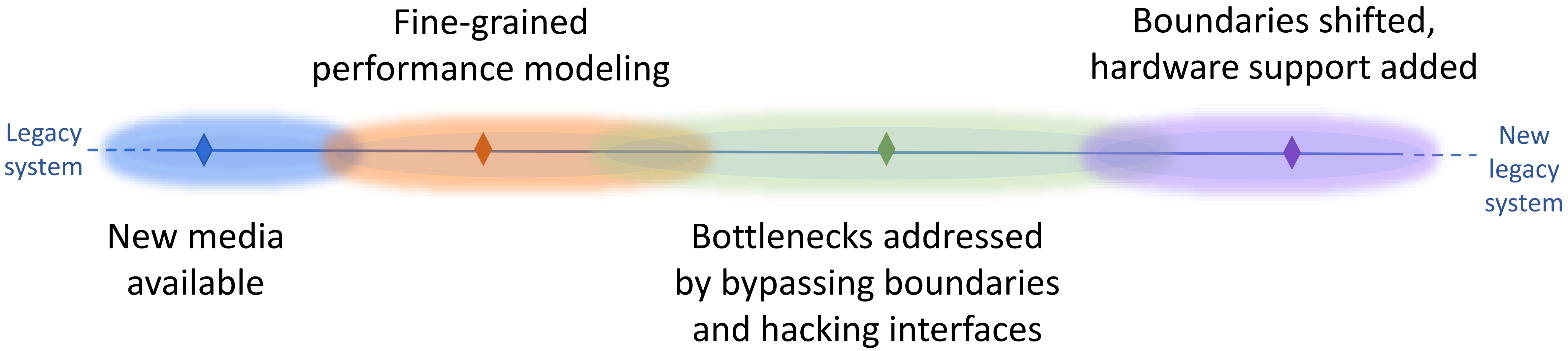


K. Wu at al. [NyxCache: Flexible and Efficient Multi-tenant Persistent Memory Caching]() USENIX FAST 2022

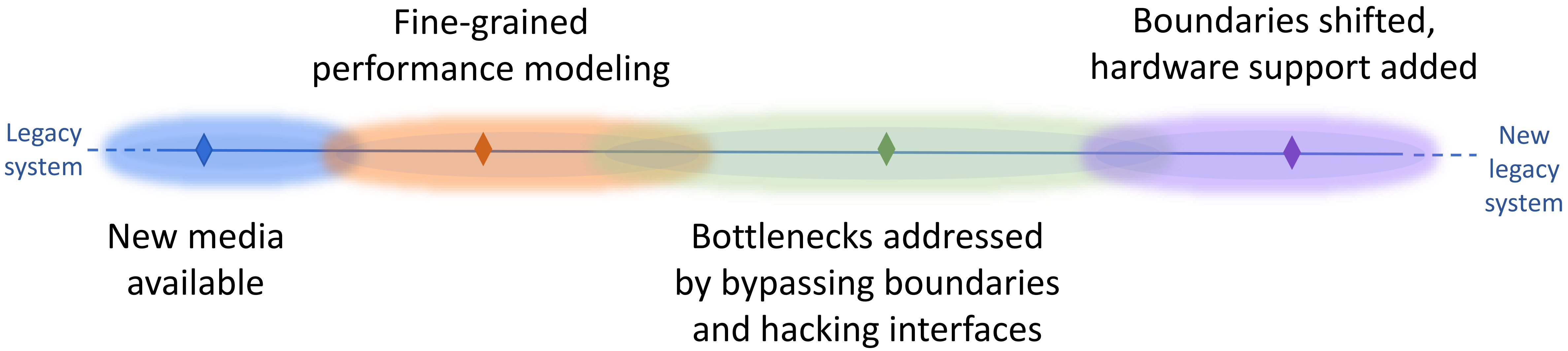
Conclusions



Conclusions

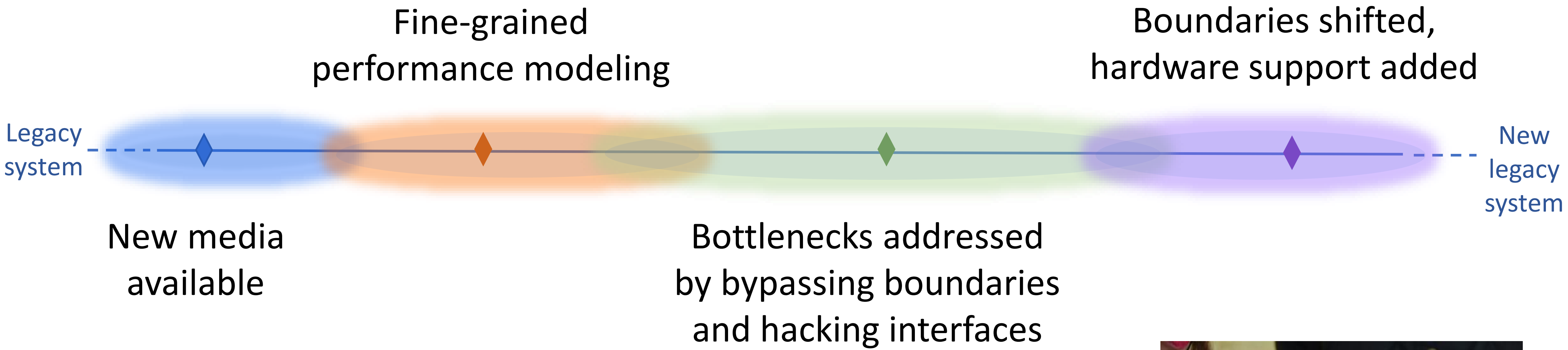


Conclusions



- And what about our written language…?

Conclusions



- And what about our written language…?

🙏4👂!

?