A Case for Efficient Hardware/Software Cooperative Management of Storage and Memory

Justin Meza[*], Yixin Luo[*], Samira Khan[*†], Jishen Zhao[§], Yuan Xie[§‡], and Onur Mutlu[*]

[*]Carnegie Mellon University
[§]Pennsylvania State University
[†]Intel Labs [‡]AMD Research

SAFARI

Carnegie Mellon

Overview

- Traditional systems have a two-level storage model
  - Access **volatile** data in memory with a **load/store** interface
  - Access **persistent** data in storage with a **file system** interface
  - Problem: Operating system (OS) and file system (FS) code and buffering for storage lead to energy and performance inefficiencies

- Opportunity: New non-volatile memory (NVM) technologies can help provide fast (similar to DRAM), persistent storage (similar to Flash)
  - Unfortunately, OS and FS code can easily become energy efficiency and performance bottlenecks if we keep the traditional storage model

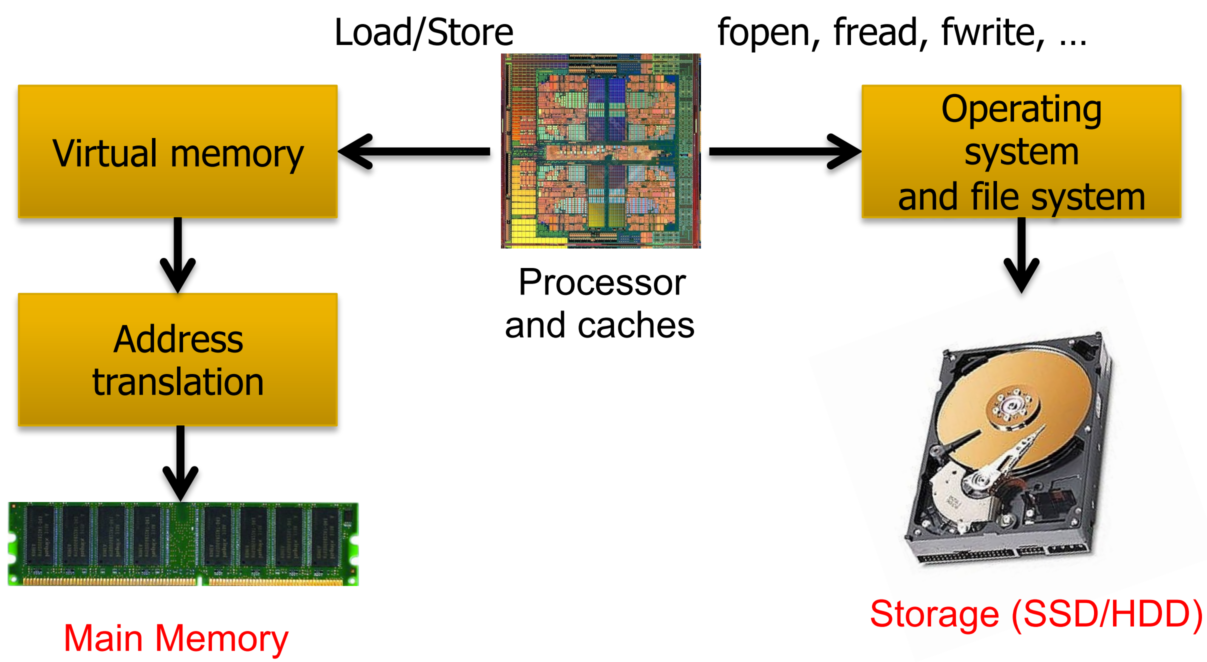
- This work: makes a case for hardware/software cooperative management of storage and memory within a single-level
  - We describe the idea of a Persistent Memory Manager (PMM) for efficiently coordinating storage and memory, and quantify its benefit
  - And, examine questions and challenges to address to realize PMM

Talk Outline

- Background: Storage and Memory Models
- Motivation: Eliminating Operating/File System Bottlenecks
- Our Proposal: Hardware/Software Coordinated Management of Storage and Memory
  - Opportunities and Benefits
- Evaluation Methodology
- Evaluation Results
- Related Work
- New Questions and Challenges
- Conclusions

A Tale of Two Storage Levels

- Traditional systems use a two-level storage model
  - Volatile data is stored in DRAM
  - Persistent data is stored in HDD and Flash
- Accessed through two vastly different interfaces



A Tale of Two Storage Levels

- Two-level storage arose in systems due to the widely different access latencies and methods of the commodity storage devices
  - Fast, low capacity, volatile DRAM → working storage
  - Slow, high capacity, non-volatile hard disk drives → persistent storage

- Data from slow storage media is buffered in fast DRAM
  - After that it can be manipulated by programs → programs cannot directly access persistent storage
  - It is the programmer's job to translate this data between the two formats of the two-level storage (files and data structures)

- Locating, transferring, and translating data and formats between the two levels of storage can waste significant energy and performance

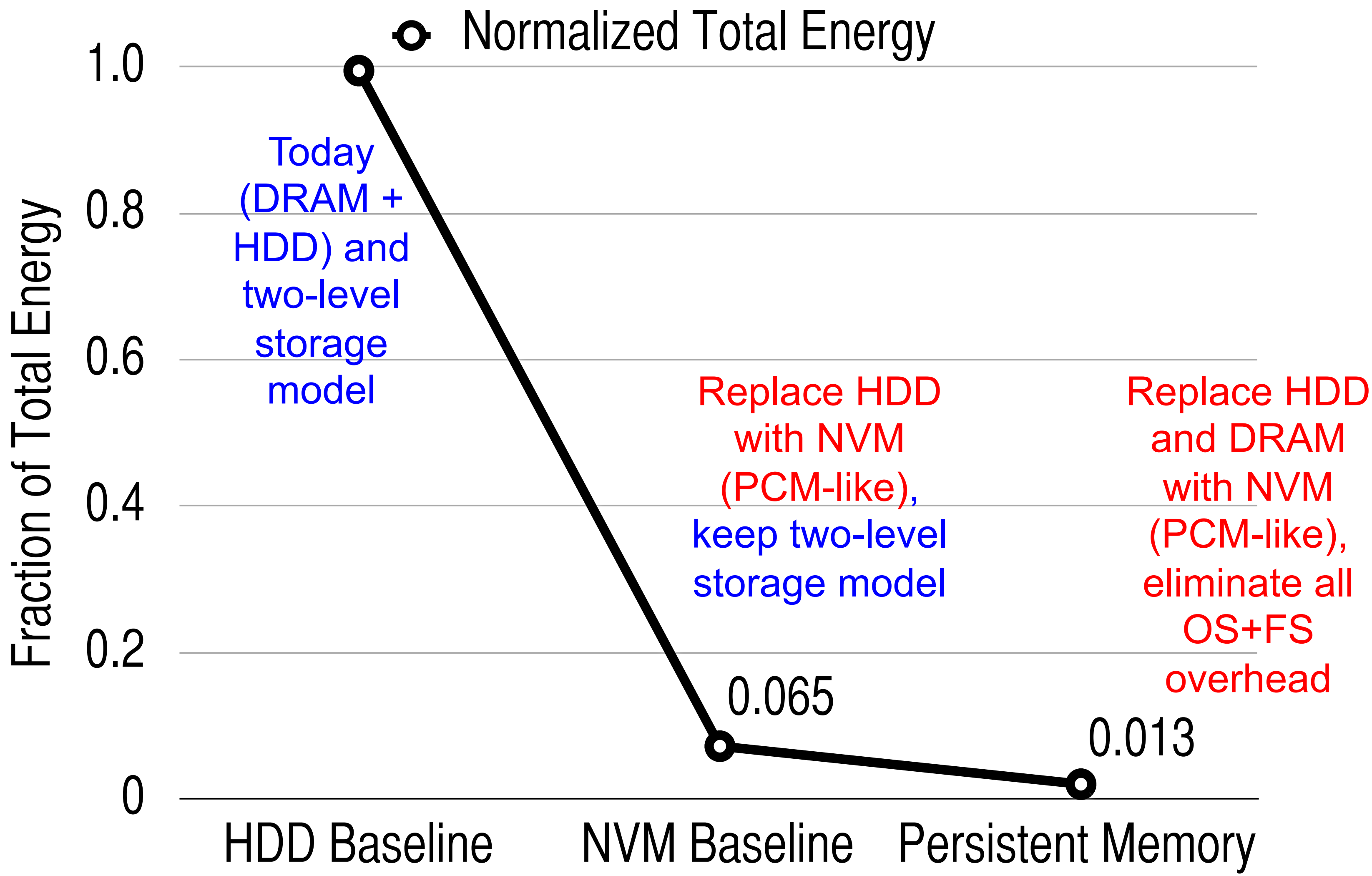
Opportunity: New Non-Volatile Memories

- Emerging memory technologies provide the potential for unifying storage and memory (e.g., Phase-Change, STT-RAM, RRAM)
  - Byte-addressable (can be accessed like DRAM)
  - Low latency (comparable to DRAM)
  - Low power (idle power better than DRAM)
  - High capacity (closer to Flash)
  - Non-volatile (can enable persistent storage)
  - May have limited endurance (but, better than Flash)

- Can provide fast access to ***both*** volatile data and persistent storage

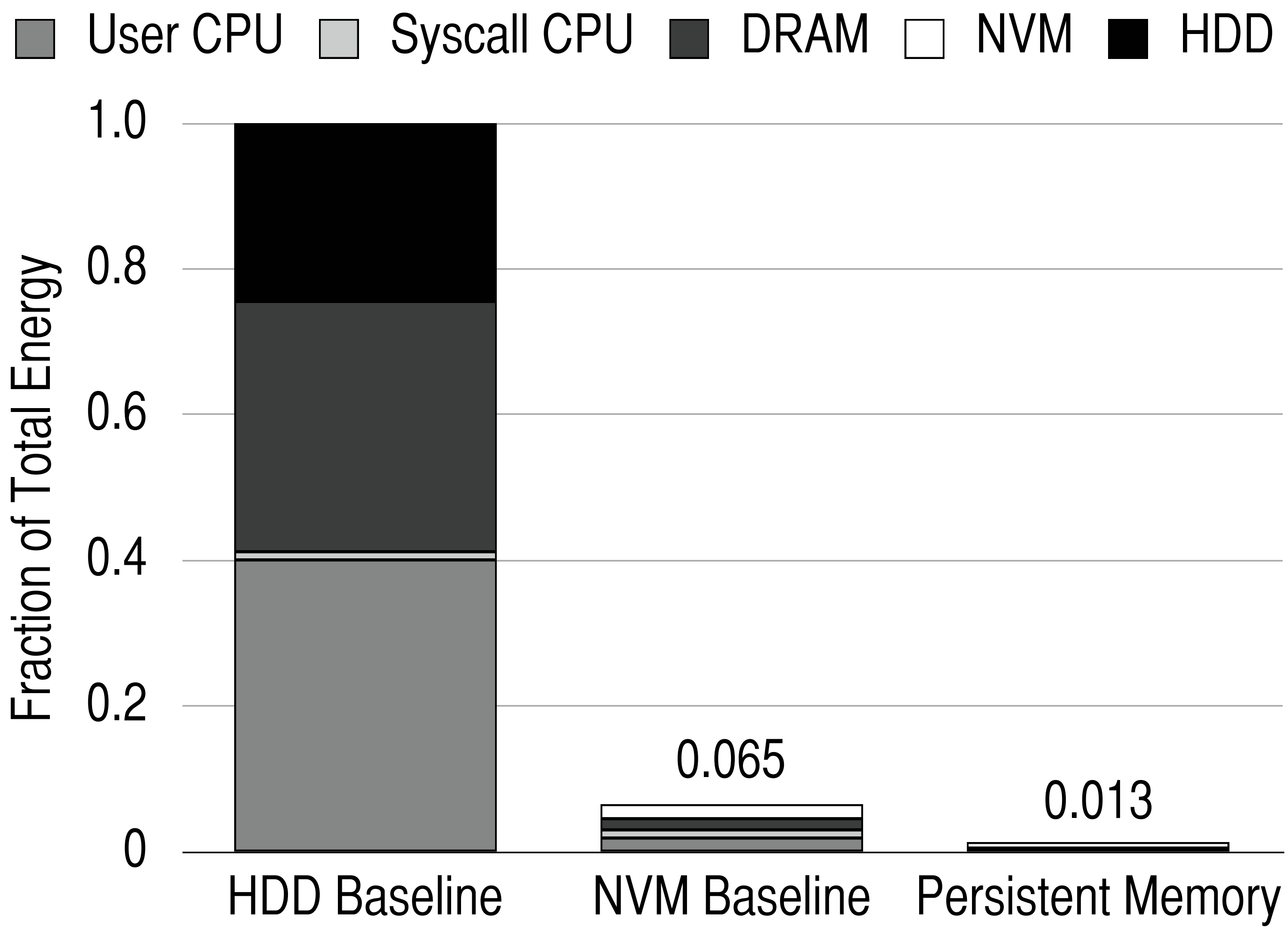
- Question: if such devices are used, is it efficient to keep a two-level storage model?

Eliminating Traditional Storage Bottlenecks



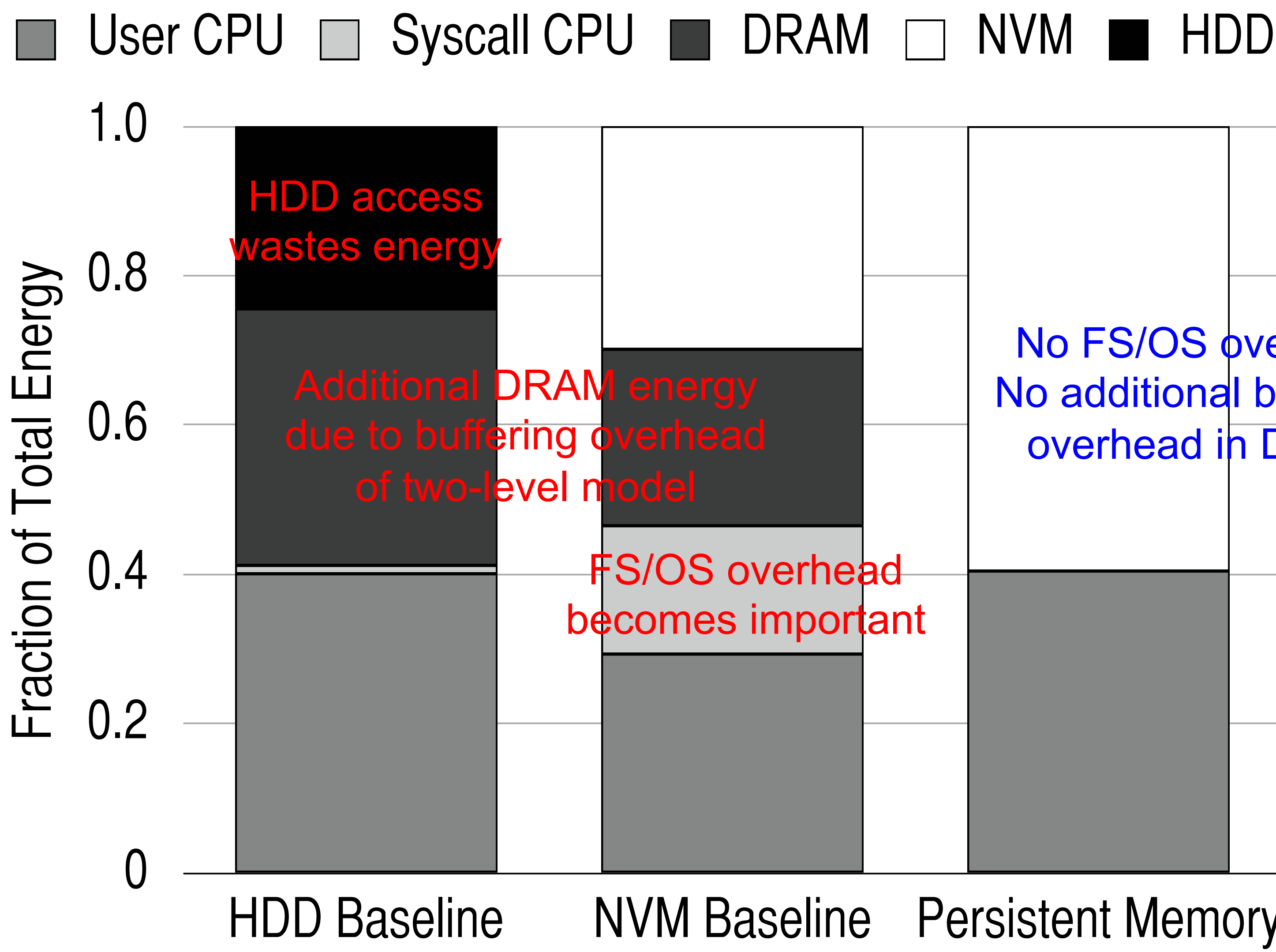
Results for PostMark

Eliminating Traditional Storage Bottlenecks



Results for PostMark

Where is Energy Spent in Each Model?



Results for PostMark

Outline

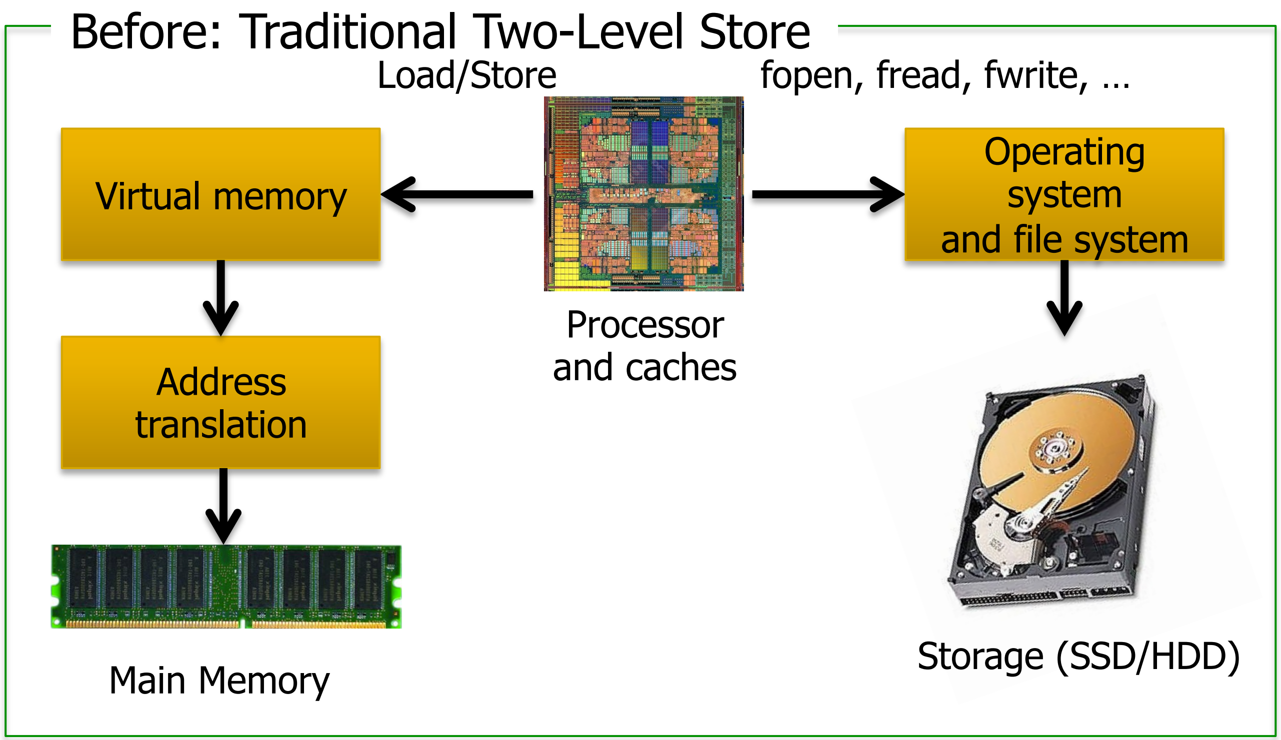
- Background: Storage and Memory Models
- Motivation: Eliminating Operating/File System Bottlenecks
- Our Proposal: Hardware/Software Coordinated Management of Storage and Memory
  - Opportunities and Benefits
- Evaluation Methodology
- Evaluation Results
- Related Work
- New Questions and Challenges
- Conclusions

Our Proposal: Coordinated HW/SW Memory and Storage Management

- Goal: Unify memory and storage to eliminate wasted work to locate, transfer, and translate data
  - Improve both energy and performance
  - Simplify programming model as well

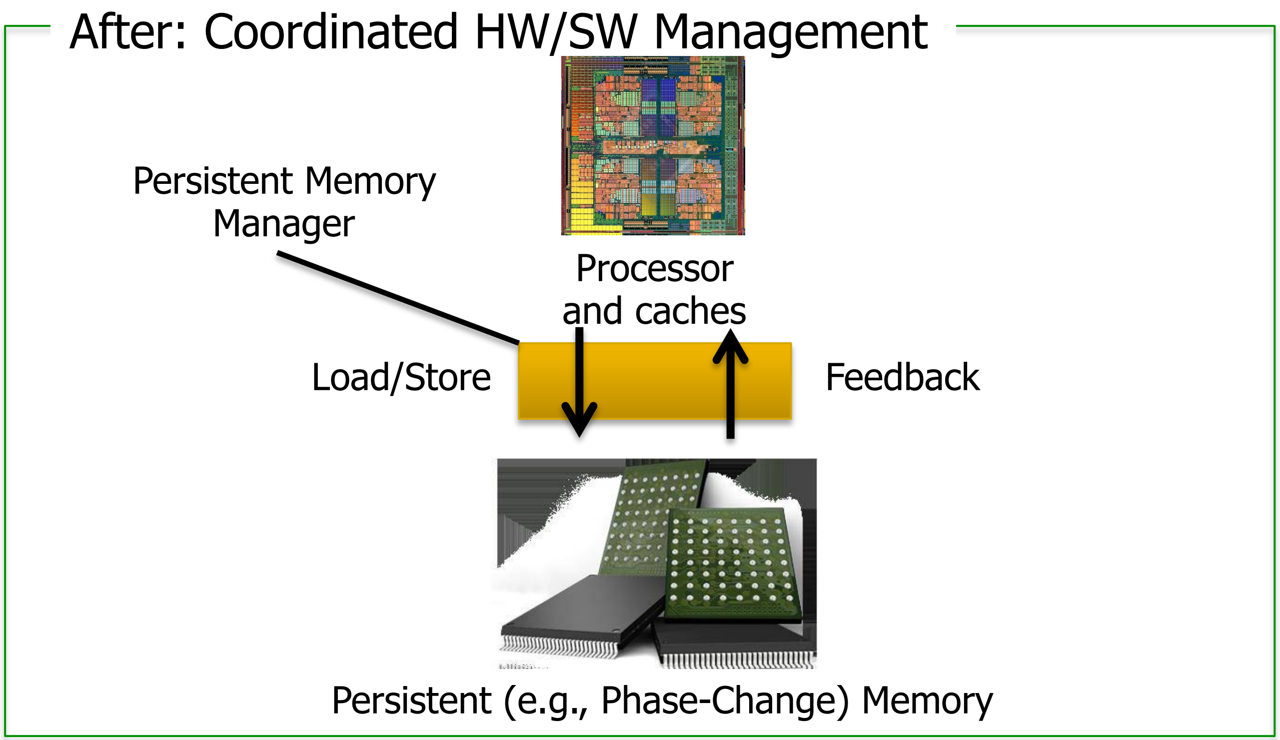
Our Proposal: Coordinated HW/SW Memory and Storage Management

- Goal: Unify memory and storage to eliminate wasted work to locate, transfer, and translate data
  - Improve both energy and performance
  - Simplify programming model as well



Our Proposal: Coordinated HW/SW Memory and Storage Management

- Goal: Unify memory and storage to eliminate wasted work to locate, transfer, and translate data
  - Improve both energy and performance
  - Simplify programming model as well



The Persistent Memory Manager (PMM)

- Exposes a load/store interface to access persistent data
  - Applications can directly access persistent memory → no conversion, translation, location overhead for persistent data
- Manages data placement, location, persistence, security
  - To get the best of multiple forms of storage
- Manages metadata storage and retrieval
  - This can lead to overheads that need to be managed
- Exposes hooks and interfaces for system software
  - To enable better data placement and management decisions

The Persistent Memory Manager

- Persistent Memory Manager
  - Exposes a load/store interface to access persistent data
  - Manages data placement, location, persistence, security
  - Manages metadata storage and retrieval
  - Exposes hooks and interfaces for system software

- Example program manipulating a persistent object:

```
int main(void) {
  // data in file.dat is persistent
  FILE myData = "file.dat";
  myData = new int[64];
}
void updateValue(int n, int value) {
  FILE myData = "file.dat";
  myData[n] = value; // value is persistent
}
```

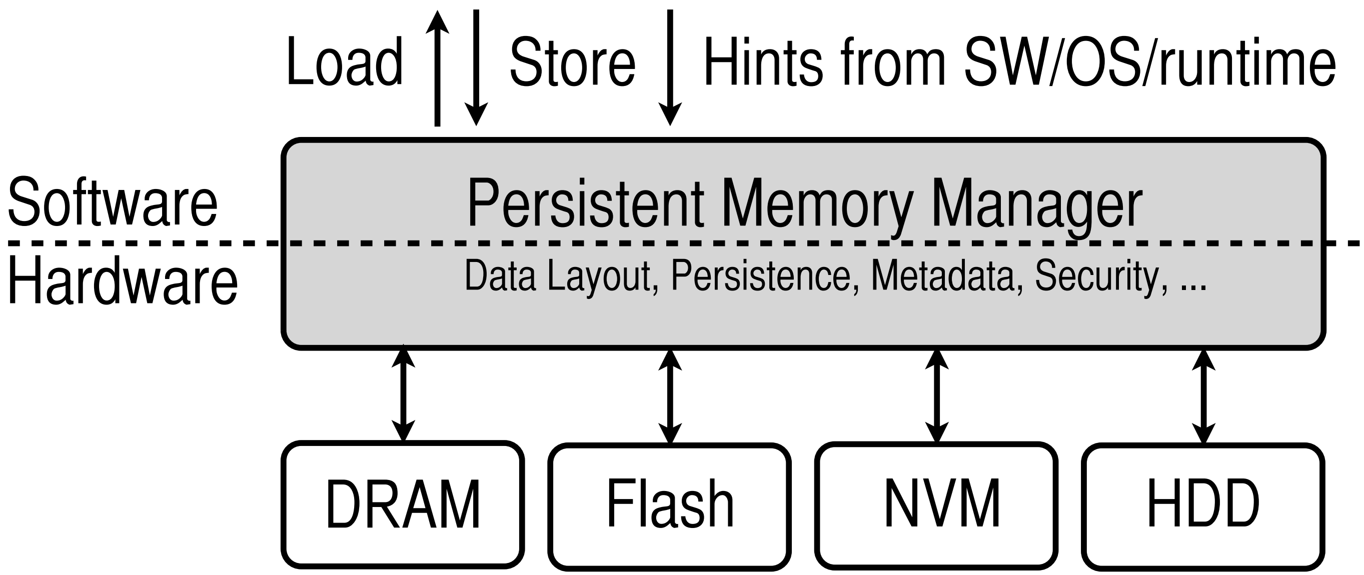
Create persistent object and its handle

Allocate a persistent array and assign

Load/store interface

Putting Everything Together

```
int main(void) {
  // data in file.dat is persistent
  FILE myData = "file.dat";
  myData = new int[64];
}
void updateValue(int n, int value) {
  FILE myData = "file.dat";
  myData[n] = value; // value is persistent
}
```



PMM uses access and hint information to allocate, locate, migrate and access data in the heterogeneous array of devices

Outline

- Background: Storage and Memory Models
- Motivation: Eliminating Operating/File System Bottlenecks
- Our Proposal: Hardware/Software Coordinated Management of Storage and Memory
  - Opportunities and Benefits
- Evaluation Methodology
- Evaluation Results
- Related Work
- New Questions and Challenges
- Conclusions

Opportunities and Benefits

- We've identified at least five opportunities and benefits of a unified storage/memory system that gets rid of the two-level model:
  1. Eliminating system calls for file operations
  2. Eliminating file system operations
  3. Efficient data mapping/location among heterogeneous devices
  4. Providing security and reliability in persistent memories
  5. Hardware/software cooperative data management

Eliminating System Calls for File Operations

- A persistent memory can expose a large, linear, persistent address space
  - Persistent storage objects can be directly manipulated with load/store operations
- This eliminates the need for layers of operating system code
  - Typically used for calls like `open`, `read`, and `write`
- Also eliminates OS file metadata
  - File descriptors, file buffers, and so on

Eliminating File System Operations

- Locating files is traditionally done using a *file system*
  - Runs code and traverses structures in software to locate files

- Existing hardware structures for locating data in virtual memory can be extended and adapted to meet the needs of persistent memories
  - Memory Management Units (MMUs), which map virtual addresses to physical addresses
  - Translation Lookaside Buffers (TLBs), which cache mappings of virtual-to-physical address translations

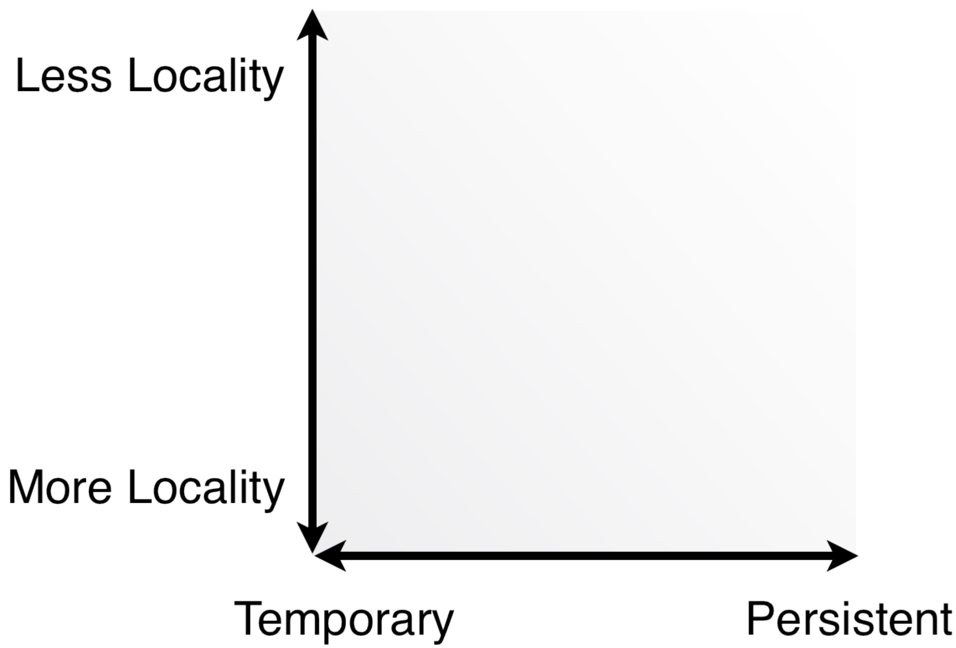
- Potential to eliminate file system code
- At the cost of additional hardware overhead to handle persistent data storage

Efficient Data Mapping among Heterogeneous Devices

- A persistent memory exposes a large, persistent address space
  - But it may use many different devices to satisfy this goal
  - From fast, low-capacity volatile DRAM to slow, high-capacity non-volatile HDD or Flash
  - And other NVM devices in between

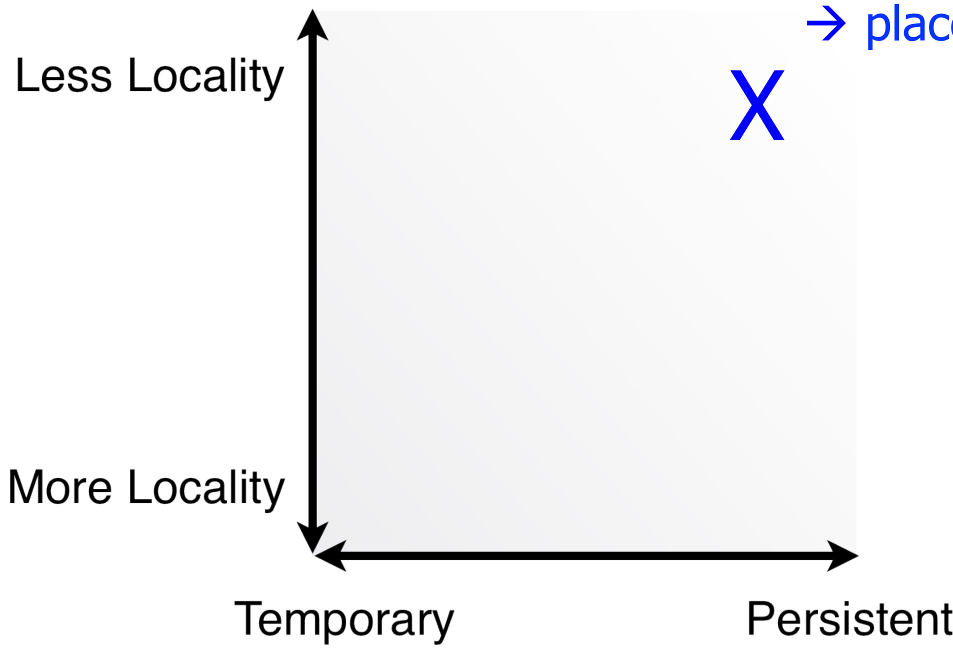
- Performance and energy can benefit from good placement of data among these devices
  - Utilizing the strengths of each device and avoiding their weaknesses, if possible
  - For example, consider two important application characteristics: locality and persistence

Efficient Data Mapping among Heterogeneous Devices

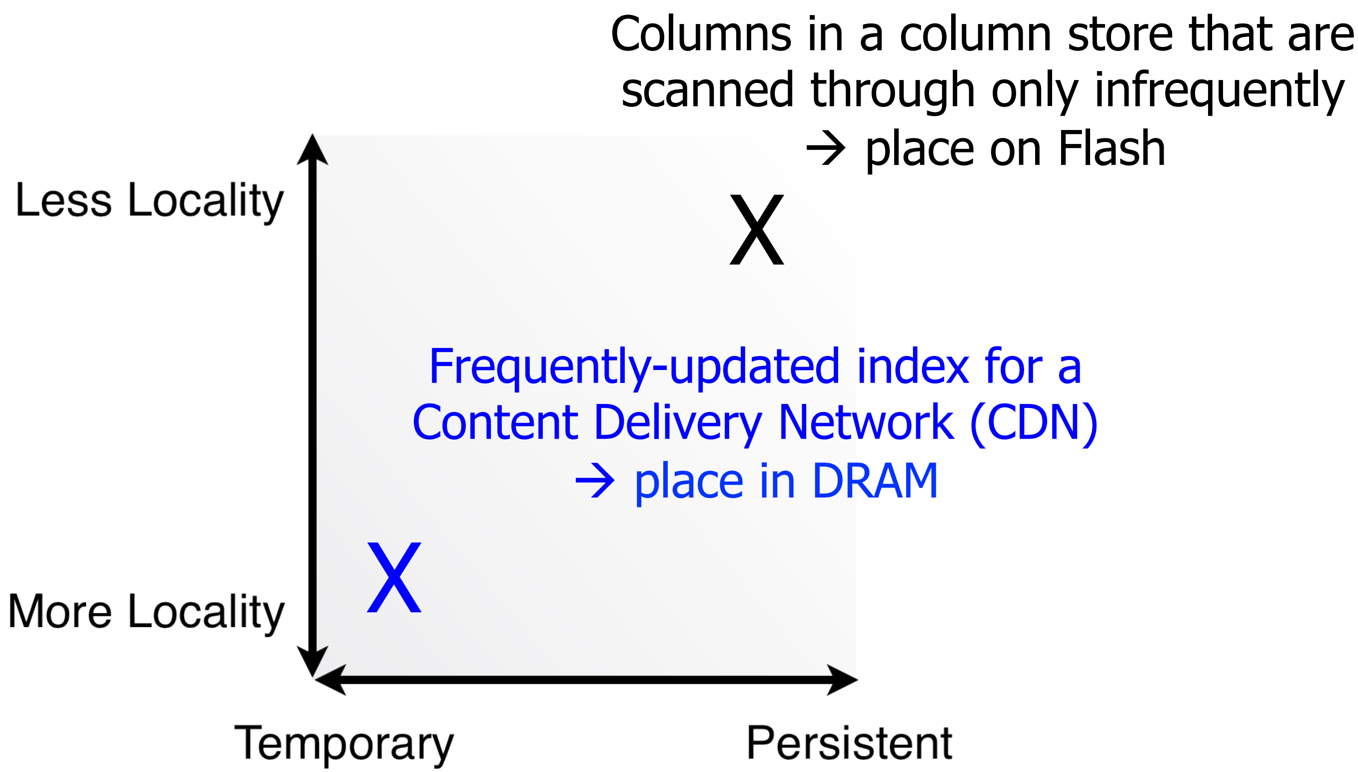


Efficient Data Mapping among Heterogeneous Devices

Columns in a column store that are scanned through only infrequently
→ place on Flash



Efficient Data Mapping among Heterogeneous Devices



Applications or system software can provide hints for data placement

Providing Security and Reliability

- A persistent memory deals with data at the granularity of bytes and not necessarily files
  - Provides the opportunity for much finer-grained security and protection than traditional two-level storage models provide/afford
  - Need efficient techniques to avoid large metadata overheads

- A persistent memory can improve application reliability by ensuring updates to persistent data are less vulnerable to failures
  - Need to ensure that changes to copies of persistent data placed in volatile memories become persistent

HW/SW Cooperative Data Management

- Persistent memories can expose hooks and interfaces to applications, the OS, and runtimes
  - Have the potential to provide improved system robustness and efficiency than by managing persistent data with either software or hardware alone

- Can enable fast checkpointing and reboots, improve application reliability by ensuring persistence of data
  - How to redesign availability mechanisms to take advantage of these?

- Persistent locks and other persistent synchronization constructs can enable more robust programs and systems

Quantifying Persistent Memory Benefits

- We have identified several opportunities and benefits of using persistent memories without the traditional two-level store model

- We will next quantify:
  - How do persistent memories affect system performance?
  - How much energy reduction is possible?
  - Can persistent memories achieve these benefits despite additional access latencies to the persistent memory manager?

Outline

- Background: Storage and Memory Models
- Motivation: Eliminating Operating/File System Bottlenecks
- Our Proposal: Hardware/Software Coordinated Management of Storage and Memory
    - Opportunities and Benefits
- Evaluation Methodology
- Evaluation Results
- Related Work
- New Questions and Challenges
- Conclusions

Evaluation Methodology

- Hybrid real system / simulation-based approach
  - System calls are executed on host machine (functional correctness) and timed to accurately model their latency in the simulator
  - Rest of execution is simulated in Multi2Sim (enables hardware-level exploration)
- Power evaluated using McPAT and memory power models
- 16 cores, 4-wide issue, 128-entry instruction window, 1.6 GHz
- Volatile memory: 4GB DRAM, 4KB page size, 100-cycle latency
- Persistent memory
  - HDD (measured): 4ms seek latency, 6Gbps bus rate
  - NVM: (modeled after PCM) 4KB page size, 160-/480-cycle (read/write) latency

Evaluated Systems

- HDD Baseline (HB)
  - Traditional system with volatile DRAM memory and persistent HDD storage
  - Overheads of operating system and file system code and buffering
- HDD without OS/FS (HW)
  - Same as HDD Baseline, but with the ideal elimination of all OS/FS overheads
  - System calls take 0 cycles (but HDD access takes normal latency)
- NVM Baseline (NB)
  - Same as HDD Baseline, but HDD is replaced with NVM
  - Still has OS/FS overheads of the two-level storage model
- Persistent Memory (PM)
  - Uses only NVM (no DRAM) to ensure full-system persistence
  - All data accessed using loads and stores
  - Does not waste energy on system calls
  - Data is manipulated directly on the NVM device

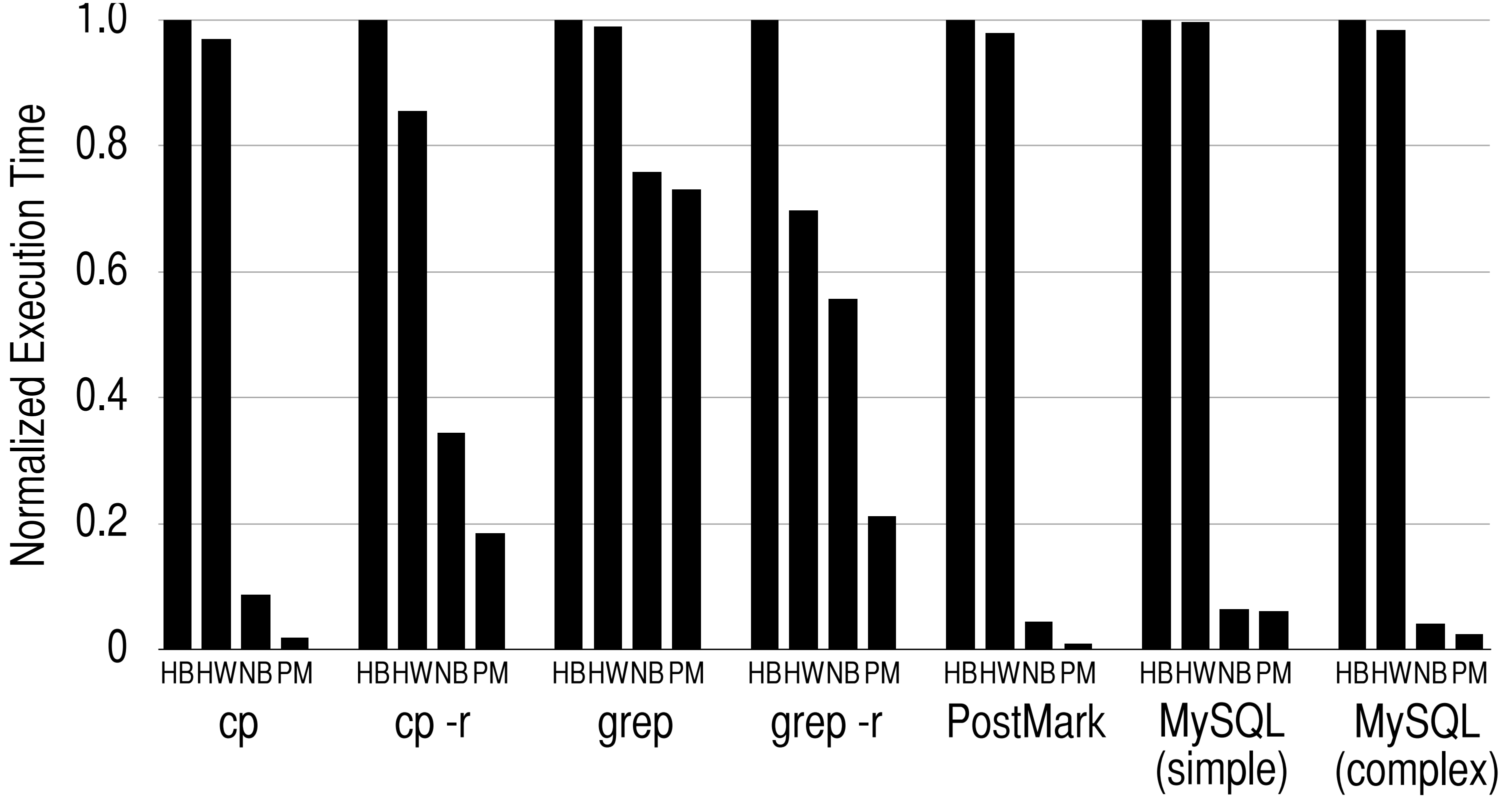
Evaluated Workloads

- Unix utilities that manipulate files
  - cp: copy a large file from one location to another
  - cp –r: copy files in a directory tree from one location to another
  - grep: search for a string in a large file
  - grep –r: search for a string recursively in a directory tree

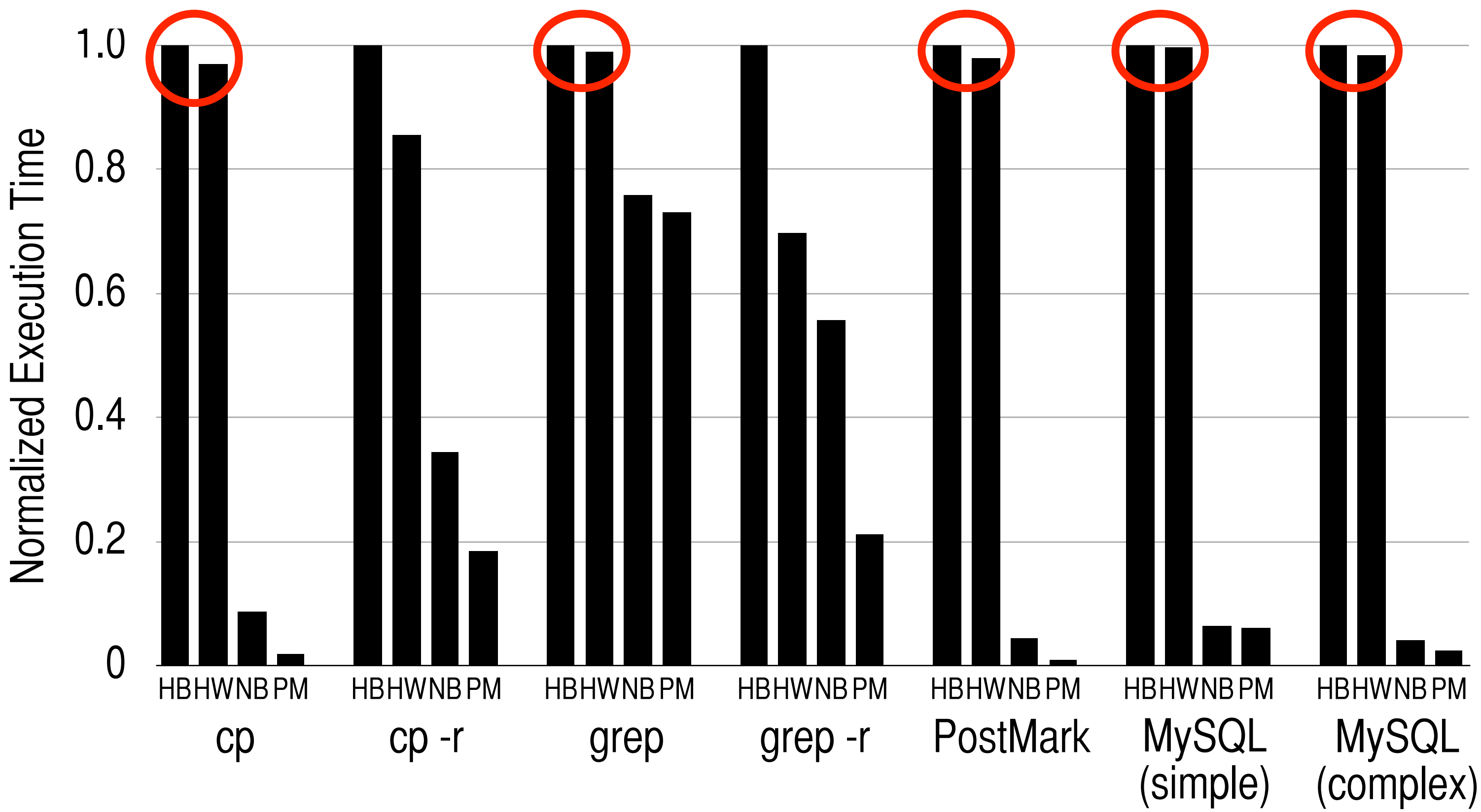
- PostMark: an I/O-intensive benchmark from NetApp
  - Emulates typical access patterns for email, news, web commerce

- MySQL Server: a popular database management system
  - OLTP-style queries generated by Sysbench
  - MySQL (simple): single, random read to an entry
  - MySQL (complex): reads/writes 1 to 100 entries per transaction

Performance Results

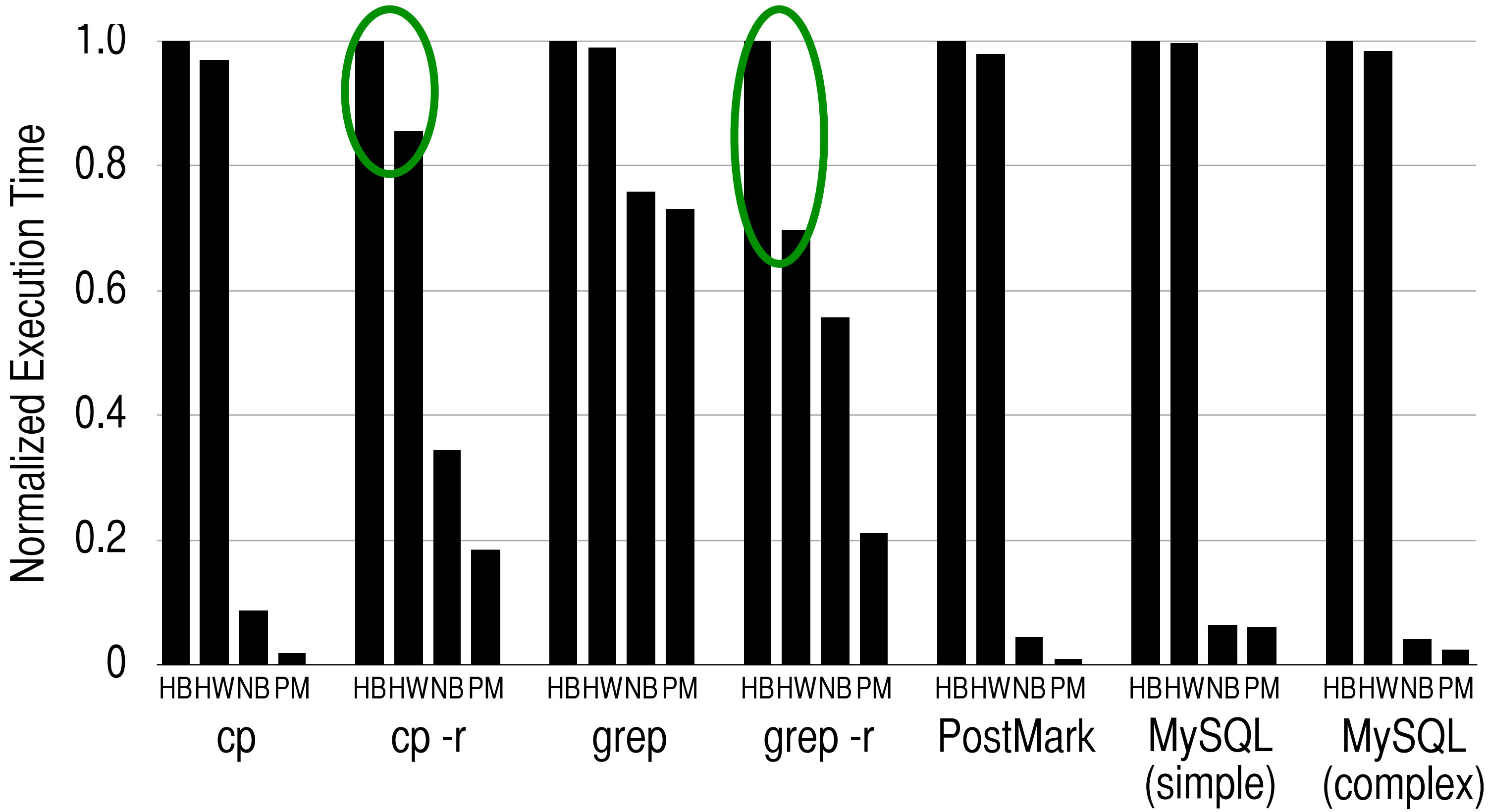


Performance Results: HDD w/o OS/FS



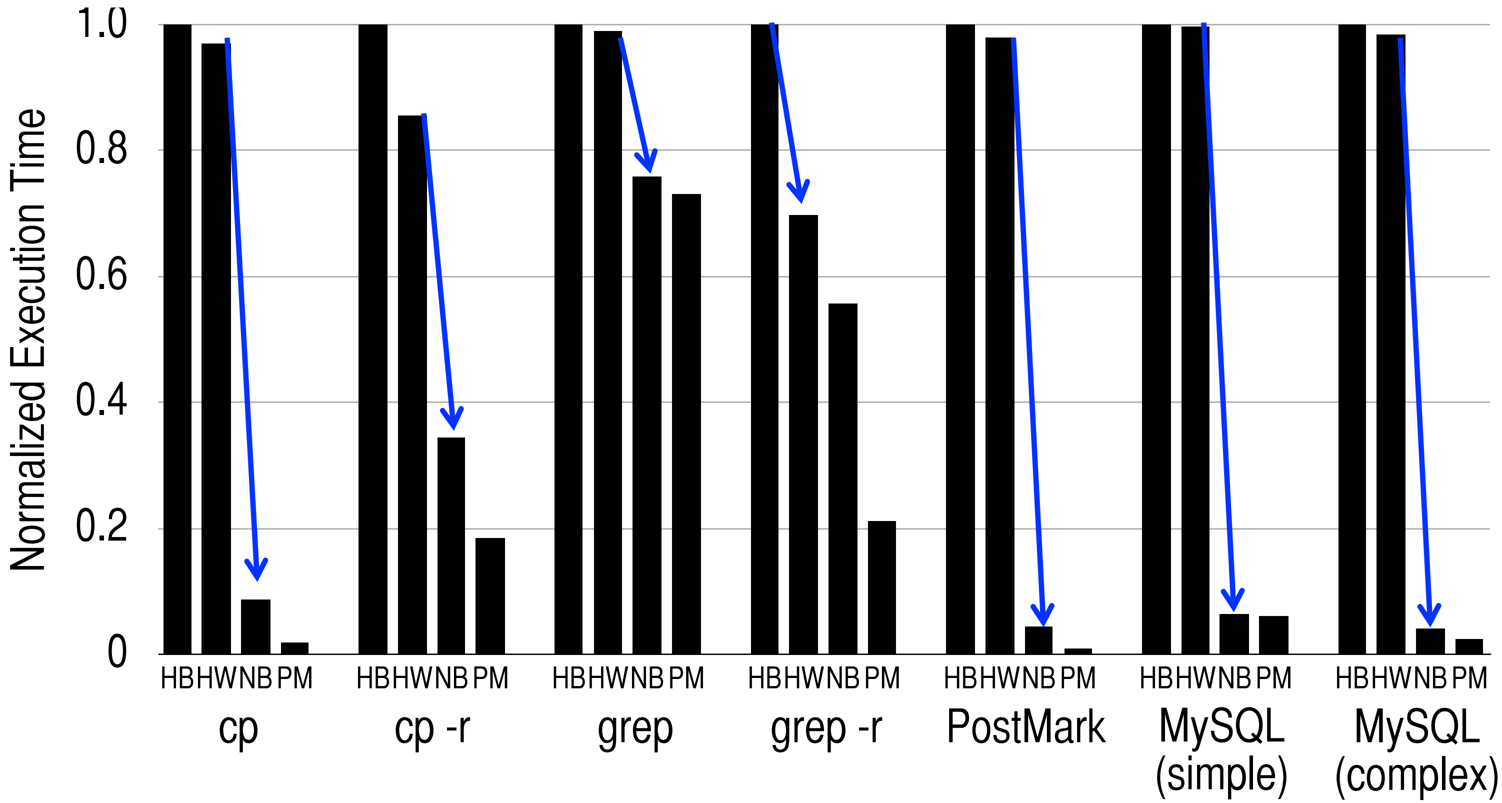
For HDD-based systems, eliminating OS/FS overheads typically leads to small performance improvements → execution time dominated by HDD access latency

Performance Results: HDD w/o OS/FS



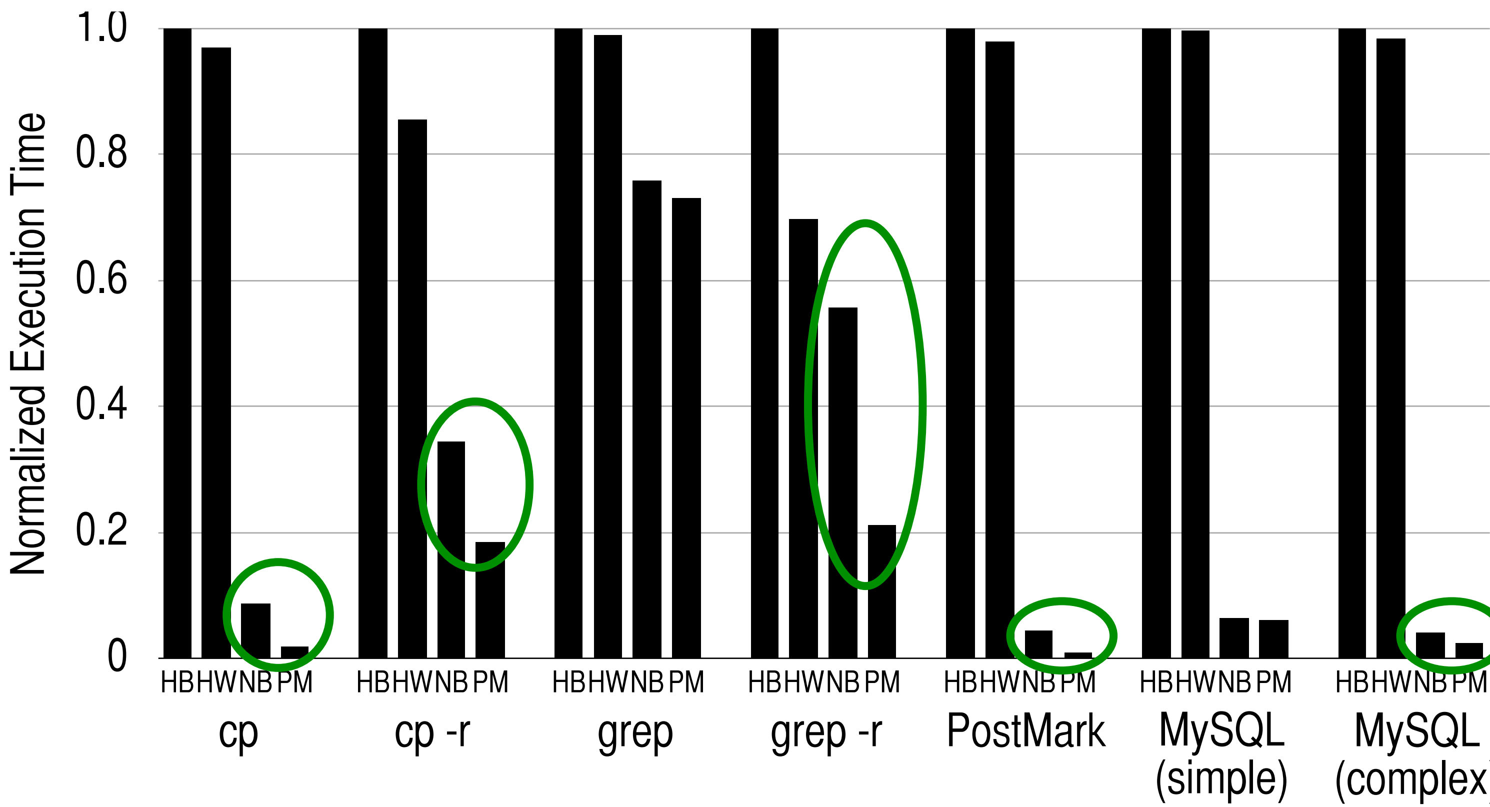
Though, for more complex file system operations like directory traversal (seen with cp -r and grep -r), eliminating the OS/FS overhead improves performance

Performance Results: HDD to NVM



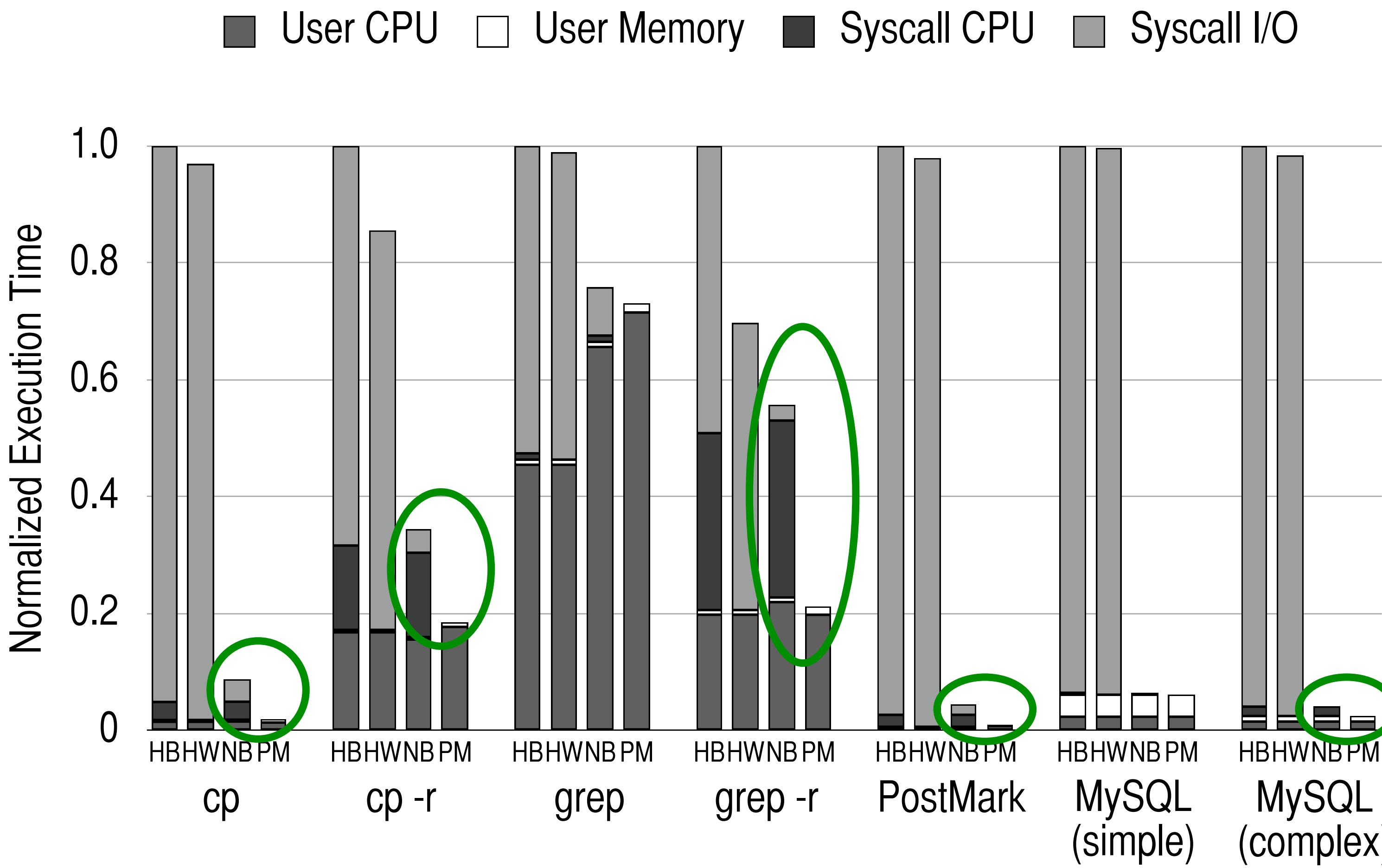
Switching from an HDD to NVM greatly reduces execution time due to NVM's much faster access latencies, especially for I/O-intensive workloads (cp, PostMark, MySQL)

Performance Results: NVM to PMM



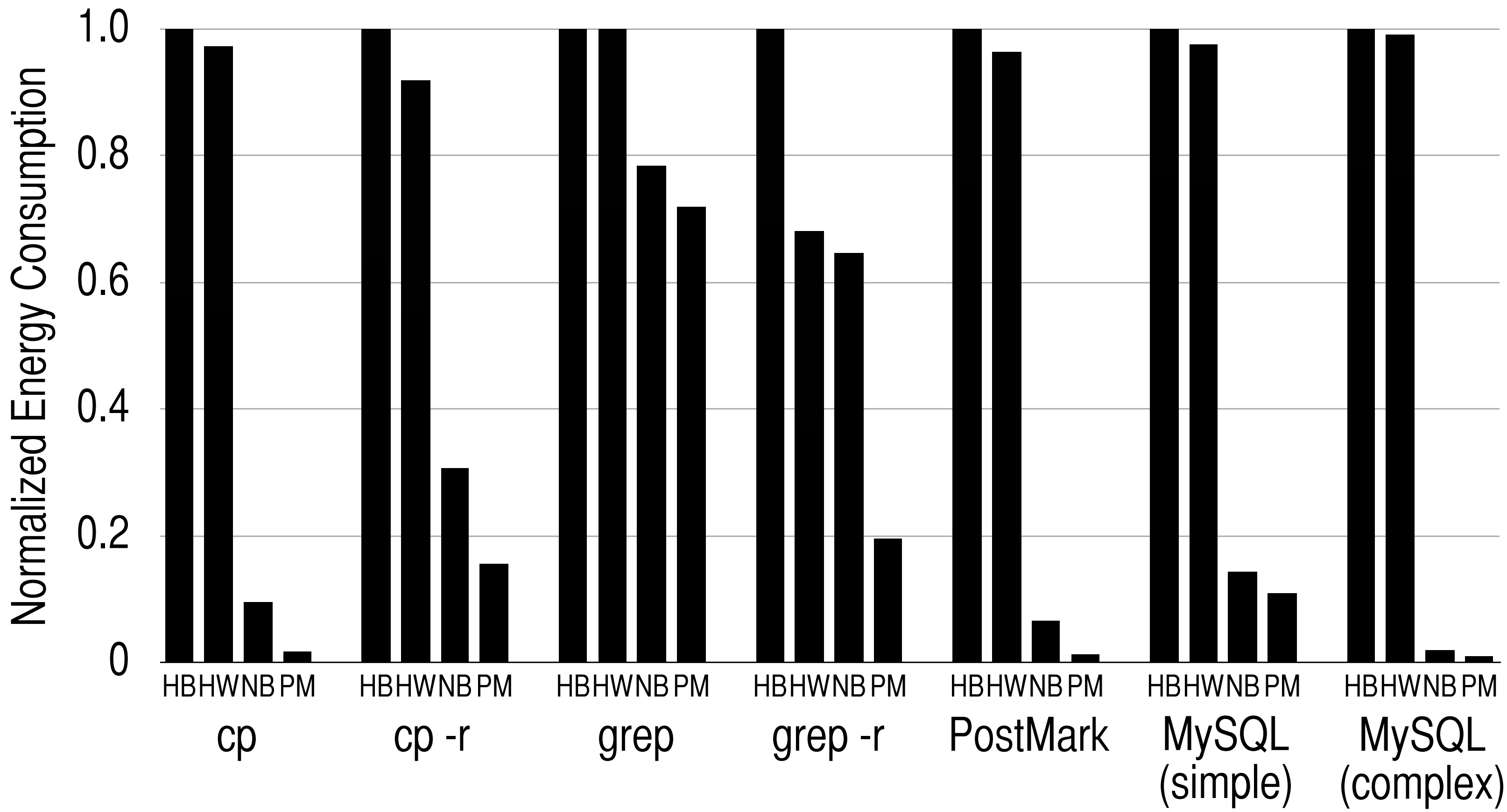
For most workloads, eliminating OS/FS code and buffering improves performance greatly on top of the NVM Baseline system
(even when DRAM is eliminated from the system)

Performance Results

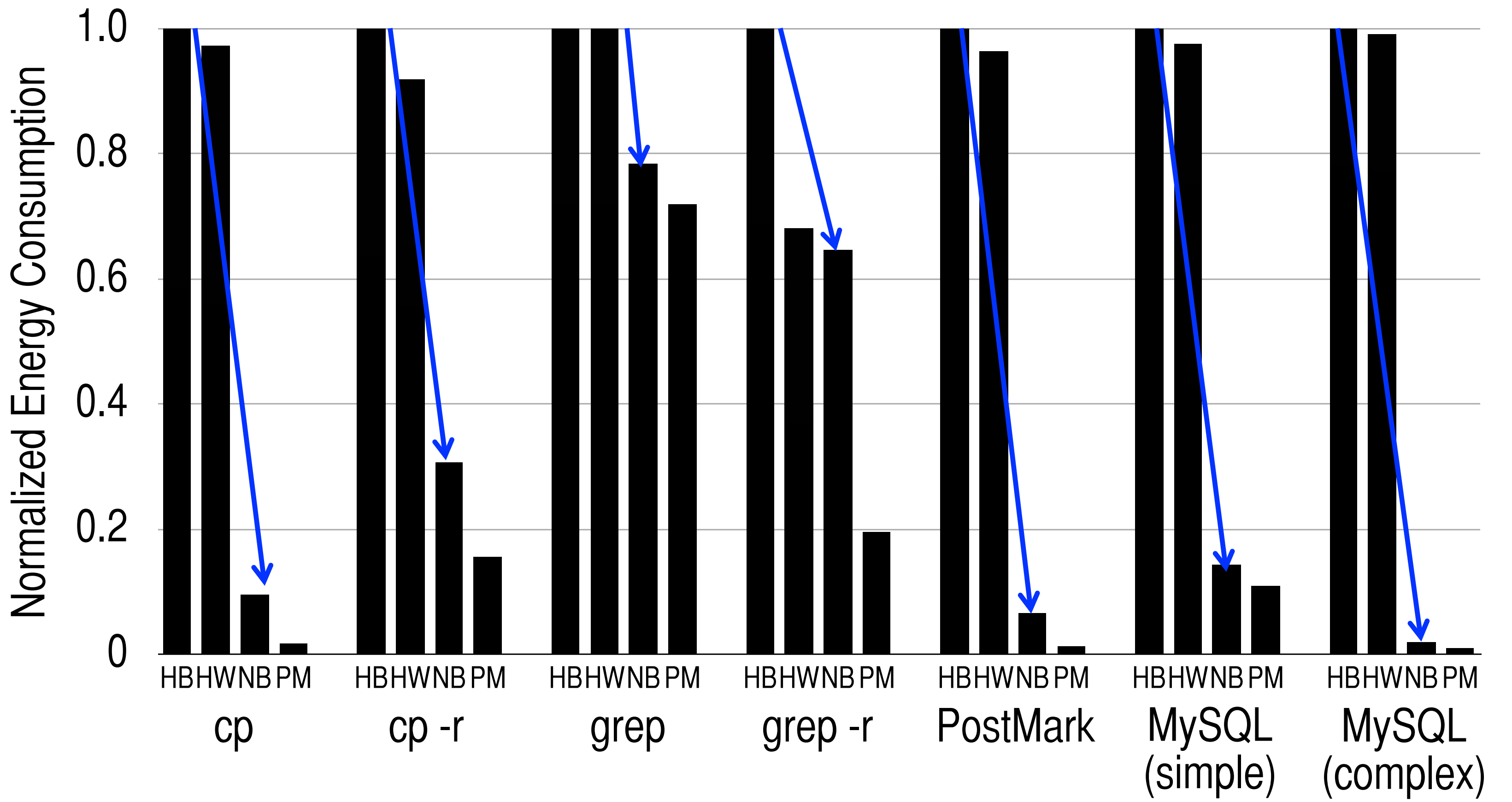


The workloads that see the greatest improvement from using a Persistent Memory are those that spend a large portion of their time executing system call code due to the two-level storage model

Energy Results

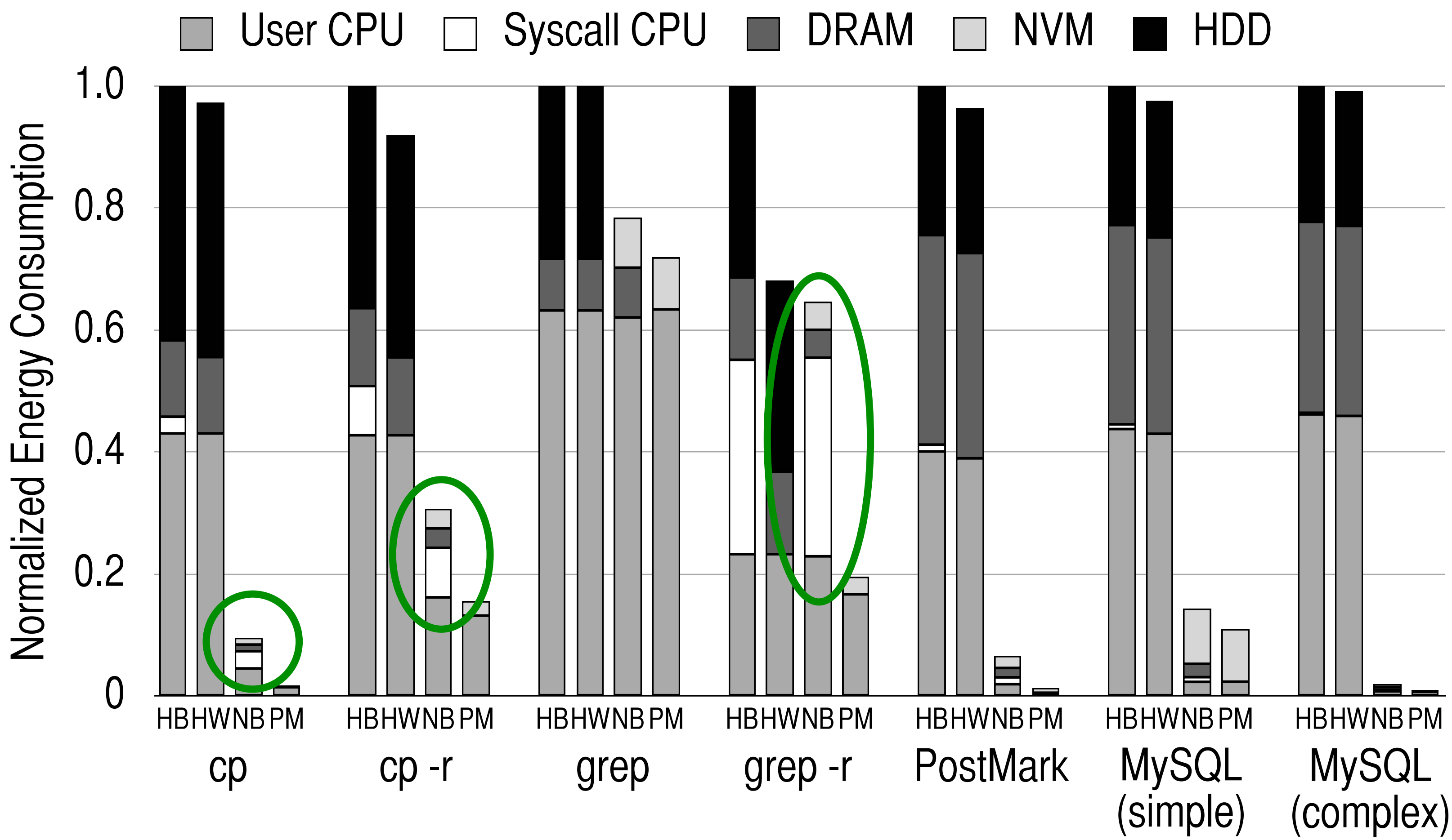


Energy Results: HDD to NVM



Between HDD-based and NVM-based systems, lower NVM energy leads to greatly reduced energy consumption

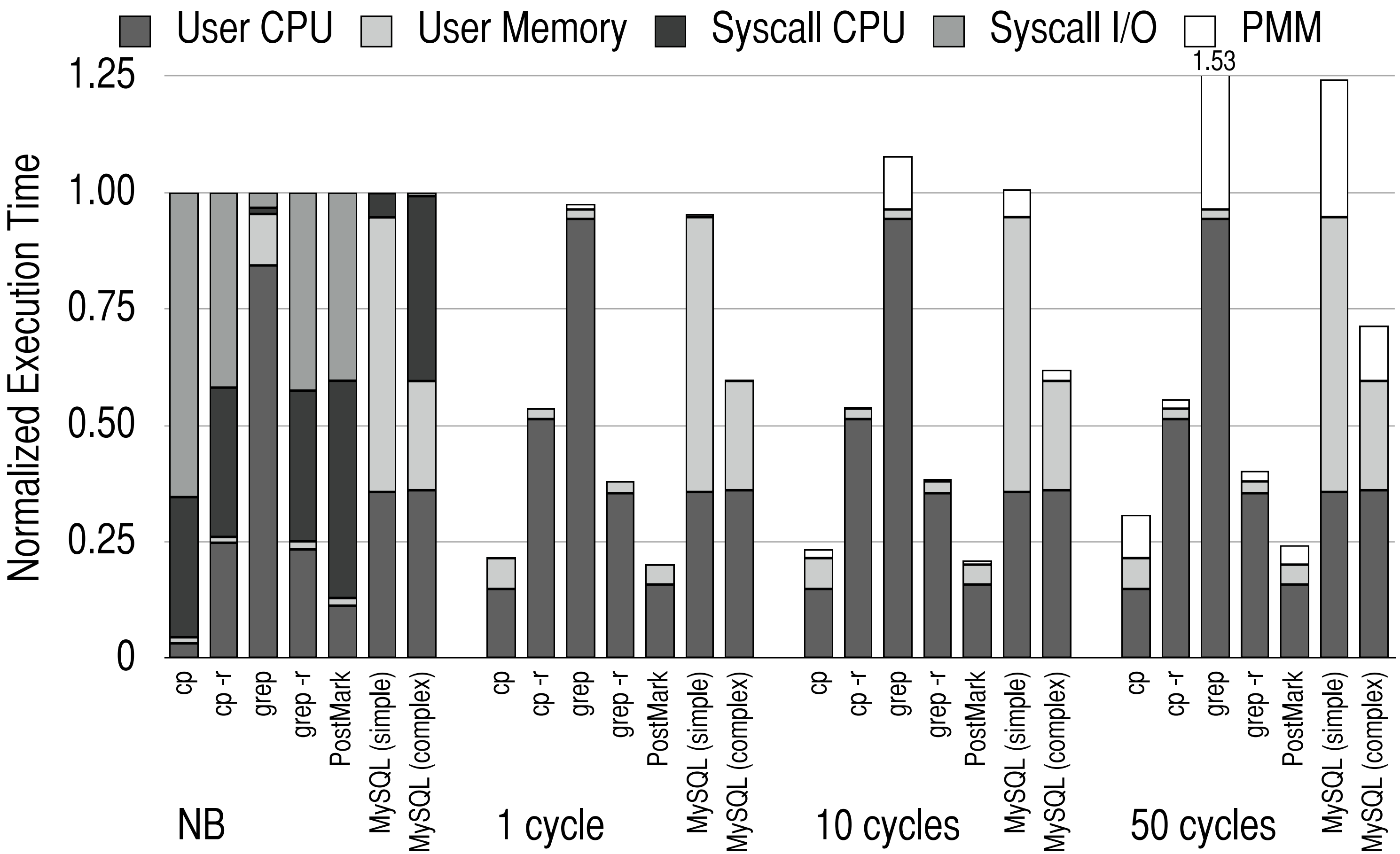
Energy Results: NVM to PMM



Between systems with and without OS/FS code, energy improvements come from:
1. reduced code footprint, 2. reduced data movement

Large energy reductions with a PMM over the NVM based system

Scalability Analysis: Effect of PMM Latency



Even if each PMM access takes a non-overlapped 50 cycles (conservative), PMM still provides an overall improvement compared to the NVM baseline

Future research should target keeping PMM latencies in check

Outline

- Background: Storage and Memory Models
- Motivation: Eliminating Operating/File System Bottlenecks
- Our Proposal: Hardware/Software Coordinated Management of Storage and Memory
  - Opportunities and Benefits
- Evaluation Methodology
- Evaluation Results
- Related Work
- New Questions and Challenges
- Conclusions

Related Work

- We provide a comprehensive overview of past work related to single-level stores and persistent memory techniques

1. Integrating file systems with persistent memory
   - Need optimized hardware to fully take advantage of new technologies
2. Programming language support for persistent objects
   - Incurs the added latency of indirect data access through software
3. Load/store interfaces to persistent storage
   - Lack efficient and fast hardware support for address translation, efficient file indexing, fast reliability and protection guarantees
4. Analysis of OS overheads with Flash devices
   - Our study corroborates findings in this area and shows even larger consequences for systems with emerging NVM devices

- The goal of our work is to provide cheap and fast hardware support for memories to enable high energy efficiency and performance

Outline

- Background: Storage and Memory Models
- Motivation: Eliminating Operating/File System Bottlenecks
- Our Proposal: Hardware/Software Coordinated Management of Storage and Memory
  - Opportunities and Benefits
- Evaluation Methodology
- Evaluation Results
- Related Work
- New Questions and Challenges
- Conclusions

New Questions and Challenges

- We identify and discuss several open research questions
  - Q1. How to tailor applications for systems with persistent memory?
  - Q2. How can hardware and software cooperate to support a scalable, persistent single-level address space?
  - Q3. How to provide efficient backward compatibility (for two-level stores) on persistent memory systems?
  - Q4. How to mitigate potential hardware performance and energy overheads?

Outline

- Background: Storage and Memory Models
- Motivation: Eliminating Operating/File System Bottlenecks
- Our Proposal: Hardware/Software Coordinated Management of Storage and Memory
  - Opportunities and Benefits
- Evaluation Methodology
- Evaluation Results
- Related Work
- New Questions and Challenges
- Conclusions

Summary and Conclusions

- Traditional two-level storage model is inefficient in terms of performance and energy
  - Due to OS/FS code and buffering needed to manage two models
  - Especially so in future devices with NVM technologies, as we show
- New non-volatile memory based persistent memory designs that use a single-level storage model to unify memory and storage can alleviate this problem
- We quantified the performance and energy benefits of such a single-level persistent memory/storage design
  - Showed significant benefits from reduced code footprint, data movement, and system software overhead on a variety of workloads
- Such a design requires more research to answer the questions we have posed and enable efficient persistent memory managers

  → can lead to a fundamentally more efficient storage system

Thank you.

A Case for Efficient Hardware/Software Cooperative Management of Storage and Memory

Justin Meza[*], Yixin Luo[*], Samira Khan[*†], Jishen Zhao[§], Yuan Xie[§‡], and Onur Mutlu[*]

[*]Carnegie Mellon University
[§]Pennsylvania State University
[†]Intel Labs [‡]AMD Research

SAFARI

Carnegie Mellon