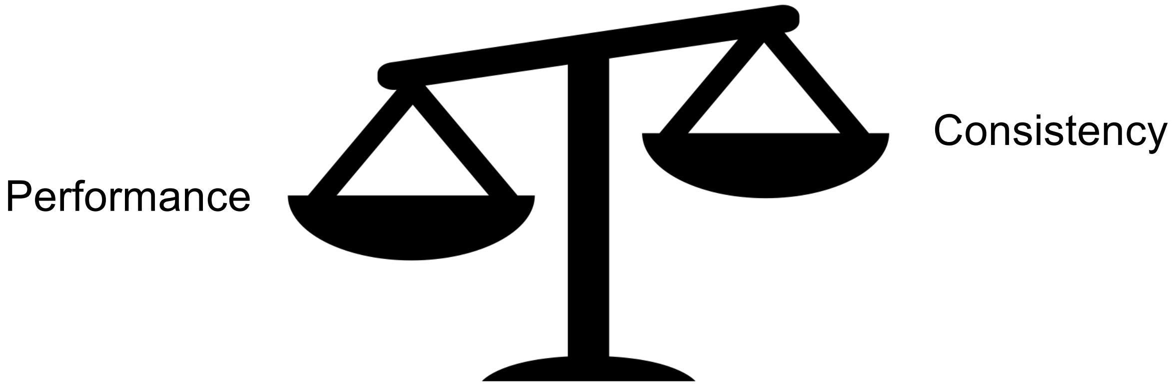
HTMFS: Strong Consistency Comes for Free with Hardware Transactional Memory in Persistent Memory File Systems

Jifei Yi, Mingkai Dong, Fangnuo Wu, Haibo Chen

Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University
Engineering Research Center for Domain-specific Operating Systems, Ministry of Education, China

Performance vs Consistency

- **Early days**
  - Loose consistency guarantees
  - Fsck (file system consistency check) attempt to recover without guarantee after crash



Performance vs Consistency

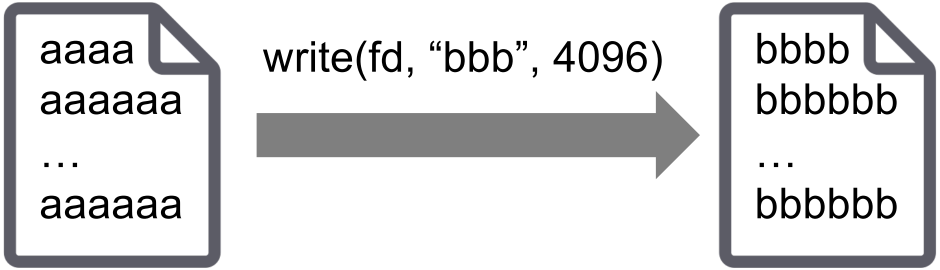
- **Early days**
  - Loose consistency guarantees
  - Fsck (file system consistency check) attempt to recover without guarantee after crash
- **Storage device is getting faster**
  - Crash consistency is important for file systems

Strong Consistency

- **Per-request sequential consistency**
  - Concurrency control
  - Crash consistency

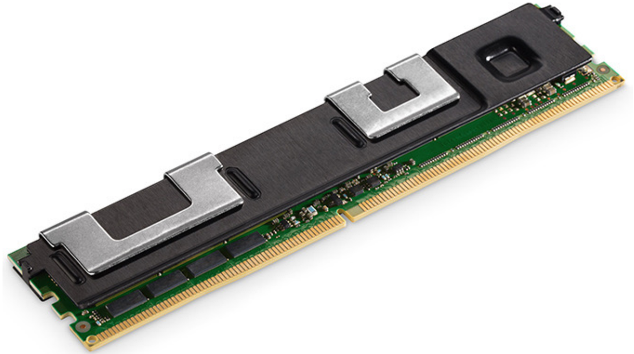
Strong Consistency

- **Per-request sequential consistency**
  - Concurrency control
  - Crash consistency
- **All-or-nothing semantics**



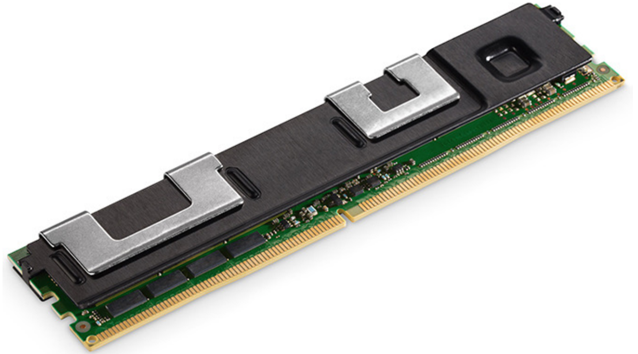
Persistent Memory

- **Pros**
  - Fast
  - Byte-addressable
  - Non-volatile



Persistent Memory

- **Pros**
  - Fast
  - Byte-addressable
  - Non-volatile



Providing strong consistency guarantees is particularly challenging for memory-based file systems because maintaining data consistency in NVMM can be costly. [1]

[1] Jian Xu and Steven Swanson. NOVA: A log-structured file system for hybrid volatile/non-volatile main memories. FAST 16, 2016.

Hardware Transactional Memory

- **Wrap memory accesses with _xbegin() and _xend()**
  - Successful commit: all changes complete atomically (become globally visible)
  - Failure: No changes are applied

```
int *addr = xxx;
_xbegin();
int a = *addr;
*addr = a + 1;
_xend();
```

Hardware Transactional Memory

- **Wrap memory accesses with _xbegin() and _xend()**
  - Successful commit: all changes complete atomically (become globally visible)
  - Failure: No changes are applied

```
int *addr = xxx;
_xbegin();
_xend();
```

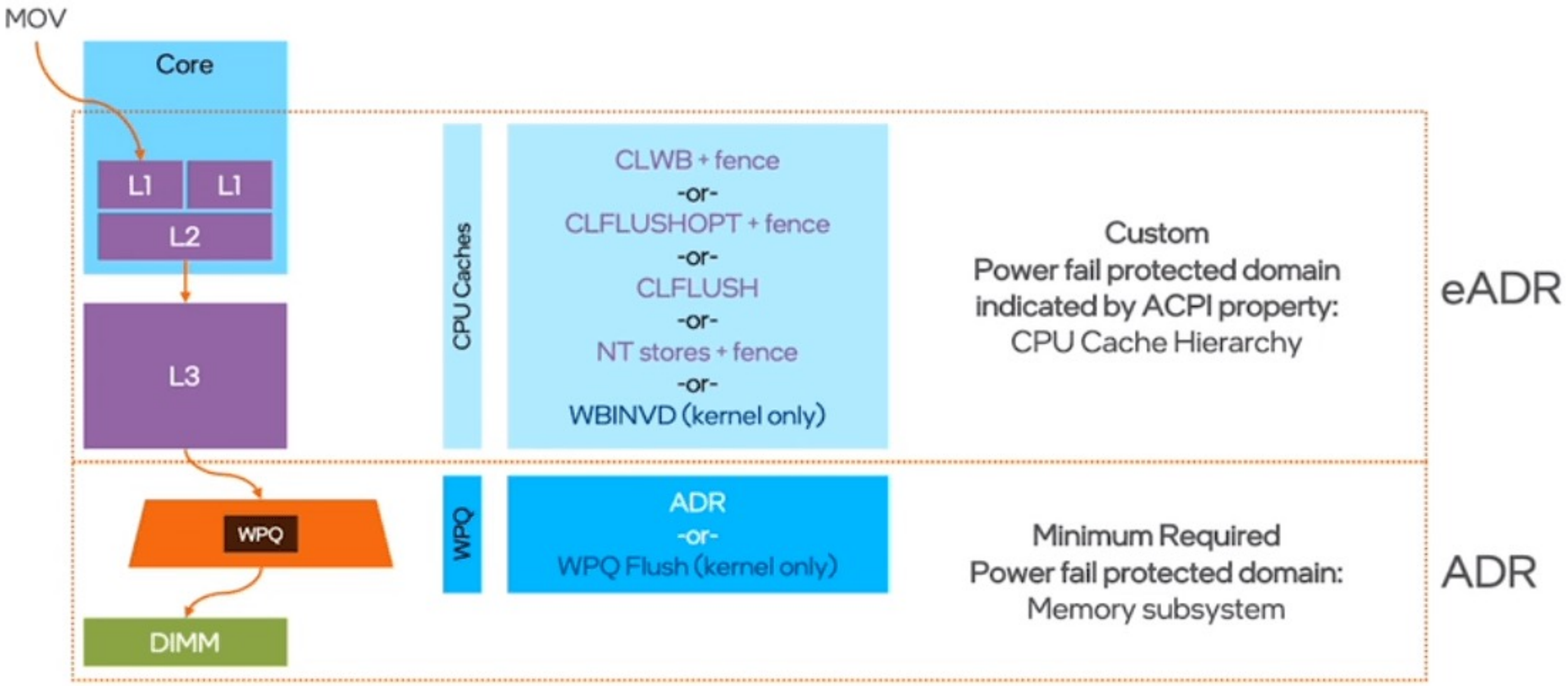
HTM inherently satisfies all-or-nothing semantics!

Hardware Transactional Memory

- **Wrap memory accesses with _xbegin() and _xend()**
  - Successful commit: all changes complete atomically (become globally visible)
  - Failure: No changes are applied
- **Reason for failure**
  - Limited write set (hardware limitation)
  - Memory access conflict
  - Cache line flush (cannot be used in persistent memory)
  - …

```
int *addr = xxx;
_xbegin();
int a = *addr;
*addr = a + 1;
_xend();
```

eADR: New opportunity



Ref: https://www.intel.com/content/www/us/en/developer/articles/technical/eadr-new-opportunities-for-persistent-memory-applications.html

Hardware Transactional Memory

- **Wrap memory accesses with _xbegin() and _xend()**
  - Successful commit: all changes complete atomically (become globally visible)
  - Failure: No changes are applied
- **Reason for failure**
  - Limited write set (hardware limitation)
  - Memory access conflict
  - Cache line flush
  - …

```
int *addr = xxx;
_xbegin();
int a = *addr;
*addr = a + 1;
_xend();
```

HTM can be used in persistent memory to guarantee crash consistency!

Crash Consistency Mechanism Comparison

Mechanism	Write Amplification	Write Set	Data Structure	Crash Consistency
In-place Update	1	Unlimited	Any	No guarantee
Journaling	>2	Unlimited	Any	Strong
Shadow Paging	>1	Unlimited	Dedicated	Strong
Soft Updates	1	Unlimited	Dedicated	Weak
Intel RTM	1	< 16K	Any	Strong

Challenges

- RTM is limited in both read and write set size, thus can easily abort due to file data copy.
- There are certain dependencies in the code paths of FS-related system calls.

Outline

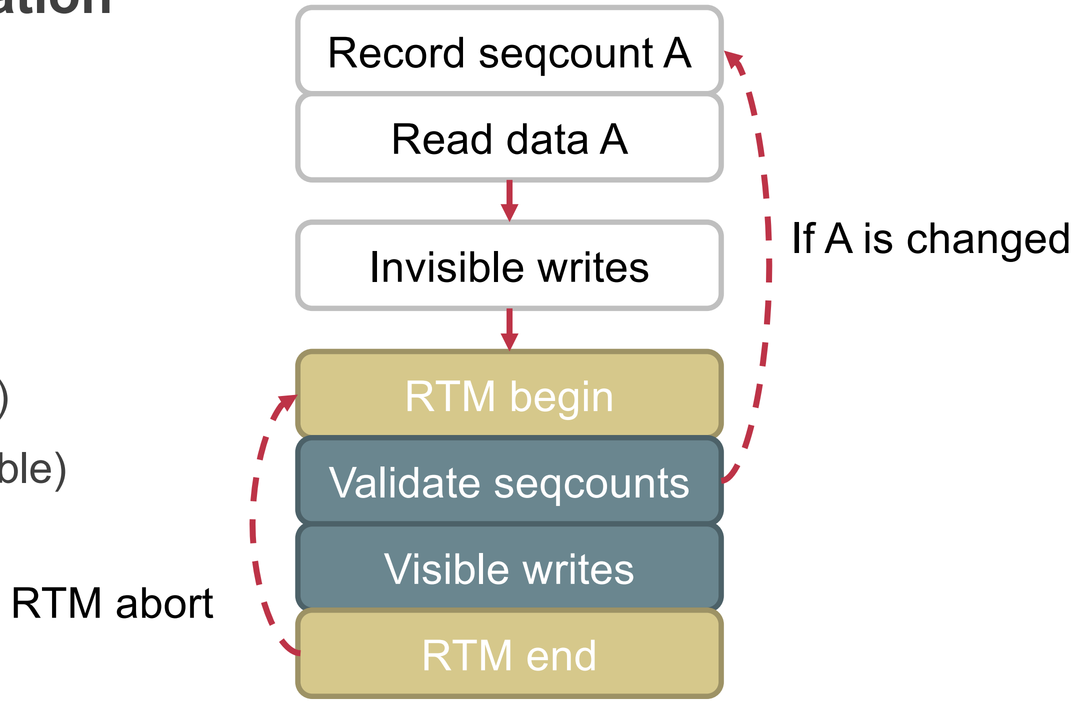
- **Background**
- **Design & Implementation**
  - HOP: a lightweight hardware-software cooperative mechanism
    - strong crash consistency
    - fine-grained concurrency control
  - Use HOP to build a strong crash consistency file system
- **Evaluation**

HOP (Hardware-assisted Optimistic Persistence)

- **Memory access classification**
  - Reads
  - Invisible writes
    - Updates that cannot be observed via the file system interface
  - Visible writes
    - Updates that can be observed by the file system interface
- **HOP only wraps visible writes with HTM**
- **Convert visible writes to invisible writes if needed**

HOP (Hardware-assisted Optimistic Persistence)

- **Memory access classification**
  - Reads
  - Invisible writes
  - Visible writes
- **write(fd, buf, size)**
  - Allocate new pages (invisible)
  - Copy data to new pages (visible)
  - Modify file metadata (visible)

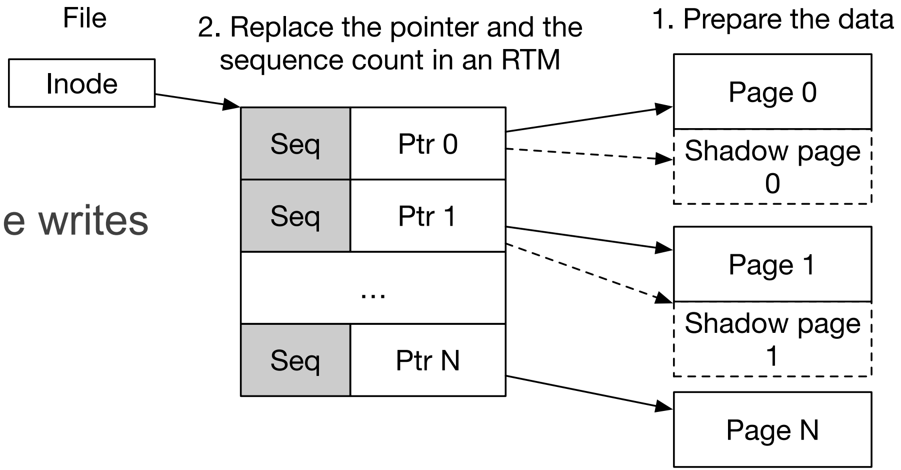


Use HOP to build HTMFS

- **Challenges**
  - The size of data write can exceed the write set size of HTM
  - Wrap memory allocation in HTM?
    - Yes: concurrent memory allocation may abort the transaction
    - No: Memory leak may happen after system crash
- **Optimization**
  - Improve the scalability
    - HTM can also handle concurrent accesses

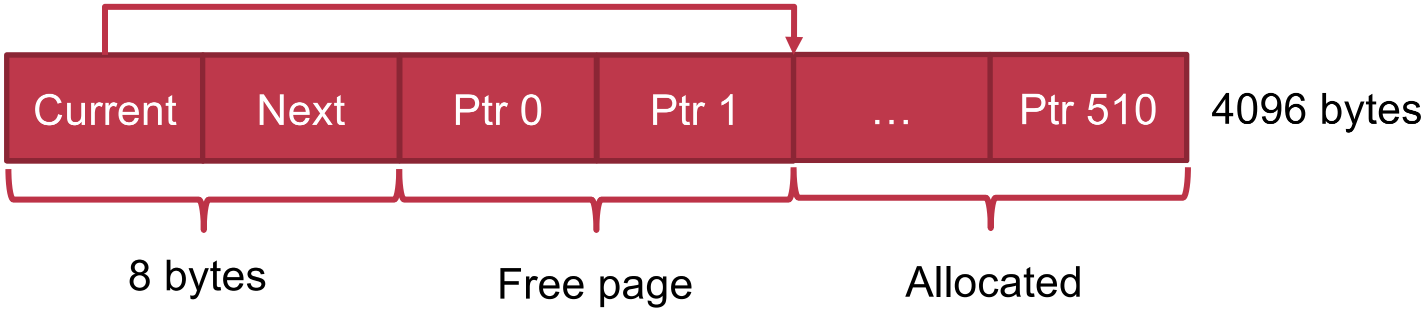
Data Accesses

- **Data Read**
  - Protected by sequence count
- **Single-page update**
  - Wrap the updates and metadata updates in a single transaction
- **Multi-page update**
  - Combined with shadow pages
  - Convert visible writes into invisible writes



Atomic Memory Allocator

- **Structure of a free list**



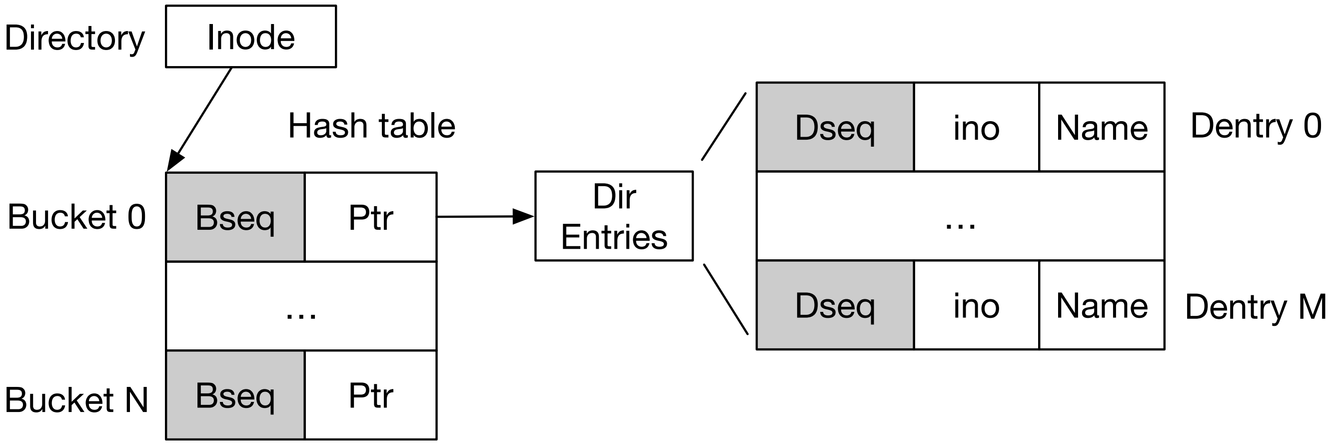
- **Per-thread allocator (no contention)**
  – Free list
  – Allocated list (NULL represents that all page allocation is persisted)

Atomic Memory Allocator

- **Allocate a page (not in HTM)**
  - Add a page from free list to allocated list
- **Persist memory allocation (in HTM)**
  - Drop allocated list
- **Revert memory allocation**
  - Link allocated list to free list

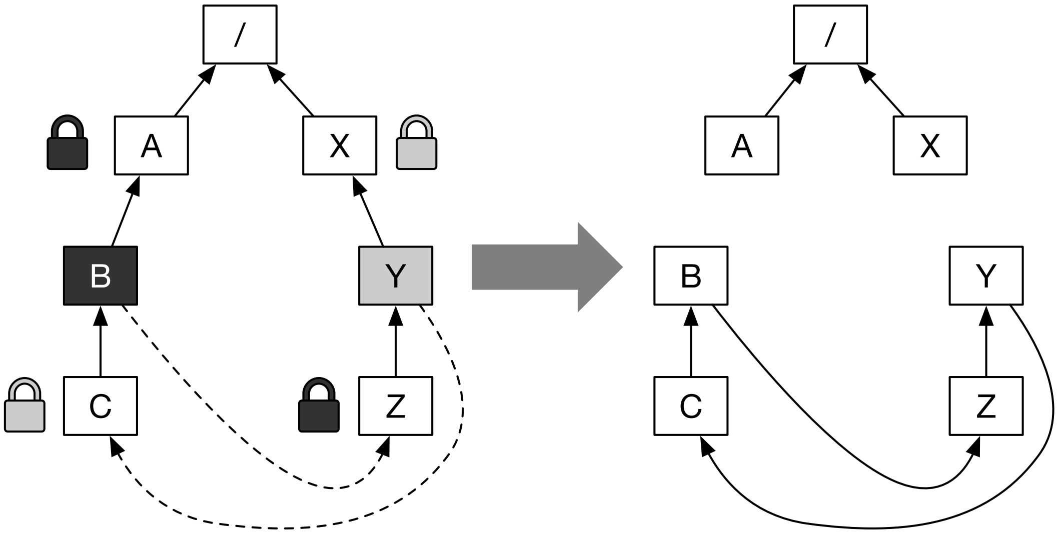
Improve Dentry Scalability

- **Scalability**
  - Different name distributes in different buckets, scaling well with threads increase



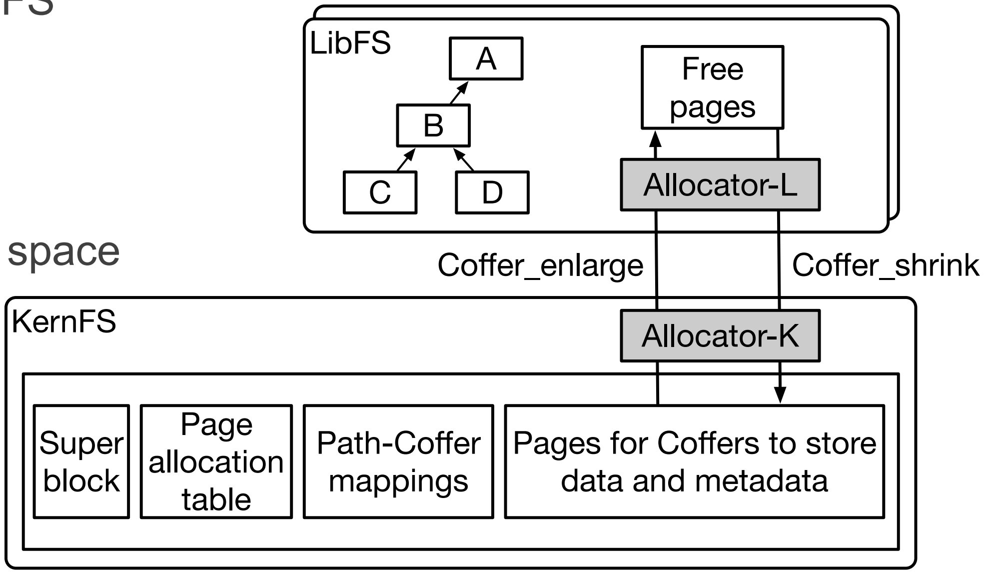
Prevent Rename Cycle

- **Correctness**
  - Check all sequence count in the whole path



Implementation based on ZoFS[1]

- **LibFS**
  - User-space FS libraries
  - All FS logic is implemented in LibFS
  - HTMFS only modifies LibFS
- **KernFS**
  - Protect global metadata and free space



[1] Mingkai Dong, Heng Bu, Jifei Yi, Benchao Dong, and Haibo Chen. Performance and protection in the zofs user-space nvm file system. SOSP 19, 2019.

Outline

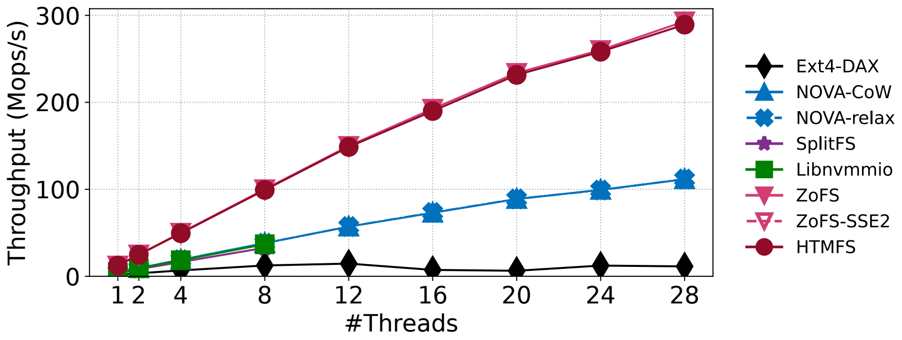
- **Background**
- **Design & Implementation**
  - HOP: a lightweight hardware-software cooperative mechanism
  - Use HOP to build a strong crash consistency file system
- **Evaluation**

Performance Evaluation

- **Evaluation setup**
  - Intel Xeon Gold 6330 CPU (28 cores) with hyper-threading disabled
  - 512GB DDR4 DRAM
  - 8*128GB Intel Optane Persistent Memory 200 series
- **Benchmarks**
  - Fxmark: read, data/metadata write in different contention level
  - TPCC on SQLite, LevelDB
  - …
- **File systems**
  - Ext4-DAX, NOVA-CoW, NOVA-relax, SplitFS, Libnvmmio (on NOVA), ZoFS

Evaluation: FxMark Data Read/Write

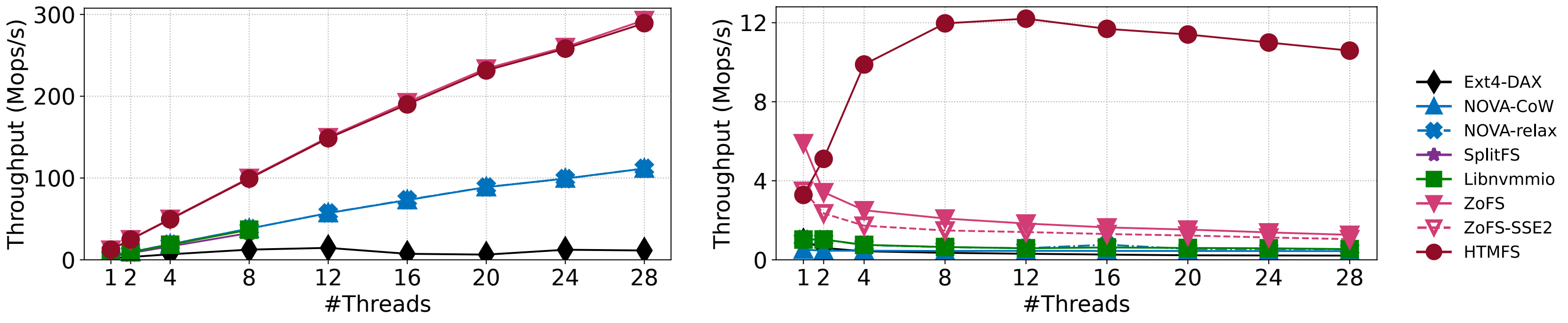
- Data Read: HTMFS's read performance is the **same** as ZoFS (weak crash consistency)



DRBL

Evaluation: FxMark Data Read/Write

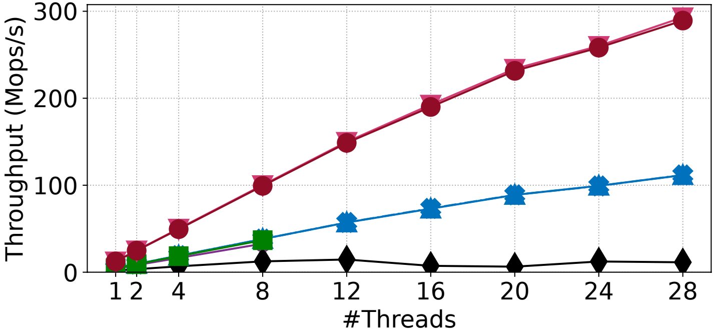
- Data Read: HTMFS's read performance is the same as ZoFS (weak crash consistency)
- Data Write (medium contention): HTMFS has **best performance and scalability**



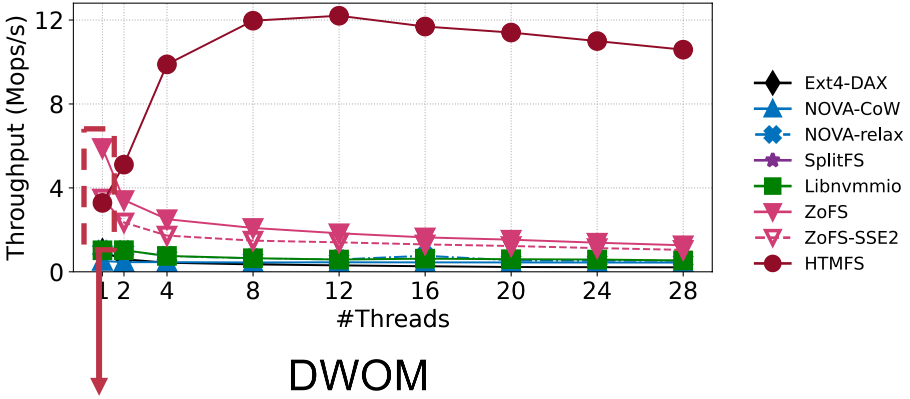
DRBL DWOM

Evaluation: FxMark Data Read/Write

- Data Read: HTMFS's read performance is the same with ZoFS (weak crash consistency)
- Data Write (medium contention): HTMFS has best performance and scalability



DRBL

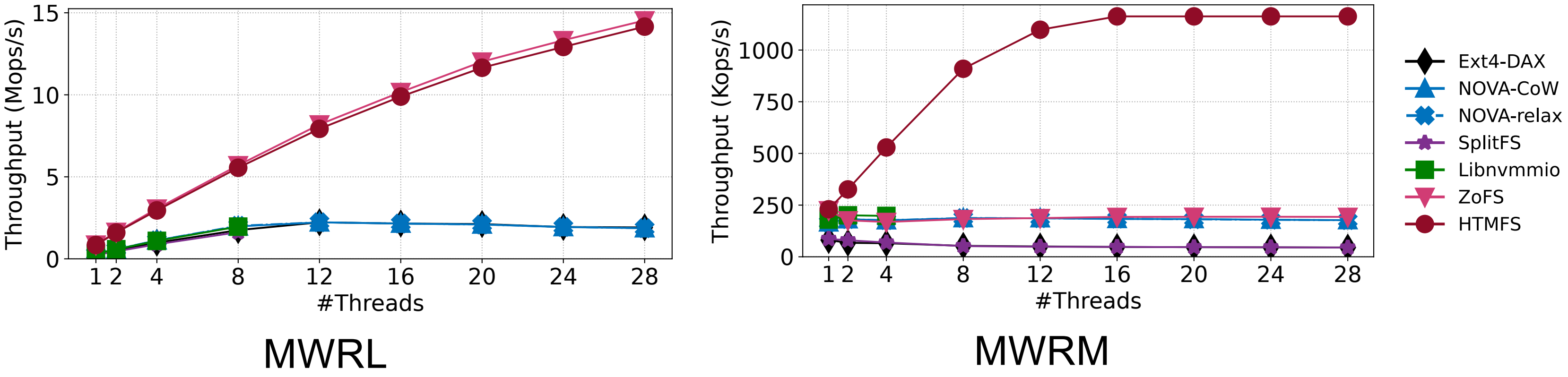


DWOM

REP-based MOV is faster than SSE2-based MOV in this workload (cache hit only)
ZoFS-SSE2's performance is the same with HTMFS

Evaluation: FxMark Rename

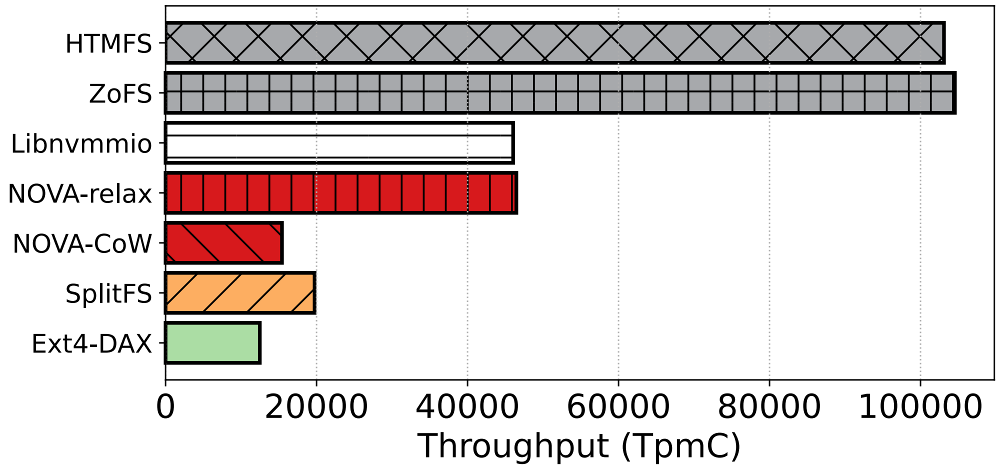
- Low contention: HTMFS has similar performance with ZoFS
- Medium contention: HTMFS is better than others



- The performance of HTMFS is similar or even better than that of a weak crash consistency file system (ZoFS)!

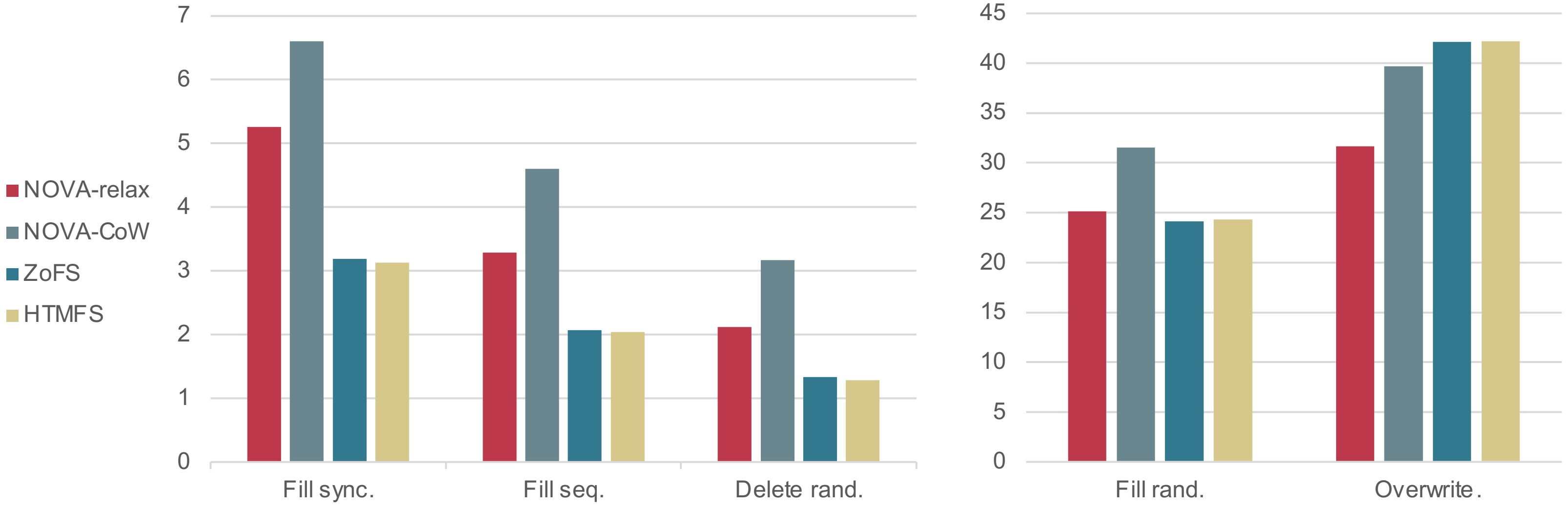
Evaluation: Real-world Applications

- TPCC on SQLite (higher is better)
  - HTMFS is as good as ZoFS while NOVA-relax is much better than NOVA-CoW



Evaluation: Real-world Applications

- LevelDB (latency/us, lower is better)
  - HTMFS is as good as ZoFS while NOVA-relax is much better than NOVA-CoW



- HTMFS gets strong crash consistency for nearly free

Conclusion

- HTM can guarantee crash consistency for PM file systems on the eADR platforms
- The write set of HTM is limited, making it difficult to use HTM directly to build a PM file system
- We propose HOP, a lightweight hardware-software cooperative mechanism, to provide both strong crash consistency and fine-grained concurrency control
- We apply HOP to build the first HTM-based user-space file system, HTMFS
- The performance of HTMFS is comparable to or better than file systems that only provide weak crash consistency

IPADS
INSTITUTE OF PARALLEL AND DISTRIBUTED SYSTEMS

Thanks!