32nd International Conference on Massive Storage Systems and Technology (MSST 2016)
May 2 - 6, 2016

Fine-grained Metadata Journaling on NVM

Cheng Chen, **Jun Yang**, Qingsong Wei, Chundong Wang, and Mingdi Xue

Data Storage Institute, A*STAR, Singapore

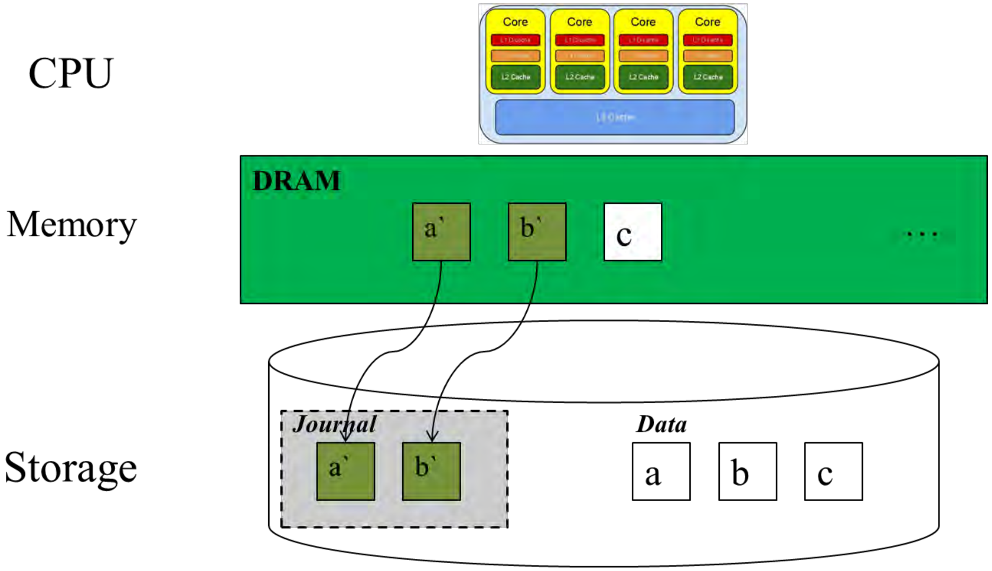
CREATING GROWTH, ENHANCING LIVES

A*STAR Data Storage Institute

Introduction

- Journaling file system
  - Write a “journal” to a circular log area before updating actual content
  - Can be **metadata only** or both metadata and data

- Problems
  - Performance penalty
  - Inefficient journal writes due to block-based interface



Overview

- Enable journaling has performance penalty

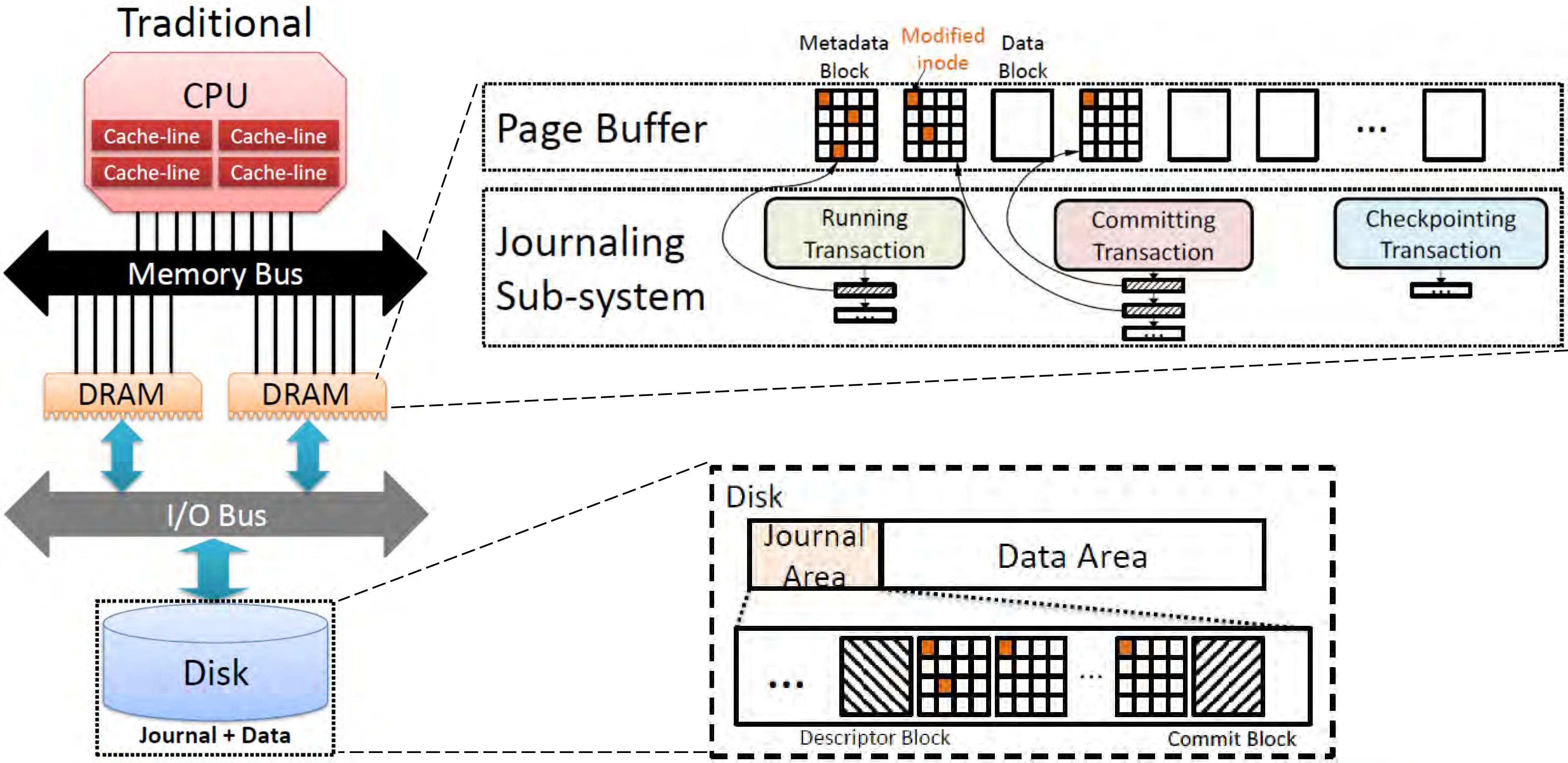
- Our observation
  - Around ~40% performance drop under common workloads
  - Journal write amplification due to block-based design
    - E.g. few inode changes cause the **entire** inode block to be written

- Next generation of non-volatile memory (NVM)
  - DRAM-like byte-addressability and performance + persistency
  - But journaling on NVM still costs ~35% performance drop
  - **How to improve? Eliminate journal write amplification**

- Our solution: ***Fine-grained metadata journaling***
  - A new journal format to fully utilize the byte-addressable of NVM
  - Redesign the journaling process to reduce the writes
  - Reduce more than **90%** unnecessary journal writes
  - Achieve up to **15x** performance improvement under different workloads

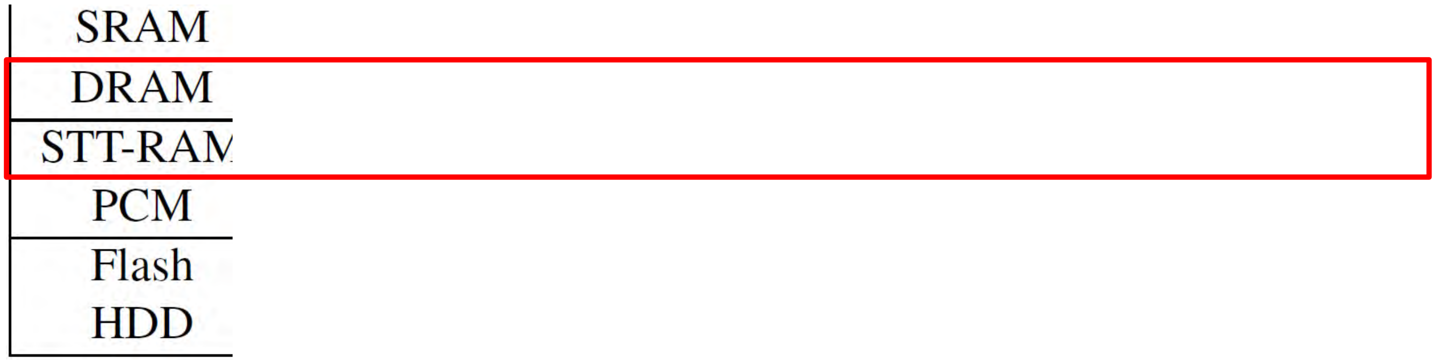
Background

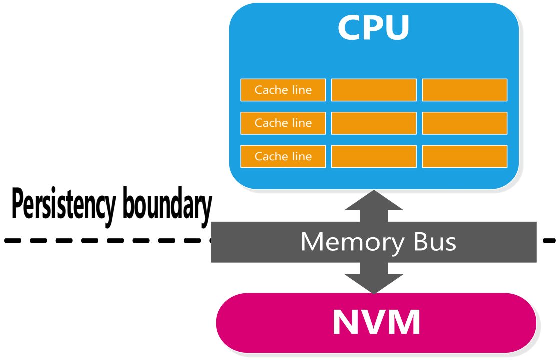
Conventional Journaling File System



Background

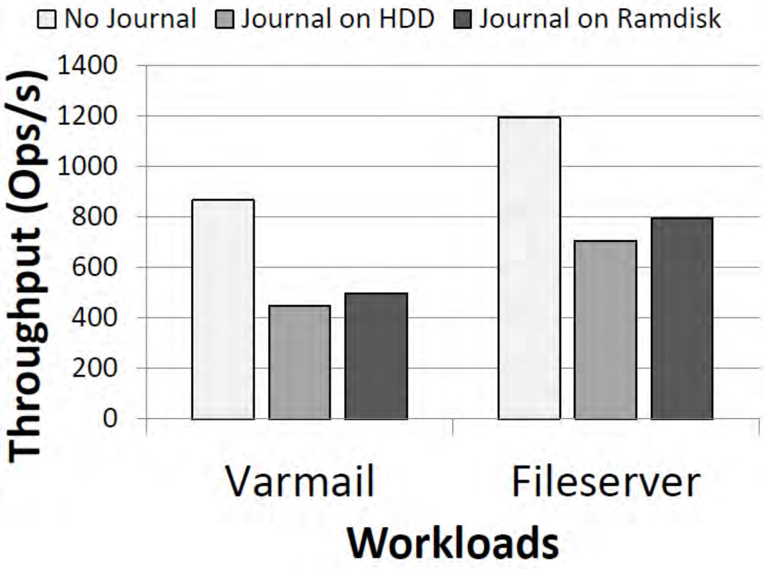
- NVM (Next Generation of Non-volatile Memory)
  - Provides DRAM-like performance and disk-like persistency





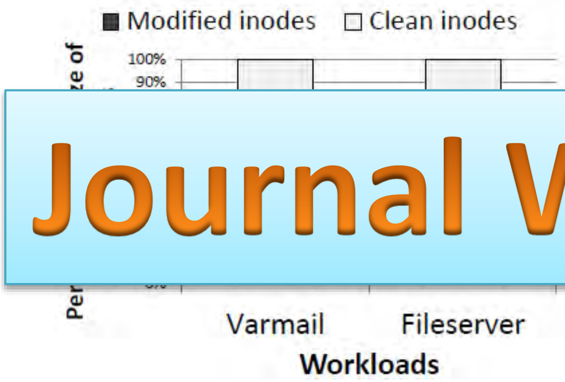
- Data consistency in NVM requires ordered memory writes
  - Non-trivial due to CPU design
    - E.g, ***w1***, (MFENCE,CLFLUSH,MFENCE), ***w2***, (MFENCE,CLFLUSH,MFENCE)

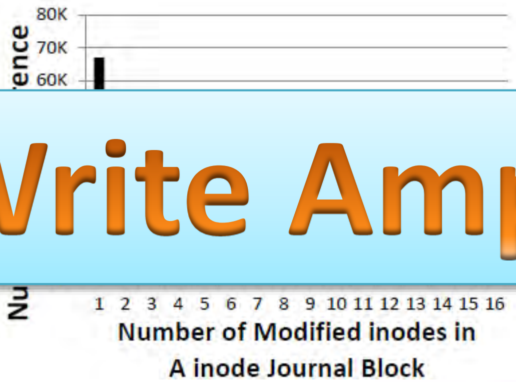
Motivation



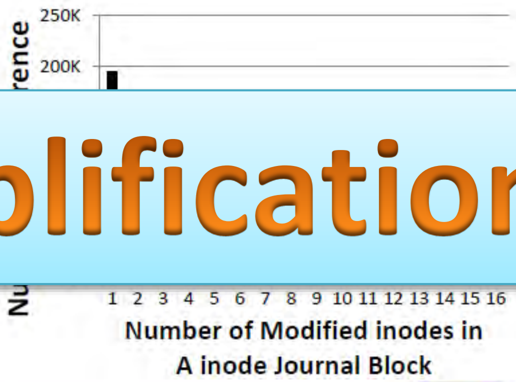
High Journaling Overhead

	Varmail	Fileserver
HDD	↓48.2%	↓40.9%
Ramdisk	↓42.5%	↓33.6%





Varmail



Fileserver

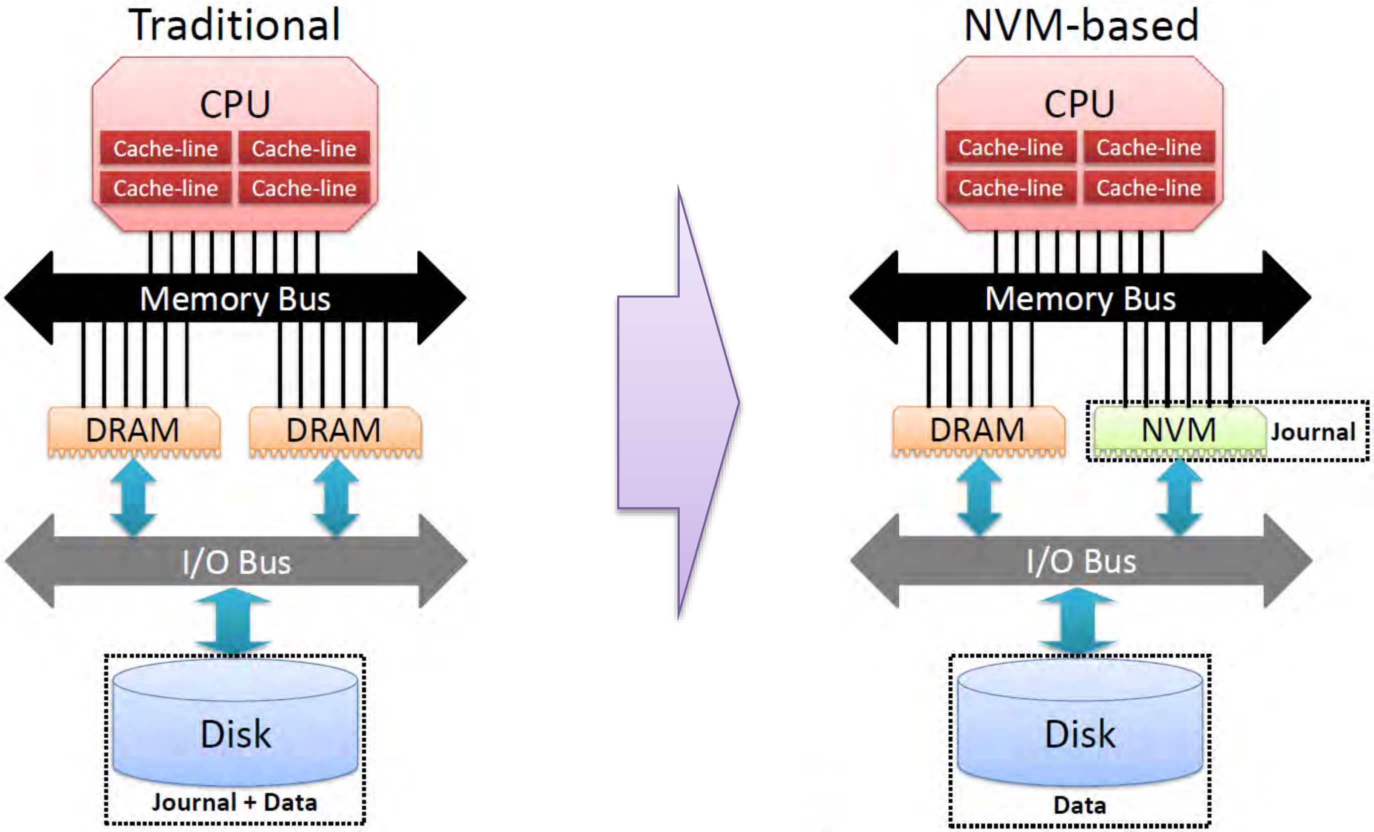
Journal Write Amplification

Design Decisions

I. Use NVM as the journaling device

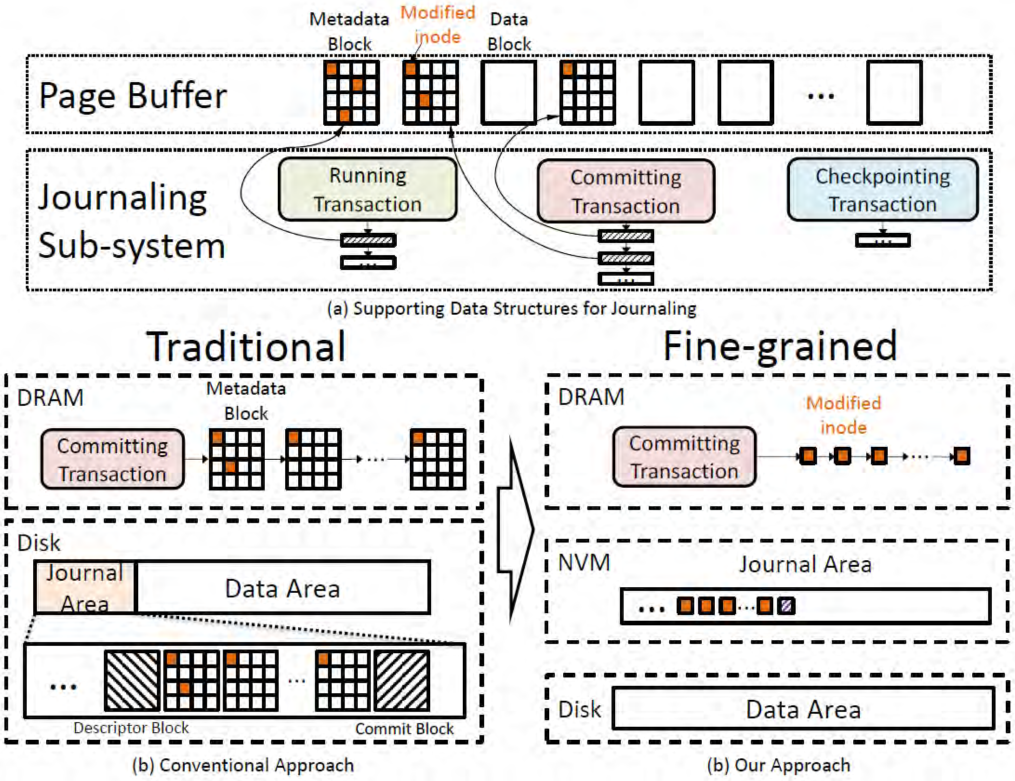
II. Utilize the byte-addressability to eliminate the **journal write amplification**

III. Further reduce the journal writes that requires ordered memory writes



Our Solution

Fine-grained Metadata Journaling

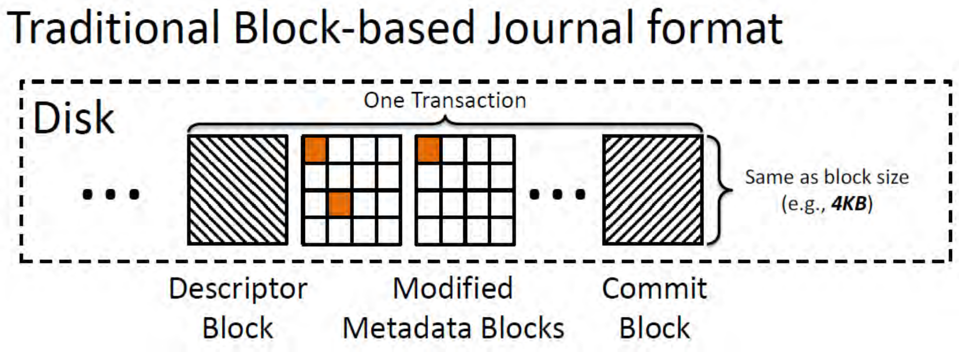


- Move all the journal to NVM
- Use inode as the basic unit for journaling

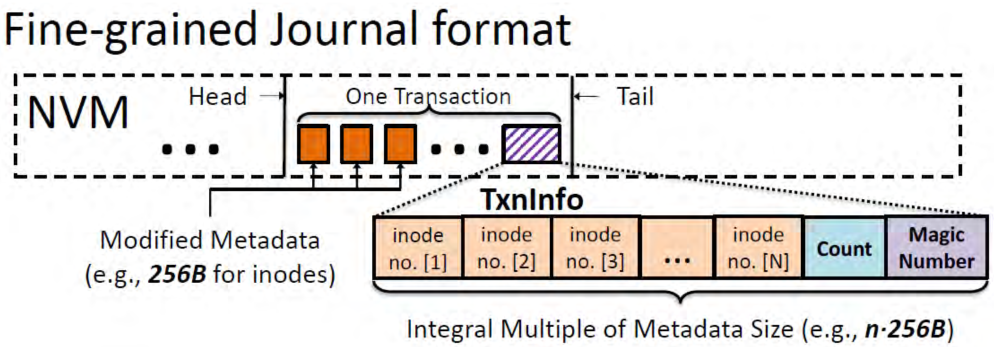
CREATING GROWTH, ENHANCING LIVES

A*STAR Data Storage Institute

Fine-grained Journal Format



- Traditional approach
  - Block-based
  - Descriptor/Commit Block
  - Wasted space and writing time

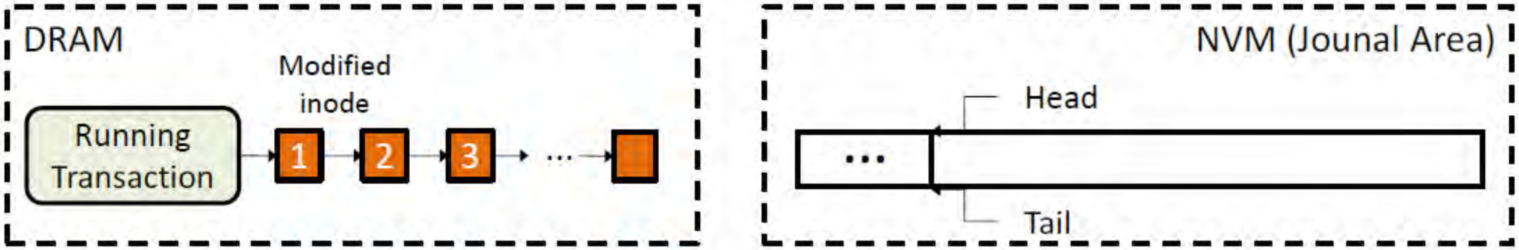


- **TxnInfo**
  - CPU-cache friendly
  - Configurable size
  - Consistent

CREATING GROWTH, ENHANCING LIVES

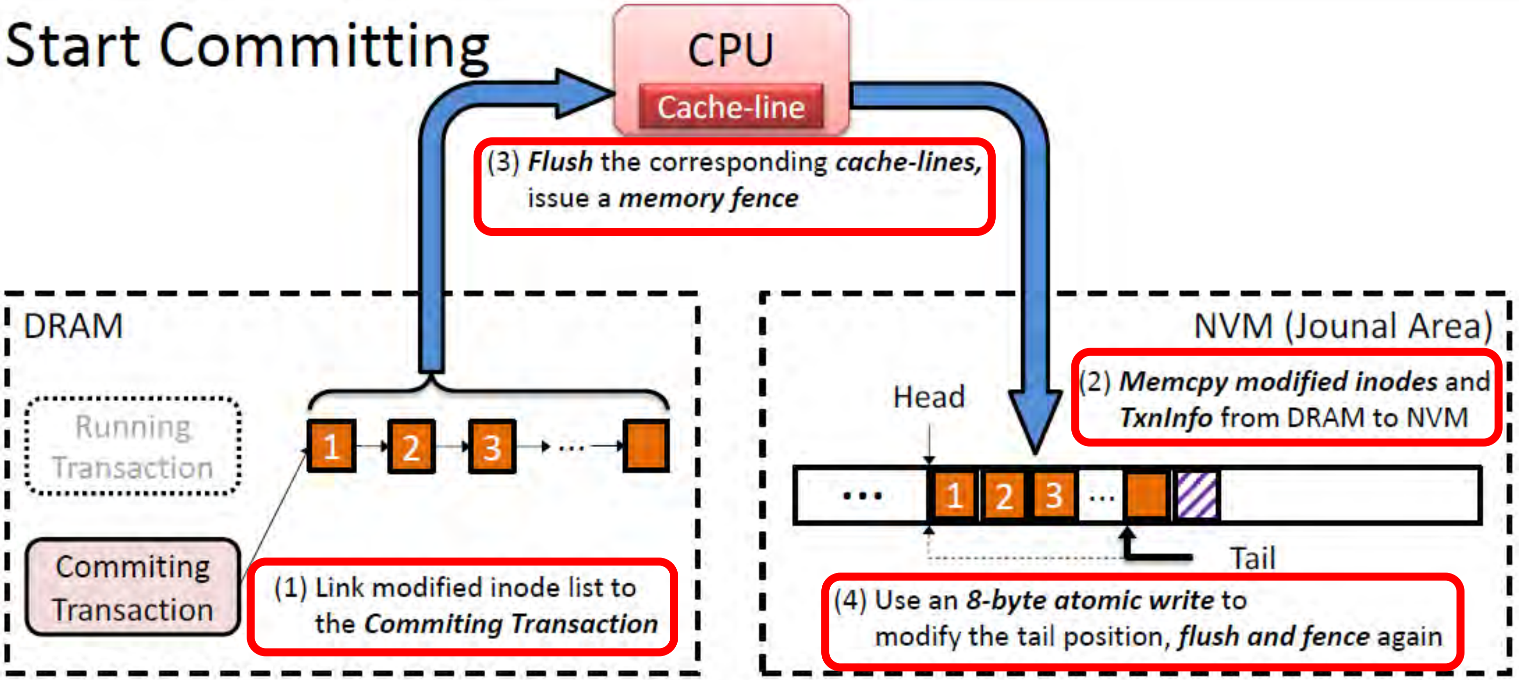
A*STAR Data Storage Institute

Optimized Workflow - Commit



Before Committing

Start Committing



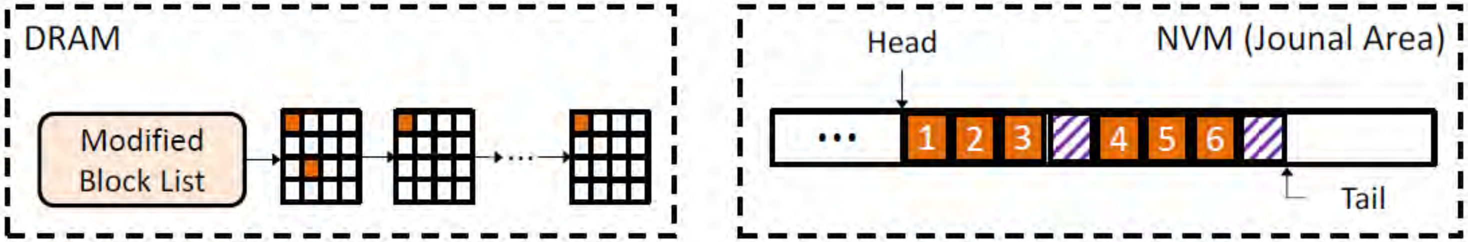
(1) Link modified inode list to the *Commiting Transaction*

(2) ***Memcpy modified inodes*** and ***TxnInfo*** from DRAM to NVM

(3) ***Flush*** the corresponding ***cache-lines,*** issue a ***memory fence***

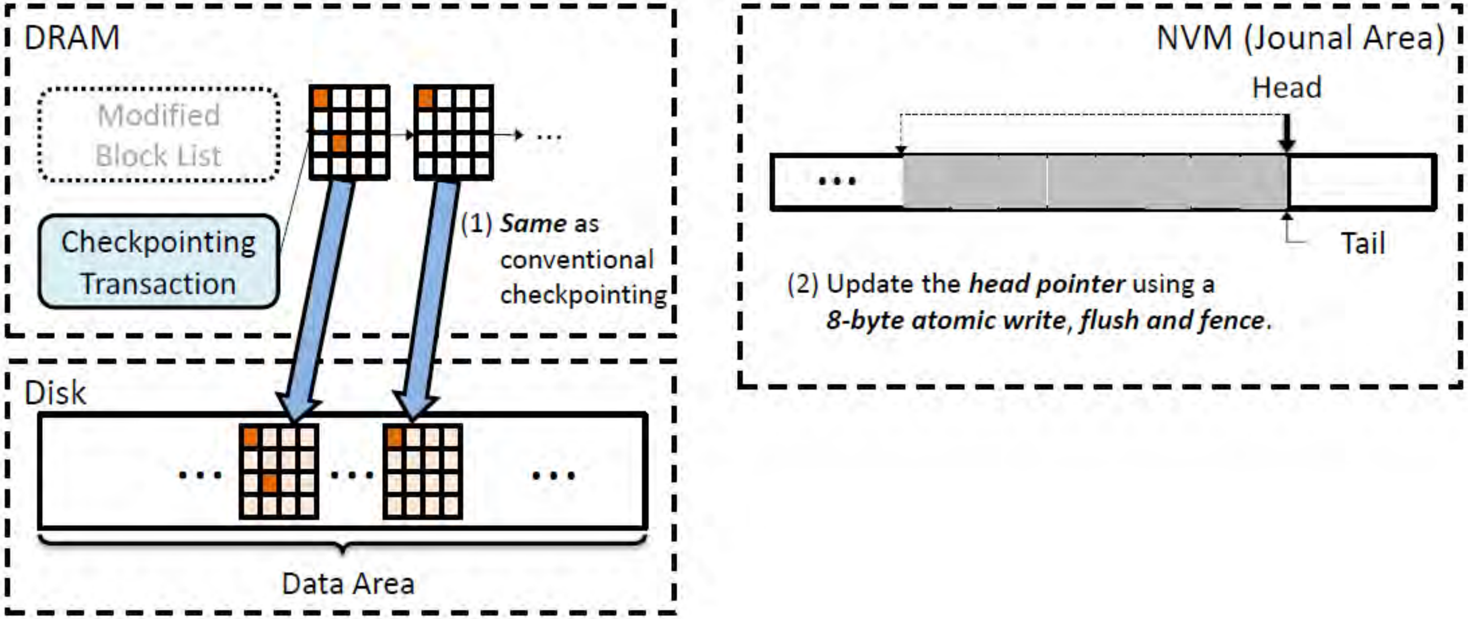
(4) Use an ***8-byte atomic write*** to modify the tail position, ***flush and fence*** again

Optimized Workflow - Checkpoint



Before Checkpointing

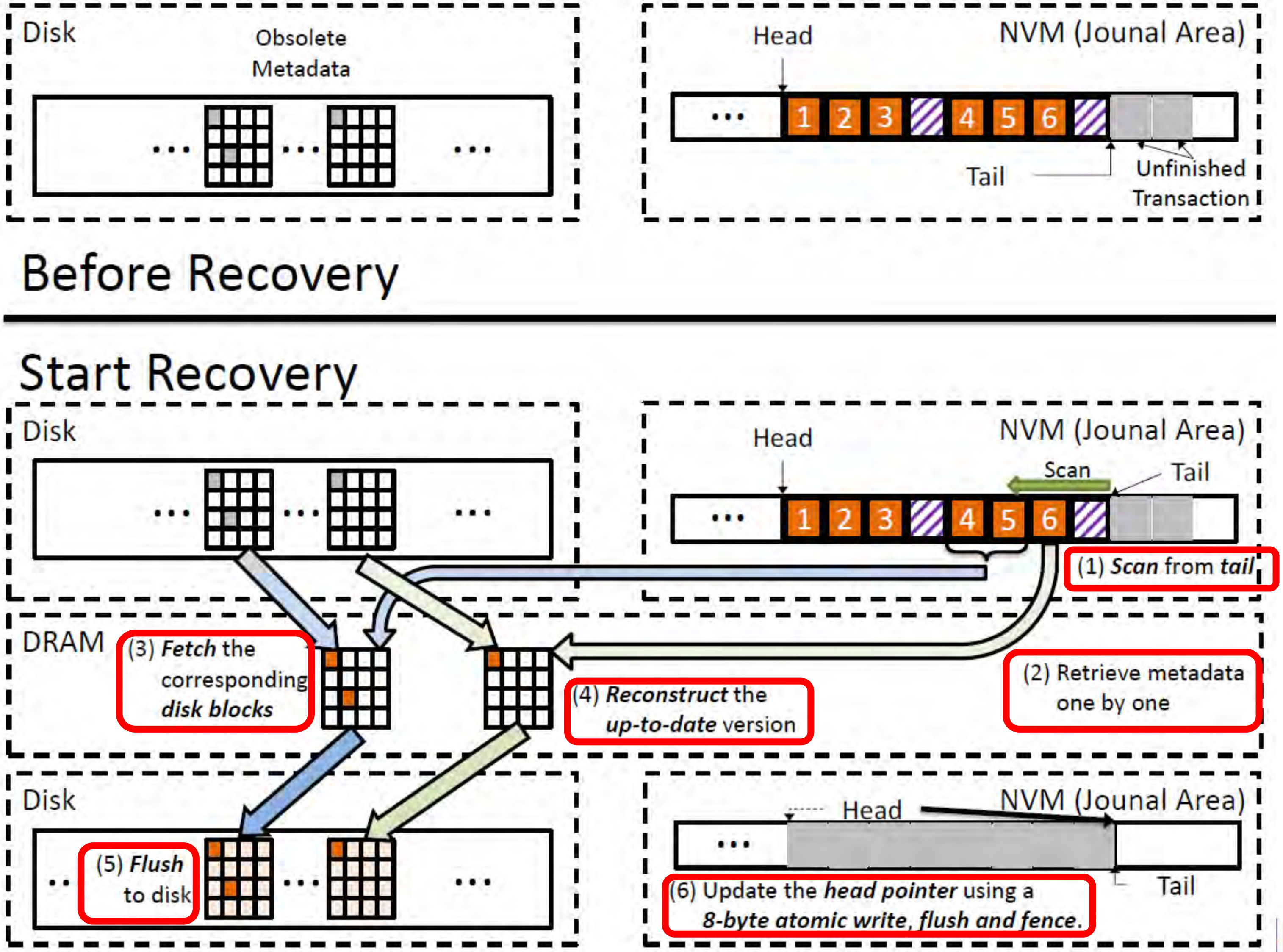
Start Checkpointing



(1) ***Same*** as conventional checkpointing

(2) Update the ***head pointer*** using a ***8-byte atomic write, flush and fence***.

Optimized Workflow - Recovery



(1) *Scan* from ***tail***

(2) Retrieve metadata one by one

(3) ***Fetch*** the corresponding ***disk blocks***

(4) ***Reconstruct*** the ***up-to-date*** version

(5) ***Flush*** to disk

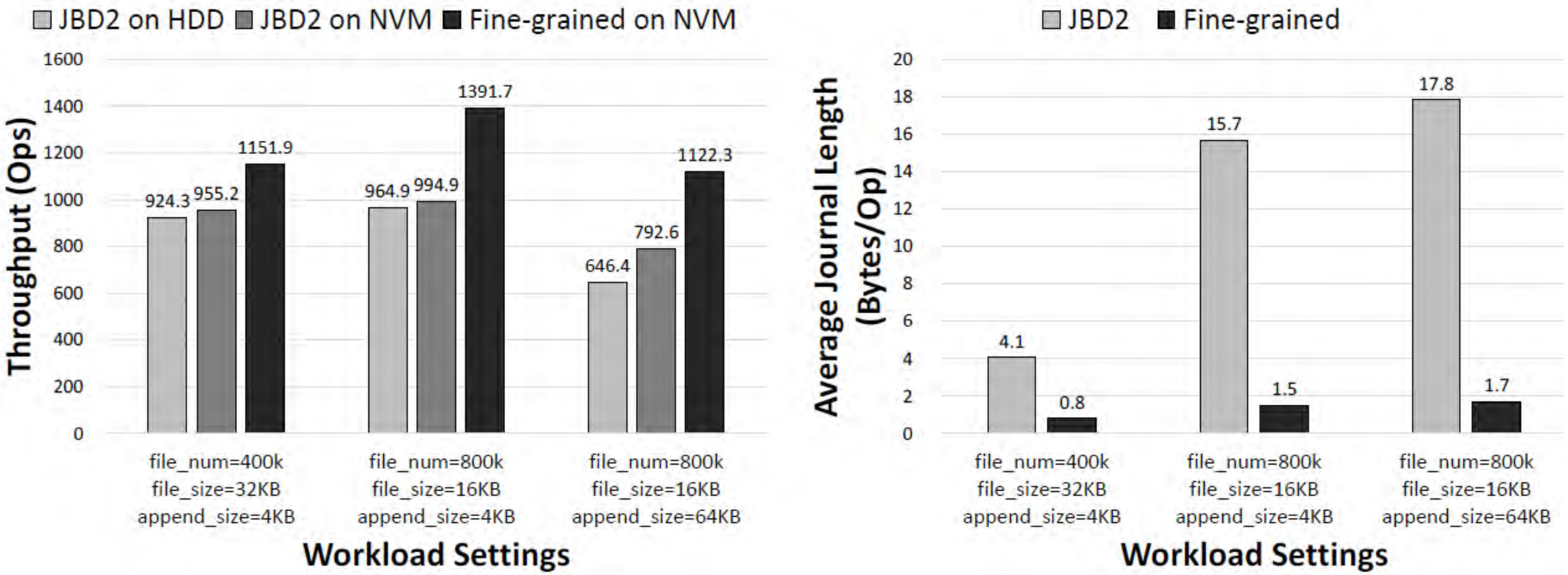
(6) Update the ***head pointer*** using a ***8-byte atomic write, flush and fence.***

Experimental Setup

- NVDIMM server
  - Intel Xeon E5-2650
    - 2.4GHz, 512KB/2MB/20MB L1/L2/L3 Cache
  - 4GB DRAM, 4GB NVDIMM
    - NVDIMM has the same performance as DRAM
  - 300GB 15K-RPM HDD x 2

- Testing target
  - Baseline: Ext4 with JBD2 on Disk
    - “ordered” mode
  - Ext4 with JBD2 on NVM
    - Still block-based
    - Use memcpy with CLFLUSH and MFENCE
  - **Our solution**
    - Modified JBD2 with new log format and commit, checkpoint, recovery process
    - Write journal to NVM through memcpy with CLFLUSH and MFENCE

Performance Result (1)



(a) Throughput

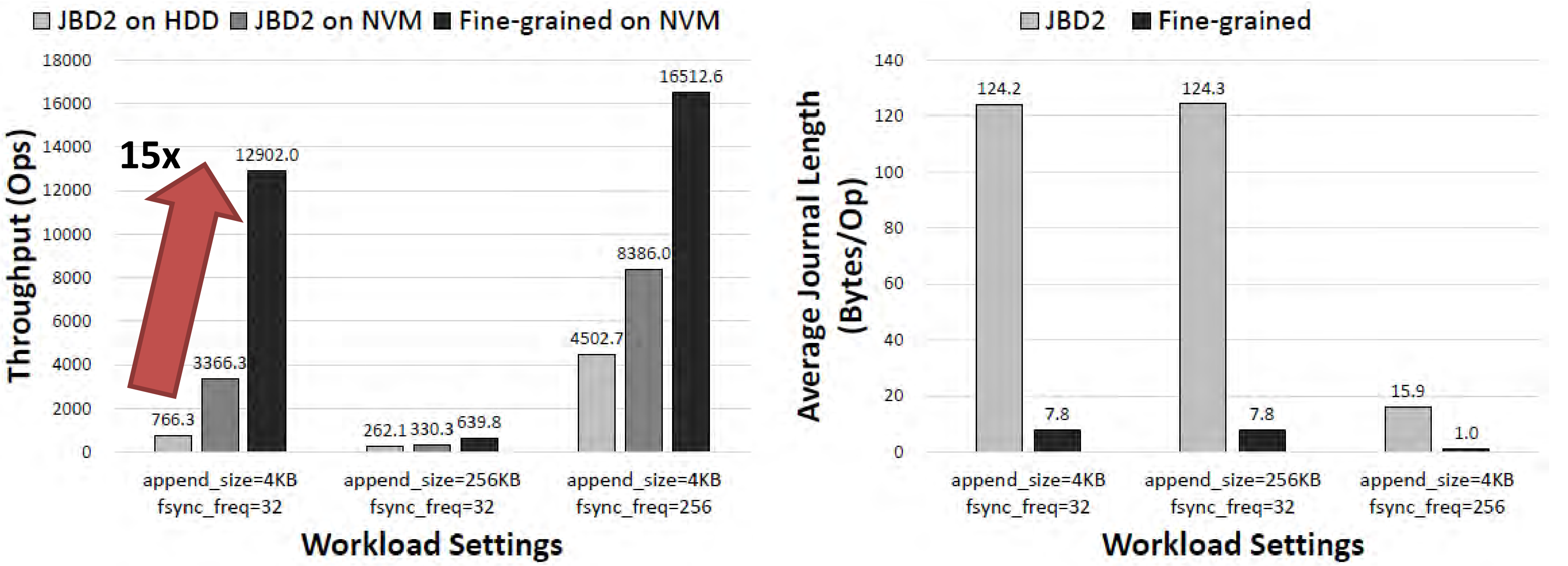
(b) Journal Writes

Fileserver Workloads

Performance Improvement	
Conventional Journaling on **HDD**	Conventional Journaling on **NVM**
↑73.6%	↑41.6%

Journal Write Reduction
Block-based Journaling
↓**90.4%**

Performance Result (2)



(a) Throughput

(b) Journal Writes

FileMicro_Writefsync Workloads

Performance Improvement	
Conventional Journaling on **HDD**	Conventional Journaling on **NVM**
↑**15.8x**	↑**2.8x**

Journal Write Reduction
Block-based Journaling
↓**93.7%**

More in The Paper

- Performance of other workloads
  - FileBench – Varmail
  - Postmark

- Impact of the size of TxnInfo
  - Commit behavior
  - Overall throughput tuning

Conclusion

- We reveal the **journal write amplification** problem
  - Mainly due to the block interface
  - Journaling penalty is still high with high-performance NVM as journal device

- We propose **Fine-grained Metadata Journaling**
  - Exploit the **byte-addressability** and high-performance of **NVM**
  - A new fine-grained journal format
    - CPU-cache friendly
    - Further reduce the amount of journal writes
  - Modified workflow of commit, checkpoint and recovery in journaling

- Achieve up to **15x** performance boost under different workloads

THANK YOU!

Q & A

Jun Yang

Email: yangju@dsi.a-star.edu.sg