Möbius: A High Performance Transactional SSD with Rich Primitives

Wei Shi, Dongsheng Wang,
Zhanye Wang, Dapeng Ju
Tsinghua University

Microprocessor and SoC Technology R&D Center

Summary

- **Challenge**: Software transaction processing schemes might not be suitable for a out-of-place update NAND SSD
- **Our goal**: propose a new high performance transactional SSD architecture with rich primitives
- **Observation**
  - **Serialized transaction processing**: caused by ordered transaction recovery
  - **Long recovery time**: caused by scanning unfinished transactions
  - **Extra Sudden Power-Off Recovery (SPOR) logic**: lived in SSD FTL
- **Key Ideas**
  - **Atom file**: to abstract transaction into a "file"
  - **DAG commit protocol**: by skipping unnecessary scanning
  - **Recovery logic combination**: by combining SPOR with transaction aborting
- **Möbius: a new transactional SSD architecture**
  - **Rich primitives: support both static and dynamic transactions**
  - **Avoid unnecessary scanning by DAG verification method**
  - **Recover FTL and transaction processing logic after power failures**
- **Results:** Möbius expect to save 4~29 times of recovery time and offer a 67% higher throughput than other transactional SSD designs

Outline

- **Motivations**
- Möbius Design
- Implementation
- Evaluations
- Conclusions

Jim Gray (I)

What is a transaction?

A serial of operations must succeed or fail as a complete unit.

- **Atomicity**
- **Consistency**
- **Isolation**
- **Durability**

Software transaction processing schemes

a) Write-ahead logging (WAL)

- InnoDB (MySQL)
- PostgreSQL
- JBD (Ext 3 and Ext 4)

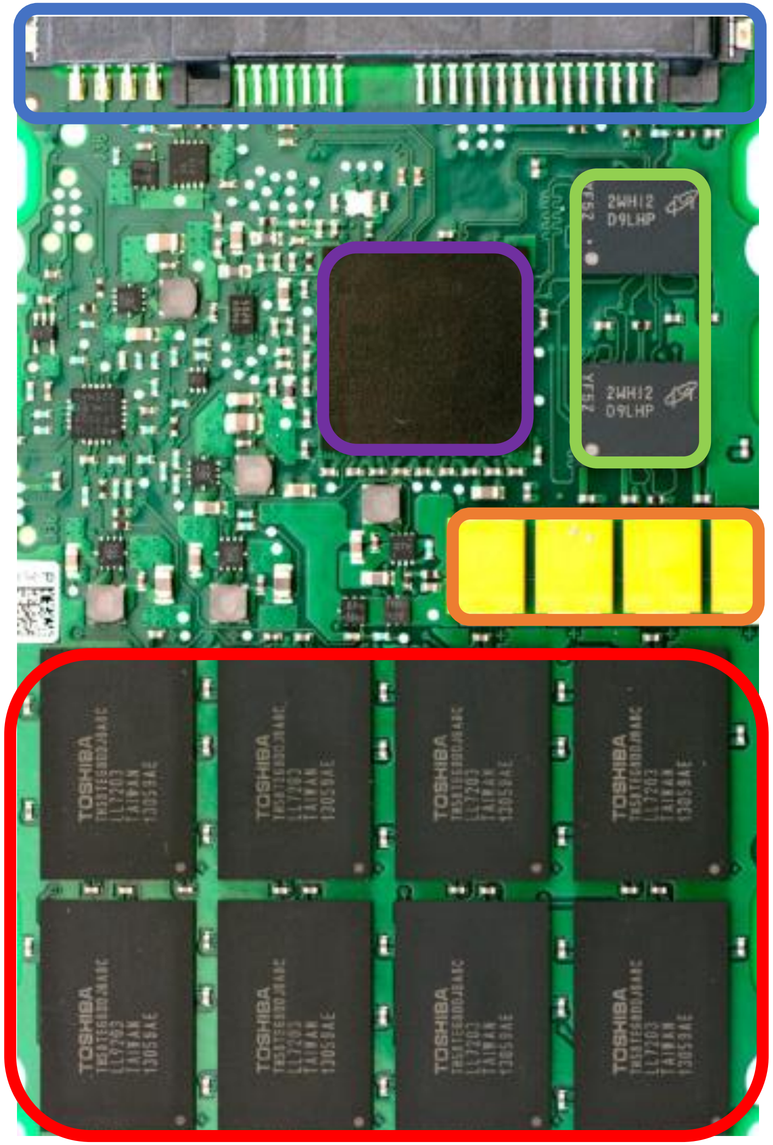
b) Shadow paging

- ZFS

Jim Gray (II)

- Tape is Dead
- Disk is Tape
- ***Flash is Disk***

NAND Flash SSDs



SSD Components
NAND flash packages
Host interface controller
Microprocessor
DRAM (buffers + FTL cache)
Flash controllers

NAND Flash SSDs

Write (p, RED)

- Allocating a physical page
- Updating mapping table

Write (p, GREEN)

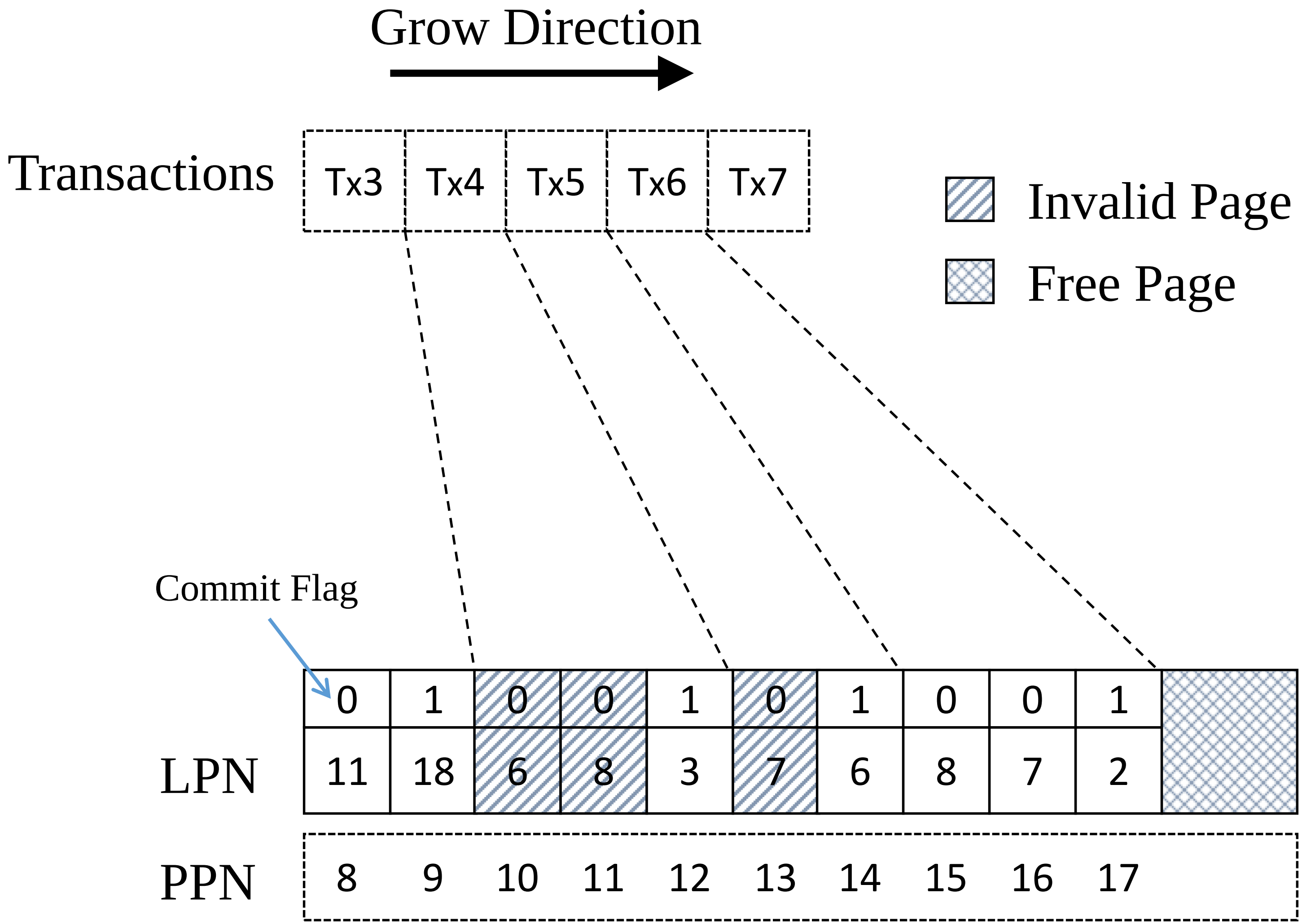
- Allocating a physical page
- Updating mapping table

Out-of-place Write

Existing Transactional SSD Designs

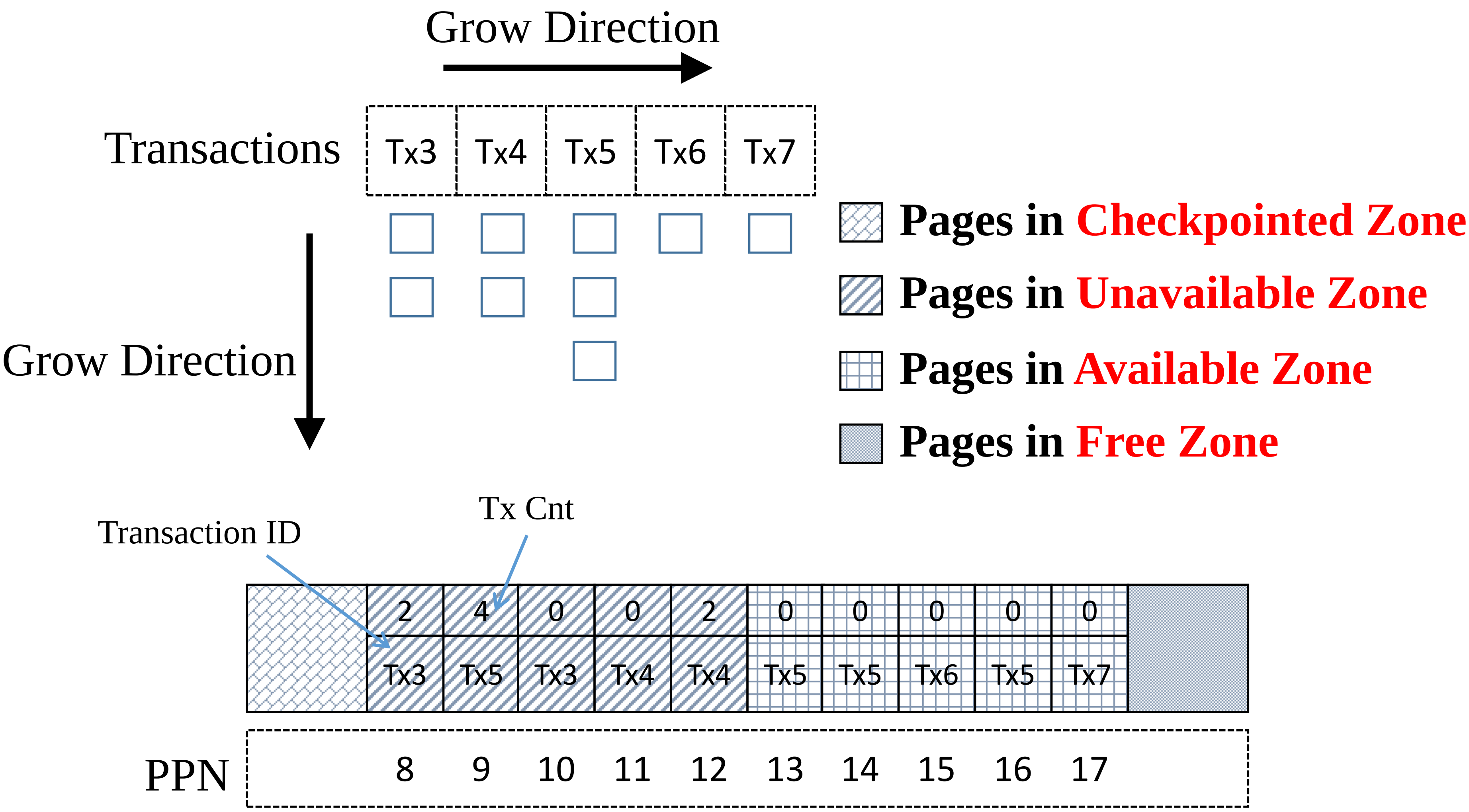
- TxFlash (without persistent FTL)
- Atomic-Write
- LightTx
- MARS (NVM SSD)

Write-Atomic (HPCA 2011)



	0	1	0	0	1	0	1	0	0	1	
LPN	11	18	6	8	3	7	6	8	7	2	
PPN	8	9	10	11	12	13	14	15	16	17	

LightTx (ICCD 2013)



Grow Direction

Transactions: Tx3, Tx4, Tx5, Tx6, Tx7

Grow Direction

- ▨ Pages in **Checkpointed Zone**
- ▨ Pages in **Unavailable Zone**
- ▦ Pages in **Available Zone**
- ▩ Pages in **Free Zone**

Transaction ID

Tx Cnt

	2	4	0	0	2	0	0	0	0	0
	Tx3	Tx5	Tx3	Tx4	Tx4	Tx5	Tx5	Tx6	Tx5	Tx7
PPN	8	9	10	11	12	13	14	15	16	17

Problems of state-of-the-art transactional SSDs

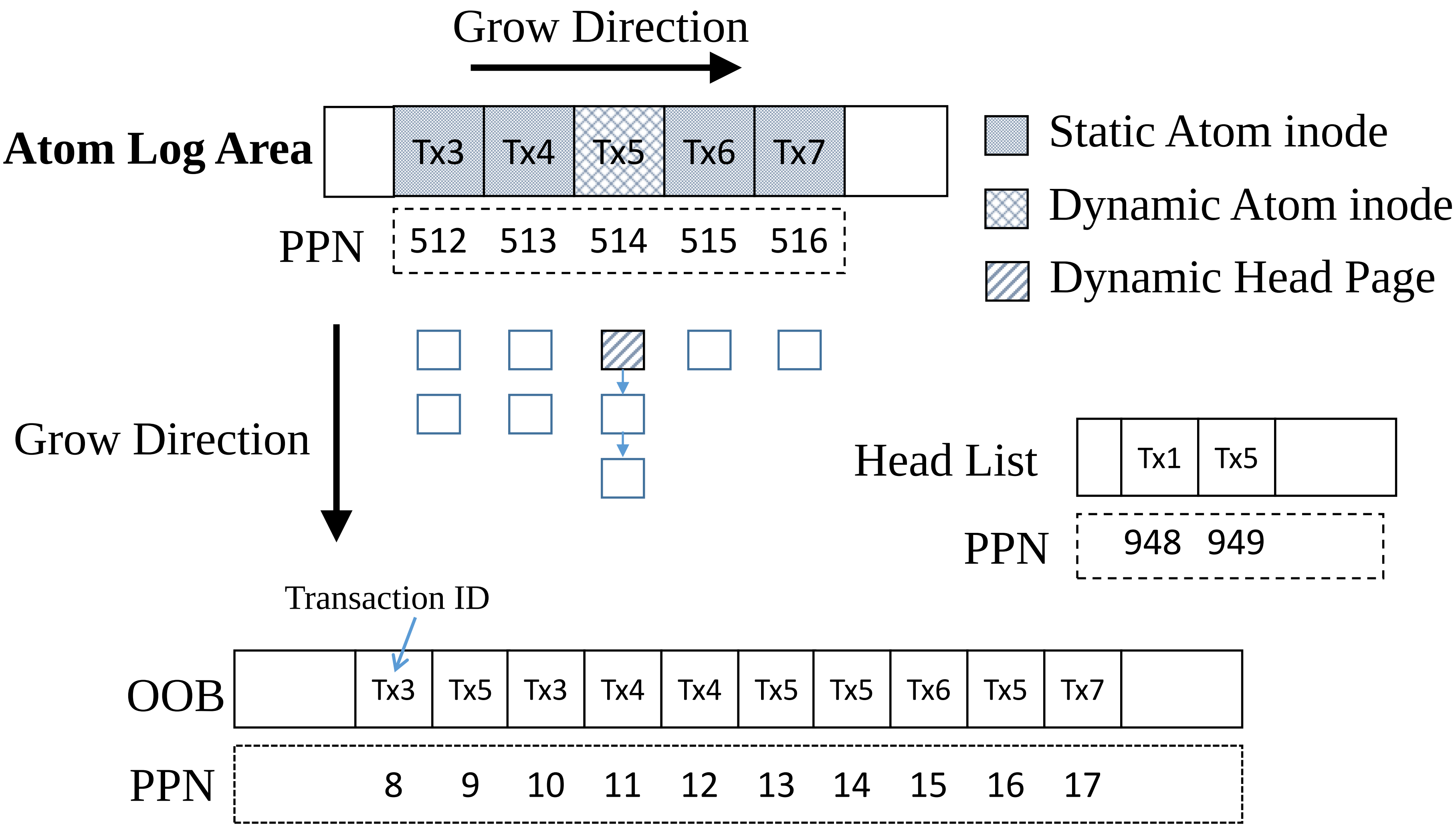
- Limit the parallelism of SSD

Since in SSD performance mainly benefits by internal parallelism, serialized transaction processing limits the whole SSD performance.

- Long time scanning

Recovery is based on unselective scanning which is very expensive.

Möbius (our design)



Outline

- Motivations
- **Möbius Design**
- Implementation
- Evaluations
- Conclusions

Host interface

- **WRITE(p)**
- **READ(p)**
- **SWRITE(uuid, p1, …, pn)**
- **SREAD(p)**
- **DWRITE(p, flag)**
- **ABORT(uuid)**

Static transactions

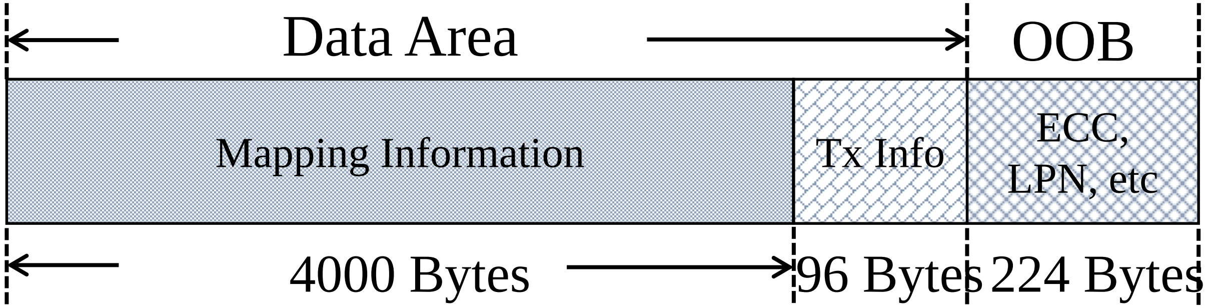
- Transaction that all data manipulated in the transaction is determined before the transaction begins, e.g., all data are already in system block/page cache

Dynamic transactions

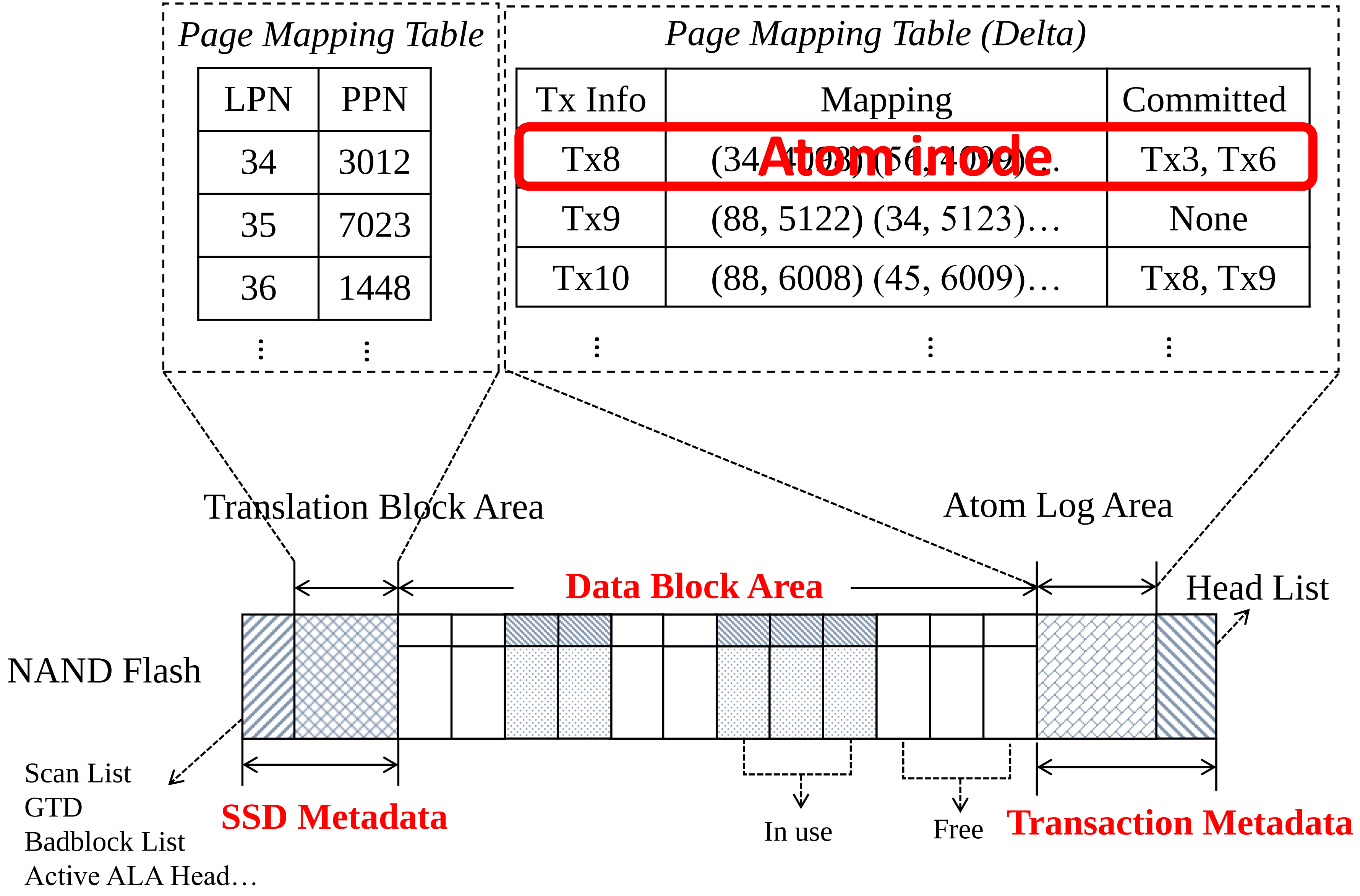
- Transaction that all data operations in this transaction are not determined when it begins

Atom inode

A 4 KB physical page example



Möbius architecture overview



Page Mapping Table

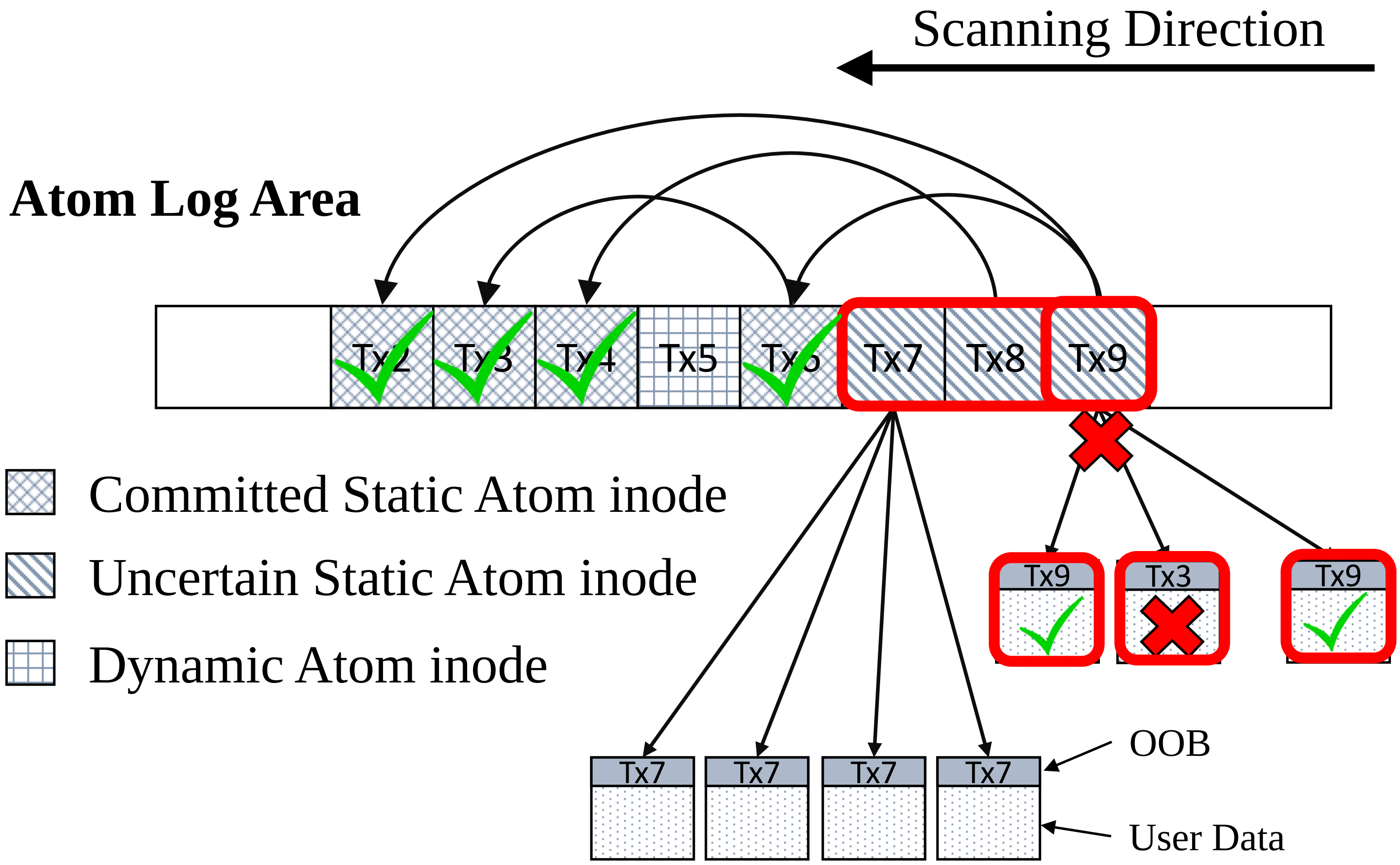
LPN	PPN
34	3012
35	7023
36	1448
⋮	⋮

Page Mapping Table (Delta)

Tx Info	Mapping	Committed
Tx8	(34, 4098) (56, 4099)…	Tx3, Tx6
Tx9	(88, 5122) (34, 5123)…	None
Tx10	(88, 6008) (45, 6009)…	Tx8, Tx9
⋮	⋮	⋮

DAG verification method

For static transactions



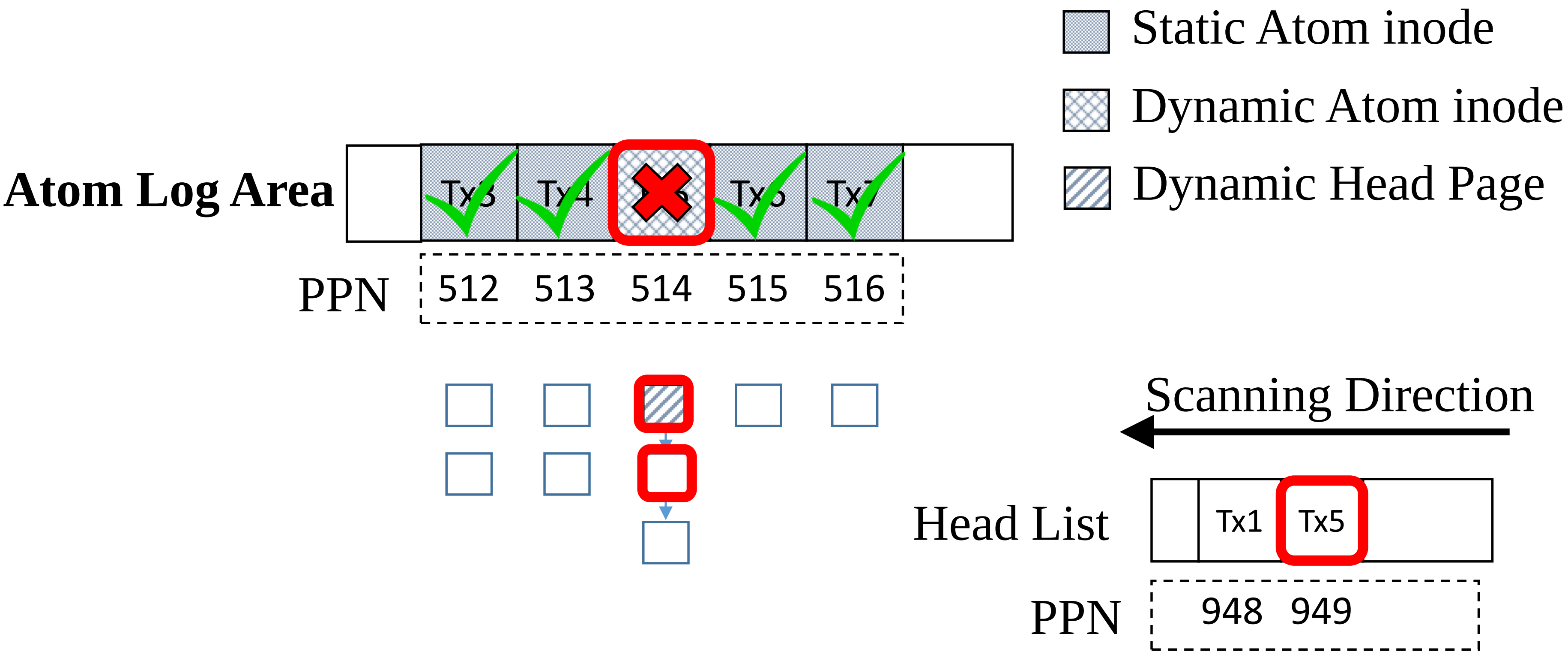
▨ Committed Static Atom inode

▨ Uncertain Static Atom inode

⊞ Dynamic Atom inode

DAG verification method

For dynamic transactions



▦ Static Atom inode

▩ Dynamic Atom inode

▨ Dynamic Head Page

Atom Log Area

Tx3 ✓ Tx4 ✓ Tx5 ✗ Tx6 ✓ Tx7 ✓

PPN 512 513 514 515 516

Scanning Direction

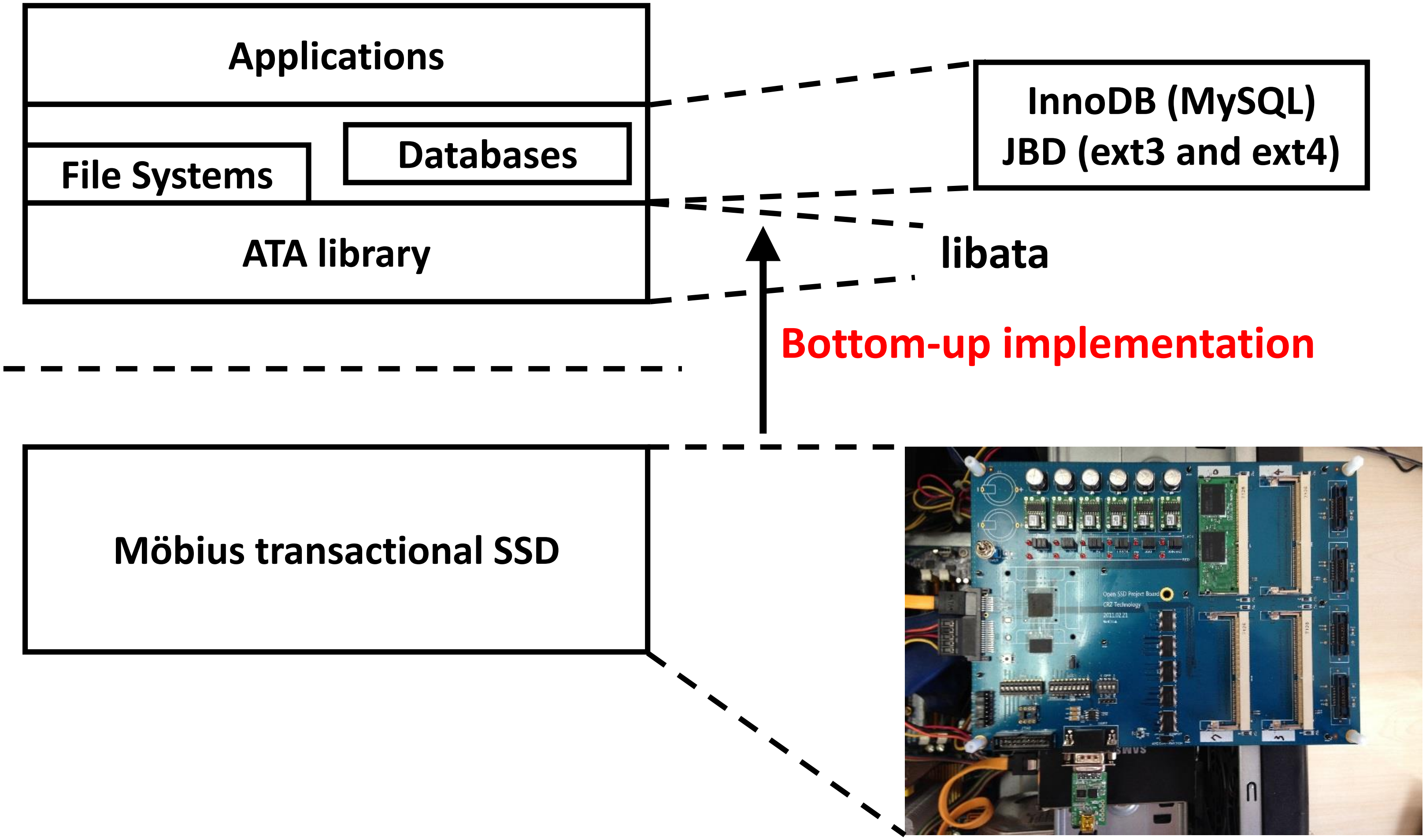
Head List Tx1 Tx5

PPN 948 949

Outline

- Motivations
- Möbius Design
- **Implementation**
- Evaluations
- Conclusions

Implementation



Applications

File Systems

Databases

ATA library

InnoDB (MySQL)
JBD (ext3 and ext4)

libata

Bottom-up implementation

Möbius transactional SSD

SWRITE and DWRITE

- SWRITE
  - Sync and Async modes
- DWRITE
  - Serializable and Read-committed

Garbage collection

For data area

- GC cannot affect recovery or abort procedure, we simply forbid GC to be applied in updating transactions. Since updating transactions are limited, it will not affect the performance

For ALA area

- ALA is a cyclic log structure, and there is no logical address pointing to them, garbage collection procedure in ALA is simple

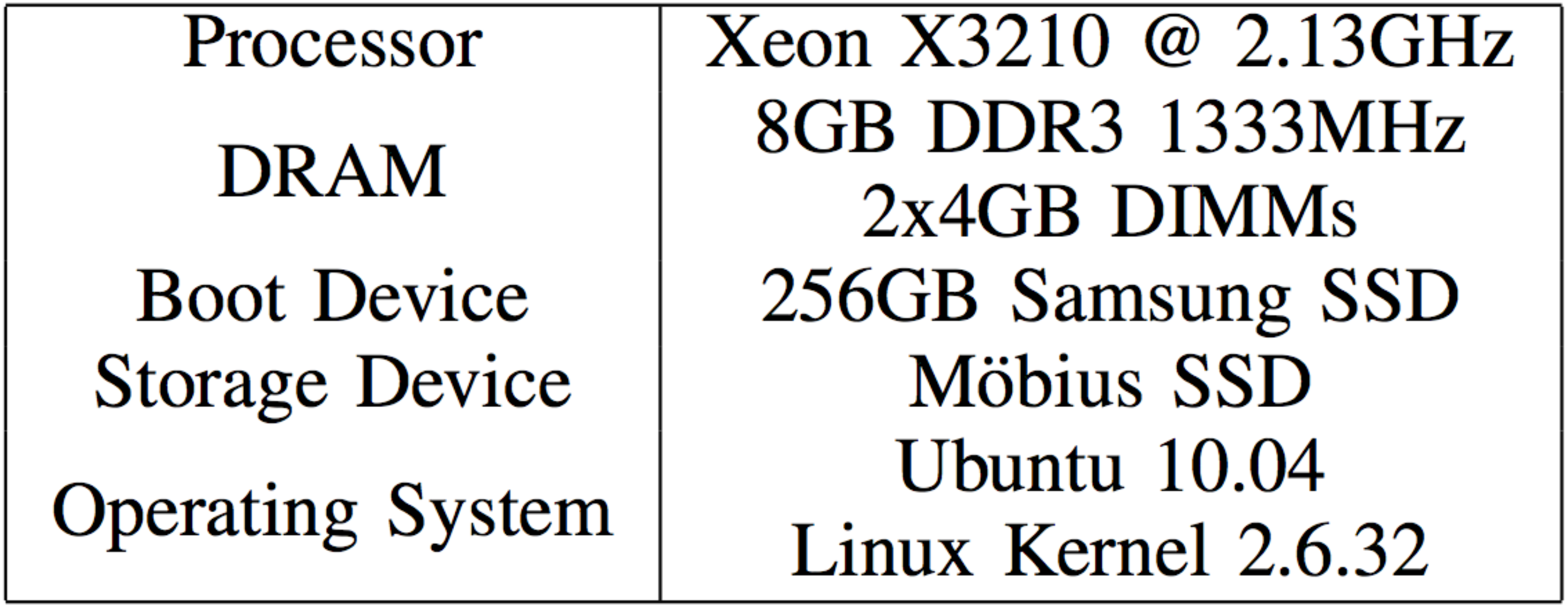
Limitations

- Big transactions
- Small transactions
- “False positive” Async-SWRITE
  - Möbius will return “done” after atom inode is written to flash

Outline

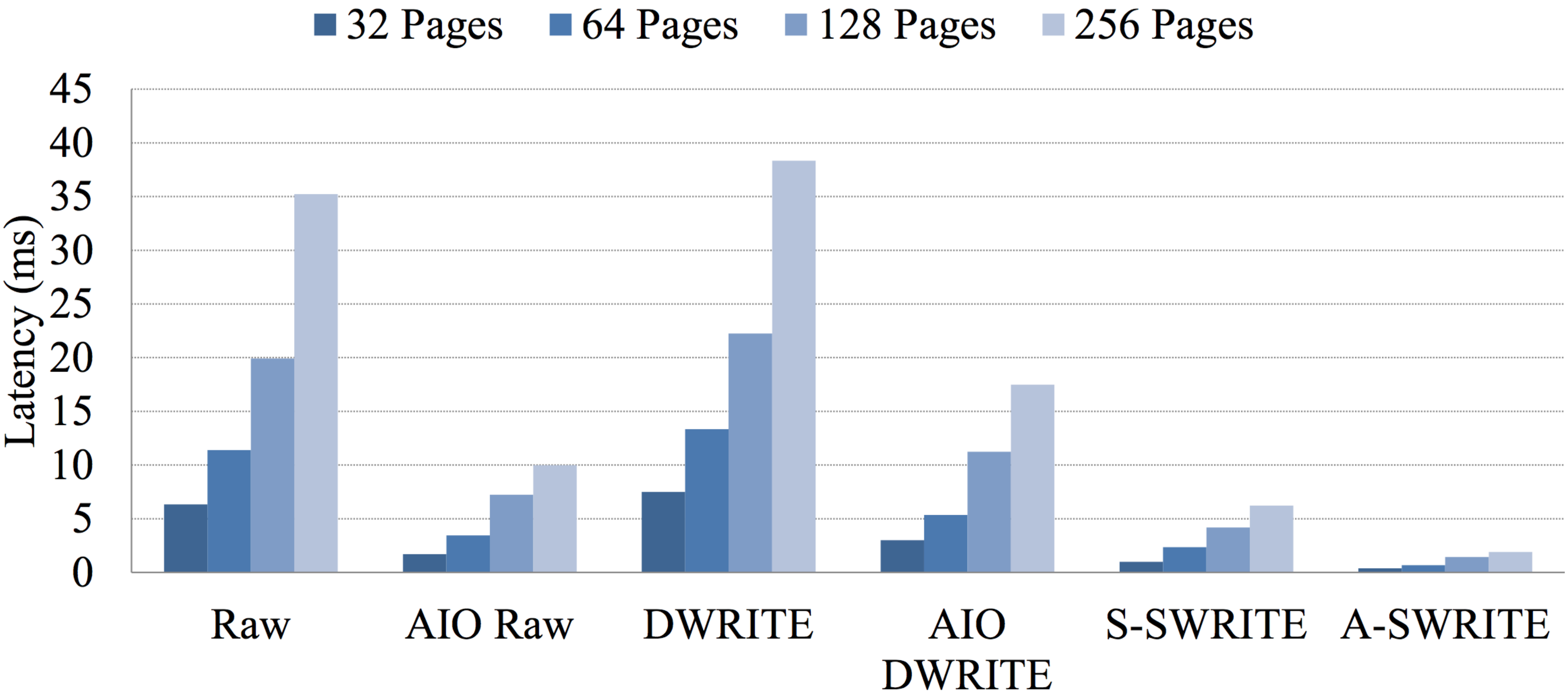
- Motivations
- Möbius Design
- Implementation
- **Evaluations**
- Conclusions

Experimental configurations

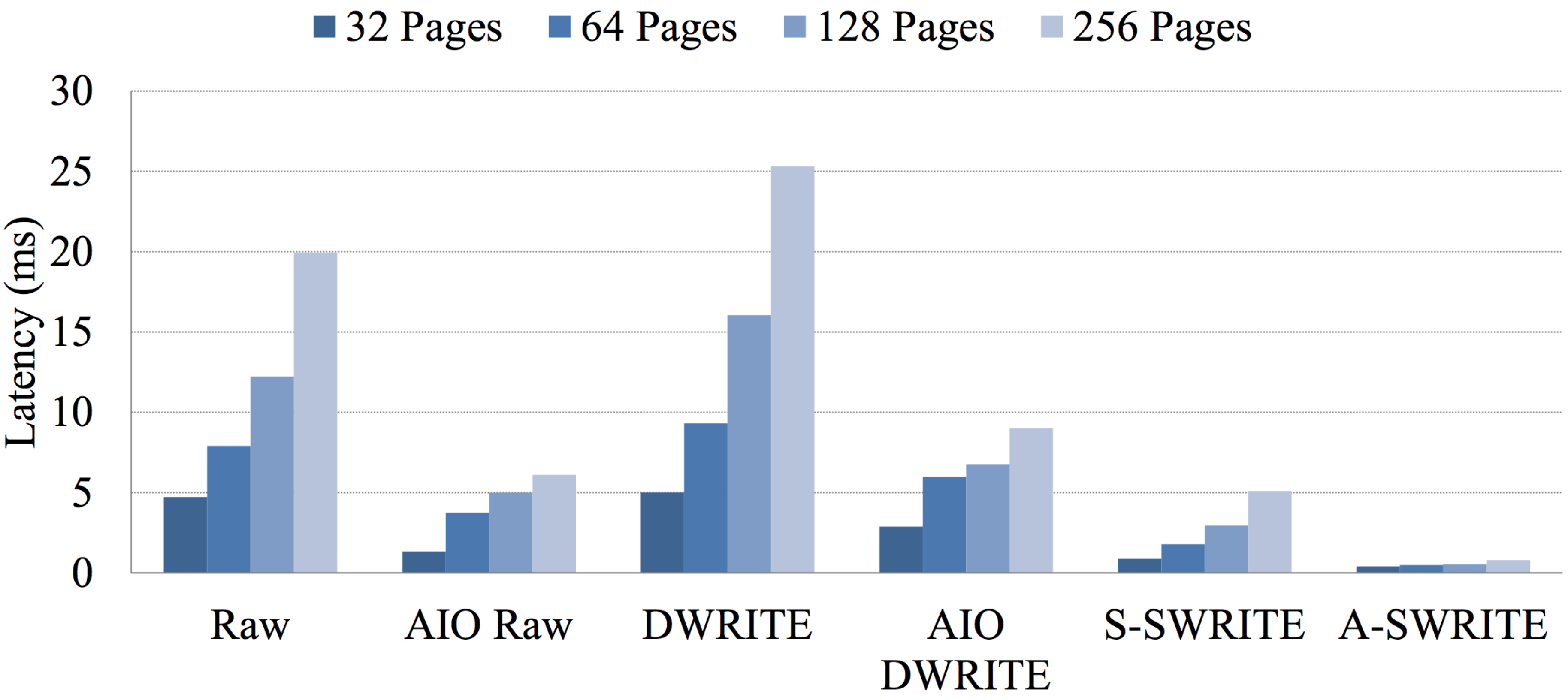


Processor	Xeon X3210 @ 2.13GHz
DRAM	8GB DDR3 1333MHz 2x4GB DIMMs
Boot Device	256GB Samsung SSD
Storage Device	Möbius SSD
Operating System	Ubuntu 10.04 Linux Kernel 2.6.32

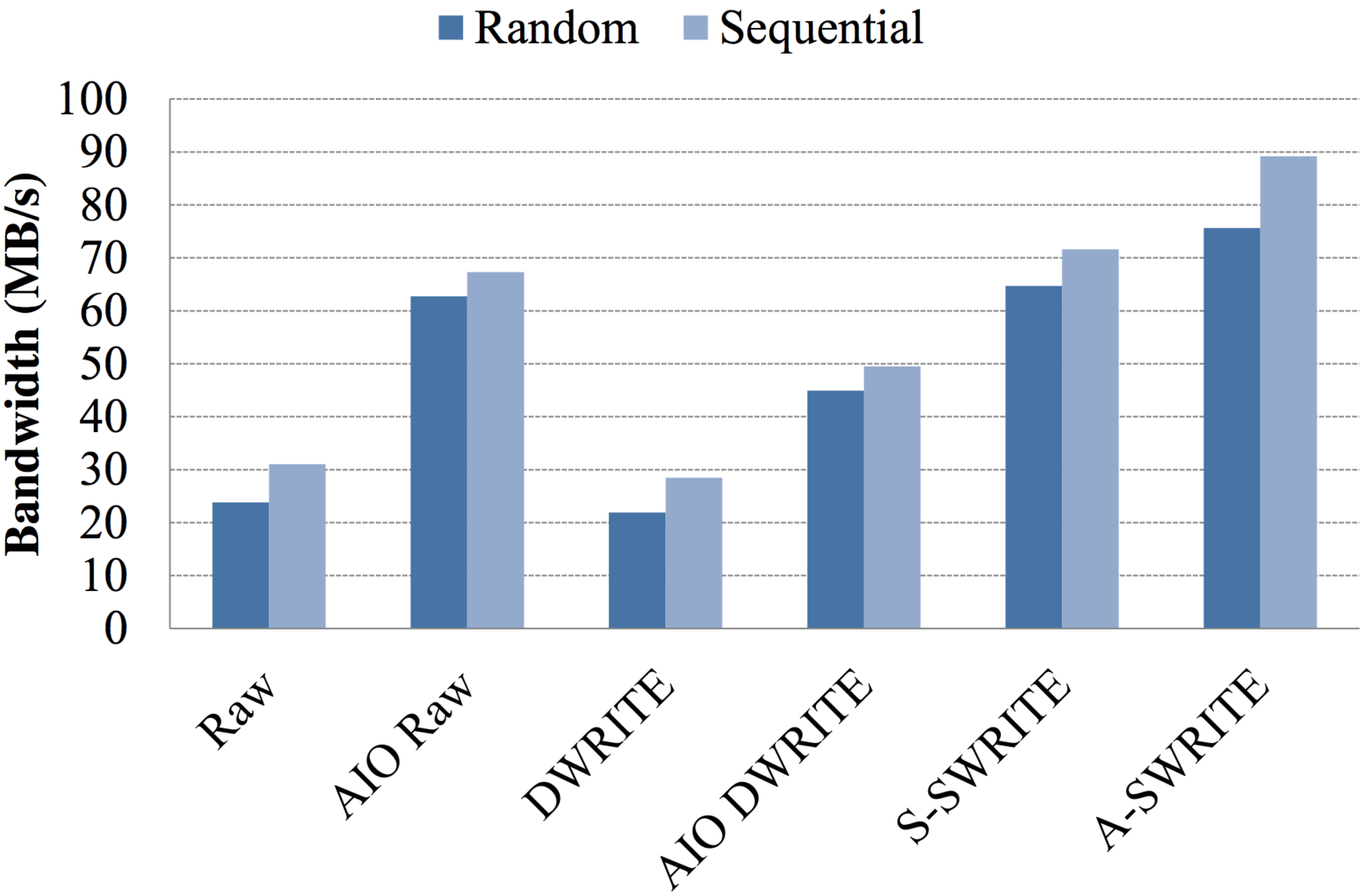
Möbius vs. raw DFTL SSD



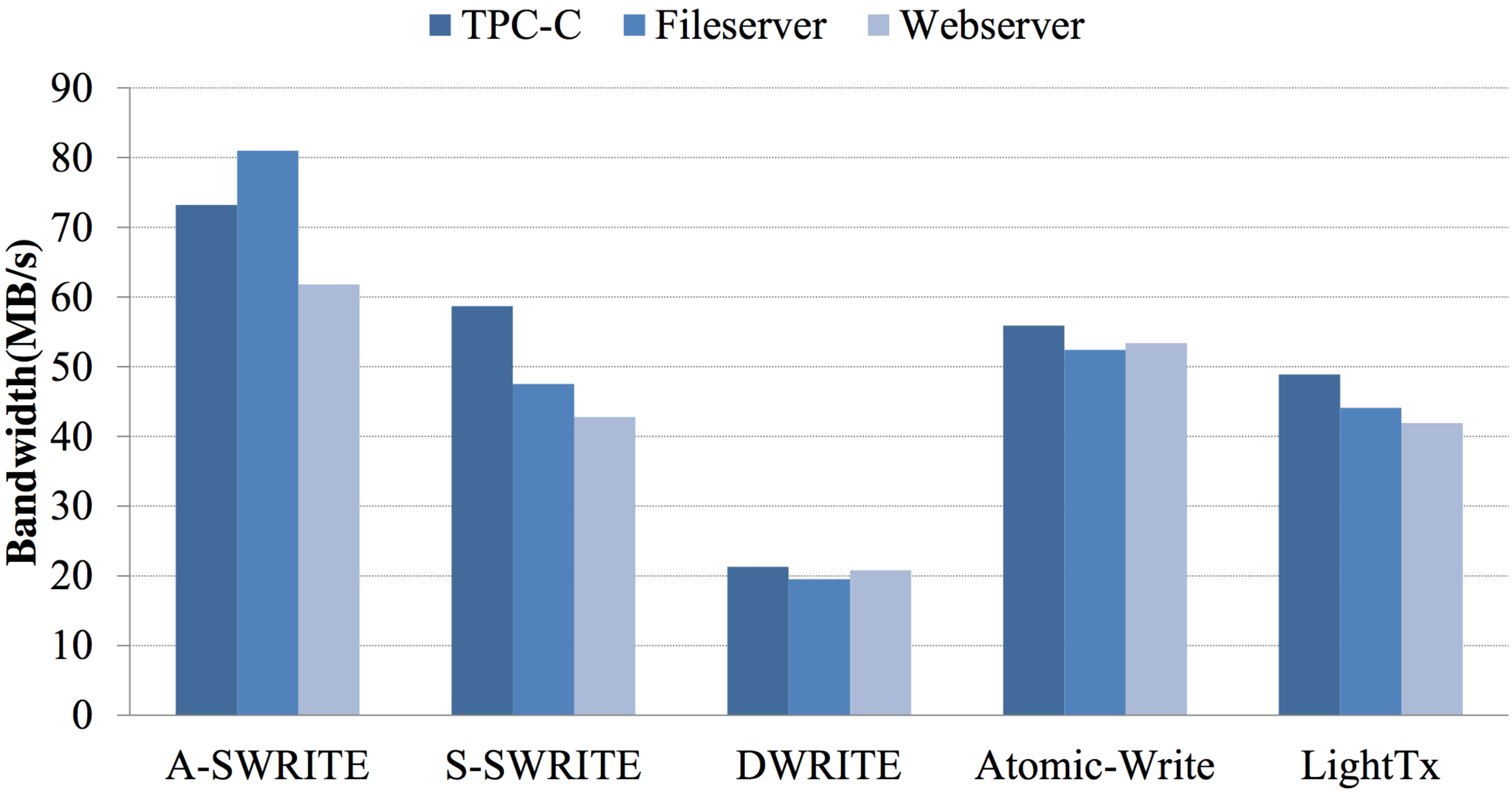
Möbius vs. raw DFTL SSD



Möbius vs. raw DFTL SSD

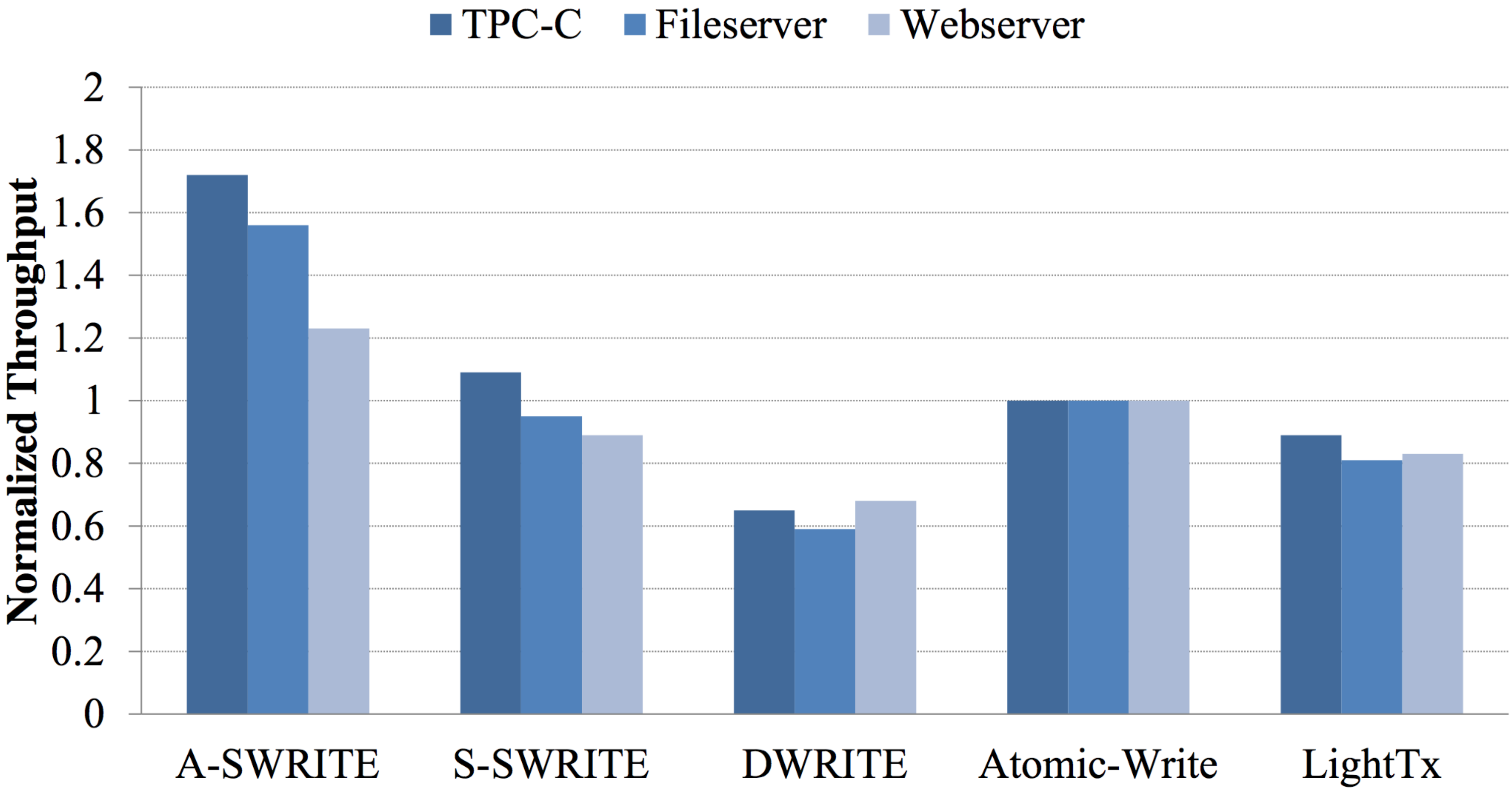


Möbius vs. other transactional SSD designs



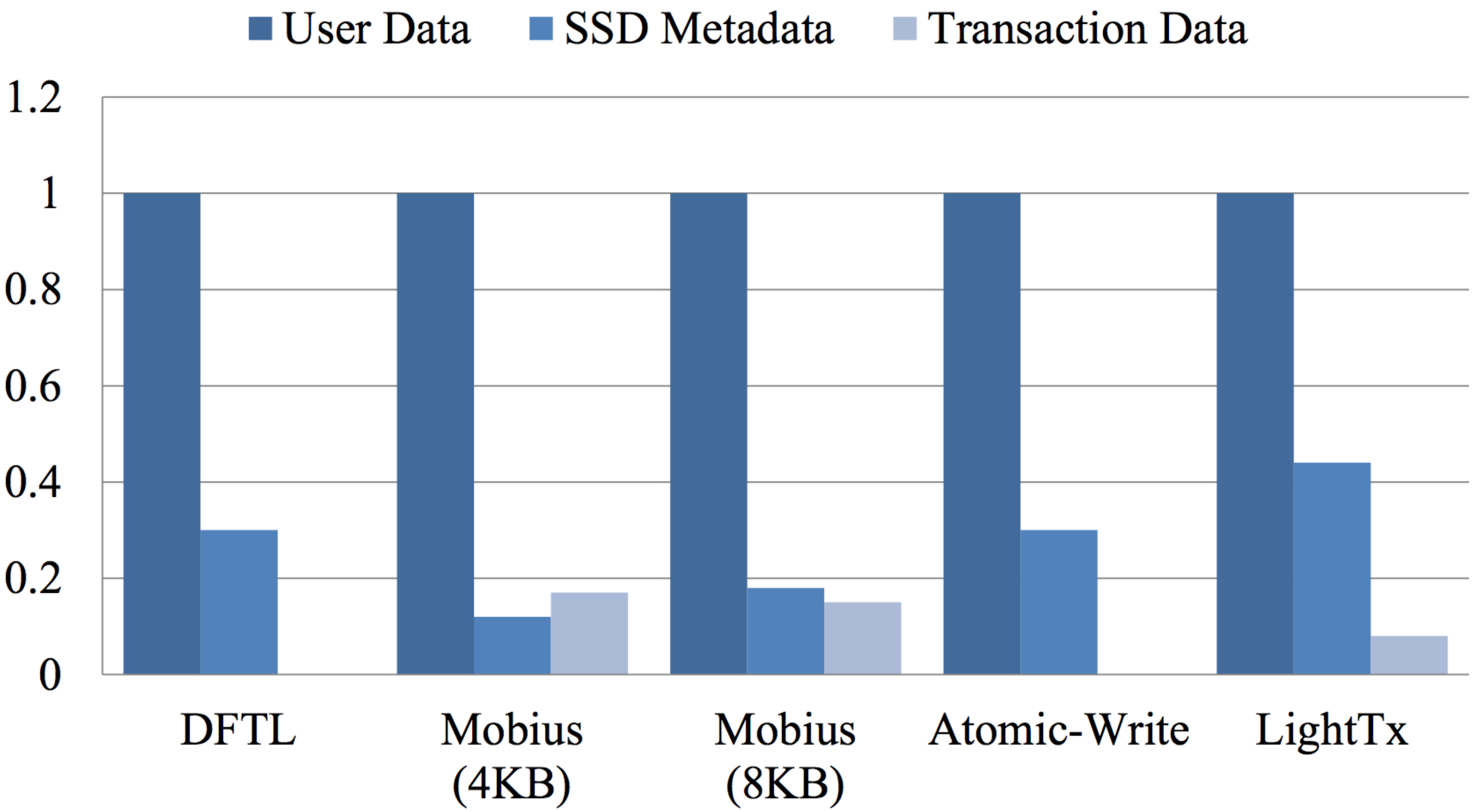
(a) Bandwidth

Möbius vs. other transactional SSD designs

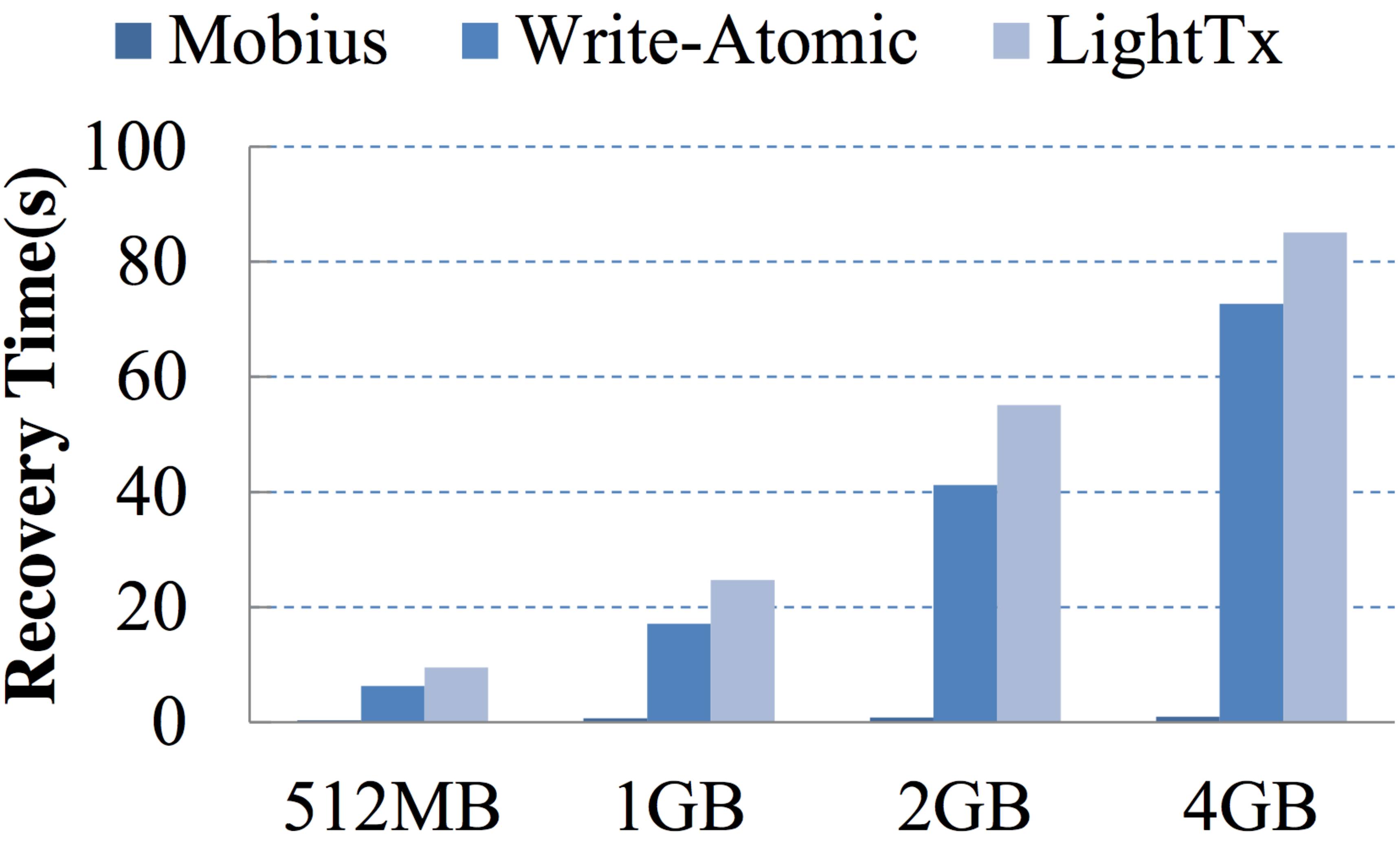


(b) Transaction per Second

Möbius vs. other transactional SSD designs



Möbius vs. other transactional SSD designs



Related Works (1)

- Academia study (Transactional SSD)
  - [***Y. Lu, ICCD'13***] proposed sliding-zone based transactional SSD to support flexible isolation levels (LightTx)
  - [***X. Ouyang, HPCA'11***] proposed a prototype of transactional SSD based on log-based FTL with FusionIO (Write-Atomic)
  - [***V. Prabhakaran, OSDI'08***] proposed a link based transactional flash device (TxFlash)
- Academia study (SSD SPOR)
  - [***T. Chung, J. Syst. Archit.***] proposed a recovery scheme for block level FTL SSD and mainly focus on consistency problem when SSD faces power failure in GC operation (PORCE)
  - [***S. Moon, SEUS'08***] proposed a recovery scheme which works on SSD crash recovery based on a hybrid FTL named FAST (CR-FAST)

Related Works (2)

- Academia study (Database Optimization for SSD)
  - [***J. Do, SIGMOD'13***] explored the opportunities and challenges associated with exploiting this functionality of Smart SSDs for relational analytic query processing
  - [***P. Wang, EuroSys'14***] investigated internal flash channels to applications to work with the LSM-tree-based KV store, specifically LevelDB
- Academia study (File system consistency)
  - [***V. Chidambaram, FAST'12***] addressed NoFS, a lightweight file system that employs a backpointer-based consistency to provide crash consistency without ordering write
  - [***A. Ma, FAST'13***] presented a modified ext3 file system, rext3, to directly support the fast file system checker, ffsck

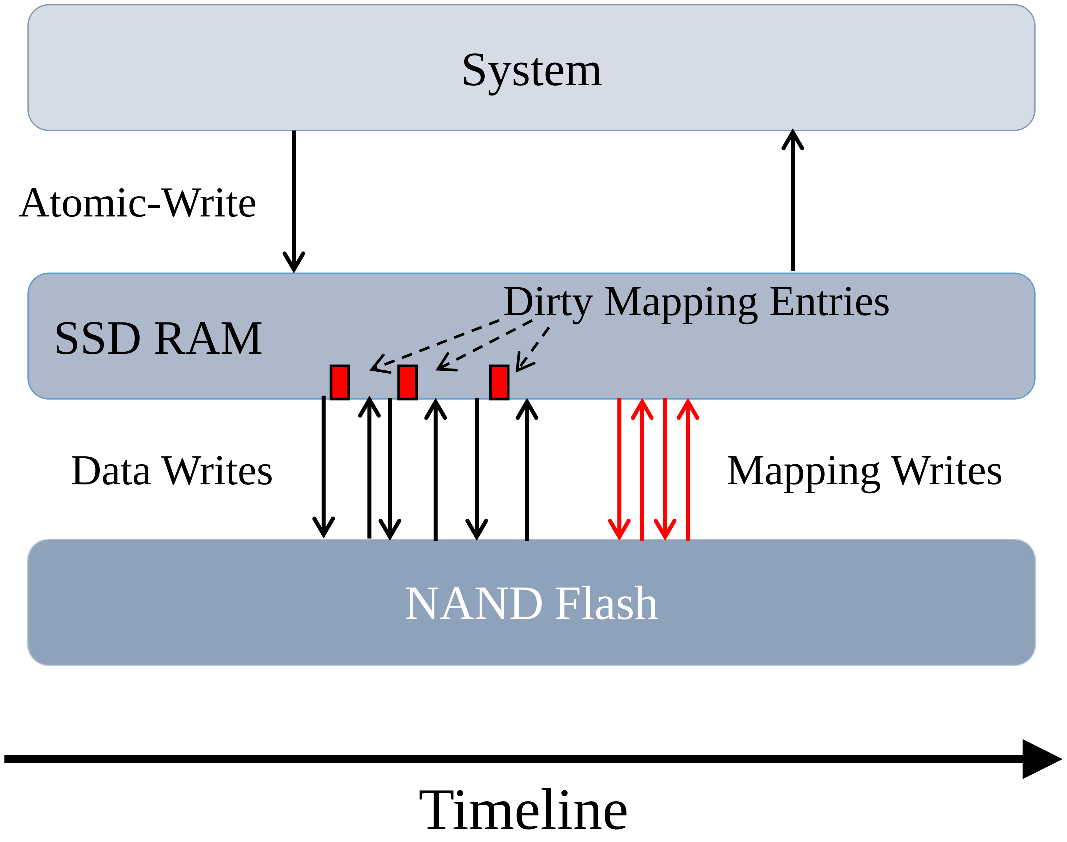
Conclusions

- **Challenge**: Software transaction processing schemes might not be suitable for a out-of-place update NAND SSD
- **Our goal**: propose a new high performance transactional SSD architecture with rich primitives
- **Observation**
  - **Serialized transaction processing**: caused by ordered transaction recovery
  - **Long recovery time**: caused by scanning unfinished transactions
  - **Extra Sudden Power-Off Recovery (SPOR) logic**: lived in SSD FTL
- **Key Ideas**
  - **Atom file**: to abstract transaction into a “file”
  - **DAG commit protocol**: by skipping unnecessary scanning
  - **Recovery logic combination**: by combining SPOR with transaction aborting
- **Möbius: a new transactional SSD architecture**
  - **Rich primitives**: support both static and dynamic transactions
  - Avoid unnecessary scanning by **DAG verification method**
  - **Recover FTL and transaction processing logic after power failures**
- **Results:** Möbius expect to save 4~29 times of recovery time and offer a 67% higher throughput than other transactional SSD designs

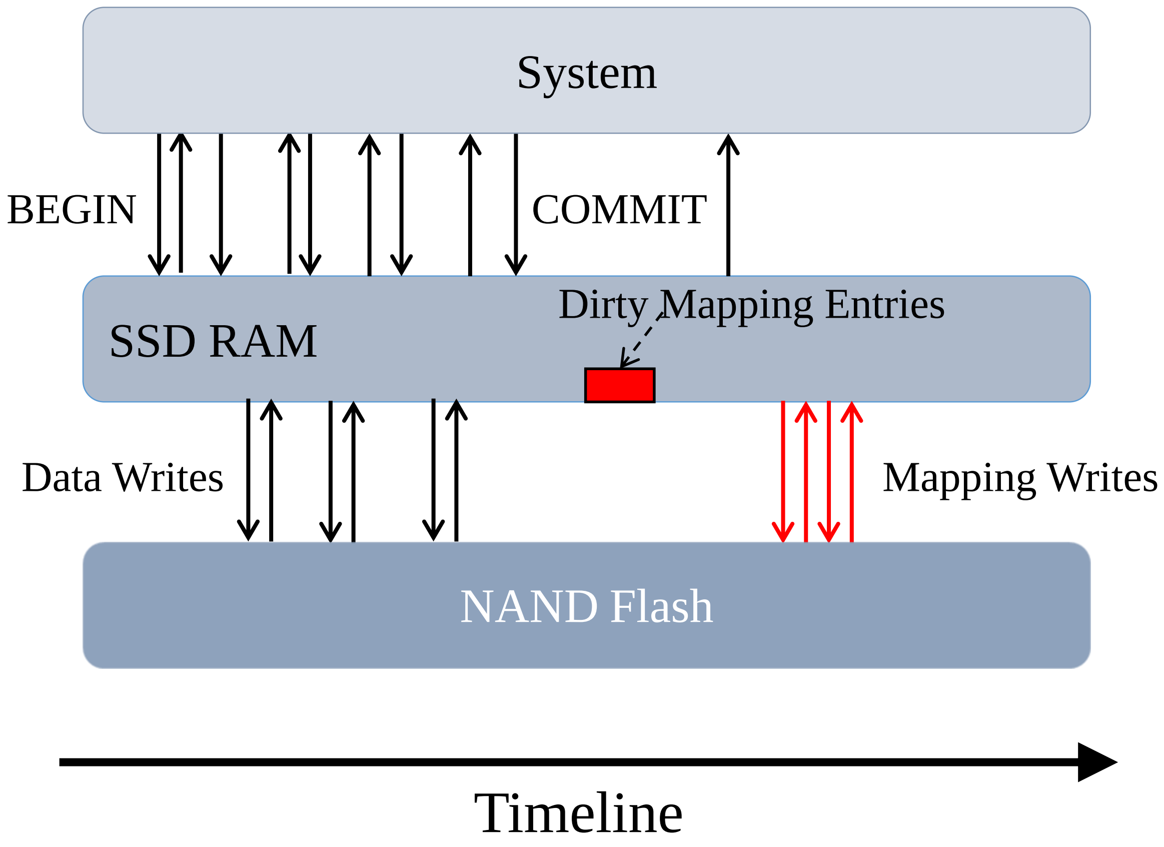
Questions?

Backup

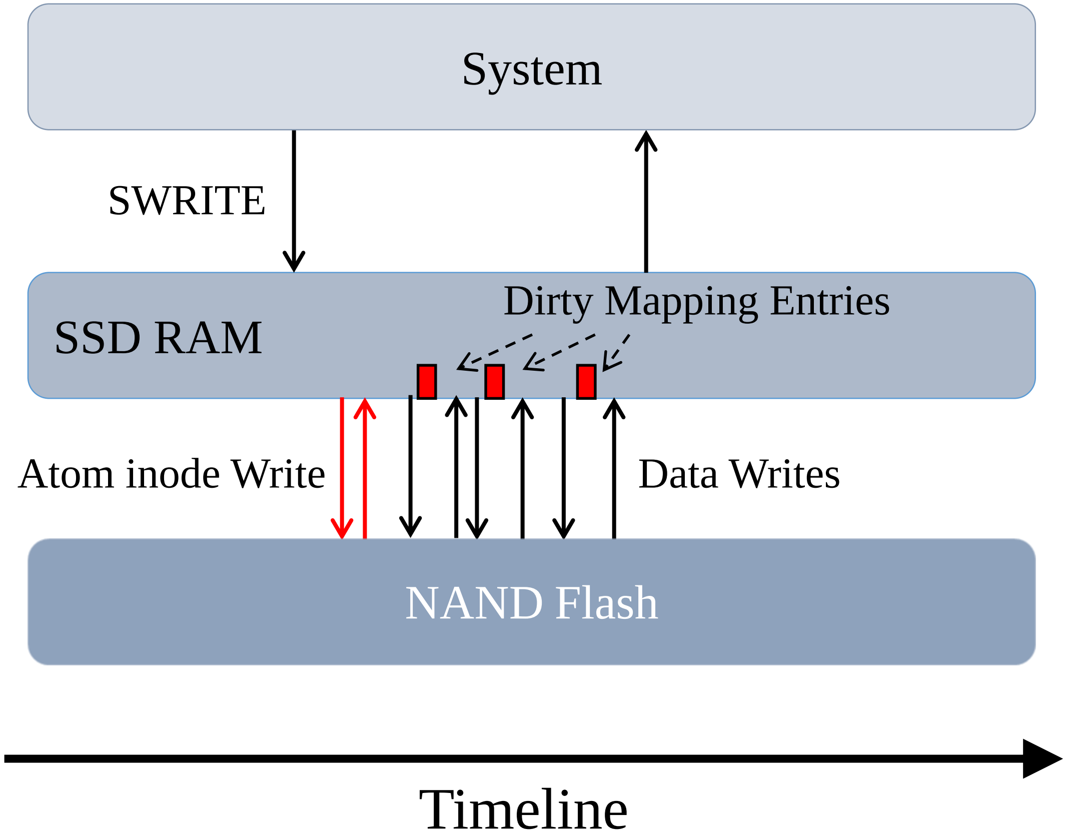
Persistence order in Atomic-Write



Persistence order in LightTx



Persistence order in Möbius


System
SWRITE
SSD RAM
Dirty Mapping Entries
Atom inode Write
Data Writes
NAND Flash
Timeline