Lab 8: Kernel-based FTL

Application-Managed Flash(FAST'16)

Sungjin Lee, Ming Liu, Sangwoo Jun, and Shuotao Xu, MIT CSAIL;

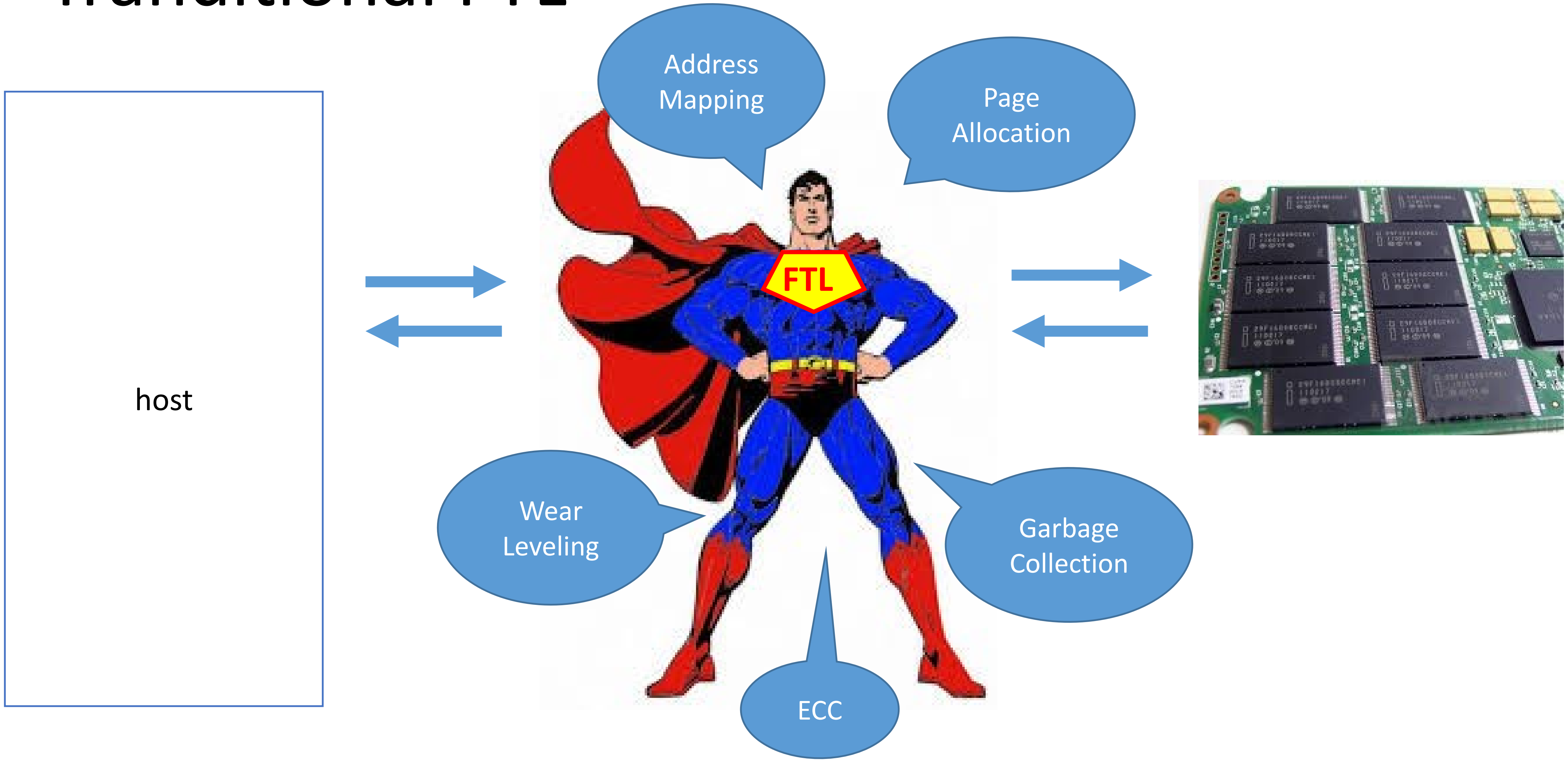
Jihong Kim, Seoul National University; Arvind, MIT CSAIL

Joohyung Park(joohyungpark@csl.skku.edu)

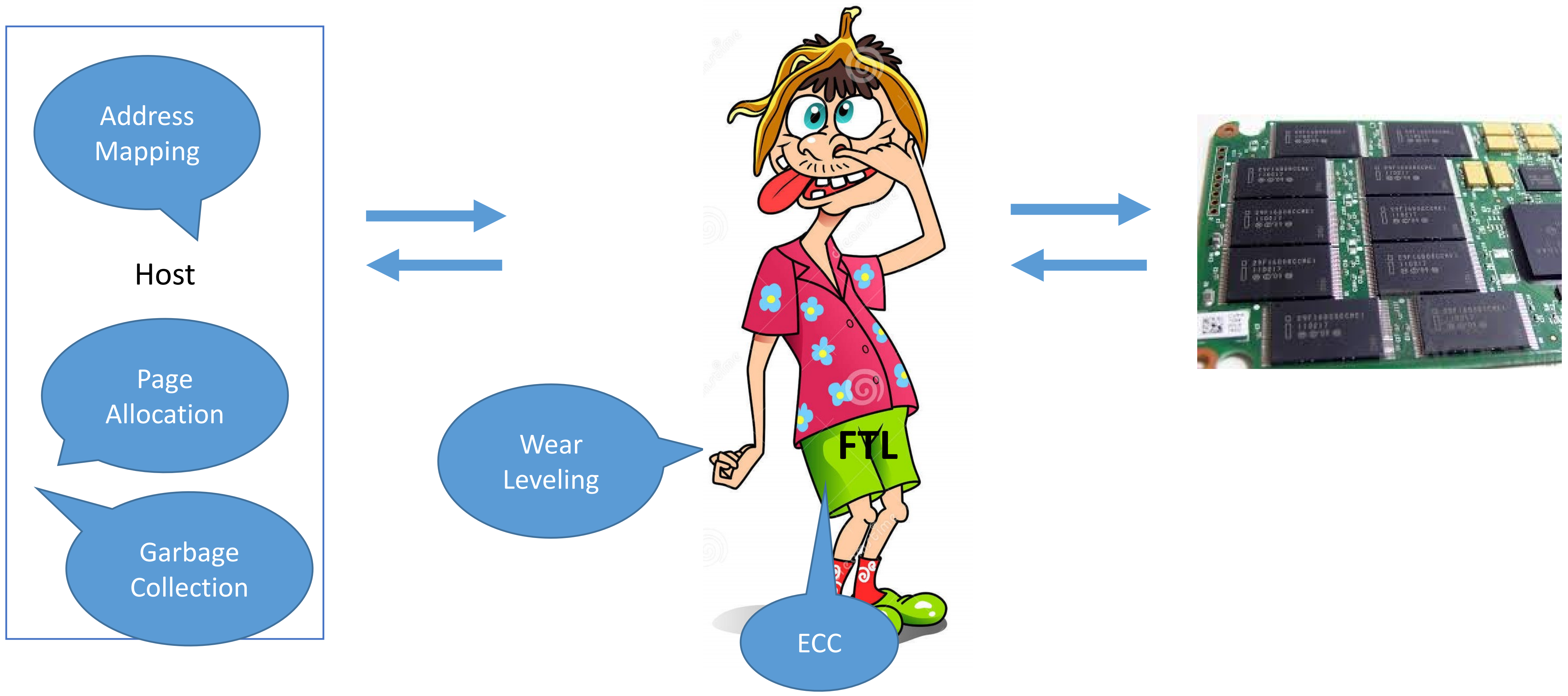
Computer Systems Laboratory

Sungkyunkwan University

Tranditional FTL

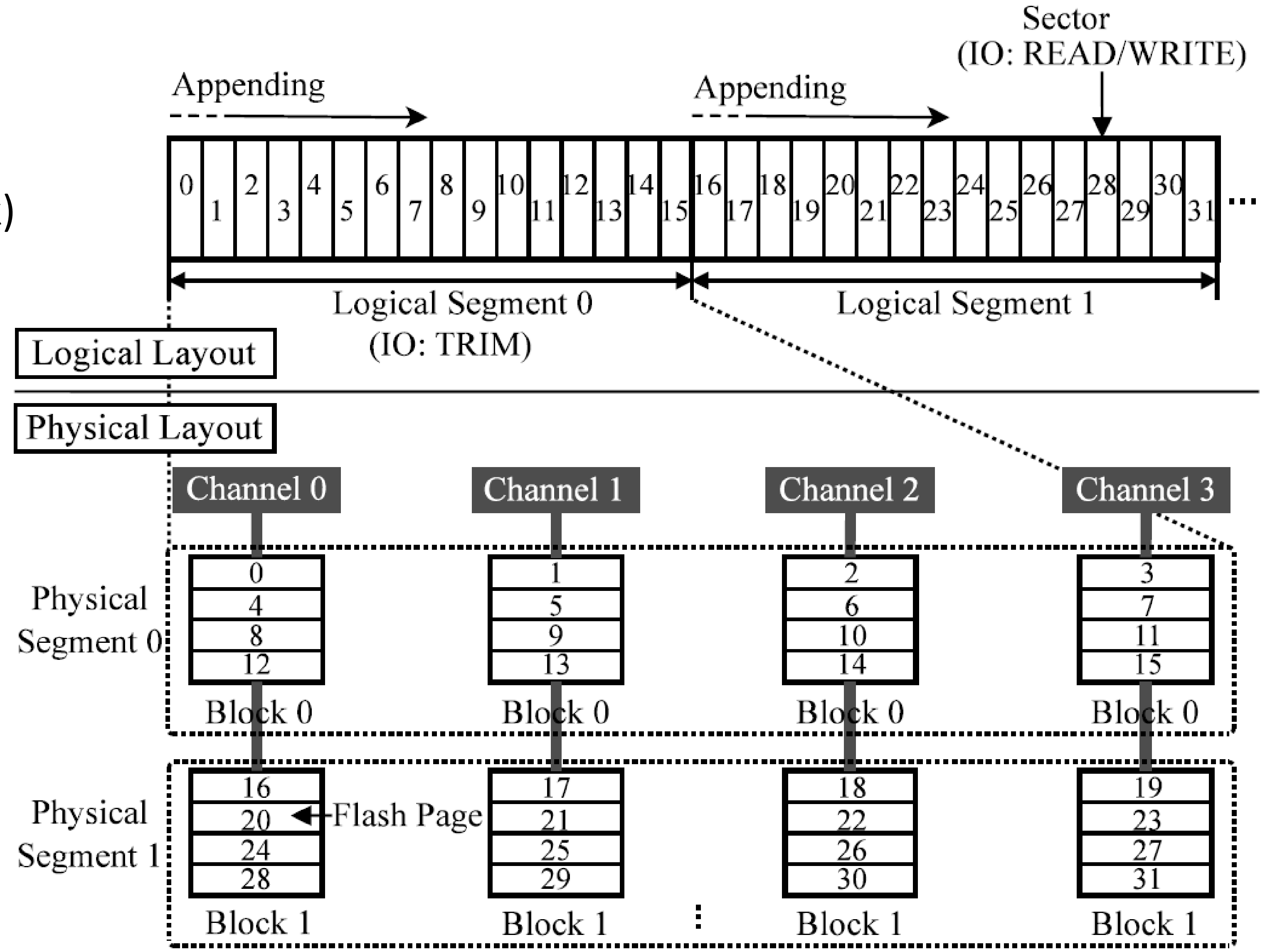


This Work: Buck-Passing FTL



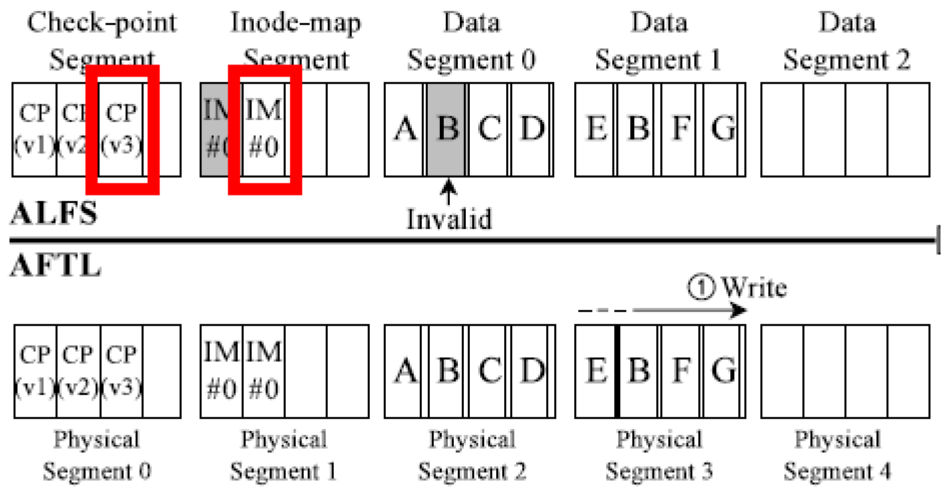
Page Allocation via Static Mapping

- A Segment
  - a group of blocks from each channel in a same way(bank)
- Fixed Size
  - # chs * # pages/blk * 8K
- No implicit invalidation in the unit of segment via trim
- Error returns for overwrite requests

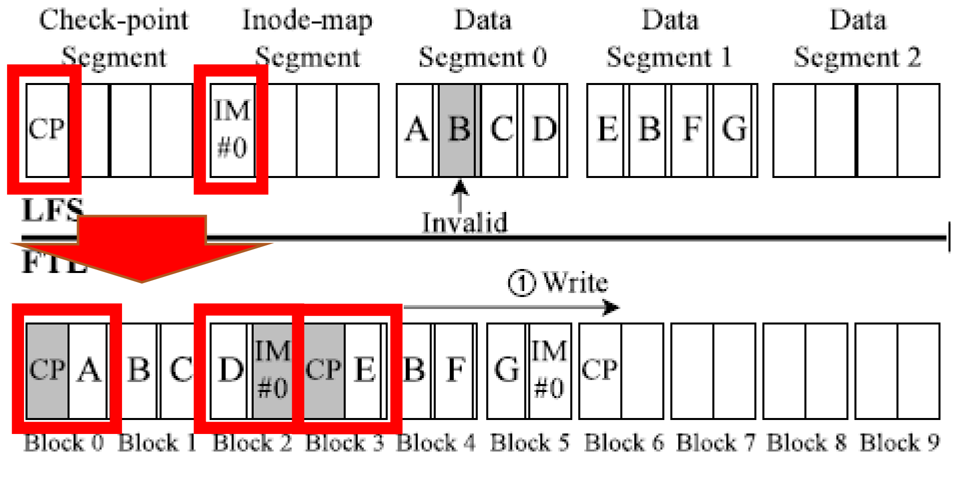


File Modification

No invalidation for block-level append only system

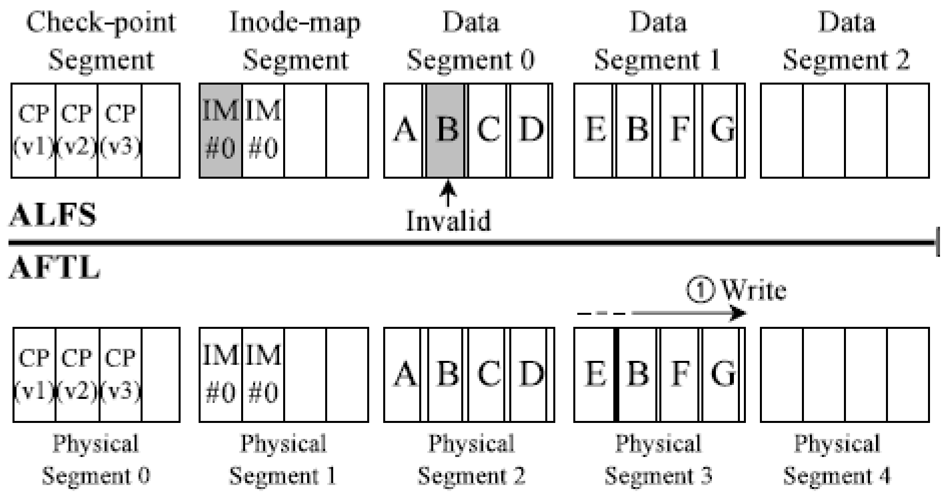


In-place updates on little metadata generating victims

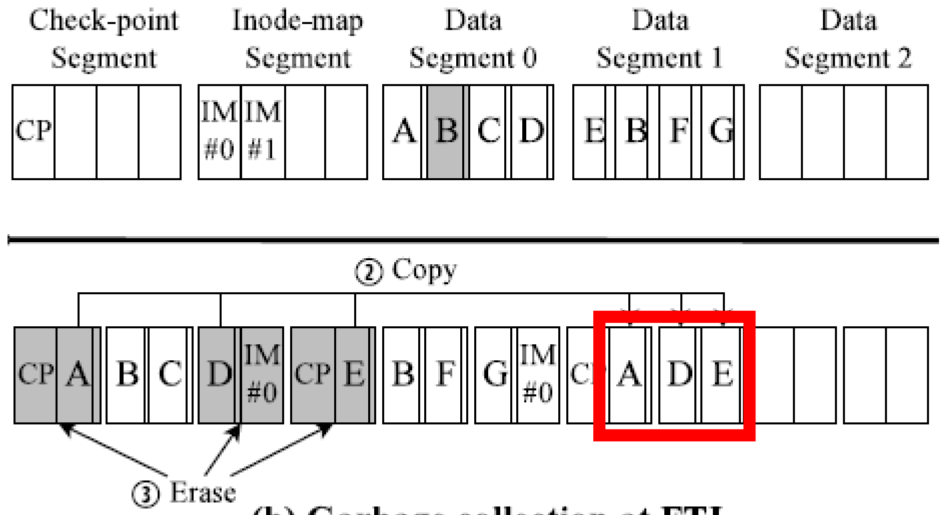


(a) Initial State

Device-level GC



No implicit device-level GC

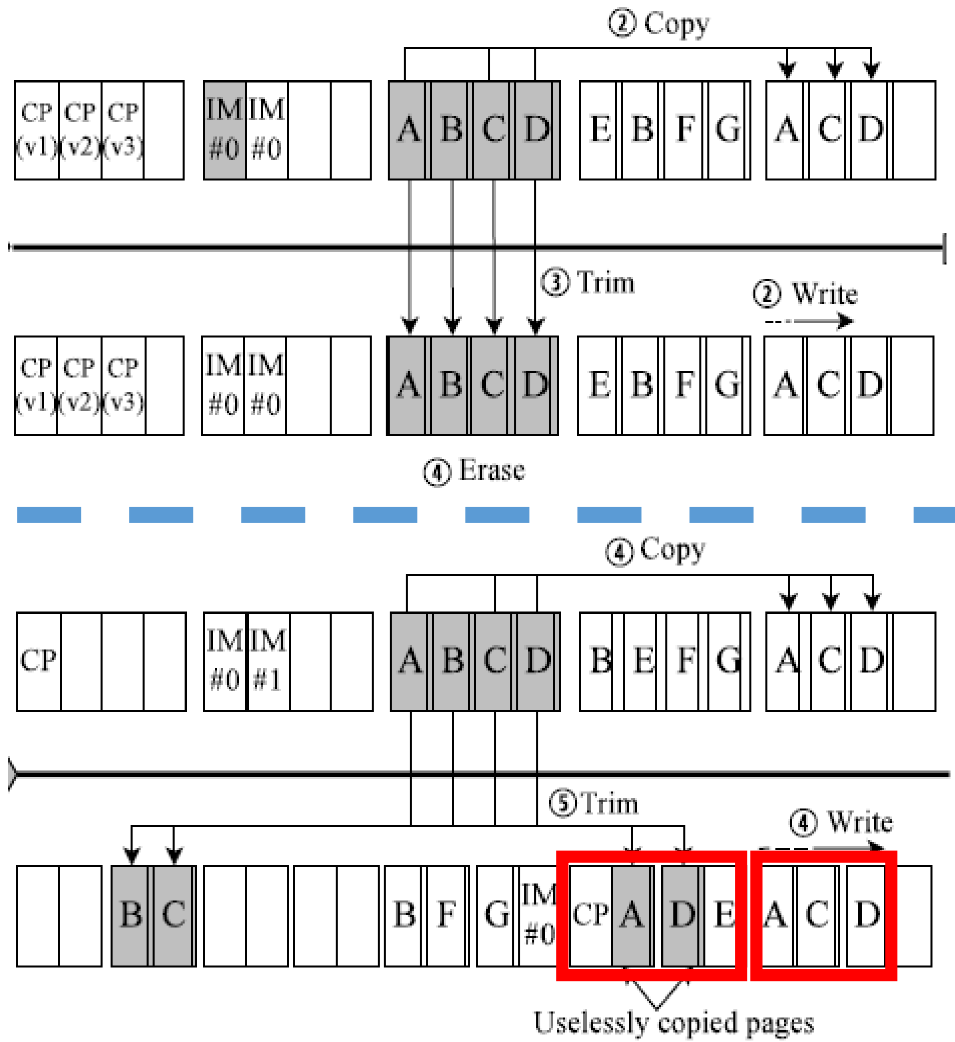


(b) Garbage collection at FTL

Device-level valid pages are copied

Are they valid at the host-side also?

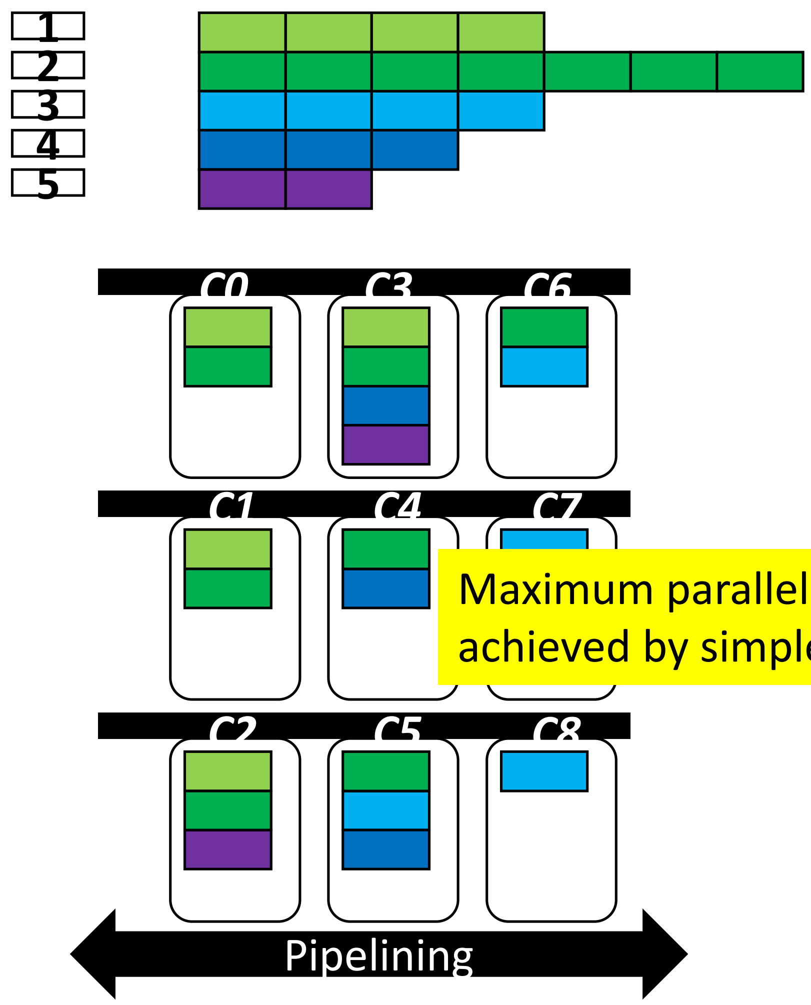
Host-level GC



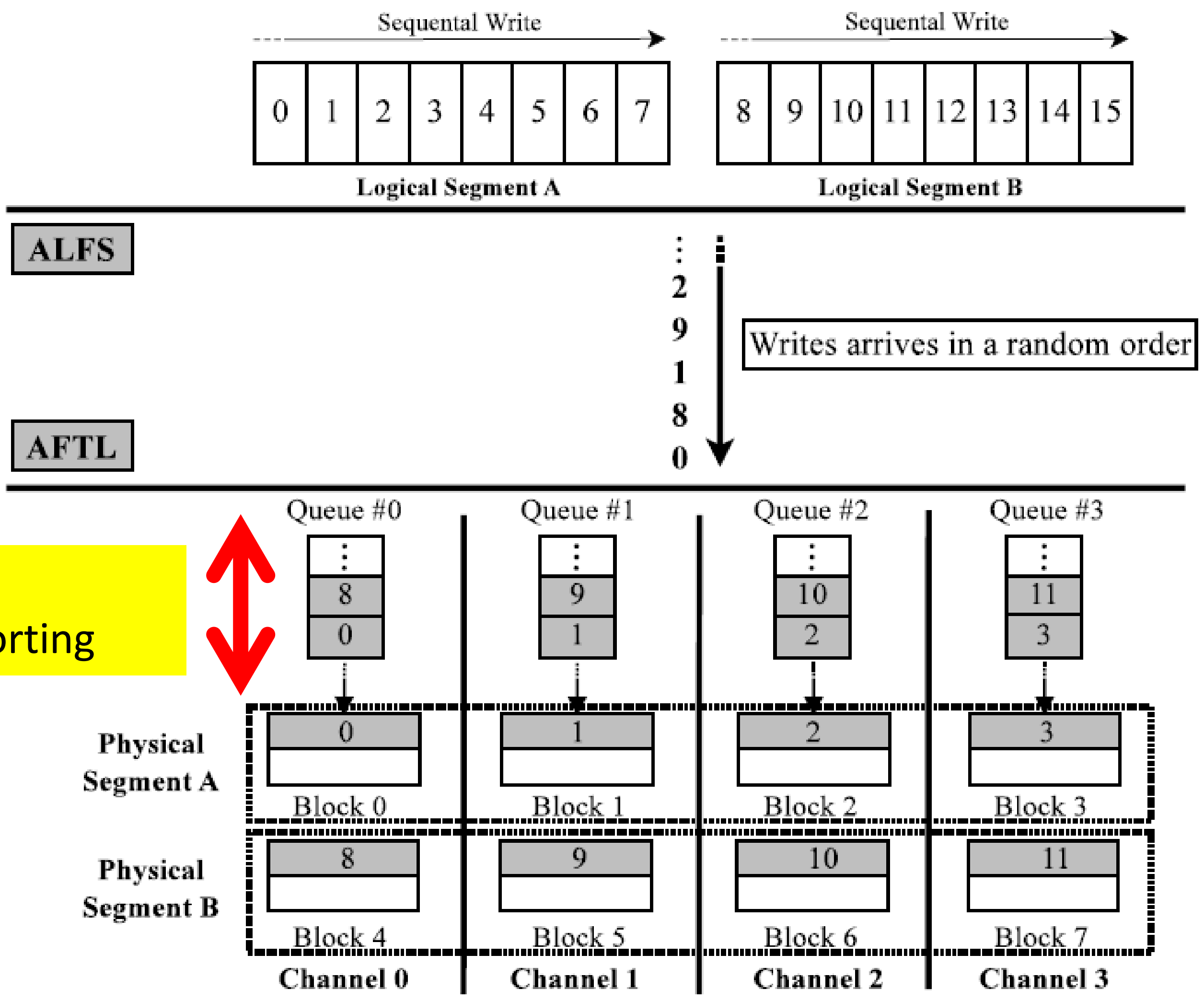
(c) Garbage collection at LFS

Re-invalidate / re-written by host with generating other victims to be merged

Simple Device-Level I/O Scheduler



Maximum parallelism achieved by simple sorting



Advantages for Static Mapping

- No fine-grained mapping and GC
  - Reduce HW overhead(mapping table, computing resources)
  - Guarantee predictable performance from user
- Easy to exploit system level parallelism

Disadvantages for Static Mapping

- The size of allocation unit is
  - Large and fixed
  - Hard to exploit flash level parallelism in the worst case

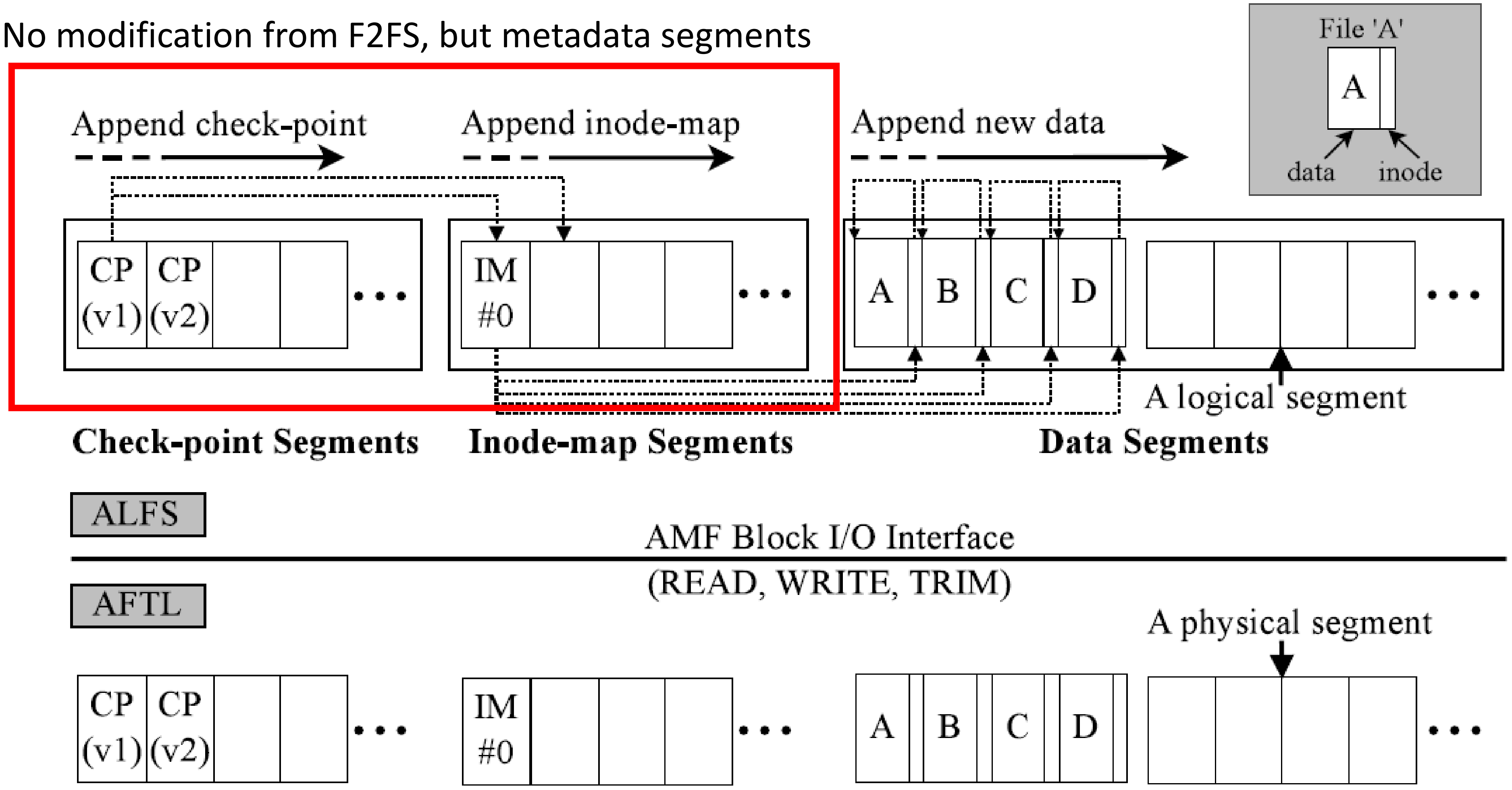
- Most of user platforms are forced to fix their codes
  - Even platforms issuing I/O in log structured manner, there are many overwrites on the metadata to manage their system

Compatibility of AMF

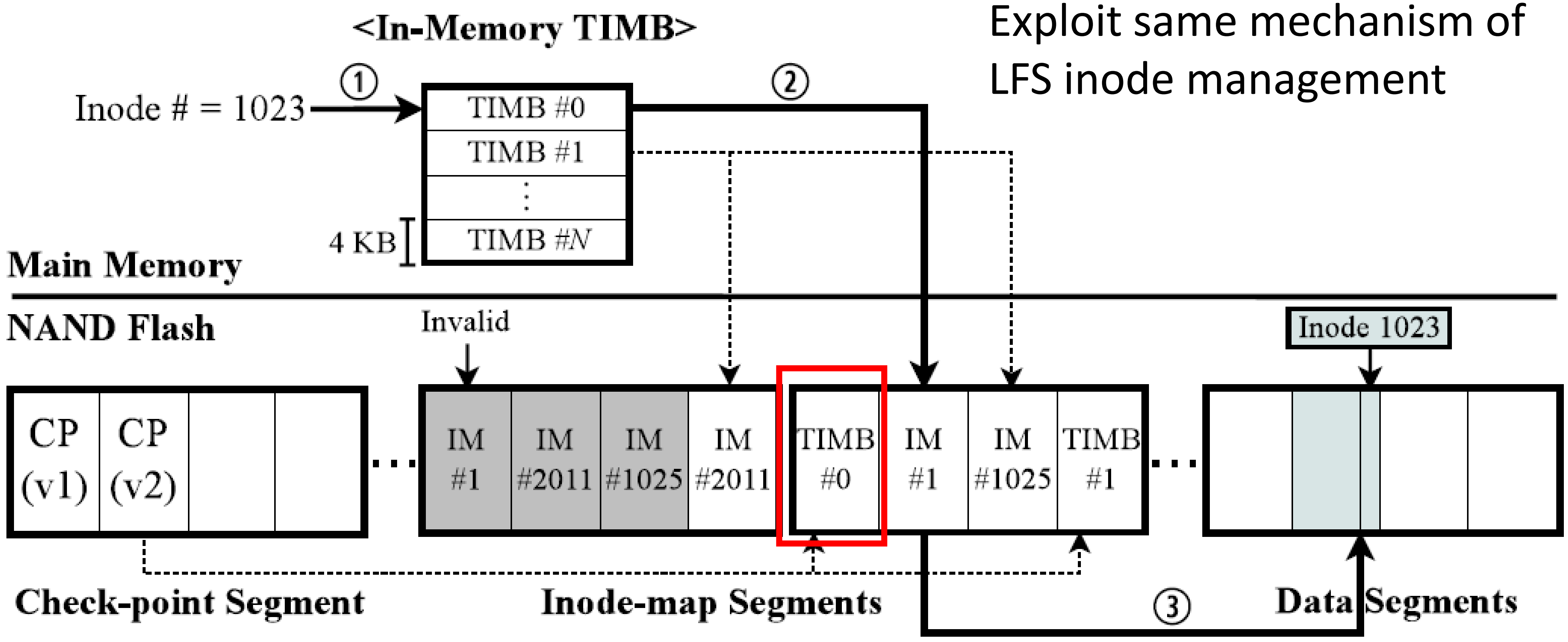
- Same set of I/O interface
- Newly define block I/O interfaces
  - Non-rewritable sectors
  - Linear array of sectors to form a segment
  - Unit of TRIM
- Advantage of AMF comparing with SDF is compatibility
  - Only prerequisite process is modification on User platform to eliminate in-place-updates
- No consideration for MLC/TLC power failure at all

Modified F2FS: ALFS

No modification from F2FS, but metadata segments



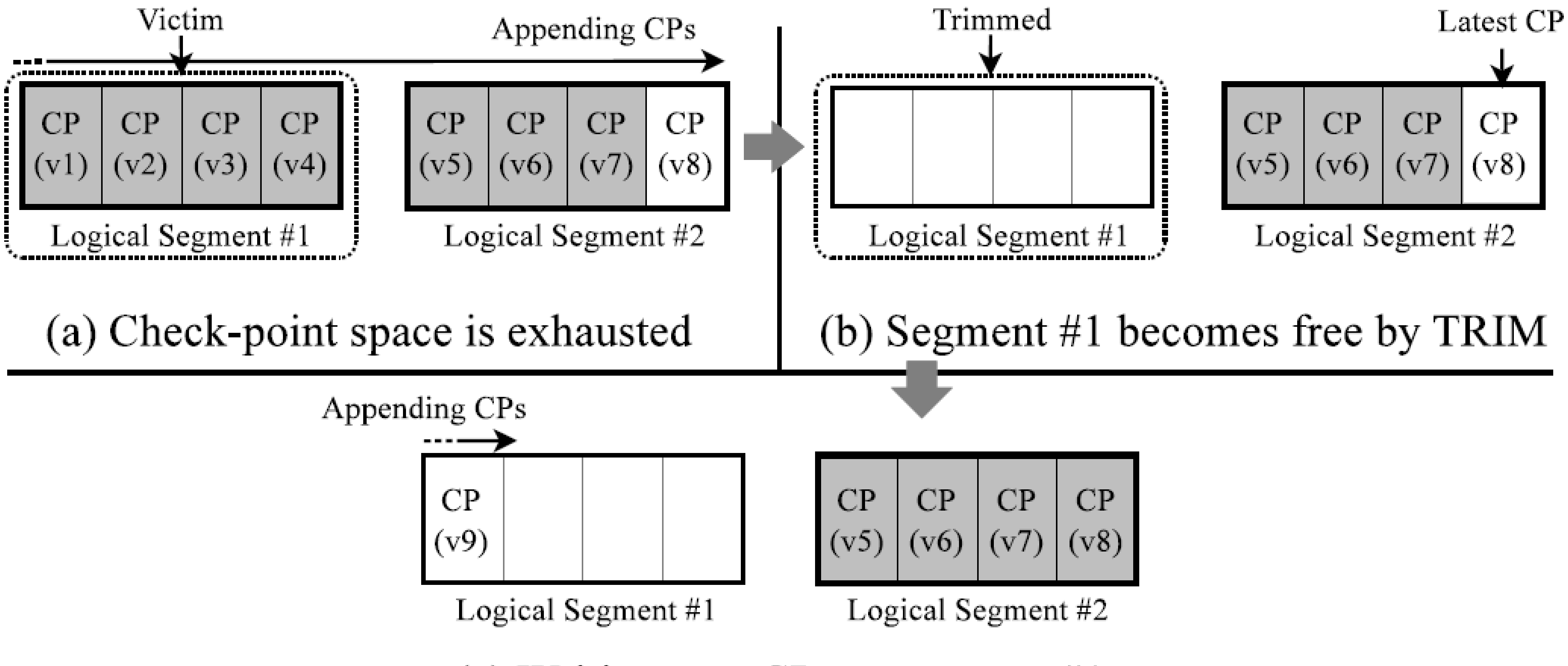
Inode-Map Segment Management



Exploit same mechanism of LFS inode management

Flush with modification of TIMB block

Check-point Segment Management



(a) Check-point space is exhausted

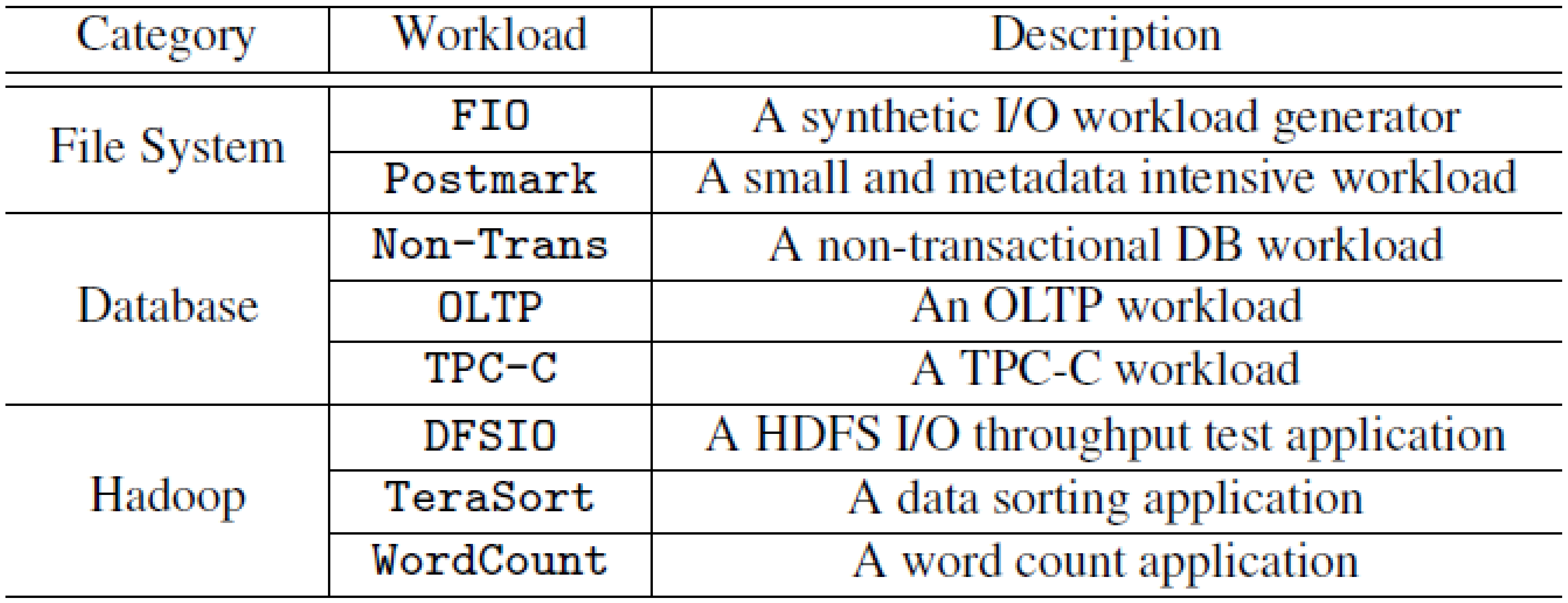
(b) Segment #1 becomes free by TRIM

(c) Writing new CPs to segment #1

Evaluation Environment

- CPU
  - Xeon 24 cores, 1.6GHz
- DRAM
  - Physically 24GB, but set to 1.5GB not to load whole mapping table
- SSD
  - 8ch X 4wy, 512GB NAND flash
  - 1 block = 128 * 4K pages
  - Raw performance:
    - RR(240K IOPS) RW(67K IOPS) SR(930MB/s) SW(260MB/s)

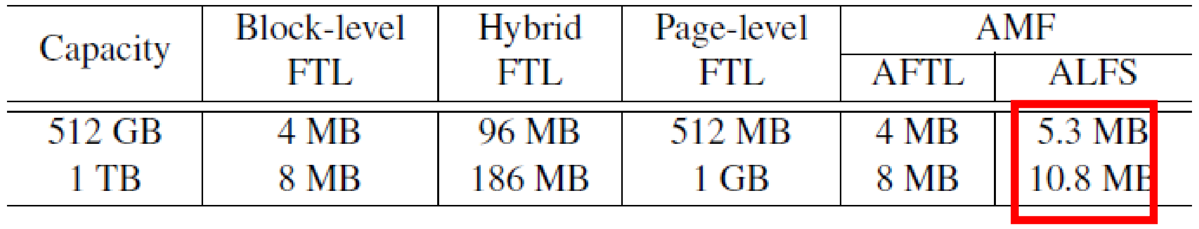
Benchmark Workloads



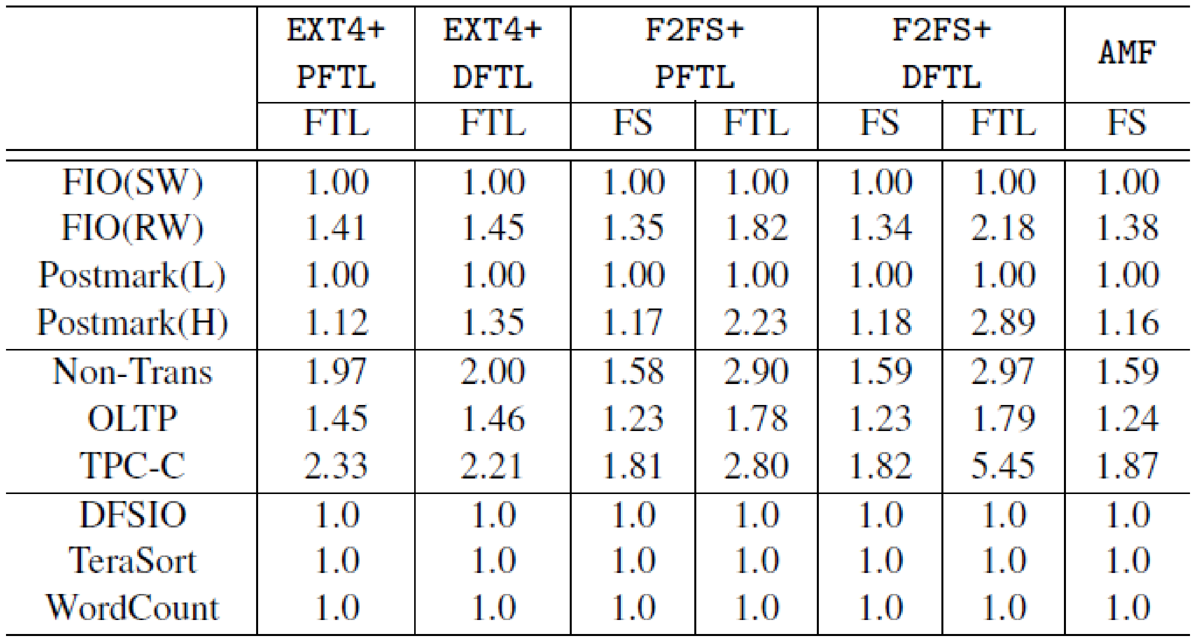
Category	Workload	Description
File System	`FIO`	A synthetic I/O workload generator
	`Postmark`	A small and metadata intensive workload
Database	`Non-Trans`	A non-transactional DB workload
	`OLTP`	An OLTP workload
	`TPC-C`	A TPC-C workload
Hadoop	`DFSIO`	A HDFS I/O throughput test application
	`TeraSort`	A data sorting application
	`WordCount`	A word count application

Memory Overhead and WAF

- Low memory overhead to manage mapping table, but additional overhead for TIMBs
- EXT4 vs F2FS
  - Duplication of log-structured management
- PFTL vs DFTL
  - I/Os of mapping table

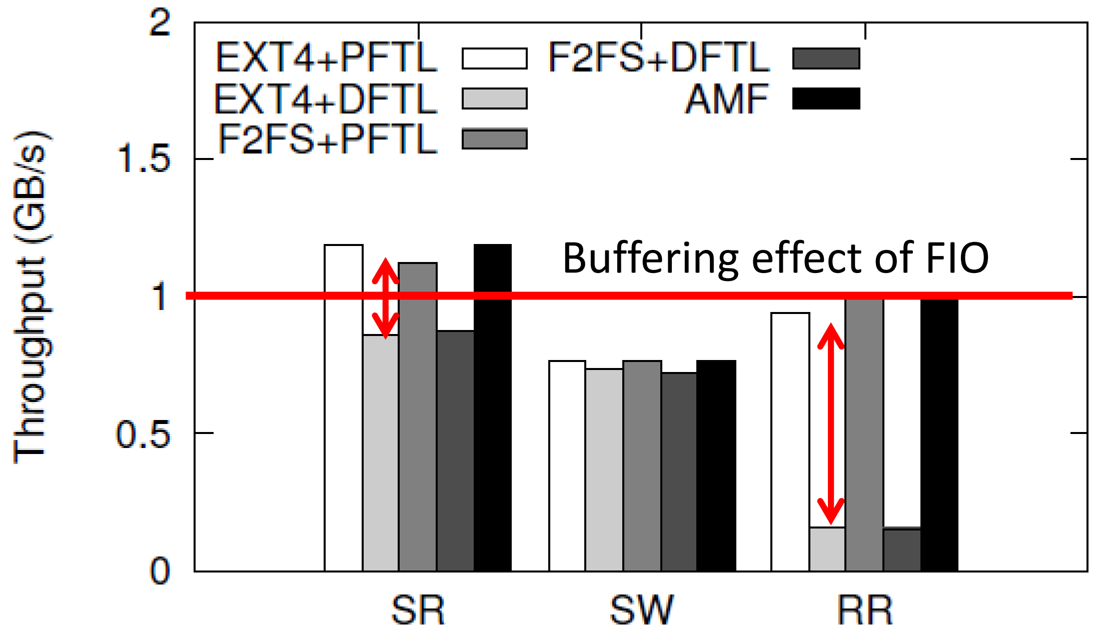


Capacity	Block-level FTL	Hybrid FTL	Page-level FTL	AMF	
				AFTL	ALFS
512 GB	4 MB	96 MB	512 MB	4 MB	5.3 MB
1 TB	8 MB	186 MB	1 GB	8 MB	10.8 MB



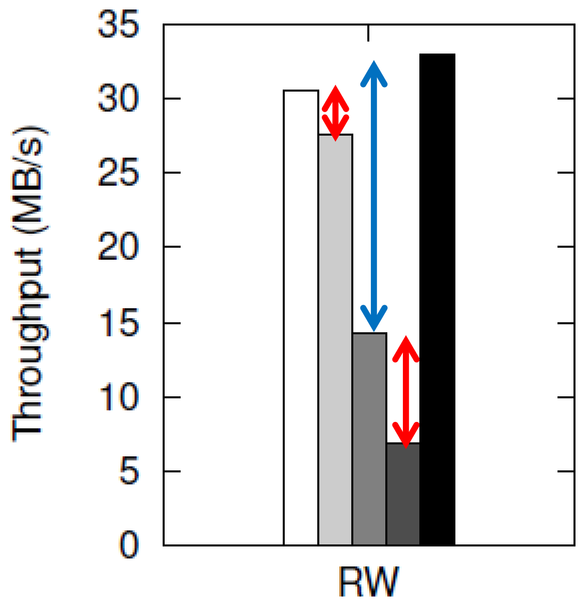
	EXT4+ PFTL	EXT4+ DFTL	F2FS+ PFTL		F2FS+ DFTL		AMF
	FTL	FTL	FS	FTL	FS	FTL	FS
FIO(SW)	1.00	1.00	1.00	1.00	1.00	1.00	1.00
FIO(RW)	1.41	1.45	1.35	1.82	1.34	2.18	1.38
Postmark(L)	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Postmark(H)	1.12	1.35	1.17	2.23	1.18	2.89	1.16
Non-Trans	1.97	2.00	1.58	2.90	1.59	2.97	1.59
OLTP	1.45	1.46	1.23	1.78	1.23	1.79	1.24
TPC-C	2.33	2.21	1.81	2.80	1.82	5.45	1.87
DFSIO	1.0	1.0	1.0	1.0	1.0	1.0	1.0
TeraSort	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WordCount	1.0	1.0	1.0	1.0	1.0	1.0	1.0

FIO Benchmark Results



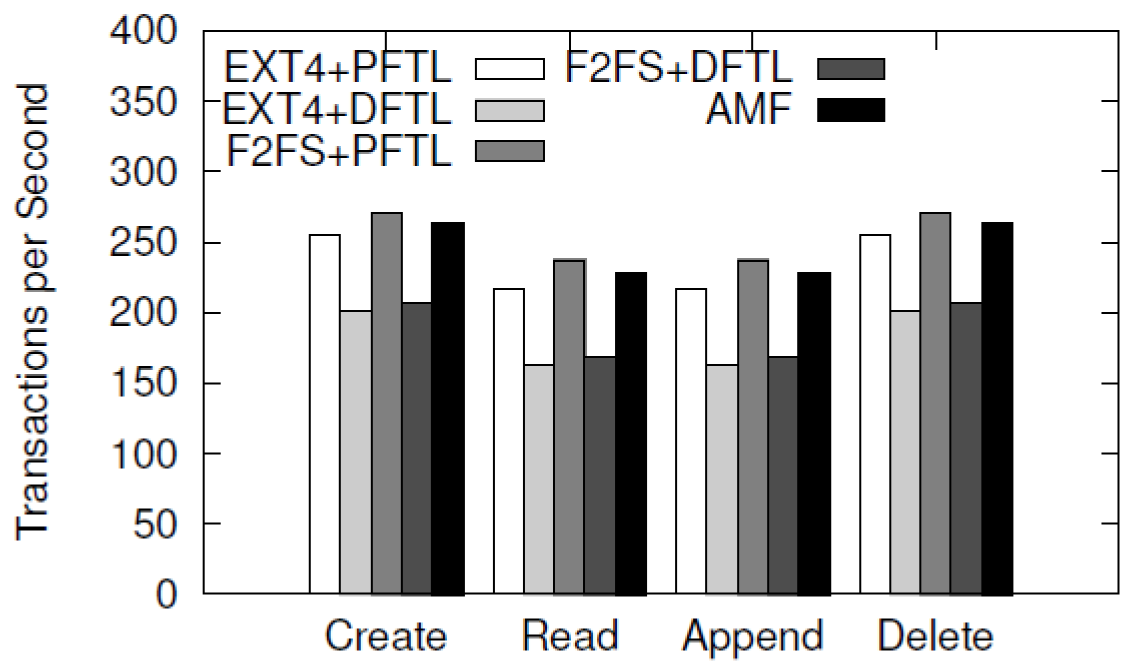
I/O suspension cause by dirty eviction of mapping entries

Amplified by low hit ratio of mapping table



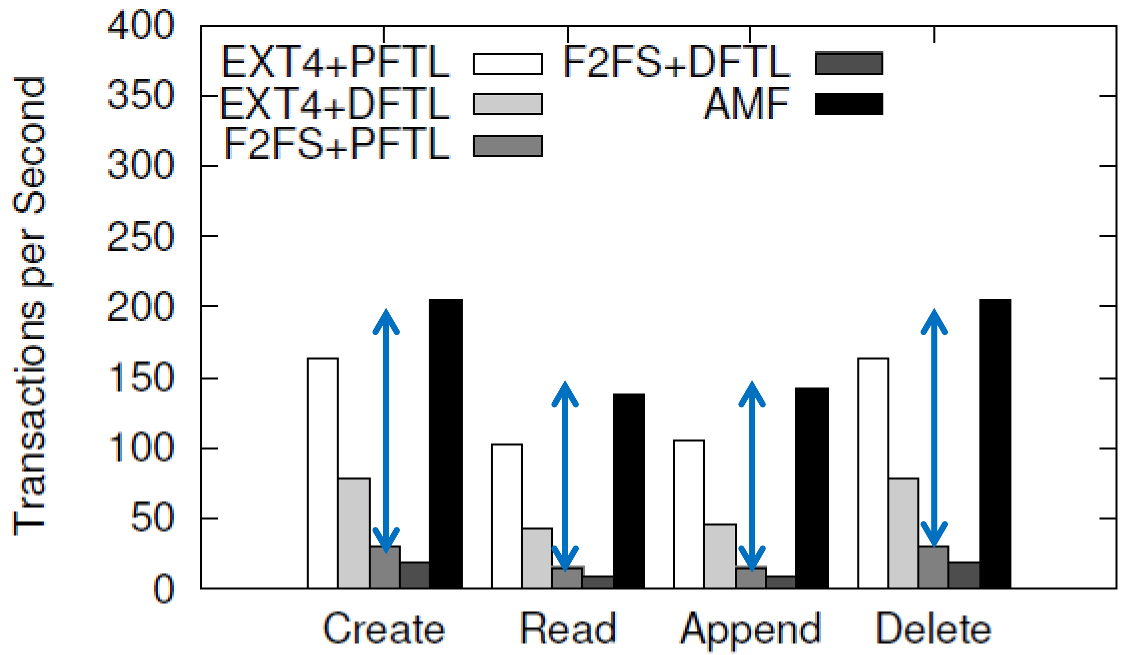
Duplication of log-structured management

Postmark Benchmark Results



(a) Postmark(L)

Low utilization of storage -> Reduced GC



(b) Postmark(H)

High utilization of storage -> Great GC