Write Optimization of Log-structured Flash File System for Parallel I/O on Manycore Servers

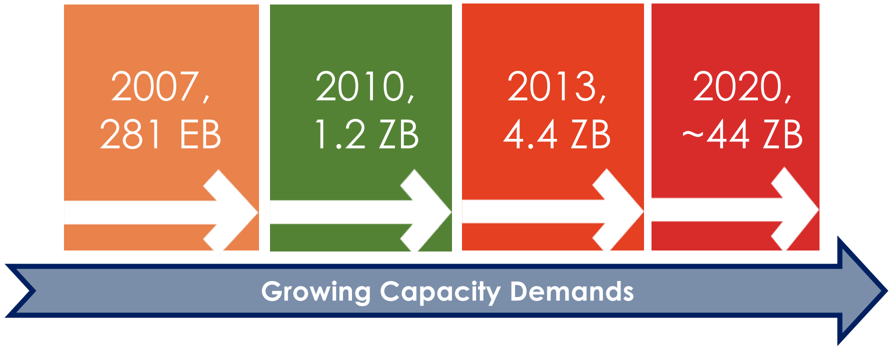
Chang-Gyu Lee, Hyunki Byun, Sunghyun Noh, Hyeongu Kang, Youngjae Kim
Department of Computer Science and Engineering
Sogang University, Seoul, Republic of Korea

SYSTOR '19

Sogang University

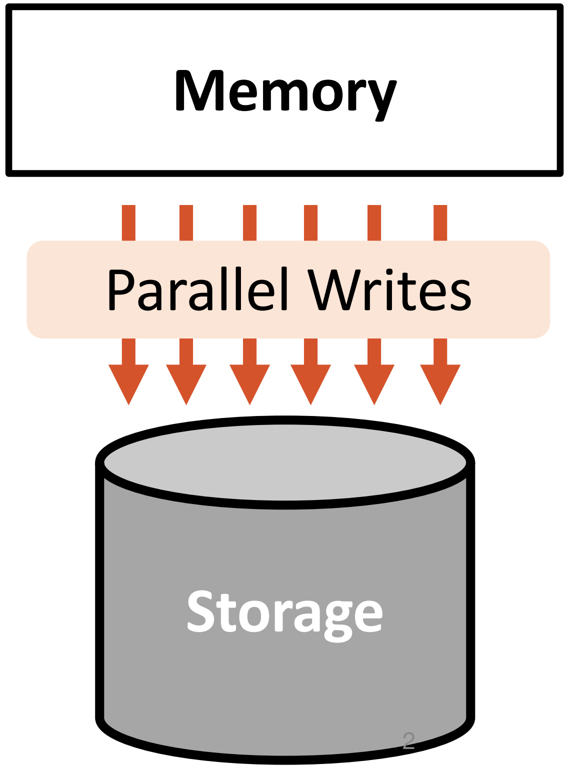
Data Intensive Applications

- Massive data explosion in recent years and expected to grow



- Database Applications

MySQL
LEVELDB
RocksDB



Manycore CPU and NVMe SSD

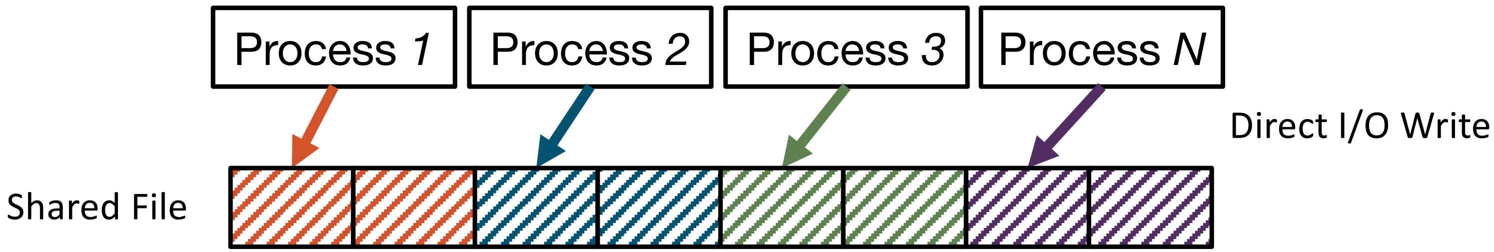
Manycore Servers

Parallel Writes

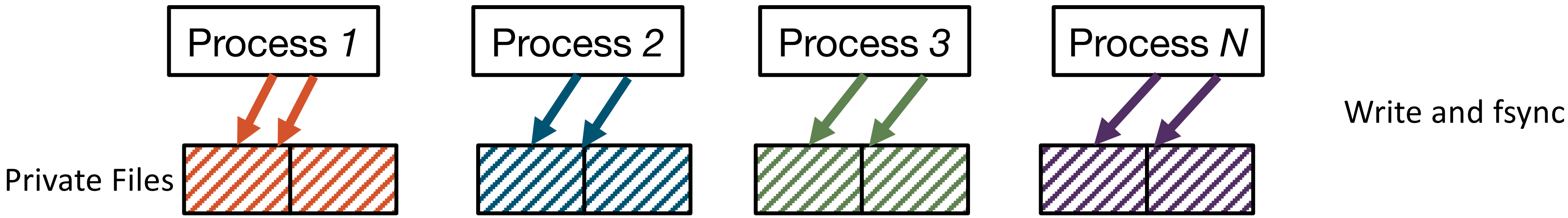
High-Performance SSD

What are Parallel Writes?

- Shared File Writes (DWOM from FxMark[ATC'16])
  - Multiple processes write private regions on a single file.

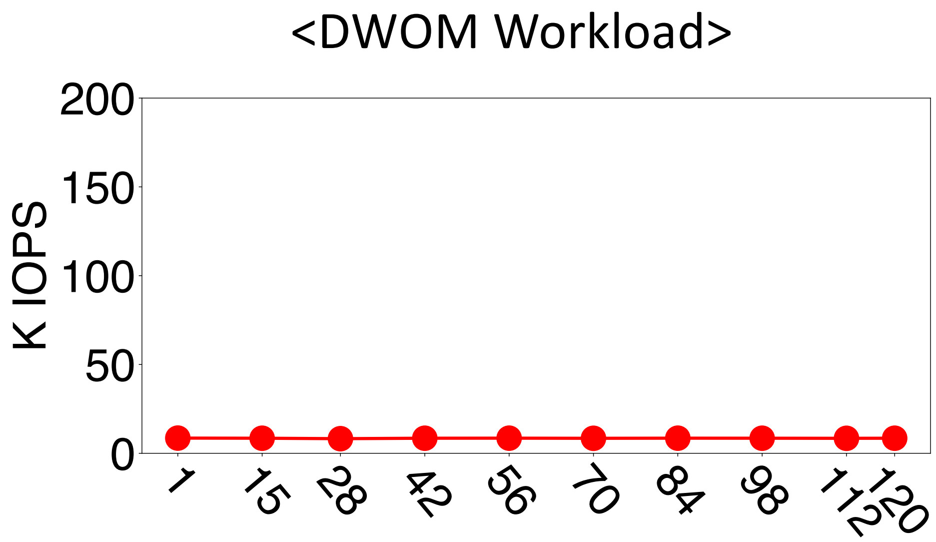


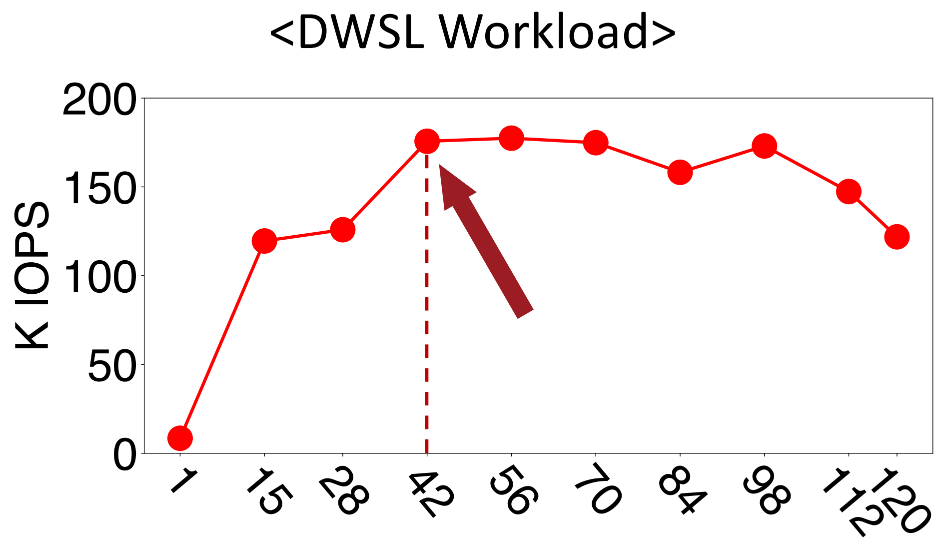
- Private File Write with FSYNC (DWSL from FxMark[ATC'16])
  - Multiple processes write private files, then call fsync system calls.



* FxMark[ATC'16]: Min. et. al., "Understanding Manycore Scalability of File Systems", USENIX ATC 2016

Preliminary Results





- In DWOM workload, the performance does not scale.
- In DWSL workload, the performance does not scale after 42 cores.

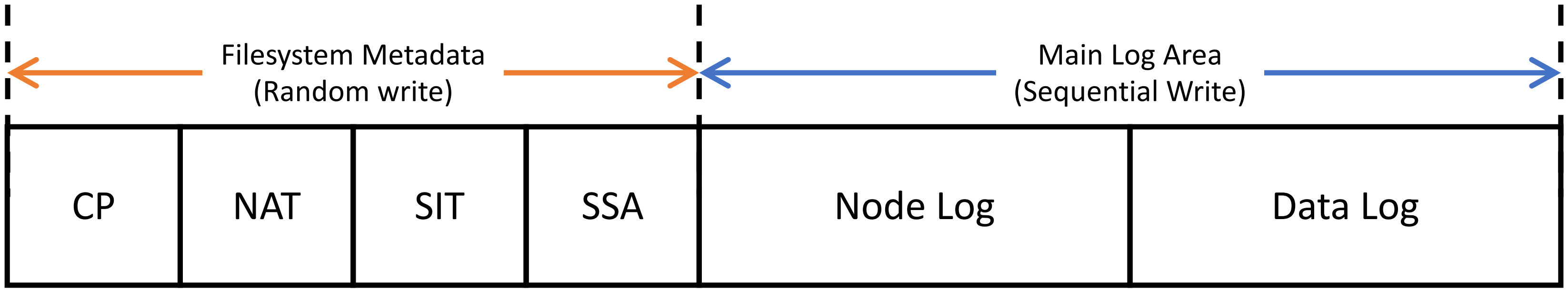
Sogang University

Contents

- Introduction and Motivation
- Background: F2FS
- Research Problems
  - Parallel Writes do never scale with respect to the increased number of cores on Manycore servers.
- Approaches
  - Applying Range-Locking
  - NVM Node Logging for file and file system metadata
  - Pin-Point Update to completely eliminate checkpointing
- Evaluation Results
- Conclusion

F2FS: Flash Friendly File System

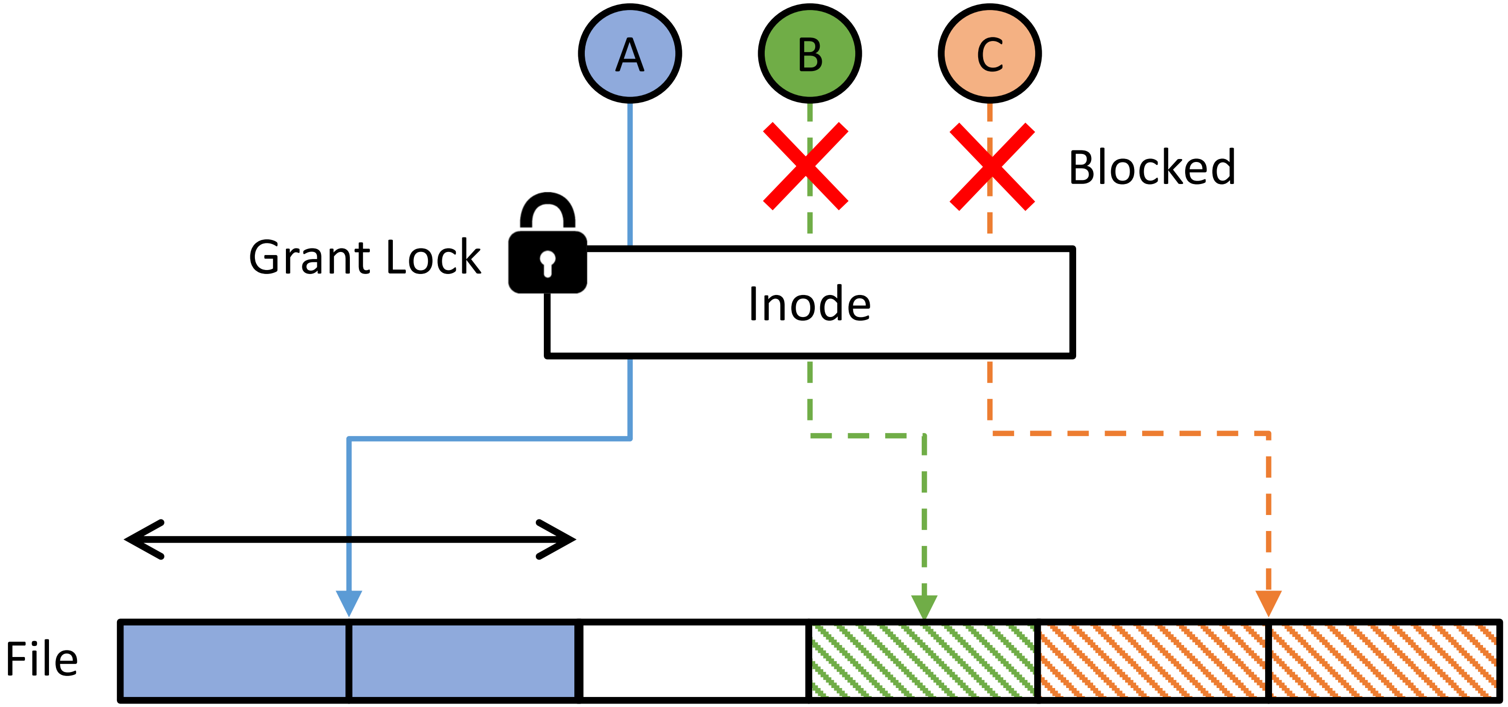
- F2FS is a log-structured file system designed for NAND Flash SSD.
- F2FS employs two types of logs to benefit with Flash's parallelism and garbage collection.
  - **Data log** for directory entry and user data
  - **Node log** for inode and indirect node
- **Node Address Table (NAT)** translates *Node id (NID)* to *block address*.
- In memory, block address of an NAT entry is updated when corresponding Node Log is flushed.
- Entire **NAT** is flushed to the storage device during <u>checkpointing</u>.



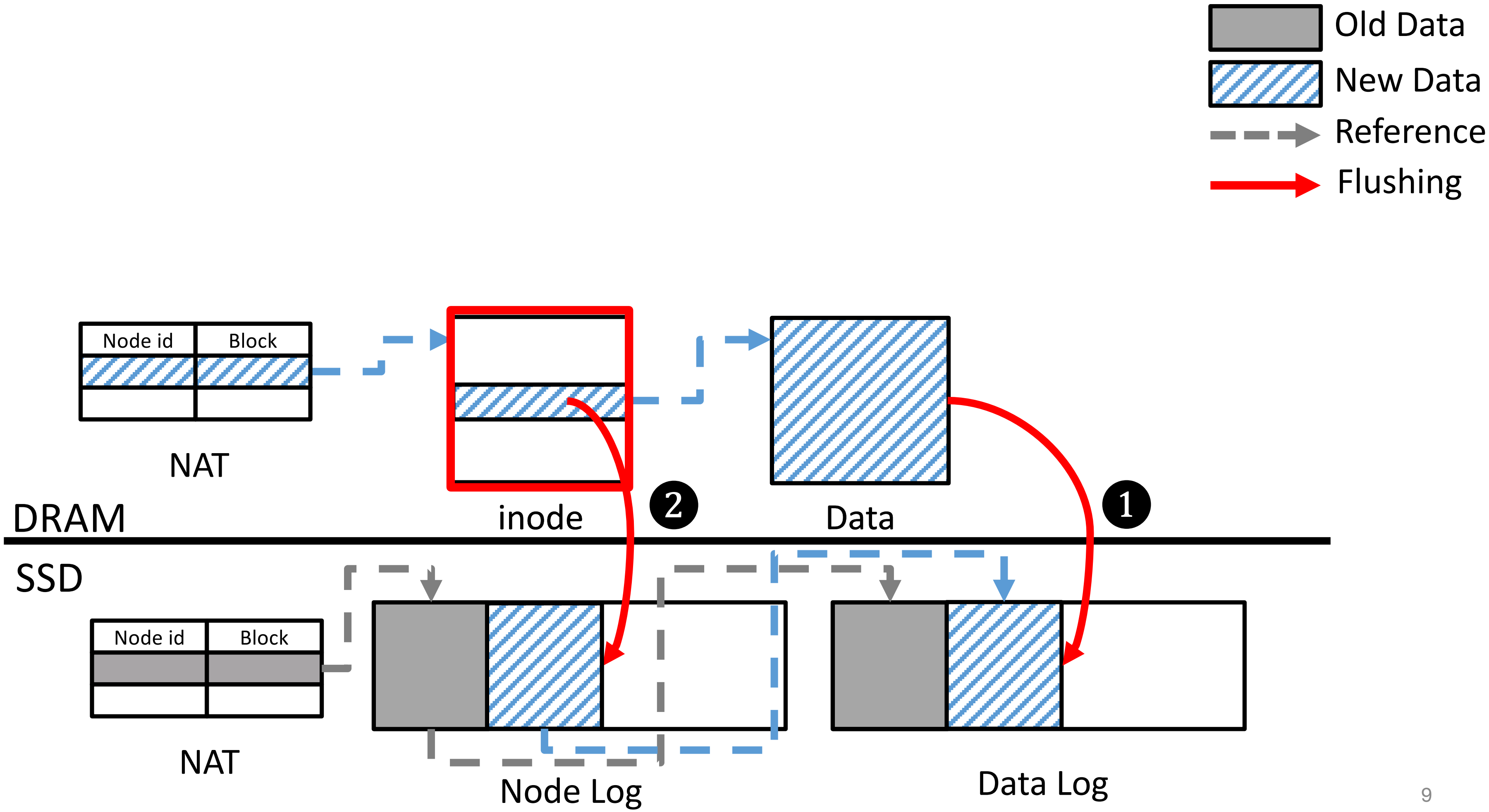
SOGANG UNIVERSITY

Problem(1): Serialized Shared File Writes

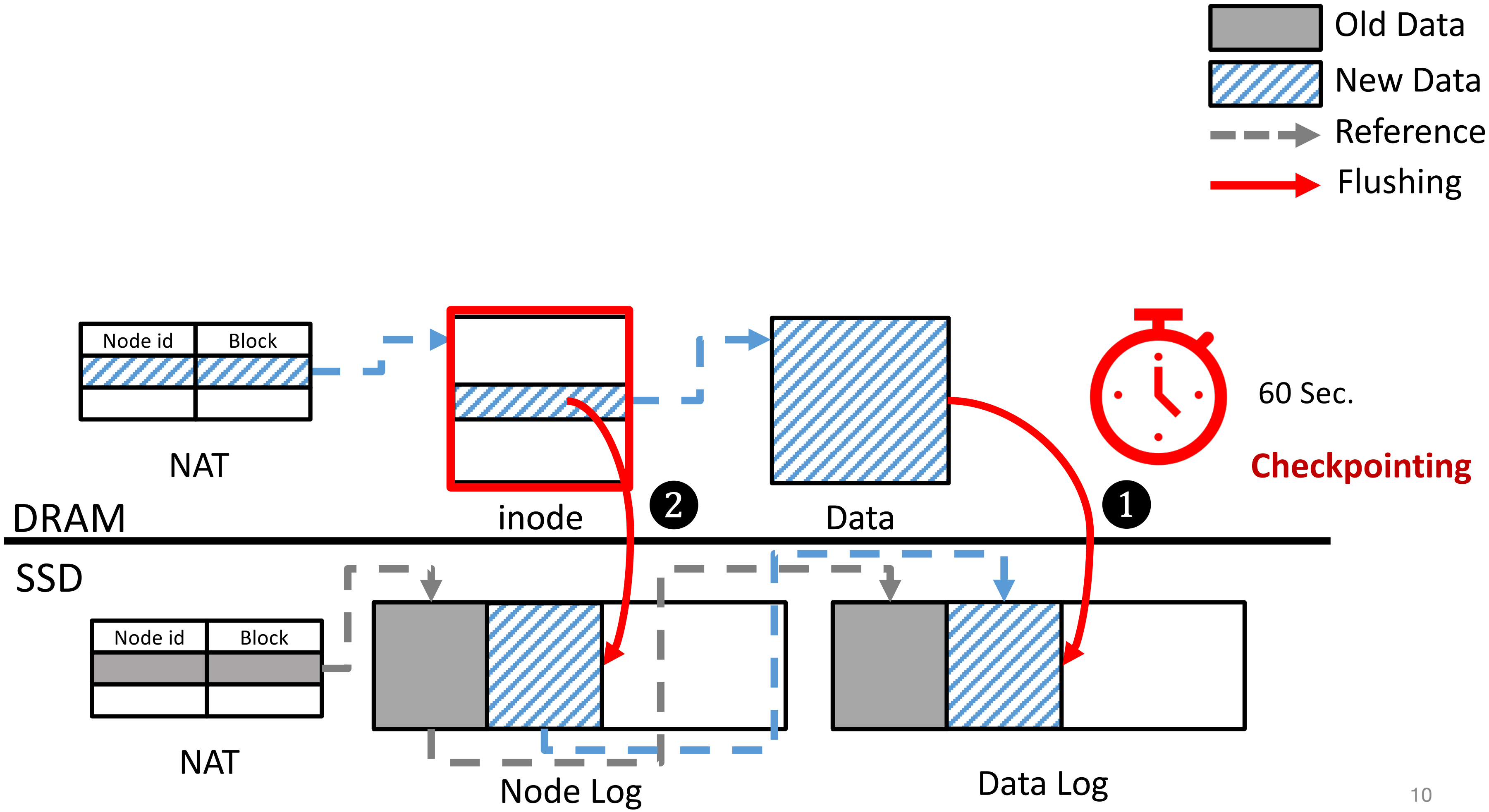
- Single file write



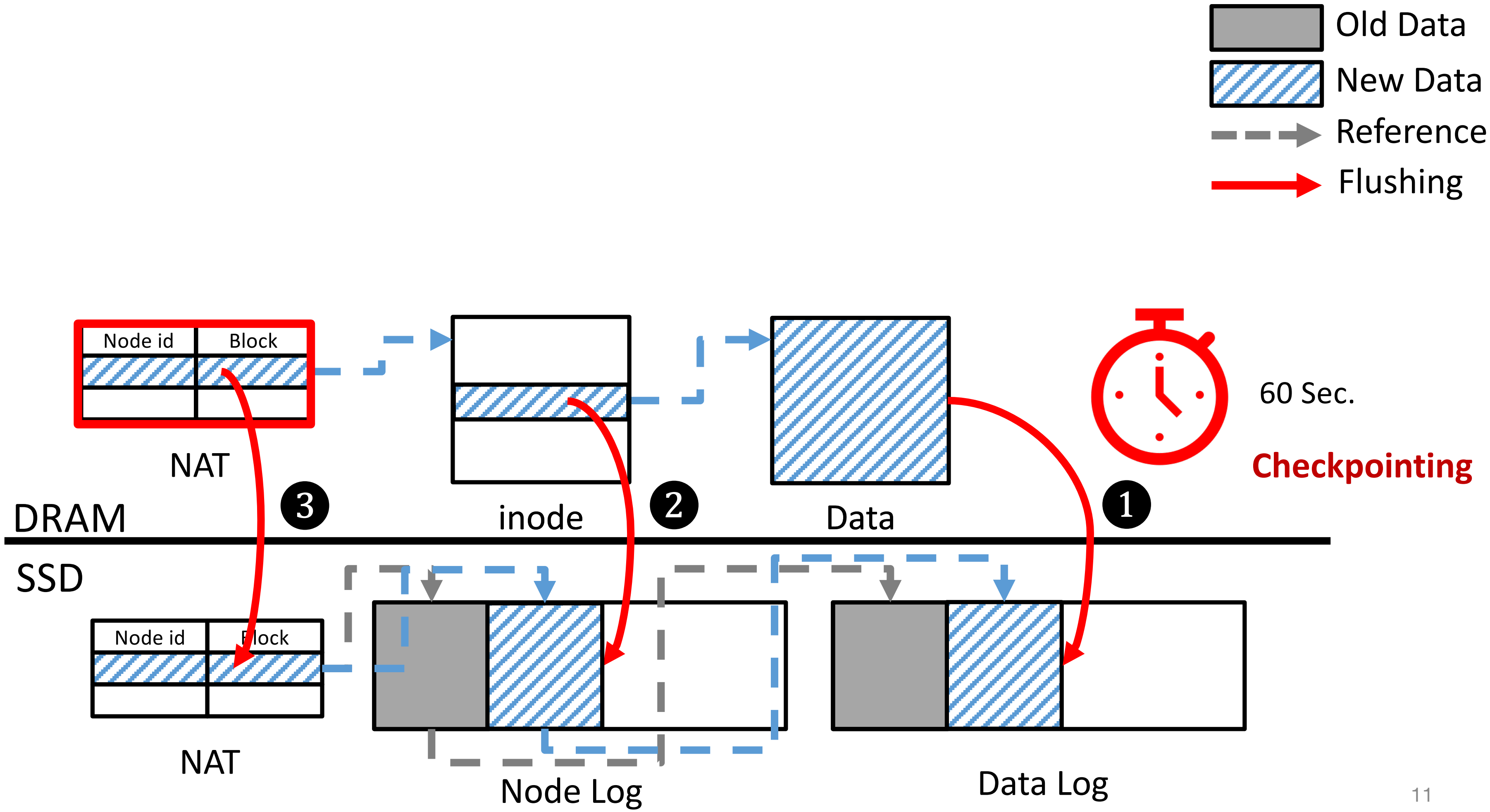
Problem(2): *fsync* Processing in F2FS



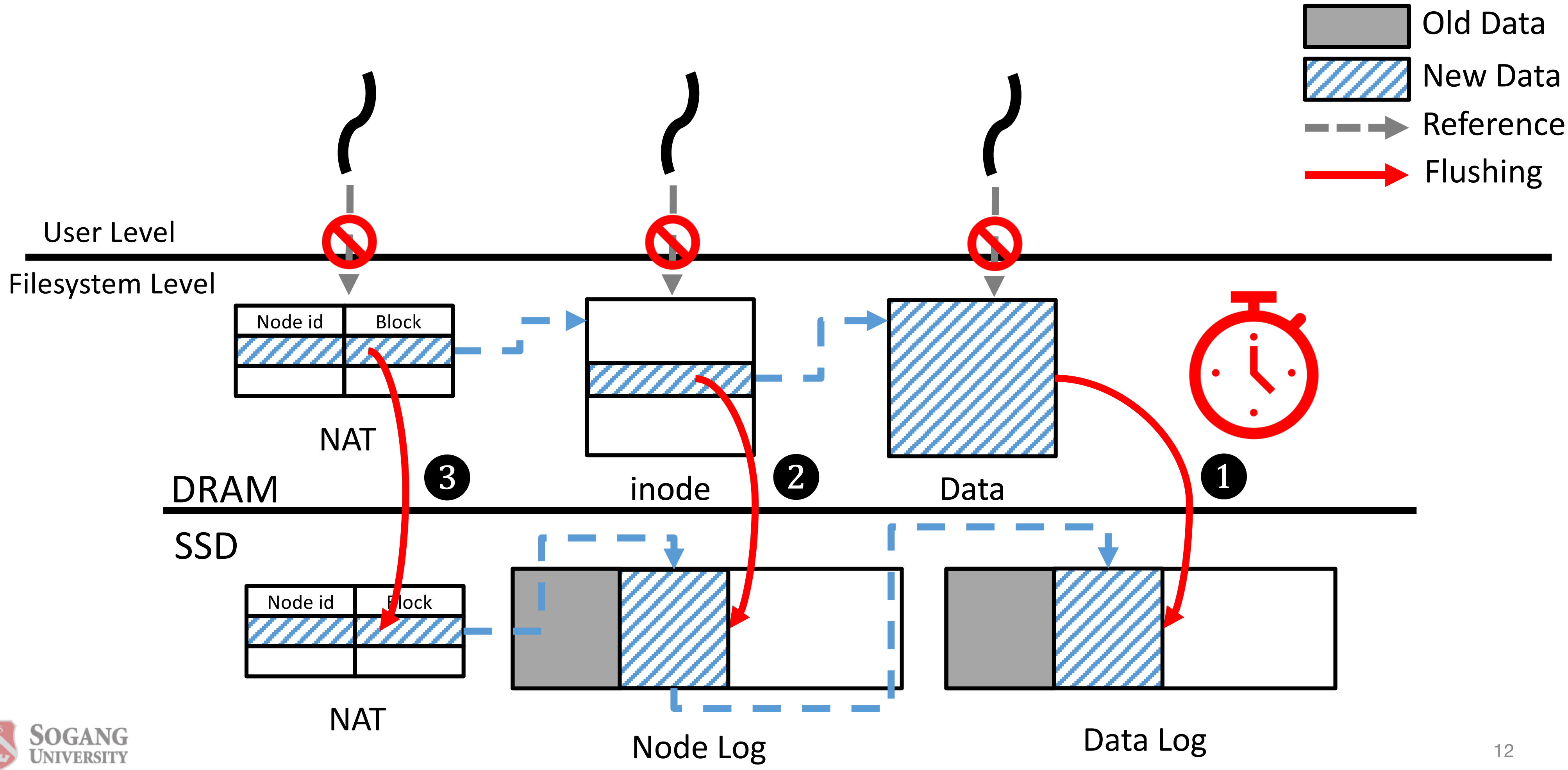
Problem(3): I/O Blocking during *Checkpointing*



Problem(3): I/O Blocking during *Checkpointing*



Problem(3): I/O Blocking during *Checkpointing*



Summary

- We identified the causes of bottlenecks in F2FS for parallel writes as follows.

1. Serialization of parallel writes on a single file

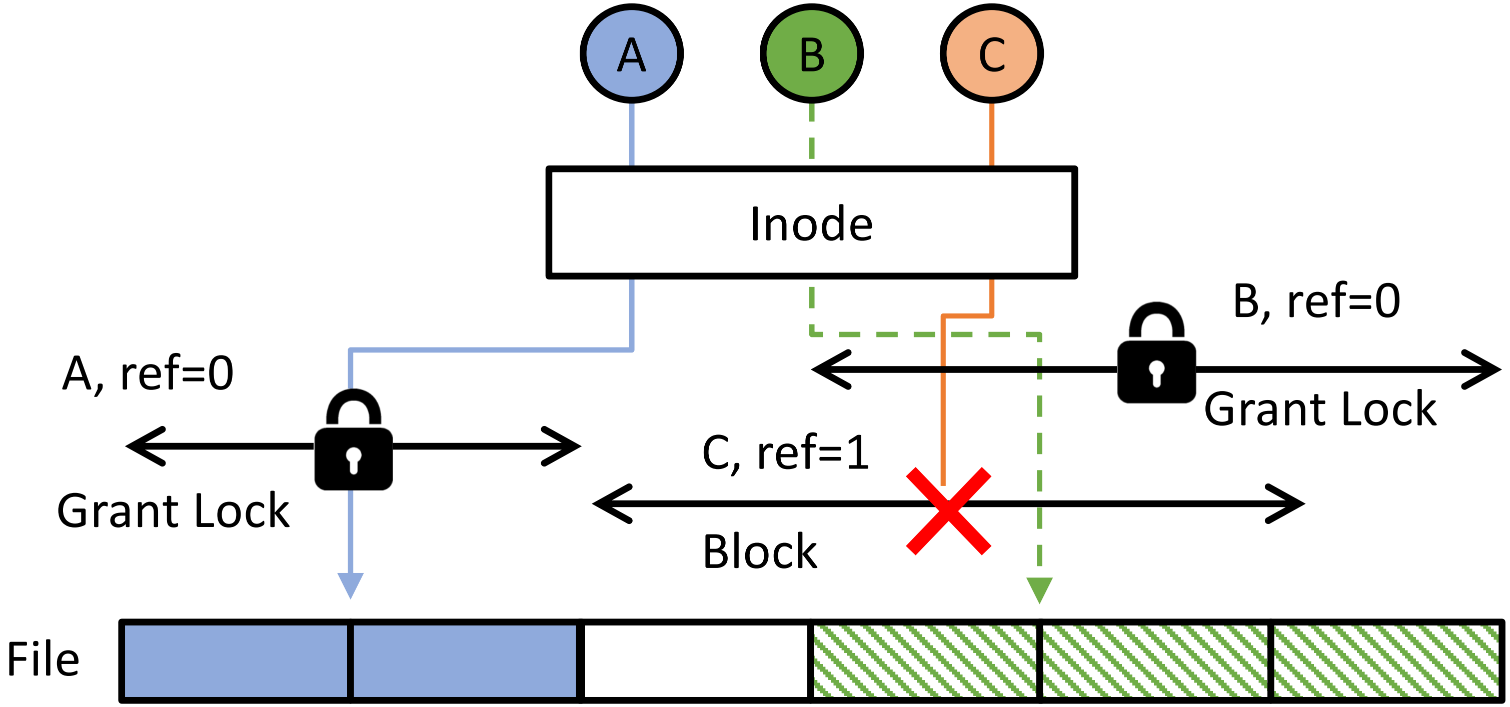
2. High latency of *fsync* system call

3. I/O blocking by *checkpointing* of F2FS

Approach(1): Range Locking

- In F2FS, parallel writes to a single file are serialized by *inode mutex* lock.

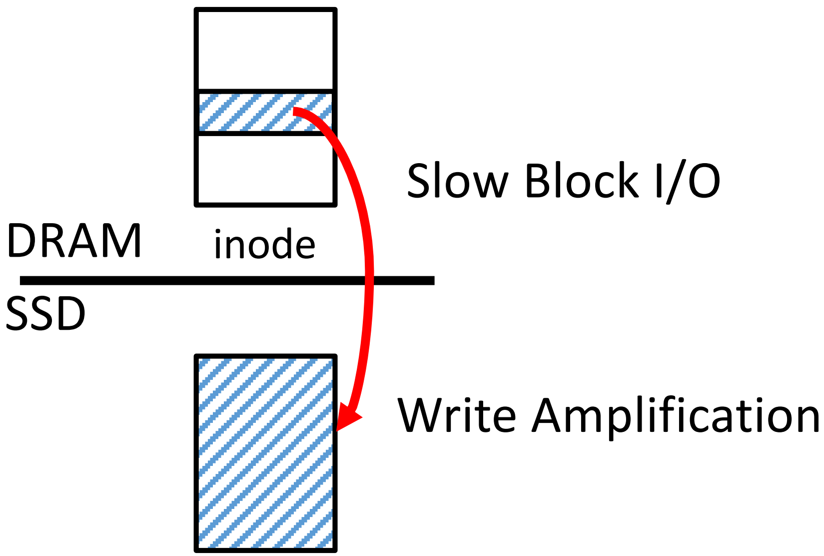
We employed a range-based lock to allow parallel writes on a single file.

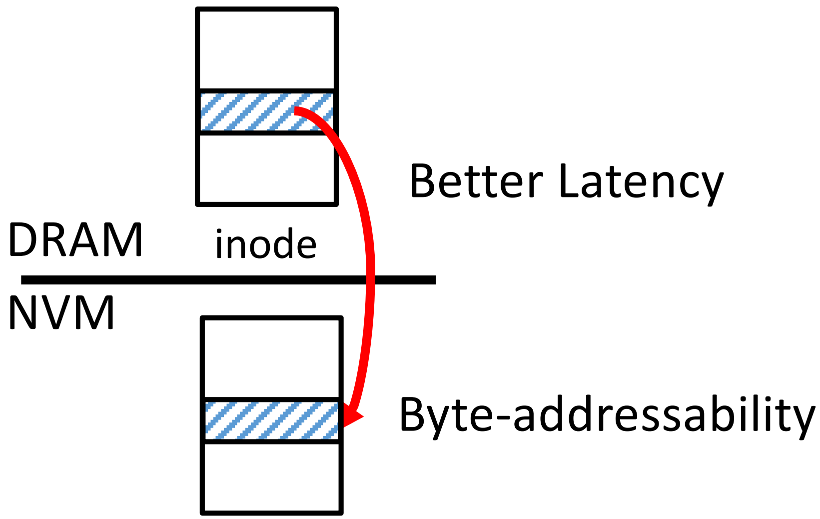


Approach(2): High Latency of *fsync* Processing

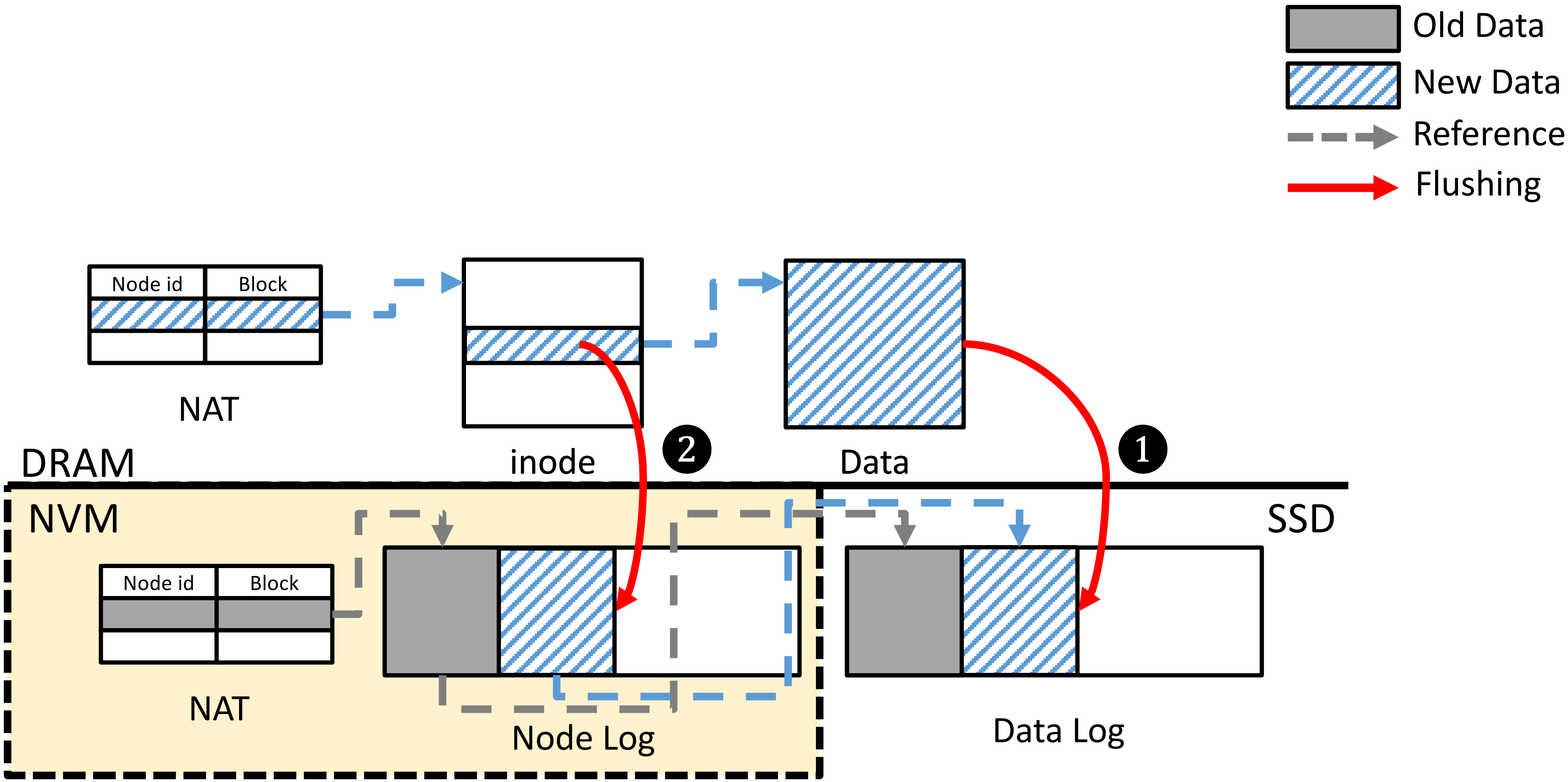
- When *fsync* is called, F2FS has to flush data and metadata.
  - Even if only small portion of metadata is changed, a block has to be flushed.
  - The latency of *fsync* is dominated by block I/O latency.

To mitigate high latency of fsync, we propose NVM Node Logging and fine-graind inode.

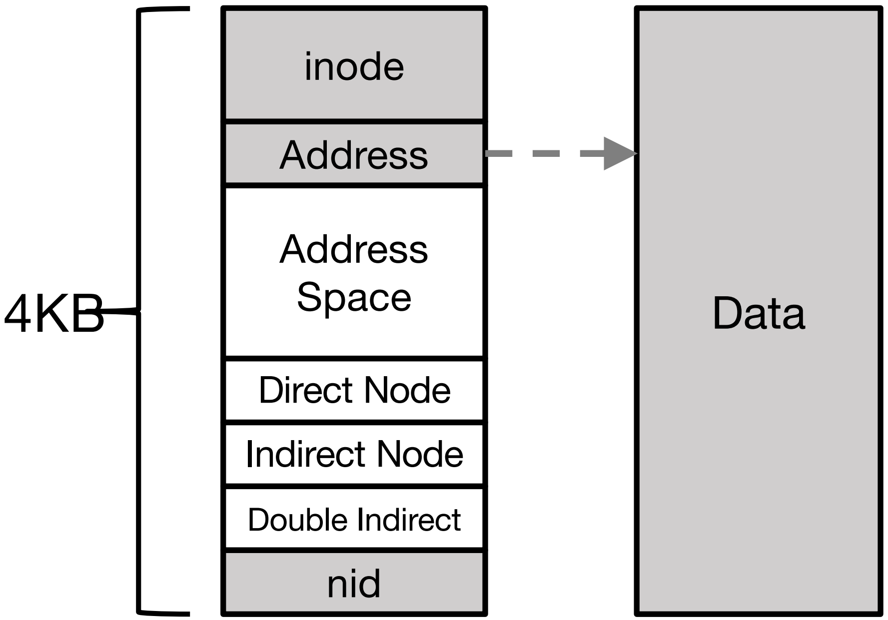




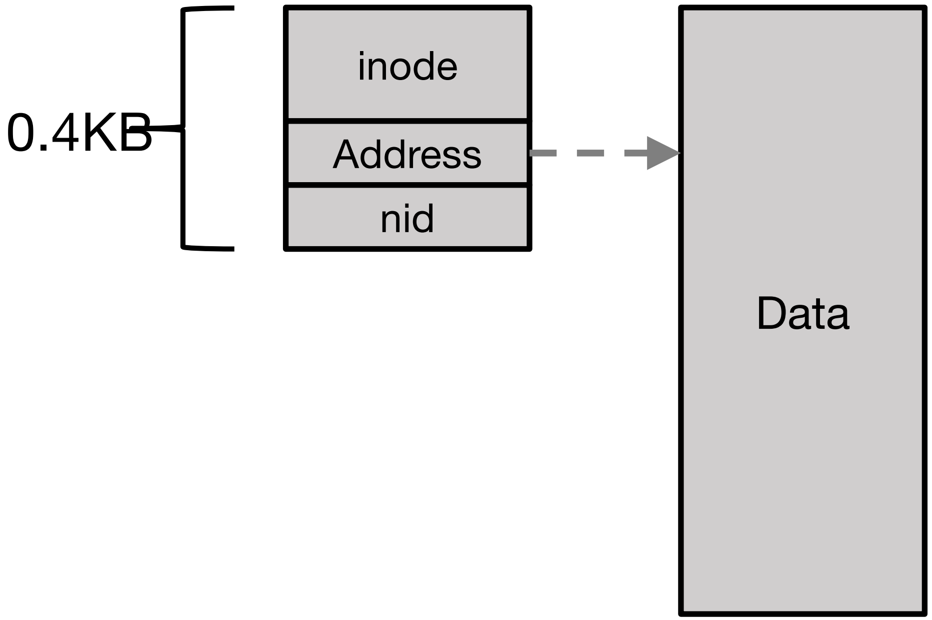
Approach(2): Node Logging on NVM



Approach(3): Fine-grained inode Structure



inode in baseline F2FS



Fine-grained inode

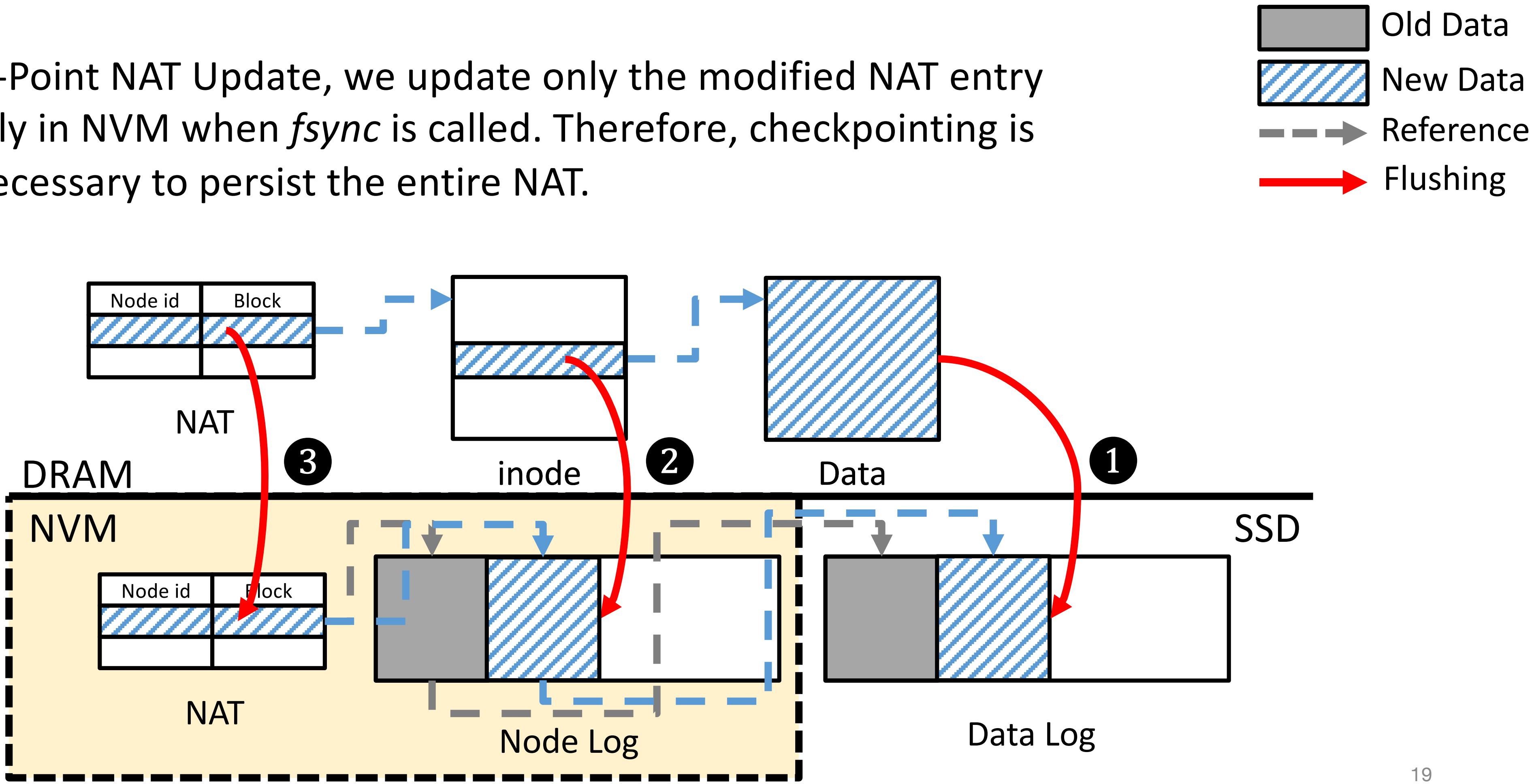
Approach(4): Pin-Point NAT Update

- Frequent *fsync* calls trigger checkpointing in F2FS
- However, F2FS blocks all incoming I/O requests during checkpointing.

To eliminate checkpointing, we propose Pin-Point NAT Update.

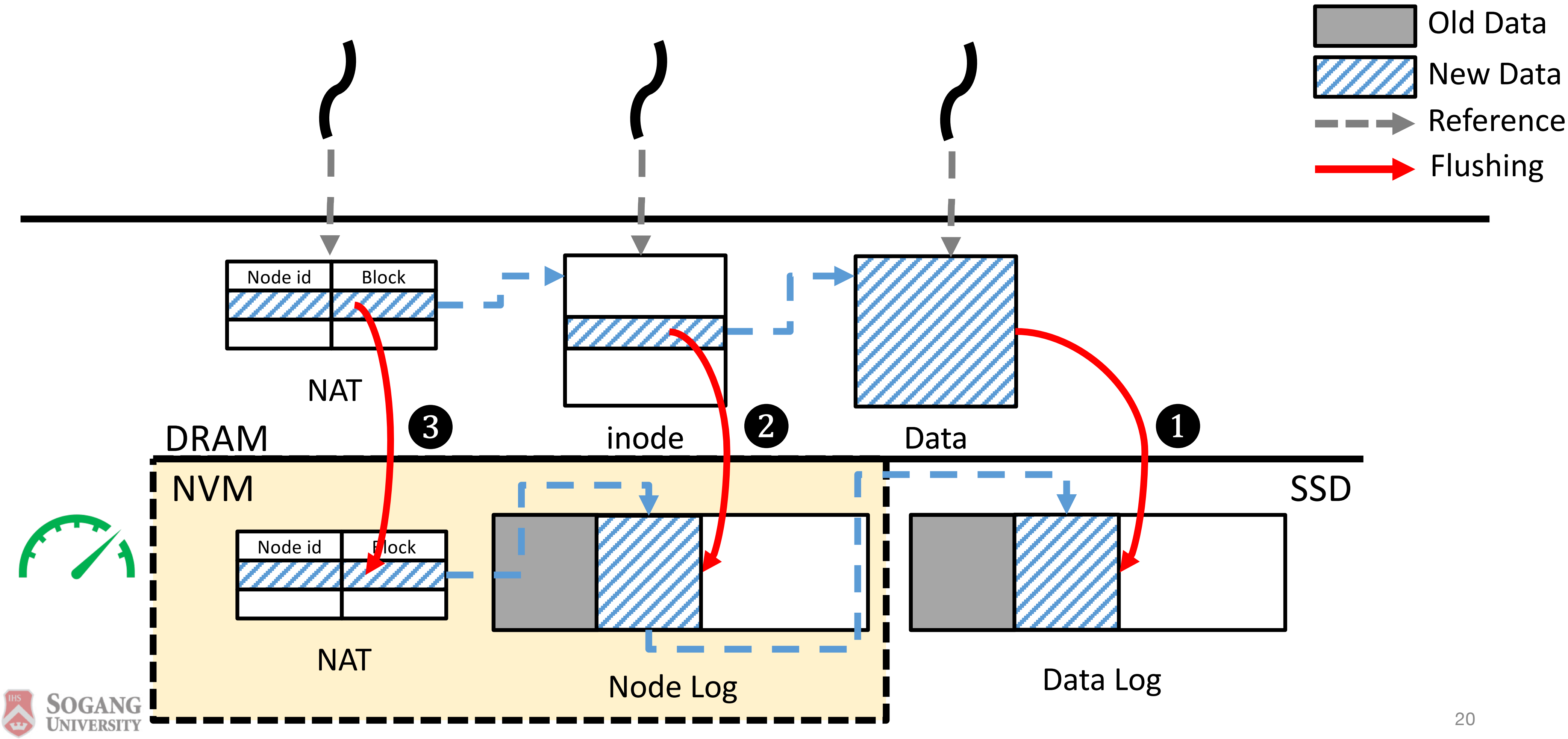
Approach(4): Pin-Point NAT Update

In Pin-Point NAT Update, we update only the modified NAT entry directly in NVM when *fsync* is called. Therefore, checkpointing is not necessary to persist the entire NAT.



SOGANG UNIVERSITY

Approach(4): Pin-Point NAT Update



Evaluation Setup

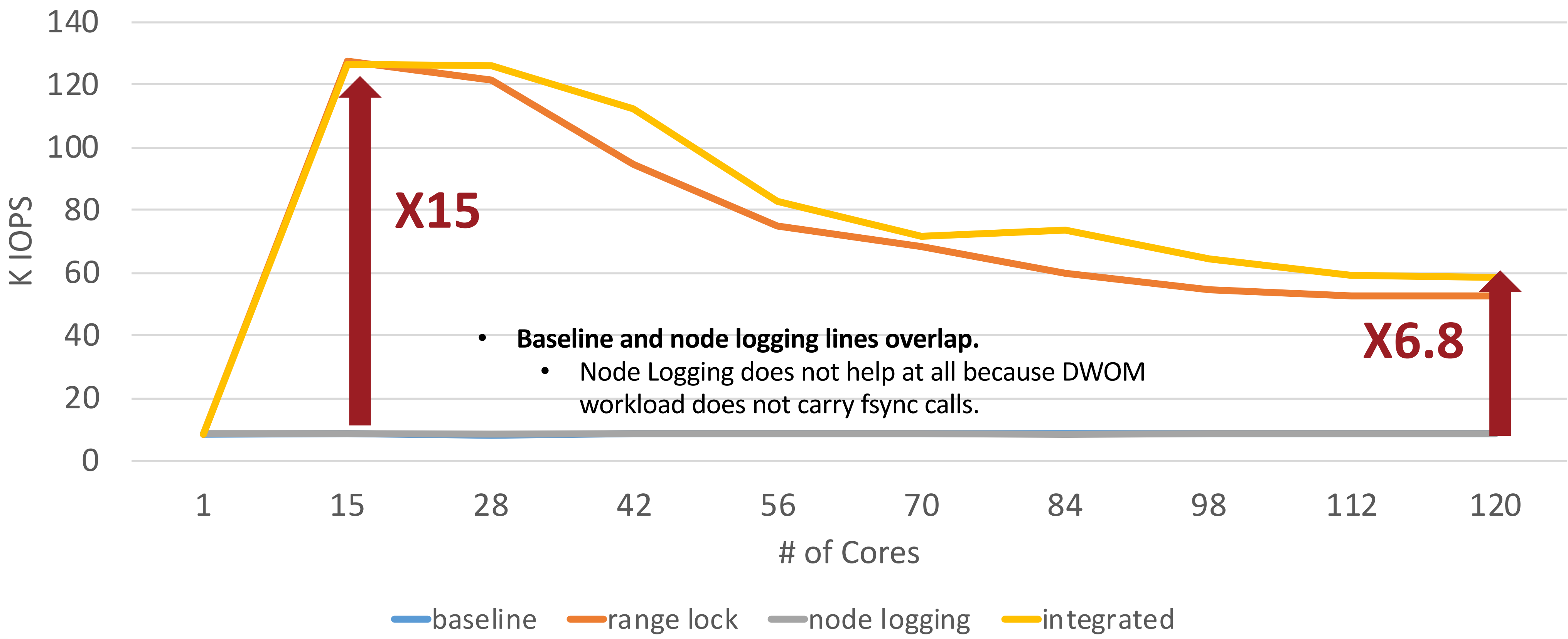
- Microbenchmark (FxMark)
  - DWOM
    - Shared File Write
  - DWSL
    - Private File Write with *fsync*

- Test-bed
  - IBM x3950 X6 Manycore Server

CPU	Intel Xeon E7-8870 v2 2.3GHz 8 CPU Nodes (15 Cores per Node) Total 120 cores
RAM	740GB
SSD	Intel SSD 750 Series 400GB (NVMe) Read: 2200 MB/s, Write: 900 MB/s
NVM	32GB Emulated as PMEM device on R AM
OS	Linux kernel 4.14.11

* FxMark[ATC'16]: Min. et. al., "Understanding Manycore Scalability of File Systems", USENIX ATC 2016

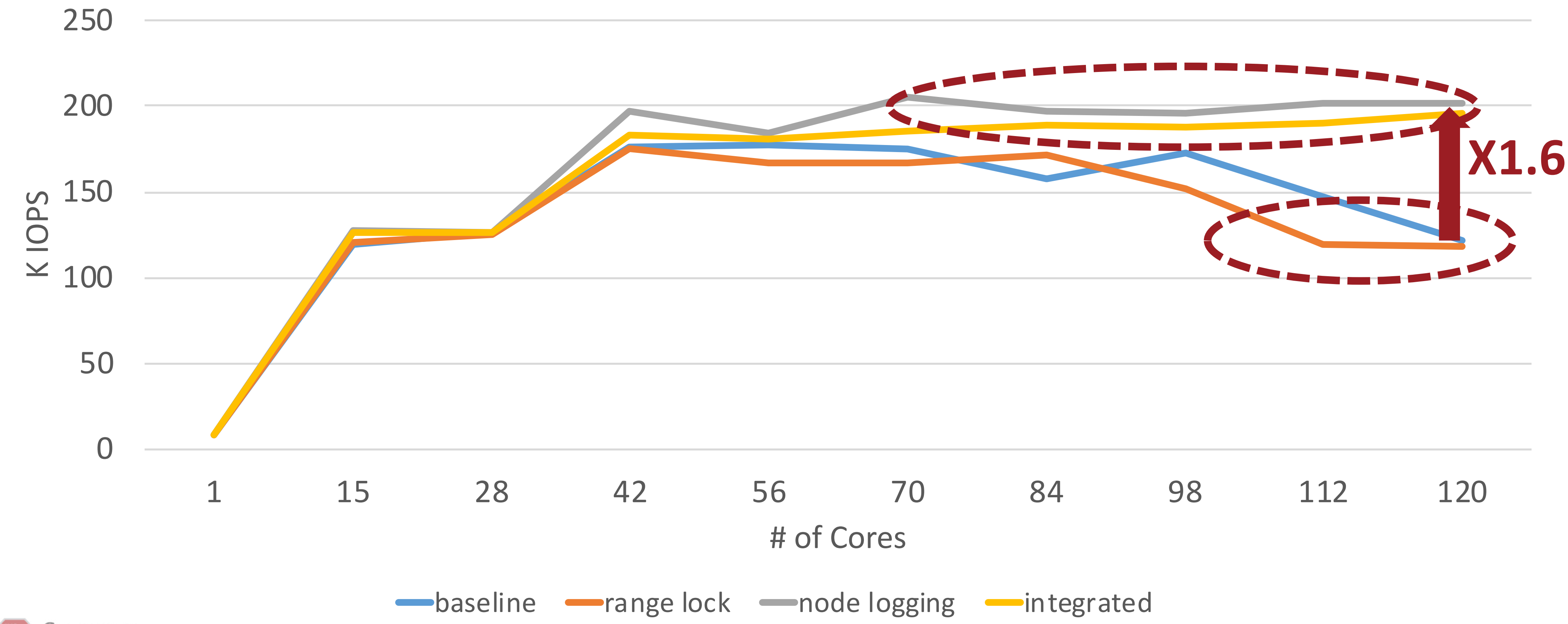
Shared File Write (DWOM Workload)



- **Baseline and node logging lines overlap.**
  - Node Logging does not help at all because DWOM workload does not carry fsync calls.

Sogang University

Frequent *fsync* (DWSL Workload)



Conclusion

- We identified performance bottlenecks of F2FS for parallel writes.
  1. Serialization of share file writes on a file
  2. High latency of fsync operations in F2FS
  3. High I/O blocking times during checkpointing.

- To solve these problem, we proposed
  1. File-level Range Lock to allow parallel writes on a shared file
  2. NVM Node Logging to provides lower latency for updating file/file system metadata
  3. Pin-Point NAT Update to eliminate I/O blocking times of checkpointing

Q&A

Thank you!

- Contact: **Changgyu Lee** (changgyu@sogang.ac.kr)
  Department of Computer Science and Engineering
  Sogang University, Seoul, Republic of Korea