iJournaling: Fine-Grained Journaling for Improving the Latency of Fsync System Call

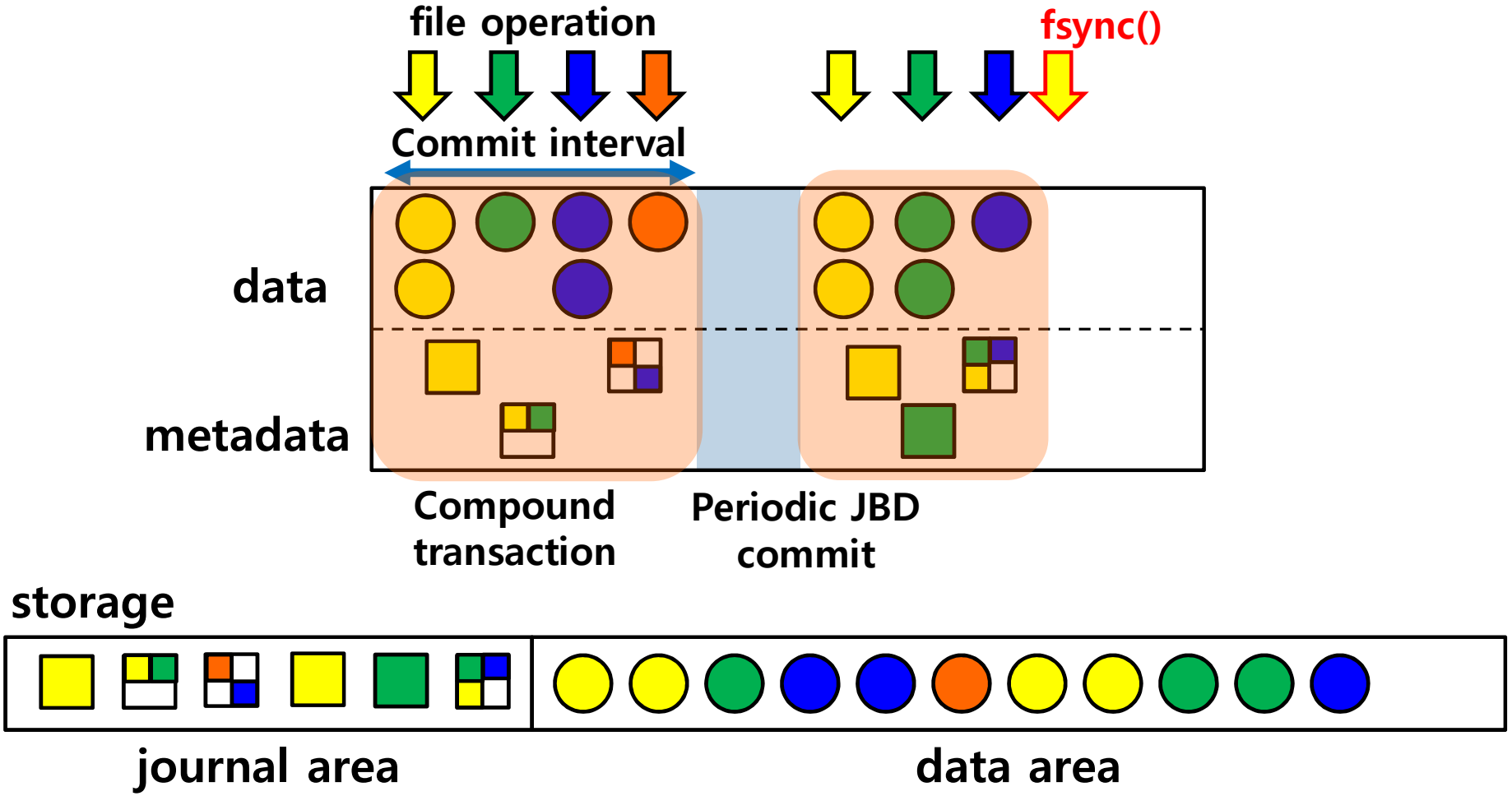
Daejun Park and Dongkun Shin
Sungkyunkwan University, Korea
pdaejun@skku.edu, dongkun@skku.edu

Why fsync() latency is important?

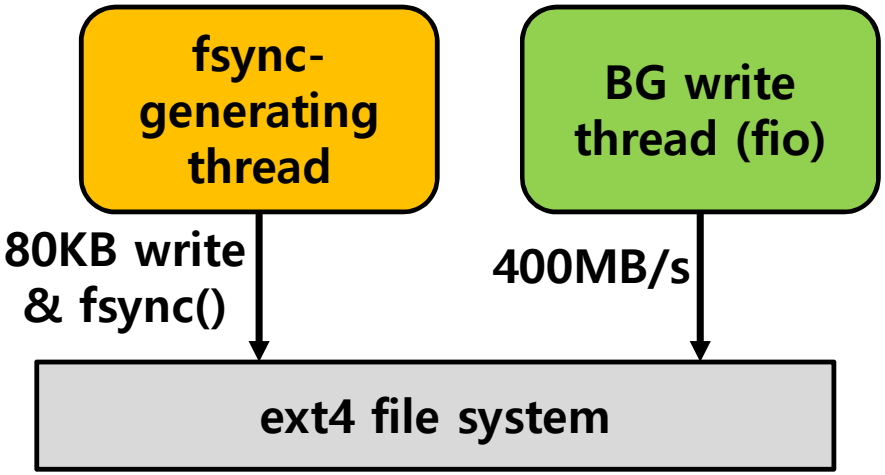
- **Fsync() system call**
  - used by many applications to guarantee durability of a file
  - blocks until the flushing data is completed.
- **Database management system**
  - MYSQL (tpcc-mysql) calls fsync() about **140 call/s**
- **Smartphone application**
  - A burstshot (20 pics) at smartphone calls fsync() **100 times**
  - Adding a phone number on contract app calls fsync() **7 times**

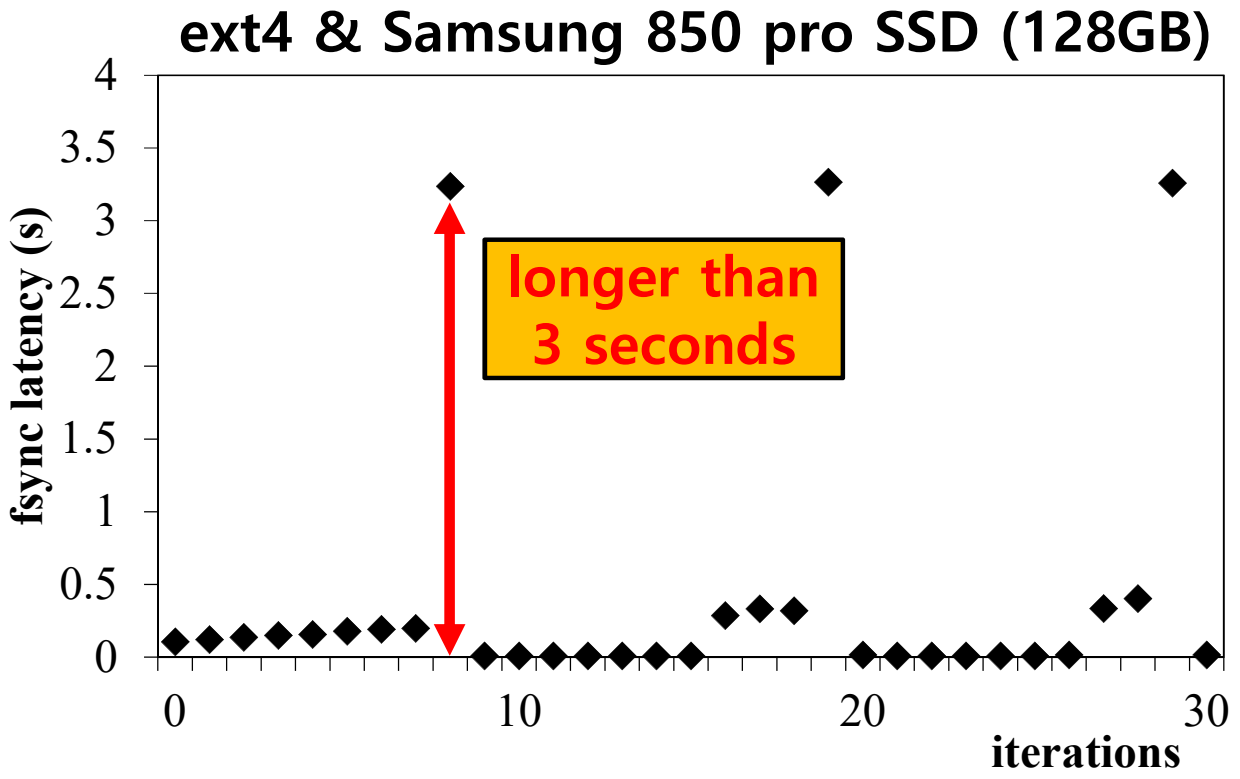
Fsync latency can affect on application performance

Legacy journaling in the ext4

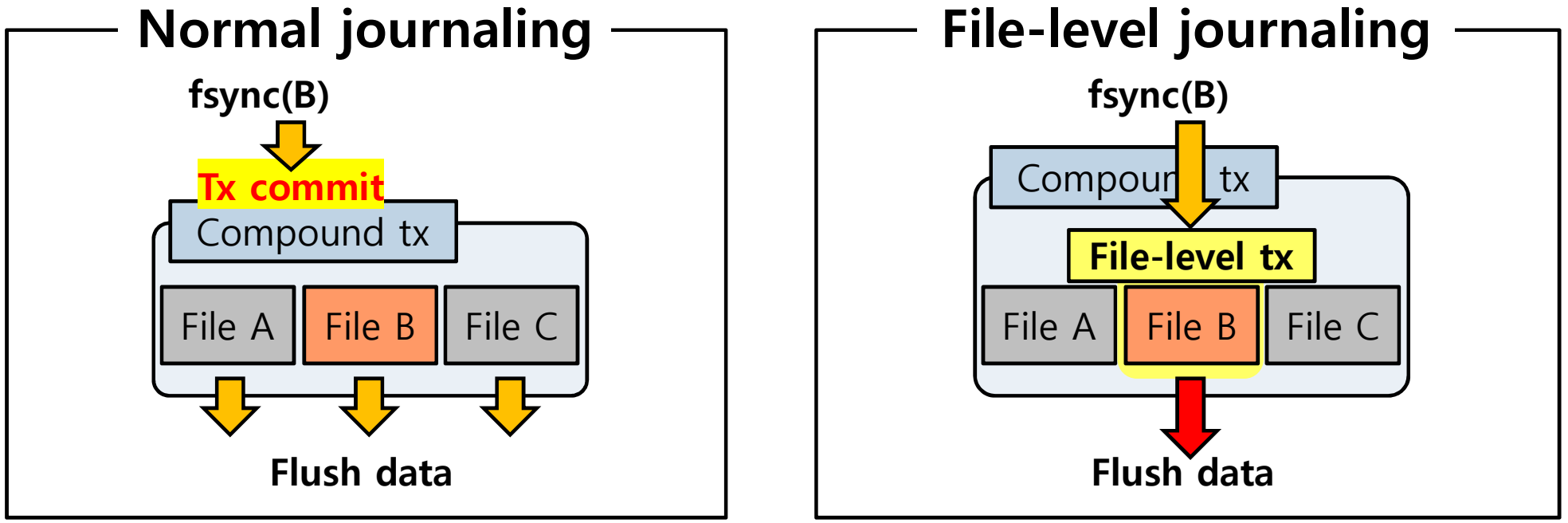


Long fsync() latency with heavy BG write





File level journaling

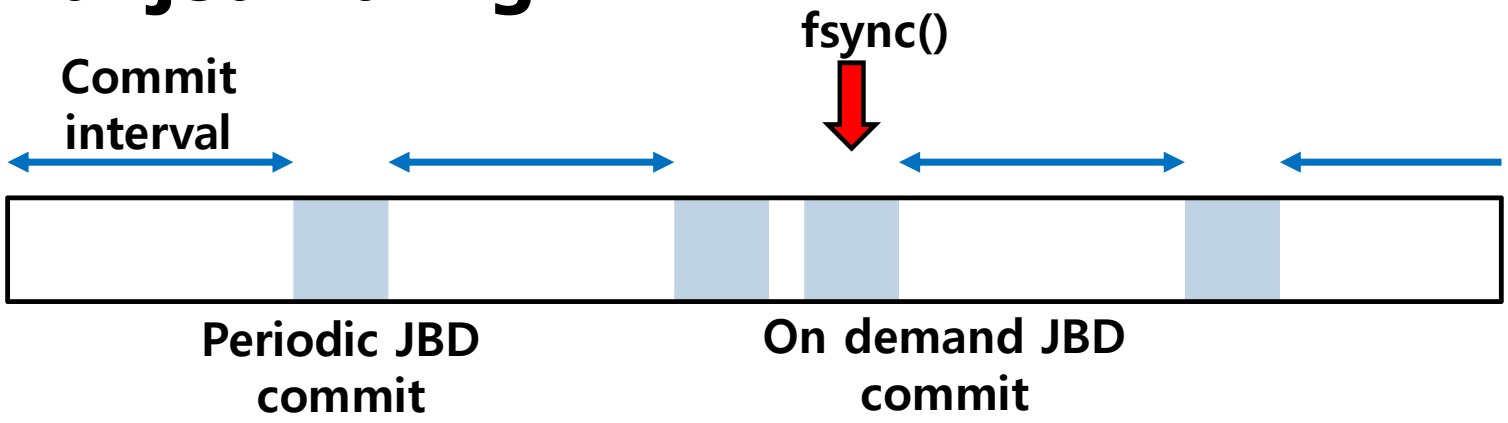


- **File-level journaling is needed for fsync()!**

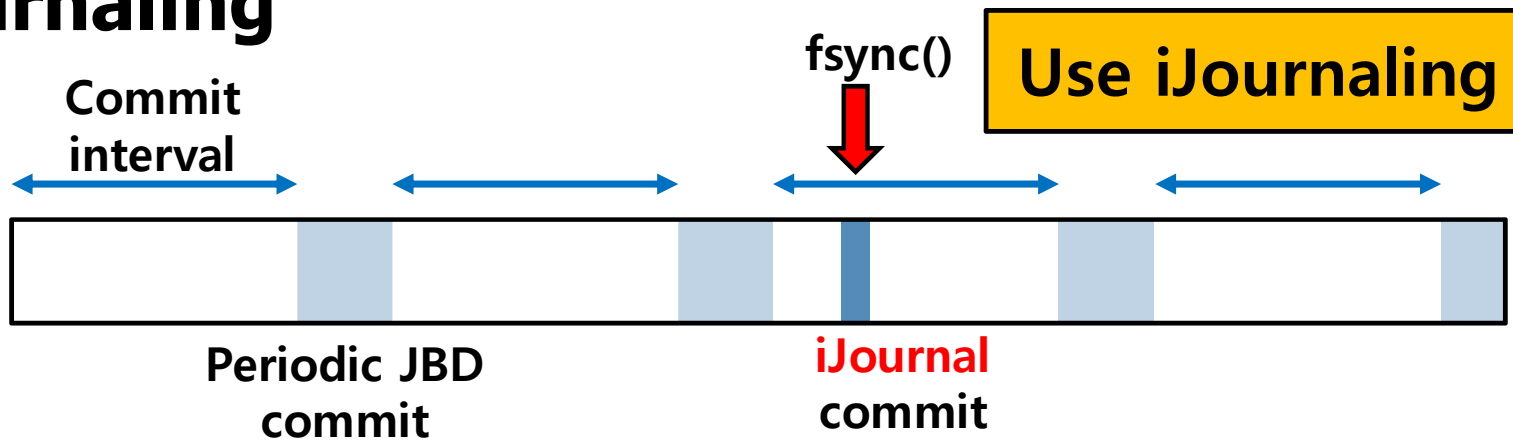
iJournaling

Hybrid journaling

- **Normal journaling**



- **iJournaling**



Use iJournaling on fsync

Related Work

- **CCFS[FAST '17], PBG[ISCE '14], Eager synching[NVMSA '14], IceFS[OSDI '14], Spanfs[ATC '05]**
  - Minimizes the compound transaction problem by isolation
- **ScaleFS[*], ZFS**
  - uses a logical logging technique.
  - performance overhead occurs because each file-system operation must record its own log
- **Xsyncfs [TOC '08], NoFS [FAST '12], and OptFS [SOSP '13]**
  - improved the fsync latency by delaying sync operations or changing the implementation of ordering constraint.

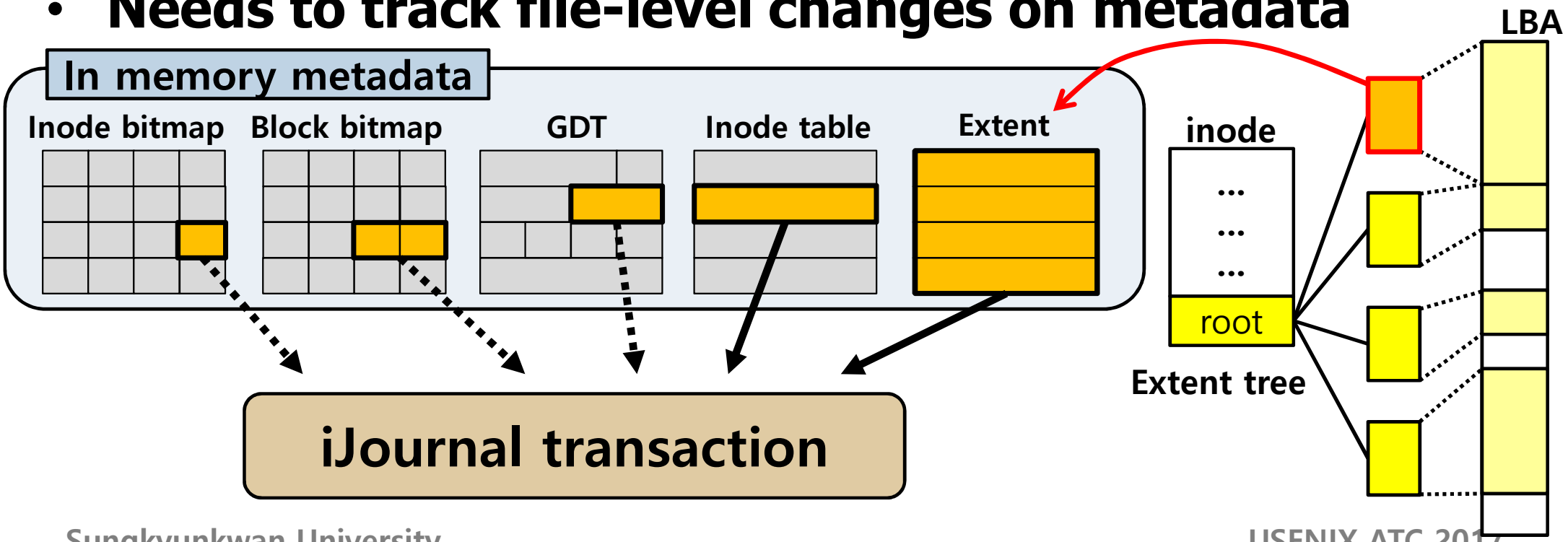
[*] Rasha Eqbal. ScaleFS: A multicore-scalable file system. Master's thesis, Massachusetts Institute of Technology, August 2014.

Challenges for iJournaling

1. **Physical logging**
   - One metadata block includes info. of multiple file operations
2. **Inter-file dependency**
   - Needs to flush files related to fsynced file
   - e.g. parent DE, hard link
3. **Managing iJournal area**
   - Sharing normal journal area or separated iJournal area

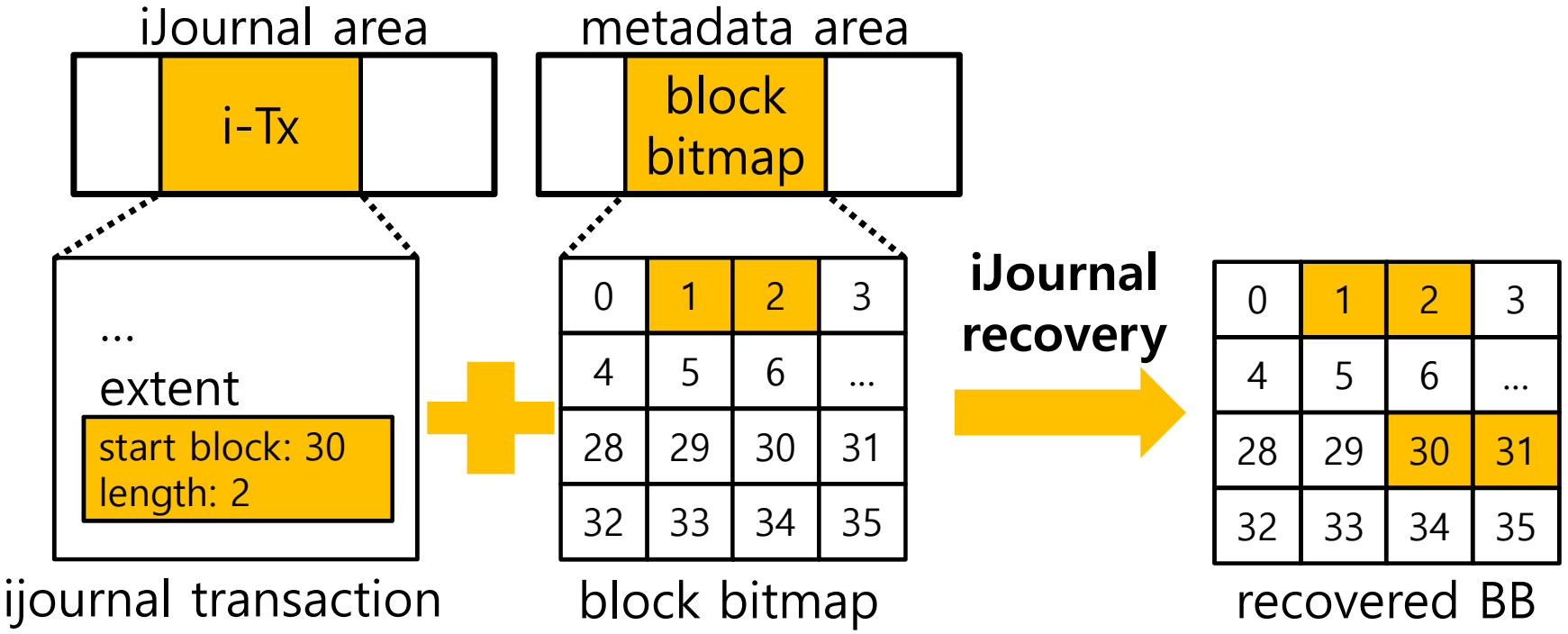
Legacy journaling – physical logging

- **Physical logging with 4KB block granularity**
  - multiple file operations share metadata
- **Needs to track file-level changes on metadata**



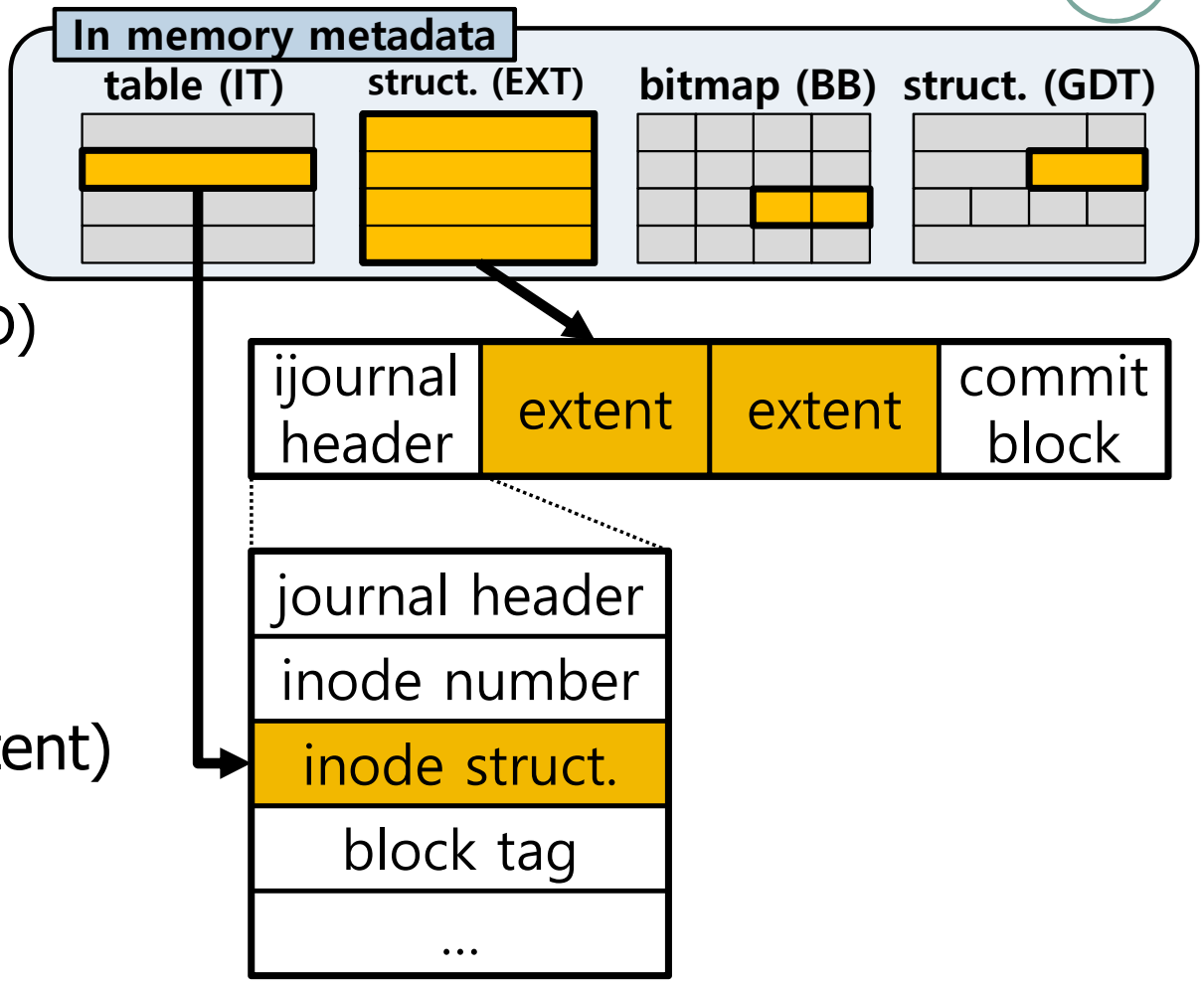
iJournaling – Logging and Recovery

- **Compare & update metadata**



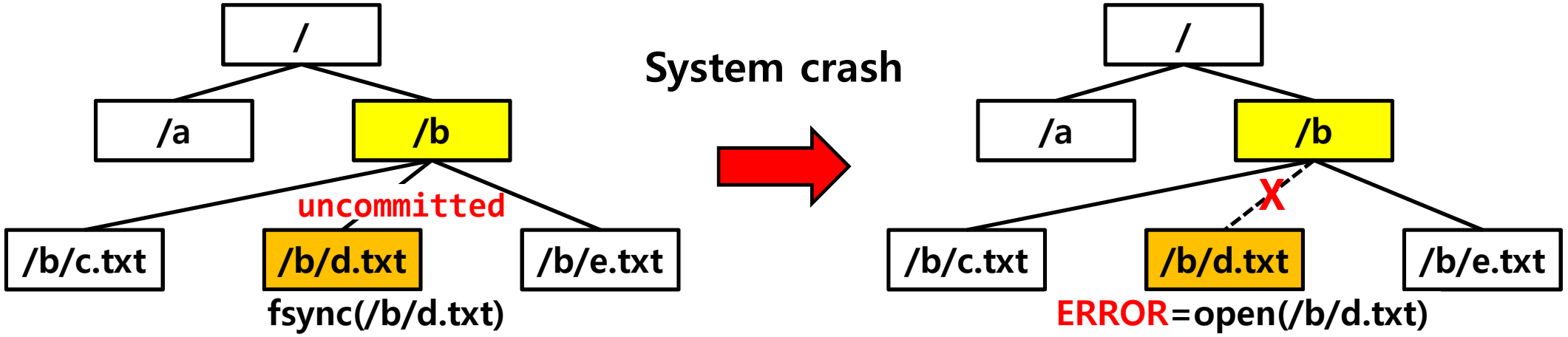
iJournal transaction

- **iJournal header block**
  - journal header
    - magic number
    - transaction ID (+ sub-txID)
  - inode number
  - inode structure
  - block tag
- **Journal**
  - metadata contents (ex. extent)
- **Commit block**



Inter-file dependencies

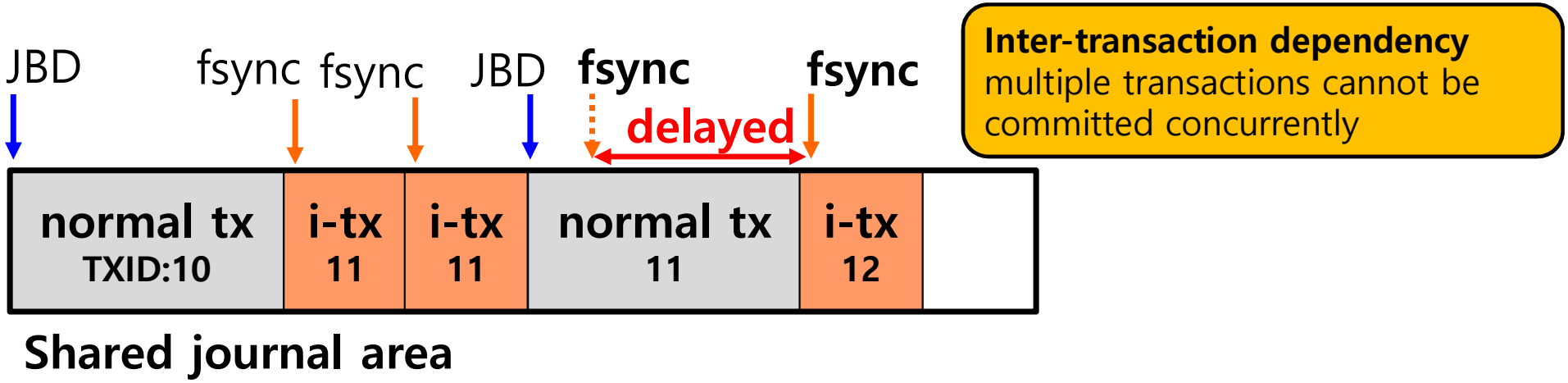
- **Parent DE**
  - If a file is fsynced but its parent directory entry is not committed ➔ unreachable
  - Log DE of `uncommitted` parent directory



- **Uncommitted hard link modification of fsynced file**

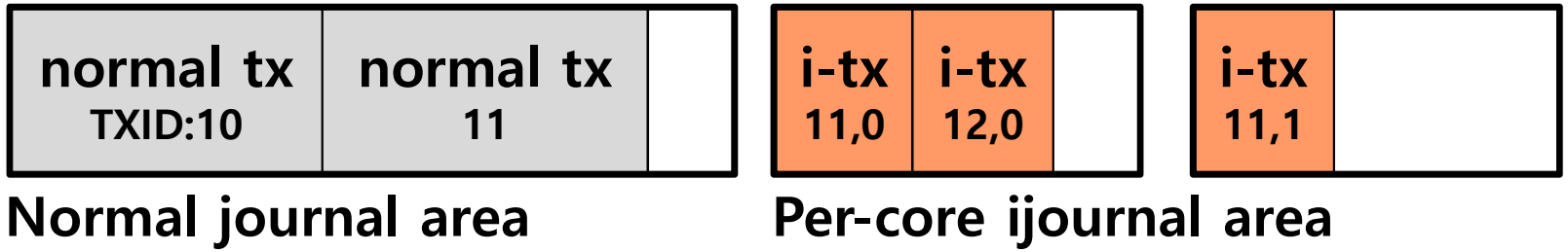
Managing iJournal area

- **Shared journal area**
  - No required additional space
  - iJournaling performed at the service routine of fsync()
  - However, ijournaling must wait until the block allocation of JBD is completed

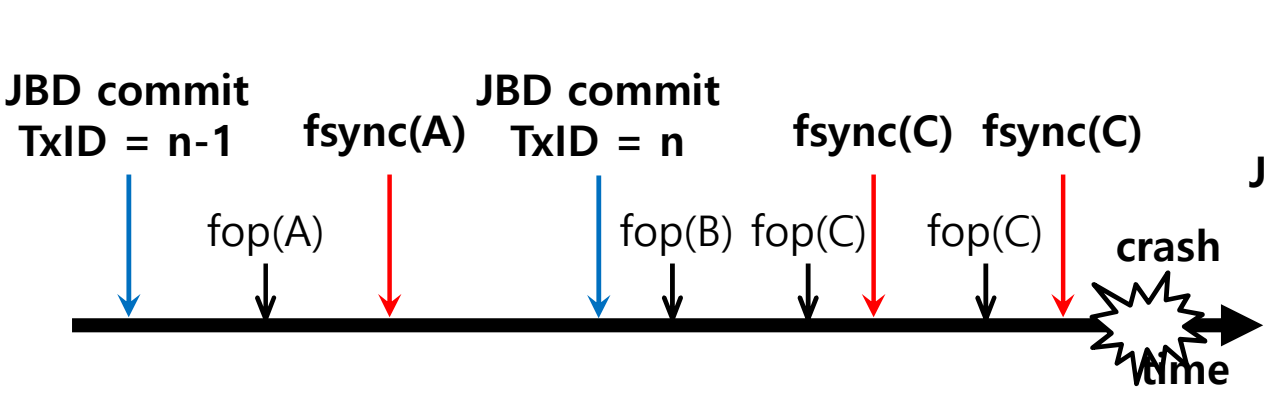


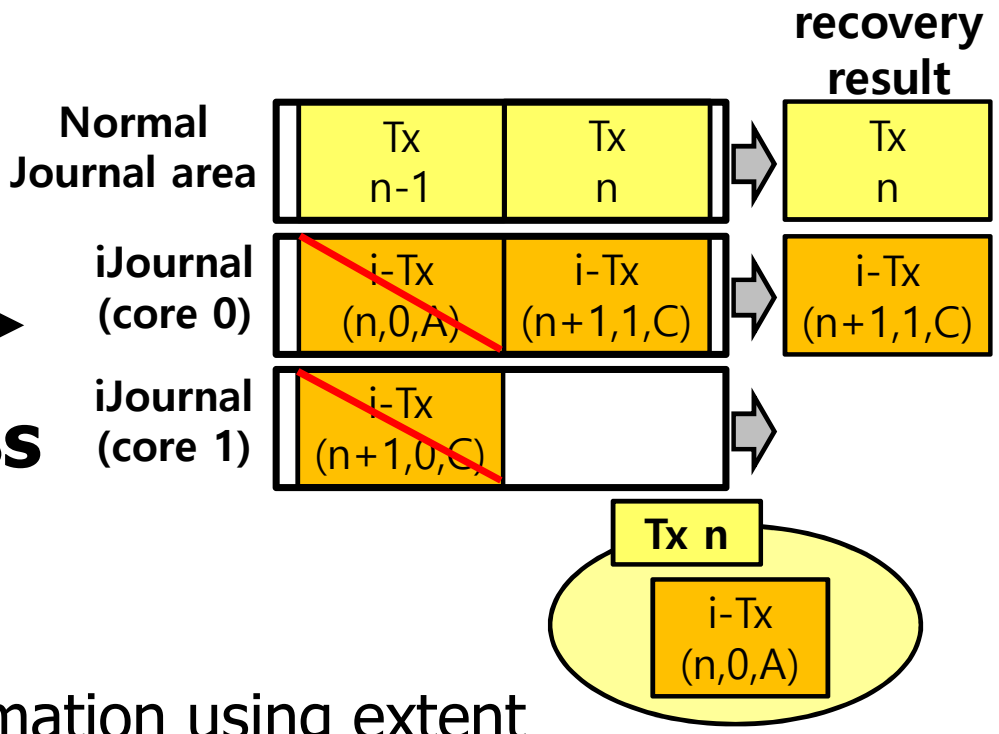
Managing iJournal area

- **Per-core iJournal area**
  - Required to additional space for per-core iJournal area
  - separating journal areas can improve the concurrency of journaling operations
  - sub-TxID is needed for sorting i-txs in order



iJournal recovery





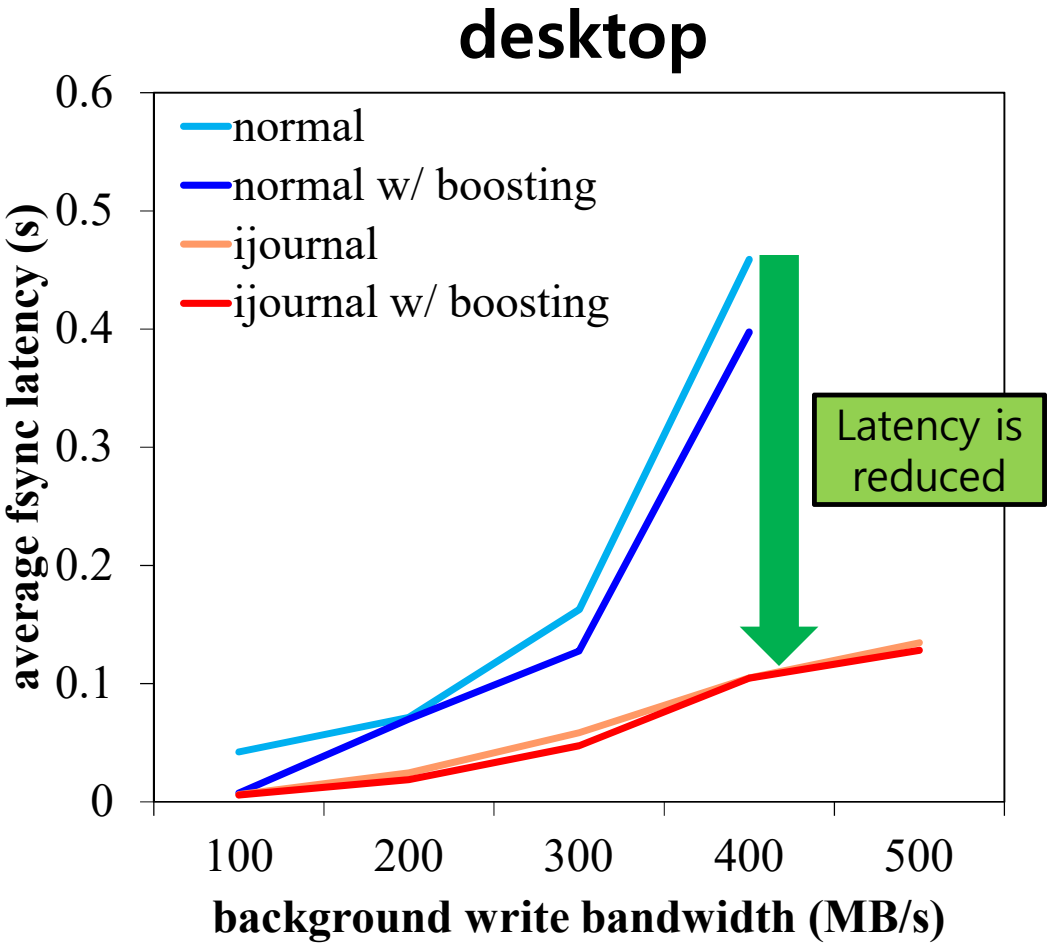
- **iJournaling recovery process**
  1. normal journal recovery
  2. file i-tx recovery
     - Recover block allocation information using extent
  3. dir i-tx recovery
     - Recover DE to make accessible for fsynced file

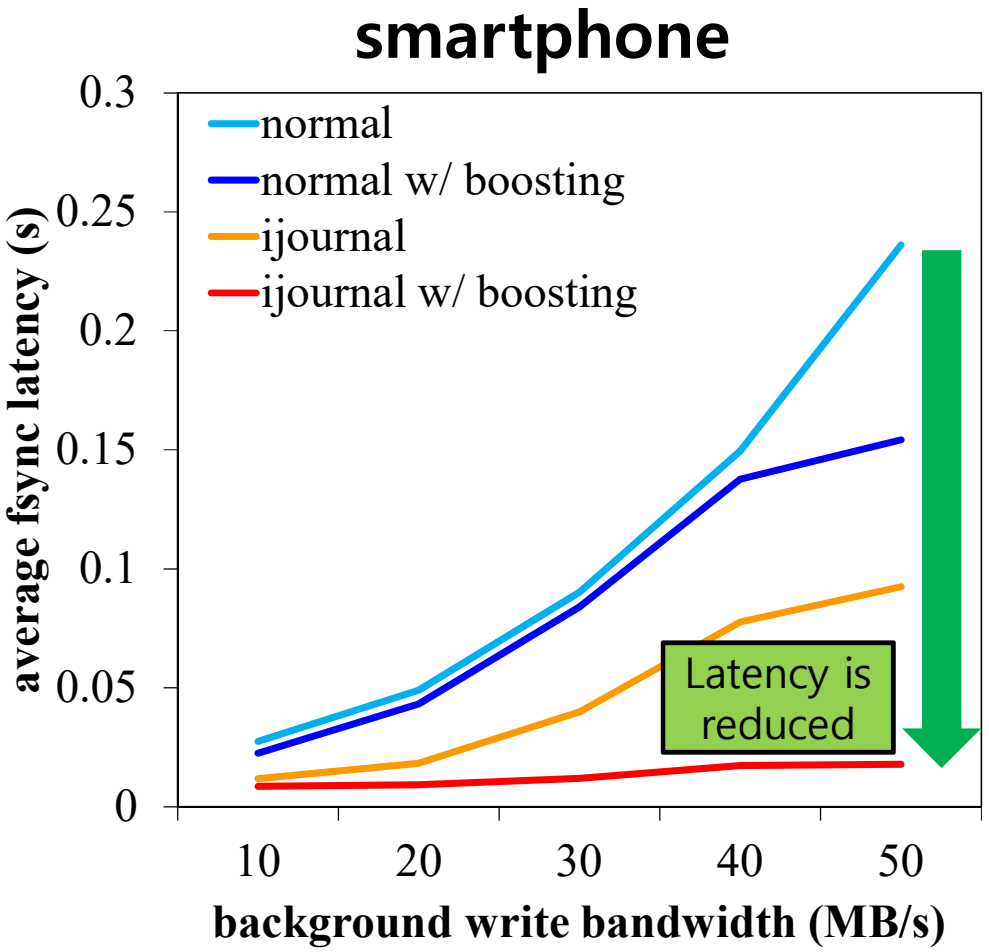
Implementation

- **iJournaling**
  - Based on ext4 file system (ordered mode)
  - iJournaling on fsync() system call applied
  - Delayed allocation enabled
  - boosting technique[FAST '15] optionally applied
- **Desktop**
  - Linux kernel 4.7.3
  - Samsung 850 Pro SSD
- **Smartphone**
  - Linux kernel version: 3.4.5
  - Android OS version: 4.2.2 (Jelly Bean)
  - eMMC 32 GB

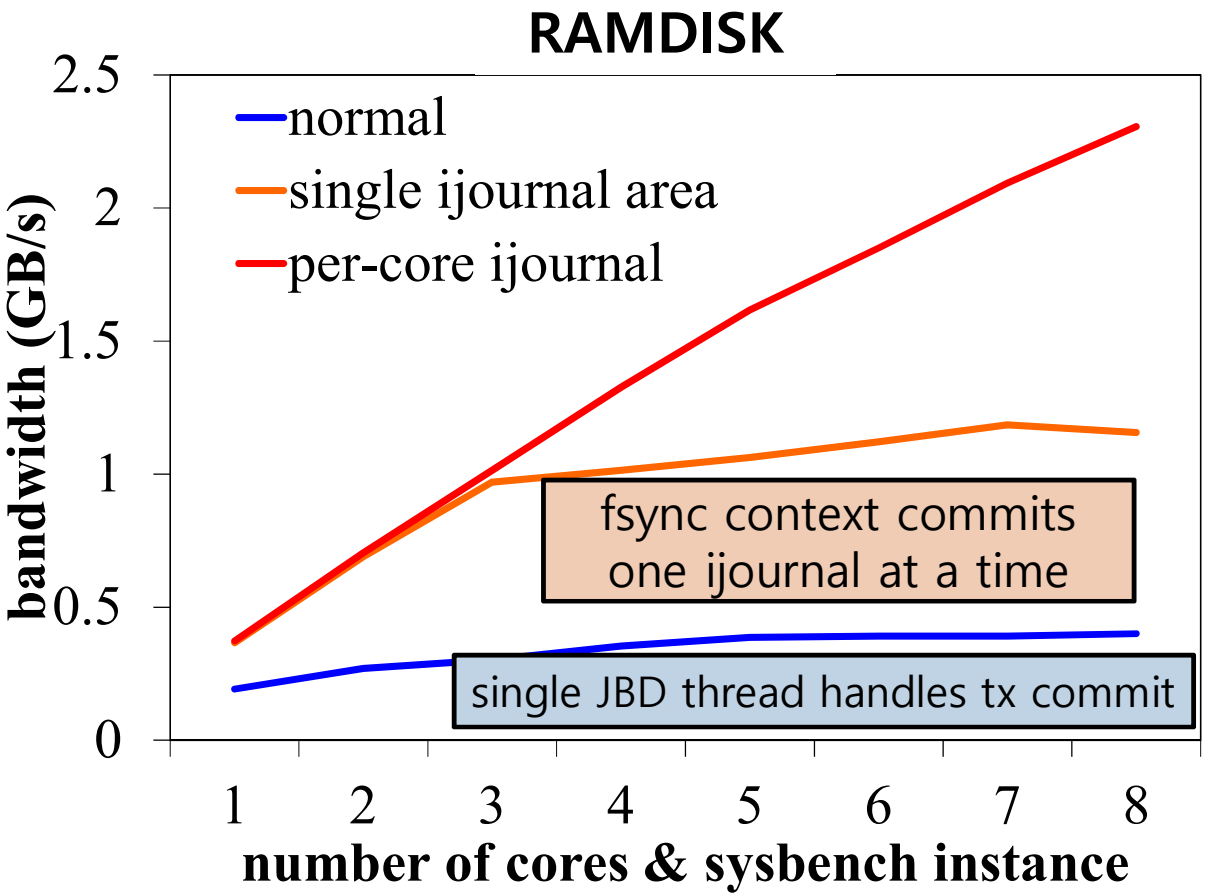
[1] Jeong et al. Boosting quasi-asynchronous I/O for better responsiveness in mobile devices. FAST '15

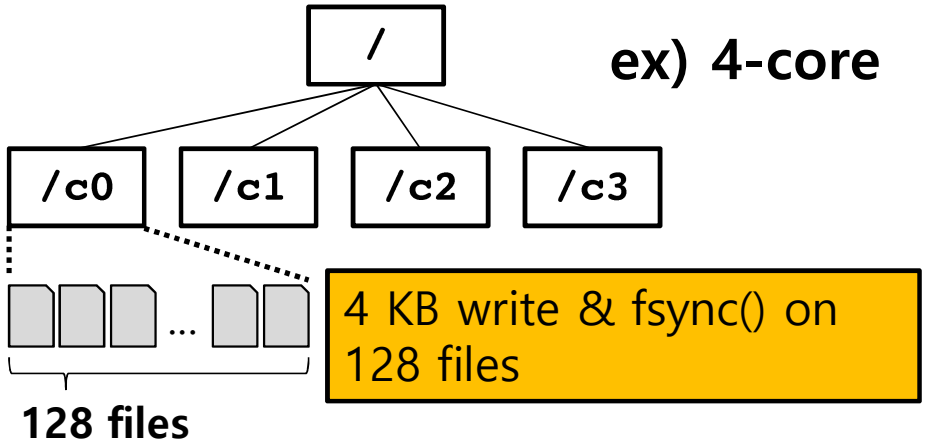
Fsync latency on various BG write traffic



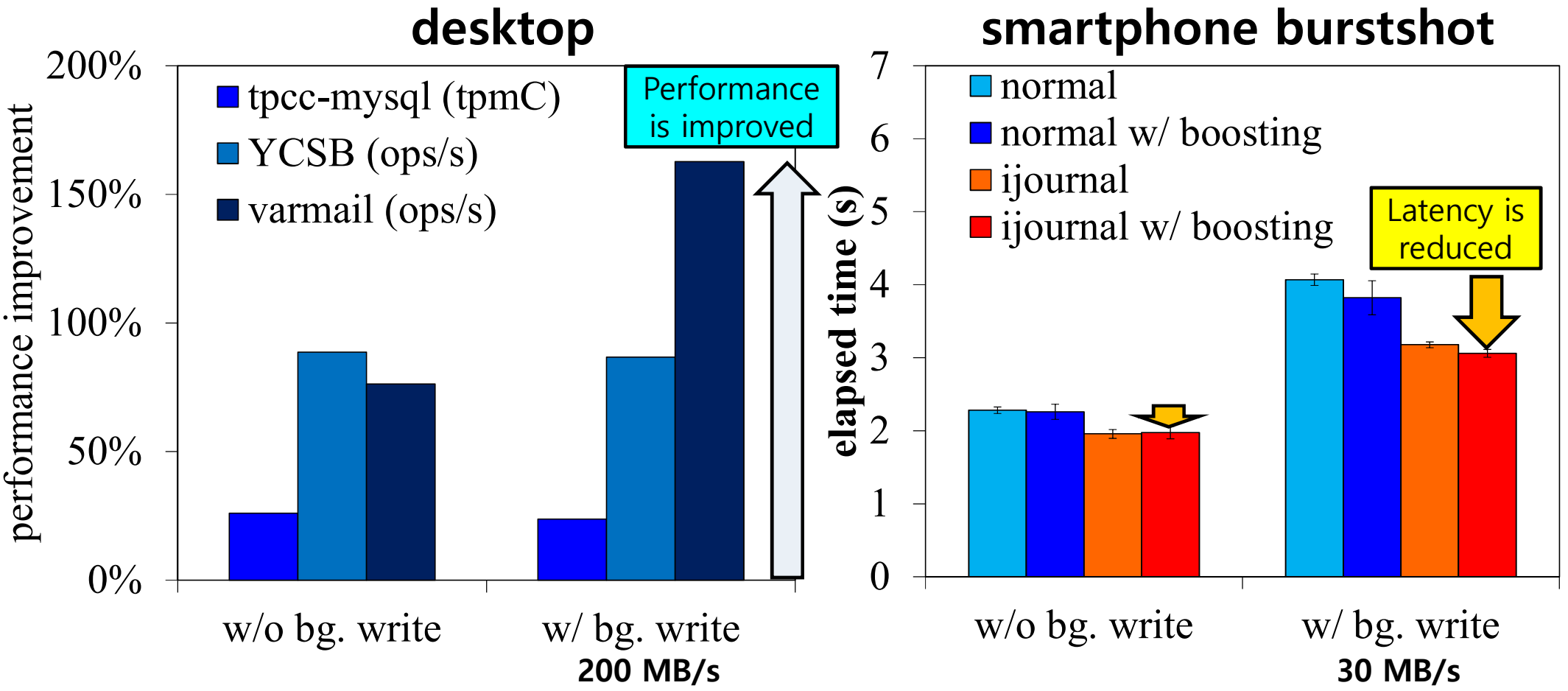


Multicore scalability on iJournaling





Real workload on iJournaling



Conclusion

- **fsync() system call can be delayed under the compound transaction scheme**
- **a hybrid journaling technique, called ijournaling**
  - journals only the related file-level transactions of an fsync call
  - recovers the file-system consistency through file-level journals upon a crash recovery
- **We implemented ijournaling and showed significant improvements to the fsync latencies**

Thank you!

Q&A