Looking for Ways to Improve the Performance of Ext4 Journaling

Hyeoncheol Lee
(cheol.lee at lge.com)

SW Platform Lab., Corporate R&D
LG Electronics, Inc.

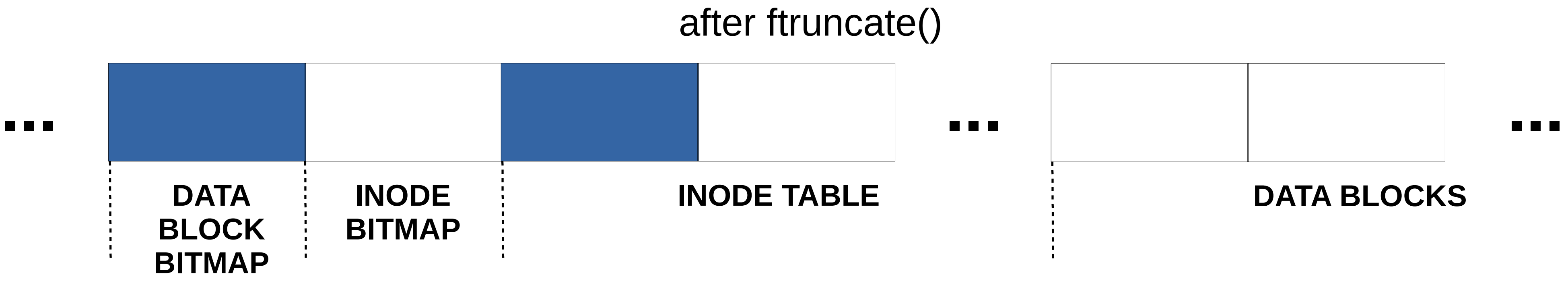
2015/10/23

Contents

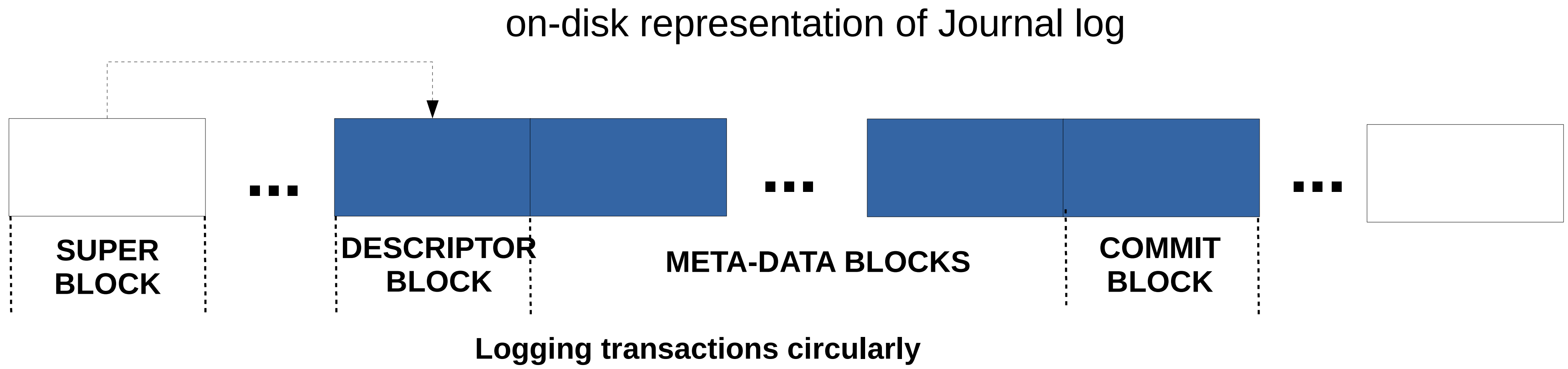
- Ext4 Journaling
- Major Patches submitted since kernel 3.11
- Sequential I/O Journaling
- File-Adaptive Journaling
- Experimental Result

Ext4 Journaling(1)

- Meta-data blocks modified by a file update operation should be written atomically



- Logging related blocks as compound transaction in Journal log, and apply this to file system


on-disk representation of Journal log
SUPER BLOCK
...
DESCRIPTOR BLOCK
META-DATA BLOCKS
...
COMMIT BLOCK
...
Logging transactions circularly

Ext4 Journaling(2)

- Consists of starting transaction, committing compound transaction, check-pointing compound transactions, and replaying Journal
- Starting transaction

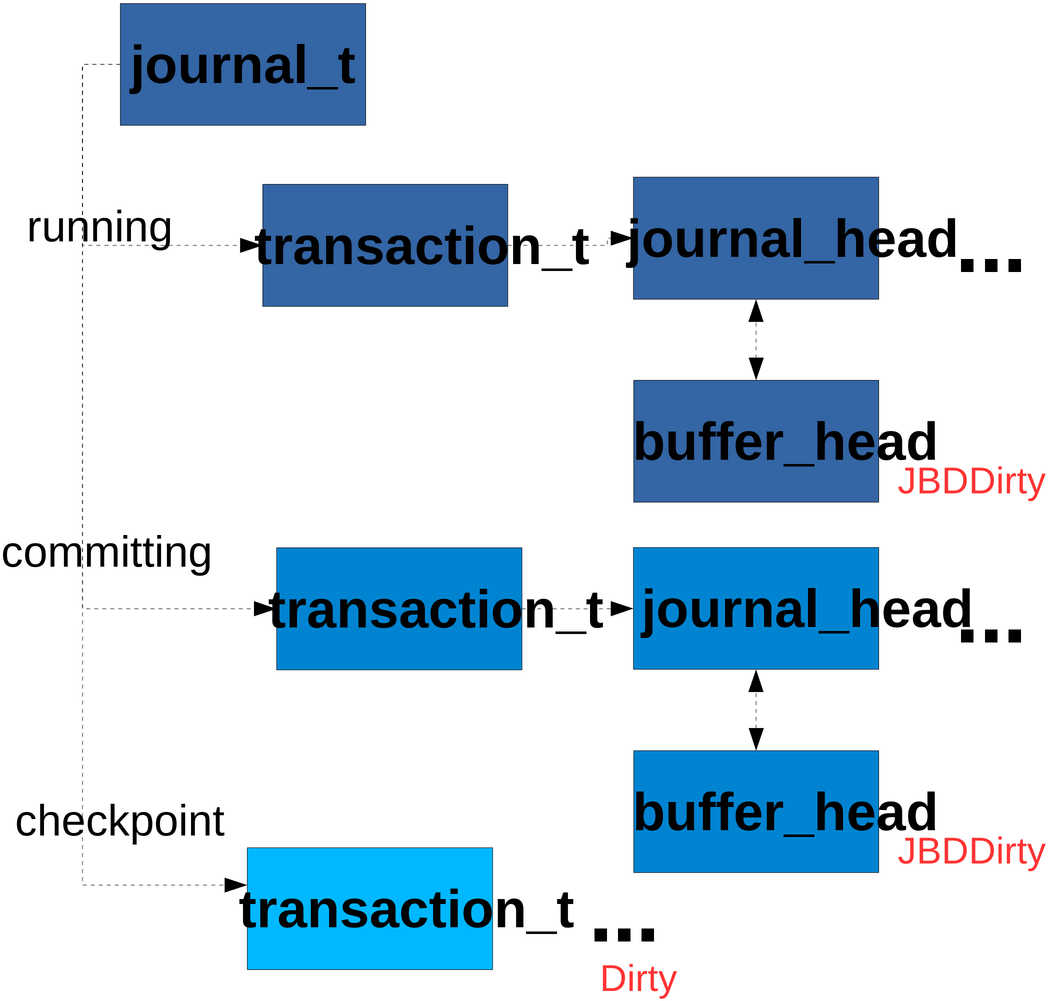
  ext4_start_transaction
  ? request committing transaction
  ? request check-pointing transactions
  ...
  size changed? ext4_mark_inode_dirty
  …
  ext4_stop_transaction

- Committing compound transaction
  - *by kjournald2 when*
    - *running transaction is full*
    - *time is expired*
    - *fsync(2)*

  1) change running to committing transaction

  2) request writing data pages

  3) request writing descriptor and meta-data blocks



Ext4 Journaling(3)

- Committing compound transaction

  4) wait for completion

  5) barrier

  6) request writing commit block

  7) barrier

  8) wait for completion

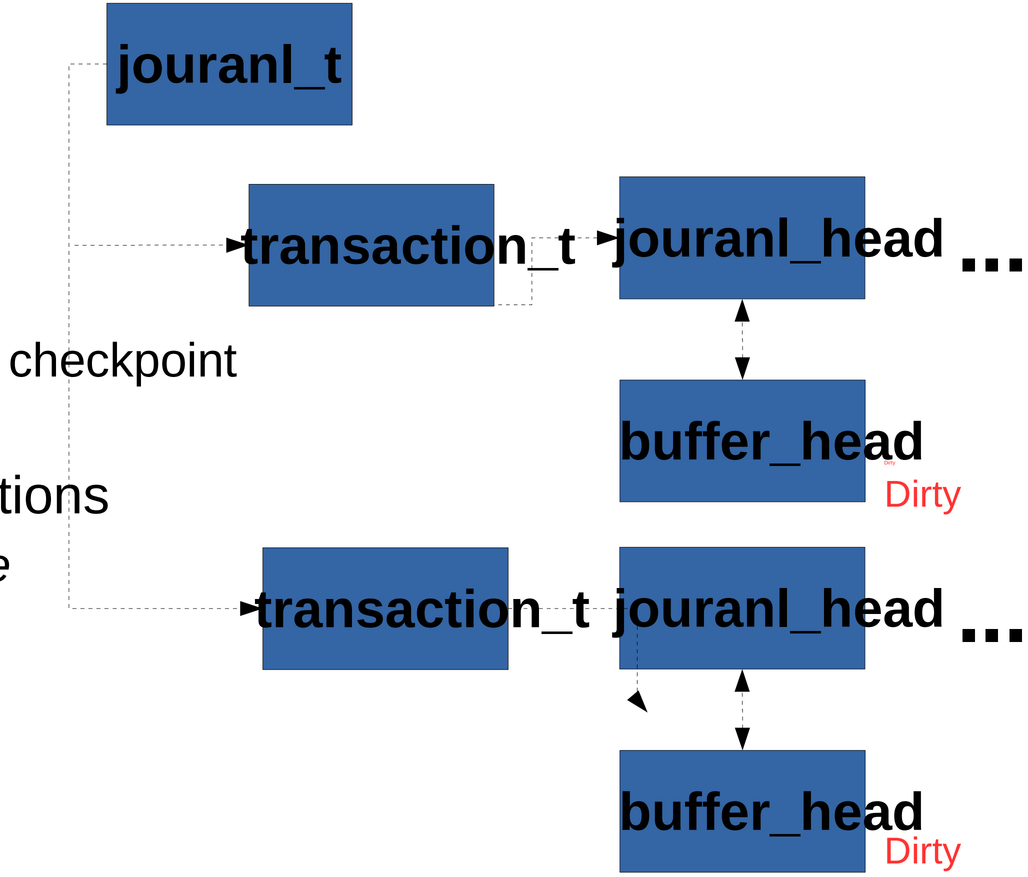
  9) dirty meta-data blocks and append these to checkpoint lists

  10) ? cleaning up Journal log

- Check-pointing compound transactions
  - *by write-back, meta-data blocks are written to file system*
  - *by starting transaction*

    1) Cleaning up Journal log

    2) Writing meta-data blocks in a transaction



jouranl_t

checkpoint

transaction_t

jouranl_head

...

buffer_head

Dirty

transaction_t

jouranl_head

...

buffer_head

Dirty

Major Patches submitted since kernel 3.11(1)

- Avoid pointless scanning of checkpoint lists
  - By Jan Kara, merged into 3.18
  - Scanning checkpoint lists for freeing memory consumed a lot of CPU cycles in fsync(2) heavy workload

Committing transaction

1) Scanning checkpoint list for free memory
Full scanning of checkpoint list →
Stop scanning if buffer_head can't be released

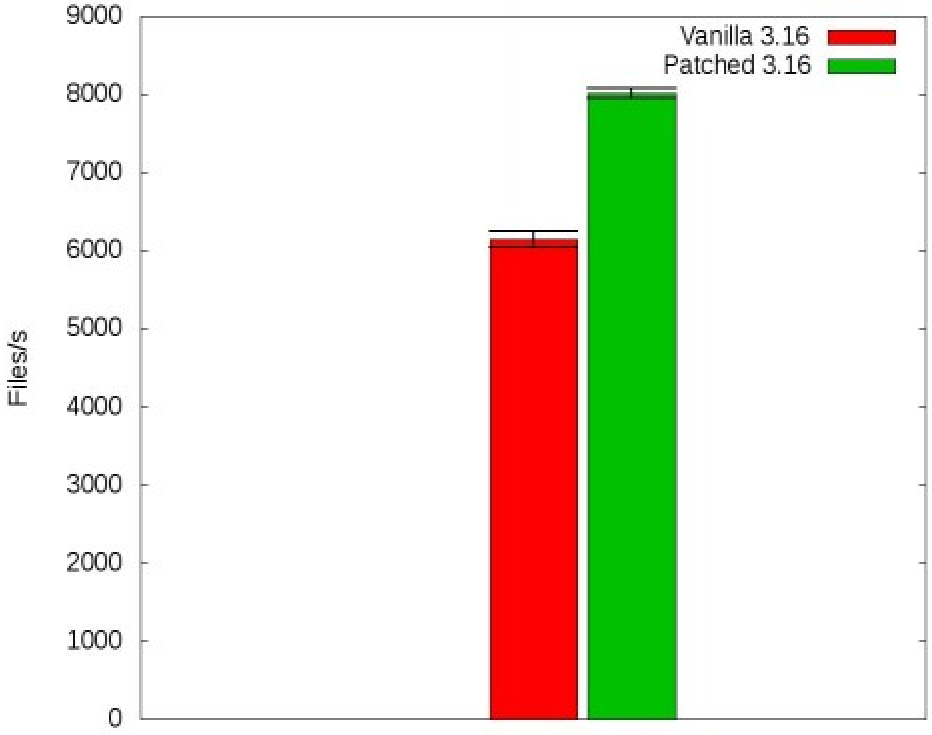
2) Changing running to committing transaction

3) Writing dirty pages

4) Writing descriptor and meta-data blocks

...

fs_mark creating 100000 files, 4 KB each, with fsync



From Ext4 Filesystem Scaling by Jan Kara

Major Patches submitted since kernel 3.11(2)

- Optimize jbd2_journal_force_commit
  - By Dmitry Monakhov, merged into 3.11
  - instead of waiting for some milliseconds unconditionally, if there are running or committing transaction, start committing transaction and wait for completion

```
ext4_sync_file() for data journal
    request writing dirty pages
    wait for completion
    data journal? ext4_force_commit
        jbd2_journal_force_commit
```

- Defer clearing of PageWriteback after extent conversion and remove i_mutex from ext_sync_file()
  - By Jan Kara, merged into 3.11
  - because ext4_flush_unwritten_io() call is removed, Holding i_mutex in ext_sync_file() is unnecessary.

```
 ext4_sync_file()
  -  mutex_lock(&inode->i_mutex)
 -   ext4_flush_unwritten_io
     ext4_force_commit
```

Major patches submitted since kernel 3.11(3)

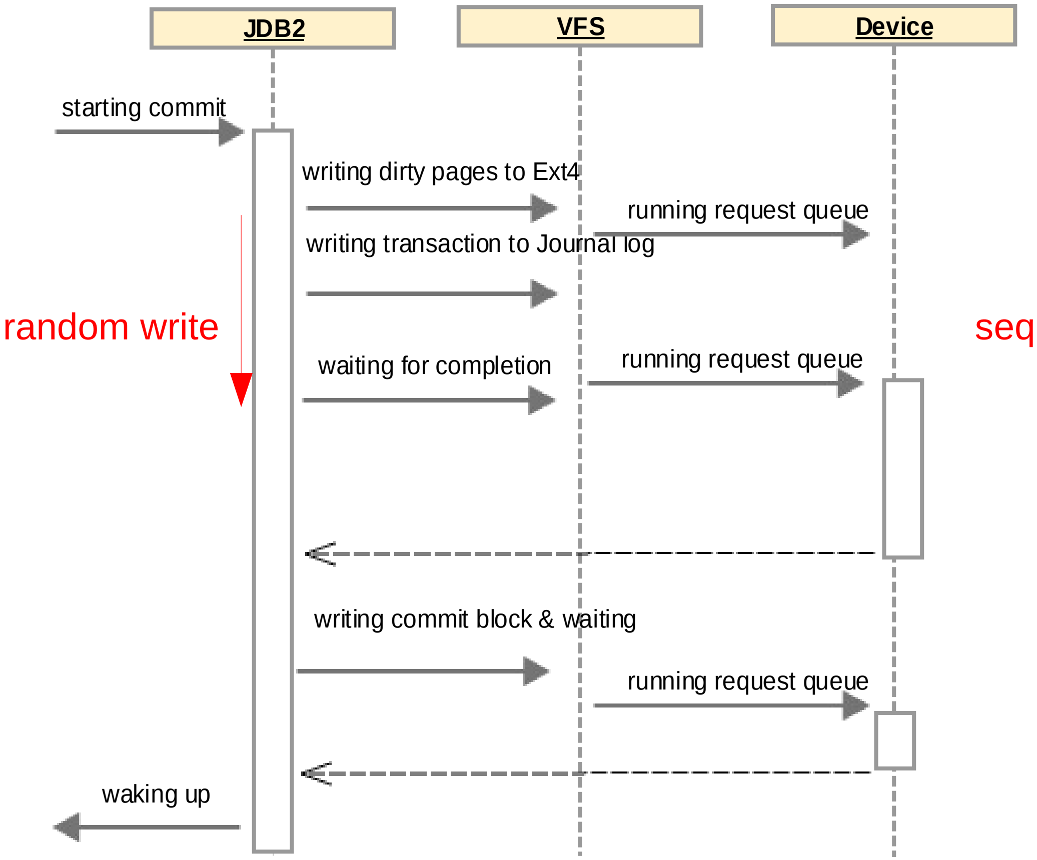
- Patches that reduce lock contention
  - Speed up jbd2_journal_dirty_metadata
    - Not hold BH_JournalHead bit lock
    - Just return if the given buffer_head is dirty
  - …

- A proposal for making ext4's journal more SMR(and flash) friendly
  - By Theodore Ts'o, http://thread.gmane.org/gmane.comp.file-systems.ext4/42069
  - Suppress check-pointing transactions and use meta-data blocks in Journal log → Relieve random writes of meta-data blocks
  - Truncate transactions only when there isn't no space in Journal log
    - Move transactions to head of Journal log or check-point transactions

Sequential I/O Journaling(1)

- Proposed in “Journaling of journal is almost free, FAST'14”
- for frequent fsync(2), data journaling is more suitable than ordered journaling

ORDERED JOURNAL



JDB2

VFS

Device

starting commit

writing dirty pages to Ext4

running request queue

writing transaction to Journal log

random write

waiting for completion

running request queue

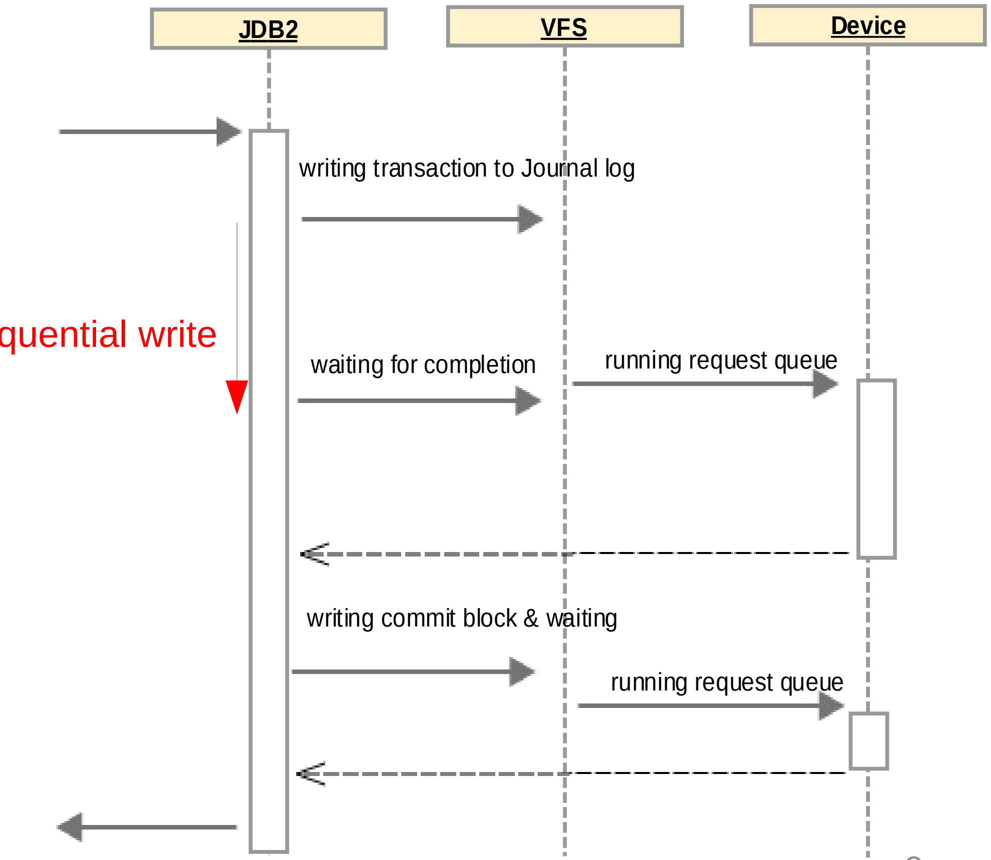
writing commit block & waiting

running request queue

waking up

DATA JOURNAL

sequential write



JDB2

VFS

Device

writing transaction to Journal log

waiting for completion

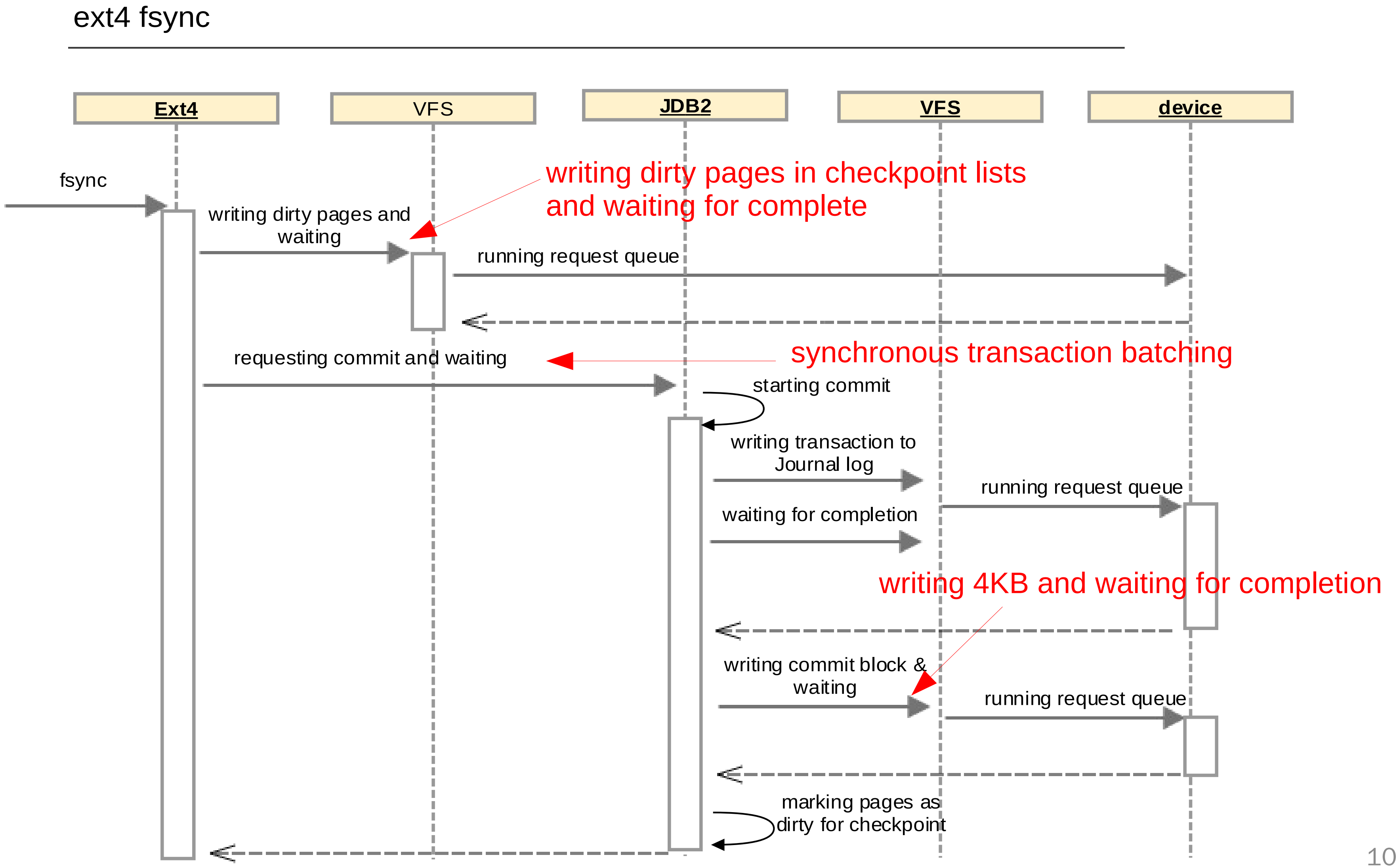
running request queue

writing commit block & waiting

running request queue

Sequential I/O Journaling(2)

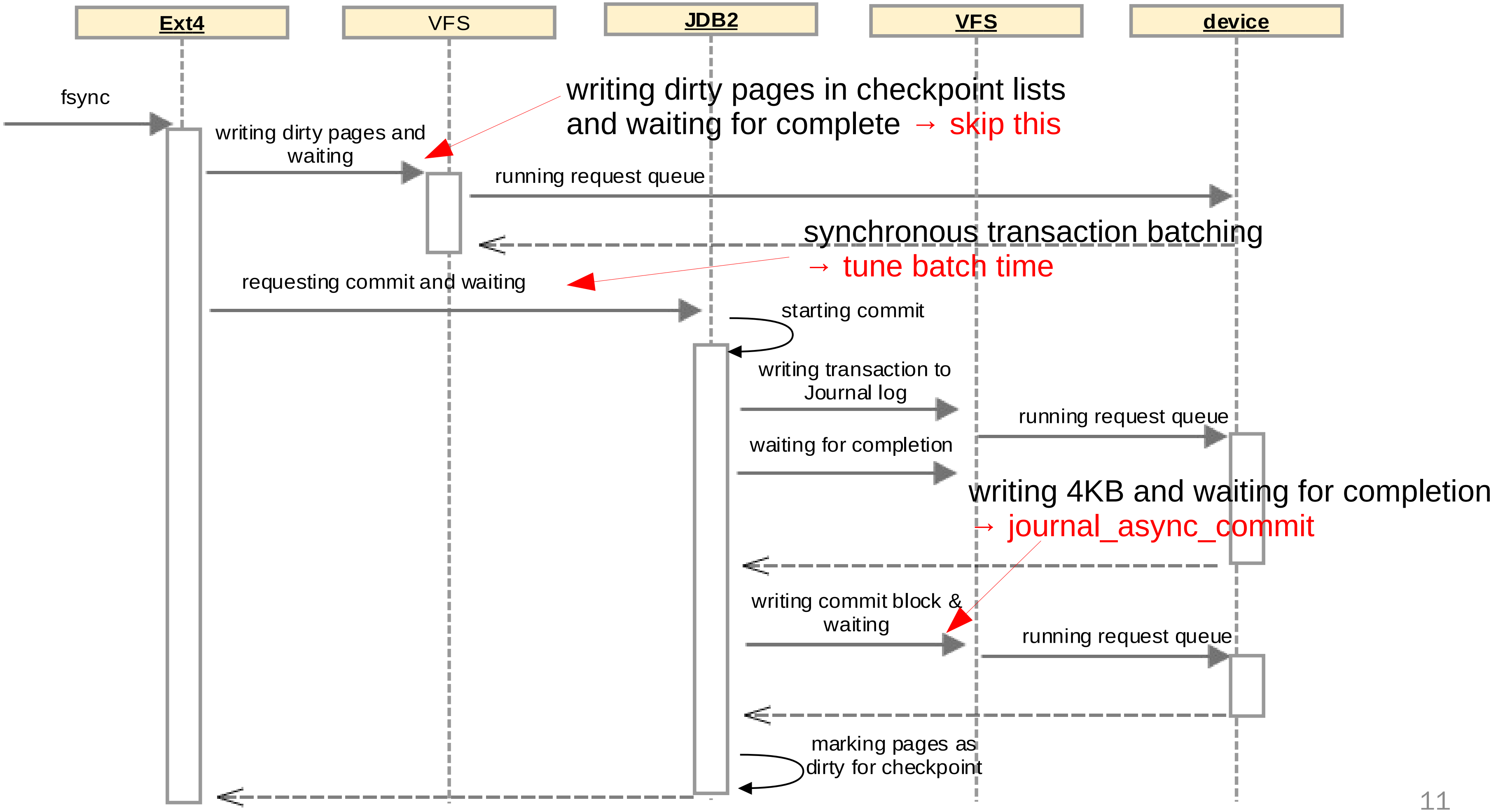
- But, problems in kernel 3.10:
  - Not want to mount Ext4 with data journaling, and ..


ext4 fsync
Ext4
VFS
JDB2
VFS
device
fsync
writing dirty pages and waiting
writing dirty pages in checkpoint lists and waiting for complete
running request queue
requesting commit and waiting
synchronous transaction batching
starting commit
writing transaction to Journal log
running request queue
waiting for completion
writing 4KB and waiting for completion
writing commit block & waiting
running request queue
marking pages as dirty for checkpoint

Sequential I/O Journaling(3)

- Solutions:
  - Not want to mount Ext4 with data journaling → File-Adaptive Journaling

ext4 fsync



Ext4
VFS
JDB2
VFS
device
fsync
writing dirty pages and waiting
running request queue
requesting commit and waiting
starting commit
writing transaction to Journal log
running request queue
waiting for completion
writing commit block & waiting
running request queue
marking pages as dirty for checkpoint

writing dirty pages in checkpoint lists and waiting for complete → skip this

synchronous transaction batching → tune batch time

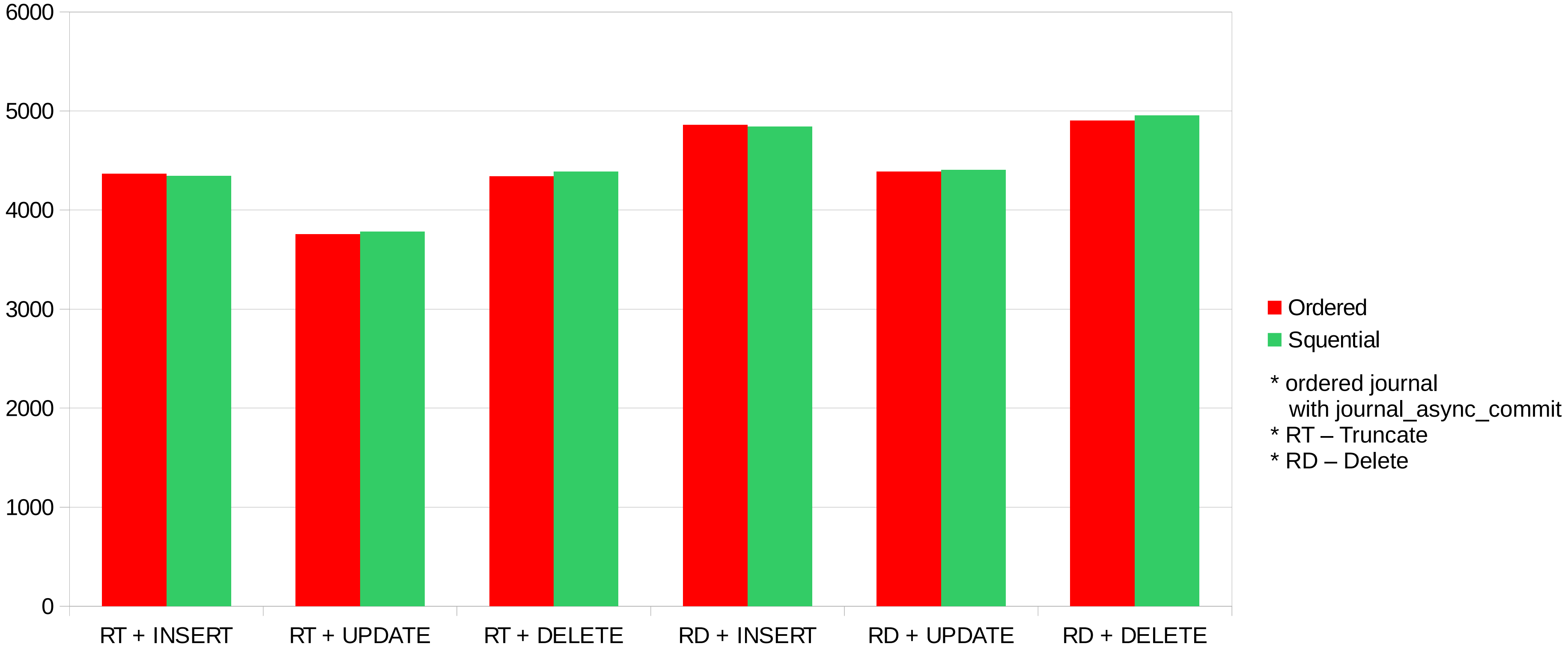
writing 4KB and waiting for completion → journal_async_commit

File Adaptive Journaling

- Changing to data journal for file is already implemented in Ext4
  - Ioctl(..., EXT4_IOC_SETFLAGS, EXT4_JOURNAL_DATA_FL)
  - CAP_SYS_RESOURCE capability is required
  - Be cautious. change from data journal to original journal needs to flush all of journal transactions!

Experimental Result

- Kernel 3.10, Android Lollipops
- Sqlite3, average response time in usec for each 10,000 * 1-operation transaction with 100+16 bytes record



* ordered journal
with journal_async_commit
* RT – Truncate
* RD – Delete

- We need to evaluate this in newer kernel
  - i_mutex in ext4_sync_file() is gone in 3.11
  - Synchronous transaction batch is gone in 3.11
  - Ordered journal with journal_async_commit is invalid in kernel 3.19

Q&A