Case study: ext3 FS

The ext3 file system

- Design goals
  - Add journaling capability to the ext2 FS
  - Backward and forward compatibility with ext2
    - Existing ext2 partitions can be mounted as ext3
  - Leverage the proven ext2 performance
  - Reuse most of the ext2 code base
  - Reuse ext2 tools, including e2fsck

The ext3 journal

Option1: Journal FS data structure updates

- Example:
  - **Start transaction**
  - Delete dir entry
  - Delete i-node
  - Release blocks 32, 17, 60
  - **End transaction**

Option2: Journal disk block updates

- Example:
  - **Start transaction**
  - Update block #n1 *(contains the dir entry)*
  - Update block #n2 *(i-node allocation bitmap)*
  - Update block #n3 *(data block allocation bitmap)*
  - **Add transaction**

Question: which approach is better?

The ext3 journal

Option1: Journal FS data structure updates

- ✔ Efficient use of journal space; hence faster journaling

- ✘ Individual updates are applied separately

- ✘ The journaling layer must understand FS semantics

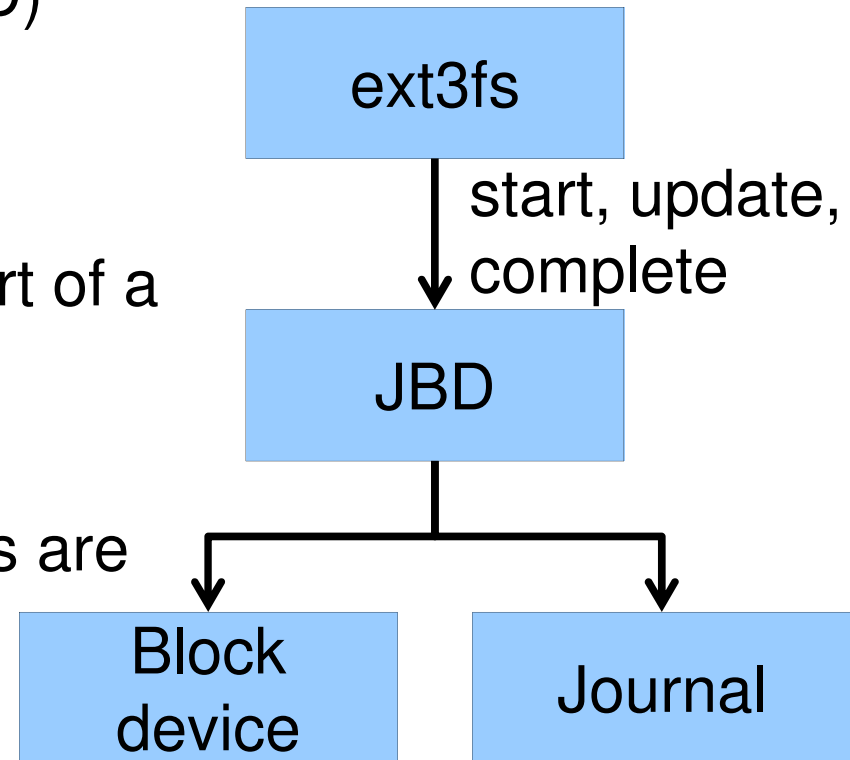**Option2: Journal disk block updates**

- ✘ Even a small update adds a whole block to the journal

- ✔ Multiple updates to the same block can be aggregated into a single update

- ✔ The journaling layer is FS-independent (easier to implement)

Ext3 implements Option 2
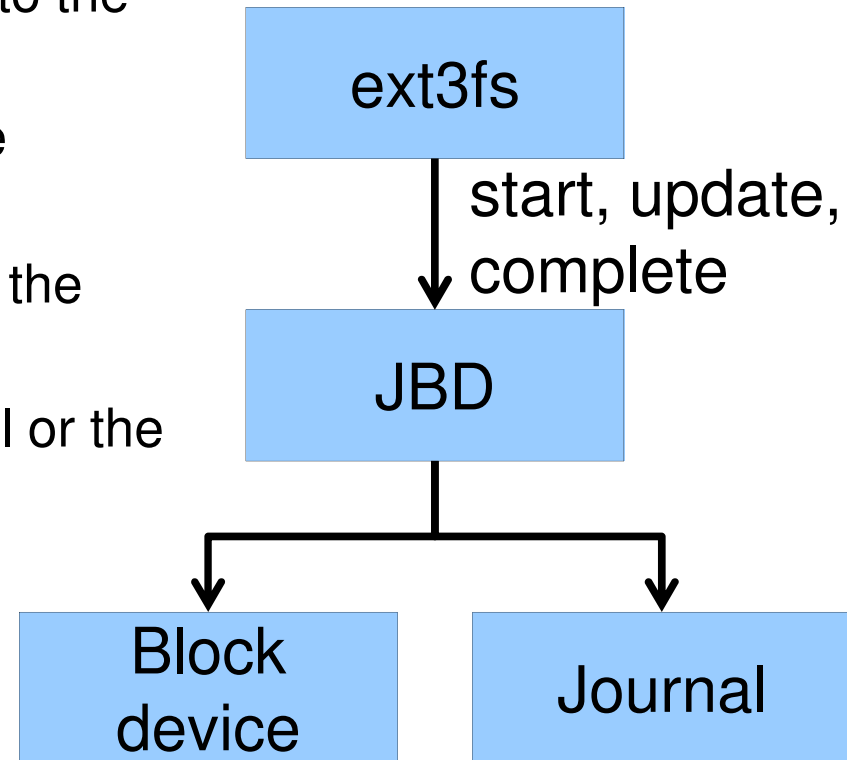
THE UNIVERSITY OF
NEW SOUTH WALES

Journaling Block Device (JBD)

- The ext3 journaling layer is called Journaling Block Device (JBD)

- JBD interface
  - Start a new transaction
  - Update a disk block as part of a transaction
  - Complete a transaction
    - Completed transactions are cached in RAM

```
┌─────────────┐
│   ext3fs    │
└─────────────┘
       │  start, update,
       ▼  complete
┌─────────────┐
│     JBD     │
└─────────────┘
    ┌──┴──┐
    ▼     ▼
┌────────┐ ┌─────────┐
│ Block  │ │ Journal │
│ device │ │         │
└────────┘ └─────────┘
```
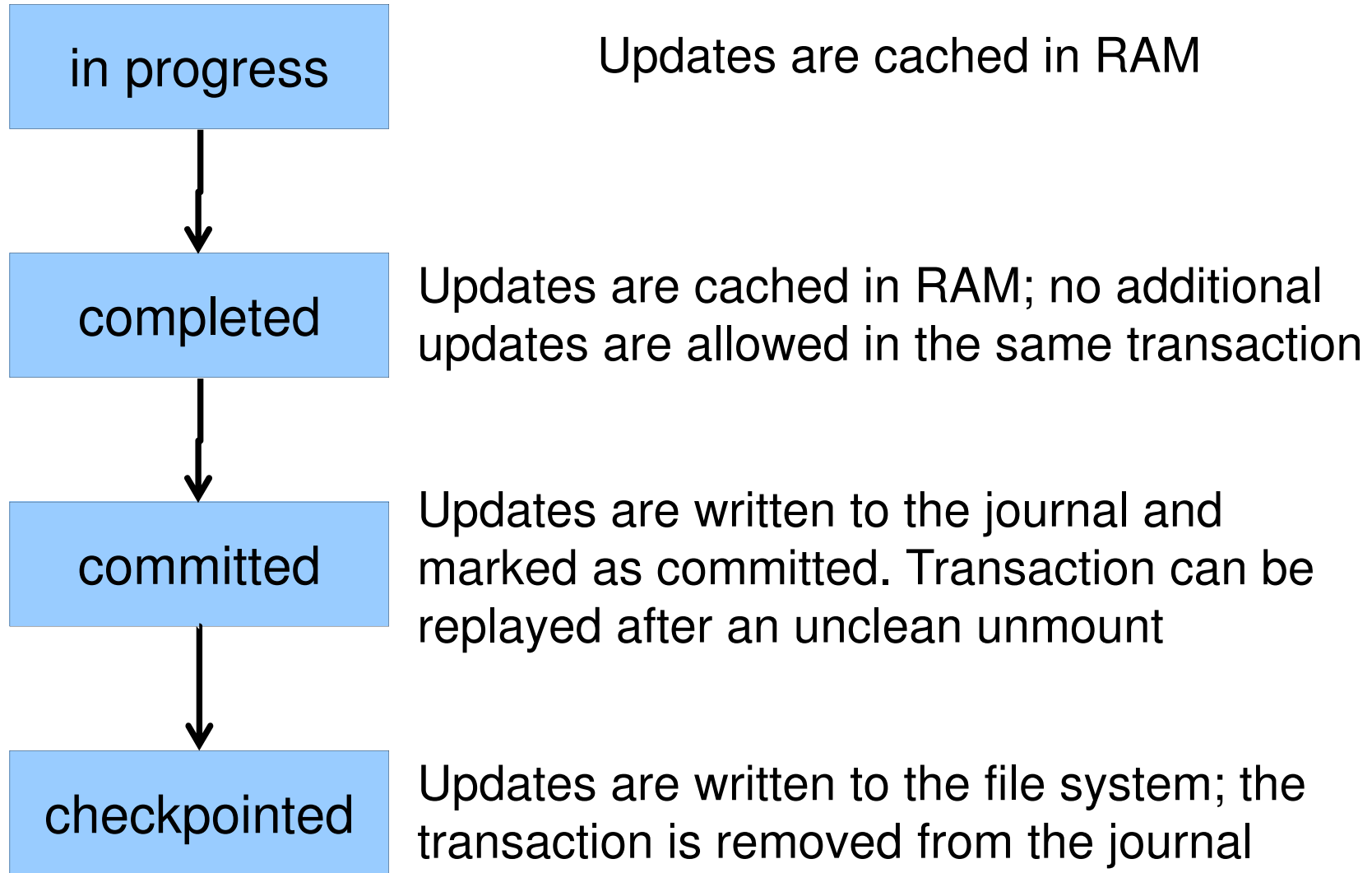
Journaling Block Device (JBD)

- JBD interface (continued)
  - Commit: write transaction data to the journal (persistent storage)
    - Multiple FS transactions are committed in one go
  - Checkpoint: flush the journal to the disk
    - Used when the journal is full or the FS is being unmounted

ext3fs

start, update, complete

JBD

Block device

Journal

Transaction lifecycle

| in progress | Updates are cached in RAM |

↓

| completed | Updates are cached in RAM; no additional updates are allowed in the same transaction |

↓

| committed | Updates are written to the journal and marked as committed. Transaction can be replayed after an unclean unmount |

↓

| checkpointed | Updates are written to the file system; the transaction is removed from the journal |

Journaling modes

- Ext3 supports two journaling modes
  - Metadata+data
    - Enforces atomicity of all FS operations
  - Metadata journaling
    - Metadata is journaled
    - Data blocks are written directly to the disk
    - Improves performance
    - Enforces file system integrity
    - Does not enforce atomicity of write's

THE UNIVERSITY OF
NEW SOUTH WALES

JBD

- JBD can keep the journal on a block device or in a file
  - Enables compatibility with ext2 (the journal is just a normal file)
- JBD is independent of ext3-specific data structures
  - Separation of concerns
    - The FS maintains on-disk data and metadata
    - JBD takes care of journaling
  - Code reuse
    - JBD can be used by any other FS that requires journaling