CJFS : Concurrent Journaling for Better Scalability

Joontaek Oh*, Seung Won Yoo*, Hojin Nam*, Changwoo Min†, Youjip Won*

*KAIST  †Virginea Tech
Outline

✓ Background and Motivation

✓ Design

➢ Dual Thread Journaling
➢ Multi-Version Shadow Paging
➢ Opportunistic Coalescing
➢ Compound Flush

✓ Evaluation

✓ Conclusion
Background and Motivation
Hardware and Software @2023+

**Hardware:**
- 2 cores
  - Intel Core 2 Duo @2006

**Software:**
- 877 IOPS
  - Western Digital Caviar SE16 @2006
Hardware and Software @2023+

**Hardware:**

- 2 cores
  - Intel Core 2 Duo
  - @2006

- 877 IOPS
  - Western Digital Caviar SE16
  - @2006

**Software:**

- 128 cores
  - AMD EPYC 7763
  - @2021

- 700K IOPS
  - Seagate FireCuda 530
  - @2021
Hardware and Software @2023+

Hardware:

2 cores
Intel Core 2 Duo
@2006

128 cores
AMD EPYC 7763
@2021

877 IOPS
Western Digital Caviar SE16
@2006

700K IOPS
Seagate FireCuda 530
@2021

Software:

64X

798X
Hardware and Software @2023+

Hardware:

2 cores
Intel Core 2 Duo @2006

64X

128 cores
AMD EPYC 7763 @2021

798X

877 IOPS
Western Digital Caviar SE16 @2006

700K IOPS
Seagate FireCuda 530 @2021

Software:

1,000+ VMs

VM VM VM VM VM

VM VM VM VM VM

VM VM VM VM VM

... VM...

ino

dentry
dentry
dentry

Journal

Block
dentry
dentry

JD JD JD

...
Hardware and Software @2023+

Hardware:

- **2 cores**
  - Intel Core 2 Duo @2006
- **128 cores**
  - AMD EPYC 7763 @2021

- **877 IOPS**
  - Western Digital Caviar SE16 @2006
- **700K IOPS**
  - Seagate FireCuda 530 @2021

Software:

- **1,000+ VMs**

- **64X**
- **798X**
Hardware and Software @2023+

**Hardware:**

- **Intel Core 2 Duo** @2006
- **AMD EPYC 7763** @2021

**Software:**

- **1,000+ VMs**
- 128 cores
- **Block**
- **dentry**
- **Journal**
- **VMs**
- **Western Digital Caviar SE16** @2006
- **Seagate FireCuda 530** @2021

**Performance:**

- **798X**
- **64X**
Serial Commit in EXT4 Journaling

- All steps of journal commit are serialized
  - Lock-Up: Lock the running transaction and waiting for remained file operation
Serial Commit in EXT4 Journaling

- All steps of journal commit are serialized
  - Lock-Up: Lock the running transaction and waiting for remained file operation
  - Prepare DMA: Create and dispatch write command for the transaction
Serial Commit in EXT4 Journaling

- All steps of journal commit are serialized
  - Lock-Up: Lock the running transaction and waiting for remained file operation
  - Prepare DMA: Create and dispatch write command for the transaction
  - DMA Transfer: Waiting for the completion of DMA Transfer of the transaction
Serial Commit in EXT4 Journaling

- All steps of journal commit are serialized
  - Lock-Up: Lock the running transaction and waiting for remained file operation
  - Prepare DMA: Create and dispatch write command for the transaction
  - DMA Transfer: Waiting for the completion of DMA Transfer of the transaction
  - Flush: Flush transferred data
Serial Commit in EXT4 Journaling

- All steps of journal commit are serialized
  - Lock-Up: Lock the running transaction and waiting for remained file operation
  - Prepare DMA: Create and dispatch write command for the transaction
  - DMA Transfer: Waiting for the completion of DMA Transfer of the transaction
  - Flush: Flush transferred data
Serial Commit in EXT4 Journaling

- All steps of journal commit are serialized
  - Lock-Up: Lock the running transaction and waiting for remained file operation
  - Prepare DMA: Create and dispatch write command for the transaction
  - DMA Transfer: Waiting for the completion of DMA Transfer of the transaction
  - Flush: Flush transferred data

![Diagram showing the serialization process](image-url)
Serial Commit in EXT4 Journaling

- All steps of journal commit are serialized
  - Lock-Up: Lock the running transaction and waiting for remained file operation
  - Prepare DMA: Create and dispatch write command for the transaction
  - DMA Transfer: Waiting for the completion of DMA Transfer of the transaction
  - Flush: Flush transferred data

- Delayed Commit:

\[ \text{Tx}_1 \quad \text{Lock Up, Prepare DMA, DMA Transfer, Flush} \]

\[ \text{Tx}_2 \quad \text{Delayed Lock Up, Prepare DMA, DMA Transfer, Flush} \]

\[ \text{Tx}_3 \quad \text{Delayed Lock Up, Prepare DMA} \]
Existing Works
Existing Works

Multiple journal regions:

IceFS (OSDI '14), SpanFS (ATC '15), Z-journal (ATC'21)
Existing Works

Multiple journal regions:

IceFS (OSDI '14), SpanFS (ATC '15), Z-journal (ATC'21)

Still serial transaction commit in each journal region
Existing Works

Multiple journal regions:
IceFS (OSDI '14), SpanFS (ATC '15), Z-journal (ATC'21)
Still serial transaction commit in each journal region

Per-core running transaction:
ScaleFS (SOSP '17), MQFS (SOSP '21)
Existing Works

Multiple journal regions:

IceFS (OSDI '14), SpanFS (ATC '15), Z-journal (ATC'21)

Still serial transaction commit in each journal region

Per-core running transaction:

ScaleFS (SOSP '17), MQFS (SOSP '21)

Conflict between multiple transactions and Still serial commit
Existing Works

Multiple journal regions:
IceFS (OSDI '14), SpanFS (ATC '15), Z-journal (ATC'21)
Still serial transaction commit in each journal region

Per-core running transaction:
ScaleFS (SOSP '17), MQFS (SOSP '21)
Conflict between multiple transactions and Still serial commit

Parallel journal commit: BarrierFS (FAST '18)
Existing Works

Multiple journal regions:
IceFS (OSDI '14), SpanFS (ATC '15), Z-journal (ATC'21)
Still serial transaction commit in each journal region

Per-core running transaction:
ScaleFS (SOSP '17), MQFS (SOSP '21)
Conflict between multiple transactions and Still serial commit

Parallel journal commit: BarrierFS (FAST '18)
Main reasons

- Transaction conflict
- Transaction Lock-Up
Transaction conflict

The situation that a file operation modifies a page which is being committed

User

JBD

\(Tx_1\)

\(Tx_2\)
Transaction conflict

The situation that a file operation modifies a page which is being committed

User

create()
Start

JBD

Committing Tx₁
Start

Tx₁

Tx₂

Time
Transaction conflict

The situation that a file operation modifies a page which is being committed

create()
Start

User

JBD

Committing Tx
Start

Committing

Time
Transaction conflict

The situation that a file operation modifies a page which is being committed
Transaction conflict

The situation that a file operation modifies a page which is being committed

User

create()
Start

Modify

JBD

Committed Tx₁
Start

Committed

Tx₁

Tx₂

Time
Transaction conflict

The situation that a file operation modifies a page which is being committed.

User

JBD

Tx₁

Tx₂

create() Start

Modify

Blocked

Committing Tx₁ Start

Committing

Committing Tx₁ End

Time
Transaction conflict

The situation that a file operation modifies a page which is being committed

- User
  - create(): Start
  - Committing Tx
    - Tx
      - Committing Tx
        - Tx
          - Start
      - End
    - Blocked
  - Tx
    - Start
  - Committing Tx
  - Running Tx

- JBD
  - Committing Tx
    - Modify
      - Blocked
  - Committing Tx
    - End

Time
### Transaction Lock-Up

The locked period for isolating the running transaction from file operations

<table>
<thead>
<tr>
<th>Time</th>
<th>OP1</th>
<th>OP2</th>
<th>OP3</th>
<th>OP4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>TX1</th>
<th>TX2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Transaction Lock-Up

the locked period for isolating the running transaction from file operations

OP₁

OP₂

OP₃

OP₄

TX₁

TX₂

Coalesced to TX₁

Coalesced to TX₁

Coalesced to TX₁

Running

Time
Transaction Lock-Up

the locked period for isolating the running transaction from file operations

OP₁

Coalesced to Tx₁

OP₂

Coalesced to Tx₁

OP₃

Coalesced to Tx₁

OP₄

---

TX₁ Running

TX₂

---

Time
Transaction Lock-Up

the locked period for isolating the running transaction from file operations

<table>
<thead>
<tr>
<th>OP₁</th>
<th>Coalesced to Tx₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP₂</td>
<td>Coalesced to Tx₁</td>
</tr>
<tr>
<td>OP₃</td>
<td>Coalesced to Tx₁</td>
</tr>
<tr>
<td>OP₄</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tx₁</th>
<th>Running</th>
<th>Locked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx₂</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Transaction Lock-Up

the locked period for isolating the running transaction from file operations

OP₁  Coalesced to Tx₁

OP₂  Coalesced to Tx₁

OP₃  Coalesced to Tx₁

OP₄

Tx₁  Running  Locked

Tx₂

Time
Transaction Lock-Up

the locked period for isolating the running transaction from file operations

OP_1 \hspace{2cm} \text{Coalesced to Tx}_1

OP_2 \hspace{2cm} \text{Coalesced to Tx}_1

OP_3 \hspace{2cm} \text{Coalesced to Tx}_1

OP_4

Tx_1\hspace{2cm} \text{Running} \hspace{2cm} \text{Locked}

Tx_2
Transaction Lock-Up

the locked period for isolating the running transaction from file operations

OP\textsubscript{1}  \hspace{2cm} \textcolor{orange}{Coalesced to Tx\textsubscript{1}}

OP\textsubscript{2}  \hspace{2cm} \textcolor{orange}{Coalesced to Tx\textsubscript{1}}

OP\textsubscript{3}  \hspace{2cm} \textcolor{orange}{Coalesced to Tx\textsubscript{1}}

OP\textsubscript{4}

Tx\textsubscript{1}  \hspace{2cm} Running \hspace{1cm} Locked \hspace{1cm} Committing Tx

Tx\textsubscript{2}  \hspace{2cm} Running

Time
Transaction Lock-Up

the locked period for isolating the running transaction from file operations

OP₁  Coalesced to Tx₁
OP₂  Coalesced to Tx₁
OP₃  Coalesced to Tx₁
OP₄  Coalesced to Tx₂

Tx₁  Running  Locked  Committing Tx
Tx₂  Running
Design:

Concurrent Journaling Filesystem (CJFS)
Design Goals

EXT4:

\[Tx_1 \rightarrow Tx_2 \rightarrow Tx_3\]

Concurrent Journaling Filesystem (CJFS):

\[Tx_1 \rightarrow Tx_2 \rightarrow Tx_3\]
Design Goals

EXT4:

Concurrent Journaling Filesystem (CJFS):

Dual Thread Journaling

- Dispatch
- Transfer and Flush
- JBD
- Commit
- Flush

TX1
TX2
TX3

TX1
TX2
TX3
Design Goals

EXT4:

Dual Thread Journaling
- Dispatch
- Transfer and Flush

JBD: Tx1 Tx2 Tx3

Commit: 1 2 3
Flush: 1 2 3

Multi-Version Shadow Paging

Commit: 1 2 3
Flush: 1 2 3

Concurrent Journaling Filesystem (CJFS):

Wait and Move
Non-wait Versioning
Design Goals

EXT4:

- Concurrent Journaling Filesystem (CJFS):

Dual Thread Journaling

- Dispatch
- Transfer and Flush

JBD

- Commit

Flush

Multi-Version Shadow Paging

- Dispatch
- Transfer and Flush

Opportunistic Coalescing

Commit

- Running
- Locked
- Committing

Wait and Move

Non-wait Versioning

Time

Page ... Page ... Page ...
Design Goals

EXT4: Dual Thread Journaling
- Dispatch: Transfer and Flush
- JBD: $Tx_1$, $Tx_2$, $Tx_3$
- Commit: 1, 2, 3
- Flush: 1, 2, 3

Multi-Version Shadow Paging
- Page: $Tx_1$, $Tx_2$, $Tx_3$
- Wait and Move

Opportunistic Coalescing
- Commit: Running, Locked, Committing
- Time

Compound Flush
- Commit: 1, 2, 3, 4
- Flush: 1, 2, 3, 4
- cache_barrier

Concurrent Journaling Filesystem (CJFS):
- $Tx_1$, $Tx_2$, $Tx_3$
Dual Thread Journaling

Serial commit for Write order

Commit Tx₁ Commit Tx₂

Time

Blocked

Tx₁

Lock Up Prepare DMA DMA Transfer Flush

Tx₂

Lock Up Prepare DMA DMA Transfer Flush
Dual Thread Journaling

Host-side

<table>
<thead>
<tr>
<th>Lock Up</th>
<th>DMA Prepare</th>
<th>DMA Transfer</th>
<th>Flush</th>
</tr>
</thead>
</table>

Storage-side

<table>
<thead>
<tr>
<th>Lock Up</th>
<th>DMA Prepare</th>
<th>DMA Transfer</th>
<th>Flush</th>
</tr>
</thead>
</table>

Serial commit for Write order

Blocked

Commit Tx1
Commit Tx2

Time
Dual Thread Journaling

- **Serial commit for Write order**
  - $T_x_1$
    - Lock Up
    - Prepare DMA
    - DMA Transfer
    - Flush
  - $T_x_2$
    - Lock Up
    - Prepare DMA
    - DMA Transfer
    - Flush
  - Blocked

- **Pipelined commit with Order-Preserving block device (USENIX FAST ’18)**
  - $T_x_1$
    - Lock Up
    - Prepare DMA
    - DMA Transfer
    - Flush
  - $T_x_2$
    - Lock Up
    - Prepare DMA
    - DMA Transfer
    - Flush

Time
Multi-Version Shadow Paging

\[ \text{Tx}_1 \quad \text{commit} \quad \text{Tx}_3 \]

\[ \text{Time} \]

\[ \text{Tx}_1, \text{Tx}_2, \text{Tx}_3 \]
Multi-Version Shadow Paging

```
Tx_1
Tx_2
Tx_3

Tx_1 Commit  Tx_2 Commit

Time
```
Multi-Version Shadow Paging

Tx₁ Commit  Tx₂ Commit

Tx₁
Tx₂
Tx₃

Time
Multi-Version Shadow Paging
Multi-Version Shadow Paging

�单元的版本影子页面

$Tx_1$  $Tx_2$  $Tx_3$

$Tx_1$ Commit  $Tx_2$ Commit  $Tx_3$ Commit  Time
Multi-Version Shadow Paging

Original page cache entries: □ □ □ □
Multi-Version Shadow Paging

Original page cache entries:

Tx1

Tx2

Tx3

Time
Multi-Version Shadow Paging

Original page cache entries:

**Time**

**TX1**

**TX2**

**TX3**

**Original page cache entries:**
Multi-Version Shadow Paging

Original page cache entries:

- **Tx1** Commit
- **Tx2** Commit
- **Tx3** Commit

---

**Tx1**

- V₁
- V₁
- V₁

**Tx2**

- V₂
- V₁

**Tx3**

- V₂
- V₂
- V₂

---

**Tx1**

- Commit

**Tx2**

- Commit

**Tx3**

- Commit

---

**Time**
Multi-Version Shadow Paging

Original page cache entries:

Time

TX1

TX2

TX3

TX1 Commit

TX2 Commit

TX3 Commit

V1  V1  V1

V2  V1

V2  V2  V2

Time
Multi-Version Shadow Paging

Original page cache entries:

File operations

Time
Opportunistic Coalescing

- When versions are exhausted, transaction commits are serialized
- The running transaction is locked and waits for preceding transaction commits

![Diagram showing the process of commit and flush with time on the x-axis and steps on the y-axis. The steps include Lock, Prepare, DMA, and DMA Transfer.]
Opportunistic Coalescing

- When versions are exhausted, transaction commits are serialized
- The running transaction is locked and waits for preceding transaction commits
Opportunistic Coalescing

- When versions are exhausted, transaction commits are serialized.
- The running transaction is locked and waits for preceding transaction commits.

![Diagram of Opportunistic Coalescing]

**Diagram Explanation:**

- `Tx1` and `Tx2` represent two transactions.
- The diagram illustrates the sequence of operations including Commit, Lock, Prepare, DMA, Flush, DMA Transfer, and Flush.
- The transactions are shown from left to right with time progression.
- The coalescing technique allows transactions to be processed more efficiently by minimizing the overhead of locking and preparing.

**Key Points:**

- **Commit** operations are followed by **Lock** operations, which are then followed by **Prepare**.
- **DMA** transfers data between memory and a device.
- **Flush** operations are used to clear transaction logs.
- The diagram visually demonstrates how transactions can be coalesced or merged to optimize performance.

---

**Technical Details:**

- **Lock Up**: Acquires a lock on a resource.
- **Prepare**: Prepares a transaction for commit.
- **DMA Transfer**: Data Memory Access (DMA) transfers data.
- **Flush**: Clears the transaction log.

---

**Conclusion:**

Opportunistic coalescing optimizes transaction processing by minimizing the overhead of lock and prepare operations, allowing for more efficient data management and processing.
Opportunistic Coalescing

- When versions are exhausted, transaction commits are serialized
- The running transaction is locked and waits for preceding transaction commits
Compound Flush

- Commit:
  - Tx₁: Lock Up, Prepare DMA
  - Tx₂: Lock Up, Prepare DMA
  - Tx₃: Lock Up, Prepare DMA

- Flush:
  - DMA Transfer, Flush
  - DMA Transfer, Flush
  - DMA Transfer, Flush

Time
Compound Flush

- Commit
  - Lock Up
  - Prepare DMA

- Flush
  - DMA Transfer
  - Flush
  - DMA Transfer
  - Flush

Time

Still serial
Compound Flush

**Diagram:**

- **Tx₁:** Commit → Lock Up → Prepare DMA → DMA Transfer → Flush
- **Tx₂:** Commit → Lock Up → Prepare DMA → DMA Transfer → Flush
- **Tx₃:** Commit → Lock Up → Prepare DMA → DMA Transfer → Flush

**Time Line:**

- **Commit:**
  - **Tx₁:** Lock Up → Prepare DMA
  - **Tx₂:** Lock Up → Prepare DMA
  - **Tx₃:** Lock Up → Prepare DMA

- **Flush:**
  - **Tx₁:** DMA Transfer
  - **Tx₂:** DMA Transfer
  - **Tx₃:** DMA Transfer

**Notes:**

- Still serial
Compound Flush

- **Commit**: Lock Up, Prepare DMA.
- **Flush**: DMA Transfer, Flush.
- **Tx1**, **Tx2**, **Tx3**: DMA Transfer, Flush, DMA Transfer, Flush.

Still serial
Compound Flush

Time

Commit

Flush

Tx₁

Lock

Prepare

DMA

Lock

Prepare

DMA

DMA

Transfer

Flush

DMA

Transfer

Flush

DMA

Transfer

Flush

Still serial

Time

Tx₂

Lock

Prepare

DMA

Lock

Prepare

DMA

DMA

Transfer

Flush

DMA

Transfer

Flush

DMA

Transfer

Flush

Tx₃

Lock

Prepare

DMA

Lock

Prepare

DMA

DMA

Transfer

Flush

DMA

Transfer

Flush

DMA

Transfer

Flush
Compound Flush

Still serial

cache_barrier
Compound Flush

- Tx₁
  - Lock Up
  - Prepare DMA

- Tx₂
  - Lock Up
  - Prepare DMA

- Tx₃
  - Lock Up
  - Prepare DMA

Still serial

- Commit
- Flush

- DMA Transfer
- DMA Transfer
- DMA Transfer
- DMA Transfer
- DMA Transfer

- Flush

- cache_barrier

Time
Compound Flush

Time

Commit

- Lock Up
- Prepare DMA

Flush

- DMA Transfer
- Flush
- DMA Transfer
- Flush
- DMA Transfer
- Flush

Still serial

cache_barrier
Compound Flush

Time

Commit

Flush

cache_barrier

DMA Transfer

DMA Transfer

DMA Transfer

Commit

Flush

Commit

Flush

Tx₁

Tx₂

Tx₃

Lock Up

Prepare DMA

Lock Up

Prepare DMA

Lock Up

Prepare DMA

Lock Up

Prepare DMA

Lock Up

Prepare DMA

Still serial
Compound Flush

Commit

Flush

Time

\( \text{Tx}_1 \)

\( \text{Tx}_2 \)

\( \text{Tx}_3 \)

Still serial

\( \text{Lock Up} \)

\( \text{DMA Prepare} \)

\( \text{DMA Transfer} \)

\( \text{Flush} \)

\( \text{DMA Transfer} \)

\( \text{Flush} \)

\( \text{DMA Transfer} \)

\( \text{Flush} \)

\( \text{Lock Up} \)

\( \text{DMA Prepare} \)

\( \text{DMA Transfer} \)

\( \text{Flush} \)

\( \text{DMA Transfer} \)

\( \text{DMA Transfer} \)

\( \text{Lock Up} \)

\( \text{DMA Prepare} \)

\( \text{DMA Transfer} \)

\( \text{Flush} \)

\( \text{DMA Transfer} \)

\( \text{DMA Transfer} \)

\( \text{Lock Up} \)

\( \text{DMA Prepare} \)

\( \text{DMA Transfer} \)

\( \text{ Flush} \)

\( \text{DMA Transfer} \)

\( \text{DMA Transfer} \)

\( \text{Lock Up} \)

\( \text{DMA Prepare} \)

\( \text{DMA Transfer} \)

\( \text{Flush} \)

\( \text{DMA Transfer} \)

\( \text{DMA Transfer} \)

\( \text{Lock Up} \)

\( \text{DMA Prepare} \)

\( \text{DMA Transfer} \)

\( \text{Flush} \)

\( \text{DMA Transfer} \)

\( \text{DMA Transfer} \)

cache_barrier
Compound Flush

Time

Commit

Flush

_cache_barrier

Still serial
Evaluation
Evaluation Setup

- CPU: Intel Xeon Gold 6320 (2.1 GHz, 2 Socket X 20 core = 40 core)
- Memory: 512GB DRAM
- Storage: Samsung 970 Pro (MLC, NVMe)
- OS (Kernel)
  - CentOS 7.4 (Linux Kernel 5.18.18)
- Filesystem: EXT4, BarrierFS, EXT4 with fast commit, SpanFS, CJFS
- Workloads: Varmail-shared, Varmail-split, Dbench, OLTP-Insert
  - Varmail-shared: Varmail with a shared directory
  - Varmail-split: Varmail with a per-thread directory
Macro Benchmarks

**Variants**
- **EXT4**
- **BarrierFS**
- **Fast Commit**
- **SpanFS**
- **CJFS**

**Varmail**

**Varmail-shared**
- Compared to **EXT4**: 1.6X
- **BarFS**: 2.7X
- **FC**: 6.3X
- **SpanFS**: 17X

**Varmail-split**
- Compared to **EXT4**: 1.3X
- **BarFS**: 1.1X
- **FC**: 1.6X
- **SpanFS**: 16X

**Dbench**

**OLTP-Insert**

- Compared to **EXT4**: 1.2X
- **BarFS**: 1X
- **FC**: 1.4X
- **SpanFS**: 15X

- Compared to **EXT4**: 1.7X
- **BarFS**: 1X
- **FC**: 2.4X
- **SpanFS**: 2.3X
Command Queue Depth

- Workload: Varmail with 40 threads

Transactions are transferred and flushed concurrently
Conclusion

- We propose CJFS, Concurrent Journaling Filesystem
- CJFS achieves concurrent transaction commit with four techniques
  - Dual Thread Journaling
  - Multi-Version Shadow Paging
  - Opportunistic Coalescing
  - Compound Flush
- CJFS improves the throughput in macro benchmarks
  - Varmail-shared: 1.6X, Varmail-split: 1.3X, Dbench: 1.2X, OLTP-Insert: 1.7X
Question & Answer

https://github.com/ESOS-Lab/cjfs