Unioning of Buffer cache and Journaling layers with NV-Memory

FAST'13

데이터 공학 연구실 이지환

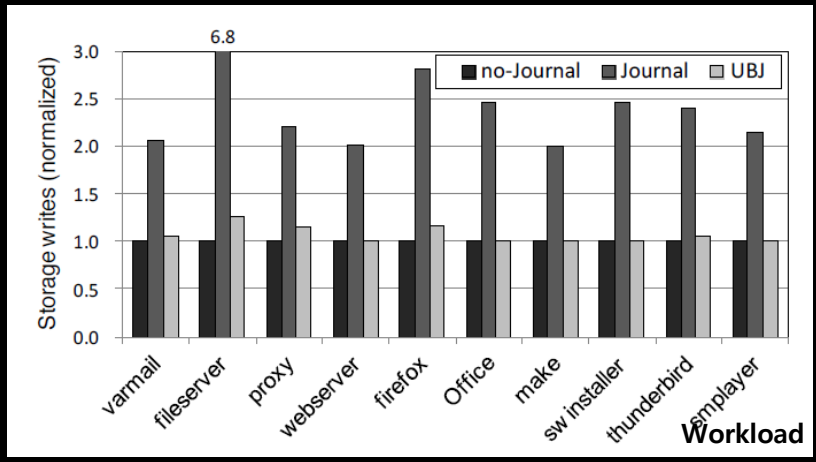
Contents

0. Introduction

- Journaling

  logs updates to storage

  + High reliability, fast recovery
  − Write traffic (**commit**)



Write traffic

- NV RAM (Non-volatile memory)

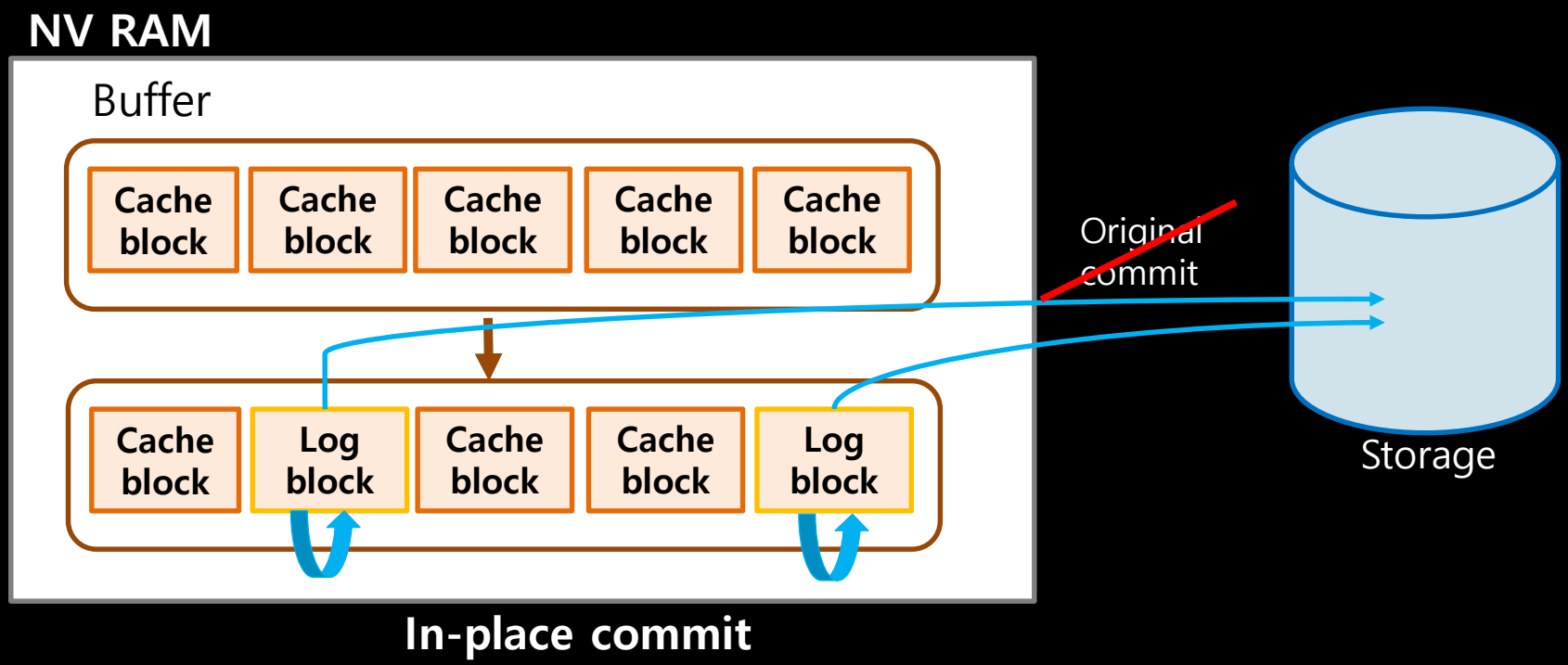
  + Performance, Durability
  − Cost, Consistency

1. Key Concepts

- **Buffer cache**
- **Journaling** (recovery)

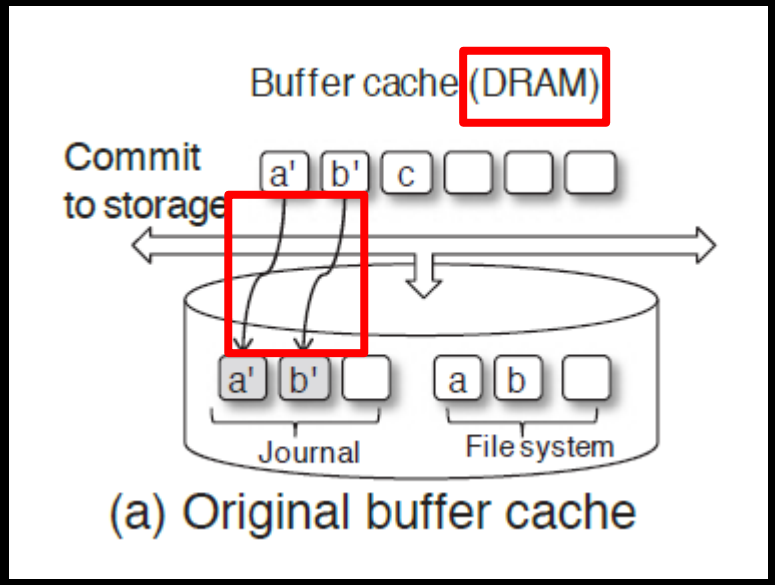
} **UBJ** (**U**nion **B**uffer cache and **J**ournaling layers)

- **NV RAM** (durability)
- **In-place commit** (write ↓)

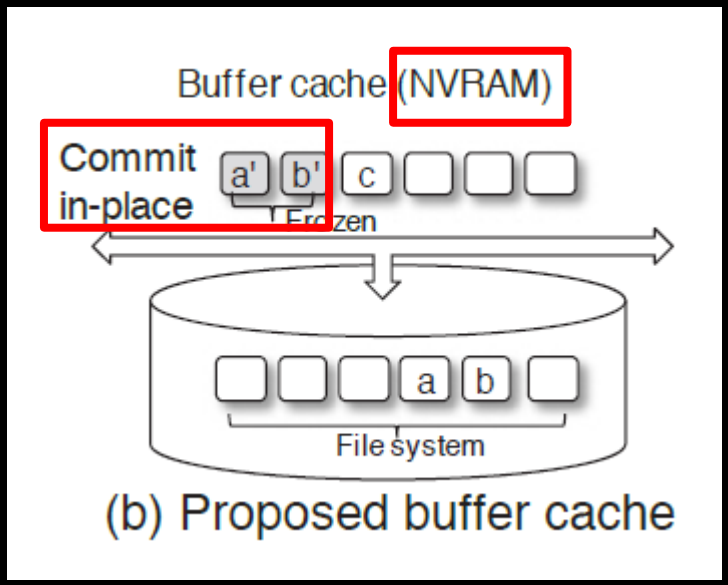


1. Key Concepts

- Original vs UBJ Architecture



(a) Original buffer cache



(b) Proposed buffer cache

- Write

	Original / Journaling X	Original / Journaling O	UBJ
Dirty block	O	O	O
Commit to storage	X	**O**	X
Checkpoint	X	O	O

2. Implementation - UBJ

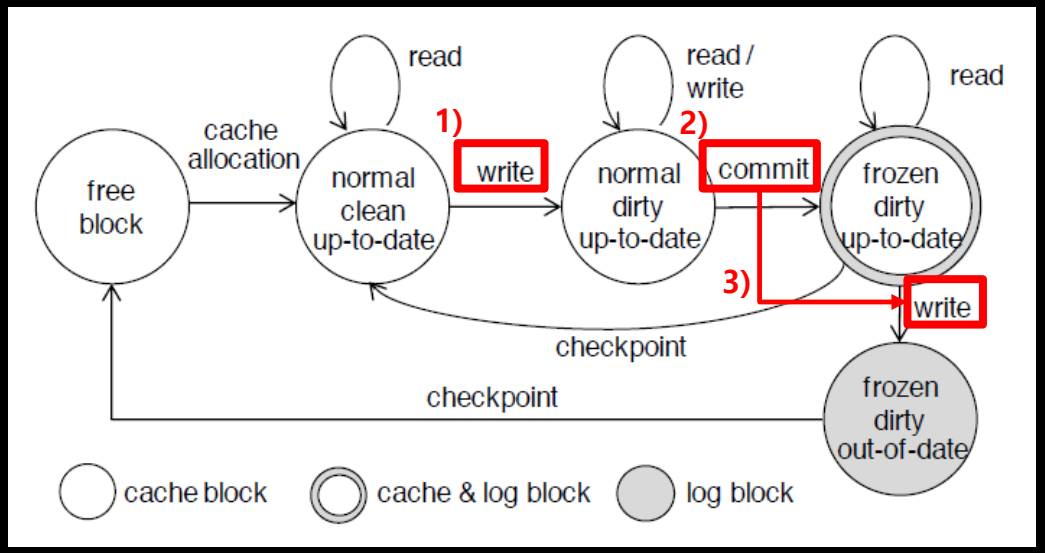
- 3 States

1) **clean / dirty**

Whether the block has been modified since it entered the cache

2) **normal / frozen**

Whether the block is a (normal) cache block or a (frozen) log block



State diagram in UBJ scheme

3) **up-to-date / out-of-date**

Whether the block is the most recent version or not

↳ last write → whether be committed or not

2. Implementation - UBJ

- 3 Operations

1) Read / Write operations

Read - do not alter the states

Write - normal → update
frozen → copy to new block

2) Commit operations

: normal → frozen
by 3 types of transactions

(1) **Running transaction**

Commit

: A list of normal dirty blocks

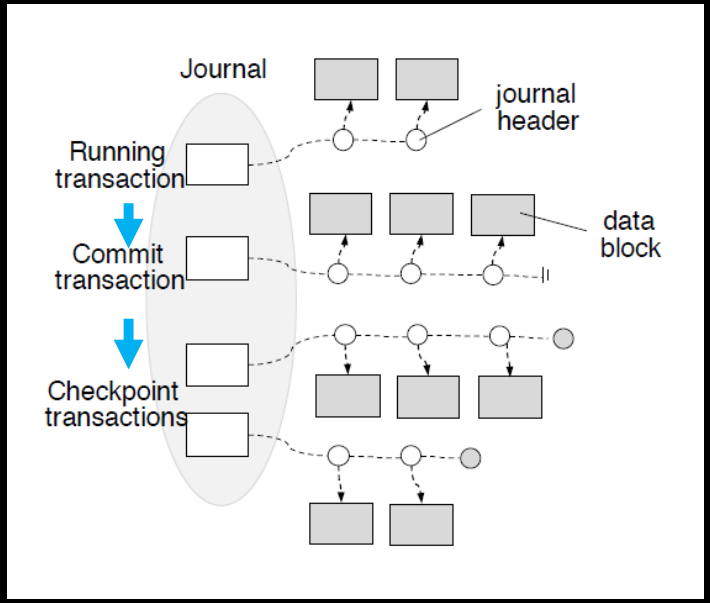
(2) **Commit transaction**

All frozen block

: Running → Commit , normal → frozen (**In-place commit**)

(3) **Checkpoint transaction**

: Commit → Checkpoint, located in buffer cache until checkpoint operation



Data Structures for UBJ

2. Implementation - UBJ

- 3 Operations

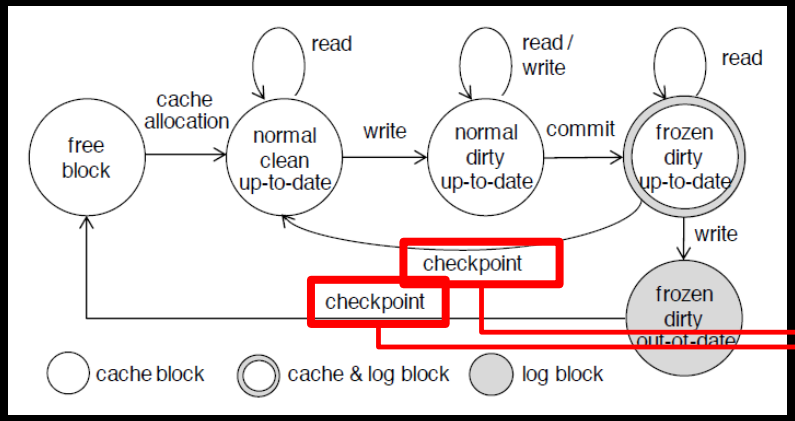
3) Checkpoint operations

: updates storage with committed data

(1) scans checkpoint transaction lists

(2) reflects them on to their permanent locations in the storage

After checkpoint operation is completed,

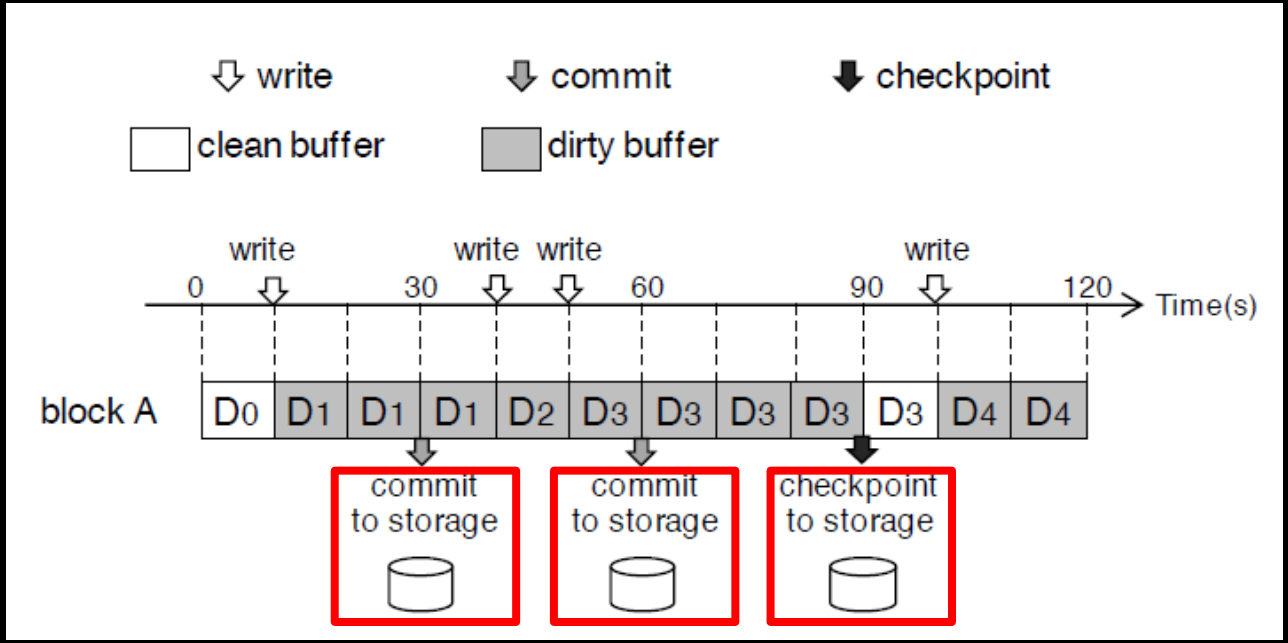


up-to-date → **Reuse**

out-of-date → **Free**

2. Implementation - UBJ

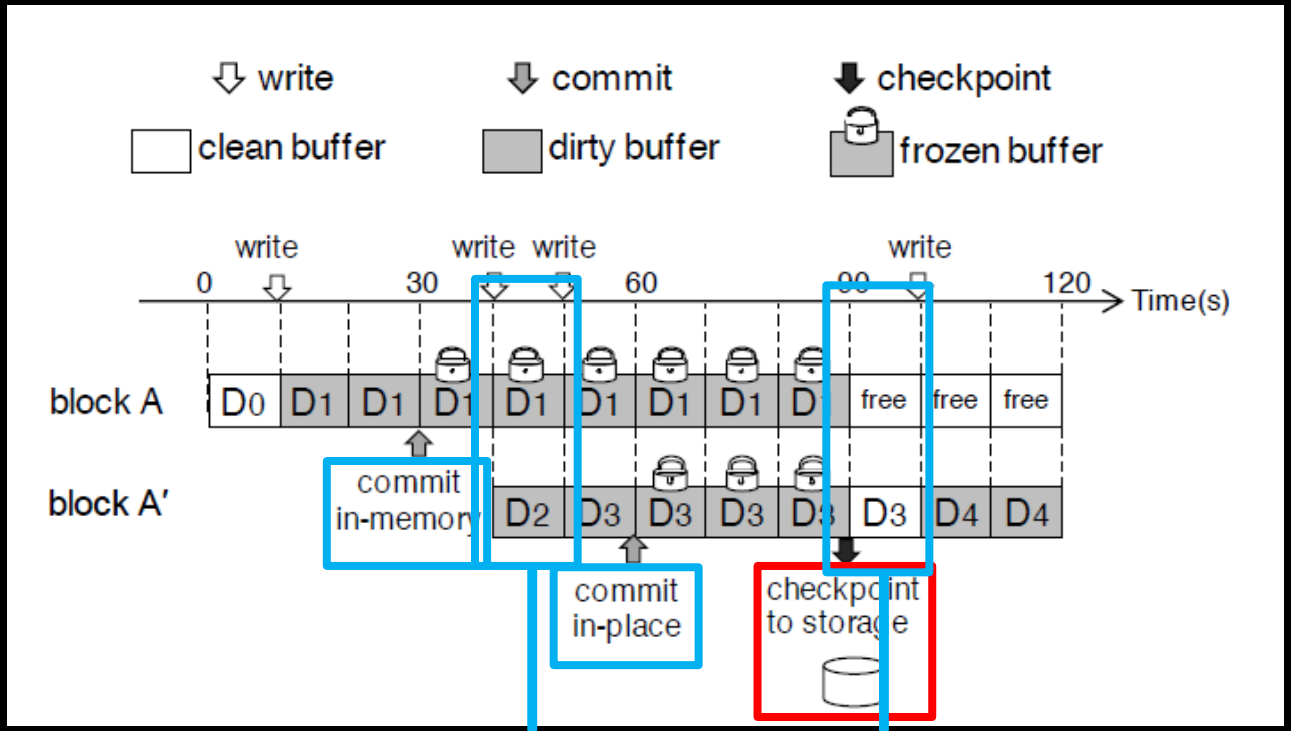
- Example - original



2. Implementation - UBJ

- Example - UBJ

* Write - normal → update
  frozen → copy to new location
* commit - normal → frozen
* checkpoint - out-of-date → free
  up-to-date → reuse



Commit and then write
block A : out-of-date
block A' : up-to-date

Checkpoint
block A - out-of-date → free
block A' - up-to-date → reuse

2. Implementation - UBJ

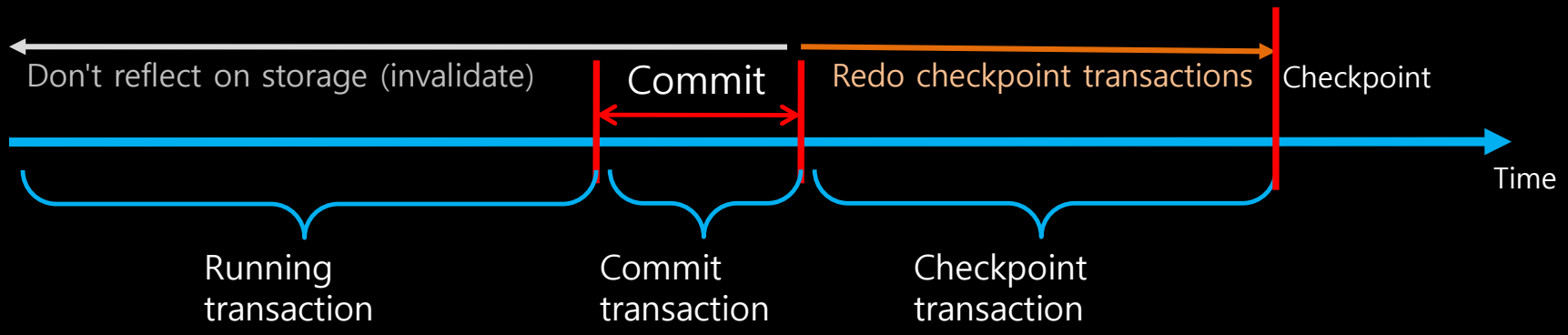
- Recovery rules ≈ Redo log

Check whether transactions are committed or not

1) Running transactions

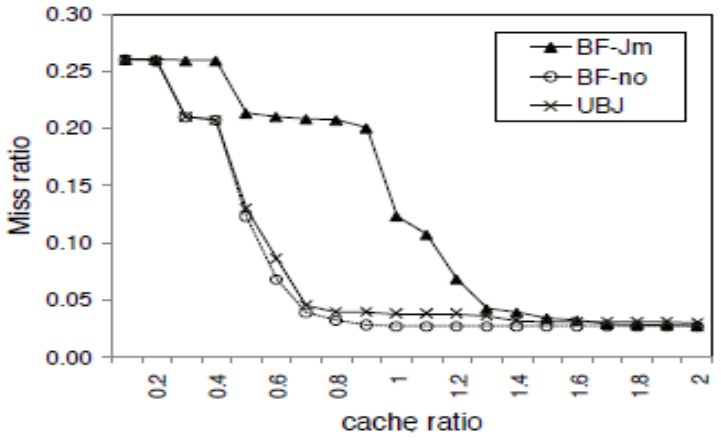
2) Commit transactions

3) Checkpoint transactions

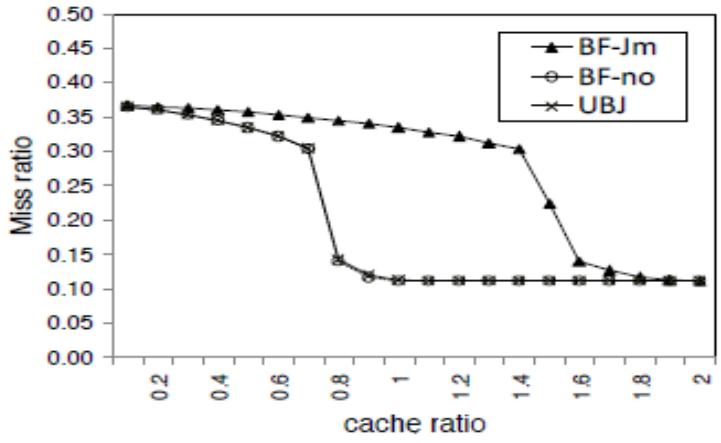


3. Performance

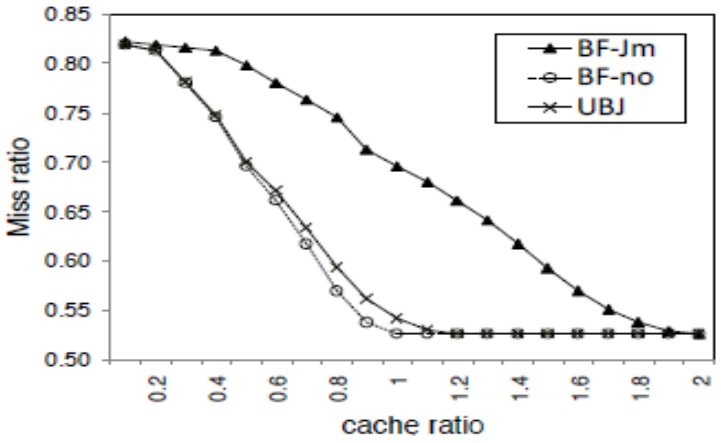
- Cache miss ratio



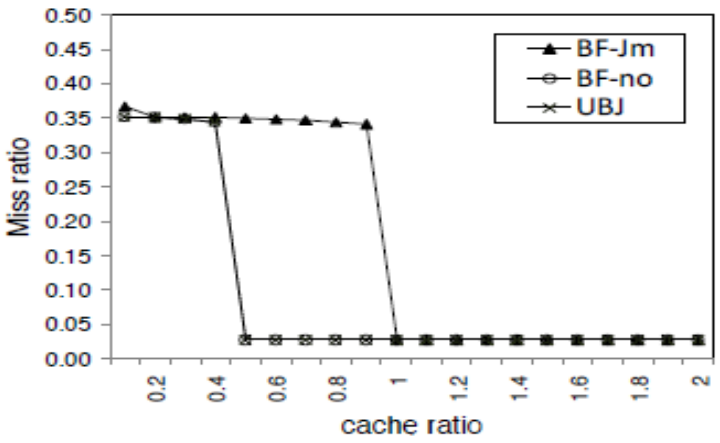
(a) varmail



(b) proxy



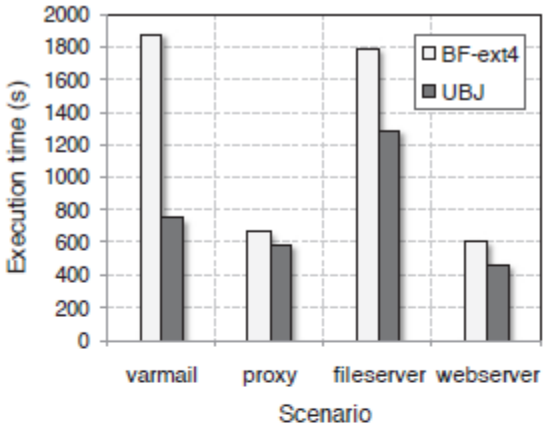
(c) fileserver



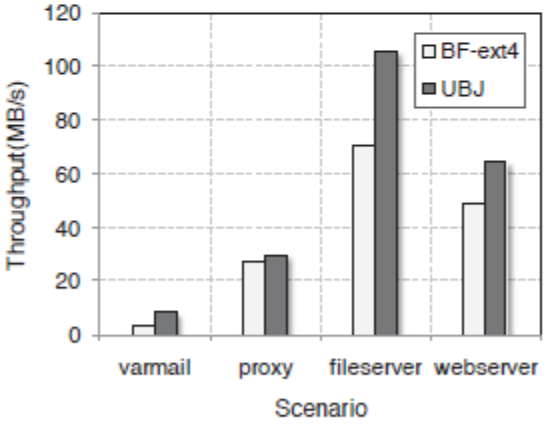
(d) webserver

Fig. 6 Cache miss ratio as a function of the cache size.

3. Performance

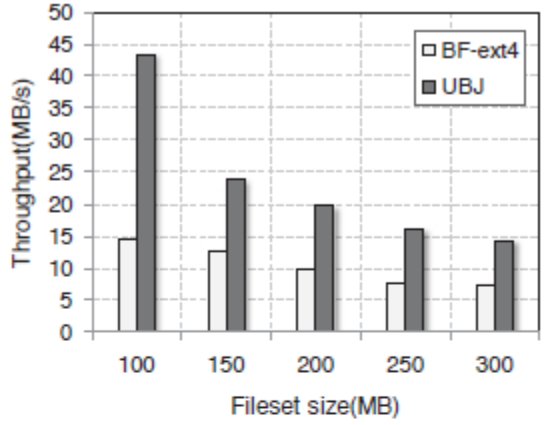


(a) execution time

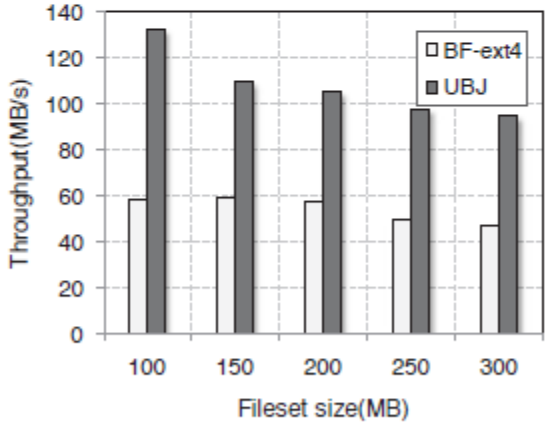


(b) throughput

Fig. 10. Throughput and execution time of the Filebench workloads.

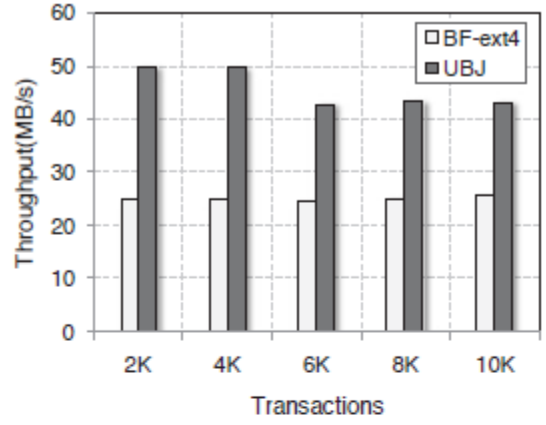


(a) random write

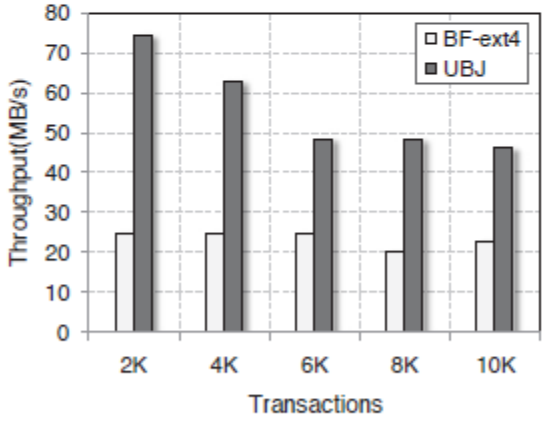


(b) sequential write

Fig. 11. Throughput of IOzone as the fileset size is varied.



(a) read



(b) write

Fig. 12. Throughput of Postmark as the number of transactions is varied.

4. Limitation

- Implementation
  - Design assumes non-volatile main memory.
    → Actually, they used a portion of the DRAM as buffer/journal space
  - Structural difference between NVRAM and DRAM
    → Need to adjust discrepancy
  - Actual performance
    → NVRAM < DRAM

Q & A