Non-volatile Memory (NVM) Logging

2016.05.20

Takashi HORIKAWA

Who am I

Name

Takashi HORIKAWA, Ph. D.

Research interests

Performance evaluation of computer & communication systems, including

performance engineering of IT systems

with slightly shifting the focus of the research to CPU scalability

Papers

Latch-free data structures for DBMS: design, implementation, and evaluation, SIGMOD ‘13

An Unexpected Scalability Bottleneck in a DBMS: A Hidden Pitfall in Implementing Mutual Exclusion, PDCS ‘11

An approach for scalability-bottleneck solution: identification and elimination of scalability bottlenecks in a DBMS, ICPE '11

A method for analysis and solution of scalability bottleneck in DBMS, SoICT '10

Contents

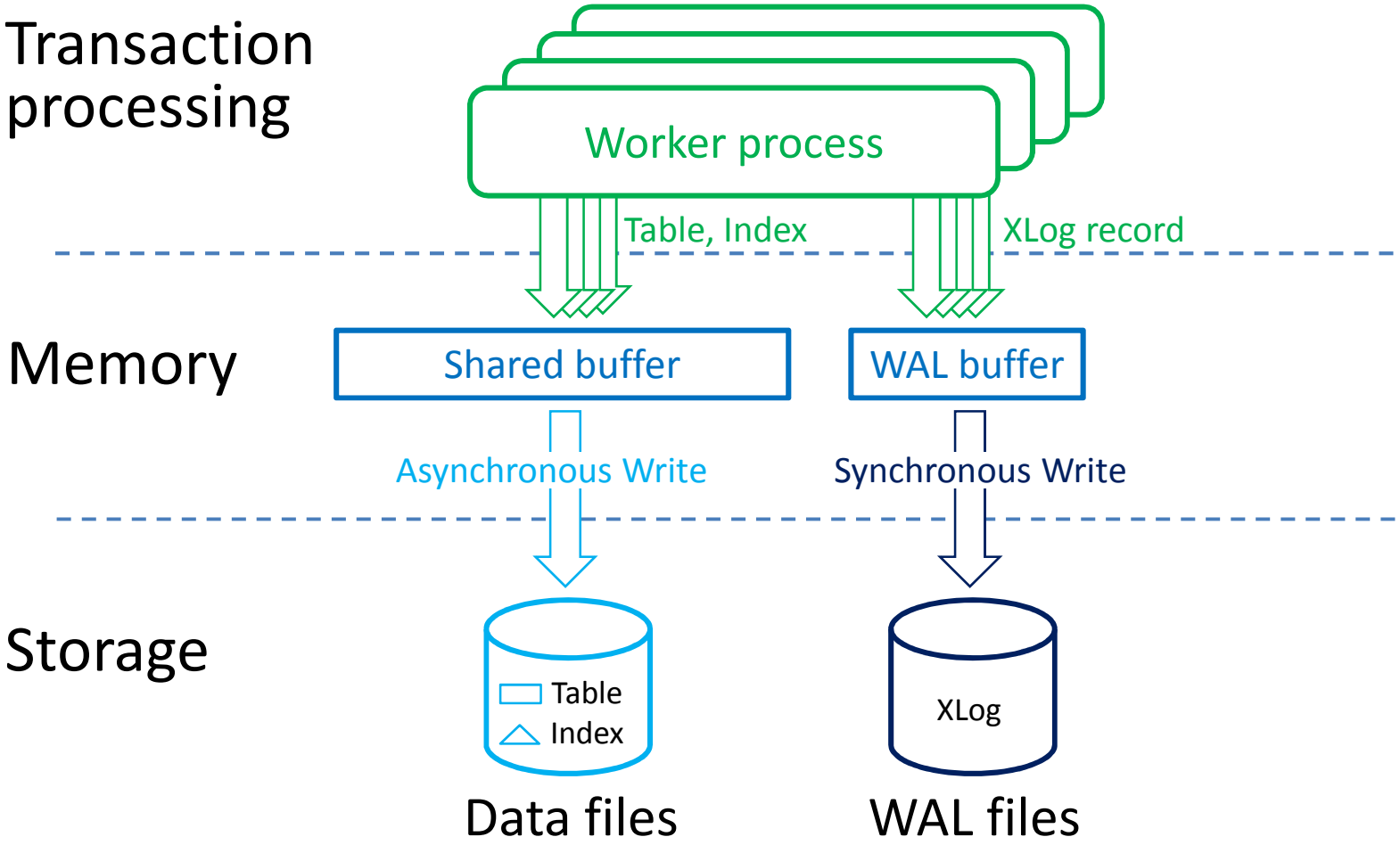
- Introduction
- Problems to be solved
- Implementation
- Evaluation
- Technical trends
- Conclusion

Introduction

Write Ahead Logging
Sync. vs Async. Commit
Difference in Performance
Fundamental idea for NVM Logging
Byte or Block Addressable NVM
Byte addressable NVMs

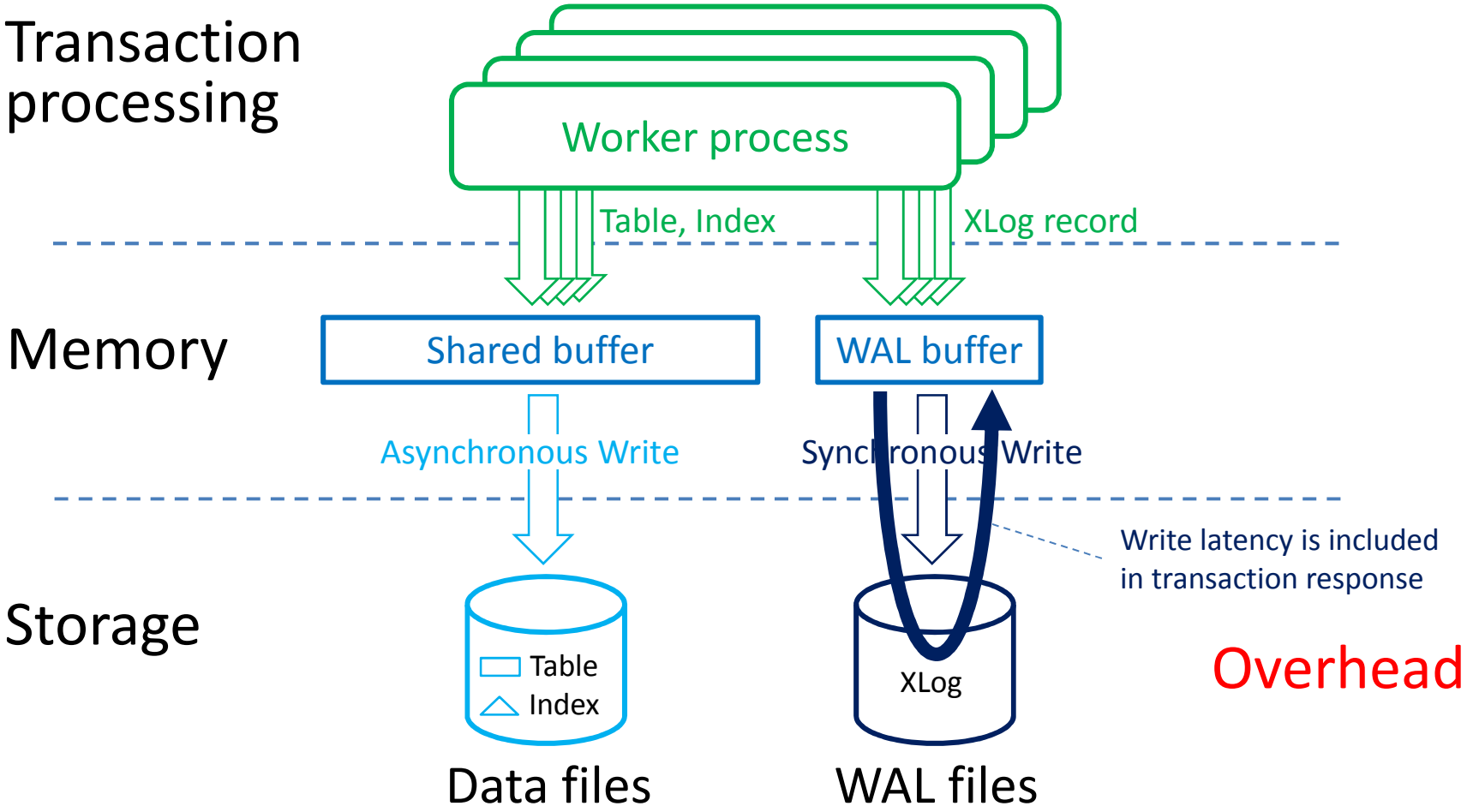
Write Ahead Logging

Widely used method to make transaction durable

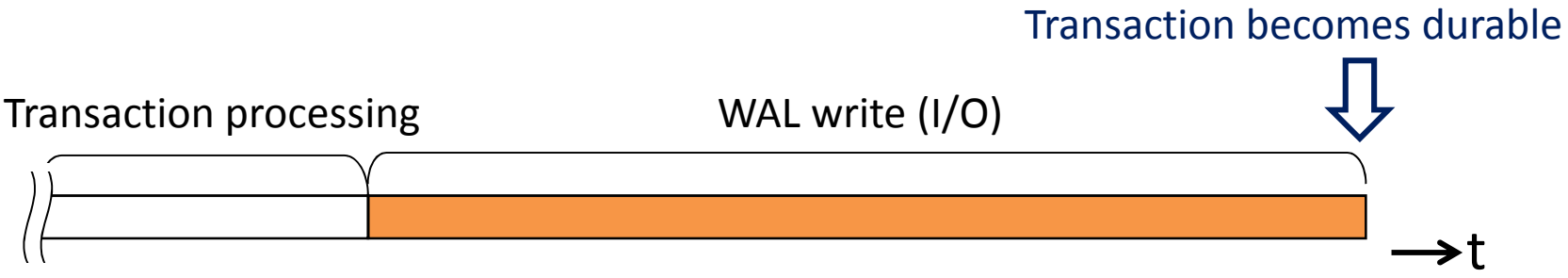


Write Ahead Logging

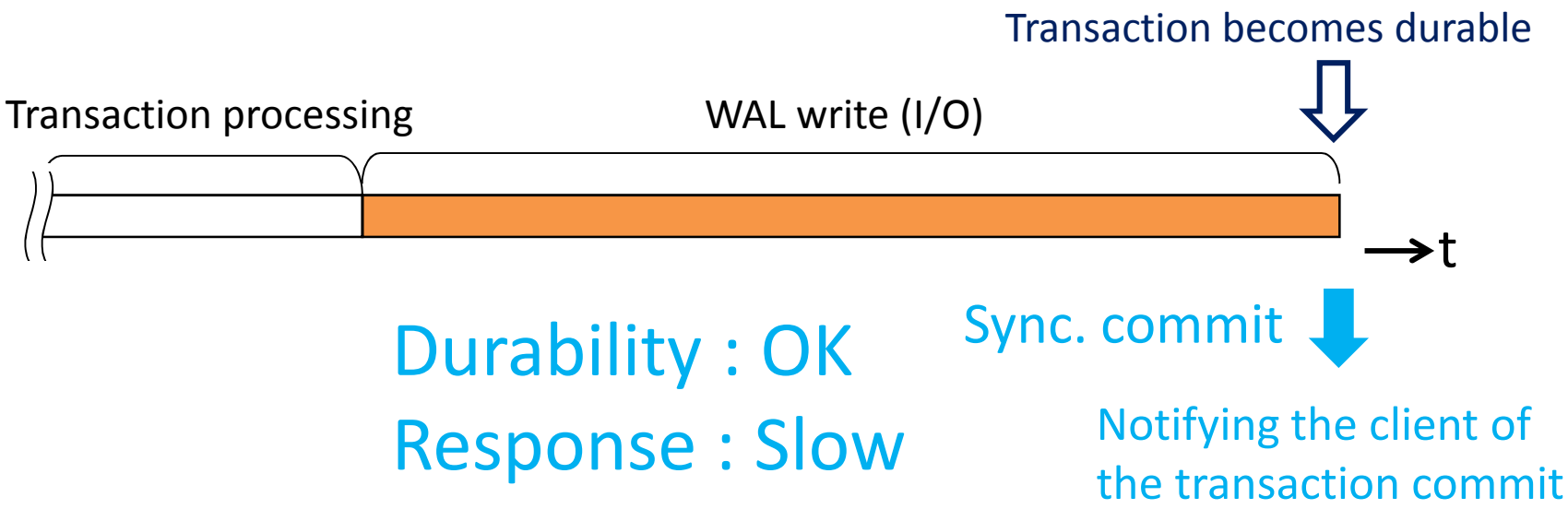
Widely used method to make transaction durable



Sync. vs Async. Commit

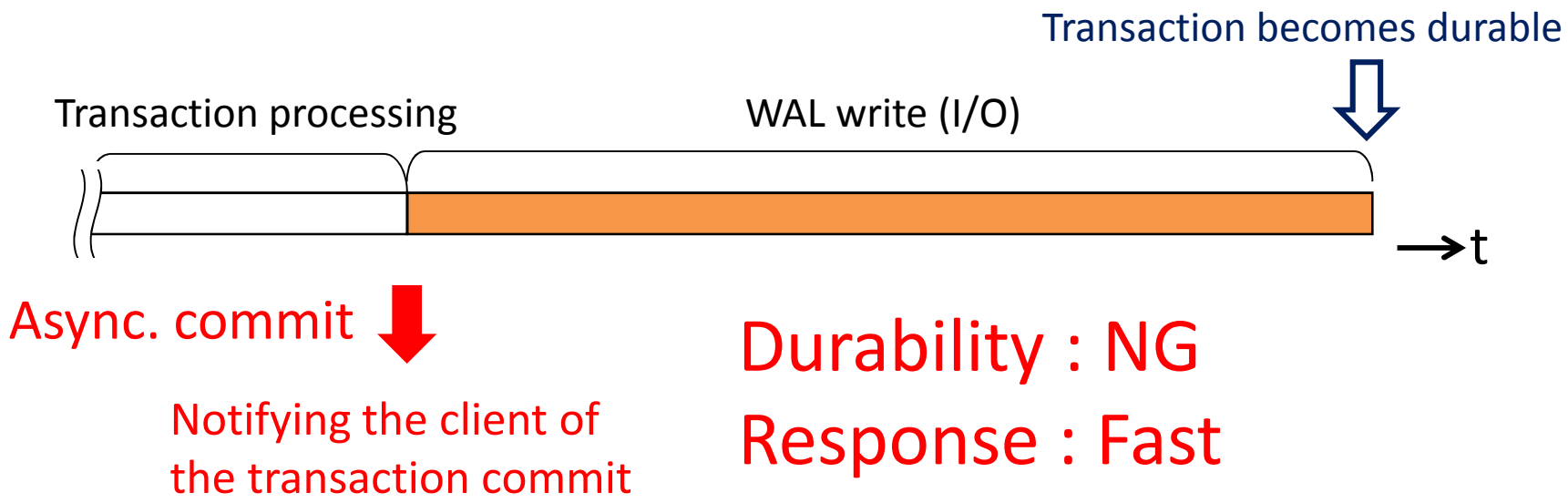


Sync. vs Async. Commit



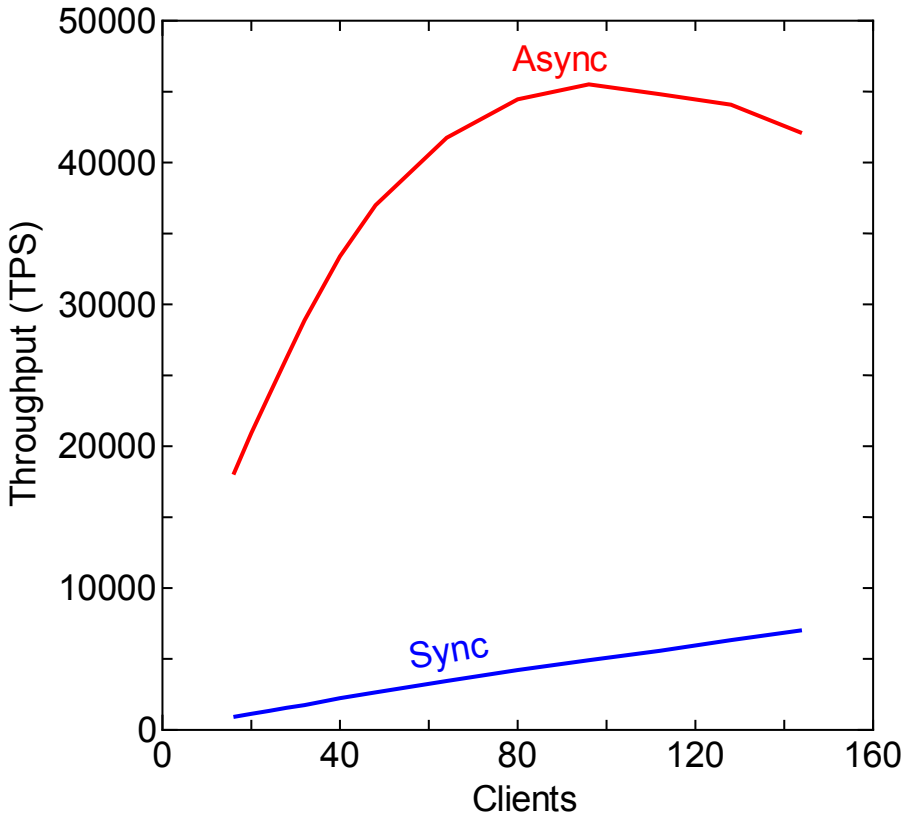
In sync. commit
the client is notified of the transaction commit
after the transaction becomes durable.

Sync. vs Async. Commit

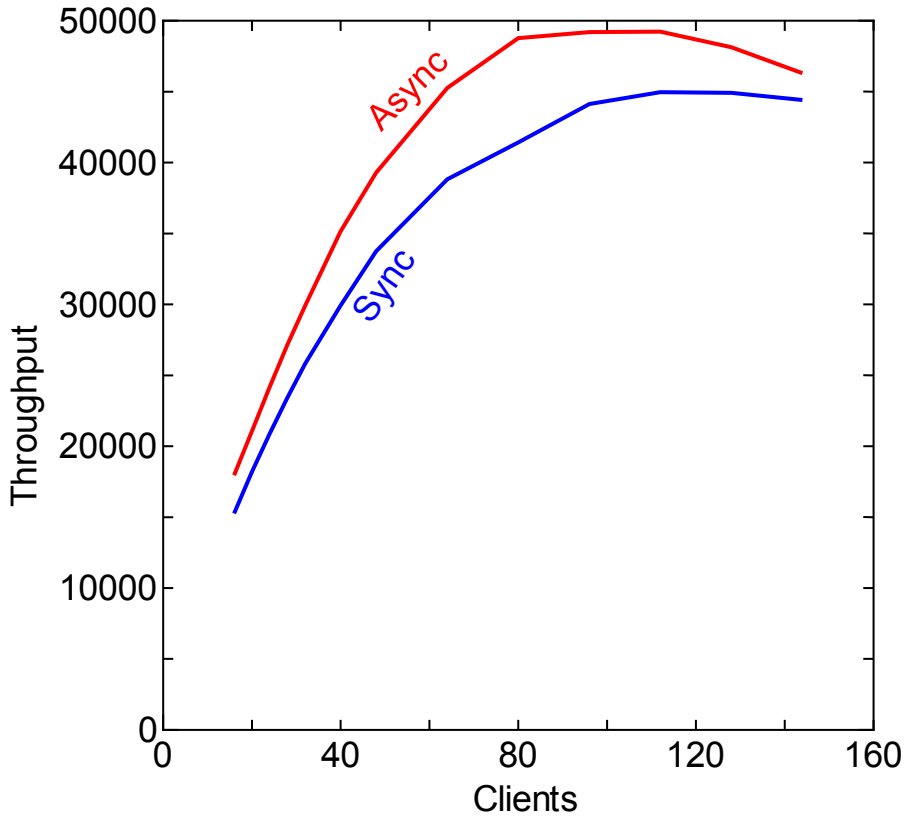


In async. commit
the client is notified of the transaction commit
before the transaction becomes durable.

Difference in Performance (PGBENCH)

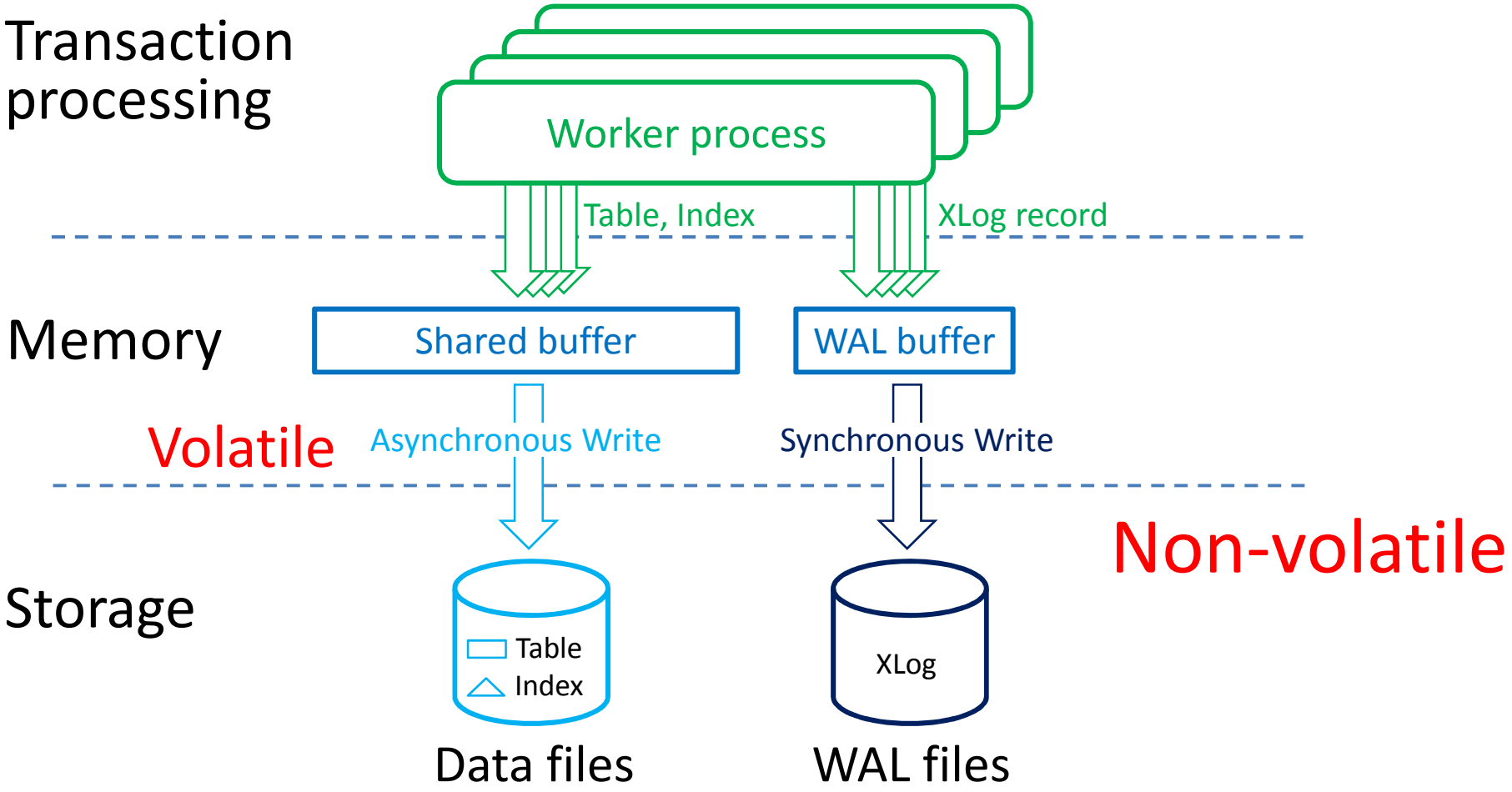


Disk-drive cache off

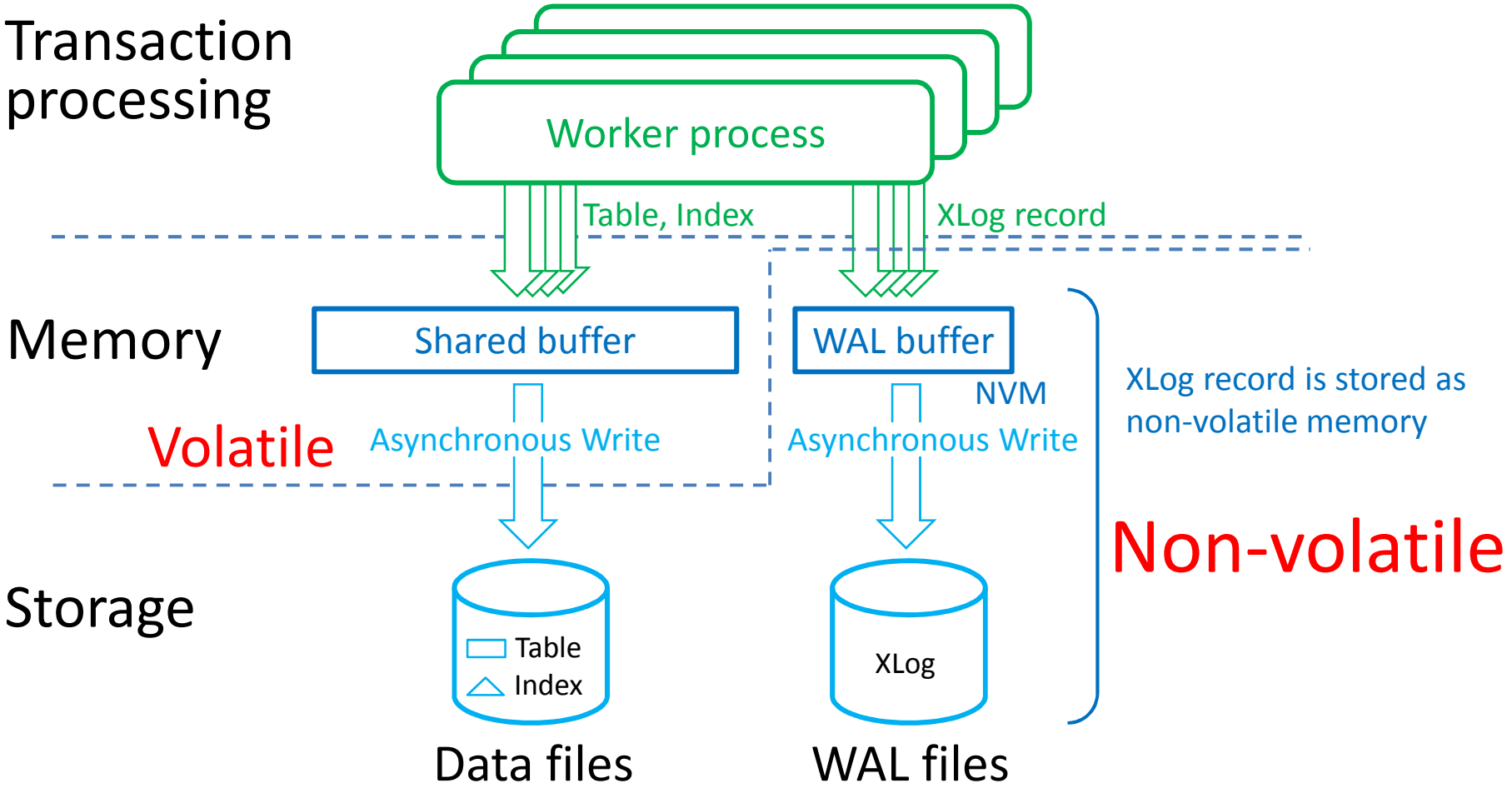


Disk-drive cache on

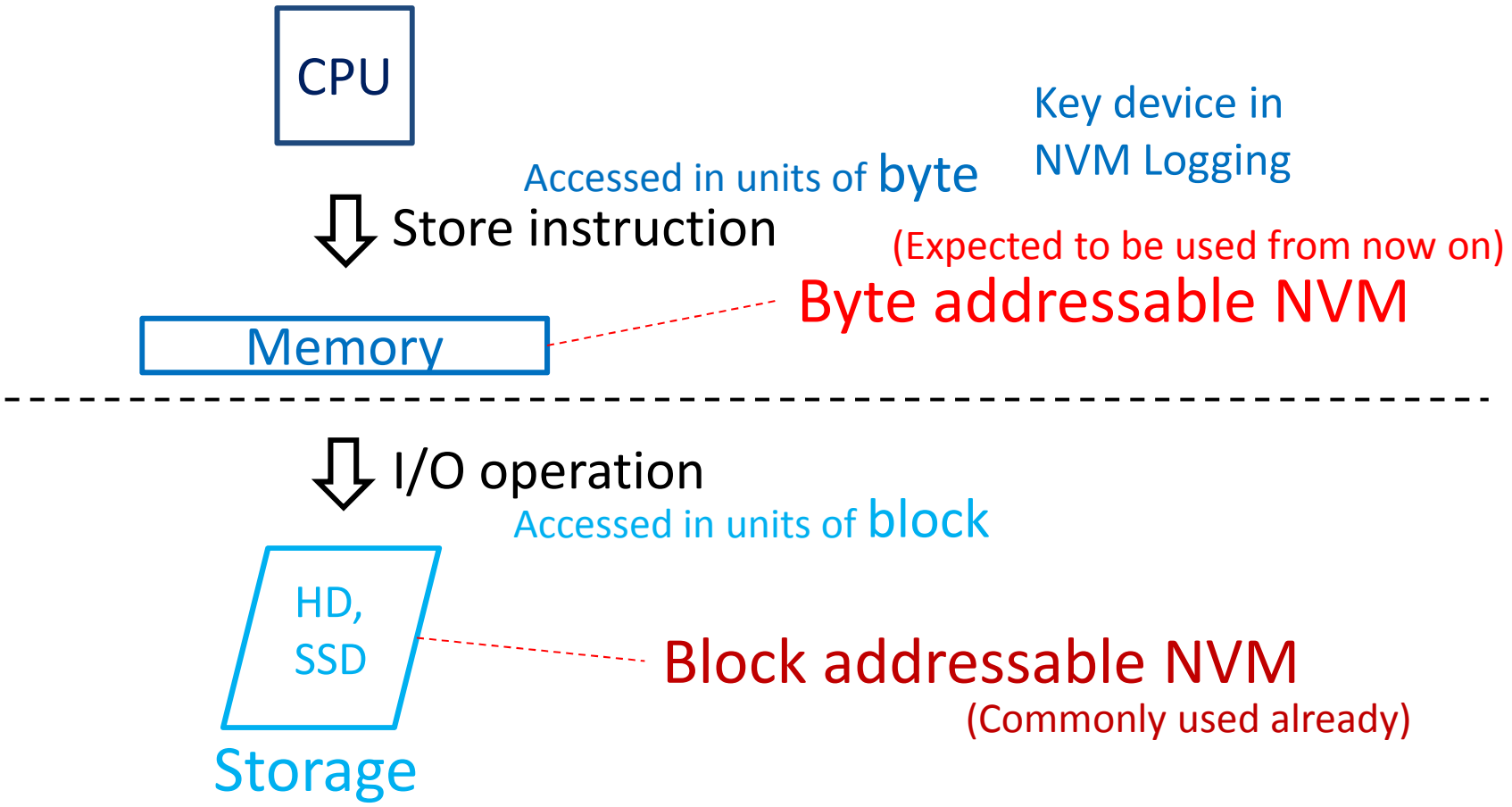
Fundamental idea for NVM Logging



Fundamental idea for NVM Logging

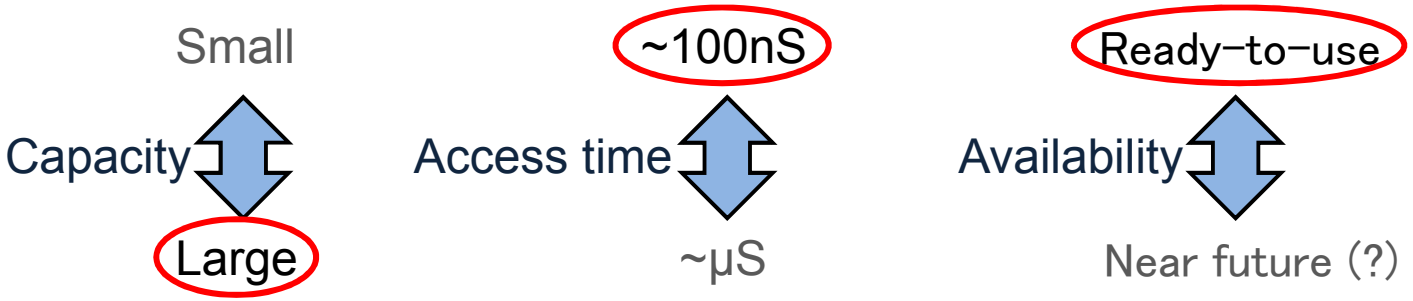


Byte or Block Addressable NVM



Byte addressable NVMs

- Combination of existing technologies -- DRAM, SSD, battery（NVDIMM）
  - AgigA Tech (Micron Technology)
  - Viking Techonogy
  - SK Hynix



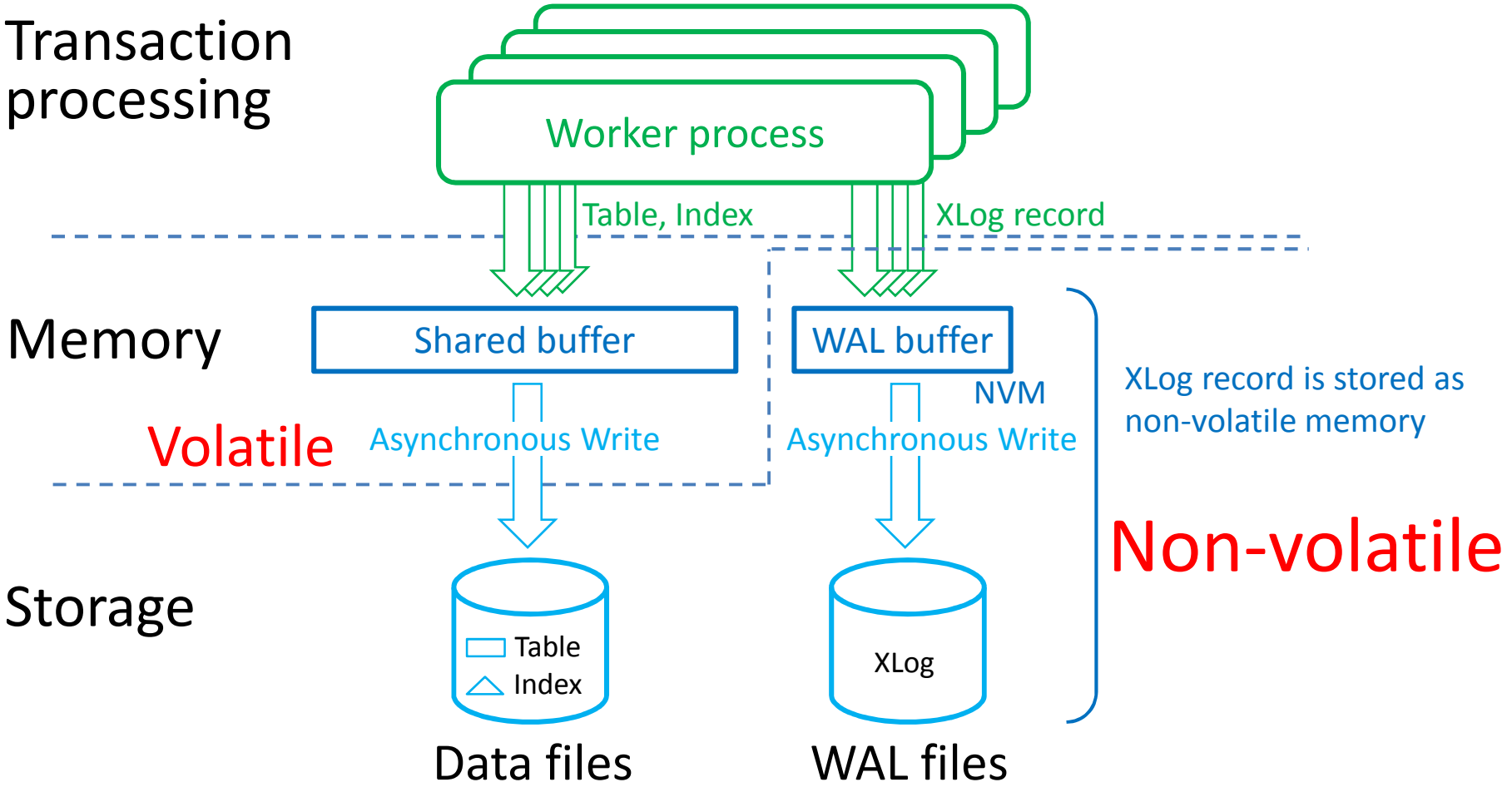
- Use of a new memory cell（Storage Class Memory）
  - Phase Change Memory (PCM)
  - Magnetic Random Access Memory (MRAM)
  - Ferroelectric RAM (FRAM)
  - The memristor

Problems to be solved

Fundamental idea for NVM Logging (Again)
It is not simple than it looks
Necessary condition for Recovery
Problems
- Partial write
- Unreachable XLog Record
- CPU cache effect

Fundamental idea for NVM Logging

(Again)

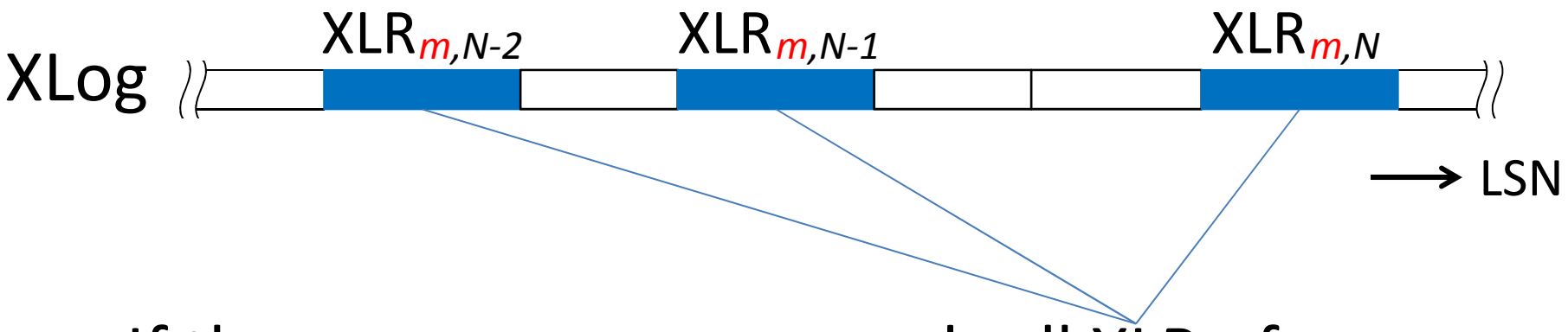


It is not simple than it looks

- Naive implementation of NVM Logging
  - Allocating WAL buffer in NVM area
  - Using asynchronous commit mode

- Problems are:
  - Partial write
  - Unreachable XLog Record
  - CPU cache effect

Necessary condition for Recovery



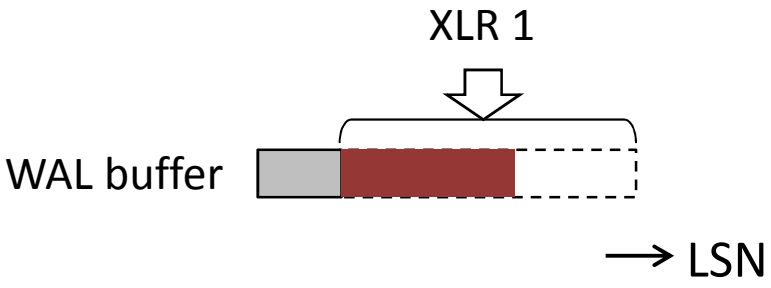
- If the recovery process reads all XLR of transaction *m* correctly
  - ➢ transaction *m* is possible to be recovered
- Else transaction *m* will be lost

A worker process finishes the commit of the transaction after all XLRs of the transaction are stored in the non-volatile memory.

Partial write

- The recovery process will read an incomplete XLR

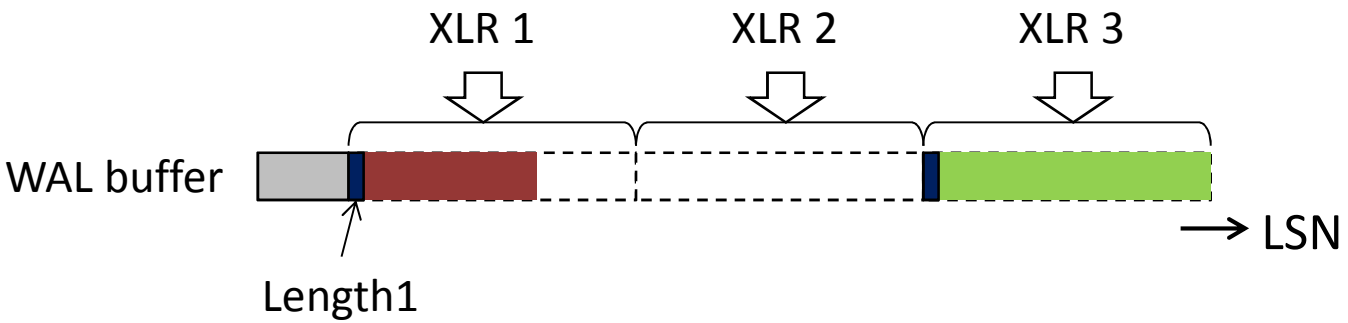
  if the system crashes in the middle of writing a XLR



(CRC in the XLR may be effective but it is not perfect)

Unreachable XLog Record

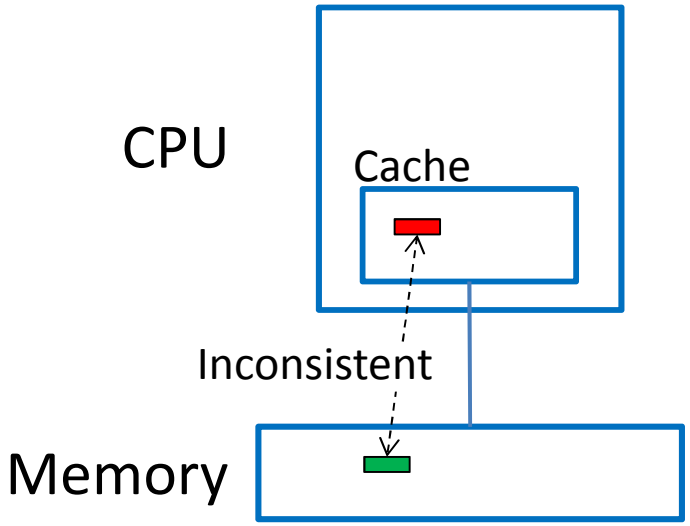
- The recovery process cannot find a XLR of a committed transaction
  - A worker process finishes writing of XLR 3 before another worker process begin to write XLR 2



XLog reader finds the head position of a XLR by adding the head position of the previous XLR and its length

CPU cache effect

- The recovery process will read inconsistent XLR

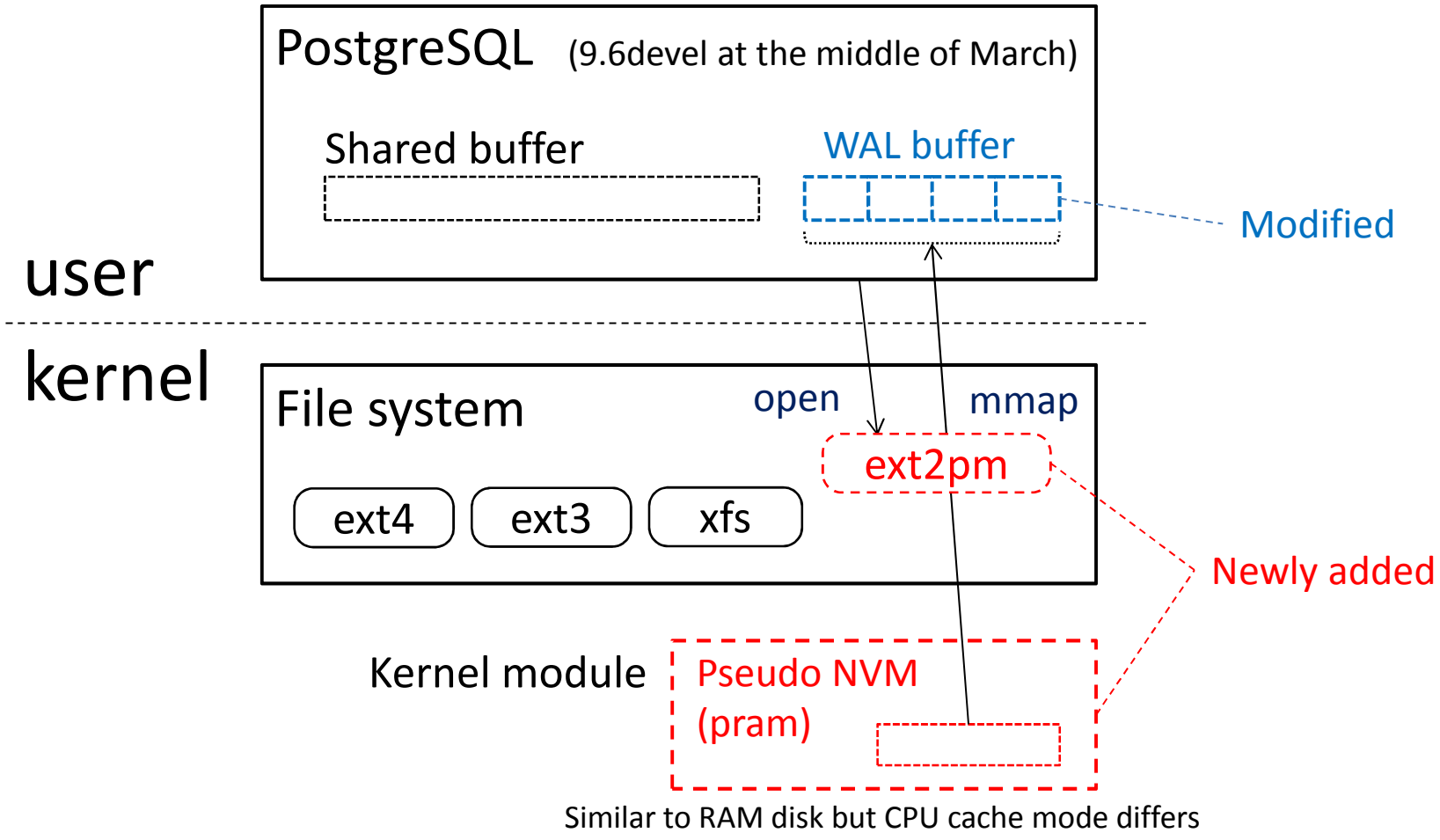


If the CPU internal cache uses write-back policy, a XLR written by the CPU does not reach to the memory immediately

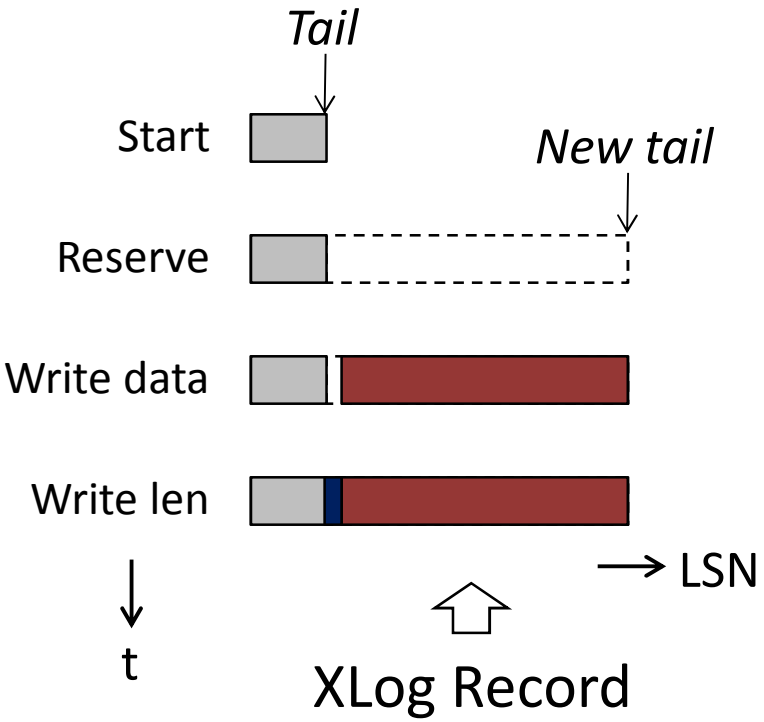
Implementation

Prototype architecture
Preventing partial write
Preventing an unreachable XLR
Use of Write-combined mode
GUC parameter for NVM Logging
Accessing NVM at recovery
Wrap around of WAL buffer

Prototype architecture



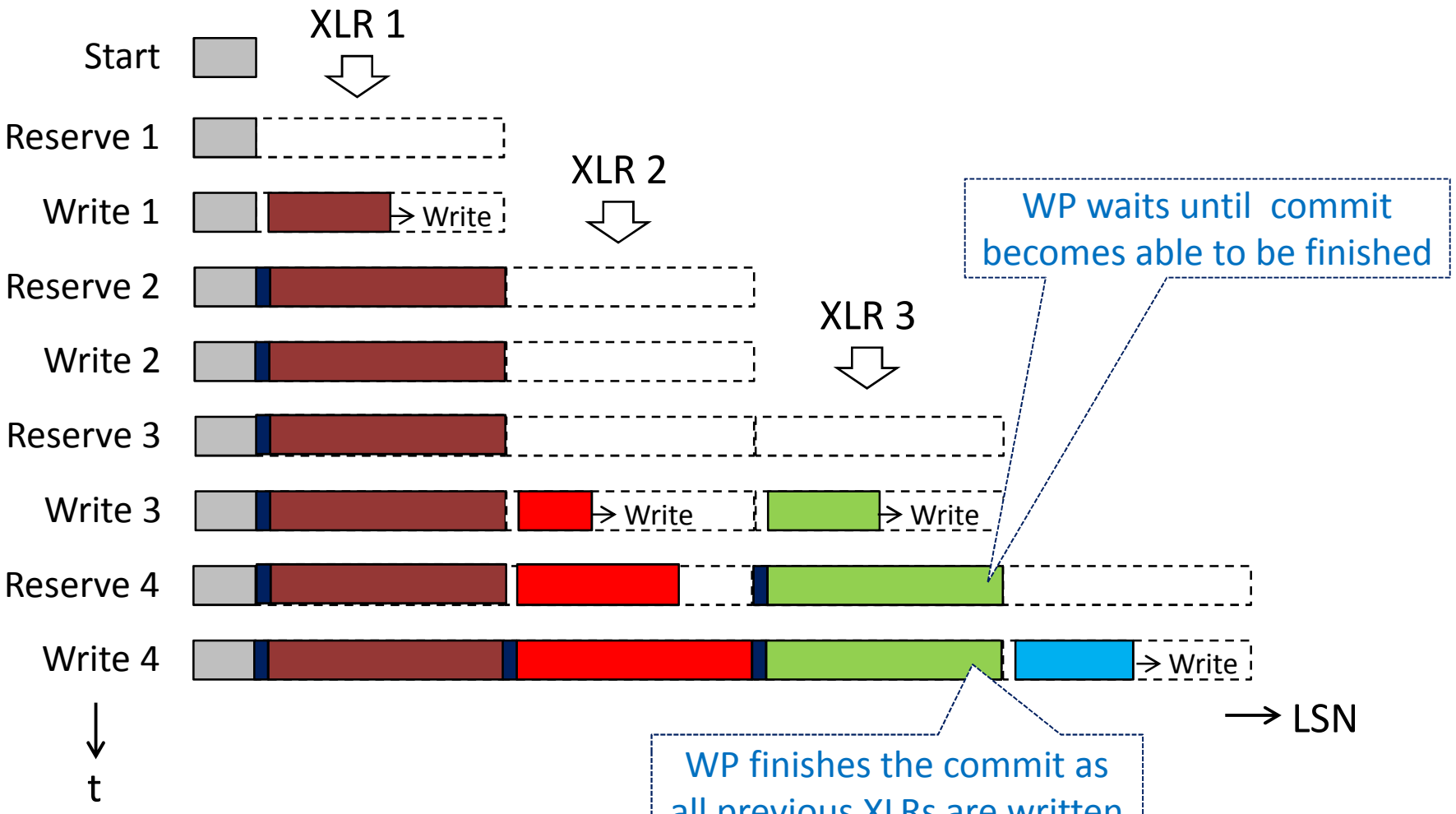
Preventing partial write



1. Move the tail pointer to reserve buffer area
2. Write the XLR data other than length field
   (At this point length field of XLR in XLog buffer is 0)
3. Write length field of the XLR

If XLog reader find a XLR whose length field is not zero,
all XLR data is written in the XLog buffer.

Preventing an unreachable XLR



Wait control

- A wait mechanism is already implemented in PostgreSQL

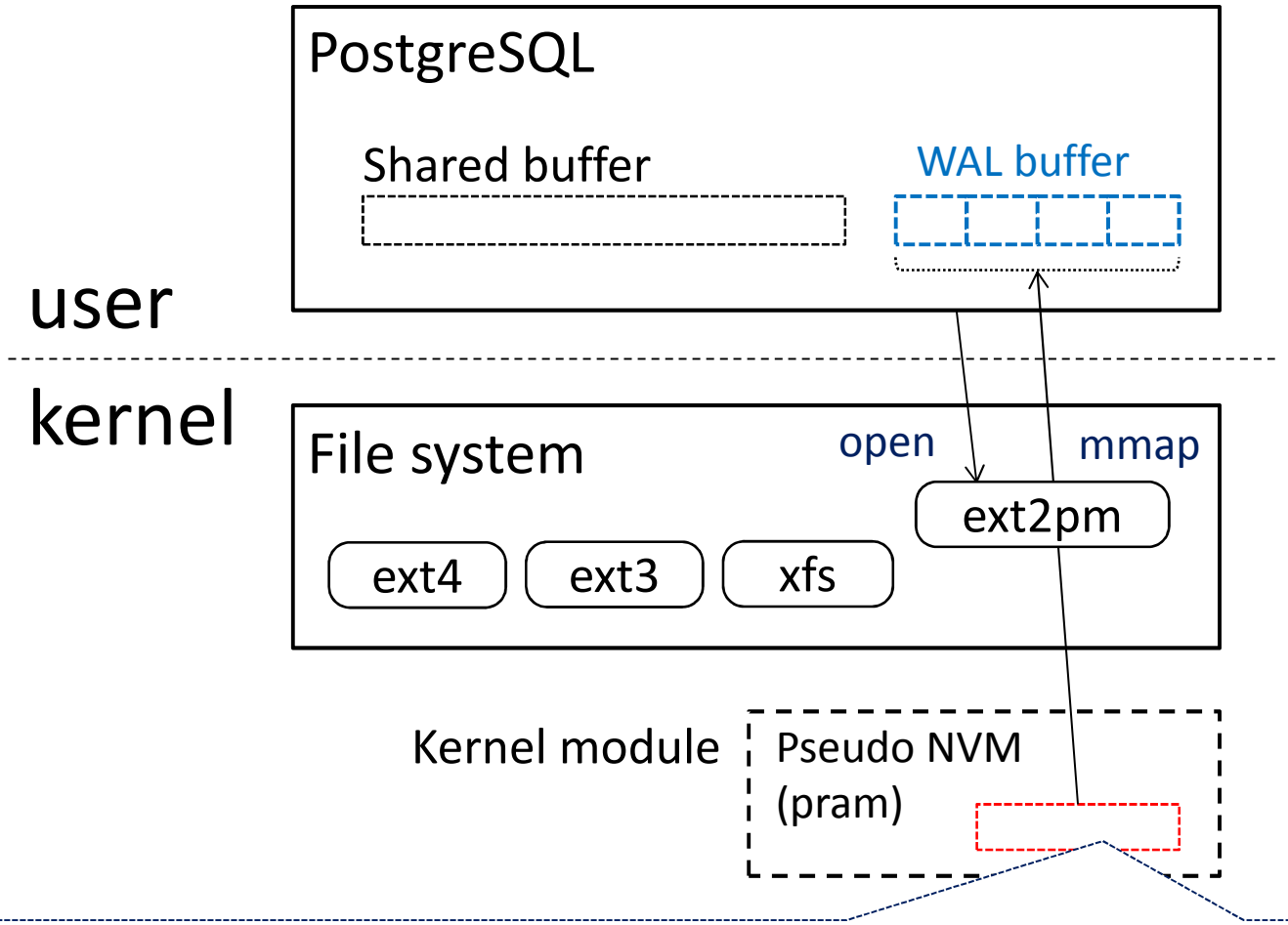
  static XLogRecPtr

  WaitXLogInsertionsToFinish(XLogRecPtr upto)

  If any XLR with a smaller LSN than the upto parameter is not finished to copy in the WAL buffer, the worker process sleeps until all of those XLRs are copied in the WAL buffer.

  NVM Logging implements the wait mechanism by using this function

Use of Write-combined mode



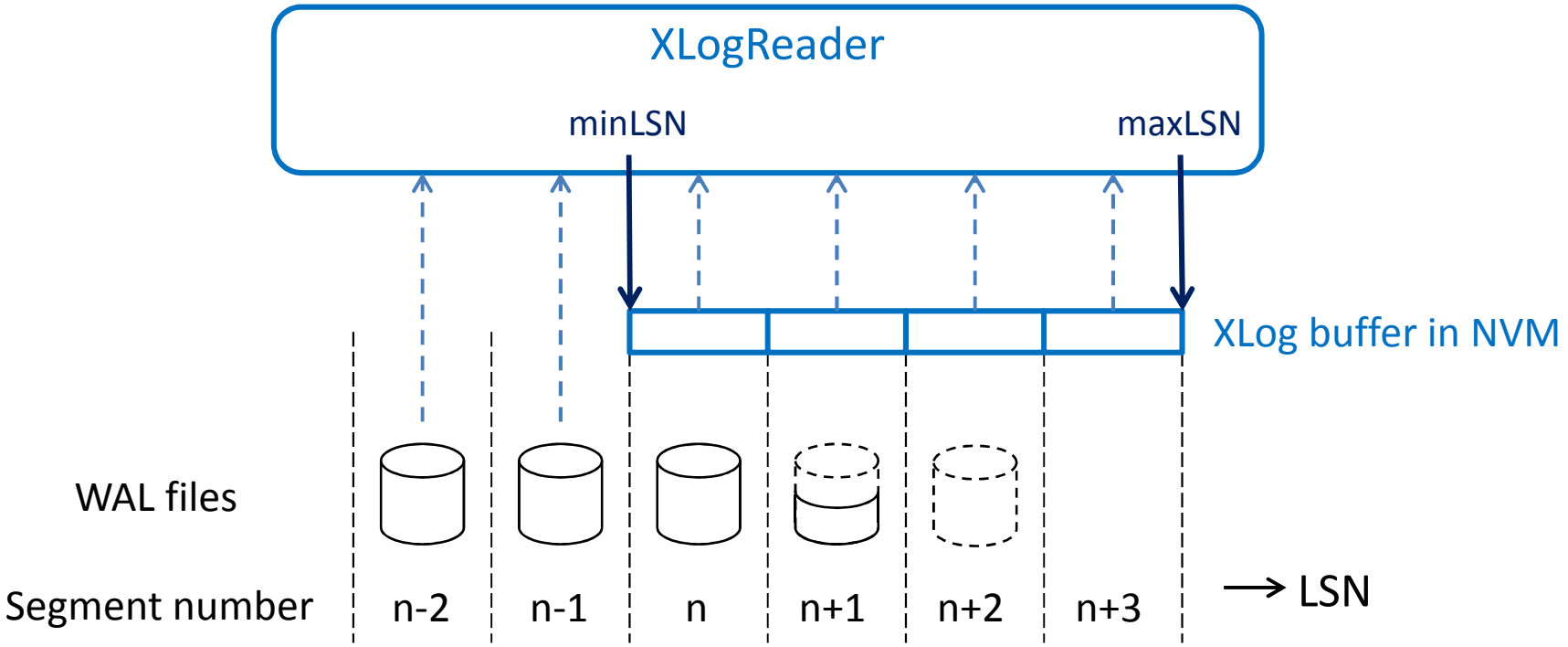
Write-combined mode, which is a variation of **write-through**, is set for the memory pages for pseudo NVM.

GUC parameter for NVM Logging

- NVM Logging is enabled through one GUC parameter (described in postgresql.conf)
  - PRAM_FILE_NAME = "*NVM File name*"

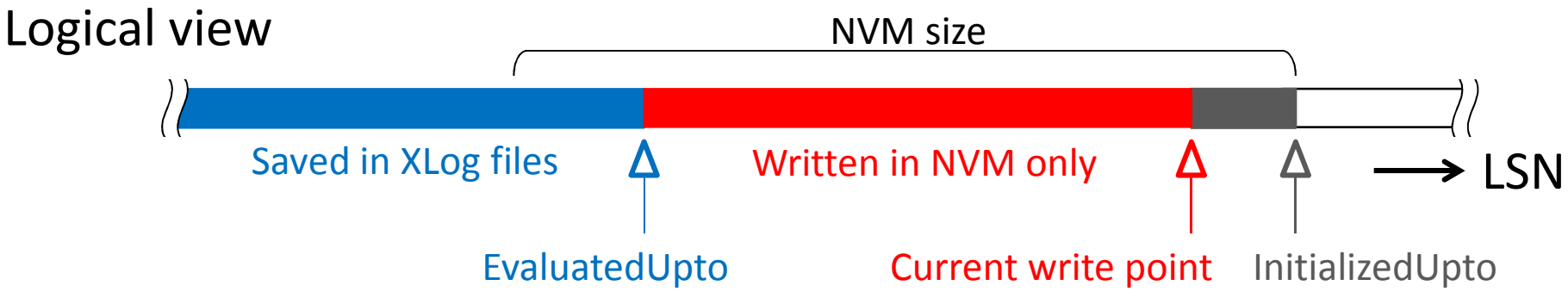
- When PRAM_FILE_NAME is set
  - XLOGShmemInit() invokes open() and mmap() to "*NVM File name*" and uses the memory area for WAL buffer
  - CopyXLogRecordToWAL() copies XLR in WAL buffer according to the procedure that prevents partial write

Accessing NVM at recovery

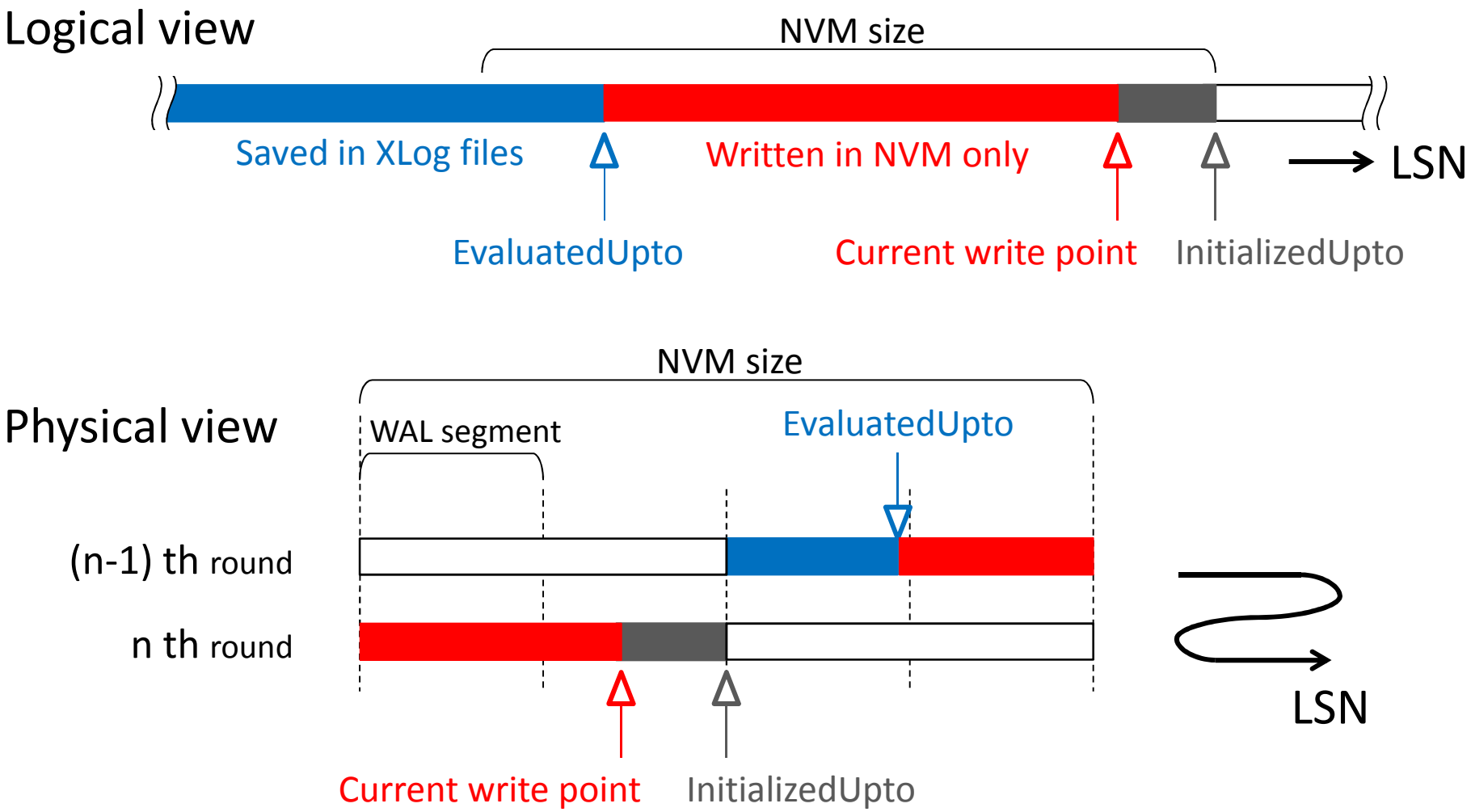


XLogReader accesses NVM XLog buffer to obtain XLog records whose LSN is between minLSN and maxLSN

Wrap around of WAL buffer



Wrap around of WAL buffer



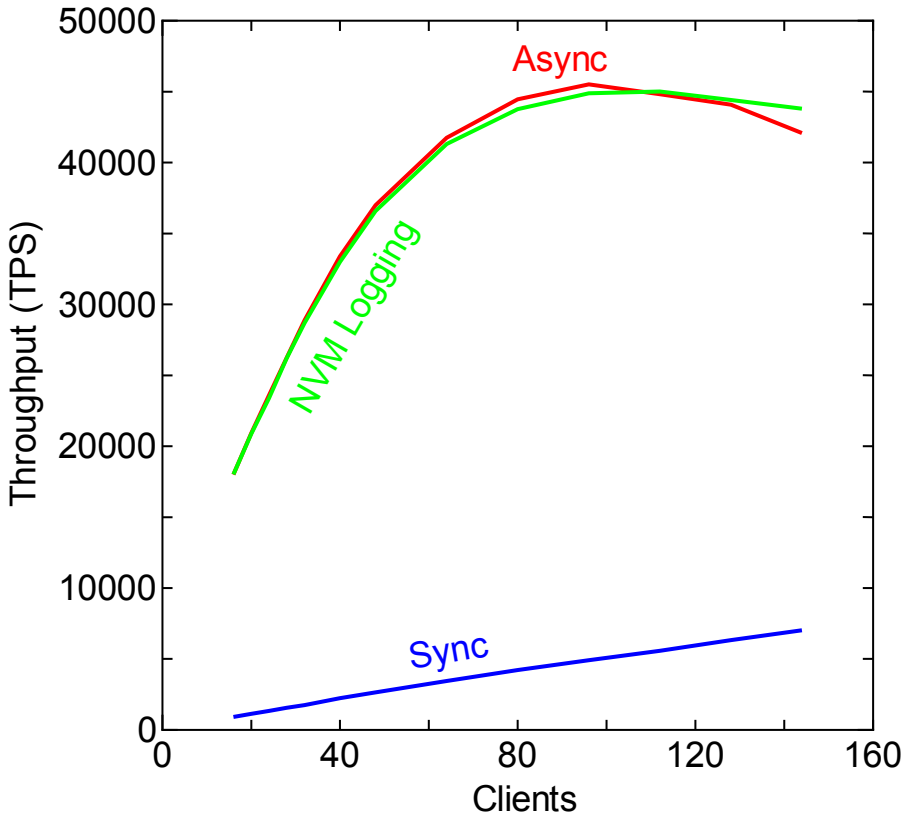
Evaluation

- Experimental Setup
- Performance
  - PGBENCH
  - DBT-2
- Durability
  - Durability test
  - Result
- Write amplification Reduction

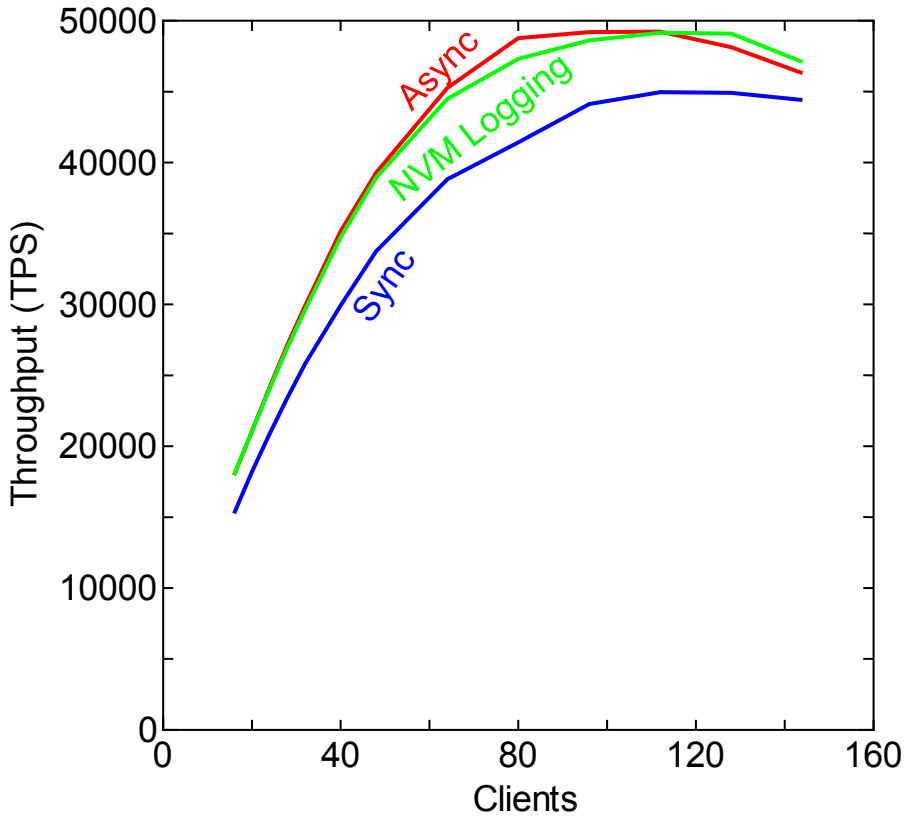
Experimental setup

- DB server
  - CPU: E5-2650 v2 x 2 (16 cores)
  - Memory: 64GB
  - Storage
    - RAID0: 200GB SSD x 2 for data
    - RAID0: 1TB ATA HD x 4 for WAL
- Client
  - CPU: E7420 x 4 (16 cores)
  - Memory: 8GB
- Network
  - GB ether x 1

PGBENCH Performance

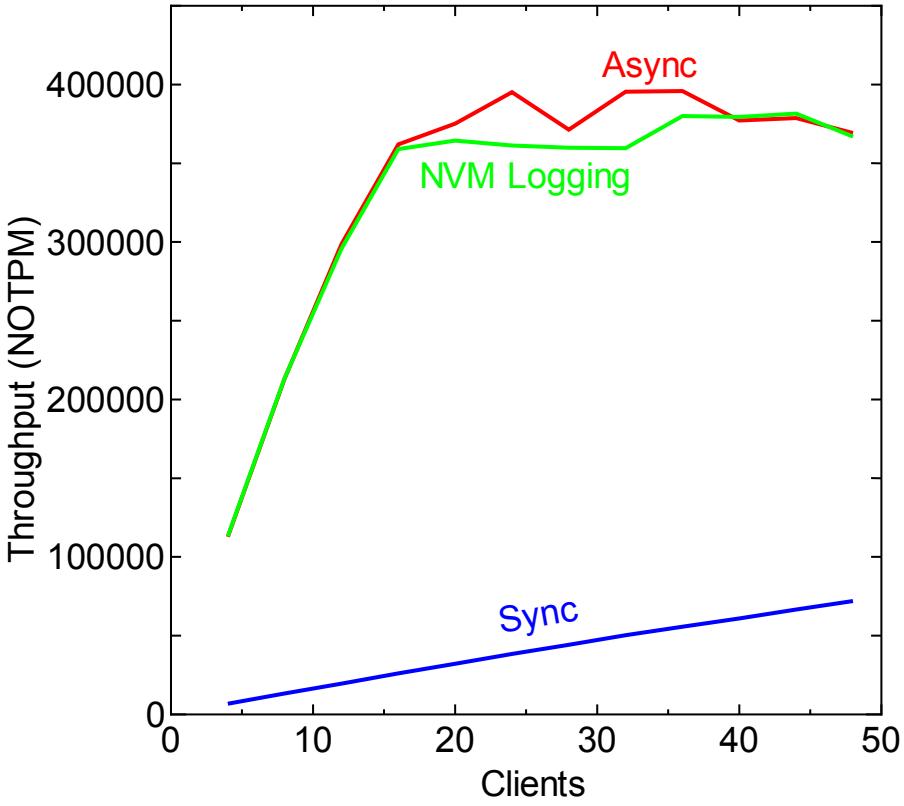


Disk-drive cache off

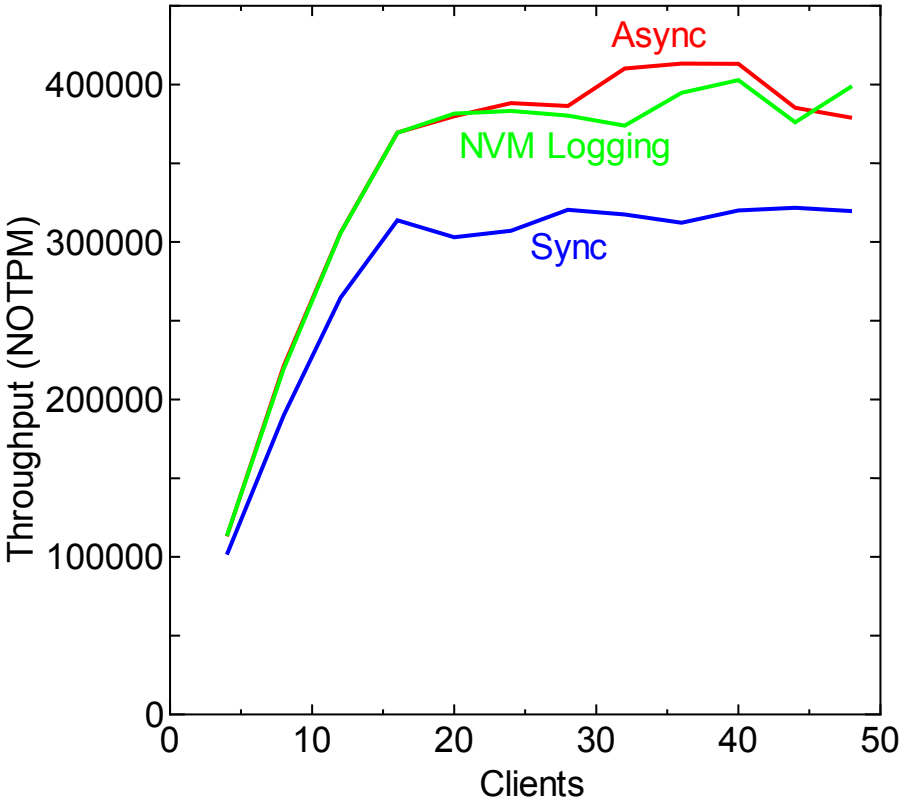


Disk-drive cache on

DBT-2 Performance



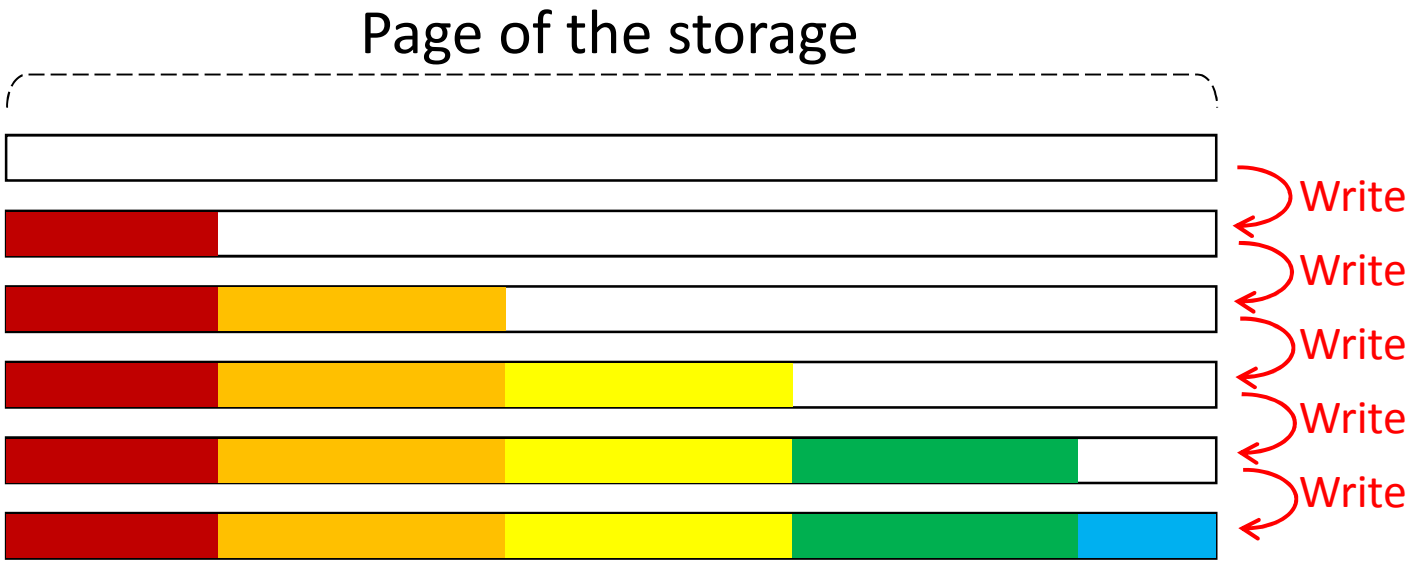
Disk-drive cache off



Disk-drive cache on

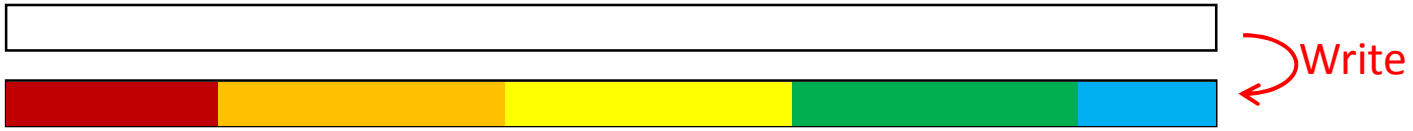
Write amplification Reduction

Sync. commit



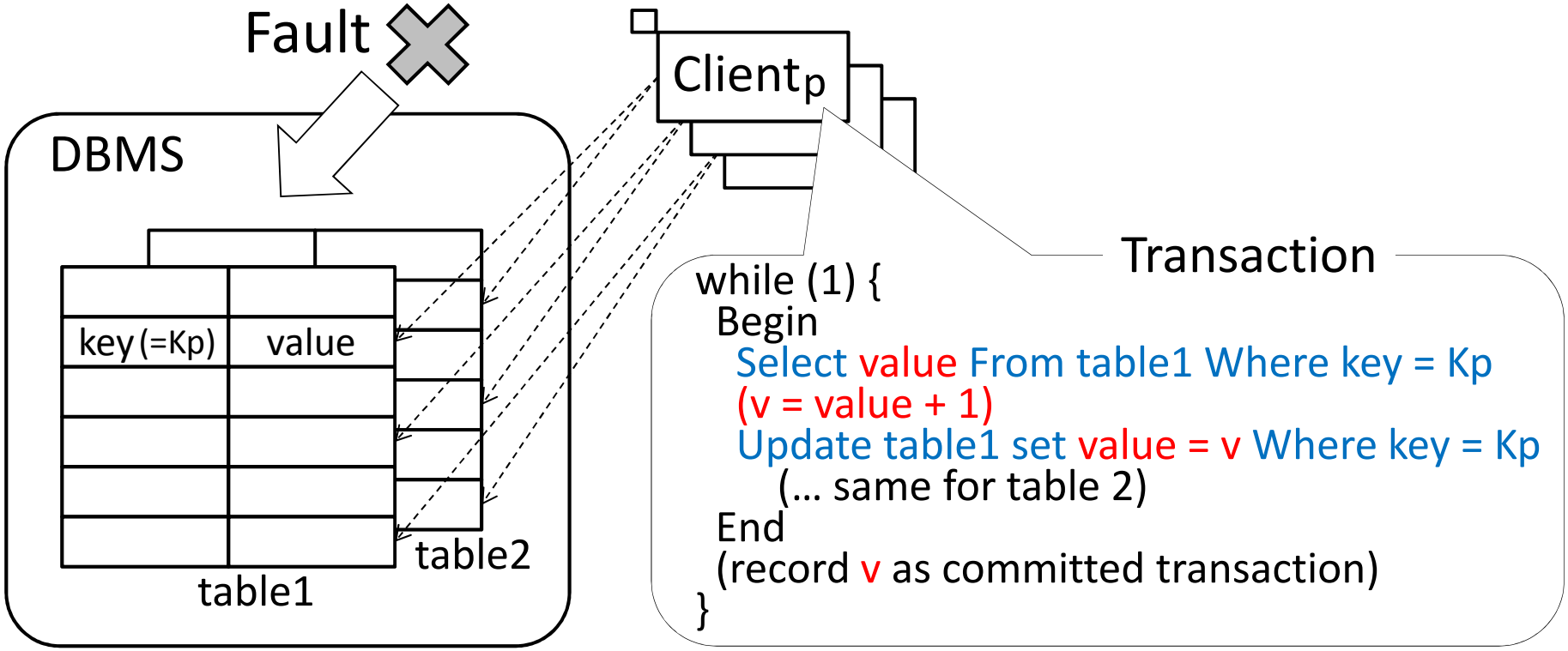
The tail WAL block is written in at every commit

NVM Logging



The tail WAL block is written in once

Durability test



After the recovery, durability is examined by checking whether the value of table1 and table2 is equal and value is equal to or greater than v that each client recorded as the result of the last transaction.

Results

- The results were just what we expected

  - Durability is ensured
    - Sync. commit, NVM Logging
  - Durability is not ensured
    - Async. commit

Technical trends

NVDIMM for DB servers
Programming support for NVM

NVDIMM in DB Servers

- NVDIMM-N Standardization
  - JEDEC Hybrid Memory Task Group
  - SNIA NVDIMM SIG

- Server product: HP ProLiant XL230a Server
  - Up to 2 Intel® Xeon® E5-2600 v3 Series, 6/8/10/12/14/16 Cores (16)
  - DDR4, (512GB max), support for NVDIMM
  - … http://community.hpe.com/t5/Servers-The-Right-Compute/Address-your-Compute-needs-with-HP-ProLiant-Gen9/ba-p/6794213#.VybkulK3GA9

DB server with NVDIMM is just around the corner!!

Programming support for NVM

- pmem.io
  - The Linux NVM Library builds on the **Direct Access** (DAX) changes under development in Linux.
  - This project focuses specifically on how persistent memory is exposed to server-class applications which will explicitly manage the placement of data among the three tiers (volatile memory, persistent memory, and storage).

http://pmem.io/

Conclusion

- NVM is becoming commodity
  - NVDIMM is already shipped as a product
  - Servers began to equipped with NVDIMM

- Benefits of NVM Logging
  - Performance improvement — Almost the same as async. commit
  - Durability ensurance — Similar to sync. commit
  - Write amplitude reduction — Good for SSD lifetime

Future work

- Bring to a state acceptable for the mainline
  - Cope with standard for NVM access
    - libpmem is a promising candidate
  - Check the operation in using real NVM

That’s it

Thank you for listening