LEAST : Logging Exploiting A Split snapshoT

연세대학교 컴퓨터과학과 성한승

2020년 6월

SW STAR LAB
Software Technology Advanced Research

과제명: IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발

과제번호: 2017-0-00477

과학기술정보통신부 Ministry of Science and ICT
연세대학교 YONSEI UNIVERSITY
IITP 정보통신기술진흥센터 Institute for Information & communications Technology Promotion

Contents

1. Introduction
2. Background
3. Motivation
4. The design of LEAST
5. Evaluation
6. Conclusion

1. Introduction

- In-Memory Key-Value Store
- Redis Persistence Method (Basic)
  - Redis Database (RDB)
  - Append-Only File (AOF)

1 Introduction

In-Memory Key-Value Store

ex) **Redis**, Memcached,
Apache Ignite, RAMCloud

Key	Value
key1	HS
key2	WG

Store data as key-value pair

Store all dataset in memory

High data processing performance

Risk of data loss

REmote DIctionary Server

redis

Provide various data structure
String, List, Set, Hash …

Single thread-based process

Support cluster and partitioning

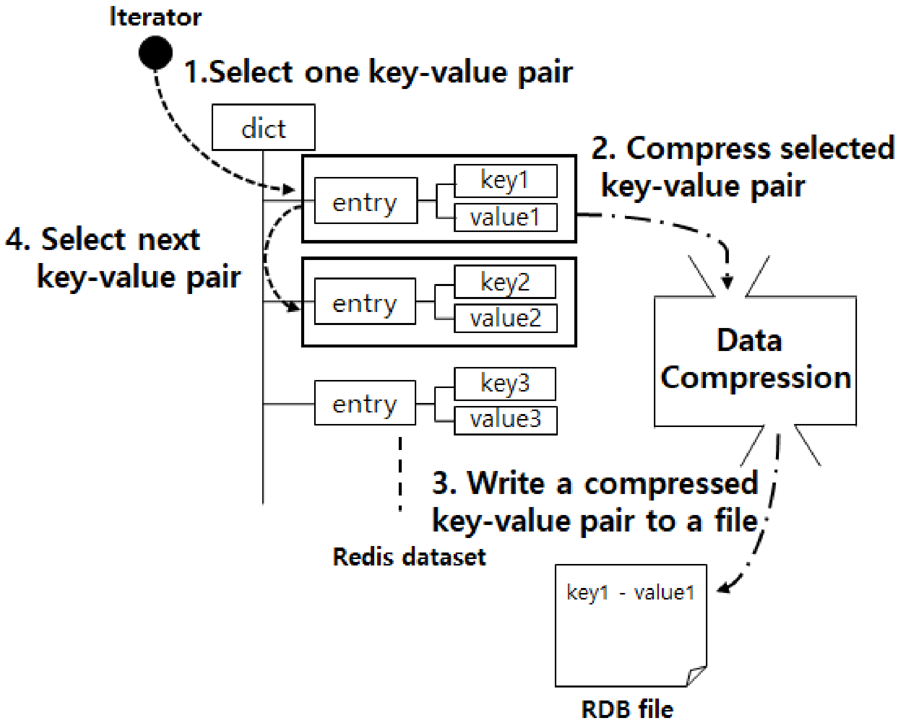
Provide persistence methods to preserve data

1 Introduction

- **Redis Database (RDB)**
  - Creates a **snapshot** of the data stored up to a certain point-in-time at regular intervals

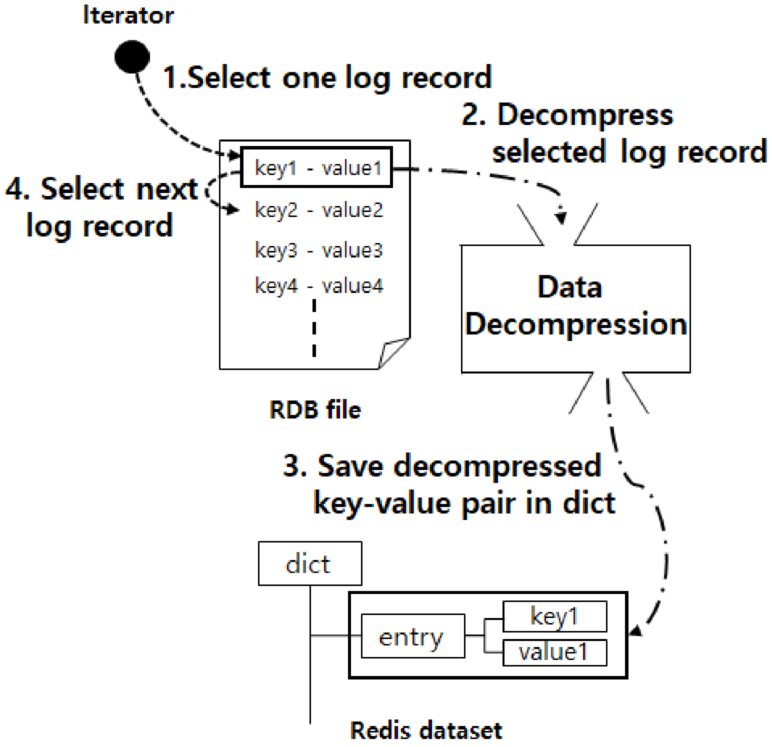
    ☺ Small log file size, Fast backup & recovery

    ☹ Risk of data loss …



Iterator
1.Select one key-value pair
dict
entry
key1
value1
entry
key2
value2
entry
key3
value3
2. Compress selected key-value pair
4. Select next key-value pair
Data Compression
3. Write a compressed key-value pair to a file
Redis dataset
key1 - value1
RDB file

RDB logging operation process



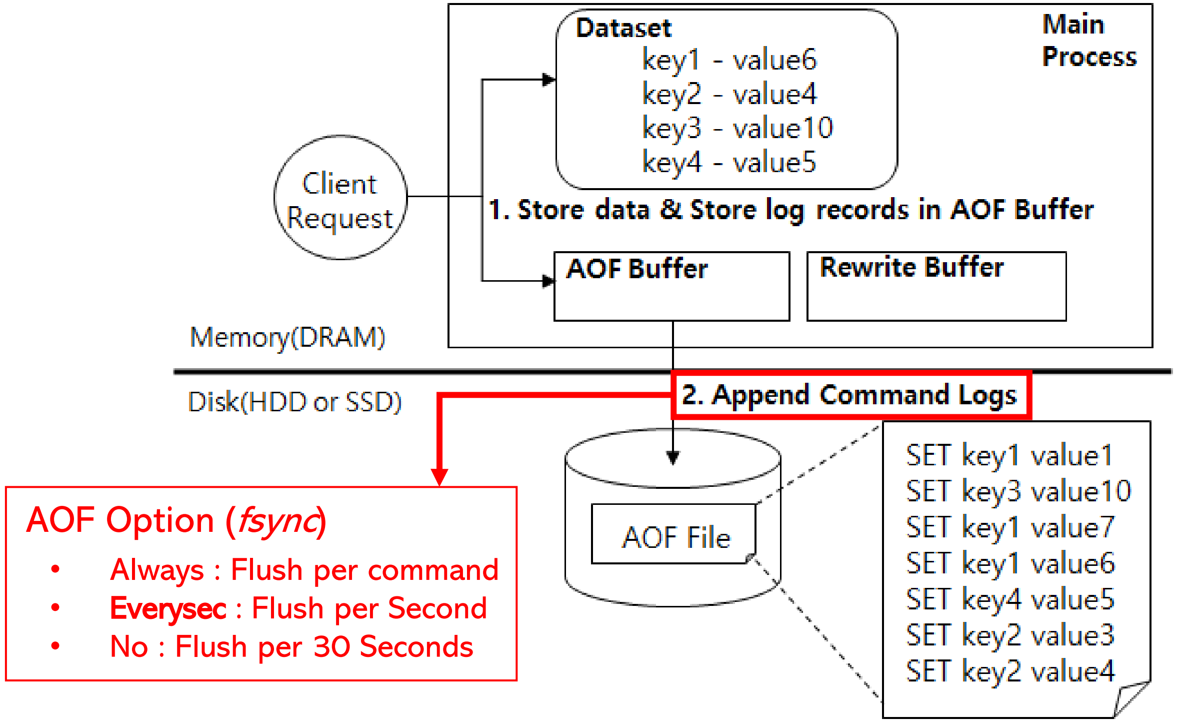
RDB recovery operation process

1 Introduction

- **Append-Only File (AOF)**
  - Writes a log record in the AOF log file each time data is **inserted**, **modified**, or **deleted**

☺ Ensure data persistence

☹ Large log file size, Slow performance & recovery …

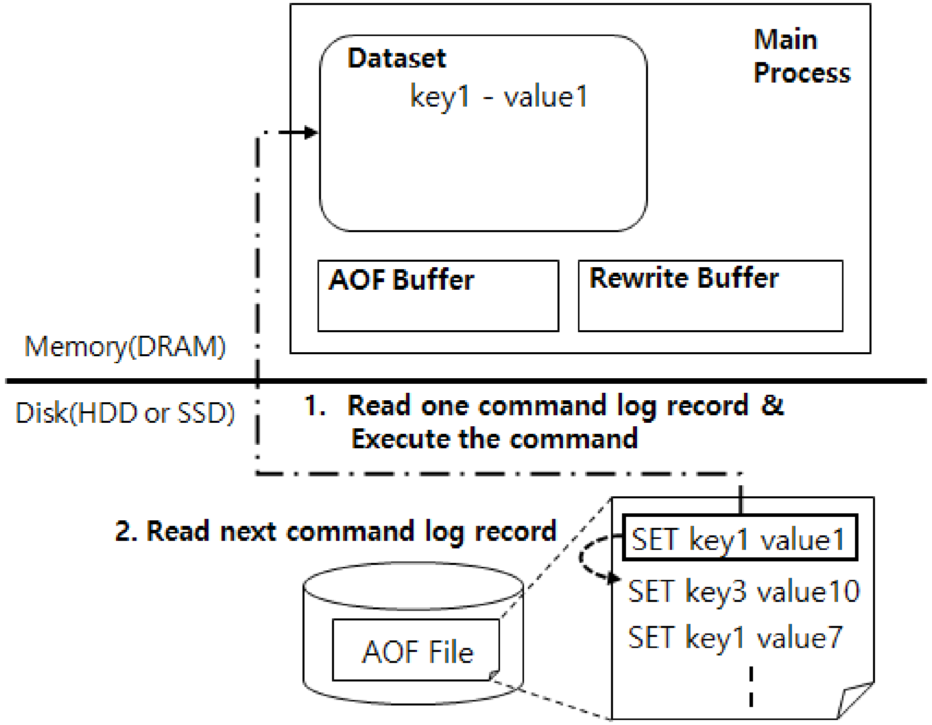


```
SET key1 value1
SET key3 value10
SET key1 value7
SET key1 value6
SET key4 value5
SET key2 value3
SET key2 value4
```

AOF Option (*fsync*)

- Always : Flush per command
- **Everysec** : Flush per Second
- No : Flush per 30 Seconds

AOF logging operation process



```
SET key1 value1
SET key3 value10
SET key1 value7
```

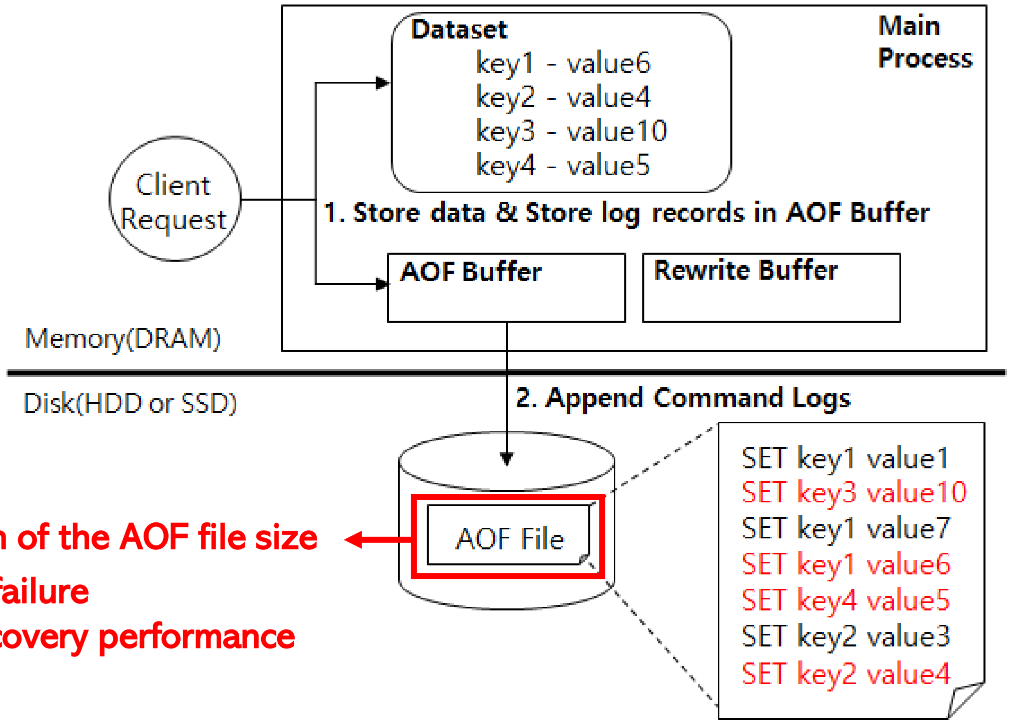
AOF recovery operation process

2. Background

- Redis Persistence Method (Advanced)
  - AOF Rewrite
  - AOF-USE-RDB-PREAMBLE

2 Background

- **AOF Rewrite**
  - Reduce the AOF file size by preserving only the log records of the final state of the current dataset



Excessive growth of the AOF file size

- System failure
- Poor recovery performance

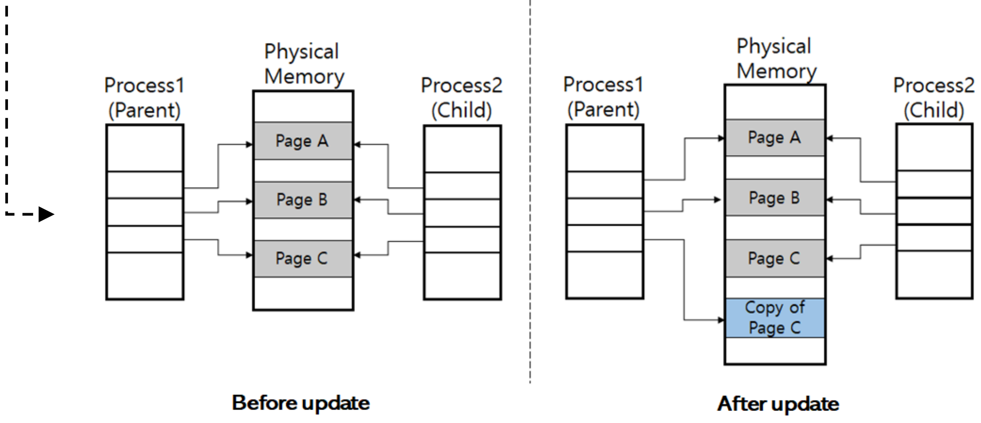
AOF logging operation process

AOF Rewrite trigger condition

- AOF file size > threshold (64 MB)

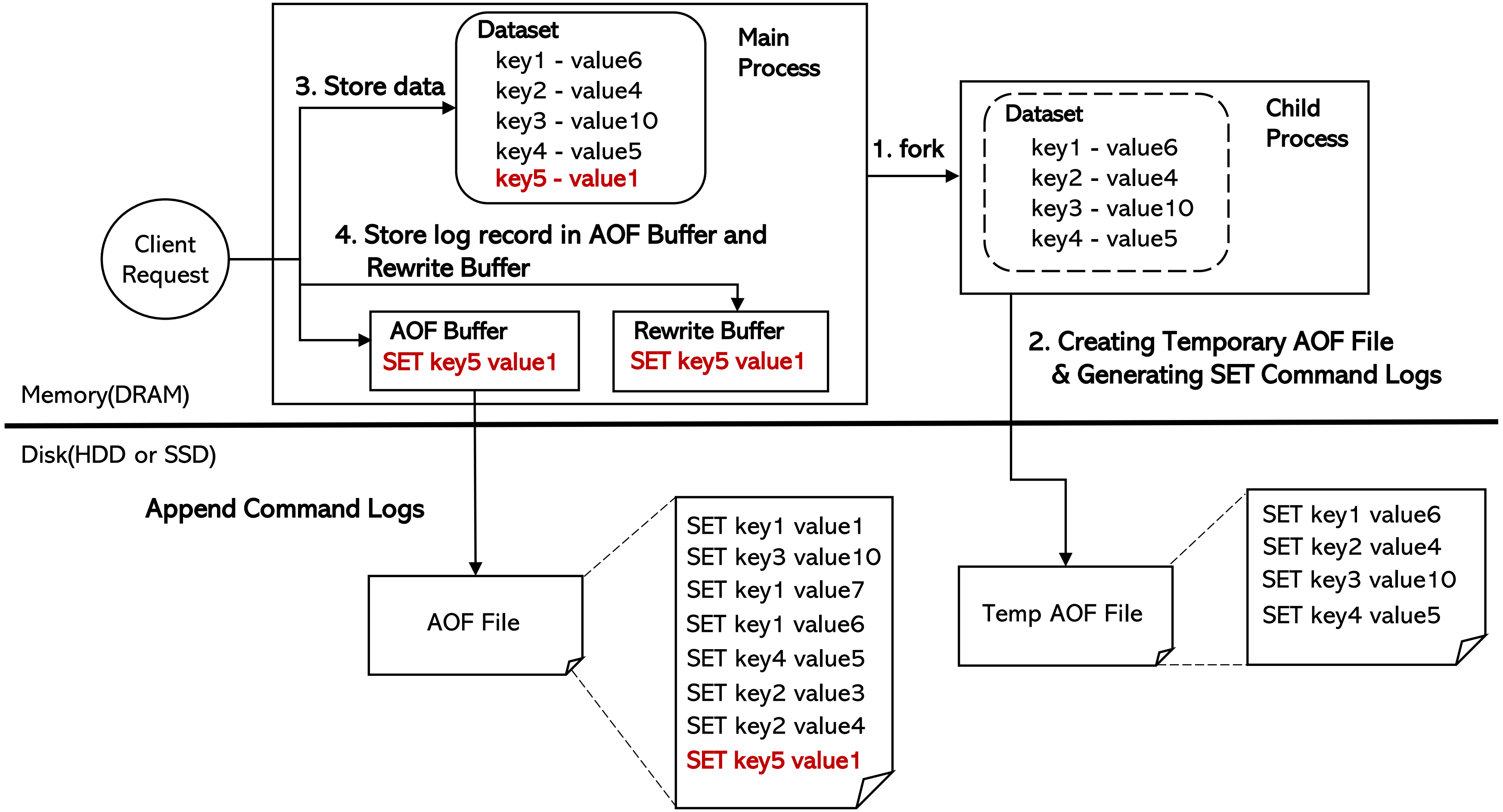
Redis uses ***fork*** to create a child process

- Main process: execute client requests
- Child process: reconstruct AOF file



Copy-on-write

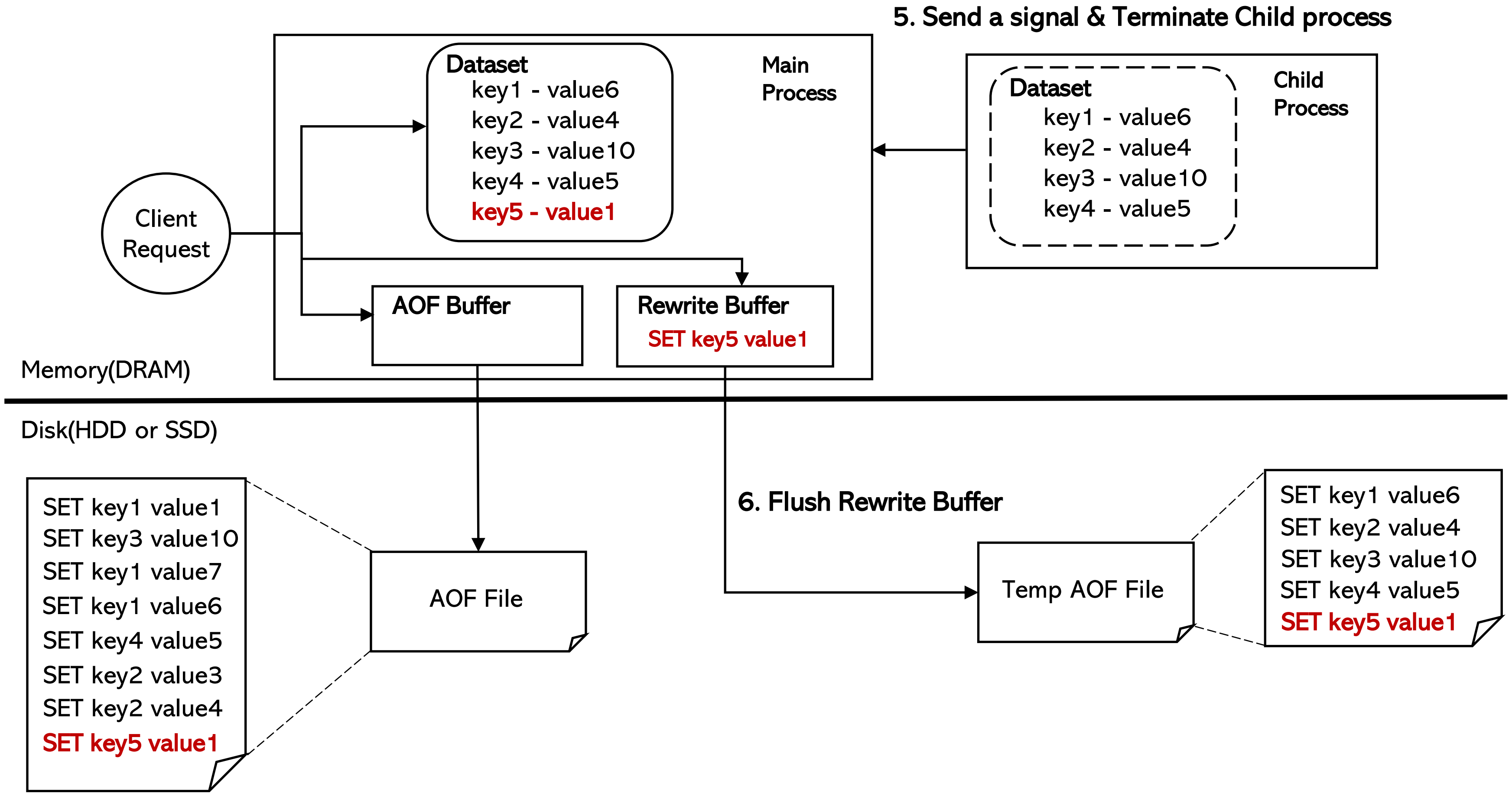
- **AOF Rewrite**



```
SET key1 value1
SET key3 value10
SET key1 value7
SET key1 value6
SET key4 value5
SET key2 value3
SET key2 value4
SET key5 value1
```

```
SET key1 value6
SET key2 value4
SET key3 value10
SET key4 value5
```

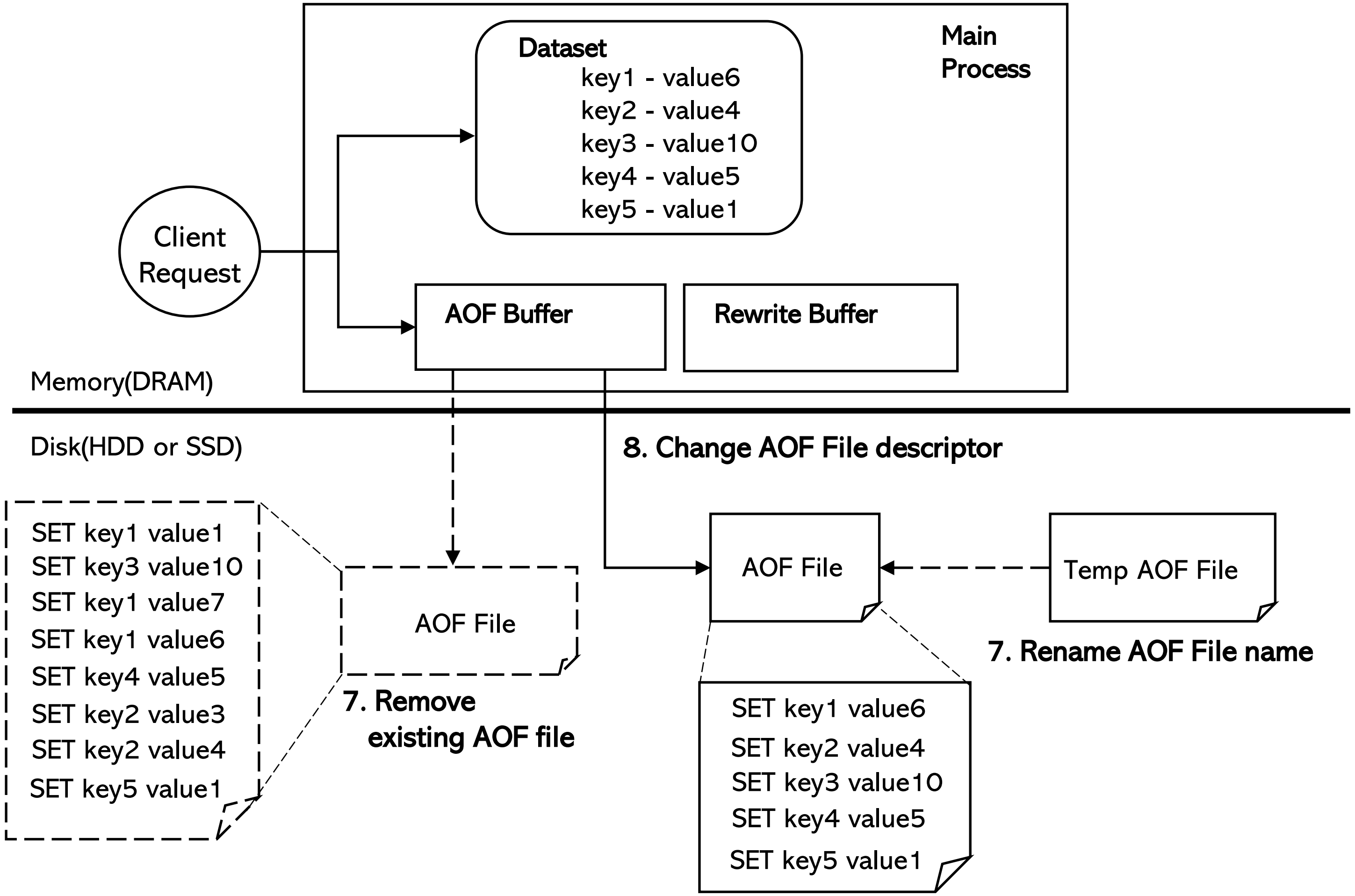
- **AOF Rewrite**



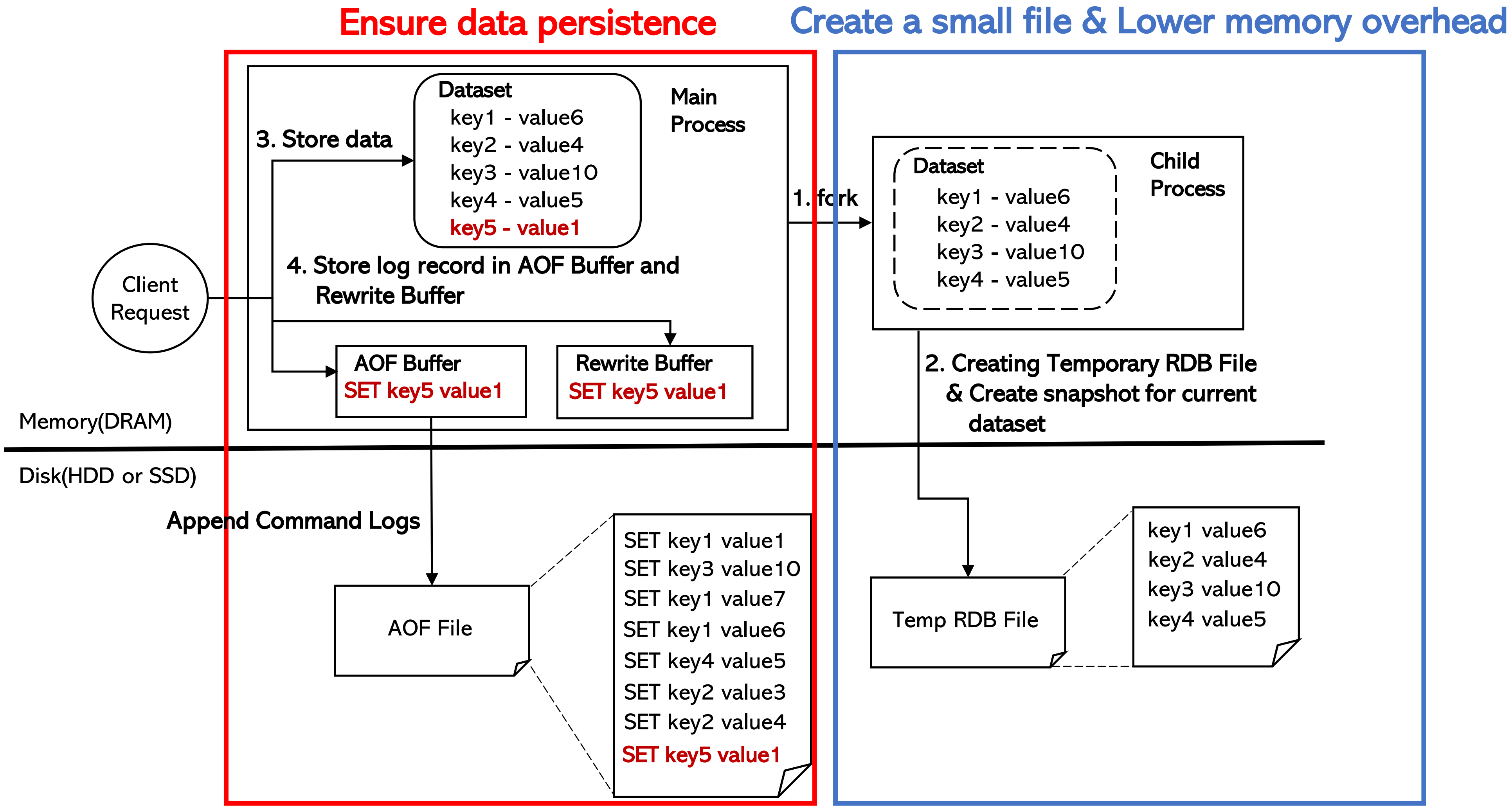
5. Send a signal & Terminate Child process

6. Flush Rewrite Buffer

- **AOF Rewrite**

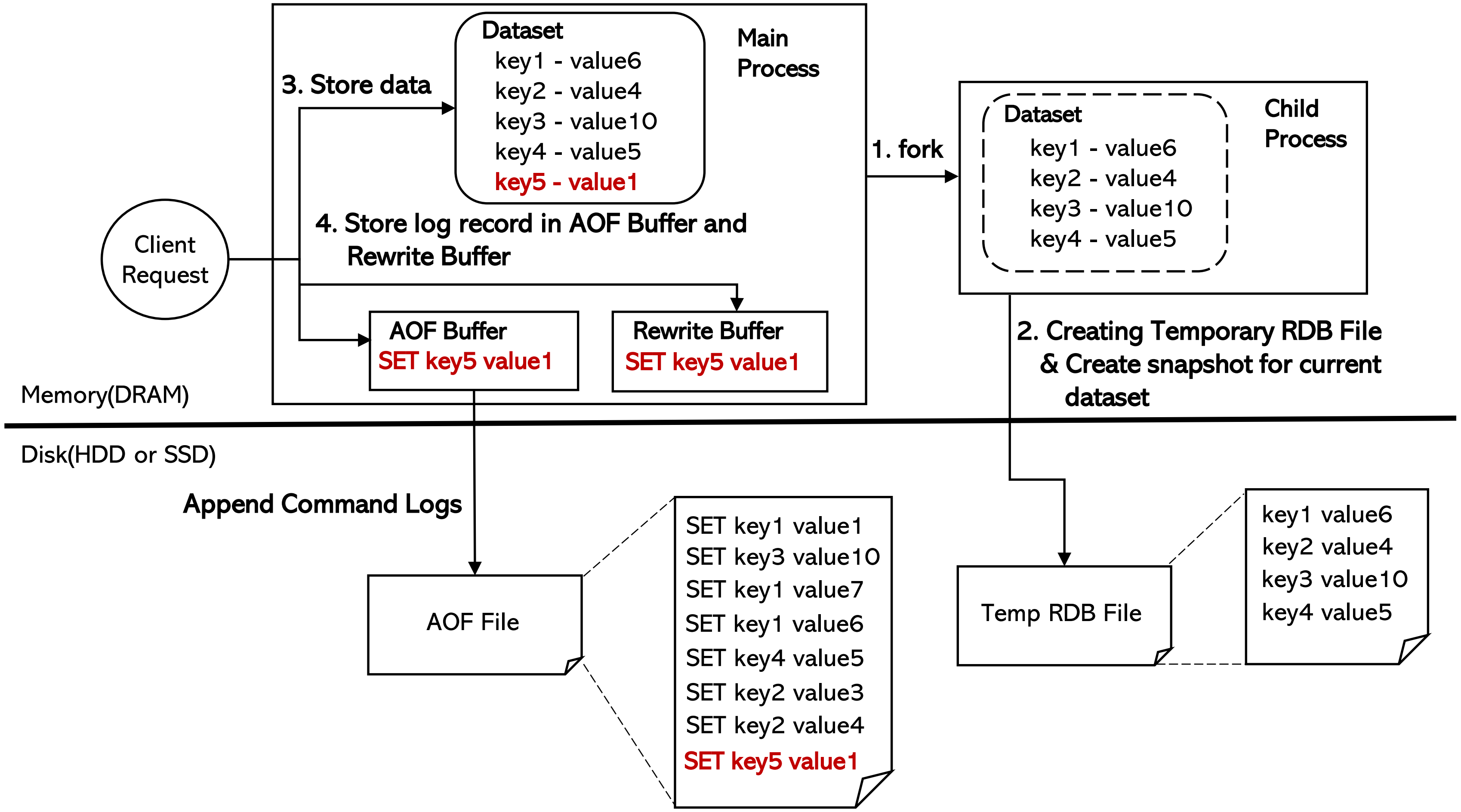

Dataset
key1 - value6
key2 - value4
key3 - value10
key4 - value5
key5 - value1
Main Process
Client Request
AOF Buffer
Rewrite Buffer
Memory(DRAM)
Disk(HDD or SSD)
8. Change AOF File descriptor
SET key1 value1
SET key3 value10
SET key1 value7
SET key1 value6
SET key4 value5
SET key2 value3
SET key2 value4
SET key5 value1
AOF File
7. Remove existing AOF file
AOF File
Temp AOF File
7. Rename AOF File name
SET key1 value6
SET key2 value4
SET key3 value10
SET key4 value5
SET key5 value1

- **AOF-USE-RDB-PREAMBLE**

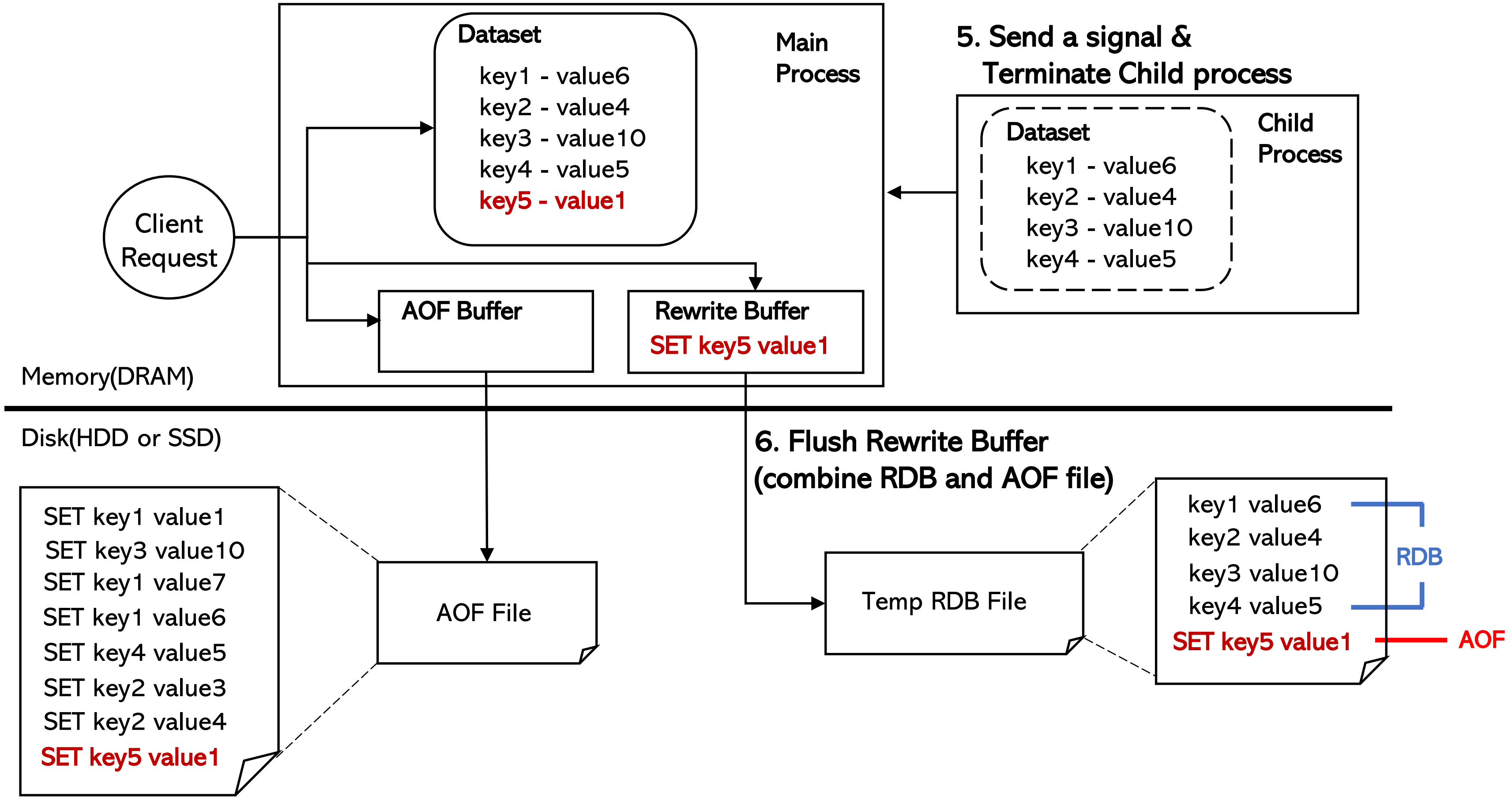


- The AOF-USE-RDB-PREAMBLE method is a persistence method that uses a mixture of AOF and RDB

- **AOF-USE-RDB-PREAMBLE**

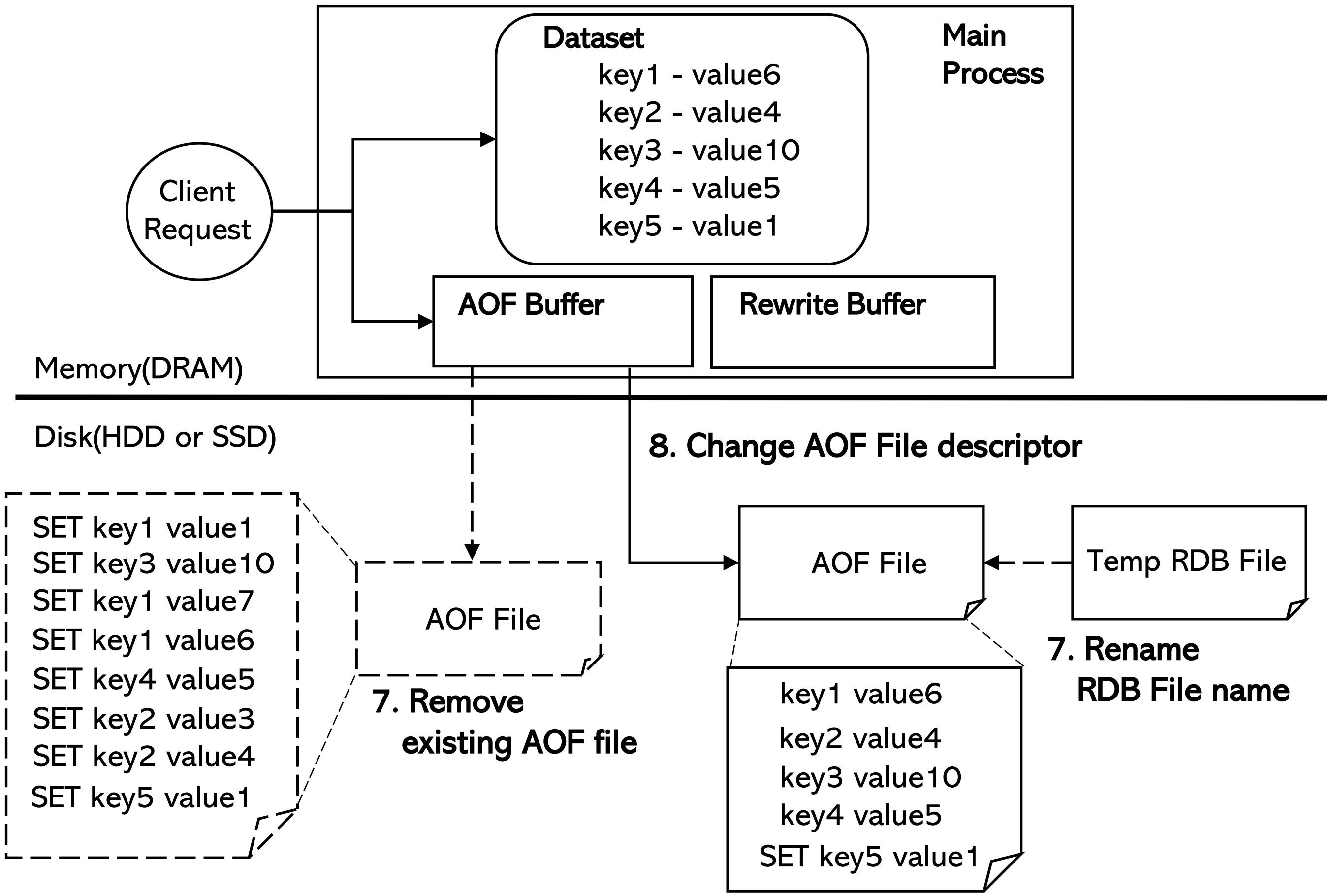


- **AOF-USE-RDB-PREAMBLE**



Background

- **AOF-USE-RDB-PREAMBLE**

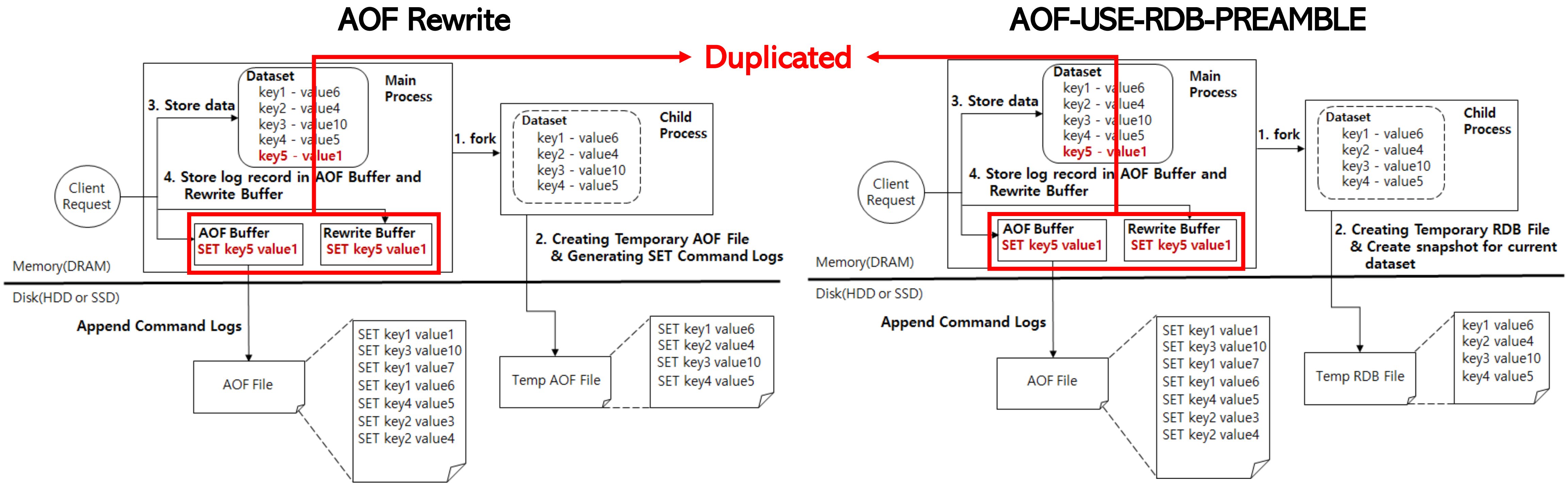


3. Motivation

- Memory Overhead
- Throughput Degradation
- Logging Overhead Test
  - AOF Rewrite
  - AOF-USE-RDB-PREAMBLE

Motivation

- **Memory Overhead**



```
SET key1 value1
SET key3 value10
SET key1 value7
SET key1 value6
SET key4 value5
SET key2 value3
SET key2 value4
```

```
SET key1 value6
SET key2 value4
SET key3 value10
SET key4 value5
```

```
SET key1 value1
SET key3 value10
SET key1 value7
SET key1 value6
SET key4 value5
SET key2 value3
SET key2 value4
```

```
key1 value6
key2 value4
key3 value10
key4 value5
```

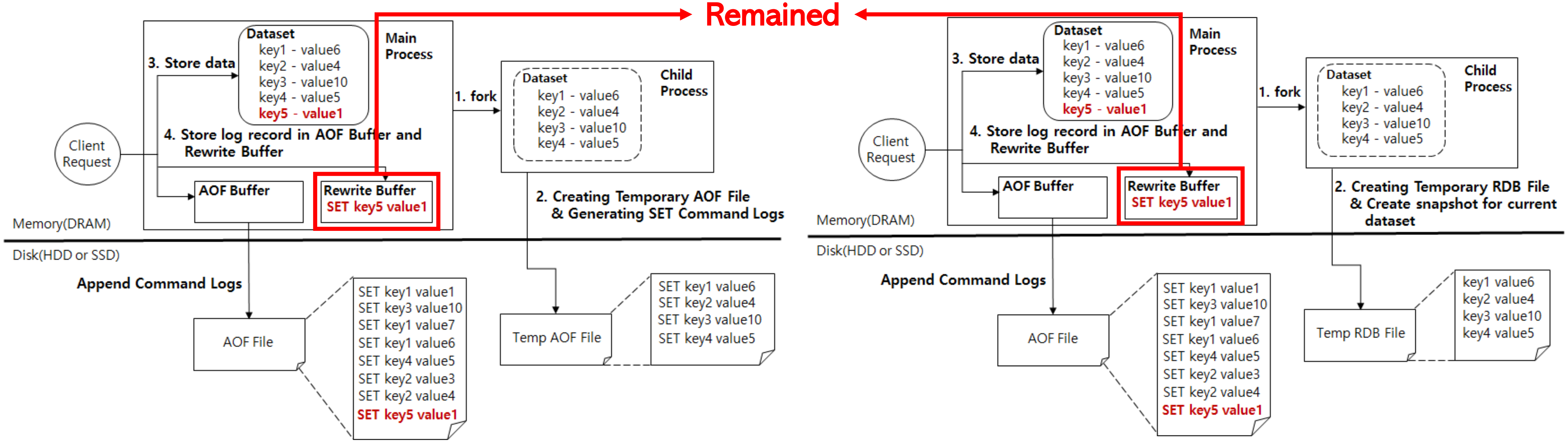
- Log records for the newly requested command are stored in the AOF Buffer and Rewrite Buffer

➔ Increase memory usage

3 Motivation

- **Memory Overhead**

AOF Rewrite **AOF-USE-RDB-PREAMBLE**



Remained

AOF Rewrite:

Client Request

3. Store data

Dataset
key1 - value6
key2 - value4
key3 - value10
key4 - value5
key5 - value1

Main Process

4. Store log record in AOF Buffer and Rewrite Buffer

AOF Buffer

Rewrite Buffer
SET key5 value1

1. fork

Child Process

Dataset
key1 - value6
key2 - value4
key3 - value10
key4 - value5

2. Creating Temporary AOF File & Generating SET Command Logs

Memory(DRAM)

Disk(HDD or SSD)

Append Command Logs

AOF File

SET key1 value1
SET key3 value10
SET key1 value7
SET key1 value6
SET key4 value5
SET key2 value3
SET key2 value4
SET key5 value1

Temp AOF File

SET key1 value6
SET key2 value4
SET key3 value10
SET key4 value5

AOF-USE-RDB-PREAMBLE:

Client Request

3. Store data

Dataset
key1 - value6
key2 - value4
key3 - value10
key4 - value5
key5 - value1

Main Process

4. Store log record in AOF Buffer and Rewrite Buffer

AOF Buffer

Rewrite Buffer
SET key5 value1

1. fork

Child Process

Dataset
key1 - value6
key2 - value4
key3 - value10
key4 - value5

2. Creating Temporary RDB File & Create snapshot for current dataset

Memory(DRAM)

Disk(HDD or SSD)

Append Command Logs

AOF File

SET key1 value1
SET key3 value10
SET key1 value7
SET key1 value6
SET key4 value5
SET key2 value3
SET key2 value4
SET key5 value1

Temp RDB File

key1 value6
key2 value4
key3 value10
key4 value5

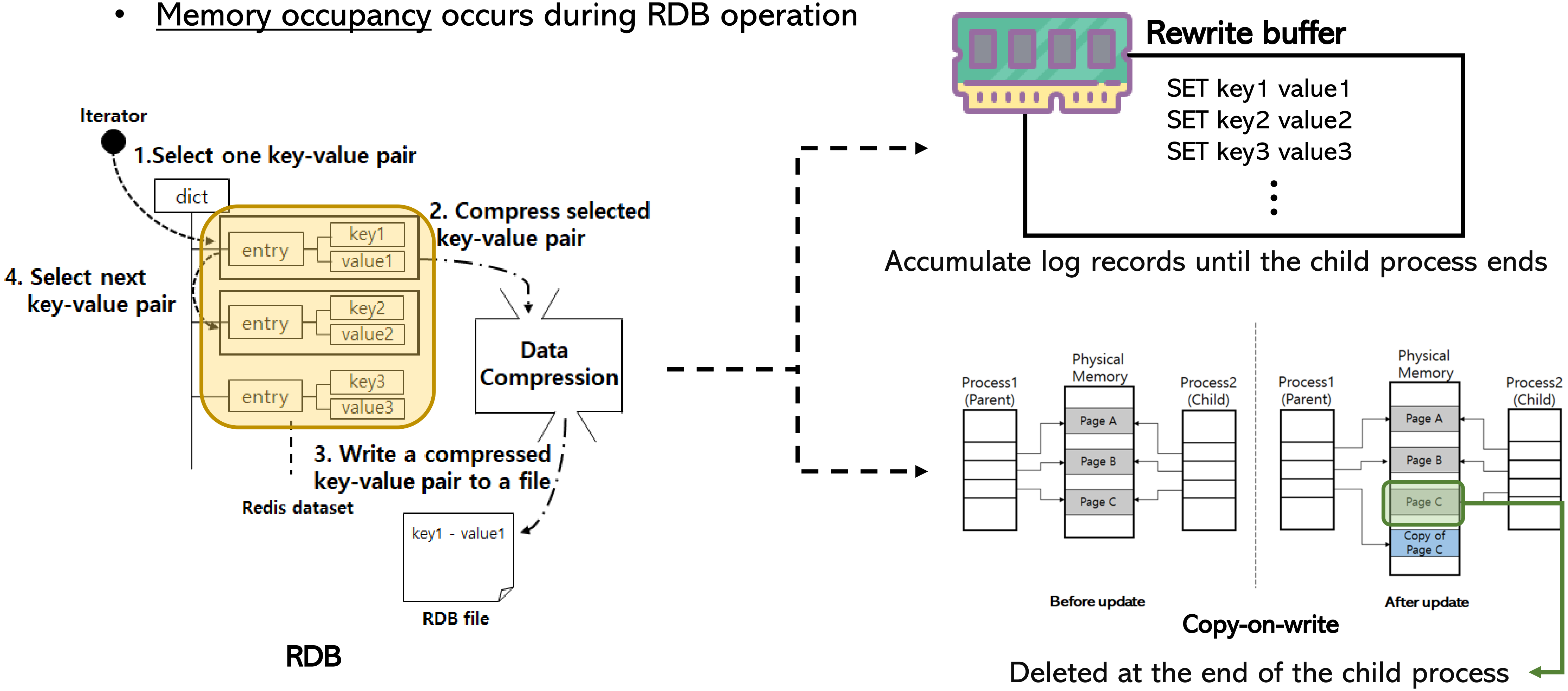
- Log records stored in the Rewrite Buffer are remained until the child process is terminated

➔ The state of increased memory usage is continued

- AOF Rewrite and AOF-USE-RDB-PREAMBLE may result in **out-of-memory** and **system shutdown** issues

3 Motivation

- **Memory Overhead**
  - Stored key-value pairs affect RDB generation time
  - Memory occupancy occurs during RDB operation



RDB

Rewrite buffer

SET key1 value1
SET key2 value2
SET key3 value3
⋮

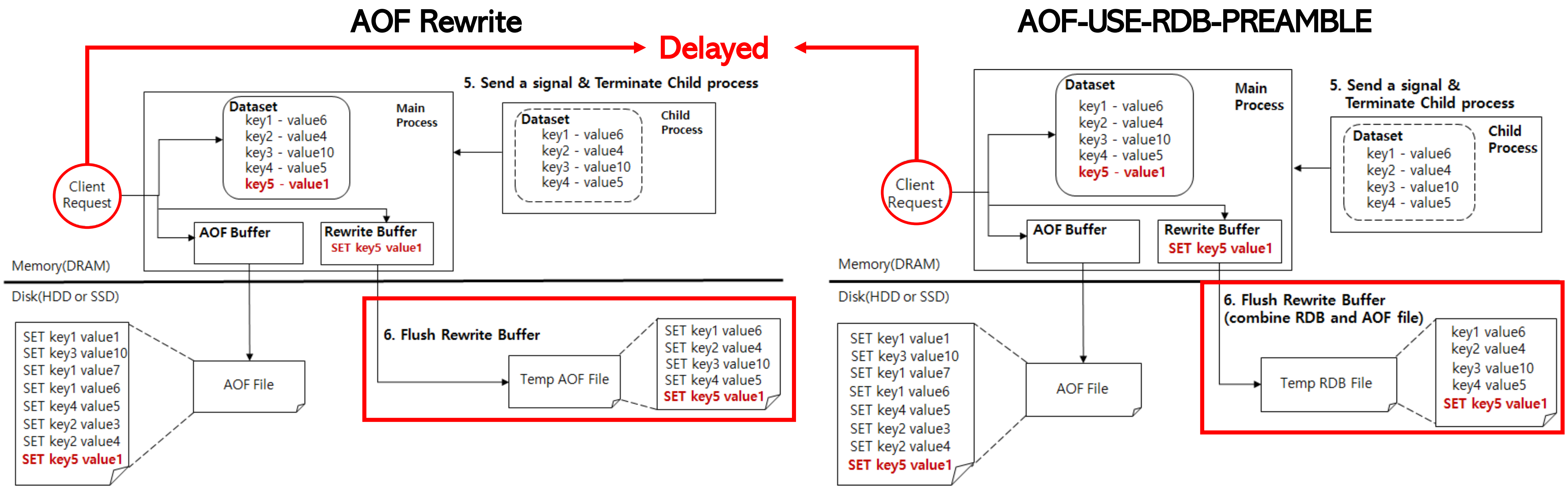
Accumulate log records until the child process ends

Copy-on-write

Deleted at the end of the child process

3 Motivation

- **Throughput Degradation**



AOF Rewrite

Delayed

AOF-USE-RDB-PREAMBLE

Client Request

Dataset
key1 - value6
key2 - value4
key3 - value10
key4 - value5
key5 - value1

Main Process

5. Send a signal & Terminate Child process

Dataset
key1 - value6
key2 - value4
key3 - value10
key4 - value5

Child Process

AOF Buffer

Rewrite Buffer
SET key5 value1

Memory(DRAM)

Disk(HDD or SSD)

SET key1 value1
SET key3 value10
SET key1 value7
SET key1 value6
SET key4 value5
SET key2 value3
SET key2 value4
SET key5 value1

AOF File

6. Flush Rewrite Buffer

Temp AOF File

SET key1 value6
SET key2 value4
SET key3 value10
SET key4 value5
SET key5 value1

Client Request

Dataset
key1 - value6
key2 - value4
key3 - value10
key4 - value5
key5 - value1

Main Process

5. Send a signal & Terminate Child process

Dataset
key1 - value6
key2 - value4
key3 - value10
key4 - value5

Child Process

AOF Buffer

Rewrite Buffer
SET key5 value1

Memory(DRAM)

Disk(HDD or SSD)

SET key1 value1
SET key3 value10
SET key1 value7
SET key1 value6
SET key4 value5
SET key2 value3
SET key2 value4
SET key5 value1

AOF File

6. Flush Rewrite Buffer (combine RDB and AOF file)

Temp RDB File

key1 value6
key2 value4
key3 value10
key4 value5
SET key5 value1

- *Flush* operation incurs heavy disk I/O
- During a *Flush* operation, the requested command is delayed without execution

➔ Redis' data processing performance is degraded

- **Logging Overhead Test**

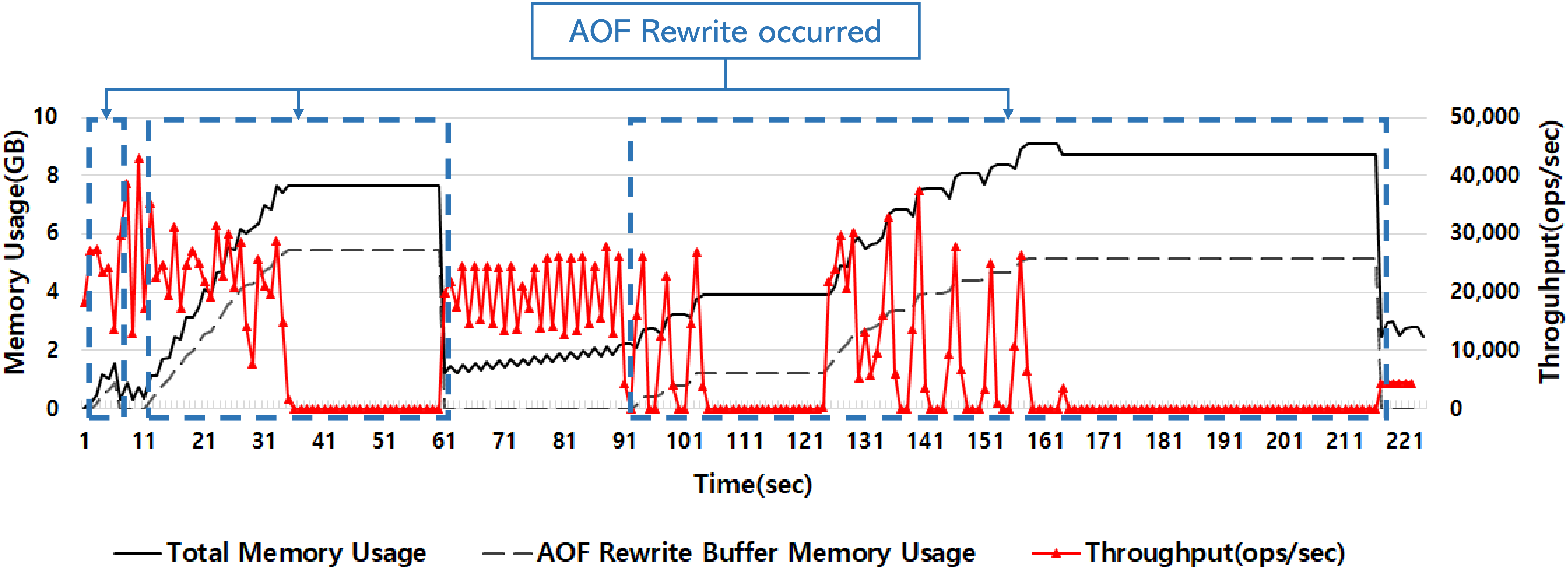
Redis setup

Redis version	4.0.10
AOF Option	everysec
Max Memory Option	50GB
Memtier-benchmark version	1.2.13

Memtier-benchmark Test Set

Clients	10	
Total Requests	2,000,000	
Request Type	SET	Duplicated SET
Num of Requests	200,000	1,800,000
Key Size (Byte)	16	
Data Size (KByte)	10	

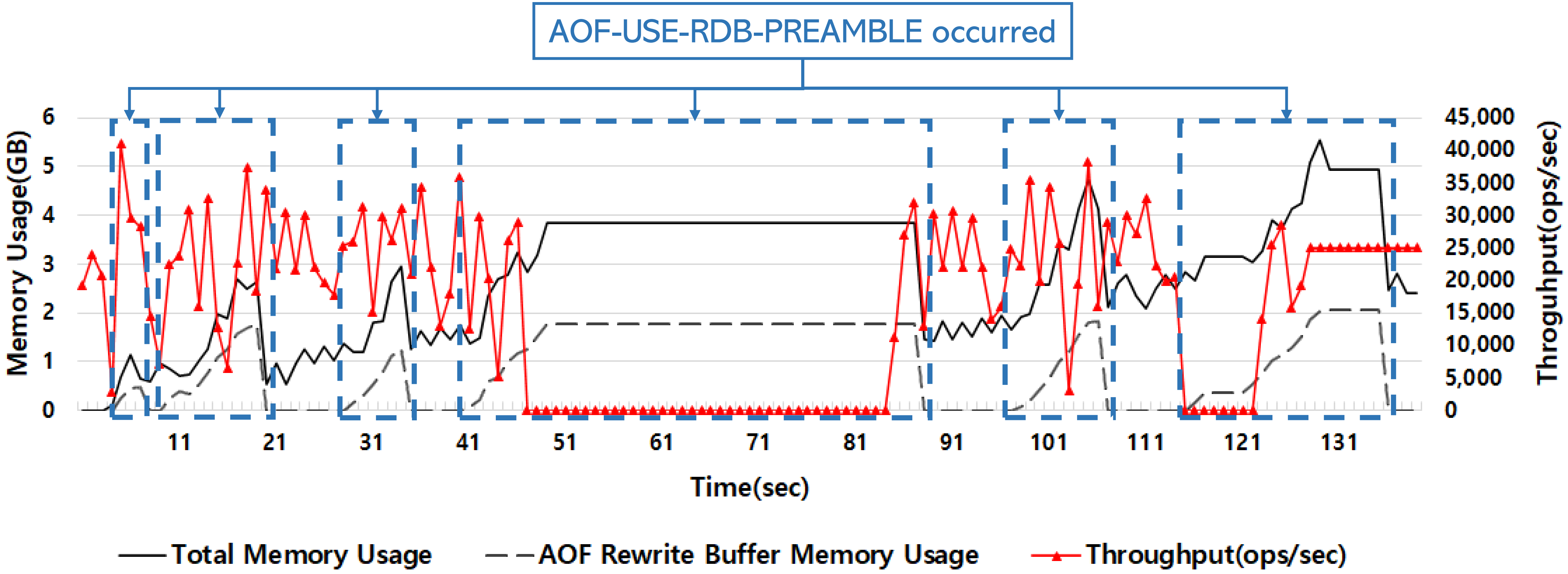
- **Logging Overhead Test**



AOF Rewrite method overhead measurement
(x-axis: flow of time, y-axis: memory usage and throughput)

Motivation

- **Logging Overhead Test**



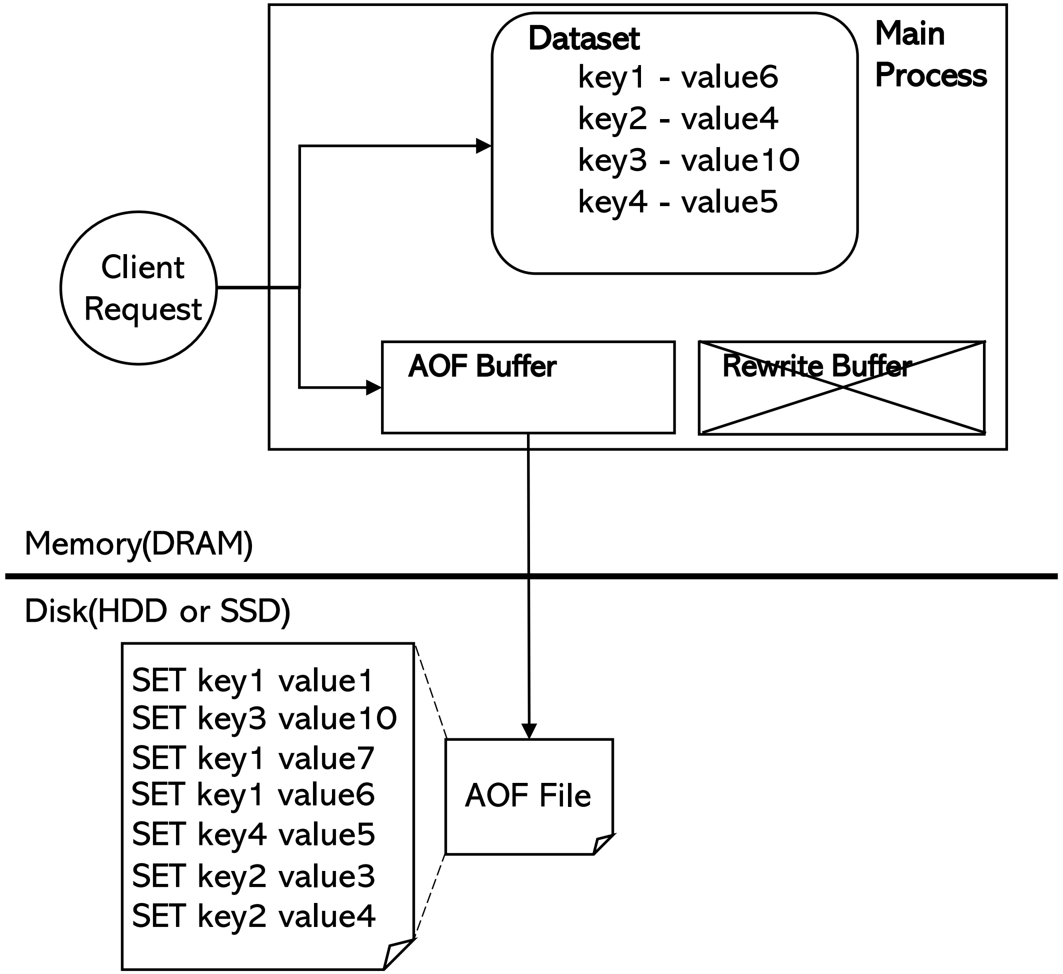
AOF-USE-RDB-PREAMBLE method overhead measurement
(x-axis: flow of time, y-axis: memory usage and throughput)

4. The design of LEAST

- Logging Exploiting A Split snapshot (LEAST)
  - LEAST Logging Mechanism
  - LEAST Recovery Mechanism

The design of LEAST

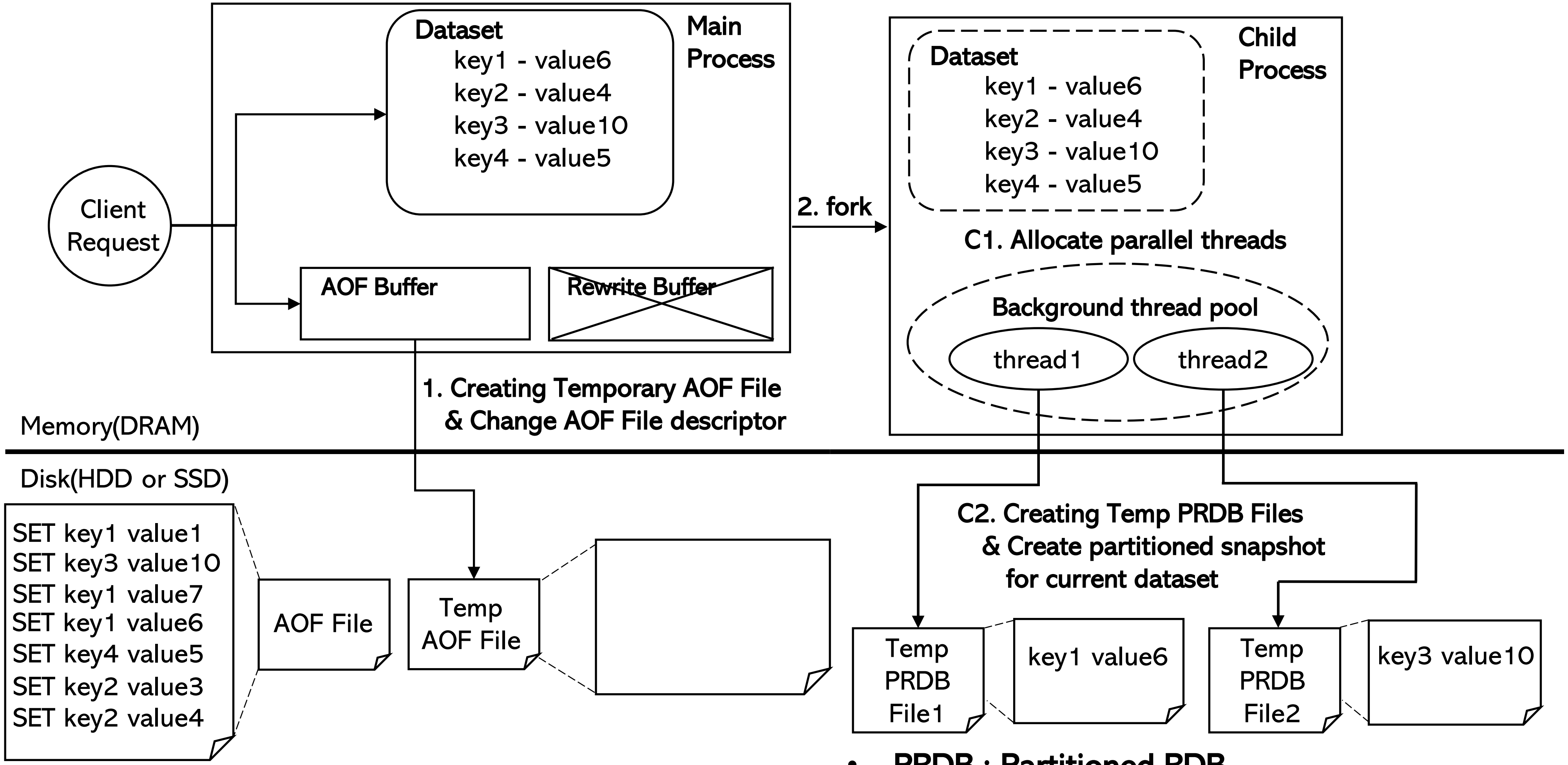
- **LEAST Logging Mechanism**
  - Designed to reduce memory usage and improve data processing performance



- Features of LEAST method
  1. combine AOF and RDB
  2. leverage data parallelism
  3. exclude the use of Rewrite buffer
  4. manage log files separately
  5. restore a dataset using multiple log files
- Perform AOF until LEAST is triggered
- LEAST trigger condition
  - AOF file size > threshold (64 MB)

4 The design of LEAST

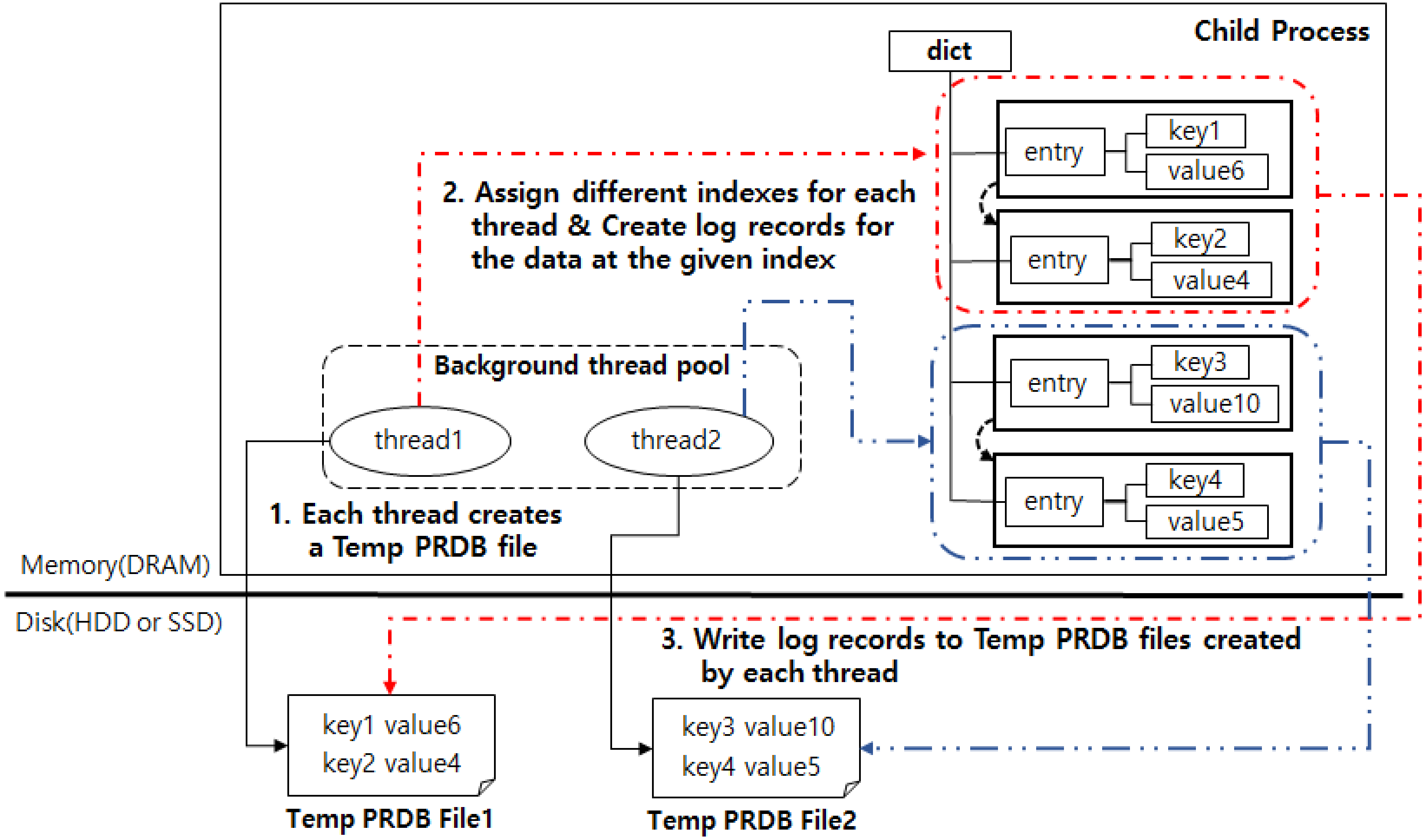
- **LEAST Logging Mechanism**



- **PRDB : Partitioned RDB**
- **Temp PRDB : Partitioned temporary RDB**

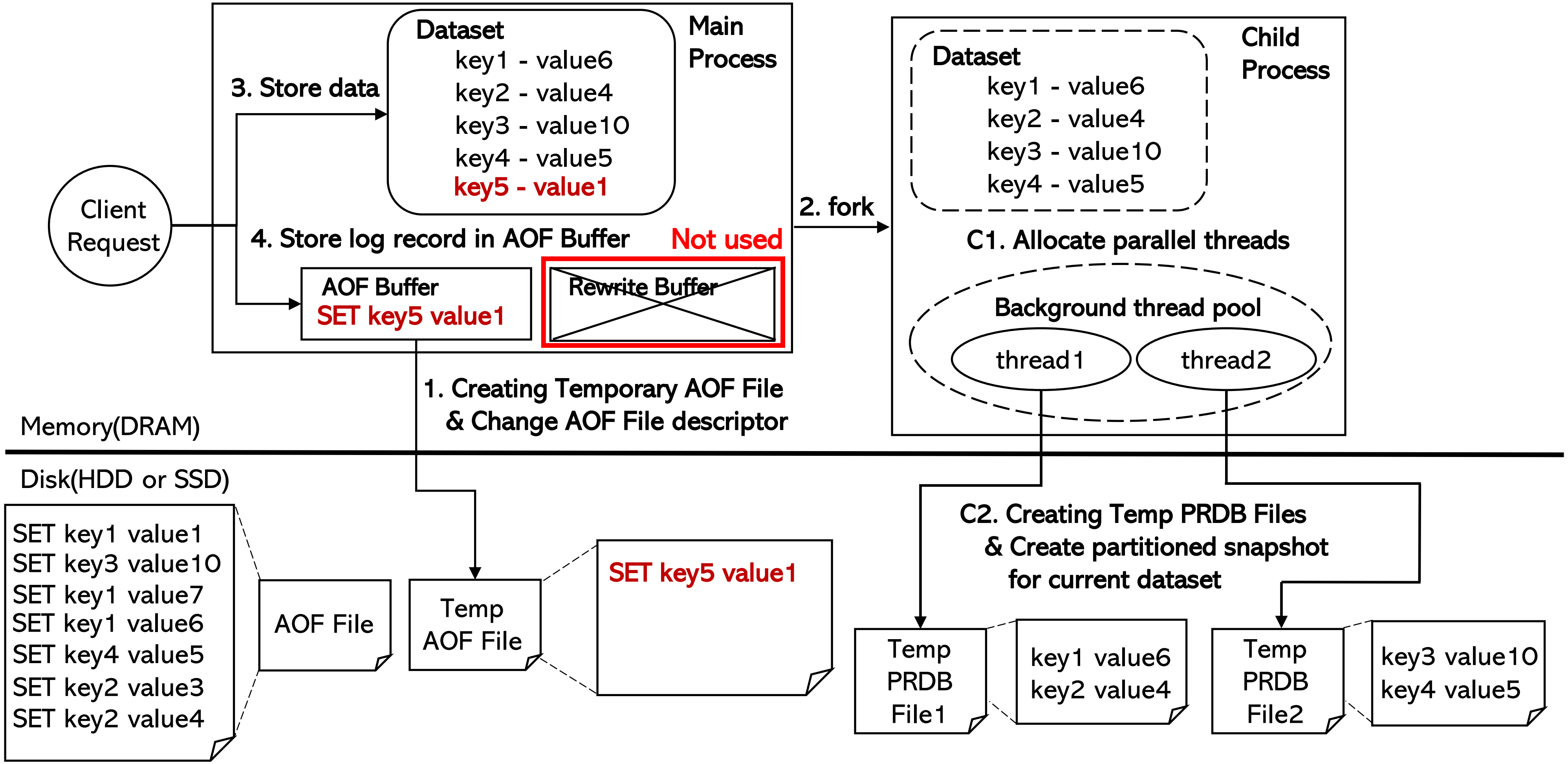
The design of LEAST

- **LEAST Logging Mechanism**



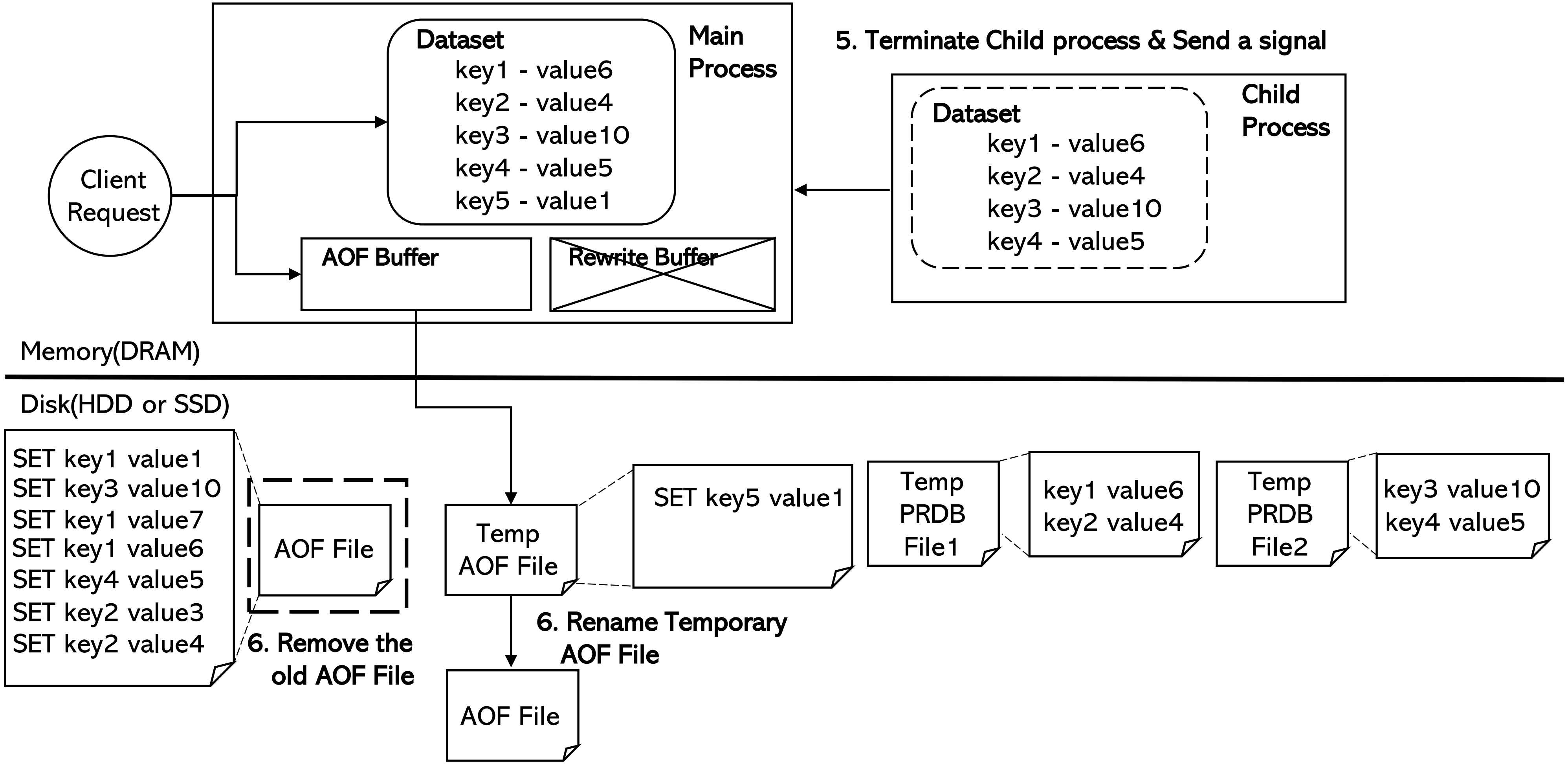
The design of LEAST

- **LEAST Logging Mechanism**



The design of LEAST

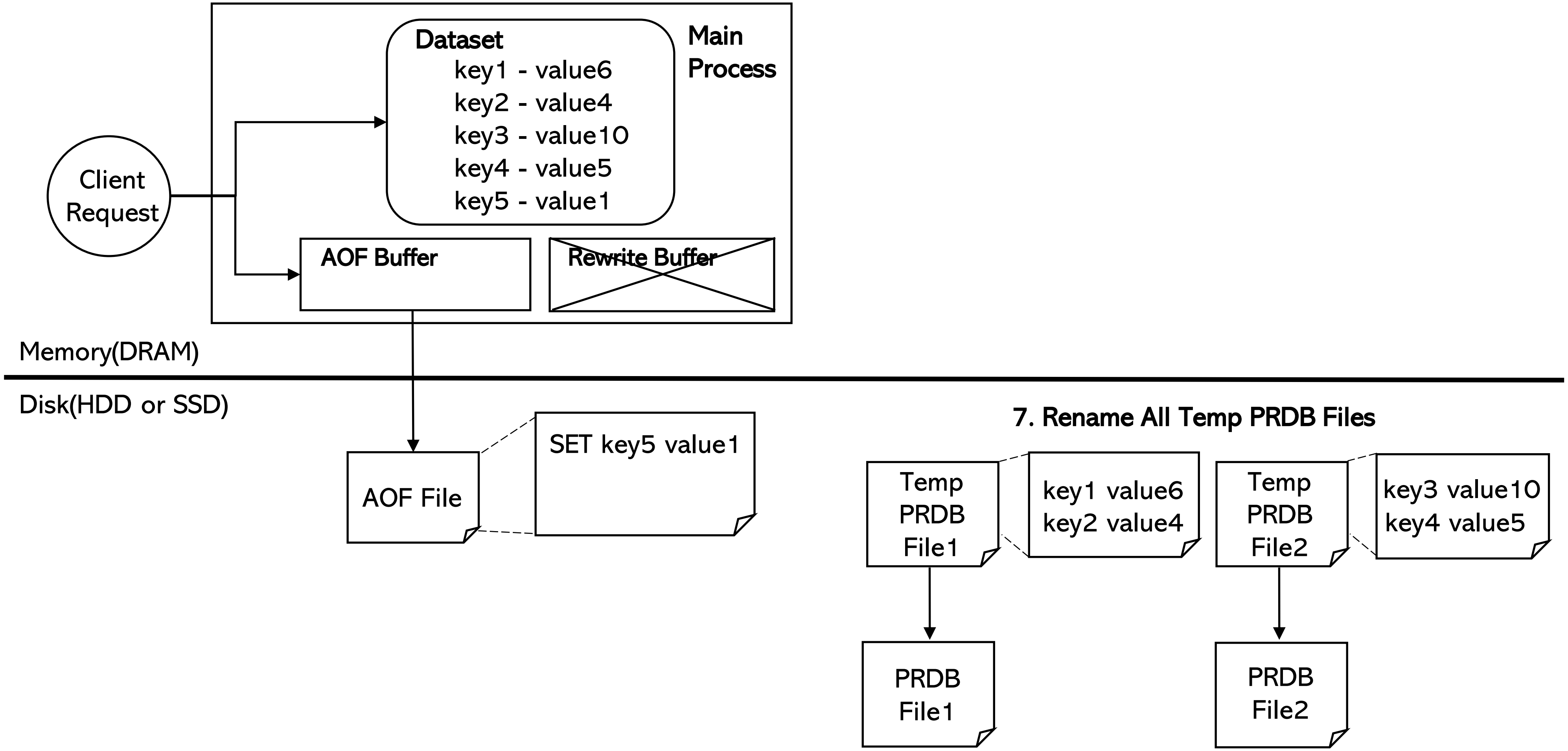
- **LEAST Logging Mechanism**



- LEAST method does not perform *Flush* operation ➔ Reduce the amount of disk I/O

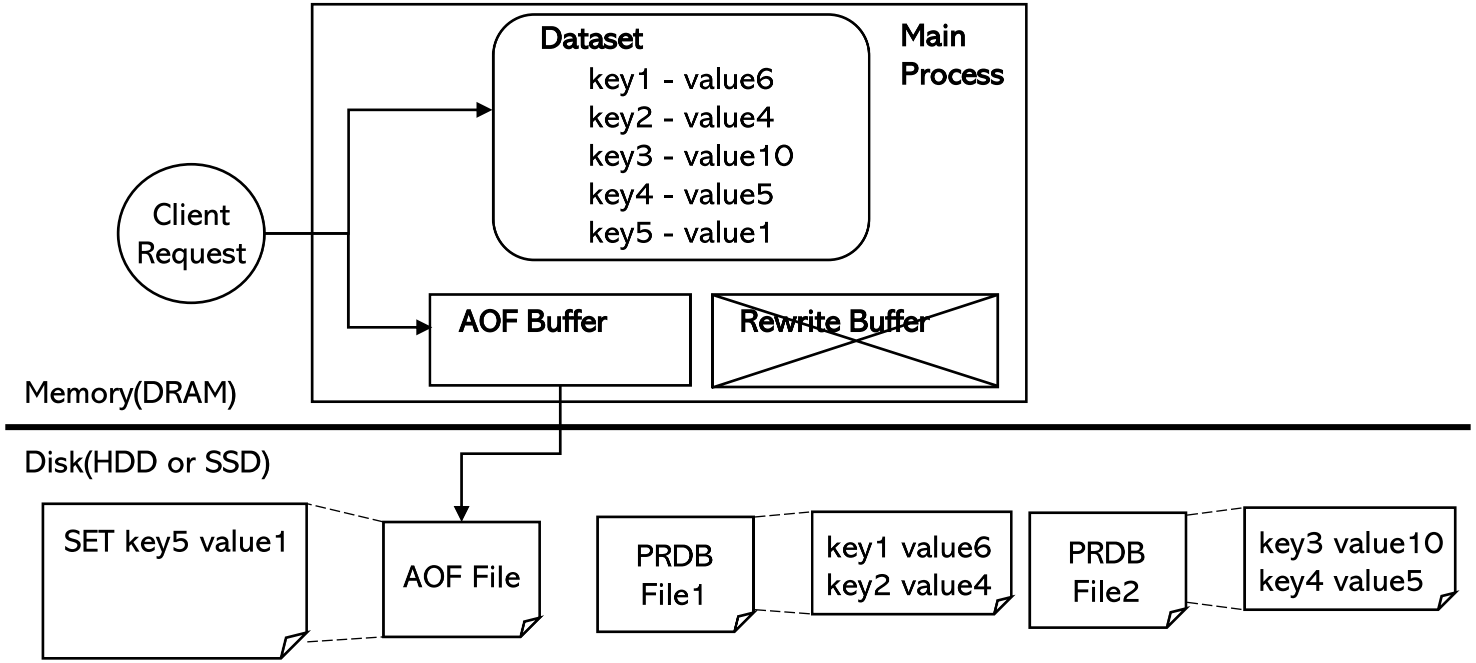
The design of LEAST

- **LEAST Logging Mechanism**



4 The design of LEAST

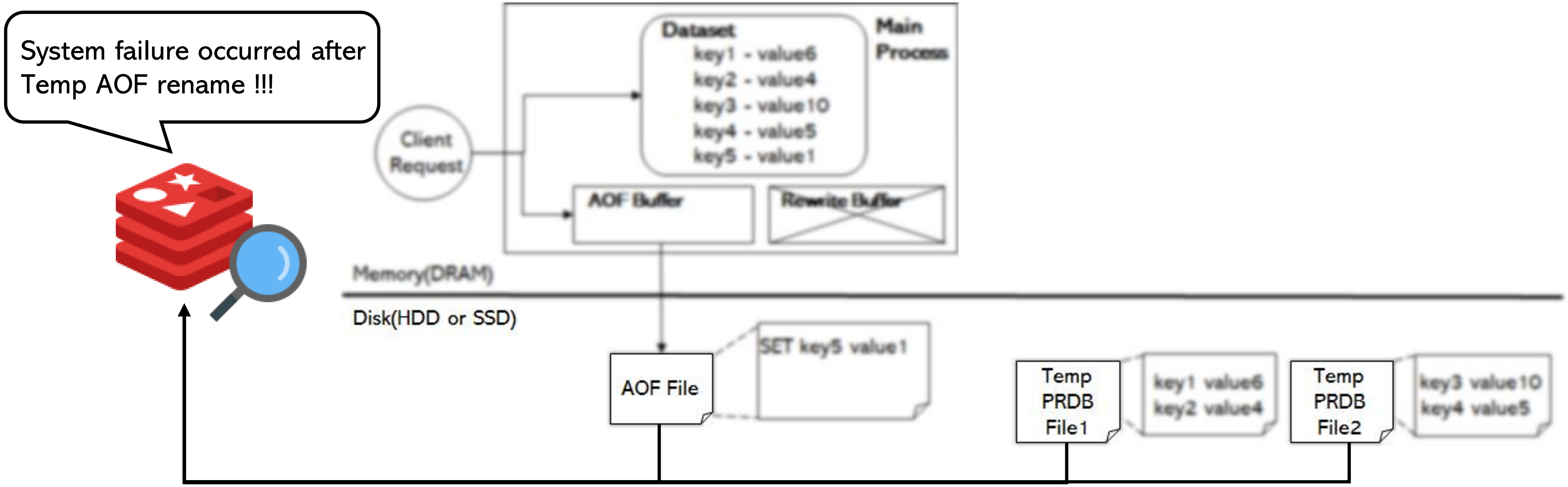
- **LEAST Logging Mechanism**



- LEAST method manages log files separately ➔ <u>Reduce occurrence of disk I/O</u>

4 The design of LEAST

- **LEAST Recovery Mechanism**
  - During LEAST operation, up to four types of files are generated
    - ✓ **AOF File**
    - ✓ **Temp AOF File**
    - ✓ **PRDB File**
    - ✓ **Temp PRDB File**
  - LEAST creates a different types of files for each step
  - By examining files stored on disk, Redis can infer when a system failure occurred



System failure occurred after Temp AOF rename !!!

The design of LEAST

- **LEAST Recovery Mechanism**

Recovery mechanism of LEAST in each case

When failure occurs during LEAST operation	List of files present on disk	Recovery order of LEAST
Before the LEAST operates	PRDBs, AOF	1) PRDBs 2) AOF
Before starting parallel RDB creation	PRDBs, Temp AOF, AOF	1) PRDBs 2) AOF 3) Temp AOF
During parallel RDB creation	Temp PRDBs, Temp AOF, PRDBs, AOF	1) PRDBs 2) AOF 3) Temp AOF
After temporary AOF rename	Temp PRDBs, PRDBs, AOF	1) Temp PRDBs 2) AOF
During Temp PRDB rename	Renamed Temp PRDBs, Temp PRDBs, AOF	1) Renamed Temp PRDBs 2) Temp PRDBs 3) AOF
After Temp PRDB rename	PRDBs, AOF	1) PRDBs 2) AOF

5. Evaluation

- Experimental Setup
- Comparison of Logging Overhead
- The Effect of the Number of Threads on RDB
  - RDB Creation Time
  - RDB Recovery Time
- The Effect of the Number of Threads on LEAST
- Performance Evaluation
  - Throughput
  - Maximum Memory Usage
  - Average Memory Usage
- Recovery Time

- **Experimental Setup**

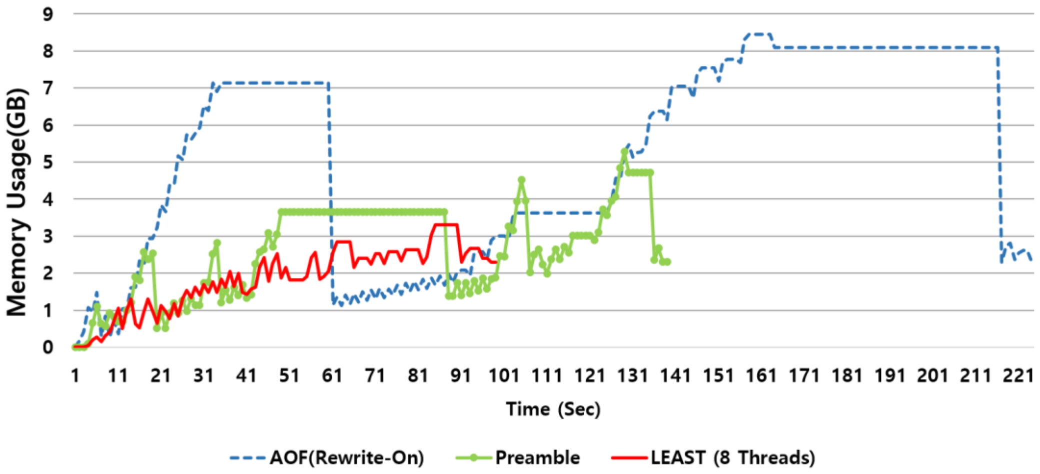
Hardware setup

CPU	Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz
DRAM	DDR3 64GB
SSD	Crucial_CT250MX200SSD1 250 GB * 3

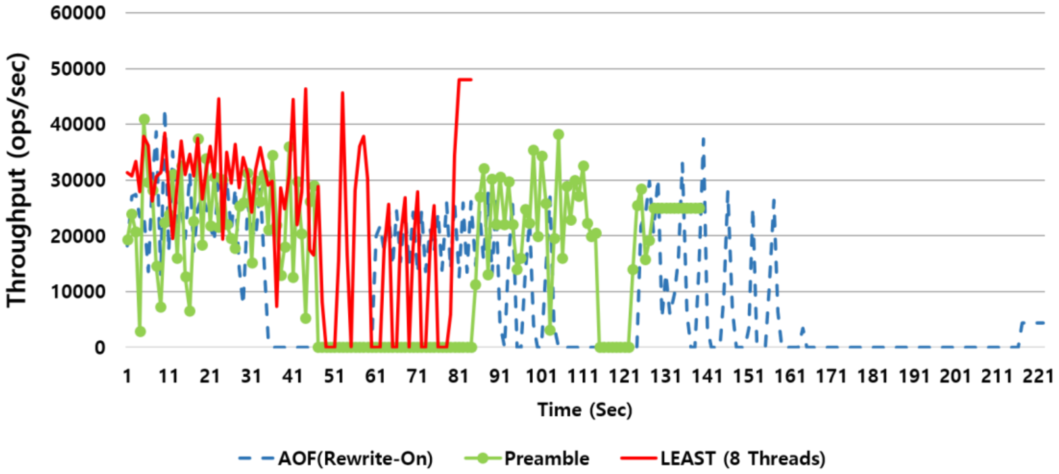
Software setup

OS	Cent OS 7.3.1611 (Core)
Kernel version	3.10.0-514.26.2.el7.x86_64
Redis version	4.0.10
AOF Option	everysec
Max Memory Option	50GB
Memtier-benchmark version	1.2.13

- **Comparison of Logging Overhead**
  - Logging overhead is measured by simulating a situation with high frequency of updates
  - LEAST method has the lowest memory usage and the fastest data processing performance

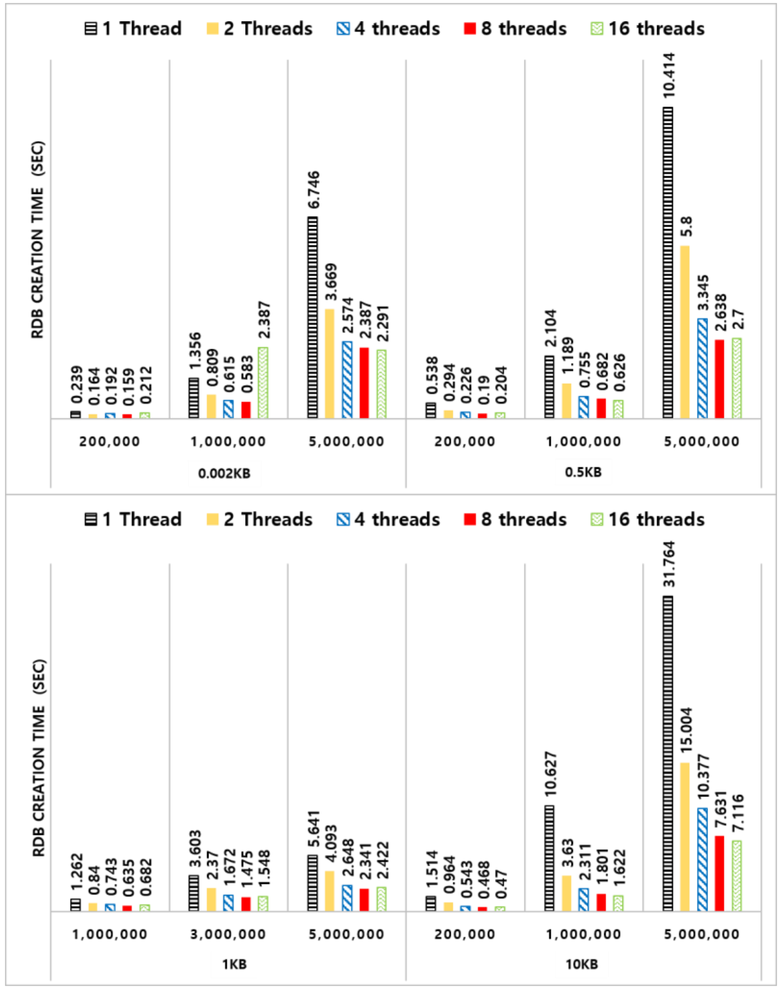


Memory usage
(x-axis: flow of time, y-axis: memory usage)

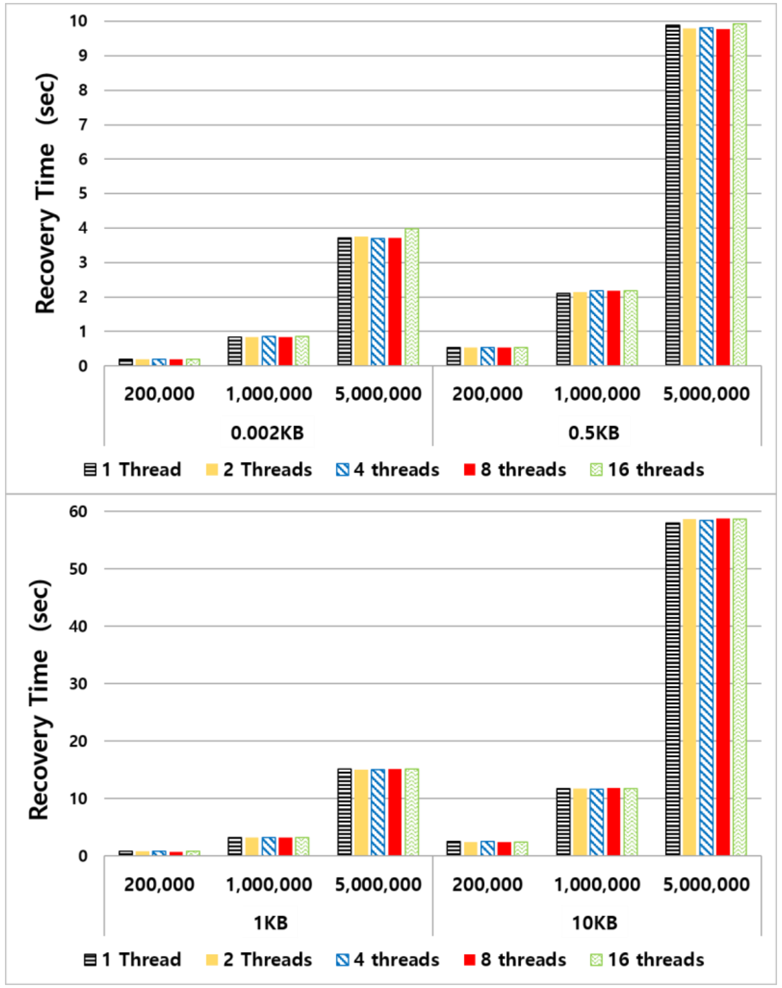


Throughput
(x-axis: flow of time, y-axis: throughput)

- **The Effect of the Number of Threads on RDB**



RDB Creation Time



RDB Recovery Time

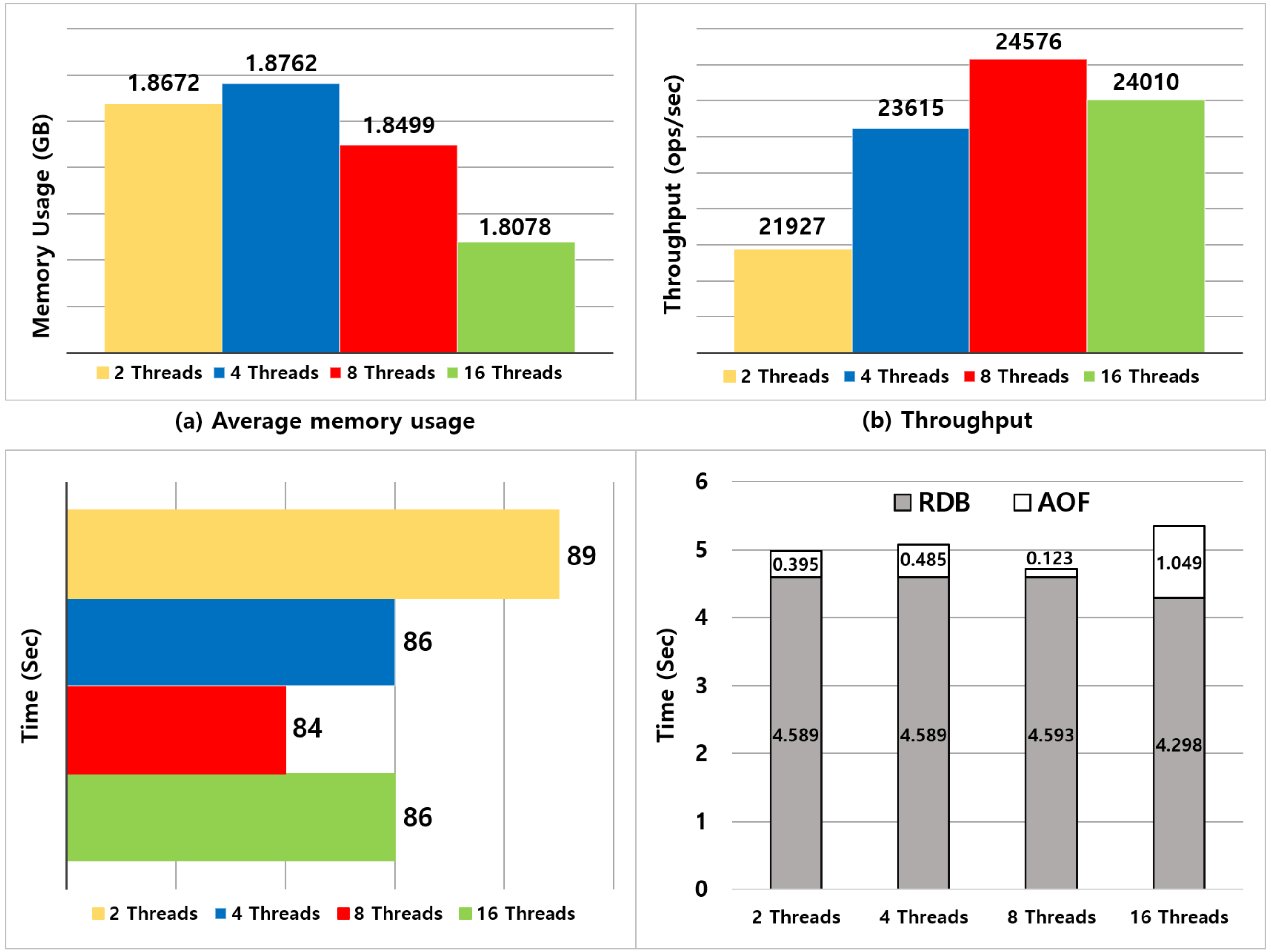
(x-axis: number of requests (upper) and size of values (lower), y-axis: time)

- RDB Creation Time
  - As the number of threads increases…
    - ✓ the amount of time to complete RDB creation decreases
    - ✓ the ratio of time reduction gradually decreases

- RDB Recovery Time
  - As the number of threads increases…
    - ✓ the number of PRDB files generated also increases
    - ✓ the time to restore dataset was similar in all cases

5 Evaluation

- **The Effect of the Number of Threads on LEAST**
  - Measure the overhead of LEAST according to the number of threads used



(a) Average memory usage

(b) Throughput

(c) Execution time

(d) Recovery time

Comparison of the results of changing the number of threads in LEAST method

Size of the log files created after the workload is performed

Num of threads	AOF file size	PRDB file size
2	627 MB	14 MB per file
4	818 MB	7 MB per file
8	219 MB	3.5 MB per file
16	1.8 GB	1.6 MB per file

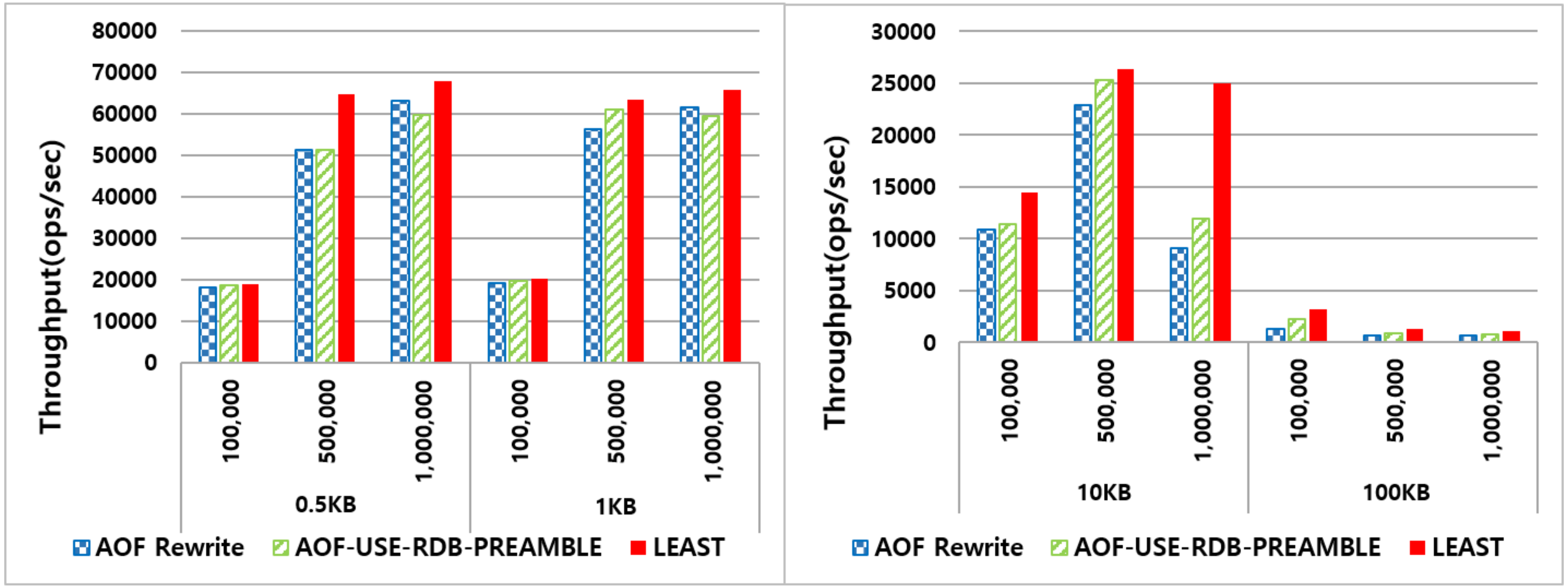
- To measure logging overhead, workload with frequent updates is used

- The best results are in the case of 8 threads
  - ✓ Throughput
  - ✓ Execution time
  - ✓ Recovery time

- Recovery time depends on the size of AOF file

5 Evaluation

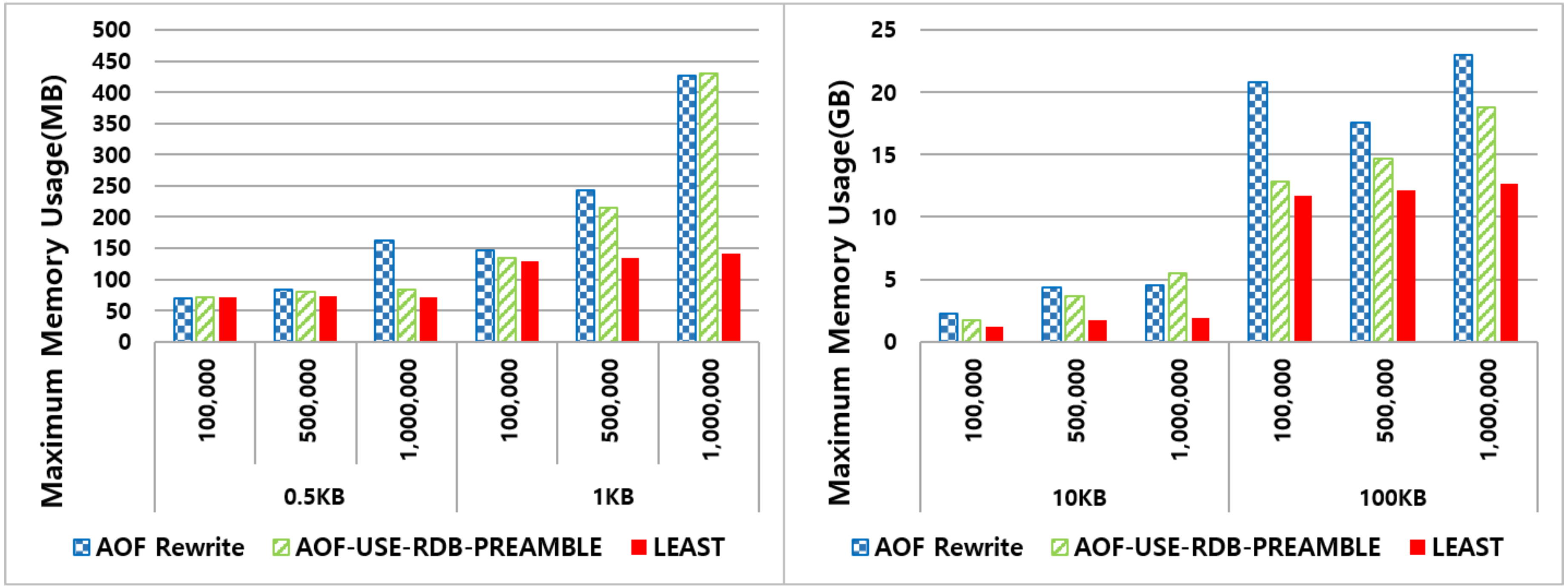
- **Performance Evaluation**
  - Comparison of throughput AOF Rewrite, AOF-USE-RDB-PREAMBLE, LEAST in various environment
  - Redis with LEAST method achieves the fastest data processing performance



Throughput for various number of requests and sizes of values applied
(x-axis: number of requests (upper) and size of values (lower), y-axis: throughput)

Evaluation

- **Performance Evaluation**
  - Comparison of maximum memory usage AOF Rewrite, AOF-USE-RDB-PREAMBLE, LEAST in various environments
  - LEAST shows almost constant maximum memory usage ➔ Safe from **out-of-memory**



Maximum memory usage for various number of requests and sizes of values applied (lower memory usage is better)

5 Evaluation

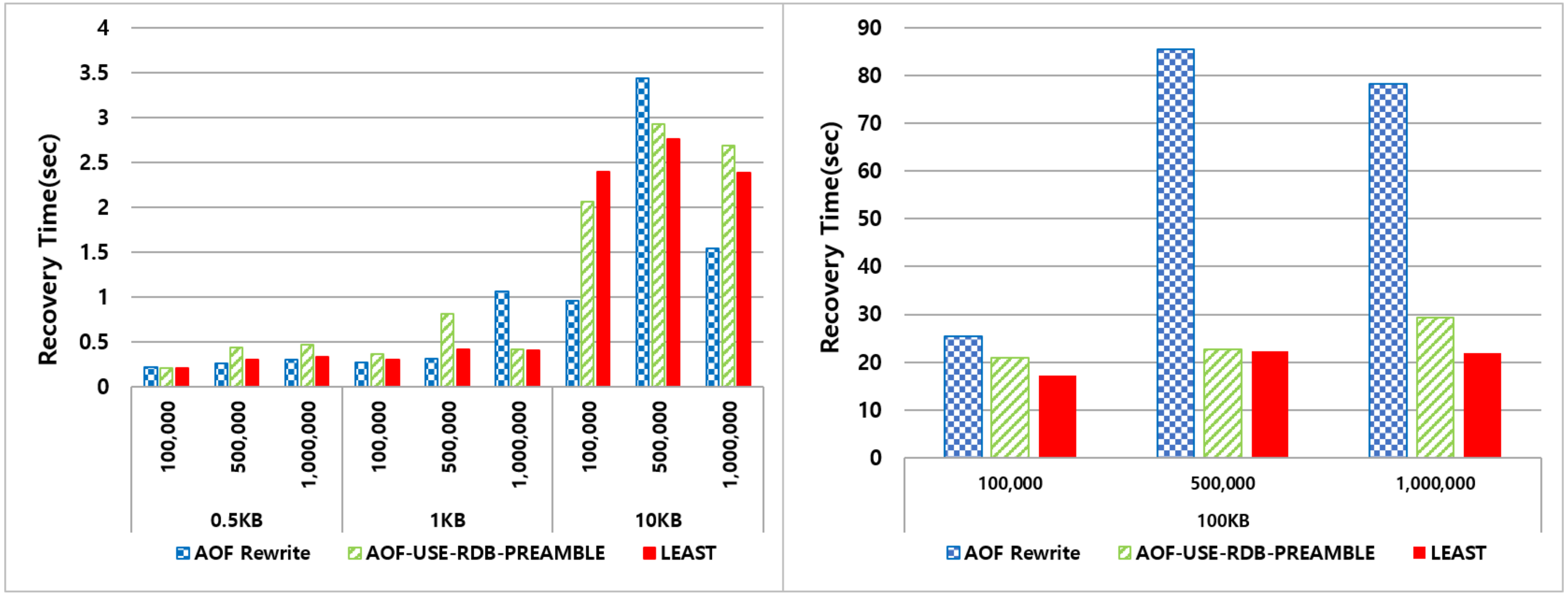
- **Performance Evaluation**
  - Comparison of average memory usage AOF Rewrite, AOF-USE-RDB-PREAMBLE, LEAST in various environments

Average memory usage measurement results for various workloads

Value size	Number of Requests	AOF Rewrite	AOF-USE-RDB-PREAMBLE	LEAST
0.5 KB	100,000	42.62 MB	41.89 MB	41.78 MB
	500,000	44.82 MB	44.51 MB	39.99 MB
	1,000,000	51.11 MB	44.89 MB	43.68 MB
1 KB	100,000	90.5 MB	89.16 MB	69.97 MB
	500,000	140.45 MB	105.05 MB	74.31 MB
	1,000,000	169.14 MB	146.52 MB	81.4 MB
10 KB	100,000	1.28 GB	1.01 GB	0.74 GB
	500,000	2.32 GB	1.42 GB	0.93 GB
	1,000,000	2.51 GB	2.53 GB	0.94 GB
100 KB	100,000	11.6 GB	6.97 GB	5.79 GB
	500,000	9.95 GB	8.21 GB	6.67 GB
	1,000,000	11.47 GB	9.22 GB	6.77 GB

5 Evaluation

- **Recovery Time**
  - Use log files generated after each operation performed in performance evaluation
  - All three persistence methods recover the data completely
  - In most cases, LEAST's recovery time is shorter than that of the existing methods



Recovery time for various numbers of requests and sizes of values applied

6 Conclusion

- In summary,
  - analyze logging overhead of AOF Rewrite and AOF-USE-RDB-PREAMBLE
    1. Memory overhead: Rewrite buffer & Copy-on-write
    2. Throughput degradation: Flush operation(Heavy disk I/O)
  - propose novel design of persistence method leveraging data parallelism and snapshot
    1. Combine AOF and RDB → guarantee data persistence & maintain minimal memory usage
    2. Parallel RDB generation → improve RDB generation performance
    3. Exclude the use of Rewrite buffer → reduce memory usage
    4. Manage log files separately → reduce heavy disk I/O
    5. Recovery mechanism that uses multiple log files
  - improve RDB generation performance
  - show better throughput and lower memory usage compared to the existing persistence methods
  - after system failure, system can reactivate normally through fast data recovery

End.

Q & A