CP-Redis

연세대학교 컴퓨터과학과 성한승

2020년 2월

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

과제번호: 2017-0-00477

Contents

01 Introduction

Introduction

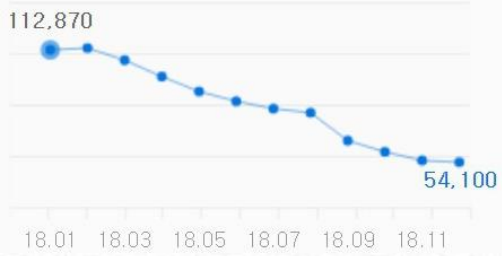
- Changes in Trends
  - Device
    - ✓ Demand for rapid processing of large volumes of data
      - ➔ Need high-performance device

Hard Disk Drive & Solid-State Drive

Dynamic Random-Access Memory

- ✓ DRAM Cost per GB($/GB) is expensive
  - ✓ DRAM Cost per GB: $2.57
  - ✓ SSD Cost per GB: $0.23

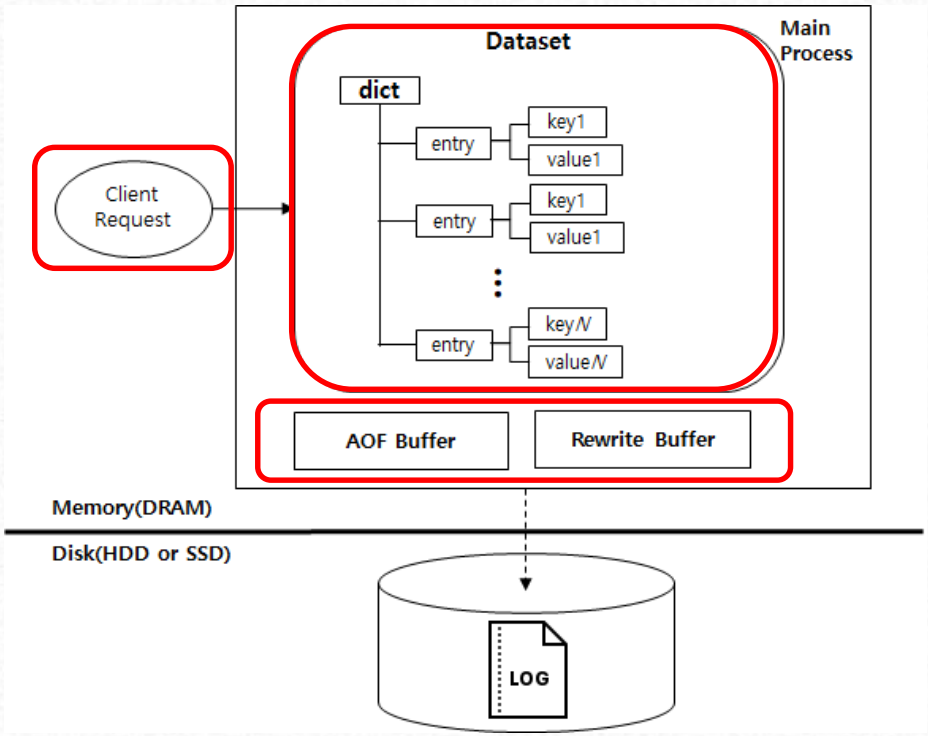
DRAM(DDR4 PC4-21300 8GB)
Price Trend



SSD(250GB)
Price Trend

Introduction

- In-Memory Key-Value Stores require memory space for system operations
  - Data storage
  - Log buffers
  - Index
  - Etc…



02 Background & Motivation

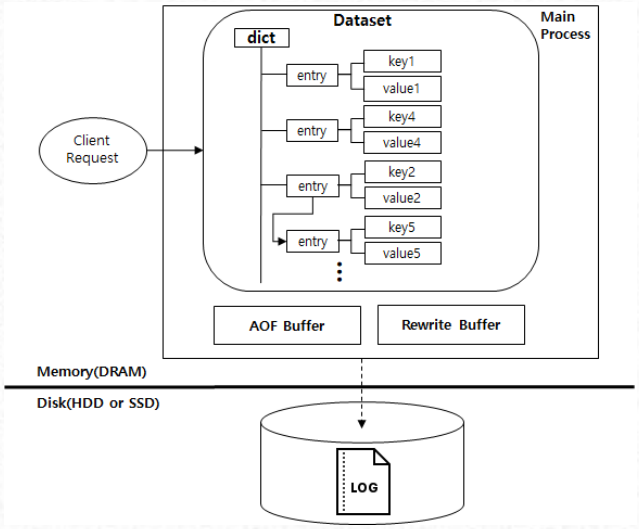
Background

- Redis
  - SET command process
    1. Redis client send key-value data to server
    2. Redis server check the requested parameters
    3. Calculate the hash value of the string for the key to determine the location where data will be stored
    4. Make dictEntry & pointing requested key-value data
    5. Store dictEntry in dict

  - Additional memory required for data management
    - ✓ dictEntry
      - ✓ Key pointer
      - ✓ Redis object pointer
      - ✓ Next entry pointer
    - ✓ Redis object
      - ✓ Data type
      - ✓ Reference count
      - ✓ Time

  - Redis stores requested data in its original form
    - ✓ Fast data processing performance
    - ✓ Increased burden of data storage & system operation

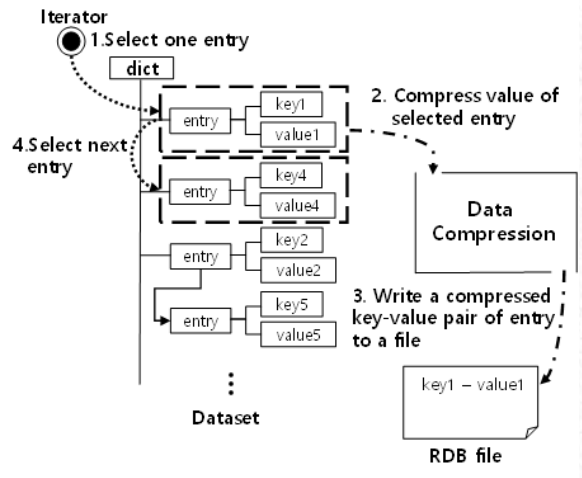
  - If Redis runs out of available memory…
    1. Redis will not be able to execute commands
    2. Out-Of-Memory ➔ System Shutdown



Redis dict structure

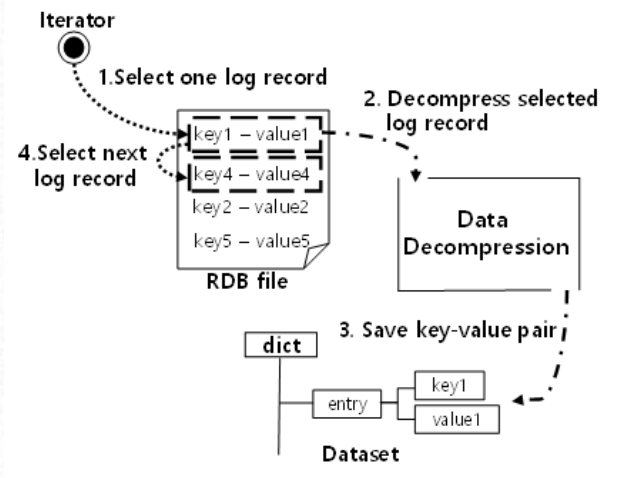
Background

- Redis Database(RDB)
  - RDB Logging



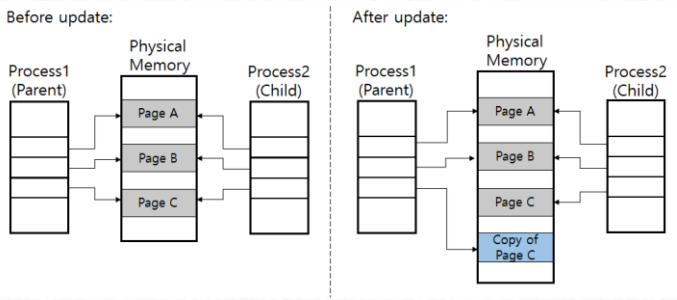
- Generate point-in-time snapshots(compressed binary format) of a dataset at specific intervals
- Advantage
  - Small file size
  - Higher performance than AOF
  - Fast recovery speed
- Disadvantage
  - Can not guarantee data persistency

  - RDB Recovery

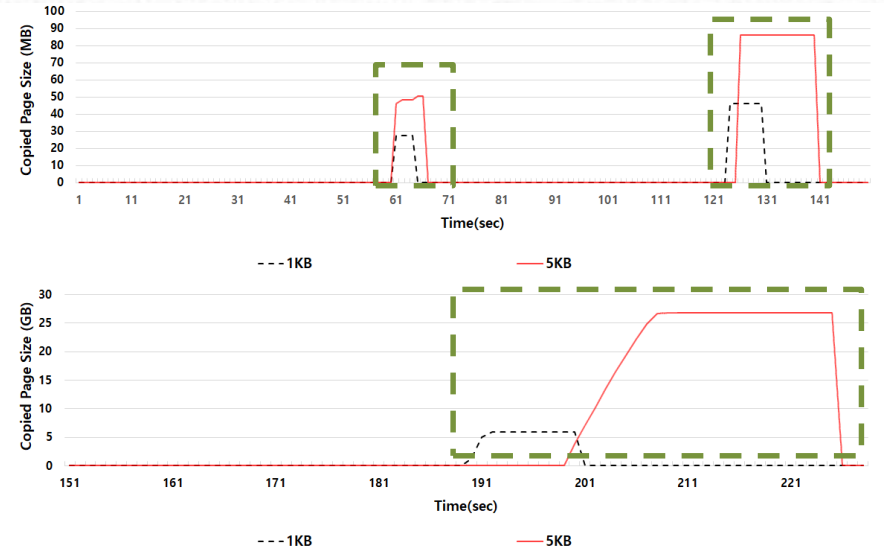


- Redis restarts, it uses RDB file to recover data
- RDB recovery is sequentially performed for all data recorded in the RDB file
- Compared AOF Recovery
  - RDB recovery is faster
  - RDB recovers data by performing only one recovery operation on a key

Motivation

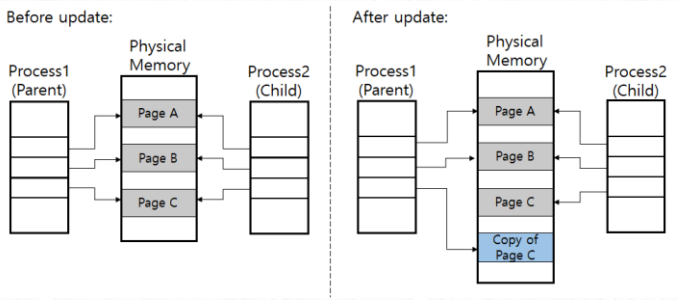


Copy-on-write

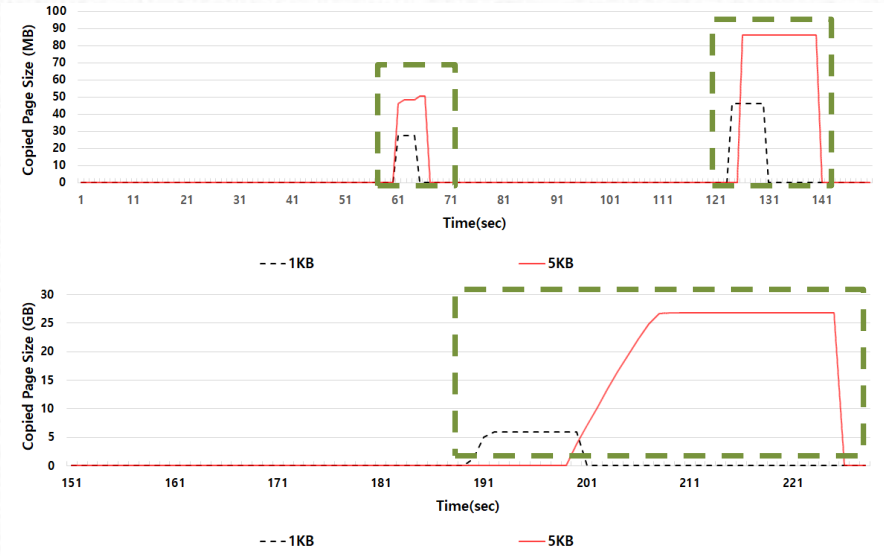


- Redis utilizes copy-on-write technique to perform RDB
- Copy-on-write: copy the page and reflect changes made to the copied page
- Original page of the copied page is maintained until RDB is finished ➔ Memory occupancy
- Copy-on-write allow Redis to handle requested commands without interruption
- But!!! It causes system memory shortage

Motivation



Copy-on-write

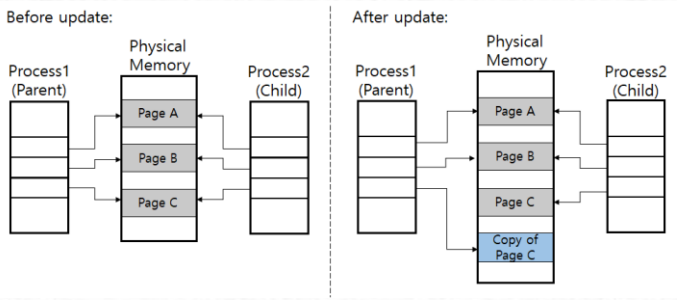


- Simulate workloads of data insertion continues to be requested by users
  ➔ The effect of the copy-on-write can be identified

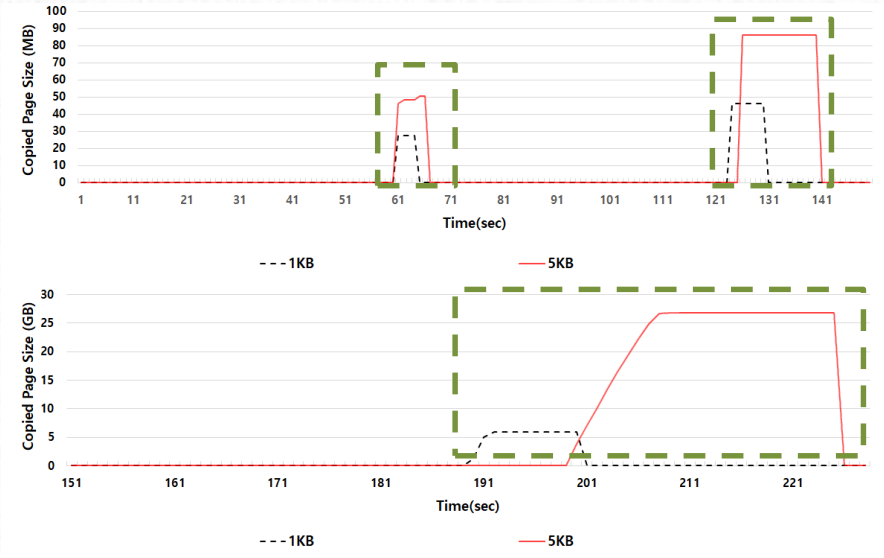
  - Workload A
    - Key – 16B, Value – 1KB
    - 5,000,000 SET Commands

  - Workload B
    - Key – 16B, Value – 5KB
    - 5,000,000 SET Commands

Motivation



Copy-on-write

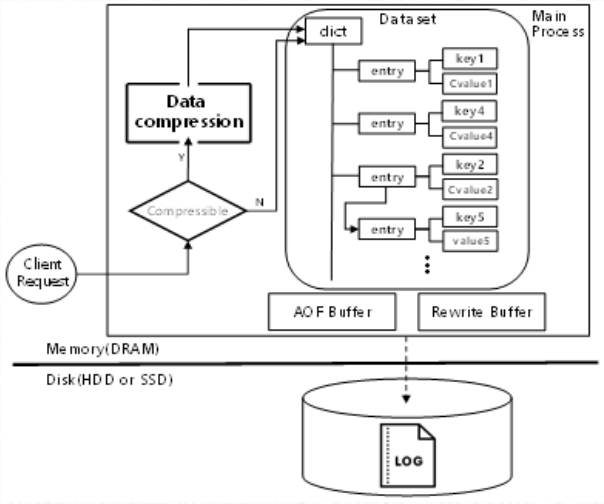


- During both workloads, RDB occurs three times each at 60 seconds intervals.
- As the size of the dataset stored in Redis increases, the size of the page copied during the snapshot creation operation increases.
- The longer the snapshot creation task increases, the longer the original pages exist
- The experimental results indicate a significant memory load on Redis used in production environments where a large amount of data is constantly requested.

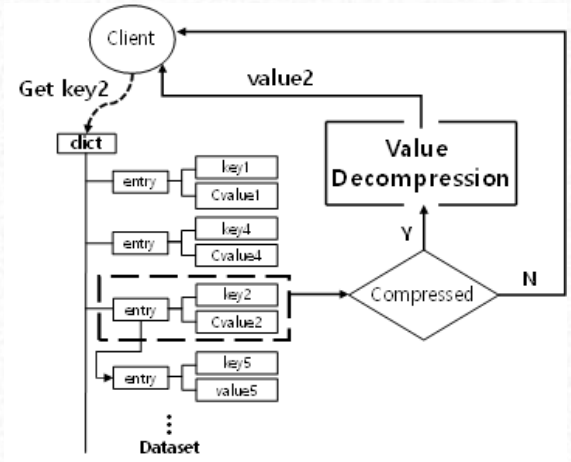
03 CP-Redis

CP-Redis

- Data Compression Storage



Data compression storage

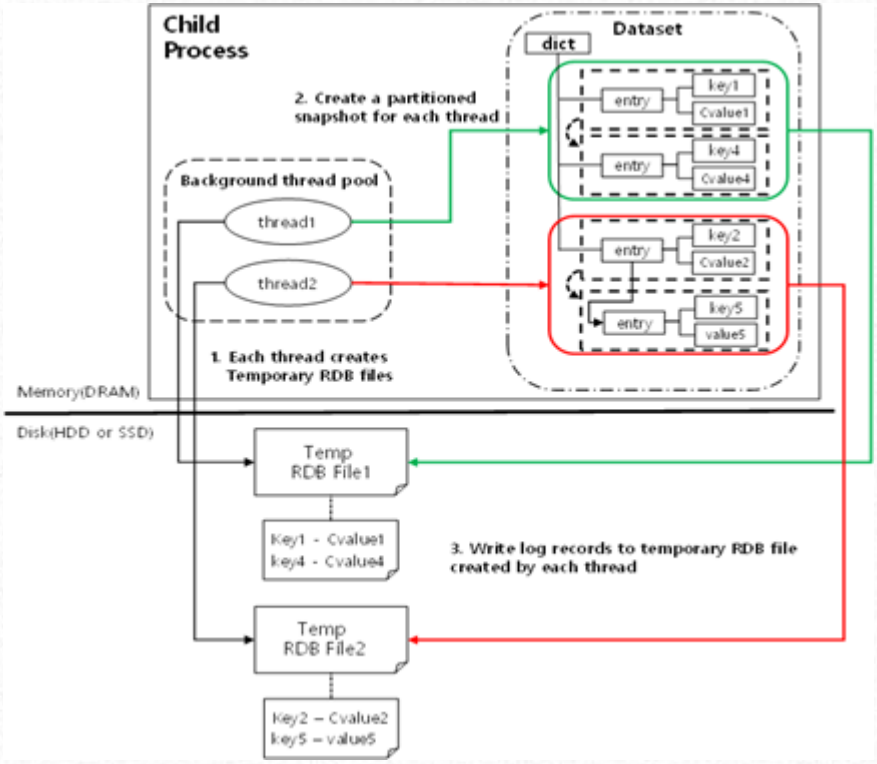


Compressed value data lookup

- Compress & store value data over 5KB
- To minimize data processing due to compression, use LZF (Lempel-Ziv-Free) compression algorithms
- LZF algorithm: Fast compression speed with low memory consumption
- To maintain data processing performance, key string is stored in the original form without compressing.
- If the value data is stored in compressed form when the GET command is executed, the value data is decompressed and returned.

CP-Redis

- Parallel RDB Generation

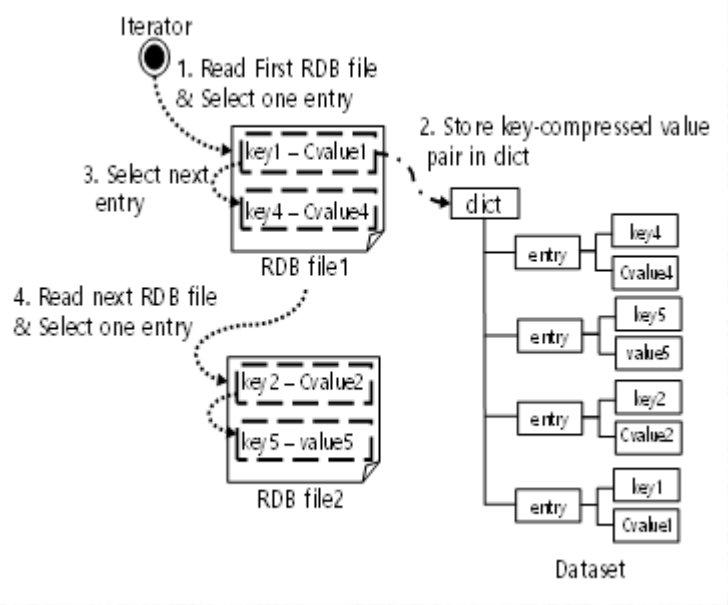


PRDB

- Increase the size and number of storable data ➔ Increase logging overhead
- RDB operation is performed using parallel processing
- During RDB operations, log records are written directly to the log file without compressing data
- Data stored in Redis is partitioned into PRDB files and stored

CP-Redis

- PRDB Recovery



- All PRDB files that exist on disk are used to recover 100% of the data sets stored in Redis
- Performs a recovery by reading the PRDB files that exist on the disk in sequence of file numbers
- Recover data directly to the disk without decompression, even if the value data of the log record recorded in the PRDB file is compressed

04 | Evaluation

Evaluation

- Configuration

Hardware Setup	
CPU	Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz
RAM	DDR3 64 GB
Disk(SSD)	Crucial_CT250MX200 SSD1 250 GB * 3
Software Setup	
OS	Cent OS 7.3.1611 (Core)
OS Kernel	3.10.0-514.26.2.el7.x86_64
Redis version	4.0.10
Maxmemory option	60 GB
RDB option	save 60 10000
Memtier-Benchmark Version	1.2.13

- Data insertion workload

Workload Name	Key Size (B)	Number of SET Requests	Value size (KB)	Total Data Size
W1	16	100,000	0.002	1.72 MB
W2			0.5	50.35 MB
W3			1	99.18 MB
W4			10	978.09 MB
W5			20	1.91 GB
W6		500,000	0.002	8.58 MB
W7			0.5	251.77 MB
W8			1	495.91 MB
W9			10	4.78 GB
W10			20	9.54 GB
W11		2,500,000	0.002	42.92 MB
W12			0.5	1.23 GB
W13			1	2.42 GB
W14			10	23.88 GB
W15			20	47.72 GB

Evaluation

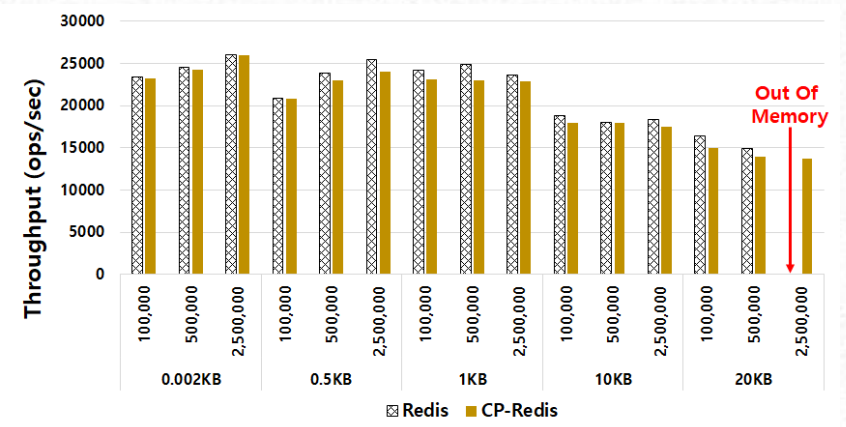
- Configuration

Hardware Setup	
CPU	Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz
RAM	DDR3 64 GB
Disk(SSD)	Crucial_CT250MX200 SSD1 250 GB * 3
Software Setup	
OS	Cent OS 7.3.1611 (Core)
OS Kernel	3.10.0-514.26.2.el7.x86_64
Redis version	4.0.10
Maxmemory option	60 GB
RDB option	save 60 10000
Memtier-Benchmark Version	1.2.13

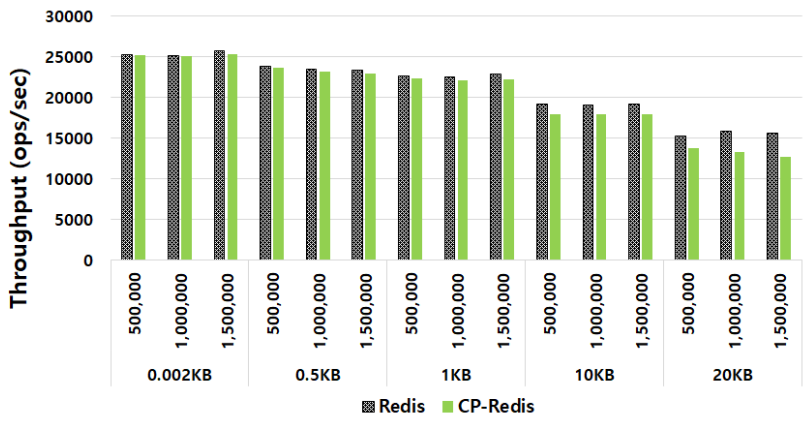
- Data lookup workload

Workload Name	Key Size (B)	Number of SET Requests	Value size (KB)	Number of GET Requests
R1	16	1,500,000	0.002	500,000
R2				1,000,000
R3				1,500,000
R4			0.5	500,000
R5				1,000,000
R6				1,500,000
R7			1	500,000
R8				1,000,000
R9				1,500,000
R10			10	500,000
R11				1,000,000
R12				1,500,000
R13			20	500,000
R14				1,000,000
R15				1,500,000

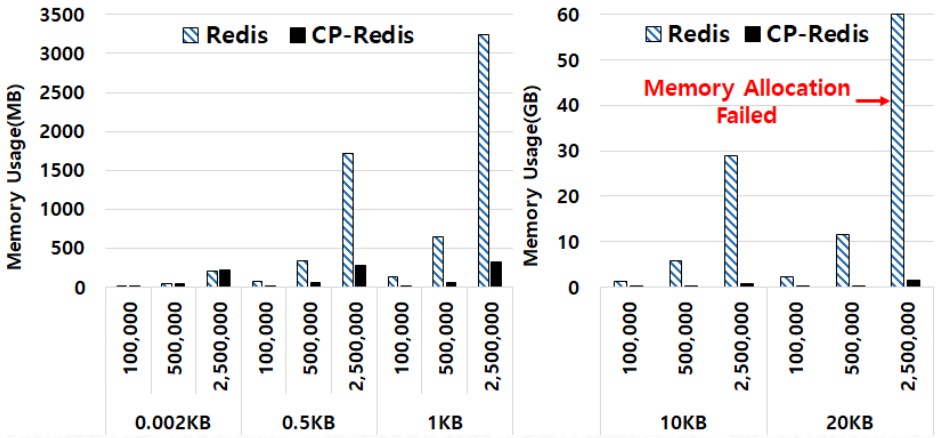
Evaluation



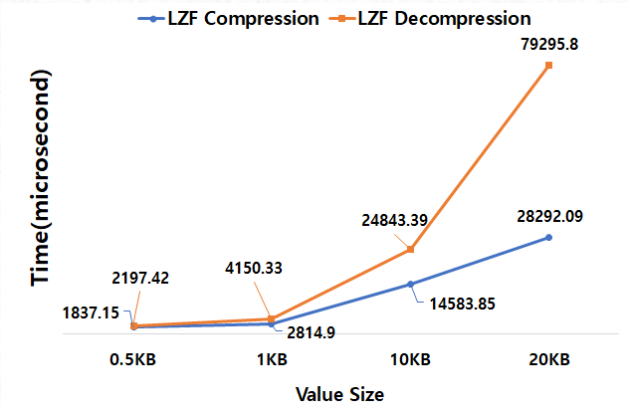
Data Insertion Performance



Data Lookup Performance

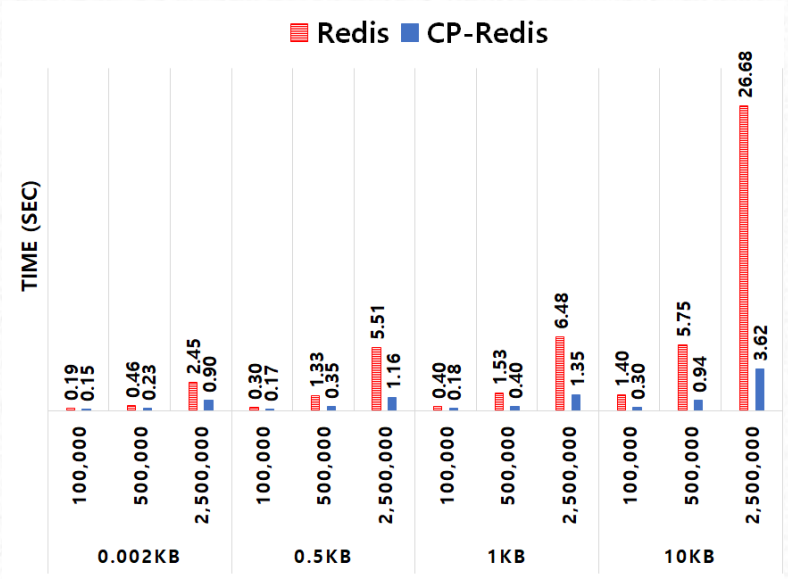


Memory Usage

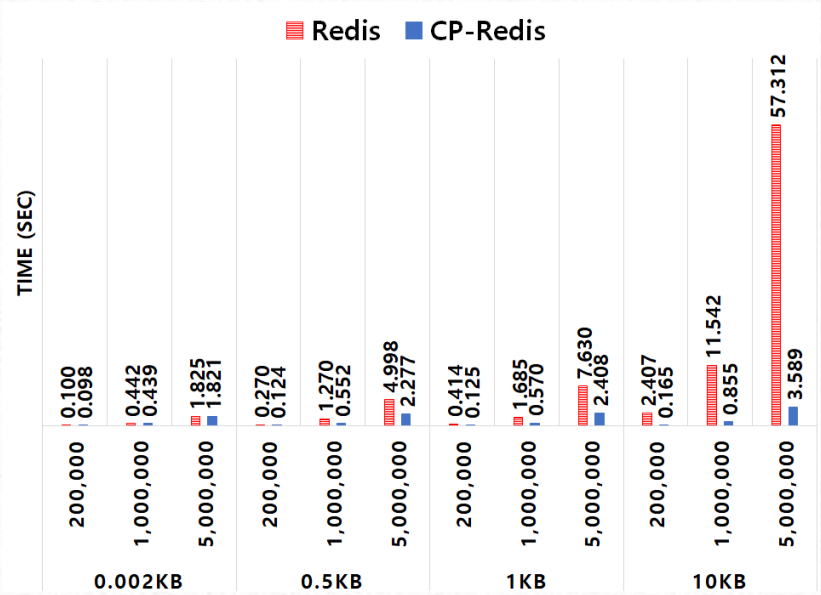


LZF Compression and Decompression Time

Evaluation



PRDB Generation Time



PRDB Recovery Time

Number of Requests	Value size (KB)	Redis		CP-Redis	
		Copied page (MB)	Time (sec)	Copied page (MB)	Time (sec)
2,500,000	0.002	28	1.10	22	0.44
	0.5	30	2.80	36	0.66
	1	36	3.48	36	0.79
	10	94	12.09	28	1.59
		26	26.72	38	2.97

Copy-on-write overhead due to snapshot creation

Thank You