(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0078313
(43) 공개일자 2024년06월03일

(51) 국제특허분류(Int. Cl.)
G06F 3/06 (2006.01) *G06F 16/215* (2019.01)
G06F 16/22 (2019.01)
(52) CPC특허분류
G06F 3/0626 (2013.01)
G06F 16/215 (2019.01)
(21) 출원번호 10-2023-0125271
(22) 출원일자 2023년09월20일
심사청구일자 없음
(30) 우선권주장
1020220160920 2022년11월25일 대한민국(KR)

(71) 출원인
서강대학교산학협력단
서울특별시 마포구 백범로 35 (신수동, 서강대학교)
(72) 발명자
김영재
서울특별시 마포구 백범로 35 (서강대학교) 아담샬관 911호
김기현
경기도 고양시 일산동구 일산로 205, 209동 704호
자밀 사프다르
서울특별시 서대문구 연세로2라길 19, 201호
(74) 대리인
이철희

전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **중복 제거 기능을 갖는 확장형 LSM 트리 기반 키-밸류 데이터 저장 장치 및 방법**

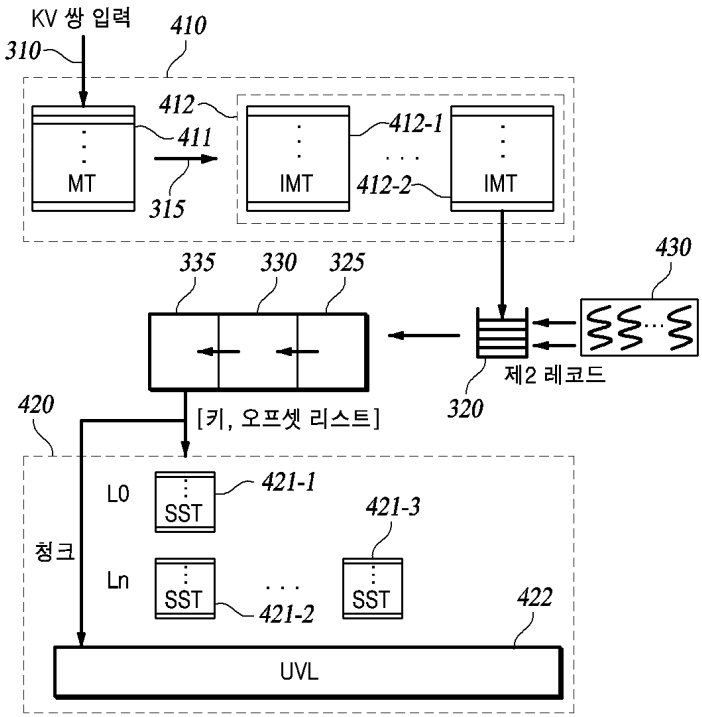
(57) 요 약

본 개시는 중복 제거 기능을 갖는 확장형 LSM 트리 기반 키-밸류 데이터 저장 장치 및 방법에 관한 것이다.

본 개시의 일 측면에 의하면, 키 및 밸류를 구성요소로서 포함하는 제2 레코드의 상기 밸류를 일정 크기의 적어도 하나의 청크로 분할하는 청크 분할 과정; 상기 분할된 각각의 청크마다 핑거프린트를 생성하고 상기 생성된

(뒷면에 계속)

대 표 도 - 도4



핑거프린트와 동일한 핑거프린트를 구성요소로서 포함하는 해시 레코드가 해시정보 테이블 내에 존재하는지 여부를 확인하는 중복청크 확인 과정; 및 상기 각각의 청크와 관련한 해당 해시 레코드가 상기 해시정보 테이블 내에 존재하는지 여부에 따라 해당 청크를 로그 저장소에 저장하고, 상기 각각의 청크마다 상기 로그 저장소 상의 저장 위치인 각각의 오프셋 정보를 획득하고 상기 키 및 상기 각각의 오프셋 정보를 구성요소로서 포함하는 제1 메타데이터 레코드를 생성하여 메타데이터 테이블에 저장하는 데이터 저장 과정을 포함하는 키-밸류 데이터 저장 방법을 제공한다.

(52) CPC특허분류
G06F 16/2246 (2019.01)
G06F 16/2272 (2019.01)
G06F 3/0604 (2013.01)
G06F 3/0608 (2013.01)
G06F 3/0652 (2013.01)
G06F 2212/1016 (2013.01)

명 세 서

청구범위

청구항 1

키 및 밸류를 구성요소로서 포함하는 제2 레코드의 상기 밸류를 일정 크기의 적어도 하나의 청크로 분할하는 청크 분할 과정;

상기 분할된 각각의 청크마다 핑거프린트를 생성하고 상기 생성된 핑거프린트와 동일한 핑거프린트를 구성요소로서 포함하는 해시 레코드가 해시정보 테이블 내에 존재하는지 여부를 확인하는 중복청크 확인 과정; 및

상기 각각의 청크와 관련한 해당 해시 레코드가 상기 해시정보 테이블 내에 존재하는지 여부에 따라 해당 청크를 로그 저장소에 저장하고, 상기 각각의 청크마다 상기 로그 저장소 상의 저장 위치인 각각의 오프셋 정보를 획득하고 상기 키 및 상기 각각의 오프셋 정보를 구성요소로서 포함하는 제1 메타데이터 레코드를 생성하여 메타데이터 테이블에 저장하는 데이터 저장 과정

을 포함하는 키-밸류 데이터 저장 방법.

청구항 2

제1항에 있어서,

상기 키 및 상기 밸류를 수신하여 상기 키 및 상기 밸류를 구성요소로서 포함하는 레코드를 제1 테이블에 저장하는 키-밸류 저장 과정;

상기 제1 테이블에 저장된 모든 레코드의 용량의 합이 제1 임계치 이상인 경우 상기 제1 테이블을 제2 테이블 집합에 포함되는 하나의 제2 테이블로서 설정하는 제2테이블 설정 과정; 및

상기 제2 테이블 집합 내의 상기 제2 테이블의 개수가 제2 임계치 이상인 경우, 상기 제2 테이블 내의 레코드를 순차적으로 획득하여 상기 제2 레코드로 설정하는 제2레코드 획득 과정

을 더 포함하는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 3

제2항에 있어서,

상기 키-밸류 저장 과정은,

상기 제1 테이블이 상기 하나의 제2 테이블로서 설정된 후 새로운 키 및 밸류 쌍을 수신하는 경우, 새로운 테이블에 상기 새로운 키 및 밸류 쌍을 저장하는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 4

제3항에 있어서,

상기 제1 테이블, 상기 새로운 테이블 및 상기 제2 테이블 집합은 휘발성 메모리에 저장되고, 상기 로그 저장소 및 상기 메타데이터 테이블은 비휘발성 메모리에 저장되는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 5

제1항에 있어서,

상기 해시정보 테이블은 적어도 하나의 해시 레코드를 포함하되,

각 해시 레코드는 서로 다른 핑거프린트, 및 해당 핑거프린트에 대응하는 청크의 상기 로그 저장소 상의 저장 위치를 나타내는 오프셋 정보를 구성요소로서 포함하는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 6

제1항에 있어서,

상기 데이터 저장 과정은,

상기 해당 해시 레코드가 상기 해시정보 테이블 내에 존재하지 않는 경우, 상기 해당 청크를 로그 저장소에 저장하고 상기 해당 청크의 상기 로그 저장소 상의 위치를 제1 오프셋 정보로서 획득하여 상기 제1 오프셋 정보를 상기 제1 메타데이터 레코드의 구성요소로서 저장하는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 7

제6항에 있어서,

상기 데이터 저장 과정은,

상기 생성된 핑거프린트 및 상기 제1 오프셋 정보를 구성요소로서 포함하는 해시 레코드를 생성하여 상기 해시정보 테이블에 저장하는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 8

제1항에 있어서,

상기 데이터 저장 과정은,

상기 해당 해시 레코드가 상기 해시정보 테이블 내에 존재하는 경우, 상기 해당 해시 레코드의 구성요소인 해당 오프셋 정보를 획득하고 상기 해당 오프셋 정보를 상기 제1 메타데이터 레코드의 구성요소로서 저장하는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 9

제8항에 있어서,

상기 데이터 저장 과정은,

상기 해당 해시 레코드가 상기 해시정보 테이블 내에 존재하는 경우, 상기 해당 해시 레코드의 참조횟수 정보의 값 및 상기 로그 저장소의 해당 청크의 참조횟수 정보의 값을 증가시키는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 10

제9항에 있어서,

상기 메타데이터 테이블 내의 메타데이터 레코드 중에서 서로 중복된 키를 갖는 중복 키 메타데이터 레코드들을 확인하고, 상기 중복 키 메타데이터 레코드들 중에서 최신 메타데이터 레코드를 제외한 나머지 메타데이터 레코드로부터 상기 로그 저장소 상의 제1 청크 및 상기 해시정보 테이블 내의 제1 해시 레코드를 검색하고, 상기 로그 저장소 상의 상기 제1 청크의 참조횟수 정보의 값 및 상기 제1 해시 레코드의 참조횟수 정보의 값을 감소시키고 상기 나머지 메타데이터 레코드를 삭제하는 가비지 컬렉션 과정을 더 포함하는 키-밸류 데이터 저장 방법.

청구항 11

제10항에 있어서,

상기 가비지 컬렉션 과정은,

상기 제1 청크의 참조횟수 정보의 값 및 상기 제1 해시 레코드의 참조횟수 정보의 값이 각각 0이 된 경우, 상기 제1 청크를 상기 로그 저장소로부터 삭제하는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 12

제10항에 있어서,

상기 가비지 컬렉션 과정은,

상기 나머지 메타데이터 레코드의 구성요소인 오프셋 정보를 기초로 상기 나머지 메타데이터 레코드에 대응되는

상기 제1 청크를 확인하는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 13

제1항에 있어서,

상기 데이터 저장 과정은,

상기 밸류가 복수의 청크로 분할된 경우,

상기 복수의 청크 각각과 관련하여 상기 로그 저장소 상의 저장 위치인 복수의 오프셋 정보를 각각 획득하고 상기 키 및 상기 복수의 오프셋 정보를 구성요소로서 포함하는 상기 제1 메타데이터 레코드를 생성하는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 14

제2항에 있어서,

제1 키에 대한 검색 요청을 수신한 경우, 상기 제1 키를 구성요소로 하는 레코드가 상기 제1 테이블에 존재하는지 여부를 확인하고 상기 제1 테이블에 상기 제1 키를 구성요소로 하는 레코드가 존재하지 않는 경우 상기 제2 테이블에 상기 제1 키를 구성요소로 하는 레코드가 존재하는지 여부를 확인하는 메모리 검색 과정; 및

상기 제2 테이블에 상기 제1 키를 구성요소로 하는 레코드가 존재하지 않는 경우 상기 메타데이터 테이블에 상기 제1 키를 구성요소로 하는 해당 메타데이터 레코드가 존재하는지 여부를 확인하는 메타데이터 검색 과정

를 더 포함하는 키-밸류 데이터 저장 방법.

청구항 15

제14항에 있어서,

상기 메타데이터 테이블에서 상기 해당 메타데이터 레코드를 찾은 경우 상기 해당 메타데이터 레코드에 구성요소로서 포함된 적어도 하나의 오프셋 정보를 획득하여 상기 적어도 하나의 오프셋 정보를 각각 이용하여 상기 로그 저장소로부터 적어도 하나의 제2 청크를 각각 획득하는 청크 획득 과정; 및

상기 해당 메타데이터 레코드 상에 저장된 상기 적어도 하나의 오프셋 정보의 순서에 따라 상기 적어도 하나의 제2 청크로부터 상기 제1 키에 해당하는 밸류를 복원하고 상기 제1 키 및 상기 복원된 밸류를 재구성하여 재구성 레코드를 생성하는 레코드 복원 과정

을 더 포함하는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 16

제14항에 있어서,

상기 메타데이터 검색 과정은,

상기 메타데이터 테이블에 가장 최근에 저장된 메타데이터 레코드 순서로 상기 해당 메타데이터 레코드가 존재하는지 여부를 확인하는 것을 특징으로 하는 키-밸류 데이터 저장 방법.

청구항 17

제1항 내지 제15항 중 어느 한 항에 따른 키-밸류 데이터 저장 방법이 포함하는 각 과정을 실행시키기 위하여 컴퓨터로 읽을 수 있는 기록매체에 저장된 컴퓨터 프로그램.

청구항 18

키 및 밸류를 갖는 제2 레코드의 상기 밸류에 해당하는 부분을 일정 크기의 적어도 하나의 청크로 분할하는 청크 분할부;

상기 분할된 각각의 청크마다 핑거프린트를 생성하고 상기 생성된 핑거프린트와 동일한 핑거프린트를 구성요소로서 포함하는 해시 레코드가 해시정보 테이블 내에 존재하는지 여부를 확인하는 중복청크 확인부; 및

상기 각각의 청크와 관련하여 해당 해시 레코드가 상기 해시정보 테이블 내에 존재하는지 여부에 따라 해당 청

크를 로그 저장소에 저장하고, 상기 각각의 청크마다 상기 로그 저장소 상의 저장 위치인 각각의 오프셋 정보를 획득하고 상기 키 및 상기 각각의 오프셋 정보를 구성요소로서 포함하는 제1 메타데이터 레코드를 생성하여 메타데이터 테이블에 저장하는 데이터 저장부

를 포함하는 키-밸류 데이터 저장 장치.

청구항 19

제18항에 있어서,

상기 데이터 저장부는,

상기 해당 해시 레코드가 상기 해시정보 테이블 내에 존재하지 않는 경우, 상기 해당 청크를 로그 저장소에 저장하고 상기 해당 청크의 상기 로그 저장소 상의 위치를 제1 오프셋 정보로서 획득하여 상기 제1 오프셋 정보를 상기 제1 메타데이터 레코드의 구성요소로서 저장하는 것을 특징으로 하는 키-밸류 데이터 저장 장치.

청구항 20

제18항에 있어서,

상기 데이터 저장부는,

상기 해당 해시 레코드가 상기 해시정보 테이블 내에 존재하는 경우, 상기 해당 해시 레코드의 구성요소인 해당 오프셋 정보를 획득하고 상기 해당 오프셋 정보를 상기 제1 메타데이터 레코드의 구성요소로서 저장하는 것을 특징으로 하는 키-밸류 데이터 저장 장치.

발명의 설명

기 술 분 야

[0001] 본 개시는 중복 제거 기능을 갖는 확장형 LSM 트리 기반 키-밸류 데이터 저장 장치 및 방법에 관한 것이다.

배 경 기 술

[0002] 이하에 기술되는 내용은 단순히 본 실시예와 관련되는 배경 정보만을 제공할 뿐 종래기술을 구성하는 것이 아니다.

[0003] 도 1은 프록시 기반 분산 키-밸류 저장장치(Proxy-based Distributed Key-Value Stores: PD-KVS) 및 NVMe(Non-Volatile Memory Express) SSD(Solid State Drive)를 포함하는 HPC(High-Performance Computing) 시스템(100)을 예시한 도면이다.

[0004] HPC 애플리케이션은 종종 컴퓨팅 및 데이터 집약적이며 오랜 시간 동안 자주 시뮬레이션을 실행한다. HPC 애플리케이션은 내부 상태를 병렬 파일 시스템(Parallel File System: PFS)에 저장하기 위해 주기적으로 체크포인트 동작을 수행한다. 하지만, 일반적으로 HDD(Hard Disk Drive)로 구성되는 PFS에 내부 상태를 저장하는 것은 상당한 입출력 오버헤드를 야기하는 문제가 있다. 따라서, 체크포인트와 같은 고대역폭의 워크 로드를 처리하기 위해, 도 1과 같이 플래시 기반의 추가 스토리지 계층인 NVMe SSD가 배치된 것이다.

[0005] PD-KVS는 HPC 시스템(100)에서 엄청난 양의 체크포인트 데이터를 키 및 밸류의 쌍(KV pair)을 구성요소로서 포함하는 레코드로 저장하는 기능을 제공한다.

[0006] 도 1에 도시된 바와 같이, HPC 시스템(100)은 각 컴퓨팅 노드(CN; 110, 120, 130)의 로컬 SSD(CN-SSD)인 NVMe SSD를 포함하고 그 위에 PD-KVS를 채용하며, PD-KVS는 NVMe SSD의 성능 특성을 최대한 활용하여 키의 질의에 대한 상당한 성능을 제공한다.

[0007] PD-KVS는 백엔드로서 복수의 CN(110, 120, 130) 및 그들 사이의 네트워크 구성을 지원하므로 HPC 설비에서 채택되기에 적합하다. 예컨대, 범위 쿼리를 지원하는 HPC 애플리케이션은 MassTree 기반 스토리지 엔진을 사용하는 CN(130)에서 동작될 수 있으며, PD-KVS를 이용한 쓰기 집약적(write-intensive) HPC 애플리케이션은 LSM 트리(Log-Structured Merge Tree) 기반 스토리지 엔진을 활용하는 CN(110)에서 동작될 수 있다.

[0008] 도 2는 LSM 트리 기반 PD-KVS(200)의 아키텍처 및 운영 흐름을 예시한 도면이다.

[0009] 도 2에 도시된 바와 같이, LSM 트리 기반 스토리지 엔진은 휘발성의 메모리 요소(memory component, 210) 및 비휘발성의 스토리지 기반 요소(storage-based component, 220)를 포함한다. 메모리 요소(210)는 DRAM(Dynamic Random Access Memory)과 같은 휘발성의 메모리로 구현되고 스토리지 기반 요소(220)는 NVMe SSD와 같은 비휘발성 메모리로서 구현될 수 있다.

[0010] 메모리 요소(210) 내에는 MemTable(MT, 211) 및 IMT(Immutable Memtable) 풀(212)이 형성될 수 있고, IMT 풀(212)은 하나 또는 복수의 IMT(212-1, 212-2)로서 형성될 수 있다. MemTable(211)은 변경 가능한 인-메모리 스킵리스트(skiplist) 기반 자료 구조로서 형성될 수 있다.

[0011] HPC 애플리케이션들은 KV 쌍을 입력받은 경우 KV 쌍을 직접 MemTable(211)에 하나의 레코드로서 쓰기 및 업데이트할 수 있으며, 쓰기 및 업데이트에 따라 MemTable(211)의 크기는 점점 증가한다.

[0012] MemTable(211)의 크기가 특정 크기 임계값에 도달하면 해당 MemTable(211)은 IMT(212-1)로 설정되고, IMT(212-1)에 저장된 레코드는 도 2와 같이 백그라운드 스레드 풀(230)의 비동기 스레드들에 의해 스토리지 기반 요소(220)의 레벨 0(L0)에 플러시(flush)될 수 있다.

[0013] 스토리지 기반 요소(220)는 적어도 하나의 레벨을 구성하는 Sorted String Table(SST) 파일들(221, 222, 223, 224)을 포함하며 SST 파일들(221, 222, 223, 224)은 크기가 커지면서 특정 크기 이상이 되는 경우 레벨이 증가되어 저장될 수 있다.

[0014] 레벨 0(L0)의 SST 파일(221) 내의 레코드는 정렬되지 않으며 저장된 레코드의 키 범위가 겹칠 수 있다. 그러나 레벨 0(L0)보다 높은 레벨의 SST 파일(222, 223, 224)의 레코드는 정렬되어 레코드 사이에 겹치는 키 범위가 존재하지 않는다.

[0015] SST 파일들의 정렬 순서 및 LSM 트리 기반 PD-KVS(200)의 조회 성능을 유지하기 위해, 한 레벨이 크기 임계값에 도달하면 압축(compact) 작업이 트리거 된다. 압축 작업은 특정 레벨 n(Ln)의 SST 파일들을 병합하고 정렬하여 레벨 n+1(Ln+1) SST 파일을 생성하며, 이러한 동작은 백그라운드 스레드 풀(230)의 비동기 스레드에 의해 수행된다.

[0016] LSM 트리 기반 PD-KVS(200)에서는 쓰기 증폭(write amplification: WA) 및 공간 증폭(space amplification: SA)이 발생한다. WA는 LSM 트리 기반 PD-KVS(200)에서 HPC 애플리케이션이 의도한 것보다 더 많은 쓰기 작업을 수행할 때 발생하며, 압축 동작과 같은 LSM-tree의 내부 작업으로 인해 발생한다.

[0017] 한편, SA는 LSM 트리 기반 PD-KVS(200)가 HPC 애플리케이션이 필요로 하는 것보다 더 많은 공간을 차지할 때 발생하며, HPC 애플리케이션의 워크로드가 업데이트 위주일 때 주로 발생한다.

[0018] 이때, LSM 트리의 상위 레벨은 여전히 오래된 데이터를 저장한다. LSM-트리는 여러 레벨로 레코드 데이터를 저장하므로 하위 레벨에는 항상 최신의 유효한 KV 쌍 레코드가 존재하고, 상위 레벨에 저장된 KV 쌍 레코드는 오래되어 무효인 데이터가 되기 쉬우며 압축 작업은 이러한 오래된 KV 쌍 레코드 저장 공간을 회수한다.

[0019] 또한, WA로 인해 발생하는 많은 쓰기 횟수는 NVMe SSD의 수명의 단축을 야기한다.

[0020] 도 1과 같이, HPC에서 동작하는 HPC 애플리케이션은 LSM 트리 기반 PD-KVS(200)를 활용하여 중간 상태 및 체크포인트 데이터를 컴퓨팅 노드(CN)의 로컬 SSD(CN-SSD)에 저장한다. HPC 시스템(100)에서 각 컴퓨팅 노드(CN) 별로 LSM 트리 기반 PD-KVS(200)를 지원하는 복수의 독립 인스턴스가 배포되어 HPC 애플리케이션의 고유한 요구사항이 지원될 수 있다.

[0021] 그러나 LSM 트리 기반 PD-KVS(200)는 높은 WA 및 SA에 시달리게 되는 것이 일반적이므로, LSM 트리 기반 PD-KVS(200)는 CN-SSD의 제한된 용량으로 인해 그 성능이 제한되게 된다. 따라서 CN-SSD에 저장되는 데이터의 중복을 최대한 제거함으로써 WA 및 SA에 의한 문제를 최소화하는 것이 바람직하다.

발명의 내용

해결하려는 과제

[0022] 본 개시는 중복 제거 기능을 갖는 확장형 LSM 트리 기반 키-밸류 데이터 저장 장치 및 방법을 제공하는 데에 주된 목적이 있다.

[0023] 본 발명이 해결하고자 하는 과제들은 이상에서 언급한 과제들로 제한되지 않으며, 언급되지 않은 또 다른 과제

들은 아래의 기재로부터 통상의 기술자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0024] 본 개시의 일 측면에 의하면, 키 및 밸류를 구성요소로서 포함하는 제2 레코드의 상기 밸류를 일정 크기의 적어도 하나의 청크로 분할하는 청크 분할 과정; 상기 분할된 각각의 청크마다 핑거프린트를 생성하고 상기 생성된 핑거프린트와 동일한 핑거프린트를 구성요소로서 포함하는 해시 레코드가 해시정보 테이블 내에 존재하는지 여부를 확인하는 중복청크 확인 과정; 및 상기 각각의 청크와 관련한 해당 해시 레코드가 상기 해시정보 테이블 내에 존재하는지 여부에 따라 해당 청크를 로그 저장소에 저장하고, 상기 각각의 청크마다 상기 로그 저장소 상의 저장 위치인 각각의 오프셋 정보를 획득하고 상기 키 및 상기 각각의 오프셋 정보를 구성요소로서 포함하는 제1 메타데이터 레코드를 생성하여 메타데이터 테이블에 저장하는 데이터 저장 과정을 포함하는 키-밸류 데이터 저장 방법을 제공한다.

[0025] 본 개시의 다른 측면에 의하면, 제1항 내지 제15항 중 어느 한 항에 따른 키-밸류 데이터 저장 방법이 포함하는 각 과정을 실행시키기 위하여 컴퓨터로 읽을 수 있는 기록매체에 저장된 컴퓨터 프로그램을 제공한다.

[0026] 본 개시의 다른 측면에 의하면, 키 및 밸류를 갖는 제2 레코드의 상기 밸류에 해당하는 부분을 일정 크기의 적어도 하나의 청크로 분할하는 청크 분할부; 상기 분할된 각각의 청크마다 핑거프린트를 생성하고 상기 생성된 핑거프린트와 동일한 핑거프린트를 구성요소로서 포함하는 해시 레코드가 해시정보 테이블 내에 존재하는지 여부를 확인하는 중복청크 확인부; 및 상기 각각의 청크와 관련하여 해당 해시 레코드가 상기 해시정보 테이블 내에 존재하는지 여부에 따라 해당 청크를 로그 저장소에 저장하고, 상기 각각의 청크마다 상기 로그 저장소 상의 저장 위치인 각각의 오프셋 정보를 획득하고 상기 키 및 상기 각각의 오프셋 정보를 구성요소로서 포함하는 제1 메타데이터 레코드를 생성하여 메타데이터 테이블에 저장하는 데이터 저장부를 포함하는 키-밸류 데이터 저장 장치를 제공한다.

발명의 효과

[0027] 본 개시의 실시예에 의하면, 비휘발성 메모리에 저장되는 키-밸류 쌍 레코드의 사이의 중복이 최대한 제거됨으로써 WA 및 SA에 의한 문제를 최소화하는 효과가 있다.

[0028] 본 개시의 효과들은 이상에서 언급한 효과들로 제한되지 않으며, 언급되지 않은 또 다른 효과들은 아래의 기재로부터 통상의 기술자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

[0029] 도 1은 프록시 기반 분산 키-밸류 저장장치(Proxy-based Distributed Key-Value Stores: PD-KVS) 및 NVMe(Non-Volatile Memory Express) SSD(Solid State Drive)를 포함하는 HPC(High-Performance Computing) 시스템을 예시한 도면이다.

도 2는 LSM 트리 기반 PD-KVS(200)의 아키텍처 및 운영 흐름을 예시한 도면이다.

도 3은 본 개시의 일 실시예에 따른 키-밸류 데이터 저장 장치(300)의 구성을 도시한 블록도이다.

도 4는 LSM 트리 기반의 키-밸류 데이터 저장 장치(300)의 동작 흐름을 예시한 도면이다.

도 5는 해시정보 테이블의 형태를 예시한 도면이다.

도 6은 메타데이터 테이블의 형태를 예시한 도면이다.

도 7은 로그 저장소의 형태를 예시한 도면이다.

도 8은 키-밸류 데이터 저장 장치(300)가 본 개시의 일 실시예에 따른 키-밸류 데이터 저장 방법을 도시한 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0030] 이하, 본 개시의 일부 실시예들을 예시적인 도면을 이용해 상세하게 설명한다. 각 도면의 구성 요소들에 참조부호를 부가함에 있어서, 동일한 구성 요소들에 대해서는 비록 다른 도면 상에 표시되더라도 가능한 한 동일한 부호를 가지도록 하고 있음에 유의해야 한다. 또한, 본 개시를 설명함에 있어, 관련된 공지 구성 또는 기능에

대한 구체적인 설명이 본 개시의 요지를 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명은 생략한다.

[0031] 본 개시에 따른 실시예의 구성요소를 설명하는 데 있어서, 제1, 제2, i), ii), a), b) 등의 부호를 사용할 수 있다. 이러한 부호는 그 구성요소를 다른 구성 요소와 구별하기 위한 것일 뿐, 그 부호에 의해 해당 구성요소의 본질 또는 차례나 순서 등이 한정되지 않는다. 명세서에서 어떤 부분이 어떤 구성요소를 '포함' 또는 '구비'한다고 할 때, 이는 명시적으로 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있는 것을 의미한다.

[0032] 첨부된 도면과 함께 이하에 개시될 상세한 설명은 본 개시의 예시적인 실시형태를 설명하고자 하는 것이며, 본 개시가 실시될 수 있는 유일한 실시형태를 나타내고자 하는 것이 아니다.

[0033] 도 3은 본 개시의 일 실시예에 따른 키-밸류 데이터 저장 장치(300)의 구성을 도시한 블록도이고, 도 4는 LSM 트리 기반의 키-밸류 데이터 저장 장치(300)의 동작 흐름을 예시한 도면이다.

[0034] 키-밸류 데이터 저장 장치(300)는 HPC 시스템(100)의 하나의 컴퓨팅 노드(110)에서 동작될 수 있다.

[0035] 도 3에 도시되듯이, 키-밸류 데이터 저장 장치(300)는 키-밸류 저장부(310), 제2테이블 설정부(315), 제2레코드 획득부(320), 청크 분할부(325), 중복청크 확인부(330), 데이터 저장부(335), 메모리 검색부(340), 메타데이터 검색부(345), 청크 획득부(350), 레코드 복원부(360) 및 가비지 컬렉션부(370)를 포함한다. 실시예에 따라서, 키-밸류 데이터 저장 장치(300)는 도 3의 구성요소 중 일부 구성요소는 생략하여 구현되거나 다른 구성요소를 추가하여 구현될 수도 있다.

[0036] 도 4에 도시된 바와 같이, 키-밸류 데이터 저장 장치(300)는 LSM 트리 기반 스토리지 상에서 동작하며, LSM 트리 기반 스토리지는 휘발성의 메모리 요소(410) 및 비휘발성의 스토리지 기반 요소(420)를 포함한다. 메모리 요소(410)는 DRAM과 같은 휘발성의 메모리로 구현되고 스토리지 기반 요소(420)는 NVMe SSD와 같은 비휘발성 메모리로서 구현될 수 있으나 본 발명이 이에 한정되는 것은 아니다.

[0037] 메모리 요소(410) 내에는 MemTable(411) 및 IMT 풀(412)을 포함할 수 있고, 스토리지 기반 요소(420) 내에는 복수의 메타데이터 테이블(421-1, 421-2, 421-3) 및 로그 저장소(422)를 포함할 수 있다.

[0038] MemTable(411)은 변경 가능한 인-메모리 스킵리스트 기반 자료 구조로서 형성될 수 있고, IMT 풀(412)은 적어도 하나의 IMT(412-1, 412-2)를 포함하도록 형성될 수 있다.

[0039] 키-밸류 저장부(310)는 키 및 밸류의 쌍에 대한 데이터를 수신하여 키-밸류 쌍 레코드로서 제1 테이블, 즉 Memtable(411)에 저장한다. 여기서 키 및 밸류의 쌍은 컴퓨팅 노드(110) 내의 다른 애플리케이션으로부터 수신될 수 있다.

[0040] 키-밸류 저장부(310)가 키-밸류 쌍 레코드를 MemTable(411)에 쓰기 동작 또는 업데이트 동작을 수행함에 따라 MemTable(411)의 크기는 점점 증가한다.

[0041] 제2테이블 설정부(315)는 Memtable(411)의 크기, 즉 Memtable(411)에 저장된 모든 키-밸류 쌍 레코드의 용량의 합이 제1 임계치 이상인 경우 MemTable(411)을 제2 테이블 집합에 포함되는 하나의 제2 테이블로서 설정한다. 여기서, 제2 테이블로서 설정한다는 의미는 이 테이블은 제2 테이블이라는 표시를 하는 것을 의미하며 제2 테이블 내부에 저장된 데이터의 변동을 수반하지 않을 수 있다.

[0042] 제2 테이블은 IMT(412-1, 412-2)로서 구현될 수 있다. 제2 테이블 집합, 즉 IMT 풀(412)은 적어도 하나의 IMT(412-1, 412-2)로서 구현되며, IMT 풀(412) 내의 IMT(412-1, 412-2)의 임계 개수(즉, 제2 임계치)가 결정된다. MemTable(411)에 키-밸류 쌍 레코드가 저장될수록 IMT(412-1, 412-2)가 하나씩 증가하면서 IMT 풀(412) 내의 IMT(412-1, 412-2)의 개수는 임계 개수만큼 증가될 수 있다.

[0043] 한편, 키-밸류 저장부(310)는 MemTable(411)이 하나의 IMT(412-1)로 설정된 후 새로운 키-밸류 쌍을 수신하는 경우, 새로운 MemTable을 생성하여 새로운 키-밸류 쌍을 새로운 MemTable에 저장하도록 구현된다.

[0044] 제2레코드 획득부(320)는 IMT 풀(412) 내의 IMT(412-1, 412-2)의 개수가 제2 임계치 이상인 경우, IMT 풀(412) 내의 IMT(412-1, 412-2)의 키-밸류 쌍 레코드를 순차적으로 획득하여 제2 레코드로 설정한다.

[0045] 제2레코드 획득부(320)는 백그라운드 스레드 풀(430)의 비동기 스레드들을 이용하여 IMT(412-1, 412-2)의 키-밸류 쌍 레코드를 순차적으로 획득하도록 구현될 수 있다.

[0046] 청크 분할부(325)는 키 필드 및 밸류 필드를 갖는 하나의 제2 레코드에서 밸류 필드에 해당하는 부분을 일정 크

기의 적어도 하나의 청크로 분할한다. 즉, 청크 분할부(325)는 밸류 필드 부분을 기설정된 청크 사이즈 단위로 나누어 복수의 청크로 분할하도록 구현될 수 있다.

[0047] 도 5는 해시정보 테이블의 형태를 예시한 도면이다.

[0048] 도 5에 도시된 바와 같이, 해시정보 테이블(500)은 복수의 해시 레코드(510, 520, 530)를 저장하도록 구성되며 해시정보 테이블(500)은 DRAM 상에 저장될 수 있다. 각 해시 레코드(510, 520, 530)는 핑거프린트 필드, 오프셋 필드 및 참조횟수 필드를 구성요소로서 포함한다.

[0049] 초기 상태의 해시정보 테이블(500)은 비어 있는 상태가 되며, 제2 레코드의 키-밸류 쌍 레코드가 스토리지 기반 요소(420)인 NVMe SSD에 저장됨에 따라 해시정보 테이블(500)에 해시 레코드(510, 520, 530)가 순차적으로 저장될 수 있다.

[0050] 중복청크 확인부(330)는 분할된 각각의 청크에 해시 알고리즘을 적용하여 핑거프린트를 생성하고 생성된 핑거프린트와 동일한 핑거프린트를 구성요소로서 포함하는 해시 레코드가 해시정보 테이블(500) 내에 존재하는지 여부를 확인한다. 예컨대, 하나의 밸류에 대하여 분할된 청크의 개수가 n개인 경우, 중복청크 확인부(330)는 n개의 분할된 청크마다 각각 핑거프린트를 생성한다. 중복청크 확인부(330)는 생성된 n개의 핑거프린트마다 해당 핑거프린트와 동일한 핑거프린트 필드가 저장된 해시 레코드가 해시정보 테이블(500) 내에 존재하는지 여부를 확인한다.

[0051] 여기서 핑거프린트는 SHA-1(Secure Hash Algorithm 1) 알고리즘 등 다양한 해시 알고리즘에 의해 생성될 수 있다.

[0052] 해시정보 테이블(500) 내의 해시 레코드(510, 520, 530)들은 서로 다른 핑거프린트를 구성요소로서 갖는다. 즉, 서로 중복된 핑거프린트는 해시정보 테이블(500) 내에 존재하지 않는다.

[0053] 도 5에서 각 해시 레코드(510, 520, 530)는 핑거프린트 필드, 해당 핑거프린트에 대응하는 청크의 로그 저장소(422) 상의 저장 위치를 나타내는 오프셋 필드 및 참조횟수 필드를 구성요소로서 포함한다.

[0054] 핑거프린트 필드, 오프셋 필드 및 참조횟수 필드에 대한 상세한 사항은 후술한다.

[0055] 도 6은 메타데이터 테이블의 형태를 예시한 도면이다.

[0056] 도 6에 도시된 바와 같이, 하나의 메타데이터 테이블, 예컨대 제1 메타데이터 테이블(421-1)은 복수의 메타데이터 레코드(610, 620, 630)를 포함할 수 있도록 구성되며, 각 메타데이터 레코드(610, 620, 630)는 키 필드와 하나 이상의 오프셋 필드(즉, 오프셋리스트 필드)를 구성요소로서 포함하도록 구현된다.

[0057] 도 6에서, 키 필드와 오프셋리스트 필드에 대한 사항은 후술한다.

[0058] 도 7은 로그 저장소의 형태를 예시한 도면이다.

[0059] 도 7에 도시된 바와 같이, 로그 저장소(422)에는 청크 저장부(710) 및 참조횟수 저장부(720)를 포함한다. 청크 저장부(710)에는 복수의 청크가 저장될 수 있으며, 참조횟수 저장부(720)에는 각 청크의 참조횟수 정보가 저장될 수 있다.

[0060] 메타데이터 테이블(421-1, 421-2, 421-3) 및 로그 저장소(422)는 NVMe SSD와 같은 비휘발성 메모리에 저장된다.

[0061] 데이터 저장부(335)는, 제2 레코드의 각각의 청크마다 해당 핑거프린트를 갖는 해시 레코드가 해시정보 테이블(500) 내에 존재하는지 여부에 따라 해당 청크를 청크 저장부(710)에 저장하고, 각각의 청크마다 로그 저장소(422) 상의 저장 위치인 각각의 오프셋 정보를 획득하고, 해당 제2 레코드의 키 및 해당하는 각각의 오프셋 정보를 구성요소로서 포함하는 제1 메타데이터 레코드를 생성하여 하나의 메타데이터 테이블(421-1)에 저장하도록 구현될 수 있다.

[0062] 하나의 청크에 해당하는 핑거프린트를 갖는 해시 레코드가 해시정보 테이블(500) 내에 존재하지 않는 경우, 데이터 저장부(335)는 해당 청크를 청크 저장부(710)에 저장하고 참조횟수 저장부(720)에 해당 청크의 참조횟수 정보를 1로 저장한다. 데이터 저장부(335)는 해당 청크의 로그 저장소(700) 상의 위치를 오프셋 정보로서 획득하고 제2 레코드의 키와 획득된 오프셋 정보를 제1 메타데이터 레코드의 구성요소로 저장한다. 여기서, 해당 청크의 로그 저장소(700) 상의 위치는 로그 저장소(700) 상의 해당 청크의 시작 주소일 수 있다. 데이터 저장부(335)는 생성된 핑거프린트 및 획득된 오프셋 정보를 구성요소로서 포함하는 해시 레코드를 생성하여 해시정보 테이블(500)에 저장한다.

[0063] 하나의 청크에 해당하는 핑거프린트를 갖는 제1 해시 레코드가 해시정보 테이블(500) 내에 존재하는 경우, 데이터 저장부(335)는 해당 청크를 청크 저장부(710)에 저장하지 않는다. 이때, 데이터 저장부(335)는 해시정보 테이블(500)로부터 제1 해시 레코드의 구성요소인 제1 오프셋을 획득하고 제1 오프셋을 제1 메타데이터 레코드의 구성요소로서 저장한다. 또한, 데이터 저장부(335)는 해당 청크의 참조횟수를 1만큼 증가시켜 참조횟수 저장부(720)에 저장하고 해시 테이블(500)에서 제1 해시 레코드의 참조횟수 필드의 값을 1만큼 증가시킨다.

[0064] 이하, 제2 레코드가 제1 키 및 제1 밸류 쌍을 갖는 레코드이고, 제1 밸류가 2개의 청크인 제1 청크 및 제2 청크로 분할된 경우를 가정하여 설명한다.

[0065] 먼저, 중복청크 확인부(330)는 제1 청크에 대하여 해시 알고리즘을 적용하여 제1 핑거프린트를 생성한다. 중복청크 확인부(330)는 해시정보 테이블(500) 내의 해시 레코드(510, 520, 530)들을 탐색하여 제1 핑거프린트와 동일한 핑거프린트를 구성요소로서 포함하는 해시 레코드가 해시정보 테이블(500) 내에 존재하는지 여부를 확인한다.

[0066] 해시정보 테이블(500) 내에서 제1 핑거프린트와 동일한 핑거프린트를 갖는 해시 레코드가 제1 해시 레코드(510)인 것으로 확인된 경우, 이미 제1 청크와 동일한 데이터가 청크 저장부(710)에 저장되어 있는 것이므로 데이터 저장부(335)는 제1 청크를 청크 저장부(710)에 다시 저장하지는 않는다. 이때, 데이터 저장부(335)는 해시정보 테이블(500)로부터 제1 해시 레코드(520)의 오프셋 필드인 오프셋1(예컨대 P_{11}^{*})를 획득하고 제1 키와 제1 오프셋 P_{11}^{*}을 각각 구성요소로 하는 제1 메타데이터 레코드(610)를 생성한다. 여기서, P_{11}^{*}은 로그 저장소(700) 상의 해당 청크(711)의 시작 주소를 가리키는 포인터이다. 또한, 데이터 저장부(335)는 오프셋1(P_{11}^{*})이 참조하는 청크(711)에 해당하는 참조횟수를 1만큼 증가시켜 참조횟수 저장부(720)에 저장하고 해시 테이블(500)에서 제1 해시 레코드의 참조횟수 필드의 값을 1만큼 증가시킨다.

[0067] 제1 청크의 처리가 완료된 경우, 중복청크 확인부(330)는 제2 청크에 대하여 해시 알고리즘을 적용하여 제2 핑거프린트를 생성한다. 중복청크 확인부(330)는 해시정보 테이블(500) 내의 해시 레코드(510, 520, 530)들을 탐색하여 제2 핑거프린트와 동일한 핑거프린트를 구성요소로서 포함하는 해시 레코드가 해시정보 테이블(500) 내에 존재하는지 여부를 확인한다.

[0068] 만일, 해시정보 테이블(500) 내에 제2 핑거프린트를 갖는 해시 레코드가 존재하지 않는다면, 데이터 저장부(335)는 제2 청크를 청크 저장부(710)에 저장하고 참조횟수 저장부(720)에 제2 청크의 참조횟수 정보를 1로 저장한다. 데이터 저장부(335)는 제2 청크의 로그 저장소(700) 상의 위치인 P_{12}^{*}를 제2 오프셋으로서 획득하고 제2 오프셋 P_{12}^{*}를 제1 메타데이터 레코드의 구성요소로서 추가한다. 여기서, P_{12}^{*}는 로그 저장소(700) 상의 해당 청크(712)의 시작 주소를 가리키는 포인터이다. 또한, 데이터 저장부(335)는 제2 핑거프린트 및 제2 오프셋 P_{12}^{*}를 구성요소로서 포함하는 새로운 제4 해시 레코드를 생성하여 해시정보 테이블(500)에 저장한다. 제4 해시 레코드의 참조 횟수 정보는 1로 설정된다.

[0069] 위에서 설명한 바와 같이, 제2 레코드의 제1 밸류에 대한 모든 청크에 대하여 오프셋 정보의 획득이 완료되면, 데이터 저장부(335)는 생성된 제1 메타데이터 레코드(610)를 하나의 메타데이터 테이블(421-1)에 저장한다. 여기서, 제1 메타데이터 레코드(610)에는 키 필드에 제1 키가 저장되고 오프셋 리스트 필드에 P_{11}^{*} 및 P_{12}^{*}가 저장된다.

[0070] 하나의 키-밸류 쌍에 해당하는 레코드마다 각각 하나의 메타데이터 레코드가 생성되어 메타데이터 테이블(421-1, 421-2, 421-3)에 저장된다.

[0071] IMT 풀(412)로부터 획득된 레코드들로부터 생성된 메타데이터 레코드들은 스토리지 기반 요소(420)의 레벨 0(L0)에 Sorted String Table(SST) 파일로서 저장될 수 있다.

[0072] 스토리지 기반 요소(420)는 적어도 하나의 레벨을 구성하는 SST 파일들을 포함한다. 레벨 0의 SST 파일은 크기가 커져서 특정 크기 이상이 되는 경우 중복이 제거되고 정렬되어 더 높은 레벨(예컨대, 레벨 1)의 SST 파일로서 저장될 수 있다.

[0073] 한편, 키-밸류 저장부(310)는, 키-밸류 쌍이 수신되는 경우 데이터 일관성을 위해 WAL(Write Ahead Log)를 위한

공간을 SSD에 확보하여 수신된 키-밸류 쌍을 WAL(미도시)에 저장하도록 구현될 수 있다.

[0074] 메모리 검색부(340)는 제1 키에 대한 검색 요청을 HPC 애플리케이션으로부터 수신한 경우, 제1 키를 구성요소로 하는 해당 키-밸류 쌍 레코드가 MemTable(411)에 존재하는지 여부를 확인한다. MemTable(411)에 제1 키를 구성요소로 하는 키-밸류 쌍 레코드가 존재하면 메모리 검색부(340)는 해당 키-밸류 쌍 레코드를 HPC 애플리케이션에게 제공한다.

[0075] MemTable(411)에 제1 키를 구성요소로 하는 키-밸류 쌍이 존재하지 않는 경우, 메모리 검색부(340)는 제1 키를 구성요소로 하는 키-밸류 쌍 레코드가 IMT 풀(412)에 존재하는지 여부를 확인한다. IMT 풀(412)에 제1 키를 구성요소로 하는 키-밸류 쌍 레코드가 존재하면 메모리 검색부(340)는 해당 키-밸류 쌍 레코드를 HPC 애플리케이션에게 제공한다.

[0076] IMT 풀(412)에 제1 키를 구성요소로 하는 키-밸류 쌍 레코드가 존재하지 않는 경우, 메타데이터 검색부(345)는 메타데이터 테이블(421-1, 421-2, 421-3)에 제1 키를 구성요소로 하는 메타데이터 레코드가 존재하는지 여부를 확인한다.

[0077] 메타데이터 검색부(345)는 메타데이터 테이블(421-1, 421-2, 421-3)에 가장 최근에 저장된 메타데이터 레코드 순서로 제1 키를 구성요소로 하는 메타데이터 레코드가 존재하는지 여부를 확인한다.

[0078] 메타데이터 테이블(421-1, 421-2, 421-3)에서 제1 키를 구성요소로 하는 메타데이터 레코드를 찾은 경우, 청크 획득부(350)는 제1 키를 구성요소로 하는 메타데이터 레코드에 포함된 오프셋 리스트 필드에 저장된 오프셋 정보들을 획득하여 획득된 오프셋 정보들을 각각 이용하여 로그 저장소(710)로부터 해당 청크를 각각 획득한다.

[0079] 레코드 복원부(360)는 오프셋 리스트 필드에 저장된 오프셋 정보의 순서에 따라 순차적으로 각 오프셋 정보에 해당하는 로그 저장소(710) 상의 위치의 청크를 획득한다. 레코드 복원부(360)는 순차적으로 획득된 청크들을 순차적으로 결합하여 제1 키에 대응하는 밸류를 복원한다. 레코드 복원부(360)는 제1 키 및 복원된 밸류 쌍을 포함하도록 재구성 레코드를 생성하여 HPC 애플리케이션에게 출력한다.

[0080] 로그 저장소(422)에는 중복되지 않는 고유 청크들만 저장되지만, 어떤 키에 의해서도 참조되지 않게 된 청크는 제거됨으로써 해당 청크에 의해 사용되던 공간이 회수되는 가비지 컬렉션 동작이 가비지 컬렉션부(365)에 의해 주기적으로 수행될 필요가 있다.

[0081] 가비지 컬렉션부(365)는 메타데이터 테이블(421-1, 421-2, 421-3) 내의 메타데이터 레코드 중에서 서로 중복된 키를 갖는 중복 키 메타데이터 레코드들을 확인한다. 가비지 컬렉션부(365)는 중복 키 메타데이터 레코드들 중에서 최신 메타데이터 레코드를 제외한 나머지 메타데이터 레코드에 대하여 가비지 컬렉션 동작을 수행한다.

[0082] 가비지 컬렉션부(365)는 나머지 메타데이터 레코드의 오프셋 리스트를 확인하여 n개의 오프셋 정보를 획득한다. 로그 저장소(422) 상의 청크들 중에서 n개의 오프셋 정보에 의해 식별되는 제1 청크 및 해시정보 테이블(500) 내의 제1 해시 레코드를 검색한다. 즉, 가비지 컬렉션부(365)는 n개의 오프셋 정보 각각에 의해 식별되는 로그 저장소(422) 상의 위치에서 제1 청크를 각각 확인하고, n개의 오프셋 정보를 가지는 제1 해시 레코드를 각각 확인한다.

[0083] 가비지 컬렉션부(365)는 제1 청크의 참조횟수 정보의 값을 1만큼 감소시키고 제1 해시 레코드의 참조횟수 정보의 값을 1만큼 감소시키고 나머지 메타데이터 레코드를 삭제한다.

[0084] 가비지 컬렉션부(365)는, 제1 청크의 참조횟수 정보의 값 및 제1 해시 레코드의 참조횟수 정보의 값이 각각 0이 된 경우, 제1 청크를 로그 저장소(422)로부터 삭제하여 그 저장공간을 회수한다. 가비지 컬렉션부(365)는 저장공간에서 삭제된 제1 청크에 해당하는 해시 레코드를 해시 테이블(500)로부터 삭제한다.

[0085] 도 8은 키-밸류 데이터 저장 장치(300)가 본 개시의 일 실시예에 따른 키-밸류 데이터 저장 방법을 도시한 흐름도이다.

[0086] 키-밸류 저장부(310)는 키 및 밸류를 수신하여 키-밸류 쌍 레코드로서 MemTable(411)에 저장하는 키-밸류 저장 과정을 수행한다(S810).

[0087] 제2테이블 설정부(315)는 MemTable(411)에 저장된 모든 키-밸류 쌍 레코드의 용량의 합이 제1 임계치 이상인 경우 MemTable(411)을 IMT 풀(412)에 포함되는 하나의 IMT(412-1, 412-2, 412-3)로서 설정하는 제2테이블 설정 과정을 수행한다(S815).

[0088] 제2레코드 획득부(320)는 IMT 풀(412) 내의 IMT(412-1, 412-2, 412-3)의 개수가 제2 임계치 이상인 경우, IMT(412-1, 412-2, 412-3) 내의 키-밸류 쌍 레코드를 순차적으로 획득하여 제2레코드로 설정하는 제2레코드 획득 과정을 수행한다(S820).

[0089] 청크 분할부(325)는 키 및 밸류를 갖는 제2 레코드에서 밸류에 해당하는 부분을 일정 크기의 적어도 하나의 청크로 분할하는 청크 분할 과정을 수행한다(S825).

[0090] 중복청크 확인부(830)는 분할된 각각의 청크마다 핑거프린트를 생성하고 생성된 핑거프린트와 동일한 핑거프린트를 구성요소로서 포함하는 해시 레코드가 해시정보 테이블(500) 내에 존재하는지 여부를 확인하는 중복청크 확인 과정을 수행한다(S830).

[0091] 데이터 저장부(335)는 각각의 청크와 관련하여 해당 해시 레코드가 해시정보 테이블(500) 내에 존재하는지 여부에 따라 해당 청크를 로그 저장소(422)에 저장하고, 각각의 청크마다 로그 저장소(422) 상의 저장 위치인 각각의 오프셋 정보를 획득하고 해당 키 및 각각의 오프셋 정보를 구성요소로서 포함하는 제1 메타데이터 레코드를 생성하여 메타데이터 테이블(421-1, 421-2, 421-3)에 저장하는 데이터 저장 과정을 수행한다(S835).

[0092] 메모리 검색부(340)는 제1 키에 대한 검색 요청을 수신한 경우, 제1 키를 구성요소로 하는 해당 키-밸류 쌍 레코드가 MemTable(411)에 존재하는지 여부를 확인하고 MemTable(411)에 해당 키-밸류 쌍이 존재하지 않는 경우 IMT 풀(412)에 해당 키-밸류 쌍이 존재하는지 여부를 확인하는 메모리 검색 과정을 수행한다(S840).

[0093] 메타데이터 검색부(845)는 IMT 풀(412)에 해당 키-밸류 쌍이 존재하지 않는 경우 메타데이터 테이블(421-1, 421-2, 421-3)에 제1 키를 구성요소로 하는 해당 메타데이터 레코드가 존재하는지 여부를 확인하는 메타데이터 검색 과정을 수행한다(S845).

[0094] 청크 획득부(350)는 메타데이터 테이블(421-1, 421-2, 421-3)에서 해당 메타데이터 레코드를 찾은 경우 해당 메타데이터 레코드에 구성요소로서 포함된 적어도 하나의 오프셋 정보를 획득하여 적어도 하나의 오프셋 정보를 각각 이용하여 로그 저장소(422)로부터 적어도 하나의 청크를 각각 획득하는 청크 획득 과정을 수행한다(S850).

[0095] 레코드 복원부(360)는 해당 메타데이터 레코드 상에 저장된 적어도 하나의 오프셋 정보의 순서에 따라 적어도 하나의 청크로부터 제1 키에 해당하는 밸류를 복원하고 제1 키 및 복원된 밸류 쌍을 재구성하여 재구성 레코드를 생성하는 레코드 복원 과정을 수행한다(S860).

[0096] 가비지 컬렉션부(365)는 메타데이터 테이블(421-1, 421-2, 421-3) 내의 메타데이터 레코드 중에서 서로 중복된 키를 갖는 중복 키 메타데이터 레코드들을 확인한다. 가비지 컬렉션부(365)는 중복 키 메타데이터 레코드들 중에서 최신 메타데이터 레코드를 제외한 나머지 메타데이터 레코드에 대하여 가비지 컬렉션 과정을 수행한다(S870).

[0097] 도 8은 중복 제거 비율이 다른 4KB 및 1MB 크기 KV 쌍의 처리량 평가 결과를 보여주는 그래프이다.

[0098] 본 실시예에 따른 키-밸류 데이터 저장 장치(300)는 작은 KV 쌍에서 중복 제거 비율이 90%일 때 중복 제거 비율이 증가함에 따라 성능이 상대적으로 더 좋아져서 RocksDB보다 성능이 우수함을 알 수 있다.

[0099] 도 9는 WA 및 SA에 대한 비교 분석한 결과를 도시한 그래프이다.

[0100] 도 9에 도시된 바와 같이, 본 실시예에 따른 키-밸류 데이터 저장 장치(300)는 RocksDB 및 BlobDB보다 더 우수한 WA 및 SA 성능을 보임을 알 수 있다.

[0101] 본 발명에 따른 장치 또는 방법의 각 구성요소는 하드웨어 또는 소프트웨어로 구현되거나, 하드웨어 및 소프트웨어의 결합으로 구현될 수 있다. 또한, 각 구성요소의 기능이 소프트웨어로 구현되고 마이크로프로세서가 각 구성요소에 대응하는 소프트웨어의 기능을 실행하도록 구현될 수도 있다.

[0102] 본 명세서에 설명되는 시스템들 및 기법들의 다양한 구현예들은, 디지털 전자 회로, 집적회로, FPGA(field programmable gate array), ASIC(application specific integrated circuit), 컴퓨터 하드웨어, 펌웨어, 소프트웨어, 및/또는 이들의 조합으로 실현될 수 있다. 이러한 다양한 구현예들은 프로그래밍가능 시스템 상에서 실행 가능한 하나 이상의 컴퓨터 프로그램들로 구현되는 것을 포함할 수 있다. 프로그래밍가능 시스템은, 저장 시스템, 적어도 하나의 입력 디바이스, 그리고 적어도 하나의 출력 디바이스로부터 데이터 및 명령들을 수신하고 이들에게 데이터 및 명령들을 전송하도록 결합되는 적어도 하나의 프로그래밍가능 프로세서(이것은 특수 목적 프로세서일 수 있거나 혹은 범용 프로세서일 수 있음)를 포함한다. 컴퓨터 프로그램들(이것은 또한 프로그램들, 소프트웨어, 소프트웨어 애플리케이션들 혹은 코드로서 알려져 있음)은 프로그래밍가능 프로세서에 대한 명령어

들을 포함하며 "컴퓨터가 읽을 수 있는 기록매체"에 저장된다.

[0103] 컴퓨터가 읽을 수 있는 기록매체는, 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록장치를 포함한다. 이러한 컴퓨터가 읽을 수 있는 기록매체는 ROM, CD-ROM, 자기 테이프, 플로피디스크, 메모리 카드, 하드 디스크, 광자기 디스크, 스토리지 디바이스 등의 비휘발성(non-volatile) 또는 비일시적인(non-transitory) 매체일 수 있으며, 또한 데이터 전송 매체(data transmission medium)와 같은 일시적인(transitory) 매체를 더 포함할 수도 있다. 또한, 컴퓨터가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어, 분산방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수도 있다.

[0104] 본 명세서의 흐름도/타이밍도에서는 각 과정들을 순차적으로 실행하는 것으로 기재하고 있으나, 이는 본 개시의 일 실시예의 기술 사상을 예시적으로 설명한 것에 불과한 것이다. 다시 말해, 본 개시의 일 실시예가 속하는 기술 분야에서 통상의 지식을 가진 자라면 본 개시의 일 실시예의 본질적인 특성에서 벗어나지 않는 범위에서 흐름도/타이밍도에 기재된 순서를 변경하여 실행하거나 각 과정들 중 하나 이상의 과정을 병렬적으로 실행하는 것으로 다양하게 수정 및 변형하여 적용 가능할 것이므로, 흐름도/타이밍도는 시계열적인 순서로 한정되는 것은 아니다.

[0105] 이상의 설명은 본 실시예의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 본 실시예가 속하는 기술 분야에서 통상의 지식을 가진 자라면 본 실시예의 본질적인 특성에서 벗어나지 않는 범위에서 다양한 수정 및 변형이 가능할 것이다. 따라서, 본 실시예들은 본 실시예의 기술 사상을 한정하기 위한 것이 아니라 설명하기 위한 것이고, 이러한 실시예에 의하여 본 실시예의 기술 사상의 범위가 한정되는 것은 아니다. 본 실시예의 보호 범위는 아래의 청구범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술 사상은 본 실시예의 권리범위에 포함되는 것으로 해석되어야 할 것이다.

부호의 설명

[0106] 100: HPC 시스템 110, 120, 130: CN

200: LSM 트리 기반 PD-KVS 210; 메모리 요소

211: MemTable 212: IMT 풀

220: 스토리지 기반 요소 230: 백그라운드 스레드 풀

300: 키-밸류 데이터 저장 장치

310: 키-밸류 저장부 315: 제2테이블 설정부

320: 제2레코드 획득부 325; 청크 분할부

330: 중복청크 확인부 335: 데이터 저장부

340: 메모리 검색부 345: 메타데이터 검색부

350: 청크 획득부 360: 레코드 복원부

370: 가비지 컬렉션부

421-1, 421-2, 421-3: 메타데이터 테이블

422: 로그 저장소 430: 백그라운드 스레드 풀

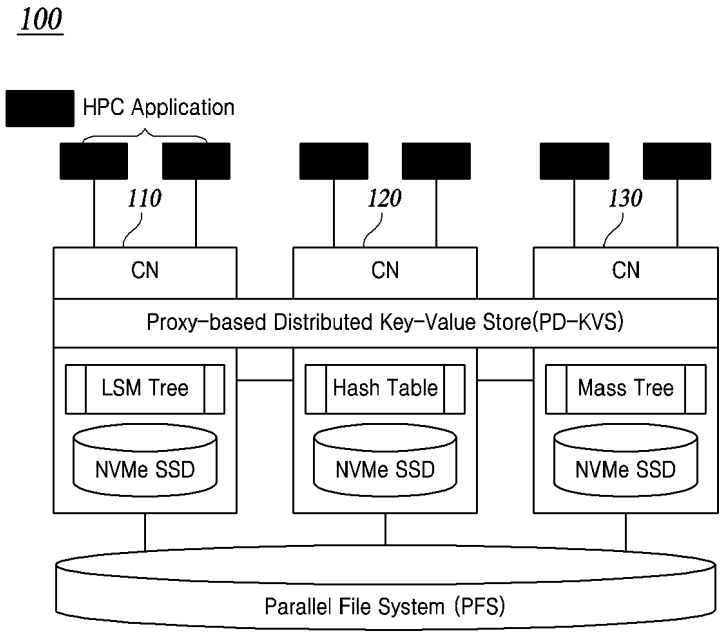
500: 해시정보 테이블 510, 520, 530: 해시 레코드

610, 620, 630: 메타데이터 레코드

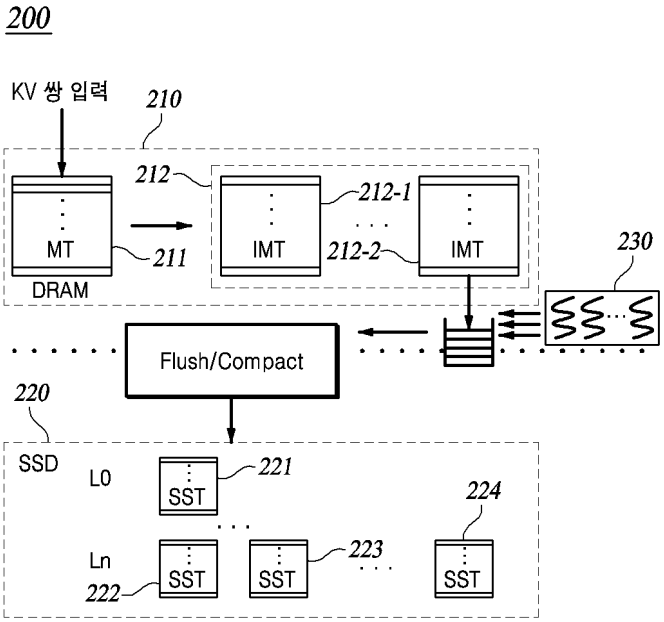
710: 청크 저장부 720: 참조횟수 저장부

도면

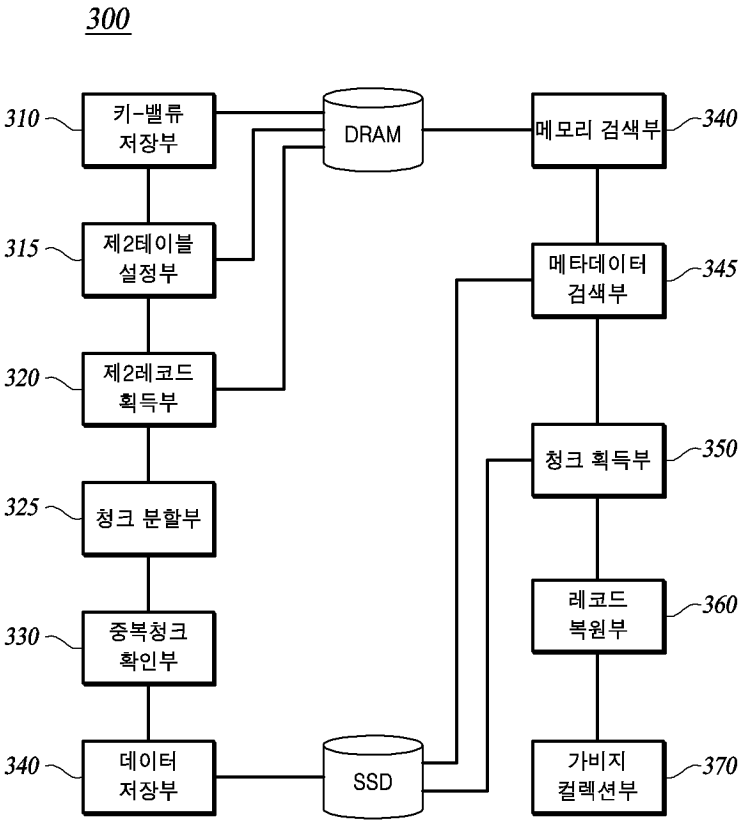
도면1


100
HPC Application
110
120
130
CN
CN
CN
Proxy-based Distributed Key-Value Store(PD-KVS)
LSM Tree
Hash Table
Mass Tree
NVMe SSD
NVMe SSD
NVMe SSD
Parallel File System (PFS)

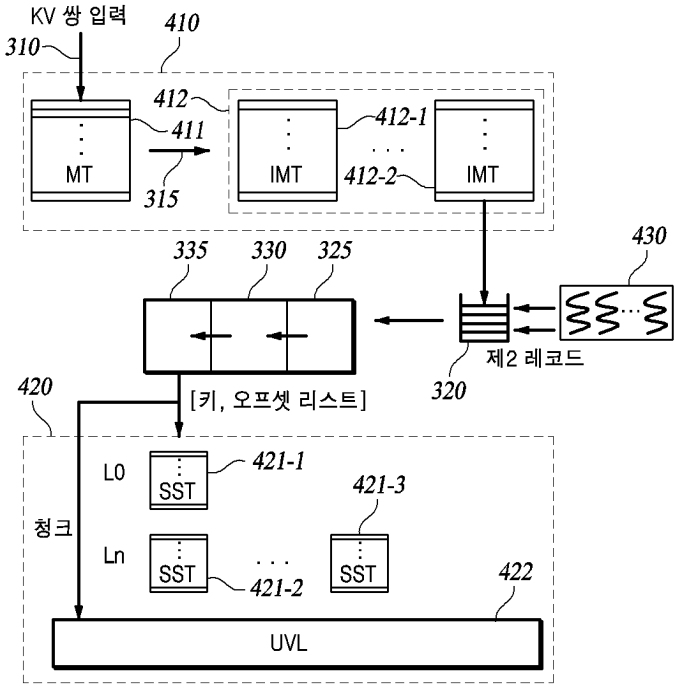
도면2


200
KV 쌍 입력
210
212
212-1
212-2
MT
211
IMT
IMT
DRAM
230
Flush/Compact
220
SSD
L0
221
SST
224
Ln
222
SST
SST
223
SST

도면3


300
310
키-밸류 저장부
DRAM
메모리 검색부
340
315
제2테이블 설정부
메타데이터 검색부
345
320
제2레코드 획득부
청크 획득부
350
325
청크 분할부
레코드 복원부
360
330
중복청크 확인부
340
데이터 저장부
SSD
가비지 컬렉션부
370

도면4


KV 쌍 입력
310
410
412
411
315
MT
IMT
412-1
412-2
IMT
430
335
330
325
제2 레코드
320
420
[키, 오프셋 리스트]
청크
L0
SST
421-1
421-3
Ln
SST
421-2
SST
422
UVL

도면5

500

	핑거프린트	오프셋	참조횟수
510	핑거프린트 1	오프셋 1	제1 참조횟수
520	핑거프린트 2	오프셋 2	제2 참조횟수
530	핑거프린트 3	오프셋 3	제3 참조횟수
	⋮	⋮	⋮

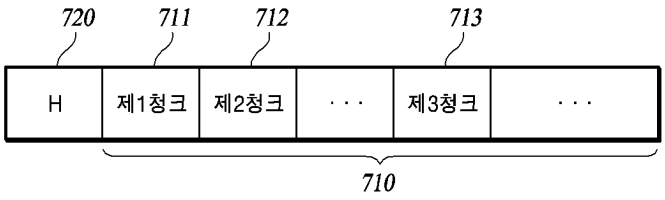
도면6

421-1

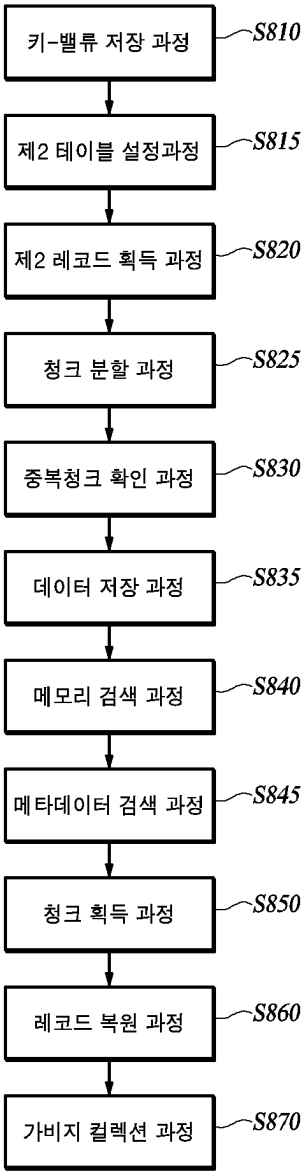
	키	오프셋 리스트
610	키 1	P_{11}^{*}, P_{12}^{*}, …
620	키 2	P_{21}^{*}, P_{22}^{*}, …
630	키 3	P_{31}^{*}, P_{32}^{*}, …
	⋮	⋮

도면7

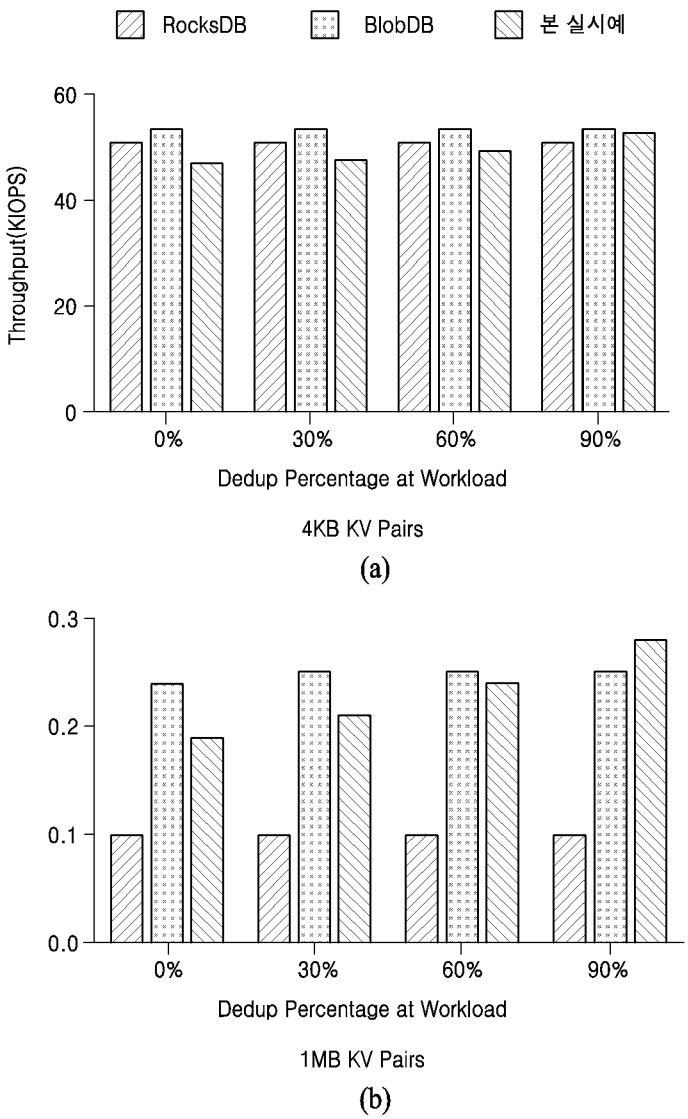
422



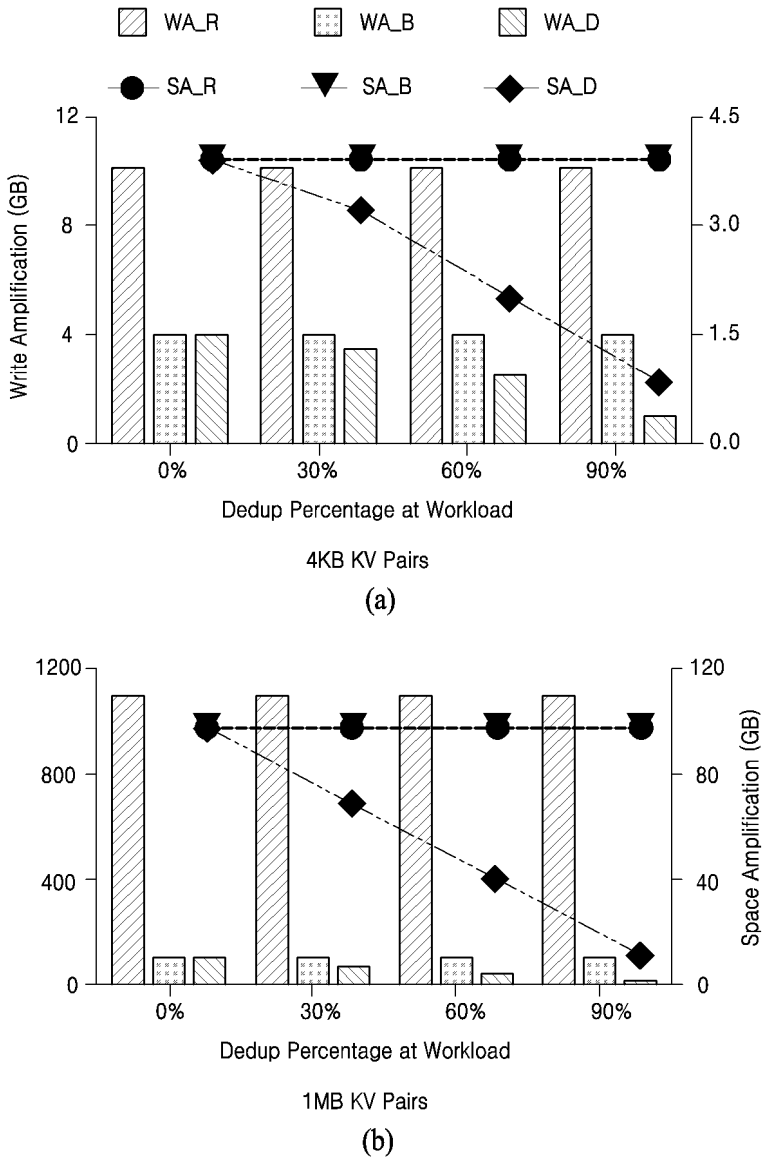
도면8


키-밸류 저장 과정
S810
제2 테이블 설정과정
S815
제2 레코드 획득 과정
S820
청크 분할 과정
S825
중복청크 확인 과정
S830
데이터 저장 과정
S835
메모리 검색 과정
S840
메타데이터 검색 과정
S845
청크 획득 과정
S850
레코드 복원 과정
S860
가비지 컬렉션 과정
S870

도면9


RocksDB
BlobDB
본 실시예
Throughput(KIOPS)
60
40
20
0
0%
30%
60%
90%
Dedup Percentage at Workload
4KB KV Pairs
(a)
0.3
0.2
0.1
0.0
0%
30%
60%
90%
Dedup Percentage at Workload
1MB KV Pairs
(b)

도면10


WA_R
WA_B
WA_D
SA_R
SA_B
SA_D
Write Amplification (GB)
12
8
4
0
4.5
3.0
1.5
0.0
0%
30%
60%
90%
Dedup Percentage at Workload
4KB KV Pairs
(a)
1200
800
400
0
120
80
40
0
Space Amplification (GB)
0%
30%
60%
90%
Dedup Percentage at Workload
1MB KV Pairs
(b)