(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0089031
(43) 공개일자 2025년06월18일

(51) 국제특허분류(Int. Cl.)
G06F 3/06 (2006.01) *G06F 12/02* (2018.01)
(52) CPC특허분류
G06F 3/0643 (2013.01)
G06F 12/0223 (2013.01)
(21) 출원번호 10-2023-0178502
(22) 출원일자 2023년12월11일
심사청구일자 2023년12월11일

(71) 출원인
한양대학교 산학협력단
서울특별시 성동구 왕십리로 222(행당동, 한양대학교내)
(72) 발명자
강수용
서울특별시 강남구 선릉로 221, 206동 1301호 (도곡동, 도곡렉슬아파트)
최기한
경기도 성남시 분당구 정자일로 140, 201동 313호 (정자동, 정자역엠코헤리츠2단지)
(뒷면에 계속)
(74) 대리인
양성보

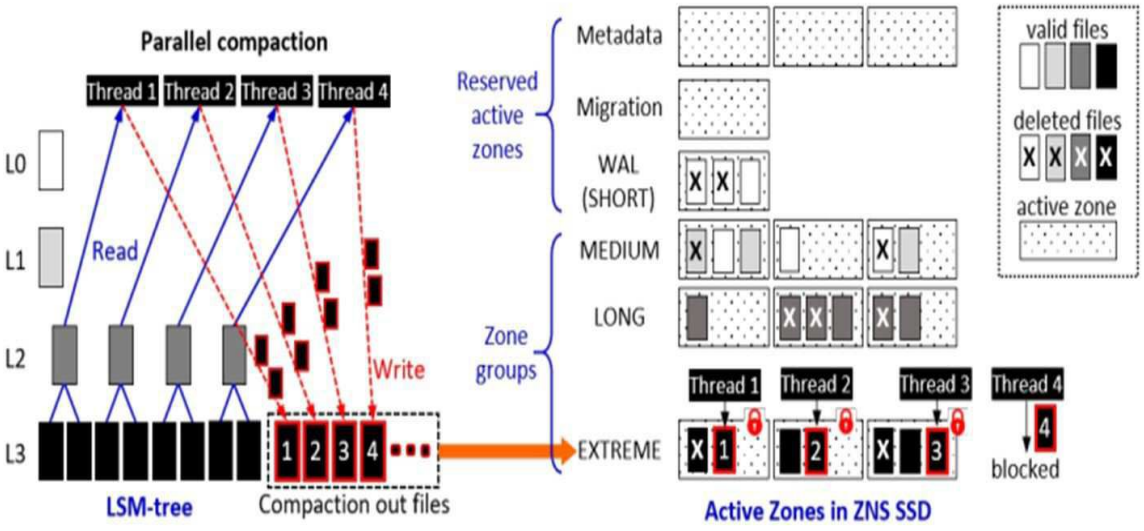
전체 청구항 수 : 총 8 항

(54) 발명의 명칭 **Zoned Namespaces(ZNS) SSD를 위한 파일시스템 데이터 관리 방법 및 시스템**

(57) 요 약

Zoned Namespaces (ZNS) SSD를 위한 파일시스템 데이터 관리 방법 및 시스템이 개시된다. 일 실시예에 따른 컴퓨터 시스템에 의해 수행되는 파일시스템 데이터 관리 방법은, 컴팩션 로컬리티(Compaction Locality)에 기초하여 영역(zone) 그룹의 사이즈를 동적으로 조정하는 동적 영역 재분배(Dynamic Zone Redistribution) 단계; 및 상기 사이즈가 조정된 영역 그룹을 이용하여 ZNS(Zoned Namespaces) SSD를 위한 파일시스템 데이터를 관리하는 데이터 관리 단계를 포함할 수 있다.

대 표 도 - 도5



(52) CPC특허분류
G06F 3/0604 (2013.01)
G06F 2212/214 (2013.01)
(72) 발명자
김도은
서울특별시 성동구 왕십리로 309, 1122호 (행당동, 왕십리역이스타빌)
김진영
경기도 의정부시 녹양로62번길 12, 104동 201호 (녹양동, 녹양힐스테이트)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711194404
과제번호	2021-0-00590
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	한국연구재단 부설 정보통신기획평가원 / 정보통신방송 연구개발사업 / 데이터 경제를 위한 블록체인 기술개발사업
연구과제명	대규모 노드에서 블록단위의 효율적인 거래 확정을 위한 최종성 보장 기술개발
기 여 율	1/1
과제수행기관명	한양대학교산학협력단
연구기간	2023.01.01 ~ 2023.12.31

명 세 서

청구범위

청구항 1

컴퓨터 시스템에 의해 수행되는 파일시스템 데이터 관리 방법에 있어서,

컴팩션 로컬리티(Compaction Locality)에 기초하여 영역(zone) 그룹의 사이즈를 동적으로 조정하는 동적 영역 재분배(Dynamic Zone Redistribution) 단계; 및

상기 사이즈가 조정된 영역 그룹을 이용하여 ZNS(Zoned Namespaces) SSD를 위한 파일시스템 데이터를 관리하는 데이터 관리 단계

를 포함하는 파일시스템 데이터 관리 방법.

청구항 2

제1항에 있어서,

상기 동적 영역 재분배 단계는,

각 영역 그룹을 복수 개의 라이프타임 각각에 연결하고, 상기 각 영역 그룹에 각 영역 그룹과 동일한 라이프타임을 갖는 파일을 수용하는 단계

를 포함하고,

상기 복수 개의 라이프타임은, 제1 라이프타임(MEDIUM), 제2 라이프타임(LONG) 및 제3 라이프타임(EXTREME)을 포함하는 것을 특징으로 하는 파일시스템 데이터 관리 방법.

청구항 3

제1항에 있어서,

상기 동적 영역 재분배 단계는,

각 영역 그룹에 활성 영역의 개수를 균등하게 배분하여 각 영역 그룹의 사이즈를 제한하는 단계

를 포함하는 파일시스템 데이터 관리 방법.

청구항 4

제1항에 있어서,

상기 동적 영역 재분배 단계는,

상기 컴팩션 로컬리티의 위치와 강도에 기초하여 각 영역 그룹을 확장 또는 축소하는 단계

를 포함하는 파일시스템 데이터 관리 방법.

청구항 5

제4항에 있어서,

상기 동적 영역 재분배 단계는,

상기 각 영역 그룹에 영역 잠금 경합의 강도를 나타내는 차단 시간 비율(Blocking Time Ratio)에 기초하여 컴팩션 로컬리티를 추적하는 단계

를 포함하는 파일시스템 데이터 관리 방법.

청구항 6

제5항에 있어서,

상기 동적 영역 재분배 단계는,

상기 각 영역 그룹 내에서 영역을 할당할 때마다 영역 그룹의 차단 시간을 누적하고, 상기 누적된 영역 그룹의 차단 시간을 합산하여 영역 그룹별 차단 시간 비율을 도출하고, 상기 도출된 영역 그룹별 차단 시간 비율에 기초하여 영역 그룹별 누적 시간을 기준으로 각 영역 그룹의 사이즈 제한을 재할당하고, 상기 재할당된 각 영역 그룹의 사이즈 제한에 따라 활성 영역을 재분배하는 단계

를 포함하는 파일시스템 데이터 관리 방법.

청구항 7

제3항에 있어서,

상기 동적 영역 재분배 단계는,

백그라운드 스레드의 개수에 따라 각 영역 그룹에 설정된 최소 활성 영역의 개수를 제공하는 단계

를 포함하는 파일시스템 데이터 관리 방법.

청구항 8

컴퓨터 시스템에 있어서,

컴팩션 로컬리티(Compaction Locality)에 기초하여 영역(zone) 그룹의 사이즈를 동적으로 조정하는 동적 영역 재분배(Dynamic Zone Redistribution)부; 및

상기 사이즈가 조정된 영역 그룹을 이용하여 ZNS(Zoned Namespaces) SSD를 위한 파일시스템 데이터를 관리하는 데이터 관리부

를 포함하는 컴퓨터 시스템.

발명의 설명

기 술 분 야

[0001] 아래의 설명은 파일시스템 데이터 관리 기술에 관한 것이다.

배 경 기 술

[0003] 도 1을 참고하면, 바닐라 Zenfs 및 처리량이 최적화된 Zenfs를 통한 RocksDB의 성능을 설명하기 위한 예이다. 파일의 라이프타임(lifetime)을 고려한 Zenfs와 고려하지 않은 Throughput Optimized Zenfs(TO-Zenfs)를 비교하였을 때, 2개 쓰레드에서는 TO-Zenfs 대비 7배 낮은 쓰기 증폭을 보이지만, 8개 쓰레드에서는 14% 이상의 쓰기 증폭을 보인다. 이러한 결과는 Zenfs의 파일의 라이프타임을 기반으로 한 영역(Zone) 할당 정책이 멀티 쓰레드 환경에서 잠금(lock)을 회피하고자 하는 정책의 부작용으로 인해 파일의 라이프타임을 제대로 고려하지 못하고 있음을 반증한다.

[0004] 도 2를 참고하면, 할당된 영역에서 새로운 파일과 기존 파일 간의 최대 라이프타임 차이의 분포를 나타낸 예이다. 이는, 영역에 저장되어 있는 파일의 라이프타임과 영역의 라이프타임 간의 차이 분포를 통해 자세히 확인할 수 있다. Zenfs의 영역 할당 정책에서 가장 우선시하는 영역과 파일 간의 라이프타임 차이는 Difference 1이지만 2개 쓰레드에서 43% 차지하던 비율이 8개 쓰레드에서는 16%로 감소하며, 가장 선호하지 않는 Difference 2와 Difference 3은 2개 쓰레드에서 28% 차지하던 비율이 8개 쓰레드에서 52%로 증가하는 현상을 확인할 수 있다.

[0005] 이처럼 영역 내부에서 파일의 라이프타임이 혼재되면 파일의 라이프타임을 동일하게 관리함으로써 리퀘스트(Reset) 커맨드를 통해 하나의 영역 내부에서 일괄적으로 파일을 삭제할 수 없게 되고, 흩어진 유효한 파일을 라이프타임에 맞게 영역에 재할당하는 과정이 필요하게 된다. 즉, 호스트(Host) 차원에서의 Garbage

Collection(GC) 작업이 야기되어 디바이스의 쓰기 증폭을 초래한다.

발명의 내용

해결하려는 과제

[0007] Zenfs 파일시스템에서 여러 프로세스가 동시에 file I/O 작업을 요청할 경우 파일의 생명 주기가 서로 혼재되어 발생하는 높은 쓰기 증폭 문제를 해결하기 위한 라이프타임(lifetime) 기반의 동적 영역 재분배(Dynamic Zone Redistribution)를 수행할 수 있다.

과제의 해결 수단

[0009] 컴퓨터 시스템에 의해 수행되는 파일시스템 데이터 관리 방법은, 컴팩션 로컬리티(Compaction Locality)에 기초하여 영역(zone) 그룹의 사이즈를 동적으로 조정하는 동적 영역 재분배(Dynamic Zone Redistribution) 단계; 및 상기 사이즈가 조정된 영역 그룹을 이용하여 ZNS(Zoned Namespaces) SSD를 위한 파일시스템 데이터를 관리하는 데이터 관리 단계를 포함할 수 있다.

[0010] 상기 동적 영역 재분배 단계는, 각 영역 그룹을 복수 개의 라이프타임 각각에 연결하고, 상기 각 영역 그룹에 각 영역 그룹과 동일한 라이프타임을 갖는 파일을 수용하는 단계를 포함하고, 상기 복수 개의 라이프타임은, 제1 라이프타임(MEDIUM), 제2 라이프타임(LONG) 및 제3 라이프타임(EXTREME)을 포함할 수 있다.

[0011] 상기 동적 영역 재분배 단계는, 각 영역 그룹에 활성 영역의 개수를 균등하게 배분하여 각 영역 그룹의 사이즈를 제한하는 단계를 포함할 수 있다.

[0012] 상기 동적 영역 재분배 단계는, 상기 컴팩션 로컬리티의 위치와 강도에 기초하여 각 영역 그룹을 확장 또는 축소하는 단계를 포함할 수 있다.

[0013] 상기 동적 영역 재분배 단계는, 상기 각 영역 그룹에 영역 잠금 경합의 강도를 나타내는 차단 시간 비율(Blocking Time Ratio)에 기초하여 컴팩션 로컬리티를 추적하는 단계를 포함할 수 있다.

[0014] 상기 동적 영역 재분배 단계는, 상기 각 영역 그룹 내에서 영역을 할당할 때마다 영역 그룹의 차단 시간을 누적하고, 상기 누적된 영역 그룹의 차단 시간을 합산하여 영역 그룹별 차단 시간 비율을 도출하고, 상기 도출된 영역 그룹별 차단 시간 비율에 기초하여 영역 그룹별 누적 시간을 기준으로 각 영역 그룹의 사이즈 제한을 재할당하고, 상기 재할당된 각 영역 그룹의 사이즈 제한에 따라 활성 영역을 재분배하는 단계를 포함할 수 있다.

[0015] 상기 동적 영역 재분배 단계는, 백그라운드 스레드의 개수에 따라 각 영역 그룹에 설정된 최소 활성 영역의 개수를 제공하는 단계를 포함할 수 있다.

[0016] 컴퓨터 시스템은, 컴팩션 로컬리티(Compaction Locality)에 기초하여 영역(zone) 그룹의 사이즈를 동적으로 조정하는 동적 영역 재분배(Dynamic Zone Redistribution)부; 및 상기 사이즈가 조정된 영역 그룹을 이용하여 ZNS(Zoned Namespaces) SSD를 위한 파일시스템 데이터를 관리하는 데이터 관리부를 포함할 수 있다.

발명의 효과

[0018] 컴팩션 로컬리티(compaction Locality)를 고려하여 영역 그룹의 사이즈를 조정하여 결정된 활성 영역이 잠금 경합을 최소화 할 수 있도록 한 결과, 기존 Zenfs 성능의 4.9%의 성능을 희생하여 쓰기 증폭을 94.5% 감소시킬 수 있다.

도면의 간단한 설명

[0020] 도 1은 바닐라 Zenfs 및 처리량이 최적화된 Zenfs를 통한 RocksDB의 성능을 설명하기 위한 예이다.

도 2는 할당된 영역에서 새로운 파일과 기존 파일 간의 최대 라이프타임 차이의 분포를 나타낸 예이다.

도 3은 일 실시예에 있어서, 컴퓨터 시스템의 구성을 설명하기 위한 블록도이다.

도 4는 일 실시예에 있어서, 파일시스템 데이터 관리 방법을 설명하기 위한 흐름도이다.

도 5는 일 실시예에 있어서, 영역 그룹 기반 기본 영역을 할당하는 동작을 설명하기 위한 도면이다.

도 6은 일 실시예에 있어서, 영역 할당 요청별 차단 시간을 설명하기 위한 도면이다.

도 7 및 도 9은 일 실시예에 있어서, 파일시스템 데이터 관리 방법의 기술적 효과를 설명하기 위한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0021] 이하, 실시예를 첨부한 도면을 참조하여 상세히 설명한다.

[0023] 도 3은 일 실시예에 있어서, 컴퓨터 시스템의 구성을 설명하기 위한 블록도이고, 도 4는 일 실시예에 있어서, 파일시스템 데이터 관리 방법을 설명하기 위한 흐름도이다.

[0024] 컴퓨터 시스템(100)의 프로세서는 동적 영역 재분배부(310) 및 데이터 관리부(320)를 포함할 수 있다. 이러한 프로세서의 구성요소들은 컴퓨터 시스템에 저장된 프로그램 코드가 제공하는 제어 명령에 따라 프로세서에 의해 수행되는 서로 다른 기능들(different functions)의 표현들일 수 있다. 프로세서 및 프로세서의 구성요소들은 도 4의 파일시스템 데이터 관리 방법이 포함하는 단계들(410 내지 420)을 수행하는 컴퓨터 시스템을 제어할 수 있다. 이때, 프로세서 및 프로세서의 구성요소들은 메모리가 포함하는 운영체제의 코드와 적어도 하나의 프로그램의 코드에 따른 명령(instruction)을 실행하도록 구현될 수 있다.

[0025] 프로세서는 파일시스템 데이터 관리 방법을 위한 프로그램의 파일에 저장된 프로그램 코드를 메모리에 로딩할 수 있다. 예를 들면, 컴퓨터 시스템에서 프로그램이 실행되면, 프로세서는 운영체제의 제어에 따라 프로그램의 파일로부터 프로그램 코드를 메모리에 로딩하도록 컴퓨터 시스템을 제어할 수 있다. 이때 프로세서는 동적 영역 재분배부(310) 및 데이터 관리부(320) 각각에 메모리에 로딩된 프로그램 코드 중 대응하는 부분의 명령을 실행하여 이후 단계들(410 내지 420)을 실행하기 위한 프로세서의 서로 다른 기능적 표현들일 수 있다.

[0026] 단계(410)에서 동적 영역 재분배부(310)는 컴팩션 로컬리티(Compaction Locality)에 기초하여 영역(zone) 그룹의 사이즈를 동적으로 조정할 수 있다. 동적 영역 재분배부(310)는 각 영역 그룹을 복수 개의 라이프타임 각각에 연결하고, 각 영역 그룹에 각 영역 그룹과 동일한 라이프타임을 갖는 파일을 수용할 수 있다. 동적 영역 재분배부(310)는 각 영역 그룹에 활성 영역의 개수를 균등하게 배분하여 각 영역 그룹의 사이즈를 제한할 수 있다. 동적 영역 재분배부(310)는 컴팩션 로컬리티의 위치와 강도에 기초하여 각 영역 그룹을 확장 또는 축소할 수 있다. 동적 영역 재분배부(310)는 각 영역 그룹에 영역 잠금 경합의 강도를 나타내는 차단 시간 비율(Blocking Time Ratio)에 기초하여 컴팩션 로컬리티를 추적할 수 있다. 동적 영역 재분배부(310)는 각 영역 그룹 내에서 영역을 할당할 때마다 영역 그룹의 차단 시간을 누적하고, 누적된 영역 그룹의 차단 시간을 합산하여 영역 그룹별 차단 시간 비율을 도출할 수 있다. 동적 영역 재분배부(310)는 도출된 영역 그룹별 차단 시간 비율에 기초하여 영역 그룹별 누적 시간을 기준으로 각 영역 그룹의 사이즈 제한을 재할당하고, 재할당된 각 영역 그룹의 사이즈 제한에 따라 활성 영역을 재분배할 수 있다. 동적 영역 재분배부(310)는 백그라운드 스레드의 개수에 따라 각 영역 그룹에 설정된 최소 활성 영역의 개수를 제공할 수 있다.

[0027] 단계(420)에서 데이터 관리부(320)는 사이즈가 조정된 영역 그룹을 이용하여 ZNS(Zoned Namespaces) SSD를 위한 파일시스템 데이터를 관리할 수 있다.

[0028] 도 5는 일 실시예에 있어서, 영역 그룹 기반 기본 영역을 할당하는 동작을 설명하기 위한 도면이다.

[0029] 컴퓨터 시스템은 다중 스레드 병렬 압축 환경에서 컴팩션 로컬리티(compaction locality)에 의해 유발되는 강력한 영역(zone) 잠금 경합을 효과적으로 완화하는 새로운 영역(zone) 관리 체계인 동적 영역 재분배(Dynamic Zone Redistribution)를 제공할 수 있다.

[0030] 병렬 압축을 사용하여 압축 성능을 향상시키려면 가능한 한 많은 압축 스레드를 활성화하여 압축 파일을 동시에 쓰기(write)할 수 있도록 하여 영역 잠금 경합을 효과적으로 방지하는 것이 중요하다. 그런 의미에서 활성 영역 제한은 한 번에 쓰기 가능한 영역의 수를 제한하기 때문에 압축 성능에 직접적인 영향을 미친다. 그러나 장치 제조업체에서 결정하는 시스템 상수이므로 라이프타임 기반의 영역 할당 정책을 존중하여 다양한 라이프타임에 걸쳐 활성 영역을 체계적으로 분배하여 압축 성능을 향상시킬 수 있다.

[0031] RocksDB에서 관찰한 컴팩션 로컬리티는 ZenFS의 라이프타임 기반 영역 할당 정책을 망치는 반면, 효율적인 영역 관리 체계를 고안하는 데 매우 유용한 단서를 제공한다. 즉, 컴팩션 로컬리티가 발생한 라이프타임 동안 더 많은 활성 영역을 제공함으로써 영역 잠금 경합을 약화시키고 각 영역에서 전체 라이프타임 이질성을 낮출 수 있다. 그러나 두 가지 과제 때문에 이를 실현하는 것은 간단한 작업이 아니다. 첫째, 컴팩션 로컬리티가 발생하는 이동 레벨을 정확하게 추적하고 로컬리티(locality)의 강도를 측정해야 하며, 둘 다 키-값 크기 분포, 액세스 패턴 및 키-범위 로컬리티를 포함한 작업량 특성에 따라 동적으로 변화한다. 둘째, 추적된 컴팩션 로컬리티 하에서 가장 적절한 활성 영역 분포를 결정해야 하며, 이는 컴팩션 로컬리티의 강도를 통합한 영역 잠금 경합을 수치로 표현해야 한다.

[0032] 이를 위해 컴팩션 로컬리티의 현재 위치를 간접적으로 추적하고 그 강도를 수치적으로 나타낼 수 있는 새로운 척도인 차단 시간 비율(blocking time ratio)을 고안하였다.

[0033] 기본 영역 할당 정책에 대하여 설명하기로 한다. 컴퓨터 시스템은 영역 그룹이라고 하는 '활성' 영역 그룹을 세 개의(MEDIUM, LONG, EXTREME) 라이프타임 각각에 연결할 수 있다. 각 영역 그룹은 자신과 동일한 라이프타임을 갖는 새로운 파일을 수용한다. 이러한 기본 영역 할당 정책은 각 영역의 파일 간의 라이프타임 동질성을 강제하므로 GearDB에서와 같이 쓰기 증폭을 효과적으로 억제한다. 이 정책의 설계 근거는 두 가지 관찰을 기반으로 한다. 첫째, 컴팩션 로컬리티로 인해 로컬리티가 지속되는 동안 동일한 라이프타임을 가진 많은 새로운 파일이 집중적으로 생성된다. 둘째, 다중 스레드 압축 환경에서 병렬 압축은 전체 압축 프로세스를 빠르게 하고 RocksDB의 쓰기 처리량을 증가시킨다. 증가된 처리량은 상위 레벨에서 더 빈번한 압축과 빠른 파일 삭제로 이어진다. 결과적으로 라이프타임이 더 짧은 파일이 라이프타임이 더 긴 영역에 흩어져 있는 것을 방지하여 유효한 파일 복사본 없이 상위 레벨에 대한 영역을 더 빠르게 회수할 수 있다.

[0034] 도 5를 참고하면, 활성 영역 제한(limit)이 14라고 가정했을 때, 영역 그룹 기반 기본 영역 할당을 보여준다. ZenFS는 처음에 물리적 또는 논리적으로 몇 개의 활성 영역을 준비한다. 논리적 영역을 준비한다는 것은 필요할 때 향후 새로운 영역을 열기 위한 '슬롯'을 따로 남겨둔다는 것을 의미한다. 구체적으로, ZenFS 및 RocksDB 메타데이터를 위한 3개의 물리적 영역과 익스텐트 마이그레이션을 위한 것과 WAL 파일을 위한 것으로 2개의 논리적 영역을 구성한다. 나머지 활성 영역 개수는 SST 파일에 사용할 수 있다. 컴퓨터 시스템은 ZenFS에서 이러한 활성 영역 예약 정책을 상속할 수 있다.

[0035] 그러나 컴퓨터 시스템은 ZenFS와 다른 방식으로 WAL 파일을 처리할 수 있다. ZenFS는 WAL 및 SST 파일을 동일한 영역에 저장할 수 있도록 한다. WAL 파일의 경우 SST 파일과 동일한 라이프타임 기반 할당 정책을 사용하여 영역을 할당한다. ZenFS는 WAL 파일에 할당할 비잠금 영역이 없을 때만 '예약된 슬롯'을 사용하여 WAL 영역을 오픈한다. 예약된 슬롯을 사용하면 ZenFS가 활성 영역 제한을 고려하지 않고 즉시 새로운 영역을 오픈할 수 있으므로 WAL 파일의 지연된 쓰기를 방지할 수 있다. 그러나 실제로 WAL 및 SST 파일은 특히 다중 스레드 환경에서 스토리지에서의 파일 생성 및 삭제 측면에서 크게 다른 동작을 보여준다. 한 번에 하나의 WAL 파일만 생성되며(동시성 없음), 해당 memtable이 SST 파일로 성공적으로 플러시되면 WAL 파일을 즉시 폐기할 수 있다(빠른 삭제). 위 속성을 기반으로 컴퓨터 시스템은 물리적으로 WAL 영역을 예약하여 WAL 파일과 SST 파일을 분리할 수 있다. 전자의 속성은 WAL 영역에서 잠금 경합이 발생하지 않도록 하여 WAL 파일의 신속한 쓰기를 가능하게 하고, 후자의 속성은 마지막 WAL 파일을 폐기할 때 유효한 파일 복사본 없이 WAL 영역을 회수할 수 있도록 보장하며, 이 둘 모두는 WAL 영역의 물리적 분리를 정당화한다. 컴퓨터 시스템은 확장 크기 미만의 용량이 적게 남아 있을 때 WAL 영역을 완료한 다음 향후 WAL 파일 쓰기가 지연되지 않도록 즉시 새로운 파일을 오픈한다.

[0036] 마지막으로, 컴퓨터 시스템은 3개의 영역 그룹에 나머지 활성 영역의 수를 균등하게 배분하여 초기 영역 그룹 크기 제한(각 그룹에 허용되는 활성 영역의 최대 수)을 결정한다. 예를 들어, 활성 영역 제한이 14개로 주어지면 5개의 영역을 확보하고 각 그룹의 크기 제한을 3개로 동일하게 설정한다.

[0037] 백그라운드 스레드가 새로운 파일을 쓰기 위해서, 컴퓨터 시스템은 파일 라이프타임과 관련된 특정 영역 그룹에서만 비잠금 영역을 찾아야 하는 반면 ZenFS는 파일 라이프타임보다 라이프타임이 긴 모든 영역에서 찾을 수 있다. 영역 그룹의 모든 활성 영역이 다른 파일 쓰기를 위해 이미 잠겼다면 컴퓨터 시스템은 현재 영역 그룹 크기가 제한보다 작은 경우, 새로운 영역을 오픈하거나, 새로운 영역을 오픈하지 않으면 영역 그룹의 활성 영역 중 하나를 잠금 해제될 때까지 차단(busy wait)할 수 있다. 이러한 영역 그룹 기반 영역 할당 정책은 ZenFS보다 주어진 파일을 수용할 수 있는 활성 영역의 수가 더 제한되어 있기 때문에 파일 쓰기 성능을 저하시킬 수 있다.

[0038] 또한, 컴퓨터 시스템은 영역 그룹의 크기를 조정할 수 있다. 컴퓨터 시스템의 궁극적인 목표는 높은 쓰기 처리

량과 낮은 쓰기 증폭을 동시에 달성하는 것이다. 컴퓨터 시스템은 동일한 라이프타임을 갖는 파일들만 하나의 영역에 저장하기 때문에 병렬 압축 환경에서 ZenFS가 겪는 라이프타임 이질성 문제가 없으므로 ZenFS보다 훨씬 낮은 쓰기 증폭률을 보일 것으로 쉽게 예상할 수 있다. 이에, 컴팩션 로컬리티가 발생하는 영역 그룹 내에서 제한된 수의 활성 영역 중 영역 잠금 경합을 극복하여 높은 쓰기 처리량을 유지해야 한다.

[0039] 또한, 컴퓨터 시스템은 컴팩션 로컬리티를 추적할 수 있다. 컴퓨터 시스템은 다양한 컴팩션 로컬리티의 위치와 강도를 고려하여 각 영역 그룹의 크기 제한을 동적으로 조정하여 문제를 해결한다. 이를 위해 컴퓨터 시스템은 각 영역 그룹에 영역 잠금 경합의 강도를 나타내는 '차단 시간 비율' 개념을 도입한다. 컴팩션 로컬리티로 인해 경합이 심한 영역 그룹에서는 비잠금 영역을 대기하면서 영역 할당 요청이 자주 차단될 수 있다. 도 6을 참고하면, 서로 다른 압축 스레드에 의해 레벨에서 6개의 새로운 파일이 동시에 생성된 경우를 보여준다. 스레드로부터 영역 할당 요청을 받은 컴퓨터 시스템은 파일 라이프타임을 기반으로 대상 영역 그룹을 식별하고 그룹의 모든 영역에 대한 잠금을 획득하려고 시도한다. 영역 할당 요청의 차단 시간은 컴퓨터 시스템이 잠금을 획득할 때까지 잠금을 대기하기 시작하는 경과 시간으로 정의된다. 도 6의 스레드 1과 스레드 2의 경우, 영역 그룹에 비잠금 영역이 존재하기 때문에 컴퓨터 시스템은 즉시 잠금을 획득할 수 있다. 다른 스레드의 경우, 컴퓨터 시스템은 zone 1 또는 zone 2 중 하나가 잠금 해제될 때까지 기다리지 않을 수 없다. 따라서 경합이 심한 영역 그룹의 전체 차단 시간이 커진다.

[0040] 컴퓨터 시스템은 그룹 내에서 영역을 할당할 때마다 영역 그룹의 차단 시간을 누적하고, 누적된 영역 그룹의 차단 시간을 합산하여 최종적으로 영역 그룹별 차단 시간 비율을 획득할 수 있다. 영역 그룹 G_i의 차단 시간 비율 $R(G_i)$은 다음과 같이 정의된다.

[0041] $$R(G_i) = \sigma_{G_i} / \sum_j \sigma_{G_j}$$

[0042] 여기서 σ_{G_i}는 영역 그룹 G_i의 누적 차단 시간을 나타낸다. 차단 시간 비율이 클수록 영역 그룹의 경합이 강화된다. 컴팩션 로컬리티는 발생하는 영역 그룹에서 잠금 경합을 강화하기 때문에 차단 시간 비율은 컴팩션 로컬리티의 위치와 강도를 모두 알려준다. 수치 형태의 정보를 통해 추적된 컴팩션 로컬리티에서 가장 적절한 활성 영역 분포를 쉽게 결정할 수 있다. 기본적으로 컴퓨터 시스템은 차단 시간 비율에 비례하여 영역 그룹의 크기 제한을 재할당한다. 따라서 경합이 심한 영역 그룹은 더 많은 활성 영역을 가질 수 있어 쓰기 처리량이 증가한다.

[0043] 컴퓨터 시스템은 차단 시간 비율을 업데이트하고 최신 누적 차단 시간을 기준으로 각 영역 그룹의 크기 제한을 재할당한다. 컴퓨터 시스템은 로컬리티 변화를 감지할 때마다 모든 영역 그룹의 누적 차단 시간을 재설정하여 변경된 로컬리티에 즉시 적응한다. 다음 부울 표현식이 참일 때 로컬리티 변화를 인식한다.

[0044] $$(P_{LONG} \oplus C_{LONG}) \vee (P_{EXTREME} \oplus C_{EXTREME})$$

[0045] 여기서 P_{LONG} 및 C_{LONG}은 각각 이전 및 현재 영역 할당에서 작성되는 LONG 파일의 존재를 나타내는 부울 플래그이다(EXTREME의 경우에도 동일). $P_{LONG} \oplus C_{LONG} = 1$ 은 L1→L2 컴팩션이 시작되거나 완료되었음을 의미하고 $P_{EXTREME} \oplus C_{EXTREME} = 1$ 은 L2→L3 압축이 시작되었거나 마지막 레벨 컴팩션이 완료되었음을 의미한다.

[0046] 재할당된 영역 그룹 크기 제한에 따른 활성 영역 재분배는 영역 할당 알고리즘(Algorithm 1)에 의해 자연스럽게 수행된다.

Algorithm 1 Zone allocation algorithm in DZR

L_{act}: active zone limit
G_i: zone group i, $i \in$ MEDIUM, LONG, EXTREME
G_T: (target) zone group having the same lifetime as the new file
$S(G_i)$: current size of (# of existing active zones in) G_i
$L(G_i)$: size limit of G_i

```
1: if ∃ non-locked zone ∈ G_T then
2:     return a non-locked zone with the least free space;
3: end if

   /* There is no non-locked zone in G_T. We need to either open a new
   zone or wait for any existing active zone in G_T to be unlocked. */
4: if S(G_T) < L(G_T) then /* ZONE GROUP EXTENSION */
   /* G_T can be extended. So, we will open a new zone in G_T.*/
5:     if Σ_i S(G_i) = L_act then /* ZONE GROUP REDUCTION */
       /* Opening a new zone violates the active zone limit. So, we
   first need to finish an active zone in other zone groups that need
   reduction. */
6:         Z = {zone ∈ G_i | S(G_i) > L(G_i)};
           /* Z is a set of zones in zone groups that need reduction. */
7:         Z_V = a non-locked zone in Z that has the least free space;
8:         issue a finish command for Z_V;
           /* Finishing Z_V may take time. So, a zone in G_T may be
   unlocked before Z_V finishes. Then, we allocate it without opening a
   new zone. */
9:         while Z_V not finished yet do
10:            if any zone in G_T is unlocked then
11:                return the unlocked zone;
12:            end if
13:        end while
           /* Z_V has been finished. We can now open a new zone. */
14:    end if
15:    open a new empty zone in G_T and return it;
16: else
   /* G_T has as many active zones as its size limit. So, we cannot help
   waiting for a non-locked zone in G_T. */
17:    wait for any zone in G_T to be unlocked and return it;
18: end if
```

[0047]

[0048] 영역 할당 과정에서 컴퓨터 시스템은 영역 그룹 내 기존 활성 영역의 개수가 크기 제한(영역 그룹 확장)보다 작다면 영역 그룹 내 모든 활성 영역이 이미 잠길 때마다 영역 그룹 내 새로운 비어 있는 영역을 오픈하려고 한다. 따라서 재할당 후 크기 제한이 증가한 경합이 심한 영역 그룹에서는 새로운 크기 제한까지 활성 영역의 수가 급격히 증가하여 그룹 내 더 많은 동시 파일 쓰기 작업을 수용할 수 있다. 그러나 새로운 영역을 개방하려면 활성 영역 제한 제약 조건을 준수하기 위해 기존 활성 영역을 완료해야 할 수도 있다. 이를 위해 컴퓨터 시스템은 기존 활성 영역의 개수가 새로운 크기 제한보다 큰 모든 영역 그룹에서 활성 영역을 선택하고 완료한다(영역 그룹 축소). 컴퓨터 시스템은 비잠금 영역 중 남은 용량이 가장 적은 영역을 희생 영역으로 선택한다.

[0049] 위의 영역 그룹 확장 및 축소를 통해 컴퓨터 시스템은 컴팩션 로컬리티에 따라 활성 영역을 동적으로 재분배하여 강력한 경합 영역 그룹에서 영역 잠금 경합을 완화하고 성능 저하를 극복한다.

[0050] 컴퓨터 시스템은 병렬도가 높은 압축 환경에서 잠금 경합을 효과적으로 완화하지만 성능을 저하시킬 수 있는 두 가지 주목할 만한 문제를 가지고 있다. 첫째, 잠금 경합이 강하지 않은 적은 수의 압축 스레드로 불필요한 오버헤드를 발생시킬 수 있다. 예를 들어, 두 개의 압축 스레드를 사용하면 각 영역 그룹에 두 개의 활성 영역을 정적으로 할당하여 잠금 경합에서 자유로울 수 있다. 그러나 컴퓨터 시스템에서 차단 시간 비율 기반 활성 영역 재분배는 각 영역 그룹에 대해 두 개의 활성 영역을 보장하지 못해 불필요하게 잠금 경합을 발생시킬 수 있다. 둘째, 영역 그룹이 컴팩션 로컬리티를 벗어났기 때문에 누적 차단 시간을 재설정한 후 영역 할당 요청이 발행되지 않은 영역 그룹이 있을 수 있다. 그러면 차단 시간 비율이 0이 되고 따라서 컴퓨터 시스템이 크기 제한을 0으로 재할당한다. 그 경우 그룹의 모든 활성 영역이 활성 영역 재분배를 위해 강제로 종료될 수 있으며, 남은 여유 공간이 많은 새로운 영역이 바람직하지 않은 조기 종료를 수반할 수 있다.

[0051] 컴퓨터 시스템은 백그라운드 스레드 수에 따라 보장된 수를 변경하면서 각 영역 그룹에 보장된 활성 영역 개수

를 제공하여 문제를 해결한다. 특히 각 그룹에 최소($\lfloor N_z/N_{th} \rfloor, \lfloor N_z/N_{zg} \rfloor$)개의 활성 영역을 보장하며, 여기서 N_z, N_{th}, N_{zg}는 각각 예약된 활성 영역을 제외한 전체 활성 영역의 수(즉, 활성 영역 제한 - 예약된 영역의 개수), 배경 스레드의 개수 및 영역 그룹의 개수(= 3)를 나타낸다. N_z = 9인 도 5의 경우, 컴퓨터 시스템은 배경 스레드가 각각 2개, 4개, 8개일 때 각 그룹에 대해 3개, 2개, 1개의 활성 영역을 보장한다. 나머지 활성 영역만 재분배에 사용된다. 따라서 컴퓨터 시스템은 2개의 스레드와 잠금 경합에서 자유롭다. 4개 및 8개의 스레드를 사용할 때 경합이 심한 영역 그룹에 최대 5개 및 7개의 활성 영역을 할당할 수 있으므로 영역 그룹에 동시 파일 쓰기를 수용하기에 충분하다. 이러한 보장된 활성 영역 할당은 또한 신생 영역의 바람직하지 않은 조기 완료되는 것을 억제할 수 있게 한다.

[0052] 도 7 및 도 9은 일 실시예에 있어서, 파일시스템 데이터 관리 방법의 기술적 효과를 설명하기 위한 도면이다.

[0053] 도 7을 참고하면, 각각 20바이트의 키와 800바이트(400바이트로 압축)의 값으로 구성된 3억개의 KV 쌍의 벤치마크이다. 그런 다음 덮어쓰기 벤치마크를 사용하여 무작위로 덮어쓸 수 있다. 덮어쓰기 벤치마크를 실행하는 동안 모든 성능 지표를 측정할 수 있다. GearDB는 가비지 컬렉션을 제거하지만 재귀적 컴팩션 알고리즘으로 인해 상당한 컴팩션 오버헤드가 발생하여 다른 것보다 훨씬 낮은 쓰기 처리량을 달성한다. GearDB를 제외하고 각 체계의 전체 처리량은 스레드 수에 비례한다. CAZA는 주로 가비지 수집 및 영역 할당의 직렬 실행으로 인해 ZenFS 및 DZR(본 발명)보다 낮은 처리량을 나타낸다. 파일에 영역을 할당할 때 CAZA는 잠기지 않은 모든 영역을 참조하여 파일과 가장 긴 중첩 키 범위를 갖는 영역을 찾는다. 이 시간 동안 모든 영역에서 가비지 수집이 실행되는 것을 방지하여 가비지 수집 및 영역 할당을 직렬화한다.

[0054] 가장 중요한 점은 본 발명이 성능 저하(즉, ZenFS보다 주어진 파일을 수용할 수 있는 활성 영역 수가 더 제한됨)에도 불구하고 ZenFS와 비슷한 처리량을 보여준다는 것이다. 스레드가 8개 미만인 ZenFS보다 성능이 약간 뛰어나며 스레드가 8개인 ZenFS보다 약간(4.9%) 낮은 처리량을 달성한다(ZenFS: 180.2 Kops/s, DZR: 171.3 Kops/s). 본 발명은 압축 지역성에 의해 유발된 강력한 영역 잠금 경합을 완화하기 위해 다양한 영역 그룹에 걸쳐 활성 영역을 적시에 정확하게 재분배함으로써 페널티를 극복한다.

[0055] 도 8을 참고하면, 영역 그룹 변화 없음, 백그라운드 스레드 수 8, 사용 가능한(예약되지 않은) 활성 영역의 총 수 9이다. 실험 중 영역 그룹의 크기 변화를 나타낸다. 컴팩션 로컬리티가 변경됨에 따라(맨 위 도면) 컴퓨터 시스템은 각 영역 그룹의 크기 제한을 업데이트하고(중앙 도면) 크기 제한에 따라 활성 영역을 동적으로 재분배한다(맨 아래 도면). 중앙 도면에서 알 수 있듯이, 컴퓨터 시스템은 변화하는 로컬리티를 정확하게 추적하고 로컬리티의 강도에 비례하여 영역 그룹 크기 제한을 업데이트하는 동시에 각 영역 그룹에 최소 1개의 크기 제한을 보장한다. 그러나 중간 및 최저 수치를 비교하면 크기 제한을 업데이트한 후 활성 영역의 재분배 지연을 관찰할 수 있다. 이러한 지연은 영역 완료 작업의 지연으로 인해 발생한다. 실험에 사용한 ZNS SSD는 부분적으로 작성된 10개 영역을 완료할 때 상당한 대기 시간이 발생하며, 대기 시간을 해당 영역에서 사용되지 않은 공간의 부분에 비례한다. 이러한 장치의 특성은 사용되지 않은 공간을 가비지 데이터로 채워 영역을 가득 채우는 특유의 마감 방식에서 비롯된다. 마감이 지연되면 확장할 영역 그룹에서 새로운 영역을 오픈하는 것이 필연적으로 방해되고, 이는 활성 영역의 재분배를 방해한다. 초기 영역 완료(early zone finishing)가 자주 발생하는 8 스레드의 경우 컴퓨터 시스템의 처리량이 낮은 것은 이러한 현상에 기인한다. 그러나 작업의 구현 방식에 따라 종료 작업의 지연 시간이 달라진다. 예를 들면, 장치가 가비지 데이터를 쓰지 않고 영역 메타데이터(예를 들면, 쓰기 포인트)를 직접 업데이트하여 영역을 완료하면 대기 시간이 크게 줄어들 수 있으며, 컴퓨터 시스템은 장치에서 더 나은 성능을 획득할 것으로 예상된다.

[0056] 도 9를 참고하면, 쓰기 증폭을 비교한 예이다. 쓰기 증폭은 가비지 수집 중에 복사된 유효한 데이터의 양으로 표시된다. 대규모 쓰기 증폭은 ZNS SSD의 라이프타임과 성능을 저하시킨다. 도 9는 스레드 수에 따라 복사되는 데이터의 양을 보여준다. 영역을 할당하는 동안 라이프타임을 고려하면 쓰기 증폭이 향상된다. 그러나 도면에서 볼 수 있듯이 ZenFS의 능동적 경합 회피 정책은 압축 지역성이 중요한 병렬 압축 환경에서 라이프타임 기반 영역 할당 체계를 손상시킨다. CAZA는 키 범위가 겹치는 파일을 동일한 영역에 통합하여 쓰기 증폭을 효과적으로 줄인다. 그러나 컴팩션 로컬리티로 인해 활성 영역이 제한되어 이러한 파일을 통합하지 못하는 경우가 많다. 특히 새로운 파일에 대한 영역을 찾을 때 파일과 키 범위가 겹치는 대부분의 영역은 컴팩션 로컬리티로 인해 이미 잠겨 있을 수 있습니다. 이 경우, 키 범위가 겹치지 않는 잠기지 않은 영역을 할당하지 않을 수 없다. 본 발명은 각 영역에서 파일 라이프타임 동질성을 유지하여 다른 제품에 비해 쓰기 증폭이 크게 감소한 것을 보여준다. 8개의 백그라운드 스레드를 사용하여 ZenFS(33.1GB)에 비해 쓰기 증폭이 약 94.5% 감소한 것을

확인할 수 있다.

[0057] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0058] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치에 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

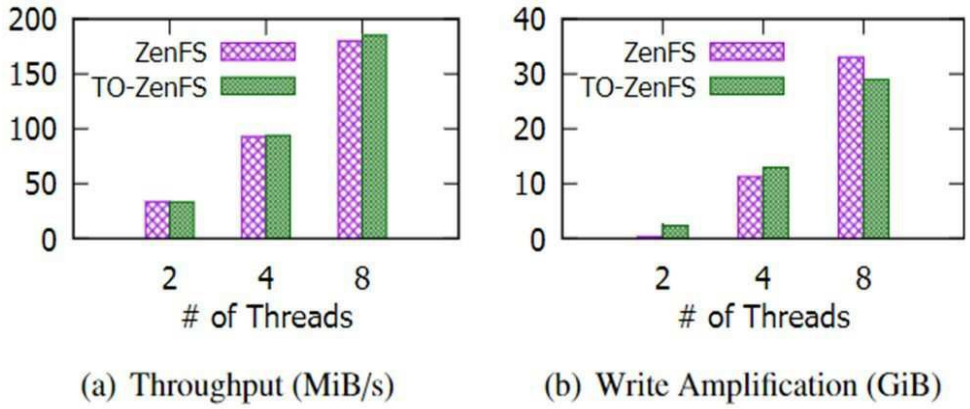
[0059] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다.

[0060] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

[0061] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

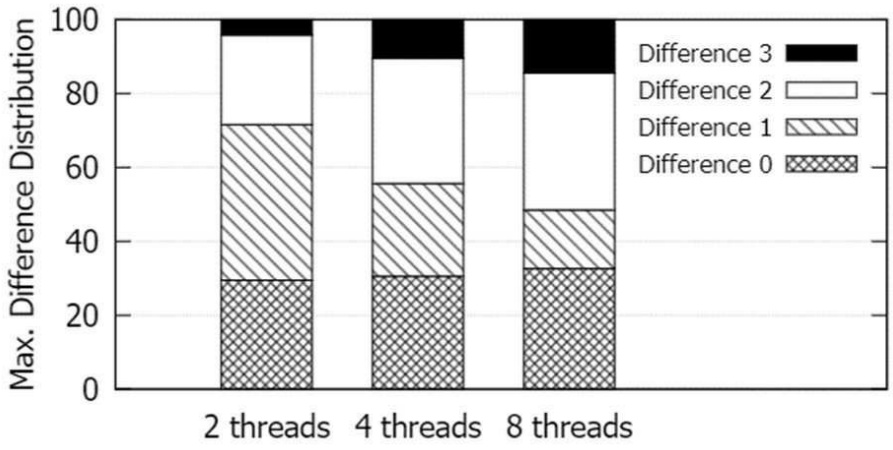
도면

도면1

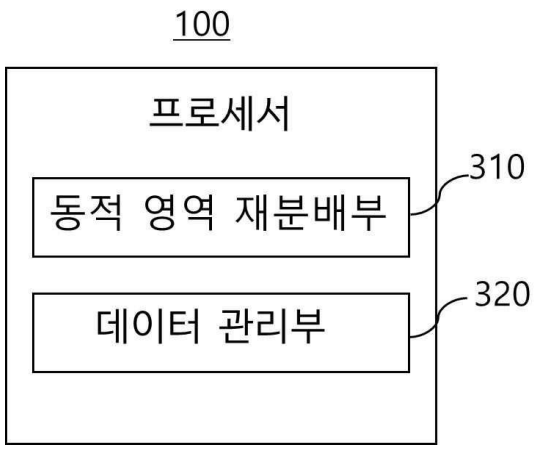


(a) Throughput (MiB/s) (b) Write Amplification (GiB)

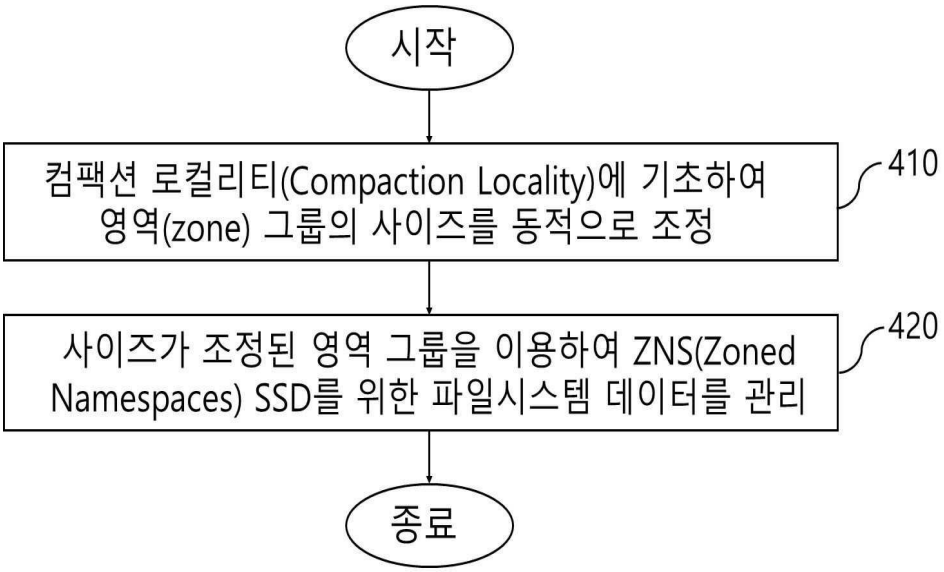
도면2



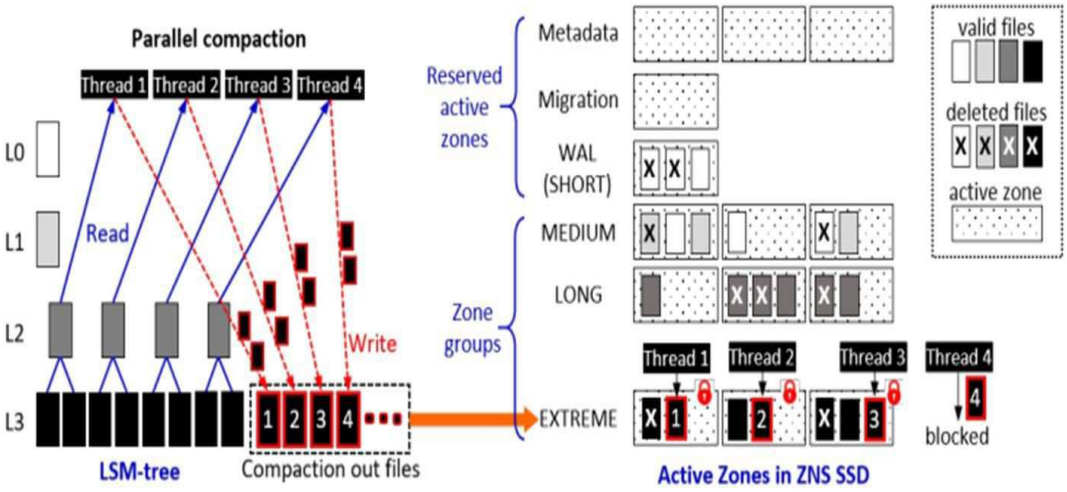
도면3



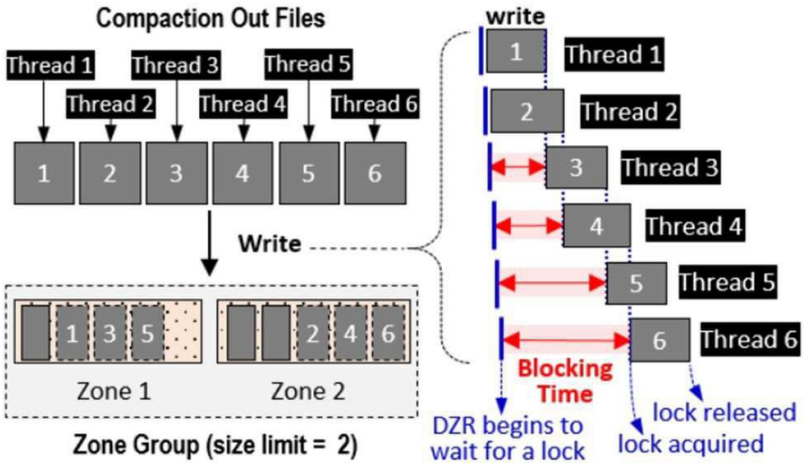
도면4


시작
컴팩션 로컬리티(Compaction Locality)에 기초하여 영역(zone) 그룹의 사이즈를 동적으로 조정
410
사이즈가 조정된 영역 그룹을 이용하여 ZNS(Zoned Namespaces) SSD를 위한 파일시스템 데이터를 관리
420
종료

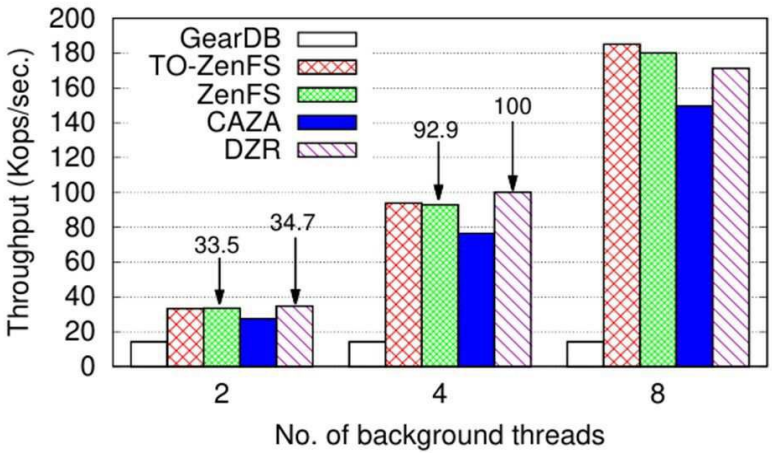
도면5


Parallel compaction
Thread 1
Thread 2
Thread 3
Thread 4
L0
L1
L2
L3
Read
Write
1 2 3 4
LSM-tree
Compaction out files
Reserved active zones
Metadata
Migration
WAL (SHORT)
Zone groups
MEDIUM
LONG
EXTREME
valid files
deleted files
active zone
blocked
Active Zones in ZNS SSD

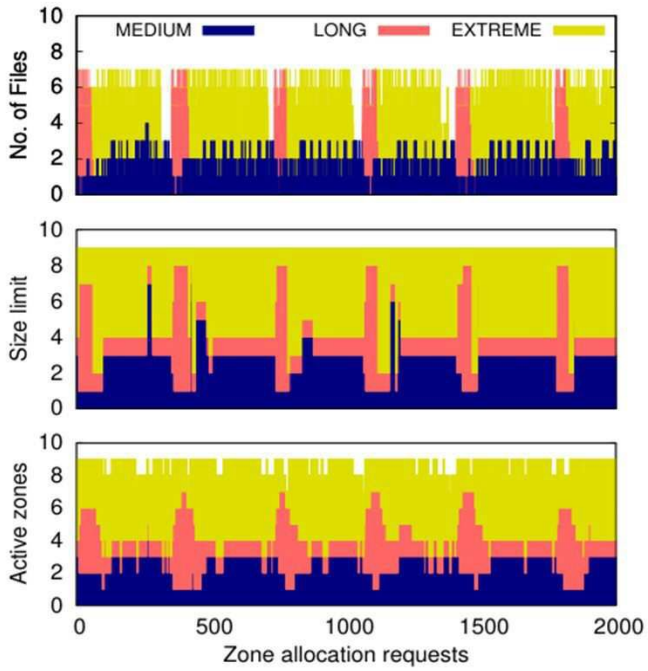
도면6


Compaction Out Files
Thread 1
Thread 3
Thread 5
Thread 2
Thread 4
Thread 6
1
2
3
4
5
6
Write
1
3
5
2
4
6
Zone 1
Zone 2
Zone Group (size limit = 2)
write
Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Thread 6
Blocking Time
DZR begins to wait for a lock
lock released
lock acquired

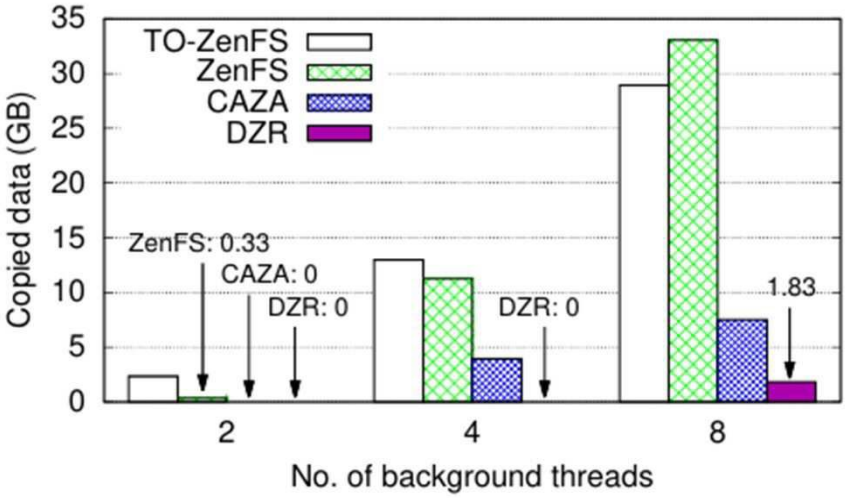
도면7


GearDB
TO-ZenFS
ZenFS
CAZA
DZR
Throughput (Kops/sec.)
200
180
160
140
120
100
80
60
40
20
0
33.5
34.7
92.9
100
2
4
8
No. of background threads

도면8


MEDIUM
LONG
EXTREME
No. of Files
Size limit
Active zones
10
8
6
4
2
0
0
500
1000
1500
2000
Zone allocation requests

도면9


TO-ZenFS
ZenFS
CAZA
DZR
Copied data (GB)
35
30
25
20
15
10
5
0
ZenFS: 0.33
CAZA: 0
DZR: 0
DZR: 0
1.83
2
4
8
No. of background threads