

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0060970

(43) 공개일자 2020년06월02일

(51) 국제특허분류(Int. Cl.)

G06F 16/00 (2019.01)

(52) CPC특허분류

G06F 16/21 (2019.01)

G06F 16/27 (2019.01)

(21) 출원번호 10-2018-0146224

(22) 출원일자 2018년11월23일

심사청구일자 2018년11월23일

(71) 출원인

연세대학교 산학협력단

서울특별시 서대문구 연세로 50 (신촌동, 연세대학교)

(72) 발명자

박상현

서울특별시 서대문구 연세로 50(신촌동)

진민화

서울특별시 서대문구 연세로 50(신촌동)

(뒷면에 계속)

(74) 대리인

특허법인우인

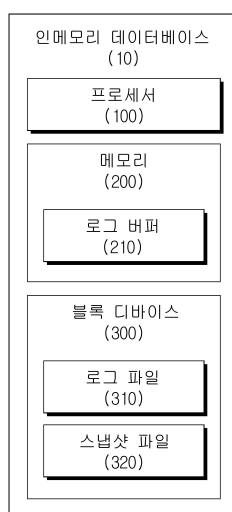
전체 청구항 수 : 총 13 항

(54) 발명의 명칭 스냅샷을 이용한 인메모리 데이터베이스의 데이터 처리 방법 및 인메모리 데이터베이스

(57) 요약

본 실시예들은 블록 디바이스에 텍스트 파일 형식의 임시 로그 파일을 생성하고, 임시 로그 파일을 생성하면 메모리의 로그 버퍼로부터 플러싱하는 저장 위치를 임시 로그 파일을 가리키도록 설정하고, 임시 로그 파일을 가리키도록 설정한 후에 블록 디바이스에 바이너리 파일 형식의 임시 스냅샷 파일을 생성하고, 로그 기록을 작은 단위로 주기적으로 플러싱함으로써, 메모리 사용량을 감소시키고 메모리 사용의 변동성을 최소화하고 데이터베이스의 처리 속도를 향상시키는 인메모리 데이터베이스를 제공한다.

대표도 - 도1



(52) CPC특허분류

G06F 16/90 (2019.01)

(72) 발명자

성한승

서울특별시 서대문구 연세로 50(신촌동)

최원기

서울특별시 서대문구 연세로 50(신촌동)

이 발명을 지원한 국가연구개발사업

과제고유번호 1711065240

부처명 과학기술정보통신부

연구관리전문기관 정보통신기술진흥센터

연구사업명 정부-과학기술정보통신부-정보통신기술진흥센터(NIPA산하)-정보통신방송연구개발사업-SW컴퓨팅산업원천기술개발사업

연구과제명 (SW스타랩)IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발

기 여 율 1/1

주관기관 연세대학교 산학협력단

연구기간 2017.04.01 ~ 2024.12.31

명세서

청구범위

청구항 1

인메모리 데이터베이스의 데이터 처리 방법에 있어서,

블록 디바이스에 텍스트 파일 형식의 임시 로그 파일을 생성하는 단계;

상기 임시 로그 파일을 생성하면, 메모리의 로그 버퍼로부터 플러싱하는 저장 위치를 기억하는 로그 파일 서술자가 상기 임시 로그 파일을 가리키도록 설정하는 단계; 및

상기 임시 로그 파일을 가리키도록 설정하면, 상기 블록 디바이스에 바이너리 파일 형식의 임시 스냅샷 파일을 생성하는 단계

를 포함하는 인메모리 데이터베이스의 데이터 처리 방법.

청구항 2

제1항에 있어서,

상기 메모리의 로그 버퍼에 명령어가 기록된 로그 기록이 기 설정된 크기 또는 비율 이상으로 증가하면, 상기 블록 디바이스에 상기 임시 로그 파일을 생성하며,

상기 로그 기록은 키-값 기반의 데이터인 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

청구항 3

제1항에 있어서,

상기 메모리의 로그 버퍼로부터 상기 블록 디바이스로 기 설정된 단위 시간마다 주기적으로 플러싱하는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

청구항 4

제1항에 있어서,

상기 인메모리 데이터베이스의 프로세서는 부모 프로세스를 동작시켜 상기 메모리에 자식 프로세스를 생성하고, 상기 자식 프로세스는 현재 시점에서의 명령어 집합을 저장하도록 상기 임시 스냅샷 파일로 생성하는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

청구항 5

제4항에 있어서,

상기 인메모리 데이터베이스의 프로세서는 상기 자식 프로세스가 상기 임시 스냅샷 파일을 생성하는 동안에, 새로 입력된 명령어를 부모 프로세스를 동작시켜 상기 로그 버퍼에 저장하는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

청구항 6

제5항에 있어서,

상기 인메모리 데이터베이스의 프로세서는 상기 임시 스냅샷 파일의 생성이 완료되면, 상기 블록 디바이스에 저장된 정식 로그 파일을 삭제하고, 상기 임시 로그 파일의 이름을 상기 정식 로그 파일로 변경하는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

청구항 7

제6항에 있어서,

상기 인메모리 데이터베이스의 프로세서는 상기 임시 스냅샷 파일의 생성이 완료되면, 상기 임시 스냅샷 파일의 이름을 정식 스냅샷 파일로 변경하는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

청구항 8

제7항에 있어서,

상기 메모리는 하나의 로그 버퍼를 사용하고,

상기 블록 드라이브는 상기 정식 로그 파일과 상기 정식 스냅샷 파일을 별도로 저장하여 관리하는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

청구항 9

프로세서, 메모리, 및 블록 드라이브를 포함하며,

상기 블록 디바이스는 텍스트 파일 형식의 임시 로그 파일을 생성하며,

상기 임시 로그 파일을 생성하면, 상기 프로세서는 상기 메모리의 로그 버퍼로부터 플러싱하는 저장 위치를 기억하는 로그 파일 서술자가 상기 임시 로그 파일을 가리키도록 설정하며,

상기 임시 로그 파일을 가리키도록 설정하면, 상기 블록 디바이스는 바이너리 파일 형식의 임시 스냅샷 파일을 생성하는 것을 특징으로 하는 인메모리 데이터베이스.

청구항 10

제9항에 있어서,

상기 메모리의 로그 버퍼로부터 상기 블록 디바이스로 기 설정된 단위 시간마다 주기적으로 플러싱하는 것을 특징으로 하는 인메모리 데이터베이스의

것을 특징으로 하는 인메모리 데이터베이스.

청구항 11

제9항에 있어서,

상기 프로세서는 부모 프로세스를 동작시켜 상기 메모리에 자식 프로세스를 생성하고, 상기 자식 프로세스는 현재 시점에서의 명령어 집합을 저장하도록 상기 임시 스냅샷 파일을 생성하는 것을 특징으로 하는 인메모리 데이터베이스.

청구항 12

제9항에 있어서,

상기 메모리는 하나의 로그 버퍼를 사용하고,

상기 블록 드라이브는 상기 임시 로그 파일을 정식 로그 파일로 변경하고, 상기 임시 스냅샷 파일을 정식 로그 파일로 변경하고,

상기 정식 로그 파일과 상기 정식 스냅샷 파일을 별도로 저장하여 관리하는 것을 특징으로 하는 인메모리 데이터베이스.

청구항 13

프로세서에 의해 실행 가능한 컴퓨터 프로그램 명령어들을 포함하는 비일시적(Non-Transitory) 컴퓨터 판독 가능한 매체에 기록되어 데이터 처리를 위한 컴퓨터 프로그램으로서, 상기 컴퓨터 프로그램 명령어들이 인메모리 데이터베이스의 적어도 하나의 프로세서에 의해 실행되는 경우에,

블록 디바이스에 텍스트 파일 형식의 임시 로그 파일을 생성하는 단계;

상기 임시 로그 파일을 생성하면, 메모리의 로그 버퍼로부터 플러싱하는 저장 위치를 기억하는 로그 파일 서술자가 상기 임시 로그 파일을 가리키도록 설정하는 단계; 및

상기 임시 로그 파일을 가리키도록 설정하면, 상기 블록 디바이스에 바이너리 파일 형식의 임시 스냅샷 파일을 생성하는 단계

를 포함한 동작들을 수행하는 컴퓨터 프로그램.

발명의 설명

기술 분야

[0001] 본 발명이 속하는 기술 분야는 인메모리 데이터베이스의 데이터 처리 방법 및 인메모리 데이터베이스에 관한 것이다.

배경 기술

[0002] 이 부분에 기술된 내용은 단순히 본 실시예에 대한 배경 정보를 제공할 뿐 종래기술을 구성하는 것은 아니다.

[0003] 인메모리 데이터베이스는 디스크가 아닌 메모리에 모든 데이터를 보유하고 있는 데이터베이스이다. 저장매체가 휘발성이기 때문에, 데이터베이스 서버의 전원이 꺼지면 매체에 있는 자료들이 삭제되는 문제가 있다.

[0004] 인메모리 데이터베이스는 지속성을 보장하기 위해서 메모리에 삽입/갱신/삭제된 값들을 디스크에 로그로 기록하며, 인메모리 데이터베이스가 재구동될 때 디스크로부터 로그 파일을 읽고 메모리에 데이터 구조를 모두 재구축한다. 로그 기록 방식은 명령 기록을 로그 형태로 작성하여 메모리 공간의 로그 버퍼에 저장하고, 로그 버퍼로부터 명령 기록을 파일에 플러싱한다. 플러싱은 메모리의 데이터를 다른 저장매체에 다시 써서 일치화하는 동작이다.

[0005] 로그 파일을 기록하는 과정에서 디스크 I/O를 발생시키며, 지속적으로 데이터가 입력되면 로그 파일이 선형적으로 증가하는 문제가 있다. 일정 크기 이상으로 로그 파일이 증가하면 파일 크기를 재구축하지만 파일 크기를 재구축하는 과정은 데이터베이스의 성능에 영향을 주고 과도한 메모리를 사용하는 문제가 있다.

선행기술문헌

특허문헌

[0006] (특허문헌 0001) 한국공개특허공보 제10-2017-0045928호 (2017.04.28.)

발명의 내용

해결하려는 과제

[0007] 본 발명의 실시예들은 블록 디바이스에 텍스트 파일 형식의 임시 로그 파일을 생성하고, 임시 로그 파일을 생성하면 메모리의 로그 버퍼로부터 플러싱하는 저장 위치를 임시 로그 파일을 가리키도록 설정하고, 임시 로그 파일을 가리키도록 설정한 후에 블록 디바이스에 바이너리 파일 형식의 임시 스냅샷 파일을 생성함으로써, 로그 기록을 플러싱하는 동안 소요되는 디스크 I/O 부담을 최소화하는 데 발명의 주된 목적이 있다.

[0008] 본 발명의 명시되지 않은 또 다른 목적들은 하기의 상세한 설명 및 그 효과로부터 용이하게 추론할 수 있는 범위 내에서 추가적으로 고려될 수 있다.

과제의 해결 수단

[0009] 본 실시예의 일 측면에 의하면, 인메모리 데이터베이스의 데이터 처리 방법에 있어서, 블록 디바이스에 텍스트 파일 형식의 임시 로그 파일을 생성하는 단계, 상기 임시 로그 파일을 생성하면 메모리의 로그 버퍼로부터 플러싱하는 저장 위치를 기억하는 로그 파일 저술자가 상기 임시 로그 파일을 가리키도록 설정하는 단계, 및 상기 임시 로그 파일을 가리키도록 설정하면 상기 블록 디바이스에 바이너리 파일 형식의 임시 스냅샷 파일을 생성하는 단계를 포함하는 인메모리 데이터베이스의 데이터 처리 방법을 제공한다.

[0010] 본 실시예의 다른 측면에 의하면, 프로세서, 메모리, 및 블록 드라이브를 포함하며, 상기 블록 디바이스는 텍스트 파일 형식의 임시 로그 파일을 생성하며, 상기 임시 로그 파일을 생성하면 상기 프로세서는 상기 메모리의

로그 버퍼로부터 플러싱하는 저장 위치를 기억하는 로그 파일 서술자가 상기 임시 로그 파일을 가리키도록 설정하며, 상기 임시 로그 파일을 가리키도록 설정하면 상기 블록 디바이스는 바이너리 파일 형식의 임시 스냅샷 파일을 생성하는 것을 특징으로 하는 인메모리 데이터베이스를 제공한다.

[0011] 본 실시예의 또 다른 측면에 의하면, 프로세서에 의해 실행 가능한 컴퓨터 프로그램 명령어들을 포함하는 비일시적(Non-Transitory) 컴퓨터 판독 가능한 매체에 기록되어 데이터 처리를 위한 컴퓨터 프로그램으로서, 상기 컴퓨터 프로그램 명령어들이 인메모리 데이터베이스의 적어도 하나의 프로세서에 의해 실행되는 경우에, 블록 디바이스에 텍스트 파일 형식의 임시 로그 파일을 생성하는 단계, 상기 임시 로그 파일을 생성하면 메모리의 로그 버퍼로부터 플러싱하는 저장 위치를 기억하는 로그 파일 서술자가 상기 임시 로그 파일을 가리키도록 설정하는 단계, 및 상기 임시 로그 파일을 가리키도록 설정하면 상기 블록 디바이스에 바이너리 파일 형식의 임시 스냅샷 파일을 생성하는 단계를 포함한 동작들을 수행하는 컴퓨터 프로그램을 제공한다.

발명의 효과

[0012] 이상에서 설명한 바와 같이 본 발명의 실시예들에 의하면, 블록 디바이스에 텍스트 파일 형식의 임시 로그 파일을 생성하고, 임시 로그 파일을 생성하면 메모리의 로그 버퍼로부터 플러싱하는 저장 위치를 임시 로그 파일을 가리키도록 설정하고, 임시 로그 파일을 가리키도록 설정한 후에 블록 디바이스에 바이너리 파일 형식의 임시 스냅샷 파일을 생성하고, 로그 기록을 작은 단위로 주기적으로 플러싱함으로써, 메모리 사용량을 감소시키고 메모리 사용의 변동성을 최소화하고 데이터베이스의 처리 속도를 향상시키는 효과가 있다.

[0013] 여기에서 명시적으로 언급되지 않은 효과라 하더라도, 본 발명의 기술적 특징에 의해 기대되는 이하의 명세서에서 기재된 효과 및 그 잠정적인 효과는 본 발명의 명세서에 기재된 것과 같이 취급된다.

도면의 간단한 설명

[0014] 도 1은 본 발명의 일 실시예에 따른 인메모리 데이터베이스를 예시한 블록도이다.
 도 2 및 도 3은 본 발명의 일 실시예에 따른 인메모리 데이터베이스가 데이터를 백업하는 동작을 예시한 도면이다.
 도 4는 본 발명의 다른 실시예에 따른 인메모리 데이터베이스의 데이터 처리 방법을 예시한 흐름도이다.
 도 5 및 도 6은 본 발명의 실시예들에 따라 수행된 모의실험 결과를 도시한 것이다.

발명을 실시하기 위한 구체적인 내용

[0015] 이하, 본 발명을 설명함에 있어서 관련된 공지기능에 대하여 이 분야의 기술자에게 자명한 사항으로서 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략하고, 본 발명의 일부 실시예들을 예시적인 도면을 통해 상세하게 설명한다.

[0016] 기존의 인메모리 데이터베이스는 지속성을 확보하기 위해서 임시 로그 파일을 생성하는 방식이 있다. 임시 로그 파일이 생성되는 동안에 입력되는 명령어를 다시 쓰기 버퍼(Rewrite Buffer)에 저장한다. 다시 쓰기 버퍼에 누적된 명령어 기록을 임시 로그 파일로 방출하고, 로그 파일 서술자가 임시 로그 파일을 가리키도록 설정한다.

[0017] 임시 로그 파일 대신에 임시 스냅샷 파일을 생성하는 방식이 있으나, 이러한 방식 역시 임시 스냅샷 파일이 생성되는 동안에 입력되는 명령어를 다시 쓰기 버퍼(Rewrite Buffer)에 저장한다. 다시 쓰기 버퍼에 누적된 명령어 기록을 임시 스냅샷 파일로 방출하고, 임시 스냅샷 파일을 로그 파일로 변경하고 파일 서술자가 로그 파일을 가리키도록 설정한다.

[0018] 인메모리 시스템에서 메모리를 확보하는 것은 중요한데, 이러한 방식들은 다시 쓰기 버퍼를 사용함에 따라 메모리 사용량이 증가하고, 다시 쓰기 버퍼에 누적된 명령어 기록을 방출하는 과정에서 디스크 I/O를 급격히 증가시키는 문제가 있다.

[0019] 본 실시예에 따른 인메모리 데이터베이스는 이러한 문제를 해결하기 위해서 다시 쓰기 버퍼를 사용하지 않고, 로그 파일과 스냅샷 파일을 각각 저장하고 관리하며, 임시 스냅샷 파일을 생성하기 전에 임시 로그 파일을 생성하고 파일 서술자가 임시 로그 파일을 가리키도록 설정한다.

[0020] 도 1은 본 실시예에 따른 인메모리 데이터베이스를 예시한 블록도이다. 도 1에 도시한 바와 같이, 인메모리 데이터베이스(10)는 프로세서(100), 메모리(200), 및 블록 디바이스(300)를 포함한다. 인메모리 데이터베이스(10)

0)는 도 1에서 예시적으로 도시한 다양한 구성요소들 중에서 일부 구성요소를 생략하거나 다른 구성요소를 추가로 포함할 수 있다. 본 실시예에 따른 인메모리 데이터베이스(10)는 다시 쓰기 버퍼를 포함하지 않고 일반 로그 버퍼를 사용하여 데이터를 백업한다.

- [0021] 인메모리 데이터베이스(10)는 디스크가 아닌 메모리에 모든 데이터를 보유하고 있는 데이터베이스이다. 프로세서(100)는 메모리(200) 및 블록 디바이스(300)에 기 정의된 명령어를 전송하여, 각종 신호 및 데이터 흐름을 제어한다. 메모리(200)는 휘발성 메모리로 구현될 수 있다. 예컨대, 휘발성 메모리로는 DRAM(Dynamic Random Access Memory) 등이 있다. 블록 디바이스(300)는 블록 단위로 임의 접근이 가능한 저장매체이다. 예컨대, 블록 디바이스로는 HDD(Hard Disk Drive), SSD(Solid State Drive) 등이 있다.
- [0022] 인메모리 데이터베이스(10)가 모든 쓰기(Write)/갱신(Update) 연산을 로그 파일에 기록하고 재 시작될 때, 기록된 쓰기/갱신 동작을 순차적으로 재 실행하여 데이터를 복구한다. 파일을 쓰는 시점에 관한 옵션으로 백그라운드 쓰레드에서 1초마다 플러싱을 수행하는 Everysec 방식, 메인 쓰레드에서 명령이 입력될 때마다 매번 플러싱을 수행하는 Always 방식, 또는 OS에 의해 설정된 시점에 플러싱을 수행하는 방식 등으로 설정할 수 있다.
- [0023] 인메모리 데이터베이스(10)는 키-값 기반으로 데이터를 저장한다. 키와 값을 한 쌍으로 저장하며, 키를 사용하여 값을 검색하거나 값을 저장하거나 값을 삭제하거나 값을 호출할 수 있다. 값의 데이터 타입은 해시(Hash), 정렬 집합(Sorted Set), 연결 리스트(Linked List), 스트링(String) 등으로 다양하게 정의될 수 있다.
- [0024] 메모리(200)는 로그 버퍼(210)를 갖고 로그 버퍼(210)는 명령어가 기록된 로그 기록을 저장한다. 로그 기록은 키-값 기반의 데이터일 수 있다. 메모리(200)는 키-값의 주소를 갖는 주소 리스트를 포함할 수 있다.
- [0025] 블록 디바이스(300)는 텍스트 파일 형식의 임시 로그 파일과 바이너리 파일 형식의 임시 스냅샷 파일을 각각 저장한다. 로그 파일은 AOF(Append Only File) 방식으로 스냅샷 파일은 RDB 방식으로 디스크 쓰기를 수행할 수 있다.
- [0026] AOF 방식은 입력/수정/삭제 명령이 실행될 때 마다 기록되며, 계속 추가하면서 기록된다. 추가하면 기록되는 동안 파일 사이즈가 계속 커지기 때문에, 특정 시점에 데이터 전체를 다시 쓰기(rewrite)를 수행한다. 다시 쓰기를 수행하면, 파일 사이즈가 작아 지게 된다. 예를 들어, SET 명령이 key는 같고 값을 다른 조건에서 5번 수행되었다고 하면, 메모리에는 마지막 수행된 값만 남는다. 다시 쓰기를 수행하면 이전 기록은 모두 사라지고 최종 데이터가 기록된다. AOF 파일은 텍스트 파일이며, 편집이 가능하다.
- [0027] RDB 방식은 특정 시점의 메모리에 있는 데이터 전체를 바이너리 파일로 저장한다. AOF 파일보다 사이즈가 작고, 로딩 속도가 AOF 파일보다 빠르다.
- [0028] 인메모리 데이터베이스(10)는 블록 디바이스(300)에 저장된 AOF 파일 및/또는 RDB 파일에 저장된 데이터를 읽고 메모리(200)에 데이터를 재구축하여 시스템을 정상 복구한다. 프로세서(100)는 로그 파일 또는 스냅샷 파일을 읽고 로그 파일 또는 스냅샷 파일에 기록된 키-값 기반의 처리 명령을 실행한다.
- [0029] 이하에서는 도 2 및 도 3을 참조하여, 인메모리 데이터베이스가 데이터를 백업하는 동작을 설명하기로 한다.
- [0030] 인메모리 데이터베이스는 메모리의 용량 중에서 일정 비율만큼의 용량을 데이터 저장 공간으로 설정할 수 있다. 인메모리 데이터베이스는 명령어가 기록된 로그 기록이 기 설정된 크기 또는 비율을 초과하면, 일부 또는 전부 데이터를 블록 디바이스로 방출한다. 로그 기록은 키-값 기반의 데이터로 구현될 수 있다.
- [0031] 인메모리 데이터베이스는 블록 디바이스에 임시 로그 파일을 생성한다. 임시 로그 파일은 텍스트 파일 형식으로 저장된다.
- [0032] 임시 로그 파일이 생성되면, 프로세서는 메모리의 로그 버퍼로부터 플러싱하는 저장 위치를 기억하는 로그 파일 서술자가 임시 로그 파일을 가리키도록 설정한다.
- [0033] 임시 로그 파일을 가리키도록 설정하면, 프로세서는 메모리 상에서 부모 프로세스를 동작시켜 메모리에 자식 프로세스를 생성한다. 즉, 포크(fork)를 수행한다. 자식 프로세스는 블록 디바이스에 바이너리 파일 형식의 임시 스냅샷 파일을 생성한다. 자식 프로세스는 현재 시점에서의 명령어 집합을 임시 스냅샷 파일에 저장한다. 명령어 집합은 동일한 키에 대해서 중복으로 입력된 값의 최종 데이터가 키-값 기반으로 매칭된 데이터 집합이다.
- [0034] 프로세서는 자식 프로세스가 임시 스냅샷 파일을 생성하는 동안에, 새로 입력된 명령어를 부모 프로세스를 동작시켜 로그 버퍼에 저장한다. 메모리는 다시 쓰기 버퍼가 사용하지 않는다. 즉, 메모리는 하나의 로그 버퍼를 사용한다.

- [0035] 프로세서는 메모리의 로그 버퍼로부터 블록 디바이스로 기 설정된 단위 시간마다 주기적으로 플러싱을 수행한다. 주기적으로 플러싱하는 방식은 크게 Everysec 방식과 Always 방식이 있다. Everysec 방식은 1초마다 플러싱을 수행하며, 불안정한 데이터 지속성을 갖지만 백그라운드 쓰레드에서 플러싱을 수행하기 때문에 처리속도에 영향을 주지 않는다. Always 방식은 명령이 입력될 때마다 매번 플러싱을 수행하며, 데이터 지속성을 유지하지만 메인 쓰레드에서 플러싱을 수행하기 때문에 처리속도를 저감시킨다.
- [0036] 프로세서는 임시 스냅샷 파일의 생성이 완료되면, 블록 디바이스에 저장된 정식 로그 파일을 삭제하고, 임시 로그 파일의 이름을 정식 로그 파일로 변경한다. 프로세서는 임시 스냅샷 파일의 생성이 완료되고 로그 파일의 이름이 변경되면, 임시 스냅샷 파일의 이름을 정식 스냅샷 파일로 변경한다. 즉, 블록 드라이브는 임시 로그 파일을 정식 로그 파일로 변경하고, 임시 스냅샷 파일을 정식 로그 파일로 변경한다.
- [0037] 본 실시예에 따른 인메모리 데이터 베이스는 정식 로그 파일과 정식 스냅샷 파일을 별도로 저장하여 관리함으로써, 로그 기록을 작은 단위로 주기적으로 플러싱하여 디스크 I/O 부담을 최소화하고, 메모리 사용량을 감소시키고 메모리 사용의 변동성을 최소화하고 데이터베이스의 처리 속도를 향상시킬 수 있다.
- [0038] 인메모리 데이터베이스에 포함된 구성요소들이 도 1에서는 분리되어 도시되어 있으나, 복수의 구성요소들은 상호 결합되어 적어도 하나의 모듈로 구현될 수 있다. 구성요소들은 장치 내부의 소프트웨어적인 모듈 또는 하드웨어적인 모듈을 연결하는 통신 경로에 연결되어 상호 간에 유기적으로 동작한다. 이러한 구성요소들은 하나 이상의 통신 버스 또는 신호선을 이용하여 통신한다.
- [0039] 인메모리 데이터베이스는 하드웨어, 펌웨어, 소프트웨어 또는 이들의 조합에 의해 로직회로 내에서 구현될 수 있고, 범용 또는 특정 목적 컴퓨터를 이용하여 구현될 수도 있다. 장치는 고정배선형(Hardwired) 기기, 필드 프로그래밍 가능한 게이트 어레이(Field Programmable Gate Array, FPGA), 주문형 반도체(Application Specific Integrated Circuit, ASIC) 등을 이용하여 구현될 수 있다. 또한, 장치는 하나 이상의 프로세서 및 컨트롤러를 포함한 시스템온칩(System on Chip, SoC)으로 구현될 수 있다.
- [0040] 인메모리 데이터베이스는 하드웨어적 요소가 마련된 컴퓨팅 디바이스에 소프트웨어, 하드웨어, 또는 이들의 조합하는 형태로 탑재될 수 있다. 컴퓨팅 디바이스는 각종 기기 또는 유무선 통신망과 통신을 수행하기 위한 통신 모듈 등의 통신장치, 프로그램을 실행하기 위한 데이터를 저장하는 메모리, 프로그램을 실행하여 연산 및 명령하기 위한 마이크로프로세서 등을 전부 또는 일부 포함한 다양한 장치를 의미할 수 있다.
- [0041] 도 4는 본 발명의 다른 실시예에 따른 인메모리 데이터베이스의 데이터 처리 방법을 예시한 흐름도이다.
- [0042] 데이터 처리 방법은 인메모리 데이터베이스에 의하여 수행될 수 있으며, 인메모리 데이터베이스가 수행하는 동작에 관한 상세한 설명과 중복되는 설명은 생략하기로 한다.
- [0043] 단계 S410에서, 인메모리 데이터베이스는 블록 디바이스에 텍스트 파일 형식의 임시 로그 파일을 생성한다.
- [0044] 단계 S420에서, 인메모리 데이터베이스는 임시 로그 파일을 생성하면, 메모리의 로그 버퍼로부터 플러싱하는 저장 위치를 기억하는 로그 파일 서술자가 임시 로그 파일을 가리키도록 설정한다.
- [0045] 단계 S430에서, 인메모리 데이터베이스는 임시 로그 파일을 가리키도록 설정하면, 블록 디바이스에 바이너리 파일 형식의 임시 스냅샷 파일을 생성한다.
- [0046] 프로세서는 부모 프로세스를 동작시켜 상기 메모리에 자식 프로세스를 생성하고, 자식 프로세스는 현재 시점에서 명령어 집합을 임시 스냅샷 파일로 저장한다. 프로세서는 자식 프로세스가 임시 스냅샷 파일을 저장하는 동안에, 새로 입력된 명령어를 부모 프로세스를 동작시켜 로그 버퍼에 저장한다. 메모리는 하나의 로그 버퍼를 사용하고, 다시 쓰기 버퍼를 사용하지 않는다.
- [0047] 프로세서는 임시 스냅샷 파일의 저장이 완료되면, 블록 디바이스에 저장된 정식 로그 파일을 삭제하고, 임시 로그 파일의 이름을 정식 로그 파일로 변경하고, 임시 스냅샷 파일의 이름을 정식 스냅샷 파일로 변경한다.
- [0048] 블록 드라이브는 정식 로그 파일과 정식 스냅샷 파일을 별도로 저장하여 관리한다.

[0049] 데이터베이스를 재구축하는 동작에 관한 알고리즘은 다음과 같다.

```

1  if AOF Current Size > AOF LESS Min Size then
    /* Create Temp AOF file */
2  TempAOF fd ← createFile(TempAOF)
    /* Change File Descriptor to Temp AOF file, command logs are
    stored to Temp AOF */
3  AOF fd ← TempAOF fd
    /* Create Child Process using fork system call */
4  childpid ← fork()
5  if childpid == 0 then /* Child process */
    /* Generate Temp RDB file */
6  retval = rdbGenerate(TempRDB, rdbSaveInfo)
7  if retval == OK then
    /* Send terminate signal to parent process */
8      exitCode ← OK
9      exitFromChild(exitCode)
10 else /* Parent process */
    /* This procedure everything directly called here will be called 10
    times per second */
11    ServerCron :
12        if TerminalChild() == True then
13            exitCode ← readexitCode()
14            if exitCode == OK then
                /* If success generating RDB file, rename AOF & RDB file
                */
15                renameFile(TempAOF, AOF filename)
16                renameFile(TempRDB, RDB filename)
17    End Procedure
18 End

```

[0050]

[0051] AOF_Current_Size는 현재 AOF 파일 크기이고, AOF_LESS_Min_Size는 알고리즘 시작시 최소 AOF 파일 크기이고, TempRDB는 임시 RDB 파일의 이름이고, TempAOF는 임시 AOF 파일의 이름이고, rdbSaveInfo는 RDB 파일에 관한 메타데이터이다.

[0052] 도 5 및 도 6은 본 발명의 실시예들에 따라 수행된 모의실험 결과를 도시한 것이다. 시뮬레이션한 컴퓨터 시스템은 64GB의 DDR3 RAM을 갖고, 인텔 제온 E5-2660 CPU를 동작시키고, 250G*3의 SSD를 사용하였다. SET과 GET의 비율을 조절하여 키를 빈번하게 갱신하였다.

[0053] 도 5를 참조하면, 본 실시예에 따른 데이터베이스(LESS)가 기존 방식보다 메모리 사용량을 1/5 수준으로 감소시키고, 변동성이 거의 없음을 쉽게 파악할 수 있다. 도 6을 참조하면, 본 실시예에 따른 데이터베이스(LESS)가 기존 방식보다 처리속도를 2~3배 향상시킬 수 있음을 쉽게 파악할 수 있다.

[0054] 도 4에서는 각각의 과정을 순차적으로 실행하는 것으로 기재하고 있으나 이는 예시적으로 설명한 것에 불과하고, 이 분야의 기술자라면 본 발명의 실시예의 본질적인 특성에서 벗어나지 않는 범위에서 도 4에 기재된 순서를 변경하여 실행하거나 또는 하나 이상의 과정을 병렬적으로 실행하거나 다른 과정을 추가하는 것으로 다양하게 수정 및 변형하여 적용 가능할 것이다.

[0055] 본 실시예들에 따른 동작은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능한 매체에 기록될 수 있다. 컴퓨터 판독 가능한 매체는 실행을 위해 프로세서에 명령어를 제공하는 데 참여한 임의의 매체를 나타낸다. 컴퓨터 판독 가능한 매체는 프로그램 명령, 데이터 파일, 데이터 구조 또는 이들의 조합을 포함할 수 있다. 예를 들면, 자기 매체, 광기록 매체, 메모리 등이 있을 수 있다. 컴퓨터 프로그램은 네트워크로 연결된 컴퓨터 시스템 상에 분산되어 분산 방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수도 있다. 본 실시예를 구현하기 위한 기능적인(Functional) 프로그램, 코드, 및 코드 세그먼트들은 본 실시예가 속하는 기술분야의 프로그래머들에 의해 용이하게 추론될 수 있을 것이다.

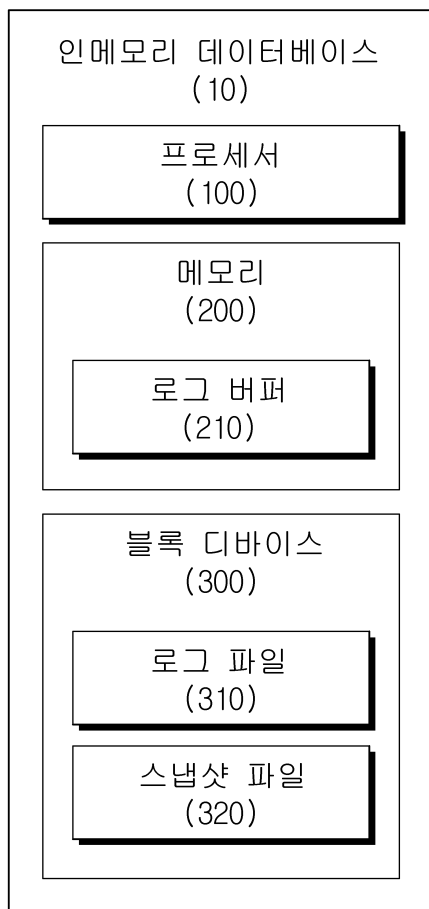
[0056] 본 실시예들은 본 실시예의 기술 사상을 설명하기 위한 것이고, 이러한 실시예에 의하여 본 실시예의 기술 사상의 범위가 한정되는 것은 아니다. 본 실시예의 보호 범위는 아래의 청구범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술 사상은 본 실시예의 권리범위에 포함되는 것으로 해석되어야 할 것이다.

부호의 설명

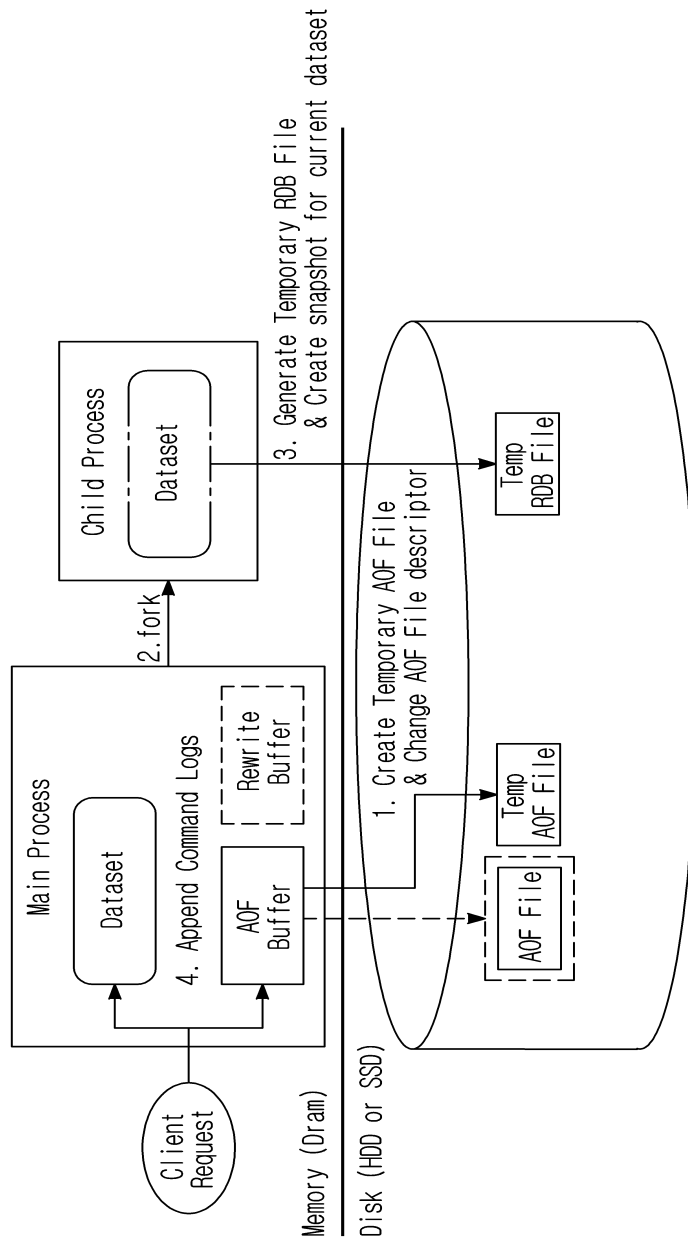
[0057] 10: 인메모리 데이터베이스 100: 프로세서
210: 로그 버퍼 300: 블록 디바이스
310: 로그 파일 320: 스냅샷 파일

도면

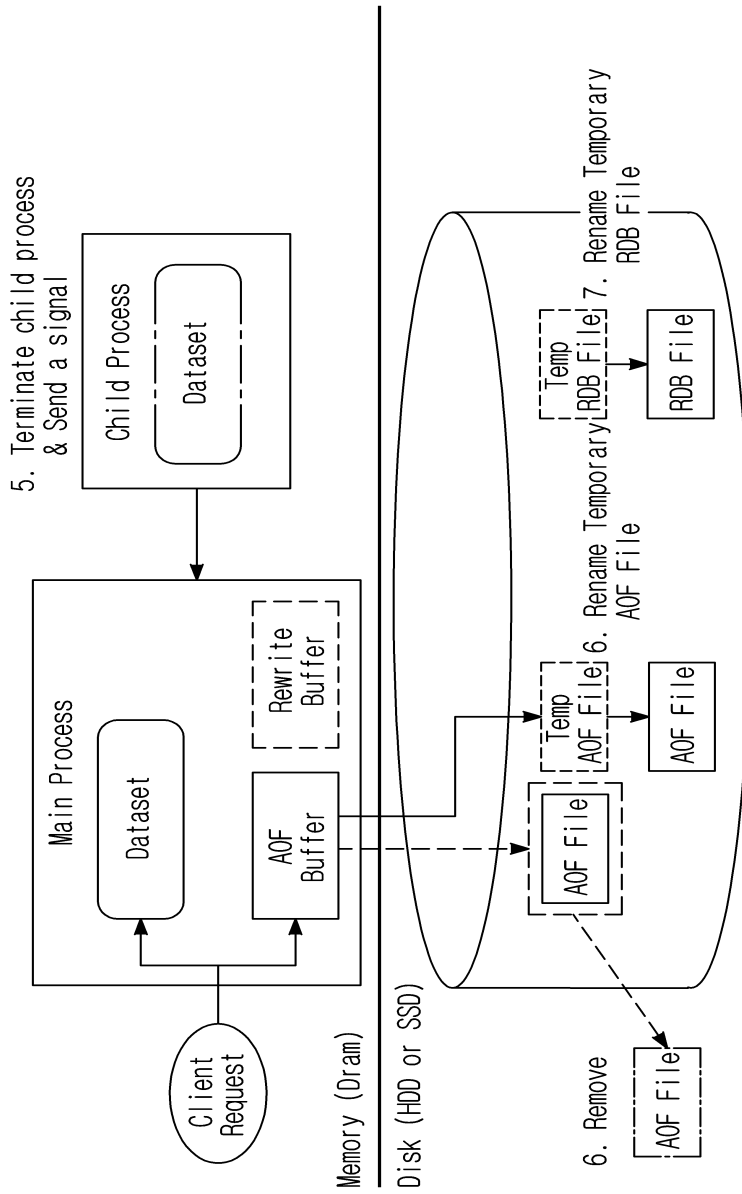
도면1



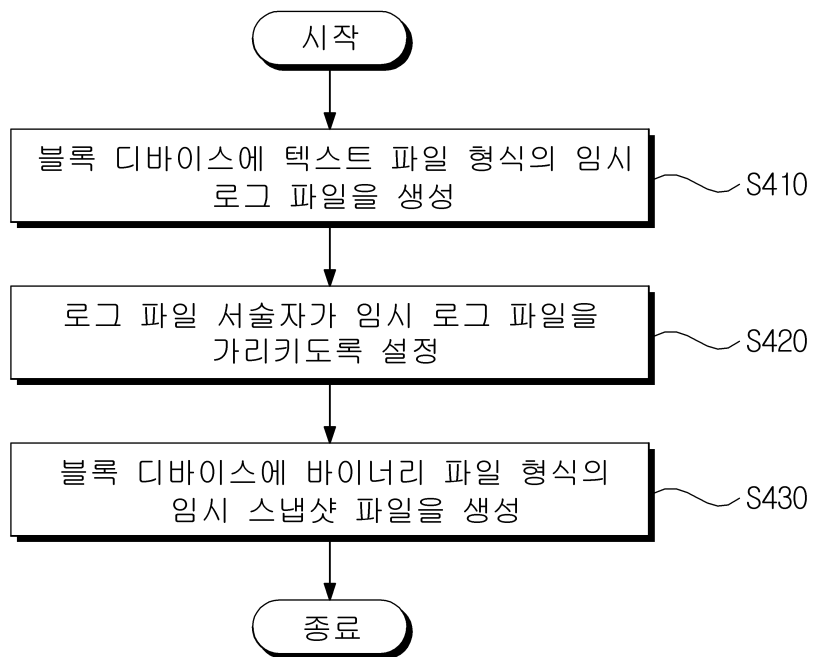
도면2



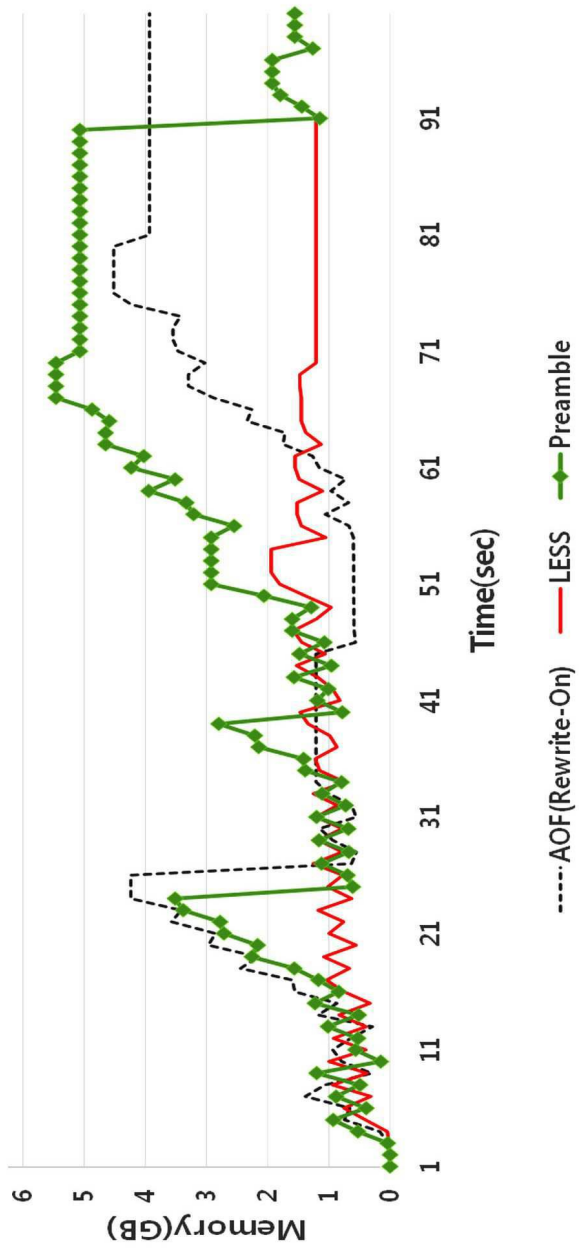
도면3



도면4



도면5



도면6

