2018-06-14

【서지사항】

【서류명】 특허출원서

【참조번호】 0289

【출원구분】 특허출원

【출원인】

【명칭】 연세대학교 산학협력단

【특허고객번호】 2-2005-009509-9

【대리인】

【명칭】 특허법인 우인

【대리인번호】 9-2006-100082-1

【지정된변리사】 양승식, 지현수

【포괄위임등록번호】 2007-057789-6

【발명의 국문명칭】 비휘발성 메모리를 이용한 인메모리 데이터베이스의 데이터 처리 방법 및 인메모리 데이터베이스

【발명의 영문명칭】 Method for Processing Data in In-Memory Database Using Non-Volatile Memory and In-Memory Database

【발명자】

【성명】 박상현

【성명의 영문표기】 PARK, Sang Hyun

【주민등록번호】 670101-1XXXXXX

【우편번호】 08004

【주소】 서울특별시 양천구 오목로 300, 204동 3701호 (목동, 현대하이페리온2)

【발명자】

【성명】 김도영

【성명의 영문표기】 KIM, Do Young

【주민등록번호】 930920-1XXXXXX

【우편번호】 07229

【주소】 서울특별시 영등포구 국회대로55길 28, 704호 (당산동)

【발명자】

【성명】 벅스텔러번트

【성명의 영문표기】 Burgstaller Bernd

【주소】 경기도 고양시 덕양구 호국로742번길 53, 804동 703호

【주소의 영문표기】 804-703, 53, Hoguk-ro 742beon-gil, Deogyang-gu, Goyang-si, Gyeonggi-do

【발명자】

【성명】 최원기

【성명의 영문표기】 CHOI, Won Gi

【주민등록번호】 910204-1XXXXXX

【우편번호】 03724

【주소】 서울특별시 서대문구 연희로16길 17, 202호 (연희동)

【출원언어】 국어

【심사청구】 청구

【이 발명을 지원한 국가연구개발사업】

【과제고유번호】 1711065240

【부처명】 과학기술정보통신부

【연구관리 전문기관】 정보통신기술진흥센터(NIPA산하)

【연구사업명】 정보통신방송연구개발사업

【연구과제명】 [이지바로] (SW스타랩)IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발 (2/8)

【기여율】 1/2

【주관기관】 연세대학교 산학협력단

【연구기간】 2018.01.01 ~ 2018.12.31

【이 발명을 지원한 국가연구개발사업】

【과제고유번호】 1711060376

【부처명】 과학기술정보통신부

【연구관리 전문기관】 한국연구재단

【연구사업명】 원천기술개발사업

【연구과제명】 [Ezbaro]이종 멀티코어 기반의 클라우드 상에서 프로그래머 생산성 및 퍼포먼스를 위한 엑사스케일 빅 데이터 분석 플랫폼(1/2단계)(3/3연차)

【기여율】 1/2

【주관기관】 연세대학교

【연구기간】 2017.11.01 ~ 2018.08.31

【취지】 위와 같이 특허청장에게 제출합니다.

대리인 특허법인 우인 (서명 또는 인)

【수수료】

【출원료】	0	면	46,000 원
【가산출원료】	37	면	0 원
【우선권주장료】	0	건	0 원
【심사청구료】	17	항	891,000 원
【합계】	937,000 원		
【감면사유】	전담조직(50%감면)[1]		
【감면후 수수료】	468,500 원		

【수수료 자동납부번호】 05801106348

【발명의 설명】

【발명의 명칭】

비휘발성 메모리를 이용한 인메모리 데이터베이스의 데이터 처리 방법 및 인메모리 데이터베이스 {Method for Processing Data in In-Memory Database Using Non-Volatile Memory and In-Memory Database}

【기술분야】

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

【발명의 배경이 되는 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 주로 휘발성이기 때문에, 데이터베이스 서버의 전원이 꺼지면 저장매체에 있는 자료들이 삭제되어 버리는 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.

【0005】 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.

【0006】주기적으로 플러싱하는 방식은 크게 Everysec 방식과 Always 방식이 있다.

【0007】Everysec 방식은 1초마다 플러싱을 수행하며, 불안정한 데이터 지속성을 갖지만 백그라운드 쓰레드에서 플러싱을 수행하기 때문에 처리속도에 영향을 주지 않는다.

【0008】Always 방식은 명령이 입력될 때마다 매번 플러싱을 수행하며, 데이터 지속성을 유지하지만 메인 쓰레드에서 플러싱을 수행하기 때문에 처리속도를 저감시키는 문제가 있다.

【0009】데이터 지속성을 유지하기 위해서 비휘발성 메모리(NVM)를 이용하는 인메모리 데이터베이스가 있다. 비휘발성 메모리를 이용하면 Everysec 방식의 처리속도 수준으로 성능이 향상된다. 다만 비휘발성 메모리 제품은 현재 상용화 단계가 아니므로, 용량 및 가격에서 한계가 있다. 현재까지는 비휘발성 메모리로 구축한 인메모리 데이터베이스에 대용량의 데이터를 저장하기가 곤란한 실정이다.

【선행기술문헌】

【특허문헌】

【0010】**(특허문헌 0001)** 한국등록특허공보 제10-1190001호 (2012.10.05.)

(특허문헌 0002) 한국공개특허공보 제10-2016-0121819호 (2016.10.21.)

【발명의 내용】

【해결하고자 하는 과제】

【0011】 본 발명의 실시예들은 인메모리 데이터베이스가 비휘발성 메모리를 1차 저장소로 사용하고, 비휘발성 메모리의 일정 용량을 초과한 데이터에 관하여 휘발성 메모리를 2차 저장소로 사용하고, 휘발성 메모리에 저장된 데이터에 관한 로그 파일을 블록 디바이스에 주기적으로 저장함으로써, 비휘발성 메모리의 용량 한계를 극복하면서 데이터 지속성을 확보하는 데 발명의 주된 목적이 있다.

【0012】 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】

【0013】 본 실시예의 일 측면에 의하면, 인메모리 데이터베이스의 데이터 처리 방법에 있어서, 상기 인메모리 데이터베이스의 비휘발성 메모리에 우선적으로 데이터를 저장하는 단계, 및 상기 인메모리 데이터베이스의 비휘발성 메모리에 저장된 데이터의 일부 데이터를 상기 인메모리 데이터베이스의 휘발성 메모리에 백업하는 단계를 포함하는 인메모리 데이터베이스의 데이터 처리 방법을 제공한다.

【0014】 본 실시예의 다른 측면에 의하면, 프로세서, 비휘발성 메모리, 휘발성 메모리, 및 블록 디바이스를 포함하며, 상기 비휘발성 메모리에 우선적으로 데이터를 저장하고, 상기 비휘발성 메모리에 저장된 데이터의 일부 데이터를 상기 휘발성 메모리에 백업하는 것을 특징으로 하는 인메모리 데이터베이스를 제공한다.

【0015】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, 상기 인메모리 데이터베이스의 비휘발성 메모리에 우선적으로 데이터를 저장하는 단계, 및 상기 인메모리 데이터베이스의 비휘발성 메모리에 저장된 데이터의 일부 데이터를 상기 인메모리 데이터베이스의 휘발성 메모리에 백업하는 단계를 포함한 동작들을 수행하는 컴퓨터 프로그램을 제공한다.

【발명의 효과】

【0016】이상에서 설명한 바와 같이 본 발명의 실시예들에 의하면, 인메모리 데이터베이스가 비휘발성 메모리를 1차 저장소로 사용하고, 비휘발성 메모리의 일정 용량을 초과한 데이터에 관하여 휘발성 메모리를 2차 저장소로 사용하고, 휘발성 메모리에 저장된 데이터에 관한 로그 파일을 블록 디바이스에 주기적으로 저장함으로써, 비휘발성 메모리의 용량 한계를 극복하면서 데이터 지속성을 확보할 수 있는 효과가 있다.

【0017】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】

【0018】도 1은 본 발명의 일 실시예에 따른 인메모리 데이터베이스를 예시한 블록도이다.

도 2는 본 발명의 일 실시예에 따른 인메모리 데이터베이스가 데이터를 백업하는 동작을 예시한 흐름도이다.

도 3a 및 도 3b는 본 발명의 일 실시예에 따른 인메모리 데이터베이스가 데이터를 백업하는 동작을 예시한 도면이다.

도 4는 본 발명의 일 실시예에 따른 인메모리 데이터베이스가 키-값을 갱신하는 동작을 예시한 흐름도이다.

도 5는 본 발명의 일 실시예에 따른 인메모리 데이터베이스가 키-값을 갱신하는 동작을 예시한 도면이다.

도 6은 본 발명의 일 실시예에 따른 인메모리 데이터베이스가 데이터를 복구하는 동작을 예시한 흐름도이다.

도 7은 본 발명의 다른 실시예에 따른 인메모리 데이터베이스의 데이터 처리 방법을 예시한 흐름도이다.

도 8a 내지 도 8d는 본 발명의 실시예들에 따라 수행된 모의실험 결과를 도시한 것이다.

【발명을 실시하기 위한 구체적인 내용】

【0019】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.

【0020】 도 1은 인메모리 데이터베이스를 예시한 블록도이다. 도 1에 도시한 바와 같이, 인메모리 데이터베이스(10)는 프로세서(100), 비 휘발성 메모리(200), 휘발성 메모리(300), 및 블록 디바이스(400)를 포함한다. 장치(100)는 도 1에서 예시적으로 도시한 다양한 구성요소들 중에서 일부 구성요소를 생략하거나 다른 구성요소를 추가로 포함할 수 있다.

【0021】 인메모리 데이터베이스(10)는 디스크가 아닌 메모리에 모든 데이터를 보유하고 있는 데이터베이스이다. 비 휘발성 메모리(200)는 전원이 공급되지 않아도 저장된 정보를 계속 유지하는 메모리이다. 휘발성 메모리(300)는 저장된 정보를 계속 유지하기 위하여 전원 공급이 필요한 메모리이다. 예컨대, 휘발성 메모리(300)로는 DRAM(Dynamic Random Access Memory) 등이 있다. 블록 디바이스(400)는 블록 단위로 임의 접근이 가능한 저장매체이다. 예컨대, 블록 디바이스(400)로는 HDD(Hard Disk Drive), SSD(Solid State Drive) 등이 있다. 비 휘발성 메모리(200)는 SSD(Solid State Drive) 등으로 구현될 수도 있다.

【0022】 인메모리 데이터베이스(10)는 비휘발성 메모리(200)를 1차 저장소로 사용하고, 비휘발성 메모리(200)의 일정 용량을 초과한 데이터에 관하여 휘발성 메모리(300)를 2차 저장소로 사용하고, 휘발성 메모리(300)에 저장된 데이터에 관한 로그 파일을 블록 디바이스(400)에 주기적으로 저장함으로써, 비휘발성 메모리의 용량 한계를 극복하면서 데이터 지속성을 확보한다.

【0023】모든 쓰기(Write)/갱신(Update) 연산을 로그 파일에 기록하는 데이터베이스 서버가 재 시작될 때, 기록된 쓰기/갱신 동작을 순차적으로 재 실행하여 데이터를 복구한다. 파일을 쓰는 시점에 관한 옵션으로 백그라운드 쓰레드에서 1초마다 플러싱을 수행하는 Everysec 방식 또는 메인 쓰레드에서 명령이 입력될 때마다 매번 플러싱을 수행하는 Always 방식으로 설정할 수 있다.

【0024】인메모리 데이터베이스(10)는 Always 방식과 같이 데이터 지속성을 갖고, Everysec 방식의 처리 성능을 확보할 수 있다.

【0025】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.

【0026】프로세서(100)는 비휘발성 메모리(200), 휘발성 메모리(300), 블록 디바이스(400)에 기 정의된 명령어를 전송하여, 각종 신호 및 데이터 흐름을 제어한다.

【0027】비휘발성 메모리(200)는 키-값의 주소를 갖는 주소 리스트를 포함한다. 주소 리스트(210)는 선형 데이터 구조이며, 헤드(Head)와 리어(Rear)를 갖고, 키의 주소와 값의 주소를 각각 갖는다.

【0028】 휘발성 메모리(300)는 주소 리스트(210)를 참조하여 키-값을 복구하기 위하여 키-값을 참조하는 데이터 구조(310)를 포함한다. 휘발성 메모리(300)의 데이터 구조(310)는 비휘발성 메모리(200)에 저장된 데이터 및 휘발성 메모리(300)에 저장된 데이터를 관리한다. 데이터 구조(310)는 해시 함수를 이용하여 데이터를 관리할 수 있다. 데이터 구조(310)의 엔트리(Entry)는 비휘발성 메모리(200)에 저장된 키-값을 참조하거나 휘발성 메모리(300)에 저장된 키-값을 참조한다. 휘발성 메모리(300)는 로그 버퍼(320)를 포함한다. 로그 버퍼(320)는 키-값의 처리 명령에 관한 로그를 기록한다.

【0029】 블록 디바이스(400)는 로그에 관한 로그 파일을 저장한다.

【0030】 이하에서는 도 2, 도 3a 및 도 3b를 참조하여, 인메모리 데이터베이스가 데이터를 백업하는 동작을 설명하기로 한다.

【0031】 인메모리 데이터베이스(10)는 비휘발성 메모리(200)에 저장된 데이터가 기 설정된 비율을 초과하면, 일부 데이터를 휘발성 메모리(300)로 방출한다. 인메모리 데이터베이스(10)는 비휘발성 메모리(200)의 용량 중에서 일정 비율만큼의 용량을 데이터 저장 공간으로 설정할 수 있다.

【0032】 단계 S210에서, 프로세서(100)는 비휘발성 메모리(200)의 주소 리스트(210) 중에서 희생 노드를 선택한다. 단계 S220에서, 프로세서(100)는 선택된 희생 노드의 키-값을 휘발성 메모리(300)에 복사한다. 단계 S230에서, 휘발성 메모리(300)의 데이터 구조(310)는 휘발성 메모리(300)에 복사된 키-값을 참조한다. 단계

S240에서, 휘발성 메모리(300)의 로그 버퍼(320)는 희생 노드의 키-값의 처리 명령에 관한 로그를 기록한다. 단계 S250에서, 로그 버퍼(320)는 로그를 블록 디바이스(400)의 로그 파일로 플러싱한다. 단계 S260에서, 플러싱을 완료하면, 프로세서(100)는 비휘발성 메모리(200)의 주소 리스트(210)에서 희생 노드를 삭제한다. 로그 작성이 완료된 후에 희생 노드를 삭제하므로, Everysec 방식으로 운용하더라도 비휘발성 메모리를 이용하여 데이터 지속성을 유지할 수 있다.

【0033】도 3a를 참조하면, NVM의 Pool 중에서 희생 노드에 관한 희생 리스트를 선택하고, 희생 노드에 해당하는 (K, V)를 DRAM에 저장하고, 키-값의 처리 명령을 로그 버퍼에 저장한다.

【0034】키-값의 처리 명령은 (SET K, V), (DEL K, V) 등으로 정의될 수 있다.

【0035】AOF(Append On File) 방식은 키-값 기반의 데이터베이스 중에서 하나인 레디스(Redis)가 모든 쓰기(Write)/갱신(Update) 연산을 로그 파일에 기록하는 방식이다. 데이터베이스 서버가 재 시작될 때, 기록된 쓰기/갱신 동작을 순차적으로 재 실행하여 데이터를 복구한다. 파일을 쓰는 시점에 관한 옵션으로 'Everysec' 또는 'Always'로 설정할 수 있다. Everysec 방식은 백그라운드 쓰레드에서 1초마다 플러싱을 수행한다. Always 방식은 메인 쓰레드에서 명령이 입력될 때마다 매번 플러싱을 수행한다.

【0036】PMDK(Persistent Memory Development Kit) API는 NVM를 이용하여 키-값 기반의 데이터베이스를 제공한다.

【0037】도 3b를 참조하면, 로그 버퍼(320)로부터 블록 디바이스(400)의 로그 파일로 플러싱을 완료하면, 주소 리스트(210)에서 희생 노드를 삭제한다. 로그 버퍼(320)의 로그 역시 삭제된다. 주소 리스트(210)는 희생 노드를 제외하고 남은 노드들을 저장한다.

【0038】본 실시예에 따른 인메모리 데이터베이스(10)는 비휘발성 메모리(200)의 기 설정된 용량에 관한 비율을 초과하지 않는 데이터는 비휘발성 메모리(200)에 저장하고, 비휘발성 메모리(200)의 기 설정된 비율을 초과하는 데이터는 휘발성 메모리(300)에 저장한다. 초과하는 데이터, 즉, 희생 노드를 선택하여 데이터를 방출하는 순서는 최저 사용 빈도(Least Recently Used, LRU) 정책에 따라 결정된다.

【0039】이하에서는 도 4 및 도 5를 참조하여, 인메모리 데이터베이스가 키-값을 갱신하는 동작을 설명하기로 한다.

【0040】인메모리 데이터베이스(10)는 최저 사용 빈도(Least Recently Used, LRU) 정책에 따라 희생 노드를 선택한다. 비휘발성 메모리(200)를 1차 저장소로 사용하고, 비휘발성 메모리(200)의 일정 용량을 초과한 데이터에 관하여 휘발성 메모리(300)를 2차 저장소로 사용하기 때문에, 휘발성 메모리(300)에 저장된 데이터가 갱신되면 갱신된 데이터는 비휘발성 메모리(200)에 저장된다. 최저 사용 빈도 정책

을 이용하면, 비휘발성 메모리(200)에서 휘발성 메모리(300)로 데이터를 방출하는 과정을 최소화할 수 있다.

【0041】 갱신이 자주 일어나지 않는 데이터는 더 이상 새로운 데이터를 쓰지 않고 데이터를 읽기만 할 확률이 크다. 제안한 모델에서는 플러싱된 데이터에 대해 갱신이 일어나면 해당 데이터를 비휘발성 메모리로 복귀시킨다. 그러므로 최저 사용 빈도 정책을 이용하면 갱신시 발생하는 재방출 과정을 줄임으로써 비휘발성 메모리에서 휘발성 메모리로 데이터를 방출하는 과정을 최소화할 수 있다.

【0042】 LRU 선정 과정에서 부하를 줄이기 위하여, 비휘발성 메모리(300)의 주소리스트를 LRU 우선 큐(Priority Queue)로 구현할 수 있다.

【0043】 비휘발성 메모리(300)에 저장된 데이터가 갱신되면, 단계 S410에서, 주소 리스트(210)의 헤드에 신규 노드를 삽입한다. 단계 S420에서, 신규 노드는 갱신된 데이터를 참조한다. 즉, 신규 노드의 엔트리는 갱신된 값을 참조하고, 갱신되기 전의 데이터를 가진 노드의 키를 참조한다. 단계 S430에서, 주소 리스트(210)는 갱신되기 전의 데이터를 가진 노드를 삭제하고, 삭제된 노드의 선행 노드와 후행 노드를 연결하여, 주소 리스트(210)를 재정렬한다. 데이터를 갱신하는 과정에서 사용 빈도에 따라 자동으로 주소 리스트(210)가 정렬된다. 주소 리스트(210)의 리어 또는 리어를 포함하는 희생 리스트를 선택하고 삭제하기가 용이하다.

【0044】 도 5를 참조하면, (K_B, V_B)를 참조하는 노드의 V_B를 V'으로 변경할 때, 주소 리스트의 헤드에 신규 노드를 삽입한다. 신규 노드는 K_B를 참조한다. (K_B, V_B)를 참조하는 노드를 삭제하고, 주소 리스트를 연결함으로써, 리스트 내에서 값을

변경한 노드의 순서를 변경한다. 즉, 값을 갱신한 노드를 주소 리스트의 헤드에 위치시킬 수 있다.

【0045】 이하에서는 도 6을 참조하여, 인메모리 데이터베이스가 데이터를 복구하는 동작을 설명하기로 한다.

【0046】 프로세서(100)는 비휘발성 메모리(200)의 주소 리스트 및 블록 디바이스(400)의 로그 파일을 이용하여 데이터를 복구한다.

【0047】 단계 S610에서, 프로세서(100)는 블록 디바이스(400)의 로그 파일을 기반으로 휘발성 메모리(300)의 데이터 구조(310)를 복구한다. 프로세서(100)는 로그 파일을 읽고 로그 파일에 기록된 키-값 기반의 처리 명령을 실행한다.

【0048】 단계 S620에서, 프로세서(100)는 주소 리스트(210)에서 삭제되지 않은 희생 노드의 키-값을 휘발성 메모리(300)에 복구한다. 프로세서(100)는 희생 리스트의 헤드부터 리어에 해당하는 키와 값을 획득한다. 희생 노드의 키-값의 처리 명령에 관한 로그가 아직 기록되지 않았기 때문에, 휘발성 메모리(300)의 로그 버퍼(320)는 희생 노드의 키-값의 처리 명령에 관한 로그를 다시 기록한다. 로그 버퍼(320)는 로그를 블록 디바이스(400)의 로그 파일로 플러싱한다. 백그라운드 쓰레드에서 플러싱을 수행할 수 있다. 플러싱을 완료하면, 프로세서(100)는 비휘발성 메모리(200)의 주소 리스트(210)에서 희생 리스트를 삭제한다.

【0049】 단계 S630에서, 프로세서(100)는 주소 리스트(210)를 이용하여 비휘발성 메모리(200)의 키-값을 복구한다. 프로세서(100)는 주소 리스트(210)의 헤드

부터 리어에 해당하는 키와 값을 획득한다. 프로세서(100)는 휘발성 메모리(300)의 데이터 구조(310)에 키-값을 추가한다.

【0050】 데이터베이스를 복구하는 동작에 관한 알고리즘은 다음과 같다.

```
    /* Reconstruct data from AOF log file */
1   cmds = loadAppendOnlyFile();
    While (cmd of cmds) {
       processCommand(cmd);
    }

    /* Reconstruct data from NVM victim list */
2   victimHead = getVictimHeadPtr(NVMPool);
    While (victimHead != NULL) {
       key = getKeyFromNVMNode(victimHead);
       val = getValFromNVMNode(victimHead);
       writeAppendOnlyLog(key, val);
       victimHead = victimHead->next;
    }
    /* Flush AOF log buffer with removing victims */
    flushAppendOnlyInBackground();

    /* Reconstruct data from NVM list */
3   NVMHead = getNVMHeadPtr(NVMPool);
    While (NVMHead != NULL) {
       key = getKeyFromNVMNode(NVMHead);
       val = getValFromNVMNode(NVMHead);
       addKeyValToDatabase(key, val);
       NVMHead = NVMHead->next;
    }
```

【0051】

【0052】 본 실시예에 따른 데이터베이스에서 희생 노드는 로그 파일 또는 주소 리스트 중에서 하나에는 저장되어 있기 때문에, 데이터베이스의 지속성을 확보

할 수 있다.

【0053】 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.

【0054】 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 프로그램 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.

【0055】 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 모뎀 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.

【0056】 도 7은 본 발명의 다른 실시예에 따른 인메모리 데이터베이스의 데

이터 처리 방법을 예시한 흐름도이다.

【0057】 데이터 처리 방법은 인메모리 데이터베이스에 의하여 수행될 수 있으며, 인메모리 데이터베이스가 수행하는 동작에 관한 상세한 설명과 중복되는 설명은 생략하기로 한다.

【0058】 단계 S710에서, 인메모리 데이터베이스는 비휘발성 메모리에 우선적으로 데이터를 저장한다. 비휘발성 메모리에 저장된 데이터는 키-값 기반의 데이터이며, 비휘발성 메모리는 키-값의 주소를 갖는 주소 리스트를 포함한다.

【0059】 단계 S720에서, 인메모리 데이터베이스는 비휘발성 메모리에 저장된 데이터의 일부 데이터를 인메모리 데이터베이스의 휘발성 메모리에 백업한다. 휘발성 메모리는 주소 리스트를 참조하여 키-값을 복구하기 위하여 키-값을 참조하는 데이터 구조를 포함한다.

【0060】 휘발성 메모리의 데이터 구조는 비휘발성 메모리에 저장된 데이터 및 휘발성 메모리에 저장된 데이터를 관리하며, 비휘발성 메모리에 저장된 키-값을 참조하거나 휘발성 메모리에 저장된 키-값을 참조한다.

【0061】 백업하는 단계(S720)는 비휘발성 메모리에 저장된 데이터가 기 설정된 용량에 관한 비율을 초과하면, 일부 데이터를 휘발성 메모리로 방출한다. 일부 데이터를 휘발성 메모리로 방출하는 것을 (i) 비휘발성 메모리의 주소 리스트 중에서 희생 노드를 선택하고, (ii) 희생 노드의 키-값을 휘발성 메모리에 복사하고, (iii) 휘발성 메모리의 데이터 구조는 휘발성 메모리에 복사된 키-값을 참조하고,

(iv) 희생 노드의 키-값의 처리 명령에 관한 로그를 휘발성 메모리의 로그 버퍼에 기록하고, (v) 로그를 로그 버퍼로부터 블록 디바이스의 로그 파일로 플러싱하고, (vi) 플러싱을 완료하면 비휘발성 메모리의 상기 주소 리스트에서 희생 노드를 삭제할 수 있다.

【0062】 희생 노드를 선택하여 상기 데이터를 방출하는 순서는 최저 사용 빈도(Least Recently Used, LRU) 정책에 따라 결정될 수 있다.

【0063】 휘발성 메모리에 저장된 데이터가 갱신되면 갱신된 데이터는 비휘발성 메모리에 저장될 수 있다. 비휘발성 메모리에 저장된 데이터가 갱신되면, (i) 주소 리스트의 헤드에 신규 노드를 삽입하고, (ii) 신규 노드가 갱신된 데이터를 참조하고, (iii) 갱신되기 전의 데이터를 가진 노드를 삭제하여, 주소 리스트를 사용 빈도 순으로 재정렬할 수 있다.

【0064】 데이터 처리 방법은 주소 리스트 및 로그 파일을 이용하여 데이터를 복구하는 단계를 추가로 포함할 수 있다. 데이터를 복구하는 단계는 (i) 블록 디바이스의 로그 파일을 기반으로 휘발성 메모리의 데이터 구조를 복구하고, (ii) 주소 리스트에서 삭제되지 않은 희생 노드의 키-값을 휘발성 메모리에 복구하고 희생 노드의 키-값의 처리 명령에 관한 로그를 생성하고, (iii) 주소 리스트로부터 비휘발성 메모리의 키-값을 복구할 수 있다. 즉, 기록중인 로그를 복구할 수 있으므로 데이터 지속성을 보장할 수 있다.

【0065】도 8a 내지 도 8d는 본 발명의 실시예들에 따라 수행된 모의실험 결과를 도시한 것이다.

【0066】시뮬레이션한 컴퓨터 시스템은 64GB의 DRAM(2400 MHz)을 갖는 인텔 i7 6700K CPU를 동작시키고, AOF 로그 파일을 Seagate 3TB HDD(7200RPM)에 저장하고, PMDK에서 제공하는 에뮬레이터를 이용하여 DRAM에서 8GB의 NVM을 설정하였다. 도 8a는 데이터의 사이즈를 32B, 128B, 512B, 2048B, 8192B로 증가시키고, 0부터 1,000,000까지의 키를 거의 갱신하지 않고 설정하였고, 도 8b는 0부터 100까지의 키를 빈번하게 갱신하였다. 본 실시예에 따른 데이터베이스(NDHedis)가 Redis-Everysec 또는 PMDK-Redis의 처리 속도에 준함을 쉽게 확인할 수 있다.

【0067】도 8c는 SET과 GET의 비율을 3 대 7로 설정하고, 도 8 d은 SET과 GET의 비율을 7 대 3으로 설정하였다. 본 실시예에 따른 데이터베이스(NDHedis)는 Redis-Everysec와 달리 NVM에서 데이터를 갱신하기 때문에, 데이터 사이즈가 증가하더라도 처리 속도가 거의 일정하다. 본 실시예에 따른 데이터베이스(NDHedis)는 PMDK-Redis의 처리 속도에 준하면서, NVM의 용량 한계를 극복할 수 있다.

【0068】도 2, 도 4, 도 6, 및 도 7에서는 각각의 과정을 순차적으로 실행하는 것으로 기재하고 있으나 이는 예시적으로 설명한 것에 불과하고, 이 분야의 기술자라면 본 발명의 실시예의 본질적인 특성에서 벗어나지 않는 범위에서 도 2, 도 4, 도 6, 및 도 7에 기재된 순서를 변경하여 실행하거나 또는 하나 이상의 과정을 병렬적으로 실행하거나 다른 과정을 추가하는 것으로 다양하게 수정 및 변형하여 적용 가능할 것이다.

【0069】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.

【0070】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】

【0071】10: 인메모리 데이터베이스　100: 프로세서

200: 비 휘발성 메모리　　210: 주소 리스트

300: 휘발성 메모리　　310: 데이터 구조

320: 로그 버퍼

400: 블록 디바이스

【청구범위】

【청구항 1】

인메모리 데이터베이스의 데이터 처리 방법에 있어서,

상기 인메모리 데이터베이스의 비휘발성 메모리에 우선적으로 데이터를 저장하는 단계; 및

상기 인메모리 데이터베이스의 비휘발성 메모리에 저장된 데이터의 일부 데이터를 상기 인메모리 데이터베이스의 휘발성 메모리에 백업하는 단계

를 포함하는 인메모리 데이터베이스의 데이터 처리 방법.

【청구항 2】

제1항에 있어서,

상기 비휘발성 메모리에 저장된 데이터는 키-값 기반의 데이터이며,

상기 비휘발성 메모리는 상기 키-값의 주소를 갖는 주소 리스트를 포함하며,

상기 휘발성 메모리는 상기 주소 리스트를 참조하여 상기 키-값을 복구하기 위하여 상기 키-값을 참조하는 데이터 구조를 포함하는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

【청구항 3】

제2항에 있어서,

상기 휘발성 메모리의 데이터 구조는 상기 비휘발성 메모리에 저장된 데이터 및 상기 휘발성 메모리에 저장된 데이터를 관리하며, 상기 비휘발성 메모리에 저장

된 키-값을 참조하거나 상기 휘발성 메모리에 저장된 키-값을 참조하는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

【청구항 4】

제2항에 있어서,

상기 백업하는 단계는 상기 비휘발성 메모리에 저장된 데이터가 기 설정된 비율을 초과하면, 상기 일부 데이터를 상기 휘발성 메모리로 방출하는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

【청구항 5】

제4항에 있어서,

상기 일부 데이터를 상기 휘발성 메모리로 방출하는 것을 (i) 상기 비휘발성 메모리의 상기 주소 리스트 중에서 희생 노드를 선택하고, (ii) 상기 희생 노드의 키-값을 상기 휘발성 메모리에 복사하고, (iii) 상기 휘발성 메모리의 데이터 구조는 상기 휘발성 메모리에 복사된 키-값을 참조하고, (iv) 상기 희생 노드의 키-값의 처리 명령에 관한 로그를 상기 휘발성 메모리의 로그 버퍼에 기록하고, (v) 상기 로그를 상기 로그 버퍼로부터 블록 디바이스의 로그 파일로 플러싱하고, (vi) 플러싱을 완료하면 상기 비휘발성 메모리의 상기 주소 리스트에서 상기 희생 노드를 삭제하는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

【청구항 6】

제5항에 있어서,

상기 희생 노드를 선택하여 상기 데이터를 방출하는 순서는 최저 사용 빈도(Least Recently Used, LRU) 정책에 따라 결정되는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

【청구항 7】

제2항에 있어서,

상기 휘발성 메모리에 저장된 데이터가 갱신되면 상기 갱신된 데이터는 상기 비휘발성 메모리에 저장되고,

상기 비휘발성 메모리에 저장된 데이터가 갱신되면, (i) 상기 주소 리스트의 헤드에 신규 노드를 삽입하고, (ii) 상기 신규 노드가 상기 갱신된 데이터를 참조하고, (iii) 갱신되기 전의 데이터를 가진 노드를 삭제하여, 상기 주소 리스트를 재정렬하는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

【청구항 8】

제2항에 있어서,

상기 주소 리스트 및 로그 파일을 이용하여 데이터를 복구하는 단계를 추가로 포함하며,

상기 데이터를 복구하는 단계는 (i) 상기 블록 디바이스의 로그 파일을 기반으로 상기 휘발성 메모리의 데이터 구조를 복구하고, (ii) 상기 주소 리스트에서 삭제되지 않은 희생 노드의 키-값을 상기 휘발성 메모리에 복구하고 상기 희생 노드의 키-값의 처리 명령에 관한 로그를 생성하고, (iii) 상기 주소 리스트로부터

상기 비휘발성 메모리의 키-값을 복구하는 것을 특징으로 하는 인메모리 데이터베이스의 데이터 처리 방법.

【청구항 9】

프로세서, 비휘발성 메모리, 휘발성 메모리, 및 블록 디바이스를 포함하며,

상기 비휘발성 메모리에 우선적으로 데이터를 저장하고, 상기 비휘발성 메모리에 저장된 데이터의 일부 데이터를 상기 휘발성 메모리에 백업하는 것을 특징으로 하는 인메모리 데이터베이스.

【청구항 10】

제9항에 있어서,

상기 비휘발성 메모리에 저장된 데이터는 키-값 기반의 데이터이며,

상기 비휘발성 메모리는 상기 키-값의 주소를 갖는 주소 리스트를 포함하며,

상기 휘발성 메모리는 상기 주소 리스트를 참조하여 상기 키-값을 복구하기 위하여 상기 키-값을 참조하는 데이터 구조를 포함하는 것을 특징으로 하는 인메모리 데이터베이스.

【청구항 11】

제10항에 있어서,

상기 휘발성 메모리의 데이터 구조는 상기 비휘발성 메모리에 저장된 데이터 및 상기 휘발성 메모리에 저장된 데이터를 관리하며, 상기 비휘발성 메모리에 저장된 키-값을 참조하거나 상기 휘발성 메모리에 저장된 키-값을 참조하는 것을 특징

으로 하는 인메모리 데이터베이스.

【청구항 12】

제10항에 있어서,

상기 프로세서는 상기 비휘발성 메모리에 저장된 데이터가 기 설정된 비율을 초과하면, 상기 일부 데이터를 상기 휘발성 메모리로 방출하도록 명령하는 것을 특징으로 하는 인메모리 데이터베이스.

【청구항 13】

제12항에 있어서,

(i) 상기 프로세서는 상기 비휘발성 메모리의 상기 주소 리스트 중에서 희생 노드를 선택하고, (ii) 상기 프로세서는 상기 희생 노드의 키-값을 상기 휘발성 메모리에 복사하고, (iii) 상기 휘발성 메모리의 데이터 구조는 상기 휘발성 메모리에 복사된 키-값을 참조하고, (iv) 상기 휘발성 메모리의 로그 버퍼는 상기 희생 노드의 키-값의 처리 명령에 관한 로그를 기록하고, (v) 상기 로그 버퍼는 상기 로그를 블록 디바이스의 로그 파일로 플러싱하고, (vi) 플러싱을 완료하면 상기 프로세서는 상기 비휘발성 메모리의 상기 주소 리스트에서 상기 희생 노드를 삭제하도록 명령하는 것을 특징으로 하는 인메모리 데이터베이스.

【청구항 14】

제13항에 있어서,

상기 희생 노드를 선택하여 상기 데이터를 방출하는 순서는 최저 사용 빈도

(Least Recently Used, LRU) 정책에 따라 결정되는 것을 특징으로 하는 인메모리 데이터베이스.

【청구항 15】

제10항에 있어서,

상기 휘발성 메모리에 저장된 데이터가 갱신되면 상기 갱신된 데이터는 상기 비휘발성 메모리에 저장되고,

상기 비휘발성 메모리에 저장된 데이터가 갱신되면, (i) 상기 주소 리스트의 헤드에 신규 노드를 삽입하고, (ii) 상기 신규 노드가 상기 갱신된 데이터를 참조하고, (iii) 갱신되기 전의 데이터를 가진 노드를 삭제하여, 상기 주소 리스트를 재정렬하는 것을 특징으로 하는 인메모리 데이터베이스.

【청구항 16】

제10항에 있어서,

상기 프로세서는 상기 주소 리스트 및 로그 파일을 이용하여 데이터를 복구하며,

상기 프로세서는 (i) 상기 블록 디바이스의 로그 파일을 기반으로 상기 휘발성 메모리의 데이터 구조를 복구하고, (ii) 상기 주소 리스트에서 삭제되지 않은 희생 노드의 키-값을 상기 휘발성 메모리에 복구하고, (iii) 상기 주소 리스트로부터 상기 비휘발성 메모리의 키-값을 복구하는 것을 특징으로 하는 인메모리 데이터베이스.

【청구항 17】

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,

상기 인메모리 데이터베이스의 비휘발성 메모리에 우선적으로 데이터를 저장하는 단계; 및

상기 인메모리 데이터베이스의 비휘발성 메모리에 저장된 데이터의 일부 데이터를 상기 인메모리 데이터베이스의 휘발성 메모리에 백업하는 단계

를 포함한 동작들을 수행하는 컴퓨터 프로그램.

【요약서】

【요약】

본 실시예들은 비휘발성 메모리를 1차 저장소로 사용하고, 비휘발성 메모리의 일정 용량을 초과한 데이터에 관하여 휘발성 메모리를 2차 저장소로 사용하고, 휘발성 메모리에 저장된 데이터에 관한 로그 파일을 블록 디바이스에 주기적으로 저장한 후, 복구함으로써, 비휘발성 메모리의 용량 한계를 극복하고 데이터 지속성을 확보할 수 있는 인메모리 데이터베이스를 제공한다.

【대표도】

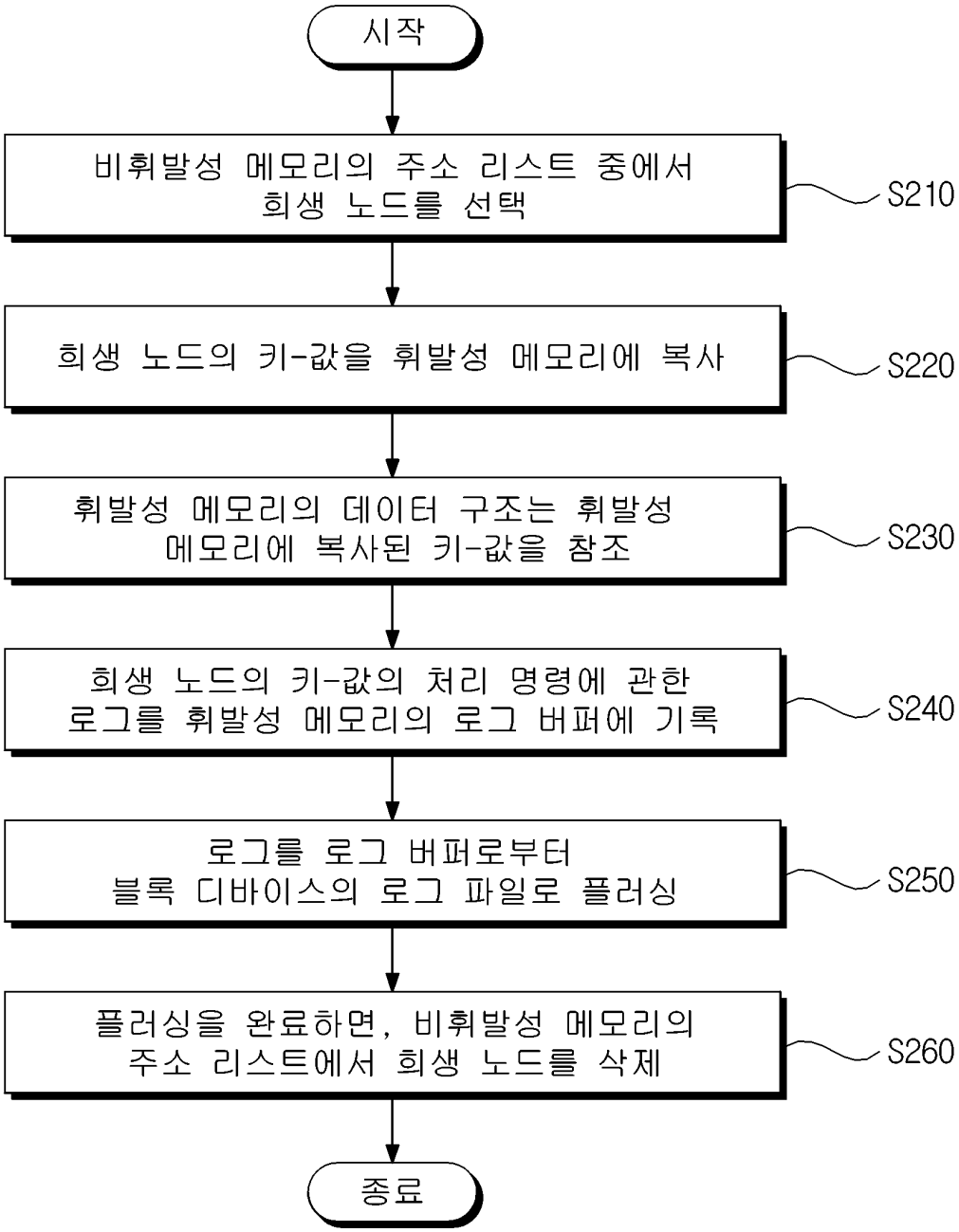
도 1

【도면】

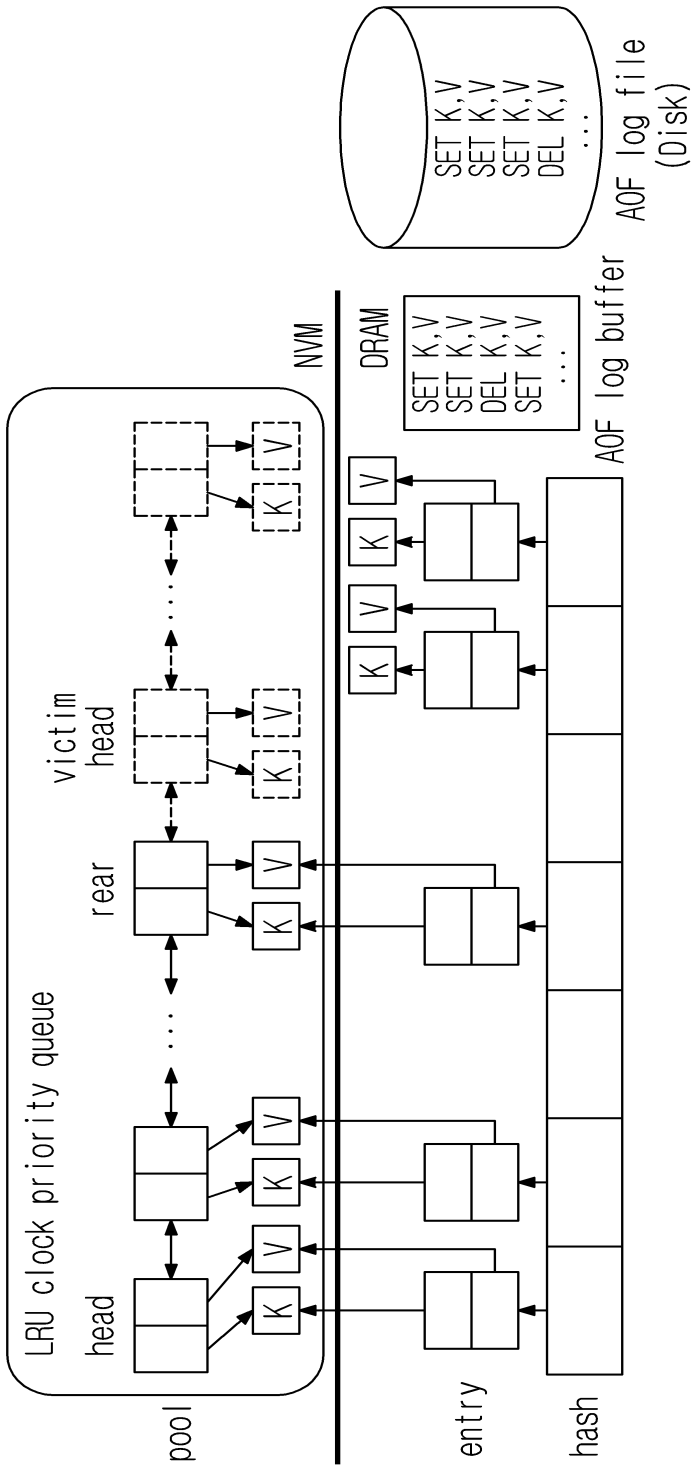
【도 1】


인메모리 데이터베이스
(10)
프로세서
(100)
비 휘발성 메모리
(200)
주소 리스트
(220)
휘발성 메모리
(300)
데이터 구조
(310)
로그 버퍼
(320)
블록 디바이스
(400)

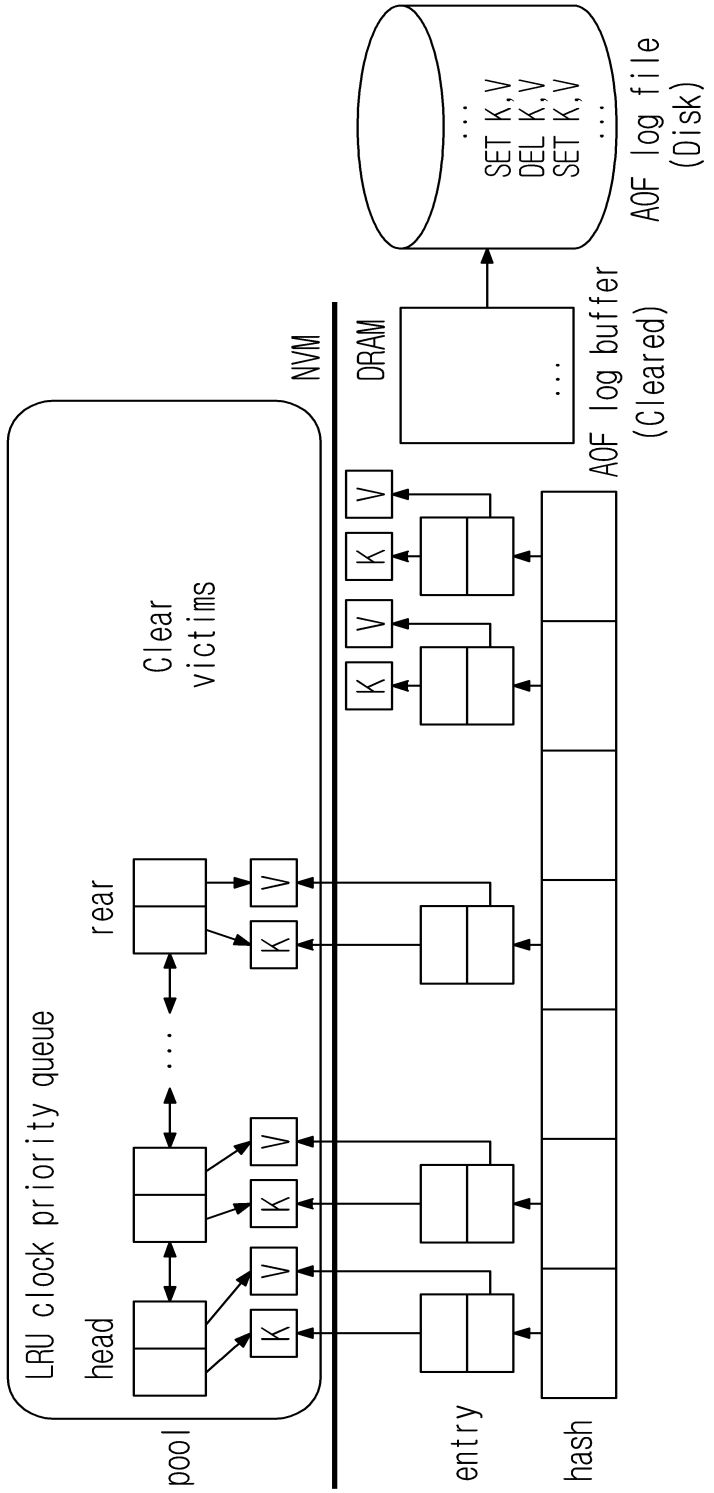
【도 2】


시작
비휘발성 메모리의 주소 리스트 중에서 희생 노드를 선택
S210
희생 노드의 키-값을 휘발성 메모리에 복사
S220
휘발성 메모리의 데이터 구조는 휘발성 메모리에 복사된 키-값을 참조
S230
희생 노드의 키-값의 처리 명령에 관한 로그를 휘발성 메모리의 로그 버퍼에 기록
S240
로그를 로그 버퍼로부터 블록 디바이스의 로그 파일로 플러싱
S250
플러싱을 완료하면, 비휘발성 메모리의 주소 리스트에서 희생 노드를 삭제
S260
종료

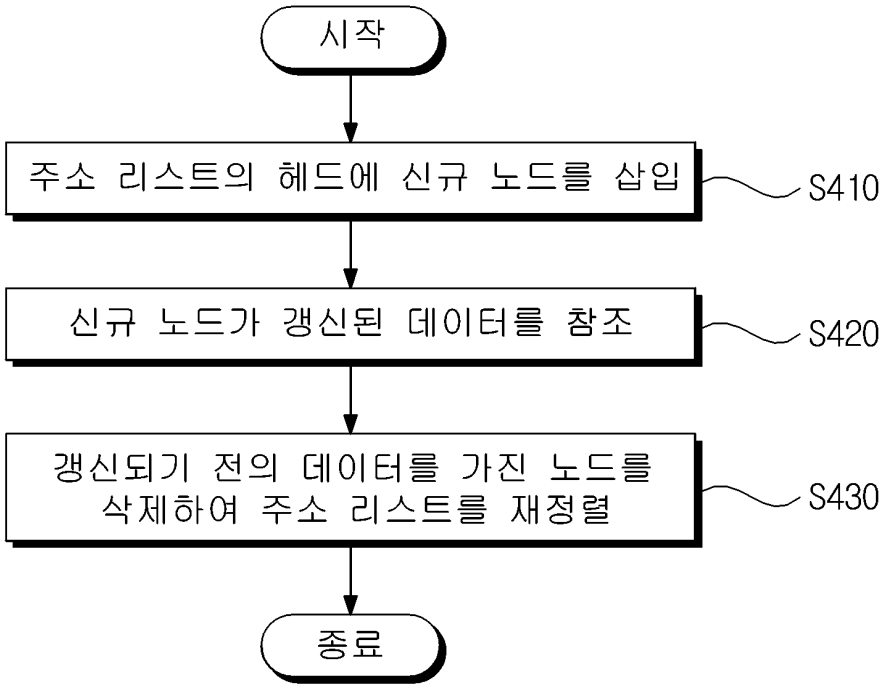
【도 3a】


LRU clock priority queue
head
victim
head
rear
pool
NVM
DRAM
SET K,V
SET K,V
DEL K,V
SET K,V
...
AOF log buffer
entry
hash
SET K,V
SET K,V
SET K,V
DEL K,V
...
AOF log file
(Disk)

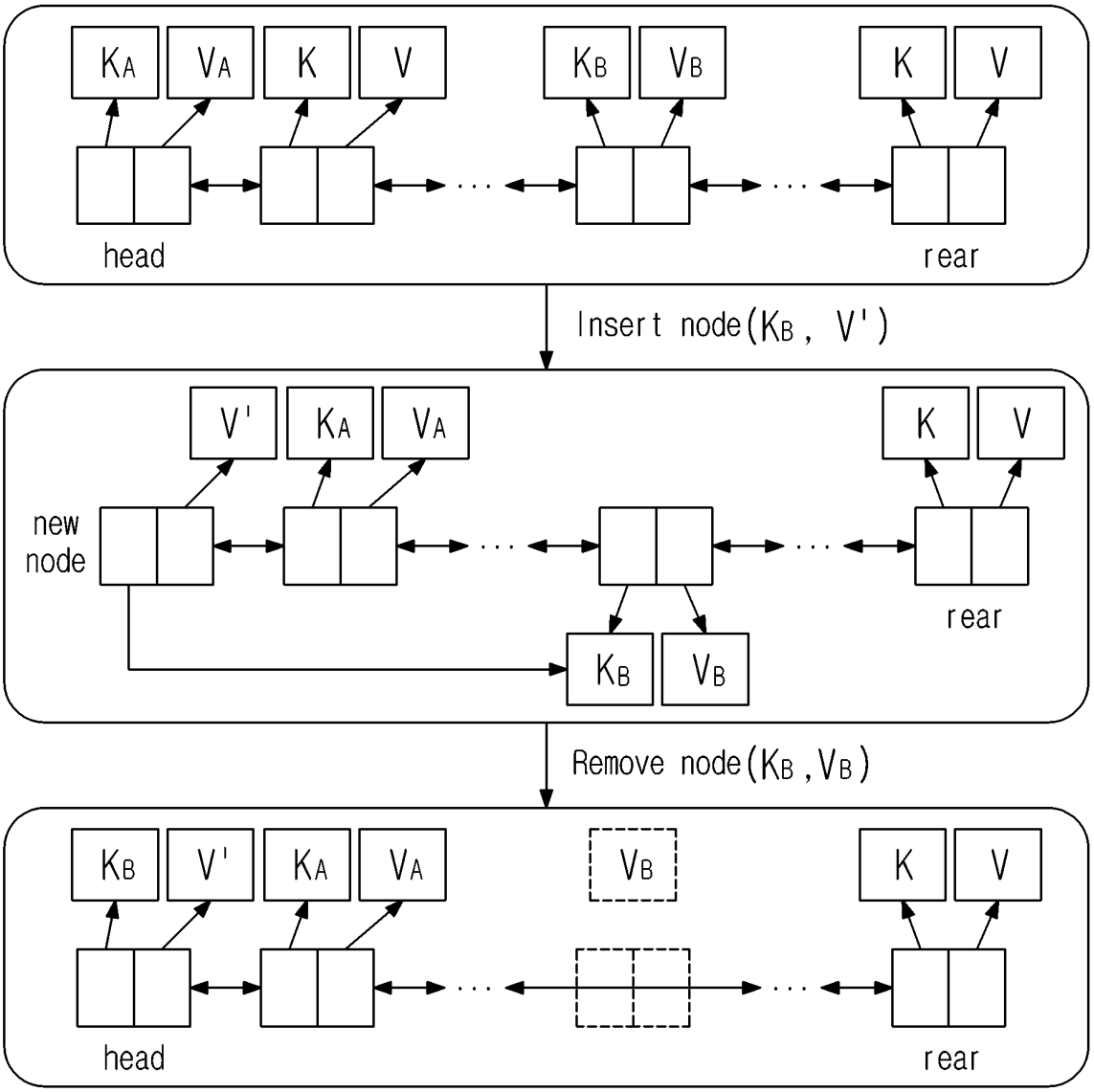
【도 3b】


LRU clock priority queue
head
rear
Clear victims
pool
NVM
DRAM
entry
hash
AOF log buffer (Cleared)
AOF log file (Disk)
SET K,V
DEL K,V
SET K,V
K
V

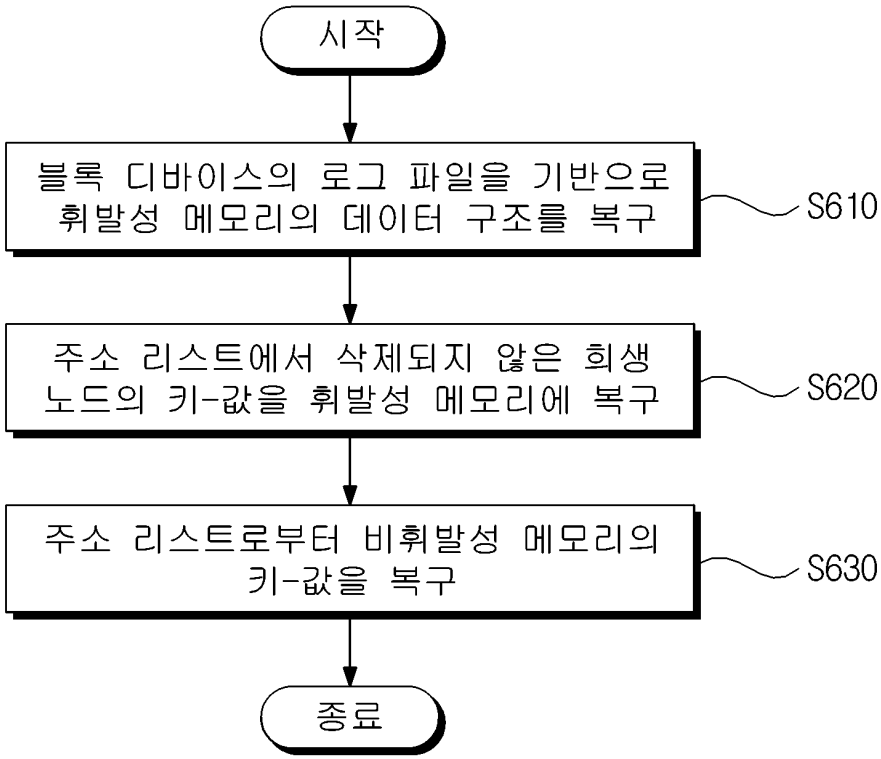
【도 4】


시작
주소 리스트의 헤드에 신규 노드를 삽입
S410
신규 노드가 갱신된 데이터를 참조
S420
갱신되기 전의 데이터를 가진 노드를 삭제하여 주소 리스트를 재정렬
S430
종료

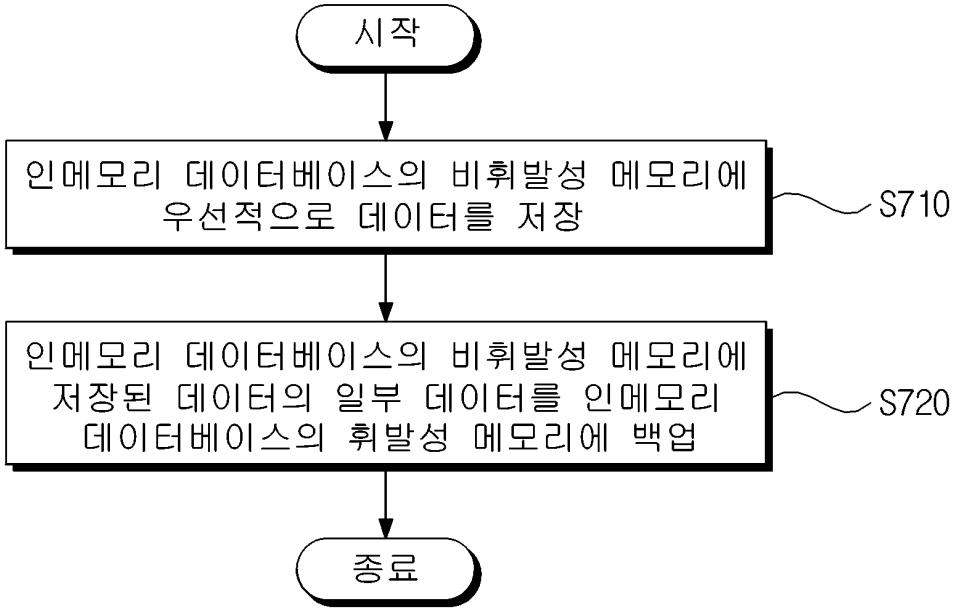
【도 5】


KA
VA
K
V
KB
VB
K
V
head
rear
Insert node(KB, V')
V'
KA
VA
K
V
new
node
rear
KB
VB
Remove node(KB,VB)
KB
V'
KA
VA
VB
K
V
head
rear

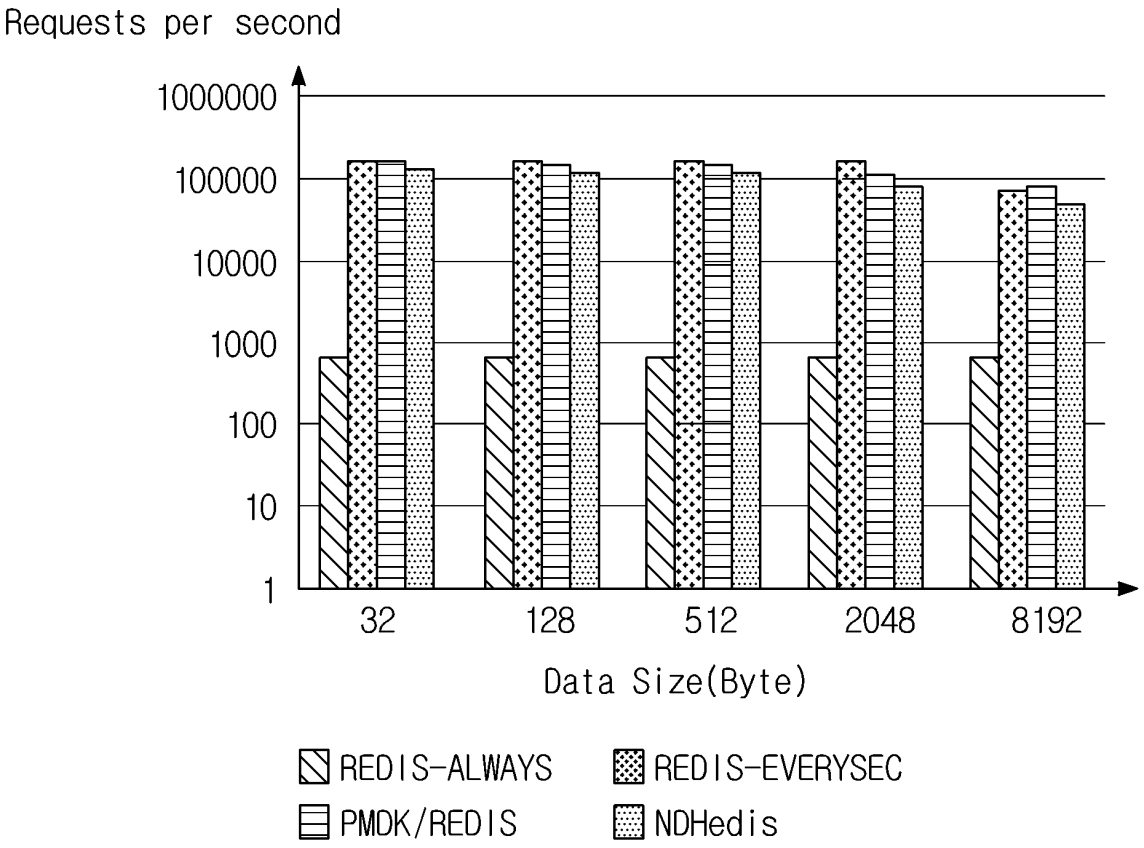
【도 6】


시작
블록 디바이스의 로그 파일을 기반으로 휘발성 메모리의 데이터 구조를 복구
S610
주소 리스트에서 삭제되지 않은 희생 노드의 키-값을 휘발성 메모리에 복구
S620
주소 리스트로부터 비휘발성 메모리의 키-값을 복구
S630
종료

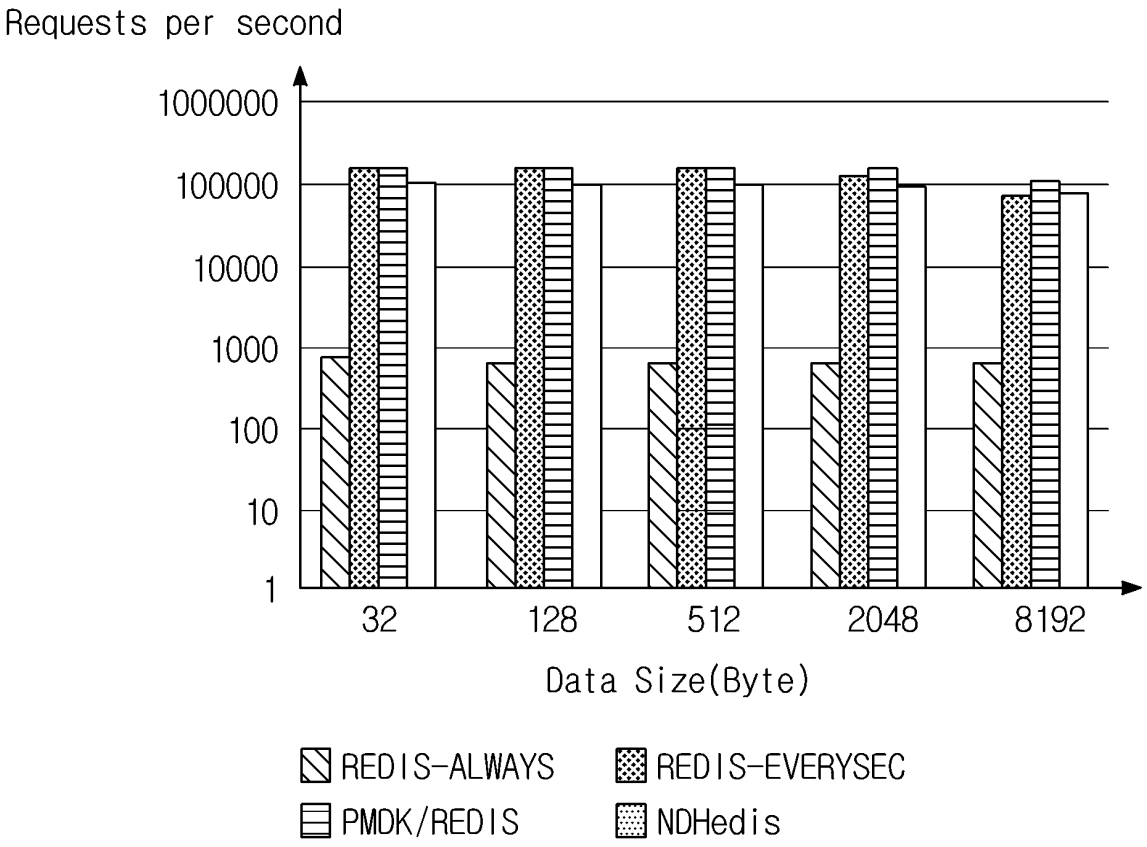
【도 7】


시작
인메모리 데이터베이스의 비휘발성 메모리에 우선적으로 데이터를 저장
S710
인메모리 데이터베이스의 비휘발성 메모리에 저장된 데이터의 일부 데이터를 인메모리 데이터베이스의 휘발성 메모리에 백업
S720
종료

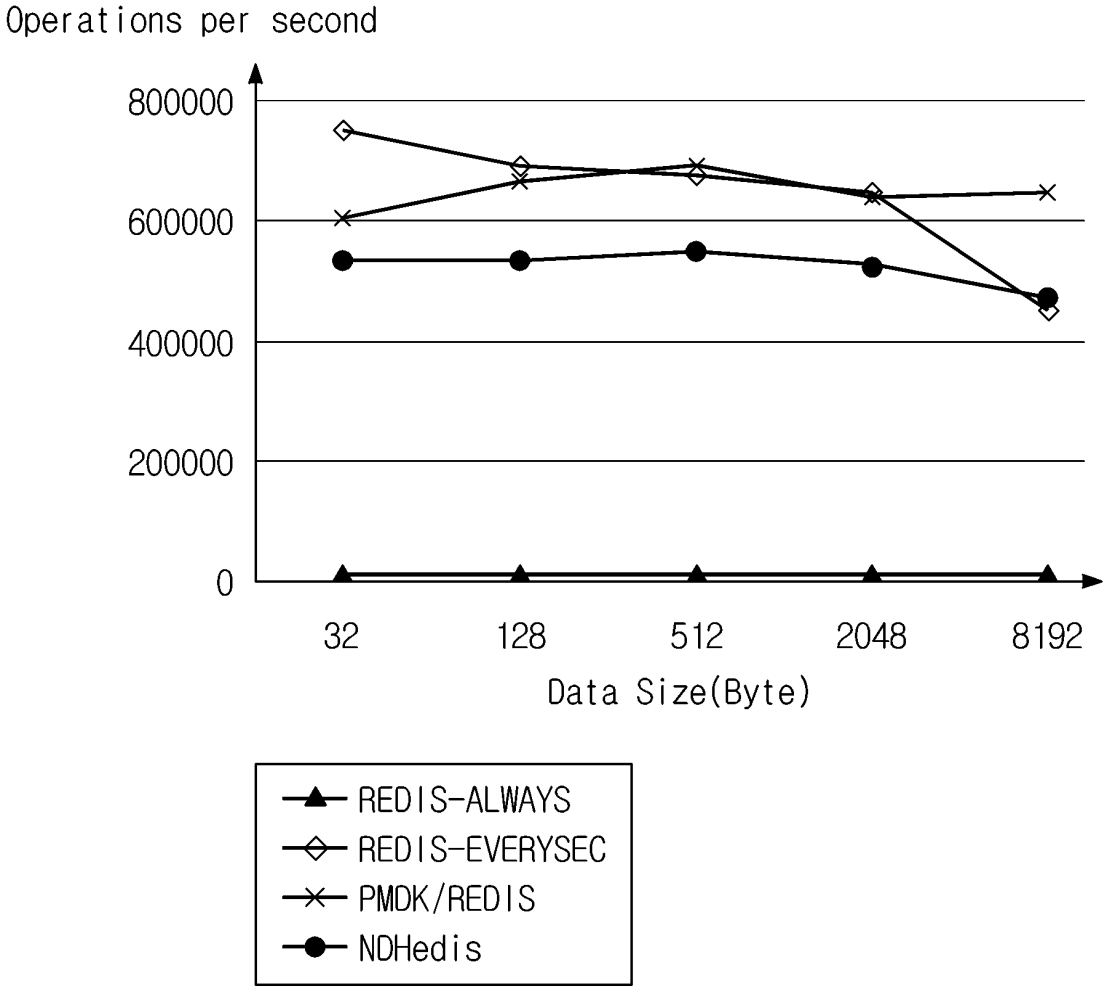
【도 8a】


Requests per second
1000000
100000
10000
1000
100
10
1
32
128
512
2048
8192
Data Size(Byte)
REDIS-ALWAYS
REDIS-EVERYSEC
PMDK/REDIS
NDHedis

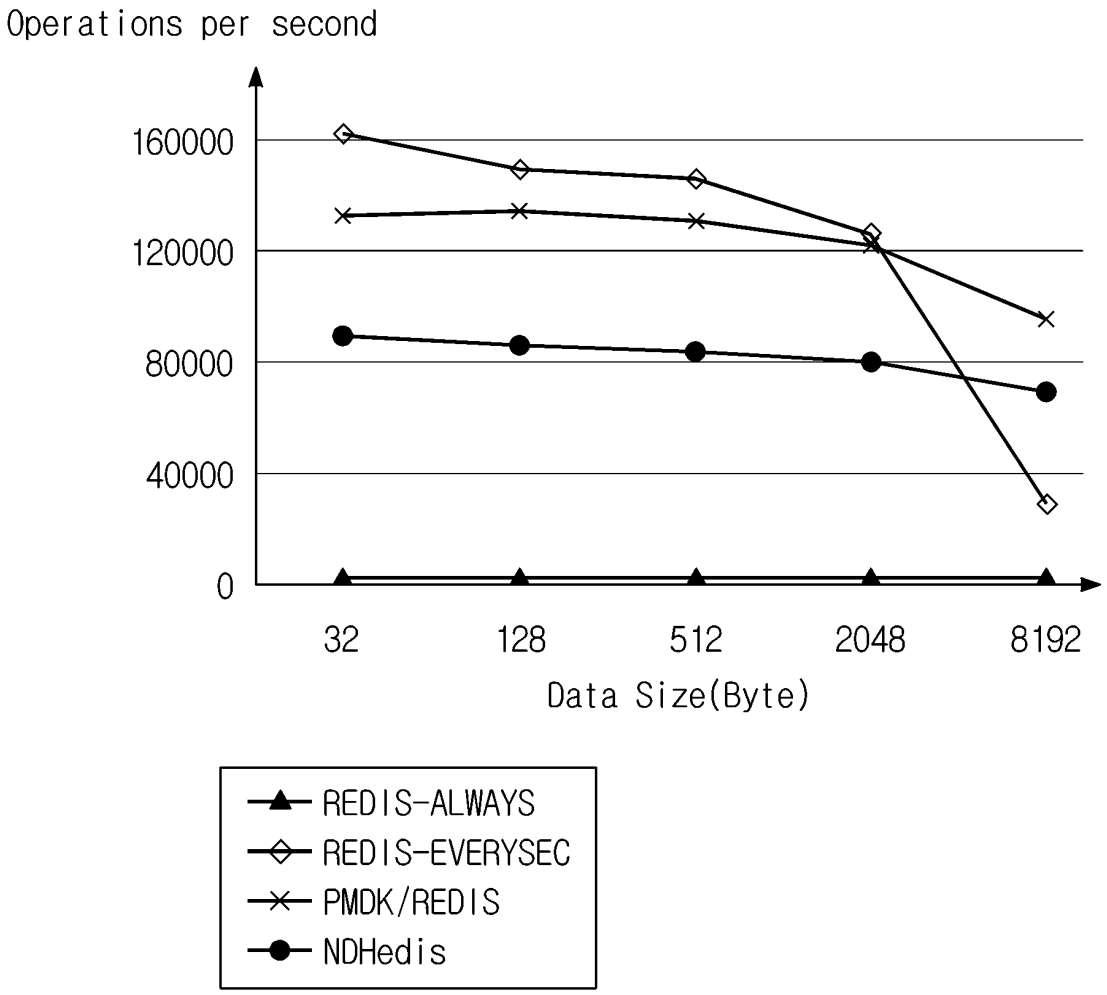
【도 8b】


Requests per second
1000000
100000
10000
1000
100
10
1
32
128
512
2048
8192
Data Size(Byte)
REDIS-ALWAYS
REDIS-EVERYSEC
PMDK/REDIS
NDHedis

【도 8c】


Operations per second
800000
600000
400000
200000
0
32
128
512
2048
8192
Data Size(Byte)
REDIS-ALWAYS
REDIS-EVERYSEC
PMDK/REDIS
NDHedis

【도 8d】


Operations per second
160000
120000
80000
40000
0
32
128
512
2048
8192
Data Size(Byte)
REDIS-ALWAYS
REDIS-EVERYSEC
PMDK/REDIS
NDHedis