PCT 출원서

출력(전자적 형태가 원본)

0	**수리관청 전용**	
0-1	국제출원번호	
0-2	국제출원일자	
0-3	수리관청 명칭 및 "PCT 국제출원"	

0-4	**서식 PCT/RO/101 - PCT 출원서**	
0-4-1	우측에 기재된 바와 같이 작성되었다.	**PCT-SAFE** **버전 3.51.080.256 MT/FOP 20180101/0.20.5.24**
0-5	**신청** 아래 서명인은 본 국제 출원서가 특허협력조약에 의해 처리될 것을 청구합니다.	
0-6	**출원인이 지정한 수리관청**	**대한민국 특허청 (RO/KR)**
0-7	**출원인 또는 대리인의 서류참조기호**	**PCT18-0025**
I	**발명의 명칭**	**플래시 저장장치의 내부 병렬성을 이용하는 키 값 기반의 데이터 액세스 장치 및 방법**
II	**출원인**	
II-1	이 사람은	**오직 출원인 (applicant only)**
II-2	우측 지정국에 관한 출원인	**모든 지정국 (all designated States)**
II-4ko	성명	**연세대학교 산학협력단**
II-4en	Name:	**UNIVERSITY-INDUSTRY FOUNDATION(UIF), YONSEI UNIVERSITY**
II-5ko	주소	**대한민국** **03722** **서울시 서대문구 연세로 50**
II-5en	Address:	**50, Yonsei-ro,** **Seodaemun-gu, Seoul 03722** **Republic of Korea**
II-6	국적	**대한민국 KR**
II-7	거주국	**대한민국 KR**
II-8	전화번호	**82-2-2123-5138**
II-9	팩스번호	**82-2-2013-8618**
II-10	이메일 주소	**patent@yonsei.ac.kr**
II-11	출원인 코드	**2-2005-009509-9**

III-1	출원인 및/또는 발명자	
III-1-1	이 사람은	**오직 발명자 (inventor only)**
III-1-3	우측 지정국에 관한 발명자	**모든 지정국 (all designated States)**
III-1-4ko	성명	**박상현**
III-1-4en	Name (LAST, First):	**PARK, Sang Hyun**
III-1-5ko	주소	**대한민국 08004 서울시 양천구 오목로 300, 204동 3701호**
III-1-5en	Address:	**204-3701, 300, Omok-ro, Yangcheon-gu, Seoul 08004 Republic of Korea**
III-2	출원인 및/또는 발명자	
III-2-1	이 사람은	**오직 발명자 (inventor only)**
III-2-3	우측 지정국에 관한 발명자	**모든 지정국 (all designated States)**
III-2-4ko	성명	**송경태**
III-2-4en	Name (LAST, First):	**SONG, Kyung Tae**
III-2-5ko	주소	**대한민국 13616 경기도 성남시 분당구 정자일로 1, B동 1715호**
III-2-5en	Address:	**B-1715, 1, Jeongjail-ro, Bundang-gu, Seongnam-si, Gyeonggi-do 13616 Republic of Korea**
III-3	출원인 및/또는 발명자	
III-3-1	이 사람은	**오직 발명자 (inventor only)**
III-3-3	우측 지정국에 관한 발명자	**모든 지정국 (all designated States)**
III-3-4ko	성명	**김재형**
III-3-4en	Name (LAST, First):	**KIM, Jae Hyung**
III-3-5ko	주소	**대한민국 22365 인천시 중구 신도시북로 44, 612동 201호**
III-3-5en	Address:	**612-201, 44, Sindosibuk-ro, Jung-gu, Incheon 22365 Republic of Korea**

IV-1	**대리인 또는 대표자** 아래에 기재된 자는 관할 국제기관에 대하여 우측에 표시된 자격으로 출원인을 대리하는 것으로 선임되었다.	대리인
IV-1-1ko	성명	특허법인 우인
IV-1-1en	Name:	**WOOIN PATENT & LAW FIRM**
IV-1-2ko	주소	대한민국 **06246** 서울시 강남구 역삼로 **157, 2**층 **(**역삼동**,** 중평빌딩**)**
IV-1-2en	Address:	**(Yeoksam-dong, Jungpyeong Bldg.) 2Fl., 157 Yeoksam-ro,** **Gangnam-gu, Seoul 06246** **Republic of Korea**
IV-1-3	전화번호	**82-2-541-9841**
IV-1-4	팩스번호	**82-2-541-9842**
IV-1-5	이메일 주소	**patent@wooinlaw.com**
IV-1-5(a)	이메일 사용동의 수리관청, 국제조사기관, 국제사무국, 국제예비심사기관이 필요 시 이 이메일 주소를 사용하여 이 국제 출원과 관련하여 발행된 통지서를 송부할 것에 동의한다.	서면 통지서에 앞서 선람용 사본 송부
IV-1-6	대리인 코드	**9-2006-100082-1**
V	**지정국**	
V-1	**본 출원서의 제출로, 규칙 4.9(a)에 따라, 부여될 수 있는 모든 종류의 권리 보호를 위하여, 그리고 해당하는 경우 지역특허 및 국내특허 모두를 위하여 당해 국제출원일에 PCT에 기속되는 모든 체약국이 지정된다.**	
V-2	**V-2란은 출원서 제출시 또는 규칙 26의2.1에 의해 그 이후 출원서 제6기재란에 위 특정 관련 체약국의 국내 선출원에 대한 우선권주장이 포함되어 있을 경우 당해 체약국의 국내법에 의해 해당 국내 선출원의 효력이 상실되는 것을 방지하기 위한 목적으로 당해 체약국의 지정을 제외하는 데에만 사용될 수 있다 (지정 제외시 이의 취소 불가능).**	**KR**
VI-1	**선국내출원에 대한 우선권 주장**	
VI-1-1	출원일	**2018**년 **02**월 **09**일 **(09.02.2018)**
VI-1-2	출원번호	**10-2018-0016169**
VI-1-3	파리협약 당사국명 또는 WTO 회원국명	**KR**
VI-2	**우선권서류 신청** 수리관청에 대하여 위에 명시된 선출원의 인증등본을 준비하여 국제사무국에 송부하여 줄 것을 신청한다.	**VI-1**
VI-3	**인용에 의한 보완** 조약 제11조(1)(iii)(d) 또는 (e)에서 규정하는 국제출원의 요소, 또는 규칙 20.5(a)에서 규정하는 명세서, 청구 범위 또는 도면의 일부가 본 국제출원에는 포함되어 있지 않지만 조약 제11조(1)(iii) 규정의 요소 중 하나 이상이 수리관청에 최초로 접수된 날에 우선권주장의 기초가 된 선출원에 완전히 포함되어 있는 경우, 그 요소 또는 부분은 규칙 20.6 규정에 의한 확인을 조건으로, 규칙 20.6의 규정과 관련하여 본 국제출원에 있어서 인용에 의해 보완된다.	

VII-1	국제조사기관(ISA) 선택	**대한민국 특허청 (ISA/KR)**	
VIII	**선언서**	선언서 개수	
VIII-1	발명자의 신원에 관한 선언	-	
VIII-2	국제출원일에 특허출원 및 특허를 받을 수 있는 출원인의 자격에 관한 선언	-	
VIII-3	국제출원일에 선출원의 우선권을 주장할 수 있는 출원인의 자격에 관한 선언	-	
VIII-4	발명자 선언(미국에 대한 지정의 경우에 한함)	-	
VIII-5	신규성을 해치지 아니하는 개시 또는 신규성 상실의 예외에 관한 선언	-	
IX	**체크 리스트**	용지 수	전자적 파일 첨부
IX-1	출원서(선언서 포함)	**5**	✓
IX-2	명세서	**20**	✓
IX-3	청구범위	**4**	✓
IX-4	요약서	**1**	✓
IX-5	도면	**11**	✓
IX-7	용지매수 소계	**41**	
	첨부 항목	서면 첨부	전자적 파일 첨부
IX-8	수수료 계산 용지	-	✓
IX-9	개별위임장 원본	-	✓
IX-20	**요약서에 수반되어야 할 도면 번호**	**1**	
IX-21	**국제출원의 출원 언어**	한국어	
X-1	**출원인, 대리인 또는 대표자의 서명 또는 날인**		
X-1-1	성명	특허법인 우인	
X-1-2	서명인의 성명	특허법인 우인	
X-1-3	권한 (출원서를 통해 서명자의 자격이 명백하지 않은 경우에는 그 자격도 표시)	특허법인 우인	

출력(전자적 형태가 원본)

수리관청 전용

10-1	국제출원으로 제출된 서류의 실제 접수일	
10-2	도면	
10-2-1	접수	
10-2-2	미접수	
10-3	국제출원으로 제출된 서류를 완성하는 서류 또는 도면의 추후 기간내 제출에 따른 정정된 실제 접수일	
10-4	**PCT** 제**11**조**(2)**에 따라 제출이 요구된 보완서로서 기간내 제출된 보완서의 접수일	
10-5	국제조사기관**(ISA)**	**ISA/KR**
10-6	조사료 납부시까지 지연된 조사용 사본의 송부	

국제 사무국 전용

11-1	국제 사무국의 기록원본 접수일	

PCT 위임장

출력(전자적 형태가 원본)

0-1	**PCT 위임장 (특허 협력 조약에 의거하여 제출된 국제 출원) (PCT 규칙 제90.4조)**	
0-1-1	우측에 기재된 바와 같이 작성되었다.	**PCT-SAFE** **버전 3.51.080.256 MT/FOP 20180101/0.20.5.24**

1	아래에 서명한 출원인	연세대학교 산학협력단
1-1-1	우측에 기재된 사람을 아래의 자격으로 선임한다.	특허법인 우인 **WOOIN PATENT & LAW FIRM** 대한민국 **06246** 서울시 강남구 역삼로 **157, 2**층 (역삼동, 중평빌딩) **(Yeoksam-dong, Jungpyeong Bldg.) 2Fl., 157 Yeoksam-ro,** **Gangnam-gu, Seoul 06246** **Republic of Korea**
1-2	자격	대리인
1-3	우측 기관에 대하여	모든 관할 국제 기관
1-4	아래의 국제 출원에 관한 서명의 출원인을 대리함	
1-4-1	발명의 명칭	플래시 저장장치의 내부 병렬성을 이용하는 키 값 기반의 데이터 액세스 장치 및 방법
1-4-2	출원인 또는 대리인의 서류참조기호	**PCT18-0025**
1-4-3	국제출원번호(이용 가능한 경우)	
1-4-4	수리관청	대한민국 특허청 **(RO/KR)**
1-5	그리고 아래 서명인을 대신하여 지불하거나 지불받았다.	

PCT 위임장

출력(전자적 형태가 원본)

2-1	출원인 서명	
2-1-1	성명	연세대학교 산학협력단
2-1-2	서명인의 성명	연세대학교 산학협력단
2-1-3	권한 (출원서를 통해 서명자의 자격이 명백하지 않은 경우에는 그 자격도 표시)	연세대학교 산학협력단
3	일자	**2018년 10월 17일 (17.10.2018)**

PCT(부속문서 - 수수료 계산용지)

출력(전자적 형태가 원본)
이 페이지는 국제 출원서의 일부가 아니며 페이지수에 포함되지 않는다

0	**수리관청 전용**			
0-1	국제출원번호			
0-2	수리관청의 우편 소인 일자			

0-4	**Form PCT/RO/101 (부속문서)** **PCT 수수료 계산 용지**			
0-4-1	우측에 기재된 바와 같이 작성되었다.	**PCT-SAFE** **버전 3.51.080.256 MT/FOP 20180101/0.20.5.24**		
0-9	**출원인 또는 대리인의 서류참조기호**	**PCT18-0025**		
2	**출원인**	연세대학교 산학협력단		
12	**규정 수수료 계산**	수수료 금액/계수	총 금액 (CHF)	총 금액 (KRW)
12-1	송달료 **T**	➪		**45000**
12-2-1	조사료 **S**	➪		**450000**
12-2-2	국제조사기관	**KR**		
12-3	국제 출원 수수료 최초 30장 **i1**	**1330 CHF**		
12-4	최초 30장 초과 장수	**11**		
12-5	최초 30장 초과 1장당 추가 수수료 **(X)**	**15 CHF**		
12-6	총 추가금액 **i2**	**165 CHF**		
12-7	i1 + i2 = **i**	**1495 CHF**		
12-12	XML 전자출원 감면 **R**	**CHF-300**		
12-13	총 국제출원 수수료(i-R) **I**	➪	**1195**	
12-14	우선권 서류에 대한 수수료 우선권 서류를 요청한 개수	**1**		
12-15	문서별 수수료 **(X)**	**0 KRW**		
12-16	총 우선권 서류 수수료 **P**	➪		
12-17	우선권 주장 회복에 대한 수수료 **RP** 우선권 주장 회복에 대한 요청 개수	**0**		
	우선권 회복에 대한 수수료 총 금액			
12-19	**총 금액 (T+S+I+P+RP)**	➪	**1195**	**495000**
12-21	**결제 방법**	현금		

PCT

13-2-2	검증 메시지 지정국	녹색입니까? 다음의 국가가 지정되지 않았음을 알립니다: **KR**
13-2-3	검증 메시지 이름	녹색입니까? 출원인 **1:** 전자메일 사용승인 항목이 체크되지 않았습니다. 이 전자메일 주소는 전화로 할 수 있는 종류의 연락을 대신하기 위해서만 사용됩니다.
13-2-7	검증 메시지 내용	노란색 처리된**(rendering)** 출원서 본체에 **300dpi** 이상의 이미지들이 포함되어 있습니다. 국제공개시 품질저하를 피하기 위해서는 좀 더 낮은 해상도로 이미지들을 저장하시기 바랍니다.
13-2-8	검증 메시지 수수료	녹색입니까? 가장 최근의 수수료표가 사용되었는지 확인하십시오.

명세서

발명의 명칭: 플래시 저장장치의 내부 병렬성을 이용하는 키 값 기반의 데이터 액세스 장치 및 방법

기술분야

[1] 본 발명은 데이터 액세스 장치 및 방법에 관한 것이다. 보다 상세하게는 플래시 메모리 기반 저장 장치에서 데이터를 액세스 하는 장치에 관한 것이다. 또한, 본 발명은 2018년도 과학기술정보통신부(정부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구사업((SW스타랩)IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발)의 결과물에 해당한다(과제고유번호: 1711065240, 세부과제번호: 2017-0-00477-002).

배경기술

[2] 클라우드 컴퓨팅 및 복수의 데이터 센터들에 의하여 저장되고 처리되는 거대한 양의 빅 데이터를 처리하는 기술이 활발하게 연구되고 있다. 빅 데이터를 처리함에 있어서, 빠른 데이터의 검색, 빠른 웹서비스들, NoSQL 데이터 베이스들 기타 키-벨류(Key-Value) 저장소들은 그 연구 분야의 중심에 있다.

[3] 예를 들어, 페이스북 및 기타 SNS 제공자들은 빅 데이터를 이용하여 서비스를 효과적으로 제공하기 위하여 운영하는 자사의 데이터베이스에 저장된 데이터들에 빠르게 액세스 하기 위한 기술 및 자사의 데이터베이스에 저장된 데이터들을 효율적으로 저장하고 관리하기 위한 기술들을 활용한다.

[4] 키-벨류 스토어(Key-Value Store)는 NoSQL 데이터 베이스의 일종으로, Dynamo, Level DB, Rocks DB, Cassandra, HB BASE 등이 있다. 키 벨류 스토어는 관계형 데이터베이스보다 빅 데이터를 처리함에 있어 데이터를 더 유연하고 효율적으로 관리할 수 있는데, 키-벨류 스토어는 향상된 데이터 관리 성능을 위하여 메인 스토리지로 솔리드 스테이트 드라이버(Solid State Driver, SSD)를 사용하는 경우가 많다. 키-벨류 스토어 중 대표적인 예로, 오픈 소스 키-벨류 기반의 데이터베이스인 페이스 북의 RocksDB는 Get method의 반복적인 호출로 인한 오버헤드를 제거하기 위한 Multi get Method를 제공한다. 이를 이용하여 Rocks DB는 데이터 액세스 속도를 향상시킬 수 있다.

[5] 하지만, 종래의 데이터 액세스 방법은 내부적으로 get Method를 단순히 반복하는데 그치고, 낸드 플래시 메모리 기반의 SSD의 내부 병렬성을 완전히 이용하지 못하여 데이터 I/O 성능 향상에 한계가 있었다.

[6] 플래시 메모리 기반 SSD(Solid State Drive)는 내부에 복수의 플래시 메모리 칩의 병렬성(Parallel)으로 인하여 데이터를 병렬적으로 처리하는 특성이 있는데, 이를 이용하여 키 벨류 스토어에서 데이터 액세스 기술 개발이 요구되고 있다.

[7] (특허문헌 1) 한국 공개 특허 제10-2010-0121389호 (2010.11.17. 공개)

발명의 상세한 설명

기술적 과제

[8] 본 발명은 상기한 문제점을 해결하기 위하여 안출된 것으로서, 키 값 기반의 데이터 액세스 장치를 개시한다. 특히, 플래시 메모리에 저장된 데이터를 액세스 하는 장치를 개시한다.

[9] 그러나 본 발명의 목적은 상기에 언급된 사항으로 제한되지 않으며, 언급되지 않은 또 다른 목적들은 아래의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

과제 해결 수단

[10] 본 발명은 상기한 목적을 달성하기 위해 안출된 것으로서, 사용자의 입력에 기초하여 적어도 하나 이상의 키 값들을 포함하는 키 리스트를 입력 받고, 상기 키 리스트에 따라 액세스된 데이터를 출력하는 서버; 상기 데이터를 저장하기 위한 복수의 비휘발성 메모리 셀들로 구분되는 저장영역을 포함하는 저장부; 및 상기 메모리 셀들에 저장된 데이터의 위치 정보를 병렬적으로 요청하여 수신하고, 상기 수신된 위치 정보로부터 저장 데이터 리스트를 생성하며, 상기 생성된 저장 데이터 리스트와 상기 키 리스트를 비교하여 상기 메모리 셀들에 저장된 데이터를 액세스 하는 제어부;를 포함하는 것을 특징으로 하는 키 값 기반의 데이터 액세스 장치를 제안한다.

[11] 본 발명에서 상기 서버는 상기 키 값들을 기반으로 동작하는 적어도 하나의 인터페이스(Key Value Operations Interface)를 이용하여 상기 키 리스트를 입력 받을 수 있다.

[12] 본 발명에서 상기 저장부는 상기 데이터를 계층적으로 처리하는 로그 구조 병합 트리(Log Structure Merged Tree) 방식으로 상기 메모리 셀들에 저장하고, 상기 메모리 셀들에 저장된 데이터들은 적어도 하나의 계층들로 구분되어 상기 계층들 간의 압축(compaction)을 통하여 관리될 수 있다.

[13] 본 발명에서 상기 제어부는 상기 위치 정보를 이용하여 상기 로그 구조 병합 트리 방식으로 상기 메모리 셀들에 저장된 데이터의 하위 계층으로부터 순차적으로 상기 저장 데이터 리스트를 생성하는 저장 데이터 리스트 생성부; 를 더 포함하고, 상기 생성된 저장 데이터 리스트를 이용하여 상기 메모리 셀들에 저장된 데이터를 액세스할 수 있다.

[14] 본 발명에서 상기 제어부는 상기 메모리 셀들에 저장된 데이터의 위치 정보를 오퍼레이션 변수를 기반으로 동작하는 병렬 동기 I/O 인터페이스(Parallel Synchronous I/O Interface)를 이용하여 일괄 수신하고, 상기 오퍼레이션 변수는 버퍼, I/O 파라미터 및 상기 키 값들에 연관된 포인터 세트 중 적어도 하나를 포함할 수 있다.

[15] 본 발명에서 상기 병렬 동기 I/O 인터페이스를 이용하여 일괄 수신된 위치 정보는 데이터의 입출력을 요청하는 I/O 요청 셋을 포함하고, 상기 I/O 요청 셋은

상기 메모리 셀들에 저장된 데이터의 위치에 따라 순차로 정렬될 수 있다.

[16] 본 발명에서 상기 압축(compaction)은 상기 적어도 하나의 계층들에서 병렬로 수행되고, 상기 압축은 스레드 수를 고려하여 주기적으로 수행되며, 상기 저장부는 복수의 키-벨류(Key-Value) 쌍을 이용하여 상기 데이터를 로그 구조 병합 트리(Log Structure Merged Tree) 방식으로 저장할 수 있다.

[17] 본 발명에서 상기 제어부는 상기 생성된 저장 데이터 리스트와 상기 키 리스트를 상기 데이터의 하위 계층으로부터 순차적으로 비교하여 상기 키 리스트에 대응되는 상기 위치 정보가 나타내는 위치에 저장된 데이터를 반환 데이터 리스트에 저장하는 반환 데이터 리스트 저장부; 를 더 포함하고, 상기 데이터의 최상위 계층까지 비교 후 상기 데이터가 저장된 반환 데이터 리스트를 이용하여 상기 메모리 셀들에 저장된 데이터에 액세스할 수 있다.

[18] 본 발명에서 상기 메모리 셀들은 상기 데이터의 전송을 위한 멀티 채널이 마련된 낸드(nand)형 플래시 메모리를 포함하고, 상기 낸드형 플래시 메모리는 상기 키 값들의 크기에 따른 키 범위(Key Range)가 미리 할당되어 상기 데이터를 구분하여 저장하는 복수의 블록을 포함할 수 있다.

[19] 본 발명에서 상기 블록은 상기 데이터가 상기 블록에 포함되었는지 여부를 해쉬값을 출력으로 가지는 미리 설정된 해쉬 함수를 이용하여 판단하는 블룸 필터 파일 및 상기 블룸 필터 파일에서 상기 데이터가 포함된 것으로 판단되는 경우 상기 데이터가 상기 블록의 어느 위치에 있는지에 관한 오프셋 정보를 출력하는 인덱스 파일을 포함하고, 상기 키 리스트는 상기 데이터의 위치를 나타내는 이진 시퀀스로 마련될 수 있다.

[20] 또한 본 발명은 사용자의 입력에 기초하여 적어도 하나 이상의 키 값들을 포함하는 키 리스트를 입력 받는 단계; 데이터를 저장하기 위한 복수의 비휘발성 메모리 셀들에 저장된 상기 데이터의 위치 정보를 병렬적으로 요청하는 단계; 상기 위치 정보로부터 상기 메모리 셀들에 저장된 데이터 목록을 나타내는 저장 데이터 리스트를 생성하는 단계; 및 상기 생성된 저장 데이터 리스트와 상기 키 리스트를 비교하여 상기 메모리 셀들에 저장된 데이터를 액세스 하는 단계;를 포함하는 것을 특징으로 하는 키 값 기반의 데이터 액세스 방법을 제안한다.

[21] 본 발명에서 상기 키 리스트를 입력 받는 단계는 상기 키 값들을 기반으로 동작하는 적어도 하나의 인터페이스(Key Value Operations Interface)를 이용하여 상기 키 리스트를 입력 받고, 상기 메모리 셀들은 상기 데이터를 계층적으로 처리하는 로그 구조 병합 트리(Log Structure Merged Tree) 방식으로 저장하며, 상기 저장된 데이터는 적어도 하나의 계층들로 구분되어 상기 계층들 간의 압축(compaction)을 통하여 관리될 수 있다.

[22] 본 발명에서 상기 생성하는 단계는 상기 위치 정보를 이용하여 상기 로그 구조 병합 트리 방식으로 상기 메모리 셀들에 저장된 데이터의 하위 계층으로부터 순차적으로 상기 저장 데이터 리스트를 생성하고, 상기 메모리 셀들에 저장된 데이터의 위치 정보를 오퍼레이션 변수를 기반으로 동작하는 병렬 동기 I/O

인터페이스(Parallel Synchronous I/O Interface)를 이용하여 일괄 수신하며, 상기 병렬 동기 I/O 인터페이스를 이용하여 일괄 수신된 위치 정보는 데이터의 입출력을 요청하는 I/O 요청 셋을 포함하고, 상기 I/O 요청 셋은 상기 메모리 셀들에 저장된 데이터의 위치에 따라 순차로 정렬되며, 상기 오퍼레이션 변수는 버퍼, I/O 파라미터 및 상기 키 값들에 연관된 포인터 세트 중 적어도 하나를 포함하고, 상기 액세스 하는 단계는 상기 생성된 저장 데이터 리스트와 상기 키 리스트를 상기 데이터의 하위 계층으로부터 순차적으로 비교하는 단계; 및 비교 결과에 따라 키 리스트에 대응되는 상기 위치 정보가 나타내는 위치에 저장된 데이터를 반환 데이터 리스트에 저장하는 단계; 를 더 포함하고, 상기 데이터의 최상위 계층까지 비교 후 상기 데이터가 저장된 반환 데이터 리스트를 이용하여 상기 메모리 셀들에 저장된 데이터에 액세스 할 수 있다.

[23] 본 발명에서 상기 메모리 셀들은 상기 데이터의 전송을 위한 멀티 채널이 마련된 낸드(nand)형 플래시 메모리를 포함하고, 상기 낸드형 플래시 메모리는 상기 키 값들의 크기에 따른 키 범위(Key Range)가 미리 할당되어 상기 데이터를 구분하여 저장하는 복수의 블록을 포함하며, 상기 블록은 상기 데이터가 상기 블록에 포함되었는지 여부를 해쉬값을 출력으로 가지는 미리 설정된 해쉬 함수를 이용하여 판단하는 블룸 필터 파일 및 상기 블룸 필터 파일에서 상기 데이터가 포함된 것으로 판단되는 경우 상기 데이터가 상기 블록의 어느 위치에 있는지에 관한 오프셋 정보를 출력하는 인덱스 파일을 포함하고, 상기 메모리 셀들은 복수의 키-벨류(Key-Value) 쌍을 이용하여 상기 데이터를 로그 구조 병합 트리(Log Structure Merged Tree) 방식으로 저장하며, 상기 키 리스트는 상기 데이터의 위치를 나타내는 이진 시퀀스로 마련될 수 있다.

[24] 또한, 본 발명은 컴퓨터에서 상기한 키 값 기반의 데이터 액세스 방법을 실행시키기 위한 컴퓨터에서 판독 가능한 기록매체에 저장된 컴퓨터 프로그램을 제안한다.

발명의 효과

[25] 본 발명에 따르면, 데이터 액세스 장치의 입출력 성능을 향상 시킬 수 있다.

[26] 특히, 메모리 셀에 저장된 데이터에 효과적으로 액세스 할 수 있는 잇점이 있다.

도면의 간단한 설명

[27] 도 1은 본 발명의 일 실시 예에 따른 키 값 기반의 데이터 액세스 장치의 블록도이다.

[28] 도 2는 도 1의 실시 예에서 저장부에 로그 병합 트리 구조로 저장된 데이터를 나타내는 개념도이다.

[29] 도 3은 도 1의 실시예에서 제어부의 확대 블록도이다.

[30] 도 4는 플래시 메모리 기반 저장 장치에 저장된 데이터의 구조를 나타내는 개념도이다.

[31] 도 5는 종래의 키 값 기반의 데이터 액세스 장치에서 수행되는 데이터 액세스

과정을 나타낸다.

[32] 도 6은 본 발명의 일 실시 예에 따른 키 값 기반의 데이터 액세스 장치에서 수행되는 데이터 액세스 과정을 나타낸다.

[33] 도 7은 멀티겟 수의 변화에 따라 측정된 키 값 기반의 데이터 검색 장치의 성능을 나타낸다.

[34] 도 8은 멀티겟 비율의 변화에 따라 측정된 키 값 기반의 데이터 검색 장치의 성능을 나타낸다.

[35] 도 9는 데이터 블록의 사이즈에 따라 측정된 키 값 기반의 데이터 검색 장치의 성능을 나타낸다.

[36] 도 10은 본 발명의 일 실시 예에 따른 키 값 기반의 데이터 검색 방법의 흐름도를 나타낸다.

[37] 도 11은 도 10의 실시 예에서 액세스 하는 단계의 확대 흐름도이다.

발명의 실시를 위한 형태

[38] 이하, 본 발명의 일 실시예를 첨부된 도면들을 참조하여 상세히 설명한다.

[39] 첨부 도면을 참조하여 설명함에 있어, 동일하거나 대응하는 구성 요소는 동일한 도면번호를 부여하고 이에 대한 중복되는 설명은 생략하기로 한다.

[40] 또한 본 발명을 설명함에 있어, 관련된 공지 구성 또는 기능에 대한 구체적인 설명이 본 발명의 요지를 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명은 생략할 수 있다.

[41] 본 출원에서 사용한 용어는 단지 특정한 실시 예를 설명하기 위해 사용된 것으로, 용어를 한정하려는 의도가 아니니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 이하에서 설명하는 각 단계는 하나 또는 여러 개의 소프트웨어 모듈로도 구비가 되거나 또는 각 기능을 담당하는 하드웨어로도 구현이 가능하며, 소프트웨어와 하드웨어가 복합된 형태로도 가능하다. 각 용어의 구체적인 의미와 예시는 각 도면의 순서에 따라 이하 설명 한다. 이하에서는 본 발명의 실시 예에 따른 키 값 기반의 데이터 액세스 장치(10)의 구성을 관련된 도면을 참조하여 상세히 설명한다.

[42] 도 1은 본 발명의 일 실시 예에 따른 키 값 기반의 데이터 액세스 장치(10)의 블록도이다.

[43] 키 값 기반의 데이터 액세스 장치(10)는 서버(100), 저장부(200) 및 제어부(300)를 포함한다. 예를 들어, 키 값 기반의 데이터 액세스 장치(10)는 키-벨류 스토어(Key-Valute Store)에 저장된 데이터를 액세스하기 위하여 사용될 수 있다. 본 발명에서 언급되는 키-벨류 스토어는 NoSQL 데이터 베이스의 일종으로 빅 데이터 처리를 위한 비 관계형(NonRelational) 데이터 베이스 관리 시스템 중 키 값 또는 포인터 셋을 이용하여 해당 키 값에 대응되는 데이터인 벨류(Value)를 관리하는 시스템이다. 일반적으로 키-벨류 스토어는 테이블-컬럼과 같은 스키마 없이 분산 환경에서 단순 검색 및 추가 작업을 위한

키 값을 최적화하기 때문에, 입력요청에 따른 응답 지연 시간(LATENCY)과 데이터 처리율(Throughput)이 우수한 장점이 있다.

[44] 본 발명의 키 값 기반의 데이터 액세스 장치(10)는 대표적인 키-벨류 스토어로서, 서버 워크로드와 같은 빅 데이터 처리에 적합하고 빠른 플래시 저장 장치에서 높은 성능을 내는 RocksDB 플랫폼에서 구동될 수 있다. RocksDB는 메모리, 플래시, 하드디스크 및 HDFS등 다양한 환경에서 실행이 가능한데, 이는 오픈 소스 프로젝트인 LevelDB를 기반으로 한다. 본 발명에서 언급되는 키-벨류는 순수한 비트 스트림으로서, 본 발명에서 Multi get API(Application Programming Interface)를 통해 호출되고 반환되는 모든 키 벨류는 일관성(Consistent)를 유지한다.

[45] 본 발명의 키 값 기반의 데이터 액세스 장치(10)가 구동되는 RocksDB는 복수의 키-벨류 쌍(Key-Value Pairs)을 처리하기 위한 애플리케이션 프로그래밍 인터페이스(Application Programming Interface, API)로서 Multi-get Method 또는 get Method를 사용한다. Multi-get Method 또는 get Method는 어플리케이션이 RocksDB에서 키를 호출하기 위한 오퍼레이션(Operation)인데, Multi-get은 동시에 복수의 키 값을 호출할 수 있도록 한다. 다만, 종래의 Multi-get은 내부적으로 get method를 반복할 뿐 낸드 플래시 메모리 기반의 SSD의 내부 병렬성(Internal Parallelism)을 제대로 이용하지 못하는 문제점이 있었다.

[46] 본 발명의 키 값 기반의 데이터 액세스 장치(10)가 사용되는 키-벨류 스토어는 종래 관계형 데이터 베이스보다 데이터를 더 유연하고 효율적으로 관리할 수 있고, 보다 향상된 성능을 위하여 메인 스토리지로 SSD를 사용하는 경우가 많다. 키 벨류 스토어는 가능한 쓰기 증폭(Write Amplification)을 줄이면서, SSD의 랜덤 I/O 요청 특성을 이용하여 성능을 향상할 수 있다. 본 발명의 키 값 기반의 데이터 액세스 장치는 복수의(Multiple) I/O 요청을 단일의 시스템 콜(Single System Call)에서 전달(Parallel)할 뿐만 아니라, 복수의 I/O 요청을 정렬하여(Sequential) 이용하기 때문에 복수의 플래시 칩을 포함하는 메모리 셀들에 저장된 데이터에 Parallel Sequential 액세스할 수 있고, 플래시 메모리 기반의 저장 장치가 보다 향상된 I/O 성능을 달성할 수 있게 한다.

[47] 즉 본 발명의 키 값 기반의 데이터 액세스 장치(10)는 PSYNC I/O를 이용하여, 단일의 PSYNC 시스템 콜에서 메모리 셀들에 저장된 데이터에 대하여 복수의 읽기 접근(Multiple Read Accesses)이 가능하고, 종래 키-벨류 데이터베이스에서 저장된 데이터에 액세스하기 위한 패턴을 반복적 랜덤 액세스(Iterative random access)에서 병렬 시퀀셜 액세스(Parallel Sequential Access)로 변경함으로서 플래시 메모리 기반의 저장 장치의 I/O 성능을 향상시킬 수 있다. 결과적으로 키 값 기반의 데이터 액세스 장치(10)를 이용하는 경우, 보다 빠르게 데이터에 액세스 가능할 뿐만 아니라 SSD의 대역폭(Bandwidth)을 향상시킬 수 있다.

[48] 서버(100)는 사용자(20)의 입력에 기초하여 적어도 하나 이상의 키 값들을 포함하는 키 리스트를 입력 받고, 상기 키 리스트에 따라 액세스된 데이터를

출력한다. 예를 들어, 서버(100)는 저장부(200) 및 제어부(300)와 분리되어 외부에 별도로 마련될 수 있지만, 저장부(200)내에 데이터들을 관리하기 위한 플랫폼으로 구현될 수 있다. 서버(100)는 사용자(20)로부터 키 리스트를 입력 받고, 키 리스트에 대응되어 액세스된 데이터들이 저장된 반환 데이터 리스트를 제어부(300)로부터 전송 받아 출력할 수 있다. 본 발명의 서버(100)가 입력 받는 키 리스트는 저장부(200)에 저장된 데이터들의 위치를 나타내는 이진 시퀀스로 마련도리 수 있다.

[49] 예를 들어, 서버(100)는 키 값들을 기반으로 동작하는 적어도 하나의 인터페이스(Key Value Operations Interface)를 이용하여 상기 키 리스트를 입력 받을 수 있다. 서버(100)가 사용하는 키 값들을 기반으로 동작하는 인터페이스는 복수의 키-벨류 쌍(Key-Value Pairs)을 처리하기 위한 애플리케이션 프로그래밍 인터페이스(Application Programming Interface, API)를 포함하고, Multi-get Method 또는 get Method로 마련될 수 있다. 서버(100)는 RocksDB에서 키를 호출하기 위한 오퍼레이션(Operation)인 Multi-get Method 또는 get Method를 사용하여 사용자로부터 키 리스트를 입력 받을 수 있다. 도 2를 참조하여 설명한다.

[50] 저장부(200)는 데이터를 저장하기 위한 복수의 비휘발성 메모리 셀들로 구분되는 저장영역을 포함 한다. 예를 들어, 저장부(200)가 포함하는 메모리 셀들은 데이터 전송을 위한 멀티 채널이 마련된 낸드(nand)형 플래시 메모리를 포함하고, 상기 플래시 메모리는 상기 키 값들의 크기에 따른 키 범위(Key Range)가 미리 할당되어 상기 데이터를 구분하여 저장하는 복수의 블록을 포함할 수 있다. 또한, 상기 블록은 상기 데이터가 블록에 포함되었는지 여부를 해쉬값(Hashvalue)을 출력으로 가지는 미리 설정된 해쉬 함수(Hashfunction)를 이용하여 판단하는 블룸 필터(Bloomfilter) 파일 및 상기 블룸 필터 파일에서 데이터가 포함된 것으로 판단되는 경우 상기 데이터가 상기 블록의 어느 위치에 있는지에 관한 오프셋 정보를 출력하는 인덱스 파일을 포함한다.

[51] 저장부(200)는 복수의 메모리 셀들에 저장된 데이터를 로그 구조 병합 트리 방식(LSM 트리)으로 관리함은 전술한 바와 같다. LSM 트리는 LevelDB, RocksDB 및 Hbase 와 같은 키 벨류 스토어에서 사용되는 베이스 트리(BaseTree) 알고리즘으로서, 쓰기 집중 데이터 워크로드(Write Intensive Data Workload)를 위해 설계된 데이터 구조이며, 낮은 쓰기 레이턴시(Latency)를 유지할 수 있다.

[52] 예를 들어, 데이터가 계층적으로 관리되는 로그 구조 병합 트리(LSM 트리)는 메모리가 가득 차는 경우에 한해 디스크 스토리지(Disk Storage)로 플러싱하기 때문에 낮은 쓰기 레이턴시(Latency)를 달성할 수 있다. LSM 트리는 낮은 쓰기 레이턴시를 위하여, 데이터를 업데이트하거나 삭제하지 않는 대신, 삭제된 데이터 항목을 따로 표지(mark)하는 톰스톤 엔트리(Tombstone entries)를 이용하여 삭제된 데이터에 액세스하는 것을 피할 수 있다.

[53] 본 발명의 저장부(200)에 마련된 메모리 셀들에 저장된 데이터들의 삭제는 LSM 트리의 계층간 합병(Merge)동작에 의하여 수행될 수 있다. LSM 트리는

계층간 트리 성분(TreeComponent)에 저장된 데이터의 크기가 미리 설정된 임계치(threshold)이상인 경우 상위 계층의 트리 성분을 하위 계층의 트리 성분으로 합병(Merge)하여 실제 트리 성분에 저장된 데이터를 삭제할 수 있다.

[54] 일 실시 예에 따른 로그 병합 트리 구조에서 데이터의 이동을 설명하면, 먼저, In-Memory 영역에 속하는 C0 레벨에서 데이터를 입력 받고, C0레벨에 포함된 memtable이 모두 차면 Immutable Memtable을 거쳐 디스크 영역으로 플러싱 된다. 디스크 영역에 포함된 로그 병합 트리 구조 레벨 중에서 C1레벨은 C0레벨에서 플러싱된 데이터를 저장하고, C0레벨에서 저장된 데이터의 사이즈가 미리 설정된 크기 이상인 경우, C1레벨로 트리 성분을 합병(Merge)한다. 마찬가지로, C1레벨에 저장된 데이터의 크기가 미리 설정된 임계치 이상인 경우, C1레벨의 데이터는 C2 레벨로 합병(Merge)될 수 있다.

[55] 전술한 계층간 트리 성분들의 합병(Merge) 동작에서 합병 커서(merge cursor)가 삭제된 데이터 항목을 표지하는 톰스톤 데이터 표지를 인식하는 경우, 데이터의 삭제가 일어날 수 있다. 일반적으로 많은 키 벨류 스토어들이 빠른 쓰기 속도를 달성하기 위하여 LSM 트리를 이용하여 저장된 데이터를 관리하지만, LSM 트리 특성상 다층 계층으로 인한 한계로 인하여 쓰기 레이턴시의 감쇠없이 읽기 성능을 향상시키기 위한 bLSM, LSM-trie 및 SILT와 같은 기술들이 개발되었다. 본 발명에서 전술한 메모리와 디스크 스토리지(Disk Storage)는 저장부(200)에 포함되어 구현될 수 있지만, 메모리는 저장부(200)와 별도로서, 제어부(300)상의 RocksDB 프로세서상에서 구현될 수 있음은 물론이다.

[56] 본 발명의 키 값 기반의 데이터 액세스 장치(10)가 구동되는 RocksDB는 LevelDB에 기초하고, 키-벨류 스토어의 성능을 최적화 하기 위하여 SSD의 특성을 이용한다. 따라서, RocksDB는 플래시 메모리 기반 저장 장치에서 작동되므로 데이터 쓰기 집중에 효율적이다. 또한, RocksDB는 읽기 성능을 향상하기 위하여 블룸 필터 및 전술한 LSM트리에서 계층간 합병(Merge)에 상응하는 병렬 압축(Compaction)을 이용할 수 있다.

[57] 예를 들어, 키 값 기반의 데이터 액세스 장치(10)의 데이터 관리 체계로 사용되는 RocksDB는 FIFO(First in first out) 알고리즘이 적용되는 In-Memory table(224, 226)을 포함하여, 저장부(200)에 저장된 데이터들을 LSM 트리 구조로 유지할 수 있다.

[58] 본 발명에서 RocksDB는 저장부(200)에 LSM 트리방식으로 저장된 데이터 체계를 의미할 수 있지만, 키 값 기반의 데이터 액세스 장치(10)에서 제어부(300)의 기능을 수행하는 RoclsDB 플랫폼을 의미할 수 있다. 먼저, 키 값 기반의 데이터 액세스 장치(10)는 In-Memory 테이블로 데이터를 입력 받는다. In-Memory 테이블은 memtable(224)과 내부에 저장된 데이터가 데이터의 입력에 의해 영향을 받지 않는 immutable memtable(226)을 포함하는데, memtable에 저장된 데이터의 사이즈가 기 설정된 임계치(threshold)를 초과하는 경우 memtable에 저장된 데이터는 데이터의 입출력에 영향 받지 않는 immutable

memtable로 전달된다. 이때 immutable memtable에 저장된 데이터의 사이즈가 미리 설정된 임계치 이상인 경우, 저장부(200)로 플러싱 될 수 있다.

[59] 저장부(200)의 메모리 셀들에 저장된 데이터들은 RocksDB 플랫폼에서 관리될 수 있다. LevelDB 기반의 시스템은 LSM 트리와 같이 데이터를 계층적으로 관리하기 위하여 SST 파일을 사용한다. LevelDB 시스템의 각 레벨(262, 264, 266)은 LSM 트리의 각 트리 성분에 대응되고, 각 레벨의 데이터들은 정렬되어 SST 파일에 저장된다. 본 발명의 저장부(200)의 메모리 셀들은 플래시 기반 메모리 셀들을 포함하고, 따라서, SST 파일을 기반으로 계층적으로 데이터를 관리하는 경우, 효율적인 데이터 쓰기 동작이 가능하다.

[60] LevelDB 기반의 시스템에서 사용하는 SST 파일은 데이터 셋의 최대 최소값을 포함하는 메타 데이터를 포함하고, 메타 데이터는 데이터가 저장된 위치에 관한 정보로서, 블룸 필터 파일과 인덱스 파일을 포함한다. 즉, LevelDB 기반의 시스템은 메모리 셀들에 마련된 특정 블록에 저장된 데이터에 액세스 하기 위하여, 메타 데이터를 인덱스로서 활용하여 액세스 할 수 있다.

[61] 예를 들어, 저장부(200)는 상기 데이터를 계층적으로 처리하는 로그 구조 병합 트리(Log Structure Merged Tree) 방식으로 상기 메모리 셀들에 저장하고, 상기 메모리 셀들에 저장된 데이터들은 적어도 하나의 계층들로 구분되어 상기 계층간 압축(compaction)을 통하여 관리될 수 있다. 키-벨류 스토어에서 압축(compaction)은 상기 적어도 하나의 계층들에서 병렬로 수행되고, 상기 압축은 스레드 수를 고려하여 주기적으로 수행되며, 저장부(200)는 복수의 키-벨류(Key-Value) 쌍을 이용하여 상기 데이터를 로그 구조 병합 트리(Log Structure Merged Tree) 방식으로 저장할 수 있다.

[62] 본 발명에서 저장부(200)가 복수의 페이지가 마련된 블록들을 포함하는 낸드 플래시 칩 기반의 SSD로 구현되는 경우에, 저장부(200)가 제어부(300)의 제어에 따라 데이터에 액세스하기 위한 블록은 블룸 필터 파일과 인덱스 파일을 포함함은 전술한 바와 같다. 예를 들어, 블룸 필터는 통계적 로직을 이용하여 해당 key의 데이터가 SST파일에 저장되어 있는지 여부를 판단하여 결과를 반환한다. 각 SST 파일의 블룸 필터의 데이터는 메모리에 로딩이 되어있는데, SST 파일에 접근하여 디스크 I/O를 발생시키기 전에 먼저 해당 SST파일에 데이터가 있는지 없는지를 판단한다. 예를 들어, 블룸 필터는 미리 설정된 개수의 해쉬 함수들에서 출력되는 해쉬값들을 이용하여 해당 블록에 요청된 데이터가 있는지 여부를 판단할 수 있다.

[63] 예를 들어, 키 값 기반의 데이터 액세스 장치(10)는 블록 내의 SST 파일에 데이터가 있는 것으로 판단이 되면, 인덱스 파일을 검색한다. 인덱스 파일은 해당 키 값에 해당하는 데이터가 블록 내의 어느 위치에 있는지에 대한 위치 정보(파일 포인터, 오프셋 정보)를 가지고 있다. 즉 저장부(200)의 각 메모리 셀들에 포함된 복수의 블록은 블룸 필터를 포함하고, 제어부(300)로 하여금 각 블록에 요청된 데이터가 있는지 여부를 판단하게 하며, 데이터가 있는 것으로

판단이 되면, 인덱스 파일을 포함하여 제어부(300)로 하여금 블록 내 어느 위치에 데이터가 있는지를 검색하게 할 수 있다.

[64] 저장부(200)내 마련되어 데이터들이 저장된 블록의 SST 파일은 점차 사이즈가 커질 수 있고, 이러한 경우 키 값 기반의 데이터 액세스 장치(10)는 전술한 압축(Compaction)을 이용하여 목표 레벨의 SST 파일을 다음 레벨의 SST 파일(예를 들어, 262에서 264 또는 264에서 266)에 통합하고, 이러한 방식으로 저장부(200)에 저장된 데이터 들의 수와 크기를 조절할 수 있다. 키 값 기반의 데이터 액세스 장치(10)는 전술한 압축(Compaction)을 기반으로 RocksDB 가 데이터에 더욱 빨리 접근할 수 있도록 한다.

[65] 특히, 본 발명에서 압축(Compaction)은 major compaction과 major compaction을 복수의 sub-compaction으로 나누어 병렬 처리될 수 있고, 이때 Sub-compaction에서, 압축 범위는 스레드 수로 나누어지고, 각 스레드는 스몰 compaction을 할당된 키-벨류 쌍(KeyValuePairs)로 처리할 수 있다. 저장부(200)에 저장된 데이터를 관리하는 체계로서, RocksDB 는 멀티-스레드된 압축(compaction)과 블록(Block) 기반 데이터 액세스를 통하여 다른 키-벨류 저장 장치보다 뛰어난 성능을 달성할 수 있다. 다만, 전술한 특징에도 불구하고 종래의 RocksDB의 Multi-get은 SSD의 내부 병렬성을 완전히 이용하지 못하는 한계가 있었음은 전술한 바와 같다. 도 3 및 도 4를 참조하여 설명한다.

[66] 제어부(300)는 저장 데이터 리스트 생성부(320) 및 반환 데이터 리스트 저장부(340)를 포함한다. 예를 들어, 제어부(300)는 메모리 셀들에 저장된 데이터의 위치 정보를 병렬적으로 요청하여 수신하고, 상기 수신된 위치 정보로부터 저장 데이터 리스트를 생성하며, 상기 생성된 저장 데이터 리스트와 상기 키 리스트를 비교하여 상기 메모리 셀들에 저장된 데이터를 액세스한다.

[67] 예를 들어, 제어부(300)가 사용자로부터 입력받은 키 리스트를 이용하여 액세스 하고자 하는 데이터가 저장된 저장부(200)는 플래시 메모리 기반 저장 장치로서 SSD로 마련될 수 있다. SSD는 복수의 플레인을 포함하는 플래시 칩들, 프로세서 및 램이 마련된 SSD 컨트롤러를 포함한다. 도 3에 도시된 SSD의 구조에서 SSD 컨트롤러는 본 발명의 제어부(300)에, 복수의 플래시 칩들이 마련된 영역은 본 발명의 저장부(200)로 마련될 수 있지만, SSD 컨트롤러는 저장부(200)에 통합되어 구현될 수 있음은 물론이다.

[68] 제어부(300)는 통하여 멀티 I/O REQUEST를 저장부(200)에 포함된 복수의 메모리 셀들(플래시 칩, 272 274)으로 전송하고, 메모리 셀들은 멀티플 채널을 이용하여 제어부(300)로 데이터를 전송할 수 있다. 제어부(300)가 액세스 하는 데이터가 저장된 저장부(200)의 일 실시 예로 SSD는 페이지 단위 블록으로 데이터를 저장하며, 각각의 페이지 사이즈는 4~8KB로 마련될 수 있고, 복수의 페이지들은 하나의 블록을 형성한다. 블록 기반의 구조를 가지는 SSD는 페이지 단위로 데이터를 처리할 수 있고, 삭제는 페이지 단위가 아니라 블록단위로만 가능하다. 전술한 SSD의 내부 병렬성으로 인하여 제어부(300)가 저장부(200)에

페이지 기반 데이터 액세스를 시도할 때 시퀀셜(Sequential) 접근이 효율적이다.

[69] 본 발명에서 제어부(300)는 메모리 셀들에 저장된 데이터의 위치 정보를 오퍼레이션 변수를 기반으로 동작하는 병렬 동기 I/O 인터페이스(Parallel Synchronous I/O Interface, PSYNC I/O)를 이용하여 일괄 수신한다. 예를 들어, 제어부(300)가 이용하는 오퍼레이션 변수는 버퍼, I/O 파라미터 및 상기 키 값들에 연관된 포인터 세트 중 적어도 하나를 포함할 수 있다. 또한, 상기 병렬 동기 I/O 인터페이스를 이용하여 일괄 수신된 위치 정보는 데이터의 입출력을 요청하는 I/O 요청 셋을 포함하고, 상기 I/O 요청 셋은 상기 메모리 셀들에 저장된 데이터의 위치에 따라 순차로 정렬될 수 있다. 도 5를 참조하여 설명한다.

[70] 종래, 키 값 기반의 데이터 액세스 장치가 사용되는 RocksDB에서 SSD의 내부 병렬성을 이용하기 위해, PSYNC I/O인터페이스를 이용하여 단일의 시스템 콜에서 멀티 I/O요청(I/O Set)을 전송하는 기술들이 연구되었으나, 종래의 기술은 복수의(Multiple) I/O요청을 PSYNC I/O를 이용하여 멀티로 전달할 뿐, SSD의 페이지 기반의 블록들에 데이터 액세스를 시도함에 있어서 랜덤하게 액세스를 시도하였다. 다만 저장부(200)가 플래시 메모리 기반 저장 장치인 경우, 페이지 기반 블록 구조로 데이터를 저장하는 특성상, 시퀀셜 접근이 더 효율적이기 때문에, 종래 기술은 SSD의 내부 병렬성을 완전히 이용하지 않았다. 따라서, 여전히, 블록 기반 데이터 구조에서 읽기 동작의 성능 개선을 위한 여지가 존재함은 전술한 바와 같다.

[71] 종래의 키 값 기반의 데이터 액세스 장치(10)는 목표로 하는 키 리스트에 도달할 때까지 Multiget method를 통하여 get 오퍼레이션(operation)을 반복하여 데이터에 접근하고, Get Method는 바이너리 서치(binary search)를 이용하여 수행되며, Index Iterator 및 Block iterator를 이용하여 전술한 블록 내의 SST파일에 마련된 데이터에 액세스 하였다. 다만, 종래의 종래의 키 값 기반의 데이터 액세스 장치(10)는 한번에 하나의 I/O 요청만을 전송하고, SSD에 저장된 데이터들에 액세스 함에 있어, 하나의 I/O 요청에 따른 수 많은 랜덤 패턴으로 액세스 하였기 때문에, SSD의 성능을 완전히 향상시킬 수 없었다. 특히, 종래의 키 값 기반의 데이터 액세스 장치(10)는 블록 사이즈가 SSD의 페이지 사이즈보다 작은 경우, 작은 사이즈의 데이터로 인한 내부 단편(Internal fragmentation)의 증가 문제 때문에, 비효율성을 야기하였다.

[72] 저장 데이터 리스트 생성부(320)는 수신된 위치 정보를 이용하여 상기 로그 구조 병합 트리 방식으로 상기 메모리 셀들에 저장된 데이터의 하위 계층으로부터 순차적으로 상기 저장 데이터 리스트를 생성한다. 예를 들어, 저장 데이터 리스트 생성부(320)가 이용하는 위치 정보는 저장부(200)의 메모리 셀들에 저장된 데이터의 위치에 관한 정보로서 블룸 필터 파일, 인덱스 파일, 파일 포인터 및 오프셋 정보를 포함할 수 있다. 본 발명의 제어부(300)는 저장 데이터 리스트 생성부(320)를 통하여 저장부(200)의 메모리 셀들에 저장된 전체 데이터 목록인 저장 데이터 리스트를 포함하고, 이를 키 리스트와 비교한다.

[73] 반환 데이터 리스트 저장부(340)는 생성된 저장 데이터 리스트와 상기 키 리스트를 상기 데이터의 하위 계층으로부터 순차적으로 비교하여 상기 키 리스트에 대응되는 상기 위치 정보가 나타내는 위치에 저장된 데이터를 반환 데이터 리스트에 저장한다. 예를 들어, 반환 데이터 리스트 저장부(340)는 저장 데이터 리스트와 키 리스트를 비교함에 있어, 데이터가 계층 구조로 저장된 저장부(200)의 하위 계층에 대응되는 저장 데이터 리스트의 목록부터 비교를 시작하여 최상위 계층에 대응되는 저장 데이터 리스트의 목록까지 순차로 비교할 수 있다. 도 6을 참조하여 설명한다.

[74] 본 발명의 키 값 기반의 데이터 액세스 장치(10)는 하기의 Multipath Multiget Method알고리즘들을 이용함으로서, 저장부(200)가 플래시 메모리 기반 저장 장치로 마련되는 경우, 내부 병렬성을 완전히 이용할 수 있다. Multipath Multiget Method 알고리즘들은 하기와 같다.

[75] [표1]

Algorithm 1: makeBlockIterator

```
Prepare psync I/O variables ;
for i← 0 to sizeof(mlist) do
    for j← 0 to ITER do
        Prepare async I/O with
          mlist.contents[j+i*ITER];
    end
    Submit async I/Os ;
    Get the results of async I/O ;
end
Free the allocated variables ;
```

[76] 상기 표1의 알고리즘 1은 PSYNC I/O 과정을 나타내는 알고리즘이다. 여기에서, PSYNC I/O 인터페이스를 위한 오퍼레이션 변수가 미리 마련될 수 있다.

[77] 오퍼레이션 변수는 버퍼, I/O 파라미터 및 상기 키 값들에 연관된 포인터 세트를 포함한다.

[78] [표2]

Algorithm 2: Multipath Multiget Method

```
   Input: keylists : list of keys that needs Multiget
          batch operation
   Output: valuelist : the result of Multiget method is
           value according to key
1  Get metadata for locking;
2  Material mlist;
   /* material consists of list of file
      descriptor(fd), block size(size), and block
      off-set(offset)                                  */
3  if keylist contents exists in memory then
4  |  Get(keys, value);
5  end
6  else
7  |  Sort (keylist) ;
8  |  Get appropriate file list by using keylist and
   |   metadata ;
9  |  Make (key, file) set with keylist and the matched
   |   filelist ;
10 |  for i← 0 to sizeof(filelist) do
11 |  |  FilePicker fp = filelist[i] ;
12 |  |  TableReader t =
   |  |   getTableReader(fp.file_descriptor) ;
   |  |  /* TableReader contains materials for
   |  |     read access                                */
13 |  |  t.appendMaterials(keylist[i], mlist) ;
14 |  |  IndexIter iiter = makeIndexIterator(t,key) ;
15 |  |  iiter.appendMaterials(keylist[i], mlist) ;
16 |  end
17 |  Sort mlist with fds and offsets ;
18 |  for i← 0 to sizeof(mlist) do
19 |  |  BlockIter biter = makeBlockIterator(mlist[i]) ;
   |  |  /* makeBlockIterator function performs
   |  |     psync operation with fd, size, offset
   |  |     from mlist                                  */
20 |  |  while biter.valid() do
21 |  |  |  biter.seek(keylist[i]) ;
22 |  |  |  if biter.value == available then
23 |  |  |  |  valuelist[i] = biter.value ;
24 |  |  |  |  break;
25 |  |  |  end
26 |  |  end
27 |  end
28 end
29 Release metadata used for locking ;
30 Return with stats of the operation ;
```

[79] 상기 표2의 알고리즘 2는 본 발명의 키 값 기반의 데이터 액세스 장치(10)가 사용하는 Multipath Multiget Method 과정을 나타낸다. 키 값 기반의 데이터 액세스 장치(10)가 사용하는 상기 표2의 알고리즘 2는 메모리에서 디스크 접근을 위한 메타데이터 수집(7~16 라인), 수집된 메타데이터의 분류(17라인) 및 I/O 요청 알고리즘(18~28 라인)을 포함한다.

[80] 먼저 본 발명의 Multipath Multiget Method 알고리즘을 이용하는 제어부(300)는 데이터 액세스를 위한 위치 정보를 수집한다. 데이터 액세스를 위한 위치 정보는

파일 디스크립터, 오프셋, 데이터의 크기에 대한 정보 및 기타 I/O 요청 셋을 포함한다. 본 발명의 저장부(200)의 데이터들은 키 값들의 최소 또는 최대값을 고려한 키 범위(KeyRange)가 미리 할당된 메모리 셀들에 구분되어 미리 저장되며, 보다 상세하게는 메모리 셀들에 마련된 내부 블록의 SST 파일 각각은 키 값의 크기에 따른 키 범위(KeyRange)가 미리 할당되어 있다.

[81] 본 발명의 제어부(300)는 일괄 수신된 전술한 파일 디스크립터, 오프셋 및 기타 I/O 요청 정보를 포함하는 위치 정보를 데이터의 위치에 따라 순차로 정렬하여 이용함으로서, 저장부(200)의 메모리 셀들에 저장된 데이터에 순차로 액세스 할 수 있다. 보다 상세하게는 본 발명의 제어부(300)는 PSYNC I/O를 이용하여 일괄 수신된 위치 정보 중 I/O 요청 셋을 데이터의 위치에 따라 순차로 정렬하여 이용함으로서 저장부(200)의 메모리 셀들에 저장된 데이터에 순차로 액세스 할 수 있다.

[82] 도 2는 도 1의 실시 예에서 저장부(200)에 로그 병합 트리 구조로 저장된 데이터를 나타낸다. 본 발명의 저장부(200)는 복수의 메모리 셀들에 저장된 데이터를 계층 구조를 가지는 트리 성분(Tree Component)으로 데이터를 관리하는데, 각 계층간 트리 성분은 각 트리 성분에 저장된 데이터 사이즈의 크기가 미리 설정된 임계치 이상인 경우 하위 트리 성분으로 합병(merge)됨은 전술한 바와 같다. LSM트리를 베이스 트리 알고리즘으로 사용하는 LevelDB 또는 RocksDB와 같은 키-벨류 스토어에서는 In-Memory 영역에서 데이터를 입력받고, 해당 Immutable Memtable의 데이터가 차면 저장부(200) 또는 디스크 스토리지로 데이터를 플러싱한다.

[83] 본 발명에서 저장부(200)는 복수의 블록들이 마련된 메모리 셀들을 포함하고, 각각의 블록은 SST파일을 포함하여 데이터를 계층적으로 저장함은 전술한 바와 같다. 저장부(200)의 메모리 셀들은 키 값들의 크기에 따른 키 범위가 미리 할당되어 데이터를 구분하여 저장하는데, 보다 상세하게는 메모리 셀들에 마련된 각각의 블록들은 데이터의 최대 최소값을 고려하여 설정된 키 범위가 미리 할당되어, 데이터를 구분하여 저장할 수 있다. 예를 들어, L1에서 file1로 표기된 SST 파일 1은 100,200의 키 범위(Key Range)를 가질 수 있다.

[84] 도 3은 도 1의 실시예에서 제어부(300)의 확대 블록도이다.

[85] 제어부(300)는 저장 데이터 리스트 생성부(320) 및 반환 데이터 리스트 저장부(340)를 포함한다. 예를 들어, 제어부(300)는 메모리 셀들에 저장된 데이터의 위치 정보를 병렬적으로 요청하여 수신하고, 상기 수신된 위치 정보로부터 저장 데이터 리스트를 생성하며, 상기 생성된 저장 데이터 리스트와 상기 키 리스트를 비교하여 상기 메모리 셀들에 저장된 데이터를 액세스함은 전술한 바와 같으므로 생략한다.

[86] 도 4는 플래시 메모리 기반 저장 장치에서 저장된 데이터의 구조를 나타내는 개념도이다.

[87] SSD는 SSD 컨트롤러(300) 및 복수의 플래시 칩을(272, 274)를 포함할 수 있다.

하지만 본 발명에서 SSD 컨트롤러는 제어부(300), 복수의 플래시 칩이 포함된 영역은 저장부(200)로 구현될 수 있음은 전술한 바와 같다. 저장부(200)의 메모리 셀들은 복수의 블록들을 포함하고, 계층으로 데이터를 구분하여 저장할 수 있다. 예를 들어, 저장부(200)의 메모리 셀들에 포함된 하나의 블록은 SST파일을 포함하고, SST파일은 블룸 필터 파일과 인덱스 파일을 포함하여 블록 내 저장된 데이터에 액세스 할 수 있다.

[88] 도 5는 종래의 키 값 기반의 데이터 액세스 장치에서 수행되는 데이터 액세스 과정을 나타낸다.

[89] 종래의 키 값 기반의 데이터 액세스 장치는 사용자(20)로부터 적어도 하나의 키 값을 포함하는 키 리스트를 입력 받고, 입력된 키 리스트를 이용하여 저장부(200)에 저장된 데이터에 단일의 순차 I/O 요청들을 이용하여 반복적인 랜덤 접근 패턴으로 액세스 하였다. SSD는 페이지 기반 블록들을 이용하여 데이터를 저장하기 때문에 데이터에 접근시 시퀀셜(Sequential) 패턴으로 액세스 하는 것이 효율적인데, 종래의 키 값 기반의 데이터 액세스 장치는 단일의 I/O요청을 이용하여 데이터에 접근할 뿐만 아니라, 랜덤 액세스 패턴으로 접근하였기 때문에 SSD의 성능을 온전히 이용할 수 없었음은 전술한 바와 같다.

[90] 도 6은 본 발명의 일 실시 예에 따른 키 값 기반의 데이터 액세스 장치에서 수행되는 데이터 액세스 과정을 나타낸다.

[91] 본 발명의 키 값 기반의 데이터 액세스 장치(10)는 메모리 셀들에 저장된 데이터의 위치 정보를 오퍼레이션 변수를 기반으로 동작하는 병렬 동기 I/O 인터페이스(Parallel Synchronous I/O Interface, PSYNC I/O)를 이용하여 일괄 수신한다. 또한, 키 값 기반의 데이터 액세스 장치(10)가 병렬 동기 I/O 인터페이스(PSYNC I/O)를 이용하여 일괄 수신된 위치 정보는 데이터의 입출력을 요청하는 I/O 요청 셋을 포함하고, 상기 I/O 요청 셋은 상기 메모리 셀들에 저장된 데이터의 위치에 따라 순차로 정렬될 수 있다.

[92] 따라서, 본 발명의 키 값 기반의 데이터 액세스 장치(10)는 RocksDB에서 SSD의 내부 병렬성을 이용하기 위해, PSYNC I/O인터페이스를 이용하여 단일의 시스템 콜에서 멀티 I/O요청(I/O Set)을 전송함과 동시에, SSD의 페이지 기반의 블록들에 데이터 액세스를 시도함에 있어서 시퀀셜 액세스함으로서 키 값 기반의 데이터 액세스 장치의 성능을 향상시킬 수 있다. 키 값 기반의 데이터 액세스 장치(10)는 플래시 메모리 기반 저장 장치를 저장부로 포함하는 경우 페이지 기반 블록 구조로 데이터를 저장하는 특성상, 효율적인 시퀀셜 액세스를 통하여 완전히 SSD의 내부 병렬성을 이용한다.

[93] 도 7은 멀티겟 수의 변화에 따라 측정된 키 값 기반의 데이터 검색 장치(10)의 성능을 나타낸다.

[94] 종래 Original Multiget 방법과 달리 Sequemtial Multiget 방법을 사용하는 키 값 기반의 데이터 액세스 장치(10)의 성능을 RocksDB 벤치마크를 이용하여 측정하였다. 디스크 성능 차이를 구분하기 위하여 측정 시 OS buffer를 disabled로

설정하였고, 구체적인 하드웨어 및 소프트웨어의 설정은 하기와 같다.

[95] [표3]

CPU	Intel Core i7-6700K CPU @ 4.00 GHz
Memory	64 GB
SSD	Samsung 850 Pro 512 GB

[96] 상기 표 3은 하드웨어 설정을 나타낸다. 키 값 기반의 데이터 액세스 장치(10)의 성능을 측정하기 위하여 CPU는 Interl Core i7-6700K CPU @ 4.00 GHz를 사용하였고, 64 GB의 메모리 및 Samsung 850 Pro 512 GB를 사용하여 측정을 실시 하였다.

[97] [표4]

RocksDB	4.4.1
OS	CentOS 7.3
Kernel	Linux Kernel 3.1
File System	ext4

[98] 상기 표 4는 소프트웨어 설정을 나타낸다. 키 값 기반의 데이터 액세스 장치(10)의 성능을 측정하기 위하여 RocksDB 버전 4.4.1 환경에서 벤치마크 툴을 사용하였고, RocksDB의 성능 측정을 위하여 구체적으로는 fillseq, fillrandom 및 multiget 벤치마크를 사용하였다. 측정을 위하여 키 값은 16 byte 랜덤 정수로 설정하였고, 각 키당 벨류는 512 bytes로 설정하였으며, 플래시 페이지의 사이즈가 8KB임을 고려하여 8KB 사이즈의 블록을 사용하였다. 도 7의 차트들은 multiget 엔트리(entries)의 범위는 1000~1000000으로 설정하였고, fetch 사이즈는 8MB~8GB로 설정하였다. 도 7(a) 및 도 7(c)에서 보면, Sequemtial Multiget(704)을 이용하는 본 발명의 키 값 기반의 데이터 액세스 장치(10)는 종래의 기법(702) 보다 Elapsed timse이 낮게 측정되며, multiget 엔트리가 증가할수록 그 성능차이가 두드러지는 것을 확인할 수 있다. 이러한 성능 차이는 키 값 기반의 데이터 액세스 장치(10)가 정렬된 오프셋 정보를 사용하여 SSD에 시퀀셜 액세스함에 따른 것임은 전술한 바와 같다. 도 7(b) 는 키당 멀티 겟을 분류하는데 걸리는 시간을 나타내는데, 평균적으로 0.1 us가 소모되고 분류하는데 걸리는 시간(sorting cost)에 의한 영향은 데이터량이 증가할수록 미미하다.

[99] 도 8은 멀티겟 비율의 변화에 따라 측정된 키 값 기반의 데이터 검색 장치의 성능을 나타낸다. 도 9는 데이터 블록의 사이즈에 따라 측정된 키 값 기반의 데이터 검색 장치의 성능을 나타낸다.

[100] [수식1]

$$multiget\ ratio = \frac{number\ of\ multiget\ keys}{number\ of\ total\ keys}$$

[101] 여기에서 multiget ratio는 멀티 겟 비율이고, number of multiget keys는 멀티겟(multiget) 키의 수, number of total keys는 총 키의 수를 나타낸다. 키 값 기반의 데이터 액세스 장치(10)의 성능은 멀티 겟 비율(multiget ratio)에 따라 달라질 수 있다. 멀티 겟 비율은 1~100%까지 변화시킴에 있어서, 키 값 기반의 데이터 액세스 장치(10)의 성능은 도 8에 도시된 바와 같다. 벨류 사이즈는 블록 내 페이지의 사이즈와 동일한 8KB로 고정하고, 멀티 겟 비율을 변화시킨 결과, 키 값 기반의 데이터 액세스 장치의 성능을 크게 달라지지 않음을 볼 수 있다.

[102] 본 발명의 키 값 기반의 데이터 액세스 장치가 이용하는 multi path multi get(MPMG)은 배치(batch)사이즈가 증가할수록, 종래의 multiget 방법을 사용하는 경우보다 더 나은 성능을 나타내었고, 배치(batch) 사이즈가 10000인 경우 본 발명의 MPMG 방법을 이용하는 키 값 기반의 데이터 액세스 장치(10)의 성능을 종래의 multiget 방법을 이용하는 경우보다 100% 향상된 성능을 나타내었다. 본 발명의 키 값 기반의 데이터 액세스 장치(10)는 벨류 사이즈가 256 byte인 경우 종래의 multiget 방법에 비하여 가장 향상된 성능을 나타내었다. 본 발명의 키 값 기반의 데이터 액세스 장치(10)는 많은 수의 멀티플 get operation을 처리하는 경우, 효과적으로 동작하며, 종래의 기법과 달리 multiple batch asynchronous i/o를 이용하기 때문에 플래시 기반 저장 장치의 병렬성을 완전히 이용할 수 있다. 따라서, 키 값 기반의 데이터 액세스 장치(10)는 간단히 디스크 파일에 random 접근 패턴으로 액세스 하던 종래 방법과 달리, 병렬 접근 패턴으로 디스크 파일에 액세스 함으로서, 좀더 많은 수의 multiple i/o 요청을 전송할 수 있는 장점이 있다. 본 발명의 키 값 기반의 데이터 액세스 장치(10)를 이용하는 경우 RocksDB 벤치마크의 쿼리 실행 타임을 80%까지 감소시킬 수 있으며, multiget 엔트리(entry) 사이즈가 클수록 좋은 성능을 나타냄은 전술한 바와 같다.

[103] 도 10은 본 발명의 일 실시 예에 따른 키 값 기반의 데이터 검색 방법의 흐름도를 나타낸다.

[104] 키 값 기반의 데이터 검색 방법은 키 값 기반의 데이터 검색 장치(10)에서 시계열적으로 수행되는 하기의 단계들을 포함한다.

[105] S100에서, 서버(100)는 사용자(20)의 입력에 기초하여 적어도 하나 이상의 키 값들을 포함하는 키 리스트를 입력 받는다. 예를 들어, 서버(100)는 상기 키 값들을 기반으로 동작하는 적어도 하나의 인터페이스(Key Value Operations Interface)를 이용하여 상기 키 리스트를 입력 받을 수 있다. 상기 서버(100)가 이용하는 키 발류 동작 인터페이스는 multiget으로 마련될 수 있다. 또한, 서버(100)가 입력 받는 키 값들은 저장된 데이터의 위치를 나타내는 이진 시퀀스로 마련될 수 있음은 전술한 바와 같다.

[106] S200에서, 제어부(300)는 데이터를 저장하기 위한 복수의 비휘발성 메모리 셀들에 저장된 상기 데이터의 위치 정보를 병렬적으로 요청한다. 제어부(300)는 복수의 비휘발성 메모리셀들에 저장된 데이터의 위치 정보를 요청함에 있어 PSYNC I/O를 이용할 수 있음은 전술한 바와 같다.

[107] S300에서, 저장 데이터 리스트 생성부(320)는 위치 정보로부터 상기 메모리 셀들에 저장된 데이터 목록을 나타내는 저장 데이터 리스트를 생성한다. 예를 들어, 저장 데이터 리스트 생성부(320)는 위치 정보를 이용하여 상기 로그 구조 병합 트리 방식으로 상기 메모리 셀들에 저장된 데이터의 하위 계층으로부터 순차적으로 상기 저장 데이터 리스트를 생성할 수 있다. 또한, 저장 데이터 리스트 생성부(320)는 메모리 셀들에 저장된 데이터의 위치 정보를 오퍼레이션 변수를 기반으로 동작하는 병렬 동기 I/O 인터페이스(Parallel Synchronous I/O Interface)를 이용하여 일괄 수신하고, 상기 오퍼레이션 변수는 버퍼, I/O 파라미터 및 상기 키 값들에 연관된 포인터 세트 중 적어도 하나를 포함함은 전술한 바와 같다. 여기에서, 병렬 동기 I/O 인터페이스를 이용하여 일괄 수신된 위치 정보는 데이터의 입출력을 요청하는 I/O 요청 셋을 포함하고, 상기 I/O 요청 셋은 상기 메모리 셀들에 저장된 데이터의 위치에 따라 순차로 정렬될 수 있다.

[108] 예를 들어, 본 발명의 메모리 셀들은 상기 데이터 전송을 위한 멀티 채널이 마련된 낸드(nand)형 플래시 메모리를 포함하고, 상기 플래시 메모리는 상기 키 값들의 크기에 따른 키 범위(Key Range)가 미리 할당되어 상기 데이터를 구분하여 저장하는 복수의 블록을 포함할 수 있음은 전술한 바와 같다. 또한, 상기 메모리 셀들은 데이터를 계층적으로 처리하는 로그 구조 병합 트리(Log Structure Merged Tree) 방식으로 저장하고, 상기 저장된 데이터는 적어도 하나의 계층들로 구분되어 상기 계층간 압축(compaction)을 통하여 관리될 수 있음은 전술한 바와 같다.

[109] 또한, 상기 블록은 상기 데이터가 블록에 포함되었는지 여부를 해쉬값을 출력으로 가지는 미리 설정된 해쉬 함수를 이용하여 판단하는 블룸 필터 파일 및 상기 블룸 필터 파일에서 데이터가 포함된 것으로 판단되는 경우 상기 데이터가 상기 블록의 어느 위치에 있는지에 관한 오프셋 정보를 출력하는 인덱스 파일을 포함하고, 상기 메모리 셀들은 복수의 키-벨류(Key-Value) 쌍을 이용하여 상기 데이터를 로그 구조 병합 트리(Log Structure Merged Tree) 방식으로 저장할 수 있다. S400에서, 제어부(300)는 생성된 저장 데이터 리스트와 상기 키 리스트를 비교하여 상기 메모리 셀들에 저장된 데이터를 액세스한다. 제어부(300)가 생성된 저장 데이터 리스트와 키 리스트를 비교하여 저장부(200)에 저장된 데이터에 액세스 하는 구체적인 방법은 전술한 바와 같으므로 생략한다.

[110] 도 11은 도 10의 실시 예에서 액세스 하는 단계의 확대 흐름도이다.

[111] S420에서, 제어부(300)는 생성된 저장 데이터 리스트와 상기 키 리스트를 상기 데이터의 하위 계층으로부터 순차적으로 비교한다. 예를 들어, 저장부(200)에 저장된 데이터들은 계층 구조로서 LSM트리로 관리되므로, 저장 데이터 리스트 역시 하위 계층에서 상위 계층 방향으로 작성될 수 있다.

[112] S440에서, 제어부(300)는 비교 결과에 따라 키 리스트에 대응되는 상기 위치 정보가 나타내는 위치에 저장된 데이터를 반환 데이터 리스트에 저장한다. 보다 상세하게는 제어부(300)에 마련된 반환 데이터 리스트 저장부(340)는 비교

결과에 따라 키 리스트에 대응되는 상기 위치 정보가 나타내는 위치에 저장된 데이터를 반환 데이터 리스트에 저장한다. 예를 들어, 제어부(300)는 키 리스트와 저장 데이터 리스트를 비교함에 있어 데이터의 최상위 계층까지 비교 후 상기 데이터가 저장된 반환 데이터 리스트를 이용하여 상기 메모리 셀들에 저장된 데이터에 액세스할 수 있다.

[113] 상기 설명된 본 발명의 일 실시예의 방법의 전체 또는 일부는, 컴퓨터에 의해 실행되는 프로그램 모듈과 같은 컴퓨터에 의해 실행 가능한 기록 매체의 형태(또는 컴퓨터 프로그램 제품)로 구현될 수 있다. 여기에서, 컴퓨터 판독 가능 매체는 컴퓨터 저장 매체(예를 들어, 메모리, 하드디스크, 자기/광학 매체 또는 SSD(Solid-State Drive) 등)를 포함할 수 있다. 컴퓨터 판독 가능 매체는 컴퓨터에 의해 액세스될 수 있는 임의의 가용 매체일 수 있고, 휘발성 및 비휘발성 매체, 분리형 및 비분리형 매체를 모두 포함한다.

[114] 또한, 본 발명의 일 실시예에 따르는 방법의 전체 또는 일부는 컴퓨터에 의해 실행 가능한 명령어를 포함하며, 컴퓨터 프로그램은 프로세서에 의해 처리되는 프로그래밍 가능한 기계 명령어를 포함하고, 고레벨 프로그래밍 언어(High-level Programming Language), 객체 지향 프로그래밍 언어(Object-oriented Programming Language), 어셈블리 언어 또는 기계 언어 등으로 구현될 수 있다.

[115] 본 명세서에서의 부(means) 또는 모듈(Module)은 본 명세서에서 설명되는 각 명칭에 따른 기능과 동작을 수행할 수 있는 하드웨어를 의미할 수도 있고, 특정 기능과 동작을 수행할 수 있는 컴퓨터 프로그램 코드를 의미할 수도 있고, 또는 특정 기능과 동작을 수행시킬 수 있는 컴퓨터 프로그램 코드가 탑재된 전자적 기록 매체, 예를 들어 프로세서 또는 마이크로 프로세서를 의미할 수 있다. 다시 말해, 부(means) 또는 모듈(Module)은 본 발명의 기술적 사상을 수행하기 위한 하드웨어 및/또는 상기 하드웨어를 구동하기 위한 소프트웨어의 기능적 및/또는 구조적 결합을 의미할 수 있다.

[116] 따라서 본 발명의 일 실시예에 따르는 방법은 상술한 바와 같은 컴퓨터 프로그램이 컴퓨팅 장치에 의해 실행됨으로써 구현될 수 있다. 컴퓨팅 장치는 프로세서와, 메모리와, 저장 장치와, 메모리 및 고속 확장포트에 접속하고 있는 고속 인터페이스와, 저속 버스와 저장 장치에 접속하고 있는 저속 인터페이스 중 적어도 일부를 포함할 수 있다. 이러한 성분들 각각은 다양한 버스를 이용하여 서로 접속되어 있으며, 공통 머더보드에 탑재되거나 다른 적절한 방식으로 장착될 수 있다.

[117] 이상의 설명은 본 발명의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자라면 본 발명의 본질적인 특성에서 벗어나지 않는 범위 내에서 다양한 수정, 변경 및 치환이 가능할 것이다. 따라서, 본 발명에 개시된 실시예 및 첨부된 도면들은 본 발명의 기술 사상을 한정하기 위한 것이 아니라 설명하기 위한 것이고, 이러한 실시예 및 첨부된 도면에 의하여 본 발명의 기술 사상의 범위가 한정되는 것은

아니다. 본 발명의 보호 범위는 아래의 청구 범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술 사상은 본 발명의 권리 범위에 포함되는 것으로 해석되어야 할 것이다.

청구범위

[청구항 1] 사용자의 입력에 기초하여 적어도 하나 이상의 키 값들을 포함하는 키 리스트를 입력 받고, 상기 키 리스트에 따라 액세스된 데이터를 출력하는 서버;
상기 데이터를 저장하기 위한 복수의 비휘발성 메모리 셀들로 구분되는 저장영역을 포함하는 저장부; 및
상기 메모리 셀들에 저장된 데이터의 위치 정보를 병렬적으로 요청하여 수신하고, 상기 수신된 위치 정보로부터 저장 데이터 리스트를 생성하며, 상기 생성된 저장 데이터 리스트와 상기 키 리스트를 비교하여 상기 메모리 셀들에 저장된 데이터를 액세스 하는 제어부; 를 포함하는 키 값 기반의 데이터 액세스 장치.

[청구항 2] 제1항에 있어서, 상기 서버는
상기 키 값들을 기반으로 동작하는 적어도 하나의 인터페이스(Key Value Operations Interface)를 이용하여 상기 키 리스트를 입력 받는 것을 특징으로 하는 키 값 기반의 데이터 액세스 장치.

[청구항 3] 제2항에 있어서, 상기 저장부는
상기 데이터를 계층적으로 처리하는 로그 구조 병합 트리(Log Structure Merged Tree) 방식으로 상기 메모리 셀들에 저장하고,
상기 메모리 셀들에 저장된 데이터들은 적어도 하나의 계층들로 구분되어 상기 계층들 간의 압축(compaction)을 통하여 관리되는 것을 특징으로 하는 키 값 기반의 데이터 액세스 장치.

[청구항 4] 제3항에 있어서, 상기 제어부는
상기 위치 정보를 이용하여 상기 로그 구조 병합 트리 방식으로 상기 메모리 셀들에 저장된 데이터의 하위 계층으로부터 순차적으로 상기 저장 데이터 리스트를 생성하는 저장 데이터 리스트 생성부; 를 더 포함하고,
상기 생성된 저장 데이터 리스트를 이용하여 상기 메모리 셀들에 저장된 데이터를 액세스하는 것을 특징으로 하는 키 값 기반의 데이터 액세스 장치.

[청구항 5] 제4항에 있어서, 상기 제어부는
상기 메모리 셀들에 저장된 데이터의 위치 정보를 오퍼레이션 변수를 기반으로 동작하는 병렬 동기 I/O 인터페이스(Parallel Synchronous I/O Interface)를 이용하여 일괄 수신하고,
상기 오퍼레이션 변수는 버퍼, I/O 파라미터 및 상기 키 값들에 연관된 포인터 세트 중 적어도 하나를 포함하는 것을 특징으로 하는 키 값 기반의 데이터 액세스 장치.

[청구항 6] 제5항에 있어서,

상기 병렬 동기 I/O 인터페이스를 이용하여 일괄 수신된 위치 정보는 데이터의 입출력을 요청하는 I/O 요청 셋을 포함하고,
상기 I/O 요청 셋은 상기 메모리 셀들에 저장된 데이터의 위치에 따라 순차로 정렬되는 것을 특징으로 하는 키 값 기반의 데이터 액세스 장치.

[청구항 7] 제5항에 있어서,
상기 압축(compaction)은 상기 적어도 하나의 계층들에서 병렬로 수행되고, 상기 압축은 스레드 수를 고려하여 주기적으로 수행되며,
상기 저장부는 복수의 키-벨류(Key-Value) 쌍을 이용하여 상기 데이터를 로그 구조 병합 트리(Log Structure Merged Tree) 방식으로 저장하는 것을 특징으로 하는 키 값 기반의 데이터 액세스 장치.

[청구항 8] 제3항에 있어서, 상기 제어부는
상기 생성된 저장 데이터 리스트와 상기 키 리스트를 상기 데이터의 하위 계층으로부터 순차적으로 비교하여 상기 키 리스트에 대응되는 상기 위치 정보가 나타내는 위치에 저장된 데이터를 반환 데이터 리스트에 저장하는 반환 데이터 리스트 저장부; 를 더 포함하고,
상기 데이터의 최상위 계층까지 비교 후 상기 데이터가 저장된 반환 데이터 리스트를 이용하여 상기 메모리 셀들에 저장된 데이터에 액세스하는 것을 특징으로 하는 키 값 기반의 데이터 액세스 장치.

[청구항 9] 제1항에 있어서,
상기 메모리 셀들은 상기 데이터의 전송을 위한 멀티 채널이 마련된 낸드(nand)형 플래시 메모리를 포함하고,
상기 낸드형 플래시 메모리는 상기 키 값들의 크기에 따른 키 범위(Key Range)가 미리 할당되어 상기 데이터를 구분하여 저장하는 복수의 블록을 포함하는 것을 특징으로 하는 키 값 기반의 데이터 액세스 장치.

[청구항 10] 제9항에 있어서, 상기 블록은
상기 데이터가 상기 블록에 포함되었는지 여부를 해쉬값을 출력으로 가지는 미리 설정된 해쉬 함수를 이용하여 판단하는 블룸 필터 파일 및 상기 블룸 필터 파일에서 상기 데이터가 포함된 것으로 판단되는 경우 상기 데이터가 상기 블록의 어느 위치에 있는지에 관한 오프셋 정보를 출력하는 인덱스 파일을 포함하고,
상기 키 리스트는 상기 데이터의 위치를 나타내는 이진 시퀀스로 마련되는 것을 특징으로 하는 키 값 기반의 데이터 액세스 장치.

[청구항 11] 사용자의 입력에 기초하여 적어도 하나 이상의 키 값들을 포함하는 키 리스트를 입력 받는 단계;
데이터를 저장하기 위한 복수의 비휘발성 메모리 셀들에 저장된 상기 데이터의 위치 정보를 병렬적으로 요청하는 단계;
상기 위치 정보로부터 상기 메모리 셀들에 저장된 데이터 목록을 나타내는 저장 데이터 리스트를 생성하는 단계; 및

상기 생성된 저장 데이터 리스트와 상기 키 리스트를 비교하여 상기 메모리 셀들에 저장된 데이터를 액세스 하는 단계; 를 포함하는 키 값 기반의 데이터 액세스 방법.

[청구항 12] 제11항에 있어서, 상기 키 리스트를 입력 받는 단계는
상기 키 값들을 기반으로 동작하는 적어도 하나의 인터페이스(Key Value Operations Interface)를 이용하여 상기 키 리스트를 입력 받고,
상기 메모리 셀들은 상기 데이터를 계층적으로 처리하는 로그 구조 병합 트리(Log Structure Merged Tree) 방식으로 저장하며,
상기 저장된 데이터는 적어도 하나의 계층들로 구분되어 상기 계층들 간의 압축(compaction)을 통하여 관리되는 것을 특징으로 하는 키 값 기반의 데이터 액세스 방법.

[청구항 13] 제12항에 있어서, 상기 생성하는 단계는
상기 위치 정보를 이용하여 상기 로그 구조 병합 트리 방식으로 상기 메모리 셀들에 저장된 데이터의 하위 계층으로부터 순차적으로 상기 저장 데이터 리스트를 생성하고, 상기 메모리 셀들에 저장된 데이터의 위치 정보를 오퍼레이션 변수를 기반으로 동작하는 병렬 동기 I/O 인터페이스(Parallel Synchronous I/O Interface)를 이용하여 일괄 수신하며,
상기 병렬 동기 I/O 인터페이스를 이용하여 일괄 수신된 위치 정보는 데이터의 입출력을 요청하는 I/O 요청 셋을 포함하고,
상기 I/O 요청 셋은 상기 메모리 셀들에 저장된 데이터의 위치에 따라 순차로 정렬되며,
상기 오퍼레이션 변수는 버퍼, I/O 파라미터 및 상기 키 값들에 연관된 포인터 세트 중 적어도 하나를 포함하고,
상기 액세스 하는 단계는,
상기 생성된 저장 데이터 리스트와 상기 키 리스트를 상기 데이터의 하위 계층으로부터 순차적으로 비교하는 단계; 및
비교 결과에 따라 키 리스트에 대응되는 상기 위치 정보가 나타내는 위치에 저장된 데이터를 반환 데이터 리스트에 저장하는 단계; 를 더 포함하고,
상기 데이터의 최상위 계층까지 비교 후 상기 데이터가 저장된 반환 데이터 리스트를 이용하여 상기 메모리 셀들에 저장된 데이터에 액세스 하는 것을 특징으로 하는 키 값 기반의 데이터 액세스 방법.

[청구항 14] 제11항에 있어서,
상기 메모리 셀들은 상기 데이터의 전송을 위한 멀티 채널이 마련된 낸드(nand)형 플래시 메모리를 포함하고,
상기 낸드형 플래시 메모리는 상기 키 값들의 크기에 따른 키 범위(Key Range)가 미리 할당되어 상기 데이터를 구분하여 저장하는 복수의 블록을 포함하며,

상기 블록은 상기 데이터가 상기 블록에 포함되었는지 여부를 해쉬값을 출력으로 가지는 미리 설정된 해쉬 함수를 이용하여 판단하는 블룸 필터 파일 및 상기 블룸 필터 파일에서 상기 데이터가 포함된 것으로 판단되는 경우 상기 데이터가 상기 블록의 어느 위치에 있는지에 관한 오프셋 정보를 출력하는 인덱스 파일을 포함하고,

상기 메모리 셀들은 복수의 키-벨류(Key-Value) 쌍을 이용하여 상기 데이터를 로그 구조 병합 트리(Log Structure Merged Tree) 방식으로 저장하며,

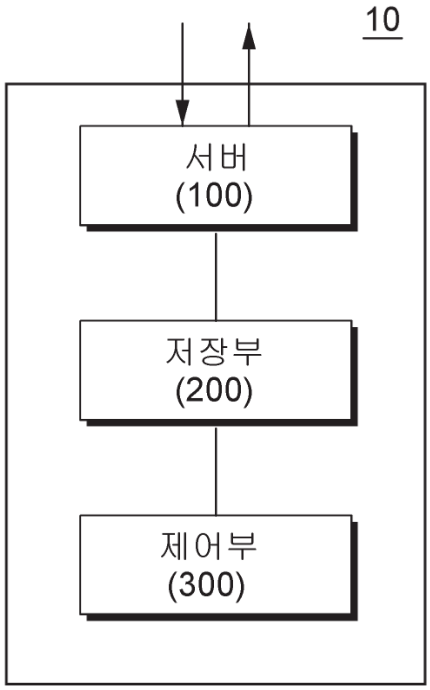
상기 키 리스트는 상기 데이터의 위치를 나타내는 이진 시퀀스로 마련되는 것을 특징으로 하는 키 값 기반의 데이터 액세스 방법.

[청구항 15] 프로세서에 의해 실행되는 것을 통하여 제11항 내지 제14항 중 어느 한 항에 기재된 키 값 기반의 데이터 액세스 방법을 실현하는 컴퓨터에서 판독 가능한 기록매체에 저장된 프로그램.

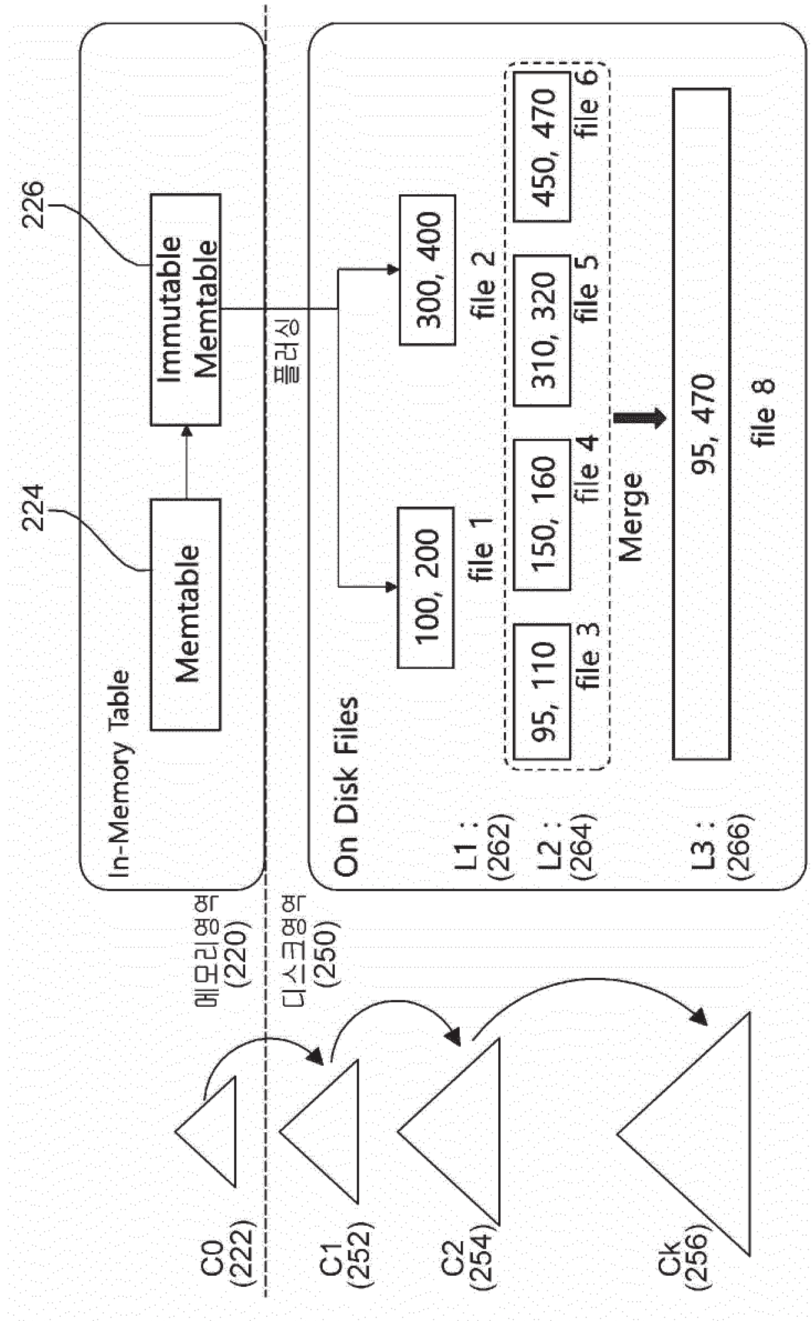
요약서

본 발명은 데이터 입출력 성능을 향상시키도록 사용자의 입력에 기초하여 적어도 하나 이상의 키 값들을 포함하는 키 리스트를 입력 받고 키 리스트에 따라 액세스된 데이터를 출력하는 서버, 복수의 비휘발성 메모리 셀들로 구분되는 저장영역을 포함하는 저장부 및 메모리 셀들에 저장된 데이터의 위치 정보를 병렬적으로 요청하여 수신하여 생성한 저장 데이터 리스트와 키 리스트를 비교하여 메모리 셀들에 저장된 데이터를 액세스 하는 제어부를 포함하는 키 값 기반의 데이터 액세스 장치를 개시한다.

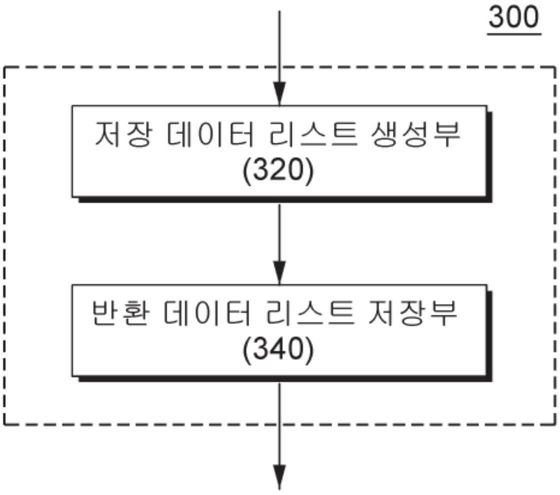
[도1]



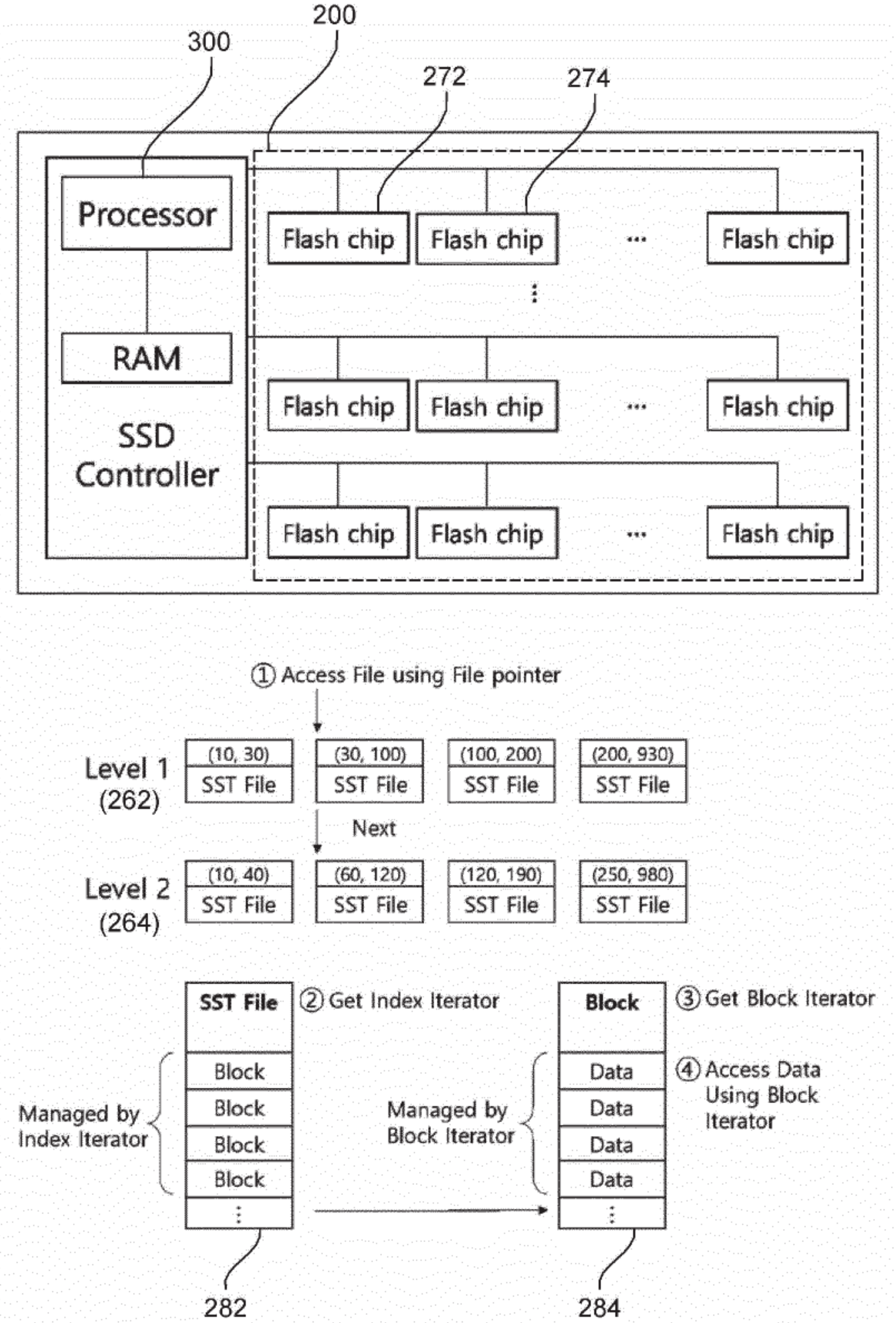
[도2]


In-Memory Table
Memtable
Immutable Memtable
224
226
플러싱
메모리영역 (220)
디스크영역 (250)
On Disk Files
L1 : (262)
L2 : (264)
L3 : (266)
100, 200
file 1
300, 400
file 2
95, 110
file 3
150, 160
file 4
310, 320
file 5
450, 470
file 6
Merge
95, 470
file 8
C0 (222)
C1 (252)
C2 (254)
Ck (256)

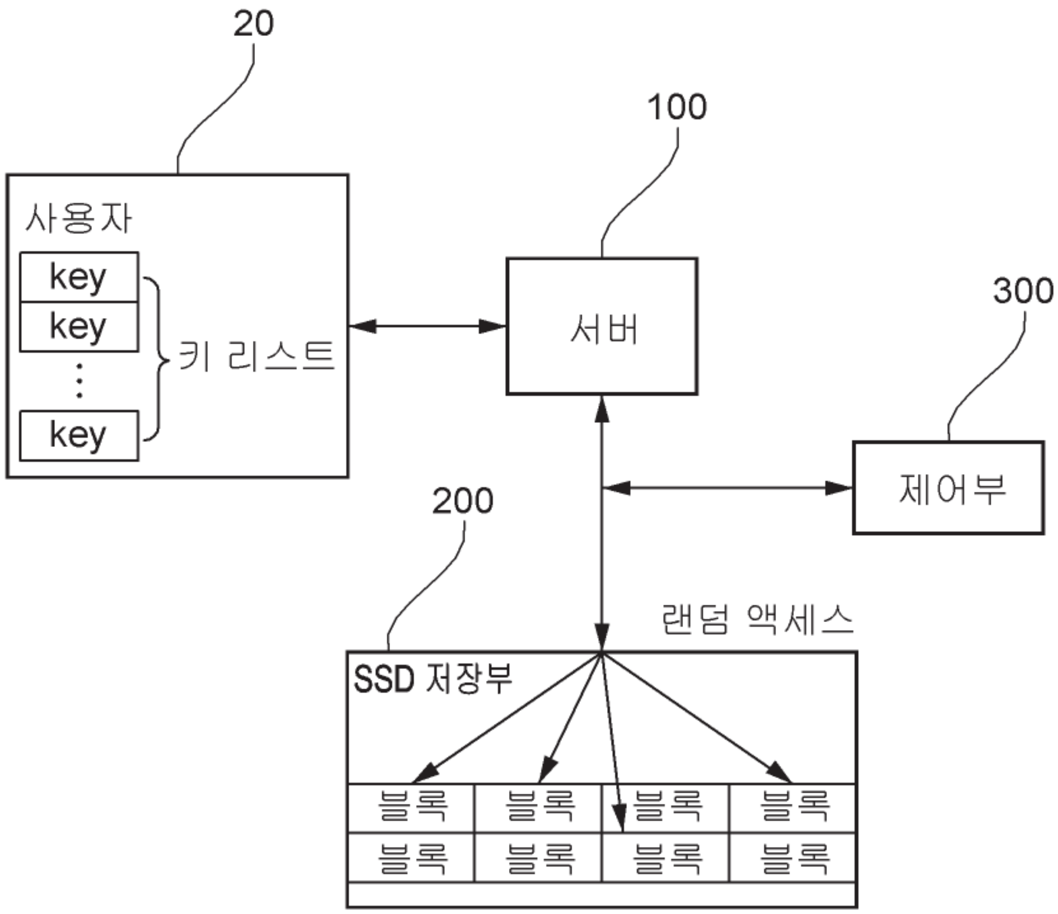
[도3]



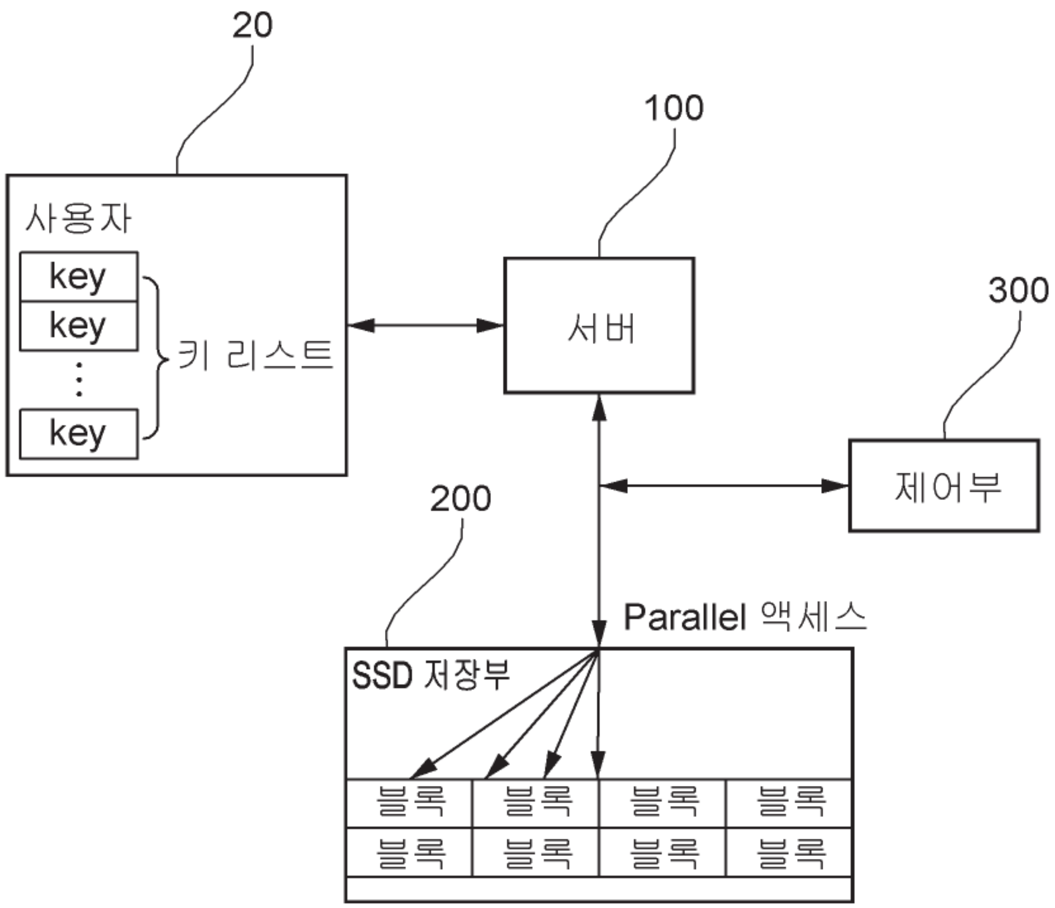
[도4]


200
300
272
274
Processor
RAM
SSD
Controller
Flash chip
Flash chip
Flash chip
Flash chip
Flash chip
Flash chip
Flash chip
Flash chip
Flash chip
① Access File using File pointer
Level 1
(262)
(10, 30)
SST File
(30, 100)
SST File
(100, 200)
SST File
(200, 930)
SST File
Next
Level 2
(264)
(10, 40)
SST File
(60, 120)
SST File
(120, 190)
SST File
(250, 980)
SST File
SST File
② Get Index Iterator
Block
③ Get Block Iterator
Block
Block
Block
Block
Managed by
Index Iterator
Managed by
Block Iterator
Data
Data
Data
Data
④ Access Data
Using Block
Iterator
282
284

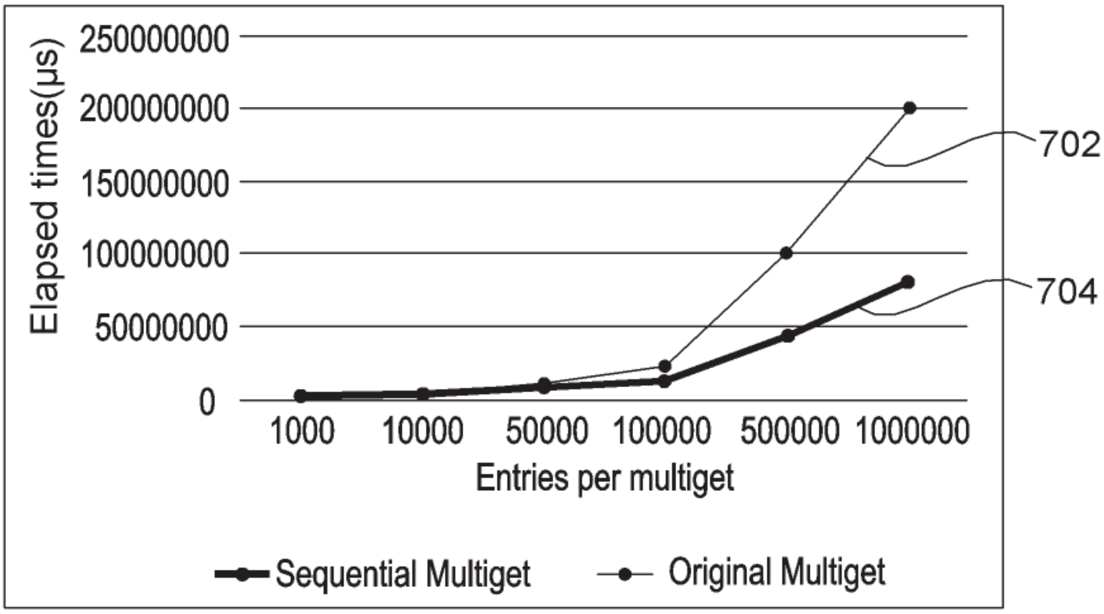
[도5]


20
100
300
사용자
key
key
key
키 리스트
서버
제어부
200
랜덤 액세스
SSD 저장부
블록
블록
블록
블록
블록
블록
블록
블록

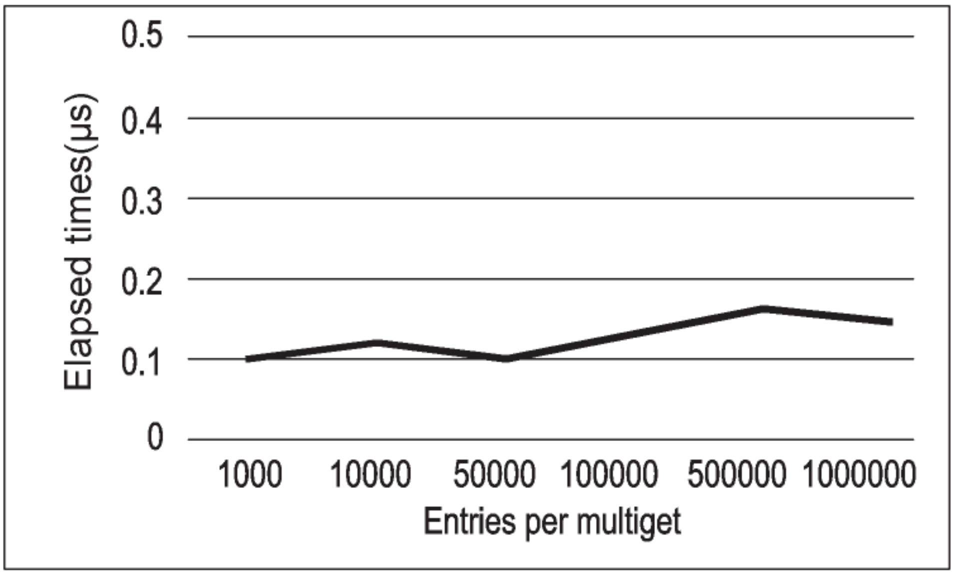
[도6]


20
100
300
사용자
key
key
key
키 리스트
서버
제어부
200
Parallel 액세스
SSD 저장부
블록
블록
블록
블록
블록
블록
블록
블록

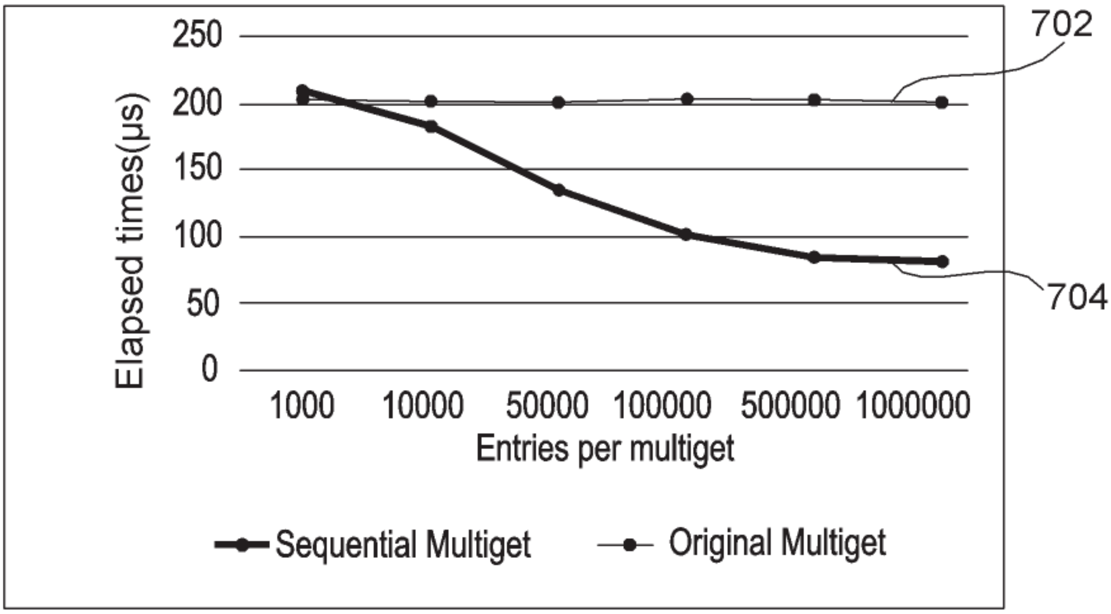
[도7a]


250000000
200000000
150000000
100000000
50000000
0
Elapsed times(μs)
1000
10000
50000
100000
500000
1000000
Entries per multiget
702
704
Sequential Multiget
Original Multiget

[도7b]


0.5
0.4
0.3
0.2
0.1
0
Elapsed times(μs)
1000
10000
50000
100000
500000
1000000
Entries per multiget

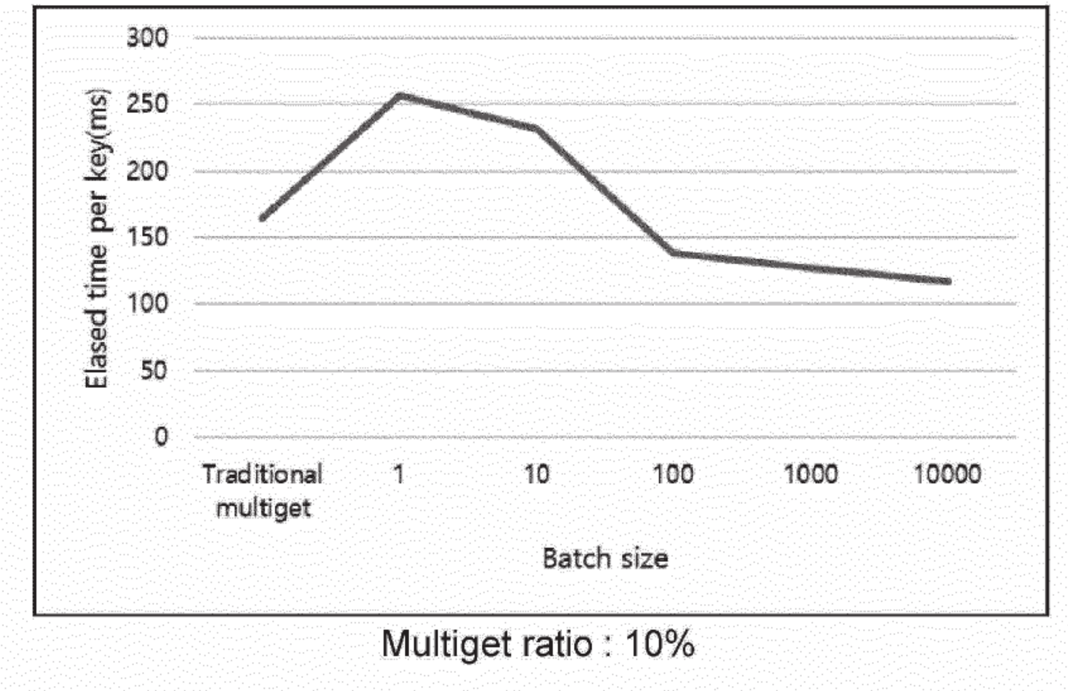
[도7c]



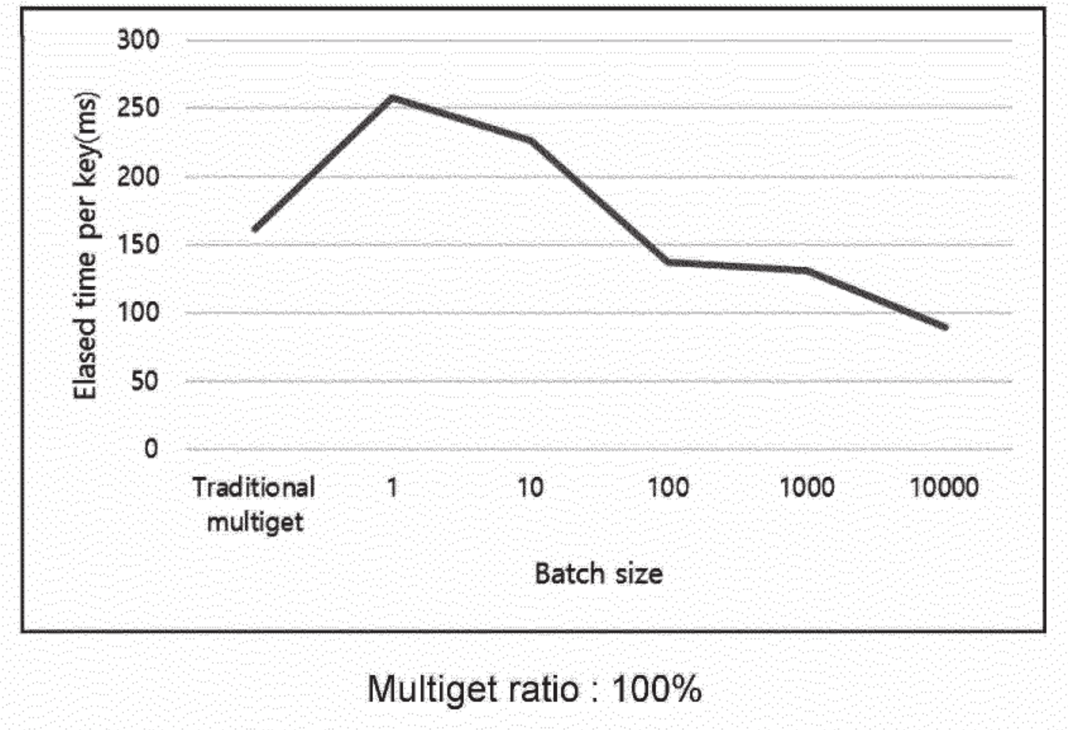
[도8a]

Multiget ratio : 1%

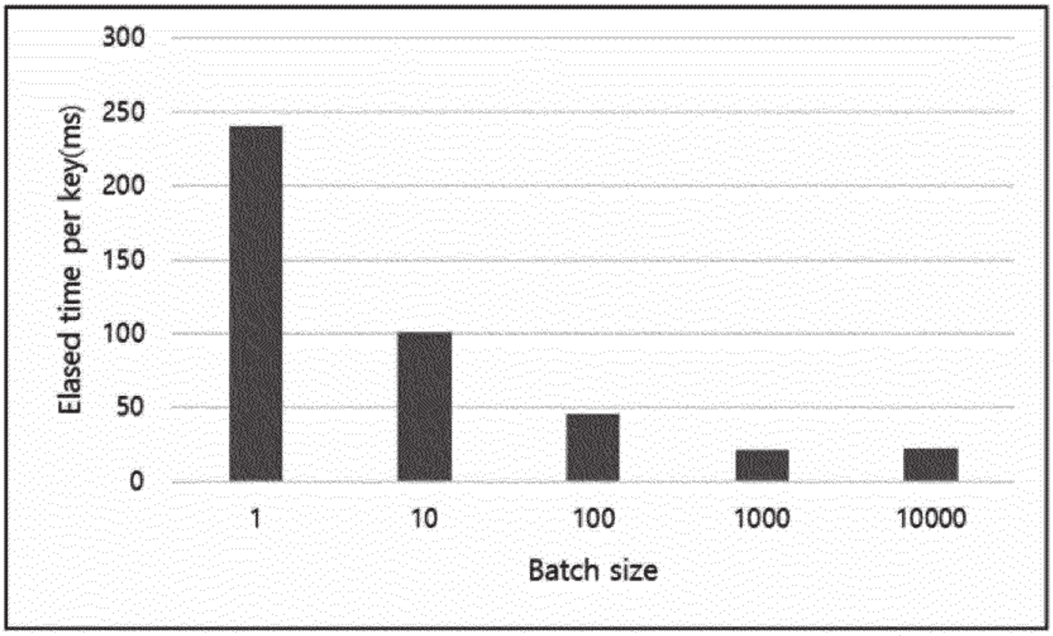
[도8b]


300
250
200
150
100
50
0
Elased time per key(ms)
Traditional multiget
1
10
100
1000
10000
Batch size
Multiget ratio : 10%

[도8c]

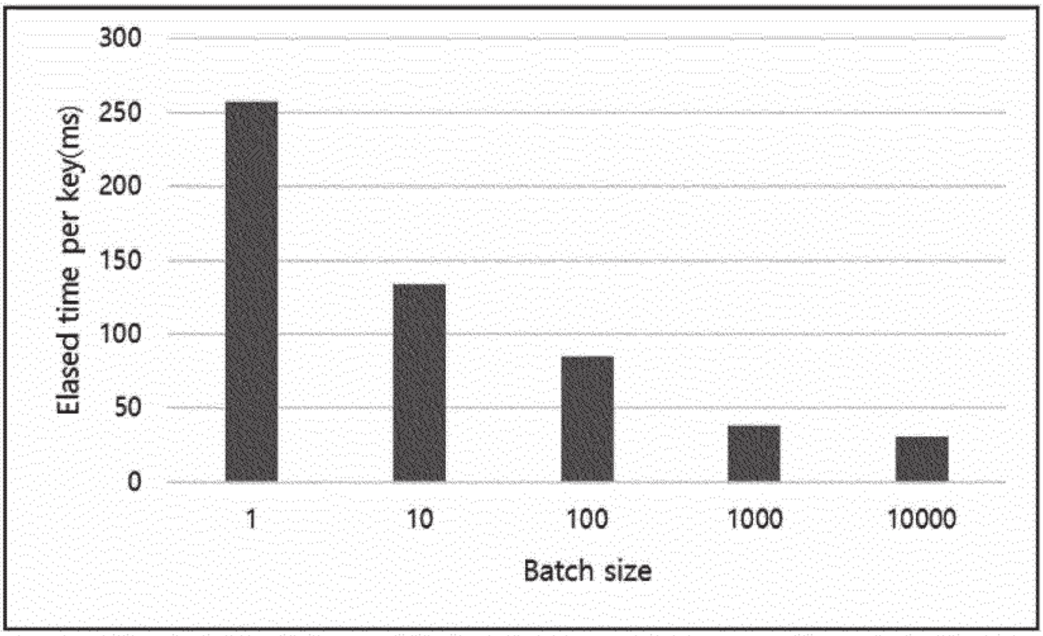

300
250
200
150
100
50
0
Elased time per key(ms)
Traditional multiget
1
10
100
1000
10000
Batch size
Multiget ratio : 100%

[도9a]



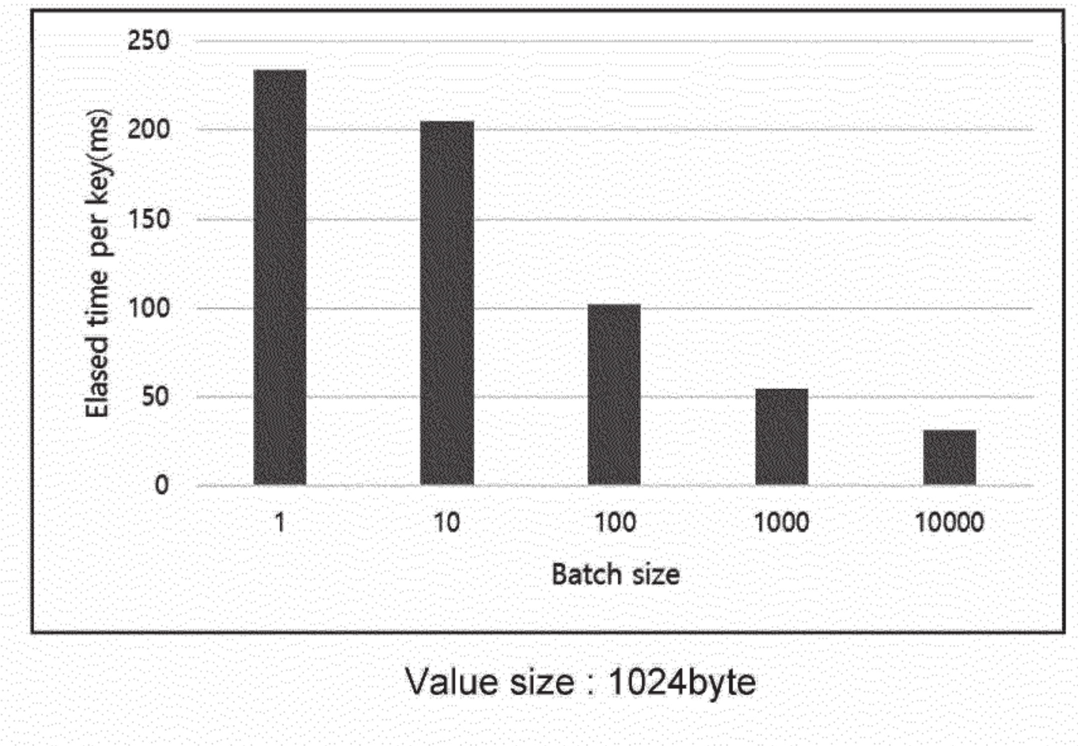
Value size : 256byte

[도9b]

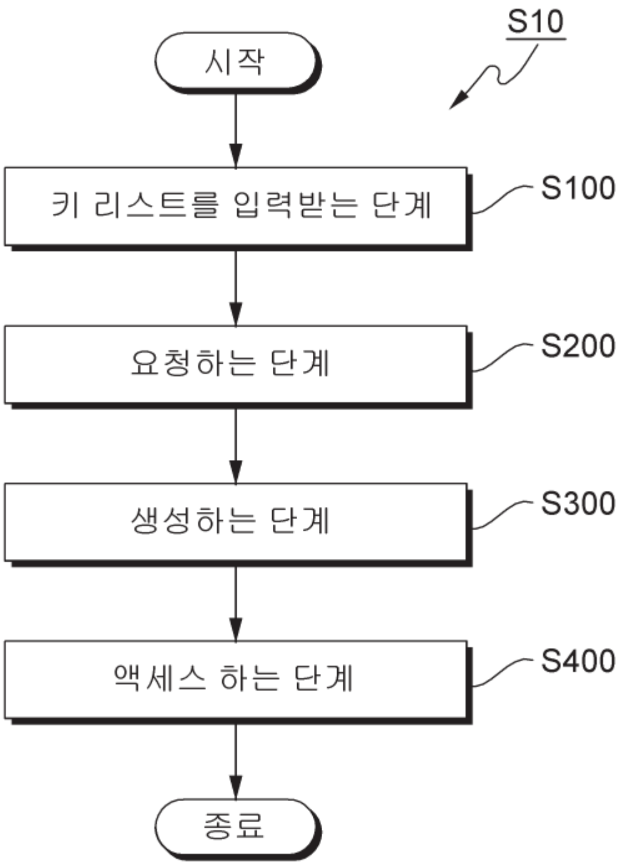


Value size : 512byte

[도9c]


250
200
150
100
50
0
Elased time per key(ms)
1
10
100
1000
10000
Batch size
Value size : 1024byte

[도10]


S10
시작
키 리스트를 입력받는 단계
S100
요청하는 단계
S200
생성하는 단계
S300
액세스 하는 단계
S400
종료

[도11]