(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0087067
(43) 공개일자 2020년07월20일

(51) 국제특허분류(Int. Cl.)
G06F 9/46 (2006.01) *G06F 9/38* (2006.01)
G06F 9/50 (2018.01) *G06F 9/54* (2018.01)
(52) CPC특허분류
G06F 9/468 (2013.01)
G06F 9/3885 (2013.01)
(21) 출원번호 10-2019-0161560
(22) 출원일자 2019년12월06일
심사청구일자 없음
(30) 우선권주장
62/790,705 2019년01월10일 미국(US)
(뒷면에 계속)

(71) 출원인
삼성전자주식회사
경기도 수원시 영통구 삼성로 129 (매탄동)
(72) 발명자
양, 징페이
95148 미국 캘리포니아주 산호세 브레든 코트 3390
브히마니, 잔키
01821 미국 매사추세츠주 빌레리카 폴거 스트리트 46
최창호
95120 미국 캘리포니아 산호세 마운트 홀리 드라이브 6622
(74) 대리인
하영욱

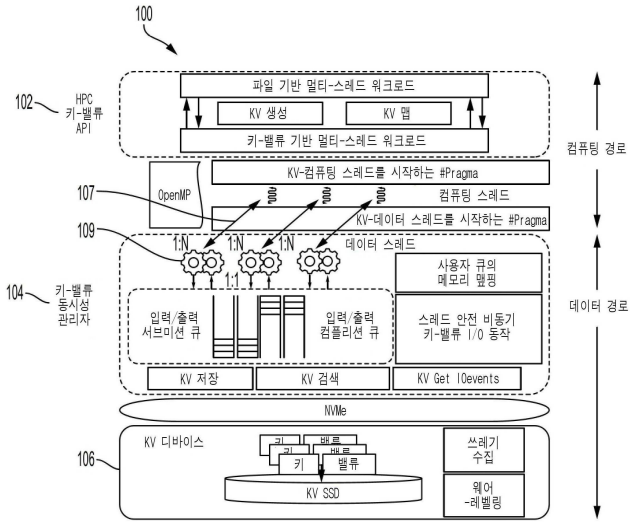
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **병렬 컴퓨팅을 위한 키-밸류 기반 시스템 및 병렬화된 애플리케이션을 동작하는 방법**

(57) 요 약

키 밸류 솔리드 스테이트 드라이브의 사용을 활용하는 병렬 컴퓨팅을 위한 시스템에 관한 것이다. 상기 시스템은 병렬 컴퓨팅 및 데이터 스레드의 사용을 가능하게 하는 수정된 컴파일러 지시문의 세트를 포함한다. 상기 시스템은 병렬 데이터 스레드가 스레드-안전 방식으로 동작되는 것을 보장하기 위해 동시성 관리자를 더 포함할 수 있다.

대 표 도



(52) CPC특허분류
G06F 9/5044 (2013.01)
G06F 9/546 (2013.01)
G06F 2209/5017 (2013.01)
G06F 2209/5018 (2013.01)

(30) 우선권주장
62/863,558 2019년06월19일 미국(US)
16/528,492 2019년07월31일 미국(US)

명 세 서

청구범위

청구항 1

키 밸류 솔리드 스테이트 드라이브; 및

프로세싱 회로를 포함하고,

상기 프로세싱 회로는:

멀티스레디드 애플리케이션을 실행하고;

키 밸류 병렬 컴퓨팅 시스템을 실행하고; 및

상기 멀티스레디드 애플리케이션과 병렬로 복수의 데이터 스레드를 동작시키도록 구성되고,

상기 데이터 스레드는 상기 키 밸류 솔리드 스테이트 드라이브를 사용하여 병렬 키-밸류 입력 동작 및 출력 동작을 수행하고,

각각의 데이터 스레드는 상기 멀티스레디드 애플리케이션의 적어도 하나의 컴퓨팅 스레드와 연관되고,

상기 키 밸류 병렬 컴퓨팅 시스템은:

멀티스레딩 플랫폼,

멀티스레디드 커널 디바이스 드라이버, 및

키-밸류 동시성 관리자를 포함하고,

상기 키-밸류 동시성 관리자는 상기 키 밸류 솔리드 스테이트 드라이브로 스레드-안전 비동기식 키-밸류 입력 동작 및 출력 동작을 관리하도록 구성되는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 2

제1항에 있어서,

상기 키-밸류 동시성 관리자는 복수의 사용자 큐를 관리하도록 더 구성되고, 각각의 사용자 큐는 연관된 사용자 큐 ID를 갖는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 3

제1항에 있어서,

상기 멀티스레디드 애플리케이션의 실행은:

파일 기반 멀티스레드 워크로드를 키-밸류 기반 멀티스레드 워크로드로 변환하는 것;

제1 파일 데이터 청크에 대한 제1 밸류를 생성하는 것;

상기 제1 밸류에 대한 제1 키를 생성하는 것;

상기 제1 키를 상기 제1 밸류와 연관시키고 상기 제1 키를 다시 제1 파일에 맵핑하는 제1 메타데이터를 생성하는 것; 및

상기 제1 키를 제1 스레드 ID 및 제1 사용자 큐 ID와 연관시키는 제2 메타데이터를 생성하는 것을 포함하는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 4

제3항에 있어서,

상기 파일 기반 멀티스레드 워크로드를 키-밸류 기반 멀티스레드 워크로드로 변환하는 것은 상기 제1 파일을 적

어도 하나의 데이터 청크로 분할하는 것을 포함하고, 상기 데이터 청크는 상기 키 밸류 솔리드 스테이트 드라이브의 허용 가능한 밸류 크기 파라미터에 따라 크기가 정해지는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 5

제1항에 있어서,

상기 복수의 데이터 스레드 중 하나는 복수의 컴퓨팅 스레드에 대한 입력 출력 동작을 처리하도록 구성되는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 6

제1항에 있어서,

상기 복수의 데이터 스레드 각각은 관련된 스레드 ID를 갖는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 7

제1항에 있어서,

상기 키 밸류 솔리드 스테이트 드라이브와 함께 상기 스레드-안전 비동기식 키-밸류 입력 동작 및 출력 동작은 일관성 관리 프로토콜을 사용하여 수행되는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 8

제7항에 있어서,

상기 일관성 관리 프로토콜은 MOESI(Modified Owned Exclusive Shared Invalid) 프로토콜인 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 9

애플리케이션 프로그램 인터페이스("API");

동시성 관리자; 및

커널 디바이스 드라이버를 포함하고,

상기 애플리케이션 프로그램 인터페이스는:

파일을 수신하고;

상기 파일을 하나 이상의 데이터 청크로 분할하고;

상기 적어도 하나의 데이터 청크에 키를 할당하고; 및

상기 키를 수신된 파일에 관련시키는 제1 메타데이터 테이블을 생성하도록 구성되고,

상기 동시성 관리자는:

제2 메타데이터 테이블 내에서, 상기 적어도 하나의 데이터 청크에 대응하는 키를 제1 데이터 스레드 ID 및 서브미션 큐 ID와 연관시키고;

상기 키를 사용하여 수행되는 판독 또는 기록 동작의 상태를 추적하기 위해 제1 데이터 스레드 ID 및 서브미션 큐 ID를 모니터하고; 및

상기 키를 사용하여 수행되는 판독 또는 기록 동작의 완료에 기초하여 제2 메타데이터 테이블을 업데이트하고; 및

상기 커널 디바이스 드라이버는 키 밸류 솔리드 스테이트 드라이브에서 병렬로 키 밸류 저장 및 검색을 수행하도록 구성되는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 10

제9항에 있어서,

상기 파일이 블록 솔리드 스테이트 드라이브에 저장되는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 11

제9항에 있어서,

상기 파일은 상기 키 밸류 솔리드 스테이트 드라이브에 저장되는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 12

제9항에 있어서,

상기 파일을 적어도 하나의 데이터 청크로 분할하는 것은:

상기 파일의 크기를 계산하는 것; 및

상기 키 밸류 솔리드 스테이트 드라이브의 크기 및 정렬 사양에 의해 결정된 크기를 갖는 적어도 하나의 데이터 청크로 상기 파일을 분할하는 것을 포함하는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 13

제9항에있어서,

상기 커널 디바이스 드라이버는 복수의 데이터 스레드를 이용하여 상기 키 밸류 솔리드 스테이트 드라이브에 대한 판독 및 기록 동작을 수행하도록 더 구성되며, 각각의 데이터 스레드는 스레드 ID 및 사용자 큐 ID를 갖는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.방법.

청구항 14

제13항에 있어서,

상기 데이터 스레드는 복수의 컴퓨팅 스레드와 병렬로 동작하는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 15

제14항에있어서,

상기 판독 및 기록 동작은 상기 데이터 스레드에 의해 비동기식으로 수행되는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템.

청구항 16

프로세싱 회로에 의해, 제1 스토리지 디바이스에 저장된 파일을 처리하기 위한 명령어을 수신하는 것;

상기 프로세싱 회로에 의해, 상기 제1 스토리지 디바이스로부터 상기 파일을 검색하는 것;

상기 프로세싱 회로에 의해, 상기 파일을 복수의 데이터 청크로 분할하는 것;

상기 프로세싱 회로에 의해, 상기 복수의 데이터 청크 각각에 대한 키를 할당하는 것;

상기 프로세싱 회로에 의해, 상기 키 및 관련 데이터 청크를 적어도 2개의 병렬 데이터 스레드를 사용하는 키 밸류 솔리드 스테이트 드라이브에 저장하는 것; 및

상기 프로세싱 회로에 의해, 적어도 2개의 병렬 데이터 스레드를 사용하는 키 밸류 솔리드 스테이트 드라이브로부터 키 및 관련 데이터 청크를 검색하는 것을 포함하는 병렬화된 애플리케이션을 동작하는 방법.

청구항 17

제16항에 있어서,

상기 프로세싱 회로에 의해, 상기 키를 대응하는 데이터 청크에 링크하는(linking) 제1 메타데이터 테이블을 생성하는 것을 더 포함하는 병렬화된 애플리케이션을 동작하는 방법.

청구항 18

제16항에 있어서,

상기 프로세싱 회로에 의해, 상기 스레드에 연관된 스레드 ID를 사용하는 데이터 스레드, 및 사용자 큐에 연관된 사용자 큐 ID를 사용하는 사용자 큐에 키를 링크하는(linking) 제2 메타데이터 테이블을 생성하는 것을 더 포함하는 병렬화된 애플리케이션을 동작하는 방법.

청구항 19

제16항에있어서,

상기 프로세싱 회로에 의해, 동일한 스레드에 의해 다른 태스크(task)가 수행되기 전에 상기 태스크(task)가 상기 데이터 스레드에 의해 완료되도록 실시함으로써, 상기 데이터 스레드 중 적어도 하나에 할당된 상기 키 중 적어도 하나와 연관된 태스크(task)를 관리하는 것을 더 포함하는 병렬화된 애플리케이션을 동작하는 방법.

청구항 20

제16항에 있어서,

상기 프로세싱 회로에 의해, 적어도 2개의 데이터 스레드와 병렬로 복수의 컴퓨팅 스레드를 동작시키는 것을 더 포함하는 병렬화된 애플리케이션을 동작하는 방법.

발명의 설명

기 술 분 야

[0001] 본 개시는 병렬 컴퓨팅을 위한 키-밸류 기반 시스템 및 병렬화된 애플리케이션을 동작하는 방법에 관한 것이다.

배 경 기 술

[0002] 키 밸류 솔리드-스테이트 드라이브(Key value solid-state drives: KV-SSDs)는 기본적으로 키-밸류 기반 동작을 지원하는 애플리케이션(예: RocksDB)을 위한 키-밸류 저장소의 속도를 높이기 위해 스토리지 디바이스로 사용될 수 있다. 사용되는 일부 고성능 컴퓨팅(High Performance Computing: HPC) 및 머신 러닝(Machine Learning: ML) 애플리케이션은 키-밸류 동작을 위해 설계되기 보다는, 기본적으로 파일에 기반한다. 많은 HPC 및 ML 애플리케이션은, 예를 들어, OpenMP 멀티스레딩 플랫폼을 사용함으로써 지원되는 멀티스레딩을 사용한다. 이러한 애플리케이션은 파일 시스템 및 운영 체제의 블록 계층에 의존하여 메모리 맵핑과 스레드 안전성을 유지한다. 따라서, KV-SSDs를 사용함으로써 제공되는 잠재적인 속도 증가를 활용하는 것과 동시에, HPC 및 ML 애플리케이션 프로그램의 파일-기반 동작을 키-밸류 기반 동작으로 포팅(porting)할 수 있는 애플리케이션 프로그래밍 인터페이스(Application Programming Interface: API)를 갖는 것이 바람직할 수 있다. 또한, KV-SSD를 사용함으로써 제공되는 속도 증가를 구현하는 것 뿐만 아니라, KV-SSD와 같은 SSD를 사용하여 얻을 수있는 데이터 검색과 스토리지 병렬 처리의 가용성을 활용하는 것 또한 유리할 수 있다.

발명의 내용

해결하려는 과제

[0003] 본 개시의 실시예들에 따른 과제는 HPC 및/또는 ML 시스템 내에 구현된 키 밸류 스토리지 시스템(예를 들어, KV-SSDs)을 제공하는 것이다. 보다 구체적으로, 본 개시는 병렬 컴퓨팅을 위한 키 밸류 기반 스토리지 인프라스트럭처를 제공하는 것이다.

과제의 해결 수단

[0004] 본 개시는 HPC 및/또는 ML 시스템 내에 구현된 키 밸류 스토리지 시스템(예를 들어, KV-SSDs)에 관한 것이다. 보다 구체적으로, 본 개시는 병렬 컴퓨팅을 위한 키 밸류 기반 스토리지 인프라스트럭처에 관한 것이다. 이 인프라스트럭처는 운영 체제 커널의 입력/출력 스택에서 중간 계층을 제거하여 애플리케이션 데이터 관리를 단순화한다. 부가적으로, 일부 실시예들에 따르면, 병렬 데이터 스레드들은 수정된 OpenMP #pragma와 같은, 수정된 멀티스레딩 플랫폼 컴파일러 지시문들에 의해 가능한 병렬 컴퓨팅 스레드들과 동시에 구현될 수 있다. 이를 통해 영구 스토리지 내의 데이터에 대한 병렬 컴퓨팅의 세밀한 제어 및 병렬 액세스를 허여할 수 있어, 리소스 활

용도가 향상될 수 있다.

[0005] 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 키 밸류 솔리드 스테이트 드라이브; 및 프로세싱 회로를 포함한다. 상기 프로세싱 회로는: 멀티스레디드 애플리케이션을 실행하고; 키 밸류 병렬 컴퓨팅 시스템을 실행하고; 및 상기 멀티스레디드 애플리케이션과 병렬로 복수의 데이터 스레드를 동작시키도록 구성된다. 상기 데이터 스레드는 상기 키 밸류 솔리드 스테이트 드라이브를 사용하여 병렬 키-밸류 입력 동작 및 출력 동작을 수행하고, 각각의 데이터 스레드는 상기 멀티스레디드 애플리케이션의 적어도 하나의 컴퓨팅 스레드와 연관된다. 상기 키 밸류 병렬 컴퓨팅 시스템은: 멀티스레딩 플랫폼, 멀티스레디드 커널 디바이스 드라이버, 및 키-밸류 동시성 관리자를 포함하고, 상기 키-밸류 동시성 관리자는 상기 키 밸류 솔리드 스테이트 드라이브로 스레드-안전 비동기식 키-밸류 입력 동작 및 출력 동작을 관리하도록 구성된다.

[0006] 상기 병렬 컴퓨팅을 위한 키 밸류 기반 시스템은 상기 키-밸류 동시성 관리자는 복수의 사용자 큐를 관리하도록 더 구성되고, 각각의 사용자 큐는 연관된 사용자 큐 ID를 갖는다.

[0007] 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 멀티스레디드 애플리케이션의 실행이: 파일 기반 멀티스레드 워크로드를 키-밸류 기반 멀티스레드 워크로드로 변환하는 것; 제1 파일 데이터 청크에 대한 제1 밸류를 생성하는 것; 상기 제1 밸류에 대한 제1 키를 생성하는 것; 상기 제1 키를 상기 제1 밸류와 연관시키고 상기 제1 키를 다시 제1 파일에 맵핑하는 제1 메타데이터를 생성하는 것; 및 상기 제1 키를 제1 스레드 ID 및 제1 사용자 큐 ID와 연관시키는 제2 메타데이터를 생성하는 것을 포함하도록 구현될 수 있다.

[0008] 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 파일 기반 멀티스레드 워크로드를 키-밸류 기반 멀티스레드 워크로드로 변환하는 것이 상기 제1 파일을 적어도 하나의 데이터 청크로 분할하는 것을 포함하고, 상기 데이터 청크는 상기 키 밸류 솔리드 스테이트 드라이브의 허용 가능한 밸류 크기 파라미터에 따라 크기가 정해지도록 구현될 수 있다.

[0009] 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 복수의 데이터 스레드 중 하나는 복수의 컴퓨팅 스레드에 대한 입력 출력 동작을 처리하도록 구성되는 것이 구현될 수 있다.

[0010] 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 복수의 데이터 스레드 각각은 관련된 스레드 ID를 갖도록 구현될 수 있다.

[0011] 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 키 밸류 솔리드 스테이트 드라이브와 함께 상기 스레드-안전 비동기식 키-밸류 입력 동작 및 출력 동작은 일관성 관리 프로토콜을 사용하여 수행되도록 구현될 수 있다.

[0012] 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 일관성 관리 프로토콜은 MOESI(Modified Owned Exclusive Shared Invalid) 프로토콜이도록 구현될 수 있다.

[0013] 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 애플리케이션 프로그램 터페이스("API"); 동시성 관리자; 및 커널 디바이스 드라이버를 포함하고, 상기 애플리케이션 프로그램 인터페이스는: 파일을 수신하고; 상기 파일을 하나 이상의 데이터 청크로 분할하고; 상기 적어도 하나의 데이터 청크에 키를 할당하고; 및 상기 키를 수신된 파일에 관련시키는 제1 메타데이터 테이블을 생성하도록 구성되고, 상기 동시성 관리자는: 제2 메타데이터 테이블 내에서, 상기 적어도 하나의 데이터 청크에 대응하는 키를 제1 데이터 스레드 ID 및 서브미션 큐 ID와 연관시키고; 상기 키를 사용하여 수행되는 판독 또는 기록 동작의 상태를 추적하기 위해 제1 데이터 스레드 ID 및 서브미션 큐 ID를 모니터하고; 및 상기 키를 사용하여 수행되는 판독 또는 기록 동작의 완료에 기초하여 제2 메타데이터 테이블을 업데이트하고; 및 상기 커널 디바이스 드라이버는 키 밸류 솔리드 스테이트 드라이브에서 병렬로 키 밸류 저장 및 검색을 수행하도록 구성될 수 있다.

[0014] 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 파일이 블록 솔리드 스테이트 드라이브에 저장되도록 구현될 수 있다.

[0015] 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 파일이 상기 키 밸류 솔리드 스테이트 드라이브에 저장되도록 구현될 수 있다.

[0016] 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 파일을 적어도 하나의 데이터 청크로 분할하는 것이: 상기 파일의 크기를 계산하는 것; 및 상기 키 밸류 솔리드 스테이트 드라이브의 크기 및 정렬 사양에 의해 결정된 크기를 갖는 적어도 하나의 데이터 청크로 상기 파일을 분할하는 것을 포함하도록 구현될 수 있다.

[0017] 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 커널 디바이스 드라이버가 복수의 데이터 스레드를 이용하여 상기 키 밸류 솔리드 스테이트 드라이브에 대한 판독 및 기록 동작을 수행하도록 더 구성되며, 각각의 데이터 스레드는 스레드 ID 및 사용자 큐 ID를 갖도록 구현될 수 있다.

[0018] 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 데이터 스레드는 복수의 컴퓨팅 스레드와 병렬로 동작하도록 구현될 수 있다.

[0019] 상기 병렬 컴퓨팅을 위한 키-밸류 기반 시스템은 상기 판독 및 기록 동작이 상기 데이터 스레드에 의해 비동기식으로 수행되도록 구현될 수 있다.

[0020] 프로세싱 회로에 의해, 제1 스토리지 디바이스에 저장된 파일을 처리하기 위한 명령어를 수신하는 것; 상기 프로세싱 회로에 의해, 제1 스토리지 디바이스로부터 상기 파일을 검색하는 것; 상기 프로세싱 회로에 의해 상기 파일을 복수의 데이터 청크로 분할하는 것; 상기 프로세싱 회로에 의해, 상기 복수의 데이터 청크 각각에 대한 키를 할당하는 것; 상기 프로세싱 회로에 의해, 상기 키 및 관련 데이터 청크를 적어도 2개의 병렬 데이터 스레드를 사용하는 키 밸류 솔리드 스테이트 드라이브에 저장하는 것; 및 상기 프로세싱 회로에 의해, 적어도 2개의 병렬 데이터 스레드를 사용하는 키 밸류 솔리드 스테이트 드라이브로부터 키 및 관련 데이터 청크를 검색하는 것을 포함한다.

[0021] 상기 방법은 상기 프로세싱 회로에 의해, 상기 키를 대응하는 데이터 청크에 링크하는(linking) 제1 메타데이터 테이블을 생성하는 것을 더 포함한다.

[0022] 상기 방법은 상기 프로세싱 회로에 의해, 상기 스레드에 연관된 스레드 ID를 사용하는 데이터 스레드, 및 상기 사용자 큐에 연관된 사용자 큐 ID를 사용하는 사용자 큐에 키를 링크하는(linking) 제2 메타데이터 테이블을 생성하는 것을 더 포함할 수 있다.

[0023] 상기 방법은 상기 프로세싱 회로에 의해, 동일한 스레드에 의해 다른 태스크(task)가 수행되기 전에 상기 태스크(task)가 상기 데이터 스레드에 의해 완료되도록 실시함으로써, 상기 데이터 스레드 중 적어도 하나에 할당된 상기 키 중 적어도 하나와 연관된 태스크(task)를 관리하는 것을 더 포함할 수 있다.

[0024] 상기 방법은 상기 프로세싱 회로에 의해, 적어도 2개의 데이터 스레드와 병렬로 복수의 컴퓨팅 스레드를 동작시키는 것을 더 포함할 수 있다.

발명의 효과

[0025] 본 개시의 실시예들에 따르면, 인프라스트럭처는 운영 체제 커널의 입력/출력 스택에서 중간 계층을 제거하여 애플리케이션 데이터 관리를 단순화한다. 병렬 데이터 스레드들은 수정된 OpenMP #pragma와 같은, 수정된 멀티 스레딩 플랫폼 컴파일러 지시문들에 의해 가능한 병렬 컴퓨팅 스레드들과 동시에 구현될 수 있으며, 이를 통해 영구 스토리지 내의 데이터에 대한 병렬 컴퓨팅의 세밀한 제어 및 병렬 액세스를 허여할 수 있어, 리소스 활용도가 향상될 수 있다.

도면의 간단한 설명

[0026] 도 1은 본 개시의 일부 실시예들에 따른, HPC 키-밸류 API 및 키-밸류 동시성 관리자를 예시하는 병렬 컴퓨팅을 위한 키-밸류 기반 스토리지 인프라스트럭처(Infrastructure)를 도시한다.

도 2는 본 개시의 일부 실시예들에 따른 KV 생성 기능의 예시적인 동작 단계들을 나타내는 흐름도이다.

도 3은 본 개시의 일부 실시들예에 따른 프로그래밍 변경이 발생하는 시스템의 일부를 포함하는, 병렬 컴퓨팅을 위한 키-밸류 기반 스토리지 인프라스트럭처 내의 다양한 계층들을 도시한다.

도 4는 본 개시의 일 실시예에 따른, 키-밸류 기반 병렬 컴퓨팅 인프라스트럭처를 사용하기 위한 예시적인 프로세스를 도시한다..

도 5는 본 개시의 일부 실시예들에 따른, 데이터 스레드 병렬화를 도시한다.

도 6은 본 개시의 일부 실시예들에 따른, KV-SSD를 활용하는 병렬 컴퓨팅 시스템을 동작하기 위한 방법의 예시적인 동작 단계들을 나타내는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0027] 본 발명의 개념 및 이를 달성하는 방법의 특징은 하기의 실시예들의 상세한 설명 및 첨부 도면을 참조하여 보다 쉽게 이해될 수 있다. 이하, 첨부된 도면을 참조하여 실시예들을 보다 상세하게 설명한다. 그러나, 설명된 실시예들은 여러 가지 상이한 형태로 구현될 수 있으며, 여기에서 설명된 실시예에만 한정되는 것으로 해석되어서는 안된다. 오히려, 이들 실시예들이 예시로서 제공됨으로써 본 개시가 철저하고 완전할 것이며, 본 발명 개념의 양태 및 특징을 당해 분야의 통상의 기술자에게 완전히 전달할 수 있다. 따라서, 당해 분야의 통상의 기술자에게 본 발명의 양태와 특징을 완전히 이해하는데 필요하지 않은 프로세스, 구성 요소 및 기술은 설명되지 않을 수있다. 달리 언급되지 않는 한, 유사한 참조 부호는 첨부된 도면 및 기재된 설명 전체에 걸쳐 유사한 구성 요소를 나타내므로, 그에 대한 설명은 반복되지 않을 것이다. 또한, 실시예들의 설명과 관련이 없는 부분들은 설명을 명확하게 하기 위해 도시되지 않을 수 있다. 도면에서, 구성 요소, 계층 및 영역의 상대적인 크기는 명확성을 위해 과장될 수 있다.

[0028] 이하의 설명에서는, 설명의 목적으로, 다양한 실시예들의 철저한 이해를 제공하기 위해 다수의 특정 세부 사항이 제시된다. 그러나, 그러한 다양한 실시예들은 이러한 특정 세부 사항 없이 또는 하나 이상의 동등한 배치로 다양한 실시예들이 실시될 수 있음이 명백하다. 다른 경우에, 공지된 구조 및 디바이스는 불필요하게 다양한 실시예들이 불명료해지는 것을 피하기 위해 블록도 형태로 도시된다.

[0029] 비록 "제1", "제2", "제3" 등의 용어가 본 명세서에서 다양한 요소, 구성 요소, 영역, 계층 및/ 또는 섹션을 설명할 수 있으며, 이들의 요소, 구성 요소, 영역, 계층 및/또는 섹션이 이들 용어에 의해 제한되는 것은 아니다. 이들 용어는 하나의 요소, 구성 요소, 영역, 층 또는 섹션을 다른 요소, 구성 요소, 영역, 층 또는 섹션과 구별하기 위해 사용된다. 따라서, 후술되는 제1 요소, 제1 구성 요소, 제1 영역, 제1 층 또는 제1 섹션은 본 발명의 사상 및 범위를 벗어나지 않으면서 제2 요소, 제2 구성 요소, 제2 영역, 제2 층 또는 제2 섹션으로 지칭될 수 있다.

[0030] 요소, 계층, 영역 또는 구성 요소가 다른 요소, 계층, 영역 또는 구성 요소에 대하여 "위에", "연결되는" 또는 "결합되는" 것으로 언급될 때, 그것은 다른 요소, 계층, 또는 구성요소에 대해 직접 위에 있거나, 연결되거나, 또는 결합되는 것일 수 있고, 또는 하나 이상의 개재되는 요소, 계층, 영역 또는 구성 요소가 존재할 수 있다. 그러나, "직접 연결/직접 결합된"은 중간 구성 요소 없이 다른 구성 요소에 직접 연결되거나 결합되는 하나의 구성 요소를 지칭한다. 한편, "사이", "바로 사이" 또는 "인접한" 및 "바로 인접한"과 같은 구성 요소들 간의 관계를 설명하는 다른 표현들도 마찬가지로 해석될 수 있다. 또한, 요소 또는 계층이 2개의 요소 또는 계층 "사이"인 것으로 언급될 때, 이는 2개의 요소 또는 계층 사이에 유일한 요소 또는 계층일 수 있고, 또는 하나 이상의 개재된 요소 또는 계층이 또한 존재하는 것일 수 있다.

[0031] 본 출원에서 사용된 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥 상 명백하게 다르게 나타나지 않는 한, 복수의 표현을 포함한다. 본 명세서에서 용어 "포함하다", "포함하는", "구비하다" 및 "구비하는"의 용어가 사용될 때, 그 용어들은 언급된 특징, 정수, 단계, 동작, 요소 및/또는 구성 요소의 존재를 지정하는 것으로 이해될 것이다. 하지만, 하나 이상의 다른 특징, 정수, 단계, 연산, 요소, 구성 요소 및/또는 이들의 그룹의 존재 또는 추가를 배제하는 것은 아니다. 본원에 사용된 용어 "및/또는"은 하나 이상의 연관 리스트 항목의 임의의 및 모든 조합을 포함한다.

[0032] 본원에서 사용되는 용어 "실질적으로", "약", "대략" 및 이와 유사한 용어는 정도가 아닌 근사치의 용어로 사용되며, 측정되거나 계산된 값의 고유 편차를 설명하기 위해 의도된 것으로 당해 분야의 통상의 기술자게 인식될 수 있다. 본원에 사용된 "약" 또는 "대략"은, 특정 수량의 측정과 관련된 오류 및 질의(즉, 측정 시스템의 한계) 내에서 측정을 고려하여, 당해 분야의 통상의 기술자에 의해 결정된 특정 값에 대한 허용 가능한 편차 범위 내의 명시된 값 및 평균을 포함한다. 예를 들어, "약"은 하나 이상의 표준 편차 이내, 또는 명시된 값의 ± 30%, 20%, 10%, 5% 이내를 의미할 수 있다. 또한, 본 개시의 실시예들을 설명할 때 "~일 수 있다"의 사용은 "본 개시의 하나 이상의 실시 예들"을 지칭한다. 본 명세서에서 사용되는 용어 "사용하다", "사용하는" 및 "사용된"은 "이용하다", "이용하는" 및 "이용된"으로 각각 간주될 수 있다. 또한, "예시적인"이라는 용어는 예 또는 예시를 지칭하는 것으로 의도된다.

[0033] 특정 실시예가 다르게 구현될 때, 특정 프로세스 순서는 설명된 순서와 다르게 구현될 수 있다. 예를 들어, 2개의 연속적으로 설명된 프로세스는 실질적으로 동시에 수행되거나 설명된 순서와 반대 순서로 수행될 수 있다.

[0034] "프로세싱 회로"라는 용어는 본 명세서에서 데이터 또는 디지털 신호를 처리하는데 사용되는 하드웨어, 펌웨어

및 소프트웨어의 임의의 조합을 의미하는 것으로 사용된다. 프로세싱 회로 하드웨어는, 예를 들어, ASICs(Application Specific Integrated Circuits), 범용 또는 특수 목적의 중앙 처리 장치(CPUs), 디지털 신호 프로세서(DSPs), 그래픽 처리 장치 (GPUs) 및 FPGAs(Field Programmable Gate Arrays)와 같은 프로그램 가능한 로직 디바이스를 포함할 수 있다. 프로세싱 회로에서, 본 명세서에서 사용되는 바와 같이, 각각의 기능은 그 기능을 수행하도록 구성된 하드웨어, 즉, 하드-와이어(hard-wired)에 의해 수행되고, 또는 CPU와 같이, 보다 일반적인 목적의 하드웨어에 의해, 비-일시적인 저장 매체 내에 저장되는 명령어을 수행하도록 구성된다. 프로세싱 회로는 싱글 인쇄 회로 기판(PCB) 상에 제조되거나 여러 개의 상호 연결된 PCB에 분산될 수 있다. 프로세싱 회로는 다른 프로세싱 회로를 포함 할 수있다; 예를 들어, 프로세싱 회로는 PCB 상에 상호 연결된 2개의 프로세싱 회로, FPGA 및 CPU를 포함할 수 있다.

[0035] 본 명세서에 설명된 본 개시의 실시예들에 따른 전자 또는 전기 디바이스 및/또는 다른 관련 디바이스 또는 구성 요소는 임의의 적절한 하드웨어, 펌웨어(예를 들어, 애플리케이션-특정 집적 회로), 소프트웨어, 또는 소프트웨어, 펌웨어 및 하드웨어의 조합을 이용하여 구현될 수 있다. 예를 들어, 이들 디바이스의 다양한 구성 요소는 하나의 집적 회로(IC) 칩 또는 별도의 IC 칩 상에 형성될 수 있다. 또한, 이들 디바이스의 다양한 구성 요소는 플렉서블(Flexible) 인쇄 회로 필름, 테이프 캐리어 패키지(TCP), 인쇄 회로 기판(PCB) 상에 구현되거나, 하나의 기판 상에 형성될 수 있다. 또한, 이들 디바이스의 다양한 구성 요소는, 하나 이상의 컴퓨팅 디바이스 내에, 하나 이상의 프로세서에서 실행되며, 본 명세서에서 설명되는 다양한 기능들을 수행하기 위해 컴퓨터 프로그램 명령어를 실행하고, 다른 시스템 구성요소와 상호 작용하는 프로세스 또는 스레드일 수 있다. 컴퓨터 프로그램 명령어는, 예를 들어, 랜덤 액세스 메모리(RAM)와 같은 표준 메모리 디바이스를 사용하여 컴퓨팅 디바이스에서 구현될 수 있는 메모리에 저장된다. 컴퓨터 프로그램 명령어는 또한, 예를 들어, CD-ROM, 플래시 드라이브 등과 같은 다른 비-일시적 컴퓨터 판독 가능 매체에 저장될 수 있다. 또한, 당해 분야의 통상의 기술자는, 본 개시의 실시예들의 사상 및 범위를 벗어나지 않는 한도에서, 다양한 컴퓨팅 디바이스의 기능이 싱글 컴퓨팅 디바이스에 결합되거나 통합될 수 있고, 또는 특정 컴퓨팅 디바이스의 기능이 하나 이상의 다른 컴퓨팅 디바이스에 걸쳐서 분산될 수 있음을 인식해야한다.

[0036] 다르게 정의되지 않는 한, (기술적이거나 과학적인 용어를 포함하는) 본 명세서에서 사용되는 모든 용어들은 본 발명 개념이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 또한, 일반적으로 사용되는 사전에 정의된 용어와 같은 용어는 관련 기술 및/또는 본 명세서의 맥락에서 그 의미와 일치하는 의미를 갖는 것으로 해석되어야 하고, 본 명세서에서 명시적으로 정의되지 않는 한 이상적이거나 지나치게 형식적인 의미로 해석되어서는 안된다는 것이 이해될 것이다.

[0037] 본 명세서에서 KV-SSDs를 활용하는 키-밸류 기반 병행 컴퓨팅에 대한 시스템이 설명된다. 시스템은 키-밸류 기반 영구 스토리지 디바이스(즉, KV-SSD)를 사용할 수 있게 하는 일련의 소프트웨어 구성요소를 사용할 수 있으며, 병렬화된 컴퓨팅 스레드와 함께 KV-SSD에 대한 병렬 데이터 액세스를 허용하는 추가 소프트웨어 명령어를 추가할 수 있다. 병렬 데이터 액세스는 본 명세서에서 "데이터 스레드(data threads)"로 지칭되는 것에 의해 수행된다. 데이터 스레드는 각 데이터 스레드를 유지하기 위한 ID(IDs) 및 입력/출력 큐 ID(Queue IDs)를 키(Keys) 및 밸류(values)와 연관시키는 키-밸류 동시성 관리자에 의해 유지되는 인프라스트럭처(Infrastructure) 내에서 관리된다. 후술되는 바와 같이, 병렬 데이터 스레드가 스레드-안전 방식으로 동작하는 것을 보장하기 위해 스레드 ID(Thread IDs) 및 큐 ID(Queue IDs)는 동시성 관리자에 의해 유지된다. 이러한 시스템은 고성능 컴퓨팅(high performance computing: HPC) 또는 머신 러닝(Machine Learning: ML) 애플리케이션에서 일반적으로 발생하는 대량의 데이터 처리에서 더 나은 리소스 활용률과 개선된 전체 처리 성능을 제공할 수 있다. 본 명세서에 설명된 예시적인 실시예들은 단지 예시를 위한 것이며, 본 개시의 범위를 제한하는 것은 아니다.

[0038] 도 1에 도시된 바와 같이, 실시예들의 일부 양상들은 병렬 컴퓨팅을 위한 키-밸류 기반 스토리지 인프라스트럭처(100)에 관한 것이다. 이 인프라스트럭처(100)는 파일 기반 병렬 처리 시스템을 키-밸류 기반 병렬 처리 시스템으로 변환하기 위한 애플리케이션 프로그래밍 인터페이스(Application Programing Interface: API)(102)를 포함할 수 있다. 이 병렬 처리는 멀티스레디드 사용자 애플리케이션의 사용을 통해 달성된다. 이러한 애플리케이션은 사용자가 병렬화된 코드를 보다 쉽게 생성할 수 있는 멀티스레딩 플랫폼을 사용하여 구축될 수 있다. OpenMP는 멀티스레디드(multithreaded) 병렬 처리 애플리케이션을 생성하기 위한 멀티스레딩 플랫폼의 예이다. POSIX 스레드와 같은 OpenMP에 대한 대안적인 플랫폼이 본 개시의 범위 내에서 사용될 수 있다는 것이 당해 분야의 통상의 기술자에 의해 이해될 것이다. 참조의 용이성을 위해, 이하의 개시 내용은 주로 OpenMP 플랫폼을 사용하는 실시예를 설명할 것이다. OpenMP에는 #pragmas로 알려진 컴파일링 지시문이 있으며 병렬화된 코드의 실행을 생성하는데 사용할 수 있다. 인프라스트럭처(100)는 또한 병렬 컴퓨팅 스레드의 사용과 동시에 병렬 데

이터 스레드(107)의 사용을 가능하게 하는 수정된 OpenMP #pragma와 같은, 수정된 컴파일러 지시문 세트를 포함할 수 있다. 이러한 수정된 컴파일러 지시문은 KV-SSD(106)로부터 메모리로 데이터의 비동기식(Asynchronous) 병렬 검색을 가능하게 하고, 메모리로부터 KV-SSD(106)로의 데이터 저장을 가능하게한다. 본 명세서에서 사용되는 바와 같이, 용어 "비동기식"는, 데이터가 시스템 메모리로부터 스토리지로 성공적으로 이동되었다는 확인을 기다리는 대신, 영구 스토리지에 기록하기 위해 동작중인 데이터를 큐로 이동한 후에 데이터 스레드가 릴리즈(Released)될 수 있음을 나타내는데 사용된다.(즉, KV-SSD 106)

[0039] 또한, 인프라스트럭처(100)는 시스템이 KV-SSD(106)를 사용하여 동작하는 동안 메모리 맵핑 및 스레드 안전성을 관리하기 위해 운영 체제 내에서 실행되는, 이하에서 동시성 관리자(104)로 지칭되는, 프로세스를 포함할 수있다. 동시성 관리자(104)는 KV-SSD(106)를 사용하는 동안 멀티스레디드 애플리케이션을 실행할 때 메모리 맵핑 및 스레드-안전성을 유지하기 위한 인프라스트럭처일 수 있다. 동시성 관리자(104)는 호스트 시스템에 의해 관리되는 입력/출력(I/O) 서브미션 및 컴플리션 큐(109)를 포함할 수 있다. 서브미션 및 컴플리션 큐(109)와 함께, 동시성 관리자(104)는 판독 요청 또는 저장 요청을 서브미션 큐에 추가하는 일련의 기능을 구현할 수 있다. 대기(queued) 요청의 컴플리션을 추적하고, 컴플리션 큐를 사용하고, 데이터 스레드에 대응하는 스레드 ID 및 큐 ID에 요청을 연관시키는 추가 기능이 구현될 수 있다.

[0040] 보다 구체적으로, 기능 "KV 저장" 및 "KV 검색"은 동시성 관리자(104)에 의해 구현되어 호스트 메모리로부터 KV-SSD(106)로의 데이터 흐름을 관리할 수 있으며, 그 역도 마찬가지이다. KV 저장은 KV-SSD(106)에 대한 기록/업데이트를 위한 스레드에 대해 키-밸류 페어(key-value pairs)를 서브미션 큐에 추가할 수 있다. KV 검색은 키-밸류 페어를 찾아서 호스트 메모리 버퍼의 지정된 부분으로 반환할 것이다. 메모리 버퍼 할당의 동작은 사용자 계층에서 명시적으로 처리될 필요가 있다(즉, 사용자에 의해 멀티스레디드 애플리케이션으로 프로그래밍될 필요가 있다). KV 저장 및 KV 검색의 데이터의 스트림에 대한 메모리 버퍼 할당은 서브미션 큐 깊이(depth) 및 각 큐의 병렬 데이터 스레드의 수를 고려하여 수행될 것이다. 예를 들어, key_buf 및 val_buf를 포함하는 “kv_data” 구조의 경우, 다음 커맨드가 필요한 메모리를 예약한다: cmd_data = valloc(sizeof(struct kv_data) * qdepth * iothread).

[0041] 동시성 관리자(104)는 또한 비동기식 I/O의 스레드-안전성을 보장하기 위해 발행된 I/O의 수, 대기(queued) I/O의 수, 및 각 데이터 스레드 및 서브미션 큐에 의해 완료된 I/O의 수를 추적하는 카운터를 구현한다. 이들 카운터 및 대응하는 서브미션 및 컴플리션 큐(109)는 임의의 완료된 비동기식 I/O가 존재하면 업데이트될 수 있다. 런타임 동안, 각 스레드에 의해 완료된 I/O의 수는 I/O를 발행한 동일한 스레드 내에서 "KV Get IOevent" 기능을 호출하여 계측될 수 있다. 특히, 비트맵은 모든 데이터에 대한 스레드 ID 및 서브미션 큐 ID에 키의 맵핑을 기록하도록 유지될 것이다. 따라서, 일련의 I/O 요청이 서브미션 큐에 들어가면, 이 비트맵이 업데이트 되고, 서브미션 큐의 빈 자리를 유지하는 카운터가 줄어들 것이다. 컴플리션 큐를 폴링하면, KV 검색 기능은 비트맵을 사용하여 완료된 요청이 속하는 서브미션 큐 ID 및 스레드 ID를 식별한 다음, 대응되는 컴플리션 큐의 빈 자리를 증가시킬 것이다. 그 후, 검색된 데이터는 추가 컴퓨팅을 위해 프로세싱 회로(processing circuit)로 전달될 수 있다. 새로운 I/O 요청은 그것의 빈 자리에 따라 각각의 서브미션 및 컴플리션 큐(109)에 들어갈 수 있다. 동시에 서브미션 큐에 제출될 수 있는 최대 I/O 수는 큐 깊이(depth)와 그 서브미션 큐와 연관된 병렬 데이터 스레드 수를 곱한 값과 동일하다.

[0042] 위에서 언급한 바와 같이, 본 발명의 일부 실시예들은 파일-기반 애플리케이션을 키-밸류 기반 애플리케이션으로 변환하기 위한 HPC 키-밸류 API(102)에 관한 것이다. 이 변환은 HPC 키-밸류 API(102)에 의해 수행되는 것으로 도 1에 도시되어 있다. 이를 통해 HPC 시스템이 KV-SSD(106) 사용과 관련된 속도 향상을 활용할 수 있게 한다. 일부 실시예들은 블록 SSD(도시되지 않음) 및 KV-SSD(106) 모두를 포함할 수 있으며, 이러한 구성은 본 명세서에서 "하이브리드" 시스템으로 지칭된다.

[0043] 하이브리드 시스템과 관련하여, HPC 키-밸류 API(102)의 동작은 도 4에 도시된 병렬 컴퓨팅 인프라스트럭처의 실시예를 구현하는 예시적인 프로세스로 전환함으로써 더 잘 이해될 수 있다. 하이브리드 시스템을 사용하는 경우, 사용자는 HPC 또는 ML 애플리케이션에 의해 처리될 데이터 모음(예를 들어, jpeg 파일에 저장된 이미지)을 선택할 수 있고(단계 402), 그 데이터는 블록 디바이스(예를 들어, 블록 SSD)에 저장될 수 있다. 그 후, 선택된 파일은 블록 디바이스로부터 검색된다(단계 404). 선택된 파일은 전-처리 단계(단계 406)를 거칠 수 있으며, 예를 들어, jpeg 파일은 캡슐화된 포스트스크립트(Encasulated PostScript: EPS)로 변환될 수 있으며, 이는 이후에 인간-판독가능 벡터 포맷으로 변환될 수 있다. 그 후, 인간-판독가능 벡터 포맷 파일은 "프레임 5 차원 데이터"(즉, 5-차원 벡터 데이터)를 포함하는 파일로 (예를 들어, 파이썬을 사용하여) 변환될 수 있다. 각각의 이와 같은 파일은 5-차원 벡터의 세트를 포함하고, 각각은 이미지의 하나의 픽셀을 나타내고, 각각의 5-차원 벡터는

적색, 녹색 및 청색 값을 지정하는 3개의 성분과 X 및 Y 좌표를 지정하는 2개의 성분을 갖는다. 이 유형의 벡터 파일은, 예를 들어, K-평균 클러스터링 알고리즘과 같은 HPC 애플리케이션 또는 ML 애플리케이션에 의해 처리될 수 있다. 당해 분야의 통상의 기술자는 파일에 대한 임의의 적합한 유형의 전-처리가 데이터가 원하는 애플리케이션을 위해 사용 가능한 포맷이도록 보장하는데 이용될 수 있음을 이해할 것이다. 일부 실시예들에서, 데이터는 전-처리의 필요없이 블록 SSD 상에 저장될 수 있고, 이 경우 전-처리 단계는 생략될 수있다. 다른 실시예들에서, 처리되지 않은 데이터는 KV-SSD(106)에 저장될 수 있다.

[0044] 다른 실시예들에서, 블록 SSD는 시스템으로부터 생략될 수 있고, 처리되지 않은 파일들은 KV-SSD(106)에 저장된다. 이러한 실시예에서, 하이브리드 시스템에서 하이브리드 시스템의 블록 SSD와 연관될 수 있는 그러한 저장 및 검색 기능은, KV-SSD(106)에 의해 수행될 수 있다.

[0045] 데이터가 사용 가능한 포맷이되면, HPC 키-밸류 API(102)는 처리를 위해 요청된 파일을 포함할 수 있는 파일 기반 멀티스레드 워크로드를 키-밸류 기반 멀티스레드 워크로드로 변환할 수 있다.(단계 408). HPC 키-밸류 API(102)는 "KV 생성" 및 "KV 맵" 기능으로 지칭되는 한 쌍의 기능에 대한 호출을 통합함으로써 멀티스레디드 애플리케이션으로 통합될 수 있다. 그 후, 키-밸류 데이터는 본 명세서에서 "KV 저장"으로 지칭되는 동작을 사용하여 KV-SSD에 저장될 수 있다(단계 410). KV-SSD에 저장되면, 데이터는 본 명세서에서 "KV 검색"으로 지칭되는 동작을 사용하여 처리를 위해 검색될 수 있다(단계 412). 그 후, 예를 들어 K-평균 클러스터링 알고리즘을 사용하는, 데이터의 처리가 수행될 수 있고(단계 416), 그런 다음 처리된 데이터는 블록 디바이스 또는 KV-SSD에 다시 저장될 수 있다(단계 414).

[0046] 요청된 파일을 키와 연관될 수 있는 밸류로 변환하는 기능을 전술한 "KV 생성"으로 지칭하며, 도 2의 흐름도에 도시되어있다. 이 기능은 처리 요청을 수신하고(단계 202) 요청된 파일을 검색함(단계 204)으로써 달성된다. 그 후, 프로세스는 KV-SSD에 기록될 데이터의 유형에 대한 네임 스페이스가 이미 존재하는지를 결정하는 단계(단계 206)로 진행하는데, 각 네임 스페이스는 데이터의 유형에 대응하기 때문이다. KV-SSD(106)에 저장될 데이터가 연관된 네임 스페이스를 KV-SSD(106)에 이미 가지고 있다면, 네임 스페이스 생성의 단계는 생략될 수 있다. 데이터 유형과 연관된 네임 스페이스가 아직 존재하지 않는다면, HPC 키-밸류 API (102)는 데이터에 대한 네임 스페이스를 KV-SSD(106)에 생성한다(단계 207). 예를 들어, 전술한 바와 같이, jpeg 이미지 내의 픽셀에 대응하는 5-차원 벡터 데이터가 저장된다면, 그 유형의 데이터에 대응하는 네임 스페이스가 생성될 것이다. 예를 들어, 프레임 5-차원 데이터의 모음을 처리한 K-평균 클러스터링 알고리즘에 의해 생성된 클러스터 데이터가 저장되는 경우, KV-SSD(106)에 저장되는 데이터의 유형 사이의 구별을 유지하기 위해 추가적인 네임 스페이스를가 생성될 수 있다.

[0047] 그 후, 개별 파일로부터의 데이터는 KV-SSD(106)의 크기 및 정렬 사양에 기초하여 KV-SSD(106)에 저장될 하나 이상의 밸류로 변환될 수 있다. 이 단계는 KV-SSD(106)의 밸류 크기 파라미터에 기초하여 큰 파일 크기를 가질 수 있는 파일을 청크(chunks)로 분할하는 것(단계 208)을 포함한다. 그 후, 각 데이터 청크(chunks)가 밸류로 처리될 수 있다. 각각의 새로운 밸류에 대한 키가 생성되고, 각 키는 밸류에 할당된다(단계 210). 이러한 대응하는 밸류에 대한 키의 할당은 제1 메타데이터 테이블에 저장된다. 이하에서, 이 동작을 "KV 맵"이라 지칭한다. 아래에서 더 상세히 논의될 바와 같이, KV-SSD에 대해 키-밸류 페어의 병렬화된 스토리지를 용이하게 하기 위해 각각의 키를 대응하는 스레드 ID 및 큐 ID에 더 연관시키는 KV 맵 기능에 의해 제2 메타데이터 테이블이 또한 생성된다(단계 212). 그 후, KV 저장 기능을 사용하는 페어링된 키 및 밸류는 네임 스페이스를 사용하여 KV-SSD(106)에 저장될 수 있다(단계 214). 그 후, 키-밸류 데이터는 영구 스토리지 디바이스(즉, KV-SSD)에 저장된다(단계 216).

[0048] 비-제한적인 예로서, 아래의 목록 1의 C ++코드는 KV 생성 및 KV 맵에 의해 수행되는 프로세스를 더 잘 설명하는 사용자 애플리케이션의 일부의 예이다. KV 생성 및 KV 맵 함수가 호출된 후, 목록 1의 C++ 코드는 "a" 배열에 저장된 데이터를 페치(fetch)하기 위해 "opcode" 파라미터를 사용하는 동작 유형을 "검색"으로 설정한다. 다음으로, 어레이 "a"의 특정 인덱스에 저장된 데이터를 페치(fetch)하기 위해 필요한 메모리가 할당된다. 마지막으로, 제곱이 계산된다. 목록 1의 예시적인 C++ 코드는 #pragmas를 사용하여, 행 11과 12에서, 컴파일러에게 for 루프(for loop)에 대한 연산을 (병렬로) 실행하도록 병렬 컴퓨팅 스레드와 병렬 데이터 스레드를 생성하는 것을 지시한다.

[0050] 1 char data = "application_data";

[0051] 2 kvcreate(*namespace, key_len, data_len, *key_addr, *data_addr, data, cmd_type);

[0052] 3 kvmap(*map, *namespace, data, *keyid, *key_addr);

[0053] 4 int* num_points;

[0054] 5 cmd_ctx[keyid[num_points]].cmd.opcode = nvme_cmd_kv_retrieve;

[0055] 6 num_points = (int*)cmd_ctx[keyid[num_points]].cmd.data_addr;

[0056] 7 int thread_no, init_thread_no = 2, max_thread_no = omp_get_max_threads(), *square, iter, *a;

[0057] 8 omp_set_num_threads(thread_no);

[0058] 9 square=(int*)malloc(sizeof(int)*num_points);

[0059] 10 a=(int*)malloc(sizeof(int)*num_points);

[0060] 11 #pragma omp parallel default(shared)

[0061] 12 #pragma omp for

[0062] 13 for (iter=1; iter<=num_points; iter++)

[0063] 14 {

[0064] 15 cmd_ctx[keyid[a[iter]]].cmd.opcode = nvme_cmd_kv_retrieve;

[0065] 16 a[iter] = (int*)cmd_ctx[keyid[a[iter]]].cmd.data_addr;

[0066] 17 square[iter] = a[iter]*a[iter];

[0067] 18 }

[0068] 19 free(square);

[0069] 20 square = NULL;

[0070] 목록 1

[0071] 목록 1의 코드에서, 컴파일러는, for 루프(for loop)의 각 반복에 대해, 행 16의 데이터 연산을 각각의 데이터 스레드에 할당할 수 있고, 컴파일러는 행 17의 계산을 각각의 컴퓨팅 스레드에 할당할 수 있고, 또는 이러한 스레드는 루프의 여러 반복을 처리할 수 있다. 각 스레드에 할당하는 반복 횟수는 런타임에서 결정된다. 예를 들어, 하나의 데이터 스레드는 for 루프(for loop)의 5회 반복에 대해 행 16을 처리할 수 있고, 하나의 컴퓨팅 스레드는 for 루프(for loop)의 10회 반복에 대해 행 17을 처리할 수 있다.

[0072] 도 3의 병렬 컴퓨팅을 위한 인프라스트럭처 내의 다양한 계층의 묘사로 넘어가면, 일부 실시예들은 멀티스레디드 애플리케이션(110)과 함께 사용하기 위한 공유 메모리 스페이스 및 스레딩의 활용을 위한 OpenMP 런타임 라이브러리(108) 및 운영 체제("OS") 서포트로의 변경에 관한 것이다. 일부 실시예들은 병렬 컴퓨팅 및 데이터 스레드(107)의 사용을 가능하게 하는, 컴파일러 지시문의 서브 세트, 예를 들어, OpenMP #pragmas, 의 수정에 관한 것이다. 예를 들어, OpenMP #pragma “omp parallel” 및 “omp for”에 대한 수정은 (i)병렬 데이터 스레드를 생성 및 인식할 수 있는 코드와 (ii) 오류 처리 코드를 통합하여 이루어진다. POSIX 스레드와 같은 대안적인 멀티스레딩 플랫폼의 컴파일러 지시문에 대해 유사한 수정이 이루어질 수 있음을 당해 분야의 통상의 기술자는 이해할 것이다. 이러한 병렬 데이터 스레드(107)의 예들는 컴퓨팅 스레드로부터 I/O 서브미션 및 컴플리션 큐(109)로 향하는 화살표로서 도 1에 도시된다. 병렬 데이터 스레드의 다른 예시적인 모습이 도 5에 나타날 수 있다. 이는 (왼쪽에서 오른쪽으로): 싱글 컴퓨팅 스레드(501)를 제공하는 싱글 데이터 스레드; 멀티플 컴퓨팅 스레드(502)를 제공하는 싱글 데이터 스레드; 및 멀티플 컴퓨팅 스레드(503)를 각각 제공하는 멀티플 병렬 데이터 스레드의 동작을 설명한다.

[0073] OpenMP 런타임 라이브러리 내의 #pragmas와 같은 컴퓨팅 스레드의 병렬화와 관련된 멀티스레딩 플랫폼의 컴파일러 지시문을 구성하는 코드의 각 행에 대해, 데이터 스레드의 인식 및 활용을 허여하도록 유사한 코드 행이 삽입될 수 있다. 이는 멀티스레딩 플랫폼 내에서 수정된 컴파일러 지시문의 사용을 허여하고, 병렬 데이터 스레드의 생성을 허여할 수 있다. 목록 1의 C++을 다시 참조하면, 수정된 #pragmas의 사용은 도 5에 따라 시각화 될

수 있다. 행 11과 12는 for 루프(for loop)의 실행을 병렬화하기 위한 #pragma가 포함된다. 이와 같은 #pragma의 사용이 없으면, 코드는 도 5의 501 도시되는 싱글 스레드 예시에 따라 실행될 것이다. 병렬 데이터 스레드의 생성을 가능하게 하기 위해 컴파일러 지시문이 수정되지 않은 실시 예에서, 행 11 및 12에서 #pragma의 사용은, 도 5의 502에서 보여지는 예시와 같이, 컴퓨팅 태스크(task)의 병렬 실행을 가능하게 한다. 병렬 데이터 스레드의 생성을 가능하게 하도록 컴파일러 지시문이 수정된 실시예에서, 행 11 및 12에서 #pragma의 사용은 컴퓨팅 태스크(task)의 병렬 실행을 가능하게 하고, 도 5의 503에 도시되는 예시와 같이, 데이터 태스크(task)의 병렬 실행(예를 들어, 각각의 데이터 스레드에서)을 가능하게 한다.

[0074] 단순성을 위해 목록 1에서 생략된 에러 핸들링 코드는 (i) 병렬 데이터 스레드의 가용성 및 (ii) 키 밸류 기반의 영구 스토리지를 이용하도록 멀티스레디드 애플리케이션(110)에 대한 변경에 의해 도입될 수 있는 에러를 처리(예를 들어, 보고)하기 위해 도입될 수 있다. 당해 분야의 통상의 기술자가 이해할 수 있는 바와 같이, 에러를 검출하는 것과 처리하는 것(예를 들어, 보고하는 것)을 위한 임의의 적절한 에러 처리 방법론이 본 발명의 범위 내에서 이용될 수 있다.

[0075] 블록 디바이스(예를 들어, 블록 SSD)로만 동작하는 시스템에서, 파일 시스템은 애플리케이션 데이터의 메모리 맵핑을 위해 사용될 수 있고, 블록 계층은 블록 디바이스에 대한 스레드-안전 비동기식 입력/출력을 위해 멀티스레디드 애플리케이션(110)에 의해 사용될 수 있다. 그러나, 본 개시의 일부 실시예들은 파일 시스템 및 블록 계층이 제거되는 KV-SSD(106)의 사용에 관한 것이다. 이와 같은 디자인의 장점은 당해 분야의 통상의 기술자에게 이해될 것이다. 파일 시스템 및 블록 계층의 제거는, 비동기식 병렬 데이터 요청이 메모리 맵핑 및 스레드 안전성을 위해 파일 시스템 및 블록 계층을 사용할 수 있기 때문에, HPC 병렬 처리 기술의 사용에 대한 문제를 발생시킨다. 이는 비동기식 동작이 스레드가 릴리즈(released) 되는 이후의 시간을 포함하는 영구 스토리지 디바이스(즉, KV-SSD(106)) 내의 스레드-안전 방식으로 수행되어지는 것을 보장하도록 병렬 데이터 스레드에 의해 수행되는 비동기식 동작의 상태를 모니터할 수 있는 모듈에 대한 필요성을 발생시킬 수 있다. 따라서, 일부 실시예들은 KV-SSD(106)을 사용하는 시스템 내에서 멀티스레딩 플랫폼을 사용하는(멀티스레디드 애플리케이션 110(도 3)과 같은) 애플리케이션에 대한 파일 시스템 및 블록 계층의 기능을 대체하는 키-밸류 동시성 관리자(도 1의 104) 모듈에 관한 것이다.

[0076] 수정된 컴파일러 지시문은 또한 운영 체제 커널 디바이스 드라이버(Kernel Device Driver: KDD)에 통합되어 OpenMP에 의해 인에이블(enabled) 된 병렬 컴퓨팅 스레드와 동시에 실행될 수 있는 병렬 데이터 스레드의 생성을 허여할 수 있다. 이러한 통합은 병렬화된 컴퓨팅 태스크(task)와 동시에 병렬 판독/기록 태스크(task)를 허용할 수 있다. (i) (OpenM과 같은) 멀티스레딩 플랫폼, (ii) 멀티스레디드 커널 디바이스 드라이버, 및 (iii) 키-밸류 동시성 관리자의 조합은 본 명세서에서 "키 밸류 병렬 컴퓨팅 시스템"으로 지칭될 수 있다. 동시에 발생하는 동작 중에, 병렬 데이터 스레드의 각각은 하나 이상의 컴퓨팅 스레드와 연관된다.

[0077] 각각의 데이터 스레드는 자체 입출력("I/O") 서브미션 및 컴플리션 큐를 가질 수 있다. 다시 도 1로 돌아와서, 이들 큐는 도 1에서 기어(109)로 도시된다. 큐의 깊이(depth)를 예시하는 한 쌍의 서브미션 및 컴플리션 큐(109)의 추가적인 표시가 도 1 내의 기어(109) 아래에 도시된다. 이 큐는 데이터 스레드가 연관된 컴퓨팅 스레드에 할당된 컴퓨팅 태스크(task)를 완료하는데 필요한 판독/기록 동작을 명령하는 역할을 한다. 각 스레드는 관련 스레드 ID를 가질 수 있다. 큐 깊이(depth) 및 컴플리션 속도에 따라 각 스레드에 대한 서브미션 큐는 FIFO(First In First Out) 정책으로 다시 채워질 수 있다.

[0078] 이용된 병렬 데이터 스레드의 수는, 일부 실시예들에서, 사용자에 의해 특정될 수 있다. 다른 실시예들에서, 이용된 데이터 스레드의 수는 멀티스레디드 애플리케이션(110)에 의해 자동으로 할당될 수 있다. 이러한 자동화는 멀티스레디드 애플리케이션(110)의 최적의 병렬화를 달성하기 위해 데이터 스레드의 수를 조정하도록 대역폭 활용율 및 CPU 성능을 고려할 수있다. 병렬 데이터 스레드에 의해 제공되는 추가적인 병렬 처리는, 병렬 데이터 스레드에 의해 제공되는 추가적인 기능을 추가함으로써 특정 동작에 대한 병렬화의 최적의 레벨을 찾을 수 있다는 점에서, 병렬화를 보다 세밀하게 제어하는 것을 허여할 수 있다. 병렬화에 대한 이와 같은 추가적인 제어는, 일부 실시예들에서, 개선된 시스템 리소스 활용율과 개선된 CPU 및 메모리 파이프라이닝을 야기할 수 있다.

[0079] 도 1에 도시된 바와 같이, 일부 실시예들에서, 키-밸류 동시성 관리자(104)는 멀티스레디드 애플리케이션(110)(도 3)에 의해 의존되는 파일 시스템 및 블록 계층의 기능을 대체하기 위해 "KV 저장", "KV 검색" 및 "KV Get IOevents" 기능을 사용한다. KV 저장 동작은 서브미션 및 컴플리션 큐를 갖는 각각의 스레드, 키-밸류 페어에 대한 복수의 기록 커맨드를 저장하는 서브미션 큐, 및 완료된 커맨드에 대해 복수의 커맨드 컴플리션을 저장하는 컴플리션 큐와 함께, 키-밸류 데이터가 병렬 데이터 스레드를 사용하는 KV-SSD에 저장될 수 있게 한다.

NVMe 시스템에서, 데이터 스레드에 의해 예시된 서브미션 및 컴플리션 큐에 더하여 추가적으로 NVMe IO 큐가 존재할 수 있다. 데이터 스레드에 의해 예시된 서브미션 및 컴플리션 큐는 동시성 관리자(104)에 의해 관리될 수 있으며, 동시성 관리자는 스레드 안전 동작 및 메모리 맵핑을 보장하도록 호스트 시스템(예를 들어, 호스트 시스템의 프로세싱 회로)에서 실행되는 프로세스이다.

[0080] 메모리 맵핑 기능은 도 3에 도시된 병렬 컴퓨팅 인프라스트럭처의 다양한 계층의 모습으로 되돌아감으로써 더 잘 이해될 수 있다. 어떤 파일이 키-밸류 데이터로 변환되어 KV-SSD에 저장되었는지를 추적하는 메모리 맵핑은, 도 3에서 "파일 ↔ 키 맵핑 테이블"(111)로서, 제1 메타데이터 테이블과 함께, 표현되는, 제2 메타데이터 테이블을 통해 달성된다. 이 제2 메타데이터 테이블은 각 키를 해당 밸류가 추출된 파일, 스레드 ID 및 큐 ID와 연관시킨다. 이 메모리 맵핑 기능은 어떤 파일이 처리를 위해 KV-SSD(106)에서 비동기식 병렬 방식으로 변환되고 저장되었는지 추적하는 것을 허용할 수 있다. 또한, 메타데이터 테이블은 각 데이터 스레드를 사용하여 전송되는 키를 능동적으로 모니터링 하는 것을 허용하고, 각 데이터 스레드의 관련 I/O 서브미션 및 컴플리션 큐(109)를 모니터링하는 것을 허용한다. 예를 들어, KV-SSD에 대해 상이한 판독 요청을 발행할 수 있는 2개의 데이터 스레드가있는 실시예에서, 동시성 관리자(104) 없이는 멀티스레딩 플랫폼(예를 들어, OpenMP)은 완료된 판독이 제1 스레드 또는 제2 스레드로부터의 응답인지 알아낼 방법이 없다. 따라서, 추가 실행 시 사용자 애플리케이션이 중단될 수 있는 부정확한 밸류가 반환될 수 있다.

[0081] KV 검색 동작은 데이터 스레드가 KV-SSD(106)에 병렬로 판독 요청하는 것을 허용한다. KV-SSD에서 수행되는 판독 동작의 병렬화는 시스템이, KV-SSD(106)와 같은, 최신 SSD에서 사용 가능한 멀티플 플래시 채널을 사용하도록 한다. KV 검색 동작은 KV 저장 동작과 동일한 데이터 스레드 서브미션 큐에 배치될 수 있으며, 다수의 데이터 스레드에서 수행되는 KV 저장 동작과 병렬로 처리될 수 있다.

[0082] 영구 스토리지(예를 들어, KV-SSD) 내의 데이터의 일관성은 "MOESI"(Modified Owned Exclusive Shared Invalid) 프로토콜에 기초하여 동시성 관리자(104)에 의해 병렬 데이터 스레드의 동작 동안 유지될 수 있으며, "MOESI"(Modified Owned Exclusive Shared Invalid)은 다른 애플리케이션에서, 메인 메모리 및 복수의 캐시를 갖는 시스템에서 캐시 일관성을 보장하기 위해 사용된다. MOSIF의 MOSI와 같은, 임의의 다른 적합한 일관성 프로토콜이 본 발명의 범위 내에서 이용될 수 있다는 것이 통상의 기술자에게 이해될 것이다. 동시성 관리자의 코딩에 의해, MOESI 프로토콜 구현시 공유 메모리 스페이스의 영역에 대한 개별 데이터 스레드의 액세스를 제어하도록, 제한되지는 않으나 예를 들어,<pthread_mutex_lock>, <pthread_cond_wait> 및 <pthread_mutex_unlock>과 같은 조건부 변수가 이용될 수 있다. 예를 들어, 데이터 스레드의 서브미션 큐는 캐시와 유사할 수 있고, KV-SSD의 영구적인 메모리는 메인 메모리와 유사할 수 있다. MOESI 프로토콜이 적용되는 세분성(granularity)은 KV-SSD(106)에 저장된 개별 밸류일 수 있다. 이와 같이, 5개의 이용 가능한 상태들(수정, 소유, 독점, 공유 및 무효) 중 하나의 상태가 임의의 서브미션 큐 내의 각 밸류에 할당될 수 있고, MOESI 프로토콜은, 예를 들어, 임의의 특정 스레드가 해당 큐 내의 임의의 밸류를 수정(즉, 영구 스토리지에 기록)할 수 있는지를 결정하는데 사용될 수 있다.

[0083] The KV Get IOevents 동작은, 일부 실시예들에서, 데이터 스레드와 관련된 서브미션 및 컴플리션 큐의 현재 상태를 얻기 위해 사용될 수 있다. 각각의 데이터 스레드에 대한 I/O 서브미션 및 컴플리션 큐(109)는 모두 이 동작을 사용하여 모니터링될 수 있다. KV Get IOevents 동작은 어떤 파일이 KV-SSD(106)로 처리되고 저장되었는지 추적하는데 사용될 수 있다. 어떤 파일이 처리되고 KV-SSD(106)에 저장되었는지 추적하는 이러한 능력은 인프라스트럭처(100)에 의해 수행된 프로세스가 OpenMP 기반 멀티스레디드 애플리케이션(110)을 사용하는 사용자에게 명료할 수 있도록 한다. 일부 실시예들에서, 사용자는 단지 수정된 컴파일러 지시문에 의해 얼마나 많은 병렬 데이터 스레드가 생성되었는지를 알면 된다. 다른 실시예들에서, 병렬 데이터 스레드 수가 자동으로 할당되는 경우, 사용자는 KV-SSD 병렬 컴퓨팅 인프라스트럭처의 기능에 관한 어느 지식 없이도 멀티스레디드 애플리케이션을 이용할 수 있다.

[0084] 따라서, 사용중인 인프라스트럭처(100)는 멀티스레디드 애플리케이션(110)의 보다 세분화된 병렬화를 가능하게 하는 병렬 데이터 스레드의 활용율을 제공하면서, 또한 KV-SSD(106)와 함께 병렬 스레드의 스레드 안전 사용을 허용하는데 필수적인 모니터링 구성요소를 제공한다. 멀티스레디드 애플리케이션(110)의 파일 기반 동작을 키-밸류의 사용으로 포팅(port)하기 위해 HPC 키 밸류 API(102)가 제공된다. 이는, OpenMP #pragmas와 같이, 컴파일러 지시문의 수정을 통한 멀티스레디드 애플리케이션(110)의 추가적인 병렬 처리를 가능하게 함에 의해 더 활용되는 속도 향상을 제공하고, 동시성 관리자(104) 모듈의 포함을 제공할 수 있다.

[0085] 인프라스트럭처(100)의 사용은 도 6의 맥락에서 추가로 이해될 수 있다. 우선, 처리 요청은 프로세싱 회로에 의

해 수신된다(단계 602). 이러한 요청은 OpenMP 플랫폼을 사용하여 구축된 애플리케이션과 같은 멀티스레디드 애플리케이션(110)의 사용자에 의해 개시될 수 있다. 요청에 따라, 일부 실시예들에서 블록 디바이스에 저장된 파일이 검색된다(단계 604). 그 후, KV 생성 기능을 사용하여, 검색된 파일은 데이터 청크(즉, 밸류)로 분할된다(단계 606). 제1 메타데이터 테이블이 생성된다.(단계 608) 그 후, 각각의 키에는 제1 메타데이터 테이블 내의 밸류가 할당된다(단계 610). 또한, 키를 스레드 ID 및 큐 ID에 링크(linking)하는 제2 메타데이터 테이블이 생성된다(단계 612). 그 후, 키-밸류 페어는 적어도 2개의 병렬 데이터 스레드(107)를 사용하여 KV-SSD(106)에 저장될 수 있다(단계 614). 이는 병렬 데이터 스레드(107)와 병렬인 복수의 컴퓨팅 스레드의 동작을 포함할 수 있다(단계 616). 병렬 데이터 스레드에 의해 수행된 태스크(task)는 동시성 관리자(104)에 의해 관리될 수 있다(단계 618). KV-SSD(106)에 저장된 후, 키-밸류 페어는 적어도 2개의 병렬 데이터 스레드(107)를 사용하여 KV-SSD(106)로부터 검색될 수 있다(단계 620). 이는 KV-SSD(106)의 사용과 관련된 속도 증가를 활용하면서 멀티스레디드 애플리케이션(110)의 병렬화를 세밀하게 제어할 수 있게 한다.

부호의 설명

[0087] 100: 인프라스트럭처

102: HPC 키-밸류 API

104: 동시성 관리자

106: KV-SSD

107: 데이터 스레드

108: OpenMP 런타임 라이브러리

109: 서브미션 및 컴플리션 큐

110: 멀티스레디드 애플리케이션

111: "파일 ↔ 키 맵핑 테이블

501: 싱글 컴퓨팅 스레드

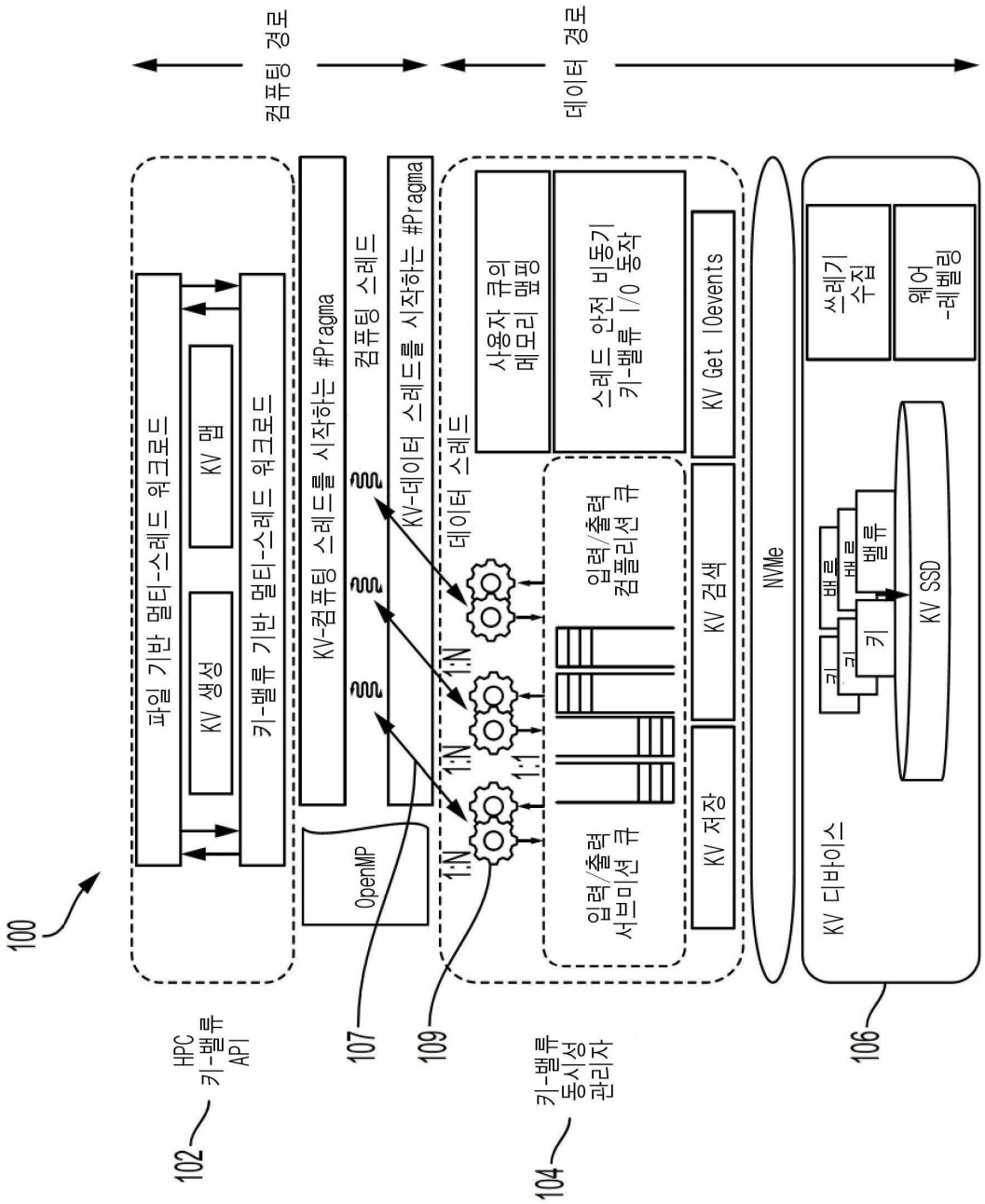
502: 싱글 데이터 스레드

503: 멀티플 컴퓨팅 스레드

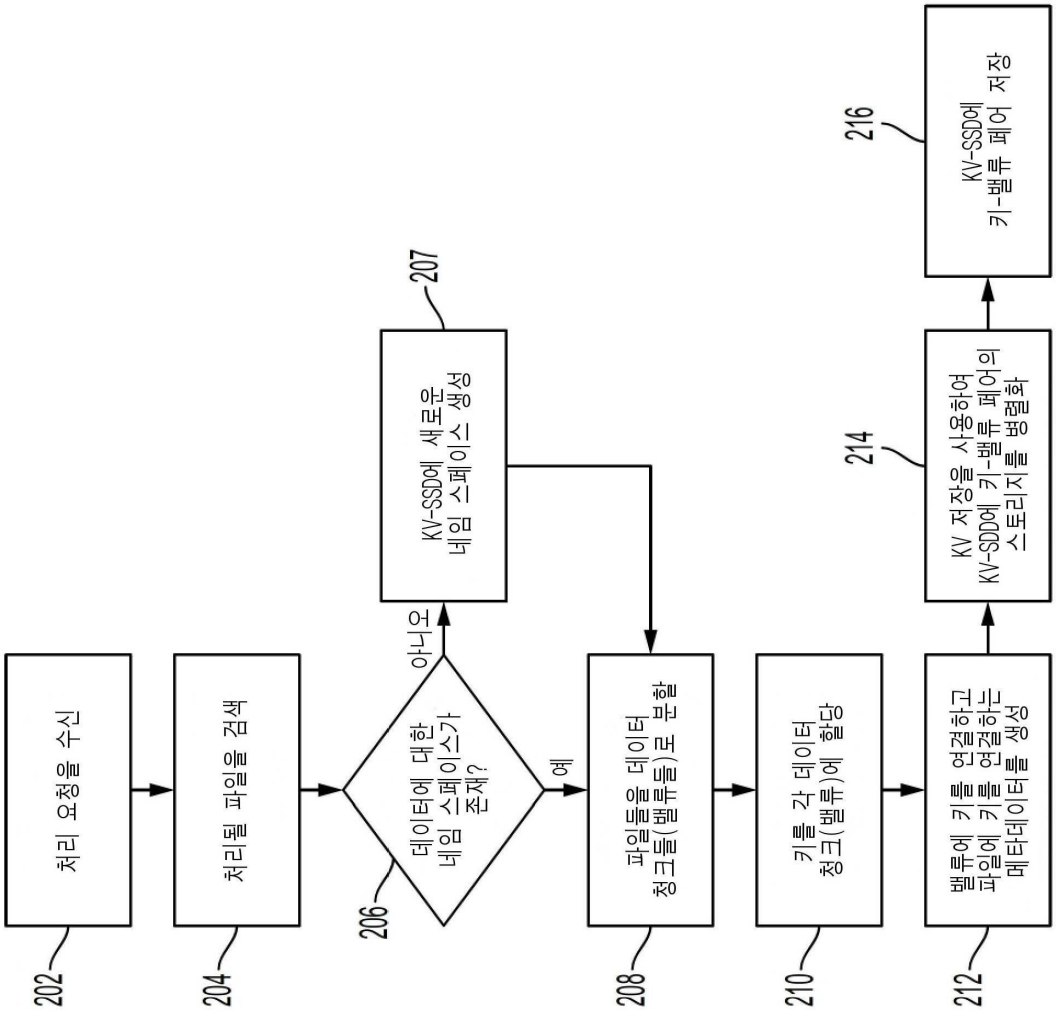
IC: 집적 회로

도면

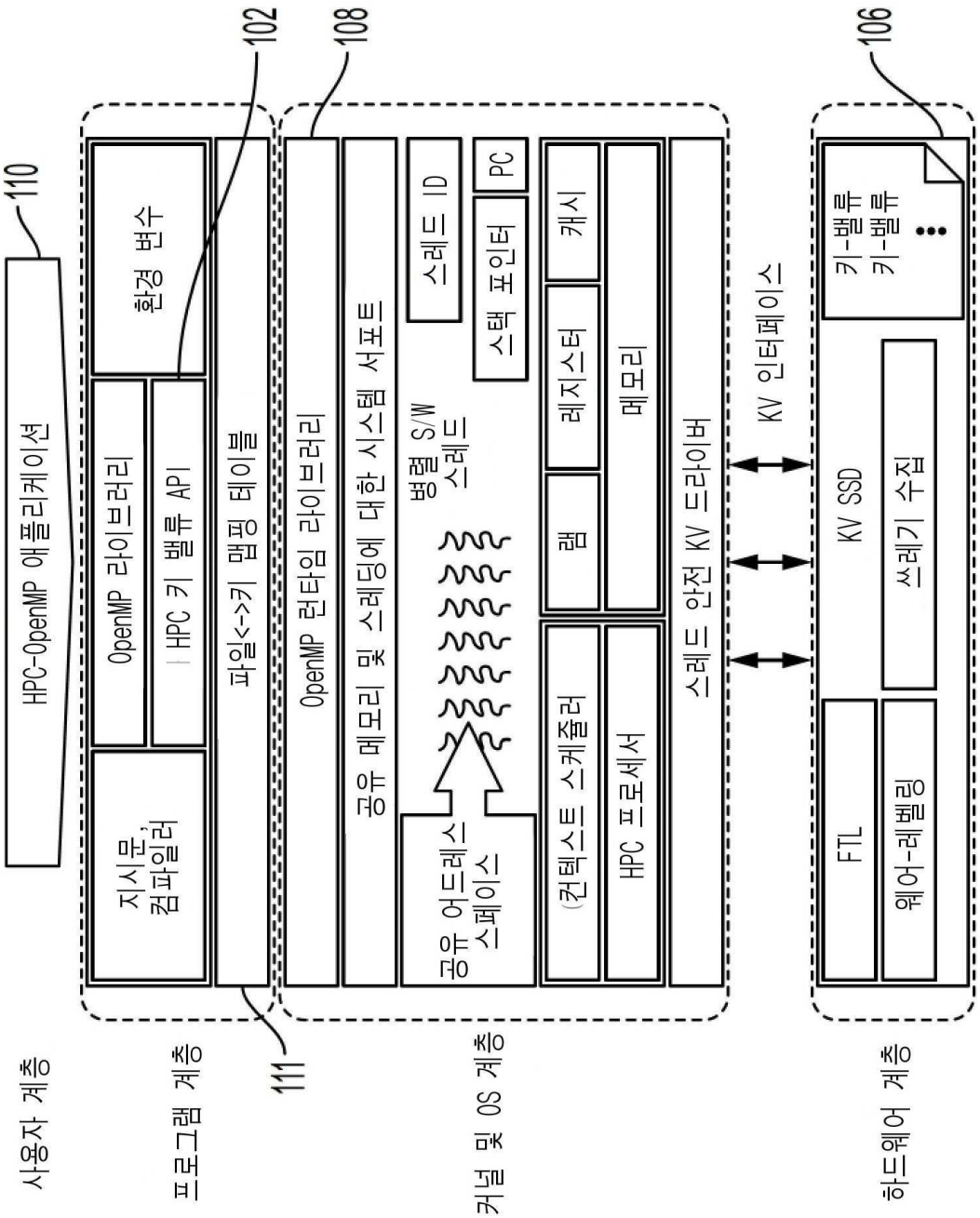
도면1


컴퓨팅 경로
데이터 경로
100
파일 기반 멀티-스레드 워크로드
KV 생성
KV 맵
키-밸류 기반 멀티-스레드 워크로드
KV-컴퓨팅 스레드를 시작하는 #Pragma
OpenMP
컴퓨팅 스레드
KV-데이터 스레드를 시작하는 #Pragma
데이터 스레드
1:N
1:N
1:N
1:1
사용자 큐의 메모리 맵핑
스레드 안전 비동기 키-밸류 I/O 동작
입력/출력 컴플리션 큐
입력/출력 서브미션 큐
KV Get IOevents
KV 검색
KV 저장
NVMe
쓰레기 수집
웨어 -레벨링
KV 디바이스
키
키
키
밸류
KV SSD
HPC 키-밸류 API
102
107
109
키-밸류 동시성 관리자
104
106

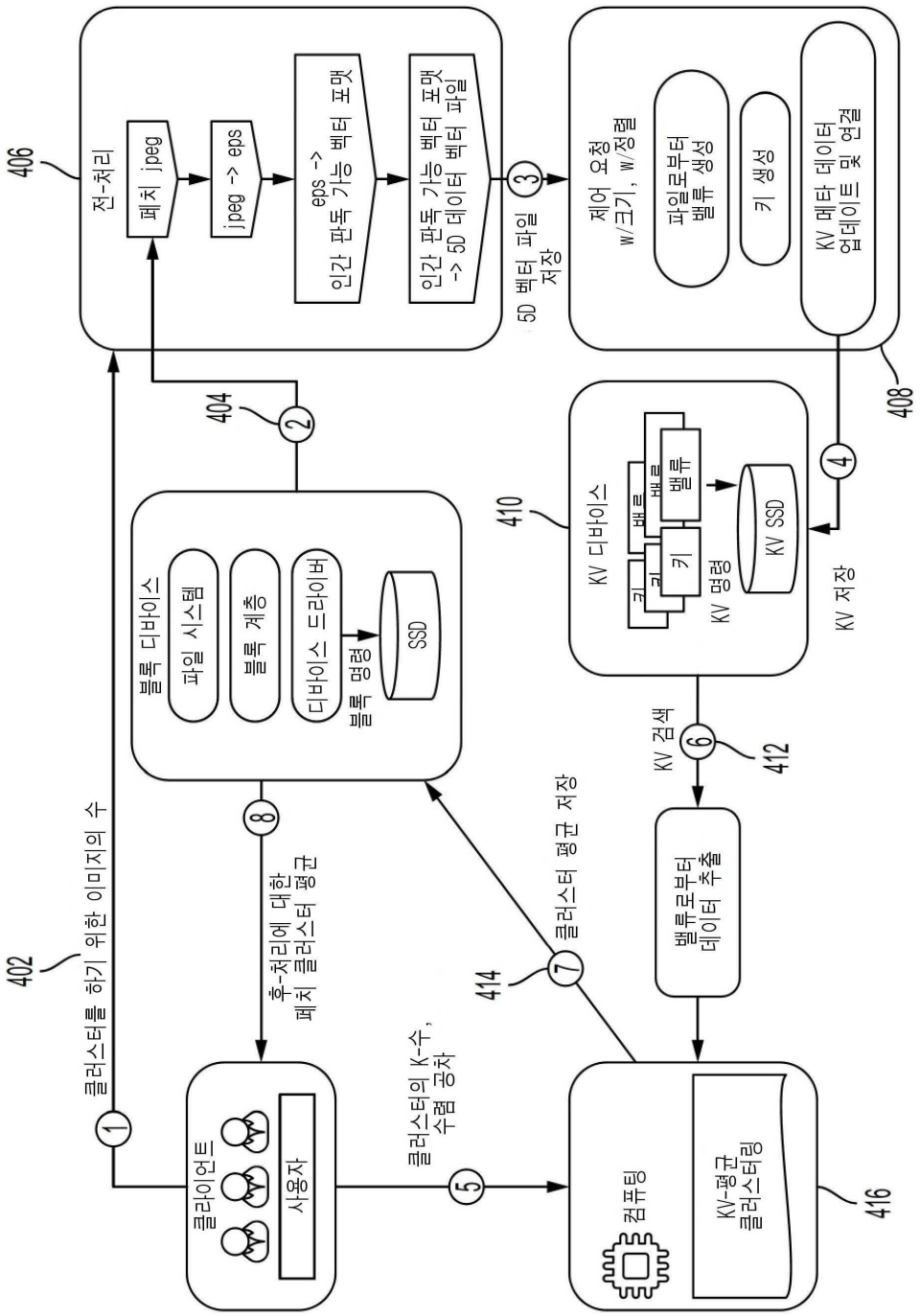
도면2


처리 요청을 수신
202
처리될 파일을 검색
204
데이터에 대한 네임 스페이스가 존재?
206
아니어
예
KV-SSD에 새로운 네임 스페이스 생성
207
파일들을 데이터 청크들(밸류들)로 분할
208
키를 각 데이터 청크(밸류)에 할당
210
밸류에 키를 연결하고 파일에 키를 연결하는 메타데이터를 생성
212
KV 저장을 사용하여 KV-SSD에 키-밸류 페어의 스토리지를 병렬화
214
KV-SSD에 키-밸류 페어 저장
216

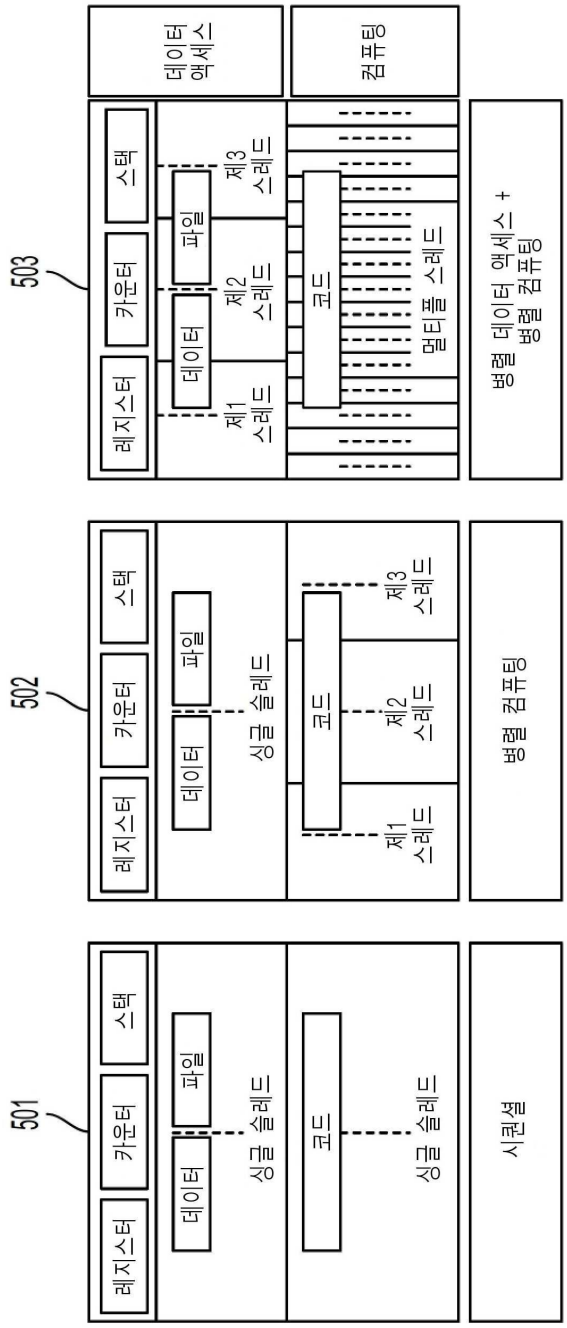
도면3


110
HPC-OpenMP 애플리케이션
102
환경 변수
OpenMP 라이브러리
HPC 키 밸류 API
지시문,
컴파일러
파일<->키 맵핑 테이블
111
108
OpenMP 런타임 라이브러리
공유 메모리 및 스레딩에 대한 시스템 서포트
스레드 ID
스택 포인터
PC
범용 S/W 스레드
공유 어드레스 스페이스
(컨텍스트 스케줄러
HPC 프로세서
램
레지스터
캐시
메모리
스레드 안전 KV 드라이버
KV 인터페이스
106
키-밸류
키-밸류
...
KV SSD
쓰레기 수집
FTL
웨어-레벨링
사용자 계층
프로그램 계층
커널 및 OS 계층
하드웨어 계층

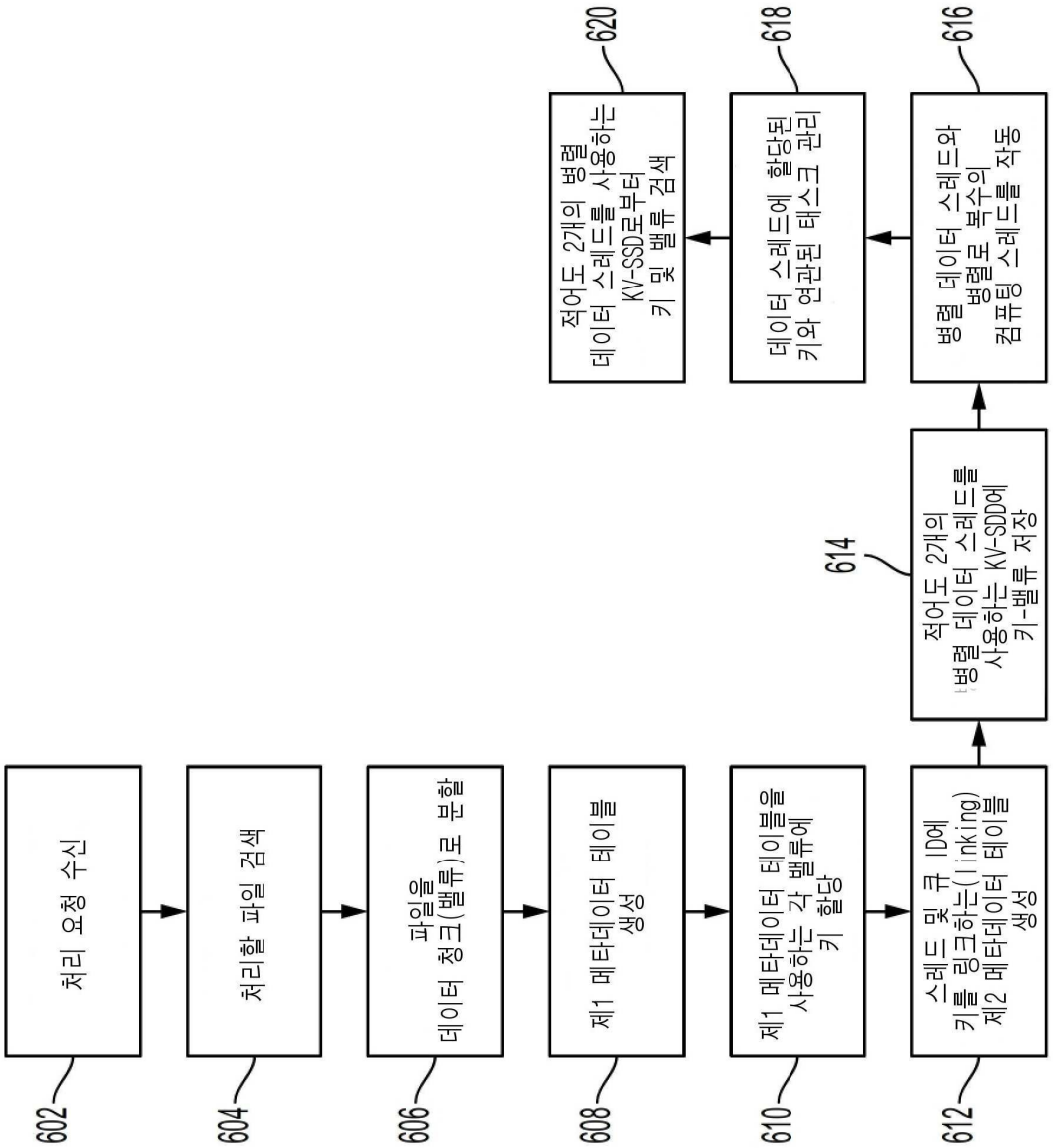
도면4


406
전-처리
패치 jpeg
jpeg -> eps
eps -> 인간 판독 가능 벡터 포맷
인간 판독 가능 벡터 포맷 -> 5D 데이터 벡터 파일
3
5D 벡터 파일 저장
제어 요청 w/크기, w/정렬
파일로부터 밸류 생성
키 생성
KV 메타 데이터 업데이트 및 연결
408
404
2
블록 디바이스
파일 시스템
블록 계층
디바이스 드라이버
블록 명령
SSD
410
KV 디바이스
키
밸류
KV 명령
KV SSD
4
KV 저장
KV 검색
6
412
8
후-처리에 대한 패치 클러스터 평균
414
7
클러스터 평균 저장
밸류로부터 데이터 추출
402
클러스터를 하기 위한 이미지의 수
1
클라이언트
사용자
클러스터의 K-수, 수렴 공차
5
컴퓨팅
KV-평균 클러스터링
416

도면5


데이터 액세스
컴퓨팅
503
스택
카운터
레지스터
데이터
파일
제1 스레드
제2 스레드
제3 스레드
코드
멀티플 스레드
병렬 데이터 액세스 + 병렬 컴퓨팅
502
싱글 스레드
병렬 컴퓨팅
501
시퀀셜

도면6


처리 요청 수신
602
처리할 파일 검색
604
파일을
데이터 청크(밸류)로 분할
606
제1 메타데이터 테이블
생성
608
제1 메타데이터 테이블을
사용하는 각 밸류에
키 할당
610
스레드 및 큐 ID에
키를 링크하는(linking)
제2 메타데이터 테이블
생성
612
적어도 2개의
병렬 데이터 스레드를
사용하는 KV-SSD에
키-밸류 저장
614
병렬 데이터 스레드와
병렬로 복수의
컴퓨팅 스레드를 작동
616
데이터 스레드에 할당된
키와 연관된 태스크 관리
618
적어도 2개의 병렬
데이터 스레드를 사용하는
KV-SSD로부터
키 및 밸류 검색
620