



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0055319
(43) 공개일자 2020년05월21일

(51) 국제특허분류(Int. Cl.)
G06F 9/48 (2018.01) G06F 11/10 (2006.01)
G06F 9/50 (2018.01)
(52) CPC특허분류
G06F 9/4893 (2013.01)
G06F 11/1004 (2013.01)
(21) 출원번호 10-2018-0138848
(22) 출원일자 2018년11월13일
심사청구일자 2018년11월13일

(71) 출원인
인하대학교 산학협력단
인천광역시 미추홀구 인하로 100(용현동, 인하대학교)
(72) 발명자
김덕환
서울특별시 양천구 목동로 230, 103동 1004호(목동, 아이파크아파트)
메히디
인천광역시 미추홀구 경인남길 134, 105호(용현동, 대우주택)
(74) 대리인
양성보

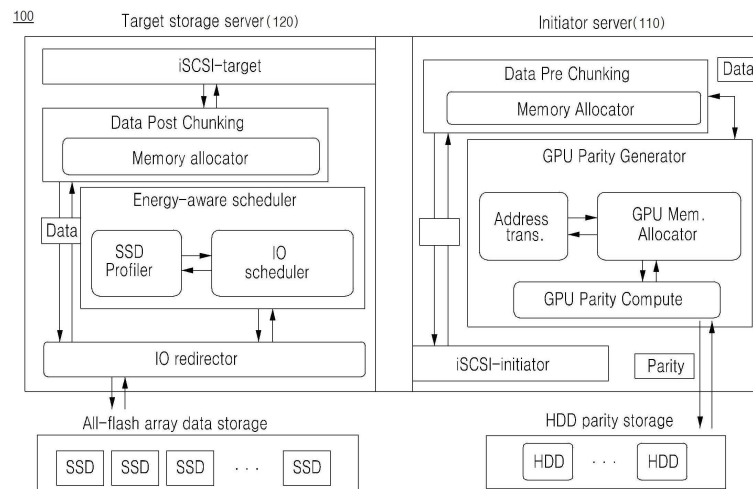
전체 청구항 수 : 총 5 항

(54) 발명의 명칭 분산 스토리지 어플리케이션의 저전력 GPU 스케줄링 방법

(57) 요약

분산 스토리지 어플리케이션의 저전력 GPU 스케줄링 방법이 제시된다. 일 실시예에 따른 GPU(Graphics Processing Unit) 스케줄링 방법은, 이니시에이터 서버의 인코딩 스케줄러는 GPU 패리티 계산 함수를 호출하여 CPU 메모리에 저장된 데이터 블록으로부터 패리티 블록을 생성하는 단계; 상기 이니시에이터 서버의 인코딩 스케줄러는 상기 패리티 블록을 HDD 스토리지에 스트라이프하기 위하여, HDD의 전력 모드를 대기 모드에서 활성 모드로 전환시키는 단계; 상기 이니시에이터 서버의 인코딩 스케줄러는 상기 패리티 블록들을 패리티 HDD로 스트라이프하는 단계; 및 상기 이니시에이터 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위하여, HDD의 전력 모드를 활성 모드에서 대기 모드로 전환시키는 단계를 포함할 수 있다.

대표도



(52) CPC특허분류

G06F 9/5061 (2013.01)

G06F 9/5083 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호 2018R1D1A1B07042602

부처명 교육부

연구관리전문기관 한국연구재단

연구사업명 기본연구(교육부)

연구과제명 [Ezbaro] 고가용성, 고신뢰성 클라우드 시스템을 위해 딥러닝 기술을 적용한 Key-Value 기반의 지능형 소프트웨어 정의 스토리지

기 여 율 1/2

주관기관 인하대학교 산학협력단

연구기간 2018.06.01 ~ 2021.05.31

이 발명을 지원한 국가연구개발사업

과제고유번호 1218210209

부처명 인하대학교 산학협력단

연구관리전문기관 인하대학교 산학협력단

연구사업명 연구비 수주 목적사업

연구과제명 WCSL_(2차년도) 저전력 전기자동차용 자율주행시스템 개발 및 인력양성

기 여 율 1/2

주관기관 인하대학교 산학협력단

연구기간 2016.12.01 ~ 2021.11.30

명세서

청구범위

청구항 1

GPU(Graphics Processing Unit) 스케줄링 방법에 있어서,

이니시에이터 서버의 인코딩 스케줄러는 GPU 패리티 계산 함수를 호출하여 CPU 메모리에 저장된 데이터 블록으로부터 패리티 블록을 생성하는 단계;

상기 이니시에이터 서버의 인코딩 스케줄러는 상기 패리티 블록을 HDD 스토리지에 스트라이프하기 위하여, HDD의 전력 모드를 대기 모드에서 활성 모드로 전환시키는 단계;

상기 이니시에이터 서버의 인코딩 스케줄러는 상기 패리티 블록들을 패리티 HDD로 스트라이프하는 단계; 및

상기 이니시에이터 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위하여, HDD의 전력 모드를 활성 모드에서 대기 모드로 전환시키는 단계

를 포함하는, GPU 스케줄링 방법.

청구항 2

제1항에 있어서,

목표 스토리지 서버의 인코딩 스케줄러는 복수의 SSD를 포함하는 SSD 프로파일러를 생성하는 단계;

상기 목표 스토리지 서버의 인코딩 스케줄러는 상기 목표 스토리지 서버에서 데이터 블록을 읽어 CPU 메모리로 할당하는 단계;

상기 목표 스토리지 서버의 인코딩 스케줄러는 상기 데이터 블록을 SSD에 스트라이프하기 위하여, SSD의 전력 모드를 유틸 모드에서 활성 모드로 전환하는 단계;

상기 목표 스토리지 서버의 인코딩 스케줄러는 상기 CPU 메모리의 상기 데이터 블록들을 복수의 데이터 SSD로 스트라이프하는 단계; 및

상기 목표 스토리지 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위해, SSD의 전력 모드를 활성 모드에서 유틸 모드로 전환하는 단계

를 더 포함하는, GPU 스케줄링 방법.

청구항 3

GPU(Graphics Processing Unit) 스케줄링 시스템에 있어서,

GPU 패리티 계산 함수를 호출하여 CPU 메모리에 저장된 데이터 블록으로부터 패리티 블록을 생성하고, 생성된 상기 패리티 블록들을 HDD로 스트라이프하는 이니시에이터 서버; 및

데이터 블록을 읽어 CPU 메모리로 할당하고, 상기 CPU 메모리의 상기 데이터 블록들을 복수의 데이터 SSD로 스트라이프하는 목표 스토리지 서버

를 포함하는, GPU 스케줄링 시스템.

청구항 4

제3항에 있어서,

상기 이니시에이터 서버는,

상기 이니시에이터 서버의 인코딩 스케줄러가 GPU 패리티 계산 함수를 호출하여 CPU 메모리에 저장된 데이터 블록으로부터 패리티 블록을 생성하고, 상기 이니시에이터 서버의 인코딩 스케줄러는 상기 패리티 블록을 HDD 스토리지에 스트라이프하기 위하여, HDD의 전력 모드를 대기 모드에서 활성 모드로 전환시키며, 상기 이니시에이

터 서버의 인코딩 스케줄러는 상기 패리티 블록들을 패리티 HDD로 스트라이프한 후, 상기 이니시에이터 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위하여, HDD의 전력 모드를 활성 모드에서 대기 모드로 전환시키는 것

을 특징으로 하는, GPU 스케줄링 시스템.

청구항 5

제3항에 있어서,

상기 목표 스토리지 서버는,

상기 목표 스토리지 서버의 인코딩 스케줄러가 복수의 SSD를 포함하는 SSD 프로파일러를 생성하고, 상기 목표 스토리지 서버의 인코딩 스케줄러는 상기 목표 스토리지 서버에서 데이터 블록을 읽어 CPU 메모리로 할당하며, 상기 목표 스토리지 서버의 인코딩 스케줄러는 상기 데이터 블록을 SSD에 스트라이프하기 위하여, SSD의 전력 모드를 유틸 모드에서 활성 모드로 전환하고, 상기 목표 스토리지 서버의 인코딩 스케줄러는 상기 CPU 메모리의 상기 데이터 블록들을 복수의 데이터 SSD로 스트라이프한 후, 상기 목표 스토리지 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위해, SSD의 전력 모드를 활성 모드에서 유틸 모드로 전환하는 것

을 특징으로 하는, GPU 스케줄링 시스템.

발명의 설명

기술 분야

[0001] 아래의 실시예들은 분산 스토리지 어플리케이션의 저전력 GPU 스케줄링 방법에 관한 것으로, 더욱 상세하게는 빅 데이터 처리를 위한 하이브리드 클라우드 스토리지 시스템에 대한 에너지 인식 GPU-RAID 스케줄링 방법에 관한 것이다.

배경 기술

[0002]최근 스토리지 동향은 클라우드 스토리지 시스템의 성능, 신뢰성 및 에너지 소비라는 관점에서 연구자들 및 SSD(Solid State Disk) 제조자들이 돌파구를 생각해야 할 필요성이 있다는 점을 알린다. HDD(Hard Disk Drives), SSD 및 하이브리드 클라우드 스토리지 시스템의 에너지 소비를 감소시키기 위해 많은 방법들이 제안되었다. 이 방법들은 에너지 비용 프로파일링(profiling energy cost), 그래픽 처리 장치(Graphics Processing Unit, GPU) 기반 파일 시스템(GPUfs), 에너지 절약 RAID(Redundant Array of Inexpensive Disks) 구조 및 전력 모드 스케줄링을 포함한다.

[0003]클라우드 스토리지 시스템에서, RAID는 에너지 소비가 크고 데이터와 패리티를 스토리지에 스트라이프(stripe)하기 위해 이레이저 코딩을 적용해야 한다. 이 성질은 스토리지 시스템이 디스크 실패에 강인하도록(resist) 만든다.

[0004]그러나, 입력/출력(Input/Output, I/O) 연산은 CPU를 사용하는 코딩과 스케줄링을 동반하기 때문에, 기존 RAID 시스템의 성능은 중앙 처리 장치(Central Processing Unit, CPU)의 순차적인 성질에 의해 제한된다.

선행기술문헌

비특허문헌

[0005] (비특허문헌 0001) A.Kadav, M. Balakrishnan, V. Prabhakaran, D. Malkhi, "Differential RAID: rethinking RAID for SSD reliability," ACMTrans. Operating Systems Review, pp. 55-59. 2010.

발명의 내용

해결하려는 과제

[0006]실시예들은 분산 스토리지 어플리케이션의 저전력 GPU 스케줄링 방법에 관하여 기술하며, 보다 구체적으로 클라우드 스토리지 시스템을 위한 새로운 에너지 인식 GPU 기반의 이레이저(erasure) 코딩 스케줄러에 관한 기술을

제공한다.

- [0007] 실시예들은 GPU를 사용해 패리티를 생성하며, 이니시에이터 서버에 패리티를 스트라이프하고, 목표 스토리지 서버에 데이터를 스트라이프함으로써, 코딩 시간 및 GPU 사이클의 수를 감소시킬 수 있는 분산 스토리지 어플리케이션의 저전력 GPU 스케줄링 방법을 제공하는데 있다.

과제의 해결 수단

- [0008] 일 실시예에 따른 GPU(Graphics Processing Unit) 스케줄링 방법은, 이니시에이터 서버의 인코딩 스케줄러는 GPU 패리티 계산 함수를 호출하여 CPU 메모리에 저장된 데이터 블록으로부터 패리티 블록을 생성하는 단계; 상기 이니시에이터 서버의 인코딩 스케줄러는 상기 패리티 블록을 HDD 스토리지에 스트라이프하기 위하여, HDD의 전력 모드를 대기 모드에서 활성 모드로 전환시키는 단계; 상기 이니시에이터 서버의 인코딩 스케줄러는 상기 패리티 블록들을 패리티 HDD로 스트라이프하는 단계; 및 상기 이니시에이터 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위하여, HDD의 전력 모드를 활성 모드에서 대기 모드로 전환시키는 단계를 포함할 수 있다.
- [0009] 목표 스토리지 서버의 인코딩 스케줄러는 복수의 SSD를 포함하는 SSD 프로파일러를 생성하는 단계; 상기 목표 스토리지 서버의 인코딩 스케줄러는 상기 목표 스토리지 서버에서 데이터 블록을 읽어 CPU 메모리로 할당하는 단계; 상기 목표 스토리지 서버의 인코딩 스케줄러는 상기 데이터 블록을 SSD에 스트라이프하기 위하여, SSD의 전력 모드를 유틸 모드에서 활성 모드로 전환하는 단계; 상기 목표 스토리지 서버의 인코딩 스케줄러는 상기 CPU 메모리의 상기 데이터 블록들을 복수의 데이터 SSD로 스트라이프하는 단계; 및 상기 목표 스토리지 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위해, SSD의 전력 모드를 활성 모드에서 유틸 모드로 전환하는 단계를 더 포함할 수 있다.
- [0010] 다른 실시예에 따른 GPU(Graphics Processing Unit) 스케줄링 시스템은, GPU 패리티 계산 함수를 호출하여 CPU 메모리에 저장된 데이터 블록으로부터 패리티 블록을 생성하고, 생성된 상기 패리티 블록들을 HDD로 스트라이프하는 이니시에이터 서버; 및 데이터 블록을 읽어 CPU 메모리로 할당하고, 상기 CPU 메모리의 상기 데이터 블록들을 복수의 데이터 SSD로 스트라이프하는 목표 스토리지 서버를 포함하여 이루어질 수 있다.
- [0011] 상기 이니시에이터 서버는, 상기 이니시에이터 서버의 인코딩 스케줄러가 GPU 패리티 계산 함수를 호출하여 CPU 메모리에 저장된 데이터 블록으로부터 패리티 블록을 생성하고, 상기 이니시에이터 서버의 인코딩 스케줄러는 상기 패리티 블록을 HDD 스토리지에 스트라이프하기 위하여, HDD의 전력 모드를 대기 모드에서 활성 모드로 전환시키며, 상기 이니시에이터 서버의 인코딩 스케줄러는 상기 패리티 블록들을 패리티 HDD로 스트라이프한 후, 상기 이니시에이터 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위하여, HDD의 전력 모드를 활성 모드에서 대기 모드로 전환시킬 수 있다.
- [0012] 상기 목표 스토리지 서버는, 상기 목표 스토리지 서버의 인코딩 스케줄러가 복수의 SSD를 포함하는 SSD 프로파일러를 생성하고, 상기 목표 스토리지 서버의 인코딩 스케줄러는 상기 목표 스토리지 서버에서 데이터 블록을 읽어 CPU 메모리로 할당하며, 상기 목표 스토리지 서버의 인코딩 스케줄러는 상기 데이터 블록을 SSD에 스트라이프하기 위하여, SSD의 전력 모드를 유틸 모드에서 활성 모드로 전환하고, 상기 목표 스토리지 서버의 인코딩 스케줄러는 상기 CPU 메모리의 상기 데이터 블록들을 복수의 데이터 SSD로 스트라이프한 후, 상기 목표 스토리지 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위해, SSD의 전력 모드를 활성 모드에서 유틸 모드로 전환할 수 있다.

발명의 효과

- [0013] 실시예들에 따르면 GPU를 사용해 패리티를 생성하며, 이니시에이터 서버에 패리티를 스트라이프하고, 목표 스토리지 서버에 데이터를 스트라이프함으로써, 코딩 시간 및 GPU 사이클의 수를 감소시킬 수 있는 분산 스토리지 어플리케이션의 저전력 GPU 스케줄링 방법을 제공할 수 있다.
- [0014] 또한, 실시예들에 따르면 병렬성(parallelism)을 증가시키며, 패리티 생성 시 어떠한 CPU 병목 현상도 해결할 수 있다.

도면의 간단한 설명

- [0015] 도 1은 종래의 클라우드 스토리지 시스템의 구조를 설명하기 위한 도면이다.
- 도 2는 일 실시예에 따른 이니시에이터 서버의 GPU 스케줄링 방법을 나타내는 흐름도이다.

도 3은 일 실시예에 따른 목표 스토리지 서버의 GPU 스케줄링 방법을 나타내는 흐름도이다.

도 4는 일 실시예에 따른 GPU 스케줄링 시스템을 나타내는 블록도이다.

도 5는 일 실시예에 따른 클라우드 스토리지 시스템의 구조를 설명하기 위한 도면이다.

도 6은 일 실시예에 따른 에너지 소비 결과를 나타내는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0016] 이하, 첨부된 도면을 참조하여 실시예들을 설명한다. 그러나, 기술되는 실시예들은 여러 가지 다른 형태로 변형될 수 있으며, 본 발명의 범위가 이하 설명되는 실시예들에 의하여 한정되는 것은 아니다. 또한, 여러 실시예들은 당해 기술분야에서 평균적인 지식을 가진 자에게 본 발명을 더욱 완전하게 설명하기 위해서 제공되는 것이다. 도면에서 요소들의 형상 및 크기 등은 보다 명확한 설명을 위해 과장될 수 있다.

[0018] RAID(Redundant Array of Inexpensive Disks) 기반 스토리지 시스템은 중앙 처리 장치(Central Processing Unit, CPU)의 순차적인 성질에 의해 성능에 한계가 있으며, 데이터와 패리티(parity)를 스토리지 장치에 스트라이프(stripe)하기 위해 이레이저(eraser) 코딩을 필요로 하기 때문에 많은 에너지를 소비한다.

[0019] 아래의 실시예들은 에너지 인식 GPU 기반 스케줄링을 제공하는 것으로, 빅 데이터 처리를 위한 하이브리드 클라우드 스토리지 시스템에 대한 에너지 인식 GPU-RAID 스케줄링 기술을 제공한다. 제안하는 하이브리드 클라우드 스토리지 시스템은 하나의 이니시에이터 서버(initiator sever), 하나의 목표 스토리지 서버(target storage server) 및 10G 고속 스토리지 네트워크로 구성될 수 있다. 제안하는 하이브리드 클라우드 스토리지 시스템은 목표 서버(목표 스토리지 서버)에 데이터를 스트라이프하고, GPU를 사용해 패리티를 생성하며, 이니시에이터 서버에 패리티를 스트라이프할 수 있으며, 이를 GPU-RAID라고 할 수 있다.

[0020] 제안하는 GPU-RAID는 CPU 사이클의 수 및 코딩 시간을 감소시킬 수 있다는 점에서 기존의 RAID와 차이가 있다. 또한, CPU의 순차적 성질에 기인한 패리티 및 데이터 스트라이핑 병목 현상(striping bottleneck)을 해결하여 IO 성능을 향상시킬 수 있다. 또한, 새로운 에너지 인식 스케줄링 방식이 이니시에이터 서버 및 목표 서버(목표 스토리지 서버)에 적용될 수 있다.

[0021] 에너지 소비의 관점에서, 클라우드 스토리지 시스템의 IO 성능은 패리티를 생성하기 위한 CPU 사이클의 수와 초당 IO 연산의 수와 관련된다. 이레이저 코딩 기술은 클라우드 스토리지 내의 데이터 실패에 대한 강인성(resistance)을 제공한다. 그러나, CPU의 순차적인 성질 및 비탄력적인 전력 스케줄링 방식에 기인하여 기존의 이레이저 코딩 성능은 낮아진다. 결과적으로, 데이터를 인코딩하고 디코딩하는 것은 많은 에너지와 시간을 필요로 한다.

[0022] 에너지 효율의 관점에서, SSD가 ON, OFF, 슬립(sleep), 활성(active) 및 유휴(idle)와 같은 다양한 전력 모드를 갖고 있고 HDD는 슬립(드라이브가 꺼짐), 대기(standby)(저전력 모드, 드라이브가 느리게 회전함), 활성 및 유휴(일반적인 연산)와 같은 또 다른 전력 모드를 갖기 때문에, 다양한 IO 스케줄링 방식들이 제시되었다. CPU 기반 인코딩의 총 에너지 비용 E_{encoding} 은 수학적 식 1 및 수학적 식 2를 기초로 계산될 수 있다.

[0023] [수학적 식 1]

$$e_{\text{coding}} = e_{\text{CPU cycle}} \times N_{\text{core}}$$

[0025] [수학적 식 2]

$$E_{\text{encoding}} = (e_{\text{active}} t_{\text{active}} + e_{\text{idle}} t_{\text{idle}} + e_{\text{coding}} t_{\text{coding}}) \times N_{\text{thread}} \times N_{\text{disk}}$$

[0027] 여기서, t_{idle} , t_{coding} 및 t_{active} 는 각각 유휴 시간, 패리티 생성 시간 및 읽기/쓰기 시간으로 표현된다. 코딩, 활성 모드 및 유휴 모드를 위한 전력 소비는 각각 e_{coding} , e_{active} 및 e_{idle} 로 표현된다. 따라서, 각 CPU 사이클에 대한 전력 소비는 $e_{\text{CPU cycle}}$ 라고 표현된다. 디스크, 쓰레드(threads) 및 CPU 코어의 개수는 N_{disk} , N_{thread} 및 N_{core} 라고 표현된다.

- [0029] 도 1은 종래의 클라우드 스토리지 시스템의 구조를 설명하기 위한 도면이다.
- [0030] 도 1을 참조하면, 종래의 플래시 배열 클라우드 스토리지(flash array cloud storage) 시스템(10)의 구조를 확인할 수 있다. 인코딩 요청은 이니시에이터 서버(initiator server)(11)의 쓰기 연산을 통해 발생한다.
- [0031] 이니시에이터 서버(11)에서 데이터 청킹 방법(data chunking method)은 파일들을 청크들로 분류한다.
- [0032] 그리고, 목표 스토리지 서버(target storage server)(12)에서 리눅스 다중 장치(multiple devices, MD) RAID는 데이터와 패리티 청크들을 플래시 배열 스토리지 시스템에 스트라이프(stripe)한다. 리눅스 MD RAID는 데이터 D 를 k 개의 데이터 원소로 스트라이프하여 $D = \{d_{1,1}, \dots, d_{n, \beta}\}$ 이며, 여기서 n 이 디스크의 수이고 $\beta (= k/n)$ 이 스트라이프의 수라고 가정한다. 패리티 원소를 생성하기 위한 일련의 배타적 논리합(exclusive or, XOR) 연산은 다음 식과 같이 나타낼 수 있다.
- [0033] [수학식 3]
- [0034]
$$p_1 = d_{1,1} \oplus d_{1,2} \oplus \dots \oplus d_{1,n}$$
- [0035] XOR 연산은 하나의 스트라이프 내 데이터 원소들 간에 수행된다. 모든 패리티 데이터는 데이터 벡터의 XOR 연산을 $\beta \times (n-1)$ 번 반복함으로써 생성되며, 각 XOR 연산은 다음 XOR 연산이 수행되기 전에 끝나야 한다. 결과적으로, 총 $\beta \times (n-1)$ 번의 지연이 CPU의 순차적 성질에 의해 발생한다.
- [0037] 도 2는 일 실시예에 따른 이니시에이터 서버의 GPU 스케줄링 방법을 나타내는 흐름도이다.
- [0038] 도 2를 참조하면, 일 실시예에 따른 GPU(Graphics Processing Unit) 스케줄링 방법은, 이니시에이터 서버의 인코딩 스케줄러는 GPU 패리티 계산 함수를 호출하여 CPU 메모리에 저장된 데이터 블록으로부터 패리티 블록을 생성하는 단계(S110), 이니시에이터 서버의 인코딩 스케줄러는 패리티 블록을 HDD 스토리지에 스트라이프하기 위하여, HDD의 전력 모드를 대기 모드에서 활성 모드로 전환시키는 단계(S120), 이니시에이터 서버의 인코딩 스케줄러는 패리티 블록들을 패리티 HDD로 스트라이프하는 단계(S130), 및 이니시에이터 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위하여, HDD의 전력 모드를 활성 모드에서 대기 모드로 전환시키는 단계(S140)를 포함할 수 있다.
- [0039] 도 3은 일 실시예에 따른 목표 스토리지 서버의 GPU 스케줄링 방법을 나타내는 흐름도이다.
- [0040] 도 3을 참조하면, GPU 스케줄링 방법은, 목표 스토리지 서버의 인코딩 스케줄러는 복수의 SSD를 포함하는 SSD 프로파일러를 생성하는 단계(S210), 목표 스토리지 서버의 인코딩 스케줄러는 목표 스토리지 서버에서 데이터 블록을 읽어 CPU 메모리로 할당하는 단계(S220), 목표 스토리지 서버의 인코딩 스케줄러는 데이터 블록을 SSD에 스트라이프하기 위하여, SSD의 전력 모드를 유향 모드에서 활성 모드로 전환하는 단계(S230), 목표 스토리지 서버의 인코딩 스케줄러는 CPU 메모리의 데이터 블록들을 복수의 데이터 SSD로 스트라이프하는 단계(S240), 및 목표 스토리지 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위해, SSD의 전력 모드를 활성 모드에서 유향 모드로 전환하는 단계(S250)를 더 포함할 수 있다.
- [0041] 도 2 및 도 3에서 설명한 일 실시예에 따른 GPU 스케줄링 방법은 아래에서 하나의 예를 들어 설명하기로 한다.
- [0043] 일 실시예에 따른 GPU 스케줄링 방법은 일 실시예에 따른 GPU 스케줄링 시스템을 이용하여 보다 구체적으로 설명할 수 있다.
- [0044] 도 4는 일 실시예에 따른 GPU 스케줄링 시스템을 나타내는 블록도이다.
- [0045] 도 4를 참조하면, 일 실시예에 따른 GPU 스케줄링 시스템은 이니시에이터 서버 및 목표 스토리지 서버를 포함하여 이루어질 수 있다.
- [0046] 이니시에이터 서버는 GPU 패리티 계산 함수를 호출하여 CPU 메모리에 저장된 데이터 블록으로부터 패리티 블록

을 생성하고, 생성된 패리티 블록들을 HDD로 스트라이프할 수 있다.

- [0047] 보다 구체적으로, 단계(S110)에서, 이니시에이터 서버의 인코딩 스케줄러는 GPU 패리티 계산 함수를 호출하여 CPU 메모리에 저장된 데이터 블록으로부터 패리티 블록을 생성할 수 있다.
- [0048] 단계(S120)에서, 이니시에이터 서버의 인코딩 스케줄러는 패리티 블록을 HDD 스토리지에 스트라이프하기 위하여, HDD의 전력 모드를 대기 모드에서 활성 모드로 전환시킬 수 있다.
- [0049] 단계(S130)에서, 이니시에이터 서버의 인코딩 스케줄러는 패리티 블록들을 패리티 HDD로 스트라이프할 수 있다.
- [0050] 단계(S140)에서, 이니시에이터 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위하여, HDD의 전력 모드를 활성 모드에서 대기 모드로 전환시킬 수 있다.
- [0051] 한편, 목표 스토리지 서버는 데이터 블록을 읽어 CPU 메모리로 할당하고, CPU 메모리의 데이터 블록들을 복수의 데이터 SSD로 스트라이프할 수 있다.
- [0052] 보다 구체적으로, 단계(S210)에서, 목표 스토리지 서버의 인코딩 스케줄러는 복수의 SSD를 포함하는 SSD 프로파일러를 생성할 수 있다.
- [0053] 단계(S220)에서, 목표 스토리지 서버의 인코딩 스케줄러는 목표 스토리지 서버에서 데이터 블록을 읽어 CPU 메모리로 할당할 수 있다.
- [0054] 단계(S230)에서, 목표 스토리지 서버의 인코딩 스케줄러는 데이터 블록을 SSD에 스트라이프하기 위하여, SSD의 전력 모드를 유틸 모드에서 활성 모드로 전환할 수 있다.
- [0055] 단계(S240)에서, 목표 스토리지 서버의 인코딩 스케줄러는 CPU 메모리의 데이터 블록들을 복수의 데이터 SSD로 스트라이프할 수 있다.
- [0056] 단계(S250)에서, 목표 스토리지 서버의 인코딩 스케줄러는 전력 소비를 감소시키기 위해, SSD의 전력 모드를 활성 모드에서 유틸 모드로 전환할 수 있다.
- [0057] 아래에서 GPU 스케줄링 방법 및 시스템을 계속하여 설명한다.
- [0059] 도 5는 일 실시예에 따른 클라우드 스토리지 시스템의 구조를 설명하기 위한 도면이다.
- [0060] 도 5를 참조하면, 일 실시예에 따른 하이브리드 클라우드 스토리지 시스템은 에너지 인식 GPU 기반 플래시 배열 클라우드 스토리지(energy-aware GPU-based flash array cloud storage)의 구조를 나타낸다. 이는, 도 4에서 설명한 GPU 스케줄링 시스템(100)을 포함하거나 포함될 수 있으며, 아래에서는 GPU 스케줄링 시스템(100)을 중심으로 설명한다.
- [0061] 일 실시예에 따른 GPU 스케줄링 시스템(100)은 하나의 이니시에이터 서버(110) 및 목표 스토리지 서버(120)를 포함할 수 있다. 실시예에 따라 10G 고속 스토리지 네트워크를 더 포함하여 이루어질 수 있다.
- [0062] 인코딩은 실제 데이터에서 패리티 데이터를 생성하여 데이터와 패리티를 데이터 디스크에 스트라이프하는 과정이다(비특허문헌 1). 일 실시예에 따른 GPU 스케줄링 시스템(100)은 목표 스토리지 서버(120) 및 이니시에이터 서버(110)에 에너지 인식 GPU-RAID 스케줄링을 제공할 수 있다. 제안하는 방법은 패리티를 생성하고 이를 패리티 서버에 스트라이프하기 위해 GPU를 사용하며, 목표 스토리지 서버(120)에 데이터를 스트라이프할 수 있다. 이는 코딩 시간과 에너지 소비를 감소시키고 스토리지 네트워크의 부담을 경감시킬 수 있다. 데이터와 패리티 스토리지 양측에 대해, 에너지 인식 스케줄링 방식을 적용할 수 있다.
- [0063] 이니시에이터 서버(110)에서, 데이터 사전 청킹 모듈(data pre-chunking module)은 파일을 청크들로 분류하고, GPU 패리티 생성자(parity generator)는 GPU 패리티 계산 모듈(GPU parity computing module), 메모리 할당자(memory allocator) 및 주소 변환 함수(address translation function)를 사용해 패리티 데이터를 생성할 수 있다.
- [0064] GPU 패리티 생성자는 메인 메모리로부터 GPU 메모리까지의 주소 변환을 필요로 한다. 그리고, 패리티는 이니시에이터 서버(110) 측 HDD 스토리지에 쓰인다. 전력 소비를 감소시키기 위해, 제안하는 스케줄러는 활성 모드 및 대기 모드 사이에서 HDD 전력 모드를 전환시킨다.

[0065] 목표 스토리지 서버(120)에서, 데이터 사후 청킹 모듈(data post-chunking module)은 사전 청킹 모듈에 의해 정의된 것과 동일한 크기로 청크들을 분류한다. 제안하는 스케줄러는 SSD 프로파일 및 IO 스케줄러 함수를 사용하여 플래시 배열 기반 데이터 스토리지의 전력 모드를 전환시킨다. 에너지 인식 스케줄러 모듈은 SSD 전력 모드를 활성 모드와 유헤 모드 사이에서 전환함으로써 전력 소비를 감소시키는 데 사용된다. 마지막으로, IO 리디렉터(IO redirector)는 에너지 인식 스케줄러 방식을 사용하여 데이터를 플래시 배열 스토리지에 쓴다.

[0067] 표 1은 이니시에이터 서버 및 목표 스토리지 서버 측 인코딩 스케줄러의 슈도 코드(pseudo-code)를 나타낸다.

[0068] [표 1]

<i>Encoding scheduler</i> ($d[nw]$, n , w , m)	
Input: $d[nw]$: data blocks. n : the number of data disks. w : word size. m : the number of parity disks.	
<pre> /* Data initialization at initiator server */ 1. read data blocks from host and allocate it into $d[nw]$ array. /* Parity creation using GPU based parity scheduler at initiator */ 2. parity = GPU_parity_compute ($d[nw]$, n, w, m). 3. switch HDD power mode from "standby" to "active" mode. 4. write parity blocks from CPU memory into Parity HDDs. 5. switch HDD power mode from "active" to "standby" mode. /* Data stripping at target server */ 6. generate SSD profiler consisting of n SSDs. 7. read data blocks from <i>iSCSI target</i> and allocate it into CPU memory $d[nw]$. 8. switch SSDs power mode from "idle" to "active" mode. 9. write data blocks from CPU memory into n SSDs. 10. switch SSDs power mode from "active" to "idle" mode. </pre>	

[0069]

[0070] 표 1을 참조하면, 라인 (1)에서, 데이터 블록은 CPU 메모리에 저장될 수 있다. 라인 (2)에서, 이니시에이터 서버(110)에 있는 인코딩 스케줄러는 GPU_parity_compute 함수를 호출하여 데이터 블록으로부터 패리티 블록을 생성할 수 있다. 라인 (3)에서, 패리티 블록을 HDD 스토리지에 쓰기 위하여, HDD를 대기 모드에서 활성 전력 모드로 전환할 필요가 있다. 라인 (4)에서, 이니시에이터 서버(110) 스케줄러는 패리티 블록들을 패리티 HDD에 스트라이프할 수 있다. 마지막으로, 라인 (5)에서, 전력 소비를 감소시키기 위하여, 스케줄러는 HDD를 활성 모드에서 대기 전력 모드로 전환시킬 수 있다.

[0071] 라인 (6)에서, 목표 스토리지 서버(120) 측 인코딩 스케줄러는 n 개의 SSD를 포함하는 SSD 프로파일러(profile r)를 생성할 수 있다. 라인 (7)에서, 데이터 블록은 목표 스토리지 서버(120) 측에서 iSCSI 목표(target)로부터 CPU 메모리로 저장될 수 있다. 라인 (8)에서, 데이터 블록을 SSD에 스트라이프하기 위하여, SSD 전력 모드를 유헤 모드에서 활성 모드로 바꿔야 한다. 라인 (9)에서, 목표 스토리지 서버(120) 측 인코딩 스케줄러는 데이터 블록들을 n 개의 데이터 SSD로 스트라이프할 수 있다. 마지막으로, 라인 (10)에서, 전력 소비를 감소시키기 위해, 스케줄러는 SSD 전력을 활성 모드에서 유헤 모드로 바꿀 수 있다.

[0072] 이상과 같이, 실시예들은 에너지 인식 GPU 기반 스케줄링 기술을 제안하는 것으로, 제안하는 스케줄러는 CPU 사이클의 수와 코딩 시간을 감소시킬 수 있다는 점에서 기존의 RAID와 차이가 있다. 실시예들에 따르면 GPU를 사용하여 패리티를 생성하고, 이니시에이터 서버에 패리티를 스트라이프하며, 목표 스토리지 서버에 데이터를 스

트라이프할 수 있다. 여기서 데이터 스토리지인 SSD(Solid State Disk)와 패리티 스토리지인 HDD(Hard Disk Drive)를 기반으로 하는 에너지 인식 스케줄링 방식을 적용할 수 있다.

[0074] 에너지 인식 GPU 기반 플래시 배열 클라우드 스토리지에 대해 제안하는 구조는 오픈 소스인 Jersure 소프트웨어를 사용하는 CentOS 운영 체제에서 구현될 수 있다. 하이브리드 클라우드 스토리지에 대한 SSD 및 HDD 사양은 다음과 같이 나타낼 수 있다.

[0075] 표 2는 하이브리드 클라우드 스토리지에 대한 SSD와 HDD 사양을 나타낸다.

[0076] [표 2]

	Samsung SSD 830	Samsung SSD 840	WD Caviar HDD WD10EZEX
P_{active}	0.15 watts	0.07 watts	9 watts
P_{idle}	0.08 watts	0.05 watts	-
$P_{standby}$	-	-	0.09 watts
Capacity	64 GB	128 GB	1 TB
Max read	520 MB/s	530 MB/s	150 MB/s

[0077]

[0078] 제안하는 에너지 인식 특징들을 가능하게 하기 위하여, CentOS 6.2 운영 체제에서 hdparm tool이 사용되었다. Linux-RAID 6이 기존의 RAID 6 구조로서 사용되는 반면, GPU-RAID 6은 제안하는 RAID 6 구조로서 사용될 수 있다.

[0079] 도 6은 일 실시예에 따른 에너지 소비 결과를 나타내는 도면이다.

[0080] 도 6을 참조하면, Linux-RAID 6 및 GPU-RAID 6의 에너지 소비를 보여준다. GPU-RAID 6의 평균적인 에너지 소비는 Linux-RAID 6과 비교했을 때 45% 향상되는 것을 확인할 수 있다. 이는, 청크 사이즈가 증가할 때 Linux-RAID 6에서 보다 GPU-RAID 6에서 t_{active} 및 t_{coding} 가 더 급격히 감소하기 때문이다.

[0081] 실시예들에 따르면 목표 스토리지 서버 및 이니시에이터 서버에 있는 GPU-RAID를 위한 에너지 인식 스케줄링에 더 높은 입력/출력 성능을 제공할 수 있다. GPU-RAID는 GPU를 사용해 패리티를 생성하고 이를 이니시에이터 서버에 스트라이프하여, 목표 스토리지 서버에 데이터를 스트라이프 함으로써 코딩 시간 및 GPU 사이클의 수를 감소시킨다는 점에서 기존의 RAID와 차이가 있다. 이는, 병렬성(parallelism)을 증가시키며, 패리티 생성 시 어떠한 CPU 병목 현상도 해결할 수 있다.

[0083] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 컨트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPA(field programmable array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 컨트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0084] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상

장치(virtual equipment), 컴퓨터 저장 매체 또는 장치에 구체화(embodiment)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

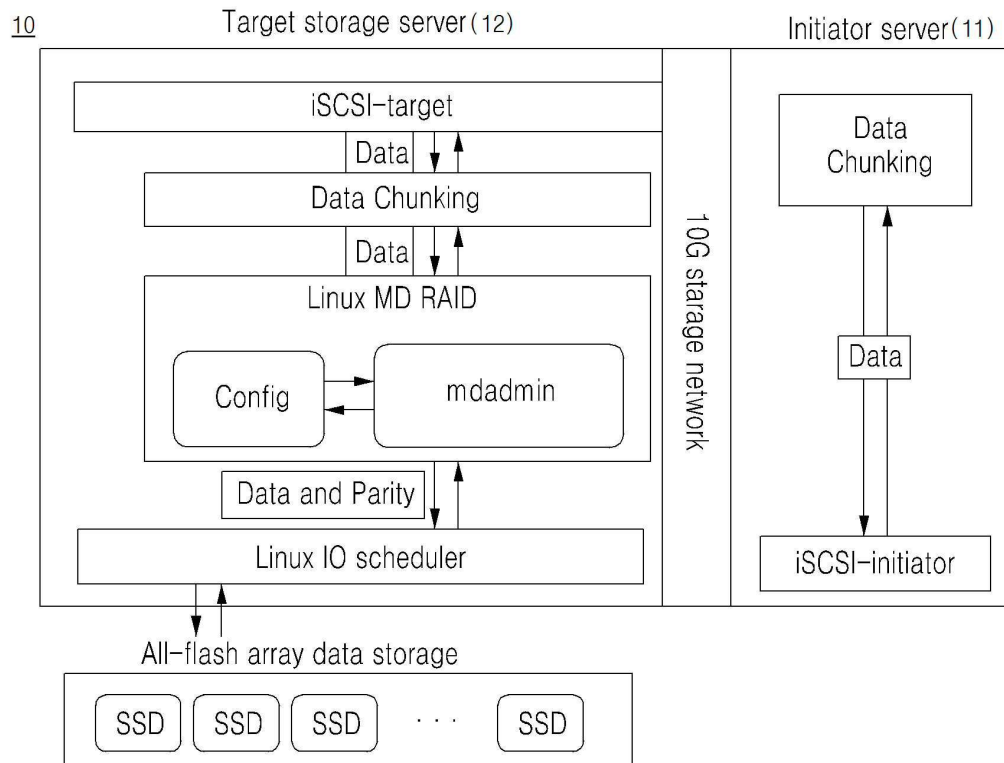
[0085] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다.

[0086] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

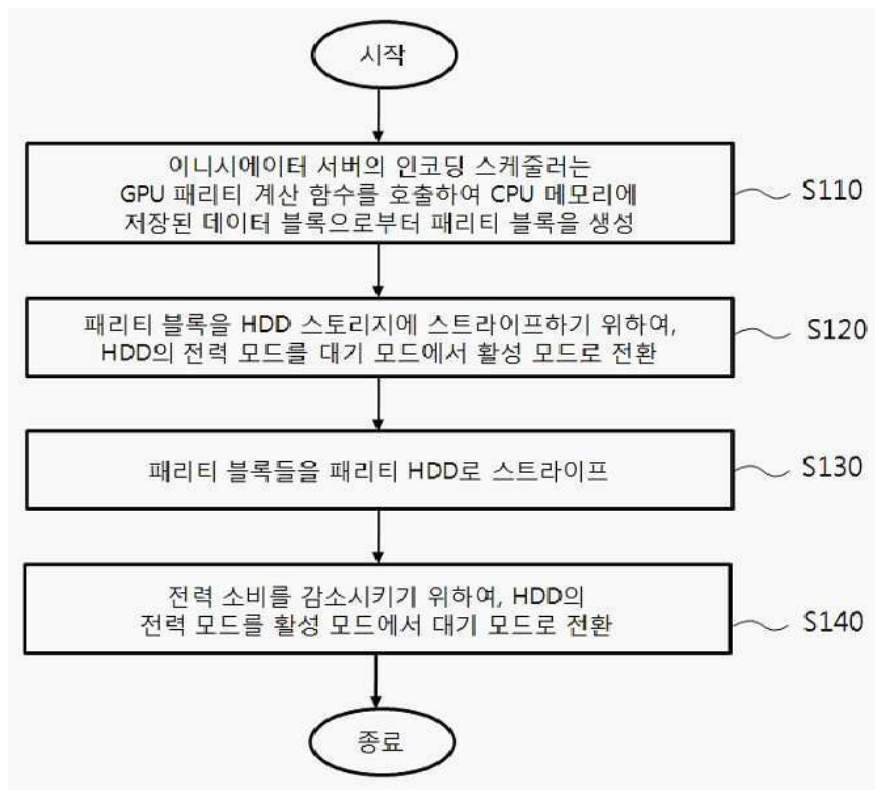
[0087] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

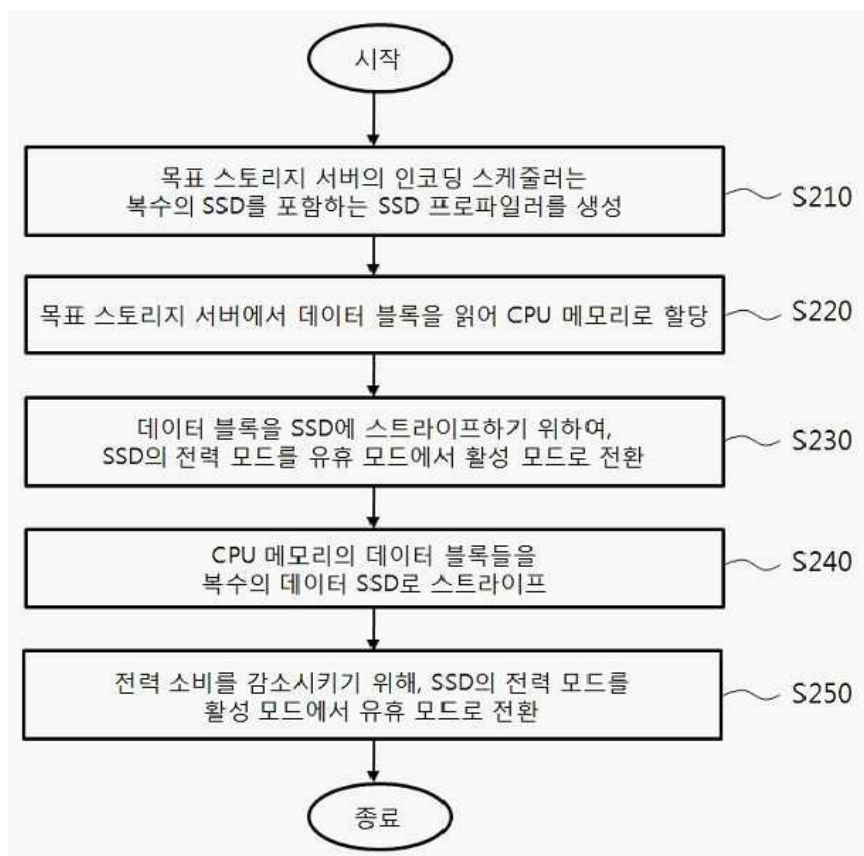
도면1



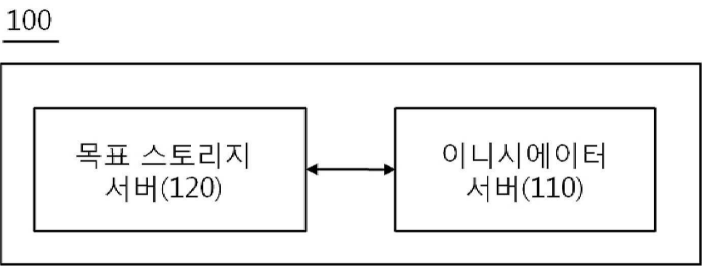
도면2



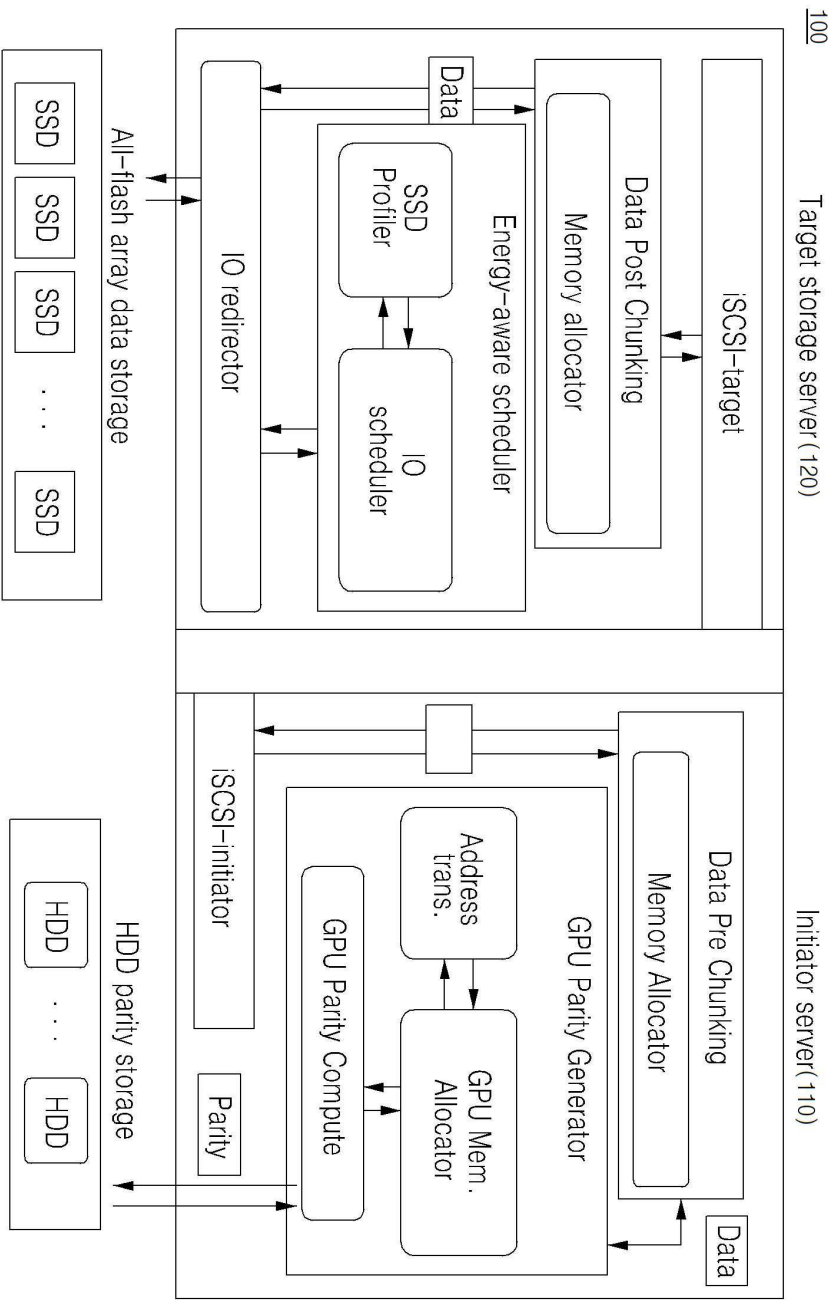
도면3



도면4



도면5



도면6

