



(19) 대한민국특허청(KR)

(12) 등록특허공보(B1)

(45) 공고일자 2015년06월04일

(11) 등록번호 10-1525453

(24) 등록일자 2015년05월28일

(51) 국제특허분류(Int. Cl.)

G06F 11/08 (2006.01)

(21) 출원번호 10-2014-0042386

(22) 출원일자 2014년04월09일

심사청구일자 2014년04월09일

(56) 선행기술조사문헌

조정흠 외 3인, 'SSD 기반 클라우드 스토리지 시스템에서 Erasure code를 이용한 데이터 복제 기법', 2013년도 대한전자공학회 하계학술대회 제36권 1호, pp. 1539-1542 (2013)*

양유석 외 1인, 'SSD의 성능향상을 위한 접근빈도에 따른 데이터 할당 및 교체기법', 2011년 9월 전자공학회 논문지 제 48 권 CI 편 제 5 호

*는 심사관에 의하여 인용된 문헌

(73) 특허권자

인하대학교 산학협력단

인천광역시 남구 인화로 100, 인하대학교 (용현동)

(72) 발명자

김덕환

서울 양천구 목동로 230, 103동 1004호 (목동, 목동아이파크아파트)

(74) 대리인

특허법인 정안

전체 청구항 수 : 총 2 항

심사관 : 이정은

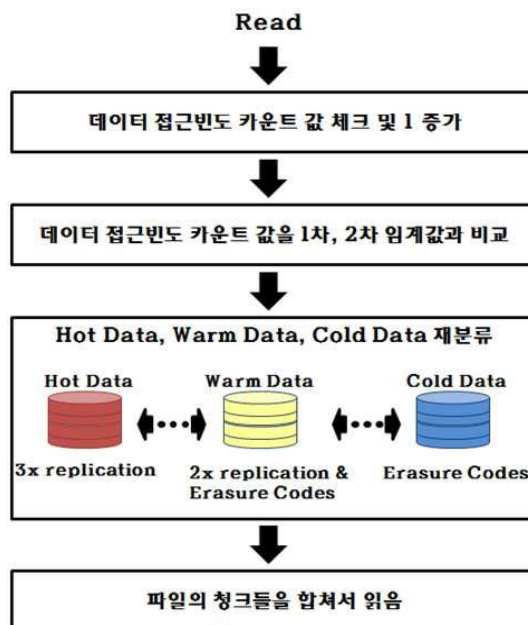
(54) 발명의 명칭 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법

(57) 요약

본 발명은 SSD 기반 R A I D 스토리지 시스템에서, 저장공간 효율성 향상과 입출력 성능 향상을 위하여, 소거 코드(Erasure Codes)를 이용하여 데이터를 복제하는 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법에 관한 것으로서, (a) 입력 파일의 쓰기 명령에 대하여 상기 입력 파일을 청킹하는 단계; (b) 상기 입력 파

(뒷면에 계속)

대표도 - 도10



일에 대한 접근 빈도 파라미터를 생성하는 단계; (c) 상기 접근 빈도 파라미터에 따라 상기 입력 파일을 3개의 유형 중 어느 하나의 유형으로 판단하는 단계; 및, (d) 판단된 유형에 따라 상기 입력 파일의 저장할 복제본의 개수 및, 소거 코딩 여부를 결정하여, 상기 입력 파일의 복제본 또는 소거 코딩 결과를 저장하는 단계로 구성된다.

상기와 같은 데이터 복제 방법에 의하여, R A I D 스토리지 시스템에서 원본 데이터에 중복하여 복제본을 저장하는 종래 기술과 동일한 복구성능을 유지하면서 저장공간 낭비와 입출력이 증가하는 문제를 해결할 수 있다.

이 발명을 지원한 국가연구개발사업

과제고유번호	NIPA-2013-H0502-13-1016
부처명	미래창조과학부
연구관리전문기관	정보통신산업진흥원
연구사업명	IT/SW 창의연구과정
연구과제명	Flash Array를 이용한 Net-Centric 기반의 Cloud Storage 어플라이언스 개발
기 여 율	1/2
주관기관	인하대학교 산학협력단
연구기간	2013.05.01 ~ 2014.04.30

이 발명을 지원한 국가연구개발사업

과제고유번호	2013R1A1A2006912
부처명	교육부
연구관리전문기관	한국연구재단
연구사업명	기초연구사업
연구과제명	클라우드 컴퓨팅 환경에서 차세대 저장장치의 고성능, 고신뢰성, 저전력을 위한 임베디드 소프트웨어 기술 개발
기 여 율	1/2
주관기관	인하대학교 산학협력단
연구기간	2013.06.01 ~ 2014.05.31

명세서

청구범위

청구항 1

적어도 2개의 디스크를 병렬로 연결한 RAID 시스템에서 입력 파일을 복제하여 각 디스크에 분산하여 저장하는, 데이터 접근빈도를 이용한 RAID 시스템의 데이터 복제 방법에 있어서,

- (a) 상기 입력 파일의 쓰기 명령에 대하여 상기 입력 파일을 청킹하는 단계;
- (b) 상기 입력 파일에 대한 접근 빈도 파라미터를 생성하는 단계;
- (c) 상기 접근 빈도 파라미터에 따라 상기 입력 파일을 3개의 유형 중 어느 하나의 유형으로 판단하는 단계;
- (d) 판단된 유형에 따라 상기 입력 파일의 저장할 복제본의 개수 및, 소거 코딩 여부를 결정하여, 상기 입력 파일의 복제본 또는 소거 코딩 결과를 저장하는 단계;
- (e) 상기 입력 파일의 읽기 명령에 대하여 상기 입력 파일의 접근 빈도 파라미터를 갱신하는 단계;
- (f) 갱신된 입력 파일의 접근 빈도에 따라 상기 입력 파일의 유형을 갱신하는 단계;
- (g) 상기 입력 파일의 유형이 변경되면, 변경된 유형에 따른 복제본 개수에 맞추어 상기 입력 파일의 복제본을 삭제하거나 추가 복제본을 생성하고, 상기 변경된 유형에 따른 소거 코딩 여부에 따라 소거 코딩 결과를 생성하거나 원래의 소거 코딩 결과를 삭제하는 단계; 및,
- (h) 상기 입력 파일의 청크들을 합쳐서 읽는 단계를 포함하고,

상기 유형은 핫 데이터 유형, 웜 데이터 유형, 및, 콜드 데이터 유형으로 분류되고, 상기 접근 빈도 파라미터가 사전에 정해진 1차 임계값 보다 작으면, 콜드 데이터 유형으로 판단하고, 상기 접근 빈도 파라미터가 상기 1차 임계값과 같거나 커지면 웜 데이터 유형으로 판단하고, 상기 접근 빈도 파라미터가 사전에 정해진 2차 임계값과 같거나 커지면 핫 데이터 유형으로 판단하고,

상기 2차 임계값은 상기 1차 임계값 보다 크고,

상기 접근 빈도 파라미터는 접근 횟수를 누적하되, 사전에 정해진 시간 범위 내에 데이터 접근이 없으면 감소시키고,

상기 입력 파일이 핫 데이터 유형이면 복제본 3개를 생성하여 저장하고,

상기 입력 파일이 웜 데이터 유형이면 복제본 2개와 소거 코딩 결과를 생성하여 저장하고,

상기 입력 파일이 콜드 데이터 유형이면 파일 1개와 소거 코딩 결과를 생성하여 저장하는 것을 특징으로 하는 데이터 접근빈도를 이용한 RAID 시스템의 데이터 복제 방법.

청구항 2

삭제

청구항 3

삭제

청구항 4

삭제

청구항 5

제1항에 있어서,

상기 소거 코딩 결과는 패리티 청크인 것을 특징으로 하는 데이터 접근빈도를 이용한 RAID 시스템의 데이터 복제 방법.

발명의 설명

기술 분야

[0001] 본 발명은 SSD 기반 RAID 스토리지 시스템에서, 저장공간 효율성 향상과 입출력 성능 향상을 위하여, 소거 코드(Erasure Codes)를 이용하여 데이터를 복제하는 데이터 접근빈도를 이용한 RAID 시스템의 데이터 복제 방법에 관한 것이다.

배경 기술

[0002] 최근 수년간 모바일 기기의 발전과 사용자의 증가로 인하여 인터넷을 기반으로 한 클라우드 서비스가 점차 활성화되고 있다. 이러한 클라우드 서비스를 지원하기 위해서는 고성능, 대용량의 스토리지가 필요하다. 이에, 여러 기업과 대학에서 클라우드 스토리지의 성능향상을 위한 연구가 진행되고 있다[비특허문헌 1].

[0003] 한편, 클라우드 스토리지의 입출력 성능 향상을 위해서는 RAID 기술이 필수적이다. RAID(Redundant Array of Independent Disks)는 두 개 이상의 디스크를 동시에 병렬적으로 접근함으로써 입출력 속도를 향상시키는 방법이며 그 종류도 다양하다[비특허문헌 2].

[0004] 클라우드 서비스의 활성화로 인하여 폭발적으로 증가하는 데이터를 저장할 스토리지의 부족이 문제가 되고 있다. 지속적으로 증가하는 데이터에 따라 스토리지도 더 많이 필요하기 때문이다.

[0005] 또한, 두 개 이상의 디스크를 병렬로 연결한 RAID 시스템을 이용하여, 저장공간의 신뢰성을 위해서 데이터를 64MB의 블록 단위로 청킹한 후, 2 ~ 3개의 복제본을 생성하여 각 블록들을 각 디스크에 분산하여 저장한다. 따라서 하나의 디스크에 문제가 발생하여도 다른 디스크에 저장된 데이터로 복원될 수 있다. 동시에 모든 디스크에 문제가 발생할 수 있는 확률은 매우 낮기 때문에, 저장공간의 신뢰성을 향상시킬 수 있다.

[0006] 그러나 도 1은 상기와 같은 RAID 시스템에서의 파일 복제과정의 문제점을 보여주고 있다. 만약 320MB의 데이터가 들어오면 64MB의 블록 단위로 데이터를 청킹한 후, 3개의 복제본을 생성하고 각 디스크로 나누어 저장한다. 이 방법은 원본 데이터 크기에 3배의 저장공간이 필요하기 때문에 데이터가 커질수록 저장공간 낭비가 심해진다[비특허문헌 3].

[0007] 또한, 대부분의 클라우드 스토리지 시스템은 비용을 절감하기 위해서 저장공간당 비용이 저렴한 HDD를 저장장치로 사용하고 있다. 하지만 스핀들 모터를 사용해 동작하는 HDD는 임의 I/O(Random I/O) 성능이 낮아서 속도의 저하가 발생한다[비특허문헌 4]. 따라서 임의 I/O가 많이 발생하는 클라우드 환경에서는 HDD가 병목현상을 발생시키는 요인이 된다.

[0008] SSD(Solid State Disk)는 NAND 플래시 메모리 기반의 저장장치로 순차 및 임의 I/O 성능이 모두 HDD 보다 높은 저장장치이다. 클라우드 스토리지로 SSD를 사용할 경우 HDD의 사용으로 생기는 병목현상을 줄일 수 있다. 비용면에서도 점차 저렴해지고 있기 때문에 현재 플래시 어레이(Flash Array) 기반의 클라우드 스토리지 제품들이 개발되고 있다.

선행기술문헌

비특허문헌

[0009] (비특허문헌 0001) [비특허문헌 1] A. Bessani, M. Correia, B. Quaresma, F. Andre, P. Sousa, "DEPSKY: Dependable and Secure Storage in a Cloud-of-Clouds", ACM Eurosys'11, pp.31-45, April 2011.

(비특허문헌 0002) [비특허문헌 2] A. Kadav, M. Balakrishnan, V. Prabhakaran, D. Malkhi, "Differential RAID: Rethinking RAID for SSD Reliability", Workshop on Hot Topics in Storage and File

Systems(HotStorage'09), pp.55-59, October 2009.

(비특허문헌 0003) [비특허문헌 3] J. Wang, W. Gong, P. Varman, C. Xie, "Reducing Storage Overhead with Small Write Bottleneck Avoiding in Cloud RAID System," in Proc. of ACM and IEEE Conf. on 13th Grid Computing, pp.174~183, Beijing, China, September 2012.

(비특허문헌 0004) [비특허문헌 4] B. Mao, H. Jiang, S. Wu, Y. Fu, L. Tian, "SAR: SSD Assisted Restore Optimization for Deduplication-based Storage System in the Cloud," in Proc. of IEEE Conf. on 7th Networking, Architecture, and Storage, pp.328~337, Xiamen, China, June 2012.

(비특허문헌 0005) [비특허문헌 5] B. Fan, W. Tantisiriroj, L. Xiao, G. Gibson, "DiskReduce: RAID for Data-Intensive Scalable Computing," in Proc. of ACM Conf. on Supercomputing PDSW'09, pp.6~10, Portland, Oregon, USA, November 2009.

(비특허문헌 0006) [비특허문헌 6] S. Plank, S. Simmerman, C. D. Schuman, "Jerasure: A Library in C/C++ Facilitating Erasure Coding for Storage Applications," Technical Report CS-08-627, University of Tennessee Department of Electrical Engineering and Computer Science, pp.1~59, August 2008.

(비특허문헌 0007) [비특허문헌 7] J. S. Plank, "A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems," Software-Practice & Experience, Vol.27, No.9, pp.995~1012, September 1997.

(비특허문헌 0008) [비특허문헌 8] M. Blaum, J. Brandy, J. Bruck, M. Jai, "EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures," IEEE Transactions on Computers, Vol.44, No.2, pp.192~202, February 1995.

(비특허문헌 0009) [비특허문헌 9] L. H. James, "WEAVER Codes: Highly Fault Tolerant Erasure Codes for Storage Systems," in Proc. of ACM Conf. on FAST, pp.1~10, 2005.

(비특허문헌 0010) [비특허문헌 10] X. Lihao, J. Bruck, "X-code: MDS array codes with optimal encoding," IEEE Transactions on Information Theory, Vol.45, No.1, pp.272~276, January 1999.

(비특허문헌 0011) [비특허문헌 11] J-H. Jo, J-K. Kim, P. Mehdi, D-H. Kim, "Data Replication Method using Erasure Code in SSD based Cloud Storage System," in Proc. of IEEE Conf. on Summer Conference, Vol.36, No.1, pp.1539~1542, Jeju, Korea, July 2013.

(비특허문헌 0012) [비특허문헌 12] J-K. Kim, J-H. Jo, P. Mehdi, D-H. Kim, "Unified De-duplication Method of Data and Parity Disks in SSD-based RAID Storage," in Proc. of IEEE Conf. on Summer Conference, Vol.36, No.1, pp.1543~1546, Jeju, Korea, July 2013.

(비특허문헌 0013) [비특허문헌 13] D. Park, D. H. C. Du, "Hot Data Identification for Flash-based Storage Systems Using Multiple Bloom Filters," in Proc. of IEEE on 27th Mass Storage Systems and Technologies(MSST), pp.1~11, Denver, Colorado, USA, May 2011.

(비특허문헌 0014) [비특허문헌 14] J.-W. Hsieh, L.-P. Chang, T.-W. Kuo, "Efficient Online Identification of Hot Data for Flash-Memory Management," in Proc. of ACM on 20th Symposium on Applied Computing(SAC), pp.838~842, Santa Fe, New Mexico, March 2005.

(비특허문헌 0015) [비특허문헌 15] H.-S. Lee, H.-S. Yun, D.-H. Lee, "HFTL: Hybrid Flash Translation Layer based on Hot Data Identification for Flash Memory," IEEE Transactions on Consumer Electronics, Vol.55, No.4, pp.2005~2011, November 2009.

발명의 내용

해결하려는 과제

[0010]

본 발명의 목적은 상술한 바와 같은 문제점을 해결하기 위한 것으로, SSD 기반 RAID 스토리지 시스템에서, 저장 공간 효율성 향상과 입출력 성능 향상을 위하여, 소거 코드(Erasure Codes)를 이용하여 데이터를 복제하는 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법을 제공하는 것이다.

[0011] 특히, 본 발명의 목적은 데이터 접근 빈도에 따라 복제 횟수를 줄이더라도 소거 코드(Erasure Codes)를 사용하여 데이터 복구 성능을 동일하게 유지하는 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법을 제공하는 것이다.

과제의 해결 수단

[0012] 상기 목적을 달성하기 위해 본 발명은 입력 파일을 복제하여 저장하는, 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법에 관한 것으로서, (a) 상기 입력 파일의 쓰기 명령에 대하여 상기 입력 파일을 청킹하는 단계; (b) 상기 입력 파일에 대한 접근 빈도 파라미터를 생성하는 단계; (c) 상기 접근 빈도 파라미터에 따라 상기 입력 파일을 3개의 유형 중 어느 하나의 유형으로 판단하는 단계; 및, (d) 판단된 유형에 따라 상기 입력 파일의 저장할 복제본의 개수 및, 소거 코딩 여부를 결정하여, 상기 입력 파일의 복제본 또는 소거 코딩 결과를 저장하는 단계를 포함하는 것을 특징으로 한다.

[0013] 또한, 본 발명은 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법에 있어서, 상기 방법은, (e) 상기 입력 파일의 읽기 명령에 대하여 상기 입력 파일의 접근 빈도 파라미터를 갱신하는 단계; (f) 갱신된 입력 파일의 접근 빈도에 따라 상기 입력 파일의 유형을 갱신하는 단계; (g) 상기 입력 파일의 유형이 변경되면, 변경된 유형에 따른 복제본 개수에 맞추어 상기 입력 파일의 복제본을 삭제하거나 추가 복제본을 생성하고, 상기 변경된 유형에 따른 소거 코딩 여부에 따라 소거 코딩 결과를 생성하거나 원래의 소거 코딩 결과를 삭제하는 단계; 및, (h) 상기 입력 파일의 청크들을 합쳐서 읽는 단계를 포함하는 것을 특징으로 한다.

[0014] 또한, 본 발명은 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법에 있어서, 상기 유형은 핫 데이터 유형, 웜 데이터 유형, 및, 콜드 데이터 유형으로 분류되고, 상기 접근 빈도 파라미터가 사전에 정해진 1차 임계값 보다 작으면, 콜드 데이터 유형으로 판단하고, 상기 접근 빈도 파라미터가 상기 1차 임계값과 같거나 커지면 웜 데이터 유형으로 판단하고, 상기 접근 빈도 파라미터가 사전에 정해진 2차 임계값과 같거나 커지면 핫 데이터 유형으로 판단하고, 상기 2차 임계값은 상기 1차 임계값 보다 큰 것을 특징으로 한다.

[0015] 또한, 본 발명은 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법에 있어서, 상기 입력 파일이 핫 데이터 유형이면 복제본 3개를 생성하여 저장하고, 상기 입력 파일이 웜 데이터 유형이면 복제본 2개와 소거 코딩 결과를 생성하여 저장하고, 상기 입력 파일이 콜드 데이터 유형이면 파일 1개와 소거 코딩 결과를 생성하여 저장하는 것을 특징으로 한다.

[0016] 또한, 본 발명은 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법에 있어서, 상기 소거 코딩 결과는 패리티 청크인 것을 특징으로 한다.

발명의 효과

[0017] 상술한 바와 같이, 본 발명에 따른 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법에 의하면, RAID 스토리지 시스템에서 원본데이터에 중복 복제본을 저장하는 종래 기술과 동일한 복구성능을 유지하면서 저장공간 낭비와 입출력이 증가하는 문제를 해결할 수 있는 효과가 얻어진다.

[0018] 특히, 본 발명의 실험을 통하여, 중복되는 데이터 복제본을 생성하는 방법과 패리티만을 생성하는 방법을 비교하여, 데이터 접근빈도를 핫(Hot), 웜(Warm), 콜드(Cold)로 세분화하여 나눔으로써 소거 코드(Erasure Codes)를 이용한 데이터 복제 방법이 종래기술 보다 저장공간 효율성은 40%, 읽기 성능 11% 및 쓰기 성능 10% 향상되는 효과가 얻어진다.

도면의 간단한 설명

[0019] 도 1은 종래기술에 따른 중복 복제 방식의 저장공간 사용의 예시도.

도 2는 본 발명을 실시하기 위한 전체 시스템에 대한 블록 구성도.

도 3은 본 발명의 제1 실시예에 따른 RAID 시스템의 구성도.

- 도 4는 본 발명의 제1 실시예에 따른 핫 데이터 유형의 데이터 복제 방법의 예시도.
- 도 5는 본 발명의 제1 실시예에 따른 ấm 데이터 유형의 데이터 복제 방법의 예시도.
- 도 6은 본 발명의 제1 실시예에 따른 콜드 데이터 유형의 데이터 복제 방법의 예시도.
- 도 7은 본 발명의 제2 실시예에 따른 핫 데이터 유형의 데이터 복제 방법의 예시도.
- 도 8은 본 발명의 제1 실시예에 따른 소거 코드를 이용한 데이터 복제 방법의 예시도.
- 도 9는 본 발명에 따른 제1 실시예에 따른 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법으로서, 쓰기 명령시의 방법을 설명하는 흐름도.
- 도 10은 본 발명에 따른 제1 실시예에 따른 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법으로서, 읽기 명령시의 방법을 설명하는 흐름도.
- 도 11은 본 발명에 따른 제1 실시예에 따른 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법을 구현한 알고리즘의 일례.
- 도 12는 본 발명의 실험에 따라, 실험에 사용한 SSD의 스펙 정보를 나타낸 표.
- 도 13은 본 발명의 실험에 따라, 실험 환경을 나타낸 표.
- 도 14는 본 발명의 실험에 따라, 저장공간 효율성에 대한 실험 성능 비교를 나타낸 표.
- 도 15는 본 발명의 실험에 따라, 데이터 쓰기에 대한 실험 성능 비교를 나타낸 표.
- 도 16은 본 발명의 실험에 따라, 데이터 읽기에 대한 실험 성능 비교를 나타낸 표.

발명을 실시하기 위한 구체적인 내용

- [0020] 이하, 본 발명의 실시를 위한 구체적인 내용을 도면에 따라서 설명한다.
- [0021] 또한, 본 발명을 설명하는데 있어서 동일 부분은 동일 부호를 붙이고, 그 반복 설명은 생략한다.
- [0022] 먼저, 본 발명을 실시하기 위한 전체 시스템을 도 2를 참조하여 설명한다.
- [0023] 도 2에서 보는 바와 같이, 본 발명을 실시하기 위한 전체 시스템은 운영체제(OS Layer)(10), 파일시스템(20), RAID 시스템(30), 및, SSD 스토리지(40)로 구성된다.
- [0024] 운영체제(10)는 통상의 컴퓨터 시스템에서의 운영체제 시스템으로서, 컴퓨터 시스템에 설치되어 실행된다. 컴퓨터 시스템에서 설치되어 실행되는 사용자 프로그램 또는 시스템 프로그램으로부터 I/O(입출력) 명령을 받으면, 해당 입출력 명령에 따라 파일 시스템을 통해, RAID 시스템(30) 및 SSD 스토리지(40) 등 하드웨어를 제어하여 입출력 기능을 수행한다. 파일 시스템(20)은 컴퓨터 시스템에서 파일을 관리하는 통상의 파일 관리 시스템이다.
- [0025] SSD 스토리지(40)는 SSD(Solid State Disk) 방식의 디스크로 구성된 스토리지 또는 저장장치이다. SSD는 NAND 플래시 메모리가 병렬로 결합하여 구성된 저장장치이다.
- [0026] RAID 시스템(30)은 적어도 3개 이상의 디스크로 구성된다. 이때, 디스크는 물리적인 저장장치인 SSD 스토리지(40)를 디스크 단위로 논리적인 장치로 구성한 것이다. RAID 시스템(30)은 데이터를 복제하여 2개 이상의 복제 데이터를 저장하는 경우, 서로 다른 디스크에 나누어 저장한다. 특히, RAID 시스템(30)은 입력 데이터에 대하여 청킹 과정을 거친 후, 청킹 단위로 데이터를 저장한다.
- [0027] 또한, RAID 시스템(30)은 소거 코딩(Erasure Codes)을 이용하여 패리티 정보를 저장하는 경우, 청크 단위로 패리티 데이터를 생성하고, 청크 단위로 데이터 및 패리티 데이터를 저장한다.
- [0028] 도 3에서 보는 바와 같이, 데이터 청크와 패리티 청크는 스트라이프 방식으로 각각의 디스크에 저장된다. 바람직하게는, 청크 크기의 범위는 32~128KB이다. RAID에서 스트라이프 방식은 데이터를 청크 사이즈 단위로 나누고 디스크 수 만큼의 청크를 하나의 세트 구성하여 저장하는 방법이다.
- [0029] 이때 도 3에서와 같이, RAID-5,6 에서의 스트라이프 방식에 의해, 데이터 세트의 청크와 패리티 청크를 순서대로 A1, A2, A3, PA 저장한다. 그리고 다음 데이터 세트에서는 B2, B3, PB, B1 순서로 저장한다. 이것은 패리티

체크가 하나의 디스크에 편중될 때 신뢰성 저하를 방지하기 위함이다. 패리티 체크가 하나의 디스크에 편중될 경우 패리티 디스크에서 장애가 발생하면 패리티 정보가 없어서 데이터 복구를 할 수 없게 되기 때문이다.

[0030] 한편, RAID 시스템(30)은 디스크 손상 등에 의해 데이터가 손실되는 경우, 다른 디스크에 복제된 데이터 체크를 이용하거나, 패리티 데이터를 이용하여 데이터를 복구할 수 있다.

[0031] 다음으로, 본 발명에 사용하는 소거 코딩(Erasure Coding) 방법에 대하여 구체적으로 설명한다.

[0032] 소거 코딩(Erasure Coding)(또는 소거 코드)은 데이터와 인코딩 과정을 통해 생성한 코드들을 저장하여 데이터 손실시 디코딩 과정을 거쳐 원본 데이터를 복구하는 방법이다[비특허문헌 6]. 뛰어난 오류 복구 성능과 입출력 성능으로 인해서 데이터 신뢰성이 중요한 시스템에 소거 코드(Erasure Codes)를 사용한다. 소거 코드(Erasure Codes)로 생성된 패리티가 데이터 복제본 생성보다 적은 저장공간을 차지하므로 신뢰성을 제공하면서 저장공간 효율성 또한 높일 수 있다.

[0033] 소거 코드(Erasure Codes)의 종류로는 리드-솔로몬 코드(Reed-Solomon Code)[비특허문헌 7], 에베노드 코드(EVENODD code)[비특허문헌 8], 위버 코드(Weaver code)[비특허문헌 9], 엑스 코드(X-code)[비특허문헌 10] 등 다양한 코드가 있다. 각 소거 코드(Erasure Codes) 별로 다른 알고리즘을 사용하며 연산 복잡도를 줄이면서 복구 성능을 높이기 위한 기술이 개발되고 있다.

[0034] 다음으로, 본 발명의 제1 실시예에 따른 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법을 도 4 내지 도 6을 참조하여 설명한다.

[0035] 종래기술에 따른 RAID 스토리지에서의 데이터 복제 방법은 데이터 손실로부터 신뢰성을 유지하기 위해 2개 이상의 복제본을 생성한다. 따라서 종래기술에 따른 복제 방법은 저장공간 낭비를 유발한다. 이에 반해 본 발명에 따른 방법은 저장공간 효율성과 입출력 속도를 높이기 위해서 데이터 접근빈도에 따라 세 가지 다른 복제 방법을 사용한다.

[0036] 본 발명에 따른 방법은 접근빈도에 따라 데이터를 핫 데이터(Hot Data), 웜 데이터(Warm Data), 콜드 데이터(Cold Data)로 분류한다.

[0037] 핫 데이터(Hot Data)는 접근빈도가 높아 자주 사용되는 데이터이다. 핫 데이터(Hot Data)는 데이터 손실시에 강한 신뢰성을 보장하기 위해서 도 4와 같이 3개의 복제본을 생성한다[비특허문헌 11].

[0038] 웜 데이터(Warm Data)는 핫 데이터(Hot Data) 보다 접근 빈도가 낮은 데이터이지만 추후 클라이언트의 잦은 사용으로 핫 데이터(Hot Data)가 될 가능성이 높은 데이터이다. 따라서 웜 데이터(Warm Data)는 도 5와 같이 2개의 복제본 생성과 소거 코드(Erasure Codes)를 사용하여 신뢰성을 확보한다.

[0039] 콜드 데이터(Cold Data)는 접근 빈도가 가장 낮아 자주 사용되지 않는 데이터로 도 6과 같이 소거 코드(Erasure Codes)만을 사용해 데이터 손실에 대비한다.

[0040] 핫 데이터(Hot Data)는 일반적으로 전체 데이터에서 차지하는 비율이 상대적으로 낮기 때문에 3개의 복제본을 생성하더라도 저장공간 효율성이 크게 떨어지지 않는다. 핫 데이터(Hot Data)에 비해 접근빈도가 낮은 웜 데이터(Warm Data), 콜드 데이터(Cold Data)의 경우 데이터 양이 많기 때문에 저장공간 효율성을 위해 소거 코드(Erasure Codes)를 적용하여 패리티를 생성한 후 저장한다. 도 5와 도 6은 소거 코드(Erasure Codes)를 적용했을 때의 데이터 크기를 보여준다.

[0041] 도 4와 비교해 보면, 복제방식은 데이터의 크기에 비해 3배의 저장공간이 필요하다. 그러나 콜드 데이터(Cold Data)일 경우 소거 코드(Erasure Codes)를 적용하면 데이터에 1.4배의 저장공간이 필요하다. 또한, 저장속도도 패리티 연산시간이 있지만 3개의 복제본을 생성하는 것보다 적은 시간이 소모되기 때문에 쓰기 속도도 향상된다.

[0042] 즉, 소거코딩 결과를 "입력 파일의 데이터 체크 + 패리티 체크"라고 보면, 핫 데이터, 웜 데이터, 콜드 데이터는 다음과 같이 요약될 수 있다.

[0043] 핫 데이터는 복제본 3개, 또는 "입력 파일 + 복제본 2개"로 나누어 분산되어 저장되고, 웜 데이터는 복제본 1개 + 소거코딩 결과(입력파일 체크 + 패리티 체크)로 나누어 분산되어 저장된다. 또한, 콜드 데이터는 소거코딩 결

과(입력파일 청킹 + 패리티 청킹)로 나누어 분산되어 저장된다.

- [0044] 다음으로, 본 발명의 제2 실시예에 따른 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법을 도 7을 참조하여 설명한다.
- [0045] 본 발명의 제2 실시예는 본 발명의 제1 실시예에서 데이터의 유형을 핫 데이터 유형과 콜드 데이터 유형의 2개로 구분하는 방식의 방법이다. 나머지 기술 내용은 앞서 설명한 제1 실시예와 동일하다.
- [0046] 즉, 본 발명의 제2 실시예에 따른 방법은 접근빈도에 따라 데이터를 핫 데이터(Hot Data), 및, 콜드 데이터(Cold Data)로 분류한다.
- [0047] 핫 데이터(Hot Data)는 접근빈도가 높아 자주 사용되는 데이터이다. 핫 데이터(Hot Data)는 데이터 손실에 강한 신뢰성을 보장하기 위해서 도 7과 같이 2개의 복제본을 생성한다.
- [0048] 콜드 데이터(Cold Data)는 접근 빈도가 가장 낮아 자주 사용되지 않는 데이터로 앞서 제1 실시예의 도 6과 같이 소거 코드(Erasure Codes)만을 사용해 데이터 손실에 대비한다.
- [0049] 핫 데이터(Hot Data)는 일반적으로 전체 데이터에서 차지하는 비율이 상대적으로 낮기 때문에 2개의 복제본을 생성하더라도 저장공간 효율성이 크게 떨어지지 않는다. 핫 데이터(Hot Data)에 비해 접근빈도가 낮은 콜드 데이터(Cold Data)의 경우 데이터 양이 많기 때문에 저장공간 효율성을 위해 소거 코드(Erasure Codes)를 적용하여 패리티를 생성한 후 저장한다.
- [0050] 이하의 구체적 복제 방법은 제1 실시예를 중심으로 설명한다. 그러나 제1 실시예에 적용되는 이하의 구체적 방법은 본 발명의 제2 실시예에도 동일하게 적용될 수 있다.
- [0051] 다음으로, 본 발명의 제1 실시예에 따른 RAID 스토리지 시스템에서 소거 코드(Erasure Codes)를 이용한 데이터 복제 방법을 도 8을 참조하여 보다 구체적으로 설명한다.
- [0052] 웜 데이터(Warm Data)와 콜드 데이터(Cold Data)를 위한 데이터 복제 방법은 원본 데이터와 소거 코드(Erasure Codes)를 이용하여 패리티 정보를 저장한다.
- [0053] 도 8은 데이터 복제를 위해 RAID-6와 소거 코드(Erasure Codes)를 사용하는 예를 보여준다. RAID-6는 두 개의 패리티 블록 저장하여 두 개의 데이터 블록이 손실될 경우 복구할 수 있게 하는 기능을 한다. 또한, 소거 코드(Erasure Codes)는 원본 데이터에 연산 과정인 인코딩을 거쳐 패리티 정보를 생성하는 역할을 한다[비특허문헌 12].
- [0054] 소거 코드(Erasure Codes)를 이용한 데이터 복제 방법의 쓰기 과정은 먼저 입력 데이터를 청킹하는 과정까지 거친 후에 도 9과 같이 각 청크들을 소거 코딩(Erasure Coding) 과정을 거쳐 패리티 청크를 생성한다.
- [0055] 생성된 패리티 청크는 데이터 청크와 합쳐져 하나의 스트라이프를 구성하게 된다. 따라서, 하나의 스트라이프 안에는 데이터와 패리티 청크가 합쳐져 있다. 여기서 각 데이터 및 패리티 청크들은 하나의 데이터 노드에 할당되어 저장해야 한다. 또한, 각각의 스트라이프는 패리티 청크가 배치될 데이터 노드를 변경하면서 저장하여 하나의 데이터 노드에 패리티 청크가 몰리는 현상을 방지한다.
- [0056] 다음으로, 본 발명의 제1 실시예에 따른 데이터 접근빈도를 이용한 R A I D 시스템의 데이터 복제 방법을 도 9 내지 도 11을 참조하여 설명한다.
- [0057] 도 9과 도 10는 접근빈도에 따라서 소거 코드(Erasure Codes)를 이용하는 데이터 복제 방법의 순서도로서, 각각 쓰기 명령 및 읽기 명령에 대한 순서도를 표시한 것이다. 도 11은 본 발명에 따른 데이터 복제 방법을 구현한 알고리즘을 보여준다.
- [0058] 먼저 시스템에 쓰기 명령이 들어오면 64MB 크기의 블록 단위로 데이터를 청킹한다.
- [0059] 다음으로, 쓰기 요청된 데이터에 대해 메타 데이터 영역에 해당 데이터의 접근빈도 카운트 값과 저장방식을 저장한다.

- [0060] 접근 빈도 파라미터는 데이터가 입력된 후부터 누적된 접근 횟수이다. 그리고 특정한 시간 범위(예를 들어 6시간) 내에 데이터 접근이 없다면 접근 빈도 파라미터를 (예를 들어 1씩) 감소시킨다. 이는 핫 데이터에서 콜드 데이터로 전환시키기 위한 것이다.
- [0061] 처음 쓰기 명령된 파일은 접근횟수가 없어 콜드 데이터(Cold Data)가 되므로 소거 코딩(Erasure Coding)을 적용하여 원본 파일과 패리티 정보를 저장한다.
- [0062] 그리고 각 파일에 읽기 명령이 들어오면, 메타 데이터를 참조하여 데이터가 어느 방식으로 저장 되어 있는지 확인하고 주소값을 확인한다. 그리고 접근횟수가 늘어났기 때문에 메타 데이터에서 접근빈도 카운트(Access Frequency Count) 값을 1씩 증가시킨다. 그 후 파일을 구성하는 청크들을 모아 읽기 요청된 데이터를 제공한다.
- [0063] 만약, 읽기 명령이 빈번히 발생해서 접근빈도 카운트 값이 증가하여 1차 임계값과 같거나 커지면 웜 데이터(Warm Data)가 되어 신뢰성을 위해 원본 파일의 복제본을 1회 더 저장한다.
- [0064] 접근빈도 카운트 값이 더 증가하여 2차 임계값과 같거나 커지게 되면 핫 데이터(Hot Data)가 되어 복제본을 1회 더 저장한다.
- [0065] 반대로 일정 시간 동안 파일에 대한 읽기 명령이 발생하지 않는다면 접근빈도 카운트값을 감소시켜 저장공간 효율성 향상을 위해 소거 코딩(Erasure Coding) 방법으로 저장한다. 접근빈도 카운트값이 2차 임계값보다 작아지면 복제본을 하나 삭제하여 핫 데이터(Hot Data)를 웜 데이터(Warm Data)로 바꾸고 소거 코드(Erasure Codes)의 패리티 정보를 생성한다. 1차 임계값보다 작아지면 복제본을 하나 더 삭제하여 웜 데이터(Warm Data)를 콜드 데이터(Cold Data)로 바꾸고 원본 데이터와 패리티 정보만 남게 된다.
- [0066] 도 7는 데이터 청크들이 데이터 노드에 어떻게 저장된 것인지를 보여준다. 즉, 도 7에 데이터 노드(Data Node)가 1~8까지 표시되고 있다. 복제본 또는 소거코드 결과는 데이터 노드(Data Node)에 분산되어 저장된다. 하나의 데이터 노드(Data Node)에 청크를 모두 저장하지 않는다. 도 7 은 소거 코딩일 때 데이터 청크와 패리티 청크가 스트라이프되어 여러 데이터 노드에 분산되어 저장되는 것을 나타내고 있다.
- [0067] 복제본의 저장은 도 7에서 패리티 청크만 없을 뿐이고 데이터 청크를 여러 데이터 노드에 분산하여 저장한다. 즉, 접근빈도가 1차 임계값 이상이면, 콜드 데이터에서 웜 데이터로 변경되고, 이때, 복제본이 1회 더 저장된다. 그리고 접근빈도가 2차 임계값 이상이면 웜 데이터에서 핫 데이터로 변경되고, 이때, 패리티를 식제하고 복제본 1회 더 생성하여 저장한다.
- [0068] 다음으로, 본 발명의 효과를 실험결과를 통해 설명한다.
- [0069] 본 발명에 따른 방법의 성능평가를 위해 워크로드 데이터 크기를 256MB, 512MB, 1GB, 2GB로 각각 설정했다. 워크로드의 크기가 작아서 청크사이즈는 4KB 크기로 청킹하였다.
- [0070] 데이터 저장방식을 결정하는 기준인 임계값은 각 실험 전에 무작위로 읽기 명령을 내려 각 데이터의 접근빈도 카운트 값을 확인하였다.
- [0071] 핫 데이터(Hot Data), 웜 데이터(Warm Data), 콜드 데이터(Cold Data)의 비율은 실제 서버에서 추출한 워크로드에서 얻은 비율을 이용하였다[비특허문헌 13~15]. 각 참고논문에서는 Financial1, MSR, Distilled, RealSSD, digital camera, Linux O/S의 워크로드를 사용하였다.
- [0072] 각 워크로드에서 핫 데이터(Hot Data)의 비율은 평균이 10%였으며 따라서 실험에서도 핫 데이터(Hot Data)의 비율을 10%로 정하였다. 웜 데이터(Warm Data)는 참고논문에서 핫 데이터(Hot Data)의 비율이 30%까지 올라가는 점을 참고하여 20%로 비율을 정하였다. 실험에서 핫 데이터(Hot Data)가 차지하는 비율은 전체 데이터의 약 10%가 되도록 설정하였고, 웜 데이터(Warm Data)는 전체 데이터의 약 20%, 콜드 데이터(Cold Data)는 약 70%가 되도록 했다. 패리티생성 방식은 리드-솔로몬(Reed-Solomon) 코드를 적용하여 패리티 비트를 생성했다. 복제시에는 기존의 분산 파일 시스템과 같이 3개의 복제본을 만들어 저장했다.
- [0073] 실험환경은 다음과 같다. 벤치마크 툴로 IOzone을 사용하여 다양한 크기의 워크로드 데이터를 생성하였으며, 2개의 서로 다른 디스크를 사용하였다. PC의 사양은 동일하게 CPU는 Intel Core2(2.4Ghz)이고, DDR2 2GB메모리를 사용하였다. 운영체제는 리눅스 CentOS 6.2에 커널 버전 3.8.13을 이용했다. 저장장치는 SAMSUNG SSD 64GB를

PC에 2개를 장착하였다. 사용된 SSD의 스펙은 도 12의 표와 같고, 실험에 사용된 컴퓨터의 성능 및 환경은 도 13의 표와 같다.

[0074] 실험 결과는 도 14, 15, 16과 같다. 각각 저장공간 효율성, 쓰기/읽기 성능을 비교하였으며 데이터 복제방식만을 사용하는 복제방식 방법과 패리티 방식만을 사용하는 패리티 방식 방법을 본 발명에 따른 방법의 비교대상으로 하였다.

[0075] 실험결과 도 14의 저장공간 효율성 면에서는 본 발명에 따른 방법이 복제방식 방법 보다 약 40%정도 적게 저장공간을 사용하였다. 복제방식 방법은 3개의 복제본을 만들기 때문에 가장 저장공간을 많이 소모하고 패리티 방식 방법은 소거 코드(Erasure Codes)만을 사용하기 때문에 가장 저장공간을 적게 차지한다. 본 발명에 따른 방법은 이 두가지 사이에 위치하며 패리티 방식 방법에 근접한 수치를 보여주어 저장공간 효율성이 향상됨을 알 수 있었다.

[0076] 도 15, 16은 각각 쓰기 성능과 읽기 성능을 비교한 그래프이다. 쓰기 성능에서는 본 발명에 따른 방법이 복제방식 방법 보다 약10%, 패리티 방식 방법 보다 약7% 향상되었다. 읽기 성능에서는 본 발명에 따른 방법이 복제방식 방법 보다 약11%, 패리티 방식 방법 보다 약12% 향상되었다. 실험결과 본 발명에 따른 방법은 Hot, Warm, Cold로 세분화하여 데이터 접근빈도를 나눔으로써 데이터 복제본만 생성하거나 패리티만을 생성하는 방법들보다 성능이 향상되었다. 이는 다수의 복제본이나 패리티만을 생성하는것 보다 접근빈도를 고려하여 복제와 패리티 방법을 같이 사용하는 것이 효율적임을 알려준다.

[0077] 본 발명에서는 클라우드 스토리지 시스템에서 원본데이터에 3개의 복제본을 저장하는 방법에서 발생하는 저장공간 낭비와 입출력이 증가하는 문제를 해결하기 위해 데이터 접근빈도에 따라 기존의 소거 코드(Erasure Codes)를 이용한 데이터 복제 방법을 개발하였다. 기존의 소거 코드(Erasure Codes)를 데이터 접근빈도에 따라 사용하여 3개의 복제본을 생성하는 방법과 동일한 복구성능을 유지하였다. 실험에서는 3개의 데이터 복제본을 생성하는 방법과 패리티만을 생성하는 방법을 비교하였다. 데이터 접근빈도를 핫(Hot), 웜(Warm), 콜드(Cold)로 세분화하여 나눔으로써 소거 코드(Erasure Codes)를 이용한 데이터 복제 기법이 종래기술 보다 저장공간 효율성은 40%, 읽기 성능 11% 및 쓰기 성능 10% 향상되었다.

[0078] 이상, 본 발명자에 의해서 이루어진 발명을 실시 예에 따라 구체적으로 설명하였지만, 본 발명은 실시 예에 한정되는 것은 아니고, 그 요지를 이탈하지 않는 범위에서 여러 가지로 변경 가능한 것은 물론이다.

[0079]

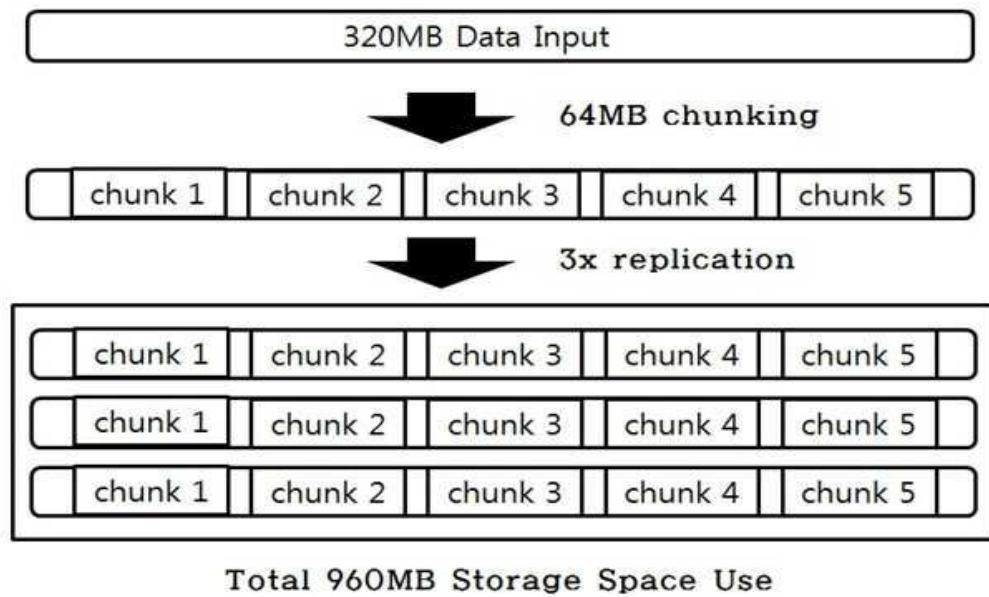
부호의 설명

[0080]

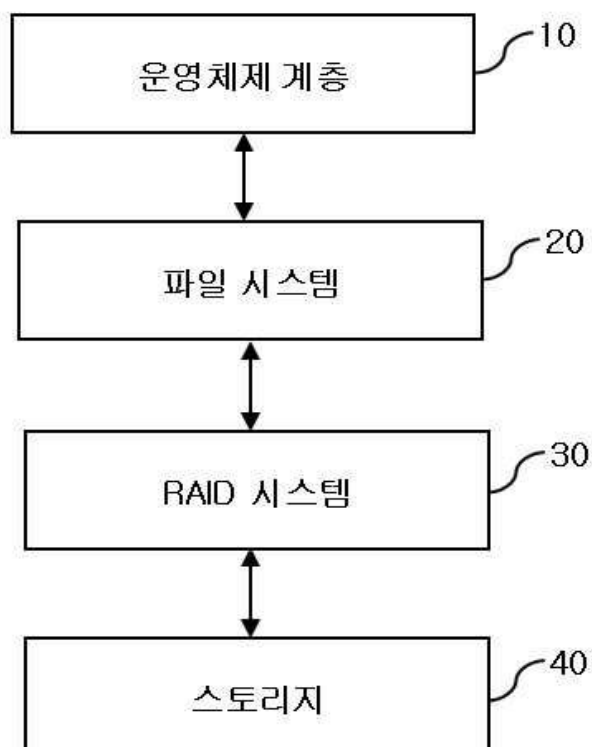
10 : 운영체제	20 : 파일 시스템
30 : RAID 시스템	40 : SSD 스토리지

도면

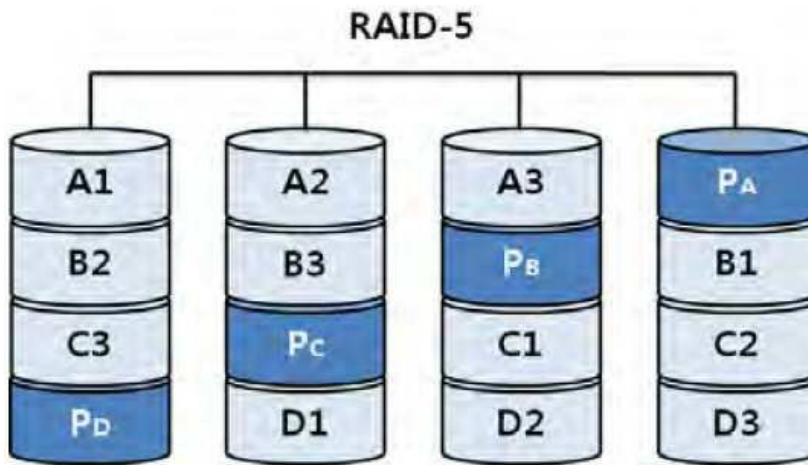
도면1



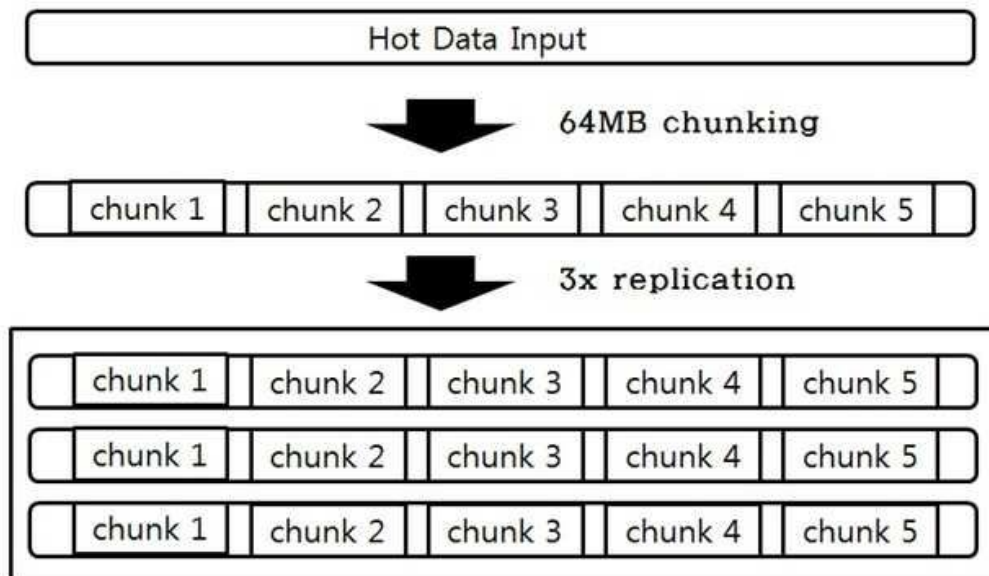
도면2



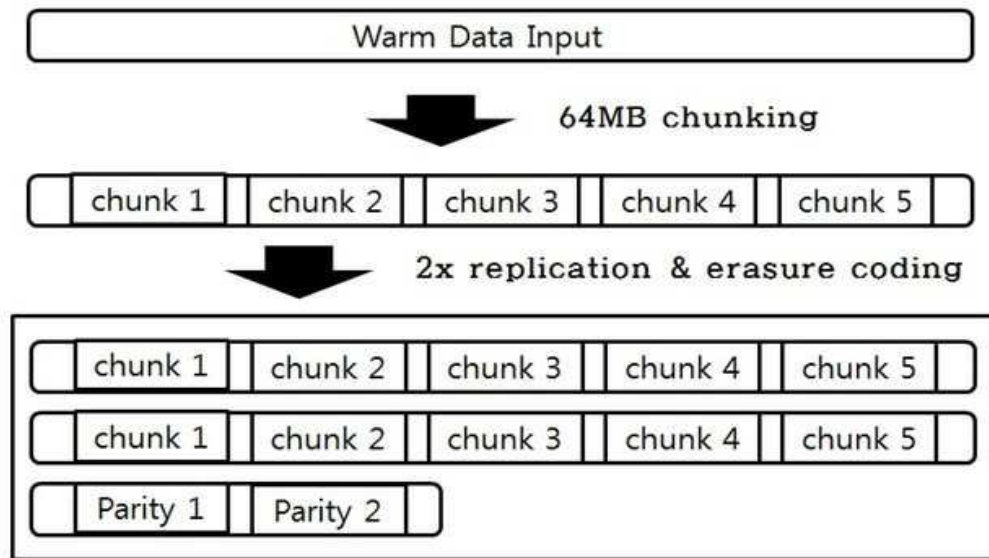
도면3



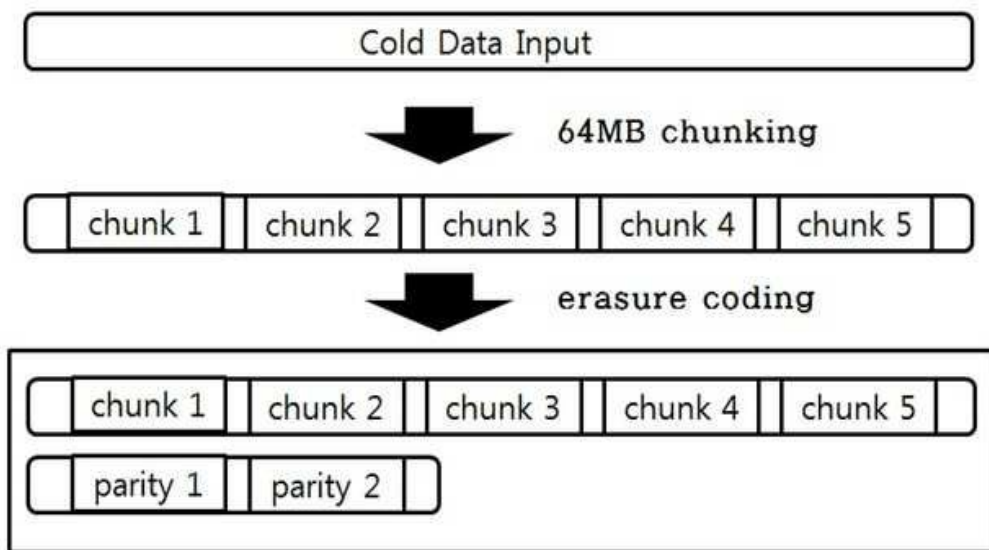
도면4



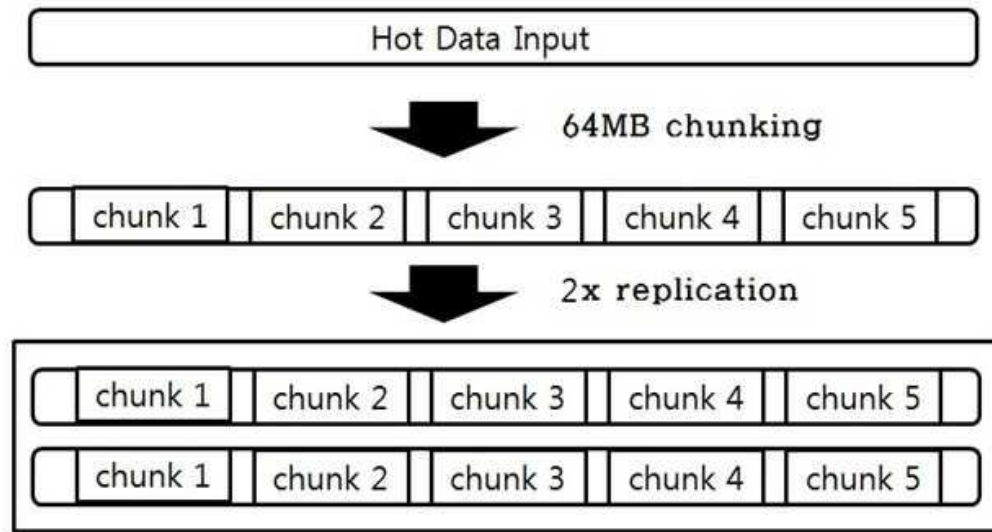
도면5



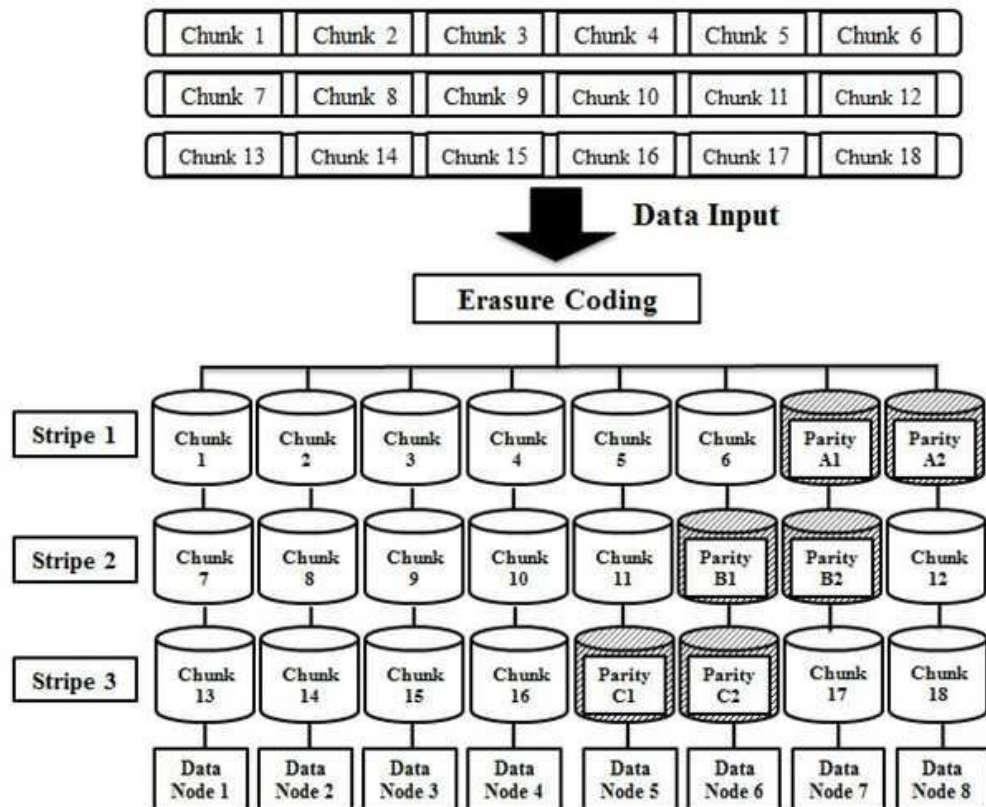
도면6



도면7



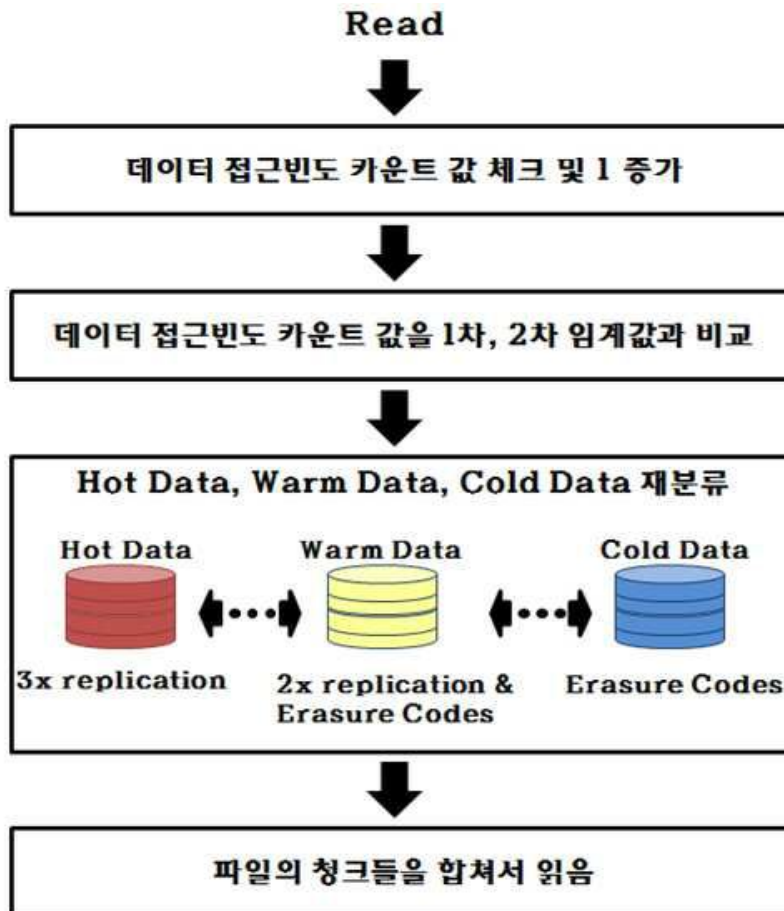
도면8



도면9



도면10



도면11

```

/*   File List :  $F = \{f_0, f_1, f_2 \dots f_m\}$    */
/*    $f_i$  write operation is started   */
1. while
2.   count $i$ ++;
3.    $f_i$  access time check;
4.   if( count $i$  < 1st threshold ) // cold data
5.     generate parity;
6.     write  $f_i$  & parity;
7.   else if( count $i$  < 2st threshold ) // warm data
8.     write  $f_i$  one replication;
9.   else // hot data
10.    write  $f_i$  one replication;
11.    delete parity;
12.   end if
13. end while

/*    $f_i$  access frequency decrease check   */
14. last_access_time(  $f_i$  )
15.   execute  $f_i$  access time interval;
16.   if(Hot data  $f_i$  access frequency decreased)
17.     count $i$  = count $i$  -  $f_i$  access time interval;
18.     if( count $i$  < 2st threshold ) // warm data
19.       generate parity;
20.       write  $f_i$  parity;
21.       delete  $f_i$  one replication;
22.     else if( count $i$  < 1st threshold ) // cold data
23.       delete  $f_i$  one replication;
24.     else // hot data
25.       do nothing;
26.     end if
27.   end if
28. end last_access_time

```

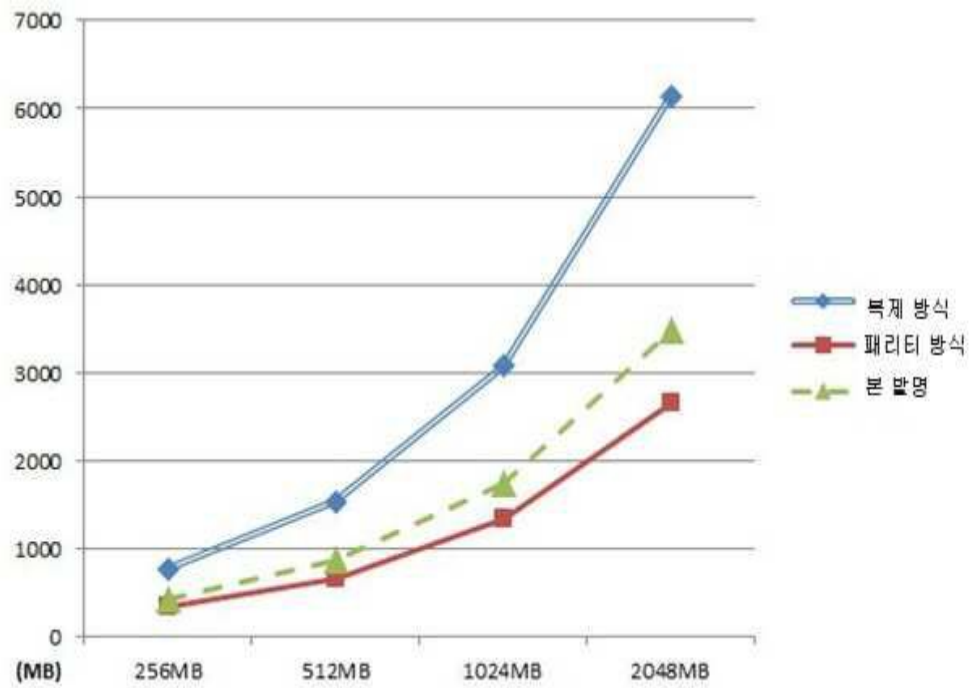
도면12

분류	스펙
SSD model	Samsung SSD 830 series
Capacity	64GB
Controller	Samsung 3-core MCX
Cache memory	256MB DDR2 SDRAM
Sequential Read	520MB/s
Sequential Write	400MB/s
Random Read	80000 IOPS
Random Write	36000 IOPS

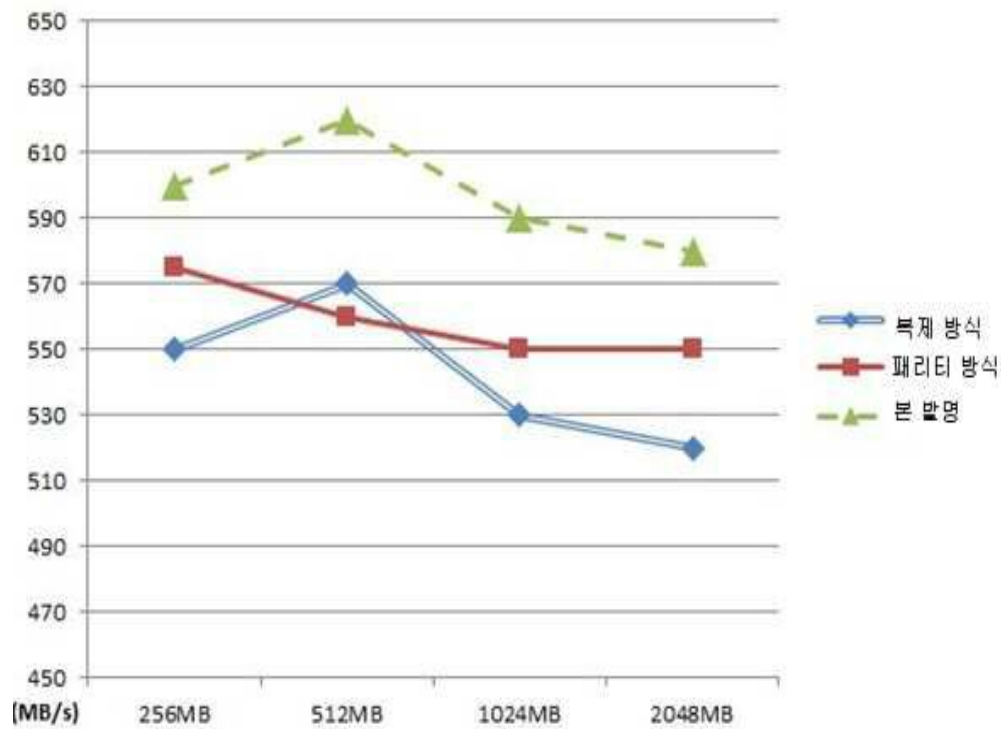
도면13

분류	스펙
CPU	Intel Core 2 (2.4Ghz)
메모리	DDR2 2GB
운영체제	CentOS 6.2
리눅스 커널 버전	3.8.13
저장장치	Samsung SSD 830 series

도면14



도면15



도면16

