논문 2013-50-1-13

SSD 기반의 RAID 시스템에서 패리티 디스크의 중복 제거

(De-duplication of Parity Disk in SSD-Based RAID System)

양 유 석*, 이 승 규*, 김 덕 환**

(Yu-Seok Yang, Seung-Kyu Lee and Deok-Hwan Kim)

요 약

데이터 입출력의 지연 및 병목현상을 해결하기 위해, 여러 개의 디스크를 병렬 구조로 연결한 RAID 시스템이 널리 사용되고 있다. 현재 HDD에 비해 입출력 성능이 좋은 SSD 기반의 RAID 시스템이 활성화 되고 있으나, SSD를 사용하여 RAID 시스템을 구현 할 경우 SSD의 쓰기 횟수 제한 문제와 빈번한 쓰기 연산으로 인한 전력소모의 문제가 발생한다. 본 논문에서는 갱신 비용이 많이 드는 SSD 기반의 RAID 시스템에서 parity 디스크의 중복된 데이터를 제거하는 방법을 제안한다. 제안한 방법은 parity 데이터의 chunk 보다 작은 크기로 분할 하고, 중복된 데이터를 제거 하여 쓰기 연산을 줄이고 마모도 및 전력 소모를 낮춘다. 실험결과 EVENODD 코드를 사용한 RAID-6 시스템의 경우 제안한 방법이 전체 디스크의 약 16%, parity 디스크에서 31% 마모도의 감소를 보였으며, 30% 전력 감소를 보여 중복제거기법을 사용하지 않았을 때 보다 성능이 증가 한 것을 알 수 있다. RAID-5 시스템에서는 전체 디스크의 약 12%, parity 디스크의 32%의 마모도 감소를 보였고, 전력소모의 경우 36%의 전력 소모 감소를 보인다.

Abstract

RAID systems have been widely used by connecting several disks in parallel structure. to resolve the delay and bottleneck of data I/O. Recently, SSD based RAID systems are emerging since SSDs have better I/O performance than HDD. However, endurance and power consumption problems due to frequent write operation in SSD based RAID system should be resolved. In this paper, we propose a de-duplication method of parity disk in SSD based RAID system with expensive update cost. The proposed method segments chunk of parity data into small pieces and removes duplicate data, therefore, it can reduce wear-leveling and power consumption by decreasing write operation for duplicated parity data. Experimental results show that bit update rate of the proposed method is 16% in total disk, 31% in parity disk less than that of existing method in RAID-6 system using EVENODD erasure code, and the power consumption of the proposed method is 30% less than that of existing method. Besides the proposed method is 12% in total disk, 32% in parity disk less than that of existing method in RAID-5 system, and the power consumption of the proposed method is 36% less than that of existing method.

Keywords : SSD, RAID-6, de-duplication, erasure code.

Ⅰ. 서 론

기술의 급속한 발달에 따라 메인 프로세서와 주 기억 장치의 성능이 급격하게 향상된 데 비해 비휘발성 보조 기억 장치인 HDD (Hard Disk Driver)의 경우 공간 수용 능력에 비해 자기 기록 매체의 기계적인 한계로 인

* 학생회원, ** 정회원, 인하대학교 전자공학과
(Department of Electronics Engineering, Inha University)

※ 본 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2012-0001773)
※ 본 과제(연구)는 지식경제부와 한국산업기술진흥원의 전략기술 인력양성사업으로 수행된 결과임
접수일자: 2012년1월16일, 수정완료일: 2012년12월27일

해 속도 면에서 성장하지 못하였다. 이러한 데이터 입출력의 지연 현상을 해결하기 위해, 여러 개의 디스크를 병렬 구조로 연결하여 일부 중복된 데이터를 나눠서 저장하여 데이터의 병목현상을 줄이고 전반적인 속도향상을 가져오기 위한 RAID (Redundant Array of Independent Disks) 시스템이 사용되고 있으며, RAID 시스템의 데이터를 나누어 저장하는 기술은 낮은 가격으로 신뢰성과 성능 향상 이라는 장점을 가지고 있다[1]. 기존의 RAID 방식의 경우 HDD를 사용하였지만, 비휘발성 메모리인 플래시 메모리가 등장하여 SSD(Solid State Disk)가 사용되고 있다. 따라서 현재 SSD를 기반으로 하는 RAID 시스템에 대한 연구가 진행되고 있다[2~3].

SSD는 두 가지의 다른 특성을 가지고 있는데 첫째, 쓰기 횟수의 제한을 가지고 있다. 현재 플래시 메모리는 MLC (Multi Level Cell) 타입의 경우 약 1만번의 쓰기 후에 셀이 마모되는 특징을 가지고 있다. 추후 메모리의 집적도가 더 높아질수록 셀의 내구성이 더 낮아질 수 있다. SSD를 사용하여 RAID를 구축 할 경우 이러한 쓰기 횟수 제한의 문제를 고려해야 한다. 최근에는 하나의 SSD에 쓰기/삭제 연산을 집중시켜 수명저하로 손상된 디스크를 교체해주는 방식의 연구가 진행되었다[2].

둘째, 빈번한 쓰기 연산으로 인해 전력소모가 많게 된다. 플래시 메모리로 구성된 SSD의 경우 기계적인 움직임이 없어 데스크 탑용 HDD보단 전력 소모가 더 낮다. 하지만 노트북 같은 모바일 기기에 들어가는 HDD와는 비슷한 전력 소비율을 보였다[3]. 이처럼 전력 소비율이 비슷한 이유는 쓰기 연산에서 전력소비가 가장 많이 발생하기 때문이다. 플래시 메모리는 제자리 덮어쓰기가 안 되는 문제로 인해, 기존의 데이터를 무효로 표시하고 다른 곳에 쓰는 방법을 사용한다. 또한 쓰기 단위와 삭제 단위가 다른 특징이 있다. 따라서 삭제연산 시 그 전의 데이터를 옮겨야하는 갱신비용이 발생하고, 그로 인해 쓰기연산이 많이 발생하는 문제가 있다[4].

특히 RAID-5 이상의 시스템의 경우 쓰기시 XOR 연산으로 parity 데이터를 생성하는 Read-Modify-Write (R-M-W)의 방식으로 인해 데이터의 작은 쓰기 작업에도 parity 데이터의 갱신이 일어나게 되어 쓰기 연산이 많아져 마모도와 전력소모의 문제가 발생하게 된다[5].

본 논문에서는 SSD의 특징을 이용한 서버단위의 RAID 시스템의 문제점을 해결한다. 우선 RAID 시스템의 chunk-size 의 크기는 SSD의 page 크기보다 크다. 이와 같은 특징 때문에 parity 데이터의 값이 크게 변하지 않았을 경우, 중복 데이터가 생기는 문제가 있다. 따라서 parity 디스크에 chunk 단위 보다 작은 크기로 de-duplication 기법을 적용함으로써 parity 데이터의 기존과 같은 부분의 중복을 줄일 수 있다. 그로 인해, 쓰기 연산이 줄어들게 되고, 마모도 및 전력소모를 낮추는 기법을 제안한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 RAID-6 시스템과 Erasure 코드 중 하나인 EVENODD 코드에 대해 기술한다. Ⅲ장에서는 본 논문의 주요 내용인 parity 디스크의 중복 제거 기법에 대해 기술하고, Ⅳ장에서는 실험을 통해 제안한 방법의 신뢰성 향상 및 전력소모 감소에 대한 실험결과를 보인다. 마지막 Ⅴ장에서는 논문의 결론을 맺는다.

Ⅱ. 관 련 연 구

본 장에서는 기본적인 RAID-6 시스템과 EVENODD 코드에 대하여 설명 한다.

2. 1 RAID-6

RAID-6 시스템은 기존의 RAID-5 시스템이 1개의 디스크 손상시 복구 할 수 있었던 반면에 RAID-6 시스템은 2개의 parity 디스크를 사용함으로써, 공간 효율성 측면에서 조금 떨어지지만, 디스크 2개가 손상되어도 데이터를 복구 할 수 있는 기능을 가지고 있다[5]. 즉

Disk1	Disk2	Disk3	Disk4	Disk 5
D11	D21	D31	P41	Q51
D12	D22	P32	Q42	D52
D13	P23	Q33	D43	D53
P14	Q24	D34	D44	D54
Q15	D25	D35	D45	P55

그림 1. 기본적인 RAID-6 시스템.
Fig 1. The basic RAID-6 System.

RAID-6의 경우 데이터 디스크가 N개 일 때 2개의 parity 디스크, N+2 최소 4개의 디스크를 사용하게 된다. 그리고 parity를 생성하는 복잡한 연산으로 인해 RAID-5에 비해 성능 면에서 조금 떨어진다는 문제점이 있다. 하지만, RAID-5에 비해 높은 신뢰성을 가지는 장점이 있어, 다수의 디스크를 사용하는 환경에서 좋은 성능을 보인다. 그림 1은 기본적인 RAID-6 시스템을 나타낸다.

2. 2 EVENODD 코드

RAID-6의 기법 중 EVENODD 코드는 Reed-Solomon의 사선 패리티를 사용하는 방식과 다르게 RAID-

4의 방식과 같이 패리티 디스크를 독립적으로 사용한다. EVENODD의 특징으로는 XOR 기법만을 사용하여 parity 디스크를 구성하기 때문에 Galois field를 사용하는 Reed-Solomon 기법에 비해 연산이 단순하여 보다 빠른 성능을 가진다. 그림 2는 EVENODD 코드의 기본 구조를 나타낸다[6].

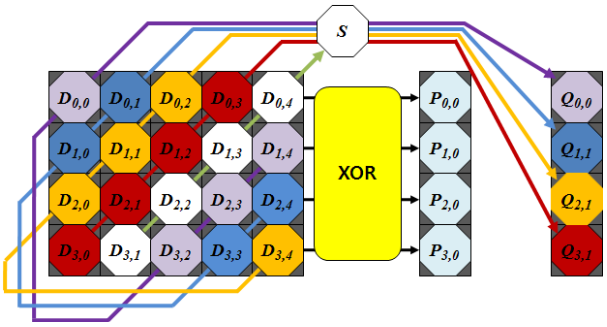


그림 2. EVENODD 코드 구조
Fig 2. EVENODD Code Structure.

첫 번째 P parity는 각 행의 데이터를 XOR 한 값이며, 두 번째 Q parity는 데이터의 대각 패턴을 XOR한 값을 신드롬으로 설정 하여, 대각선의 데이터 값을 신드롬 값과 XOR 시켜 저장한다. 예를 들어, 그림 2의 사선 데이터 $D_{0,2}$ ~ $D_{3,4}$ 데이터 값과 신드롬의 값을 XOR 연산하여 $Q_{2,1}$ 에 저장하게 된다.

Ⅲ. Parity 디스크의 중복 제거

3. 1 SSD를 이용한 RAID 시스템

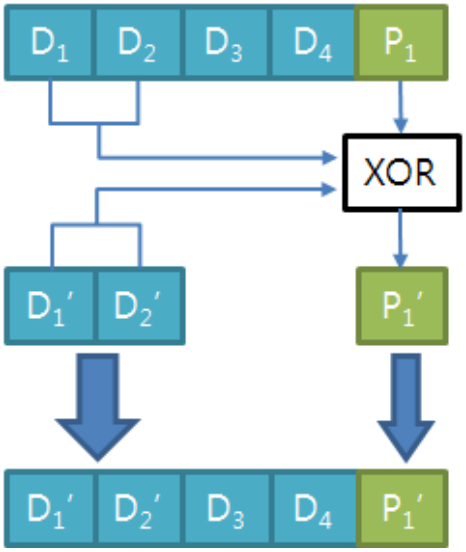


그림 3 Read-Modify-Write 연산
Fig. 3 Read-Modify-Write Operation.

플래시 메모리의 문제점은 쓰기 단위와 지우기 단위가 다르며, 셀 당 지우기 횟수가 정해져 있고, 쓰기 및 삭제 연산의 전력 소모율이 높다는 것이다. 플래시 메모리의 배열로 구성된 SSD의 경우도 마찬가지로 이런 마모도 및 전력 소모의 문제를 가지고 있다. 특히 SSD를 RAID로 구성하였을 경우 RAID 시스템의 쓰기 작업은 R-M-W의 연산을 거쳐 parity 데이터를 갱신 하게 되고, 그로인해 작은 쓰기 갱신의 경우에도 parity 데이터에 대한 쓰기 연산도 같이 이루어지게 되어 parity 디스크의 쓰기 연산이 많아지게 된다. 이렇게 많아진 쓰기 연산은, 플래시 메모리의 마모도 소모를 가져오게 되고, 전력 소비 또한 늘어나는 문제를 가지고 있다. 그림 3은 쓰기 명령이 요청되었을 때 이루어지는 R-M-W 연산을 보여준다.

그림 3과 같이 변경된 데이터 D_1', D_2' 와 기존의 데이터 D_1, D_2 그리고 기존의 parity 데이터 P_1을 XOR 하여 P_1'값을 만들고 기존의 데이터를 갱신 시킨다. 그래서 하나 이상의 데이터가 변경 되었을 때, parity 또한 재계산이 이루어지게 되는 문제점이 있다.

3. 2 중복제거 기법

중복제거기법은 입력되는 데이터에 해시(hash) 값을 부여하고, 기존의 데이터와 비교하여 해시 값이 같으면 동일 데이터로 인식한다. 그래서 동일한 데이터는 데이터를 제거하여 저장하지 않고, 그로인해 공간 활용도를 높이는 방법을 중복제거기법이라고 한다. 또한 데이터

의 중복 제거를 수행하는 방식에 따라 인라인(In-line) 방식과 포스트 프로세스(Post-process)방식으로 분류할 수 있다. 인라인 방식의 경우 데이터가 저장되기 전에 중복제거를 수행하는 방식으로, 디스크 용량을 줄일 수 있다는 장점이 있지만, 연산이 많을 경우 중복제거로 인한 병목 현상이 발생하는 단점이 있다. 포스트 프로세스 방식의 경우 임시 디스크에 저장 후, idle 시간에 중복된 데이터를 찾아 제거해주는 방식이며, 공간 활용 면에서 단점을 가진다. SSD를 이용한 시스템의 경우 디스크에 저장되기 전에 제거하는 인라인 방식을 사용하여 마모도를 줄일 수 있다.

3. 3 제안하는 Parity 디스크의 중복 제거

SSD를 이용하여 RAID 시스템을 구성 하였을 경우 앞서 말한 바와 같이 parity 디스크의 마모도 손상이 커지는 문제를 가지고 있다.

또한 RAID 시스템에서 데이터를 병렬로 저장하기 위해 chunk-size를 정하여 사용한다. 그림4와 같이 chunk-size가 4kb이고, 디스크가 4개 일 때 32kb 크기의 데이터는 4kb의 크기로 8등분 되어 4개의 디스크에 저장되고 5번째 데이터는 다시 첫 번째 디스크에 저장된다. 그러므로 R-M-W의 연산에 의한 갱신비용이 많이 드는 RAID-5 이상의 시스템에서는 chunk-size를 크게 함으로 갱신 비용을 줄일 수 있다. 일반적으로

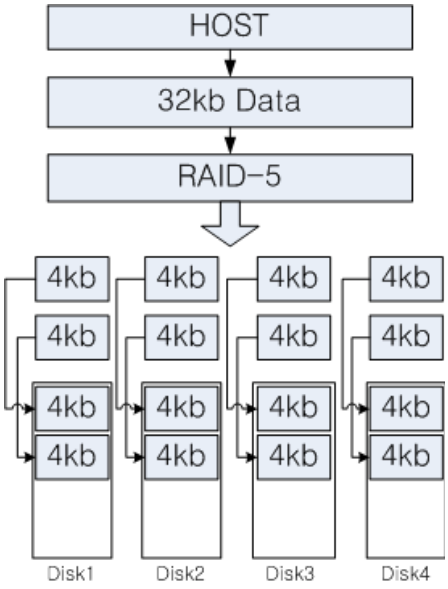


그림 4. RAID 시스템의 데이터 병렬 저장 방법.
Fig 4. Parallel data store method in RAID system.

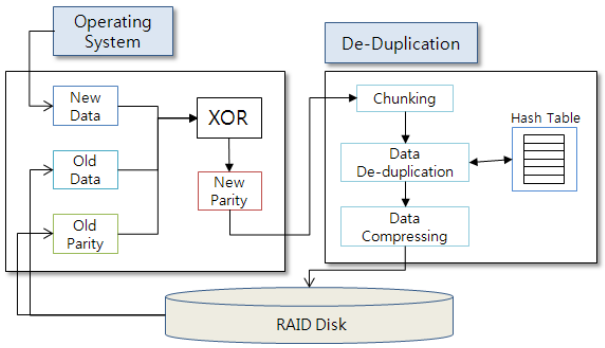


그림 5. 중복제거를 이용하는 RAID 시스템
Fig. 5. RAID system use de-duplication method.

RAID-5이상의 시스템에서는 chunk-size는 128kb 정도가 적당하다.

SSD의 경우 쓰기 단위인 Page 의 크기는 4kb인데 반해 RAID 시스템의 chunk-size는 32kb ~ 128kb로 크다. 따라서 128kb 미만의 작은 쓰기 작업의 요청이 들어왔을 경우 chunk 크기에 대비하여 변경된 parity의 부분이 작을 때 기존의 저장되어 있는 parity 와 중복된 데이터가 저장되어 있을 수 있다. 그로인한 쓰기연산의 갱신 비용이 추가로 들게 된다. 따라서 parity 디스크에 변경된 parity 값이 저장되기 전에 중복제거기법을 사용하여 더 작은 단위의 chunk로 나누어 중복된 데이터를 저장하지 않음으로써, 쓰기 연산을 줄이고 그로인한 전력 소모 및 마모도를 줄일 수 있다.

그림 5는 제안된 RAID 기법의 전체 구조를 보여준다. 그림 5와 같이 새로운 데이터가 들어오면, 기존의 데

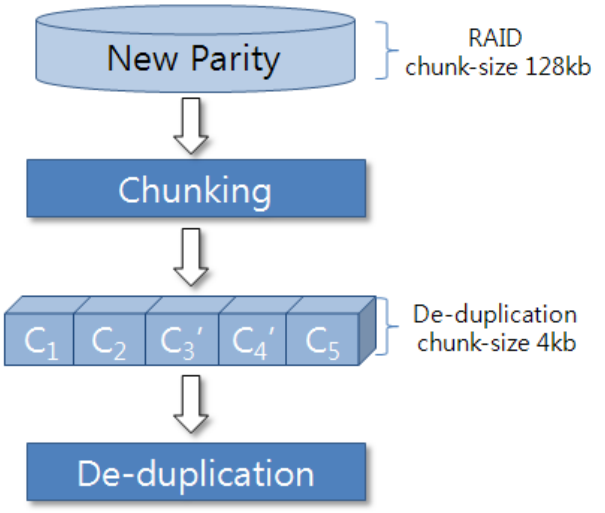


그림 6. Chunking 기법
Fig. 6. Chunking method.

이터와 기존의 parity 값을 읽어서 XOR 연산을 통해 새로운 parity를 생성하게 된다. 새로운 parity 값이 디스크에 저장되기 전 chunking을 수행 한다. 그 후 중복제거 연산을 수행하여, 새로운 해시 값을 만들고 기존의 해시 값과 비교하여, 중복된 데이터를 구별한다.

그림 6은 parity 디스크의 중복제거 기법 중 chunking 과정을 보여준다. RAID parity disk의 Chunk는 XOR 연산을 통해 parity 데이터가 생성되고 다시 4kb의 고정된 값으로 chunking 한다. 다시 chunking 하는 이유는 SSD의 page 크기와 동일하게 하는 것과 더 작은 크기로 조각내어 실제 parity 데이터에서 변경되지 않은 부분의 쓰기 연산을 줄이기 위함이다. chunking 한 후의 중복제거 기법은 그림 7과 같다.

그림 6의 방법을 통해 chunking 된 데이터들을 그림 7과 같이 기존의 데이터들의 해시 값을 읽어 들여 테이블을 만들고, 새로운 데이터는 SHA-1 기법을 사용해 해시 값을 만든다. SHA-1을 이용해 만든 해시 값은 16byte 정도의 크기를 가진다. 이렇게 생성된 값을 해시 테이블을 검색하여 동일한 해시 값이 발견되면 중복된 데이터로 처리한다. 중복된 데이터를 제거한 후 새로 바뀐 해시 값은 플래시 메모리의 Spare 영역을 이용하여 저장한 후 RAID 디스크로 저장한다. Spare 영역에 저장하는 이유는, 해시 값을 데이터로 저장 하게 될 경우 그로인한 데이터의 변화가 있을 수 있기 때문이다. 따라서 Spare 영역에 저장함으로 인해 기존의 데이터는 변화 시키지 않고 저장 할 수 있으며, 해시 테이블 또한 Spare 영역을 읽어 들여서 구현한다.

제안한 방법을 통해 작은 갱신으로 인해 중복된 값이 많던 parity 데이터의 중복된 부분을 저장하기 전에 제거해 줌으로 인해 쓰기 연산의 감소를 가져올 수 있고, 쓰기 연산의 감소로 인해 전력 소모 또한 감소시킬 수 있다.

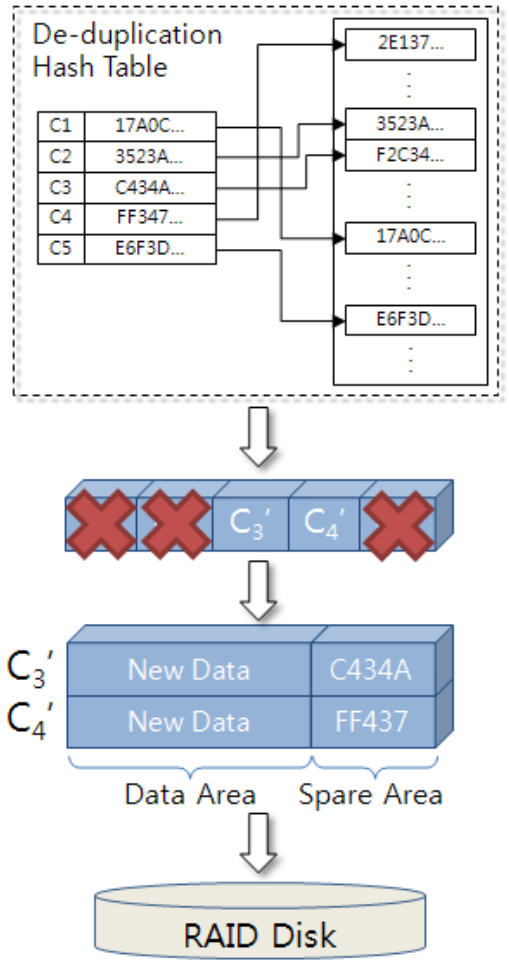


그림 7. 중복제거 기법
Fig 7. De-duplication method.

Ⅳ. 실험 및 성능 평가

제안한 parity 디스크 중복제거 기법의 성능을 평가하기 위하여 EVENODD를 사용하는 RAID-6 기법과 전통적인 RAID-5 기법에 각각 적용 하였다.

본 실험을 통하여 마모도 감소 및 전력 소모, 입/출력 성능에 대하여 측정 및 비교 분석 하였다.

4. 1 실험 환경

리눅스 환경에서 Jerasure 시뮬레이터를 사용하여 Erasure 코드 중 EVENODD를 사용하는 RAID-6와 RAID-5로 각각 구현하였으며, 동일하게 6개의 SSD를

표 1. 실험 환경
Table 1. Experimental Environment.

항목	시스템 사양
CPU	Intel core 2 duo (2.4Ghz)
메모리	DDR2 1G * 3 개
운영체제	Linux kernel 2.6.29
비교 대상	EVENODD, RAID-5
SSD	SSD(16 GB) * 6 개

저장장치로 사용하였다[7]. 실험 데이터로는 일반적인 PC 환경의 데이터들을 이용하여 실험하였다. 표 1은 사용된 실험 환경을 보여준다.

4. 2 EVENODD 코드를 사용한 RAID-6 시스템의 실험 결과

MLC 타입의 SSD로 EVENODD 코드를 사용한 RAID-6 시스템과 RAID-5 시스템을 각각 구축하고, chunk 크기 별로 읽기/쓰기 성능을 비교하였다.

그림 8은 chunk 크기별로 EVENODD를 사용한 RAID-6의 경우 chunk 크기가 128kb나 256kb 일 때 읽기/쓰기 모두 가장 좋았으며, RAID-5의 경우 chunk 크기가 128kb 일 때 좋음을 알 수 있다. 또한 chunk 크기가 512kb 이상 일 때 256kb의 경우 보다 오히려 성능이 저하되는 것을 알 수 있었다. 따라서 RAID 시스템의 chunk 크기를 128kb로 구현 하였다.

그림 9는 EVENODD 코드를 적용한 RAID-6 시스템에서 제안한 방법의 마모도 성능을 보여준다. 제안한 방법이 기존의 방법에 비해 전체 디스크에서 약 16%, parity 디스크에서는 31% 마모도가 감소됨을 알 수 있다.

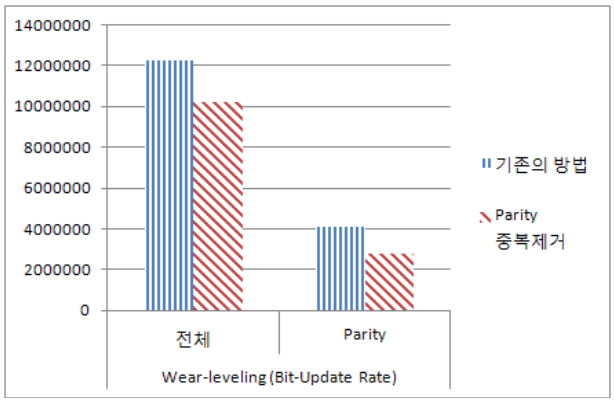
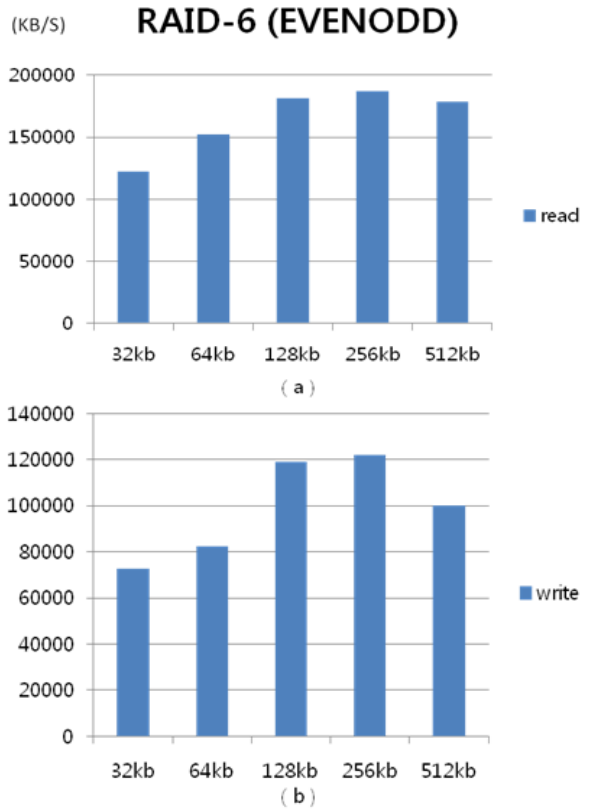


그림 9. RAID-6 에서 제안한 방법의 마모도 비교
Fig 9. Comparison of wear-leveling in RAID-6 system.

그림 10은 EVENODD 코드를 적용한 RAID-6에서 제안한 방법의 전력 소모를 보여준다.

제안한 방법은 parity 중복제거를 적용하지 않은 기



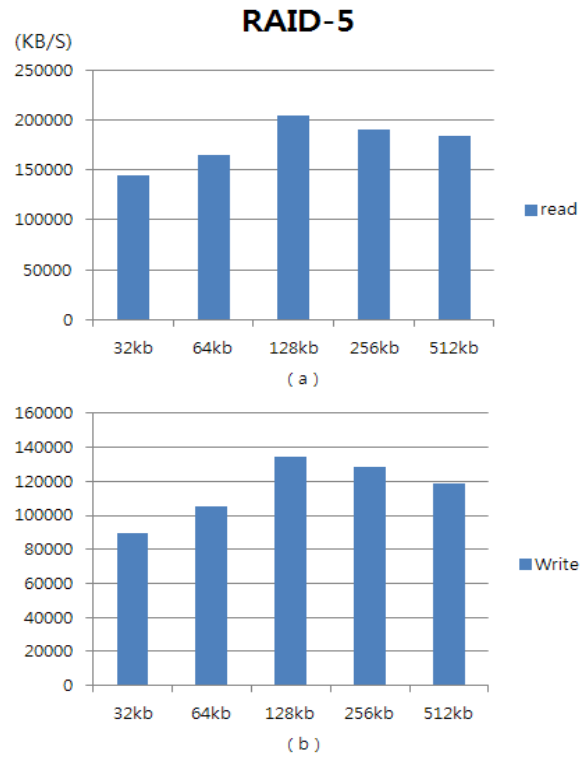


그림 8. Chunk 크기에 대한 읽기/쓰기 성능 비교
Fig 8. read/write performance with respect to chunk-size.

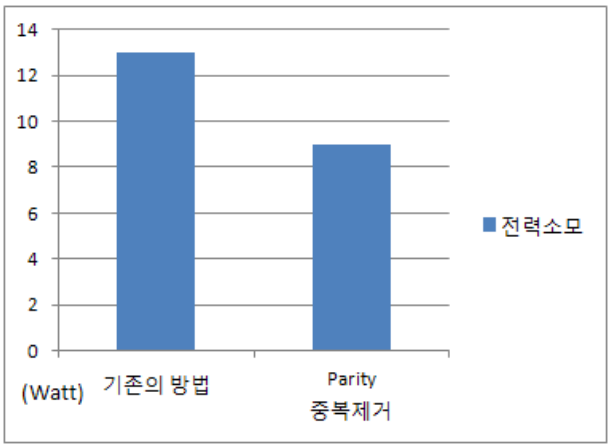


그림 10. RAID-6에서 전력소모 성능 비교.
Fig 10. Comparison of energy consumption in RAID-6.

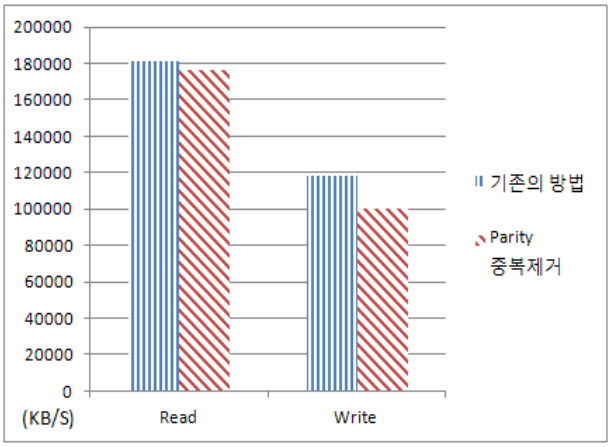


그림 11. RAID-6에서 제안한 방법의 입출력 성능 비교.
Fig 11. Comparison of I/O performance in RAID-6.

존의 방법에 비해 30% 절감된 전력 소모율을 보였다.

그림 11은 EVENODD 코드를 적용한 RAID-6 시스템에서 입출력 성능을 보여준다.

제안한 parity에 중복제거 기법을 적용하였을 경우, 기존의 방법이 읽기는 2%, 쓰기는 18% 정도 더 좋은 성능을 보여주었다. 그 이유는 parity 데이터의 중복제거 연산에 의한 복잡도가 올라갔기 때문이다.

4. 3 RAID-5 시스템의 실험 결과

MLC 타입 SSD를 사용하여 RAID-5 시스템을 구축하고 같은 방법으로 실험 하였다.

그림 12는 RAID-5의 마모도 기법을 비교한 실험이다. RAID-5의 마모도 실험에서는 제안한 방법이 전체 디스크에서 12% 정도 parity 디스크에서 32%정도의 감소율을 보였다. 전력 소모의 경우 그림 13과 같이 제안

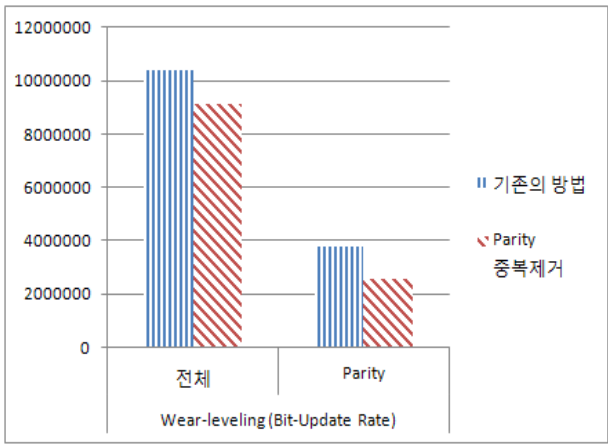


그림 12. RAID-5 에서 제안한 방법의 마모도 비교
Fig 12. Comparison of wear-leveling in RAID-5 system.

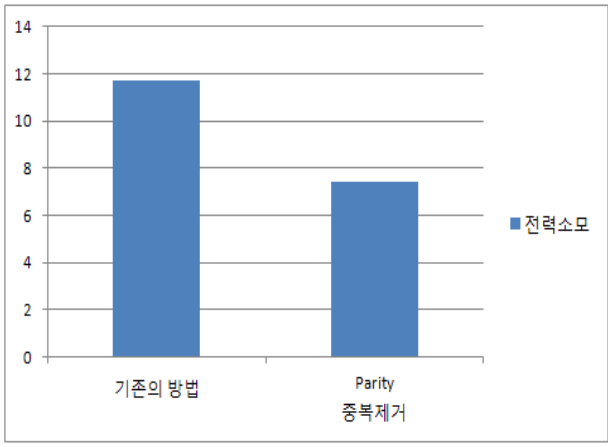


그림 13. RAID-5에서 전력소모 성능 비교.
Fig 13. Comparison of energy consumption in RAID-5.

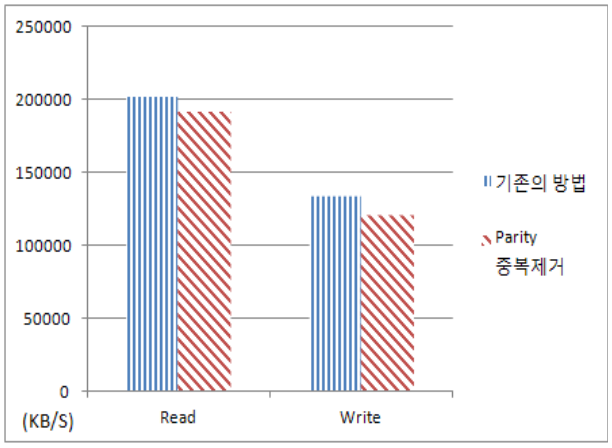


그림 14. RAID-5에서 제안한 방법의 입출력 성능 비교
Fig 14. Comparison of I/O performance in RAID-5.

한 방법이 36%정도 전력 소모가 감소한 것을 볼 수 있었다.

그림 14는 RAID-5에서 제안한 방법의 성능을 보여준다.

RAID-5 역시 연산 복잡도로 인한 성능저하로 인해 제안한 방법이 기존의 방법보다 읽기는 5%정도 쓰기는 10% 정도 낮은 성능을 보였다.

V. 결 론

본 논문에서는 SSD 기반의 RAID 시스템에서 갱신이 많은 parity 디스크에 RAID의 chunk 크기보다 작은 단위의 chunk 크기를 적용하여, 변하지 않은 parity 데이터의 중복된 데이터를 제거하여 디스크의 마모도를 줄이고 쓰기 연산을 낮추어 전력 소모를 감소시키는 방법을 제안한다.

RAID-5 시스템과 EVENODD 코드를 사용한 RAID-6 시스템에서 비교 했을 때, 읽기/쓰기 성능에서는 제안한 방법의 연산 복잡도가 올라가는 이유로, 기존의 방법이 제안한 방법에 비해 조금 나은 성능을 보였지만, 마모도와 전력소모의 경우 기존의 방법보다 성능이 향상되었음을 확인할 수 있었다.

참 고 문 헌

[1] D. Patterson, G. Gibson, and R. Katz, “A Case for Redundant Arrays of Inexpansive Disks (RAID),” In Proc. of ACM SIGMOD Conf. on Management of Data, pp.109-116, 1988.

[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) colocated with SOSP, October 2009.

[3] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, A. Rowstron. “Migrating server storage to SSDs: analysis of tradeoffs,” In Proc. of the 4th ACM European Conference on Computer systems (Eurosys ′09), pp. 145-158, 2009.

[4] B. Yoo, Y. Won, S. Cho, S. Kang, J. Choi, S. Yoon, “SD Characterization: From Energy Consumption’s Perspective,” In Proc. of the 3rd USENIX conference on Hot Topic in Storage and File Systems (HotStorage′ 11) June, 2011.

[5] E. Lee, R. Kartz, “Performance Consequences of parity Placement in Disk Arrays,” In Proc. of ASPLOS-IV Prec. pp.190-199, 1999.

[6] Blaum, M., Brady, J., Bruck, J., and Menon, J. EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. IEEE Transactions on Computing 44, 2, Feb. 1995.

[7] James S. Plank, “The RAID-6 liberation codes”, In Proc. of the 6th USENIX Conference on File and Storage Technologies, San Jose, California, pp.1-14, February 2008.

저 자 소 개

양 유 석(학생회원)
2012년 인하대학교 전자공학과 석사졸업
2012년~현재 LG전자 CTO Convergence R&D Lab.
<주관심분야 : 임베디드 시스템, 스토리지 시스템>

이 승 규(학생회원)
2011년 대진대학교 통신공학과 학사 졸업.
2011년~현재 인하대학교 전자공학과 석사과정
<주관심분야 : 임베디드 시스템, 스토리지 시스템>

김 덕 환(정회원)-교신저자
2003년 한국과학기술원 컴퓨터 공학 박사.
2006년~현재 인하대학교 전자공학부 교수
<주관심분야 : 시각정보처리, 스토리지 시스템, 임베디드 시스템>