2014년 2월 전자공학회 논문지 제 51권 제 2 호
Journal of The Institute of Electronics and Information Engineers Vol. 51, NO. 2, February 2014 http://dx.doi.org/10.5573/ieie.2014.51.2.085

논문 2014-51-2-12

클라우드 스토리지 시스템에서 데이터 접근빈도와 Erasure Codes를 이용한 데이터 복제 기법

(Data Access Frequency based Data Replication Method using Erasure Codes in Cloud Storage System)

김 주 경*, 김 덕 환**

(Ju-Kyeong Kim and Deok-Hwan Kim©)

요 약

클라우드 스토리지 시스템은 데이터의 저장과 관리를 위해서 분산 파일시스템을 사용한다. 기존 분산 파일시스템은 데이터 디스크의 손실 발생시 이를 복구하기 위해서 3개의 복제본을 만든다. 그러나 데이터 복제 기법은 저장공간을 원본 파일의 복제 횟수만큼 필요로하고 복제과정에서 입출력 발생이 증가하는 문제가 있다. 본 논문에서는 SSD 기반 클라우드 스토리지 시스템에서 저장공간 효율성 향상과 입출력 성능 향상을 위하여 Erasure Codes를 이용한 데이터 복제 기법을 제안한다. 특히, 데이터 접근 빈도에 따라 복제 횟수를 줄이더라도 Erasure Codes를 사용하여 데이터 복구 성능을 동일하게 유지하였다. 실험 결과 제안한 기법이 HDFS 보다 저장공간 효율성은 최대 약40% 향상되었으며, 읽기성능은 약11%, 쓰기성능은 약10% 향상됨을 확인하였다.

Abstract

Cloud storage system uses a distributed file system for storing and managing data. Traditional distributed file system makes a triplication of data in order to restore data loss in disk failure. However, enforcing data replication method increases storage utilization and causes extra I/O operations during replication process. In this paper, we propose a data replication method using erasure codes in cloud storage system to improve storage space efficiency and I/O performance. In particular, according to data access frequency, the proposed method can reduce the number of data replications but using erasure codes can keep the same data recovery performance. Experimental results show that proposed method improves performance in storage efficiency 40%, read throughput 11%, write throughput 10% better than HDFS does.

Keywords : Cloud Storage, SSD, Hadoop File System, Distributed File System, Erasure Codes

Ⅰ. 서 론

전 세계에서 PC, 모바일, 노트북 기기의 사용으로 생성되는 데이터양이 2012년에는 2.8ZB(제타바이트, Zeta Byte)이다. 향후 2020년까지 40ZB가 생성될 것이라는 전망이 나왔으며 데이터 규모가 급격하게 늘어남에 따라 빅데이터 처리가 큰 이슈가 되고 있다. 클라우드 스토리지에서 생성된 방대한 빅데이터를 저장하기 위해선 많은 양의 저장공간을 필요로 하게 된다.

* 학생회원, ** 정회원, 인하대학교 전자공학과
(Department of Electronic Engineering, Inha University)
© Corresponding Author(E-mail: deokhwan@inha.ac.kr)
※ 이 논문은 2013년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (2013R1A1A2006912)
※ 본 연구는 미래창조과학부 및 정보통신산업진흥원의 IT/SW 창의연구과정의 연구결과로 수행되었음 (NIPA-2013-H0502-13-1016)
※ 본 논문은 인하대학교 연구비 지원에 의하여 연구되었음
접수일자: 2013년11월8일, 수정완료일: 2014년2월1일

빅데이터를 효과적으로 저장 및 관리하기 위해서 클라우드 스토리지 시스템은 분산 파일시스템을 사용하여 여러 서버 및 노드들을 네트워크로 연결하여 처리함으로써, 고가용성과 높은 확장성을 제공한다[1]. 분산 파일시스템 중에 아파치(Apache) 재단에서 개발한 오픈소스인 하둡 분산 파일시스템(HDFS)은 클라우드 스토리지 시스템에 적용되어 많이 사용되고 있다. HDFS는 저장공간의 신뢰성을 위해서 데이터를 64MB의 블록 단위로 청킹한 후, 3개의 복제본을 생성하여 각 블록들을 데이터 노드에 분산하여 저장한다[2]. 그림1은 HDFS의 파일 복제과정의 문제점을 보여주고 있다. 만약 320MB의 데이터가 들어오면 64MB의 블록 단위로 데이터를 청킹한 후, 3개의 복제본을 생성하고 분산하여 저장한다. 이 방법은 원본 데이터 크기에 3배의 저장공간이 필요하기 때문에 데이터가 커질수록 저장공간 낭비가 심해진다[3].

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[4]. 따라서 임의 I/O가 많이 발생하는 클라우드 환경에서는 HDD가 병목현상을 발생시키는 요인이 된다.

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

따라서 본 논문에서는 클라우드 스토리지 시스템에서 데이터 저장공간의 효율성과 입출력 속도 향상을 위해서 데이터 접근빈도에 따라 Erasure Codes를 이용한 데이터 복제 기법을 제안한다.

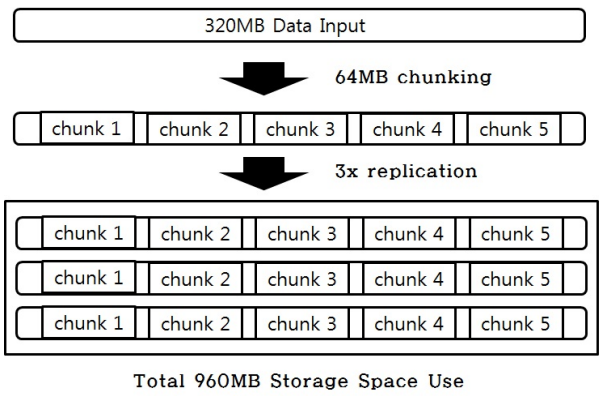

그림 1. HDFS의 비효율적인 저장공간 사용
Fig. 1. nefficient Storage Space Usage in HDFS.

Ⅱ. 관련 연구

1. 하둡 분산 파일시스템(HDFS)

하둡 분산 파일시스템(HDFS)은 대량의 자료를 처리할 수 있게 해주는 오픈소스 소프트웨어이다. HDFS는 하나의 네임노드와 다수의 데이터 노드로 구성된다. 네임노드는 HDFS의 네임스페이스를 관리하면서 클라이언트의 파일 접근 요청을 처리한다. HDFS의 데이터 노드는 클라이언트의 데이터 입출력 요청을 기본 64MB 블록 단위로 저장한다. 또한, 컴퓨터 클러스터에서 복제본을 만들어 저장장치 중 일부의 손실을 빨리 자동으로 복구하는 기능을 제공한다. 복구 기능은 사용자가 지정한 만큼의 데이터 복제본을 만들어 동작하는데 통상 3개의 복제본을 만든다. 이러한 3개의 복제본 생성은 데이터 규모가 커질수록 저장공간 사용량이 증가하게된다. HDFS는 클라우드 환경에 적합하지만 3개의 복제본을 만듬으로써 저장공간 소모가 크다는 단점이 있다.

2. RAID를 사용한 하둡 분산 파일시스템 (HDFS-RAID)

HDFS-RAID는 기존의 HDFS에 데이터 복제 방법(Data Replication Method) 대신 RAID를 적용한 분산 파일시스템이다. HDFS-RAID는 DRFS(Distributed RAID File System)이라고도 불리며 아파치 재단에서 연구되었다[5].

HDFS-RAID는 신뢰성을 위해 데이터의 복제본을 만들지 않고 RAID 6의 스트라이프를 만들어 패리티 블록을 같이 저장한다. 데이터 손실시에는 패리티 블록으로부터 데이터 복구과정을 거쳐서 원본 데이터를 복원한다. HDFS-RAID는 기존 HDFS의 3개의 복제본을 생성하는 방법과 비교했을 때 동일하게 2개의 손실을 복구할수 있지만 저장공간을 적게 사용한다. 따라서 데이터 저장량이 많은 클라우드 스토리지 시스템에 적용시 효과적이다.

일례로 Facebook에서 HDFS-RAID를 적용하여 저장공간 효율성을 향상한 사례가 있다. Facebook은 전세계 약 10억명 이상의 유저가 활동하는 세계 최대의 소셜 네트워크 서비스로 하루 평균 25억개의 컨텐츠가 공유

되고 500TB 이상의 데이터를 처리한다. 이처럼 Facebook에서 매일 생성되는 방대한 데이터가 쌓여서 빅데이터를 형성하게 되는데, Facebook에서는 HDFS-RAID를 도입하였고 이를 통해 약 5PB의 용량을 감소시키는 성과를 달성하였다. Facebook의 사례는 데이터 복제 방식에 Erasure Codes를 적용할 경우 저장공간 사용이 효율적이라는 것을 보여준다.

3. Erasure Codes

Erasure Codes는 데이터와 인코딩 과정을 통해 생성한 코드들을 저장하여 데이터 손실시 디코딩 과정을 거쳐 원본 데이터를 복구하는 기법이다[6]. 뛰어난 오류 복구 성능과 입출력 성능으로 인해서 데이터 신뢰성이 중요한 시스템에 Erasure Codes를 사용한다. Erasure Codes로 생성된 패리티가 데이터 복제본 생성보다 적은 저장공간을 차지하므로 신뢰성을 제공하면서 저장공간 효율성 또한 높일 수 있다.

Erasure Codes의 종류로는 Reed-Solomon Code[7], EVENODD code[8], Weaver code[9], X-code[10]등 다양한 코드가 있다. 각 Erasure Codes 별로 다른 알고리즘을 사용하며 연산 복잡도를 줄이면서 복구 성능을 높이기 위한 연구가 진행되고 있다.

Ⅲ. 데이터 접근빈도에 따라 기존의 Erasure Codes를 이용한 데이터 복제 기법

본 논문에서는 SSD기반 클라우드 스토리지 시스템에서 데이터 접근빈도에 따라 기존의 Erasure Codes를 이용한 데이터 복제 기법을 제안한다.

기존의 클라우드 스토리지에서의 데이터 복제 기법은 데이터 손실로 부터 신뢰성을 유지하기 위해 3개의 복제본을 생성하기 때문에 저장공간 낭비를 유발한다. 제안한 방법은 저장공간 효율성과 입출력 속도를 높이기 위해서 데이터 접근빈도에 따라 세 가지 다른 복제 방법을 사용한다.

1. 접근빈도에 따른 새로운 데이터 복제 기법

제안한 기법은 접근빈도에 따라 데이터를 Hot Data, Warm Data, Cold Data로 분류한다. Hot Data는 접근빈도가 높은 자주 사용되는 데이터로 데이터 손실시에 강건한 신뢰성을 보장하기 위해서 그림 2와 같이 3개의 복제본을 생성한다[11]. Warm Data는 Hot Data 보다 접근 빈도가 낮은 데이터이지만 추후 클라이언트의 잦은 사용으로 Hot Data가 될 가능성이 높은 데이터이다. 따라서 Warm Data는 그림 3과 같이 2개의 복제본 생성과 Erasure Codes를 사용하여 신뢰성을 확보한다. Cold Data는 접근 빈도가 가장 낮아 자주 사용되지 않는 데이터로 그림 4와 같이 Erasure Codes만을 사용해 데이터 손실에 대비한다.

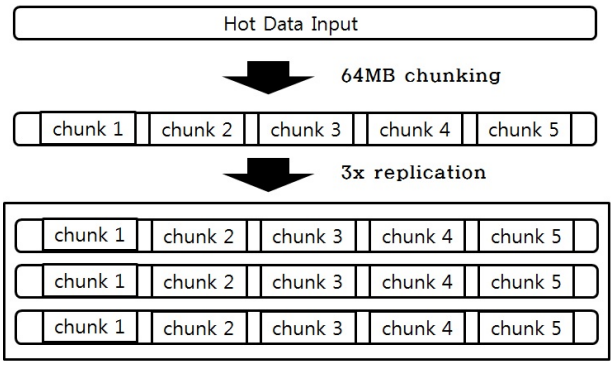

그림 2. Hot Data의 데이터 복제
Fig. 2. Data Replication of Hot Data.

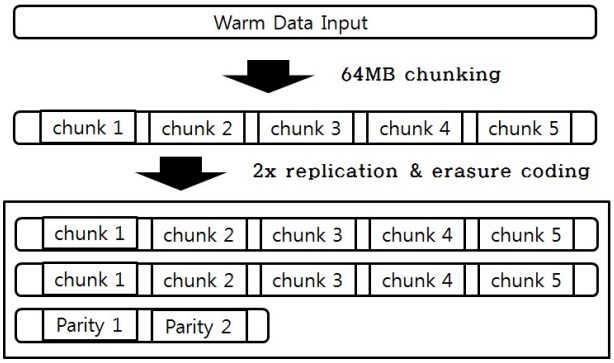

그림 3. Warm Data의 데이터 복제
Fig. 3. Data Replication of Warm Data.

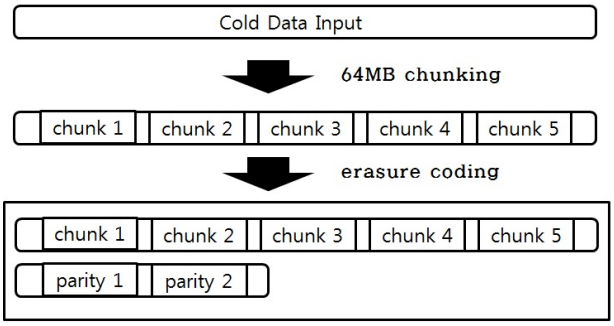

그림 4. Cold Data의 데이터 복제
Fig. 4. Data Replication of Cold Data.

Hot Data는 일반적으로 전체 데이터에서 차지하는 비율이 상대적으로 낮기 때문에 3개의 복제본을 생성하더라도 저장공간 효율성이 크게 떨어지지 않는다. Hot

Data에 비해 접근빈도가 낮은 Warm Data, Cold Data의 경우 데이터 양이 많기 때문에 저장공간 효율성을 위해 Erasure Codes를 적용하여 패리티를 생성한 후 저장한다. 그림 3, 4는 Erasure Codes를 적용했을 때의 데이터 크기를 보여준다.

그림2와 비교해 보면, 복제방식은 데이터에 3배의 저장공간이 필요하지만, Cold Data일 경우 Erasure Codes를 적용하면 데이터에 1.4배의 저장공간이 필요하다. 또한, 저장속도도 패리티 연산시간이 있지만 3개의 복제본을 생성하는 것보다 적은 시간이 소모되기 때문에 쓰기 속도도 향상 된다.

2. 기존 Erasure Codes를 이용한 데이터 복제 기법

Warm data와 Cold data를 위한 데이터 복제 기법은 원본 데이터와 Erasure Codes를 이용하여 패리티 정보를 저장한다. 그림5는 데이터 복제를 위해 RAID-6와 Erasure Codes를 사용하는 예를 보여준다. RAID-6는 두 개의 패리티 블록 저장하여 두 개의 데이터 블록이 손실될 경우 복구할 수 있게하는 기능을 하고 Erasure Codes는 원본데이터에 연산 과정인 인코딩을 거쳐 패리티 정보를 생성하는 역할을 한다[12].

Erasure Codes를 이용한 데이터 복제 기법의 쓰기 과정은 먼저 입력 데이터를 청킹하는 과정까지 거친 후에 그림5에서 처럼 각 청크들을 Erasure Coding 과정을 거쳐 패리티 청크를 생성한다. 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 하며 각각의 스트라이프는 패리티 청크가 배치될 데이터 노드를 변경하면서 저장하여 하나의 데이터 노드에 패리티 청크가 몰리는 현상을 방지한다.

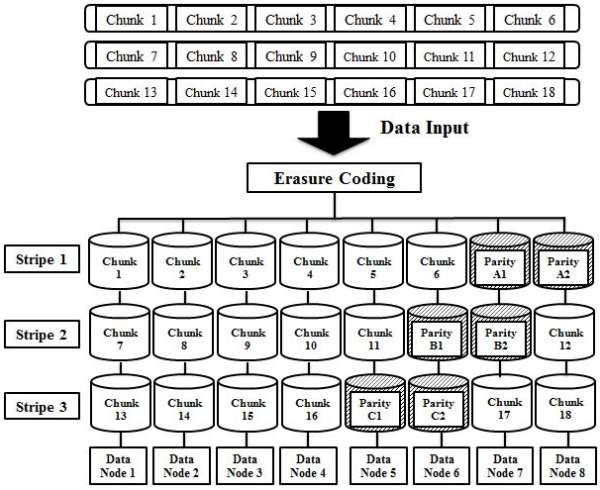

그림 5. Erasure Codes를 이용한 데이터 복제기법
Fig. 5. Data Replication Method using Erasure Codes.

3. 접근빈도에 따른 Erasure Codes를 이용한 데이터 복제 알고리즘

그림 6은 접근빈도에 따라서 Erasure Codes를 이용하는 데이터 복제 기법의 순서도이고 표 1은 제안한 데이터 복제 알고리즘을 보여준다. 먼저 시스템에 쓰기 명령이 들어오면 64MB 크기의 블록 단위로 데이터를 청킹한다. 쓰기 요청된 데이터에 대해 메타데이터 영역에 해당 데이터의 접근빈도 카운트 값과 저장방식을 저장한다. 처음 쓰기 명령된 파일은 접근횟수가 없어 Cold Data가 되므로 Erasure Coding을 적용하여 원본

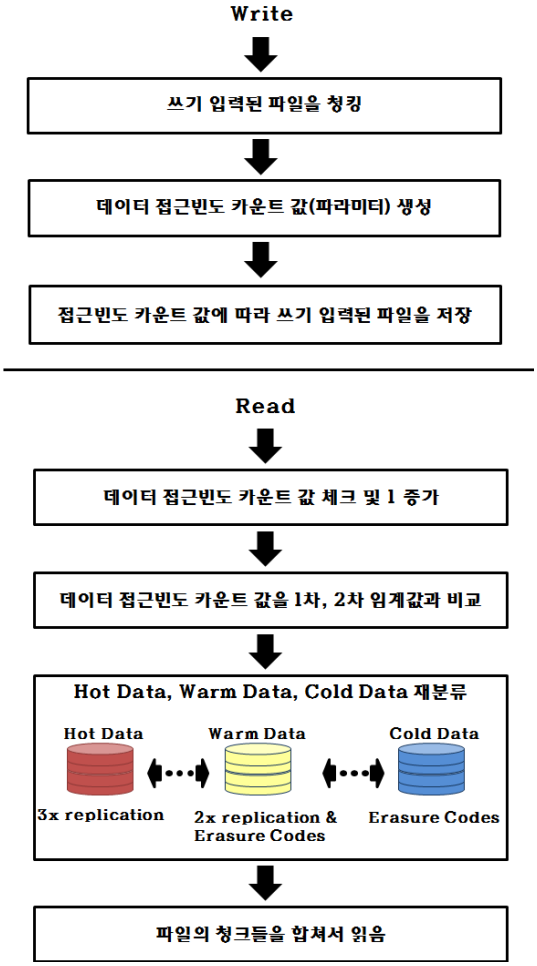

그림 6. Erasure Codes를 이용한 접근빈도에 따른 데이터 복제 기법의 순서도
Fig. 6. Sequence of Data Replication Method using Erasure Codes according to Data Access Frequency.

파일과 패리티 정보를 저장한다. 그리고 각 파일에 읽기 명령이 들어오면, 메타데이터를 참조하여 데이터가 어느 방식으로 저장 되어 있는지 확인하고 주소값을 확인한다. 그리고 접근횟수가 늘어났기 때문에 메타데이터에서 접근빈도 카운트(Access Frequency Count) 값을 1씩 증가시킨다. 그 후 파일을 구성하는 청크들을 모아 읽기 요청된 데이터를 제공한다.

만약, 읽기 명령이 빈번히 발생해서 접근빈도 카운트 값이 증가하여 1차 임계값과 같거나 커지면 Warm Data가 되어 신뢰성을 위해 원본 파일의 복제본을 1회 더 저장한다. 접근빈도 카운트 값이 더 증가하여 2차 임계값과 같거나 커지게 되면 Hot Data가 되어 복제본을 1회 더 저장한다.

반대로 일정 시간 동안 파일에 대한 읽기 명령이 발생하지 않는다면 접근빈도 카운트값을 감소시켜 저장공간 효율성 향상을 위해 Erasure Coding 방법으로 저장한다. 접근빈도 카운트값이 2차 임계값보다 작아지면 복제본을 하나 삭제하여 Hot Data를 Warm Data로 바꾸고 Erasure Codes의 패리티 정보를 생성한다. 1차 임계값보다 작아지면 복제본을 하나더 삭제하여 Warm Data를 Cold Data로 바꾸고 원본 데이터와 패리티 정보만 남게된다.

Ⅳ. 실험 및 성능평가

제안한 기법의 성능평가를 위해 워크로드 데이터 크기를 256MB, 512MB, 1GB, 2GB로 각각 설정했다. 워크로드의 크기가 작아서 청크사이즈는 4KB 크기로 청킹하였다. 데이터 저장방식을 결정하는 기준인 임계값은 각 실험 전에 무작위로 읽기 명령을 내려 각 데이터의 접근빈도 카운트 값을 확인하였다. Hot Data, Warm Data, Cold Data의 비율은 실제 서버에서 추출한 워크로드에서 얻은 비율을 이용하였다[13~15]. 각 참고논문에서는 Financial1, MSR, Distilled, RealSSD, digital camera, Linux O/S의 워크로드를 사용하였다. 각 워크로드에서 Hot Data의 비율은 평균이 10%였으며 따라서 실험에서도 Hot Data의 비율을 10%로 정하였다. Warm Data는 참고논문에서 Hot Data의 비율이 30%까지 올라가는 점을 참고하여 20%로 비율을 정하였다. 실험에서 Hot Data가 차지하는 비율은 전체 데이터의 약10%가 되도록 설정하였고, Warm Data는 전체 데이터의 약 20%, Cold Data는 약70%가 되도록 했다. 패리티생성 방식은 Reed-Solomon 코드를 적용하여 패리티

표 1. Erasure Codes를 이용한 접근빈도에 따른 데이터 복제 알고리즘
Table 1. Data Replication Algorithm using Erasure Codes according to Data Access Frequency.

```
/*   File List : F = { f_0, f_1, f_2 ··· 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
```

표 2. 실험에 사용한 SSD의 스펙 정보
Table 2. Specification Information of SSD using in Experiments.

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

비트를 생성했다. 복제시에는 기존의 분산 파일 시스템과 같이 3개의 복제본을 만들어 저장했다.

실험환경은 다음과 같다. 벤치마크 툴로 IOzone을 사용하여 다양한 크기의 워크로드 데이터를 생성하였으며, HDFS의 최소 구성 개수인 3대의 PC를 사용하였다.

표 3. 실험환경
Table 3. Experimental Environment.

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

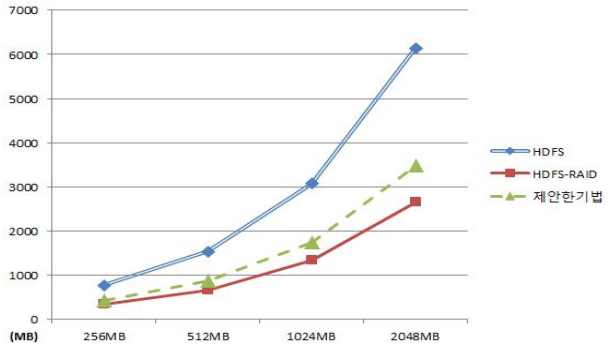

그림 7. 저장공간 효율성 실험 성능 비교
Fig. 7. Experimental Performance Comparison of Storage Space Efficiency.

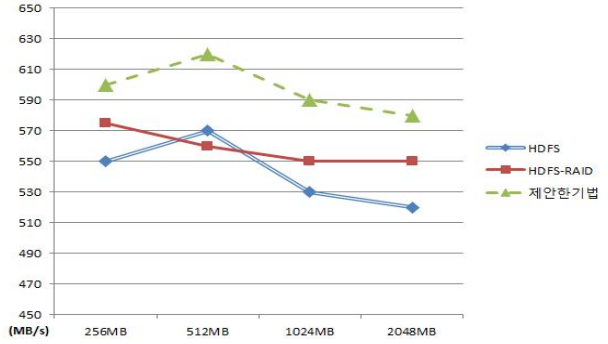

그림 8. 데이터 쓰기 실험 성능 비교
Fig. 8. Experimental Performance Comparison of Write Throughput.

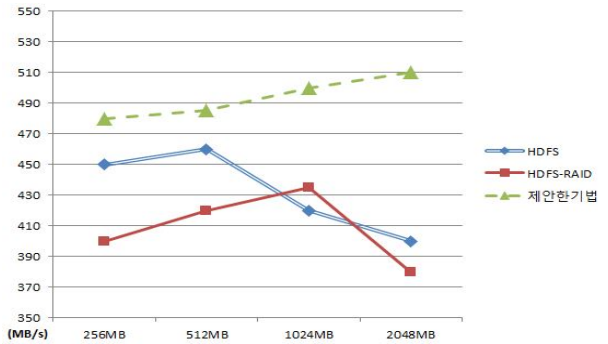

그림 9. 데이터 읽기 실험 성능 비교
Fig. 9. Experimental Performance Comparison of Read Throughput.

각 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 각 PC에 2개씩 장착하였다. 사용된 SSD의 스펙은 표 2와 같고, 실험에 사용된 컴퓨터의 성능 및 환경은 다음 표 3과 같다.

실험 결과는 그림 7, 8, 9와 같다. 각각 저장공간 효율성, 쓰기/읽기 성능을 비교하였으며 데이터 복제방식만을 사용하는 HDFS와 패리티 방식만을 사용하는 HDFS-RAID를 제안한 기법의 비교대상으로 하였다.

실험결과 그림 7의 저장공간 효율성 면에서는 제안한 기법이 HDFS보다 약 40%정도 적게 저장공간을 사용하였다. HDFS는 3개의 복제본을 만들기 때문에 가장 저장공간을 많이 소모하고 HDFS-RAID는 Erasure Codes만을 사용하기 때문에 가장 저장공간을 적게 차지한다. 제안한 기법은 이 두가지 사이에 위치하며 HDFS-RAID에 근접한 수치를 보여주어 저장공간 효율성이 향상됨을 알 수 있었다.

그림 8, 9는 각각 쓰기 성능과 읽기 성능을 비교한 그래프이다. 쓰기 성능에서는 제안한 기법이 HDFS 보다 약10%, HDFS-RAID 보다 약7% 향상되었다. 읽기 성능에서는 제안한 기법이 HDFS 보다 약11%, HDFS-RAID 보다 약12% 향상되었다. 실험결과 제안한 기법은 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.

V. 결 론

본 논문에서는 클라우드 스토리지 시스템에서 원본 데이터에 3개의 복제본을 저장하는 방법에서 발생하는 저장공간 낭비와 입출력이 증가하는 문제를 해결하기 위해 데이터 접근빈도에 따라 기존의 Erasure Codes를 이용한 데이터 복제기법을 제안하였다. 기존의 Erasure Codes를 데이터 접근빈도에 따라 사용하여 3개의 복제본을 생성하는 방법과 동일한 복구성능을 유지하였다. 실험에서는 3개의 데이터 복제본을 생성하는 방법과 패리티만을 생성하는 방법을 비교하였다. 데이터 접근빈도를 Hot, Warm, Cold로 세분화하여 나눔으로써 Erasure Codes를 이용한 데이터 복제 기법이 HDFS 보

다 저장공간 효율성은 40%, 읽기 성능 11% 및 쓰기 성능 10% 향상되었다.

REFERENCES

[1] D. J. Abadi, "Data Management in the Cloud: Limitations and Opportunities," in Proc. of IEEE Conf. on Data Engineering, pp.1~10, Shanghai, China, March 2009.
[2] K. Shvachko, H. Kuang, S. Radia, "The Hadoop Distributed File System," in Proc. of IEEE Conf. on Mass Storage System and Technologies, pp.1~10, Santa Clara, California, USA, May 2010.
[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.
[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.
[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.
[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.
[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.
[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.
[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.
[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.
[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 IEEK Conf. on Summer Conference, Vol.36, No.1, pp.1539~1542, Jeju, Korea, July 2013.
[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 IEEK Conf. on Summer Conference, Vol.36, No.1, pp.1543~1546, Jeju, Korea, July 2013.
[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.
[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.
[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.

저 자 소 개

김 주 경(학생회원)
2009년 인하대학교 전자공학과 학사 졸업.
2014년 인하대학교 전자공학과 석사 졸업.
2014년~현재 LG전자 CTO SIC 연구소 연구원
<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, BCI>