논문 2010-47CI-4-11

# 파일 시스템 스냅샷

## ( File System Snapshot )

석 진 선*, 노 재 춘*

( Jinsun Suk and Jaechun No )

요 약

IT 기술이 발전하면서 반드시 유지해야하는 중요한 데이터들 또한 스토리지에 저장되기 시작했다. 이러한 현상은 저장된 데이터를 보호하기 위한 백업 작업의 중요성을 증가시켰으며 데이터의 양이 증가하면서 백업 작업의 수행 시간 또한 중요한 문제로 대두되었다. 스냅샷은 데이터 일관성을 유지하기 위한 서비스 중단 시간을 최소화하면서 데이터 백업을 수행할 수 있는 방식 중 하나이다.

본 논문에서는 디스크 기반 파일 시스템 스냅샷과 네트워크 기반 파일 시스템의 스냅샷에 대해서 연구한다. 디스크 기반 파일 시스템 스냅샷 연구에서는 Ext2, Ext3, XFS와 같이 스냅샷 기능을 제공하지 않는 리눅스 파일 시스템에서 사용 가능한 스냅샷 라이브러리인 PSnap을 제안하다. 또한 네트워크 기반 파일 시스템 스냅샷 연구에서는 ETRI에서 개발한 대용량 분산 파일 시스템인 GloryFS에 적용할 수 있는 GlorySnap를 제안한다.

Abstract

As the development of IT technologies, storages stored very sensitive data which should not be damaged, too. It increased the importance of data backup and makes the time need to backup data important issues. Snapshot is one of the backup technologies which needs short downtime to maintain consistency of data during backup data.

In this paper, we studied two kinds of snapshots, local file system based snapshot and network file system based snapshot. In the local file system based snapshot part, we propose the PSnap which is a snapshot library for non-snapshot file system as like Ext2, Ext3 and XFS. In another part, network file system based snapshot, we propose the GlorySnap which snapshot utilities for GloryFS is a distributed file system was made by ETRI.

**Keywords** : Snapshot, File system, Backup

## I. 서 론

IT 기술의 발전으로 인해 스토리지에 저장되는 데이터의 양과 중요성이 증가되면서 스토리지에 저장된 데이터의 손실은 정부, 기업 그리고 개인 모두에게 금전적으로 환산할 수 없는 막대한 피해를 유발하게 되었다. 따라서 이러한 경제적 손실을 예방하기 위해 데이터를 보호하기 위한 다양한 방법이 제안되었다. 그 중 스냅샷은 데이터 백업에 의한 스토리지 가용성 저하를 감소시킬 수 있는 방법 중 하나로 특정 시점의 스토리지에 저장되어 있는 데이터에 대한 스냅샷 이미지를 생성하고, 데이터 백업에 활용함으로서 스토리지 서비스가 정지되는 시간을 최소화한다.

이 논문에서는 디스크 기반 파일 시스템과 네트워크 기반 파일 시스템에서 활용 가능한 두 종류의 스냅샷, PSnap(Portable Snapshot Library)과 GlorySnap (Glory File System Snapshot)을 제안한다.

PSnap은 Ext2, Ext3, XFS 등의 스냅샷 기능을 제공

* 정회원, 세종대학교 컴퓨터공학과
(Department of Computer Engineering, Sejong University)

※ "This paper was supported by National Research Foundation of Korea Grant funded by the Korean Government(KRF-2008-314-D00336)"

접수일자: 2010년5월13일, 수정완료일: 2010년7월7일

하지 않는 리눅스 파일 시스템에서 사용 가능한 스냅샷 라이브러리로 특정 파일 시스템에 종속되지 않고 모든 파일 시스템에서 공통적으로 사용 가능한 스냅샷 라이브러리를 제공한다.

GlorySnap은 대용량 분산 파일 시스템인 GloryFS[2]에 스냅샷 기능을 추가하고 데이터 백업에 활용할 수 있게 함으로서 데이터 백업에 의한 시스템 가용성 저하를 감소시켰다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 관련 연구로 데이터 백업과 GloryFS에 대한 내용을 기술한다. Ⅲ장에서는 디스크 기반 파일 시스템 스냅샷인 PSnap에 대해서 설명하고, Ⅳ장에서는 네트워크 기반 파일 시스템 스냅샷인 GlorySnap에 대해서 설명한다. Ⅴ장에서는 PSnap과 GlorySnap의 성능을 측정하고 평가한다. 마지막으로 Ⅵ장에서는 결론을 내리고 향후 추가 연구가 필요한 부분에 대해서 논의한다.

## Ⅱ. 관련 연구

### 1. 데이터 백업

데이터 손실은 [표 1]과 같이 크게 물리적인 손실과 논리적인 손실로 구분될 수 있으며 각 정의는 다음과 같다. 물리적인 손실은 데이터가 저장된 스토리지가 화재, 충격, 노화 등으로 손상되어 더 이상 사용할 수 없는 경우를 의미한다. 반면에 논리적인 손실은 시스템 오류, 사용자 실수 등으로 스토리지에 저장된 데이터가 잘못된 값으로 변경되거나 삭제되었을 경우를 의미한다. 이러한 물리적인 손실과 논리적인 손실로부터 데이터를 보호하기 위해 데이터를 복사하는 단순한 방식으로 수행되던 기존의 데이터 백업 방식은 스토리지에 저장된 데이터의 양이 증가하면서 보다 긴 수행 시간을 필요로 했으며 그 만큼 서비스의 중단 시간도 길어져 스토리지의 가용성 저하의 원인이 되었다.

디스크 미러링[10]과 스냅샷[9]은 앞에서 언급한 스토리지의 가용성 저하 문제를 극복한 데이터 백업 방식으로 미러링 디스크와 스냅샷 이미지를 사용하여 데이터 백업에 의한 스토리지 가용성 저하를 감소시켰다[1].

표 1. 데이터 손실의 종류
Table 1. The reasons of data loss.

| | 발생 이유 | 영향 | 해결 방안 |
|---|---|---|---|
| 물리적 | 화재, 충격, 노화 | 스토리지 손상 | 백업, 미러링 |
| 논리적 | 시스템 오류, 사용자 실수 | 데이터 손상, 데이터 손실 | 백업, 스냅샷 |

### 2. SnapFS 파일 시스템[11]

SnapFS는 기존의 파일이 수정되는 경우에 Primary Inode를 복사하여 Indirect Inode를 생성하고 Primary Inode가 수정된 블록을 가리키도록 함으로써 Primary Inode는 사용자의 데이터 수정 작업이 반영되도록 하고 Indirect Inode는 기존의 데이터를 유지하도록 한다. SnapFS는 크게 2가지 구성요소, snap_current와 snap_clone로 구성되어 있다. Snap_current는 파일 시스템에서의 블록 단위로 수행되는 COW(Copy On Write)를 지원하기 위한 인터페이스이고 snap_clone은 스냅샷 이미지만을 위한 전용 파일 시스템을 관리한다. SnapFS는 VFS와 파일 시스템 사이에 위치하며 저널링 파일 시스템이 제공하는 원자적인 데이터의 기록에 의존하여 동작하고 추가적인 메타데이터를 필요로 하기 때문에 extended attributes를 가진 저널링 파일 시스템에서만 사용 가능하다.

### 3. Ext3COW 파일 시스템[12~13]

스냅샷 기능을 제공하는 파일 시스템으로 name space를 오염시키지 않고 적은 저장 공간과 부하만으로 수행되며 VFS 또는 커널의 변경 없이 독립적인 모듈로 구성된다는 장점을 가지고 있다.

### 4. Logical Volume Manager[14]

디스크 드라이브의 용량을 논리적으로 조절, 관리하여 디스크의 추가 및 타 시스템으로의 이동을 가능하게 하는 시스템이다. 디스크를 증설하여야 하는 경우, 데이터 백업, 재분할, 초기화, OS 재설치, 데이터 복구 등의 기능을 지원하며 디스크 공간을 복수의 논리 볼륨에 할당하여 손쉽게 디스크 재조정이 가능한 파일 시스템을 구현하는데 사용된다.

### 5. GloryFS

GloryFS[2]는 ETRI에서 개발한 대용량 분산 파일 시스템으로 저비용 스토리지 서버를 이용하여 대용량 파일 저장 공간을 제공하며 주요 설계 목표는 다음과 같다. 첫째, 스토리지는 저장 공간뿐만 아니라 대역폭도 함께 확장될 수 있어야 한다. 둘째, 클러스터를 구성하

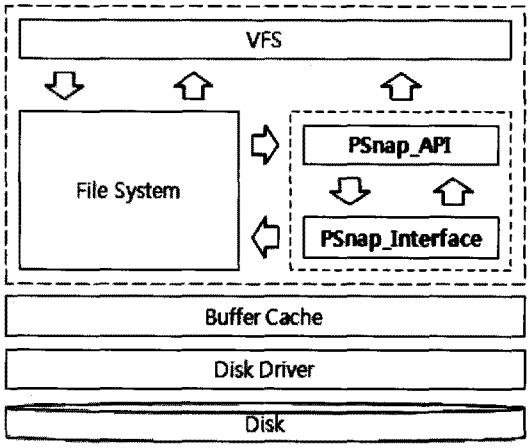


그림 1. PSnap 구조
Fig. 1. The design of Psnap.

는 노드들의 잦은 고장에도 스토리지에 저장된 데이터의 높은 가용성을 보장해야한다. 마지막으로 관리자가 손쉽게 설정하고, 감시하고, 관리할 수 있도록 중앙 집중적인 관리 포인트를 제공해야한다.

GloryFS는 3가지 구성 요소인 메타데이터 서버, 데이터 서버 그리고 클라이언트로 구성된다. 메타데이터 서버는 클러스터를 구성하는 모든 노드의 정보와 파일 시스템에 위치하는 모든 파일의 메타데이터를 로컬 디스크에 설치된 데이터베이스를 이용하여 관리한다.

데이터 서버는 파일의 실제 데이터가 저장되는 디스크를 포함한다. 클라이언트가 디스크 서버에 I/O 처리를 요청하면 디스크 서버는 해당하는 디스크에 I/O를 전달한다. 디스크 서버에 연결된 디스크들은 디스크 서버와 독립적으로 관리되기 때문에 특정 디스크 서버에서 다른 디스크 서버로 자유롭게 이동할 수 있다. 이러한 특징은 디스크 서버에 고장이 발생할 경우 디스크 서버 내의 디스크들을 다른 디스크 서버로 이동해서 서비스를 재개할 수 있게 하고, 디스크 서버 내의 특정 디스크에서 고장이 발생할 경우 고장이 발생한 디스크를 제외한 다른 디스크들을 지속해서 서비스 할 수 있게 한다.

GloryFS의 클라이언트는 FUSE(Filesystem in Userspace)[4~5]를 사용해서 구현된다. 따라서 POSIX 인터페이스를 대부분 지원하고, 운영체제가 FUSE를 지원한다면 운영체제에 비종속적으로 사용할 수 있다.

## Ⅲ. PSnap

파일 시스템 기반의 스냅샷은 파일 시스템의 자료 구조와 데이터 블록 처리 알고리즘에 밀접하게 연관된다. 따라서 서로 다른 파일 시스템의 스냅샷은 서로 다르게 구현되는 것이 일반적이다. 이러한 특성은 스냅샷의 이식성을 저하시키는 주요 원인이다. 하지만 PSnap[3]은 특정 파일 시스템에 종속되지 않고 모든 파일 시스템에서 공통적으로 사용 가능한 스냅샷 라이브러리를 제공하여 이러한 한계점을 극복하고자 했다.

PSnap은 [그림 1]과 같이 PSnap_API와 PSnap_Interface로 구성된다. PSnap_API는 파일 시스템에 비종속적으로 사용되며 스냅샷 정보 관리, 스냅샷 처리 등을 담당한다. PSnap_Interface는 파일 시스템의 자료 구조 접근, 데이터 블록 조작 등과 같이 파일 시스템에 종속적인 처리를 담당한다. PSnap_API는 스냅샷 처리 과정 중에 파일 시스템에 종속적인 처리가 필요할 경우 직접 처리하지 않고 PSnap_Interface를 사용함으로서 파일 시스템 종속성을 제거한다.

### 1. PSnap_Countmap

PSnap_API는 파일 시스템에 비종속적으로 스냅샷 정보를 관리하기 위해서 PSnap_Countmap을 사용한다. PSnap_Countmap은 [그림 2]와 같이 파일 시스템의 모든 데이터 블록과 일대일로 맵핑되는 엔트리들로 구성된다. 각각의 엔트리들은 0부터 255까지의 값을 가질 수 있으며, 맵핑된 데이터 블록을 공유하는 파일의 개수를 의미한다. 파일 시스템이 PSnap 파일 시스템으로 변환되면 파일 시스템의 루트 디렉토리에 정규 파일 형태로 PSnap_Countmap이 생성된다. PSnap_API는 PSnap_ Countmap을 조작할 때 리눅스 가상 파일 시스템이 제공하는 함수를 사용함으로서 파일 시스템 종속성을 제거한다.

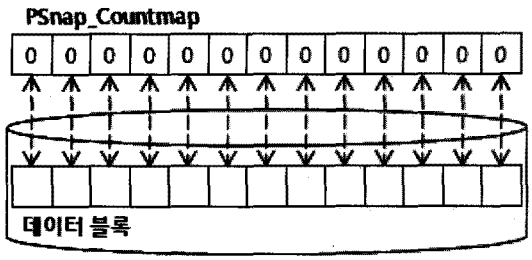


그림 2. PSnap_Countmap 구조.
Fig. 2. The structure of PSnap_Countmap.

### 2. 스냅샷 처리

PSnap 파일 시스템에서 파일에 대한 스냅샷 생성 요

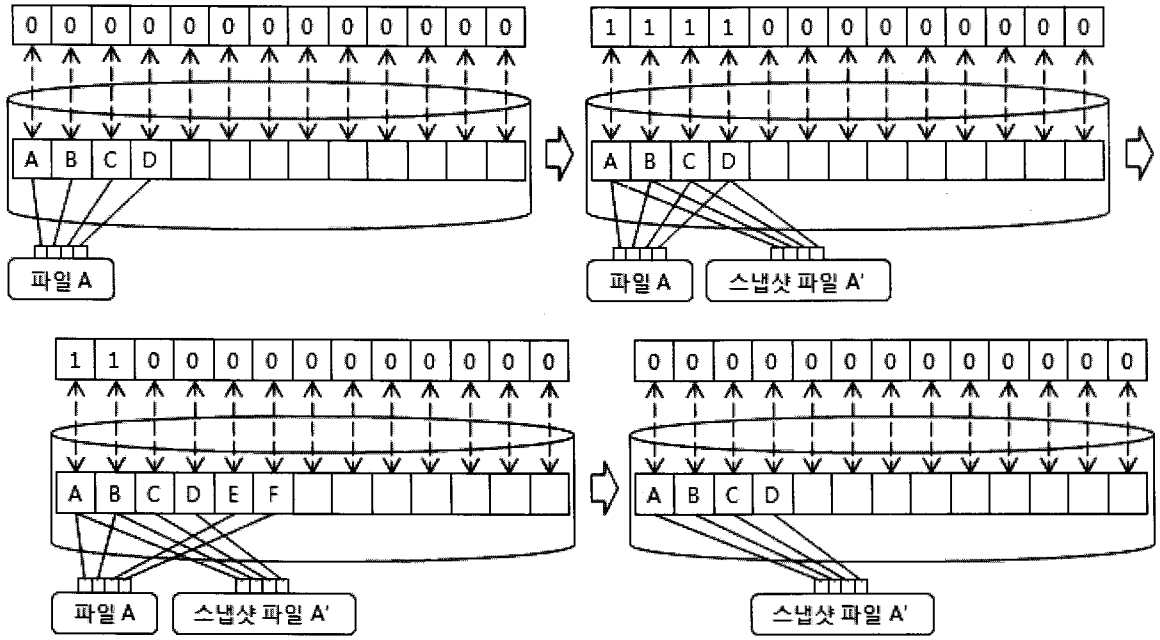


그림 3. 스냅샷 처리
Fig. 3. The procedure of snapshot operations.

청이 발생하면 원본 파일과 동일한 디렉토리에 '원본파일명. 생성 시간'의 이름을 가지는 스냅샷 파일이 생성된다. 생성된 스냅샷 파일은 원본 파일의 메타데이터를 상속받고 데이터 블록을 공유한다.

[그림 3]은 PSnap 파일 시스템에서 새로운 스냅샷 파일 생성, 스냅샷 파일이 생성된 원본 파일 수정, 스냅샷 파일이 생성된 원본 파일 삭제의 처리 과정을 보인다. PSnap 파일 시스템에 4개의 데이터 블록을 가지는 파일 A를 생성하고, 파일 A에 대한 스냅샷 생성을 요청하면 스냅샷 파일 A'가 생성된다. 스냅샷 파일 A'는 파일 A와 데이터 블록을 공유하기 때문에 공유되는 데이터 블록에 맵핑되는 PSnap_Countmap 엔트리 값들이 1씩 증가한다. 파일 A에 데이터 수정 요청이 발생하면 PSnap_API는 데이터 블록을 수정하기 전에 해당 데이터 블록이 스냅샷 파일에 의해서 공유중인지 확인한다. 데이터를 수정하려는 3번과 4번 데이터 블록에 맵핑된 PSnap_ Countmap의 엔트리 값이 0보다 크기 때문에 수정하려는 데이터 블록이 스냅샷 파일과 공유중임을 알 수 있다. 따라서 PSnap_API는 파일 A에 새로운 데이터 블록을 할당해서 수정된 데이터를 저장하고 기존의 데이터 블록에 맵핑됐었던 PSnap_Countmap 엔트리 값을 1씩 감소시킨다. 파일 A에 대한 삭제 요청이 발생하면 PSnap_API는 데이터 블록을 해제하기 전에 해당 데이터 블록이 스냅샷 파일에 의해서 공유중인지 확인한다. 파일 A의 1번과 2번 데이터 블록은 스냅샷 파일과 공유중이기 때문에 데이터 블록을 해제하지 않고 맵핑된 PSnap_Countmap 엔트리의 값만 1씩 감소시킨다. 3번과 4번 데이터 블록을 스냅샷 파일과 공유되고 있지 않기 때문에 일반적인 파일 시스템의 처리 과정에 따라서 해당 데이터 블록을 해제한다. 그리고 파일 A의 메타데이터를 삭제함으로서 파일 삭제 작업을 완료한다.

## Ⅳ. GlorySnap

ETRI에서 개발한 대용량 분산 파일 시스템인 GloryFS[2]는 파일 시스템을 구성하는 구성 요소의 고장 등의 이유로 발생하는 데이터의 물리적 손실로부터 데이터를 보호하기 위해서 데이터 리플리케이션을 사용한다. 그러나 사용자의 실수와 같은 이유로 발생하는 데이터의 논리적인 손실로부터 데이터를 보호할 수 있는 적절한 방법을 제공하지 않기 때문에 데이터를 완벽하게 보호하기 위해서는 주기적으로 데이터 백업을 수행해야만 한다.

GlorySnap은 GloryFS에 스냅샷 기능을 추가하고 데이터 백업에 활용할 수 있게 함으로서 데이터 백업에 의한 시스템 가용성 저하를 감소시킨다. 또한 생성된 스냅샷 파일을 유지할 경우 논리적인 손실 발생 시 스냅샷 파일을 사용해서 과거의 데이터로 신속하게 복구

할 수 있다.

### 1. 스냅샷 정보 관리

GloryFS는 파일 시스템의 메타데이터를 관리하기 위해서 메타데이터 서버의 로컬 디스크에 설치된 데이터베이스에 [표 2]와 같은 테이블을 생성하여 사용한다.

GloryFS는 파일의 데이터를 저장하기 위해서 하나의 파일을 64MB 단위의 청크로 나누며 각각의 청크는 [표 2]에 나타난 것과 같이 offset과 version을 값을 가진다. Offset 값이 0번인 청크는 파일의 시작부터 64Mbytes까지의 데이터를 저장하고, offset이 1인 청크는 64Mbytes부터 28Mbytes에 해당하는 데이터를 저장한다. Version은 각각의 청크마다 독립적으로 유지되며 청크가 수정될 때마다 증가하여 데이터 리플리케이션으로 인해 다수의 디스크에 존재하는 동일 데이터의 일관성을 유지할 수 있도록 한다.

스냅샷 테이블의 snap_inoid는 레코드를 소유한 스냅샷 파일의 아이노드 번호를 저장하고, snap_orig_inoid는 스냅샷 파일을 생성한 원본 파일의 아이노드 번호를 저장한다. Snap_offset은 스냅샷 파일의 몇 번째 청크를 나타내는 정보인지를 의미하며, snap_ver는 스냅샷 생성 시점에 해당 청크의 version을 저장한다. 새로운 스냅샷 파일을 생성하면 GlorySnap은 원본 파일이 가지고 있는 모든 유효한 청크들에 대한 레코드를 스냅샷 테이블에 생성함으로서 원본 파일과 스냅샷 파일이 청크를 공유할 수 있게 했다.

표 2. 스냅샷 정보 테이블
Table 2. The information table of snapshot.

```
create table snapshot
(
        snap_inoid              bigint unsigned,
        snap_orig_inoid         bigint unsigned,
        snap_offset             int unsigned,
        snap_ver                int unsigned
);
```

### 2. 스냅샷 처리

GloryFS에서 클라이언트가 파일에 접근하기 위해서 메타데이터 서버에 파일의 메타데이터를 요청하면 메타데이터 서버는 아이노드 테이블에서 파일이 포함하는 청크들의 개수를 얻고 청크 테이블에서 청크가 저장된 위치를 찾아서 클라이언트에 전송한다. GlorySnap에서는 클라이언트가 스냅샷 파일에 접근하기 위한 요청을 처리하는 부분이 추가되어야 한다. GlorySnap은 클라이언트가 스냅샷 파일의 메타데이터를 요청하면 아이노드 테이블에서 파일의 청크 개수를 얻고 스냅샷 테이블에서 스냅샷 생성 당시 원본 파일의 청크 정보를 얻은 다음 청크 테이블에서 청크 가 저장된 위치를 찾아서 클라이언트에 전송한다.

스냅샷 파일이 생성된 원본 파일에 데이터 수정이나 삭제가 발생하는 경우, GlorySnap은 원본 파일의 청크가 스냅샷 파일과 공유중이기 때문에 수정되거나 삭제되는 것을 방지해야한다. GloryFS는 리플리케이션을 사용하기 때문에 동일한 데이터에 대해서 두 개 이상의 청크를 유지한다. 기존의 GloryFS는 데이터 수정 발생 시에 'primary chunk'를 수정한 후 'secondary chunk'를 삭제하고 'primary chunk'를 복사해서 새로운 'secondary chunk'를 만든다. GlorySnap은 데이터 수정 과정에서 삭제되는 'secondary chunk'를 삭제하지 않고 스냅샷 파일 전용 청크로 유지함으로서 원본 파일이 수정되더라도 스냅샷 파일로 스냅샷 파일이 생성될 당시에 원본 파일이 가지고 있던 데이터에 접근할 수 있다. 스냅샷 파일이 생성된 원본 파일의 삭제가 발생하는 경우에도 GlorySnap은 원본 파일의 청크를 삭제하지 않고 스냅샷 파일 전용 청크로 유지함으로서 스냅샷 파일로 삭제된 원본 파일의 데이터에 접근하는 것이 가능한다.

## V. 성능 측정

스냅샷[9]은 데이터 백업에 의한 스토리지 가용성 저하를 감소시키기 위한 목적으로 사용된다. 따라서 스냅샷 생성 시간이 데이터 백업 수행 시간보다 신속하게 완료될 수 있어야 한다. 성능 측정에서는 가장 단순한 형태의 데이터 백업 방법인 파일 복사와 연구 결과인 Psnap에 의한 스냅샷 생성 속도를 비교한다. 그리고

파일 시스템에 스냅샷을 추가함으로서 발생할 수 있는 성능 저하를 측정하기 위해서 간단한 I/O 테스트를 수행하였다.

### 1. PSnap

PSnap의 성능 측정은 표 3과 같은 환경에서 수행되었다. [그림 3]의 "cp vs PSnap"은 HDD1에 설치된 파일 시스템과 동일한 파일 시스템이 설치된 HDD2로 'cp'를 이용하여 데이터를 복사하는 시간과 HDD1에 설

표 3. PSnap 테스트 환경
Table 3. The test environments.

| | |
|---|---|
| OS | CentOS 5.3 |
| CPU | Intel(R) Core(TM)2 CPU 6300 @ 1.86GHz |
| RAM | Samsung DDR2 1GB PC2-6400U CL6 |
| HDD1 | Seagate 750GB Barracuda 7200.11<br>ST3750330AS (SATA2/7200/32M) |
| HDD2 | Seagate 250GB Barracuda 7200.10<br>ST3250620AS (SATA2/7200/16M) |

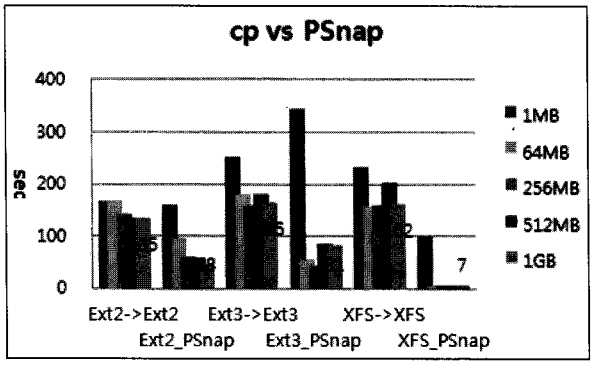


그림 4. 백업과 스냅샷 속도 비교(PSnap)
Fig. 4. The comparison of the performance of oopy and PSnap.

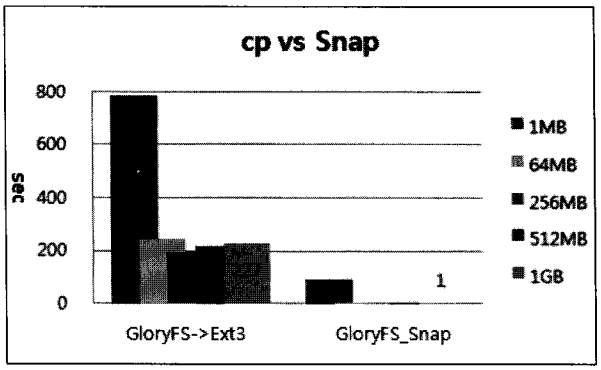


그림 5. 백업과 스냅샷 속도 비교(GlorySnap)
Fig. 5. The comparison of the performance of copy and GlorySnap.

치된 파일 시스템에 PSnap을 사용하여 스냅샷 기능을 추가하고 스냅샷을 생성하는데 걸리는 시간을 측정하여 비교하였다. 성능 측정에는 1MB부터 1GB까지 다양한 크기의 파일이 사용되었으며, 파일 개수는 파일 크기와 파일 개수를 곱한 값이 10GB가 되도록 설정하였다.

Ext2[6]는 파일 크기가 작을 경우 'cp'와 PSnap이 유사한 성능을 보였지만 파일 크기가 증가하면서 PSnap이 'cp'보다 좋은 성능을 보였다. Ext3는 파일 크기가 작을 경우 'cp'가 PSnap보다 좋은 성능을 보이지만, 파일 크기가 커지면서 PSnap이 더 좋은 성능을 보인다. XFS[8]는 파일 크기와 상관없이 항상 PSnap이 좋은 성능을 보였다. 1GB 파일의 경우 10개를 'cp'를 사용해서 백업하려면 162초가 필요하지만 PSnap을 사용하여 스냅샷을 생성할 경우 7초 만에 작업을 완료할 수 있었다.

[그림 4]의 "random path test"는 HDD1에 설치된 파일 시스템과 HDD1에 설치된 파일 시스템에 PSnap을 사용하여 스냅샷 기능을 추가한 파일 시스템에서 일반적인 I/O 수행 성능을 보인다. 각각의 파일 시스템에서 각각의 I/O에 대해 4KB 크기로 2,621,440번의 I/O를 수행하였고, I/O 수행에 사용된 파일의 경로명은 랜덤하게 설정하였다. 그림 4에서 볼 수 있듯이 Ext2, Ext3, XFS 모두 PSnap에 의한 오버헤드가 I/O 수행에 크게 영향을 미치지 않음을 보였다.

## 2. GlorySnap

GlorySnap의 성능 측정은 [표 4], [표 5], [표 6]과 같은 노드들로 구성된 환경에서 수행되었다. 320GB 크기의 디스크가 하나씩 연결된 4대의 디스크 서버가 1.2TB의 저장 공간을 구성하였고, 메타데이터 서버와 클라이언트는 각각 1대씩 사용하였다.

[그림 5]의 "cp vs PSnap"은 GloryFS에 저장된 파일을 클라이언트에 연결된 HDD의 Ext3 파일 시스템에

표 4. PSnap 테스트 환경(메타데이터 서버)
Table 4. The test environments of metadata server.

| | |
|---|---|
| OS | CentOS 5.3 |
| CPU | AMD Phenom(tm) 8650<br>Triple-Core Processor 2.31GHz |
| RAM | Samsung DDR2 2GB PC2-6400 x 2 |
| HDD | Seagate 750GB Barracuda 7200.11<br>ST3750330AS (SATA2/7200/32M) |

표 5. PSnap 테스트 환경(디스크 서버)
Table 5. The test environments of disk server.

| | |
|---|---|
| OS | CentOS 5.3 |
| CPU | AMD Athlon(tm)<br>Dual Core Processor 4050e 2.1GHz |
| RAM | Samsung DDR2 1GB PC2-6400U CL6 |
| HDD | Seagate 320GB Barracuda 7200.11<br>ST3320613AS (SATA2/7200/16M) |

표 6. PSnap 테스트 환경(클라이언트)
Table 6. The test environments of clients.

| | |
|---|---|
| OS | CentOS 5.3 |
| CPU | Intel(R) Core(TM)2 CPU 6300 @ 1.86GHz |
| RAM | Samsung DDR2 1GB PC2-6400U CL6 |
| HDD | Seagate 250GB Barracuda 7200.10<br>ST3250620AS (SATA2/7200/16M) |

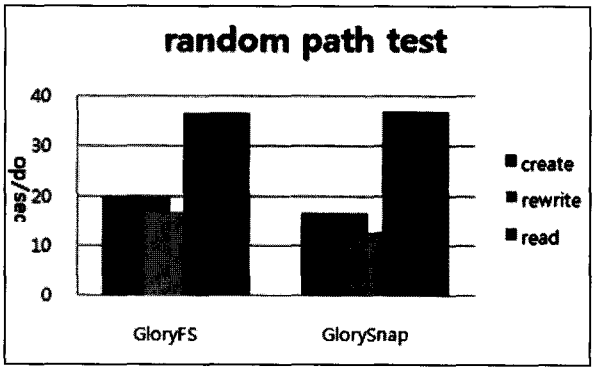


그림 6. 파일 IO 성능 측정(GlorySnap)
Fig. 6. The file I/O performance results of GlorySnap.

cp를 이용하여 복사하는 시간과 GlorySnap을 사용하여 스냅샷을 생성하는데 걸리는 시간을 측정하여 비교하였다. 성능 측정은 1MB부터 1GB까지 다양한 크기의 파일이 사용되었으며, 파일 개수는 파일 크기와 파일 개수를 곱한 값이 10GB가 되도록 설정하였다. 테스트 결과 모든 크기의 파일에서 GlorySnap이 더 좋은 성능을 보였으며 1MB 파일의 경우 8배, 1GB 파일에서는 200배 이상의 성능 차이를 보였다.

[그림 6]의 "random path test"는 GloryFS에 GlorySnap을 적용하기 전과 적용한 후에 일반적인 IO 수행 성능을 측정한 결과를 보인다. 각각의 IO에 대해서 4KB 크기로 2,621,440번의 IO를 수행하였고, IO 수행에 사용된 파일의 경로명은 랜덤하게 설정하였다. 테스트 수행 결과 create와 rewrite에서는 GlorySanp 고유의 데이터베이스 처리로 인해서 약간의 성능 저하를 보였지만 read에서는 성능 변화를 보이지 않았음을 알 수 있다.

## VI. 결 론

스토리지에 저장되는 데이터의 중요성이 증가하면서 저장된 데이터를 안전하게 보호할 수 있는 방법이 요구되고 있다. 따라서 데이터 백업은 저장된 데이터를 보호하기 위한 필수적인 선택이다. 또한 저장된 데이터의 양이 증가하면서 데이터 백업 수행 시간이 스토리지 가용성에 미치는 영향이 점차 증가하고 있다.

본 논문에서 소개한 디스크 기반 파일 시스템 스냅샷인 PSnap과 네트워크 기반 파일 시스템 스냅샷인 GlorySnap은 스냅샷 이미지를 생성하고 데이터 백업에 사용할 수 있게 함으로서 데이터 백업에 의한 스토리지 가용성 저하를 효율적으로 감소시킨다. PSnap을 사용하여 데이터 백업을 수행할 경우 1MB 이상의 파일에 대해서 비슷하거나 2배 이상의 효과를 보였고, GlorySnap은 모든 크기의 파일에 대해서 3배 이상의 효과를 보였다.

앞으로 PSnap에서 작은 크기 파일에 대한 처리 효율성을 증가시키고, 스냅샷 처리 과정에서 발생할 수 있는 데이터 블록 단편화를 최소화 할 수 있는 방법에 대해서 연구할 계획이다. 또한 GlorySnap에서 스냅샷 처리를 위한 데이터베이스 접근을 최소화함으로서 성능을 향상시킬 계획이다.

## 참 고 문 헌

[1] W. Curtis Preston, *"Using SANs and NAS"*, OReilly, 2002
[2] Keun-Tae Park, Hong-Yeon Kim, Young-Chul Kim, Sang-Min Lee, Young-Kyun Kim, Myung-Joon Kim, *"Lake: Towards highly manageable clustre storage for extremely scalable services"*, ICCSA, pp. 122~131, 2008
[3] 김선태, 석진선, 노재춘, *"이기종 파일시스템을 위한 스냅샷 라이브러리"*, 한국정보과학회 2009 한국컴퓨터종합학술대회 논문집 제36권 제1호(A), pp. 314~315, 2009
[4] "FUSE Documentation", http://www.prism.uvsq.fr
[5] "Introduction to FUSE and Working of FUSE", http://fuse.sourceforge.net/
[6] R´emy Card, Theodore Ts'o, Stephen Tweedie." Design and Implementation of the Second Extended Filesystem." : Proceedings of the First Dutch International Symposium on Linux. State University of Groningen, 1995.
[7] S. C. Tweedie. Journaling the Linux ext2fs File System. In The FourthAnnual Linux Expo, Durham, North Carolina, May 1998.
[8] Sweeney, A., Doucette, D., Hu, W., Anderson, C.,Nishimoto, M., and Peck, G. "Scalability in the XFS file system." In Proceedings of the USENIX 1996 Technical Conference, pages 1.14, San Diego, CA, USA, 1996.
[9] A. Azagury, M. E. Factor, and J. Satran. Point-in-time copy: Yesterday, today and tomorrow. In Proceedings of the Tenth Goddard Conference on Mass Storage Systems and Technologies, pages 259 - 70, April 2002.
[10]D. Bitton and J. Gray, "isk Shadowing," Proceedings of the 14th Conference on Very Large Data Bases, 1988, pp. 331-338.

[11] SnapFS,
Http://SOURCEFROG.NET/PROJECTS/SNAPFS
[12] Z. N. J. Peterson, R. Burns. \Ext3cow: The design, implementation, and anal-ysis of metadata for a timeshifting -le system." : Technical report, HopkinsStorage Systems Lab., Department of Computer Science, The Johns Hopkins University, 2003.
[13] Z. Peterson, R. Burns. \Ext3cow: A Time-Shifting File System for Regulatory Compliance." : ACM Transactions on Storage, 1(2):190-212, 2005.
[14] SuSE Inc., \The Logical Volume Manager (LVM)" : Technical report, 2002.

## 저 자 소 개

**석 진 선**(정회원)
2005년 세종대학교 소프트웨어 공학과 학사 졸업
2007년 세종대학교 컴퓨터공학과 석사 졸업
2010년 세종대학교 컴퓨터공학과 박사 과정
<주관심 분야 : 파일시스템, 저장장치 시스템>

**노 재 춘**(정회원)-교신저자
1985년 이화여자대학교 전산과 학사 졸업
1993년 Western Illinois Univ. 전산과 석사 졸업
1999년 Syracuse Univ. 전산과 박사 졸업
<주관심 분야: 파일시스템, 저장장치 시스템, 모바일네트워크>