한국항행학회 논문지 제17권 제1호 2013년 2월 (JKONI 17(1): 39-46, Feb. 2013)

리눅스에서 Iozone 도구를 이용한 저널링 파일 시스템 성능 평가 : read, write 중심으로

Performance Evaluation on Journaling File Systems using Iozone Tool in the Linux : Focus on read, write

박홍진*

Hong-Jin Park*

요 약

비정상적인 시스템 종료로 인해 파일 시스템이 손상되었을 경우 시스템은 fsch을 이용하여 일관성 검사를 수행하며 이는 오랜 시간이 소요된다. 특히 대용량의 파일 시스템인 경우에는 상당한 시간이 걸린다. 저널링 기법을 이용한 저널링 파일 시스템은 메타 데이터를 이용하기 때문에 복구 시간을 상당히 단축시킬 수 있으며, 복구시 복구 확률도 높일 수 있다. 본 논문의 목적은 현재 리눅스에서 사용되고 있는 저널링 파일 시스템을 커널 기반의 벤치 마킹 도구인 Iozone을 사용하여 파일의 read, write 중심으로 성능을 비교평가하는 것이다. 본 논문에서는 현재 리눅스의 기본 파일 시스템인 Ext4 파일 시스템이 파일 읽기 성능이 경우 XFS 파일 시스템보다 1.28배, 파일의 쓰기 성능의 경우 Ext3 보다 1.22배 빠르게 전송되었다.

Abstract

If a file system is damaged because of the unusual system close, the system performs the consistency test using fsch and it takes long time. Especially, if it is a big file system, it will take a lot of time. The journaling file system that uses journaling technique, can reduce the restoring time because it uses meta data and it may increase the chance of restoration when restoring.

The goal of this paper compared performance evaluation journaling file systems focused on the reading and writing using Iozone tool which is the kernel based benchmarking tool in linux operating system. In this paper, Ex4 which is the current basic Linux file system. is 1.28x faster than XFS file system in terms of file read performance and 1.22x faster than Ext3 file system in terms of file write performance.

Key words : Journaling file system, Iozone, Kernel-based benckmarking

I. 서 론

리눅스에서 비정상적으로 시스템이 종료되면, 재부팅 시 리눅스 운영체제가 이를 감지하여 fsck(file system check)을 통해 파일 시스템 일관성 검사를 수행한다. 파일 시스템이 손상되지 않았더라도 운영체제는 부팅 시 파일 시스템의 일관성이 올바르게 되어 있는 지 확인하기 위해 파일 시스템의 마운트 시점에

* 상지대학교 컴퓨터정보공학부(School of Computer, Information and Communications, Sangji-Ji University)
· 제1저자 (First Author) : 박홍진(Hong-JIn Park, Tel : +82-33-730-0489, email : hjpark1@sangji.ac.kr)
· 접수일자 : 2012년 12월 17일 · 심사(수정)일자 : 2012년 11월 15일 (수정일자 : 2013년 2월 14일) · 게재일자 : 2013년 2월 28일
http://dx.doi.org/10.12673/jkoni.2013.17.01.039

서 자동으로 fsck를 통해 파일 시스템의 일관성을 점검한다. 파일 시스템의 일관성 검사 중에 파일 시스템의 이상이 발생되면 시스템 관리자는 fsck를 통해 파일 시스템을 복구해야 한다. 그러나 fsck를 이용한 파일 시스템 일관성 검사는 상당히 오래 시간이 걸리며 특히 저널링 기법이 있는 파일 시스템과 비교하였을 때 비정상 종료 시 복구 시간도 많은 시간이 소요된다. 저널링 파일 시스템은 비정상 종료나 전원 문제가 발생했을 때 파일 시스템의 손상을 방지면서 안정성은 높여주는 파일 시스템 기법이다. 그림 1은 저널을 이용한 저널링 파일 시스템을 나타내고 있다.

그림 1. 저널링 파일 시스템
Fig. 1. Journaling file system

저널링 파일 시스템은 저널(journal)을 관리하는 방법으로 파일 시스템 손상을 관리한다. 저널링 기능이란 데이터베이스의 로깅(logging) 기법을 이용한 것으로 파일을 쓰는 작업을 수행할 때 파일 시스템의 특정 영역인 저널이라고 불리는 공간에 파일과 관련된 메타 데이터(meta data)를 기록한 뒤 정상적으로 쓰기가 완료되었을 경우 실제 영역에 기록하는 방법을 의미한다. 저널링 파일 시스템의 구조는 파일 시스템의 구조는 저널 영역과 일반적인 파일 시스템 영역으로 구분되어 진다. 저널 영역은 파일 시스템의 파일 시스템의 존재하는 파일 데이터를 관리하는 구조체로 파일의 생성과 삭제, 디렉토리의 생성과 삭제 파일 크기등의 정보가 있다. 시스템은 주기적으로 저널의 메타 데이터를 파일 시스템에 저장된다. 저널에 저장되어 있는 메타 데이터는 원형 버퍼 구조 형태로 되어 있어서 시간이 지나면 포화 상태가 된다. 저널 영역이 포화가 되면 가장 오래된 레코드 순으로 삭제하는 동작을 수행하게 된다.

비정상 종료가 발생되면 저널은 저장되지 않은 데이터를 복구하기 위해 저널에 저장되어 있는 메타 데이터를 참조하여 파일 시스템을 재작성하거나 복구하게 된다. 파일 시스템의 용량이 대용량일수록 기존의 시스템은 오랜 복구 시간이 걸리는데 비해 저널링 파일 시스템은 저널 공간의 메타 데이터를 이용하기 때문에 파일 시스템 복구 확률도 높아지며 복구 시간도 상당히 단축되어 진다. 이는 서버 운영에 있어 매우 중요하다.

저널링 파일 시스템에서 저널에 기록하는 방법에는 쓰기저장(writeback) 방법, 순서(ordered) 방법, 자료(data) 방법이 있다[1]. 쓰기저장 방법은 메타자료만 저널에 기록되며, 자료는 해당 디스크에 직접 저장된다. 이 경우에는 메타자료를 저장하고 나서 자료가 저장되기 전에 시스템이 비정상 종료가 발생 시 자료 손상이 발생된다. 순서 방법은 모든 자료를 기록한 다음 메타자료를 저널에 기록한다. 이 방식은 복구 이후에 자료와 파일 시스템의 일관성을 보장한다. 자료 방법에서는 메타자료와 자료가 모두 저널에 기록하는 방식을 말한다. 모든 메타데이타와 자료가 먼저 저널에 기록된 후에 나중에 디스크에 저장되기 때문에 메타 데이터와 데이터가 두 번에 걸쳐서 저장되며 이는 파일 시스템 일관성 측면에서 가장 강력한 기법이지만 성능 저하의 문제점이 존재한다.

본 논문은 다양한 저널링 파일 시스템의 성능을 비교 평가한다. 최초의 저널링 파일 시스템인 JFS 파일 시스템의 성능을 개선한 JFS2 파일 시스템을 비롯하여, Ext3 파일 시스템, 현재 리눅스의 기본 파일 시스템인 Ext4 파일 시스템, ReiserFS 파일 시스템, SGI사의 Irix 운영체제의 기본 파일 시스템인 XFS 파일 시스템을 비교 평가한다. 성능평가를 위해 사용된 도구는 커널 기반의 벤치 마킹도구인 Iozone을 이용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 기반 연

구로 저널링 파일 시스템의 벤치마킹 방법과 본 연구에서 성능 평가하게 될 저널링 파일 시스템에 대해서 살펴본다. 3장에서는 Iozoen을 이용한 저널링 파일 시스템의 성능을 평가하며, 4장에서 결론을 맺는다.

II. 기반 연구

본 장에서는 리눅스에서 제공되는 다양한 저널링 파일 시스템을 살펴보고, 이를 평가할 수 있는 벤치마킹 방법을 살펴본다[2].

2-1 저널링 파일 시스템

① JFS2(Journaling File System 2) 파일 시스템

JFS 파일 시스템은 IBM에서 개발된 최초의 저널링 파일 시스템이다[3]. 1990년도 처음 출시된 JFS는 현재 리눅스에서 지원되는 버전은 JFS의 성능을 개선한 JFS2 파일 시스템이며, 리눅스에서 사용되고 있는 JFS는 JFS2을 의미한다. JFS2는 JFS의 확장성을 향상시켰으며 다중 프로세서 아키텍쳐를 지원한다. JFS2 파일 시스템은 순서 기반의 저널링 기법을 제공한다. 또한 JFS2는 성능향상을 위한 익스텐트(extent) 기반 파일 할당을 제공하고 있다. 익스텐트 기반 파일 할당은 파일 할당 시에 연속적인 블록 집합을 할당하는 것을 말한다. 파일 할당 시 연속적인 블록 할당 방식은 디스크에 파일이 연속적으로 배치되어 있으므로 읽기와 쓰기의 성능이 좋다. 또한, 연속적으로 파일 할당되어 있기 때문에 이를 관리하는 메타 데이터의 저장 공간도 절약된다. 파일 시스템이 마운트 되어 사용 중에 있더라고 파일 시스템 크기를 변경하거나 단편화를 제거할 수 있다. B+ 트리를 이용하여 디렉토리 검색을 수행하는 JFS2 파일 시스템은 IBM 운영체제인 OS/2, AIX와 Linux 운영체제에서 사용되고 있다. 이후 본 논문에서는 JFS 파일 시스템은 JFS2 파일 시스템을 의미한다.

② Ext3(Third Extended File System) 파일 시스템

2001년 11월 개발된 Ext3 파일 시스템은 Ext2 파일 시스템의 저널링 기능을 추가한 저널링 파일 시스템이다[4]. Ext3는 Ext2의 부분에서 대부분 호환이 가능하도록 설계되었기 때문에 Ext2 파일 시스템에서 자료 수정 및 손실 없이 Ext3 파일 시스템으로 변경할 수 있다. Ext3의 저널링 방법은 쓰기저장 방법, 순서 방법, 자료 방법 모두 지원하며 순서 방법이 기본으로 설정되어 있다.

③ Ext4(Fourth Extended File System) 파일 시스템

Ext4 파일 시스템은 기존 Ext3과 비교하여 확장성과 안정성이 크게 향상된 저널링 파일 시스템이다[5]. Ext4 파일 시스템은 1TB 디스크를 최대 백 만개까지 사용할 수 있는 대규모의 파일 시스템으로 확장할 수 있다. Ext4 파일 시스템은 B 트리 변종인 H 트리를 이용하여 디렉토리를 유지하기 때문에 하위 디렉터리의 수가 32,000개로 제한되어 있는 Ext3 파일 시스템보다 하위 디렉토리 수를 64,000개로 크게 향상시켰다. 또한, Ext4 파일 시스템은 이전 버전들은 파일과 디렉토리 정보를 포함하고 있는 inode의 크기를 128바이트에서 256바이트로 확장시켰다. 리눅스 커널 2.6.19부터 포함되어 있는 Ext4 파일 시스템은 2008년부터 현재까지 페도라, 데비안, 우분투, 민트 등 많은 리눅스 패키지의 기본 파일 시스템으로 이용되고 있다.

④ ResierFS 파일 시스템

Hans Resier에 의해 개발된 ResierFS 파일 시스템은 메타 데이터를 이용한 저널링 파일 시스템중 하나이며, 크기가 작은 파일에 좋은 성능을 나타내고 있다[6]. ResierFS은 B+ 구조를 사용하고 있다. 데이터는 트리에 단말 노드(terminal node)에 위치하고 있으며, 내부 노드(internal node)에는 메타 데이터가 저장되어 있다. 또한 기존 ext2나 ext3 파일 시스템과 다르게 저널이 파일되어 있지 않고 파일 시스템 시작에서부터 연속적으로 저널 공간이 있다. 이는 빠른 디렉토리 검색이나 삭제 수행이 가능하다.

⑤ XFS(eXtended File System) 파일 시스템

SGI사 Irix 운영체제의 기본적인 파일 시스템인 XFS은 64비트를 지원하고 확장성 있는 자료구조를 지원하는 저널링 파일 시스템이다[7]. 2001년부터 리눅스에 이식되어 사용되었기 때문에 안정성이 높은 파일 시스템이다. XFS 파일 시스템은 블록의 크기를 512B부터 64KB까지의 변경이 되는 가변 블록 크기를 지원하며, 익스텐트 기반의 파일 할당 기법을 사

용한다. 디스크에 블록을 쓰기 전에 디스크 블록 할당을 연기시킬 수 있는 지연 할당 기법을 사용한다. 지연 할당 기법을 사용하면 파일의 전체 블록 개수를 미리 파악하기 때문에 연속적인 디스크 블록 할당의 가능성이 향상된다. XFS 파일 시스템은 디렉토리와 파일 할당에 B+를 사용함으로써 성능을 향상시켰으며 쓰기저장 저널링 방법을 사용한다.

2-2 저널링 파일 시스템의 벤치마킹 방법

저널링 파일 시스템의 성능 평가는 다양한 벤치마킹 방법이 있다[8][9]. 저널링 파일 시스템 벤치마킹의 접근 방법은 다음과 같이 분류된다.

① 작업 부하 기반의 벤치마킹

이 방법은 벤치마킹의 현실성을 강조하고 실제 응용 프로그램으로부터 미리 정의된 작업부하에 의존하는 것이다. TPC(Transaction Processing Performance), SPEC(Standard Performance Evaluation Corporation) 벤치마킹이 이 분류에 속한다. 작업 부하 기반의 벤치마킹은 실제 성능 평가에 가장 적합한 평가를 제공하고 있으나, 이식성 문제가 자주 발생하고, 공정한 플랫폼에서의 성능 평가 비교는 어렵다. 더 나아가서 작업 부하 기반의 벤치마킹은 시스템의 특이한 모습에 관한 정보도 제공하지 않고 있다.

② 커널 기반의 벤치마킹

이 방법은 시스템의 다른 부분과의 상호 작업을 최소화시키면서 부분적인 관점에서 평가하도록 설계되었다. 커널 기반의 벤치마킹은 실제 응용 프로그램의 부분을 평가하기 위해 작성된 특별한 프로그램이다. Iozone, Bonnie, Bonnie++, Winbench 등이 이 분류에 속한다. 일반적으로 커널 기반의 벤치마킹은 보다 더 집중적이면서 성능 병목현상에 국한되어서 평가할 수 있는 이점이 있다.

③ 추적 기반의 벤치마킹

이 방법은 시스템에서 실제적으로 추적 프로그램을 설정한 후, 시스템의 행위와 성능을 평가하기 위해 추적 로그를 분석한다. 이 방법은 다른 방법에 비해 오래 전부터 연구 되어왔다. 하나의 특별한 시스템에 대해 성능을 평가하는데 매우 좋은 방법이고 시스템 향상에 대해 귀중한 정보를 제공한다. 그러나 이 방법은 다른 시스템과의 비교 평가로써는 적합하지 못하다.

III. Iozone 도구를 이용한 저널링 파일 시스템 성능 평가

본 논문에서는 다양한 저널링 파일 시스템을 비교 평가하기 위해 커널 기반의 벤치마킹 도구로 Iozone을 선택하였다. 2장에서 기술한 것처럼 커널 기반 기반의 벤치마킹인 Iozone 도구는 보다 집중적이면서 성능 병목현상에 국한되어 평가할 수 있는 장점이 있다. 또한, Iozone 도구는 다양한 운영체제에서 사용될 수 있어 다양한 파일 시스템을 성능평가 할 수 있다[10]. 리눅스 우분투 12.04 커널 버전 3.2.0.23에서 하드디스크 5개를 추가 설치하여 본 논문에서 성능평가 해야 할 파일 시스템을 각각 생성시켜 비교 평가하였다. 메인 메모리(RAM)는 2GB이며, 블록 크기를 64KB로 하였다. 평가의 공정성을 보다 높이기 위해 20회의 비교 평가를 실시하여 평균값을 가지고 평가하였다.

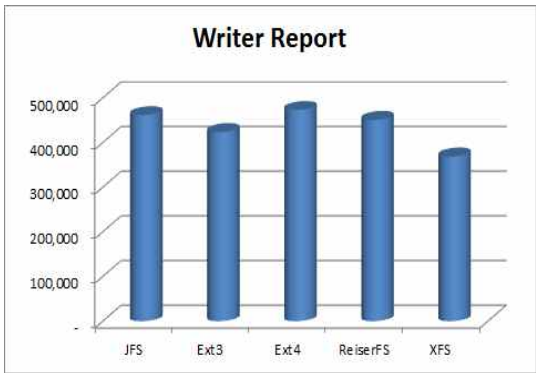

그림 2. 파일 쓰기 성능 비교
Fig. 2. Performance comparison of file write

그림 2는 새로운 파일에 쓰기(write) 성능 비교 평가를 나타내고 있다. 저널링 파일 시스템에서 데이터 저장은 새로운 파일에서 데이터를 단순히 저장하는 하는데 걸리는 시간 뿐만아니라, 메타 정보를 생성하는데 걸리는 시간도 포함하고 있다. 파일의 크기가 1GB일 때 JFS 파일 시스템, Ext4 파일 시스템과 ReiserFS 파일 시스템이 좋은 성능을 나타내고 있다.

Ext4 파일 시스템은 초당 전송률이 472,822KB로 가장 좋은 성능을 보이고 있으며, XFS 파일 시스템은 초당 전송률이 368,751KB로 가장 낮은 성능을 나타내었다. Ext4 파일 시스템은 XFS 파일 시스템보다 약 1.28배 정도 파일 쓰기 속도가 빨랐다.

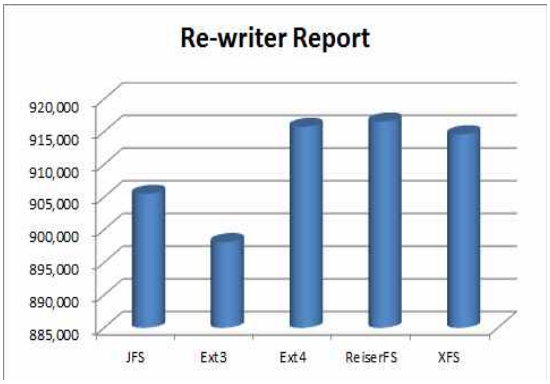

그림 3. 존재하는 파일에 쓰기 성능 비교
Fig. 3. Performance comparison of file re-write

그림 3은 다시 쓰기(re-write) 성능 평가를 나타내고 있다. 다시 쓰기란 파일을 새롭게 만든 것이 아니고 이미 존재하는 파일에 데이터를 추가적으로 쓰기하는 것이다. 다시 쓰기에서 Ext4 파일 시스템, ReiserFS 파일 시스템과 XFS 파일 시스템이 좋은 성능을 보이고 있다. 전반적으로 다시 쓰기 성능은 그림 2의 쓰기 성능과 비교하면 빠르게 전송시킴을 알 수 있다. 이는 다시 쓰기는 파일을 새롭게 생성시키지 않고 파일이 이미 존재하기 때문에 이와 관련된 메타 데이터도 새롭게 만들지 않기 때문이다. 예를 들어 Ext4 파일 시스템은 쓰기의 경우 초당 전송률이 472,822KB이고, 다시 쓰기를 수행하면 초당 전송률이 915,754KB로 약 194%정도로 대략 1.9배 정도 빠르게 전송한다.. Ext3 파일 시스템인 경우에도 새롭게 파일을 생성시키는 쓰기 성능은 초당 전송률이 423,451KB이고, 이미 파일이 존재하는 다시 쓰기 수행은 초당 전송률이 898,113KB로 약 212%정도로 2.1배 이상 빠르게 전송됨을 알 수 있었다.

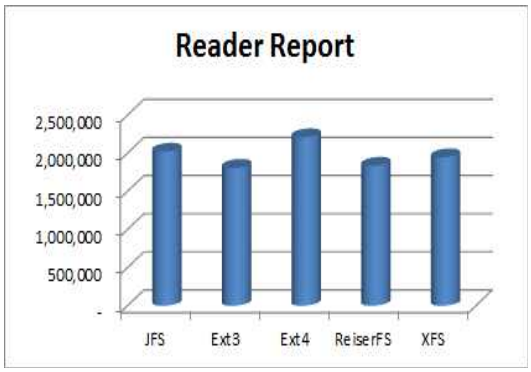

그림 4. 파일 읽기 성능 비교
Fig. 4. Performance comparison of file read

그림 4는 읽기(read) 성능 비교를 나타내고 있다. 읽기 성능은 존재하는 파일에 대한 데이터를 읽는 것이다. Ext4 파일 시스템이 초당 전송률이 2,209,285KB로 우수한 성능을 나타내고 있다. 가장 낮게 성능을 나타내고 있는 것은 초당 전송률이 1,808221KB인 Ext3 파일 시스템이다. 이는 Ext4 파일 시스템이 Ext3 파일 시스템 보다 1.22배 정도 빠르게 전송되었다.

읽기 성능은 그림 2의 쓰기 성능과 비교해 보면 빠르게 전송됨을 알 수 있다. 이는 단순히 데이터를 읽고 쓰는 속도 이외에 읽기의 경우 저널링 파일 시스템은 메타 데이터를 이용하여 데이터를 읽으나, 쓰기의 경우 메타 데이터를 생성시키기 때문에 속도가 차이가 많이 보이고 있다. 예를 들어 Ext4 파일 시스템인 경우 쓰기일 경우 초당 전송률이 472,822KB로 전송함에 비해 읽기인 경우 초당 전송률이 2,209,785KB이다. 읽기 성능이 쓰기 성능 보다 약 467% 정도로 4.6배 빠르게 전송되고 있다. 가장 늦은 Ext3 파일 시스템인 경우에도 427%로 대략 4.3배 빠르게 전송하였다.

그림 5는 램덤 읽기의 성능을 나타낸다. 램덤 읽기는 파일 안의 임의의 위치에서 접근하여 파일의 데이터를 읽었을 때의 성능을 나타낸다. Ext4 파일 시스템의 초당 전송률이 1,878,373KB로 가장 좋은 성능을 나타내고 있다.

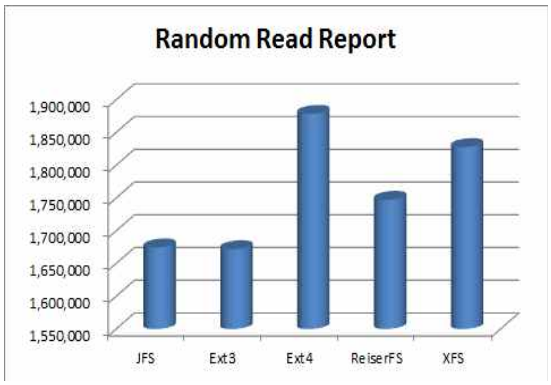

그림 5. 램덤 읽기 성능 비교
Fig. 5. Performance comparison of random read

JFS 파일 시스템과 Ext3 파일 시스템은 각각, 1,674,456KB와 1,671,767KB로 낮은 성능을 나타내고 있다. 읽기 성능과 램덤 읽기 성능을 비교하여 보면, 전반적으로 읽기 성능이 좋음을 알 수 있다. 이와 같은 이유는 램덤 읽기 경우 읽기와 비교하여 임의의 위치에 접근하기 때문에 추가적인 시간이 소요되기 때문이다. 예를 들어, Ext4 파일 시스템인 경우 읽기 성능의 경우 초당 전송률이 2,209,285KB이고 램덤 읽기의 경우 초당 전송률이 1,878,373KB로 85%정도 되었다. 이는 램덤하게 파일의 내용을 읽을 경우 이에 따른 추가적인 탐색 시간이 소요되기 때문에 읽기 보다 15% 정도 느린 전송을 보여주고 있다.

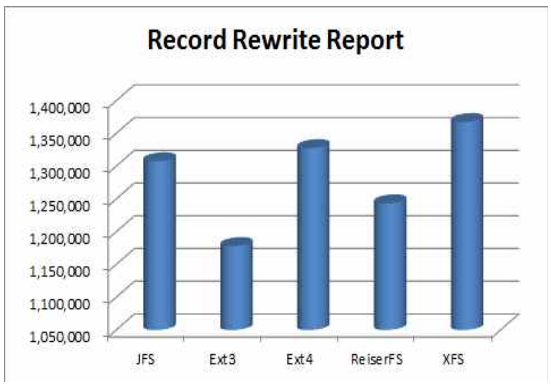

그림 6. 기록된 다시 쓰기 성능 비교
Fig. 6. Performance comparison of record re-write

그림 6은 기록된 다시 쓰기 성능 평가를 나타낸 것이다. 기록된 다시 쓰기는 파일의 임의의 특정한 부분을 쓰고, 그 부분을 다시 쓰기 성능을 평가하는 것이다. 운영체제 관점에 있어서 특정한 부분의 기억장소가 참조가 되면 그 근처의 기억장소가 계속 참조되는 공간적 구역성(spatial locality)을 의미한다. 이 특정한 부분을 핫 스팟(hot spot)이라고 부르며. 이는 CPU 데이터 캐쉬와 관련되어 있다. 핫 스팟 크기가 CPU 데이터 캐쉬보다 훨씬 작으며, 모든 데이터는 캐쉬가 되어 전송률 성능은 높을 것이고, 핫 스팟 크기가 CPU 데이터 캐쉬보다 크면 전송률은 낮아질 것이다. 기록된 다시 쓰기 성능은 XFS 파일 시스템과 Ext4 파일 시스템의 성능이 각각 초당 전송률이 1,367,552KB와 1,327,664KB로 우수하였다. 초당 전송률이 1,178,149KB로 Ext3 파일 시스템의 성능이 가장 낮았다. 일반적으로 파일의 특정한 부분을 쓰기하고 다시 쓰기 할 경우 새로운 파일 데이터의 쓰기 성능이나 이미 생성되어 있는 파일의 데이터는 다시 쓰기 성능보다도 성능이 우수하게 나타나고 있다. 이러한 이유는 핫 스팟인 경우 한번 쓰기한 경우 CPU 데이터에 캐쉬되어 다시 쓰기할 때 전송률이 좋게 나타나기 때문이다. Ext4 파일 시스템인 경우 쓰기는 초당 전송률 472,822KB이고, 다시 쓰기는 915,754KB이나 기록된 다시 쓰기의 성능은 1,327,664KB로 쓰기에 비해 2.8배, 다시 쓰기에 비해 1.4배로 전송률이 높음을 알 수 있다.

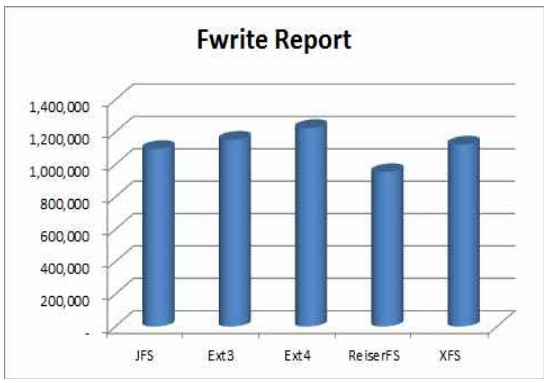

그림 7. fwirte() 성능 비교
Fig. 7. Performance comparison of fwrite()

그림 7은 fwrite() 라이브러리 함수를 수행 할 때 성능평가를 나타내고 있다. fwrite() 함수는 한 블록의 데이터를 스트림 파일에 기록을 수행한다. fwrite() 함수는 사용자 주소공간에 버퍼를 이용하여 수행하기 때문에 적은 용량에 데이터를 기록하면 빠르게 수행되어 진다. fwrite() 함수는 새로운 파일을 생성시키기 때문에 메타 데이터도 새롭게 생성시키는 오버헤드 시간도 포함되고 있다. 그림 7에 보이는 것처럼 Ext4

파일 시스템이 초당 전송률이 1,224,973KB로 성능이 가장 우수하였다. 가장 낮은 성능은 952,236KB을 전송하는 ReiserFS 파일 시스템이다. 단순히 쓰기 성능보다 사용자 주소 공간에 버퍼를 이용한 fwrite() 함수를 이용한 데이터 쓰기 보다 빠르게 전송되었다. 예를 들어 Ext4 파일 시스템인 경우, 단순 쓰기 성능의 초당 전송률이 472,822KB인데 반해 fwrite() 함수를 이용한 성능은 초당 전송률이 1,224,973KB으로 대략 2.6배 정도 빠르게 수행되었으며, XFS 파일 시스템인 경우 쓰기 성능의 초당 전송률이 368,751KB이며, fwrite() 함수를 이용한 성능은 초당 전송률이 1,122,783KB으로 대략 3.2배 빠르게 수행되었다.

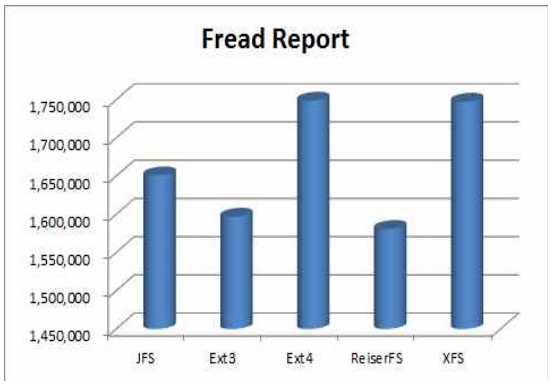

그림 8. fread() 성능 비교
Fig. 8. Performance comparison of fwrite()

그림 8은 fread() 라이브러리 함수를 수행할 때 성능평가를 나타내고 있다. fread() 함수는 스트림 파일에 한 블록의 데이터 읽기를 수행한다. fread() 함수는 fwrite() 함수와 마찬가지로 사용자 주소공간에 버퍼를 이용하여 수행하기 때문에 적은 용량에 데이터를 읽기를 수행하면 빠르게 수행되어 진다. 그림 8에 보이는 것처럼 Ext4 파일 시스템이 초당 전송률이 1,748,700KB로 성능이 가장 우수하였다. 가장 낮은 성능은 1,580,700KB을 전송하는 ReiserFS 파일 시스템이다. fread() 함수는 존재하는 파일에서 블록을 읽어 새로운 파일을 생성하지 않기 때문에 메타 데이터도 생성시키지 않는다. 새롭게 파일 생성시키고, 이에 따른 메타 데이터도 새롭게 생성되는 그림 7과 비교해 보면, Ext4 파일 시스템의 경우 fwrite()는 1,224,973KB이고 fread()인 경우 1,748,700KB으로 약 1.4 배 정도 fread()가 빠르게 전송되었다. ReiserFS 파일 시스템인 경우 fwrite()는 952,236KB이고 fread()인 경우 1,580,700KB으로 약 1.6 배 정도 fread()가 빠르게 수행되었다.

IV. 결 론

리눅스가 부팅시 fsck가 시작되어 시스템의 /etc/fstab 파일에 있는 모든 로컬 파일 시스템의 일관성을 확인하기 때문에 많은 시간이 필요하게 된다. 저널링 파일 시스템은 저널이라는 공간에 메타 데이터를 관리하여 파일 시스템의 손상이 발생시에 복구 시간과 복구 확률을 높여주는 파일 시스템이다. 본 논문은 현재 사용되고 있는 저널링 파일 시스템인 JFS2 파일 시스템, Ext3 파일 시스템, Ext4 파일 시스템, ReiserFS 파일 시스템, XFS 파일 시스템을 커널 기반 벤치 마킹 도구인 Iozon 도구를 이용하여 성능을 비교하여 보았다. 성능을 평가한 결과 Ext4 파일 시스템이 대부분 높은 성능을 보였으며, 대체로 Ext3 파일 시스템이 낮은 성능을 나타내었다. 파일의 읽기 성능인 경우 쓰기의 성능 보다 대략 4배 이상 빠르게 전송되었다. 쓰기 성능인 경우 존재하고 있는 파일에 데이터를 쓸 경우 새롭게 파일을 생성하여 쓰는 성능에 비해 대략 2배 정도 빨랐다. 이러한 이유는 저널링 시스템의 저널에 저장되어 있는 메타 데이터를 이용하기 때문이다. 사용자 주소공간에 버퍼를 이용하는 fwrite() 성능은 쓰기 성능보다 2.5배에서 3.2배정도 빠르게 수행되었다.

Reference

[1] http://www.ibm.com/developerworks /kr/library/l-journaling-filesystems/
[2] Hong-jin, Park, A Study on the Performance Evaluation of File Systems using Kernel-based Benchmarking Tool in the Linux, *The Journal of Korean Institute of Information Technology,* Vol. 10, No. 12, pp. 120-126, Dec. 2012.
[3] http://www.softpanorama.org/Internals/
[4] http://wiki.archlinux.org/index.php/Ext3
[5] http://wiki.archlinux.org/index.php/Ext4
[6] http://en.wikipedia.org/wiki/Hans_Reiser
[7] http://en.wikipedia.org/wiki/XFS

[8] Matti Vanninen and James Z. Wang. On Benchmarking Popular File Systems, *In Proceedings of the International Conference on Computer, Communication and Control Technologies (CCCT'03)*, Orlando, FL, August 2003.
[9] Traeger A. et. al., "A nine year study of file system and storage benchmarking", *ACM Transactions on Storage (TOS) TOS*, Volume 4, No. 2, pp. 54~56 May 2008.
[10] http://www.iozone.org/

박 홍 진 (Hong-Jin Park)

1993년 2월 : 원광대학교 컴퓨터공학과(공학사)
1995년 8월 : 중앙대학교 컴퓨터공학과(공학석사)
2001년 8월 : 중앙대학교 컴퓨터공학과(공학박사)
2001년 9월~현재 : 상지대학교 컴퓨터정보공학부 교수
관심분야 : 시스템 프로그래밍, 운영체제, 분산 시스템