JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 19, No. 3, pp. 497-503, Mar. 2018

XFS 파일 시스템 채택에 따른 리눅스 시스템 변화 분석

성 경
목원대학교 융합컴퓨터미디어학부

Analysis of Linux System changes by adoption XFS File System

Kyung Sung
Division of Convergence Computer & Media, Mokwon University, Daejeon 35349, Korea

[요 약]

엔터프라이즈 리눅스 시장의 대표주자인 RHEL 7에서 기본 파일 시스템을 EXT에서 XFS로 변경하면서 파일 시스템의 크기, 파일 크기 등과 같이 최대 지원 사양을 대폭 증가시켰다. 단순히 지원 사양만 증가시킨 것 아니라, 데몬 기반으로 동작하면서 고용량 디스크 및 SSD(solid state drive)와 같은 고성능 디스크에서 탁월한 성능을 보이는 것으로 나타나고 있다. 파일 시스템의 변경은 관련 명령어의 변경, 백업 도구의 변경, 디스크 쿼터 설정 변경과 같은 직접적인 운영 기법의 변화를 의미한다. XFS 파일시스템의 변경은 리눅스 시스템 운영에 많은 변화를 주고 있지만 서버 분야에서 차지하는 리눅스 운영체제의 위치를 더욱 굳건히 하게 되는 계기가 될 것으로 판단된다.

[Abstract]

RHEL 7, the leader in the enterprise Linux market, has dramatically increased the maximum support specification, such as file system size, file size, etc., by changing the default file system from EXT to XFS. It's not just an increase in support specifications, it's working on daemons, and it's showing excellent performance on high-performance disks such as high-capacity disks and solid state drives. Changes in the file system mean changes in direct operating techniques, such as changing related commands, changing backup tools, and changing disk quota settings. The changes to the XFS file system are making a lot of changes to the operation of the Linux system, but we believe that the position of the Linux operating system in the server field will become stronger.

Key word : EXT, File system, Linux, RHEL 7, XFS

색인어 : EXT, 파일 시스템, 리눅스, 레드햇 엔터프라이즈 리눅스 7, XFS

http://dx.doi.org/10.9728/dcs.2018.19.3.497

Received 23 March 2018; **Revised** 29 March 2018
Accepted 29 March 2018

***Corresponding Author; Kyung Sung**

Tel: +82-42-829-7643
E-mail: skyys04@mokwon.ac.kr

Ⅰ. 서 론

1991년에 리누스 토발즈(Linus Torvalds)가 교육용 유닉스인 미닉스(minix)를 기반으로 리눅스 커널을 만들고, 인터넷에 공개한 후 일부 개발자들의 도움을 받아서 리눅스를 운영체제화하는 시기에는 낮은 사양의 PC(personal computer)에서도 잘 구동되는 유닉스 운영체제의 아류 정도로 인식되어왔다. 그러나 리눅스를 공개 소프트웨어 프로젝트인 GNU(gnu is not unix) 프로그램에 적용시키는 과정을 거치면서 공개 소프트웨어를 지지하는 전 세계 개발자들의 큰 호응을 얻게 된다[1]. 프로그램의 소스(source)가 공개되어 있고 누구나 패키징하여 자유롭게 사용할 수 있으며 비용 부담 없이 사용할 수 있다는 장점이 부각되면서, 많은 기업들도 리눅스를 사용하기 시작했다. 현재 리눅스는 커뮤니티 차원에서 배포하는 데비안, 젠투 리눅스 등과 기업 차원에서 배포하는 레드햇 리눅스, 수세 리눅스, 우분투 등 300여개 이상의 리눅스 배포판이 존재하고 있다. 리눅스의 사용 영역도 데스크톱 및 서버 분야뿐만 아니라, 안드로이드와 같은 모바일 분야, TV나 냉장고와 같은 가전제품, 자동차에 장착되는 IVI(in-vehicle infotainment)에도 사용될 만큼 거의 모든 분야에 활용되고 있다[2]. 특히 서버 분야에서는 유닉스의 시장을 대체하면서 점유율을 계속적으로 끌어올리고 있다. 리눅스 서버 분야에서 독보적인 위치에 있는 레드햇의 Red Hat Enterprise Linux(이하 RHEL)가 기본 파일 시스템을 전통적인 ext에서 XFS로 변경하면서 엔터프라이즈 리눅스로서의 성능적인 향상도 꾀하고 있다. 운영체제를 이용하는 사용자 입장에서는 크게 달라졌다고 느낄 수 있는 파일 시스템의 변경은 시스템 운영에 있어서도 다양한 변화를 주고 있다. 본 논문에서는 레드햇의 상용 버전 리눅스인 RHEL 7 버전을 기반으로 XFS 파일 시스템으로 변경에 따른 시스템의 변화에 대해 분석한다.

Ⅱ. 파일 시스템의 이해

2-1 파일 시스템의 개요

파일 시스템(file system)이란 운영체제가 파티션이나 디스크에 데이터를 저장하고, 읽고, 쓰고 찾기 위해 구성하는 일련의 체계를 의미하는데, 운영체제가 사용자에게 제공하는 가장 직접적인 서비스 형태 중에 하나이다[3]. 파일 시스템의 구성은 운영체제 설치 시에 일어난다. 운영체제를 설치하면 파티션 분할 작업 후에 포맷(format)이라는 행위를 가장 먼저 하게 되는데, 포맷은 파일을 저장하기 위해 디스크를 일정한 크기로 분할하고 주소를 설정하는 작업이라고 할 수 있다. 각각의 운영체제들은 포맷이라는 작업을 통해 고유한 파일 시스템을 구축하게 되고, 다양한 규칙을 설정하게 된다. 대부분의 운영체제들은 파일이라는 단위로 저장하고, 파일에 이름을 부여한 뒤에 디렉터리에 저장한다. 이 과정에서 파일명의 길이를 제한하기도 하고, 어떤 문자들이 사용될 수 있는지를 정하기도 한다. 파일명에 확장자를 쓰도록 지정하기도 하며, 확장자의 길이도 제한할 수 있다. 또한, 파일 시스템은 지원하는 파티션의 개수, 크기, 파일 크기 등에도 직접적인 관계가 있으며, 파일 복구와 같은 기능을 부여하기도 한다. 따라서 파일 시스템의 성능은 운영체제의 성능에도 밀접한 관계가 있다고 볼 수 있다.

2-2 파일 시스템의 기능

파일 시스템은 사용자가 파일을 생성(create), 수정(modify), 삭제(delete)할 수 있도록 제공한다. 또한 사용자가 파일을 사용하기 적합한 형태의 구조로 구성하고, 다양한 추가 정보 제공한다. 다른 사용자와의 파일을 공동으로 사용할 수 있는 적절한 제어 방법 제공하고 파일 공유를 위하여 관독 접근, 기록 접근, 수행 접근 등의 다양한 접근 제어 방법 제공한다. 정보 손실이나 파괴를 방지하기 위하여 백업(backup)이나 복구(recovery)를 위한 기능도 준비하고 사용자와 장치 간의 독립성(device independence)을 유지하기 위해, 사용자가 물리적인 장치 이름 대신에 적절한 이름 제공한다. 아울러 저장된 정보가 안전하게 보호되고 비밀이 보장될 수 있도록 정보의 암호화(encryption) 및 복호화(decryption) 기능도 제공한다. 최근에는 사용자의 편의성과 고려해서 파일이나 디렉터리에 접근하기 쉬운 인터페이스 및 명령어를 제공한다.

Ⅲ. 리눅스 파일 시스템의 역사와 ext

3-1 리눅스 파일 시스템의 역사

리눅스는 유닉스의 영향을 받기도 하였고, 개방적인 특징으로 인해 다양한 파일 시스템을 지원한다. 초기의 리눅스는 미닉스(minix) 파일 시스템을 기반으로 만든 ext(extended file system) 파일 시스템을 사용했다. 1987년에 개발된 미닉스 파일 시스템은 파티션 사이즈가 64MB로 제한이 있고, 파일 이름도 14자까지만 지원되었다. 또한 단일 타임스탬프(timestamp) 체제여서 주로 램디스크나 부팅디스크에 이용되었고, 디스크에 적용하기에는 미흡한 부분이 많았다. 1992년 4월에 등장한 ext는 전통적인 유닉스 파일 시스템인 UFS의 구조를 기반으로 미닉스 파일시스템의 파티션 크기 제한과 파일이름 제한을 해결한 파일 시스템으로 리눅스 커널에 구현된 최초의 파일 시스템이다. ext는 2GB의 데이터와 파일 이름도 255자까지 지원하였다. 그러나 파일 접근에 다른 타임스탬프 및 아이노드(i-node) 수정 등을 지원하지 않는 문제점을 가지고 있었다. 1993년 1년에 등장한 ext2는 ext2(second extended file system)의 대표적인 문제점인 아이노드의 불변성과 타임스탬프 수정 문제를 해결하였다. 특히 고용량 디스크 사용 등에 대비하여 확장성에 염두에 두고 설계한 파일 시스템이다. 2011년 11월에 등장한 ext3(third extended file system)는 ext2의 확장판으로 저널링 파

일 시스템(journaling file system) 기능을 탑재하고 커널 2.4.15 버전부터 포함되었다. ext3에는 ACL(access control list)를 통한 접근 제어를 지원해서 전통적인 유닉스 체제가 가지고 있는 파일 접근 권한에 대한 문제점을 어느 정도 해소할 수 있었다. ext3의 확장 버전인 ext4(fourth extended file system)는 2006년 6월 커널 2.6.19 버전에 제안되었고, 2008년 10월 커널 2.6.28에 정식으로 채택되었다. ext2 및 ext3의 호환성이 있는 확장 버전으로 64비트 기억 공간 제한을 없애고, 최대 1EB의 디스크 볼륨과 16TB의 파일을 지원하는 등 대형 파일 시스템과 관련된 기능을 대폭 강화하였다. 그러나 대용량 디스크의 사용이 보편화되고 더 큰 파일 크기 및 파일시스템 크기가 요구되면서 가장 최신 버전의 엔터프라이즈 리눅스인 RHEL 7에서는 ext4 대신에 XFS를 기본 파일 시스템을 채택하였다.

3-2 저널링 파일 시스템

리눅스에 사용되는 파일 시스템이 서버 분야에서 두각을 나타내기 시작한 계기로는 저널링 파일 시스템을 손꼽을 수 있다. 저널링 파일 시스템은 파일 시스템에 대한 변경사항을 반영하기 전에 저널이라 부르는 로그에 변경사항을 저장하여 추적이 가능하게 만든 파일 시스템이다. 시스템에 충돌현상이 발생하거나 전원 문제가 발생된 경우에 데이터 복구 확률을 높여준다. 특히 파일 시스템을 검사하는 명령인 fsck(file system check)에 걸리는 시간을 단축하기 위해 데이터를 디스크에 쓰기 전 로그(log)에 데이터를 남겨 시스템의 비정상적인 종료에도 로그를 사용해 fsck보다 빠르고 안정적인 복구 기능을 제공하는 기술이다. 리눅스 초기에 사용하던 ext2 파일 시스템의 경우에는 시스템이 갑작스럽게 동작을 멈추면 어떠한 수정을 하고 있었는지 전혀 알 수가 없었다. 따라서 이를 복구하기 위해서는 관리자가 직접 fsck라는 명령을 입력해야 했고, fsck는 슈퍼 블록, 비트맵, 아이노드 등을 모두 검사해야 되므로 때문에 많은 시간을 걸렸다. 저널링 기술을 사용한 파일 시스템은 파일을 실제로 수정하기 전에 우선 로그에 수정된 내용을 저장해서 비정상적으로 동작이 멈추었더라도 시스템 복구를 위해 단지 로그만 검사하면 된다. 이 로그 정보를 바탕으로 파일 시스템에 수정 내용을 적용하면 되기 때문에 속도도 빠르고, 복구의 안정성도 뛰어난 성능을 보이게 된다. 이러한 저널링 기술이 적용된 파일 시스템은 ext3, ext4, XFS, JFS, ReiserFS 등이 있다.

3-3 ext 파일 시스템

일반적으로 디스크 드라이브 이용 시에 파티션을 분할하고 흔히 포맷(Format)이라고 부르는 작업을 통해 파일시스템을 생성한다. ext 파일 시스템들은 ext2 파일 시스템을 기반으로 확장되었는데, 해당 파일 시스템의 구조를 살펴보면 그림 1과 같다.

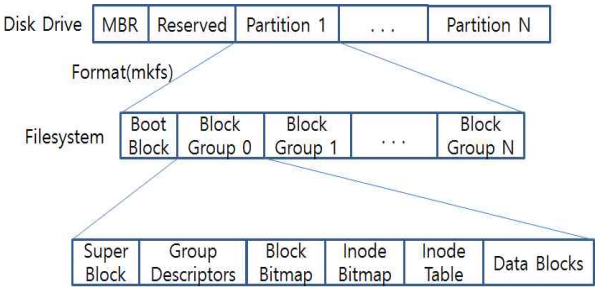

그림 1. ext2 파일 시스템의 구조
Fig. 1. Structure of the ext2 file system

ext2 파일 시스템은 부트 블록(boot block, 일반적으로 부트 섹터라 부름)과 블록 그룹(block group)으로 구성된다. 블록(block)이란 파일 시스템에서 기본적으로 데이터를 저장하는 단위로서 메모리에서 입출력 작업 시 한 번 거칠 때 읽거나 쓰여 지는 단위이기도 하다. 보통 리눅스에서 1KB ~ 8KB 블록의 크기를 지정가능한데, 최근 배포되는 리눅스를 설치하면 기본 4KB로 지정된다. 블록 그룹은 슈퍼블록(super block), 그룹 기술자(group descriptor), 블록 비트맵(block bitmap), 아이노드 비트맵(inode bitmap), 아이노드 테이블(inode table, 또는 아이노 inode block이라고도 함), 데이터 블록으로 구성되어 있다. 슈퍼 블록과 그룹 기술자는 마운트될 때 첫 번째 그룹은 Group-0의 정보를 가져오지만, 손상될 때를 대비하여 모든 Group에 사본이 저장되어 있다.

슈퍼 블록은 파일 시스템에 대한 전체적인 정보를 가지고 있는데 주요 정보로는 매직 넘버(magic number: 특정 파일 시스템임을 알리는 정보로서 ext 파일 시스템인 경우에는 0xEF53이다.), 마운트 정보, 전체 아이노드 수 및 남은 수, 전체 블록 수 및 남은 수, 블록 그룹 번호, 블록 크기, 그룹 당 블록 수 등이 있다. 그룹 기술자는 각각의 블록 그룹을 기술하는 자료 구조로서, 저장되는 주요 정보는 블록 비트맵, 아이노드 비트맵, 아이노드 테이블(또는 아이노드 블록)이다. 블록 비트맵은 블록 사용 현황을 비트(bit)로 표현해주고, 아이노드 비트맵은 아이노드 할당 상태를 비트로 표현해준다. 아이노드 테이블은 파일이나 디렉터리 관리를 위해 아이노드라는 것을 사용하는데, 이 아이노드에 대한 정보가 들어 있는 영역이다. 일반적으로 'ls -l' 명령 실행 시 나타내는 정보인 파일 유형, 접근권한, 하드링크 수, 소유자, 소유그룹명, 파일크기, 날짜 관련 정보, 파일명 등을 저장한다.

데이터 블록은 파일이 보관해야 하는 정보를 저장하는 영역으로 파일의 데이터가 존재한다. 추가적으로 간접 블록(indirection block)과 홀(hole)이 있다. 간접 블록은 추가적인 데이터 블록을 위한 포인터들이 사용할 동적으로 할당되는 공간이다. 실제적으로 아이노드는 적은 수의 데이터 블록을 가지고 있다. 그러므로 더 많은 데이터 블록이 필요할 경우에 이를 지정할 포인터가 필요하게 되는데, 이 포인터들이 사용할 동적인 블록이 간접 블록이다. 홀은 아이노드나 간접 블록 안의 데이터

블록의 주소로 특별한 값을 저장한다. 홀은 파일 시스템에 의해서 파일 안에 자리하게 된다. 하지만 이 홀을 실질적으로 디스크 상에 공간은 할당되지 않는다. 단지 0byte가 파일 안에 특정 공간을 차지하고 있다고 가정한다.

Ⅳ. XFS

4-1 개요와 특징

XFS는 SGI(silicon graphics, inc)에서 고성능의 64비트 저널링 파일시스템을 구현하기 위해 1993년에 개발되었다. 1994년 SGI사의 유닉스 계열 운영체제에 IRIX 5.3 버전에 처음 출시되었다. 2000년 5월 GNU GPL 라이선스로 전환되고, 2001년 리눅스 커널에 포함되었다. 현재 대부분의 리눅스 배포판에서 지원되고, 레드햇 계열 리눅스인 RHEL 7 버전에서는 기본 파일 시스템으로 채택하고 있다[4][5].

XFS는 빠른 복구를 위해 메타데이터 저널링을 지원하고 마운트가 되어 활성화된 상태에서도 조각 모음 및 확장을 지원한다. 데이터 읽기 및 쓰기 트랜잭션으로 인한 성능 저하를 최소화한다. XFS의 저널링 구조와 알고리즘은 트랜잭션에 대한 로그 기록을 신속하게 할 수 있도록 최적화되어 있다. raw I/O 성능에 가까운 성능을 낼 수 있는 뛰어난 처리량을 보인다. 완전한 64비트 파일 시스템이기 때문에 높은 확장성을 제공한다.

XFS는 블록을 기반으로 하는 파일 시스템을 한계를 극복하기 위해 익스텐트(extent) 기반 할당을 사용하여 지연 할당 및 명시적인 사전 할당과 같은 여러 할당 체계를 가지고 있다. 익스텐트 기반 할당은 파일 시스템에서 사용된 공간을 추적하는 것보다 간결하고 효율적인 방법을 제공하여 메타데이터에 의해 소비되는 공간 및 조각화를 줄임으로써 대용량 파일의 성능을 향상시킨다. 지연 할당은 파일이 연속적인 블록 그룹에 기록될 가능성을 높이는 것으로 단편화를 줄이고 성능을 향상시킨다. 사전 할당은 애플리케이션이 사전에 기록해야 할 데이터양을 알고 있는 경우 완전히 조각화하는 것을 방지하는데 사용될 수 있다[6].

표 1. XFS 최대 지원 사양
Table 1. XFS Limits

Item	Value
Max. Volume size	8 exbibytes
Max. file size	8 exbibytes
Max. number of files	2^{64}
Max. filename length	255 bytes

4-2 XFS의 자료 구조

XFS는 탐색 기법으로 B-트리(b-tree)를 사용하여 우수한 I/O 확장성을 제공하고 모든 사용자 데이터 및 메타데이터를 인덱스(index)한다. B-트리에 대해 살펴보면 다음과 같다[7].

1) B-트리

데이터베이스와 파일 시스템에서 널리 사용되는 트리 구조의 일종으로 이진 트리를 확장해 하나의 노드가 가질 수 있는 자식 노드의 최대 숫자가 2보다 큰 트리 구조이다. 따라서 B 트리의 노드는 일반적인 이진 트리보다 많은 자식 노드를 가질 수 있으므로 트리의 높이가 낮아져서 방문해야 할 노드의 수가 줄어들게 된다. 일반적으로 자식 노드의 수 천개를 가지는 구조로 설계되는 정확한 개수는 디스크 블록 크기에 따라 결정된다. B 트리는 자료를 정렬된 상태로 보관하는데 그림 2처럼 각 노드에는 검색에 사용될 키를 가지고 있으며 각각의 키는 그에 연관된 데이터를 가리키는 포인터 정보를 유지하게 된다[8].

2) B+ 트리

B+ 트리는 모든 레코드들이 트리의 가장 하위 레벨인 리프 노드(leaf node)에 정렬되어 있고, 이를 순차적으로 연결한 구조이다. B+ 트리에서 중간 노드에는 리프 노드에 있는 키 값을 찾아갈 수 있는 키들만이 내부 블록에 저장된다. 리프 노드들은 모두 링크드 리스트(Linked List)로 연결되어 있어서 파일의 내용을 순서대로 읽는 경우와 순차적 접근 처리에 효율적이다. B+ 트리 구조를 사용하는 파일 시스템에는 XFS 이외에도 ReiserFS, JFS, NTFS 등이 있다.

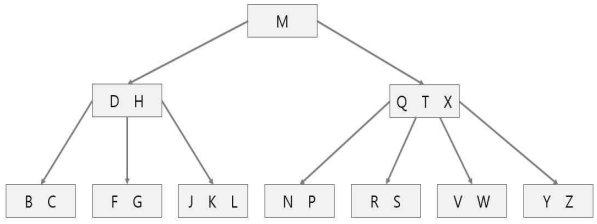

그림 2. B-트리의 구조
Fig. 2. Structure of B-tree

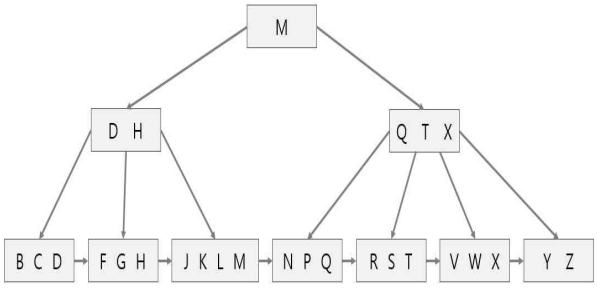

그림 2. B+ 트리의 구조
Fig. 2. Structure of B+ tree

4-3 XFS의 디스크 구조

XFS는 할당 그룹(allocation group)이라는 단위로 나눠지고, 각각의 할당 그룹은 독립적으로 존재하며 병렬적으로 처리된다. 파일 시스템을 생성할 때 할당 그룹의 크기와 수를 지정할 수 있는데 이를 지정하지 않은 경우에는 기본적으로 주어진 디스크를 8등분하여 8개의 할당 그룹을 생성한다. 각각의 할당 그룹의 0번 블록에는 슈퍼 블록 정보가 저장되고, 나머지 블록에는 할당 그룹 헤더 정보가 저장된다. 마운트 시에는 첫 번째 할당 그룹의 슈퍼 블록만을 사용하며 나머지는 슈퍼 블록의 데이터가 깨친 상황에서 응급 복구 시에 사용된다. 할당 그룹 헤더에는 남은 공간 정보, 아이노드 정보 등이 저장된다.

아이노드의 구조는 디스크 상에 직접 기록되는 데, 아이노드를 필요에 따라 크기가 증가될 수 있는 가변적인 구조로 설계로 되어 있다.

Ⅴ. 리눅스 시스템 변화 분석

5-1 XFS 데몬에 의한 처리

XFS는 웹 서버, 메일 서버, DNS 서버 등과 같은 네트워크 서비스와 유사하게 데몬(daemon) 형태로 파일 시스템 관련 처리를 담당한다. XFS는 기본적으로 4 종류의 데몬을 이용하는데, xfssyncd, xfsbufd, xfsdatad, xfslogd가 해당된다[9]. xfssyncd는 로그 정보와 메타 데이터 정보들을 기록하는 역할을 수행하고, xfsbufd는 I/O 요청을 처리한다. xfsdatad와 xfslogd는 작업 큐에 수행할 작업이 있는 경우에 관련 함수를 호출해서 수행하는 역할을 담당한다. RHEL 7에서는 그림 3과 같이 더 많은 데몬을 운영해서 파일 시스템 관련 작업의 처리 속도를 높이고 있다.

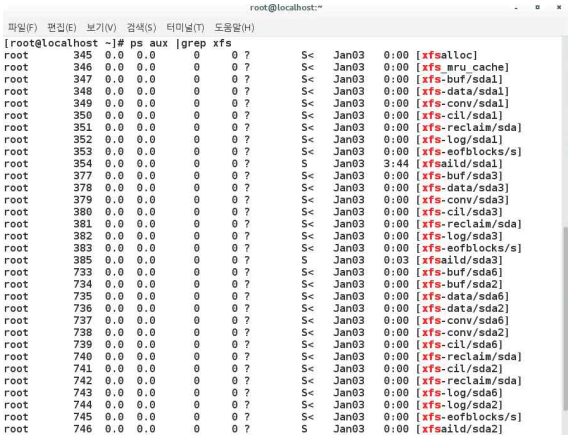

그림 3. RHEL 7의 XFS 데몬
Fig. 3. XFS daemon on RHEL 7

표 2. EXT 파일 시스템의 저장 용량 제한
Table 2. EXT's File systems and storage limits

Item	ext2	ext3	ext4	
			RHEL 6	RHEL 7
Maximum file size	2GB	2TB	16TB	
Maximum file system size	4TB	16TB	16TB [1EB]	50TB [1EB]
Maximum subdirectories	32000	32000	65000	

표 3. XFS 파일 시스템의 저장 용량 제한
Table 3. File systems and storage limits in XFS

Item	XFS	
	RHEL 6	RHEL 7
Maximum file size	100TB [8EB]	500TB [8EB]
Maximum file system size	300TB [16EB]	500TB [16EB]
Maximum subdirectories	unlimited	

5-2 파일 시스템 및 저장 공간 제한의 확대

리눅스 운영체제에서 기본적으로 사용되는 파일 시스템이 ext 계열에서 XFS로 변경되면서, 사용자 입장에서 가장 체감할 수 있는 영역이 파일 시스템 및 저장 공간의 제한이 크게 확대된 것이다. 그 중에서 파일 최대 크기 및 파일 시스템 최대 크기가 대폭 상향되고, 생성할 수 있는 서브 디렉터리의 개수 제한이 풀리면서 고용량 하드 디스크 사용 시에 발생할 수 있는 불편 요소를 크게 개선하였다. RHEL 기준으로 ext와 XFS 파일 시스템의 저장 용량 제한 내용을 정리해보면 표 2 및 표 3과 같다[10].

5-3 파일 시스템 관련 명령어의 변경

파일 시스템의 변경은 특정한 파일 시스템을 기반으로 사용되는 명령어의 변경을 의미한다. RHEL 7 버전의 파일 시스템이 XFS로 바뀌면서 새롭게 등장한 명령어의 확인 방법은 관리자 계정인 root가 사용하는 명령어가 들어있는 디렉터리인 /sbin 이나 /usr/sbin 디렉터리에서 확인할 수 있다. 해당 디렉터리를 ls 명령으로 살펴보면 그림 4처럼 XFS 관련 명령어들이 많이 추가된 것을 확인할 수 있다. 주요 변경 사항을 정리해보면 다음과 같다.

그림 4. XFS 파일 시스템 관련 명령어
Fig. 4. Commands Related to XFS File systems

1) 파일 시스템 생성 명령

ext 계열 파일 시스템을 사용하는 시절에는 파일 시스템 생성 명령어로 전통적으로 사용하던 mkfs를 비롯하여 mke2fs, mkfs.ext3, mkfs.ext4 등을 사용했다. XFS 파일 시스템 기반에서도 mkfs 명령어가 '-t xfs' 형식으로 XFS 파일 시스템을 지원하지만 실제 실행되는 명령어는 mkfs.xfs이다. 따라서 XFS 환경에서 파일 시스템 생성 명령어는 mkfs.xfs를 권장하고 있다.

2) 파일 시스템 점검 명령

파일 시스템을 점검 및 복구하는 명령으로 fsck, e2fsck, fsck.ext2, fsck.ext3 등의 명령을 사용했으나 XFS 파일 시스템에서는 지원하지 않는다. XFS 파일 시스템에서는 fsck.xfs나 xfs_repair를 사용해야 한다.

3) 유용하지만 사용할 수 없는 명령

유용하지만 사용할 수 없는 대표적인 명령어로는 tune2fs, dume2fs, debugfs가 있다. tune2fs는 최대 마운트 횟수, 마운트 횟수, 저널링 옵션 등 파일 시스템의 다양한 파라미터값을 수정하여 튜닝(tuning)할 때 사용하는 명령이다. debugfs는 블록 사이즈, 슈퍼 블록 등과 같이 파일 시스템의 상태값을 수정하는 명령이다. dumpe2fs는 슈퍼 블록, 블록 사이즈 등 파일 시스템의 정보를 출력하는 명령이다. 마지막으로 오랫동안 유용하게 사용되었던 파일 시스템 백업 및 복구 명령어인 dump와 restore도 XFS 파일 시스템에서는 사용할 수 없다.

4) 새롭게 등장한 유용한 명령

XFS 파일 시스템에 새롭게 등장한 명령으로는 xfs_admin, xfs_growfs, xfs_info를 손꼽을 수 있다. 이 명령어는 기존 명령어인 tune2fs, dumpe2fs, debugfs의 기능을 대체하고 있다. xfs_admin는 파일 시스템의 UUID(universally unique identifier), 라벨(label) 등과 같은 파라미터(parameter) 값을 변경하는 명령이다. xfs_growfs와 xfs_info는 XFS 파일 시스템을 확장하는 명령이다. xfs_info 명령은 분할된 파티션의 정보를 출력하는 명령인데, 이 명령어를 실행하면 'xfs_growfs -n ' 옵션을 사용한 것과 동일하다.

5) 디스크 쿼터(quota) 설정 변경

디스크 쿼터란 사용자 또는 그룹 단위로 디스크의 사용량 및 생성할 수 있는 파일의 개수(i-node의 수)를 제한하는 것을 의미한다. ext 파일 시스템 환경에서는 convertquota, quotacheck 명령을 사용해서 관련 설정 파일을 관리했는데, XFS 환경에서는 xfs_quota라는 명령을 사용한다. 특히 xfs_quota 명령은 edquota, repquota, setquota의 역할을 모두 수행할 수 있다. 아울러 관련 명령어가 바뀌면서 /etc/fstab에 설정하는 쿼터 설정 옵션도 usrqutoa, grpquota에서 uquota, gquota로 변경되었다. 이러한 외적인 변경 이외에도 특정 디렉터리(프로젝트) 단위로도 제한이 가능해져 기능적으로도 확장되었다.

Ⅵ. 결 론

엔터프라이즈 리눅스 시장의 대표주자인 RHEL 7에서 기본 파일 시스템을 XFS로 변경하면서 파일 시스템의 크기, 파일 크기 등과 같이 최대 지원 사양을 대폭 증가시켰다. 단순히 지원 사양만 증가시킨 것 아니라, 데몬 기반으로 동작하면서 고용량 디스크 및 SSD(solid state drive)와 같은 고성능 디스크에서 탁월한 성능을 보이는 것으로 나타나고 있다. 그러나 파일 시스템의 변경은 리눅스 시스템을 운영하는 관리자 입장에서는 관련 명령어의 변경, 백업 도구의 변경, 디스크 쿼터 설정 변경과 같은 직접적인 운영 기법의 변화를 의미한다. 이와 같이 XFS 파일시스템의 변경은 리눅스 시스템 운영에 많은 변화를 주고 있지만 서버 분야에서 차지하는 리눅스 운영체제의 위치를 더욱 굳건히 하게 되는 계기가 될 것으로 사료된다.

참고문헌

[1] S. J. Jung, and K. Sung, "Trend analysis and Classification of Linux distributions," *The Journal of Digital Contents Society*, Vol. 18, No. 2, pp. 357-363, April 2017.

[2] S. J. Jung, and Y. M. Bae, *To Conquer the Linux Master First Class*. Seoul, Booksholic publishing, 2015.

[3] S. J. Jung, *Linux Master Second Class Complete Conquest*, Seoul, Booksholic publishing, 2017.

[4] Wikipedia, XFS [Internet]. Available: https://en.wikipedia.org/wiki/XFS/.

[5] XFS Project, Primary XFS Documentation Papers [Internet]. Available: http://xfs.org/index.php/XFS_Papers_and_Documentation

[6] Red Hat, Performance Tuning Guide Papers [Internet]. Available: https://access.redhat.com/documentation/ko-kr/red_hat_enterprise_linux/6/html/performance_tuning_guide/s-storage-xfs

[7] Zdnet, Journaling File System XFS [Internet]. Available: http://www.zdnet.co.kr/news/news_view.asp?artice_id=00000039135642

[8] Wikipedia, B-tree [Internet]. Available: https://en.wikipedia.

org/wiki/B-tree
[9] Wikipedia, B+tree [Internet]. Available: https://en.wikipedia.org/wiki/B%2B_tree
[10] Red Hat, Red Hat Enterprise Linux technology capabilities and limits [Internet]. Available: https://access.redhat.com/articles/rhel-limits?tour=7

성경(Kyung Sung)

1993년 경희대학교 대학원(공학석사)
2003년 한남대학교 대학원(공학박사)

1994년～2004년: 동해대학교 컴퓨터공학과 교수
2004년～2014년: 목원대학교 컴퓨터교육학과 교수
2006년～현 재: 목원대학교 융합컴퓨터미디어학부 교수
※관심분야 : 정보보호 및 정보관리, 증강현실, 빅데이터, 컴퓨터네트워크, 신경회로망, 컴퓨터교육