대용량 파일의 빠른 처리를 지원하는 파일 시스템에 관한 연구

박 기 현*

A study on the file system that supports fast processing of large files

Ki-Hyun Park*

요 약

본 연구는 대용량의 파일 크기를 지원하고, 파일의 내용이 자주 변경되지 않는 환경을 가정하여 파일의 처리 속도 향상에 관하여 다룬다. 기존 파일 시스템은 다양한 크기의 파일을 지원하기 위하여 사용자 데이터가 저장된 디스크 블록의 접근 방식을 다양화시키고 있다. 이와 같은 다양한 접근 방식은 하나의 파일에 모두 적용된다. 제안된 파일 시스템은 파일 크기에 따라 여러 접근 방식 중의 하나만을 채택하여 파일의 접근 속도를 향상하는 파일 시스템 구조를 지원한다. 이론적인 연구 결과는 디스크 블록에 대한 접근 횟수를 줄여서 성능 향상이 가능함을 보이고 있다.

Abstract

This study supports large file sizes, and deals with file throughput improvements assuming that the contents of the files do not change frequently. Traditional file systems are diversifying the approach of disk blocks where user data is stored to support files of varying sizes. These different approaches apply to all one file. The proposed file system supports a file system structure that increases the speed of access to files by adopting only one of several approaches depending on the file size. Reduce the number of access to theoretical studies have shown that disk blocks, and performance improvement is seen as possible.

* 위덕대학교 경찰정보보안학과 (Department of Police and Cyber Security Uiduk University)

Ⅰ. 서 론

컴퓨터 파일 시스템은 사용자의 데이터를 저장하여 보관하고, 저장된 데이터를 사용자가 다시 읽어서 사용할 수 있도록 해준다[1][8][10][12]. 컴퓨터가 개발되었던 초기 환경은 물론이고, 영상 등의 멀티미디어 데이터가 사용되기 전까지는 파일의 용량이 크지 않았었다. 텍스트 데이터 위주로 사용하던 시기를 가정하면 파일의 크기가 몇 K 바이트에 머무는 경우가 대부분이고, 큰 파일의 경우도 M 바이트 단위를 크게 넘는 경우가 흔하지 않았다. 현재의 컴퓨터 환경은 텍스트 기반의 파일보다는 영상 위주의 환경으로 발전하였다. 특히 유투브, 넥플릭스 등과 같은 영상 지원 서비스가 보편적으로 이루어지면서 몇백 M 단위의 파일뿐만이 아니라, G 단위의 대용량 파일의 사용도 보편화되어 있다[6][7][9][11].

유닉스와 윈도우 운영체제는 텍스트 기반으로 한 파일 시스템 환경에서 개발이 되었으며, 파일 시스템의 기본 구조는 당시의 소용량 파일의 처리에 빠른 속도를 지원하는 구조를 유지하고 있다[8][10][12][13]. 이러한 파일 시스템의 구조들은 소용량 파일에 대해서는 빠른 속도를 지원하지만, 파일의 용량이 커지면서 파일 처리의 속도도 증가하게 된다. 이와 같은 특징을 보이는 이유는 기본적으로 파일이 보관된 위치 정보를 몇 단계의 간접 처리 방식으로 이루어지느냐에 영향을 받는다. 소용량 파일의 경우는 간접 처리 과정이 생략되기 때문에 가장 빠르게 처리되지만, 파일의 용량이 커질수록 추가적인 처리가 필요한 간접 접근의 과정이 추가되면서 속도가 느려지게 된다.

본 연구에서는 대용량 파일을 가정하여 추가적인 간접 처리 과정을 줄임으로써 파일의 접근 속도를 향상시키는 방안에 대하여 다루고, 그러한 구조를 지원하는 새로운 파일 시스템의 구조에 대하여 다루고 있다. 기존 연구[2][3]는 하나의 파일에 여러 직간접 처리를 혼용하여 지원하지만, 본 연구는 하나의 파일에 하나의 직간접 처리를 적용하여 지원한다. 새로운 파일 시스템의 구조는 명백히 처리 속도를 향상시키지만, 소용량 파일의 경우는 오히려 처리 속도가 느려지는 단점도 존재할 수 있다. 연구의 목적이 대용량 파일을 전제로 하고 있음을 밝히며, 소용량 파일의 처리 속도의 저하 문제점은 파일 별로 소용량과 대용량 파일을 구분하는 형식으로 구조를 확장하여 해결하였다.

본 논문의 구성은 먼저 기존 파일 시스템에서 파일 데이터에 접근하는 원리와 성능에 대하여 다루고, 이어서 새로운 파일 시스템에서 파일 데이터에 접근하는 원리와 성능에 대하여 다룬다. 마지막으로 두 가지 방식의 성능에 대한 비교 분석에 대하여 다룬다.

Ⅱ. 연구 방법

1. 기존 파일 시스템의 분석

기존 파일 시스템에서 파일 크기에 따른 처리 방식의 분석은 유닉스 시스템의 구조를 기반으로 한다. 유닉스 운영체제는 하드 디스크에 분산되어 보관된 파일의 위치 정보를 보관하기 위하여 I-node라는 자료 구조를 지원한다[1][4][5][8][10][12].

I-node 구조에는 디스크의 블록 정보를 보관하기 위하여 13개의 필드를 지원하고, 그림 1은 이를 설명한다.[8][10] 그림에서 1번부터 10번까지에 보관된 주소는 파일 데이터가 보관된 디스크 주소이다. 사용자 데이터가 보관된 디스크 블록에 추가적인 처리 없이 바로 접근할 수 있다. 11번에 보관된 주소는 사용자 데이터에 바로 접근할 수 없다. 대신에 11번 주소가 가리키는 디스크 블록을 따라가면 그곳에 다시 주소들이 보관되어 있다. 그리고 이 주소들을 따라가면 사용자 데이터가 보관된 디스크 블록이 된다. 1번의 간접 접근 방식이다.

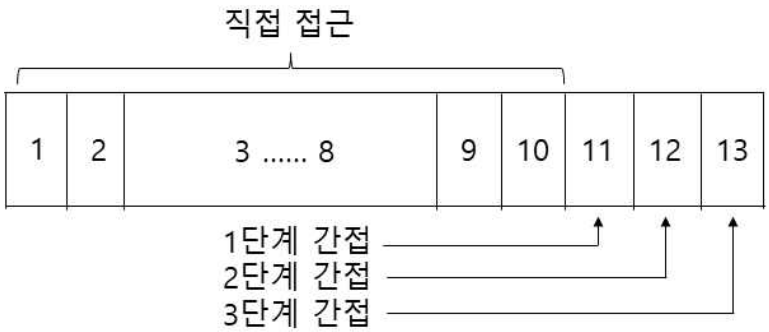

그림 1. I-node 구조에서 디스크 블록 주소

12번의 주소는 이러한 과정을 한번 더해서 2번의 간접 접근 방식을 사용하고, 13번의 주소는 이러한 과정을 한번 더 해서 3번의 간접 접근 방식을 사용한다. 이와 같이 간접 접근 방식을 사용하는 이유는 더 많은 사용자 데이터를 연결하는 장점을 제공한다. 반면에 간접 접근이 많아지면 사용자 데이터가 보관된 디스크 블록에 접근하는데 더 많은 절차가 필요하기 때문에 속도가 느려진다. 기존의 선행 연구[2][3]는 한 파일에 대하여 여러 접근 방식을 혼용하여 접근하고 그에 따른 성능 분석을 다루었으나, 본 연구에서는 한 파일에 대하여 하나의 접근 방식만을 적용하여 성능 분석을 다룬다.

파일 시스템은 4K 파일 시스템을 가정하고, 따라서 디스크 1 블록의 크기는 4K가 된다. 한 개의 디스크 블록에 저장할 수 있는 주소의 개수는 1024(4K / 주소 4바이트)개이다. 한 개의 디스크 블록을 읽는데 필요한 시간인 처리 속도는 T라고 가정한다.

먼저 직접 접근 방식에서 저장할 수 있는 데이터의 크기와 접근 속도에 대하여 살펴본다. 사용자 데이터를 가리키는 주소의 개수는 10개이기 때문에 사용자 데이터의 크기와 이 데이터의 처리에 필요한 속도는 다음과 같다.

직접 접근에서 데이터 크기 = 10 * 4K
직접 접근에서 처리 속도 = T

1단계 간접 방식에서는 사용자 데이터를 가리키는 주소의 개수가 1개이고, 1번의 간접

처리가 이루어지기 때문에 사용자 데이터의 크기와 이 데이터의 처리에 필요한 속도는 다음과 같다.

1단계 간접에서 데이터 크기 = 1024 * 4K
1단계 간접에서 처리 속도 = 2T

2단계 간접 방식에서는 사용자 데이터를 가리키는 주소의 개수가 1개이고, 2번의 간접 처리가 이루어지기 때문에 사용자 데이터 크기와 이 데이터의 처리에 필요한 속도는 다음과 같다.

2단계 간접에서 데이터 크기 = 1024 * 1024 * 4K
2단계 간접에서 처리 속도 = 3T

3단계 간접 방식에서는 사용자 데이터를 가리키는 주소의 개수가 1개이고, 3번의 간접 처리가 이루어지기 때문에 사용자 데이터 크기와 이 데이터의 처리에 필요한 속도는 다음과 같다.

3단계 간접에서 데이터 크기 = 1024 * 1024 * 1024 * 4K
3단계 간접에서 처리 속도 = 4T

2. 새로운 파일 시스템의 구조

본 연구에서 제안하는 파일 시스템의 구조는 파일의 크기에 따라 I-node에서 사용자 데이터가 보관된 디스크 블록까지의 연결 구조를 다르게 지원한다[2][3]. 즉, 현재의 파일 시스템은 사용자 데이터 크기와 관계없이 모든 파일이 여러 방식으로 혼합하는 연결 구조를 지원하지만, 본 연구가 제안하는 파일 시스템은 파일마다 그 크기에 따라 한 방식의 구조를 지원한다. 따라서 파일 크기가 다르면 지원하는 방식도 다를 수 있다.

먼저 크기가 가장 작은 용량의 파일을 지원하는 구조는 그림 2의 zero-type이다. 이 구조는 I-node가 지원하는 13개의 디스크 주소 필드를 모두 직접 접근의 용도로 사용한다. 이 방식으로 지원할 수 있는 사용자 데이터 크기와 이 데이터의 처리에 필요한 속도는 다음과 같다.

zero-type에서 지원하는 데이터 크기 = 13 * 4K
zero-type에서 처리 속도 = T

그림 2. zero-type

zero-type보다 큰 용량의 파일을 지원하는 구조는 그림 3의 one-type이다. 이 구조는 I-node가 지원하는 13개의 디스크 주소 필드를 모두 1단계 간접 접근의 용도로 사용한다. 이 방식으로 지원할 수 있는 사용자 데이터 크기와 이 데이터의 처리에 필요한 속도는 다음과 같다.

zero-type에서 지원하는 데이터 크기 = 13 * 1024 * 4K

zero-type에서 처리 속도 = 2T

그림 3. one-type

one-type보다 큰 용량의 파일을 지원하는 구조는 two-type이다. 이 구조는 그림 2와 3에서 I-node가 지원하는 13개의 디스크 주소 필드를 모두 2단계 간접 접근의 용도로 사용한다. 이 방식으로 지원할 수 있는 사용자 데이터 크기와 이 데이터의 처리에 필요한 속도는 다음과 같다.

zero-type에서 지원하는 데이터 크기 = 13 * 1024 * 1024 * 4K

zero-type에서 처리 속도 = 3T

two-type보다 큰 용량의 파일을 지원하는 구조는 three-type이다. 이 구조는 그림 2와 3에서 I-node가 지원하는 13개의 디스크 주소 필드를 모두 3단계 간접 접근의 용도로 사용한다. 이 방식으로 지원할 수 있는 사용자 데이터 크기와 이 데이터의 처리에 필요한 속도는 다음과 같다.

zero-type에서 지원하는 데이터 크기 = 13 * 1024 * 1024 * 1024 * 4K

zero-type에서 처리 속도 = 4T

우리가 제안하는 파일 시스템 구조는 파일별로 그 크기에 따라 다른 속성 정보를 갖는다. 즉, 가장 작은 파일부터 파일의 크기에 따라 zero-type, one-type, two-type, three-type의 서로 다른 정보를 가진다. 이 속성 정보에 따라 I-node의 디스크 주소 블록의 정보를 다르게 해석하여 처리하고, 그에 따른 하부 연결 구조도 다르게 처리된다는 점이다. 제안된 파일 시스템은 파일의 크기가 빈번하게 변경되는 환경에서는 파일 속성의 변경에 따른 연결 구조의 변경이 필요하다. 따라서 이러한 환경에서는 기존 파일 시스템보다 비효율적일 수 있다.

현재의 네트워크 환경에서 지원하는 대부분의 대용량 파일들은 영상정보인 경우가 많고, 이러한 파일들은 성격상 파일의 크기가 변경되는 경우가 거의 없다. 이와 같이 파일의 내용이 자주 변경되지 않고, 따라서 파일의 크기가 자주 변하지 않는 환경에서 본 연구가 제안하는 파일 시스템 구조는 성능면에서 장점을 지닌다.

3. 성능 분석

먼저 zero-type이 지원하는 파일 크기에 대한 성능 비교는 그림 4와 같다. 기존 파일 시스템은 11번 블록부터 간접 처리가 이루어지기 때문에 11~13번 블록에서 간접 블록을 읽는 시간 T가 추가되어 2T임을 알 수 있다. 그러나 제안된 파일 시스템은 T 시간만 소요된다.

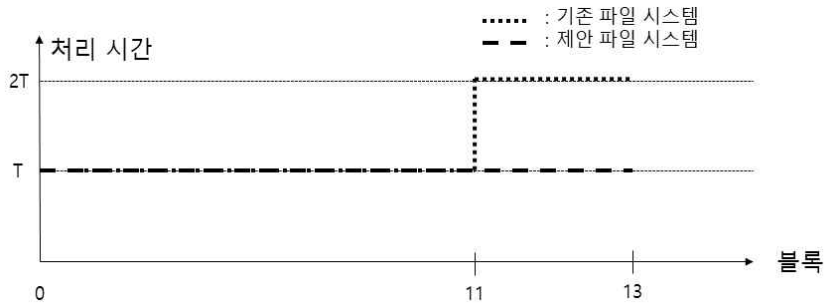

그림 4. zero-type 성능 비교

one-type이 지원하는 파일 크기에 대한 성능 비교는 그림 5와 같다. 기존 파일 시스템은 파일 크기가 (10 + 1024) * 4K까지는 1번의 간접 연결이지만 이후부터는 2번의 간접 연결이 이루어진다. 따라서 제안된 파일 시스템보다 디스크 블록의 접근 속도가 T만큼 늘어나게 된다.

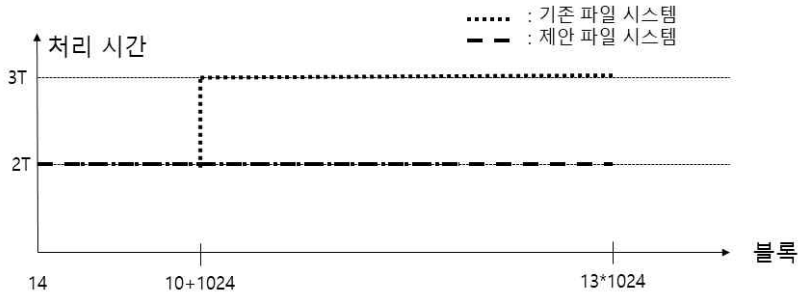

그림 5. one-type 성능 비교

two-type이 지원하는 파일 크기에 대한 성능 비교는 그림 6과 같다. 기존 파일 시스템은 파일 크기가 (10 + 1024 + 1024 * 1024) * 4K까지는 2번의 간접 연결이지만 이후부터는 3번의 간접 연결이 이루어진다. 따라서 제안된 파일 시스템보다 디스크 블록의 접근 속도가 T만큼 늘어나게 된다.

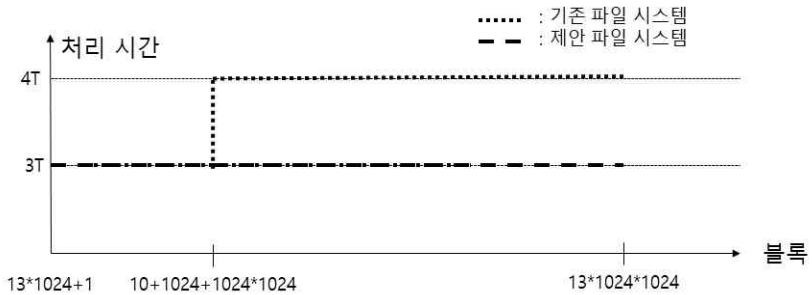

그림 6. two-type 성능 비교

마지막으로 three-type이 지원하는 파일 크기에 대한 성능은 기존 파일 시스템과 동일하게 4T이다. 대신에 기존 파일 시스템이 지원하는 파일의 최대 크기는 (10 + 1024 + 1024 * 1024 + 1024 * 1024 * 1024) *4K 이지만, 제안 파일 시스템에서의 최대 크기는 13 * 1024 * 1024 * 1024 * 4K로 증가한다.

III. 결 론

본 연구는 대용량 파일에 대한 접근 속도를 향상시키기 위한 파일 시스템 구조에 다루고 있다. 본 연구에서 제안하는 파일 시스템의 구조는 I-node 구조로부터 파일 데이터가 저장된 하드 디스크에 접근하여 사용자 데이터를 읽는데 걸리는 시간에 초점을 두었다.

제안된 파일 시스템에서 개별 파일에는 서로 다른 속성 정보가 추가된다. 파일의 크기에 따라 서로 다른 속성 정보 (zero, one, two, three-type)를 가지며, 이 정보에 따라 디스크

블록에 접근하는 방식도 다르게 처리되었다. 본 연구에서는 하드 디스크에 접근하는 간접 접근의 처리 방식에 대해서는 다루지 않았다. 기존 파일 시스템에 그에 관한 처리 구조와 방식은 모두 존재하기 때문이다.

제안된 파일 시스템 구조는 파일의 내용이 자주 변경되지 않는 경우를 가정하였다. 파일의 크기가 서로 다른 type의 크기로 변경되면 I-node에서 디스크 블록까지의 연결 구조가 바뀌기 때문에 이 작업을 하는데 추가적인 시간이 걸린다. 제안된 파일 시스템 구조는 파일의 크기가 자주 바뀌지 않는 구조를 가정하였지만, 각 type에서 지원하는 파일 크기의 범위 내에서는 파일 크기가 바뀌어도 추가적인 작업 시간을 필요로 하지 않는다. type이 변경될 정도로 파일의 크기가 변경되는 경우에는 파일을 복사하는 정도의 작업이 소요된다.

본 파일 시스템의 구현 알고리즘은 이미 기존 파일 시스템에 구현되어 있으므로 어렵지 않게 실현할 수 있다. 현재의 네트워크 환경과 같이 파일의 크기가 자주 변경되지 않는 대용량 멀티미디어 환경에서 파일 처리 속도를 향상할 것으로 기대한다.

참고 문헌

[1] 이종원, “유닉스 시스템 관리”, 한빛미디어, 2009
[2] 박기현, “파일 속도 접근 향상을 지원하는 파일 시스템 구조”, 위덕대학교 산업기술연구소 제18권 제1호, 2014
[3] 박기현, “파일 접근 속도 향상을 지원하는 파일 시스템의 성능 분석”, 위덕대학교 산업기술 연구소 제19권 제1호, 2015
[4] Prabhat K. Andleigh, "UNIX System Architecture", Prentice Hall, 1990
[5] Naghibzadeh, "Operating System - Conecpts and Techniques", Iuniverse Inc, 2005
[6] Andrew S. Tanenbaum, "Computer Network, 4Ed", Prentice Hall, 2002
[7] 박기현, “데이터통신과 컴퓨터 네트워크”, 한빛미디어, 2016
[8] William Stallings 저, 전광일 역, “운영체제 내부구조 및 설계원리”, 그린출판사, 2009
[9] 나종화, 강순주, 윤용익, 박윤용, 은성배, 김홍남, “Embedded System Programming", ETRI, 2003
[10] 구현회, “운영체제 - 그림으로 배우는 원리와 구조”, 한빛미디어, 2007
[11] W. Richard Stevens, "UNIX Network Programming", Prentice Hall Software Series, 1994.
[12] 김용석, Operating System", Scitech 미디어, 2007.
[13] 윤소정,이정원,“유닉스 이론과 실습”, 한빛미디어