임베디드 보드에 최적화된 Network Access Storage Software에 대한 연구

이정수, 이상호
한국산업기술대학교 컴퓨터공학과
e-mail:rsyosi@kpu.ac.kr

A Study on Network Access Storage Software on embedded board

Jung-Soo Lee, Sang-Ho Lee
Dept of Computer Engineering, Korea Polytechnic University

요 약

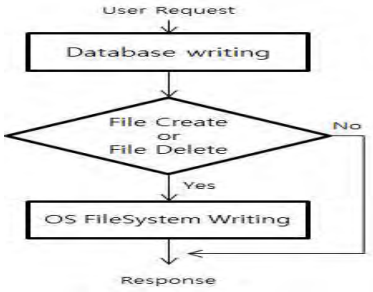
네트워크 속도가 발전되고 하드디스크의 비용이 감소함에 따라 NAS(Network Access Storage) 시장이 빠르게 발전하고 있다. 현재 상용화된 많은 NAS는 고성능의 하드웨어를 기반으로 하기 때문에 높은 가격대를 형성하고 있다. 본 논문에서는 NAS Management Software를 임베디드 보드에 최적화하기 위해 어떠한 요소들이 NAS의 성능에 영향을 미치는지 분석한다. 이후 운영체제에서 관리하는 파일 시스템을 데이터베이스 또는 세션에 옮기는 방법을 제안한다. 마지막으로 기존의 시스템과 비교를 통해 우수성을 보인다.

1. 서론

최근 지속적인 네트워크 인프라의 발전과 하드디스크의 단가 인하로 인해 NAS(Network Access Storage)의 시장은 2010년 전후로 폭발적인 성장을 하고 있다. 클라우드 스토리지의 종류엔 웹 기반 파일서버, 퍼블릭 클라우드 서비스 등이 있지만 보안 문제, 정부의 검열 등으로 인해 대기업 및 중소기업은 전산시스템 구축 또는 파일 백업/보안을 위해 NAS를 활용하고 있다. 하지만 고가의 하드웨어를 사용함으로써 NAS의 가격은 많은 일반 그룹 사용자들과 중소기업이 이용하기에 버거운 가격대를 형성하고 있다. 이에 본 논문은 저가의 미니보드에 NAS를 최적하기 위한 방법과 그 결과를 서술한다.

2. 관련 시스템

NAS는 하드디스크를 포함한 하드웨어와 파일을 송수신하고 관리하기 위한 NAS 관리 소프트웨어로 이루어져 있다. 기존에 출시된 NAS의 경우 우선 FTP 또는 WebDav 등과 같은 파일 전송 프로토콜을 사용해 파일을 송수신한 후 파일 연산을 시작하고 마지막으로 데이터베이스에 로그를 기록하는 형태를 취하고 있다. 파일을 전송하는 경우를 예로 든다면 파일 전송 프로토콜을 이용해 파일을 수신하는 시간 T1과 파일을 물리적 디스크에 쓰는 시간 T2, 로그를 데이터베이스에 기록하는 시간 T3로 이루어져 있다.



(그림 1) 3.2절에서 제안한 NAS 순서도

T1의 경우 네트워크 회선과 파일 전송 프로토콜에 가장 많은 영향을 받기 때문에 NAS 자체보다는 외부의 영향을 가장 많이 받는다. 하지만 T2와 T3는 NAS의 하드웨어(임베디드 보드) 성능에 가장 큰 영향을 받는다. 본 논문에서는 임베디드 보드에서 NAS 관리 프로그램을 최적화시키기 위해서는 T2와 T3의 시간을 줄이기 위한 방법을 제안한다. 본론 3에서는 T2와 T3의 시간을 줄이기 위한 방안을 서술하며 본론 4에서 직접적인 성능비교를 통해 우월성을 입증한다.

3. 파일연산에 대한 연구

본론 2에서 언급한 T2와 T3의 시간을 줄이기 위해서는 하드웨어의 성능을 높이는 방법이 존재한다. 하지만 이는 본 논문이 의도하는 바가 아니므로 위 방법을 제외하고 T2를 조건적으로 최소화하는 방안을 제안한다. 물리적

인 디스크의 접근을 유발하는 [1]파일연산의 종류로는 파일생성, 파일읽기, 파일수정, 파일삭제가 있다. 여기서 파일 수정 연산은 파일 이동, 파일 복사, 파일명 수정, 파일 잘라넣기 등이 있다. 위 연산이 실행되고 나면 데이터베이스의 쓰기작업이 시작되므로 불필요한 파일 연산은 실행하지 않는다면 T2의 시간이 0이 되므로 전체적인 작업시간을 크게 줄일 수 있다. 3.1절에서는 불필요한 파일 연산을 수행하지 않도록 하는 방법을 서술하며 3.2절에서는 3.1절의 방법을 사용하기 위해 해야 하는 작업에 대해 서술한다.

3.1 불필요한 파일연산 제거

파일시스템을 유지하기 위해서는 [2]운영체제에 파일구조를 담은 트리를 유지해야 하는데 이를 위해서 NAS 소프트웨어는 파일연산을 실행해야 한다. 만약 파일 시스템을 유지할 필요가 없다면 T2의 시간은 0에 수렴하므로 NAS의 속도를 크게 올릴 수 있다. 하지만 파일 생성 및 파일 삭제를 하지 않는다면 NAS의 기능을 수행할 수 없기 때문에 본 논문에서는 파일 생성 및 삭제 연산을 제외한 나머지 연산을 수행하지 않는 방법을 제안한다.

3.2 데이터베이스 파일시스템 구축 제안

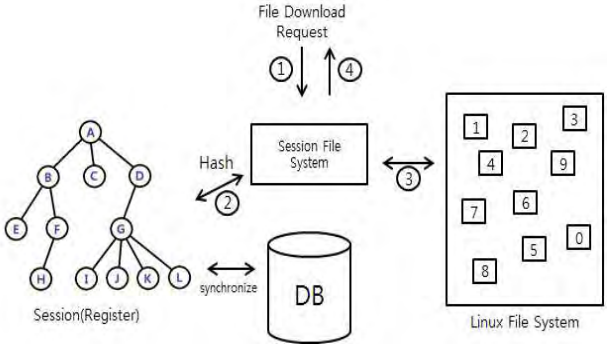
파일 구조를 저장한 트리를 [3]데이터베이스에 저장한다면 본론 3.1절에서 언급한 파일 생성 및 삭제 연산을 제외한 나머지 연산은 수행할 필요가 없고 물리적 디스크에 접근하지 않아도 된다. 그림1은 이 절에서 제안한 방법을 사용하여 나타낸 순서도다. 위 방법을 사용하면 대부분의 파일 연산을 수행할 때 물리적 디스크에 접근할 필요가 없으므로 T1과 T3만으로 작업을 완료할 수 있다. 하지만 데이터베이스에 파일 시스템을 구축해야 하는 작업을 해야 한다. SQL을 사용하여 파일에 대한 정보의 레코드를 수정하는 작업은 하드디스크에 접근하여 파일연산을 하는 속도에 비해 매우 빠르므로 속도에 있어서 많은 이점을 가진다.

3.3 해쉬 구조의 데이터베이스 파일시스템 구조

데이터베이스에 파일 시스템을 구축했을 경우 파일 삭제 및 파일 다운로드와 같이 접근이 필요한 경우 파일을 찾을 수 없는 현상이 발생한다. 이 문제를 해결하기 위한 방법으로 본 논문의 저자는 해쉬 형태의 구조를 제안한다. 파일이 생성되었을 때 서버에서는 각 파일의 이름을 데이터베이스에 저장한 후 이에 해당하는 고유한 일련번호를 부여한다. 이후 이 파일에 접근이 필요한 경우 데이터베이스에 일련번호를 조회하여 파일명과 파일의 위치를 찾을 수 있다. 이 방법을 사용하면 실재 파일 시스템 내에서는 모든 파일명과 파일 위치를 은닉할 수 있다. 일반적인 NAS의 파일시스템은 하드디스크를 분실 또는 도난당했을 때 [2]파일 도난에 매우 취약하다. 하지만 본 저자가 제안한 방법을 사용한다면 보안기능이 없는 하드디스크를 사용하더라도 보안에 강점을 가질 수 있다.

3.4 세션 파일시스템 구축 제안

사용자가 NAS에 접근하기 위한 방법으로는 클라이언트 프로그램 또는 웹 애플리케이션이 주로 이용된다. 본 절에서는 3.2절에서 제안한 방법보다 더 빠르고 오버헤드가 적은 방법을 제안한다. 3.2절에서 제안한 방법은 기존의 방법에 비해 속도단축을 얻을 수 있지만 데이터베이스에 더 많이 접근해야 한다는 단점이 존재한다. 이를 위해 [3]데이터베이스 풀을 사용하는 방법이 있다. 하지만 이보다 더 좋은 방법은 파일구조를 담은 트리를 레지스터 또는 세션에 유지하는 것이다.



(그림 2) 세션 파일 시스템에서 파일 다운로드 과정

만약 클라이언트 프로그램이 JSP로 이루어진 웹 애플리케이션이라면 파일 시스템을 저장한 트리를 Context Session에 를 저장한다면 데이터베이스에 접근하는 방법보다 더욱 효율적이다. 그림 2는 세션 파일 시스템의 작동 구조를 도식화한 그림이다. 3.3절과 3.4절에서 제안한 방법으로 구축된 시스템에서 파일 다운로드를 할 경우 크게 4가지 과정을 가진다. 먼저 파일 다운로드 요청이 오면 세션 또는 레지스터에 파일 구조 트리를 탐색하여 해당 파일명에 해당하는 키 값을 가져온다. 이후 실재 파일 시스템에 접근하여 해당 키 값과 동일한 파일을 찾아낸 후 파일 다운로드를 진행한다.

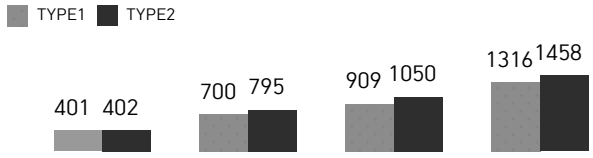
4. 성능평가

3.2절에서 제안한 데이터베이스 파일 시스템 구축의 이점을 검증하기 위해 실험환경을 구축하고 millisecond 단위로 성능평가를 진행하였다. 본 논문에서의 실험 환경은 다음과 같다.

Embeded board	CubieTruck2
Operating System	Devian Linux
Programming language	JAVA
File I/O Method	JAVA NIO 2.0
Database	Mysql 5.5
UI Type	Web UI

<표 1> 전체적인 실험환경

2장에서 언급하였듯이 NAS의 작업을 분류하면 파일을 전송하는 시간 T1, 파일 시스템에 접근하여 파일연산을 하는 시간 T2, 데이터베이스에 로그를 기록하는 시간 T3가 있다. T1은 네트워크 인프라의 영향을 받으므로 이를 제외하고 T2와 T3의 시간만을 측정하였다. 아래의 그림 3은 데이터베이스 파일 시스템을 구축한 TYPE2와 일반적인 NAS 시스템 TYPE1을 잘라넣기 연산을 수행했을 때 시간을 나타낸다. 가로축은 각각 1MB의 크기를 가지는 파일의 개수이며 세로축은 이 파일들을 폴더 내에 저장해 놓고 다른 폴더로 파일 연산을 수행했을 때의 걸리는 소요시간(단위:MILLISECOND)을 나타낸다.



본 논문의 저자가 구현한 데이터베이스 파일 시스템은 잘라넣기 연산을 수행하기 위해 먼저 폴더 용량을 재귀를 사용해 구한다. 이후 파일을 이동 및 삭제하며 결과를 데이터베이스에 저장해주었다. 그림 3는 파일의 개수와 용량이 증가할수록 소요시간은 반비례하는 것을 볼 수 있다. 따라서 NAS에 많은 파일을 보관할수록 본 논문이 제안한 방법은 더 많은 이점을 보이는 것을 볼 수 있다.

4.1 부수적인 이점

4.1절에서는 속도의 이점 외에 다른 이점을 서술한다. 본 논문에서 제시한 방법의 핵심은 불필요한 물리적 디스크의 접근을 하지 않는 것에 있다. 이 방법은 하드디스크의 접근을 최소화하며 NAS 수명을 연장하는 효과를 발휘한다. 또한 전력소요가 줄어드는 부수적인 효과를 얻을 수 있다. 또한 3.3절에서 언급했듯이 기존의 NAS에 비해 파일명과 파일구조를 은닉하는 효과가 있다.

5. 결론

본 논문에서는 기존의 Network Access Storage 제품들의 동작 구조를 분석하였고 임베디드 보드 성능에 가장 영향을 미치는 요인을 분석하였다. 이 중에서 파일 연산 수행하는 과정을 최소화하기 위해 불필요한 파일 연산을 제거하고 데이터베이스에 파일 시스템을 구축하는 방법뿐만 아니라 세션과 레지스터를 사용하여 속도를 증가시키는 방법을 제안하였다. 마지막으로 본 논문이 제시한 방법과 기존의 방법을 비교함으로 우수성을 보였다.

본 논문에서 제안한 방법은 기존의 NAS의 성능을 최적화하여 결과적으로 저성능의 임베디드 보드에 기반한 NAS 개발에 많은 도움을 줄 것이라 생각된다.

참고문헌

[1] Anghel Leonard “PRO JAVA7 NIO.2, Apress”, 2012

[2] William Stallings “Operating Systems : Internals and Design Principles” 6th, Prentice Hall, 2009

[3] Kevin Mukhar “Beginning Java Databases” Wrox, 2002

[4] Yoonmyung Kim “JSP&Servlet stimulating brain” HanbitMedia, 2011