임베디드 리눅스 기반의 네트워크 저장장치의 구현 및 성능 평가

이현석*, 강용혁*, 손재기**, 이형수**, 민수영**, 박창원**, 엄영익*
*성균관대학교 전기전자 및 컴퓨터공학부
**전자부품연구원 정보 시스템 연구센터
email : {hyunsuk, yhkang1}@ece.skku.ac.kr
{jgson, hslee, minsy, parkcw}@keti.re.kr
yieom@ece.skku.ac.kr

Implementation and Performance Evaluation of Embedded Linux-based Network Attached Storage

Hyun-suk Lee*, Yong-Hyeog Kang*
Jae Gi Son**, Hyung Su Lee**, Soo Young Min**, Chang Won Park**
Young Ik Eom*
*School of Electrical and Computer Engineering, SungKyunKwan University
**IT System Research Center, Korea Electronics Technology Institute

요 약

네트워크의 고속화로 IT산업이 발달함에 따라 정보를 저장할 대용량의 저장장치가 필요하게 되었다. 그러나 디스크 드라이브는 느린 입출력 속도와 오류로 인한 성능 저하 및 정보 손실이 있을 수 있고 저장장치를 운영하는 서버에 부하를 증가시켜 서버의 성능을 저하시킬 수 있기 때문에 보다 안정적이고 성능이 좋은 네트워크 저장장치의 필요성이 증가하게 되었다. 본 논문은 리눅스 커널 축약(shrinking)을 통하여 구현한 임베디드 리눅스 기반의 네트워크 저장장치에 관한 것이다. 네트워크 저장장치의 운영체제로서 리눅스를 사용함으로써 운영체제의 비용을 줄이고, 기존 리눅스 커널보다 축약된 리눅스 커널을 사용함으로써 메모리의 효율성 및 성능이 향상된 네트워크 저장장치 시스템을 구축할 수 있다.

1. 서론

정보화 사회와 더불어 텍스트기반의 정보뿐만 아니라 이미지, 영상, 음성, 오디오와 같은 대용량의 데이터가 폭발적으로 늘어나고 있다. 또한 인터넷의 발전으로 네트워크 상에서의 통신과 게임 그리고 자료공유 등 여러 종류의 수많은 데이터의 전송이 빈번해져 디스크의 사용량이 증가하게 되었고 대용량의 정보를 저장할 수 있는 저장장치가 필요하게 되었다. 최근에는 수십 기가 바이트의 용량을 갖는 디스크 드라이브가 등장하여 대용량의 정보를 저장할 수 있게 되었다. 그러나 대용량의 디스크 드라이브는 저장하는 용량이 커짐으로 인해 디스크 드라이브의 오류 발생률도 증가하여 저장된 데이터를 한꺼번에 잃어버릴 수 있고 저장장치를 운용하는 서버의 부하도 증가시키는 문제점이 발생한다. 따라서 안정적이고 신뢰성 있는 저장장치의 개발하는 것 외에도 서버의 성능을 향상시키는 네트워크 저장장치의 개발이 필요하다.

이러한 필요성 때문에 저장장치를 운영하는 서버로 범용 서버보다는 저장장치의 기능을 전담하는 서버를 사용하는 추세이다. 또한 네트워크 저장장치를 서비스하는 서버는 네트워크 상의 이 기종 클라이언트에게 투명하고 효율적인 파일 서비스를 제공하여야 한다[1]. 최근에 네트워크 저장장치 개발 방식에는 SAN(Storage Area Network) 방식과 NAS(Network Attached Storage) 방식이 있으며 이 두 가지 방식을 혼합한 방식도 개발되고 있다. 본 논문에서는 네트워크 저장장치로서 NAS 방식을 채택하였으며, 저장장치의 기능을 전담하는 전용 서버를 구축하기 위해 리눅스 커널을 축약(shrinking)한 임베디드 리눅스를 사용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대해서 기술하고 3장에서는 NAS를 위한 임베디드 리눅스에 대해서 기술한다. 그리고 4장에서는 임베디드 리눅스 기반의 NAS에서의 성능을 평가하며 마지막으로 5장에서 결론에 대해 기술한다.

2 관련연구

2.1 임베디드 리눅스 기반의 NAS

NAS(Network Attached Storage)란 파일 서버 기능이 내장된 네트워크 저장장치이다. 기존의 일반적인 서버는 파일 서비스와 응용 서비스를 같이 제공함에 반해 NAS는 응용 서비스를 제공하지 않는다. 즉, 이더넷과 같은 LAN 인터페이스를 통해 독립적으로 네트워크에 연결되어 응용 서버와 분리시켜 놓아 파일 서비스만 제공함으로써 파일 서버 기능만을 전담하는 저장장치 전용시스템인 것이다. 이를 통해 일반적인 서버에서 응용 서비스와 파일 서비스를 동시에 수행함으로써 발생할 수 있는 네트워크 병목현상을 줄일 수

있으며 응용 서비스를 위해 사용되던 CPU 점유가 파일 서비스를 위하여 사용되므로 보다 효율적이다.

NAS의 운영체제는 기존의 일반적인 서버의 운영체제와 달리 네트워크와 파일 오퍼레이션만을 중점으로 관리하면 된다. 따라서 NAS의 운영체제로 임베디드 운영체제를 사용할 수 있고 이를 통하여 파일 서버로서의 성능을 향상시킬 수 있다. 본 논문에서는 NAS의 운영체제로 임베디드 리눅스를 사용하였다. 임베디드 시스템을 위하여 기존의 상용 운영체제 대신에 리눅스를 선택한 이유는 리눅스는 오픈 소스 정책을 취하고 있으므로 개발할 하드웨어에 적합한 소스로 기존 소스의 수정이 용이하여 개발비용을 저렴화 할 수 있기 때문이다.

2.2 리눅스에서의 NFS

NFS(Network File System)란 원격 호스트 상의 파일을 사용자가 마치 자신의 로컬 파일에 엑세스하는 것처럼 사용할 수 있게 해주는 분산 파일시스템이다. NAS가 제공하는 파일 서비스를 받기 위해서는 파일 서버 프로토콜이 필요하며 본 논문에서는 파일 서버 프로토콜로 NFS를 채택하였다. 그 이유는 NFS가 현재 산업계에서 많이 사용되고 있을 뿐만 아니라 신뢰성이 높기 때문이다.

리눅스에서는 NFS 버전 2와 NFS 버전 3이 지원된다. NFS 버전 3은 리눅스 버전 2.2부터 지원되나 TCP 위에서 동작하는 NFS는 아직 지원되지 않고 있다. 리눅스에서 NFS서버는 커널 모드와 유저 모드에서 동작한다. 커널 모드에서 동작하는 NFS 서버는 클라이언트로부터의 요청을 서비스해주는 커널 프로세스이며 빠르고 용량이 크다. 반면 유저 모드에서 동작하는 NFS 서버는 커널 모드에서의 NFS 서버보다 느리다[2].

2.3 커널 축약(shrinking)

커널 축약은 커널 이미지 크기를 줄이고 축약된 커널을 통하여 커널이 사용하는 메모리 크기를 줄이는 것이다. 커널 축약의 장점으로는 불필요한 부분을 제거함으로써 효율적인 메모리 사용을 가능하게 한다는 점과 불필요한 연산을 배제하고 필요한 기능만을 수행하도록 함으로써 전체적인 시스템 속도를 개선할 수 있다는 점을 들 수 있다. 단점으로는 다른 시스템이나 응용 프로그램과의 호환성문제가 발생할 수 있다. 커널 축약의 방식으로는 직접 커널 소스 레벨에서 커널을 수정하는 법과 일반적인 커널 컴파일 방식이 있다. 그리고 일반적인 커널 컴파일 방식은 내장(built-in) 커널 방식과 모듈 지원 방식으로 나눌 수 있다.

내장 커널 방식이란 시스템에 맞게 필요한 기능을 커널에 포함시키는 것이다. 이 방식은 시스템에 높은 성능을 기대할 수는 있지만 기능 수정 및 새로운 기능 추가 시에는 커널을 다시 컴파일하여 테스트를 해야하므로 개발단계에서는 적합하지 않고 임베디드 시스템을 위한 커널로는 비교적 커널의 크기가 크다는 단점이 있다. 따라서 다음에 기술하는 모듈 지원 방식과 적절한 혼합으로 커널을 축약해야 한다. 모듈 지원 방식이란 필요한 기능을 커널 모듈로 만들어서 필요시에만 커널에 의하여 호출되어 사용되도록 만드는 것이다. 호출된 커널 모듈은 커널에 동적으로 결합되므로 작은 크기의 커널을 가능하게 한다. 또한 모듈 개발 시에 커널을 다시 컴파일하고 재부팅하여 테스트 할 필요없이 모듈만 빼내어 모듈만 컴파일 함으로써 커널 개발 및 오류수정을 용이하게 한다. 물론 커널 모듈과 관련하여 성능과 메모리 면에서 약간의 손해가 있을 수 있다. 이것은 커널이 모듈을 로드할 수 있도록 모듈이 제공해야 하는 약간의 코드가 있고, 별도의 자료구조가 메모리를 차지하기 때문이다. 또한 모듈이 커널에 의해 호출될 때 심볼 테이블에 저장되어있는 모듈에 커널이 포인터로 접근해야하므로 효율성이 조금 떨어진다. 그러나 이와 같은 성능 저하는 아주 작으며 더 높은 성능이 필요할 경우 모듈로 된 부분을 내장(buint-in) 커널 방식으로 컴파일하여 성능 저하를 줄일 수 있다[4].

3. NAS를 위한 임베디드 리눅스 구축

본 논문에서는 리눅스 버전 2.4를 사용하여 커널 축약 및 성능 평가가 이루어졌다. 물론 리눅스 버전 2.2나 그 이하의 커널 버전을 사용한다면 커널 축약 시에 좀 더 작은 사이즈로 만들 수 있겠지만 리눅스 버전 2.4는 기존의 버전에 비해 임베디드 시스템을 위해 많은 부분을 최적화 시켰고 기존의 모듈화 되지 않은 부분들을 모듈화 시켜 성능 향상이 이루어져 있다. 또한 엔터프라이즈 급 서버에서도 사용될 수 있으며 기술도입과 용이한 버그 패치 등의 장점을 활용할 수 있기 때문에 리눅스 버전 2.4를 사용하게 되었다.

커널 축약을 위해서 커널 소스 레벨에서 커널을 수정해 봤을 때 이전 버전과의 호환성을 위해 존재하는 부분과 응용에 따라 불필요한 부분들을 제거해 보았으나 각 부분에 대하여 10-40바이트 정도밖에 줄어들지 않았다. 이는 이미 리눅스 커널 소스가 최적화 되어있기 때문이다. 직접 소스의 수정을 통하여 줄어든 커널의 크기도 작을 뿐 아니라 호환성 문제를 생각해 볼 때 기존의 커널을 사용하는 것이 더 안정적이므로 커널 소스를 수정하는 방식 대신에 일반적인 커널 컴파일 방식을 통하여 커널 축약을 했다. 본 논문에서의 커널 축약 방식은 리눅스 커널이 지원하는 기능들을 내장 커널 방식과 모듈 지원 방식 그리고 지원하지 않는 기능들의 세부분으로 나눠 커널의 크기를 줄이는 것이다.

[표 1] 내장 커널 방식

	커널에서 지원하는 기능
내장 커널 방식	Virtual 파일시스템 /proc 파일시스템 /dev/pts filesystem for Unix98 PTYS Second extended 파일시스템
	IDE 방식의 하드디스크를 사용
	TCP/IP networking 방식 지원 Ethernet (10 of 100Mbit) 지원
	PCI 지원 ELF binaries 지원

[표 1]에서는 내장 커널 방식에 대하여 나타낸다. NAS를 위해 네트워크와 파일시스템 그리고 시스템의 특성에 맞게 PCI와 IDE를 지원하며 이는 커널이 운영되기 위하여 기본적으로 필요한 것들이다. 그리고 Virtual 파일시스템과 /dev/pts filesystem for Unix98 PTYS는 텔넷을 지원하기 위하여 내장 커널 방식으로 지원하였다. 그 이유는 이들은 모듈로 지원되지 않기 때문이다. [표 2]에서는 NFS 서버기능을 모듈로 지원하며 호환성을 높이기 위하여 각종 SCSI

드라이버와 각종 랜카드를 모듈로 지원하였다. 이는 컴파일 후 실제 커널 크기에는 영향을 끼치지 않고 커널 개발이 용이하며 이더넷 카드를 새로 바꿨을 때 이를 인식하는데 있어서 새로 커널을 컴파일할 필요가 없기 때문이다. 또한 어떠한 기능이 필요할 때는 그에 맞는 데몬을 인식시키기만 하면 된다. 즉, 어떤 모듈이 요구될 때는 커널에 더해졌다가 불필요하면 커널에서 제거되어 유연성이 높아지므로 모듈을 지원한다.

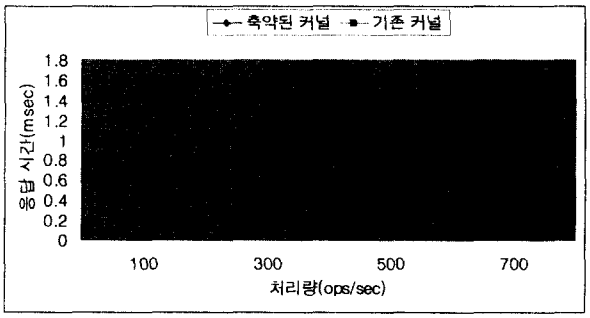
[표 2] 모듈 지원 방식

	커널에서 지원하는 기능
모듈 지원 방식	SCSI 드라이버 지원
	NFS 서버
	랜카드
	Multi-device(RAID와LVM) 지원

모듈로도 지원하지 않는 대표적인 것들에는 parallel port, floppy, cd-rom, USB, 주변장치, 멀티미디어 장치, core, console drivers 등이 있다. 이처럼 입출력 장치들은 대부분 지원하지 않았다. 그 이유는 serial port로 접속하여 모니터링 할 수 있을 뿐 아니라 telnet으로 접속하여 작업을 할 수 있기 때문이다. 물론 대부분 모듈로도 지원할 수 있는 것들이지만, NFS와 같은 꼭 필요한 기능만 모듈로 지원했다. 이러한 커널 축약방식으로 커널의 크기를 500KB이하로 줄였으며 NAS의 운영체제로 기존의 커널을 사용하는 대신 축약된 임베디드 리눅스를 사용하여 NAS의 성능이 향상된다.

4. 임베디드 리눅스 기반에서의 NAS의 성능 평가

성능 평가에서 사용한 서버의 사양은 펜티엄 Ⅲ 660Mhz 이며 128Mbyte 램을 사용하였다. 성능 평가 도구로는 SPECsfs2.0을 사용하였다. SPECsfs2.0은 NAS의 성능을 평가하는 도구로 버전 2와 버전 3의 NFS를 모두 지원하며 네트워크 프로토콜로 TCP와 UDP 모두 사용 가능하다[5]. 본 논문에서는 버전 2의 NFS와 UDP를 사용하여 SPECsfs2.0이 제공하는 성능 평가 항목 중에서 NAS에서 파일 서비스를 처리하는 응답속도 및 처리량을 측정하여 성능 평가를 수행하였다. 응답속도를 측정하기 위하여 클라이언트가 발생시키는 로드를 증가시켜서 NAS의 운영체제로 기존의 리눅스 커널을 사용하는 경우와 축약된 임베디드 리눅스 커널을 사용하는 경우로 나눠 각각 SPECsfs2.0을 사용하여 측정한 후 그 응답 속도 결과를 비교 정리하여 [그림 1]에 나타낸다.



[그림 1] 응답 시간 성능 평가 결과

[그림 1]에서 세로축은 응답시간을 의미하며 가로축에서 100, 300, 500, 700 은 주어진 로드량을 의미하고 가로축에 처리량도 같이 표시하였다. 이와같은 표기법은 SPECsfs2.0에서 사용하는 결과 표기법이며 본 논문에서는 NAS의 운영체제로 기존의 리눅스 커널을 사용하는 경우와 축약된 임베디드 리눅스 커널을 사용하는 경우를 비교 분석하기 위하여 위와 같은 그림으로 결과를 표현한 것이다. 로드량은 클라이언트가 서버에 부가한 초당 연산수(ops/sec)를 의미하며 처리량은 서버가 처리한 초당연산수를 말한다. 이때 주어진 로드량을 처리하는데 걸리는 시간이 응답시간이 된다. [그림 1]에서 로드량이 100인 지점에서는 기존 커널의 경우 응답시간이 1.1(msec)이며 임베디드 리눅스 커널의 경우는 1.0(msec)이다. 따라서 로드량이 적은 경우는 그 차이가 미묘하나 로드량이 높아지면서 응답시간도 차이가 남을 알 수 있다. 로드량이 700인 지점에서 축약한 임베디드 리눅스 커널을 사용한 것의 응답 속도가 1.1(msec)로 기존의 리눅스 커널의 응답 시간 1.7(msec)보다 0.6(msec) 정도 적게 나왔으며 이는 임베디드 리눅스를 사용한 경우 NAS의 파일 서비스 처리 속도가 빨라짐으로써 보다 효율적인 서비스를 제공할 수 있음을 의미하는 것이다.

5. 결론

본 논문은 임베디드 리눅스 기반에서의 네트워크 저장장치의 구현 및 성능 평가에 관한 것이다. 네트워크 저장장치의 운영체제로 임베디드 리눅스 커널을 사용함으로써 적은 비용으로 빠른 파일 서비스를 제공할 수 있다. 커널 축약을 통하여 구현한 임베디드 리눅스는 기본적으로 시스템에 지원해야 하는 부분만을 내장 커널 방식으로 했으며 NAS를 위하여 필요한 NFS는 모듈로 지원하여 모듈만 컴파일 가능하게 하였다. 성능 평가의 결과에서 보이듯이 NAS의 운영체제로 기존의 리눅스 커널을 사용하는 대신 임베디드 리눅스 커널을 사용함으로써 NFS 요청에 대한 응답시간이 줄어들었다. 이는 NAS의 성능이 향상되어 NAS에서의 파일 서비스를 보다 빠르게 받을 수 있음을 의미한다.

참고문헌

[1] 손재기, 이형수, 민수영, 박창원, "임베디드 리눅스를 이용한 네트웍 저장장치 연구 개발," 자료저장 시스템 연구회 워크샵, 2001.

[2] Tomas, "NFS/NIS(YP)," http://bukharin.hiof.no/fag/iad22099/innhold/lab/lab3/nfsnis_slides/text6.htm

[3] Brian Ward, "The Linux Kernel HOWTO," http://www.linuxdoc.org.

[4] David A Rusling, The Linux Kernel, http://www.linuxdoc.org.

[5] Standard Performance Evaluation Corporation(SPEC), "SFS2.0 Documentation Version 1.0," http://www.spec.org/osg/sfs97, 1997.

[6] Remy Card, Eric Dumas, and Frank Mevel, the Linux kernel book, 2ed., John Wiley & Sons, 1998.

[7] M Beck, H Bohme, M Dziadzka, and U Kunitz, Linux Kernel Internal, Addison Wesley, 2ed., 1998.

[8] Daniel P.Bovet and Marco Cesati, Understanding the Linux Kernel, O'Reilly & Associates Pub., 2001.