리눅스 저장 서버를 위한 적응 제어 버퍼 플러쉬 정책

박상수[0] 김태웅 신현식
서울대학교 컴퓨터공학부
{sspark, twkim}@cslab.snu.ac.kr, shinhs@snu.ac.kr

An adaptive buffer flush policy for Linux-based storage servers

Sangsoo Park[0] Taewoong Kim Heonshik Shin
School of Computer Science & Engineering, Seoul National University

요 약

대용량 데이터 및 사용자간 데이터 공유의 증가로 네트워크 저장 서버의 필요성이 증대되고 있다. 리눅스는 네트워크 저장 서버의 운영체제로 널리 사용되고 있지만 디스크 쓰기 성능에 큰 영향을 미치는 버퍼 플러쉬 정책이 다양한 디스크 입출력 부하에 적합하도록 되어 있지 않다. 본 연구에서는 효율적인 리눅스 기반 네트워크 저장 서버를 위한 디스크 입출력 부하에 따른 적응 제어 버퍼 플러쉬 정책을 제안한다.

1. 서 론

대용량 데이터 및 사용자간 데이터 공유의 증가로 대용량 데이터의 저장과 보호, 보다 효율적인 입출력 방식, 많은 사용자들의 데이터 공유 등의 문제가 발생하게 되었다. 최근의 고속 네트워크 장치 및 대용량 저장 장치의 개발은 이러한 문제를 효율적으로 처리할 수 있는 네트워크 저장 서버(NAS: Network Attached Storage)를 가능하게 하였다[1].

리눅스는 소스 코드가 공개된 운영체제로 안정성 및 성능이 검증되었을 뿐만 아니라 NFS, CIFS 등의 다양한 클라이언트를 지원 할 수 있기 때문에 네트워크 저장 서버를 위한 운영체제로 널리 사용되고 있다. (그림 1은 리눅스 기반 네트워크 저장 서버의 개요를 나타낸다.)

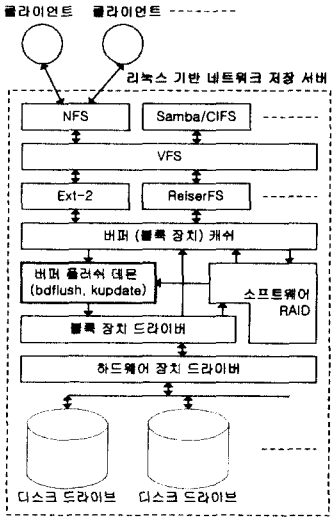

그림 1 리눅스 기반 네트워크 저장 서버의 개요

리눅스 기반 네트워크 저장 서버를 구축하는데 있어서 온라인 백업과 같은 매우 높은 디스크 입출력 부하를 가정하고 NFS로 마운트 된 파일 시스템(/nfs)에 대해 다음과 같이 8개의 1GB 디스크 쓰기 명령을 동시에 수행하여 보았다.

```
csh$ foreach i (1 2 3 4 5 6 7 8)
? dd if=/dev/zero of=/nfs/file.$i bs=1k count=1M &
? end
```

수행되는 과정에서 리눅스 서버에 시스템 멈춤 현상과 NFS 클라이언트에서 서버가 응답하지 않는다는 오류 메시지가 다수 발생하였다.

이러한 문제점이 발생하게 된 원인으로 본 연구에서는 리눅스의 버퍼 플러쉬 정책이 다양한 디스크 입출력 부하에 대해서 효율적이지 않다는 것을 발견하였다. 2절에서는 이러한 리눅스의 버퍼 플러쉬 정책의 문제점을 살펴보고 3절에서 다양한 디스크 입출력 부하에 따라 동적으로 조절되는 버퍼 플러쉬 정책을 제안하며 마지막으로 4절에서는 결론을 살펴본다.

2. 리눅스 버퍼 플러쉬 정책의 문제점

전통적인 유닉스는 응용 프로그램에서 디스크 쓰기 연산을 수행하면 커널의 버퍼 영역에 해당 데이터를 캐쉬하고 있고 주기적으로 동작하는 버퍼 플러쉬 데몬에 의해 버퍼 영역에 있던 데이터를 디스크에 쓰게 된다[2]. 고정된 버퍼 영역 크기와 정해진 버퍼 플러쉬 동작 주기를 갖는 전통적인 유닉스에 비해서 리눅스는 슬라브 할당기[3]의 사용으로 현재 동작중인 프로그램이 사용하는 영역 외의 모든 메모리 영역을 커널의 버퍼 영역으로 사용할 수 있으며 개선된 버퍼 플러쉬 데몬의 사용으로 커널 인자에 의해 버퍼가 플러쉬 되는 시점 및 한번에 플러쉬 되는 버퍼의 양을 조절하여 디스크 쓰기 성능을 조절 할 수 있다.[4].

일반적으로 네트워크 저장 서버는 많은 사용자들로부터 대용량의 데이터를 처리하기 때문에 대량의 버퍼 영역을 사용하기 위해 많은 메모리를 탑재하게 된다. 그런데, 온라인 백업 등과 같이 짧은 시간에 많은 데이터가 네트워크 저장 서버로 전달이 되면 운영체제는 데이터를 디스크에 바로 쓰지 않고 버퍼 영역에 저장한 후 버퍼 플러쉬 데몬이 동작되게 될 때 디스크에 쓰게 된다. 즉, 버퍼 영역에 저장되었던 대량의 데이터가 버퍼 플러쉬 데몬이 동작할 때 한꺼번에 디스크에 쓰이게 될 수 있으며 디스크 쓰기가 진행되는 동안 최악의 경우 시스템 멈춤 현상(stall)이 발생하게 되어 클라이언트로부터의 요청을 처리 할 수 없게 된다.

리눅스의 버퍼 플러쉬 데몬의 커널 인자를 높은 디스크 입출력 부하에 적합하도록 조절함으로써 이러한 시스템 멈춤 현상을 어느 정도 해결할 수 있지만 낮은 부하에 대해서는 디스크 입출력 성능이 나빠질 수 있는 문제점이 있다. 이러한 현상을 확인하기 위해 다음의 실험을 수행하였다.

실험은 펜티엄 III 850Mhz, 512MB 램, 4개의 4GB 5400RPM UW SCSI 디스크 드라이브를 갖춘 하드웨어와 레드햇 리눅스 6.2, 리눅스 커널 2.2.19, RAID 0의 소프트웨어 RAID 기반에서 수행되었다.

리눅스의 버퍼 플러쉬 데몬의 커널 인자는 /proc/sys/vm/bdflush 파일의 9개의 정수를 통해 확인 및 수정할 수 있는데[4] 디스크 입출력 성능에 영향을 미치는 인자는 nfract(1),ndirty(2),interval(5)로 표 1은 각 인자의 역할과 기본값(redhat) 및 [5]에서 제시하는 디스크 입출력 성능을 최적화하기 위한 인자값(SOL)을 나타낸다.

표 2 bdflush 인자

	역할	redhat	SOL
nfract	버퍼 영역에서 아직 디스크에 쓰이지 않은 버퍼의 최대 비율	40	100
ndirty	버퍼 플러쉬 데몬이 한번에 디스크에 쓸 수 있는 최대 버퍼의 수 (버퍼=4KB)	500	1200
interval	버퍼가 디스크에 쓰일 때까지의 최대 시간 (1초=100)	500	15

실험에는 연속적인 파일 쓰기 및 읽기를 주어진 크기만큼 각각 다른 방법으로 반복하여 파일 시스템의 읽기 쓰기 성능을 검사하는 벤치마크 프로그램인 bonnie[6]를 사용하였다. 실험은 디스크 입출력 부하가 낮은 경우와 높은 경우를 비교하기 위해서 디스크 읽기 연산의 경우 버퍼 영역에 모두 수용 가능한 100MB의 실험(-s 100) 및 버퍼 영역의 범위를 크게 벗어나는 1GB의 실험(-s 1000)을 수행하였다.

표 2는 벤치마크 수행 결과를 나타내고 그림 2,3,4는 sard[7]로 측정한 벤치마크 수행 시간 동안 발생한 디스크 읽기,쓰기 연산을 나타낸다.

sard는 SysV 계열 유닉스의 시스템 성능 측정 도구인 sar의 -d 옵션을 리눅스 상에서 구현한 것으로 시스템의 각 디스크 및 파티션별로 읽기,쓰기 명령어의 수행 시간과 양을 등의 디스크 입출력 정보를 /proc/partitons 파일을 통해 확인할 수 있다.l

표 3 redhat/SOL에 대한 bonnie 실험 결과

		100MB		1GB	
		redhat	SOL	redhat	SOL
쓰기	문자단위(kb/s)	11302	10984	10525	10566
	블록단위	**149559**	**212797**	**32053**	**15590**
읽기	문자단위	9652	9459	9723	7746
	블록단위	441398	440131	27695	27395

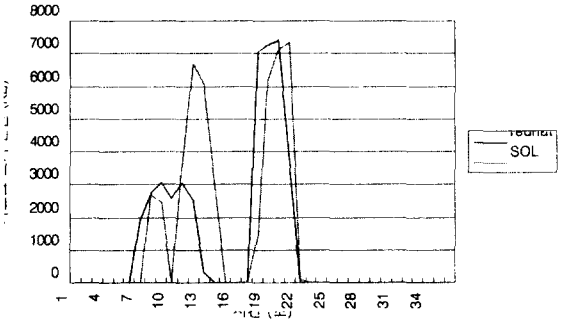

그림 2 bonnie -d /raid -s 100

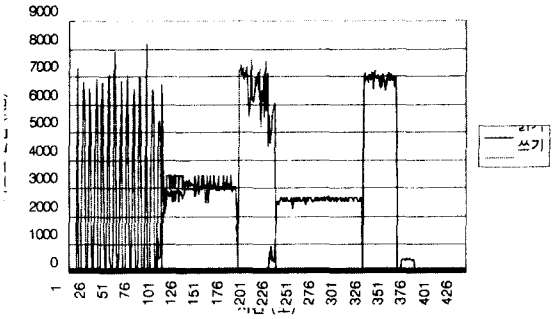

그림 3 bonnie -d /raid -s 1000 (redhat)

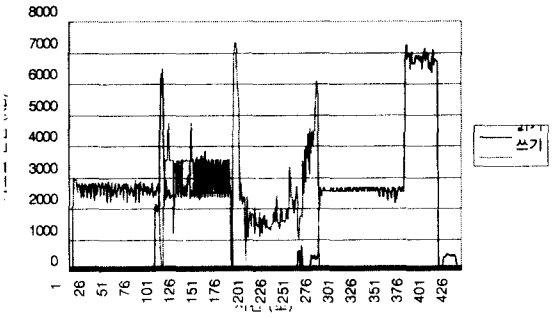

그림 4 bonnie -d /raid -s 1000 (SOL)

표 2를 보면 같은 벤치마크에 대해서 블록 단위로 쓰기 연산의 결과가 bdflush 인자에 따라 큰 차이를 나타낸다는 것을 알 수 있다. 또, redhat의 bdflush 인자가 낮은 디스크 입출력 부하 실험인 100MB 실험에서는 70% 정도 블록 단위 쓰기 성능이 나쁘지만 높은 디스크 입출력 부하 실험인 1GB 실험에서는 105% 정도 성능이 좋다는 것을 알 수 있다. 실험의 결과로 부하에 따라서 디스크 입출력 성능을 최적화 할 수 있는 bdflush 인자가 다르다는 것을 알 수 있다.

3. 디스크 입출력 부하에 따른 버퍼 플러쉬 정책

본 연구에서는 동적으로 변하는 디스크 입출력 부하에 따라서 디스크 입출력 부하를 감시하고 이 정보를 토대로 bdflush 인자를 주기적으로 갱신하는 방법으로 다양한 디스크 입출력 부하에 따른 버퍼 플러쉬 정책(DYN)을 수립하였다.

① 매 초마다 디스크 입출력 부하, 부하 중에서 디스크 쓰기 연산의 비율을 샘플링하고 n초 단위로 샘플링 된 값들을 평균 낸다.
② nfract는 부하 중에서 디스크 쓰기 연산의 비율로 하되 min_nfract 이상이 되게 한다. 읽기 연산만 수행되는 프로그램이 수행되는 중에 대량의 쓰기 연산이 들어오면 효율적으로 쓰기 연산을 수행할 수 없기 때문에 최소 일정 수준의 nfract 값을 유지한다.
③ ndirty는 부하와 디스크 쓰기 연산의 비율에 반비례한다. 단, ndirty는 min_ndirty와 max_ndirty 사이의 값을 유지한다.
④ interval은 부하와 디스크 쓰기 연산의 비율에 비례한다. 단, interval은 min_interval과 max_interval 사이의 값을 유지한다.
단, 이 때 n, min_fract, {min,max}_{ndirty,interval} 값은 디스크 입출력 부하에 종속적인 것이 아니라 시스템 하드웨어의 구성 및 성능에 종속적이며 실험에서는 순서대로 3, 30, 300, 1000, 15, 500의 값을 사용하였다.

표 3은 벤치마크 수행 결과를 나타내며 그림 5는 1GB의 실험의 디스크 읽기, 쓰기 연산을 나타내고 그림 6은 1GB의 실험의 수행 중 시간에 따른 디스크 입출력 부하 및 nfract, ndirty, interval 값을 백분율로 변환해 나타낸 것이다.
표 3의 수행 결과를 보면 redhat, SOL의 결과와 유사하거나 더 나은 결과를 나타내고 있고 그림 5는 redhat과 SOL의 실험의 디스크 읽기, 쓰기 연산을 혼합해 놓은 것과 같은 결과를 나타낸다.

4. 결론

리눅스의 버퍼 플러쉬 정책이 다양한 디스크 입출력 부하에 적합하지 않다는 것을 실험을 통해 확인하였고 동적으로 변하는 부하에 따라 bdflush 인자를 변경함으로써 시간에 따라 변하는 부하에 적합하도록 개선하였다.

표 4 DYN에 대한 bonnie 실험 결과

		100MB	1GB
쓰기	문자단위 (kb/s)	10580	10562
	블록단위	193302	38457
읽기	문자단위	9423	8553
	블록단위	428715	27279

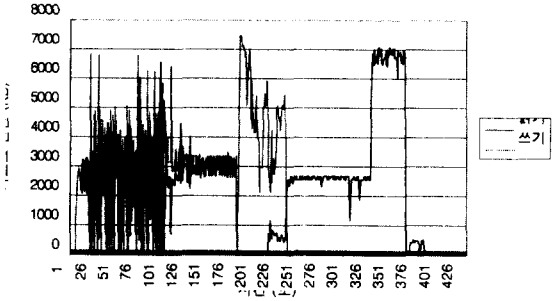

그림 5 bonnie -d /raid -s 1000 (DYN)

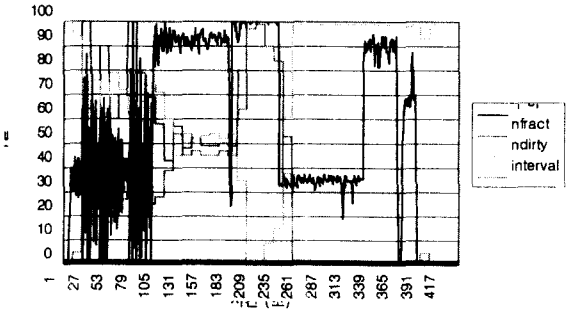

그림 6 시간에 따른 부하 및 nfract, ndirty, interval 값의 변화

5. 참고 문헌

[1] Hitachi Data Systems Corp. Planning for Network Attached Storage and Storage Area Network Convergence, 2001.
[2] Unix Internals
[3] Jeff Bonwick. The slab allocator: An object-caching kernel memory allocator. In Proceedings of the Summer 1994 USENIX Technical Conferrence, 1994.
[4] Michael Beck and Harald Bohme. Linux Kernel Internals 2nd Edition. Addison Wesley, 1997.
[5] Gerhard Mourani. Securing and Optimizing Linux: RedHat Edition - A Hands on Guide. Open Network Architecture, 2000.
[6] Linux Benchmark Suite. http://lbs.sourceforge.net/
[7] sard. ftp://ftp.linux.org.uk/pub/linux/sct/fs/profiling/