제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

GNBD/VIA 의 성능 분석

김강호*, 김진수**, 정성인*
*한국전자통신연구원 운영체제연구팀
**한국과학기술원 전자전산학과
e-mail : khk@etri.re.kr, jinsoo@computer.org, sijung@etri.re.kr

An Analysis of GNBD/VIA 's Performance

Kangho Kim*, Jin-Soo Kim**, Sungin Jung *
*Operating System Team, ETRI
**Computer Science Division, KAIST

요 약

VIA 는 클러스터 또는 시스템 영역 네트워크를 위한 표준화된 사용자수준 통신 아키텍쳐이고, GNBD 는 LINUX 클러스터에서 IP 네트워크 설비를 기반으로 GFS 공유 파일 시스템을 설치할 때 사용하는 네트워크 블록 디바이스이다. GNBD 는 TCP/IP 상의 소켓을 기반으로 구현되어 있기 때문에, VIA 를 사용하는 클러스터이더라도 VIA 하드웨어 상에서 TCP/IP 소켓을 통하여 GNBD 를 작동시킨다. VIA 와 같이 물리적 연결이 신뢰성이 높고 높은 수준의 기능을 제공하는 경우는 같은 클러스터 안에서 TCP/IP 프로토콜 스택을 사용할 필요가 없다. 그래서 우리는 VIA 를 이용하지만 TCP/IP 를 사용하지 않는 GNBD/VIA 를 구현하였고, 동일한 VIA 하드웨어를 사용하면서 TCP/IP 모듈을 이용하는 GNBD 보다 파일시스템의 읽기(쓰기) 성능이 약 20%(30%) 향상된다는 것을 확인하였다. 본 논문에서는 VIA 상에서 동작하는 GNBD/VIA 의 성능 측정값과 그 위에 설치된 파일시스템의 을 보여주고, 그 결과를 상세히 분석하여 GNBD/VIA 상에 설치된 파일 시스템이 발휘할 수 있는 성능의 한계를 제시한다. 제시하는 한계치는 GNBD/VIA 뿐만 아니라 TCP/IP 상의 소켓을 사용하는 GNBD 에도 적용할 수 있다.

1. 서론

새로운 네트워킹 기술이 등장함에 따라 저장 매체를 여러 노드가 공유할 수 있게 되었다. 공유된 저장 매체에 기록된 파일을 여러 노드가 동시에 접근할 수 있도록 만들어 주는 것이 공유 디스크 파일 시스템이다. GFS 는 LINUX 에서 사용할 수 있는 공유 클러스터 파일 시스템이다. GFS 클러스터 내의 노드들은 동일한 저장매체를 Fiber Channel, 공유 SCSI, 또는 네트워크 블록 디바이스(Network Block Device)를 사용하여 물리적으로 공유한다[1].

FC 또는 공유 SCSI 와 같은 별도의 스토리지 영역 네트워크(Storage Area Network)을 갖추지 않은 클러스터 환경에서 GFS 를 사용하려면 GFS 가 제공하는 NBD(GNBD)를 사용해야 한다. GNBD 를 포함한, 일반적인 NBD 는 TCP/IP 계층 위의 소켓을 사용하여 구현되어 있기 때문에 전통적인 LAN 또는 WAN 환경에 적합하지만 고성능 네트워크를 구비한 클러스트 환경에 그대로 적용하는 것은 적합하지 않다. 특히 VIA 와 같이 물리적 연결이 신뢰성이 있고 높은 수준의 기능을 제공한다면 같은 클러스터 안에서 TCP/IP 프로토콜 스택을 사용하여 통신할 필요가 없다. 그래서 [8]에서 VIA 상에서 GNBD 를 사용할 때, TCP/IP 스택을 사용하지 않는 간단한 커널 소켓 API(VCONN)를 구현하여 GNBD 인터페이스는 유지하면서 VIA 하드웨어의 성능을 최대한 이용할 수 있는 기법을 제안하였고, 간단한 성능 측정 결과를 보였다. 본 논문에서는 GNBD/VIA 에 관한 좀더 상세한 성능측정값을 보이고, 그 성능 측정 결과를 상세히 분석하여 GNBD/VIA 를 사용하는 파일 시스템이 발휘할 수 있는 성능의 한계를 제시하고 성능향상의 방향을 모색한다.

본 논문의 구조는 다음과 같다. 2 절에서 VIA 와 GNBD 를 간단히 설명하고, 3 장에서 VIA 상에서 GNBD 를 위한 통신모듈을 설계할 때 고려한 요소들을 간단히 설명한다. 4 장은 GNBD/VIA 를 이용한 파

일시스템의 성능측정 결과를 보이고, 5 장은 측정된 성능을 분석하여 GNBD 의 문제점을 파악한다. 6 장 결론을 맺으면서 성능향상을 위한 방향을 제안한다.

2. 배경

2.1 Virtual Interface Architecture (VIA)

VIA 는 네 개의 기본 요소, 즉 가상 인터페이스(VI: Virtual Interface), 완료 큐, VI-제공자, VI-수요자로 구성된다. VI-제공자는 물리적인 네트웍 어댑터와 커널 에이전트 및 라이브러리를 포함하며, VI-수요자는 VIA 를 이용하는 응용 프로그램을 말한다.

일반적으로 노드상의 여러 프로세스는 하나의 네트웍 어댑터를 공유하지만, VIA 에서는 각각의 VI-수요자에게 가상으로 하나의 전용 어댑터가 할당되어 있는 것과 같이 보이도록 지원한다. VI 는 송신 큐와 수신 큐의 쌍으로 이루어지며, VI-수요자는 전달할 또는 받을 메시지에 대한 정보를 담고 있는 디스크립터(descriptor)를 큐에 추가함으로써 데이터를 교환한다.

현재 사용할 LINUX 에서 사용할 수 있는 VIA 구현은 MVIA[7], Berkeley VIA[6], cLAN[3]이 있다.

2.2 GNBD

NBD 는 리눅스 커널에 포함된, 기본적인 네트워크 블록 디바이스로서 클라이언트가 다른 노드의 자원을 블록 디바이스처럼 사용하고자 할 때 필요하다. NBD 는 원격지의 블록 디바이스가 클라이언트에게 마치 로칼 디스크인 것처럼 보이도록 만들어준다.

GNBD 는 GFS 에 포함된 네트워크 블록 디바이스로서 위에서 설명한 NBD 를 변형한 것이다. NBD 와 GNBD 의 가장 큰 차이점은 NBD 가 한번에 오직 하나의 클라이언트에게 블록 디바이스 접근을 허용하는 반면에 GNBD 는 동시에 여러 클라이언트에게 접근을 허용한다는 것이다[4]

3. GNBD/VIA 설계

3.1 KVIPL

CLAN1000 드라이버가 커널 수준의 VIPL 을 제공하고 있으나, 하나의 독립 API 로 지원하는 것이 아니라 LANEVI 를 구현하기 위한 서브 모듈이어서 완전한 API 을 갖추지 않았다. 이 문제를 해결하기 위하여 KVIPL 을 확장하여 부족한 기능을 구현하였다[8].

3.2 흐름제어

Send()와 recv()를 동기화 시키기 위해서 기본적으로 2-way handshaking 기법을 사용한다. 수신측은 수신큐에 미리 한 개의 디스크립터를 두어서 recv()함수 호출과는 비동기적으로 도착하는 메시지를 수신한다. 메시지 수신 후 ACK 을 보내기 전에 소비한 디스크립터를 보충한다. 수신된 메시지는 버퍼에 저장했다가 recv()가 호출될 때 전달한다. 송신측은 ACK 을 받았을 때 다음 메시지를 전달할 수 있다. 기본 구조에 추가하여, 송신측이 ACK 을 받지 않고 여러개의 메시지를 연속으로 보낼 수 있게 하였다[8].

[표 1] 통신 지연시간

Message size (bytes)	Native_VIA(us)	VCONN(us)	LANEVI(us)
22	9	17	31
512	15	26	39
1024	20	32	48
4096	51	68	100

[표 2] 파일 입출력 대역폭

RW	Partition(MB/s)	File(MB/s)
Read(osync=1)	35.2	36.3
Read(osync=0)	35.3	36.4
Write(osync=1)	30.3	25.6
Write(osync=0)	42.2	37.3

4. 성능

실험 장비는 Intel 815EP 메인보드를 사용하는 두 대의 LINUX(커널 버전 2.2.18) 서버를 사용하였다. 각 서버는 256KB L2 케쉬와 512MB 메인 메모리를 가진 Pentium III-1GHz CPU 로 구성되어 있다. 두 서버는 두 장의 cLAN100 네트워크 어뎁터를, 별도의 스위치를 사용하지 않고, 바로 연결하였다. 디스크는 IDE UDMA66 인터페이스의 MAXTOR 40G 이고 디스크의 성능을 측정하는 도구는 bonnie++ 1.0 과 LMbench 를 사용하였다.

그림 4 는 GNBD 에 EXT2 파일시스템을 생성한 후 bonnie++를 사용하여 성능을 측정한 결과이다. VCONN 으로 통신했을 때 LANEVI 보다 읽기가 약 22.7%, 쓰기가 약 30.7% 향상되었다. 읽기의 경우 디스크를 직접 읽을 때와 비슷한 성능을 보일 정도로 우수한 성능을 보인다.

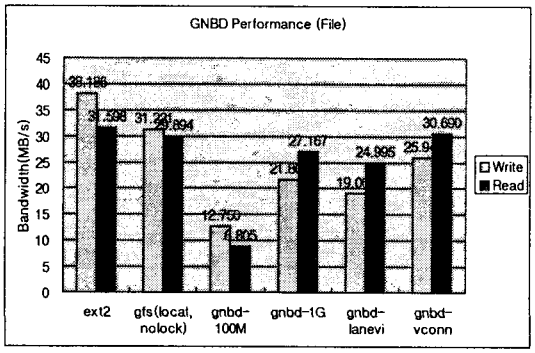
GNBD 의 통신이 요구(request)/응답(response) 형태이므로 지연시간(latency) 성능이 우수한 VCONN 을 사용하는 것이 성능을 향상시키는 것으로 판단된다. 표 1 에 나타나 있듯이 전체적으로 VCONN 이 LANEVI 보다 성능이 우수함을 알 수 있다.

5. 성능 분석

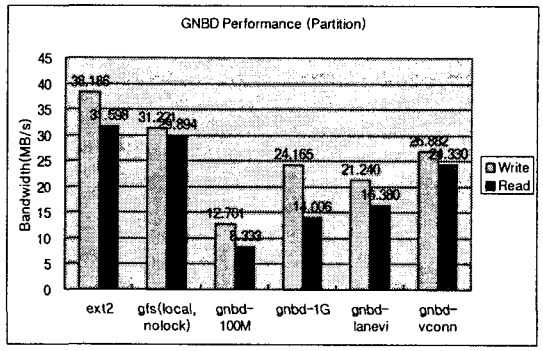
그림 1 에서 보여준 성능 결과에 의하면 GNBD 서버가 파일 또는 파티션을 제공하는 경우 모두 로칼의 디스크 파티션을 사용한 경우보다 전체적인 성능이 낮게 나타난다. 이 절에서는 그 이유를 분석하고, GNBD 의 개선점을 찾아본다.

전체적으로 GNBD 를 사용한 경우의 성능이 ext2-local 의 성능보다 낮은 이유는 크게 3 가지를 들 수 있다.

- **동기식 IO(synchronous IO)**: GNBD 서버는 의도적으로 동기식 모드로 파일을 열고 그 파일을 클라이언트에서 보여준다. 동기식 모드의 파일 입출력은 쓰기의 경우 버퍼링 시스템을 사용하지 않기 때문에 쓰기 대역폭이 크게 줄어 든다. 표 2 에 나타난 실험결과를 보면, 동기식 모드(osync=0)로 열어둔 블록 디바이스의 쓰기 대역폭은 약 30MB/s 인 반면 비동기식(osync=1) 즉 일반모드로 열어둔 것의 쓰기 대역폭은 약



[그림 1] 서버가 파일을 사용하는 경우 파일시스템의 성능



[그림 2] 서버가 파티션을 사용하는 경우 파일 시스템의 성능

42MB/s 이다. 표 2 의 값은 LMbench 의 lmdd 를 사용하여 측정한 것이다.

- **블록 클러스터링 않음**: 커널은 디스크로의 실제 읽기/쓰기 요구를 줄이기 위해서 여러 번의 작은 입출력 요구를 한번의 큰 연산으로 병합한다. 그런데 GNBD 를 사용하면 이러한 기능을 사용할 수 없기 때문에 입출력 대역폭이 낮아진다.
- **이중 버퍼링**: 하나의 블록이 원격지의 실제 디스크에 기록되기 위해서 클라이언트의 버퍼와 서버의 버퍼에 각각 복사된다.

파일 대 파티션: GNBD 를 사용하지만 서버가 사용하는 저장 형태가 다른 두 경우를 비교한다. 그림 1 의 그래프는 서버가 하나의 큰 파일을 사용하는 경우이고 그림 2 는 파티션을 사용하는 경우이다. 두 그래프의 값을 비교했을 때 그림 2 의 경우 쓰기 성능이 1 의 경우보다 조금 높은 것을 볼 수 있다. 이 차이는 파티션의 쓰기 과정이 파일보다 간단하기 때문이다. 파티션을 사용하면 클라이언트가 하나의 블록을 기록할 때 요구한 그 블록만 저장하는 반면에 파일을 사용하면 요구한 블록과 더불어 서버의 파일시스템에서 그 블록과 관련된 메타 데이터를 수정하고 디스크에 기록해야 하게 된다.

쓰기 대역폭의 경우와는 반대로, 읽기 대역폭에 관해서는 그림 2 가 그림 1 에 비해서 낮은 성능을 보인다. 그 이유는 파티션에서 순차적 블록 읽기 기능이 비효율적이기 때문이다. 파티션을 사용할 때 기본 미리 읽기(read ahead) 크기는 겨우 8 섹터(8 x 512 = 4KB)로 정해져 있기 때문에 미리 가져오기(refetching)에 의한 성능 향상을 기대할 수 없다. EXT2 파일시스템의 파일을 사용하면 최대 31 페이지만큼 미리 가져오기 때문에 파티션의 경우보다 읽기 성능이 높게 나타난다.

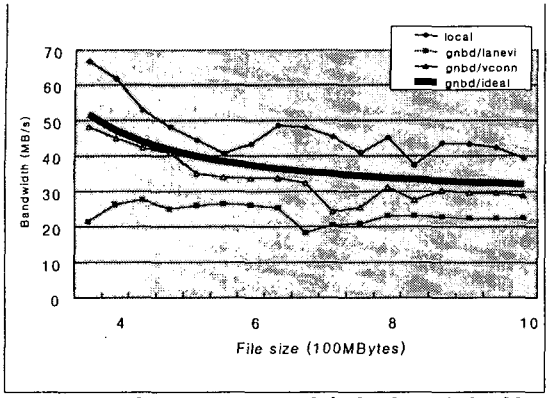
쓰기 대역폭의 상한값: 우리가 실험한 환경에서 가장 성능이 좋은 조합을 사용했을 때 얻을 수 있는 쓰기 대역폭의 최대값은 아래의 세가지 값을 더한 식 $B_w(f)$ 으로 나타낼 수 있다. f 는 파일의 크기, 0.4 는 버퍼교체가 일어나야 하는 버퍼사용율(40%), s 는 버퍼의 크기(512MB), b 는 블록의 크기(4KB), t_b 는 b 크기의 블록을 버퍼 메모리에 기록하는데 걸리는 평균시간(19us), t_d 는 b 크기의 블록을 디스크에 기록하는데 걸리는 평균시간(135us)이다.

- $\frac{0.4st_b}{b}$: $0.4s/b$ 개의 블록을 버퍼 메모리에 쓰는 총 시간
- $\frac{(f-0.4s)t_b}{b}$: $(f-0.4s)/b$ 개의 블록을 버퍼 메모리에 쓰는 총 시간
- $\frac{(f-0.4s)t_d}{b}$: 서버에 도착한 블록을 디스크에 기록하는데 걸리는 총 시간

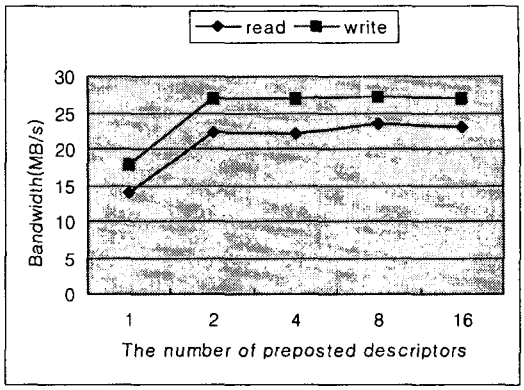
$$B_w(f)=\begin{cases}\dfrac{b}{t_b} & \text{if } f<0.4s\\ \dfrac{bf}{0.4st_b+(f-0.4s)(t_b+t_d)} & \text{otherwise}\end{cases}$$

이 식에는 통신 지연시간을 포함되어 있지 않다. 디스크 대역폭이 통신 대역폭보다 작기 때문에 입출력 요구를 연속으로 보내면 네트워크 지연시간을 무시할 수 있다. 이 식을 사용하여 그래프를 작성하면 그림 3 의 gnbd/ideal 처럼 그릴 수 있다. 예상한 대로 VCONN 을 사용한 GNBD (gnbd/vconn)이 LANEVI 를 사용한 경우보다 전체 구간에서 높은 성능을 보이고, 이상적인 GNBD 성능(gnbd/ideal)에 가까워지고 있다는 것을 알 수 있다. 그리고 실험환경과 같은 형태로 연결할 경우 절대로 gnbd/ideal 이상의 성능을 달성할 수 없다는 것과 gnbd/local 의 성능보다 항상 낮게 나타난다는 것도 알 수 있다.

읽기 대역폭의 상한값: bonnie++는 연속된 블록을 순차적으로 읽어서 대역폭을 측정하기 때문에 GNBD 서버가 파일을 사용하는 경우라면 클라이언트와 서버에서 미리 가져오기 기능을 최대한 이용할 수 있어서



[그림 3] GNBD 를 사용할 때 쓰기의 성능



[그림 4] 디스크립터의 개수와 대역폭

읽기 대역폭이 로칼 디스크를 사용할 경우와 차이가 없을 것이라고 예상할 수 있다. 그림 1 에서 gnbd-vconn 의 읽기 대역폭이 거의 ext2-local 의 대역폭과 같은 것으로 위 가정을 확인할 수 있다.

최적 디스크립터의 개수: GNBD/VIA 상에서 구축한 파일시스템의 블록 읽기/쓰기의 평균 응답시간이 GNBD 서버측 디스크 입출력 대역폭에 따라 결정되기 때문에 각 VCONN 연결에 단 두개의 디스크립터만 미리 포스팅하는 것으로 충분하다. 하나는 현재의 요구를, 다른 하나는 다음에 도착할 요구를 처리하는데 사용된다. 이 가정은 그림 4 에서의 결과로서 사실임을 알 수 있다. 미리 준비하는 디스크립터의 개수가 2 개 이상이 되더라도 입출력 대역폭은 변화가 없다. 클라이언트가 요청을 보내고 서버가 받는데 걸리는 시간(17us)과 서버가 응답을 되돌려 보내는데 걸리는 시간(68us)의 합이 85us 이고 그 값은 한 개의 블록을 기록하는데 걸리는 평균 시간(135us)보다 작기 때문에 순차적 접근에서는 첫번째 요청이 서버에 도착하고 그 요청을 처리하는 동안 두번째 요청에 서버에 도착해 있게 된다.

6. 결론

본 논문은 [8]에서 제안한, VIA 상에서 GNBD 를 위한 고속 통신 계층(VCONN)을 사용하여 파일시스템을 구축했을 때 그 파일시스템의 성능을 보여주고 성능을 상세히 분석하였다. 분석 결과를 바탕으로 GNBD 를 사용했을 때 파일 시스템이 나타내는 입출력 대역폭의 상한값을 제시하였다.

· 우리의 실험 환경과 같은 일대일 연결에서는 제시한 상한값을 초과할 수 없다는 것을 알 수 있다. 이러한 GNBD 의 구조적문제로 인하여 상한값이 존재하기 때문에 구조적 문제를 해결하지 않으면 일대일 연결에서 더 이상 성능향상이 불가능하다. 구조를 변경하지 않고 GNBD 의 성능을 최대한 이용하기 위해서는 파일 시스템을 사용하는 응용프로그램의 특성에 따라 GNBD 서버가 사용하는 저장 형태를 선택하면 된다. 응용프그램이 읽기 연산이 많은 경우는 파일을, 쓰기 연산이 많은 경우는 파티션을 사용하는 것이 유리하다.

GNBD 를 사용하면서 VIA 또는 다른 고속통신 수단의 대역폭을 최대한 이용하기 위해서는 GNBD 클라이언트와 서버가 일대다 연결을 가져야 한다. 클라이언트는 각 연결을 하나의 블록 디바이스로 볼 수 있으므로 그 디바이스를 소프트웨어 RAID 로 묶어서 사용하면 성능이 향상될 것으로 기대한다.

참고문헌

[1] Andrew Barry, at el., "Implementing Journaling in a Linux Shared Disk File System," Sistina Software, Inc.

[2] Jin-Soo Kim, Kangho Kim, and Sung-In Jung, "Building a High Performance Communication Layer Over Virtual Interface Architecture on Linux Clusters," Proceedings of the 15th ACM International Conference on Supercomputing (ACM ICS '01), pp.335-347, Sorrento, Italy, June 2001.

[3] "VI Architecture Software Developer's Guide," Giganet, 1999.

[4] Kevin Duncan, "Fiber channel and Gigabit Ethernet: A Look at Technology for Storage Networking Solutions," University of Minnesota Fiber Channel Group, Minneapolis, MN May 14, 2001.

[5] B. Chun, A. Mainwaring, and D. Culler. Virtual Networking Transport Protocols for Myrinet. IEEE Micro, 31(1):53-63, Jan. 1998.

[6] Philip Buonadonna, Andrew Geweke, and David E. Culler, "An Implementation and Analysis of the Virtual Interface Architecture," In Proc. SC '98, 1998.

[7] M-VIA, "http://www.nersc.gov/research/FTG/via/

[8] Kangho Kim, Jin-Soo Kim, and Sung-In Jung, "GNBD/VIA: A Network Block Device over Virtual Interface Architecture on Linux," Proceedings of the 16th International Parallel & Distributed Processing Symposium, Florida, USA, April 2002.