i²SCSI: Storage Area Network에서 캐시 일관성을 제공하는 지능적인 iSCSI 공유 디스크

이 주 평, 황 주 영, 임 승 호, 박 규 호
한국과학기술원 전자전산학과 전기및 전자공학 전공
전화 : 042-869-5425 / 핸드폰 : 016-598-4182

i²SCSI: Intelligent iSCSI Shared Disk Providing Cache Consistency in Storage Area Network

Jupyung Lee, Joo Young Hwang, Seung-Ho Lim, Kyu Ho Park
Electrical Engineering and Computer Science, KAIST
E-mail : jplee@core.kaist.ac.kr

Abstract

The internet SCSI(iSCSI) disk has been studied as a storage system which can be directly connected to TCP/IP network. We designed and implemented a shared disk following the iSCSI protocol and providing cache consistency. It is named as intelligent iSCSI(i2SCSI) disk.

The i2SCSI disk provides cache consistency of all blocks that belong to the disk using a conventional lease method and it uses 'contiguous blocks-level locking'. The prototype of the i2SCSI disk emulator and its client is designed and implemented in Linux 2.4.

I. 서론

여러개의 클라이언트 노드들이 공유 디스크(shared disk)를 공유하고 직접 접근하게 해 주는 네트워크를 storage area network(SAN) 이라고 부른다[1].

storage area network(SAN)는 여러개의 클라이언트 노드들이 공유 디스크(shared disk)를 공유하고 직접 접근하게 해 주는 네트워크이다. SAN을 구성하는데는 주로 fibre channel 네트워크가 이용되고 있다. fibre channel 네트워크는 넓은 대역폭을 지원하고 디스크의 확장성이 우수하지만 TCP/IP 네트워크와 같이 널리 쓰이고 있지 않고 설치 및 유지 비용이 비싸다는 단점이 있다.

이러한 단점 때문에 이미 널리 쓰이고 있는 TCP/IP 네트워크를 이용하여 SAN을 구성하고자 하는 연구가 활발히 진행되어 왔고 이를 위한 프로토콜로 iSCSI 프로토콜이 제안되었다[2]. iSCSI는 TCP/IP 네트워크 상에서 SCSI 명령과 데이터들이 오고 갈 수 있게 한 SCSI transport protocol이다.

iSCSI 공유 디스크는 TCP/IP 네트워크를 통해 여러개의 클라이언트 노드들이 동시에 접근하는 연산을 허용하기 때문에 디스크에 저장된 데이터의 캐시 일관성을 유지하기 위한 장치가 필요하다. 기존의 연구들에서는 주로 락 서버를 두고 클라이언트들이 공유 디스크로 접근하기 전에 먼저 락 서버로부터 락을 부여받게 하는 기법을 이용한다. 이 기법은 공유 디스크의 개수가 많아져서 대용량으로 확장될 때 모든 디스크에 대한 락 관리를 락 서버가 담당해야 하므로 병목점이 될 가능성이 있으며 락 서버가 고장나게 되면 모든 공유 디스크들에 대한 락 정보를 잃게 되므로 전체 시스템의 데이터 일관성이 깨진다는 단점이 있다.

본 연구에서는 iSCSI 공유 디스크 스스로가 자신의 데이터에 대해 캐시 일관성을 보장하는 보다 지능적인 iSCSI 공유 디스크(intelligent iSCSI; i²SCSI)를 제안한다. i²SCSI 공유 디스크는 내부적으로 자신의 데이터에 대한 락을 관리한다. 각 공유 디스크는 자신의 데이터에 대한 락만을 내부적으로 관리하므로 기존에 락 서버로 집중되던 부하가 각 디스크들에게 분산된다. 또

한 공유 디스크 중 하나가 고장났을 때 해당 디스크에 대한 락 정보를 잃을 수 있지만 다른 디스크에 대한 락 정보에는 아무런 영향을 주지 않게 되서 전체 시스템의 가용성이 높아지게 된다.

이 논문에서는 i²SCSI 디스크의 개념, 설계, 및 구현에 대해 설명하고 성능 측정 결과를 제시한다.

II. 캐시 일관성을 유지하기 위한 기법

본 절에서는 캐시 일관성을 유지하기 위한 기법들을 설명한다.

대부분의 기법들은 읽기 또는 쓰기 연산을 하기 전에 해당 데이터에 대해 락을 잡는 과정을 거치게 함으로써 캐시 일관성을 유지한다. 락은 shared lock과 exclusive lock으로 나눌 수 있다. shared lock은 여러 클라이언트가 동시에 잡고 있을 수 있는 락이고 exclusive lock은 한 클라이언트만이 잡고 있을 수 있는 락이다.

캐시 일관성을 유지하기 위한 기법은 다음과 같이 나눌 수 있다.

- Poll-Each-Read
- Poll
- Callback [3]
- Lease [4]

처음의 두 가지 방식은 client polling 방식이며 뒤의 두 가지 방식은 server invalidation 방식이다.

poll-each-read는 가장 간단한 기법이다. 클라이언트는 읽기 연산을 하기 전에 shared lock을, 쓰기 연산을 하기 전에 exclusive lock을 잡는다. 연산을 끝낸 후에 각 클라이언트는 다른 클라이언트들이 락을 잡을 수 있도록 곧바로 락을 놓는다. 다른 클라이언트들이 exclusive lock 또는 shared lock을 잡고 있는 경우 클라이언트는 exclusive lock을 잡을 수 없고 모든 락이 놓여질 때 까지 계속 polling해야 한다. 이 기법은 간단하지만 매 읽기 연산 전에 락을 받아와야 하기 때문에 읽기 연산의 응답시간이 길다는 단점이 있다. (클라이언트에 캐시되어있지 않은 데이타 뿐만 아니라 캐시되어있는 데이터에 접근하기 전에도 락을 받아와야 한다.) 뿐만 아니라 락을 요청하는 횟수가 많아 서버에 큰 부하를 주게 된다.

poll은 기본적으로 poll-each-read와 같은데 읽기 연산 후에 캐시된 데이터가 일정 시간 동안 유효하다고 가정함으로써 한 번 캐시된 데이터에 대한 읽기 연산의 응답시간을 줄이고 락을 요청하는 횟수를 줄인다. 이 기법은 poll-each-read에 비해 읽기 성능이 좋아지지만, 캐시가 유효한 시간을 길게 잡았을 때 캐시 일관성이 깨질 수 있을 뿐만 아니라 그러한 가정을 하기가 어려우므로 실제로 사용하기는 힘들다.

callback[3]의 경우 각 클라이언트들은 락을 잡고 연산이 끝난 후에도 바로 락을 놓지 않고 서버로부터 락을 놓아라는 요청이 들어오기 전 까지는 계속 락을 잡고 있다. 이 경우 캐시된 데이터에 대한 읽기 연산의 성능이 좋아질 수 있는데 이는 이미 해당 데이터에 대한 락을 가지고 있을 확률이 높기 때문이다. 보통 파일시스템의 workload는 temporal locality가 높기 때문에 이러한 특성은 전체 성능을 상당히 높일 수 있다. 또한 락 요청의 횟수가 줄어들어서 서버의 부하가 작아진다. 이 기법의 단점은 모든 클라이언트로부터의 락이 반환된 후에야 exclusive lock을 잡을 수 있게 되므로 쓰기 연산의 성능이 나빠진다는 점, 그리고 락을 잡고있던 클라이언트가 다운되면 영원히 락을 반환하지 않게 되어 영원히 쓰기 연산을 수행할 수 없게 된다는 점이 있다.

lease[4]는 위에서 언급한 callback 기법의 문제를 해결한 기법이다. lease는 기본적으로 callback과 같은데 락이 일정 시간 후에 만료된다는 점에서 callback기법과 다르다. 락이 만료된 후 해당 데이터에 접근하기 위해서는 다시 락을 잡아야 한다. 이 기법의 경우 만료된 락은 다시 서버로 반환되므로 exclusive lock을 잡기 위해 반환되어야 하는 락의 갯수가 줄어든다. 또한 한 클라이언트가 락을 잡은 채로 다운되더라도 일정 시간이 지난 후에 해당 락이 만료되므로 영원히 쓰기 연산이 지연되는 경우가 없다.

III. i²SCSI 클라이언트와 디스크의 설계 및 구현

3.1 용어 정의

이 절에서는 이 논문에서 이용하는 용어에 대한 정의를 한다.

이 논문에서 하나의 락이 담당하는 데이터의 범위를 'lock unit'이라고 정의한다. 예를 들어 파일단위로 락을 잡는 경우 lock unit은 하나의 파일이 포함하는 디스크 블록들이 된다.

이 논문에서 i²SCSI 디스크로 읽기 또는 락 요청이 왔을 때 바로 서비스를 해 주게 되면 캐시 일관성이 깨지는 경우 'lock conflict 사건이 발생했다' 고 부른다.

3.2 i²SCSI의 캐시 일관성을 유지하기 위한 기법

이 절에서는 i²SCSI 디스크의 캐시 일관성 유지를 위한 기법을 설명한다.

i²SCSI는 lease 기법을 통해 캐시 일관성을 유지한다. 이 기법에 대해서는 2장에서 자세하게 설명했으므로 여기에서는 설명을 생략한다.

대부분의 경우 캐시 일관성을 유지하기 위해 파일 단위로 락을 잡는다. 그런데 i²SCSI 디스크에서는 다음과 같은 이유로 연속된 디스크 블록들을 단위로 락을 잡는다.

- 파일단위로 락을 잡는 경우 파일의 갯수에 비례하는 메모리 크기를 요구하게 되는데 디스크 내부에 그렇게 큰 메모리를 집적하기가 힘들다. 전체 디스크 블록을 연속된 디스크 블록 단위로 크게 나누어서(lock unit을 크게 해서) 락을 잡으면 요구되는 메모리 크기가 줄어든다. 예를 들어 전체 디스크 용량이 100Gbytes고 7대의 클라이언트의 동시 접속을 허용하며 lock unit 크기가 100kbytes인 경우 1Mbytes의 메모리를 요구하게 된다. 파일단위의 락을 허용하게 되면 이보다 큰 메모리를 요구하게 될 것이다.
- 연속된 디스크 블록들을 단위로 락을 잡는 경우 읽기 연산 이전에는 락을 잡을 필요가 없어진다. (즉, shared lock은 필요없어진다.) 왜냐하면 읽기 연산을 위해 클라이언트가 보내는 명령에는 클라이언트가 읽고자하는 블록을 알 수 있는데 이를 통해 클라이언트가 어느 락을 요청하는지를 정확히 알 수 있기 때문이다. 즉, 읽기 요청 자체가 락 요청의 의미까지 포함한다. (파일단위 락의 경우 디스크는 어느 블록이 어느 파일에 속하는지를 알 수 없기 때문에 읽기 요청과 락 요청을 합칠 수 없다.) 이러한 특성은 읽기 연산의 응답시간을 크게 단축시킨다.
- 데이타베이스와 같이 파일시스템을 거치지 않고 디스크에 접근하는 어플리케이션들이 있다. 이러한 어플리케이션은 파일 단위로 디스크에 접근하지 않기 때문에 파일 단위로 락을 잡는 기법을 사용할 수 없다. 연속된 디스크 블록들을 단위로 락을 잡는 i²SCSI는 파일시스템을 거치는 어플리케이션 뿐만 아니라 그렇지 않은 어플리케이션에도 적용될 수 있다.

위에서 언급했듯이 i²SCSI 디스크는 shared lock이 필요없고 exclusive lock 만을 필요로 하기 때문에 이 논문에서 '락'은 exclusive lock을 의미하는 것으로 한다.

연속된 디스크 블록들을 단위로 락을 잡는 경우 lock unit의 크기가 전체 성능에 큰 영향을 미치게 된다. lock unit이 커지면 디스크 블록을 순차적으로 접근할 때 적은 횟수의 락을 필요로 하므로 순차적 읽기/쓰기 성능이 좋아질 수 있고 요구되는 메모리 크기가 작아지는 장점이 있는 반면에 lock conflict 사건의 빈도가 높아질 수 있다.

3.3 i²SCSI 클라이언트와 디스크가 교환하는 메시지의 정의

i²SCSI 클라이언트와 디스크는 SCSI command, SCSI response, SCSI data, Ready-to-Transfer 등 iSCSI 프로토콜 메시지를 주고받으며 동시에 캐시 일관성을 유지하기 위한 메시지를 주고받는다. i²SCSI에서 캐시 일관성을 보장하기 위한 메시지들을 다음과 같이 정의한다.

- LOCK : 클라이언트가 디스크에게 락을 요청한다.
- LOCKED : 디스크가 클라이언트에게 락을 보낸다.
- DO_UNLOCK : 디스크가 클라이언트에게 락을 해지할 것을 요청한다.
- UNLOCKED : 클라이언트가 디스크에게 DO_UNLOCK명령을 완료했음을 알린다.
- DO_FLUSH : 디스크가 클라이언트에게 캐시를 invalidate시키고 락을 가지고 있다면 락을 해지할 것을 요청한다.
- FLUSHED : 클라이언트가 디스크에게 DO_FLUSH명령을 완료했음을 알린다.

3.4 i²SCSI 클라이언트와 디스크의 구현

i²SCSI 클라이언트는 linux 2.4.13 커널, ext2 파일시스템, University of New Hampshire의 iSCSI initiator device driver를 기반으로 구현되었다. i²SCSI 디스크는 linux 2.4.13 커널, University of New Hampshire의 iSCSI target emulator를 기반으로 구현되었다[5]. (하나의 PC가 하나의 디스크에 대한 에뮬레이터 역할을 한다.) 지면관계상 자세한 설명을 생략한다.

IV. i²SCSI 디스크의 성능 측정

4.1 실험환경

네 대의 클라이언트와 한 대의 i²SCSI target

emulator(disk)로 구성했다. client와 target emulator 모두 Intel Pentium III 450MHz, 128MB SDRAM 의 사양을 가지고 target emulator에는 SEAGATE ST39175LW SCSI disk가 장착되어 있다. 각 client와 target emulator는 100MHz ethernet에 연결되어 있다.

4.2 Realistic Workload Test

우리는 실제로 i²SCSI 디스크가 사용될 때의 성능을 예측해보고 lock unit의 크기가 실제 성능에 얼마나 영향을 끼치는지 알아보기 위해 real workload에 가까운 벤치마크를 설계하여 돌려보았다.

사용한 벤치마크는 다음과 같은 연산을 수행한다. 디스크에 미리 각 request size(4k, 8k, 16k, 32k, 64k)에 대해 1000개의 파일을 생성해 둔다. 각 클라이언트는 1000개 중의 하나의 파일을 랜덤하게 열고 읽기:쓰기 = 7:3의 비율로 읽기 또는 쓰기를 수행한 다음에 파일을 닫는다. 각 클라이언트는 이 트랜잭션을 2000번씩 수행한다. 두 대의 클라이언트가 동시에 이 벤치마크를 수행하게 되면 확률적으로 lock conflict 사건이 발생할 것이다.

벤치마크 실행 결과를 그림 1에 나타내었다. 전체적으로 request size의 크기에 따라 500kbytes/sec에서 3000kbytes/sec까지의 bandwidth가 나왔다. 생각보다 낮은 bandwidth가 나온 것은 lock conflict 사건을 해결하기 위한 오버헤드 때문이다. 그리고 request size가 작을 때 성능이 낮은 이유는 request size가 작아서 네트워크 bandwidth를 충분히 사용하지 못하기 때문이다. lock unit 크기가 성능에 미치는 영향을 살펴보면 request size가 작을 때는 lock unit이 작을 때의 성능이 더 좋고 request size가 클 때는 lock unit이 클 때의 성능이 더 좋음을 볼 수 있다. 3.2절에서 lock unit이 커질수록 순차적 읽기/쓰기 성능이 좋아질 수 있으나 lock conflict이 발생하는 횟수가 늘어날 수 있음을 설명했다. request size가 클 때는 순차적 읽기/쓰기 성능이 좋아지는 것이 더 유리하므로 lock unit이 큰 경우가 성능이 더 좋게 나왔으나 request size가 작을 때는 lock conflict가 적게 발생하는 것이 더 유리하므로 lock unit이 작은 경우가 성능이 더 좋게 나온 것으로 해석된다.

V. 결론 및 추후과제

이 논문에서는 TCP/IP 기반 storage area network 환경에서 클라이언트 사이의 캐시 일관성을 보장하는 i²SCSI 디스크를 제안하였다. 제안된 i²SCSI 디스크는

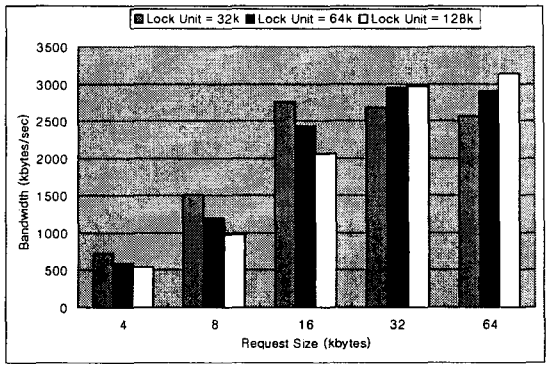

그림 1 Realistic Workload Test

lease 기법을 이용하여 캐시 일관성을 보장하고 연속된 디스크 블록 단위로 락을 잡는데 파일 단위로 락을 잡는 경우에 비해 요구되는 메모리 크기가 작고 읽기 락이 필요없으며 파일시스템을 거치지 않는 어플리케이션에도 사용될 수 있다는 장점을 가진다.

Linux 클러스터 상에서 i²SCSI 클라이언트와 디스크 에뮬레이터를 구현하여 그 성능을 측정해 보았다.

앞으로는 클라이언트들 간의 cooperative caching 기능 등을 갖춘 보다 지능적인 i²SCSI 디스크를 제안하기 위한 연구를 진행할 예정이다.

참고문헌

[1] A. Benner, Fibre Channel: Gigabit Communications and I/O For Computer Networks, McGraw-Hill, 1996

[2] Julian Satran, et. all, iSCSI. http://www.ietf.org/internet-drafts/draft-ietf-ips-iscsi-19.txt

[3] J.H.Howard, "An Overview of the Andrew File System," In Proceeding of the USENIX 1998 Winter Conference, pp. 23-26, 1998.

[4] C. Gray, et. all, "Leases: An efficient fault-tolerant mechanism for distributed file cache consistency," In Proceeding of the 12th ACM Symposium on Operating Systems Principles, pp. 202-210, 1989

[5] Ashish Palekar, et. all, "Design and Implementation of a Linux SCSI Target for Storage Area Networks," In Proceeding of the 5th Annual Linux Showcase & Conference, November 2001.