SAN 환경에서 공유 디스크 파일 시스템을 위한 전역 버퍼 관리자

(A Global Buffer Manager for a Shared Disk File System in SAN Clusters)

박 선 영 † 손 덕 주 †† 신 범 주 ††† 김 학 영 † 김 명 준 ††††
(Seon-Yeong Park) (Duk Joo Son) (Bum-Joo Shin) (Hak-Yong Kim) (Myung-Joon Kim)

요 약 네트워크를 통해 전송되는 데이타의 양이 급속히 증가함에 따라 확장성 있는 저장 시스템에 대한 사용자 요구가 증가하고 있다. 네트워크 연결형 자료 저장 시스템인 SAN(Storage Area Network)은 호스트와 디스크를 광채널 스위치로 연결하는 구조로서 저장 공간과 서버에 대한 확장성을 제공한다. SAN 환경에서는 다수의 호스트가 네트워크에 연결된 저장 장치를 공유하므로 공유 데이타에 대한 일관성 유지가 필요하다. 이를 위해 각 호스트가 수정한 데이타를 즉시 디스크에 반영하는 방법을 사용하고 있지만 이는 느린 디스크 접근 시간(Disk Access Time)으로 인해 시스템의 성능을 저하시키는 요인이 된다.

본 논문에서는 필요한 공유 데이타를 다른 호스트의 메모리를 통해서 직접 전송 받을 수 있도록 하여 공유 데이타의 접근 속도를 향상시킬 수 있는 전역 버퍼 관리자의 설계와 구현에 대해 소개한다. SANtopia 전역 버퍼 관리자는 SAN에 연결된 호스트들이 서로의 버퍼 캐시를 공유하도록 함으로써 블록 데이타로의 빠른 접근을 가능하게 한다. 마이크로 벤치마크를 통한 블록 단위 I/O의 성능 측정 결과, 전역 버퍼 관리자를 사용하는 것이 기존의 디스크 I/O를 사용하는 방법에 비해 약 1.8-12.8배 정도 빠른 성능을 보였으며 파일 시스템 벤치마크를 통한 성능 측정 결과, 전역 버퍼 관리자를 사용한 SANtopia 파일 시스템은 사용하지 않은 것과 비교해서 디렉터리 파일 시스템 콜의 경우 약 1.06배 정도 빠르고 일반 파일시스템 콜은 약 1.14배 정도 빠른 성능을 보였다.

키워드 : 전역 버퍼 관리자, 공유 파일 시스템, 캐시 일관성, SAN(Storage Area Network), 클러스터

Abstract With rapid growth in the amount of data transferred on the Internet, traditional storage systems have reached the limits of their capacity and performance. SAN (Storage Area Network), which connects hosts to disk with the Fibre Channel switches, provides one of the powerful solutions to scale the data storage and servers. In this environment, the maintenance of data consistency among hosts is an important issue because multiple hosts share the files on disks attached to the SAN. To preserve data consistency, each host can execute the disk I/O whenever disk read and write operations are requested. However, frequent disk I/O requests cause the deterioration of the overall performance of a SAN cluster.

In this paper, we introduce a SANtopia global buffer manager to improve the performance of a SAN cluster reducing the number of disk I/Os. We describe the design and algorithms of the SANtopia global buffer manager, which provides a buffer cache sharing mechanism among the hosts in the SAN cluster. Micro-benchmark results to measure the performance of block I/O operations show that the global buffer manager achieves speed-up by the factor of 1.8-12.8 compared with the existing method using disk I/O operations. Also, File system micro-benchmark results show that SANtopia file system with the global buffer manager improves performance by the factor of 1.06 in case of directories and

† 비 회 원 : 한국전자통신연구원 컴퓨터시스템연구부 연구원
sypark@etri.re.kr
h0kim@etri.re.kr
†† 종신회원 : 한국전자통신연구원 인터넷컴퓨팅연구부 연구원
djson@etri.re.kr
††† 비 회 원 : 밀양대학교 컴퓨터공학부 교수
bjshin@mnu.ac.kr
†††† 종신회원 : 한국전자통신연구원 컴퓨터소프트웨어연구소 연구원
joonkim@etri.re.kr
논문접수 : 2001년 12월 17일
심사완료 : 2003년 10월 27일

1.14 in case of files compared with the file system without a global buffer manager.
Key words : Global Buffer Manager, Shared File System, Cache Consistency, SAN(Storage Area Network), Cluster

1. 서 론

인터넷 사용자가 증가함에 따라 네트워크를 통해 전달되는 데이타의 양도 급속히 증가하고 있다. 특히 음성, 이미지, 동영상 등과 같은 대용량 멀티미디어 데이타가 차지하는 비중이 증가함에 따라 네트워크의 전송 속도나 서버의 처리 속도에 못지않게 대용량 저장 시스템에 대한 필요성도 증대되고 있다. 또한, 더 많은 사용자를 수용하고 저장 장치의 확장과 백업을 용이하게 하기 위해 전통적인 저장 방식이 아닌 새로운 저장 시스템에 대한 요구가 대두되고 있다. 이에 대한 하나의 해결책으로 서버와 스토리지 사이를 광 채널(Fibre Channel) 네트워크로 연결하는 SAN(Storage Area Networks) 방식의 네트워크 연결형 저장 시스템이 제안되었다[1]. SAN 방식의 저장 시스템은 그림 1과 같이 다수의 호스트가 스위치를 통해서 여러 대의 저장 장치를 공유할 수 있도록 구성된다.

그림 1에서 네트워크에 연결된 저장 장치 상의 데이타를 다수의 호스트가 공유하기 위해서는 특별한 파일 시스템이 필요하며 이러한 목적으로 구현된 파일 시스템을 공유 디스크 파일 시스템(Shared Disk File System)이라 한다. 이러한 공유 디스크 파일 시스템은 대용량 파일을 저장할 수 있도록 특별한 자료 구조를 사용해야 함은 물론 네트워크 저장 장치 상에 생성한 파일 시스템을 여러 호스트가 공유할 수 있도록 해야 한다. 각 호스트의 논리 볼륨 관리자(Logical Volume Manager)는 네트워크에 연결된 다수의 저장 장치를 마치 하나의 볼륨으로 사용할 수 있도록 한다.

각 호스트는 자신의 메모리를 사용하여 파일 데이타를 저장하기 때문에 여러 호스트가 동시에 공유 디스크 파일 시스템을 사용하기 위해서는 일반 데이타 혹은 메타 데이타에 대한 일관성을 보장하는 방법이 필요하다. 즉, 파일에 대한 읽기 혹은 쓰기 동작을 수행하더라도 단일 파일 시스템에서 수행한 것과 같은 결과를 보장해야 한다. 호스트들간 데이타 일관성을 보장하기 위한 하나의 방법으로 변경된 데이타를 즉시 공유 디스크에 반영하고(write-through) 읽기 요구는 메모리를 거치지 않고 디스크 I/O를 통해 수행 할 수 있다. 그러나 각 호스트가 데이타의 읽기 혹은 쓰기 동작을 수행할 때마다 디스크 I/O가 수행된다면 느린 디스크 접근 시간(Disk Access Time)으로 인해 파일 시스템의 성능을 저하시키게 된다. 또한 여러 호스트가 동시에 디스크 I/O를 수행하게 되면 디스크 접근에 대한 경쟁이 발생하여 이로 인한 추가적이 지연이 초래된다[2]. 따라서 공유 데이타를 디스크를 통해서가 아닌 다른 호스트의 메모리를 통해서 직접 전송 받는 방법으로 좀더 빠른 데이타 접근이 가능하다.

본 논문에서 제안하는 SANtopia 전역 버퍼 관리자(Global Buffer Manager)는 SAN 환경의 공유 디스크 파일 시스템에서 다수의 호스트가 버퍼 캐시를 서로 공유하게 함으로써 디스크 I/O로 인한 지연 시간과 스위치 혹은 디스크 컨트롤러의 부하를 감소시키도록 고안

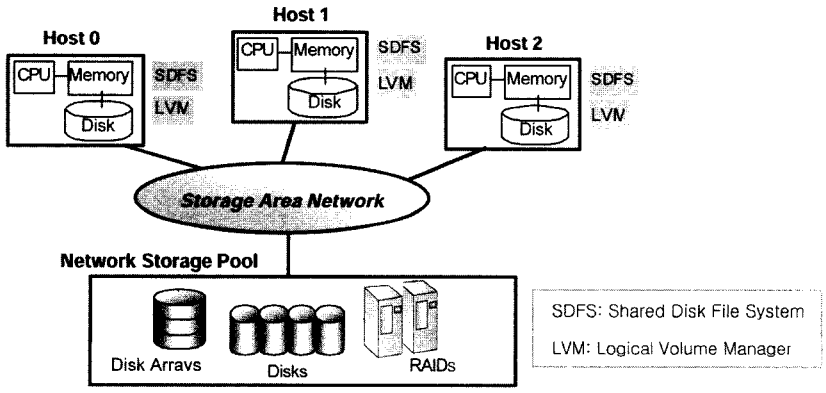


그림 1 Storage Area Network의 구조

되었다. SANtopia 시스템은 공유 데이타에 대한 접근 제어를 위해 호스트간 잠금 방법을 이용하며 전역 버퍼 관리자는 잠금 정보를 사용하여 추가적인 네트워크 트레픽을 발생하지 않고 전역 버퍼 캐시의 일관성을 유지한다. SANtopia 시스템은 SAN 환경에서 동작하는 논리 볼륨 관리자와 파일 시스템으로 구성되며 전역 버퍼 관리자는 이 시스템의 성능 향상을 위한 일부분으로 구현되었다.

본 논문의 구성은 다음과 같다. 제2장에서 관련 연구를 소개하고 제3장에서 SANtopia 시스템의 전체 구조를 소개한다. 제4장은 SANtopia 전역 버퍼 관리자의 기본적인 설계 내용을 다루고 제5장에서 캐시 대체 방법과 부하 분산 등의 논의 사항을 다룬다. 제6장은 전역 버퍼 관리자의 구현에 대해 설명하고 제7장에서 성능 측정 결과를 보인다. 마지막으로 제8장에서 앞으로의 연구 방향을 제시하고 결론을 맺는다.

2. 관련 연구

버퍼 캐시는 운영체제의 특정 메모리 영역으로서 물리적인 장치로의 접근 시간을 절약하기 위해 사용된다. 버퍼 캐시에는 디스크와 같은 블록 장치로부터 읽고 쓰는 모든 블록 데이타가 저장된다[3]. 그러나 기존 운영체제의 버퍼 캐시는 단일 호스트에서 사용되도록 고안되었기 때문에 공유 디스크 파일 시스템에서 사용할 때는 여러 호스트간에 캐시 일관성이 깨지는 문제점이 발생한다. 이를 해결하기 위해 SAN 환경의 공유 디스크 파일 시스템인 GFS는 버퍼 블록의 데이타가 수정되면 이를 직접 디스크에 반영하는 방법을 사용한다[4]. 그러나 이는 디스크 I/O로 인한 추가적인 지연 시간을 초래한다.

GFS[5] 이외의 공유 파일 시스템에 관한 연구로는 NFS[6], AFS[7], Sprite[2], xFS[8], Frangipani[9] 등이 있는데 이들 파일 시스템은 한 대 이상의 서버가 네트워크에 연결되어 있는 클라이언트들에게 파일 데이타를 제공하는 형태로 SAN 환경에서의 공유 디스크 파일 시스템과는 다른 구조를 가지고 있다. 또한 이들 연구는 버퍼 캐시를 이용한 메모리 공유를 고려하지 않는다는 점에서 본 연구와 차이가 있다.

GMS(Global Memory System)는 Washington 대학에서 진행하는 프로젝트로 클러스터 시스템에서 전역 메모리 관리 시스템을 구현하였다[11]. 이 연구는 운영체제의 페이지 캐시를 수정하여 분산된 여러 호스트 상의 페이지 캐시를 하나의 캐시처럼 사용할 수 있도록 전역 대체 알고리즘(global replacement algorithm)을 제안하였다. 그러나 이 연구는 수정되지 않은 데이타만을 고려하고 있기 때문에 본 논문의 연구와는 차이가 있다. SANtopia 전역 버퍼 관리자는 버퍼 블록에 수정이 발생하여도 캐시 일관성을 유지하면서 다수의 호스트가 블록 데이타를 공유하는 방법을 제공한다.

그 외에 사용되지 않는 호스트(idle host)의 메모리 영역을 로컬 메모리와 디스크 사이에 존재하는 빠른 중간 저장 장치로 활용하려는 연구[12-14]가 있다. 이들 연구는 커널 레벨의 수정을 통해 다른 호스트의 메모리를 활용하는 연구[12]와 사용자 레벨의 프로그램 인터페이스를 제공하여 명시적으로 다른 호스트의 메모리를 사용하는 연구[14]로 크게 분류된다. 그러나 이들 연구는 단지 리모트 호스트의 사용되지 않는 메모리 영역을 활용하기 위한 것이므로 데이타 공유를 목적으로 하는 본 연구와 차이가 있다.

3. SANtopia 시스템의 구조

SANtopia 시스템은 서로 다른 특성을 갖는 여러 대의 저장 장치들을 연결하여 하나의 논리적인 볼륨으로 보이게 하고 다수의 호스트가 이 대용량 볼륨을 서로 공유할 수 있도록 한다. SANtopia 논리 볼륨 관리자(SVM; SANtopia Logical Volume Manager)와 공유 디스크 파일 시스템(SFS; SANtopia Shared Disk File System)은 이를 위한 주된 기능을 제공하며 그 밖에 전역 잠금 관리자와 버퍼 관리자 그리고 유틸리티 프로그램들이 추가적인 기능을 제공한다. SANtopia 시스템의 전체 구조는 그림 2와 같다.

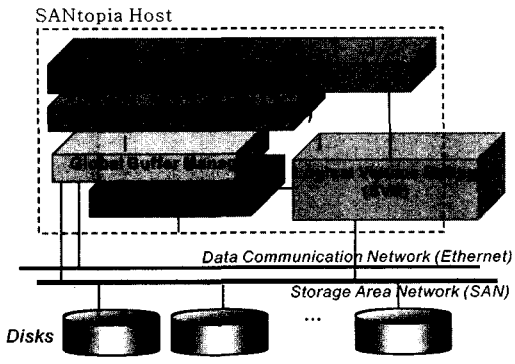


그림 2 SANtopia 시스템의 구성

그림 2에서 여러 대의 SANtopia 호스트들이 SAN과 데이타 통신 네트워크를 통해 연결될 수 있다. SANtopia 논리 볼륨 관리자의 역할은 SAN에 연결된 독립적인 저장 장치들을 하나의 볼륨으로 보이게 하며 다수의 호스트가 이 저장 공간을 서로 공유할 수 있도록 제어 정보를 관리한다. 또한 소프트웨어 레이드(software RAID), 블록 레벨 스냅샷(block-level snapshot), 온라

인 크기 변경(online-resizing) 등의 기능을 제공한다. SANtopia 공유 디스크 파일 시스템은 논리 볼륨 관리자가 생성한 대용량 볼륨에 파일 시스템을 생성할 수 있도록 파일과 디렉터리에 특별한 메타 데이타 구조를 제공한다. 전역 잠금 관리자(Global Lock Manager)는 다수의 호스트가 공유 데이타에 동시에 접근하지 못하도록 상호 배제(mutual exclusion)를 수행한다. 각 호스트에 설치된 전역 잠금 관리자는 잠금 정보를 분할 관리하며 전역 버퍼 관리자와 밀접하게 동작하여 전역 버퍼 캐시의 일관성 유지를 돕는다. 전역 버퍼 관리자는 메모리 상에 존재하는 디스크 블록 캐시인 버퍼 캐시를 SAN에 연결된 다른 호스트들이 사용할 수 있도록 버퍼 캐시 공유 기법을 제공한다. 제4장과 5장에서는 전역 버퍼 관리자의 설계 사항과 알고리즘에 대해 구체적으로 다룬다.

4. SANtopia 전역 버퍼 관리자의 설계

전역 버퍼 관리자는 필요한 블록 데이타가 자신의 버퍼 캐시에 없을 경우, 디스크 I/O를 통하지 않고 다른 호스트의 버퍼 캐시로부터 직접 블록을 전송 받아 사용한다. 이것은 일관성 유지뿐만 아니라 디스크와 SAN 스위치의 접근 횟수를 최소화하여 전체 시스템의 성능을 향상시키기 위함이다. 본 장에서는 전역 버퍼 관리자의 구조와 동시성 제어, 일관성 유지 방법에 관한 구체적인 전역 버퍼 설계 내용을 다룬다.

4.1 전역 버퍼 관리자의 구조

전역 버퍼 관리자는 그림 3과 같이 전역 버퍼 제공자(Global Buffer Provider), 캐시 일관성 관리자(Cache Consistency Manager), 장치 매핑 관리자(Device Mapping Manager), 블록 요청 처리자(Block Request Handler), 버퍼 캐시 공간 관리자(Buffer Cache Space Manager)로 구성된다.

전역 버퍼 제공자는 공유 디스크 파일 시스템으로부터 블록 데이타를 요청 받고 이를 처리한다. 먼저 캐시 일관성 관리자로부터 버퍼 블록의 위치 정보를 얻어 최근에 수정된 버퍼 블록이 자신의 버퍼 캐시에 있는지 찾는다. 만약 다른 호스트의 버퍼 캐시에 최근에 수정된 버퍼 블록이 있다면 블록 요청 처리자를 통해 다른 호스트로부터 버퍼 블록을 전송 받는다. 캐시 일관성 관리자는 SANtopia 전역 잠금 관리자로부터 각 호스트의 잠금 획득 정보를 전달 받아 이를 통해 최근에 수정된 버퍼 블록을 위치를 파악하고 자신이 속한 호스트의 버퍼 캐시에 일관성을 유지하는 일을 담당한다. 이와 관련된 자세한 내용은 4.3절에서 다룬다. 장치 매핑 관리자는 SAN에 연결된 저장 장치에 생성된 논리 볼륨 정보를 관리하는 일을 담당한다. 블록 요청 처리자는 다른 호스트들에게 전역 버퍼 블록을 요청하고 전송 받는 일을 담당한다. 또한 다른 호스트로부터 전역 버퍼 요청을 수락하고 전역 버퍼 제공자로부터 적당한 버퍼 블록을 전달 받아 응답 메시지를 보낸다. 버퍼 캐시 공간 관리자는 자신이 속한 호스트의 버퍼 캐시에 자유 공간이 부족할 경우, 전역 버퍼로 등록된 버퍼 블록들을 메모리에서 해지(free)하는 일을 담당한다. 버퍼 블록을 해지하기 위한 가장 간단한 메모리 대체 방법은 운영체제가 제공하는 LRU(Least Recently Used) 방법을 따르는 것이다. 하지만 보다 복잡한 버퍼 캐시 대체 정책을 사용하여 버퍼 블록의 메모리 적중률(hit rate)을 높일 수 있다. 이와 관련한 구체적인 내용은 5.1절에서 다룬다.

4.2 동시성 제어(Concurrency Control)

SANtopia 시스템은 여러 호스트가 동시에 공유 데이

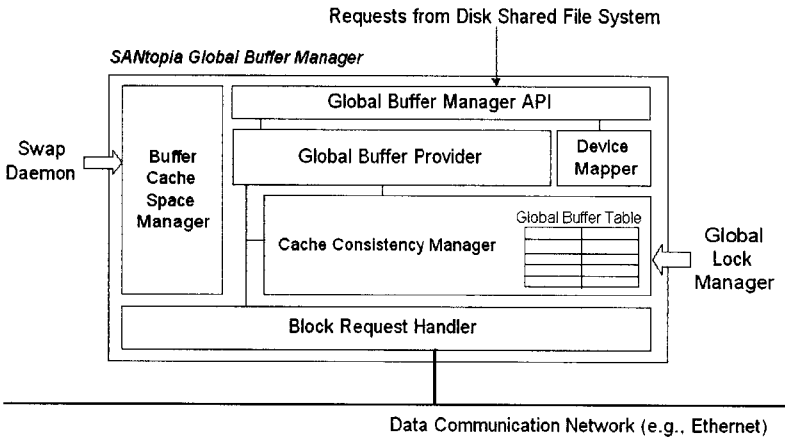


그림 3 SANtopia 전역 버퍼 관리자의 구조

타에 접근하여 일관성을 깨는 것을 방지하기 위해 잠금 방법을 이용한다. 이러한 기능은 그림 2에서 보인 바와 같이 각 호스트의 SANtopia 전역 잠금 관리자가 분할 담당하며 각 호스트들에게 공유 데이타에 대한 사용 권한을 부여한다. 전역 잠금 방법은 이를 사용하지 않는 방법과 비교하여 호스트 간에 통신 등의 추가적인 비용이 필요하지만 데이타의 일관성 유지를 위해 반드시 필요한 작업이므로 다중 호스트 환경에서 이를 위한 오버헤드는 불가피하다. 또한 전역 잠금 관리자에 전역 버퍼 기능을 추가하는 비용은 7.2절의 표 2(c)에 나타낸 바와 같이 상대적으로 작다.

전역 잠금 관리자에서 블록 데이타의 경우, 장치 혹은 볼륨 번호와 블록 번호를 조합하여 공유 데이타에 대한 식별자를 생성한다. 이 식별자를 잠금 객체 식별자(LOID; Lock Object ID)라고 하고 모든 호스트는 이 식별자를 통해 전역 잠금 관리자에게 잠금을 요청한다. 공유 디스크 파일 시스템에서 LOID는 슈퍼 블록, inode 블록, 비트맵 블록 등을 나타내며 각 블록에 접근하기 위해 특정 모드의 잠금 권한을 획득해야 한다.

SANtopia는 콜백(callback)[15][16] 방식의 잠금 기법을 사용한다. 콜백 방식은 다른 호스트로부터 잠금 요청을 받기 전까지 잠금을 소유하고 계속 사용하는 것이다. 이 방법은 한 호스트가 연속해서 동일한 모드로 버퍼 블록을 사용할 경우, 전역 잠금 관리자로부터 잠금을 여러 번 획득하지 않아도 되므로 잠금 요청 횟수를 줄여서 네트워크 트래픽 양을 감소시킬 수 있다. 그림 4는 한 호스트가 특정 LOID X에 해당하는 잠금을 전역 잠금 관리자로부터 획득하여 사용하고 반납하는 과정을 나타낸다.

잠금을 획득한 호스트는 해당 버퍼 블록의 사용 권한을 가진다. 호스트가 사용 권한을 획득한 버퍼 블록은 잠금이 회수되기 전까지 메모리에서 계속 재사용될 수 있으므로 메시지 전송량이나 디스크 I/O 횟수를 감소시킬 수 있다.

버퍼 블록 사용과 관련하여 전역 잠금 관리자는 공유 모드(Shared Mode)와 배타 모드(Exclusive Mode)의 두 가지 모드를 제공한다. 두 가지 모드의 차이점은 읽기 동작에 대해서는 공유 모드를 사용하여 다수의 호스트가 사용할 수 있고 쓰기 동작에 대해서는 배타 모드를 사용하여 단지 하나의 호스트가 쓰기 동작을 수행할 수 있다. 이러한 공유 혹은 배타 모드로의 전환은 전역 잠금 관리자가 제어한다.

그림 5는 특정 inode 블록에 대해 배타 모드의 잠금을 획득하고 해당 블록 데이타를 수정하는 간단한 예를 보여준다. 모든 호스트는 블록 데이타를 버퍼 캐시에 읽어 사용하기 위해서 이와 같은 절차를 수행해야 한다. SANtopia는 파일에 접근하기 위해서 먼저 inode 블록에 대한 잠금을 획득하기 때문에 파일 단위로 동시성 제어를 수행하게 된다. 또한 슈퍼 블록, 비트맵 블록과 같은 기타 메타 데이타들은 메타 데이타가 저장된 블록 단위로 동시성 제어가 수행된다.

```
If(santopia_lock(inode_num, exclusive_mode) == SUCCESS)
    global_buffer_read(device_num, block_num, block_size);
    /* modify block data */
    santopia_unlock(inode_num);
```

그림 5 배타 모드로 inode 블록을 수정하는 예제

파일 시스템은 여러 가지 데이타 블록이나 제어 변수 등의 리소스에 대해서 해당 LOID로 잠금을 요청하고 사용 권한을 얻어 사용하는데 이때, 호스트 간에 잠금 요청이 데드락(deadlock)을 발생시킬 수 있으므로 파일

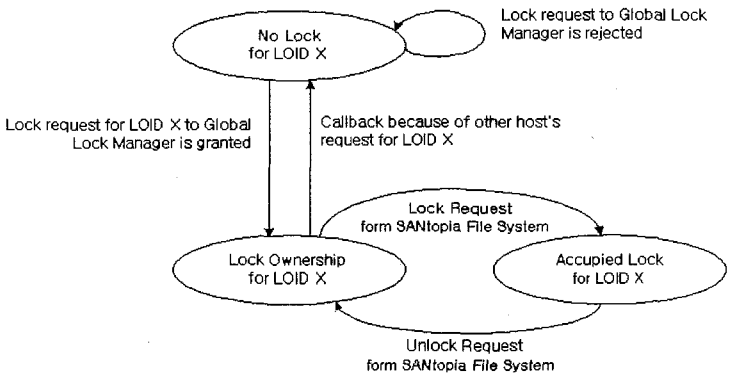


그림 4 전역 잠금의 획득과 반납 과정

시스템은 데드락이 발생하면 타임 아웃(time-out) 등을 통하여 이를 탐지하고 회복해야 한다. 이와 관련된 파일 시스템의 데드락 방지 방법은 본 논문의 범위를 벗어나므로 구체적으로 언급하지 않겠다.

4.3 일관성(Consistency)

전역 버퍼 관리자는 SAN에 연결된 모든 호스트가 가장 최근에 수정된 블록 데이타만을 사용한다는 것을 보장함으로써 전역 버퍼 캐시에 대한 일관성을 유지한다. 이를 위해 다음과 같은 방법이 사용된다. 잠금을 획득한 호스트는 바로 이전에 동일한 LOID로 잠금을 획득했던 호스트로부터 잠금과 관련된 버퍼 블록을 전송받는다. 잠금과 관련된 정보는 SANtopia 전역 잠금 관리자로부터 얻을 수 있다. 공유 모드로 잠금을 획득한 호스트는 현재 공유 모드로 동일한 잠금을 소유하고 있는 다른 호스트들에게 버퍼 블록을 전송 받을 수 있다. 또한 특정 호스트가 배타 모드로 잠금을 획득하고 버퍼 블록을 요청하면 버퍼 블록을 전송해 주고 즉시 버퍼 캐시에서 전송한 버퍼 블록을 무효화 시킨다. 이는 다른 호스트가 이미 수정한 버퍼 블록(stale buffer block)이 재 사용되는 것을 방지하기 위한 것이다. 그림 6은 잠금과 관련하여 버퍼 블록의 일관성을 유지하기 위한 방법을 요약한 것이다. 디스크 블록은 공유 모드 혹은 배타 모드로 잠금을 획득한 호스트에 의해서 버퍼 캐시로 복사되며 잠금 모드에 해당하는 권한을 가지고 사용되다가 LRU와 같은 대체 알고리즘에 의해서 버퍼 캐시에서 삭제 된다. 이와 같이 SANtopia 버퍼 관리자는 잠금 방법을 통해서 버퍼 캐시의 일관성을 보장한다.

그림 7은 전역 잠금 관리자와 전역 버퍼 관리자 그리고 공유 디스크 파일 시스템 간의 상호 작용을 나타낸다. 공유 디스크 파일 시스템의 요청에 의해서 특정 호스트가 잠금을 획득하고 회수(callback)되는 시점에

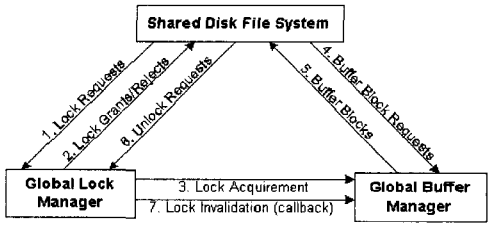


그림 7 전역 잠금 및 버퍼 관리자와 파일 시스템의 상호 작용

전역 잠금 관리자는 전역 버퍼 관리자에게 이러한 이벤트를 전달한다(과정 1,2). 제 4.2절에서 언급한 바와 같이, 공유 디스크 파일 시스템은 공유 블록 데이타에 접근하기 전에 전역 버퍼 관리자에게 잠금을 요청하고 이를 수락 받으면(과정 3,4) 전역 버퍼 관리자를 통해 버퍼 블록을 메모리 캐시로 가져와 읽기/쓰기를 수행한다(과정 5,6). 마지막으로 블록 데이타에 대한 사용을 마치면 공유 디스크 파일 시스템은 잠금을 푼다(과정 7). 공유 디스크 파일 시스템이 잠금을 풀어야 전역 잠금 관리자는 특정 호스트가 소유한 잠금을 회수(callback) 할 수 있다.

5. 논의 사항

본 장에서는 전역 버퍼 관리자에 대한 여러 가지 논의 사항들 중에서 버퍼 캐시 대체 방법과 호스트의 부하에 따라 버퍼 전송 요구를 분산시키는 방법에 대해 다룬다. 또한 버퍼 블록의 선인출과 호스트의 고장에 대한 대처 방법에 대해 논의한다.

5.1 캐시 대체 기법(Cache Replacement)

전역 버퍼 관리자의 사용 효과를 높이기 위해서는 여러 호스트 중 적어도 하나의 버퍼 캐시에 특정 블록 데

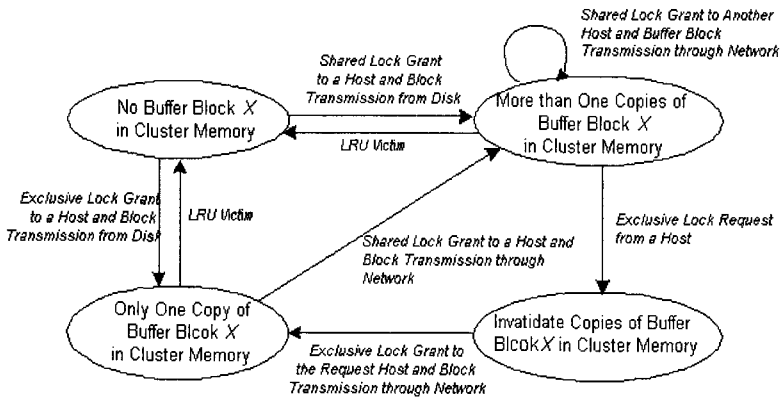


그림 6 잠금을 통한 버퍼 블록의 일관성 관리

이타의 복사본이 존재하여 전역 버퍼 요구에 응답해야 한다. 그러나 자유 메모리 공간이 부족한 경우, 버퍼 블록의 해제(free)와 생성(allocation)이 빈번하게 발생하여 필요한 버퍼 블록이 메모리에서 삭제될 수 있다. 이러한 경우, 디스크 I/O를 통해 해당 블록 데이타를 다시 버퍼 캐시로 읽어와야 하므로 지연 시간이 증가한다. 이를 해결하기 위해 각 LOID에 대한 주 버퍼 소유 호스트(Primary Buffer Holding Host)를 지정하고 이 호스트의 버퍼 캐시에서는 가능한 지정된 LOID에 해당하는 버퍼 블록이 대체 되지 않도록 한다. 이러한 버퍼 블록을 주 버퍼 블록(Primary Buffer Block)이라 한다. LRU(Least Recently Used) 대체 알고리즘에 이와 같은 우선 순위를 추가하여 표 1과 같은 버퍼 블록 대체에 대한 우선 순위를 얻을 수 있다.

표 1 버퍼 블록의 대체 우선 순위

Buffer Cache Replacement Priority	Buffer Blocks
1	Primary and Recently Used Buffer Blocks
2	Normal and Recently Used Buffer Blocks
3	Primary and Least Recently Used Buffer Blocks
4	Normal and Least Recently Used Buffer Blocks

버퍼 블록의 대체가 필요한 경우, 마지막 접근 시간이 가장 오래되고 주 버퍼 블록이 아닌 블록을 메모리에서 먼저 대체하고 마지막 접근 시간이 가장 최근인 주 버퍼 블록은 가능한 메모리에 남아 있도록 높은 우선 순위를 부여한다. 주 버퍼 블록이지만 오랫동안 사용되지 않은 블록은 최근에 사용된 일반 버퍼 블록들을 위해서 비교적 낮은 대체 우선 순위를 갖는다.

5.2 부하 분산(Load Distribution)

전역 버퍼 관리자는 자신의 버퍼 캐시에 원하는 버퍼 블록이 존재하지 않으면 적당한 호스트를 선택하여 버퍼 블록 전송 요청을 보낸다. 이때, 전송 요청이 특정 호스트에게 집중되면 버퍼 전송으로 인한 호스트 부하가 증가하게 된다. 예를 들어, 호스트 ID가 작은 순서로 버퍼 블록 전송 요청을 보내게 되면 ID가 작은 몇 대의 호스트에 부하가 집중되는 현상이 발생한다. 따라서 적당한 호스트를 선택하여 전역 버퍼 요청을 보내야 한다. SANtopia는 두 가지 방법을 제안하고 있다. 첫 번째 방법은 버퍼 블록을 소유한 호스트들 중 하나를 임의로 선택(random selection)하여 전송 요청을 보내는 것이다. 이 방법은 간단하고 구현이 쉽지만 시간에 따른 각 호스트의 부하 정보를 반영하지 못한다는 단점이 있다. 두 번째 방법은 버퍼 블록을 소유한 호스트들 중 부하가 가장 적은 호스트에게 전송 요청을 보내는 것이다. 이 방법은 각 호스트의 부하를 측정하기 위한 오버헤드가 추가로 필요하지만 보다 효과적으로 부하를 분산할 수 있다. 각 호스트의 부하는 식 (1)과 같이 계산된다. 총 호스트 부하는 공유 디스크 파일 시스템을 운용하는데 필요한 원래의 호스트 부하(Load $_{Original}$)에 전역 버퍼 요구를 처리하는데 필요한 부하(Load $_{Global\_Buffer}$)를 더한 값이며 각 부하 요소는 시간 비용으로 계산된다.

$$Host\_Load = Load_{Original} + Load_{Global\_Buffer} \quad (1)$$

공유 디스크 파일 시스템을 운용하기 위해 필요한 호스트 부하는 잠금 요청 빈도수(N_{Lock})를 가지고 계산할 수 있다. 잠금 요청 빈도수는 최근 일정 기간 동안 측정된 잠금 요청 횟수이다. 잠금 요청 빈도수는 공유 디스크 파일 시스템의 데이타 사용 빈도를 나타내므로 잠금 요청이 빈번해질수록 호스트의 부하는 높아진다. 즉, 잠금 요청 빈도수와 호스트 부하는 비례한다. 식 (2)는 Load $_{Original}$을 계산하는 식이며 상수 C_P는 잠금을 획득하고 푸는 사이의 평균 처리 시간(average processing time)을 나타낸다.

$$Load_{Original} = C_P \times N_{Lock} \quad (2)$$

$$C_P = \frac{Average\ Processing\ Time\ per\ System\ Call}{Average\ Number\ of\ Locks\ per\ System\ Call} \quad (3)$$

전역 버퍼 요구를 처리하는데 필요한 부하는 식 (4)와 같이 계산된다. N_{GLOBAL_HIT}는 호스트가 최근 일정 기간 동안 받은 전역 버퍼 요구 중에서 요청된 버퍼 블록이 버퍼 캐시에 존재하는 경우의 횟수를 나타내고 N_{GLOBAL_MISS}는 버퍼 캐시에 존재하지 않는 경우의 횟수를 나타낸다. 상수 C_{TP}는 버퍼 블록을 버퍼 캐시에서 찾아 다른 호스트로 전송하는데 걸리는 평균 처리 시간을 나타내고 상수 C_{TN}는 버퍼 블록 전송 요청이 실패했을 경우에 이에 대한 응답 메시지를 보내는데 필요한 평균 처리 시간을 나타낸다.

$$Load_{Global\ Buffer} = C_{TP} \times N_{GLOBAL\_HIT} + C_{TN} \times N_{GLOBAL\_MISS} \quad (4)$$

좀 더 능동적인 부하 분산을 위해서 부하가 적은 호스트로 버퍼 블록들을 이동시키는 방법(Buffer Block Migration)을 고려할 수 있다. SANtopia는 현재 부하 분산을 전담하는 앞단 서버(Front-end Server)가 없기 때문에 특정 호스트에 부하가 집중되는 현상을 막기 어렵다. 따라서 부하가 적은 호스트로 버퍼 블록들을 이동시켜서 전역 버퍼 요구를 분산할 수 있다. 또한, 부하가 적은 호스트를 제 5.1절에서 언급한 주 버퍼 소유 호스트로 지정하여 버퍼 블록 요청에 대한 전역 버퍼 캐시 적중률과 부하 분산 효과를 동시에 높일 수 있다.

5.3 선인출(Prefetching)

호스트가 블록 데이타를 다른 호스트의 버퍼 캐시로부터 혹은 디스크로부터 로컬 버퍼 캐시로 가져오는 시기에 따라 두 가지 방법이 존재한다. 첫 번째는 버퍼 블록이 요청된 시점에 가져오는 것이고 두 번째는 전역 잠금 관리자로부터 잠금을 획득함과 동시에 관련된 블록 데이타를 가져오는 것이다. 두 번째 방법은 실제로 버퍼 블록이 요청되기 전에 필요한 블록 데이타를 버퍼 캐시로 읽어오는 것으로 버퍼 블록 요청에 대한 로컬 버퍼 캐시 적중률을 높일 수 있다. SANtopia 전역 버퍼 관리자는 선인출을 하지 않는 첫 번째 방법을 사용하지만 차후에 선인출 방법도 고려될 예정이다.

5.4 회복(Recovery)

수정된 버퍼 블록이 디스크에 반영되지 않고 버퍼 캐시에서 계속 사용되다가 고장이 발생하면 수정된 데이타를 잃게 된다. SANtopia는 잠금 회수 시에 관련된 버퍼 블록들을 디스크에 반영하므로 잠금 회수 이후의 데이타는 보전되지만 이전의 데이타는 잃게 될 가능성이 있다. SANtopia 공유 디스크 파일 시스템은 파일 동작의 결과가 디스크에 쓰여지지 않고 고장이 발생하는 경우를 대비하여 시스템 호출 단위로 저널링(Journaling)을 수행한다[10]. 전역 잠금 관리자는 SAN 클러스터 내의 각 호스트들을 주기적으로 검사하여 고장 발생 여부를 알아내고 고장 발생 즉시 저널링의 회복 루틴을 수행 시킨다. 파일 연산에 대한 결과로 수정된 버퍼 블록이 디스크에 반영되지 않은 채로 고장이 발생하면 디스크 상의 저널 공간에 기록된 메타 데이타를 이용하여 회복을 수행한다.

6. 구 현

SANtopia는 리눅스 커널 상에 구현되며 전역 버퍼 관리자는 커널에 추가되는 모듈의 형태로 제공된다. 그림 8은 전역 버퍼 관리자를 추가한 커널 구조를 나타낸다. 전역 버퍼 기능을 사용하기 위해서는 반드시 SANtopia 전역 버퍼 관리자를 통해 버퍼 블록을 요청해야 한다. 그러나 전역 버퍼 관리자를 통하지 않고 커널이 제공하는 일반적인 방법으로 버퍼 블록을 읽으면 기존의 버퍼 정책을 그대로 따를 수 있다. SANtopia 전역 버퍼 캐시는 커널의 버퍼 캐시 내에 포함되어 있고 전역 버퍼 캐시 내의 버퍼 블록들은 커널이 제공하는 버퍼 헤더에 추가하여 전역 버퍼 관리를 위한 확장된 헤더를 가지고 있다. 전역 버퍼 관리자는 전역 버퍼 캐시에 속한 버퍼 블록들을 해쉬 리스트로 관리한다.

앞서 제 5장에서 전역 버퍼 관리자에 관한 여러 가지 논의 사항들을 다뤘지만 실제 구현은 다음과 같다. 전역 버퍼 캐시의 대체 정책은 커널이 제공하는 LRU(Least Recently Used) 방법을 따르도록 스왑 데몬(swap daemon)이 호출하는 버퍼 해지 함수를 수정하였다. 또한 부하 분산 방법은 버퍼 블록을 소유한 호스트들 중 하나를 임의 선택하는 방법을 채택한다.

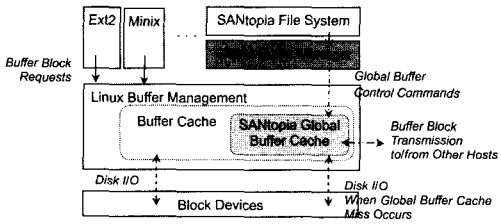


그림 8 SANtopia 전역 버퍼 관리자를 추가한 커널 구조

7. 성능 측정

본 장에서는 전역 버퍼 관리자의 성능 측정 결과와 분석 내용을 보이고자 한다.

7.1 실험 환경

실험은 Pentium III 700MHz SMP 서버 4대에서 수행하였고 각 호스트 당 메모리의 크기는 256MByte이다. OS는 리눅스 커널 2.2.18버전을 사용한다. 각 호스트와 FC 디스크는 SAN 스위치로 연결하고 호스트간에 버퍼 블록과 잠금 메시지 전송을 위해 사용되는 데이타 통신 네트워크는 Gigabit Ethernet 스위치로 연결한다. SAN으로 연결된 FC 디스크의 컨트롤러는 128MByte의 메모리가 장착되어 있다.

7.2 성능 측정

7.2.1 마이크로 벤치마크(Microbenchmarks)

마이크로 벤치마크 실험은 전역 버퍼 관리자를 통한 블록 읽기 동작의 수행 시간을 분석하고 전역 버퍼 관리자와 디스크 I/O의 버퍼 블록 읽기 동작의 수행 시간을 비교한다. 표 2는 1GByte 데이타를 4KByte의 블록 단위로 버퍼 캐시에 읽었을 때, 전역 버퍼 관리자의 평균 수행 시간을 항목별로 나타낸 것이다.

전역 버퍼 관리자의 세부 동작을 크게 세 가지로 나타낼 수 있는데 첫 번째는 다른 호스트의 버퍼 캐시로부터 혹은 디스크로부터 로컬 버퍼 캐시로 블록 데이타를 읽는 동작, 두 번째는 다른 호스트로부터 전역 버퍼 요청을 받아 이를 처리하고 응답 메시지를 보내는 동작, 세 번째는 전역 버퍼 관리를 위한 정보를 유지하는 것이다. 표 2의 (a), (b), (c)는 이 세가지 동작의 수행 시간을 각각 나타낸 것이다. 표 2(a)에서 블록 읽기 동작은 다음과 같은 네 가지 경우가 존재한다. 첫째, 로컬 버퍼 캐시에 요청된 블록 데이타가 있는 경우, 둘

째, 다른 호스트의 버퍼 캐시에 요청된 블록 데이타가 있어 전역 버퍼 전송이 필요한 경우, 셋째, 다른 호스트로 전역 버퍼 전송을 요청했지만 버퍼 블록이 존재하지 않아 디스크 I/O를 수행한 경우(Global Buffer Cache Miss), 넷째, 모든 호스트의 버퍼 캐시에 요청된 블록 데이타가 존재하지 않아 직접 디스크 I/O를 수행한 경우(No Global Buffer Blocks)이다. 표 2(a)의 각 열은 이러한 네 가지 경우의 수행 시간을 보여준다. 전역 버퍼 관리자는 로컬 버퍼 캐시에서 전역 버퍼로 등록된 블록이 삭제되면 이 정보를 즉시 다른 호스트에게 알리지 않는다. 따라서 세 번째와 같이 전역 버퍼 요청에 실패하는 경우(Global Buffer Cache Miss)가 발생할 수 있다. 이 경우, 다른 호스트에게 전역 버퍼 전송 요청을 받는 시점에 실패 응답 메시지를 보냄으로써 버퍼 블록이 이미 메모리에서 삭제되었음을 알린다. 이러한 방법을 사용해서 전역 버퍼 정보를 유지하는데 필요한 호스트간 메시지 전송량을 줄일 수 있다. 표 2(b)는 전역 버퍼 관리자가 다른 호스트를 위하여 전역 버퍼 서비스를 수행하는데 필요한 시간을 분석한 것이며 표 2(c)는 전역 잠금 혹은 해제가 발생했을 때 전역 버퍼 관리자가 이에 따른 관리 정보를 변경하는데 필요한 시간을 나타낸다.

그림 9는 블록의 크기를 변화시키면서 전역 버퍼 관리자와 디스크 I/O를 통한 블록 읽기 동작의 처리 시간을 측정한 그래프이다. 두 방법 모두, 블록 크기에 따른 처리 시간 차이는 크지 않다. 그러나 전역 버퍼 관리자의 성능은 디스크 I/O를 수행하는 것에 비해 약 1.8-12.8배 빠르다. 디스크 컨트롤러 상에도 일정 크기의 메모리가 존재하기 때문에 디스크 메모리에 저장된 블록 데이타를 읽는 것(warm start)이 디스크에서 직접 읽는 것(cold start)보다 빠르다. 그러나 그림 9에서 보는 바와 같이 전역 버퍼 관리자를 통한 호스트간 데이타 전송이 디스크 메모리에서 블록을 읽는 것보다 약 1.8배 정도 빠르다.

그림 10은 전역 버퍼 요청의 실패(Global Buffer Cache Miss)가 블록 읽기 시간에 미치는 영향을 나타낸 것이다. 이 그래프는 전역 버퍼 요청에 실패한 이후에 다른 호스트로부터 버퍼 블록을 전송 받았을 경우

표 2 블록 읽기의 수행 시간 분석 (4Kbyte Block Size)

(a) Block Read (μsec)

	Local Buffer Cache	Global Buffer Cache	Disk	
			Global Buffer Cache Miss	No Global Buffer Blocks
Get Buffer Block & Check It's Validity	1.04	1.04	1.04	1.04
Check Global Buffer Table Whether Buffer is in Other Hosts	-	1.0	1.0	1.0
Send Global Buffer Request	-	15.63	15.63	-
Receive Nack Msg.	-	-	271.57	-
Receive Buffer Block	-	411.17	-	-
Read Block from Disk	-	-	5798.58	5798.58
Update Buffer Block Data & Global Buffer Table	-	2.57	2.57	2.57
Total	**1.04**	**431.41**	**6090.39**	**5803.19**

(b) Global Buffer Service (μsec)

	Global Buffer Miss	Global Buffer Hit
Search Requested Buffer Block in Global Buffer Cache	1.59	1.59
Send Nack Msg.	9.88	-
Copy Buffer Block from Global Buffer Cache	-	20.61
Send Ack Msg. and Buffer Block	-	39.79
Total	**11.47**	**61.99**

(c) Maintenance of Global Buffer Information (μsec)

	Lock Acquisition	Lock Invalidation (Callback)
Update Global Buffer Table	1	2.21
Total	**1**	**2.21**

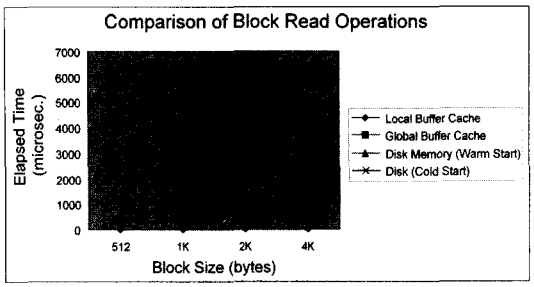


그림 9 블록 크기에 따른 블록 읽기 동작의 처리 시간

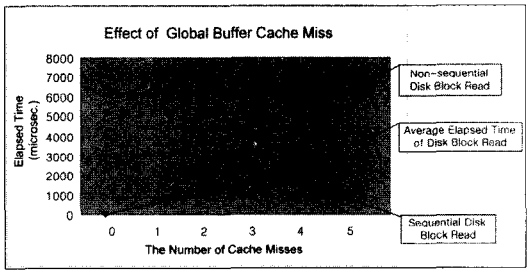


그림 10 전역 버퍼 요청의 실패에 따른 지연 시간 (4Kbyte Block Size)

(Global Buffer Cache Hit)의 지연 시간과 디스크 I/O 시간을 비교하였다. x-축은 전역 버퍼 요청이 실패한 횟수이고 y-축은 지연 시간을 나타낸다. x-축의 0은 로컬 버퍼 캐시에 원하는 버퍼 블록이 존재하는 경우를 나타낸다.

디스크 블록을 순차적으로 읽었을 경우와 그렇지 않을 경우 그리고 그들의 평균 시간을 전역 버퍼 관리자의 블록 읽기 시간과 비교하였는데, 그 이유는 몇 개의 디스크 블록을 순차적으로 읽으면 나머지 블록들을 디스크 메모리에 미리 읽도록 되어 있기 때문에 순차 블록 읽기는 디스크 메모리로부터 블록을 읽는 시간과 비슷하다. 그러나 블록 사이의 간격을 띄워서 읽으면 미리 읽기(read ahead)가 수행되지 않기 때문에 수행 시간이 길어진다. 그래프가 나타낸 바와 같이 전역 버퍼 요청 실패가 2회 이상 발생하면 디스크 메모리에서 버퍼 블록을 전송 받는 것이 빠르지만 미리 읽기가 수행되지 않는 경우는 전역 버퍼 요청 실패가 4회 이상 발생하여도 전역 버퍼 관리자를 이용하는 것이 디스크 I/O보다 빠르다.

파일 시스템 벤치마크 (File System Benchmarks)

표 3은 SANtopia 공유 디스크 파일 시스템이 전역 버퍼 관리자와 함께 수행될 경우와 디스크 I/O를 통해 일관성을 유지하는 경우의 파일 시스템 콜(File System Call) 단위의 수행 시간을 측정한 것이다. 표 3(a)은

표 3 파일 시스템 콜의 수행 시간 비교

(a) Directory Related System Call (μsec)

File System Call	Disk Only	Global Buffer Manager
mkdir	7334.23	7089.89
opendir	2090.03	1631.92
readdir	217.94	225.69
closedir	5.58	5.60
rmdir	8030.42	7707.40
Average	3535.64	3332.10

(b) File Related System Call (μsec)

File System Call	Disk Only	Global Buffer Manager
open	1933.52	1568.29
read	111.50	113.30
write	5708.67	5097.00
seek	3.13	3.17
close	5.89	6.38
Average	1552.54	1357.63

디렉터리 관련 파일 시스템 콜을 나타내고 (b)는 일반 파일 관련 시스템 콜을 나타낸다.

전역 버퍼 관리자는 디렉터리 관련 시스템 콜 중 opendir의 수행 시에 약 1.3배 정도 빠르고 일반 파일의 경우 open의 수행 시에 약 1.2배 정도 빠르다. 디렉터리의 읽기(readdir), 닫기(closedir) 혹은 일반 파일의 읽기(read), 쓰기(write) 등은 열기를 위한 파일 시스템 콜을 먼저 수행하여 관련 버퍼 블록들이 이미 로컬 버퍼 캐시에 있는 경우가 많으므로 수행 시간에 큰 차이가 없다. 평균적으로 디렉터리 관련 파일 시스템 콜은 약 1.06배 정도 빠르고 일반 파일시스템 콜은 약 1.14배 정도 빠르다.

8. 결 론

SANtopia 시스템은 다수의 호스트가 SAN에 연결된 디스크 상의 데이타를 공유하도록 설계되었다. 이러한 환경에서 서로 다른 호스트가 동시에 같은 파일에 대한 읽기/쓰기를 수행하면 공유 데이타에 대한 일관성이 깨지게 되므로 특별한 일관성 유지 방법이 필요하다. 이를 위한 한가지 방법으로 데이타가 수정되면 바로 디스크에 반영하고 다른 호스트들은 변경된 데이타를 디스크로부터 읽도록 할 수 있다. 그러나 이러한 방법은 빈번한 디스크 I/O를 발생시켜 시스템의 성능을 저하 시키는 요인이 된다.

SANtopia 전역 버퍼 관리자는 SAN에 연결된 다수의 호스트가 버퍼 캐시를 서로 공유할 수 있도록 함으로써 디스크 I/O 횟수를 감소시키고 블록 데이타의 접근 속도를 향상시킨다. 4Kbyte 크기의 블록을 읽을 때, 디스크 I/O를 수행하는 것에 비해 SANtopia 전역 버퍼

관리자를 사용하는 것이 약 1.8-12.8배 정도 빠른 성능을 보였다. 또한 파일 시스템의 벤치마킹을 수행한 결과, open 파일 시스템 콜의 경우에 전역 버퍼 관리자가 1.2-1.3배 정도 빠른 성능을 보였다.

앞으로 SANtopia 시스템은 클러스터 내의 호스트 간에 버퍼 캐시뿐만 아니라 페이지 캐시를 공유하는 방법을 고려하고 있으며 공유 데이타를 보다 빠르게 전송하기 위해 커널 레벨에서 VIA(Virtual Interface Architecture)와 같은 통신 프로토콜을 적용할 계획이다.

참 고 문 헌

[1] D. Sacks, "Demystifying DAS, SAN, NAS, NAS Gateways, Fibre Channel and iSCSI," IBM Storage Networking, March 2001.

[2] M. N. Nelson, B. B. Welch and J. K. Ousterhout, "Caching in the Sprite network file system," ACM Transactions on Computer Systems, 6(1), pages 134-154, February 1988.

[3] David A. Rusling, "The Linux Kernel," p.131, http://www.tldp.org/LDP/tlk/tlk.html, 1999.

[4] Jonathan Brassow and David Teighland, "Buffer Caching in the Global File System," EE8362 Project Report, 1998.

[5] S. R. Soltis, T. M. Ruwart, M. T. O'Keefe, "The Global File System," Proceedings of the Fifth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies, September 1996.

[6] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network Filesystem," Proceedings of the Summer 1985 USENIX Conference, pages 119-130 June 1985.

[7] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and Michael J. West, "Scale and Performance in a Distributed File System," ACM Transactions on Computer Systems, 6(1), pages 51-81, February 1988.

[8] T. E. Anderson, M. D. Dahlin, J. M. Neefe, D. A. Patterson, D. S. Roselli and R. Y. Wang, "Serverless network file systems," In Proceedings of the 15th Symposium on Operating Systems Principles, pages 109-126, December 1995.

[9] C. A. Thekkath, T. Mann and E. K. Lee, "Frangipani: A scalable distributed file system," In Proceedings of 16th ACM Symposium on Operating Systems Principles, pages 224-237, October 1997.

[10] 박춘서, 김경배, 이용주, 박선영, 신범주, "SAN 환경에서의 전역 버퍼를 이용한 효율적인 회복 기법", 한국정보처리학회 논문지 A 제8-A권, 2001.12.

[11] M. J. Feeley, W. E. Morgan, F. H. Pighin, A. R. Karlin, H. M. Levy, and C. A. Thekkath, "Implementing Global Memory Management in a Workstation Cluster," In Proc. of the 15th ACM Symposium on Operating Systems Principles, pages 201-212, December 1995.

[12] E. P. Markatos and G. Dramitinos, "Implementation of a Reliable Remote Memory Pager," In Proceedings of the USENIX 1996 Annual Technical Conference, pages 177-189, January 1996.

[13] A. Acharya and S. Setia, "Availability and utility of idle memory in workstation clusters," Proceedings of ACM SIGMETRICS Conference on Measuring and Modeling of Computer Systems, pages 35-46, May 1999.

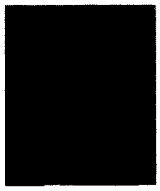
[14] S. Koussih, A. Acharya, and S. Setia, "Dodo: A User-Level System for Exploiting Idle Memory in Workstation Clusters," Technical Report TRCS98-35, University of California, Santa Barbara, December 1998.

[15] J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham, and M. West, "Scale and Performance in a Distributed File System," ACM Transactions on Computer Systems, Vol. 6, No. 1, pp. 51-81, February 1988.

[16] K. Amiri, G. A. Gibson, and R. Golding, "Scalable concurrency control and recovery for shared storage arrays," Technical Report CMU--CS--99--111, Dept. of Computer Science, Carnegie-Mellon Univ., 1999.

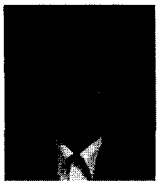
박 선 영

1999년 충남대학교 컴퓨터공학과 학사 2001년 한국과학기술원 전산학과 석사 2001년~현재 한국전자통신연구원 컴퓨터시스템연구부 연구원. 관심분야는 Clustering, Web Server System, Storage Networking



손 덕 주

1976년 서울대학교 수학교육과(학사). 1978년 한국과학기술원 전산학과(석사). 1978년~현재 한국전자통신연구원 인터넷컴퓨팅연구부 책임연구원. 관심분야는 이동컴퓨팅, 분산시스템, 네트웍 자료저장 시스템



신 범 주

1976년~1983년 경북대학교 전자공학과 (학사). 1989년~1991년 경북대학교 컴퓨터공학과(석사). 1919년~1998년 경북대학교 컴퓨터공학과(박사). 1987년~2002년 한국전자통신연구원(책임연구원, 시스템소프트웨어연구팀장). 2002년~현재 밀양대학교 컴퓨터공학부 교수. 관심분야는 분산시스템, 고장감내 미들웨어, 시스템 S/W 등

김 학 영

1983년 경북대학교 전자공학과 전자계산전공 학사. 1985년 경북대학교 대학원 전자공학과 전자계산전공 공학석사. 2003년 충남대학교 대학원 컴퓨터공학과 공학박사. 1988년~현재 한국전자통신연구원 컴퓨터시스템연구부 책임연구원. 관심분야는 인터넷 서버, 컨텐츠 스트리밍, 컨텐츠 분배, 시스템 소프트웨어, 분산파일시스템, GRID

김 명 준

1978년 서울대학교 계산통계학과 학사 1980년 한국과학기술원 전산학과 이학석사. 1986년 프랑스 Nancy 제1대학교 응용수학 및 전산학과 이학박사. 1980년~1986년 아주대학교 종합연구소 연구원 1981년~1986년 프랑스 Nancy 전산학연구소(CRIN) 연구원. 1993년 프랑스 Univ. of Nice Sophia-Antipolis 방문교수. 1986년~현재 한국전자통신연구원 컴퓨터소프트웨어 연구소 책임연구원. 관심분야는 데이타베이스, 분산시스템, 인터넷서비스