論文
5-1-11

# 클러스터공유파일시스템의전역버퍼관리기설계 (DesignofGlobalBufferManagerinClusterSharedFileSyste)

이 규 웅(( Kyu Woong Lee)[1] 차 영환(Tscha, Yeong Hwa)[2]

## 요 약

네트워크 시스템의 빠른 발전으로 인해 인터넷을 통한 대용량 멀티미디어 데이터가 폭발적으로 증가하고 있으며 대용량 데이터의 효율적인 저장 및 제공을 위해 기존 컴퓨팅 중심으 저장 방식이 아닌 데이터 중심의 새로운 저장 시스템 패러다임이 요구되고 있다. 또한 저장장치 전용 네트워크인 SAN(Storage Area Network)과NAS(network attached storage)의 보급으로 인해 클러스터 공유 파일 시스템의 요구가 증가하고 있다.

본 논문은 SAN 기반의 네트워크-부착형 저장 장치들을 군집화하여 파일 서버 없이 직접 데이터 전송이 가능한 클러스터 공유 파일 시스템인 SANiqueTM의 설계방법 및 각 주요 구성요소들의 기능을 기술한다. 특히 효율적인 데이터 접근을 극대화하기 위한 전역 버퍼 관리기의 설계를 제안하며 그 프로토콜에 대하여 기술한다.

## ABSTRACT

As the dependency to network system and demands of efficient storage systems rapidly grows in every networking filed, the current trends initiatedby explosive networked data grow due to the wide-spread of internet multimedia data and internet requires a paradigm shift from computing-centric to data-centric in storagesystems. Furthermore, the new environment of file systems such as NAS(Network Attached Storage) and SAN(Storage Area Network) is adopted to the existing storage paradigm for providing high availability and efficient data access.

We describe the design issues and system components of SANiqueTM, which is the cluster file system based on SAN environment. SANiqueTM has the capability of transferring the user data from the network-attached SAN disk to client applications directly. We, especially, present the protocol and functionality of the global buffer manager in our cluster file system.

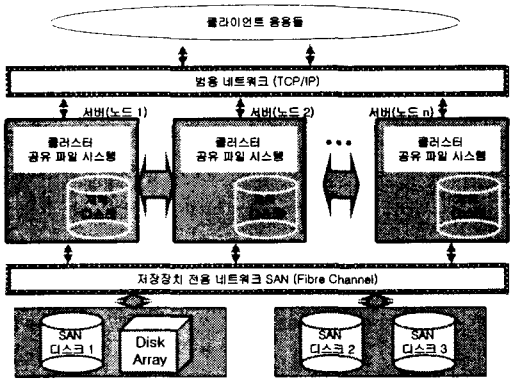
1) 정회원 : 상지대학교 컴퓨터정보공학부 조교수
2) 정회원 : 상지대학교 컴퓨터정보공학부 부교수

논문접수 : 2004. 1. 9.
심사완료 : 2004. 1. 15.

* 이 논문은 2002년도 상지대학교 연구비 지원에 의한 것임

## 1. 서론

네트워크의 발전으로 인하여 파일 시스템의 구조는 점차 서버들의 클러스터링을 이용한 분산 파일 시스템 및 공유 파일 시스템으로 발전하고 있다. 전형적인 분산 파일 시스템 모형은 클라이언트-서버 모델로서 분산 시스템에 참여하고 있는 여러 노드 중 한 노드가 서버 역할을 하여 공유 파일 접근에 대한 중앙 집중적인 관리를 맡게 된다. 따라서 분산된 각 클라이언트들이 공유 파일을 접근하기 위해서 서버의 중앙집중적 관리를 받아야 하므로 파일 서버의 병목현상을 피할 수 없다. 이러한 분산 파일 시스템의 대표적인 예는 썬 마이크로 시스템사의 NFS, AT&T사의 RFS, IBM과 CMU에서 초기 모형으로 개발한AFS등이 있다[2,3,5]. 그러나 최근 네트워크 부착형 저장장치(Network Attached Storage)와 저장장치 전용 네트워크(Storage Area Network)가 개발됨에 따라 NAS와 SAN 기반의 공유 파일 시스템들이 등장하고 있다[1,4]. SAN 기반의 공유 파일 시스템은 [그림 1]과 같이 지역 서버가 네트워크상에 부착된 저장장치의 데이터를 다른 서버의 도움 없이 직접 데이터를 전송 받을 수 있는 파일 시스템이다.




[그림 1] 클러스터 공유 파일 시스
[Fig. 1] Cluster Shared File Syst

SAN 기반 클러스터 파일 시스템은 특정한 서버와의 접근없이 SAN에 부착된 공유 저장장치들을 자유롭게 접근할 수 있다는 장점을 제공하지만 클러스터내의 각 서버들은 파일 공유를 위해서 기존의 파일 시스템을 사용할 수 없으며, 클러스터를 위한 전용 파일 시스템을 운영해야 한다. 즉 파일 공유를 위한 동시성 제어나 메타 데이터의 관리기능을 할 수 있는 클러스터 파일 시스템으로 대치되어야 한다. 클러스터 파일 시스템은 [그림 1]과 같이 여러 서버들이 클러스터에 참여하여 하나의 파일 시스템을 구성하게 된다. 기존의 클라이언트-서버 구조가 아니므로 디스크 접근시 특정 서버와의 연결이 필요없지만, 파일의 공유시 필요한 동시성 제어나 메타 데이터의 관리를 위해서 클러스터 파일 시스템의 일부 참여 노드들이 서로 송수신을 할 필요가 있다. 따라서 각 노드내에는 기존의 파일 시스템 대신 클러스터용 공유 파일 시스템이 운영되어야 한다. 그러나 현재 제안된 클러스터용 공유 파일 시스템들은 기존 분산 파일 시스템에서 제공되는 수준의 공유, 즉 서버가 제공해주는 데이터의 사용만이 가능하다. 그러나 진정한 파일 공유는 데이터의 소유자가 없는, 즉, 서버없는 공유 파일 시스템(serverless shared file system)을 제공해야 한다. 이러한 분류의 대표적인 공유 파일 시스템으로는 Veritas사의 CFS와 미네소타 대학에서 원형을 개발한 GFS 등이 있다[1,6,8]. SANiqueTM은 리눅스 클러스터 시스템을 기반의 SAN 기반 공유 파일 시스템으로서 I/O 병목현상을 제거하고 확장성 및 가용성을 극대화하였다. 본 논문에서는 클러스터 공유 파일 시스템 SANiqueTM의 기본적인 시스템 구조와 각 구성요소에 대하여 기술하며, 특히 각 노드간의 데이터 접근 효율을 극대화하기 위한 전역 버퍼 관리기법에 대하여 설명한다.

본 논문의 구성은 다음과 같다. 제2장에서 기존 공유 파일 시스템의 문제점 및 SANiqueTM의 시스템 구성요소를 기술하며 제3장에서 클러스터 공유 파일 시스템을 위한 전역 버퍼 관리기의 기존방법 문제점 및 설계방법을 제안한다. 제4장에서는 클러스터 공유 파일 시스템의 기대효과 및 향후 연구방향에 대하여 결론을 맺는다.

## 2.관련연구및공유파일시스템구조

### 2.1기존공유파일시스템기능및분석

클러스터 파일 시스템에 가장 필수적인 기능은 클러스터 노드간의 파일 공유 기능이다. 한 노드가 소유하고 있는 디스크의 파일 시스템 영역을 다른 노드에서도 읽기/쓰기 작업이 가능한 공유가 제공되어야 한다. 파일 시스템의 공유 기능에서 가장 신중하게 고려되어야 하는 문제점은 디스크를 소유한 노드의 병목현상이다. 즉, 특정 노드가 소유한 디스크를 여러 노드에서 집중적으로 접근하는 경우 그 노드에 부하가 걸려 효율적인 데이터 공유 서비스를 제공할 수 없다는 것이다. 분산파일 시스템중 클라이언트/서버 유형의 대표적 예인 네트워크 파일 시스템(NFS)은 바로 이와 같은 병목 현상을 피할 수 없다. 따라서 이러한 특정 노드의 병목현상을 없애기 위해 저장장치를 위한 전용 네트워크인 SAN을 구축하고, 또한 네트워크에 직접 부착하여 사용하는 NAS(network attached storage)를 이용하여 특정 디스크가 특정 노드에 소유되지 않는 새로운 기법의 클러스터 공유파일 시스템들이 등장하게 되었다.

IBM의 Tivoli사에서 나온 SANergy라는 제품은 파일 공유 솔루션으로 가장 먼저 출시되었으며 많이 사용되고 있다. 이 제품은 메타 데이터 서비스에 기반하는 미들웨어 방식을 채택하고 있다. 또한 SAN을 통한 물리적인 공유를 전제로 하고 있으며, 서버 역할을 하는 MDC 모듈과 클라이언트 모듈로 구성되는 클라이언트/서버 형식을 취하고 있어, 서버에 오버헤드가 발생 할 수 있으며 병목현상의 문제점을 안고 있다. 파일 공유의 특성상 대용량 멀티미디어 데이터를 취급하는 환경이 많은데 비해, 이 제품은 공유 데이터 전송에 많은 시간이 소요된다는 단점을 지니고 있다..

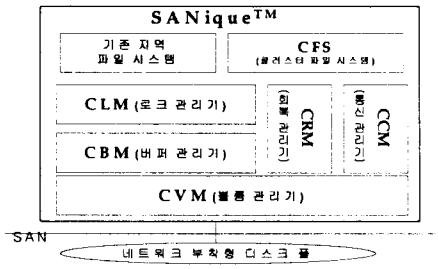
리눅스 시스템을 기반으로 배포되어온 Minnesota 대학의 GFS(Global File System)은 주로 개인적, 연구용 목적으로 사용되었으나 Sistina라는 기업으로 창업하여 현재 제품으로 출시되고 있다[4,7,9]. 이 공유 파일 시스템은 GFS 로크 서버 역할을 하는 노드와 클라이언트 노들드로 구성되는 비대칭형 구조이다. 각 클라이언트들은 GFS 로크 서버의 제어 하에 로크를 획득한 후, 공유 디스크를 접근하게 된다. 이 제품 역시 비대칭형 구조의 전형적인 문제인 병목현상으로 인한 성능 저하 현상을 갖게 되며, 로크 서버의 오류로 인하여 전체 시스템 서비스가 중단되는 저가용성 문제를 갖는다.

폴리서브 사의 Matrix 서버 제품은 본래 클러스터링 솔루션으로 개발되었으나, 최근 공유 파일 시스템의 수요가 증가함에 따라 공유 파일 기능이 추가된 경우이다. 최근 출시 제품이므로, 아직 시장 평가가 많이 이루어 지지 않은 상태로서 현재 10노드 클러스터 규모를 지원하며 플랫폼의 운영체제 또한 리눅스로 제한되고 있다.

썬 마이크로 시스템의 기술진들이 설립한 Veritas에서 출시한 SAN Point Foundation Suite는 클러스터 파일 시스템, 볼륨 관리기를 포함하는 제품으로 가장 최근 출시되었으며, 아직 오류로 인한 고가용성 제공의 미흡과 비용 및 번들 구성으로 인한 사용자 요구 만족 등의 문제점을 안고 있다[8].

### 2.2SANiqueTM의 시스템 구성도

SANiqueTM 은 SAN 기반의 클러스터 서버들을 위한 클러스터 공유 파일 시스템으로서, 전형적인 분산 파일 서버 시스템 구조의 병목현상을 제거한 공유 파일 시스템이다[10]. SANiqueTM 파일 시스템은 [그림 1]의 각 노드에 기술된 클러스터 공유 파일 시스템의 역할을 하는 파일 시스템이며 디스크의 소유 개념이 없는 진정한 클러스터 공유 파일 시스템이다. 각 노드에 위치한 클러스터 공유 파일 시스템은 크게 클러스터 볼륨 관리기(CVM), 클러스터 로크 관리기(CLM), 클러스터 버퍼 관리기(CBM), 클러스터 회복 관리기(CBM), 클러스터 파일 시스템(CFS)로 구성되며 [그림 2]와 같다.




[그림 2] SANiqueTM 시스템 구성

[Fig. 2] SANiqueTM System Architectu

SANiqueTM CFS에 의해 제공되는 노드간 파일 공유는 클러스터의 모든 단일 노드들이 SAN에 직접 연결된 디스크 장치들에 대해 병행적 공유 접근이 가능하다는 점에서 기존 NFS 등의 파일 시스템의 공유와 구별된다. SANiqueTM CFS에 의해 클러스터의 모든 단일 노드들은 SAN에 직접 연결된 디스크 장치들에 대해 병행적 공유 접근이 가능하다. CFS는 수퍼블록 영역, 비트맵 영역, 데이터 블록 영역으로 구성된다. 수퍼블록 영역은 파일 시스템의 데이터를 관리하기 위한 메타 데이터를 저장하고, 비트맵 영역은 디스크 공간의 할당 영역과 가용 영역을 관리하기 위한 영역이다. 전형적인 분산 파일 시스템에서는 파일 서버가 비트맵 영역을 관리하지만, 공유 파일 시스템상에서는 여러 노드에 의해 관리 및 접근되므로 상호 배타적인 접근 방법이 필요하고, 비트맵 수정 연산에 대한 일관성 유지 방법이 필요하게 된다.

SANiqueTM CLM은 클러스터의 여러 노드에 의해 동시 접근되는 같은 파일에 대해 직렬성을 보장하기 위한 기능을 제공한다. 공유 볼륨 상의 파일 접근을 직렬화하고 일관된 데이터 접근을 보장하기 위하여 파일 수준의 병행수행 제어 방법을 이용한다. 이에 따라 같은 파일에 접근하는 모든 연산은 파일 단위의 데이터 일관성을 보장 받는다. SANiqueTM은 대칭형 클러스터 파일 시스템의 구성이므로, 특정 서버가 로크 서버 역할을 하는 대신, 모든 노드들에게 지역 로크 서버 역할을 부여하고, 그 중 일부 몇 개의 노드들이 전역적 통제를 위한 전역 서버 역할을 한다.

SANiqueTM CVM은 주어진 클러스터 내에 사용할 수 있는 모든 디스크 공간을 사용 가능한 가상 볼륨으로 구성하여 저장 풀(pool)을 형성하고 파일 시스템으로부터 사용될 수 있게 한다. 물리적 디스크 집합으로부터 논리적 디스크 영역을 구성하고 이를 파일 시스템의 주소 공간으로 전환할 수 있는 매핑 테이블을 유지한다. 또한 논리적 디스크는 물리적 디스크의 구성 방식에 따라, 스트라이핑, 미러링(RAID-1), 패러티 스트라이핑(RAID-5)등을 지원할 수 있다.

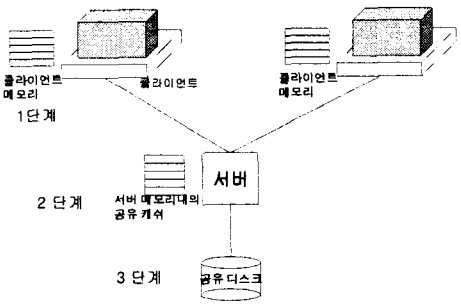
SANiqueTM CRM은 한 노드의 오류로 인하여 서비스 장애가 발생하는 경우 다른 노드에 영향을 주지 않도록 온라인상에서 오류 노드를 복구하는 회복 관리기이다. 특히 두 개 이상의 노드에서 서로 상대방 노드를 오류로 인식하여 발생하는 분할 그룹(split brain) 문제는 전형적인 분산 시스템의 문제이다. 공유 파일 시스템 상에서는 이러한 분할 그룹 문제가 파일의 일관성에 치명적인 문제를 발생시킨다. CRM은 이러한 분할 그룹 문제를 해결하기 위하여 공유 디스크 보드(shared disk board)를 활용하여 양분 또는 다분화된 클러스터내의 노드들을 장애 복구 하는 회복 관리기를 설계하였다.

SANiqueTM CBM 은 클러스터 노드들이 보유하고 있는 데이터 캐쉬 블록들이 다른 노드에서 필요할 때 전달될 수 있는 전역 버퍼 관리 기법(Global Buffer Management)을 이용한다. 기존 버퍼 관리 기법과는 달리, 전역 버퍼 관리 기법인 상호협조 캐싱(cooperative caching) 기법을 활용한 전역 버퍼 관리 기법을 제공한다. 캐싱된 메타 데이터를 요구하는 서로 다른 노드에게 전달하게 함으로서 공유 파일 시스템 관리를 위한 제어 정보가 효율적으로 전달될 수 있게 한다. 다음 절에서는 본 시스템에서 제안하는 전역 버퍼 관리 기능을 설명한다.

## 3. 전역버퍼관리기의설계

### 3.1 상호협조캐쉬알고리즘의특징및분석

기존 분산 파일 시스템은 데이터 블록을 효율적으로 접근하기 위하여 3단계에 걸친 캐쉬 기법을 제공한다[11]. 즉, [그림 3]과 같이 서버 디스크, 서버 메모리내의 공유 캐쉬, 그리고 클라이언트 메모리를 사용한다.




[그림 3] 기존 분산 파일 시스템의
상호협조 캐쉬 구조
[Fig. 3] Cache Memory Layout in the
Existing Distributed File Systems

지역 노드들을 연결하는 네트워크의 속도가 이더넷에서 광통신 등을 통한 100Mbps 이상의 초고속 네트워크로 발전함에 따라 외부 클라이언트의 메모리에서 원하는 데이터 블록을 전송받는 것이 서버의 디스크에서 전송받는 것 보다 유리한 환경으로 발전하게 되었다. 따라서 기존의 3단계 구성의 캐쉬 구조가 외부 클라이언트 메모리를 이용하는 4단계 상호협조적 캐쉬(cooperative cache) 구조로 발전하게 되었다.

상호협조 캐쉬를 이용한 방법중 가장 간단하면서 구현하기 쉬운 방법은 직접 클라이언트 협조(direct client cooperation) 방법으로 외부 클라이언트중에서 작업이 거의 없는 클라이언트를 선택하여 그 노드의 메모리를 보조 캐쉬 메모리로 사용하는 방법이다. 즉, 활동이 많은 클라이언트에서 캐쉬 오버플로우가 생기면 유휴상태의 외부 클라이언트 메모리에 저장하여 논리적으로 캐쉬의 크기를 증가하는 방법이다. 이 방법은 서버의 수정없이 쉽게 사용할 수 있으며, 또한 일시적으로 지역 캐쉬의 크기를 증가시킬 수 있는 방법이다. 그러나 서버의 제어가 되지 않는 방법이므로 다른 클라이언트와 전혀 공유되지 않으며, 같은 데이터 블록이 중복 저장될 수 있다.

그리디 포워딩 (greedy forwarding) 방법[11,14]은 서버의 제어를 통하여 외부 클라이언트의 캐쉬에 존재하는 데이터 블록을 전송 받는 방법이다. 즉 클라이언트에서 데이터 블록을 요청하면 먼저 자기 자신의 메모리 영역을 찾게 되고, 그 후 서버 공유 캐쉬에 존재하는지 찾은 후 원하는 데이터 블록이 존재하지 않으면 서버에 의해 관리되는 다른 클라이언트 캐쉬 목록을 찾게 된다. 외부 클라이언트 캐쉬 목록상에 데이터 블록이 존재하면, 해당 클라이언트에게 데이터 블록 전송 명령을 보내 데이터를 요청한 클라이언트로 그 블록을 전송하게 된다. 자신의 메모리, 서버 캐쉬 영역, 서버의 외부 클라이언트 캐쉬 목록 모두 원하는 데이터 블록이 존재하지 않으면 서버 디스크에서 직접 데이터 블록을 읽어 전송하게 된다. 그러나 이 방법 역시 각 클라이언트의 캐쉬 내용은 지역적, 독자적으로 운영되므로 중복된 데이터 블록을 갖게 될 수 있어 전역적인 버퍼 관리 기능을 제공할 수 없다.

중앙 상호협조 캐쉬(centrally coordinated caching) 방법[11,12,14]은 위에서 언급한 그리디 방법의 외부 클라이언트 데이터 블록 전송 방법과 동일하다. 그러나 각 클라이언트의 캐쉬 영역은 지역적으로만 관리되는 것이 아니라, 서버에 의해 전역적으로 관리되는 부분을 갖게 된다. 즉, 하나의 클라이언트 캐쉬 영역은 클라이언트에서 지역적으로 관리하는 영역과 서버에 의해 전역적으로 관리되는 두 부분으로 나뉘게 된다. 전역적 캐쉬 영역은 서버에 의해 전역적 버퍼 블록 대치 방법에 따라 데이터 블록이 할당된다. 따라서 그리디 방법 보다 전역적 버퍼 기능을 효율적으로 제공하게 되어 전체적으로 중복된 데이터 블록을 적게 가질 수 있지만 활동이 많은 클라이언트 입장에서는 독자적으로 운영할 수 있는 클라이언트 캐쉬가 줄어들게 되므로 캐쉬 미스가 많이 발생할 수 있는 단점을 갖는다.

또 다른 방법으로는N-chance 포워딩 캐쉬 방법[13,14]으로 지역 클라이언트의 캐쉬 관리 방법에 따라 캐쉬에서 제거되는 희생 데이터 블록(victim)을 다른 클라이언트의 캐쉬로 보내어 전역적 캐쉬 성공률을 높이는 방법이다. 기본적으로 외부 클라이언트의 캐쉬를 이용하는 방법은 앞서 언급한 그리디 포워딩 방법과 동일하지만, 캐쉬 데이터 블록을 관리하는 방법에 있어

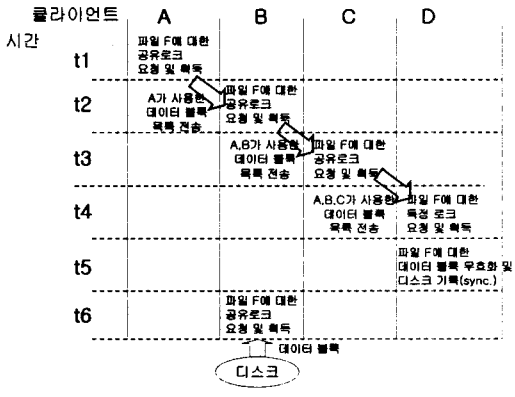
서 무조건 제거하기 보다는 활동이 적은 클라이언트로 보내어 캐쉬 성공율을 높이게 하는 방법이다. 지역 클라이언트에서 선정된 희생 데이터 블록이 다른 클라이언트 캐쉬에도 존재하는 중복 데이터 블록인 경우에는 무조건 제거된다. 또한 유일한 데이터 블록이라 할지라도 시스템 내에 설정된 N번의 희생 블록으로 결정되는 동안 데이터 접근이 없는 경우에는 전체 캐쉬내에서 제거되게 된다. 따라서 이 방법은 데이터 블록이 시스템 내에 유일한 블록인지를 판단하기 위한 방법과 각 데이터 블록 마다 순환 횟수(recirculation count)를 유지해야 한다. 또한 모든 데이터 블록이 시스템 캐쉬에서 제거되기 전에 주어진 횟수만큼 순환된 후 제거된다는 단점을 가지고 있다.

한편 OSF의 GMS[12]에서 제안하여 사용하는 캐쉬 방법은 중앙 상호협조 캐쉬 방법처럼 각 클라이언트의 캐쉬를 지역적인 데이터 블록과 전역적인 데이터 블록을 저장하여 사용하게 하지만, 그 영역의 비율이 고정되지 않고 그 클라이언트의 활동성에 따라 지역적 영역이 증가하게 되는 방법을 사용한다. 즉 외부 클라이언트 캐쉬에서 전송받은 데이터 블록은 자신의 지역 캐쉬영역에 저장하게 되고, 자신의 전역적 캐쉬 영역중 가장 오래된 블록을 다른 클라이언트의 전역 캐쉬영역으로 전송하여 지역적 캐쉬 영역을 증가시키게 된다. 반면 활동이 적은 클라이언트의 캐쉬 영역은 다른 클라이언트들로부터 전송 받은 데이터 블록을 저장하게 되어 전역적 캐쉬영역이 증가하게 된다. 캐쉬 성능을 높일 수 있는 좋은 방법이지만 클러스터 공유 파일 시스템 상에서 구현하기 어려운 단점을 갖고 있다.

### 3.2SANiqueTM의 전역버퍼관리기법

클러스터 공유 파일 시스템의 전역 버퍼 관리 기본 목적은 기존 분산 파일 시스템과 같이 외부의 클라이언트 메모리를 자신의 캐쉬영역처럼 사용할 수 있게 하는 것이다. 그러나 기존 시스템과 같이 중앙집중적인 서버 운영은 디스크 접근의 병목현상을 유발하므로 서버 없이 클라이언트 메모리를 공유할 수 있게 해야 하는 어려움이 있다. 또한 중앙통제 없이 다른 클라이언트의 메모리를 이용하기 위해서는 모든 클라이언트들에 적재되어 있는 데이터 블록 목록을 모든 클라이언트들이 유지해야 하는 오버헤드가 있으며 상대적으로 전송되는 제어 메시지가 기하급수적으로 증가하게 된다. 따라서 SANiqueTM 시스템의 전역 버퍼 관리기는 이러한 오버헤드를 제거하기 위하여 각 노드마다 설치 운영되는 클러스터 로크 관리기(CLM)의 정보를 공유하여 사용한다. 중앙집중적인 전역적 버퍼 관리기능을 사용할 수 없으므로 각 지역 클라이언트의 메모리는 전적으로 지역적인 데이터 캐쉬를 목적으로 사용되며, 외부 클라이언트의 요청시 그리디 포워딩 방법과 같이 요청 데이터 블록을 전송하게 된다.

SANiqueTM 시스템의 클러스터 로크 관리기는 파일 간의 공유를 위해 읽기/쓰기 작업 전에 획득한 로크들을 관리한다. 로크의 기본 단위는 파일 단위로 이루어진다. 따라서 특정 파일에 대한 읽기/쓰기 작업을 하게 되면 그 파일에 대한 해당 로크를 획득하고, 사용한 데이터 블록 목록을 유지한다. 이 로크 관리기의 로크 획득 정보와 사용한 데이터 목록을 이용하여 SANiqueTM 시스템의 전역 버퍼 관리기능을 구현한다. 기본적인 전역 버퍼 관리 시나리오는 [그림 4]와 같다




[그림 4] 전역 버퍼 관리 시나리

[Fig. 4] Scenario for Global Buffer Management

클라이언트에서 특정 파일에 대한 로크를 요청하게 되면 . [그림 4]의 클라이언트 B와 같이

로크 획득과 동시에 바로 이전에 로크를 획득하였던 노드로부터 사용하던 데이터 블록의 목록을 동시에 전달받게 된다. 실제 데이터 블록은 전송되지 않는다. 클라이언트 C는 로크 획득과 동시 파일 F에 대한 공유 로크를 획득하였던 A, B의 데이터 블록 목록을 전송받는다. 이 때, 클라이언트 C에서 A 또는 B에서 사용하던 데이터 블록을 요청하게 되면 디스크 영역으로부터 데이터 블록을 가져오지 않고, 해당 클라이언트의 메모리로부터 전송받게 된다. 한편 클라이언트 D 처럼 독점 로크를 요청하게 되면 이전 공유 로크를 획득하였던 클라이언트로부터 데이터 블록의 목록을 전송받아 외부 클라이언트의 메모리를 자신의 캐쉬처럼 활용하지만, 파일 F에 대한 사용이 끝난 후 로크를 반환하는 시점에 데이터 블록의 무효화를 각 클라이언트에 알리게 되고, 실제 디스크에 기록하여 캐쉬와 디스크의 동기화를 실행하게 된다. 또한 시간 t6와 같이 클라이언트 B가 공유 로크를 요청하는 경우에는 시간 t5에서 이미 독점 로크를 획득하여 디스크 동기화 작업을 수행한 이후이므로 모든 요청한 데이터 블록은 디스크로부터 전송받아 사용하게 된다.

데이터의 동기화 작업은 파일 공유의 일치성을 위해 필요한 작업이며, 전역 버퍼 관리기는 동기화 작업으로 인해 전역 버퍼를 위한 데이터 블록 목록의 무제한적 증가를 피할 수 있게 된다. 또한 클러스터 로크 관리기에 의해 관리되던 정보를 활용하여 전역 버퍼 관리를 하게 되므로, 기존의 다른 캐쉬 기법들 처럼 전역 버퍼 관리를 위한 추가 정보의 저장 및 운영을 위한 불필요한 메시지 전송을 피할 수 있으며 클러스터 내의 클라이언트 오류로 인해 전역 버퍼 기능이 영향을 받지 않아 확장성 및 가용성을 높일 수 있다.

## 4.결론및향후연구

본 논문에서는 클러스터 기반의 SAN 기반 공유 파일 시스템인 SANiqueTM 에 대한 시스템 구성도 및 설계 방법을 기술하였다. SANiqueTM 시스템은 대용량 데이터 처리를 요구하는 모든 분야에서 활용될 수 있다. 특히 대규모 웹 서비스 및 대용량 웹 컨텐츠 서비스 부분, 메일 서비스 및 금융 서비스 부분등에 활용될 수 있다. 특히 본 논문에서는 데이터 서비스의 성능 향상을 위해 다른 클라이언트의 메모리를 활용하는 전역 버퍼 관리기법을 논의하였다. 이 전역 버퍼기능은 기존 분산 시스템의 상호협조 체계로 운영되던 모든 기본기능을 제공하면서 동시에 전역 버퍼 관리를 위한 중앙집중적 서버를 운영하지 않는 클러스터 공유 파일 시스템에 적합한 방법이다. 향후 SANiqueTM 은 이기종 클러스터 환경에서 운영가능하도록 공유 파일 시스템 모듈을 개선하고 있으며, 전역 버퍼의 데이터 블록을 최소화 할 수 있도록 개발을 진행 중에 있다.

## 참고 문헌

[1] M. D. Dahlin, "Severless Network File Systems", Ph. D. Thesis at Computer Science Graduate Division of University of California at Berkely, 1995

[2] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network File Systems", Proc. Of the Summer USENIX Conf. 1985

[3] U. Vahalia, Unix Internals : The New Frontiers, Prentice-Hall, NJ, 1996

[4] S. R. Soltis, T. M. Ruwart, and M. T. O'keefe, "The Global File Systems", Proc. Of the 5thNASA Goddard Conference on Mass Storage Systems and Technologies, 1996.

[5] M. Satyanarayanan, "Scalable, Secure, and Highly Available Distributed File Access", IEEE Computer, 1990

[6] S. R. Soltis, T. M. Ruwart, and M. T. O'keefe, "The Global File Systems", Proc. Of the 5thNASA Goddard Conference on Mass Storage Systems and Technologies, 1996.

[7] David Teigland, "The Pool Driver : A

volume Driver for SANs", Master's Degree Thesis, University of Minnesota, Dept. of Electrical and Computer Engineering, 1999.

[8] VERITAS Software Corp., *Veritas Volume Manager,* http://www.veritas.com

[9] Heinz Maulshagen, "Logical Volume Manager for Linux", Sistina Technical Memo, http://www.sistina.com.

[10] K. Lee, "Failure Recovery in the Linux Cluster File System SANique ", KIPS Trans. Part A. Vol. 8, No. 4, 2001

[11] Michael D. Dahlin, Randolph Y.Wang, Thomas E. Aderson, David A. Patterson, "Cooperative Caching Using Remote Client Memory to Improve File System Performance", Proceedings of the First Symposium on Operating Systems Design and Implementation, 1994.

[12] Michael J. Feeley, William E. Morgan, Frederic H. Pighin, Anna R. Karlin, Henry M. Levy and Chandramohan A. Thekkath, "Implementing Global Memory Management in a Workstation Cluster", Proc. of the Symposium on Operating Systems Principles, 1995.

[13] Prasenjit Sarkar and John Hartman, "Efficient Cooperative Caching using Hints", In Proceedings of the 2nd Symposium on Operating System Design and Implementation, October 1996.

[14] Prasenjit Sarkar and John Hartman, "Hint-based Cooperative Caching", ACM Trans. on Computer Systems, Vol. 18, No. 4, 2000.

이 규 웅

1990년 한국외국어대학교 전자계산학과(이학사)
1992년 서강대학교 대학원 전자계산학과(공학석사)
1998년 서강대학교 대학원 전자계산학과(공학박사)
1998년-2000년 8월 한국전자통신연구원 인터넷서비스 연구부 선임연구원
2000년 9월 ~ 현재 상지대학교 컴퓨터.정보공학부 조교수
관심분야 : 자료저장 시스템, 분산 및 실시간 DB

차 영 환

1983년 인하대학교 전자계산학과(공학사)
1985년 한국과학기술원 전자계산학과(공학석사)
1993년 인하대학교 대학원 전자계산학과(공학박사)
1985년-1990년 한국전자통신연구원 선임연구원
1994년 ~ 현재 상지대학교 컴퓨터.정보공학부 부교수
관심분야 : 통신 프로토콜, 네트워크 구조, 모바일 네트워크