SAN 환경의 대용량 클러스터 파일 시스템을 위한 광역 버퍼 관리기의 설계

이규웅*

Design of Global Buffer Manager in SAN-based Cluster File Systems

Kyu-Woong Lee*

이 논문은 2010년도 상지대학교 교내연구비 지원에 의한 결과임

요 약

본 논문은 SAN 기반의 클러스터 파일 시스템 SANique™의 설계 고려사항을 설명하고 특히 대규모의 호스트들이 연결된 상황에서 광역 버퍼 관리기의 데이터 공유 문제점을 기술하고 클러스터 환경에서 확장성과 가용성을 제공하기 위한 광역 버퍼 관리 기법을 제시한다. 제안하는 광역 버퍼 관리 기법은 SANique™ 시스템의 로크 관리기에서 관리되는 로크 정보를 활용하여 부가적인 통신 및 시간소요 없이 효율적인 데이터 공유를 제공하는 방법을 제시하였다. 또한 대규모 클러스터 환경에 적용 가능한 광역버퍼 관리기법의 의사코드 및 데이터 공유 예제를 통하여 본 방법의 효율성을 보였다.

ABSTRACT

This paper describes the design overview of cluster file system SANique™ based on SAN(Storage Area Network) environment. The design issues and problems of the conventional global buffer manager are also illustrated under a large set of clustered computing hosts. We propose the efficient global buffer management method that provides the more scalability and availability. In our proposed global buffer management method, we reuse the maintained list of lock information from our cluster lock manager. The global buffer manger can easily find and determine the location of requested data block cache based on that lock information. We present the pseudo code of the global buffer manager and illustration of global cache operation in cluster environment.

키워드

클러스터 시스템, 공유 파일 시스템, 캐시 데이터, 광역 버퍼

Key Words

Cluster System, Shared File System, Cache data, Global Buffer

* 정회원 : 상지대학교 컴퓨터정보공학부

접수일자 : 2011. 06. 08
심사완료일자 : 2011. 07. 15

Ⅰ. 서 론

클러스터 파일 시스템이란 독립적 저장장치들을 광채널 기반 저장장치 전용 네트워크(SAN)에 직접 연결하여 특정 지역 서버의 간섭 및 제어 없이 네트워크를 통해 직접적인 접근이 가능한 데이터 중심적인 새로운 저장장치 환경이다[1]. 기존의 분산 구조에 근간을 두고 있는 서버 지향적 파일 시스템은 파일 서버 자체적인 용량의 한계와 서버와 클라이언트 간의 잦은 데이터 전송에 따른 메모리 복사가 요구되며, 서버의 오류 발생시 모든 클라이언트들이 서비스 받지 못하는 치명적인 오류를 유발한다. 따라서 최근과 같이 사용자의 대용량 데이터 요구가 급증하는 모바일 인터넷 서비스 시대에 적합하지 못한 파일 시스템 구조이다[2, 3]. 대표적인 클러스터 파일 시스템의 구조는 그림 1과 같이 컴퓨팅 호스트들이 광채널과 같은 저장장치 전용 네트워크인 SAN상에 연결된 저장공간을 공유 할 수 있도록 하는 공유 파일 시스템을 제공하는 구조이다. 클러스터 시스템에 참여하고 있는 각 호스트들은 공유 파일 시스템의 자원을 항상 접근 가능하며, 중앙 제어 역할을 하는 서버가 존재하지 않으므로 접근 병목 현상을 유발하지 않는다.

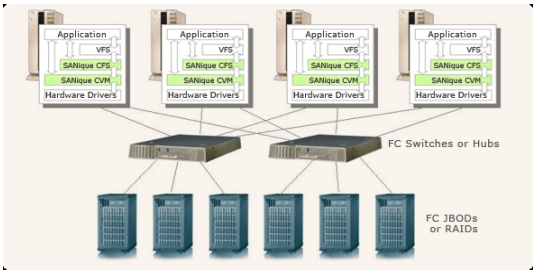

그림 1. SAN 기반 클러스터 파일 시스템 구조
Fig. 1 SAN-based Cluster File System

이러한 클러스터 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.

컴퓨팅 호스트는 각 독립된 자신의 메모리를 사용하여 클러스터 파일 데이터를 저장하기 때문에 여러 호스트가 동시에 공유 디스크 파일 시스템을 사용하기 위해서는 일반 및 메타 데이타에 대한 일관성을 보장하는 방법이 필요하다. 즉, 파일에 대한 읽기 혹은 쓰기 동작을 수행하더라도 여러 컴퓨팅 노드에서 일관된 뷰를 가질 수 있도록 보장해야 한다. 호스트들간 데이타 일관성을 보장하기 위한 하나의 방법으로 변경된 데이타를 즉시 클러스터 파일 시스템에 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 클러스터 파일 시스템의 성능을 저하시키는 주된 원인이 된다. 또한 여러 호스트가 동시에 디스크 I/O를 수행하게 되면 디스크 접근에 대한 경쟁이 발생하여 추가적이 성능저하가 발생한다. 그러므로 클러스터 공유 파일 시스템을 위한 버퍼 관리 기법의 설계가 필요하다. 기존 분산 시스템과의 버퍼관리 측면의 차이점은 버퍼 관리를 위한 주 서버 역할을 하는 호스트가 없는 것이다. 제어를 담당하는 마스터 서버가 있는 경우 클러스터 파일 시스템의 전체적인 성능에 병목현상을 유발할 수 있으므로, 버퍼 관리를 위한 마스트 서버의 역할 없이 클러스터에 참여하는 호스트들이 데이터 버퍼를 공유할 수 있는 클러스터 광역 버퍼 관리기법의 설계가 필요하다. 따라서 본 논문에서는 클러스터 파일 시스템의 광역 데이터 블록에 대한 효율적 탐지 및 전달방법을 사용하는 광역 버퍼 관리기를 제안한다.

본 논문은 제2장에서 대표적 상용 시스템인 클러스터 파일 시스템 SANique™의 구조 및 특징을 설명한다. 제3장에서 클러스터 시스템의 광역 버퍼 관리기의 문제점 및 개선된 방법을 제안하고 제4장에서 결론을 맺는다.

Ⅱ. 연구 배경 및 관련 연구

가. 클러스터 파일 시스템의 개요 및 배경

최근 클러스터 파일 시스템은 구글 파일 시스템 GFS(Google File System)[4, 5, 6], 한국전자통신연구원의 OASIS[7]와 같이, 대규모 사용자와 대용량 데이터 서비스 요구를 맞추기 위해 각 호스트들을 클러스터링 하여 하나의 대용량 파일 시스템으로 형성하는 클러스터 파일 시스템으로 확장 개발되고 있다. 구글 클러스터 파일 시스템은 그림 2와 같이 구글의 동영상 검색 및 저장 서비스, 이미지 서비스, 구글 어스 등과 같은 구글의 모든 데이터 집중적인 웹 서비스를 제공하기 위한 구글 플랫폼으로 사용되는 대표적인 분산 클러스터 파일 시스템

이다. 구글 파일 시스템은 GFS 서버 그룹들로 클러스터링 되어 있으며 하나의 응용을 서비스 하기 위하여 전담 서버 역할을 하는 마스터 서버로 구성된다[4,5]. 마스터 서버에 의해 파일 이름, 접근 오프셋 등이 결정되어 지면, 공유 파일 시스템내의 디스크 풀에 존재하는 공유 데이터들은 직접 응용으로 전달된다[6].

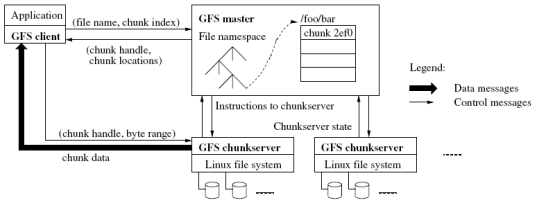

그림 2. 구글 파일 시스템 GFS의 아키텍쳐
Fig. 2 The Architecture of Google File System

나. 광역 버퍼 관리기 기존연구 및 문제점

버퍼 캐시는 주기억장치내의 특정 영역으로서 물리적인 장치로의 접근 시간을 절약하기 위해 사용된다. 버퍼 캐시에는 디스크와 같은 블록 장치로부터 읽고 쓰는 모든 블록 데이타가 저장된다. 그러나 기존 독립시스템의 버퍼 캐시는 단일 호스트에서 사용되도록 고안되었기 때문에 공유 디스크 파일 시스템에서 사용할 때는 여러 호스트 간에 데이터 일관성을 보장할 수 없다[7, 8, 11, 12]. 또한 기존의 분산 시스템에서 활용되는 버퍼 캐싱 기법은 그림 3과 같이 공유 디스크를 효율적으로 지원하기 위하여 디스크를 제어하는 마스터 서버에서 광역 버퍼 관리를 담당하고 각 클라이언트에게 필요한 데이터를 전달해 주는 방식의 3단계 기법을 사용한다. 그러나 SAN 기반 클러스터 파일 시스템에서는 범용 네트워크를 통한 데이터 버퍼를 전송하는 속도가 증가함에 따라 각 클라이언트들 간의 데이터 버퍼 교환을 통해 캐시를 업데이트하는 새로운 4단계 방식의 상호 협동 캐싱 구조로 발전하게 되었다.

상호 협동 캐시를 이용한 방법 중에서 가장 간단하면서 구현하기 쉬운 방법은 직접 클라이언트 협조(direct client cooperation) 방법으로 외부 호스트 서버들 중에서 작업이 거의 없는 호스트의 메모리를 보조 캐시 메모리로 사용하는 방법이다[11].

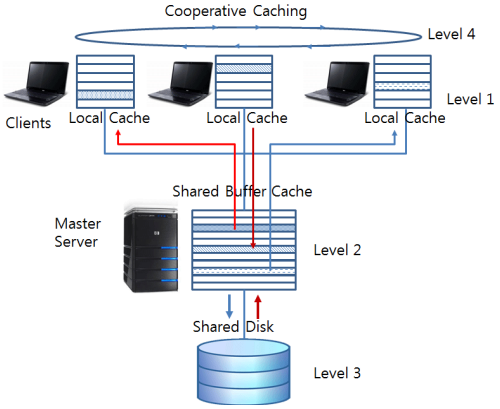

그림 3 기존 분산 시스템의 광역 버퍼 관리 기법
Fig. 3 Traditional Distributed Global Buffer Management

즉, 디스크 작업 요청이 많은 호스트에서 캐시 오버플로우가 생기면 유휴상태의 외부 호스트 메모리에 저장하여 논리적으로 캐시의 크기를 증가하는 방법이다. 이 방법은 기존 3단계 캐싱구조의 기법을 거의 수정하지 않고 쉽게 구현할 수 있는 간단한 방법이다. 그러나 서버의 중앙제어 하에 버퍼 캐시가 되지 않으므로, 각 호스트들 간에 공유되지 못하며 중복 데이터가 저장될수 도 있다.

그리디 포워딩(greedy forwarding) 방법[7,8]은 서버의 제어를 통하여 외부 클라이언트의 캐시에 존재하는 데이터 블록을 전송 받는 방법이다. 즉 먼저 자신의 메모리 영역을 찾게 되고, 그 후 서버 공유 캐시에 존재하는지 찾은 후 원하는 데이터 블록이 존재하지 않으면 서버에 의해 관리되는 다른 호스트들의 캐시 목록을 찾게 된다. 그러나 이 방법 역시 각 호스트들의 캐시 내용은 지역적으로 운영되므로 중복된 데이터 블록을 갖게 될 수 있어 광역적인 버퍼 관리 기능을 제공할 수 없다.

중앙 상호협조 캐시(centrally coordinated caching) 방법[7,8,9]은 위에서 언급한 그리디 방법의 다른 호스트에 존재하는 데이터 블록을 활용하는 방법과 동일하다. 그러나 각 호스트 서버의 캐시 영역은 지역적으로만 관리되는 것이 아니라, 서버에 의해 광역적으로 일부 관리된다. 광역적 캐시 영역은 서버에 의해 광역적 버퍼 블록 대치 방법에 따라 데이터 블록이 할당된다. 따라서 그리디 방법 보다 광역적 버퍼 기능을 효율적으로 제공하게 되어 전체적으로 중복된 데이터 블록을 적게 가질 수 있지만 활동이 많은 클라이언트 입장에서는 독자적으로

운영할 수 있는 클라이언트 캐시가 줄어들게 되므로 캐시 미스가 많이 발생할 수 있는 단점을 갖는다.

N-chance 포워딩 캐시 방법[10]은 각 자신의 호스트 캐시 관리 방법에 따라 캐시에서 제거되는 희생 데이터 블록(victim)을 다른 호스트의 캐시로 보내어 전역적 캐시 성공률을 높이는 방법이다. 기본적으로 외부 호스트의 캐시를 이용하는 방법은 앞서 언급한 그리디 포워딩 방법과 동일하지만, 캐시 데이터 블록을 관리하는 방법에 있어서 무조건 제거하기 보다는 활동이 적은 클라이언트로 보내어 캐시 성공율을 높이게 하는 방법이다. 유일한 데이터 블록이라 할지라도 시스템 내에 설정된 N번의 희생 블록으로 결정되는 동안 데이터 접근이 없는 경우에는 전체 캐시에서 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 캐시에서 제거되기 전에 주어진 횟수만큼 순환된 후 제거된다는 단점을 가지고 있다.

한편 OSF의 GMS[9]에서 제안하는 캐시 방법은 중앙 상호협조 캐시 방법처럼 각 클라이언트의 캐시를 지역적인 데이터 블록과 전역적인 데이터 블록을 저장하여 사용하게 하지만, 그 영역의 비율이 고정되지 않고 그 클라이언트의 활동성에 따라 지역적 영역이 증가하게 되는 방법을 사용한다. 활동이 낮은 클라이언트의 캐시 영역은 다른 클라이언트들로부터 전송 받은 데이터 블록을 저장하게 되어 전역적 캐시영역이 증가하게 된다. 캐시 성능을 높일 수 있는 좋은 방법이지만 클러스터 공유 파일 시스템 상에서 구현하기 어려운 단점을 갖고 있다.

Ⅲ. 클러스터 파일 시스템의 광역 버퍼 관리기

가. SANique™ 클러스터 파일 시스템의 구조

SANique™ 시스템은 현재 국내 외의 많은 곳에서 활용되고 있는 (주)매크로임팩트사의 SAN 기반의 지능형 범용 클러스터 공유 파일 시스템이다. 최근 상용화되고 있는 클러스터 파일 시스템은 특정 목적을 전제로 서비스되는 반면 SANique™은 범용 클러스터 공유 파일 시스템이다. 클러스터 파일 시스템의 각 호스트들에게 진정한 공유를 제공하기 위하여 그림 4와 같은 시스템 구조를 갖는다. 각 호스트 서버들은 기존 파일 시스템 위에 주요 구성 모듈인 CFS(Cluster File System)와 CVM (Cluster Volume Manager)을 설치하여, 공유 논리적 볼륨을 생성하고 그 공유 디스크 영역에 대해 읽기/쓰기에 대한 모든 제어를 담당하게 된다. 각 호스트 마다 존재하는 CFS 모듈은 저널링 공유 파일 시스템이다. CFS의 파일 시스템 메타 정보는 특정 노드에 할당되지 않고 클러스터를 구성하는 모든 호스트들에 분할 저장되므로 공유 디스크를 접근할 때 마다 각 서버들의 정보 공유를 통해 디스크 접근을 수행하게 된다.

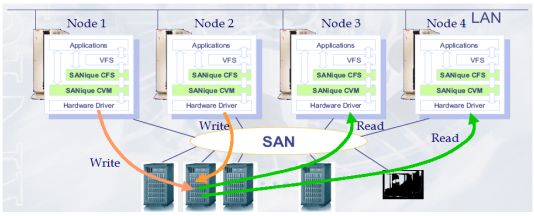

그림 4. SANique™의 시스템 구성도
Fig. 4 System Architecture of SANique™

CVM은 클러스터 볼륨 관리기로서 SAN에 부착된 모든 디스크들을 논리적 공유 볼륨으로 구성하는 볼륨 관리기이며 볼륨은 물리적 디스크의 구성 방식에 따라 스트라이핑, 미러링, 패러티 스트라이핑 등 다양한 형식으로 지원될 수 있다.

나. 광역 버퍼 관리기의 시스템 구조 및 개요

광역 버퍼 관리기는 필요한 데이터 블록이 자신의 호스트 버퍼 캐시에 없는 경우, 직접적인 공유 디스크의 I/O를 통하지 않고 다른 호스트의 버퍼 캐시로부터 데이터 블록을 전송받아 사용하는 기법이다. 그림 5는 광역 버퍼 관리기(CBM)의 주요 구성 요소를 보인 것이며, 또한 클러스터 파일 시스템의 다른 주요 모듈들과 상호 동작되는 통신 흐름을 보이고 있다.

광역 버퍼 관리기의 데이터 검색 관리기(lookup manager)는 자신의 호스트 버퍼 캐시 내에 요청하는 데이터가 없는 경우 다른 호스트 캐시 내에 요청 데이터가 있는지 검색하는 역할을 담당한다. 광역 버퍼 관리기의 기본적인 기능은 관련 연구에서 소개하였던 그리디 메소드를 기반으로 개발되었다. 제안하는 방법은 원하는 데이터가 위치하는 호스트를 탐색하기 위해 모든 주위

클러스터 호스트를 검색하는 비용을 줄이기 위해 로크 관리기의 정보를 활용하여 어느 호스트가 원하는 데이터 블록을 캐시하고 있는지 효율적으로 탐색하게 된다.

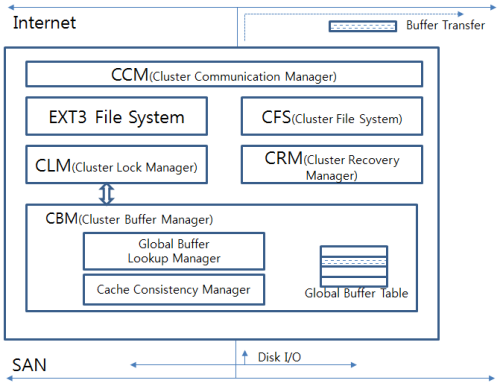

그림 5. 클러스터 파일 시스템의 모듈 구성
Fig. 5 Component of Cluster File System

기존 광역 버퍼 관리 방법의 핵심적인 문제점은 호스트들간에 보유하고 있는 데이터 블록의 공유 문제이다. 성능이 낮은 방법일수록 공유의 정도가 낮다. 또한 지역적 메모리에서 방출해야하는 희생 데이터 블록을 사용율이 낮는 타 호스트의 메모리를 활용하여 디스크로 스왑되지 않도록 하여 데이터 블록의 적중률을 높이면서 공유될 수 있도록 하는 문제이다. 이러한 문제점을 해결하면서 동시에 중앙 제어 역할을 하는 마스터 호스트가 없어야 하는 것이 또 하나의 문제점이다.

본 논문에서 제안하는 광역 버퍼 관리 기법은 로크 관리기(CLM)의 정보를 활용하여 자신이 원하는 데이터 블록이 어느 호스트에 존재하는지 실시간으로 검색하여 상수시간에 광역 데이터 버퍼를 찾을 수 있다. 또한 로크 관리기 및 광역 버퍼 관리기는 그림 5에서 설명한 바와 같이 특정 호스트에서 수행되는 모듈이 아니라 모든 호스트에서 수행되는 모듈이다. 즉, 특정 마스터 서버의 역할을 하는 호스트 없이 전체 파일 서비스를 수행하고 있으므로 성능상의 병목현상을 유발하지 않으면서 광역 버퍼 관리 기능을 제공할 수 있도록 한다. 클러스터 파일 시스템의 로크 관리기는 로크 개체를 파일 단위로 사용하며, 로크 관리 서버가 각 파일 들에 대한 로크를 어느 호스트가 사용 중인지 관리한다. 또한 데이터의 빈번한 접근시 로크 서버관리기의 중복적인 통신을 피하기 위해 다른 노드의 로크 요청시 로크 릴리즈를 수행한다. 그림 6의 예와 같이 호스트 2가 파일 A에 대한 로크를 요청(1)하면 로크 서버로부터 현재 호스트 1이 로크 소유자임을 확인하고 로크 반환을 요청한다.

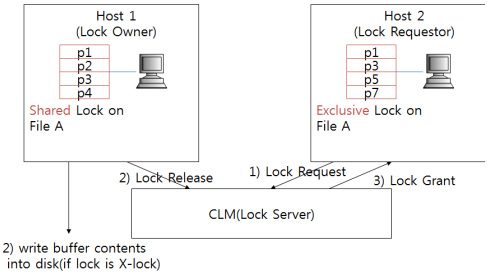

그림 6. 클러스터 로크 관리기의 수행 예
Fig. 6 Example of Lock Acquisition Process

호스트 2는 현재 파일 A가 사용 중이지 않는 경우 로크를 반환(2)한다. 만약 소유하고 있는 로크가 배타적 로크인 경우에는 디스크에 강제 내려쓰기를 수행한 후 로크를 반환(2)한다. 반환된 로크는 로크 관리 서버로부터 로크를 전달받아 최종 획득(3)하게 된다. 로크 서버 관리기는 해당 로크 정보와 함께 그 파일에서 사용되었던 데이터 블록의 리스트를 유지한다. 따라서 제안하는 광역 버퍼 관리기는 이 로크 서버의 정보를 활용하여 효율적인 상호 협동 캐시를 제공한다.

다. 로크 정보를 활용한 광역 버퍼 관리 기법

클러스터 파일 시스템의 광역 버퍼 관리기의 개략적인 의사코드는 그림 7과 같다. 그림 8은 제안된 광역 버퍼 관리 기법을 적용하여 4개의 서로 다른 호스트에서 파일 A의 데이터 블록 d1에 대한 접근할 때 발생하는 상호 협조 캐시 전달의 예를 보인 것이다. 가장 먼저 호스트 A에서 데이터 블록 d1의 접근을 요청하면 파일 A에 대한 로크를 획득하고 접근한다(그림 8의 (1)과(2)). 그 후 호스트 B에서 같은 데이터 블록을 요청하면 그림 6에 기술한 로크 요청 방식에 따라 로크를 요청 및 획득하게 되고(그림 8의 (4)), 동시에 호스트 A에서 사용하던 파일 A의 모든 데이터 블록을 호스트 B에 전달하게 된다(그림 8의(5)). 즉, 호스트 B는 데이터 블록 d1이 자신의 캐시에 없으므로 상호 협조 캐시기법에 따라 다른 호스트가 그 블록을 소유하고 있는지 판단해야 한다.

```
1.  CBM_Mgmt_at_a_node(data  block p, File A)
2.  {
3.        - Lookup its own cache memory for the requested data block
4.        - if ( not exists in my cache)  {
5.                  - determine the lock for the data block and the file
6.                  - send a request to the Lock server to acquire the lock.
7.                  - wait the lock server's response for the requested lock.
8.                  - grant a lock for file A
9.                  - receive the data blocks that is used a previous host
10.                 if ( there is no data block in the received data blocks)
11.                       - request the disk I/O for that block to shared file system
12.       }
13.}
```

그림 7. 광역 버퍼 관리 방법의 의사 코드
Fig. 7 Pseudo Code of Global Buffer Management

본 방법에서 제안하는 광역 버퍼 기법은 로크 관리기의 정보를 이용하여 호스트 A가 그 블록을 소유하고 있음을 알 수 있어, 부가적인 통신 및 탐색 없이 효율적으로 데이터 블록을 전송받아 상호 협조 캐시 기법을 구현할 수 있게 된다.

호스트 C에서 데이터 블록 d1에 대한 쓰기 연산을 수행하는 경우에도 그림 6과 그림 7에 기술한 방법에 따라 로크를 요청 및 획득하고, 호스트 B로부터 데이터 블록을 전달받아 쓰기 연산을 수행한다. 쓰기 연산은 다른 모든 호스트들의 캐시 데이터를 무효화하게 된다. 따라서 다른 모든 호스트들에게 소유하고 있는 파일 A의 데이터 블록들에 대해 무효화를 알리고, 쓰기연산을 수행한 후 클러스터 파일 시스템의 디스크에 동기화 연산을 수행하여 물리적 장치에 영구적으로 연산내용을 반영한다(그림 8의 (5)~(9)). 이 후 데이터 블록 d1을 요청하는 호스트 D는 로크 서버 관리 기법에 따라 호스트 C로부터 로크를 얻게 되지만 쓰기 연산 직후 파일 A의 모든 데이트 블록이 무효화 된 후 이므로 해당 데이터 블록은 디스크로부터 읽어오게 된다.

클러스터 파일 시스템에서 호스트 간의 데이터의 동기화 작업은 공유 데이터의 일관성을 위해 반드시 필요하다. 본 논문에서 제안된 광역 버퍼 관리기법은 동기화 작업으로 인해 전역 버퍼를 위한 데이터 블록 목록의 무제한적 증가를 피할 수 있게 된다. 또한 클러스터 로크 관리기에 의해 관리되는 정보를 부가적으로 활용하여 광역 버퍼 관리기에 적용함으로써, 기존의 다른 캐시 기법들과 같이 광역 버퍼 관리를 위한 추가 정보의 저장 및 운영을 위한 불필요한 메시지 전송을 피할 수 있으므로 효율적이다. 특히 대규모 클러스터 환경에서 수십만대 이상의 서버들이 클러스터링 되어 있는 경우 모든 서버에 요청 데이터 블록이 존재하는지 탐색해야 하는 오버헤드가 없으므로 대규모 클러스터 환경일수록 본 논문에서 제안하는 방법은 효율적이다. 또한 각 호스트 서버들이 독립적인 광역 버퍼 기능을 운영하므로 클러스터 내의 호스트 서버의 오류로 인해 전체적인 광역 버퍼 관리 기능이 영향을 받지 않아 확장성 및 가용성을 높일 수 있다.

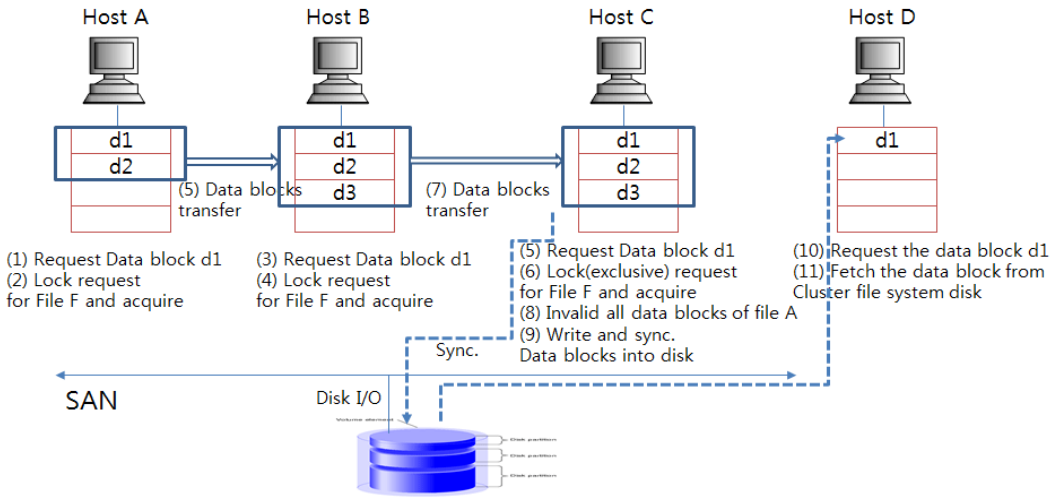

그림 8. 로크 정보를 이용한 광역 버퍼 관리 기법의 예
Fig. 8 Example of Global Buffer Management with Lock Information

Ⅳ. 결론

클러스터 파일 시스템은 SAN에 연결된 대량의 물리적 디스크 상의 데이터를 공유하는 공유 파일 시스템이다. 서로 다른 호스트가 동시에 같은 파일에 대한 읽기/쓰기를 수행할 때 상호 일관성을 제공하기 위한 기초적인 방법은 데이터가 수정되면 바로 디스크에 반영하고 다른 호스트들은 변경된 데이터를 디스크로부터 읽도록 하는 것이다. 이러한 방법은 빈번한 디스크 I/O를 발생시켜 시스템의 성능을 저하 시키는 요인이 된다. 본 논문에서 제안하는 광역 버퍼 관리기법은 다수의 호스트가 버퍼 캐시를 서로 공유할 수 있도록 함으로써 디스크 I/O 횟수를 감소시키고 데이터 블록의 접근 속도를 향상시킨다. 특히 로크 관리기에서 관리되는 정보를 활용하여 광역 버퍼 관리기에 적용함으로써, 기존 분산 시스템의 상호협조 체계로 운영되던 모든 기본기능을 제공하면서 동시에 전역 버퍼 관리를 위한 부가적인 통신 및 관리 정보를 줄인 효율적 방법이다. 향후 로크의 단위가 세분화 되는 상황에서 보다 효율적인 광역 버퍼 관리가 될 수 있도록 성능향상을 목표로 하고 있다.

참고문헌

[1] Symantc Corp.. "Veritas File System and Volume Manager", http://www. symantec.com

[2] The Red Hat Global File System, Red Hat Technical Document, http://www. redhat.com/gfs

[3] MacroImpact, Inc., "SANique Cluster Volume Manager Functional Specification", MacroImpact Technical Memo, 2008.

[4] Ghemawat, S., Gobioff, H., and Leung, S. -T. The Google File System, In 19th SOSP, Dec. 2003. pp29-43.

[5] Burrows, M. The Chubby Lock Service for Loosely-Coupled Distributed Systems, In Proc. of the 7th OSDI, 2006. 11

[6] Jeffrey Dean and Sanjay Ghemawat, "MapReduce : Simplified Data Processing on large Clusters", In Proc. of the 5th OSDI, 2004. 11

[7] 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.

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

[9] 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.

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

[11] Youhui Zhang, Weimin Zheng, User-level communication based cooperative caching, ACM SIGOPS Operating Systems Review Homepage archive, Volume 37 Issue 1, January 2003

[12] Thomas E. Anderson, Michael D. Dahlin, Jeanna M. Neefe, David A. Patterson, Drew S. Roselli, and Randolph Y. Wang. 1996. Serverless network file systems. ACM Trans. Comput. Syst. 14, 1 February 1996, 41-79

저자소개

이규웅(Kyu Woong Lee)

1986년 한국외국어대학교 이학사
1990년 서강대학교대학원 공학석사
1998년 서강대학교대학원 공학박사
1998년~2000년 한국전자통신연구원 선임연구원
2007년~2008년 한국전자통신연구원 초빙연구원
2000년~ 상지대학교 컴퓨터정보공학부 부교수
※관심분야 : 데이터베이스, 클러스터 시스템 등