병렬 파일 시스템에서의 하이브리드 채널 모델

이 윤 영[†] · 황보 준 형[††] · 서 대 화[†††]

요 약

병렬 파일 시스템은 고속의 네트웍으로 여러 대의 컴퓨터들을 서로 연결하여 컴퓨터들 간에 메시지를 주고받으면서 파일을 분산 저장하고 병렬로 읽어오는 방식으로 파일 입출력 장치의 병목현상을 해결한다. 그러나 대부분의 병렬 파일 시스템은 전달하려는 메시지의 특성을 고려하지 않은 프로토콜의 사용으로 성능저하의 문제를 가지고 있다. 이에 따라 본 논문에서 메시지 처리 방법으로 HCM(Hybrid Channel Model)을 제안한다. 본 논문에서 제안하는 HCM은 병렬 파일 시스템에서 전달되는 메시지를 그 특성에 따라 분리해서 별개의 프로토콜을 이용하여 제어 메시지와 파일 데이터 블록을 전송한다. 안정성이 검증된 TCP/IP를 이용하여 구현한 메시지 채널을 통해 제어 메시지를 고속의 데이터 전송이 가능한 VIA를 이용하여 구현한 데이터 채널을 통해 파일 데이터 블록을 각각 분리하여 처리하도록 하고 있다. HCM을 병렬 파일 시스템에 구현하고 실험해본 결과 본 논문에서 제안한 채널 모델이 상당한 성능향상을 보였다.

Hybrid Channel Model in Parallel File System

Yoon-Young Lee[†] · Jun-Hyung Hwangbo[††] · Dae-Wha Seo[†††]

ABSTRACT

Parallel file system solves I/O bottleneck to store a file distributedly and read it parallel exchanging messages among computers that is connected multiple computers with high speed networks. However, they do not consider the message characteristics and performances are decreased. Accordingly, the current study proposes the Hybrid Channel model (HCM) as a message-management method, whereby the messages of a parallel file system are classified by a message characteristic between control messages and file data blocks, and the communication channel is divided into a message channel and data channel. The message channel then transfers the control messages through TCP/IP with reliability, while the data channel that is implemented by Virtual Interface Architecture (VIA) transfers the file data blocks at high speed. In tests, the proposed parallel file system that is implemented by HCM exhibited a considerably improved performance.

키워드 : 병렬 파일 시스템(Parallel File System), VIA, 분산처리(Distributed Processing), 클러스터(Cluster)

1. 서 론

마이크로프로세서는 눈부신 발전을 이루어 클록 속도가 GHz 단위로 넘어서 대용량의 멀티미디어 정보를 처리하는데 무리가 없을 정도로 발전하였다. 또한 분산·병렬 컴퓨팅을 위한 다양한 소프트웨어 도구들이 개발되었고, 운영 체제도 분산 컴퓨팅을 위한 기능을 지원하고 있는 추세이다. 그러나 마이크로프로세서와 분산·병렬 소프트웨어 기술의 급속 발전과는 달리 입출력 장치의 발전 속도는 하드웨어의 성능을 따라잡지 못해 병목현상을 초래하고 있다. 이를 해소하기 위한 한 방안으로 입출력 장치에 병렬성(parallelism)을 도입하는 방식으로 병렬 파일 시스템(Parallel File System)에 관한 연구가 활발히 진행 중이다[1, 5, 10].

하지만 현재까지 대부분의 병렬 파일 시스템은 전달하려는 메시지의 특성을 고려하지 않고, 단일 통신 프로토콜로 모든 메시지를 처리하고 있다. 메시지의 특성에 따라 적절한 통신 프로토콜로 나누어 메시지를 전달할 수 있다면 훨씬 효과적인 메시지 처리가 가능할 것이다.

본 논문에서는 Hybrid Channel Model(HCM)이라는, 병렬 파일 시스템에서 전달되는 메시지를 제어 메시지와 파일 데이터 블록으로 나누어, 이들을 각각 TCP/IP를 이용한 메시지 채널과 VIA(Virtual Interface Architecture)[2]를 이용해서 구현한 데이터 채널을 통해 각각 분리하여 전송하는 기법을 제안한다. TCP/IP를 이용하는 메시지 채널은 일반적인 소켓 라이브러리를 이용하여 구현을 하고, 데이터 채널은 VIA 기반 위에서 고속의 데이터 전송이 가능하도록 구현하였다.

† 준 회 원 : 넷컴스토리지 스토리지연구팀
†† 준 회 원 : 경북대학교 대학원 전자공학과
††† 종신회원 : 경북대학교 전자전기공학부 교수
논문접수 : 2002년 8월 16일, 심사완료 : 2002년 12월 29일

HCM을 사용자 수준의 병렬 파일 시스템인 PFSL(Parallel File System for Linux clusters)[4]에 구현하여 성능을 평가한 결과 TCP/IP나 VIA만으로 모든 메시지를 처리한 경우보다 훨씬 좋은 성능을 발휘함을 확인할 수 있었다. 이것은 병렬 파일 시스템에서 메시지의 특성에 따라 다른 채널을 이용하여 메시지를 처리하는 것이 상당한 의미를 지님을 보여준다.

본 논문은 2장에서 병렬 파일 시스템의 특성과 고속의 메시지 전달을 위한 경량 메시징 기법에 대해 살펴본다. 3장에서는 병렬 파일 시스템의 메시지 특성을 고려한 채널 연결 모델인 HCM에 대해 서술하고, 4장에서는 HCM을 통한 파일의 읽기와 쓰기 과정에 대해 알아본다. 5장에서는 HCM을 적용한 병렬 파일 시스템의 성능을 다양한 실험을 통해 분석한 후, 마지막으로 6장에서는 실험결과를 바탕으로 결론을 내리고 앞으로의 과제를 제시한다.

2. 관련 연구

PC와 워크스테이션이 점점 더 강력해지고 고속 네트웍 하드웨어의 가격이 내려가면서, 기존의 통신 소프트웨어가 클러스터 시스템의 병목이 되는 것을 막기 위해서 클러스터 시스템의 통신에 관련된 여러 연구가 하드웨어와 소프트웨어 양쪽 측면에서 동시에 진행되어 왔다. 그 중에서 기존의 통신 프로토콜이 클러스터를 위한 용도로는 효율성이 높지 않기 때문에, 이를 대신하기 위해 새로이 개발된 것이 바로 경량 메시징 시스템이다[1].

경량 메시징 시스템에는 Beowulf, Fast Sockets, PARMA와 같은 업계 표준의 통신 인터페이스를 계속 유지하면서 높은 이식성을 제공하는 산업 표준 API 경량 메시징 시스템이 있고[1], Genoa Active Message MAchine(GAMMA), Net*, Oxford BSP Clusters, U-Net on Fast Ethernet와 같이 시스템의 최대 성능을 보장 받기 위해서 운영체제 커널에서 효율적인 저수준 통신 구조의 새로운 경량 메시징 프로토콜을 지원하는 시스템이 있다. 또한 커널 수준이 아닌 사용자 수준에서의 경량 메시징 시스템도 많이 연구 개발되고 있다. 사용자 수준의 경량 메시지 시스템에는 프랑스 Lyon 대학에서 Myrinet 네트웍 환경을 기반으로 개발한 BIP(Basic Interface for Parallelism), California San Diego 대학의 Concurrent Systems Architecture Group(CSAG)에서 HPVM(High Performance Virtual Machines) 프로젝트의 핵심 통신 계층으로 개발한 FM(Fast Messages), 버클리 대학에서 개발한 U-Net, Compaq, Intel, Microsoft의 3개의 선두 기업들이 클러스터 환경이나 SAN(System Area Network)에서 사용할 고성능의 네트웍 기술을 위한 인터페이스 표준의 필요에 따라서 제안한 VIA(Virtual Interface Architecture) 등이 있다[1, 7-9].

그러나 현재까지 대부분의 병렬 파일 시스템에서 데이터나 메시지를 고속으로 전송하는 데에만 관심을 둘 뿐 이들의 특성은 고려하지 않고 하나의 채널을 통해서만 전달하였다. 따라서 채널의 특성에 적합한 데이터나 메시지의 경우에는 효율적으로 전송이 되지만, 그렇지 않은 경우에는 필연적으로 병렬 파일 시스템의 성능에 과부하를 주게 된다.

예를 들어 병렬 파일 시스템의 제어 메시지는 그 크기가 매우 작아서 이를 고속의 데이터 전송을 지원하는 경량 메시징 기법을 이용하더라도 전체 파일 시스템의 성능 측면에서는 큰 효과를 기대할 수 없다. 그리고 대부분의 경량 메시징은 크기가 큰 데이터의 전송에 초점을 맞추어 개발되어 있어서 잦은 작은 메시지 전송시에는 신뢰성 보장을 위한 추가적인 통신 오버헤드가 발생하게 된다. 따라서 제어 메시지는 얼마나 빨리 전송하는가 보다는 얼마나 안전하게 전송하는가가 훨씬 중요함으로 해서 추가적으로 복잡한 프로토콜의 재설계가 이루어져야 한다.

병렬 파일 시스템의 파일 데이터 블록의 전송은 그 발생 회수는 제어 메시지에 비해 매우 적지만, 전체 메시지 전송량의 대부분을 차지한다. 따라서 파일 데이터 블록은 고속의 데이터 전송을 지원하는 경량 메시징 기법을 이용할 경우 전체 파일 시스템 성능을 크게 향상시킬 수 있다. 즉, 파일 데이터 블록의 전송에는 얼마나 안전하게 전송하는가보다는 얼마나 빠르게 전송하는가가 훨씬 중요한 것이다.

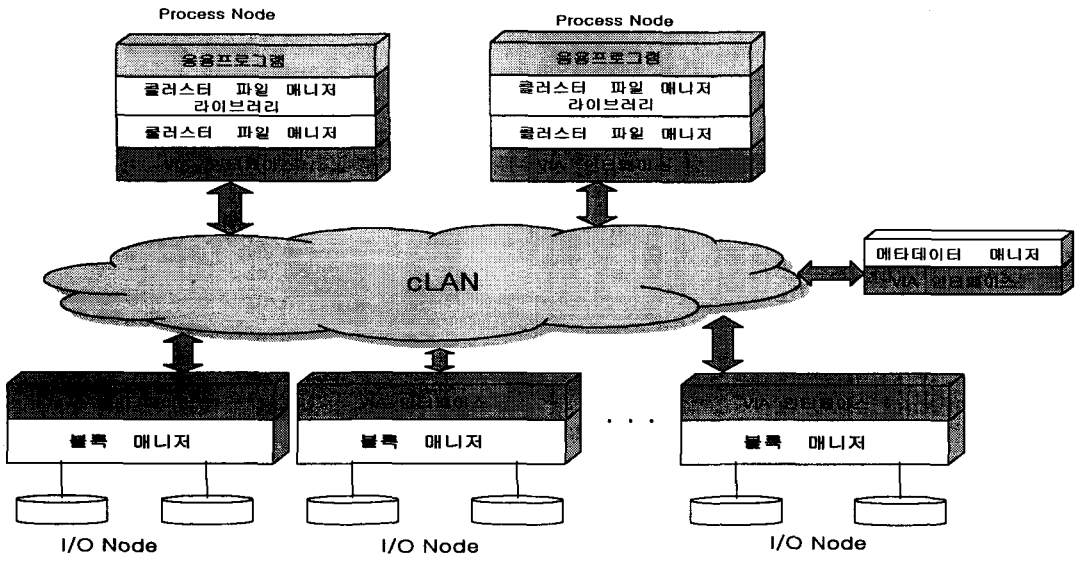
그러므로 병렬 파일 시스템에서는 메시지들의 특성 고려하여 효율적인 메시지 처리를 위한 기법에 대한 연구가 필요하며, 이러한 연구를 통해서 병렬 파일 시스템의 성능을 한층 더 높일 수 있을 것이다.

3. 병렬 파일 시스템을 위한 하이브리드 채널 모델

3.1. PFSL(Parallel File System for Linux)

본 논문에서 제안한 Hybrid Channel Model(HCM)은 사용자 수준의 병렬 파일 시스템인 PFSL(Parallel File System for Linux)에 구현되었다. (그림 1)은 PFSL의 구조이다.

PFSL은 각각 컴퓨팅 노드의 클러스터 파일 매니저(CFM)와 입출력 노드의 블록 매니저(BM), 그리고 메타데이터를 관리하는 메타데이터 매니저(MM)로 구성되어 있다. 클러스터 파일 매니저는 응용 프로그램의 요청을 받아 처리해주는 역할을 담당하며 파일 블록을 서비스해 주는 경우에는 내부의 버퍼 캐쉬를 통해서 모든 서비스를 처리한다. 블록 매니저는 클러스터 파일 매니저로부터 파일 블록 서비스 요청을 받고 해당 블록을 디스크에서 읽어 클러스터 파일 매니저로 제공하는 역할을 하며, 클러스터 파일 매니저와 마찬가지로 자신만의 버퍼 캐쉬를 두고 이를 통해 블록



(그림 1) PFSL의 구성

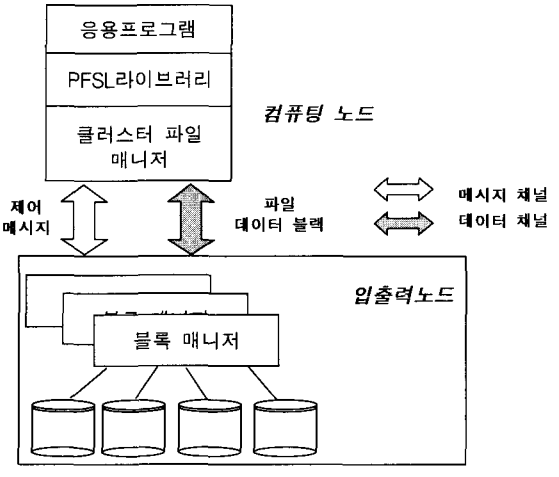
들을 클러스터 파일 매니저로 전달한다. 메타데이터 매니저는 분산 저장된 파일에 대한 모든 메타데이터를 관리하는 서버이며, 응용 프로그램이 클러스터 파일 매니저에게 파일에 대한 요청을 할 경우 클러스터 파일 매니저가 올바른 파일 데이터 블록을 서비스할 수 있도록 분산 저장된 파일에 대한 정보를 클러스터 파일 매니저에게 제공한다. 또한 고가용성을 위하여 블록 매니저가 정상적으로 동작하는지를 주기적으로 검사하는 역할도 한다.

3.2. Hybrid Channel Model(HCM)

Hybrid Channel Model(HCM)은 병렬 파일 시스템의 제어 메시지 전송을 위한 메시지 채널과 파일 데이터 블록의 전송을 위한 데이터 채널로 분리하여 전달하는 모델이다. 병렬 파일 시스템의 메시지를 특성에 따라 분리하여 전송함으로써, 메시지 전달에 소모되는 비용을 줄여 병렬 파일 시스템의 성능을 개선할 수 있다.

HCM은 (그림 2)의 클러스터 파일 매니저와 블록 매니저 간의 메시지 처리를 위한 채널 모델이다. 클러스터 파일 매니저는 블록 매니저에 제어 메시지를 통해 블록 서비스 요청과 응답을 주고받게 된다. 만약 블록 읽기와 쓰기 요청일 경우에는 제어 메시지 이외에 파일 데이터 블록도 전송하게 된다.

제어 메시지는 그 종류가 다양하고 메시지의 종류마다 크기가 조금씩 다르며, 대부분 작은 크기의 메시지로써 메시지 전송 속도는 느리지만 안정성을 보장할 수 있는 메시지 채널이 효과적이다. 이에 반해, 파일 데이터 블록은 그 크기와 형식이 고정되어 있어 매우 단순하며, 메시지의 크기가 상대적으로 크다. 그러므로 데이터 채널을 통해 파일 데이터 전송할 때 버퍼 관리가 매우 용이하고 버퍼 관리 과정에서 메시지가 잘리거나 오류가 발생할 확률이 매우 적어 안정적인 채널을 사용하는 것 보다는 전송 속도가 빠른 채널이 더 적합하다. 이에 HCM에서는 제어 메시지 전송을 위해서는 안정성이 검증된 TCP/IP를 이용한 메시지 채널을 이용하였고, 파일 데이터 블록을 위해서는 빠른 전송 속도를 보장하는 VIA를 이용한 데이터 채널을 이용하였다.



(그림 2) 하이브리드 채널 모델

3.2.1 채널의 상태

메시지 채널과 데이터 채널의 특성을 분석하기 전에 채널이 가질 수 있는 상태들을 먼저 살펴보고 각각이 가지는 의미를 파악하여, 채널의 상태 변화를 통한 채널의 특성을 파악하는 기반을 마련하고자 한다. 채널의 상태는 다음과 같이 구분할 수 있다.

- 생성

- 준비
- 전송
- 종료

생성 상태는 채널이 생성되어 연결되는 순간까지를 포함한다. 채널을 위한 디바이스의 특성을 파악하고, 채널을 위한 자원들을 할당받은 다음, 채널을 연결하기 위한 상대방 노드에 대한 정보를 얻어 연결을 시도하거나, 연결 요청을 받아 이를 수락하여 이루어진다. 채널의 준비 단계에 해당하며, 메시지 송수신을 하기 이전에 반드시 수행되어야 하는 상태이다.

준비 상태는 채널이 연결된 상태이지만 데이터 전송이 없는 상태이다. 즉 채널을 통해 연결된 상대방 노드로부터 데이터가 오기를 대기하는 상태, 데이터 전송에 앞서 전송하기 위한 데이터를 가공하고 있는 상태를 의미한다. 이 상태에서 파일 서비스 요청을 위한 메시지를 받아서, 요청 받은 서비스를 수행하게된다. 단, 파일 데이터 블록의 송수신을 위한 요청은 예외적으로 준비 상태와 전송 상태가 번갈아 가면서 서비스를 수행한다.

전송 상태는 연결된 채널을 통해 실제로 데이터를 송수신하고 있는 상태이다. 안전한 데이터 전송을 위해 흐름 제어를 하고, 받은 데이터를 버퍼링한 후 실제로 요구한 메모리 영역으로 복사하는 과정 혹은 보낼 데이터가 있는 메모리 영역에서 데이터 전송을 위한 버퍼로 복사하는 과정을 포함한다. 전송 상태에서 채널이 어떻게 동작하는 가가 채널의 성능을 크게 좌우하게 되므로, 전체 병렬 파일 시스템 성능 향상을 위해 상당한 주의를 요구하는 상태이다.

종료 상태는 이름 그대로 연결된 채널을 종료하는 상태이다. 채널을 통한 데이터 송수신이 더 이상 사용하지 않아 채널을 닫고, 채널을 위해 할당된 모든 자원을 반환하는 과정을 포함한다. 종료 상태에서 자원의 반납이 완전히 이루어지지 않을 경우에는 병렬 파일 시스템을 장시간 운영할 경우, 시스템이 패닉 상태가 되어 예상치 못한 오류가 발생할 수 있으므로 상당한 주의가 필요하다.

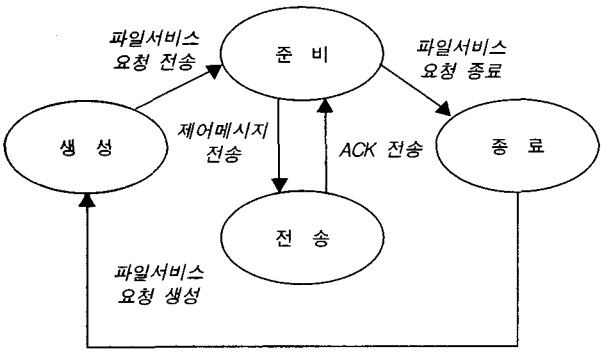
메시지 채널과 데이터 채널은 위에서 설명한 바와 같은 4가지 상태를 천이하면서 메시지 전송해 필요한 모든 작업을 수행하고 있다.

3.2.2 메시지 채널

(그림 3)은 병렬 파일 시스템에 HCM을 적용하여 제어 메시지 송수신을 하는 경우 메시지 채널의 상태 천이도이다.

메시지 채널의 목적이 제어 메시지를 전송하는 것이기 때문에 파일 서비스를 할 때마다 계속해서 하나의 채널을 생성하였다가 해제하는 일을 반복한다.

파일에 대한 요구가 생성되면 메시지 채널은 생성 상태가 되고 그 동안 파일 시스템은 연결될 노드의 주소와 포트 번호를 확인후 연결을 설정하게 된다. 채널이 생성되면 메시지 채널은 준비 상태가 되고 각 매니저는 제어 메시지를 만들거나 메시지를 받을 준비를 하게된다. 그런 후, 제어 메시지는 메시지 채널을 통해 전송이 되면서 전송 상태가 된다. 파일 서비스가 종료되면 메시지 채널을 닫게 되는데 메시지 채널을 종료하기 위해 채널은 종료 상태가 되어 채널을 위해 할당되었던 모든 자원을 반납하여 시스템 자원의 낭비를 막는다.



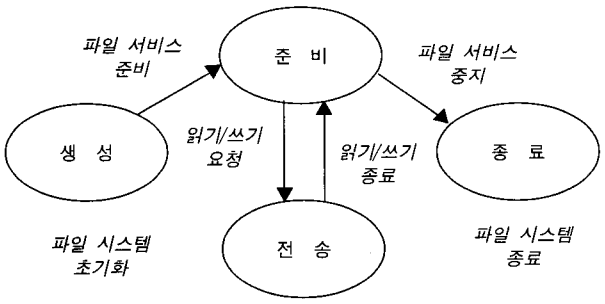
(그림 3) 메시지 채널의 상태 천이도

3.2.3 데이터 채널

데이터 채널은 용량이 큰 파일 데이터 블록을 전송하는 것이 목적이므로 파일 서비스가 요청될 때마다 열기와 닫기를 반복해서 수행하는 메시지 채널과는 달리 한번 생성이 되면 병렬 파일 시스템이 끝날 때 까지 계속 연결을 유지한다. (그림 4)는 병렬 파일 시스템에서 3HCM을 적용하여 파일 데이터 블록을 전송하는 경우의 데이터 채널의 상태 천이도이다.

파일 시스템이 초기화되면 데이터 채널은 생성 상태가 되고, 이 상태에서 병렬 파일 시스템 내에 존재하게 될 모든 데이터 채널이 생성된다. 각 데이터 채널은 미리 정해진 포트 번호를 가지고, 시스템 구성 시에 설정된 이름을 가진 호스트들의 물리 주소로 접근하여 데이터 채널을 형성한다. 연결이 설정되면 데이터 전송 준비 상태인 준비 상태가 된다. 그러다가 파일 읽기/쓰기 요청이 들어오면 전송 상태가 되어 CFM과 BM사이에서 데이터를 전송한다.

파일 데이터 블록의 전송이 끝나면 다시 데이터 채널은 준비 상태가 되어 다음에 발생할 파일 읽기/쓰기 요청을 기다리며 병렬 파일 시스템이 종료될 때까지 데이터 채널은 준비 상태와 전송 상태를 반복하면서 병렬 파일 시스템의 파일 데이터 블록의 전송을 담당한다. 병렬 파일 시스템이 서비스를 종료할 때, 데이터 채널은 종료 상태가 되어 데이터 전송을 위해 할당하였던 시스템의 자원들을 모두 반환하고, 병렬 파일 시스템 내에 연결된 모든 데이터 채널의 연결을 해제한다.



(그림 4) 데이터 채널의 상태 천이도

4. HCM이 적용된 파일 읽기/쓰기 과정

병렬 파일 시스템 전체 성능을 평가하는 기준으로 사용되는 오퍼레이션은 파일 읽기와 쓰기이다. 이 절에서는 HCM의 적용을 통하여 병렬 파일 시스템의 성능이 어떻게 개선될 수 있는 가를 알아본다.

먼저 파일의 읽기/쓰기를 제외한 오퍼레이션들이 어떤 과정으로 수행되는 지를 살펴보기로 한다. 응용 프로그램으로부터 파일 서비스 요청을 받은 클러스터 파일 매니저는 요청 받은 파일 서비스를 응용 프로그램이 넘겨 준 정보와 함께 처리한다. 이 과정에서 블록 매니저로부터 블록 서비스를 받을 필요가 있으면 블록 매니저에 블록 서비스를 요청한다. 이 때의 블록 서비스는 파일 데이터 블록을 블록 매니저에 저장하거나 읽어오는 서비스를 제외한 슈퍼 블록 관리, i-node 관리, 캐쉬 일관성 유지, 블록 접근 권한 관리, 동기화 등의 서비스를 의미한다.

블록 매니저는 해당하는 블록 서비스를 수행한 다음 그에 대한 응답을 클러스터 파일 매니저에 전달하고, 이 응답을 받은 클러스터 파일 매니저는 응용 프로그램에 메시지 채널을 통해 응답을 보내주어 해당 파일 서비스가 완료되었음을 알려준다.

일반 파일 오퍼레이션은 모두 메시지 채널을 통해서만 이루어진다. 그것은 일반 파일 시스템 오퍼레이션의 종류만큼이나 다양한 크기와 형식을 가진 제어 메시지들을 안정적으로 전송하기 위함이다. 이러한 일반 파일 시스템 오퍼레이션들을 고속의 데이터 채널을 통하여 전송하지 않는 이유는 메시지를 전송하는 것이 가지는 의미가 그다지 크지 않아 속도는 큰 문제가 되지 않음에도 불구하고, 데이터 채널을 통해 전송할 경우 다양한 크기의 메시지를 처리하기 위한 효율적인 버퍼 관리가 어렵기 때문이다.

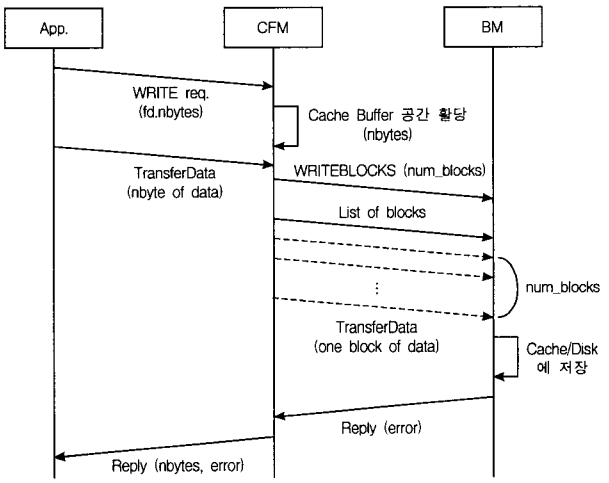
이와 달리 파일의 읽기 쓰기 과정은 HCM을 적용하여 처리하게 된다.

4.1 파일 쓰기 과정

(그림 5)는 파일 쓰기 과정을 나타낸다. 먼저 응용 프로그램(APP)이 클러스터 파일 매니저에게 쓰기를 수행하겠다는 메시지를 전달한다. 이 메시지에는 쓰기 오퍼레이션 코드, 파일의 디스크립터(fd)와 데이터의 크기가 포함되어 있다. 클러스터 파일 매니저는 넘겨받은 파일의 디스크립터를 이용해 파일의 offset을 읽어와 파일에 쓰기 시작할 부분을 결정한다.

다음으로 클러스터 파일 매니저는 캐싱을 위해 응용 프로그램이 요청한 크기의 캐쉬 버퍼를 할당한 후, 응용 프로그램이 저장할 파일 데이터를 넘겨주기를 기다린다. 파일 데이터를 넘겨받은 클러스터 파일 매니저는 그 데이터를 캐쉬 버퍼에 캐싱한다. 캐싱을 수행하면서 교체되거나 플러쉬되는 데이터 블록들은 블록 매니저에게 분산 저장을 요청한다.

데이터 블록을 블록 매니저들에 전달하기 위해, 클러스터 파일 매니저가 블록 매니저에 데이터 블록을 저장하겠다는 메시지를 전달한다. 이 메시지에는 저장할 블록의 개수가 포함되어 있다. 그런 다음 데이터 블록 리스트를 블록 매니저에 전달하여 데이터 블록을 저장할 공간을 할당하도록 한다. 블록 리스트에는 쓰려고 하는 블록의 id들이 포함되어 있다. 그 후에 블록 매니저에 파일 데이터 전달하여 해당하는 블록 id로 저장하게 한다.



(그림 5) 파일 쓰기 과정

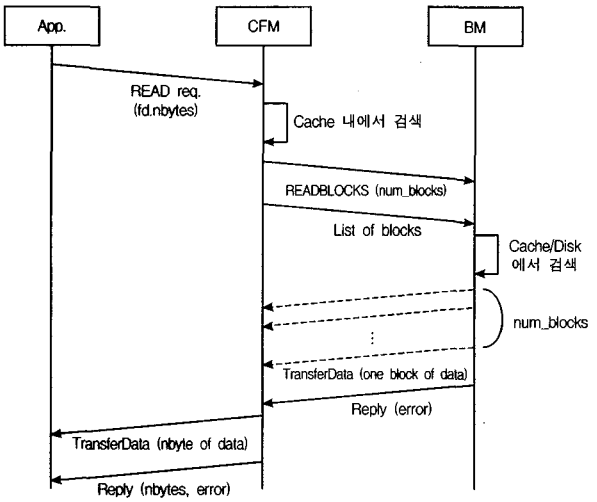
파일 데이터 블록을 전달받은 블록 매니저는 이들을 캐싱하고, 디스크에 저장한다. 파일 데이터 블록의 저장이 끝난 다음 블록 매니저에게 에러 발생 여부를 알려준다. 이 메시지를 받은 클러스터 파일 매니저는 블록 매니저가 실제로 저장한 블록의 개수와 에러 여부를 다시 응용 프로그램에 전달하여 한번의 응용 프로그램의 쓰기 요청을 완료한다.

HCM에서는 파일 데이터 블록만이 VIA로 구현된 데이터 채널을 통해 전달된다. 그러나 병렬 파일 시스템에서는 일반적으로 파일 데이터 블록의 크기가 다른 제어 메시지들보다 그 단위가 훨씬 크고, 전체 전송량도 많다. 따라서 고속의 데이터 채널을 통해 모든 메시지를 전송하지 않고, 파

일 데이터 블록을 빠른 속도로 전달하는 것만으로도 병렬 파일 시스템의 성능이 크게 개선될 수 있다. 그리고 제어 메시지를 메시지 채널을 통해 전송하기 때문에 병렬 파일 시스템 전체의 안정성을 유지할 수 있다. 그러므로 HCM을 병렬 파일 시스템에 도입하여 파일 쓰기 오퍼레이션의 전체적인 성능은 향상시키면서도, 그 안정성은 일정 수준을 유지할 수 있었다.

4.2 파일 읽기 과정

(그림 6)은 파일 읽기 요청이 처리되는 과정을 나타내고 있다. 먼저 응용 프로그램이 클러스터 파일 매니저에 읽기 요청을 위한 메시지를 보낸다. 이 메시지에는 읽기 오퍼레이션 코드, 파일 디스크립터(fd)와 데이터의 크기(n bytes)가 포함된다. 클러스터 파일 매니저는 넘겨받는 파일의 디스크립터를 이용해 offset을 구하여 파일 읽기를 시작할 위치와 데이터의 크기를 통해 얼마만큼의 데이터를 읽어올지 결정하게 된다.



(그림 6) 파일 읽기 과정

메시지를 받은 클러스터 파일 매니저는 해당하는 파일 데이터 블록이 캐쉬 내에 존재하는 지를 검사하고, 캐쉬 내에 있으면 이것을 바로 응용 프로그램에 보내주고, 그렇지 않을 경우 해당 데이터 블록이 있는 블록 매니저에 요청을 한다.

블록 매니저에 읽기 요청을 하기 위해 블록 읽기 오퍼레이션 코드와 읽으려는 블록의 개수를 전달한다. 그리고 읽어오려는 데이터 블록의 리스트를 보낸다. 블록 리스트에는 읽어올 블록의 id를 포함하고 있어서, 블록 매니저는 요청받은 블록들을 블록 매니저 캐쉬나 디스크로부터 읽어 클러스터 파일 매니저에 전달한다. 그런 다음 블록 리스트를 통해 요청 받은 블록을 읽어 오는 중에 에러 발생 여부를 클러스터 파일 매니저에 알려준다.

응용 프로그램이 요청한 파일 데이터 블록을 모두 읽은 클러스터 파일 매니저는 데이터 블록들을 하나로 묶어 응용 프로그램에 전달한다. 그 다음, 실제 읽은 크기와 읽기 오퍼레이션 수행 중 에러 여부 및 에러를 가리키는 번호를 응용 프로그램에 보내어 주는 것으로 읽기 작업을 마친다.

읽기 오퍼레이션에서도 쓰기 오퍼레이션과 마찬가지로 HCM을 적용하여 블록 매니저에서 클러스터 파일 매니저로 데이터 채널을 통해 파일 데이터 블록의 전송을 고속으로 수행하여 전체 병렬 파일 시스템의 성능을 개선하면서도 메시지 채널을 통한 안정적인 제어 메시지 전달로 파일 시스템의 안정성을 유지하도록 하였다.

5. 구현 및 실험

본 장에서는 TCP/IP로 구현한 메시지 채널과 VIA로 구현한 데이터 채널을 이용하여 HCM을 병렬 파일 시스템에 적용하였을 경우의 성능을 TCP/IP만을 이용하여 모든 메시지를 처리할 경우와 비교하여 성능 향상 정도를 확인하고자 한다.

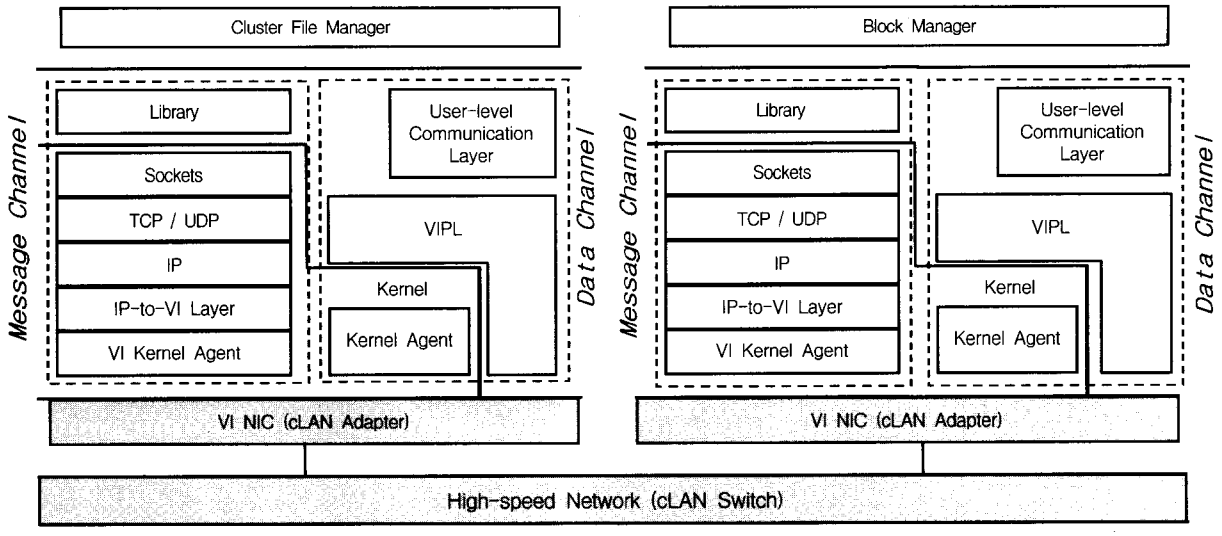
먼저 실험에 사용된 시스템 환경, 실험 방법 및 내용을 살펴본다. 그리고, 실험의 결과를 확인하고 분석한다. 실험 시 하드웨어의 성능 차이에 의한 메시지 채널과 데이터 채널의 성능 차이가 발생하지 않도록 하기 위해서, TCP/IP와 VIA 모두 동일한 cLAN 어댑터와 cLAN 스위치를 사용하였다.

5.1 구현 환경

HCM은 병렬 파일 시스템의 메시지 전달을 위하여 메시지 채널과 데이터 채널의 두 개의 채널을 두고, 전달하고자 하는 메시지의 특성을 고려하여 두 가지 채널 중 적합한 채널을 통하여 메시지를 전달하도록 하는 방식이다. 메시지 채널은 TCP/IP를 이용하여 구현하였으며, 데이터 채널은 VIA를 이용하여 구현하였다. TCP/IP를 사용하는 메시지 채널은 소켓 라이브러리를 이용하여 쉽게 구현할 수 있었고, cLan상에서 사용하기위해 IP-to-VI 레이어가 추가 되었다.

VIA를 이용하는 데이터 채널은 VIA에서 제공하는 기본 라이브러리인 VIPL(Virtual Interface Provider Library)[2]을 이용하여 구현하였다. (그림 7)은 CFM과 BM사이에서 채널 역할을 하는 HCM의 구조를 보여준다.

실험은 병렬 파일 시스템인 PFSL(Parallel File System for Linux clusters)을 이용하여 이루어졌다. HCM은 PFSL의 클러스터 파일 매니저와 블록 매니저간의 메시지 전송에 적용되었으며 성능 측정을 위한 응용 프로그램을 PFSL 라이브러리로 작성하여 클러스터 파일 매니저가 운용 중인 프로세서 노드에서 수행하였다. 네트웍의 구성은 실험에 사용된 노드 사이에만 지역 네트웍을 구성하여 외부 네트웍의 영향을 배제하였다. 다음은 실험 환경과 관계된 세부사항이다.



(그림 7) HCM의 구조

- 운영 체제 : Linux Kernel version 2.2.14
- 파일 시스템 : Linux 기반의 사용자 수준 병렬 파일 시스템인 PFSL(Parallel File System for Linux clusters)
- 클러스터 파일 매니저 : 1 노드
- 블록 매니저 : 1 노드~4 노드
- Network Interface Controller : cLAN 1000(Emulex)
- Network Switch : cLAN 5000(Emulex)
- CPU : Pentium-Ⅲ 600
- RAM : 256MB
- HDD : 20GB

실험에 사용된 TCP/IP와 VIA의 성능 차에 하드웨어적인 특성의 영향을 없애기 위해 동일한 하드웨어인 cLAN을 사용한다. TCP/IP는 기존의 응용 프로그램을 수정하지 않고, LANE(LAN Emulatiom)라는 IP-to-VI 계층을 두어 cLAN 네트웍 장치를 이용하도록 하는 방식으로 사용한다[3, 6].

5.2 실험 방법 및 내용

TCP/IP로 구현한 메시지 채널과 VIA로 구현한 데이터 채널을 이용하여 HCM을 병렬 파일 시스템에 적용하였을 경우의 성능을 TCP/IP만을 이용하여 모든 메시지를 처리할 경우와 비교하여 성능 향상 정도를 확인하였다.

먼저 병렬 파일 시스템의 성능에 큰 영향을 끼치는 입출력 노드 수를 변화시키면서 연속적인 읽기 요청과 쓰기 요청을 처리하는 성능을 측정하여, HCM을 적용한 경우의 파일 시스템의 성능과 TCP/IP만을 사용하는 파일 시스템의 성능을 비교하였다. 이 실험에 사용된 파일의 크기는 20MB이고, 하나의 파일 데이터 블록의 크기를 16KB로 정하였다. 응용 프로그램이 읽기 요청과 쓰기 요청을 할 때 요청하는 데이터의 크기는 256KB로 고정하였다. 실험의 변수는 입출력 노드의 수이며 1대~4대로 변화시키면서 각각의 성능을 평균 대역폭과 전체 소요시간을 측정하여 확인하였다.

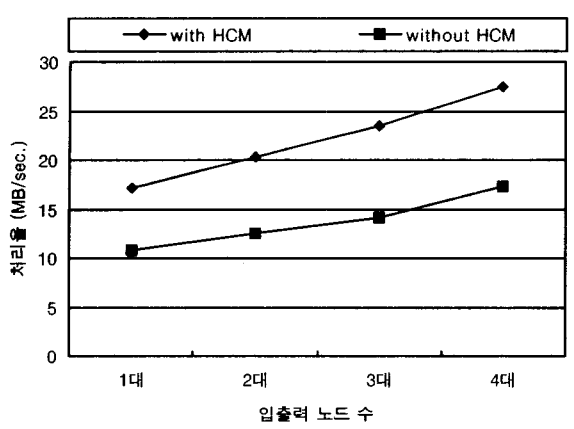
다음은 응용 프로그램에서 한번에 요청하는 데이터의 양을 변화시키면서 실험을 하였다. 이 실험은 제어 메시지의 발생 정도에 따라 병렬 파일 시스템의 성능이 변화할 것이라는 예상에서 시작되었다. 실험을 위해 입출력 노드를 3대로 고정시켰으며, 하나의 파일 데이터 블록의 크기를 16KB, 전체 파일의 크기를 20MB로 고정하였다. 이 때 한번에 요청하는 데이터의 양을 16KB~512KB로 변화시키면서 파일 읽기와 쓰기의 성능을 평균 대역폭과 전체 소요시간을 측정하여 확인하였다.

마지막으로 하나의 파일 데이터 블록의 크기가 병렬 파일 시스템의 성능에 미치는 영향을 실험하였다. 실험을 위해 입출력 노드를 3대로 고정시켰으며, 응용 프로그램이 한번에 요청하는 데이터의 크기를 256KB, 전체 파일의 크기를 20MB로 고정하였다. 이 때 파일 데이터 블록의 크기가 8KB, 16KB, 32KB, 64KB일 경우를 각각 측정하여 읽기와 쓰기에 소요되는 시간과 각 경우의 평균 대역폭을 측정하여 성능을 확인할 수 있었다.

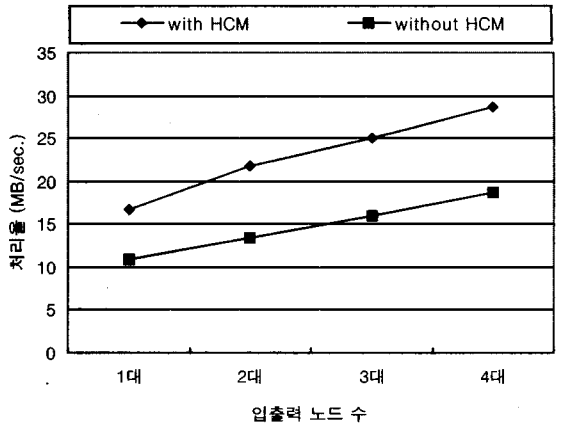
5.3 실험 결과 및 분석

병렬 파일 시스템의 성능은 파일의 열기부터 닫기까지의 시간을 측정한 것을 바탕으로 읽기와 쓰기의 성능을 평가하였다. 모든 경우에 대해서 HCM을 적용한 병렬 파일 시스템의 성능과 TCP/IP만을 사용하는 병렬 파일 시스템의 성능을 측정하여 비교하였다.

(그림 8)은 1대에서 4대까지의 I/O노드 수 변화에 따른 성능을 보여주고 있다. 전체 파일의 크기는 20MB이고 256KB씩 쓰기를 요청하며 입출력 노드에 저장될 때의 단위 블록의 크기는 16KB이다.

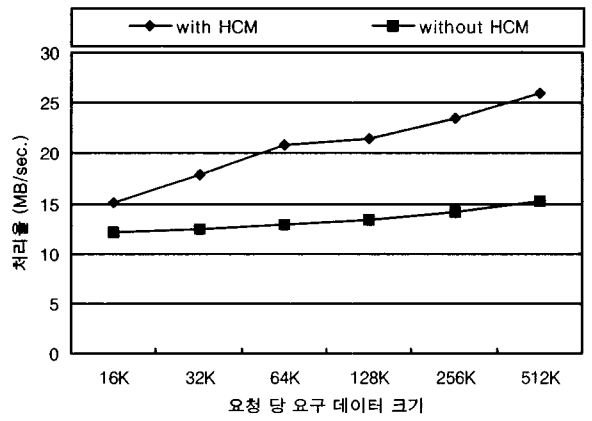


(a) 쓰기(Writing)

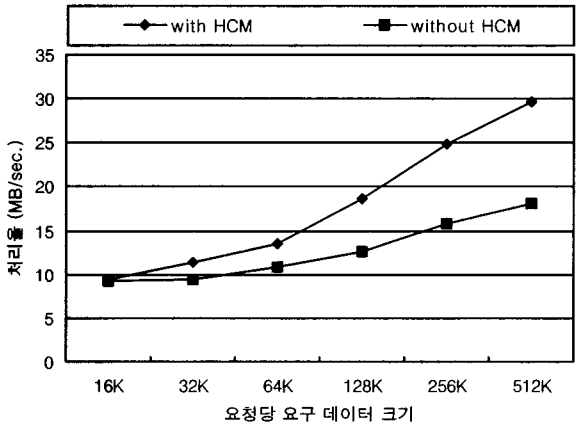


(b) 읽기(Reading)

(그림 8) 입출력 노드 수에 따른 읽기, 쓰기 성능

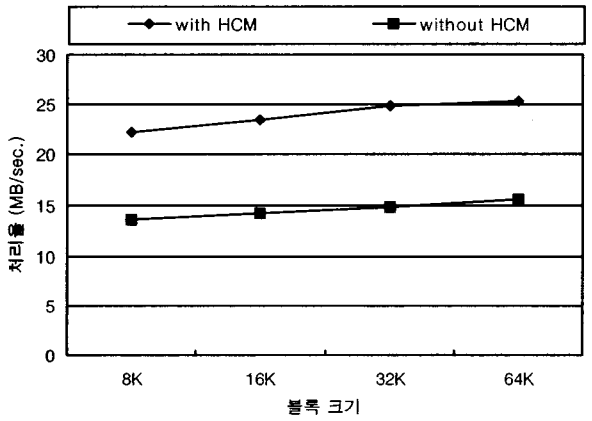


(a) 쓰기(Writing)

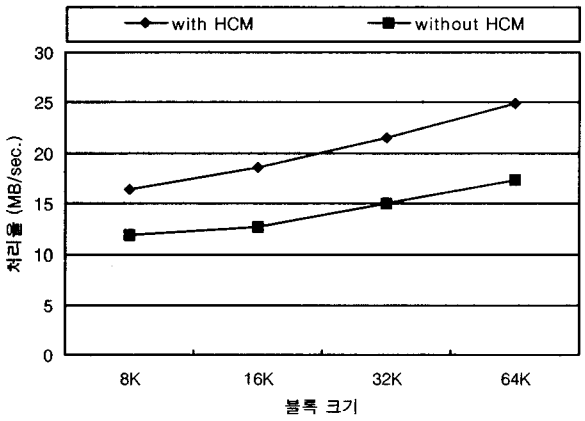


(b) 읽기(Reading)

(그림 9) 요청 크기에 따른 읽기, 쓰기 성능



(a) 쓰기(Writing)



(b) 읽기(Reading)

(그림 10) 블록 크기에 따른 읽기, 쓰기 성능

입출력 노드 수를 증가시킬수록 작업이 분산되어 쓰기와 읽기의 속도가 증가하는 것을 확인할 수 있었다. TCP/IP만을 이용하여 메시징하는 경우보다 HCM을 적용하였을 때 쓰기의 경우 약 59%~65%, 그리고 읽기의 경우 약 52%~60%의 성능 향상 효과가 있었다.

(그림 9)는 응용 프로그램이 한번에 요청하는 데이터의 크

기를 16KB~512KB로 변화시키면서 읽기와 쓰기 성능을 측정한 결과이다. 사용된 파일의 크기는 20MB이고 I/O노드는 3대, 그리고 입출력 노드에 읽고 쓰는 단위 블록 사이즈는 16KB이다. 그림에서 보듯이 단위 요청 사이즈가 클수록 성능이 향상됨을 알 수 있다. 즉, 단위 요청 사이즈가 증가할수록 제어메세지의 수가 줄어듦을 나타낸다. HCM을 적용한 병렬 파일 시스템의 성능이 그렇지 않은 시스템보다 읽기와 쓰기에서 각각 약 1.5%~64%, 23%~69%의 성능향상이 있었다.

(그림 10)은 단위 블록 사이즈를 8KB에서 64KB까지 변화시켜가면서 성능측정을 한 결과이다. 실험에 사용된 파일의 크기는 20MB이고 단위 요청 크기는 256KB, 그리고 입출력 노드의 수는 3대이다. 그림에서 보듯이 단위 블록 사이즈가 커지면 커질수록 성능이 향상되는 것을 볼 수 있다. 그러나 성능 향상의 정도가 (그림 9)에서 보다는 높지 않음을 알 수 있는데 이는 단위 블록 사이즈가 변화하더라도 제어메세지의 수는 줄어들지 않음을 뜻한다.

HCM을 적용한 병렬파일 시스템의 성능이 그렇지 않은 시스템보다 읽기와 쓰기에서 각각 약 37%~46%, 62%~67%의 성능향상이 있었다.

6. 결론 및 향후 과제

본 논문은 메시지의 특성을 고려한 효율적인 메시지 전달을 위한 채널 모델로서 Hybrid Channel Model(HCM)을 제안하였다. HCM은 병렬 파일 시스템의 메시지를 그 특성에 따라 제어 메시지와 파일 데이터 블록으로 나누고, 이들을 각각 메시지 채널과 데이터 채널을 통해 전달한다. 메시지 채널은 다양한 종류의 제어 메시지를 안정적으로 전달하기 위해 TCP/IP를 이용하여 구현하였으며, 데이터 채널은 파일 데이터 블록을 고속으로 전송하기 위해 Virtual Interface Architecture(VIA)를 이용하여 구현하였다.

HCM을 병렬 파일 시스템인 PFSL(Parallel File System for Linux clusters)에 적용하여 병렬 파일 시스템의 성능을 측정하였다. 이 때 HCM을 적용하지 않고 TCP/IP만을 이용하여 메시지를 전달하는 경우와 그 성능을 비교하였다.

병렬 파일 시스템의 입출력 노드의 개수의 변화에 따른 성능과 응용 프로그램이 한번에 요청하는 데이터의 크기 변화에 따른 성능, 그리고 입출력 노드에 저장되는 단위인 블록의 크기에 따른 성능의 변화를 측정하고 이를 분석하였다. 대부분의 경우 TCP/IP만을 이용해 메시지를 전달하는 경우보다 HCM을 병렬 파일 시스템에 적용한 경우 그 성능이 경우에 따라 적게는 20%~30%에서 많은 경우는 70%~80%정도 그 성능이 개선됨을 확인할 수 있었다. 그리고 실험 결과를 통해서 HCM을 적용한 병렬 파일 시스템에서 좋은 성능의 응용 프로그램을 작성하기 위해 고려할 사항에 대해서도 발견할 수 있었다.

병렬 파일 시스템의 성능을 위한 향후 과제로는 논블로킹(non-blocking) 모드로 동작하는 입출력을 구현하기 위한 연구를 통해 파일 입출력 속도를 개선하고, 이를 통하여 확장성(scalability)을 보완하여야 한다. 그리고, VIA로 구현된 채널에 자가 점검 및 자가 복구 기능에 관한 연구를 통해 데이터 채널의 안정성을 높일 수 있어야 한다.

참 고 문 헌

[1] Rajkumar Buyya, "High Performance Cluster Computing," Prentice Hall, Vol.1, 1999.

[2] "Virtual Interface Architectrue Specification," draft version 1.0, 1997.

[3] J. S. Kim, K. K. Kim, and S. I. Jung, "SOVIA : A User-level Sockets Layer Over Virtual Interface Architecture," Proceedings of the 3rd IEEE International Conference on Cluster Computing (IEEE CLUSTER 2001), Newport Beach, CA, USA, pp.399-408, October, 2001.

[4] J. Cho, C. Kim, and D. Seo, "A Parallel File System Using Dual Cache Scheme and Prefetching," The 2000 International Conference on Parallel/Distributed Processing Techniques and Application (PDPTA2000), June, 2000.

[5] A. Purakayastha, C. S. Ellis, D. Kotz, N. Nieuwejaar and M. Best, "Characterizing Parallel File-access Patterns on a Large-scale Multiprocessor," In Proc. of the Ninth International Parallel Processing Symposium, pp.165-172, April, 1995.

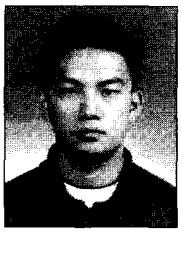
[6] "cLan for Linux, Software User's Guide," Emulex, 2001.

[7] L. Iftode, "Home-based Shared Virtual Memory," PhD thesis, Princeton University, 1998.

[8] M. LAuria, S. Pakin, and A. Chien, "Efficient Layering for High Speed Communication : Fast Messages 2.x," In Proc. 7th IEEE Int'l Symp. HPDC-7, Chicago, IL, July, 1998.

[9] M-VIA : A High Performance Modular VIA for Linux, http://www.nersc.gov/research/FTG/via/.

[10] M. Baker, "Cluster Computing White Paper," University of Portsmouth, UK, December, 2000.



이 윤 영

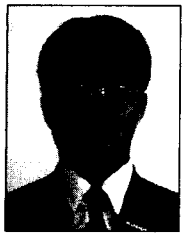
e-mail : yylee@netcomstorage.com

2000년 경북대학교 전자전기 공학부(학사)

2002년 경북대학교 전자공학과 (공학석사)

2002년~현재 넷컴스토리지 스토리지연구팀

관심분야 : 병렬파일 시스템, 클러스터, 분산처리



황 보 준 형

e-mail : bluesky@palgong.knu.ac.kr

2000년 경북대학교 전자전기 공학부(학사)

2002년 경북대학교 전자공학과(공학석사)

2002년~현재 경북대학교 전자공학과 박사과정

관심분야 : 보안OS, 병렬파일시스템, 클러스터

서 대 화

e-mail : dwseo@ee.knu.ac.kr

1981년 경북대학교 전자공학과(학사)

1983년 한국과학기술원 전산학과(석사)

1993년 한국과학기술원 전산학과(박사)

1981년~1995년 한국전자통신연구소 시스템 S/W 연구실 근무

1998년~1999년 University of California Irvine 연구 교수

1995년~현재 경북대학교 공과대학 전자전기공학부 부교수

관심분야 : 병렬분산처리, 운영체제, 병렬처리, 컴퓨터 구조