Fast Ethernet 환경에서 병렬 멀티미디어 파일 시스템의 설계와 구현

박 성 호[†]·김 광 문[††]·정 기 동[†††]

요 약

대용량 멀티미디어 미디어 서버를 구성함에 있어 I/O 병목현상을 극복하기 위하여 저장 서버들과 제어 서버로 구성되어진 2계층 분산 클러스터 서버구조가 많이 사용된다. 2 계층 분산 클러스터 서버는 부하 균등, 대역폭 관리 및 저장 서버의 관리 측면에서 유리한 반면, 저장 서버와 제어 서버간의 통신 오버헤드를 발생시킨다. 이러한 오버헤드를 줄이기 위해서는 저장 서버에서 읽은 미디어 데이터를 제어 서버를 거치지 않고 직접 클라이언트에 전송할 수 있어야 한다. 그리고, 저장 용량을 확장하거나 손상된 디스크를 교체하는 경우를 대비하여 분산 클러스터 서버는 다양한 성능의 이기종 디스크를 지원하여야 한다. 또한, I/O 장치와 운영체제가 빠르게 발전됨에 따라 미디어 서버는 새로운 I/O 장치 및 운영체제 등에 쉽게 이식될 수 있어야 하고, 응용 소프트웨어 개발자가 시스템의 환경에 따라 블록크기, 데이터 배치정책, 사본 정책 등을 유연하게 조절할 수 있어야 한다. 본 논문에서 위에서 언급한 멀티미디어 서버의 요구를 고려하여 Fast Ethernet 환경에서 병렬 멀티미디어 파일 시스템(PMFS : Parallel Multimedia File System)을 설계 및 구현하고 실험을 통해 PVFS(Parallel Virtual File System)와 성능을 비교 분석하였다. 이 실험의 결과에 따르면 PMFS는 멀티미디어 데이터에 대하여 PVFS보다 3%~15%의 향상된 성능을 보였다.

Design and Implementation of the Parallel Multimedia File System on Fast Ethernet

Seong-Ho Park[†]·Gwang-Moon Kim[††]·Ki-Dong Chung[†††]

ABSTRACT

The two-layered distributed clustered server architecture consisting of a control server and a group of storage servers has been widely used to support multimedia file systems. With this kind of server architecture, it is easy to maintain disk load balancing and scalable disk bandwidth. However, it brings about the communication overhead between the control server and storage servers. Therefore, to reduce such overhead, media data should be transmitted from the storage servers to the users directly without the intervention of the control server. In addition, the file system must support the effective data placement and disk scheduling policies and should be designed to support portability and flexibility in any hardware or software environment to cope with rapidly developed hardware and software technologies. So, to fulfill such requirements described in the above statements, we designed and implemented a parallel file system named PMFS (Parallel Multimedia File System) on Fast Ethernet and analyzed its performance by comparing it with PVFS (Parallel Virtual File System). As a result of our experimental simulation, we came to a conclusion that the PMFS shows better performance than the PVFS does in real time data processing.

키워드 : multimedia, file system, dustered system, VoD system

1. 서 론

최근 컴퓨터와 통신기술의 발달로 인해 다양한 멀티미디어 응용 소프트웨어에 대한 연구가 활발히 진행되고 있다. 현재 인터넷의 대중화로 인해 인터넷을 기반으로 한 다양한 웹 멀티미디어 응용 소프트웨어가 연구되어 상용화되고 있고, NOD(News On Demand), VOD(Video On Demand), Digital Library 등과 같은 멀티미디어 응용들도 많은 연구와 개발이 이루어지고 있다.

이러한 멀티미디어 응용 소프트웨어들은 텍스트나 이미지와 같은 비실시간 데이터 뿐만 아니라 오디오나 비디오와 같은 실시간 데이터를 다루고 있으며, 사용자의 접근 패턴을 예측하기 어려운 특성을 지닌다[4, 5, 22, 24].

일반적으로 대용량 멀티미디어 미디어 서버를 구성할 때, I/O 병목현상 때문에 여러 개의 디스크를 클러스터화 하여 디스크 배열의 형태로 구성하는데[1, 3, 8], 부하 균등과 대역폭 측면에서 유리한 2계층 분산 클러스터 서버 구조에서는 저장 서버와 제어 서버간의 통신 오버헤드가 발생한다[22]. 이러한 오버헤드를 줄이기 위해서는 저장 서버에서

† 준 회 원 : 부산대학교 대학원 전자계산학과
†† 준 회 원 : LG전자 핵심망연구소 운영체계실 연구원
††† 종신회원 : 부산대학교 전자계산학과 교수
논문접수 : 2000년 12월 13일, 심사완료 : 2001년 1월 31일

읽은 미디어 데이터를 제어 서버를 거치지 않고 직접 클라이언트에 전송할 수 있어야 한다. 그리고, 저장 용량을 확장하거나 손상된 디스크를 교체하는 경우에 동종 디스크의 생산이 중단 되었거나 디스크 기술의 발전에 따른 가격대 성능 비율을 고려하여 보다 나은 새로운 기종의 디스크를 선택하는 경우가 있기 때문에 각각의 디스크는 다양한 성능의 이기종 디스크로 구성될 수 있어야 한다[18, 19]. 또한, I/O 장치와 운영체제가 빠르게 발전됨에 따라 미디어 서버는 새로운 I/O 장치 및 운영체제등에 쉽게 이식될 수 있어야 하고, 응용 소프트웨어 개발자가 시스템의 환경에 따라 블록크기, 데이터 배치정책, 사본 정책 등을 유연하게 조절할 수 있어야 한다[21].

본 논문에서 구현한 병렬 멀티미디어 파일 시스템(PMFS : Parallel Multimedia File System)은 이식성, 유연성 그리고 확장성을 고려한 파일 시스템으로써, 2계층 분산 클러스터 서버 구조를 가지고 있다. 그리고 제어 메시지와 RTP(Real Time Transfer Protocol)[16]을 기반으로 하여 저장서버에서 읽은 데이터를 제어 서버를 거치지 않고 직접 클라이언트에 전송하고 있으며, 멀티미디어를 효과적으로 지원하기 위한 데이터 배치기법과 스케줄링 기법을 사용한다.

본 논문은 다음과 같이 구성되어있다. 2장에서는 기존에 연구되었던 다양한 형태의 병렬 파일 시스템과 서버구조, 데이터 전송기법, 데이터 배치기법 그리고 실시간 스케줄링 기법에 대해 알아본다. 3장에서는 PMFS의 주요 특징들과 구조, 그리고 구현상의 문제점에 대해 설명하고, 4장에서는 성능을 평가하고 분석한다. 마지막 5장에서는 결론을 맺고 향후 연구방향을 제시하였다.

2. 관련 연구

현재 많은 연구와 개발이 이루어지고 있는 클러스터 시스템은 데이터의 처리 방식에 따라 각각의 노드가 개별적으로 처리하는 분산 시스템(Distributed System)과 전체 노드가 하나의 서버처럼 처리하는 병렬 시스템(Parallel System)으로 구분된다. 그리고, 병렬 시스템은 구현 목적에 따라 병렬처리(Parallel Processing)와 병렬 입출력(Parallel I/O) 시스템으로 나눈다. Beowulf나 Now(Network of Workstation) 시스템[9]은 상대적으로 느린 여러 개의 저가격 시스템으로 빠른 계산을 수행하는 병렬처리 시스템이고, Symphony[12]나 Tiger Shark시스템[17]은 빠른 데이터 입출력을 위한 병렬 입출력 시스템이다. 병렬 입출력 시스템은 다시 병렬 입출력 라이브러리(Parallel I/O Library)와 병렬 파일 시스템(Parallel File System)으로 구분된다. 병렬 입출력 라이브러리는 병렬 프로그래밍 모델(Parallel Programming Model)에 적합한 인터페이스를 가지고 응용 프로그램 수준(Application Level)에서 병렬적으로 입출력을 수행할 수 있도록 한 것으로 MPI-IO 표준을 따른 ROMIO[2]와 HPF언어에 근거한 PASSION(Parallel And Scalable Software for I/O)[6]등이 있다. 특히, 병렬 파일 시스템은 파일 시스템이 요구하는 기본적인 입출력 인터페이스에 따라 구현된 것으로 다양한 종류의 병렬 파일시스템이 연구 개발되고 있다.

본 장에서는 기존의 병렬 파일 시스템에 대해 종류와 병렬 파일 시스템이 구동 되는 클러스터 서버의 구조, 데이터 전송기법 그리고 데이터 배치기법에 대해 살펴본다.

2.1 병렬 파일 시스템(Parallel File System)

기존에 연구 개발된 병렬 파일 시스템에는 일반적인 목적의 병렬 파일 시스템과 연속미디어를 지원하기 위한 병렬 파일 시스템이 있다. 일반적인 목적의 병렬 파일 시스템은 과학연구나 통계시스템과 같은 비실시간 응용이나, 범용적인 목적을 위해 구현되었으며, 이 중 대표적인 것은 Galley[11], PVFS(Parallel Virtual File System)[21], CFS(Concurrent File System), Vesta, GFS(Global File System 등이 있다[6]. 그러나 이들 일반적인 목적의 병렬 파일 시스템은 실시간성을 지닌 연속미디어 데이터 응용에 적용하기에는 적합하지 않다. 또한 대용량 실시간 연속미디어 데이터를 지원하기 위해 구현된 병렬 파일 시스템으로는 Symphony, Tiger Shark, Felli, Mitra, Video Server Array, Lancaster CMSS, Calliope, Clockwise, MiPFS 등이 있으며, Symphony와 Clockwise는 실시간 연속 미디어와 이미지나 텍스트와 같은 비실시간 미디어를 동시에 지원하는 멀티미디어 파일 시스템이다[6]. 그러나 위에서 언급한 클러스터 기반의 연속 미디어 서버를 구축하기 위해서는 별도의 Internal High Speed Communication Network이 요구됨으로써 클러스터 멀티미디어 서버의 구축 시 높은 비용이 요구된다.

2.2 클러스터 서버 구조

클러스터 기반 멀티미디어 저장서버의 구조는 제어 서버와 저장 서버의 물리적 위치에 따라 수평구조(Flat Architecture)와 2계층 구조(2-Tier Architecture)로 나누고, 미디어 객체의 배치 정책에 따라 독립 구조(Independent Architecture)와 분산 구조(Distributed Architecture)로 분류된다[22].

2.2.1 제어 서버와 저장 서버의 물리적 위치에 따른 분류

- 수평 구조 : 제어 서버와 저장 서버가 각 노드에 존재하여 모든 노드가 동일하게 동작하는 구조로서 2계층 구조(2-Tier Architecture)에 비해 노드간의 통신 오버헤드가 적은 장점을 지닌다(예 : Fellini, Video Server Array).

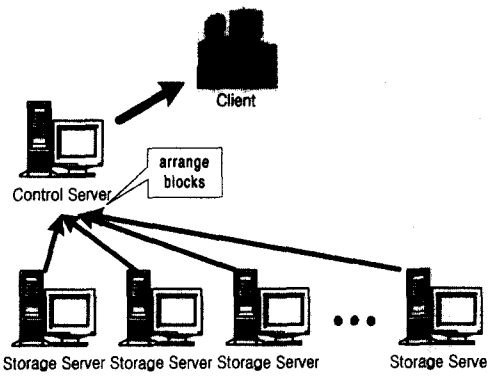
- 2계층 구조 : 제어 서버와 저장 서버가 물리적으로 다른 노드에 존재하여 각각의 노드가 다르게 동작하는 구조로서 부하균등 측면에서 수평 구조에 비해 유리하나 노드간 통신 오버헤드가 크고 제어 서버에 부하가 집중되는 현상이 발생한다(예 : Tiger Shark, Mitra).

2.2.2 미디어 개체에 대한 데이터 배치정책에 따른 분류

- 독립 구조 : 미디어 개체가 단일 노드 내에 저장되어 미디어 데이터에 대한 조작이 노드 단위로 수행되는 구조로서 미디어 개체에 대한 관리가 분산 구조에 비해 용이하나 서버의 확장 어렵다(예 : Fellini, Mitra).
- 분산 구조 : 미디어 개체가 여러 서버 노드에 분산 저장되는 구조로서 미디어 데이터에 대한 조작을 위해 서버간 동기화가 필요하나 서버의 확장이 용이한 장점을 지닌다(예 : Tiger Shark, Video Server Array).

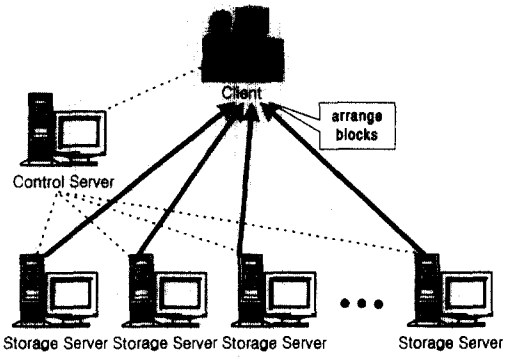
2.3 내부 데이터 전송 구조

내부 데이터 전송을 위한 데이터 전송 구조로는 Centralized Shared Data구조와 Distributed Message 기반 구조로 나눌 수 있다. Centralized Shared Data 구조에서는 전체 저장 서버의 데이터가 하나의 저장 서버의 데이터처럼 공유되고 모든 데이터의 입출력이 제어 서버를 경유하여 이루어진 구조이다. 클러스터 서버가 2계층 분산 구조를 가질 때 Centralized Shared Data구조에서 제어서버의 NIC(Network Interface Card)은 전체 시스템의 병목(Bottleneck)이 되어 성능 저하를 초래한다. 그리고, 전체 저장 서버들이 수용할 수 있는 Stream수 만큼의 추가적인 메모리가 필요하다. 이러한 구조의 대표적인 예로는 PPFS, Galley, PVFS등이 있다(그림 1).



(그림 1) Centralized Shared Data 전송 구조

Distributed Message 기반 구조는 저장 서버의 데이터가 제어 서버를 거치지 않고 제어 메시지에 의해 클라이언트로 전송이 이루어지는 구조이다. 이러한 구조는 제어서버의 부하가 집중되는 현상을 막을 수 있으나, 분산 저장된 연속 미디어 데이터의 정렬을 위하여서는 클라이언트가 각각의 저장서버에서 전송된 데이터를 블록의 순서에 따라 재조합해야 하는 오버헤드가 발생하고, 이를 처리하기 위한 별도의 클라이언트 프로그램이 필요함으로 범용성이 떨어지는 문제점이 있다(그림 2).



(그림 2) Distributed Message 기반 전송구조

2.4 데이터 배치기법

대용량 연속 미디어 데이터를 부하균등을 고려하여 다수의 디스크에 분산 배치하여 서버의 동시 사용자수를 늘리려는 다양한 연구들이 진행되고 있다. 다수의 디스크간 데이터를 배치하는 기법에 대한 연구는 크게 동일 기종 디스크간 데이터 배치기법, 이기종 디스크간 데이터 배치기법, 고장 허용성을 보장하기 위한 데이터 배치기법 그리고 시작블록의 위치에 따른 배치기법 등으로 나눈다[24].

2.4.1 디스크간 데이터 배치기법

동일 기종 디스크간의 부하균등을 위한 데이터 배치기법은 미디어 데이터의 블록들을 각각의 디스크에 배치하는 순서에 따라 다양한 기법들이 연구되고 있다. 몇 가지 주요한 배치기법을 살펴보면 다음과 같다.

Round-Robin[3, 20] 기법은 가장 기본적인 배치기법으로 데이터 블록을 각각의 디스크에 순서대로 돌아가며 배치하는 기법이다.

Staggered Striping[13, 15]기법은 시작 응답시간을 줄이기 위해 시작 블록의 위치를 하나씩 이동시키는 기법이다. Round-Robin과 Staggered Striping기법은 미디어 데이터 블록이 규칙적으로 배열되어 하나의 디스크에 사용자 요구충돌이 발생했을 때, 연속적인 요구충돌이 발생하는 문제점이 있다[23.]

Random Permutation[10]과 DIS(Diverse Interval Striping)[23]기법은 이를 해결하기 위해 연구되었으며 Random Permutation은 중복되지 않는 디스크 순열을 모두 구하여 임의적으로 그 순열 중 하나를 선택하여 배치하는 기법으로 전처리 시간이 많이 소요되는 문제점이 있고, DIS는 디스크사이의 간격을 규칙적으로 변화시키는 방법으로 Random Permutation의 전처리 시간을 줄이기 위해 연구된 배치 기법이다[23].

RIO(Randomized I/O)[4,5]기법은 미디어 블록들을 임의적으로 각각의 디스크에 배치하는 기법으로 사용자의 Interactivity가 높은 대화형 멀티미디어 응용과 혼합 미디어 데이터를 이기종 디스크에 배치했을 때 좋은 성능을 보이는 배치 기법이다[14].

2.4.2 시작블록의 위치에 따른 배치기법

혼합 미디어 데이터를 저장하는 서버에서는 실시간 미디어 데이터와 비실시간 데이터가 동시에 요구될 뿐만 아니라 사용자 접근 패턴이 임의적으로 나타나기 때문에 모든 미디어 개체의 첫번째 블록을 고정된 위치의 디스크에 저장하는 고정 시작블록(Static Starting Block) 데이터 배치기법[24]의 경우 시작블록에 저장되는 디스크에 부하가 집중될 수 있다. 따라서, 시작 지연시간과 부하 균등을 위해 시작 블록을 임의적으로 배치하는 임의 시작블록(Random Starting Block) 데이터 배치기법[24]이 있다.

PMFS는 Round Robin, Staggered Striping 그리고 DIS 데이터 배치정책을 지원하며 시작 블록이 임의적으로 배치되는 배치정책을 사용하고 있고, 결함 허용 배치 정책으로는 Mirroring Base 배치 정책으로 디스크 실패 시에도 연속적으로 서비스가 가능할 수 있도록 설계되었다.

2.5 실시간 스케줄링 기법

실시간 데이터를 스케줄링하기 위한 알고리즘으로는 EDF(Earliest Deadline First), LLF(Least Laxity First), RMS (Rate Monotonic Scheduling), QEDF(QoS-driven EDF), QLLF(QoS-driven LLF)등이 있다[25].

3. PMFS의 설계와 구현

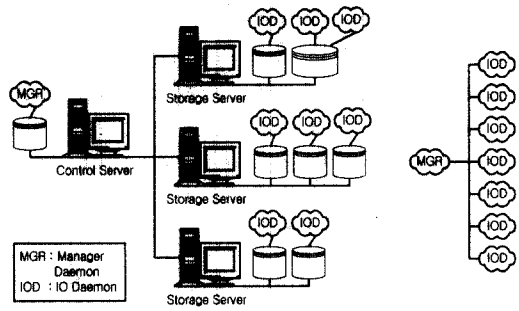
PMFS는 확장성과 이식성을 고려하여 2계층 분산 구조상에서 효과적으로 멀티미디어를 서비스하기 위해 설계되었고 Fast Ethernet 환경에서 구현되었다. 이에 따른 PMFS의 구조와 주요특징을 살펴보면 다음과 같다.

3.1 PMFS의 구조

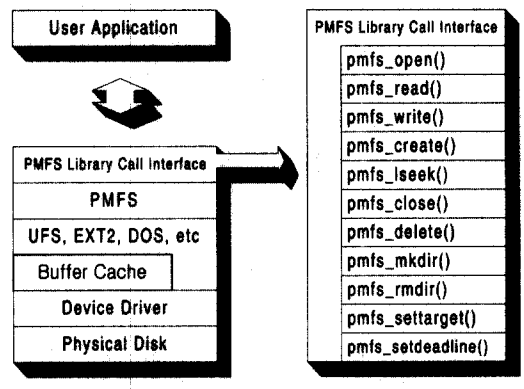
PMFS는 다수의 저장노드에 장착된 디스크들이 하나의 저장그룹(Storage Group)이 되며 하나의 스트라이핑(Striping) 단위가 된다. 하나의 저장 그룹은 하나의 MGR(Manager Daemon)과 다수의 IOD(IO Daemon)로 구성된다. MGR은 미디어 개체의 메타 정보를 관리하는 역할을 하고 IOD는 미디어 개체의 데이터를 저장하고 검색하는 데이터의 입출력을 수행하고 데이터를 입출력하는 논리적인 단위가 된다(그림 3).

3.1.1 PMFS의 계층구조

PMFS는 EXT2와 같은 상위레벨 파일 시스템 상에서 구동 되며 사용자 프로그램을 위한 인터페이스로 11개의 사용자 라이브러리 함수로 구성된다(그림 4).



(그림 3) PMFS의 구조



(그림 4) PMFS의 계층구조

PMFS를 사용하기 위한 인터페이스로는 C로 구현된 CAPI와 JNI(Java Native Interface)로 구현된 Java API가 있다. PMFS에는 Unix 파일 시스템과 유사한 인터페이스를 가지는 9개의 기본함수와 멀티미디어 데이터를 지원하기 위해 미디어 개체의 입출력 마감시간(Deadline)을 설정하는 함수인 pmfs_setdeadline()과 저장서버가 클라이언트로 직접 전송할 수 있도록 저장서버의 데이터 전송방향을 설정하는 함수인 pmfs_settarget ()함수가 있다. 특히 파일을 생성하는 함수인 pmfs_create()은 일반적인 Unix System Call과는 달리 생성시킬 파일에 대한 데이터 배치기법, 미디어 개체의 미디어 형태, 블록크기, 사본정책을 사용자가 설정할 수 있도록 구현되었다

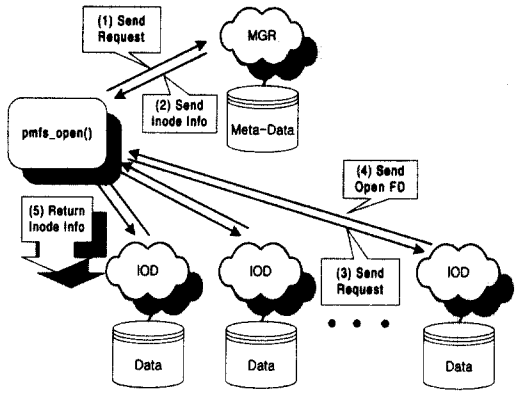
3.1.2 PMFS의 파일 입출력

● 파일 열기

pmfs_open()은 파일을 열기 위한 함수이며, 파일의 I-node 정보를 읽어온다. MGR로부터 I-node정보를 요구하여 전송받고, 각각의 IOD에 파일 열기 요구를 보낸다. 해당 IOD에 응답이 없을 경우, 그 IOD를 실패 된 IOD로 표시하고, 사본이 있을 경우 사본 디스크에서 파일을 연다. 각 IOD는 해당 파일을 열고 FD(File Descriptor)를 보낸다(그림 5).

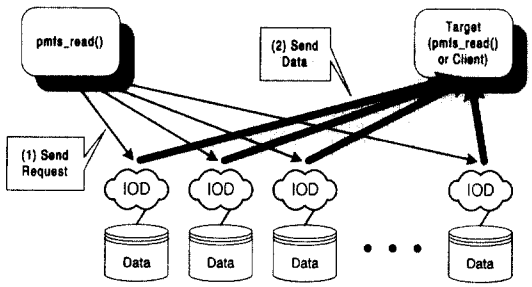
● 파일 읽기

pmfs_read() 는 파일을 읽기 위한 함수이며, 파일을 읽기



(그림 5) pmfs_open()의 동작 구조

위해 pmfs_read()에서 각각의 IOD에 읽기 요구를 보내고 IOD가 목표 주소로 데이터를 전송한다. 이때 목표 주소는 pmfs_settarget()에 의해 설정되며, 설정되지 않았을 경우 pmfs_read()로 데이터가 전송된다(그림 6).



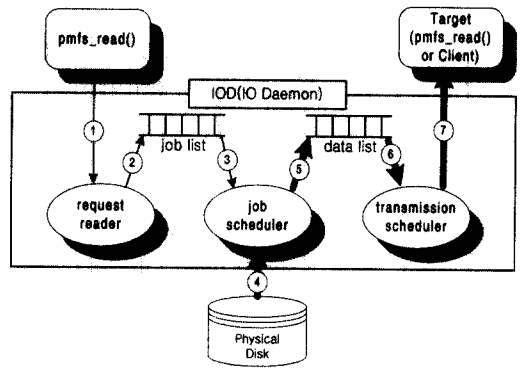
(그림 6) pmfs_read()의 동작 구조

IOD는 Request Reader, Job Scheduler 그리고 Transmission Scheduler로 구성되며, Request Reader는 입출력 요구를 읽어서, 작업 리스트(Job List)에 삽입하는 역할을 하고, Job Scheduler는 작업 리스트에서 해당 작업을 EDF순서로 스케줄링하여 실행하는 역할을 하며, 마감시간(Deadline)이 설정되지 않았을 경우 FIFO 순서로 스케줄링한다. Transmission Scheduler는 데이터 리스트(Data List)에 데이터가 있을 경우 FIFO순서에 의해 데이터를 전송하는 역할을 한다.

실시간으로 데이터를 읽기 위해 Non-Blocking I/O를 하며 정해진 시간 내에 데이터를 읽어 오지 못할 경우 해당 IOD를 실패한 디스크로 표시하고 사본 디스크에서 블록을 읽어온다. 따라서, 데이터 읽기를 수행하는 도중 하나의 디스크가 결함이 발생하여 서비스가 불가능해 졌을 때, 사본 디스크에서 정상적으로 읽어 들이면서 정상적인 서비스를 계속 할 수 있게 된다(그림 7).

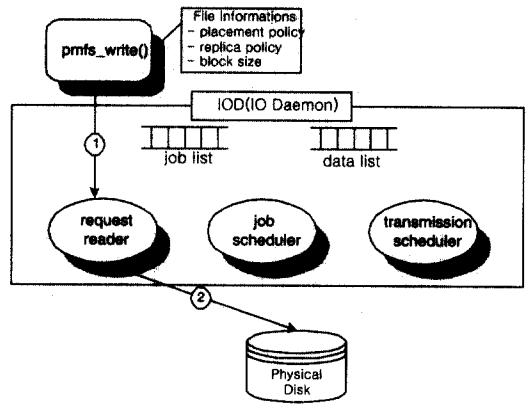
● 파일 쓰기

pmfs_write()는 데이터를 파일에 쓰기 위한 함수이며, pmfs_read() 와 달리 작업 리스트에서 스케줄 되지 않고 곧바로 수행된다. 이것은 pmfs_write()가 파일을 쓰기 위한



(그림 7) pmfs_read()와 IOD의 세부 동작 구조

요구를 전송하고 데이터를 전송할 때 작업리스트에 들어갈 경우 Job Scheduler가 데이터를 읽어야 하는데 동시 사용자수가 2명 이상일 경우 하나의 소켓을 Request Reader와 Job Scheduler가 경쟁하는 상황이 발생한다. 별도의 소켓을 여는 경우 오버헤드가 발생하기 때문에 스케줄하지 않고 Request Reader가 처리하는 구조를 가진다. pmfs_write()는 각 IOD에게 파일 쓰기 요구를 보낼 때 열려있는 파일 포인터를 통해 I-node정보를 가지고 데이터 배치정책과 블록크기, 사본정책 등에 맞추어 적절한 순서로 각 IOD에 파일 쓰기 요구를 보낸다(그림 8).



(그림 8) pmfs_write()와 IOD의 세부 동작구조

3.2 PMFS의 주요특징

PMFS는 확장성과 이식성을 고려하여 파일시스템 수준에서 멀티미디어를 효과적으로 서비스하기 위해 구현된 파일 시스템으로서 다음과 같은 특징을 지닌다.

● 이식성

상위레벨 파일시스템 뷸 파일 시스템으로 Linux, FreeBSD, Solaris 등 다양한 Unix계열 운영체제에서 구동될 수 있으며, MPI-IO에서 쓰이는 ADIO(Abstract Device I/O)[7]와 같이 입출력에 관련된 인터페이스를 다양한 입출력 시스템에 이식하기 쉽도록 표준화 하였다.

• RTP에 기반한 분산 전송 지원

TCP에 의해 제어 메시지가 전송되며, 데이터는 UDP에 의해 전송되고, 전송된 데이터 패킷을 조합할 수 있는 RTP클라이언트에 대해 저장서버에서 읽은 데이터를 제어서버를 거치지 않고 직접 클라이언트에 전송할 수 있어 내부 통신과 메모리 복사를 줄인다.

• 텍스트, 이미지, 오디오, 비디오 등 비 실시간 및 실시간 멀티미디어 데이터 지원

미디어 형을 지정할 수 있고 각각의 미디어 형에 따라 마감시간을 설정할 수 있으며 IOD는 I/O Request에 대해 EDF스케줄링 정책으로 입출력 요구를 스케줄한다.

• 2계층 분산 클러스터 서버 구조에 적합한 파일 시스템

PMFS는 제어 서버에서 구동 되는 MGR과 저장 서버에서 구동 되는 다수의 IOD로 구성된다.

• 멀티미디어 데이터를 지원하기 위한 데이터 배치기법

PMFS는 데이터 배치기법으로 Round Robin, Staggered Striping 그리고 DIS기법을 지원한다. 그리고, 각각의 시작 블록을 임의적으로 배치하는 배치기법을 사용한다.

• 유연성

PMFS는 멀티미디어 응용 프로그램 개발자가 미디어 개체를 시스템 환경에 따라 유연하게 저장할 수 있도록 한다. PMFS는 미디어 개체를 저장할 때, 데이터 배치기법, 미디어 형태, 블록크기, 사본정책 등을 지정할 수 있으며, 논리적 디스크의 개수와 Striping크기 등을 설정할 수 있다.

• 이기종 디스크의 지원과 결함허용

PMFS는 디스크 저장공간 비나 대역폭 비에 의해 이기종 물리적 디스크를 동일 기종의 논리적 디스크로 구성할 수 있고, 필요에 때라 사본 디스크에 데이터의 사본을 둘 수 있어서 서비스 도중에 디스크에 결함이 발생하더라도 연속적으로 서비스가 가능하다.

4. 성능평가 및 분석

Fast Ethernet상에서 구현된 PMFS의 데이터 배치기법 및 데이터 전송구조의 타당성을 검증하기 위하여 사용자수에 따른 마감시간 실패율을 측정하였고, PMFS의 멀티미디어 데이터 처리의 적합성을 검증하기 위하여 동일하게 Fast Ethernet상에서 구현된 PVFS(Parallel Virtual File System)와 성능을 비교하였다. 실험값은 20회 반복실험의 평균값이며 유의수준 0.01이내에서 χ^2검정에 의해 검증하였다.

4.1 실험 환경

1개의 제어서버와 3개의 저장서버로 구성된 2계층 분산 클러스터 구조의 서버를 구성하였다. 4개의 서버는 Fast Ethernet과 내부 Switching Hub에 의해 연결되고 사용자 입력은 외부서버(Sun OS, Ultra Sparc 30)에서 가상으로 발생시켰다. 8개의 디스크에 Text, Image, Audio(MP3), Video(MPEG1, MPEG2)데이터를 임의적으로 선택하여 600개의 데이터로 파일 쓰기 및 읽기를 실험하였고 사용자의 실시간 데이터와 비실시간 데이터의 요구 빈도를 변화시키면서 실험하였다. 실험을 위한 H/W, S/W환경 그리고 실험을 위해 배치된 미디어 데이터는 다음과 같다.

〈표 1〉 실험에 사용된 H/W 환경

CPU	Intel Pentium II 300 MHZ Dual(1), Pentium 100 MHZ(2)
Main Memory	256 Mbyte(1), 32 Mbyte(2)
SCSI Card	Adaptec AIC-7860 Ultra SCSI
Network Interface Card	Intel EtherExpress Pro 100
Switching Hub	OmniSTACK Hub
HDD	Segate ST34520W(5), ST34371W(2),ST39140W(1)

〈표 2〉 실험에 사용된 S/W 환경

운 영 체 제	Linux Kernel 2.2.9
배치된 미디어	Text, Image, Audio(MP3), Video(MPEG1)
사용자 입력 패턴	사용자 요구 도착 간격이 Exponential 분포

〈표 3〉 배치된 미디어 데이터

미디어 형(Media Type)	미디어 데이터
Text(HTML)	1999년 조선일보 News Data
Image(JPG,GIF)	1999년 조선일보 News Data
Audio(MP3)	1999년 대중가요 Data
Video(MPEG1)	1999.5.27, 6.1, 6.2 MBC, KBS 9시 News Data

4.2 성능 평가

4.2.1 데이터 배치기법에 따른 성능평가

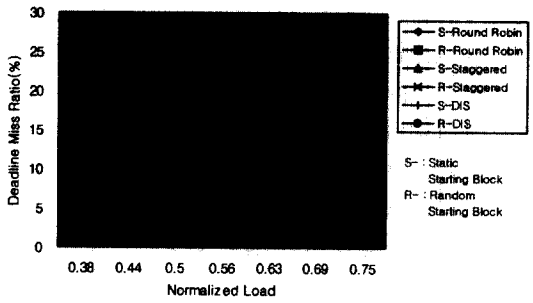
데이터 저장에 있어 시작 블록기법, 배치기법과 데이터 전송 기법에 따른 성능을 비교 분석하기 위해 실시간 데이터와 비실시간 데이터의 요구 빈도수가 동일하다고 가정하였다.

1) 시작 블록기법에 따른 성능 비교

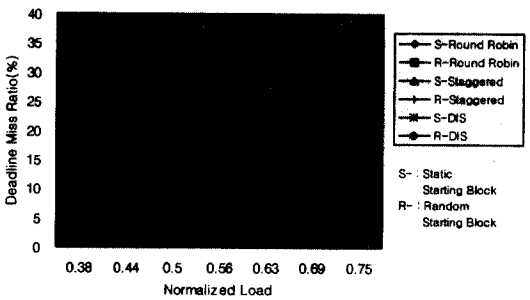
실험결과 실험환경에서는 임의 시작블록 배치기법이 고정 시작블록 배치기법에 비해 평균 7~14% 향상된 성능을 보였다(그림 9), (그림 10). 이것은 고정 시작블록 배치기법의 경우 첫번째 디스크에 배치되는 미디어 개체의 수가 많아 첫번째 디스크에 대한 접근 빈도가 높아지고 이로 인한 부하 불균형이 발생되기 때문이다. 반면 모든 디스크에 균등하게 시작블록이 배치되는 임의 시작블록 배치기법이 부하가 높을수록 향상된 성능을 보였다.

2) 배치기법에 따른 성능 비교

실험결과 실험환경에서는 DIS배치기법이 Round Robin에



(그림 9) Distributed Message 기반 구조에서 데이터 배치기법별 마감시간 실패율



(그림10) Centralized Shared Data 구조에서 데이터 비치기법별 마감시간 실패율

비해 3~16%, Staggered Striping에 비해 2~11% 향상된 성능을 보였다. 이것은 DIS배치기법으로 데이터를 배치하였을 경우 디스크에 대한 입출력 요구 충돌 시 연속적인 요구 충돌을 피할 수 있기 때문이다(그림 9), (그림 10).

3) 데이터 전송구조에 따른 성능 비교

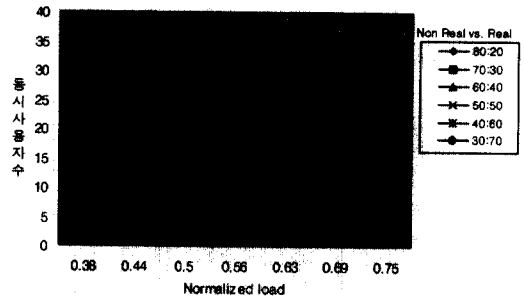
데이터 배치기법 중 향상된 성능을 보이고 있는 임의 시작블록 배치에 의한 DIS배치기법에서 내부 데이터 전송구조별로 성능을 평가한 결과 실험환경에서 Distributed Message 기반 전송구조에 의해 서비스를 할 시 Centralized Shared Data전송구조에 의한 것보다 21~30% 향상된 성능을 보였다. 이것은 Centralized Shared Data전송구조에서는 제어서버가 모든 저장서버의 데이터를 읽어와서 재조합한 후 다시 클라이언트에게 전송해야 하므로 시스템의 부하가 높을수록 제어서버에 부하가 가중되기 때문이다(그림 9), (그림 10).

4.2.2 사용자 요구 데이터의 유형에 따른 성능 평가

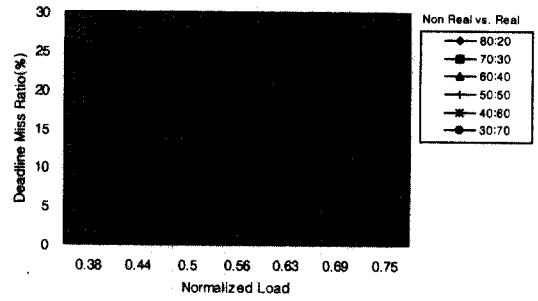
(그림 11)은 임의시작 블록 DIS 배치기법 알고리즘 상에서 사용자의 비실시간 데이터의 요청과 실시간 데이터의 요청 빈도에 따른 동시 사용자수를 측정한 결과이다. 비실시간 데이터의 요청 빈도가 높을수록 동시 사용자 수가 증가한다. 그러나 (그림 12)에서 보듯이 마감시간 실패율은 실시간 데이터의 요청 빈도가 높을수록 증가함을 알 수 있다.

4.2.3 PVFS와 성능 비교 및 분석

PMFS와 유사한 시스템 환경에서 구동하는 병렬 파일



(그림 11) 사용자 요구 데이터의 유형에 따른 동시사용자 수



(그림 12) 사용자 요구 데이터의 유형에 따른 마감시간 실패율

시스템으로는 PVFS[표 4]가 있다. PMFS의 성능을 평가하기 위하여 PMFS와 PVFS에 의해 구성된 서버에서 사용자 수의 증가에 따른 마감시간 실패율을 비교하였다.

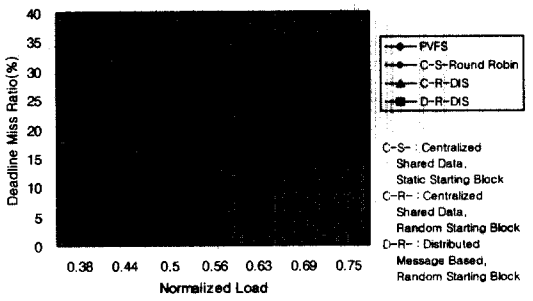
〈표 4〉 PVFS와 PMFS의 비교

파일 시스템	PVFS	PMFS
서버 구조	2계층 분산 클러스터 구조	2계층 분산 클러스터 구조
운영체제 환경	Linux 2.x	Linux 2.x , Solaris, FreeBSD
전송 프로토콜	TCP/IP	TCP/IP, UDP/IP
내부 데이터 전송	Centralized Shared Data	Centralized Shared Data, Distributed Message 기반
데이터 배치기법	고정 시작블록 Round Robin	임의 시작블록 Round Robin, Staggered Striping, DIS
스케줄링 기법	FIFO	EDF
구현 목적	General Purpose	멀티미디어 서버 개발

가장 좋은 성능을 보이는 D-R-DIS배치(Distributed Message 기반 데이터 구조상에서 임의 시작블록에 의한 DIS 데이터 배치)와 가장 성능이 좋지 않은 C-S-RR배치(Centralized Shared Data 구조상에서 고정 시작블록에 의한 Round Robin데이터 배치)의 PMFS와 PVFS를 비교하였다.

실험결과 PVFS와 C-S-RR배치에서의 PMFS는 비슷한 성능을 보였고, D-R-DIS환경에서의 PMFS는 부하가 높아질수록 더 향상된 성능을 보이고 있다. 이것은 PMFS가 입출력 작업을 마감시간을 기준으로 스케줄하고, Distributed Message 기반 구조에서는 제어서버의 부하를 줄일 수 있

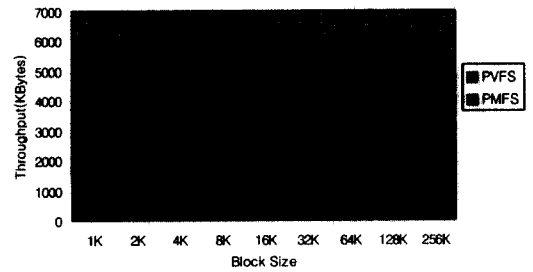
으며, DIS배치기법은 연속적인 디스크 요구충돌을 줄일 수 있기 때문이다(그림 13).



(그림 13) PMFS와 PVFS의 사용자수에 따른 마감시간 실패율

4.3 파일 쓰기

PMFS와 PVFS의 블록크기별 파일 쓰기 시 처리율(Throughput)을 비교한 결과 PMFS가 평균 5% 향상된 성능을 보이며 블록크기가 클수록 성능이 향상되었으며, 블록 크기가 128K일 때 가장 좋은 성능을 보였다(그림 14). 이는 PMFS 시스템이 대용량 실시간 데이터만을 처리하는 것이 아니라 저용량인 텍스트 및 이미지 데이터를 같이 처리하기 때문에 지나치게 큰 블록은 시스템의 성능을 저하시킨다.



(그림 14) PMFS와 PVFS의 블록별 파일 쓰기 처리율 비교

5. 결론 및 향후 연구방향

본 논문에서는 현재 많은 연구와 개발이 이루어지고 있는 인터넷을 기반으로 한 다양한 멀티미디어 응용 소프트웨어와 NOD그리고 Digital Library와 같은 멀티미디어 응용의 개발을 위한 병렬 멀티미디어 파일 시스템을 Fast Ethernet에서 구현하였고, 파일 시스템의 구조와 특성 그리고 구현상의 문제점을 설명하고 성능을 평가하였다.

본 논문에서 제시한 파일 시스템인 PMFS는 하나의 제어노드와 다수의 저장노드가 하나의 저장그룹을 이루는 2계층 분산구조로 구성된 클러스터 기반 파일 시스템이며, 사용자 함수와 제어노드 및 저장노드 간의 제어 메시지를 통해 입출력을 수행하는 구조로 구성된 메시지 전송 기반 파일 시스템이다.

PMFS는 제어노드의 부하를 줄이고 내부 통신 오버헤드를 줄이기 위해 저장노드에서 클라이언트로 직접 데이터를 전송할 수 있도록 하였고, 실시간 및 비실시간 멀티미디어 데이터를 지원하기 위해 Round Robin, Staggered Striping 그리고 DIS배치기법과 각 배치기법에서 시작블록을 임의적으로 배치하는 데이터 배치기법을 지원하고, EDF에 의해 작업을 스케줄한다. 그리고, 상위레벨 파일 시스템상에서 구동 됨에 따라 커널의 수정 없이 다양한 운영체제와 입출력 시스템에 설치 및 이식이 용이하고, 사용자 수준에서 블록크기, 데이터 배치기법, 복사본 정책, 마감 시간 등을 유연하게 설정할 수 있는 특성을 지닌다.

PMFS의 성능을 평가한 결과 실험환경에서 임의 시작블록 데이터 배치에 의한 DIS배치기법에 의해 데이터를 배치하고 Distributed Message 기반 데이터 전송 구조에 의해 데이터를 전송할 경우 가장 좋은 성능을 보였으며 PMFS와 유사한 구조를 가지는 PVFS와 비교결과 부하가 높을수록 향상된 성능을 보였다.

다양한 형태의 실시간 스케줄링 정책에 대한 연구와 결함 디스크의 인식 시 응답 대기시간에 대한 연구는 아직 부족하며 이는 향후 연구해야 할 과제이다.

참 고 문 헌

[1] Asit Dan, Dinkar Sitaram, "An Online Video Placement Policy based on Bandwidth to Space Ratio (BSR)," Proceedings of ACM SIGMOD International Conference on Management of Data, 1995.

[2] Rajeev Thakur, William Gropp, Ewing Lusk, "Data sieving and Collective I/O in ROMIO," 7th Symposium on the Frontiers of Massive Parallel Computation, 1999.

[3] C. Bernhardt and E. W. Biersack, "The Server Array : A Scalable Video Server Architecture," in High Speed Networking for Multimedia Applications, 1996.

[4] Emmanuel Cecchet, "SciFS Technical & Practical Guide," http : //sci-serv.inrialpes.fr/SciFS/scifs_doc. html, 1999.

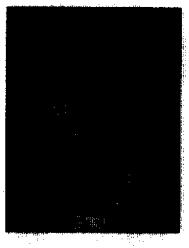
[5] Frank Fabbrocino, Jose Renato Santos, Richard Muntz, "An Implicitly Scalable, Fully Interactive Multimedia Storage Server," 2nd International Workshop on Distributed Interactive Simulation and Real Time Applications, 1998.

[6] Heinz Stockinger, "Dictionary on Parallel Input/ Output," Master's Thesis, University of Vienna, Austria, 1998.

[7] Rajeev Thakur, William Gropp, Ewing Lusk, "An Abstract Device Interface for Implementing Portable Parallel I/O Interfaces," 6th Symposium on the Frontiers of Massively Parallel Computation, 1996.

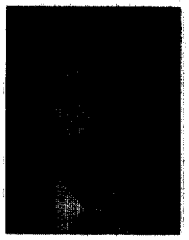
[8] Jamel Gafsi, Ernst W. Biersack, "Data Striping and Reliability Aspects in Distributed Video Servers," In Cluster Computing : Networks, Software Tools, and Applications, 1998.

[9] John Gustafson, "Pentium Pro Cluster Workshop," http : // www.scl.ameslab.gov/-workshops/PPCworkshop.html, 1997.
[10] Robert Flynn, William Tetzlaff, "Disk Striping and Block Replication Algorithms For Video File Servers," IEEE Conference on Multimedia Computing and Systems, 1996.
[11] Nils Nieuwejarr, "Galley : A New Parallel File System For Scientific Workloads," Ph.D Dissertation, Dartmouth College, Computer Science Department, 1996.
[12] P. J. Shenoy, P. Goyal, S. Rao, H. M. Vin, "Design and Implementation of Symphony An Integrated Multimedia File System," In Proceedings of ACM/ SPIE Multimedia Computing and Networking 1998, 1998.
[13] R. Friedman, D. Mosse, "Load Balancing Schemes for High Throughput Distributed Fault-Tolerant Servers," Symposium on Reliable Distributed Systems, 1997.
[14] Richard Muntz, Jose Renato Santos, Frank Fabbrocino, "Design of a Fault Tolerant Real-Time Storage System for Multimedia Applications," International Computer Performance and Dependability Symposium, 1998.
[15] Richard Muntz, Xiangyu Ju, "Staggered Striping : A Flexible Technique to Display Continuous Media," Proceedings of ACM Multimedia, 1994.
[16] Henning Schulzrinne, "RTP : About RTP and the Audio-Video Transport Working Group," http : // www.cs.columbia.edu/~hgs/rtp/, 1997.
[17] Roger Haskin, Frank Schmuck, "The Tiger Shark File System," Proceedings of IEEE COMPCON, 1996.
[18] Roger Zimmermann, "Continuous Display Using Heterogeneous Disk Subsystems," Ph.D Dissertation, University of Southern California, Computer Science Department, 1998.
[19] Roger Zimmermann, Shahram Ghandeharizadeh, "Continuous Display Using Heterogeneous Disk- Subsystems," Proceedings of ACM Multimedia, 1997.
[20] W. Bolosky et al., "The Tiger Video Fileserver," in 6th Workshop on Network and Operating System Support for Digital Audio and Video, 1996.
[21] W. B. Ligon III and R. B. Ross, "An Overview of the Parallel Virtual File System," Proceedings of the 1999 Extreme Linux Workshop, 1999.
[22] 김영주, "확장성을 가진 연속 미디어 저장 서버의 설계 및 성능 분석", 부산대학교 일반대학원 이학박사 학위논문, 1999.
[23] 정귀옥, "Clustered NOD 저장 서버에서 데이터 Striping 및 Replica 배치기법", 부산대학교 일반대학원 이학석사 학위논문, 1998.
[24] Gwang-Moon Kim, Seung-Ho Park, Young-Ju Kim, Ki-Dong Chung, "Mixed-Media Data Placement Policy in Heterogeneous Disk Arrays," Systemics, Cybernetics and Informatics and International Systems Analysis and Synthesis (SCI/ISAS 99), 1999.
[25] 강연경, "Clustered MNOD(Multimedia News On Demand)서버 환경에서 QoS에 기반한 실시간 스케줄러", 부산대학교 일반대학원 이학석사 학위논문, 1998.



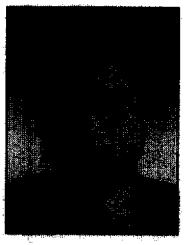
박 성 호

e-mail : shpark@melon.cs.pusan.ac.kr
1996년 부산대학교 전자계산학과 졸업(학사)
1998년 부산대학교 전자계산학과 대학원 졸업(석사)
2000년 부산대학교 전자계산학과 대학원 수료(박사)
관심분야 : VOD 시스템, 멀티미디어 통신, 인터넷 캐싱



김 광 문

e-mail : gmkim@rex.lgic.co.kr
1992년~1998년 부산대학교 전자계산학과 (이학사)
1998년~2000년 부산대학교 전자계산학과 대학원(이학 석사)
2000년~현재 LG전자 핵심망연구소 운영체계실(연구원)
관심분야 : 대용량 멀티미디어 저장서버, 멀티미디어 운영체제, 내장형 운영체제



정 기 동

e-mail : kdchung@melon.cs.pusan.ac.kr
1973년 서울대학교 졸업(학사)
1975년 서울대학교 대학원 졸업(석사)
1986년 서울대학교 대학원 계산통계학과 졸업(이학박사)
1990년~1991년 MIT, South Carolina 대학 교환 교수
1995년~1997년 부산대학교 전자계산소 소장
1978년~현재 부산대학교 전자계산학과 교수
관심분야 : 병렬처리, 멀티미디어