□기술해설□

대용량 데이타 관리를 위한 병렬 입출력시스템

부산대학교 노영욱* · 정기동**

목 차

1. 서 론

1985년 이후로 10년간 프로세서 속도는 매년 50%~100% 증가하였으나 이 기간 동안 디스크의 전송 속도는 약 2배 향상되었다[1]. 그리고 반도체 기술의 발달로 마이크로 프로세서와 메모리 속도가 향상되고 적은 비용으로 높은 계산력을 얻기 위해 프로세서를 여러 개 사용한 다중 프로세서와 MPP(Massively Parallel Processing) 시스템에 대한 연구가 많이 이루어졌다. 프로세서의 속도 향상과 프로세서 수의 증가로 계산력은 많이 개선된 반면에 I/O의 성능은 그렇지 못하여 I/O 성능이 전체 시스템 성능의 병목으로 작용하게 되었다.

대규모 과학계산, 데이타베이스와 정보 처리, 하이퍼텍스트와 멀티미디어, GIS(Geographic Information System), VOD(Video On Demand)와 같은 응용에서는 대용량의 데이타 처리를 요구한다. 과학 기술 분야의 Grand Challenge 응용에서는 응용 프로그램을 한번 수행하는데 1 GByte에서 4 TeraByte의 데이타를 필요로 한다[2]. VOD 응용 분야에서 1시간 분량의 비디오 데이타를 압축하면 약 1 GByte의 저장 공간을 필요로 한다. VOD 시스템은 수천 편의 비디오를 저장하고 수 천명의 사용자 요구를 동시에 처리하여야 하므로 대용량의 저장 능력과 전송속도를 제공하여야 한다[3]. GIS 분야에서는 수 TeraByte의 데이타를 관리할 수 있는 병렬 GIS 시스템이 필요하다[4].

대용량의 데이타를 처리하는 응용 분야의 요구를 만족하고 I/O 병목현상을 해결하기 위한 방법 중의 하나로 입출력시스템을 병렬화하는 방법이 제시되었다. 병렬 I/O를 제공하는 방법으로는 다중 디스크, 다중 제어기, 다중 채널을 사용하여 하드웨어적으로 병렬성을 얻는 방법과 이러한 하드웨어를 바탕으로 병렬 화일시스

*정 회 원
**종신회원

템으로 구현하는 방법이 있다. 본 논문에서는 대용량의 데이타를 저장관리하기 위한 병렬 입출력시스템에 대하여 전반적으로 살펴보고 본 연구팀이 연구 중인 멀티미디어 데이타 저장 서버에 대하여 기술한다.

본 논문의 구성은 다음과 같다. 2장에서는 병렬화일의 접근 형태(access pattern)와 작업부하(workload)의 특성, 병렬 입출력 프로그래밍 지원 방법에 대해 기술한다. 3장에서는 병렬 입출력시스템 설계시 고려 사항을 소프트웨어적인 기술을 중심으로 기술한다. 4장에서는 고속의 네트워크로 연결된 분산 환경에서 병렬 입출력시스템 기술을 이용한 멀티미디어 데이타 저장시스템을 제안한다. 5장에서는 관련 연구에 대해서 살펴보고, 6장에서 결론을 기술한다.

2. 접근 형태와 작업부하 특성

2.1 접근 형태

I/O 시스템에 관한 연구를 하기 위해서는 응용 분야의 I/O 특성을 이해하여야 한다. 효과적으로 병렬 입출력시스템을 사용하기 위해서 응용 프로그램으로부터 화일시스템이나 데이타 관리시스템으로 접근 형태에 대한 정보를 전달하여야 한다. 본 장에서는 I/O의 접근 형태와 작업부하를 파악하기 위해 디스크와 화일시스템 수준에서의 접근 형태와 여러 응용 분야의 작업 부하 특성에 관한 기존 연구를 살펴본다.

응용 프로그램의 I/O 접근 형태는 매우 다양하며 I/O 성능은 접근 형태에 많이 의존한다. I/O 접근 형태에 따라 시스템이 I/O를 관리하는 방법, 저장 장치에 데이타를 분산하는 방법과 사용되는 캐싱 알고리즘이 달라진다.

대부분의 접근 형태에 대한 연구는 운영체제의 화일시스템 수준에서 이루어졌다. 하위 수준인 디스크에 대한 접근 형태의 정보는 많이 알려져 있지 않으나 디스크의 접근 형태는 화일시스템 수준에서의 접근 형태와는 다르다고 알려져 있다[5].

화일시스템 수준에서는 단일 UNIX 워크스테이션[6]과 분산환경[7]에서 화일의 접근 형태에 대한 연구가 있었다. 응용 프로그램에서 화일시스템을 접근하는 형태는 일반적으로 과학 계산, 트랜잭션 처리, 공학과 사무용으로 나눌 수 있다[8]. 과학 계산 분야에서는 큰 화일을 순차적으로 접근하므로 높은 전송률이 매우 중요하다. 트랜잭션 처리 분야는 데이타베이스 시스템처럼 많은 수의 짧은 요구를 포함하므로 전체 I/O 처리율이 중요하다. 공학과 사무 분야는 많은 수의 짧은 요구를 가지나 트랜잭션 처리에 비해 병행성이 적은 특성이 있으므로 개별 디스크의 응답시간이 매우 중요하다. VOD와 같은 멀티미디어 응용 분야에서는 큰 화일을 많은 사용자가 동시에 요구하고 멀티미디어 데이타를 실시간으로 서비스하기 원하므로 I/O의 지연(latency)과 처리율이 중요하다.

[9, 10]에서는 실제 응용프로그램을 분석하지 않고 병렬 화일시스템에서 병렬화일에 대한 접근 형태를 분류하였다. 그림 1은 [9]에서 분류한 방법에 따라 3개의 프로세스로 구성된 가상적인 병렬 프로그램의 순차적인 접근 형태를 나타낸 것이다. [11]에서는 슈퍼컴퓨터에서의 병렬 입출력 접근을 compulsory, checkpoint, out-of-core로 분류하였다.

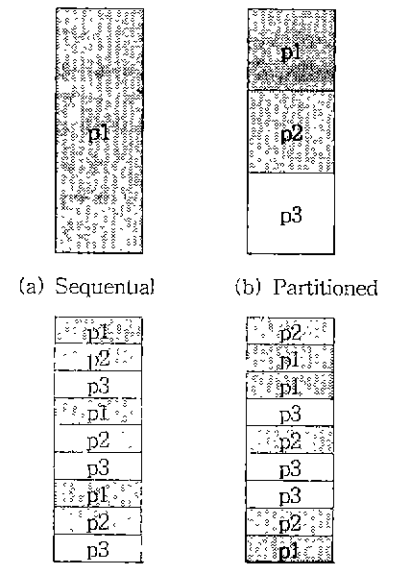


그림 1 화일의 접근 형태 분류

그리고 iPSC/860[12]와 CM-5[13]에서 일

장기간 동안(20일 이상) 수행된 각종 응용 프로그램이 CFS(Concurrent File System)[12]와 SDA(Scalable Disk Array)[13]을 사용한 데이타를 분석한 연구가 있다. 이 연구에 따르면 과학 계산용 병렬화일시스템을 설계할 때 고려할 사항은 다음과 같다. 첫째, 수행 시간이 짧은 작업과 수행 시간이 긴 작업 모두를 효율적으로 접근 가능하여야 하고 변화하는 시스템의 부하 조건에 적절히 반응하여야 한다. 둘째, 같은 작업에서 동시에 open한 여러 화일을 최적으로 접근하여야 한다. 셋째, 작은 I/O에 대해서는 낮은 지연 시간을 보장하고 큰 I/O에 대해서는 높은 대역폭을 제공하여야 한다. 넷째, 쓰기 공유의 수는 적고 읽기 공유의 수는 많으므로 I/O 노드 뿐만 아니라 계산 노드에서도 데이타를 캐싱하여야 한다. 다섯째, 재쓰기(rewrite)의 시간은 짧으므로 버퍼에 쓰기를 하고 디스크로의 쓰기는 지연하는 것이 좋다. 여섯째, 고 수준에서 I/O 속도가 향상될 때까지 병렬 화일시스템에서 "raw" I/O를 제공하여야 한다. 일곱째, 기존의 UNIX I/O를 확장하여 사용하는 것은 효과적이지 못하다.

특정 시스템에서 작업 부하를 분석할 때 표 1과 같은 데이타를 파악하여 병렬 저장 시스템과 병렬 화일시스템 연구에 사용하여야 한다[12, 13].

표 1 작업부하 분석 시에 고려하여야 할 특성

작업	동시에 수행되고 있는 작업 수 각 작업이 사용하는 프로세서 수 작업의 수행 시간 각 작업당 open하는 화일 수 각 작업당 수행하는 응용 프로그램 수
화일	읽기만 하고 쓰기만 하는 화일 수 화일 크기 화일이 open되는 기간 화일을 공유하는 노드 수
I/O 요구	I/O 요구 크기 I/O 요구가 분산되는 형태 화일당 다른 요구 크기를 갖는 수 접근의 순차성 정도
정책	캐싱과 선반입의 유용성

2.2 병렬 입출력 프로그래밍 지원 방법

화일 접근 형태에 대한 정보를 프로그래밍 언어, 컴파일러, 수행시간 수준에서 제공하여 병렬 입출력을 프로그래밍 할 수 있다. 그림 2는 프로그램으로부터 추출된 접근 형태의 정보를 이용하여 효과적인 캐쉬와 버퍼의 할당 및 관리, 디스크로의 I/O 요구 스케줄링, 선반입을 최적화하기 위한 정보 흐름을 나타낸다[14].

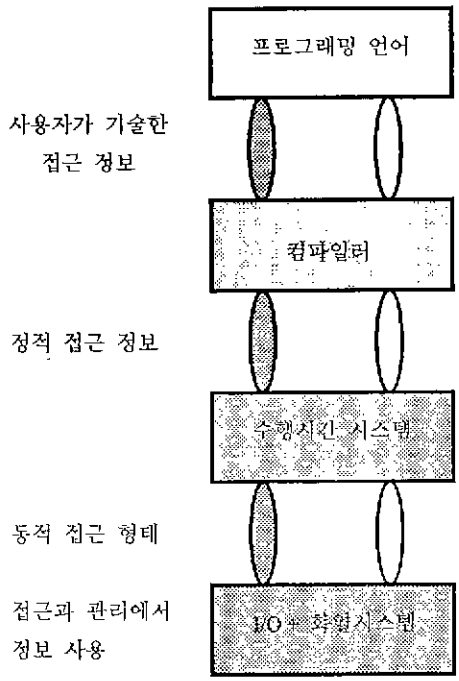


그림 2 접근 형태에 대한 정보 흐름

2.1.1 언어에서 지원

사용자가 프로그램 내에서 자료구조와 병렬성, 화일접근과 데이타 분산, 데이타 이동을 제어하는 접근 방법을 명시적으로 기술할 수 있도록 병렬 프로그램언어를 확장하거나 directive를 제공한다.

2.2.2 컴파일러 지원

MPP나 슈퍼 컴퓨터에서 시스템의 메모리 크기보다 큰 배열(out-of-core array)과 같은 자료구조를 많이 처리한다. 프로그래머는 메모리 크기보다 큰 배열을 관리하여야 한다. Fortran D나 HPF(High Performance Fortran)과 같은 언어는 사용자가 일반 배열을 사용하는 것과 같은 방법으로 out-of-core 배열을 선

언하여 프로세서 사이에 배열의 분산을 기술할 수 있다. 그러면 Fortran D/HPF 컴파일러는 데이타가 요구될 때 프로세서에 데이타를 보내고 프로세서에서 데이타를 수신하는 코드를 생성한다.

병렬 I/O를 하기 위해 병렬 컴파일러는 다음 사항들을 최적화하여야 한다. 첫째, 병렬 I/O 연산을 인식하고 여러 가지형의 화일과 데이타 형식을 병렬화한다. 둘째, I/O 연산과 계산을 overlap하여야 한다. 셋째, I/O 시스템과 계산 노드 사이에 효과적인 데이타 mapping을 하여야 한다.

2.2.3 수행 시간 지원

I/O 연산을 최적화하기 위해 컴파일러는 접근 형태에 대한 정보를 수행시간 시스템에 제공한다. 수행 시간 시스템은 컴파일러로부터 받은 접근 정보를 사용하여 적당한 I/O 스케쥴과 접근을 생성하고 컴파일러가 생성한 접근 정보를 병렬 저장시스템에 전달한다. 수행 시간에 대한 연구는 다음 사항을 포함한다. 첫째, 지연을 줄이기 위해 수행시간에 데이타 접근 형태의 특성을 사용하여 I/O 요구를 재구성한다[15]. 둘째, I/O를 재순서화하여 캐쉬에 저장될 수 있는 디스크 데이타 양을 증가시켜 I/O 지연을 줄인다. 그리고 디스크에 저장되어 있는 데이타의 위치 정보를 이용하여 최적의 I/O 성능을 얻기 위해 I/O 요구를 재배열한다. 셋째, 효과적인 collective I/O를 지원하는 수행시간 primitive를 제공한다[16].

3. 병렬 입출력시스템 설계시 고려 사항

병렬 입출력시스템을 구현할 때 입출력시스템의 성능에 영향을 미치는 요인을 고려하여야 한다. 본 장에서는 입출력시스템의 성능에 영향을 미치는 요인으로 다중 디스크, 데이타 분산 프로토콜, 디스크와 프로세서의 연결 구조, 병렬 화일 접근 전략, 디스크 선반입과 캐싱, 화일 시스템 인터페이스 등에 대해 살펴본다.

3.1 다중 디스크

고성능 병렬 입출력시스템에서는 일반적으로 가격이 저렴한 여러 대의 디스크를 사용하여 전체 시스템의 처리율을 향상하는 RAID (Redundant Array of Inexpensive Disk)[17]을 많이 사용한다. 데이타를 다중 디스크에 분할 저장하는 방법으로는 독립적인 방법, 디스크를 동기적 배열로 구성하는 방법과 비동기적인 배열로 구성하는 방법이 있다.

독립적인 방법은 화일 전체를 하나의 디스크에 저장하는 방법으로 많은 수의 독립적인 요구가 있는 응용 분야에서 적합하나 화일이 균등하게 분산되지 않을 경우에 부하를 균등하게 유지할 수 없는 단점이 있다.

동기적 디스크 배열 방법은 연속된 데이타 바이트(블록)를 각 디스크의 같은 위치에 저장하도록 모든 디스크의 회전을 동기화시키는 방법이다.

비동기적 디스크 배열 방법은 연속한 데이타 블록을 인접한 디스크에 독립적으로 저장한다. 데이타를 전송하는데 소요되는 디스크 탐색시간과 회전 지연 시간은 각 디스크에 다르다. 따라서 m 개의 디스크로 구성된 비동기 배열에서 데이타

블록에 대한 요구 지연 시간은 m개의 접근 중에 가장 느린 것이 된다. 비동기 배열 방법은 동기적 배열 방법 보다 확장성은 좋다.

RAID는 다중 디스크의 사용으로 인하여 디스크의 평균 고장 시간은 감소되므로 데이타를 중복하여 저장하거나 패리티를 이용하여 디스크에 고장이 발생한 경우에도 데이타를 신속히 회복할 수 있어야 한다.

3.2 데이타 분산 방법

여러 디스크에 분산된 데이타를 접근하는 시간은 디스크 시스템에서 사용하는 데이타 분산 방법과 기본 블록의 크기에 의존한다. 데이타 분할 방법은 난수 발생기를 사용하여 블록을 임의로 분산하는 방법, modulo 함수를 사용하여 round-robin 방식으로 데이타를 분할하는 방법과 관련된 데이타 집합을 minimum spanning 트리를 형성하도록 분할하는 방법 등이 있다.

데이타의 집합이 분산되는 기본 블록의 크기(스트라이핑 인수)가 커지면 데이타가 분할되

표 2 기술 진보에 따른 LAN과 채널 대역폭 비교[20]

시 기	LAN	채 널
1970s~mid 1980s	1-2 Mbyte/sec(CSMA/CD)	> 4 M byte/sec(IBM 370)
Early 1990s	10 Mbyte/sec(FDDI)	17 Mbyte/sec(ESCON I/O)
Mid 1990s~late 1990s	100 Mbyte(ATM)	100 Mbyte/sec(Fibre Channel)

는 개수(병행성)가 작아진다. 스트라이핑 인수와 병행성의 정도는 I/O 작업 부하의 성능에 많은 영향을 미친다[18].

3.3 프로세서와 디스크의 연결 구조

디스크를 제어하는 방법으로는 CM-2와 같이 모든 디스크를 하나의 제어기가 처리하는 방법과 Intel i860, Intel Touchstone, nCUBE-2와 같이 여러 개의 제어기가 존재하여 몇 개의 디스크를 하나의 제어기가 처리하는 방법이 있다.

프로세서와 디스크를 연결하는 방법으로는 CM-5와 같이 I/O 전용망을 구성하는 방법과 RAMA[19]와 같이 상호 연결망(interconnection network)을 사용하여 I/O와 프로세서 간의 통신을 겸용하는 방법이 있다. I/O 전용망을 사용하면 균등하고 높은 대역폭의 I/O가 가능하나 디스크 제어와 프로세서와 디스크 상호 연결을 보다 복잡하게 하는 단점이 있다.

최근의 통신 네트워크 기술의 발달로 LAN (Local Area Network)을 I/O 채널로 사용하는 경향이 점점 많아지고 있다[20]. 표 2는 기술 발달에 따른 LAN과 채널 대역폭을 비교한 것이다.

3.4 캐싱과 선반입 기법

캐싱(caching)은 자주 접근되는 데이타를 주기억 장치에 저장하여 I/O 연산의 지연을 줄이는 방법이다. 선반입(prefetching)은 읽기 지연을 줄이기 위해 다음에 사용될 블록을 미리 캐쉬로 읽는 방법이다. 성능을 최대한 발휘하기 위해서 선반입은 캐싱 기법과 밀접한 연관관계를 가지고 협력하여야 한다.

3.4.1 캐싱 기법

단일 프로세서에서 디스크 캐싱은 디스크 I/O 성능을 효과적으로 향상하는 방법이고 메모리 캐쉬는 miss rate를 가장 많이 줄일 수 있는 방법이다[21]. 단일 프로세서 시스템과는 달리 병렬 I/O에서는 캐쉬의 위치를 고려하여야 한다. 일반적인 메모리 계층에서 캐쉬는 빠른 메모리와 느린 메모리 사이에 위치한다. 병렬 I/O 시스템에서는 이러한 경계가 병렬 보조 기억장치와 병렬 프로세서 사이로 분산된다.

화일 캐쉬를 설계할 때 버퍼 대치 정책(buffer replacement policy)과 쓰기 정책을 고려하여야 한다. 버퍼 대치 정책은 시간적 지역성(locality), 공간적 지역성, 프로세스간 지역성에 따르는 데이타 접근 형태를 반영하여야 한다. 쓰기 정책은 버퍼내의 'dirty'된 버퍼를 언제 디스크에 쓰느냐를 결정하는 정책으로 write back, delayed write, write full가 있다. 쓰기만 하는 접근 형태를 갖는 과학계산용 작업 부하에서는 write full의 성능이 가장 좋다[10].

3.4.2 선반입 기법

단일 프로세서에서 연구된 선반입 기법은 CPU 캐쉬와 주기억장치 사이 또는 주기억 장치와 디스크 사이에 명령과 데이타를 선반입하는 방법과 화일 블록을 디스크 캐쉬로 선반입하는 방법으로 나눌 수 있다.

기존에 연구된 선반입 방법으로는 현재 읽은 블록의 다음 블록을 선반입하는 순차적인 선반입방법, 과거의 접근 형태를 기반으로 미래의 접근 형태를 추론하여 화일을 비순차적으로 선반입하는 방법[22], TIP[23, 24]에서와 같이 응용프로그램이 미래의 접근 형태를 운영체제에 알리는 방법이 있다. 병렬 화일시스템에서 선반입 기법을 사용하면 수행시간을 15%에서 70% 까지 줄일 수 있다[22].

3.5 화일시스템 인터페이스

병렬 화일시스템은 UNIX 화일시스템과 같은 일반적인 화일시스템에서 제공하지 않는 특수한 기능을 포함하여야 한다. 병렬 화일시스템

인터페이스는 작업 부하 모델, 병렬 시스템 호출(system call), 분산된 데이타 관리, 병렬 요구 스케줄링과 같은 요인들을 고려하여야 한다. 병렬 화일시스템 인터페이스의 주요 목표는 사용자에게 투명성(transparency)을 제공하고 I/O 시스템의 성능을 충분히 활용하는데 있다. 병렬 화일시스템 인터페이스를 설계할 때 다음 사항을 고려하여야 한다[25].

3.5.1 일의적 디렉토리 구조

화일은 데이타가 저장된 형식(format)과 화일 접근 형태에 관계없이 간단하고 일의적인 계층 구조를 가져야 한다. 병렬 화일시스템은 사용자의 순차적인 화일과 병렬 화일을 모두 제공하여야 한다.

3.5.2 다중 화일 접근 능력

동시에 발생하는 다중 화일 접근 요구를 만족하기 위해서 여러 개의 화일 포인터를 제공해야 한다. 화일 포인터는 프로세스당 하나를 가질 수 있고 모든 프로세스가 하나의 화일 포인터를 공유할 수도 있다. 화일 포인터 수는 화일 접근 형태에 의존한다. 다중 화일 연산은 화일시스템으로 요구하는 회수를 줄이기 위해 최적으로 수행되어야 한다.

SPIFII[3, 26]에서 제공하는 4 가지 화일 포인터를 기준으로 다른 병렬 화일시스템에서 제공하는 화일 포인터는 표 3과 같다.

표 3 병렬 화일시스템에서 제공하는 화일포인터의 종류

화일포인터 종류	SPIFFI	Vesta	Bridge	CMMD	PFS	CFS	PFS
LFP	O	O	O	O	O	O	O
GFP	O	O	O	X	O	X	X
SGFP	O	O	X	O	O	X	X
DFP	O	X	X	X	X	X	X

LFP(Local File Pointer)는 UNIX 화일 포인터처럼 프로세스마다 하나의 화일 포인터를 가진다. GFP(Global File Pointer)는 동일한 화일을 접근하는 모든 프로세스들이 화일 포인터를 공유하도록 한다. 따라서 프로세스는 어느 블록을 읽을 것인지 미리 알지 못한다. GFP는 다른 프로세스가 같은 블록을 반복해서 읽지 않도록 하여 부하를 자동적으로 균등하게 한다. SGFP(Synchronized Global File Pointer)에서는 화일 포인터를 공유하는 프로세스들이 고정된 순환적인(cycle) 순서에 따라 화일 포인터를 접근하며, 프로세스가 순환적 순서에 따라 요구를 하지 않으면 차례가 될 때까지 대기한다. DFP(Distributed File Pointer)는 각 디스크 노드가 저장하고 있는 화일 부분에 대해 화일 포인터를 유지한다. 그리고 각 프로세스는 모든 블록을 읽을 때까지 같은 디스크 노드에서 읽기를 계속한다. 디스크 노드로부터 사용가능한 블록을 모두 읽으면 프로세스는 새로운 디스크 블록을 선정한다. 이 방법은 프로세스 수가 디스크 노드 수보다 적을 경우에 디스크 노드 중에 일부가 idle 상태에 있는 단점이 있다.

3.5.3 효과적인 읽기/쓰기 시스템 호출

대부분의 병렬 화일시스템은 화일을 여러 디스크에 분할하여 저장하고 각 응용 프로그램에서의 요구는 보다 작은 요구로 분할하여 각각 다른 IOP(Input Output Processor)로 보낸다. 화일시스템은 이러한 요구가 하나의 협력된 요구라는 것을 인식하고 I/O를 최적화 하기 위해 이런 정보를 사용하기는 어렵다.

모든 계산 노드가 하나의 큰 요구를 위해 협력하고 I/O 성능을 최적화하기 위해 이러한 정보를 사용하는 방법으로 collective-IO 인터페이스가 있다. 일반적으로 SPMD(Single Program Multiple Data) 프로그래밍 모델에서 collective-IO 인터페이스를 많이 사용하며 ELFS [27], two-phase I/O[28]과 directed I/O[16]에서 collective I/O를 제공한다.

I/O 중심의 응용에서는 프로세스가 동기화되지 않으면 캐쉬는 thrashing하고 캐쉬에서 프로세스간의 지역상의 장점을 얻을 수 없다.

그리고 읽기와 쓰기 시스템 호출을 설계할 때 한번에 디스크에 요구하는 기본 데이타 양을 결정하여야 한다. 요구당 읽는 데이타 양이 많으면 전체 데이타를 읽는 시간은 작아진다.

3.5.4 투명한 데이타 관리 방법

여러 개의 프로세스가 동시에 분산된 데이타를 접근하므로 병렬 화일시스템에서의 데이타

관리는 매우 중요하다. 병렬 화일시스템도 UNIX 시스템의 inode와 같은 곳에 분산된 화일의 메타데이타를 저장하여야 한다.

여러 개의 프로세서가 있고 각 프로세서에 독립된 디스크를 갖고 있는 MPP와 같은 시스템은 데이타와 화일의 메타 데이타도 분산하여 저장할 필요가 있다. 화일의 메타 데이타는 데이타의 성질을 기술하는 본질적인 메타 데이타와 위치를 나타내는 위치 데이타로 구분된다. RAMA[19]에서는 본질적인 메타 데이타는 화일 데이타의 시작 부분에 저장하고 이 부분을 저장하고 있는 프로세서가 화일에 대한 정보를 관리한다. 위치메타 데이타는 해싱함수와 line descriptor에 저장된 정보에 의해 구해진다.

3.5.5 표준 화일시스템과의 호환성

병렬 화일시스템은 일반 화일시스템과 다른 자료 구조와 데이타 관리 정책을 사용한다. 그리고 다중 화일 요구를 처리하기 위해 병렬 화일시스템은 특수한 시스템 호출을 제공하고 일반 화일시스템에서 제공하는 시스템 호출을 수행할 수 있어야 한다. 또한, 사용자는 tar, compress, df, du와 같은 화일에 관련된 일반 명령어를 병렬 화일시스템에서도 수행할 수 있어야 한다.

4. 멀티미디어 데이타 관리

멀티미디어 데이타는 본질적으로 대용량의 데이타와 실시간 성질을 요구한다. 특히, 비디오 데이타에 대해 각 표준기관에서 요구하는 데이타 대역폭은 NTSC(National Television System Committee)에서는 45Mbps, CCIR(International Radio Consultative Commitee) 601은 216Mbps, HDTV(High Definition Television)에서는 700Mbps의 대역폭을 요구한다[29]. 반면에 현재 전송 속도가 빠른 자기 디스크 드라이브는 초당 5MByte의 전송 속도를 지원하므로 하나의 디스크를 사용하면 멀티미디어 데이타를 압축하여 저장하더라도 여러 개의 비디오 데이타 대역폭 요구를 만족시킬 수 없다. 따라서 병렬 입출력 기술을 이용하여 높은 I/O 대역폭을 제공할 수 있는 멀티미디어 데이타 저장 서버에 대한 연구가 필요하다.

4.1 멀티미디어 데이타 저장 서버

VOD, Home Shopping, 전자 박물관과 같은 응용 분야에서는 대용량의 멀티미디어 데이타를 실시간으로 저장 관리하여야 한다. 본 논문에서는 이와 같은 응용분야에서 사용될 멀티미

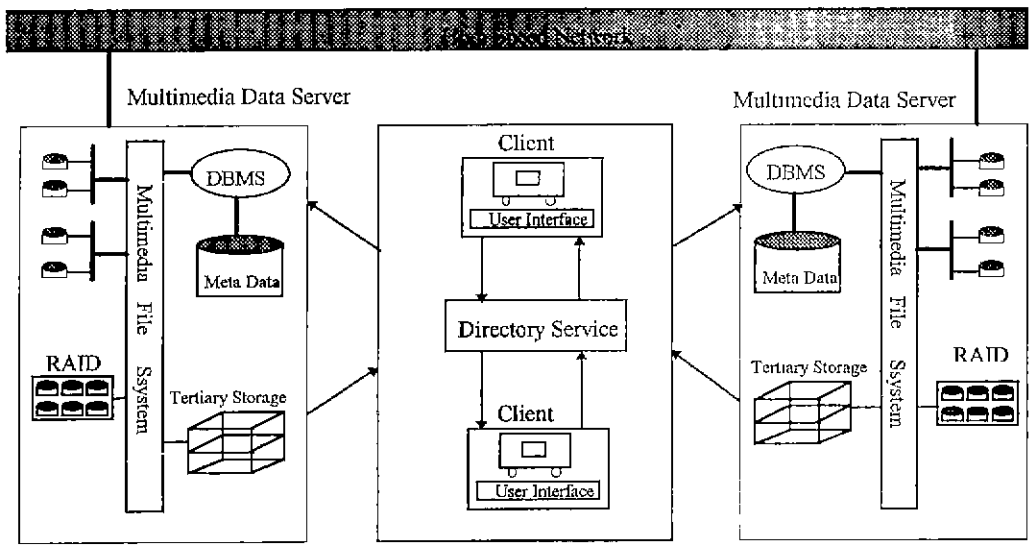


그림 3 멀티미디어 데이타 저장시스템 구조

디어 데이타 저장 서버로 그림 3과 같은 구조를 제안한다.

그림 3과 같은 구조를 갖는 저장시스템 모델을 제시한 배경은 다음과 같다.

첫째, 최근의 통신 기술 발달로 통신 대역폭은 충분히 제공되고 있다. ATM과 같은 교환기는 초당 수십 Mega bit(Mbps)에서 수 Tera bit(Tbps)까지의 통신 대역폭을 제공하고 있다. 수 Tbp정도의 통신 대역폭이 제공되면 멀티미디어 데이타를 분산 저장하여도 통신망에서 병목현상이 발생하지 않는다.

둘째, 분산된 환경에서 멀티미디어 데이타를 저장 관리하면 분산 시스템의 장점인 확장성(scalability), 신뢰성(reliability), 가용성(availability)을 얻을 수 있다.

셋째, stripe 단위로 멀티미디어 데이타를 분할 저장하면 높은 대역폭의 병렬성을 얻을 수 있다.

그림 3의 멀티미디어 데이타 저장서버는 다중 디스크, RAID, Tertiary 저장 시스템, 메타데이타를 저장하는 DBMS, 멀티미디어 데이타를 저장하는 화일시스템으로 구성되어 있다. 클라이언트가 사용자 인터페이스를 통해 멀티미디어 데이타를 요구하면 디렉토리 서버가 DBMS와 연동하여 멀티미디어의 속성(atrribute)에 대한 정보와 멀티미디어 데이타가 저장된 위치 정보를 클라이언트에게 제공한다. 그러면 클라이언트는 멀티미디어 화일시스템을 통해 멀티미디어 데이타를 접근한다.

4.2 병렬 멀티미디어 화일시스템

본 연구팀은 그림 3과 같은 구조에서 병렬 화일시스템 기술을 이용한 멀티미디어 화일시스템을 개발 중에 있다. 본 연구에서 고려하고 있는 사항은 다음과 같다.

4.2.1 VOD 특성을 반영

VOD에서 인기 있는 비디오는 주, 요일, 시간 단위로 변하는 특성이 있다. 이렇게 동적으로 변하는 작업부하에 자동적으로 적응하기 위해 데이타를 스트라이핑(striping)한다. 비디오 화일은 크기가 크고 특수한 경우를 제외하고는 대부분 순차적인 접근 형태를 가진다. 따라서 비디오의 일부 내용을 재사용하기 위해 메모리에 캐싱하는 것은 불가능하다. 그러나 인기 있는 비디오인 경우에 동시에 여러 사용자가 같은 비디오를 요청할 가능성이 높으므로 각 비디오에 대한 접근 회수 정보를 사용하여 선반입과 버퍼를 관리한다.

4.2.2 원거리 캐싱 기법 사용

캐싱할 때 고속의 네트워크를 이용하는 cooperative 캐싱 기법[30]을 VOD의 특성에 맞도록 수정하여 동시에 지원 가능한 사용자 수를 최대화한다. Cooperative 캐싱은 클라이언트가 원하는 데이타가 지역 메모리에 없으면 다른 클라이언트의 캐시에서 원하는 데이타를 가져오는 방법으로 읽기 성능을 향상시킬 수 있는 장점이 있다.

4.2.3 QoS를 고려한 Admission 제어 기능 제공

통신 부분에서 제공되는 QoS(Quality of Service) 기능을 admission 제어와 결합하여 멀티미디어 화일시스템에서 제공한다. 기존의 admission 제어를 지원하는 멀티미디어 화일시스템은 주로 디스크 헤드 스케쥴링과 버퍼 관리 방법을 결합하여 연구되었다. 그리고 대부분의 admission 제어는 최악의 상태를 가정한 결정적인 방법(deterministic)[31]으로 최대 지원 가능한 사용자 수를 결정하였다. 따라서 동시에 지원 가능한 사용자 수를 제한하는 단점이 있다. 통계적인 방법을 사용한 관찰에 기반한(Observation-based) admission 제어 방법[32]은 결정적인 방법의 문제점을 해결하나 QoS를 고려하지 않는다. 본 연구에서는 멀티미디어 데이타가 분산 저장 관리되는 시스템 구조에서 다양한 사용자의 QoS 요구를 만족할 수 있는 admission 제어 기능을 화일시스템 수준에서 제공한다.

5. 관련 연구

5.1 디스크 배열

자기 디스크의 용량은 매년 27% 증가하고 전송 비율은 매년 22% 증가하며 탐색 시간은 매년 8% 개선되고 있다. 그리고 디스크 크기도

계속적으로 감소하여 1995년도에 5.25″에서 1993년도에는 1.3″으로 줄었다. 중복 데이타를 저장하는 방법과 디스크에 고장이 발생하였을 때 회복하는 방법에 따라 일반적으로 RAID를 5가지 레벨로 분류한다[17]. 대용량 데이타 저장시스템, 병렬 화일시스템과 멀티미디어 데이타 저장시스템에 관한 최근의 연구에서는 RAID를 기본적인 데이타 저장장치로 사용하고 있다. RAID에 관한 최근 연구는 다음과 같다.

5.1.1 작은 쓰기 성능 개선

작은 쓰기를 할 때 RAID 레벨 5는 4번의 디스크 접근이 필요하다. 이러한 부담을 줄이기 위해 RAID 레벨 5에서 버퍼와 캐싱을 사용하는 방법, floating parity방법, parity logging 방법에 대한 연구가 이루어졌다.

5.1.2 투명한 온라인 회복 기법

데이타베이스, 트랜잭션처리, 비디오 서비스와 같은 응용에서 디스크 중의 하나에 고장이 발생하더라도 온라인으로 회복되어 수행이 계속되어야 한다. 표준 RAID level 5에서는 디스크가 고장나면 많은 성능 저하가 발생한다. 디스크 고장 발생으로 인해 발생하는 증가된 부하를 모든 디스크에 분산시키는 declustered parity에 대한 연구가 있다.

5.1.3 잉여 디스크 사용

고장난 디스크를 즉각 재구성하여 추가적으로 발생한 디스크 고장으로 인한 데이타 손실을 최초화하는 위한 방법으로 온라인으로 잉여 디스크(On-Line Spare Disk)를 사용하는 방법이 제시되었다.

5.1.4 데이타 스트라이핑 단위 결정

하나의 논리적 I/O 당 전송되는 데이타량을 최대화하는 방법과 모든 디스크를 동시에 사용하는 방법 사이에 trade-off가 존재한다. 모든 디스크에 데이타를 분할하면 하나의 논리적 I/O로 모든 디스크를 사용할 수 있으나 디스크당 전송되는 데이타 양이 적어진다. 따라서 스트라이핑 단위 크기를 시뮬레이션과 해석적인 모델을 사용하여 결정하는 연구가 있다.

5.1.5 성능과 신뢰성 모델링

디스크 배열 요구는 독립적으로 큐(queue)에 저장되고 서비스된다. 그리고 디스크 배열에 대한 요구는 여러 개의 디스크 요구로 분할되므로 디스크 배열의 성능 모델을 구성하기가 어렵다. 따라서 RAID의 성능과 신뢰성을 측정할 수 있는 모델링에 대한 연구가 필요하다.

5.2 병렬 데이타 저장 시스템

대용량의 데이타를 저장하고 높은 I/O 대역폭을 제공하기 위해 병렬 데이타 저장 서버에 관한 연구가 많이 이루어지고 있다. 특히, 과학계산용 뿐만 아니라 멀티미디어 데이타를 저장하기 위한 병렬 멀티미디어 데이타 저장 서버에 관한 연구가 최근에 많이 이루어지고 있다.

속도가 빠르고 신뢰성이 높은 네트워크를 기반으로 하는 RAID 형태의 저장 서버로 TickerTAIP[33], RAID-II[34], Scotch 저장 서버[24], HPSS(High Performance Storage System) 등이 있다. TickerTAIP는 RAID 제어기의 병목 문제를 해결하고 호스트에 여러 개의 접속점을 가지고 있는 고장 허용(fault-tolerant) 형의 구조를 가지고 있다. RAID-II는 고속의 네트워크와 디스크 배열을 이용하여 높은 대역폭을 얻도록 하드웨어를 설계하고 이 하드웨어를 효과적으로 이용하기 위하여 LFS (Log-structured File System)을 구현하였다. Scotch 저장 서버는 CMU에서는 새로운 RAID 구조와 RAID 제어기에서 오류처리를 간단히 할 수 있는 코딩 방법을 개발하기 위한 테스트베드로 구현된 시스템이다. HPSS은 미국 NSL (National Storage Laboratory)에서 IEEE Mass Storage System Reference Model(V5)를 기반으로 설계하고 슈퍼컴퓨터, 워크스테이션 클라스트와 병렬컴퓨터에서 사용 가능한 고성능 병렬 저장시스템이다.

멀티미디어 미디어 데이타를 저장 관리하기 위한 것으로는 GigaView 이미지 서버, Lancester 대학의 CMSS, Oracle Media 서버, Streaming RAID 등이 있다.

5.3 병렬 화일시스템

병렬 화일시스템은 연구용에서부터 상용화

된 것에 이르기까지 다양한 형태로 연구 개발되고 있다. 라이브러리 수준에서 제공하는 병렬 화일시스템으로는 PPFS(Portable Parallel File System), PIOUS, ELFS[27] 등이 있다. PPFS는 기존의 UNIX 화일시스템을 이용하여 구현된 것으로 클라이언트/서버 모델을 기반으로 하고 있다. PIOUS는 PVM(Parallel Virtual Machine) 상에서 수행되는 화일시스템으로 데이타 declustering과 트랜잭션에 기반한 병행제어(concurrency control) 방법을 사용한다. ELFS는 객체지향 개념을 사용한 화일시스템으로 고성능 I/O 문제를 해결하기 위해 화일 구조를 데이타의 형과 응용의 접근 형태에 일치시켰다. ELFS는 선반입 기법, 캐싱 기법과 병렬 검색 기술을 선택적으로 사용할 수 있다.

상용화된 병렬화일시스템으로는 Intel의 CFS/PFS, IBM의 Vesta[35], CM-5의 sfs 등이 있다. CFS는 Intel TouchStone에서 수행되며 사용자는 화일이 분산되는 디스크 수를 제한할 수 있다. CFS 화일은 화일에 관련된 메타 데이타를 저장하는 헤드와 실제 데이타인 몸체로 구성되어 있다. PFS는 Intel Paragon에서 수행되며 화일을 여러 디스크 또는 RAID로 분산 저장한다. Sfs는 CM-5에서 수행되며 UNIX와 호환성을 유지하기 위해 UFS(UNIX File System)를 수정한 화일시스템이다. Vesta는 IBM SP-n에서 수행되며, 사용자는 여러 개의 디스크에 화일을 분산시켜 화일을 생성할 수 있고 화일이 저장되는 노드의 수, 스트라이핑 되는 기본 블록의 크기 등을 기술할 수 있다.

MPI-IO[36] 인터페이스는 I/O를 메시지 전송으로 -쓰기는 메시지를 송신하는 것으로, 읽기는 메시지를 수신하는 것으로 -모델 할 수 있다는 사실을 기반으로 한다. MPI-IO는 데이타 분할 기능, collective 인터페이스, 비동기적 I/O 연산, 물리적 화일의 layout을 최적화 할 수 있는 기능을 제공한다.

이 외에도 SPIFFI, RAMA, Hurricane, SPFS[24]와 같은 병렬화일시스템이 있으며, 데이타를 분산 저장하나 응용 계층에서 분산과 정책 제어를 제공하지 않는 Zebra[37], Swift[38] 같은 분산형 화일시스템에 관한 연구가 있다.

최근에는 SPIFFI와 Vesta 같은 병렬 화일시스템을 멀티미디어 저장시스템으로 활용하기 위한 연구가 있다.

5.4 성능 측정

이상적인 I/O 벤치마크는 시스템 성능을 이해하는데 도움이 되어야 하고, I/O를 중심으로 측정하고, 다양한 형태의 기계에서 수행 가능하고, 여러 기계를 비교 가능하고, 다양한 응용을 반영하고, 기능을 명시화할 수 있는 특성이 있어야 한다[39]. 병렬 처리시스템의 계산 능력을 측정하기 위한 도구는 많이 연구되었으나 I/O의 성능을 신뢰성 있고 공정하게 측정할 수 있는 벤치마크에 대한 연구는 부족하다. 병렬 I/O의 성능을 측정하는 것으로는 NHT-1[40]와 PIOS가 있다. NHT-1은 NASA의 NAS(Numerical Aerodynamics Simulation)에서 처리하는 응용에 맞는 작업부하에 대해서 I/O 성능을 측정하는 벤치마크이다. PIOS는 시뮬레이션 방식을 사용하여 병렬 I/O 시스템의 성능을 측정한다.

6. 결 론

본 논문에서는 병렬 I/O를 요구하는 응용 분야의 I/O 접근 형태와 병렬 입출력 프로그램을 지원하는 방법, 병렬 입출력시스템 설계시 고려사항, 본 연구팀에서 연구 중인 대용량 멀티미디어 데이타 저장 서버와 관련된 연구를 살펴보았다. 병렬 입출력시스템 연구와 관련하여 유의하여야 할 사항과 추가적으로 연구되어야 할 부분은 다음과 같다.

첫째, 고성능 I/O 연구를 위하여 응용 수준에서 작업 부하를 분석할 때 프로그래머는 응용 프로그램이 수행되는 병렬처리시스템의 처리 능력을 고려한다는 사실을 유의하여야 한다. 예로, CM-5는 독립된 I/O 성능이 좋지 않으므로 CM-5에서 수행되는 응용 프로그램은 지역적으로 독립된 고성능(local-independence) I/O를 사용하지 않는다. 따라서 CM-5에서 추출한 데이타를 토대로 과학계산용 응용프로그램은 지역적으로 독립된 형태로 I/O를 하지 않는다고 분석하여서는 안된다.

둘째, 입출력시스템의 병렬화는 처리율을 높

일 수 있으나 지연을 줄이지 못하는 문제점이 있다. 이 문제를 해결하기 위한 방법 중의 하나로 고속 통신과 메모리를 사용하여 지연을 줄일 수 있는 효과적인 원거리 캐싱 기법에 대한 연구가 필요하다.

셋째, 이전의 접근 형태를 토대로 추측에 기반한 선반입 기법은 성능을 개선하기보다는 성능을 저하시킬 위험이 있다. 따라서 몇 가지 요구 접근에 의해 이론이 확인될 때까지 추측에 기반한 선반입 기법은 신중하게 사용하여야 한다.

넷째, 순차적인 성질을 갖는 UNIX 화일은 병렬 I/O에 적합하지 못하다. 그러나 존재하는 시스템과의 호환성을 위해서 병렬 화일시스템에서는 UNIX 인터페이스를 제공하여야 한다.

다섯째, 프로그래머는 입출력 성능을 최대화하기 위해서 접근 형태를 시스템에서 제공하는 최선의 접근 형태로 수정해야 하는 부담이 있다. 이러한 문제점을 해결하기 위해 병렬 화일시스템은 다양한 접근 형태와 요구의 크기를 제공하여야 한다. 그리고 사용자가 화일 접근 형태, 화일 분산, 캐싱, 선반입 정책을 선택할 수 있어야 한다.

여섯째, 병렬 화일시스템에 객체지향과 collective-I/O와 같은 새로운 개념을 적용하여야 한다.

일곱째, 멀티미디어 데이타의 특성을 고려한 병렬 멀티미디어 화일시스템의 기초 기술에 대한 연구가 필요하다.

참 고 문 헌

[1] John Hennessy and David Patterson, Computer Architecture : A Quantitative Approach, Morgan Kaufmann Publishers, 1990.

[2] J. M. del Rosario and A. Choudary, "High Performance I/O for Parallel Computers : Problems and Prospects," IEEE Computers, Vol.27 No.3, pp.59-68, March 1994.

[3] Crag S. Freedman and David J. DeWitt, "The SPIFFI Scalable Video-on-Demand System," Proceddings of the 1995 ACM SIGMOD, pp.352-363, May 1995.

[4] David J. Dewitt, Navin Kabra, Jun Luo, Jignesh M. Patel, and Jie-Bing Yu, "Client-Server Paradise," Proc. of the 20th VLDB Conf., 1994.

[5] Chris Ruemmler and Jhon Wilkes, "UNIX disk access pattern," Proc. of 1993 Winter USENIX, pp.405-420, Jan. 1993.

[6] J. K. Ousterout, et al. "A Trace-Driven Analysis of the UNIX 4.2 BSD File System," Proc. of 10th Symp. on OS Principles, pp.15-24, Dec. 1985.

[7] M. G. Baker et al., "Measurements of a Distributed File System," Proc. of the 13th Symp. on OS Principles, pp.198-212, Oct. 1991.

[8] John Ousterhout and Fred Douglis, "Beating the I/O Bottleneck : A Case for Log-Structured File System," Operating Systems Review, Nol.23 No.1, pp.11-27, Jan. 1989.

[9] T. W. Crockett. "File concepts for Parallel I/O," Proc. of Supercomputing'89, pp.574-579, 1989.

[10] David Kotz and Carla Schlatter Ellis, "Caching and Writeback Policies in Parallel File Systems," Journal of Parallel and Distributed Computing, Vol.17, No.1-2., pp.140-145, Jan./Feb. 1993.

[11] E. L. Miller, R. H. Katz, "Input/Output Behavior of Supercomputing Application," Proc. of Supercomputing'91, pp.567-576, Nov. 1991.

[12] A. Purakayastha, Carla Ellis, David Kotz, N. Nieuwejaar and Michael L. Best, "Characterizing Parallel File-access Patterns on a Large-scale Multiprocessor," Proc. of 9th International Parallel Processing Symposium, pp.165-172, 1995.

[13] David Kotz and N. Nieuwejaar, "Dynamic File-Access Characteristics of a Production Parallel Scientific Workload," Proc. Supercomputing'94, pp.640-649, Nov. 1994.

[14] Alok Choudhary, Rajesh Bordawekar, Michael Harry and Rakesh Krishnaiyer, PASSION : Parallel And Scalable Software for Input-Output, Tech. Report, Syracuse University, Sep. 1994.

[15] Ravi Jain, Kiran Somalwar, John Werth

and J. C. Browne, "Scheduling Parallel I/O Operations," Computer Architecture News, Vol.21, No.5, pp.47-54, Dec. 1993.

[16] D. Kotz "Disk-directed I/O for MIMD Multiprocessor," Proc. of 1st USENIX Symp. on OSDI, pp.61-74, Nov. 1994.

[17] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz and D. P. Patterson, "RAID : High-Performance, Reliable Srcondary Storage," ACM Computing Surveys, Vol.26 No.2 pp. 145-185, 1994.

[18] Peter M. Chen and David Patterson, "Maximizing Performance in a Striped Disk Array," proc. of the 17th ISCA, pp.322-331, May 1990.

[19] Ethan L. Miller and Randy H. Katz, "RAMA : Easy Access to a High-Bandwidth Massively Parallel File System," Proc. of the 1995 Winter USENIX Conf., 1995.

[20] Martin W. Sachs, Avraham leff, and Denise Sevigny, "LAN and I/O Convergence : A Survey of the Issues," IEEE Computer, pp.24-32, Dec. 1994.

[21] Alan Jay Smith, "Disk cache-miss ratio analysis and design consideration," ACM Tran. on Computer Systems, Vol.3 No.3, pp.161-203, 1985.

[22] D. Kotz and C. S Ellis, "Practical Prefetching Techniques for Parallel File Systems," Proc. First Int. Conf. on PDIS, pp. 182-189, Dec. 1991.

[23] H. Patterson, G. Gibson, and M. Satyanarayanan, "A Status Report on research in Transparent Informed prefetching," Operating Systems Review, Vol.27, No.2, pp.21-34, April 1993.

[24] Garth A. Gibson, et. al. "The Scotch Parallel Storage System," Proc. 1995 IEEE CompCon Conf., May, 1995.

[25] Rajesh R. Bordawekar, Issues in Software Support for Parallel I/O, Master Thesis, Syracuse University, May 1993.

[26] Crag S. Freedman, J. Burger, and David J. DeWitt, "SPIFFI-A Scalable Parallel File System for the Intel Paragon," submitted IEEE Transaction on Parallel and Distributed System

[27] John F. karpovich, Andrew S. Grimshaw, and James C. French, "ExtensibLe File Systems(ELFS) : An Object-Oriented Approach to High Performance File I/O," Proc. of the 9th Conf. on OOPSLA, Oct. 1994.

[28] Juan Miguel del Rosario, Rajesh Bordawekar and Alok Choudhary, "Improved Parallel I/O via a two-phase Run-time Access Strategy," Computer Arcitecture News, Vol.21, No.5, pp.31-38, Dec. 1993.

[29] Shahram Ghandeharizadeh, Cyrus Shahabi, Luis Ramos, "An Overvie of Techniques to Support Continuous Retrieval of Multimedia Objects," Computer Arcitecture News, Vol.21, No.5, pp.39-46, Dec. 1993.

[30] Michael D. Dahlin, Radolph Y. Wang, Thomas E. Anderson and David A. Patterson, "Cooperative Caching : Using Remote Client Memory to improve File System performance," Proc. of 1st USENIX Symp. on OSDI, pp.267-280, Nov. 1994.

[31] J. Gemmell and S. Christodoulakis, "Principles of delay Sensitive Multimedia Data Storage and Retrieval," ACM Transaction on Information Systems, Vol.10 No.1, pp. 51-90, 1992.

[32] Harrick M. Vin, Alok Goyal, Anshuman Goyal, and Pawan Goyal, "An Observation-based Admission Control Algorithm for Multimedia Servers," Proc. of Interantional Conference on Multimedia Computing and Systems, pp.234-243 May 1994.

[33] Pei Cao, Swee Boon Lim, Shivakumar Venkataraman, and John Wilk, "The TickerTAIP Parallel RAID Architecture," Proc. of 1993 Int. Sym. on Computer Architecture, pp., May, 1993.

[34] Ann L. Drapeau, et al., "RAID-II : High-bandwidth Network File Server," Proc. of 21st ISCA, pp.234-244, 1994.

[35] Peter F. Corbett, Sandra Johnson baylor, and Dror G. Feitelson, "Overview of the Vesta Parallel File System," Computer

Arcitecture News, Vol.21, No.5, pp.39-46, Dec. 1993.

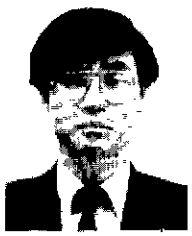
[36] Peter Corbett, "Overview of the MPI-IO Parallel I/O Interface," Third Workshop on I/O in Parallel and Distributed Systems, April 1995.

[37] Cabrera L. Long, D. D. E, "Swift : Using Distributed Disk Striping to Provide High I/O Data rates," Computer Systems, Vol.4, No.4, pp.405-436, 1991.

[38] J. H. Hartman and J. K. Ousterhout, "The Zebra Striped Network File," Proc. 14th ACM Sym. on OS Principles, pp.29-43, 1994.

[39] Peter M. Chen, David A. Patterson, "A New Approach to I/O Performance Evaluation-Self-Scaling I/O Benchmarks, Predicted I/O Performance," ACM Transactions on Compuer Systems, Vol.12, No.4, pp.308-339, Nov. 1994.

[40] Samuel A. Fineberg, "Implementing the NHT-1 Application I/O Benchmark," Computer Architecture News, Vol.21, No.5, pp. 23-30, Dec. 1993.



노 영 욱

1985 부산대학교 계산통계학과 졸업(이학사)
1989 부산대학교 계산통계학과 졸업(이학 석사)
1989~1993 한국전자통신연구소 연구원
1993~현재 부산대학교 전자계산학과 박사과정
관심분야 : 병렬처리, 분산처리, 멀티미디어

정 기 동

1973 서울대학교 공과대학 졸업(공학사)
1975 서울대학교 공과대학 졸업(공학 석사)
1986 서울대학교 계산통계학과 졸업(이학박사)
1990~1991 MIT, South Carolina 대학 교환교수
1978~현재 부산대학교 전자계산학과 교수
1993~현재 부산대학교 정보통신연구소 소장
1993~현재 한국정보과학회 이사
관심 분야 : 병렬처리, 분산처리, 멀티미디어, 운영체제, 프로그래밍언어