분산 UNIX 환경에서 Shared-Concurrent File System의 설계 및 구현

장 시 웅[†] · 정 기 동[††]

요 약

본 연구에서는 소규모 서버로 사용될 수 있는 Workstation Cluster 환경에서 전통적인 디스크들을 디스크 배열 처럼 사용할 수 있는 병행 화일시스템(S-CFS)을 설계하고 구현하였다. S-CFS는 범용의 UNIX 운영체제를 기반으로 구현되어서 융통성과 이식성이 높으며 별도의 입출력 노드가 불필요하므로 시스템 자원을 효율적으로 사용한다. 성능 분석 결과에 의하면, 소규모 서버에서 디스크의 수가 충분할 경우, 트랜잭션 처리에서의 병행 화일시스템의 성능은 CPU 계산 능력에 의하여 제한받는 것으로 나타났으며 대용량 데이타 입출력에서는 성능이 버퍼 간의 데이타 복사시간에 의하여 제한받는 것으로 나타났다. Workstation Cluster에서 구현된 병행 화일시스템은 8개의 디스크에서 트랜잭션 처리의 경우에는 초당 388 트랜잭션의 처리율을 보였으며, 대용량 데이타의 경우에는 15.8 MBytes/sec의 대역폭을 보였다. 그리고 사용자가 병행 화일시스템의 병렬성을 제어할 수 있도록 설계함으로써 고속 입출력을 요구하는 사용자의 처리율을 높일 수 있도록 하였다.

Design and implementation of a Shared-Concurrent File System in distributed UNIX environment

Si Woong Jang[†] · Ki Dong Chung[††]

ABSTRACT

In this paper, a shared-concurrent file system (S-CFS) is designed and implemented using conventional disks as disk arrays on a Workstation Cluster which can be used as a small-scale server. Since it is implemented on UNIX operating systems, S-CFS is not only portable and flexible but also efficient in resource usage because it does not require additional I/O nodes. The result of the research shows that on small-scale systems with enough disks, the performance of the concurrent file system on transaction processing applications is bounded by the

• 이 논문은 1994년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음
† 정 회 원 : 동의대학교 전산통계학과 전임강사
†† 종신회원 : 부산대학교 전자계산소장
논문접수 : 1996년 1월 24일, 심사완료 : 1996년 4월 2일

bottleneck of CPUs computing powers while the performance of the concurrent file system on massive data I/Os is bounded by the time required to copy data between buffers. The concurrent file system, which has been implemented on a Workstation Cluster with 8 disks, shows a throughput of 388 tps in case of transaction processing applications and can provide the bandwidth of 15.8 MBytes/sec in case of massive data processing applications. Moreover, the concurrent file system has been designed to enhance the throughput of applications requiring high performance I/O by controlling the parallelism of the concurrent file system on user's side.

1. 서 론

반도체 기술의 발전으로 처리기의 속도는 지난 수십년간 10^3배 정도의 개선을 이루었고 광통신 기술의 발전으로 통신망의 속도는 10^5배 이상으로 개선되었다. 반면, 디스크의 속도는 탐색시간, 회전지연시간 등과 같은 기계적 특성의 제한에 의해 커다란 진전을 보지 못했다. 지난 10여년 간 디스크 장치의 발전은 용량에 있어 10배 정도의 발전을 이루었지만 성능에 있어서는 2배 정도의 향상을 이루는데 그쳤다[1, 2].

이러한 시스템 요소간 불균형적 발전은 디스크 입출력의 성능 향상에 관한 연구를 촉진시켰다. 디스크 입출력의 성능을 개선하기 위한 방법에는 디스크 캐쉬, 디스크 헤드 스케줄링, 디스크 배열 등이 있으며 이들 중 디스크 캐쉬와 디스크 헤드 스케줄링에 관한 연구는 과거에 많이 연구되어 온 전형적 기법이며 최근에는 디스크 배열에 관한 연구가 많이 이루어져 왔다[3, 4, 5, 6].

디스크 배열에 관한 연구는 디스크 제어기 수준의 관점에서 다중 디스크를 제어하는 RAID 시스템과 화일 시스템 수준에서 다중 디스크를 제어하는 병렬 화일시스템으로 분류할 수 있다. RAID 시스템은 RAID 등급 조정과 같은 일부 사용자 옵션이 존재하나 근본적으로 RAID 제품 자체 내에 있는 디스크들을 사용자의 목적에 맞게 재구성할 수 없다는 단점이 있다[7]. RAID 제품은 전체 시스템의 입장에서 볼 때 디스크 제어기 수준에서 구현한 종합 디스크 시스템이다. 따라서 RAID 제품 자체로는 종래에 사용되고 있는 전통적인 디스크들을 병렬화 하여 사용할 수 없다.

디스크 배열에 관한 연구를 RAID 제품과 병렬 화일시스템에 관한 연구로 분류할 경우, RAID는 주로 트랜잭션 처리 위주로 응용이 맞추어져 있는 반면 병렬 화일시스템은 과학계산용이나 멀티미디어 데이타 처리와 같은 대규모 데이타 처리에 응용되고 있다.

현재 멀티미디어 처리 기술의 발전으로 멀티미디어와 같은 대용량 데이타의 고속처리가 요구되는 멀티미디어 서버로서의 기능을 담당할 저장장치에 관한 연구가 많이 진행되고 있으나 대부분의 연구에서는 성능 분석에 관한 체계적인 연구를 수행하지 않았다. 또한 각 연구에서 개발된 시스템들은 특수한 컴퓨터나 특정 목적을 겨냥하고 있어서 이식성과 융통성이 결여되어 있으며 병렬 화일시스템의 중요 요소인 병렬성 제어에 관한 연구가 부족하다.

따라서 본 논문에서는 로컬 네트워크를 통하여 다수의 Workstation들이 연결되어 있는 분산 UNIX 환경 하에서 로컬 네트워크 내에 있는 다수의 디스크 자원들을 디스크 배열처럼 공유하여 사용할 수 있는 Shared-Concurrent File System(S-CFS)을 설계, 구현하여 성능을 평가하고 분석한다.

본 논문에서 제안하고 설계한 S-CFS는 50여명 정도의 동시 사용자가 사용할 수 있는 범용의 시스템에서 구현되어 이식성과 융통성이 높고 사용자가 작업의 종류에 따라 병렬성을 제어할 수 있어서 멀티미디어 응용이나 트랜잭션 처리 등과 같은 다양한 목적으로 사용될 수 있다.

2. 병렬 화일시스템의 고찰

병렬 화일시스템에 관한 연구는 응용 목적에 따라 과학계산 응용, 트랜잭션 처리 응용, 멀티미디어 응용 등으로 분류된다. 병렬 화일시스템은 저자에 따라 병행 화일시스템과 구분하여 분류하기도 하지만 본 연구에서는 병행 화일시스템을 병렬 화일시스템과 동일한 종류로 본다. 따라서 본 논문에서는 기존 연구

로서 병행 화일시스템과 병렬 화일시스템을 함께 비교 분석한다.

2.1 iPSC/2 CFS

Intel의 iPSC/2 상에서 구현되어 사용되고 있는 CFS[8]는 큰 데이타 화일 접근시 병렬 입출력 기능을 제공함으로써 입출력 성능 향상을 지원하기 위해 구성된 패키지 형태의 소프트웨어이다. iPSC/2에서는 어느 정도 이상의 크기를 갖는 화일들은 모두 블럭 단위(4 KBytes)로 나뉘어져 각 입출력 노드에 분산 저장되고 화일을 접근할 때는 기존의 UNIX 입출력 인터페이스와 비슷한 형태로 제공되는 라이브러리를 사용한다. iPSC/2에서는 호스트의 화일들과 CFS에 의해 관리되는 화일들을 구분하기 위해 CFS에 의해 관리되는 화일을 접근할 때는 항상 접두사로 "/cfs/"를 화일명 앞에 쓰도록 한다. 데이타가 입출력 노드에만 분산되어 있어서 많은 입출력 노드가 필요하므로 이에 따른 비용이 많이 드는 것이 단점이다.

iPSC/2 CFS는 iPSC/2와 같은 특정 H/W에 기반하는 화일시스템으로서 이식성과 융통성이 부족하며 최초의 병행 화일시스템이라는 의의를 가지지만 다른 병렬 화일시스템에 비해 성능이 현저히 떨어진다. 또한 화일시스템의 병렬성이 요구된 데이타가 스트라이핑 되어 있는 디스크 수에 의해 단순히 결정되므로 화일 입출력의 병렬성 제어에 관한 융통성이 없다.

2.2 Vesta 병렬 화일시스템

Vesta 병렬 화일시스템[9]은 IBM T. J. Watson 연구소에서 개발 중인 초대규모 병렬 처리 시스템인 Vulcan에서 사용하기 위하여 개발된 병렬 화일시스템으로 수치연산이 많은 과학계산 응용에서 MPP의 I/O 문제를 해결하는 것을 목표로 삼고 있다.

Vesta는 Vulcan의 병렬 I/O 하드웨어 능력을 최대한 사용하기 위해 설계되었으며 사용자가 화일을 분할할 수 있는 기능을 제공한다. 화일은 사용자의 지정에 따라 여러 개의 I/O 노드로 물리적으로 분할되어 배치된다. 화일이 사용자에 의해 기술된 I/O 노드로 분산할 수 있다는 점이 화일시스템이 암시적으로 데이타를 여러 노드로 분산하는 병행 화일시스템과 다른 점이다.

Vesta는 특정 화일에 대한 화일 접근 형태가 항상 일정할 것이라는 가정하에 화일 생성시에 화일 접근 형태를 사용자가 지정함으로써 화일 입출력의 병렬성을 제어하도록 하였으나 화일의 접근 형태가 일정하지 않은 경우에는 화일 입출력의 병렬성을 전혀 얻지 못하는 단점이 있다.

본 논문에서는 화일 입출력의 병렬성을 사용자가 화일을 읽고 쓸 때 옵션으로 제어할 수 있도록 하였다. S-CFS에서는 사용자의 화일 입출력 요구를 데몬 프로세스가 받아 사용자가 원하는 데이타 입출력을 수행해 주며 각각의 데몬 프로세스는 임의의 디스크를 접근할 수 있다. 따라서 사용자는 화일 입출력시 자신의 입출력 요구를 수행해 줄 데몬의 수를 명시함으로써 화일 입출력의 병렬성을 제어한다.

2.3 sfs 화일시스템

sfs(scalable file system)[10]는 확장 가능한 디스크 배열을 갖는 Thinking Machine사의 CM-5에서 UNIX와 호환을 유지하면서 고도의 확장성을 갖도록 설계된 병렬 화일시스템이다. sfs는 사용자의 병렬 I/O 요구를 전송하기 위해 시분할 데몬을 사용한다. 사용자가 I/O를 요구하면 시분할 데몬은 화일시스템에 해당 디스크 블럭의 목록을 요청한다. 화일시스템은 블럭 정보를 시분할 데몬에 알려 주고 시분할 데몬은 SDA(Scalable Disk Array)에 I/O 요구를 보낸다. 결과적으로 sfs에서는 고성능 화일시스템 기능을 성취하기 위해 시분할 데몬, SDA, 화일시스템이 협력하여 화일 입출력을 수행하게 된다.

sfs는 SDA를 사용함으로써 전통적인 입출력 버스나 전통적인 디스크가 아닌 별도의 H/W가 필요하게 된다. 따라서, 보편적으로 많이 사용되는 Workstation 환경 등에 호환성을 가지고 사용될 수 없다.

2.4 RAID-Ⅱ

RAID-Ⅱ[11]는 화일서버와 클라이언트에게 디스크 배열의 높은 대역폭을 제공하기 위해 H/W와 S/W를 설계하였다. RAID-Ⅱ H/W는 디스크 시스템과 네트워크 사이에 상호 연결망을 사용하여 높은 대역폭의 네트워크와 낮은 대역폭의 네트워크를 동시에 제공한다. 높은 대역폭의 네트워크는 대용량 데이타를 고속으로 전송하기 위하여 사용되며 낮은 대역폭의 네트워크는 제어 연산과 같은 작은 데이타 전송을

위하여 사용된다. 디스크와 HIPPI 네트워크 사이의 데이타 전송을 위해 호스트 Workstation의 메모리를 통과하지 않는 높은 대역폭의 데이타 경로를 사용하며 화일 이름과 디스크 내의 데이타 위치를 찾기 위한 메타 데이타를 호스트에 보내기 위해 XBUS와 호스트 컴퓨터 사이에 연결된 낮은 대역폭의 VME를 사용한다.

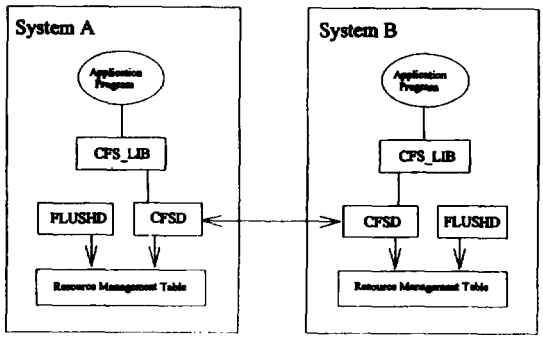
RAID-Ⅱ는 XBUS 보드와 같은 별도의 H/W에 기반을 두었으므로 융통성과 이식성이 결여되어 있고 각 보드에 대한 최대 성능이 기술되어 있으나 각종 성능에 대한 체계적인 분석이 이루어지지 않았다.

3. S-CFS의 설계 및 구현

본 장에서는 네트워크 상에 존재하는 전통적인 디스크들을 디스크 배열 형태로 공유하여 사용할 수 있도록 지원하는 S-CFS를 설계하고 구현한다. 본 연구에서는 시스템 구성 정보에 따라 로컬 시스템에 있는 전통적인 디스크들을 디스크 배열 형태로 사용할 수도 있고, 네트워크 상에 분산되어 있는 디스크들을 디스크 배열 형태로 사용할 수 있도록 융통성 있게 설계하고 구현한다.

3.1 S-CFS의 구조

S-CFS는 시스템의 이식성과 융통성을 위해 커널의 변경 없이 시스템 라이브러리와 화일서버로 구현한다. 응용 프로그램 수준에서 구현하는 경우에 오버헤드로 인한 시스템 성능 저하는 UNIX 화일시스템에서 제공하는 생 디스크 인터페이스(raw disk interface) 기능을 사용함으로써 최소화한다.

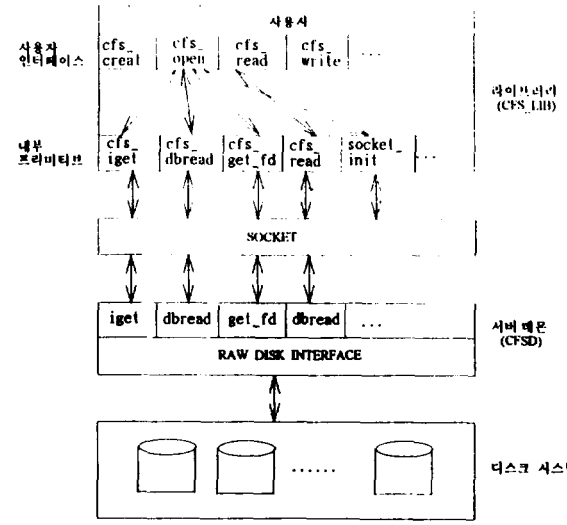


(그림 1) S-CFS의 소프트웨어 블럭 구성도
(Fig. 1) Software block diagram of S-CFS

S-CFS는 (그림 1)에 보여진 것처럼 CFS_LIB(Concurrent File System Library), CFSD (Concurrent File System Daemon), FLUSHD(Flush Daemon)의 3개 소프트웨어 블럭으로 구성된다. CFS_LIB는 사용자 모듈에 함께 링크되어 동작하며, CFSD는 라이브러리로부터의 요구를 받아 실제 입출력을 수행하고 자원을 관리한다. FLUSHD는 inode 등과 같은 메타 데이타들을 주기억장치에서 디스크로 주기적으로 flush 하는 역할을 수행한다. 각 블럭의 상세한 기능은 다음과 같다.

3.1.1 CFS_LIB

CFS_LIB는 사용자 프로그램과 서버 데몬 사이의 인터페이스 역할을 담당한다. S-CFS에서 서버 데몬은 한 번에 하나의 블럭에 대한 입출력만을 처리하고 라이브러리에서 사용자의 데이타를 여러 개의 단일 블럭으로 분할하여 각각의 블럭을 서로 다른 서버 데몬에게 입출력을 수행하도록 함으로써 화일 입출력의 병렬성을 제어한다. 이 때, 라이브러리는 사용자가 명시한 요구 병렬성에 따라 해당 입출력을 서비스할



(그림 2) 라이브러리와 서버 데몬 간의 동작 메카니즘
(Fig. 2) Communication mechanism between the library and servers

데몬의 수를 결정한다.

S-CFS는 기존의 시스템에서 제공하는 화일시스템이 아니므로 기존의 시스템 호출로는 접근이 불가능하다. 따라서 사용자가 S-CFS를 접근할 수 있도록 하기 위한 인터페이스로서 입출력 라이브러리를 제공하며 이 라이브러리는 기존의 화일 관련 시스템콜과 같은 형식으로 응용 프로그램에서 사용할 수 있도록 설계되어 있다. (그림 2)는 사용자, 라이브러리, 서버 데몬 사이의 상호 인터페이스 관계를 설명한다.

사용자가 입출력 함수들을 사용하여 입출력을 요구하면 라이브러리는 로컬 시스템의 서버 데몬에게 필요한 기능을 수행하도록 요청하고 그 결과를 돌려받아 사용자에게 되돌려 주는 역할을 한다.

3.1.2 CFSD

CFSD는 클라이언트(라이브러리) 및 다른 시스템의 서버로부터 오는 요청을 처리하기 위해 동작하는 서버 데몬으로서 시스템에 하나 이상의 CFSD가 대기상태로 존재한다. CFSD는 처음 구동될 때 클라이언트로부터의 요구를 서비스하기 위해 소켓을 초기화하고 클라이언트로부터의 병렬 요구를 동시에 처리하기 위해 다수의 데몬 프로세스를 생성한다. 이때 생성할 프로세스의 수는 운용자가 지정한다. 각 CFSD는 특정 포트로 바인드되며 S-CFS를 접근하는 응용 프로그램은 바인드된 포트를 통해 로컬 시스템에서 동작하는 CFSD에게 원하는 기능을 요청한다. CFSD는 각 요청을 분석하여 로컬 시스템에 있는 서브디스크에 대한 요청이면 스스로 처리하고 그 외의 서브디스크에 대한 요청은 해당 서브디스크를 관리하는 리모트 시스템의 CFSD에게 요청하여 처리하도록 한다.

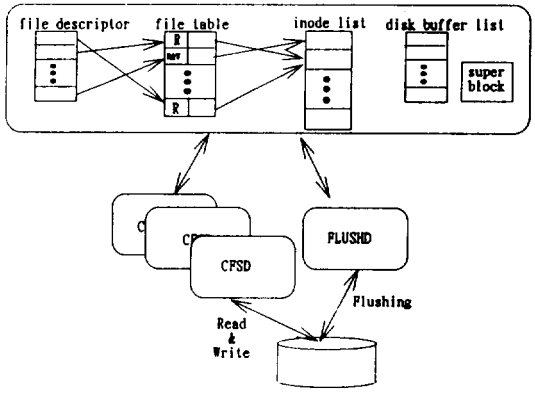
3.1.3 FLUSHD

FLUSHD는 S-CFS에서 데몬들 사이에 공유하는 자원들(버퍼, 화일 테이블, inode 리스트 등)을 (그림 3)과 같이 관리하는 역할을 수행한다.

FLUSHD는 각 시스템에 하나씩 동작하며 다음과 같은 두 가지 기능을 가지도록 설계되었다.

첫째, 버퍼나 inode 리스트 및 화일 테이블과 화일 디스크립터 테이블 등 FLUSHD와 CFSD에 의해 공유되는 자원들을 공유 메모리 상에서 초기화시킨다.

둘째, 버퍼나 inode와 같이 메모리와 디스크 간에 동기화시켜야 하는 자원들을 주기적으로 디스크에 플러쉬시킨다.



(그림 3) CFSD와 FLUSHD의 메타 데이타 접근
(Fig. 3) Meta data access in CFSD and FLUSHD

3.2 사용자 인터페이스 설계

병렬 화일시스템에서 사용자 인터페이스는 화일시스템의 성능을 제어할 수 있는 주요 요소가 된다. 대부분의 병렬 화일시스템은 화일 입출력의 병렬성을 디스크의 개수로 암시적으로 정의한 반면 Vesta 화일시스템에서는 화일 입출력의 병렬성을 화일 생성시 화일 접근 형태를 사용자가 지정하도록 함으로써 화일 입출력의 병렬성을 제어하였다.

본 연구에서는 사용자가 화일 입출력시 서버 데몬을 통하여 디스크 입출력을 수행하도록 화일시스템을 설계하였다. 따라서 사용자가 화일 입출력을 수행할 때 몇 개의 서버 데몬이 해당 사용자를 위해 디스크 입출력을 수행하는지의 여부에 따라 화일시스템의 병렬성이 결정된다. 디스크 개수만큼의 서버 데몬이 하나의 사용자를 위해 디스크 입출력을 수행할 경우에 병렬성이 최대가 된다. 그러나 사용자가 많은 경우에는 디스크 접근시 데몬들 간의 충돌이 발생하여 개별 사용자들의 병렬성은 낮아진다.

그러므로 본 연구에서는 화일 입출력시 사용자가 서버 데몬의 수를 지정함으로써 화일 입출력의 병렬성을 제어하도록 하였다. 고속 입출력이 필요하지 않은 사용자는 화일 입출력의 병렬성을 디폴트 값인 1

로 하고 고속 입출력이 필요한 사용자만 병렬성을 높게 함으로써 높은 병렬성을 요구한 사용자의 입출력이 고속으로 수행될 수 있도록 하였다.

사용자 인터페이스는 화일 관련 인터페이스와 프로세스 수행 환경 인터페이스 및 화일시스템 관리 인터페이스로 구분된다.

3.2.1 화일 관련 인터페이스

화일에 관련된 인터페이스는 화일 입출력 인터페이스와 화일 속성 변환 인터페이스로 나누어진다. 화일 입출력 인터페이스는 화일의 생성이나 쓰기 및 읽기 등에 관련된 것들로서 S-CFS에서는 이러한 인터페이스를 기존의 UNIX 화일시스템에서의 시스템 호출과 같은 방식으로 사용할 수 있도록 라이브러리로 제공하고 있다.

〈표 1〉 S-CFS 라이브러리 내의 함수들
〈Table 1〉 Functions of S-CFS library

구분	함수명
사용자 인터페이스 함수	cfs_creat 0, cfs_open (), cfs_close (), cfs_read (), cfs_write (), cfs_fsize (), cfs_lseek () 등
프리미티브 형태의 함수	cfs_namei (), cfs_iget (), cfs_iput (), cfs_get (), cfs_free (), cfs_ifree (), cfs_dbread (), cfs_dbwrite () 등

라이브러리는 사용자가 호출하여 사용할 수 있는 함수들과 호출된 함수 내에서 호출되는 프리미티브 형태의 함수들로 구성되어 있으며, 프리미티브 형태의 함수는 로컬 서버 데몬과 통신하여 입출력을 수행한다. 〈표 1〉은 라이브러리 내의 두 가지 형태에 속하는 함수들의 일부를 보여 준다.

화일 속성 변환 인터페이스로는 chmod, chown 등이 있으며 이들은 화일의 속성을 변경하고자 할 때 사용하는 인터페이스이다. 경로명을 통해 변경시키려는 화일이 S-CFS가 관리하는 화일인지 기존 시스템이 관리하는 화일인지를 구분할 수 있으므로 (그림 4)처럼 시스템이 관리하는 화일은 기존의 명령으로 처리하고 S-CFS가 관리하는 화일은 S-CFS 고유의 명령으로 처리한다.

```
cfs_function_name (path_name, argument)
{
  if (path_name starts with "/cfs")
    executes a function of the CFSD
  else
    system("function_name(path_name,argument)")
}
```

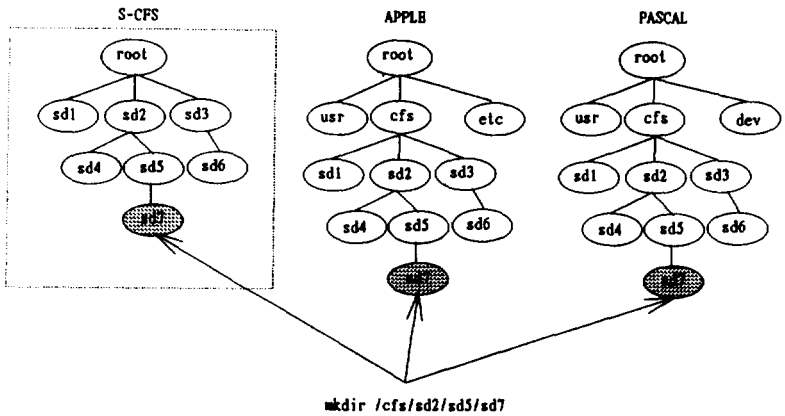
(그림 4) 화일 속성 변환 알고리즘
(Fig. 4) Algorithm for file attribute modification

3.2.2 프로세스 수행 환경 인터페이스

프로세스 수행 환경 인터페이스는 cd, mkdir, rmdir 등과 같이 프로세스가 수행되는 디렉토리의 환경을 변경시키는 명령어들로 구성된다. 본 논문에서는 기존 UNIX 시스템과의 동기적인 처리를 수행함에 따른 오버헤드를 감수하고 사용자에게 완벽한 투명성을 제공하는 방법을 제시한다.

사용자가 어떤 명령을 수행할 때, 그 명령에 의해 접근되어지는 화일은 항상 경로명을 통해 위치를 파악하게 된다. 따라서 S-CFS가 관리하는 화일의 위치를 시스템과 사용자가 똑같이 보도록 만드는 것이 중요하다. 이를 위해서 본 논문에서는 S-CFS의 디렉토리 계층구조가 각 로컬 시스템의 루트 디렉토리에 똑같은 형태로 존재하도록 하는 방식을 사용한다. 즉, mkdir가 수행될 때 경로명이 "/cfs"를 포함하면, (그림 5)에서처럼 병행 화일시스템(S-CFS)의 메타 데이타에 뿐만 아니라 기존의 화일시스템 메타 데이타에도 경로명을 추가하여 경로명에 관한한 S-CFS와 기존의 화일시스템이 동일하게 인식하도록 하여 프로세스 수행 환경 변화에 대해 완전한 접근 투명성을 제공한다. 이런 방식은 디렉토리 생성을 위한 mkdir의 수행시간을 느리게 하며 중복되는 디렉토리 생성에 의한 블럭 할당이라는 오버헤드가 발생하는 문제점이 있다.

그러나 mkdir와 같은 명령어는 화일 입출력에 비교하면 사용 빈도가 극히 적으므로 전체 시스템에 미치는 영향은 매우 미미하다. 디렉토리에 관한 메타 데이타가 두 곳에 있게 되므로 디스크 공간의 문제가



(그림 5) mkdir 명령 수행
(Fig. 5) Command execution of mkdir

발생하나 디렉토리 숫자 또한 화일 숫자에 비하면 무시할 정도이므로 디스크 공간에 대한 낭비는 무시할 수 있다.

따라서 S-CFS는 사용자에게 프로세스 수행 환경의 변화에 대한 완벽한 투명성을 제공할 수 있는 장점이 있다. rmdir는 mkdir과 같은 방식으로 처리를 하고 cd는 기존의 명령을 그대로 사용하면 되므로 디렉토리 이동을 위한 새로운 명령이 필요 없다.

3.2.3 화일 관리 인터페이스

앞에서 다루었던 화일 관련 인터페이스와 프로세스 수행 환경 인터페이스를 제외한 사용자 인터페이스에는 ls, sync, df, makecfs, cp 등이 있다. 여기서는 이러한 인터페이스들을 화일 관리 인터페이스라 한다. 화일 관리 인터페이스는 사용자나 운용자가 화일을 관리할 수 있도록 기본적으로 제공되어야 하는 기능들이다.

S-CFS의 인터페이스 중 라이브러리에서 제공되는 인터페이스들을 제외한 명령어 형태의 인터페이스들은 모두 기존의 UNIX 화일시스템에서 제공하는 인터페이스 명칭으로 aliasing함으로써 사용자에게 접근 투명성을 제공한다. 예를 들면, 기존의 ls 명령을 cfs_ls로 aliasing 시키면 사용자가 ls를 수행했을 때 실제는 cfs_ls가 수행되며 cfs_ls 내부에서 경로명을 통해 접근되는 화일이 S-CFS의 화일인지 기존 시스템의 화일인지를 구분한다.

3.3 S-CFS의 병렬성 제어

VOD 등의 멀티미디어 응용에서는 한 번의 입출력으로 수백 KBytes~수 MBytes 단위의 큰 데이타 접근을 처리한다. 큰 데이타 입출력에서는 대부분이 순차 접근이므로 데이타의 위치를 찾는데 소모되는 탐색시간 및 회전지연시간보다는 데이타 전송에서 소모되는 시간이 대부분을 차지하므로 디스크의 전송시간이 전체 입출력의 성능에서 큰 비중을 차지한다.

하나의 입출력을 다수의 디스크에 분산시켜 처리하기 위하여는 하나의 사용자 프로그램이 여러 개의 서버 데몬들과 통신할 수 있어야 한다. 그러나 UNIX 시스템에서 사용자 프로그램은 동일한 포트번호를 가진 여러 개의 서버 데몬들과 통신할 수 없다. 사용자 프로그램이 여러 개의 서버 데몬들과 통신하기 위하여는 서버 데몬의 포트번호가 달라야 한다. 따라서 사용자에게 최대 p의 병렬성을 제공하기 위하여는 포트번호가 서로 다른 서버 데몬이 p개 있어야 한다.

UNIX 시스템에서 서로 다른 사용자는 동일한 포트번호를 가진 서버 데몬들과 통신할 수 있으므로 동일 포트번호를 가진 서버 데몬이 n개인 경우 n명의 사용자가 동일 포트번호를 가진 서버 데몬들과 통신할 수 있다. 따라서, p의 병렬성을 요구하는 n명의 사용자를 서비스하기 위하여는 $n \times p$개의 서버 데몬이 필요하게 된다.

cfsd 명령어를 수행함으로써 CFSD 서버 데몬을 구동시키며 n명의 사용자 모두에게 최대 p의 병렬성을

제공하고자 하는 경우에는 (그림 6)과 같이 명령어를 수행하면 된다.

(그림 6)에서 cfsd는 CFSD 서버 데몬의 실행모듈이고 0,1,...,p-1은 논리 포트번호를 의미하며 n는 동일한 포트번호를 가지는 서버 데몬의 수를 의미한다. cfsd는 물리적인 포트번호를 2011번부터 사용하므로 논리적인 포트번호를 p-1로 하면 물리적인 포트번호는 2011 + p − 1이 된다.

```
$ cfsd  0  n ↵
$ cfsd  1  n ↵
        .
        .
        .
$ cfsd  p-1  n ↵
```

(그림 6) CFSD의 서버 데몬 구동
(Fig. 6) Start-up of CFSD server daemon

S-CFS에서는 화일 입출력의 병렬성을 얻기 위하여 다수의 데몬 프로세스가 한 사용자의 입출력을 서비스 할 수 있도록 하여서 화일 입출력의 병렬성을 제어한다.

4. S-CFS의 성능 분석

본 장에서는 성능 분석 환경에 대해 기술한 후 트랜잭션 처리 및 대용량 데이타 처리에 대해 각각의 성능을 분석한다.

4.1 성능 분석 환경

S-CFS는 널리 사용되고 있는 범용의 네트워크 환경에서 다수의 호스트에 있는 디스크 자원들을 사용자가 디스크 배열처럼 사용할 수 있도록 지원함을 목표로 한다. 이러한 S-CFS의 성능을 평가하기 위해 UNIX 운영체제를 탑재한 다수의 워크스테이션들을 로컬 네트워크로 연결하여 Workstation Cluster를 구성함으로써 실험 환경을 구축하였다.

4.1.1 실험 환경

성능 분석을 위해 사용한 시스템의 구성요소들과 각각의 구체적인 사양은 다음과 같다.

1) Workstation Cluster의 구성 및 사양

S-CFS를 구동하기 위해 모두 4대의 워크스테이션을 사용하였으며 4대 중 1대는 SPARC-20이고 3대는 SPARC-10이다. Workstation 본체에 포함된 주요 구성요소로는 중앙처리장치(CPU), 주기억장치(Main Memory), CPU 캐쉬 등을 들 수 있으며 각 요소의 구체적인 내용은 〈표 2〉와 같다.

〈표 2〉 Workstation의 구성요소별 사양
〈Table 2〉 Specification of Workstations

종류	SPARC-10	SPARC-20
CPU Clock	50 MHz	60 MHz
Main Memory	32 MBytes	32 MBytes
운영체제	Solaris 1.1	Solaris 1.1
On-chip 캐쉬	36 KBytes	36 KBytes
External 캐쉬	1 MBytes	1 MBytes

2) 디스크 서브시스템의 구성 및 사양

SPARC-20와 SPARC-10 각각에 2~3개의 실험용 디스크를 추가로 장착하여 S-CFS 구동 환경을 구축하였다.

S-CFS의 성능 평가를 위하여는 다수의 디스크를 사용하여야 하며 디스크의 사양은 벤더에 따라 커다란 차이가 있으므로 동일 벤더의 전통적인 디스크들을 사용하였다. 성능 평가를 위하여 사용된 디스크는

〈표 3〉 디스크 서브시스템의 사양
〈Table 3〉 Specification of the disk subsystem

항목(Item)	값(Values)
Random Average (read)	11.5ms
Random Average (write)	13ms
Average Rotational Latency	6.67ms
디스크 드라이브의 전송 속도	4 MBytes/sec
SCSI 버스 전송 속도	10 MBytes/sec

540 MBytes Quantum 디스크 12대이며 디스크의 사양과 입출력 버스(SCSI-BUS)의 사양은 〈표 3〉과 같다.

3) 네트워크의 구성 및 대역폭 조정

4개의 워크스테이션을 연결하기 위하여 사용한 네트워크 장비는 기존 시스템에 설치되어 있던 Ethernet 장비이다. 한번에 대용량의 데이타를 처리하기 위하여는 FDDI, ATM 등과 같은 높은 대역폭의 네트워크 장비가 요구되나 본 연구에서는 높은 대역폭의 네트워크 장비를 별도로 사용하지 않고 네트워크 장비의 대역폭이 FDDI나 ATM 등과 같이 높은 대역폭을 가진다고 가정하였다.

FDDI의 전송 대역폭을 가정하기 위해 FDDI의 전송 대역폭이 Ethernet의 전송 대역폭의 10배이므로 화일 입출력시 네트워크 장비를 거쳐 데이타를 전송할 때 데이타 크기를 1/10로 줄여 전송하는 방법을 사용하였다. 마찬가지로 ATM의 경우 전송속도가 622.08 Mbits/sec이므로 ATM 네트워크를 가정하기 위하여 네트워크를 통한 데이타 전송시 데이타의 크기를 1/62로 줄여 전송하였다.

4.1.2 화일시스템의 구성 및 화일 생성

실측으로 성능을 분석하기 위하여 S-CFS 자체의 화일시스템을 생성하고 flushd와 cfsd 데몬 프로세스들을 구동시켜 S-CFS가 사용자 요구를 서비스할 수 있도록 한 후 사용자가 화일 생성 및 화일의 읽기와 쓰기를 수행하도록 하였다.

본 논문에서는 트랜잭션 처리에 대한 실험을 수행하기 위하여 20 MBytes 화일 24개를 생성하였고, 대용량 데이타에 대한 실험을 수행하기 위하여 30 MBytes 화일 24개를 생성하였다. 화일 생성을 위하여 S-CFS에서 제공하는 프리미티브인 cfs_creat와 cfs_write 프리미티브를 사용하였다.

4.2 트랜잭션 처리에 대한 성능 분석

트랜잭션 응용은 화일에 대해 작은 쓰기와 작은 읽기를 요구하는 응용이므로 일정한 디스크 수에 대하여 화일 입출력을 동시에 요구하는 사용자 수의 변화에 따른 단위 시간당 트랜잭션 처리 수를 계산하였다. 트랜잭션 처리에 대한 성능은 실측 결과로 분석하였다.

4.2.1 실측 방법

화일 입출력은 화일 읽기와 쓰기로 구분되며 대개의 경우 두 가지 모두 유사한 접근 형태를 나타내지만 디스크 특성에 따라 서로 다른 형태를 나타내는 경우도 있다. 본 연구에서는 일반적인 디스크 입출력의 성능 분석에 초점을 두고 있으므로 실험의 단순화를 위해 화일 읽기에 대한 실험만을 수행하였다.

화일 읽기의 실험은 4.1.2절에서 언급된 화일들에 대하여 서로 다른 사용자가 서로 다른 화일을 랜덤하게 주어진 회수(1000회)만큼 입출력을 수행하도록 하여 단일 입출력에 대한 평균반응시간을 구한 후 사용자 수와 디스크 수에 따른 단위시간당 트랜잭션 처리 수를 계산하였다. 여기서 평균반응시간은 한 사용자의 입출력 중 앞의 작업이 끝난 후 다음 작업이 시작해서 마칠 때까지의 평균시간을 의미한다.

[정의 1] Workstation Cluster의 전체 디스크의 수는 $d(wd_1, wd_2,...,wd_n)$의 형태로 정의하며 wd_1은 첫 번째 Workstation의 디스크 수를 의미하고 wd_n은 n번째 Workstation의 디스크 수를 의미한다.

4.2.2 실측 결과 분석

본 절에서는 트랜잭션 크기가 4 KBytes, 0.5 KBytes인 경우에 대하여 성능을 분석하고 트랜잭션 처리시 병목현상을 일으키는 시스템 구성요소를 분석한다.

1) 트랜잭션 크기가 4 KBytes인 경우

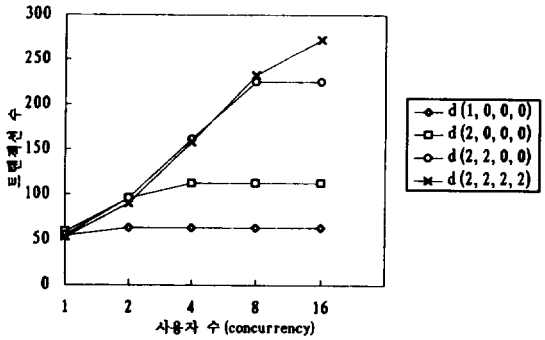
(그림 7)은 디스크 수에 따른 트랜잭션 처리의 변화를 나타낸다. 디스크 수가 8개인 경우, 사용자 수가 8명이 될 때까지는 초당 트랜잭션의 처리 수는 선형적으로 증가하나 사용자 수가 8보다 큰 경우에는 트랜잭션 처리 수의 증가율이 둔화됨을 보여 준다.

디스크의 수가 8개일 때 디스크 수에 따른 처리율의 병렬성이 둔화된 것은 네트워크에 병목현상이 발생했기 때문으로 분석되었다. 실험 환경에서 사용한 네트워크 장비는 Ethernet이며 최대 전송 속도는 10 Mbits/sec(1.25 MBytes/sec)의 전송율을 가진다. 디스크의 개수가 8개이고 사용자 수가 16명인 경우 초당 272개의 트랜잭션을 처리했으므로 네트워크는 초당 1.09 MBytes의 데이타를 전송한 결과가 된다.

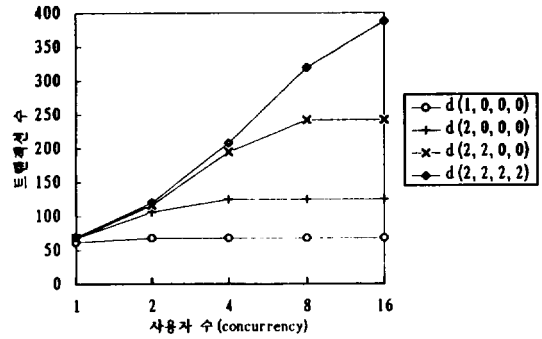
$$bandwidth = 272 transactions/\sec \times 4\ KBytes/transaction$$
$$= 1.09\ MBytes/\sec$$

이 경우 네트워크의 점유율은 다음 식에서와 같이 87.2%로서 네트워크의 최대 전송 속도에 도달하여 네트워크에 병목현상이 발생했음을 알 수 있다.

$$\text{네트워크 점유율} = \frac{1.09\ MBytes/\sec}{1.25\ MBytes/\sec} = 87.2\%$$



(그림7) 디스크 수의 증가에 따른 단위시간당 트랜잭션 처리 수(4 KBytes)
(Fig. 7) The number of transactions per sec according to the number of disks (4 KBytes)



(그림 8) 디스크 수의 증가에 따른 단위시간당 트랜잭션 처리 수(0.5 KBytes)
(Fig. 8) The number of transactions per sec according to the number of disks (0.5 KBytes)

2)트랜잭션 크기가 0.5 KBytes인 경우

(그림 8)은 (그림 7)에서 각 트랜잭션의 크기를 1/8로 줄인 것으로서 트랜잭션의 크기가 작기 때문에 디스크가 8대인 경우에도 네트워크에 병목현상이 발생하지 않는다. 따라서 이 경우에는 CPU의 병목현상이 발생할 때까지 디스크 수의 증가에 따라 단위시간당 트랜잭션의 처리 수가 계속 증가한다.

디스크 수가 8대이고 사용자의 수가 8명일 때까지는 트랜잭션의 처리율이 계속해서 선형적으로 증가하나 사용자가 16명일 때는 트랜잭션 처리율의 증가가 둔화된다. 이것은 계산 능력의 병목현상에 의해 처리율의 증가율이 제한받고 있음을 보여 준다. 따라서 이 상태에서는 디스크 수와 네트워크의 대역폭을 증가시켜도 더 이상 초당 평균 트랜잭션 처리 수가 증가될 수 없음을 보여준다. 따라서 더 높은 트랜잭션의 처리율을 얻기 위하여는 디스크 수와 더불어 CPU의 성능을 향상시켜야 한다.

4.3 대용량 데이타의 성능 분석

트랜잭션 응용의 경우 한 번에 입출력하는 데이타의 양이 소량인 반면 멀티미디어나 과학계산 응용의 경우 한 번에 입출력하는 데이타의 크기가 매우 크다. 이 중 과학계산 응용의 특징은 한 번에 입출력하는 데이타의 양이 100 KBytes에서 1.5 MBytes 정도로 데이타의 크기가 매우 크며 접근 형태가 순차성을 가진다[12].

그리고 멀티미디어 데이타의 특징은 한 번에 입출력하는 데이타의 요구량이 비교적 크지만 시스템의 특성에 따라서 한 번에 입출력하는 데이타의 크기를 조정할 수 있다. 따라서 본 논문에서는 멀티미디어 응용과 과학계산 응용에 대해 별도의 성능 평가 방법을 적용하지 않고 두 경우 모두를 대용량 데이타의 입출력으로 간주하여 성능을 평가한다.

4.3.1 실측 방법

대용량 데이타 입출력시 시스템의 단위시간당 처리율을 측정하기 위해 한 번에 요청하는 데이타 크기, 스트라이핑 단위, 단일 사용자에게 주어지는 병렬성에 따른 시스템의 성능을 분석한다. 요구 데이타의 특성은 정의 2로 나타낸다.

[정의 2] 화일 입출력 요구의 특성을 나타내기 위한 기호는 r(rs, su, rp)로 나타내며, rs는 요청 데이타 크

기(request size)를 의미하고, su는 스트라이핑 단위(striping unit)를 의미하며, rp는 사용자가 요구하는 병렬성을 의미한다. rp가 s라 함은 단일 사용자를 위해 입출력을 수행해 줄 서버 데몬의 수가 s임을 의미한다.

이상의 기호 정의에 따라 다양한 입출력 요구의 특성에 대한 실험을 수행하고 rs, su, rp의 특성과 사용자 수에 따른 성능을 분석한다. 실험시에 디스크의 개수는 8개로 고정하였다.

4.3.2 성능 분석 결과

성능 분석은 요구 데이타의 크기가 64 KBytes, 256 KBytes, 1024 KBytes의 세 가지 경우로 나누어 분석한다.

1) 요구 데이타의 크기가 64 KBytes인 경우

(그림 9)에서 보는 것처럼, 화일시스템의 최대 처리율은 사용자가 16명일 경우이며 최대 처리율은 스트라이핑 단위에 크게 의존함을 알 수 있다.

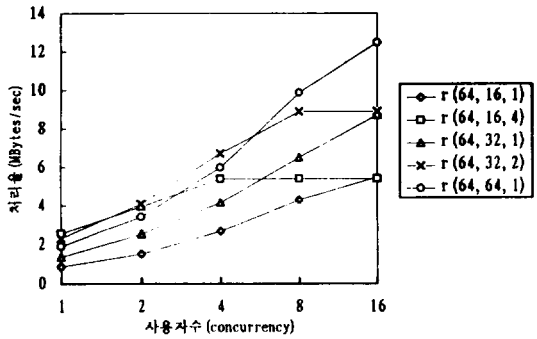
스트라이핑 단위가 64 KBytes인 경우 화일시스템의 최대 처리율은 12.6 MBytes/sec로 나타난 반면 스트라이핑 단위가 32 KBytes일 경우에는 9 MBytes/sec 이하로 나타나서 64 KBytes인 경우의 75% 수준이며, 16 KBytes일 경우에는 6 MBytes/sec 이하로 64 KBytes인 경우의 50% 수준에 그쳐 화일시스템의 단위시간당 처리율은 스트라이핑 단위에 크게 영향을 받는 것으로 나타났다.

스트라이핑 단위가 작을 경우 전체 화일시스템의 처리율이 낮게 나타나는 것은 전체 입출력시간 중 탐색시간이나 회전지연시간과 같은 기계적 특성이 차지하는 비율이 크기 때문으로 분석된다. 스트라이핑 단위가 커짐에 따라 탐색시간과 회전지연시간의 합이 차지하는 비율이 작아지므로 화일시스템 전체의 처리율은 일정 수준까지는 계속하여 증가하나 스트라이핑 단위가 계속 증가하면 버퍼 간의 데이타 이동시간에 의한 수행시간 오버헤드때문에 화일시스템의 최대 처리율은 오히려 감소된다. 이것에 대한 구체적인 내용은 요구 데이타 크기가 256 KBytes인 경우의 분석에서 밝혀진다.

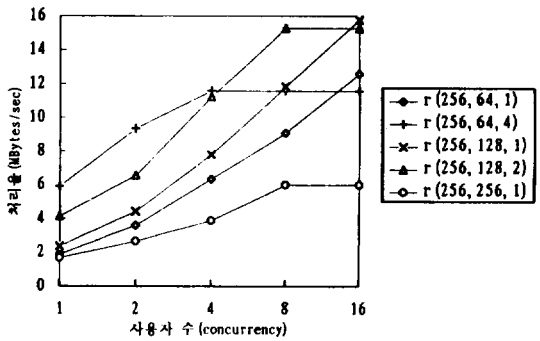
2) 요구 데이타 크기가 256 KBytes인 경우

(그림 9)에서 보듯이 요구 데이타 크기가 64 KBytes (rs = 64 KBytes, su = 64 KBytes)인 경우에 시스템의 최대 처리율이 12.6 MBytes/sec로 나타난 반면, (그림 10)에서 요구 데이타 크기가 256 KBytes(rs = 256 KBytes, su = 128 KBytes)인 경우에 시스템의 최대 처리율이 15.8 MBytes로서 최대 처리율은 25%나 증가했다. 그러나 스트라이핑 단위가 256 KBytes인 경우 버퍼간 데이타 복사에 의한 오버헤드 때문에 스트라이핑 단위가 64 KBytes인 경우보다도 시스템의 처리율이 낮아졌다. 따라서 시스템의 최대 처리율은 스트라이핑 단위가 128 KBytes인 경우로 분석되었다.

단일 사용자를 위한 화일시스템의 최대 병렬성을 비교하기 위하여 요구 병렬성이 1이면서 최대 처리율을 가지는 경우(su = 128 KBytes, rp = 1)와 요구 병렬



(그림 9) rs = 64 KBytes인 경우의 단위시간당 처리율
(Fig. 9) Throughput per time unit with rs = 64KBytes

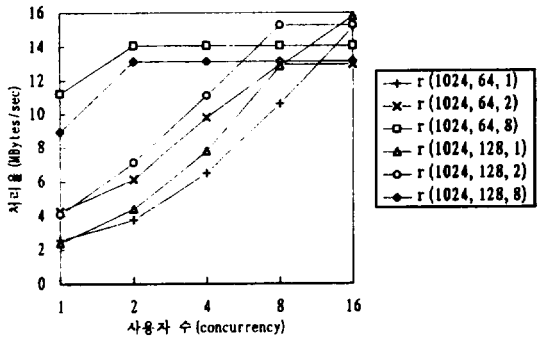


(그림 10) rs = 256 KBytes인 경우의 단위시간당 처리율
(Fig. 10) Throughput per time unit with rs = 256 KBytes

성이 4인 경우(su=64 KBytes, rp=4)의 처리율을 비교하였다. 요구 병렬성이 4일 경우에 화일시스템의 처리율은 5.9 Mbytes/sec이고 요구 병렬성이 1인 경우에는 2.3 MBytes/sec이므로 물리적인 병렬성은 2.6이 된다.

3) 요구 데이타 크기가 1024 KBytes인 경우

(그림 10)과 (그림 11)를 비교하여 보면 요구 데이타의 크기가 256 KBytes인 경우와 1024 KBytes인 경우에 화일시스템의 처리율은 비슷하게 나타났다. 이는 요구 데이타의 크기가 256 KBytes 이상인 경우에는 요구 데이타의 크기와 관계 없이 스트라이핑 단위에 의존하여 시스템의 최대 처리율이 결정됨을 나타낸다.



(그림 11) rs = 1024 KBytes인 경우의 단위시간당 처리율
(Fig. 11) Throughput per time unit with rs = 1024KBytes

그러나, 과학계산 응용의 경우에는 주로 소수의 사용자가 고속 입출력을 요구하는 경우가 많으므로 요청 데이타의 크기를 크게 하여 화일 입출력의 병렬성을 높이는 것이 필요하다.

단일 사용자 환경에서 요구 병렬성이 1(su=64 KBytes, rp=1)인 경우 화일시스템의 단위시간당 처리율은 2.6 MBytes/sec인데 비해, 요구 병렬성이 8(su=64 KBytes, rp=8)인 경우 화일시스템의 단위시간당 처리율은 11.2 MBytes/sec로서 실질적인 병렬성은 4.3배 증가하였다.

이것은 요구 데이타 크기가 256 KBytes(rs=256 KBytes, su=64 KBytes, rp=4)일 경우(그림 10 참조)의 최대 병렬성이 2.6인 것과 비교하면 단일 사용자에 대한 병렬성은 크게 증가한 것이다.

4.4 다른 연구와의 성능 비교

기존 병렬 화일시스템과 본 논문의 S-CFS를 최대 대역폭 및 단위시간당 트랜잭션 처리 수를 중심으로 비교한다. 비교 대상 화일시스템들 중 sfs와 RAID-Ⅱ는 부가적인 H/W 보드를 포함하고 있으며 Vesta, Intel의 CFS, S-CFS는 순수 S/W로 구성되어 있다. 각각의 성능을 비교하면 〈표 4〉와 같다.

Compute node에서 일정 수의 사용자가 입출력을 발생시켰을 때에 일정 시간 동안 시스템 내에서 모든 사용자에게 주어진 대역폭을 모두 합한 값을 누적 대역폭(aggregate bandwidth)이라 한다[7]. 〈표 4〉의 최대 대역폭은 누적 대역폭 중 가장 큰 값을 의미한다.

Intel의 CFS는 I/O의 노드 수가 8이고 I/O 노드당 1대의 디스크를 장착하였을 때 최대 대역폭이 6~7 MBytes/sec로 나타나서 다른 화일시스템에 비해 입출력 대역폭이 크게 낮다. 그 이유는 Intel CFS에서는 블럭의 크기를 16 KBytes 이하로 작게하여서 디스크의 탐색시간과 회전지연시간이 차지하는 비율이 크기 때문이다.

sfs는 디스크가 8대일 때 12 MBytes/sec의 대역폭을 제공하는 것으로 나타나서 본 연구와 비교할 때 디스크당 평균 대역폭은 낮은 것으로 분석된다. RAID-Ⅱ 화일서버는 디스크가 24개일 때 20 MBytes/sec의 대역폭을 제공하는 것으로 나타나서 본 연구 결과와 비교할 때 디스크당 평균 대역폭은 낮은 것으로 분석된다. 단위시간당 트랜잭션 처리 수는 15개 디스크에서 424트랜잭션을 나타내므로 S-CFS의 디스크당 트랜잭션 처리 수보다 낮다. Vesta 화일시스템은 9개의 디스크에서 최대 대역폭이 20 MBytes/sec이므로 8개의 디스크에서 15.8 MBytes인 S-CFS보다 디스크당 대역폭이 높다.

S-CFS는 4.2절과 4.3절에서 분석한 성능 결과이다. S-CFS는 Intel의 CFS와 비교하면 적은 노드 수로 높은 대역폭을 제공한다는 장점이 있으며, sfs와 RAID-Ⅱ에 비해서는 별도의 H/W 설계 없이 기존의 H/W 사양만으로도 높은 대역폭을 제공한다는 장점이 있다. S-CFS는 Vesta 화일시스템에 비해 디스크 수에 따른 최대 대역폭은 낮다. 그러나 Vesta는 노드당 1대의 디

〈표 4〉 화일시스템의 성능 비교

〈Table 4〉 Performance comparison among file systems

화일시스템 명칭	부가적인 H/W	최대 대역폭 및 트랜잭션 처리 수
CFS	없음	6~7 MBytes/sec (8 compute nodes, 8 I/O nodes, 8 disks)
sfs	DSN Board SDA Controller	12 MBytes/sec (8disks) 18 MBytes/sec (12disks)
RAID-Ⅱ	X-Bus Board Cougar Controller	20 MBytes/sec (24disks) 424 transactions/sec (15disks)
Vesta	없음	약 20 MBytes/sec (3 compute nodes, 9 I/O nodes, 9 disks)
S-CFS	없음	15.8 MBytes/sec (4 compute node & I/O node, 8 disks) 388 transactios/sec (4 compute node & I/O node, 8 disks)

스크를 장착하였고 S-CFS는 노드당 2대의 디스크를 사용하였으므로 S-CFS는 Vesta에 비해 시스템 자원의 사용이 효율적이다.

그러므로 본 연구에서 구현한 S-CFS는 성능과 시스템 자원의 사용면에서 매우 효율적인 병렬 화일시스템이다.

5. 결 론

최근 멀티미디어 데이타와 같은 대용량 데이타에 대한 수요가 급증함에 따라 소규모의 서버 시스템에서도 고성능 입출력 시스템을 필요로 하게 되었다. 현재까지의 병렬 화일시스템에 관한 연구는 MPP와 같은 대규모의 고성능 시스템을 중심으로 연구되어 왔으나 성능 분석에 관한 체계적인 연구는 미흡하였다.

고성능 컴퓨터 시스템에서는 입출력 향상을 위한 병렬 화일시스템의 오버헤드가 문제되지 않지만 소규모 서버로 사용되는 워크스테이션의 경우에는 병렬 화일시스템의 오버헤드가 높은 비중을 차지함이 본 연구 결과에서 밝혀졌다.

트랜잭션처리에 관한 성능 분석 결과, 입출력을 동시에 수행하는 사용자의 수가 16명 이상인 경우에는 단위시간당 처리되는 트랜잭션 처리 수가 8대의 디스크까지는 증가하였지만 8대 이상에서는 더 이상 증가하지 않았다. 그러므로 단위시간당 트랜잭션 처리 수를 증가시키기 위해서는 디스크 개수(8개 이상)의 증가와 더불어 계산 능력이 높은 CPU를 선택하여야 한다.

대규모 데이타를 입출력하는 경우, 화일시스템 전체의 성능이 메모리 복사의 오버헤드로 인해 제한을 받으므로 본 연구에서는 메모리 복사의 오버헤드를 최소로 하기 위하여 사용자와 병렬 화일시스템이 메모리를 공유하도록 하였다.

공유 메모리를 사용하는 경우, 8대의 디스크를 사용하면 최대 15.8 MBytes/sec의 처리율을 보이므로 대규모 컴퓨터 시스템이나 별도의 H/W 보드를 설계하여 구축한 화일시스템보다 우수한 성능을 보였다.

멀티미디어 서비스의 경우에는, 디스크 입출력, 데이타 압축 및 동기화와 같은 제반 사항을 고려하여야 하므로 향후에는 이와 같은 요소들을 추가로 연구하여 구현된 병행 화일시스템을 멀티미디어 서버로 사용될 수 있도록 확장할 예정이다.

참 고 문 헌

[1] J. Ousterhout, F. Douglis, "Beating the I/O Bottleneck: A Case for Log-Structured File Systems," Operating Systems Review, Vol. 23, No. 1, pp. 11-28, Jan. 1989.

[2] J. Hennessy and D. Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufmann

Publishers, 1990.

[3] M. Livny, S. Khoshafian, H. Boral, "Multi-Disk Management Algorithms," Proceedings of the 1987 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pp 69-77.

[4] D. A. Patterson, P. Chen, G. Gibson, R. H. Katz, "Introduction to Redundant Arrays of Inexpensive Disks," Proceedings of IEEE COMPCON, pp. 112-117, Spr. 1989.

[5] M. Y. Kim, and A. N. Tantawi, "Asynchronous Disk Interleaving: Approximating Access Delays," IEEE Transactions on Computers, Vol. 40, No. 7, July 1991.

[6] R. H. Katz, G. A. Gibson, D. A. Patterson, "Disk System Architectures for High Performance Computing", Proceedings of the IEEE, Vol. 77, No. 12, Dec. 1989.

[7] D. G. Feitelson, P. F. Corbett, J. P. Prost, "Performance of the Vesta Parallel File System," Proceedings of 9th International Parallel Processing Symposium, pp. 150-158, 1995.

[8] P. Pierce, "A Concurrent File System for a Highly Parallel Mass Storage Subsystem", 4th Conference on Hypercubes, Concurrent Computers and Applications, Monterey, CA, pp. 155-160, March 1989.

[9] Peter F. C., "Overview of the Vesta Parallel File System," Computer Architecture News, Vol. 21, No. 5, pp. 7-14, 1993.

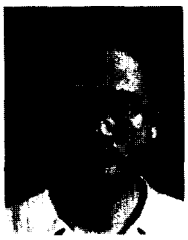
[10] Susan J. L., Marshall Isman, Andy Nanopoulos, "sfs: A Parallel File System for the CM-5," Proceedings of 1993 Summer USENIX, pp. 291-304, 1993.

[11] A. L. Drapeau, et. al, "RAID-Ⅱ: High-bandwidth Network File Server," Proceedings of 21st Annual International Symposium on Computer Architecture, pp. 234-244, 1994.

[12] P. M. Chen, D. A. Patterson, "Maximizing Performance in a Striped Disk Array," The 17th Annual International Symposium on Computer Architecture, pp. 322-331, May. 1990.

장 시 웅

1984년 부산대학교 계산통계학과 졸업(학사)
1993년 부산대학교 대학원 전자계산학과 졸업(이학석사)
1996년 부산대학교 대학원 전자계산학과 졸업(이학박사)
1986년~1993년 대우통신종합연구소 주임연구원
1996년~현재 동의대학교 전산통계학과 전임강사
관심분야: 병렬 화일시스템, 멀티미디어



정 기 동

1973년 서울대학교 졸업(학사)
1975년 서울대학교 대학원 졸업(석사)
1986년 서울대학교 대학원 계산통계학과 졸업(박사)
1990년~1991년 MIT, South Carolina 대학 교환 교수
1978년~현재 부산대학교 전자계산학과 교수
1995년~현재 부산대학교 전자계산소장
관심분야: 병렬처리, 멀티미디어