클러스터 컴퓨터를 위한 단일 I/O 공간 서비스의 구현 및 성능분석

김 태 규[†] · 김 방 현[†] · 김 종 현[††]

요 약

클러스터 컴퓨터에 있어서 I/O 중심적인 응용을 효과적으로 처리하기 위해서는 통합 I/O 하부 구조를 지원하는 단일 I/O 공간(SIOS)이 필수적으로 구현되어야 한다. SIOS 서비스는 클러스터 컴퓨터 내의 어느 노드에서든지 자신 혹은 원격 노드에 위치한 주변기기 및 하드 디스크들을 직접 액세스할 수 있도록 전역 I/O 주소 공간을 구축해준다. 본 논문에서는 리눅스 클러스터에서 프리웨어들만을 이용하여 SIOS를 구현하는 방법을 제안하였다. 이 방법은 ENBD를 이용한 디바이스 드라이버 레벨과 S/W RAID 및 NFS를 이용한 파일 시스템 레벨에서 구현되었다. 이 방법의 주요 장점은 프리웨어들만을 이용하기 때문에 구현이 용이하고 비용이 거의 들지 않는다는 것이다. 또한 본 연구에서 사용한 프리웨어들은 공개 소스이기 때문에 다른 플랫폼에서도 약간의 수정을 통하여 적용이 가능하다는 장점이 있다. 이러한 장점을 가지면서도 실험 결과에서 나타난 I/O 처리율은 커널 수준에서 별도로 개발된 디바이스 드라이버를 사용하는 CDD보다 쓰기 동작에서는 최대 5.5배, 읽기 동작에서는 2.3배정도 더 높게 나타났다.

키워드 : SIOS, 클러스터 컴퓨터, 리눅스 프리웨어, ENBD, S/W RAID, NFS

Implementation and Performance Analysis of Single I/O Space Service for Cluster Computers

Tae-Kyu Kim[†] · Bang-Hyun Kim[†] · Jong-Hyun Kim[††]

ABSTRACT

In cluster computers, it is essential to implement the single I/O space(SIOS) supporting integrated I/O substructure to efficiently process I/O intensive applications. SIOS service provides with global I/O address space to directly access peripherals and hard disks in its own or remote nodes from any node in the cluster computer. In this thesis, we propose the implementation method of SIOS in Linux clusters by using only freewares. This method is implemented at device driver level that uses Enhanced Network Block Device(ENBD) and file system level that uses S/W RAID and NFS. The major strengths of this method are easiness of implementation and almost no cost due to using freewares. In addition, since freewares used are open sources, it is possible to apply this method to other platforms with only slight modification. Moreover, experiments show that I/O throughputs are up to 5.5 times higher in write operations and approximately 2.3 times higher in read operations than those of CDD method that uses the device driver developed at kernel level.

Key Words : Single I/O Space, Cluster Computer, Linux Freeware, ENBD, S/W RAID, NFS

1. 서 론

클러스터 컴퓨터는 저렴한 비용으로 고성능의 병렬컴퓨팅 환경을 구현할 수 있다는 점에서 그 잠재력이 입증되어 왔다. 여러 개의 컴퓨터들을 상호 연결하여 구성되는 클러스터 시스템이 하나의 통합된 자원으로서 사용될 수 있다면, 그 시스템은 단일 시스템 이미지(single system image: SSI)를 가지고 있다고 말한다. SSI 서비스는 성능, 편의성, 확장성, 그리고 신뢰성 측면에서 클러스터 컴퓨터의 이용률을 향상시킬 수 있는 주요 방법이다[1, 2]. 특히 I/O 중심의 응용을 고속으로 처리하기 위해서는 단일 I/O 공간(single I/O space: SIOS)을 구축하는 것이 필요하다. SIOS는 하나의 클러스터 내에서 모든 데이터 블록(data block)에 대한 단일 주소 공간(single address space)을 제공한다. 이것이 의미하는 것은 사용자가 어떤 파일을 사용하거나 데이터 블록을 참조할 때, 그것이 저장되어 있는 디스크의 물리적인

※ 이 논문은 2004년도 연세학술연구비의 지원에 의해 이루어졌음.
† 준 회 원 : 연세대학교 컴퓨터정보통신공학부 박사과정
†† 정 회 원 : 연세대학교 컴퓨터정보통신공학부 교수
논문접수 : 2006년 5월 13일, 심사완료 : 2006년 9월 18일

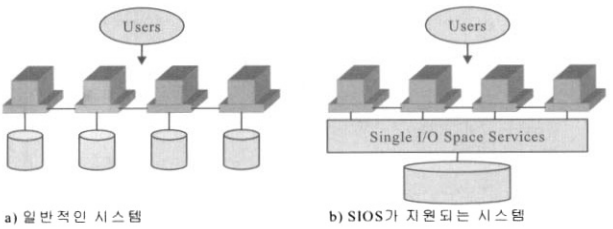
위치에 상관없이 사용자가 파일이나 데이터 블록을 사용할 수 있음을 말한다. 즉, SIOS 프레임워크의 구축은 SSI 서비스를 제공하는 클러스터 환경에 있어서 I/O 중심의 모든 응용들을 효과적으로 지원하는 기반 구조가 된다[3].

따라서 본 연구에서는 일반 사용자가 저비용으로 SIOS를 쉽게 구현할 수 있도록 하기 위하여, 리눅스 운영체제 환경에서 프리웨어(freeware)들만을 이용하여 SIOS를 구현할 수 있는 방법을 제안한다. 본 연구에서 제안한 방법은 리눅스 운영체제를 사용하는 클러스터 컴퓨터에 즉시 적용이 가능하다. 그리고 본 연구에서 사용하는 프리웨어들은 ANSI C로 만들어진 공개 소스(open source)이기 때문에 다른 유닉스 운영체제를 사용하는 시스템의 경우에도 약간의 수정 작업만으로 쉽게 적용할 수 있으며, 최근 활발하게 진행되는 그리드 컴퓨터 연구에도 활용될 수 있다. 또한, 제안된 방법에 대하여 다양한 측면의 성능 평가 결과들을 제시함으로써 사용자가 본 연구 방법을 적용할 경우에 기술 데이터로 이용이 가능하도록 하였다.

2. 관련 연구

SIOS는 클러스터 시스템에서 개별 노드에 장착된 I/O 장치 혹은 저장 장치의 물리적인 위치를 알지 못하는 상태에서도 어떤 노드에서든 접근 가능하게 하는 것으로서 SSI의 기본 목적을 달성하기 위한 필수적인 기능이다. 디스크의 경우, SIOS는 사용자의 관점에서 (그림 1)과 같은 SSI 서비스를 제공함으로써 지역 디스크(local disk)와 원격 디스크(remote disk)에 대한 접근의 차이를 제거하며, 디스크에 저장될 데이터를 클러스터 시스템의 각 노드로 분산시킴으로써 노드 간의 데이터 이동을 위한 I/O 연산의 성능을 향상시킨다[4].

SIOS를 지원하려는 시도들을 접근 방식에 따라 분류하면 사용자 레벨(user level)의 구현 방법과 파일 시스템 레벨(file system level)의 구현 방법, 그리고 디바이스 드라이버 레벨(device driver level)의 구현 방법으로 구분할 수 있다[5]. 사용자 레벨의 구현 방법으로는 Clemson 대학의 PVFS(parallel virtual file system)[6]와 Argonne 국립 연구소의 RIO(remote I/O) 프로젝트[7]가 있고, 파일 시스템 레벨의 구현 방법으로는 UC Berkeley 대학의 xFS(serverless network file system)[8]와 Sun Microsystems 연구소의 Solaris MC 프로젝트[9]가 있으며, 디바이스 드라이버 레벨의 구현 방법으로는 DEC 사의 Petal 프로젝트[10]와 USC 대학의 CDD(cooperative disk driver)[11]가 있다.



(그림 1) SIOS 서비스의 개념

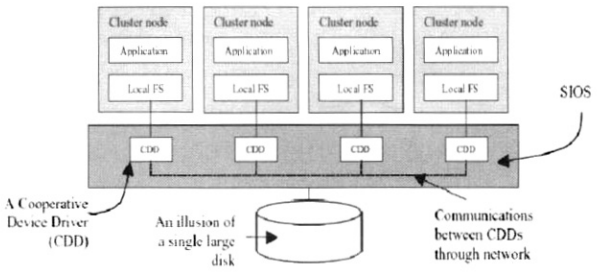
사용자 레벨의 구현 방법은 다른 방법들에 비하여 상대적으로 적은 비용으로 구현이 가능하며, 구현된 시스템의 이식성이 높다는 장점이 있다. 그러나 이 방법은 패키지의 모든 기능을 이용하기 위해서 사용자가 특수한 API(application programming interface)들이나 식별자들을 사용해야 하기 때문에 완전한 SIOS를 제공하지 못하게 되고, 네트워크나 파일 시스템 서비스를 위하여 시스템 호출을 사용하기 때문에 성능에 한계가 있다는 문제점이 있다.

파일 시스템 레벨의 구현 방법은 사용자가 원격 데이터(remote data)를 지역 데이터(local data)처럼 액세스할 수 있는 완전한 SIOS를 제공해 준다. 이 경우에 파일 시스템은 원격 디스크를 액세스할 수 있도록 특별하게 설계되기 때문에 성능 최적화가 가능하고, 또한 이를 위하여 분산 방식을 조정하는 것이 가능하다는 장점이 있다. 그러나 새로운 분산 파일 시스템을 설계 및 구현하기 위해서는 비용이 많이 들고, 파일 시스템을 변경하는 것은 이전의 응용 프로그램에 대하여 완전한 호환성을 보장하지 못한다는 문제점이 있다.

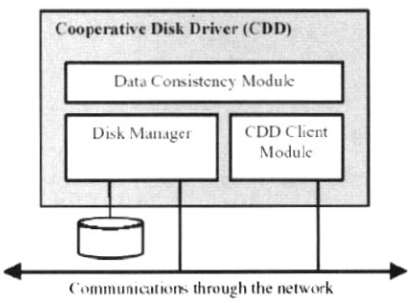
디바이스 드라이버 레벨의 구현 방법은 사용자뿐만 아니라 파일 시스템에게도 SIOS 서비스를 제공한다. 이것은 디스크 I/O를 위한 SIOS 서비스를 파일 시스템의 수정 없이 제공할 수 있다는 것을 의미한다. 구현 방법은 파일 시스템 레벨의 구현에 비하여 비용이 적게 들고, 파일 시스템을 변경하지 않기 때문에 이전의 응용 프로그램에 대한 호환성이 높다는 장점이 있다. 그러나 이 방법의 가장 큰 문제점은 성능 최적화를 위하여 파일의 분산방식을 조정하는 것이 어렵다는 것이다[5].

디바이스 드라이버 레벨로 SIOS를 구현한 Petal 프로젝트는 네트워크로 연결된 저장장치 서버들로 구성된 분산 저장장치 시스템을 제공한다. 이 프로젝트는 클러스터 환경에서 전역 이름 공간을 사용하여 공유 가상 디스크 배열의 개념을 처음으로 구현한 ds-RAID(distributed software RAID)이다. Petal 프로젝트는 SIOS를 구현하기 위하여 커널 수준이 아니라 사용자 수준에서 디바이스 드라이버를 개발하였고, 분산된 디스크들을 관리하기 위하여 Frangipani라는 분산 파일 시스템을 개발하였으며, 체인 디클러스터링(chained declustering) 구조를 적용하였다.

Petal 프로젝트가 사용자 수준에서 디바이스 드라이버를 구현한 것과 달리, CDD는 USC 대학의 Trojans 프로젝트 수행과정에서 개발된 것으로서, 커널 수준에서 디바이스 드라이버를 이용하고 있다. CDD는 SIOS 서비스를 위하여 가상 디스크를 이용한 원격 디스크 액세스 기능과 데이터 일관성 유지 기능, 그리고 S/W RAID와 같은 디스크 배열 기술 등을 포함하고 있다. (그림 2)는 CDD를 이용하여 SIOS를 구현한 방식을 보여준다. CDD는 클러스터를 구성하는 모든 노드에 분산되어 있으며, 각 노드들은 CDD간의 내부 동작을 통하여 클러스터 사용자에게 SIOS 서비스를 제공하게 된다. 따라서 SIOS를 구성하는데 있어서 가장 핵심적인 역



(그림 2) CDD를 이용한 SIOS



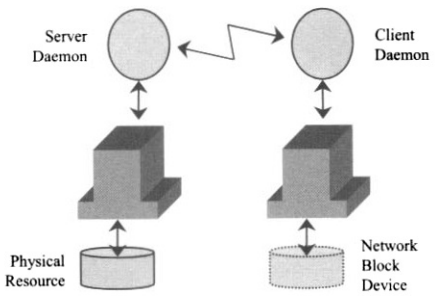
(그림 3) CDD의 구조

할을 수행하는 부분은 CDD가 된다.

(그림 3)은 CDD의 내부 구조를 보여주고 있다. (그림 3)에서 디스크 관리자(disk manager)는 원격지의 CDD 클라이언트 모듈(CDD client module)로부터의 I/O 요청을 받아서 처리하고, CDD 클라이언트 모듈(CDD client module)은 자신의 I/O 요청을 원격지의 디스크 관리자(disk manager)로 전달한다. 그리고 데이터 일관성 모듈(data consistency module)은 분산된 디스크들 간에 존재하는 데이터 사이에 일관성을 유지하는 역할을 수행한다[11].

가상 디스크 방식을 이용한 원격 디스크 액세스에 관한 다른 연구로는 P. Machek [12]이 제안한 네트워크 블록 장치(network block device: NBD) 방식이 있다. NBD는 지역 클라이언트에 하드 디스크 혹은 블록 장치의 파티션을 시뮬레이션 하는 액세스 모델을 제공하며, 이것은 실제 물리적인 저장장치를 가지고 있는 원격 서버와 네트워크를 통해 연결된다. 이렇게 제공된 가상 디스크가 클라이언트에게는 지역 디스크 파티션처럼 보이지만, 실제는 원격 디스크로의 통로 역할만 한다. 실제 액세스 요구들과 데이터 블록들이 네트워크 상에서 전송되지만, NBD 계층이 모든 상세 동작들을 숨겨주기 때문에 클라이언트는 가상 디스크를 지역 디스크처럼 액세스할 수 있게 된다. NBD의 이러한 동작은 원격 호스트에 있는 파일 계층의 액세스 요구를 정확하게 제어하기 위하여 NFS나 Samba와 같이 커널과의 더 많은 상호작용을 요구하는 네트워크 파일 시스템보다 낮은 계층에서 동작한다. (그림 4)는 이러한 NBD의 기본 개념을 보여주고 있다.

NBD는 응용에 따라 다양하게 변경되어 적용되었는데, 대표적인 예로는 Linux/NBD, DRBD(distributed replicated block device), ODR(online disk replicator), GNBD(global



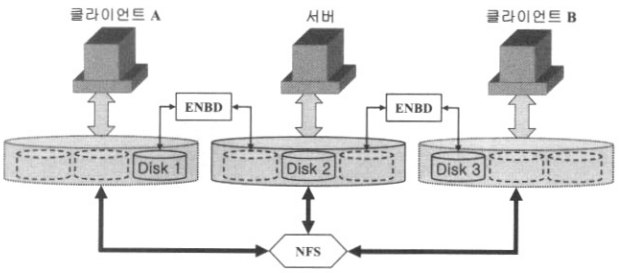
(그림 4) NBD를 이용한 원격 디스크 접근

NBD), 그리고 ENBD(enhanced NBD)가 있다. Linux/NBD는 리눅스 커널에 포함되어 있는 기본 NBD 드라이버이고, DRBD는 빠른 동기화 옵션을 제공하여 LAN 상에서 파일 시스템을 미러링하기 위하여 사용되었다. 그리고 ODR은 DRBD와 유사하지만 두 노드 이상을 지원하고, GNBD는 GFS(global file system)에 포함된 NBD로서 위에 언급된 Linux/NBD의 수정판이다. Linux/NBD와 GNBD의 가장 큰 차이점은 GNBD는 하나의 블록 디바이스에 복수 개의 클라이언트들이 동시 액세스할 수 있는 반면에, Linux/NBD는 한 시점에 단지 하나의 클라이언트만이 액세스가 가능하다는 것이다. 마지막으로, ENBD는 Linux/NBD의 기능을 확장한 것이다. ENBD는 내부적인 블록 단위의 저널링(journalling)과 다중 채널 오류복구(failover)를 사용함으로써 네트워크를 이용하여 보다 손쉽고 안전하게 실시간 미러링을 제공한다. 저널링이란, 파일 시스템 드라이버가 파일 시스템에 어떠한 변경을 가하기 이전에 어떤 일을 할 것인지에 대한 내용을 저널에 기록한 다음 파일 시스템을 수정하는 것이다. 그렇게 함으로써 저널은 최근의 파일 시스템 수정사항의 로그를 유지하게 된다. 이 저널링은 시스템 크래쉬나 정전 후의 복구를 도와준다[12]. 이 ENBD는 본 연구에서 이용한 공개 소스 프리웨어들 중의 하나이다.

3. 프리웨어를 이용한 SIOS의 구현

본 연구에서는 리눅스 프리웨어인 ENBD와 S/W RAID 및 NFS를 이용하여 SIOS를 구현하였으며, 이 소프트웨어들은 모두 공개 소스이다. ENBD는 디바이스 드라이버 레벨의 하위 계층에서 동작하고, S/W RAID와 NFS는 파일 시스템 레벨의 상위 계층에서 동작한다. ENBD는 가상 디스크를 이용하여 원격 디스크를 액세스할 수 있게 하고, S/W RAID는 서버에서 클러스터에 분산되어 있는 디스크들을 하나의 메타 디스크(meta disk)로 묶는 기능과 RAID 시스템에서 제공하는 결함 허용 기능을 자체적으로 제공한다. 그리고 NFS는 서버에서 생성된 메타 디스크를 클라이언트들이 액세스할 수 있게 해주며, 다중 액세스를 위하여 데이터 일관성을 유지할 수 있게 한다.

SIOS의 구성 과정은 다음과 같은 순서로 진행된다. 먼저 클러스터 시스템의 서버 노드는 ENBD를 이용하여 다른 노드의 원격 디스크들을 가상 디스크들로 만들고, S/W RAID



(그림 5) 프리웨어를 이용한 SIOS

〈표 1〉 CDD와 ENBD의 비교

	CDD	ENBD
구조	분산형	중앙 집중형
구현 모듈	일체형	조립형
확장성	제한적	높음
S/W 종류	비공개 S/W	공개소스 프리웨어
이식성	낮음	높음
설치	커널 수정 필요	용이

방식을 이용하여 가상 디스크들을 하나의 거대한 메타 디스크로 구성한다. 이렇게 구성된 메타 디스크는 단일 주소 공간을 가지게 되고, NFS를 통하여 다른 모든 클라이언트 노드들에 의해 사용될 수 있게 된다. 결과적으로 클러스터 시스템 내의 모든 노드들은 SIOS를 가지게 된다. (그림 5)는 이러한 방법으로 구현된 SIOS의 형태를 보여주고 있다.

클러스터 시스템을 구성하는 서버 노드와 클라이언트 노드들은 SIOS에 접근하는 방법에 차이가 있다. 서버 노드는 ENBD만 경유하면 다른 노드의 디스크를 액세스할 수 있다. 그러나 클라이언트 노드가 다른 노드의 디스크를 액세스하기 위해서는 NFS 혹은 NFS 및 ENBD를 모두 경유해야 한다: (그림 5)에서 클라이언트 A가 서버 노드의 디스크를 액세스하려면 NFS만 이용하면 된다; 그러나 클라이언트 B의 디스크를 액세스하려면 NFS와 ENBD를 모두 경유해야 가능하다.

본 연구는 리눅스 환경에서 가상 디스크와 S/W RAID를 사용하여 SIOS를 구현한다는 측면에서는 앞에서 언급한 CDD와 유사하지만, <표 1>과 같은 차이점이 있다.

먼저 구조 측면에서 보면, CDD는 분산형이기 때문에 노드에 SIOS 서비스를 제공하기 위한 모든 기능들이 CDD에 포함되어 있다. 따라서 CDD 모듈은 일체형으로 되어 있어 SIOS의 크기나 내부 구성을 변경하는 것이 어렵다는 문제점이 있다. 반면에 본 연구에서 구현한 ENBD 방식은 클라이언트-서버 형태를 가지는 중앙집중형이므로 서버에 SIOS 서비스 기능이 집중되어 있고, 구현 모듈이 상위 계층과 하위 계층으로 분리되어 있기 때문에 SIOS의 확장 및 축소가 용이하다. 또한 CDD에서의 SIOS 서비스는 CDD를 포함한 노드들에게만 제공되지만, 본 연구에서 구현한 ENBD 방식은 클러스터 시스템의 구성에 참여하지 않았지만 서버에 NFS로 연결 가능한 모든 컴퓨터에 동일한 SIOS를 제공한다.

SIOS 구축 측면에서 보면, CDD의 소프트웨어 라이브러리는 비공개되어 SIOS 구축이 어렵지만, 본 연구에서 구현한 ENBD 방식은 리눅스 프리웨어들만을 사용하였기 때문에 비용을 들이지 않고 쉽게 SIOS를 구현할 수 있다. 이식성 측면에서도 CDD는 클러스터 내의 모든 노드에 CDD 라이브러리를 설치한 후에 커널 재컴파일이 필요하지만, 본 연구에서 구현한 ENBD 방식은 커널 재컴파일 없이 ENBD 소프트웨어 설치만이 필요하다.

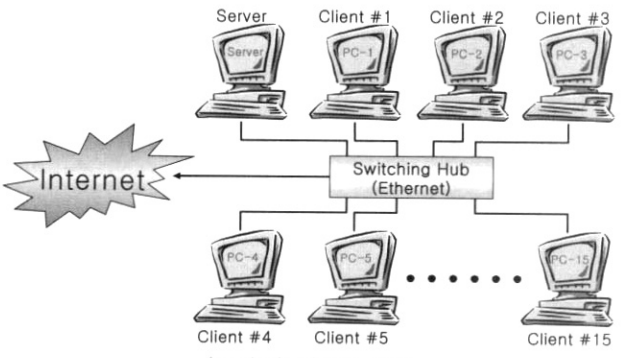
4. 실험 방법

실험은 본 연구에서 제안한 방법을 객관적인 조건에서 평가하기 위하여 앞에서 언급한 CDD[11]와 유사한 클러스터 환경을 구성하여 수행하였다. 클러스터 시스템은 (그림 6)과 같이 스위칭 허브를 통하여 10 Mbps 또는 100 Mbps의 네트워크 대역폭을 가지는 이더넷으로 16개의 PC 노드들을 연결하여 구성하였다.

각 노드의 운영체제는 리눅스 커널 2.4.20의 Red-Hat 9이며, SIOS를 위하여 각 노드의 하드 디스크에 1 GB 용량을 할당하여 전체 SIOS 용량을 최대 16 GB로 설정하였다. 디스크 분산방식은 RAID-5로 구성하였고, RAID-5에 포함되는 노드의 수를 변화시키면서 진행하였으며, Bonnie 벤치마크를 각각 10회씩 실행하여 평균값을 산출하였다.

Bonnie 벤치마크는 디스크 성능을 평가하기 위하여 널리 사용되는 도구로서 '문자 읽기', '문자 쓰기', '블록 읽기', '블록 쓰기', '파일 재쓰기', 그리고 '임의 검색'에 대하여 결과를 도출한다. 본 실험에서는 이 항목들 중에서 작은 읽기/쓰기(small read/write)인 '문자 읽기'와 '문자 쓰기', 그리고 큰 크기의 읽기/쓰기(large read/write)인 '블록 읽기'와 '블록 쓰기'에 초점을 두었다. 한편 작업부하 크기는 리눅스에서 블록 디바이스마다 설정되어 있는 4 MB의 디스크 버퍼의 영향을 최소화하기 위하여 작업부하를 500 MB로 설정하였고, 블록 크기는 기본값인 16 KB로 설정하였다. 따라서 벤치마크에서 각 노드가 블록 읽기 혹은 쓰기를 하는 경우에는 한 번에 16 KB씩 반복 수행하여 500 MB를 액세스하게 된다.

본 연구에서는 클러스터 시스템의 I/O 처리율을 다양한 측면에서 측정하기 위하여 서비스 방식, 네트워크 대역폭, 다중 노드 액세스, 그리고 노드의 하드웨어 성능에 변화를



(그림 6) 시스템 구성

〈표 2〉 노드의 하드웨어 성능

		저성능 PC	고성능 PC
CPU		Pentium-Pro 180 MHz	Pentium-4 1.8 GHz
RAM		64 MB	256 MB
HDD	Model	Seagate ST33232A	Samsung SP0411N
	Capacity	3 GB	40 GB
	Average seek time	12 ms	10 ms
	Internal data transfer rate	87.8 Mbits/s	728 Mbits/s
	I/O data transfer rate	33.3 MB/s	133 MB/s
	Spindle speed	4,500 rpm	7,200 rpm
	MTBF	300,000 hours	500,000 hours

주면서 실험하였다. 서비스 방식으로는 NFS를 이용한 서비스, ENBD를 이용한 서비스, 그리고 NFS 및 ENBD를 모두 이용한 서비스에 대하여 분석하였고, 네트워크 대역폭은 100 Mbps인 경우와 10 Mbps인 경우에 대하여 각각 실험하였다. 그리고 다중 노드 동시 액세스(multiple-node concurrent access)에 관한 분석은 클러스터 컴퓨터를 구성하는 16개의 노드들 중에서 동시에 디스크 액세스 동작을 수행하는 노드의 수를 한 개부터 다섯 개까지 증가시키면서 수행하였다. 각 클러스터 노드의 하드웨어로는 <표 2>와 같은 저성능 PC(low-end PC) 및 고성능 PC(high-end PC)인 경우를 각각 구분하여 실험하였다.

5. 실험 결과

5.1 NFS와 ENBD의 오버헤드

<표 3>은 고성능 PC 노드들이 100 Mbps로 연결된 클러스터 컴퓨팅 환경에서 디스크 분산 방식을 RAID-5로 구성한 후, 서비스 방식에 따른 I/O 처리율을 측정한 결과를 보여주고 있다. 아래 <표 3>에서 지역 디스크 서비스란 프로세서가 자신의 노드내 디스크를 액세스하는 경우를 말한다. NFS 서비스란 클라이언트 노드가 서버 노드의 디스크를 액세스하는 경우로서, 예를 들어 (그림 5)에서 클라이언트 A가 디스크 2를 액세스하려면 서버가 그 요구를 받아서NFS를 통하여 처리해주게 된다. ENBD 서비스란 서버 노드가 클라이언트 노드의 디스크를 액세스하기 위하여 ENBD를 이용하는 경우에 해당한다. 예를 들어 (그림 5)에서 서버가 디스크 1을 액세스하는 경우에는 ENBD를 통해야 한다. 마지막으로, NFS를 통한 ENBD 서비스는 원격 디스크 액세스로서, 클라이언트 노드가 다른 클라이언트 노드의 디스크를 액세스하는 경우에 먼저 NFS를 통하여 서버로 액세스 요구를 보내면 서버는 ENBD를 이용하여 목적지 클라이언트 노드의 디스크를 액세스한 다음에 결과를 보내준다. 그 동작의 예로는 클라이언트 A가 디스크 3을 액세스하는 경우이다.

〈표 3〉 액세스 서비스에 따른 I/O 처리율

	블록 쓰기	블록 읽기
지역 디스크	28.80	42.14
NFS	7.66	12.93
ENBD	7.23	11.00
NFS+ENBD	3.99	10.10

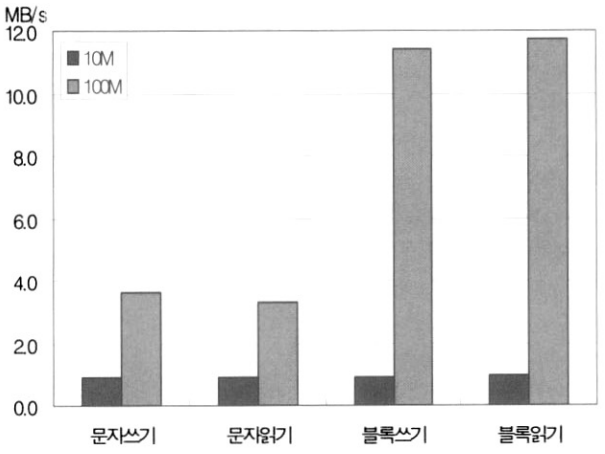
단위: MB/s

<표 3>의 결과에 따르면, 지역 디스크를 액세스하는 경우에는 네트워크 지연과 소프트웨어 오버헤드가 없기 때문에 간접 액세스에 해당하는 다른 서비스들에 비하여 I/O 처리율이 훨씬 더 높다. 반면, NFS를 통한 액세스의 경우에는 NFS 프로그램 처리 시간에 의한 지연 때문에 I/O 처리율은 크게 떨어진다. 그러나 NFS는 그간 많은 연구를 통하여 오버헤드가 최소화된 S/W이므로 이와 같은 간접 액세스 성능으로는 받아들일만한 수준이다. 다른 간접 액세스 방식인 ENBD 서비스는 프리웨어를 이용한 구현이지만 NFS 방식에 비하여 성능이 85%~94% 정도로서, 크게 뒤떨어지지 않는 것으로 나타났다. 그리고 마지막으로 NFS를 통한 ENBD 서비스의 경우에는 쓰기 연산에 있어서는 52%~55% 정도로서, 다소 큰 성능의 저하를 보이고 있지만 읽기 연산에 있어서는 NFS 서비스나 ENBD 서비스에 비하여 78%~92% 정도로서, 비슷한 성능을 나타내고 있다. 실험 결과를 보면 읽기 연산에 비하여 쓰기 연산에 있어서 다소 큰 성능의 저하가 나타나고 있는데, 이것은 RAID-5의 특성상 나타나는 작은 쓰기 문제(small write problem)에 기인한 오버헤드 증가를 그 원인으로 볼 수 있다. RAID-5에서 쓰기 연산을 수행할 때 순차적으로 여러 블록들을 동시에 쓰는 큰 쓰기(large write)의 경우에는 별 문제가 없지만, 어느 한 블록만 쓰는 작은 쓰기(small write)의 경우에는 매 쓰기 동작 때마다 네 번의 디스크 액세스 요구가 발생하게 되는 문제가 나타나는데 이것을 작은 쓰기 문제라 부른다. 이 결과들을 종합해 보면, ENBD 서비스의 I/O 처리율이 NFS 서비스의 I/O 처리율에 비하여 전반적으로 크게 뒤떨어지지는 않는다는 것을 알 수 있으며, NFS를 통한 ENBD 서비스에 있어서는 쓰기 연산에서는 다소 큰 성능의 저하가 나타나지만 읽기 연산에 있어서는 I/O 처리율이 크게 뒤떨어지지는 않는다는 것을 알 수 있다.

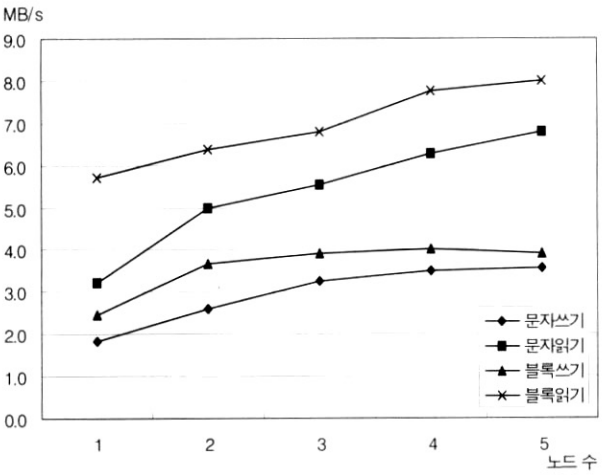
5.2 네트워크 대역폭에 따른 I/O 처리율

(그림 7)은 16대의 고성능 PC 노드들로 구성된 클러스터에서 노드 간의 네트워크 대역폭이 10 Mbps인 경우와 100 Mbps인 경우에 대하여 각각 I/O 처리율을 측정한 결과를 보여주고 있다. 이 실험에서는 한 개의 노드가 시스템 내 임의의 노드에 있는 디스크를 액세스하며, 작업부하의 크기는 500 MB이다. 즉, 문자 읽기/쓰기 연산의 경우에는 바이트 단위의 동작이 반복되며, 블록 읽기/쓰기 연산에서는 블록 단위의 동작이 반복 수행된다.

(그림 7)에서 보는 바와 같이 네트워크 대역폭이 10 Mbps에서 100 Mbps로 증가할 경우에 전반적으로 I/O 처리율이 문자 단위 연산에 있어서는 4배, 블록 단위 연산에 있어서는 12~14배로 크게 증가하는 것으로 나타났다. 네트워크 대역폭이 10 Mbps일 경우에는 각 노드를 구성하는 하드웨어의 성능이 높아지더라도 I/O 연산을 위한 데이터 전송 시 네트



(그림 7) 네트워크 대역폭에 따른 I/O 처리율



(그림 8) 다중 노드 액세스에 따른 I/O 처리율

워크 대역폭이 제한됨에 따라 모든 종류의 I/O 연산들에 대하여 처리율이 매우 낮게 나타나는 것을 볼 수 있다. 대역폭이 100 Mbps인 경우에는 블록 단위 연산에 비해 문자 단위 연산의 I/O 처리율이 30% 정도로서, 크게 낮은 것으로 나타났는데, 이것은 동일한 크기의 작업부하에 대하여 블록 단위 연산보다 문자 단위 연산이 훨씬 더 많은 횟수의 독립적인 I/O 연산을 수행해야 하기 때문이다. 실제로 Bonnie 벤치마크 프로그램의 문자 단위 연산 부분은 "getc()" 함수 또는 "putc()" 함수로 되어 있고, 블록 단위 연산은 "read()" 함수 또는 "write()" 함수로 되어 있다. 따라서 16 KB의 작업부하일 경우에 블록 I/O는 한 번의 함수 호출로 연산이 끝나지만, 문자 I/O의 경우에는 16,384번의 함수 호출이 발생하게 된다. 특히 블록 읽기 및 쓰기의 경우에는 문자 읽기 및 쓰기 경우에 비하여 I/O 처리율이 3배 정도로서, 현저히 높은 것으로 나타났다. 이 결과들을 종합해 보면, 클러스터 환경에 있어서 네트워크 대역폭이 SIOS의 성능에 매우 큰 영향을 미치는 요인이라는 것을 알 수 있다.

5.3 다중 노드 액세스에 따른 I/O 처리율

이 절에서는 클러스터 내의 여러 노드들이 동시에 서로 다른 노드의 디스크를 액세스하는 경우의 I/O 처리율을 측정한 결과에 대하여 설명한다. 이 경우에는 서버가 클러스터 노드들로부터의 요구들을 처리하기 위하여 NFS를 순차적으로 수행해야 하기 때문에 S/W 오버헤드가 커질 것이며, 노드 수가 증가함에 따라 네트워크 트래픽도 점차 더 높아질 것이다. (그림 8)은 16대의 고성능 PC 노드들이 100 Mbps의 네트워크에 의해 접속된 클러스터에서 동시 액세스 노드의 수를 한 개부터 다섯 개까지 순차적으로 증가시키며 실험한 결과를 보여주고 있다. 이때 서버를 제외한 클라이언트 노드들만 Bonnie 벤치마크를 수행하여 디스크 액세스를 발생하도록 하였다.

전반적으로, 동시에 액세스하는 노드들의 수가 증가함에 따라 I/O 처리율이 어느 정도 증가하지만, 그 노드 수에 비례하지는 못한다는 것을 알 수 있다. 특히 쓰기 동작들의 경우에는 노드 수가 네 개 이상일 때는 I/O 처리율이 증가하지 못하고 포화되는 것으로 나타났다. 이러한 현상은 다수의 노드들이 동시에 다른 노드들의 디스크를 액세스함으로써 네트워크 트래픽이 증가되어 데이터 전송 시간이 크게 증가하기 때문이다. 즉, 그 시점부터 네트워크가 병목(bottleneck)이 된다는 것을 의미한다. 따라서 클러스터 시스템을 구성할 때 네트워크 대역폭을 가능한 한 높여야 SIOS가 효과적으로 이용될 수 있다는 것을 확인할 수 있다. 실제로 최근의 클러스터 컴퓨터들은 기가빗(gigabit) 이더넷이나 미리넷(myrinet)등과 같은 1 Gbps급 이상의 네트워크가 사용되고 있다.

5.4 CDD와의 성능 비교

(그림 9)는 네트워크 대역폭이 100 Mbps인 클러스터에서 노드들에 위치한 디스크들을 RAID-5로 구성하고, 노드 하드웨어로서 저성능 PC를 사용하는 경우와 고성능 PC를 사용하는 경우의 I/O 처리율을 측정하여 CDD[11]와 비교한 결과를 보여주고 있다. 앞에서 언급한 관련 연구들 중에서 리눅스 환경에서 가상 디스크와 S/W RAID를 사용하여 SIOS를 구현한다는 측면에 있어서 CDD가 본 연구에서 구현한 ENBD 방식과 가장 유사하기 때문에 CDD를 비교 대상 시스템으로 선정하였다. CDD에서의 노드 하드웨어는 본 연구의 저성능 PC와 고성능 PC의 중간 정도인 400 MHz 펜티엄-II CPU와 10 GB IDE 하드디스크를 사용하는 PC들로 구성하였으며, 각 노드의 운영체제는 리눅스 2.2.5의 Red-Hat 6가 사용되었다. 실험은 노드의 수를 2개부터 최대 16개까지 증가시키면서 수행하였다. 따라서 RAID-5가 그 수만큼의 디스크들로 구성된다. 그리고 실험 조건을 CDD의 실험 환경과 동일하게 해주기 위하여, 5.3절의 실험 환경과는 달리 클러스터 내의 서버 노드만 Bonnie 벤치마크를 수행하면서 디스크 액세스 요구를 발생하도록 하였다.

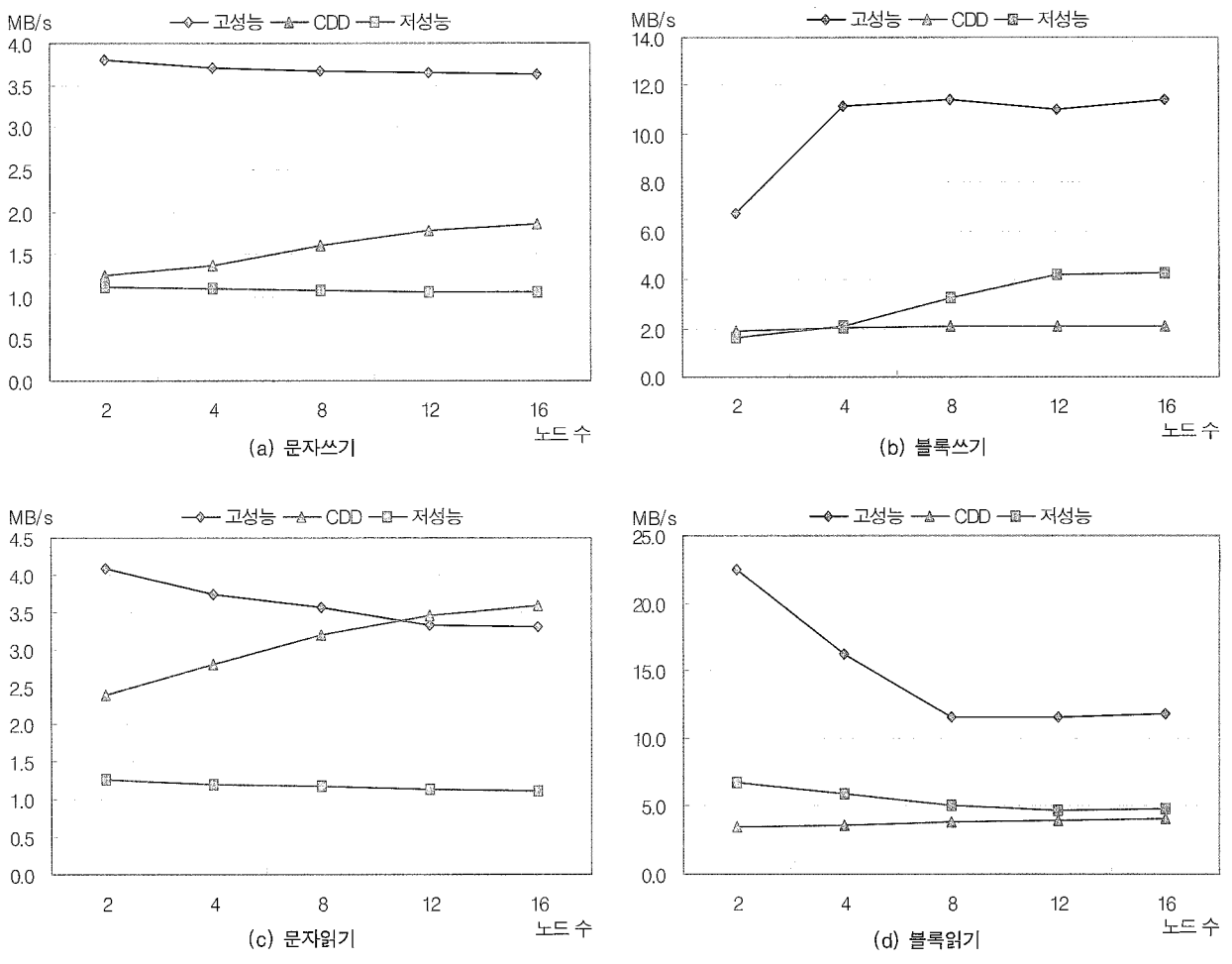
결과에 따르면, 고성능 PC들로 구성된 클러스터에서는 CDD에 비하여 문자 쓰기의 경우에 2배, 블록 쓰기의 경우

에 5.5배 더 높은 I/O 처리율을 나타내었다. 문자 읽기의 경우에는 클러스터를 구성하는 노드의 수가 12개 이하인 경우에는 본 연구의 결과가 더 높게 나타났으나, 그 이상의 노드에 대해서는 CDD보다 성능이 다소 낮아졌다. 또한 블록 읽기의 경우에는 노드의 수가 2개 및 4개인 경우에는 월등히 높은 I/O 처리율을 보였으며, 그 이상의 노드 수에 대해서도 CDD보다 2.3배 더 높은 성능을 보였다. 저성능 PC들로 구성된 클러스터에서는 문자 읽기 및 쓰기의 경우에 CDD보다 낮은(31%~56%) I/O 처리율을 가지는 것으로 확인되었다. 그러나 블록 읽기 및 쓰기에 대해서는 CDD보다 I/O 처리율이 1.2~2배 더 높은 것으로 나타났다. 결과적으로, 커널 수준에서 별도로 개발된 디바이스 드라이버를 사용하는 CDD에 비하여, 본 연구에서 개발한 SIOS 구현 방식은 프리웨어를 사용하는 이점을 가지면서도 I/O 처리율도 전반적으로 더 높게 나타난다는 것을 확인할 수 있었다.

6. 결 론

본 연구에서는 리눅스 PC 클러스터 시스템에서 현재 공개되어 있는 프리웨어인 ENBD, S/W RAID 그리고 NFS들만을 이용하여 SIOS를 구현하는 방법을 제시하였다. 이 방법의 주요 장점은 프리웨어들만을 이용하기 때문에 구현이 용이하고 비용이 들지 않는다는 점이다. 또한 본 연구에서 사용한 프리웨어들은 공개 소스이기 때문에 다른 플랫폼, 즉 그리드 컴퓨터와 같이 이기종(heterogeneous)으로 구성된 시스템에도 약간의 수정을 통하여 본 연구의 방법이 적용이 가능하다는 장점이 있다. 그리고 본 연구의 실험 결과에 따르면, 그러한 장점을 가지면서도 I/O 처리율도 커널 수준에서 별도로 개발된 디바이스 드라이버를 사용하는 CDD보다 쓰기 동작의 경우에는 최대 5.5배, 읽기 동작의 경우에는 2.3배정도 더 높게 나타났다.

Bonnie 벤치마크를 이용한 다양한 성능 측정 결과에 따르면, 클러스터 시스템에 있어서 서비스 방식에 따라 발생되는 오버헤드, 네트워크 대역폭에 따른 I/O 처리율의 제한, 다중 노드 액세스에 따라 발생되는 병목 현상, 그리고 클러스터를 구성하는 노드의 하드웨어 성능 등이 SIOS 서비스의 I/O 처리율에 영향을 주는 요소라는 것이 확인되었다. 이러한 요소들이 클러스터 시스템의 성능에 많은 영향을 주는 것은 사실이지만, 클러스터 시스템의 성능을 높이기 위하여 이러한 모든 요소들의 성능을 높이는 것은 많은 비용이 든다는 문제가 있다. 클러스터 시스템을 구축하는 근본적인 목적이 저렴한 비용으로 고성능 병렬컴퓨팅 환경을 구축하



(그림 9) CDD와의 I/O 처리율 비교

는 것인 만큼, 클러스터 시스템의 사용 목적에 맞게 각각의 요소들의 성능을 적절하게 조합하여 클러스터 시스템을 구축할 필요가 있다. 그런 의미에서 본 연구에서 제시한 방법들은 클러스터 시스템에서 효율적인 SIOS를 구축하는데 필요한 기술적 요소들을 결정하기 위한 기초 자료로 유용하게 사용될 수 있을 것이다.

참 고 문 헌

[1] K. Hwang and Z. Xu, 'Scalable Parallel Computing: Technology, Architecture, Programming,' McGraw-Hill, New York, 1998.

[2] K. Hwang, H. Jin, E. Chow, C. L. Wang and Z. Xu, " Designing SSI Clusters with Hierarchical Check-pointing and Single I/O Space," IEEE Concurrency Magazine, pp.60-99, March 1999.

[3] R. Ho, K. Hwang and H. Jin, "Design and Analysis of Clusters with Single I/O Space," Proc. 20th Int's Conf. Distributed Computing Systems (ICDCS 2000), pp.120-127, Apr. 2000.

[4] G. F. Pfister, 'In Search of Clusters,' 2nd Ed., Prentice Hall, 1998.

[5] R. Ho, K. Hwang and H. Jin, "Single I/O space for scalable cluster computing," Proc. 1th IEEE Computer Society Int's Workshop Cluster Computing, pp.158-166, Dec. 1999.

[6] P. H. Carnset et al., "PVFS: A Parallel File System for Linux Clusters," Proc. Extreme Linux Track: Fourth Anm. Linux Showcase and Conf., Oct. 2000.

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

[8] P. F. Corbett, D. G. Feitelson, J. P. Prost and S. J. Baylor, "Parallel Access to Files in the Vesta File System," Proc. Supercomputing '93, 1993.

[9] M. Dahlin, R. Wang, T. Anderson and D. Patterson, " Cooperative Caching: Using Remote Client Memory to Improve File System Performance," Proc. Operating System Design and Implementation, 1994.

[10] I. Foster, D. Kohr Jr., R. Krishnaiyer and J. Mogill, "Remote I/O: Fast Access to Distant Storage," Proc. Fifth Workshop I/O in Parallel and Distributed Systems, pp.14-25, Nov. 1997.

[11] K. Hwang, H. Jin and R. Ho, "Orthogonal striping and mirroring in distributed RAID for I/O-centric cluster computing," IEEE Transactions on Parallel and Distributed Systems, Vol.13, No.1, pp.26-44, Jan. 2002.

[12] P. T. Breuer, A. M. Lopez and A. G. Ares. "The network block device," Linux Journal,(73), May 2000.

김 태 규

e-mail : windcry@korea.com
2002년 연세대학교 전산학과(학사)
2004년 연세대학교 전산학과(이학석사)
2004년~현재 연세대학교 컴퓨터정보통신공학부 박사과정
관심분야 : 센서 네트워크, 클러스터 컴퓨터, 병렬처리, 시뮬레이션

김 방 현

e-mail: legnamai@chol.com
1996년 연세대학교 전산학과(학사)
2001년 연세대학교 전산학과(이학석사)
2002년~현재 연세대학교 컴퓨터정보통신공학부 박사과정
관심분야 : 센서 네트워크, 클러스터 컴퓨터, 병렬처리, 시뮬레이션

김 종 현

e-mail : jhkim34@yonsei.ac.kr
1976년 연세대학교 전기공학과(학사)
1981년 연세대학교 전기공학과(공학석사)
1988년 Arizona State University 전기 및 컴퓨터공학과(Ph.D)
1976년~1982년 국방과학연구소 연구원
1988년~1990년 한국전자통신연구소 실장
1990년~현재 연세대학교 컴퓨터정보통신공학부 교수
1996년 Oregon State University 전산학과 방문교수
2003년 Florida State University 전산학과 방문교수
관심분야 : 컴퓨터구조, 병렬처리, 시뮬레이션, 센서 네트워크