프로세스 그룹화를 이용한 집합 I/O

차광호° 홍정우 이지수
한국과학기술정보연구원 슈퍼컴퓨팅센터
{khocha°, jwhong, jysoo}@kisti.re.kr

Collective I/O with Process grouping

Kwangho Cha°, Jeongwoo Hong, Jysoo Lee
Supercomputing Center
Korea Institute of Science and Technology Information

요 약

병렬 처리를 요구하는 계산 과학 분야의 문제들 중에는 대용량 데이터 처리를 필요로 하는 경우가 많다. 그러나 기존의 파일 시스템을 그대로 병렬처리 환경에 적용하기에는 많은 문제가 따른다. 이를 위해서 병렬처리를 지원하는 파일 시스템에 대한 연구와 개발이 진행되어 오고 있다. 이와 같은 연구 중 하나인 집합 I/O(Collective I/O)를 본 논문에서 다루고자 한다. 이 집합 I/O는 여러 프로세스의 파일 I/O 요청을 효과적으로 처리하는 방법으로 MPI2의 MPI-IO에도 포함되어 있다. 본 논문에서는 어플리케이션 프로그램 측면에서 MPI-IO의 집합 I/O를 효과적으로 사용하기 위한 방안을 제시하며, 보편적으로 사용되는 NFS를 이용한 클러스터 시스템에서의 실험 결과를 분석한다.

1. 서론

병렬 처리를 요구하는 계산 과학 분야의 문제들 중 대부분이 대용량 데이터 처리를 필요로 하고 있는데, 이러한 범주의 응용 프로그램들을 'Parallel out-of-core'라고 부르고 있다. 이러한 어플리케이션들을 처리하기 위한 노력 중 하나로 MPI2 라이브러리에 포함된 MPI-IO를 생각해 볼 수 있다[1]. 본 논문에서는 MPI-IO 기능 중, 파일 시스템의 성능에 중요한 영향을 미치는 집합 I/O(Collective I/O)에 대하여 다루고자 한다.

MPI-IO에서 제공하는 집합 I/O(Collective I/O)를 구현하는 방법들 중 대부분은 2 단계 I/O(2 Phase I/O)와 같이 별도의 특수한 I/O프로세스 노드들을 가지고 있는 병렬 아키텍쳐를 필요로 하고 있으며, MPI-IO가 NFS보다 PVFS에서 우수한 성능을 나타낸다는 연구 결과[1]는 NFS와 같은 일반적인 파일 시스템에서는 오히려 성능 저하를 가져 올 수 있다는 점을 보여주고 있다.

반면 병렬 컴퓨터의 한 분야로 간주되고 있는 클러스터 시스템은 일반적으로 NFS를 주요 파일 시스템으로 사용하고 있는 상황이다. 이처럼 실제 병렬 파일 시스템을 갖추지 못한 클러스터 시스템에서 MPI-IO를 사용하는 경우에는, 성능 저하 문제를 고려하지 않을 수 없다. 물론 MPI-IO대신, 기존의 I/O라이브러리를 이용하여 순차파일을 사용할 수 도 있으나, MPI-IO가 갖는 많은 장점을 포기하여야 하는 제약이 따른다[2].

본 논문에서 제안한 프로세스 그룹화는 기존의 MPI함수들을 이용하여 MPI-IO를 보다 효과적으로 사용할 수 있도록 제시한 프로그래밍 모델이다. 클러스터 시스템에서의 성능 측정 결과, 기존의 MPI-IO함수만을 사용한 결과보다 개선된 성능을 보여주고 있다.

2. 관련 연구

본 장에서는 집합 I/O의 개념과 MPI-IO에 사용된 집합 I/O에 대하여 간략하게 설명하고자 한다.

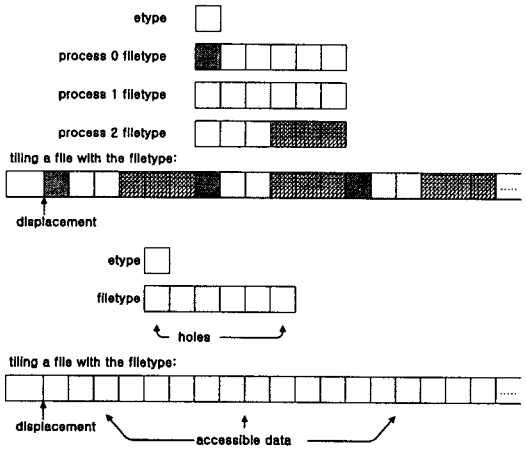
2.1 집합 I/O

병렬 어플리케이션이 요구하는 파일처리의 특성은 각각의 프로세스가 자신의 데이터를 파일에 기록하고 참조하려는데 있는데, 이때 데이터는 메모리 또는 파일시스템에 불연속적으로 존재하는 특징을 갖는다[3, 4]. 이처럼 복수의 프로세스가 수행하는 불연속적인 파일 참조를 지원하기 위한 방법중 하나가 집합 I/O로서 이를 구현하는데는 2 단계 I/O(2 Phase I/O)나 디스크 다이렉트 I/O(Disk Direct I/O) 같은 방법이 제시되고 있다[3].

2.2 MPI-IO의 집합 I/O

위에서 설명한 집합 I/O를 MPI 라이브러리에서도 지원하고 있다. MPI를 보강하여 MPI2 라이브러리가 발표되면서 추가된 기능으로는 원격 메모리 접근(RMA)와 MPI-IO가 있는

데[2] MPI-IO의 기능 중, 일부로 집합 I/O가 제공되고 있다. 집합 I/O는 기존 MPI 라이브러리에서 제공되는 원시 자료형 또는 파생 자료형(Derived Data Type)를 이용하여 파일형을 생성한뒤 파일에 접근하게 된다. [그림 1]은 파일형의 예를 보여 준다[2,5].



[그림 1] MPI-IO에서의 파일형과 병렬 프로세서의 파일 분할 예 [2,5]

이러한 MPI-IO가 제공되는 MPI라이브러리로는 NAS의 PMPIO와 Argonne National Laboratory의 ROMIO를 들 수 있는데, 이 중 ROMIO는 MPICH 라이브러리에 포함되어 제공되고 있으며, NFS와 같은 일반적인 파일 시스템뿐만 아니라, PVFS[6] 같은 병렬 처리용 분산 파일시스템을 지원하기 위한 프로그래밍 인터페이스도 제공하고 있다[7].

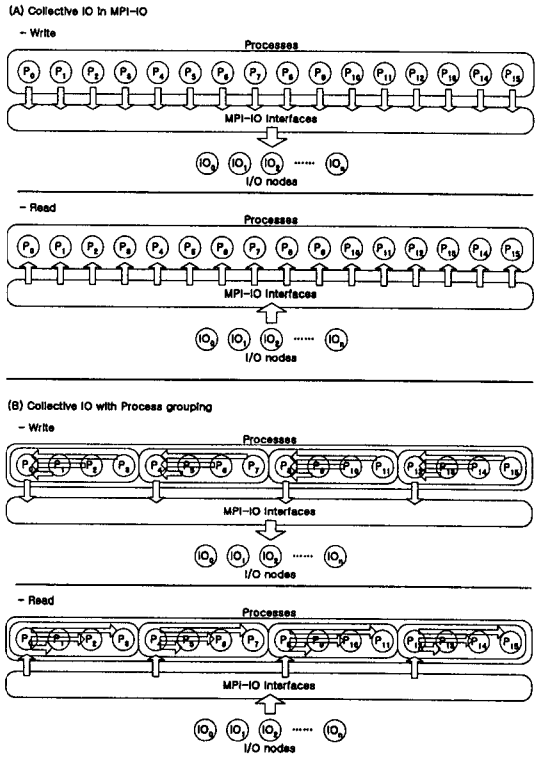
3. 프로세스 그룹화를 이용한 집합 I/O

기존의 집합 I/O의 구현을 살펴보면, 집합Write나 집합Read같은 I/O작업을 수행하기 위해서는 모든 프로세스간에 I/O수행에 필요한 정보 조율이 필요하다. 이외에도 데이터를 I/O프로세스(IOP)로 보내야 하므로 한번의 집합 I/O를 수행 할 때 여러 번의 전체 통신이 발생하게 된다[3,8].

이와 같은 통신 부하는 프로세스수에 증가에 따라 급격하게 증가할 것이며, 또한 특수한 IOP가지고 있는 병렬 파일 시스템이 아닌 일반적인 파일 시스템으로 구성된 시스템의 경우에는, 병렬 파일 시스템의 장점을 살리지 못하므로 통신 부하로 인한 성능 저하가 급격하게 나타 날 것이라는 가정 아래, 본 논문에서는 집합 I/O를 수행하는 프로세스의 수를 줄이고자 하는 시도를 하였다.

본 논문에서 제안하는 프로세서 그룹화 방법은 응용 프로그램에서 집합 I/O를 수행할 시에, 프로세스들을 몇 개의 그룹으로 묶은뒤 이들 그룹당 1 개의 프로세스만 집합 I/O를 수행토록하는 서브 루틴의 일종이다.

[그림 2]는 프로세스 그룹화의 예를 보여주고 있다. 프로세스 그룹화를 거치지 않은 경우에는 모든 프로세스가 집합 I/O를 수행하게되나, 프로세스 그룹화를 거친 경우에는 그룹화 계수([그림 2]에서는 4)만큼의 프로세스들을 그룹으로 묶은뒤 집합 I/O를 수행하게 된다. 즉, 그룹화를 수행하는 경우에 각 프로세스들은 자신이 속한 그룹의 마스터 프로세스에게 자신의 데이터를 전달하고, 그후 마스터 프로세스는 자신이 속한 그룹의 데이터를 가지고 집합 I/O를 호출하는 구조이다.



[그림 2] 프로세스 그룹화 : 프로세스 수 = 16, 그룹화 계수 = 4

이와 같은 그룹화 과정은 기존의 기본적인 MPI 원시 함수들을 이용하여 처리가 가능하다. 그룹화 계수 및 MPI 커뮤니티 관리 함수를 이용하여 프로세스 그룹화를 수행하였고, 그룹 내에서의 데이터 취합 및 배분은 MPI 집합 통신(Collective Communication)을 사용하여 처리하였다.

4. 성능 측정

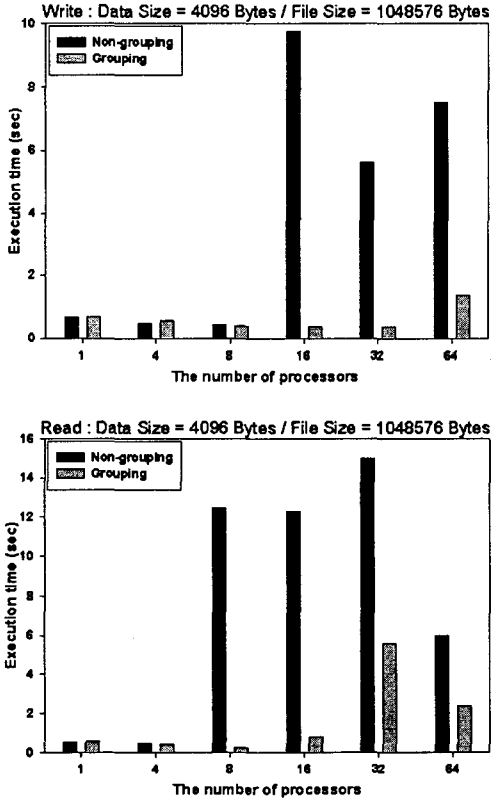
성능 측정을 하기 위하여 사용된 시스템은 한국과학기술정보연구원 슈퍼컴퓨팅센터에서 구축한 PLUTO 클러스터 시스템으로 구성 요소는 [표 1]과 같다. 본 시스템의 파일 시스템 구성은 4 개의 파일 서버에 클러스터 파일 시스템을 탑재한 뒤 각각의 계산 노드에 NFS를 이용하여 공유 스토

리지를 제공하는 형태이다.

[표 1] PLUTO 시스템의 구성 요소

계산 노드/128nodes		CPU : Intel Pentium IV (1.7GHz)
서버	4 로긴 서버	Memory : RDRAM 1GB
	관리 서버	PC Server
	FTP 서버	PC Server
	4 파일 서버	Compaq DL360
스토리지	RAID	Uniwide Willow 6000
	File System	Macro Impact SANique CFS
네트워크	Myrinet	Myrinet 2000 (3.2Gbps)
	Ethernet	3Com SuperStack 4300 (100Mbps)
		3Com SuperStack 4900 (1Gbps)
	FC Switch	Brocade Silkworm 2400

계산 노드 수를 증가 시키면서 테스트를 수행하였는데, 각 노드들은 4096 바이트를 생성하고 동일한 파일 크기를 유지하기 위하여 반복 횟수를 조정하였다. 테스트 결과는 [그림 3] 과 같다.



[그림 3] 성능 측정 결과

5. 결론

이전의 연구 결과와 같이 성능 평가에 사용된 파일시스템이 병렬 파일을 지원하는 특수 파일 시스템이 아닌 NFS이므로 프로세서 수가 증가함에 따라 성능 저하가 나타남을 확인 할 수 있었다. 그러나 그룹화를 사용한 경우에는, 집합 I/O가 갖는 부하를 줄여 줄 수 있음을 살펴 볼 수 있었다.

그룹화 없이 MPI 집합 I/O를 사용할 경우 성능 저하가 급격히 나타났으나, Read와 Write모두에서 그룹화를 거쳐 집합 I/O를 수행할 경우, 성능 저하를 현저히 줄일 수 있슴을 확인할 수 있었다.

참고문헌

[1] Hakan Taki and Gil Utard, "MPI-IO on a parallel file system for cluster of workstations," Proc. 1st IEEE Computer Society International Workshop on Cluster Computing, 1999, pp 150~157.

[2] William Gropp, Ewing Lusk, and Rajeev Thakur, "Using MPI-2: Advanced Features of the Message Passing Interface," 1999, The MIT Press.

[3] David Kotz, "Disk-directed I/O for MIMD multiprocessors," ACM Transactions on Computer Systems, Vol. 15, No. 1, Feb 1997, pp 41~74.

[4] Avery Ching, Alok Choudhary, Wei-keng Liao, Rob Ross, and William Gropp, "Noncontiguous I/O through PVFS," Proc. IEEE International Conference on Cluster Computing, 2002, pp 405~414.

[5] MPI-2.0 Documents, http://www.mpi-forum.org/docs/docs.html .

[6] Philip H. Carns, Walter B. Ligon, III, Robert B. Ross, and Rajeev Thakur, "PVFS: A Parallel File System for Linux Clusters," Proc. 4th Annual Linux Showcase and Conference, 2000, pp 317~327.

[7] MPICH Web, http://www-unix.mcs.anl.gov/mpi/mpich/ .

[8] Rajesh Bordawekar, "Implementation of collective I/O in the Intel Paragon parallel file system: initial experiences," Proc. 11th international conference on Supercomputing, 1997, pp 20~27.