집합 I/O와 부분군 기법의 성능 분석

An Analysis of the Performance of Collective I/Os and the Subgroup Method

차광호, 조혜영, 김성호
한국과학기술정보연구원 슈퍼컴퓨팅센터

Kwangho Cha, Hyeyoung Cho, Sungho Kim
Korea Institute of Science and Technology Information

요약

많은 과학 분야 응용 어플리케이션들이 대규모 데이터 처리를 수행하면서, 병렬 I/O의 중요성이 급속도로 부각되고 있다. 특히 집합 I/O는 병렬 I/O의 중요한 개념 중 하나이며, 응용 프로그래머들이 쉽게 대용량 데이터를 취급할 수 있도록 해주고 있다. 본 연구에서는 원래의 집합 I/O들과 집합 I/O를 효과적으로 쓰기 위한 방법 중 하나인 부분군 기법의 성능을 측정하고 분석하였다. 실험 결과를 통하여 두 종류의 부분군 기법이 서로 다른 성능을 보임을 확인하였다. 집합 쓰기의 경우 부분군 기법은 성능저하를 나타냈으나 집합 읽기의 경우 적은 데이터를 사용하는 경우 우수한 성능을 보여 주고 있음을 확인하였다.

Abstract

Because many scientific applications require large data processing, the importance of parallel I/O has been increasingly recognized. Collective I/O is one of the considerable features of parallel I/O and enables application programmers to easily handle their large data volume. In this paper we measure and analyze the performance of original collective I/Os and the subgroup method, the way of using collective I/O of MPI effectively. From the experimental results, we found that the two kinds of subgroup method showed different performance. In terms of collective write operation, the subgroup method caused the performance degradation. However, the subgroup method for collective read showed good performance with small data size.

I. 서 론

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[1,2]. 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/O를 구현하는 방법들 중 대부분은 2단계 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.

프로세스 부분군 기법은 기존의 MPI함수들을 이용하여 MPI-IO를 보다 효과적으로 사용할 수 있도록 제시한 프로그래밍 모델이다[3]. 기존 연구가 기본 개념에 중점을 두고 있어, 본 연구에서는 부분군 기법의 오버헤드와 같은 제약 사항과 같은 한계점을 파악하는데 노력하였다. 클러스터 시스템에서의 성능 측정 결과, 쓰기에서는 성능 저하가 나타났으나 읽기의 경우, 기존의 MPI-IO함수만을 사용한 결과보다 개선된 성능을 보여주고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 집합 I/O에 대하여 살펴보고 3장에서는 부분군 기법의 주요 개념과 구조에 대하여 알아보며, 4장에서는 실험 결과, 5장에서는 결론 및 향후 계획에 대하여 기술한다.

II. 집합 I/O

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

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[4, 5]. 이처럼 복수의 프로세스가

수행하는 불연속적인 파일 참조를 지원하기 위한 방법 중 하나가 집합 I/O로서 이를 구현하기 위해 2단계I/O(2 Phase I/O)나 디스크 다이렉트 I/O(Disk Direct I/O)같은 방법이 제시되고 있다[4].

2. MPI-IO의 집합 I/O

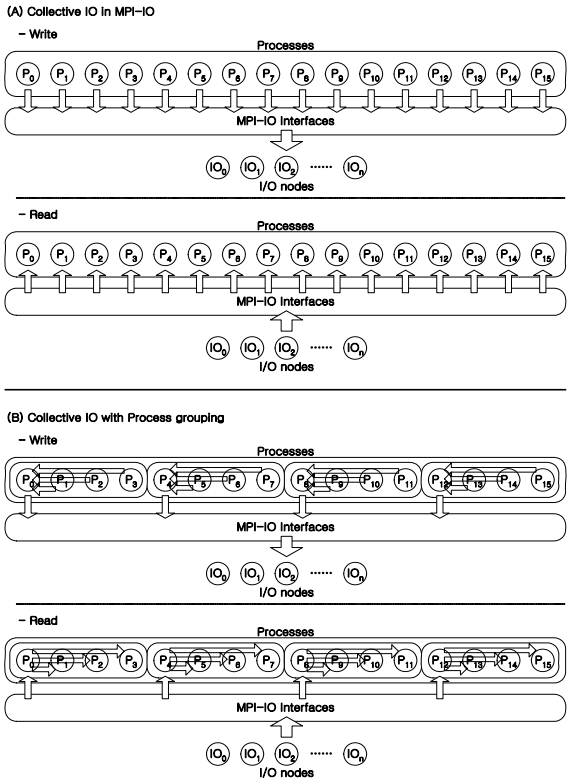
위에서 설명한 집합 I/O를 MPI 라이브러리에서도 지원하고 있다. MPI를 보강하여 MPI2 라이브러리가 발표되면서 추가된 기능으로는 원격 메모리 접근(RMA)과 MPI-IO가 있는데[2] MPI-IO의 기능 중, 일부로 집합 I/O가 제공되고 있다. 집합 I/O는 기존 MPI 라이브러리에서 제공되는 원시 자료형 또는 파생 자료형(Derived Data Type)를 이용하여 파일형을 생성한 뒤 파일에 접근하게 된다. [2,6].

III. 집합 I/O를 위한 부분군 기법

기존의 집합 I/O의 구현을 살펴보면, 집합 Write나 집합 Read같은 I/O작업을 수행하기 위해서는 모든 프로세스 간 I/O수행에 필요한 정보 조율이 필요하다. 이외에도 데이터를 I/O프로세스(IOP)로 보내야 하므로 한번의 집합 I/O를 수행할 때 여러 번의 전체 통신이 발생하게 된다[4,7].

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, 부분군 기법에서는 집합 I/O를 수행하는 프로세스의 수를 줄이고자 하는 시도를 하였다[3]. 즉, 부분군 기법은 응용 프로그램에서 집합 I/O를 수행할 시에, 프로세스들을 몇 개의 그룹으로 묶은 뒤 이들 그룹 당 1개의 프로세스만 집합 I/O를 수행토록 하는 서브 루틴의 일종이다.

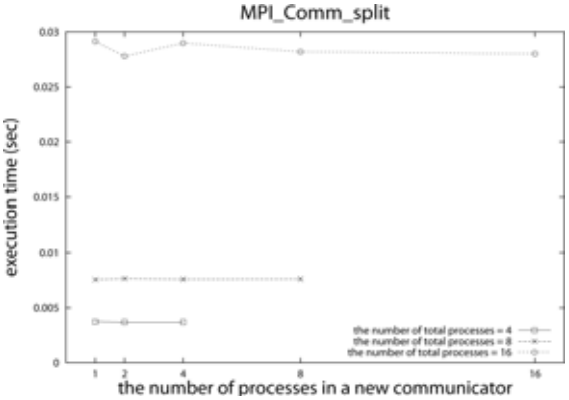
그림 1은 부분군 기법의 예를 보여주고 있다. 부분군 기법을 거치지 않은 경우에는 모든 프로세스가 집합 I/O를 수행하게 되나, 부분군 기법을 거친 경우에는 그룹화 계수(그림 1에서는 4)만큼의 프로세스들을 그룹으로 묶은뒤 집합 I/O를 수행하게 된다. 즉, 부분군 기법을 수행하는 경우에 각 프로세스들은 자신이 속한 그룹의 마스터 프로세스에게 자신의 데이터를 전달하고, 그후 마스터 프로세스는 자신이 속한 그룹의 데이터를 가지고 집합 I/O를 호출하는 구조이다.



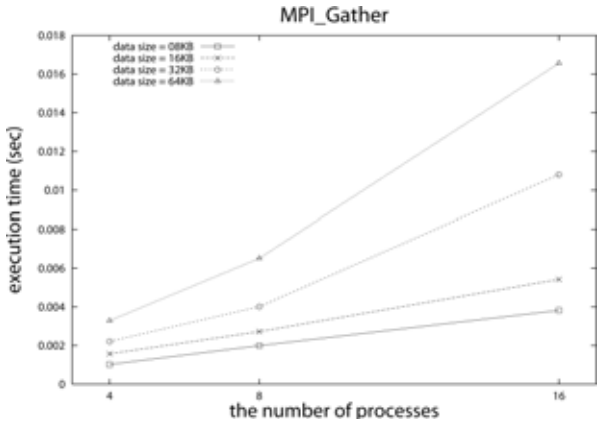
▶▶ 그림 1. 부분군 기법의 예: 프로세스 수=16, 그룹화 계수=4

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.

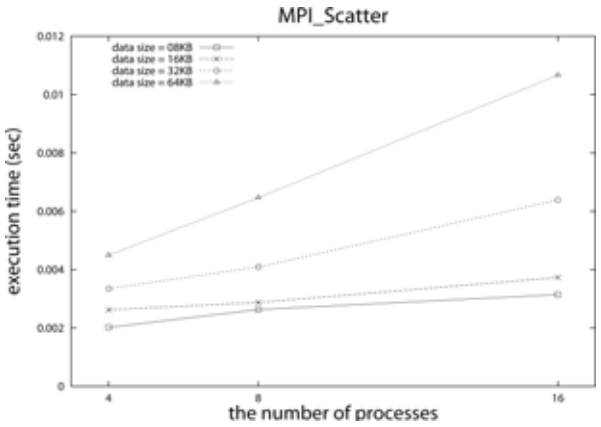
부분군 기법의 오버헤드 비용을 살펴보면, 우선 부분군을 생성하기 위해서는 기존의 커뮤니케이터이외에 새로운 커뮤니케이터를 생성하여야 한다. 그림 2는 이때 사용되는 MPI함수의 실행 시간은 측정한 결과이다. 새로운 커뮤니케이터에 속하는 프로세스의 수 보다는 최초 기본 프로세스 수에 비례하여 수행 시간이 결정되는 것을 확인할 수 있다.



▶▶ 그림 2. MPI_Comm_split()의 실행 비용



▶▶ 그림 3. MPI_Gather()의 실행 비용



▶▶ 그림 4. MPI_Scatter()의 실행 비용

두번째 오버헤드는 집합 I/O 전후에 수행되는 집합통신이다. 집합 쓰기를 위해서는 집합 I/O 이전에 MPI_Gather()를 필요로 하며 집합 읽기는 집합 I/O 이후에 MPI_Scatter()를 필요로 한다. 그림 3, 4는 이 두 함수의 실행 시간을 보여주고 있는데, 상대적으로 MPI_Gather()가 더 오래 걸림을 확인할 수 있다.

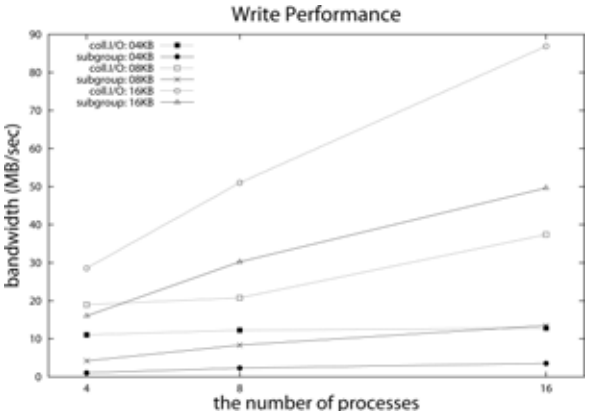
IV. 성능 평가

부분군 기법과 집합 IO의 전반적인 성능을 확인하기위하여 표 1과 같은 테스트베드 환경에서 실험을 진행하였다. 병렬 파일 시스템으로는 PVFS, MPI라이브러리로는 MPICH2가 각각 사용되었다[8,9].

[표 1] 테스트 베드 환경

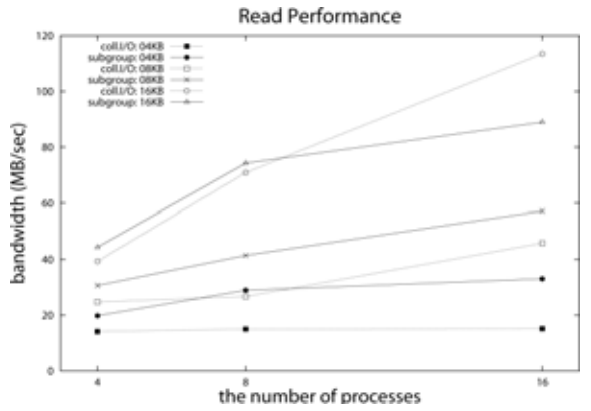
CPU	AMD Opteron 240 (1.4Ghz)
Network	Gigabit Ethernet
OS	Linux 2.6.9
File System	PVFS 2.6.1
MPI Library	MPICH2-1.0.5

그림 5에서 보는 바와 같이 집합 쓰기의 경우, 부분군 기법이 오히려 좋지 않은 결과만을 보여 주었다. 이는 이전 장에서 언급한 바와 같이 매 집합 IO 수행시 호출되는 추가적인 집합통신 함수 중 MPI_Scatter()와는 비교되는 성능을 보인 MPI_Gather()의 영향을 받은 것으로 보인다.



▶▶ 그림 5. 집합 쓰기 성능 비교

반면 그림 6에서 보는 바와 같이 집합 읽기의 경우, 사용되는 데이터의 크기가 작은 경우, 부분군 기법이 월등히 좋은 성능을 보여줌을 확인할 수 있었다.



▶▶ 그림 6. 집합 읽기 성능 비교

V. 결 론

부분군 기법과 집합 IO에 대하여 전반적인 성능 평가를 진행하였다. 집합 쓰기의 경우, 일반적인 집합 IO가 보다 좋은 성능을 보인 반면, 집합 읽기의 경우에는 부분군 기법이 데이터 사이즈가 작은 경우 월등히 좋은 성능을 보여 주었다. 이에 읽기 위주의 fine-grained 응용 프로그램에 부분군 기법이 유용할 것으로 예상된다.

▌참 고 문 헌▐

[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 Clus-ter 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] Kwangho Cha, Taeyoung Hong, and Jeongwoo Hong, "The Subgroup Method for Collective I/O," Proc. The 5th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2004), LNCS 3320, pp. 301~304, Dec. 2004.

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

[5] 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.