두 단계 프로토콜 : PVFS를 위한 상호 협력 캐쉬에서 쓰기 성능 향상 기법

황인철° 정한조 맹승렬 조정완
한국과학기술원 전자전산학과 전산학전공
{ichwang°, hanjo, maeng, jwcho}@calab.kaist.ac.kr

Two-Phase Protocol : Write Performance Enhancement Scheme of the Cooperative Cache for PVFS

In-chul Hwang°, Hanjo Jung, Seung-Ryoul Maeng, Jung-Wan Cho
Division of Computer Science, Dept. of Electrical Engineering & Computer Science, KAIST

요 약

요즘 값싼 PC들을 빠른 네트웍으로 묶어 높은 성능을 얻고자하는 클러스터 컴퓨팅에 대한 연구가 활발히 이루어지면서 CPU나 메모리, 네트웍보다 상대적으로 느린 디스크에서 데이터를 읽어 효율적으로 파일 서비스를 하는 분산 파일 시스템이 개발되었다.

기존 분산 파일 시스템 중 클러스터 컴퓨팅에서 많이 사용하는 Linux 운영 체제에서 병렬 I/O를 사용하여 사용자에게 빠른 파일 서비스를 제공하여 주는 PVFS가 개발되었다. 기존 PVFS에서는 캐쉬 시스템을 제공하고 있지 않기 때문에 읽기 성능을 향상시키기 위하여 PVFS를 위한 상호 협력 캐쉬를 설계하고 구현하였다.

PVFS를 위한 상호 협력 캐쉬는 클라이언트의 파일 캐쉬를 공유하여 파일 요구를 처리하는 기법으로 읽기 성능은 크게 향상되었다. 하지만 쓰기의 경우에는 다른 클라이언트에서 가지고 있던 모든 데이터를 찾아 해제하는 부하가 있기 때문에 성능이 좋지 않다. 따라서 본 논문에서는 PVFS를 위한 상호 협력 캐쉬에서 쓰기 성능 향상 기법인 두 단계 프로토콜을 제시하고 구현한다. 그리고 두 단계 프로토콜을 기존 PVFS와 PVFS를 위한 상호 협력 캐쉬 시스템과 성능을 비교, 분석한다.

1. 서론

요즘 값싼 PC들을 빠른 네트웍으로 묶어 높은 성능을 얻고자 하는 클러스터 컴퓨팅에 대한 연구가 활발히 이루어지고 있다. 클러스터 컴퓨팅을 효율적으로 하기 위해서는 빠른 네트웍의 구성 및 효율적인 운영체제의 서비스가 필요하다. 이러한 연구가 이루어지면서 CPU나 메모리, 네트웍보다 상대적으로 느린 디스크에서 데이터를 읽어 효율적으로 파일 서비스를 하는 분산 파일 시스템이 개발되었다.

기존 분산 파일 시스템 중 클러스터 컴퓨팅에서 많이 사용되는 Linux 운영체제에서 병렬 I/O를 이용하여 사용자에게 파일 서비스를 제공하여 주는 PVFS(Parallel Virtual File System) [1,2]가 개발되었다. PVFS에서는 파일 데이터를 분산시켜 병렬 I/O를 이용하여 높은 대역폭을 제공하여 준다. 그러나 PVFS는 어떤 캐싱 기법도 사용하지 않고 단순히 사용자에게 데이터를 전달하여 준다. 이를 해결하기 위하여 PVFS를 위한 상호 협력 캐쉬 [3]를 설계하고 구현하였다.

PVFS를 위한 상호 협력 캐쉬는 클라이언트들의 파일 캐쉬를 공유하여 파일 읽기 요구를 처리하는 기법으로 서버 대신 다른 클라이언트 캐쉬로부터 블록을 읽어 오기 때문에 읽기 성능을 크게 향상시킨다. 하지만 쓰기의 경우 다른 클라이언트에서 캐싱하고 있던 모든 블록을 해제하는 추가적인 부하가 있기 때문에 성능이 좋지 않다. 본 논문에서는 PVFS를 위한 상호 협력 캐쉬에서 이러한 비효율적인 쓰기 성능을 향상시키는 기법인 두 단계 프로토콜을 제시하고 구현한다. 그리고 기존 PVFS와 PVFS를 위한 상호 협력 캐쉬와 성능을 비교 평가한다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구로서 PVFS와 PVFS를 위한 상호 협력 캐쉬에 대해서 살펴본다. 3장에서는 PVFS를 위한 상호 협력 캐쉬에서 쓰기 향상 기법인 두 단계 프로토콜을 제시하고 구현에 대하여 설명한다. 4장에서는 기존 PVFS, PVFS를 위한 상호 협력 캐쉬 그리고 구현한 두 단계 프로토콜의 성능을 비교 분석한다. 5장에서는 향후 연구 방향과 결론을 맺는다.

2. 관련 연구

2.1 PVFS(Parallel Virtual File System)

PVFS [1,2]는 Linux 운영체제에서 병렬 I/O를 제공하기 위하여 Clemson University에서 개발한 병렬 파일 시스템이다.

PVFS는 크게 계산 노드, 관리자 노드, 그리고 I/O 서버로 구성된다.

PVFS에서 사용자 접근 방법은 크게 두 가지로 나뉜다. 첫 번째 방법은 사용자 라이브러리를 이용한 방법으로 새로 정의된 사용자 라이브러리를 이용하여 프로그램을 재구성하는 방법이다. 이 방법은 사용자가 접근 패턴과 같은 힌트를 주어 성능 향상을 할 수 있다는 장점이 있지만 사용자가 새로 프로그램을 구성해야 한다는 단점이 있다. 다른 한 가지 방법은 클라이언트와 PVFS 커널 모듈을 사용하여 기존 UNIX I/O API를 사용하여 접근하는 방법이다. 이 방법은 기존 프로그램의 변경 없이 병렬 I/O를 사용할 수 있다는 장점이 있지만 사용자로부터 힌트를 얻어 성능 향상을 할 수 없다는 단점이 있다.

2.2 PVFS를 위한 상호 협력 캐쉬

기존 PVFS에서는 파일 캐싱을 제공하지 않는다. 이러한 문제점을 해결하기 위하여 기존에 Vilaynannur [4]는 PVFS에서 사용자 라이브러리를 이용한 방법에서 PVFS를 위한 단일 노드 캐쉬 시스템을 구현하였다. 그러나 실제 클러스터 컴퓨팅에서는 여러 사용자가 여러 노드에서 파일을 공유하기 때문에 효율적인 파일 서비스를 위해서는 상호 협력 캐쉬 [5,6,7]가 더 효율적이다. 이를 위하여 PVFS 커널 모듈을 이용한 방법에서 PVFS를 위한 상호 협력 캐쉬를 구현하였다 [3]. PVFS를 위한 상호 협력 캐쉬는 힌트를 기반으로 하는 상호 협력 캐쉬로서 블록을 단위로 캐쉬를 관리하며 기존 Linux 운영체제의 버퍼 캐쉬 시스템을 사용하지 않고 구현되었다. PVFS를 위한 상호 협력 캐쉬에서는 읽기의 경우 다른 노드에서 블록을 캐싱하고 있을 경우 그 내용을 디스크가 아닌 다른 노드의 캐쉬로부터 읽어오기 때문에 읽기 성능이 좋아진다. 하지만 쓰기의 경우

상호 협력 캐쉬에 있는 모든 캐쉬 블록을 해제시키는 부하 때문에 쓰기 성능은 나빠진다.

3. PVFS를 위한 상호 협력 캐쉬에서 쓰기 향상 기법

3.1 기존 PVFS를 위한 상호 협력 캐쉬에서 쓰기 방법

다음 그림 1은 기존 PVFS를 위한 상호 협력 캐쉬에서 쓰기 기법에 대해 나타낸 그림이다.

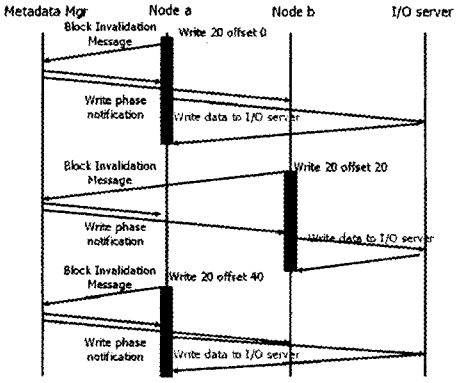

그림 1. PVFS를 위한 상호 협력 캐쉬에서 쓰기 기법

그림에서와 같이 모든 쓰기에 대해서 다른 클라이언트의 캐쉬에 있는 블록을 해제시켜야 한다. 따라서 다른 클라이언트의 캐쉬에 자신이 쓸 블록이 없을 경우에도 메타데이터 관리자에게 캐쉬 블록 해제 요청을 하게 된다. 그리고 같은 블록을 이미 해제 요청을 통해 해제를 하였어도 중간이 이미 다른 클라이언트가 블록을 다시 읽어갈 수 있기 때문에 다시 블록 해제 요청을 하여야 한다. 이러한 쓰기 방법은 PVFS를 위한 상호 협력 캐쉬가 힌트 기반 상호 협력 캐쉬로 구현되어 블록에 대한 정확한 읽기/쓰기 정보 관리를 하지 않기 때문에 비효율적인 쓰기를 수행할 수밖에 없다. 따라서 본 논문에서는 이러한 쓰기의 비효율성을 해결할 방법으로 두 단계 프로토콜을 제안한다.

3.2 두 단계 프로토콜 : PVFS를 위한 상호 협력 캐쉬에서 쓰기 향상 기법

두 단계 프로토콜은 기본적으로 모든 프로그램의 파일 접근이 읽기/쓰기 단계를 가지고서 접근한다는 가정에 기반을 두어 제시한다. 파일 접근의 단계를 파일단위로 읽기/쓰기 단계로 나누고 쓰기 단계에는 파일 단위의 관리를 수행하여 캐싱하고 있던 모든 파일 블록을 모두 해제시킨다. 따라서 한 파일을 동시에 접근하는 프로그램들은 쓰기가 쓰기 단계에서 여러 번 이루어 질 때 매번 메타데이터 관리자에게 블록 해제 요청을 하는 기존 PVFS를 위한 상호 협력 캐쉬와 달리 쓰기 단계일 때 쓰기는 블록 해제 요청을 수행하지 않는다. 그리고 읽기 단계에서는 블록단위의 읽기를 수행함으로써 읽기 성능의 저하 없이 쓰기 성능을 향상시킬 수 있다. 이를 위하여 두 단계가 변경됨에 따라 다음과 같은 동작을 수행한다.

- 파일이 처음 열렸을 때 : 읽기 단계로 단계를 정의한다.
- 읽기 단계에서 쓰기 단계로의 전이될 때 : 파일 단계가 읽기이고 쓰기가 수행될 때 이와 같은 상황이 발생한다. 이때 클라이언트는 메타데이터 관리자에게 전이가 일어났음을 알리면 메타데이터 관리자는 파일을 열고 있는 모든 클라이언트에게 파일이 읽기 단계에서 쓰기 단계로 전이되었음을 알리게 되고 이 메시지를 받은 모든 클라이언트는 캐싱하고 있던 모든 파일 블록을 해제시킨다.
- 쓰기 단계에서 읽기 단계로의 전이될 때 : 파일이 쓰기단계이며 읽기가 수행되었을 때 이와 같은 상황이 발생된다. 이때 클라이언트는 메타데이터 관리자에게 상태의 전이를 알리고 메타데이터 관리자는 파일을 열고 있는 모든 클라이언트에게 파일이 쓰기 단계에서 읽기 단계로 전이되었음을 알린다.

다음 그림 2는 두 단계 프로토콜이 연속적인 쓰기를 기존 PVFS를 위한 상호 협력 캐쉬에서 어떻게 향상시켰는지 보여준다.

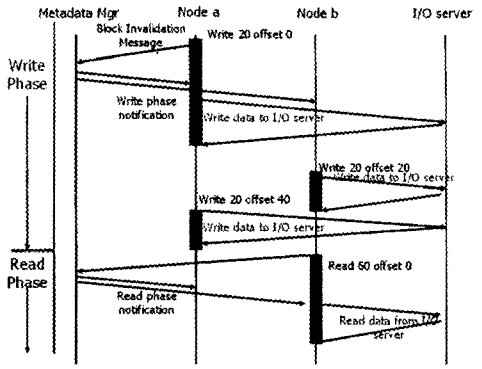

그림 2. 두 단계 프로토콜에서 쓰기/읽기

기존 PVFS를 위한 상호협력 캐쉬에서는 쓰기가 연속적으로 일어날 경우에도 블록 해제 요청을 메타데이터 관리자에게 보냄으로서 메시지 수가 증가하고 쓰기의 처리 속도도 감소하게 된다. 두 단계 프로토콜은 쓰기 단계에서 쓰기가 일어날 경우는 기존 PVFS에서 쓰기와 같이 추가적 부하 없이 I/O서버에게 쓰기 내용을 전달함으로서 쓰기를 처리하게 된다.

4. 성능 평가

성능 평가를 위해 사용된 환경은 다음 표 1과 같다.

CPU	Pentium IV 1.8GHz
Memory	512MByte 266MHz DDR Memory
Disk	IBM 60G 7200rpm
Network	3c996B-T(Gigabit Ethernet) 3c17701-ME (24port Gigabit Ethernet Switch)
OS	Linux 2.4.18
PVFS	Version 1.5.3

표 1. 성능 평가 환경

메타데이터 관리자와 I/O 서버는 각각 하나의 노드에 할당하였고 네 개의 노드에 클라이언트를 할당하여 다음 프로그램들을 PVFS와 기존 PVFS를 위한 상호 협력 캐쉬, 그리고 두 단계 프로토콜을 사용하는 PVFS를 위한 상호 협력 캐쉬에서 각각 수행하였다.

- 행렬 곱셈 프로그램: 파일에 저장되어 있는 1024*1024 행렬 두개를 읽은 후 곱하여 그 결과를 파일에 적는 프로그램을 수행한다. 네 개의 클라이언트에서 MPI로 수행되도록 하였다.
- BTIO 프로그램:BTIO [8]은 병렬 파일 시스템 벤치마크 프로그램으로 NAS 병렬 벤치마크의 하나이다. BTIO에는 네 가지 종류의 프로그램이 있고 표 2는 각 프로그램에 대한 설명이다. 데이터 s class에 대하여 네 가지 프로그램을 수행하여 각 성능을 측정하였다.

4.1 행렬 곱셈 프로그램에서의 성능

다음 그림 3은 행렬 곱셈 프로그램에서 수행 시간 중 기존 PVFS를 위한 상호 협력 캐쉬에서 읽기/쓰기 시간을 기준으로 읽기/쓰기 시간을 상대적으로 나타낸 그래프이다.

Full	Collective buffering을 사용한 MPI-IO를 사용하는 프로그램
Simple	Collective buffering을 제외한 MPI-IO를 사용하는 프로그램
Fortran	Fortran 77 파일 입출력을 사용하는 프로그램
Epio	각 프로세서가 각각의 부분을 적는 프로그램

표 2. BTIO 프로그램

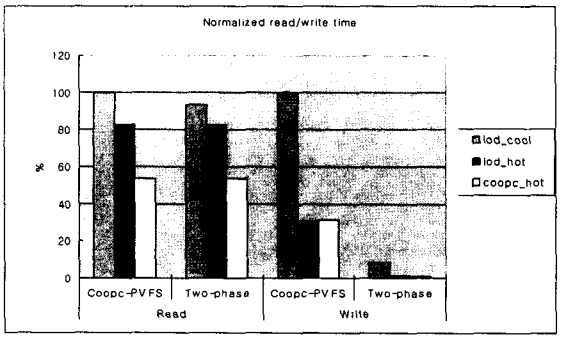

그림 3. 행렬 곱셈 프로그램에서 상대적 읽기/쓰기 시간

행렬 쓰기 프로그램은 읽기 위주의 프로그램으로 기존 PVFS를 이용할 경우 읽기는 약 28초의 시간이 걸린다. 이에 반하여 PVFS를 위한 상호 협력 캐쉬나 두 단계 프로토콜을 사용할 경우 읽기 시간을 0.5초로 줄인다. 그림에서와 같이 읽기의 경우 기존 PVFS를 위한 상호 협력 캐쉬와 두 단계 프로토콜의 차이가 거의 없으나, 쓰기의 경우 두 단계 프로토콜은 쓰기-쓰기의 부하를 줄임으로서 약 10%정도의 시간만으로 쓰기 요구를 처리할 수 있다.

4.2 BTIO 프로그램에서의 성능

다음 그림 4는 BTIO 프로그램에서의 성능을 나타낸다.

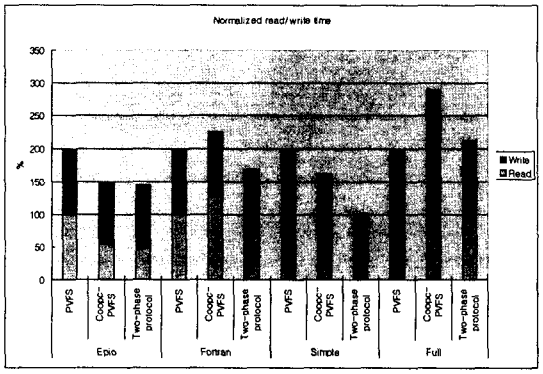

그림 4. BTIO에서의 상대적 읽기/쓰기 시간

읽기의 경우 PVFS를 위한 상호 협력 캐쉬와 두 단계 프로토콜을 사용하는 것은 기존 PVFS의 읽기 성능을 향상시킨다. 하지만 읽기/쓰기가 혼용되어 자주 사용되는 경우(Fortan)에는 다른 노드에 읽을 블록이 없음에도 불구하고 다른 노드에서 읽어본 후 서버에게 요청하기 때문에 PVFS를 위한 상호 협력 캐쉬에서는 읽기 시간이 많이 걸린다. 하지만 본 논문에서 제안한 두 단계 프로토콜을 사용할 경우 쓰기 후 읽기의 경우 정확한 블록의 위치를 알 가능성이 높기 때문에 읽기 시간이 줄어든다. 쓰기의 경우 기존 PVFS가 가장 좋은 성능을 나타낸다. 이는 어떤 부하도 갖지 않고 바로 서버에게 쓴 내용을 전달하기 때문이다. 두 단계 프로토콜에서도 PVFS보다 좋지 않은 성능이지만 PVFS를 위한 상호 협력 캐쉬보다 블록 해제의 부하가 적기 때문에 더 좋은 읽기 성능을 나타낸다.

5. 향후 연구 방향 및 결론

값싼 PC를 빠른 네트웍으로 묶어 높은 성능을 내고자 하는 클러스터 컴퓨팅에서 CPU나 메모리, 네트웍보다 느린 디스크에 접근하는 효율적인 파일 시스템의 구성이 필요하다. PVFS는 클러스터 컴퓨팅에서 많이 사용되는 운영체제인 Linux에서 병렬 I/O를 지원하여 높은 대역폭을 제공해 주는 파일 시스템이다. PVFS에서는 파일 캐쉬를 제공하지 않고 이를 위하여 기존에 PVFS를 위한 상호 협력 캐쉬를 설계하고 구현하였다.

PVFS를 위한 상호 협력 캐쉬는 읽기 요구에 대하여 그 내용을 서버가 아닌 다른 노드의 캐쉬로부터 읽어옴으로써 읽기 성능을 향상시킨다. 그러나 PVFS를 위한 상호 협력 캐쉬는 힌트 기반 상호 협력 캐쉬를 기반으로 구현되어 쓰기 요청마다 다른 노드에 캐슁된 블록을 해제시키는 부하가 있다는 단점이 있다.

본 논문에서는 PVFS를 위한 상호 협력 캐쉬에서 쓰기 성능을 향상시키기 위한 기법으로 두 단계 프로토콜을 제시하고 구현하였다. 행렬 곱셈 프로그램과 BTIO 벤치마크를 이용하여 읽기/쓰기 수행시간을 측정하여 본 결과 기존 PVFS와 PVFS를 위한 상호 협력 캐쉬보다 두 단계 프로토콜이 읽기/쓰기 시간이 줄어들었다.

앞으로는 두 단계 프로토콜에서 쓰기 요청 때 파일단위 관리를 효율적으로 할 수 있는 방법에 대한 연구와 쓰기 성능을 향상시킬 수 있는 쓰기 버퍼링 방법에 대해 연구할 예정이다.

6. 참고 문헌

[1] P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur, "PVFS: A Parallel File System For Linux Clusters", Proceedings of the 4th Annual Linux Showcase and Conference, Atlanta, GA, October 2000, pp. 317-327

[2] R.B.Ross,"Providing Parallel I/O on Linux Clusters" ,Second Annual Linux Storage Management Workshop, Miami, FL, October 2000.

[3] 황인철, 정한조 외, "PVFS를 위한 상호 협력 캐쉬의 설계 및 구현", 한국정보과학회 2003년도 춘계 학술대회 발표 논문집, 2003년 5월

[4] M.Vilayannur,M.Kandemir, A.Sivasubramaniam, "Kernel-Level Caching for Optimizing I/O by Exploiting Inter-Application Data Sharing", IEEE International Conference on Cluster Computing (CLUSTER'02),September 2002

[5] Dahlin, M., Wang, R., Anderson, T., and Patterson, D. 1994. "Cooperative Caching: Using remote client memory to improve file system performance", In Proceedings of the First USENIX Symposium on Operating Systems Design and Implemntation. USENIX Assoc., Berkeley, CA, 267-280

[6] Feeley, M. J., Morgan, W. E., Pighin, F. H., Karlin, A. R., and Levy, H. M. 1995. "Implementing global memory management in a workstation cluster", In Proceedings of the 15th symposium on Operating System Principles(SOSP). ACM Press, New york, NY, 201-212

[7] Prasenjit Sarkar , John Hartman, "Efficient cooperative caching using hints", Proceedings of the second USENIX symposium on Operating systems design and implementation, p.35-46, October 29-November 01, 1996, Seattle, Washington, United States

[8] Parkson Wong, Rob F. Van der Wijngaart, NAS Parallel Benchmark I/O Version 2.4, NAS Technical Report NAS-03-002, NASA Ames Research Center, Moffett Field, CA 94035-1000