PVFS를 위한 홈 기반 상호 협력 캐쉬의 설계 및 구현

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

Design and Implementation of a Home-based 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.

기존에 구현된 PVFS를 위한 상호 협력 캐쉬는 힌트 기반 상호 협력 캐쉬로서 부정확한 읽기/쓰기를 수행함으로서 읽기/쓰기 부하가 커지는 단점이 있다. 따라서 본 논문에서는 기존 PVFS를 위한 상호 협력 캐쉬의 읽기/쓰기 성능 향상을 위해 PVFS를 위한 상호 협력 캐쉬를 홈 기반 상호 협력 캐쉬로서 설계 및 구현한다. 그리고 PVFS, 기존 PVFS를 위한 힌트 기반 상호 협력 캐쉬와 PVFS를 위한 홈 기반 상호 협력 캐쉬의 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 연구들 중 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 높은 대역폭을 제공하여 주는 PVFS(Parallel Virtual File System)[1,2]가 개발되었다. PVFS에서는 어떤 캐싱 기법도 사용하지 않고 단순히 사용자에게 데이터를 전달하여 준다. 이러한 문제점을 해결하기 위하여 PVFS를 위한 상호 협력 캐쉬[3]를 설계하고 구현하였다.

기존 PVFS를 위한 상호 협력 캐쉬는 힌트 기반 상호 협력 캐쉬로서 클라이언트들의 파일 캐쉬를 공유하여 파일 읽기 요구를 처리하는 기법으로 힌트를 이용하여 읽기/쓰기를 수행하기 때문에 힌트가 부정확 할 경우 읽기/쓰기의 부하가 많다는 단점이 있다. 이러한 문제를 해결하기 위하여 PVFS를 위한 상호 협력 캐쉬를 홈 기반 상호 협력 캐쉬로 설계하고 구현한다. 그리고 PVFS, 기존 PVFS를 위한 상호 협력 캐쉬와 홈 기반 상호 협력 캐쉬의 성능을 비교 평가한다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구로서 PVFS와 기존 PVFS를 위한 상호 협력 캐쉬에 대해서 살펴본다. 3장에서는 PVFS를 위한 홈 기반 상호 협력 캐쉬의 설계 및 구현에 대하여 설명한다. 4장에서는 PVFS, 기존 PVFS를 위한 상호 협력 캐쉬 그리고 홈 기반 상호 협력 캐쉬의 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 대학에서 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 구현하고 그 효율성을 보여주었다. 하지만 실제 클러스터 컴퓨팅에서는 여러 사용자가 여러 노드에서 파일을 공유하기 때문에 효율적인 파일 서비스를 위해서는 상호 협력 캐쉬[5,6,7]가 필요하다. 따라서 기존에 우리는 PVFS 커널 모듈을 이용한 방법에서 PVFS를 위한 상호 협력 캐쉬를 구현하였다[3].

기존 PVFS를 위한 상호 협력 캐쉬는 힌트를 기반으로 하는 상호 협력 캐쉬로서 블록을 단위로 캐쉬를 관리하며 각각의 힌트-기존에 파일을 열었던 노드 IP 주소-를 이용하여 클라이언트간 자신이 캐싱한 블록의 정보를 교환함으로서 상호 협력 캐쉬에 존재하는 블록의 정보를 유지한다. 기존 PVFS를 위한 상호 협력 캐쉬는 힌트를 이용하여 정확하지 않은 캐싱 정보를 유지하기 때문에 읽기/쓰기의 경우 정보의 정확도가 떨어지면 읽기/쓰기를 수행할 때 큰 부하가 있다는 단점이 있다. 또한 쓰기의 경우 매번 쓰기 요청이 일어날 때마다 블록 해제 요청을 블록을 가지고 있을만한 모든 노드에게 요청하는 부하가 존재하고 이를 해결하기 위하여 기존에 두 단계 프로토콜[9]을 제안하였다. 하지만 두 단계 프로토콜은 기본적으로 PVFS를 위한 힌트 기반 상호 협력 캐쉬의 쓰기 부하만 줄였기 때문에 PVFS의 쓰기 성능보다 좋지 않다.

3. PVFS를 위한 홈 기반 상호 협력 캐쉬의 설계 및 구현

3.1 PVFS를 위한 홈 기반 상호 협력 캐쉬의 설계

PVFS를 위한 홈 기반 상호 협력 캐쉬는 블록마다 홈을 설정하여 홈에게 그 블록에 대한 읽기/쓰기 요구를 처리하도록 한다. 그리고 홈 설정에 대한 역할을 확장성을 위하여 파일 단위로 홈 설정 서버를 결정하고 여러 노드에 그 부하를 분산시킨다. 파일 단위 홈 설정 서버는 처음 시스템에서 그 파일을 열 경우 설정하며 그 내용을 메타데이터 관리자가 가지고 있음으로서 차후 클라이언트가 파일을 열 경우 그 파일 단위 홈 서버를 알려줄 수 있다.

다음 그림 1은 파일 블록에 대한 읽기 요구가 처리되는 모습을 나타낸 그림이다.

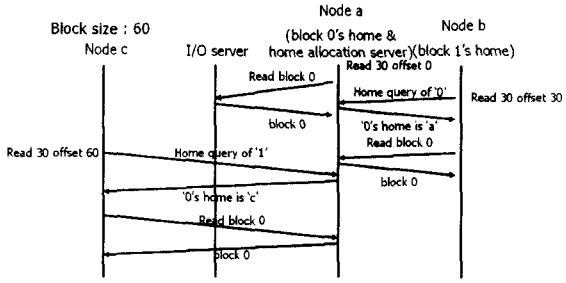

그림 1. 홈 기반 상호 협력 캐쉬에서 읽기 처리

파일 블록에 대한 읽기 요구가 되었을 경우 먼저 자신의 캐쉬 관리자에게 캐슁이 되어있는지 살펴본 후 캐슁되어 있으면 읽기 요구가 완료된다. 만약 그 내용이 캐슁되어 있지 않을 경우 블록의 홈 노드를 찾아보고 홈 노드가 설정되어있지 않으면 홈 노드 설정 서버에게 블록의 홈 노드를 가져온다. 그리고 난 후, 블록의 홈 노드의 캐쉬 관리자에게 블록을 요구하여 가져와 자신이 캐슁한 후 읽기 요구 수행을 완료한다. 블록의 홈 노드의 캐쉬 관리자는 블록의 요구를 받았을 경우 자신이 블록을 캐슁하고 있으면 블록을 요구한 노드의 캐쉬 매니저에게 그 내용을 전달하여 주고, 만약 자신이 캐슁하고 있지 않았을 경우에는 자신이 직접 I/O 서버에게서 읽어 와서 자신이 캐슁한 후 그 내용을 전달하여 준다. 그리고 블록의 홈 노드의 캐쉬 관리자는 블록을 읽어간 노드의 주소를 기록하여 나중에 쓰기 요청이 있을 때 블록을 해제하는 요청을 수행할 수 있도록 한다. 이와 같이 힌트 기반 상호 협력 캐쉬에 비하여 블록이 존재할 가능성이 높은 홈을 설정함으로서 읽기 성능을 높일 수 있다.

다음 그림 2는 파일 블록에 대한 쓰기 요구가 처리되는 모습을 나타낸 그림이다.

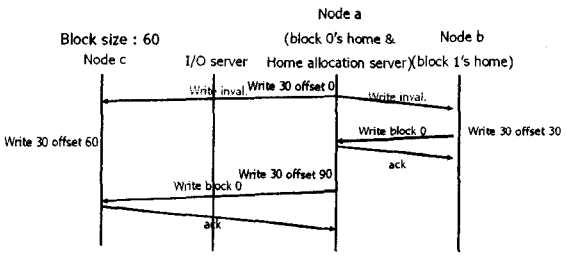

그림 2. 홈 기반 상호 협력 캐쉬에서 쓰기 요구 처리

파일 블록에 대한 쓰기 요구가 되었을 경우 노드의 캐쉬 관리자는 블록의 홈이 자신인지 아니면 다른 노드의 캐쉬 관리자인지 살펴본다. 만약 블록의 홈이 설정되어 있지 않다면 블록 할당 서버에게 요청하여 블록의 홈을 가져온다.

블록의 홈이 자신일 경우 기존에 그 블록을 읽어갔던 노드의 캐쉬 관리자에게 블록 해제 요청을 하고 자신의 캐슁하고 있던 블록에 쓰기를 수행하여 쓰기 버퍼링을 수행한다. 만약 블록의 홈이 다른 노드일 경우 쓰기 블록을 블록의 홈 노드의 캐쉬 관리자에게 그 내용을 전달하여 주고, 쓰기 블록의 내용을 받은 캐쉬 관리자는 그 블록을 읽어갔던 다른 노드에게 블록 해제 요청을 하고 자신이 캐슁하고 있던 블록에 쓰기를 수행함으로서 쓰기 버퍼링을 수행한다. 이와 같이 PVFS를 위한 홈 기반 상호 협력 캐쉬에서는 쓰기 버퍼링을 수행함으로서 쓰기의 수행 속도를 높일 수 있을 뿐 아니라 여러 노드에서 수행되는 쓰기를 하나의 블록에 모아 I/O 서버에 적기 때문에 I/O서버의 부하를 줄여줄 수 있는 장점을 지닌다.

3.2 PVFS를 위한 홈 기반 상호 협력 캐쉬의 구현

PVFS를 위한 홈 기반 상호 협력 캐쉬는 기존 PVFS를 위한 힌트 기반 상호 협력 캐쉬와 유사하게 구성되어 구현되었다. 캐쉬 관리자는 기본적으로 리눅스 페이지 캐쉬 시스템을 사용하지 않기 때문에 캐슁 블록을 시스템 메모리의 양에 따라 관리하는 캐쉬 대체 관리자가 존재하고 이를 위하여 LRU 리스트를 관리한다. 그리고 파일 단위 홈 서버를 위하여 블록 홈 리스트를 두어 블록의 홈을 저장한다.

PVFS를 위한 홈 기반 상호 협력 캐쉬는 처음 파일을 읽어올 때 메타데이터 관리자로부터 파일의 메타데이터와 파일 단위의 블록 홈 설정 서버 정보를 가져와서 추후 이 파일 블록들의 홈 정보를 가져오게 된다. 이를 위하여 메타데이터 관리자는 파일 단위의 홈 서버 노드의 정보만 유지하고 있으면 된다. 그리고 이 정보를 이용하여 읽기/쓰기를 처리하게 된다.

PVFS를 위한 홈 기반 상호 협력 캐쉬는 기존 PVFS를 위한 힌트 기반 상호 협력 캐쉬와 달리 자신이 캐슁하고 있는 블록의 정보를 가지고 있을 필요가 없고 다른 노드와 이 정보를 교환할 필요 없이 홈 노드의 정보만 가지고서 읽기/쓰기를 수행한다. 그리고 자신이 홈인 블록들 중 쓰기가 이루어진 더티 블록을 연결하는 리스트를 가지고 있음으로서 차후 써진 캐쉬 블록 대체 쓰기 요청을 받았을 때 효율적으로 I/O 서버에 전달할 수 있다.

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 1.6.0

표 1. 성능 평가 환경

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

4. 성능 평가

성능 평가를 위해 사용된 환경은 다음 표 1과 같다.

메타데이터 관리자와 I/O 서버는 각각 하나의 노드에 할당하였고 네 개의 노드에 클라이언트를 할당하여 다음 프로그램들을 PVFS와 기존 PVFS를 위한 힌트 기반 상호 협력 캐쉬, 그리고 PVFS를 위한 홈 기반 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.

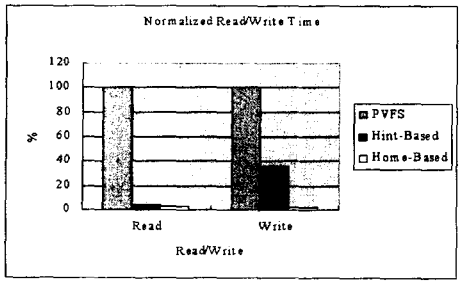

그림 3. 행렬 곱셈 프로그램에서의 읽기/쓰기 시간

행렬 쓰기 프로그램은 읽기 위주의 프로그램으로 기존 PVFS를 이용할 경우 읽기가 약 12초의 시간이 걸린다. 이에 반하여 PVFS를 위한 힌트 기반 상호 협력 캐쉬와 홈 기반 상호 협력 캐쉬는 읽기 시간을 0.5초와 0.3초로 줄인다. 쓰기의 경우 홈 기반 상호 협력 캐쉬는 모든 쓰기를 자신의 메모리에 버퍼일 하는 경우이므로 가장 좋은 성능을 나타내고 힌트 기반 상호 협력 캐쉬의 경우 PVFS보다 약 30%정도의 시간으로 쓰기를 수행 할 수 있다.

4.2 BTIO 프로그램에서의 성능

다음 그림 4는 BTIO 프로그램에서의 성능을 나타낸다.

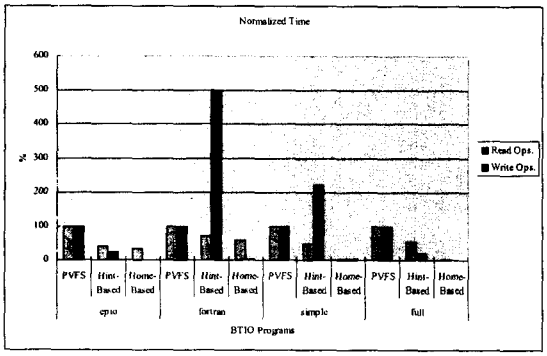

그림 4. BTIO 프로그램에서 상대적인 읽기/쓰기 시간

읽기의 경우 PVFS를 위한 힌트 기반 상호 협력 캐쉬와 홈 기반 상호 협력 캐쉬를 기존 PVFS에 비해 성능이 더 좋음을 알 수 있다. 하지만 읽기/쓰기가 혼용되어 자주 사용되는 경우(fortran, simple)에는 힌트 기반 상호 협력 캐쉬는 높은 쓰기 부하로 PVFS보다 약 2~5배정도의 시간이 더 걸린다. 이에 비하여 본 논문에서 구현한 홈 기반 상호 협력 캐쉬는 모든 경우에 쓰기가 거의 자신의 메모리에서 버퍼링이 됨으로서 다른 것들에 비해 쓰기 성능이 월등히 좋음을 알 수 있다.

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.

기존 PVFS를 위한 상호 협력 캐쉬는 힌트 기반 상호 협력 캐쉬로서 부정확한 읽기/쓰기를 수행함으로서 읽기/쓰기의 부하가 커지는 단점이 있다. 이를 해결하기 위하여 본 논문에서는 PVFS를 위한 상호 협력 캐쉬를 홈 기반 상호 협력 캐쉬로 설계 및 구현하였다.

실제 평가를 해 본 결과 읽기와 쓰기 모두 기존 PVFS를 위한 힌트 기반 상호 협력 캐쉬에 비하여 좋은 성능을 나타내었다. 읽기의 경우는 홈에 캐쉬 블록을 가져다 놓아 여러 노드에서 요청할 경우 그 요청이 처리될 수 있기 때문에 성능이 좋아진다. 쓰기의 경우 쓰기 버퍼링을 통하여 PVFS와 PVFS를 위한 힌트 기반 상호 협력 캐쉬보다 좋은 성능을 나타내고 PVFS를 위한 힌트 기반 상호 협력 캐쉬에 비하여 쓰기 때 존재하였던 블록 해제 요청을 줄임으로서 더 높은 성능을 얻을 수 있었다.

앞으로 홈 기반 상호 협력 캐쉬에서 캐쉬 블록에 대한 관리나 홈 노드를 효율적으로 구성할 수 있는 방법에 대하여 연구하고 이를 실제 사용되는 과학 계산 응용 프로그램을 이용하여 평가할 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, 김호중 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

[9] 황인철, 정한조 외, "두 단계 프로토콜: PVFS를 위한 상호 협력 캐쉬에서 쓰기 성능 향상 기법", 2003년 10월