확장 가능한 두 단계 프로토콜을 이용한 상호 협력 캐쉬의 쓰기 성능 향상

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

Enhancing Write Performance in Cooperative Cache using Extensible 2-Phase Protocol

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

요 약

요즘 네트웍과 PC의 성능이 향상됨에 따라 값싼 PC를 빠른 네트웍으로 묶어 높은 성능을 얻고자 하는 클러스터 시스템에 대하여 많이 연구 되어 왔다. 이러한 연구의 한 분야로서 클러스터 I/O 하위 시스템의 성능을 향상시키고자 하는 상호 협력 캐쉬가 제시되었다. 기존 상호 협력 캐쉬에 대한 연구는 주로 효율적인 캐쉬 공유 기법에만 집중되어있고 쓰기 성능에 대한 고려는 하지 않고 있다. 또한 대부분의 읽기 데이터는 상호 협력 캐쉬를 통하여 처리되지만 쓰기 데이터는 디스크에 접근하기 때문에 쓰기가 병목현상이 될 수 있다. 따라서 상호 협력 캐쉬에서 읽기 뿐 아니라 쓰기 성능 향상 기법에 대한 연구가 필요하다.

본 논문에서는 상호 협력 캐쉬에서 쓰기 성능 향상 기법으로 확장 가능한 두 단계 프로토콜을 제시한다. 확장 가능한 두 단계 프로토콜은 기존 두 단계 프로토콜과 같이 파일에 읽기/쓰기 접근을 연속된 읽기/쓰기 단계로 나누고, 쓰기 단계에서 연속된 쓰기사이의 불필요한 동작을 제거할 뿐 아니라 쓴 데이터에 대한 일시적 버퍼링을 수행함으로서 쓰기 성능을 향상시킨다. 그리고 확장 가능한 두 단계 프로토콜을 상호 협력 클러스터 파일 시스템의 홈 기반 상호 협력 캐쉬에 적용하여 성능을 비교, 분석한다.

1. 서론

요즘 네트웍과 단일 노드의 성능이 향상됨에 따라 값싼 노드들을 빠른 네트웍으로 묶어 높은 성능을 얻고자 하는 클러스터 시스템에 대하여 많이 연구 되어 왔다. 이러한 연구의 한 분야로서 클러스터 시스템에서 각 노드의 CPU나 메모리에 비하여 상대적으로 느린 디스크에 접근하는 I/O 하위 시스템을 효율적으로 구성하려는 연구가 이루어지고 있다.

클러스터 I/O 하위 시스템의 성능을 향상시키는 방법으로 상호 협력 캐쉬[1]가 제시되었다. 클러스터의 빠른 네트웍을 이용하면 다른 노드의 캐쉬에 접근하는 시간이 서버의 디스크에 접근하는 시간보다 빠르다. 따라서 상호 협력 캐쉬에서는 클라이언트가 데이터를 서버의 디스크에서 읽기 전에 데이터를 캐싱하고 있는 다른 노드로부터 데이터를 읽어올 수 있다. 기존 상호 협력 캐쉬에 대한 연구들[1,2,3]은 주로 노드들의 캐쉬를 효율적으로 공유하는 방법에 대하여 집중되어있다. 하지만 실제 데이터에 대한 접근은 읽기 뿐 아니라 쓰기도 존재하고 쓰기가 전체 디스크 I/O의 대부분을 차지한다. 따라서 상호 협력 캐쉬에서 읽기 뿐 아니라 쓰기 성능 향상 기법에 대한 연구가 필요하다.

본 논문에서는 상호 협력 캐쉬에서 쓰기 성능 향상 기법으로 확장 가능한 두 단계 프로토콜을 제안한다. 기존 두 단계 프로토콜[4]에서는 여러 파일에 대한 읽기/쓰기 접근을 여러개의 연속된 읽기/쓰기로 나누고 각 단계를 읽기/쓰기 단계로 지정한다. 이를 이용하여 두 단계 프로토콜에서는 PVFS를 위한 상호 협력 캐쉬에서 연속된 쓰기 사이의 불필요한 작업을 제거함으로서 성능을 향상시켰다. 본 논문에서는 두 단계 프로토콜을 확장하여 여러 상호 협력 캐쉬에서 확장 가능한 두 단계 프로토콜을 제안하였다. 확장 가능한 두 단계 프로토콜에서는 연속된 쓰기 사이의 불필요한 작업과 쓰기 데이터에 대한 일시적 버퍼링을 수행함으로서 쓰기 성능을 향상 시킨다. 이러한 확장 가능한 두 단계 프로토콜을 상호 협력 클러스터 파일 시스템[5]의 홈 기반 상호 협력 캐쉬에 적용하여 성능을 비교, 분석한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 기존의 상호 협력 캐쉬에 대한 관련 연구를 살펴본다. 3장에서는 확장 가능한 두 단계 프로토콜에 대하여 설명한다. 4장에서는 기존 상호 협력 클러스터 파일 시스템의 홈 기반 상호 협력 캐쉬와 두 단계 프로토콜을 적용하였을 때의 성능을 비교, 분석한다. 5장에서는 향후 연구 방향과 결론을 맺는다.

2. 관련 연구

상호 협력 캐쉬[1]는 기존 데이터 요구 처리 단계 중 클라이언트가 자신의 요구가 자신의 캐쉬에서 처리가 되지 않을 경우 서버에게 요청하기 전 그 블록을 캐싱하고 있는 다른 클라이언트에게 그 블록에 대한 요청을 하여 자신의 요구를 처리하는 방법이다.

이러한 상호 협력 캐쉬에 대하여 많은 연구가 이루어 졌다. Dahlin[1]은 캐쉬 블록들의 관리를 위하여 N-chance 알고리즘을 제안하였고, Feeley[2]는 GMS(Global Memroy Service) 상에서 효율적인 캐쉬 블록 알고리즘을 제안하였다. 그리고 Sarkar[3]는 기존 상호협력 캐쉬에서 정확한 클라이언트 캐싱 정보를 가지고 있던 것을 단순한 힌트에 의해 블록의 캐싱 정보를 유지함으로서 캐싱 정보를 유지하는데 필요한 부하를 줄이는 방법을 제안하였다.

기존 상호 협력 캐쉬에서 쓰기 성능을 향상 시키기 위한 방법

으로서 홈 기반 상호 협력 캐쉬[6]와 로그를 기반으로 하는 쓰기 지연 상호 협력 캐쉬[7]가 제시되었다. 홈 기반 상호 협력 캐쉬에서는 쓰기 데이터를 홈 노드에서 버퍼링 함으로서 쓰기 성능을 향상 시킨다. 로그를 기반으로 하는 쓰기 지연 상호 협력 캐쉬에서는 디스크에 데이터를 쓰기 전에 각 노드에서 데이터를 메모리에 할당된 로그에 기록하여 쓰기 성능을 향상시킨다. 이러한 방법들은 디스크에 대한 쓰기를 메모리에 대한 쓰기로 대체함으로서 쓰기 성능을 향상 시킨다.

3. 확장 가능한 두 단계 프로토콜

3.1 두 단계 프로토콜

두 단계 프로토콜[4]은 그림 1에서와 같이 기존 PVFS를 위한 상호 협력 캐쉬에서 연속된 쓰기 간의 불필요한 작업을 줄이기 위하여 제안되었다. 두 단계 프로토콜에서는 파일에 대한 읽기/쓰기를 연속된 읽기/쓰기 단계로 나누어 쓰기 단계에서 데이터에 대한 쓰기가 요청되면 불필요한 캐쉬 블록 해제 작업을 하지 않음으로서 성능을 향상 시킨다.

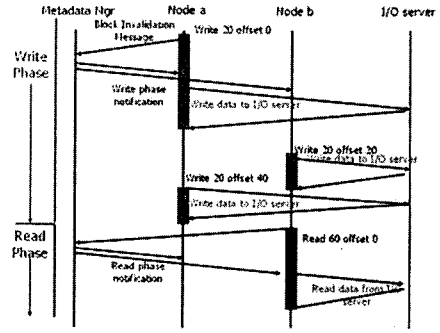

그림 1. 두 단계 프로토콜에서 쓰기/읽기

3.2 확장 가능한 두 단계 프로토콜

확장 가능한 두 단계 프로토콜은 두 단계 프로토콜에서와 같이 연속된 읽기/쓰기를 읽기/쓰기 단계로 나눈다. 그리고 쓰기 단계에서 데이터에 대한 쓰기가 요청되면 단계가 변경되기 전까지 꼭 필요하지 않은 작업들은 제거되거나 미루어진다. 확장 가능한 두 단계 프로토콜에서의 쓰기 성능 향상을 위하여 묶음 블록 업데이트와 일시적 버퍼링을 수행한다.

묶음 블록 업데이트는 쓰기 단계 내에서 쓰기가 요청되었을 때의 블록 업데이트를 하지 않고 쓰기 단계에서 읽기 단계로 전환될 때 한번의 블록 업데이트로 대체하는 방법이다. 이러한 방법을 사용하면 연속적인 쓰기 요청 사이의 불필요한 블록 업데이트를 줄일 수 있어서 쓰기 시간을 줄일 수 있다.

일시적 버퍼링 방법은 쓰기 단계에서 쓰기 요청되었을 때 쓰기 요청된 데이터를 쓰기 요청된 노드에서 일시적으로 버퍼링하는 방법이다. 쓰기 요청된 노드에서는 데이터를 보내지 않고 쓰기 헤더만을 서버에게 전달함으로서 서버에서 나중에 데이터 업데이트 할 때에 정상적으로 데이터 일관성을 유지할 수 있게 한다. 그리고 쓰기 단계에서 읽기 단계로 전환되었을 경우 모든 노드에서 일시적으로 버퍼링된 데이터들을 각 서버에 전송하여 서버가 데이터를 쓰도록 한다. 일시적 버퍼링을 이용함으로서 쓰기 단계내에서의 버퍼링으로 묶음 쓰기 방법을 사용할 수 있고 디스크 까지 요청되는 시간을 줄일 수 있다는 장점이 있다.

확장 가능한 두 단계 프로토콜은 노드들 사이의 동기화를 통하여 단계를 변경시켜야 하는 부하가 있다. 따라서 노드들 사이의 동기화 후 파일에 접근할 경우 부하가 적어지기 때문에 그 성능을 크게 향상시킬 수 있다. 하지만 동기화 없이 각 노드에서 독립적으로 파일에 대한 접근을 요청할 경우 두 단계 프로토콜에 의한 동기화 부하가 커져서 성능 향상이 없을 수 있다. 따라서 확장 가능한 두 단계 프로토콜에서는 각 노드에서 동기화 정도를 계산하여 동기화 정도가 높을 경우에만 위 두 방법을 사용하고 정도가 낮아지면 사용하지 않는 적응성을 사용하였다.

3.3 확장 가능한 두 단계 프로토콜의 적용

본 논문에서는 확장 가능한 두 단계 프로토콜을 상호 협력 클러스터 파일 시스템[5]의 홈 기반 상호 협력 캐쉬에 적용하였다. 상호 협력 클러스터 파일 시스템에서는 홈 기반 상호 협력 캐쉬를 이용하여 메타데이터 블록들은 업데이트 기반 일관성 유지 정책을 사용하고 데이터 블록들은 해제 기반 일관성 유지 정책을 사용한다. 상호 협력 클러스터 파일 시스템에서는 메타데이터 블록 업데이트의 양을 줄이기 위하여 메타데이터 업데이트 방법을 작은 여러 개의 작은 업데이트로 나누어 진행한다. 따라서 상호 협력 클러스터 파일 시스템에서는 이러한 메타데이터 블록 업데이트를 꼭 필요한 파일 크기, 블록 포인터, 블록 개수 등의 업데이트만 진행하도록 구성되었다. 상호 협력 클러스터 파일 시스템에서 쓰기 요청에서 데이터 블록 쓰기와 메타데이터 블록 업데이트로 구성되어 있으며 메타데이터 업데이트 중 파일 크기 업데이트만을 진행하도록 구성되어있다.

확장 가능한 두 단계 프로토콜의 묶음 블록 업데이트 방법을 이용하여 그림 2에서와 같이 연속된 쓰기 요청 사이의 파일 크기 업데이트를 제거하도록 구성되었다. 그리고 확장 가능한 두 단계 프로토콜을 이용하여 쓰기 요청시 쓰기 데이터를 요청된 노드에서 일시적으로 버퍼링을 수행함으로서 쓰기 성능을 향상 시킨다.

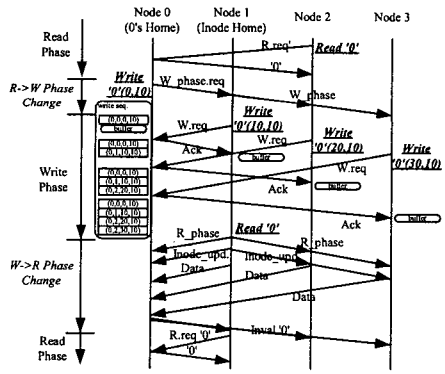

그림 2. 확장 가능한 두 단계 프로토콜의 적용

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	RedHat 9.0(Kernel 2.4.20)

표 1. 성능 평가 환경

상호 협력 클러스터 파일 시스템은 4노드에서 동작하며 각 노드의 구성은 위 표1과 같다. 메타데이터 블록은 1KB이고 데이터 블록은 64KB블록으로 설정하여 사용하였다.

성능 평가를 위하여 표 2와 같은 BTIO[8] 벤치마크 프로그램

을 사용하였다. BTIO 벤치마크 프로그램은 병렬 파일 시스템의 성능을 측정하기 위하여 사용된다. 본 논문에서는 여러 데이터 종류 중 중간정도의 크기인 's'를 사용하여 성능 측정을 수행하였다.

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

4.1 두 단계 프로토콜의 부하

다음 그림 3은 상호 협력 클러스터 파일 시스템의 홈 기반 상호 협력 캐쉬에서 두 단계 프로토콜의 단계 변경기능만을 추가하였을 경우 각 프로그램에서의 기존 환경을 기준으로 한 정규화한 읽기/쓰기 시간을 나타낸다.

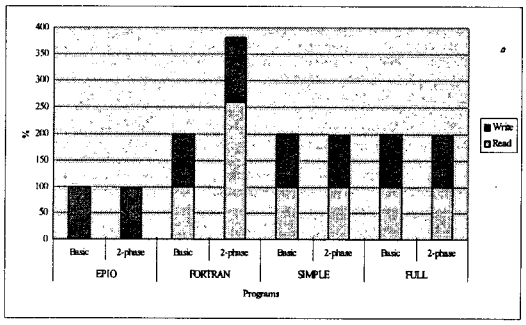

그림 3. 두 단계 프로토콜의 부하

Fortran 프로그램의 경우 동기화 율이 약 9.8%로 다른 노드와의 동기 없이 독립적으로 단계 변경을 수행한다. 따라서 두 단계 프로토콜의 부하가 커져서 이 프로그램에서는 확장 가능한 두 단계 프로토콜이 사용되지 않는다. Simple과 Full 프로그램의 경우 동기화가 100%로 큰 부하없이 단계 변경을 수행할 수 있고 따라서 확장 가능한 두 단계 프로토콜이 수행됨으로서 큰 부하없이 성능 향상을 기대할 수 있다.

4.2 확장 가능한 두 단계 프로토콜 수행 결과

다음 그림 4는 확장 가능한 두 단계 프로토콜을 수행한 결과를 모음 블록 업데이트(Meta)와 일시적 버퍼링(Buf)을 각각, 그리고 같이 수행했을 때(All)의 읽기/쓰기 수행시간을 사용하지 않은 것을 기준으로 정규화한 상태를 나타낸다.

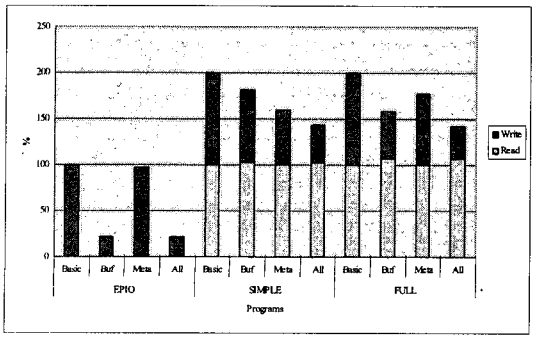

그림 4. 확장 가능한 두 단계 프로토콜에서의 읽기/쓰기 시간

확장 가능한 두 단계 프로토콜을 사용할 경우 쓰기 성능을 크게 향상시킨다. 이는 홈 기반 상호 협력 캐쉬(Basic)이 홈 노드에서 버퍼링을 수행하지만 각 노드에서 일시적으로 쓰기 단계 동안에 버퍼링을 수행하는 것(Buf)이 더 좋을 뿐 아니라 메타데이터 블록 업데이트도 자주하지 않고 단계 변경 시에만 함으로서 불필요한 작업이 제거되어 쓰기 성능을 향상시킨다. 하지만 확장 가능한 두 단계 프로토콜에서 읽기 성능의 경우 약간의 추가적인 부하를 갖게 되는데 쓰기에서 읽기 단계 변경시에 블록을 업데이트하거나 쓴 데이터를 적용하는 부하 때문에 생긴다. 하지만 이러한 부하들은 쓰기 성능 향상에 비하여 매우 적고 결과적으로 프로그램 수행시간은 확장 가능한 두 단계 프로토콜을 사용하였을 경우 4~16%가 줄어드는 것을 알 수 있다.

5. 향후 연구 방향 및 결론

본 논문에서는 상호 협력 캐쉬의 쓰기 성능 향상 기법으로 확장 가능한 두 단계 프로토콜에 대하여 설명하였다. 확장 가능한 두 단계 프로토콜에서 연속적은 쓰기간의 메타데이터 블록 업데이트와 일시적인 버퍼링을 통하여 상호 협력 캐쉬에서 쓰기 성능을 향상 시킬 수 있다.

앞으로 확장 가능한 두 단계 프로토콜을 이용하여 여러 파일 시스템에 적용시킬 예정이다. 이러한 확장 가능한 두 단계 프로토콜을 이용하여 각 단계의 불필요한 작업을 찾아내어 제거하고 묶음 가능한 것들을 찾고 그것들을 확장 가능한 두 단계 프로토콜을 사용하도록 할 계획이다. 그리고 가능한 병렬 I/O를 사용하는 실제 응용 프로그램을 사용하여 더 많은 실험을 수행할 계획이다.

6. 참고 문헌

[1] 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

[2] 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

[3] 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

[4] In-Chul Hwang,Hanjo Jung, Seung-Ryol Maeng, Jung-Wan Cho, "2-Phase Protocol: Enhancing Write Performance in Cooperative Cache for PVFS",The IASTED International Conference on Parallel and distributed computing and networks, Feb. 15-17, 2005, Innsbruck,Austria

[5] In-Chul Hwang,Donghyuk Lim, Seung-Ryol Maeng, Jung-Wan Cho, "Design and Implementation of Cooperative Cluster File System", International Conference on Parallel and Distributed Processing Techniques and Applications, June 27-30, 2005, Las Vegas, Nevada, USA.

[6] In-Chul Hwang,Hanjo Jung, Seung-Ryol Maeng, Jung-Wan Cho, "Design and Implementation of the Home-based Cooperative Cache for PVFS", International Conference on Computational Science(ICCS'05), May 22~25, 2005, Emory University Atlanta, USA

[7] Vlad Olaru, Walter F.,"CARDs: Cluster-Aware Remote Disks", In Proceedings of the Third IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003) , pages 112-119, Tokyo, Japan, May 2003.

[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