2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

리눅스 클러스터 환경에서 단일 블록 디바이스 이미지에 관한 연구

김태호*, 이종우*
*한림대학교 컴퓨터공학과
e-mail: thkhyme@center.cie.hallym.ac.kr,
jwlee44@sun.hallym.ac.kr

A Study on Single Block Device Image for Linux Cluster Environment

Taiho Kim*, Jongwoo Lee*
*Dept of Computer Engineering, Hallym University

요약

현대 사회의 모든 분야가 정보화에 의해 통합되면서 여러 분야에서 고성능 서버에 대한 수요가 증가하고 있다. 그러나, 그것의 높은 비용과 상대적으로 제한된 성능으로 인하여 최근에는 여러 대의 호스트를 네트워크로 연결하는 클러스터링 기술이 각광을 받고 있다. 이러한 다수의 호스트로 구성된 클러스터 시스템의 성능을 최적화하기 위해, 각 노드에 분산된 자원을 효율적으로 통합하고 관리함으로써 사용자에게 투명하고 일관된 인터페이스를 제공하는 단일 시스템 이미지의 지원이 요구된다. 본 논문에서는 리눅스 클러스터 시스템의 입출력 공간에서 단일 시스템 이미지를 지원하기 위한 가상 블록 디바이스 드라이버 설계를 제안하였다. 가상 블록 디바이스 드라이버는 원격 노드의 디스크를 가상의 지역 디스크로 다루기 위해 기존의 파일 시스템을 수정하지 않고 디바이스 드라이버 수준에서 접근함으로써 파일 시스템과의 호환성을 유지하며 사용자에게 투명성을 제공한다.

1. 서론

오늘날의 컴퓨팅 환경에서 클러스터 시스템은 저비용으로 고성능을 제공함으로써 무한한 잠재력을 인정받고 있다. 이러한 환경에서 단일 시스템 이미지의 지원은 사용자로 하여금 클러스터 시스템을 보다 효율적으로 관리 및 활용할 수 있게 한다. 특히, 클러스터 시스템에서 입출력 연산은 전체 시스템의 병목점이 되지 않도록 하기 위해 데이터를 분산하여 저장할 필요가 있으며 이러한 서비스를 지원하는 단일 입출력 공간을 요구하게 된다.

본 논문에서는 단일 입출력 공간을 제공하기 위해 기존에 제시된 접근 방식들의 문제점을 분석하고, 이를 해결하기 위해 리눅스 클러스터 환경에서 가상 블록 디바이스 드라이버 설계를 제안한다.

2. 관련 연구

본 장에서는 단일 시스템 이미지와 단일 입출력 공간에 대해 고찰하고, 단일 입출력 공간을 지원하기 위해 운영체제의 계층별로 제시된 기존의 접근 방식을 기술한다.

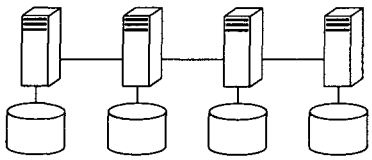
2.1. 단일 시스템 이미지와 단일 입출력 공간

단일 시스템 이미지는 여러 대의 PC 또는 워크스테이션들이 네트워크로 연결된 클러스터 시스템을 사용자에게 하나의 시스템으로 보이게 하는 것이다. 완전한 단일 시스템 이미지는 사용자에게 투명성, 확장성, 가용성을 제공해야 하며, 이를 지원하기 위해 다음과 같은 서비스가 요구된다[2].

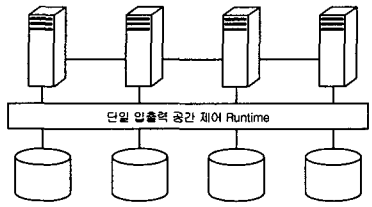
- Single entry point
- Single file hierarchy
- Single control point
- Single virtual networking
- Single memory space
- Single job-management system
- Single user interface

단일 입출력 공간은 클러스터 시스템에서 개별 노드에 장착된 입출력 장치 또는 저장 장치의 물리적인 위치에 대한 지식 없이 어떤 노드에서도 접근 가능하게 하는 것으로 단일 시스템 이미지의 기본 목적을 달성하기 위한 필수적인 기능이다. 단일 입

출력 공간은 사용자의 관점에서 그림 1과 같은 단일 시스템 이미지 서비스를 제공함으로써 지역 디스크 접근과 원격 디스크 접근간의 차이를 제거하며, 디스크에 저장될 데이터를 클러스터 시스템의 각 노드로 분산시킴으로써 노드간 데이터 이동을 위한 입출력 연산의 성능을 향상시킨다.



(a) 단일 입출력 공간을 지원하지 않는 경우



(b) 단일 입출력 공간을 지원하는 경우

그림 1. 단일 입출력 공간

2.2. 단일 입출력 공간에 대한 기존 연구

기존의 연구를 살펴보면 사용자 수준 접근[3, 4], 파일 시스템 수준 접근[5], 디바이스 드라이버 수준 접근[1] 등이 있다.

2.2.1. 사용자 수준 접근

사용자 수준에서 단일 시스템 이미지 서비스를 구현하는 것은 저 수준의 접근과 비교하여 고유한 몇 가지 장점을 갖는다. 사용자 수준 접근은 상대적으로 그 구현 비용이 낮으며 설계, 구현, 적용시 오류를 발견하기 쉽다. 또한, 구현된 시스템은 저 수준에서 구현된 시스템보다 쉽게 타 시스템으로 이식될 수 있다. 그러나, 사용자 수준 접근은 두 가지의 고유한 단점을 안고 있다. 첫 째, 사용자 수준 접근은 완전한 단일 시스템 이미지 서비스를 제공할 수 없다. 사용자는 서비스를 제공받기 위해서 별도의 노드 구별자나 API를 사용해야 한다. 둘 째, 사용자 수준 접근에서 필연적으로 사용하게 되는, 커널이 제공하는 파일 입출력 서비스와 네트워크 서비스는 클러스터 시스템을 위해 최적화되지 않을 수 있다. 또한, 그러한 서비스를 수행하기 위한 시스템 호출은 전체 시스템의 성능을 제한한다.

사용자 수준에서 단일 입출력 공간의 예로 Clemson 대학의 PVFS[3]과 Argonne 국립 연구소의 RIO[4] 등이 있다.

2.2.2. 파일 시스템 수준 접근

파일 시스템 수준에서의 구현은 클러스터 시스템을 위해 파일 시스템을 최적화하고 파일의 접근과 관련하여 분산 패턴을 제어함으로써 사용자에게 완전한 단일 시스템 이미지 서비스를 제공할 수 있다. 그러나, 새로운 파일 시스템을 설계, 구현, 적용하는 것은 비용이 크고, 기존 시스템 또는 프로그램과의 엄격한 호환성을 제공하기 어렵다.

파일 시스템 수준에서 단일 입출력 공간의 예로 U. C. Berkeley의 xFS[5] 등이 있다.

2.2.3. 디바이스 드라이버 수준 접근

디바이스 드라이버 수준 접근은 파일 시스템을 수정하지 않기 때문에, 디스크 입출력 연산에 대해 사용자 뿐만 아니라 파일 시스템에게도 단일 시스템 이미지를 보여준다. 그리고 수정되지 않은 파일 시스템은 기존의 응용 프로그램과 호환성을 유지한다. 또한, 디바이스 드라이버 수준으로 제한된 접근은 설계, 구현, 적용을 용이하게 하여 개발비용을 절감할 수 있다. 그러나, 디바이스 드라이버 수준에서는 입출력 연산의 최적화를 위하여 파일의 분산 패턴을 제어할 수 없다.

디바이스 드라이버 수준에서 단일 입출력 공간의 예로 SIOS[1] 등이 있다. SIOS는 디바이스 드라이버 계층에서 사용되는 물리 주소를 상위 계층에서 논리 주소에 의해 맵핑함으로써 각 데이터 블록에 대해 단일 주소 공간을 제공한다. 따라서 각 클러스터 노드의 지역 파일 시스템은 각 노드에 분산되어 있는 디스크의 집합을 하나의 디스크처럼 보게된다. 디바이스 드라이버 수준의 접근이 상위 계층을 지원함에 있어서, SIOS가 분산된 디스크 자원을 단일 디스크처럼 보이게 하는데 반하여, 본 연구에서 제안하는 가상 블록 디바이스 드라이버는 원격 디스크를 지역 디스크와 별개의 가상 지역 디스크로 보이게 한다.

3. 가상 블록 디바이스 드라이버

본 장에서는 본 연구에서 제안하는 가상 블록 디바이스의 설계와 관련된 쟁점들, 그리고 그것의 구조와 구성요소들간의 동작을 기술한다. 가상 블록 디바이스 드라이버는 각 노드에 존재하는 지역 디스크와 더불어 원격 노드에 존재하는 디스크를 가상의

지역 디스크로 다룬다. 가상 블록 디바이스 드라이버는 디바이스 드라이버 수준으로 제한된다.

3.1. 설계와 관련된 쟁점들

· 버퍼 캐쉬 일관성(consistency)

버퍼 캐쉬의 일관성에 대한 문제는 두 가지 경우로 나누어 볼 수 있다. 클러스터 시스템 내에서 다중 사본(multiple copy)을 허용하지 않을 경우는 논의할 필요가 없다. 다중 사본을 허용할 경우에는 각 노드를 연결하는 상호연결망의 조건, 구현의 용이성, 성능을 고려하여 lazy release consistency model 또는 release consistency model을 적용하여 일관성을 보장할 수 있다[7, 8].

· 저장 장치 확장성(scalability)

가상 블록 디바이스 드라이버는 서비스를 제공하기 위해 별도의 중앙 서버를 두지 않으며, 클러스터 내의 저장 장치 개수는 클러스터내의 노드수 * 노드당 장착가능한 디스크수 까지 자유로이 확장 가능하다. 각 노드는 단일 입출력 공간 초기화시 자신이 보유한 장치가 단일 입출력 공간에 참여할지 말지를 결정한다.

· 기존 클러스터 응용 프로그램과의 호환성(compatibility)

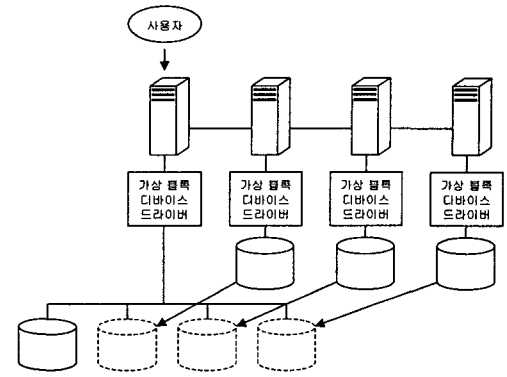
가상 블록 디바이스 드라이버는 기존 파일 시스템의 수정을 요구하지 않으며, 따라서 기존 클러스터 응용 프로그램과의 호환성을 유지할 수 있다.

· 새로운 커널 버전으로의 이식성(portability)

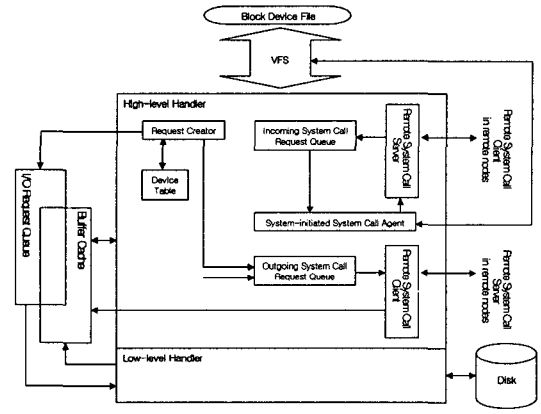
가상 블록 디바이스 드라이버의 구조는 디바이스 드라이버 수준의 모듈 형태로 제한됨으로써 새로운 커널에 대해 참조한 심볼의 수정만으로 이식이 가능하다.

3.2. 기본 구조와 동작

가상 블록 디바이스 드라이버는 지역 입출력 요구와 원격 입출력 요구 모두를 처리하며 크게 두 부분으로 구성된다. 상위 계층 핸들러는 기존 블록 디바이스 드라이버의 상위 계층 핸들러와 유사하나 원격 디스크를 접근하기 위한 기능이 추가된다. 첫 째, 버퍼 캐쉬에서 캐쉬 미스의 경우, 생성되는 입출력 요구를 지역 디스크에 대한 요구와 원격 디스크에 대한 요구로 구분한다. 둘 째, 지역 노드의 가상 블록 디바이스 드라이버는 노드간 네트워크를 통하여 원격 노드의 가상 블록 디바이스 드라이버와 상호 통신이 가능하다. 셋 째, 원격 노드로부터의 입출력 요구를 처리한다. 하위 계층 핸들러는 지역 디스크 상의 파티션에 저장된 데이터에 대한 입출력 연산을



(a) 전체 시스템 이미지



(b) 내부 구조

그림 2. 가상 블록 디바이스 드라이버의 구조

수행하는 부분으로 기존 블록 디바이스 드라이버의 하위 계층 핸들러와 동일하다. 그림 2는 가상 블록 디바이스 드라이버의 구조를 보이고 있는데, 가상 블록 디바이스 모듈을 이루는 구성요소들은 다음과 같다.

· Request Creator

버퍼 캐쉬에서 캐쉬 미스(cache miss)가 발생되면 request creator는 입출력 요구를 생성하여 적절한 큐로 삽입한다. 이 과정에서 request creator는 device table을 참조하여 요구된 데이터의 물리적인 위치를 결정하여 지역 입출력 요구는 I/O request queue로, 원격 입출력 요구는 outgoing system call request queue로 삽입한다. 지역 입출력 요구를 위한 I/O request queue는 기존의 것과 동일하다. Outgoing system call request queue에는 원격 디스크에 대한 실질적인

I/O 연산 요청 뿐만아니라 원격 시스템 호출 요구도 삽입된다.

- Remote System Call Client

  Remote system call client는 비어있는 outgoing system call request queue에 새로운 요구가 삽입된 후에 활성화되어 원격 시스템 호출 요구를 적절한 원격 노드로 보낸다. 그리고, 원격 입출력 요구의 경우 요구한 데이터를 받아 버퍼 캐쉬에 기록한다.

- Remote System Call Server

  Remote system call server는 타 노드의 가상 블록 디바이스 드라이버와의 통신을 위해 사용되는 포트를 검사하여 원격 노드의 remote system call client로부터 수신된 요구를 incoming system call request queue로 삽입한다.

- System-initiated System Call Agent

  System-initiated system call agent는 비어있는 incoming system call request queue에 새로운 요구가 삽입된 후에 활성화되어 VFS(virtual file system)로 적절한 시스템 호출을 요청하고 그 결과를 받아 remote system call server로 보낸다. 원격 노드로부터의 요구가 블록 I/O 연산일 경우에는 이 모듈에 의한 시스템 호출의 결과로써 요구된 데이터가, 지역 노드의 지역 디스크에 대한 I/O 시스템 호출에서와 같이, 버퍼 캐쉬에 저장된다.

4. 결론 및 향후 연구과제

본 연구에서 제시한 가상 블록 디바이스 드라이버 설계는 클러스터 환경에서 입출력과 관련된 하위 시스템을 파일 시스템과 가상 디스크의 집합으로 분리시킴으로써 상위 계층에서의 접근 방식에서 제시된 문제점들을 해결하며, 구조를 모듈단위로 단순화하여 실제 구현시 용이하게 하였다. 가상 블록 디바이스 드라이버의 구현은 진행 중이며, 향후 연구는 본 논문에서 제안한 가상 블록 디바이스 드라이버를 기반으로 다양한 프로그램을 적용하여 문제점을 파악하고, 클러스터 환경에서 사용되는 프로그램들의 입출력 패턴을 분석하여 디바이스 드라이버 수준에서 이를 제어하고 최적화하는 것이다.

참고문헌

[1] Roy S. C. Ho, K. Hwang, and H. Jin, "Single I/O Space for Scalable Cluster Computing", Proceedings of the 1st IEEE Computer Society International Workshop on Cluster Computing, pp.158-166, December 1999

[2] K. Hwang, H. Jin, E. Chow, C. L. Wang, and Z. Xu, "Designing SSI Clusters with Hierarchical Checkpointing and Single I/O Space", IEEE Concurrency, pp.60-69, January-March 1999

[3] 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, pp.317-327, October 2000

[4] I. Foster, D. Kohr Jr.,R. Krishnaiyer, and J. Mogill, "Remote I/O: Fast Access to Distant Storage", Proceedings of the 5th Annual Workshop on I/O in Parallel and Distributed Systems, pp.14-25, November 1997

[5] T. Anderson, M. Dahlin, J. Neefe, D. Patterson, D. Roselli, and R. Wang, "Serverless Network File Systems", ACM Transactions on Computer Systems, Vol.14, No.1, pp.41-79, 1996

[6] E. K. Lee and C. A. Thekkath, "Petal: Distributed Virtual Disks", Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.84-92, October 1996

[7] P. Keleher, A. Cox, and W. Zwaenepoel, "Lazy Release Consistency", Proceedings of the 19th ACM International Symposium on Computer Architecture, pp.13-21, 1992

[8] K. Gharachorloo, A. Gupta, and J. Hennessy, "Performance Evaluation of Memory Consistency Models for Shared-Memory Multiprocessors", Proceedings of the 4th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.245-257, April 1991

[9] D. P. Bovet and M. Cesati, "Understanding Linux Kernel" 1st Ed., O'reilly, 2001

[10] A. Rubini, "Linux Device Drivers" 1st Ed., O'reilly, 1998