신뢰성 있는 네트웍 램디스크의 설계 및 구현

허준호, R. S. Ramakrishna
광주과학기술원 정보통신공학과
e-mail : jhher@kjist.ac.kr

Design and Implementation of Reliable Network RamDisk

Jun-Ho Her, R. S. Ramakrishna
Dept. of Information and Communications, Kwang-Ju Institute of Science and Technology (K-JIST)

요 약

전통의 자기 디스크 형태 저장 장치로의 접근 지연 시간을 줄이고자, 최근에 서로 네트워크로 연결된 컴퓨터간의 메모리 자원들을 파일을 저장하는 장치로써 활용해 볼 수 있는 가상 장치(Network RamDisk)가 개발되었다. 이 시스템은 메모리 공간을 제공해 주는 서버의 노드들이 패리티 그룹(parity group)을 이루는 일종의 software RAID 시스템으로 한 서버의 노드 crash 에 대해 대응하는 것을 고려하지 않아 제 이의 서버가 노드 crash, 프로세스 crash 혹은 데이터 손실을 발생시키면 완전히 데이터를 잃어버리는, 신뢰성이 낮은 문제점이 있다. 본 논문은 클라이언트가 서버측의 노드 crash 를 적절히 감지하고 새로운 서버로 fail over 할 수 있는 여러 가지 기법들을 설계하고 구현하였다. 또한 서버 풀(server pool)이라는 가상의 공간을 따로 관리하여 서버들에 대한 메모리 자원 요구 부하를 균등히 분산 시키는 효과도 얻도록 하였다.

1. 서론

근대의 컴퓨터 산업에 있어서, 저장 시스템의 발전은 계산 성능 향상에 비해 크게 뒤떨어지는 경향을 나타내왔다. 대표적인 저장 기술인 자기 디스크의 경우, 블록 접근 속도와 대역폭이 계산 속도에 비해 매우 큰 격차를 드러내 왔고 점차 이러한 격차는 커지는 추세에 있다[1]. 그러나 고속의 네트웍 기반설비와 저장 매체 접근의 병렬화에 의해 이런 한계점들을 극복하고자 하는 연구가 최근 대두되고 있다. 그러한 예로써 기반 환경을 제공해 주는 기술로는 NAS 나 SAN 과 같이 여러 대의 서버와 여러 대의 저장 시스템이 TCP/IP 혹은 Fibre Channel 의 네트웍에 직접 연결되는 네트웍 연결 스토리지 기술이 있다[2]. 또한 이들을 뒷받침 해주는 파일 시스템으로는 NFS[3]와 GFS[4]가 있는데, 최근의 다른 파일 시스템들의 기본 개념을 이루는 대표적인 파일 시스템이라고 할 수 있다. 마지막으로 스토리지 매체의 성능 향상을 위한 기술로는 상용 디스크를 병렬적으로 배열하여 제어기에 의해 제어하는 RAID 시스템이 있다.

하지만 저장 매체의 성능향상을 위한 RAID 시스템은 대량의 데이터를 처리하기 위한 고 대역폭을 실현한 기술일 뿐 저장 매체의 물리적 블록 접근 속도의 향상에 대해서는 해결점을 제시하지 못하고 있다. Web server 와 같은, 다수의 파일들을 빠르게 처리해 주어야 되는 응용 프로그램들에 대해서는 빠른 블록 접근 시간이 요구되며, 최근 이러한 문제의 해결을 위한 연구가 시도되었다. 이것이 네트웍 램디스크(Network RamDisk)라고 명명된 기술로[5], 이는 기존의 Zebra[6]와 같은 software RAID[7]의 개념과 빠른 블록 접근시간을 제공하는 전통적인 램디스크의 개념을 결합한 것으로 NOW 나 Beowulf 와 같은 클러스터 형태의 컴퓨팅 환경에서 응용하기 위한 한 시도라고 볼 수 있다. 더구나 한 연구에서는 클러스터 컴퓨터의 각 노드들의 메모리 자원의 가용성과 유용성이 실험을 통해서 검증되었다[8]. Software RAID 를 구현할 때는 저장 공간을 제공하는 서버 노드와 그 것을 이용하는 클라이언트 노드로 구성되는 것이 일반적인데[6], 이

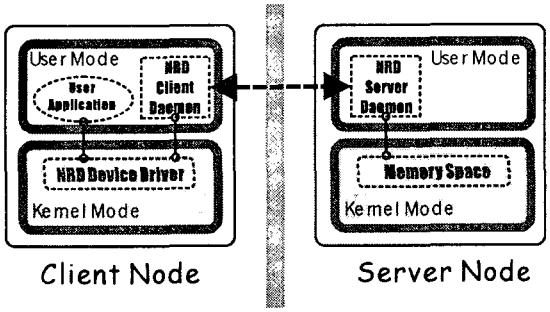
런 구조가 가지는 장점이라면, fail over 나 replication 과 같은 기법을 통해 잠재적으로 안정성과 가용성을 높일 수 있다는 것이다.

본 논문에서는 프로토타입으로 구현된 네트웍 램디스크의 취약점을 알아보고 이를 해결하기 위한 기법들을 살펴본 후 실질적인 구현에 대해 기술하고 간단한 기능 테스트에 대해서 기술한다. 본 논문의 구성은 다음과 같다. 2 장에서는 네트웍 램디스크의 구성요소와 동작원리에 대해서 설명하고 그 것이 갖고 있는 문제점과 안정성과 가용성에 대한 연구에 대해 기술한다. 3 장에서는 해결을 위한 새로운 요소들의 추가와 변경되는 부분의 설계 및 구현에 대해 살펴본다. 4 장에서는 간단한 테스트를 통해 구현의 작동을 확인하고 결과에 대해 고찰한다. 마지막으로 5 장에서는 결론과 향후 과제에 대해서 알아본다.

2. 관련연구

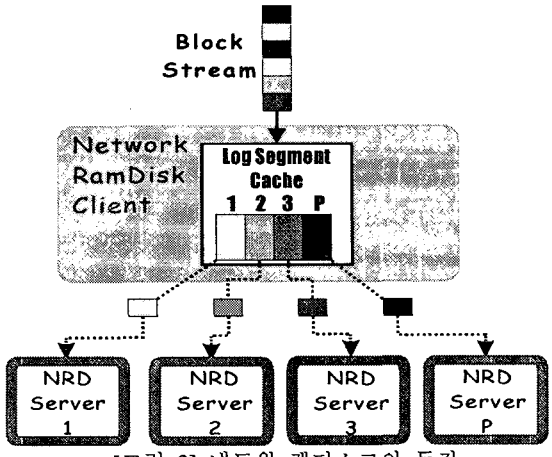
2.1 네트웍 램디스크 (Network RamDisk)[5]

물리적인 디스크 접근 시간의 한계를 극복하기 위한 한 연구로 최근에 네트웍 램디스크(NRD)라고 하는 가상의 블록 저장장치가 구현되었다. 이 가상장치는 기존의 로컬 램을 디스크처럼 사용하는 램디스크의 개념을 네트웍으로 연결된 다른 PC 나 웍스테이션들의 메모리로 확장한 것으로 Zebra[7]에서 적용된 LFS[9]개념이 적용된 일종의 software RAID 다. 다만 기존의 ramdisk 처럼 실제의 물리적인 메모리를 kernel 레벨에서 제공하는 것이 아니라 서버 프로세스의 메모리 공간을 통해서 우회적으로 블록 공간을 제공한다.



[그림 1] NRD 의 접근 레벨 구조도

그림 1 은 NRD 를 이루는 구성 요소들의 노드간 접근 레벨 구조를 도식화하고 있다. 클라이언트측 사용자 응용 프로그램의 일반 파일 입출력 요구에 대해 NRD 디바이스 드라이버가 그 요구를 사용자 레벨의 클라이언트 데몬 프로그램에 전달한다. 그 후 데몬 프로그램에 의해 관리되는 정보를 이용 원격의 서버 데몬 프로그램으로부터 메모리 공간을 블록 저장공간으로 이용하게 된다.



[그림 2] 네트웍 램디스크의 동작

그림 2 에서 보는 바와 같이 서버 1, 서버 2, 서버 3 그리고 패리티 서버는 클라이언트 데몬의 요청에 의해 고정적으로 하나의 패리티 그룹(parity group)을 형성하게 되고 이런 그룹에 대한 모든 정보는 클라이언트 데몬 프로그램에 의해 관리된다. 클라이언트의 log segment cache 는 캐쉬의 역할을 함과 동시에 LFS 에서처럼 일정량의 블록들을 비축하여 small write problem 을 해결하였다[9]. 파일 쓰기를 예로 들자면, 쓰기 요청으로 들어오는 블록 스트림을 log segment cache 에 저장하고 캐쉬가 다 차게 되면 각 대응되는 striping 블록들을 해당 서버에 보낸다.

하지만 현재 구현된 네트웍 램디스크는 서버의 구성이 고정적이어서 한 서버에, 전력차단에 의한 다운이나 OS crash 등의 노드 crash 가 발생한 상태에서 패리티 메커니즘에 의한 데이터 복구만 수행할 뿐 그에 적절한 대응이 전무하다. 따라서 그 서버에 대한 완전한 복구가 일어나기 전에 나머지 서버 중 하나라도 노드 crash, 프로세스 crash 혹은 데이터 손실을 갖게 된다면 블록을 완전히 잃게 된다. 이런 취약점에 대해 머신의 전기적 다운이나 노드의 OS crash 와 같은 노드 crash 가 발생했을 때 이를 빠르게 감지하고 바로 fail over 를 수행할 수 있도록 하여 연속된 노드 crash 나 데이터 손실등에 대해 유연하게 대처할 수 있는 방안을 모색해야 한다. 또한 여러 클라이언트 노드들이 존재하는 경우 서버들의 메모리 자원에 대한 관리가 전혀 없기 때문에 부하가 특정 서버에 집중되는 현상이 발생할 가능성이 있다. 따라서 서버들의 리소스를 관리해 주는 추가적인 구성 성분이 요구된다.

2.2 신뢰성과 가용성

신뢰성이라 함은 한 시스템에서 오류나 구성 성분의 failure 에 대한 포용력의 정도이다. 신뢰성 있는 시스템의 한 예로는, 구성 성분의 failure 로 인한 정보의 손실을 미연에 방지하는 시스템이라 할 수 있다[10]. 가용성이라 함은 특정 시간에 대하여 계속해서 시스템을 쓸 수 있는 시간의 비로써, 고 가용성에서의 '높은'이라는 것의 척도는 계산하는 목적의 특성에 따라

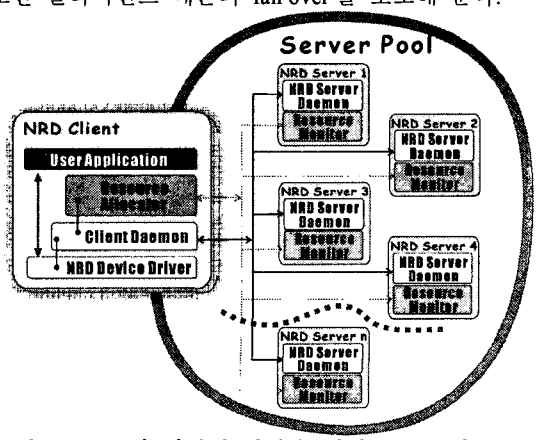
서 범위가 정해지는 추상적인 서비스 질의 높음을 의미하다. 예를 들어 전화 교환기의 정지 시간은 1 년에 약 30 초미만 이어야 어느 정도 양질의 서비스를 제공할 수 있다. 비율로 다시 표현한다면 약 99.9999%이상의 가용성을 제공하여야 고 가용성의 시스템이라고 할 수 있다[11].

하지만 단순히 특정시간에 대한 시스템의 가용시간만 가지고 고 가용성을 논하는 것은 의미가 없다. 예를 들어서 웹 서버의 경우, 한 달에 30 분 정도 시스템이 정지된다고 한다면 하루에 1 분씩 정지되는 것과 단 하루에 30 분이 정지되는 것은 엄연히 차이가 있다. 이런 경우 후자보다 전자가 더 가용성이 크다고 볼 수 있다. 따라서 문제가 발생했을 때 이를 빠르게 체크하고 fail over 가 일어나야 고 가용성의 시스템이 될 수 있다.

3. 설계 및 구현

3.1 자원 할당자와 자원 모니터

서버 노드에 간단하게 자신의 메모리 상황을 주기적으로 체크 하는 자원 모니터 데몬 프로그램을 만들고 클라이언트 노드에 자원 모니터들에 연결되는 자원 할당자 데몬 프로그램을 두어 메모리 자원을 제공해 주는 서버들의 가상적인 풀(server pool)을 형성한다. 자원 할당자는 메모리 자원에 대한 상황을 기록하고 있다가 클라이언트 데몬의 서버 할당 요구에 대해 자유 메모리의 양이 많은 서버의 순서로 호스트 이름을 되돌려 준다. 그림 3 에서 이러한 상황을 나타내고 있다. 예를 들어 클라이언트 데몬이 패리티 그룹을 형성하는 4 개의 서버 정보를 요구하면 자원 할당자는 현재 사용되고 있지 않고, 메모리 자원이 많은 순서대로 4 개의 서버를 할당해 주며, 이런 상황에 대한 정보를 갱신하게 된다. 그리고 fail over 등의 상황이 발생하게 되어 클라이언트 데몬이 새로운 서버를 요구하게 되는 경우, 사용되고 있지 않은 나머지 서버 중 메모리 자원이 가장 많은 서버 하나의 정보를 준다. 이러한 구조를 통해 자연스럽게 자원의 부하를 조율해 주고, 또한 클라이언트 데몬의 fail over 를 보조해 준다.

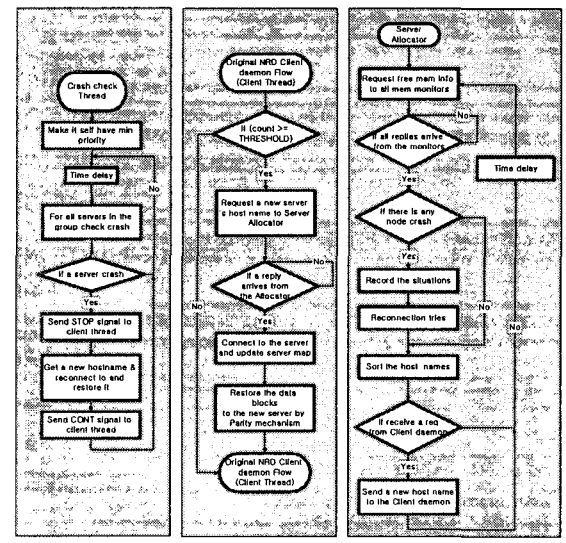


[그림 3] NRD 의 신뢰성 향상을 위한 요소들의 구성도

3.2 클라이언트 데몬의 수정

클라이언트 데몬이 종전에 명시적으로 NRD 서버에 연결하던 것을 자원 할당자로부터의 정보를 이용하여 해당 서버에게 연결하도록 하고, 기존의 단일 프로그램을 기 연결되어 있는 서버 노드들의 crash 유무를 판별하는 쓰레드와 기존의 클라이언트의 기능을 하는 쓰레드로 운영한다. 다만 성능 저하의 오버헤드를 최소한으로 하기위해 노드 crash 를 판별하는 쓰레드를 쓰레드 스케줄링상 가장 낮은 우선 순위 값으로 설정한다. 노드 crash 유무를 판별하는데 있어서, 클러스터와 같은 국지적인 멀티 컴퓨팅 환경에서는 ping 프로그램만으로 충분할 것이므로 노드 crash 를 판별하는 쓰레드는 기 연결되어 있는 서버 노드에 주기적으로 ping 을 실행한다. 이 쓰레드에 의해 서버 노드의 crash 가 감지되면 쓰레드는 클라이언트 쓰레드의 동작을 정지 시키고 자원 할당자로부터 새로운 서버의 정보를 받아 패리티 메커니즘에 의해 crash 가 발생한 노드에 대응되는 블록을 재생시킴으로써 fail over 를 수행하게 된다. Fail over 가 완전히 이루어 지면 클라이언트 쓰레드를 재게 하도록 하여 정상적인 동작을 유도한다.

또한 기존의 클라이언트 쓰레드 내에, send/recv 의 실패 회수를 세어 일정한 threshold 값을 넘어서면 서버 데몬 프로세스의 crash 에 대한 관리자의 부적절한 대응으로 판단하여, 이에 대해 fail over 를 수행할 수 있는 부가적인 기능을 삽입하였다. 이 때의 threshold 값은 시스템 관리 차원에서 설정될 수 있는 옵션으로 실제 시스템 이용 시에 조율할 수 있을 것이다. 그림 4 는 이러한 구현의 순서도를 나타내고 있다.



(a) Crash Checking Thread (b) Client Thread (c) Resource Allocator

[그림 4] 주요 구현 요소들의 순서도

3.3 복구된 서버에 대한 처리

Crash 가 발생한 서버 노드들이 복구되었을 때는 자

원 할당자가 이를 다시 Server Pool 로 등록 시킬 수 있도록 하였으며 서버 노드의 복구 유무 판별은 문제를 간단히 하기 위해 자원 할당자가 자원 모니터로의 연결을 시도해 성공하면 복구된 것으로 간주하도록 하였다.

4. 실험 및 고찰

16 노드의 pc 클러스터 시스템[12]에서 한 노드를 클라이언트로 하고 나머지 노드를 서버로 하여 그 중 3 대의 노드에 패리티 그룹을 형성하고 간단히 fail over 기능을 검증해 보았다. 노드 crash 테스트를 위해 기 연결된 서버 중 임의의 한 노드를 shutdown 하고 fail over 되는 상황을 확인하였다. 그리고 한 서버의 데몬 프로그램을 종료하고 설정된 threshold 의 시간이 지난 후 역시 fail over 가 일어나는 것을 확인 할 수 있었다. 또한 shutdown 한 노드를 reboot 하여 서버 데몬 프로그램을 실행시킨 후 자원 할당자가 이를 감지하고 server pool 에 등록하는 것을 확인할 수 있었고, 그 서버를 클라이언트가 다시 새로운 클라이언트가 구동되거나 기존의 클라이언트가 fail over 를 수행하면서 이 서버를 패리티 그룹 멤버로 이용하는 것을 확인 할 수 있었다.

이렇게 단순한 기능을 확인해 보았으나 여기서 이런 기능이 얼마나 신뢰성을 제공하는 지에 분석적 방법이 요구된다. 앞으로 MTTF(Mean Time To Failure)와 MTTR(Mean Time To Repair)를 실험을 통해 측정하여 신뢰도를 정성적으로 분석하는 것이 필요할 것이다. 또한 성능 테스트를 통하여 기존의 NRD 에 대해 어느 정도의 오버헤드를 갖고 있는지에 대한 실험도 요구된다.

5. 결론 및 향후 과제

물리적 저장 장치의 접근 지연 시간을 극복하려는 시도로 연구된 네트웍 램디스크의 시험적 구현이 실험을 통해 만족할만한 성능을 드러냈고, 또한 앞으로의 네트웍 기반 기술의 발전과 메모리 디바이스의 질적 향상과 고 용량화는 이런 시도에 대해 고무적인 분위기를 조성하고 있다. 본 논문에서는 네트웍 램디스크가 가지고 있는 큰 취약점에 대해서 알아보았고 그에 대한 해결책들을 기술하고 구현하였다.

무엇보다 이렇게 신뢰성에 초점을 맞추어 질적향상을 꽤 하였으나, 적절한 성능 평가에 대한 방법론이 애매한 것이 바로 신뢰성 연구의 어려움이다. 앞으로 평가 방법에 대해 연구해 보아야 할 것이고, 다수의 패리티 그룹을 통한 확장성에 대한 연구가 필요할 것이다.

참고문헌

[1] Jeanna Neefe Matthews "Improving File System Performance With Adaptive Methods," Ph.D. Dissertation, December 1999.

[2] SAN(Storage Area Network), *정보과학회지 2001 3 월*.

[3] Sandberg, R. "The Sun Network File System: design, implementation, and experience," Sun Microsystems, Inc. 1986.

[4] Steven R. Soltis, Thomas M. Ruwart, Matthew T. O'Keefe, "The Global File System", Proceedings of the Fifth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies, September 17-19, 1996, College Park, MD.

[5] Michail D. Flouris and Evangelos P. Markatos, "The Network RamDisk: Using Remote Memory on Heterogeneous NOWs", in Cluster Computing, Special Issue on I/O in Shared-Storage Clusters, 2(4), pp. 281-293, 1999, Baltzer Science Publishers.

[6] J. Hartman and J. Ousterhout, "The Zebra Striped Network File System," In Proceedings of the 14th ACM Symposium on Operating Systems Principles, pages 29-43, December 1993.

[7] Rajkumar Buyya, *High Performance Cluster Computing*, Prentice-Hall, Inc., 1999.

[8] Anurag Acharya and Sanjeev Setia, "Availability and Utility of Idle Memory in Workstation Clusters", SIGMETRICS 1999.

[9] M. Resenblum, J. Ousterhout, "The Design and Implementation of a Log-Structured File System," ACM Trans. On Computer Systems, 10(1): 26-52, February 1992.

[10] Pradeep K. Sinha, *Distributed Operating Systems: Concepts and Design*, IEEE, Inc., 1997.

[11] Gregory F. Pfister, *In search of Clusters*, Revised updated version. Prentice-Hall, Inc., 1998.

[12] KNIGHTS, http://knights.kjist.ac.kr