안정적인 네트웍 램의 설계와 구현

황인철[0] 정한조 맹승렬 조정완
한국과학기술원 전자전산학과
{ichwang, hanjo, maeng, jwcho}@camars.kaist.ac.kr

The Design and Implementation of Reliable Network RAM on Linux

In-Chul Hwang[0] Han-Jo Jung Seung-Ryoul Maeng Jung-Wan Cho
Dept. of Electrical Engineering & Computer Science, KAIST

요 약

기존 운영체제들은 물리적 메모리보다 더 많은 양의 메모리를 사용자에게 제공하기 위하여 가상 메모리 페이징 시스템을 사용한다. 가상 메모리 페이징 시스템에서는 물리적 메모리가 부족해지면 그 내용을 저장시킬 수 있는 스왑 장치를 필요로 하는데, 기존 운영체제들에서는 디스크를 스왑 장치로 사용한다. 디스크는 물리적 메모리에 비해 그 접근 속도가 매우 느리기 때문에 상대적으로 스왑핑이 일어나면 물리적 메모리의 접근 시간에 비해 엄청난 시간을 기다려야 한다. 여러 대의 컴퓨터를 빠른 네트웍으로 묶는 클러스터 환경에서는 디스크의 접근 시간보다 네트웍을 통하여 다른 워크스테이션의 메모리에 접근하는 시간이 더 빠르기 때문에 유효한 다른 워크스테이션의 메모리를 스왑 공간으로 사용하고자 하는 네트웍 램이 제시되었다. 본 논문에서는 Linux 운영체제에서 스왑 장치 관리자로 네트웍 램을 설계, 구현하여 그 성능을 측정하였다. 그리고 새로운 안정성 제공 방법을 제시하고 기존에 제시된 안정성 제공방법들과 비교, 평가하였다.

1. 서론

최근 운영체제들은 가상 메모리 페이징을 지원함으로써 응용 프로그램에게 물리적 메모리보다 더 많은 양의 메모리를 사용할 수 있게 해 준다. 가상 메모리 페이징은 응용 프로그램에서 요구하는 일부분 만을 물리적 메모리에 사상시키고 부족한 양을 스왑 장치에 사상시키는 방법을 사용한다. 기존의 운영체제들에서는 스왑 장치로 디스크를 사용한다. 디스크는 물리적 메모리에 보편적으로 사용되는 램에 비해 가격이 저렴하고 대용량의 저장 공간을 제공하지만 접근시간은 램보다 훨씬 길다[2]. 따라서 페이지 오류 발생 후 디스크에 사상된 페이지에 접근하는 것은 물리적 메모리에 있는 페이지에 접근하는 것 보다 상대적으로 오랜 시간을 기다려야 한다.

요즘 많은 컴퓨터를 빠른 네트웍으로 묶어 연산을 빠르게 하거나 여러 컴퓨터의 자원을 공유하자는 클러스터 컴퓨팅이 제안되고 있다. 이러한 클러스터 환경에서는 디스크 접근 시간보다 네트웍에 연결된 다른 워크스테이션의 메모리에 대한 접근 시간이 빠르다. 이러한 클러스터 환경에서 빠른 스왑 장치로 제안된 것이 네트웍 램이다[2].

기존 네트웍 램에 대한 연구들에서는 요즘 클러스터 환경에서 선호되는 운영체제인 Linux를 기반으로 하고 있는 것이 구현되어 있지 않다. Linux는 다른 운영체제에 비해 자유롭게 쓸 수 있는 장점이 있어서 많은 클러스터 환경 에서 사용되고 있다. 본 논문에서는 운영체제가 Linux인 시스템에서 네트웍 램을 설계하고 구현한다. 그리고 기존 연구에서 제시된 안정성 제공 방법들은 기존의 RAID[3] 시스템에서 제시하고 있는 방법에서 크게 벗어나지 못하고 있는 기존 연구에서 제시된 안정성 제공 방법들에 대하여, 본 논문에서는 새로운 안정성 제공방법을 제시하고 구현을 통하여 기존의 연구에서 제시 된 방법들과 비교, 평가한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 네트웍 램에대한 연구들과 네트웍 램의 안정성 제공 방법들을 살펴본다. 3장에서는 네트웍 램의 디자인과 구현 방법에 대하여 살펴본다. 4장에서는 구현한 네트웍 램의 성능 평가와 안정성 제공 방법들의 성능을 비교분석한다. 마지막으로 5장에서는 결론을 맺는다.

2. 관련연구

2.1 네트웍 램의 구현

구현하는 방식은 크게 세가지로 나눌 수 있다. 첫째는 기존의 운영체제를 수정하지 않고 메모리 할당에 관련된 사용자 라이브러리를 제공함으로서 네트웍 램을 구현하는 방법이다[1]. 이 방법은 운영체제를 수정할 필요가 없다는 장점이 있지만 응용 프로그램이 사용자 라이브러리에 맞게 고쳐져야 하고 사용자 레벨에서 메모리를 관리하기 때문에 성능이 좋지 않다는 단점이 있다. 둘째는 기존의 운영체제를 고치지 않고 단지 스왑 장치 관리자를 이용하여 구현하는 방법이다[4]. 스왑 장치 관리자를 이용하여 구현하는 방법은 사용자 레벨보다 좋은 성능을 얻을 수 있으며 기존의 운영체제를 고치지 않아도 된다는 장점을 지니고 있다. 하지만 네트웍 램을 페이징 시스템에서만 사용하게 된다는 단점이 있다. 셋째는 운영체제를 수정하여 여러 워 크스테이션의 메모리를 전체적으로 관리하는 방법이다[5]. 이 방법은 클러스터 메모리를 전체적으로 관리함으로서 가장 좋은 성능을 나타내지만 운영체제를 고쳐야 하는 어려움과 이식성이 떨어진다는 큰 단점이 있다.

2.2 안정성 제공 방법

램 자체는 전원이 꺼지면 가지고 있던 내용 자체가 모두 사라지는 속성이 있다. 만약 현재의 워크스테이션의 메모리를 다른 워크스테이션에 제공하고 있는 상태에서 워크스테이션의 메모리가 손상되면 그 워크스테이션의 메모리에 페이징을 하고 있던 다른 워크스테이션의 메모리 내용도 손상되게 된다. 이러한 일을 방지하기 위하여 네트웍 램을 사용할 때는 원본 메모리 내용을 복구할 수 있는 안정성 제공 방법을 고려하여야 한다. 이러한 방법에는 여러 개의 복사본을 두는 방법, 패리티를 두는 방법, 동적인 집합들의 패리티 두는 방법 등이 제시되었다[1,4].

3. 네트웍 램의 설계와 구현

본 논문에서는 클러스터 시스템에서 스왑 장치로 사용할 수 있는 네트웍 램을 구현하였다. 본 절에서는 네트웍 램의 구조, 그리고 네트웍 램을 이루고 있는 각각의 동작과정에 대하여 서술한다.

3.1 네트웍 램의 구조

본 논문에서 구현한 네트웍 램의 구조는 다음 [그림 1]과 같다.

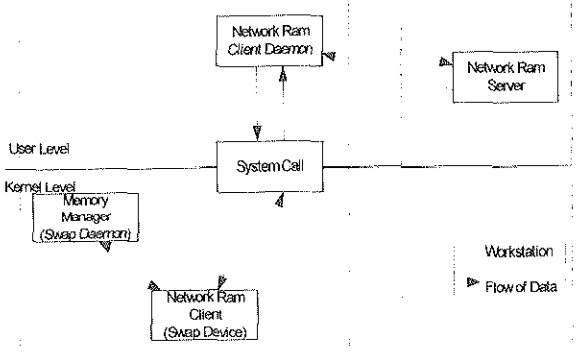

그림 1. 네트웍 램의 구조

네트웍 램은 스왑 데몬, 클라이언트, 서버, 그리고 클라이언트 데몬 네 부분으로 구성된다. 각 부분이 하는 역할은 다음과 같다.

- 스왑 데몬: 물리적 메모리가 부족해 지면 메모리 관리 서브시스템은 물리적 메모리를 해제 하려고 한다. 이 일은 커널 스왑 데몬에게 할당된다. 스왑 데몬은 타이머를 두고서 타이머가 만료될 때마다 주기적으로 깨어나 시스템의 사용 가능한 페이지 수가 어느 기준보다 적어지게 되면 스왑 장치에 페이지의 내용을 전달하고 그 페이지를 사용할 수 있게 해 주는 역할을 한다.
- 클라이언트: 메모리가 부족해지면 스왑 데몬이 클라이언트에게 페이지 아웃될 페이지의 내용과 쓰여질 장소를 넘겨주게 된다. 그러면 클라이언트는 그 내용을 클라이언트 데몬에게 넘겨줘어 처리하게 하는 역할을 한다. 만약 페이지의 내용이 필요해지면 페이지가 저장되어 있는 장소를 스왑 데몬이 클라이언트에게 넘겨주고 클라이언트는 그 내용을 클라이언트 데몬에게 전달되어 처리하게하는 역할을 한다.
- 서버: 클라이언트에게 자신의 메모리를 제공해 주는 역할을 한다. 즉, 클라이언트 데몬에서 메모리를 요구하면 자신의 메모리의 일부를 제공해 주고 클라이언트 데몬에서 저장된 내용을 요구하면 그 내용을 전달해 주는 역할을 한다.
- 클라이언트 데몬 : 클라이언트와 서버 사이의 통신과 서버들의 관리, 페이지들의 위치정보에 관한 관리를 한다.

3.2 안정성 제공 방법

[그림 2]는 본 논문에서 제시한 안정성 제공 방법에 대한 그림이다.

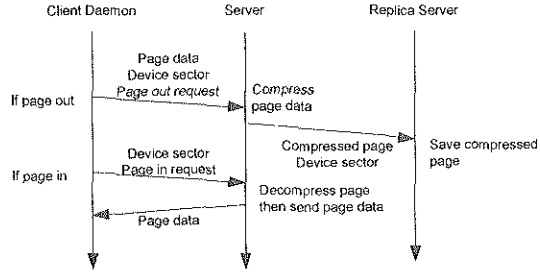

그림 2. 압축을 이용한 복사본을 두는 방법

페이지 아웃이 일어나서 페이지의 내용이 전달되어 오면 그 내용을 전달받은 서버에서 적절한 압축 알고리즘에 의해 그 페이지의 내용을 압축시키고 저장한다. 이후 압축된 복사본을 다른 서버에게 전달하여 저장하게 한다. 이와 같은 방법을 사용하면 압축에 의해 사용되는 메모리의 양을 줄일 수 있고, 네트웍을 통하여 전달되는 내용의 양을 줄일 수 있는 장점이 있다.

4. 네트웍 램의 구현

본 논문에서는 스왑 데몬을 제외한 모든 부분을 구현하였다. 본 절에서는 각 부분의 구현에 대하여 자세히 서술한다.

4.1 클라이언트

클라이언트는 스왑 장치 관리자로 구현된다. 스왑 장치 관리자는 블록 장치 관리자로서 임의의 많은 내용을 한꺼번에 전송할 수 있는 장치 관리자이다. 블록 장치 관리자에서는 블록장치에 데이터를 쓰거나 읽을 때 그 장치에 대한 요구를 연결 리스트로 묶어서 처리해 주는 처리함수를 정의 하게 된다. 쓰기 요구나 읽기 요구가 일어났을 때 그 정보들을 클라이언트 데몬을 넘겨주기 위해 ioctl 함수를 이용하여 구현하였다.

4.2 클라이언트 데몬

클라이언트 데몬은 사용자 프로세스로서 클라이언트로부터 넘어온 정보를 가지고서 서버와 TCP/IP 소켓을 사용하여 통신을 하고 요구를 처리하는 역할을 한다. 서버에 대한 정보와 페이지의 위치 정보를 테이블로 저장하여 관리한다.

4.3 서버

서버는 페이지의 내용을 일정 부분에 저장하는 사용자 프로세스로 구현하였다. 나중에 안정성 제공 방법을 구현할 때는 페이지를 압축하여 저장하고 압축한 복사본을 다른 서버에 저장한다.

4.4 안정성 제공 방법

본 논문에서는 기존에 제시된 복사본을 두는 방법, 패리

티, 패리티 로깅과 본 논문에서 제시한 압축을 이용한 복사본을 두는 방법을 모두 구현하였다.

5. 성능 평가

본 장에서는 네트웍 램의 구현 환경과 구현된 네트웍 램의 성능 평가, 그리고 안정성 제공 방법들의 성능을 비교, 분석한다.

5.1 구현 환경

구현은 128Mbyte의 메모리를 갖는 Pentium-III 클라이언트와 256Mbyte의 메모리를 갖는 Alpha 21164서버들을 100Mbps 이더넷으로 연결하여 구성된 클러스터 시스템에서 이루어 졌다.

5.2 기본적인 성능 평가

구현된 시스템에서 부분별 수행시간의 측정결과는 다음과 같다.

- 네트웍 지연 시간(4Kbyte) : 0.7ms
- 디스크 지연 시간(4Kbyte) : 3.4ms
- 압축 & 압축 푸는 시간(4Kbyte) : 1.3ms & 0.02ms

5.3 네트웍 램의 성능 측정

본 절에서는 120Mbyte의 디스크를 스왑 장치로 사용하는 시스템과 같은 양의 네트웍 램을 스왑 장치로 사용하는 시스템의 성능을 비교한다. 성능 측정을 위해 사용된 프로그램은 MVEC과 GAUSS, QSORT로서 데이터의 크기는 약 200Mbyte정도의 크기이다.

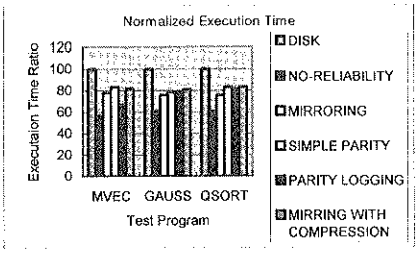

그림 3. 수행 시간

[그림 3]은 수행시간을 나타낸다. 네트웍 램을 스왑 장치로 사용한 시스템에서의 수행시간이 디스크를 스왑 장치로 사용한 시스템에서의 수행시간에 비하여 평균 40.3% 더 빠르다는 것을 알 수 있다. 그리고 안정성 제공 방법들을 살펴보면 본 논문에서 제시한 압축을 이용한 복사본을 두는 방법은 기존의 다른 방법들과 유사한 성능을 보인다.

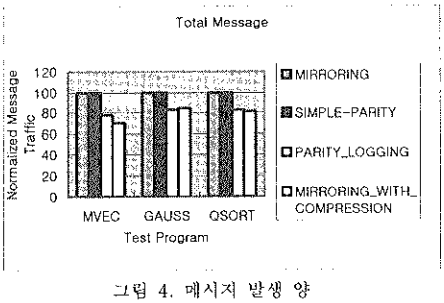

그림 4. 메시지 발생 양

[그림 4]는 안정성 제공 방법들의 메시지의 양을 나타낸다. 본 논문에서 제시한 압축을 이용한 복사본을 두는 방법이 다른 방법들에 비해 비교적 적은 메시지의 양을 발생시킨다.

[그림 5]는 각 안정성 제공 방법을 사용하였을 때 네트웍 램의 서버가 사용하는 메모리의 양을 나타낸다. 압축을 이용한 복사본을 두는 방법이 다른 방법들에 비해 압축을 이용하기 때문에 훨씬 적은 양의 서버 메모리를 사용한다.

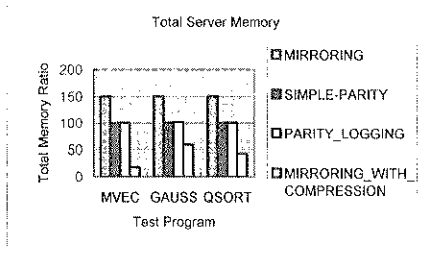

그림 5. 서버 메모리 사용 량

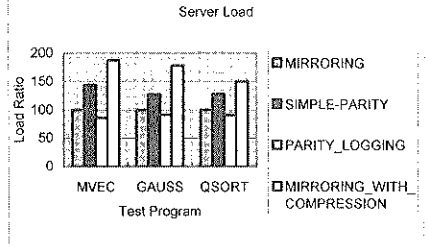

그림 6. 서버 부하

[그림 6]는 안정성 제공 방법들이 서버에 주는 부하의 양을 측정한 결과이다. 압축을 이용한 복사본을 두는 방법은 가장 많은 서버 부하를 발생시킨다.

6. 결론

본 논문에서는 Linux를 운영체제로 사용하는 클러스터 환경에서 네트웍 램을 구현하였다. 디스크를 스왑 장치로 사용하는 시스템보다 네트웍 램을 스왑 장치로 사용하는 시스템이 프로그램 수행 속도에 있어 평균 40.3%의 성능향상이 있었다. 그리고 본 논문에서 제시한 새로운 안정성 제공 방법인 압축을 이용한 복사본을 두는 방법은 기존의 다른 방법들에 비해 더 적은 서버 메모리와 메시지를 사용하여 유사한 성능을 나타내었다.

7. 참고 논문

[1] E. Anderson and J. Neefe. An Exploration of Network RAM. Technical Report CSD-98-1000, Computer Science Division, University of California, Berkeley, July 1998.

[2] Rajkumar Buyya. *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall, 1999.

[3] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson. RAID: High-Performance, Reliable Secondary Storage. *ACM Computing Surveys*, 26(2):145-185, June 1994.

[4] George Dramitinos and Evangelos P. Markatos. Adaptive and Reliable Paging to Remote Main Memory. *Journal of Parallel and Distributed Computing*, 58(3):357-388, September 1999.

[5] M. J. Feeley, W. E. Morgan, F. H. Pighin, A. R. Karlin, H. M. Levy, and C. A. Thekkath. Implementing Global Memory Management in a Workstation Cluster. *In Proceedings of the 15th Symposium on Operating Systems Principles*, December 1995.