리눅스에서 압축을 이용한 안정적인 네트웍 램의 설계 및 구현

(The Design and Implementation of the Reliable Network RAM using Compression on Linux)

황 인 철 † 정 한 조 † 맹 승 렬 †† 조 정 완 ††

(In-Chul Hwang) (Han-Jo Jung) (Seung-Ryoul Maeng) (Jung-Wan Cho)

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 교체되는 메모리 내용을 저장시킬 수 있는 스왑 장치를 필요로 하는데, 기존 운영 체제들에서는 디스크를 스왑 장치로 사용한다. 디스크는 물리적 메모리에 비해 접근 속도가 매우 느리기 때문에 스왑핑이 일어나면 물리적 메모리의 접근 시간에 비해 많은 시간을 기다려야 한다.

여러 대의 PC를 빠른 네트웍으로 묶는 클러스터 환경에서는 디스크의 접근 시간보다 네트웍을 통하여 다른 워크스테이션의 메모리에 접근하는 시간이 더 빠르기 때문에 사용 가능한 다른 워크스테이션의 메모리를 디스크 대신 빠른 장치로 사용하고자 하는 네트웍 램이 제시되었다.

본 논문에서는 Linux 운영 체제에서 스왑 장치 관리자로 네트웍 램을 설계, 구현하여 디스크를 스왑 장치로 사용하는 시스템보다 네트웍 램을 스왑 장치로 사용하는 시스템이 프로그램 수행 속도에 있어 평균 40.3%의 성능 향상이 있었다. 그리고 기존 RAID 시스템에서 사용하던 안정성 제공 방법과 다른 프로세서의 성능을 효율적으로 이용하는 새로운 안정성 제공방법을 제시하였고 평가 결과 본 논문에서 제시한 새로운 안정성 제공 방법인 압축을 이용한 복사본을 두는 방법은 적은 서버 메모리와 메시지를 사용하여 유사한 성능을 나타낸다.

키워드 : 네트웍 램, 안정성 제공 방법, 리눅스

Abstract Traditional operating systems use a virtual memory to provide users with a bigger memory than a physical memory. The virtual memory augments the insufficient physical memory by the swap device. Since disks are usually used as the swap device, the cost of a page fault is relatively high compared to the access cost of the physical memory. Recently, numerous papers have investigated the Network RAM in order to exploit the idle memory in the network instead of disks. Since today's distributed systems are interconnected with high-performance networks, the network latency is far smaller than the disk access latency.

In this paper we design and implement the Network RAM using block device driver on Linux. This is the first implementation of the Network RAM on Linux. We propose the new reliability method to recover the page when the other workstation's memory is damaged. The system using the Network RAM as the swap device reduces the execution time by 40.3% than the system using the disk as the swap device. The performance results suggest that the new reliability method that use the processor more efficiently has the similar execution time with others, but uses smaller server memory and generates less message traffic than others.

Key words : Network RAM, Reliable Method, Linux

† 비 회 원 : 한국과학기술원 전산학과
ichwang@camars.kaist.ac.kr
hanjo@camars.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 교수
maeng@camars.kaist.ac.kr
jwcho@camars.kaist.ac.kr

논문접수 : 2002년 6월 18일
심사완료 : 2003년 1월 23일

1. 서 론

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 메모리의 일부분만을 물리적 메모리에 사상시키고 부족한 양을 스왑 장치에 사상하는 방법이다. 기존의 운영 체제들에서는 스왑 장치로 디스크를 사용한다. 디스크는 물리적 메모리에 보편적으로 사용되는 램(Random Access Memory)에 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. 클러스터 환경에서는 디스크 접근 시간보다 네트웍에 연결된 다른 워크스테이션의 메모리에 대한 접근 시간이 빠르다. 따라서 이러한 클러스터 환경에서 디스크 대신 빠른 스왑 장치로 제안된 것이 네트웍 램이다[2].

기존 네트웍 램에 대한 연구들에서는 요즘 클러스터 환경에서 많이 사용되는 운영 체제인 Linux를 기반으로 하고 있는 것이 구현되어 있지 않다. Linux는 다른 운영체제에 비해 자유롭게 쓸 수 있는 장점이 있어 많은 클러스터 환경에서 사용되고 있다. 본 논문에서는 Linux를 운영체제로 사용하는 시스템에서 네트웍 램을 설계하고 구현한다.

그리고 기존에 네트웍 램에서 사용된 안정성 제공 방법들은 RAID[3] 시스템에서 제시하고 있는 방법에서 크게 벗어나지 못하고 있다. 따라서 본 논문에서는 네트웍이나 메모리 접근 시간에 비해 상대적으로 매우 빠른 프로세서를 효율적으로 이용하는 새로운 안정성 제공 방법을 제시하고 구현을 통하여 기존의 연구들에서 제시된 안정성 제공 방법들과 성능을 비교하고 평가하였다. 본 논문에서 제시한 압축을 이용한 복사본을 두는 방법은 기존 안전성 제공 방법들에 비하여 많은 서버의 부하를 요구한다는 단점이 있었다. 하지만 이 점은 프로세서의 성능 증가가 메모리나 기타 부분의 성능 증가보다 빠르기 때문에 큰 문제가 되지 않는다. 그리고 기존 안정성 제공 방법들에 비하여 사용하는 서버 메모리의 양이 적고 메세지의 양이 적다는 장점이 있다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 네트웍 램에 대한 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 비교, 분석한다. 마지막으로 5장에서는 결론을 맺는다.

2. 관련연구

본 장에서는 기존에 이루어진 네트웍 램의 구현 방법들과 안정성 제공 방법들을 살펴본다.

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 측면에서 메모리를 관리하기 때문에 성능상의 부하가 많다는 단점이 있다. 둘째는 기존의 운영체제를 고치지 않고 장치 관리자(device driver)를 이용하여 구현하는 방법이다[4]. 장치 관리자를 이용하여 구현하는 방법은 사용자 측면보다 좋은 성능을 얻을 수 있으며 기존의 운영체제를 고치지 않아도 된다는 장점을 지니고 있다. 하지만 네트웍 램을 페이징 시스템에 국한하여 사용한다는 단점이 있다. 셋째는 운영체제를 수정하여 여러 워크스테이션의 메모리를 전체적으로 관리하는 방법이다[5]. 이 방법은 클러스터 메모리를 시스템 전체적으로 관리함으로서 다른 방법들에 비해 가장 좋은 성능을 나타내지만 운영 체제를 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 사라지게되면 그 워크스테이션의 메모리에 페이징을 하고 있던 다른 워크스테이션의 메모리 내용도 사라지게 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(mirroring), 패리티를 두는 방법(parity), 동적인 집합들의 패리티 두는 방법(parity logging) 등이 제시되었다[1,4]. 다음 그림은 이에 대한 방법들에 대한 그림이다.

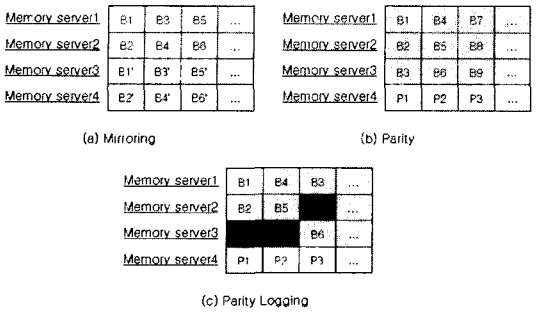

그림 1 기존 안정성 제공 방법

그림에서 보는 바와 같이 복사본을 두는 방법(a)는 네개의 서버 중 절반인 두개가 복사본 서버의 역할을 함으로서 전체 서버 메모리의 절반 밖에 사용할 수 없다는 단점이 존재한다. 패리티 방법은 일정 블럭들을 하나의 패리티 집합으로 묶어 하나의 서버가 그에 대한 패리티 블럭을 저장하게 된다. 이 방법은 패리티 집합의 하나의 블럭만 바뀌어도 패리티 블럭에 영향을 주기 때문에 패리티 서버에 부하가 집중되는 단점이 존재한다. 마지막으로 패리티 로깅 방법은 기존 패리티 방법에 대한 단점을 보완한 것으로 일정한 집합이 아닌 필요한 블럭들(예를 들면 접근되는 순서에 의한 블럭들)을 동적으로 패리티 집합으로 묶어 그에 대한 패리티 블럭을 두는 방법이다. 그림에서와 같이 기존에 묶여 있던 블럭(B3,B6,B8)이 고쳐 졌을 경우에는 이전 블럭은 그냥 두고서 새로이 패리티 집합으로 구성하여 새로운 패리티 블럭 값을 계산하게 된다. 이 방법의 단점은 추후 디스크 용량이 부족하여 졌을 때 이전의 고쳐진 블럭들에 대한 garbage collection을 해야 한다는 점이다.

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 2와 같다.

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 서브 시스템은 물리적 메모리를 해제 하려고

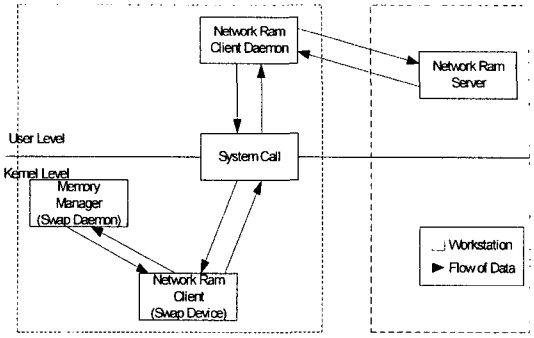

그림 2 네트웍 램의 구조

한다. 이 일은 커널 스왑 데몬이 수행한다. 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. 이 부분은 Linux에 이미 구현되어 있다.

- 클라이언트 : 메모리가 부족해지면 스왑 데몬이 클라이언트에게 페이지 아웃 될 페이지의 내용과 쓰여질 장소를 넘겨주게 된다. 그러면 클라이언트는 그 요구 사항을 클라이언트 데몬에게 넘겨주어 처리하게 하는 역할을 한다. 만약 페이지의 내용이 필요해지면 페이지가 저장되어 있는 장소를 스왑 데몬이 클라이언트에게 넘겨주고 클라이언트는 그 내용을 클라이언트 데몬에게 전달되어 처리하게한다.
- 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.
- 클아이언트 데몬 : 클라이언트와 서버 사이의 통신과 서버들의 관리, 페이지들의 위치정보에 관한 관리를 한다.

3.2 안정성 제공 방법

요즘 PC에서 CPU는 다른 부품에 비하여 그 성능이 빠르게 향상되고 있다. 본 논문에서는 복사본을 둘 때 CPU 부하는 조금 늘어나지만 메세지 전달이나 내용의 저장에 큰 잇점을 가지고 있는 압축 방법을 사용하여 새로운 안정성 제공 방법을 설계하였다.

그림 3은 본 논문에서 제시한 안정성 제공 방법에 대한 그림이다.

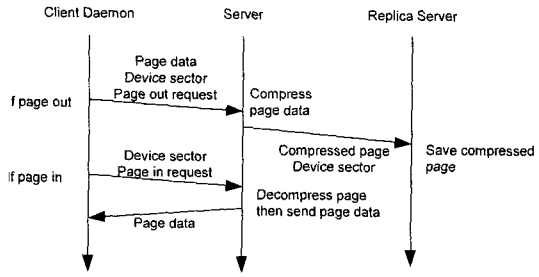

그림 3 압축을 이용한 복사본을 두는 방법

페이지 아웃이 일어나서 페이지의 내용이 전달되어 오면 그 내용을 전달받은 서버에서 적절한 압축 알고리즘에 의해 그 페이지의 내용을 압축시키고 저장한다. 이후 압축된 복사본을 복사본을 저장하는 서버에게 전달하여 저장한다. 이와 같은 방법을 사용하면 압축 알고리즘에 의해 메모리 내용을 압축함으로서 사용되는 메모리의 양을 줄일 수 있고, 네트웍을 통하여 전달되는 내용의 양을 줄일 수 있는 장점이 있다. 다음 그림 4는 이 방법에 대한 블럭들의 분포도를 보여준다.

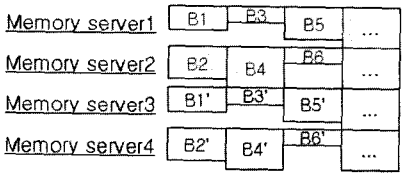

그림 4 압축을 이용한 복사본을 두는 방법

그림 4에서와 같이 기본적으로는 기존의 복사본을 두는 방법과 유사한 모습을 보이지만, 압축을 사용하여 블럭의 내용을 압축하여 전달 및 저장다는 장점이 존재하게 된다.

4. 네트웍 램의 구현

본 논문에서는 리눅스 운영체제에서 구현되어 있는 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 정의하게 된다. 이곳에서 스왑 데몬으로 부터 읽기나 쓰기 요구를 받아서 클라이언트에 전달할 큐에 저장하게 된다. 이 큐에 클라이언트 데몬이 접근하기 위하여 ioctl 함수를 이용하여 구현하였다. ioctl은 커널 모듈에서 시스템 호출과 같은 서비스를 사용자 프로그램에게 제공하여 주는 방법이다. 본 논문에서 구현한 네트웍 램에서는 ioctl을 정의하여 클라이언트 데몬이 계속적인 폴링을 통하여 요구 큐에 접근, 그 내용을 읽어오게 된다.

다음은 각 요구에 따른 작동 동작 순서는 다음과 같다.

- 읽기 요구 : 읽기 요구일 경우 블럭 번호와 읽기 요구라는 것을 클라이언트에 전달할 큐에 삽입한다. 이후 클라이 언트가 이 요구를 처리하여 주면 그 결과를 다시 스왑 데몬에게 전달하여 준다.
- 쓰기 요구 : 쓰기 요구일 경우 블럭 번호와 쓰기 요구, 그리고 쓸 내용을 클라이언트 데몬에게 전달할 큐에 삽입한다.

4.2 클라이언트 데몬

클라이언트 데몬은 사용자 프로세스로서 클라이언트로부터 넘어온 정보를 가지고서 요구를 처리하고 그 결과를 전달해 주는 역할을 수행한다. 계속적으로 클라이언트에 ioctl로 접근하여 새로운 요구가 들어오면 그 내용을 전달 받아 요구에 따라서 서버와 TCP/IP 소켓을 사용하여 통신을 하고 요구에 알맞는 처리를 수행한다. 서버에 대한 정보와 페이지의 위치 정보를 테이블로 저장하여 관리 한다.

다음은 각 요구에 따른 작동 동작 순서는 다음과 같다.

- 읽기 요구 : 읽기 요구가 일어날 경우 먼저 블럭이 존재하는 서버를 찾는다. 이는 일정한 알고리즘을 통하여 서버를 찾을 수 있다. 그리고 이 서버에 블럭을 요구하여 받아 온 후, 그 내용을 클라이언트에게 전달하여 준다.
- 쓰기 요구 : 쓰기 요구도 읽기 요구와 마찬가지로 블럭을 저장할 서버를 찾는다. 이후 그 서버에게 내용을 전달하여 저장하게 한다.

4.3 서버

서버는 페이지의 내용을 일정 부분에 저장하는 사용자 프로세스로 구현하였다. 단순히 클라이언트에서 요구된 메모리를 저장하는 역할을 수행한다.

다음은 각 요구에 따른 작동 동작 순서는 다음과 같다.

- 읽기 요구 : 읽기 요구일 경우 블럭의 내용을 찾아 클라이언트 데몬에게 내용을 전달하여 준다.
- 쓰기 요구 : 쓰기 요구일 경우 블럭의 내용을 클라이언트 데몬으로 부터 받은 후 그 내용을 알맞은 다른 서버에게(예를 들면 복사본을 두는 서버) 전달

하여 저장하게 한다.

4.4 안정성 제공 방법

본 논문에서는 기존에 제시된 복사본을 두는 방법, 패리티, 패리티 로깅 방법들과 본 논문에서 제시한 압축을 이용한 복사본을 두는 방법을 모두 구현하였다. 각 방법들은 서버의 역할을 변경시킴으로서 구현하였다.

복사본을 두는 방법은 두개의 원본 서버와 두개의 복사본 서버로 구성하여 서버가 쓰기 요구를 받을 경우 그 내용을 자신의 메모리에 저장할 뿐 아니라 그 내용을 복사본 서버에게 전달하여 저장하게 한다. 만약 서버 중 하나가 실패할 경우 다른 복사본 서버가 그 역할을 대신한다.

패리티와 패리티 로깅 방법은 세개의 원본 서버와 하나의 패리티 서버로 구성하여 쓰기 요구를 받을 경우 그 내용을 가지고 알맞은 패리티 블럭에 패리티를 계산하여 다시 패리티 서버에 그 내용을 저장하게 한다. 만약 서버 중 하나가 실패가 일어날 경우 패리티 블럭과 다른 패리티 집합의 블럭들을 가지고서 원본 블럭을 계산하여 사용한다.

본 논문에서 제시한 압축을 이용한 복사본을 두는 방법에서는 복사본을 두는 방법과 유사하지만 서버에서 압축을 수행한 후, 그 내용을 복사본 서버에게 전달하여 저장하게 한다. 이때 클라이언트 데몬은 단지 요구를 전달하여 준 다음 계속적으로 응용프로그램이 수행되기 때문에 압축을 하는 시간은 응용프로그램 수행 시간에 큰 영향을 미치지 않는다. 본 논문에서는 압축을 위하여 Lempel-Ziv 알고리즘을 사용하였다.

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 및 기본 수행 시간

구현은 20GByte E-IDE 디스크가 부착된 128Mbyte의 메모리를 갖는 Pentium-III 700MHz 클라이언트 한 대와 네대의 256Mbyte의 메모리를 갖는 Alpha 21164 서버들을 100Mbps 이더넷으로 연결하여 구성된 클러스터 시스템에서 이루어 졌다. 성능 측정을 위해 사용된 프로그램은 MVEC과 GAUSS, QSORT로서 MVEC은 두개의 벡터를 곱셈하는 프로그램이고 GAUSS는 가우스 제거법을 수행하는 프로그램, 그리고 QSORT는 데이터를 퀵 정렬하는 프로그램이다. 각 프로그램들은 약 200MByte의 데이터를 사용한다.

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

5.2 네트웍 램의 성능 측정

본 절에서는 120Mbyte의 디스크를 스왑 장치로 사용하는 시스템과 같은 양의 네트웍 램을 스왑 장치로 사용하는 시스템의 성능을 비교한다.

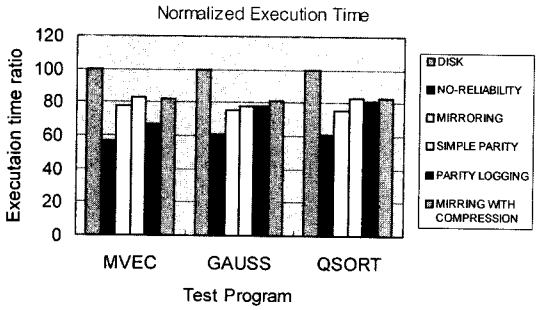

그림 5 수행 시간

그림 5는 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. 이는 디스크를 사용하는 것 보다 네트웍에 연결된 다른 클라이언트의 메모리를 스왑 장치로 사용하는 것이 더 빠르기 때문에 이러한 성능 차이를 나타낸다. 그리고 안정성 제공 방법들을 살펴보면 본 논문에서 제시한 압축을 이용한 복사본을 두는 방법은 기존의 다른 방법들과 유사한 성능을 보인다. 이는 압축을 이용하여 복사본을 두는 방법이 압축을 하고, 압축을 풀고 하는 작업이 실제 프로그램 수행 시간에 다른 방법들과 유사하게 영향을 미침을 나타낸다. 즉, 압축을 푸는 시간은 압축을 하는 시간과 네트웍 지연 시간에 비하여 매우 작은 값을 갖고, 압축을 하는 시간은 응용프로그램이 수행되는 것과 겹쳐 다른 안정성 제공 방법에 비해 성능에 대한 영향이 거의 없음을 알 수 있다.

그림 6은 안정성 제공 방법들의 메시지의 양을 나타낸다. 본 논문에서 제시한 압축을 이용한 복사본을 두는 방법이 다른 방법들에 비해 비교적 적은 메시지의 양을 발생시킨다. 이는 본 논문에서 제시한 방법은 압축을 이용하기 때문에 메시지 전달에 있어서 다른 방법들에 비하여 압축한 후 그 내용을 전달하기 때문에 메시지 양에서 장점을 갖는다는 것을 알 수 있다.

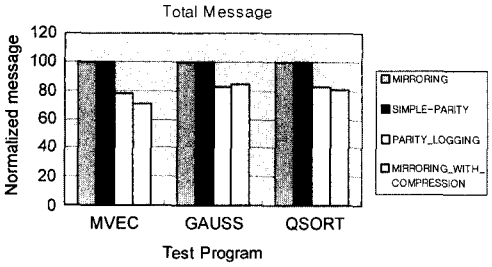

그림 6 메시지 발생 양

그림 7은 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. 즉, 압축율에 따라 그 결과가 다르지만 일반적으로 복사본을 두는 방법과 패리티, 패리티 로깅에 비해 약 11~40% 정도의 메모리만 사용한다는 것을 알 수 있다.

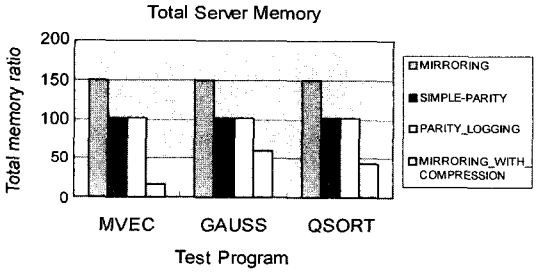

그림 7 서버 메모리 사용 량

그림 8은 안정성 제공 방법들이 서버에 주는 부하의 양을 측정한 결과이다. 압축을 이용한 복사본을 두는 방법은 서버에서 요구 처리후 그 내용을 압축해야 하기 때문에 다른 방법들에 비하여 가장 많은 서버 부하를 발생시킨다. 그림에서 보는 바와 같이 압축을 이용한 복사본을 두는 방법은 다른 방법들에 비하여 약 1.5~2배의 서버 부하를 발생시킨다. 이점은 서버 설계시 고려해 주어야 하나 앞으로의 프로세서 성능증가가 메모리나 기타 부분의 성능 증가보다 빠르기 때문에 큰 문제가 되지 않을 것이다.

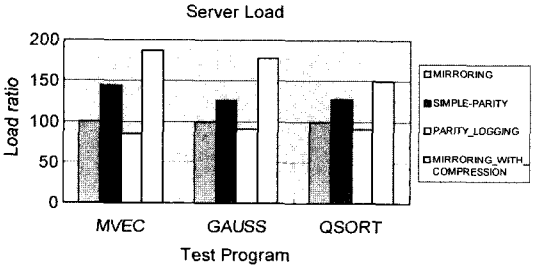

그림 8 서버 부하의 양

6. 결 론

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 기존의 운영체제에서는 디스크를 스왑 장치로 사용한다. 하지만 디스크는 물리적 메모리에 비해 그 접근 속도가 매우 느리기 때문에 상대적으로 스왑핑이 일어나면 엄청난 시간을 기다려야 한다.

여러 대의 컴퓨터를 빠른 네트웍으로 묶는 클러스터 환경에서는 디스크에 접근하는 시간보다 네트웍을 통해 다른 워크스테이션의 메모리에 접근하는 시간이 더 빠르기 때문에 유효한 다른 워크스테이션의 메모리를 스왑 공간으로 활용하고자 하는 네트웍 램이 제시되었다.

본 논문에서는 Linux를 운영체제로 사용하는 클러스터 환경에서 네트웍 램을 구현하였다. 그리고 기존 시스템에서 사용된 RAID에서 제시된 안정성 제공 방법과 다른 새로운 안정성 제공 방법을 제시하고 구현하여 그 성능을 다른 방법들과 비교하였다. 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 비하여 많은 서버의 부하를 요구한다는 단점이 있었다. 하지만 이 점은 프로세서의 성능 증가가 메모리나 기타 부분의 성능 증가보다 빠르기 때문에 큰 문제가 되지 않는다. 그리고 본 논문에서 제시한 안정성 제공 방법은 기존 안정성 제공 방법들에 비해 더 적은 서버 메모리와 메시지를 사용하여 유사한 성능을 나타내었다.

참 고 논 문

[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 Com-

puting: 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.

황 인 철

1995. 3~1999. 2 한국과학기술원 전산학과 졸업(학사). 1999. 3~2001. 2 한국과학기술원 전산학과 졸업(석사). 2001. 3~현재 한국과학기술원 전산학과 재학(박사). 관심분야 : 클러스터 컴퓨팅, 분산 파일 시스템, 분산 공유 메모리, 그리드 컴퓨팅

정 한 조

1991. 3~1995. 2 한국과학기술원 전산학과 졸업(학사). 1995. 3~1997. 2 한국과학기술원 전산학과 졸업(석사). 1997. 3~현재 한국과학기술원 전산학과 재학(박사). 관심분야 : 클러스터 컴퓨팅, 분산 파일 시스템, 분산 공유 메모리

맹 승 렬

1973. 3~1977. 2 서울대학교 전자공학과 졸업(학사). 1977. 3~1979. 2 한국과학기술원 전산학과 졸업(석사). 1979. 3~1984. 2 한국과학기술원 전산학과 졸업(박사). 관심분야 : 컴퓨터 구조, 병렬 시스템, 클러스터 컴퓨팅, 그리드 컴퓨팅

조 정 완

1964년 서울대학교 공과대학 전자공학과 졸업. 1968년 와이오밍 주립대학 전기공학과 석사학위 취득. 1973년 노스웨스턴 대학 전산학과 박사학위 취득. 1968년~1973년 미국 IBM 연구원. 1982년~1984년 삼성전자 고문. 1985년~1988년 금성소프트웨어 대표, 고문. 1973년~현재 한국과학기술원 전산학과 교수. 관심분야는 computer architecture, parallel processing, knowledge systems, man-machine interfaces