論文 97-22-6-14

NOW(Network of Workstations) 환경에서 소프트웨어 RAID 파일 시스템의 설계 및 성능 평가

正會員 김 종 훈*, 노 삼 혁*, 원 유 헌*

Design and Performance Evaluation of the Software RAID File System in the NOW Environment

Jong-Hoon Kim*, Sam H. Noh*, Yoo-Hun Won* *Regular Members*

요 약

최근 프로세서와 네트워크의 급속한 발전으로 인해 병렬 프로그램을 분산 시스템 환경에서 수행시키고자 하는 NOW(Network of Workstations)가 대두되고 있다. 이러한 컴퓨팅 자원의 빠른 발전과는 상대적으로 입출력 자원의 발전은 이를 따라가지 못하고 있다. 그로 인하여 여전히 디스크 입출력이 시스템 전체 성능의 큰 병목으로 지목되고 있다. 이와 같은 입출력의 병목 문제를 해결하기 위해서는 파일 시스템의 효율적인 지원이 있어야 한다. 분산 환경에서 높은 신뢰성과 성능을 제공하기 위해 나온 개념이 소프트웨어 RAID 파일 시스템이다. 본 논문에서는 소프트웨어 RAID 파일 시스템 모델을 제시하고, 기존의 클라이언트/서버 파일 시스템과 제안하는 소프트웨어 RAID 파일 시스템의 성능을 트레이스 기반 시뮬레이션을 통해 다양하게 비교한다. 실험을 통해 기존의 클라이언트/서버 파일 시스템보다 본 논문에서 제안한 소프트웨어 RAID 파일 시스템 모델이 효율적인 성능을 나타내며 확장성이 뛰어난 파일 시스템임을 확인한다.

ABSTRACT

Due to the price and performance of uniprocessor workstations and off-the shelf networking, network of workstations(NOW) are now a cost-effective parallel processing platform that is competitive with supercomputers. Meanwhile, current network file system protocols rely heavily on a central server to coordinate file activity among client workstations. This central server can become a bottleneck that limits scalibility for environments with large numbers of clients. In this paper, we propose a highly reliable and effective software RAID file system on the

*홍익대학교 컴퓨터공학과
論文番號:97166-0516
接受日字:1997年 5月 16日

network of workstation environment. We present results from a trace-driven simulation study that shows that the designed software RAID file system is more effective in the aspect of elapsed time when compared with client/server file systems.

Ⅰ. 서 론

현재까지 병렬 프로그램은 주로 MPP(Massively Parallel Processors)와 같은 슈퍼 컴퓨터에서 수행되고 있다. 그러나 최근에 MPP는 비용/성능비가 높다는 단점과 시스템 개발시 추가적으로 새로운 운영체제 및 응용 소프트웨어를 개발해야 하는 단점이 있다는 사실을 인식하게 되었다[1]. 동시에 워크스테이션이나 개인용 컴퓨터는 하드웨어의 기술 발달과 더불어 경제성이 매우 높아졌고 ATM[2]이나 Myrinet[3]과 같은 초고속의 범용 네트워크들이 개발됨으로 네트워크 하드웨어 지연 시간을 줄이게 되었다. 아울러 AM(Active Message)[4]과 같은 통신 프리미티브를 통해 소프트웨어 오버헤드인 시스템 호출에 걸리는 시간과 운영체제와 통신 프로토콜에서 걸리는 시간을 줄일 수 있게 되었다. 이러한 환경의 변화는 병렬 프로그램(parallel program)을 분산 시스템 환경에서 수행시키고자 하는 시도에 박차를 가하고 있다. 이러한 분산 시스템 환경에서 병렬 프로그램을 수행시키고자 하는 것을 NOW(Network of Workstations)라 일컫는다[1].

이러한 상황에서 디스크의 입출력은 전체 시스템의 병목으로 지목되고 있다[5]. 결국 암달의 법칙에 의해 효율적인 디스크의 입출력을 위한 파일 시스템 없이는 전체 시스템의 성능 향상을 기대할 수 없다. 특히, 상당한 입출력 성능을 요구하는 병렬 프로그램을 수행시켜야 하는 NOW 환경에서는 효율적인 디스크의 입출력을 위한 파일 시스템의 지원이 반드시 있어야 한다. 분산 시스템 환경에서 이와 같은 입출력 병목현상을 해결함과 동시에 높은 신뢰성을 지원하기 위해 도래된 개념이 소프트웨어 RAID 파일 시스템이다[6]. 소프트웨어 RAID 파일 시스템이란 네트워크로 연결된 다수의 시스템들이 보유한 각 디스크를 통해 하드웨어 RAID[7]의 기능을 소프트웨어적으로 제공하는 시스템을 의미한다.

본 논문에서는 우선 NOW 시스템과 소프트웨어 RAID 파일 시스템에 관해 상세하게 살펴보고 새로운 소프트웨어 RAID 파일 시스템 모델을 제안한다. 그리고 기존의 클라이언트/서버 파일 시스템과 제안한 소프트웨어 RAID 파일 시스템의 성능을 다양하게 비교한다. 이를 위하여 클라이언트/서버 파일 시스템과 소프트웨어 RAID 파일 시스템을 세부적으로 모델링한 시뮬레이터를 개발하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 NOW 환경에 관해서 살펴보고, Ⅲ장에서는 소프트웨어 RAID 파일 시스템에 관하여 상세히 살펴보고, Ⅳ장에서 기존의 클라이언트/서버 파일 시스템과 본 논문에서 제안하는 소프트웨어 RAID 파일 시스템에 대한 모델을 제시한다. 그리고 Ⅴ장에서 기존의 클라이언트/서버 파일 시스템과 소프트웨어 RAID 파일 시스템의 성능을 실험을 통해 비교하고, 마지막으로 Ⅵ장에서 결론을 맺는다.

Ⅱ. NOW 시스템

워크스테이션 클러스터(workstation cluster)는 자치적으로 운용되고 있는(autonomous) 워크스테이션의 집합으로, 이들 각 워크스테이션들은 네트워크로 연결되어 있다. 프로세서와 ATM과 같은 네트워크 기술이 발전함에 따라 워크스테이션 클러스터의 성능 또한 발전하여 순차적인 프로그램뿐 아니라 병렬 프로그램도 MPP(Massively Parallel Processors)에서와 같은 성능으로 수행할 수 있게 되었는데, 이와 같은 목적을 가지고 연구되는 워크스테이션 클러스터를 NOW(Network of Workstations)라고 한다[1].

University of California, Berkeley에서는 현재 NOW 프로젝트를 수행하고 있으며, 버클리 NOW의 기본 구조는 그림 1과 같다. 그림에서 GLUnix는 일반적인 운영체제인 Unix상에서 워크스테이션 클러스터에게 전역적 시스템 관점(global system view)을 제공하는 범 운영체제이다. 즉, 사용자가 순차 프로그램의 수행을 요청한 경우에는 사용자가 사용 중인 워크스테이션에서 순차 프로그램을 수행시키고 병렬 프로그램

의 수행을 요청한 경우에는 현재 사용되지 않는 워크스테이션들을 찾아 그 워크스테이션들에서 병렬 프로그램을 실행시켜주는 일을 담당한다. 그리고 Communication Software는 운영체제의 간섭을 극소화하여 사용자 프로세스에서 직접 네트워크 인터페이스로 메세지를 주입하고 발췌할 수 있는 고성능 통신 메카니즘을 의미한다.

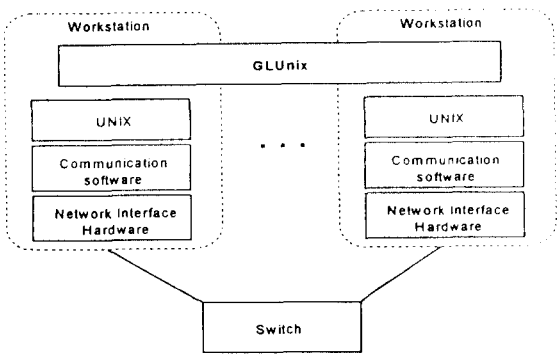


그림 1. NOW의 기본 구조

이와 같이 운영체제에 있어서 전역적 시스템 관점을 제공하고 고성능 통신 메카니즘의 지원이 있어야 어느정도 NOW 시스템이 실현될 수 있다. 그러나 극대화된 성능을 갖는 NOW의 실현을 위해서는 기존의 분산 파일 시스템이 아닌 높은 성능과 신뢰도를 갖는 병렬/분산 파일 시스템의 지원이 있어야 한다. 이를 위하여 본 논문에서는 소프트웨어 RAID 파일 시스템의 지원을 목표로 하고 있다.

Ⅲ. 소프트웨어 RAID 파일 시스템

분산 시스템 환경에서 입출력 병목현상을 해결함과 동시에 높은 신뢰성을 지원하기 위해 도래된 개념이 소프트웨어 RAID 파일 시스템이다. 소프트웨어 RAID 파일 시스템이란 네트워크로 연결된 다수의 시스템들이 보유한 각 디스크를 통해 하드웨어 RAID의 기능을 소프트웨어적으로 제공하는 시스템을 의미한다. RAID[7]란 데이터를 다수의 디스크들에 나누어 저장함으로 입출력 성능을 높일 수 있고, 패리티 정보를 저장함에 의해 신뢰성을 높일 수 있는 디스크 기술이므로 NOW 환경에 적합할 것으로 판단된다. RAID에 대한 내용은 본 논문의 범주에서 벗어나므로 더 이상 언급하지 않기로 하겠다. 이에 대한 상세한 내용은 [7]을 참조하기 바란다.

소프트웨어 RAID 파일 시스템은 그림 2와 같은 형태로 구성되어 있다. 그림 2에서 시스템은 초고속 통신망으로 연결되어 있는 분산 시스템 환경이며 전체 시스템들이 보유한 디스크를 통해 하드웨어 RAID의 기능을 소프트웨어적으로 제공하고 있다. 그럼으로 인해 얻어지는 장점을 살펴보면 다음과 같다. 첫째 파일 저장시 특정 워크스테이션의 디스크에만 저장되는 것이 아니라 저장될 파일을 여러 워크스테이션에 분산 저장함으로 입출력 동작에 있어서 성능 향상

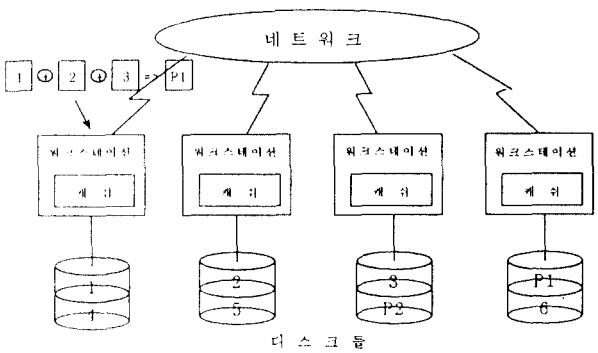


그림 2. 소프트웨어 RAID 파일 시스템

을 가져오게 된다. 그림 2를 통해 설명하면 블록 1, 2, 3으로 구성된 파일을 디스크에 저장할 때 특정 워크스테이션에만 집중적으로 저장하는 것이 아니라 세 대의 워크스테이션들의 디스크에 분산시켜 동시에 저장하게 되는 것이다. 두 번째 장점은 패리티 정보를 추가적으로 저장함으로 특정 시스템에 결함이 발생하여도 신뢰성을 유지할 수 있다는 것이다. 그림 2에서 블록 1, 2, 3을 저장할 때 이들 블록에 대한 패리티 블록 P1을 계산하여 다른 워크스테이션의 디스크에 저장함으로 특정 워크스테이션의 결함이 발생하여도 나머지 워크스테이션들에 저장된 내용을 가지고 결함이 발생한 워크스테이션의 디스크에 저장된 정보를 복구할 수 있게 된다.

Ⅳ. 시스템 모델

본 장에서는 시뮬레이션 시스템에 대한 모델을 제시하고자 하는데 클라이언트/서버 파일 시스템과 본 논문에서 제안하는 소프트웨어 RAID 파일 시스템 각각에 대해 살펴본다.

4.1 클라이언트/서버 파일 시스템

클라이언트/서버 파일 시스템[8]의 동작은 잘 알려져 있으므로 상세한 언급은 생략하고 실험에 사용된 시뮬레이터에 해당된 부분만 간단하게 살펴본다.

클라이언트와 서버의 캐쉬 교체 정책은 LRU 기법을 사용한다고 가정하였다. 동일한 블록이 여러 클라이언트의 캐쉬에 저장됨으로 인해 발생하는 캐쉬 일관성에 대한 문제는 쓰기-무효화 정책을 이용하여 해결하도록 하였으며 모든 클라이언트는 write-through 정책을 사용하여 서버는 항상 최신의 블록을 유지하게 된다. 그러나 서버의 캐쉬는 안정성이 높다는 전제하에 write-back 정책을 사용한다.

4.2 소프트웨어 RAID 파일 시스템

본 논문에서 제안하는 소프트웨어 RAID 파일 시스템 모델은 RAID 레벨 5 형태로 구성되었으며 시스템 전체의 캐쉬 내에 저장되어 있는 모든 블록은 한 개만을 유지한다. 또한 캐쉬들간의 동작은 [9]에서처럼 리모트 워크스테이션의 캐쉬에 직접적인 접근(access)이 가능하다고 가정하였다.

블록 요구시 다음의 단계로 동작한다. 우선 로칼 캐쉬에서 히트가 발생하면 LRU 리스트에서 MRU 블록으로 위치시키고 미스가 발생하면 리모트 캐쉬에 해당 블록이 있는지를 모든 워크스테이션들에게 요구한다. 만약 다른 워크스테이션의 캐쉬에서 히트가 발생하면 해당 블록을 리모트 캐쉬의 LRU 리스트에서 제거하고 이동시켜 로칼 캐쉬의 MRU 블록으로 위치시킨다. 모든 워크스테이션들의 캐쉬에서 미스가 발생하면 해당 블록을 디스크로부터 읽어서 로칼 캐쉬의 MRU 블록으로 위치시킨다. 그러나 캐쉬에서 교체될 블록이 dirty인 경우에는 그 동작이 일반적인 클라이언트/서버 파일 시스템과는 큰 차이가 있다. 그 동작은 다음과 같다.

1. 교체 블록의 옛 데이터와 교체 블록과 같은 패리티 그룹[7]에 속하는 옛 패리티 읽기
2. 새로운 데이터(교체 블록)와 옛 데이터, 그리고 옛 패리티를 가지고 새로운 패리티 계산
3. 새로운 데이터와 새로운 패리티 쓰기

이러한 일이 발생하는 이유는 결함 허용을 위해 새로운 데이터를 디스크에 저장하기 전에 새로운 패리티 정보를 계산해야 하기 때문인데 이러한 동작은 시스템의 성능을 제한할 것으로 예상된다.

Ⅴ. 실 험

5.1 실험 방법

본 실험에서는 트레이스 기반 시뮬레이션을 통하여 분산 파일 시스템들을 평가하였다. 실험에 사용된 트레이스는 스프라이트 트레이스[10]이다. 스프라이트 트레이스는 버클리 대학에서 4대의 파일 서버와 40여대의 클라이언트에서 매일 사용하는 30여명의 사용자와 간헐적으로 사용하는 40여명의 사용자들에게서 얻어진 분산 파일 시스템의 트레이스이다.

본 시뮬레이션에서는 동일한 파일 서버를 이용하는 열 여섯 대 클라이언트의 트레이스를 임의로 선택하여 사용하였다. 다양한 실험을 하였는데 여덟 개와 열 여섯 개의 트레이스를 사용하였다. 실험에 사용된 트레이스의 블록 요구 수는 각각 242069개(175232개의 읽기 요구와 66837개의 쓰기 요구)와 455584개(334503개의 읽기 요구와 121081개의 쓰기 요구)이다. 이러한 트레이스들 간에는 동일한 블록에 대한 요구가 포함되어 있어 파일을 공유하는 경우가 존재하며 이들 트레이스들은 여덟 대와 열 여섯 대의 워크스테이션으로 구성된 분산 환경에서 각 워크스테이션들의 작업 부하가 되어 시뮬레이션을 수행하게 된다. 실험에서 각 워크스테이션 캐쉬를 초기화시키는데 소요된 요구를 제외한 나머지 요구를 가지고 평가하였다.

본 논문에서 시뮬레이터는 C++를 사용하여 개발하였으며 두 시뮬레이터에서 공통적인 사항은 다음과 같다. 실험에서 캐쉬 블록의 크기는 8KB이며, 8KB 데이터에 대한 접근 시간은 로칼 메모리는 250μs[11], 네트워크 오버헤드는 200μs로 하고 데이터가 네트워크를 통해 전송되는데 걸리는 시간은 400μs[12]로 고

정시키고 실험하였으며 특히 디스크 부분은 HP 97560 하드 디스크를 모델링한 시뮬레이터[13]를 사용하였다. 각 파일 시스템에 대한 성능 비교 척도는 요구당 평균 경과 시간(elapsed time)으로 히트율에서는 측정할 수 없는 소프트웨어 RAID 파일 시스템에서 dirty 블록 교체시에 발생하는 오버헤드를 측정하기 위해서 이러한 척도를 선택하였다. 실험은 여덟 대와 열 여섯 대의 워크스테이션으로 구성된 분산 환경에서 각 워크스테이션 캐쉬의 크기를 변화시키면서 파일 시스템들의 성능을 비교하였다.

5.2 실험 결과

그림 5와 그림 6은 분산 파일 시스템들에 대하여 캐쉬의 크기를 달리하면서 요구당 평균 경과 시간을 나타낸 것이다. 그림에서 C/S(2)란 서버의 캐쉬 크기가 클라이언트 캐쉬의 크기에 두 배인 클라이언트/서버 파일 시스템을 의미하는 것이고, C/S(4)란 서버의 캐쉬 크기가 클라이언트 캐쉬의 크기에 네 배인 클라이언트/서버 파일 시스템을 의미하는 것으로 그림에서 x-축의 값이 1000이라는 것은 클라이언트 캐쉬의 크기는 1000이고 서버 캐쉬의 크기는 4000을 의미하는 것이다. 그리고 S/W RAID는 제안한 소프트웨어 RAID 파일 시스템을 나타내는 것이다.

두 그림 모두 x-축은 각 워크스테이션의 캐쉬의 크기를 나타내는 것으로 8KB 블록의 개수를 의미하고 y-축은 요구당 평균 경과 시간을 나타내는 것으로 단위는 μs이다.

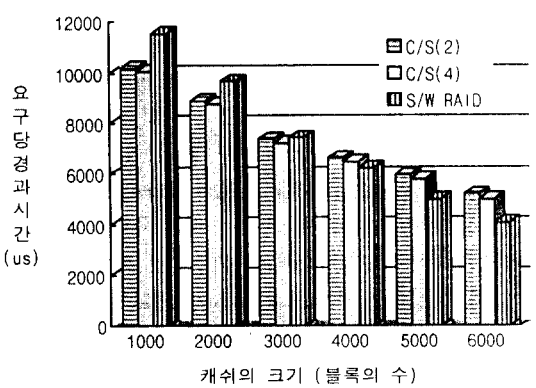


그림 3. 워크스테이션의 수가 여덟인 경우 성능 비교

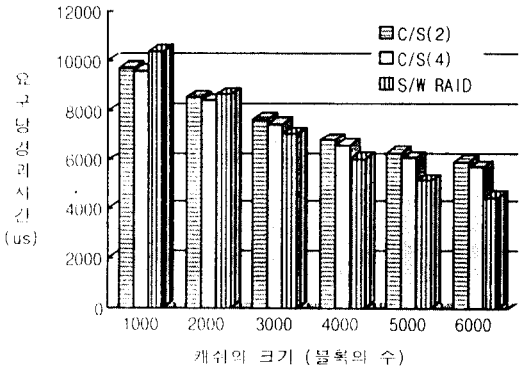


그림 4. 워크스테이션의 수가 열 여섯인 경우 성능 비교

실험을 통해 몇 가지 사항들을 관찰할 수 있다. 우선 캐쉬의 크기가 작을 경우에는 소프트웨어 RAID 파일 시스템이 클라이언트/서버 파일 시스템에 비해 약간 낮은 성능을 나타내고 있는데 이는 소프트웨어 RAID 파일 시스템에서는 네 번의 디스크 요구를 수반하는 dirty 블록 교체의 발생 빈도수가 너무 크기 때문이다. 그러나 캐쉬의 크기가 커짐에 따라 제안한 소프트웨어 RAID 파일 시스템이 훨씬 좋은 성능을 나타내는 것을 볼 수 있다. 이는 클라이언트/서버 파일 시스템에서 모든 클라이언트는 자신의 자원(캐쉬)과 서버의 자원만 활용할 수 있는 반면 소프트웨어 RAID 파일 시스템에서는 다른 모든 워크스테이션들의 자원을 활용할 수 있기 때문이다. 특히 결과에서 확인할 수 있듯이 제안한 소프트웨어 RAID 파일 시스템은 워크스테이션의 수가 증가할수록 더욱 좋은 성능을 나타내는 확장성이 뛰어난 파일 시스템임을 알 수 있다.

Ⅵ. 결 론

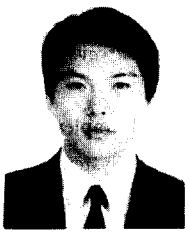
최근 프로세서 속도의 급증과 초고속 네트워크의 출현 등으로 병렬 프로그램을 분산 시스템 환경에서 수행시키고자하는 NOW(Network of Workstations)가 대두되고 있다. 그러나 상당한 입출력 성능을 요구하는 병렬 프로그램의 실행을 지원하려면 효율적인 파일 시스템의 지원이 반드시 있어야 한다. 본 논문은 NOW 환경의 파일 시스템에 성능을 높여주기 위한 방안들을 제안하고 그들을 실험을 통해 성능을 평가하였다.

본 논문에서는 NOW 환경의 파일 시스템에 성능을 최대화하기 위한 방안으로 소프트웨어 RAID 형태의 파일 시스템을 도입하고 새로운 모델을 제안하였다. 소프트웨어 RAID 파일 시스템은 파일 저장시 여러 워크스테이션의 디스크에 분산시켜 동시에 저장함으로 성능을 향상시키고, 또한 패리티 정보를 추가적으로 저장함으로 기존의 분산 파일 시스템에서는 제공하지 못하던 시스템의 결함을 허용할 수 있는 시스템이다.

본 논문에서는 우선 NOW 시스템과 소프트웨어 RAID 파일 시스템에 대해 살펴보았으며, 기존의 클라이언트/서버 파일 시스템과 본 논문에서 제안한 소프트웨어 RAID 파일 시스템의 성능을 트레이스 기반 시뮬레이션을 통해 비교하였다. 실험을 통해 본 논문에서 제안한 소프트웨어 RAID 파일 시스템이 좋은 성능을 나타내는 효율적인 파일 시스템이며 특히 확장성이 뛰어난 파일 시스템임을 확인할 수 있었다.

참 고 문 헌

1. T. E. Anderson, D. E. Culler, and D. A. Patterson, "A Case for NOW(Networks of Workstations)," *IEEE Micro*, 15(1):54-64, February 1995.
2. D. E. McDysan and D. L. Spohn, *ATM: Theory and Application*, McGraw-Hill, 1995.
3. N. J. Boden et al., "Myrinet: A Gigabit-per-Second Local Area Network," *IEEE Micro*, 15(1):29-36, February 1995.
4. T. von Eicken and David E. Culler, "Active Message: A Mechanism for Integrated Communication and Computation," In *Proceedings of 19th Annual International Symposium on Computer Architecture*, pages 256-266, May 1992.
5. J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufman, 1990.
6. T. E. Anderson, M. D. Dahlin, J. M. Neefe, D. A. Patterson, D. S. Roselli, and R. Y. Wang, "Serverless Network File System," *ACM Transactions on Computer Systems, 14(1):41-79*, February 1996.
7. P. Chen, E. Lee, G. Gibson, R. Katz, and D. A. Patterson, "RAID: High-Performance, Reliable Secondary Storage," *ACM Computing Surveys*, 26(2): 145-185, June 1994.
8. Russel Sandberg et al, "Design and Implementation of the SUN Network File System," In *USENIX Conference & Exhibition*, Portland, Oregon, 1985.
9. M. D. Dahlin, R. Y. Wang, T. E. Anderson, and D. A. Patterson, "Cooperative Caching: Using remote client memory to improve file system performance," In *Proceedings of the First Symposium on Operating System Design and Implementation*, pages 267-280, November 1994.
10. M. G. Baker, J. H. Hartman, M. D. Kupfer, K. W. Shirriff, and J. K. Ousterhout, "Measurements of a Distributed File System," In *Proceedings of the ACM Thirteenth Symposium on Operating Systems Principles*, pages 198-212, October 1991.
11. R. Martin, "HPAM: An Active Message Layer for a Network of HP Workstations," In *Proceedings of the 1994 Hot Interconnects II Conference*, 1994.
12. K. K. Keeton, T. E. Anderson, and D. A. Patterson, "LogP Quantified: The Case for Low-Overhead Local Area Networks," In *Proceedings of the 1995 Hot Interconnects III Conference*, 1995.
13. David Kotz, Song Bac Toh, and Sriram Radhakrisshnan, "A Detailed Simulation Model of the HP 97560 Disk Drive," *Technical Report PCS-TR94-220*, Dartmouth College, Computer Science, Hanover, NH, 1994.



김 종 훈(Jong-Hoon Kim) 정회원
1990년 : 목원대학교 수학교육과 이학사
1992년 : 동국대학교 대학원 통계학과 전산통계전공 이학석사
1993년~현재 : 홍익대학교 대학원 전자계산학과 박사과정
※주관심분야 : 네트워크 운영체제, 분산 멀티미디어 시스템

노 삼 혁(Sam H. Noh) 정회원
1986년 : 서울대학교 공과대 공학사
1993년 : Univ. of Maryland of College Park 공학박사
1993년 9월~1994년 6월 : George Washinton Univ. 객원 조교수
1994년 8월~현재 : 홍익대학교 컴퓨터공학과 조교수
※주관심분야 : 분산 및 병렬 파일 시스템, 실시간 운영체제

원 유 헌(Yoo-Hun Won) 정회원
1972년 : 성균관대학교 수학과 이학사
1975년 : 한국과학원 전자계산학과 이학석사
1985년 : 고려대학교 이학박사
1975년~1976년 : 한국과학기술연구소 연구원
1986년~1987년 : R.P.I 교환교수
1976년~현재 : 홍익대학교 컴퓨터공학과 교수
※주관심분야 : 프로그래밍 언어론, 병렬 프로그래밍, 실시간 프로그래밍