□ 기술해설 □

고성능 병렬 컴퓨팅 지원 클러스터링 소프트웨어

시스템공학연구소 반난주*·김태근*

1. 서 론

1980년대 컴퓨터간 접속을 가능하게 하는 네트워크 장비가 등장한 이래로 원격지의 컴퓨팅 자원을 사용하는 방법이 연구되어 왔고[1], 1990년대에 들어서는 인터넷을 통하여 세계에 널리 퍼져있는 정보들을 손쉽게 찾아볼 수 있는 웹 브라우저의 급속한 보급과 더불어 네트워크를 이용한 컴퓨팅 기술 발전이 더욱 가속화 되었다.

그동안 네트워크 컴퓨팅 환경을 구성하는 데 있어서 가장 장애 요소로 작용되었던 시스템간 이질성의 문제도 최근에 등장한 Java 언어를 통하여 어렵지 않게 극복됨에 따라 네트워크 컴퓨팅의 확산 가능성 더욱 증가되고 있다. 최근들어 상용되고 있는 네트워크 컴퓨팅이란 네트워크상에 연결되어 있는 컴퓨팅 자원들을 자신이 보유한 시스템의 일부인 것처럼 사용할 수 있는 환경을 제공하는 기술을 말한다.

네트워크 컴퓨팅은 컴퓨팅 자원을 연동시키는 공간의 범위에 따라 구분할 수 있는데, 비교적 제한된 공간에서 워크스테이션들을 연동시키는 것을 클러스터 시스템이라 한다. ATM, Myrinet과 같은 스위치에 기초한 초고속 통신 기술이 개발됨에 따라 고성능 프로세서와 대량의 메모리, 방대한 양의 디스크를 가진 고성능의 워크스테이션을 연동하는 기술에 대한 연구가 널리 진행되고 있다. 이렇게 고성능 워크스테이션을 초고속 통신 네트워크로 연동하는 기술은 저렴한 가격으로 초병렬 컴퓨터와 같은 슈퍼 컴퓨터의 성능을 구비한 워크스테이션 클러스터 시스템을 개발할 수 있는 가능성을 보이고 있으며, 일부 프로토타입 시스템이 개발 단계에 있다.

이와 같이 다수의 워크스테이션을 연동시키는 방법은 다수의 접근 방법으로 여러기관에서 연구되고 있다. 버클리 대학의 NOW, Vrije 대학의 Amoeba, MIT의 Cilk, 카네기 멜론 대학의 Dome 등은 접근 방식은 달라도 모두가 워크스테이션을 연동하여 고성능 컴퓨팅 환경을 만들고자 하는 연구들이다.

2장에서는 클러스터 시스템의 개발 배경, 3장에서는 클러스터 시스템 구축에 필요한 요소 기술 개요과 동향을 알아보고 4장에서 향후 전망을 끝으로 본 고를 마무리 한다.

2. 클러스터 시스템의 개발 배경

조사된 통계 자료에 의하면 워크스테이션의 평균 가동율은 매우 저조하여 평균 20~30%의 수준이며, 많아야 60%에 이르는 것으로 알려져 있다. 워크스테이션 클러스터링 기술은 이러한 유휴 자원을 활용시키는 기술로서, 각 사용자에게는 자신의 워크스테이션이 전용시스템으로서 기능을 보장하면서, 유휴할 때는 다른 작업을 위하여 사용될 수 있도록 한다. 이러한 워크스테이션 클러스터 시스템을 개발할 수 있는 기술적 배경은 다음과 같다.

- 프로세서의 고성능화 : 현재의 프로세서는 놀라운 정도로 빨라지고 있다. 예를 들어 DEC사의 Alpha 프로세서는 500MHz의 처리 속도를 갖고 있어 Cray C90의 프로세서와 견줄 수 있을 정도이다. 워크스테

* 정회원

이션들은 이렇게 빠른 프로세서와 함께 많은 양의 메모리와 디스크를 갖고 있다.

- 초고속 통신 기술 : 과거의 Ethernet은 10 Mbps의 전송 속도를 갖는 반면 ATM의 경우는 155Mbps, Myrinet의 경우는 640 Mbps의 전송 속도를 가지고 있다. 이러한 초고속 통신 기술은 고성능의 워크스테이션들을 초고속망으로 연동할 수 있게 한다.

클러스터 시스템을 구축하는 기술 개발의 타당성은 버클리 대학의 NOW 프로젝트 팀에서 추정한 클러스터 시스템과 초병렬 컴퓨터와의 가격대 성능비를 보더라도 잘 알 수 있다. 그림 1에 예시된 바를 살펴보면 128개의 워크스테이션을 연동하는 클러스터 시스템을 구성할 때, 128개의 프로세서를 갖는 CM-5 병렬 컴퓨터에 비하여 1/3 가격으로 구성할 수 있다[2]. 표 1에서는 슈퍼컴퓨터, 초병렬 컴퓨터와 워크스테이션 클러스터 시스템과의 성능을 기상 시뮬레이션 프로그램인 GATOR를 이용하여 연산 수행 시간 및 시스템 비용을 비교하였다. 워크스테이션 클러스터 시스템에서 병렬 파일 시스템 기술과 낮은 오버헤드 통신 소프트웨어를 제공할 경우에 벡터 슈퍼컴퓨터와 초병렬 컴퓨터에 비하여 비용면과 성능면에서 우위를 나타낸다.

3. 클러스터 시스템

클러스터 시스템을 구축하기 위한 기반 기술로는 클러스터 소프트웨어 기술과 고속 통신 기술, 병렬 파일 시스템 기술 등이 있다. 본 절에서는 클러스터 시스템 구현하기 위한 요소

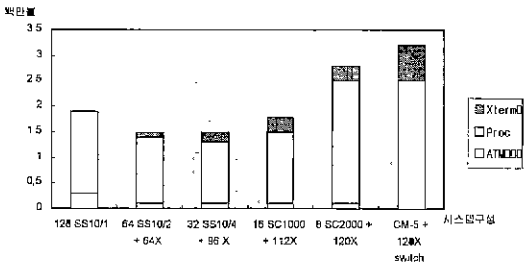

그림 1 128개 프로세서 시스템 구성 가격

기술들과 각 요소 기술들의 기술 동향에 대하여 살펴보기로 하겠다.

3.1 클러스터 소프트웨어

워크스테이션들을 연동하여 클러스터 시스템으로 사용한다는 것은 단순히 물리적인 장치의 연결만을 의미하는 것은 아니며, 사용자가 단일 컴퓨팅 시스템에서 작업을 하는 것처럼 자원의 위치나 성능 등에 대한 세부적인 사항을 알지 않아도 작업을 실행시켜 그 결과를 받아볼 수 있음을 의미한다. 클러스터 소프트웨어란 사용자에게 투명한 컴퓨팅 환경을 제공해주어야 하며, 작업의 반응 시간을 최소화할 수 있도록 해야 한다.

분산 운영체제에 기초를 둔 클러스터 소프트웨어 기술은 구현 계층 및 서비스 제공 수준에 따라 클러스터 관리 소프트웨어와 클러스터 컴퓨팅 환경으로 분류할 수 있다[3].

- 클러스터 관리 소프트웨어 : 워크스테이션에 처리 요구된 응용 프로그램들을 관리하는 것을 주 목적으로 하며 기존 운영체제 위에 구현되어 하나의 응용 프로그램처럼 실행한다.
- 클러스터 컴퓨팅 환경 : 클러스터 시스템이

표 1 슈퍼컴퓨터, 초병렬 컴퓨터 및 워크스테이션 클러스터 시스템의 성능 대 가격비

시 스 템	연산시간 (초)	데이타 교환시간(초)	입력 시간 (초)	전체 소요시간 (초)	비 용 (백만불)
CRAY-C90(16)	7	4	16	27	30
Paragon(256)	12	24	10	46	10
워크스테이션 256대	4	23,340	4,030	27,374	4
WC+ATM	4	192	2,015	2,211	5
WC+ATM+PFS	4	192	10	205	5
WC+ATM+PFS+LM	4	8	10	21	5

(WC : 워크스테이션 클러스터, PFS : 병렬 파일 시스템, LM(Low-overhead Message) 비헤드가 적은 메시지)

표 2 클러스터 소프트웨어

	클러스터 관리 소프트웨어				클러스터 컴퓨팅 환경			
	Codine	Condor	DOS	LSF	Amoeba	NOW	Cilk	DOME
이기종 환경 지원	○	○	○	○	○	×	×	○
프로세스 전이	○	○	×	○	×	○	×	○
워크스테이션의 자율성	×	○	×	×	×	○	×	×
병렬 프로그램 실행	○	×	○	○	○	○	○	○
고장 감내	○	○	○	○	×	×	○	○
Checkpointing	○	○	×	○	○	○	○	○

응용 프로그래밍 환경으로서 제공되며, 분산 공유 메모리 시스템에서 프로그래밍을 하는 것과 유사한 형태를 이룬다. 이들의 경우 기존 커널 대신 마이크로 커널을 기반으로 하여 구현된 것이 일반적이다.

클러스터 소프트웨어 구현에 있어서 가장 중요한 것은 어떠한 스케줄링 정책을 사용할 것인가 하는 문제이다. 그외 이기종 환경 지원, 병렬 프로그래밍 지원, 프로세스 전이, 동적 프로세스 전이, Checkpointing 등은 모두 클러스터 소프트웨어 구현에 있어서 고려해야 할 사항들이며 이미 개발되어 있는 클러스터 소프트웨어의 특징들을 표 2에 정리하였다.

- 독일의 GENIAS사에서 개발한 Codine (Compu-ting in Distributed Network Environment)은 워크스테이션뿐 아니라 벡터 컴퓨터 및 초병렬 컴퓨터까지도 함께 연동시킬 수 있으므로, 공학 및 과학 분야의 대규모 문제 해결에 매우 유용하게 사용할 수 있다.
- LSF(Load Sharing Facility)는 캐나다 Platform Computing사의 상용 클러스터 소프트웨어로 배치작업을 위한 효율적인 자원관리를 최우선의 목표로 하고 있다[4].
- 위스콘신 대학에서 개발한 Condor는 워크스테이션이 완전히 idle한 상태에 있는 경우에 한해서만 클러스터의 일부로서 참가하게 하여 워크스테이션의 자율성을 최대로 보장하고 있다.
- Ameoba는 마이크로 커널을 기반으로 한 분산 컴퓨팅 시스템으로서 단일 시스템 이미지를 제공하고 있다.
- Cilk는 일종의 병렬 프로그래밍 언어의 형태를 띠며, 실행 시간에 다중 쓰레드를 생성하여 동적으로 작업을 스케줄링 한다[5].
- 카네기 멜론 대학의 DOME(Distributed Object Migration Env.)는 PVM을 기반으로 하며, C++에 객체 이동에 대한 클래스들을 선언하여 프로그램이 실행될 때 시스템의 부하를 고려하여 작업이 이주될 수 있도록 하고 있다[6].
- NOW는 기존 운영체제인 UNIX를 수정하지 않는 방식을 않고 클러스터 컴퓨팅 환경이 가능하게 하고 있다. 또한 Split-C라는 프로그래밍 언어를 제공하여 병렬 프로그래밍을 지원하고 있다[2].

3.2 고속 통신 메카니즘

·네트워크 장비의 고속화에도 불구하고 실제 사용자 프로세스간의 통신 속도는 그 기대에 미치지 못하고 있는 것이 현실이다. 이러한 현실은 네트워크의 고속화 관점을 네트워크 장비(전송방식, 스위칭 기술, 고속 케이블 등)로부터 호스트 내부의 경로, 즉, 사용자 프로세스에서 네트워크 장비까지 옮겨 놓았다. 호스트 내부에서 통신을 위해서 경유하는 경로는, 일반적으로 잘 알려진 기존의 네트워크 프로토콜 중 트랜스포트 계층과 네트워크 계층(OSI)이다. 대부분의 고속 통신 프리미티브 개발자들은 기존의 네트워크 프로토콜을 이용하는 통신 방법에서 발생하는 오버헤드를 다음 세가지로 규정하고 있다.

- 프로토콜이 커널 내부에 존재하므로 커널과 사용자간의 모드 전환에 따른 오버헤드 발생
- 메모리 복사에 의한 오버헤드

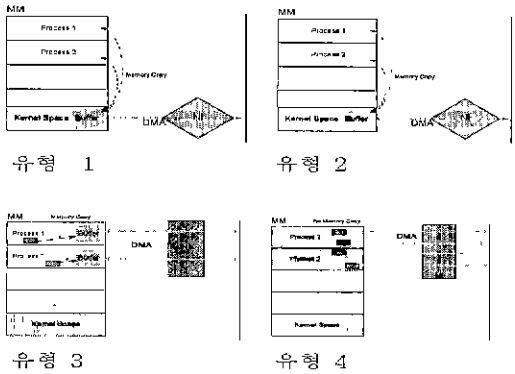

그림 2 다양한 통신 프로토콜

●프로토콜의 복잡도에 의한 오버헤드

Myrinet API, AM, FM, PM, U-Net 등은 이와 같은 오버헤드를 제거한 고속 통신 프리미티브 모델을 개발하는 연구들로 기존의 프로토콜을 이용한 통신 절차와 위에 열거된 각각이 제안하는 통신 모델을 비교하면 그림 2와 같다.

그림 2의 유형 1은 여러번의 메모리 복사와 커널과 사용자간 모드 전환이 일어나는 기존의 프로토콜을 이용한 통신 절차를 나타내고 있다. 여기에서 모드 전환 오버헤드를 없앤 것이 유형 2에 나와있는 방법으로, 커널 내부에 존재하는 네트워크 인터페이스(NI)의 디바이스 드라이버는 NI의 초기화 과정에서 NI만이 소유하는 메모리의 고유 영역을 할당 받고, 이 영역을 DMA(Direct Memory Access) 가능 지역으로 커널과 NI에 알린다. Myrinet API와 버클리 대학의 AM(Active Message)[7], 일본 RWCP(Real World Computing Progam)의 PM, 일리노이 대학의 FM(Fast Message) 등이 이에 속한다. 그러나 이 경우에서도 메모리 복사 오버헤드를 줄이지는 못하고 있다. 또한, 동시에 여러 프로세스가 통신할 수 있는 멀티 통신 채널을 위해서 동적 버퍼 관리 기능 및 프로텍션 기능 등이 필요하다.

유형 3은 모드 전환 오버헤드를 개선하고, 프로세스 통신 버퍼를 통신을 수행하는 각 사용자 프로세스 내부에 할당한다. 이 방법을 위해서는 호스트 Boot-up 시 NI의 메모리를 사용자 영역에 매핑시키는 기능과, 각 프로세스의 통신을 위해서 버퍼를 할당하는 연결 설정 작업이 필요하다. 그러나 역시 메모리 복사 오버헤드를 줄이지는 못하였고, 또한 버퍼 관리를 각각의 사용자 프로세스에서 해주어야 한며, 통신을 위한 버퍼 할당에 한계가 있다는 점이다. 코넬대학에서 개발한 U-Net의 통신 메카니즘이 이에 속한다[8].

유형 4는 U-Net의 단점을 보완하여 만든 통신 메카니즘으로 통신을 원하는 프로세스의 메모리 영역 안에 있는 전송될 자료 블럭들을 메모리 복사 없이 직접 DMA 하고 있다[9]. 이 방법은 유형 1에서 발생했던 모든 문제를 해결할 수 있는 방법으로 주목된다. 이러한 구조를 위해서는 여러가지 요구사항이 발생될 수 있다. 우선, NI에서 호스트 메모리의 임의의 영역을 직접 접근할 수 있는 방법이 필요하다. 현재는 TLB(Translation Look-aside Buffer)를 NI에 복사하는 방법을 이용해서 이 문제를 해결하고 있다. 그리고, 호스트의 TLB와 NI TLB의 일치성을 유지하는 방법이 필요하다. 또한, 페이지 Fault 문제의 해결 방안 역시 필요하다. 페이지 Fault가 많이 발생하면 전체적인 성능이 크게 저하될 우려가 있다.

3.3 병렬 파일 시스템

클러스터 관리 소프트웨어와 초경량 통신 소프트웨어로 클러스터 시스템을 구성할 경우 빠른 연산과 프로세스간의 고속 통신을 제공하나 데이타 입출력의 성능저하는 전체 시스템의 성능에 지대한 영향을 미친다. Grand Challenge Problem과 같은 대규모 시뮬레이션은 고속의 데이타 입출력이 요구되고 그 활용이 더욱 증대되고 있으나 과거 20년동안 디스크의 입출력 속도 성능 향상은 네트워크, 메모리, 프로세스의 급속한 발전에 비해 상대적으로 느리게 발전됨으로써 병렬 컴퓨팅 환경에서 입출력이 성능 향상의 걸림돌(bottleneck)이 되고 있다.

특히, 멀티미디어 어플리케이션이나 대용량 데이타의 병렬 컴퓨팅 환경에서는 다루어지는 데이타의 크기가 매우 크기 때문에, 디스크 저장 장치로의 부하 집중이 발생할 수 있고, 이는 전체 시스템의 성능을 떨어뜨리는 요인이 된다. 그러므로, 클러스터링 환경에서 병렬 수행되는 어플리케이션에는 디스크 입출력을 효

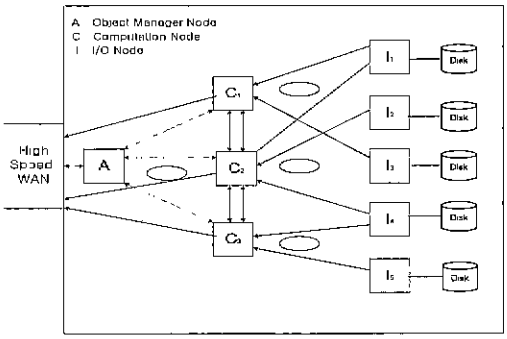

그림 3 병렬 파일시스템 개념도

율적으로 처리하기 위한 방법이 요구되는데, 근래에 데이타를 분산/병렬 저장하고 이를 동시에 접근할 수 있는 소프트웨어적 RAID 방안이 많이 연구되고 있다.

만일 데이타가 하나의 특정 워크스테이션의 저장 장치에만 저장되어 있으면, 이것을 접근하기 위한 병목현상이 발생하게 되고 이로 인한 실행 시간 지연이 심각해진다. 그러므로, 병렬 수행되는 대규모 시뮬레이션 또는 멀티미디어 어플리케이션에서의 디스크 접근을 효율적으로 처리하기 위해서, 데이타를 분산 저장하고, 이를 동시에 접근할 수 있고 단일 파일시스템 처럼 활용할 수 있는 병렬입출력 라이브러리와 분산 저장된 파일을 유지 관리 시켜주는 메타데이타 방안이 반드시 요구된다.

● 데이타 분산 저장, 동시 접근

대용량의 데이타를 관리하기 위한 파일 시스템의 성능은 데이타를 저장할 때의 분산 방법에 달려있다. 따라서, 시스템의 성능 향상을 위해서는 다양한 분산 방법을 지원해야 한다. 많이 사용하는 방법이 시스템 전체에 파일을 분산시키는 전체 분산(wide striping)이다.

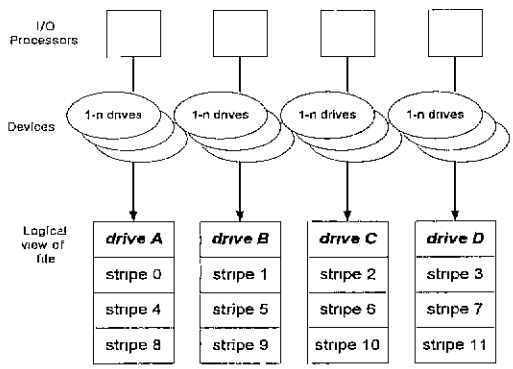

그림 4 병렬 데이타 분할 방법

병렬 파일시스템은 입출력 노드들에 접속되어 있는 다수의 디스크에 단일 파일을 분산 저장하고 사용자가 대규모 데이타의 입출력을 요구할 때 각 입출력 노드에 있는 디스크 제어기들을 동시에 데이타 블록 접근함으로써 파일 입출력의 병렬성을 제공한다.

● 데이타 동시 접근

병렬 파일시스템은 사용자에 요구에 대해 스트라핑에 참여한 모든 호스트가 작업을 하여 전체 접근 시간을 줄일 수 있다. 또한, 병렬 파일시스템은 동일한 파일에 대해서 여러 사용자가 동시에 접근할 경우가 많다. 따라서, 동시성 제어를 해야 하며 이는 공유 파일 포인터를 제공하여 처리한다. 공유 파일 포인터는 병렬 파일시스템의 특성을 이용하여 성능을 높이기 위해서 사용된다. 파일 삭제 작업은 공유 파일 포인터의 참조 횟수를 사용하여 처리한다.

또한, 데이타의 동시 접근시에 동기 모드로 접근되는 것과 비동기 모드로 접근되는 것을 모두 지원해야 해야 한다. 비동기 모드로 접근될 때에는 데이타가 사용 가능한 형태가 되는 것을 알리는 barrier 함수 등을 마련하여 어플리케이션 내에서의 동기화가 가능하도록 해야 할 것이다.

● 병렬 입출력 라이브러리

사용자에게 병렬 파일 시스템을 단일 파일 시스템(Single image view)으로 보이기 위해서 두 가지 종류의 병렬 입출력 라이브러리를 제공한다.

- 기존의 파일 시스템에서 제공하는 입출력 라이브러리로서 사용자의 요구(pull)에 따라서 응답하는 pull 형 입출력 라이브러리
  예) open, read, write, close
- stream 단위의 입출력 라이브러리로, 시작 명령(stream-start) 후에는 중지 또는 정지 명령(stream-close, stream-pause) 이 발생하기 전까지는 계속적으로 데이타를 제공하는 push 형 입출력 라이브러리
  예) sream-start, stream-close, stream-pause, stream-resume, stream-operation

● 캐싱, 선반입 방안

데이타 타입에 따른 hint가 sequential type인 경우에 선반입을 한다. 특히 push 형 함수인 경우에 선반입을 사용하면 안정된 성능을 볼 수 있다. 병렬 파일 시스템을 과학계산 응용 컴퓨팅 환경에 적용할 경우에는 hint에 따른 선반입에 따라서 성능이 크게 향상될 수 있다.

● 메타데이타

메타데이타는 병렬 파일 시스템에서 병렬 입출력 라이브러리를 만들 때, 각 구성 요소들에서 데이타의 분산 저장 상황 또는 데이타의 동시 접근 상황을 알기 위해서 요구된다. 메타데이타는 크게 두가지로 나뉘는데, 클러스터의 정보를 나타내는 메터데이타와 실제 데이타가 어떻게 분할 저장되었나를 알려주는 메타데이타로 구분된다. 메타데이타는 병렬 파일시스템에 저장된 파일을 관리,유지 시켜준다.

4. 결 론

본 논문에서는 현재 시스템공학연구소에서 개발 중인 클러스터 시스템인 CROWN(Clustering Resources On Workstations' Networks)의 구조에 근거하여 핵심 요소 기술 중심으로 기술하였다. CROWN은 대용량의 멀티미디어 정보를 고속으로 사용자에게 제공하며 다수의 사용자가 활용 가능한 대용량 멀티미디어 서버를 목표로 하고 있으며 범운영체제, 초경량 통신 소프트웨어, 고확장 파일 시스템으로 구성된다. 앞에서 살펴본 바와 같이 클러스터 시스템은 경제적인 가격으로 고성능 컴퓨팅 환경을 구성할 수 있다는 데 그 기술의 이점이 있다. 통신 장비의 고속화가 속도를 더해감에 따라 개별 시스템을 연동시키는 방법에 대한 연구는 더욱 활발해 질 것으로 보인다. 또한 데이타의 표현 형태로 단순한 텍스트에서 멀티미디어로 복잡해지고 방대한 규모로 증가하고 있어, 클러스터 시스템은 향후 정보 서비스 서버로 구축시 기반 기술로 활용할 수도 있다.

참고문헌

[1] Andrew S. Tanenbaum, Modern Operating Systems, 1992, Prentice-Hall.

[2] Thomas E. Anderson, David E. Culler, David A. Patterson, and the NOW Team, A Case for Networks of Workstations : NOW, IEEE Micro, Feb, 1995.

[3] Mark A. Baker, Geoffrey C. Fox and Hon W. Yau, Cluster Computing Review, NPAC Technical Report SCCS-748.

[4] Songnian Zhou, Xiaohu Zheng, Jingwen Wang, and Pierre Delisle, UTOPIA : A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems, University of Toronto.

[5] Robert Blumofe, Christopher F. Joerg, Bradley C. Kuszmaul, Charles E. Leiserson, Keith H. Randall, and Yuli Zhou, Cilk : An Efficient Multithreaded Runtime System, 5th ACM SIGPLAN Symposim on PPOPP, 1995.

[6] Jose Naglib, Cotrim Arabe, Adam Beguelin, Bruce Lowekamp, Erik Seligman, Mike Starkey, and Peter Stephan, Dome : Parallel programming in a heterogeneous multi-user environment, Technical Report CMU-CS-95-137, Carnegie Mellon University, April 1995.

[7] Thorsten von Eicken, David E. Culler, Seth Copen Goldstein, and Klaus Erik Schauser, Active Messages : a Mechanism for Integrated Communication and Computation, UCB/CSD, March 1992.

[8] Thorsten von Eicken, Anindya Basu, Vineet Buch, and Werner Vogels, U-Net : A User-Level Network Interface for Parallel and Distribtued Computing, Cornell University, Dept. of Computer Science, August 1995.

[9] Matt Welsh, Anindya Basu, and Thorsten von Eicken, Incorporating Memory Management into User-Level Network Interfaces, Cornell University Dept. of Computer Science, October 1996.

반 난 주

1988 아주대학교 전산학과 학사
1989~1990 금성소프트웨어
1994 미주리 주립대학교 석사
1994~현재 시스템공학연구소
관심분야 : 병렬/분산 컴퓨팅, 운영체제

김 태 근

1987 한양대학교 수학과 학사
1990 뉴욕주립대학교 석사
1992~현재 시스템공학연구소, 분산컴퓨팅연구실장
1993 뉴욕주립대학교 박사 (Parallel Processing 전공)
관심분야 : 병렬/분산 컴퓨팅, 병렬 컴파일러, 멀티미디어 서버 기술, 네트워크 컴퓨팅 기술

● '97 소프트웨어공학 추계튜토리얼 ●

- 일 자 : 1997년 9월 3일(수)~4일(목)
- 장 소 : 한국과학기술회관
- 주 제 : 웹기반 분산객체 컴퓨팅 기술
- 주 최 : 소프트웨어공학연구회
- 문 의 처 : 서울대학교 전산과학과 최순규 교수
  Tel : 02-880-6573
  E-mail : skchoi@selab.snu.ac.kr