초청논단

과학계산전용 병렬처리 컴퓨터 구조
High Performance Parallel Computer for Scientific Computations

朴圭晧, 丁鳳浚, 崔鐘赫,
洪煖誠, 沈揆玄, 李珩碩,
金鍾旭*, 曺常榮**

한국과학기술원
*대우 통신
**미국 U.C. irvine대

요 약

KAICUBE/한빛-1호는 하이퍼큐브 형태의 연결망을 가진 병렬 컴퓨터이고 각 노드는 i860 프로세서와 통신용의 i82380 DMA 콘트롤러를 탑재하고 있다. 40Mhz CPU 클럭을 사용하는 32노드로 구성되어 있고 컴퓨터의 최고 속도는 2.5G-flops 정도로써 이것은 국내 최초의 Giga급 컴퓨터 이다. DMA 콘트롤러에 의해 구동되는 노드간 통신은 채널 대역폭이 100Mbps정도이다. 0번 노드는 UNIX를 탑재한 호스트 컴퓨터와 연결되어 있고 호스트 컴퓨터는 병렬 프로그래밍 환경과 각 노드를 관리하는 역할을 한다. 익스프레스는 호스트 컴퓨터에 탑재된 병렬 운영 체제이고 사용하기 간편한 사용자 환경과 프로그래밍 방법에 따라 호스트-노드방법과 cubix 프로그래밍 환경을 각각 제공한다. 그밖에 고수준의 병렬 프로그래밍 환경으로써 기존의 순차 프로그램에 기초한 입력 프로그램을 병렬 프로그램으로 자동 변환 해주는 KAPPA가 있다. 여러 분야의 과학 계산용 프로그램이 수행되고 있으며 그의 성능 측정을 통하여 탁월한 성능을 보여 주었다. 보다 편리한 병렬 프로그래밍 환경의 개발과 범용 계산 전용 서버로써 자유로이 사용할 수 있도록 네트워크 기능을 강화하는 일이 남아있다.

I. 서 론

컴퓨터의 발달과 더불어 처리해야 할 데이터의 수도 점차 방대해지고, 계산량 또한 기하 급수적으로 늘어났다. 이와 같은 이유로 좀더 빠른 처리 속도를 가진 컴퓨터의 개발을 절실히 필요로 하지만 현재의 VLSI 기술은 반도체 기술에 의한 한계를 보이고 있고, 전체적인 고성능화를 위해서 주변 장치의 고속화에 따른 추가적인 비용이 따른다.

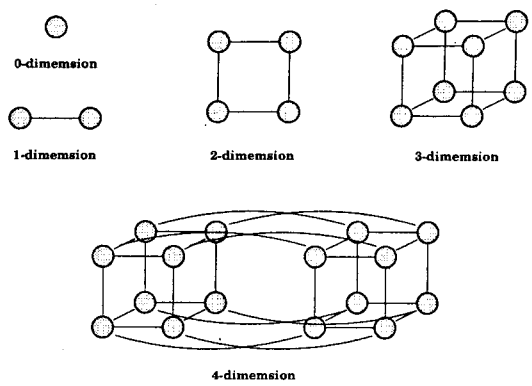
따라서, 비교적 저가의 비용을 들이면서 고성능 시스템을 만들기 위해서 CPU 내부의 병렬화에서

여러 개의 CPU를 사용한 병렬 처리에 이르기까지 다양한 방법의 병렬화 방법을 연구하고 있다 [Hwang93].

첫째로는 CPU 내부에서의 병렬화 방법으로써, 각 명령어들을 겹치며 수행, 즉, 파이프라인시킴으로써 수행성능 향상을 얻는 방법이 있다. 수퍼 스칼라와 벡터 프로세서 방식이 있는데, 전자는 CPU 내부에 계산블럭－부동 소수점 계산블럭, 정수 연산 블럭이나 처리 블럭－을 복수개로 두고 명령어(instruction)의 순서에 따라 각각의 블럭에서 동시에 처리할 수 있는지 판단하여 동시에 처리하는 방식이다. 이 경우 CPU 내의 집적도와 크기의 한계로 인해 블럭의 갯수가 한정되기 때문에 얻을 수 있는 성능향상은 수배 정도가 된다. 후자의 벡터 프로세서는 CPU 내부의 계산 모듈을 벡터화하여 벡터계산 명령어에 대해 각 벡터 데이터를 동시에 하는 방식이다. 이 방법은 계산 모듈의 수에 따라 성능 증가가 있고 벡터 연산에 대해서는 고속처리가 가능하지만 비벡터 연산에 대해서는 성능향상을 얻을 수 없으며 수퍼 파이프라인 방법과 같이 여러 개의 벡터 모듈에 의한 CPU의 크기가 커지므로 비용이 아주 비싼 단점이 있다.

둘째로는 여러 개의 CPU를 이용한 다중 컴퓨터로써, 한 개의 메모리와 여러 개의 프로세서로 구성되는 공유메모리 다중 프로세서(shared-memory multiprocessor) 방식과 CPU와 메모리가 하나의 모듈을 이루고 이들 사이에 연결망을 가진 분산메모리(distributed-memory) 혹은 메시지 전달형(message-passing) 다중 컴퓨터 방식이 있다. 다중 컴퓨터는 종래의 단일 CPU를 사용한 컴퓨터에 비하여, 대량의 계산 문제를 분할하여 병렬처리 함으로써 CPU를 많이 써서 큰 성능향상을 기대할 수 있고, 여분(redundancy) 개념을 이용하여 고장감내(fault-tolerant) 시스템의 구현이 가능하며, 확장성 또한 좋은 장점이 있다.

메시지－전달형 다중 컴퓨터의 연결망 형태는 링(ring), 메쉬(mesh), 하이퍼큐브(hypercube) 등으로 분류가 될 수 있다. 이 중 하이퍼큐브는 확장성(scalability)과 연결성(connectivity)면에서 다른 연결망 형태에 비해 우수함이 이론적으로 증명



〈그림 1〉 여러 가지 차원의 하이퍼큐브 구조

되었다. 이진－n－큐브(binary-n-cube)라고 말할 수 있는 하이퍼큐브는 차원(dimension)이라는 파라메터(parameter)에 의해 특징지워지며 이 값에 따라 노드의 수와 링크의 수가 결정된다. 몇 개의 차원에 따른 하이퍼큐브 구조를 그림 1에 보였다.

1차원이 낮은 2개를 연결하여 새로운 차원을 형성하는 구조이며 n차원 하이퍼큐브 구조는 아래와 같은 특성이 있다.

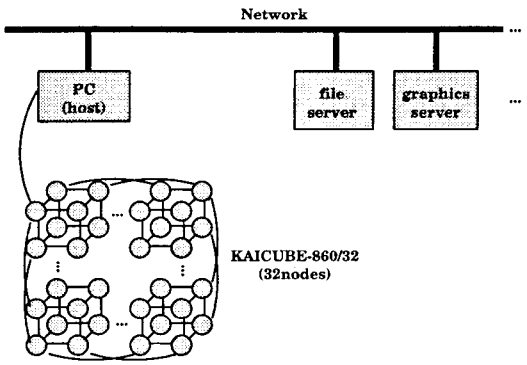
- 노드의 갯수는 $N = 2^n$이다.
- 각 노드의 링크의 수는 $n+1$개이다.
- 최대의 거리(distance)는 n이다.
- 각 노드간의 평균거리는 $\frac{n}{2}$이다.

메세지－전달형 다중 컴퓨터의 연결망 형태로 하이퍼큐브 형태를 사용할 경우 많은 수치 계산 프로그램의 경우 분할한 전체구조가 잘 매핑되고, 강력한 연결성을 가지며, 링, 메쉬, 트리 등의 일반적인 연결구조는 모두 하이퍼큐브구조로 쉽게 구현할 수 있는 장점이 있다. 또한, 메세지 통신 알고리즘(algorithm)이 간단하고 복수 개의 경로를 가지므로 통신능력의 측면에서 볼 때 효율적이다. 일정한 확장 방법을 가진 하이퍼큐브는 하드웨어의 제작뿐만 아니라 전체 시스템의 확장에 있어서도 쉽고 간편한 특성을 가진다.

하이퍼큐브형 메세지－전달형 다중 컴퓨터는 분류상 MIMD(Multiple Instruction stream , Multiple Data stream)형에 속하며, 많은 수의 프로세서와 연결 확장이 용이하고, 다른 형태의 연결망으

로 매핑이 용이하다는 많은 장점이 있다. 이러한 하이퍼큐브 컴퓨터는 California 대학에서 1983년 처음으로 발표된 이래 많은 대학에서 이에 대한 연구를 진행하였으며, 기업체에서도 제품화가 되어 Intel의 iPSC와 NCUBE사의 NCUBE 등이 발표되었다[Flyn72].

국내에서도 90년 KAIST에서 KAICUBE가 발표된 이래 93년 8노드의 KAICUBE-860이 발표되었으며, 현재 32노드의 하이퍼큐브형 컴퓨터 KAICUBE/한빛-1호기가 완성되었다. 특히 국내 최초의 Giga급 성능의 컴퓨터이므로 한빛 1호로 명명하였다. 이 시스템의 전체 구조는 그림 2과 같으며 최대 2.56G-flops의 성능을 낼 수 있다. 본 논문에서는 32노드 KAICUBE/한빛-1호의 하드웨어 및 소프트웨어 구조에 대해 알아보고, 각종 프로그램에 대해 성능 평가를 하였다.



〈그림 2〉 KAICUBE/한빛-1호 시스템의 블럭 다이어그램

II. KAICUBE/한빛-1의 하드웨어

1. 단위 보드의 특징

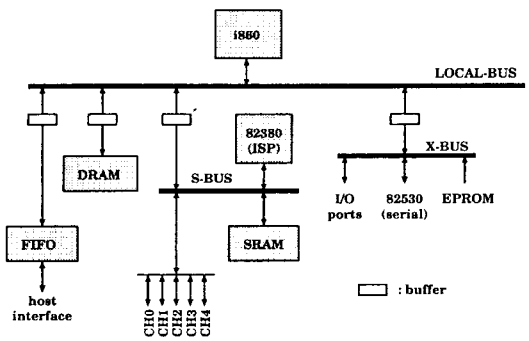
단위 노드 컴퓨터는 하이퍼큐브 컴퓨터 시스템의 최소 구성 단위가 되는 컴퓨터이고 노드 컴퓨터의 성능이 전체 하이퍼큐브 컴퓨터 시스템의 성능과 직결되므로 중요한 부분이라 볼 수 있다. KAICUBE/한빛-1에서는 고성능 마이크로프로세서와 주변장치 칩을 사용하여 연산 속도와 통신 속도의 성능을 극대화하였으며, 다음과 같이 주요 부분으로 구성되어 있다.

- 64비트 i860 마이크로프로세서(40MHz)
- DRAM 주메모리(8MB, 64비트)
- 2개의 i82380 Integrated System Peripheral (ISP)
- SRAM 통신용 버퍼 메모리(128KB, 32비트)
- 5개의 통신 채널

노드 컴퓨터는 미국 인텔사의 64-비트 RISC 마이크로프로세서인 i860-XR을 프로세서로 사용하여 빠른 수행 속도를 가질 수 있도록 하였다. i860은 부동 소수점 연산 장치와 캐쉬 메모리를 내장하고 40MHz의 동작 속도에서 최대 80M-flops의 부동소수점 연산 속도를 낼 수 있는 고성능 프로세서이다. 또한 빠른 메모리 버스 동작을 할 수 있는 설계를 위한 파이프라인 기능을 제공하고 있다[Intel90a][Intel90b].

[Daniel85]. 통신용 프로세서를 독립적으로 사용하지는 않았지만 인텔사의 i82380 고집적 주변장치(ISP)내에 포함된 DMA 콘트롤러를 사용하여 노드 컴퓨터 사이의 데이타 전송을 제어하며, i860 프로세서와 독립적으로 통신을 수행하도록 함으로써 효율적인 통신을 가능하도록 하였다. DMA는 20MHz의 클럭에 동기되며 3클럭당 16비트를 전송하므로 실제 채널의 통신속도는 100Mbps가 된다.

그림 3는 각 노드의 블럭다이어그램을 보여준다.



〈그림 3〉 단위 노드의 블럭 다이어그램

노드 컴퓨터에 있는 여러 디바이스들은 5개의 독립된 버스에 나뉘어져서 위치해 있다. 각 버스는 신호구동능력과 같은 물리적인 이유 또는 버스 파이프라인과 버스 공유와 같은 구조적 이유 때문에 사용한 것이다. 각 디바이스에 대한 어드레스는 i860에서 제공하는 범위를 캐쉬 메모리를 이용하는지의 여부와 각 디바이스가 위치한 버스에 따라서 구분지어서 할당하였다.

●CPU 버스

CPU 버스는 i860 프로세서가 연결되어 있는 버스로서 64비트의 데이타 폭을 갖는다. 어드레스와 데이타는 다른 버스와 버퍼(buffer) 또는 래치(latch)를 통해 연결되어 있고 i860이외의 다른 디바이스는 연결되어 있지 않다.

●DRAM 버스

i860에서 지원하는 3단계 버스 파이프라인 동작을 할 수 있도록 CPU와 DRAM사이에 어드레스와 데이타용 래치를 두어서 DRAM이 별도의 DRAM 버스(D-Bus)에 연결되도록 하였다. DRAM 버스의 데이타 버스는 64-비트의 데이타 폭을 갖도록 되어 있어서 파이프라인 동작과 더불어 빠른 메모리 동작을 가능하게 한다.

●SRAM 버스

노드 컴퓨터는 SRAM을 통신용 버퍼로서 사용하고 두 노드 컴퓨터 사이의 데이타 전송은 CPU가 아닌 DMA 콘트롤러에 의해서 제어되게 하였다. SRAM은 다른 디바이스와 달리 DMA 콘트롤러와 CPU에 의해서 제어를 받게 되므로 두 버스 마스터가 버스를 공유할 수 있는 별도의 버스인 SRAM 버스(S-Bus)를 두어서 이 버스에 연결하였다. 이렇게 함으로써 DMA 수행시에 CPU는 다른 버스에 대한 동작을 독립적으로 수행할 수 있게 된다. 이 버스에는 SRAM 이외에도 DMA 콘트롤러가 포함된 두 개의 82380 ISP가 위치한다. 이 버스의 데이타 폭은 32비트이다

●I/O 버스

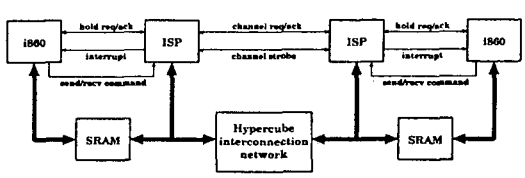
I/O 버스-X 버스-는 i860만이 사용하는 기타 디바이스들이 위치한 버스이다. 이 버스에는 시스템의 초기화 프로그램이 들어 있는 EPROM과 RS-232C 통신을 위한 82530 SCC가 있고 기타 노드 컴퓨터 자체와 통신의 상태 점검과 제어를 위한 port들이 연결되어 있다. 이 버스는 8-비트의 데이타 폭을 갖는다.

KAICUBE/한빛-1의 노드 보드에서 2레벨 캐쉬(cache)를 사용하지 않은 이유는 i860내부에서 제공하는 버스 파이프라인(bus pipeline)과 DRAM의 정적 칼럼 모드(static column mode)를 사용하여 거의 SRAM의 빠르기로 DRAM을 읽을 수 있고, i860내장 캐쉬가 충분히 크기 때문이다. 특히 i860은 부동 소숫점 연산이 하나의 CPU 클럭에 이루어지기 때문에 과학 계산용으로는 적격이라 볼 수 있다.

2. 단위 노드의 통신 기능

하이퍼큐브 컴퓨터는 전체적으로 매우 정형적인 통신 구조를 가지고 있다. 차원이 하나씩 증가할 때 마다 각 단위 컴퓨터는 새로운 차원의 대응 단위 컴퓨터와 통신을 하기 위해서 통신 채널을 하나씩 증가시켜야 한다. 5-차원의 하이퍼큐브컴퓨터의 경우에는 각 단위 컴퓨터는 5개의 통신 채널을 가지고 이웃하는 컴퓨터와 통신을 하게 된다. 최대 32개의 단위 컴퓨터를 가지는 병렬 컴퓨터까지 지원할 수 있으며 쉽게 더 높은 차원의 하이퍼큐브 컴퓨터로 확장할 수 있는 구조로 되어 있다. 우리가 사용한 DMA는 INTEL사의 82380으로 8개의 채널을 제공하며 최대 50M-Bytes/sec의 데이타 전송 기능을 가지고 있다. 현재는 단위 컴퓨터에 2개의 DMA가 있는데 각각은 메세지를 보내고 받는 용도로 쓰인다. 그림 4는 채널 5개의 경우 통신 부분의 간단한 블럭 다이어그램이다.

전체 통신을 개괄적으로 살펴보면 먼저 다른 단위 노드로 메세지를 보내고자 할 때는(send) CPU



〈그림 4〉 노드간 통신의 블럭 다이어그램

가 보내고자 하는 메세지를 SRAM에 저장하고 ISP를 초기화시킨 뒤에 메시지의 길이와 주소(address)등의 정보를 ISP 내부 레지스터에 기록한다. 이러한 레지스터는 각 채널마다 하나씩 총 7개가 있으며, ISP는 레지스터의 내용을 바탕으로 통신 채널을 따라 대응하는 컴퓨터의 ISP에게 메세지를 받아야 함을 알리는 channel request 신호를 전달해 준다. 이 때, 대응되는 노드가 받을 준비가 되어 있으면 다시 채널을 따라 channel ack를 주어 받을 수 있음을 알려주고 이것이 인지되면 두 단위 컴퓨터는 ISP 통신을 시작하게 된다. 대상 노드가 다른 통신을 진행하고 있어 받을 준비가 되어 있지 않다면 보내는 쪽의 ISP는 계속 기다리게 된다. 하지만 CPU는 단지 ISP에게 명령만 주기 때문에 이 때에 다른 작업을 계속 진행할 수 있다. 그리고 통신이 끝나게 되면 각각 CPU에 인터럽트(interrupt)를 걸어 통신의 종료를 알려 주게 된다. 통신 관련 로직은 채널이 전이중(full-duplex) 방식으로 사용되기 때문에 한 채널을 동시에 두 컴퓨터가 send용으로 사용하고자 할 때는 이것을 조정할 수 있게 하여야 하고 송신, 수신용의 ISP가 SRAM을 동시에 사용할 수 없기 때문에 이것에 대한 조정을 하는 기능이 필요하다. 또한 여러 채널에서 동시에 send 또는 receive 요청이 들어 올 수 있는데 이는 DMA 자체에서 제공하여 주는 우선순위(priority) 조정 기능을 사용하여 해결하였다.

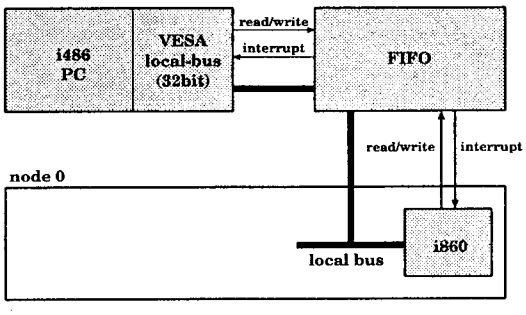
3. 호스트 컴퓨터

KAICUBE/한빛-1에서 하이퍼큐브 구조로 연결된 노드 컴퓨터들은 계산 전용으로서 사용되고 소프트웨어의 개발과 병렬 컴퓨터의 사용에 대한 관리, 그리고 네트웍과 저장 장치등의 여러 가지의 주변 장치들은 호스트 컴퓨터를 통해서 이루어진다. 먼저 사용자의 관리 부분을 살펴보면 여러 사용자가 동시에 시스템을 사용할 수 있도록 다중 사용자 환경을 제공하여야 한다. 전체 32노드는 여러 개의 작은 단위의 하이퍼큐브로 나뉘어져 여러 사용자가 동시에 사용할 수 있어야 하는데 이의 관리를 위해서는 호스트 컴퓨터에서 각각의 사용자에게 적절한 단위의 작은 하이퍼큐브 시스템을 할당할 수 있어야 한다. 그리고 모든 하이퍼큐브 시스템의 접근(access)는 호스트 컴퓨터를 통해서 수행되기 때문에 여러 사용자의 메세지가 자신의 할당받은 단위에게로 정확히 전달될 수 있도록 메세지 조정기능을 가지고 있어야 하며 메세지의 전달을 효율적으로 처리할 수 있어야 한다. 이 기능들은 Express라는 병렬, 분산 O.S.에 의해서 처리되며 4장에서 자세히 설명한다.

수행할 수 있는 응용 프로그램의 개발은 호스트에서 진행되기 때문에 호스트 컴퓨터는 병렬처리 프로그램을 위한 컴파일러와 디버거를 가지고 있어야 한다. 더욱이 병렬 프로그램은 순차적 프로그램에 비해 개발하기가 어렵기 때문에 자동화된 사용자 환경인 KAPPA가 개발되었다.

일반적으로 메시지-전달형 분산메모리 다중 컴퓨터에서는 호스트가 사용자 인터페이스를 가지고 있기 때문에 호스트-노드 간에 필요한 통신의 수가 노드간 보다 월등이 많다는 것이 알려져 있다. 그런데 호스트-노드 간에도 노드간의 통신방식을 그대로 쓸 경우 병목현상으로 인한 성능저하로 나타난다. 이 점을 극복하기 위해 KAICUBE/한빛-1의 경우 속도가 빠른 FIFO를 통하여 호스트-노드간 통신이 이루어지며 노드 0번과 직접 연결되어 있다.

그림 5에 노드 0번과 호스트 컴퓨터 사이의 연결형태를 보였다. 호스트 컴퓨터가 PC 호환기종이므로 잘 알려진 VESA 버스를 통해 노드 0번과 연결되며 32비트 전송을 할 수 있다. 즉 실제적



〈그림 5〉 호스트-노드간 통신 블럭 다이어그램

통신 대역폭은 640Mbps 정도에 이른다. 노드간의 통신처럼 호스트도 가상 노드 번호를 가지고 모든 노드와 직접 통신을 할 수 있도록 설계되었다.

III. 단위 노드의 소프트웨어

1. 초기화 진단 프로그램

상위 레벨의 소프트웨어가 실행되기 전에 각종 하드웨어의 초기화(Hardware Initialization), Diagnostic (메모리, I/O, 통신등의 테스트에 필요한 복잡한 초기화 및 테스트) 등의 기능을 수행하는 프로그램을 초기화 진단 프로그램이라 한다. 이 프로그램은 모니터나 커널을 메모리에 로딩하여 실행하는 부트스트래핑(bootstrapping)의 기능을 포함한다.

8비트의 시스템 ROM에 들어 있으며 부팅시에 CPU는 이 초기화 진단 프로그램을 읽어 들여 수행한다. 주 메모리인 DRAM은 그 리프레쉬(refresh)를 위한 하드웨어를 초기화하기 전에는 사용할 수 없기 때문에 파워업(power up) 혹은 리셋(reset)에 의하여 i860은 8비트 버스 모드(CS8)로 동작을 시작한다. CS8 모드는 명령 캐쉬 미스(instruction cache miss)가 발생할 때 8비트 버스 사이클로 명령을 메모리로부터 읽어 들이는 CPU의 초기 모드이다. 이 모드를 CPU 내부에 내장함으로써 64비트 RAM을 사용하기 전 부트 스트랩(bootstrap)용으로 잠시 사용할 롬(ROM)을 8비트 포트로 구성하여 하드웨어 부담(overhead)를 줄일 수 있는 장점이 있다.

이 초기화가 끝난 다음에 각종 메모리 장치들을 진단하는 프로그램이 수행된다. 이 과정을 통하여 메모리 장치의 불량과 각종 버퍼(buffer)들의 고장 유무를 실제 사용자가 프로그램을 수행 하기전 미리 진단하여 수행시 발생할 수 있는 하드웨어 적인 오류를 사전에 예방할 수 있다.

마지막 과정으로 ROM의 내용중 보다 상위 레벨의 모니터 혹은 커널 프로그램을 64비트 램에 옮기고 64비트 버스 모드로 스위칭하는 작업을 마지막으로 초기화 진단 프로그램을 수행을 모니터로 넘긴다.

2. 모니터 프로그램

모니터 프로그램은 하드웨어의 테스트와 소프트웨어의 개발을 도와주는 목적을 가진다. 앞 절에서의 기초적인 초기화 진단 프로그램의 간단한 테스트보다 더 나아가 모니터 레벨에서는 사용자의 의도대로 보다 복잡하게 테스트 해 볼 수 있다. 이를 위하여 여러 가지 하드웨어 테스트를 위한 명령들을 제공할 뿐만 아니라, 특수하게 설계된 하드웨어 테스트 프로그램을 로딩하여 실행시켜 볼 수 있는 기능을 제공한다. 또한 새로 설계된 프로그램을 쉽고 빠르게 로딩하여 실행시켜 볼 뿐만 아니라, 에러를 찾는데 도움을 주는 디버깅 기능을 제공한다. 이와 같이 함으로써 궁극적으로 노드에서 수행될 커널 및 각종 소프트웨어의 개발을 용이하게 하도록 한다.

전체 모니터 프로그램은 다음과 같은 소프트웨어 구성요소들을 가진다. 첫째, 시스템의 초기화를 들 수 있다. 모니터 레벨의 초기화는 ISP 클럭 타이머와 DMA 채널, 그리고 인터럽트 등에 대한 보다 복잡한 하드웨어 초기화 작업과, 시스템의 여러 변수들에 대한 소프트웨어 초기화 작업을 포함한다. 둘째는 트랩(trap) 처리루틴 인데, 리셋, 에러, 그리고 인터럽트(interrupt)에 의한 트랩에 대한 적절한 처리는 모니터 환경의 기본 소프트웨어로서 반드시 필요하다. i860과 같은 RISC 컴퓨터는 하드웨어를 간소화하여 일의 많은 부분을 소프트웨어로 넘겨 주는데, 특히 트랩에 대한 처리 소프트웨어가 복잡하므로 이 부분은 중요하고도 많은 주의를 요하게 된다. 특히 KAICUBE/한빛-1의 통신이 대부분 인터럽트에 의해 처리되므로 처리 소프트웨어를 어떻게 효율적으로 만드느냐가 전체 시스템의 성능에 많은 영향을 주게 된다. KAICUBE/한빛-1에서는 이런 곳에서는 i860 어셈블리(assembly) 언어를 사용하여 간결하고, 속도를 빠르게 처리를 하였다. 셋째는 사용자 프로그램의 다운로드 기능이다. 다운로드 프로그램은 호스트 측에서 개발된 노드 컴퓨터용 프로그램을 호

스트-노드간의 FIFO와 노드-노드간의 DMA 저수준 IO 루틴을 사용하여 실제 수행될 노드 프로세서로 다운로드 한다. 특이할 만한 점은 실제 각 노드의 번호는 하드웨어적으로 정해지는 것이 아니라 프로그램의 다운로드시 정해지게 되며 각 노드는 이 때 정해진 번호를 가지고 서로 통신을 하게 된다.

3. 커널 프로그램

커널 프로그램은 전체 소프트웨어에서 가장 중요한 부분중의 하나로써 전체 단위 노드의 관리를 수행하고 다른 노드 및 전체 시스템 관리 소프트웨어와 연결되는 핵심적인 프로그램들로 구성된다. 커널 프로그램의 내용은 크게 노드의 리소스(resource)관리 부분과 통신 관련부분으로 나누어 볼 수 있다. 각 노드의 리소스 관리는 일반 단일 프로세서에서의 O.S.와 같은 맥락의 CPU 관리와 메모리 할당, I/O 관리를 하게 된다. CPU관리라 함은 다중 작업(multi-tasking)과 같은 기능을 제공함을 말하는 것이고 메모리 할당과 제거, 다중 사용자 환경에서의 메모리 방어(protection) 등의 기능을 말한다.

통신관리의 측면에서 본다면 사용자의 입장에서 보다 간편한 통신 프리미티브(communication primitive)를 제공하여 병렬 프로그램의 작성이 최대한 기존의 순차적 프로그램의 작성방법에 가깝도록 해 주는 작업이다. 이를 위해 각 노드 커널은 동기(synchronous) 및 비동기(asynchronous)통신의 수행, 노드간의 1대1 통신, 멀티캐스팅(multi-casting), 방송(broadcasting) 등의 다양한 기능을 가지는 통신관련 기능들을 가지고 있다.

IV. 프로그래밍 환경

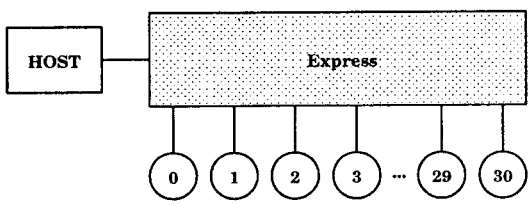
1. 익스프레스(express)

익스프레스는 Parasoft에서 만들어진 병렬 컴퓨터를 위한 O.S.이며 병렬 프로그램을 위해 기본적인 기능(functionality)을 제공한다. KAICUBE/한빛-1은 호스트 컴퓨터의 병렬 O.S.로써 익스프레스를 사용한다. 여기에서의 병렬 O.S.란 호스트 컴퓨터를 운용하는 O.S.와는 별개의 개념이며 호스트의 O.S.를 도와 노드 컴퓨터를 관리하고, 병렬 수행을 위한 통신 패키지와 디버깅기능을 제공하는 것을 의미한다[ParaSoft92].

공유-메모리 구조에서는 복잡한 잠금(locking) 기능과 세마포어(semaphore)기능을 사용하여 프로그래밍을 하여야 한다. 그리고 SIMD(single instruction multiple data) 컴퓨터 구조에서는 fine grain 병렬수행을 위한 독자적인 언어를 대부분 사용하거나 C나 Lisp과 같은 기존의 언어에서 확장된 언어를 사용한다. 반면에 KAICUBE/한빛-1과 같은 분산-메모리 구조는 비교적 유연한 확장성 때문에 대부분의 프로그래밍 모델에서 이 구조를 잘 지원한다. 그러나 하드웨어는 여러 가지의 모양으로 존재하기 때문에, 특별한 병렬 O.S.가 없다면 프로그래머는 이 모양에 따라 프로그램을 다르게 바꾸어 주어야 한다. 익스프레스는 C나 FORTRAN과 같은 고급 언어, 저수준의 통신라이브러리, 병렬 디버깅과 같은 다양한 프로그래밍 환경을 제공함으로써 프로그래머가 쉽게 접근하는 방법을 제공한다.

위의 방법외에도 사용하는 병렬 컴퓨터의 자세한 하드웨어 정보를 숨기고 일률적인 프로그래밍 모델을 제공하는 것이 필요하다. 그림 6는 익스프레스를 통하여 본 병렬 컴퓨터를 보여준다. 이 모델을 바탕으로 프로그래머는 단일 추상화된 환경하에서 쉽게 프로그래밍을 하게 도와준다.

익스프레스는 두 개의 서로 독립적인 프로그래밍 모델을 제공하며 프로그래머는 응용 프로그램



〈그림 6〉 Express를 통해 본 KAICUBE/한빛-1

의 특성에 맞게 이 두 가지 모델 중 하나를 택하여 사용한다.

Cubix 모델은 비교적 개념적으로 간단한 모델이다. 호스트에서는 cubix라는 통일된 호스트 프로그램을 수행하며 각 노드에서는 호스트의 I/O나 그래픽처리를 할 수 있는 투명성(transparency)을 보장한다. 이 I/O는 노드 프로그램의 수행 라이브러리(runtime library)를 통해 제공되며 호스트 컴퓨터의 각종 I/O장치나 그래픽 터미널 등을 직접 제어하는 것처럼 사용할 수 있다. 특별히 그래픽처리를 위해 plotix 라이브러리가 제공된다.

다른 모델은 호스트-노드 모델이며 익스프레스에서 제공하는 라이브러리에 의해서 각각 독립된 호스트 프로그램과 노드 프로그램을 작성한다. 이 때에 호스트 프로그램은 병렬 처리를 위한 라이브러리외에 호스트 컴퓨터의 각종 자원을 충분히 이용할 수 있는 장점이 있는 반면 cubix 모델에 비해 호스트 프로그램을 따로 작성해야 하는 단점이 있다. 일례로 노드 컴퓨터에서 I/O를 하기 위해서는 호스트로 메시지를 보내어 호스트 프로그램의 I/O 처리 루틴을 이용하여야 한다.

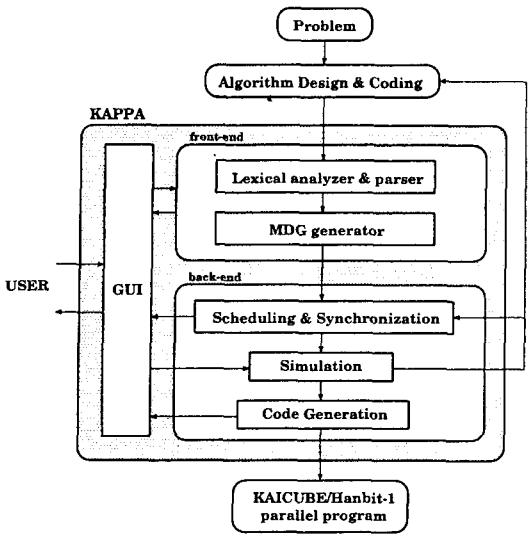
2. KAPPA

메세지 전달 방식의 많은 상업적 병렬 컴퓨터들은 단일 프로그램 다중 데이타(single program multiple data, SPMD) 형태의 프로그램 기술방식을 택하고 있다[Wu90]. 이 방식은 여러 단위 컴퓨터가 같은 프로그램을 수행하면서 각 컴퓨터의 노드 번호에 따라 자신의 메모리에 있는 데이타에 대해 프로그램의 다른 부분이 수행을 하는 형태이다. 현재 KAICUBE/한빛-1에서도 병렬 프로그램 설계의 용의성으로 이 방식을 택하고 있다. 그러나 이러한 방법은 비정규적 형태의 문제에 경우 각 단위 컴퓨터간의 부하균 등을 이루는 것이 쉽지 않다.

SPMD 형태의 프로그램 개발은 다양한 병렬화 방법이 존재하기 때문에 프로그래머가 병렬화를 모두 하는 것이 대부분이다. 이와는 다른 접근으로써 순차적 프로그램을 재구성 컴파일러로 병렬성을 추출하여 병렬 프로그램으로 변환할 수 있게 해야한다는 방식을 도입하였다. 아주 전문적인 프로그래머가 아닌 이상 프로그램의 개발이 그리 간단하지 않기 때문에 앞으로의 병렬 프로그래밍 환경에 대한 연구는 병렬 프랜스레이터 또는 지능적으로 자동으로 병렬화하는 컴파일러에 대해 진행되고 있다. 실제로 공유메모리 방식의 병렬 컴퓨터에서는 for 반복 순환문 분할과 같은 자동 병렬 변환에 대하여 많은 업적이 이루어져 실제로 사용되고 있는 상황이다. 그러나 분산메모리 시스템의 경우에는 통신에 의한 과부담과 실제 프로그램의 분할(decomposition)문제 등으로 인해 복잡한 입력 프로그램의 분석, 스케줄링, 매핑 등의 과정이 필요하며, course-grain 형태의 트랜스레이터가 필요하다.

KAPPA (KAICUBE's Parallel Programming Aid)는 병렬 프로그램을 개발하는데 있어서 두 가지 방식의 절충적 입장을 취해 문제가 주어졌을 때 기본적인 병렬 알고리즘은 사람이 생각해내고 이 병렬 알고리즘에 대해 병렬 시스템이 수행할 수 있도록 문제를 분할하고 스케줄하는 일은 자동화하여 생성한다. 즉, 설계자가 적절한 알고리즘을 설계하고 분할을 수행한 뒤에 몇 개의 프로시져로 구성된 프로그램을 작성한다[Choi92]. 이 프로그램은 순차적 프로그램 형태이기 때문에 디버깅을 위해서 순차적 머신에서 수행을 시킬 수도 있다. 이러한 프로그램은 병렬 트랜스레이터의 병렬 코드 합성과 최적화를 통해 메세지 전달 방식인 KAICUBE/한빛-1에서 수행될 수 있는 병렬 프로그램으로 바뀐다. 그림 7은 KAPPA의 구성을 보여준다.

lexer와 parser는 주어진 프로그램으로부터 프러시저 단계의 데이타 연관성(data dependencies)을 찾아내고 이를 바탕으로 그래프 생성기는 스케줄링과 매핑을 위한 단방향 그래프(directed graph)를 만들어 낸다. 여기서 각 노드는 각 프러시져를 나타내고 그 값은 프러시저가 수행될 때의 수행시간을 가지게 된다. 그리고 각 에지는 각 프로시저간의 데이타 연관성을 의미하여 그 값은 두 프러시저간의 통신량이 된다. 그다음 스케줄링은 각각의 노드를 수행속도가 최소로 될 수 있도록 각 프로세스에 할당하는 일을 한다. 매핑은 할당된 프



〈그림 7〉 KAPPA의 블럭 다이어그램

로세스를 통신시간을 최소화할 수 있게 각 단위 컴퓨터에 할당하는 일을 한다. 그 뒤에 동기화는 올바른 수행을 위해 통신 프리미티브(primitive)을 삽입하는 일을 하고 최종적으로 코드 생성기가 병렬 프로그램을 만들어낸다.

V. 성능 측정을 위한 프로그램

1. 응용 프로그램

성능 측정을 위한 예제로써 각종 수치연산 프로그램 및 시뮬레이션 프로그램을 선택하였다. 이것들은 실제 컴퓨터 사용시 시간을 요하는 프로그램들이며 대부분의 경우 프로그래머가 병렬화를 하였다. 예제들은 다음과 같다.

● 행렬 연산

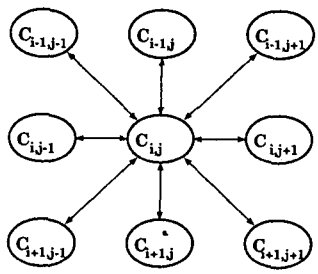
행렬의 연산을 위해 병렬 가우시안 소거법(Gaussian elimination)을 이용하였다[Fox88].

● 반도체 열 분포 해석 프로그램

4층으로 구성된 반도체에서 1층에 열원이 있을 경우 그 열이 확산되어 나아가는 것을 시뮬레이션 한 프로그램이다. 이 프로그램에서는 FDM(Finite Difference Method)의 방법을 써서 열 확산을 시뮬레이션 하였으며 SOR(Successive Over Relaxation) 방법을 써서 해에 수렴하는 시간을 줄였다[Fox88]. 병렬화를 위하여 전체 반북 계산 영역을 프로세서의 갯수만큼 분할 한 다음 각 프로세서끼리 파이프라인 처리하여 병렬화하였다.

● Mean 필터링(filtering)

화상처리에 쓰이는 필터링방법의 일종이며 이웃한 점들과의 평균값이 현재 점의 값이되는 계산방식을 가진다. 즉, 그림 8에서 C_{ij}를 계산하기 위해서 이웃한 8개의 점의 값을 이용하게 된다.

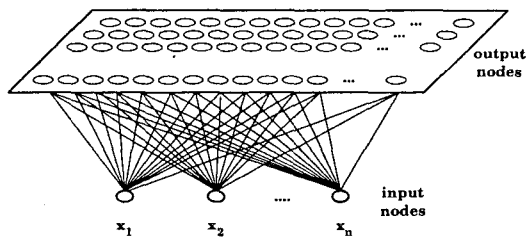


〈그림 8〉 CAS의 계산 방식

여기에서는 CAS(Cellular Automata Simulation) 라이브러리를 사용하여 계산을 하였고 병렬화된 CAS 라이브러리에 의해 병렬 수행을 하였다. 이 CAS의 응용 분야로는 미분방정식의 해석 및 유체 역학, 열 역학과 일기예보 등에 사용할 수 있다.

● 신경회로망 학습 프로그램

SOM(Self Organizing Map)방법을 써서 주어진 룰(rule)를 바탕으로 학습을 하는 프로그램으로써 그림 9과 같은 구조를 가진다.



〈그림 9〉 SOM의 계산 방식

위 신경회로망 학습 프로그램은 통계학적 패턴 인식 특히 음성인식 분야에 적합하며, 이 외에도 최적화 문제, 로보틱스(robotics), 프로세스 제어 등에 응용 되어질 수 있다.

● TSP(Traveling Salesman Problem) 프로그램

TSP 프로그램은 여러 도시를 여행하고 돌아오는 최단 경로를 찾는 문제이며 컴퓨터를 사용하여 푸는 방법으로써는 시뮬레이티드 어닐링(simulated annealing) 방법이 대표적이다. 시뮬레이티드 어닐링 방법은 컴퓨터 알고리즘을 이용하여 문제의 크기에 따라 수행시간이 다항식으로 표현되지 않는 문제를 보다 빠른 시간에 준 최적해를 구해주는 방식으로 유명하다. 병렬화를 위해서 단계적으로 중첩된 병렬 시뮬레이티드 어닐링 방법을 사용하였다[Kim90].

● FFT(Fast Fourier Transform)

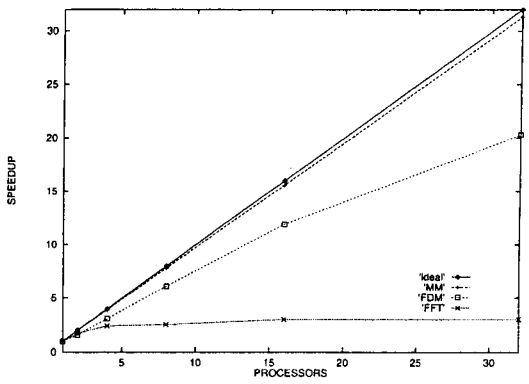
FFT는 영상처리나 각종 신호처리에 주로 쓰이는 변환방법이며 각종 신호의 주파수 대역을 볼 수 있고 영상인식 등에 응용된다. 2차원 FFT를 행과 열로 나누어 병렬화 하였다.

2. 성능 측정

성능측정을 크게 두 가지 방법으로 나누었다. 하나는 KAICUBE/한빛-1의 프로세서 수에 따른 수행시간 향상(speedup)을 비교하였고, 다른 방법으로는 KAICUBE/한빛-1 8노드와 SUN SPARC-2와의 성능을 비교하였다. 비교를 하기 위해 수행된 프로그램과 그 입력 파라메터는 다음과 같다.

- 250×250 행렬 곱셈(MM)
- 256*256 CAS mean filtering(MF)
- 400×400 신경회로망 학습(SOM)
- 40×40×20×4 반도체 열 해석(FDM)
- 512×512×10times FFT(FFT)

32개의 노드 프로세서에 대해서 프로세서의 수에 따라 성능 향상을 그림 10에 나타 내었다. 프로그램의 병렬화와 통신방식에 따라 MM과 같이 이상적에 근접한 경우가 있고, FFT와 같이 수행 향상을 별로 많이 얻지 못한 결과도 있다. 모든 프로그램에 대해서 이상적인 결과를 가져오지 못한



〈그림 10〉 프로세서 수에 대한 SPEEDUP

것은 프로그램과 데이터를 초기에 각 노드 프로세서로 다운로드(download)하는데 시간이 필요하기 때문이다. 이 시간은 비교적 수행시간이 많이 필요한 경우에 대해서는 무시할 수 있으나 수행시간이 적은 프로그램 일수록 전체 수행 시간에 대한 다운로드에 필요한 시간은 점점 큰 비율을 차지한다. 따라서 수행시간이 작은 프로그램 일수록 프로세서의 수에 대한 수행 시간 향상은 선형적(linear)에서 점점 멀어지게 된다.

MM은 나누어진 병렬화 된 연산들 사이에 종속성(dependency)이 전혀 없으므로 결과적으로 프로그램 수행 도중 통신을 발생시키지 않아 성능이 가장 이상(ideal)에 가깝게 나타났다. 반면에 FFT는 초기의 데이터와 항상 계산되어진 결과를 호스트로 전송하여야 하기 때문에 프로세서의 수가 많아 지더라도 그에 따른 성능 향상을 나타내지 못함을 알 수 있다.

그리고 일반적으로 수치 계산에 많이 쓰이는 SUN SPARC-2와 비교를 위해 각 프로그램에 대해 1절에서의 방법에 따라 병렬화를 하였고 그 결과치를 표 1에 보였다.

결과에서 알 수 있듯이 8노드에서 5~15배 정도의 성능 향상을 나타내었으며 보다 나아가 32노드에 대해서는 더욱 빠른 성능 향상이 기대된다. 이와 같은 결과를 바탕으로 KAICUBE/한빛-1은 계산 전용 서버로써의 역할을 충분히 수행할 수

〈표 1〉 KAICUBE/한빛-1 8노드와 SUN SPARC -2와의 비교

프로그램	KAICUBE/ 한빛-1	SUN SPARC -2	수행시간비
MM	2.22	29.38	13.23
MF	21	123	5.85
SOM	265	16	16.56
FDM	154	861	5.59
TSP	236	2987	12.65
FFT	1.5	5.2	3.46

있으며 각종 병렬화 알고리즘의 개발을 위한 기반 컴퓨터로 사용될 수 있다.

VI. 결론 및 추후과제

본 논문에서는 고성능 병렬처리 하이퍼큐브형 컴퓨터인 KAICUBE/한빛-1의 하드웨어 및 소프트웨어 구조와 병렬 프로그래밍 환경, 그리고 그 성능을 평가하였다. KAICUBE/한빛-1은 병렬처리 컴퓨터의 분류중 분산메모리 구조에 속하며 각각의 노드는 하이퍼큐브 연결망형태로 연결되어 있다. 하이퍼 큐브 연결망 형태는 여타 다른 연결망 형태에 비해 평균 통신거리가 짧고 확장성도 좋은 것으로 알려져 있다.

KAICUBE/한빛-1의 각 노드는 64bit 프로세서인 intel사의 i860을 사용하였으며 8M의 메인 메모리와 통신을 위한 SRAM 버퍼 및 DMA 콘트롤러로 구성되어 있다. 총 32노드로 만들어진 KAICUBE/한빛-1은 최대 2.56G-flops의 성능을 나타낸다. KAICUBE/한빛-1의 호스트 컴퓨터는 Pentium PC이며 UNIX를 기초로 하고 있다. 호스트 컴퓨터의 역할은 노드 컴퓨터들의 관리 및 사용자 프로그래밍 환경을 제공하는 것이다. 이를 위하여 Express라는 병렬 O.S.를 사용하고 사용자에게 편리한 환경을 제공하여 준다.

일반 순차적 프로그램을 쉽게 병렬화하기 위해서 KAPPA를 제공하며 각종 분석을 통해 자동적으로 KAICUBE/한빛-1의 병렬 프로그램을 생성하여 준다.

일반적으로 수치연산에 많이 쓰여지고 있는 각종 프로그램들이 성능분석을 위해 병렬화되어 수행되고 있으며 SUN SPARC-2와의 수행시간 비교를 통해 8개의 노드로 수행했을 경우 8배 정도의 속도 향상을 보였다. 따라서 일반적인 모든 수치계산 프로그램의 경우도 최대 수십배의 속도향상을 가져 올 수 있기 때문에 계산전용 서버로써의 역할을 충분히 소화해 낼 수 있다.

추후과제로는 먼저 단위 컴퓨터간의 통신 속도 개선을 들 수 있다. 현재는 메세지 전달 방식의 1세대인 저장-전달(store and forward) 방식을 사용하고 있지만 3세대 방식인 웜홀 라우터(wormhole router)방식을 채택하고 VLSI화 함으로써 보다 빠른 통신을 할 수 있다. 또한 128노드로 확장하여 다중 사용자 환경에서 충분한 노드수를 제공할 수 있어야 하며 노드 컴퓨터의 운영체제인 node kernel을 보완 개선하여야 한다.

보다 편리한 병렬 프로그래밍 환경으로써 KAPPA를 기초한 좀더 높은 차원의 환경이 구축되어야 하며 각종 수치 계산 프로그램의 병렬화된 라이브러리를 제공할 수 있어야 한다.

장기적인 목표로는 KAICUBE/한빛-1을 네트웍에 개방하여 계산전용 서버로써 다중사용자환경을 안정적으로 제공하여야 한다.

참 고 문 헌

[Choi92] jong-Heuk Choi. Implementation of parallel programming aid on hypercube multicomputer. Master's thesis, KAIST, 1992.

[Daniel85] Jih-Kwon Peir Daniel D. Gajski. Essential issues in multiprocessor systems. *IEEE Computer*, June 1985.

[Flyn72] M. Flynn. Some computer organization and their effectiveness. *IEEE Transac-*

tions on Computers, 21:948-960, Sep 1972.

[Fox88] M. Johnson G. Fox and S. Otto. *Solving Problem on ConCurrent Processos*. Prentice Hall, 1988.

[Hwang93] Kai Hwang. *Advanced Computer Architecture*. McGraw-Hill, Inc, 1993.

[Intel90a] Intel. *i860 64Bit Microprocessor Hardware Reference Manual*. Intel, 1990.

[Intel90b] Intel. *i860 64Bit Microprocessor Hardware Reference Manual*. Intel, 1990.

[Kim90] Young-Tak Kim. *Stepwise-Overlapped Parallel Annealing and its Application*. PhD thesis, KAIST, 1990.

[Laxm84] Dharma P. Agrawal Laxmi N. Bhuyan. Generalized hypercube and hyperbus structures of a computer network. *IEEE transactions on computer*, c-33 (4):13-26, Apr 1984.

[ParaSoft92] ParaSoft. *A Tutorial Introduction to Express*. ParaSoft, 1992.

[Wu90] M.Y. Wu and D.D. Gajski. Hypertool: a programming aid for messagepassing systems. *IEEE transactions on Parallel and Distributed Systems*, 21:948-960, Sep 1972.

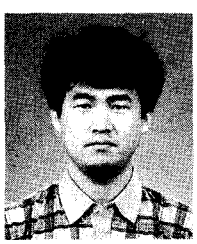
저 자 소 개

朴 圭 晧

1950年 10月 19日生
1973年 2月 서울대학교 공과대학 전자공학과 졸, 학사
1975年 8月 한국과학원 전기 및 전자공학과 졸, 석사
1983年 3月 파리 제11대학 졸, 박사

1973年 2月～1979年 동양정밀주식회사 1개발실근무
1983年 4月～현재 한국과학기술원 전기 및 전자공학과 교수

주관심 분야 : 병렬처리 컴퓨터 구조 및 운영체제

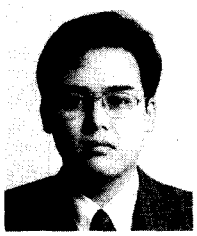


丁 鳳 浚

1966年 10月 9日生
1991年 2月 경북대학교 전자공학과 졸업(학사)
1993年 2月 한국과학기술원 전기 및 전자공학과 졸업(석사)
1995年～현재 한국과학기술원 전기 및 전자공학과 박사과정

1994年 5月～현재 "과학계산전용 고성능 병렬컴퓨터 및 전용 응용 소프트웨어 패키지 개발" 참여
1993年 5月～현재 "고성능 병렬컴퓨터의 개발" 참여

주관심분야 : 컴퓨터 구조, 병렬 처리, 컴파일러 설계



崔 鐘 赫

1968年 8月 1일生
1991年 2月 서울대학교 전기공학과 졸업(학사)
1993年 2月 한국과학기술원 전기 및 전자공학과 졸업(석사)
1995年 현재 한국과학기술원 전기 및 전자공학과 박사과정

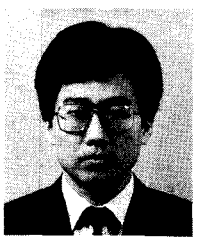
1994年 5月～현재 "과학계산전용 고성능 병렬컴퓨터 및 전용 응용 소프트웨어 패키지 개발" 참여
1993年 5月～현재 "고성능 병렬컴퓨터 개발" 참여

주관심분야 : 컴퓨터 구조, 병렬 처리, 컴파일러 설계

洪 焌 誠

1960年 3月 15일生
1991年 2月 경북대학교 전자공학과 졸업(학사)
1993年 8月 한국과학기술원 전기 및 전자공학과 졸업(석사)
1995年 현재 한국과학기술원 전기 및 전자공학과 박사과정
1994年 5月～현재 "과학계산전용 고성능 병렬컴퓨터 및 전용 응용 소프트웨어 패키지 개발" 참여
1993年 5月～현재 "고성능 병렬컴퓨터 개발" 참여

주관심분야 : 객체지향 이론, 시뮬레이션, 운영체계



沈 揆 玄

1962年 2月 24일生
1985年 2月 서울대학교 전기공학과 졸업(학사)
1985年 2月 한국과학기술원 전기 및 전자공학과 졸업(석사)
1991年 현재 한국과학기술원 전기 및 전자공학과 박사과정

1994年 5月～현재 "과학계산전용 고성능 병렬컴퓨터 및 전용 응용 소프트웨어 패키지 개발" 참여
1993年 5月～현재 "고성능 병렬컴퓨터 개발" 참여

주관심분야 : 병렬 처리, 상호 연결망 구조

李 珩 碩

1966年 1月 10일生
1988年 2月 연세대학교 전자공학과 졸업(학사)
1991年 8月 한국과학기술원 전기 및 전자공학과 졸업(석사)
1995年 현재 한국과학기술원 전기 및 전자공학과 박사과정

1994年 5月～현재 "과학계산전용 고성능 병렬컴퓨터 및 전용 응용 소프트웨어 패키지 개발" 참여
1993年 5月～현재 "고성능 병렬컴퓨터 개발" 참여

주관심분야 : 병렬 처리, 병렬/분산 운영체제

曺 常 榮

1965年 1月 23일生
1988年 2月 서울대학교 제어계측학과 졸업(학사)
1990年 2月 한국과학기술원 전기 및 전자공학과 졸업(석사)
1994年 2月 한국과학기술원 전기 및 전자공학과 졸업(박사)
1994年 3月～현재 미국 U.C. irvine대 Post Doc. 과정
1994年 5月～1994年 2月 "과학계산전용 고성능 병렬컴퓨터 및 전용 응용 소프트웨어 패키지 개발" 참여
1993年 5月～현재 "고성능 병렬컴퓨터 개발" 참여

주관심분야 : 컴퓨터 구조, 병렬 처리, 스케줄링/매핑 이론

金 鍾 旭

1964年 9月 2일生
1988年 2月 서울대학교 제어계측학과 졸업(학사)
1990年 2月 한국과학기술원 전기 및 전자공학과 졸업(석사)
1994年 2月 한국과학기술원 전기 및 전자공학과 박사과정
1994年 3月 현재 대우 통신 연구원
1994年 5月～1994年 2月 "과학계산전용 고성능 병렬컴퓨터 및 전용 응용 소프트웨어 패키지 개발" 참여
1993年 5月～현재 "고성능 병렬컴퓨터 개발" 참여

주관심분야 : 컴퓨터 구조, 병렬 처리, 상호 연결망 이론