□ 기술해설 □

병렬처리 컴퓨터의 구조 및 응용

한국과학기술원 박규호* · 정봉준** · 최종혁*** · 홍준성***

목 차

1. 서 론
2. 병렬 컴퓨터의 구조
3. KAICUBE/한빛-1의 하드웨어 구조
3.1 단위 노드의 특징
3.2 단위 노드의 통신 기능
3.3 호스트 컴퓨터
4. 단위 노드의 소프트웨어
4.1 초기화 진단 프로그램
4.2 모니터 프로그램
4.3 커널 프로그램
5. 호스트 컴퓨터의 프로그래밍 환경
5.1 익스프레스(Express)
5.2 KAPPA
6. 성능 측정을 위한 프로그램
6.1 성능 측정 프로그램
6.2 압축성 유동해석용 Euler 코드의 병렬화
6.3 PCB 내의 3차원 열/유동장 해석
7. 결론 및 추후과제

1. 서 론

컴퓨터의 발달과 더불어 처리 해야할 데이터의 수도 점차 방대해지고, 계산량 또한 기하급수적으로 늘어났다. 이와 같은 이유로 좀더 빠른 처리 속도를 가진 컴퓨터의 개발이 절실히 필요 되지만 현재의 VLSI 기술은 반도체 기술에 의한 한계를 보이고 있고, 시스템의 전체적인 고성능 화를 위해서 주변 장치의 고속화가 필수적이므로 이에 따른 추가적인 비용이 따른다. 따라서, 세계적인 연구소와 기업을 중심으로 이런 한계를 극복하기 위해 병렬 처리 컴퓨터 구조에 대한 연구가 오래 전부터 진행되어 이젠 컴퓨터의 새로운 분야로 자리잡고 있다.

초기에는 주로 CPU 내부의 계산 모듈을 벡터화하고 벡터 계산 명령어에 대해 벡터 데이터를 한꺼번에 처리하는 벡터 프로세싱 방법을 사용하여 Cray, NEC 등의 슈퍼컴퓨터가 개발되었다. 이 구조는 계산 모듈의 수에 따라 성능 증가가 있고 벡터 연산에 대해서는 고속 처리가 가능하지만, 비 벡터 연산에 대해서는 성능 향상을 얻을 수 없으며 슈퍼 파이프라인 방법과 같이 여러 개의 벡터 모듈에 의한 CPU의 크기가 커지므로 제작비용이 아주 비싸게 되는 단점이 있다[1].

보다 저렴한 가격에 고성능을 얻기 위하여 일반 CPU 를 사용하는 병렬 처리 컴퓨터에 대한 연구가 진행되었으며 여기에는 다수의 독립된 CPU 를 사용하고 각 CPU 간에 공유메모리를 가지는 공유 메모리 다중 프로세서(Shared-Memory Multiprocessor)와 CPU, 메모리가 하나의 모듈을 이루고 이들 사이에 연결 망을 가진 분산 메모리 다중 컴퓨터(Distributed-Memory Multicomputer) 방식이 있다. 이 구조들은 종래의 단일 CPU를 사용한 컴퓨터에 비하여, 대량의 계산 문제를 분할하여 병렬처리 함으로써 큰 성능향상을 기대 할 수 있고, 여분(redundancy) 개념을 이용하여 고장 감

*종신회원
**비 회 원
***학생회원

내(fault-tolerant) 시스템의 구현이 가능하며, 확장성 또한 좋은 장점이 있다[2][13]. 분산 메모리형 병렬 컴퓨터는 분산된 노드들 끼리 메시지를 통하여 데이터를 전달하여야 하므로 병렬 프로그래밍이 어려운 반면 공유메모리 구조는 각 CPU가 동일한 메모리를 접근하게 되므로 비교적 프로그래밍이 쉬운 장점이 있다. 반면, 공유메모리형의 병렬 컴퓨터는 다수의 프로세서가 동일한 버스(Bus)를 통하여 메모리를 접근 하므로 이 버스에 부하가 집중되고, 따라서 확장성이 비교적 좋지 않는 단점이 있다. 이 두 가지 방식의 장점을 취한 구조가 DSM(Distributed Shared Memory, 분산 공유 메모리) 구조이다.

현재의 병렬처리 컴퓨터 개발의 추세는 크게 고속의 통신채널과 다수의 CPU를 사용한 대규모 병렬처리구조인 MPP(Massively Parallel Processing)와 Stanford 의 DASH[4] 를 기점으로 등장한 DSM 구조로 나뉘어 진다. 전자의 경우는 이미 기술 축적이 많이 이루어져 Paragon, T3D, Computing Surface 등의 상용 MPP 가 개발 되었다. 후자의 경우는 Stanford 대학에서 DASH가 제안되어 관련된 연구가 활발히 진행 중이다. 특히 이 구조는 하드웨어 구성을 분산 메모리 형태를 취하고 있으나 특별한 메모리 콘트롤러와 통신 콘트롤러에 의해 사용자에게는 공유메모리 처럼 보이게 해주는 방식으로써, 분산 메모리 방식의 확장성과 공유메모리 방식의 쉬운 프로그래밍 환경등 두 방식의 장점을 절충하고 있다.

본 논문의 2장에서는 이러한 차세대형 병렬 컴퓨터 구조인 DSM의 구조와 동작방식에 대해 설명하고, 3장부터 하이퍼큐브 형태의 연결망을 가진 메시지-전달형 다중 컴퓨터 KAICUBE/한빛-1에 대해 설명한다. 3장에서는 이 컴퓨터의 하드웨어적인 구조를, 4장에서는 단위 노드 컴퓨터의 소프트웨어 구조를 각각 설명한다. 5장에서는 KAICUBE/한빛-1의 프로그래밍 환경에 대해 설명하고, 6장에서는 본격적인 과학계산 응용 프로그램인 유체 해석 프로그램과 열 분포해석 프로그램을 병렬화하고 그 성능을 보였다. 7장에서는 결론과 추후 과제에 대해 설명한다.

2. 병렬 컴퓨터의 구조

Flynn 은 명령어 스트림과 데이터 스트림의 중복성에 따라 컴퓨터를 다음과 같은 4 가지 범주로 분류하였다[5]. 그림 1에 이들의 구조를 나타내었다.

① SISD(Single Instruction stream, Single Data stream) : 단일 프로세서

② SIMD(Single Instruction stream, Multiple Data stream) : N 개의 노드(프로세싱 유닛+메모리)가 연결망으로 콘트롤 프로세서에 연결되어 있다. 콘트롤 프로세서에는 명령어 메모리가 있고, 여기에서 명령어를 읽어와서 이를 N 개의 노드에게 전달하고, 각각의 노드는 각각의 데이터에 대해 전달받은 명령을 수행한다. 조건에 따라서 명령어를 수행하지 않고, NOP을 수행 할 수도 있다.

③ MISD(Multiple Instruction stream, Single Data stream) : 각각의 노드에 콘트롤 유닛이 있고, 모든 노드가 하나의 메모리를 공유한다. 같은 데이터 스트림에 대해 동시에 여러 가지 연산을 한다.

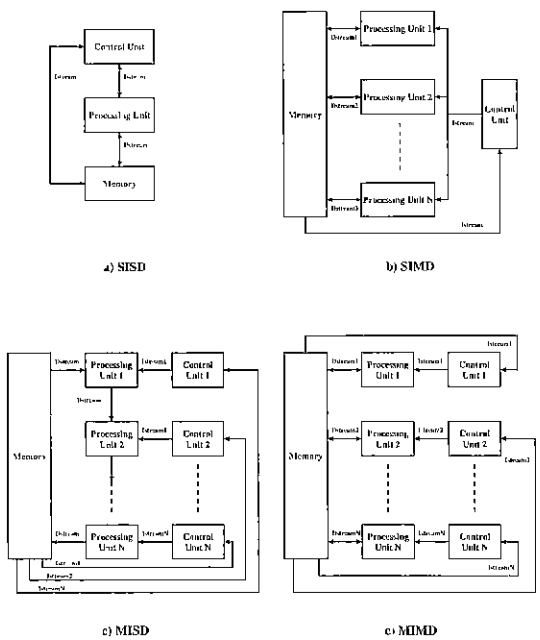

그림 1 Flynn 의 컴퓨터 분류

Systolic array 가 여기에 속한다.

④ MIMD(Multiple Instruction stream, Multiple Data stream) : 각 노드가 자신의 명령어를 읽어와서 자신의 데이터를 가지고 연산한다.

이들 중 SISD를 제외하고는 모두 병렬 컴퓨터이다. SIMD와 MISD는 특정한 계산들에서는 좋은 성능을 나타내지만, 일반적인 많은 프로그램들은 이들에서 효율적으로 수행되기 힘들다. 게다가, SIMD와 MISD는 전용 프로세서를 이용해야 하며, 따라서 SIMD와 MISD는 가격 대 성능 비가 좋지 않다. 반면에, MIMD는 범용 프로그램들이 효율적으로 수행될 수 있고, 다중 사용자/다중 프로그래밍 환경을 구축할 수 있으며, 시중의 마이크로프로세서를 사용할 수 있기 때문에 가격 대 성능 비가 좋다. 이러한 장점 때문에 최근에는 MIMD 형의 병렬 컴퓨터가 많이 제작되고 있다. Thinking Machine 사가 CM-1/2 에서는 SIMD 구조를 택했었으나 CM-5 에서 MIMD 구조를 취한 것이 그 일례라 할 수 있다[6].

기존의 MIMD 병렬 컴퓨터는 프로세서 수에 따라서 두 부류로 나눌 수 있다. 결국, 프로세서 수는 메모리 구성과 연결망 구조를 결정한다. 첫 번째 그룹은 중앙 집중형 공유 메모리 다중 프로세서이다. 프로세서 수가 작은 시스템에서는 공유 버스를 이용하여 프로세서들과 공유 메모리를 연결한다. 큰 캐쉬를 프로세서 밑에 두어서 메모리와 버스가 병목이 되는 것을 막는다. 각 프로세서가 메모리를 접근하는데 같은 시간이 걸리므로 UMA(Uniform Memory Access) 시스템이라고도 불린다. 최근에는 서버, 워크스테이션 그리고 심지어 PC 에서도 UMA 방식의 병렬 컴퓨터가 널리 사용되고 있다. 두 번째 그룹은 분산 메모리 다중 컴퓨터이다. 메모리를 프로세서와 함께 분산시켜서 많은 프로세서를 연결하였을 때, 메모리 시스템이 병목이 되지 않게 한다. 프로세서 간은 고속의 연결망으로 연결한다. 지역 메모리에로의 접근이 많다면 요구되는 메모리 대역폭이 UMA에서보다 훨씬 작아질 수 있고, 지역 메모리에로의 접근은 항상 빠르게 유지할 수 있다.

UMA 시스템에서 프로세서간의 통신은 공유 메모리를 통한 쓰기(write)와 읽기(read) 동작을 통하여 이루어 진다. 다중 컴퓨터에서는 메모리가 한 프로세서에만 지역적으로 부착되어 있고, 이들 단위컴퓨터가 연결망으로 서로 연결되어 있다. 따라서, 프로세서 간의 명시적인 메시지 전달을 통하여만 서로 통신할 수 있다. 따라서, 다중 컴퓨터에서는 다른 프로세서의 메모리를 직접 접근하지 않기 때문에, 다중 컴퓨터를 NORMA(NO Remote Memory Access) 시스템이라고도 부른다. KAICUBE/한빛-1호도 분산 메모리 다중 컴퓨터에 속하는 병렬 컴퓨터이다.

UMA에서는 공유 주소 영역이 제공되므로 프로그램이 간단하고, 범용적이고, 효율적으로 작성될 수 있고, 이식성이 좋은 반면에, 연결망과 공유 메모리에서의 접근 충돌로 인해 시스템의 확장성이 좋지 못하다. Omega Network 같은 간접 네트웍을 사용한 UMA 시스템은 버스를 사용한 시스템 보다는 많은 수의 프로세서를 연결할 수 있지만, 이러한 시스템에서도 프로세서 수가 증가하면 스위치의 지연이 누적되어 접근 지연이 커지게 된다. 이와는 반대로, 다중 컴퓨터에서는 명시적인 메시지 전달을 해야 하므로 프로그램을 작성하기가 어렵지만,

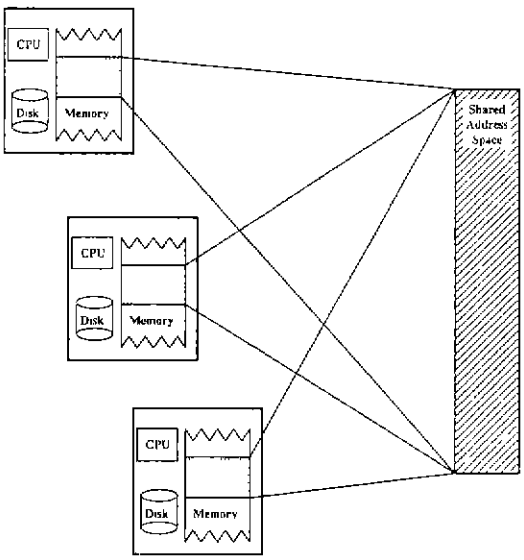

그림 2 공유 메모리의 추상화

일반적으로, 시스템 성능을 높일 수 있으며, cost-effective 하고, 확장성이 좋다.

분산 공유 메모리(Distributed Shared Memory; DSM) 다중 프로세서는 최근에 많은 연구가 되고 있는 새로운 MIMD 형의 병렬 컴퓨터이다. 그림 2 에서 볼 수 있듯이, DSM은 분산 메모리 다중 컴퓨터 구조를 가지면서 공유 메모리를 추상화 해 주어서, 물리적으로는 분산된 메모리를 논리적으로 공유되게 한다. 물리적으로 메모리가 분산되어 있으므로, 메모리의 위치에 따라 접근 지연이 다르기 때문에 DSM을 NUMA(Non Uniform Memory Access) 시스템이라고 부른다. DSM은 다중 프로세서의 장점과 다중 컴퓨터의 장점을 모두 가지고 있어서, 효율적인 프로그램을 작성하기 쉽고, 시스템의 확장성 또한 좋다. DSM의 또 다른 장점은 기존의 UMA 시스템이 워크스테이션이나 서버의 형태로 매우 널리 쓰이게 됨에 따라, UMA에서 이용되는 프로그래밍 환경을 그대로 사용할 수 있다는 것이다. 따라서, DSM은 대규모의 고성능 다중 프로세서로서 가장 주목받고 있는 컴퓨터 구조 중의 하나이다. DSM을 분류하면 크게 다음과 같은 4가지의 부류로 생각할 수 있다.

① Directory를 기반으로 하는 시스템 : 이 부류에 속하는 시스템들은 directory 라고 불리는 공유 상태에 대한 정보를 가지고 캐쉬 일관성 유지를 하드웨어적으로 지원한다. 상호 연결망은 비교적 flat 한 구조를 가지고 있다. 이 부류에 속하는 시스템들로는 Stanford의 DASH, MIT의 Alewife, Sun의 S3.mp, IEEE standard 인 SCI(Scalable Coherent Interface), Convex Exemplar 등이 있다.

② 계층적 시스템 : 계층적 시스템은 두 가지 점에서 위의 directory 시스템과 다르다. 첫째로, 일관성 유지 프로토콜은 트리 구조를 이용하여 서로간의 통신을 하는데, 트리가 재귀적인 구조를 가지기 때문에 개념적으로 간단하다. 그다음으로, 시스템의 확장이 시스템에 계층 레이어를 더 추가함으로써 쉽게 이루어 질 수 있다. 계층적 시스템은 이 두가지 장점과 메모리 접근의 순서가 자연적으로 이루어진다는 장점, 그리고 request merging과 같은 방법을 쓸 수 있다는 장점이 있다. 반면에, 계층이 많아짐으로써 메모리 접근 지연이 늘어나고, 계층 구조의 위쪽으로 올라갈수록 병목 현상이 많다는 단점이 있다. 이 부류에 속하는 시스템은 Encore의 GigaMax[7], Stanford의 Para-DiGM, COMA(Cache Only Memory Access) 시스템에 속하는 Sweden 컴퓨터 과학 연구소의 DDM [8]과 KSR 의 KSR1/2 등이 있다.

③ Reflective 메모리 시스템 : Reflective 메모리 시스템은 접근된 공유 메모리를 접근한 노드의 지역 메모리에 복사해 놓고, read miss 시에 이 지역 메모리에서 데이터를 가져가게 함으로써 접근 지연을 줄인다. write 시에는 모든 복사본들을 update 시킨다. 공유 메모리를 복사하는 것은 소프트웨어에 의해서 페이지 단위로 이루어 지며, update는 아주 작은 단위로 이루어 진다. 이 부류에 속하는 시스템은 CMU 의 Plus[9], NYU의 Merlin /Sesame 등이 있다.

④ 가상 공유 메모리 시스템 : 가상 공유 메모리 시스템은 네트웍 환경이나 메시지 전달 환경에서 공유 메모리가 있는 것 처럼 만들어 준다. 페이지 폴트를 이용하여 일관성을 유지하는 소프트웨어로 들어가며, 페이지 단위에서 소프트웨어로 일관성을 유지한다. 이 시스템은 구현하는 데 필요한 하드웨어 오버헤드가 적고, 이미 존재하는 다중 컴퓨터 위에서 DSM을 구현할 수 있다. 그리고, 소프트웨어로 일관성 유지 프로토콜을 구현하므로 유연하고, 복잡한 – 따라서 성능이 좋을 수 있는 – 프로토콜을 쉽게 구현할 수 있다. 단점으로는 데이터 공유의 오버헤드가 하드웨어 시스템 보다 매우 크다는 것이다. Yale과 Princeton의 Ivy [10]/Munin/Treadmarks, MIT의 J-Machine [11], MIT와 Motorola의 *T/*T-NG가 이 부류에 속한다.

DSM 시스템은 현재 많은 연구가 활발히 이루어지고 있으며 1990년대 초반부터는 상용 시스템들도 발표되기 시작하였다. DSM 방식의 병렬 컴퓨터는 과학 계산용 슈퍼 컴퓨터로서 뿐만이 아니라, 대용량 데이터 베이스, OLTP (On Line Transaction Processing) 등의 서버로서도 활발하게 사용될 전망이다.

3. KAICUBE/한빛-1의 하드웨어 구조

KAICUBE/한빛-1은 하이퍼큐브형의 연결망 형태를 가지는 분산 메모리형 다중 컴퓨터이다. 분산 메모리 형식 병렬 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 증명되었다. 이진-n-큐브(binary-n-cube)라고 말할 수 있는 하이퍼큐브는 차원(dimension)이라는 파라메터(parameter)에 의해 특징 지워지며 이 값에 따라 노드의 수와 링크의 수가 결정된다. 몇 개의 차원에 따른 하이퍼큐브 구조를 그림 3에 보였다[4].

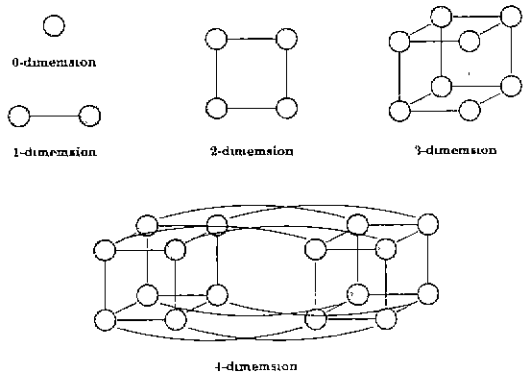

그림 3 여러 가지 차원의 하이퍼큐브 구조

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

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

메시지-전달형 다중 컴퓨터의 연결망 형태로 하이퍼큐브 형태를 사용할 경우 많은 수치 계산 프로그램의 경우 분할한 전체구조가 잘 매핑되고, 강력한 연결성을 가지며, 링, 메쉬, 트리등의 일반적인 연결구조는 모두 하이퍼큐브 구조로 쉽게 구현할 수 있다. 메시지 통신 알고리즘(algorithm)이 간단하고 복수개의 경로를 가지므로 통신능력의 측면에서 볼 때 효율적이다. 일정한 확장 방법을 가진 하이퍼큐브는 하드웨어의 제작뿐만 아니라 전체 시스템의 확장에 있어서도 쉽고 간편한 특성을 가진다. 이러한 하이퍼큐브 컴퓨터는 California 대학에서 1983년 처음으로 발표된 이래 많은 대학에서 이에 대한 연구를 진행하였으며, 기업체에서도 제품화가 되어 Intel의 iPSC 와 NCUBE 사의 NCUBE 등이 발표되었다.

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 4와 같으며 최대 2.56G-flops 의 성능을 낼 수 있다.

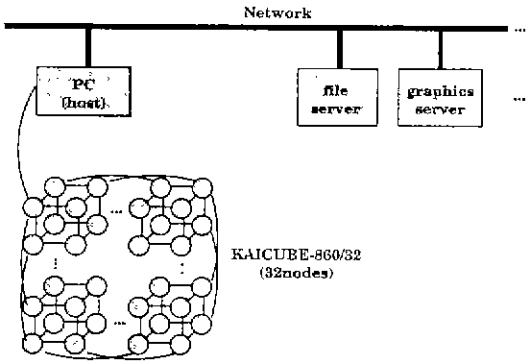

그림 4 KAICUBE/한빛-1의 시스템의 블록 다이어그램

3.1 단위 노드의 특징

단위 노드 컴퓨터는 하이퍼큐브 컴퓨터 시스

템의 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 통신속도의 성능을 극대화하였으며, 다음과 같이 주요 부분으로 구성되어 있다.

- 64비트 i860 마이크로프로세서(40 MHz)
- 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-XR 을 프로세서로 사용하여 빠른 수행 속도를 가질 수 있도록 하였다. i860은 부동 소수점 연산 장치와 캐쉬 메모리를 내장하고 40 MHz의 동작 속도에서 최대 80 M-flops의 부동소수점 연산 속도를 낼 수 있는 고성능 프로세서이다. 또한 빠른 메모리 버스 동작을 할 수 있는 설계를 위한 파이프라인 기능을 제공하고 있다[5][6]. 통신용 프로세서를 독립적으로 사용하지는 않았지만 인텔사의 i82380 고집적 주변장치(ISP)내에 포함된 DMA 콘트롤러를 사용하여 노드컴퓨터 사이의 데이터 전송을 제어하며, i860 프로세서와 독립적으로 통신을 수행하도록 함으로써 효율적인 통신을 가능하도록 하였다. DMA는 20MHz의 클럭에 동기 되며 3 클럭당 16 비트를 전송하므로 실제 채널의 통신 속도는 100Mbps 가 된다. 그림 5는 각 노드의 블록다이어그램을 보여준다.

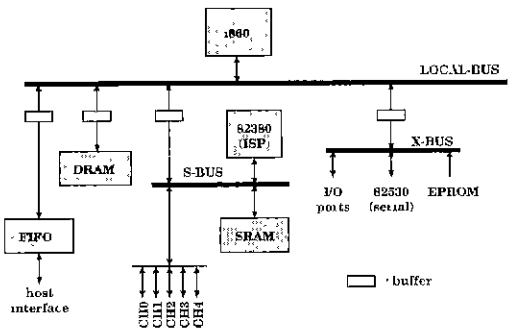

그림 5 단위 노드의 블록 다이어그램

노드 컴퓨터에 있는 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 이용하는 지의 여부와 각 디바이스가 위치한 버스에 따라서 구분 지어서 할당하였다.

— CPU 버스

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

— DRAM 버스

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

— SRAM 버스

노드 컴퓨터는 SRAM을 통신용 버퍼로서 사용하고 두 노드 컴퓨터 사이의 데이터 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 수행 시에 CPU는 다른 버스에 대한 동작을 독립적으로 수행할 수 있게 된다. 이 버스에는 SRAM 이외에도 DMA 콘트롤러가 포함된 두 개의 82380 ISP가 위치한다. 이 버스의 데이터 폭은 32 비트이다.

— 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 데이터 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 부동소수점 연산이 하나의 CPU 클럭에 이루어지기 때문에 과학 계산용으로는 적격이라 볼 수 있다.

3.2 단위 노드의 통신 기능

하이퍼큐브 컴퓨터는 전체적으로 매우 정형적인 통신 구조를 가지고 있다. 차원이 하나씩 증가할 때마다 각 단위 컴퓨터는 새로운 차원의 대응하는 단위 컴퓨터와 통신을 하기 위해서 통신 채널을 하나씩 증가 시켜야 한다. 5-차원의 하이퍼큐브컴퓨터의 경우에는 각 단위 컴퓨터는 5 개의 통신 채널을 가지고 이웃하는 컴퓨터와 통신을 하게 된다. 최대 32개의 단위 컴퓨터를 가지는 병렬 컴퓨터까지 지원할 수 있으며 더 높은 차원의 하이퍼큐브 컴퓨터로 확장할 수 있는 구조로 되어 있다. 우리가 사용한 DMA는 Intel사의 82380으로 8개의 채널을 제공하며 최대 50 M-Bytes/sec 의 데이터 전송 기능을 가지고 있다. 현재는 단위 컴퓨터에 2개의 DMA가 있는데 각각은 메시지를 보내고 받는 용도로 쓰인다. 그림 6은 채널 5개의 경우 통신 부분의 간단한 블록 다이어그램이다.

전체 통신을 개괄적으로 살펴보면 먼저 다른 단위 노드로 메시지를 보내고자 할 때는 CPU

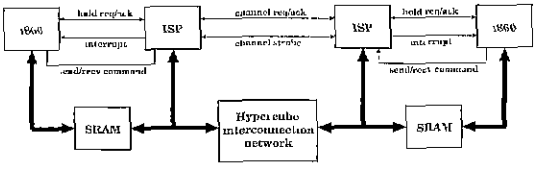

그림 6 노드간 통신의 블록 다이어그램

가 보내고자 하는 메시지를 SRAM에 저장하고 ISP를 초기화시킨 뒤에 메시지의 길이와 주소(address) 등의 정보를 ISP 내부 레지스터에 기록한다. 이러한 레지스터는 각 채널마다 하나씩 총 7개가 있으며, ISP는 이 레지스터의 내용을 바탕으로 통신 채널을 따라 대응하는 컴퓨터의 ISP에게 메시지를 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 이 것이 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 진행 할 수 있다. 그리고 통신이 끝나게 되면 각각 CPU에 인터럽트(interrupt)를 걸어 통신의 종료를 알려 주게 된다. 통신 관련 로직은 채널이 전 이중(full-duplex) 방식으로 사용되기 때문에 한 채널을 동시에 두 컴퓨터가 send용으로 사용하고자 할 때는 이 것을 조정할 수 있게 하여야 하고 송신, 수신용의 ISP가 SRAM을 동시에 사용할 수 없기 때문에 이 것에 대한 조정을 하는 기능이 필요하다. 또, 여러 채널에서 동시에 send 또는 receive 요청이 들어 올 수 있는데 이는 DMA 자체에서 제공하여 주는 우선 순위(priority) 조정 기능을 사용하여 해결하였다.

3.3 호스트 컴퓨터

KAICUBE/한빛-1에서 하이퍼큐브 구조로 연결된 노드 컴퓨터들은 계산 전용으로서 사용되고 소프트웨어의 개발과 병렬 컴퓨터의 사용에 대한 관리, 그리고 네트워크기능과 저장 장치 등의 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 사용자

에게 적절한 단위의 작은 하이퍼큐브 시스템을 할당 할 수 있어야 한다. 그리고 모든 하이퍼큐브 시스템의 접근(access)은 호스트 컴퓨터를 통해서 수행되기 때문에 여러 사용자의 메시지가 자신의 할당받은 단위에게로 정확히 전달될 수 있도록 메시지 조정기능을 가지고 있어야 하며 메시지의 전달을 효율적으로 처리할 수 있어야 한다. 이 기능들은 Express 라는 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-노드간에 필요한 통신의 수가 노드간보다 월등히 많다는 것이 알려져 있다. 그런데 호스트-노드간에도 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 7에 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 통신 대역 폭은 640Mbps 정도에 이른다. 노드간의 통신처럼 호스트도 가상 노드 번호를 가지고 모든 노드와 직접 통신을 할 수 있도록 설계되었다.

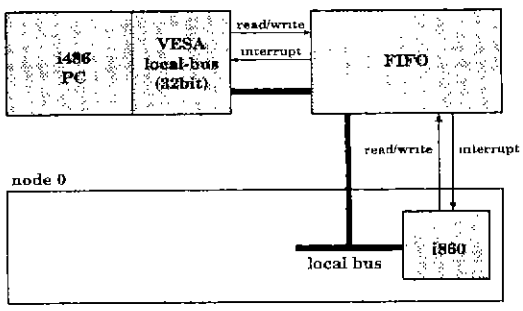

그림 7 호스트-노드간 통신 블록 다이어그램

4. 단위 노드의 소프트웨어

4.1 초기화 진단 프로그램

상위 레벨의 소프트웨어가 실행되기 전에 각종 하드웨어의 초기화(Hardware Initialization), Diagnostic(메모리, I/O, 통신 등의 테스트에 필요한 복잡한 초기화 및 테스트) 등의 기능을 수행하는 프로그램을 초기화 진단 프로그램이라 한다. 이 프로그램은 모니터나 커널을 메모리에 로딩하여 실행하는 부트스트레핑(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 초기화 하기 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 사용 할 롬(ROM)을 8비트 포트로 구성하여 하드웨어 부담(overhead)을 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 수행하기 전 미리 진단하여 수행 시 발생할 수 있는 하드웨어 적인 오류를 사전에 예방 할 수 있다. 마지막 과정으로 ROM의 내용 중 보다 상위 레벨의 모니터 혹은 커널 프로그램을 64비트 램에 옮기고 64비트 버스 모드로 스위칭 하는 작업을 마지막으로 초기화 진단 프로그램을 수행을 모니터로 넘긴다.

4.2 모니터 프로그램

모니터 프로그램은 하드웨어의 테스트와 소

프트웨어의 개발을 도와주는 목적을 가진다. 앞 절에서의 기초적인 초기화 진단 프로그램의 간단한 테스트보다 더 나아가 모니터 레벨에서는 사용자의 의도대로 보다 복잡하게 테스트해 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 클락 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 넘겨주는데, 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 주게된다. 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 호스트-노드간의 FIFO와 노드-노드간의 DMA 저 수준 IO 루틴을 사용하여 실제 수행될 노드 프로세서로 다운로드 한다. 특이할 만한 점은 실제 각 노드의 번호는 하드웨어 적으로 정해지는 것이 아니라 프로그램의 다운로드시 정해지게 되며 각 노드는 이 때 정해진 번호를 가지고 서로 통신을 하게 된다.

4.3 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 하게된다. CPU 관리라 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.

5. 호스트 컴퓨터의 프로그래밍 환경

5.1 익스프레스(Express)

익스프레스는 Parasoft 에서 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[7]. 공유-메모리 구조에서는 복잡한 잠금(locking) 기능과 세마포어(semaphore) 기능

을 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, 저 수준의 통신 라이브러리, 병렬 디버깅과 같은 다양한 프로그래밍 환경을 제공함으로써 프로그래머가 쉽게 접근하는 방법을 제공한다. 위의 방법 외에도 사용하는 병렬 컴퓨터의 자세한 하드웨어 정보를 숨기고 일률적인 프로그래밍 모델을 제공하는 것이 필요하다. 그림 8은 익스프레스를 통하여 본 병렬 컴퓨터를 보여준다. 이 모델을 바탕으로 프로그래머는 단일 추상화된 환경 하에서 쉽게 프로그래밍을 하게 도와준다.

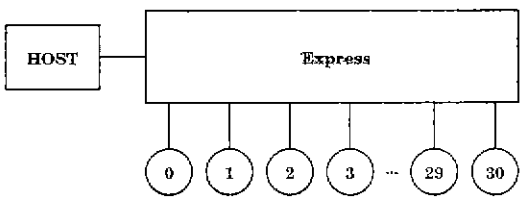

그림 8 Express를 통해 본 KAICUBE/한빛-1

익스프레스는 두개의 서로 독립적인 프로그래밍 모델을 제공하며 프로그래머는 응용 프로그램의 특성에 맞게 이 두 가지 모델 중 하나를 택하여 사용한다. Cubix 모델은 비교적 개념적으로 간단한 모델이다. 호스트에서는 cubix 라는 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 사용 할 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 병렬처리를 위한 라이브러리 외에 호스트 컴퓨터의 각종 자원을 충분히 이용할 수 있는 장점이 있는 반면 cubix 모델에 비해 호스트 프로그램을 따로 작성해야하는 단점이 있다. 일례로 노드 컴퓨터에서 I/O를 하기 위해서는 호스트로 메시지를 보내어 호스트 프로그램의 I/O 처리 루틴을 이용하여야 한다.

5.2 KAPPA

메시지 전달 방식의 많은 상업적 병렬 컴퓨터들은 단일 프로그램 다중 데이터(single program multiple data, SPMD) 형태의 프로그램 기술방식을 택하고 있다[8]. 이 방식은 여러 단위 컴퓨터가 같은 프로그램을 수행하면서 각 컴퓨터의 노드 번호에 따라 자신의 메모리에 있는 데이터에 대해 프로그램의 다른 부분이 수행을 하는 형태이다. 현재 KAICUBE/한빛-1에서도 병렬 프로그램 설계의 용의함 때문에 이 방식을 택하고 있다. 그러나 이러한 방법은 비정규적 형태의 문제에 경우 각 단위 컴퓨터 간의 부하균등을 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 트랜스레이터 또는 지능적으로 자동으로 병렬화 하는 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 스케줄 하는 일은 자동화하여 생성한다. 즉, 설계자가 적절한 알고리즘을 설계하고 분할을 수행한 뒤에 몇 개의 프로시져로 구성된 프로그램을 작성한다[9]. 이 프로그램은 순차적 프로그램 형태이기 때문에 디버깅을 위해서 순차적 머신에서 수행을 시킬 수도 있다. 이러한 프로그램은 병렬 트랜스레이터의 병렬 코드 합성과 최적화를 통해 메시지 전달 방식인 KAICUBE/한빛-1에서 수행될 수 있는 병렬 프로그램으로 바뀐다. 그림 9는 KAPPA의 구성을 보여준다.

Lexer와 parser는 주어진 프로그램으로부터 함수 단위의 데이터 연관성(data dependencies)을 찾아내고 이를 바탕으로 그래프 생성기는 스케줄링과 매핑을 위한 단 방향 그래프(directed graph)를 만들어 낸다. 여기서 각 노드는 각 병렬함수를 나타내고 그 값은 함수가 수행될 때의 수행시간을 가지게 된다. 그리고 각 에지는 각 함수간의 데이터 연관성을 의미하여 그 값은 두 함수간의 통신 량이 된다. 그 다음 스케줄링은 각각의 노드를 수행속도가 최소로 될 수 있도록 각 프로세스에 할당하는 일을 한다. 매핑은 할당된 프로세스를 통신시간을 최소화할 수 있게 각 단위 컴퓨터에 할당하는 일을 한다. 그 뒤에 동기화는 올바른 수행을 위해 통신 프리미티브(primitive)를 삽입하는 일을 하고 최종적으로 코드 생성기가 병렬 C 프로그램을 만들어낸다.

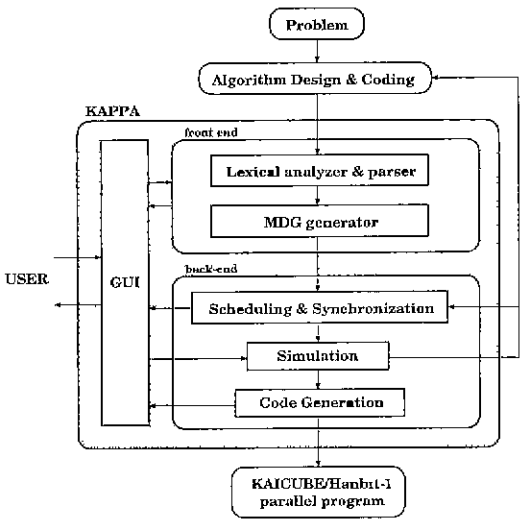

그림 9 KAPPA의 블록 다이어그램

6. 성능 측정을 위한 프로그램

6.1 성능 측정 프로그램

성능 측정을 위한 예제로써 각종 간단한 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[10].

— 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[10]. 병렬화를 위하여 전체 반복 계산 영역을 프로세서의 개수만큼 분할 한 다음 각 프로세서끼리 파이프라인 처리하여 병렬화 하였다.

— Mean 필터링(filtering)

화상처리에 쓰이는 필터링방법의 일종이며 이웃한 점들과의 평균값이 현재 점의 값이 되는 계산방식을 가진다. 여기에서는 CAS(Cellular Automata Simulation) 라이브러리를 사용하여 계산을 하였고 병렬 화된 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)을 바탕으로 학습을 하는 프로그램이다. 신경회로망 학습 프로그램은 통계학적 패턴인식 특히 음성인식 분야에 적합하며, 이외에도 최적화 문제, 로보틱스(robotics), 프로세스 제어 등에 응용되어질 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 최적 해를 구해주는 방식으로 유명하다. 병렬화를 위해서 단계적으로 중첩된 병렬 시뮬레이티드 어닐링 방법을 사용하였다[11].

— 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.

위 예제들의 성능측정을 크게 두 가지 방법으로 나누었다. 하나는 KAICUBE/한빛-1의 프로세서수에 따른 수행시간 향상(speedup)을 비교하였고, 다른 방법으로는 KAICUBE/한빛-1 8노드와 SUN SPARC-2 와의 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 나타내었다. 프로그램의 병렬화와 통신방식에 따라 MM과 같이 이상 적에 근접한 경우가 있고, FFT와 같이 수행 향상을 별로 많이 얻지 못한 결과도 있다. 모든 프로그램에 대해서 이상적인 결과를 가져오지 못한 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 일 수록 전체 수행 시간에 대한 다운로드에 필요한 시간은 점점 큰 비율을 차지한다. 따라서 수행시간이 작은 프로그램일수록 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 많아지더라도 그에 따른 성능 향상을 나타내지 못함을 알 수 있다.

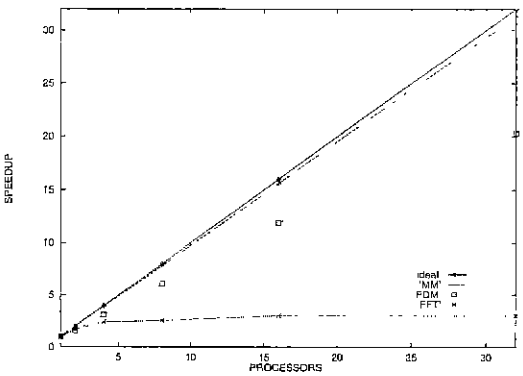

그림 10 프로세서 수에 대한 SPEEDUP

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

프로그램	KAICUBE/한빛-1	SUN SPARC-2	수행시간비
MM	2.22	29.83	13.23
MF	21	123	5.85
SOM	16	265	16.56
FDM	154	861	5.59
TSP	236	2987	12.65
FFT	1.5	5.2	3.46

그리고 일반적으로 수치 계산에 많이 쓰이는 SUN SPARC-2와 비교를 위해 각 프로그램에 대해 1절에서의 방법에 따라 병렬화를 하였고 그 결과치를 표 1에 보였다. 8노드에서 대체로 SUN SPARC-2의 5-15배정도 빠른 속도를 보여 주고 있다.

6.2 압축성 유동해석용 Euler 코드의 병렬화

Euler방정식을 수치적으로 해석하기 위해서는 시간이산화(time discretization)와 공간이산화(space discretization)과정을 통하여 비선형 연립 편미분 방정식을 연립 대수 방정식으로 변환시켜 해를 구하게된다. 여기서 시간 이산화와 공간 이산화를 어떻게 처리하는냐에 따라 여러 수치해법들이 사용된다. 이번 연구에서는 외재적 방법(Explicit scheme)과 내재적 방법(Implicit scheme)의 대표적인 Runge-Kutta Time Stepping 기법[20]과 LU-SGS기법[21]을 유한체적에 적용하여 3차원 Euler방정식을 계산하였다.

ONERA M6날개(그림 11)는 앞전의 제침각이 30도이고 가로 세로비가 3.8, 테이퍼비가 0.562인 형상을 가지고 있으면서도 비교적 두꺼운 날개이다. 이번 연구에서는 자유류의 마하수가 0.84이고 받음각이 3.06도인 유동을 해석하였다. 병렬처리기법을 전산유체역학에 적용하기 위한 방법은 영역 분할법 병렬 방정식 해법, 컴파일러를 이용한 재구성 방법등 여러

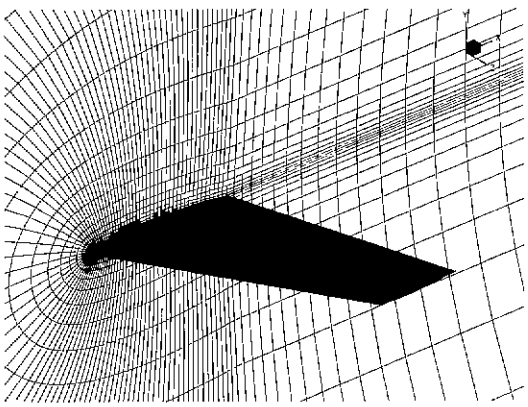

그림 11 ONERA M6날개의 격자계

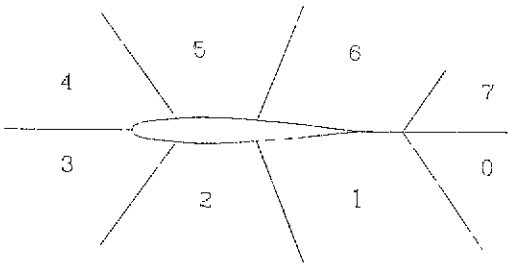

그림 12 영역분할

가지가 있다. 이 중에서 가장 많이 쓰이는 방법은 주어진 계산 영역을 적절한 크기로 분할하여 각각의 CPU에 할당을 하는 영역분할법이다. 이 방법은 기존의 단일처리프로그램을 큰 변화없이 병렬처리가 가능하도록 할 수 있고, 정렬 격자계의 경우 영역분할이 쉬운 장점이 있다.

영역분할방법은 그림 12와 같다. 그림의 표시된 번호는 domain number인데, exgrid routine에 의해 자동적으로 영역간의 거리를 최소화하는 노드로 배치된다. 격자의 분포가 159x20x40이므로 격자의 갯수가 가장 많은 곳을 분할하는 것이 통신의 양을 줄일 수 있고, 또한 노드의 갯수가 늘어날 경우에 코드의 변경이 용이하고 load balancing에도 나은 결과를 얻을 수 있다. ONERA M6주변의 격자는 단면에 대해 이차원적인 C형 격자를 구성하고, 이 격자가 적층되어 3차원 격자를 구성한다. 외재적 방법의 경우, 각 노드의 계산 수행중에 필요한 경계 조건과 residual smoothing에 필요한 경계값들을 노드간의 통신을 통하여 주고받는다. 내재적 방법의 경우는 그전 단계의 flux의 값들과 경계 조건을 노드간의 통신을 통하여 교환한다. 프로그램의 수행은 8, 16, 32개의 노드에서 하였고, 이 때 수행한 결과를 HP712/80을 사용한 결과와 비교하였다.

표 2는 LU-SGS기법을 사용했을 때 KAICUBE/한빛 1과 HP712/80에서 걸리는 시간을 비교한 것이다. 각각의 경우 real time을 시간 비교의 대상으로 정하였고, 계산을 시작한 후 1500번을 수행하는 데 걸리는 시간을 비교하였다. Speed-up ratio는 HP712/80의 계산시간과 KAICUBE/한빛 1의 계산시간의 비를

표 2 LU-SGS를 사용한 경우 병렬화 시간

(단위 : sec)

Real time	Speed-up	Ratio
HP712/80	115433.9	
8 nodes	56693.1	2.03
16 nodes	30768.6	3.75
32 nodes	18137.5	6.36

나타낸다.

Runge-Kutta Time Stepping기법을 병렬화한 계산에서는 LU-SGS를 사용한 경우보다 낮은 speed-up을 보였다(표 3). 이렇게 Runge-Kutta Time Stepping 이 LU-SGS에 비해 낮은 speed-up을 보이는 이유는 Runge-Kutta Time Stepping이 3단계의 시간적분과정에서 경계조건을 주변 노드와 교환하기 때문에 LU-SGS보다 통신에 걸리는 시간이 많기 때문이다.

표 3 Runge-Kutta Time Stepping을 사용한 경우 병렬화 시간

(단위 : sec)

Real time	Speed-up	Ratio
HP712/80	93712.9	
8 nodes	54633.4	1.72
16 nodes	30011.6	3.12
32 nodes	18118.7	5.17

그림 13은 3차원 날개주위의 압력분포를 보여준다. 이 그림은 32노드에서 LU-SGS방법으로 실행한 결과를 가시화한 것인데 ONERA-M6의 특징적인 람다충격파(lambda shock)을 잘 보여 주고 있다. 스팬방향의 44%에서의 압력계수를 실험치와 LU-SGS적분기법으로 KAICUBE/한빛 1에서 계산한 결과를 비교한 결과 계산치는 실험치와 근접한 결과를 내었으며, 32노드에서 계산한 값은 HP712/80에서 계산한 값과 일치하였다. Runge-Kutta Time Stepping을 사용하여 계산한 결과와 실험치를 비교한 경우에도 KAICUBE/한빛 1에서 계산한 결과들은 HP712/80과 일치하였다. 두 방법 모두 순차적 계산과 차이를 보이고 있는데, 이는 LU-SGS방법을 KAICUBE/한빛 1에서 병렬화하게 되면 각 영역의 경계에서는 scalar inversion을 할 때 한 단계 전의 값을 사용하기 때문이다. 또한 격자를 분할하였기 때문에 spectral radius의 값이 각 영역마다 달라져 수렴곡선의 모양이 순차적 계산의 수렴곡선과 달라진다. Runge-Kutta Time Stepping에서는 수렴을 증진시키기 위하여 Residual Smoothing을 사용하게 되는데, 이때 한단계 전의 값을 사용하기 때문에 수렴 곡선에 영향을 주는 것으로 보인다.

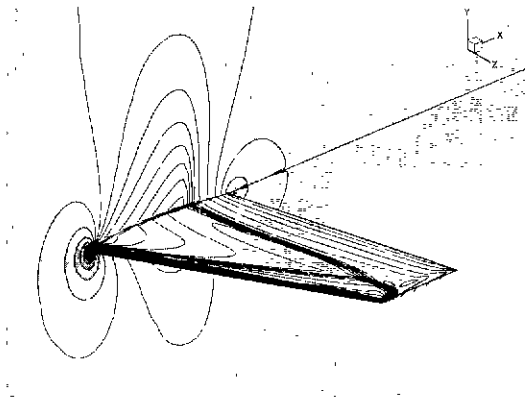

그림 13 ONERA M6 날개 주변의 압력 분포

6.3 PCB 내의 3차원 열/유동장 해석

PCB 기판위에 4각 발열블럭들이 균일하게 분포되어 있을 때 유동은 각 블럭 간에서 주기적이기는 하나 3차원성을 나타내며 또 발열체 단위체적당 발생열이 커짐에 따라 큰 온도차로 인한 부력의 증가는 혼합대류에 의한 열전달을 크게 만든다(그림 14). 이와 같은 PCB내의 유동장과 온도장을 해석하기 위해 SIMPLER 알고리즘[22]을 사용하여 프로그램을 구성 수행하였는데, 3차원 연속 방정식, Navier-Stokes방정식 그리고 에너지 방정식을 3차원 영역에서 해석하였다.

병렬처리를 위해 영역 분할법을 사용하였는데, 영역 구분을 x 방향으로만 분할하여 node의 갯수를 변화시키며 계산을 수행하였다.

계산에 사용된 격자의 크기는 $21\times21\times60$ (x, y, z) 이며 정확한 해를 얻기 위하여 블럭 주

위에 격자를 밀집시켜 계산을 수행하였다. KAICUBE/한빛 1의 성능을 알아보기 위하여 단일 처리 프로그램을 HP 735 WorkStation (45.5 MFLOPS)과 Cray Y-MP에서 계산 하여 비교하였다. 각 컴퓨터에서 행하여진 모든 계산은 처음 1000번의 계산을 수행하는데 걸린 시간을 기준으로 나타내었다.

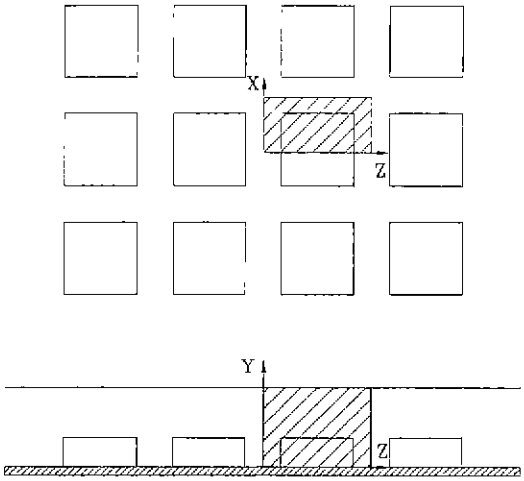

그림 14 평평한 parallel-plate channel의 벽을 따라 전개되는 rectangular module arrays

표 4 PCB내의 3차원 열/유동장 해석 수행 시간의 비교

사용 컴퓨터	Time(초)	Speed-up ratio
HP735 (45.5MFLOPS)	23238	1.000
4 nodes	9405	2.471
8 nodes	4466	5.203
16 nodes	2796	8.311
32 nodes	2190	10.611
Cray Y-MP	2834	8.198

표 4에서 보듯이 node 갯수가 증가할수록 계산 시간은 계속 감소하지만 감소율이 갈수록 둔화되고 있음을 볼 수 있다. 즉 16개의 node를 사용했을 때까지는 무려 단일 처리 프로그램의 계산 시간의 10배 정도가 감소하였다. 이는 Cray Y-MP에서 계산 하는게 걸린 시간 2834초와 거의 비슷하다. 그러나 32개의 node들을 다 사용하였으나 speed-up ratio가 더 커지지 못한 이유는 node 갯수의 증가에 따라 각 node 사이의 통신량이 많아져서 계산 성능의 증가에 비해 통신에 의해 걸리는 시간이 커졌기 때문이다. 또한 유동의 해석으로 SIMPLER 알고리즘을 사용하였는데 알고리즘 자체가 연속적으로 풀어야 하는 성질이 강하기 때문에 단순히 영역 분할법에 의한 병렬 처리도 speed-up ratio의 증가에 걸림돌이 되고 있다.

7. 결론 및 추후과제

본 논문에서는 병렬 처리 컴퓨터의 전반적인 구조와 특히 메시지 전달형 병렬 컴퓨터의 확장성과 공유메모리형 병렬 컴퓨터의 프로그램의 용이함을 동시에 고려한 DSM 구조에 대해 고찰하고, 고성능 병렬처리 하이퍼큐브형 컴퓨터인 KAICUBE/한빛-1의 하드웨어 및 소프트웨어 구조와 병렬 프로그래밍 환경, 그리고 그 성능을 평가하였다. KAICUBE/한빛-1은 병렬처리 컴퓨터의 분류 중 분산메모리 구조에 속하며 각각의 노드는 하이퍼큐브 연결망형태로 연결되어 있다. 하이퍼큐브 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 32 노드로 만들어진 KAICUBE/한빛-1은 최대 2.56G-flops 의 성능을 나타낸다. KAICUBE/한빛-1의 호스트 컴퓨터는 Pentium PC 이며 UNIX 를 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 5-10배 정도의 속도 향상을 보였다. 따라서 일반적인 모든 수치 계산 프로그램의 경우도 최대 수십 배의 속도 향상을 가져 올 수 있기 때문에 계산전용 서버로써의 역할을 충분히 소화해 낼 수 있다.

추후과제로는 먼저 단위 컴퓨터간의 통신 속도 개선을 들 수 있다. 현재는 메시지 전달 방식의 1세대인 저장-전달(store and forward) 방식을 사용하고 있지만 3세대 방식인 웜홀 라우터(wormhole router) 방식을 채택하고 VLSI화함으로써 보다 빠른 통신을 할 수 있다. 또한 128노드로 확장하여 다중 사용자 환경에서 충분한 노드 수를 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 병렬 화된 라이브러리를 제공할 수 있어야 한다. 장기적인 목표로는 KAICUBE/한빛-1을 네트워크에 개방하여 계산전용 서버로써 다중사용자환경을 안정적으로 제공하여야 한다.

참고문헌

[1] Kai Hwang, "Advanced Computer Architecture," McGraw Hill Inc, 1993.

[2] M. Flynn, "Some Computer Organization and their Effectiveness," IEEE transactions on Computer, Vol 21, pp. 948-960, Sep, 1972.

[3] Jih-Kwon Peir, Daniel D. Gajski, "-Essential Issues in Multiprocessor Systems," IEEE Computer, June, 1985.

[4] D. Lenoski, J. Laudon, K. Gharachorloo, A. Gupta, and J. Hennessy, "The directory-based cache coherence protocol for the DASH multiprocessor,", IEEE Computer, vol. 25, no. 3, pp. 63-79, Mar., 1992.

[5] M.J. Flynn, "Very high-speed computing systems," in Proc. IEEE, vol. 54, pp.1901-1909, 1966.

[6] Thinking Machines Corps., "The Connection Machine CM-5 technical summary," pp.239-509, 1991.

[7] P. Woodbury, A. Wilson, B. Shein, L. Gertner, P. Chen, J. Barttlet, and Z. Aral, "Shared memory multiprocessors : The right approach to parallel processing," in Proc. Compcon Spring 89, pp. 72-80, Feb., 1989.

[8] E. Hagersten, S. Haridi, and D. Warren, "The cache coherence protocol of the Data Diffusion Machine," in Cache and Interconnect Architectures in Mulriprocessors, ed. M. Dubois and S. Thakkar, pp. 165-188, Kluwer Academic Publishers, Norwell, MA, 1990.

[9] R. Bisiani and M. Ravishankar, "PLUS : A distrubuted shared-memory system," in Proc. 17th Int. Symp. on Computer Architecture, pp. 115-124, May, 1990.

[10] K. Li, "IVY : A shared virtual memory system for parallel computing," in Proc. 1988 Int. Conf. on Parallel Processing, pp. 94-101, Aug., 1988.

[11] W. Dally, A. Chien, S. Fiske, W. Horwat, J. Keen, M. Larivee, R. Lethin, P. Nuth, and S. Wills, "The J-Machine : A fine-grain concurrent computer," in Proc. of the IFIP Congress, pp. 1147-1153, Aug., 1989.

[12] Dhama P. Agrawal, Laxmi N. Bhuyan, "Generalized Hypercube and Hyperbus Structures of a computer Network", IEEE transactions on Computer, Vol. 33, No. 4, pp 13-26, Apl, 1984.

[13] Intel, "I860 64Bit MicroProcessor Hardware Reference Manual," Intel Manual, 1990.

[14] Intel, "I860 64Bit MicroProcessor Software Reference Manual," Intel Manual, 1990.

[15] ParaSoft, "A Tutorial Introduction to Express," ParaSoft, 1992.

[16] M. Y. Wu, and D.D. Gajski, "Hypertool : A Programming aid for Message-Passing Systems," IEEE transactions

on Parallel and Distrubuted Systems, Vol 21, pp. 948-960, Sep, 1992.

[17] 최 종혁, "하이퍼큐브 다중 컴퓨터 상에서 병렬 프로그래밍 환경의 구현," Master's Thesis, KAIST, 1992.

[18] M. Jhonson, G. Fox, and S. Otto, "Solving Problem on Concurrent Processors," Prentice Hall, 1988.

[19] Young-Tak Kim, "Stepwise-Overlapped Parallel Simulated Annealing and its Applications," Ph.D. Thesis, KAIST, 1990.

[20] A. Jameson, ""Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," AIAA Journal, Vol. 19, 1981.

[21] 윤 병택 "기판위에 균일하게 배열된 발열블럭 주위의 3차원 혼합대류 열전달 해석," 석사학위논문, 기계공학과, 한국과학기술원, 1995.

[22] S. V. Patankar, "Numerical heat transfer and fluid flow," McGraw-Hill, New York, 1980.

박 규 호

1973.2 서울대학교 공과대학 전자공학과 졸업, 학사
1975.8 한국과학기술원 전기 및 전자공학과 졸업, 석사
1983.3 파리 제 11대학 졸업, 박사
1973.2~1979 동양정밀 주식회사 1개발실 근무
1983.4 ~ 현재 한국과학기술원 전기 및 전자공학과 교수

정 봉 준

1991.2 경북대학교 공과대학 전자공학과 졸업, 학사
1993.2 한국과학기술원 전기 및 전자공학과 졸업, 석사
1993~현재 한국과학기술원 전기 및 전자공학과 박사과정
1994.5~현재 "과학계산전용 고성능 병렬 컴퓨터 및 전용 응용 소프트웨어 패키지 개발" 참여
1993.5~1996.4 "고성능 병렬 컴퓨터의 개발" 참여

최 종 혁

1991.2 서울대학교 공과대학 전자공학과 졸업, 학사
1993.2 한국과학기술원 전기 및 전자공학과 졸업, 석사
1993~현재 한국과학기술원 전기 및 전자공학과 박사과정
1994.5~현재 "과학계산전용 고성능 병렬 컴퓨터 및 전용 응용 소프트웨어 패키지 개발" 참여
1993.5~1996.4 "고성능 병렬 컴퓨터의 개발" 참여

홍 준 성

1991.2 고려대학교 공과대학 전자전산공학과 졸업, 학사
1993.9 한국과학기술원 전기 및 전자공학과 졸업, 석사
1993~현재 한국과학기술원 전기 및 전자공학과 박사과정
1994.5~현재 "과학계산전용 고성능 병렬 컴퓨터 및 전용 응용 소프트웨어 패키지 개발" 참여
1993.5~1996.4 "고성능 병렬 컴퓨터의 개발" 참여