병렬 구조해석 및 설계법 개발을 위한 PVM

권 윤 한*　　박 효 선**

1. 서　　언

컴퓨터가 출현한지 반세기 이상이 경과하였다. 초기의 1950년대 컴퓨터는 실험 단계를 거쳐, 실제 비즈니스 분야에서 실용화되게 되었고 이 단계에서는 한 대의 컴퓨터를 다수의 사용자가 공유하여 사용하였고, 데이터베이스 시스템 등의 자원이 공유되는 집중 처리형 시스템이 중심이었다. 그러나 그 뒤, 반도체 기술을 중심으로한 하드웨어 기술의 발전, 소프트웨어 기술의 진보, 통신 네트워크의 보급과 표준화에 의해 1980년대부터 컴퓨터 시스템의 이용 형태가 변화하였다. 그 중 한 가지는 근거리 네트워크 (LAN: local area network)를 중심으로한 네트워크의 보급과 표준화이고, 또한 인터넷이 등장하여 전 세계의 네트워크와 컴퓨터의 상호 접속이 가능하게 된 점이다. 다른 한가지는 다기능·고성능·소형인 워크스테이션 및 PC의 등장과 보급이다. 그 결과 동일한 컴퓨터를 2대 사용하기 보다, 한 등급 높은 컴퓨터를 1대 사용하는 편이 경제적이고 고성능이라는 그로슈(Grosch)의 법칙[1]이 붕괴하게 되었다. 결국 대형 범용 컴퓨터보다는 통신 네트워크에 의해 상호 접속된 다수의 워크스테이션이나 PC를 사용하는 편이 저렴하고, 게다가 고성능화 할 수 있게 되었다.

또한 공학분야에서도 다루어지고 있는 문제들의 대형화와 이에 따른 해석의 복잡성 그리고 설계 및 구조반응조절을 위한 해석 수의 증가 등의 요인으로 인해 고성능의 전산기가 요구되고 있으며, 이미 미국을 비롯한 유럽 등의 엔지니어링관련 연구소 및 학계에서는 다양한 형식의 고성능 컴퓨터가 상용화되어 있다. 그리고 산업계에서는 NASTRAN 및 ABACUS 등의 범용구조해석 패키지들을 고성능 컴퓨터에서 구동하여 업무의 영역 및 효율성을 높여 경쟁력을 확보하고 있다. 이러한 점을 고려하면 공학분야에 고성능 컴퓨터 활

* 학생회원·영남대학교 건축공학과, 석사과정
** 정회원·영남대학교 건축공학과 전임강사, 공학박사

용의 활성화가 시급하지만 국내 여건을 고려하면 새로운 고성능 컴퓨터를 구입하는 것 또한 적지 않은 비용이 들게 되므로 어려운 상황이다.

이를 극복하기 위한 방법 중 하나는 이미 보유하고 있는 PC나 워크스테이션을 이용하여 고성능 컴퓨터의 성능을 구현할 수 있는 대형 병렬처리 시스템을 구성하는 것이다. 이러한 병렬처리 시스템을 구성하도록 하는 소프트웨어는 인터넷상에 무료로 구할 수 있으며 기존의 하드웨어를 이용하기 때문에 더욱 경제적이다. 병렬 처리를 가능케 하는 패키지로서, 최근 PVM(Parallel Virtual Machine)[2] 소프트웨어 패키지와 MPI (Message Passing Interface)[3] 등이 있으며 윈도우 환경 (Windows95, WindowsNT)에서도 사용이 가능한 PVM－win32 버전과 WPVM, WMPI 패키지가 개발되었다. 특히 PVM은 서로 다른 운영체제의 컴퓨터들을 하나의 가상 병렬컴퓨터를 구성할 수가 있다. 예를 들면, UNIX 체제의 워크스테이션과 윈도우95 체제의 개인용 컴퓨터들을 묶어 사용하는 것이 가능하다.

병렬 계산에 대한 기본 개념과 병렬 컴퓨터의 분류 등에 관한 내용은 참고 문헌 [4, 5, 6, 7]에 자세히 설명되어 있으며 본 기사에서는 윈도우 환경의 다수의 개인용 컴퓨터를 이용하여 PVM을 통한 병렬처리 시스템을 구성하고 그 성능을 살펴보고자 한다.

2장에서는 가상 병렬 시스템을 구성하는 PVM에 대하여 설명하였고, 3장에서는 본 연구에서 사용된 병렬 시스템 모델의 설명과 이를 이용한 순수계산의 분산성능 및 메시지 전송 성능 분석, 그리고 선형 대수학에서 사용되는 벡터의 Dot Product 계산 성능을 평가하였다. 또한 실험을 통해 얻어진 분산성능의 요약과 분산 성능향상을 위한 방안을 제시하였다.

2. PVM

PVM(Parallel Virtual Machine)은 1989년 Oak Ridge National Laboratory에서 개발된 메시지 패싱(message-passing) 프로그래밍 시스템으로, TCP/IP(Transmission Control Protocol/Internet Protocol)에 기반을 둔, 네트워크로 연결된 서로 다른 운영체제를 가지는 여러 대의 컴퓨터들을 하나의 대형가상컴퓨터로 구성해주는 소프트웨어이다. PVM에서는 분할 메모리 방식의 컴퓨터에서 사용하는 메시지 패싱 방식을 지원하고 있으며 프로세서간의 통신을 지원하는 대몬 프로세서(deamon processor)인 pvmd3이라는 프로그램과 이를 통해 통신을 할 수 있도록 해주는 함수들의 라이브러리인 libpvm3.lib라는 두 부분으로 구성되어 있다. 따라서 libpvm3.lib를 이용하여 프로그램을 작성한 다음 pvmd3을 각 컴퓨터에 실행시킨 후 응용프로그램을 실행하면 각각의 함수들이 pvmd3를 통해 데이터를 교환하면서 계산을 수행하게 된다. PVM 소프트웨어에 대한 특징은 다음과 같다.

1) 사용자 환경의 가상컴퓨터 구성 : 실행하고자 하는 PVM 응용 프로그램을 사용자에 의해 선택된 머신(machine)들로 구성된 set 위에서 실행할 수 있고 이런 set은 프로그램이 실행하는 도중에도 새로운 machine을 추가하거나 제거할 수도 있다.

2) 다른 machine으로의 쉬운 접근 : PVM 응용 프로그램에서 서로 연결된 다른 machine들의 hardware 환경을 특별한 규제 없이도 접근할 수 있다.

3) process에 의한 계산 수행 : PVM에서의 병렬성의 단위는 communication과 computation을 번갈아 수행하는 하나의 독립된 연속적인 제어의 스레드(thread) 작업이다. 이러한 일련의 process에 의해 계산을 수행하게 된다. 여기서 스레드란 프로세스(process)보다 세분화된 실행의 단위를 말한다.

4) message passing model : 어떤 문제의 data를 적당히 분할하여 개개의 machine들에게 할당하여 보내고 이를 다시 받아서 조합하는 message passing model이다. 여기서 message size는 각 machine들의 memory의 크기에 좌우된다.

5) 이질성 지원(heterogeneity support) : PVM system은 서로 다른 machine, network, 운영체제,

그리고 응용프로그램과 관련된 이질성을 지원한다. message passing의 관점에서, PVM은 서로 이질의 data 표현을 갖는 machine들 사이에서 변환되어질 수 있는, 하나 이상의 datatype을 포함하는 message의 sending과 receiving이 가능하다.

6) 무제약적인 메모리 한계 : PVM에 있어서 communication 모델은 어떤 한 대의 컴퓨터에서 다른 한 대의 컴퓨터 또는 여러 대의 컴퓨터로 message를 보낼 수 있으며 그 메시지의 수나 크기에 한계가 없다. 모든 컴퓨터(host)들은 버퍼(buffer)공간을 한정하는 물리적 메모리 한계를 가지고 있지만, 이러한 communication 모델은 특별한 머신의 한계에 대한 제약이 없고 충분한 메모리를 이용할 수 있다.

또한 PVM은 전송 메시지의 순서대로 메시지가 전송되는 것을 보장한다. 만일 컴퓨터 1이 컴퓨터 2에게 A라는 메시지를 보내고 그런 후 다시 메시지 B를 컴퓨터 2에 보낸다면, 메시지 A는 메시지 B가 도착하기 전에 컴퓨터 2에 도착하게 된다. 또한 메시지 버퍼(buffer)는 동적으로 할당받게 된다. 따라서 메시지 전송에 있어서 요구되는 최대 메시지 크기는 단지 연결된 컴퓨터들의 유용 메모리의 양에 의해 좌우된다.

PVM에서의 메시지 교환은 세 가지 단계로 이루어진다. 먼저 pvmfinitsend( ) 또는 pvmfmkbuf( ) 등의 서브루틴을 사용하여 전송할 데이터를 저장할 버퍼를 만들고 초기화한다. 다음으로 전송할 메시지를 압축하여 버퍼에 넣게 되는데 여기에 사용되는 서브루틴으로는 pvmfpack( )이 있다. 마지막으로 버퍼에 저장된 데이터는 pvmfsend( ), pvmfpsend( ) 또는 pvmfmcast( ) 등의 서브루틴을 이용하여 tag를 붙여서 원하는 프로세서로 전송하게 된다. 전송되어온 데이터는 먼저 pvmfrecv( ), pvmfnrecv( ), pvmftrecv( ) 또는 pvmfprecv( ) 등의 서브루틴을 이용하여 수신된다. 그 다음에 압축을 푸는 과정인 pvmfunpack( ) 서브루틴을 이용하여 압축을 푸는 과정을 거쳐 메시지 전송을 완료하게 된다.

메시지 전송에 사용되는 Fortran 서브루틴 함수는 표 1에 잘 나타나 있다.

표 1 FORTRAN용 PVM 서브루틴 함수

분 류	서브루틴 함수
Message Buffers	call pvmfintsend(encoding, bufid) call pvmfmkbuf(encoding, bufid) call pvmffreebuf(bufid, info)
Packing Data	call pvmfpack(what, xp, nitem, stride, info)
Sending and Receiving Data	call pvmfsend(tid, msgtag, info) call pvmfmcast(ntask, tids, msgtag, info) call pvmfpsend(tid, msgtag, xp, cnt, type, info) call pvmfrecv(tid, msgtag, bufid) call pvmfnrecv(tid, msgtag, bufid) call pvmftrecv(tid, msgtag, sec, usec, bufid) call pvmfprecv(tid, msgtag, xp, cnt, type, rtid, rtag, rcnt, info)
Unpacking Data	call pvmfunpack(what, xp, nitem, stride, info)

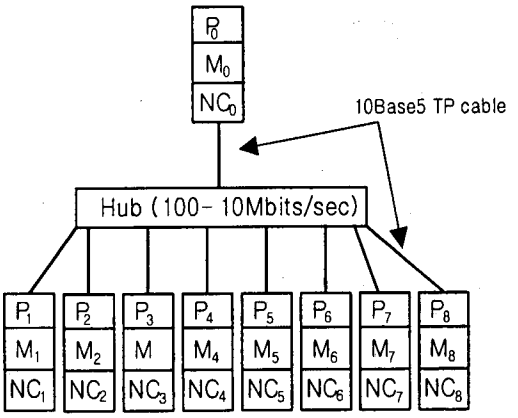
3. 병렬 연산 모델

본 기사에서는 ethernet card로 연결된 네트워크 환경 하에서 윈도우95 체제 PC의 분산형 시스템에 관한 성능을 프로세서의 수에 따른 순수 계산 성능향상(speedup)[4]과 데이터 량에 의한 메시지 전송 성능에 초점을 맞추어 평가하였으며, 이들을 바탕으로 벡터들의 dot product 연산에 적용하여 분산성능을 평가하였다.

3.1 병렬 시스템 모델

병렬 연산 모델을 적용시키기 위한 병렬 시스템 모델은 crowd 프로그래밍 패러다임[2] 중의 하나인 master-slave 모델로서 그림 1과 같이 본 연구실에서 보유하고 있는 PC를 최대 8대까지 연결하였으며 각각의 PC들의 네트워크 구성 및 각 하드웨어 사양을 나타낸다.

여기서 P_0는 master의 역할을 수행하고 $P_1 \sim P_8$은 slave 역할을 수행하게 된다.



P : 펜티엄 166MHz CPU
M : 32Mbyte 메인 메모리
NC : Ethernet 네트워크카드(10Mbps)

그림 1 병렬 시스템 모델의 하드웨어 구성도

3.2 순수 계산의 분산성능

순수 계산의 분산 성능 평가에서 master 컴퓨터는 데이터를 입력받아 분할하여 각 slave 컴퓨터들에게로 전송하는 것과 각 slave 컴퓨터들의 계산 결과를 받아 조합하여 출력하는 것을 담당하고, slave 컴퓨터들은 데이터를 master 컴퓨터로부터 전송 받아 주 연산을 담당하도록 설계하였다. 순수 계산에 적용된 모델은 10^9개의 데이터 배열을 초기화시키는 것이며, 문제 분할은 전체 데이터 량을 연결된 slave 수로 나누어 균등하게 할당하였다.

slave의 수를 1대, 2대, 4대 그리고 8대로 연결하였을 때, 순수 계산 부분만을 평가한 성능향상(speedup)의 결과는 표 2와 그림 2와 같은 결과를 얻었다. 표와 그림에 나타난 바와 같이 프로세서의 수가 증가할수록 계산성능향상 정도는 선형적으로 증가한다는 것을 알 수 있다. 이러한 원인은 본 기사에 적용된 초기화 문제와 같은 프로그램은 데이터 의존성이 거의 없기 때문에 각각의 slave 컴퓨터들이 독립적으로 연산을 수행하므로 통신시간이 거의 발생하지 않는 데에 있다.

3.3 메시지 전송 성능

메시지 전송 성능 평가는 1 대 1로 연결된 가상 병렬 환경 하에서, 데이터 량에 따른 PVM의 메시지 교환 서브루틴 함수의 각 부분별 소요 시간과 전체 메시지 전송 시간을 측정하였다. 그림 3과 4에서도 알 수 있듯이, Master 컴퓨터에서 처리되는 initsend, pack, send 부분과 Slave 컴퓨터에서 처리되는 unpack 부분에 소요되는 시간은 데이터 량의 증가에 따라 거의 큰 변화가 없는 반면, Slave 컴퓨터의 recv부분에 소요되는 시간은 데이터량이 증가함에 따라 소요 시간이 급속도로 증가한다는 것을 알 수 있다. 이로 인하여 메시지 전송의 총 소요 시간 또한 데이터의 량에 따라 증가한다.

그림 4를 보면 2^{22}byte이후부터는 급격한 경사를 이루면서 증가하는 것을 알 수 있으며 이는 가상 병렬 컴퓨터를 구성하고 있는 PC의 하드웨어 성능(main memory, 하드 전송속도 등)에 밀접한 관련이 있으며, 특히 가용 메모리 범위를 벗어난 데이터 량이 전송되면 전송을 받는 컴퓨터는 하드디스크의 가상 메모리(swap memory)

표 2 순수 계산 성능향상

연결대수	1	2	4	8
실행시간(sec)	117.0	58.3	28.9	14.4
성능향상 (Speed-up)	1.0	2.0	4.01	8.10

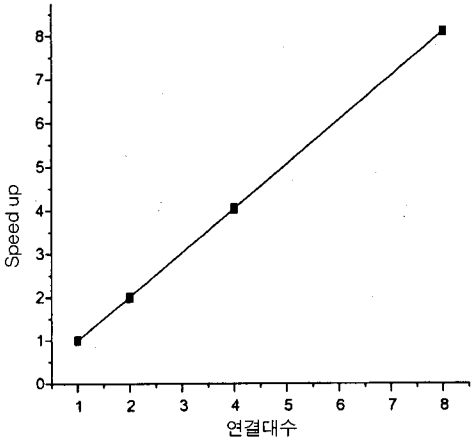


그림 2 순수 계산 성능향상(speed-up)

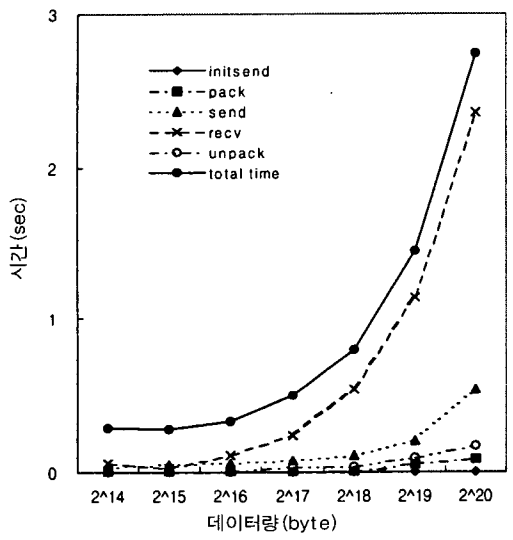


그림 3 소량의 메시지 전송 성능

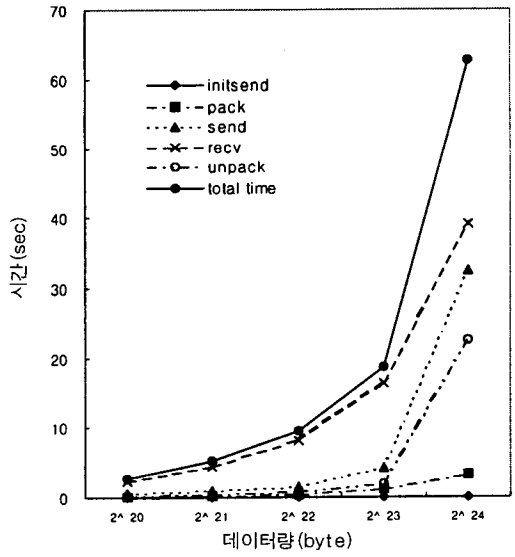


그림 4 대량의 메시지 전송 성능

를 이용하기 때문인 것으로 생각된다. 따라서 본 실험에 사용된 컴퓨터의 성능에 비추어 보면 최대 유용 전송 데이터량은 2^{22}byte(약 4Mbyte)가 된다.

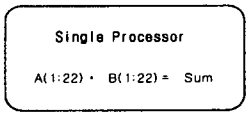
3.4 dot product 연산 모델의 분산성능

앞 절에서는 분산성능으로서 순수계산 부분만의 계산성능과 메시지 전송을 각기 따로 분류하여 측정하였으나 이 절에서는 실제 공학문제에 사용될 수 있는 dot product 모델을 사용하여 문제 규모에 따른 전체적인 분산성능을 평가하였다.

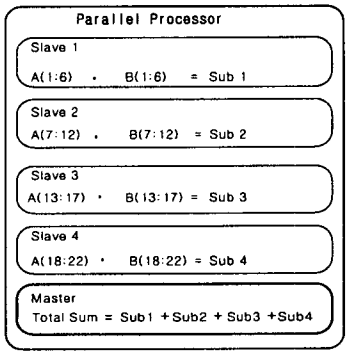
vector A · vector B의 dot product 계산에서 문제의 분할은 각각 N개의 벡터 element들을 프로세서 수 P로 나누어 각 프로세서마다 거의 동일한 양의 데이터를 전송하도록 한다. 만약, 각 벡터의 data가 22개이고 slave 컴퓨터가 4대면 S_1~S_2는 각각 6개의 데이터가, 나머지 S_3~S_4는 각각 5개의 데이터가 할당되며, 각 slave들에서 계산된 각각의 dot product 값 sub을 master가 전송 받아서 이들 합 total sum을 계산한다(그림 5).

벡터의 dot product를 계산한 결과가 그림 6과 그림 7에 나타나있다.

문제크기(problem size)가 적은 경우에는 연결대수가 많으면 많을수록 소요 시간이 증대되는 것을 볼 수 있으며 이는 소량의 문제 계산에 있



(a) 단일 프로세서 모델



(b) 분산모델

그림 5 dot product 병렬계산

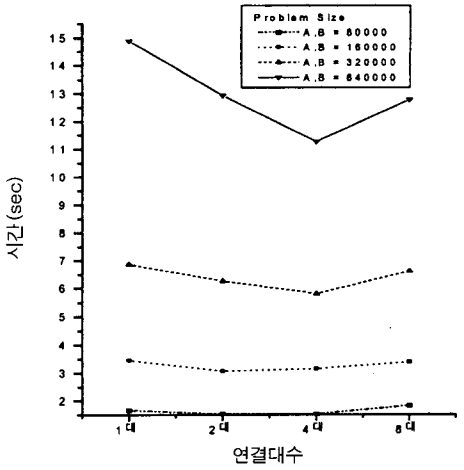


그림 6 dot product 분산 계산 성능(1)

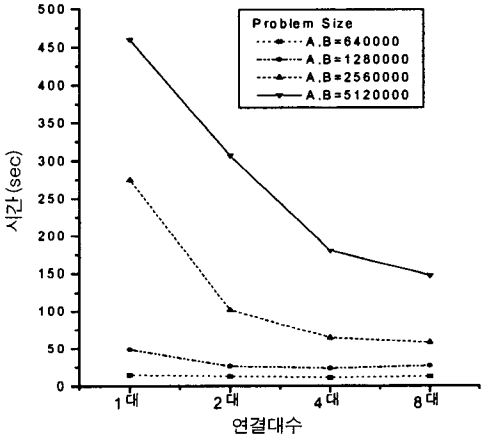


그림 7 dot product 분산 계산 성능(2)

어서 계산량 분할로 인한 성능향상에 비해 오히려 통신회수의 증대로 인한 통신시간이 더 많은 비중을 차지한다는 것을 알 수 있다. 반면 문제규모가 일정한도 이상으로 커지게 되면 오히려 연결대수가 많아질수록 계산량 분할로 인한 성능향상의 증대량이 통신시간에 걸리는 시간의 비를 초과하게 된다. 본 실험에서 사용된 모델에 있어서의 dot product 계산시, 연결대수에 따라 성능향상을 기대할 수 있는 문제규모를 분류하면 표 3과 같다.

표 3 연결대수에 따른 유효 Problem Size

연결대수	유효 Problem Size
2대	80000
4대	320000
8대	2560000

3.5 분산계산 성능분석과 성능향상 방안

순수 계산 성능 평가와 통신량 평가, 벡터의 dot product와 같은 병렬 연산에 대한 평가에서 알 수 있듯이 윈도우95 환경에서의 개인 PC의 분산 성능에 대해 요약해보면 다음과 같다.

1) 연결대수가 많으면 많을수록 순수 계산 부분만의, 즉 통신이 거의 필요치 않는 프로그램의 경우의 성능향상(speed-up)은 선형적으로 증가한다.

2) 반면, 연결대수가 많으면 많을수록 그에 따른 통신비용은 증대하고 전체 분산성능에서 통신소요시간, 즉 message passing에 의해 소요되는 시간이 상당히 큰 비중을 차지하게 되기 때문에 상호 데이터 의존성이 많은 프로그램은 분산 계산이 어렵게 된다.

3) 연결대수에 따른 일정 데이터 량을 초과하게되면 분산 성능은 분산환경을 구성하고 있는 하드웨어의 성능과도 밀접한 관계가 있다. Network를 구성하고 있는 하드웨어(LAN card, cable, hub 등)의 성능도 중요하지만, 하드디스크(hard disk)의 전송속도 및 여유공간량, 메인메모리(main memory)의 크기 및 유용 메모리량, 전송버퍼(buffer)의 크기와도 밀접한 관련이 있다.

이상과 같은 결과를 토대로 하여 최적의 병렬 시스템 성능을 구현하기 위한 방안으로 다음과 같은 사항이 고려된다.

1) 분산 성능에서 통신이 차지하는 비율이 상대적으로 크기 때문에 될 수 있으면 통신의 빈도와 통신 총량을 줄인다.

2) 전체 분산 성능은 분산 환경을 구성하고 있는 컴퓨터들 중 성능이 가장 좋지 않은 컴퓨터에 의해 결정되어지기 때문에 개개의 Slave컴퓨터들의

성능과 풀고자하는 문제의 특성을 잘 파악하여 시스템에 적합한 문제 분할과 할당이 필요하다.

3) 컴퓨터의 연결대수를 선형적으로 증가시켜도 계산 시간이 그와 비례하여 줄어드는 것이 아니기 때문에 풀고자하는 문제의 크기에 따라 적절한 컴퓨터 연결대수를 선정한다. 이를 위해서는 병렬시스템을 구성하고 있는 통신 성능 및 개개의 컴퓨터들의 기본적인 성능에 대해 사전에 숙지하고 있어야 한다.

4. 결 언

최근 컴퓨터 공학의 급속한 발전에 따른 슈퍼컴퓨터의 상용화 그리고 대형구조물의 해석 수요가 증가함에 따라 병렬해석법은 다양한 형식으로 개발되어 공학전반에 걸쳐 활발하게 적용되고 있으며 대부분의 구조해석 및 설계 알고리즘이 병렬화되고 있다. 1970년도와 1980년도의 구조해석 및 설계 분야가 유한요소법의 개발 및 적용으로 특징지어질 수 있듯이 21세기 전산구조공학분야는 병렬화로 특징지어질 것으로 판단된다[8)~11)].

그러나 국내의 전산기 보급 여건을 고려하면 대형 슈퍼컴퓨터를 이용한 병렬구조해석 및 설계법의 보편화 및 상용화는 어려운 상황이다. 그러므로 본 연구실에서는 다수의 개인용 컴퓨터를 결합하여 대형 또는 슈퍼컴퓨터의 성능을 구현할 수 있는 병렬계산기법을 PVM 및 MPI 등을 이용하여 개발 중에 있으며 이러한 가운데 병렬 구조해석 및 설계법 개발을 위한 PVM의 이해와 보급을 위해 윈도우95 환경에서의 PVM을 이용한 병렬시스템의 기본적인 분산성능을 실험·분석한 결과를 본 학술지에 기술하게 되었다.

참 고 문 헌

1. 白鳥則郎, 分散處理, Norio Shiratori, 1996
2. Al Geist, Adam Beguelin, Jack Dongarra, Robert Manchek and Vaidy Sunderam, PVM : Parallel Virtual Machine ; A Users' Guide and Tutorial for Networked Parallel Computing, The MIT Press, Cambridge, Massachusetts, 1994
3. Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker and Jack Dongarra, MPI : The Complete Reference, The MIT Press, Cambridge, Massachusetts, 1996
4. 박효선, "대형 구조물을 위한 병렬 구조해석 및 설계", *전산구조공학회지*, 제9권, 제3호, 1996년 9월
5. Lou Baker and Bradley J. Smith, Parallel Programming, McGraw-Hill, 1997
6. V. Kumar, A. Grama, A. Gupta and G. Karypis, Introduction to Parallel Computing, The Benjamin/Cummings Company, 1994
7. Dimitri P. Bertsekas and John, N. Tsitsiklis, Parallel and Distributed Computation, Prentice-Hall International, 1989
8. A. Saleh and H. Adeli, "Parallel Eigenvalue Algorithms for Large-Scale Control-Optimization Problems", *Journal of aerospace engineering, ASCE*, Vol. 9, No. 3, pp.70~79, 1996
9. Alpesh Amin and P. Sadayappan, "A Parallel Approach to Solving a 3-D Finite Element Problem on a Distributed Memory MIMD Machine", IEEE The Sixth Distributed Memory computing Conference Proceedings, pp.324~331, 1991
10. Adeli, H. and Kumar, S., "Distributed Finite-Element Analysis on Network of Workstations-Algorthms", *Journal of Structural Engineering, ASCE*, Vol. 121, No. 10, pp.1448~1455, 1995
11. Hyo Seon Park and Hojjat Adeli, "Distributed Neural Dynamics Algorithms for Optimization of Large Steel Structures", *Journal of Structural Engineering, ASCE,* Vol. 123, No. 7, pp.880~888, 1997