韓國地形空間情報學會論文集 第7卷 第2號 1999年 6月 pp. 123-132

PC-clustering을 이용한 매핑자료처리 및 변환소프트웨어에 관한 연구

A study on the process of mapping data and conversion software using PC-clustering

황보택근*, 이병욱*, 박홍기**
Whangbo, Taeg-Keun Lee, Byung-Wook Park, Hong-Gi

要 旨

컴퓨팅 알고리즘의 병렬화는 계산량 및 데이터의 증가와 더불어 필요성이 꾸준히 제기되어 왔다. 그러나 병렬처리에 사용되는 컴퓨터는 1990년대 중반까지 주로 슈퍼컴퓨터로서 가격, 사용법 등 일반인이 쉽게 접근하지 못할 요소가 많았다. 1990년대 후반에 병렬 처리를 위한 PC-cluster라는 새로운 개념이 나타나게 되었고, 아직 설치와 사용법에서 개선될 여지가 많이 있음에도 불구하고 값싼 비용으로 고성능의 계산 능력을 원하는 일반 사용자에게 PC-cluster는 가장 뛰어난 대안으로 떠오르고 있다.

GIS 데이터의 매핑은 축척변환(scale), 벡터에서 레스터로의 변환, DXF 자료구조에서 내부 자료구조로의 변환, 두 지역이 연결되었을 때 가장자리 데이터의 보정, 개체선택, Join, Cut의 처리 등 병렬 처리에 적합한 여러 가지 특성을 가지고 있다. 따라서 이들을 PC-clustering으로 구현할 경우 값싼 비용으로 실시간 처리를 할 수 있어 성능과 비용의 모든 면에서 만족할 만한 결과를 얻을 수 있을 것이다. 본 논문에서는 병렬처리 및 PC-clustring, 그리고 이들을 이용하기 위한 라이브러리 및 도구에 대한 소개와, 이들이 매핑에 어떻게 적용시킬 수 있는 가를 살펴보았다. 또한 매핑의 여러 기능을 위한 병렬 프로그램을 개발하였고, 실험 결과 노드의 수에 따라 모든 기능에서 성능이 거의 선형적으로 향상됨을 보여주고 있다.

ABSTRACT

With the rapid increases of the amount of data and computing, the parallelization of the computing algorithm becomes necessary more than ever. However the parallelization had been conducted mostly in a super-computer until the mid 1990s, it was not for the general users due to the high price, the complexity of usage, and etc. A new concept for the parallel processing has been emerged in the form of PC-clustering form the late 1990s, it becomes an excellent alternative for the applications need high computer power with a relative low cost, although the installation and the usage are still difficult to the general users.

The mapping algorithms (cut, join, resizing, warping, conversion from raster to vector and vice versa, etc) in GIS are well suited for the parallelization due to the characteristics of the data structure. If those algorithms are manipulated using PC-clustering, the result will be satisfiable in terms of cost and performance since they are processed in real time with a low cost. In this paper the tools and the libraries for the parallel processing and PC-clustering are introduced, and how those tools and libraries are applied to mapping algorithms in GIS are showed. Parallel programs are developed for the mapping algorithms and the result of the experiments shows that the performance in most algorithms increases almost linearly according to the number of node.

* 경원대학교 전자계산학과 교수 ** 경원대학교 토목공학과 교수

1. 서 론

병렬처리란 여러 대의 컴퓨터가 한 가지 목적을 위해 일을 분산하여 수행하는 방법의 통칭이다1,2,3,4). 병렬처리의 목적은 연산시간의 단축, 고장시의 대체 시스템 역할, 가용 컴퓨팅 자원의 활용, 가격 대 성능비의 향상 등이 있다. 병렬처리와 슈퍼컴퓨터 그리고 PC-clustering은 컴퓨터발전의 역사적 흐름을 간략히 소개한다. 벡터 프로세서(vector processor)를 이용한 SMP(4 CPUs)가 1990년부터 1995년까지 슈퍼컴퓨터 업계에서 두각을 나타내었으나5), RISC5 chipset의 등장으로 chipset 가격이 많이 하락하였다. 사용자 측면에서 단위 계산 성능이 향상뿐만 아니라 많은 양의 계산 작업을 빨리 끝낼 수 있는 through-put 개념 역시 중요한 관점으로 부각되어 MPP (4~1024 CPUs) 병렬 슈퍼컴퓨터가 1994년부터 1998년까지 슈퍼컴퓨터 업계를 관장하였다6). 하지만 MPP 슈퍼컴퓨터 또한 단위 작업의 계산 향상을 위한 병렬화는 고급 언어를 구사하여야 하는 힘든 작업으로 인해 보편적인 사용자들로부터 외면 당하기 시작하였다. 따라서 Vector와 MPP의 장점을 따온 CC-NUMA라는 DSMP (병렬 SMP : 4~512 CPUs) 구조를 가진 슈퍼컴퓨터 시스템의 등장이 1998년부터 현재까지 이루어지고 있다6).

그러나 사용자 입장에서는 슈퍼컴퓨터의 성능이나 기술적 발전 이전에 가격과 편의성과 상용소프트웨어의 존재여부가 더 중요한 이슈가 되었다. 미소 냉전시대 가격이라는 시장원칙에 지배되지 않는 국가연구사업으로 진행되었던 슈퍼컴퓨터가 이제는 철저한 경제원칙에 따라 이제까지 쌓아두었던 연구 성과들을 보다 값싼 시스템으로 전환하는 기술이 무엇보다 절실하게 되었다. 이러한 자구책의 성과로 등장한 PC clustering6)이란 상용화된 저 비용의 PC를 네트워크로 연결하여 병렬처리를 하는 방법을 말하는데, 이러한 시스템이 1998년부터 과학계에 널리 받아들여지고 이제 기업으로 영역을 넓혀가게 되었다. 그것은 자신들이 이미 보유하고 있는 PC, Workstation들을 네트워크로 묶어 자신들이 개발한 자체 프로그램을 수행할 때 가격 대 성능으로 슈퍼컴퓨터들을 훨씬 앞서가고 있기 때문이다.

그림 1은 cluster 시스템의 개념을 보여준다. 고속의 네트워크로 연결된 여러 대의 컴퓨터가 있고, 사용자가 이들을 하나의 컴퓨터처럼 느끼며 사용할 수 있게 하는 cluster 미들웨어가 있다1). 이 위에서 사용자들이 병렬화 언어, 라이브러리를 사용하여 프로그램을 구현하는 것이다.

클러스터 시스템의 개념도의 요소기술은 아래 3가지로 요약되는데, 이들이 현재 상용화된 제품으로 시판되고 있어 사용자들이 보다 쉽게 클러스터 시스템을 구성하여 활용할 수 있을 것이다.

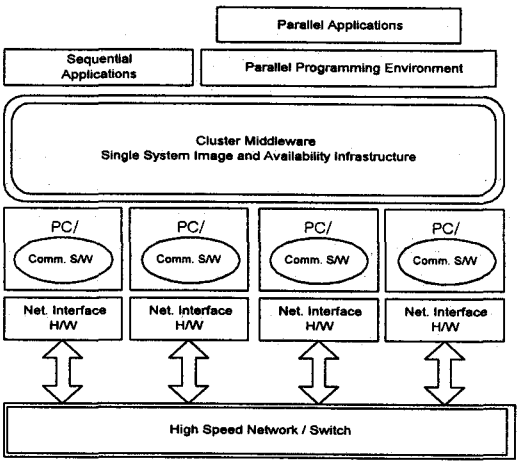


그림 1. cluster system 개념도

(가) 고성능/저가 상용 마이크로 프로세서의 유행
- 지속적으로 고속화하는 Microprocessors.
- Vector, MPP 시스템의 퇴조
- SMP, CC-NUMA 시스템의 확산

(나) 상용 고속 네트워크의 등장
- 100Mbit Ethernet
- Myricom사의 Myrinet
- DEC사의 Memory Channel

(다) 고성능 프로그래밍 환경의 일반화
- Operating System
- library(PVM, MPI, Threads,..)

- 언어(HPF, OpenMP,..)
- Single Image System S/W(RFS-1, MOSIX)

2. 병렬처리 라이브러리/병렬처리 환경

병렬처리 라이브러리는 C, FORTRAN등의 언어에서 라이브러리 형태로 사용되며 다른 노드와의 통신을 담당하는 함수들을 말한다. 그러나 일반 함수들과는 달리 다른 노드에서도 일정한 규칙을 가지는 환경이 갖추어져야 실행될 수 있으므로, 최근에는 병렬처리 환경으로 부르기도 한다. 여기에는 PVM8), MPI9,10), Linda11)등이 있으며 이중 MPI가 산업계의 표준으로 자리잡았다. 현재 OpenMP, MPI_CH 등 여러 종류의 MPI들이 출시되고 있다. PVM은 MPI보다 먼저 발표되었으나 개념 및 사용법이 유사하기 때문에 자세한 설명을 생략한다.

2.1 MPI(Message Passing Interface)

MPI는 SPC(Scallable Parallel Computers)나 NOWs(Netwroks of Workstations)에서 사용할 수 있는 message-passing에 관한 기능과 inteface를 정의한 것이다. 1991년에서 1994년에 걸쳐 MPI forum에 의해 만들어진 이 규약은 Intel의 NX12), Express13), nCube의 Vertex14), p415,16), PARMACS17) 등 많은 슈퍼컴퓨터 벤더의 영향을 받았고 산업계의 표준으로 자리잡았다. 현재 MPI가 제정한 표준은 다음과 같다.

- Point-to-point communication
- collective operations
- Process groups
- Communication domains
- Process topologies
- Environmental Management and enquiry
- Profiling interface
- Bindings for FORTRAN77 and C

이 표준들은 제정 당시 125개의 함수로 구현되어 있었으며, 현재까지 점차 기능이 추가되어 여러 버전이 만들어지고 있다. 그 중 가장 중요한 함수들은 다음과 같다.

MPI_INIT	Initialize MPI
MPI_COMM_SIZE	Find out how many processes there are
MPI_COMM_RANK	Find out which process I am
MPI_SEND	Send a message
MPI_RECV	Receive a message
MPI_FINALIZE	Terminate MPI

이 함수들을 사용하여 병렬처리를 구현한 예는 매핑 시스템의 적용에서 보여진다.

2.2 Linda

1988년에 발표된 Linda도 MPI와 같은 역할을 수행하지만 개념상 다른점이 있다. 우선 Linda는 병렬프로그램을 순차적인 연산을 전체 매니저가 조정하는 일련의 과정으로 보고 있다. 이것은 병렬처리 방법 중의 master-worker 또는 master-slave2,3,4)의 개념과 비슷하고, SMP (Shared Memory Processor)의 운영방식과도 일치한다. 다만, Linda가 사용하는 공유 메모리 공간은 가상적인 것으로 공유된 메모리 공간이 실제로 어디에 놓여있는지는 상관이 없게 설계되어 있다. Linda에서는 3-tuple 공간이라는 독특한 통신방법을 사용하는데, 이것은 게시판에 붙어있는 메모지(작업지시서)를 사용자가 한 장씩 떼어가며 일을 하는 것에 비유될 수 있다. 각 사용자는 자신의 일이 끝날 때마다 게시판을 보고 일감이 있으면 일을 하면 되는 것이고, 게시판의 관리는 매니저가 담당하면 된다. 이런 구조는 그림 2와 같다.

Linda의 장점은 MPI 방식에 비해 프로그램을 작성하기가 쉽고, 작업단위가 중간정도 크기인 문제에서는 계산부하가 잘 분산되며, 예상치 못한 에러가 발생해도 발생시점까지 역추적하기가 쉽기 때문에 무결함(fault-tolerant) 시스템을 구현하기에 용이하다. 반면, 특정한 구조를 가진 문제에만 적용될 수 있어 일반적이지 못하다는 단점이 있다. 이런 이유로 일부 은행

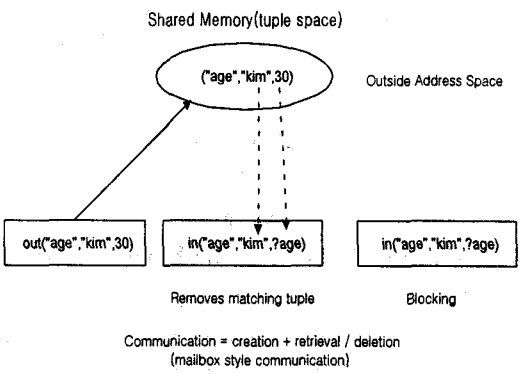


그림 2. Linda의 병렬 처리 구조

등 금융 가에서 서버로 사용되는 일은 있으나 널리 사용되지는 않고 있다.

3. Cluster Middleware

병렬처리 환경 또는 병렬처리 함수는 한 사람의 사용자가 자신의 프로그램을 병렬화 할 때 사용하는 도구라고 할 수 있다. 이때 프로그래머는 자신이 사용할 수 있는 노드가 몇 개이고, 각 노드가 해야 할 일이 무엇인지 미리 결정해주어야 한다. 또한 그 노드들을 혼자서 사용한다고 가정을 해두어야 load balancing이 된다. (MPI는 이런 가정이 필수적이며, Linda 모델은 이런 가정이 필요 없다. 현재 대부분 MPI 모델을 사용하고 있으므로 여기에 맞추어 설명한다.) 만약 다른 한 사람의 사용자가 cluster를 사용하고자 한다면 이미 수행되고 있던 프로그램의 부하를 불균등한 상태로 만들 수 있기 때문에 전체적인 성능에 큰 지장을 주게 된다. 따라서 multi-user 환경에서는 사용자간의 자원 활용법을 바꾸어 줘야 하는데 이것을 cluster 미들웨어가 담당하게 된다. 이를 해결하는 방법은 여러 가지가 있으나 많이 사용되는 방법은 partition18), gang-scheduling19), dynamic load balancing 등이 있다. 또한 Fault-tolerance 시스템을 만들기 위한 방법도 cluster-미들웨어가 담당하는 부분이나 현재 시판되기 시작한 RSF-1를 포함한 많은 경우 자세한 방법론이 알려져 있지 않다. 따라서 여기서는 Multi-user 간의 자원활용만을 다룬다.

3.1 Partition 사용

Partition이란 사용자가 자기만의 자원으로 미리 할당하여 놓은 노드를 말한다. N개의 노드를 가진 cluster에서 n(<N)개의 노드를 포함하는 partition 할당을 요청하고, 이렇게 할당되어진 노드는 다른 사용자가 사용하지 못하게 되어있다. 이것은 사용자간의 non-preemptive FIFO (First-In First-Out)라고 볼 수 있으며, Intel Paragon17) 128 노드 이상의 대형 병렬 컴퓨터에서 주로 사용하는 방법이다.

3.2 Gang-scheduling

N개의 노드로 이루어진 cluster에서 여러 사용자가 동시에 N + n (n>0)개의 process를 수행할 경우, 적어도 n개의 process는 time-sharing을 해야만 한다. 이때 각 노드의 scheduler간 동기화를 시켜줄 필요가 있고, 결과적으로 작업이 수행되는 모든 노드에서 동기화 된 time-sharing을 하게 되는 것이다. 위와 같은 방법을 Gang-scheduling이라고 부르며 일본의 RWCP(Real World Computing Partnership)에서 만든 Score-D19)는 이런 방법을 채택했다. 그림 3은 Score-D의 작동 원리를 보여준다.

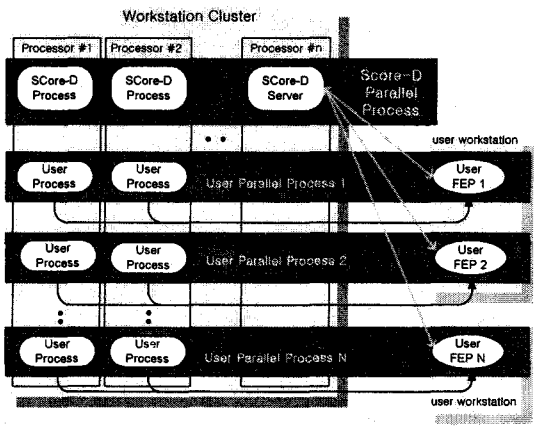


그림 3. Score-D의 작동 원리

3.3 Dynamic Load Balancing

Dynamic Load Balancing은 각 노드의 부하를 프로그램 수행 중에 자동적으로 조절하는 기능을 말한다. Clustering 환경에서 잘 알려진 것으로는 MOSIX20,21,22), RFS-1이 있다. RSF-1은 상업용으로 dynamic load balancing, fault-tolerant 알고리즘이 자세히 공개되지 않아, 공개 소프트웨어인 MOSIX를 기준으로 알아본다.

MOSIX는 클러스터 컴퓨팅을 위한 소프트웨어로 1990년대 초반부터 개발되어 버전 7이 개발되어 있다. MOSIX는 고성능계산과 확장성을 염두에 둔 적응성 자원할당 알고리즘(adaptive resource sharing)에 주안을 두고 만들어졌다. 사용자의 입장에서 가장 큰 특징은 여러 대의 컴퓨터들이 마치 하나의 시스템처럼 보인다는 것으로, 프로그램을 실행하거나 프로그램을 작성할 때 별도의 라이브러리를 사용하거나 병렬처리에 대한 지식이 없어도 가능하다는 점이다.

MOSIX 알고리즘은 하나 혹은 여러 process들을 네트워크로 연결된 컴퓨터로 자유롭게 이전할 수 있게 하여, load-balancing을 도모하고 메모리 등 자원의 무리한 사용을 막아준다. 그림 4는 부하가 많은 노드 0으로부터 부하가 적은 노드 1로 프로세스가 이동하는 모습을 보여준다. 노드 0 에는 이전된 프로세스가 되돌아올 지점을 기록하여 두어서, 노드 1에서의 작업이 완료된 프로세스의 standard In/Out 및 종료 시 반환되어야 할 지점을 알 수 있게 해둔다.

MOSIX의 내부를 이해하기 위해서는 SMP(Shared Memory Processor, 2,4-way형 PC도 포함된다)와의

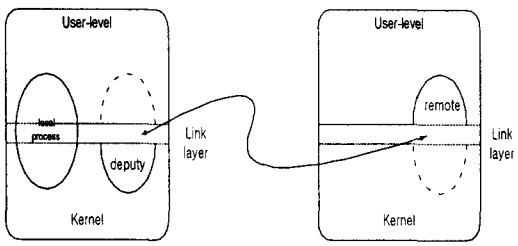


그림 4. MOSIX 에서의 프로세스 이주

비교가 필요하다. SMP 시스템에서는 몇 개의 프로세서가 하나의 메모리를 공유하게 되는데, 이것의 최대 장점은 계산 속도의 향상과 프로세서간의 빠른 정보 전달이다. 하나의 메모리만 있으므로 프로세스는 계산의 결과에 능동적으로 대처하여 전체를 효율적으로 운영하기가 매우 쉽다. 또한 사용자들은 마치 1 CPU를 사용하듯 이런 여러 프로세서간의 자원관리가 일어나고 있는지조차 알지 못한다.

SMP와는 달리, cluster computing 환경은 여러 대의 성격이 다른 컴퓨터들을 네트워크로 연결한 것이기 때문에, 이런 컴퓨터들의 상황이 제각기 다름을 인정해야 한다. 이런 컴퓨터들을 노드라고 부르며, 워크스테이션, PC, SMP 등 어떤 종류의 컴퓨터라도 연결되기 때문에 계산속도나 메모리, 디스크 등이 다르다. 그리고 각 node의 운영체제와 시스템 사양이 같다고 하더라고 각 운영체제는 자신의 자원만 관리하기 때문에, 이들은 하나의 시스템으로 사용하는 데는 한계가 있다.

PVM, MPI, LSF, Linda, Extreme Linux 등 많은 소프트웨어들이 여러 노드를 묶어 하나의 컴퓨터처럼 사용하는 방법을 제공해 주지만, MPI program 예제에서 보여주듯이 이런 방법은 응용소프트웨어로써 작동하는 것으로 사용자가 각 노드의 자원에 맞게 load balancing을 해주어야 한다. 이 작업은 프로그램을 수행하기 전에 어떤 작업이 어떤 노드에 할당되는지 미리 정해주어야 하는 것이므로, 프로그래밍의 어려움과 더불어 상황변화에 능동적으로 대처하지 못한다는 단점이 있다.

Load balancing외에도 CPU, memory, I/O, IPC등의 자원을 다루는 문제는 더욱 복잡하고 사용형태도 예측할 수가 없다. 게다가, 여러 사용자들이 함께 사용하는 경우는 수행하는 프로그램들이 상호 영향을 줄 수 있기 때문에, 한 사람이 자원을 잘 관리하도록 프로그램을 최적화하는 것이 무의미해질 수가 있다. (이 문제의 해결을 위해, Partition을 도입하거나[Intel Paragon], Gang scheduling[RWCP, SCore-D] 등을 사용하는 경우는 앞 절에서 설명했다.) SMP는 OS 및 시스템소프트웨어7), 컴파일러23,24,25), 라이브러리 26,27) 등이 거의 모든 것을 다 해결해주고 있으나, 현

재의 PC Cluster 환경은 이런 문제의 해결을 많은 부분 사용자의 능력에 의존하고 있으며, 결과적으로 성능을 저하시킬 load imbalance문제에 심각하게 노출되는 것이다.

MOSIX는 CC환경을 SMP환경에 가까워지게 만드는 것으로 클러스터에 사용된 모든 자원을 관리하여 동적 부하균형을 유지해준다. 이것은 multi-user, time-sharing, non-uniform cluster에서 큰 효과가 있다고 알려져 있다.

4. 매핑 시스템에의 적용

매핑 시스템에서 지도상의 각 개체를 나타내는 데이터는 완전히 독립적으로 추출되고 변환될 수 있기 때문에 병렬성이 가장 좋은 알고리즘을 가지고 있다. 이 시스템을 cluster에 설치하고 수행할 때, 효율을 높일 수 있는 경우는 크게 두 가지로 나눌 수 있다.

첫째는 multi-user 시스템에서 여러 사람이 동시에 작업을 수행하고자 하는 경우로 각 사용자의 프로그램이 병렬화가 되는 것이 아니라, MOSIX같은 미들웨어 수준에서 부하균등화를 시켜주는 것으로, 이는 한 사람의 입장에서 계산속도가 빨라지는 것이 아니다.

두 번째는 축척변환, 추출, 보정 등 각 단계를 병렬화 하여 사용자의 계산수행 시간을 줄이는 방법이다. 물론 두 가지 방법을 병행하여 사용할 수도 있다. middleware를 사용하는 방법은 사용자 입장에서 볼 때 하나의 PC를 사용하는 것과 동일하기 때문에, 여기서는 표준화된 병렬화 라이브러리인 MPI를 사용한다.

4.1 병렬화 대상 알고리즘

4.1.1 축척 변환 (scaling)

화면에 적재(load)된 DXF이미지는 임의의 비율로 확대/축소 될 수 있고, 특정 부분만을 선택한 후 다시 그릴수도 있다. 이 때에는 전체 화면 또는 선택된 부분에 속하는 모든 block과 entity에 대하여 같은 비율을 곱하는 단순 작업을 반복하게 된다. 이 작업은 각 entity에 대하여 독립적으로 수행할 수 있으므로 병렬화시 높은 효율이 보장된다.

4.1.2 Vector와 Raster의 변환 (frame buffer로의 copy)

DXF나 매핑을 위해 만든 HGIS 자료구조는 모두 vector로 표시되어 있다. 이들을 이미지화 하기 위해서는 이 자료구조의 모든 entity에 대하여 frame 버퍼의 각 pixel에 투영하는 작업이 필요하게 된다. Entity를 적절히 분할하여 여러 대의 컴퓨터에서 각각의 frame 버퍼에 그린 후, 이 frame 버퍼들의 pixel 값을 더하고 사용한 컴퓨터의 개수로 나누어주면 이미지가 완성된다. DXF로 표현된 하나의 entity를 frame 버퍼에 그리는 시간보다 frame 버퍼들을 더하고 나누는 시간이 훨씬 적을 경우 효과적인 방법이 될 수 있다.

4.1.3 DXF 자료구조에서 내부 자료구조로의 변환

DXF 자료구조는 크게 HEADER, TABLE, BLOCK, ENTITY section으로 이루어지고, 실제 지도에 그려질 내용물들은 BLOCK과 ENTITY에서 정의된다. 이 두 section에서 사용되는 데이터들은 line, point, circle, arc, solid, text, shape, block, polyline, vertex, trace, 3D face, 3D line, layer들로 구성되어 있다. 이 데이터들은 TIFF형식으로 저장되기 전에 매핑 프로그램에서 사용하는 내부구조체로 전환되는데, 이 과정은 DXF 각각의 entity에 대하여 개별적으로 수행할 수 있다. 다만 복잡한 entity의 경우 Block section과 같이 읽어 들여야 하므로 데이터를 분할하는 과정에서 granularity의 조정이 필요하게 된다.

4.1.4 보정

연결된 두 지역의 DXF 데이터를 이미지에 그렸을 때, 가장자리 부분에서 어긋남이 발생할 수 있다. 이를 보정할 경우 어긋남이 발생하는 가장자리만이 아닌 전체 도면이 일정한 비율로 scale되거나 변형 해주어야 전체가 자연스럽게 보인다.

4.1.5 개체 선택 (Selection)

화면상에서 한 개체를 선택하여 삭제 혹은 변경하고자 할 때는 어느 개체가 선택되었는지 먼저 판단하

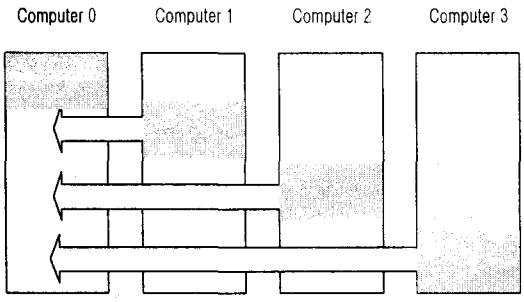
여야 한다. 이럴 경우에 전체 entity를 나누어 검색할 수 있다.

4.1.6 연결(join) 및 끊어냄(cut) 처리

원래의 scale한 후에는 수치상의 오류로 인해 cut, join이 발생하게 된다. 이를 보정하기 위해서는 전체 entity를 대상으로 검색을 진행할 필요가 있으며, 이 경우로 각각의 entity에 대해 독립적인 작업 수행이 가능하다.

4.2 MPI를 사용한 병렬 프로그램

그림 5. 병렬화 개념도



여기서는 DXF의 모든 entity를 entity[SIZE]라는 array에 저장하고 있다고 가정한 후 이를 여러 대의 컴퓨터가 이들을 적절히 분배하여 축적 변환하는 과정을 보여준다. 4대의 컴퓨터를 사용할 경우 그림 5에서와 같이 entity 「SIZE] 영역을 적절히 분배하여 축적 변환한 후 한대의 컴퓨터(computer 0)로 모은다. 이때 계산하는 시간이 전송시간보다 훨씬 크면 클수록 병렬화의 효율은 높아진다.

그림 6에 DXF의 entity를 축적 변환하는 병렬 프로그램을 보여주고 있다. 나머지 알고리즘에 대하여도 유사한 방법이 적용된다.

```
#include "mhi.h"
#include <math.h>
int main(argc, argv)
int argc;
char *argv[];
{
   int n, myid, numprocs, i;
   int num_entity;
   float scale;
   int start, end, size;
   int stat;

   MPI_Init(&argc, &argv);

   MPI_Comm_size(MPI_COMMM_WORLD,
         &numprocs);
   MPI_Comm_rank(MPI_COMM_WORLD,
&myid);

   n = num_entity / numprocs;
   start = myid*n;
   end = (myid+1)*n;
   if (end > num_entity) end = num_entity;

   for (i=start; i<end; i++)
      do scaling for each entity[i];   /* entity
data * scale */

   if (myid == 0) {
      for (i=1; i<4; i++)
         MPI_Recv (&start, 1, MPI_INT, i,
START_POS, MPI_COMM_WORLD,   &stat);
         MPI_Recv (&end, 1, MPI_INT, i,
END_POS, MPI_COMM_WORLD,      &stat);
         MPI_Recv (&entity[start], end-start,
            MPI_INT,      i,      ENTITY,
 MPI_COMM_WORLD, &stat);
    } else {
         MPI_Send (&start, 1, MPI_INT, 0,
START_POS, MPI_COMM_WORLD);
         MPI_Send (&end, 1, MPI_INT, 0,
END_POS, MPI_COMM_WORLD);
```

```
    MPI_Send (&entity[start], end-start,
MPI_INT, 0, ENTITY, MPI_COMM_WORLD);
  }

/* 여기서 computer 0은 축적 변환된 entity를 모두
가지고 있다. */
do_something(entity);

MPI_Finalize();
}
```

그림 6. DXF entity 축적 변환을 위한 병렬 프로그램

4.3 결과

병렬처리의 효율성은 사용한 노드의 수에 대한 계산시간의 감소분인 속도향상으로 측정한다. N개의 노드를 사용하여 M배의 계산속도를 얻는다면 속도향상은 M/N이며 1.0 에 가까워질수록 이상적이다. 속도향상의 이론적 한계 치는 Amdahls law3)에 따른다. 여기서는 매핑 시스템에 적용된 6개의 알고리즘에 대한 각각의 속도향상을 보여준다. 실험에 사용된 장비는 펜티엄 166MHz 4대(64MB memory), 3COM SuperStack Switch 3330 이다.

그림 7에 병렬 처리가 적용된 매핑 기능에 대한 노드 수에 대한 계산속도의 향상을 나타내었다. 평균 4대의 PC에서 0.91의 속도향상을 보여주었다. 이 결과는 노드를 구성하는 PC와 네트워크의 성능에 크게 좌우되며, 일반적으로 노드의 수가 증가하면 통신시간도 증가하기 때문에 속도향상은 감소추세를 보여준다. 그러나 매핑 시스템에 적용된 알고리즘은 대량적인 parallel3)의 성질을 지니고 있어(통신의 비중이 상대적으로 미약하다) 16-32 노드에서도 좋은 속도향상을 보여줄 것으로 기대된다.

5. 결 론

본 논문에서는 병렬처리에 대한 소개와 이를 매핑 시스템에 적용했을 때의 속도향상을 보여주었다. 매핑 시스템에 사용되는 알고리즘은 근본적으로 병렬성을 내재하고 있어 한 사람이 보다 빠른 연산을 요구하거나 여러 사람이 동시에 작업을 할 때도 성능이 저하되지 않는 시스템을 구성할 수 있었다. 병렬시스템은

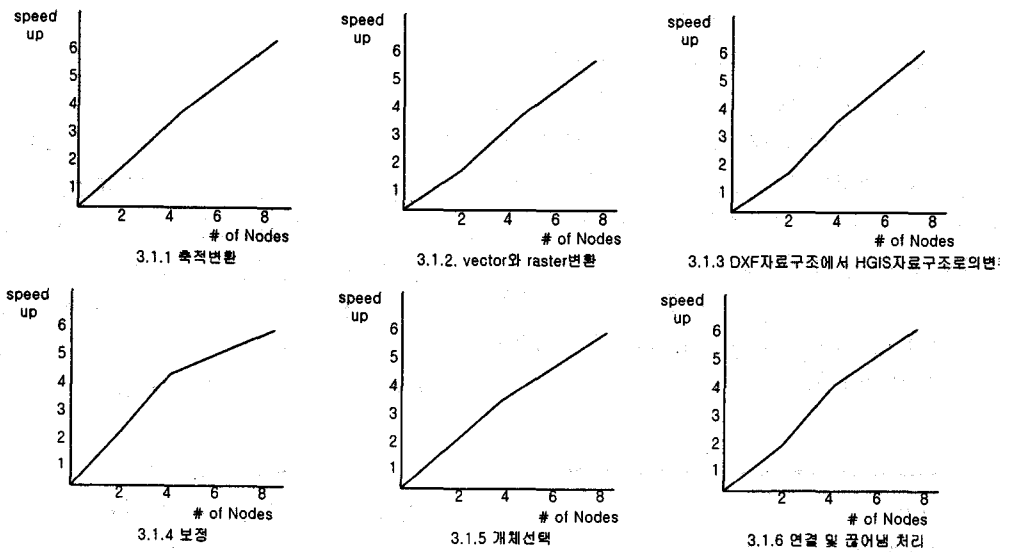


그림 7. 노드 수에 따른 매핑 알고리즘 성능 향상

Linux를 탑재한 2-8대의 PC에서 mpich 라이브러리를 사용하였고, 4대의 PC에서 평균 0.91의 속도 확장성이 있었다. 이는 4대의 PC를 사용했을 시 4*0.91배의 속도향상이 있었음을 말한다. 속도 확장성은 많은 PC를 사용할수록 감소하지만 GIS시스템의 알고리즘은 통신의 비중이 작아 16-32 노드를 사용하는 것도 가능하다.

아직 일반 프로그래머에게 병렬프로그램을 작성하기가 쉬운 것은 아니지만 값싼 비용으로 고성능의 매핑 시스템을 구성하고자 한다면 PC-cluster는 유력한 대안이며 개별 알고리즘의 병렬화를 전체 소프트웨어에 접목시키는 노력이 필요할 것이다.

참 고 문 헌

1. Jack Dongarra , Springer-Verlag, "Parallel Scientific computing", First International Workshop, PARA'94 proceedings, 1994
2. Ian T.Foster, "Designing and Building Parallel Programs", Addison-Wesley Publishing Company, 1995
3. Ted G.Lewis and In-Kyu Kim, "Introduction to Parallel Computing", Prentice-Hall Inc. 1992
4. G.Fox, "Solving Problems on Concurrent Processors volume I, II", Prentice-Hall Inc., 1988
5. Kevin Dowd, "High Performance Computing", OReilly & Associates, Inc. 1993
6. Gregory F. Pfister, "In Search of Clusters 2nd Edition", Prentice Hall PTR, 1998
7. Paul Messina, "System Software and Tools for High Performance Computing Environments", SIAM,1993,1995(2nd version)
8. Al Geist, "PVM:Parallel Virtual Machine, A User's Guide Tutorial for Networked Parallel Computing", MIT Press, 1994
9. Marc Snir, "MPI : The Completer Reference", MIT Press, 1996
10. William Gropp, "Using MPI : Portable Parallel Programming with the Message-Passing Interface", MIT Press, 1994
11. Barak A., La'adan O. and Shiloh A., "Scalable Cluster Computing with MOSIX for LINUX", Proc. Linux Expo '99, pp. 95-100, Raleigh, N.C., May 1999.
12. Paul Pierce. "The NX/2 operationg system". Proc. of the Third conference on Hypercube Concurrent Computers and Applications, pp. 384-390, ACM Press, 1988
13. Parasoft Corporation. "CA.Express Users Guide". Pasadena, 1992
14. nCube Coporation. "NCube 2 Programmers Guide", 1990
15. R. Butler and E.Luck, "Users guide to theP4 progrmming system". Technical Report TM-ANL-92/17, Argon National Lab. 1992
16. R. Butler and E. Lusk, "Monitors, messages and clusters: the P4 parallel programming system". Journal of Parallel Computing, 20(4):5547-564, April 1994
17. Robin Calkin, Rolf Hempel, Hans Christian Hoppe, and Peter Wypior. "Portable programming with the parmacs message-passing library". Parallel Computing, 20(4): pp. 615-632, April 1994
18. Intel Corporation, "Paragon System User's Guide", Intel Corporation, 1995
19. A. Hori, "Real World Computing Partnership. Highly Efficient Gang Scheduling Implementation ", Supercomputing 98, 1998, Orlando, FL, USA.
20. Barak A., La'adan O. and Shiloh A., "Scalable Cluster Computing with MOSIX for LINUX", Proc. Linux Expo '99, pp. 95-100, Raleigh, N.C., May 1999.
21. Barak A., Gilderman I. and Metrik I., "Performance of the Communication Layers of TCP/IP with the Myrinet Gigabit LAN", Accepted to the Computer Communications journal, January 1999.
22. Amir Y., Awerbuch B., Barak A., Borgstrom R.S. and Keren A., "An Opportunity Cost Approach

for Job Assignment in a Scalable Computing Cluster", Proc. 10th International Conference on Parallel and Distributed Computing and Systems (PDCS'98), pp. 639-645, Las Vegas, Nevada, October 1998.
23. Michael Wolfe, "High Performance Compilers for Parallel Computing", Addison-Wesley, 1996
24. David R. Wille, "Advanced Scientific Fortran", John Wiley & Sons, 1995
25. T.M.R Ellis, "Fortran 90 programming", Addison -Wesley, 1994
26. E. Anderson, "LAPACK User's Guide", SIAM, 1995
27. K.A.Gallivan, "Parallel Algorithms for Matrix Computations", SIAM, 1994
28. Michael S. Warren, Timothy C. Germann, Peter S. Lomdahl, David M. Beazley, and John K. Salmon, "Avalon : An Alpha/Linux cluster Achieves 10 Gflops for $150K", SuperComputing 98, Orlando, FL, USA.