구조 해석에서의 병렬처리

송 윤 환*

1. 머릿말

컴퓨터를 이용한 문제 해석에 있어서 보다 빠른 시간 내에 문제를 풀고자 하는 노력은 계산 처리 속도가 탁월한 컴퓨터를 개발시키는 동기가 되었으며 소형 컴퓨터에서부터 대형 컴퓨터에 이르기까지 매우 우수한 컴퓨터들이 현재 사용되고 있다.

컴퓨터 하드웨어에 있어서 가장 중요하다고 할 수 있는 중앙 처리 장치(Central Processing Unit, IBM PC와 같은 개인용 컴퓨터에서는 micro-processor 라고도 한다)는 컴퓨터에게 주어지는 명령들을 수행해 내는 부분이며, CPU를 여러개 가지고 있는 컴퓨터를 병렬 컴퓨터(parallel computer)라고 한다. 병렬처리(parallel processing)란 병렬 컴퓨터를 이용하여 여러개의 태스크(task)를 동시에 수행하는 것을 말한다. 대부분의 병렬 컴퓨터는 각각의 CPU들이 벡터처리(vector processing) 를 담당하는 기능부들을 가지고 있어서 기본적으로 이 벡터처리를 통해서 병렬처리를 한다.

병렬처리 컴퓨터는 이미 오래 전부터 개발되어 왔으며, 근래에 개발된 수퍼컴퓨터(예를 들어 국내 KIST에서 보유하고 있는 CRAY-2는 4개의 background processor와 1개의 foreground processor 를 가지고 병렬처리와 벡터처리를 지원한다[1])는 거의 대부분 병렬 컴퓨터이다. IBM PC와 같은 개인용 컴퓨터에서도 병렬 보드(Inmos사의 Transputer)를 PC main board의 확장 slot에 설치함으로써 병렬처리를 할 수 있다[2]. 현재까지 병렬 컴퓨터들이 많이 개발되었음에도 불구하고 대부분의 병렬 컴퓨터가 가지고 있는 벡터처리 방식을 이용하여 그 계산속도를 높이고 있을 뿐 소프트웨어적으로 병렬 알고리즘을 개발하고자 하는 노력은 미미한 실정이다.

구조 해석에 있어서 가장 많이 사용되는 유한 요소 해석은 특히 3차원 문제에 있어서 그 계산량이 많이 요구된다. 따라서 빠른 계산 처리 능력을 가지고 있는 병렬처리 컴퓨터의 사용은 연구자들에게 연구 의욕을 증대시킴은 물론, 보다 빠른 연구 진척을 기대하게 한다. 기존의 직렬 프로그램을 병렬처리 환경에 적합하도록 재작성하는 일은 사실 대단히 번거롭고 또 때로는 어려운 일이며, 또한 병렬 컴퓨터들이 여러가지 방식을 가지고 있고 표준화된 병렬처리 언어가 없다는 점 등의 문제점이 있다. 이러한 문제점들이 프로그래머들에게 약간의 걸림돌이 되고 있으나 곧 통합된 시스템이 출현하리라 예상된다.

본 고에서는 벡터 처리에 대하여 간략히 알아보고, 기존의 순차적인 구조해석 알고리즘을 어떻게 병렬화할 것인가와 병렬처리 환경에 적합한 구조 해석 알고리즘에는 어떠한 것이 있는가 등에 대하여 간단하게 살펴보기로 한다.

* 정회원, 한국과학기술원 토목공학과 박사과정

2. 벡터처리

벡터처리는 다중처리(multi processing)와 함께 대표적인 병렬처리 방식이며 대부분의 슈퍼 컴퓨터는 벡터처리와 다중처리 기능을 가지고 있다. 벡터처리는 array에 관한 연산을 벡터 단위로 pipelining에 의하여 동시에 처리하는 것을 말하며, 기존의 스칼라 처리 방식에 비하여 더욱 빠르고 효율적이다[1, 3].

기존의 프로그램에서 벡터처리 능력을 최대한 활용하기 위해서는 프로그램을 벡터처리에 맞게 수정하는 작업이 필요하다. FORTRAN에 의해 만들어진 프로그램에서 벡터처리가 가능한 부분은 가장 안쪽의 DO-loop 및 DO-loop으로 변환이 가능한 loop 구조이다. 간단한 예를 통해서 어떻게 loop 구조를 벡터처리에 맞게 고칠 수 있는가에 대하여 알아보기로 한다[1].

경우 1

```
      DO 10 I=2, N
        DO 10 J=2, N
10        A(I, J)=A(I, J-1)*2.0
```

경우 2

```
      DO 10 J=2, N
        DO 10 I=2, N
10        A(I, J)=A(I, J-1)*2.0
```

경우 1과 경우 2의 계산 결과는 동일하다. 그러나 두번째 줄의 DO-loop이 경우 1에서는 J에 관한 loop이기 때문에 $J=k(3\leq k\leq N)$일 때 세번째 줄에서의 A(I, k)는 A(I, k−1)을 미리 계산을 해야 구할 수 있으나, 경우 2에서는 I에 관한 loop이므로 A(I, J)에 대한 계산이 I가 2부터 N사이의 임의의 성분일 때의 계산 결과에 영향을 받지 않는다. 따라서 병렬 컴퓨터라 하더라도 경우 1에서는 계산 과정이 단계적으로 순차적으로 수행될 수 밖에 없으나 경우 2에서는 벡터처리 방식으로 다량의 데이타를 동시에 처리할 수 있다.

이상에서 살펴 본 벡터화(vectorization) 기법을 사용하면 기존의 순차적인 프로그램을 약간의 수정에 의하여 손쉽게 병렬화할 수 있으나, 알고리즘 전체를 병렬화하기에는 한계가 있다. 따라서 벡터화와 함께 병렬처리에 적합한 알고리즘을 개발하고자 하는 노력이 병행되어야 할 것이다.

3. 병렬처리 환경에 적합한 구조 해석 알고리즘

병렬처리 알고리즘을 구축하는 데에 있어서 항상 염두에 두어야 할 사항은 각 CPU간의 자료 교환을 최소화하면서, 모든 CPU들의 작업량이 동일하도록 해야 한다는 것이다.

기존의 유한요소 직렬 알고리즘 상에서 쉽게 병렬화가 가능한 부분은 요소강성도 계산 과정과 변위 계산 이후에 응력을 구하는 과정일 것이다. 그러나 유한 요소 해석에 있어서 가장 많은 계산 시간을 요하는 부분은 일반적으로 선형시스템의 해를 구하는 과정이므로 상기의 단순한 병렬화는 그다지 도움이 되지 못할 수도 있다.

3.1 부구조법(Substructuring technique)에 의한 병렬처리

예전의 컴퓨터의 코아 메모리가 작았던 시대에 대형 문제를 풀기 위한 매우 효율적인 방법으로 인식되었던 부구조 해석법[4]은 병렬처리 환경에 적합하다고 볼 수 있다. 부구조 해석법에서는 전체 구조물을 여러개의 부구조로 분리하여 각 부구조에 대하여 내부 자유도와 경계 자유도에 대한 강성 행렬을 구성한 후, condensation 기법을 통해 경계 자유도만에 의한 강성 행렬을 조합한다. 이때 생성된 시스템 방정식은 전체 구조물에 대한 그것에 비하여 그 크기가 작으며 이에 대한 해를 구한 후, 역대입 과정을 통해서 내부 자유도에 대한 해를 구한다. 각각의 부구조에 대한 강성 행렬의 구성, condensation 과정과 응력 회복 과정은 다른 부구조와 서로 독립적으로 진행되기 때문에 각각의 부구조들은 CPU들에게 적절히 할당함으로써 병렬처리를 할 수 있다.

경계 자유도만에 의해 구성된 시스템 방정식은 전체 자유도에 대한 시스템 방정식에 비해 그 크기가 작기는 하나 이를 가우스 소거법에 의한 직접법으로 풀 경우 순차적인 영역이 된다.

이것에 대한 해결책으로서 직접법 대신에 반복법[5]을 사용하여 해를 구하는 방법과 다중 부구조법(Multi-level substructuring technique)[6]을 이용하여 풀어야 할 시스템 방정식의 크기를 더욱 줄이는 방법등이 있다. 전체 구조물을 부구조로 분할하는 데에 있어서 가장 주의를 요하는 사항은 각각의 CPU들의 작업량을 거의 같도록 해야 한다는 것이다. 이는 구조물의 형상이 복잡할 경우에 매우 어려운 일이며, 이에 대하 자동적인 부구조 분할 기법이 연구되고 있다[7].

3.2 요소단위 수행(Element-by-element solution procedure)에 의한 병렬처리[8]

다음에 병렬처리 환경에 적합한 알고리즘으로 요소 단위 수행 구조를 들 수 있다. 요소 단위 수행 방법은 병렬처리를 위하여 특별히 개발된 방법은 아니며, 부구조 해석법처럼 적은 용량의 컴퓨터에서 대형 문제를 효과적으로 풀기 위하여 고안된 방법이라고 볼 수 있다. 이 방법에서는 전체 시스템 방정식을 구성하지 않고 요소 차원(또는 substructure 차원)에서 해를 구한다.

이 방법은 해석하고자 하는 구조물을 CPU 갯수만큼의 부구조로 분리하여 각각의 부구조에 하나의 CPU를 할당하여 각 부구조별로 동시에 해를 구하는 방법이므로 유한요소 알고리즘 전체를 병렬화했다고 볼 수 있다. 이때의 방정식 해법은 반복 해법이 적절하다고 알려져 있으며[8], 반복 해법 중 Conjugate gradient 방법[9]을 많이 사용하고 있다. 이때 Conjugate gradient 알고리즘을 요소 단위수행 구조에 적합하게 수정을 하여 사용한다.

요소 단위 수행 구조는 분명히 순차적인 영역을 완전히 제거한 병렬처리 구조이긴 하지만, 부구조 차원에서 해를 구할 때 인접된 부구조와 자료 교환을 해야 한다. 이러한 자료 교환이 많을수록 overhead 시간 때문에 효율성이 떨어질 것이며, 또한 반복 해법의 수렴 속도가 늦는 경우에 다른 병렬 알고리즘에 비하여 오히려 시간이 많이 걸릴 수 있다. 따라서 자료 교환 양을 줄이고 Conjugate gradient 방법의 수렴 속도를 높이는 방법 등이 계속 연구되어야 할 것이라고 생각된다.

4. 맺는말

본 고에서는 현재 많은 관심의 대상이 되고 있는 병렬처리에 대해서 현재까지 연구되어온 병렬 환경에 적합한 구조해석 알고리즘에 대하여 간략히 설명하였다. 앞으로 병렬 컴퓨터가 더욱 일반화되고 표준화되리라 예상되므로, 방대한 계산량을 요구하는 유한요소 해석에 대한 보다 효율적인 병렬 알고리즘의 개발을 위하여 현재까지 진행된 연구에 대한 분석 및 더욱 많은 노력을 기대한다.

참 고 문 헌

[1] CRAY Research, INC., *Training Workbook*, 1988.

[2] Inmos, *IMS B008 User Guide and Reference Manual*, 1990.

[3] 이지호, 이재석, 김문헌, "Memory-to-Memory 방식 컴퓨터에서의 외연적 유한요소법의 벡터화", 한국전산구조공학회 제4권 제1호, 1991. 3.

[4] J.S. Przemieniecki, *Theory of Matrix Structural Analysis*, McGraw-Hill, 1968.

[5] Alan Jennings, *Matrix Computation for Engineers and Scientist*, John Wiley & Sons, 1977.

[6] T. Furuike, "Computerized Multiple Level Substructuring Analaysis", Computers & Structures, Vol.2, pp.1063-1073, 1972.

[7] Charbel Farhat and Edward Wilson, "A New Finite Element Concurrent Computer Program Architecture", Int. J. for Numerical Methods in Engineering, Vol.24, pp.1771-1792, 1987.

[8] Kincho H.Law, "A Parallel Finite Element Solution Method", Computers & Structures 23., No.6, pp.845-858, 1986.

[9] David G. Luenberger, *Linear and Nonlinear Programming*-2nd Edition, Addison-Wesley Publishing Company, 1989.

[10] 황찬규, 유한요소법에서의 병렬처리 알고리즘의 개발, 석사학위논문, 서울대학교 토목과, 1990.

[11] Charbel Farhat, Edward Wilson and Graham Powell, "Solution of Finite Element Systems on Concurrent Processing Computers", Engineering with Computers 2, pp.157-165, 1987.