'96 대한전기학회 하계학술대회 논문집 (1996. 7. 22~24)

유한요소망에서의 효율적인 직접해법 병렬계산에 관한 연구

°이항범*, 최경**, 김형중***, 정현교****, 한송엽****

*기초전력공학공동연구소 **강원대학교 전자공학과
***강원대학교 제어계측공학과 ****서울대학교 전기공학과

Study of Efficient Parallel Computation of Cholesky's Method in FE Mesh

H. B. Lee*, K. Choi**, H. J. Kim***, H. K. Jung****, S. Y. Hahn****

*EESRI (Electrical Engineering and Science Research Institute)
**Dept. of Electronics Eng., Kangwon Nat'l Univ.
***Dept. of Control/Instr. Eng., Kangwon Nat'l Univ.
****Dept. of Electrical Eng., Seoul Nat'l Univ.

Abstract -- In this Paper, an efficient parallel computation method for solving large sparse systems of linear algebraic equations by using Cholesky's method in the finite element method is studied. The methods of minimizing the number of fill-ins in the factorization process of factorization are investigated for minimizing the amount of memory and computation time. The parallel programming is implemented under the PVM (Parallel Virtual Machine) environment. The method of load-distribution is studied for minimizing the computation time and the communication time.

1. 서론

수치해석기법에 대한 많은 연구 및 컴퓨터 성능의 증대에 따라 수치해석을 이용한 기술의 사용이 증대되고 있으며, 수치해석을 이용하여 다루는 문제의 크기도 증대하고 있다. 대부분의 수치해석 기법은 대형 행렬방정식을 푸는 문제가 포함되어 있으며, 대부분의 시간은 행렬식을 계산하는데 사용되고 있다. 비록 컴퓨터의 성능이 비약적인 발전을 하였지만, 해석하고자 하는 문제의 크기가 커짐에 따라 한 대의 컴퓨터에서 계산하기 힘든 문제가 발생하고 있다. 따라서, 이러한 문제는 한 대의 컴퓨터가 아닌 여러 대의 컴퓨터에 문제를 분할시켜서 계산하는 병렬처리방법을 이용하면 가능하리라고 생각할 수 있으며, 이에 대한 연구도 많이 되어 왔다. 기존의 대부분의 병렬처리는 전용통신망을 이용한 방법을 사용하거나, 한 대의 컴퓨터에서 여러 개의 중앙처리장치를 사용하는 방법으로 실행되었다. 하지만, 네트워크 망의 비약적인 발전으로 대부분의 컴퓨터는 네트워크 망에 연결되어 있는 현재의 상황에서 기존의 네트워크 망을 이용한 병렬처리기법을 개발한다면 그 효용성이 상당히 크리라고 사료된다. 따라서, 본 논문에서는 유한요소법과 같은 해석기법에서 많이 발생하는 성긴 행렬(Sparse Matrix)에 대하여 기존의 네트워크 망에서 병렬 계산하는 방법의 효율적인 구현방식에 대하여 연구하였다. 먼저 성긴 행렬에 대한 해석기법을 살펴보았고, 네트워크망 환경에 알맞는 병렬처리 기법에 대하여 살펴보았다. 또한 미지수 배열법과 부하배분법 등에 대한 병렬알고리즘의 효율성에 대하여 부하균등과 통신량 축소의 측면에서 각각의 방법의 장단점을 비교하였다.

2. 직접해법에 대한 고찰

2.1 행렬의 직접해법에 대한 고찰

식 (1)과 같은 행렬을 푸는 직접해법으로는 크게 가우스소거법 및 Cholesky법이 있다. 가우스소거법은 알고리즘이 간단한 반면에 구동항 f가 변하면 매번 해석해야한다. Cholesky법은 알고리즘이 상대적으로 복잡하지만 구동항이 바뀌더라도 다시 해석하지 않고 간단히 해를 구할 수 있다.

$$Ax = f \tag{1}$$

2.2 저장방법

수치해석 기법을 이용하여 행렬을 조립할 때, 성긴 행렬에서는 행렬의 상당부분에 값이 차지 않는다. 따라서, 값이 차는 부분만을 고려하여 메모리를 잡게 되는데, 이러한 방법으로 Skyline법(SKY)과 행압축방법(CRS)을 이용하였다. 직접해법을 적용하여 해를 구하는 경우 원래는 값이 차지 않았지만, 새로이 값이 차는 부분이 생기며, 이를 fill-in이라 한다. Skyline의 경우 추정가능한 대부분의 fill-in을 메모리로 잡으며, CRS법에서는 실제로 fill-in이 생기는 부분만을 메모리로 잡는다.

2.3 절점번호 재부여

직접해법을 이용하여 해를 계산하는 과정에 생기는 fill-in은 절점의 번호를 어떻게 부여하는가에 따라 fill-in의 개수가 달라진다. 따라서, 메모리를 효율적으로 사용하기 위해서는 최적의 절점번호재부여방법을 사용하여야 한다. 본 논문에서는 원래 유한요소생성과정에서의 절점번호(ORG) 및 유한요소망에서 한 절점을 기준으로 거리에 의한 재부여방법(DIS), Nested Dissection법(NDM), Reverse Cuthill-McKee법(RCM)에 대하여 살펴보았다.

2.4 실제 계산 예

미지수의 개수가 8481개인 경우에 대하여 다음 표에 fill-in갯수, 메모리사용량, 계산횟수 및 계산시간에 대하여 살펴보았다. 여기서, 계산을 하기전의 행렬에 값이 차는 부분은 33,801개이다. SKY의 경우 알고리즘은 단순한 반면 메모리의 사용이 많고, CRS의 경우 알고리즘은 복잡하나 메모리의 사용이 적다. 표에서 보면 CRS형식의 NDM의 경우 fill-in 및 필요메모리가 다른 방법에 비하여 현저히 적음을 알 수 있다. 또한 계산시간에서도 상당한 장점이 있는 것을 알 수 있다.

표 1. 각 방법에서의 fill-in의 갯수

	CRS	SKY
ORG	12,361,223	13,961,452
DIS	833,544	833,544
NDM	295,969	2,386,925
RCM	1,178,591	1,193,282

표 2. 각 방법에서 필요한 주메모리용량

	CRS	SKY
ORG	99,228,040	112,097,720
DIS	7,006,608	7,074,456
NDM	2,706,008	19,501,504
RCM	9,766,984	9,952,360

표 3. 각 방법에서 필요한 보조메모리용량

	CRS	SKY
ORG	50,291,032	609,156
DIS	4,180,316	609,156
NDM	2,300,424	1,116,048
RCM	5,594,428	1,082,124

표 4. 각 방법에서 계산횟수

	CRS	SKY
ORG	*	*
DIS	48,284,527	26,809,654
NDM	16,831,618	2,245,196
RCM	16,831,618	96,427,092

표 5. 각 방법에서 계산시간

	CRS	SKY
ORG	*	*
DIS	9.16	38.97
NDM	6.00	7.42
RCM	18.85	132.88

3. 병렬처리환경

본 논문에서는 기존의 네트워크 망을 이용할 수 있는 방법인 PVM(Parallel Virtual Machine)을 이용하여 병렬처리 알고리즘을 구현하였다. PVM은 인터넷에서 쉽게 구할 수 있는 프로그램으로, 같은 기종이 아닌 컴퓨터일지라도 같은 네트워크에 연결되어 있으면 병렬처리가 가능하다. 또한, Windows 95에서도 사용이 가능하므로, 병렬처리 알고리즘이 실현이 쉽다는 장점이 있다.

본 논문을 위하여 PVM 3.3.10 버전을 사용하였으며, 하드웨어 구성은 HP735 기종 8대가 FDDI로 연결된 HP-Cluster를 사용하였다. 각각의 HP735는 99MHz의 CPU속도를 가지고 있으며, 40MFLOPS의 연산속도를 가지고 있다. 또한, 각각의 메모리 및 하드디스크는 80MByte 및 1GB이다. OS는 HPUX-9.05를 사용하였다. 또한 프로그램은 포트란을 이용하여 작성되었다. FDDI의 속도는 100MBPS까지 가능하며, 일반 이더넷은 10MBPS까지 가능하다.

다음 표에서는 PVM을 이용하여 실제 느낄 수 있는 속도를 측정하였다. 6대의 컴퓨터에서 10^6크기를 가진 배정도 실수를 1000번 주고받은 경우의 속도를 측정하였다. 각각의 컴퓨터에 자료를 전송하는 경우보다는 한꺼번에 자료를 보내는 경우에 속도가 빠름을 알 수 있다.

표 6. 통신속도의 측정

	한꺼번에 6대에 보내는 경우	6대에 각각 보내는 경우
걸린 시간[초]	188.21	226.13
속도[MBPS]	40.8055	33.9628

4. 병렬처리 알고리즘에서의 부하배분

부하배분은 세 가지 경우에 대하여 살펴보았다.

- 순환배분(CYCL배분) : 1번 행은 1번 컴퓨터, 2번 행은 2번 컴퓨터에 .. 의 순으로 배분하였다. n번 행은 Mod(n/컴퓨터수)에 의한 순서이다.
- NDM Ordering에 따른 배분(NDM배분) : NDM을 이용하여 절점번호 재부여 과정에서 나오는 순서에 따른 분할
- NDM Block에 따른 배분(NDMB배분) : NDM을 이용한 절점번호 재부여 과정에서 나오는 순서를 컴퓨터의 수에 맞추어 Block단위로 분할하여 부여하였다.

각각의 경우에 대하여 통신량 및 CPU시간 사용량은 다음 표와 같다. 먼저 통신량의 변화를 살펴보면,

컴퓨터의 수의 감소에 따라 통신량은 감소하고 있다. 즉, 컴퓨터의 수의 감소에 따라 서로 주고받을 자료의 수가 적어짐을 알 수 있다. 순환배분과 NDM배분의 경우는 통신량이 크게 차이가 없음을 알 수 있고, NDMB의 경우 통신량이 약 20%정도 감소함을 알 수 있다. 이는 순환방식에 비하여 전체를 크게 분할하여 배분하는 것이 유리함을 알 수 있다.

CPU사용시간을 살펴보면 순환배분의 경우가 가장 균등하게 CPU사용을 하고 있음을 알 수 있고, NDMB의 경우에서 NDM방법을 보면 편차율이 92% 정도나 됨을 볼 수 있고, 이는 한 컴퓨터에 과도한 계산을 하게 하여 병렬처리의 효율을 떨어뜨리고 있다고 할 수 있다. 여기서, 편차율은 표준편차를 평균시간으로 나눈 것의 백분율이다.

표 7. 순환배분경우의 통신량

컴퓨터수	DIS	NDM	RCM
8	142,491,900	53,157,612	196,281,840
7	124,757,392	46,719,564	172,144,156
6	106,960,736	40,227,960	147,869,860
5	89,064,912	33,644,872	123,382,688
4	70,996,880	26,946,688	98,571,812
3	52,583,416	20,053,696	73,165,504
2	33,307,268	12,771,468	46,415,084

표 8. NDM배분경우의 통신량

컴퓨터수	DIS	NDM	RCM
8	142,283,196	52,311,144	195,142,384
7	124,601,292	46,069,672	171,091,176
6	106,846,400	39,668,204	146,926,076
5	88,976,228	33,253,644	122,637,976
4	70,917,104	26,635,008	97,957,172
3	52,519,936	19,867,792	72,747,200
2	33,258,356	12,691,820	46,260,268

표 9. NDMB배분경우의 통신량

컴퓨터수	DIS	NDM	RCM
8	108,697,200	43,892,364	154,982,024
7	94,371,588	38,104,820	133,840,776
6	83,565,392	33,667,000	118,469,464
5	71,146,508	28,394,264	101,002,784
4	56,664,912	22,811,748	80,016,400
3	41,917,620	16,993,012	59,530,016
2	26,783,584	10,796,104	37,211,040

표 10. 순환배분경우의 CPU사용시간 (컴퓨터수=8)

컴퓨터수	DIS	NDM	RCM
8	22.06	9.69	31.21
7	21.03	8.75	35.06
6	21.59	8.83	31.23
5	22.85	8.76	34.35
4	21.82	8.96	33.70
3	21.62	8.50	32.95
2	23.50	7.81	29.34
1	21.03	8.69	30.81
평균시간	21.9375	8.7488	32.3313
표준편차	0.8040	0.4853	1.8544
편차율	3.66%	5.55%	5.74%

표 11. NDM배분경우의 CPU사용시간 (컴퓨터수=8)

컴퓨터수	DIS	NDM	RCM
8	22.03	8.84	38.30
7	19.90	9.97	31.25
6	23.17	8.28	31.35
5	20.72	8.59	32.04
4	23.76	8.70	30.42
3	19.24	7.99	28.14
2	18.64	7.45	26.82
1	19.43	8.94	28.50
평균시간	20.8613	8.5950	30.8525
표준편차	1.7907	0.6944	3.2905
편차율	8.58%	8.08%	10.67%

표 12. NDMB배분경우의 CPU사용시간 (컴퓨터수=8)

컴퓨터수	DIS	NDM	RCM
8	12.50	38.51	14.95
7	15.98	9.92	19.76
6	15.94	7.05	18.67
5	15.86	5.24	18.78
4	18.45	6.93	31.87
3	14.74	8.72	33.02
2	12.87	5.93	32.88
1	16.97	8.08	34.98
평균시간	15.4138	11.2975	25.6138
표준편차	1.8659	10.3805	7.7251
편차율	12.11%	91.88%	30.16%

5. 결론

본 논문에서는 유한요소망에서의 효율적인 직접해법 병렬계산에 관하여 연구하였다. 행렬해법으로는 직접법을 사용하였으며, 유한요소망에서 나오는 성긴 행렬에 대하여 메모리의 사용량 및 해석시간에 대하여 알아보았다. 병렬처리프로그램은 이 기종간의 병렬처리에 알맞은 PVM을 사용하였다. PVM병렬처리 환경에서의 통신속도 및 통신량에 대하여 알아보았으며, 부하배분에 따른 각 컴퓨터의 CPU사용시간에 대하여 살펴보았다. 유한요소 행렬의 직접해법 병렬화에는 NDM에 의한 미지수 배열이, NDMB에 의한 부하배분이 유리함을 알 수 있었다. 앞으로의 방향으로 통신량의 최소와 각 컴퓨터에 대한 CPU사용시간의 균등을 동시에 만족하는 부하배분에 대한 연구가 요구된다.

참고문헌

[1] Alan George and Joseph W-H Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall Inc., New Jersey, ISBN 0-13-165274-5, 1981.

[2] Richard Barreit, Michael berry, and etc., Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, from netlib.ornl.gov.

[3] Al Geist, Jack Dongarra, and etc., PVM3 User's Guide and Reference manual, from netlib.ornl.gov, 1994.