論 文
41～7～4

# 트랜스퓨터를 이용한 有限要素解析의 竝列計算

# A Parallel Computation of Finite Element Analysis on a Transputer System

金 根 煥*·崔 炅**·鄭 炫 敎***·李 基 植§·韓 松 曄§§
(Keun-Hwan Kim · Kyung Choi · Hyun-Kyo Jung · Ki-Sik Lee · Song-Yop Hahn)

*Abstract* - This paper presents a parallel algorithm for the finite element analysis using relatively inexpensive transputer parallel system. The substructure method, which is highly parallel in nature, is used to improve the parallel computing efficiency by splitting up the whole structure into substructures. The proposed algorithm is applied to a simple two-dimensional magnetostatic problem. It is found that the more the number of transputer is increased, the more the total computation time is reduced. And the computational efficiency becomes better as the number of internal boundary nodes becomes smaller.

**Key Words** : Parallel Computing(병렬계산), Substructure Method(부분구조법), Transputer(트랜스퓨터) F.E.M.(유한요소법)

## 1. 序 論

有限要素法을 이용하여 電磁器機를 해석하고 설계하는 방법이 많이 보편화 되고 있다. 이때 해석대상이 복잡하거나 대형화되면 컴퓨터의 계산속도와 메모리의 한계에 부딪히게 된다. 특히 좀 더 실제적인 문제나 3차원 모델을 해석하기 위해서는 슈퍼컴퓨터 정도의 계산 능력이 필요하게 된다. 최근 공학용 워크스테이션이 싼 가격에 매우 빠른 계산속도를 보이고 있어 이의 활용이 기대되지만 위의 문제를 해결하기는 미흡하다. 또한 한개의 CPU를 사용하는 컴퓨터는 그 성능에 한계가 있어 점차로 고성능 컴퓨터는 다수의 CPU를 쓰는 경향이다.[1] 그 중의 하나가 다수의 트랜스퓨터를 사용하여 병렬계산 시스템을 구성하는 것인데 비교적 가격이 저렴하면서 계산성능을 상당히 높일 수 있다.[2, 3]

본 연구에서는 IBM PC를 기반으로 하고 다수

*正 會 員 : 서울大 大學院 電氣工學科 碩士課程
**正 會 員 : 江原大 工大 電子工學科 助敎授 · 工博
***正 會 員 : 江原大 工大 電氣工學科 副敎授 · 工博
§正 會 員 : 檀國大 工大 電氣工學科 敎授 · 工博
§§正 會 員 : 서울大 工大 電氣工學科 敎授 · 工博
接受日字 : 1992年 3月 4日
1次修正 : 1992年 5月 8日

의 트랜스퓨터를 사용하여 靜磁界등의 문제를 유한요소로 해석하는 방법을 제시하였다. 그런데, 유한요소법에 의하여 생성되는 시스템 행렬식은 매우 성기기(sparse) 때문에 기존의 방법으로는 병렬계산을 하여도 좋은 성능을 얻기 힘들다. 그러므로 항공분야나 구조역학 분야에서 일찌기 컴퓨터 메모리 한계때문에 많이 사용해오고 있는 部分構造法 (substructure method)[4~7]을 이용하여, 유한요소법을 효과적으로 병렬화하여 동시계산에 의한 계산속도 증가효과를 얻고 영역분할에 의해 메모리부족 문제를 완화하는 방법을 연구하였다. 본 연구에서는 2차원 문제로 그 효용성을 검증하였으나 3차원 문제에도 쉽게 적용될 수 있다.

## 2. 部分構造法

그림 1과 같이 전 계산영역을 여러개의 領域으로 나누자. 나누어진 小領域을 하나의 部分構造라 하고 이 부분구조에서 유한요소 방정식을 세우면 아래와 같이 쓸 수 있다.

$$[P]\{\phi\}=\{Q\} \tag{1}$$

여기서 $[P]$와 $\{Q\}$는 계수행렬 및 입력벡터이고 $\{\phi\}$는 절점의 상태변수이다. 아랫첨자 $e$를 각 부분구조에서 그 부분구조를 둘러싸고 있는 경계절점을 나타내고 아랫첨자 $i$를 그 부분구조의 내부절점을 나타내도록 하면 식(1)은 다음과 같이 분리하여 쓸 수 있다.

$$\begin{bmatrix}[P_{ee}][P_{ei}]\\ [P_{ie}][P_{ii}]\end{bmatrix}\begin{bmatrix}\{\phi_e\}\\ \{\phi_i\}\end{bmatrix}=\begin{bmatrix}\{Q_e\}\\ \{Q_i\}\end{bmatrix} \tag{2}$$

단, 여기서 $\{\phi_e\}$는 경계절점의 상태변수, $\{\phi_i\}$는 내부결점의 상태변수, $[P_{ee}]$, $[P_{ei}]$, $[P_{ie}]$, $[P_{ii}]$는 그 부분구조의 계수행력을 분리한 부분행렬, $\{Q_e\}$, $\{Q_i\}$는 $\{Q\}$를 분리한 입력벡터이다.

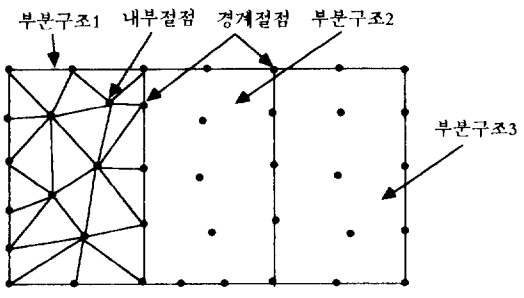



**그림 1** 세 부분구조를 가진 모델
**Fig. 1** A model with three substructure.

식(2)를 풀어쓰면

$$[P_{ee}]\{\phi_e\}+[P_{ei}]\{\phi_i\}=\{Q_e\} \tag{3}$$
$$[P_{ie}]\{\phi_e\}+[P_{ii}]\{\phi_i\}=\{Q_i\} \tag{4}$$

와 같이 된다. 식(4)로 부터

$$\{\phi_i\}=[P_{ii}]^{-1}(\{Q_i\}-[P_{ie}]\{\phi_e\}) \tag{5}$$

와 같이 되고 이것을 식(3)에 대입하면

$$[P_{ee}']\{\phi_e\}=\{Q_e\}' \tag{6}$$

와 같은 행렬방정식이 된다. 여기서 $[P_{ee}]'$과 $\{Q_e\}'$은 식(3), (4)의 부분행렬로 부터 다음과 같이 계산된다.

$$[P_{ee}]'=[P_{ee}]-[P_{ei}][P_{ii}]^{-1}[P_{ie}] \tag{7}$$
$$\{Q_e\}'=\{Q_e\}-[P_{ei}][P_{ii}]^{-1}\{Q_i\} \tag{8}$$

각 부분구조에 대해 식 (6)을 계산하고 그것을 모아서 전체 경계절점에 대한 계수행렬을 구성하고 경계조건을 대입하고 풀어서 전체 경계절점에 대한 상태변수$\{\phi_e\}$를 구한다. 이것을 해당되는 부분구조의 식(5)에 대입하여 각 부분구조의 내부절점의 상태변수 $\{\phi_i\}$를 구한다.

## 3. 部分構造法의 竝列化

부분구조법에 의한 유한요소해석 과정이 그림 2의 흐름도에 나타나 있다. 흐름도의 첫번째 과정

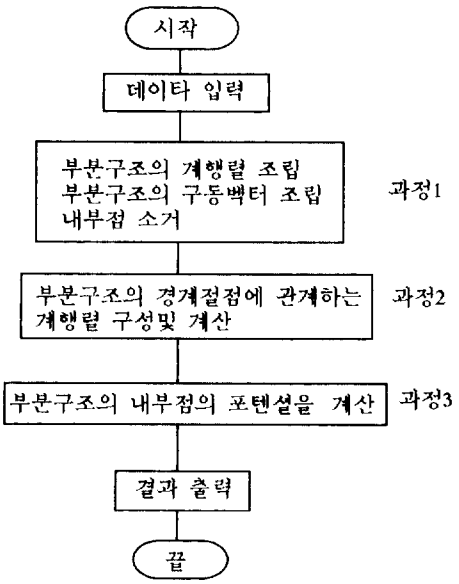



**그림 2** 부분구조법의 흐름도
**Fig. 2** The flowchart of substructure method.

에서는 각 부분구조의 계수행렬을 조립하고 내부점 소거를 하는데 이 과정은 각 트랜스퓨터가 완전히 독립적으로 수행한다. 매스터 트랜스퓨터는 IBM-PC의 화일로부터 요소데이타와 절점좌표를 읽어서 각 트랜스퓨터에 전송해준다. 각 트랜스퓨터는 자기가 맡은 부분구조들 수 만큼의 $[P_{ii}]^{-1}$, $[P_{ee}]$, $[P_{ei}]$, $[P_{ie}]$, $[P_{ii}]$, $[Q_e]$, $[Q_i]$를 저장할 배열들을 가지고 있으며 한개의 부분구조의 계행렬 $[P]$를 저장할 배열을 가지고 있다. 우선 각 부분구조에서 식(2)와 같이 경계절점과 내부절점에 대해서 행렬을 분리하기 위해서 그 부분구조의 절점번호를 경계절점, 내부절점순으로 정렬한다. 이때 각 트랜스퓨터는 각 부분구조의 요소데이타 배열에서 한 요소씩 가져와서 그 요소의 절점번호를 새로운 정수배열 $\{I\}$에 삽입하며 정렬한다. 정렬이 끝나면 $[P]$ 행렬의 원소들을 생성하고 식(7), (8)에 의해 내부절점을 消去하는 계산을 한다. 이 계산도중 $[P_{ii}]^{-1}$의 계산은 가우스-조르단법을 사용한다.

두번째과정은 全 領域의 경계절점 상태변수를 구하는 과정으로 내부점 소거가 끝난 하위 트랜스퓨터는 자신이 담당한 부분구조들의 $[P_{ee}]'$와 $\{Q_e\}'$를 매스터 트랜스퓨터에 보내고 매스터 트랜스퓨터는 이들을 받아서 자기가 가지고 있는 全 경계절점 크기의 행렬에 더해 넣는다. 모든 부분구조에 대해서 이 동작이 끝나면 이 행렬에 고정경계조건을 대입하고 이 행렬을 다음절에서 설명하는 바와 같이 순환적으로 잘라서 트랜스퓨터에 보낸다. 각 트랜스퓨터는 서로 데이타를 전송하면서 동시에 가우스 消去法의 上三角化 과정을 수행한다. 후퇴대입과정은 병렬성이 적고 계산시간이 많이 들지 않으므로 매스터 트랜스퓨터에서 일괄 계산한다. 후퇴대입후 구해진 전체 경계절점의 상태변수는 각 부분구조에 관련되는 $\{\phi_e\}$로 나누어 그 부분구조의 $\{I\}$ 배열을 참조하여 각 트랜스퓨터로 전송한다.

세번째과정은 각 트랜스퓨터가 각자의 계산된 $\{\phi_e\}$를 받아서 식(5)에 의해 각 부분구조의 내부점의 상태변수 $\{\phi_i\}$를 계산하는 과정으로 각 트랜스퓨터가 동시에 계산한다.

전체적으로 첫째과정과 세째과정은 완전히 독립적으로 동시 계산하고 둘째과정은 서로 통신을 하면서 계산한다. 이때 계산시간이 많이 소요되는 과정은 첫째와 둘째과정인데 첫째는 내부점과 관련된 부분행렬의 역행렬을 계산할때 시간을 많이 차지하고 둘째과정은 비교적 큰 경계점 계수행렬을 전진소거하는데 계산시간을 많이 차지한다. 첫째과정과 둘째과정의 시간은 서로 이율배반적으로 증가와 감소를 하므로 총 경계절점과 총 내부절점을 적절히 조절해서 두 과정에서 걸리는 시간을 합한것이 최소가 되도록 해야한다. 또한 주의할 것은 負荷均等(load balance) 문제인데 각 트랜스퓨터가 담당하는 부분구조의 절점의 갯수가 같거나 비슷해야 제대로 효율이 나온다.

## 4. 가우스 消去法의 竝列化 構造

본 연구에서는 IBM-PC와 트랜스퓨터를 파이프라인[10] 식으로 연결하였다. 이 배열은 트랜스퓨터의 갯수가 비교적 많지 않을때 유리하며 프로그램이 간편하고 해석이 용이하다.

가우스소거법 수행시 행렬배치는 그림 3과 같이 전체행렬을 행단위로 쪼개어 순환적으로 각 트랜스퓨터에 저장한다. 즉 M을 매스터 트랜스퓨터를 제외한 트랜스퓨터 갯수라 하고 N을 전체행렬의 크기라 하면 각 트랜스퓨터는 가로로는 N, 세로로는 $(N/M+1)$ 크기의 부분행렬과 $(N/M+1)$ 크기의 부분벡터를 선언해 둔다. 이렇게 순환적으로 배치하는 이유는 병렬계산시의 부하균등을 위해서이다.

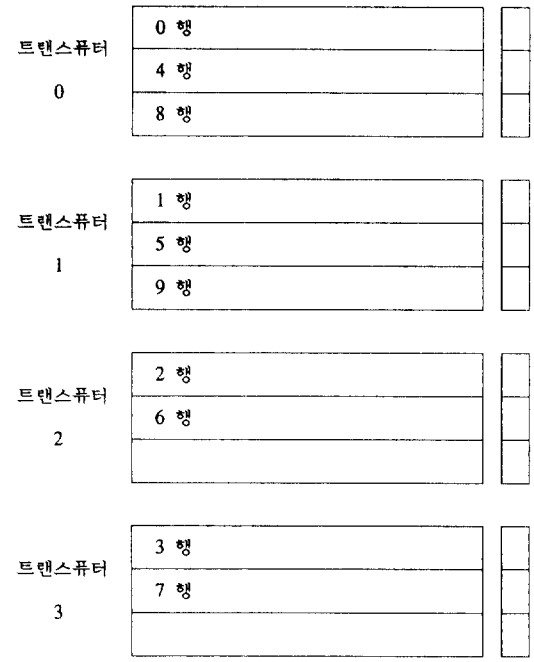



**그림 3** 가우스 소거법을 위한 행렬배치($M=4$, $N=10$인 경우)

**Fig. 3** A matrix mapping scheme for Gaussian elimination. (when $M=4$, $N=10$)

上三角化 과정은 $K$번째 行을 가진 트랜스퓨터가 그 行 전체를 다른 트랜스퓨터들에 순차적으로 전송하면서 수행된다. 다른 트랜스퓨터들이 $K$行의 원소를 받게되면 $K+1$ 행부터 $N$행까지에서 자신이 가지고 있는 行들을 列단위로 연산하여 변경한다.

## 5. 事例硏究

병렬화된 부분구조법의 효용성을 검증하기 위해서 그림 4와 같은 靜磁界문제를 다루었다. 全 領域을 16개, 24개, 32개, 48개, 64개의 부분구조로 나눈 모델을 트랜스퓨터를 8개, 16개, 24개 써서 계산하였다. 트랜스퓨터 수보다 분할 영역 수가 많은 경우는 한개 트랜스퓨터 내에 여러개의 계산 프로세서를 발생시켜 계산하였다. 또한 트랜스퓨터 한개로 순차알고리즘(sequencial algorithm)을

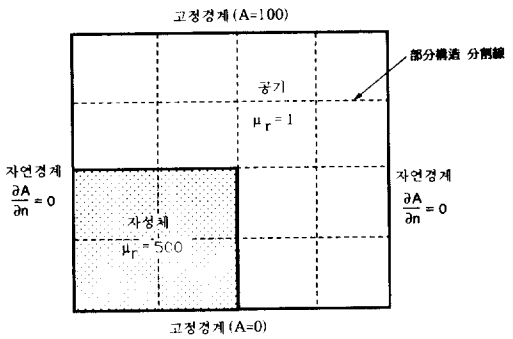



**그림 4** 적용 정자장 모델(16개 부분구조 분할인 경우)

**Fig. 4** A simple two dimensional magnetostatic model of 16 substructre.

**표 1** 5가지 경우의 부분구조로 나누어진 모델의 사양

**Table 1** Specifications of five substrutured models.

| 부분구조수 | 총절점수 | 총요소수 | 부분구조당 내부절점수 | 부분구조당 경계절점수 | 총경계절점 |
|---|---|---|---|---|---|
| 16 | 2401 | 4608 | 169 | 48 | 465 |
| 24 | 2401 | 4608 | 117 | 40 | 553 |
| 32 | 2401 | 4608 | 91 | 36 | 641 |
| 48 | 2401 | 4608 | 63 | 28 | 721 |
| 64 | 2401 | 4608 | 49 | 24 | 801 |

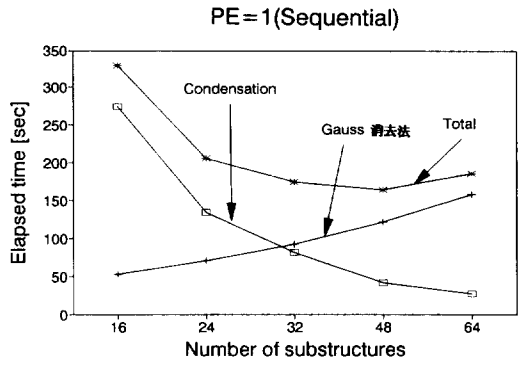



**그림 5** 순차알고리즘에 의한 계산시간(초), $PE=1$일때

**Fig. 5** Computing time of sequential algorithm (SEC), when $PE=1$.

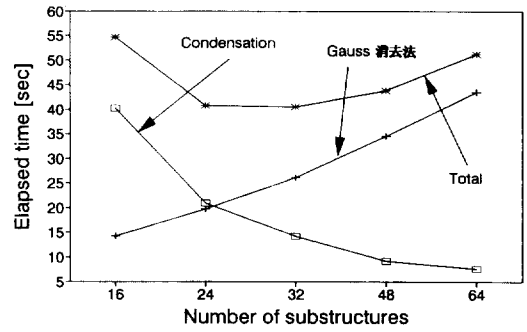



**그림 6** 병렬알고리즘에 의한 계산시간(초), $PE=8$일때

**Fig. 6** Computing time of parallel algorithm (SEC), when $PE=8$.

**표 2** 순차알고리즘에 의한 계산시간(초), $PE=1$

**Table 2** Computing time of sequential algorithm (SEC), $PE=1$.

| 부분구조수 | 조립 및 내점소거 | 경계의 포텐셜 계산시간 | 내부의 포텐셜 계산시간 | 계 |
|---|---|---|---|---|
| 16 | 275.0 | 52.5 | 1.63 | 329.2 |
| 24 | 134.1 | 70.9 | 1.09 | 206.1 |
| 32 | 81.74 | 92.26 | 0.81 | 174.8 |
| 48 | 41.87 | 121.8 | 0.55 | 164.3 |
| 64 | 27.6 | 158.2 | 0.42 | 186.2 |

이용하여 계산하고 그 계산시간을 병렬 알고리즘의 계산시간들과 비교하였다. 사용한 시스템의 매스터 트랜스퓨터는 8 MB의 메모리를 갖고 있고 다른 트랜스퓨터는 1 MB의 메모리를 가지고 있다. 각 트랜스퓨터는 T800 이며, 20 MHz의 CPU로 10 mips, 1.5 Mflops의 계산속도를 가지고 있다. 프로그램 언어는 트랜스퓨터용 병렬처리 언어 Occam을 사용했다.[9] 해석모델의 총절점수는 2401개이고 총요소수는 4608개이며 요소를 균등분할하여 여러 경우의 부분구조분할이 용이하도록 하였다. 표 1에 부분구조로 나누어진 모델의 5가지의 경우에 대한 정보를 나타내었고 그림 1에 16개 부분구조로 분할 했을 때의 分割 例를 도시하였다.

표 2에 순차알고리즘에 의한 계산시간을 나타내었고 표 3, 표 4, 표 5에 각각 트랜스퓨터를 8개, 16개, 24개 써서 병렬계산한 시간을 나타내었다. 그림 5, 그림 6, 그림 7, 그림 8에 각각의 계산시간을 그래프로 비교한 결과가 나타나 있다. 병렬계산에 의한 전체 계산시간의 단축효과는 순차알고리즘에서의 계산시간이 부분구조 수에 따라 변화가 심하여 일괄적으로 산출하기는 어려우나 8개 트랜스퓨터를 사용하였을 때는 최대 6배, 16개 트랜스퓨터를 사용하였을 때는 최대 11배의 계산속도를 얻을 수 있었다. 병렬알고리즘에서 부분구조의 분할 갯수에 따른 계산시간의 변화추이를 보면, 그림 5에 나타난 순차알고리즘 계산시간에서는 48개의 부분구조로 나누었을 때가 가장 계산시간이 적었으나 그림 6의 8개 트랜스퓨터인 경우에는 24개 부분구조일때가 가장 계산시간이 적었으며 16개일때는 32개 부분구조일때가 적은 등 트랜스퓨터가 늘수록 부분구조의 갯수가 적은 경우가 계산시간이 훨씬 적었다. 그 이유로는 부분구조법의 두번째 과정의 경계절점만이 관계하는 계수행렬을 병렬 가우스소거법으로 계산하는 부분에서 계산시간이 많이 소요되기 때문이며 이것은 이 행렬이 아직도 매우 성기기(sparse) 때문에 기존의 성긴행렬 연산기법에 의한 가우스소거법과 비교할

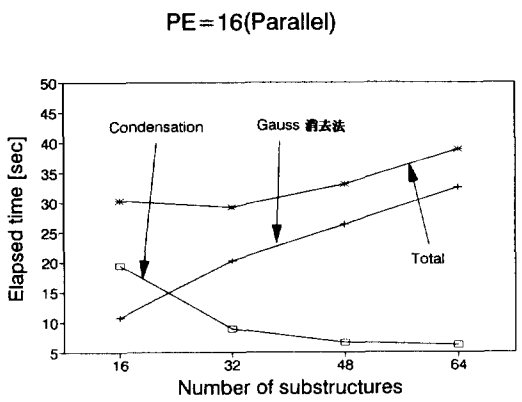



**그림 7** 병렬알고리즘에 의한 계산시간(초), $PE$=16일때

**Fig. 7** Computing time of parallel algorithm (SEC), when $PE$=16.

**표 3** 병렬알고리즘에 의한 계산시간(초), $PE$=8일때

**Table 3** Computing time of parallel algorithm (SEC), when $PE$=8.

| 부분 구조수 | 조립 및 내점소거 | 경계의 포텐셜 계산시간 | 내부의 포텐셜 계산시간 | 계 |
|---|---|---|---|---|
| 16 | 40.23 | 14.2 | 0.26 | 54.7 |
| 24 | 20.9 | 19.7 | 0.18 | 40.8 |
| 32 | 14.27 | 26.1 | 0.15 | 40.5 |
| 48 | 9.17 | 34.5 | 0.11 | 43.8 |
| 64 | 7.7 | 43.5 | 0.1 | 51.3 |

**표 4** 병렬알고리즘에 의한 계산시간(초), $PE$=16일때

**Table 4** Computing time of parallel algorithm (SEC), when $PE$=16.

| 부분 구조수 | 조립 및 내점소거 | 경계의 포텐셜 계산시간 | 내부의 포텐셜 계산시간 | 계 |
|---|---|---|---|---|
| 16 | 19.4 | 10.7 | 0.13 | 30.2 |
| 32 | 8.9 | 20.2 | 0.09 | 29.2 |
| 48 | 6.62 | 26.3 | 0.08 | 33.0 |
| 64 | 6.2 | 32.5 | 0.07 | 38.8 |

**표 5** 병렬알고리즘에 의한 계산시간(초), $PE$=24일때

**Table 5** Computing time of parallel algorithm (SEC), when $PE$=24.

| 부분 구조수 | 조립 및 내점소거 | 경계의 포텐셜 계산시간 | 내부의 포텐셜 계산시간 | 계 |
|---|---|---|---|---|
| 24 | 7.46 | 13.56 | 0.07 | 21.1 |
| 48 | 5.9 | 23.5 | 0.07 | 29.5 |

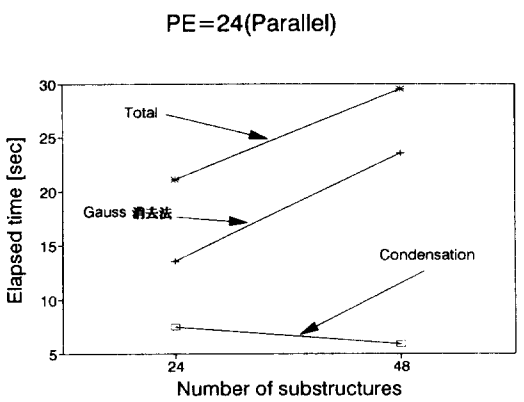



**그림 8** 병렬알고리즘에 의한 계산시간(초), $PE$ =24일때

**Fig. 8** Computing time of parallel algorithm (SEC), when $PE$=24.

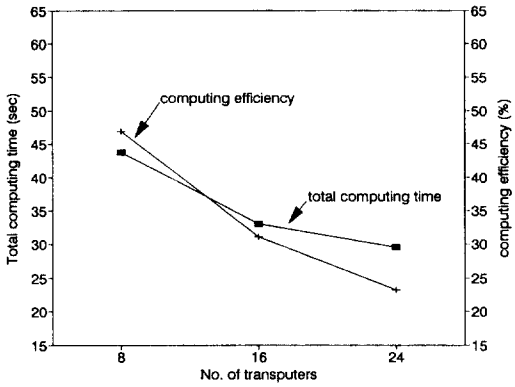



**그림 9** 48개 부분구조의 총 계산시간 및 단위 트랜스퓨터 계산효율

**Fig. 9** computing time and elementary efficiency of 48 substructured model.

때 계산생략 과정에서 실제 계산량보다는 통신에 많은 시간이 걸리기 때문이다. 특히 트랜스퓨터의 갯수가 많아질때는 정보를 전파해야할 거리가 멀어지므로 통신에 의한 오버헤드(overhead)가 더 증가한다. [10]

그림 9에는 부분구조가 48개인 경우의 트랜스퓨터 갯수에 따른 총 계산시간과 단위 트랜스퓨터당의 계산효율을 도시하였는데 트랜스퓨터의 갯수가 증가할 수록 총 계산시간이 일반적으로 감소함을 알 수 있다. 그러나 단위 트랜스퓨터의 계산효율은 오히려 매우 감소함을 알 수 있는데, 이 역시 통신에 의한 계산지연의 영향이 매우 크다는 것을 보여주고 있다. 그리고, 트랜스퓨터가 24개 이상인 경우를 보면 8개, 16개인 경우보다 계산속도가 크게 빨라지지 않았음을 볼 수 있는데 이는 전체 절점수가 비교적 크지 않으므로 너무 과도한 부분구조 분할은 계산시간 단축에 도움이 되지 않는다는 것을 알 수 있다. 따라서 병렬시스템의 계산효율을 높이기 위해서는 전체 절점수 중에서 경계절점수가 비교적 작게 배분되도록 분할하거나 경계절점만에 관계한 행렬이 더욱 밀도 있도록(dence matrix) 배치하는 것이 좋으며 또한 앞으로 통신이 매우 빠른 병렬시스템을 구축하면 되겠다.

## 6. 結 論

트랜스퓨터를 사용하여 부분구조법을 병렬화하는 유한요소 알고리즘을 도출하고 사례연구로서 2차원 정자계 문제를 다루었다. 유한요소법에서 생성된 계수행렬은 매우 성기기 때문에 기존의 성긴 행렬 처리기법에 대하여 단위 트랜스퓨터당에서는 매우 좋은 효율을 얻지는 못했으나 전반적인 계산속도의 증가와 기억용량 문제 해소의 효과를 얻을 수 있었다. 또한 총 절점수에 비해 경계절점수가 비교적 작은 경우에 좋은 효율을 얻을 수 있음을 알 수 있었다. 앞으로 통신속도의 문제가 해결되면 더 높은 효율을 얻을 수 있을 것이다. 본 연구에서는 2차원 문제로 그 효용성을 검증하였으나 3차원 문제에도 쉽게 적용이 될 수 있다.

이 방법의 장점은 계산시간이 많이 소요되고 대용량의 메모리가 요구되는 대부분의 문제 해석에 사용될 수 있다는 것이다. 단점으로는 각 트랜스퓨터의 負荷均等을 고려해야 하는데 이를 위해서 요소분할이 된 데이타를 자동으로 부분구조로 분할하는 前處理 過程이 필요하다는 것이다. 그러나 매질이나 구조에 제한받지 않으므로 비교적 수월하다.

## 참 고 문 헌

[1] K. Hwang and F.A. Briggs, "Computer architecture and parallel processing," McGraw Hill, 1985.

[2] C.F. Bryant, M.H. Roberts, and C.W. Trowbridge, "Implementing a boundary integral method on a transputer system," IEEE Trans. on Magnetics, Vol. 26, No. 2, March 1990.

[3] G. Yagawa, N. Soneda and S. Yoshimura,

"A large scale finite element analysis using domain decomposition method on parallel computer," Computers and Structures, Vol. 38, 1991.

[4] K. Preis, G. Vrisk, A. Ziegler, O. Biro, Ch. Magele, W. Renhart, K.R. Richter, "Distributed processing of FEM in a local area network," IEEE Trans. on Magnetics, Vol. 26, No. 2, Mar. 1990.

[5] K.R. Weeber and S.R.H. Hoole, "The subregion method in magnetic field analysis and design optimization," COMPUMAG 8th Conference, July 7-11, 1991, Sorrento, Italy.

[6] J.S. Przemieniecki, "Matrix structural analysis of substructures," Journal of American Institute of Aerospace and Aeronautics, Vol. 1, No. 1, 1963.

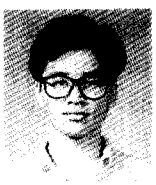
[7] R. Rosen, M. Rubinstein, "Substructure analysis by matrix decomposition," Proc. of the ASCE, Structural Division, Mar. 1970.

[8] G. Yagawa and S. Yoshimura, "有限要素法", 培風館, 1991.

[9] P. Dick and D. May, "A tutorial introduction to Occam programming," BSP Professional Books, London, 1987.

[10] D.P. Bertsekas and J.N. Tsitsiklis, "Parallel and distributed computation," Prentice-Hall, 1989.

## 저 자 소 개



**김근환(金根煥)**

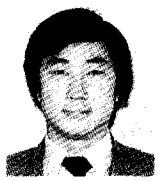
1967년 7월 8일생. 1990년 서울대 공대 전기공학과 졸업. 1992년 동 대학원 전기공학과 졸업(석사). 현재 대우전자 근무.

**정현교(鄭炫敎)**

1955년 8월 17일생. 1979년 서울대 공대 전기공학과 졸업. 1984년 동 대학원 전기공학과 졸업(공박). 1987~89년 뉴욕 Polytechnic Univ. 객원교수. 현재 강원대 공대 전기공학과 부교수. 당학회 편집위원.

**한송엽(韓松曄)**

1939년 3월 14일생. 1963년 서울대 공대 전기공학과 졸업. 1967년 동 대학원 전기공학과 졸업(석사). 1976년~79년 불란서 로렌공과대학원 졸업(공박). 현재 서울대 공대 전기공학과 교수. 당 학회 평의원.



**최 경(崔 炅)**

1958년 4월 30일생. 1981년 서울대 공대 전기공학과 졸업. 1983년 동 대학원 전기공학과 졸업(석사). 1988년 동 대학원 전기공학과 졸업(공박). 현재 강원대 공대 전자공학과 조교수.

**이기식(李基植)**

1952년 3월 20일생. 1973년 서울대 공대 전기공학과 졸업. 1977년 서울대 대학원 전기공학과 졸업. 현재 단국대 공대 전기공학과 교수. 공박