**2096** 대한기계학회논문집(A) 제21권 제12호, pp. 2096~2105, 1997

〈논 문〉

# 평판압연공정 유한요소해석의 분산병렬처리에 관한 연구

권 기 찬* · 윤 성 기**

(1997년 4월 28일 접수)

# Finite Element Analysis of Strip Rolling Process Using Distributive Parallel Algorithm

Kie-Chan Gwon and Sung-Kie Youn

**Key Words:** Parallel Finite Element Method(병렬 유한요소해석), Distributive Processing(분산처리), Strip Rolling(평판압연)

**Abstract**

A parallel approach using a network of engineering workstations is presented for the efficient computation in the elastoplastic analysis of strip rolling process. The domain decomposition mehthod coupled with the frontal solver for elimination of internal degrees of freedom in each subdomain is used. PVM is used for message passing and synchronization between processors. A 2-D plane strain problem and the strip rolling process are analyzed to demonstrate the performance of the algorithm and factors that have a great effect on efficiency are discussed. In spite of much communication time on the network the result illustrates the advantages of this parallel algorithm over its corresponding sequential algorithm.

## 1. 서 론

컴퓨터의 눈부신 발전은 여러 분야에서 기존에는 할 수 없었던 작업을 가능하게 해주고 있으며, 이제는 거의 모든 분야에서 활용되고 있다. 그러나 아직 고성능의 컴퓨터로도 처리하기 힘든 분야가 많으며, 보다 복잡한 구조물에 대한 정교한 해석의 필요성은 더 고성능인 컴퓨터를 기대하게 만든다. 그리고 컴퓨터를 이용한 해석비용은 해석의 정확도를 결정하는 해상도(resolution)의 4제곱에 비례하는 것으로 알려져 있으며, 따라서 계산시간의 절감을 위해서는 보다 빠른 컴퓨터의 사용등 여러 방법이 요구된다. 그러나 프로세서 자체의 속도증가는 거의 한계에 도달했으며 더 이상의 속도증가를 위한 노력보다는 느린 프로세서들을 연결해서 사용하는 방법이 오히려 경쟁력이 있을 수 있다. 이러한 컴퓨터 분야의 환경변화는 각 분야에서 "하나의 큰 작업을 여러 개의 작은 작업으로 나누어 여러 개의 프로세서에서 동시에 처리하게 하는 방법"인 병렬처리법(parallel processing)에 대한 관심을 증가시키고 있으며, 유한요소해석 분야에서도 상당한 연구가 이루어지고 있다.

과거의 대부분의 병렬처리에 관한 연구는 MPP (massively parallel processors)를 사용하여 이루어졌다. MPP는 현재까지 개발된 컴퓨터 중에서 가장 강력한 성능을 발휘하며, 병렬처리 환경의 한 축을 형성한다. 그러나 일반적으로 MPP는 매우 고가의 장비로 이를 모두가 사용하기는 어려운 단점을 가지고 있다. 이에 대한 대안으로 제시되고 있는 것이 컴퓨터들을 망(network)으로 엮어 하나의 작업을 각 컴퓨터에서 나누어 계산하도록 하는

*한국과학기술원 대학원 기계공학과
**회원, 한국과학기술원 기계공학과

것으로 분산처리(distributed computing)라고 한다. 이를 가능케하는 여러 소프트웨어들이 개발되었으며 PVM,[1] MPI[2,3] 등이 이에 해당한다. 분산처리는 컴퓨터간에 자료교환을 위해 많은 시간이 소요되기 때문에 이를 이용한 병렬처리는 상대적으로 비효율적이라 많은 연구가 이루어지지 않았다. 그러나 분산처리가 가지는 큰 장점은 비용으로, 사용자는 비교적 저가의 장비들을 묶어 대형의 문제를 풀 수 있어 적절한 알고리듬의 사용시 큰 효과를 얻을 수 있다고 기대되므로, 본 연구에서는 EWS (engineering workstation) 망을 이용해 분산 병렬처리한다.

병렬 유한요소 알고리듬으로는 강성행렬을 분할해서 각 프로세서에서 계산하는 방법 또는 문제영역을 여러 작은 부영역들로 분할하여 각 프로세서가 할당된 부영역을 계산하여 종합하는 영역분할법(domain decomposition method)[4~7] 등 여러 가지가 제시되고 있다. 강성행렬을 분할하는 방법은 프로세서간에 많은 양의 자료교환을 요구하므로 자료전송시간이 비교적 오래 걸리는 EWS 망을 이용한 분산처리에는 적합하지 않다. 따라서 최근에는 영역분할법의 개념을 이용한 알고리듬이 선호되고 있다. 영역분할법에서는 행렬방정식의 해를 구하기 위한 솔버로 직접솔버와 축차솔버 등이 가능하다. 직접솔버는 유한요소의 순차적 계산에는 장점이 있지만, 병렬처리의 경우 많은 프로세서를 사용할 때 계산시간의 감소에 한계가 있기 때문에 영역분할법에서는 PCG(preconditioned conjugate gradient), Jacobi 법 등과 같은 축차솔버를 사용한 연구[8~10]가 많이 이루어지고 있다. 그러나 축차솔버의 사용은 해의 수치적 안정성에 나쁜 영향을 주며, 분산처리의 경우와 같이 한정된 수의 프로세서만을 사용할때에는 오히려 직접솔버를 사용하는 것이 효과적이다. 이 중에서 프론탈솔버(frontal solver)를 사용하여 부영역 내부의 강성행렬과 하중벡터를 소거하는 것은 가장 기본적인 생각이며, 영역분할 개념과 관련하여 병렬처리의 구현을 용이하게 한다. 또한 이 알고리듬 자체의 특성때문에 영역분할로 인한 각 부영역에서의 선단폭(front-width)의 감소는 계산 효율성의 증가에 상당한 기여를 한다. 이 개념을 이용하여 간단한 탄성문제의 해석을 병렬처리하는 연구[11,12]가 수행되었으나 비선형 문제의 적용은 이루어지지 않았다.

현재까지의 유한요소해석의 병렬처리에 관한 연구는 선형문제에 국한 되어 왔다. 일반적으로 계산시간이 많이 걸리는 소성변형과 같은 비선형 공학문제의 유한요소해석 분야에서는 병렬처리의 필요성이 더욱 큼에도 불구하고 이들 분야에의 적용은 상대적으로 미미한 실정이다. 영역분할 개념을 적용[13,14] 또는 다중격자방법(multigrid method)을 병렬화[15]하여 탄소성 문제를 해석하는 등 여러 방면으로 연구가 진행되고 있다. Ferian 등[16]은 PVM을 사용해 탄소성 문제의 유한요소해석을 EWS 망에서 분산처리하는 연구를 하였으나 순차적 처리에 비해 계산시간이 크게 줄지 않는 결과를 얻었다.

본 연구에서는 평판압연공정의 유한요소해석을 빠르게 처리하기 위해 EWS의 망을 이용하여 분산병렬처리한다. 영역분할법에 위해 병렬처리를 구현하며 각 부영역의 독립적인 계산을 위해 다중 프론탈솔버(multi-frontal solver)를 비선형 문제의 해석을 위한 유한요소 알고리듬에 적용한다. 해석에 소요된 계산시간의 비교를 통해 본 알고리듬을 이용한 분산처리의 효율성을 제시한다.

## 2. 이 론

유한요소해석시 구조물의 강성행렬과 하중벡터의 계산 그리고 이것에 의해 구성된 선형방정식의 해를 구하는 단계에서 계산시간의 대부분이 소모되므로 병렬처리시 가장 중요한 부분이라 할 수 있다. 이 부분의 효과적인 처리를 위해 본 논문에서는 최근 정적인 문제의 병렬알고리듬으로 많이 쓰이는 영역분할법을 이용한다.

### 2.1 병렬처리 수식화

먼저 하나의 전영역을 $n$개의 작은 영역으로 나누었을 때, 부영역 $S^{(j)}$에서의 강성행렬과 하중벡터를 내부자유도와 공유경계상의 자유도로 분리된 형태로 표현하면 다음과 같다.

$$\begin{bmatrix} K_{ss}^{(j)} & K_{sb}^{(j)} \\ K_{bs}^{(j)} & K_{bb}^{(j)} \end{bmatrix} \begin{Bmatrix} U_s^{(j)} \\ U_b^{(j)} \end{Bmatrix} = \begin{Bmatrix} F_s^{(j)} \\ F_b^{(j)} \end{Bmatrix} \quad j=1,2,\cdots, n \tag{1}$$

여기서, 첨자 $s$와 $b$는 각각 부영역의 내부와 공유경계를 의미한다.

$F_b^{(j)}$는 $S^{(j)}$에서 독립적으로 계산된 공유경계에 대한 하중벡터로 다음과 같이 외부에서 이 경계에 작용하는 외력과 내력으로 이루어져있다.

$$F_b^{(j)} = F_{be}^{(j)} + F_{bi}^{(j)} \tag{2}$$

따라서 식 (1)의 선형방정식을 다음과 같이 풀어 쓸수 있다.

$$F_s^{(j)}=K_{ss}^{(j)}U_s^{(j)}+K_{sb}^{(j)}U_b^{(j)} \tag{3}$$

$$F_{be}^{(j)}+F_{bi}^{(j)}=K_{bs}^{(j)}U_s^{(j)}+K_{bb}^{(j)}U_b^{(j)} \tag{4}$$

공유경계에서의 변위의 적합조건은 아래와 같다.

$$U_b^{(j)}=U_b \quad j=1,2,\cdots,n \tag{5}$$

식 (3)을 $U_s^{(j)}$에 대해서 다음처럼 정리될 수 있다.

$$U_s^{(j)}=-K_{ss}^{(j)-1}(K_{sb}^{(j)}U_b^{(j)}-F_s^{(j)}) \tag{6}$$

이 식을 식 (4)에 대입하여 정리하면 다음과 같이 $s^{(j)}$의 공유경계의 자유도로 강성행렬과 하중벡터를 다음과 같이 축약할수 있다.

$$\bar{K}_{bb}{}^{(j)}U_b^{(j)}=\bar{F}_b{}^{(j)} \tag{7a}$$

여기서, $\bar{K}_{bb}{}^{(j)}$와 $\bar{F}_b{}^{(j)}$는 공유경계에서의 축약된 형태의 강성행렬과 하중벡터로서 다음과 같은 식으로 구해진다.

$$\bar{K}_{bb}{}^{(j)}=K_{bb}^{(j)}-K_{bs}^{(j)}K_{ss}^{(j)-1}K_{sb}^{(j)} \tag{7b}$$

$$\bar{F}_b{}^{(j)}=F_{be}^{(j)}+F_{bi}^{(j)}-K_{bs}^{(j)}K_{ss}^{(j)-1}F_s^{(j)} \tag{7c}$$

위 식을 살펴보면 $S^{(j)}$에서 축약된 강성행렬과 하중벡터의 계산에는 다른 부영역의 정보를 필요로 하지 않는다. 따라서 이 부분의 계산은 각 부영역별로 독립적으로 처리될 수 있다.

각 부영역의 축약된 형태의 강성행렬과 하중벡터를 결합하면 아래식처럼 전영역의 공유경계 자유도만의 선형방정식을 구할수 있다.

$$K_BU_B=F_B \tag{8a}$$

$$K_B=\sum_{j=1}^{n}\bar{K}_{bb}{}^{(j)} \tag{8b}$$

$$F_B=\sum_{j=1}^{n}\bar{F}_b{}^{(j)} \tag{8c}$$

이 과정은 각 부영역이 할당된 프로세서로부터 수집되어 이루어지며, 이 선형방정식의 해는 한 프로세서에서 순차적으로 계산한다. 각 부영역을 담당하는 각 프로세서는 이 공유경계의 해를 전달받아 내부자유도에 대한 해를 식 (6)에 의해 구한다.

강성행렬과 하중벡터의 내부자유도에 대한 성분을 소거 및 복구하기 위해 본 논문에서는 프론탈 솔버를 사용한다. 먼저 전영역을 여러개의 부영역으로 분할한 다음 각 부영역에서 프론트웨이브(front-wave)를 진행시키면서 내부의 절점에 대한 자유도를 소거시킨다. 이렇게 각 부영역에서 독립적으로 진행된 프론트웨이브는 공유경계에서 만나게 되며, 이 공유경계의 강성행렬과 하중벡터를 결합해 공유경계 자유도에 대한 해를 구한다. 이 공유경계의 해를 이용해 프론트웨이브가 진행해 온 반대방향으로 내부자유도에 대한 해를 복구한다.

### 2.2 탄소성 유한요소해석에의 적용

영역분할법을 이용한 탄소성 유한요소해석의 병렬처리 과정은 다음과 같은 단계로 나눌 수 있다.

(1) 전처리 단계 : 전 영역에 대한 정보입력
(2) 각 부영역에 대한 강성행렬과 하중벡터 구성, 각 부영역에서의 내부자유도 축약
(3) 공유경계 자유도에 대한 축약 선형방정식 구성 및 해 계산
(4) 각 부영역의 해 계산
(5) 후처리 단계

각 단계중 병렬처리의 대상이 되는 것은 단계 (2)와 단계 (4)이며, 단계 (1), (3), (5)는 주프로세서에서 담당한다. 단계 (3)의 병렬처리방법이 몇가지 제시되고 있으나[12], 부가적인 정보교환을 요구하므로 정보교환시간이 상대적으로 큰 분산처리에는 적합하지 않다.

각 단계를 간략히 설명하면 다음과 같다.

단계 (1)에서는 전체 구조물에 대한 정보 및 각 부프로세서에 대한 제어 등 전 영역에 관련된 정보를 처리한다. 여기서 처리하는 정보로는 전체 절점수, 전체 경계절점수, 전체 부영역수, 입출력에 관련된 정보, 각 축차과정에 대한 수렴조건, 부프로세서에 대한 정보 등이다. 주 프로세서는 이들 정보를 입력받아 다시 각 부프로세서에 전달한다.

단계 (2)에서는 각 부프로세스의 생성과 부영역에 대한 강성행렬과 하중벡터의 구성 및 축약이 수행된다. 이 때 각 부영역의 강성행렬 및 하중벡터가 공유경계의 자유도를 제외하면 서로 연계되지 않은 블록화된 형태를 가지므로 프로세서간의 정보교환이 없이 서로 독립적으로 계산을 수행할 수 있다. 따라서 본 논문에서는 하나의 프로세서가 하나의 부구조를 담당하여 작업을 수행하도록 구성하였다. 주 프로세서는 각 부프로세서의 생성 및 제어를 담당하며, 해석이 끝난 부프로세서로부터의 해석결과를 수집한다. 그리고 주프로세서도 하나의 부영역을 담당하여 해석을 수행한다. 여기서 부프로세서로부터의 정보입력은 주프로세서가 담당하는

작업이 끝난 후 수행되므로 주프로세서와 부프로세서간의 동기화가 필요하며, 작업량의 적당한 분배(load balancing)가 이루어져야 한다. 작업량의 분배가 적당하지 않을 경우 작업을 하지 않는 프로세서가 발생하며, 이는 전체의 효율성을 감소시키게 된다. 부프로세서에서 필요한 정보는 주프로세서와 자체 입력파일에서 제공된다.

단계 (3)에서는 각 부프로세서로부터의 결과를 전송받아 공유경계의 자유도에 대한 축약된 선형방정식을 구성하고, 이것의 해를 구한다. 그리고 구해진 공유경계의 해중 각 부영역에 해당하는 부분을 이를 담당하는 부프로세서로 전송한다. 이 단계의 작업은 주 프로세서가 담당하여 수행한다.

단계 (4)에서는 주프로세서로부터 관련되는 공유경계 자유도에 대한 해를 전달받은 각 부프로세서는 이를 이용해 내부자유도에 대한 해를 복구한다. 이 작업도 다른 부영역에 대한 정보를 필요로 하지 않으므로 서로 독립적으로 수행할 수 있다. 주프로세서는 각 부프로세서에 대한 정보제공 및 제어와 각 부프로세서로부터의 해석결과를 수집하여 해의 수렴여부를 결정한다.

단계 (5)에서는 해석이 끝났을 때 주프로세서가 각 부프로세서로부터 수집된 결과를 종합하여 해석자가 원하는 결과를 출력한다.

소성변형 문제는 구성방정식이 비선형이므로 Newton-Raphson법을 사용해 한 하중단계에서의 해를 수렴시키므로, Fig. 1이 보여주듯이 단계 (2)에서 단계 (4)는 해석이 끝날 때까지 반복된다. 그리고 소성변형의 해석시 응력상태를 구하기 위한 구성방정식의 적분과정에서 더 많은 계산량을 요구하므로, 본 알고리듬을 탄소성 문제에 적용하면 탄성문제를 병렬처리하는 경우보다 큰 효율을 얻을 수 있다.

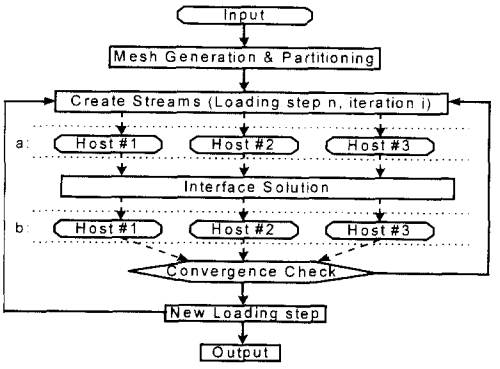



**Fig. 1** Flow diagram of parallel algorithm

### 2.3 평판압연공정 해석모델

탄소성 모델을 가정하여 평판압연공정에서의 소재의 변형을 해석할 때, 평형방정식과 유동법칙을 사용한 구성방정식 그리고 적합방정식을 전영역에 적용하고 경계조건을 부가하기 위해 다음과 같은 가상일의 원리를 이용한다.

$$\int_V \sigma_{ij}\delta\varepsilon_{ij}dV - \int_{S_T} T_i\delta u_i dS = 0 \tag{9}$$

위 식에서 $\sigma_{ij}$와 $\varepsilon_{ij}$는 각각 응력텐서와 변형률텐서의 성분을 나타내며, $T_i$와 $\delta u_i$는 각각 응력이 부가된 경계에서의 응력벡터의 성분과 변위벡터의 성분의 증분을 의미한다.

압연공정은 소재의 대변형을 유발하고, 탄소성 구성방정식이 증분식의 형태로 표현되므로, 이 문제를 유한요소 수식화하기 위해 오일러리안(Eulerian) 공식화를 사용하게 되며, 요소의 각 적분점에서 응력을 구하기 위한 구성방정식의 적분은 일반화된 미드포인트 룰(generalized midpoint rule)을 적용한다. 소재의 소성변형시 발생하는 비압축성의 제한조건이 과도하게 부가되지 않기 위해 선택 감차적분을 사용하여 한 요소내에서 평형방정식과 제한조건의 수의 비를 일치시킨다.[17]

재료의 경화를 고려하여 해석하기 위해 등방경화를 가정하며 선형 등방경화와 다음과 같이 표현되는 Ludwick-type의 모델을 사용한다.

$$\bar{\sigma} = Y_o(1+\frac{\bar{\varepsilon}}{b})^n \tag{10}$$

여기서, $Y_o$는 초기의 항복응력이고, $b$와 $n$은 실험에 의해 정해지는 각 재료의 상수이다.

롤과 소재사이의 접촉과 마찰을 처리하기 위해 접촉면에서 소재의 표면에 가상요소를 생성해 롤의 회전에 따라 이 요소가 전단변형을 하게 하여 소재에 마찰력과 같은 전단력을 작용하도록 하는 방법[18~20]을 사용하였다.

## 3. 해석결과 및 검토

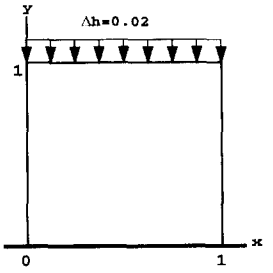
본 연구에서는 앞서 기술한 병렬처리 알고리듬을 소성문제에 적용해 유한요소해석할 때 계산시간의 효율성을 확인하기 위해 선택된 간단한 이차원 평면문제와 평판압연공정을 해석하였다. 계산에 사용된 장비는 시스템공학연구소 내에 설치된 Silicon Graphics사의 INDY(MIPS R4000, 100 MHz) 기종

5대와 INDIGO2(MIPS R4400, 150 MHz)기종 1대이다. 프로세서들간의 자료교환과 동기화를 위해 PVM을 사용한다.

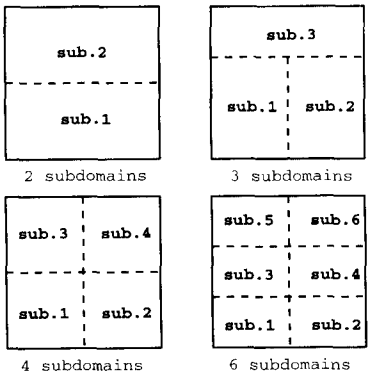
병렬처리의 효율성을 판단하기 위한 척도로 아래와 같이 정의되는 속도증가(seedup)와 효율(efficiency)을 사용하였다.

$$S_p(Speedup)=\frac{T_1}{T_n}$$

$$E_f(Efficiency)=\frac{Speedup}{n}=\frac{T_1}{(n*T_n)}$$




(a) Shape and boundary condition




(b) Decomposition of the structure

**Fig. 2** Numerical example

여기서,

$T_1$ : 1개의 프로세서로 계산했을 때 걸린 시간

$T_n$ : $n$개의 프로세서로 계산했을 때 걸린 시간

일반적으로 $0 \le E_f \le 1$이 되며, 이상적인 경우에 1이 되지만, 실제에 있어서는 프로세서간의 정보교환, 동기화를 위한 대기시간 그리고 알고리듬의 순차적 종속성(sequential dependency) 등으로 인해 그 값이 감소한다.

### 3.1 정사각형 형상의 소성변형 문제

먼저 본 알고리듬을 탄소성 문제에 적용해 분산병렬처리할때의 효율을 알아보기 위해 정사각형 형상을 갖는 문제를 요소와 절점의 수를 세가지 경우로 변화시키면서 유한요소해석하였다. Fig. 2(a)는 문제의 형상과 경계조건을 보여준다.

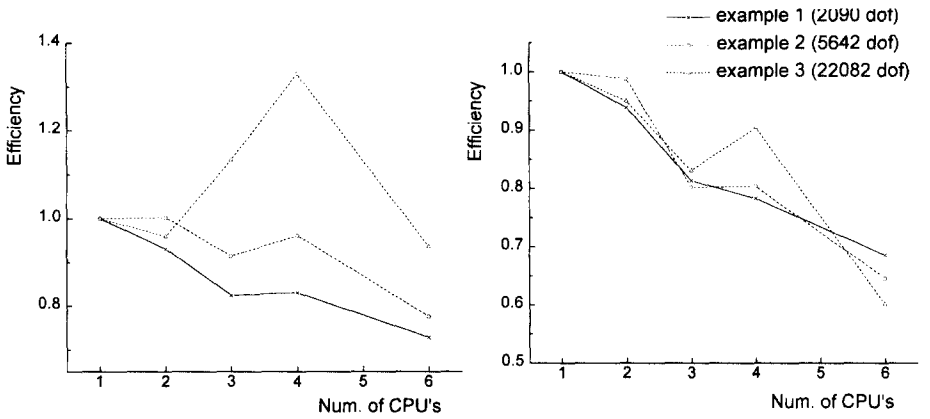
병렬처리를 위해 주어진 문제의 전영역을 2개, 3개, 4개, 6개의 부영역으로 분할하고 한 프로세서에 하나의 부영역만을 할당하여 처리하였다. Fig. 2(b)는 각 경우에 전영역을 여러 개의 부영역으로 분할한 방법을 보여주며, 각 프로세서에 일정한 작업량을 할당하기 위해 부영역 모두가 같은 크기를 갖도록 분할하였다.

Table 1은 각 예제에 대해 한 프로세서로 계산한 경우와 프로세서의 수를 증가시면서 병렬처리하였을 때의 총 계산시간(elapsed time)과 대기시간(idle time)을 보여준다. 총 계산시간에서 대기시간을 빼면 CPU시간(CPU time)이 얻어진다. 각 예제의 경우 프로세서의 수가 증가하면서 계산시간이 줄어듬을 볼수 있다.

Fig. 3(a)는 각 예제에 대해 프로세서 개수의 증가에 따른 효율을 보여준다. 예제 2를 두개의 프로세서를 이용해 해석했을 때와 예제 3을 세개와 네

**Table 1** Computation times(sec.) of elastoplastic analyses (numerical example)

| | Example 1(2090 dof) | | Example 2(5642 dof) | | Example 3(22082 dof) | |
|---|---|---|---|---|---|---|
| | Elapsed time | Idie time | Elapsed time | Idle time | Elapsed time | Idle time |
| 1 CPU | 110.83 | | 396.07 | | | |
| 2 CPU | 59.56 | 1.30 | 197.54 | 3.91 | 1828.81 | 74.15 |
| 3 CPU | 44.80 | 7.61 | 144.36 | 24.47 | 1032.57 | 211.33 |
| 4 CPU | 33.37 | 5.91 | 103.07 | 16.00 | 660.03 | 97.95 |
| 6 CPU | 25.35 | 5.09 | 85.03 | 19.19 | 624.83 | 201.72 |




(a) Sequential computing (b) Substructuring on one processor

**Fig. 3** Relation between efficiency and the number of processors-evaluated by

개의 프로세서로 해석하였을 때의 효율이 1보다 크다. 예제 3의 경우 한 프로세서만을 사용했을 때 주메모리의 부족으로 계산을 하지 못했기 때문에, 두 프로세서를 사용한 경우의 CPU 시간을 사용하여 계산하였다. 이것이 1이상의 효율을 야기시킨 하나의 원인이라 생각되지만, 아래에 설명될 다른 원인이 그보다 많은 영향을 미치고 있다. 앞에서 설명했듯이 효율이 1인 것은 이상적인 경우를 의미하며 실제의 경우에는 1보다 작게 된다. 여기서 1보다 큰 효율을 얻은 것은 선택된 예제의 특성과 본 연구에서 내부자유도의 축약을 위해 사용한 프론탈솔버의 특성에 기인한다. 일반적으로 프론탈솔버를 사용할 때 계산시간은 선단 폭의 평균제곱근의 제곱에 비례하는 것으로 알려져 있다.[21] 본 연구에서 선택한 예제의 형상이 정사각형이므로 전영역을 분할할 때 부영역의 선단 폭이 프로세서의 증가에 따라 매우 감소하게 된다. 따라서 병렬처리시 모든 부프로세서에서의 계산시간의 합이 한 프로세서에서 전영역을 순차적으로 처리한 계산시간보다 작게 된다. 따라서 공유경계의 자유도의 수가 작은 경우 이러한 특성에 의해 감소된 계산시간은 프로세서간의 정보교환시간과 대기시간을 보상할 수 있다. 프로세서의 수가 증가하여 공유경계 자유도수가 많아지면 효율은 다시 감소한다. 따라서 보다 나은 효율을 얻기 위해서는 공유경계의 자유도수를 줄임과 동시에 전영역의 선단 폭에 대해 각 부영역의 선단 폭이 작아지도록 영역을 분할해야 한다.

전영역보다 감소된 선단 폭을 갖는 부영역이 계산시간에 미치는 영향을 알아보기 위해, 영역분할 후 한 프로세서에서 모든 부프로그램을 처리하여 병렬처리의 경우와 같은 계산과정을 수행하였다. Table 2는 이때의 총 계산시간을 보여준다. 모든 경우에 부영역의 수가 증가함에 따라 계산시간이 감소함을 볼 수 있다. Fig. 3(b)는 이 계산시간을 기준으로 구한 효율을 보여준다. 동일한 계산과정을 한 프로세서에서 수행한 시간에 대해 병렬처리 효율을 계산하였을 때 모든 경우 효율은 1보다 작음을 볼 수 있다. 그리고 프로세서의 수가 증가함에 따라 효율은 점차로 감소한다.

**Table 2** Execution times(sec.) of substructuring on one processor

| | Example 1 | Example 2 | Example 3 |
|---|---|---|---|
| 2 subdomain | 111.86 | 390.61 | 3475.36 |
| 3 subdomain | 109.35 | 347.33 | 2572.69 |
| 4 subdomain | 104.51 | 331.22 | 2386.22 |
| 6 subdomain | 104.29 | 329.41 | 2253.96 |

### 3.2 평판압연공정 해석의 병렬처리

평판압연공정의 유한요소해석을 위해 세가지 문제를 채택하여 계산하였고 이를 프론탈솔버를 이용한 영역분할법으로 병렬처리하였다. 채택된 문제는 실험결과[22]와 비교할 수 있는 경우로 압연전과 압연후의 두께 감소비에 따라 14.17%, 14.18%, 22.76%로 나누었다. Fig. 4는 각 경우에 대해 정상상태에 도달했을때 소재에 작용하는 수직압력을 본 연구에서의 유한요소해석 결과와 실험결과를 비

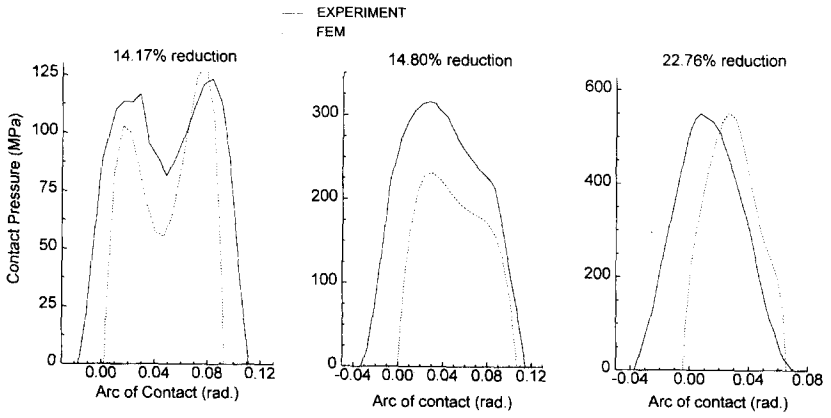



**Fig. 4** Normal pressure on contact surface of finite element analyses and experiment (22)

**Table 3** Computation times(sec.) for strip rolling analyses

| | 14.17% Rolling(4212 dof) | | 14.80% Rolling(4264 dof) | | 22.76% Rolling(10634 dof) | |
|---|---|---|---|---|---|---|
| | Elapsed time | Idie time | Elapsed time | Idle time | Elapsed time | Idle time |
| 1 CPU | 5936.44 | | 8945.29 | | 17428.20 | |
| 2 CPU | 3641.72 | 482.48 | 5514.78 | 579.08 | 9270.91 | 19.13 |
| 3 CPU | 2960.76 | 770.33 | 4428.89 | 1124.31 | 8379.03 | 2245.89 |
| 4 CPU | 2245.06 | 542.45 | 3352.37 | 826.51 | 6729.02 | 1947.29 |
| 5 CPU | 1835.45 | 403.27 | 2746.75 | 635.57 | 5153.53 | 1247.28 |
| 6 CPU | 1545.79 | 317.84 | 2414.55 | 551.55 | 4585.25 | 1221.68 |

교한 것이다.

이 압연공정의 유한요소해석의 병렬처리를 위해 전영역을 2개부터 6개의 부영역으로 분할하고, 한 프로세서에 하나의 부영역만을 할당하여 처리하였다. 전영역의 형상이 한쪽으로 긴 직사각형의 모양이기 때문에 긴 방향으로 계속 나누는 방법으로 영역을 분할하였다. 그리고 각 프로세서에 같은 작업량을 할당하기 위해 한 부영역에서의 자유도수와 요소수는 같게 하였다.

Table 3은 각 문제에 대해 한 프로세서로 계산한 경우와 프로세서의 수를 증가시키면서 병렬처리하였을 때의 계산시간을 보여준다. Fig. 5는 각 예제에 대해 프로세서의 개수의 증가에 따른 속도증가와 효율을 총 계산시간을 사용해 계산한 결과를 보여준다.

각 경우에 대해 프로세서의 수가 증가하면서 속도증가가 커졌음을 볼 수 있다. 정사각형 예제와 달리 모든 경우 효율이 1보다 작다. 이 문제의 경우 전영역의 형상이 길쭉한 직사각형이고, 모든 경우에 이 방향으로 영역을 분할하였으므로 전영역과 부영역 사이에 선단 폭 크기의 차이가 없게 된다. 따라서 이 문제는 부영역에서의 선단 폭의 감소로 인한 부프로세서의 계산시간의 감소를 기대할 수 없다. 각 경우에 여섯 개의 프로세서를 사용했을 때의 속도증가를 살펴보면 두께 감소비가 14.17%인 경우 3.84, 14.80%인 경우 3.70, 22.76%인 경우 3.80으로 모두 비슷한 값을 보인다. 모든 경우에 프로세서의 수가 증가하면서 효율이 떨어짐을 볼 수 있다. 여섯 개의 프로세서를 사용할 때 각각의 효율을 보면 차례로 0.64, 0.62, 0.63의 값을 보인다.

Fig. 6은 총 계산시간에서 대기시간을 제외시킨 CPU시간을 사용해 구한 병렬처리시의 속도증가와 효율을 나타낸다. 본 연구에서 병렬 컴퓨터환경으로 사용한 EWS 망의 경우 다른 사용자의 작업을 제어할 수 없었다. 따라서 수록된 대기시간은 프로

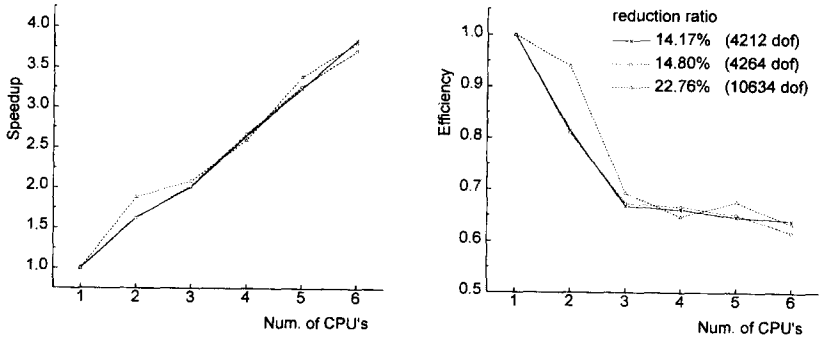



**Fig. 5** Speedup and efficienty (reference time : elapsed time)

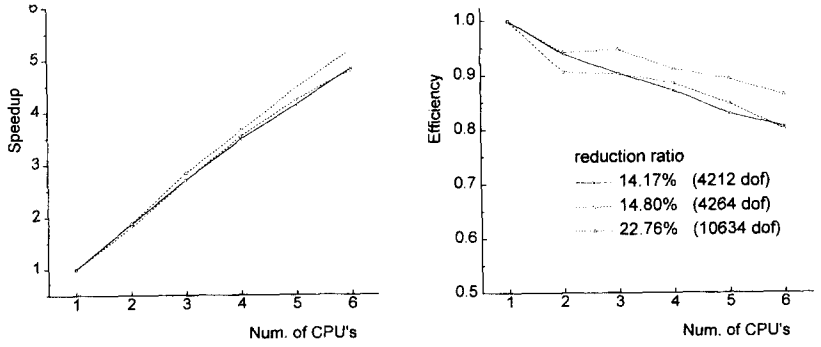



**Fig. 6** Speedup and efficiency (reference time : CPU time)

세서간의 정보교환 시간외에 다른 사용자의 사용에 의해 야기된 프로세서사이의 속도차이에 기인한 대기시간이 포함되어 있다. 따라서 이를 제외시킨 경우의 속도증가와 효율로 이상적인 병렬 컴퓨터 환경에서의 본 알고리듬의 성능을 추측할 수 있다. 이 경우 프로세서 증가에 따른 효율의 감소가 비교적 적음을 볼수 있다. 각 문제에 대해 여섯 개의 프로세서를 사용하였을 때의 효율을 살펴보면 차례로 0.81, 0.80, 0.86이다. 따라서 이상적인 병렬 컴퓨터 환경하에서 효율을 0.17에서 0.23까지 올릴 수 있다고 생각된다. 비교적 전영역의 자유도의 개수가 비슷한 14.17%의 경우와 14.80%의 경우 비슷한 효율을 보인 반면 상대적으로 자유도수가 많은 22.76%의 경우 높은 효율이 얻어졌다.

본 연구에서는 공유경계의 자유도에 대한 해는 주 프로세서에서 순차적으로 계산하였다. Fig. 7은 공유경계에서의 자유도수와 병렬처리시 이 공유경계에서의 해를 구하기 위해 소비된 계산시간의 관계를 보여준다. 공유경계의 자유도수가 증가할수록, 즉 프로세서의 수가 증가할수록 여기에 해당하

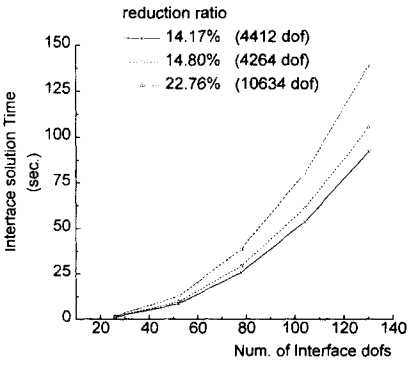



**Fig. 7** Relation between interface solution time and the number of interface dofs

는 해를 구하는데 걸리는 시간이 선형비례보다 큼을 알수 있다. 따라서 이부분의 순차적 처리가 프로세서의 수를 증가시켰을 때 효율을 감소시키는 중요한 원인중 하나임을 알 수 있다.

## 4. 결 론

본 논문에서는 평판압연공정의 탄소성 유한요소 해석의 계산시간을 효과적으로 줄이기 위해 EWS의 망을 이용해 분산병렬처리 하였다.

병렬처리방법으로 영역분할법을 사용하였으며 각 부영역간의 공유경계에서 문제의 축약된 형태를 얻기 위해 프론탈솔버를 사용하였다. 공유경계에 대한 해의 계산은 한 프로세서에서 순차적으로 처리하였다.

구성된 병렬처리 알고리듬의 일반적 특성을 이해하기 위해 정사각형 모양의 간단한 이차원 문제를 해석하였다. 문제의 자유도수와 프로세서의 수를 증가시키면서 병렬처리한 결과 본 알고리듬의 경우 영역분할 후 각 부영역의 선단 폭에 감소에 의한 결과로 효율이 상당히 커지는 것을 보여주었다. 많은 프로세서를 사용할 경우 이러한 효과는 사라지며 점차로 효율은 감소하게 된다.

평판압연공정 해석의 타당성을 보이기 위해 실험결과와 비교할 수 있는 문제들을 선택하여 해석한 후 구성된 병렬처리 알고리듬을 적용하여 해석하였다. 이 경우 각 부영역이 전영역의 선단 폭과 일치하도록 영역을 분할하였다. 프로세서 수의 증가에 따라 효율은 점차로 감소하는 경향을 보였으며, 이는 사용된 프로세서의 수에 비례하여 공유경계의 자유도가 증가하였기 때문에, 프로세서간에 교환될 자료량이 많아지게 되어 정보교환에 많은 시간을 소비하게 되었고 순차적 종속성을 갖는 공유경계에서의 해를 구하는데 또한 많은 시간이 사용되었기 때문이다. 자유도수가 많은 문제의 경우 다른 문제보다 높은 효율을 보였다.

병렬처리시 효율을 크게 좌우하는 것은 각 프로세서간의 작업량 균등분배이다. 각 프로세서에 걸리는 작업량의 변화에 따라 효율이 많이 변화하며, 대기시간이 클수록 효율은 저하된다. 대기시간을 줄이기 위해서는 각 프로세서에서 수행되는 작업에 대한 계속적인 점검을 수행하고 다음 단계에서 이를 고려하여 각 프로세서에 분배할 작업량을 결정함으로써 프로세서가 대기하지 않도록 해야한다.

앞에서 언급한 몇 가지 문제점에도 불구하고 병렬처리의 결과 모든 경우 만족할 만한 효율을 얻을 수 있었다. 따라서 소성문제에 프론탈솔버를 사용한 영역분할법으로 병렬처리하는 것은 경쟁성이 있다고 생각되며 전영역의 분할시 각 부영역에서의 선단 폭, 공유경계의 수 그리고 작업량 균등분배를 고려한다면 더 높은 효율을 얻을 것으로 기대된다.

## 참고문헌

(1) Giest, A. and Begulin, A. et al., 1994, "Parallel Virtual Machine, A Users Guide and Tutorial for Networked Parallel Computing.," MIT press.

(2) Message Interface Forum, 1995, MPI : A Message-Passing Interface Standard, University of Tennessee.

(3) GDB/RBD, 1996, MPI Primer/Developing with LAM, Ohio State University.

(4) Storaasli, O. and Bergan, P., 1987, "Nonlinear substructuring Method for Concurrent Processing Computers," *AIAA Journal*, Vol. 25, No. 6, pp. 871~876.

(5) Chiang, K. N. and Fulton, R. E., 1990, "Concepts and Implementation of Parallel Finite Element Analysis," *Comput. Struct.*, Vol. 36, No. 6, pp. 1039~1046.

(6) Yagawa, G., Soneda N. and Yoshimura, S., 1991, "A Large Scale Finite Element Analysis Using Domain Decomposition Method on a Par allel Computer," *Comput. Struct.*, Vol. 38, No. 5/6, pp. 615~625.

(7) Adeli, H. and Kamal, O., 1992, "Concurrent Analysis of Large Structures- I . Algorithms," *Comput. Struct.*, Vol. 42, No. 3, pp. 413~424.

(8) Law, K. H., 1986, "A Parallel Finite Element Solution Method," *Comput. Struct.*, Vol. 23, No. 6, pp. 845~858.

(9) Carter, W. T., Sham, T. L. and Law, K. H., 1989, "A Parallel Finite Element Method and Its Prototype Implementation on a Hypercube," *Comput. Struct.*, Vol. 31, No. 6, pp. 921~934.

(10) Oh, R. and Fulton, R. E., 1988, "An Investigation of Parallel Numerical Integration Methods for Nonlinear Dynamics," *Comput. Struct.*, Vol. 30, No. 1/2, pp. 403~409.

(11) Zhang W. and Lui, E. M. 1991, "A Parallel Frontal Solver on the Alliant FX/80," *Comput. Struct.*, Vol. 38, pp. 203~215.

(12) Rao, A., Loganathan K. and Raman, N., 1994, "Multi-frontal Based Approach for Concurrent Finite Element Analysis," *Comput. Struct.*, Vol. 52, pp. 841~846.

(13) Badea, L. and Gilormini, P., 1994, "Application of a Domain Decomposition Method to Elastoplastic Problem," *Int. J. Solids Structures*, Vol. 31, No. 5, pp. 643~656.

(14) Ning Hu, 1994, "A Parallel Algorithm for Analyzing Elasto-plastic Problems," *Comput. Struct.*, Vol. 52, pp. 1127~1133.

(15) Kacou S. and Parsons, I. D. 1993, "A Parallel Multigrid Method for History-dependent Elasto-plasticity Computations," *Comput. Methods Appl. Mech. Engrg.*, Vol. 108, pp. 1~21.

(16) Ferian, A. Franchi A. and Genna, F., 1996, "An Incremental Elastic-plastic Finite Element Solver in a Workstation Cluster Environment-Part Ⅱ.-Performance of a First Implementation," *Comput. Methods Appl. Mech. Engrg.*, Vol. 130, pp. 299~318.

(17) Nagtegall, J. C., Parks, D. M. and Rice, J. R., 1974, "On Numerically Accurate Finite Element Solutions in the Fully Plastic Range," *Comput. Methods Appl. Mech. Engrg.*, Vol. 4, pp. 153~177.

(18) Hartley, P. Sturgess C. and Rowe, G., 1979, "Friction in Finite-element Analyses of Metal-forming Processes," *Int. J. Mech. Sci.*, Vol. 21, pp. 301~311.

(19) Liu, G. and Hartley, P. et al., 1985, "Elastic-plastic Finite-element Modelling of Cold Rolling of Strip," *Int. J. Mech. Sci.*, Vol. 27, No. 7/8, pp. 531~541.

(20) Liu, G. and Hartley, P. et al., 1985, "Simulation of the Cold Rolling of Strip Using an Elastic-plastic Finite Element Technique," *Int. J. Mech. Sci.*, Vol. 27, No. 11/12, pp. 829~839.

(21) Gordon C. Everstine, 1979, "A Comparison of Three Resquencing Algorithms for the Reduction of Matrix Profile and Wavefront," *Int. J. Num. Methods Engrg.*, Vol. 14, pp. 837~853.

(22) Al-Salehi, F. A. R., Firbank T. C. and Lancaster, P. R., 1973, "An Experimental Determination of the Roll Distributions in Cold Rolling," *Int. J. Mech. Sci.*, Vol. 15, pp. 693~710.