〔논문〕 한국소성가공학회지(1998), 제7권 제5호
J. of the Korean Society for Technology of Plasticity Vol. 7, No. 5, (1998)

부구조법에 의한 영역 분할 및 강소성 유한요소해석의 병렬 계산

박 근 * · 양동열*

(1998년 6월 15일 접수)

Domain Decomposition using Substructuring Method and Parallel Computation of the Rigid-Plastic Finite Element Analysis

Keun Park and Dong-Yol Yang

Abstract

In the present study, a domain decomposition scheme using the substructuring method is developed for the computational efficiency of the finite element analysis of metal forming processes. In order to avoid calculation of an inverse matrix during the substructuring procedure, the modified Cholesky decomposition method is implemented. As obtaining the data independence by the substructuring method, the program is easily parallelized using the Parallel Virtual Machine(PVM) library on a work-station cluster connected on networks. A numerical example for a simple upsetting is calculated and the speed-up ratio with respect to various number of subdomains and number of processors. The efficiency of the parallel computation is discussed by comparing the results.

Key Words : Rigid-Plastic Finite Element Analysis(강소성 유한요소해석), Substructuring Method(부구조법), Domain Decomposition(영역 분할), Parallell Computation(병렬 계산)

1. 서 론

많은 계산량이 요구되는 대형 구조물의 유한요소해석의 경우 계산 시간이 많이 소요되고, 혹은 계산량이 컴퓨터의 용량을 초과하게 되어 계산 자체가 불가능하게 되는 경우도 있다. 이러한 경우 전체 영역을 몇 개의 부영역(sub-domain)으로 분할하여 부영역 내의 자유도를 독립적으로 응축(condensation)시킴으로써 계산량을 감소시킬 수 있는데, 이러한 기법이 부구조법(substructuring method)이라 불린다.

부구조법은 1960년대에 들어 당시의 컴퓨터 용량으로는 계산이 불가능했던 대형 구조물의 해석을 위해 제안되었고,[1] 1970년대에 들어 실제 문제의 해석에 본격적으로 적용되었으며, 당시 널리 사용되던 상용 소프트웨어인 NASTRAN도 이를 채택하였다.[2,3] 또한 부구조법의 적용에 있어서 필요한 정적응축(static condensation)을 효

* 한국과학기술원 기계공학과

과적으로 수행하기 위해 Cholesky 분할의 적용에 관한 연구가 진행되었고,[4-7] 부구조법에 의해 분할된 영역에 대한 강성행렬을 재분할함으로써 계산량을 한층더 감소시키는 반복적 부구조법(recursive substructuring)이 제안되었다.[8,9] 한편 1980년대 중반에 들어 실질적인 용도에서의 병렬처리 컴퓨터의 개발과 더불어 유한요소해석에서의 병렬처리(parallel processing) 기법에 관하여 연구가 활발히 진행되었는데, 이를 위한 영역 분할(domain decomposition)의 일환으로서 부구조법이 적용된바 있다.[10,11]

강소성 유한요소해석은 비선형 해석이므로 반복계산(iteration)을 수행해야 한다. 따라서 계산 시간이 많이 소요되게 되는데, 3차원 벌크(bulk) 성형해석의 경우 엄청난 계산량으로 인해 일반 워크스테이션으로는 해석이 불가능하여 슈퍼컴퓨터를 사용하는 경우가 많아 적용의 범위가 넓지 못한 실정이다. 본 논문에서는 이러한 문제점을 해소하기 위해 강소성 유한요소해석에 부구조법을 적용함으로써 계산 시간을 감소시키고, 또 계산용량을 줄일 수 있는 방안에 대해 검토해보고자 한다. 또한 이를 바탕으로 해석 소프트웨어의 병렬화를 구축함으로써 계산효율의 극대화를 추구하고자 한다.

2. 강소성 유한요소법의 수식화

체적이 V이고 접촉력 f_i가 정의된 표면 S_f와 속도 v가 정의된 표면 S_v로 둘러싸인 강소성체의 평형방정식은 체적력(body force)을 무시하면 다음과 같이 주어진다.

$$\sigma_{ij,j} + f_i = 0 \tag{1}$$

윗식에 변분정리(variational principle)를 적용하고 이를 정리하면 다음과 같은 변분방정식을 얻을 수 있다.[12] 여기서 K는 비압축성조건을 부과하기 위한 벌칙상수(penalty constant)로 매우 큰 양수이다.

$$\int_V \bar{\sigma}\delta\dot{\bar{\varepsilon}}dV + K\int_V \dot{\varepsilon}_v\delta\dot{\varepsilon}_v dV - \int_{S_f} f_i\delta v_i dS = 0 \tag{2}$$

여기서 $\bar{\sigma}$, $\dot{\bar{\varepsilon}}$ 는 각각 유효응력과 유효변형율 속도를 의미

$$\bar{\sigma} = \sqrt{\frac{3}{2}\sigma'_{ij}\sigma'_{ij}}\,, \quad \dot{\bar{\varepsilon}} = \sqrt{\frac{2}{3}\,\dot{\varepsilon}_{ij}\,\dot{\varepsilon}_{ij}} \tag{3}$$

한편 금형과 재료사이의 접촉면에서 작용하는 마찰력을 표현하기 위해서 다음과 같은 식을 사용하였다.[13]

$$\boldsymbol{f} = -\frac{2}{\pi}mk\,\tan\left(\frac{|\boldsymbol{v}_s|}{u_0}\right)\boldsymbol{t} \tag{4}$$

여기서 m은 마찰계수(friction factor), k는 해당 접촉점에 있는 재료의 국부적 전단항복응력(shear yield stress), u_0는 $|\boldsymbol{v}_s|$에 비해서 매우 작은 양의 상수값이며, $\boldsymbol{v}_s$는 재료와 금형간의 상대속도의 벡터이고, $\boldsymbol{t}$는 $\boldsymbol{v}_s$방향의 단위벡터(unit vector)이다.

3. 부구조법에 의한 영역 분할

3.1 부구조법의 기본 수식화

앞장에서 얻어진 변분방정식 (2)식을 유한요소 수식화하여 얻어지는 행렬방정식($\mathbf{Ku} = \mathbf{f}$)을 푸는 과정에 있어서 각 부영역에 대해 내부의 자유도(internal degree of freedom)와 경계상의 자유도(boundary degree of freedom)에 관련된 방정식으로 구분하여 정리하면 다음과 같다.

$$\begin{bmatrix}\mathbf{K}_{ii} & \mathbf{K}_{ib}\\ \mathbf{K}_{bi} & \mathbf{K}_{bb}\end{bmatrix}\begin{Bmatrix}\mathbf{u}_i\\ \mathbf{u}_b\end{Bmatrix} = \begin{Bmatrix}\mathbf{f}_i\\ \mathbf{f}_b\end{Bmatrix} \tag{5}$$

여기서 첨자 i는 부영역의 내부를, 첨자 b는 부영역간의 경계를 의미한다. 윗 식을 전개하여 정적응축 기법에 의하여 내부의 변위를 소거하고 경계상의 변위만으로 표시하면 다음과 같이 표현될 수 있다.

$$\mathbf{K}^*\mathbf{u}_b = \mathbf{f}^* \tag{6}$$

여기서

$$\mathbf{K}^* = \mathbf{K}_{bb} - \mathbf{K}_{bi}\mathbf{K}_{ii}^{-1}\mathbf{K}_{ib} \tag{7}$$

$$\mathbf{f}^* = \mathbf{f}_b - \mathbf{K}_{bi}\mathbf{K}_{ii}^{-1}\mathbf{f}_i \tag{8}$$

이다. 한편 이때 내부의 변위는 다음과 같이 구해진다.

$$\mathbf{u}_i = -\mathbf{K}_{ii}^{-1}(\mathbf{K}_{ib}\mathbf{u}_b - \mathbf{f}_i) \tag{9}$$

즉 (6)식을 풀어서 경계상의 변위 $\mathbf{u}_b$를 구한 후 이를 (9)식에 대입하여 내부절점의 변위 $\mathbf{u}_i$ 또한 구할 수 있다. 여기서 강성행렬 $\mathbf{K}$의 크기가 전체 영역의 자유도에 해당

하는 반면 $\mathbf{K}^*$의 크기는 부구조간의 경계상의 자유도만큼으로 줄어들었음을 알 수 있다. 일반적으로 선형방정식을 푸는데 소요되는 시간이 강성행렬 크기, 즉 자유도의 제곱 혹은 세제곱에 비례한다고 알려져 있는데, 이러한 점을 감안한다면 부구조법을 적용한 경우 계산량이 상당히 감소되게 된다. 그러나 (7), (8), (9) 식에서 볼 수 있듯이 $\mathbf{K}_{ii}^{-1}$의 계산과정이 필요하게 되는데, 이는 $\mathbf{K}_{ii}$의 크기가 클 때 많은 계산량이 필요하게 되어 계산의 효율이 저하된다. 따라서 본 논문에서는 역행렬의 계산과정을 피해주기 위해 다음 식과 같이 수정된 Cholesky 분할방법을 사용하였다.[6]

$$\begin{bmatrix} \mathbf{K}_{ii} & \mathbf{K}_{ib} \\ \mathbf{K}_{bi} & \mathbf{K}_{bb} \end{bmatrix} = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{R} & \mathbf{K}^* \end{bmatrix} \begin{bmatrix} \mathbf{L}^T & \mathbf{R}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{10}$$

여기서 $\mathbf{K}_{ii}$와 $\mathbf{K}_{bi}$는 하부삼각행렬(lower triangular matrix) $\mathbf{L}$을 사용하여 다음과 같이 표현될 수 있다.

$$\mathbf{K}_{ii} = \mathbf{L}\mathbf{L}^T \tag{11}$$

$$\mathbf{K}_{bi} = \mathbf{R}\mathbf{L}^T \tag{12}$$

이때 (7), (8), (9) 식은 다음과 같이 표현될 수 있어 $\mathbf{K}_{ii}$의 역행렬을 구할 필요가 없어진다. 대신 상대적으로 계산시간이 조금 걸리는 $\mathbf{L}$의 역행렬을 계산하므로 계산시간의 절감이 가능해진다.

$$\mathbf{K}^* = \mathbf{K}_{bb} - \mathbf{R}\mathbf{R}^T \tag{13}$$

$$\mathbf{f}^* = \mathbf{f}_b - \mathbf{R}\mathbf{L}^{-1}\mathbf{f} \tag{14}$$

$$\mathbf{u}_i = \mathbf{L}^{-T}(\mathbf{L}^{-1}\mathbf{f}_i - \mathbf{R}^T\mathbf{u}_b) \tag{15}$$

3.2 부구조법을 사용한 해석 및 고찰

앞절에서 소개한 부구조법의 기본 수식화를 강소성 유한요소해석 프로그램에 적용하였고 이를 검증하기 위해 축대칭 업세팅 공정을 해석하였다. 해석은 IBM-PC 586을 사용하여 1단계(step)만을 수행하여 기존의 결과와 비교하였다. 이때 영역 분할에 따른 계산시간의 변화를 고찰하기 위해 부영역의 개수를 각각 1, 2, 4, 8, 16, 32, 64개로 변화시켜가며 입력데이터를 준비하였고, 각각의 경우에 대해 해석을 수행하여 결과를 비교하였다. Fig. 1에 각 경우에 대한 격자구조를 나타내었는데, 실선으로 표시한 부분이 부영역간의 경계를 의미한다. 여기서 실선상의 절점이 앞절에서 설명한 경계절점(boundary node)에 해당하고, 점선상의 절점이 내부절점(internal node)에 해당한다.

각 격자구조에 대해 해석에 소요되는 시간은 부영역의 개수에 따라 많은 변화를 보였는데, Table 1에 이를 비교하였다. 여기서 N_s는 부영역의 개수, N_i와 N_b는 각각 내

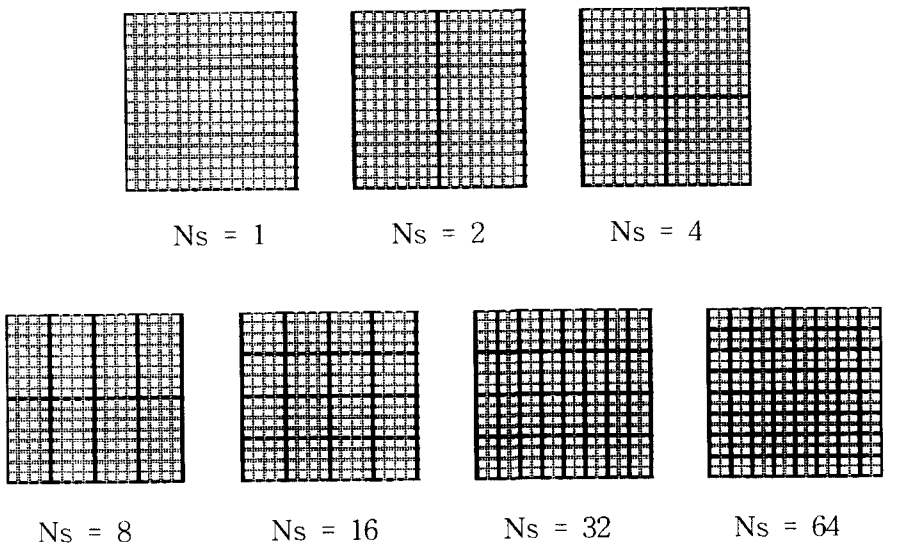

Fig. 1 Mesh structure for various no. of subaomains

부절점과 경계절점에 해당하는 경계절점의 자유도의 개수를 의미한다. 이중 N_s가 1일 때, 즉 부영역으로 분할하지 않고 전체 영역에 대해 해석을 수행하였을 때의 계산시간을 기준으로 해석시간의 비(比)를 사용하여 속도향상율(speed-up ratio)을 정의함으로써, 다른 경우에 대해 얼마만큼의 계산시간이 감소되었는지를 비교해보았다. Fig. 2에 부영역의 개수에 따른 속도향상율의 변화를 도시하였다. Fig. 2를 보면 처음에는 부영역의 개수가 증가할수록 속도향상율 또한 빠른 속도로 상승하나 N_s가 16일 때를 경계로 감소됨을 알 수 있다. 이는 부영역의 수가 증가할수록 내부절점의 자유도가 감소하여 정적응축 과정에서의 계산시간이 감소하는 반면, 상대적으로 경계절점의 자유도가 증가함으로써 응축된 방정식을 푸는 과정에서의 계산시간이 증가하기 때문이다. 따라서 계산시간을 최소화할 수 있도록 부영역의 개수를 적절히 선정해야 하는데, 본 예제의 결과로 미루어볼 때(Table 1 참조), 내부절점의 자유도와 경계절점의 자유도의 개수가 비슷한 수준일 때(N_s=16) 속도향상율이 가장 높음을 알 수 있다.

Table 1 Elapsed time w.r.t. no. of subomains

N_s	N_i	N_b	Time(sec)	Speed-up
1	450	128	1836.1	1.00
2	420	158	590.1	3.27
4	392	186	308.5	5.95
8	336	242	237.0	7.75
16	288	290	193.5	9.49
32	192	386	263.3	6.97
64	128	450	325.8	5.64

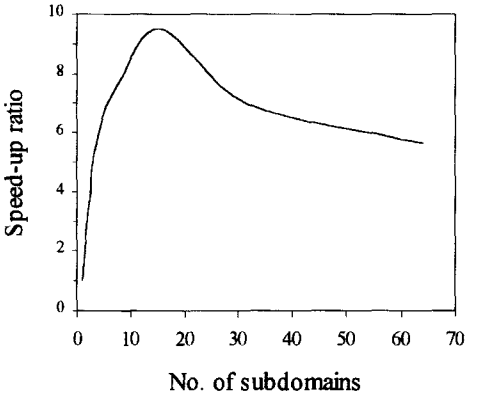

Fig. 2 Speed-up ratio w.r.t. no. of subdomains

4. 유한요소해석의 병렬 계산

4.1 PVM을 사용한 병렬 해석 프로그램 개발

유한요소법은 구조적으로 요소강성행렬(elementary stiffness matrix)의 생성, 강성행렬의 조합, 평형방정식의 계산, 동적 해석의 경우 시간적분기법 등 상대적으로 계산시간이 많이 걸리고 정보의 종속성(dependency)이 적은 부분에서 병렬처리기능을 활용하기가 유리하다. 유한요소해석에서의 병렬계산은 사용 컴퓨터의 종류와 병렬처리 기법에 따라 많은 연구 결과가 발표되었는데 주로 구조 및 충돌해석, 유동해석 등 많은 계산시간이 소요되는 문제에 대해 적용되었으며,[14,15] 최근에는 소성가공공정 유한요소해석에 병렬처리를 적용한 사례도 발표되었다.[10,11]

본 연구에서는 2차원 강소성 유한요소해석 프로그램에 부구조법을 적용하여 영역 분할 및 각 영역의 강성행렬 계산, 각 내부자유도의 정적 응축 과정 등 영역에 따른 정보의 독립성을 확보하고, 이를 병렬적으로 계산할 수 있도록 프로그램을 병렬화시켰다. 여기서 여러개의 CPU간의 정보교환(message passing)을 위한 방법으로서 PVM(Parallel Virtual Machine)을 사용하였다.

PVM은 서로 다른 컴퓨터들을 네트웍으로 연결하여 하나의 병렬 컴퓨터처럼 동작하게 해주는 정보교환 라이브러리(message passing library)로 현재 MPI(Message Passing Interface)와 더불어 세계적으로 가장 널리 통용되고 있다.[16] PVM은 호환성이 좋고 별도의 병렬컴퓨터를 구입하지 않고도 기존의 워크스테이션을 활용하여 병렬처리 환경을 구축할 수 있다는 장점이 있는 반면, 네트워크를 사용하여 정보를 전달하므로 네트워크 상태에 따라 정보전달 효율이 저하될 가능성이 있다.

본 연구에서는 PVM을 이용한 병렬처리 기법중 주종모델(master-slave model)을 사용하였다. 주종모델에서는 상호간에 연결된 여러대의 컴퓨터를 1개의 주 프로세서(master processor)와 여러개의 종 프로세서(slave processor)로 구분하여 주 프로세서의 통제하에 종 프로세서들이 병렬적으로 연산을 수행하게 된다. Fig. 3에 이러한 개념을 도시하였다.

4.2 소성가공 공정의 병렬 계산

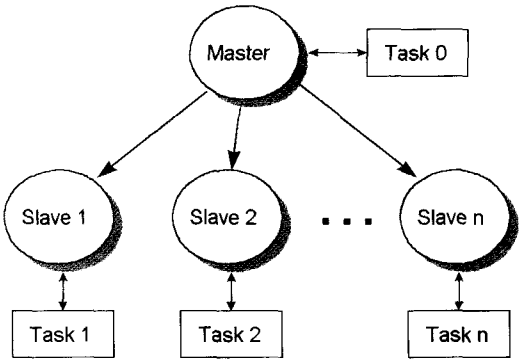

Fig. 3 Master-slave paradigm in PVM

기존에 순차적으로 진행되던 유한요소해석 과정의 병렬계산을 위해서는 2단계의 수정작업이 필요하다. 첫 번째는 데이터의 병렬화, 즉 데이터 상호간의 독립성을 확보하는 과정인데, 이는 부구조법에 의한 영역 분할을 도입하여 각 부영역내의 계산을 독립적으로 계산할 수 있도록 하였다. 두 번째로는 독립성이 확보된 계산과정을 병렬적으로 계산할 수 있는 알고리즘을 구현하는 과정으로 이는 PVM에서 제공하는 각종 라이브러리 함수를 사용하여 구현하였다. 계산에 사용된 장비는 네트워크로 연결된 Silicon Graphics사의 INDIGO 기종 4대를 사용하였다.

소성가공 공정의 해석 예제로는 축대칭 업세팅(upsetting) 공정을 압하량 20%까지 해석을 수행하였다. 이때 펀치의 이동속도는 10*mm/sec*, 마찰계수(m)는 0.12로 설정하였다. 소재는 SCr420H이며, 응력-변형율 관계식은 다음과 같다.

$$\bar{\sigma} = 510 + 863\bar{\varepsilon}^{0.15} (N/mm^2) \qquad (16)$$

본 연구에서 4대의 컴퓨터를 사용하여 PVM 환경을 구축하였으므로, Fig. 1에서 도시된 7개의 격자구조중 부영역의 개수를 1개, 2개, 4개로 나누었을 경우에 대해서 해석을 수행해보았다. 또한 각각의 경우 1개의 컴퓨터만을 사용하여 해석하였을 경우, 즉 병렬처리를 하지 않았을 경우와 부영역의 개수만큼 프로세서의 수를 늘려 사용하였을 경우에 대해 계산을 수행하여 소요 시간을 비교해보았다(Table 2). 여기서 NS는 부영역의 개수(number of subdomains)를, NP는 프로세서의 수(number of processors)를 의미한다. 계산시간은 순수한 계산을 수행하는데 소요된 시간(analysis time)과 컴퓨터간에 정보를 주고받는데 소요되는 시간(message passing time)으로 구분하여 비교하였다. 이때 속도향상율은 전체 영역을 단일 프로세서로 해석한 경우(NS1NP1)의 계산시간을 기준으로 하여 다른 경우의 계산시간을 비(比)로서 나타내었다. Fig. 4에 각각의 경우에 대한 계산시간을 그래프로 도시하였다.

이중 전체 영역을 4개의 부영역으로 분할하여 4개의 프로세서로 병렬계산을 수행한 경우(NS4NP4)가 NS1NP1의 경우보다 무려 10배의 속도향상을 보여 가장 효율적인 계산이 이루어졌음을 알 수 있다. 이는 우선적으로 부구조법에 의한 영역분할에 따른 계산시간 절감의 효과가 있는데다가 병렬처리를 구현함으로써 추가적인 성능향상이 이루어졌기 때문이다.

병렬화의 효율을 검토하기 위해 Table 3에 부영역의 개수가 4인 경우 프로세서 개수의 증가(1, 2, 4)에 따른 속도향상율을 비교하였다. 여기서 해석시간만을 고려하여 속도향상을 비교하였을 때는 각각 프로세서가 2개일 때 1.98, 프로세서가 4개일 때 3.91로 프로세서의 개수에 비례하여 속도가 증가하는 경향(scalability)을 보였다.

Table 2 Elapsed time for parallel computation

Elapsed time(sec)	NS1NP1	NS2NP1	NS2NP2	NS4NP1	NS4NP4
Analysis	1172.5	379.8	182.3	217.9	55.8
Message passing	0.0	0.0	104.7	0.0	60.1
Total	1172.5	379.8	287.0	217.9	115.9
Speed-up	1.00	3.09	4.09	5.38	10.12

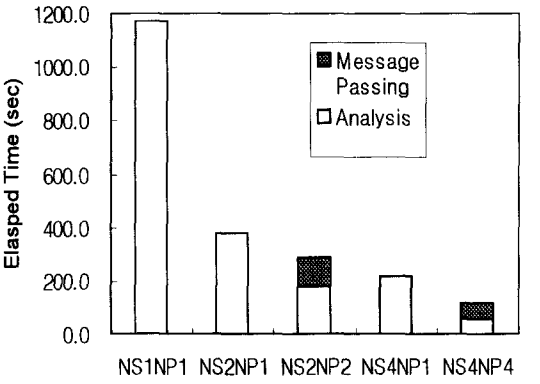

Fig. 4 Comparison of elasped time for parallel computation

Table 3 Speed-up ratio w.r.t. no. of processors

No. of processors	1	2	4
Speed-up (analysis)	1.00	1.98	3.91
Speed-up (total)	1.00	1.32	1.88

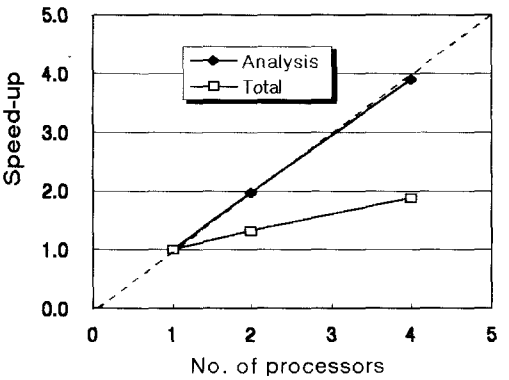

Fig. 5 Comparison of elasped time w.r.t. no. of processors

Fig. 5에 이를 그래프로 도시하였다. 점선으로 표시한 그래프가 이상적인 병렬화가 이루어졌을 경우의 속도향상율로, 해석시간만을 고려했을 경우 거의 일치하고 있는 점으로 보아 프로그램의 병렬화가 성공적으로 이루어졌음을 알 수 있다. 반면에 프로세서간의 정보교환에 소요되는 시간까지 고려하여 총 해석시간을 비교하였을 경우는 속도향상율이 저하되는 현상을 볼 수 있다. 이러한 현상은 본 연구에서 사용한 컴퓨터 네트워크를 사용해서 병렬계산 환경을 구현한 점을 볼 때, 네트워크를 통한 정보교환 과정에서 많은 시간이 소요되었기 때문으로 판단된다. 실제로 동일한 문제를 단일 기종의 병렬 컴퓨터를 사용하여 해석을 수행한다면 정보교환에 따른 시간이 상당히 절감됨에 따라 위와 같은 속도향상율의 저하량이 훨씬 줄어들 것으로 기대된다.

한편 계산의 정확도를 평가하기 위해 Fig. 6에 각각 NS1NP1, NS2NP2, NS4NP4의 3가지 경우에 대해 변형형상 및 유효변형율의 분포를 도시하였다. 각각의 결과를 비교해보면 변형형상과 유효변형율 모두 동일한 결과를 보여줌으로써 부구조법에 의한 영역 분할과 각 영역에 의한 병렬계산 기법을 사용한 해석의 정확도에는 아무런 문제점이 없음을 알 수 있다.

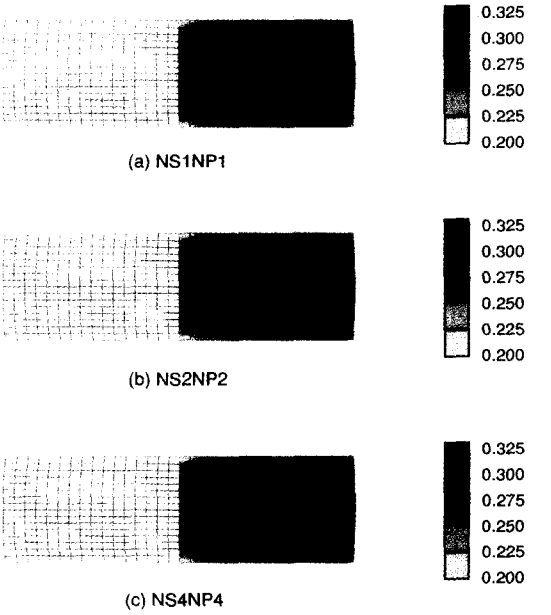

Fig. 6 Deformed shape and effective strain distribution

5. 결 론

이상으로 본 연구에서는 소성가공 공정의 유한요소해석시 계산시간을 효과적으로 절감하기 위해 부구조법을 사용한 영역분할 기법을 적용하였다. 이때 정적응축 과정에서 발생하는 역행렬의 계산과정에서 수정된 Cholesky 분할법을 도입함으로써 계산의 효율을 증대시켰다. 또한 영역을 분할함으로써 각 영역간의 정보의 독립성을 확보하였고, 해석 프로그램에서 해당 영역의 독립적인 계산과정을 PVM 라이브러리를 사용하여 병렬화시켰다. 개발된 프로그램을 사용하여 실제 예제의 해석을 통해 영역의 분할에 따른 계산시간 절감 및 병렬 계산에 의한 성능 향상 효과를 확인하였다.

이와 같은 계산시간의 효과적인 절감으로 인해 많은 계산시간이 소요되는 소성가공 공정의 유한요소해석을 보다 효율적으로 수행할 수 있을 것으로 기대된다. 현재 국내에도 병렬컴퓨터가 점차로 널리 보급되는 점을 감안할 때, 차후에는 보다 개선된 병렬처리 환경에서 계산의 효율을 제고할 수 있을 것으로 전망된다.

참고문헌

(1) Przemieniecki, J. S., 1963, "Matrix Structural

Analysis of Substructures", AIAA Journal, Vol. 1, pp. 138~147

(2) The NASTRAN, 1976, "Theoretical Manual Level 16.0", NASA SP-221

(3) Noor, A. K., Kamel, H. A. and Fulton, R. E., 1977, "Substructuring Techniques - Status and Projections", Computers & Structures, Vol. 8, pp. 621~632

(4) Hitchings, D. and Balasubramanian, K., 1984, "The Cholesky Method in Substructuring with an Application to Fracture Mechanics", Computers & Structures, Vol. 18, pp. 417~424

(5) Han, T. Y. and Abel, J. F., 1984, "Substructure Condensation Using Modified Decomposition", International Journal for Numerical Methods in Engineering, Vol. 20, pp. 1959~1964

(6) Liu, X. L. and Lam, Y. C., 1995, "Condensation Algorithms for the Regular Mesh Substructuring", International Journal for Numerical Methods in Engineering, Vol. 38, pp. 469~488

(7) Farhat, C. and Wilson, E., 1987, "Concurrent Iterative Solution of Large Finite Element Systems", Communications in Applied Numerical Methods, Vol. 3, pp. 319~326

(8) Jonsson, J., Krenk, S. and Damklide, L., 1995, "Recursive Substructuring of Finite Elements", Computers & Structures, Vol. 54, pp. 395~404

(9) Farhat, C., Crivelli, L. and Rouz, F. X., 1994, "Extending Substructure Based Iterative Solvers to Multiple Load and Repeated Analyses", Computer Methods in Applied Mechanics and Engineering, Vol. 117, pp. 195~209

(10) Doltsinis, I. St. and Nölting, S., 1991, "Generation and Decomposition of Finite Element Models for Parallel Computations", Computing Systems in Engineering, Vol. 2, pp. 427~449

(11) Hwang, K. and Briggs, F. A., 1984, "Computer Architecture and Parallel Processing", McGraw Hill

(12) Lee, C. H. and Kobayashi, S., 1973, "New Solution to Rigid Plastic Deformation using a Matrix Method", ASME, Journal of Engineering for Industry, Vol. 95, pp. 865~873

(13) C. C. Chen and S. Kobayashi, 1978, "Rigid-Plastic Finite Element Analysis of Ring Compression, Applications of Numerical Methods to Forming Processes", ASME, AMD, Vol. 28, pp. 163~174

(14) Farhat, C. and Roux, F. X., 1994, "Implicit Parallel Processing in Structural Mechanics", Computational Mechanics Advances, Vol. 2, pp. 1~124

(15) Farhat, C. and Wilson, E., 1988, "A Parallel Active Column Equation Solver", Computers & Structures, Vol. 28, pp, 289~304

(16) Geist, A, et al., 1994, "PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing", The MIT Press