PC 트랜스퓨터 시스템을 이용한 구조최적화의 병렬처리

Parallel Processing of Structural Optimization Using PC Transputer System

황 진 하* 박 종 희**
Hwang, Jin-Ha Park, Jong-Hoi

요 지

본 연구는 개별메모리를 갖는 소결합 구조의 MIMD형 병렬컴퓨터인 트랜스퓨터시스템 하에서 구조최적화를 위한 병렬처리 과정을 보이고 시험모델에 적용하여 타당성 및 효율성을 검증한다. 전체 최적화과정의 대부분을 차지하는 해석 및 민감도 알고리즘은 영역단위의 병렬성을 갖는 부구조화에 근거하고 하드웨어 구성에 맞춰 변환 재구성한다. 각 노드간 통신은 정적응축과 설계도함수에 한정, 그 횟수를 최소화하고 그들을 동기화하므로써 개별메모리형 연산모델의 약점인 통신비용의 문제를 해소한다. PC를 호스트로 한 수치실험은 고속화 효율성 면에서 고무적인 결과를 보여주고 있으며, 이런 점에서 시스템의 확장성을 함께 고려한다면 트랜스퓨터 시스템에 기초한 병렬처리는 공학 환경의 변화와 요구에 부응하는 좋은 대안이 될 수 있다.

핵심용어 : 병렬처리, 구조최적화, 트랜스퓨터, 영역단위 병렬성, 부구조화

Abstract

A parallel processing approach for structural optimization on transputer system, which is loosely coupled MIMD computer with local memory, is presented in this paper. The validity and efficiency of this approach are demonstrated and verified by some test models of trusses. Structural and sensitivity analysis algorithms, which spend most of overall structural optimization processes, are based on substructuring scheme with domain-wise parallelism and converted to be adapted to hardware architecture. The intercommunications between each node are limited to only two times of static condensation and design derivatives calculation, and synchronized in order to minimize the communication cost. PC hosted numerical experimentations show encouraging results in speed-up and efficiency. In this respect, if flexible scalability is considered together, parallel processing based on transputer system can be a good alternative to the changes and needs of engineering environment.

Keywords : *parallel processing, structural optimization, transputer, domain-wise parallelism, substructuring*

* 정회원·충북대학교 구조시스템공학과, 교수
** 정회원·충북대학교 토목공학과, 박사과정수료

• 이 논문에 대한 토론을 1999년 9월 30일까지 본 학회에 보내주시면 1999년 12월호에 그 결과를 게재하겠습니다.

1. 서 론

구조최적화는 요구된 설계조건을 만족시키면서 최대의 경제성을 갖는 구조계를 도출하는 과정으로 구조설계를 하나의 수학적 문제로 형성하고 최적화기법 등을 활용하여 체계적으로 접근하며, 실행 측면에서 해석과 최적화모듈의 반복 결합으로 이루어지는 전산과정이다.

대형시스템을 최적화하는 문제는 대단히 많은 설계변수와 제한조건을 갖기 때문에 전산처리에 상당한 메모리와 시간을 필요로 하고, 따라서 소형컴퓨터로는 실행키 어려울 뿐만 아니라 대형컴퓨터에서도 많은 비용을 요구하게 된다. 게다가 구조최적화에서 대부분의 CPU시간을 보통 수십번이상 반복되는 거동해석과 민감도해석에 사용하므로 이들에 드는 소요시간 및 그에 따른 비용이 전체 과정에 미치는 영향은 대단히 크다.

최근 컴퓨터의 고성능화로 구조공학의 단위 프로세스의 처리에는 큰 어려움이 없으나, 이러한 점이 구조 분야에 대한 최신의 강력한 컴퓨터 자원의 활용을 소홀히 하게 하는 요인이 되어서는 아니되며, 갈수록 공학 전반의 환경은 여러 전문 영역의 병행처리와 통합화 방향으로 흐르고 있어 보다 강력하고 경제적인 전산자원을 필요로 한다.

이러한 점들을 해결하기 위한 대안으로 최근의 전산환경에 맞추어 구조공학 분야에 슈퍼컴퓨터나 병렬처리 기능을 활용하는 방법이 국내·외에서 활발히 연구되고 있다. 병렬처리란 기존의 단일 프로세서로 구현할 수 있는 연산속도의 기술적 한계를 극복하기 위해 다수의 프로세서를 통한 성능 향상을 추구하는 방법이다.

먼저 구조해석문제는 매트릭스의 생성 및 조합이나 시스템방정식의 계산과정 등 이론상 병렬로 처리할 수 있는 특성을 많이 갖고 있다. 방법론적 측면에서 크게 시스템 방정식의 처리방식과 영역분할방식으로 나누어지며 병렬성의 수준이나 플랫폼에 따라 많은 연구결과가 발표되었으나[1], 이들 대부분은 특정 목적이나 기종에 국한되었다. 구조최적화의 경우 아직은 양적인 면에서 조차 미미한 수준이나 El-Sayed[2] 등은 슈퍼컴퓨터 Cray X-MP 상에서 해석과정을 병렬처리한 후 순차적인 최적화과정에 연결하였고, Mangasarian[3),4)] 등은 초병렬컴퓨터 CM-5 상에서 민감도해석까지 확장시켰다. Adeli[5]는 공유메모리형 다중프로세서 컴퓨터 Encore Multimax를 이용하여 최적규준법에 의한 병렬최적화를 수행하였다.

본 연구에서는 호스트 컴퓨터의 기종에 관계없이 보드나 프로세서의 추가만으로 시스템의 확장이 가능한 개별 메모리형 MIMD 병렬컴퓨터 구조를 갖는 트랜스퓨터시스템을 이용하여 병렬처리 환경을 구축하고, 구조해석을 위한 병렬처리 방법 중 상기 환경에 적합하고 효율성이 우수하다고 평가되는 영역단위의 병렬성을 갖는 부구조화 기반 알고리즘을 트랜스퓨터의 통신 환경에 맞춰 재구성하여, 구조최적화를 위한 병렬처리 프로그램을 개발하고, 몇 가지 트러스의 예제를 통해 타당성 및 효율성을 검증하였다.

2. 트랜스퓨터 병렬시스템

컴퓨터 구조를 분류하는 방법은 몇 가지가 있는데, 그중 명령스트림과 데이터스트림에 따른 Flynn[9]의 분류가 널리 통용되고 있다. 그에 의하면 병렬구조는 SISD, SIMD, MISD, MIMD의 4종류로 나눌 수 있는데, SISD(Single Instruction stream-Single Data stream)는 표준 비병렬 컴퓨터이며, MISD(Multiple Instruction stream-Single Data stream)는 이론상으로만 존재한다. SIMD(Single Instruction stream-Multiple Data stream)는 하나의 제어장치 아래 다수의 처리기가 수많은 데이터에 대해 같은 명령을 수행하며 배열(array)처리기나 벡터처리기가 그 예이다.

MIMD(Multiple Instruction stream-Multiple Data stream) 모델은 각 프로세서가 서로 독립적으로 작업하며 서로 전혀 다른 프로그램을 실행할 수 있는 장점을 갖는다. 또한 MIMD는 메모리사용 형태에 따라 두 가지 연산모델로 구분될 수 있다.

먼저 공유메모리(shared memory)형 MIMD 컴퓨터에서는 모든 변수가 공유메모리에 저장되어

변수의 형식에 관계없이 모든 프로세서가 함께 접근할 수 있는 전역변수로 취급된다. 그러나 접근하는 연산소자의 우선 순위를 프로그램 상의 수행순서에 맞게 지정하지 않으면 그 결과의 오차가 발생하거나 연산소자간의 충돌이 일어날 수 있다. 다음, 개별 메모리(local memory)형 MIMD 컴퓨터의 경우는 각 프로세서별로 마련된 개별메모리에 변수를 저장하고 모든 변수가 지역변수로 취급되므로 공유메모리형 모델보다 동기화(synchronization)과정 및 프로세서의 메모리 접근방식이 유리하다.

본 연구에서는 인텔 166Mhz 펜티엄프로 PC를 호스트로 하고 영국의 INMOS사에서 개발한 트랜스퓨터와 인텔의 CPU i860을 TRANSTech이 모기판 TMB08에 패키징한 Fig. 1 형태의 TRAM (Transputer Module) TTM110 3조를 사용하여 MIMD 병렬처리환경을 구축한다.

인텔사의 i860은 64bit Microprocessor로서 프로그램의 연산을 담당하고, 트랜지스터와 컴퓨터의 두 단어를 축약한 의미를 갖는 트랜스퓨터는 4개의 통신링크를 통하여 프로세서들 간의 통신을 담당한다.

트랜스퓨터를 통한 데이터 통신은 처리장치에 의해 동시에 수행되며 CPU의 대기시간 없이 모든 링커들이 연속적으로 데이터를 전송할 수 있다는 것이 특징이며 각 트랜스퓨터가 지닌 4개의 통신링크를 통해 트랜스퓨터의 갯수에 따라 파이프라인, 트리, 그리고 메쉬 등의 다양한 통신형태를 구현할 수 있다.

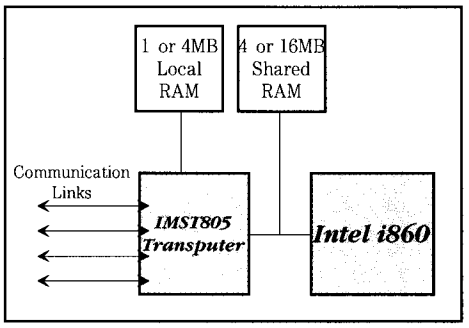

Fig. 1 TTM110 Block Diagram

3. 부구조화기반 구조해석

구조해석은 요소강성도매트릭스의 생성, 응력의 계산 등 이론상 병렬로 처리할 수 있는 부분이 많으며, 부구조화는 각 부구조별 전체강성도매트릭스의 구성, 부구조별 정적응축, 변형 및 응력계산과정들을 독립적으로 수행할 수 있으므로 병렬처리에 매우 적합한 방법이다.

병렬처리란 대규모 작업을 여러 개의 부분작업으로 분할하고 각각을 개별 프로세서에 할당하여 각 작업을 동시에 수행하는 것을 의미하며, 기존의 단일프로세서 연산속도와 비교할 때 상당한 성능향상을 추구할 수 있다. 그러나 Noor[6]가 지적한 바와 같이 병렬해석에 있어 시스템이 대형화될수록 데이터 입력과정에 상당한 통신시간이 소요되는 문제를 야기하며, 이는 자료 입출력이 하나의 프로세서에서만 이루어지는데 기인한다. 즉 자료의 입출력이 호스트나 마스터프로세서에서 수행되는 경우, 입력데이터를 모든 슬래이브 프로세서로 전송하는데 상당한 부담을 주게된다.

본 연구에서는 보다 효율적으로 노드간 통신을 수행하기 위해, 먼저 문제의 정의 및 부구조화 모델링과정에서 각 부구조에 해당하는 입력데이터들이 해당 노드에서 수행될 수 있도록 하였다.

시스템방정식은 경계와 내부에 관계된 양들로 나누어 구성하며 각 서브시스템의 강성도매트릭스와 하중벡터는 할당된 프로세서에서 독립적으로 형성된다. N개의 서브시스템 방정식은

$$\begin{bmatrix} K_{bb}^{(s)} & K_{bi}^{(s)} \\ K_{ib}^{(s)} & K_{ii}^{(s)} \end{bmatrix} \begin{bmatrix} U_b^{(s)} \\ U_i^{(s)} \end{bmatrix} = \begin{bmatrix} P_b^{(s)} \\ P_i^{(s)} \end{bmatrix} \quad (1)$$

으로 구성된다. 여기서, K는 구조강성도매트릭스, U와 P는 각각 변위벡터와 하중벡터를 나타내며, 아래첨자 b와 i는 각각 경계와 내부자유도에 관계된 양들임을 의미하고 윗첨자 s는 특정 부구조를 나타낸다.

식(1)의 경계에 해당하는 부분은 각 부구조들이 공유하고 이들과 관계되는 정보는 각 프로세서간의 통신을 통하여 해당 부구조로 송수신되며

특히 부구조화 해석과정 중 한번의 통신으로 이루어지기 때문에 통신상의 과부하를 최소화 할 수 있는 잇점을 갖는다.

각 부구조에서 내부와 관련된 항들은

$$K_b^{(s)} = K_{bb}^{(s)} + K_{bi}^{(s)} Q^{(s)} \tag{2}$$

$$F_b^{(s)} = P_b^{(s)} + Q^{T(s)} P_i^{(s)} \tag{3}$$

여기서

$$Q^{(s)} = -[K_{ii}^{(s)}]^{-1} K_{ib}^{(s)} \tag{4}$$

의 과정을 통해 경계강성도매트릭스와 경계하중벡터로 구한다. N개의 부구조를 갖는 합성된 유효경계강성도매트릭스와 유효경계하중벡터는 각 부구조에서 (N-1)번 통신과정을 통해 다음과 같이 구하여 진다.

$$K_B^{(s)} = \sum_{s=1}^{N} \beta^{(s)T} K_b^{(s)} \beta^{(s)} \tag{5}$$

$$F_B^{(s)} = \sum_{s=1}^{N} \beta^{(s)T} F_b^{(s)} \beta^{(s)} \tag{6}$$

여기서 β는 Boolean 변환매트릭스이고 각 부구조는 전체 시스템의 요소 중 각각에 해당하는 경계평형방정식을 구성한다.

$$K_B^{(S)} U_B^{(S)} = F_B^{(S)} \tag{7}$$

이 방정식을 풀어 경계변위를 구한 후 다시 식(1)로부터 각 내부변위

$$U_i^{(s)} = K_{ii}^{-1(s)} [P_i^{(s)} - K_{ib}^{(s)} U_b^{(s)}] \tag{8}$$

를 구한다. 응력 또한

$$\sigma^{(s)} = S^{(s)} U^{(s)} \tag{9}$$

을 통해 계산되며 여기서, S는 응력-변위 변환매트릭스이다.

4. 설계민감도해석 및 총합

전체구조시스템의 총 중량 또는 체적을 목표함수로 한 유한차원 공간에서 구조최적화 문제는 다음과 같이 형성 될 수 있다.

$$\text{Min. } F(X, U) = \sum_{S=1}^{S} f(x) \tag{10}$$

$$G_j(X, U) \le 0, \; j=1, 2, \cdots, J \tag{11}$$

$$X_i^L \le X_i \le X_i^U, \; i=1, 2, \cdots, I \tag{12}$$

식(11)는 구조시스템의 거동을 제한하는 조건식으로써 변위, 응력, 고유치 등에 대한 제한조건을 나타내며, 식(12)은 설계변수인 단면 특성에 대한 제한조건이다. 여기서 X^L, X^U는 설계변수의 하한치와 상한치를 각각 나타낸다.

오늘날 구조최적화에 내재되는 해석의 보편적 도구로서는 주로 유한요소법이 사용되고 있으며 그에 상응하는 최적화 방법은 수리계획법이다. 즉, 해석모듈에 의해 구조시스템의 거동이 산출되고 이들은 수리계획법에서 제한조건으로 설정되어 이 조건을 만족하는 목표함수를 최소화하게 된다. 그러나 일반적인 최적화 알고리즘의 제한조건이 설계변수로 설정되는데 반해 실제적인 구조최적화의 경우 제한조건이 해석모듈에서 산출되는 거동변수에 연관된다. 따라서 성질이 상이한 관계로 구성되는 모듈간의 연동을 위한 변환과정이 요구된다.

병렬환경에서 실행된 본 연구의 전체 설계최적화 과정에 대한 개략도는 Fig. 2에 나타낸다.

각 프로세서는 전단계 해석에서 얻은 거동 정보를 토대로 독립적으로 각 부구조에 관련된 목표함수와 제한조건들을 계산하고 자기 설계변수에 대한 설계함수들의 도함수를 산출한다. 이들 정보를 토대로 시스템최적화가 이루어지고 결과값은 수렴조건에 따라 해석모듈로 피드백된다.

여기서 각 노드간 통신은 정적응축과 설계도함수 계산에 한정하여 그 횟수를 최소화하고 그들을 동기화하므로써 개별 메모리형 연산모델의 약점인 통신 비용의 문제를 해소한다.

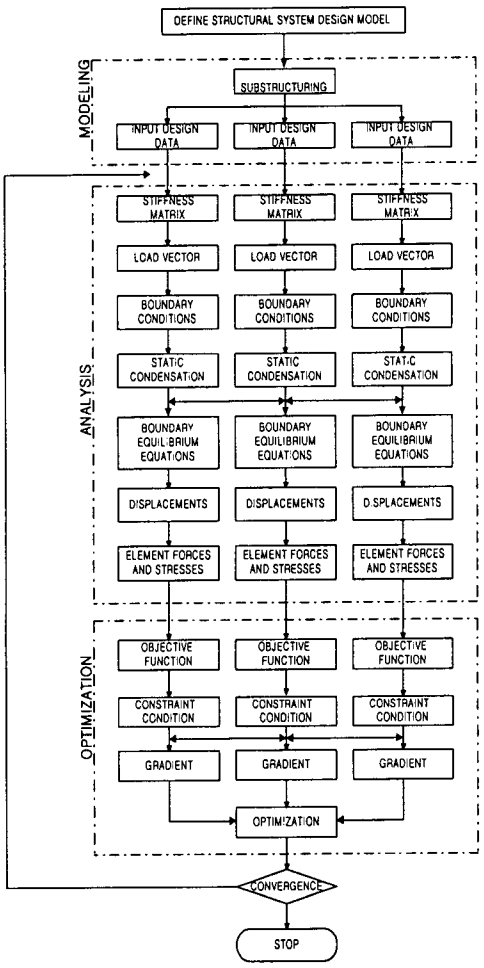

Fig. 2 Flow diagram of parallelized processes of design optimization

설계도함수를 구하는 방법에는 크게 미분법과 변분법이 있으나 이산형 구조에는 주로 음함수 미분법이 적용되며 이것은 다시 수치적 방법과 해석적 방법으로 나누어진다. 이들은 정확도나 효율성 면에서 후자가 선호되나, 설계함수들이 설계변수의 비선형 음함수로 표현되는 구조최적화에서 때로 어렵거나 불가능한 반면, 전자는 시간이 많이 걸리고 정확성이 떨어지기는 하나, 어느 문제에나 쉽게 다루어 질 수 있는 양면이 있다.

본 연구에서는 우선 병렬처리의 접근성이나 편의성을 넓히면서, 타당성과 효율성을 상대검증하기 위하여 많은 계산양을 요하는 전자의 방법을 취하기로 한다. 그중 전방차분법을 쓸 경우 변위와 응력은 각각

$$\frac{\partial U}{\partial X} \simeq \frac{\Delta U}{\Delta X} = \frac{U(X+\Delta X)-U(X)}{\Delta X} \tag{13}$$

$$\frac{\partial \sigma}{\partial X} \simeq \frac{\Delta \sigma}{\Delta X} = \frac{\sigma(X+\Delta X)-\sigma(X)}{\Delta X} \tag{14}$$

으로 계산하되, 식(13)에서 $U(X+\Delta X)$는 식(7)과 (8)로 부터

$$K_B(X+\Delta X)\cdot U_B(X+\Delta X) = F_B(X+\Delta X) \tag{15}$$

$$K_{ii}U_i(X+\Delta X) = P_i - K_{ib}(X+\Delta X)U_b(X+\Delta X) \quad \text{on substructures} \tag{16}$$

을 풀어 계산하며, $\sigma(X+\Delta X)$는 식(9)로 부터

$$\sigma(X+\Delta X) = S(X+\Delta X)U(X+\Delta X) \quad \text{on substructures} \tag{17}$$

을 풀어 구한다. 여기서 설계변화량 ΔX는 기준 설계치의 0.01로 취한다.

5. 적용 및 분석

본 연구에서 개발된 방법을 먼저 전형적인 최적화 검증모델인 10부재 트러스에 시험 적용하여 방법론적 타당성을 검증하고, 다시 200부재 트러스를 대상으로 효율성을 검증하였다.

이들에 대한 수치실험 결과는 단일 프로세서상에서 실행된 전체구조에 대한 순차 최적화방법(방법 I)과 부구조화에 근거한 순차 최적화 알고리즘(방법 II)과 비교·분석한다. 이들은 모두 동일한 기본 하드웨어 환경에서 실행되었으나, 방

법 I 과 II는 최적화 방법론에서, 방법 I ,II와 본 연구의 방법은 하드웨어 환경에서 차이가 나며, 각 방법에서 최적화 알고리즘은 Augmented Lagrange Multiplier방법을 이용하였다.

병렬처리의 성능은 고속화배수(speed up)와 효율성으로 평가할 수 있는데, 프로세서 증가시의 고속화배수 S_P와 효율성 E_P는

$$S_P = \frac{T_{sequential}}{T_{parallel}} \tag{18}$$

$$E_P = \frac{S_P}{P} \tag{19}$$

으로 표현된다. 여기서, $T_{sequential}$는 순차프로그램의 실행시간을, $T_{parallel}$은 병렬프로그램의 실행시간을 나타내고, P는 연산에 사용된 프로세서의 갯수 이다.

5. 1 Ten member truss

10개의 부재를 갖는 트러스 모델은 Fig. 3에 나타나고, 이를 2개로 분할한 부구조에 관한 정의는 Table 1, 설계자료는 Table 2와 같다.

Table 3은 각 방법들의 목표함수값을 참고문헌과 비교하여 나타냈다.

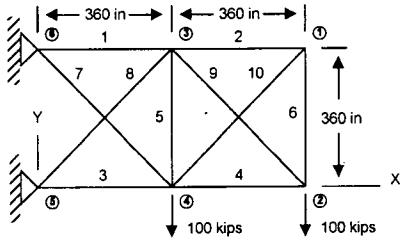

Fig. 3 10-member truss

Table 1 Data for 10-member plane truss with substructuring

Number of substructure	Substructure No.	Element No.
2	1	1, 3, 5, 7, 8
	2	2, 4, 6, 9, 10

Table 2 Design data of 10-member plane truss

Modulus of elasticity	: 10,000.0 ksi
Allowable stress limits	: ±25.0 ksi
Allowable displacement	: ±2.0 in.
Specific weight	: 0.1 lb/in.3
Lower bound of design	: 0.10 in.2
Upper bound of design	: None
No. of design variable	: 10

Table 3 Comparison of objective function

Algorithm \ Data	Objective function (Ib)
METHOD I	5060.62
METHOD II	5061.01
PRESENT	5061.01
REFERENCE[7]	5061.60

Table 4는 단일 프로세서에서 수행한 결과를 기준으로 다중프로세서상의 연산효과를 식(18) 및 (19)에 의해 나타내었다.

Table 5는 단일 프로세서상에서 전체구조를 최적화한 수행 방법 I의 결과를 기준으로 2개의 프로세서를 사용한 고속화효과를 상대·비교한 것이다.

단일프로세서를 쓰면서 부구조화에 기초한 방법 II는 구조전체를 한꺼번에 다루는 방법 I보다 1.6배 정도의 효과를 보며, 다시 프로세서 2개를 이용한 병렬처리를 통해 2배 이상의 효과를 얻고 있다.

Table 4 Parallel performance result on transputer system

Number of processor	Speed-up S_P	Efficiency E_P
1	1	-
2	2.06	1.03

Table 5 CPU time(sec) and speed up

Algorithm \ Data	METHOD I	METHOD II	PRESENT
Total CPU time	4.91	2.98	1.44
Relative Speed-up	1.0	1.65	3.40

5. 2 Two hundreds member truss

Fig. 4의 200-member truss를 3개의 부구조로 분할한 자료는 Table 6 및 Table 7과 같다.

Table 8은 각 알고리즘별 목표함수를 문헌과 비교하여 나타내었다.

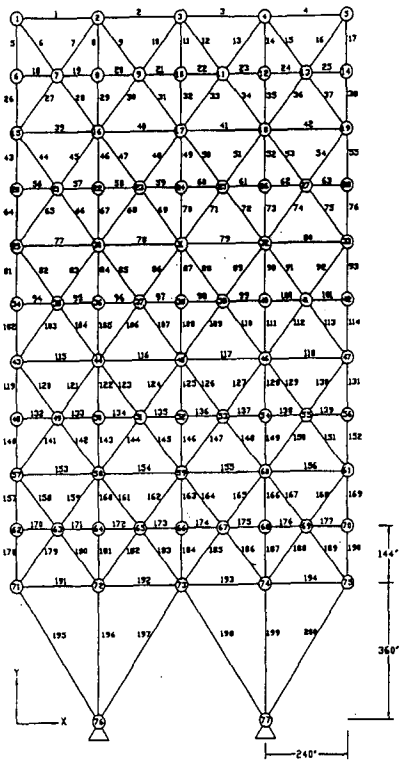

Fig. 4 200-member truss

Table 6 Data for 200-member plane truss with substructuring

Number of substructure	Substructure No.	Element No.
2	1	1,2,5,6,7,8,9,10,11,..., 191,192,195,196,197
	2	3,4,12,13,14,15,16,..., 193,194,198,199,200
3	1	1, 2, 3, ···, 80
	2	77, 78, ···, 152
	3	153, 154, ···, 200

Table 7 Design data of 200-member plane truss

Modulus of elasticity	: 30,000.0 ksi
Allowable stress limits	: ±30.0 ksi
Allowable displacement	: ±0.5 in.
Specific weight	: 0.283 lb/in.^{3}
Lower bound of design	: 0.10 in.^{2}
Upper bound of design	: None
No. of design variable	: 96 (linked)

Loading condition	Node No.	Load component (ksi) in direction		
		x	y	z
1	1, 6, 15, 20, 29, 34, 43, 48, 57, 62, 71	1.0	0.0	0.0
	1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, ..., 71, 72, 73, 74, 75	0.0	-10.0	0.0

Table 8 Comparison of objective function

Algorithm \ Data	Objective function (lb)
METHOD I	29,176
METHOD II	29,155
PRESENT	29,155
REFERENCE[7]	28,963

단일 프로세서에서 수행한 결과를 기준으로 다중프로세서상의 연산효과를 식 (18) 및 (19)에 따라 Table 9에 나타내었다.

이론상 프로세서의 수가 증가하면 고속화배수도 선형으로 증가하나 실제의 경우 프로세서간의 통신 부담 등으로 인해 차이를 나타낸다[10].

Table 9 Parallel performance result on transputer systems

Number of processor	Speed-up S_P	Efficiency E_P
1	1	-
2	1.99	0.99
3	2.93	0.97

Table 10은 3개의 프로세서를 사용할 때의 실행시간을 방법 I을 기준으로 하여 상대·비교하여 나타내었다.

먼저 단일프로세서 상에서 부구조화 알고리즘을 통해 3배 이상의 고속화를 이루고 3개의 프로세서를 통하여 다시 3배에 가까운 효과를 얻는다.

Fig. 5는 10 부재와 200 부재 트러스에 대해 적용된 각 방법들의 실행시간을 그래프화하여 나타내었다.

6. 결 론

최근 컴퓨터의 고성능화로 구조공학의 단위 프로세스의 처리에는 큰 어려움이 없다. 그러나 갈수록 공학 전반의 환경은 여러 인접한 전문 영역의 병행처리와 통합화 방향으로 흐르고 있어, 보다 강력하고 경제적인 전산자원을 필요로 한다.

슈퍼컴퓨터의 활용이나 최근 병렬처리에 대한 활발한 연구는 단일 프로세서가 갖는 연산처리의 한계를 극복하기 위한 자연스런 흐름이라 할 수 있다.

그러나 일반적으로 병렬컴퓨터는 고가에다 용도마저 국한되어있어 사용상의 제약을 받는다. 이에 본 연구는 다양한 수준의 컴퓨터를 호스트로 할 수 있을 뿐 아니라 유연한 확장성을 갖추고 있는 소결합 구조의 MIMD형 트랜스퓨터 병렬시스템을 이용하여 구조최적화를 수행하고 타당성 및 효율성을 검증하므로써 급격히 변화하는 공학 환경에서 상기한 요구에 효율적으로 부응할 수 있는 대안을 제시하였다.

여기서 전체 최적화과정의 대부분을 차지하는 거동 및 민감도 해석 알고리즘은 영역단위의 병렬성을 갖는 부구조화에 근거하고 하드웨어 구성에 맞춰 변환 재구성하였다.

각 노드간 통신은 정적응축과 설계도함수계산에 한정하여 그 횟수를 최소화하고 그들을 동기화하므로써 개별메모리형 연산모델의 약점인 통신비용의 문제를 보완하였다.

PC를 호스트로 하여 구조최적화 분야에서 흔히 취해지는 시험 모델에 적용한 수치실험은 고속화 효율성 면에서 상당히 고무적인 결과를 보여준다.

Table 10 CPU time(sec) and speed up

Algorithm / Data	METHOD I	METHOD II	PRESENT
Total CPU time	12309	3716.8	1264.3
Relative Speed-up	1.0	3.31	9.78

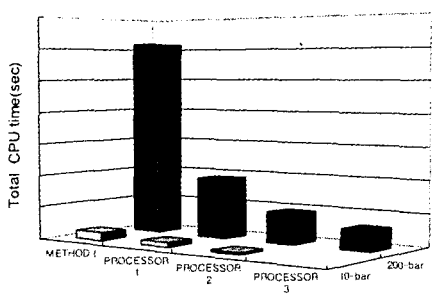

Fig. 5 Comparison of each truss results

감사의 글

본 연구는 한국과학재단의 특정기초연구과제 (95-0600-03-02-2)의 일부분이며, 재단의 지원에 감사드립니다.

참 고 문 헌

1. C. M. Foley and S. Vinnakota, "Parallel Processing in the Elastic Nonlinear Analysis of High-Rise Frameworks", *Computers & Structures*, Vol. 52, 1994, pp.1169~1179
2. M. E. M. El-Sayed and C. K. Hsiung, "Optimum Structural Design with Parallel Finite Element Analysis", *Computers & Structures*, Vol. 40, 1991, pp.1469~1474
3. O. L. Mangasarian, "Parallel Gradient Distribution in Unconstrained Optimization", *SIAM J. Control and Optimization*, Vol. 33,

1995, pp.1916~1925

4. M. E. M. Elsayed and C. K. Hsiung, "Design Optimization with Parallel Sensitivity Analysis on the CRAY X-MP", *Structural Optimization*, Vol. 3, 1991, pp.247~251
5. Adeli, H. & Kamal, O., "Concurrent Optimization of Large Structures : Part I -Algorithms", *Journal of Aerospace Engineering, ASCE*, Vol. 5, 1992, pp.79~90
6. B. C. Waston and A. K. Noor, "Nonlinear Structural Analysis on Distributed-Memory *Computers", Computers & Structures*, Vol. 58, 1996, pp.233~247
7. Haug. E. J. & Arora. J. S., *Applied Optimal Design*, John Wiley & Sons, 1976, p.506
8. Lou Baker & Bradley J. Smith, *Parallel Programming*, McGraw-Hill, 1996, p.386
9. Flynn, M. J., "Very High-Speed Computing Systems", *Proceedings of the IEEE*, Vol. 54, 1966, pp.1901~1909

(접수일자 : 1999. 4. 19)