한국전산구조공학회
1998년도 가을학술발표회 논문집

PC level 병렬 구조해석법 개발을 위한 PCG 알고리즘

PCG Algorithms for Development of PC level Parallel Structural Analysis Method

박 효 선* 박 성 무** 권 윤 한***

Park, Hyo Seon Park, Sung Moo Kwon, Yun Han

ABSTRACT

The computational environment in which engineers perform their designs has been rapidly evolved from coarse serial machines to massively parallel machines. Although the recent development of high-performance computers are available for a number of years, only limited successful applications of the new computational environments in computational structural engineering field has been reported due to its limited availability and large cost associated with high-performance computing.

As a new computational model for high-performance engineering computing without cost and availability problems, parallel structural analysis models for large scale structures on a network of personal computers (PCs) are presented in this paper. In structural analysis solving routine for the linear system of equations is the most time consuming part. Thus, the focus is on the development of efficient preconditioned conjugate gradient (PCG) solvers on the proposed computational model. Two parallel PCG solvers, PPCG-I and PPCG-II, are developed and applied to analysis of large scale space truss structures.

1. 서 론

공학분야에서 다루어지고 있는 문제들의 규모가 대형화하여 자유도 및 부재의 수가 증가하고 있으며 이러한 구조물의 정적, 동적, 비탄성 구조반응 조절을 위한 기존의 구조해석법에 의한 반복적 설계 과정은 엔지니어링 업무의 효율은 물론 설계 질을 고려하여 재고되어야 한다. 이를 위해 고성능 전산기의 도입과 이에 따른 효율적 구조해석 및 반응조절 알고리즘의 개발이 필요하지만 고성능 전산기의 이용은 많은 비용이 요구될 뿐만 아니라 이들의 운영체제가 일반인들에게 익숙하지 않은 UNIX 체제가 대부분을 차지하고 있기 때문에 능숙하게 사용하는 데에는 많은 시간과 어려움이 따르게 된다. 그러나 각 연구소나 기업체에서 기존에 보유하고 있는 윈도우 환경의 개인용 컴퓨터를 네트워크로 상호 연결함으로써 가상의 대형 병렬 컴퓨터를 구성하면 고성능 전산기를 구현하는 것이 가능하게 될 뿐

* 영남대학교 건축공학과 조교수
** 영남대학교 건축공학과 부교수
*** 영남대학교 건축공학과 석사과정

만 아니라 비용 면에서도 많은 이득을 볼 수 있다. 또한 대부분의 사용자에게 친숙한 운영체제를 사용하기 때문에 제어가 쉽다는 이점이 있다. 반면 이러한 가상의 병렬 고성능 전산기를 구현하기 위해서는 네트워크와 병렬 연산에 관한 지식이 필요하며 구조 해석에 있어서 가장 많은 계산량이 요구되는 선형방정식 $\mathbf{K}\mathbf{x} = \mathbf{f}$ 를 푸는 방정식 해법의 병렬화가 요구된다.

병렬계산의 개념과 윈도우 환경에서의 병렬계산의 기본 성능 평가에 대해서는 참고문헌 [1],[2],[3],[4]에 소개되어 있으므로 본 논문에서는 대형 구조물의 해석에 많이 적용되는, PCG (preconditioned conjugate gradient) 알고리즘의 병렬화와 네트워크로 연결된 윈도우95 환경에서의 병렬 PCG 알고리즘의 병렬 성능 및 특성을 분석하고자 한다.

2. Sequential PCG 알고리즘

PCG 알고리즘은 선형 대수식을 풀기 위한 반복법의 하나이며 제어 매트릭스 (preconditioning matrix)와 기본적인 공액경사도법 (conjugate gradient method)의 조합으로 이루어졌다[10]. PCG 알고리즘은 최근 두 가지 이유에 의해 구조해석 분야에서 널리 사용되고 있다. (1) 적은 저장량만을 필요로 하므로 대형 문제에 대해서 계산상 효율적이고, (2) 알고리즘 내의 대부분의 계산들이 매트릭스와 벡터의 곱셈들로 이루어졌기 때문에 직접법보다 병렬화가 용이하다. 이러한 이유에 의해 대형의 방정식을 풀기에 매우 적합하다는 것이 많은 연구에 의해 증명되었다.[5),6),14)]

2.1 일반적인 PCG 알고리즘

기본적인 공액경사도법에서 식 (1)과 같은 구조해석 문제는 함수 $\phi(\mathbf{x})$를 최소화함으로서 계산되며 식으로 표현하면 식(2)와 같이 표현된다.

$$\mathbf{K}\mathbf{x} = \mathbf{f} \tag{1}$$

여기서 $\mathbf{K}$는 강성매트릭스이고, $\mathbf{f}$는 하중벡터, 그리고 $\mathbf{x}$는 미지의 변위벡터이다.

$$\min \ \phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{K}\mathbf{x} - \mathbf{x}^{\mathrm{T}}\mathbf{f} \tag{2}$$

$\mathbf{x} = \mathbf{K}^{-1}\mathbf{f}$로 치환하여 풀면 ϕ의 최소값은 $-\frac{1}{2}\mathbf{f}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{f}$가 되므로, 최소화 문제의 해는 $\mathbf{K}\mathbf{x} = \mathbf{f}$를 푸는 것과 동일하게 된다. 공액 경사도법은 해에 대한 초기치를 주어 수용할 만한 해에 도달할 때까지 반복하는 과정이다. 각 반복회에서 ϕ는 모든 이전의 탐색 방향에 대해 'K-공액'하는 방향으로 해를 탐색함으로서 줄어든다. 즉, 만일 현재 탐색 방향이 벡터 $\mathbf{p}_k$이고 식 (3)을 만족한다면 모든 이전 탐색 방향은 $\mathbf{K}$매트릭스와 공액한다고 말한다.

$$\mathbf{p}^{\mathrm{T}}{}_k \, \mathbf{K}\mathbf{p}_i = 0 \qquad i = 1, \ldots, k-1 \tag{3}$$

선택된 실제 탐색 방향은 가장 빨리 하강하는 방향 또는 음의 경사도 방향에 가장 가깝게 되는 $\mathbf{K}$ 매트릭스와 공액한 방향이다. 여기서 음의 경사도 방향은 현재의 잔차 (residual) 즉, 현재 해를 이용한 식 (1)의 오른쪽 항과 왼쪽 항의 차이에 의해 계산된다. 주어진 반복회에 대한 수정된 해는 탐색방향에서 최적 거리 α 만큼 이동함으로서 구해진다. 이러한 방법은 최속강하법 (steepest descent method)에서 보다 많은 이점을 가진다. $\mathbf{p}_i$는 모두 선형독립이고 따라서 n step내에 거의 정확한 해가 보장된다. 여기서 n은 미지수의 개수이다. 또한 각 반복회마다 경사도 방향이 거의 유사하게 됨으로 인해 수렴속도가 매우 느리게 되는 최속강하법에서 발생될 수 있는 단점을 극복할 수 있다. 정확한 해를 구하기 위해 필요한 수많은 반복회수와 반복 과정에서의 roundoff error (반올림으로 인한 error)로 인한 해의 잔차가 수용할 수 있을 만큼 작게될 때 종료된

다. 수용할 만한 해에 수렴하기 위해 필요한 반복회수를 줄이기 위한 방안으로는 강성 매트릭스 **K**를 미리 구속할 수 있고 이러한 방법에 의해 PCG 알고리즘이 유도된다. PCG 알고리즘에서, 식(1)은 제어 매트릭스 **M**을 양 쪽 항에 미리 곱함으로써 구성된다.

$$\mathbf{M}^{-1}\mathbf{K}\mathbf{x} = \mathbf{M}^{-1}\mathbf{f} \tag{4}$$

제어 매트릭스 **M**은 대칭인 양정 매트릭스이고 강성매트릭스 **K**로부터 구해지며, 본 논문에서는 여러 가지 제어매트릭스 중 수렴성과 계산량의 관점에서 가장 무난하며 구성이 쉬운 Jacobi 제어 매트릭스(대각 제어 매트릭스)를 사용하였다[9].

일반적인 PCG 알고리즘 흐름도는 다음과 같다.

제어매트릭스 구성 $\mathbf{C} = \mathbf{M}^{-1}$ (5-1)

초기값 $\mathbf{x}_0$ 선택 (5-2)

$\mathbf{r}_0 = \mathbf{f} - \mathbf{K}\mathbf{x}_0$ 계산 (5-3)

$\mathbf{h}_0 = \mathbf{C}\ \mathbf{r}_0$ 계산 (5-4)

$\mathbf{p}_0 = \mathbf{h}_0$ (5-5)

for $i = 0, 1, 2, \cdots$ (until convergence)

$$\alpha_i = \frac{\mathbf{r}_i \cdot \mathbf{h}_i}{\mathbf{K}\mathbf{p}_i \cdot \mathbf{p}_i} \tag{5-6}$$

$$\mathbf{r}_{i+1} = \mathbf{f} - \mathbf{K}\mathbf{x}_{i+1} = \mathbf{r}_i - \alpha_i \mathbf{K}\mathbf{p}_i \tag{5-7}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i \tag{5-8}$$

$$\mathbf{h}_{i+1} = \mathbf{C}\ \mathbf{r}_{i+1} \tag{5-9}$$

$$\frac{\mathbf{r}_{i+1} \cdot \mathbf{h}_{i+1}}{\mathbf{r}_0 \cdot \mathbf{h}_0} \le \eta \quad \text{then stop, else continue} \tag{5-10}$$

$$\beta_i = \frac{\mathbf{r}_{i+1} \cdot \mathbf{h}_{i+1}}{\mathbf{r}_i \cdot \mathbf{h}_i} \tag{5-11}$$

$$\mathbf{p}_{i+1} = \mathbf{h}_{i+1} + \beta_i \mathbf{p}_i \tag{5-12}$$

continue iterative cycle

여기서 $\mathbf{p}_i$는 공액 탐색 방향이고 α_i는 $\mathbf{p}_i$ 방향으로 이동 거리를 제어하는 스칼라 값이며, η는 수렴오차이다.

제어 매트릭스 **M**은 식 (5-9)에 나타난 것처럼 부가적인 계산이 필요하지만 알고리즘의 수렴성을 향상시키는데 도움이 된다. 제어 매트릭스 **M**이 강성매트릭스 **K**와 비슷할수록 더 빠른 수렴율을 얻게 되나, 더 많은 계산량이 요구된다(**M**=**K**인 경우 단 1회 반복만으로도 수렴이 가능하다).

본 논문에서는 알고리즘의 병렬화의 편의성을 위해 Hughes에 의해 제안된 EBE(element by element)기법을 PCG 알고리즘에 적용하여 전체 강성 매트릭스를 구성하지 않고 각각의 부재에 대한 부재강성매트릭스만을 구성하여 계산하였다[7,8].

3. 병렬 계산 모델과 Parallel PCG 알고리즘

일반적으로 PCG 알고리즘에서 1회 반복시 소요되는 계산량은 크게 하나의 매트릭스-벡터 계산($\mathbf{K}\ \mathbf{p}_k$)과 2개의 내적 계산($\mathbf{p}^T{}_k \cdot ([\mathbf{K}]\mathbf{p}_k)$와 $\mathbf{r}^T{}_{k+1} \cdot \mathbf{z}_{k+1}$), 세 개의 벡터계산 ($\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$, $\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k [\mathbf{K}]\mathbf{p}_k$, $\mathbf{p}_{k+1} = -\mathbf{z}_{k+1} + \beta_k \mathbf{p}_k$)으로 구성된다. PCG 알고리즘의 계산량 중에서

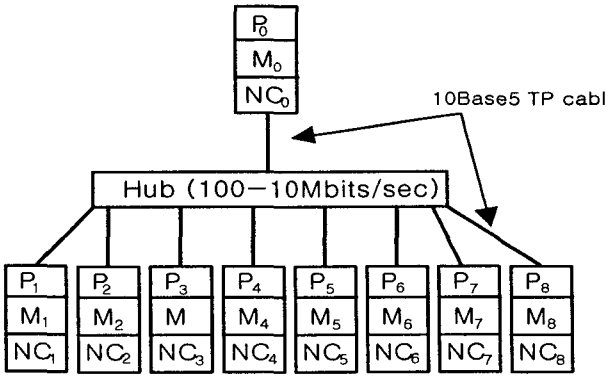
매트릭스-벡터 계산이 대부분을 차지하며 이것이 알고리즘의 판단 기준이 되며 병렬계산에 있어서 이 부분을 어떻게 분할하여 계산하느냐가 관건이 된다. PCG 알고리즘에서는 전체 강성 매트릭스를 조합할 필요가 없으므로, 대부분의 계산 과정은 절점과 부재 단위로 수행된다. 계산 시간과 필요 메모리량은 부재와 절점의 수에 따라 선형적으로 증가하게된다(반면 직접법은 비선형적으로 증가하게된다).[9]

이장에서는 본 연구를 위해 구성된 병렬시스템 모델에 대해 설명하였으며, 문제분할 방법 및 데이터 전송량에 의해 병렬 PCG 알고리즘 모델 I, II로 나누어 각 알고리즘의 특성을 분석하였다.

3.1. 병렬 시스템 모델

병렬 구조해석법 개발을 위한 병렬시스템의 구성은 그림 1과 같이 본 연구실에서 보유하고 있는 개인용 컴퓨터들(windows95 환경)을 네트워크로 연결하여, 가상 병렬시스템 구성을 가능하게 하는 소프트웨어 패키지인 PVM[11]을 이용하여 병렬시스템을 구성하였다.

병렬 계산 모델을 적용시키기 위한 병렬 시스템 모델은 crowd 프로그래밍 패러다임[11] 중의 하나인 Master - Slave 모델로서 P_0는 Master의 역할을 수행하고 P_1~P_8은 Slave 역할을 수행하게 되며 예제 적용시 연결대수를 2대, 4대, 8대로 구분하여 실험하였다.



P : 펜티엄 166MHz CPU
M : 32Mbyte 메인 메모리
NC : Ethernet 네트워크카드(10Mbps)

그림 1 병렬 시스템 모델의 하드웨어 구성도

3.2. 병렬 PCG 알고리즘 모델-I

병렬 PCG 알고리즘 모델-I(이하 PPCG-I)에서 문제의 분할과 할당은 부재수(m)와 전체 자유도수(nd)를 Slave 수(ns)로 나눈 정수만큼의 각종 데이터들(강성 매트릭스(K)와 방향 벡터(p)를 구성하기 위한 데이터들)을 분배하여 각 Slave들이 거의 동일한 데이터량을 가지도록 분할한다. 즉, 각 Slave들의 계산량은 [m/ns]과 [nd/ns]로 분할되므로 전체계산량을 Slave 수로 나눈 만큼의 양이 된다.

PPCG-I 알고리즘은 다음과 같다.

(1) 각 Slave에서 제어매트릭스 $C^{G(s)}$구성한 후, Master에 전송하여 Master에서 병합하고 다시 각 Slave에 해당하는 절점부분의 C^s를 전송 받는다. *[nd×ns + (nd/ns)×ns]*
(2) 각각의 벡터들을 초기화 시킨다.

$\mathbf{x}_0^s = 0$, $\mathbf{r}_0^s = \mathbf{f}^s - \mathbf{K}_e^s \mathbf{x}_0^s$, $\mathbf{h}_0^s = \mathbf{C}^s \mathbf{r}_0^s$, $\mathbf{p}_0^s = \mathbf{h}_0^s$, $\delta_0^s = \mathbf{r}_0 \cdot \mathbf{h}_0$

(3) δ_0^s를 Master로 전송하고 Master는 이들을 합한다 $\left[\delta_0 = \sum_{s=1}^{ns} \delta_0^s\right]$. *[ns]*

(4) 각 Slave들은 앞에서 계산된 방향벡터 $\mathbf{p}^s$를 Master에 전송하고 Master는 전체 방향벡터 $\mathbf{p}^G$를 구성한 뒤 다시 각 Slave들에게 $\mathbf{p}^G$를 전송한다.*[(nd/ns)×ns + nd×ns]*

(5) 각 Slave들은 $\mathbf{q}^{G(s)} = \mathbf{K}_e^s \mathbf{p}^G$(해당 d.o.f부분만 계산)을 계산하고 $\mathbf{q}^{G(s)}$를 Master로 전송, Master는 이들을 병합($\mathbf{q}^G$)하여 다시 각 Slave들에게 필요한 d.o.f. 부분($\mathbf{q}^s$)으로 나누어 전송한다. *[nd×ns + (nd/ns)×ns]*

(6) 각 Slave에서 $\alpha_n^s = \mathbf{r}_i^s \cdot \mathbf{h}_i^s$와 $\alpha_d^s = \mathbf{p}_i^G \cdot \mathbf{q}_i^s$ 계산하여 Master로 전송한다. Master는 $\alpha = \frac{\sum \alpha_n^s}{\sum \alpha_d^s}$를 계산하고 α를 각 Slave로 전송한다. *[ns + ns + ns]*

(7) 각 Slave에서 다음을 Update한다.

$\mathbf{x}_{i+1}^s = \mathbf{x}_i^s + \alpha \mathbf{p}_i^G$, $\mathbf{r}_{i+1}^s = \mathbf{r}_i^s - \alpha \mathbf{q}^s$, $\mathbf{h}_{i+1}^s = \mathbf{C}^s \mathbf{r}_{i+1}^s$

(8) 각 Slave에서 $\beta_n^s = \mathbf{r}_{i+1}^s \cdot \mathbf{h}_{i+1}^s$를 계산하여 Master로 전송하고 Master는 이를 전송받아 $\beta = \frac{\sum \beta_n^s}{\sum \alpha_n^s}$를 계산하고 β를 다시 각 Slave로 전송한다. *[ns + ns]*

(9) Master는 $\frac{\sum \alpha_n}{\delta_0} \leq \eta$을 만족하는지를 계산한다. 만약 만족하지 않으면 step(4)로 돌아가서 계속 반복을 진행하고 만족한다면 stop한다. 이때 수렴여부를 각 Slave들에게 전송한다. *[ns]*

(10) 각 Slave들이 수렴만족 신호를 받았다면 stop하고 그렇지 않으면 $\mathbf{p}_{i+1}^s = \mathbf{h}_{i+1}^s + \beta \mathbf{p}_i^G$(해당 d.o.f.부분만 계산)를 계산한 뒤 step(4)로 돌아가서 반복을 계속한다.

여기서 첨자 G는 전체 d.o.f만큼의 데이터를, s는 nd/ns 만큼의 분할된 데이터를, G(s)는 각 Slave에서 자체적으로 계산된 전체 d.o.f. 크기의 데이터를 나타내며, $\mathbf{K}_e^s$는 각 Slave들에게 할당된 부재들의 강성매트릭스를 나타낸다 (3-D truss인 경우 6×6 매트릭스). 그리고 [ ]부분은 통신량을 나타낸다.

3.3. 병렬 PCG 알고리즘 모델-II

병렬 PCG 알고리즘 모델-II(이하 PPCG-II)는 PPCG-I와 같이 문제분할을 부재수와 자유도수를 서로 무관하게 분할하는 것이 아니라, 먼저 절점 수를 Slave 수로 나눈 뒤, 분할된 절점들을 구성하고 있는 부재들에 대한 데이터를 각 Slave들에게 할당하는 방법을 사용하였다. PPCG-I 알고리즘에서의 통신은 Master와 Slave간의 통신인데 비해, PPCG-II에서는 데이터를 공유하고 있는 각 인접 Slave 간에 필요한 데이터들만 통신을 하도록 함으로써 전체 통신량을 줄였다.

PPCG-II 알고리즘은 다음과 같다.

(1) 각 Slave에서 제어매트릭스 $\mathbf{C}^s$구성한다. *[통신할 필요 없음]*

(2) 각각의 벡터들을 초기화 시킨다.

$\mathbf{x}_0^s = 0$, $\mathbf{r}_0^s = \mathbf{f}^s - \mathbf{K}_e^s \mathbf{x}_0^s$, $\mathbf{h}_0^s = \mathbf{C}^s \mathbf{r}_0^s$, $\mathbf{p}_0^s = \mathbf{h}_0^s$, $\delta_0^s = \mathbf{r}_0 \cdot \mathbf{h}_0$

(3) δ_0^s를 Master로 전송하고 Master는 이들을 합한다 $\left[\delta_0 = \sum_{s=1}^{ns} \delta_0^s\right]$. *[ns]*

(4) 각 Slave들은 앞에서 계산된 방향벡터 $\mathbf{p}_i^s$중 인접 Slave와 공유하고 있는 부재와 연결된 절점(경계부분) 데이터들($\mathbf{p}_i^{ssb}$)을 인접 Slave에 송신하고, 또 인접 Slave로부터 필요한 데이터($\mathbf{p}_i^{srb}$)를 수신하여 $\mathbf{p}_i^s$와 함께 $\mathbf{p}_i^{Gs}$에 저장한다. *[경계부분(sb) + 경계부분(rb)]*

(5) 각 Slave들은 $\mathbf{q}_i^s = \mathbf{K}_e^s \mathbf{p}_i^{Gs}$(해당 d.o.f부분만 계산)을 계산한다. *[통신할 필요 없음]*

(6) 각 Slave에서 $\alpha_n^s = \mathbf{r}_i^s \cdot \mathbf{h}_i^s$와 $\alpha_d^s = \mathbf{p}_i^{Gs} \cdot \mathbf{q}_i^s$ 계산하여 Master로 전송한다. Master는 $\alpha = \frac{\sum \alpha_n^s}{\sum \alpha_d^s}$를 계산하고 α를 각 Slave로 전송한다. *[ns + ns + ns]*

(7) 각 Slave에서 다음을 Update한다.

$\mathbf{x}^{s}_{i+1} = \mathbf{x}^{s}_{i} + \alpha\, \mathbf{p}^{Gs}_{i}$, $\mathbf{r}^{s}_{i+1} = \mathbf{r}^{s}_{i} - \alpha \mathbf{q}^{s}$, $\mathbf{h}^{s}_{i+1} = \mathbf{C}^{s}\, \mathbf{r}^{s}_{i+1}$

(8) 각 Slave에서 $\beta^{s}_{n} = \mathbf{r}^{s}_{i+1} \cdot \mathbf{h}^{s}_{i+1}$를 계산하여 Master로 전송하고 Master는 이를 전송받아 $\beta = \frac{\sum \beta^{s}_{n}}{\sum \alpha^{s}_{n}}$를 계산하고 β를 다시 각 Slave로 전송한다. *[ns + ns]*

(9) Master는 $\frac{\sum \alpha_{n}}{\delta_{0}} \le \eta$을 만족하는지를 계산한다. 만약 만족하지 않으면 step(4)로 돌아가서 계속 반복을 진행하고 만족한다면 stop한다. 이때 수렴여부를 각 Slave들에게 전송한다. *[ns]*

(10) 각 Slave들이 수렴만족 신호를 받았다면 stop하고 그렇지 않으면 $\mathbf{p}^{s}_{i+1} = \mathbf{h}^{s}_{i+1} + \beta\, \mathbf{p}^{Gs}_{i}$(해당 d.o.f.부분만 계산)를 계산한 뒤 step(4)로 돌아가서 반복을 계속한다.

여기서 첨자 Gs는 할당된 d.o.f. + 경계부분의 d.o.f만큼의 데이터를, s는 각 Slave에 할당된 양만큼의 데이터를, ssb는 다른 Slave로 송신하는 부분을, srb는 수신 받은 부분을 나타낸다.

4. 예제 적용 및 결과

두 개의 병렬 PCG알고리즘을 space truss 예제(그림 2,그림 3)에 적용하여 각각의 알고리즘의 병렬성능을 측정하였다. 그림 2의 예제는 147층 초고층 space 강구조물이며 1,801개의 절점과 8,904개의 부재를 가지며 상세도는 참고문헌 [15]에 나타나 있다. 그림 3의 예제는 본 연구를 위해 새롭게 제작된 space truss 구조물을 구성하는 단위 모듈이며 이들 모듈을 적용하여 8,712개의 절점과 56,160개의 부재를 갖는 120층 규모의 space truss 구조물을 구성하였다. 이들 두 가지 space truss예제를 적용시킨 Sequential PCG (SPCG) 알고리즘과 두 개의 Parallel PCG 알고리즘의 성능에 대한 결과가 표 1과 그림 4에 나타나있다. 표 1은 각 알고리즘이 1회 반복하는데 소요되는 시간을 나타내며 그림 4에서는 병렬성 평가 항목의 하나인 Speed-up(성능향상도)[12]을 나타낸다.

이들 결과를 분석해보면 PPCG-II가 PPCG-I보다 성능이 좋다는 것을 알 수 있으며 이러한 이유로는 앞

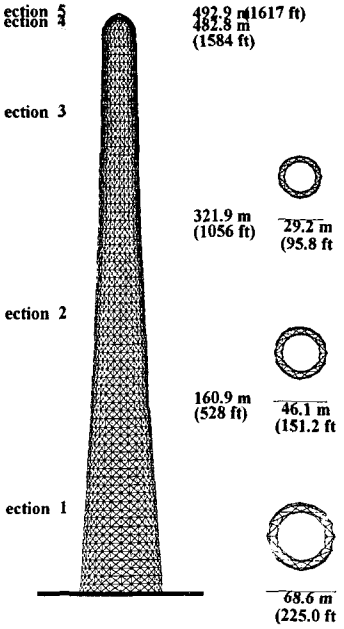
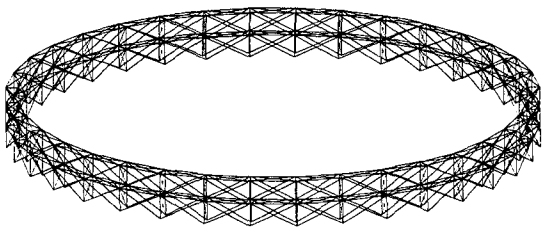
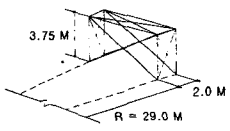


그림 2. 147층 space truss



a) 2개층 단위 모듈



b) 단위 모듈 상세

그림 3. 120층 space truss를 구성하고 있는 상세 모듈

알고리즘	적용예제	1대	Slave 2대	Slave 4대	Slave 8대
SPCG	1801 절점	0.43	–	–	–
	8712 절점	3.60			
PPCG-I	1801 절점	–	0.99	1.28	2.45
	8712 절점		4.59	5.40	9.77
PPCG-II	1801 절점	–	0.43	0.54	082
	8712 절점		2.31	1.27	1.38

표 1. 1회 반복시의 각 알고리즘의 소요시간(초)

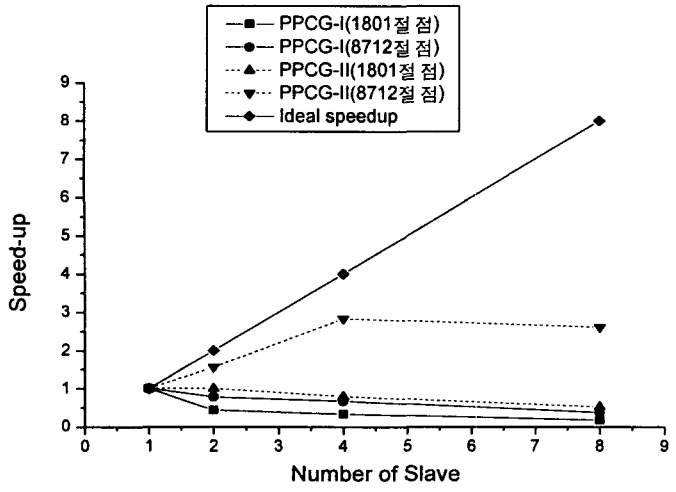


그림 4. 각 예제에 대한 알고리즘이 성능향상도(Speed-up)

장의 알고리즘에 나타난 바와 같이, PPCG-II의 통신량이 PPCG-I의 통신량에 비해 적고 모든 Slave들이 Master에 많은 데이터들을 동시에 전송함으로 인해 bottleneck과 같은 통신장애 현상이 발생하기 때문이다. 따라서 Slave 연결대수의 증가로 인한 계산량 감소에 의한 시간단축보다는 경계조건의 증대로 인한 통신시간의 증대가 훨씬 큰 비중을 차지한다는 것으로 분석할 수 있다. 그리고 1801 절점 space truss 예제의 계산 결과 두 병렬 알고리즘 모두가 1대의 컴퓨터에서 수행된 PCG 알고리즘에 비해 성능이 떨어진 것을 볼 수 있다. 이러한 이유로는 PPCG 알고리즘이 동기화 부분(synchronization 부분 - α, β, $\mathbf{p}$, $\mathbf{q}$의 통신부분)[13]을 내포하고 있기 때문이며 이것으로 인해 문제규모가 적은 경우 병렬계산이 오히려 성능이 떨어진다는 것을 알 수 있다.

5. 결 론

PC level에서의 병렬 구조해석을 위한 PCG 알고리즘을 PPCG-I과 PPCG-II로 나누어 space truss 예제에 적용시킴으로써 각 알고리즘의 병렬성능을 분석한 결과를 요약해보면 다음과 같다.

(1) Slave 연결대수를 증가시킬수록 PC 1대가 부담하는 계산량은 줄어들지만 반면, 통신량은 이에 반해 증가함을 알 수 있다.

(2) 통신량과 동기화부분이 적은 PPCG-II 알고리즘이 더 좋은 성능을 보였으며 이는 병렬 계산에 있어서 통신에 대한 비중이 매우 크다는 것을 알 수 있다.

(3) 해석하고자하는 문제의 규모가 적은 경우에서는 PPCG 알고리즘을 이용한 병렬 계산이 비효율적인 것을 알 수 있으며 문제의 규모에 따라 최대의 성능을 발휘하기 위한 Slave 연결대수가 다르게 나타날 수 있다는 것을 알 수 있다.

(4) PPCG-II 알고리즘에서 Master의 역할이 반복구간 내에서는 수렴 여부의 점검만을 수행하기 때문에 CPU의 휴지상태가 대부분을 차지하므로 Master-Slave 병렬 시스템 모델에서는 비효율적이라 할 수 있다. 따라서 Parallel I/O 기법을 적용하여 Master의 기능을 각 Slave가 분산하여 분담할 필요가 있다.

(5) 앞으로 수행되어야 할 과제로는 보다 향상된 네트워크 환경 하에서 이들 병렬 PCG 알고리즘의 성능

분석이 필요하다.

6. 참 고 문 헌

1. 박효선, "대형 구조물을 위한 병렬 구조해석 및 설계", *전산구조공학회지*, 제9권, 제3호, pp 47-53, 1996년 9월
2. 권윤한, 박효선, "병렬구조해석 및 설계를 위한 PVM", *전산구조공학회지*, 제11권, 제3호, 1998년 9월.
3. Alexandre Alves, "Parallel Computing-Windwos Style", *BYTE*, pp 169-170, May, 1996
4. A. K. Noor, " New Computing Systems and Future High-Performance Computing Environment and Their Impact on Structural Analysis and Design", *Computers & Structures*, Vol. 64, No. 1-4, pp1-30, 1997.
5. Foresti, S., Hassanzadeh, S., Murakami, H., and Sonnad, V., "A Comparison of Preconditioned Iterativ Methods using Rapid Operatio Application against Direct Solution Methods", *Int. J. Numer. Methods Eng.*, Vol. 32, pp. 1137-1144, 1991.
6. Dickinson, J. K. and Forsyth, P. A., "Preconditioned Conjugate Gradient Methods for Three-Dimensional Linear Elasticity", *Int. J. Numer. Methods Eng.*, Vol. 37, pp. 2211-2234, 1994.
7. T.J.R. Hughes, J. Winget, I. Levit and T. E. Tezduyar, "New alternating direction procedures in finite element analysis based upon EBE approximate factorization", in S. Atluri and N. Perrone(eds.), Recent Developments in Computer Methods for Nonlinear Solid and Structural Mechanics, *ASME Applied Mechanics Symposia Series*, New York, 1983, pp 75-109.
8. E. Barragy and Graham F. Carey, "A Parallel Element-by-Element Solution Scheme", *Int. J. Numer. Methods Eng.*, Vol. 26, pp 2367-2382, 1998
9. Are Magnus Bruaset, *A Survey of Preconditioned iterative methods*, Pitman Research Notes in Mathematics Series, Longman Scientific & technical, 1995
10. G. H. Golub and G. F. Van Loan, *Matrix Computations*, 2nd Ed., The Johns Hopkins University Press, Baltimore, Md., 1989
11. Al Geist, Adam Beguelin, Jack Dongarra, Robert Manchek and Vaidy Sunderam, PVM : Parallel Virtual Machine; A Users' Guide and Tutorial for Networked Parallel Computing, The MIT Press, Cambridge, Massachusetts, 1994
12. Dimitri P. Bertsekas, John N. Tsitsiklis, *Parallel and Distributed Computation - Numerical Methods*, Prentice-Hall International, 1989
13. Vipin Kumar, Ananth Grama, Anshul Gupta and George Karypis, *Introduction to Parallel Computing*, The Benjamin/Cummings Publishing, 1994
14. Hyo Seon Park and Hojjat Adeli, "Distributed Neural Dynamics Algorithms fo Optimization of Large Steel Structures", *Journal of Structural Engineering*, ASCE, Vol. 123, No. 7, pp. 880-888, July, 1997
15. Hojjat Adeli and Hyo Seon Park, *Neurocomputing for Design Automation*, CRC Press, pp. 129-131, 1998.