영역 분할에 의한 SIMPLER 모델의 병렬화와 성능 분석

곽호상*1, 이상산*1

Implementation and Performance Analysis of a Parallel SIMPLER Model Based on Domain Decomposition

Ho Sang Kwak, Sangsan Lee

Parallel implementation is conducted for a SIMPLER finite volume model. The present parallelism is based on domain decomposition and explicit message passing using MPI and SHMEM. Two parallel solvers to tridiagonal matrix equation are employed. The implementation is verified on the Cray T3E system for a benchmark problem of natural convection in a sidewall-heated cavity. The test results illustrate good scalability of the present parallel models. Performance issues are elaborated in view of convergence as well as conventional parallel overheads and single processor performance. The effectiveness of a localized matrix solution algorithm is demonstrated.

Key Words: 영역분할(Domain Decomposition), 메시지 전달 (Message Passing), 병렬 계산 효율(Computational Efficiency), 확장성 (Scalability), 병렬 행렬 해법 (Parallel Matrix Solver), 수렴성(Convergence)

1. 서론

복수의 프로세서를 장착한 저가의 고성능 병렬 컴퓨터의 보급으로 전산유체역학은 새로운 전기를 맞이하고 있다. 실용적 도구로서의 위치가 확고해지고 난류, 이상유동, 연소와 같이 막대한 연산량과 기억용량이 요구되는 초대형 과제에 도전할 수 있는 기반이 조성되고 있다. 병렬 컴퓨터의 핵심적인 장점은 프로세서 수의 증가로 연산 성능 향상과 기억용량 확대를 동시에 도모할 수 있는 확장성이다. 이러한 확장성을 향유하기 위한 필수 전제 조건은 고성능 병렬 계산 모델이다.

전산유체역학 분야에서도 병렬 모델 개발에 상당한 노력이 이루어져 왔다 [1-6]. 병렬 계산을 지향한 새로운 알고리즘의 개발과 기존의 순차형 모델의 병렬화가 광범위하게 진행되고 있다. 문제는 병렬화가 대단히 복잡한 성격의 작업이란 점이다. 실제로 효율적인 병렬화는 하드웨어(프로세서의 성능, 메모리 구조, 접속망의 성능), 해석 알고리즘, 원본 코드의 구조, 문제의 특성에 크게 의존한다. 이식성과 효율성을 갖춘 병렬 모델 개발 과정에서 여러 가지 쟁점이 제기되고 있으며, 이를 해소하기 위한 다양한 관점의 수치 실험이 요구되고 있다.

이 연구에서는 열유체 해석에 널리 사용되는 SIMPLER[7] 알고리즘의 순차형 모델을 병렬화하였다. 연구 목적은 기존의 유한체적 또는 유한차분 모델의 병렬화와 성능 분석에 도움이 될 지침을 제공하는 것이다. 원본 코드의 구조와 특성 분석에 기초하여 수립된 병렬화 전략과 방법론을 기술하였다. 측벽 가열형 용기 내에서의 자연 대류 문제를 대상으로 수치 실험을 실시하여, 개발된 모델의 신뢰성을 검증하였다. 이 논문에서는 성능에 영향을 미치는 제요소를 독립적으로 고려한 새로운 형태의 병렬 계산 성능 분석을 제안하였고 이 방법에 입각하여 병렬 모델의 효율성을 평가하고 제기되는 논점을 정리하였다.

*1 정회원, 전자통신연구원 슈퍼컴퓨터센터

2. 순차 모델

2.1. 수치 모델과 예제

병렬화 대상 코드는 비정상 자연대류 해석에 성공적으로 적용되었던 SIMPLER 알고리즘을 사용한 유한체적 모델이다[8]. 지배방정식은 비압축성 Boussinesq 유체의 2차원 Navier-Stokes 방정식이다. 공간적 이산화는 엇갈림 유한체적 격자에서 이루어지며 선형항과 비선형 대류항은 각각 중심차분법과 QUICK 기법으로 계산된다. 시간적분 방법은 내재적 Euler법이다.

다수의 subroutine으로 구성된 코드는 다음과 같은 기능집단으로 구분된다. ① Part I: 초기 및 경계조건의 정의, 격자 구성, 수렴 여부 판정, 입출력 제어; ② Part II: 속도, 압력, 보정압력, 스칼라량 계산을 위한 행렬 방정식 계수의 계산; ③ Part III: 행렬 방정식의 해석.

SIMPLER의 주 계산 과정은 행렬식을 만들고 해를 구하는 반복 작업이다. 지배방정식의 이산화로 얻어진 행렬식은 블록 tridiagonal 행렬식으로 변환되고 TDMA(TriDiagonal Matrix Algorithm [7])를 사용하여 해를 구한다.

예제로 측벽 가열형 용기 내에서의 자연대류[9]를 선택하였다. 등온의 유체가 채워져 있는 정사각형 용기의 한쪽 수직벽의 온도를 낮추고 반대편 수직벽의 온도는 올리는 문제이다. 수평벽은 단열되었다. 벤치마크 해가 잘 알려진 Rayleigh 수가 Ra=10^6, Prandtl 수가 Pr=0.71인 경우를 고려하였다. 여기서 $Ra=\alpha g\Delta\theta h^3/\nu\kappa$, $Pr=\nu/\kappa$이다 (h:용기의 높이, ν:동점성 계수, κ:열확산 계수, α:온도팽창 계수, g:중력가속도, $\Delta\theta$:양 수직벽 사이의 온도차).

모든 계산에서 수평과 수직 방향으로 같은 수의 균일 격자를 사용하였다. 이 코드는 비정상 문제도 해석 능력을 가지고 있으나 여기서는 정상해법을 사용하였다. 속도와 온도의 상대 변화량의 최대값과 연속방정식의 만족도가 각각 10^{-4}와 10^{-8} 이하이면 수렴된 것으로 판정하였다.

2.2 하드웨어와 순차 모델의 특성

계산은 128개의 PE(processing element)를 장착한 분산 메모리형 초병렬 컴퓨터 Cray T3E에서 수행하였다. 각 PE는 이론 최고 성능이 0.9 Gflops인 DEC alpha 프로세서와 128 Kbytes의 메모리로 구성된다. PE들은 1.0 Gbps의 양방향 대역폭을 가지는 고속 접속망에 의해 3차원 토러스(torus) 구조로 연결되어 있다.

원본 순차 코드의 특성을 분석하기 위해 Cray T3E의 PE 하나를 사용하여 격자수($N\times N$)를 바꾸어 가며 계산을 수행하였다. 여기서 얻은 결과를 통계 처리하여 얻은 1회 반복 계산당 필요한 소수점 연산횟수 N_{OPS}는 다음과 같다.

$$N_{OPS}=0.850\times10^3\,(N-2)^2. \qquad (1)$$

요구되는 연산량은 경계를 제외한 실제 영역의 격자수 $(N-2)^2$에 비례함을 알 수 있다.

Table 1에 소요된 계산 시간중 전술한 세 기능집단이 차지하는 점유비를 정리하였다. Part II와 III에서 계산시간의 대부분이 소비됨을 알 수 있다. 행렬 해법(Part III)에 소요되는 시간은 N의 증가에 따라 증가하다가 N=128을 정점으로 다소 감소하는데 이 연구의 제한된 수치실험 결과로부터 그 원인을 정확히 파악할 수 없다. 중요한 것은 전체적으로 행렬해법이 가장 큰 계산부하가 걸리는 병렬화의 핵심 부분이라는 점이다.

Table 1. Composition of the elapsed time [%].

N	Part I	Part II	Part III
16	6.9	48.0	45.1
32	5.2	47.5	47.3
64	3.7	38.5	57.8
128	3.4	36.1	60.5
256	3.3	37.2	59.5
362	3.1	39.4	57.5

3. 병렬화

병렬 계산은 순차 계산에서는 발생하지 않는 병렬 오버헤드를 수반한다. 대표적인 것은 ① 프로세서간의 정보 교환을 위한 통신 오버헤드; ② 알고리즘의 병렬화 과정에서 발생하는 추가적인 연산에 의한 산술 오버헤드; ③ 해석 알고리즘의 순차적 본질에 의해 발생하는 공전 프로세서의 존재에 의한 지연 오버헤드; ④ 동일 데이터의 중복 접속에 의한 메모리 오버헤드 등이다 [3]. 이와 함께 프로세서에 할당된 작업량의 차이에서 비롯되는 부하 불균형이 또 하나의 성능 제약요인이다. 병렬 오버헤드의 최소화와 부하 균형 달성이

고성능 병렬 모델 개발의 핵심적 관건이다.

3.1. 병렬화 방법론

이 연구에서 채택한 병렬화의 골간은 영역 분할법[1,6]과 외재적 메시지 전달 기법이다.

영역 분할은 전체 해석 대상 영역을 다수의 소영역으로 분리하고 소영역을 각 프로세서에 배정하여 해를 구하는 것이다. Eulerian 접근법을 사용하는 유한체적(유한차분)법의 경우, 영역 분할이 작업배분이 용이하며 논리적으로 자연스럽다. 특히, 구조격자를 사용하는 경우 일부분에만 나타나는 국소항이 없다면 프로세서당 동일량의 격자점을 할당함으로써 바로 부하 균형을 이룰 수 있다는 것이 영역분할의 장점이다.

소영역은 해를 구해야 하는 해석 영역과 이를 둘러싼 경계로 구성된다. 경계는 다시 경계 조건이 적용되어야 하는 물리적 경계와 영역 분할에 의해 생긴 인접 소영역과의 가상 경계(이하 통신 경계)로 다시 구분된다. 통신 경계는 실제로는 인접한 소영역의 해석 영역과 중첩되며 통신에 의해 경계치를 갱신하여야 한다.

프로세서간 통신은 MPI (Message Passing Interface[10])와 SHMEM[11]을 이용하여 처리하였다. 메시지 전달 라이브러리의 표준인 MPI는 이식성을 고려하여 선택하였다. 직접 메모리 접근 방식의 SHMEM은 Cray 시스템 전용 통신 환경으로 이식성이 제한적이나 통신 속도가 우수하여 Cray T3E에서의 최적 성능을 위해 선택하였다.

3.2 병렬화 과정

병렬화는 다음의 세 단계로 이루어졌다.

(1) 프로세서간 통신의 설정: 데이터 의존성 분석을 통해 동기화(synchronization) 지점과 통신 내용과 통신 방법을 결정하였다. 코드 분석 결과, 필요한 통신은 ① 초기 입력 정보와 격자 및 영역 분할 등 전체가 알아야 할 정보를 공유하기 위한 집합적 통신, ② 해의 수렴 여부 판정 및 판정 결과의 공유를 위한 집합적 통신, ③ 통신 경계에서 계산된 계수와 해를 인접 소영역과 공유하기 위한 일:일 통신으로 정리되었다. 일부 프로세서의 공전 억제와 통신 오버헤드 감소를 위하여 동기화 지점과 통신 횟수는 최소화하였다.

(2) 원본 코드의 구조 조정 : 해석 영역의 분할 및 배분, 소영역 및 프로세서간의 topology 정보와 전체와 소영역에서의 계산 인덱스 정의, 병렬 프로세스 제어, 전체적 성격의 작업과 특정 프로세서에만 국한되는 작업의 분류 등 병렬 작업에 필요한 부분을 추가하여 코드를 재구성하였다.

(3) 병렬 코드의 완성 : 결정된 원칙과 지침, 문법에 따라 병렬 프로그램을 완성하였다. 이때 통신 처리 등 새롭게 추가되어야 하는 기능은 가능한 범위 내에서 모듈화하여 원본 코드에 대한 영향을 최소화하였다.

이러한 과정을 거쳐 MPI와 SHMEM 두 가지 통신 환경을 사용할 수 있는 병렬 코드를 작성하였다. 실제로 프로그램은 하나이며 정의문(define statement)을 사용하여 컴파일할 때 택일할 수 있도록 하였다. 사용할 PE의 수는 입력에 의해 조정 가능하며 하나의 PE를 사용하는 경우 원본 순차 모델과 동일한 모델이 된다.

3.3 병렬형 행렬 해법

행렬 해법은 병렬화에 있어서도 다양한 쟁점을 유발하는 핵심적인 부분이다. 원본 모델에서 채택한 TDMA 해법은 병렬 계산시 치명적인 프로세서 공전을 유발할 수 있는 순차형 알고리즘으로 효율적인 병렬화가 쉽지 않으나 일반적으로 많이 사용되는 위상 때문에 상당한 병렬화 노력이 이루어졌다 [3,5,6,12]. Table 2에 현재 사용 가능한 병렬형 TDMA 해법과 그 특성을 정리하였다.

MBMPA(Multi-Block and Multi-Partitioning Algorithm [3])는 프로세서 하나에 다수의 다중 블록을 할당하는 방식의 해법으로 순차형 TDMA와 동일한 기법이나 코드 전체에 대한 수정이 요

Table 2. Comparison of parallel tridiagonal matrix solvers.

행렬 해법	순차형 TDMA와의 상사성		병렬화의 요구사항		적용 가능한 해석 영역
	수학적 배경	연산 과정	필요 수정 범위	통신 빈도	
MDMPA	동일	동일	코드 전체	많음	제한적
SGEA	동일	상이	행렬 해법 전체	다소 많음	다소 제한적
LTDMA	상이	상이	행렬 해법의 일부	적음	제한 없음

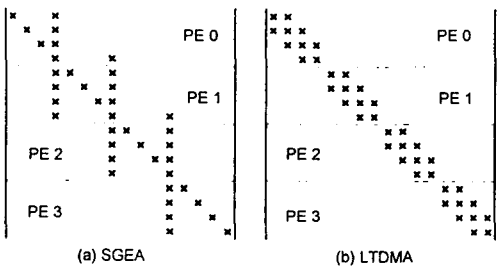

Fig. 1 Parallel matrix solution algorithms.

구되기 때문에 본 연구에서는 고려하지 않았다. SGEA(Substructuring Gaussian Elimination Algorithm [5,6])는 일련의 연산 과정을 통해 tridiagonal 행렬을 Fig. 1(a)과 같은 병렬 처리 가능 행렬로 변환하여 해석하는 방법이다. 수학적으로는 TDMA와 동일하나 해석 영역이 복잡한 경우 적용이 다소 어렵다.

LTDMA(Localized TDMA)은 Fig. 1(b)에 예시된 것처럼 인접 소영역간의 통신 경계를 마치 물리적 경계인 것처럼 처리하여 각 소영역의 해를 독립적으로 구하는 국소적 해법이다. 소영역간의 연결성은 통신 경계치의 갱신을 통해 외재적으로 고려된다. 수렴된 해를 얻기 위해서는 행렬 해법 내부의 소반복(subiteration)이 필요하다. 그러나 SIMPLER 알고리즘의 경우 본질적으로 압력과 속도의 내재적 결합을 위해 전체적 반복(global iteration)을 수행하기 때문에 이것으로 소반복을 대신할 수 있다. 이 해법은 수학적으로 TDMA와 다른 알고리즘이나 전 계산과정이 병렬적이며 요구되는 수정이 매우 적다. 이 연구에서는 SGEA와 LTDMA 해법을 고려하였다.

4. 결과 및 토론

두 가지 통신 환경과 두 가지 행렬 해법의 조합에 의한 다음의 4종의 병렬 모델을 작성하였다.

- MG 모델 : MPI와 SGEA 해법 사용
- SG 모델 : SHMEM과 SGEA 해법 사용
- ML 모델 : MPI와 LTDMA 해법 사용
- SL 모델 : SHMEM과 LTDMA 해법 사용.

계산은 PE의 수 N_{PE}를 1, 4, 9, 16, 25, 36, 48, 64, 81, 100으로 변화시키면서 수행하였다. 전체 해석 영역의 격자수는 362×362로 고정시켰다.

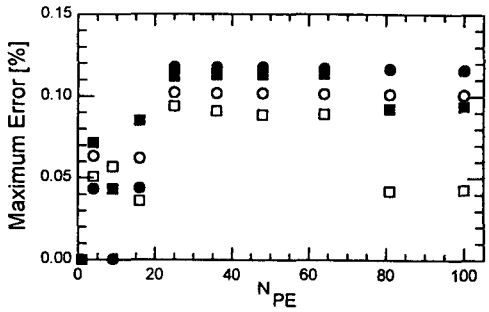

Fig. 2 Errors of the parallel models relative to the sequential model. □, E(v) for MG; ○, E(v) for ML; ●, E(θ) for MG; ■, E(θ) for ML.

N_{PE}=48(=6×8)을 제외하고는 수평과 수직 방향으로 같은 수로 분할하는 2차원 균등 영역 분할을 실시하였다. 이는 소영역에 배당된 격자수가 동일하게 하여 부하 균형을 유지하고 체적(소영역의 격자수)에 대한 면적(통신 경계의 격자수)의 비를 작게 하여 통신 부하를 최소화하기 위한 것이다.

병렬 모델의 신뢰성은 원본 순차 모델(N_{PE}=1)에 대한 비교를 통해 검증하였다. 병렬 모델로 얻은 해와 순차 모델로 얻은 해를 전체 해석 영역에서 비교하여 오차의 최대값

$$E(\phi) = \max[|\phi_p - \phi_s|] \qquad (2)$$

을 구하여 Fig. 2에 정리하였다. ϕ는 2.1절에서 언급한 수렴 조건을 만족한 해의 속도(v) 또는 온도(θ)이며 최대값이 $O(1)$이 되도록 무차원화한 것이다. 하첨자 p와 s는 병렬과 순차 계산을 의미한다. 모든 경우에 해석 영역에서 속도장과 온도장이 0.12% 오차범위 내에서 순차 모델과 일치하고 있다. 이 문제에서 중요한 결과물인 Nusselt 수를 차가운 벽에서 평가하였는데 모든 병렬 계산 결과가 순차 계산 결과(Nu=8.819)와 0.06% 오차 내에서 일치하였다.

위와 같은 검증 방법은 순차 모델의 신뢰성을 전제로 한 것이다. 순차 계산의 신뢰성과 정확도는 De Vahl Davis와 Jones[9]의 벤치마크 해(Nu=8.815)와의 비교를 통해 확인할 수 있다. 보다 상세한 검증은 이미 다른 문헌[8]에서 이루어졌으므로 여기서 피하기로 한다. .

남은 과제는 성능 평가인데 병렬 계산 성능은 여러 요소에 좌우되므로 각 요소의 영향을 독립적으로 고려하는 새로운 분석을 시도하기로 한다.

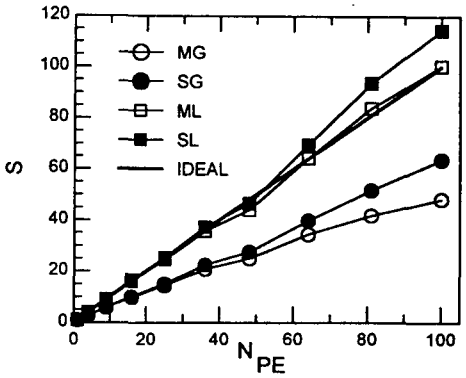

Fig. 3 Speedup of the present parallel models.

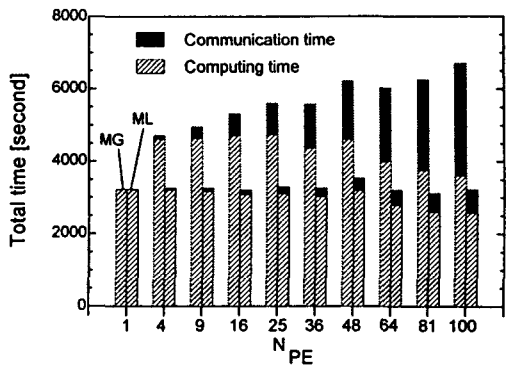

Fig. 4 Variation of composition of total time T.

4.1 코드의 병렬 계산 성능

먼저 문제에 따른 수렴성과 분리된 코드 성능 평가를 위하여 반복 1000회까지 계산을 수행하여 1회 반복 계산당 성능을 분석하기로 한다. Fig. 3은 병렬 확장성의 일반적인 지표로 사용되는 증속률(speed-up) S의 거동을 도시한 것이다.

$$S = \tau_s / \tau_p, \tag{3}$$

τ_s와 τ_p는 순차 계산과 병렬 계산의 소요 시간(elapsed time)이다.

고려된 4종의 병렬 모델에서 S는 N_{PE}에 거의 선형적으로 증가하고 있다. 이러한 선형성은 병렬 모델의 확장성과 성능 예측성을 표현하는 것이다. 또 하나의 평가 항목은 병렬 모델의 효율성인데 이상적 증속률($S=N_{PE}$)에 대한 실제 증속률의 비로 대변된다. LTDMA를 사용하는 ML과 SL 모델이 매우 우수한 계산 효율을 보이고 있다. 전체적으로 Fig. 3의 결과는 병렬화가 확장성 있게 적절히 이루어졌음을 반증하고 있다. 특기할 것은 ML과 SL 모델에서 이상적인 증속률을 상회하는 초선형적 증속 특성이 나타난다는 점이다. 이에 대해서는 4.3절에서 다시 논의하기로 하겠다.

4.2. 병렬 오버헤드 분석

Fig. 3의 결과 분석을 위하여 사용한 PE의 수를 고려한 총 작업시간을 다음과 같이 계산하였다.

$$T = \tau_p \times N_{PE}. \tag{4}$$

이 연구에서 모델의 특성상 부하 불균형과 지연 오버헤드는 고려 대상이 아니며, 해석영역이 통신 경계보다 상당히 커서 메모리 오버헤드 또한 미미하다 [1,3]. 그렇다면 T는 크게 순수 계산시간 T_{comp}와 통신시간 T_{comm}로 구분할 수 있다. 즉,

$$T = T_{comp} + T_{comm}. \tag{5}$$

Fig. 4는 N_{PE}에 따른 총 작업시간의 구성의 변화를 도시한 것이다. 통신과 연산 양쪽에서 쉽게 확인되는 오버헤드를 정량화하기 위하여, Eq. (5)로부터 통신 부하 지수 F_{comm}과 계산 부하 지수 F_{comp}를 다음과 같이 정의하였다.

$$T = \tau_s (F_{comp} + F_{comm}), \tag{6}$$

$$F_{comp} = T_{comp} / \tau_s. \tag{7}$$

$$F_{comm} = T_{comm} / \tau_s. \tag{8}$$

F_{comp}의 거동은 따로 도표로 작성하지 않더라도 Fig. 4에서 순차 계산(N_{PE}=1) 결과와의 상대 비교를 통해 확인할 수 있다. SGEA 해법을 사용하는 MG 모델의 경우, 순수 계산시간의 증가가 명확히 나타나는데 N_{PE}의 변화에는 민감하지 않았다. TDMA와 연산량이 거의 같은 LTDMA를 사용하는 ML 모델에서는 이러한 현상을 볼 수 없다. 따라서 MG 모델에서의 연산 시간의 증가는 SGEA 사용시 원행렬을 Fig. 1(a)의 형태로 변환하는데 요구되는 추가적인 연산에 의한 계산시간 증가, 즉, 산술 오버헤드로 설명할 수 있다 [6].

Fig. 5는 통신 오버헤드의 거동을 보여주고 있다. 모든 경우에 F_{comm}은 N_{PE}의 증가에 따라 선형적으로 증가하고 있다. 예상했던 것처럼 Cray 시스템에 최적화되어 있는 SHMEM이 MPI보다 약 2배 정도 빠른 통신 처리 속도를 보이고 있다

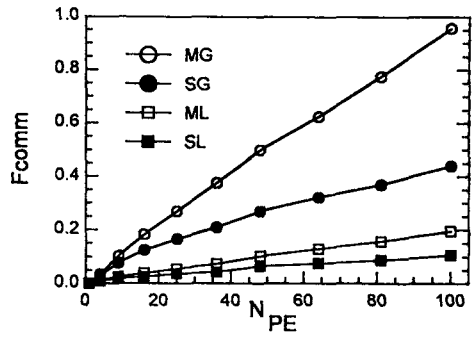

Fig. 5 Communication overhead factor vs. N_{PE}.

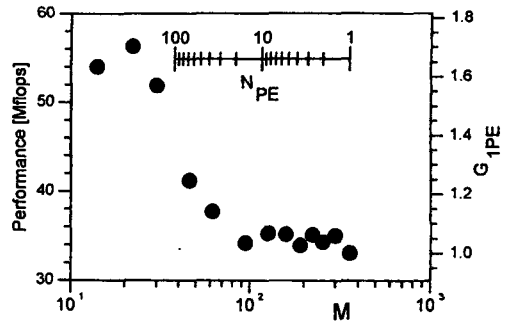

Fig. 6 Variation of single processor performance.

[11]. ML과 MG 모델은 단지 행렬 해법만이 다르므로, 통신 오버헤드의 상당 부분이 행렬 해법에서 발생함을 추론할 수 있고 SGEA가 많은 통신량을 요구하는 해법임을 재확인하였다. N_{PE}=100일 때 MG 모델의 통신 시간은 거의 순차 작업 시간과 같은 크기를 보이며 통신 오버헤드가 성능을 제약하는 일차적 요인이 되고 있다.

4.3 단일 프로세서의 실제 계산 성능

4.1절에서 언급한 바와 같이 Fig.3에 ML과 SL 모델의 초선형적 증속 특성이 나타나고 있는데 이러한 현상은 많은 양의 통신 오버헤드가 수반되는 N_{PE}가 큰 경우에 주로 발생하였다. 설명의 실마리는 Fig. 4에서 찾을 수 있다. LTDMA 해법을 사용하는 모델의 총연산량은 N_{PE}에 관계없이 순차 모델과 거의 같음에도 불구하고, N_{PE}가 증가함에 따라 순수 계산시간이 감소하고 있다. 이는 N_{PE}가 증가함에 따라 프로세서의 연산 성능이 좋아짐을 암시하는 것이다.

실제로 단일 프로세서의 연산 성능이 경우에 따라 달라질 수 있음을 [13] 고려하면 계산 부하지수 F_{comp}는 다음과 같이 재정리할 수 있다.

$$F_{comp} = F_{arith} / G_{1PE}. \qquad (9)$$

이때 F_{arith}는 진정한 의미의 산술 오버헤드를 계량하는 지수로 순차 계산시 연산량에 대한 병렬 계산시 총 연산량으로 정의된다. 경우에 따른 프로세서의 성능 변화를 대표하는 G_{1PE}는

$$G_{1PE} = P_p / P_s, \qquad (10)$$

로 정의되며 여기서 P_s는 순차 계산의 성능이며 P_p는 병렬 계산시 프로세서당 계산 성능이다.

G_{1PE}는 다음과 같은 방법으로 구하였다. 전체 격자수를 362×362로 고정시키고 경계를 제외한 해석 영역(360×360 격자점)을 2차원적으로 균등분할하여 PE 하나에 할당되는 실제 해석 영역의 격자수를 $M \times M$라고 하면 M은

$$M = 360 / N_{PE}^{1/2}. \qquad (11)$$

따라서, P_p는 동일한 격자수의 문제에 대한 순차 계산을 통해 간접적으로 평가할 수 있다. Fig. 6은 2장에서 수행한 순차 계산의 결과를 이용하여 N_{PE}와 G_{1PE}의 관계를 도시한 것이다. M이 64 ($N_{PE} \approx 36$) 이상이면, 단일 프로세서 성능은 거의 일정하다. M이 64보다 작아지면 성능이 증가하여 M=22 ($N_{PE} \approx 225$) 근처에서 최고 성능이 나타나고 있다. N_{PE}를 1에서 100까지 변화시키면서 계산을 수행할 경우, N_{PE}가 36을 넘어서면 프로세서당 연산 성능이 단조 증가하게 되는데 이것이 Fig. 4에 나타난 순수 계산시간 감소와 초선형적 증속 특성을 설명하고 있다.

이러한 결과는 시험 대상 기종의 하드웨어적 특성에 기인한 것이다. 여기서 원본 순차 모델은 원래 벡터형 컴퓨터에 최적화된 것이라는 점과 병렬화 과정에서 Cray T3E의 특성에 맞는 어떠한 최적화도 고려하지 않았다는 점을 언급할 필요가 있다. Cray T3E 시스템의 RISC 프로세서는 8 Kbytes의 데이터 캐쉬(cache)와 명령 캐쉬, 96 Kbytes의 2차 캐쉬, 그리고 정수 연산 장치와 소수점 연산 장치(덧셈 및 곱셈)를 가진다. 단일 프로세서의 연산 성능은 캐쉬와 연산 장치의 효율

적인 사용 여부에 크게 의존한다 [13]. Fig. 6의 결과는, 특별한 최적화 조치가 없는 경우 코드내의 do loop의 평균 길이인 M이 22 정도일 때 캐쉬의 최적 사용이 이루어짐을 보여주는 것이다.

실제로 RISC 프로세서의 경우 이론 성능과 실제 성능 사이에 상당한 거리가 존재하며 역설적으로 그만큼 성능 개선의 여지가 있다. 적절한 최적화를 수행하면 최소한 Fig. 6에 나타난 최고 성능 정도는 얻을 수 있을 것이다 [13]. 이상의 논의는 고성능 병렬 모델 개발에 있어 병렬 컴퓨터의 프로세서 특성을 고려한 최적화라는 후속 작업의 필요성을 제기하고 있다.

4.4 수렴성을 고려한 병렬 계산 성능

지금까지는 일반적인 병렬 컴퓨팅의 관점에서 병렬 코드의 성능을 분석하였다. 그러나 수렴성을 배제한 1회 반복계산당 성능 분석은 전산유체역학 모델의 성능을 논의하는데 한계가 있다. 병렬화 과정에서 알고리즘이나 연산 방법의 변경으로 병렬 모델은 원본 순차 모델과는 다른 수렴 과정을 보일 수 있기 때문이다 [6]. 이 논문에서는 병렬화에 따른 수렴성의 변화를 정량화하기 위하여 다음과 같은 수렴인자를 도입하였다.

$$H = ITER_p / ITER_s, \qquad (12)$$

$ITER_p$와 $ITER_s$는 병렬 모델과 순차 모델로 수렴된 해를 구하는데 소요된 총 반복횟수이다.

Fig. 7은 H의 거동을 도시한 것이다. 순차 모델과 수학적으로 동일한 MG 모델은, 연산 과정의 차이에 의한 round-off 오차의 영향으로 약간의 차이는 있으나, 순차모델과 거의 비슷한 수렴성을 보이고 있다. 수학적으로 전혀 다른 LTDMA 해법을 사용하는 ML 모델은 수렴해를 얻는데 순차 모델에 비해 약 10% 가량 추가적인 반복 계산이 요구되는 것으로 나타나고 있다.

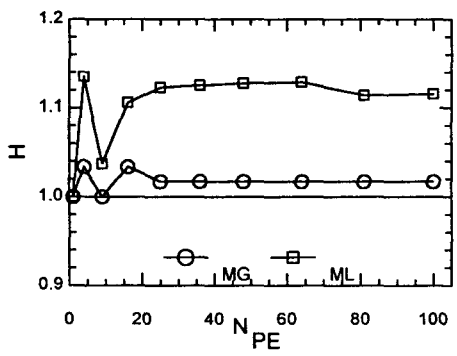

Fig. 7 Variation of total number of iteration.

논의를 정리하면 병렬 모델의 효율성은 다음의 식으로 평가할 수 있다.

$$R = T_p / \tau = (F_{comp} + F_{arith} / G_{1PE}) H. \qquad (13)$$

모든 요소가 복합된 Eq. (3)의 증속률보다 판정식 Eq. (13)이 성능을 결정하는 여러 요인의 영향을 분리하여 고려한 보다 체계적인 성능 분석법이라 할 수 있다. 여기서 R은 효율 지수이며 R=1은 프로세서의 연산 성능 변화를 무시한 경우의 이상적인 병렬화를 의마한다. R이 1보다 커지는 것은 총 작업시간의 증가, 즉 병렬 계산 성능의 감소를 나타내는 것이다.

고려된 4개의 모델에 대해 R의 거동은 Fig. 8에 도시하였다. 수렴성에서의 10% 정도의 손실에도 불구하고 (Fig. 7) 단위 반복횟수당 성능이 우수한 LTDMA를 사용하는 모델이 우수한 성능을 보이고 있다. SGEA의 경우, 통신과 산술 오버헤드 모두 성능 제약 요인으로 작용하고 있다.

이상의 논의에서 LTDMA와 같은 소영역 단위 국소적 병렬 행렬 해법이 여러 면에서 효율적임을 확인하였다. 그러나 수렴성은 문제에 따라 다를 수 있기 때문에 이러한 주장을 일반화하기 위해서는 비정상성이나 다른 유동 조건을 고려한 수치 실험과 객관적 검증이 필요하다. 그러나 몇 가지 물리적 근거로부터 LTDMA의 적용성과 유

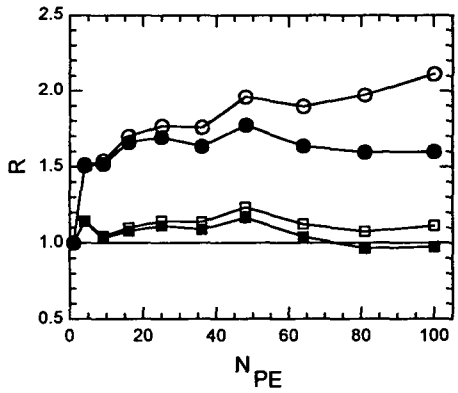

Fig. 8 Computational efficiency of the present parallel models. ○, MG; ●, SG; □, ML, ■, SL.

용성을 타진해 볼 수 있다.

LTDMA는 소영역을 독립적인 것으로 가정하여 해를 구하고 영역간의 연결성은 통신 경계치의 갱신을 통해 처리한다. 이 방법은 소영역간의 경계에서 특정 파장의 교란의 전파를 차단하여, 문제에 따라서는 중요한 물리 현상의 손실, 왜곡, 수렴성의 저하를 초래할 수 있다. 예제인 측벽 가열형 용기내의 자연대류에서 가장 중요한 특성길이는 수직벽에 형성되는 온도 경계층의 두께로 $\delta \sim O(Ra^{1/4}) \sim 1/30$ 정도의 크기를 가진다 [8]. 이 연구에서 소영역의 크기가 가장 작은 N_{PE}=100의 경우에 소영역의 길이는 10^{-1}이다. 따라서 소영역이 경계층을 충분히 수용하기 때문에 영역분할에 의해 영역 분할에 따른 수렴성에 큰 영향을 주지 않았다. 따라서 문제의 물리적 특성에 대한 정확한 이해와 이를 기반으로 한 영역분할을 수행한다면, 특히 반복 계산에 의해 정상상태만을 구하는 경우에는, LTDMA와 같은 국소적 병렬 행렬 해법의 적용이 효율적일 것으로 판단된다.

5. 결론

영역 분할과 메시지 전달을 이용하여 병렬형 SIMPLER 유한체적 모델을 작성하였다. 초병렬 컴퓨터인 Cray T3E에서 시험을 수행하여, 병렬 모델의 신뢰성과 확장성 분석을 통해 병렬화의 적절성을 검증하였다. 시험 결과는 행렬 해법이 병렬 모델의 성능을 좌우하는 병렬화의 핵심 요소임을 보여주었다.

이 연구에서는 병렬 오버헤드 분석과 함께 단일 프로세서의 연산 성능과 문제에 따른 수렴성을 독립적으로 고려한 새로운 병렬 성능 평가를 시도하였다. 종합적인 평가 결과 LTDMA와 같은 국소적 행렬 해법을 사용하는 모델의 유용성을 확인하였다.

후기

코드 병렬화와 결과 분석에 조언을 주신 세종대학교의 김 세용 교수, 한국과학기술원의 박사과정 김 성호씨, 공주대학교의 전용두 교수께 감사의 말씀을 전한다. 이 연구는 정보통신부가 지원하는 슈퍼컴퓨터 운영사업과 슈퍼컴퓨터 이용효율화사업의 일환으로 이루어졌으며 이에 감사드린다.

참고문헌

[1] Chan, T., "Domain Decomposition Algorithms and Computational Fluid Dynamics," *Int. J. Supercomputing Applications* **2** (1988), p.72.

[2] Gropp, W. and Smith, E., "Computational Fluid Dynamics on Parallel Processors," *Computers & Fluids* **18** (1990), p.289.

[3] Naik, N. H. et al., "Parallelization of a Class of Implicit Finite Difference Schemes in Computational Fluid Dynamics," *Int. J. High Speed Computing* **5** (1993), p.1.

[4] Stagg, A. K. et al., "Parallel, Scalable Parabolized Navier-Stokes Solver for Large-Scale Simulators," *AIAA J.* **33** (1995), p.102.

[5] Johnsson, L. et al., "Alternating Direction Methods on Multiprocessors," *SIAM J. Sci. Stat. Comput.* **8** (1987), p.686.

[6] Zhu, J., *Solving Partial Differential Equations on Parallel Computers*, World Scientific. New Jersey (1994).

[7] Patankar, S. V., *Numerical Heat Transfer and Fluid Flow*, McGraw-Hill, New York (1980).

[8] Kwak, H. S. and Hyun, J. M., "Natural Convection in an Enclosure Having a Vertical Sidewall with Time-Varying Temperature," *J. Fluid Mech.* **329** (1996), p.65.

[9] De Vahl Davis, G. and Jones, I. P., "Natural Convection in a Square Cavity - A Comparison Exercise," *Int. J. Num. Methods Fluids* **3** (1983), p.227.

[10] Snir, M. et al., *MPI: The Complete Reference*, MIT Press, Cambridge (1996).

[11] SHMEM, *CrayT3E Fortran Optimization Guide*, SG-2518 2.0.1, Cray Research Inc. (1996).

[12] Eltgroth, P. G., "Two Portable Parallel Tridiagonal Solvers," UCRL-JC-118017, Lowrence Livermore Nat. Lab. (1994).

[13] E. Anderson et al., *The Benchmarker's Guide to Single-Processor Optimization*, Cray Research (1997).