영역분할법에 의한 SIMPLER 기법의 병렬화

Parallel Implementation of SIMPLER by Using Domain Decomposition Technique

곽호상[1)]
Ho Sang Kwak

A parallel implementation is made of a two-dimensional finite volume model based on the SIMPLER. The solution domain is decomposed into several subdomains and the solution at each subdomain is acquired by parallel use of multiple processors. Communications between processors are accomplished by using the standard MPI and the Cray-specific SHMEM. The parallelization method for the overall solution procedure to the Navier-Stokes equations is described in detail. The parallel implementation is validated on the Cray T3E system for a benchmark problem of natural convection in a sidewall-heated cavity. The parallel performance is assessed and the issues encountered in achieving a high-performance parallel model are elaborated.

1. 서론

초병렬 컴퓨터(Massively Parallel Processors, MPP)로 대변되는 병렬 환경의 출현으로 전산유체역학은 새로운 전기를 맞이하고 있다. MPP는 CPU와 메모리로 구성된 PE(Processing Element) 수백~수천 개를 고속의 상호 접속망(Interconnection Network)으로 병렬 연결한 것으로 PE 수를 증가시킴으로써 컴퓨터의 성능 향상과 기억용량 확대를 도모할 수 있는 확장성(scaleability)이 핵심적인 장점이다. MPP의 보급으로 실용적 열유체 해석 도구로서의 전산유체역학의 위치가 더욱 다져짐은 물론 난류, 이상 유동, 연소 등 막대한 계산시간과 기억용량이 요구되는 초대형 과제에 도전할 수 있는 기반이 조성되고 있다.

그러나 현존하는 계산유체역학 수치 알고리즘과 코드는 대부분 단일 프로세서 이용을 전제로 개발된 것이다. 따라서 복수의 프로세서를 사용할 수 있는 병렬형 계산 모델의 개발은 초고성능 병렬 전산환경을 활용하기 위한 전제 조건이다. 그 동안 전산유체 분야에서도 병렬형 수치 모델 개발에 위한 적지 않은 노력이 경주되었다 [1-5]. 병렬 계산을 목적으로 한 새로운 알고리즘의 개발과 함께 이미 기득권적인 지위를 가지며 사용되고 있는 순차형(sequential) 계산 코드들의 병렬화가 광범위하게 진행되고 있다. 문제는 병렬 성능이 컴퓨터의 하드웨어적 특성과 병렬화 방법에 매우 민감하며 순차형 코드의 병렬화는 원본 코드의 구조 및 사용 알고리즘에 크게 의존한다는 점이다. 이로 인해 병렬 성능과 이식성이 뛰어난 최적의 병렬 코드의 구현 과정에서 여러 가지 쟁점이 제기되고 있으며 이의 해결을 위해서는 순차형 코드의 병렬화에 관한 다양한 관점에서의 수치 실험이 요구되고 있다.

이 연구에서는 이러한 문제의식 아래 유한체적법 또는 유한차분법에 기초한 계산유체 모델의 병렬화에 도움이 될 만한 지침을 마련하고자 열유체 해석에 널리 사용되고 있는 SIMPLER[6] 기법의 유한체적 모델을 분산 메모리형 컴퓨터에서 작동 가능하도록 병렬화하였다. 원본 순차형 코드의 구조 및 성능 분석에 기초하여 수립된 병렬화 전략과 구체적인 방법론이 기술될 것이며 MPP인 Cray T3E를 이용한 계산과 성능 분석을 통해 병렬화된 모델의 신뢰성과 병렬 컴퓨팅의 유용성이 예증될 것이다. 행렬 해법 등 병렬화 과정에서 제기되는 몇 가지 쟁점이 병렬 성능의 관점에서 조명될 것이다.

2. 원형 모델과 예제

병렬화 대상인 원본 코드는 비정상 자연대류 문제에 성공적으로 적용되었던 것으로 Kwak과

1) 시스템공학연구소 슈퍼컴퓨터센터 (대전광역시 유성구 어은동 1, Tel: 042-869-0546, hskwak@seri.re.kr)

Hyun[7]에 의한 작성된 엇갈림 격자를 사용하는 SIMPLER[6] 방식의 유한체적 모델이다. 지배방정식은 Boussinesq 유체의 2차원 비정상 Navier-Stokes 방정식이다. 시간 적분은 반복(iteration)에 기초한 Euler 음함수법으로 진행되며 선형항과 비선형 대류항은 각각 중심차분법과 QUICK 기법에 의해 계산된다. 이 모델은 시간에 대해서는 1차, 공간에 대해서는 2차의 정확도를 가진다.

전체 계산과정은 다음의 세 기능집단으로 구분된다. (I) 문제(무차원수, 초기 및 경계조건)의 정의 및 격자 구성, 수렴 여부 판정, 입출력 및 계산 과정 제어, (II) 속도, 압력, 압력보정 및 스칼라량 계산을 위한 행렬방정식의 계수 및 생성항 계산, (III) 행렬방정식의 해의 계산.

이중 III의 부분이 일반적으로 계산시간이 많이 소요되는 핵심 부분이다. 2차원의 경우 모든 격자점은 상하 및 좌우에 위치한 격자점과의 연결성을 가지므로 지배방정식을 격자화하면 5개의 항을 갖는 거대한 행렬식이 유도된다. 원본에서는 한 방향에 대한 의존성을 약화시켜 이 행렬식을 두 개의 블록 tridiagonal 행렬식으로 변환시키고 벡터형 TDMA(TriDiagonal Matrix Algorithm)를 이용 그 해를 구하는 방법을 사용하였다 [8].

검증과 성능평가를 위한 예제로 전산유체 코드의 벤치마크 시험문제로 자주 인용되는 고온 및 저온 측벽과 단열 수평벽을 가지는 사각 공간내의 자연대류[9]를 선택하였다. 모든 계산에는 균일 격자를 이용하였고 Rayleigh 수와 Prandtl 수는 각각 10^6, 0.71로 고정시켰다. 일정 온도의 초기 조건으로부터 측벽의 온도 경계조건을 변화시켜 구동되는 유동장과 온도장을 계산하였다. 원본 코드는 비정상 문제도 해석할 수 있도록 작성되었으나 여기서는 정상상태 해법을 사용하였다. 속도와 온도의 상대 변화량의 최대값과 연속방정식의 만족도가 각각 10^{-4}와 10^{-8} 이하로 떨어지면 수렴된 정상해를 얻었다고 판정하였다.

성능 비교의 기준을 마련하기 위해 원본 순차형 코드로, Cray C90[2]와 Cray T3E[3]의 PE 한 개를 사용하여 격자의 수를 달리하며 계산을 실시하여 결과를 Table 1에 정리하였다. 벡터 컴퓨터인 Cray C90에서는 계산의 평균 벡터 길이가 기준 길이인 128에 근접할수록 좋은 성능을 보이고 있다. Cray T3E의 경우는 격자수가 32×32인 경우 가장 좋은 성능이 보이며 문제의 크기가 커질수록 감소하여 격자수가 96×96 이상이면 거의 동일한 성능이 나타나고 있다. 이러한 결과는 Cray T3E의 하드웨어적 특성에 기인한 것이다. RISC 칩인 Cray T3E의 CPU는 캐쉬(Cache)를 채용하고 있어 캐쉬의 효율적 사용 여부가 코드의 성능을 좌우하는 핵심적인 인자가 된다. 코드에서 가장 안쪽에 위치한 do loop에서의 메모리 사용이 4 Kbytes 캐쉬의 크기와 잘 맞는 경우 최고의 성능이 나타나는데 원본 코드의 경우 x-방향의 격자수가 적절히 작은 경우가 여기에 해당된다.

grid size	Cray C90			Cray T3E	
	time [sec]	performance [Mflops]	average vector length	time [sec]	performance [Mflops]
32×32	3.876	198.9	29.63	14.11	54.64
64×64	10.74	305.4	61.25	87.27	37.58
96×96	21.49	350.2	89.68	209.7	35.89
128×128	35.64	379.0	120.0	374.5	36.07
160×160	61.02	348.0	77.10	587.3	36.17
192×192	82.70	371.2	92.63	852.2	36.02
224×224	109.9	381.2	106.5	1207.	34.71
256×256	140.1	391.5	121.7	1534.	35.76
362×362	284.8.	386.7	115.3	3229.	34.11

Table 1 Performance of the original sequential model.

3. 병렬화 방법

순차형 모델의 병렬화는 통상 추가적인 비용(overhead)를 요구하는데 알고리즘의 병렬화 과정에서 발생하는 추가적 소수점 연산에 의한 arithmetic overhead와 병렬화가 유발하는 프로세서간 정보교환에 의한 communication overhead, 그리고 각 프로세서에 배당된 작업량의 차이 즉 부하의

2) 공유 메모리 방식의 병렬벡터 컴퓨터로 16개의 CPU를 가지며 최대 성능은 16 Gflops(1 Gflops/CPU)이다.
3) 분산 메모리 방식의 초병렬 컴퓨터로 128개의 PE를 가지고 있다. 각 PE는 최고 성능 0.9 Gflops의 DEC alpha RISC chip CPU와 128 Mbytes 의 메모리로 구성된다. 전체 PE는 3차원 torus 구조로 연결되어 있고 PE간 전방향 대역폭은 500 Mbps이다. 이론 최고 성능은 115 Gflops이다.

불균형 (load unbalance)에 의한 프로세서의 공전 시간으로 크게 나눌 수 있다. 이러한 간접비용을 줄이는 것이 고성능 병렬 모델을 작성에 있어서 핵심적인 관건이다.

이 연구에서는 병렬화의 기본 틀로 전체 해석영역을 다수의 세부영역(subdomain)으로 나누고 다수의 프로세서가 세부영역을 하나씩 분담하여 해를 구하는 영역분할법을 선택하였다. 구조 격자를 이용하는 유한체적법 또는 유한차분법의 경우, 영역분할이 논리적으로 용이할 뿐만 아니라 유체의 물성치가 균일하고 국소적인 생성항이 없다면 격자점당 계산량이 거의 동일하기 때문에 각 프로세서에 동일한 양의 격자를 할당함으로써 부하의 균형이 만족시킬 수 있는 것이 이 방법의 주된 장점이다 [1].

영역분할에 있어 각 프로세서는 자신이 해석을 담당하는 격자점과 함께 이를 둘러싼 경계면의 정보를 갖도록 설계하였다. 이 경우 경계는 해석 영역에서의 실제 위치에 따라 문제의 경계조건이 적용되는 물리적 경계(physical boundary)와 인접 영역과의 정보 교환이 요구되는 통신 경계(communication boundary)로 나누어 진다. 본 병렬화 작업에서는 코드 변환의 편이와 원본 코드와의 상관성을 고려하여 통신 경계의 경우 해석 영역의 외부로 2개의 격자점을 추가로 정의하고 통신에 의해 인접 영역과 정보를 공유하는 영역중복(domain overlap) 기법을 사용하였다.

프로세서간의 정보 교환은 MPI(Message Passing Interface)와 SHMEM을 이용한 메세지 패싱 방법에 의해 처리하였다. MPI[9]는 메시지 패싱 라이브러리의 표준으로 발표된 것으로 다른 병렬 환경으로의 이식성을 고려하여 선택하였다. SHMEM[10]은 STREAM 기능에 기초한 메모리 접근 방식의 통신 라이브러리로 Cray 시스템에서만 활용 가능하지만 하드웨어의 특성에 맞게 설계되어 통신 속도가 빠르기 때문에 Cray T3E에서의 최적화를 고려하여 선택하였다.

병렬화 작업은 다음의 세 단계로 이루어졌다.

(1) 프로세서간 통신의 설정 : 데이터 의존성 분석을 통해 동기화(synchronization) 지점과 교환해야 할 정보의 내용을 정의하고 적절한 커뮤니케이션 방법을 결정하였다. 코드 분석 결과, 필요한 통신은 ① 계산 조건과 방법 등의 초기 입력 정보와 격자 및 영역분할에 대한 정보를 공유하기 위한 통신, ② 해의 수렴 여부 판정과 이의 전제적 공유를 위한 통신, 그리고 ③ 통신 경계면에서 계산된 행렬 계수를 공유하고 각 세부영역에서 해를 구한 후 통신 경계의 값을 update하기 위한 인접 영역과의 통신으로 정리되었다. 요구되는 통신의 내용은 ③의 경우에는 일:일 통신(point-to-point communication)이고 나머지 두 경우에는 집합적 통신(collective communication)이다. 이 과정에서 핵심적인 고려사항은 동기화 지점의 수를 최소화하는 것이었는데 이는 일부 프로세서 공전을 최소화하고 소수점 연산보다 많은 시간이 요구되는 통신의 횟수를 가능한 한 최소로 줄이기 위한 것이다.

(2) 원본 코드의 구조 조정 : 영역분할에 의한 병렬 작업이 실질적으로 이루어 지기 위해 필요한 부분을 추가하였다. 사용할 PE 수에 따른 영역의 분할, 각 프로세서에게 자기 영역의 특성 및 인접 영역에 대한 정보 제공, 전체 영역과 세부영역에서의 계산 인덱스 생산 및 격자 관련 정보의 배분이 주요 내용이다. 이와 함께, 경계조건을 적용과 같이 각 세부 영역의 특성에 따라 요구되는 업무를 선택적으로 수행하도록 조정하는 작업과 병렬 계산의 최적화를 이루기 위한 코드 수정도 여기에 포함된다.

(3) 병렬 코드의 작성 : (1)-(2) 단계에서 정해진 원칙과 방법에 따라 메시지 패싱 라이브러리의 문법에 맞추어 병렬 프로그램을 완성한다.

이러한 과정을 거쳐 MPI와 SHMEM 두 가지 환경을 이용가능한 병렬 코드를 개발하였다. 실제로 프로그램은 한 개로 작성되었는데 정의문(define statement)을 사용하여 MPI와 SHMEM 중 한 가지 통신방법을 선택할 수 있도록 하였다. PE 수(NPE)는 입력에 의해 임의의 개수로 조정 가능하도록 하였으며 NPE=1이면 순차형 원본 순차형 모델이 되도록 하였다.

일반적으로 유한체적 모델의 병렬화에 있어 핵심적 쟁점이 되는 것은 행렬 해법이다 [3]. 원본 코드에서는 사용한 TDMA는 일반적으로 많이 사용되고 있는 해법으로 이미 여러 형태의 병렬화 방법이 논의되고 있다 [3,8,12]. 그 일례가 MDMPM(Multi-Domain and Multi-Partitioning Method)로 원본 TDMA 기법과 수학적으로 계산학적으로 완전히 일치하는 병렬해법이지만 원본 코드에 대한 대대적인 수정이 요구되는 것이 단점이다. 다른 하나는 SGEA(Substructuring

Gaussian Elimination Algorithm)을 통한 행렬 변환을 통해 TDMA를 일련의 병렬화된 계산과정으로 치환하는 방법인데 수학적으로는 TDMA와 동일하나 연산 방법이 달라 계산학적인 관점에서는 다른 해법이다. 그러나 원본 TDMA에 충실한 이 두 가지 해법은 반드시 통신을 수반하며 정사각형과 규칙적 영역분할이 용이하지 않은 복잡한 영역을 가지는 경우 적용하기 어렵다. 이러한 문제를 극복할 수 있는 가능한 선택의 하나가 LTDMA(Localized TDMA)이다. 이 방법은 subdomain이 서로 독립적인 것처럼 가정하여 통신경계의 값을 경계조건으로 각 영역에서 완결적으로 TDMA를 적용하여 해를 구하고 난 후 인접 영역끼리 정보를 교환하여 경계치를 update하는 과정을 반복하여 수렴된 해를 얻는 것이다. 병렬화가 요구하는 추가적인 통신 부담이 없고 총 연산수가 약간 감소하며 원본 코드를 거의 그대로 수정없이 이용할 수 있다는 점이 장점이나 수학적으로 TDMA와는 상이한 해법이다. 이 해법은 NPE=1 이면 TDMA와 동일하나 PE수와 격자수가 동일한 극단적인 경우는 Point-Jacobi법이 된다. 본 연구에서는 이중 SGEA에 기초한 해법(Solver I)과 LTDMA(Solver II)의 두 가지 핼렬 해법을 사용하였다.

4. 검증 및 성능평가

병렬 코드의 검증 및 성능을 분석하기 위하여 2절에서 정의한 문제에 대하여 전체 격자수를 362×362로 고정하고 NPE를 1, 4, 9, 16, 25, 36, 48, 64, 81, 100로 변화시키면서 계산을 수행하였다. 영역은 NPE=48(=8x6)인 경우를 제외하고는 모두 x와 y 방향으로 같은 수의 분할(n×n)을 하였다.

병렬 모델의 신뢰성을 검증하기 위하여 병렬 계산된 해와 원형 수차 모델(NPE=1)로 계산한 결과를 비교하였다. 수련된 해의 온도장과 속도장에서의 최대 오차의 값을 Fig. 1에 도시하였다. 모든 경우에 최대 오차 0.12% 미만의 일치를 보이고 있다.

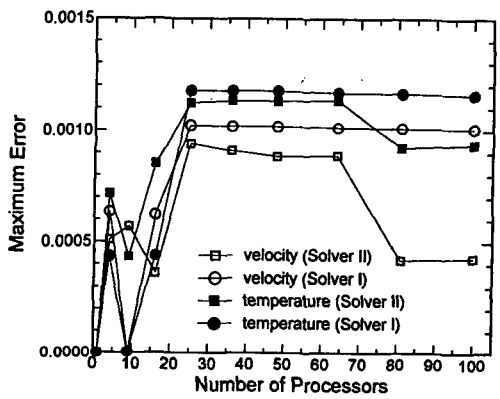

Fig. 1 Difference between the solutions acquired by sequential and parallel solutions.

먼저 코드의 수렴성이 아닌 계산 성능 즉 iteration당 요구되는 시간을 평가하기 위하여 1000 iteration step까지 계산한 결과를 분석하기로 한다. 일반적으로 병렬 성능을 표현하는 지표로 사용되는 증속률(speed-up), $S=T_s/T_p$ (T_s : 순차형 모델 이용 시의 계산 시간; T_p : 복수의 PE를 사용한 병렬 계산 시간)을 Fig. 2에 도시하였다. 모든 경우에 PE 수의 증가에 따라 S가 선형적으로 증가하는 것을 확인할 수 있다. 이러한 선형성은 병렬 모델의 확장성과 병렬 성능 예측성을 보장하기 위한 중요한 평가 항목이다. 또 하나의 관심은 실제 병렬계산 성능이 이상적인 증속률에 얼마나 근접하는가 하는 점으로 병렬화가 효율성을 가름하는 척도가 된다. Solver I의 경우는 이론 최고 성능에 못미치는 증속율을 보여주고 있으나 Solver II의 경우는 이론 최고 성능치를 넘는 초선형 현상이 나타나고 있는데 이에 대해서는 추후 다시 논의하기로 한다.

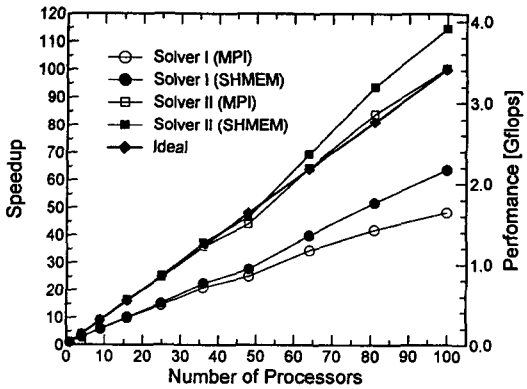

Fig. 2 Results of parallel performance tests.

Fig. 2에 나타난 결과에 대한 해석을 위해 병렬 성능 결정의 중요 인자인 communication overhead를 분석하였다. Fig. 3는 전체 계산시간 중 통신이 차지하는 비중을 백분율로 표시하는 것이다. 전체적으로 통신 비용은 NPE의 증가에 따라 선형적으로 증가하고 있으며 Solver II의 경우에 비하여 Solver I의 경우가 2배 정도의 통신비용을 요두된다. Solver I과 Solver II는 단지 행렬 해법에서만 차이가 나므로 이러한 Solver I에 나타난 추가적인 통신 부담은 주로 TDMA의 병렬화에 의한 발생한 것이다. Fig. 3가 보여주는 또 하나의 결과는 어떠한 message passing 기법을 사용하는가에 따라 성능의 편차가 크게 발생할 수 있다는 점이다. SHMEM의 경우가 MPI의 경우보다 통신속도가 두 배정도 빠른 것으로 나타나고 있다. 이러한 결과는 Cray T3E 시스템을 사용하는 경우, 하드웨어의 특성에 맞게 설계된 SHMEM을 사용하는 것이 훨씬 효율적임을 지적하는 것이다.

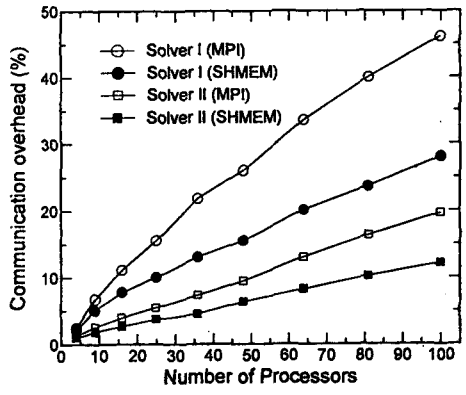

Fig. 3 Communication overhead vs. number of processors.

Figs. 2와 3에서 확인할 수 있는 것은 Solver I와 Solver II의 성능 차이가 communication overhead 만으로 설명되지 않는다는 점이다. 그 원인을 분명히 하기 위해 전체 계산 시간과 행렬 해법에 소요되는 시간중 통신 부분을 제외한 순수 계산 시간의 총량 (평균 CPU 시간 × PE 수)을 Fig. 4에 도시하였다. 이 그림에서 전체 계산에서 행렬 해법이 차지하는 비중을 쉽게 확인할 수 있다. Solver I이 성능이 이론 성능에 못미치는 커다란 원인이 SGEA법이 요구하는 행렬변환이 기존의 TDMA법에 비해 추가적인 연산을 요구함에 따라 발생한 arithmetic overhead임을 파악할 수 있다. Solver II의 경우 PE 수가 증가함에 따라 오히여 총계산 시간이 감소하는 현상을 보이고 있는데 이는 2장에서 언급한 Cache의 최적이용과 관계가 있다. 본 연구에서는 해석영역의 크기를 고정하고 PE 수를 변화시켰으므로 PE수의 증가는 각 프로세서에 할당되는 문제의 크기의 감소를 의미하며 Table 1에서 최고성능을 보여주고 있는 32x32의 크기에 근접하게 된다. 이것이 100개의 PE를 사용하는 경우 communication overhead를 감수하고도 15% 이상 이론치를 능가하는 성능을 보이는 현상을 설명하고 있다.

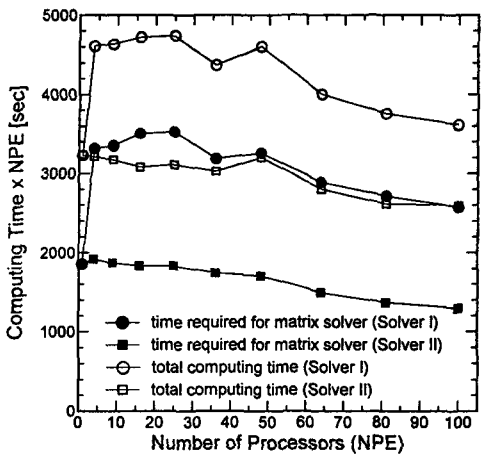

Fig. 4 Total computing time except for communication vs. number of processors.

지금까지 Solver II의 경우가 Solver I의 경우 보다 iteration 당의 계산 성능이 우수한 것을 확인하였다. 그러나 계산유체역학적 관점에서 보면 해의 수렴성을 고려하지 않은 코드의 계산 성능은 큰 의미가 없다. 사실상 Solver II는 순차형 원본 코드의 TDMA와는 전혀 다른 행렬 해법을 적용한 것이기 때문에 수렴성에 대한 시험이 필요하다. 이러한 요구에 답하기 위해 그래서 실제 수렴된 해를 얻을 때까지 계산

을 수행하여 요구된 총 Iteration 수를 Table 2에 정리하였다. Solver II의 경우 순차 해법을 사용하는 경우보다 약간 iteration 수가 증가하고 있으나 전체적인 수렴성은 비슷한 것으로 나타났다. 이러한 결과는 물리적으로 설명가능한 것으로 장파장의 변화 특성이 지배적인 경우를 제외하고는 다시 말해 subdomain의 크기가 중요한 물리적 특성길이를 포함할 수 있는 경우에는 이러한 Solver II 형의 국소적 해법과 TDMA와 같은 전체적 해법이 수렴성에 있어 큰 차이를 보이지 않기 때문이다. 특히 본 연구의 대상인 정상상태 해법의 경우에는 그 차이가 더욱 축소될 것이다. Solver II가 Solver I에 비해 거의 두 배에 가까운 성능을 보여주는 것을 감안하면 전체적으로 Solver II의 접근방법이 효율적임을 쉽게 확인할 수 있다.

NPE	total number of iteration	
	Solver I	Solver II
1	13042	13042
4	13481	14804
9	13042	13527
16	13480	14423
25	13263	14645
36	13265	14679
48	13265	14708
64	13264	14726
81	13264	14530
100	13262	14550

Table 2 Total number of iteration required for convergence

5. 결론

이 연구에서는 SIMPER 유한체적 모델에 대한 병렬화를 수행하였다. 순차형 모델의 결과와의 비교를 통해 병렬 모델의 신뢰성을 검증하였고 성능평가를 통해 우수한 선형성과 병렬 성능을 확인함으로써 병렬 계산의 유용성을 예증하였다. 병렬 성능 결과에 대한 원인 분석을 시도하면서 병렬 성능에 영향을 주는 인자들을 살펴보았고 이 과정에서 국소화된 행렬 해법의 효율성을 확인하였다.

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

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

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

[7] 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.

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

[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.

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

[10] 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).