2004 춘계학술발표회 논문집
한국원자력학회

GCR의 열유동 해석을 위한 비정렬격자 Navier-Stokes Solver LILAC 코드의 병렬화에 대한 연구

Parallelization of the Unstructured Navier-Stokes Solver LILAC for the Aero-Thermal Analysis of a Gas-Cooled Reactor

김종태, 김상백, 이원재

한국원자력연구소
대전광역시 유성구 덕진동 150

요 약

Gen Ⅳ의 원자로 중에 하나인 가스로 특히 고온 가스로의 열유동 해석을 위하여 Lilac 코드를 현재 개발하고 있다. 원래 lilac 코드는 중대사고 시 원자로 하반구에 재배치된 용융물의 자연대류 및 냉각을 해석하기 위하여 개발되었으나 GCR의 고온 고압 가스의 열유동 해석을 위하여 개선되었다. 가스로 내의 기하학적 구조와 이에 따른 열유동의 복잡성이 증가하면 할수록 이를 해석하기 위한 격자의 크기는 기하급수적으로 증가하게 되며 이것은 현재의 사용가능한 테스크탑형 컴퓨터의 성능을 넘어서게 된다. 이를 극복하고 GCR 내의 열유동의 상세 해석을 위하여 영역분할에 기초한 Lilac 코드의 병렬화를 수행하였다. 병렬화된 Lilac 코드의 속도향상을 평가하기 위하여 여러 가지 수치문제를 풀고 그 결과를 평가하였다.

Abstract

Currently Lilac code is under development to analyse thermo-hydraulics of the gas-cooled reactor(GCR) especially high-temperature GCR which is one of the Gen Ⅳ nuclear reactors. The Lilac code was originally developed for the analysis of thermo-hydraulics in a molten pool. And now it is modified to resolve the compressible gas flows in the GCR. The more complexities in the internal flow geometries of the GCR reactor and aero-thermal flows, the number of computational cells are increased and finally exceeds the current computing powers of the desktop computers. To overcome the problem and well resolve the interesting physics in the GCR it is conducted to parallelise the Lilac code by the decomposition of a computational domain or grid. Some benchmark problems are solved with the parallelised Lilac code and its speed-up characteristics by the parallel computation is evaluated and described in the article.

1. 서 론

헬륨이나 이산화탄소를 냉매로 사용하는 gas-cooled reactor(GCR) 특히 고온 가스로(HTGR)는 미래의 청정에너지인 수소를 생산하기 위한 중요한 수단으로 많은 관심과 연구의 대상이 되어가

고 있다[1,2]. GCR은 기존의 경수로 혹은 중수로에서처럼 물(경수 혹은 중수)을 열전달의 냉매로 사용하는 것이 아니라 기체를 냉매로 사용하기 때문에 열유동적 측면에서 기존의 원자로와는 다르다. 이와 같은 GCR의 설계를 위한 그리고 사고시의 열유동 특성을 파악하기 위한 실험과 해석적 연구가 진행 중에 있거나 계획되어 있다[3].

MARS[4] 등과 같은 기존의 원자로 해석 코드들은 기체의 물성치를 계산하는 모듈을 첨가함으로써 가스를 냉매로 사용하는 GCR의 열유동 해석을 위한 노력을 하고 있다. 이와 병행하여 GCR의 원자로 내부의 mechanistic한 상세 해석을 위한 코드의 개선도 이루어지고 있다.

Lilac[5,6]은 노내 노심용융물의 냉각해석을 위하여 개발된 코드로, 비정렬 격자 유한체적법을 기반으로 하는 Navier-Stokes 방정식의 solver이다. Lilac 코드의 특징은 비정렬 격자를 사용함으로써 매우 복잡한 유동장에 대하여 격자의 생성이 용이하며, 유동 해석 방법은 잘 알려진 SIMPLE 알고리즘과 모든 유동 변수를 셀 중심에 저장하는 비엇갈림 셀중심법을 사용한다. 높은 Rayleigh 수의 난류 자연대류 해석을 위하여 여러 난류 모델을 포함하고 있으며 유체와 고체의 상호열전달 즉 conjugate heat transfer의 해석이 가능하고 매우 얇은 배플(baffle)과 같은 벽을 처리할 수 있는 thin-wall 모델 등을 가지고 있다. 본 연구에서는 열수력 해석 코드인 Lilac을 개선하여 압축성 기체의 해석을 가능하게 하고 원자로 내부 열유동의 상세 해석을 위해 매우 큰 격자를 사용하는 경우 계산속도 향상을 위하여 코드의 병렬화를 시도하였다.

복잡하고 매우 큰 유동장에 대하여 작은 스케일의 유동 특성까지 엄밀하게 해석하기 위해서는 격자의 크기 혹은 계산에 사용되는 셀 수가 따라 증가하게 되며 이런 대형 격자에 대한 수치해석을 하기위해서는 병렬계산이 필수적인 요건이 되었다. 코드의 병렬화도 기존의 순차계산과 마찬가지로 하드웨어에 대한 의존성이 매우 높은 방법이며, 컴퓨터 벤더(IBM, SUN, HP, SGI 등)들에 의해 만들어진 매우 빠른 데이터 전송 능력을 지닌 병렬형 수퍼컴퓨터를 사용하는 것과, 1980년대 중후반부터 PC를 묶어(clustering) 병렬 컴퓨터를 구현하는 방법이 사용될 수 있다. 후자는 매우 급속도로 빨라지고 있는 네트워크 기술과 저렴하고 빠른 마이크로 CPU를 활용하여 매우 저렴한 비용으로 병렬형 컴퓨터를 구축할 수 있는 장점이 있다. 본 연구에서는 영역 분할(domain decomposition)에 기초한 SPMD(Single Program Multiple Data) 방법으로 코드의 병렬화를 하였으며 분할된 영역 상호간의 정보전달(message-passing)은 MPI(Message Passing Interface) 라이브러리를 사용하였다. 병렬화된 Lilac 코드의 속도향상 특성을 평가하기 위하여 KISTI에 있는 IBM Regatta 병렬형 수퍼컴퓨터[7]를 주로 사용하여 여러 수치 문제를 해석하였다.

2. 지배방정식과 수치해법

2.1 지배방정식

질량, 운동량 그리고 에너지의 보존방정식을 적분형 일반 방정식으로 표현하면 다음과 같다.

$$\frac{\partial}{\partial t}\int_{\Omega}\rho C\phi \mathrm{d}\Omega + \oint \rho C\phi \vec{V}\cdot \mathrm{d}\vec{A} = \oint \Gamma_{\phi}\nabla\phi\cdot \mathrm{d}\vec{A} + \int_{\Omega} S_{\phi}\mathrm{d}\Omega \quad (1)$$

$$-\begin{bmatrix} 0 \\ A_s u \\ A_s v \\ A_s w \\ \frac{\partial \rho \Delta H}{\partial t} + \nabla(\rho \Delta H \vec{u}) \\ 0 \\ 0 \end{bmatrix}$$

변수 ϕ는 $[1, u, v, w, T, k, \phi_{\varepsilon,\omega}]$이며, C는 변수가 온도 T인 경우만 비열 C_p를 나타내고 그 외에는 1의 값을 갖는다. 운동량방정식은 직각좌표 속도(cartesian velocity)를 종속변수로 사용하며 엔탈피(enthalpy)로 표현된 에너지방정식은 온도에 대해 현시적(explicit)으로 표현되었다. 난류방정식은 2-방정식 모델을 기본으로 사용하고 있으며 $k-\varepsilon$을 사용하는 경우는 $\phi_{\varepsilon,\omega}=\varepsilon$ 이고 $k-\omega$ 모델에서는 $\phi_{\varepsilon,\omega}=\omega$ 이다.

S_ϕ는 원천항을 나타내는 것으로 첫 번째 항은 압력구배와 부력 그리고 열원 등을 포함한다. 두 번째 항은 점도의 변화에 따라 발생하는 점성 플럭스로서 온도에 따른 점도변화, 난류유동의 와점도 등에 의해 발생한다. 그리고 세 번째 항은 용융물의 고화와 같이 상이 변화할 때 발생하는 잠열과 고화된 부분에서 운동량에 대한 저항을 의미한다. 2차원 3차원 유동에서는 r이 1이며, 축대칭 유동인 경우 r은 그 대칭 축(x 혹은 y)을 나타내며 각각에 대한 축대칭 원천항이 발생한다.

2.2 수치 알고리즘 및 이산화 기법

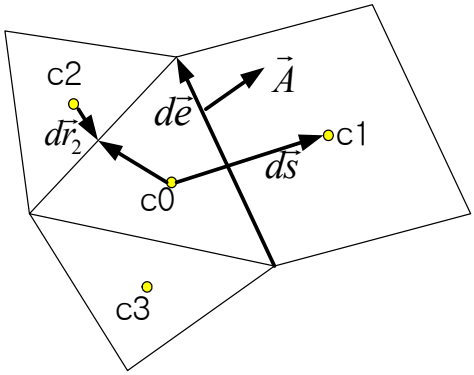

Fig. 1 Control volume and geometric vectors for discretisation.

수치계산 영역은 임의 모양의 셀로 구성되며 각각의 셀이 바로 검사체적이 되고 압력과 속도 성분을 모두 셀의 중심에 저장하는 셀중심 유한체적법을 사용하였다. 각 셀은 여러 개의 면으로 이루어져 있고 각 면의 중심에서 플럭스를 계산하여 적분한다. 대류항은 2차의 상류차분법을 사용하였으며 확산은 주확산항과 엇확산항으로 분리하여 주확산항은 셀 면에서 직접 이산화하고 엇확산항은 이웃하는 두 셀의 중심에서 구한 해의 구배를 이용하여 계산한다. 그리고 비정상항은 2차의 후방차분법으로 이산화하였다. 비정렬격자에서 이산화된 u-운동량 방정식은 다음과 같은 이산화방정식의 형태가 된다.

$$A_0^u u_0 + \sum A_{nb} u_{nb} = S_0^u - \left(\frac{\partial p}{\partial x}\Omega\right)_0 \qquad (2)$$

식 (2)는 셀의 개수만큼 연립방정식을 구성하며 이것은 Bi-CGSTAB을 이용하여 해를 구하게 된다. 비엇갈림 격자법에서 속도는 압력과 함께 셀의 중심에 저장되며 셀 면에서의 속도는 다음과 같은 방법으로 구한다.

셀 면에서의 속도

Rhie등[8]의 압력가중내삽법을 비정렬격자와 같이 임의의 셀 면에 적용하면 다음과 같다.

Rhie-Chow의 PWIM :

$$u_f = u_{CD} - \widetilde{(\frac{\alpha\Omega}{A_p})}[(p_1 - p_0) - \widetilde{\nabla} p\vec{ds}]\frac{n_x}{dn} = u_{CD} - D_p \quad (3)$$

$$u_{CD} = w_f u_0 + (w_f - 1)u_1$$

식 (6)의 압력항은 압력의 3차 미분항으로 이산화된 연속방정식에서는 4차의 압력 미분항으로 작용한다. 이 Rhie-Chow의 PWIM은 중앙차분에 의한 면 속도에 수치적 압력감쇠항을 더한 것으로 표현할 수 있다.

압력수정 방정식

식 (2)의 운동량방정식과 전단계 압력을 이용한 운동량방정식의 차로부터 보정속도 u'에 관한 식을 얻는다. 그리고 $u'_0 = u'_{nb}$ 라는 SIMPLEC의 가정을 도입하여 정리하면 다음과 같은 속도수정 방정식을 얻을 수 있다.

$$u'_0 = \frac{-\Omega_0}{Ap_0^u + \sum A_{nb}}(\frac{\partial p'}{\partial x})_0 = -d_0^u(\frac{\partial p'}{\partial x})_0 \quad (4)$$

한편 압축성 기체인 경우에는 밀도가 변하기 때문에 연속방정식의 수정 질량유량에서 밀도의 변화를 고려하여야한다. 먼저 밀도 변화를 압력변화로 표현하면 다음과 같다.

$$\rho' = \left(\frac{\partial\rho}{\partial p}\right)p' = \left(\frac{1}{\gamma RT}\right)p' = \left(\frac{\rho}{\gamma p}\right)p' \quad (5)$$

$$J' = \rho'\theta^* + \rho^*\theta' = \frac{\rho}{\gamma p}p'\theta^* - \rho^* Q(p'_1 - p'_0) \quad (6)$$

식 (6)은 밀도의 변화를 고려한 수정 질량유량으로 우변의 첫 번째 항은 압력수정방정식에서 대류항ㅇ과 같은 역할을 하며 두 번째 항은 확산항의 역할을 하게 된다. 두 번째 항의 Q는 압력구배를 포함하는 항으로 비압축성 유동해석의 경우와 마찬가지로 식 (7)을 이용하면 식 (8)과 같이 표현된다.

$$\nabla p'_f \approx (p'_1 - p'_0)\frac{\hat{n}}{\overrightarrow{ds^*}\,\hat{n}} \quad (7)$$

$$Q = d_f^u\frac{n_x}{\overrightarrow{ds^*}\,\hat{n}}A_x + d_f^v\frac{n_y}{\overrightarrow{ds^*}\,\hat{n}}A_y + d_f^w\frac{n_z}{\overrightarrow{ds^*}\,\hat{n}}A_z \quad (8)$$

식 (8)과 식(6)을 이용하여 연속방정식을 이산화하면 압력수정방정식을 얻게 된다.

2.3 코드의 병렬화

유체를 연속체라고 가정하고 해석을 하는 CFD 코드의 병렬화에는 크게 두 가지 방법이 많이 사용되고 있는데, 첫째는 OpenMP[7]라는 multi-thread 개념을 이용하는 방법으로 해석영역의 분할 없이 전체 해석 영역에 대하여 하나의 CPU가 한 프로세스를 가지고 수행하면서 Do-loop의 크기를 를 가용한 CPU 개수만큼 나누고 쓰레드라는 메인 프로세스에 종속적이고 일시적인 프로세스를 생성하여 각각의 CPU가 자신에게 할당된 크기만큼의 Do-loop를 처리하는 방법이다. 이 방법의 특징은 코드의 큰 수정 없이 병렬화가 필요한 Do-loop에 directive를 코드에 삽입하거나 혹은 컴파일러가 자동으로 가능한 Do-loop를 병렬화하기 때문에 사용하기 쉽다는 장점이 있으나 반드시 SMP(Shared Memory multi-Processors) 컴퓨터와 OpenMP를 지원하는 컴파일러가 필요하고, 쓰레드의 생성과 소멸이 빈번하게 이루어지기 때문에 병렬화에 대한 효율이 상대적으로 낮다. 반면에 message-passing을 기반으로 하는 영역분할 병렬화는 모든 multi-CPU 컴퓨터와 네트워크로 연결된 PC-cluster까지 광범위한 컴퓨터 하드웨어를 사용할 수 있으며 병렬화에 의한

효율이 OpenMP보다 월등히 높고 가용한 CPU의 개수를 최대한 활용할 수 있기 때문에 MPP(Massive Parallel Processing)이 가능하다. 본 연구에서는 영역분할과 정보전달에 기반을 둔 Lilac 코드의 병렬화를 수행하였다. CPU간 혹은 프로세스 간의 정보전달을 위하여 message passing 라이브러리가 사용되는데 현재 PVM(Parallel Virtual Machine)[9]과 MPI(Message Passing Interface)[10] 라이브러리가 사용되고 있다. PVM과 MPI 서로 장단점이 있으나 국제 표준이 되고 있고 모든 컴퓨터에서 프로그램이 이식성이 뛰어난 MPI를 사용하여 Lilac 코드를 병렬화하였다. MPI를 이용하여 코드를 병렬화하는 경우 제일 먼저 고려하여할 사항은 해석영역의 분할이다. 일반적으로 정렬격자는 격자를 생성할 때 multi-block으로 생성하여 각각의 격자 블록을 하나의 부영역으로 처리하는 것이므로 영역 분할은 실제로 수작업으로 이루어진다고 볼 수 있으며 가용한 CPU의 수가 늘어남에 따라 많은 수의 블록으로 격자를 생성하는데 어려움이 있다. 한편 비정렬격자는 주로 해석하려는 전체 영역에 대하여 격자를 생성한 뒤 영역분할을 하게 되는데 이 영역분할은 CPU간의 부하를 균등하게 만드는 load-balancing과 영역 경계를 가능한한 줄여 정보전달 양을 줄이는 방향으로 영역분할을 하게 된다. 현재 RCB(Recursive Coordinate Bisection), RSB(Recursive Spectral Bisection), RGB(Recursive Graph Bisection) 등 여려가지 방법이 개발되어 사용되고 있으며, 특히 Karypis등[11]이 개발한 공개 소프트웨어인 Metis는 비정렬 혼합격자를 매우 빠르게 영역분할을 하기 때문에 가장 많이 사용되고 있다. 본 연구에서는 격자의 분할을 위하여 주로 Metis를 사용하였으나 죄표축 방향 혹은 주유동방향으로 격자의 셀을 renumbering하면서 동일한 셀 수로 분할하는 방법도 사용하였다. 영역 분할한 격자는 영역 상호간에 정보 전달을 위한 격자 연결 정보가 필요한데 부영역간에 중첩된 부분을 두고 이 부분에서 수치적 자료를 서로 교환하게 되면 부영역의 내부는 순차해석 방법과 동일하게 된다.

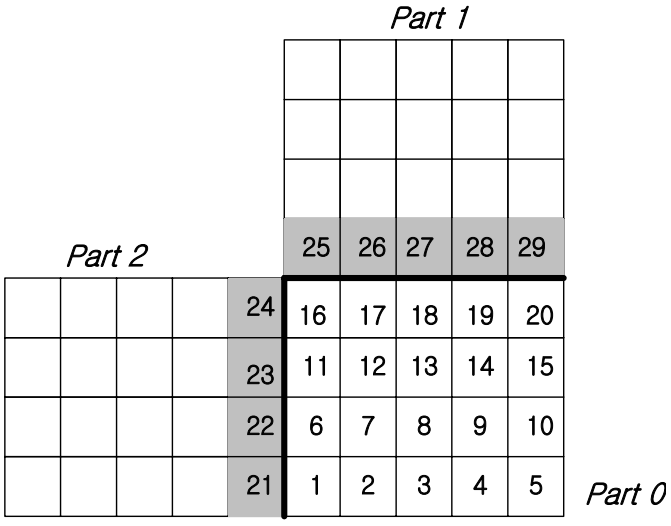

Fig. 2 Partitioned mesh and halo cells around part 0 sub-domain.

분할된 영역은 이웃한 영역과 겹치는 영역이 존재하며 이 영역을 fringe layer 혹은 halo area라고 부른다. Fig. 2는 3개의 영역으로 분할된 격자로서 part 0의 halo 셀을 보여주고 있다. 각각의 영역은 각각의 프로세스가 격자 정보를 읽어 들이고 계산을 수행하게 되는데, 위의 그림에서 세 개의 프로세서가 병렬로 동시에 작업을 수행하게 된다면 총 프로세스의 수(nprocs)는 3이 되고 프로세스 0이 part 0의 격자 정보를 읽어 들인다. 0번 영역(part 0)은 총 20개의 셀로 구성되어 있고 halo 셀은 9개가 된다. 본 연구에 사용된 Lilac 코드는 모든 수치적 플럭스를 셀의 face를 기준으로 계산하게 되며 고차의 대류 플럭스와 점성플럭스 계산을 위해 필요한 해의 구배는 셀 중심에서 재구성을 하기 때문에 영역간 경계(interpartition boundary)에 놓여 있는 셀면에 대해서 오직 한 개의 셀 정보만 알면 수치 적분이 가능하다. 그래서 part 0의 halo 셀은 영역간 경계에 바

로 이웃한 셀만으로 구성되기 때에 전체 halo 셀의 수가 상대적으로 적다. Part 0의 halo 셀은 part 0의 내부 셀 마지막 번호 이후부터 번호가 매겨지게 된다. Fig. 2에서 part 0는 halo 셀 21번부터 24번까지의 수치 값을 그 셀과 일치하는 part 2의 내부 셀로부터 전달받는다. 마찬가지로 25번부터 29번까지의 halo 셀의 값은 part 1로부터 전달 받아 수치적으로 이산화를 하게 된다.
현시적인(explicit) 방법으로 지배방정식을 이산화하는 경우에는 방정식의 반복계산 처음에 각 영역의 halo 셀에 이웃하는 영역으로부터 최근의 해를 업데이트함으로써 자료전달은 끝나고 각 영역에서 순차해석과 마찬가지로 각 셀에서 residual을 구하면 되지만, 내재적인(implicit) 방법으로 지배방정식을 이산화하면 이산방정식이 매트릭스 형태로 만들어지며 매트릭스의 계산중에도 자료의 교환이 이루어진다.

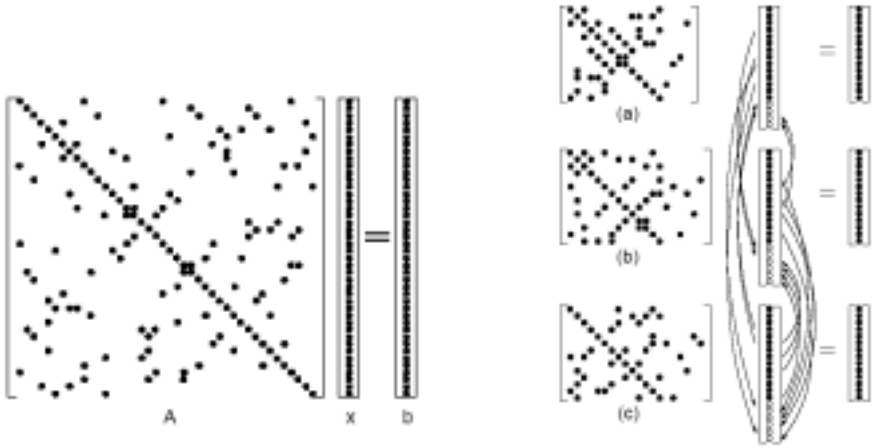

Fig. 3 (a) Matrix for the 42 triangular mesh, (b) Matrices for the 42 triangular mesh partitioned into three sub-domain; from Kevin McManus

Fig. 3(a)는 42개의 삼각형 셀로 구성된 격자를 이용하여 얻은 이산방정식의 매트릭스를 그림으로 나타낸 것이다. 매트릭스 내의 각 점들은 0이 아닌 계수가 존재하는 곳을 의미하여 수학적으로 각 노드의 연결성을 의미하기도 한다. 이 격자를 세 개의 영역으로 분할하면 fig. 3(b)에서 보는 것과 같은 영역별의 매트릭스를 얻을 수 있다. 그러나 이 경우 영역간 경계에 존재하는 셀이 이웃하는 영역의 셀과의 사이에서 이산화방정식의 계수가 만들어지지만 그 영역의 셀이 아니기 때문에 그림에서 보는 것처럼 매트릭스가 augmented된다. 그림에 화살표로 서로 이어진 부분이 영역간의 자료전달(message passing)을 의미하며 매트릭스 계산중에 이런 자료전달이 반복적으로 이루어지게 된다.

3. 결과 고찰

3.1 3차원 범프 주위의 압축성 유동 해석

입구에서 마하수 1.4로 유동이 유입하여 4% 원형 돌기(bump)의 앞과 뒤에서 충격파를 만들고, 특히 앞에서 만들어진 충격파는 위쪽의 벽면에서 반사되고 아래 벽면에서 다시 반사되는 복잡한 충격파 형상을 갖는다. 유동장의 기하학적 형상은 단순한 반면 유동이 만들어내는 충격파의 형상은 매우 복잡하여 압축성 Euler solver를 검증하는 문제로 많이 사용된다. GCR과 같이 압축성 기체에 의해 열전달이 이루어지는 시스템의 열유동 해석을 위해서는 압축성 기체의 질량 및 열량의 보존을 충분히 만족시키는 해석 툴을 사용하여야한다. 본 연구에서는 GCR의 열유동 해석을 위하여 개발하고 있는 Lilac 코드의 압축성 유동에 대한 검증 및 코드의 병렬화에 대한 계산속도 향상을 평가하기 위하여 초음속으로 범프 채널에 유입하는 유동을 해석하였다. 유동장 입구는 촘음

속 입구조건으로 대기압을 기준으로 마하수 1.4를 만들어내는 총압(total pressure)과 총엔탈피(total enthalpy)를 주었으며 출구는 모든 곳에서 초음속으로 유동이 유출되므로 모든 변수의 값이 내부 유동장에서 외삽되는 초음속 유출경계조건을 사용하였다. 사용된 격자는 먼저 정렬격자(101□31□31)를 생성한 뒤 육면체(hexa)의 비정렬격자로 분할하여 만들었으며 격자의 총 사용된 셀(cell) 수는 90,000개이다. 이 문제에 대하여 Metis를 이용하여 4개의 파트로 나누어 병렬계산을 수행하였으며 fig. 4에서 영역분할된 격자를 보여준다.

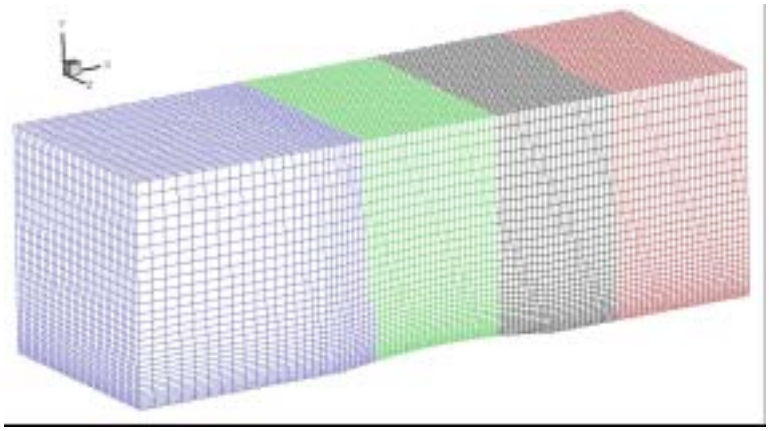

Fig. 4 Partitioned hexahedral mesh for the analysis of the inviscid supersonic flow over a 4% bump, ncell = 90,000, nparts = 4.

이 유동장은 z축 방향으로 유동에 변화가 없는 2차원 유동으로 fig. 5의 등압력선도에서 보는 것처럼 z방향으로 대칭적인 압력 패턴을 잘 보여 주고 있다. 충격파는 유동의 불연속점(혹은 선)으로 일반적인 고차의 대류항 기법을 사용하는 경우 충격파 근처에서 수치불안정성을 피할 수 없다. Lilac 코드에는 유동의 불연속점에서 대류 플럭스의 고차항을 제한하는 방법을 사용하고 있는데 2차의 중앙차분에 인공점성항을 첨가하는 방법, van alvada의 제한자(limiter)를 사용하는 TVD(Total Variation Diminishing) 기법, NDV(Normalized Variable Diagram)을 이용하는 기법 등을 갖추고 있다. Fig. 5는 2차의 중앙 차분에 10%의 1차 상류차분을 더한 블렌디드(blended) 기법을 사용하여 얻은 결과를 보여준다.

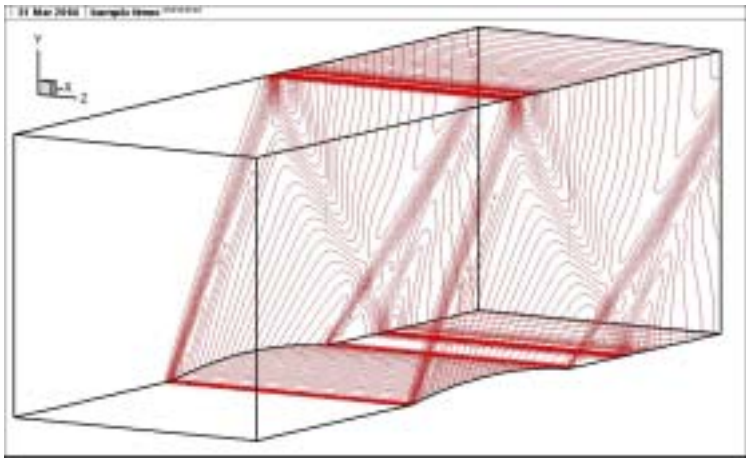

Fig. 5 Computed pressure contours of the 3-dimensional inviscid supersonic flow over a 4% bump, inlet Mach number is 1.4.

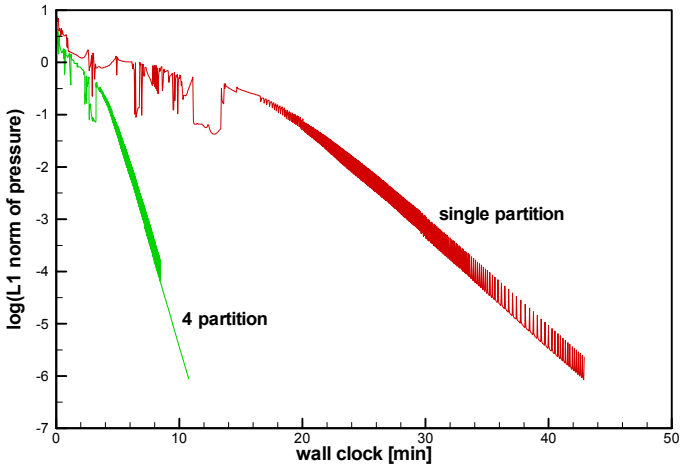

Fig. 6 Comparison of the convergence rates for the supersonic bump flow between single and 4-partitioned computational domains.

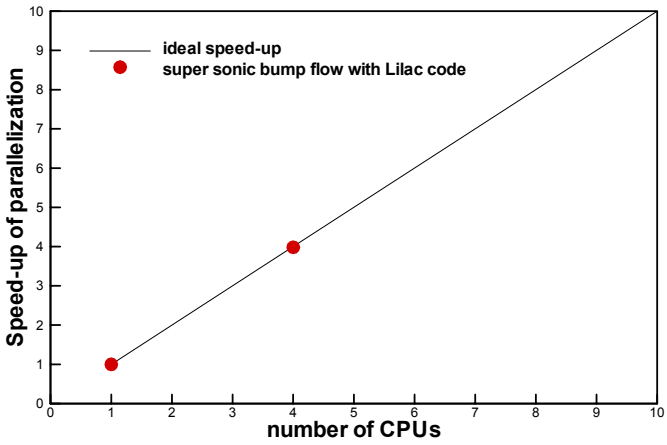

Fig. 7 Speed-up of the parallelized LILAC code for the supersonic bump flow.

3차원 범프의 초음속 유동장 해석은 IBM의 Regatta에서 수행되었으며, 단일 영역(single partition)으로 순차계산을 한 경우와 4개의 영역(4 partitions)으로 분할하여 계산한 경우에 대하여 수렴 특성 및 속도향상을 비교하였다. Fig. 6은 시간에 따른 압력 잔차의 변화를 보여주는 그림이다. 모든 계산은 ILU-preconditioned Bi-cgstab을 이용하여 이산방정식의 매트릭스를 구하였다. 앞에서 언급한 것처럼 영역의 경계에 있는 셀에서는 이웃하는 영역의 updated된 값을 사용하지 못하고 한번의 반복 계산이 끝난 후에 MPI를 이용한 정보의 교환이 이루어지므로 시간지연(혹은 Jacobi-like effect)으로 인한 수렴의 특성이 순차계산과는 다르게 나타나지만 수렴속도에 크게 나쁜 영향을 미치지 않는 것으로 보인다. Fig. 7은 병렬계산에 의한 계산속의 향상을 이상적인 것과 비교한 것으로 4개의 CPU를 사용한 경우 4배의 속도 향상이 있는 것으로 계산 결과 나타났다. 일반적으로 병렬계산 시 정보전달을 위한 시간이 필요하기 때문에 병렬효율은 100%보다 낮게 나타난다. 이 계산의 경우 전체 계산시간 중에 5%(이 경우 34초)가 MPI의 정보전달(message passing)에 사용되었다. 그러나 이 문제의 경우 병렬 효율이 100%로 나타난 것은 CPU의 구조와 무관하지 않다. 벡터형 수퍼컴퓨터를 제외한 마이크로프로세서는 낮은 메모리 접근 속도를 보완하기 위하여 캐시(cache)를 가지고 있는데 IBM Regatta에서 사용된 Power4 CPU는 매우 큰 캐시를 내장하고 있으며 이로 인하여 계산에 필요한 데이터 스트림(data stream)의 캐시 미스(cache miss)가 줄어들었기 때문으로 판단된다.

2.2 원형단면을 갖는 90° 곡관 유동

Bovendeerd등[12]은 원형 단면을 갖는 90° 곡관내의 비압축성 층류유동을 실험하였다. 관의 직경(D)과 평균유동속도(V)를 기준으로 한 레이놀즈 수는 700이며, 관의 곡률반경은 3D이고 90° 곡관부 입구에서 완전히 발달된 유동이 유입하도록 하기 위하여 직관부의 길이를 50D로 하였다. 이 수치해석에서는 5D의 직관부를 곡관부 앞에 두고 입구조건은 완전히 발달된 속도분포를 해석해를 이용하여 주었다. 유동은 x축 방향으로 유입하여 곡관부를 지나 y축 방향으로 유출하며 xy평면에 대칭이다. 해석기법이 격자의 비대칭성에 무관하게 대칭유동을 만들어냄을 보이기 위하여 전체 관에 대하여 해석하였다. Fig. 8은 사용한 격자를 보여주며, 먼저 입구의 원형단면에서 삼각형과 사각형의 혼합격자를 생성한 뒤 관의 중심축을 따라 적층하여 프리즘과 육면체로 구성된 75,024 개의 셀로 구성된 격자를 생성하였다. 정렬격자로 원형곡관 내의 격자를 생성하는 경우 일반적으로 특이점을 포함하거나 찌그러짐이 큰 격자의 생성을 피할 수 없으나, 비정렬 혼합격자를 사용함으로써 격자의 찌그러짐을 거의 없앨 수 있다. Fig. 8(b)는 병렬계산을 위하여 4개의 영역으로 분할된 격자를 보여준다. 90도 곡관 유동장의 격자는 단면에 비해 주유동 방향으로의 길이가 매우 길고 각 단면에서의 격자수가 주유동 방향으로 일정하기 때문에 주유동 방향을 축으로 하여 셀을 다시 번호 매김(renumbering) 하면서 영역별로 동일한 개수의 셀 수를 갖도록 영역 분할하는 것이 직관적이며 쉬운 방법이이다. 이렇게 함으로써 영역간의 경계를 최소화할 수 있으나, 입구와 출구를 갖는 영역이 내부의 영역과 달리 하나의 다른 영역과 접한다는 특성을 가지고 있다.

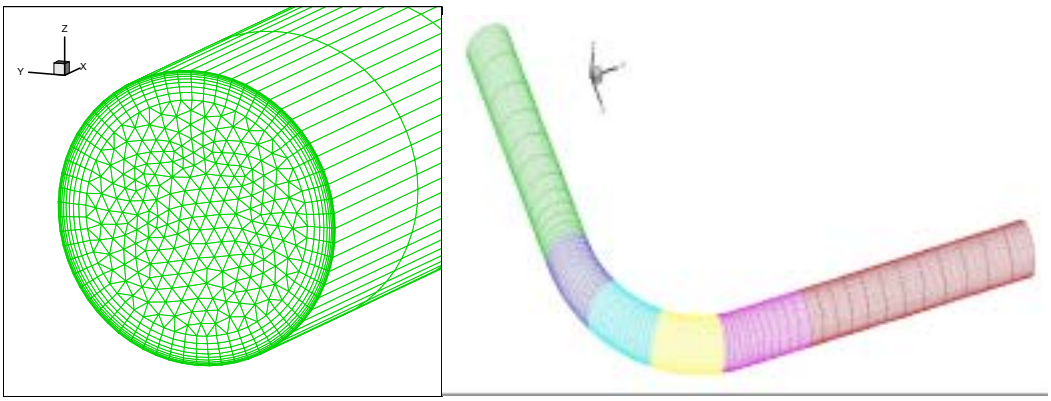

Fig. 8 (a)Hybrid mesh for the 90° curved pipe, ncell=75,024, (b) partitioned computational mesh, nparts=6

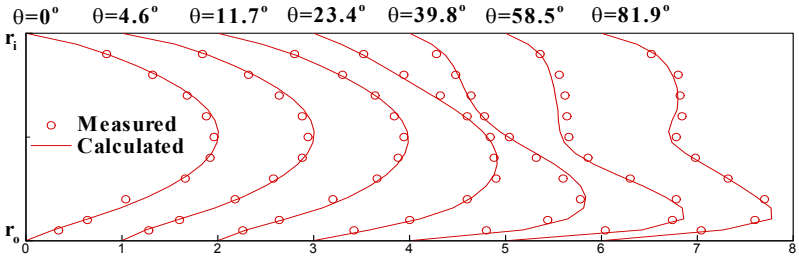

Fig. 9 Measured and calculated streamwise velocity profiles on the plane of symmetry: o measurements(Bovendeerd et al.), - calculated.

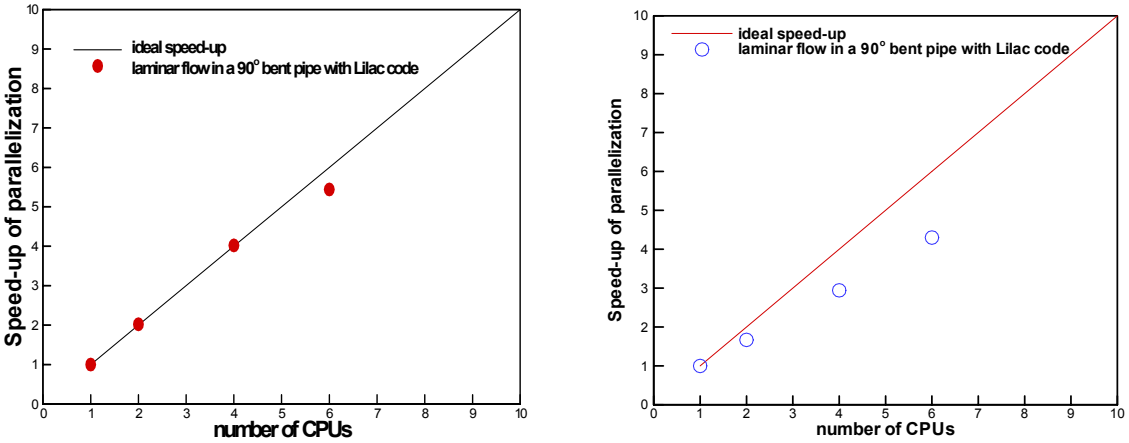

Fig. 10 Speed-up of the parallelized LILAC code for the laminar flow in the 90° curved pipe, (a) which is based on the time spent for 200 iterations. (b) based on the time to reach 4 order reduction of L1-norm of pressure equation.

Fig. 9에서는 곡관부에서의 주유동방향의 속도성분을 실험값과 비교하였으며, 잘 일치함을 알 수 있으며, 특히 영역분할에 의한 병렬계산에서 계산 중 해의 정확한 전달이 이루어지고 있음을 짐작할 수 있다. Fig. 10은 90도 곡관 유동에 대한 Lilac 코드의 병렬 특성을 보여주기 위한 그림으로 (a)는 반복회수가 200회에 도달할 때까지의 계산시간을 기준으로 속도향상 지수를 그린 것이고 (b)는 압력방정식의 잔차가 4 order 감소할 때까지 걸린 계산시간을 기준으로 한 병렬의 속도향상을 나타낸다. 반복회수에 기준으로 한 것은 거의 사용한 CPU 수에 비례하게 나타나지만, 잔차를 기준으로 한 것은 6개의 영역으로 분할된 경우 4.3배의 속도 향상을 얻었으며 이는 병렬효율로 보면 약 71% 정도를 의미한다. CFD 코드와 같이 이산방정식의 성긴 매트릭스(sparse matrix)를 반복적으로 푸는 코드의 병렬화에서 매트릭스 solver의 병렬화가 가장 큰 걸림돌로 알려져 있다. 본 연구에 사용된 Lilac 코드는 성긴 매트릭스를 풀기 위하여 Jacobi, Gauss-Seidel, preconditioned Conjugate-gradient에 기초한 CG, Bi-CGSTAB 등을 가지고 있으며,. Jacobi 반복법을 제외한 다른 방법들은 반복계산 내에서 분할영역 경계의 셀 들은 가장 최근의 엡데이트(updated)된 값을 요구한다. 그러나 분할영역 경계(inter-partition boundary)에 놓인 셀은 그와 접한 셀이 이웃 영역에 위치하고 이로 인하여 최근의 업데이트된 값을 받기 위해서는 셀 단위로 MPI를 통한 정보전달이 이루어져야한다. 그러나 이것은 많은 MPI 정보전달의 시간을 소비하는 방법으로 병렬효율을 심각하게 떨어뜨릴 수 있기 때문에, 경계 셀은 Jacobi와 같은 영향을 받게 된다. 90도 곡관내 유동장은 parobolic한 특성을 갖기 때문에 분할영역 경계에서 Jacobi와 같은 특성은 해의 수렴에 큰 영향을 미칠 수 있는 것으로 생각되며 fig. 10(b)에서와 같은 상대적으로 낮은 병렬효율을 보인다.

2.3 3차원 Lid-driven cavity 유동

3차원 lid-driven cavity 내의 비압축성 유동장을 수치해석하였다. 가로, 세로, 높이의 비가 1:1:1인 정육면체의 공동에서 위쪽 평판이 UB의 속도로 움직이며 공동 내부에서는 중심에 큰 와동이 생기고 아래 두 모서리에서 corner-vortex가 형성되는 유동 구조를 갖는다. 이 문제는 2차원에서 cavity의 크기와 평판의 속도를 기준으로 한 Reynolds수 10,000까지에 대하여 수치기법의 정확도를 평가하기 위하여 많이 사용되어 온 문제 중의 하나이다. 여기서는 격자의 크기를 ncell=1,000,000으로 하여 코드의 병렬화 특성을 평가하기 위하여 수치해석을 하였으며 계산된 Reynolds수는 1,000과 3,200 등이다.

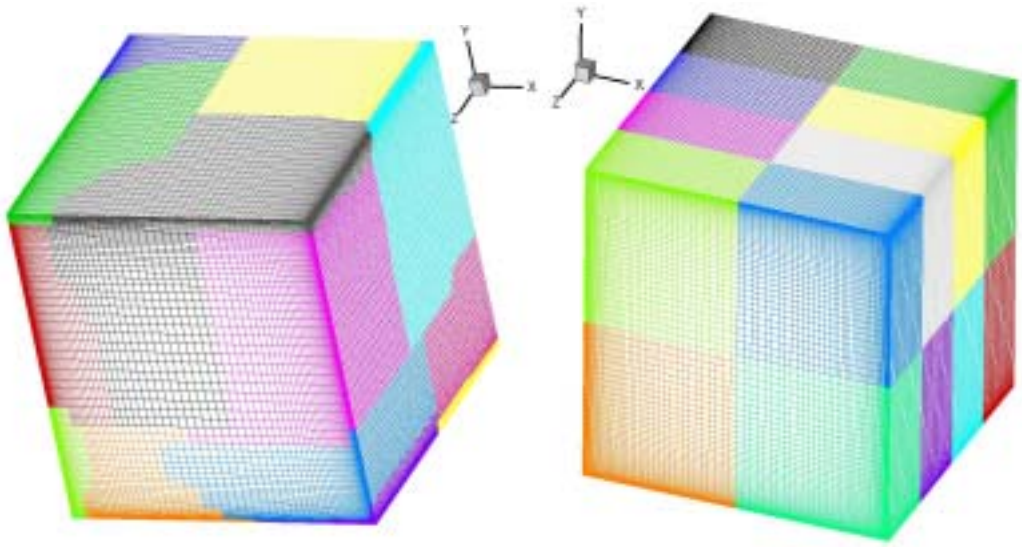

Fig. 11 Partitioned mesh for the cavity flow, ncell=1,000,000, nparts=16, (a) by the Metis program, (b) by axis-directional partitioning.

Fig. 9는 백만개의 육면체 셀로 구성된 cavity의 격자를 16개의 영역으로 분할한 모습을 보여주는 그림으로 fig. 11(a)는 Metis를 이용하여 영역분할을 한 것이고 fig. 11(b)는 x, y, z축을 기준으로 셀을 다시 번호 매김(renumbering)하면서 영역을 분할한 것이다. Metis를 이용하는 경우 벽면에 밀집된 격자의 형상으로 인하여 벽면으로 갈수록 얇은 쉘 모양으로 분할된 영역이 만들어지며, 이로인하여 해의 수렴에 나쁜 영향을 미칠 수가 있다. 반면에 좌표축을 기준으로 영역분할을 한 경우 벽면에서 전단력에 의하여 발생하는 운동량방정식의 residual이 쉽게 중심으로 확산될 수 있을 것으로 생각되어 fig. 11(b)의 분할된 격자를 사용하여 계산을 수행하였다.

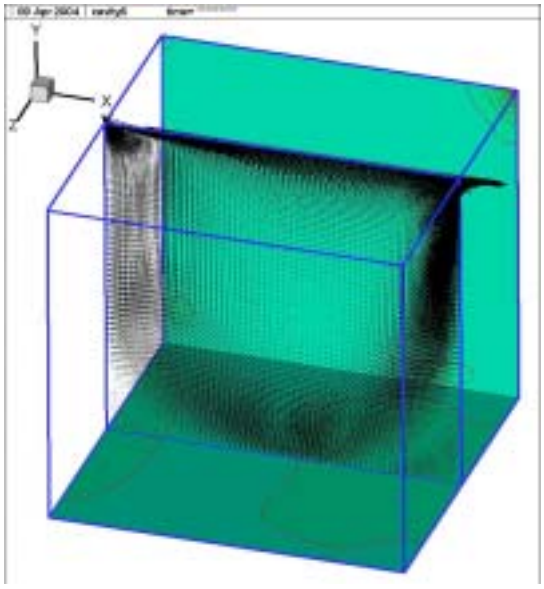

Fig. 12 Velocity vectors on the symmetric plane and pressure contours on the side and bottom walls, ncell=1,000,000, nparts=16, Re=1,000

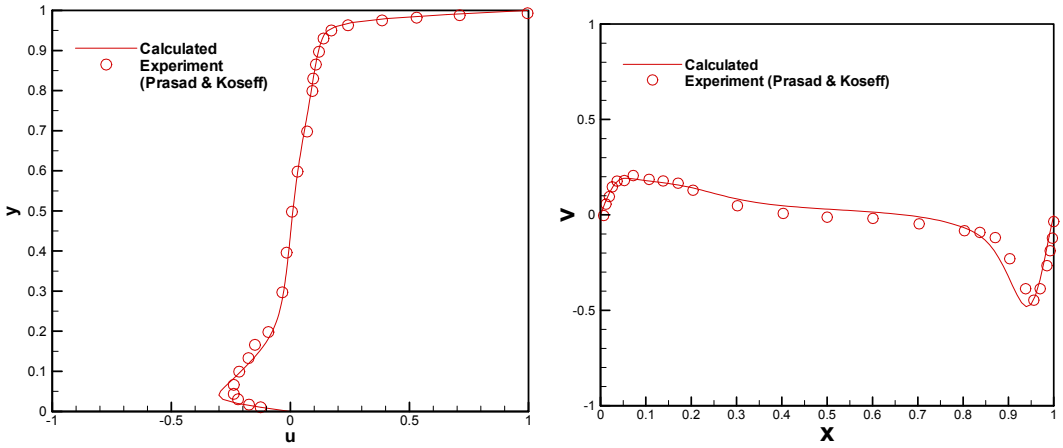

Fig. 13 Comparisons of the velocity profiles along the center line of the cavity, Re=3,200

Fig. 12는 Re=1,000인 경우에 대하여 z=0.5인 대칭 단면에서 속도장과 y=0, z=0인 벽면에서 압력장을 보여준다. 그리고 Fig. 13에서는 Re=3,200인 경우 cavity 중심축을 따라 수치결과를 Prasad 등[13]이 실험에서 얻은 결과와 비교하였다. 3차원 cavity 유동은 Reynolds수가 32,00에서 엇유동 방향으로 Taylor-Goertler Like (TGL) vortex가 형성되고 시간이 따라 진동하는 유동 특성을 보여준다. 이 계산에서는 정상상태로 가정을 하고 수치계산을 수행하였으며 시간평균된 실험값과 비교하였다. 단일영역의 순차계산과 16 영역으로 분할된 병렬계산에서 수치 잔차의 수렴특성은 서로 비슷하였으며 반복회수 1,000까지 계산하는데 걸린 시간을 기준으로 병렬계산의 속도향상 factor를 계산하면 S_p = 15.5로 이상적 속도향상 값인 16에 매우 근접한 값을 보여주었다.

2.4 3차원 실린더 주위의 층류 유동

원형 실린더 주위의 층류 유동은 직경을 기준으로 Reynolds 수에 따라 후류에 만들어진 와가 이탈하는 비정상 유동이 만들어진다. Reynolds 수가 40인 경우에는 실린더 후류에 생성된 와의 이탈은 발생하지 않으며 정상 유동을 형성한다. Coutanceau등[14]의 실험에 의하면 실린더 직경으로 무차원화한 실린더의 뒤 정체점으로부터의 유동 박리의 길이(Lsep)는 2.13으로 측정되었다.

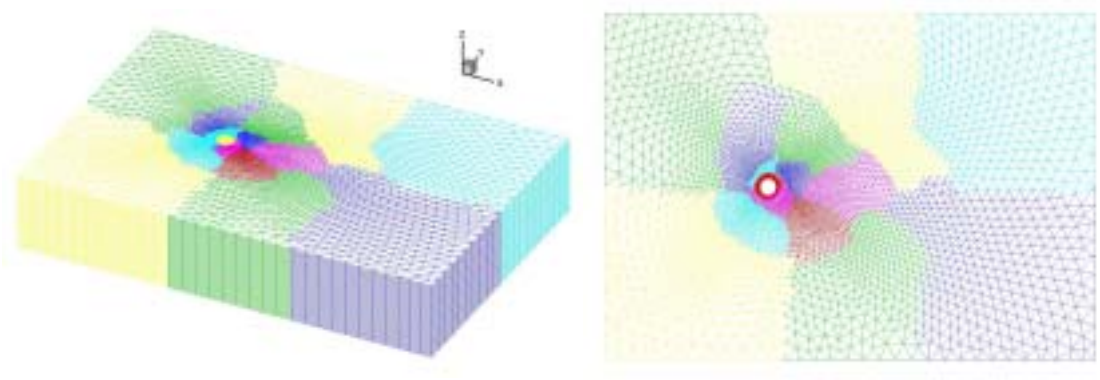

Fig. 14 Partitioned mesh for the 3-D cylinder flow by the Metis program, ncell=470,400, nparts=16

이 계산에서는 총 470,400 개의 프리즘과 육면체 셀로 구성된 격자를 이용하여 병렬화 효율을 검증하였다. 격자는 먼저 2차원 단면에서 사각형과 삼각형을 혼합한 격자를 생성한 뒤 z축 방향으로 적층하여 3차원 격자를 생성하였으며, 실린더 벽면에는 육면체의 격자로 구성하였다. Fig. 14 Metis로 영역 분할된 격자를 보여준다. 전체 해석 영역은 Metis를 이용하여 총 16개의 영역으로

분할되었으며 각 영역(partition)이 자료교환을 위해 필요로 하는 halo 셀의 수는 최대 5,623개에서 최소 1,898개까지 퍼져 있는 것으로 나타났는데 이는 영역별로 정보 전달량이 큰 차이가 있음을 의미한다.

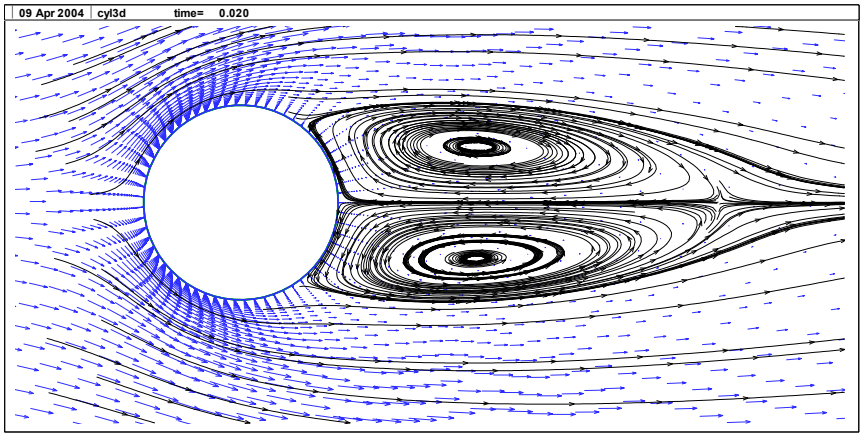

Fig. 15 Velocity vectors and stream lines at z=2.5 (center surface in z direction) for the 3-D cylinder flow with Red=40.

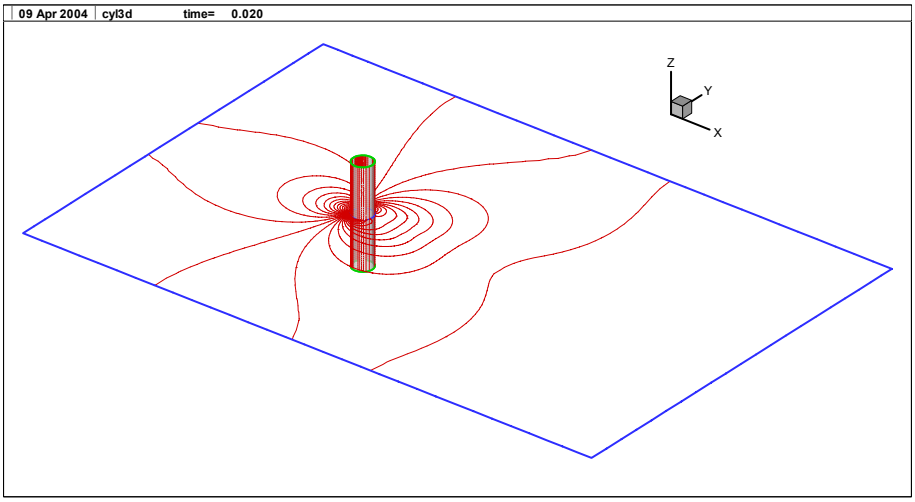

Fig. 16 pressure contours at z=2.5 (center surface in z direction) for the 3-D cylinder flow with Red=40.

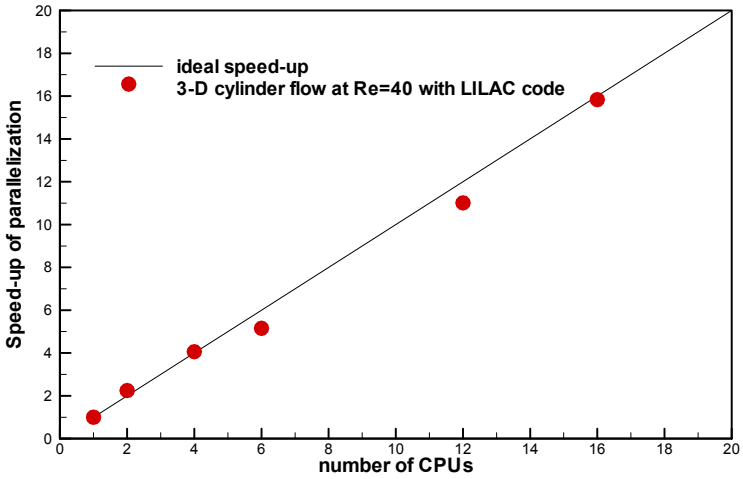

Fig. 17 Speed-up of the parallelized LILAC code for the 3-D cylinder flow with Re=40.

이 유동장은 z축 방향으로 실린더 직경의 5배이며 z=0과 z=5의 경계면에서는 대칭조건을 사용하였고 입구에서는 속도와 압력을 고정하는 입구조건을, 출구에서는 압력을 고정하는 출구압력조건을 사용하였다. Fig. 15는 z=2.5의 대칭면에서 속도장과 유선을 그린 것으로 계산에서 얻은 박리길이는 2.0으로 실험치와 비교하여 약 6% 정도의 차이를 보였다. Fig. 16은 fig. 15와 같은 단면에서 압력장을 보여주는 것으로 병렬계산에 사용된 여러 영역을 걸쳐 등압력선이 부드럽게 이어지는 것을 볼 수 있으며, 이것은 영역간 정보전달이 정확이 이루어지고 있는 것을 의미한다. 이 문제에 대하여 병렬의 속도향상 특성을 파악하기 위하여 10번의 반복계산에 사용된 시간을 사용한 CPU의 개수 즉 분할 영역의 수(number of partitions = number of CPUs)에 따라 비교하였다. 두 개의 CPU를 사용한 경우에는 이상적 속도향상(ideal speed-up)보다 높게 나타났는데 이것은 앞에서도 언급한 것처럼 CPU 당 처리하는 셀 수가 작아짐에 따라 캐시-미스(cache-miss) 수가 줄어들어 얻어진 효과로 판단된다. 일반적으로 코드의 병렬화 작업 후 그 코드의 병렬특성이 scalability가 있는가를 속도향상 값으로 판단하게 되는데 사용하는 CPU의 수다 늘어나면서도 이상적 속도향상(ideal speed-up) 근접한 값을 갖는 경우 코드의 scalability가 좋다고 표현하고 어느 정도의 CPU 사용까지는 속도향상이 증가하다가 더 이상 CPU 수를 늘려도 속도개선이 보이질 않는 경우 stall 되었다고 표현한다. 그리고 CPU 수보다 계산의 속도향상이 더 큰 경우를 super-scalability라고 부르며 이것은 주로 CPU의 하드웨어적 구조에 의해 Do-loop의 크기가 줄어들면서 캐시-히트(cache-hit) 수가 늘어나서 발생한다. Fig. 17에서 보는 것처럼 두 개의 CPU를 사용하는 경우에는 속도향상이 2보다 큰 super-scalability 현상이 나타난 것을 알 수 있으며 전체적으로 보면 CPU를 16개까지 사용한 결과 Lilac 코드의 병렬특성은 scalability가 좋다고 판단된다.

4. 결론 및 향후 과제

본 연구에서는 GCR의 열유동 해석을 위한 Lilac 코드를 병렬화하였으며, 여러 수치 문제를 통하여 코드의 병렬화에 의한 속도향상을 평가하였다. 해석 격자를 분할하기 위하여 Metis 프로그램과 좌표축을 따라 셀을 renumbering하면서 load-balancing이 되도록 분할하는 방법을 사용하였다. Metis는 일반적인 유동영역에 대하여 셀의 모양에 관계없이 매우 빠르게 영역을 분할하는 것을 알 수 있었다. Lilac 코드의 병렬화는 MPI 라이브러리를 사용하여 영역 간에 정보전달(message passing)을 하였으며 정보전달을 위한 독립적인 모듈을 구성함으로써 코드의 전체 구조

에 큰 변화가 없이 병렬화가 가능하도록 하였다. 여러 가지 문제의 해석을 통하여 병렬화에 의한 속도향상이 매우 크며 scalability가 뛰어남을 알 수 있었다. 그러나 매트릭스 solver에서 Jacobi 효과에 의해 반복계산에 대한 수렴성이 약간 약화될 수 있음을 보였으며, 병렬 매트릭스 solver에 대한 연구가 필요하다고 판단된다. 현재 난류 유동에 대한 해석을 수행 중에 있으며 난류유동 해석에서 병렬화의 속도향상을 평가할 계획이다.

후기

본 연구는 과학기술부 중장기과제에서 재정적 지원을 받았습니다.

참고문헌

[1] Fracck Carre, et. al., "Generation IV Roadmap Description of Candidate Gas-cooled Reactor System Report," GIF-016-00

[2] R. A. Moore, M. E. Kantor, et al., "HTGR Experience, Programs, and Future Applications," Nuclear Engineering and Design 72(1982) 153-174

[3] Yuanhui Xu, "High Temperature Gas-Cooled Reactor Programme in China," Proc. Of the Conference on High Temperature Reactor, Petten, NL, April 22-24, 2002

[4] 이원재 외, “고온가스로의 열전달 모델 예비 평가,” 2003 한국원자력학회 추계학술대회 논문집

[5] 김종태 외, “노내 노심 용융물의 냉각 해석 코드 LILAC-meltpool 개발,” KAERI/TR-2126, 2002

[6] 김종태 외, “다차원 열유동 해석을 위한 Lilac 코드의 검증,” 2001 한국원자력학회 추계학술대회 논문집

[7] “병렬형 슈퍼컴퓨터 기본사용법,” 한국과학기술정보연구원, KISTI, 슈퍼컴퓨팅센터 저, 2002

[8] Rhie, C.M. and Chow, W.L., "Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation," AIAA J. Vol.21, No.11, pp.1525-1532, 1983

[9] Geist, A. et al., "PVM 3 User's Guide and Reference Manual," ORNL/TM-12187, Oak Ridge National Lab., 1994

[10] Gropp, W. et al., "Using MPI Portable Parallel Programming with Message Passing Interface," The MIT Press, Cambridge, MA., 1994

[11] Karypis, G. et al., "A software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices V.4.0," U. of Minnesota, 1998

[12] Bovendeerd, P.H.M., Steenhoven, A.A., Vosse, F.N. and Vossers, G. "Steady Entry Flow in a Curved Pipe," J. Fluid Mechanics, Vol. 177 (1987), pp. 233-246

[13] Prasad, A.K. and Koseff, J.R., "Reynolds Number and End-Wall Effects on a Lid-Driven Cavity Flow," Phy. Fluids, Vol.1, No.2, pp.208-218, 1988

[14] Coutanceau, M. and Bouard, R., "Experimental Determination of the Main Features of the Viscous Flow in the Wake of a Circular Cylinder in Uniform Translation. Part 1 Steady Flow," J. of Fluid Mechanics, Vol.79, No.2, pp.231-256