그리드 환경하에서 고성능 컴퓨팅을 이용한 열유동 해석 기법에 관한 기초연구

홍승도† · 이대성* · 이재룡* · 하만영* · 이상산**

A Fundamental Study of Thermal-Fluid Flow Analysis using High Performance Computing under the GRID

Seung Do Hong, Dae Sung Lee, Jae Ryong Lee, Man Yeong Ha and Sang San Lee

Key Words : Grid(그리드), CFD(전산유체역학), Cluster(클러스터), Parallel Computing(병렬 컴퓨팅), MPI(메시지 패싱 인터페이스), LES(대형 와 모사), DNS(직접수치계산)

Abstract

For simulation of three-dimensional turbulent flow with LES and DNS takes much time and expense with current available computing resources. It is nearly impossible to simulate turbulent flow with high Reynolds number. So, the emerging alternative is the Grid computing for needed computation power and working environment. In this study, the CFD code was parallelized to adapt it for the parallel computing under the Grid environment. In the first place, the Grid environment was built to connect the PC-Cluster facilities belong to the different institutions using communication network system. And CFD applications were calculated to check the performance of the parallel code developed for the Grid environment. Although it is a fundamental study, it brings about a important meaning as first step in research of the Grid.

기호설명

u : x 방향 속도
v : y 방향 속도
w : y 방향 속도
p : 압력
ρ : 밀도
ϕ : 보존량
Γ_ϕ : 확산계수
S_ϕ : 소스항
Re : 레이놀즈 넘버

† 부산대학교 기계공학부
E-mail : mpich@pusan.ac.kr
TEL : (051)510-3090 FAX : (051)512-9835
* 부산대학교 기계공학부
** 한국과학기술정보연구원 슈퍼컴퓨팅센터

1. 서 론

1.1 연구 배경

시대의 흐름에 따라, 컴퓨터의 성능은 급격히 증가해 왔으며 이를 통해 순수과학 및 응용공학 분야가 크게 발전하였다. 전산 유체역학 분야에 있어서도 1970 년대 초반의 컴퓨터로는 압축성 포텐셜 유동 방정식으로 3 차원 날개 정도를 해석할 수 있었으며, 난류 유동은 2 차원 익형 정도를 해석 할 수 있었다. 1980 년대 초반의 컴퓨터에서는 3 차원 항공기 전체의 포텐셜 유동과 3 차원 날개의 난류 유동을 해석할 수 있는 컴퓨터 기술까지 발전이 되었으며, 1990 년대의 컴퓨터에서는 3 차원 항공기 전체 형상 주위의 난류 유동을 시뮬레이션 할 수 있게까지 발달되어 왔다[1]. 특히, LES(Large Eddy Simulation) 및 DNS(Direct Numerical Simulation)방법을 사용하여

난류유동 해석시 가장 어려운 점은 작은 스케일의 와동 구조를 해석할 수 있어야 하므로 많은 수의 계산 격자가 필요하다. DNS 방법을 사용하여 3 차원 계산을 수행할 시 $Re^{9/4}$ 의 격자수를 필요로 하게 되는데 이는 계산시간이 많이 걸릴 뿐만 아니라, 대용량의 메모리와 저장 장치를 필요로 하게 된다.

현재 단일 CPU 의 성능은 한계점에 빠르게 도달하고 있기 때문에 이와 같은 컴퓨터 성능의 한계를 극복하기 위한 방법으로, 다수의 CPU 를 병렬로 연결하여 하나의 문제를 협동적으로 계산하는, 병렬컴퓨팅이 등장하게 되었고, 기존 몇몇 벤더에 의해 제공되던 병렬컴퓨터 혹은 슈퍼컴퓨터들은 매우 고가이기 때문에 일반인들은 쉽게 접할 수가 없었다. 따라서, 누구나 쉽게 병렬컴퓨팅을 실행할 수 있도록 전세계적으로 클러스터의 연구가 진행되어 왔다.

1.2 연구 목적

최근의 뛰어난 성능의 CPU 를 장착하고 있는 단일 컴퓨터들을 고속의 네트워크로 연결함으로써 새로운 개념의 병렬컴퓨터를 만드는 것이 가능하게 되었는데 이러한 개념의 병렬컴퓨터를 통칭하여 ‘Cluster(클러스터)’라고 부른다. 그러나, 지금까지 진행되어 왔던 클러스터의 개발은, 하나의 연구실 내에 있는 몇 개의 워크스테이션이나, 싼 가격에 구입이 가능한 컴퓨터를 한 장소에 모아 병렬 처리를 시도한 것이 대부분이었다.

이제는 광대역 통신망을 활용한 공유에 있어 동일 기종 컴퓨터뿐만 아니라 이기종 컴퓨팅 자원과 대용량 저장장치, 다양한 고성능 연구 장비가 포함되어 있는데 이러한 통합 환경을 ‘Grid(그리드)’라는 용어를 가지고 표현하고 있다. 그리드는 지리학적으로 분산되어 있는 고성능 컴퓨팅 자원을 네트워크로 상호 연동하여 조직과 지역에 관계없이 사용할 수 있는 환경을 말한다. 그리드 라는 단어는 1990 년대 중반 미국의 슈퍼컴퓨팅센터들을 중심으로 고성능의 분산 컴퓨팅 인프라를 구축하는 데서 비롯되었고, 고성능 자원, 대용량 정보 및 혁신적인 응용에 초점이 맞추어진 것이 일반적인 분산 컴퓨팅과 구별된다[(2)(3)(4)].

본 연구에서는 최근에 많은 연구가 진행되고 있는 그리드 환경을 위한 테스트 베드를 직접 구축하고 그리드 환경하에서 고성능 컴퓨팅을 이용한 열유동 해석을 수행하고자 한다.

Table 1 Specification of System

시스템 형태		PC-Cluster
구성 노드 수		24
단위 노드 사양	프로세서	P4 2.0Mhz
	메모리	DDR 512MB
	디스크	60GB
	OS	Redhat Linux 7.3
네트워크		FastEthernet

2. 연구 내용

2.1 테스트 베드 구축

본 연구에서는 그리드 환경을 구축하기 위한 테스트 베드를 직접 제작하였다. 시스템의 사양은 Table 1 과 같다. 그리드 서비스를 제공하는 그리드 미들웨어로 Globus 2.0 을 설치하였으며, 병렬 컴퓨팅을 위한 MPICH-G2 와 Jobmanager 로써 OpenPBS 를 설치하였다[(5)(6)(7)].

2.2 그리드 환경 구축

그리드 환경 구축을 위해서는 실제로 지역적으로 분산되어 있는 시스템들이 필요하다. 이에 본 연구에서는 자체적으로 구축한 시스템과 한국과학기술정보연구원(KISTI)에서 구축한 테스트베드를 선도시험망으로 연결하여 그리드 환경을 구축하였다. 먼저, 응용프로그램의 병렬 성능을 측정하기 위하여 KISTI 에서 보유하고 있는 165GFlops 의 성능을 가지는 128 노드 PC-Cluster(Pluto)를 이용해 테스트 해보았다. 또한, 그리드 환경하에의 응용프로그램 적합성을 판단하고자 Fig.1 과 같이 서로 다른 3 가지 환경을 구성한 뒤 응용프로그램을 테스트 하였다.

LAN(Without Globus)는 부산대 공동실험실습관에 구축되어 있는 리눅스 클러스터 시스템을 의미한다. 이는 일반 병렬 컴퓨팅을 위한 것으로 그리드 미들웨어인 Globus 가 설치되어 있지 않다. 단일 노드의 CPU 는 P3-550 으로 성능이 좋지 않지만 네트워크는 미리넷으로 연결되어 있어 노드간 통신성능은 대단히 좋다.

LAN(With Globus)는 KISTI 테스트 베드를 의미한다. 단일 노드의 CPU 는 P4-2.0 으로 성능이 좋은 반면 네트워크는 FastEthernet 으로 연결되어 있어 노드간 통신 성능은 좋은 결과를 얻기 힘들다.

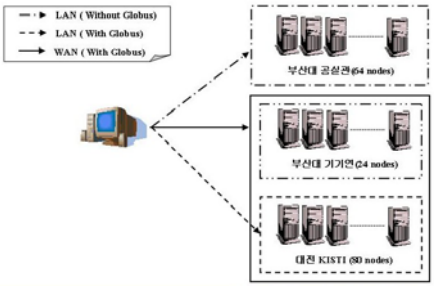

Fig. 1 Different environments for Grid computing

WAN(With Globus)는 부산대와 KISTI 사이에 놓여 있는 선도시험망을 이용해 두 기관의 테스트 베드를 연결하여 마치 하나의 클러스터처럼 사용하는 것으로 소규모의 그리드 환경을 의미한다.

2.3 계산 그리드

2.3.1 Cavity Flow

본 연구의 첫번째 응용프로그램으로써 Fig.2 와 같은 2 차원 Unsteady driven cavity 내의 열유동을 해석하는 프로그램을 사용하였다[8]. Re=400 이며, 계산 격자수는 40,000~70,000 개 정도로 비교적 작은 사이즈의 문제이다. 또한 통신량 대 한 노드가 담당하는 문제의 사이즈를 가리키는 Surface to volume ratio 의 변화에 따른 두가지 Case 로 나누어 테스트 하였다. Case1 은 전체 계산 격자수를 고정시켜 계산에 사용된 노드수가 증가할수록 한 노드가 담당하는 격자수는 줄어드는 경우이며, Case2 는 한 노드가 담당하는 격자수를 고정시켜 계산에 사용된 노드수가 증가할수록 전체 격자수는 증가하는 경우이다.

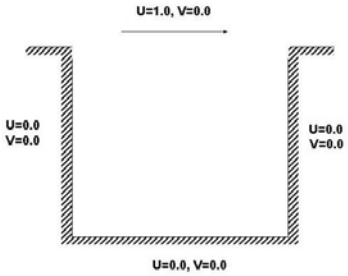

Fig. 2 Driven cavity

2.3.2 Cylinder with circular fin

두 번째 응용프로그램은, Fig.3 와 같이 원형 핀을 가지는 3 차원 실린더 주위의 열유동을 해석하는 프로그램이다. Re=300 이며, 계산 격자수는 약 70 만개로 비교적 큰 사이즈의 문제이며 사용된 수치해석 기법은 Spectral, Multidomain method 를 사용한 DNS(Direct Numerical Simulation)로, 많은 계산 시간을 요한다. 본 연구에서는 Case3 에 해당된다.

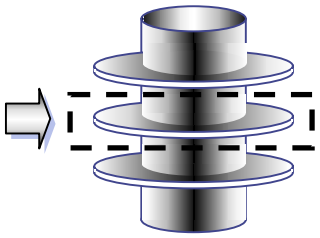

Fig. 3 Cylinder with circular fin

2.3.3 Tube banks

마지막으로 Fig.4 와 같이 3 차원 관군주위에 대한 열유동을 해석하는 프로그램을 이용하였다[9]. Re=4,000 이며, 계산 격자수는 약 200 만개로 매우 큰 사이즈의 문제라 할 수 있다. 사용된 수치해석 기법은 LES(Large Eddy Simulation)로 기초방정식에 공간적인 평균화 조작을 행하고, 유동장을 격자로 해석할 수 있는 성분과 그 이하의 작은 성분으로 분리하여 전자는 직접 계산하고 후자는 모델화화를 하는 해석 방법이다. 본 연구에서는 Case4 에 해당된다.

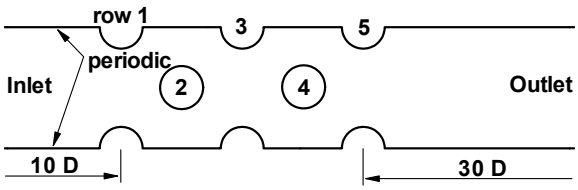

Fig. 4 Tube banks

3. 이론적 연구

3.1 지배 방정식

3.1.1 Navier-Stokes 방정식

열유동 해석을 위한 3 차원 비정상 Navier-Stokes 방정식을 Conservative form 으로 표현하면 다음과 같다.

- 연속 방정식

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \qquad (1)$$

- X 운동량 방정식

$$\frac{\partial u}{\partial t}+\frac{\partial (uu)}{\partial x}+\frac{\partial (vu)}{\partial y}+\frac{\partial (wu)}{\partial z}=-\frac{1}{\rho}\frac{\partial P}{\partial x}+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right) \qquad (2)$$

- Y 운동량 방정식

$$\frac{\partial v}{\partial t}+\frac{\partial (uv)}{\partial x}+\frac{\partial (vv)}{\partial y}+\frac{\partial (wv)}{\partial z}=-\frac{1}{\rho}\frac{\partial P}{\partial y}+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right) \quad (3)$$

- Z 운동량 방정식

$$\frac{\partial w}{\partial t}+\frac{\partial (uw)}{\partial x}+\frac{\partial (vw)}{\partial y}+\frac{\partial (ww)}{\partial z}=-\frac{1}{\rho}\frac{\partial P}{\partial z}+\nu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right) \quad (4)$$

이상의 방정식을 일반화 변수 ϕ를 사용하여 나타내면 다음과 같다.

$$\frac{\partial}{\partial t}(\rho\phi)+\frac{\partial}{\partial x}(\rho u\phi)+\frac{\partial}{\partial y}(\rho v\phi)+\frac{\partial}{\partial z}(\rho w\phi) -\frac{\partial}{\partial x}\left(\Gamma_\phi\frac{\partial\phi}{\partial x}\right)-\frac{\partial}{\partial y}\left(\Gamma_\phi\frac{\partial\phi}{\partial y}\right)-\frac{\partial}{\partial z}\left(\Gamma_\phi\frac{\partial\phi}{\partial z}\right)=S_\phi \quad (5)$$

식 (5)에 사용된 변수 ϕ는 보존량을 나타내며 Γ_ϕ, S_ϕ는 각각 ϕ의 확산계수와 소스항을 가리킨다.

3.2 병렬화 기법

MPI(Massage Passing Interface)는 병렬컴퓨팅의 기법 중 메시지 패싱에 기본을 둔 프로그래밍 방법 중의 하나이다. 메시지 패싱은 이미 여러 가지의 병렬 컴퓨팅에 다년간의 시험을 거친 프로그래밍 기법이지만, MPI 기준이 나오기 이전까지는 모든 병렬 컴퓨팅 중 각기 다른 형태의 프로그래밍 인터페이스를 제공하여 사용자 프로그램의 호환성에 문제가 있었다. 이러한 문제점을 인식하고 일정한 문법과 공통 사항을 마련하고자 Message Passing Interface Forum을 결성하고 수년간 노력한 끝에 1944년에 제정한 것이 MPI(v1.0)이다[10].

본 연구에서는 모든 Case의 병렬화는, 프로그램의 확장성과 우수한 이식성 때문에 유동장의 계산 영역을 분할하는, 영역 분할법(DDT : Domain Decomposition Technique)을 사용하였다. 또한 영역을 소 영역으로 분할함으로써 복잡한 형상을 가진 영역에 대한 계산이 용이한 점도 있다. 또한, Virtual topology 방법을 사용함으로써 병렬 계산에 참여하는 프로세서들을 2차원 혹은 3차원 배열로 가상 배치함으로써 각각의 프로세서들은 고유의 독립된 좌표계를 가지게 된다.

4. 결과 및 고찰

4.1 병렬 성능

Fig.5는 본 연구에 쓰인 병렬 기법의 성능을 측정하기 위하여 KISTI에서 보유하고 있는 Pluto 시스템[11]을 이용하여 Case1을 테스트 한 결과이다. 계산에 사용된 노드수가 증가할수록 전체 실행시간은 감소함을 볼 수 있다. 이론상으론 실행시간이 비례적으로 감소해야 하나 사용된 프로그램의 완벽한 병렬화를 기대하기 힘들고, 비교적 간단한 코드이기 때문에 비례적이지 못하다.

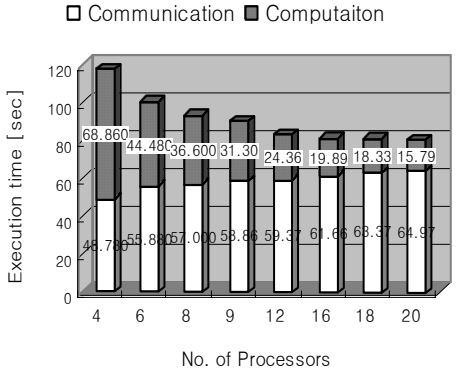

Fig. 5 Variation of run time at Pluto System

4.2 Case1 의 실행 결과

Fig.6는 계산에 사용한 노드수가 증가함에 따른 노드간 통신시간만을 비교한 결과이다. 네트워크 속도가 비교적 빠른 미리넷으로 연결되어 있는 LAN(Without Globus)환경이나 패스트 이더넷으로 연결되어 있는 LAN(With Globus)에 비하여 WAN(With Globus)환경 즉, 그리드 환경에서는 노드간 통신시간이 약 3~10 배정도 더 느림을 볼 수 있다. 이는 그리드 환경에서 응용프로그램의 실행시 보다 빠른 네트워크가 요구됨을 수치적으로 보여 주고 있다.

Fig.7는 계산에 사용한 노드수가 증가함에 따른 계산시간만을 비교한 결과이다. 세가지 환경모두 노드수가 증가함에 따라 계산시간이 감소함을 볼 수 있으며 환경에는 상관없이 단일 CPU의 성능이 결과를 결정지음을 알 수 있다.

Fig.8는 계산에 사용한 노드수가 증가함에 따른 전체 계산시간 즉, 통신시간과 계산시간의 합을 보여주고 있다. 미리넷을 사용하는 LAN(Without Globus)환경에서는 좋은 병렬 성능을 보이고 있으나, 패스트 이더넷을 사용하는 LAN(With Globus) 환경에서는 병렬 성능이 좋지 못하다. 이는 네트워크의 성능저하에 원인이 있으며 또한, 테스트한 프로그램이 비교적 작은 사이즈의 문제이기 때문에 발생되는 현상이다. LAN 환경에 비하여 그리드 환경에서는 약 2~5 배정도 느림을 볼 수 있다.

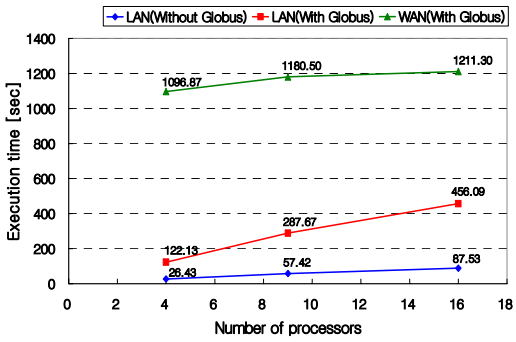

Fig. 6 Variation of communication time in Case1

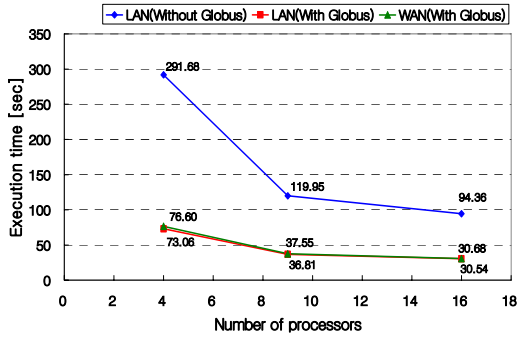

Fig. 7 Variation of computation time in Case1

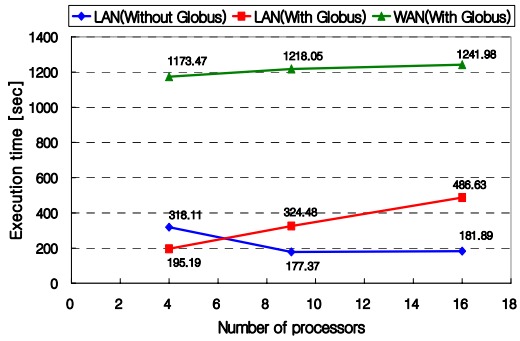

Fig. 8 Variation of total run time in Case1

4.3 Case2 의 실행 결과

Case2 의 경우 단일 노드가 담당하는 영역이 일정하므로 계산에 사용한 노드수가 증가하더라도 이론상으론 동일한 계산시간이 나와야 한다.

Fig.9 는 계산에 사용한 노드수가 증가함에 따른 노드간 통신시간만을 비교한 결과이다. 노드수가 증가할수록 통신시간이 점차 증가하는 경향을 보이고 있는데, 이는 테스트한 프로그램의 계산 격자수가 비교적 적은 경우라서 병렬효율이 좋지 못하고, 노드간 통신시 불필요한 작업들이 계산이나 통신에 영향을 미치기 때문이다. 하지만, 본 연구에서는 LAN 환경과 그리드 환경과의 결과를 비교하는 데에 목적이 있으므로, 그 차이만을 보고자 한다. 앞에서와 마찬가지로, 그리드 환경에서는 노드간 통신하는데 걸리는 시간이 약 3~10 배정도 느림을 볼 수 있다.

Fig.10 는 계산에 사용한 노드수가 증가함에 따른 계산시간만을 비교한 결과이다. Fig.9 에서와 마찬가지로 노드수가 증가할수록 계산시간이 조금씩 증가함을 볼 수 있으며, 환경에 상관없이 CPU 의 성능차이로 인한 결과를 보여주고 있다.

Fig.11 를 살펴보면, Case1 과 마찬가지로 노드수가 증가함에 따라 전체 계산시간이 조금씩 증가하는 현상을 볼 수 있으며, 그리드 환경에서의 전체 계산 시간은 약 2~7 배정도 느림을 볼 수 있다.

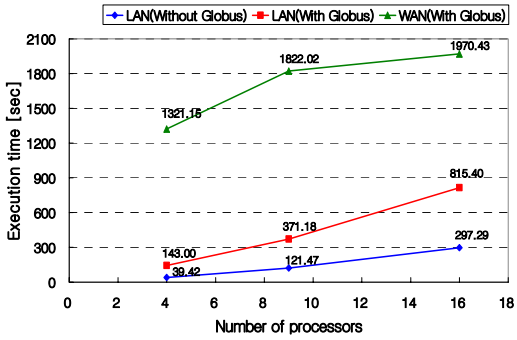

Fig. 9 Variation of communication time in Case2

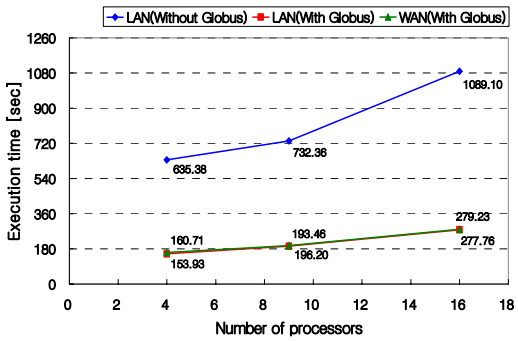

Fig. 10 Variation of computation time in Case2

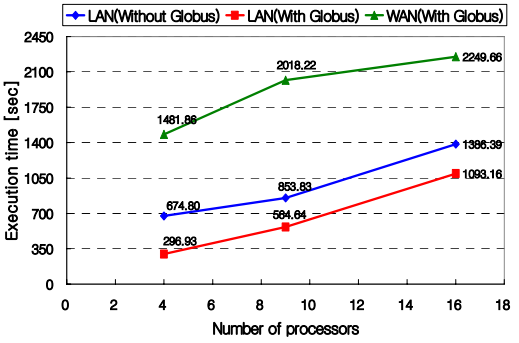

Fig. 11 Variation of total run time in Case2

4.4 Case3 의 실행 결과

Case3 는 프로그램의 복잡성 등으로 인해 계산에 사용된 노드수는 12 개로 고정되어 있다. Fig.12 를 살펴보면 계산시간은 환경에 상관없이 단일 CPU 의 성능 차이가 결과로 나타나고 있으며, 통신시간은 LAN 환경에 비해 네트워크의 성능이 저하되는 그리드 환경에서 약 8~25 배 정도 느림을 볼수 있다.

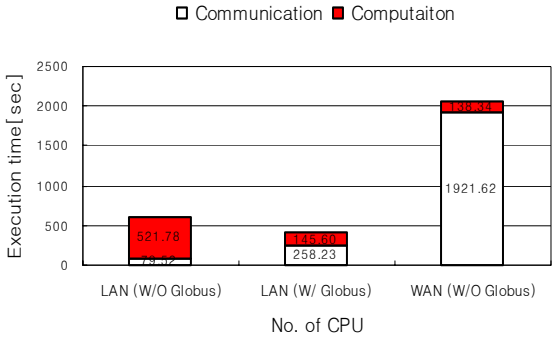

Fig. 12 Variation of total run time in Case3

4.5 Case4 의 실행 결과

Case4 는 프로그램의 복잡성 등으로 인해 계산에 사용된 노드수는 24 개로 고정되어 있다. Fig.13 를 살펴보면 Case3 와 마찬가지로, 계산시간은 환경에 상관없이 단일 CPU 의 성능의 차이가 결과로 나타나고 있으며, 통신시간은 LAN 환경에 비해 그리드 환경에서 약 30 배 정도 느림을 볼 수 있다.

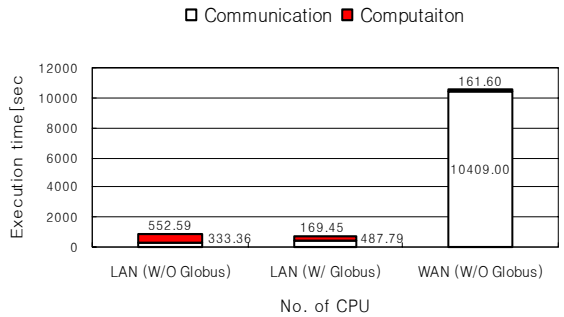

Fig. 13 Variation of total run time in Case4

5. 결 론

본 연구에서는 국내외적으로 연구가 활발히 진행되고 있는 그리드 컴퓨팅과 관련하여, 소규모의 그리드 환경을 직접 구축하였고, 그리드 환경하에서 열유동 해석 프로그램의 계산을 성공적으로 수행하였다. LAN 환경에 비하여 네트워크의 성능이 현저히 저하되는 그리드 환경에서의 실행결과가 약 2~10 배 정도 느리게 나왔다. 이는 그리드 환경에서 그만큼 네트워크의 성능이 좋아야 함을 수치적으로 대변해 주고 있으며, 그리드 컴퓨팅을 연구하는 첫 단계로써 매우 중요한 의미를 부여해 주고 있다.

후 기

본 연구는 BK21, NRL, N*Grid 사업에 의해 수행되었으며, KISTI 슈퍼컴퓨팅 센터의 Pluto 시스템 사용의 지원에 감사드립니다.

참고문헌

(1) Warn-Gyu Park, 1998, “Lecture Note on the Computational Fluid Dynamics”, Vol. I, pp. 5~7.
(2) K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman, 2001, “Grid Information Services for Distributed Resource Sharing”
(3) Kum Won Cho, Woo Hyung Park and Sang San Lee, 2002.02, “Information Communication Technology and Computational Fluid Dynamics”, Korean Society of Computational Fluids Engineering, Vol.6, No.3
(4) Kyeun Dong Kim, Woo Pill Lee and Il Sun Hwang, 2002.02, “Grid Computing”, Korea Information Science Society, Vol.20, No.02, pp. 5~19.
(5) The Globus Project, http://www.globus.org
(6) MPICH-G2, Grid enabled implementation Message Passing Interface(MPI), http://www3.niu.edu/mpi
(7) Portable Batch System, http://www.openpbs.org
(8) U. Ghia, K.N. Ghia and C.T. Shin, 1982, “High-Re solution for incompressible flow using the Navier-Stokes equation and multigrid method”, J. of Computational Physics, Vol. 48, pp.387.
(9) S.H.Kim, M.Y.Ha and S.S. Lee, 2002, “A Large eddy simulation for the three-dimensional fluid flow past tube banks,”, 5th KSME-JSME Fluid Engineering Conference, Nagoya, Japan.
(10) The Message Passing Interface(MPI) standard, http://www-unix.mcs.anl.gov/mpi/
(11) PC Cluster, 128nodes, http://www.hpcnet.ne.kr