Atmosphere. Korean Meteorological Society
Vol. 22, No. 1 (2012) pp. 109-115

기상 모델 CFD_NIMR의 최적 성능을 위한 혼합형 병렬 프로그램 구현

김민욱[1] · 최영진[1] · 김영태[2,*]
[1]국립기상연구소 응용기상연구과
[2]강릉원주대학교 컴퓨터공학과

(접수: 2011년 10월 31일, 수정: 2011년 12월 5일, 게재확정일: 2011년 12월 21일)

Hybrid Parallelization for High Performance of CFD_NIMR Model

Min-Wook Kim[1], Young-Jean Choi[1], and Young-Tae Kim[2,*]
[1]Applied Meteorology Research Division, National Institute of Meteorological Research
[2]Department of Computer Science & Engineering, Gangneung-Wonju National University

(Received: 31 October 2011, Revised: 5 December 2011, Accepted: 21 December 2011)

Abstract We parallelized the CFD_NIMR model, which is a numerical meteorological model, for best performance on both of distributed and shared memory parallel computers. This hybrid parallelization uses MPI (Message Passing Interface) to apply horizontal 2-dimensional sub-domain out of the 3-dimensional computing domain for distributed memory system, as well as uses OpenMP (Open Multi-Processing) to apply vertical 1-dimensional sub-domain for utilizing advantage of shared memory structure. We validated the parallel model with the original sequential model, and the parallel CFD_NIMR model shows efficient speedup on the distributed and shared memory system.

Keywords: CFD_NIMR, MPI, OpenMP, hybrid parallelization

1. 서 론

일반적으로 기상을 모의하는 모델은 격자 구조의 3차원의 도메인을 구성하여 수치적으로 미분방정식을 계산하는 수치 모델로서 방대한 계산을 필요로 하기 때문에 주어진 시간 내에 모의 계산을 하기 위해서는 초고속 계산이 필수적이다. 이러한 초고속 계산을 필요로 하는 수치모델의 계산에 있어서 가장 효율적인 계산 방법은 여러 프로세서를 동시에 사용하여 계산하는 병렬처리 방식이며, 병렬처리를 기반으로 하는 슈퍼컴퓨터는 하드웨어 기술의 발달과 더불어 비약적으로 발달하고 있다. 슈퍼컴퓨터의 성능은 프로세서 기술의 획기적인 발달로 적어도 10년 이후에는 초당 10^{18}번의 연산이 가능한 ExaFlops급의 슈퍼컴퓨터의 개발이 예상되고 있다 (Top500 supercomputer sites, 2009). 이러한 계산 환경에서 효율적으로 실행하기 위하여 해당 프로그램의 병렬프로그램의 구현은 필수적이다.

병렬프로그램은 크게 분산메모리형의 계산 환경에서 실행하기 위하여 메시지 전송 라이브러리를 사용하는 프로그램과 단일 노드 내에서 여러 프로세서들이 메모리를 공유하는 공유메모리 병렬프로그램으로 나누어 진다. 이 중 메시지 전송 방식은 공유메모리형의 병렬컴퓨터에서도 실행이 가능하기 때문에 메시지 전송 방식을 사용하면 어떤 방식의 병렬컴퓨터에서 실행할 수 있는 장점이 있기 때문에 많은 프로그램이 메시지 전송 방식을 사용하고 있다. 한편 메시지 전송 방식의 프로그램은 프로세서간의 데이터 전송이 병렬프로그램의 주요 부하이기 때문에 데이터 전송이 많은 프로그램에서는 효율적으로 실행이 되지 못하는 단점이 있다. 또한 최근에 들어서는 프로세서에서 공유메모리형의 다중 코어를 사용하기 때문에 과거와는 달리 데이터의 전송이 필요하지 않은 공유

*Corresponding Author: Youngtae Kim, Department of Computer Science & Engineering, Gangneung-Wonju National University, Wonju 220-711, Korea
Phone: +82-33-760-8667, Fax: +82-33-760-8661
E-mail: ykim@gwnu.ac.kr

메모리형의 프로그램이 효율적인 프로그램으로 인식되고 있다.

본 연구에서는 도시 지형을 모의할 수 있는 수치모델인 전산유체역학 (CFD_NIMR, Computational Fluid Dynamics) 모델 프로그램을 병렬화하였다. 병렬 CFD_NIMR 프로그램은 3차원 데이터 도메인을 기반으로 계산하는데 수평 2차원은 분산메모리 병렬컴퓨터에서 실행될 수 있는 메시지 전송 방식인 MPI (Message Passing Interface)를 사용하여 병렬화하였다 (Gropp *et al.*, 1999; Pacheco, 1996). CFD_NIMR 프로그램의 특성상 프로세서간의 통신이 빈번하게 발생하여 전체적인 성능에 영향을 주고 있는데 이러한 통신 부하를 줄이기 위하여 수직 1차원은 공유메모리형의 병렬 방식인 OpenMP (Open Multi-Processing)를 사용하여 병렬화하여 다중코어 프로세서와 메모리를 효율적으로 활용할 수 있도록 하였다 (Chandra *et al.*, 2000; Michalakes *et al.*, 2004). 이러한 MPI와 OpenMP의 혼합형으로 구현된 병렬 CFD_NIMR 프로그램은 분산 및 공유 메모리(다중코어) 형태의 병렬 계산 환경에서 프로세서를 효율적으로 사용하여 초고속으로 모의 계산을 할 수 있다.

대부분의 병렬 기상 모델의 경우에는 주로 MPI만 사용하고 있으며, 대표적인 community 기상 모델인 WRF (Weather Research and Forecast)의 경우 (Michalakes *et al.*, 2004) 병렬프로그램에서 MPI 방식과 OpenMP 방식을 모두 사용한다. Roe and Stevens (2010)는 고해상도의 수치예보 시스템 구축을 위하여 WRF를 사용하여 하와이제도 지역을 대상으로 병렬 시스템의 효율을 확인하였다. WRF의 OpenMP는 수직 방향의 병렬 계산이 아닌 각 노드에서 공유메모리 내에서 프로세서들을 수평 방향으로 계산하기 위한 것이므로 본 연구에서의 혼합 병렬형과는 다른 방식으로 구현되었다고 볼 수 있다.

본 논문의 2장에서는 병렬 CFD_NMR 프로그램의 구조와 병렬화 방식에 대하여 상세히 설명한다. 3장에서는 병렬프로그램을 병렬 컴퓨터 환경에서 성능을 검증한다. 그리고 4장에서는 병렬프로그램의 활용을 설명하고, 5장에서는 결론을 제시한다.

2. 병렬프로그램의 구현

이 장에서는 CFD_NIMR 프로그램의 구조와 MPI와 OpenMP의 병렬화에 대하여 설명한다.

2.1. CFD_NIMR 모델 프로그램의 구조와 격자 구조

CFD_NIMR 모델은 2005년에 국립기상연구소와 서울대학교가 유체역학모델을 통해 국지 기상을 수치모의에 적용하기 위한 목적으로 공동 개발한 수치 모델이다 (국립기상연구소, 2006). 유체의 흐름을 나타내는 역학 방정식은 이차 미분 방정식으로 구성되어 있어 방정식의 해를 구하기 위해서 수치적인 계산 방법이 사용되며 본 모델에서는 유한체적법 (finite volume method)을 사용한다. 수치적 해를 구하기 위해 방정식의 차분화는 Patankar (1980)가 제안한 SIMPLE (Semi-Implicit Method for Pressure-Linked Equation) 알고리즘을 이용한다. 모델의 계산 순서는 우선 각 변수에 대한 이차 미분방정식의 각 항에 대한 값을 계산한다. 이후 수치적 적분 방법을 활용하여 이차 미분방정식의 해를 계산한다. 계산된 값은 수렴 판정을 거쳐서 재계산되거나 다음 시간 단계로 넘어간다. 수치적 해를 계산하는 과정에서 SOR (Successive Over Relaxation) 방식을 사용한다. 유한체적법을 사용하기 때문에 변수는 정해진 위치에서 계산되며 계산 과정을 고려하여 엇갈림 격자 구조를 사용한다. 격자는 일반적으로 사용되는 x, y, z의 직교 격자계로 구성되었다. 변수는 변수 주변의 다른 변수와 인접 격자의 동일 변수 값을 통해서 계산된다. 이 때문에 변수를 계산하기 위해서는 인접한 격자 간의 변수 정보가 필요하다.

2.1.1. 엇갈림 격자 구조 (staggered grid)

CFD_NIMR 모델은 C-grid의 엇갈림 격자 구조로 구성되어 있다. Fig. 1은 CFD_NIMR 모델의 변수가 계산되는 격자의 위치를 나타낸다. 격자의 중심에는 온도, 압력, 물질의 농도 등 유체의 운동을 나타내는 변수 이외의 변수가 계산된다. 유체의 운동을 나타내는 u, v, w 변수는 각 운동의 방향에 따라 격자의 면 중심에서 계산된다. 예로 u 변수는 격자의 x 방향 면의 중심에서 계산된다. 엇갈림 격자 구조는 역학 방

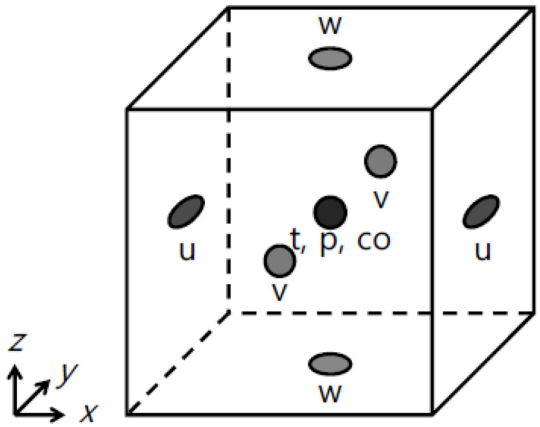

Fig. 1. Grid system of CFD_NIMR model and positions of each variable on grid.

정식의 형태와 연결되어 각 변수간 계산 과정을 간략하게 한다.

2.2. MPI를 사용한 병렬화

CFD_NIMR 모델은 분산메모리형의 병렬 계산 환경에서 실행하기 위하여 메시지 전송을 제공하는 표준 라이브러리인 MPI를 이용하여 병렬화하였다. 일반적으로 MPI는 메시지 전송을 위한 기본적인 함수들만 제공하기 때문에 함수들을 병렬프로그램에서 직접 호출하기에는 어려움이 많다 (Pacheco, 1996). 따라서 MPI함수들을 효율적으로 사용하기 위하여 중간 인터페이스의 형태의 라이브러리를 사용하는 경우가 많은데 WRF에서 사용하는 RSL이 대표적인 경우이다 (Michalakes, 2000). 병렬 CFD_NIMR 프로그램은 상위 라이브러리인 객체지향형 HPCC (High Performance Computing Class)를 사용하였으며 HPCC를 통하여 각 프로세서로의 데이터 분산, 인덱스의 전환 및 메시지 전송을 위한 다양한 방식들이 구현되었다 (Kim, 2011).

2.2.1. 병렬라이브러리 HPCC

HPCC는 Fortran으로 구현된 라이브러리로서 Fig. 2와 같이 병렬프로그램과 MPI의 중간에 위치하여 병렬프로그램에서 MPI를 직접 호출하지 않기 때문에 쉽고 효율적으로 사용할 수 있으며, 파일 입출력, 도메인 분할, 통신 데이터 정의 및 송수신, 그리고 기타 병렬프로그램에서 필요한 다양한 함수들 등 병렬화에 필요한 다양한 기능을 함수의 형태로 제공하기 때문에 병렬 프로그램의 구현 및 프로그램의 유지 및 보수를 쉽게 한다 (Kim, 2011).

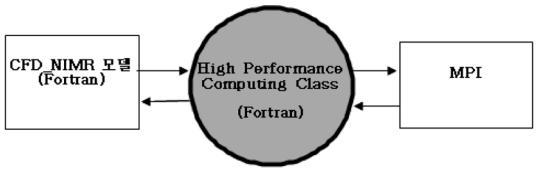

Fig. 2. Function of HPCC.

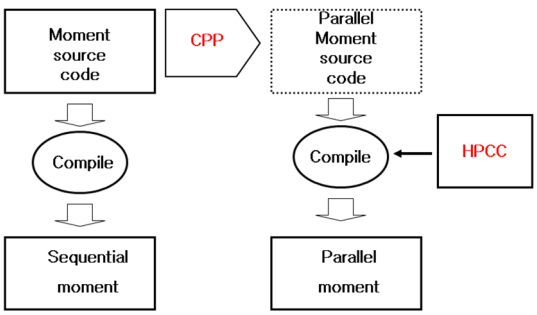

Fig. 3. Compile system of parallel CFD_NIMR model.

병렬프로그램의 컴파일 과정은 Fig. 3과 같다. CFD_NIMR의 코드는 순차 코드와 병렬 코드가 하나의 코드로서 존재하며 옵션을 통하여 구분하여 컴파일을 할 수 있다. 병렬 옵션을 사용하면 전처리 프로그램인 CPP를 통하여 매크로를 적용한 가상 상태의 병렬프로그램을 만들고 이를 HPCC를 통하여 MPI를 사용하는 병렬프로그램으로 구현된다.

2.2.2. 메시지 전송을 사용한 병렬프로그램의 구현

HPCC를 사용하기 위해서는 최초에 병렬프로그램 및 시스템의 정보를 저장하는 객체를 수평의 2차원 도메인을 사용하여 정의해 주어야 한다. 코드 (1)에서 *process*는 병렬프로그램의 기본 정보를 저장하는 객체이며 이는 *hpcc_define_parallel*을 통하여 초기화된다. 다음에서 초기값으로 제공되는 *imax2*와 *jmax2*는 모델에서 사용하는 2차원 도메인의 크기를 나타낸다.

$$\$parallel\_only\ process = hpcc\_define\ parallel(imax,\ jmax2) \quad (1)$$

병렬프로그램은 분할된 서브도메인을 동시에 계산함으로써 계산 시간을 단축시킨다. 서브도메인은 계산하기 위한 인덱스의 크기에 따라 달라지는데 인덱스의 전환을 위한 매크로는 HPCC에서 제공이 된다. 코드 (2)는 인덱스의 전환을 위해 매크로를 사용하여 전환한 예로서 프로그램 내에서 모든 반복문은 이와 같이 전환이 되며 병렬프로그램에서는 자동으로 각 프로세서에서 사용하는 지역인덱스로 전환이 된다.

$$\begin{array}{ll} DO\ I = 1,imax2 & \$parallel\_do(process, i, 2, jmax2) \\ DO\ J = 1,imax2 & \$parallel\_do(process, i, 2, jmax2) \\ \cdots & \Rightarrow \quad \cdots \\ ENDDO & \$parallel\_enddo \\ ENDDO & \$parallel\_enddo \end{array} \quad (2)$$

분산메모리 환경에서 실행되는 병렬프로그램은 순차프로그램과는 달리 프로세서간의 데이터의 송수신이 필요하다. 이는 분산된 도메인의 데이터 의존도 때문에 발생한다. Fig. 4는 통신이 필요한 경우의 예이다.

위에서 경계부분의 데이터는 분산메모리 환경의 다른 프로세서에 상주하기 때문에 $B(i, j, k)$를 계산하기 위해서는 계산하기 전에 해당 프로세서들로부터 데이터를 전달 받아야 한다. 이 데이터는 프로그램의 분석을 통하여 프로그램의 초반부에 통신 템플릿의 형태로 정의한다. 템플릿 (3)은 위 경우의 통신을 위한 변수와 그 형태이다.

$$A:\ HPCC\_IP + HPCC\_JP + HPCC\_IPJP \quad (3)$$

HPCC에서 제공하는 상수인 *HPCC_IP*는 i+1,

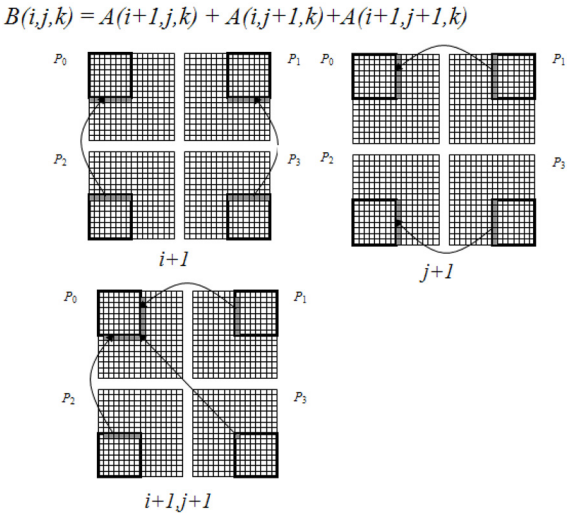

Fig. 4. Communication of data between each process.

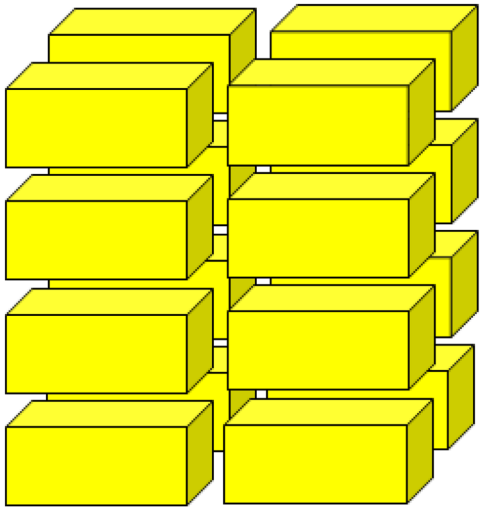

Fig. 5. Partitioning of Domain of parallel CFD_NIMR model.

*HPCC_JP*는 *j*+1 등을 나타내며 각 템플릿은 코드 (4)와 같이 병렬코드에 정의된다. 따라서 변수 *A*는 (*i*+1), (*j*+1), (*i*+1, *j*+1) 방향으로 통신이 필요하다는 것을 정의하며 다음과 같이 HPCC의 함수를 통하여 *comm_A*의 이름으로 정의한다.

```
call hpcc_add_comm(comm_A,A,HPCC_IP
    + HPCC_JP + HPCC_IPJP)                    (4)
```

위와 같이 정의된 통신 템플릿인 *comm_A*는 다음 코드 (5)에 의하여 실행이 된다. 이 코드에 의하여 해당 변수들 정의된 방향으로의 통신이 실행된다.

```
call hpcc_exec_comm(comm_A)                   (5)
```

이 때 *comm_A*에는 하나 이상의 변수가 정의될 수 있기 때문에 같은 방향으로 정의된 변수들은 동시에 같은 버퍼를 사용하여 통신이 되기 때문에 통신 시간을 단축할 수 있게 된다. 본 연구에서는 이와 같은 변수의 통신 패턴은 전체 코드의 수작업을 통하여 정의되었다.

2.3. 공유메모리 활용을 위한 OpenMp 병렬화

2.3.1. 도메인 분활

OpenMP를 사용한 병렬 프로그램은 메모리를 공유하는 쓰레드 (프로세서)들이 운영체제에 의하여 할당된 도메인의 각 영역을 동시에 계산하는 방식이다. OpenMP는 1개의 반복문만 병렬화하기 때문에 1차원의 분할만이 가능하다. 앞에서 MPI를 사용하는 분산 병렬프로그램은 수평 방향으로 2차원의 도메인 분할을 사용하였다. 따라서 MPI와 OpenMP를 동시에 사용하는 프로그램은 Fig. 5와 같이 수평 및 수직 방향으로 3차원의 분할을 사용하게 된다.

2.3.2. OpenMP 프로그래밍

OpenMP 프로그램은 프로그램 내에서 병렬화가 필요한 반복문 (do-loop)에 병렬 코드를 나타내는 !$omp로 시작하는 지시문을 삽입하여 이루어지며 이는 컴파일러에 의하여 자동으로 병렬화가 이루어진다. 코드 (6)은 OpenMP를 사용한 병렬 코드의 예로서 CFD_NIMR의 전체 프로그램 중 반복문에 이러한 방식의 지시문을 삽입하였다.

```
#ifdef OMP
!$omp parallel do default(shared) private(i, j, k)
!$omp + private(dlzm, dlzp, dlym, dlyp)
…                                             (6)
#endif
        do k = 2, kmax1
        $parallel_do(process,j,2,jmax1)
        $parallel_do(process,i,2,imax1)
```

위 코드에서 프로그램은 인덱스 *k*를 사용하는 반복문을 분할하여 병렬로 계산하기 때문에 연직 방향 (*k*-방향)으로 분할되어 계산이 되는 형태이다. 지시문에서 shared는 쓰레드들이 공유하는 변수이며, private로 정의되는 변수들은 각 쓰레드들이 반복문 내에서 독립된 값을 유지하기 위하여 지역 변수로 복사하는 변수들을 나타내며, 본 연구에서는 private 변수들을 원시 프로그램에서 수작업으로 분류하였다. 참고로 위 코드에서 $parallel_do로 나타나는 코드는 MPI를 사용한 수평 분할의 코드이다.

3. 병렬프로그램의 성능 분석

본 연구에서 사용된 병렬컴퓨터는 Intel XEON 2.0 GHz CPU를 사용한 2대의 노드로 구성된 분산메모리형의 계산 환경이며 각 노드는 8개의 다중 코어를 가지고 있다. 성능 분석을 위하여 $100 \times 100 \times 60$의 크기를 가진 도메인의 계산을 다른 수의 프로세서를 사용하여 병렬 프로그램의 실행시간을 비교하였다. 실행 시간은 초기의 100번 반복 시간의 평균을 초로 나타내었다.

3.1. 병렬프로그램간의 결과 차이

Fig. 6은 CFD_NIMR의 결과의 일부로써 시간에 따른 u와 v 값의 변화를 나타낸 것이다. 두 변수에서 확인할 수 있듯이 병렬화에 상관없이 CFD_NIMR의 결과는 동일하게 구해졌다. 그림에서 포함되지 않은 OpenMP와 MPI를 각각 사용한 결과도 동일한 값을 보였다. CFD_NIMR의 결과는 병렬화에 따라 영향을 받지 않는 것으로 확인되었다.

3.2. 분산메모리에서의 성능

Fig. 7은 병렬 CFD_NIMR 프로그램의 MPI를 사용한 속도 개선을 병렬 컴퓨터가 보일 수 있는 이상적인 속도개선인 선형의 값과 비교하여 보여준다. 병렬프로그램은 프로세서의 1차원 배열을 사용한 경우가 2차원 배열을 사용한 경우보다 계산 속도가 빠른 것을 보여 주는데 이는 프로세서가 1차원으로 배열한 경우에 프로세서간의 통신 부하를 줄일 수 있기 때문이다. 하지만 1차원의 분할을 사용하면 많은 수의 프로세서를 사용할 경우에는 제약이 있을 수 있다. MPI를 사용한 병렬 CFD_NIMR 프로그램의 성능 부하는 크게 통신에서의 부하와 계산의 차이에서 나타나는 프로세서의 휴지 현상으로 나누어지며 일반적으로 프로세서의 수가 많아질수록 프로세서간의 통신 횟수가 많아지고 이에 따라 부하불균형이 발생하게 되므로 프로그램의 실행 시에 부하가 많이 나타나게 된다.

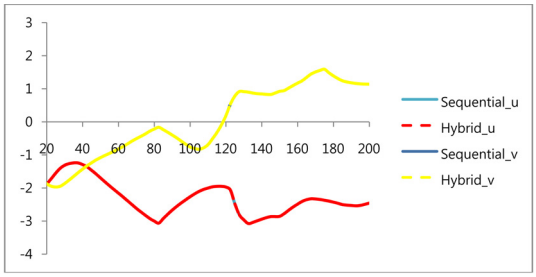

Fig. 6. Time series of u and v from result of sequential and parallel CFD_NIMR model.

MPI 병렬프로그램이 이상 속도와 비교하여 비교적 낮은 성능의 원인은 프로그램의 특성상 프로세서간에 통신이 많이 발생하여 생긴 결과이며 또한 병렬프로그램이 파일의 입출력 등을 포함하는데 이 작업은 한 개의 프로세서만이 작업을 하기 때문에 작업을 하게 되어 성능에 영향을 주게 된다.

3.3. 혼합형 병렬프로그램의 성능

먼저 Fig. 8은 OpenMP를 사용한 병렬 CFD_NIMR 프로그램의 실행 결과를 보여준다. 병렬프로그램은 OpenMP만을 사용하기 때문에 하나의 노드에서만 실행이 가능하며 노드 내의 최대 8개의 프로세서 (다중코어)를 사용할 수 있다. 이 경우에 프로세서간의 데이터 통신이 없기 때문에 이상적인 속도 개선과 비교하여 효율적인 성능을 보여준다.

Fig. 9는 MPI와 OpenMP를 혼합하여 사용한 속도 개선의 결과이다. 2대의 노드에서 각각 8개의 프로세서를 공유메모리형으로 사용하며 다른 노드에서는 메시지 전송을 이용하여 사용하였다. 이 경우 MPI만 사용하여 16개의 프로세서를 사용하는 경우보다 (Fig. 7 참조) 뛰어난 성능의 개선을 보여주고 있음을 알 수

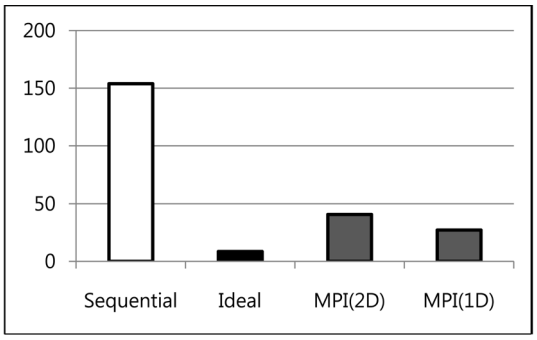

Fig. 7. Performance of sequential and MPI parallel CFD_NIMR model.

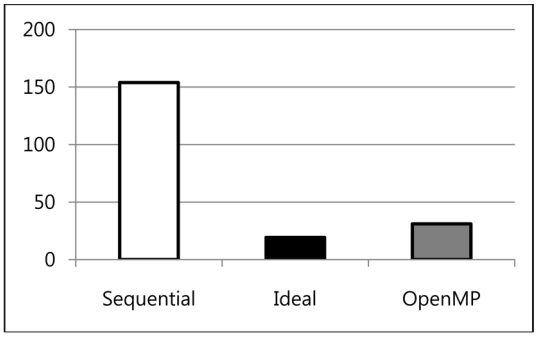

Fig. 8. Performance of sequential and OpenMP parallel CFD_NIMR model.

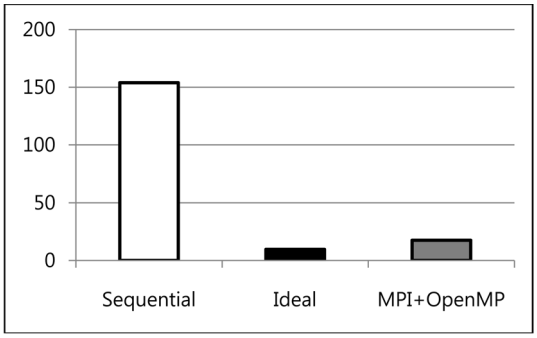

Fig. 9. Performance of sequential and Hybrid parallel CFD_NIMR model.

있다.

또한 병렬 CFD_NIMR 모델의 성능은 이상적인 속도 개선 대비 효율이 55%인데 이는 유사한 통신 방식을 사용하는 다른 병렬 기상 모델들과 비교했을 때 평균 이상의 성능을 보여준다 (Kim, 2011).

4. 병렬프로그램의 활용

CFD_NIMR 모델은 국지적인 기상 흐름을 분석하여 연구 및 실제 기상에 활용하기 위한 목적으로 개발되었다. 하지만 기존의 CFD_NIMR 모델은 계산 속도가 느려 계산량을 줄이거나, 모델에서 모의하는 시간을 줄이는 등 연구 설계의 수정이 불가피했다. 이는 원하는 연구 목표를 제한하는 한편, 실제 기상에 적용시키기에도 큰 제약을 가하였다. 특히나 실시간 운영이 가능한 예측 모델로써 기상 예측 정보를 산출하기 위해서는 계산 속도의 개선이 절실히 필요하였다. 따라서 병렬화를 통한 CFD_NIMR 모델의 속도 향상은 단순히 계산 속도의 증가가 아니라 CFD_NIMR 모델의 적용이 가능한 연구 범위를 확장시키고 실시간 운영의 가능성을 보여줌으로써 국지 기상 예측 모델로써 활용성을 높였다는데 의의를 찾을 수 있다. Fig. 10은 병렬화와 관련된 부분을 제외한 나머지 부분은 동일한 하드웨어 자원을 사용했을 때, 약 500초 전후의 계산 시간에서 산출된 CFD_NIMR 모델의 결과이다. 순차적인 CFD_NIMR 모델은 격자 수가 수평으로 60개인 반면에 병렬화 CFD_NIMR모델은 140개로 비슷한 계산 시간 동안 5.5배 큰 공간에 대해 계산을 완료했다. 같은 자원을 사용하였을 때, 병렬화는 많은 계산량을 처리할 수 있어 기존에 비해 계산 영역을 확장하거나 계산 시간을 증가시킬 수 있다. 즉, 병렬화 CFD_NIMR 모델은 기존에 비해 많은 정보를 담을 수 있어 연구 범위의 확장 가능성을 보여준다.

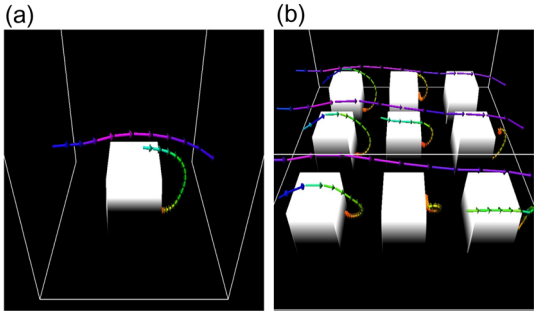

Fig. 10. Streamlines from result of (a) sequential and (b) parallel CFD_NIMR model.

5. 결 론

본 연구에서는 중규모 기상 모델인 CFD_NIMR 모델을 분산 및 공유메모리형의 병렬컴퓨터에서 효율적으로 실행하기 위하여 혼합형으로 병렬화하였다. 병렬 CFD_NIMR 프로그램은 분산메모리환경에서 효율적으로 실행하기 위하여 수평 방향으로 프로세서들을 분산하기 위하여 MPI를 사용하여 병렬화하였으며, MPI 함수들을 효율적으로 사용하기 위하여 객체지향형 병렬 라이브러리인 HPCC를 사용하였다. 또한 병렬프로그램은 노드당 다중코어의 사용을 극대화하기 위하여 수직 방향으로는 OpenMP를 사용한 병렬화를 하였다. 이러한 혼합형 병렬프로그래밍 방식은 순수한 분산메모리형의 컴퓨터를 위한 메시지 전송보다 효율적으로 실행될 수 있음을 보였다.

대부분의 국내에서 이루어지는 수치모델의 연구는 하드웨어의 비약적인 발달에도 대기 역학체계나 물리과정 계산식의 변화, 입력 자료 구축 등의 분야에 제한되어 있었으며 수치모델의 병렬화를 통하여 컴퓨터의 최적의 성능을 얻기 위한 노력은 부족하다고 볼 수 있다. 특히 지형이 복잡하여 고해상도의 모의가 요구되는 CFD_NIMR 모델의 특성 상 수치적 계산 시간의 향상은 반드시 필요한 것으로 CFD_NIMR 모델의 현실적인 사용을 가능케 한다는 점에서 큰 의의가 있다.

또한 혼합형 병렬 프로그램 방식은 CFD_NIMR 모델과 유사한 3차원 도메인을 사용하는 수치 모델들에 적용할 수 있으며, 이러한 병렬처리 방식은 향후 점차 수가 증가하는 프로세서당 다중 코어를 효율적으로 활용할 수 있게 된다. 또한 어떤 방식의 병렬 컴퓨터에서도 실행이 가능하기 때문에 호환성이 뛰어나다.

감사의 글

본 연구는 국립기상연구소, 응용기상연구과의 “녹색성장 지원기술 개발 연구 사업”의 일환으로 수행되

었습니다.

참고문헌

국립기상연구소, 2006: 국지 기상 특성 진단 및 기상환경영향평가 기술개발 연구 (I), 218pp.

Chandra, R., L. Dagum, D. Kohr, D. Maydan, J. McDonald and R. Menon, 2000: Parallel Programming in OpenMp. *Morgan Kaufmann*, 8-12.

Chapman, B., G. Jost and R. Pas, 2007: Using OpenMP: Portable Shared Memory Parallel Programming. MIT Press, 300pp.

Gropp, W., E. Lusk and R. Thakur, 1999: Using MPI-2: Advanced Features of the Message-Passing Interface. MIT Press, 382pp.

Kim, Y., 2011: High Performance Computing Classes (HPCC) for Parallel Fortran Programs using Message Passing. *정보과학회논문지*, **38**, 59-66.

Michalakes, J., 2000: RSL: A parallel runtime system library for regional atmospheric model with nesting, in Structured Adaptive Mesh Refinement (SAMR) Grid Methods. *IMA Volumes in Mathematics and Its Applications*, **117**, 59-66.

Michalakes, J., J. Dudhia, D. Grill, T. Henderson, J. Klemp, W. Skamarock and W. Wang, 2004: The Weather Research and Forecast Model: Software Architecture and Performance. *Proc. 11th ECMWF Workshop on the Use of high Performance Computing in Meteorology*, 25-29.

Pacheco, P., 1996: Parallel Programming with MPI. *Morgan Kaufmann*, 12-31.

Patankar, S. V., 1980: Numerical Heat Transfer and Fluid Flow. McGraw-Hill, 126-131.

Roe, P. K. and D. Stevens, 2010: Maximizing the Performance of the Weather Research and Forecast Model over the Hawaiian Islands. *High Performance computing Modernization Program Users Group Conference*, 303-310.

Top500 Supercomputer Sites, The Top 500 Report, 23 June 2009, 1pp.