<학술논문> DOI http://dx.doi.org/10.3795/KSME-B.2014.38.1.071 ISSN 1226-4881(Print)
2288-5234(Online)

# Symmetric Multi-Processing 시스템에서 다양한 병렬 기법 모델을 적용한 병렬 CUPID 코드의 성능분석

전병진* · 이재룡**· 윤한영**· 최형권***†
* 서울과학기술대학교 에너지환경대학원 에너지시스템공학과,
** 한국원자력연구원 열수력안전연구부, *** 서울과학기술대학교 기계자동차공학과

# Performance Analysis of the Parallel CUPID Code for Various Parallel Programming Models in Symmetric Multi-Processing System

Byoung Jin Jeon*, Jae Ryong Lee**, Han Young Yoon** and Hyoung Gwon Choi***†
* Dept. of Energy System, Graduate School of Energy and Environment, Seoul Nat'l Univ. of Science and Technology
** Thermal Hydraulics Safety Research Division, Korea Atomic Energy Research Institute
*** Dept. of Mechanical/Automotive Engineering, Seoul Nat'l Univ. of Science and Technology

(Received August 23, 2013 ; Revised October 1, 2013 ; Accepted October 23, 2013)

**Key Words**: Symmetric Multi-Processing(대칭형 다중처리), Bi-Conjugate Gradient(이중공액구배법), Parallel Programming Model(병렬 프로그래밍 모델), CUPID(큐피드 프로그램)

**초록**: 본 연구에서는 가압경수로 주요 기기의 고정밀 열수력 해석을 위한 CUPID(Component Unstructured Program for Interfacial Dynamics) 코드의 압력장 해석을 위한 이중공액구배법(Bi-Conjugate Gradient) 알고리즘의 병렬화를 SMP(Symmetric Multi Processing) 시스템에서 고찰한다. 비압축성 후향계단 유동문제의 병렬해석을 다양한 격자 조밀도를 가지는 격자들에 대하여 세 가지 대표적인 병렬 기법(MPI, OpenMP, 하이브리드)을 적용하여 병렬성능 비교를 수행하였다. 병렬처리 성능은 해석 문제의 크기뿐만 아니라 캐쉬 메모리 크기에도 영향을 받으므로, 전체 계산량이 매우 적거나 개별 쓰레드에 사용되는 메모리가 캐쉬 메모리보다 매우 큰 경우에는 병렬화에 의한 성능 향상이 낮음을 확인하였다. 또한, 문제 크기에 상관없이 MPI 기법이 OpenMP보다 성능이 우수했으며, 상대적으로 적은 쓰레드를 사용한 경우엔 하이브리드 기법이 가장 우수한 성능을 보였다.

**Abstract**: A parallelization of the bi-conjugate gradient solver for the pressure equation of the CUPID (component unstructured program for interfacial dynamics) code, which was developed for analyzing the components of a pressurized water-cooled reactor, was studied in a symmetric multi-processing system. The parallel performance was investigated for three typical parallel programming models (MPI, OpenMP, Hybrid) by solving incompressible backward-facing step flow at various grid resolutions. It was confirmed that parallel performance was low when problem size was small or the memory requirement for each thread was considerably higher than the cache memory. Furthermore, it was shown that MPI was better than OpenMP regardless of the problem size, and Hybrid was the best when the number of threads was relatively small.

## 1. 서 론

컴퓨터의 성능은 지속적으로 빠르게 증가하였고 풀고자하는 해석 대상의 규모는 점점 커지고 있다. 그러나 에너지 소모와 발열처리 문제로 인하여 CPU의 클럭 주파수를 증가시키는 것은 제한되었다. 따라서, 연산 능력을 증가시키기 위해 하드웨어 업체들은 CPU의 클럭 주파수를 낮추어 하나의 칩에 다수의 프로세서를 집적하고 여러 개의 컴퓨터를 네트워크로 연결하여 대량의 작업량을 분산시켜 계산하는 병렬 클러스터를 개발하고 있으며 최근에도 지속적으로 고성능/고집적의 병렬 클러스터들이 출시되고 있다.[1] 이러한 하드웨어의 발전에 따라 소프트웨어 개발자들은 다양한 병렬 기법을 이용하여 계산을 수행하고 있다.

† Corresponding Author, hgchoi@seoultech.ac.kr

본 연구에서는 최근의 고속연산용 병렬 클러스터를 효과적으로 활용하기 위하여, 몇 가지 대표적인 병렬 기법을 CFD(Computational Fluid Dynamics) 문제의 해법에 적용하여 그 특성을 파악하고자 한다.

본 내용을 서술하기에 앞서, 본 논문에서 주요하게 사용되는 용어는 다음과 같이 정의된다. 쓰레드(코어)는 컴퓨터에서 작업을 수행하는데 필요한 최소 단위의 연산 장치이며, 멀티쓰레드(멀티코어)는 두 개 이상의 쓰레드를 동시에 사용하는 것을 의미한다. 그리고 하이퍼 쓰레딩은[2]은 인텔 CPU에서 사용하지 않은 실행유닛에 다른 작업을 할당하여 성능을 높이려는 기술이다. 노드는 독립적으로 계산을 수행할 수 있는 하나의 컴퓨터를 의미한다. MPP(Massively Parallel Processing) 시스템은 노드마다 다른 CPU와 메모리를 사용하며, 각 노드는 네트워크(이더넷, 인피니트 밴드)를 통해 서로 유기적으로 연결되어 있다. SMP(Symmetric Multi Processing) 시스템은 모든 CPU가 하나의 메모리를 공유하는 구조로 구성되어 있다. CC-NUMA (Cache Coherent Non-Uniform Memory Access) 시스템은 분산된 메모리들을 결합하여 하나의 메모리로 구성한 단일 노드 개념으로 캐시 일관성 문제를 하드웨어적으로 해결하여 메모리 접근 지연 시간을 단축할 수 있도록 만들었다.

MPI(Message Passing Interface)[3]는 여러 병렬 프로그래밍 모델 중에서 메시지 패싱 모델의 표준으로 알려져 있다. 메시지 패싱 모델은 MPP 시스템과 같은 각자의 메모리를 지역적으로 따로 가지는 프로세스들로 구성된 분산 시스템 환경에서 이더넷 또는 인피니트 밴드를 통해 메시지들의 송신과 수신으로만 구현하는 프로그래밍 모델을 의미한다. 강성우 등[4,5]은 자동차 모델 주위의 유동해석을 위하여 Metis 라이브러리[6]를 이용한 영역분할 기법과 MPI를 적용한 병렬프로그램을 개발하였다. 그들은 영역분할로부터 얻어진 행렬을 위한 Block ILU(Incomplete LU) 예조건화 기법을 적용하여 50개의 쓰레드를 사용하여 50배 이상의 속도 증가를 얻을 수 있었다. 김정녀 등[7]은 CC-NUMA 시스템에서 원격 메모리 접근 대기 시간과 지역 메모리 접근 대기 시간을 비교하면서 MPI 병렬 방식에 의한 통신 부하량에 대해서 보고하였다. 이와 같이 MPI 병렬 방식은 작업량의 분할에 의한 성능 향상과 원격 메모리 접근 횟수에 의한 성능 감소를 고려하여 계산을 수행해야 한다. OpenMP(Open Multi- Processing)[8]는 공유메모리 환경에서 멀티쓰레딩 기반의 프로그램으로 병렬화하는 지시어 기반의 병렬 모델이다. OpenMP를 이용한 병렬화가 쉬우나 MPI 모델이 계속 사용되는 이유는 일반적이고 다양한 플랫폼에서 이용할 수 있기 때문이다[8]. 김종관 등[9]은 멀티쓰레드 프로세서가 장착된 시스템에서 OpenMP를 사용하여 병렬 프로그래밍을 적용한 CFD 코드를 이용하여 쓰레드 수와 하이퍼 쓰레딩(Hyper-threading)에 의한 계산 성능 향상에 대해서 연구를 수행하였다. 김종관 등[9]은 하이퍼 쓰레딩을 이용하였을 때 약간의 성능 향상을 얻을 수 있다고 보고하였다.

MPI와 OpenMP를 결합한 병렬처리방식은 통상 하이브리드 병렬 방식으로 명칭된다. 이 병렬 모델은 개별 노드가 SMP 방식으로 구성된 클러스터 시스템 환경에서 사용되며, 노드 간의 병렬화는 메시지 패싱 모델인 MPI를 사용하고 각 노드에서는 OpenMP를 이용한 멀티쓰레드 방식의 병렬화를 사용하는 방법이다. Rabenseifner, R.[10]은 SMP 방식의 노드로 구성된 클러스터 시스템 환경에서 NAS Parallel Benchmark 문제를 이용하여 MPI와 OpenMP, 하이브리드 병렬 모델의 성능 테스트 수행 및 장단점에 대해서 보고하였다.

본 연구에서는 SMP 방식의 단일 시스템에서 다양한 병렬 기법(MPI, OpenMP, 하이브리드)의 특성을 파악하기 위하여, 가압경수로 주요 기기의 고정밀 열수력 해석을 위한 CUPID(Component Unstructured Program for Interfacial Dynamics) 코드[11-14]의 압력방정식의 병렬연산을 수행하고 그 결과를 비교하고 분석하고자 한다. 이산화된 압력방정식의 해는 병렬 이중공액구배법(Parallel Bi-Conjugate Gradient) 알고리즘를 이용하여 구하였으며, CUPID 코드의 요약은 2.2절을 참고한다.

## 2. 시스템과 병렬 CUPID 코드 소개

### 2.1 소프트웨어 및 하드웨어 환경

본 연구에서는 다양한 병렬 기법과 문제 크기에 따른 병렬 성능을 분석하기 위해서 Red Hat Enterprise Linux Server 6.2를 운영체제를 사용하는 MPP 방식의 클러스터 시스템에서 계산을 수

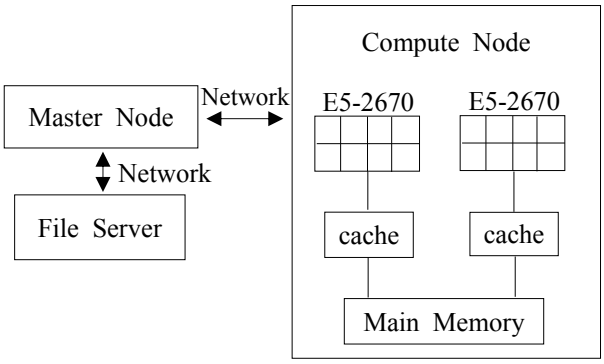



**Fig. 1** Structure of SMP parallel computing system

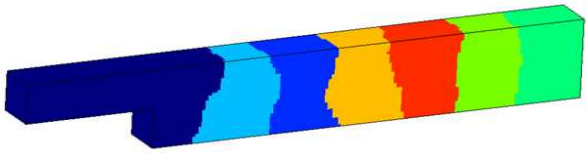


**Fig. 2** Partitioned mesh for backward-facing step flow problem

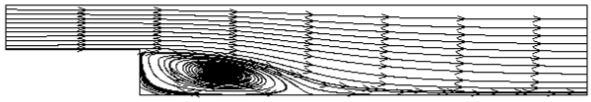


**Fig. 3** Streamline of backward-facing step flow problem

행하였다. 컴파일러는 최근에 출시된 Intel(R) Fortran Composer XE 2013 for linux[15]를 사용하였다. 본 연구에서 사용한 계산 노드는 Intel(R) Xeon CPU E5-2670 제품 2개와 64GB의 메인 메모리로 구성되어 있다. Intel(R) Xeon CPU E5-2670 제품은 클럭수가 2.60GHz인 쓰레드가 8개로 구성되어 있으며, 각 쓰레드는 20MB의 독립적인 캐쉬 메모리를 보유하고 있다. Fig. 1은 본 연구에서 사용한 SMP 방식의 시스템 구조를 나타낸다. 사용자는 마스터 노드에 접속하여 컴파일 및 계산을 위한 준비 작업을 수행한다. 계산을 수행할 때 필요한 파일들은 파일 서버에 저장된다. 준비 작업이 끝나고 계산 노드에서 실행 파일을 실행하면 네트워크를 통해 파일 서버에 접속하여 필요한 데이터를 계산 노드의 메인 메모리로 복사하고 계산을 수행한다.

### 2.2 CUPID 코드

원자로 냉각계통에서의 열수력 현상은 이상유동 및 복잡한 기하학적 형상들로 정확한 예측이 매우 어렵다. 이에 따라 원자력 연구소에서는 가압경수로 주요 기기의 고정밀 열수력 해석을 위한 CUPID(Component Unstructured Program for Interfacial Dynamics) 코드를 개발하고 있다. CUPID 코드는 다차원 3-장(three-field), 2-유체(two-fluid) 보존 방정식을 채택하고 있다. 여기서 3-장은 액체(continuous liquid), 기체(gas), 액적(droplet)을 나타내며, 2-유체는 액체 및 기체를 나타낸다. 이 외에 비응축 기체 및 붕소 농도 해석을 위한 열전도 방정식, 그리고 물성치 계산을 위한 상태방정식을 주요 지배방정식으로 채택하였다. 또한 난류 방정식과 이상유동의 고정밀 해석을 위한 계면 면적 수송 방정식을 포함하였다. 수치 해법은 반 내재적 기법(Semi-Implicit Scheme)에 기초하고 있으며 지배방정식의 이산화는 유한체적법(Finite Volume Method) 사용한다. 계산 격자는 복잡한 기하학적 형태를 갖는 문제에 대한 적용이 매우 유용한 비엇갈림(non-staggered) 비정렬(unstructured) 격자를 사용한다. CUPID 코드는 대표적인 확인 및 검증 문제들을 통하여 수치 안정성, 정확성 및 적합성을 확인하였으며,[11,12] 원자로 안전과 관련한 현안 문제들의 수치해석에 활용하고 있다.[13,14] Fig. 2는 비압축성 후향 계단 유동 문제에 대한 3차원 형태의 계산 영역을 METIS 라이브러리를 이용하여 8개의 영역으로 분할한 것이다. Fig. 3은 비압축성 후향 계단 유동 문제 해석을 수행하여 얻은 유선 분포 결과이다.

**Table 1** Memory requirement for each problem size

| Number of cells | 23,000 | 100,283 | 144,000 | 288,305 | 1,472,000 |
|---|---|---|---|---|---|
| Memory requirement (MB) | 12 | 36 | 40 | 120 | 480 |

* CPU E5-2670(Cache memory of 1 thread : 20MB)

Table 1은 다양한 문제 크기에 대해서 CUPID 코드의 이중공액구배법 알고리즘을 수행할 때 요구되는 메모리량을 나타낸다. 셀의 개수가 23,000개인 격자계의 경우에 사용하는 메모리는 1개의 쓰레드가 가지고 있는 캐쉬 메모리보다 작다. 셀의 개수가 100,283~ 288,305개인 격자계의 경우에 병렬화를 수행하면 요구되는 메모리량이 캐쉬 메

모리에 적합한 크기로 나누어진다. 셀의 개수가 1,472,000인 격자계의 경우에는 16개의 쓰레드를 사용하여 요구되는 메모리량을 나누어도 캐쉬 메모리보다 크게 된다.

### 2.3 병렬 연산 기법

본 연구에서는 이산화된 압력방정식에서 얻어진 행렬의 해법을 위하여 예조건화된 이중공액구배법(Preconitioned Bi-Conjugate Gradient)을 사용하였으며, 예조건화 기법으로는 대각 예조건화(Diagonal Preconditioner)를 사용하였다. 예조건화된 이중공액구배법과 영역분할, MPI에 관련된 자세한 내용은 강성우 등[16]의 논문을 참고한다. 이 절에서는 세 가지의 병렬 연산 기법에 대하여 간략히 서술하며, 자세한 내용은 인용된 문헌을 참고한다.

#### 2.3.1 OpenMP를 이용한 이중공액구배법 알고리즘의 병렬화

이중공액구배법에서의 대부분의 연산은 행렬과 벡터의 곱, 벡터의 내적 및 스칼라가 곱해진 벡터에 벡터를 더하는 연산으로 구성된다. 이들 연산들은 모두 do 루프로 구성되어 있다. 본 연구에서는 do 루프에 대해서 parallel do 지시어와 parallel do reduction 지시어 등을 이용하여 이중공액구배법 알고리즘의 병렬화를 수행하였다. 자세한 내용은 Lof, H. 등[17]의 논문을 참고한다.

#### 2.3.2 영역분할과 MPI를 이용한 이중공액구배법 알고리즘의 병렬화[16]

효과적인 병렬화를 위해서 쓰레드간의 계산량을 균등하게 분배하고 자료 교환을 최적화하기 위해 Metis 라이브러리를 사용하여 영역을 분할하고 쓰레드간의 자료 교환은 MPI를 사용하여 예조건화된 이중공액구배법 알고리즘을 병렬화하였다. Fig. 4는 2차원 형태의 계산 영역을 4개의 영역으로 분할한 것이다. 분할한 영역의 개수가 증가하면 내부 셀들의 수에 비하여 경계 셀들의 수가 상대적으로 증가하여 통신 부하가 커지게 된다. 그리고 Fig. 5와 같이 분할된 영역은 내부 셀, 내부 경계 셀, 외부 경계 셀로 구성된다. 내부 셀은 각 쓰레드가 독립적으로 계산하며, 경계 셀들은 계산하기 전에 주위 쓰레드와 값의 교환을 필요로 한다. 분할된 각 영역에서는 내부 셀

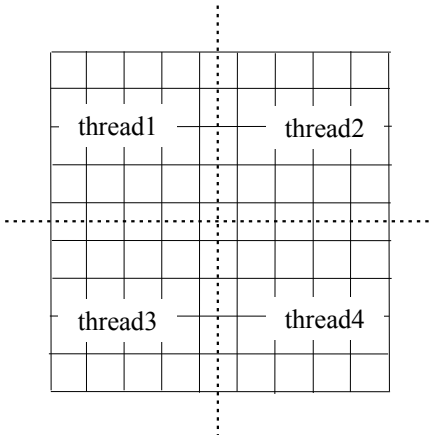



**Fig. 4** Decomposition of computational domain

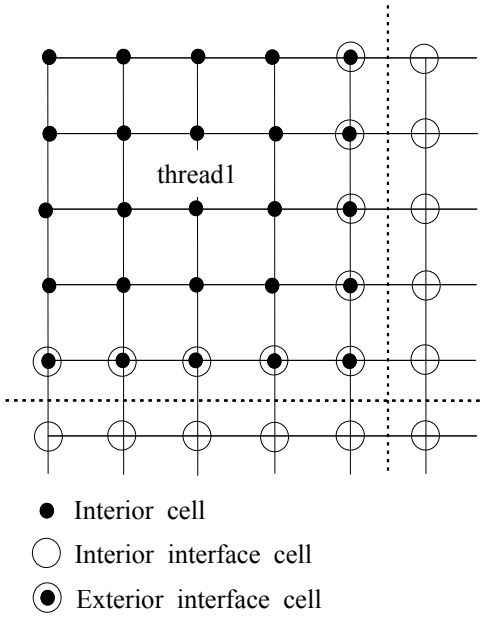



**Fig. 5** Classification of cells in a sub-domain[16]

과 내부 경계 셀에 대해서만 계산하며, 각 영역에서 국소행렬이 구성될 때 외부 경계 셀에 대한 행은 무시된다. 자세한 내용은 강성우 등[16]의 논문을 참고한다.

#### 2.3.3 하이브리드 병렬 기법을 이용한 이중공액구배법 알고리즘의 병렬화

서론에서 설명한 대로 하이퍼 쓰레딩은 개별 쓰레드에서 사용하지 않은 실행유닛에 다른 작업을 할당하여 성능을 높이려는 기술이다. 즉. 사용하지 않은 실행유닛에 다른 작업을 할당하기 때문에, 병렬 연산의 측면에서 하이퍼 쓰레딩은 하나의 쓰레드가 두 개로 분리된 작업을 병렬로 수행하는 방식으로 설명할 수 있다. 따라서, 하이브리

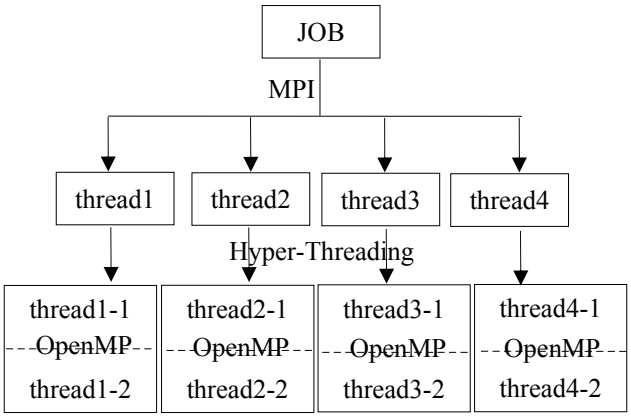



**Fig. 6** Schematic of hybrid parallel computing using hyper-threading

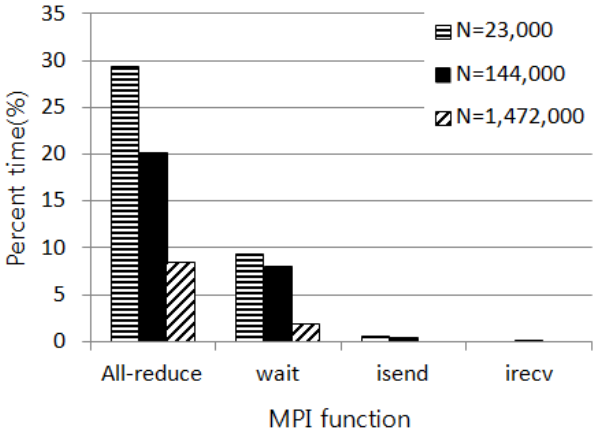



**Fig. 7** Overhead of the MPI communications for various problem sizes

드 병렬기법은 Fig. 6에서 도식화한 것처럼, 각 영역에 MPI 방식에 의하여 할당된 개별 쓰레드가 하이퍼 쓰레딩에 의해 두 개로 분리된 작업을 OpenMP 방식으로 수행되는 것으로 정의한다. 즉, 영역간의 병렬연산은 MPI에 의해서, 영역내의 연산은 개별 쓰레드가 하이퍼쓰레딩에 의해서 두 개로 분리된 작업을 OpenMP 방식에 의해서 수행한다.

따라서, 본 연구에서는 이중공액구배법 알고리즘에 하이브리드 병렬 기법을 적용하기 위해 영역분할과 MPI를 이용하여 병렬화를 수행한 다음에 OpenMP를 이용하여 do 루프에 대해서 parallel do 지시어와 parallel do reduction 지시어를 적용하였다.

## 3. 해석 결과 및 토의

본 연구에서는 16개의 쓰레드로 구성된 SMP 방식의 시스템에서 다양한 셀의 개수를 가지는 격자계들에 대해 MPI, OpenMP, 하이브리드 방식의 병렬 기법을 적용하여 병렬 성능을 측정하였다.

### 3.1 전체 계산 수행 시간에 대한 MPI 함수 수행 시간 비교

Fig. 7은 3가지 격자계에 대해서 쓰레드를 8개 사용하였을 때, 전체 계산 시간에 대한 MPI 함수 수행 시간의 비율을 나타낸 결과이다. 여기서, N은 셀의 수를 의미한다. 셀의 개수가 작을수록 전체 계산 시간에서 통신 부하가 차지하는 비율이 높은 것을 알 수 있다. 셀의 개수가 23,000개인 격자계의 경우에는 MPI 함수 수행 시간이 전체 계산 시간에 약 40%를 차지한다. 그리고, 각 쓰레드에서 계산된 벡터들을 모두 모으는 All reduce 함수를 수행하는데 걸리는 시간은 MPI 함수 중에서 가장 많은 비율을 차지하였고, 그 다음이 다중 쓰레드의 동기화를 위한 Wait 함수이다. 쓰레드 간의 경계 데이터를 주고 받는 Isend 함수나 Irecv 함수는 매우 적은 비율을 차지하고 있다. Fig. 7에서 제시한 MPI 함수 문법에 대한 자세한 설명은 Snir, M. 등[3]의 논문을 참고한다.

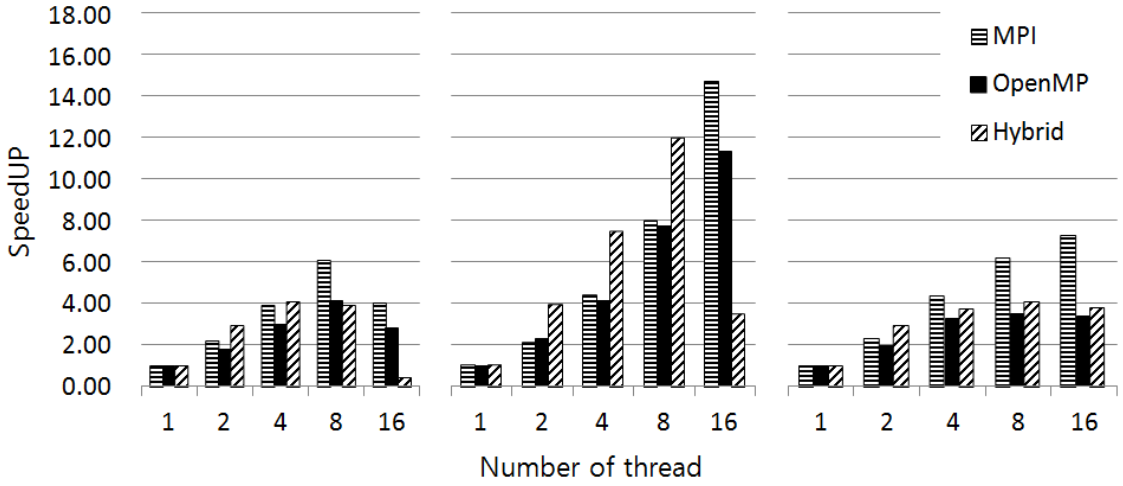
### 3.2 병렬처리 성능 결과

Fig. 8은 2.2 절에서 언급한 CUPID 코드의 이중공액구배법 알고리즘에 대해서 다양한 셀의 개수를 가지는 격자들에 세 가지 대표적인 병렬 기법을 적용하여 얻은 병렬 성능 결과를 나타낸다. 사용된 병렬 기법은 OpenMP, MPI, 하이브리드 기법들로 각 기법들의 사용방법에 대한 자세한 설명은 2.3절에 서술되어 있다.

#### 3.2.1 셀의 개수가 적은 격자계에 대한 병렬처리 성능 결과 분석

Fig. 8(a)는 셀의 개수가 23,000개일 때 다양한 병렬 기법에 대해서 쓰레드 수에 따른 속도 향상 결과를 나타낸 그림이다. MPI 병렬 기법의 성능은 쓰레드 수가 4개일 때까지 선형으로 증가하다가 8개 일 때 증가 폭이 감소하고 16개일 때에는 오히려 성능이 떨어지는 것을 볼 수 있다.

OpenMP 병렬 기법은 MPI보다 성능이 떨어지지만 쓰레드 수가 8개일 때까지 병렬 성능이 증가하다가 16개일 때 성능이 떨어진다. 그리고 하




(a) N=23,000 (b) N=144,000 (c) N=1,472,000

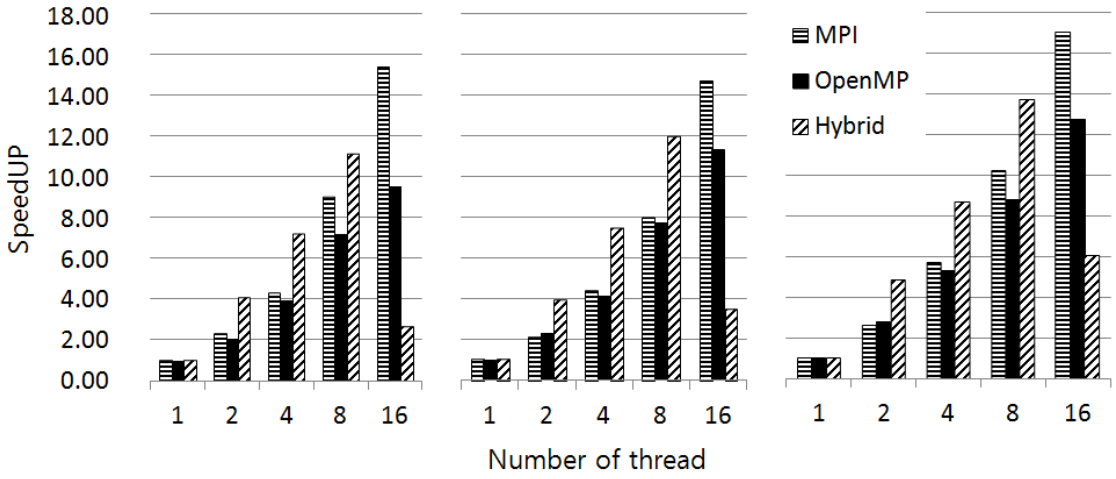
**Fig. 8** Comparison of parallel performances of Bi-CG algorithm in CUPID code for the three meshes

이브리드 병렬 기법의 경우에는 쓰레드 수가 4개일 때까지 성능 향상이 나타나는 것을 볼 수 있다. 쓰레드의 개수가 8개일 때 하이브리드 병렬 기법은 MPI 병렬 기법의 8개 영역분할에 의해서 각각의 영역을 담당한 개별 쓰레드가 하이퍼 쓰레딩에 의해 2개의 서브 쓰레드를 생성하고(Fig. 6 참고), 각 영역에서 생성된 2개의 서브 쓰레드는 OpenMP 병렬 기법을 사용하게 된다. 이 경우에 각 서브 쓰레드 당 OpenMP 병렬 기법에 의해서 할당된 계산량은 Fig. 8(a)의 16개의 쓰레드를 사용한 OpenMP 병렬 기법에서 각 쓰레드가 담당한 계산량과 동일하다. 그런데, Fig. 8(a)처럼 문제의 크기가 적은 계산에서는 8개로 영역분할된 국소영역에 해당하는 행렬의 크기가 더욱 작아진다. 이 경우에 각 영역에서 서브 쓰레드들이 OpenMP 병렬 기법을 사용하면 OpenMP 병렬 기법을 사용함에 따라 발생하는 개별 서브 쓰레드들 간의 통신 부하의 영향 때문에 속도가 오히려 느려지게 된다. 이는 Fig. 8(a)에서 OpenMP 병렬 기법을 사용하는 경우에 8개에서 16개로 쓰레드의 개수가 증가했을 때 속도가 오히려 낮아지는 결과에서도 다시 확인할 수 있다.

한편, 프로그램이 실행될 때 개별 쓰레드는 쓰레드의 성능을 100% 사용하지 않고 약 50~60%만을 사용한다.[18] 이러한 이유로 하이퍼 쓰레딩 기술을 사용하는 것이 가능하며, 하이퍼 쓰레딩은 쓰레드 성능의 거의 100%를 사용하게 된다. 따라서, CPU를 구성하는 모든 쓰레드가 하이퍼 쓰레딩 연산을 수행하는 경우에는 운영체제 시스템 등이 쓰레드의 일부를 사용하면서 작업을 분배하는 균형이 깨지기 때문에 성능이 급격하게 저하되는 현상이 나타난게 된다.[9] 그러므로, 하이브리드 기법을 사용하면 쓰레드 개수가 16개일 경우에 성능이 급격히 떨어진다.

### 3.2.2 문제의 크기와 캐쉬 메모리에 따른 병렬 처리 성능 결과 분석

Fig. 8(b)는 셀의 개수가 144,000개인 격자계에 대한 결과를 나타낸 그림이다. 셀의 개수가 144,000개인 격자의 경우에 MPI 병렬 기법은 적은 쓰레드를 사용할 때 super-linear한 속도 향상이 나타나는 것을 볼 수 있다. 이 결과는 캐쉬 메모리의 효과로 설명할 수 있다. 2.2절의 Table 1에서 제시된 자료와 같이 셀의 개수가 144,000개인 격자계의 경우에는 계산을 수행할 때 약 40MB의 메모리를 사용한다. 이 문제를 병렬화하면 개별 쓰레드에서 계산에 사용되는 메모리의 크기(40MB/쓰레드수)가 각 쓰레드 당 20MB로 주어지는 캐쉬 메모리 크기보다 적게 되어 캐쉬 메모리의 최대 성능을 사용하기 때문에 병렬 성능이 매우 좋게 나타난다. OpenMP 병렬 기법을 적용한 경우에는 MPI 병렬 기법보다 성능 향상은 적지만 쓰레드가 증가할수록 병렬 성능이 높아지는 것을 볼 수 있다. 하이브리드 병렬 기법은 MPI 또는 OpenMP 기법보다 쓰레드 수가 2개, 4개일 경우에 약 2배 빠르고, 8개일 경우에는 약 1.5배 빠른 것을 알 수 있다. 그러나 쓰레드 수가 16개가 되면 3.2.1절에서 설명한 이유와 마찬가지로 성능이 급




(a) N=100,283 (b) N=144,000 (c) N=288,305

**Fig. 9** Parallel performances of Bi-CG algorithm in CUPID code for the optimal sized problems

격히 낮아지는 것을 볼 수 있다.

Fig. 8(c)는 셀의 개수가 1,472,000개인 격자계에 대한 결과를 나타낸 그림이다. 셀의 개수가 1,472,000개인 격자계의 병렬 성능은 셀의 개수가 144,000개인 격자계보다 좋지 않은 것을 볼 수 있다. 셀의 수가 1,472,000인 격자계에 대해서 계산을 수행할 때 요구되는 메모리양은 480MB이며 쓰레드를 16개까지 사용하여 병렬화를 수행해도 쓰레드당 주어진 20MB의 캐쉬 메모리보다 매우 크다. 따라서, 캐쉬 메모리에 의한 속도 향상 효과는 나타나지 않는다. OpenMP 병렬 기법을 적용한 경우에는 MPI 병렬 기법을 적용한 경우보다 병렬 성능이 좋지 않다. 그 이유는 큰 문제의 경우에 캐쉬 메모리의 부족과 계산을 수행하기 위해 메모리에 접근할 때 같은 파이프라인을 사용하는 공유 메모리 환경의 특성으로 인한 지연 문제 때문인 것으로 여겨진다. 그리고 셀의 수가 큰 경우에 하이브리드 병렬 기법이 MPI 병렬 기법보다 성능이 낮은 이유는 캐쉬 메모리의 부족과 OpenMP 병렬 기법에서 나타나는 문제에 의한 성능 감소 때문인 것으로 판단된다.

Fig. 9에서는 문제의 크기와 캐쉬 메모리 크기에 따른 병렬 처리 성능을 재확인하기 위해서 셀의 개수가 144,000개인 격자를 기준으로 100,283개와 288,305개의 셀로 구성된 격자에 대한 계산 결과를 비교하였다. Table 1에서 정리한 바와 같이 셀의 개수가 100,283개인 격자의 경우에는 36MB, 144,000개인 격자의 경우에는 40MB, 288,305개인 격자의 경우에는 120MB의 메모리를 사용한다. 3가지 격자계는 계산량이 충분하고 병렬화에 의해서 분배된 메모리의 크기가 20MB의 캐쉬 메모리 크기에 적당하다. MPI 병렬 기법을 적용했을 때 셀의 개수가 100,283개와 144,000개인 격자계의 경우에는 쓰레드를 8개까지 사용했을 때 linear 또는 super linear한 형태의 속도 향상을 보이며, 쓰레드를 16개 사용했을 때는 계산 속도가 약 15배정도 향상되는 것을 알 수 있다. 셀의 개수가 288,305인 격자계의 경우에는 3가지 격자계 중에서 영역분할 후에 경계 셀들의 수에 대한 내부 셀들의 수의 비율이 커서 통신부하가 적고, 각 영역에서 계산에 사용되는 메모리의 크기가 캐쉬 메모리 크기와 크게 다르지 않기 때문에 전체적으로 super linear한 형태의 속도 향상이 나타난다. 쓰레드 수가 16개인 경우에 계산 속도가 17배까지 향상되는 것을 볼 수 있다. OpenMP 병렬 기법을 적용한 경우에는 쓰레드의 증가에 따라서 계산 속도가 향상되지만 MPI 병렬 기법보다는 병렬 성능이 좋지 않은 것을 알 수 있다. 하이브리드 병렬 기법을 적용한 경우에는 쓰레드를 적게 사용하였을 때 병렬에 의한 속도 향상이 크게 나타났으며, 셀의 개수가 증가할수록 병렬 성능이 좋아지는 것을 볼 수 있다. 그러나, 쓰레드 수가 16개가 되면 3.2.1절에서 설명한 이유와 마찬가지로 세 가지 격자계 모두 성능이 급격히 낮아지는 것을 볼 수 있다.

## 4. 결 론

본 연구에서는 CUPID 코드의 이중공액구배법 알고리즘에 대해서 셀의 개수와 병렬 기법(MPI,

OpenMP, 하이브리드)에 따른 병렬처리 성능 비교를 수행하였다.

(1) 문제의 크기가 작은 경우에는 큰 경우보다 전체 계산 수행에 대한 통신에서 사용되는 시간의 비율이 상대적으로 높다. 그리고 쓰레드 수가 많아질수록 통신에서 사용되는 시간은 상대적으로 증가한다.

(2) 병렬 처리 성능은 풀고자하는 문제의 크기와 캐쉬 메모리의 크기에 영향을 많이 받는다. 계산량이 적거나 계산을 푸는데 사용되는 메모리가 캐쉬 메모리보다 매우 큰 경우에는 병렬화에 의한 성능 향상이 낮다.

(3) 같은 단일 노드 시스템 환경에서 MPI는 OpenMP보다 더 좋은 병렬처리 성능을 보인다. 하이브리드 병렬 기법은 적은 쓰레드를 사용하는 경우에 MPI, OpenMP 병렬 기법보다 다소 좋은 성능 향상이 나타남.

## 후 기

본 연구는 미래창조과학부와 한국연구재단에서 지원하는 원자력연구개발사업의 일환으로 수행되었다.(Grant Code : 2012M2A8A4025647)

## 참고문헌

(1) Kirk, D. B. and Hwu, W. W, 2010, Programming Massively Parallel Processors, Elsevier Inc, pp. 19~27.

(2) Lee, J. H., Oh, Y. E. and Kim, J. S., 2003, "Design and Implementation of GRID MDS for Hyperthreading," *KISS 2003 Fall Conference*, Vol. 30, pp. 166~168.

(3) Snir, M., Otto, S., Huss-Lederman, S., Walker, D. and Dongarra, J., 1996, MPI: The Complete Reference, The MIT Press.

(4) Kang, S. W., Choi, H. G. and Yoo, J. Y., 2002, "Parallelized Dynamic Large Eddy Simulation of Turbulent Flow Around a Vehicle Model," *Proceedings of the KSME 2002 Spring Annual Meeting*, pp. 1562~1567.

(5) Choi, H. G., Kang, S. W. and Yoo, J. Y., 2008, "Parallel Large Eddy Simulation of Turbulent Flow around MIRA Model using Linear Equal-order Finite Element Method," *International Journal for Numerical Methods in Fluids,* Vol. 56, pp. 823~843.

(6) http://www-users.cs.umm.edu/~karypis/metis

(7) Kim, J. N., Kim, H. J. and Lee, C. H., 2000, "The Node Scheduling of Multi-Threaded Process for CC-NUMA System," *The Transactions of the Korea Information Processing Society*, Vol. 22, pp. 488~496.

(8) Jung, Y. H, 2011, OpenMP Parallel Programming, freelec, pp. 21~30.

(9) Kim, J. K., Jang, K. J., Kim, T. Y. and Choi, J. Y., 2011, "OpenMP Parallel Performance of a CFD Code on Multi-core Systems," *Proceedings of the Korean Society of Computational Fluids Engineering 2011 Fall Annual Meeting,* pp. 254~258.

(10) Rabenseifner, R., 2003, "Hybrid Parallel Programming: Performance Problems and Chances," *Proceedings of the 45th CUG Conference* 2003*,* pp. 1~11.

(11) Jeong, J. J., Yoon, H. Y., Park, I. K., Cho, H. K. and Kim, J., 2008, "A Semi-implicit Numerical Scheme for Transient Two-Phase Flows on Unstructured Grids," *Nuclear Engineering and Design,* Vol. 238*,* pp. 3403~3412.

(12) Yoon, H. Y., Park, I. K., Kim, Y. I., Hwang, Y. D. and Jeong, J. J., 2010, "A Fast-Running Semi-Implicit Numerical Scheme for Transient Two-Phase Flows on Unstructured Grids," *Numerical Heat Transfer Part B: Fundamentals*, Vol. 56*,* pp. 432~454.

(13) Cho, H. K., Lee, S. J., Yoon, H. Y., Kang, K. H. and Jeong, J. J., 2013, "Simulation of Single- and Two-phase Natural Circulation in the Passive Condensate Cooling Tank using the CUPID code," *Journal of Nuclear Science and Technology,* Vol. 50*,* pp. 709~722.

(14) Yoon, H. Y., Jeong, J. J., Cho, H. K., Bang, Y. S. and Seul, K. W., 2013, "A Multi-scale Analysis of the Transient Behavior of an Advanced Safety Injection Tank," *Annals of Nuclear Energy,* Vol. 62*,* pp. 17~25.

(15) http://software.intel.com/en-us/fortran-compilers/

(16) Kang, S. W., Choi, H. G. and Yoo, J. Y.,

2003, “Parallel Preconditioner for the Domain Decomposition Method of the Discretized Navier-Stokes Equation,” *Trans. Korean Soc. Mech. B,* Vol. 27, pp. 753~765.

(17) Lof, H. and Rantakokko, J., 2006, “Algorithmic Optimizations of a Conjugate Gradient Solver on Shared Memory Architectures,” *International Journal of Parallel, Emergent and Distributed Systems,* Vol. 21, pp. 345~363.

(18) http://www.intel.com/support/kr/processors/pentium4/sb/CS-017371.htm