멀티코어 시스템에서 쓰레드 수에 따른 CFD 코드의 OpenMP 병렬 성능

김 종 관,[1] 장 근 진,[1] 김 태 영,[1] 조 덕 래,[1] 김 성 돈,[2] 최 정 열*[2]
[1]부산대학교 대학원 항공우주공학과
[2]부산대학교 항공우주공학과

OPENMP PARALLEL PERFORMANCE OF A CFD CODE ON MULTI-CORE SYSTEMS

J.-K. Kim,[1] K.-J. Jang,[1] T.Y. Kim,[1] D.-R Cho,[1] S.-D. Kim[2] and J.-Y. Choi*[2]
[1]Dept. of Aerospace Engineering, Graduate School of Pusan Nat'l Univ.
[2]Dept. of Aerospace Engineering, Pusan Nat'l Univ.

OpenMP is becoming more and more useful as a simple parallel processing paradigm on SMP (Shared Memory Multi-Processors) computing environment with the development of multi-core processors. However, very few data is available publically regarding the OpenMP performance in CFD (Computational Fluid Dynamics). In the present study a CFD test suite is prepared for the performance evaluation of OpenMP on various multi-core systems. The test suite is composed of two-dimensional numerical simulations for inviscid/viscous and reacting/non-reacting flows using three different levels of grid systems. One to five test runs were carried out on various systems from dual-core dual threads to 16-core 32-threads systems by changing the number of threads engaged for each test up to 80. The results exhibit some interesting results and the lessons learned from the tests would be quite helpful for the further use of OpenMP for CFD studies using multi-core processor systems.

Key Words : OpenMP, 멀티코어(Multi-core), 멀티쓰레드(Multi-thread), 속도향상(Speed-up), 전산유체역학(CFD)

1. 서 론

네트워크로 연결된 다중 컴퓨터에서 데이터 통신을 이용하는 병렬처리기법에 의하여 전산 해석 능력은 지난 20년 동안 기하급수적인 향상을 보여 왔으며, 마이크로프로세서의 발전은 과거 대형 슈퍼컴퓨터에서나 가능하던 대형의 공학 문제를 개인용 컴퓨터(PC) 수준에서 해결 할 수 있게 하였다. 지난 수년간 마이크로프로세서의 발전 방향이 집적도 향상에서 multi-core 프로세서 방식으로 전환되면서, message-passing 방식의 병렬처리기법에서 벗어나 메모리를 공유하는 단일 컴퓨터 내부에서 다중 프로세서를 이용한 효과적인 병렬처리기법의 중요성이 부각되고 있다. 최근에는 60개 이상의 프로세서 코어가 집적된 프로세서가 발표되는 등[1], 단일 컴퓨터 내의 다중프로세서를 이용한 병렬처리의 중요성이 더욱 증가하고 있다.

OpenMP는 SMP(Shared Memory Multi-Processors) 전산 환경에서 병렬처리를 손쉽게 구현할 수 있는 장점이 있어 최근들어 이용 분야가 급속히 넓어지고 있음에도 불구하고, 전산유체 해석에서 OpenMP를 활용할 경우 어느 정도 성능향상이 가능한지 알려진 자료는 많지 않은 상황이다. 따라서 본 연구에서는 OpenMP로 병렬 프로그래밍이 된 CFD 코드를 이용하여 몇 가지 유동 모델 및 문제의 크기를 고려한 병렬 처리 성능 평가 시험 패키지를 구성하여, 여러 기종의 multi-core 프로세서 장착 시스템에서 전산유체역학 병렬 처리 성능 특성을 살펴보았다.

Received: January 9, 2013, Revised: March 4, 2013, Accepted: March 5.
* Corresponding author, E-mail: aerochoi@pusan.ac.kr
DOI http://dx.doi.org/10.6112/kscfe.2013.18.1.083

2. OpenMP 병렬처리 성능평가 방법

2.1 OpenMP

OpenMP 는 SMP 환경에서 병렬 프로그래밍을 간단하게 하기 위해 개발된 API(Application Programing Interface)로써,

최근 대부분 컴파일러에 기본적으로 포함되어 있다. OpenMP는 컴파일러 지시어(compiler directives), 라이브러리 루틴(run-time library), 환경 변수(environment variable)로 구성되어 있으며, 컴파일러 지시어(complier directives)를 사용하여 프로그램의 루프 구조를 병렬로 처리하는 것이 가능하다[2]. 런타임 함수(예, omp_get_num_threads)는 프로그램이 실행 중 스레드 수, 스케줄링 방식, 내포(nesting) 레벨 등의 환경 변수를 제어할 수 있고, 임계 영역을 설정해 동기화 제어가 가능하다. 런타임 함수에서 변수제어 및 임계영역의 설정을 하지 않았을 경우 True Sharing /False Sharing 문제가 발생하는데, 이러한 문제는 동기화 오버헤드를 증가시키거나 공유변수들이 같은 캐시라인에 있을 경우 다른 코어의 캐시라인을 무효 시키는 등의 문제를 야기하게 되어 계산 성능을 저하시키는 원인이 되기도 한다[3].

2.2 병렬성능 평가모델

전산 유체역학에서 OpenMP 병렬 성능을 파악하기 위하여 본 연구실에서 개발하여 이용하고 있는 압축성 난류 연소 유동의 이차원 해석 코드를 이용하였다. 연계된 반응 유동 및 및 난류 방정식을 벡터형으로 정리하면 다음과 같다. 여기서 하첨자, $k = 1 \sim Ns$이다.

$$\frac{\partial \boldsymbol{Q}}{\partial t}+\frac{\partial \boldsymbol{E}}{\partial x}+\frac{\partial \boldsymbol{F}}{\partial y}+\frac{1}{y}\boldsymbol{H}=\frac{\partial \boldsymbol{E_v}}{\partial x}+\frac{\partial \boldsymbol{F_v}}{\partial y}+\frac{1}{y}\boldsymbol{H_v}+\boldsymbol{W} \quad (1)$$

여기서,

$$\boldsymbol{Q}=\begin{bmatrix}\rho_k\\ \rho u\\ \rho v\\ \rho e\\ \rho k\\ \rho\omega\end{bmatrix},\ \boldsymbol{E}=\begin{bmatrix}\rho_k u\\ \rho u^2+p\\ \rho uv\\ (e+p)u\\ \rho uk\\ \rho u\omega\end{bmatrix},\ \boldsymbol{F}=\begin{bmatrix}\rho_k v\\ \rho uv\\ \rho v^2+p\\ (e+p)v\\ \rho vk\\ \rho v\omega\end{bmatrix},\ \boldsymbol{H}=\begin{bmatrix}\rho_k v\\ \rho uv\\ \rho v^2\\ (e+p)v\\ \rho vk\\ \rho v\omega\end{bmatrix},$$

$$\boldsymbol{E_v}=\begin{bmatrix}-\rho_k u_k^d\\ \tau_{xx}\\ \tau_{xy}\\ \beta_x\\ \mu_k\partial k/\partial x\\ \mu_\omega\partial\omega/\partial x\end{bmatrix},\boldsymbol{F_v}=\begin{bmatrix}-\rho_k v_k^d\\ \tau_{yx}\\ \tau_{yy}\\ \beta_y\\ \mu_k\partial k/\partial y\\ \mu_\omega\partial\omega/\partial y\end{bmatrix},\boldsymbol{H_v}=\begin{bmatrix}-\rho_k v_k^d\\ \tau_{yx}\\ \tau_{yy}-\tau_{\theta\theta}\\ \beta_y\\ \mu_k\partial k/\partial y\\ \mu_\omega\partial\omega/\partial y\end{bmatrix},\boldsymbol{W}=\begin{bmatrix}\omega_k\\ 0\\ 0\\ 0\\ S_k\\ S_\omega\end{bmatrix}$$

이 코드는 Ns개의 여러 성분에 대한 질량 보존 방정식과 운동량, 에너지 방정식 및 난류 전달 방정식을 연계된 형태로 해석한다. 본 연구에 이용한 연소 반응 기구는 CO, CO_2, H_2, H, OH, H_2O, O, O_2, N_2의 9개 성분과 9개 상세 반응으로 구성되어 있다[4]. 난류 해석 모델로는 벽면 경계층에서 작은 스케일의 와동 포착에 따른 문제를 회피하기 위하여 hybrid RANS/LES 기법 가운데 Menter의 SST 모델을 기반으로 확장된 DES 모델을 이용하였다[5].

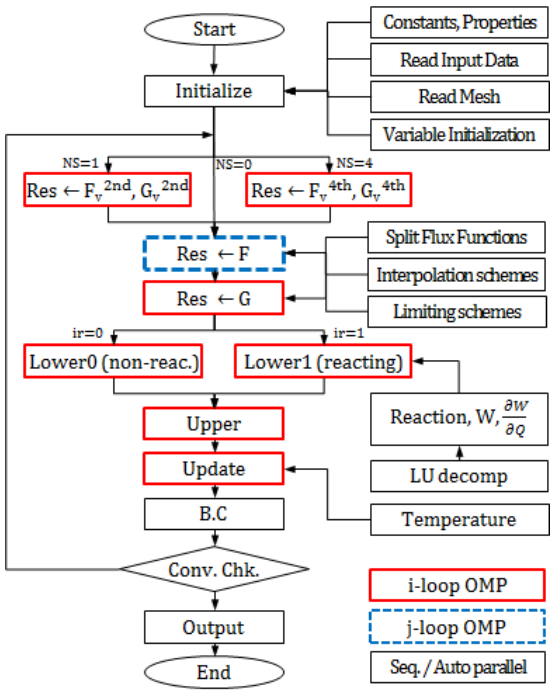

Fig. 1 The flow chart for the parallel performance evaluation of 2D reacting/non-reacting flow analysis code

유동방정식은 구조격자계에서 완전 연계된 형태로 유한체적법을 이용하여 이산화하였다. 대류 플럭스는 Roe의 플럭스 분할 기법을 이용하였으며 점성항에는 중심 차분을 이용하였다. 고차정확도 해법으로는 기존의 3차 정확도 MUSCL 기법(MUSCL3), 5차정확도의 WENO 기법 (WENO5) 및 5차 정확도 변수 외삽 방법 및 o-MLP 기법(oMLP5)을 선택적으로 이용할 수 있으며, 본 연구에서는 5차 정확도 변수 외삽 방법을 이용하였다. 시간 적분에는 2차 정확도 시간 차분 및 화학 반응 및 대류 플럭스를 연계하는 완전 음해법과 시간 반복 해법을 이용하였다. 기본 해석 코드는 이전의 초음속 연소 유동 연구들에 이용되어 검증된 바 있다[5,6].

이 코드의 개략적인 순서도를 Fig. 1에 정리하였다. 코드의 계산 순서는 점성항, 대류항, 생성항 등의 잔차항 계산 후, Lower 및 Upper 음해법 과정으로 구성되어 있으며, 각 단계의 최외각 loop가 주어진 쓰레드(thread) 수에 따라 “parallel do” 구문을 이용한 일차원적 병렬 처리되는 구조를 가지고 있다. 한편 자동 분할 된 병렬 loop에서 발생할 수 있는 데이터 중복의 문제는 “private” 및 “threadprivate” 구문을 이용하여 회피하였다.

2.3 성능평가 Hardware 및 소프트웨어 환경

본 연구에서는 전산 유체 역학 해석에 이용될 수 있는 프로세서와 쓰레드 수에 따른 성능을 비교 분석하기 위하여, 본 연구팀에서 운용하는 Windows 7 운영체제의 데스크탑 및 Windows Server 2008 R2 HPC Edition 운영체제의 서버 시스템들을 시험 대상으로 하였으며 컴파일러로는 여러 테스트에서 가장 우수한 성능을 보이는 것으로 알려진 Intel® Visual Fortran Composer XE 2013을 이용하였다[7].

Table 1에 본 연구에서 이용한 시스템들의 하드웨어 및 소프트웨어 사양을 정리하였다. 본문에서는 Table 1의 여러 시스템 가운데 대형 전산 유체 해석에 많이 이용될 수 있는 아래 5가지의 dual-CPU 시스템들의 결과에 대하여 주로 살펴보았다. 시험에 사용된 5개의 시스템 중 Windows 7 운영 체제의 Supermicro X8DAL 시스템 및 R410 시스템은 동일한 E5520 프로세서를 이용하고 있다. 따라서 이용된 프로세서는 E5520, X5650 및 E5-2670 3가지 종류이며, 중복되는 E5520의 경우 R410†을 기본모델로 삼아서 비교하도록 하였다. E5520은 Nehalem, X5650은 Westmere, E5-2670은 Sandy Bridge 제조 기술로 만들어졌으며 각각 2009, 2010, 2012년에 출시되었다. 보다 상세한 프로세서의 비교는 참고문헌[8]의 링크를 통하여 확인할 수 있다. 모든 시스템은 기본 설정이 hyper-threading (HT) On이므로 HT Off의 영향에 대한 비교는 Supermicro X8DAL에서만 수행하였다.

2.4 병렬성능 평가방법

일반적인 병렬처리 성능 평가를 위해서는 다양한 분야의 여러 문제를 이용하여 가중 평균하는 방식을 이용하지만, 본 연구에서는 전산 유체역학에서의 성능 평가를 목적으로 하고 있으므로 네 가지 유동 모델 및 세 가지 격자 수준의 조합인 12가지 경우에 대한 평균을 통하여 성능 평가를 수행하였다. 고려한 유동 모델은 압축성의 이차원 비반응/비정섬, 비반응/점성, 반응/비점성 및 반응/점성 유동이며, 각 모델은 동일한

Table 1 Systems specification used in the performance test[6], (*): HT off, (†): reference system

System	CPU	Total No. of cores	Total No. of threads	No. of runs
Supermicro X8DAL*	Xeon E5520	8	8	1
Supermicro X8DAL	Xeon E5520	8	16	1
DellTM R410† (Windows 2008 R2)	Xeon E5520	8	16	5
	2.26GHz, 8MB cache, 45nm, SSE4.2, Nehalem			
DellTM R410 (Windows 2008 R2)	Xeon X5650	12	24	5
	2.66GHz, 12MB cache, 32nm, SSE4.2, Westmere			
DellTM R720 (Windows 2008 R2)	Xeon E5-2670	16	32	5
	2.6GHz, 20MB cache, 32nm, AVX, Sandy Bridge			

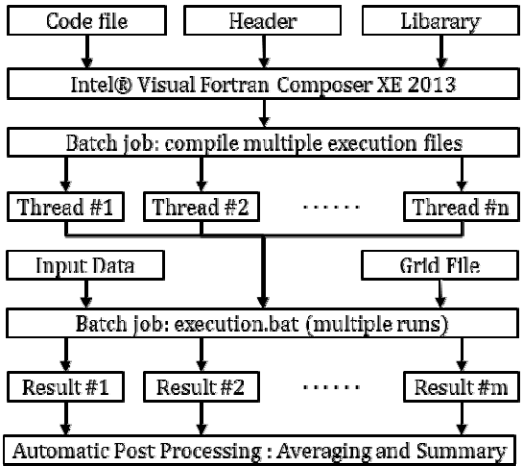

Fig. 2 The structure of parallel performance evaluation package

실행 코드에서 서로 다른 flag를 지정하여 수행할 수 있다. 해석에 사용된 세 가지 격자 수준은 96×96, 192×192, 384×384 이며 유동 모델에 따라 다소 차이가 있지만 Windows task manager를 이용하여 확인 한 결과 96×96 격자는 54~55 MB, 192×192 격자는 120~123 MB, 그리고 384×384 격자는 299~307 MB의 주 기억장치 용량을 필요로 하였다. 해석 격자에 따라 동일한 수준의 계산 시간이 소요되도록 최대 시간 전진 반복 횟수를 96×96 격자에서는 700회, 192×192 격자에서는 200회, 그리고 384×384 격자에서는 50회로 지정하였다.

쓰레드 수에 따른 계산 시간은 각 유동 모델 및 격자에서 계산에 이용되는 쓰레드 수를 1에서 시스템 최대 쓰레드 수의 2배 이상까지 증가시켜가며 "secnds" 내장 함수를 이용하여 wall-clock time을 측정하였다. 실행 코드는 /fast 옵션을 이용하여 각 시스템에서 최고 수준의 최적화를 수행하였으며, /Qopenmp 및 /Qpar-num-threads 옵션을 이용하여 계산에 사용하는 쓰레드 수가 서로 다른 다수의 실행코드를 작성하였다. 매우 많은 수의 시험을 체계적으로 수행하기 위하여 배치 파일의 형태의 시험 패키지를 구성하였으며, 시스템의 상황에 따라 1~5회 테스트를 수행 하였다. Fig. 2는 테스트 패키지의 과정과 구성을 나타낸 그림이다. 실행 쓰레드 수를 지정하지 않은 경우 Openmp 실행 코드는 최외각 loop를 디폴트값인 시스템 쓰레드 수로 분할하여 실행한다.

3. 병렬처리 성능 비교

3.1 쓰레드 수에 따른 성능 향상

위의 패키지를 이용하여 각 시스템에서 쓰레드 수에 따른 계산 수행 시간을 기록하였으며, 각 유동 모델 및 격자에 대

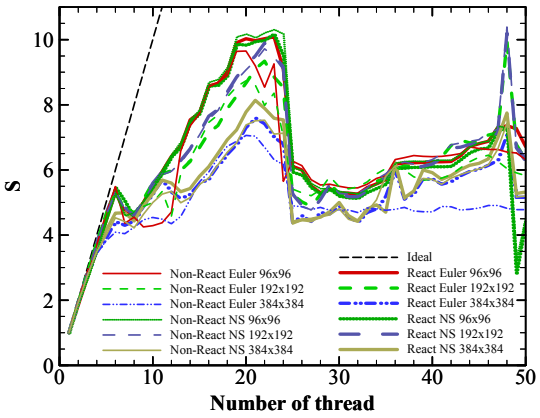

Fig. 3 The speed-ups for different grid resolutions and flow models on X5650

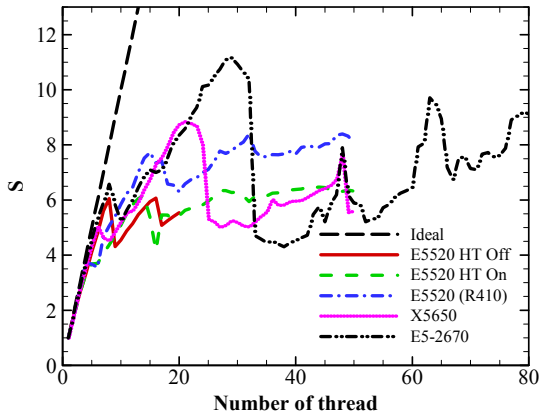

Fig. 4 The average speed-up for each system

하여 쓰레드 1개를 이용한 계산 시간을 쓰레드 N개를 이용한 계산 시간으로 나누어 쓰레드 N개를 이용한 속도향상(speed-up)을 정의하였다. 다음의 결과에서 쓰레드 수에 따른 속도향상은 별도의 설명이 없는 경우 4가지 유동 모델 및 3가지 격자 해상도에 따른 속도향상을 산술 평균한 것이다.

Fig. 3은 여러 유동 모델 및 격자 해상도에 따른 성능향상 비교 그래프로써, X5650 프로세서를 가진 R410서버 5대에서 수행한 결과를 평균한 것이다. 이 시스템은 6개 코어를 가진 2개 프로세서, 총 12개의 물리 코어/24개의 논리 코어(쓰레드)를 가진다. 이 결과에서 유동 모델 및 격자 해상도에 따라 서로 다른 속도향상 특성을 가짐을 알 수 있으며 이에 대해서는 뒤에서 별도로 논의하겠다. 유동 모델 및 격자 해상도와 관계없이 나타나는 일반적인 특징은 쓰레드 수가 6개 이하로 작은 경우에는 거의 선형적 속도 향상을 보이지만 쓰레드 수가 6 또는 12개를 지나면서 다소 성능향상이 저하되는 경향을 보이는 점, 그리고 시스템 쓰레드 수에 근접할 때까지 성능향상 기울기가 줄어들었지만 꾸준한 속도향상을 보이는 점이다. 이는 한 프로세서의 메모리 컨트롤러에 종속된 메모리의 데이터를 복제 혹은 중복사용이 아닌 공유사용이 가능하게 하는 NUMA(Non-Uniform Memory Access) 구조에 기인하는 것으로 보인다. NUMA 구조는 CPU끼리 연결된 QPI(QuickPath Interconnect)라는 통로를 통해 다른 메모리 컨트롤러가 관장하는 데이터에 직접 관여할 수 있게 하는 구조를 가지고 있기 때문에 한 개의 데이터를 동시다발적으로 입출력 하게 되면 입출력이 적을 때보다 혼잡하게 되어 성능향상이 더뎌지는 것으로 추측된다[9,10]. 계산에 이용되는 쓰레드 수가 시스템 쓰레드 수(24개)에 가까운 경우 속도 향상이 보이지 않거나 감소하는 것을 볼 수 있으며, 시스템 쓰레드 수(24개) 이상으로 증가시키는 경우 급격한 속도향상의 저하를 볼 수 있다. 그러나 쓰레드 수를 48개까지 증가시키는 경우 대체로 완만한 성능향상을 보이며, 쓰레드 수가 48인 경우, 24개와 버금가는 속도향상을 보이는 경우도 발견된다.

서로 다른 시스템에서 쓰레드 수에 따른 전반적인 성능 향상 특성을 살펴보기 위하여 유동 모델 및 격자 해상도에 따른 전체 성능향상을 평균한 결과를 Fig. 4에 정리하였다. 이 결과에서 여러 시스템에 공통적으로 나타나는 일반적인 특징은 계산에 할당된 쓰레드 수가 프로세서 당 코어 수나 쓰레드 수를 넘어서면서 속도 저하가 보이지만, 이후 쓰레드 수를 더 증가시킴에 따라 시스템 쓰레드 수에 가까워 질 때까지 완만하지만 지속적인 속도향상을 보이는 점이다. 그러나 최대 성능은 대체로 시스템 쓰레드 수보다 2 또는 3개 정도 작은 쓰레드 수에서 최고를 보이며 이후 급격한 성능 저하를 보인다. 이러한 문제는 계산 전용으로 사용되는 서버 시스템에서는 다소 작게, PC용 Windows 7 시스템에서는 다소 크게 나타나는데, 이는 운영체제의 후순위작업(background) 작업이나 서비스 등이 계산 자원을 일부 이용하기 때문에 부하 균형(load balancing)을 깨뜨리기 때문으로 여겨진다. 따라서 계산에 이용되는 쓰레드 수를 시스템 쓰레드 수 보다 다소 작게 지정하는 것이 안정된 최고 성능을 얻는 데 유리한 방법으로 판단된다.

한편, 계산에 이용되는 쓰레드 수를 최대 시스템 쓰레드 수 이상으로 지정하는 경우 E5520 시스템에서는 완만한 상승을 보이며 최대 시스템 쓰레드 수의 2배 또는 3배 정도에서 최고의 성능을 보이기도 하였다. 그러나 X5650이나 E5-2670 시스템에서는 최대 시스템 쓰레드 수 이상에서 급격한 성능 저하 후, 완만한 성능 변화가 있으며 코어 수 또는 쓰레드 수의 정수배가 되는 조건에서 두드러진 속도향상이 보이지만 최대 성능에 미치지는 못한다. 계산에 이용되는 쓰레드 수를 증가시키는 것은 explicit 해법의 경우 문제가 되지 않지만,

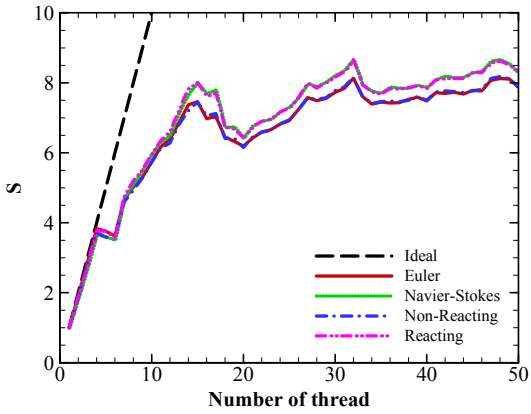

Fig. 5 The speed-up for different flow models on E5520

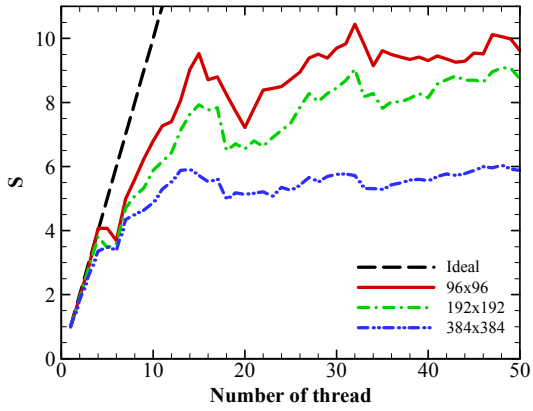

Fig. 6 Speed-up dependency on grid resolutions for E5520

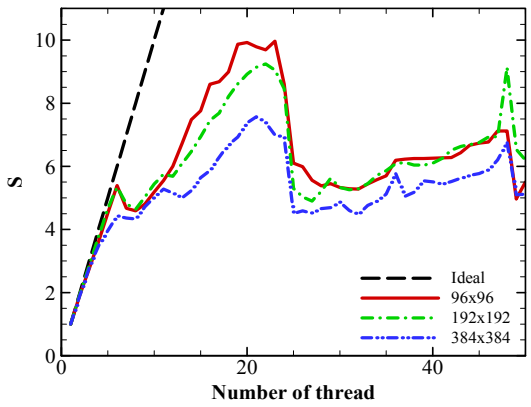

Fig. 7 Speed-up dependency on grid resolutions for X5650

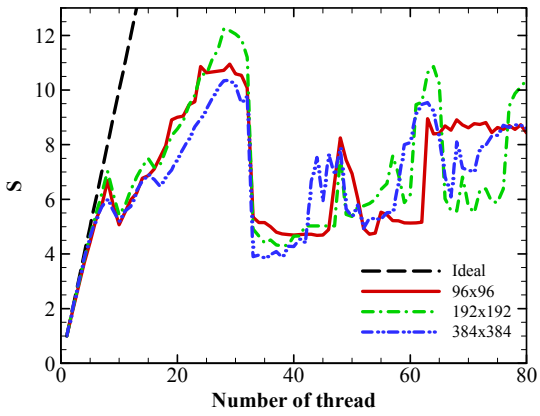

Fig. 8 Speed-up dependency on grid resolutions for E5-2670

implicit 해법의 경우에는 수렴성 저하를 야기할 수 있으므로 최대 시스템 쓰레드 수 이상으로 계산 쓰레드 수를 지정할 필요는 없는 것으로 보인다. 본 연구에서 해법의 수렴성의 문제는 고려하지 않았다.

3.2 유체 모델에 따른 병렬처리 성능

유체 모델에 따른 병렬처리 속도향상 특성을 살펴보기 위하여 E5520 시스템에서 유체 모델에 따른 속도향상 결과를 Fig. 5에 정리하였다. 이 결과에서 Euler 결과는 3가지 격자에서 비점성 반응 유동 및 비반응 유동의 해석 결과를 평균한 것이며, 다른 유체 모델의 결과도 같은 방법으로 평균 한 결과이다. 이 결과에서 비점성 유동 보다는 점성 유동 해석이, 비반응 유동 보다는 반응 유동이 좋은 성능 향상을 보이고 있어 유동 해석 모델이 복잡해지고, 쓰레드 당 계산 부하가 큰 해석에서 성능 향상이 큰 것으로 나타나지만, 전반적으로 유동 모델에 따른 성능 향상의 차이는 크지 않은 것으로 보인다.

3.3 격자 크기에 대한 의존도

병렬 처리 성능은 계산에 필요한 메모리 용량에도 영향 받을 수 있으므로, 이에 따른 영향을 살펴보기 위하여 세 가지 격자 수준에 따른 속도향상 특성을 시스템 별로 Fig. 6~8에 정리하였다. 이 결과들의 공통적인 특징은, 작은 격자 문제에서 속도향상이 크게 나타나며, 큰 격자를 이용하는 경우에는 속도향상이 작게 나타난다는 것이다. 이는 필요한 메모리가 커질수록 캐쉬 메모리 적중률이 상대적으로 저하되며[11], QPI 를 통한 프로세서 사이의 데이터 교환이 늘어나기 때문으로 여겨진다[9,10].

한편 최신의 프로세서일수록 시스템 쓰레드 수 이하에서 격자 해상도에 따른 영향은 작게 나타나며, 시스템 쓰레드 수 이상에서 급격한 성능 저하가 나타난다. 이는 프로세서의 작동속도는 큰 차이가 없지만, 메모리의 작동 속도 증가와 더불어, 프로세서의 데이터 버스의 속도 증가는 물론, 코어당 데이터 버스의 수가 현저히 증가되어 전체적인 데이터 교환

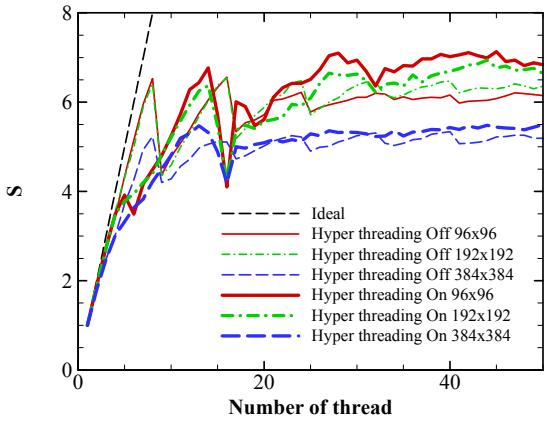

Fig. 9 Speed-up dependency on HT for E5520

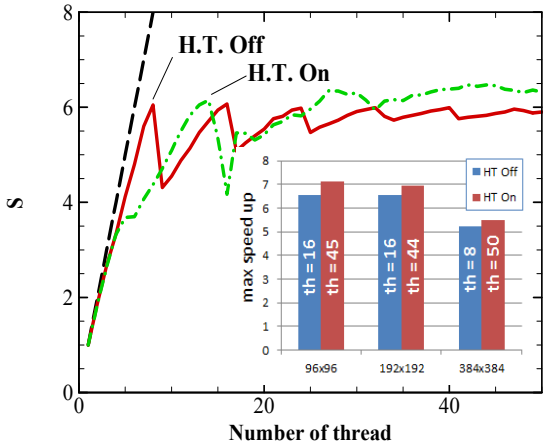

Fig. 10 Effects of HT on/off on E5520

속도가 상당히 증가하기 때문으로 여겨진다. 또한 시스템 쓰레드 수 이상에서는 계산 자원은 제한된 반면 데이터 교환이 늘어나 성능향상 저하요인으로 작용하는 것으로 보인다. 즉 최신의 프로세서 구조가 프로세서의 잠재적인 성능을 문제 크기의 영향 없이 최대한 이끌어 내도록 구성된 것으로 생각된다[8].

3.4 Hyper-threading (HT)의 효과

Hyper-threading은 프로세서의 유휴 자원을 최대한 이용하기 위하여 1개의 물리적 코어를 2개의 쓰레드로 나누어 이용하는 Intel의 기술로써, 사용 가능한 경우 대부분의 시스템 출고시 On으로 설정되어 있으나, 소프트웨어 라이센스 비용 등의 문제로 많은 경우 Off로 설정하여 사용하기도 한다. 각 격자 해상도에 대한 결과 그래프를 Fig. 9에 나타내었다. 각 격

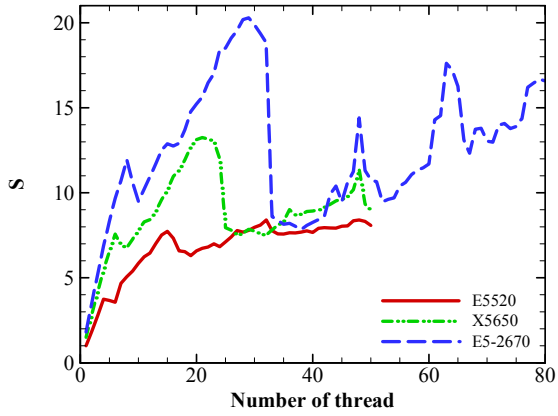

Fig. 11 Comparison of speed-up for different susystems

자 별로 성능향상의 차이는 보이지만 대체로 일관된 경향을 보여 주었다. Fig. 10은 Xeon E5520에서 HT On/Off 성능을 평균하여 비교한 그림이다. HT Off일 때는 물리 코어 수의 정수배 조건에서 최대 성능향상이 얻어지며, 그 직후 성능향상은 일정한 정도로 수렴함을 알 수 있다. Off의 경우 최대 코어 수 이후, 부하 균형의 문제로 성능이 감소하다가 쓰레드 수가 최대 코어 수의 2배가 될 때까지 다시 증가한다. HT On의 경우 HT Off에 비해서 초기 속도향상은 낮으나 15 쓰레드까지는 계속 증가한다. 이후 부하 균형의 문제로 다시 감소하다가 대체로 일정한 정도의 높은 속도 향상을 보임으로써 HT Off와 경우와 비슷하거나 다소 향상된 성능을 보였다. HT On/Off의 최대 성능향상을 막대그래프로 표시하였으며, Off의 경우보다는 On의 경우가 10% 정도의 향상된 성능을 보이지만, 기대만큼 차이는 크지 않음을 알 수 있다. 이는 컴파일러 기술이 향상되어 HT를 사용하지 않더라도 계산 자원을 상당히 활용할 수 있도록 최적화된 실행코드를 작성하기 때문으로 여겨진다. 한편, E5520 시스템에서 최대 성능향상을 얻은 쓰레드 수는 시스템 쓰레드 수의 정수배보다 다소 작은 쓰레드 수인 경우가 종종 보이는데, 프로세서 사이의 데이터 교환을 위한 버스 수와 속도가 작고 메모리 속도가 작아서 나타나는 문제로 보이며, 보다 최신의 프로세서에서는 나타나지 않기 때문에 최대의 성능향상을 얻기 위하여 시스템 쓰레드 수 이상으로 계산 쓰레드 수를 지정하는 것은 좋은 방법으로 보이지 않는다.

3.5 프로세서 간의 성능 향상 특성 비교

이 절에서는 비교에 이용된 시스템 가운데 가장 먼저 출시되어 상대적으로 낮은 성능을 가지고 있는 E5520의 쓰레드 하나를 이용한 계산시간을 각 프로세서의 쓰레드 개수에 따른 계산시간으로 나누어 Fig. 11에 정리하였다. 이 그림은 프

로세서 간의 성능향상을 특성을 보여주어, 높은 사양의 최신 프로세서가 절대적으로 우위에 있는 성능 향상 특성을 보여줌을 알 수 있으며 이는 물리 코어수의 증가와 함께 쓰레드 당 성능이 향상되었기 때문에 나타나는 자연스러운 결과로 보인다. 각 프로세서의 성능을 좀더 쉽게 비교할 수 있도록 최고 속도향상 조건의 결과를 Table 2에 정리하였다.

Table 2의 결과에서 OpenMP 병렬처리에 의한 성능이 쓰레드 수 보다는 코어 수에 더 의존하는 경향을 보이며, 이는 앞 절에서 HT의 효과가 크지 않았던 이유에 기인한다. 각 시스템의 속도 향상은 8 코어의 E5520 시스템에서 7.73배, 12 코어의 X5650 시스템에서 8.84배, 16 코어의 E5-2670 시스템에서 11.16배로, 코어 수 증가에 따른 병렬처리 효과는 줄어드는 경향을 보인다. 한편 E5520과 비교하여 상대적인 최대 성능향상비가 X5650에서는 13.25배, E5-2670에서는 20.29배까지 향상되어, 전체 시스템의 성능은 1.71배 및 2.62배까지 증가하였으나, 이는 작동 클럭 수의 증가 및 쓰레드 당의 성능 향상에 크게 영향 받은 결과이며, 코어 수의 증가에 따른 영향은 상대적으로 작은 것으로 보인다. 한편 최대 성능을 얻은 쓰레드 수는 기본 값인 시스템 쓰레드 수보다 1~3개 작은 경우에 얻을 수 있었다. 따라서 멀티코어 SMP 시스템에서 OpenMP에 의한 최대의 성능 향상을 얻기 위해서는 시스템 쓰레드 수보다 다소 작게 쓰레드 수를 지정하는 것이 효과적인 이용 방법임을 알 수 있다.

4. 결 론

본 연구에서는 서로 다른 점성/비점성, 반응/비반응 유동 등 서로 다른 유체 모델에 대하여 서로 다른 수준의 격자계를 이용하여 OpenMP 병렬처리 성능을 살펴보기 위한 시험 패키지를 구성하였으며, 여러 종류의 전산 시스템에서 계산에 이용되는 쓰레드 수를 시스템 쓰레드 수의 두 배 이상으로 강제하여 계산 성능 향상을 살펴보았다. 이 시험에 의한 성능 향상 결과로부터 시스템의 종류, 유동 모델, 격자 크기 및 HT 등의 영향에 대하여 살펴보았으며, 이로부터 다음과 같이 몇 가지 특징적인 결론을 얻을 수 있었다.

Table 2 Processor specifications and the best speed-up characteristics

	E5520	X5650	E5-2670
No. of core/1 CPU	4	6	8
No. of system core	8	12	16
No. of system thread	16	24	32
Performance/1 thread	1.00	1.50	1.83
Max. performance	7.73	8.84	11.16
Relative max. performance	7.73	13.25	20.29
Relative system performance	1.00	1.71	2.62
No. threads for max. performance	15	21	29

(1) 병렬처리에 이용하는 쓰레드 수를 기본 값인 시스템 쓰레드 수보다 다소 작게 지정하는 것이 안전하게 최고의 속도 향상을 얻는 방법임을 알 수 있었다. 시스템 쓰레드 수 이상을 이용하는 경우 급격히 성능이 저하하며, 이는 최신 프로세서일수록 더욱 두드러진다.

(2) 동일한 시스템 환경에서 HT를 이용하는 것이 최대 8.73% 더 좋은 성능 향상을 얻을 수 있어 HT의 효과가 크지는 않은 것으로 여겨진다.

(3) 메모리 용량이 적게 필요한 작은 격자 문제에서 보다 나은 성능 향상을 얻을 수 있었으나, 최신의 프로세서 일수록 격자의 크기나 메모리 용량에 의한 차이는 줄어드는 것으로 보인다.

(4) 복잡한 유동 모델일수록 병렬처리 성능 향상이 크게 나타났으나, 유동 모델에 따른 차이가 크지는 않았다.

(5) 전체 시스템의 성능 향상은 프로세서의 발전에 따른 쓰레드 당의 성능 향상, 작동 클럭 수의 증가 및 코어 수의 증가가 복합적으로 나타난 결과이다.

후 기

이 논문은 부산대학교 자유과제 학술연구비(2년)에 의하여 연구되었습니다.

참고문헌

[1] Intel® Xeon Phi™ Coprocessor 5110P, http://www.intel.co.kr/content/www/kr/ko/processors/xeon/xeon-phi-detail.html.

[2] 2011, OpenMP Application Program Interface, Version 3.1 OpenMP Architecture Review Board, http://www.openmp.org/mp-documents/OpenMP3.1.pdf.

[3] 2012, Intel® 64 and IA-32 Architecture Software Developer's Manual Vol.1,2,3, Intel Corporation, http://www.intel.com.

[4] 1994, Singh, D.J. and Jachimowski, C.J., "Quasigolbal Reaction Model for Ethylene Combustion," *AIAA Journal*, Vol.32-1, p213.

[5] 2009, Shin, J.-R., Moon, S.-H., Won, S.-H. and Choi, J.-Y., "Detached Eddy Simulation of Base Flow in Supersonic Mainstream," *Journal of KSAS*, Vol.37-10, p.955.

[6] 2005, Choi, J.-Y., Yang, V. and Ma, F., "Combustion Oscillations in a Scramjet Engine Combustor with Transverse Fuel Injection," *Proc. Combust. Inst.* Vol.30, p. 2851.

[7] 2012, Fortran Compiler Comparisons, Polyhedron Software, http://www.polyhedron.com/compare0html.

[8] Intel Processor Comparison, http://ark.intel.com/compare/40200,47922,64595.

[9] 2010, Balakrishnan, G., Begun, R.M. and Kochuparambil, B., Understanding Intel Xeon 5600 Series Memory Performance and Optimization in IBM System x and BladeCenter Platforms, IBM, http://public.dhe.ibm.com/common/ssi/ecm/en/xsw03075usen/XSW03075USEN.PDF.

[10] 2009, An Introduction to the Intel® QuickPath Interconnect, Intel, http://www.intel.com/content/dam/doc/white-paper/quick-path-interconnect-introduction-paper.pdf.

[11] 2002, Choi, J.-Y. and Oh, S., "An Acceleration Scheme of LU-SGS Code on Latest Microprocessors Considering Increased Hit-ratio of Level 2 Cache," *Journal of KSAS*, Vol.30-7, p.68.