OpenMP와 MPI 코드의 상대적, 혼합적 성능 고찰

이 명 호†

요 약

최근의 고성능 컴퓨팅 플랫폼들은 공유 메모리 다중 프로세서(SMP: Shared Memory Multiprocessor) 시스템, 대규모 병렬 프로세서(Massively Parallel Processor) 시스템, 여러 개의 컴퓨팅 노드들을 연결한 클러스터(Cluster) 시스템 등으로 분류된다. 이러한 고성능 컴퓨팅 시스템들은 높은 수준의 컴퓨팅 성능을 요구하는 과학 기술용 응용 프로그램들을 위하여 사용된다. 이러한 응용 프로그램들의 실행시 최적의 성능을 얻기 위해서는 적절한 컴퓨팅 플랫폼과 프로그래밍 방식의 선택이 중요하다. 본 연구 논문에서는 여러 방식의 병렬 프로그래밍 모델을 사용하여 개발된 SPEC HPC2002 벤치마크 suite을 위한 최적의 컴퓨팅 플랫폼과 프로그래밍 모델을 그들의 성능 분석 및 평가 작업을 통하여 찾아간다.

키워드 : 고성능 컴퓨팅, MPI, OpenMP, 하이브리드 병렬 프로그래밍, 클러스터, 공유 메모리 다중 프로세서

Comparative and Combined Performance Studies of OpenMP and MPI Codes

Myungho Lee†

ABSTRACT

Recent High Performance Computing (HPC) platforms can be classified as Shared-Memory Multiprocessors (SMP), Massively Parallel Processors (MPP), and Clusters of computing nodes. These platforms are deployed in many scientific and engineering applications which require very high demand on computing power. In order to realize an optimal performance for these applications, it is crucial to find and use the suitable computing platforms and programming paradigms. In this paper, we use SPEC HPC 2002 benchmark suite developed in various parallel programming models (MPI, OpenMP, and hybrid of MPI/OpenMP) to find an optimal computing environments and programming paradigms for them through their performance analyses.

Key Words : High Performance Computing, MPI, OpenMP, Hybrid Parallel Programming, Cluster, SMP

1. 서 론

최근의 고성능 컴퓨팅(High Performance Computing: HPC) 플랫폼들은 공유 메모리 다중 프로세서(Shared-Memory Multiprocessor: SMP) 시스템, 대규모 병렬 프로세서(Massively Parallel Processor) 시스템, 여러 개의 컴퓨팅 노드들을 고성능 인터커넥트(High-Speed Interconnect)를 이용하여 연결한 클러스터(Cluster) 시스템 등으로 자리 잡아가고 있다. 클러스터 시스템의 개발은 원래 여러 대의 개인용 워크 스테이션 이나 컴퓨터들을 네트워크로 연결하여 사용하면서 보편화되기 시작했는데, 최근 들어 기본 컴퓨팅 노드로서 소형 SMP 시스템을 사용하는 것이 추세가 되어가고 있다. 이러한 HPC 플랫폼들은 그 응용 분야에 따라서 최적의 플랫폼들이 선택 되어진다.

† 정 회 원 : 명지대학교 컴퓨터소프트웨어학과 조교수
논문접수 : 2006년 2월 23일, 심사완료 : 2006년 3월 10일

HPC 시스템들을 프로그램 하는 방식은 크게 공유 메모리(Shared-Memory)를 사용하거나, 메시지를 전달(Message Passing)하는 방식으로 나눌 수 있다. OpenMP[5]는 기존의 응용 프로그램을 상당히 적은 코딩 노력만으로 쉽게 병렬화 시킬 수 있는 장점을 보유하고 있으며, 그로 인해 상당히 빠른 시일 안에 SMP용 응용 프로그램 개발 방식의 표준으로 자리 잡게 되었다. 메시지 전달 방식으로 응용 프로그램을 병렬화 하는 표준에는 MPI(Message Passing Interface)[4]가 있다. OpenMP 프로그래밍 방식과 비교해서 MPI 프로그래밍은 사용자에게 훨씬 많은 코딩 노력을 요구 하지만, 성능과 확장성 면에서 OpenMP보다 일반적으로 우수하다. MPI 프로그래밍 방식은 또한 SMP 시스템에도 바로 적용할 수 있다는 장점이 있다. 최근의 병렬 프로그래밍 환경 개발의 추세는 (1) OpenMP의 편리성을 MPP나 클러스터에까지 확장시키기 위한 노력들(OpenMP over clusters)과 (2) OpenMP와 MPI를 혼합하여 프로그래밍 하는 방식(하이브리드 병렬 프

로그래밍)[1, 2] 등이 있다.

높은 컴퓨팅 성능을 요구하는 과학 기술용 응용 프로그램에서 최적의 컴퓨팅 플랫폼과 그에 맞는 병렬 프로그래밍 방식을 찾아내어 적용하는 것은 응용 프로그램들의 성능 요구사항들을 만족시키는 데 있어서 매우 중요하다. 본 논문에서는 다양한 병렬 프로그래밍 모델들(MPI, OpenMP, 하이브리드)을 사용할 수 있도록 개발된 과학 기술용 응용 프로그램 suite인 SPEC HPC2002[6]를 사용하여, 그들의 최적 실행 환경(대형 SMP 또는 SMP 클러스터) 및 프로그래밍 방식을 성능 분석을 통하여 찾아 나가는 연구를 수행한다.

본 논문의 나머지 부분은 다음과 같이 구성되었다. 2장에서는 본 논문의 연구를 위하여 사용된 실험용 컴퓨팅 플랫폼들(Sun Microsystems사의 Sun Fire 서버들과 그들을 네트워크로 연결한 클러스터들)과 사용된 소프트웨어 등에 대한 자세한 내용들을 설명한다. 3장에서는 세 개의 SPEC HPC 2002 벤치마크들에 대하여 설명하고, 이 벤치마크들을 병렬화하는 방법론, 그리고 벤치마크들의 간단한 성능 특성에 대해서 설명한다. 4장에서는 여러 시스템에서 측정한 HPC 2002 벤치마크들의 성능 결과를 보여준다. 5장에서는 본 연구 논문에서 얻은 결론을 서술한다.

2. 시스템 개요

본 논문의 연구를 위하여 사용되는 Sun Micorsystems사의 제품들은 UltraSPARC III Cu 프로세서와 Fireplane 인터커넥트를 기본으로 하는 소형, 중형, 그리고 대형 서버들이다. 이런 시스템들은 서로 다른 네트워크 기술을 사용하여 연결되기도 한다. 본 논문의 실험 결과를 얻기 위해 다음과 같은 세 가지의 시스템들이 사용되었다:

• 시스템 1

72개의 UltraSPARC IIICu 프로세서(clock speed가 900Mhz)들로 구성된 Sun Fire 15K[8] 서버로서 전체 메모리 용량은 144GB.

• 시스템 2

8개의 Sun Fire V880[9] 서버들을 Myrinet 인터커넥트로 연결한 시스템으로, 각각의 Sun Fire V880 서버는 8개의 UltraSPARC IIICu 프로세서(clock speed 가 1050Mhz) 들로 구성되어 있으며 메모리 용량은 16GB. 시스템 전체로는 64개의 프로세서가 사용되며, 총 메모리 용량은 128GB이다.

• 시스템 3

3개의 Sun Fire 6800[7] 서버들을 Sun Fire Link 인터커넥트로 연결한 시스템으로서 각각의 Sun Fire 6800 서버는 24개의 Ultrasparc IIICu 프로세서(clock speed가 900Mhz) 들로 구성되어 있으며 메모리 용량은 96GB. 시스템 전체로는 72개의 프로세서가 사용되며, 총 메모리 용량은 288GB이다.

Sun Fire 15K와 6800 서버들을 구성하는 기본 컴포넌트는 UniBoard이다. 각 UniBoard는 네 개의 UltraSPARC IIICu 프로세서들과 메모리로 구성 되어있다. 각각의 UltraSPARC IIICu에는 메모리 컨트롤러, level-2 (L2) 캐시(cache)를 위한 캐시 태그(tag)를 포함하고 있다. L2 캐시는 chip 외부에 있으며 크기가 8MB이다. UniBoard내의 프로세서들과 메모리의 연결은 Fireplane System Interconnect를 이용하며, 시스템 버스의 clock은 150Mhz이다.

Sun Fire 6800 서버는 전형적인 공유 메모리 다중프로세서 시스템으로서 중형(mid-range) 서버로 설계되었다. 최대 6개의 UniBoard 들로 구성되며. 시스템 전체적으로 캐시 일관성을 유지하기 위해 스누핑 캐시 유지 프로토콜을 사용한다.

Sun Fire 15K는 대형(high-end) 서버로 설계되었으며 최대 18개의 UniBoard 들로 구성된다. 6800서버에 비하여 상당히 용량이 큰 서버인데, 시스템 전체의 캐시 일관성을 유지하기 위해 같은 UniBoard내 에서는 버스기반의 스누핑 캐시 유지 프로토콜을 사용하며, 다른 UniBoard간에는 디렉토리(directory) 기반의 캐시 일관성 유지 프로토콜을 사용한다.

V880 시스템은 프로세서 8개가 사용되는 소형 서버로 설계되었으며, Sun Fire 6800 이나 15K와는 다른 보드(board)가 사용된다. 각 보드 당 2개의 프로세서가 사용되며, 시스템 전체에 4개의 보드를 삽입할 수 있도록 설계되었다. 시스템 내의 캐시 일관성 유지는 스누핑 캐시 유지 프로토콜을 사용한다.

<표 1>에서는 각 시스템의 MPI 대기시간(latency)과 대역폭(bandwidth)을 보여준다. MPI 대기시간과 대역폭을 구하기 위해 한 방향으로 데이터를 보내고(MPI SEND 함수 이용) 받는(MPI RECV 함수 이용) 시간차를 측정하는 Fortran 프로그램을 사용하였다.

<표 2>는 메모리 대기시간과 대역폭을 나타낸다. 메모리 대기시간은 lmbench 벤치마크의 lat_mem_rd() 함수를 사용하여 측정하였다. 메모리 대역폭은 Stream 벤치마크의 COPY 함수를 사용하여 측정하였다.

〈표 1〉 시스템별 MPI 대기 시간과 대역폭

시스템	인터커넥트	MPI 대기시간 (μsec)	MPI 대역폭 (MB/sec)
System 1	Backplane	4.1	600
System 2	Myrinet	14	130
System 3	Sun Fire Link	5.5	520

〈표 2〉 시스템별 메모리 대기 시간과 대역폭

		System 1	System 2	System 3
Memory 대기시간 (nsec)	Local access	250	205	230
	Remote access	470	205	270
Memory 대역폭 (MB/sec, N cpu's)	1	2013	1651	2356
	4	3294	5589	4491
	8	6525	8490	6934
	16	12909		9239
	24			9357
	32	25889		

위의 시스템들은 다음과 같은 소프트웨어 환경을 갖추고 있다:

- 운영체제는 Solaris 9[10]을 사용하였다.
- 벤치마크 프로그램들을 컴파일하기 위해 Sun Studio 9 컴파일러 suite[11]을 활용 하였다.
- 성능 분석을 위하여 Sun Studio 9의 performance analyzer[11]를 사용하였다.
- MPI code를 실행시키기 위해 Sun HPC Clustertools 5.0을 사용하였다.

3. SPEC HPC2002 벤치마크 Suite

본 논문에서 SPEC HPC2002 벤치마크 suite을 사용하여 앞에서 언급한 세 개의 시스템 상에서 실험을 수행하였다. SPEC HPC2002는 세 가지의 벤치마크들로 구성되어 있다: CHEM 2002, ENV 2002, 그리고 SEIS 2002. CHEM 2002는 Gamess라고 불리는 양자 화학용 응용 프로그램이다. ENV 2002는 Wrf라고 불리는 기상예측용 응용 프로그램이다. SEIS 2002는 Seis 라고 불리는 지진 탐지용 응용 프로그램이다. 각각의 벤치마크에는 세 가지 크기의(small, medium, large) 데이터 세트가 있는데, 본 논문에서는 중간 크기(medium)의 데이터 세트를 사용하여 실험을 수행한다. 이어지는 절 들에서는 각각의 벤치마크에 대한 설명과 병렬화하는 방법, 그리고 기본적인 성능 특성에 대하여 알아본다.

3.1 Gamess

Gamess는 양자화학 코드이다. Gamess는 네 가지의 다른 모드로 실행 된다: 순차적 실행 모드, OpenMP 모드, MPI 모드, 그리고 하이브리드 모드(MPI 와 OpenMP를 함께 사용하여 병렬화 한 모드). MPI를 사용한 병렬화는 마스터-슬레이브 모델을 기반으로 한다. 마스터 MPI 프로세스는 작업을(반복되는 루프들의 세트) 각각의 슬레이브 프로세스들로 보내어질 조각으로 나눈다. 그러면, 각각의 슬레이브 프로세스는 마스터 프로세스에게서 받은 작업을 실행하게 된다.

마스터 프로세스의 실행은 다음과 같이 요약할 수 있다:

```
while (num_iter > 0) {
    MPI_recv (recpt);
    chunk = num_iter / (num_proc * 2);
    work[2] = work[1] + chunk;
    num_iter -= chunk + 1;
    MPI_send (work, recpt);
    work[1] = work[2] + 1;
```

수행해야 할 작업이 있는 동안, 마스터 프로세스는 작업을 요청하는 슬레이브 프로세스를 기다린다. 요청을 받으면 남아있는 작업을 2*N(N = 슬레이브 프로세스의 숫자)으로 나누어 슬레이브 프로세스로 그만큼 크기의 작업을 보낸다.

슬레이브 프로세스에서는 반복해서 작업 요청을 마스터에게 보내며, 작업이 주어지면 계산을 수행한다. 계산을 마치고 나면 시스템 전체적으로 barrier 연산을 통해 동기화를 시킨 후, 계산된 값을 정리(reduce 연산)하여 최종 결과를 산출한다. 이러한 정리 작업에 마스터 프로세스는 참가하지 않는다. 슬레이브 프로세스의 실행은 다음과 같이 요약할 수 있다:

```
while (TRUE) {
    lp_next(istart, iend);
    if (istart == -1) break;
    for (i=istart; I <= iend; ++i)
        compute for iteration no i;
}
MPI_barrier ();
MPI_reduce ();
MPI_broadcast ();
```

32 MPI 프로세스를 사용한 실험 성능 측정에서, MPI 함수를 수행하는데 드는 시간은 총 계산 시간의 37%를 차지한다. 그 중 18%는 MPI reduction연산에 사용되었고, 14%는 MPI barrier에, 그리고 5%는 MPI broadcast와 MPI receive에 사용되었다.

OpenMP를 사용한 병렬화 작업은 대부분 루프 수준에서 수행된다. Gamess 벤치마크에는 두개의 주요 OpenMP 병렬 루프가 있다. 20개의 프로세스를 사용한 경우 이 두 개의 루프 실행에 전체 실행시간의 66%가 소요된다. 그들은 모두 OpenMP dynamic 스케쥴링을 사용한다. 예를 들면, 다음과 같이 사용된다:

```
c$omp do schedule (dynamic, 20)
```

하이브리드 병렬은 MPI 모델과 OpenMP 모델을 결합시킴으로써 완성된다. MPI 코드와 OpenMP 코드에 대한 분석을 결합하면 하이브리드 코드를 분석할 수 있게 된다.

3.2 Wrf

Wrf는 중간 규모의(mesoscale) 기상 예측용 모델을 이용한 응용 프로그램이다. Gamess와 마찬가지로 네 가지의 다른 모드로 실행될 수 있다(순차적, OpenMP, MPI, 하이브리드). MPI 병렬화는 2차원적 데이터 분할을 사용하여 수행된다. 먼저, 각각의 MPI 프로세스는 2차원의 지역(local) 데이터를 이용하여 계산을 한다. 경계에 위치한 데이터들은 스텐실(stencil) 패턴으로 교환되는데, Argonne 국립 연구소에서 개발된 일반화된 스텐실 패키지를 사용하여 데이터 교환 코드가 작성된다. 스텐실 패키지는 비동기화(asynchronous) 메시지를 사용하여 데이터를 교환시키는데, 데이터 교환이 완전히 끝날 때까지 기다리는 MPI_wait 함수를 사용한다.

20개의 프로세서를 사용한 경우, 총 계산 시간의 대략 15% 정도는 MPI 함수를 실행하는데 사용되었다: 12.5%는 데이터 교환으로 사용되었고, 나머지 MPI 시간은 프로그램 시작에서 초기화 데이터를 broadcasting 하는데 사용되었다.

OpenMP를 사용하여 병렬화된 코드는 solve_em 함수에 집중되어 있다. 20개의 프로세서를 사용한 경우, 사용자 cpu 사이클의 99.5%는 이 함수에 사용되었다. 사용자 cpu 사이클의 91%는 이 함수의 병렬 루프에 사용되었다. 그러나 총 cpu 사이클의 14.8%는 반복 루프를 끝내고 나머지 쓰레드들이 끝날 때까지 barrier에 대기해야 하는 쓰레드들에 의해 사용되었다. 이러한 부하 배분 불균형 때문에 이 프로그램의 확장성이 제한된다.

Gamess와 마찬가지로 하이브리드 병렬화는 단순히 MPI 모델과 OpenMP 모델을 결합시킨 것이다.

3.3 Seis

Seis는 지진 탐지용 응용 프로그램이다. Gamess와 Wrf와는 달리, 실행 가능한 모드가 순차적, OpenMP, 그리고 MPI 모드들이다(하이브리드 모드의 실행이 불가능). 동일한 실행 파일을 네 가지의 다른 데이터 세트를 사용하여 각각 네 번 실행하는데, 다른 데이터 세트의 실행들은 전혀 다른 프로파일 결과를 보인다. 첫 번째와 네 번째 실행은 MPI(93%) 와 OpenMP(97%) 양쪽에서 대부분의 실행 시간을 차지하는 것으로 나타난다. 그러므로 두 번째와 세 번째 실행은 성능 분석 과정에서 무시하게 된다(두 번째와 세 번째 실행들이 많은 프로세서를 사용한 계산에서 중요할 수도 있는 반면, 현재 Seis 벤치마크는 32개 이상의 cpu를 사용할 경우 정상적 실행에 문제가 발생한다).

MPI 버전의 경우 첫 번째 데이터의 실행 시 16 개의 프로세서를 사용한 실행에서 OpenMP 실행보다 4배 이상 속도가 느리다. Sun Studio 9의 performance analyzer의 타임 라인(time line)을 이용한 검사 결과 실질적으로 모든 시간이 read() 시스템 콜에 소요된다는 것을 보여준다. 모든 MPI 프로세스들이 동시에 동일한 파일을 읽기 때문이다. 이것이 운영체제에서 병목현상을 유발한다. 네 번째 데이터의 실행에서 관찰된 바로는, MPI 병렬 방식과 OpenMP 병렬 방식은 동일한 수준의 성능에 이른다. OpenMP는 대부분의 계산에 사유 변수(private variable)가 사용된 SPMD 스타일로 코딩되어졌다. 데이터는 할당된 공유 버퍼를 사용하여 쓰레드간에 이동된다. MPI의 경우, 데이터는 메시지 전달을 사용하여 프로세스간에 이동이 된다. 그러므로, Seis의 네 번째 데이터의 실행은 SMP 시스템에서 SPMD 방식의 SMP용 프로그래밍과 MPI 프로그래밍을 비교하는 방법을 제공한다.

MPI 버전의 경우, 16 프로세서를 사용하여 실행할 경우 6% 정도의 시간이 MPI 함수 실행에 소요된다. 이 시간의 대부분은 점-대-점(point-to-point) 메시지 전달에 사용된다.

4. 성능 측정과 결과

앞의 장에서 소개한 SPEC HPC2002 벤치마크들을 세 가지의 Sun Fire 시스템들에 실행하여 성능을 측정하였다. 모든 성능 측정은 프로그램의 총 실행 시간을 기준으로 한다. 프로세서의 속도가 다른 시스템들 간에 성능을 비교하기 위하여, 1050Mhz cpu를 이용한 실행 시간들은(Sun Fire V880 서버의 경우) 900Mhz로 실행되는 cpu의 실행 시간으로 변환한다. (단순히 실행 시간에 1050/900을 곱한다.) 여러 개의 프로세서를 사용하여 병렬 환경에서 실행한 경우, 그 실행 시간을 프로세서 하나만을 사용하여 순차적으로 실행한 경우의 실행 시간과 비교하여 speed-up을 구한다. 이러한 방식으로 계산된 숫자들은 순차적 버전과 비교한 속도 향상을 나타낸다.

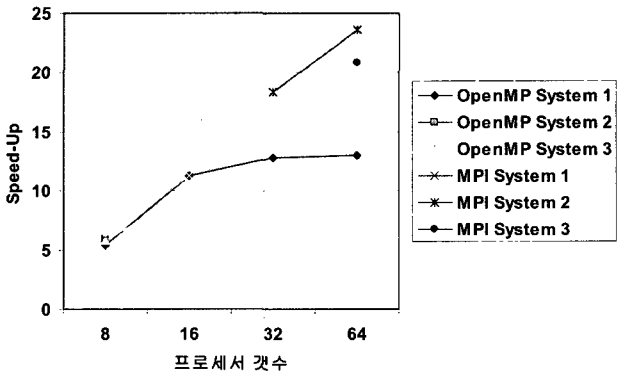
4.1 Gamess

(그림 1)은 OpenMP 와 MPI 모드로 세 가지 시스템에서 실행한 Gamess의 성능을 나타낸다. OpenMP는 하나의 노드에서만 실행 할 수 있으므로, 그림에 몇 개의 빈 칸이 남게 되었다. MPI의 경우는 시간의 제약(machine access)때문에 모든 조합을 실행시킬 수 없었다.

MPI 코드의 확장성은 OpenMP보다 훨씬 뛰어나다. 병렬성의 수준 때문에 그러하다. OpenMP 병렬성은 루프 수준에 머물러 있어서 프로세서 당 작업의 양이 루프 반복의 수에 의해 제한된다. System 2의 경우, 낮은 메모리 대기시간은 Myrinet 인터커넥트의 높은 MPI 대기시간을 상쇄한다. 그러므로 System 1과 2는 실제적으로 실행 시간이 같게 된다. System 3는 MPI와 OpenMP 양쪽 다 성능이 저조하다. System 1과 비교했을 때, 제한된 메모리 대역폭 때문이다.

<표 3>은 M 프로세스 x N 쓰레드의 다양한 조합에 대한 Gamess의 하이브리드 모드의 성능을 나타낸다. MxN으로 표시한 표는 Sun Clustertools5.0 의 "mprun" 코맨드를 각각 사용하여 M 프로세스와 N 쓰레드를 실행한 결과이다. 1+Nx4로 표시한 표는 마스터 프로세스(3.1절 참조)로서 하나의 여분의 프로세스를 추가하여, N 프로세스가 각각 4개의 쓰레드를 실행한 결과이다. 또한 1+Nx4의 경우에서는 각각 슬레이브 프로세스에서 4개의 쓰레드가 4개의 프로세서가 있는 Uniboard에 바인딩(binding) 된다.

Gamess는 하이브리드 모드에서 가장 뛰어난 성능을 보였다. 적은 수의 MPI 프로세스들 덕분에 로드 불균형이 줄어들었고, 적은 수의 쓰레드들에 대해서 OpenMP 병렬처리 방식이 적합하기 때문이다. 마스터 프로세스에 대하여 추가적으로 쓰레드를 지정하고 Uniboard에 바인딩하는 것은 추가적인 성능을 얻는데 도움이 된다. System 3의 경우 성능이



(그림 1) Gamess 의 MPI 와 OpenMP 성능

〈표 3〉 Gamess 의 하이브리드 성능

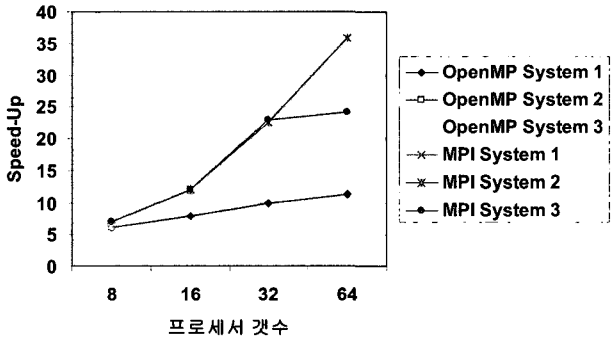
사용된 프로세서 개수	64		32			16			8
조합	1+ 16x4	8x8	1+ 8x4	8x4	4x8	1+ 4x4	8x2	4x4	4x2
System 1	35.9	31.0	21.4	18.3	15.2	11.8	10.5		4.9
System 2		33.3		19.1	16.5		10.3	9.1	4.9
System 3	32.7		20.8						

비교적 처지는 이유는 제한된 메모리 대역폭이 병목으로 작용하기 때문이다. 그러나 제한된 메모리 대역폭에도 불구하고 Sun Fire Link는 세 개의 Sun Fire 6800 서버들로 구성된 시스템이 하나의 대용량 서버인 Sun Fire 15K(System 1)의 수준에 가깝게 실행할 수 있도록 하는데 도움이 됨을 볼 수 있다.

4.2 Wrf

Wrf의 성능을 (그림 2)는 보여주는데, OpenMP와 MPI 모드로 세 가지 시스템에서 실행한 결과이다. 루프 수준의 병렬처리 때문에 Wrf는 OpenMP 모드에서 높은 확장성을 얻지 못한다. MPI를 사용할 경우 모든 시스템들은 32개의 프로세스까지는 속도 향상이 잘 이루어진다. 64개의 프로세스에서 System 3의 성능이 떨어지는 이유는 제한된 메모리 대역폭과 시스템 프로세스들의 활동 때문이다.

<표 4>는 하이브리드 모드에서 Wrf의 성능을 나타낸다. N x 4* 로 표시된 표는 4개의 쓰레드들이 Uniboard의 4개의 프로세서들에 바인딩되어 있는 것을 나타낸다. 하이브리드 병렬처리는 System 3에서 64 개의 프로세서를 사용하여



(그림 2) Wrf 의 MPI와 OpenMP 성능

〈표 4〉 Wrf 의 하이브리드 성능

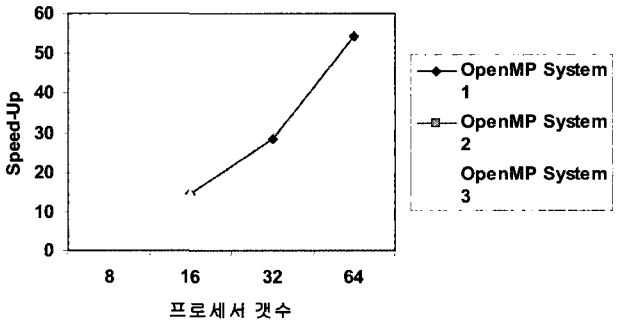
사용된 프로세서 개수	64		32			16			8
조합	16x4*	8x8	8x4*	8x4	4x8	4x4*	8x2	4x4	2x4*
System 1	30.5	19.6	20.6	18.2	14.4	11.6	11.4	10.4	7.4
System 2		24.7		18.8	16.5		12.1	11.1	
System 3	27.3	25.0	20.5	19.4	12.7	11.4	12.1	11.1	

얻을 수 있는 성능을 개선한다. 이것은 쓰레드 바인딩 때문이다.

4.3 Seis

앞에 3.3절에서 언급했듯이 Seis 벤치마크는 네 번째 데이터 세트만이 실행된다. Seis에서는 여러 다른 방법으로 MPI 프로세스들을 구성하는 방식을 System 2와 3에 적용한다. (예를 들어, 16 프로세스를 구성할 때 System 2에서 2x8, 4x4를 사용한다. 이전의 Gamess와 Wrf에서는 2x8만 사용했다.) 이렇게 함으로써, 인터커넥트가 성능에 미치는 효과를 알아볼 수 있게 된다. OpenMP 데이터는 (그림 3)에서 그리고 MPI 데이터는 <표 5>에서 보여주고 있다.

<표 5>에서 NxM 표시법은 각각의 노드에 M개의 MPI 프로세스가 있는 N개의 노드를 의미한다. OpenMP는 MPI보다 더 좋은 확장성을 보인다. OpenMP의 공유 메모리를 통한 데이터 복사가 MPI의 메시지 전달보다 더 적은 오버헤드를 보이기 때문이다. MPI 프로세스의 구성 방식은 성능에 별 영향을 미치지 못한다. 바꾸어 말해, 인터커넥트가 성능에 별 영향을 끼치지 못한다.



(그림 3) Seis의 OpenMP 성능

〈표 5〉 Seis의 MPI 성능

사용된 프로세서 개수	64	32	16		8	
System 1	27.9	25.1		14.3		7.5
System 2		(4x8)	(2x8)	(4x4)	(2x4)	(4x2)
		20.0	13.7	13.6	7.3	7.4
System 3		(2x16)	(2x8)	(1x16)	(2x4)	(1x8)
		25.7	13.8	14.2	7.3	7.6

5. 결 론

본 연구 논문에서는 HPC2002 벤치마크 suite을 이용하여, 대용량 SMP 서버인 Sun Fire 15K, Myrinet으로 연결된 8개의 Sun Fire V880 소형 서버들, 그리고 Sun Fire Link로 연결된 3개의 Sun Fire 6800 중형 서버들의 성능을 비교 분석하였다. Myrinet과 Sun Fire Link 모두 효과적인 인터커넥트들로 성능측정 결과 밝혀졌다. V880 클러스터는 MPI 와 하

이브리드 모드에서 뛰어난 성능을 보인다. 낮은 메모리 대기 시간 덕분이다. 만일 대규모의 SMP 시스템이 요구된다면 SF15K는 선택할 수 있는 훌륭한 대안이 될 것이다. Sun Fire Link로 연결된 Sun Fire 6800 클러스터는 노드의 높은 성능, 높은 대역폭과 낮은 대기 시간을 가진 인터커넥트와 함께 훌륭한 조합을 이룬다. 그러나 노드내의 제한된 메모리 대역폭은 성능 향상의 병목으로 작용할 수 있다.

참 고 문 헌

[1] Rocco Aversa, Beniamino Di Martino, Nicola Mazzoca, Salvatore Venticinque, "Performance Analysis of Hybrid OpenMP/MPI N-Body Application," 5th International Workshop on OpenMP Applications and Tools, Houston, TX, May, 2004.

[2] Manojkumar Krishnan, Yuri Alexeev, Theresa L. Windus, Jarek Nieplocha, "Multilevel Parallelism in Computational Chemistry using Common Component Architecture and Global Arrays," Super Computing (SC) 05, Seattle, WA, November, 2005.

[3] Myungho Lee, Larry Meadows, Darryl Gove, Dominic Paulraj, Sanjay Goil, Brian Whitney, Nawal Copty, and Yonghong Song, "Compiler Support and Performance Tuning of OpenMP Programs on SunFire Servers," European Workshop on OpenMP, Aachen, Germany, September, 2003.

[4] Message Passing Interface Standard, http://www-unix.mcs.anl.gov/mpi/

[5] OpenMP Architecture Review Board, http://www.openmp.org

[6] SPEC HPC2002 benchmark suite, http://www.spec.org/hpc2002

[7] Sun Fire 6800 server, http://www.sun.com/servers/mid-range

[8] Sun Fire 15K server, http://www.sun.com/servers/highend

[9] Sun Fire V880 server, http://www.sun.com/servers/entry

[10] Solaris 9 Operating System, http://www.sun.com/software/solaris

[11] Sun Studio 9 Compiler Suite, http://www.sun.com/software/products/studio/index.html

이 명 호

e-mail : myunghol@mju.ac.kr

1986년 서울대학교 계산통계학과(학사)

1988년 미국 University of Southern California 컴퓨터 과학과(석사)

1999년 미국 University of Southern California 컴퓨터 공학과(박사)

1989년~1995년 미국 CompuTech Corp., Technical Consultant

1999년~2003년 미국 Sun Microsystems, Inc. Scalable Systems Group, 책임 연구원

2003년~2004년 미국 Sun Microsystems, Inc. Scalable Systems Group, 수석 연구원

2004년~현재 명지대학교 컴퓨터소프트웨어학과 조교수

관심분야 : 고성능 컴퓨팅, 병렬 알고리즘, 마이크로 프로세서, 컴파일러