칩 멀티쓰레딩 서버에서 OpenMP 프로그램의 성능과 확장성

이 명 호[†] · 김 용 규[††]

요 약

최근 Chip-level MultiThreading(CMT) 기술을 내장한 프로세서 들이 출시되면서 그들을 기반으로 하는 공유 메모리 다중 프로세서(SMP: Shared Memory Multiprocessor) 서버 또한 그 사용이 점점 더 보편화 되고있다. OpenMP는 그 사용의 효율성으로 인하여 SMP 시스템을 위한 응용 프로그램의 병렬화를 위한 표준이 되었다. 고성능 컴퓨팅(HPC: High Performance Computing) 응용프로그램 분야에서 더욱 더 빠른 컴퓨터의 처리 능력에 대한 요구가 증가함에 따라, OpenMP 지시어를 사용하여 병렬화된 HPC 응용 프로그램 들의 성능과 확장성을 높이는 일은 그 중요성이 점차 증대되고 있다. 본 논문에서는 CMT 기술을 내장한 대용량 SMP 서버인 Sun Fire E25K에서 OpenMP 지시어를 사용하여 병렬화된 HPC 응용 프로그램 들의 suite인 SPEC OMPL(OpenMP를 위한 표준 벤치마크 suite)의 성능과 확장성에 관해 연구했다. 본 논문에서는 또한 SPEC OMPL에 대한 CMT 기술의 효능을 평가하였다.

키워드 : 칩 멀티쓰레딩, 공유 메모리 다중 프로세서, 고성능 컴퓨팅, OpenMP, 확장성

Performance and Scalability of OpenMP Programs on Chip-MultiThreading Server

Myungho Lee[†] · Yongkyu Kim[††]

ABSTRACT

Shared Memory Multiprocessor (SMP) systems adopting Chip-level MultiThreading (CMT) technology are becoming mainstream servers in commercial applications and High Performance Computing (HPC) applications as well. OpenMP has become the standard paradigm to parallelize applications for SMP mostly because of its ease of use. As the demand for more computing power in HPC applications is growing rapidly, obtaining high performance and scalability for these applications parallelized using OpenMP API's will become more important. In this paper, we study the performance and scalability of HPC applications parallelized using OpenMP, SPEC OMPL (standard OpenMP benchmark suite), on the Sun Fire E25K server which adopts CMT technology. We also study the effect of CMT on SPEC OMPL.

Key Words : Chip-MultiThreading, SMP, High Performance Computing, OpenMP, Scalability

1. 서 론

트랜지스터 집적도의 증가에 따라 마이크로 프로세서 설계자 들은 상대적으로 넓어진 실리콘 영역을 효과적으로 활용하기 위해 여러 다른 디자인 들을 고려해 왔다. 여러 선택 가능한 디자인 들 중에서 주목할만한 두 가지 추세는 (1) 단일 칩에 비교적 단순한 CPU 코어(core)를 여러개 내장하여 (CMP: Chip Multi-Processor[8]) 단위시간당 칩 전체의 처리량(throughput)을 극대화하는 것과 (2) 하나의 프로세서 코어 안에 동시적 멀티쓰레딩(SMT: Simultaneous MultiThreading) [15] 또는 이와 유사한 기술, 예를 들어 Intel의 하이퍼 쓰레딩(Hyper Threading)[4], 을 내장하여 여러 개의 소프트웨어 쓰레드를 실행할 수 있도록 하는 것이다. 이러한 디자인 들은 공통적으로 하나의 칩에서 여러 개의 소프트웨어 쓰레드를 동시에 실행하도록 하는 Chip-level MultiThreading(CMT) [11]을 지원한다. CMT 프로세서의 예로는 Dual-Core Intel Xeon[3], Pentium IV[4], Sun Microsystems사의 UltraSPARC IV, IV+[13] 프로세서 등이 있다. IBM[5], AMD[1], Fujitsu와 같은 업체 들 또한 유사한 제품들을 개발하고 있거나 곧 출시를 앞두고 있다.

최신의 CMT 프로세서 디자인에서는 위의 두 가지 디자

† 정 회 원 : 명지대학교 컴퓨터소프트웨어학과 조교수
†† 정 회 원 : 명지대학교 컴퓨터소프트웨어학과 석사과정
논문접수 : 2005년 11월 28일, 심사완료 : 2006년 2월 14일

인 추세들이 결합되어 사용되면서 하나의 칩상에 여러 개의 코어들이 탑재되고 각각의 코어는 SMT와 같은 기술을 내장하는 형태를 띄게된다[2]. CMT프로세서들은 상용 응용 프로그램 분야뿐만 아니라 고성능 컴퓨팅(HPC: high Performance Computing) 응용 프로그램 분야에서도 현재의 주류 프로세서가 되어가고 있으며, 이러한 프로세서를 장착한 소용량에서 대용량에 이르는 공유 메모리 다중 프로세서(SMP: Shared Memory Multiprocessor) 시스템도 그 사용이 점점 더 보편화 될 전망이다. 현재 출시되어 있는 대용량 서버의 예로는 Sun의 Sun Fire E25K[13]를 들 수 있다. E25K는 72개의 UltraSPARC IV 프로세서 칩을 기반으로 하며 최대 144개의 쓰레드를 실행할 수 있다.

OpenMP는 SMP 시스템을 위한 응용 프로그램의 병렬화를 위한 표준이 되었다. OpenMP는 C, C++, 그리고 Fortran 프로그램에서 병렬화를 구현하기 위해 사용되는 컴파일러 지시어(directive), 라이브러리 루틴과 환경 변수에 대한 명세이다[9]. OpenMP 사용의 주요 목적은 성능, 확장성, 효율성, 그리고 표준화이다. OpenMP는 비교적 적은 코딩 노력으로 기존의 순차적 실행 프로그램을 병렬화하는데 효과적인 도구로 증명이 되었다. HPC 분야에서 더욱 더 빠른 컴퓨터의 처리 능력에 대한 요구가 증가함에 따라 OpenMP 지시어를 사용하여 병렬화된 응용 프로그램의 성능과 확장성을 높이는 것은 더욱 중요한 과제가 되었다. 본 논문에서는 CMT 기술을 장착한 대용량 SMP 서버인 Sun Fire E25K에서 OpenMP 지시어를 사용하여 병렬화된 HPC 응용 프로그램 suite인 SPEC OMPL(OpenMP 를 위한 표준 벤치마크 suite)[11]의 성능과 확장성에 관해 연구했다. 또한 이 응용 프로그램 들에 미치는 CMT 기술의 효능을 평가하였다.

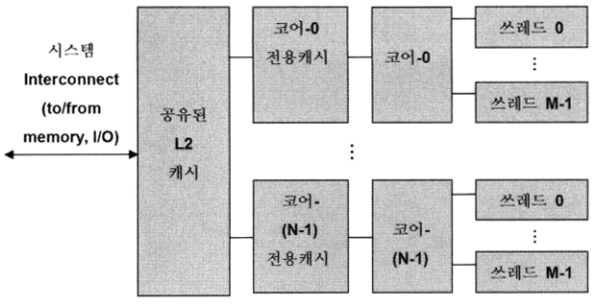
본 논문의 연구를 위하여 우리는 C, C++, Fortran OpenMP Version 2.0 표준을 지원하는 Sun Studio 9 컴파일러 suite을 활용하였다[14]. 먼저, Sun Studio 9 컴파일러에서 제공하는 상당히 고급 수준의 컴파일러 최적화 옵션을 적용하여 각각의 벤치마크 프로그램을 컴파일하였다. (많은 공통적인 컴파일러 최적화 작업을 수행하는 Sun Studio 9 의 -fast 옵션과 OpenMP 지시어를 처리하는 -openmp 옵션을 기본으로 사용하였다.) 그런 다음, 성능 분석 도구를 사용하여 각각의 벤치마크의 수행 파일에서 실행시 장애가 생기는 곳과 비 효율적으로 실행되는 부분을 찾아냈다. 장애가 되는 부분 들에 대하여 비능률을 최소화 함으로써 최적의 성능을 얻기위해 원인을 규명한 후 적절한 치유책(최적화 작업)을 적용하였다. 그러한 최적화 작업 들은 개선된 루프 변형(loop transformation), 코드 스케쥴링(code scheduling), 서브루틴 인라이닝(subroutine inlining), 데이터 프리페칭(data prefetching) 등이 있으며, 그밖에도 프로그래머가 소스 코드를 수정함으로써 데이터 분할(data distribution) 방법을 개선하는 것, 자동 병렬화(automatic parallelization), 병렬 루프에 적용되는 환경 변수 들의 활용, 메모리 친화적인 OS의 기능 적용 등이 있다. 최적화된 벤치마크의 실행 파일 들을 생성해낸 후, 사용되는 쓰레드의 갯수를 여러가지로 바꾸어가며 성능 측정 실험을 수행하였다. 또한 각각의 벤치마크에 미치는 CMT 기술의 효능을 측정하기 위해 두 가지의 다른 쓰레드 배치법(thread placement)을 이용하여 성능을 측정하였다. 실험 결과는 SPEC OMPL이 Sun Fire E25K에서 상당한 수준의 성능과 확장성에 이름을 보여준다. 또한 각각의 벤치마크 프로그램 별로 CMT의 효능을 보여준다.

본 논문의 나머지 부분은 다음과 같이 구성되었다. 2장에서는 CMT 서버의 일반적인 구조를 설명하고, 예로서 Sun Fire E25K 서버의 구조와 그 기본 OS인 Solaris 10 Operating System의 기능 들 중 OpenMP 응용 프로그램의 성능에 도움이 되는 기능 들을 설명한다. 3장에서는 OpenMP 지원을 위한 소프트웨어에 관하여 일반적인 사항을 설명하고, 예로서 Sun Studio 9의 OpenMP 컴파일러, 런타임 라이브러리와 성능 분석 도구 들에 대하여 설명한다. 4장에서는 SPEC OMPL 벤치마크 suite에 대하여 기술하고, Sun Studio 9 컴파일러를 활용하여 OpenMP 응용 프로그램의 성능을 최적화 시키는 방법론을 설명한다. 5장에서는 E25K 서버에서의 SPEC OMPL 성능 결과를 보여준다. 마지막으로 6장에서는 본 논문에서 얻은 결론과 향후 연구방향을 제시한다.

2. CMT 서버

트랜지스터 집적도의 증가에 따라 마이크로 프로세서 설계자 들은 상대적으로 넓어진 실리콘 영역을 효과적으로 활용하기 위해 여러 다른 디자인 들을 고려해 왔다. 여러 선택 가능한 디자인 들 중에서 새로운 기능들이 많이 추가된 복잡한 파이프라인을 가진 CPU 하나를 단일 칩에 내장하여 하나의 프로그램 성능을 극대화하는 것보다, 단일 칩에 CMP를 내장하여 단위시간당 칩 전체의 처리량을 극대화하는 것이 최근의 중요한 디자인 추세중 하나이다. 또 하나의 중요한 추세는 하나의 프로세서 코어안에 동시적 멀티쓰레딩[15] 또는 이와 유사한 기술(Intel의 하이퍼 쓰레딩[4] 등)을 내장하여 여러 개의 소프트웨어 쓰레드를 실행할 수 있도록 하는 것이다. 이러한 프로세서 들은 하나의 칩에서 여러 개의 소프트웨어 쓰레드가 동시에 실행되도록 지원하는 공통점을 지니므로 Chip-level MultiThreading(CMT)[11] 프로세서로 불린다.

최신의 CMT 프로세서에서는 위의 두 가지 디자인 추세들이 결합되어 사용되면서 하나의 칩상에 여러 개의 코어들이 탑재되고 각각의 코어는 SMT와 같은 기술을 내장하는 형태를 띄게된다. (그림 1)은 최신의 CMT 프로세서의 일반적인 구조를 보여주고 있다[2]. 각 프로세서 칩에는 N개의 코어 들이 있고, 각각의 코어는 개별적으로 할당된 캐시를 칩 내부에 가지고 있다. N개의 코어 들은 보다 용량이 큰 L2 캐시를 공유한다. 이 공유 캐시는 칩안에 내장되거나 칩 외부에 배치될 수 있다. 각각의 코어는 또한 SMT 또는 그와 유사한 기능을 수행하는 M개의 쓰레드를 갖는다. 가장 최근에 발표된 Sun Microsystems사의 T1 마이크로 프로세서의 경우 8개의 코어를 장착하고 있으며, 각 코어당 4개의 쓰레



(그림 1) 일반적인 CMT 프로세서의 구조

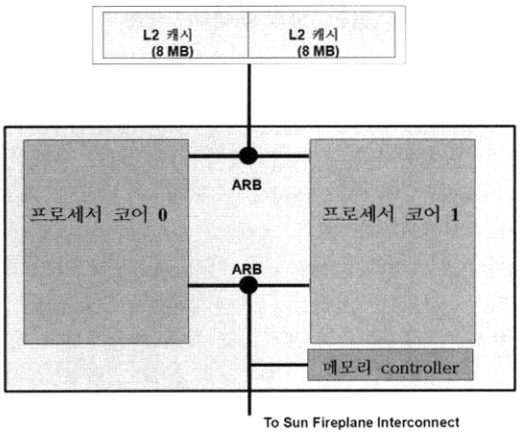
드를 지원한다. 각 코어당 데이터 캐시는 8KB이고, 공유되는 L2 캐시는 3MB이다. 앞으로의 서버 들은 이러한 CMT 프로세서 들을 기본 단위로 하여 설계될 것이다.

2.1 CMT 서버의 예: Sun Fire E25K

현재 출시되어 있는 CMT 기술을 내장한 대용량 서버로는 Sun Microsystems사의 Sun Fire E25K가 있다. 이 서버는 CMT기술을 이용하여 응용 프로그램의 단위 시간당 처리량(throughput)을 획기적으로 증가시키도록 하는 것을 목표로 설계되었다. 이 서버는 UltraSPARC IV 프로세서 칩을 기반으로 하고 있으며 72개의 프로세서 칩들이 각각 2개의 쓰레드를, 시스템 전체로는 144개의 쓰레드를 실행시킬 수 있도록 설계되었다. 이 새로운 시스템은 Sun의 이전 세대 시스템인 Sun Fire 15K와 비교할 때 전체 처리용량이 두 배까지 증가된다.

Sun Fire E25K 서버를 구성하는 기본 컴포넌트는 UniBoard이다. 각 UniBoard는 네 개의 UltraSPARC IV 프로세서 칩들과 메모리로 구성되어져있다. UltraSPARC IV는 하드웨어로 두 개의 쓰레드를 지원한다((그림 2) 참조). 각각의 쓰레드를 지원하기 위해서는 하나의 향상된 기능의 UltraSPARC III Cu 프로세서(UltraSPARC IV 이전 세대의 프로세서로서, 이전 세대의 서버인 Sun Fire 15K에 사용됨)코어가 사용된다. 그 이외에도 각각의 UltraSPARC IV에는 메모리 컨트롤러, 코어당 할당된 8MB 크기의 L2 캐시를 위한 캐시 태그(tag)를 포함하고 있다. L2 캐시는 칩 외부에 있으며 크기가 16MB(코어당 8MB)이다. 두 개의 코어 들은 L2 캐시 버스(bus)와 Fireplane System Interconnect를 공유한다.

Sun Fire E25K는 최대 18개의 UniBoard들로 구성되어 있다. 각각의 UniBoard에 4개의 UltraSPARC IV 프로세서 칩이 장착될 수 있으므로, 시스템 전체로는 72개의 프로세서칩이 장착된다. 최대 메모리 용량은 576GB이다. 시스템 버스의 clock은 150Mhz이다. E25K에서 캐시 일관성(cache coherency)을 유지하기 위해 같은 UniBoard상에서는 버스기반의 스누핑(snooping) 캐시 일관성 유지 프로토콜을 사용하며, 다른 UniBoard간에는 디렉토리(directory) 기반의 캐시 일관성 유지 프로토콜을 사용한다. 최대 메모리 전송 용량은 80GB/sec이다. 메모리 접근 시간(latency)을 lmbench의 lat_mem_rd 루틴을 사용하여 측정한 경우 동일한 UniBoard 상에서 240nsec이고, 다른 UniBoard의 메모리에 접근하는 경우 455nsec가 걸린다.



(그림 2) UltraSPARC IV 프로세서

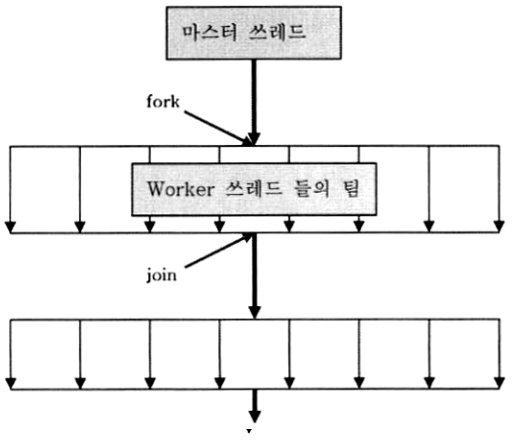
2.2 Solaris 10: Sun Fire E25K의 기본 OS

Solaris 10 OS는 Sun Fire E25K 서버를 위한 확장성과 고성능을 제공한다[10]. Solaris 10 OS의 기능 중 MPO (Memory Placement Optimization: 메모리 배치 최적화)와 MPSS(Multiple Page Size Support: 다중 페이지 크기 지원)는 OpenMP 프로그램의 고성능 확보에 매우 효율적으로 사용될 수 있다:

- MPO는 Solaris 10 OS가 특정 프로세서에게 그 프로세서가 자주 접근하는 페이지를 접근이 용이하도록 될 수 있는데로 가까운 곳에 할당하는 기능을 지원한다. 기본 MPO 정책(first-touch라고 불리움)은 처음으로 특정 페이지에 접근하는 프로세서가 속한 UniBoard의 메모리에 그 페이지를 배치한다. 이러한 first-touch 정책은 메모리로의 접근이 빈번한 응용 프로그램에서 메모리 접근 들이 같은 UniBoard에 되도록 많이 배치되게 할 수 있기 때문에 성능 향상에 매우 중요한 영향을 준다. 특히 Sun Fire E25K 서버와 같이 Non-Uniform Memory Access(NUMA) 시스템에서는 MPO 기능이 성능에 큰 향상을 가져올 수 있다.
- MPSS는 프로그램에게 가상메모리의 상이한 지역에 각각 다른 크기의 페이지를 사용하도록 해준다. UltraSPARC IV는 네가지 페이지 크기 들(8KB, 64KB, 512KB, 4MB)을 하드웨어적으로 지원한다. MPSS는 이들 네가지 페이지 크기 들 중 한가지를 특정 메모리 영역에 사용하도록 요청할 수 있다. 예를 들면, 스택과 힙 영역에는 4MB 크기의 페이지를 할당하고, 그 이외의 메모리 영역에는 기본 페이지 크기인 8KB를 사용할 수 있다. MPSS를 사용하면 대용량 메모리를 사용하는 응용 프로그램을 실행할 때 기본 크기의 페이지를 사용할 때보다 필요한 페이지 갯수가 크게 줄어든다. 그러므로 Translation-Lookaside Buffer(TLB)에서의 접근 실패 횟수를 대폭 줄임으로써 응용 프로그램의 성능을 크게 향상시킬 수 있는 방법이기도 하다. SPEC OMPL은 대용량 메모리를 사용하는 응용 프로그램으로서 MPSS 기능이 성능에 큰 영향을 준다.

3. OpenMP를 위한 소프트웨어 지원

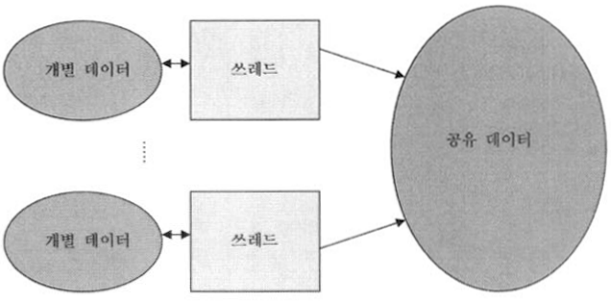
OpenMP의 기본적인 실행 모델은 fork-join 모델이다((그림 3) 참조)[9]. 코드의 병렬 구간을 만나기 전까지는 마스터 쓰레드가 혼자서 순차적인 실행을 하다가, 병렬 구간이 시작되는 지점에서 마스터 쓰레드는 worker 쓰레드 들의 팀(team)을 생성해낸다. 모든 쓰레드는 동시에 병렬구간 처리에 간여하고 병렬구간이 끝나는 지점(join)부터 마스터 쓰레드 혼자서 순차적인 처리를 계속하게 된다. OpenMP를 사용한 병렬화 방식은 순차적 프로그램 들에는 존재하지 않는 비용을 발생시킨다. 그러한 비용 들은 쓰레드 들을 생성하고, 동기화하고(synchronization), 공유 데이터에 접근하고, 개별 쓰레드와 관련된 정보를 기록하는 등의 일 들을 포함한다. 본 연구를 위해 사용된 Sun Studio 9 소프트웨어 suite은 C, C++, Fortran OpenMP 버전 2.0을 지원하는 컴파일러와 도구를 포함한다. 아래의 절 들에서는 이들에 대하여 자세히 알아본다.



(그림 3) OpenMP 실행 모델

3.1 컴파일러와 런타임 지원

Sun Studio 9 컴파일러는 OpenMP 지시어 들이 사용된 프로그램을 처리하여 다중 쓰레드에 의해 실행될 수 있는 코드로 변형시킨다. OpenMP 런타임 라이브러리(libmtsk)는 다중 사용자 쓰레드의 실행을 지원하며, 쓰레드 들의 관리, 동기화, 병렬 작업의 스케쥴링 등을 지원한다[14]. OpenMP 지시어를 사용한 병렬화 방식은 앞서 기술한 여러 비용 들을 발생시킨다. Sun Studio 9 컴파일러와 libmtsk는 다양한 기법을 사용하여 이러한 비용을 줄인다. 동기화하는 데 소요되는 비용은 빠른 lock 메커니즘, lock으로 사용되는 변수에 대한 패딩(padding)의 첨부, 개선된 트리 구조를 사용한 barrier의 실행 속도 향상 등의 방법 등이 사용된다. 공유 데이터(shared data)에 대한 접근 비용은 각 쓰레드 스택에 읽기 전용 공유데이터를 복사하여 놓음으로써 최소화 시킨다. 개별 데이터(private data) 할당에 따른 비용은 쓰레드 스택 들에 작은 배열을 할당 함으로써(힙에 할당하는 대신) 감소시킨다. 쓰레드 전용(threadprivate) 데이터 접근에 의한 비용은 쓰레드 전용 데이터 할당 시 거짓 공유(false sharing)를 최소화



(그림 4) OpenMP 프로그램의 데이터 사용

하도록 함으로써 감소시킨다. 쓰레드 생성에 따른 비용은 병렬 영역이 끝났을 때 worker 쓰레드를 제거하지 않고 추후에 다음 병렬 영역이 실행될 때 재활용하는 방식으로 개선시킨다.

OpenMP 프로그램이 실행될 때 성능을 향상시키기 위해 사용자는 실행 환경을 제어하는 다음과 같은 환경 변수 들을 사용할 수 있다:

- SUNW_MP_PROCBIND 환경 변수는 OpenMP 프로그램의 경량 프로세스(lightweight processes: LWP)들을 프로세서에 바인딩(binding) 한다. 이것은 바인딩된 쓰레드 들이 다른 프로세서로 이동하는 것을 막음으로써 쓰레드들에 의해 사용된 데이터가 캐시에 남아 있도록 해서 그들이 재사용되도록 한다. 이렇게 함으로써 캐시의 성능을 향상시킬 수 있다.
- SUNW_MP_THR_IDLE 환경 변수를 이용하면 worker 쓰레드가 작업이 할당되기를 기다리는 동안 스핀(spin)이나 슬립(sleep)상태로 있도록 사용자가 지정할 수 있다. 쓰레드 들을 슬립 상태로 놓는 것은 과 부하가 걸린 시스템(실행되는 쓰레드의 숫자가 이용 가능한 프로세서 코어의 숫자보다 큰 경우)에서 스핀하는 경우 보다 성능면에서 장점이 있다.
- SUNW_MP_GUIDED_SCHED_WEIGHT 환경 변수는 GUIDED 스케쥴 루프를 처리할 때 사용자가 chunk의 크기(chunk size)를 계산 함에 있어서 가중치(weight factor) 설정을 가능토록 한다. (기본 가중치는 1.0.) 신중한 가중치 선택에 의해서 사용자는 부하(load)를 균등하게 배분시킬 수 있다.

3.2 Tool 지원

Sun studio 9은 병렬화를 위한 다양한 도구를 제공한다. 제공하는 도구는 디버깅, 에러 체크, OpenMP 프로그램 들의 성능 분석 등을 위한 도구 들이다. 몇가지 도구를 살펴본다:

- **OpenMP 디버깅**

쓰레드용 dbx 명령어 들은 모두 C와 Fortran OpenMP 프로그램의 디버깅에 사용되어 질 수 있다. Dbx는 사용자에게 병렬 구간으로의 진입, breakpoint의 설정, 주어진 쓰레드에서 SHARED, PRIVATE, THREADPRIVATE 등의 변수값 출력과 같은 기능 들을 제공한다.

• 정적/동적 에러 체킹

Sun Studio 9은 정적/동적으로 OpenMP 프로그램을 체킹할 수 있도록 해주는 컴파일러 옵션과 런타임용 환경 변수를 제공한다. 이 도구를 사용하면 OpenMP 지시어 들의 사용에 있어서 일관적으로 사용되지 않은 점 들, 쓰레드간의 데이터 경쟁 조건(data race condition) 탐지 등과 같은 에러 들을 미리 발견해 낼 수 있다.

• Collector와 Performance Analyzer

수집 기능을 하는 도구(collector)와 성능 분석을 하는 도구(performance analyzer)의 쌍으로 이루어져 있다. 이것은 응용 프로그램 들의 성능과 관계된 데이터를 수집하고 분석하는데 사용되어진다. 수집기는 통계적 방법을 이용한 프로파일링 기법과 함수 호출을 추적하는 기법을 통하여 성능 데이터를 수집한다. 성능분석기는 수집기를 통해서 기록된 데이터를 처리하여 전체 프로그램 수준의 정보, 함수 수준의 정보, 함수 호출-피호출 관계, 소스 코드 수준의 정보, 어셈블리 명령어 수준의 정보 등을 다양한 방식으로 보여준다. OpenMP 프로그램과 관련된 정보는(barrier에서 소요된 시간 등과 같은 정보 들) 소스 코드에 바로 대응되는 방식으로 보여주기도 한다. 이러한 정보는 성능이 저하 되는 부분의 소스 코드를 사용자에게 바로 보여줌으로써 프로그래머의 성능 최적화 작업을 도울 수 있다.

3.3 그 밖의 컴파일러 최적화 작업

Sun Studio 9 컴파일러들은 OpenMP를 위한 지원 이외에 프로그램의 성능향상을 위한 코드 최적화 작업을 수행한다. OpenMP 프로그램의 성능향상을 위한 최적화는 자동 병렬화, 스칼라(scalar) 최적화, 루프 변형, 호출 관계가 있는 함수간의 최적화(inter-procedural optimization), 데이터 프리페칭 기능의 향상, 메모리 최적화, 인라이닝, 코드 복제(cloning), 루프 전개(unrolling), 편향된 타일링(skewed tiling)외에 여러가지가 포함된다. 이들 중 몇 가지에 대해서 설명한다:

• 스칼라 최적화

Sun Studio 9은 매우 많은 스칼라 최적화 작업을 한다. 예로서 산술계산 re-association, 조건문 제거, forward 대체, 분기문 합병, 루프와 무관한 문장을 루프 밖으로 이동 시키기, 유도 변수(induction variable) 사용의 최적화, 죽은 코드 제거(dead code elimination), 공통적인 sub-expression 제거(common sub-expression elimination) 등이 있다.

• 루프 최적화와 캐시 사용의 최적화

Sun Studio 9은 많은 루프 변형중 메모리 성능을 위한 변형을 개선함으로써 메모리 계층 구조와 캐시 성능 향상을 이룰 수 있다. 루프 분산(distribution), 루프 합성(fusion), 루프 교체(interchange), 루프 타일링(tiling), 루프 unimodular 변형, 배열 수축(array contraction) 등을 수행한다.

• 데이터 프리페칭

메모리 접근을 위한 대기시간 감소를 위해서 컴파일러는 코드에 데이터를 프리페치하는 명령어를 삽입한다. 프리페치할 후보 데이터 들은 데이터의 재사용(reuse) 정보를 기반으로 선별해낸다. 간접적인 배열 참조(포인터 사용 등으로 인한)와 배열의 인덱스가 복잡한 방식으로 표현된 데이터 들도 컴파일러는 프리페치할 수 있다.

• 어셈블리 명령어 수준의 병렬화를 위한 최적화

최신 프로세서는 많은 기능 유니트(functional unit) 들을 가지고 있다. 그리고 주어진 싸이클 안에서 여러개의 명령어들이 이슈(issue)되고 실행된다. 이러한 기능을 이용하기 위해 컴파일러가 어셈블리어 수준에서의 최적화 작업을 수행한다. 예로써 unroll-and-jam, 스칼라 대체, 코드 스케쥴링 등과 같은 것 들이 있다.

• 호출 관계가 있는 함수간의 분석과 최적화

Sun Studio 9은 함수간의 사이드 이펙트(side-effect) 분석, 에일리어스(alias) 분석, 전체 프로그램 분석 작업을 수행한다. 이들 분석은 캐시의 이용을 최적화 시키기 위한 데이터 레이아웃(layout) 최적화, 함수 인라이닝, 함수 복제, 상수 전달(propagation)을 포함한 다양한 최적화에 사용된다.

• 자동 병렬화

Sun Studio 9은 데이터 의존 관계(dependence)에 관한 분석과 다양한 루프 변형을 수행함으로써 루프의 자동 병렬화를 가능하게 한다. 루프의 병렬화가 가능하도록 하기 위해 루프 분산을 수행하며, 병렬화하려는 루프 들의 크기(granularity)를 증가시키기 위한 루프 합병을 수행하고, 병렬화가 가능한 루프를 중첩된 루프 들의 가장 바깥쪽에 두기 위해 루프 교환을 수행한다. 뿐만 아니라 컴파일러는 스칼라와 배열 변수 모두에 관하여 reduction 연산의 가능성을 인식한다. 또한 지시어 기반의 병렬화와 자동 병렬화 방식 들이 자유로이 조합되어 사용될 수 있도록 한다.

4. SPEC OMPL 벤치마크의 성능 최적화

이 장에서 우리는 SPEC OMPL 벤치마크 suite에 대해서 먼저 설명할 것이다. 그런 다음 OpenMP 응용 프로그램 들의 성능 최적화 작업에 대한 우리의 접근 방법을 보여줄 것이다.

4.1 SPEC OMPL 벤치마크

SPEC OMPL 벤치마크 suite은 C 와 Fortran으로 쓰여진 아홉 가지의 응용 프로그램 들로 구성되어 있고 OpenMP 지시어를 사용하여 병렬화 되었다[11]. 이들 벤치마크 들은 화학, 기계공학, 기상 모델링, 물리학 등의 분야를 대표하는 고성능 컴퓨팅 응용 프로그램 들이다(<표 1> 참조). 각 벤치마크는 한 개의 프로세서에서 실행시 6.4GB 이상의 메모리를 필요로 한다. 그러므로 벤치마크 들을 수행하기 위해서는 64bit 주소공간을 사용하는 대용량 시스템이 필요하게 된다.

〈표 1〉 SPEC OMPL 벤치마크

벤치마크	응용 분야	프로그래밍 언어
311.wupwise_l	Quantum chromodynamics	Fortran
313.swim_l	Shallow water modeling	Fortran
315.mgrid_l	Multi-grid solver	Fortran
317.applu_l	Partial differential equations	Fortran
321.equake_l	Earthquake modeling	C
325.apsi_l	Air pollutants	Fortran
327.gafort_l	Genetic algorithm	Fortran
329.fma3d_l	Crash simulation	Fortran
331.art_l	Neural network simulation	C

4.2 성능 최적화

OpenMP 응용 프로그램의 성능 최적화를 위하여 우리는 먼저 Sun Studio 9 컴파일러에서 지원하는 상당히 고급 수준의 컴파일러 옵션(-fast -openmp)을 사용하여 각각의 벤치마크 프로그램을 컴파일하였다. 그런 다음, 성능 분석 도구를 사용하여 각각의 벤치마크를 실행할 때 성능면에서 장애가 생기는 곳과 비 효율적으로 실행되는 부분을 찾아낸다. 그러한 부분 들에 대하여 원인을 규명한 후 다음과 같은 최적화 작업 들을 수행하였다:

- 사용자 코드에서 발견된 OpenMP 지시어와 관련이 없는 병목 들은 3.3절에서 설명한 고급 컴파일러 최적화 옵션 들을 사용하여 제거하도록 한다. (스칼라 최적화, 루프 변형, 메모리 계층 구조를 위한 최적화, 고급 데이터 프리페칭, 호출 관계가 있는 함수간의 최적화 등을 이용.)
- 컴파일러에 의한 프로그램의 자동 병렬화는 사용자가 OpenMP 지시어를 이용하여 표기한 병렬화에 추가해서 (혹은 사용자가 삽입한 OpenMP 지시어를 무시하고 전적으로 컴파일러의 병렬화 작업에 의존하는 경우) 성능을 더욱 향상시킬 수 있다. 예를 들면, 실행 횟수가 작은 루프를 병렬화하면 병렬화된 루프를 실행하는데 드는 비용이 병렬화 함으로써 생기는 이점보다 더 클 수도 있다. 315.mgrid_l 벤치마크에서 소스코드에 사용자가 OpenMP 지시어를 사용하여 지정한 병렬화 방식은 루프의 실행 횟수가 작은 경우나 큰 경우에 상관없이 루프에 똑같은 병렬화 방식이 적용된다. 사용자가 지정한 OpenMP 지시어 들을 무시하고 컴파일러로 하여금 루프 들을 자동으로 병렬화 하도록 하면 루프의 수행 횟수가 큰 경우는 병렬화하고 작은 경우 병렬화하지 않도록 한다. (이와 같은 컴파일러 최적화 작업을 루프 versioning이라고 부른다.) 이 최적화의 결과로 64개의 쓰레드를 사용할 경우 315.mgrid_l에서 10%의 성능 향상을 이룰 수 있었다.
- GUIDED SCHEDULE 지시어가 붙은 OpenMP 루프 들은 환경 변수 SUNW_MP_GUIDED_SCHED_WEIGHT를 사용하여 적절한 가중치를 지정하면 부하 배분을 개선시켜 성능 향상을 이룰 수 있다. 예를 들어, SPEC OMPM(SPEC OMPL의 축소판으로서 중간 크기의 입력 데이터를 이용하는 OpenMP 벤치마크 suite)에 속한 332.ammp_m 벤치마크의 경우 가장 실행 시간이 많이 걸리는 루프에서 부하 불균형이 상당히 심한데, 위의 환경 변수를 이용하여 가중치를 2.0으로 지정한 결과 64개의 쓰레드를 사용한 경우 25%의 성능 향상이 이루어졌다[16].
- 소스 코드의 부분중 컴파일러 최적화가 극히 제한되는 부분 들은 컴파일러의 최적화 기능이 좀 더 잘 적용될 수 있도록 코드를 수정할 수 있다. (SPEC OMPL의 성능 결과를 심사하고 발표하는 SPEC HPG 위원회는 이러한 소스 코드 수정 제안을 검토하여 합당하다고 판단되는 경우 승인한다.) 예로서 325.apsi_l 벤치마크에서 최초 데이터 분산을 수행하는 부분의 소스 코드를 수정하여 큰 성능 향상(64개의 쓰레드를 사용할 경우 22%)을 이룰 수 있었다. 원래의 소스 코드는 먼저 입력 배열을 일정 크기의 chunk로 분할해서 모든 프로세서(또는 프로세서 코어)에 배분한 후, 그 다음부터는 chunk 크기를 줄여나가며 배분하는 방식을 사용했다. 이렇게 하면 모든 프로세서가 거의 같은 양의 데이터를 할당받는 장점이 있으나, chunk의 크기가 작아지면서 하나의 페이지가 여러 프로세서에 의해 공유되는 단점이 생긴다. 즉 프로세서 들이 다른 UniBoard 상에 있는 메모리를 접근하게 될 가능성이 높아진다. 이러한 문제는 Sun Fire E25K와 같은 NUMA 시스템의 성능에 치명적인 영향을 줄 수 있다. 개선된 코드는 입력 배열을 4MB 크기의 chunk 들로 분할해서 프로세서 들에게 라운드-로빈(round-robin) 방식으로 배분한다. 각 프로세서마다 할당되는 데이터의 양이 차이가 날 수 있지만, 원래 코드의 문제점인 한 페이지가 여러 프로세서에 의해 공유되는 비효율이 사라져 큰 폭의 성능 향상이 이루어졌다.
- OpenMP 쓰레드들은 환경변수 SUNW_MP_PROCBIND를 사용함으로써 프로세서 들에게 바인딩 시킬 수 있다. 정적 스케쥴을 사용한 프로세서 바인딩은 응용프로그램의 성능 향상에 기여하게 되는데, 특정 쓰레드에 의해 이전에 사용된 데이터가 그 쓰레드가 바인딩된 프로세서의 캐시 또는 메모리에 아직 남아 있는 상태에서 그 데이터 들이 재사용될 경우 캐시 또는 메모리 접근 실패가 줄어들게 되기 때문이다. 프로세서 바인딩은 같은 프로그램을 여러번 실행할 경우 실행 시간 결과가 고르게 나타나게 하는 데도 도움이 되는데, 이것은 벤치마킹(benchmarking)에서 매우 중요한 요소이다.
- Solaris 10 OS의 MPO 기능을 효율적으로 사용하면 메모리 접근이 집중적으로 일어나는 응용프로그램의 성능을 크게 향상시킬 수 있다. MPO를 사용하면 메모리 접근 들이 실행 프로세서가 속한 Uniboard에 포함된 메모리로 향하게 될 확률이 증가되어 성능을 크게 향상시킬 수 있다. MPO를 사용치 않을 경우 그러한 메모리 접근

들이 다른 프로세서 들이 속한 UniBoard의 메모리로 향하게 되어 많은 비용을 초래하게 된다. 참고로, 2.1절에서 보았듯이 다른 UniBoard의 메모리 접근 시간은 프로세서가 속한 UniBoard의 메모리 접근 시간의 거의 두 배에 이른다: 455nsec vs. 240nsec.

- 마지막으로, 대용량의 메모리를 사용하는 프로그램에 대용량 페이지를 사용하게 되면 적은 수의 페이지 들로 전체 사용 메모리를 다룰 수 있게 된다. 따라서 작은 크기의 페이지를 사용할 때 보다 TLB 접근 실패 숫자가 크게 줄어 실제적으로 큰 성능 향상이 이루어진다. 64개의 쓰레드를 사용하여 OMPL suite에 대용량 페이지를 적용할 경우(각각의 벤치마크 별로 각각 다른 크기의 대용량 페이지 사용) 전체적으로 약 15% 가량의 성능 향상이 이루어졌다.

5. SPEC OMPL의 성능 결과

이 장에서는 앞서 4장에서 기술한 방법론을 이용하여 성능을 최적화 시킨 SPEC OMPL suite을 Sun Fire E25K 서버에 실행했을 때의 성능 결과를 보여준다. 먼저 Sun Fire E25K 서버에서 얻어진 성능과 확장성 결과를 보여줄 것이다. 그런 다음 SPEC OMPL에 대한 CMT의 효능을 평가한 결과를 보여주고, E25K 성능 결과를 이전 세대의 서버인 Sun Fire 15K의 결과와 비교해 보여줄 것이다.

5.1 Sun Fire E25K에서 성능과 확장성

우리는 SPEC OMPL suite을 72개의 UltraSPARC IV (1200Mhz 로 실행되는) 들을 탑재한 Sun Fire E25K에서 실행하여 성능과 확장성을 측정하였다. 앞서 말한대로 Sun Studio 9 컴파일러와 Solaris 10 OS를 사용하였다.

SPEC OMPL의 성능을 측정하는 방법은 다음과 같다:

- 각각의 벤치마크에서 실행 시간을 측정한 후, 그것을 기준 실행 시간(reference run time)으로 나눈다. (한 벤치마크의 실행 시간을 측정할 때 측정 시간의 편차를 고려하여, 세 번 이상 실행하며 그 중 중간 측정 값(median)을 취한다.) 그 나눈 값에 16,000을 곱하여 각 벤치마크의 성능 값을 계산한다.
- 9개의 벤치마크에 대하여 각각의 성능 값 들을 계산한 후 그들의 기하 평균(geometric mean) 값을 구한다. 이 값이 전체 벤치마크 suite의 성능 값이 된다.
- 벤치마크 성능 값을 계산할 때는 base 값과 peak 값 들을 측정할 수 있다. Base 성능값을 측정할 때는 동일한 컴파일러 옵션과 실행 환경을 모든 벤치마크 프로그램에 동일하게 적용하도록 SPEC HPG 위원회에서 규칙으로 정하고 있다. Peak 성능 값을 측정할 때는 그와 같은 제약이 없다. 본 논문에서는 peak 성능 값만 고려한다.

먼저 E25K 서버의 최대 성능 값을 측정하기 위해 143개의 프로세서 코어 들을 사용하여 143 쓰레드의 성능을 측정하였다. (하나의 프로세서 코어는 시스템 프로세스 들을 담당토록 하기 위해 벤치마크 프로그램 수행에는 할당하지 않고 남겨둔다. 이렇게 함으로써 프로그램이 실행되는 동안 프로그램 실행에 참가한 프로세서 코어 들이 다른 시스템 프로세스 들의 수행을 위하여 인터럽트 되는 일이 없도록 해서 균등한 실행 시간을 얻는데 도움이 된다.) 143개의 쓰레드를 이용하여 모든 벤치마크를 실행했을 때 전체 성능 값 355,169를 얻었다. 그 다음 확장성 비교를 위하여 72개의 쓰레드를 이용하여 성능을 측정하였다. 36개의 UltraSPARC IV 프로세서를 사용하여 각 프로세서당 두개의 코어를 모두 사용하는 방식으로 72개의 쓰레드를 할당했다. 결과로는 전체 성능 값 213,709를 얻었다. <표 2>는 정리된 성능과 확장성 결과를 보여준다.

〈표 2〉 SPEC OMPL 성능 결과 비교－143 쓰레드 경우와 72 쓰레드 경우

벤치마크 프로그램	72 쓰레드	143 쓰레드	143/72 확장성
311.wupwise_l	322866	576138	1.78
313.swim_l	173631	329112	1.90
315.mgrid_l	203995	345387	1.69
317.applu_l	178296	318168	1.78
321.equake_l	109876	138229	1.26
325.apsi_l	121673	196985	1.62
327.gafort_l	168407	254469	1.51
329.fma3d_l	177221	329635	1.86
331.art_l	1144272	1889675	1.65
기하 평균값	213709	355169	1.66

143 쓰레드의 전체 성능값을 72 쓰레드의 전체 성능값으로 나누었을 때 1.66의 확장성을 얻었다. 4개의 프로그램 들이 1.78 이상의 높은 확장성을 보여준다: 311.wupwise_l, 313.swim_l, 317.applu_l, 329.fma3d_l.

- 313.swim_l의 경우, 높은 메모리 대역폭(bandwidth)이 성능을 높이는데 필수적이다. Sun Fire E25K의 경우 최대 80GB/sec의 대역폭을 갖추고 있어 313.swim_l의 성능향상과 확장성에 상당히 큰 도움이 된다.
- 329.fma3d_l의 경우 72 쓰레드가 실행될 때 L2 캐시에서 접근 실패 횟수가 상당히 많은데, 사용되는 프로세서 코어의 갯수가 두 배로(72에서 143으로) 증가함에 따라 각 프로세서 코어당 L2 캐시에 적재될 데이터의 양이 크게 줄어든다. 따라서 L2 캐시 접근 실패가 크게 줄어 높은 확장성이 이루어진 경우이다.

321.equake_l과 327.gafort_l 등은 확장성이 상당히 낮다: 1.26, 1.51.

- 321.equake_l의 경우 동기화를 위한 비용이 쓰레드의 갯수가 많아지면서 상당히 늘어난 것과 순차적으로 실행되는 부분(serial portion)이 상대적으로 큰 것이 주된 이

유이다.

- 325.gafort_l는 lock을 이용하여 한번에 하나의 쓰레드만 접근하는 임계 구역(critical section)이 두 군데 있는데, 그 안에 많은 실행 시간을 요하면서 프로세서와 메모리 간의 데이터 이동이 많이 일어나는 루프 들이 존재한다. 이러한 임계 구역 루프 들이 확장성을 높이는데 장애가 된다.

5.2 CMT의 효능

SPEC OMPL 벤치마크에 대한 CMT 기술의 효능을 측정하기 위해 72개의 쓰레드를 두가지 방식으로 프로세서 코어들에 배치했다:

- 36개의 UltraSPARC IV를 사용하여, 각 프로세서에 있는 두개의 코어를 모두 사용하는 방법
- 72개의 UltraSPARC IV에서 코어 하나씩만 사용하는 방법.

첫째 방법은 이미 앞의 5.1절에서 143개의 쓰레드와 확장성 비교 연구를 위하여 사용했던 방법이다. <표 3>은 둘째 방법을 사용한 72개 쓰레드의 성능 결과를 첫번째 방법으로 얻은 성능 결과와 비교해서 보여주고 있다.

〈표 3〉 72 쓰레드의 경우－36x2 vs. 72x1

벤치마크 프로그램	36 chip X 2 코어	72 chip x 1 코어	1 코어 vs. 2 코어
311.wupwise_l	322866	341995	1.06
313.swim_l	173631	241152	1.39
315.mgrid_l	203995	244380	1.20
317.applu_l	178296	183623	1.03
321.equake_l	109876	112975	1.03
325.apsi_l	121673	148611	1.22
327.gafort_l	168407	214397	1.27
329.fma3d_l	177221	177211	1.00
331.art_l	1144272	1477963	1.29
기하 평균값	213709	247534	1.16

전체적으로 두번째 방법이 첫째 방법에 비해 1.16배의 성능 향상이 됨을 보여준다. 비교적 큰 성능 향상이 이루어 지는 벤치마크 들의 경우 높은 메모리 대역폭을 필요로 하거나 사용되는 메모리 용량이 상당히 크다는 특징이 있다:

- 313.swim_l의 경우 앞서 설명한 대로 메모리 대역폭이 중요한 벤치마크로서 프로세서 하나당 코어 하나만 사용하는 경우 메모리 전송 용량을 더 잘 활용할 수 있게 되어 1.39배나 되는 큰 성능 향상이 이루어 짐을 볼 수 있다.
- 315.mgrid_l의 경우도 313.swim_l과 비슷하게 높은 메모리 대역폭을 요구하는 벤치마크로서 프로세서 하나당 코어 하나만 사용하는 경우 메모리 전송 용량을 더 잘 활용할 수 있게 되어 큰 성능 향상이 이루어진다. 단 앞 절에서 보았듯이 확장성(143 쓰레드와 72 쓰레드의 비교)이 낮은 이유는 가장 실행 시간이 많이 걸리는 병렬 루프의 경우 최대 실행횟수가 512로서, 사용되는 쓰레드의 갯수에 비해 상당히 작다. 따라서 동기화하는데 드는 비용이 차지하는 부분이 쓰레드의 갯수가 늘어날수록 점점 커져서 높은 확장성을 얻는데 장애가 된다.
- 325.apsi_l의 경우 프로세서당 두개의 코어 들을 사용하면 한 UniBoard에서 8개의 쓰레드 들이 동시에 여러 개의 4MB 크기의 페이지를 배치하려 하기 때문에(4.2절 참조) 메모리에 병목 현상이 발생하게 된다. 프로세서당 하나의 코어만 사용하면 이러한 메모리 병목 현상이 줄어들어 성능 향상에 도움이 된다.
- 327.gafort_l의 경우 프로세서당 두개의 코어 들을 사용하여 8개의 쓰레드가 한 UniBoard 상에서 실행되게 되면 임계 구역에 진입하기 위해 접근하는 lock 들이 같은 UniBoard의 메모리나 캐시에 있을 확률이 상대적으로 높아져 lock에 대한 접근이 빨라지는 장점이 있다. 반면, 메모리와 프로세서 코어간의 데이터 이동이 잦은 경우 메모리 대역폭을 많이 사용하게 되어 성능 향상에 장애가 된다. 후자가 전자보다 전체적인 성능에 더 많은 영향을 미친다고 볼 수 있다.
- 331.art_l의 경우 각 쓰레드 당 여러개의 큰 배열 들을 동적으로 할당받아 사용한다. 프로세서당 두개의 코어들을 사용하게 되면 한 UniBoard에서 8개의 쓰레드 들이 동시에 많은 양의 메모리를 동적으로 할당 받아 사용하려 경쟁하기 때문에 메모리에 병목 현상이 걸리게 된다. 프로세서당 하나의 코어만 사용하면 쓰레드간의 메모리 할당 및 사용을 위한 경쟁이 줄어들어 성능 향상에 도움이 된다.

종합해 보면 위의 벤치마크 들은 메모리 대역폭을 집중적으로 사용하기 때문에 프로세서당 하나의 코어만을 사용할 경우에 두개의 코어를 다 사용할 때와 비교해서 비교적 큰 폭으로(20% 이상) 성능 향상이 이루어진다고 할 수 있다. 그 이외의 벤치마크(311.wupwise_l, 317.applu_l, 321.equake_l, 329.fma3d_l) 들은 프로세서당 하나의 코어만을 사용할 경우와 두개의 코어를 다 사용할 경우 성능 차이가 거의 생기지 않는다. 그러므로 CMT 기술이 이들 벤치마크에 더 적합하다고 할 수 있다.

5.3 Sun Fire 15K 성능 결과와 비교

이 절에서는 SPEC OMPL의 Sun Fire E25K에서의 성능과 이전 세대의 서버인 Sun Fire 15K에서의 성능[6]을 비교 분석한다. Sun Fire 15K는 72개의 UltraSPARC III Cu 프로세서(1200Mhz로 실행되는)들을 기반으로 한다. UltraSPARC IIICu는 앞에서도 설명했듯이 UltraSPARC IV 이전 세대의 프로세서로서 한 칩에 하나의 코어밖에 존재하지 않는다. 그러므

로 E25K와는 달리 최대 72개의 쓰레드 들밖에 실행하지 못한다. SPEC OMPL suite을 Sun ONE Studio 8 컴파일러(Sun Studio 9 의 이전 세대 컴파일러)를 사용하여 컴파일한 후 Sun Fire 15K에서 71개의 쓰레드를 사용하여 실행한 결과를 앞 절에서 보여준 Sun Fire E25K의 143개의 쓰레드를 사용한 경우의 성능과 비교해 표 4와 같은 결과를 얻었다. (5.1절에서 설명했듯이 Sun Fire 15K 성능 측정에서도 프로세서 하나는 다른 시스템 프로세스 들을 실행하도록 남겨두었다.)

종합 증가비율은 1.66배(355169 vs. 213466)이다. Sun Fire 15K의 전체 성능 값 213466은 E25K에서 프로세서당 두개의 코어를 사용했을 경우의 전체 성능 값(213709)과 비슷하다. 하지만 E25K에서 프로세서당 하나의 코어만 사용했을 경우의 전체 성능 값(247534)과 비교하면 약 14% 낮은 성능이다. 그 이유는 UltraSPARC IV의 개선된 L2 캐시 구조, Sun Studio 9 컴파일러의 개선된 성능(Sun ONE Studio 8과 비교해서) 등을 들 수 있겠다. 표 4는 벤치마크 별로 자세한 결과(Sun Fire E25K vs. Sun Fire 15K)를 보여준다.

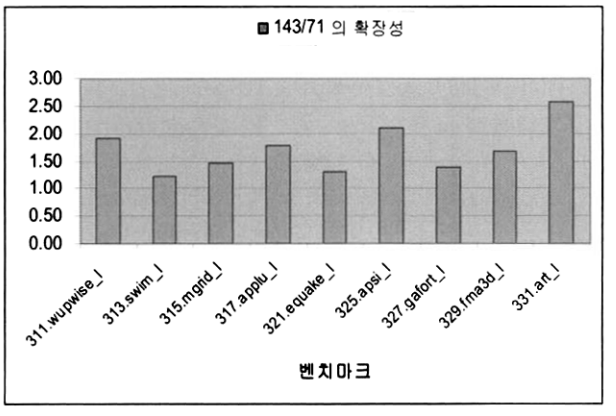
<표 4>에서 보듯이 대부분의 벤치마크 들은 좋은 성능 향상을 보여주고 있다. 그 증가정도는 1.22배에서부터 2.57배에 이른다. 331.art_l 같은 경우는 CMT 기술을 비롯한 하드웨어 성능뿐만 아니라 컴파일러의 향상된 메모리 최적화 작업에 의해서 굉장히 두드러진 성능 향상(2.57배)을 보여주고 있다. 325.apsi_l 또한 그 향상치가 눈에 띄게 두드러진다(2.1배). 4.2절에서 설명한 소스 코드의 수정이 성능 향상의 큰 이유이다. 311.swim_l과 321.equake_l 같은 벤치마크 들에서는 그다지 좋은 성능 향상이 보이지 않고 있다. 이것은 UltiraSPARC IV 프로세서 칩의 두개의 코어 들이 같은 메모리와 L2 캐시 버스를 공유하기 때문에 각 코어당 실제적인 메모리 대역폭이 줄어든 탓이다. 그림 5는 표 4의 결과를 그래프로 보여주고 있다.

<표 4> Sun Fire 15K(71 쓰레드) vs. Sun Fire E25K(143 쓰레드)

벤치마크 프로그램	71 쓰레드	143 쓰레드	143/71 확장성
311.wupwise_l	303373	576138	1.90
313.swim_l	270239	329112	1.22
315.mgrid_l	237237	345387	1.46
317.applu_l	177501	318168	1.79
321.equake_l	105699	138229	1.31
325.apsi_l	93757	196985	2.10
327.gafort_l	184157	254469	1.38
329.fma3d_l	198462	329635	1.66
331.art_l	734672	1889675	2.57
기하 평균값	213421	355169	1.66

6. 결론과 향후 연구방향

본 논문에서 우리는 CMT 기술을 내장한 대용량 서버 Sun Fire E25K의 구조와 Solaris 10 OS에 대하여 먼저 살펴보았다. 그런 후 E25K 서버에서 OpenMP 벤치마크의 성능을 최적화하는 방법론에 대하여 설명하였다. 이 방법론을 이용하여 SPEC OMPL 벤치마크 suite의 성능을 최적화 한 후 E25K 서버에서 측정한 성능과 확장성을 살펴 보았다. 결과는 SPEC OMPL 벤치마크 suite이 143개의 쓰레드를 이용하여 실행되었을 때 72개의 쓰레드 결과와 비교할 경우 1.66배의 성능 향상을 보여준다. 또한 72개의 쓰레드를 사용하여 SPEC OMPL 벤치마트 suite을 실행하는데 쓰레드가 프로세서 코어에 배치되는 방법을 두 가지로 바꾸어 가며 성능을 측정함으로써 CMT 기술의 효용성에 대하여도 살펴보았다. 전체적으로 하나의 프로세서 칩에서 하나의 코어만을 사용하는 경우가 두개의 코어를 모두 사용하는 경우보다 1.16배의 성능 향상이 됨을 보여준다. 비교적 큰 성능 향상이 이루어 지는 벤치마크 들의 경우 높은 메모리 대역폭을 필요로 하거나 사용되는 메모리 용량이 상당히 크다는 특징이 있다. 이러한 벤치마크들에는 CMT의 효능이 떨어진다고 할 수 있다. 마지막으로 E25K의 성능을 이전 세대의 서버인 Sun Fire 15K의 성능과도 비교해 보았다. 대부분의 벤치마크 들은 E25K에서 좋은(1.22배~2.57배) 성능 향상을 보여주고 있다. 이러한 성능 향상은 CMT 기술을 비롯한 하드웨어 성능뿐만 아니라 컴파일러의 향상된 최적화 작업에 기인한다.



(그림 5) Sun Fire 15K(71 쓰레드)와 Sun Fire E25K의 확장성 비교

CMT 기술은 최근에 출시된 마이크로 프로세서 들에 사용되기 시작했으며 앞으로 활용 범위가 서버뿐만 아니라 개인용 컴퓨터에 이르기까지 크게 넓어질 핵심 기술이다. CMT 기술이 적용된 프로세서와 서버의 성능을 향상 시키기 위하여 그에 맞는 소프트웨어(컴파일러, OS)의 개발이 필요하게 되었다. 예를 들면, 우리의 실험 결과에서 보듯이 높은 메모리 대역폭을 필요로 하는 응용 프로그램의 경우 동일한 칩 상의 두개의 프로세서 코어로부터 공유하고 있는 캐시 및 메모리 버스를 사용하기 위해 발생하는 병목 문제를 어떻게 해결할 것인지 하는 것과, 두개의 쓰레드가 공유하는 캐시를 사용하면서 생기는 경쟁과 충돌을 최소화하는 문제 등은 중요한 연구 과제들이다. 이러한 과제들에 대한 추가 연구를 진행할 계획이다.

참 고 문 헌

[1] AMD Multi-Core: Introducing x86 Multi-Core Technology & Dual-Core Processors, http://multicore.amd.com/2005.

[2] *AMD Multi-Core: Introducing x86 Multi-Core Technology & Dual-Core Processors*, http://multicore.amd.com/2005

[3] Shailender Chaudhry, Paul Caprioli, Sherman Yip, and Marc Tremblay, *High-Performance Throughput Computing*, IEEE Micro, May-June, 2005.

[4] Intel Dual-Core Server Processor, http://www.intel.com/business/bss/products/server/dual-core.htm

[5] Intel Hyperthreading Technology, http://www.intel.com/technology/hyperthread/index.htm

[6] R. Kalla, B. Sinharoy, and J. Tendler, *IBM POWER5 chip: a dual core multithreaded processor*, IEEE Micro, March-April, 2004.

[7] 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.

[8] Yuan Lin, Christian Terboven, Dieter an Mey, and Nawal Copty, *Automatic Scoping of Variables in Parallel Regions of an OpenMP Program*, 5th International Workshop on OpenMP Applications and Tools, Houston, Texas, May, 2004 (LNCS 3349).

[9] K. Olukotun et. al., *The Case for a single Chip-Multiprocessor*, International Conference on Architectural Support for Programming Languages and Operating Systems, 1996.

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

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

[12] *The SPEC OMP benchmark suite*, http://www.spec.org/omp

[13] L. Spracklen and S. Abraham, *Chip MultiThreading: Opportunities and Challenges*, 11th International Symposium on High-Performance Computer Architecture (HPCA-11), pp.248-252, 2005.

[14] *Sun Fire E25K server*, http://www.sun.com/servers/highend/sunfire_e25k/index.xml

[15] *Sun Studio 9 Software*, http://www.sun.com/software/products/studio/index.html

[16] D. Tullsen, S. Eggers, and H. Levy, *Simultaneous MultiThreading: Maximizing On-Chip Parallelism*, International Symposium on Computer Architecture, 1995.

[17] Brian Wylie and Darryl Gove, *OMP AMMP analysis with Sun ONE Studio 8*, European Workshop on OpenMP, Aachen, Germany, September, 2003.

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

김 용 규

e-mail : neoqkim@mju.ac.kr
2002년 명지대학교 전자정보통신공학과 (학사)
2001년~2002년 ㈜시큐어엔지니어링 연구개발팀 연구원
2002년~현재 남양공업㈜ 정보화 추진팀 계장
2005년~현재 명지대학교 컴퓨터소프트웨어학과 석사과정
관심분야 : SI, Computer 보안, 자동차 전자제어, MCU