클러스터 시스템을 위한 효과적인 OpenMP 디렉티브 변환

기양석° 하순회
서울대학교 전기/컴퓨터공학부
{yskee°, sha}@iris.snu.ac.kr

김진수
KAIST 전자전산학과 전산학 전공
jinsoo@cs.kaist.ac.kr

Efficient Translation of OpenMP Directives for Cluster Systems

Yang-Suk Kee° Soonhoi Ha
School of Electrical Engineering and Computer Science Department, Seoul National University
Jin-Soo Kim
Division of Computer Science, KAIST

요 약

SMP 클러스터가 고성능 계산을 위한 플랫폼으로 등장함에 따라, 이 시스템을 활용하기 위한 프로그래밍 환경에 대한 관심이 증가하고 있다. 이 논문에서 우리는 ParADE라고 부르는 쉽고, 이식성이 높으며, 고성능의 프로그래밍이 가능한 새로운 프로그래밍 환경을 소개한다. ParADE는 OpenMP 프로그래밍 환경으로 HLRC 변종 프로토콜을 구현한 다중 쓰레드 DSM 시스템을 기반으로 하고 있다. 특별히, 이 논문에서는 성능 개선을 위한 OpenMP 변환기의 역할에 중점을 둔다. OpenMP 변화기는 OpenMP 프로그램 모델과 실행 시간 시스템의 수행 모델 사이에서 가교 역할을 한다. 특히, OpenMP 변환기는 동기화 디렉티브를 변환하고 임계 영역에 있는 작은 변수의 메모리 일관성을 유지하기 위해 집합 통신 함수를 활용한다. 동기화 디렉티브 성능 측정을 위한 마이크로벤치마크 프로그램을 통한 실험에서 ParADE 시스템은 기존의 DSM 시스템에 비해 우수한 성능을 보였다.

1. 서 론

마이크로프로세서와 네트워크 장비들의 대중화는 SMP 클러스터 시스템을 고성능 계산을 위한 플랫폼으로 등장시켰다. 비록, 물리적으로 작은 크기의 클러스터 시스템을 구성하는 것은 용이하지만, 이 시스템을 효과적으로 활용하기 위한 병렬 프로그래밍 환경은 아직 미진하다.

공유 주소 공간(shared address space) 프로그래밍 모델은 공유 메모리 다중 프로세서를 위한 모델로써, 일반적인 메모리 연산을 통해 프로세서 사이의 통신을 기술하여 병렬 컴퓨터를 추상화한다. 특히, 병렬 컴퓨터 제조 회사들에 의해 제안된 OpenMP[1]는 공유 주소 공간 모델의 표준으로 그 위치를 확고히 하고 있다. OpenMP는 쓰레드 프로그래밍에 대한 상위 수준의 인터페이스를 제공하는 컴파일러 디렉티브의 집합이다. OpenMP는 프로그래밍이 쉽다는 공유 주소 공간 모델의 고유한 특징에 더불어 여러 과학 계산 응용에서 고성능을 발휘한다.

DSM(distributed shared memory) 시스템은 클러스터와 같은 메시지 전달 구조에서 공유 메모리를 제공하는 미들웨어이다. 따라서, SMP 클러스터에서 OpenMP 모델을 사용하기 위해서는, OpenMP와 DSM 시스템을 통합하는 것이 자연스러워 보인다. 하지만, SMP 클러스터 시스템은 공유 메모리 구조와 메시지 전달 구조가 공존하는 구조적인 특징을 가지고 있기 때문에, 기존의 프로그래밍 모델을 그대로 적용하는 것은 시스템의 잠재된 성능을 발휘하는데 부적합하다.

이 논문에서 우리는 SMP 클러스터를 활용하기 위한 OpenMP 기반의 병렬 프로그래밍 환경으로 ParADE 시스템을 소개한다. ParADE 시스템은 크게 OpenMP 변환기와 ParADE 실행 시간 시스템으로 구성되어 있다. OpenMP 변환기는 OpenMP 프로그래밍 모델과 ParADE 실행 시간 시스템의 프로그래밍 인터페이스 사이에서 가교 역할을 한다. 반면, ParADE 실행 시간 시스템은 다중 쓰레드 기반의 소프트웨어 DSM 시스템을 통해 단일 시스템 이미지를 제공한다. 특별히, 이 논문에서 우리는 OpenMP 변환기에 중점을 두고, OpenMP 디렉티브를 효과적으로 변환하여 기존의 DSM 기반의 시스템보다 성능을 개선시킬 수 있음을 보인다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 ParADE 시스템에 대한 개략적인 소개를 한다. 3장에서는 OpenMP 변환기의 역할과 성능 개선 방법을 제시한다. 4장에서는 마이크로벤치마크 프로그램을 통한 실험 결과를 제시한다. 5장에서는 현재 진행 중인 일과 향후 연구 주제에 대한 언급을 함으로써 결론을 맺는다.

2. ParADE 시스템

ParADE 시스템은 OpenMP 변환기와 ParADE 실행 시간 시스템으로 구성되어 있다 (그림 1). 실행 시간 시스템은 다중 쓰레드 기반의 DSM 시스템과 메시지 전달 라이브러리로 이루어진다. 다중 쓰레드 기반의 DSM 시스템은 데이터 지역성을 높이기 위해 이동 홈(migratory home)을 갖는 HLRC(home-based lazy release consistency) 프로토콜[2]을 바탕으로 하고, 메시지 전달 라이브러리는 MPI를 사용한다.

ParADE 시스템의 특징은 명시적인 메시지 전달을 통해, 공유 메모리를 접근하는 최상 경로(critical path)에서

락 기전을 제거한다는 점이다. ParADE 시스템은 변수를 크기에 따라 분류하고, 다른 프로토콜을 적용한다. 크기가 작은 변수는 메시지 전달을 통한 갱신 프로토콜(update protocol)을, 크기가 큰 변수는 HLRC를 이용한 무효화 프로토콜(invalidate protocol)을 사용한다.

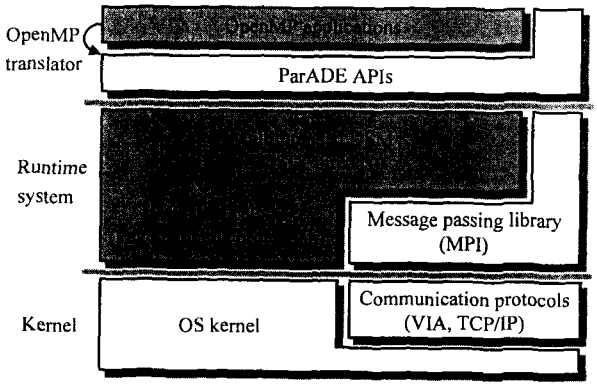

그림 1. ParADE 시스템 구조

OpenMP 변환기는 OpenMP 프로그램을 ParADE 실행 시간 라이브러리를 이용하여 다중 쓰레드 기반의 프로그램으로 변환하여 SMP 클러스터에서 수행되도록 한다. 특히, 변환기는 동기화 디렉티브를 변환할 때 메시지 전달 연산을 어떻게 활용할 것인가에 중점을 둔다.

3. OpenMP 변환기

OpenMP 변환기의 기본적인 역할은 OpenMP 시방서와 하부 실행 시간 시스템이 제공하는 프로그래밍 인터페이스 사이의 괴리를 해소하는 것이다. ParADE의 OpenMP 변환기는 OpenMP 프로그램을 POSIX 쓰레드와 MPI를 이용한 C 코드로 변환한다. POSIX 쓰레드와 MPI에 대한 자세한 내용은 ParADE API에 감추어진다. 이 OpenMP 변환기는 Omni OpenMP 컴파일러[3]의 변종이다. 이 장에서 우리는 몇 가지 중요한 OpenMP 디렉티브가 어떻게 ParADE API로 변환되는지 설명한다.

3.1 Parallel 디렉티브

Parallel 디렉티브는 기본 디렉티브로 병렬 수행을 시작한다. Parallel 디렉티브로 둘러싸인 코드 블록은 하나의 쓰레드 함수로 정의되고, 이 디렉티브는 *fork-join* 형태의 병열성을 실현하기 위한 ParADE 실행 시간 인터페이스로 치환된다. 이 함수에는 shared, reduction, firstprivate, lastprivate으로 선언된 변수에 대한 포인터들이 함수 인자로 전달되고, private 변수들은 함수 내에 지역변수로 선언된다.

한 프로세스 내의 쓰레드들은 쓰레드 스택을 제외한 모든 자료 구조를 공유하기 때문에 parallel 블록 내의 변수들의 기본 스코프 규칙은 shared 이다. 그러나, 이 규칙은 서로 다른 노드에 있는 변수를 공유하는 기전이 없는 메시지 전달 구조에서는 부적절하다. 따라서, 좋은 성능을 얻기 위해서는 parallel 블록에 정의된 모든 지역 변수들은 private이라고 명시적으로 주석을 붙여 불필요한 네트워크 소통 양을 줄여야 한다.

3.2 동기화 디렉티브

OpenMP에는 몇 가지의 동기화를 위한 디렉티브가 정의되어 있다. Critical 디렉티브는 쓰레드 사이의 상호 배제를 제공하며, 주로 스칼라 타입이 아닌 변수 값을 환산(reduce)하는데 사용된다. 이 디렉티브는 하나의 POSIX 쓰레드 락과 하나의 집합 통신 함수로 치환될 수 있다. 그림 2는 전통적인 DSM 시스템과 ParADE 시스템에 대하여, critical 디렉티브가 어떻게 변환되는지 보여준다. ParADE 시스템에서 상호 배제는 계층적으로 이루어진다. 먼저, pthread 락이 하나의 프로세스 내에서 쓰레드 간의 상호 배제를, 집합 통신 함수가 프로세스간의 상호 배제를 보장한다. 하지만, DSM 시스템에서는 노드 내와 노드 사이의 상호 배제에 대하여 단일 락 원시 함수를 사용한다. 이와 같은 명시적인 메시지 전달 함수는 전통적인 DSM에서 프로세스간의 락을 얻기 위한 절차와 동기화 유지를 위한 twin[1]과 diff[2]를 생성하는 절차를 생략함으로써 동기화 성능을 개선할 수 있다.

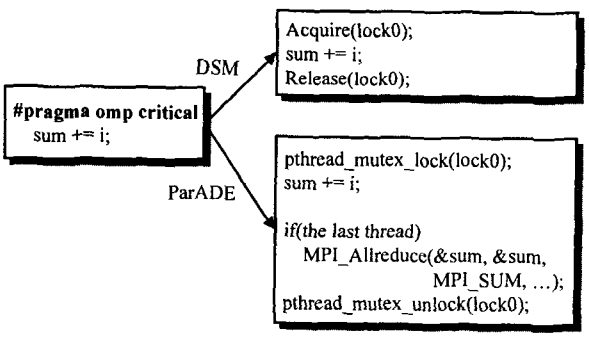

그림 2. critical 디렉티브 변환 코드 예시

다른 몇 가지 디렉티브들도 이와 동일한 방식으로 변환된다. Atomic 디렉티브는 critical 디렉티브의 특별한 경우로 간주된다. 또한, reduction 변수들도 메시지 전달 함수를 사용하여 즉시 갱신된다. 만약, 하나 이상의 reduction 변수가 선언되면, 이 변수들은 하나의 구조체 타입의 변수로 선언되고, 사용자가 정의한 reduction 함수에 의해 한꺼번에 환산된다. 가능한 많이 메시지 전달에 의해 혜택을 누리기 위해, 프로그래머는 critical 디렉티브 대신 reduction 변수를 사용할 것을 권장한다. 또한, critical 디렉티브를 위한 코드 블록은 어휘적으로 분석 가능하도록 작성하는 것이 바람직하다.

3.3 부하-분배 디렉티브

부하-분배 디렉티브는 부하를 쓰레드에 분배한다. For 디렉티브는 루프의 반복 회수를 여러 묶음으로 나누어 쓰레드에 할당한다. 변환기가 루프로부터 범위와 증가

1) twin: 한 페이지의 복사본
2) diff: twin과 현재 페이지의 차이

값을 추출하면, 실행 시간 시스템의 루프 스케줄러가 쓰레드를 위한 반복 회수의 범위를 동적으로 결정한다. 현재 실행 시간 시스템은 반복 회수를 균등하게 분배하는 static 방식만을 지원한다.

Single 디렉티브는 가장 먼저 임계 영역에 진입한 쓰레드에 의해 코드 블록이 수행되도록 한다. 그림 3은 전통적인 DSM 시스템과 ParADE 시스템을 위한 single 디렉티브의 변환된 코드의 모습이다. Critical 디렉티브와 유사하게 노드 내의 동기화는 하나의 pthread 락으로, 노드 사이의 동기화는 명시적인 메시지 전달 연산으로 치환된다. 배리어는 암시적으로 방송 연산으로 대치하여 생략한다.

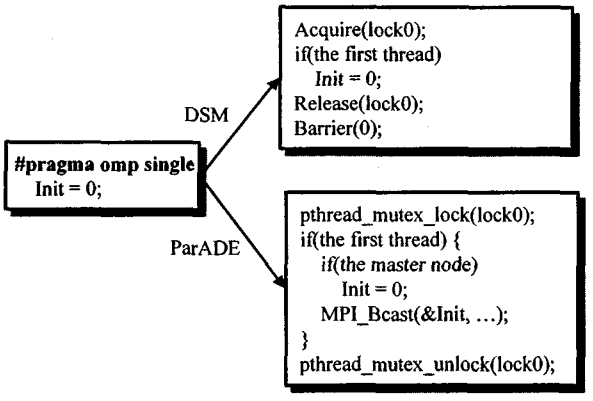

그림 3. single 디렉티브 변환 코드 예시

4. 실험

명시적인 메시지 전달 연산이 성능에 미치는 영향을 평가하기 위해, ParADE 시스템과 HLRC 기반의 소프트웨어 DSM 시스템에서 critical 디렉티브와 single 디렉티브의 성능을 비교한다. 실험을 위한 시스템은 두 개의 Pentium III 프로세서를 갖는 SMP 노드와 Fast Ethernet로 구성된 리눅스 클러스터이다. 각 노드에는 커널 2.4.18-3 SMP 버전의 레드헷 리눅스가 수행되고 있다. 각 응용 프로그램은 -o2 옵션의 gcc 컴파일러를 이용하여 컴파일한다. 두 디렉티브에 대한 변환 코드는 그림 2와 3에 제시되어 있고, 그림 4와 5는 각 디렉티브에 대한 수행 결과를 보여준다. 우리는 HLRC 기반의 DSM으로 KDSM[4]을 사용하였다.

Critical과 single 디렉티브 모두, ParADE 버전이 DSM 버전을 능가하고 그 차이는 노드의 수에 따라 늘어나는 추세이다. 이는 KDSM에서는 락을 얻기 위한 제어 메시지의 수와 움직이는 데이터의 양이 노드 수가 늘어남에 따라 증가하기 때문이다. 따라서, 동기화 디렉티브를 효과적으로 구현하는데 메시지 전달 함수를 활용하는 것이 바람직하다.

5. 결론 및 토의

이 논문에서 우리는 ParADE라 부르는 SMP 클러스터 시스템을 위한 OpenMP 기반의 병렬 프로그래밍 환경을 소개하였다. 특히, OpenMP 프로그램을 실행 시간 시스템 인터페이스로 변환하는 과정에서 명시적인 메시지 전달 함수를 활용하였다. 이로 인해, 락 메커니즘을 메모리를 접근하는 최상 경로에서 제거할 수 있고, 이는 병렬 프로그램의 성능에 가장 큰 영향을 미치는 동기화 디렉티브의 변환에 있어 기존의 DSM 기반의 방식보다 성능 향상을 가져왔다.

앞으로 큰 배열에 대한 데이터 지역성을 높이기 위한 다양한 기법을 모색해야 한다. 이 기법은 DSM 시스템에서 성능에 가장 큰 영향을 미치는 페이지 이동을 줄여 성능 향상을 가져올 것으로 예상한다.

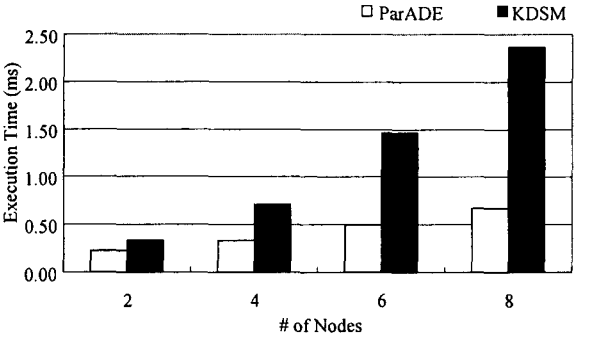

그림 4. critical 디렉티브의 성능 비교

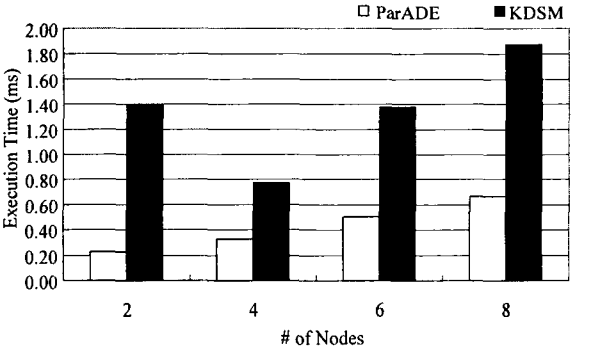

그림 5. single 디렉티브의 성능 비교

참고문헌

[1] OpenMP C and C++ Application Programming Interface, Version 1.0, http://www.openmp.org, Oct. 1998

[2] L. Iftode. "Home-based Shared Virtual Memory". PhD thesis, 1998.

[3] Mitsuhisa Sato, et al., Design of OpenMP Compiler for an SMP Cluster, In Proc. of EWOMP99, 1999.

[4] Hee-Chul Yun, et al., An Efficient Lock Protocol for Home-based Lazy Release Consistency, International Workshop on SDSM System, 2001