C 언어에서 자동 병렬 수행을 위한 부작용의 제거

이정호* 이갑래** 유원희*
*인하대학교 전자계산공학과
**김천과학대학 컴퓨터정보계열
e-mail : g1991274@inhavision.inha.ac.kr

Removal of side effects for the automatic parallelization in C language

Jung-Ho Lee*, Kab-Lae Lee**, Weon-Hee Yoo*
*Dept. of Computer Science and Engineering, In-ha University
**Dept. of Computer and Information, Kim-Cheun College of Science

요 약

프로그램 언어의 수행순서는 종속성으로 인해 결정된다. 병렬 수행을 위해서는 수행 단위 사이의 종속성을 제거해야 한다. 함수 간의 종속성을 발생시키는 주요 요인으로는 전역 변수가 있다. 본 논문의 자동 병렬 수행 시스템은 순차 C 언어 프로그램을 병렬 수행하여 순차 C 언어 프로그램과 동일한 결과를 내게 한다. 전역 변수를 위한 프레임이 프로세서 내의 지역 메모리에 할당되며 전역 변수의 최종 결정 값을 프로세서 간에 메시지로 전달하고 복사하여 전역변수의 부작용이 발생하지 않도록 한다. 또한 피호출 함수가 수행중인 호출 함수에서는 최종 결정된 전역 변수의 값을 피호출 함수로부터 받아오기까지는 전역 변수를 참조할 수 없고 봉쇄 상태가 되는데 피호출 함수가 복귀하지 않아도 전역 변수에 대해 더 이상의 값 변경이 없음을 알게 되면 곧바로 그 값을 호출 프로세서에 전달함으로써 전역 변수 참조로 인한 수행 지연을 최대한 줄이는 방법을 제안한다.

1. 서론

C 언어는 다양한 응용분야에서 사용될 수 있도록 충분한 제어문과 데이터를 구조화할 수 있는 특징을 제공하고, 높은 표현력을 허용하기 위해 풍부한 연산자를 제공한다. 1980 년대 이후로 C 언어의 인기가 급증하게 된 이유는 널리 사용되고 있는 UNIX 운영체제의 일부이기 때문이다[6].

프로그램 언어를 자동 병렬 수행하기 위한 많은 연구들이 있었다[1,2,4,5]. 프로그램을 다중 프로세서에서 수행시키는 것은 적지않은 수행 성능의 향상을 가져온다. 이러한 병렬 수행은 프로세서에 작업을 분배하는 방식에 따라 크게 두 가지로 나눌 수 있는데 데이터 분할 방식과 제어 분할 방식이 있다. 데이터 분할 방식은 프로그램이 사용하는 데이터를 분할하고 그 분할된 데이터를 동일한 작업을 하는 여러 프로세서에 분배하는 방식이고 제어 분할 방식은 프로세서들에 다른 작업을 분배하여 처리하게 한다. 본 연구는 제어 분할 방식인 함수 단위의 병렬 수행을 위한 것이다. C 언어의 자동 병렬 수행을 위해서는 병렬 수행 단위 사이의 종속성을 제거하는 것이 가장 중요한 문제이다.

본 논문에서는 순차적인 프로그램 실행 순서와 동일한 결과값을 내면서 함수 단위로 병렬 수행하기 위해서 공유 변수, 즉 전역 변수의 부작용(side effect)을 해결함으로써 함수 단위의 병렬 수행 성능을 높이는 방법을 제안한다.

2. 종속성

사건 A 가 발생하여야만 사건 B 가 발생할 수 있을 때 사건 B 가 사건 A 에 종속된다고 말한다. 이러한

관계를 종속성(dependency)이라고 하는데 종속성은 크게 데이터 종속성과 제어 종속성으로 나눌 수 있다. 데이터 종속성은 메모리 연산에 의해 발생하고 제어 종속성은 분기나 루핑이 끝나야만 다음 수행을 할 수 있을 때를 말한다.

데이터 종속성에는 흐름 종속성(flow-dependency), 역 종속성(anti-dependency), 출력 종속성(output-dependency)이 있다. 흐름 종속성은 어떤 연산이 수행되어야 다른 연산이 수행될 수 있는 경우의 종속성이다. 때때로 흐름 종속성은 너무 종속성이 크기 때문에 이전 연산의 수행이 완료되기를 기다리는 것 외에는 다른 방법이 없는 경우가 많다. 역 종속성은 특정 변수를 사용하는 연산이 끝나야만 그 변수의 값을 재정의할 수 있는 경우에 발생하는 종속성이다. 역 종속성을 해결하는 방법으로는 새로운 임시 변수에 값을 할당하는 것이다. 출력 종속성은 모든 계산이 끝났을 때 값이 옳은 변수에 들어가야 한다는 종속성이다[3].

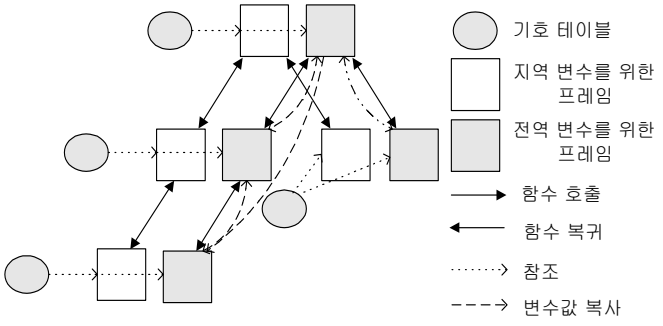
만약 함수 단위의 병렬 수행에서 함수 간에 공유 변수가 없고 함수의 복귀값도 없다면 함수 간에는 종속성이 없으므로 병렬 수행이 매우 효율적일 것이다. 그러나 공유 변수와 복귀값을 배정해야 하는 문제로 인해 값이 결정되기를 기다려야 한다.

본 논문에서는 특히 자동 병렬 시스템을 위한 프로세서간 공유 변수 문제 해결을 위한 메시지 전달 과정을 서술한다. 전역 변수를 프로세서 간의 메시지 전달을 통해 지역 메모리로 복사하여 사용하는 방법을 통해 다중 프로세서의 병렬 수행으로 인한 전역 변수의 부작용을 방지할 수 있다.

3. 프로세서 구조 및 메시지 전달 방법

3.1 프레임

본 연구에서는 프레임을 지역 변수를 위한 프레임과 전역 변수를 위한 프레임으로 구분하였는데 모두 프로세서의 지역 메모리에 할당한다. 지역 프레임은 순차 프로그램과 동일한 방식의 자료 구조를 갖지만 전역 변수를 위한 프레임은 각 변수마다 태그저장을 위한 공간을 추가로 갖는다.



[그림 1] 지역 메모리의 지역 프레임과 전역 프레임

두 개 이상의 호출 함수를 수행하는 프로세서가 2 개 이상일 때 [그림 1]과 같이 프레임은 트리 구조를 형성한다. 본 논문의 다중 프로세서 모델은 공유 메모리가 있고 각 프로세서가 지역 메모리를 갖는 모델이다.

두 개 이상의 자식 함수를 호출되어 수행중일 때 2 개 이상의 자식 노드를 갖는 트리 구조가 되는데 각 자식 노드는 호출 번호를 갖는다. 부모 함수는 호출 번호가 0 이고 자식 함수는 호출된 순서대로 1 부터 N 까지의 값을 갖는다.

3.2 기호 테이블

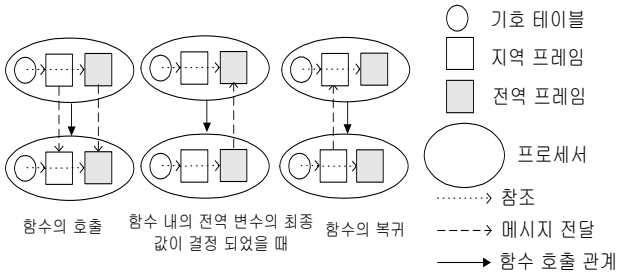
기호 테이블은 컴파일 시간에 소스 코드를 분석하여 생성되어 실행 시간으로 전달된다. 각 함수는 변수 참조를 위해서 기호 테이블을 가지며 전체 기호 테이블은 리스트로 연결되며 각 프로세서는 전체 기호 테이블의 리스트를 갖는다. 또한 각 프로세서는 전체 프로그램의 코드를 갖는다.

3.3 함수 호출 과정

컴파일 시간에 함수 이름은 함수 번호로 변환되며 함수 번호를 통해서 함수가 호출된다. 함수 호출은 함수 호출 메시지를 통하여 이루어지고 호출 메시지 안에는 매개변수의 값과 호출되는 함수가 필요로 하는 전역 변수의 값을 담고 있다. 호출되는 함수가 필요로 하는 전역 변수의 집합은 그 함수에 대한 기호테이블을 검색함으로 얻어낼 수 있다.

각 프로세서가 수행하는 함수를 알기 위해서 전역 메모리에 테이블을 유지하여 함수의 호출과 복귀시에 테이블의 정보를 변화시킨다.

만약 호출함수가 호출되는 함수에서 필요로 하는 전역 변수의 값을 갖고 있지 않다면 보다 상위 함수를 검색하여 전역 변수의 값을 얻어오도록 한다. 따라서 최상위 함수를 수행하는 최상위 프로세서는 최상위 함수의 전역 변수 뿐만 아니라 모든 전역 변수의 초기값을 갖고 있어야 한다.



[그림 2] 함수 호출과 복귀시에 값 메시지의 전달

[그림 2] 보는 바와 같이 값이 호출된 함수에 도달하였을 때 매개변수의 값은 지역 변수를 위한 프레임에 저장되고 전역 변수의 값은 전역 변수를 위한 프레임에 저장된다.

3.4 최종값 전달 메시지

컴파일 시간에 함수의 코드 분석을 통해서 더 이상

각 전역 변수 값의 갱신이나 현재 호출되어 수행중인 자식 함수가 없다는 두 가지 조건이 만족될 때 그 전역 변수를 얻어온 프로세서에 즉시 최종값을 담은 메시지를 전달하는 어셈블리 코드를 즉시 생성한다. 이를 위해서는 코드를 역으로 검색하는 과정을 필요로 한다. 만약 전역 변수의 값을 갱신하는 코드가 루프나 선택문 속에 있다면 최외각 루프와 선택문을 벗어날 때까지 기다렸다 부모 함수로 최종값을 보내야 한다. 부모 함수는 늦게 호출되어 수행중인 자식 함수와 그 전역 변수 값을 가져온 함수에 그 값을 보내어 준다. 이 방법을 통해 피호출 함수가 복귀하기 이전에 최종 전역 변수 값을 부모 함수에 전달하므로 그 전역 변수를 참조함으로써 봉쇄 상태에 들어간 부모 함수나 형제 함수가 보다 빨리 봉쇄 상태를 벗어난다

4. 함수간 종속성의 해결

4.1 함수간의 흐름 종속성과 역 종속성의 해결

피호출 함수가 수행중인 상태에서는 함수 간의 흐름 종속성으로 인해 전역 변수의 값을 참조할 수 없다.
그러나 지역 메모리에 전역 변수의 값을 사용하는 경우에 역 종속성으로부터 자유롭다.

```
Int a;
main()
{
        ①  A();
        if(조건)
        ②  Printf("%d",a);
        Else {
        ③  a=1;
        ④  printf("%d",a);
         }
        ⑤  B();
}
```

[예제 1]

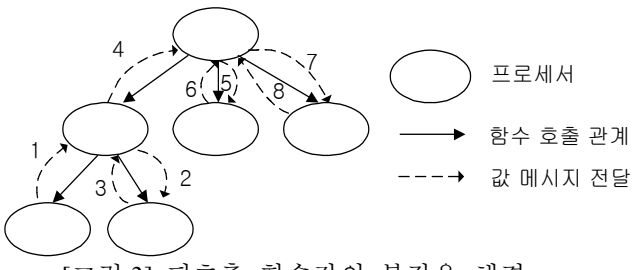
[예제 1]에서 함수 main 에서 함수 A 를 원격 호출하면 두 개의 프로세서에서 함수 main 과 함수 A 가 동시에 실행된다. 함수 main 의 실행 중 문장 ②에서 전역 변수 a 의 값을 참조할려고 할 때 함수 A 에서 변수 a 의 값을 보내올때까지는 a 의 값을 정확히 알 수 없으므로 이 문장을 수행할 수 없다. 바꿔 말하면, 함수 main 과 함수 A 는 흐름 종속성이 있다. 그러나 만약 조건이 거짓이 되어 else 구문이 수행된다면 문장 ③에 의해 그 흐름 종속성은 끊어지게 되므로 문장 ④를 수행하기 위해서 함수 A 에서 값을 보내오기를 기다릴 필요가 없이 문장 ⑤를 곧바로 수행할 수 있다.
만약 함수 main 과 함수 A 의 전역 변수가 같은 기억 공간을 사용한다면 문장 ③을 수행하고자 할 때 역 종속성이 발생하여 부작용을 야기한다. 그러나 전역 변수의 값의 저장장소가 지역 메모리이므로 역 종속성으로부터 자유롭다.

.4.2 함수간의 출력 종속성의 해결

자식 함수가 호출된 이후에라도 자식 함수를 호출한 함수가 수행 중에 전역 변수의 값을 갱신한다면 흐름 종속성이 끊어지므로 그 후에 나타나는 그 전역 변수의 참조로 인해서는 자식 함수가 값을 보내오기까지 기다릴 필요가 없다. 또한 출력 종속성으로 인해서 자식 함수가 보내온 최종값에 의해 그 전역 변수의 값이 변경되어서는 안된다. 전역 변수의 값이 변경될 때 그 변수가 갖는 태그에 바로 그 지점까지 호출했던 함수의 수를 기록한다. 각 자식 함수를 호출할 때는 부모 함수가 몇 번째 호출하는 피호출 함수인가 하는 번호를 함수 호출 메시지에 포함시켜 보내어 주고 전역 변수의 최종값을 보내올 때는 그 번호를 메시지에 포함시켜 보내오는데 호출 번호가 태그 값보다 클 때만 그 전역 변수의 값을 갱신하게 함으로써 출력 종속성을 해결한다.

4.3 함수 호출 깊이가 2 이상일 때 최종값 메시지 전달 과정

두 개 이상의 자식 함수가 실행 중일 때 보다 늦게 호출된 함수가 수행 중에 전역 변수의 값을 참조하는 문장을 만나면 먼저 호출된 함수가 그 전역 변수의 값을 변경시키는지 알 수 없으므로 그 전역 변수의 값을 참조할 수 없다. 그러므로 늦게 호출된 함수에서 전역 변수의 값을 참조할 때는 바로 이전에 호출된 자식 함수에서 최종값을 부모 함수에 전달하고 부모 함수가 다시 그 값을 그 다음 그 전역변수를 사용하는 호출된 함수에 보내어 올바른 전역 변수 값을 참조할 수 있게 한다.
[그림 3]에서는 모든 자식 함수가 해당 전역 변수를 사용한다고 가정할 때 가장 늦게 호출된 함수가 전역 변수 참조로 인해 봉쇄상태가 되었을 때 가장 먼저 호출된 함수로부터 전역 변수의 최종값이 전달되는 과정을 나타낸 것이다. 호출 함수가 그 함수를 호출한 함수로 전역 변수의 최종값을 전달 할 수 있는 때는 가장 늦게 호출된 함수, 즉 [그림 3]의 가장 오른쪽 그 전역 변수를 사용하는 자식 함수에서 최종값 전달 메시지를 받은 이후이다. 만약 어떤 피호출 함수의 코드에 전역 변수가 등장하지 않아서 전역 변수를 보내줄 필요가 없으면 다른 피호출 함수에 현재의 전역 변수 값을 바로 보낼 수 있다.



[그림 3] 피호출 함수간의 부작용 해결

만약 메시지에서 받은 값 중에서 부모 함수에서 사용하는 전역 변수에 해당 사항이 없고 그 부모함수의 하위 함수의 전역 변수에 대해서도 해당사항이 없으면 그 다음 상위 함수로 받았던 전역 변수의 값을 보내어 준다.

5. 결론 및 향후 연구 과제

본 논문은 함수를 수행하는 프로세서간의 메시지 전달을 통해 전역 변수 값을 지역 메모리에 복사하여 전역 변수로 인한 부작용을 해결하는 방법을 제안한다. 또한 전역 메모리에 대한 접근 횟수가 제한되므로 원격 메모리 참조로 인한 수행시간의 지연을 방지할 수 있다.

만약 복귀값이나 전역 변수를 사용하지 않는 경우 피호출 함수의 완전한 동시 수행이 가능하며 공유 변수가 있는 경우 함수 복귀보다 빨리 최종값을 부모 함수에 전달함으로 효율적인 원격 함수 호출 병렬 수행을 제공한다.

그러나 어느 프로세서에서 어떤 전역 변수를 사용하는가 검색하는 시간이 걸리므로 수행 시간이 늘어난다. 이런 문제에 대한 연구가 더욱 필요하다.

참고문헌

[1] F. Corbera. “New shape analysis technique for automatic parallelization of C codes,” ICS ’99 Rhodes Greece, ACM Press, June, 1999

[2] J. Hummel, L. J. Hendren and A. Nicolau. “A General Data Dependence Test for Dynamic Pointer-Based Data Structures,” Proceedings of the SIGPLAN Conference on Programming Language Desining and Implementation, pages 218-229, ACM Press, 1994

[3] Kevin Dowd, High Performance Computing, O’Reilly & Associates, Inc., 1999

[4] L. Eendren and A. Nicolau. “Parallelizing Programs with Recursive Data Structures,” IEEE Transations on Parallel and Distributed Systems,1(1):35-47, January 1990

[5] N. Jones and S.Muchnick. “Flow analysis and Optimazation of Lisp-Like Structures,” Program Flow analysis: Theory and Application, S.Muchnick and N. Jones, Englewood Cliffs, NJ:Prentice Hall, Chapter 4, 102-131, 1981

[6] Robert W. Sebesta., Concepts of Programming Languages, 4th Ed. Addison Wesley Longman, Inc., 1999