그래프 분할을 사용한 레지스터 할당의 성능 예측

김 원태*, 한 경숙*, 표 창우**
*홍익대학교 대학원 전자계산학과
**홍익대학교 컴퓨터공학과

Performance Estimation of Register Allocation using Graph Partitioning

Wontae Kim*, Kyungsook Han*, Changwoo Pyo**
*Dept. of Computer Science, Graduate School, Hongik University
**Dept. of Computer Engineering, Hongik University

요 약

그래프 분할을 사용한 레지스터 할당과 Chaitin의 레지스터 할당 방법의 성능을 비교하였다. 실험 데이터로 Appel이 제시한 간섭 그래프를 사용하였고, 각 알고리즘에서 요구되는 최소 레지스터 수를 비교하였다. 그 결과 그래프 분할을 사용한 방법에서 더 적은 수의 레지스터가 요구되었다. 가용 레지스터가 제한되어 있는 경우, 레지스터 요구 수가 감소되면 삽입되는 대피 코드의 수도 감소된다. 대피 코드의 발생이 줄어들면 메모리를 참조하는 인스트럭션의 수가 감소하여 실행시간을 단축시킬 수 있다. 따라서 컴파일러의 최적화 단계에서 그래프 분할 방법을 사용한 레지스터 할당으로 성능 향상을 기대할 수 있다.

1. 서론

컴파일러에서 레지스터 할당은 최적화 단계의 일부로 코드 생성 직전에 수행된다[6]. 컴파일러의 전단 부는 무한개의 가상 레지스터를 가지는 CPU를 가정하고 코드를 생성하게 된다. 레지스터 할당 단계에서 무한개의 가상 레지스터를 유한개의 실제 레지스터에 할당한다. 이 최적화 과정은 프로시저나 함수 단위로 수행된다.

레지스터 할당은 인접해 있는 노드들에게 가장 적은 색을 사용하여 서로 다른 색을 부여하는 그래프 컬러링 문제로 볼 수 있다. 컬러링을 하기 위해서 간섭 그래프의 노드는 가상 레지스터들의 생존 범위를 나타내고, 에지는 간섭 관계를 나타낸다. 가용 레지스터가 k개라고 할 때, 레지스터 할당은 k-컬러링과 동일한 문제가 된다. 그래프 컬러링은 NP-complete 문제이므로 휴리스틱을 사용하여 해결한다[5].

레지스터를 할당하는 방법으로 그래프 감축을 이용한 것과 그래프 분할을 이용하는 것이 있다. 그래프 감축을 이용한 방법으로 Chaitin이 제안한 레지스터 할당 방법이 사용되어 왔다[2,3]. 그래프 분할을 이용한 레지스터 할당 방법은 그래프 감축에 의한 방법과 달리 그래프를 독립집합으로 분할하여 레지스터를 할당한다[1].

그래프 분할을 이용한 레지스터 할당 방법에서의 성능과 Chaitin의 방법에서의 성능을 비교하기 위한 실험을 하였다. 성능 평가는 최소 레지스터 요구 수의 관점에서 이루어진다.

관련 연구

레지스터를 할당하는 방법으로 Chaitin은 그래프 감축에 의한 그래프 컬러링 방법을 제안했다[2,3].

Chaitin의 알고리즘은 전역자료 흐름 분석, 간섭그래프 구축, 노드 병합, 그래프 감축, 컬러링의 단계로 구분된다.

전역자료 흐름 분석단계에서는 중간 코드 변수들의 생존여부를 분석하여 생존 범위 정보를 구한다. 간섭 그래프 구축 단계에서는 생존 범위 정보를 이용하여 간섭 그래프를 구축한다. 노드 병합 단계에서는 간섭 관계에 있지 않는 두 노드가 복사 문에 의해 값을 전달할 때 두 노드를 합쳐 하나의 노드를 생성한다.

그래프 감축 단계는 가용 레지스터의 수인 k개의 색으로 컬러링 될 수 있는지 조사하는 단계이다. 간섭 그래프의 노드 중 에지가 k보다 작은 임의의 노드를 간섭 그래프에서 삭제하고 스택에 넣는다. 모든 노드가 삭제 가능하면 그 간섭 그래프는 k-컬러링이 가능하다. 만약 간섭 그래프에 남아 있는 모든 노드들의 에지 수가 k와 같거나 클 경우, k-컬러링을

위한 그래프의 감축이 불가능하게 되어 대피 코드가 삽입된다. 대피 코드가 삽입되면 간섭 그래프를 구축하는 단계부터 다시 수행된다. 이것은 간섭 그래프가 완전히 감축될 때까지 계속 반복된다.

컬러링 단계에서, 노드는 스택에 저장되었던 역순으로 간섭 그래프에 복구된다. 그때 인접한 노드에 할당되지 않은 색이 할당된다. 스택에는 간섭 관계가 k보다 작은 것들만 존재하므로 간섭 그래프에 대한 k 컬러링은 항상 가능하게 된다.

Chaitin의 알고리즘을 개선한 레지스터 할당 방법으로 Briggs의 방법이 있다[4].

Chaitin의 방법에서는 실제 레지스터 수 이상의 에지를 가지는 노드는 모두 대피시키므로 불필요한 대피를 발생시킬 수 있다. Briggs의 알고리즘에서는 대피시키는 시점을 그래프 감축 단계에서 컬러링 단계로 옮겼다. 실제 레지스터 수가 에지 수 이상이라 하더라도 컬러링이 가능하면 색을 할당받을 수 있도록 하였다. 그러나 색을 할당하는 과정에서, Chaitin 방식의 경우 스택에 컬러링이 가능한 것들만 존재하는 반면, Briggs의 방식에서는 그렇지 못하므로, 컬러링이 불가능한 노드는 색을 할당하지 않은 상태로 남겨 두고 컬러링 단계를 수행한다 컬러링 단계가 끝났을 때 모든 노드가 색을 할당받았다면 레지스터 할당이 성공적으로 수행된 것이다. 그렇지 않다면, 색을 할당하지 않은 노드들은 대피되며, 이 알고리즘은 생존 범위 정보를 구하는 단계에서부터 다시 시행된다.

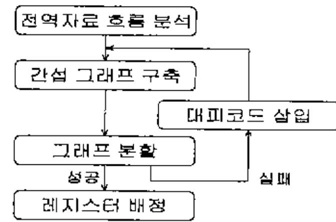
2. 그래프 분할에 의한 레지스터 할당 알고리즘

간섭 그래프의 노드들 중 간섭 관계가 없는 노드만으로 이루어진 집합을 독립집합이라 하고 독립집합의 특성을 위배하지 않고 추가 될 수 있는 노드가 더 이상 존재하지 않을 때, 그 독립집합을 최대 독립집합이라고 부른다. 최대 독립집합으로 그래프를 분할하는 것은 NP-complete문제이다.

그래프 분할에 의한 레지스티 할당 알고리즘이란 k개의 독립집합으로 그래프를 분할하여 n개의 가상 레지스터를 k개의 실제 레지스터로 할당하는 방법이다. 그러므로 그래프 분할에 의한 레지스터 할당 방법에는 독립집합을 구하는 휴리스틱이 필요하다[1].

그래프 분할에 의한 레지스터 할당 방법은 <그림 1>과 같다. 그래프 분할단계에서, 초기 분할은 하나가 만들어진다. 임의의 노드 n이 어떤 분할에서 그 분할의 모든 노드들과 간섭 관계가 없다면 n은 그 분할의 독립집합에 추가된다. 만약 n이 기존의 모든 분할에 추가되어 질 수 없다면 n을 담고 있는 새로운 분할을 생성한다. 이 방법으로 간섭 그래프의 모든 노드들은 독립집합으로 분할된다.

독립집합으로 분할된 개수가 실제 레지스터 개수인 k개와 같거나 작다면 그래프 분할을 이용한 레지스터 할당은 성공적으로 수행된다. 만약 그렇지 않다면, 대피 코드를 발생시킨



<그림 1> 그래프 분할을 이용한 레지스터 할당

후, 새로운 간섭 그래프를 구축해 위의 과정을 되풀이해야 한다. 대피코드를 발생시키기 위해서 분할된 독립집합 중 대피되어 삭제될 분할을 찾아야 한다. 발생한 분할의 개수를 p라 하고, 실제 레지스터의 개수를 k라 할 때, 대피되어 삭제될 분할은 생성된 순서대로 p-k개만큼 선택되어 지며, p-k 개의 분할에 속해 있는 변수들이 대피 될 대상이 된다. 대피될 대상의 변수들은 나머지 다른 분할로 이동 가능한지를 검사하여 이동 가능한 것은 이동시키고 그렇지 못한 것들만 실제로 대피 시켜 대피 코드를 발생시킨다.

삭제될 분할을 생성된 순서대로 선택한 이유는 대피 코드 발생 이전에 대피 대상의 코드들을 다른 분할로 이동 시켜 대피 코드를 줄이기 위함이다. 만약 최근 생성된 순서대로 삭제될 분할을 선택한다면 어떤 변수도 다른 분할로 이동시킬 수 없을 것이다.

3. 실험 및 결과분석

그래프 분할을 이용한 레지스터 할당 방법의 성능을 예측하기 위해서 실험을 통해 Chaitin의 레지스터 할당 방법과 비교했다.

실험을 위해 Appel이 제시한 간섭그래프 데이터를 사용하였다[7]. 이 데이터는 각 가상 레지스터의 간섭관계와 복사관계를 나타낸다. 이 데이터를 각각의 레지스터 할당 알고리즘에 적용하여 최소 레지스터 요구 수를 비교했다.

Appel의 데이터는 가상 레지스터의 생존 범위 정보와 정의·사용 정보를 포함하지 않으므로 대피 비용 계산이나 고려 노드 순서 계산을 위한 정보를 얻을 수 없다. 또한 대피시의 새로운 간섭 관계 계산이 불가능하므로 대피 시에는 간섭 관계가 삭제되는 것으로 가정하였다. 중간 코드에서 불필요한 복사 문을 제거하기 위해 노드를 병합하는 단계는 이번 실험에서는 고려하지 않았다.

<표 1>은 Appel의 그래프에[7] 대한 실험 결과이다. 이 결과를 보면, 그래프 분할을 이용한 방법이 Chaitin의 방법보다 더 적은 수의 레지스터를 요구함을 알 수 있다.

그래프 분할의 방법에서 그래프를 분할 할 때는 모든 노드들의 간섭관계를 파악하여 각 분할을 독립집합으로 만든다. 그리고 각 독립집합은 최대 독립집합을 지향하여 분할의 개

그래프 번호 /노드수	그래프 분할을 이용한 방법	Chaitin의 방법
G4048 / 83	10	12
G7660 / 78	7	9
G17801 / 185	11	13
G17855 / 820	20	23

< 표 1 > 실험결과, 최소 레지스터 요구 수

수를 최소화한다.

간섭 그래프에서 이웃하는 노드들이 같은 색을 할당받는 경우에는 간섭 수가 실제 레지스터 수 이상이 되어도 컬러링이 가능한 경우가 있다. 그러나 Chaitin의 방법에서는 삭제하려는 노드의 간섭 수만을 고려하여 그래프를 감축하므로 색을 할당받을 수 있는 경우에도 불필요한 대피를 발생시킨다. 그러므로 일반적인 경우에 그래프 분할의 방법이 Chaitin의 방법보다 더 좋은 성능을 보일 것으로 기대 된다.

4. 결론

그래프 분할을 이용한 레지스터 할당 방법과 그래프 감축을 이용한 Chaitin의 레지스터 할당 방법의 성능을 비교하기 위하여 실험적으로 각 방법이 요구하는 레지스터 수를 계산하였다. 이 실험을 위해 Appel의 간섭 그래프 데이터를 사용하였다. 그 결과 그래프 분할을 이용한 레지스터 할당 방법이 그래프 감축을 사용한 방법보다 적은 수의 레지스터를 요구함을 확인하였다. 그래프 감축 방법에서는 삭제하려는 노드의 간섭 수만을 고려하기 때문에 불필요한 대피가 발생된다. 그래프 분할을 이용한 방법에서 그래프 감축을 이용한 방법보다 더 좋은 성능을 보인 이유는 이러한 불필요한 대피를 발생시키지 않았기 때문이다.

대피 코드의 발생이 줄어들면 메모리를 참조하는 인스트럭션의 수가 감소하여 실행시간을 단축시킬 수 있다. 이러한 레지스터 할당 단계의 성능 개선은 컴파일러의 성능 향상에도 기여할 것으로 기대 된다.

향후의 연구에서는 이번 실험에서는 고려하지 않은 노드병합 단계를 고려하여 실험할 것이다. 그리고 최소 레지스터 요구 수 뿐 아니라 제한된 레지스터에서 발생하는 대피 코드와 같은 여러 기준에 대해 성능을 비교할 것이다.

5. 참고문헌

[1] 한경숙, 표창우, "그래프 분할을 이용한 레지스터 할당", Proceeding of The 21st KISS Fall Conference, 1994
[2] Gregory J, Chaitin, Mark A. Auslander, Ashok K. Chandra, John Coke, Martin E. Hopkins, Peter W. Markstein, "Register Allocation via Coloring", 1981
[3] Gregory J, Chaitin, "Register Allocation & Spilling via Graph Coloring", Proceedings of the ACM SIGPLAN '82 Symposium on Compiler construction, 1982
[4] P. Briggs, D. Cooper, K Kennedy, L. Torezon, "Coloring Heuristics for Register Allocation", ACM, 1989
[5] NARSINGH DEO, "GRAPH THEORY with applications to Engineering and Computer Science", Prentice-Hall, 1974
[6] Alfred V. Aho, Ravi Sethi, Jeffery D. Ullman, "Compilers, Principles, Techniques, and Tools", 1986
[7] Appel, A. Sample Graph Coloring Problems. URL:http://www.cs.princeton.edu/fac/appel/graphdata, 1996