Non-uniform에서 종속성 제거 방법을 이용한 프로시저 변환 알고리즘

송월봉*

요약

일반적으로 응용프로그램에서 병렬성 추출에 대한 핵심 부분은 루프이고 루프내의 첨자변수들 사이에는 자료종속성이 존재한다. 특히 문장들 사이에 가변 및 불변종속거리를 갖는 종속관계는 매우 복잡하다. 본 논문에서는 자료종속거리가 non-uniform 코드에 대해 적용 가능한 프로시저 변환 알고리즘을 제시하였으며 성능평가를 통해 제시된 방법이 효과적임을 보였다.

The Procedure Transformation Algorithm Using Dependency Elimination in Non-uniform

Worl-Bong Song*

ABSTRACT

Generally, In a application program the core part for parallel processing is a loop. Because the loop exists data dependencies between the array index variable of a loop. Specially, the data dependence relations between statements which of variable or constant dependence distance are very complex. In this paper propose procedure transformation Algorithm which can be apply to non-uniform code. A performance test results show the proposal scheme is superior to conventional method.

Key Words : data dependency, non-uniform, parallel processing, parallel loop

* 인천대학교 컴퓨터공학과(✉wbsong@incheon.ac.kr)
· 제1저자(First Author) : 송월봉 · 교신저자(Correspondent Author) : 송월봉
· 접수일(2011년 1월 26일), 수정일(1차 : 2011년 3월 8일), 게재확정일(2011년 3월 14일)

Ⅰ. 서 론

정보화 사회에서는 수많은 정보들이 있다. 이렇게 많은 정보들을 빠른 시간 안에 처리하기 위한 방법으로 병렬처리 시스템과 같은 더 빠른 컴퓨터를 만드는 노력들이 있었고, 최근에 상용화된 병렬처리 시스템들이 등장하였다. 또한 대규모 병렬처리 시스템들의 출현은 시스템 소프트웨어와 컴파일러에 새로운 기능을 추가하도록 요구하고 있으며, 각종 컴퓨터 응용 분야들에서도 병렬처리를 위한 새로운 알고리즘의 개발을 촉진하고 있다.

이러한 병렬화 기법 중에서 가장 기본적이고, 가장 많이 활용할 수 있는 부분은 순차프로그램에 있는 루프로부터 병렬성을 추출하여 병렬코드로 변환해주는 루프 재구조화 방법이다. 기존에 제시되어 있는 루프 변환 방법들은 종속거리가 불변인 경우를 처리하는 방법들과 가변인 경우를 처리하는 방법등으로 나누어 볼 수 있다.

종속거리가 불변인 경우는 interchanging[2], skewing[2], Chen & Wang[3]방법, Shang&Fortes의 방법[6], unimodular[5]등이 있으며 자료종속거리가 가변인 경우는 DCH[8]방법이 처음으로 제시되었고 IDCH[7]방법 등이 있다. 이러한 기존의 방법들은 모두다 종속성을 분석해서 분할한 다음 스케줄링하는 방법들을 택하고 있으며 이 방법들로는 효율성에서 한계점을 가지고 있다. 따라서 본 논문에서는 자료종속거리가 non-uniform 코드에 대해 적용 가능한 선형 변환 알고리즘을 제시하고, 성능평가를 통해 제시된 방법이 효과적임을 보이고자 한다.

Ⅱ. 관련연구

2.1 Diophantine 방정식

방정식 $ax + by = c$ (a≠0 또는 b≠0)의 정수해가 존재하기 위한 필요 충분 조건은 GCD $(a, b) \mid c$ 인 것이다.

또, $d = GCD(a, b)$, $d \mid c$ 일 때, (x_0, y_0)를 이 diophantine 방정식의 한 정수해라고 하면, 이 방정식의 모든 정수해 (x, y)는 다음과 같다.

$$x = x_0 + \frac{b}{d}t,\ y = y_0 - \frac{a}{d}t\ (t \in Z)$$

2.2 종속거리

L = L1,L2,...,Lm(m≥1)을 문장 S를 포함하는 루프(이때, 가장 바깥 루프부터 순차적으로 번호를 부여)라 할 때

① Lk(1≤k≤m)를 k차 레벨 루프라 하고,(L1,L2,...,Lm)를 문장 S를 포함하는 중첩 루프(nested loop)라 한다.

② 모든 루프 Lk의 제어는 Ik = tk, uk의 형태를 갖는다. 여기서 tk와 uk는 루프의 상한값과 하한값을 표시하고, Ik를 루프 변수라 한다.

③ 임의의 한 시점에서 S를 수행할 때 (Tk, Uk 그리고 ik를 각각 tk, uk 그리고 Ik의 값을 나타낸다고 하자), 벡터 $\hat{I}$=(i1,i2,...,im) ∈ Zn을 반복 벡터(iteration vector)라 하고 L의 한 반복(iteration)을 표현한다.

또한 모든 반복 벡터의 집합을 문장 S의 실행 인덱스 집합(execution index order)이라 하고 [S]로 표현한다.

따라서 m차 중첩 루프내의 문장 Sp와 Sq가 동일한 배열 변수를 포함하여 두 문장 Sp와 Sq의 데이타의 첨자식이 각각 [a1i1+b1,a2i2+b2,...,amim+bm], [c1i1+d1,c2i2+d2,...,cmim+dm] (여기서 ak,bk,ck,dk는 정수, ik는 루프 변수(1≤k≤m))와 같은 선형 결합 첨자(linear coupled subscript)를 갖는다면, 이중 임의의 한 k에 대한 두 문장사이의 종속 거리, Dk(Sp,Sq)는 다음과 같이 정의할 수 있다.

Dk(Sp,Sq) = | bk - dk |
if ak = ck : uniform
if ak ≠ ck : nonuniform

2.3 관련 기법들에 대한 분석

자료종속거리가 가변인 경우, 즉 DCH[8]방법과 IDCH[7]방법 등 기존 제안된 기법들에 대한 내용별 분석을 종합해보면 다음 표 1과 같다.

표 1. 기존 방법들에 대한 분석
Table 1. The analysis contents about the typical methods

측도 / 기존방법	적용범위	병렬 처리 방법	통합성	수행빈도수
DCH	가변 종속거리	분할	×	$N1 / 2$
IDCH	가변 종속거리	분할	×	$N1 * N2 / Tn$

III. 병렬성 추출 알고리즘

3.1 병렬성 추출을 위한 정의

(정의 1)

DMLCS(Dependence Matrix with the number of nested Loop, the number of Computation, and the number of Statement)는 종속성을 계산하기 위해 종속성을 표현하는 정방행렬이다.

DMLCS의 각각의 원소는 순서쌍으로서 순서쌍의 원소는 종속성을 갖는 문장과 문장 사이에 존재하는 첨자의 1차원 변수, 2차원 변수, ... , N차원 변수에 대한 종속성 표현이다.

DMLCS(k,l,m)은 중첩된 루프의 갯수가 k(≥ 1)개이고, 1은 종속성 행렬이 계산된 횟수로서 초기치는 1이다.

[m(≥ 2)은 루프안에 있는 문장의 갯수를 나타낸다.]

(정의 2)

((aij∈uij, bij∈vij), (cij∈wij, dij∈xij))는 문장 Si로부터 문장 Sj로 종속관계를 나타내고,

((akl∈ukl, bkl∈vkl), (ckl∈wkl, dkl∈xkl))이 문장 Sj로 부터 문장 Sk로 종속관계를 나타내면

$$\left(\begin{array}{c} ((a_{ij} \ominus u_{ij}, b_{ij} \ominus v_{ij}), (c_{ij} \ominus w_{ij}, d_{ij} \ominus x_{ij})) \\ ((a_{kl} \ominus u_{kl}, b_{kl} \ominus v_{kl}), (c_{kl} \ominus w_{kl}, d_{kl} \ominus x_{kl})) \end{array} \right)$$

는 문장 Si → Sj → Sk로 종속관계를 의미한다.

3.2 확장된 자료종속성 제거 방법을 이용한 선형 변환 알고리즘

[알고리즘 1]

통로의 길이가 1 인 경우 (정의1)에서 정의된 변수들을 이용하면 DOALL문은 다음과 같다.

(DMLCS의 원소가 0이 아닌 모든 원소에 대하여)

```
DOALL K=1, N
 IF Uij 와 Wij 가 1 THEN
  DOALL K=1, M-bij +1
   DOALL L=1, N-dij +1
     Si
   ENDDOALL
  ENDDOALL
 ELSE
  DOALL K=Uij, M, Uij
   DOALL L=Wij, N, Wij
     Si
   ENDDOALL
  ENDDOALL
ENDDOALL
```

[알고리즘 2]

통로의 길이가 2 인 경우 (정의1),(정의2)에서 정의된 변수들을 이용하면 DOALL문은 다음과 같다.

(DMLCS의 원소가 0이 아닌 모든 원소에 대하여)

```
DOALL K=1, N
  IF U_kl 와 W_kl 이 1 THEN
   DOALL K=b_kl, M
    DOALL L=d_kl, N
       S_j
    ENDDOALL
   ENDDOALL
  ELSE IF mU'_K1≠b_ij + nV_ij THEN
   DOALL K=b_ij, M, V_ij
    DOALL L=W'_kl, N, W'_kl
      S_j
    ENDDOALL
   ENDDOALL
  ELSE IF mW'_kl≠d_ij + nX_ij THEN
   DOALL K=U'_kl, M, U'_kl
    DOALL L=d_ij, N, X_ij
      S_j
    ENDDOALL
   ENDDOALL
  ELSE
   DOALL K=U'_kl, M, U'_kl
    DOALL L=W'_kl, N, W'_kl
      S_j
    ENDDOALL
   ENDDOALL
 ENDDOALL
```

두 문장 S_i와 S_{i+1}사이에 종속성이 존재하는 경우 먼저 두 문장 S_i 와 S_{i+1}를 동시에 수행시킨다. 그러면 문장 S_{i+1}의 a_{i+1} 값은 a_i의 값 때문에 올바르게 수행될 수도 있지만 올바르게 수행되지 않을 수가 있다. 이런 문제 때문에 종속성이 존재하면 병렬처리가 불가능하다. 이를 해결하기 위해서 두 번째로 문장 S_{i+1}만을 다시한 번 수행시키면 문장 S_{i+1}의 a_{i+1}값도 올바르게 수행된다. 그러나 문장 S_1이 S_i에 종속성을 갖고 문장 S_i가 문장 S_{i+1}에 종속성을 갖는다면 이 경우에는 위와 같이 수행해도 문장S_{i+1}의 a_{i+1}값은 아직도 올바르게 되지 않는다. 이 경우에는 한번 더 문장 S_{i+1}을 수행시켜야 문장 S_{i+1}의 a_{i+1}값도 올바르게 된다.

VI. 성능 평가

관련기법에서 분석한 기존의 방법 즉 DCH[8]방법 및 IDCH[7]방법과 본 논문의 알고리즘을 비교, 분석하고자 한다. 합당한 비교 분석을 위해서 기존의 논문[7,8]에서 제시한 예 즉 그림 1 예제를 가지고 하였다.

```
for I = 1, N1
   for J = 1, N2
      A(2*I+J+1, I+J+3) = …
                    … = A(2*j+3, I+1)
   endfor
endfor
```

그림 1. non-uniform에 대한 예
Fig. 1. A example of non-uniform

본 연구에서 제시한 방법을 적용한 병렬코드는 그림 2이다. 분석을 위해서 병렬코드의 실행수를 계산해 보면 다음과 같다.

일반적으로 DCH 방법은 λ=N1/2번 만에 수행할 수 있고 IDCH 방법은 λ=N1*N2/Tn (Tn은 Tile 수)만에 수행 할 수 있다. 그러나 본 논문에서 제시된 방법을 적용하면 병렬코드는 λ=1+ ⌈ Min(N1,N2)/4 ⌉ 번

만에 수행할 수 있다. 결론적으로 세가지 방법들을 비교해보면, N1과 N2는 바깥루프의 최종치와 안쪽루프의 최종치이므로 서로 교환해서 비교가 가능하다. 따라서 N1의 값을 고정하고 N2값을 50부터 200까지 증가시키는 실행 수에 대한 성능평가 를 백분율을 적용하여 표로 표시하면 다음과 같다. 즉 N1을 50부터 200까지 50씩 고정시키고 N2의 값을 50부터 200까지 가변시킨 경우 기존의 제안된 기법과 비교하여 본 논문에서 제안된 기법의 향상된 백분율은 표 2와 같으며 평균 77.8%의 성능 향상을 확인할 수 있다.

```
DOALL I=1, 10
  DOALL J=2, 10, 2
          A(2*I+J+1, I+J+3) = . . .
  ENDDOALL
ENDDOALL
DOALL K=1, 2
 IF K=1 THEN
    DOALL I=1, 10
      DOALL J=1, 10
          A(2*I+J+1, I+J+3) = . . .
      ENDDOALL
    ENDDOALL
 IF K=2 THEN
    DOALL I=1, 10
      DOALL J=1, 10
                . . . =A(2*j+3, I+1)
      ENDDOALL
    ENDDOALL
 ENDDOALL
```

그림 2. 본 연구에서 제시한 방법을 적용한 병렬코드
Fig. 2. Proposed parallel code

표 2. N1의 값을 고정시키고 N2의 값을 가변 시킨 경우
Table 2. Fixing the N1 value and the case of changing the N2 value

N1 값 \ 기존의방법	DCH	IDCH	평 균
50	44.0	99.3	71.6
100	54.0	99.4	76.7
150	60.0	99.5	79.7
200	67.0	99.6	83.3
평균	56.2	99.4	77.8

V. 결 론

병렬성 추출에 대한 핵심 부분은 루프이고 루프내의 첨자변수들 사이에는 자료종속성이 존재한다. 특히 문장들 사이에 가변 및 불변종속거리를 갖는 종속관계는 매우 복잡하다. 본 논문에서는 자료종속거리가 non-uniform 코드에 대해 적용 가능한 선형 변환 알고리즘을 제시하였으며 성능평가를 통해 제시된 방법이 기존의 방법들 보다 약 78%의 성능이 향상되었음을 확인하였다. 그러나 본 논문에서 제시한 방법은 프로세서의 수가 매우 많을 때 효과적인 방법이라는 제한점을 갖고 있다. 향후 과제로는 복수개의 프로시저간이나 프로시저내부에 프러시저를 포함하는 코드들에 대한 병렬처리[10]를 비교 분석하고 본 알고리즘의 효율성을 확인하고자 한다.

참고문헌

[1] Agustin Fernandez, Jose M. Llaberia, “Loop Tronsformation Using Non-unimodular Matrices”, IEEE Transaction on Parallel and Distributed Systems, August 1995.

[2] K.D cooper, M. W, Hall, L. Torczon, "An experiment with

inline substitution," Technical Report Tr90-128, Dept. of computer Science, Rice University, 1990.

[3] Keith D. Cooper, Ken Kenned y, and Linda Torcaon. "In-terprocedural constant propagation," Technical Report TR-85-29, Department of Computer Science, Rice University, 1985.

[4] Z. Li aand P. C. Yew, "Efficient interprocedural analysis for program restructuring for parallel programs," In Pro-ceedings of the SIGPLAN : Experience with Applications, Languages and Systems, 1998.

[5] Hollander, E. H., "Partitioning and labeling of loops by unimodular transformations", IEEE Trans. on Parallel and Distributed Systems,Vol. 3, No. 4, pp. 465-476, July 1992.

[6] Shang, W., M. T. O'Keefe, and J. A. B. Fortes, "On loop transformations for generalized cycle shrinking", Proceedings of nternational Conference Parallel Processing, pp. 132- 141, 1991.

[7] R. W. Scheifler. "An analysis of inline substitution for a structured programming language," Communications of the ACM, 1995.

[8] M. J. Wolfe, "High Performance Compilers for Parallel Computing," Addison-Wesley Publishing Company, 1995.

[9] 송월봉, 박두순, 김병수, 공용해, "Extracting parallelism in Nested Loops", Proceedings of International Computer Software & Applications Conference, IEEE, Seoul, pp. 41-47, Aug. 1996.

[10] 장유숙, 박두순, "자료종속성 제거 방법을 이용한 프로스저변환" 정보처리학회 논문지, 제9-A권 제1호, pp.37-44, 2002.

감사의 글

본 논문은 인천대학교 2010년도 자체연구비 지원에 의하여 연구되었음.

저자소개

송월봉(Worl-Bong Song)

1974년 숭실대학교 공학사
1982년 한양대학교 공학석사
1998년 순천향대학교 공학박사

2010년~현재 인천대학교 컴퓨터공학과 교수
※ 관심분야 : 병렬처리, 컴파일러, 알고리즘