불변 및 가변 종속거리를 위한 최적 병렬알고리즘

송 월 봉
(인천전문대학 전자계산과 교수)

W.B. Song
(Junior College of Inchon)

초 록

중첩 루프의 전체적인 병렬화를 하기 위해서 자료 종속을 효과적으로 제거하는 알고리즘이다. 즉 순차 루프를 중첩된 DOALL루프로의 자동 변환에 대한 절차이다.

Ⅰ. 서 론

재구성 컴파일러는 프로그램의 병렬성(parallelism)을 추출하기 위해서 자료 종속성(data dependency)을 분석한다. 기존의 재구성 컴파일러는 자료 종속 거리가 모두 불변(constant)인 경우에 제한되어서 처리된다. 일반적인 프로그램의 경우 이를 분석해보면 자료 의존성이 사이클을 이루거나 반종속, 출력종속이 섞여있는 경우가 대부분이므로 기존의 재구성 컴파일러 가지고는 해결할 수가 없다. 이를 위하여 병렬 컴퓨터(parallel computer)에서 꼭 필요로 하는 자료종속의 거리가 불변만 있거나, 가변(배열 첨자들이 a*i+c(a와c는 상수))만 있거나, 불변과 가변이 섞여져 있는 경우에도 처리할 수 있고 자료 의존성의 사이클을 이루거나 반종속, 출력종속이 섞여져 있는 일반 프로그램에 적용할 수 있는 재구성 컴파일러의 기법이 필요하다.

본 논문에서는 재구성 컴파일러를 구성하기 위해서 우선적으로 필요한 자료종속의 거리가 가변만 있는 경우, 불변만 있는 경우, 가변과 불변이 섞여져 있는 경우에도 모두 적용할 수 있는 새로운 알고리즘을 제시하고자 한다.

Ⅱ. 병렬성 추출 알고리즘

루프는 프로그램 내에서 병렬처리에 이용될 수 있는 가장 중요한 근원이며, 따라서

프로그램 재구조화 기법은 루프내에서 수행된다. 루프내의 배열들 간에 자료종속이 존재하게 되며, 이러한 자료 종속들 때문에 그 루프는 완전히 순차적이거나 혹은 부분적으로만 병렬화 될 수 있다.

이 장에서는 정의와 병렬성 추출 알고리즘을 제시한다.

[정의 1]

종속성 행렬 DM(Dependence Matrix with the number of nested Loop, the number of Computation, and the number of Statement)은 자료 종속성을 계산하기 위해 중첩된 루프들의 모든 첨자들을 이용해서 자료 종속성을 표현하는 정방행렬이다. DM의 각각의 원소는 순서쌍으로서 순서쌍의 원소는 종속성을 갖는 문장과 문장 사이에 존재하는 첨자의 1차원 변수, 2차원 변수, ...N차원 변수에 대한 종속성 표현이다. 또한, DM(k,l,m)은 중첩된 루프의 갯수가 k(≥1)개 이고, 1은 종속성 행렬이 계산된 횟수로서 초기치는 1이다. m(≥2)은 루프안에 있는 문장의 갯수를 나타낸다.

DM(2,1,n)은 루프안에 있는 문장은 n개이고 2개의 중첩된 루프이고, 종속성 행렬의 초기상태를 나타내는 것으로 다음과 같다.

$$
\mathrm{DMLCS}(2,1,n) = \begin{bmatrix}
((a_{11}\oplus u_{11}, b_{11}\oplus v_{11}),(c_{11}\oplus w_{11}, d_{11}\oplus x_{11})) & \cdots\cdots & ((a_{1n}\oplus u_{1n}, b_{1n}\oplus v_{1n}),(c_{1n}\oplus w_{1n}, d_{1n}\oplus x_{1n})) \\
((a_{21}\oplus u_{21}, b_{21}\oplus v_{21}),(c_{21}\oplus w_{21}, d_{21}\oplus x_{21})) & \cdots\cdots & ((a_{2n}\oplus u_{2n}, b_{2n}\oplus v_{2n}),(c_{2n}\oplus w_{2n}, d_{2n}\oplus x_{2n})) \\
\vdots & \vdots & \vdots \\
((a_{n1}\oplus u_{n1}, b_{n1}\oplus v_{n1}),(c_{n1}\oplus w_{n1}, d_{n1}\oplus x_{n1})) & \cdots\cdots & ((a_{nn}\oplus u_{nn}, b_{nn}\oplus v_{nn}),(c_{nn}\oplus w_{nn}, d_{nn}\oplus x_{nn}))
\end{bmatrix}
$$

단, ⊕는 첨자식 sI+t의 형태에서 상수 s와 t를 한번에 표시하기 위한 기호이며, 임의의 i, j($1 \le i, j \le n$)대해 $((a_{ij}\oplus u_{ij}, b_{ij}\oplus v_{ij}),(c_{ij}\oplus w_{ij}, d_{ij}\oplus x_{ij}))$는 문장 S_i로 부터 문장 S_j로 자료 종속성이 존재함을 나타내고, 0은 자료 종속성이 존재하지 않음을 나타낸다. 또한 문장 S_i로 부터 문장 S_j로 자료 종속성이 존재하는 경우에 (a_{ij}, b_{ij}) 는 중첩 루프의 첫 번째 첨자식들 간의 diophantine방정식 $v_{ij} \times I + g = u_{ij} \times P + h$의 초기해이고 (c_{ij}, d_{ij})는 중첩 루프의 두 번째 첨자식들 간의 diophantine방정식 $x_{ij} \times J + g' = w_{ij} \times Q + h'$의 초기해이다. @$((a_{ij}\oplus u_{ij}, b_{ij}\oplus v_{ij}),(c_{ij}\oplus w_{ij}, d_{ij}\oplus x_{ij}))$는 불변 종속거리를 나타내며 # $((a_{ij}\oplus u_{ij}, b_{ij}\oplus v_{ij}),(c_{ij}\oplus w_{ij}, d_{ij}\oplus x_{ij}))$는 수정된 첨자의 상계(upper bound)이다.

예를 들어 <그림 1>과 같은 프로그램이 있다고 하자.

```
   DO I = 1 , N
    DO J = 1 , N
S1 :  A(I-2, J-1) = B(4*I, 3*J) + C(4*I, 5*J-2)
S2 :  B(5*I, 4*J) = A(I-4, J-4) + B(I-4, 3*J) +
                    C(3*I-1, 4*J-2)
S3 :  C(7*I, 6*J) = A(I-5, J-3) + B(I-2, J-3) +
                    C(I-1, I-2)
    ENDDO
   ENDDO
```

<그림 1> 중첩루프의 예

<그림 1>은 2개의 중첩된 루프를 가졌고, 루프 안에 있는 문장의 갯수는 3개이다. 이에 대한 종속성 행렬의 초기상태는 DM(2,1,3)이다. 여기서 변수 B에 대해서 문장 S_2로 부터 문장 S_3으로 가는 흐름 종속이 존재한다. 이 경우에 자료 종속성은 5*I와 P-2의 값이 같아지는 정수해 I,P의 값을 구해야 하고 4*J와 Q-3의 값이 같아지는 정수해 J,Q값을 구해야 한다. 이에 대한 초기해를 DM(2,1,3) 행렬의 2행 3열에 나타내면 되는 것이다. <그림 1>의 DM(2,1,3)은 <그림 2>이다.

	S_1	S_2	S_3
S_1	0	@((1⊕1,3⊕1),(1⊕1,4⊕1))	@((1⊕1,4⊕1),(1⊕1,3⊕1))
S_2	((0⊕4,0⊕5),(0⊕3,0⊕4))	((1⊕1,9⊕5),(0⊕3,0⊕4))	((1⊕1,7⊕5),(1⊕1,7⊕4))
S_3	((0⊕4,0⊕7),(3⊕5,4⊕6))	((2⊕3,5⊕7),(1⊕4,2⊕6))	((1⊕1,8⊕7),(1⊕1,8⊕6))

<그림 2> <그림 1>의 DM(2,1,3)

※ <그림 2>에서 문장 S_2에서 문장 S_1으로의 자료종속성이 존재하고 초기치는 ((0⊕4, 0⊕5),(0⊕3, 0⊕4))이다.

<알고리즘 1>

1, FOR 0이 아닌 모든 a_{ij}, c_{ij}에 대해서

IF 통로의 길이 = 1 THEN

IF a_{ij} 혹은 c_{ij} 가 상수 THEN

DOALL K=1, M-b_{ij} +1

```
      DOALL L=1, N-dij +1
                Si
        ENDDOALL
       ENDDOALL
      ELSE
        DOALL K=aij, M, aij
         DOALL K=cij, N, cij
                Si
         ENDDOALL
        ENDDOALL
ELSE IF 통로의 길이 = 2 THEN
   IF  aij 혹은 cij 가 상수 THEN
        DOALL K=bij, M
         DOALL L=dij, N
                Sj
         ENDDOALL
        ENDDOALL
       ELSE
        DOALL K=a'K1, M, c'K1
         DOALL K=c'K1, N, c'K1
                SJ
         ENDDOALL
        ENDDOALL
      ELSE
      IF  auv 혹은 cuv 가 상수 THEN
        DOALL K=buv + bxy - 1, M
         DOALL L=duv + dxy - 1, N
                Sv
         ENDDOALL
        ENDDOALL
      ELSE
        DOALL K=a'xy, M, a'xy
         DOALL K=c'xy, N, c'xy
                S1
         ENDDOALL
       ENDDOALL
```

<알고리즘 2> DMLCS의 변형 및 새로운 자료 종속을 가지는 변수계산

1. LCM은 최소공배수이다.
2. IF 통로의 길이가 2이고

$$\text{DMLCS}=\begin{bmatrix}(a_{ij}\oplus u_{ij},b_{ij}\oplus v_{ij}) & (c_{ij}\oplus w_{ij},d_{ij}\oplus x_{ij})\\ (a_{kl}\oplus u_{kl},b_{kl}\oplus v_{kl}),(c_{kl}\oplus w_{kl},d_{kl}\oplus x_{kl})\end{bmatrix}$$

THEN

$a'_{kl}=LCM(b_{ij}\oplus v_{ij},a_{kl}\oplus u_{kl})$

$c'_{kl}=LCM(d_{ij}\oplus x_{ij},c_{kl}\oplus w_{kl})$

DMLCS의 변형 =

$$\begin{bmatrix}(a_{kl}\oplus u_{kl},b_{kl}\oplus v_{kl}) & (c_{kl}\oplus w_{kl},d_{kl}\oplus x_{kl})\\ (a'_{kl},a'_{kl}\times b_{kl}/a_{kl}) & (c'_{kl},c'_{kl}\times d_{kl}/c_{kl})\end{bmatrix}$$

3. IF통로의 길이가 3 이상이고

$$\text{DMLCS}=\begin{bmatrix}(a_{kl},\ b_{kl}) & (c_{kl},\ d_{kl})\\ (a_{uv},\ b_{uv}) & (c_{uv},\ d_{uv})\\ (a_{xy},\ b_{xy}) & (c_{xy},\ d_{xy})\end{bmatrix}$$

THEN

$a'_{xy}=LCM(b_{uv},\ a_{xy})$

$c'_{xy}=LCM(d_{uv},\ c_{xy})$

DMLCS의 변형 =

$$\begin{bmatrix}(a_{xy},\ b_{xy}) & (c_{xy}, & d_{xy})\\ (a'_{xy},a'_{xy}\times b_{xy}/a_{xy}) & (c'_{xy},c'_{xy}\times d_{xy}/c_{xy})\end{bmatrix}$$

<알고리즘 3>

1. 통로의 길이가 1인 DMLCS를 만든다.
2. IF DMLCS에 첨자값이 같은 것이 있다면
   THEN 다른 이름으로 대체한다.

3. For all i, j

IF(a_{ij}⊕u_{ij}) > M 혹은 (c_{ij}⊕w_{ij}) > N

THEN 그 원소는 0으로 치환한다.

4. 모든 0이 아닌 원소에 대해 <알고리즘 1>을 적용하라.
5. IF b_{ij} > M 혹은 d_{ij} > N THEN 그 원소는 0으로 치환한다.
6. IF 모든 원소 =0 THEN goto 12

ELSE 통로의 길이를 1 을 증가시킨다.

7. IF 똑같은 원소가 존재한다면

THEN 하나만 남겨놓고 모두 제거한다.

8. DMLCS에서 <알고리즘 2>를 적용하여 아직도 남아있는 자료 종속을 찾고 DMLCS를 변형한다.
9. IF LCM을 취한 변수의 값이 DO루프의 최종치보다 크면

THEN 그 원소는 0으로 치환한다.

10. 모든 0이 아닌 원소에 대해

<알고리즘 1>을 적용하라.

11. Goto 6
12. IF 첨자값이 바뀐 원소가 존재

THEN <알고리즘 1>을 적용하라.

13. 변형된 문장을 제외한 모든 문장에 대해 doall S_i를 수행한다.

III . 성능 평가

이 알고리즘은 불변 거리 종속과 가변 거리 종속이 모두 존재하는 경우에 아주 뛰어난 기법이기 때문에 가변 거리 종속과 불변 거리 종속이 있는 예를 다루고자 한다.

[예] : 다음과 같이 가변 거리 종속과 불변거리 종속이 혼합되어 있는 경우를 고려해 보자

```
      DO I = 1, 20
      DO J= 1, 100
S1  :  A(I-2, J) = B(4*I, 3*J)
```

```
S2  :  B(5*I, 4*J) = A(3*I, J-4) + B(I-4, 3*J) + C(3*I, 4*J)
S3  :  C(7*I, 5*J) = A(I-5, J-3)
       ENDDO
       ENDDO
```

<그림 3>의 DM(2,1,3)는 다음과 같다.

	S_{11}	S_2	S_3
S_1	0	0	@((1⊕1,4⊕1),(1⊕1,4⊕1))
S_2	((0⊕4,0⊕5),(0⊕3,0⊕4))	((1⊕1,9⊕5),(0⊕3,0⊕4))	0
S_3	0	((0⊕3,0⊕7),(0⊕4,0⊕5))	0

<그림 3>에 자료 종속성 제거 알고리즘을 적용한 병렬 코드는 <그림 4>이다. 여기서, 변수 AA는 반 종속을 가지는 변수 A의 재이름(rename)이다.

```
DOALL K=1, 4
  IF K=1 THEN
    DOALL I = 1, 20-4+1
      DOALL J = 1, 100-4+1
            A(I-2, J) = B(4*I, 3*J)
      ENDDOALL
    ENDDOALL
  IF K=2 THEN
    DOALL I = 4, 20, 4
      DOALL J = 3, 100, 3
            B(5*I, 4*J) = AA(3*I, J-4)
              + B(I-4, 3*J) + C(3*I, 4*J)
      ENDDOALL
    ENDDOALL
  IF K=3 THEN
    DOALL I = 1, 20
      DOALL J = 3, 100, 3
            B(5*I, 4*J) = AA(3*I, J-4)
              + B(I-4, 3*J) + C(3*I, 4*J)
      ENDDOALL
    ENDDOALL
  IF K=4 THEN
```

```
    DOALL I = 3, 20, 3
      DOALL J = 4, 100, 4
            C(7*I, 5*J) = A(I-5, J-3)
      ENDDOALL
    ENDDOALL
ENDDOALL
DOALL K=1, 4
  IF K=1 THEN
    DOALL I = 3, 20, 3
      DOALL J = 4, 100, 4
            C(7*I, 5*J) = A(I-5, J-3)
      ENDDOALL
    ENDDOALL
  IF K=2 THEN
    DOALL I = 9, 20, 5
      DOALL J = 12, 100, 12
            B(5*I, 4*J) = AA(3*I, J-4)
              + B(I-4, 3*J) + C(3*I, 4*J)
      ENDDOALL
    ENDDOALL
  IF K=3 THEN
    DOALL I = 5, 20, 5
      DOALL J = 4, 100, 4
            A(I-2, J) = B(4*I, 3*J)
      ENDDOALL
    ENDDOALL
  IF K=4 THEN
    DOALL I = 7, 20, 7
      DOALL J = 15, 100, 15
            B(5*I, 4*J) = AA(3*I, J-4)
              + B(I-4, 3*J) + C(3*I, 4*J)
      ENDDOALL
    ENDDOALL
ENDDOALL
    DOALL I = 7, 20, 7
      DOALL J = 15, 100, 15
            B(5*I, 4*J) = AA(3*I, J-4) +
              B(I-4, 3*J) + C(3*I, 4*J)
      ENDDOALL
    ENDDOALL
```

```
DOALL K=1, 3
  IF K=1 THEN
    DOALL I = 1, 20
      DOALL J = 1, 100
            A(I-2, J) = B(4*I, 3*J)
      ENDDOALL
    ENDDOALL
  IF K=2 THEN
    DOALL I = 1, 20
      DOALL J = 1, 100
            B(5*I, 4*J) = AA(3*I, J-4) +
              B(I-4, 3*J) + C(3*I, 4*J)
      ENDDOALL
    ENDDOALL
  IF K=3 THEN
    DOALL I = 1, 20
      DOALL J = 1, 100
            C(7*I, 5*J) = A(I-5, J-3)
      ENDDOALL
    ENDDOALL
ENDDOALL
```

<그림 4> <그림 5>에 대한 병렬 코드

<그림 3>에 있는 프로그램에 대한 성능평가는 순차적인 실행방법과 본 논문에서 제시한 방법을 적용한 결과인 <그림 4>을 병렬 실행하는 것에 의해서만 성능평가가 가능하다. <그림 3>에 있는 프로그램을 순차적으로 실행할 경우에는 6000번을 수행해야 하지만 본 논문에서 제시한 방법을 적용한 <그림 4>을 병렬수행하면 단 4번만에 실행이 가능하다. 즉, 본 논문에 의한 방법이 1500배 정도 빠름을 알 수 있다. 이러한 경우처럼 한가지 예만을 가지고 비교할 수는 없지만 다른 예들도 모두 실행속도면에서 매우 빨라짐을 알 수 있다. 결국 본 논문에서 제시한 방법은 아주 성능이 뛰어난 알고리즘이라고 할 수 있다.

Ⅳ. 결 론

본 논문에서는 병렬 처리 시스템에 적용할 수 있는 새로운 알고리즘을 제시하였다. 이 방법은 기존의 실행 시간에만 처리할 수 있었던 병렬성 검사및 병렬 코드로의 변환

을 컴파일 시간에 처리함에 따라서 수행 시간을 매우 단축시킬 수 있다.

앞으로 이 방법은 병렬 처리하는데 있어서 분할 및 스케쥴링 방법에도 많이 사용될 수 있으리라 생각한다.

참 고 문 헌

[1] U. Banerjee, "Dependence Analysis for Supercomputing", Kluwer Academic Pub. 1988

[2] H. Cheng, "Vector Pipelining, Chaining and Speed on the IBM 3090 and Cray X-MP", IEEE Comp. pp.31-46, Sep. 1989

[3] X. Kong, D. Klappholz, K. Psarris, "The I Text: An Improved Dependence Test for Automatic Parallelism and Vectorization." IEEE Trans. on Parallel and Sistributed Systems. Vol. 2, No. 3, July 1991

[4] Z. Li, P. C. Yew, C. Q. Zhu, "An Efficient Data Dependence Analysis for Paralleling Compilers." IEEE Trans. on Parallel and Distributed Systems, Vol. 1, No. 1, pp. 26-34, Jan. 1990

[5] A. Malony, J. Larson, D. Read, "Tracing Application Program Execution on the Cray X_MP and Cray-2", CSRD Report No. 985, CSRD Univ. of Illinois at Urbana-Champaign, Nov. 1990

[6] V. Msalov, "Delinearization: An Efficient Way to Break Multiloop Dependence Equations." ACM SIGPLAN, June 1992

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

[8] P. Tang, P. C. Yew, C. Q. Zhu, "Compiler Techniques for Data Synchronization in Nested Parallel Loops", Proc. of the ACM Intl. Conference on Supercomputing, pp. 176-181, July. 1990

[9] M. E. Wolfe, "Optimizing supercompiler for supercomputing." Ph. D. Thesis, Report NO. 82-1105, CSRD Univ. of Illinois at Urbana-Champaign, 1982

[10] M. E. Wolfe, C. W. Tseng, "The Power Test for Data Dependence." IEEE Trans. on Parallel and Distributed Systems, Vol. 3, No. 5, Sep. 1992