論文
4-4-8

병렬처리를 위한 프로그램 재구조화 (A Program Restructuring framework for Parallel Processing)

송 월 봉(Worl-Bong Song)[1]

요 약

본 논문에서는 자료종속성제거와 관련해서 선형루프변환의 새로운 이론을 제안한다. 이는 선형루프변환이 병렬성 추출의 주요한 대상이 되기 때문이다. 이러한 이유 때문에 완전한 중첩루프에서 최대의 루프 병렬성을 추출하는 방법이 제안되었다. 제안된 이론은 거리와 방향에 관계없이 종속성을 갖는 일반적인 중첩루프에 적용할 수 있다.

ABSTRACT

In this paper A new theory of linear loop transformation called Elimination of Data Dependency(EDD) is presented. The current framework of linear loop transformation cannot identify a significant fraction of parallelism. For this reason, a method to extract the maximum loop parallelism in perfect nested loops is presented. This technique is applicable to general loop nests where the dependence include both distance and directions.

1. 서론

순차 프로그램을 자동적으로 병렬 프로그램으로 변환하는 것이 재구성 컴파일러이다. 재구성 컴파일러는 일반 컴파일러의 전단계로서 순차 프로그램에서 병렬성을 탐지하고, 다중 처리기에서 효율적으로 처리하게 함으로써 컴퓨터 성능을 높이는 방법이다. 이에 대한 연구는 프로그램 수행 시간의 대부분이 루프 구조에서 소비되고 있기 때문에 순차 프로그램을 병렬 프로그램으로 변환하는 연구의 대부분이 루프 구조의 변환에 치중되고 있다[4, 8].

루프 구조의 변환에 관한 연구는 루프내에 존재하는 데이터 종속성을 분석하는 연구가 필수적이다.

일반적인 프로그램의 경우 이를 분석해 보면 자료 존송성이 사이클을 이루거나 역 종속성, 출력 종속성이 섞여 있는 경우가 대부분이므로 기존의 재구성 컴파일러로는 해결 할 수가 없다. 이를 위하여 병렬 컴퓨터에서 필요한 자료 종속성의 거리가 불변(uniform)이거나 가변(non-uniform)인 경우에도 처리할 수 있고, 자료 의존성이 사이클을 이루거나 역 종속성, 출력 종속성이 모두 섞여져 있는 일반 프로그램에 적용할 수 있는 재구성

1) 정회원 : 인천전문대학 컴퓨터정보과 교수

논문심사 : 2003. 6. 2.
심사완료 : 2003. 6. 23.

※ 본 논문은 시립인천전문대학 2002년도 연구지원비에 의한 것임.

컴파일러의 기법이 필요하다.

선형 변환의 종류는 loop interchange, loop skewing, loop reversal, loop scaling, loop distribution, cycle shrinking 그리고 loop tiling등이 있다[6, 7].

본 논문에서는 자료 종속성의 거리가 불변인 경우에 적용할 수 있는 프로그램 재구조화를 제시하고, 병렬 처리에서 가장 많이 등장하는 예[3, 5]를 들어 기존의 방법들과 성능 평가한다.

2. 기본 사항

본 절에서는 루프를 재구조화 하기위하여 필요한 기본 사항으로 프로그램 종속과 기존의 선형변환 방법들을 살펴본다. 프로그램의 종속성은 자료 종속성(data dependence)과 제어 종속성(control dependence)으로 나눈다. 제어 종속성은 조건 분기에 의해서 발생되는 종속성으로 자료 종속성과 유사한 방법으로 처리할 수 있기 때문에 본 논문에서는 자료 종속성에 대해서만 다룬다.

```
DO  I = 3, N1
  DO  J = 5, N2
    A(I, J) = B(I-3, J-5)
    B(I, J) = A(I-2, J-4)
```

[그림 1] 불변거리에 대한 예
[Fig. 1] The example of uniform

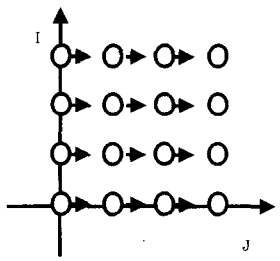
2.1 Unimodular 변환

Unimodular 변환은 unimodular matrix에 의해 표현될 수 있는 변환이다. Unimodular matrix의 유리한점은 "*phase-ordering problem*" 즉, 루프 변환의 순서를 쉽게 결정할 수 있다는 것이다. Unimodular matrix는 3가지 특성을 갖는다. 첫째, n X n의 정사각형 matrix이다. 둘째, 모든 요소가 정수이다. 셋째, 행렬식이 ±1이다. 이러한 특성 때문에 2개의 unimodular matrix의 곱과 역도 unimodular이고, unimodular 루프 변환의 혼합과 unimodular 루프 변환의 역도 unimodular 루프이다[1, 2].

Unimodular matrix로 표현할 수 있는 기본적인 변환은 loop interchanging, loop skewing, loop reversal이고, 병렬화를 최대화하기 위하여 세가지를 통합하는 방법을 사용한다. unimodular 변환으로 재구성하는 것을 서술하면 다음과 같다.

우선, Fourier-Motzkin 방법을 사용하여 루프 한계 값 Lk,l, Uk,l을 계산한다. [그림 2]는 이중 루프의 예제 프로그램과 반복 공간을 보여준다.

```
for I = 1 to N step 1
  for J = 1 to N step 1
    ...
  endfor
endfor
```



[그림 2] 이중 루프 프로그램과 iteration space
[Fig.2] Double loop program and iteration space

행렬 부등식으로 표현하면,

$$\begin{matrix} 0 \le I \le N \\ 0 \le J \le N \end{matrix} \quad \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} I \\ J \end{bmatrix} \le \begin{bmatrix} 0 \\ N \\ 0 \\ N \end{bmatrix}$$

unimodular 변환 $T = \begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix}$ 에 따라

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 0 & 1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{matrix} i \\ j \end{matrix} \le \begin{matrix} -1 \\ N \\ -1 \\ N \end{matrix}$$

삼각시스템으로 재구성하기 위하여,

Fourier-Motzkin 방법을 사용하면,

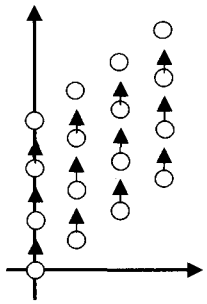
$\max(1,i-N) \le J \le \min(N,i-1)$

J의 최저 값과 최고 값을 계산하였다. J를 제거하면,

$$\begin{aligned} 1 &\le N \\ 1 &\le i-1 \\ i-N &\le N \\ i-N &\le i-1 \end{aligned}$$

정리하여, $2 \le I \le 2N$으로 I의 최저 값과 최고 값을 계산하였다. [그림 3]은 unimodular 변환으로 변환된 루프 한계값과 반복 공간을 보여준다.

```
for i = 2, 2N
   for j = max(1,i-N), min(n,i-1)
      ...
   endfor
endfor
```



[그림 3] Unimodular 변환으로 변환된 프로그램과 반복 공간
[Fig. 3] Unimodular transformed program and iteration space

2.2 Cycle Shrinking 방법

중첩 루프의 경우에 적용할 수 있는 cycle shrinking에는 세 가지 방법이 있다.

① Simple shrinking
② Selective shrinking
③ True Dependence shrinking

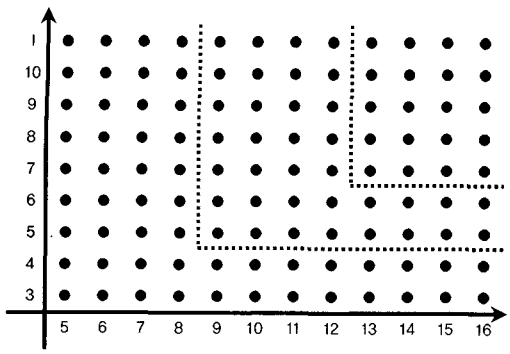
Simple shrinking방법은 내포된 각 루프에 대해서 독립적으로 종속 그래프를 고려하여 각 루프에 대하여 종속거리가 최소인 값을 reduction factor로 구한다. Selective shrinking 방법은 simple shrinking의 경우와 같이 중첩된 각 루프에 대해서 독립적으로 종속 그래프를 고려하여 각 루프에 대하여 각각의 reduction factor를 구한다. 그리고 바깥쪽 루프로부터 안쪽 방향으로 처음으로 1보다 큰 reduction factor를 가진 루프를 찾아서 그 루프를 simple shrinking 방법을 적용하고, 그 안쪽에 있는 모든 루프를 병렬화한다. True Dependence shrinking 방법은 중첩된 루프 구조에 존재하는 데이터 종속관계의 종속거리를 iteration space를 1차원적으로 생각했을때의 거리라고 간주하여, 병렬 수행할 수 있는 iteration의 수를 최대화 하는 방법이다[5].

2.3 Chen&Wang 방법

종속성 행렬(dependence matrix)을 이용하여 초기에 독립적인 계산 셑을 찾는다[4]. [그림 1]의 예의 유효 분할 영역(R'_0)인 $[2,\infty]\cup[\infty,4]$을 이용하여 병렬화 코드를 표현한다. [그림 4]는 불변 거리 예제[그림 1]를 이용하여 변환된 프로그램(a)과 반복 공간(b)을 보여준다.

```
DO 10 K = 0, max(⌈N1/2⌉, ⌈N2/4⌉)-1
  DOALL 20 FOR (I, J) ∈ {(I, J) | K*2+3 ≤ I
       ≤ min((K+1)*2+3-1,N1),  K*4+5 ≤ J ≤ N2}
       A(I, J) = B(I-3, J-5)
       B(I, J) = A(I-2, J-4)
  DOALL 30 FOR (I, J) ∈ {(I, J) | (K+1)*2+3 ≤ I
    ≤ N1, K*4+5 ≤ J ≤ min((K+1)*4+5-1,N2)}
       A(I, J) = B(I-3, J-5)
       B(I, J) = A(I-2, J-4)
```

(a) 변환된 프로그램



(b) 변환된 코드의 반복 공간

[그림 4] Chen&Wang 방법
[Fig. 4] Chen and Wang method

3. 선형 변환 프로그램 재구조화

이 장에서는 병렬성 추출을 위한 프로그램 재구조화 기법을 제시한다.

일반적으로 불변 종속거리이거나 가변 종속거리를 갖는 루프의 일반문장은 다음과 같다.

$S_1 : a_1(a_1I+b_1,\ e_1J+f_1) = \cdots$

$S_2 : a_2(a_2I+b_2,\ e_2J+f_2) = a_1(c_1I+d_1,\ g_1J+h_1) + \cdots$

$\vdots \qquad\qquad \vdots$

$S_{i-1} : a_{i-1}(a_{i-1}I+b_{i-1},\ e_{i-1}J+f_{i-1})$
$= a_{i-2}(c_{i-2}I+d_{i-2},\ g_{i-2}J+h_{i-2}) + \cdots$

$S_i : a_i(a_iI+b_i,\ e_iJ+f_i)$
$= a_{i-1}(c_{i-1}I+d_{i-1},\ g_{i-1}J+h_{i-1}) + \cdots$

$S_{i+1} : a_{i+1}(a_{i+1}I+b_{i+1},\ e_{i+1}J+f_{i+1})$
$= a_i(c_iI+d_i,\ g_iJ+h_i) + \cdots$

$\vdots \qquad\qquad \vdots$

$S_n : a_n(a_nI+b_n,\ e_nJ+f_n)$
$= a_{n-1}(c_{n-1}I+d_{n-1},\ g_{n-1}J+h_{n-1}) + \cdots$

단, $a_i(1\le i\le n)$는 다른 변수일수도 있다.

임의의 $i(1\le i\le n)$에 대하여 문장 S_i와 S_{i+1} 사이에 자료종속이 존재한다면 먼저 문장 S_i와 문장 S_{i+1}를 수행시킨다. 그러면 문장 S_{i+1}의 a_{i+1} 값은 a_i의 값 때문에 올바르게 수행되지 않는다. 그러나 여기서 문장 S_{i+1} 만을 다시 한번 수행시키면 문장 S_{i+1}의 a_{i+1}값도 올바르게 된다. 그러나 문장 S_1이 S_i에 자료 종속성을 갖고 문장 S_i가 문장 S_{i+1}에 자료 종속성을 갖는다면 이 경우에는 위와 같이 수행해도 문장 S_{i+1}의 a_{i+1}값은 아직도 올바르게 되지 않는다. 이 경우에도 한번 더 문장 S_{i+1}을 수행하면 문장 S_{i+1}의 a_{i+1}값도 올바르게 된다. 결국 자료 종속성을 완전히 제거하기 위해서는 임의의 두 문장 사이에 추이관계가 존재할 경우 몇 번의 Path를 거쳐서 추이관계가 형성되고 있는지 즉, 통로의 길이를 알아야 하며 Path의 길이 즉, 통로의 길이가 K라고 하면 앞의 수행관계를 K번 반복적으로 수행하므로서 자료종속관계는 사라지게 될 것이다. 이와 같은 방법이 〈프로그램 재구조화 1〉, 〈프로그램 재구조화 2〉, 〈프로그램 재구조화 3〉이며 어떠한 자료종속이 존재하는 순차 루프에 대해서 〈프로그램 재구조화 3〉을 수행하면 그 결과는 가장 많은 병렬처리가 가능한 병렬코드로 변환된다.

〈프로그램 재구조화 1〉

```
FOR 0이 아닌 모든 aij, cij에 대해서
IF 통로의 길이 = 1 THEN
   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 K=Wij, N, Wij
                  Si
         ENDDOALL
       ENDDOALL
 ELSE IF 통로의 길이 = 2 THEN
   IF  UK1와 WK1 이 1 THEN
       DOALL K=bK1, M
```

```
        DOALL L=dK1, N
              Sj
        ENDDOALL
      ENDDOALL
    ELSE
      DOALL K=U'K1, M, U'K1
        DOALL K=W'K1, N, W'K1
              Sj
        ENDDOALL
      ENDDOALL
ELSE IF 통로의 길이 = 3 THEN
    IF  Uuv 와 Wuv 가 1 THEN
      DOALL K=bK1 + buv - 1, M
        DOALL L=dK1 + duv - 1, N
              Sv
        ENDDOALL
      ENDDOALL
    ELSE
      DOALL K=U'UV, M, U'UV
        DOALL K=W'uv, N, W'uv
              SV
        ENDDOALL
      ENDDOALL
```

〈프로그램 재구조화 2〉

DM의 변형 및 새로운 자료 종속을 가지는 변수 계산

1. LCM은 최소공배수이다.
2. IF 통로의 길이가 2이고

$$DM=\begin{bmatrix}((a_{ij}\oplus u_{ij}, b_j \oplus v_{ij}),\ (c_j \oplus w_{ij}, d_j \oplus x_{ij}))\\ ((a_{kl}\oplus u_{kl}, b_{kl} \oplus v_{kl}),(c_{kl} \oplus w_{kl}, d_{kl} \oplus x_{kl}))\end{bmatrix}$$

THEN

$U''_{K1} = LCM(bij \oplus vij, akl \oplus u_{kl})$

$W'_{K1} = LCM(d_{ij} \oplus x_{ij}, c_{kl} \oplus w_{kl})$

3. IF통로의 길이가 3 이상이고

$$DM=\begin{bmatrix}((a_{ij}\oplus u_{ij}, b_j \oplus v_{ij}),\ (c_j \oplus w_{ij}, d_j \oplus x_{ij}))\\ ((a_{kl}\oplus u_{kl}, b_{kl} \oplus v_{kl}),\ (c_{kl} \oplus w_{kl}, d_{kl} \oplus x_{kl}))\\ ((a_{uv}\oplus U_{uv}, b_{uv}\oplus V_{uv}),\ (C_{uv}\oplus W_{uv}, d_{uv} \oplus X_{uv}))\end{bmatrix}$$

THEN

$U'_{UV} = LCM(b_{KI} \oplus v_{KI}\ ,\ auv \oplus uuv)$

$W'_{UV} = LCM(d_{KI} \oplus X_{KI}\ ,\ C_{uv} \oplus W_{uv})$

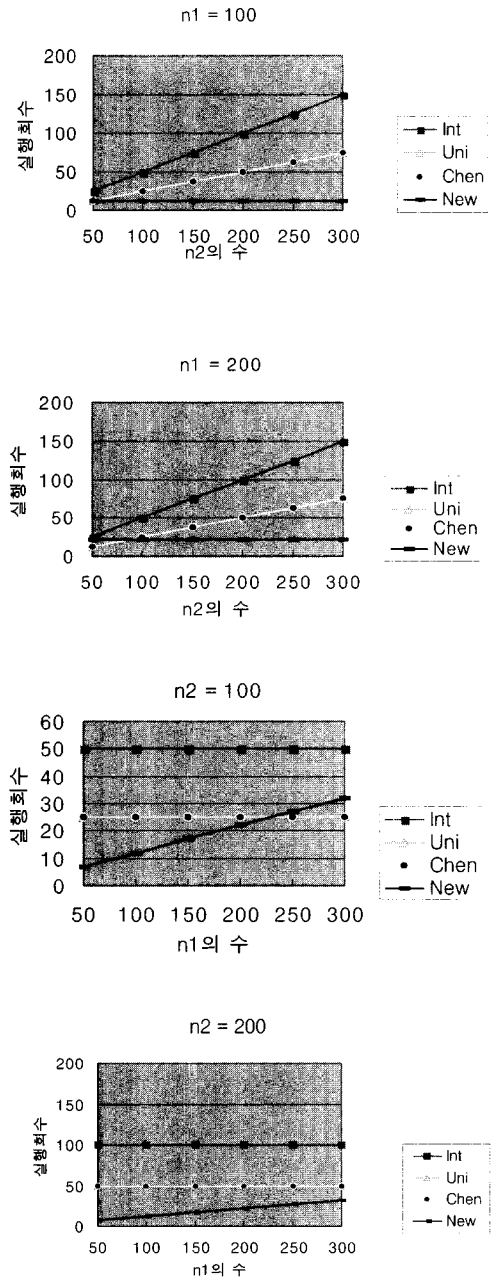
〈프로그램 재구조화 3〉

1. 통로의 길이가 1인 DM를 만든다.
2. IF DM에 첨자값이 같은것이 있다면 THEN 다른 이름으로 대체한다.
3. For all i, j
   IF($a_{ij} \oplus u_{ij}$) 〉 M 혹은 ($c_{ij} \oplus w_{ij}$) 〉 N THEN 그 원소는 0으로 치환한다.
4. 모든 0이 아닌 원소에 대해 〈프로그램 재구조화 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 1을 증가 시킨다.
7. IF 똑같은 원소가 존재한다면 THEN 하나만 남겨 놓고 모두 제거한다.
8. DM에서 〈프로그램 재구조화 2〉를 적용하여 아직도 남아있는 자료 종속을 찾고 DM를 변형한다.
9. IF LCM을 취한 변수의 값이 DO루프의 최종치보다 크면 THEN 그 원소는 0으로 치환한다.
10. 모든 0이 아닌 원소에 대해 〈프로그램 재구조화 1〉을 적용하라.
11. Goto 6
12. IF 첨자값이 바뀐 원소가 존재 THEN 〈프로그램 재구조화 1〉을 적용하라.
13. 변형된 문장을 제외한 모든 문장에 대해 doall Si를 수행한다.

4. 성능 평가

불변거리(uniform)에 대한 예[그림 1]로 선형 변환인 interchanging[2], 혼합된 unimodular [4], Chen&Wang[3]방법과 본 논문의 프로그램 재구조화를 비교하고자 한다. 우선, 병렬 코드의 실행 수를 계산하면 interchanging의 병렬코드는 λ =2n1-4번 만에 수행 할 수 있고, 통합된 unimodular와 Chen& Wang은 $\lambda = \lceil (n2-4)/4 \rceil$ 번 수행한다. 하지만 본 논문에서 제시된 방법을 적용하면 λ =2+ $\lceil$ n1/10 $\rceil$ 번 만에 실행 할 수 있어 가장 우수하다는 것을 알 수 있었다. 실행 수에 대한 성능평가 그래프를 [그림 5]에서 보여준다. N1(n2)의 값을 50에서 300까지 50씩 증가하면서 이에 따른 실행 수를 비교하였다. N1의 값을 각각 50, 100, 200으로 고정하고 n2의 값을 증가 했을 때 본 논문의 방법은 n2의 값에 따라 변하지 않았다. 그러나 interchanging, 통합된 unimodular와 Chen&Wang은 선형적으로 증가됨을 알 수 있었다.

그리고 n2의 값을 각각 50, 100, 200으로 고정하고 n1의 값을 증가 했을때 interchanging, 통합된unimodular와 Chen& Wang은 값에 따라 변하지 않았고, 본 논문의 방법은 선형적으로 증가됨을 알 수 있었다.



[그림 5] 실행 수에 따른 비교
[Fig. 5] The number of execution for n1, n2

5. 결론

본 논문에서는 병렬 처리 시스템에 적용할 수 있는 새로운 프로그램 재구조화 기법을 제시하였다. 기존의 방법과 본 논문의 제안된 기법을 비교였을 때 본 프로그램 재구조화 기법이 매우 효과적임을 성능평가에서 알 수 있었다. 앞으로 이 프로그램 재구조화 기법을 Unimodular 변환과 같은 선형 변환과 혼합하여 더 많은 병렬화 효과를 도출하여야 할 것이다.

※ 참고문헌

[1] U. Banerjee, "*Dependence Analysis For Supercomputing*", Kluwer Academic Publishers, 1988

[2] U. Banerjee, "*Unimodular transformations of double loops*", In 3rd Workshop on Languages and Compilers for parallel computing, August 1990.

[3] Yeong-Sheng Chen and Sheng-De Wang, "*A Parallelizing Compilation Approach to Maximizing Parallelism within Uniform Dependence Nested Loops*", Dep. of Electrical Engineering, National Taiwan University, 1993.

[4] W. Li and K. Pingali, "*A singular loop transformation framework based on non-singular matrices*", TR92-1294, Dept. of Computer Science, Cornell University, July 1992.

[5] Constantine D. Polychronopoulos, "*Parallel Programming and Compilers*", Kluwer Academic Publishers, 1988.

[6] M. E. Wolf and Monica S. Lam, "*A loop transformation theory and an algorithm to maximize parallelism*", IEEE Transactions on Parallel and Distributed Systems, July 1991.

[7] M. E. Wolf, "*Improving Locality and Parallelism in Nested Loops*", PhD thesis Dept. of Computer Science, University of Stanford, August 1992.

[8] M. J. Wolf, "*High Performance compilers for Parallel Computing*", Oregon Graduate Institute of Science & Technology, 1996.

[9] 이만호, "*병렬화를 위한 루프구조의 변환*", 정보과학회지 제12권 제5호, 1994. 6.

[10] 송월봉외, "*Extracting Parallelism in Nested oops*", COMPSAC, IEEE, 서울, 1996. 8.

송 월 봉

1974년 숭실대 공학사
1982년 한양대 공학석사
1998년 순천향대학교
공학박사
1978년 ~ 현재
시립인천전문대학
컴퓨터정보과 교수
관심분야 : 병렬처리,
컴파일러, 알고리즘