論文
6-2-17

Unimodular 및 Non-unimodular 변환의 성능평가 (Performance Evaluation of Unimodular and Non-unimodular Transformation)

송 월 봉(Worl-Bong Song)[1)]

요 약

일반적으로 응용프로그램에서 병렬성 추출에 대한 핵심 부분은 루프이고 루프내의 첨자변수들 사이에는 자료종속성이 존재한다. 특히 문장들 사이에 가변 및 불변종속거리를 갖는 종속관계는 매우 복잡하다. 따라서 이 경우 컴파일 시 병렬성 추출은 매우 어렵다. 따라서 본 논문에서는 기존에 제시되어있는 병렬성 추출 방법들 중에서 unimodular방법과 non-unimodular방법에 대하여 분석하고 이들의 장단점을 파악하여 향후 효율적인 종속성 제거방법을 제안하는데 도움이 되고자한다.

Abstract

Generally, In a application program the core part for parallel processing is a loop. therefore exist data dependencies between the array index variables of a loop. The data dependence relations between statements which from variable or constant dependence distance are specially complex. Therefore extracting parallelism for those statements at compile time is very difficult. in this paper, among the proposed methods of extracting parallelism, analysis the unimodular method and non-unimodular method and grasping the merits and demerits of them. hereafter, this method will go far toward solving the effectively extracting parallelism of the loop.

논문접수 : 2005. 5. 14.
심사완료 : 2005. 6. 10.

1) 정회원 : 시립인천전문대학 컴퓨터정보과 교수
본 논문은 2004학년도 교내학술연구지원비에 의한 것임.

1. 서 론

병렬화 컴파일러 중에서 가장 기본적이고, 가장 많이 활용할 수 있는 부분은 순차프로그램에 있는 루프로부터 병렬성을 추출하여 병렬 코드로 변환해주는 루프 재구조화 방법이다. 이 방법은 병렬처리 시스템의 속도를 향상시키는데 상당한 효과를 가지고 있다. 루프 재구조화 방법중 Loop spreading은 공유기억장치 멀티프로세서에서 중첩된 루프의 반복들을 분산시키기 위한 방법이다.
이러한 방법으로는 loop distribution, cycle shrinking, doacross, minimum distance, loop partition, unimodular transformation 그리고 non-unimodular transformation 등이 있다.

본 연구는 루프 재구조화 방법 중 병렬컴퓨터의 성능을 향상시키기 위하여 반드시 필요한 병렬성 추출방법 중 unimodular 행렬(행렬식 ±1)을 사용하여 체계적인 방법으로 중첩 루프를 재구성하는Unimodular 변환 방법과, 통신을 최소화하거나 계층 기억장치를 효율적으로 사용하기 위한 non-unimodular 변환 방법에 대하여 각각의 알고리즘을 비교 분석 후 성능 평가하고 결과를 이용하여 향후 효율적인 종속성 제거 방법을 제안하고자 한다.

2. Unimodular, Non-unimodular 변환

Unimodular 변환 방법은 unimodular 행렬(행렬식±1)을 사용하여 체계적인 방법으로 중첩 루프를 재구성하는 방법이다[1,2,3,4,5,6,7]. 이러한 방법들은 모든 중첩 루프의 정확한 한계 값을 계산하게 한다. 최근에, 통신을 최소화하거나 계층 기억장치를 효율적으로 사용하기 위하여 non-unimodular 변환을 사용한다[3]. Non-unimodular 변환(행렬식>1)은 변환된 반복 공간에 "holes"을 생성한다. Non-unimodular 변환에서 정확한 한계 값을 계산하기 위하여 hole을 건너뛰기 위하여 루프의 증가 값을 변환하여야 한다. 증가 값을 계산하기 위하여 Hermite Normal Form을 사용한다[8]. 이 matrix의 주 대각 원소의 값이 각 반복 공간의 증가값이다.
Unimodular 행렬로 표현할 수 있는 기본적인 변환은 loop interchanging, loop skewing, loop reversal이고, 병렬화를 최대화하기 위하여 위 세 가지를 통합하는 방법을 사용한다.

2.1 Unimodular, Non-unimodular 변환과 *Hermite Normal form*

Unimodular 변환은 unimodular 행렬에 의해 표현될 수 있는 변환이다. Unimodular 행렬의 장점은 "phase-ordering problem". 즉, 루프 변환의 순서를 쉽게 결정할 수 있다는 것이다. Unimodular 행렬은 3가지 특성을 갖는다. 첫째, n x n의 정방 행렬이다. 둘째, 행렬의 모든 원소가 정수이다. 셋째, 행렬식이 ±1이다. 이러한 특성 때문에 2개의 unimodular 행렬의 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 reversal이다.

2.1.1 루프 Interchanging
2개의 열(행)을 교환한다.

$$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \xrightarrow{\text{interchange}} \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

k와 k+1 루프 교환은 일치하는 distance (direction) vector를 변환한다.

$$\text{int}(k, k+1)$$
$$(d1,d2,\cdots,dk,dk+1,\cdots,dn) \rightarrow (d1,d2,\cdots,dk+1,dk,\cdots,dn)$$

예를 들면, 이중 루프에서는 종속 관계 (=, <)을 (<, =)로 변환하고, 삼중 루프는 IJK 루

프를 IJK → IKJ → KIJ → KJI나 IJK → JIK → JKI → KJI로 변환한다.

2.1.2 루프 Skewing

주대각 원소의 위(upper skewing 행렬)나 아래(lower skewing 행렬)에 0 이외의 정수값(f)으로 대체한다. f는 skewing factor이다.

$$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \xrightarrow{\text{skewing}} \begin{bmatrix}1 & f\\ 0 & 1\end{bmatrix}, \quad \begin{bmatrix}1 & 0\\ f & 1\end{bmatrix}$$

Wavefront방법의 일종으로, 루프 반복의 실행순서에 영향을 주지 않고 반복 공간의 모양만 변화한다.

skew(k,f,m)

(d1,d2,⋯,dk,⋯,dm⋯,dn) → (d1,d2,⋯,dk,⋯,dm+fdk⋯,dn)

2.1.3 루프 Reversal

주 대각 원소에 -1을 곱한다.

$$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \xrightarrow{\text{reversal}} \begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}, \quad \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$$

reverse

(d1,d2,⋯,dk,⋯,dn) → (d1,d2,⋯,-dK,⋯,dn)

Non-unimodular 변환(행렬식>1)은 반복 공간에 hole이 생기기 때문에 Fourier-Motzkin 알고리즘으로 계산된 한계 값은 정확하지 않다. 정확한 한계 값을 계산하기 위하여 1보다 큰 증가 값을 사용하여 hole을 건너뛰어야 한다. 증가 값을 계산하기 위하여 Hermite Normal Form을 사용하여야 한다[8].

Hermite Normal Form은 다음과 같은 특성을 가진다[8].

i) lower triangular; hij = 0 for i < j.

ii) hii > 0 for I = 1, ⋯, n.

iii) hij ≤ 0 and | hij | < hij for i > j

그리고, gcd(hii,hij) = g = hiix + hijy이고,

$H = \begin{bmatrix}h11 & h12\\ h21 & h22\end{bmatrix}$은 $\begin{bmatrix}x & -hij/g\\ y & hii/g\end{bmatrix}$로 구성된다.

2.2 한계 값 계산 알고리즘

일반적으로, x의 ceiling을 ⌈x⌉로 표기하고, x보다 작지 않은 가장 작은 정수 값을 나타낸다. x의 floor을 ⌊x⌋로 표기하고, x보다 크지 않은 가장 큰 정수 값을 나타낸다.

중첩 루프는 다음과 같은 형태로 표현한다.

for I1 = max(⌈L1,1⌉, ⌈L1,2⌉,⋯) to min(⌊U1,1⌋, ⌊U1,2⌋,⋯) by step1

⋮

for In = max(⌈Ln,1⌉, ⌈Ln,2⌉,⋯) to min(⌊Un,1⌋, ⌊Un,2⌋,⋯) by stepn

최저 값은 Lk,i이고, 최고 값은 Uk,j이다. 만약 증가 값이 1 보다 크면, 정규화해야 한다. 중첩 루프 내부의 문장 변환은 간단하기 때문에 생략한다[9].

인덱스 벡터 (i1,⋯,in)은 중첩 루프의 각 반복과 동일하다. 중첩 루프의 한계에 의해 정의된 반복들은 Zn의 convex subset이고, 다음과 같이 정의된다.

max(L1,1,L1,2,⋯) ≤ I1 ≤ min(U1,1,U1,2,⋯)

⋮

max(Ln,1,Ln,2,⋯) ≤ In ≤ min(Un,1,Un,2,⋯)

위와 같이 나타내는 것을 삼각 시스템(triangular system)이라고 하고, 중첩 루프을 재구성하는 알고리즘은 정규화 되어 있다. 루프 변환 T는 반복(i1,⋯,in)을 (i'1,⋯,i'n)와 연결된다.

$$\begin{bmatrix}I'_1\\ \vdots\\ I'_n\end{bmatrix} = T \begin{bmatrix}I_1\\ \vdots\\ I_n\end{bmatrix}$$

$$\begin{bmatrix} I_1 \\ \vdots \\ I_n \end{bmatrix} = T^{-1} \begin{bmatrix} I'_1 \\ \vdots \\ I'_n \end{bmatrix}$$

루프 한계값을 계산하는 알고리즘은 2 단계로 구성한다.

i) Fourier-Motzkin elimination방법을 사용하여 루프 한계 값 Lk,l, Uk,l을 계산한다.

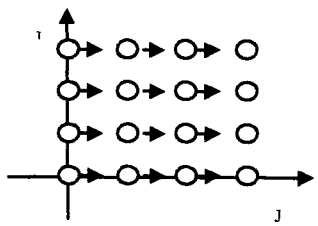
ii) Hermite normal form으로 증가 값(stepk)을 계산한다.

반복공간의 모든 정수는 lattice이다. 만약 루프 변환 T가 unimodular이면, L(T) = Zn이고, 루프 변환 T가 non-unimodular이면, L(T) ≠ Zn이다. 즉, non-unimodular는 반복 공간의 일부가 정수가 아니고 non-lattice는 hole을 가진다. T의 역도 정수가 아니다.

3. unimodular, non-unimodular의 성능비교

3.1 Unimodular 변환

병렬화를 높이기 위하여 중첩 루프를 unimodular 변환으로 재구성하는 것을 서술한다. 우선, Fourier-Motzkin 방법을 사용하여 루프 한계 값 Lk,l, Uk,l을 계산한다. [그림 1]은 이중 루프의 예제 프로그램과 반복 공간을 보여준다.




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

[그림 1] 이중 루프 프로그램과 반복 공간

[Fig. 1] Nested 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 $T = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ 에 따라

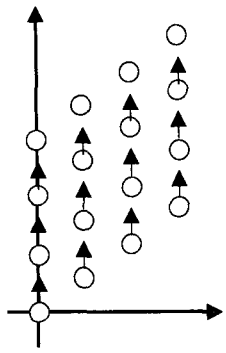
$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 0 & 1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \le \begin{bmatrix} -1 \\ N \\ -1 \\ N \end{bmatrix}$$

삼각 시스템으로 재구성하기 위하여, Fourier-Motzkin 방법을 사용하면,

max(1,i-N) ≤ J ≤ min(N,i-1)

J의 최저 값과 최고 값을 계산하였다. J를 제거하면,

```
  1 ≤ N
  1 ≤ i-1
i-N ≤ N
i-N ≤ i-1
```



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

I'_1

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

I'_2

[그림 2] Unimodular 변환으로 변환된 프로그램과 반복 공간

[Fig. 2] A transformed Program into unimodular transformation and iteration space

3.2 Non-unimodular 변환

루프 변환이 non-unimodular이면, 첫째, Fourier-Motzkin방법을 사용하여 루프 한계 값을 계산한다. Non-unimodular 변환

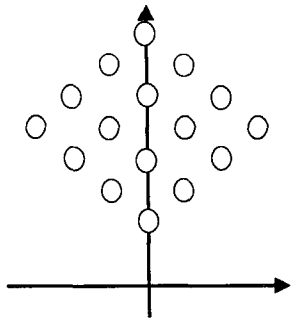
$T = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$에 따라, $2 \le I \le 2N$와 $\max(2-i, i-2N) \le J \le \min(2N-i, i-2)$으로 I와 J의 최저 값과 최고 값을 계산한다.

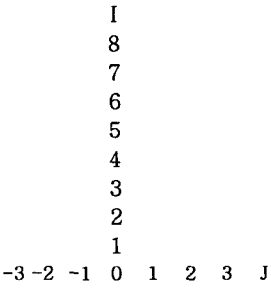
둘째, hole을 건너뛰기 위하여 루프 증가 값을 계산해야한다. 루프 증가 값은 *Hermite Normal Form*을 사용하고, 이 행렬의 주대각 원소 값이 각 반복의 증가 값이다.

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1 & 2 \end{pmatrix}$$

따라서, step1 = 1, step2 = 2이 계산되었다. [그림 3]은 non-unimodular 변환으로 변환된 루프 한계 값과 반복 공간을 보여준다.

```
for i = 2, 2N step 1
  for j = max(2-i,i-2N), min(2N-i,i-2) step
2
```





[그림 3] Non-unimodular 변환으로 변환된 프로그램과 반복공간

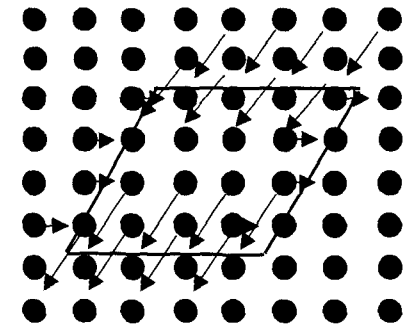
[Fig. 3] A transformed Program into non-unimodular transformation and iteration space

3.3 성능 비교

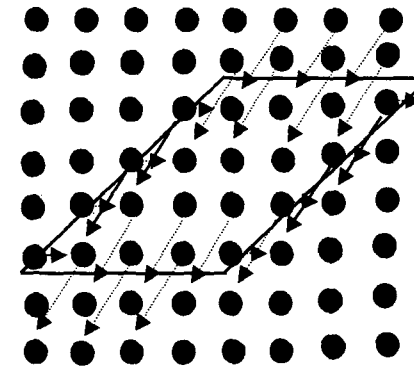
반복 공간 *tiling*은 n-차원 반복 공간을 각 블록이 독립적으로 실행하는 n-차원 블록으로 분리된다. 각 블록 사이에는 종속성이 없어야 한다. Tiling은 분산 메모리 기계(distributed memory machine)와 계층적 공유 메모리(hierarchical shared memory)에서 매우 유용하다. 여기에서는, non-unimodular 변환의 필요성을 tiling을 이용하여 통신을 감소하는 예를 보인다.

거리 행렬 $D=\begin{bmatrix} 0 & 2 & 1 \\ 1 & -1 & 0 \end{bmatrix}$일 때, 최소의 통신을 가지는 tiling 변환 T는 $T=\begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}$ 이다. 정칙 행렬 T을 찾아야 하므로, $T=\begin{bmatrix} 1 & t_1 \\ t_2 & 1 \end{bmatrix}$은 D+=TD= $\begin{bmatrix} t_1 & 2-t_1 & 1 \\ 1 & 2t_2-1 & t_2 \end{bmatrix}$이다. D+가 최소가 될 set은 $t_1=0,\ t_2=\frac{1}{2}$와 $t_1=2,\ t_2=\frac{1}{2}$이다. 행렬식이 0이 아니어야 한다(1-t2t1≠0). 그래서, 두 번째 것은 불가능하고(1-2·½=0), 해는 t1 = 0, t2 = ½이다.

[그림 4]는 *non-unimodular* 행렬 $\begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}$와 unimodular 행렬 $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$의 tile 공간을 보여준다.



(a) Non-unimodular transformations



(b) Unimodular transformations

[그림 4] 두 변환의 통신비교
[Fig.4] Communication of unimodular and non-unimodular transformations

[그림 4]에서 unimodular, non-unimodular tiling을 비교해 보면, [그림 4] (b)의 두꺼운 화살표는 non-unimodular에는 없는 통신이 unimodular에 추가된 것을 표현한다. 즉, unimodular tiling에 의한 통신의 양이 non-unimodular tiling 보다 더 많은 것을 볼 수 있다.

4. 결론

본 연구에서는 루프 재구조화 방법 중 병렬 컴퓨터의 성능을 향상시키기 위하여 반드시 필요한 병렬성 추출방법 중 unimodular 행렬(행렬식±1)을 사용하여 체계적인 방법으로 중첩 루프를 재구성하는Unimodular 변환 방법과, 통신을 최소화하거나 계층 기억장치를 효율적으로 사용하기 위한 non-unimodular 변환 방법에 대하여 각각의 알고리즘을 비교 분석 후 성능 평가하였다. 평가결과 반복 공간 Tiling을 이용한 두 변환의 성능 비교에서 unimodular tiling에 의한 통신의 양이 non-unimodular tiling 보다 더 많은 것을 확인하였다.

앞으로 이 결과는 perfect benchmark와 같은 benchmark 프로그램에 적용해서 실질적인 문제에 도입하는 부분과, 궁극적으로는 MIMD 기계에 적용하기위한 새로운 병렬처리 컴파일러를 구현하는 과제가 남아있다고 볼 수 있다.

참고문헌

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

[2] Erik H. D'Hollander, "Partitioning and Labeling of Loops by Unimodular Transformations", IEEE Trans. on Parallel and Distributed systems Vol.3, No. 4, July, 1992.

[3] Fernandez, A., J.M. Llaberia and M. Valero-Garcia, "Loop Transformation Using Nonunimodular Matrices," IEEE Trans. on Parallel and Distributed Systems, vol. 6, No. 7, Aug. pp832-840, 1995

[4] Ju., J. and V. Chaudhary, "Unique sets oriented partitioning of nested loops with non-uniform dependences," in Proc., Int. Conf. Parallel Processing, Vol III, pp45-52, 1996

[5] 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.

[6] 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.

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

[8] Polychronopoulos, c. d., "More on advanced loop optimization ", CSRD Report No. 667, Univ. of Illinois at Urbana-Champaign, Oct. 1987.

[9] Shen, Z., Z. li, P. C. Yew, "An Empirical Study on Array Subscripts and Data Dependencies", Proc. of International Conference Parallel Processing, pp. 145-152, aug. 1989.

송월봉

1974년 숭실대 공학사
1982년 한양대 공학석사
1998년 순천향대 공학박사
1978년 ~ 현재 시립인천전문대학
컴퓨터정보과 교수
관심분야 : 병렬처리컴파일러, 알고리즘, 프로그래밍언어