비균일 단일루프에서의 효율적인 루프 분할 방법

An Efficient Loop Splitting Method on Single Loop with Non-uniform Dependences

정삼진
천안대학교 정보통신학부
Sam-Jin Jeong(sjjeong@cheonan.ac.kr)

요약

본 논문은 비균일 단일루프의 병렬성을 향상시키기 위해서 지금까지 개발된 최소 종속 거리 분할 방법, Polychronopoulous 분할 방법, 그리고 최초 종속 분할 방법과 같은 세 가지 루프 분할 방법을 소개하며, 기존의 분할 방법의 문제점들을 제시한다. 그리고, 기존의 세 가지 루프 분할 방법 중에서 가장 효과적인 최초 종속 분할 방법을 확장하여 병렬성을 향상시킨 보다 강력한 루프 분할 방법을 제안한다. 제안된 알고리즘은 역 종속성일 경우와 gcd(최대공약수)값이 1보다 클 경우와 같이 최초 종속 분할 방법에서 해결하지 못한 문제점들을 해결하였다.

■ 중심어 : | 병렬 컴파일러 | 루프 분할 | 루프 전환 | 비균일 루프 |

Abstract

This paper introduces three loop splitting methods such as minimum dependence distance method, Polychronopoulous' method, and first dependence method for exploiting parallelism from single loop which already developed. And it also indicates their several problems. We extend the first dependence method which is the most effective one among three loop splitting methods, and propose more powerful loop splitting method to enhance parallelism on single loop. The proposed algorithm solves several problems, such as anti-flow dependence and g=gcd(a,c) > 1, that the first dependence method has.

■ keyword : | Parallelizing Compiler | Loop Splitting | Loop Transformation | Non-uniform Dependence |

I. 서 론

1945년 폰 노이만이 프로그램 내장형 컴퓨터 구조를 발표한 이래로 급속한 발달을 거듭하여 초고속으로 수행하는 수퍼 컴퓨터가 탄생되어 여러 분야에 사용되고 있으나, 많은 과학 계산을 필요로 하는 응용분야에서는 더욱 빠른 컴퓨터가 요구되고 있다. 그래서 동일한 프로세서를 여러개 사용하여, 한 프로그램을 여러개의 세부 단위로 나누어서 이들을 병렬로 처리하도록 하는 병렬 컴퓨터가 등장하게 되었다. 이로서 병렬 컴퓨터를 이용한 병렬 처리의 구현을 위해서 병렬 프로그래밍 언어에 대한 연구가 활발하게 진행되고 있다. 병렬 프로그램에 관한 연구로서는 순차 프로그램을 병렬 프로그램으로

접수번호 : #050410-001
접수일자 : 2005년 04월 10일
심사완료일 : 2005년 05월 19일
교신저자 : 정삼진, e-mail : sjjeong@cheonan.ac.kr

자동적으로 변환하는 것이다. 이러한 방법은 병렬 컴퓨터가 개발되기 이전에 오랜 동안 사용되던 순차 컴퓨터를 위해 개발된 막대한 양의 프로그램들을 병렬 컴퓨터에서도 이용할 수 있도록 하기위한 것이었다. 프로그램 실행시 수행시간은 프로그램 길이에 비례하지 않는다.[3] 대부분의 수행시간이 지극히 적은 부분, 특히 루프(loop) 구조에 의해 소비되고 있다. 이러한 이유 때문에 순차 프로그램을 병렬 프로그램으로 변환하는 연구의 대부분이 루프 구조의 변환과 루프 내에 존재하는 데이터 종속성을 분석하는 연구에 치중하고 있다. 이런 데이터 종속성 분석은 프로그램의 작업 수행 순서에 따라 하며, 하드웨어의 특성과는 전혀 무관하다. 데이터 종속성 분석 결과에 따라 병렬화를 위한 변환을 하게 되는데, 변환된 병렬 프로그램이 효과적으로 수행되기 위해서는 수행될 병렬 컴퓨터의 하드웨어적 특성을 잘 활용하여야 한다. 본 논문에서는 단일루프에서 종속거리가 불규칙한 경우에 가장 효율적으로 병렬성을 증가시키기 위한 루프 분할 방법을 제안하고자 한다.

본 논문의 구성은 2장에서는 루프 구조, 기존의 연구된 최소 종속거리를 이용한 루프 분할 방법, Polychronopoulous의 루프 분할 방법, 그리고 최초 종속성을 이용한 루프 분할 방법을 각각 소개하였고, 그들의 장·단점을 설명하였다. 3장에서는 그중에서도 가장 효율적인 최초 종속성을 이용한 루프 분할 방법의 단점을 개선하여 확장된 최초 종속성을 이용한 루프 분할 방법을 제시하였고, 4장에서는 본 논문에서 제안한 방법과 최초 종속성을 이용한 루프 분할 방법을 비교 분석하여 성능을 평가하였으며, 마지막으로 결론을 내렸다.

II. 관련 연구

1. 루프 구조

루프 구조 변환에 관한 연구에서 주로 다루고 있는 일반적인 다중 루프 구조는 [그림 1]에서 보는 것과 같다. 본 구조는 n개의 루프와 변수들로 이루어져 있고, 각 변수는 각각 초기값(L_i, Lower Limit)과 최대값(U_i, Upper Limit) 범위, 증가치(S_i)가 주어진다. [그림 2]는 일반적인 단일 루프 구조를 나타내는데, 정규화되어 있지 않기 때문에 변수 I는 초기값(L)과 최대값(U) 범위, 증가치(S)가 있다. 이와 같이 정규화 되지 않는 루프 구조는 [그림 3]과 같이 초기값과 증가치가 1인 정규화된 루프 구조로 변형할 수 있다. 따라서 루프 구조와 관련된 모든 이론은 정규화 된 루프 구조를 대상으로 하여도 일반적인 루프 구조에 적용할 수 있게 된다.

본 논문에서 대상으로 삼고 있는 루프는 정규화 된 단일 루프로서 기본 모형은 [그림 4]와 같다.

```
DO I1 = L1, U1, S1
   . . .
   DO In = Ln, Un, Sn
      . . .
   END DO
   . . .
END DO
```

그림 1. 일반적인 다중 루프 구조

```
DO I = L, U, S
   . . . I . . .
END DO
```

그림 2. 일반적인 단일 루프 구조

```
DO K = 1, (U-1/S)+L, 1
   . . . L+(K-1)*S . . .
END DO
```

그림 3. 정규화 된 단일 루프 구조

```
        DO I = 1, N
S1 :       A(a*I+b) = . . .
S2 :                = A(c*I+d)
        END DO
```

그림 4. 정규화된 단일 루프의 일반적인 유형

2. 루프 분할 (Loop Splitting)

루프 분할(Loop Splitting)은 단일 루프에 적용할 수 있는 루프 변환 방법으로서, 하나의 순차 루프에서 병렬 수행이 가능한 부분을 분리해서 2개 이상의 부분 병렬 루프로 변환하여 루프의 병렬 실행 능력을 높여주는 루프변환 방법 중 하나이다.

[그림 4]에서 문장 S_1과 문장 S_2 사이에 데이터 종속

성이 존재한다는 것은 두 문장이 같은 메모리의 영역을 참조하기 때문에 병렬로 수행할 수 없다는 것을 뜻한다. 단일 루프 내에서 일차원 배열의 일반적인 형식은 [그림 4]와 같고, 첨자 식에서 루프변수 I의 계수인 a와 c의 관계에 따라 [그림 5]와 같이 구분한다[1].

```
1. a = c = 0,
2. a = 0, c ≠ 0 or a ≠ 0, c = 0,
3. a = c ≠ 0,
4. a ≠ 0, c ≠ 0, a ≠ c.
```

그림 5. 루프변수 I의 계수인 a와 c의 관계에 따른 분류

```
        DO I = 1, N
S1 :        A(3*I+1) = . . .
S2 :                 = A(2*I-4)
        END DO
```

(a) 루프의 예

I	A(3*I+1)	A(2*I-4)
1	A(4)	A(-2)
2	A(7)	A(0)
3	A(10)	A(2)
4	A(13)	A(4)
5	A(16)	A(6)
6	A(19)	A(8)
7	A(22)	A(10)
8	A(25)	A(12)
9	A(28)	A(14)
10	A(31)	A(16)

(b) (a)의 루프 펼침

그림 6. 종속성들의 종속거리가 모두 다른 경우의 루프

[그림 5]에서 루프변수 I의 계수인 a와 c의 관계에 따른 분류 중 1번에서 3번까지는 정형화된 형태이기 때문에 별로 문제가 되지 않는다. 본 연구는 마지막 네번째 경우인, a ≠ 0, c ≠ 0, 그리고 a ≠ c에 적용할 수 있는 루프 분할 방법에 대해서 살펴보고자 한다. 이 경우에는 루프 내에 존재하는 데이터 종속성들의 종속거리가 다르기 때문에 루프의 병렬화를 어렵게 하는 원인이 된다. [그림 6] (a)는 네 번째 경우에 대한 루프의 한 예이고, [그림 6] (b)는 [그림 6] (a)의 루프를 루프 펼침을 이용하여 표현한 것이다.

2.1 최소 종속거리를 이용한 루프 분할

앞의 예에서 루프 내에 존재하는 종속성들의 종속거리는 각각 3, 4, 5이었다. 최소 종속거리를 이용한 루프 분할은 이들 중에 가장 작은 종속거리인 3을 기준으로 하여 순차루프를 여러 개의 병렬 수행 가능한 부분 병렬루프로 변환하게 된다. 이때 변환된 각 병렬 루프의 크기는 최소 종속거리의 값인 3과 같다. [그림 7]은 최소 종속거리를 이용하여 [그림 6] (a)의 루프를 병렬루프로 변환한 결과이다. 그러나 이 방법은 가장 작은 종속거리인 3을 기준으로 하여 병렬루프로 변환하므로 효율적인 분할 방법이라고 할 수 없다.

```
        DO I = 1, 10, 3
            DOALL I = J, J+2
S1 :        A(3*I+1) = ...
S2 :                 = A(2*I-4)
            END DOALL
        END DO
```

그림 7. 최소 종속거리를 이용한 루프 분할

2.2 Polychronopoulous의 루프 분할

Polychronopoulous는 최소 종속거리를 이용한 루프 분할 방법보다 좀더 루프의 병렬성을 증가시킬 수 있는 새로운 루프 분할 방법을 제안하였다.[4]

이 방법은 Increment factor ρ와 Difference D를 이용한 루프 분할 방법으로서, 각각에 대한 정의는 다음과 같다.

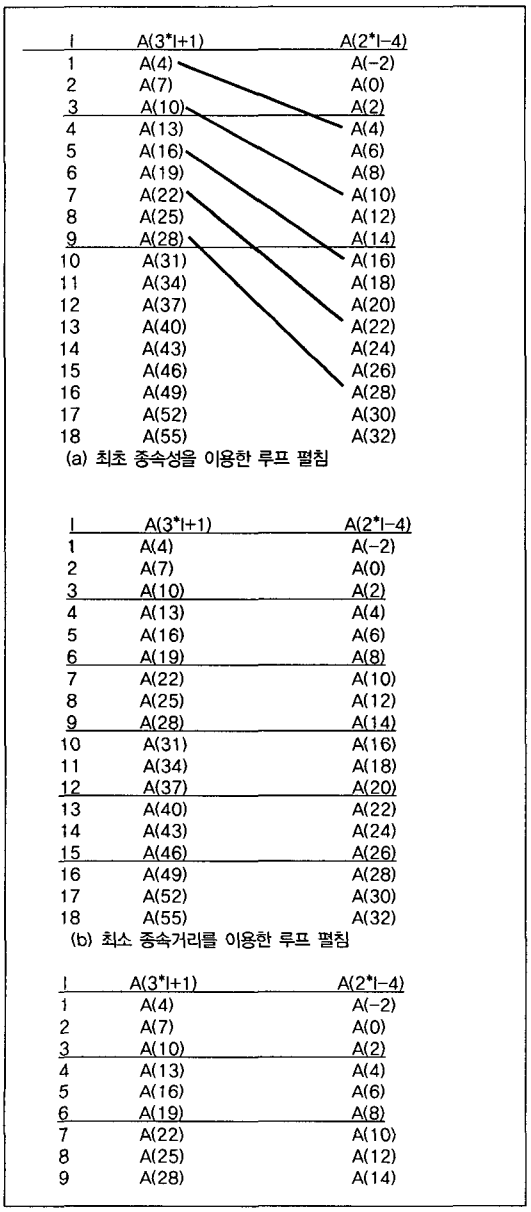
1) 식 a*I+b = c*I+d 가 주어졌을 때 오른편 식 I의 상수 c를 Increment factor ρ 라 정의한다.
2) 식 a*I+b = c*I+d 가 주어졌을 때 Difference D = a*I+b-(c*I+d)라 정의한다.
3) 루프 내에 존재하는 종속성의 Instance들의 종속거리 Diff = Ceil(D/ρ)라 정의한다.

```
    J = 1
L:  inc = MIN(10-J, Ceil((J+5)/2-1)
    DOALL I = J, J+inc
    S1 :     A(3*I+1) = . . .
    S2 :              = A(2*I-4)
    END DOALL
    J = J+inc+1
    if J <= 10 goto L
```

그림 8. Polychronopoulous의 루프 분할

Polychronopoulous의 루프 분할 방법을 [그림 6] (a)

를 예를 들어서 설명한다면 Increment factor ρ는 2이고, Difference D는 (3*I+1) - (2*I-4) = I+5이다. 그래서 문장 S_1에서 실제 정의되는 배열의 원소들은 적어도 루프변수 I가 종속거리 Diff 만큼 증가하기 전까지는 문장 S_2에 의해서 사용되지 않는다는 것을 알 수 있다. 이 방식을 이용하면 [그림 8]과 같은 병렬루프로 변환한 결과를 얻을 수 있다.



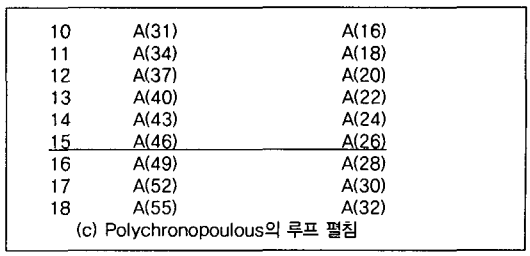


그림 9. 루프 분할 방법에 따른 루프 펼침

[그림 8]과 같은 병렬루프로 변환한 결과를 루프 펼침을 이용하여 종속성을 보인 것이 [그림 9] (c)이며, 루프 반복들이 네 개의 큰 블록으로 나뉘어 있다. [그림 9] (b)는 최소 종속거리를 이용한 루프 분할 방법에 의하여 병렬루프로 변환한 결과를 루프 펼침을 이용하여 종속성을 보인 것인데, 루프 반복들이 여섯 개의 큰 블록으로 나뉘어 있다. [그림 9] (a)를 보면 루프반복들이 세 개의 큰 블록으로 나뉘어 있는데, 이렇게 나누는 것이 루프 내에 존재하는 병렬성을 가장 최적화하여 루프를 분할한 것이다. [그림 9]를 통하여 Polychronopoulous의 루프 분할 방법은 최소 종속거리를 이용한 루프 분할 방법보다는 효율적이지만, 이 방법 역시 가장 효율적인 분할 방법이라고 할 수 없다는 사실을 알 수 있다. 뿐만 아니라 다음과 같은 문제점을 가지고 있다[6].

1) 루프의 첫 번째 반복에서 반드시 종속성이 존재하여야 하며, 그 종속성의 종속거리는 반드시 1보다 크거나 같아야 한다.
2) 루프 독립적인(Loop-independent) 종속성에 대하여 고려하지 않았다. 즉 루프의 첫 번째 반복에서 루프 독립적인(Loop-independent) 종속이 발생하면 변형방법의 적용이 불가능하다.

2.3 최초 종속성을 이용한 루프 분할

Polychronopoulous가 제안한 방법이 갖는 문제점을 보안하여 루프 내에 최초로 존재하는 종속성을 이용하여 보다 효율적인 분할 방법을 제안하였다[6].

본 분할 방법은 루프 내에 최초로 존재하는 종속성을 이용하여 종속 거리를 계산하여 Sink값을 알아내고, 현재의 Sink값과 다음 종속성의 Source값의 차이를 계산

하여 다음 종속성의 Source반복을 찾는다. 최초 종속성을 이용하여 병렬 루프로 변환한 결과를 루프 펼침을 이용하여 종속성을 보인 것이 [그림 9] (a)이다.

그러나, 본 분할 방법은 다음과 같은 문제점을 가지고 있다.

1) gcd(a,c)의 값이 1보다 큰 정수일 때 변형 방법의 적용이 불가능하다.
2) 최초 종속성이 처음부터 시작하는 경우에는 가능하나, 처음이 아닌 경우에는 최초 종속성이 발생하는 Source s 값이 주어져야 한다.
3) 흐름 종속성이 존재할 때만 가능하다. 즉 a*I+b > c*I+d 일 때만 가능하다.

III. 확장된 최초 종속 분할 방법

본 논문은 최초 종속성을 이용한 루프 분할 방법이 갖는 문제점을 보안하여 보다 효율적인 분할 방법을 제안하고자 한다.

1. GCD(Greatest Common Divisors) 테스트

gcd 테스트[2]는 대상 루프에 종속성이 존재할 가능성에 대해서 가장 간단하게 테스트할 수 있는 방법으로서, gcd 테스트에서 결과가 정수 값이 아니면 대상 루프에 종속성이 존재하지 않는 것이 확실하나, 결과가 정수 값이면 대상 루프에 종속성이 존재할 가능성이 있다. 또한 gcd 테스트의 결과로서 최대공약수 g=gcd(a,c)를 구할 수 있다. gcd 테스트에 대한 알고리즘은 다음과 같다.

〈알고리즘 1〉 (Euclid's Algorithm)

/* a, b가 정수일 때 최대공약수 g=gcd(a, b) */

Step 1 : u=|a|, v=|b|

Step 2 : If v=0, then g=u, *terminate the algorithm*

Step 3 : r=u mod v, u=v, and v=r. goto step 2

2. 종속식의 일반해를 구하는 방법

우리가 대상으로 삼고 있는 루프에서 두 첨자식 a*I+b와 c*I+d로부터 종속식 a*x-c*y=d-b를 얻을수 있다. Euclid 알고리즘을 이용하여 두 정수 a, c의 최대공약수 g와 Extended Euclid 알고리즘을 이용하여 g=a*u+c*v를 만족하는 u와 v를 구할 수 있다. 또한 r=d-b라면 종속식은 a*x-c*y=r이 된다. 그러면 다음 정리를 이용하여 x와 y의 해를 구할 수 있다[5].

[정리 1] a*x-c*y=r은 linear diophantine equation이고, g=gcd(a,c), r/g, u, 그리고 v는 g=a*u-c*v를 만족하는 정수라고 가정하자. 그러면 위 식을 만족하는 모든 해(x_t, y_t)들의 집합은 다음과 같다.

$$x_t = u*(r/g)-t*(c/g), \quad y_t = v*(r/g)-t*(a/g)$$

3. 최초 종속성의 Source s를 구하는 방법

위 정리를 이용하여 구한 해들은 종속식을 만족하는 정수영역내의 해들이기 때문에, 대상루프 영역 내에 존재하는 해를 아래의 정리 2를 이용하여 구해야 한다.

[정리 2] 대상 루프의 하한값을 1이라 하고, 상한값을 U(Upper Bound)라 하면 x_t, y_t가 루프 범위 내에 속하는 t값의 범위는 다음과 같다.

$$\alpha = \text{Ceil}(-(1-(u*(r/g)))/(c/g))$$
$$\beta = -(U-u*(r/g))/(c/g)$$
$$\gamma = \text{Ceil}((1-v*(r/g))/(a/g))$$
$$\delta = (U-v*(r/g))/(a/g)$$

그러면 다음 식을 만족하는 t를 얻을 수 있다.

$$\max(\min(\alpha, \beta), \min(\gamma, \delta)) \le t \le \min(\max(\alpha, \beta), \max(\gamma, \delta))$$

위 정리를 이용하여 얻은 해들의 쌍들이 루프 내에 존재하는 모든 종속성의 Source와 Sink를 의미하며, 가장 적은 값이 최초 종속성의 Source s를 의미한다.

4. 확장된 최초 종속성을 이용한 루프 분할 방법

최초 종속성을 이용한 루프 분할 방법이 갖는 문제점을 보안하여 보다 효율적인 분할 방법을 적용한 알고리

즘은 다음과 같다.

〈 알고리즘 2 〉 (흐름 종속성)

/* a*x+b=c*y+d 형태의 식을 ma*x+mb*y=mc형태의 식으로 변형하여 사용한다. */

coef : difference에서 루프변수 I의 계수, 즉 a-c값을 뜻한다.

U : Upper Bound

g : gcd(ma, mb, mc)

s : 대상 순차루프의 맨처음에 존재하는 종속성의 Source반복이다. 최초 종속성을 말하며, find_first(ma, mb, mc, U)에 의해서 구한 값이다.

t : Sink값, 즉 다음 병렬루프로의 변환 대상이 되는 순차루프의 첫 번째 반복

$Dist_1$: 대상 순차루프의 맨 처음에 존재하는 종속성의 Source s로부터의 종속거리

q : 현재의 Sink값과 다음 종속성의 Source값의 차이

k : 종속성의 순서 값

$Dist_k$: k번째에 존재하는 종속성의 Source로부터의 종속거리

NS : t+q, 즉 루프분할시 다음 순차루프 블록의 처음에 존재하는 종속성의 Source반복이다.

i : 루프분할 순서값

SC_i : t+q-1, i번째 루프분할 값

Step 1 : /* 초기값을 설정한다 */

i = 0, ma=a, mb=-c, mc=d-b, SC_i = 1

Step 2 :/* 최대공약수 g=gcd(ma,mb,mc)를 구한다 */

if g가 정수가 아니면 terminate,

else if g>1 then ma=ma/g, mb=mb/g, mc=mc/g

coef = ma+mb

Step 3 : /* 대상 순차루프의 맨 처음에 존재하는 종속성의 Source s를 구한다. */

s = find_first(ma,mb,mc,U)

Step 4 : /* 첫 번째 종속 거리를 구한다 */

$Dist_1$ = Ceil((coef*s - mc)/-mb)

Step 5 : /* 첫 번째 Sink값을 구한다.

만일 $Dist_1$가 0이면 increment factor c만큼 증가 시킨다 */

if $Dist_1$ = 0 then t = s-mb,

else t = s + $Dist_1$

Step 6 : /* 다음 순차루프 블록의 첫번째 종속성의 Source값을 구한다. */

Ns = t+q where q = (t-s) mod (-mb)

Step 7 : /* 루프를 분할할 값을 구한다. 만일 그 값이 Upper Bound보다 크면 끝낸다. */

SC_i = Ns-1,

if (SC_i >= U) then SC_i = U and terminate

Step 8 : /* 다음번 종속거리와 Sink값을 구한다. */

k = (Ns-s)/c+1

$Dist_k$ = $Dist_1$ + (k-1) * coef

t = Ns + $Dist_k$

i = i+1 goto step 6

〈 알고리즘 3 〉 (역종속성)

/* a*x+b=c*y+d 같은 형태의 식이 주어 졌을 때 Difference D값 (a*I+b)-(c*I+d) 이 음수일 경우 c*x+d=a*y+b 식으로 변형하여 <알고리즘 2>를 적용한다. */

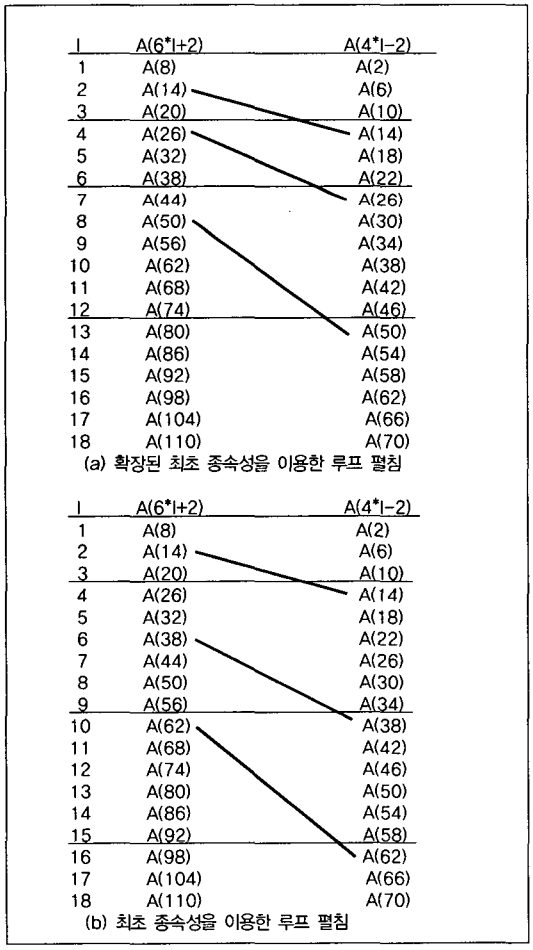
위 <알고리즘 2>는 흐름 종속성일 때의 루프 분할 방법으로서, a*x+b=c*y+d 형태의 식을 ma*x+mb*y=mc 형태의 두개의 변수를 가진 일반적인 linear diophantine 식으로 변형한 후, 먼저 gcd 테스트를 함으로서 대상 루프 내에 종속성이 존재할 가능성에 대해서 간단하게 테스트하여 결과가 정수 값이 아니면 대상 루프에 종속성이 존재하지 않는 것이 확실하기 때문에 더 이상 진행하지 않고 중단한다. 그림10과 같이 gcd 테스트의 결과로서 최대공약수 g=gcd(ma,mb,mc)가 1보다 큰 정수일 경우에는 g값을 이용하여 새로운 상수값을 구하여 ma'*x+mb'*y=mc'(여기서 ma'=ma/g, mb'=mb/g, 그리고 mc'=mc/g)식으로 변형하여 사용함으로서 최초종속성에서의 첫 번째 문제점을 해결하였다. 또한 최초종속성을 이용한 알고리즘에서는 최초 종속성의 Source s를 구하지 않고 주어진 s를 사용하였으나, 확장된 알고리즘은 Extended Euclid 알고리즘을 이용하여 최초 종속성의 Source s를 구하는 기능을 더 하였다. 그리고 최초종속성을 이용한 알고리즘에서는 흐

름 종속성이 존재할 때만 가능하였으나, 본 알고리즘에서는 종속성이 흐름 종속성(<)일 때 뿐 만 아니라, [그림 12]와 같이 역종속성(>)일 때, 여러 종속성이 혼재해 있을 경우((=, <)인 경우와 (=, >))에서도 <알고리즘 3>을 사용함으로서 루프 분할이 가능하다.

IV. 성능 분석

```
        DO I = 1, 18
S1 :      A(6*I+2) = ...
S2 :               = A(4*I-2)
        END DO
```

그림 10 . gcd값이 1보다 큰 루프



I	A(6*I+2)	A(4*I-2)
1	A(8)	A(2)
2	A(14)	A(6)
3	A(20)	A(10)
4	A(26)	A(14)
5	A(32)	A(18)
6	A(38)	A(22)
7	A(44)	A(26)
8	A(50)	A(30)
9	A(56)	A(34)
10	A(62)	A(38)
11	A(68)	A(42)
12	A(74)	A(46)
13	A(80)	A(50)
14	A(86)	A(54)
15	A(92)	A(58)
16	A(98)	A(62)
17	A(104)	A(66)
18	A(110)	A(70)

(a) 확장된 최초 종속성을 이용한 루프 펼침

I	A(6*I+2)	A(4*I-2)
1	A(8)	A(2)
2	A(14)	A(6)
3	A(20)	A(10)
4	A(26)	A(14)
5	A(32)	A(18)
6	A(38)	A(22)
7	A(44)	A(26)
8	A(50)	A(30)
9	A(56)	A(34)
10	A(62)	A(38)
11	A(68)	A(42)
12	A(74)	A(46)
13	A(80)	A(50)
14	A(86)	A(54)
15	A(92)	A(58)
16	A(98)	A(62)
17	A(104)	A(66)
18	A(110)	A(70)

(b) 최초 종속성을 이용한 루프 펼침

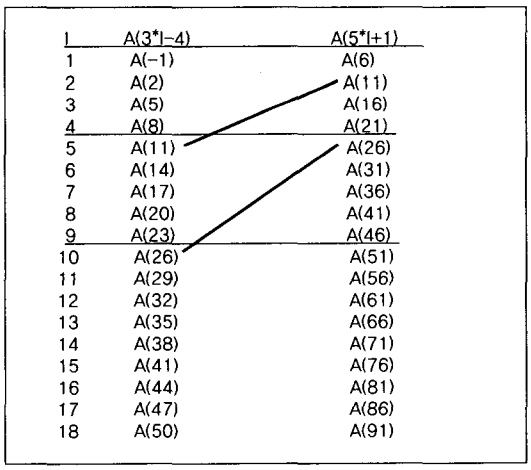
그림 11. gcd값이 1보다 큰 루프의 루프 펼침

[그림 10]과 같이 gcd값이 1보다 큰 예가 주어 졌을 때, 먼저 3장 3절에서 소개한 최초 종속성의 Source s를 구하는 방법에 의하여 s=2 값을 구한다. 그리고 <알고리즘 2>를 이용하여 주어진 6*x+2=4*y-2 형태의 식을 6*x-4*y=-4형태의 두개의 변수를 가진 일반적인 linear diophantine 식으로 변형한 후, gcd 테스트를 하였을 때 최대공약수 g=gcd(6, -4, -4)=2 임을 알 수 있고, g값을 이용하여 새로운 상수 값을 구하여 3*x-2*y=-2식으로 변형하여 사용한다. 이때 Increment factor ρ는 2이고, Difference D는 (3*I+1)-(2*I-1) = I+2이다. 위에서 구한 식과 값들을 이용하여 [그림 11] (a)와 같은 루프 펼침을 할 수 있다.

그러나, 최초 종속성을 이용한 방법으로는 [그림 11] (b)에서 보는 것처럼 첫 번째의 종속이 두 번째에 발생함으로서 최초 종속성의 Source s값을 별도로 제공해 주어야 한다. 그리고, 6*x+2=4*y-2 형태의 식을 그대로 사용함으로써 Increment factor ρ는 4이고, Difference D는 (6*I+2) - (4*I-2) = 2*I+4 와 같은 식과 값들을 이용함으로서 [그림 11] (b)와 같은 잘못된 결과의 루프 펼침을 생성한다.

```
        DO I = 1, 18
S1 :      A(3*I-4) = ...
S2 :               = A(5*I+1)
        END DO
```

그림 12. 역종속성이 존재하는 루프



I	A(3*I-4)	A(5*I+1)
1	A(-1)	A(6)
2	A(2)	A(11)
3	A(5)	A(16)
4	A(8)	A(21)
5	A(11)	A(26)
6	A(14)	A(31)
7	A(17)	A(36)
8	A(20)	A(41)
9	A(23)	A(46)
10	A(26)	A(51)
11	A(29)	A(56)
12	A(32)	A(61)
13	A(35)	A(66)
14	A(38)	A(71)
15	A(41)	A(76)
16	A(44)	A(81)
17	A(47)	A(86)
18	A(50)	A(91)

그림 13. 역종속성이 존재하는 루프의 확장된 최초 종속성을 이용한 루프 펼침

[그림 12]의 예에서 Difference D값이 (3*I-4) - (5*I+1) = -(2*I+5) 이므로 D값이 항상 음수임을 알 수 있다. 그러므로, 본 예는 항상 역종속성(>)이 존재하는 루프이고, 이 경우에는 <알고리즘 3>을 사용하여 a*x+b=c*y+d 형태의 식을 c*x+d=a*y+b 식으로 변형하여 <알고리즘 2>를 적용한다. 그리하여 [그림 13]과 같은 루프 분할이 가능하다. 그러나, 최초 종속성을 이용한 방법에서는 Difference D값이 음수일 경우에 정확한 루프 편침을 제공하지 못한다.

위의 두 가지 예를 통하여, 본 논문에서 제안한 루프 펼침 방법이 기존의 최초 종속성을 이용한 방법의 문제점들을 해결하고, 비균일 단일루프에서 보다 효율적인 루프 분할 방식임을 알 수 있다.

V. 결 론

본 논문에서는 단일루프에서 종속거리가 불규칙한 경우, 즉 루프 내 배열의 두 첨자식이 a*I+b와 c*I+d이며 a≠0, c≠0, a≠c인 경우에 기존의 연구된 최소 종속거리를 이용한 루프 분할 방법, Polychronopoulous의 루프 분할 방법, 그리고 최초 종속성을 이용한 루프 분할 방법을 각각 소개하였고, 그들의 장·단점을 설명하였다.

본 논문에서는 그중에서도 가장 효율적인 최초 종속성을 이용한 루프 분할 방법의 단점을 개선하여 확장된 최초 종속성을 이용한 루프 분할 방법을 제시하였다.

앞으로 본 연구를 활용하여 이중루프뿐만 아니라 다중루프에 대한 연구가 필요할 것이다.

참 고 문 헌

[1] R. Allen and K. Kennedy, "Automatic translation of Fortran Programs to Vector Form," ACM Transactions on Programming Language and Systems, Vol.9, No.4, pp.491-532. October 1987.

[2] U. Banerjee, Loop Transformations for Restructuring Compilers: The Foundations, Kluwer Academic Publishers, 1993.

[3] D. E. Knuth, "An Empirical Study of Fortran Programs," Software Practice and Experience, pp.105-133, 1971.

[4] C. D. Polychronopoulous, "Compiler Optimizations for Enhancing Parallelism and Their Impact on Architecture Design," IEEE Transactions on Computers, Vol.37, No.8, pp.991-1004, August 1988.

[5] H. Zima and B. Chapman, "Supercompilers for Parallel and Vector Computer," New York, N.Y., ACM press, 1991.

[6] 심재찬, 조철권, 이만호, "단일루프에서 병렬성 증가를 위한 루프의 최적화 기법", 한국정보과학회 가을 학술발표논문집, Vol.21, No.2, pp.875-878, 1994.

저 자 소 개

정 삼 진(Sam-Jin Jeong) 정회원

- 1979년 2월 : 경북대학교 고분자공학과(공학사)
- 1986년 12월 : 인디애나대학교 전산학과(이학석사)
- 2000년 8월 : 충남대학교 전산학과(이학박사)
- 1988년 2월~2002년 2월 : 삼성종합기술원, 삼성전자(주) 선임 연구원
- 2002년 3월~2007년 2월 : 혜천대학 전산정보처리과 교수
- 2007년 3월~현재 : 천안대학교 정보통신학부 교수

<관심분야> : 병렬 컴파일러, 프로그래밍 언어, 병렬 처리, 컴파일러