중첩루프에서 병렬화를 위한 자료 종속성제거

송 월 봉† · 박 두 순††

요 약

본 논문에서는 루프구조의 효율적인 병렬수행을 위한 병렬성 추출에 대하여 불변과 가변 종속거리에 모두 적용할 수 있는 통합된 새로운 기법을 제시한다. 이것은 컴파일시간에 순차 루프를 중첩된 DOALL루프로의 자동 변환에 대한 절차로서, 중첩 루프의 전체적인 병렬화를 하기 위하여 분장들을 반복적으로 수행시키는 것에 의해서 자료 종속을 효과적으로 제거하는 알고리즘이다. 본 논문에 제시된 방법은 성능평가에서도 매우 뛰어난 방법임을 보였다.

Data Dependency Elimination for Parallelism in Nested Loops

Weol-Bong Song† · Doo-Soon Park††

ABSTRACT

In this paper, a general method the extracting parallelism in nested loops is presented. This is a procedure for the automatic conversion of a sequential loop into a nested parallel DOALL loops at compile time. Moreover, this algorithm can be applicable where the dependency relation is both uniform and non-uniform in distance. Our test results show the proposed scheme is superior to conventional methods. The algorithm which effectively removes these kind of data dependencies is developed in order to present the total parallelization of nested loops.

1. 서 론

21세기 정보화 시대에서의 다양한 컴퓨터의 응용분야는 기존의 컴퓨터 구조로서는 효율적인 해결책을 제공하기가 어려우며 따라서 다양한 형태의 정보를 신속히 처리할수 있는 새로운 구조의 컴퓨터가 요구되고 있다.

이에따라 IBM 3090/VF[6]와 같은 고도의 파이프라인(pipeline)으로 연결된 벡터(vector) 컴퓨터와 Cray X-MP[6], Cary-2[15], Alliant FX/8[3] 시스템처럼 여러개의 벡터 처리기들이 한 시스템 내에서 상호 연결되어 프로그램을 병렬처리 할 수 있는 형태의 시스템들이 등장하였다. 그러나, 단순히 더 많은 프로세서를 시스템에 부착하는 것만으로는 시스템의 성능 향상에 커다란 영향을 미치지 못한다. 따라서 대규모 병렬처리 시스템 상에서 처리 속도의 극대화를 꾀하기 위해서는 하드웨어 뿐만 아니라 병렬성 추출 알고리즘과 같은 소프트웨어적인 요소들이 함께 고려되어야 한다.

이러한 연구는 기존의 프로그래밍 환경과는 다른 것으로서 병렬 프로그래밍 환경(parallel programming environment)이라고 부른다. 이러한 병렬 프로그래밍 환경중에서 직접 병렬 언어로 프로그램을 작성하는 것이나 이미 제작된 순차 프로그램을 새로운 병렬언어로 수정하는 작업은 주어진 병렬 처리 시스템의 구조를 이해하고 병렬 처리에 관한 전문지식을 새로이 익혀야 하는 등 프로그래머에게 상당한 부담이 되기 때문에 순차 프로그램을 자동적으로 병렬처리가 가능한 형태로 변환해 주는 재구성 컴파일러(reconstruction compiler)

† 정 회 원 : 인천전문대학 전자계산과 교수
†† 정 회 원 : 순천향대학교 공과대학 컴퓨터학부 교수
논문접수 : 1998년 3월 13일, 심사완료 : 1998년 5월 12일

나는 기법이 많이 연구되고 있다.

재구성 컴파일러는 프로그램의 병렬성(parallelism)을 추출하기 위해서 자료 종속성(data dependency)을 분석한다. 기존의 재구성 컴파일러는 자료 종속 거리가 모두 불변(uniform)인 경우에 제한되어서 처리된다. 일반적인 프로그램의 경우 이를 분석해보면 자료 의존성이 사이클을 이루거나 반종속, 출력종속이 섞여 있는 경우가 대부분이므로 기존의 재구성 컴파일러 가지고는 해결할 수가 없다. 이를 위하여 병렬 컴퓨터(parallel computer)에서 꼭 필요로 하는 자료종속의 거리가 불변만 있거나, 가변(non-uniform)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.

기존에 제시되어 있는 루프변환 방법들은 자료종속 거리가 불변인 경우를 처리하는 방법들과 가변인 경우를 처리하는 방법등으로 나누어 볼 수 있다. 자료종속 거리가 불변인 경우는 tiling[18], interchanging[4], skewing[4], selected cycle shrinking[13,14], Holland[8]방법, Chen & Wang[7]방법, Shang& Fortes의 방법[23], unimodular[4]와 혼합된 unimodular방법[22], non-unimodular[1]등이 있으며 자료종속거리가 가변인 경우는 DCH[25]방법이 처음으로 제시되었고 IDCH[21]방법, CDCH[10]방법 등이 있다. 이러한 기존의 방법들은 모두다 자료종속성을 분석해서 분할(Partition)한 다음 스케쥴링하는 방법들을 택하고 있으며 이 방법들로는 효율성에서 한계점을 가지고 있다. 이를 위해서 본 논문에서는 이 방법들보다는 매우 효율적인 자료 종속을 제거하는 방법을 제시하고자 한다.

본 논문에서는 재구성 컴파일러를 구성하기 위해서 우선적으로 필요한 자료종속의 거리가 가변만 있는 경우, 불변만 있는 경우, 가변과 불변이 섞여져 있는 경우에도 모두 적용할 수 있는것으로서 문장들을 수행하는 것에 의해 자료종속성을 제거하는 알고리즘을 제시하였으며, 이에 대한 성능평가를 하였다.

2. 자료 종속성

프로그램의 종속성은 자료(data) 종속성과 제어(control) 종속성으로 구성된다. 제어 종속성은 조건문에 의해서 발생되는 종속성으로 자료 종속성과 유사한 방법으로 처리할 수 있기 때문에[16] 본 논문에서는 자료 종속성에 대해서만 설명한다.

자료 종속성은 흐름 종속(flow dependence : read-after-write), 반 종속(anti dependence : write-after-read) 그리고 출력 종속(output dependence: write-after-write)으로 나눌 수 있다[4]. 두 문장 S_i와 S_j사이에서, S_i에서 변수 X가 선언(define)되고 S_j에서 X가 사용(use)되며 S_i가 S_j이전에 수행되면 흐름 종속성($S_i\ \delta\ S_j$)이 존재하고 두 문장 S_i와 S_j에서, 같은 변수가 선언되고, S_i와 S_j이전에 수행되면 출력 종속성($S_i\ \delta^o\ S_j$)이 존재하며 문장 S_i에서 X를 사용하고 S_j에서 X가 정의되며 S_i가 S_j이전에 수행되면 반종속성($S_i\ \delta^a\ S_j$)이 존재한다.

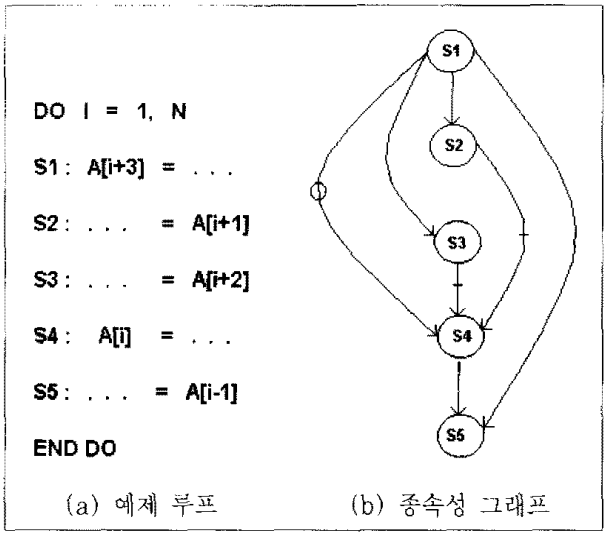
이 중에서 흐름 종속은 참 종속(true dependence)이라 하고, 다른 두가지 종속은 인위 종속(artificial dependence)이라 한다. 그 이유는 출력 종속과 반 종속은 프로그램을 변환함으로써 제거할 수 있으나, 흐름 종속은 제거할 수 없기 때문이다[11].

자료 종속성을 분석하는 방법에는 여러가지 종류가 있다. 그 중에서 가장 간단한 방법은 seperability test[27]이다. 이 방법은 두 문장에서 사용된 동일한 변수에 대해서, 사용된 두개의 첨자식이 공통적인 루프 변수를 한개 이하로 표현된 경우에만 적용될 수 있는 방법이다. 또 GCD(Greatest Common Divisor) test[2]는 루프의 영역과는 상관없이 종속 방정식이 정수해를 가지는 지의 여부를 판단하는 방법이다.

그러나 이 방법들은 모두 자료 종속성의 거리가 불변(uniform: constant)인 경우에만 적용이 가능하고 자료 종속성의 거리가 가변(non-uniform: variable)인 경우에는 적용하지 못한다. 배열 변수의 첨자식과 자료 종속성에 대한 연구[22,24]에 의하면 자료 종속 형태에서 작은 부분(13.65%)만이 불변 종속거리를 갖는다. 즉, 86.35%가 가변 종속거리를 갖는다. 따라서 병렬처리를 효율적으로 수행하기 위해서는 불변 종속거리는 물론 가변 종속거리가 존재하는 루프에 대해서도 적용할 수 있어야 한다.

(그림 1)의 종속 그래프에서는 4개의 흐름 종속(S1 δ S2, S1 δ S3, S1 δ S5, S4 δ S5), 2개의 반 종속(S2 δ^a S4, S3 δ^a S4), 그리고 하나의 출력 종속(S1 δ^o S4)이 존재한다. 여기서, ——〉는 흐름 종속(flow dependence), —|-〉 는 반 종속(anti dependence) 그리

고, 0 이면 출력 종속(output dependence)를 의미한다.



(그림 1) 루프의 종속성 그래프
(Fig. 1) Dependence graph

자료 종속 거리(data dependence distance)는 최적화된 병렬 루프를 변환하는데 중요한 정보로 사용되며, 첨자식에 따라 불변 종속 거리와 가변 종속 거리로 나눌 수 있다. 만약 하나의 루프 첨자 변수로만 구성된 첨자식이 각각 ai+b, ci+d(여기서 a, b, c, d는 정수, i는 루프 변수)라 하면, 첨자식에서 a=c이면, 종속 거리는 |(ai±b)-(ci±d)|인 불변 종속 거리가 존재하고 a≠c이면 가변 종속 거리가 존재한다.

3. 자료 종속성 제거 알고리즘

이 장에서는 7개의 정의와 2개의 정리 그리고 병렬성을 추출하기 위한 자료 종속성 제거 알고리즘을 제시한다.

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

DMLCS(k,l,m)에서 루프안에 있는 문장은 n개이고 2개의 중첩된 루프를 가지면서 종속성 행렬의 초기상태를 나타내는 경우, 즉 DMLCS(2,1,n)의 표시 방법은 다음과 같다.

$$DMLCS(2,1,n)=\begin{pmatrix}(X_{11},Y_{11}),(X_{12},Y_{12}),\ldots,(X_{1n},Y_{1n})\\(X_{21},Y_{21}),(X_{22},Y_{22}),\ldots,(X_{2n},Y_{2n})\\\vdots\\\cdots,(X_{ij},Y_{ij}),\cdots\\\vdots\\(X_{n1},Y_{n1}),(X_{n2},Y_{n2}),\ldots,(X_{nn},Y_{nn})\end{pmatrix}$$

(단, 각 쌍의 원소는 임의의 i, j(1≤i, j≤n)에 대하여 $X_{ij}=(a_{ij}\oplus u_{ij},\ b_{ij}\oplus v_{ij})$, $Y_{ij}=(c_{ij}\oplus w_{ij},\ d_{ij}\oplus x_{ij})$가 된다.) DMLCS의 임의의 원소 $(X_{ij}, Y_{ij}) = ((a_{ij}\oplus u_{ij}, b_{ij}\oplus v_{ij}), (c_{ij}\oplus w_{ij}, d_{ij}\oplus x_{ij}))$ 의 의미는 다음과 같다.

① 문장 S_i로 부터 문장 S_j로 자료 종속이 존재하는 경우 a_{ij}와 b_{ij}는 중첩루프의 첫 번째 첨자식들 간의 diophantine 방정식 $v_{ij} * I' + g = u_{ij} * I'' + h$ 의 초기해이고 c_{ij}와 d_{ij}는두 번째 중첩루프의 첨자식들 간의 diophantine 방정식 $x_{ij} * J' + g' = w_{ij} * J'' + h'$ 의 초기해 이다. a_{ij} 와 c_{ij}는 종속관계에 있는 문장 S_i의 첫 번째 첨자와 두 번째 첨자의 초기치가 되고 u_{ij} 와 w_{ij} 는 각첨자의 증가치가 되며, b_{ij} 와 d_{ij} 는 종속관계에 있는 문장 S_j의 첫 번째 첨자와 두 번째 첨자의 초기치가 되고 v_{ij} 와 x_{ij}는 각 첨자의 증가치가 된다.

단, a_{ij} 또는 b_{ij} 또는 c_{ij} 또는 d_{ij} 가 0 인 경우에는 u_{ij} 또는 v_{ij} 또는 w_{ij} 또는 x_{ij} 가 초기치 및 증가치가 된다.

② DMLCS에서 원소가 0인 경우는 자료 종속성이 존재하지 않음을 의미한다.

③ ⊕는 첨자식 SI+t의 형태에서 상수 S와t를 한번에 표시하기 위한 기호이다.

④ $@((a_{ij}\oplus u_{ij}, b_{ij}\oplus v_{ij}),\ (c_{ij}\oplus w_{ij}, d_{ij}\oplus x_{ij}))$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 2)와 같은 두 개의 중첩루프를 갖는 프로그램이 있다고 하자.

```
DO I = 1, M
  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,J-2)
  ENDDO
ENDDO
```

(그림 2) 중첩 루프의 예
(Fig. 2) Example of nested loop

(그림 2)는 2개의 중첩된 루프를 가졌고, 루프 안에 있는 문장의 갯수는 3이다. 이에 대한 종속성 행렬의 초기 상태는 DMLCS(2,1,3)이다. 여기서 변수 B의 경우 문장 S_2로부터 문장 S_3으로 가는 흐름 종속이 존재한다. 이경우에 자료종속성은 5*I′와 I″-2의 값이 같아지는 정수해 I′,I″의 값을 구해야 하고 4*J′와 J″-3의 값이 같아지는 정수해 J′,J″값을 구해야한다. 이에 대한 초기해는 DMLCS(2,1,3) 행렬의 2행 3열에 나타내어 진다.

(그림 2)의 DMLCS(2,1,3)은 (그림 3)이다.

	S_1	S_2	S_3
S_2	0	@((1⊕1,3⊕1),(1⊕1,4⊕1))	@((1⊕1,4⊕1),(1⊕1,3⊕1))
S_3	((0⊕4,0⊕5),(0⊕3,0⊕4))	((1⊕1,9⊕5),(0⊕3,0⊕4))	((1⊕1,7⊕5),(1⊕1,7⊕4))
	((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 3) (그림 2)의 DMLCS(2,1,3)
(Fig. 3) DMLCS(2,1,3) of (Fig. 2)

(그림 3)에서 문장 S_2에서 문장 S_3으로의 자료종속성이 존재하고, 초기해는 ((1⊕1,7⊕5),(1⊕1,7⊕4))이며, ((1⊕1,7⊕5),(1⊕1,7⊕4)) 의 의미는 다음과 같다

즉, ((1⊕1,7⊕5),(1⊕1,7⊕4)) = $((a_{23}\oplus u_{23},\ b_{23}\oplus v_{23}),\ (c_{23}\oplus w_{23},\ d_{23}\oplus x_{23}))$가 되므로 종속관계는 앞의 설명 ①에 의하여 $I'=a_{23}=1$, $J'=c_{23}=1$ 인 경우 S_2는 $\underline{B(5,4)}=A(-3,-3)+B(-3,3)+C(2,2)$가 되고 $I''=b_{23}=7$, $J''=d_{23}=7$인 경우 S_3는 $C(49,42)=A(2,4)+\underline{B(5,4)}+C(6,5)$가 되어 종속관계가 존재함을 알 수 있고 또한, 종속관계에 있는 S_2의 I′와J′의 증가치가 각각 1이므로 $I'=a_{23}+u_{23}=1+1=2$, $J'=c_{23}+w_{23}=1+1=2$인 경우S_2는 $\underline{B(10,8)}=A(-2,-2)+B(-2,6)+C(5,6)$가 되고 S_3의 I″과J″의 증가치는 각각 5와4이므로$I''=b_{23}+v_{23}=7+5=12$, $J''=d_{23}+x_{23}=7+4=11$인 경우S_3는 $C(84,66)=A(7,8)+\underline{B(10,8)}+C(11,9)$가 되어 증가치에 따라서 종속관계가 형성됨을 알 수 있다. 그러므로 ((1⊕1,7⊕5),(1⊕1,7⊕4))는 문장 S_2에서 문장 S_3로의 종속관계를 함축시켜 표시하고 있음을 알 수 있다.

〈 정의 2 〉
종속성 행렬 DMLCS(m,n,p)와 DMLCS(m,1,p)의 곱은 연산자 ◎에 대한 연산으로 다음과 같은 종속성 행렬 DMLCS(m,n+1,p)가 된다.

앞에서 정의한 DMLCS의 표현법을 이용하면

$$\begin{vmatrix} (X_{11},Y_{11}),(X_{12},Y_{12}),\cdots,(X_{1n},Y_{1n}) \\ (X_{21},Y_{21}),(X_{22},Y_{22}),\cdots,(X_{2n},Y_{2n}) \\ \vdots \\ (X_{n1},Y_{n1}),(X_{n2},Y_{n2}),\cdots,(X_{nn},Y_{nn}) \end{vmatrix} \odot \begin{vmatrix} (X_{11},Y_{11}),(X_{12},Y_{12}),\cdots,(X_{1n},Y_{1n}) \\ (X_{21},Y_{21}),(X_{22},Y_{22}),\cdots,(X_{2n},Y_{2n}) \\ \vdots \\ (X_{n1},Y_{n1}),(X_{n2},Y_{n2}),\cdots,(X_{nn},Y_{nn}) \end{vmatrix}$$

$$= \begin{vmatrix} A & \cdots & B \\ \vdots & & \\ C & \cdots & D \end{vmatrix}$$

가 되며 A,B,C,D의 구체적인 내용은 다음과 같다.

$$A = \begin{vmatrix} \begin{pmatrix} ((a_{11}\oplus u_{11},\ b_{11}\oplus v_{11}),(c_{11}\oplus w_{11},\ d_{11}\oplus x_{11})) \\ ((a_{11}\oplus u_{11},\ b_{11}\oplus v_{11}),(c_{11}\oplus w_{11},\ d_{11}\oplus x_{11})) \end{pmatrix} \\ \text{◎}\cdots\text{◎} \\ \begin{pmatrix} ((a_{1n}\oplus u_{1n},\ b_{1n}\oplus v_{1n}),(c_{1n}\oplus w_{1n},\ d_{1n}\oplus x_{1n})) \\ ((a_{n1}\oplus u_{n1},\ b_{n1}\oplus v_{n1}),(c_{n1}\oplus w_{n1},\ d_{n1}\oplus x_{n1})) \end{pmatrix} \end{vmatrix}$$

$$B = \begin{vmatrix} \begin{pmatrix} ((a_{11}\oplus u_{11},\ b_{11}\oplus v_{11}),(c_{11}\oplus w_{11},\ d_{11}\oplus x_{11})) \\ ((a_{1n}\oplus u_{1n},\ b_{1n}\oplus v_{1n}),(c_{1n}\oplus w_{1n},\ d_{1n}\oplus x_{1n})) \end{pmatrix} \\ \text{◎}\cdots\text{◎} \\ \begin{pmatrix} ((a_{1n}\oplus u_{1n},\ b_{1n}\oplus v_{1n}),(c_{1n}\oplus w_{1n},\ d_{1n}\oplus x_{1n})) \\ ((a_{nn}\oplus u_{nn},\ b_{nn}\oplus v_{nn}),(c_{nn}\oplus w_{nn},\ d_{nn}\oplus x_{nn})) \end{pmatrix} \end{vmatrix}$$

$$C = \begin{vmatrix} \begin{pmatrix} ((a_{n1}\oplus u_{n1},\ b_{n1}\oplus v_{n1}),(c_{n1}\oplus w_{n1},\ d_{n1}\oplus x_{n1})) \\ ((a_{11}\oplus u_{11},\ b_{11}\oplus v_{11}),(c_{11}\oplus w_{11},\ d_{11}\oplus x_{11})) \end{pmatrix} \\ \text{◎}\cdots\text{◎} \\ \begin{pmatrix} ((a_{nn}\oplus u_{nn},\ b_{nn}\oplus v_{nn}),(c_{nn}\oplus w_{nn},\ d_{nn}\oplus x_{nn})) \\ ((a_{n1}\oplus u_{n1},\ b_{n1}\oplus v_{n1}),(c_{n1}\oplus w_{n1},\ d_{n1}\oplus x_{n1})) \end{pmatrix} \end{vmatrix}$$

$$D=\begin{pmatrix} \begin{pmatrix}((a_{n1}\oplus u_{n1}, b_{n1}\oplus v_{n1}),(c_{n1}\oplus w_{n1}, d_{n1}\oplus x_{n1}))\\ ((a_{1n}\oplus u_{1n}, b_{1n}\oplus v_{1n}),(c_{1n}\oplus w_{1n}, d_{1n}\oplus x_{1n}))\end{pmatrix} \\ \cdots\cdots \\ \begin{pmatrix}((a_{nn}\oplus u_{nn}, b_{nn}\oplus v_{nn}),(c_{nn}\oplus w_{nn}, d_{nn}\oplus x_{nn}))\\ ((a_{nn}\oplus u_{nn}, b_{nn}\oplus v_{nn}),(c_{nn}\oplus w_{nn}, d_{nn}\oplus x_{nn}))\end{pmatrix} \end{pmatrix}$$ 가 된다.

여기서 종속성 행렬의 한 원소중 A의 경우,

원소들 중에

$$\begin{pmatrix}((a_{11}\oplus u_{11}, b_{11}\oplus v_{11}),(c_{11}\oplus w_{11}, d_{11}\oplus x_{11}))\\ ((a_{11}\oplus u_{11}, b_{11}\oplus v_{11}),(c_{11}\oplus w_{11}, d_{11}\oplus x_{11}))\end{pmatrix}$$

는 자료 종속성을 계산하기 위해 사용되는 행렬의 형태이다. 또한,

$$\begin{pmatrix} \begin{pmatrix}((a_{11}\oplus u_{11}, b_{11}\oplus v_{11}),(c_{11}\oplus w_{11}, d_{11}\oplus x_{11}))\\ ((a_{11}\oplus u_{11}, b_{11}\oplus v_{11}),(c_{11}\oplus w_{11}, d_{11}\oplus x_{11}))\end{pmatrix} \\ \cdots\cdots \\ \begin{pmatrix}((a_{1n}\oplus u_{1n}, b_{1n}\oplus v_{1n}),(c_{1n}\oplus w_{1n}, d_{1n}\oplus x_{1n}))\\ ((a_{n1}\oplus u_{n1}, b_{n1}\oplus v_{n1}),(c_{n1}\oplus w_{n1}, d_{n1}\oplus x_{n1}))\end{pmatrix} \end{pmatrix}$$

에서 연산자 ◎는 종속성 행렬에서 한 개의 원소가 n개의 원소들로 구성되는 것을 나타내며 각각의 원소들이 모두 자료 종속성을 계산하는데 사용된다.

〈 정의 3 〉
$((a_{ij}\oplus u_{ij}, b_{ij}\oplus v_{ij}), (c_{ij}\oplus w_{ij}, d_{ij}\oplus x_{ij}))$는 문장 S_i로부터 문장 S_j로 종속관계를 나타내고,
$((a_{kl}\oplus u_{kl}, b_{kl}\oplus v_{kl}), (c_{kl}\oplus w_{kl}, d_{kl}\oplus x_{kl}))$이 문장 S_j로 부터 문장 S_k로 종속관계를 나타내므로
$\begin{pmatrix}((a_{ij}\oplus u_{ij}, b_{ij}\oplus v_{ij}),(c_{ij}\oplus w_{ij}, d_{ij}\oplus x_{ij}))\\ ((a_{kl}\oplus u_{kl}, b_{kl}\oplus v_{kl}),(c_{kl}\oplus w_{kl}, d_{kl}\oplus x_{kl}))\end{pmatrix}$는
문장 $S_i \rightarrow S_j \rightarrow S_k$로 종속관계를 의미한다.

(증명)

〈정의3〉으로부터 $((a_{ij}\oplus u_{ij}, b_{ij}\oplus v_{ij}), (c_{ij}\oplus w_{ij}, d_{ij}\oplus x_{ij}))$는 문장 S_i로부터 문장 S_j로 종속관계이고 $(a_{kl}\oplus u_{kl}, b_{kl}\oplus v_{kl}), (c_{kl}\oplus w_{kl}, d_{kl}\oplus x_{kl}))$는 문장 S_j로부터 문장 S_k로의 종속관계를 나타낼 때 문장 S_i로부터 S_j를 거쳐서 S_k로 가는 Path가 2인 종속관계가 LCM에 의해서 구해진다는 것이다. a_{ij}, b_{ij}, u_{ij}, v_{ij} 는 첫번째 첨자의 초기해와 증가치를 나타내며 c_{ij}, d_{ij}, w_{ij}, x_{ij} 는 두번째 첨자의 초기해와 증가치이다. 같은 방법으로 S_j에서 S_k로 가는 종속관계에 대해서는 a_{kl}, b_{kl}, u_{kl}, v_{kl} 는 첫번째 첨자의 초기해와 증가치를 나타내며 c_{kl}, d_{kl}, w_{kl}, x_{kl} 는 두번째 첨자의 초기해와 증가치를 나타낸다. $S_i \rightarrow S_j \rightarrow S_k$로 가는 종속관계인 $*((a_{mn}\oplus u_{mn}, b_{mn}\oplus v_{mn}), (c_{mn}\oplus w_{mn}, d_{mn}\oplus x_{mn}))$에서, $S_i \rightarrow S_k$로 가는 추이종속관계의 초기치와 증가치인 a_{mn}과 u_{mn}은 $a_{mn}=u_{mn}=\mathrm{LCM}(v_{ij}, u_{kl})$임을 알 수 있다. 이는 가변이든 불변이든 종속관계는 증가치마다 종속관계가 성립한다.

〈 정리 1 〉
종속성 행렬 DMLCS(m,2,p) 의 임의의 한 원소
$\begin{pmatrix}((a_{ij}\oplus u_{ij}, b_{ij}\oplus v_{ij}),(c_{ij}\oplus w_{ij}, d_{ij}\oplus x_{ij}))\\ ((a_{kl}\oplus u_{kl}, b_{kl}\oplus v_{kl}),(c_{kl}\oplus w_{kl}, d_{kl}\oplus x_{kl}))\end{pmatrix}$는 path가 2 즉, 통로의 길이가 2인 종속관계를 나타내며
$\begin{pmatrix}((a_{ij}\oplus u_{ij}, b_{ij}\oplus v_{ij}),(c_{ij}\oplus w_{ij}, d_{ij}\oplus x_{ij}))\\ ((a_{kl}\oplus u_{kl}, b_{kl}\oplus v_{kl}),(c_{kl}\oplus w_{kl}, d_{kl}\oplus x_{kl}))\end{pmatrix}$
$= *((a_{mn}\oplus u_{mn}, b_{mn}\oplus v_{mn}), (c_{mn}\oplus w_{mn}, d_{mn}\oplus x_{mn}))$ 로 치환할 경우
$a_{mn} = u_{mn} = \mathrm{LCM}(v_{ij}, u_{kl})$,
$c_{mn} = w_{mn} = \mathrm{LCM}(x_{ij}, w)$
$b_{mn} = v_{ij} + b_{kl}(b_{kl} = 0$ 이면 $b_{kl} = v_{kl}) - 1$
$d_{mn} = x_{ij} + d_{kl}(d_{kl} = 0$ 이면 $d_{kl} = x_{kl}) - 1$
$v_{mn} = \mathrm{LCM}(v_{ij}, v_{kl})$
$x_{mn} = \mathrm{LCM}(x_{ij}, x_{kl})$가 된다.

두 번의 종속관계에 의한 추이 종속관계는 증가치와 증가치에 대한 계산으로 수행된다.

결국 한 종속관계의 증가치가 3이고 또 종속관계의 증가치가 4이면 (3,4), (6,8), (9,12)・・・・ 에서 종속관계가 성립되어 결국 두 증가치에 대한 LCM이 된다. 같은 방법으로 다른 증가치도 LCM으로 계산되어진다.

〈정리2〉
LCM에 의한 종속관계는 유한번 안에 종속관계가 없어진다.

(증명)

〈정리 1〉에서 추이 종속관계는 각각의 종속관계에 따른 증가치에 대한 LCM으로 계산되어진다. 두 종속관계의 증가치를 m_1, m_2 라고 하면 $m_3 = LCM(m_1, m_2)$가 되고 m_3 가 루프 첨자값의 최종치 n보다 크지 않다면 LCM을 계속 반복적으로 수행하면 된다.

그래서 LCM의 값을 m_1, m_2, m_3, ・・・・ , m_K 라 하면 m_1, m_2, m_3, ・・・・ , m_K는 1 보다 큰 정수가 되며 $1 \leqq m_1 \leqq m_2 \leqq m_3 \leqq$ ・・・・ $\leqq m_K$ 가 된다.

결국 m_1부터 m_K까지의 개수를 k라 하면 $m_K \geqq n$ 이 되는 k가 항상 존재한다는 것이다. 만약 k=n 이라하면 순차적으로 수행하는 결과이므로 n번 반복수행하면

자료종속성이 사라지고, 만약 k>n 이라면 k번 안에 모든 자료 종속성이 사라진다. 그러므로 LCM에 의한 종속관계는 유한번 안에 사라진다.

〈 정의 4 〉
Path가 3인 경우 (3이상)
종속성 행렬 DMLCS(m,3,p)의 임의의 한 원소는

$$\begin{pmatrix} ((a_{ij}\oplus u_{ij}, b_{ij}\oplus v_{ij}),(c_{ij}\oplus w_{ij}, d_{ij}\oplus x_{ij})) \\ ((a_{kl}\oplus u_{kl}, b_{kl}\oplus v_{kl}),(c_{kl}\oplus w_{kl}, d_{kl}\oplus x_{kl})) \\ ((a_{uv}\oplus u_{uv}, b_{uv}\oplus v_{uv}),(c_{uv}\oplus w_{uv}, d_{uv}\oplus x_{uv})) \end{pmatrix}$$

$$= \begin{pmatrix} *((a_{mn}\oplus u_{mn}, b_{mn}\oplus v_{mn}),(c_{mn}\oplus w_{mn}, d_{mn}\oplus x_{mn})) \\ ((a_{uv}\oplus u_{uv}, b_{uv}\oplus v_{uv}),(c_{uv}\oplus w_{uv}, d_{uv}\oplus x_{uv})) \end{pmatrix}$$

최소공배수 LCM은 LCM (v_{mn}, u_{uv}) = u'_{uv},
LCM (x_{mn}, w_{uv}) = w'_{uv} 로 나타낸다.

〈정의 5〉
Path가 1 인 경우 〈정의1〉에서 정의된 변수들을 이용하면 DOALL문은 다음과 같다.
FOR 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
```

여기서 U_{ij}와 W_{ij}가 1 인 경우는 uniform인 경우에 적용되며, 그렇지 않은 경우는 non-uniform에 적용된다.

〈정의 6〉
Path가 2 인 경우 〈정의1〉,〈정의2〉,〈정의3〉에서 정의된 변수들을 이용하면 DOALL문은 다음과 같다.
FOR DMLCS의 원소가 0이 아닌 모든 원소에 대하여

```
DOALL K=1, N
  IF Ukl 와 Wkl 이 1 THEN
  DOALL L=dkl, N
          Sj
    ENDDOALL
   ENDDOALL
  ELSE IF mU'kl+bij + nVij  THEN
    DOALL K=bij, M, Vij
     DOALL L=W'kl, N, W'kl
          Sj
     ENDDOALL
    ENDDOALL
   ELSE IF mW'kl+dij + nXij THEN
    DOALL K=U'kl, M, U'kl
     DOALL L=dij, N, Xij
          Sj
     ENDDOALL
    ENDDOALL
   ELSE
    DOALL K=U'kl, M, U'kl
     DOALL L=W'kl, N, W'kl
           Sj
     ENDDOALL
    ENDDOALL
 ENDDOALL
```

여기서 U_{kl}와 W_{kl}가 1 인 경우는 uniform인 경우에 적용되며, 그렇지 않은 경우는 non-uniform에 적용된다.

〈정의 7〉
Path가 3 이상인 경우 〈정의1〉,〈정의2〉,〈정의3〉,〈정의4〉에서 정의된 변수들을 이용하면 DOALL문은 다음과 같다.
FOR DMLCS의 원소가 0이 아닌 모든 원소에 대하여

```
DOALL K=1, N
  IF Uuv 와 Wuv 가 1 THEN
    DOALL K=bkl + buv - 1, M
      DOALL L=dkl + duv - 1, N
        Sv
      ENDDOALL
     ENDDOALL
    ELSE
      DOALL K=U'uv, M, U'uv
        DOALL L=W'uv, N, W'uv
              Sv
        ENDDOALL
       ENDDOALL
 ENDDOALL
```

여기서 U_{uv}와 W_{uv}가 1 인 경우는 uniform인 경우에 적용되며, 그렇지 않은 경우는 non-uniform에 적용된다.

이제 불변 종속거리 이거나 가변 종속거리를 갖는 루프에서의 종속성 제거 알고리즘을 도입한다. 일반적으로 불변종속 거리이거나 가변 종속 거리를 갖는 루프의 일반문장은 (그림 4)와 같다.

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

S_2 : $a_2(a_2I+b_2,\ e_2J+f_2)$

= $a_1(c_1I+d_1,\ g_1J+h_1)$ + ·····

··· ···

S_{i-1} : $a_{i-1}(a_{i-1}I+b_{i-1},\ e_{i-1}J+f_{i-1})$

= $a_{i-2}(c_{i-2}I+d_{i-2},\ g_{i-2}J+h_{i-2})$ + ·····

S_i : $a_i(a_iI+b_i,\ e_iJ+f_i)$

= $a_{i-1}(c_{i-1}I+d_{i-1},\ g_{i-1}J+h_{i-1})$ + ·····

S_{i+1} : $a_{i+1}(a_{i+1}I+b_{i+1},\ e_{i+1}J+f_{i+1})$

= $a_i(c_iI+d_i,\ g_iJ+h_i)$ + ····

··· ···

S_n : $a_n(a_nI+b_n,\ e_nJ+f_n)$

= $a_{n-1}(c_{n-1}I+d_{n-1},\ g_{n-1}J+h_{n-1})$ + ····

단, $a_i(1\le i\le n)$는 다른 변수일수도 있다.

(그림 4) 종속성을 갖는 루프의 일반문장
(Fig. 4) Statement of loop with a dependence

임의의 $i(1\le i\le n)$에 대하여 문장 S_i와 S_{i+1} 사이에 자료 종속이 존재한다면 기존의 방법들은 자료 종속성 관계를 알아본 다음 주로 분할하는 방법에 의해 병렬 코드들을 만들어 낸다. 본 연구에서는 두 문장 사이에 자료 종속성이 존재한는 경우 분할을 하지 않고 문장들을 수행시키는 방법에 의해서 자료 종속성을 제거하는 방법이다. 두 문장 S_i와 S_{i+1}사이에 자료 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 자료 종속성이 존재하면 병렬처리가 불가능하다. 이를 해결하기 위해서 두 번째로 문장 S_{i+1}만을 다시한번 수행시키면 문장 S_{i+1}의 a_{i+1} 값도 올바르게 수행된다. 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 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.

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 자료 종속 관계는 사라지게 될 것이다. 이를 구현하는 가장 간단한 방법은 통로의 길이 만큼 모든 문장들을 반복 수행하면 되지만 그런 경우 프로세서의 갯수가 매우 많이 필요하게 된다. 프로세서의 갯수를 가장 줄일 수 있는 방법은 정확하게 어떤 문장과 어떤 문장 사이에 자료 종속 관계가 있는지를 아는 것이다. 이를 위해서 최대공약수와 최소공배수를 이용하면 해결할 수 있다. 이와 같은 방법이 제안된 알고리즘이며 어떠한 자료종속이 존재하는 순차 루프에 대해서 본 알고리즘을 수행하면 그 결과는 가장 많은 병렬처리가 가능한 병렬코드로 변환된다.

3.1 Algorithm

① Path가 1인 DMLCS를 만든다음 DMLCS의 구성 원소중에서 첨자값이 같은 것이 있을 경우에는 다른 이름(rename)으로 대체한다.

② 중첩 loop의 변수를 각각 I와 J라하고 최대치를 M과 N이라 할 경우 $(a_{ij}\oplus U_{ij}) > M$ 또는 $(C_{ij}\oplus W_{ij}) > N$ 이 되는 원소와 $b_{ij} > M$ 이거나 $d_{ij} > M$ 인 원소는 0으로치환하며 0이 아닌 모든 원소에 대하여 〈정의 5〉와 같이 DOALL을 적용한다.

③ 모든 원소가 0인 경우에는 GOTO 7 그렇지 않은 경우에는 Path를 1 증가시키고 다음으로 간다.

④ LCM을 이용하여 남아있는 종속관계를 찾아내고, DMLCS의 원소를 변형하여 LCM의 값이 중첩 루프의 최종치(M 또는 N)보다크면 그 원소는 0으로 치환하고 0이 아닌 모든 원소에 대하여 〈정의 6〉 또는 〈정의 7〉과 같이 DOALL을 적용한다.

⑤ DMLCS의 원소 중 $mU'_{K1} * b_{ij} + nV_{ij}$ 또는 $mW'_{K1} * d_{ij}+nX_{ij}$ 인 원소와 $a_{mn}>b_{mn}$ 인 원소는 0으로 치환한다.

⑥ GOTO 3

⑦ 변형된 문장을 제외한 모든 문장에 대하여 각각 DOALL을 수행한다

4. 성능 평가

이 장에서는 제시한 알고리즘에 대한 성능 평가를 한다.

4.1 불변 종속거리에 대한 성능평가

불변(uniform)종속 거리에 대해서는 지금까지 가장 뛰어난 방법으로 알려진 Polychronopoulos[19,20]의 Cycle Shrinking 방법과 Shang&Fortes방법[23], tiling[26], interchanging[4], skewing[4], 혼합된 unimodular[22], Hollander[8], Chen& Wang [7]방법과 본 논문의 알고리즘을 비교하고자 한다. 비교 분석을 위하여 기존의 논문에서 가장 많이 제시한 예(Figure 6a in [9], Figure 1 in [13])를 가지고 하였다. 이 예제는 (그림 5) 예제이다.

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

(그림 5) 불변거리에 대한 예
(Fig. 5) Example of uniform distance

이 예제를 Cycle Shrinking 방법, Shang& Fortes 방법, tiling, interchanging, skewing, 혼합된 unimodular, Hollander, Chen&Wang방법과 본 논문의 알고리즘과 비교하고자 한다. 이 예는 사이클을 갖고 자료 종속 거리는 min{3,2} 과 min{5,4}인 2차인 루프이다. 본 연구에서 제시한 방법을 적용한 병렬 코드는 (그림 6)이다.

```
DOALL K=1,2
 IF K=1 THEN
        DOALL I=3,N1-3+1
          DOALLJ=5,N2-5+1
            A(I,J)=B(I-3,J-5)
          ENDDOALL
      ENDDOALL
 ELSE
        DOALL I=3,N1-4+1
          DOALL J=5,N2-6+1
              B(I,J)=A(I-2,J-4)
          ENDDOALL
        ENDDOALL
ENDDOALL
   DOALL K=1,2
    IF K=1 THEN
          DOALL I=6,N1
            DOALL J=10,N2
              A(I,J)=B(I-3,J-5)
            ENDDOALL
          ENDDOALL
        ELSE
          DOALL I=5,N1
            DOALL J=9,N2
              B(I,J)=A(I-2,J-4)
            ENDDOALL
          ENDDOALL
ENDDOALL
      :
```

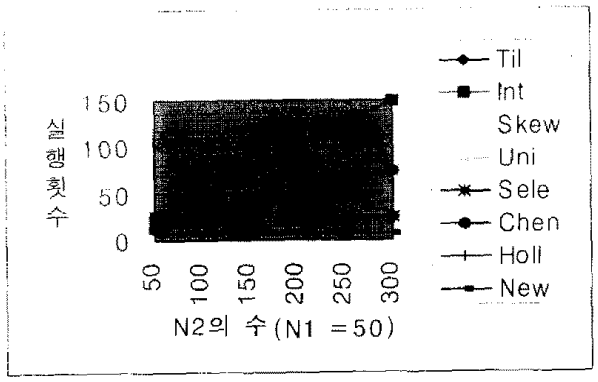
(그림 6) 종속사이클을 갖는 중첩루프
(Fig. 6) Nested loop with a dependence cycle

분석을 위해서 병렬 코드의 실행 수를 계산해 보면 다음과 같다. Tiling, interchanging와 selective shrinking의 병렬코드는 $\lambda=(N1-2)/2$번만에 수행 할 수 있고, skewing, 통합된 unimodular와 Chen& Wang은 $\lambda=\lceil(N2-4)/4\rceil$번 수행한다. Shang& Fortes방법이 적용되면 병렬코드는 $\lambda = 2 * \lceil (N2-4)/4 \rceil$ 번만에 수행할 수 있다. 하지만 본 논문에서 제시된 방법을 적용하면 $\lambda=2+\lceil N1/10\rceil$번만에 실행된다.

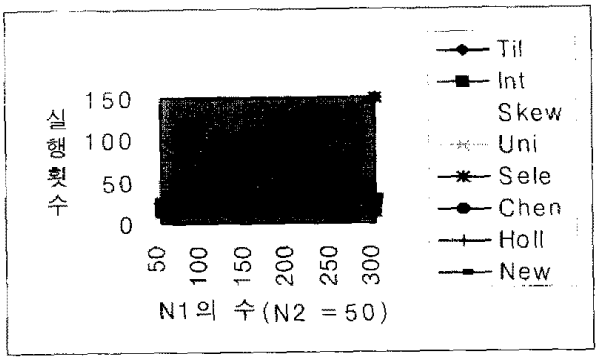
실행 수에 대한 성능평가 그래프를 (그림 7)에서 보여준다.

(그림 7)에서 N1=50일 때 N2의 값을 50부터 300까지 50씩 증가시켰으며 이에 따르는 수행 횟수를 비교하였다. N1의 값을 50으로 고정하면 tiling 방법, cycle shrinking방법, Hollander 방법과 본 연구의 방법은 N2값이 증가함에 따라 실행 횟수가 증가하지 않았으며 interchanging, skewing, Chen&Wang 방법, Shang&Fortes 방법, 통합된 unimodular 방법들은 기울기에는 차이가 있지만 거의 선형적으로 증가됨을 알 수 있다. 같은 경우로 N1 의 값을 100으로 고정한 경우에도 결과는 같았다.

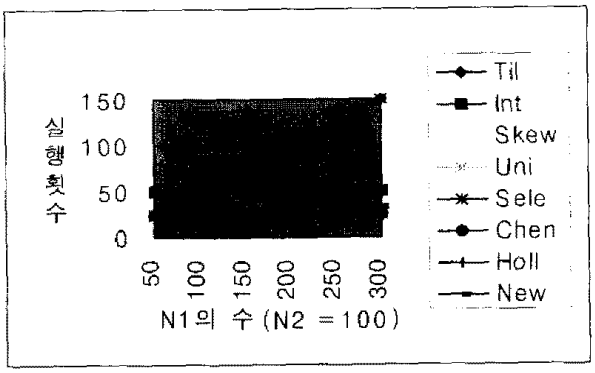
이번에는 N2의 값을 50으로 고정하고 N1의 값을 50부터 300까지 50씩 증가시켰더니 interchanging, skewing, Chen&Wang 방법, Shang&Fortes 방법, 통합된 unimodular 방법들은 N2값이 증가함에 따라 실행 횟수가 증가하지 않았으며 tiling 방법, cycle shrinking방법, Hollander 방법과 본 연구의 방법은 N2값이 증가함에 따라 기울기에는 차이가 있지만 거의



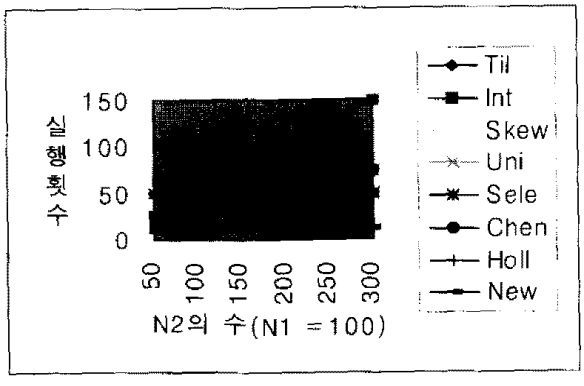
(a)



(b)



(c)



(d)

(그림 7) N1(N2)에 따른 실행 수 비교
(Fig. 7) The number of execution for N1(N2)

선형적으로 증가됨을 알 수 있다. 같은 경우로 N2 의 값을 100으로 고정한 경우에도 결과는 같았다.

이상에서 본바와 같이 이제까지 제시된 방법들과 본 연구에서 제시한 방법은 N1이나 N2중 하나에 따라서 실행 횟수가 증가됨을 알 수 있다. 그렇다면 기울기의 차이가 그 방법의 효율을 나타냄을 알수있고 기울기가 가장 낮은 방법은 본 연구에서 제시된 방법임을 알 수 있다.

4.2 가변 종속거리에 대한 성능평가

가변(non-uniform)종속거리인 경우에 대해서는 최근에 발표된 DCH[29]방법 및 IDCH[21]방법과 본 논문의 알고리즘을 비교, 분석하고자 한다. 합당한 비교 분석을 위해서 기존의 논문[21,29]에서 제시한 예즉 (그림 8) 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 8) 가변거리에 대한 예
(Fig. 8) Example of non-unifiorm

이 예제를 DCH[29]방법 및 IDCH[21]방법과 본 논문의 알고리즘과 비교하고자 한다.

```
doacross I = 1, N1
 shared integer J[N2]
   doserial J[I] = 1, N2
     if(I > 1) then
       while (J[I-1] < J[I]+2)
           do no-op;
S1:      A(2*I+J+1,I+J+3) = ···
S2:                  ··· = A(2*j+3, I+1)
       enddo
       J[I] = N1 + 2
enddo
```

(그림 9) DCH 방법에 의한 병렬코드
(Fig. 9) The parallel code of DCH algorithm

DCH방법과 IDCH방법에 의한 병렬코드는 각각 (그림 9)와 (그림 10)이며, 본 연구에서 제시한 방법을 적용한 병렬코드는 (그림 11)이다. 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 $\lambda=1+\lceil Min(N1,N2)/4\rceil$ 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 400까지 증가시키는 실행수에 대한 성능평가 그래프와, N2의 값을 고정하고 N1의 값을 50부터 400까지 증가시키는 실행수에 대한 성능평가 그래프는 (그림 12)와 같다.

```
Tile num Tn = ⌈N1/dmin⌉

DOserial K = 1, Tn
  DOparallel I = (k-1) * dmin + 1, min(K*dmin, N1)
    DOparallel J = 1, N2
      A(2*I+J+1, I+J+3)= ···;
                  ··· = A(2*J+3, I+1);
    ENDDOparall
  ENDDOparall
 ENDDOserial
```

(그림 10) IDCH 방법에 의한 병렬코드
(Fig. 10) The parallel code of IDCH algorithm

```
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 11) 본 연구에서 제시한 방법을 적용한 병렬코드
(Fig. 11) The parallel code of proposed algorithm in this paper

(그림 12)의 (a)를 보면 N1=50일 때 N2의 값을 50부터 400까지 50씩 증가 시켰으며 이에 따르는 수행 횟수를 비교 하였다. N1의 값이 50으로 고정된 (a)의 경우 DCH방법과 본 논문의 방법은 N2의 값에 따라 거의 변하지않으며 IDCH방법은 거의 선형적으로 증가됨을 알 수 있다. 또한 (b), (c)와 같이 N1의 값을 증가시킬 경우 DCH나 IDCH방법 보다 본 논문의 방법이 몇 십배의 속도가 향상되는 효과가 있음을 도표를 통하여 확인 할 수 있다.

같은 방법으로 N2의 값을 고정하고 N1의 값을 변경시키는 (d), (e), (f)의 경우에도 본 논문에서 제안된 방법이 몇 십배의 속도가 향상됨을 확인 할 수 있다.

그러므로 본 논문의 알고리즘이 N1이나 N2값에 관계없이 항상 좋은 방법이며, 특히 중첩루프의 첨자 변수의 값이 큰 경우에 더욱 성능이 우수함을 알 수 있다.

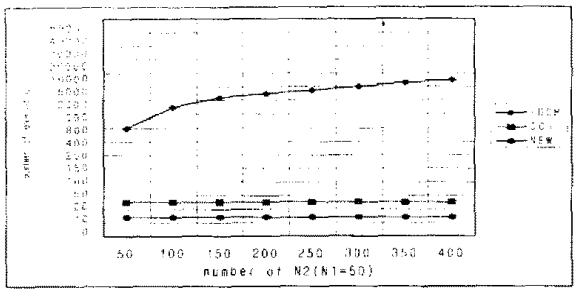
5. 결 론

본 논문에서는 병렬처리 시스템에 적용할 수 있는 새로운 변환 기법을 제시하였다. 기존의 방법들은 병렬성을 추출하기 위해서 자료 종속성 검사를 하고 자료 종속 관계가 존재하면 자료 종속을 최소화 시키면서 분할하는 방법에 의해 병렬 코드들을 추출하였다. 그러나 본 연구에서는 LCM을 이용하여 각각의 문장들을 수행하면 자료 종속성이 점차적으로 줄어든다는 자료 종속성 제거 방법을 제안하고 이 방법에 대한 성능 평가를 실시하였다.

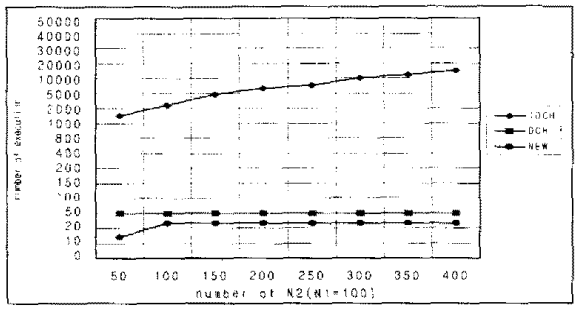
성능 평가에서는 본 연구에서 제시하는 방법이 매우 우수한 방법임을 보여주었다.

앞으로 이 알고리즘을 Unimodular 변환과 같은 선형 변환과 혼합하여 더욱 효과적인 실행 결과를 기대하는 문제 및, 불완전 중첩 루프에의 적용 방법, Benchmark 프로그램에 적용해서 실질적인 문제에 도입하는

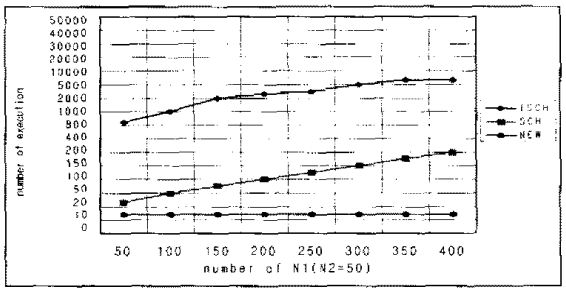
분해와 본 논문을 병렬 컴파일러에 적용하여 새로운 병렬처리 컴파일러를 만드는 것들은 향후에 실시할 것이다.



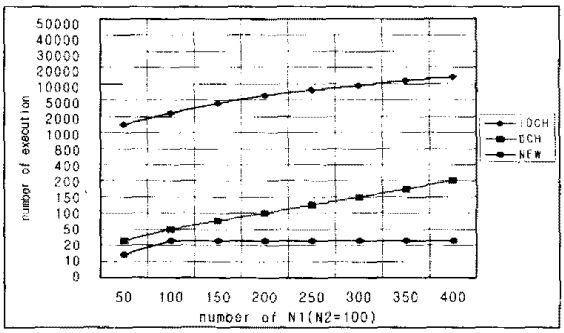
(a)



(b)



(c)



(d)

(그림 12) N1과 N2에 따른 실행수 비교
(Fig. 12) The number of execution for N1 and N2

참 고 문 헌

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

[2] Allen, F., M. Burke, P. Charles, R. Cytron, and J. Ferrante, "An overview of the PTRAN analysis system for multiprocessing", Journal of parallel and distributed computing, Vol. 5, No.5, Oct. 1988.

[3] Alliant Computer Systems Corp., "Alliant FX /Series Architecture Manual", Acton, MA, Aug. 1986.

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

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

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

[7] Dowling, M. L., "Optimal code parallelization using unimodular transformations", Parallel Computing 16, 1990.

[8] 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.

[9] Houck, C., G. Agha, "HAL:A High-Level Actor Language and Its Distributed Implementation", Proc. of Intl. Conf. on Parallel Processing, pp.158-165, 1992.

[10] Ju., J. and V. Chaudhary, "Unique Sets Orinted Partitioning of Nested Loops with Non-uniform Dependencies", 1996.

[11] Kong, X., 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.

[12] Kuck, D. J., A. H. Sameh, R. Cytron, A. V. Veidenbaum et al., "The effect of program restructuring, algorithm changes, and architecture choice on program performance", Proc. Int. Conf. Parallel Processing '84, pp.129-138, 1984.

[13] Li, j. and M. Wolfe, "Defining, Analyzing, and Transforming Program Constructs", IEEE Parallel & Distributed Technology, pp.32-39, 1994.

[14] Li, Z., 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.

[15] Malony, A., J. Larson, D. Read, "Tracing Application Program Execution on the Cray X_MP and Cray-2", CSRD Report No. 985, CSoptimizations for supercomputers." *Comm. ACM*, Vol.29, No.12, Dec. 1986.

[19] Polychronopoulos, C. D., "More on Advanced Loop Optimization", CSRD Report *No. 667, CSRD at Univ. of Illinois, Oct. 1987.*

[20] Polychronopoulos, C. D., "Compiler optimizations for enhancing parallelism and their impact on architecture design", *IEEE Trans. on Comp.*, Vol.37, No.8, Aug., 1988.

[21] Punyamurtula, V. Chaudhary, J. Ju, and S. Roy, "Compile time Partitioning of Nested Loop Iteration space with Non-uniform Dependencies", In Journal of Parallel Algorithm and Architecture, 1996.

[22] Ramanujam, J., "Beyond unimodular RD Univ. of Illinois at Urbana-Champaign, Nov. 1990.

[16] Midkiff, S. P., D. Padua, "Compiler Algorithms for Synchronization", IEEE Tran. on Computer Vol.c-36, No.12, pp.1485-1495, Dec. 1987.

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

[18] Padua, D. A. and M. Wolfe, "Advanced compiler transformatjuons", The Journal of Supercomputing, 9, pp.365-389, 1995.

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

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

[25] Tzen, T. H., and L. M. Ni, "Dependence Uniformization : A Loop Parallelzation Technique", IEEE Trans. on Parallel and Distributed Systems, Vol.4, No.5, May, 1993.

[26] Wolfe, M. J., "More Iteration Space Tiling", Oregon Graduate Center 19600 NW won Newmann Drive Beaverton, OR 97006.

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

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

[29] Zaafrani, A., and M. R. Ito, "Parallel region execution of loops with irregular dependencies", IEEE, 1994 International Conference on Parallel Processing, 1994.

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

송 월 봉

1974년 숭실대학교 공학사
1982년 한양대학교 전자계산학(공학석사)
1995년~현재 순천향대학교 컴퓨터학부 박사과정
1978년~현재 인천전문대학 전자계산과 재직
관심분야 : 병렬처리, 컴파일러, 알고리즘

박 두 순

1981년 고려대학교 수학과(학사)
1983년 충남대학교 대학원 계산통계학과(석사)
1988년 고려대학교 대학원(전산학 전공) 졸업(박사)
1992년~1993년 미국 Univ. of Illinois at Urbana Champaign CSRD 객원교수
1985년~현재 순천향대학교 컴퓨터학부 교수
관심분야 : 병렬처리컴파일러, 계산이론, 프로그래밍언어론등.