불변 및 가변 루프구조의 병렬성 제고를 위한 프로그램 재구조화

송월봉*

요약

병렬 화 컴파일러를 구현하기 위하여 반드시 필요한 병렬 성 추출방법 중 중첩 루프를 재구성하는 Unimodular 변환 방법과, 계층 기억장치를 효율적으로 사용하기 위한 non-unimodular 변환 방법에 대하여 각각의 알고리즘을 비교 분석 하였으며 이를 토대로 병렬 성을 제고하기 위한 통합된 방법을 보였다. 앞으로 이 결과는 perfect benchmark와 같은 benchmark 프로그램에 적용해서 실질적인 문제에 도입하는 부분과 일반화 될 수 있는 병렬 화 컴파일러를 제시할 계획이다.

Program Restructuring for Promotion Parallelism of the Uniform & Non-uniform Loop

Worl-Bong Song*

ABSTRACT

The general methods for the extracting parallelism in order to parallel processing effectively are unimodular transformation which restructure a nested loop and non-unimodular transformation for using effectively hierarchy memory device. This paper compare and analyze this two algorithm. And also propose mixed method in order to raising the parallelism. From now on, this result will be applied to the bench mark program like a perfect benchmark program and introduced in the real problem and I plan to propose a general parallel compiler.

Key Words : unimodular, non-unimodular, parallelism, program restructuring

* 인천대학교 컴퓨터공학부(✉wbsong@incheon.ac.kr)
· 제1저자(First Author) : 송월봉 · 교신저자(Correspondent Author) : 송월봉
· 접수일(2012년 1월 31일), 수정일(1차 : 2012년 2월 30일), 게재확정일(2012년 3월 2일)

Ⅰ. 서 론

자료종속성 제거 방법의 경우 기존의 방법들은 루프를 펼쳐서 자료종속성 관계를 이용하여 펼쳐진 문장들을 분할하는 방법을 이용한다. 그러나 본 연구[1]에서는 펼쳐진 루프로부터 문장들을 분할하지 않고 루프 문장들을 수행하는 방법에 의해 자료종속성을 제거하는 방법을 제시하였다. 이 방법은 MIMD(Multiple Instruction Stream Multiple Data Stream)기계이어야 하며 첫 번째 모든 문장들을 병렬 수행하여 첫 번째 자료 종속성을 제거한다. 같은 방법으로 통로의 길이가 2인 자료종속성을 갖는 문장들을 LCM(Least Common Multiple)을 이용하여 구한다음 자료 종속성을 제거하여 병렬 수행하는 방법이다. 본 연구에서는 이를 실질적으로 구현 하기위한 전 단계로서, 우선적으로 Parafrase II 병렬 화 컴파일러의 원시 프로그램 중에서 대표적으로 Shang&Fortes의 방법[2], Hollander방법[3]등을 보이고 특히, unimodular변환방법[4,5], non-unimodular변환방법[4,5] 등을 분석하였다. 이를 토대로 이들의 장단점을 파악하고 제시된 방법들보다도 매우 효율적인 통합된 방법을 보이고자 한다. 이는 향후 이를 이용하여 종속거리에 무관하게 모두다 컴파일 시간에 병렬 성 검사를 하고 병렬코드를 성해내서 병렬처리 시스템의 실행시간을 단축시킬 수 있는 병렬 화 컴파일러를 개발 구현하는데 토대가 될 것이다.

Ⅱ. 관련연구

2.1 Shang & Fortes

기존의 병렬구간을 최대화하기 위한 프로그램 재구조화 방법들에 대하여 설명하기 위해서 <그림 1>의 예를 가지고 설명한다.

```
DO I = 2, N1
DO J = 4, N2
   A(I, J) = B(I-2, J-4)
   B(I, J) = A(I-1, J-3)
ENDDO
ENDDO
```

그림 1. 중첩루프 예
Fig. 1. A sample of nested loop

Shang & Fortes방법은 Hollander의 방법과 마찬가지로 병렬 구간을 최대화시키기 위한 방법이다. Shang & Fortes[2]알고리즘을 <그림 1>에 적용한 결과는 <그림 2>와 같다.

```
DO J=4, N1,3
 DOALL J1=J,J+1,J+2,J+3
  DOALL I=2,N1
   A(I,J1)=B(I-1,J1-4)
   B(I,J1)=A(I-1,J1-3)
  ENDDOALL
 ENDDOALL
ENDDO
```

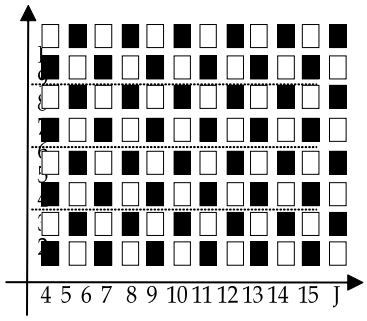
그림 2. Shang&Fortes 방법
Fig. 2. Shang & Fortes Algorithm

2.2 Hollander

이 방법[5]은 정규화 된 프로그램만 가능하고 알고리즘을 이용하여 삼각 행렬로 종속 행렬 D를 변환한다. 변환된 행렬 D′의 주 대각 원소들이 각 루프의 병렬의 정도를 나타낸다. <그림 3>은 <그림 1>을 이용하여 병렬 화 된 프로그램(a)과 반복 공간(b)을 나타낸다. 반복 공간에서 각 블록 의 검정 부분과 하얀 부분은 병렬로 수행한다.

```
DOALL II = 1, 1
DOALL JJ = 1, 2
 DO I = II, N1-3+1
  Y = (I-II) / 1
  K = 1 + (jj - 1 + Y) MOD 2
  DO J = K, N2-5+1, 2
 A(2+(I-1), 4+(J-1)) = B(2+(I-1)-2, 4+(J-1)-4)
 B(2+(I-1), 4+(J-1)) = A(2+(I-1)-1, 4+(J-1)-3)
  ENDDO
 ENDDO
ENDDOALL
ENDDOALL
```

<a> 재구조화 코드



<b> 재구조화 변환의 반복 공간

그림 3. Hollender 알고리즘
Fig. 3. Hollender Algorithm

III. unimodular, Non-unimodular

unimodular 재구조화는 unimodular 행렬(행렬식 ±1)을 사용하여 체계적인 방법으로 중첩 루프를 재구성하는 방법이다[4]. 이러한 방법들은 모든 중첩 루프의 정확한 한계 값을 계산하게 한다. 최근에는 통신을 최소화하거나 계층 기억장치를 효율적으로 사용하기 위하여 non-unimodular 변환을 사용한다[5]. 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[6]. 이 matrix의 주 대각 원소의 값이 각 반복 공간의 증가 값이다.

unimodular 행렬로 표현할 수 있는 기본적인 변환은 loop interchanging, loop skewing, loop reversal이고, 병렬 화를 최대화하기 위하여 위 세 가지를 통합하는 방법을 사용한다.

3.1 unimodular 변환

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[4].

unimodular matrix로 표현할 수 있는 기본적인 변환은 loop interchnging, loop skewing, loop reversal이다.

3.1.1 루프 Interchanging

2개의 열(행)을 교환한다.

$$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \xrightarrow{interchanging} \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

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

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

예를 들면, 이중 루프에서는 종속 관계 (=,<)을 (<,=)로 변환하고, 삼중루프는 IJK 루프를 $IJK\rightarrow IKJ\rightarrow KIJ\rightarrow KJI$ 나 $IJK\rightarrow JIK\rightarrow JKI\rightarrow KJI$ 로 변환한다.

3.1.2 루프 Skewing

주 대각 원소의 위(upper skewing 행렬)나 아래(lower skewing 행렬)에 0 이외의 정수 값(f)으로 대체한다. f는 skewing factor이다.

$$\begin{bmatrix}1&0\\0&1\end{bmatrix}\ \underset{\rightarrow}{skewing}\ \begin{bmatrix}1&f\\0&1\end{bmatrix},\begin{bmatrix}1&0\\f&1\end{bmatrix}$$

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

$$skew(k,f,m)$$
$$(d1,d2,\cdots,dk,\cdots,dm\cdots,dn)\rightarrow(d1,d2,\cdots,dk,\cdots,dm+fdk\cdots,dn)$$

3.1.3 루프 Reversal

주 대각 원소에 -1을 곱한다.

$$\begin{bmatrix}1&0\\0&1\end{bmatrix}\ \underset{\rightarrow}{reversal}\ \begin{bmatrix}-1&0\\0&1\end{bmatrix},\begin{bmatrix}1&0\\0&-1\end{bmatrix}$$

$$reverse$$
$$(d1,d2,\cdots,dk,\cdots,dn)\rightarrow(d1,d2,\cdots,-dK,\cdots,dn)$$

3.2 unimodular변환 방법을 이용한 재구조화

병렬 화를 높이기 위하여 중첩 루프를 unimodular 변환으로 재구성하기 위해서는 우선, Fourier-Motzkin 방법을 사용하여 루프 한계 값 Lk,1, Uk,1을 계산한다. <그림 4>는 이중 루프의 예제 프로그램과 반복 공간을 보여준다.

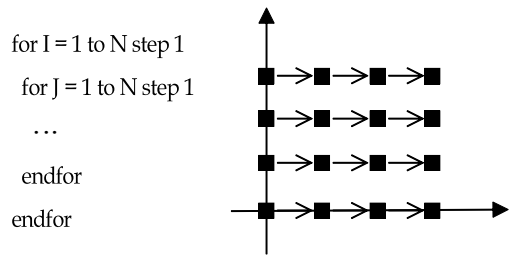


그림 4. 이중 루프 프로그램과 반복 공간
Fig. 4. Nested loop & Iteration space

행렬 부등식으로 표현하면,

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

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,

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

J의 최저 값과 최고 값을 계산하였다. J를 제거하면,

$$1 \le N$$
$$1 \le i-1$$
$$i-N \le N$$
$$i-N \le i-1$$

정리하여, $2 \le I \le 2N$으로 I의 최저값과 최고값을 계산하였다. <그림 5>는 unimodular 변환으로 변환된 루프 한계 값과 반복 공간을 보여준다.

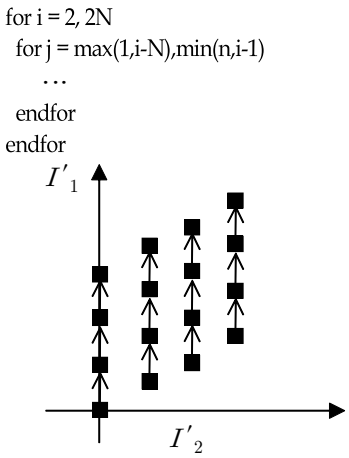


그림 5. unimodular 변환으로 변환된 프로그램과 반복 공간

Fig. 5. Transformed program into unimodular transformation & Iteration space

3.3 Non-unimodular 변환

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[4].

Hermite Normal Form은 다음과 같은 특성을 가진다.

- $lowertriangular;\ hij = 0 \quad for\ \ i < j.$
- $hii > 0 \quad for\ \ I = 1, \cdots, n.$
- $hij \le 0 \quad and\ \ |hij| < hij \quad for\ \ i > j$

3.4 Non-unimodular변환 방법을 이용한 재구조화

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 $\max(2-i, i-2N) \le J \le \min(2N-1, i-2)$으로 I와 J의 최저값과 최고값을 계산한다.

둘째, hole을 건너뛰기 위하여 루프 증가 값을 계산해야 한다. 루프 증가 값은 Hermite Normal Form을 사용하고, 이 행렬의 주 대각 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 6>은 non-unimodular 변환으로 변환된 루프 한계 값과 반복 공간을 보여준다.

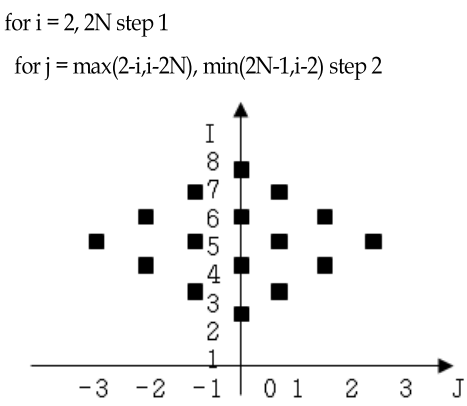


그림 6. Non-unimodular 변환으로 변환된 프로그램과 반복 공간

Fig. 6. Transformed program into non-unimodular transformation & Iteration space

Ⅳ. unimodular, non-unimodular 변환의 성능비교

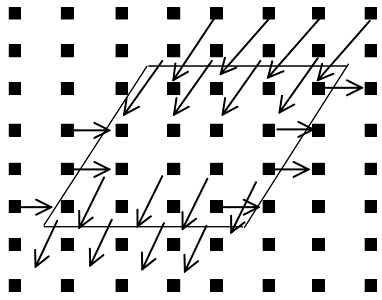
두 변환의 성능 비교를 위하여 Tiling을 이용한다. 반복 공간 tiling은 n-차원 반복 공간을 각 블록이 독립적으로 실행하는 n-차원 블록으로 분리된다. 각 블록 사이에는 종속성이 없어야 한다.

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.

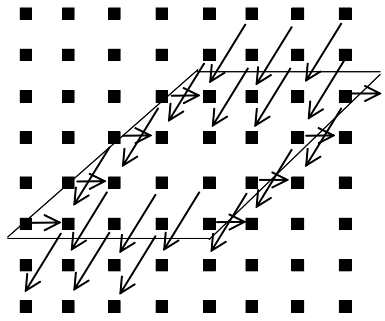
거리 행렬 $D = \begin{bmatrix} 0 & 2 & 1 \\ 1 & -1 & 0 \end{bmatrix}$ 일 때, 최소의 통신을 가지는 tiling 변환 T는 $T = \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}$ 이다. 정칙 행렬 T를 찾아야 하므로, $T = \begin{bmatrix} 1 & t_1 \\ t_2 & 1 \end{bmatrix}$ 은 $D+= TD = \begin{bmatrix} t_1 & 2 & -t_1 & 1 \\ 1 & 2t_2 & -1 & t_2 \end{bmatrix}$ 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 $(1 - t2t1 \neq 0)$. 따라서 두 번째 것은 불가능하고 $(1 - 2 \cdot 1_{/2} = 0)$, 해는 $t1 = 0,\ t2 = 1_{/2}$ 이다.

<그림 7>은 non-unimodular 행렬 $\begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix}$ 와 unimodular 행렬 $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ 의 tile 공간을 보여준다. <그림 7>에서 unimodular, non-unimodular tiling을 비교해 보면, <그림 7>. (b)의 추가된 화살표는 non-unimodular에는 없는 통신이 unimodular에 제고된 것을 표현한다. 즉, unimodular tiling에 의한 통신의 양이 non-unimodular tiling 보다 더 많은 것을 볼 수 있다.



(a) Non-unimodular transformations



(b) unimodular transformations

그림 7. unimodular, non-unimodular의 성능비교
Fig. 7. Performance test of un imodular & non-unimodular

Ⅴ. unimodular, Non-unimodular 통합변환

중첩 루프에서 data locality을 최대화하기 위하여 interchange, reversal, skewing와 tiling의 최적의 혼합 변환을 적용한다. 그리고, 항상 tiling이 가능한 것이 아니고, 중첩된 모든 루프가 fully permutable할 때만 가능하다. 여기에서는, 데이터 종속성 때문에 tiling 전에 skewing 하여 fully permutable로 변환 후 tiling을 적용한다.

<그림 8>은 예제 프로그램과 반복 공간이고, 거리 벡터 D는 (0, 1), (1, 0), (1, -1)이다. 우선, 중첩 루프의 안쪽

을 fully permutable로 만들기 위하여 skewing을 한다.

```
for I = 0 to 5 do
  for J = 0 to 6 do
    A(J+1) = A(J) + A(J+1) + A(J+2)
```

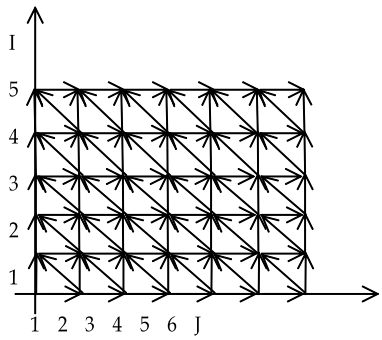


그림 8. 혼합 변환 예제 프로그램과 반복 공간
Fig. 8. A mixed transformation program & Iteration space

변환 행렬 $T = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ 으로 skewing하면, 거리 벡터 D'는 TD = {(0, 1), (1, 1), (1, 0)}으로 변한다. <그림 9>는 변환된 프로그램과 반복 공간을 보여준다.

```
for I = 0 to 5 do
  for J = I to I+6 do
    A(J-I+1) = A(J-I) + A(J-I+1) +A(J-I+2)
```

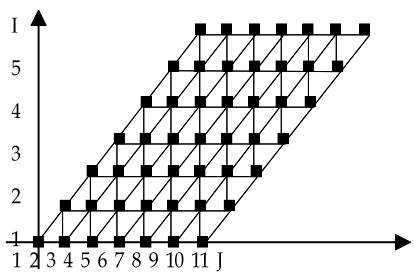


그림 9. Skewing 후 변환된 프로그램과 반복 공간
Fig. 9. A transformed program after Skewing & Iteration space

반복 공간이 non-rectangular로 변환 후, tiling을 적용한다. tiling된 프로그램과 반복 공간은 <그림 10>과 같다.

```
for JJ = 0 to 11 by 2 do
  for I =  to 5 do
    for J = max(I, JJ) to min(I+6, JJ+1) do
      A(J-I+1) = 1/3 * A(J-I) + A(J-I+1)
                 + A(J-I+2));
```

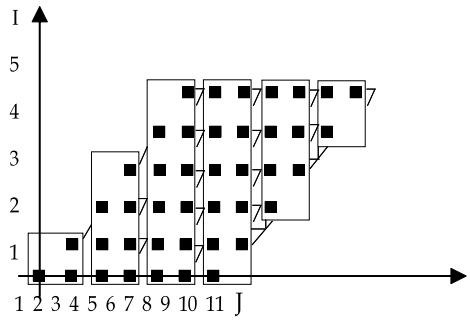


그림 10. Tiling후 변환된 프로그램과 반복 공간
Fig. 10. A transformed program after Tiling & Iteration space

Ⅵ. 결론

종속거리가 불변이든 가변이든 혹은 혼재되어 있는 모든 순환프로그램에서 컴파일 시간에 병렬 성 검사를 하고 병렬코드를 생성해내서 병렬처리 시스템의 실행시간을 단축시킬 수 있는 병렬 화 컴파일러를 개발 구현하기 위하여 반드시 필요한 병렬 성 추출방법 중 unimodular 행렬(행렬식±1)을 사용하여 체계적인 방법으로 중첩 루프를 재구성하는 Unimodular 재구조화 방법과, 통신을 최소화하거나 계층 기억장치를 효율적으로 사용하기 위한 non-unimodular 변환 방법에 대하여 각각의 알고리즘을 비교 분석하였다. 분석 결과 반

복 공간 Tiling을 이용한 두 변환의 성능 비교에서 unimodular tiling에 의한 통신의 양이 non-unimodular tiling 보다 더 많은 것을 확인 하였으며 이를 토대로 병렬 성을 제고하기 위하여 통합된 방법을 보였다.

앞으로 이 결과는 다중 명령어 흐름 다중 데이터 흐름 기계에 적용하기위한 새로운 병렬처리 컴파일러를 구현하는 과제가 남아 있으며, 향후 일반화 될 수 있는 병렬 화 컴파일러를 제시할 계획이다.

참고문헌

[1] Worl-Bong Song, “The Procedure Transformation Algorithm Using Dependency Elimination in Non-uniform”, kkits, vol. 6, No. 2, 2011, 4.

[2] Shang, W., and Fortes J., "On loop transformations for generalized cycle shrinking," POICPP. pp. 132- 141, 1991.

[3] 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

[4] J. Ramanujam, “Beyond unimodular transformatjuons”, The Journal of Supercomputing, 9, 365-389, 1995.

[5] 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

[6] Polychronopoulos, C.D., "More on advanced loop optimization," CSRD Report No. 667, Univ. of Illinois at Urbana-Champaign, Oct. 2003.

[7] Sang Il Park, Weol Seon Park, Sung-Dae Youn, "A Data Dependency Elimination Method in Multidimensional Index Loop Using a Dependency Function", 「ICIS ’0 1」 pp 53-57, 2001.10

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

[9] Maydan, D.E., J. Hennessy and M.S. Lam, "Effectiveness of Data Dependence Analysis," Int. J. Parallel Programming, Vol. 23, No. 1, pp 63-81, 1995

[10] Worl-Bong Song, “Extracting Parallelism in Nested Loops”, COMPSAC, IEEE, seoul, 1996. 8.

감사의 글

본 논문은 인천대학교 2011년도 자체연구비 지원에 의하여 연구되었음.

저자소개

송월봉(Worl-Bong Song)

1974년 숭실대학교 공학사

1982년 한양대학교 공학석사

1998년 순천향대학교 공학박사

2010년 ~ 현재 인천대학교 컴퓨터공학과 교수

※ 관심분야 : 병렬처리, 컴파일러, 알고리즘