영상정보를 이용한 병렬 프로그램내의 병행성 판별

박명철* · 하석운**

Check of Concurrency in Parallel Programs using Image Information

Myeong-chul Park* · Seok-wun Ha**

요 약

내포 병렬성을 포함하는 병렬 프로그램은 복잡한 수행양상을 가지며, 태스크들은 병행적으로 수행되는 경향이 있다. 이러한 병행성은 대부분의 오류을 유발하는 근본적인 원인이 된다. 본 논문은 병렬 프로그램 수행양상을 영상화하여 두 태스크간의 병행성을 판별할 수 있는 새로운 방법을 제안한다. 기존의 방법들은 전역적인 구조를 보이는데 제약이 있고 과다한 추상화로 인하여 수행양상의 직관성을 저해하는 경향이 있다. 본 논문에서 제안한 기법은 영상 가시화에 적합한 레이블링 기법을 제안하고, 제안한 레이블링 기법을 적용하여 산출된 각 태스크의 레이블 정보를 2차원 평면상에서 분할된 영역으로 표현한다. 이를 토대로 각 태스크의 순서화 관계를 식별할 수 있는 독립된 영상을 생성한다. 결과로 생성된 영상은 관련 태스크의 의미론적 분석을 간소화하고 전체 프로그램의 전역적 수행 구조의 개요를 사용자에게 효과적으로 제공한다.

ABSTRACT

A parallel program including a nested parallelism has a complex execution aspects and tasks are executed concurrently. This concurrency is a main cause raising most of errors. In this paper, a new method for checking concurrency between two tasks is proposed. The existing techniques for checking the concurrency have their limits to represent a global structure. A new labeling technique that appropriate for image visualization is proposed. To show the global structure by imaging of execution aspects through region partition on 2D plane. On the basis of it, each of the tasks that can distinguish the ordered relation create an independent image. Image information generated by the result simplifies semantic analysis of the related task, and provides an outline of a global execution aspects structure of the program to user effectively.

키워드

Concurrency, image Information, Labeling, Parallel Program, Program Visualization

Ⅰ. 서 론

내포 병렬성을 가진 병렬 프로그램은 복잡한 프로그램 수행의 성능과 신뢰성을 향상시킬 수 있다.[8] 그러나 이러한 병렬 프로그램은 그 수행양상의 복잡성과 태스크간의 비동기적 접근에 의해 다양한 오류를 유발하여 비결정적인 결과를 초래할 수 있다.[1] 특히 공유메모리를 사용하는 병렬프로그램의 경우에는 그 심각성이 한층 더 높다.[10,11] 전통적인 오류 중에 경합조건과 교착상태등은. 태스크간의 병행성 관계에서만 유발되기 때문에 병행성 여부를 판별하는 것은 병렬프로그램의 오류를 디버깅하기 위해 매우 중요하다. 이를 위한 다양한 시도 중에서 수

* 국립 경상대학교 컴퓨터과학과 박사수료
** 국립 경상대학교 컴퓨터과학과 교수

접수일자 : 2006. 7. 4

행양상을 가시화하여 사용자에게 제공하고 오류 탐지의 결과를 보여주는 시각화 기법이 사용되고 있다.[3,4,5,6,7] 그러나 프로그램의 병렬성 증가로 인한 수행 정보의 복잡도와 대량화로 인해 효과적인 시각화에 많은 어려움이 있으며, 이 문제점을 극복하기 위하여 수행정보를 부분적으로 수집하거나 특정 정보의 그룹을 단위 심볼로 대체하는 추상화 방법이 적용되고 있다.[5,6,7] 그러나 기존의 이러한 시각화 방법들은 표현 공간상의 제약으로 인하여 전체 프로그램의 전역적 수행구조를 보이지 못하므로 부분적인 시각화에 국한된다는 문제점을 가지고 있다.[3,4] 또한, 과다한 추상화로 인하여 본질적인 태스크의 수행정보를 사용자에게 효과적으로 제공하기 어렵다.[5,6,7] 2차원 평면상의 제약성을 극복하기 위한 방법으로 3차원적 시각화 방법들이 제안되었지만, 실제 공간상의 제약적인 문제점은 여전하며, 수행 양상에 대한 사용자의 식별력을 저해하는 결과를 초래하고 있다.[16] 병렬 프로그램에서 발생하는 오류의 대부분은 두 사건간의 병렬성 관계에서 유발된다.[1] 프로그램의 오류를 탐지하기 위해서는 두 사건이 병렬성 관계를 가지고 있는지를 식별할 수 있어야 한다.[8,9] 공유메모리를 사용하는 데스크들의 접근역사(access history)를 관리하여 병렬성을 식별하는 Task Labeling 기법들은 대부분 내포깊이와 병렬성의 증가에 따른 많은 메모리를 필요로 한다.[2,12,13,14,15]

본 논문에서는 내포깊이나 병렬성 증가에 독립적인 레이블링 기법을 제안하고, 2차원 평면상에서 병렬프로그램의 태스크 수행양상을 분할 영역으로 나타내며, 이를 영상화하여 병행성 여부를 판별하는 새로운 방법을 제시한다. 먼저, 레이블링은 Fork되는 시점에서 각 태스크를 식별할 수 있는 분할 영역좌표 생성된다. 생성된 좌표는 Join 시에 다시 통합되게 되는데 Fork이전의 태스크와 구분하기 위하여(Multi-Way: 그림1에서 TC, TG, TM의 관계) Join 횟수을 가지게 된다. 구체적인 레이블링 방법은 3장에서 소개한다. 두 번째 단계에서는 이전 단계의 레이블링 정보를 이용하여 2차원 평면상에 색상정보로 표현되게 된다. Fork와 Join사건을 식별하기 위하여 구분선을 만들고 각 구분선에 RGB 값을 다르게 하여 Join 레벨을 식별할 수 있게 한다. 이렇게 생성된 영상정보는 프로그램의 전역 구조의 개요를 사용자에게 제공할 수 있다. 그리고 생성된 영상정보의 경계 색상값을 통하여 서로 다른 두 태스크의 병행성 여부를 판별할 수 있다.

본 논문의 2장에서는 논문에서 제시하는 영상 시각화 방법에 대한 주요 아이디어에 대해 살펴보고, 3장에서는 구현 과정에 대해 기술한다. 4장에서는 시각화 결과인 영상을 통해 제안한 기법의 병렬성 판별 여부를 확인하고 기존 레이블링 방법과 비교한다. 그리고 5장에서는 결론과 향후 연구를 밝힌다.

Ⅱ. 연구배경

2차원 평면상에 표현하는 내포 병렬성을 가진 병렬프로그램의 대표적인 표현 방법으로는 그림 1에서와 같은 POEG(Partial Order Execution Graph) 형태를 일반적으로 사용하여 왔다.[2]

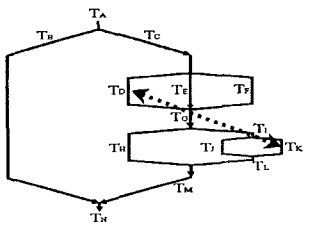


그림 1. POEG 구조상의 표현
Fig. 1. POEG(Partial Order Execution Graph)

POEG 형태의 표현 방법들은 수행의 부분적인 순서화를 표현하는 것으로는 적합하지만 전역적인 수행 양상을 보이지는 못하고 있다. 또한 복잡도가 높고 방대한 자료를 2차원에 표현하기에는 공간적인 부족함이 유발된다. 그림 1의 T_D 와 T_K 사건의 병행성 여부를 판별하기 위해서는 관련 레이블링 정보를 비교하여야 하는데, 이는 각 태스크마다 이전의 태스크 정보를 보존하여야 한다는 공간적 제약성을 가지게 된다. 그리고 3차원 공간상에 표현하는 방법들은[16] 공간적 제약성을 해소하여 전체적인 수행정보를 담고는 있지만, 과다한 추상화와 최적화로 인하여 수행정보의 본질적인 유효성을 잃어버리는 경향을 가진다.

본 논문에서 제안하는 방법은 영상 표현 기법을 이용하여 병렬성을 판별하여 오류탐지의 가능성을 탐지하는 특징을 가진다. 또한, 기존의 시각화 방법들에서 병렬성 분석을 위한 관련 태스크간의 관계를 시각적으로 인지하기에 힘들었던 제한점을 해소하는 장점을 가지고 있다.

본 논문에서 제시하는 영상 시각화 방법은 크게 3가지

특징을 가지도록 설계되었다. 먼저, 사용자가 2차원 평면상에서 태스크의 수행양상을 식별 가능하게 한다. 병렬프로그램의 대표적인 예인 OpenMP[10,11]과 같은 내포병렬성을 가지는 프로그램에서 내포성을 2차원 평면상에 중첩됨 없이 색상정보와 *Fork-Join*을 표시하는 기호를 사용하여 표현한다. 이를 위해 고안된 새로운 레이블링 기법은 2차원 평면의 방향축을 세로와 가로로 번갈아 사용함으로써, 전체적인 좌우상하 대칭성을 유지할 뿐만 아니라, 내포성에 대한 정보가 분할된 공간의 내부에 존재하게 함으로써 사용자로 하여금 하부구조에 대한 식별을 가능하게 한다. 두 번째, 전체 프로그램의 전역적인 구조의 개요를 제공한다. 병렬프로그램을 시각적으로 표시하는 대표적인 방법인 POEG의 문제점은 프로그램의 수행양상을 부분적으로만 표현한다는 것이다. 전역적인 수행구조를 표현하기 위하여 3차원 공간상에 각 양상을 그룹화하여 시각 복잡도를 줄이는데, 이는 오히려 표현정보를 모호하게 하는 결과를 초래한다. 2차원 평면상에 표현된 전역구조는 세부 구조에 접근하기 위한 고수준의 맵으로 사용할 수 있어야 하고 그룹화된 표시양상을 유추할 수 있는 시각적 정보를 제공하여야 한다. 본 논문은 이러한 문제를 2차원공간의 색상과 공간의 격리와 내포성을 적용하여 표현한다. 마지막으로 영상화하여 사용자에게 제공하고 영상정보만으로 병행성을 판별한다. 기존의 기법에 의한 레이블 정보는 사용자가 직관적으로 식별하기에 적합하지 않다. 사용자의 식별성과 상호작용성을 증대시키기 위하여 레이블 정보를 영상화하여 사용자에게 제공한다. 영상정보내의 각 영역과 색상에 레이블 정보의 의미를 부여하여 영상정보만으로 병렬성을 판별할 수 있는 방법을 제공한다.

Ⅲ. 제안 방법의 구현

본 논문에서 제안하는 병렬 프로그램의 수행양상에 관한 정보를 영상화하고 병행성을 판별하는 과정은 다음의 세 단계로 구성된다.

첫 단계에서는 내포 병렬성이 있는 태스크를 식별하기 위해 각 태스크에 레이블 정보를 생성한다. 본 논문에서는 기존의 레이블링 기법들을 사용하지 않고 영상 표현에 적합한 새로운 레이블링 기법을 제시하여 적용한다. 두 번째 단계에서는 제시한 레이블링 기법에 의해 생성된 레이블 정보를 직관적으로 판단할 수 있도록 2차원 평면상에서 분할된 고유 영역 정보로 변환한다. 세 번째 단계에서는 영상 정보만으로 병렬 프로그램의 수행양상을 탐지할 수 있도록 하기 위해 2차원 평면상에 표현된 레이블 영역 정보를 컬러 영상화한다.

3.1. Labelling

영상 시각화를 위한 첫 번째 단계는 태스크에 대한 레이블 정보를 생성하는 것이다. 2차원 평면상에서 분할되는 고유 영역들이 각 태스크에 대한 레이블 정보를 내포할 수 있도록 하는 것이 본 논문에서 새롭게 제시하는 레이블링 기법의 기본적인 개념이다. 영역 분할과 관련된 표현 인자에 대한 의미는 표 1과 같다.

표 1. 레이블 정보의 인자
Table 1. Parameter of label information

인 자	의 미
J_t	현재 태스크까지의 조인횟수
N_{dep}	현재 태스크의 내포깊이
(Xs,Xe)	현재 태스크의 *x*축 시작, 끝좌표
(Ys,Ye)	현재 태스크의 *y*축 시작, 끝좌표

위의 표현 인자에 의해 형성되는 레이블 표현은 아래와 같다.

$$[J_t, N_{dep}, (Xs, Xe), (Ys, Ye)]$$

여기서 J_t 와 N_{dep} 는 해당 *Fork-Join* 연산시에 관련 태스크에 의해 생성 될 수 있는 정보이다. J_t 는 내포 병렬구조에서 *Multi-way* 형태의 순서적인 태스크를 식별하기 위한 용도로 사용되고, N_{dep} 는 영역의 경계값을 나타내며 내포성에 관한 색상과 조인 연산과정에서 색상정보로 이용된다. 레이블링을 위한 영역의 초기값은 (*MinInt, MaxInt*)로 표현되는데, *MinInt* 는 1이고 *MaxInt*는 시스템에서 표현 가능한 양의 정수의 최대값이다. *n* bit의 워드를 가지는 시스템에서 표현가능한 양의 정수는 $2^{n-1}-1$ 이 된다. 예를 들어, 그림 1에서 표현된 것과 같은 태스크의 수행양상에서 T_A 가 시작 태스크이고 T_B, T_C 는 T_A 태스크의 *Fork* 연산에 의해 생성된 병렬 태스크라고 하면, T_A의 초기 좌표값과 레이블은 다음과 같다.

$$T_A(Xs,Xe) = T_A(Ys,Ye) = (MinInt, MaxInt)$$
$$T_A\ Label = [0,0,(1,MaxInt),(1,MaxInt)]$$

태스크 T_α의 *Fork* 연산에 의해 생성된 하위 노드를 T_β라 하면 *Fork* 알고리즘은 다음과 같다.

가정 1 : T_α의 x축 영역의 크기를 $Vx(T_\alpha)$=MaxInt-MinInt+1 라고 하고, y축 영역의 크기를 $Vy(T_\alpha)$=MaxInt-MinInt+1 라고 한다.
가정 2 : Fork 시 생성되는 Task의 수를 N 이라고 한다.

Algorithm 1 [Fork]:

if $T_\alpha(N_{dep})$ is odd

$$T_\beta(Xs) = T_\alpha(Xs) + (N-I) \times Vx(T_\alpha)/N$$
$$T_\beta(Xe) = \begin{cases} T_\beta(Xs) + Vx(T_\alpha)/N - 1 & \text{if } I < N \\ T\alpha(Xe) & otherwise \end{cases}$$

where I = N, N-1, N-2,...,1

$$T_\beta(Ys) = T_\alpha(Ys)$$
$$T_\beta(Ye) = T_\alpha(Ye)$$

else if $T_\alpha(N_{dep})$ is even

$$T_\beta(Xs) = T_\alpha(Xs)$$
$$T_\beta(Xe) = T_\alpha(Xe)$$
$$T_\beta(Ys) = T_\alpha(Ys) + (N-I) \times Vy(T_\alpha)/N$$
$$T_\beta(Ye) = \begin{cases} T_\beta(Ys) + Vy(T_\alpha)/N - 1 & \text{if } I < N \\ T\alpha(Ye) & otherwise \end{cases}$$

where I = N, N-1, N-2,...,1

end if

$$T_\beta(J_t) = T_\alpha(J_t)$$
$$T_\beta(N_{dep}) = T_\alpha(N_{dep}) + 1$$

여기서, 내포깊이에 따라 영역분할을 달리하는 이유는 분할의 균형감을 유지하고 내포 태스크를 쉽게 식별하기 위함이다. 이는 상위 태스크의 한 축이 하위 태스크의 한 축과 항상 일치하므로 상호간 식별을 용이하게 한다.

태스크 T_α의 *Join* 연산에 의해 결합된 태스크를 T_γ라 하면 *Join* 알고리즘은 다음과 같다.

Algorithm 2 [Join]:

$$T_\gamma(Xs) = \min\{T_\alpha(Xs), T_\gamma(Xs)\}$$
$$T_\gamma(Xe) = \max\{T_\alpha(Xe), T_\gamma(Xe)\}$$
$$T_\gamma(Ys) = \min\{T_\alpha(Ys), T_\gamma(Ys)\}$$
$$T_\gamma(Ye) = \max\{T_\alpha(Ye), T_\gamma(Ye)\}$$
$$T_\gamma(J_t) = \max\{T_\alpha(J_t), T_\gamma(J_t)\} + 1$$
$$T_\gamma(N_{dep}) = T_\alpha(N_{dep}) - 1$$

위의 알고리즘에 의해 형성된 각 태스크의 레이블링 결과는 그림 2와 같다.

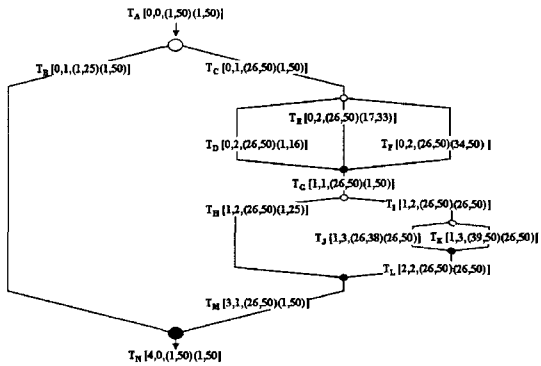


그림 2. 레이블이 붙혀진 병렬 스레드
Fig. 2. Labeled parallel threads

3.2. Representation on 2-D plane

두 번째 단계는 첫 번째 레이블링 단계에서 생성된 레이블 정보를 이용하여 2차원 평면상에서의 고유 영역분할 표현이다. 이는 영상화를 시키기 위한 전처리 단계로써 사용자에게 태스크의 내포성과 병렬성을 직관적으로 판단할 수 있는 2차원 정보를 제공한다. 레이블 정보는 각 태스크를 식별하기 위한 단순 표현인 반면에 평면상의 표현은 사용자에게 보다 직관적으로 보여져야하므로 분할 영역의 좌표를 내포 수분이 생길 때 마다 가변적으로 조정할 필요가 있다. 예를 들면 TB 와 같이 내포성이 없는 태스크에 넓은 영역이 할당되지 않게 하기 위함이다.

영역분할은 내포깊이에 따라 분할 방향을 다르게 하는데, 내포깊이가 홀수이면 가로영역을 분할하게 되고 짝수이면 세로영역을 분할영역으로 확보하게 된다. 영역 분할시 *Fork*에 의한 태스크 생성은 삼각형 기호를 사용하고 내부에는 내포깊이를 표시한다. *Join*에 의한 태스크의 결합은 사각형으로 표시하고 해당 레이블을 내부에 표시한다. 또한 특정 태스크의 내포성이 증가하게 되면 레이블 정보의 좌표와 달리 좁은 영역에 표현되는 양상을 방지하기 위하여 최종적인 영역은 태스크별 균형감을 유지하면서 재조정된다. 그림 3는 그림 2의 POEG에 해당하는 레이블 정보에 의해 영역분할 방법을 적용한 2차원 평면 표현의 결과이다.(*MaxInt*=50라고 하면)

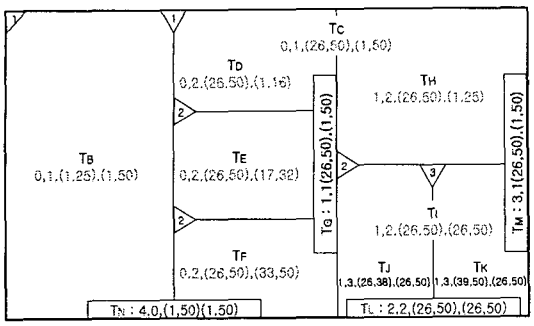


그림 3. 2차원 상의 영역 분할
Fig. 3. Partition volume on 2-dimensional plane

3.3. Image Visualization

세 번째 단계는 2차원 평면상에서의 분할 영역을 영상 시각화하는 것이다. 영상 시각화를 위해서는 몇 가지 조건이 필요하다. 먼저, 평면상의 대칭성을 중요시 한다. 이는 본래의 태스크 영역구분의 특징을 유지하기 위한 조건이다. 두 번째로는 영상화의 결과가 이전 단계에서 만들어진 레이블 정보와 무관하지 않게 시각적으로 표현되어야 하며, 사용자가 영상의 시각적 효과만으로 본연의 태스크 수행양상을 직관적으로 판단할 수 있어야 한다. 세 번째로는 태스크내의 내포깊이나 병렬성에 맞게 크기를 상대적으로 유지해야 한다. 이는 많은 병렬성을 가지는 태스크는 보다 넓은 평면분할에 표현되어야 한다는 것이다. 네 번째로는 영상 시각화의 결과는 향후 다른 영상과 비교대상이 될 수 있으므로 일정한 규칙에 의해 영역별 색상을 유지해야 한다는 것이다. 2차원의 공간상의 문제를 극복하기 위한 3차원적인 시각화 방법들은 단위 공간상에 많은 정보를 보일 수는 있지만, 사용자로 하여금 수행양상을 직관적으로 식별하는데 어려움이 있다. 이는 공간좌표를 사용함으로써 논리적인 가독성을 저해하기 때문이다. 영상 시각화를 위한 Fork와 Join 연산에 따른 기본적인 처리는 다음과 같다.

Fork 연산:

1. 레이블 정보의 영역좌표에 흰색 *Fork* 라인을 긋는다.
2. 영역색상은 같은 상위 태스크내의 각 태스크를 식별하기 위하여 동일 색상이 아닌 다른 색상으로 채운다.
   a. 홀수내포는 상단레벨의 Red에서 좌측레벨의 Blue방향
   b. 짝수내포는 좌측레벨의 Blue에서 상단레벨의 Red방향
   c. 우측레벨의 투명도를 적용한다.(N_{dep} * 0.1)

Join 연산:

1. 레이블 정보의 영역좌표에 의해 청색 *Join*라인을 긋는다.
2. 내포깊이별 구분을 위하여 해당 레벨별로 색상을 가진다.
   a. Join level color (RGB) = (0, 0 , N_{dep} * 10 + 100)

각 축의 색상은 0부터 255사이의 레벨로 표현되며, 각 분할 영역의 시작 색상값이 된다. *Fork* 연산에 의해 생성된 태스크는 이미 분할영역과 관련한 레이블 정보를 가지고 있기 때문에 그 영역에 해당하는 위치의 시작 색상값이 영역의 색상이 된다. 영역의 색상은 상단축과 좌측축의 색상값을 가로방향으로 혼합하여 표시한다. 내포깊이가 홀수이면 상단축을 시작으로 혼합하고, 내포 깊이가 짝수이면 좌측축을 시작으로 혼합한다. 태스크간 *Join* 연산에 의해 생성되는 조인 라인은 파랑 색상으로 표시되는데 내포깊이에 다른 서로 다른 색상 레벨을 가진다. 영역의 구분은 *Join*이 되는 태스크 중에서 최소 영역좌표에서 최대 영역좌표까지를 구분한다.

위의 처리 과정을 통해서 영상 시각화된 병렬 프로그램 수행양상의 한 예를 그림 4에 나타내었다. 이것은 그림 3의 평면 영역 표현을 영상 시각화한 것이다.

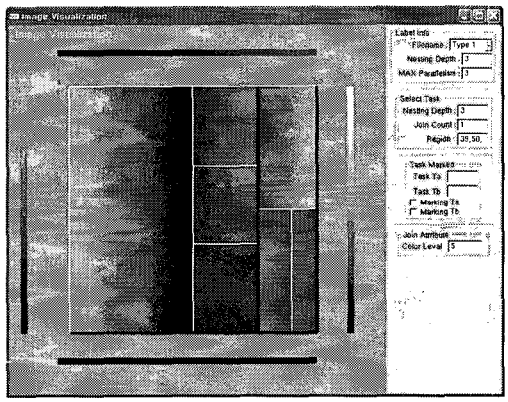
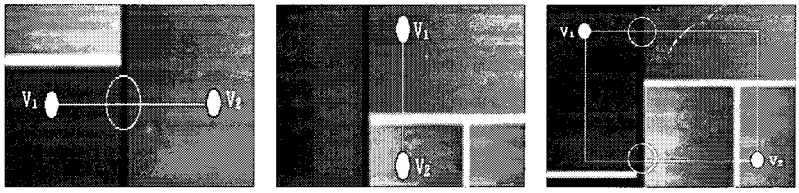


그림 4. 그림 3의 영상 시각화
Fig. 4. Image visualization of Fig. 3

영상 시각화에 내포되어 있는 수행양상 정보는 차후에 본 도구에서 로드하여 해당 수행양상을 재표현할 수 있도록 적절한 포맷으로 저장된다. 영역별 좌표와 색상정보를 저장함으로써 실제 영상을 저장하는데 소요되는 정보량을 줄여 유연성을 높인다.

Ⅳ. Verification and Results

본 논문에서 제안한 기법의 성능을 검증하고 그 결과를 보이기 위해 Windows 환경에서 C++/OpenGL를 사용



(a) (b) (c)

그림 5. 검증 절차에 따른 두 태스크사이의 사건을 위한 영상 가시화 결과

Fig. 5. Results of image visualization for the events between two tasks by the verification procedure

하여 도구를 구현하였다. 그림 4에 나타낸 도구는 사용자와의 상호작용을 효과적으로 수행하기 위하여 다양한 옵션을 내장하였다. 영상 시각화된 수행양상 정보에 대해 부분적인 뷰 기능을 내장하여 사용자가 전역적인 정보이외에 상세한 수행양상을 기존의 POEG등의 방법으로 사용자에게 제공할 뿐만 아니라, 해당 소스 라인을 보여줌으로써 디버깅과정에도 이용될 수 있다.

4.1. Verification

병렬 프로그램의 대표적인 오류 중에서, 경합의 경우에 있어서 두 사건간의 순서화와 병렬성을 검사하는 것은 매우 중요하다. 읽기와 쓰기사건의 경우 순서화된 관계에서는 경합이 발생하지 않기 때문에 두 사건의 병렬성을 검사하는 것이 경합 탐지를 위해서 가장 중요한 절차이기 때문이다. 영상 시각화 결과를 통해 병렬성을 검사하기 위한 절차는 다음과 같다.

가정 3 : 시작 정점좌표(낮은 내포깊이)를 $V_\alpha(x,y)$, 끝 정점좌표(높은 내포깊이)를 $V_\beta(x,y)$ 라고 할 때,

- V_α 와 V_β 가 동일 레이블 정보를 갖고 있으면 순서화관계에 있다.
- $V_\alpha(x) = V_\beta(x)$ or $V_\alpha(y) = V_\beta(y)$: V_α에서 V_β를 잇는 직선을 연결한다. ($\overrightarrow{V_\alpha V_\beta}$)
  만약, $\overrightarrow{V_\alpha V_\beta}$ 상에 V_α의 내포깊이보다 작거나 같은 조인 라인이 존재하면, 순서화관계에 있고 그렇지 않으면 병렬관계에 있다.
- $V_\alpha(x) <> V_\beta(x)$ and $V_\alpha(y) <> V_\beta(y)$: V_α과 V_β를 대각좌표로 하여 직각사각형을 그린다. 사각형의 외부 테두리 상에 V_α의 내포깊이보다 작거나 같은 조인 라인이 존재하면, 순서화관계에 있고 그렇지 않으면 병렬관계에 있다.

4.2. Results

그림 5은 위의 병렬성 검사 절차를 적용한 두 태스크간의 사건에 대해 영상 가시화 결과를 나타내었다. 그림 5에서 (a)는 서로 다른 태스크에서 V_1과 V_2사건이 동일한 y축에 존재할 때의 예이다. 내포 깊이가 낮은 V_1에서 V_2를 잇는 직선상에 V_1의 내포 깊이(N_{dep}=2)와 동일한 조인색상(RGB code=(0,0,120))이 존재하므로 두 사건은 순서화 관계에 있다고 판단할 수 있다. (b)는 서로 다른 태스크에서 V_1과 V_2사건이 동일한 x축에 존재할 때의 예이다. 내포 깊이가 낮은 V_1에서 V_2를 잇는 직선상에 V_1의 내포 깊이(N_{dep}=2)보다 작거나 같은 조인색상이 존재하지 않으므로 두 사건은 병렬 관계에 있다고 판단할 수 있다. 마지막으로 (c)는 서로 다른 태스크에서 V_1과 V_2 사건이 x, y축 어느 한 축도 동일하지 않기 때문에 직선을 연결 할 수는 없다. 그래서 두 정점을 잇는 직각사각형의 외부 테두리 상에서 V_1의 내포 깊이보다 작거나 같은 조인색상을 탐색한다. 흰색 타원과 같이 두 조인색상이 탐색되었으므로 두 사건은 순서화 관계에 있다고 판단할 수 있다.

4.3. Analysis

본 논문에서 제안한 병행성 판별 방법은 레이블링 정보만을 이용하는 기존의 판별 방법들과는 접근상의 차이가 있어 직접적인 비교분석을 힘들게 한다. 기존의 시각화 방법들은 판별보다 수행양상을 사용자에게 보여 주는 것이 목적이지만, 본 본문에서 영상정보를 이용한 시각화는 병행성을 판별하기 위한 정보로 이용되고 있다. 이는 성능개선을 위한 접근보다는 영상 정보를 이용한 새로운 판별법을 제안하는 것이 목적이였기 때문이다.

표 2. 레이블링 기법의 효율성

Table 2. Efficiencies of labeling techniques

Labeling	Space
Tsak Recycling [4]	$O(VT+T^2)$
OS Labeling [17]	$O(VN)$
NR Labeling [15]	$O(V+NT)$
제안기법	$O(VT)$

하지만, 레이블링 단계에서 필요로 하는 기억공간은 기존의 레이블링 기법에 비해 현저히 향상됨을 표 2에서 확인할 수 있다. 여기서, V는 감시을 위한 공유변수의 개수를 의미하고, T는 최대병렬성을 의미한다. 그리고 N은 최대 내포 깊이을 나타낸다. 이전의 레이블링는 내포깊이에따라 복잡도가 높아지는 반면, 본 논문에서 제안하는 레이블링 기법은 각 태스크는 다른 태스크의 레이블링 정보의 관계정보를 보유할 필요가 없기 때문에 내포 깊이에 독립적이기 때문에 공간 복잡도를 크게 향상시킬 수 있다.

V. 결론 및 향후연구

본 논문은 내포 병렬성을 가지는 병렬 프그램에서 사건간의 병렬성을 판별하는 새로운 방법을 제시하였다. 복잡한 수행 양상을 추상화하는 동시에 직관력을 높일 수 있도록 2차원 레이아웃으로 전역적인 구조의 개요를 보이는데 집중하였다. 또한, 이를 영상으로 가시화함으로써 수행양상의 보존과 영상정보로 병렬관계인지를 식별할 수 있었다. 2차원 평면상에서 분할영역의 에지에 색상에 따른 의미를 부여함으로써 레이블 정보 없이도 태스크상의 이벤트에 대한 병렬성 여부를 결정할 수 있었다. 향후에는 제안한 레이블링 기법을 이용하여 공간적 복잡도가 최적의 상태($O(1)$)를 가질 수 있는 방법에 대한 연구가 이어질 것이며, 이 방법의 3D 버전도 고려해 볼 것이다. 영상정보의 의미를 모두 수용하면서 더 효과적인 3차원 가시화 방법에 대한 연구도 이루어 질 것이다.

참고문헌

[1] Audenaert, K., "Maintaining Concurrency Information for On-the-fly Data Race Detection," *Int'l Conf. on Parallel Computing : Fundamentals, Applications and New Directions*, Bonn, Germany, pp.319-326, Sept., 1997, Advances in Parallel Computing, Elsevier, North-Holland, Amsterdam, Vol.12, 1998.

[2] Citron D., D. G. Feitelson, and I. Exman, "Parallel Activity Roadmaps," *Int'l Conf. on Parallel Computing (ParCo'93)*, Parallel Computing: Trends and Applications pp. 593-596, Elsevier Science, 1994.

[3] Dagum, L. and R. Menon, "OpenMP : an Industry-Standard API for Shared-Memory Programming," *Computational Scince and Engineering*, 5(1), IEEE, 46-55, Jan.-March, 1998.

[4] Dinning A., and E. Schonberg, "An Empirical Comparison of Monitoring Algorithms for Access Anomaly Detection," *2nd Symp. on Principles and Paractice of Parallel Programming*, pp. 1-10, ACM, 1990.

[5] Dinning A., and E. Schonberg, "Detecting Access in Programs with Critical Sections," *2nd Workshop on Parallel and Distributed Debugging*, pp.85-96, ACM, May 1991.

[6] G. Kim, Y. Kim and Y. Jun, "Effective Race Visualization for Debugging OpenMP Programs," *Proc. 31nd KISS Conference*, Vol. 31 No. 02 pp. 13-15 Oct. 2004.

[7] Helmbold, D. P., C. E. McDowell, and J. Wang, "TraceViewer: A Graphical Browser for Trace Analysis," TR-90-59, UCSC. 1990

[8] K. Audenaert, "Clock Trees: Logical Clocks for Programs with Nested Parallelism," *IEEE Trans. Software Eng.*, vol. 23, no. 10, 1997.

[9] Kim, J., D. Kim, and Y. Jun, "Scalable Visualization for Debugging Races in OpenMP Programs," *Proc. of the 3rd Int'l Conf. on Communications in Computing (CIC)*, pp.259-265, Las Vegas, Nevada, June 2002.

[10] Kuhn, B., P. Petersen, and E. O'Toole, "OpenMP versus Threading in C/C++," EWOMP '99, Lund, Sweden, Sept. 1999.

[11] Netzer R. H. B., and B. P. Miller, "What are Race Condition? Some Issues and Formalization," *Letters on Programming Lang. and System*, 1(1):74-88, ACM, 1992.

[12] OpenMP Architecture Review Board, OpenMP Fortran Application Program Interface, Version 2.0, Nov., 2000.

[13] S. Park, M. Park, and Y. Jun, "A Comparision of Scalable Labeling Schemes for Detecting Races in OpenMP Programs," *Int'l Workshop on OpenMP Applications and Tools (Wompat)*, pp. 66-80, West lafayette, Indiana, July 2001.

[14] Sue, U. H. and C. M. Pancake, "Graphical Animation of Parallel Fortran Programs," *Supercomputing '91*, pp. 491-500, ACM/IEEE, Nov. 1991.

[15] Y-K. Jun and K. Koh, "On-the-fly Detection of Access Anomalies in Nested Parallel Loops," *Proc. ACM/ONR Workshop Parallel and Distributed Debugging*, pp. 107-117, May 1993.

[16] Zernik, D., M. Snir, and D. Malki, "Using Visualization Tools to Understand Concurrency," *Software*, 9(3): 87-92, IEEE, May 1992.

[17] Mellor-Crummey, J., "On-the-fly Detection of Data Races for Program with Nested Fork-Join Parallelism," *Supercomputing '91*, pp. 24-33, ACM/IEEE, Nov. 1991.

저자소개

박 명 철(Myeong-Chul Park)

1999년 한국방송대학교 컴퓨터과학과 (이학사)
2002년 경상대학교 소프트웨어학과 (공학석사)
2005년 경상대학교 컴퓨터과학과 박사수료
※관심분야 : 컴퓨터 비젼, 영상처리, 시각화, 임베디드 소프트웨어, 병렬프로그래밍 및 디버깅 등

하 석 운(Seok-Wun Ha)

1979년 부산대학교 전자공학과 (공학사)
1986년 부산대학교 전자공학과 (공학석사)
1995년 부산대학교 전자공학과(공학박사)
1993년 3월~ 경상대학교 컴퓨터과학과 교수
※관심분야 : 디지털신호처리, 신경회로망, 영상처리, 컴퓨터비젼, 임베디드 시스템