병렬 프로그램의 디버깅을 위한 관심정보 모니터링 시스템

박명철

송호대학 컴퓨터정보과

Interest-Information Monitoring System for Debugging of Parallel Programs

Myeong-Chul Park

Dept. of Computer Information, Songho College

E-mail : africa@songho.ac.kr

요 약

본 논문에서는 OpenMP 기반의 병렬프로그램을 대상으로 각 스레드의 수행양상을 추적할 수 있는 감시 시스템을 제안한다. 기존의 감시 시스템은 각 스레드의 레이블링 정보를 이용하여 접근역사를 통한 분석기법이 대부분 이였다. 이는 대량의 정보 생성으로 인한 시간적, 공간적 복잡도를 높이는 문제점을 가진다. 본 논문에서는 관심 정보에 따른 스레드만을 대상으로 추적 정보를 생성하고 사용자에게 직관성 높은 정보로 제공하기 위한 시각화 시스템을 동시에 제공한다. 시각화를 위한 모델은 영상정보를 기반하여 구성되며 이는 영상처리 기법을 통하여 프로그램 수행양상을 인지할 수 있게 한다. 따라서, 본 논문은 병렬프로그램을 효과적으로 디버깅할 수 있는 환경을 제공한다.

ABSTRACT

In this paper, proposes the monitoring system it will be able to trace the executed of each threads in OpenMP based a parallel program. The monitoring system of existing in uses each threads label information and the analysis technique which uses the access-history was most. This has the problem which raises the time and space complexity which is caused by with massive information creation. In this paper, only the thread which includes interest information it creates tracking information with the target. And it provides information which is intuitive to the user it provides the visualization system for to a same time. The visualization model is composed the images-information of a base. This does to be it will be able to understandable a program execute situation using an image processing technique. Therefore, this paper provides the parallel program an effective debugging environment.

키워드

parallel program, debugging, monitoring system, OpenMP

Ⅰ. 서 론

병렬 프로그램은 스레드간의 수행을 독립적으로 병행 수행할 수 있으므로 프로그램의 전체적인 성능과 신뢰성을 향상시킬 수 있는 장점을 가지고 있다.[1] 그러나 이러한 병렬 프로그램은 각 스레드간의 비동기적 접근에 의해 많은 오류를 야기시켜 비결정적인 결과를 초래하는 경우도 발생한다.[2] OpenMP와 같은 공유메모리 기반의 병렬프로그램은 스레드간의 병행성 관계에서 메모리 접근사건이 발생할 경우 경합이라는 치명적인 이상접근 오류를 발생하게 된다.[3,4] 이러한 오류를 디버깅하기 위한 방법으로 스레드간의 병행성 관계를 모든 접근사건에 따른 접근역사를 분석하여 디버깅하는 기법이 사용되고 있다.[5] 그러나, 이 기법들은 방대한 메모리를 필요로 하고 분석 시 매우 큰 시간적 복잡도를 가지게 된다. 접근정보의 추상화를 통해 이 문제를 해결하는 시도도 있기는 하지만, 이 또한 본질적인 스레드들의 수행양상을 사용자가 인식하기 어려워진다는 문제점을 가지게 된다.[6] 본 논문에서는 스레드 생성 전에 사용자로 하여금 관심정보를 선

택하게 하고, 선택된 관심정보에 대한 접근역사만을 기록하여 모니터링 할 수 있는 시스템을 제안한다. 모니터링 결과는 사용자의 이해와 식별을 용이하게 하기 위하여 3차원 시각화를 통하여 제공하게 된다. 병행성 판별을 위한 레이블링 기법은 기존의 연구결과인 e-NR 레이블링[7]을 이용하였다. 본 논문의 2장에서는 논문에서 제시하는 모니터링 시스템 구조에 대해 설명하고, 3장에서는 시각화 시스템에 대해 설명한다. 그리고 4장에서는 결론과 향후 연구를 밝힌다.

II. 모니터링 시스템의 구조

본 장에서는 제안하는 모니터링 시스템의 구조에 대해 설명한다. 첫 번째 단계에서는 OpenMP 디렉티브를 사용한 소스 프로그램을 기반으로 프로그램 스캔 작업부터 이루어진다. 소스 프로그램은 C언어를 기반으로 한다. 프로그램 스캔에서는 OpenMP의 shared 디렉티브를 파싱하여 해당 공유메모리 리스트를 작성한다. 두 번째 단계에서는 제공되는 리스트에서 사용자가 관심을 가지는 공유 변수를 선택하게 된다. 선택된 관심정보에 대한 변형된 소스코드를 생성하는 것이 이 단계에서 이루어진다. 변형코드에서는 기존의 e-NR 레이블링을 위한 레이블링 생성 함수가 삽입되게 된다. 최종적으로 변형된 목적코드를 수행하여 분석 작업에 필요한 스레드 정보와 접근 정보를 얻게 된다.

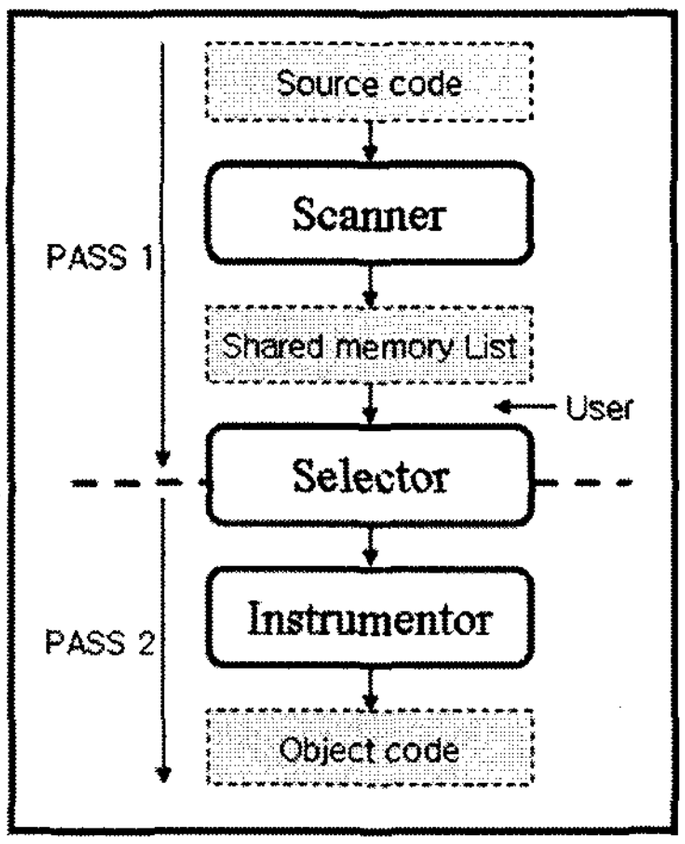

그림 1. 모니터링 시스템의 Pass1-2

세 번째 단계에서는 이 정보를 이용하여 접근역사를 분석하게 되는데, 크게 세 가지 기능을 담당하게 된다. 먼저, 스레드간의 병행성 정보를 판단하기 위한 스레드 레이블링 정보를 생성하는 기능과 관심정보에 대한 접근사건들을 기록한 접근역사 정보를 생성하는 기능을 가진다. 그리고 마지막으로 이상 접근의 존재 여부를 판단하는 기능을 가진다.

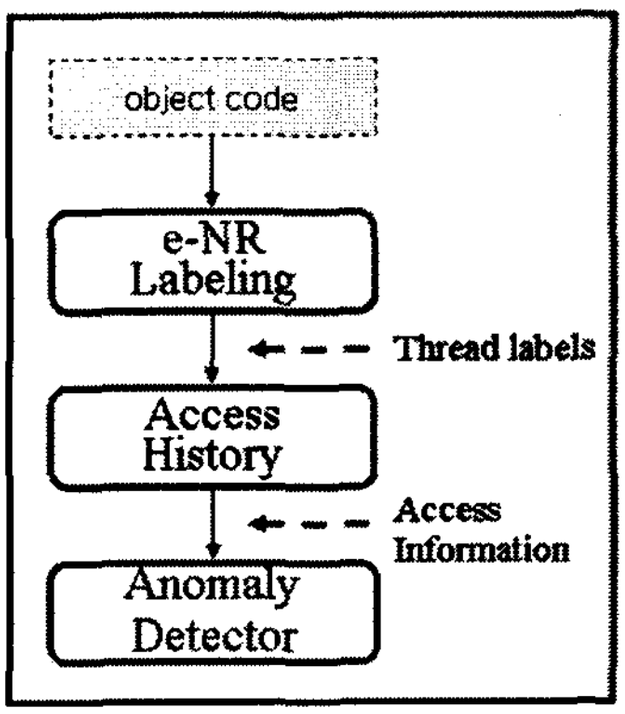

그림 2. 모니터링 시스템의 Pass3

세 번째 단계를 거치고 나면, 프로그램의 스레드 정보 및 병행성 정보, 공유 메모리 접근 정보가 수집되게 된다. 이 정보들은 최종적으로 시각화 엔진으로 넘겨지게 되고 사용자에게 3차원 영상 정보로 제공되게 된다. 생성되는 각 정보는 표 1, 표2, 표3에서 소개한다. 표 1은 각 스레드의 레이블정보을 보이고 있다. 그리고 표 2는 관심정보에 대한 접근정보를 보이고 있다. 마지막으로 표 3은 이상 현상을 유발할 수 있는 접근사건의 쌍을 보이고 있다.

표 1. 스레드의 레이블링 정보

```
01:[0,0,( 1,50),( 1,50),-,-]
02:[0,1,( 1,25),( 1,50),2,1]
03:[0,1,(26,50),( 1,50),2,1]
04:[0,2,(26,50),( 1,16),3,1]
05:[0,2,(26,50),(17,33),3,2]
06:[0,2,(26,50),(34,50),3,3]
07:[1,1,(26,50),( 1,50),-,-]
```

표 1의 레이블링 정보의 타입정보는 스레드 식별번호, 스레드의 조인횟수, 스레드의 내포깊이, X축 분할영역, Y축 분할영역, 형제 스레드 수, 형제 스레드의 순번 순으로 이루어져 있다.

표 2. 관심정보의 접근역사

```
02:R, 37[0,1,( 1,25),( 1,50),2,1]
04:R, 48[0,2,(26,50),( 1,16),3,1]
05:W, 52[0,2,(26,50),(17,33),3,2]
07:W, 60[1,1,(26,50),( 1,50),-,-]
```

표 2의 접근역사 정보는 스레드 식별번호, 접근구분, 소스라인번호, 스레드 정보 순으로 기록되어 있음을 보이고 있다.

표 3. 이상 접근 사건 리스트

02:R, 37 - 05:W, 52
02:R, 37 - 07:W, 60

표 3의 이상 접근 리스트에서 첫 줄의 의미는 37번 행에서 02번 스레드가 읽기 접근을 했을 때 52번 행에서 05번 스레드가 쓰기 접근을 하게 된 경우이다. 두 스레드는 병행관계에 있음으로 이 두 사건의 쌍은 이상 현상을 유발 할 수 있다. 그림 3은 이 상황을 POEG(Partial Order Execution Graph)으로 보이고 있다.

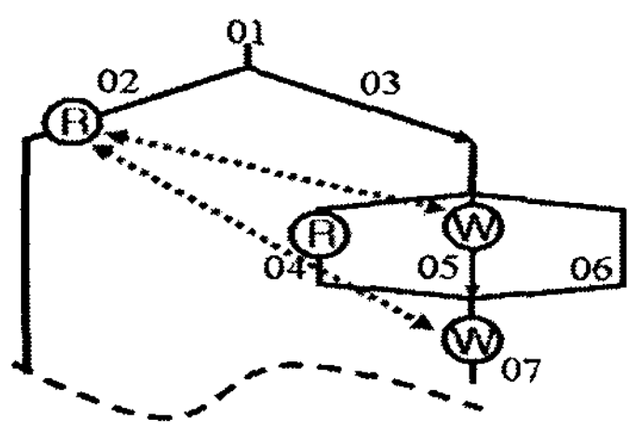

그림 3. 이상 현상의 쌍

그림 4는 실험에 사용된 OpenMP프로그램의 한 부분을 보여 주고 있다.

```
#pragma omp parallel for shared(S)
   for(i=0;i<2;i++) {
      if(i==0)  S++;
      else {
        #pragma omp parallel for shared(S)
          for(j=0;j<3;j++) {
            y++;
            S = S - y;
          }
      }
   . . .
```

그림 4. OpenMP프로그램의 예제

III. 시각화 시스템의 구조

세 번째 단계의 분석엔진을 통하여 생성된 스레드, 접근역사 정보는 시각화 엔진으로 전이되어 사용자를 위한 시각화 모델을 구성하게 된다.

그림 5는 시각화 시스템의 전체적인 구조를 보이고 있다. 모니터링 시스템의 3단계 분석엔진에 의해 생성된 스레드의 레이블링 정보와 접근정보 및 이상접근 사건 정보가 2차원 시각화 엔진과 3차원 엔진에 각각 부여 된다. 레이블링 정보의 X, Y 좌표 분할 정보를 이용하여 2차원 시각화의 영상화면을 구성하게 된다. 또한, 내포깊이와 형제 노드 수 등을 이용하여 3차원 디스크 뷰를 구성하게 된다. 그리고 사용자의 이해도를 높이기 위하여 사용자 인터페이스(UI) 화면을 추가하였다.

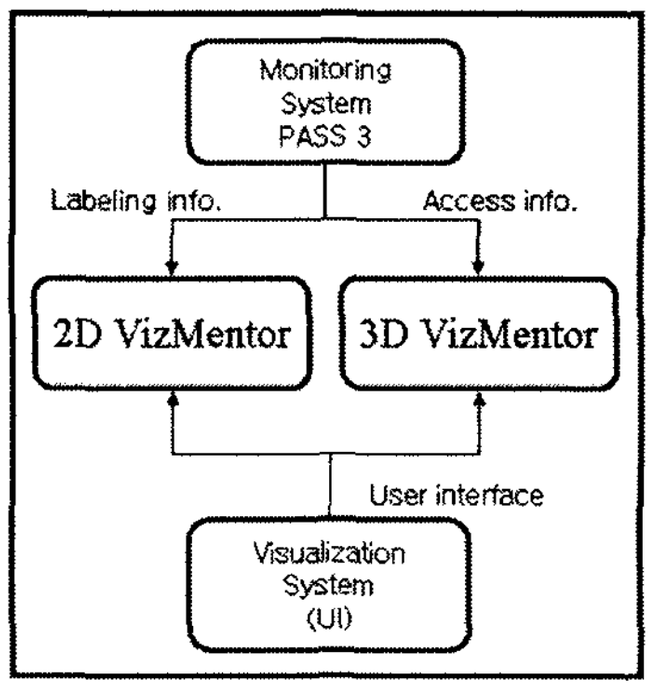

그림 5. 시각화 시스템의 구조

그림 6은 실제 구현된 시각화 화면을 보여 주고 있다. 그림 6의 왼쪽은 2차원 뷰로서 스레드 레이블링 정보에 의해서 생성된 영상정보이다. 그림에서 흰색 라인은 스레드가 생성(FORK)되는 부분을 의미하고 파란색 라인은 스레드가 합쳐지는(JOIN)부분을 의미한다. 초록색에서 시작하여 붉은색으로 그라데이션 된 영역은 홀수 내포깊이를 의미하고 붉은색에서 초록색으로 그라데이션 된 부분은 짝수 내포깊이를 의미한다. 이렇게 내포깊이에 따라 다르게 배치하는 것은 좌우의 균형감을 맞춤으로서 제한된 영역에 더 많은 스레드 정보를 배치하기 위함이다. 영역 내에 사용자가 마우스를 이용하여 영역의 특정 부분을 클릭하면 해당 스레드 정보를 화면 하단의 인터페이스 뷰에 보여주게 된다. 인터페이스 뷰에서 레이블 정보, 내포 깊이 등의 식별 정보를 확인 할 수 있다. 또한, 해당 생성 소스라인과 조인라인을 보여 주게 된다.

그림 6의 오른쪽은 3차원 뷰를 보이고 있다. 여기서, 빨간색 원 객체(NT)는 내포스레드를 가지는 스레드를 표시하는 것이고, 파란색 원 객체는 내포스레드가 없는 독립된 스레드를 표시한 것이다. 그리고 초록색 원 객체(nNT)는 Join된 이후의 스레드를 표시하며 파란 원판 객체(TBone)는 병행성을 가지고 있는 형제 스레드를 표시하는 백본 디스크이다. 백본 디스크상의 스레드의 배치는 레이블링 정보의 형제 스레드 수 정보를 이용하여 균등 분할하여 배치된다. 배치된 스레드 중에서 NT스레드가 사용자에 의해 활성화가 되면, 디스크의 외곽방향으로 회전하여 새로운 백본 디스크를 생성하게 된다. 동시의 동일 디스크의 NT 스레드가 활성화가 되면 상위 디스크의 반경이 늘어나면서 하위 디스크를 보이게 된다.

일반적으로 사용자는 두 사건간의 병행성 관계나 순서화 관계를 식별하기 위한 목적으로 스레드를 활성화 시키므로 디스크 반경 상의 제한은 큰 문제가 되지 않을 것이다. 또한 스레드 추상화를 이용하여 사용자의 관심 스레드 외에는 백본 디스크를 디스플레이 하지 않고 NT스레드 상태로 표현되기 때문에 기존의 복잡한 스레드 표현은 줄어들 것이다. 사용자가 전체 스레드의 확장된 모습을 보고자 할 때는 축소, 확대, 회전, 백본 온닉 등의 다양한 사용자 인터페이스를 이용하여 효과적으로 표현하게 된다. 또한 표 3의 이상 접근사건 리스트를 이용하여 3차원 공간상의 상대 스레드를 보여주므로서 순서화 관계를 고려해 볼 수 있는 정보를 제공한다. 본 논문에서 제안한 기법의 성능을 검증하고 그 결과를 보이기 위해 Windows 환경에서 C++/OpenGL를 사용하여 도구를 구현하였다.

IV. 결 론

본 논문은 병렬 프로그램에서 특정 관심정보에 대한 모니터링 및 시각화 시스템을 제안하였다. 병렬 프로그램의 복잡한 수행양상의 전역적인 수행구조를 보이는 동시에 사용자로 하여금 직관적으로 스레드들 간의 관계를 인지할 수 있도록 설계하였다. 제안한 시스템은 스레드 정보에 해당하는 소스코드를 보임으로서 프로그램을 이해하는데 도움이 되며, 디버깅 환경에 활용될 수 있을 것이다. 향후에는 각 엔진들이 수행 중에 동작할 수 있는 동적 분석도구를 개발하여 연계시키는 연구가 이루어 질 것이다.

참고문헌

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

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

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

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

[5] Kim, J., D. Kim, and Y. Jun, "Scalable Visua lization for Debugging Races in OpenMP P rograms," *Proc. of the 3rd Int'l Conf. on Com munications in Computing (CIC)*, pp.259-265, Las Vegas, Nevada, June 2002.

[6] 임재선 "*OpenMP Program의 경합 탐지를 위한 3 차원 원추상의 확장적 사건 시각화*", 경상대학교 석사 학위논문, 2006

[7] 박명철, 하석운 "영상정보를 이용한 병렬 프로그램내의 병행성 판별", *한국해양정보통신학회논문지*, 10권 12호, pp.2132-2139, 2006.

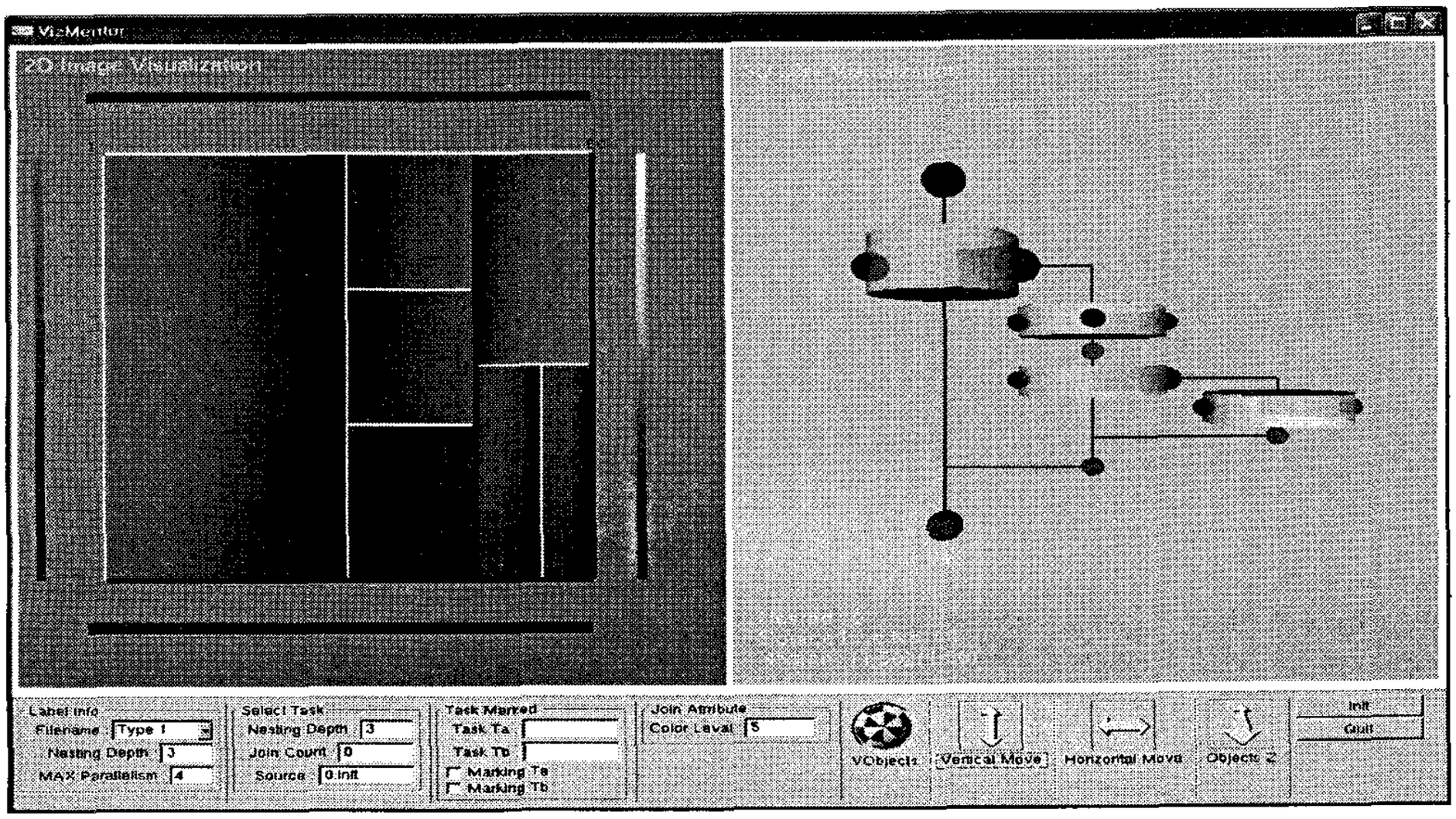

그림 6. 모니터링을 위한 시각화 시스템