성능 감시기의 가시화층을 위한 일반화된 뷰의 설계

마 대 성 유 진 호 김 병 기
전남대학교 전산학과

A Design of Generalized View for the Visualization Layer of Performance Monitoring Tool

Dae-Sung Ma, Jin-Ho You, Byung-Gi Kim
Department of Computer Science, Chonnam National University

요약

본 논문은 병렬 프로그램의 성능 분석을 위한 성능 감시기에서 가시화층의 일반화된 뷰를 설계하고 구현하였다 대부분의 성능 감시기는 하드웨어에 의존적인 특성화된 뷰를 제공함으로서 이식성이나 확장성이 부족하다. 일반화된 뷰를 제공하는 성능 감시기는 데이터 필터층에서 필터링된 성능 분석 데이터를 이용하여 프로그래머가 정의한 데이터의 범위에 따라 뷰를 스스로 확장할 수 있다 또한, CallBack 기능을 제공하여 관심 있는 데이터를 쉽게 볼 수 있다 프로그래머는 성능 감시기의 일반화된 뷰를 이용하여 다양한 형태의 성능 분석 결과를 볼 수 있다

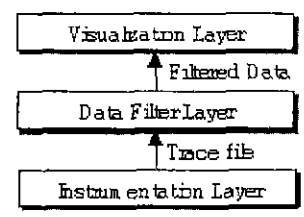
1. 서론

병렬 컴퓨터에서는 기존의 순차 컴퓨터에서 해결할 수 없는 대규모 문제를 해결할 수 있는 처리 능력이 있다. 그러나, 병렬 컴퓨터에서의 프로그래밍은 동시성 때문에 순차적 프로그램보다 효율적인 설계와 개발, 디버깅, 최적화가 어렵다. 프로그래머는 프로세서 사이의 동기적 통신(Synchronize communication)을 제어해 주어야 하며 공유자원(shared resources)을 접근할 수 있게 해 주어야 한다. 최근에 병렬 시스템과 병렬 프로그램이 급속한 성장을 이루고 있지만 병렬 프로그램 개발자를 지원하기 위한 도구는 아직 시작단계에 있다

병렬 프로그램 개발자를 위한 지원 도구로는 병렬 프로그램 디버거(Parallel Program Debugger), 병렬 프로그램 성능 분석기 (Parallel Program Performance Analyzer), 병렬 프로그램 성능 감시기(Parallel Program Performance Monitoring Tool) 등이 연구되고 있다.

병렬 컴퓨터는 시스템이 매우 복잡하고, 구성 요소간의 상호 교류가 빈번하므로 병렬 프로그램의 복잡한 작업을 설명하고 이해를 쉽게 하기 위해서는 텍스트 기반의 분석 도구보다 의미 있는 그래픽으로 성능을 표현해 주는 가시화 도구가 훨씬 더 유용하다. 이중 병렬 프로그램의 성능 분석을 위한 성능 감시기는 병렬프로그램으로부터 추출된 방대한 양의 성능 관련 데이터를 그래픽 요소로 변환하여 가시화 함으로써 프로그래머에게 효율적인 성능분석 환경을 제공하고 있다.

병렬 프로그램의 성능 분석을 위한 성능 감시기의 형태는 인스트루멘테이션(Instrumentation)층, 데이터 필터층, 가시화(Visualization)층으로 구성된다.



<그림 1> 일반적인 성능 감시기의 구조

가시화층에서는 사용자 인터페이스와 함께 다양한 성능 분석 뷰를 제공해주고 있다. 성능 감시기에서 제공하는 뷰는 프

로그래머가 분석하고자 하는 분석 목표에 대한 결과를 애니메이션 형태로 보여준다.

병렬 프로그램은 프로세서의 수, 메모리의 형태, 동시에 수행되는 작업의 양 등 병렬 컴퓨터의 수행 환경에 따라 수행 형태가 달라진다. 따라서 성능 분석을 위한 뷰는 병렬 프로그램이 수행되는 형태에 따라 확장성을 가지고 있어야 한다. 또한, 분석 뷰는 사용자에게 분석 결과에 대한 확실성을 주어야 하고, 분석결과를 쉽게 표현할 수 있어야 한다.

대부분의 성능 감시기는 특정 하드웨어에 의존적으로 개발되므로, 제공되는 성능 분석 뷰 역시 특정 하드웨어 시스템의 분석을 목표로 개발된다 이러한 이유로 병렬 프로그램 성능 감시기는 개발 목표 시스템 외의 다른 시스템에서 동작하는 병렬 프로그램의 성능을 분석하기가 어렵다.

본 연구에서는 다양한 환경에서 동작하는 병렬 프로그램의 성능을 분석하여 보여주는 성능 감시기 개발을 위해 확장성을 지닌 일반화된 뷰를 정의하고 이를 구현하였다. 확장성을 지닌 일반화된 뷰는 병렬 컴퓨터의 특성에 좌우되지 않고 분석된 결과를 나타낼 수 있는 장점을 가졌다.

2. 관련 연구

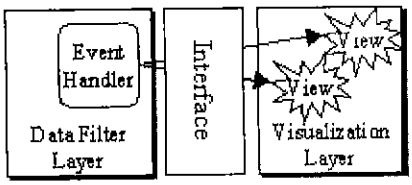
병렬 프로그램의 성능 분석을 위한 성능감시기에서 뷰의 형태는 특성화된 뷰와 일반화된 뷰로 구분 지을 수 있다. 메시지 전달 뷰(Message-Passing View), 시간선 뷰(Time-Line View), 쓰레드 뷰(Thread View)와 같은 특성화된 뷰는 특정 하드웨어에서 동작하는 병렬 프로그램의 성능을 보여주는 뷰이다 일반화된 뷰로는 막대 그래프(BarGraph), 간트 그래프(GanttGraph), 차트 그래프(ChartGraph), 행열 그래프(MatrixGraph)등으로 일반적인 데이터를 이용해 분석을 가능하게 하는 뷰이다.

ParaGraph[2]은 메시지 전달 프로그램의 분석을 위해 개발되었다. 실제 병렬 컴퓨터 상에서 동작하는 메시지 전달 프로그램의 동작과 성능을 애니메이션 규칙에 따라 동작하는 특성화된 뷰를 제공한다 Voyeur[1]는 병렬프로그램의 프로그램에 따른 가시적 뷰의 생성을 용이하게 해주는 프로토타입(prototype) 시스템이다 Voyeur는 병렬프로그램의 뷰를 생성하기 위한 언어 독립적이고 시스템 독립적인 메카니즘을 제공한다 Voyeur는 추상적인 뷰에서, 단지 프로그램 변수들을 보여주는 언어 고유의 혹은 아키텍처 고유의 뷰에 이르기까지 계층적인 뷰를 지원하고 있다.

Panorama[3]는 기존의 병렬 디버거들이 특정 하드웨어 구조에 한정되어 제한된 관점만을 제공한다는데 관심을 두고 개발되었다 특히, MIMD 메세지 전달 시스템에 이식성이 뛰어나고, 기본적으로 제공되는 가시화 뷰외에 사용자 정의형 뷰를 생성할 수 있다 사용자 정의형 뷰는 Display Builder를 사용하여 뷰의 외양과 뷰의 기본적인 요소들을 설계하고, 프로그램에서 추적된 정보를 가지고 뷰를 갱신하기 위한 Tcl 코드를 생성한다. 그러나, 대부분의 도구들은 하드웨어에 의존적인 특성화된 뷰를 제공하고 있고, 일반화된 뷰를 제공하더라도 성능 분석에 제한이 따르는 단점을 지니고 있다.

3. 성능 분석을 위한 일반화된 뷰

성능 감시기의 뷰의 구현은 X 윈도우 시스템 기반의 MOTIF를 이용하여 개발하였다. 뷰의 애니메이션 형태는 각각 뷰가 가지고 있는 Drawing 알고리즘에 의해 구현하였다. 뷰의 애니메이션은 데이터 필터층에서 해당 뷰에 대한 입력값이 갱신되면, 성능 데이터에 연결되어 있는 뷰에 입력값이 전달된다 각 뷰에 대한 동기화를 위해 데이터 필터층과 뷰 사이에는 이벤트 핸들러가 뷰를 제어한다. 또한 각 뷰에는 Call-Back 기능을 제공하여 사용자가 애니메이션되는 뷰의 관심 있는 데이터값을 보고자 할 때 마우스를 클릭하면 마우스 위치의 현재 값을 디스플레이 한다.



<그림 2> DataFilter층과 Visualization층의 인터페이스

3.1 막대그래프(Bargraph)

막대그래프는 각각의 막대가 뷰에 입력되는 노드, 프로세서, 메시지 큐 등 사용자가 정의한 값을 나타내며, 뷰에 대한 입력값이 이벤트 핸들러에 의해 갱신될 때마다 막대 그래프에 대한 값이 변화한다. 막대그래프는 입력값의 최대값과 최소값을 프로그래머가 지정할 수 있으며 각 막대마다 최대 값을 기억하는 히스토리를 제공한다. 막대그래프에 입력 가능한 데이터 형은 integers, float, doubles 형을 지원한다

3.2 구현

막대그래프의 갱신은 기존 성능 데이터(old.value)와 입력되는 성능 데이터(w.value)를 비교하여 값이 변하는 경우 막대 그래프를 다시 그린다. 그래프의 높이는 resources 파일의 막대 높이(resources.height)를 곱하여 구하게 된다.

/* 데이터값에 의한 막대그래프의 높이 */

$$resources.height \times \frac{value - min.value}{max.value - min.value}$$

막대그래프의 Drawing은 이벤트 핸들러에서 가시화층에 전달되는 데이터를 입력으로 사용자가 정의한 최대, 최소값 범위에 드는지 비교한다. 비교된 값은 막대그래프를 다시 그리기 위한 PaintBar( ) 함수를 호출한다. DrawHistory( ) 함수는 각 막대에 입력된 최대값을 히스토리에 저장시켜 갱신하도록 하였다.

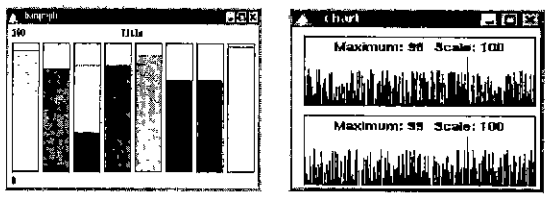
```
/* 막대그래프 Drawing 알고리즘 */

BargraphSetValue(w, value)
BargraphWidget   w;
double           value;
{ if (value < w->bargraph.min_value) {
    perror(err); } /* 최소값과 비교 */
  else if (value > w->bargraph.max_value) {
    perror(err), } /* 최대값과 비교 */
  else  w->bargraph.value = value;
  }
  PaintBar(w, EVENT_VALUE_UPDATE);
  /* 히스토리 Drawing*/
  DrawHistory(w),
}
```

Call-Back 기능은 사용자가 프로그램 분석 중에 흥미 있는 막대를 선택하면 그 당시 Drawing 되었던 데이터 값을 제공한다.

```
CallBack( w, client_data, call_data );
```

<그림 3-(a)>은 막대그래프의 입력 값으로 각 프로세서의 부하를 측정한 것이다. 그림에서 보는 바와 같이 3번과 8번 프로세서의 부하가 낮고 1번과 5번 프로세서의 부하가 높다는 것을 알 수 있다.



(a) (b)

<그림 3> 일반화된 그래프

4. 결론

병렬 컴퓨터의 보급으로 병렬 프로그래밍에 대한 관심이 높아가고 있다. 대부분의 성능 감시기는 병렬 컴퓨터의 특성상 하드웨어에 의존하도록 구현되어 서로 다른 구조를 갖는 이 기종간의 성능 비교 분석이 어려웠다.

본 논문에서는 병렬 프로그램의 성능을 분석하는 성능 감시기에서 가시화층의 일반화된 성능 분석 뷰를 구현하였다. 일반화된 성능 분석 뷰는 데이터 필터층에서 제공되는 각종 성능 분석 데이터를 이용하여 사용자가 요구하는 분석 결과를 보여줌으로써 프로그래머가 다양한 성능 분석 결과를 나타낼 수 있는 장점을 가지고 있다.

참고문헌

[1] D Socha, M L. Bailey, D. Notkin, "Voyeur: Graphical Views of Parallel Programs", "Proceedings of the ACM SIGPLAN/SIGOPS Workshop on Parallel and Distributed Debugging", VOL 24, pp 206-215, January, 1989.

[2] M. T. Health, J. A. Etheridge, "Visualizing the Performance of Parallel Programs," IEEE Software, 8, no 5, pp. 29-39, September, 1991.

[3] J May, "Panorama User's Guide," Department of Computer Science and Engineering University of California, San Diego, August, 1994.

[4] J. May, F. Berman, ''Creating Views for Debugging Parallel Programs'', Dept. of Computer Science and Engineering, Univ. of California, 1994.

[5] J. May, ''Panorama Display Builder: A User's Guide'', Dept of Computer Science and Engineering, Univ. of California, August, 1994.

[6] Deborah Silver, ''Object-Oriented Visualization'', IEEE, pp.54-62, May, 1995.