병렬 프로그램을 위한 PnP 스타일의 성능 가시화기

문상수 °, 김정선, 문영식
한양대학교 전자계산학과
E-mail : {ssmoon, jskim, ysmoom}@cse.hanyang.ac.kr

Plug and Play Style Performance Visualizer for Parallel Programs

Sangsu Moon, Jungsun Kim, Youngshik Moon
Dept. of Computer Science and Engineering, Hanyang University

요 약

본 논문에서는 최적의 성능을 갖는 병렬 프로그램을 개발하는데 필수 도구인 성능가시화기를 이식성, 확장성 그리고 효율성을 고려해 설계 및 구현한 PnP 스타일의 성능 가시화기에 대하여 기술한다. 본 가시화기는 기존 가시화기의 문제점인 수정 및 변용에의 어려움을 해결하기 위하여 독립된 계층구조인 인스트루멘테이션층, 인터페이스층, 가시화층으로 구성함으로써 확장성 및 이식성을 갖도록 하였다. 인스트루멘테이션층은 사건(event)을 포획하기 위해 개발된 라이브러리인 ECL(Event Capture Library)로 구성되며, 인터페이스층은 인스트루멘테이션층과 가시화층간에 확장성 있는 문제중심 인터페이스를 제공하기 위해 개발된 사건 기술 언어 및 Java 문제중심 엑세스 라이브러리로 구성되었다. 그리고 PnP 스타일의 성능 가시화기를 설계함으로써 뷰와 필터의 추가 및 수정이 용이하도록 가시화층을 구현하였다. 이렇게 구현된 성능가시화기는 독립된 도구로 사용될 수 있을 뿐 아니라 병렬 프로그래밍, 디버깅, 그리고 성능 분석이 통합된 프로그램 개발환경 구축의 핵심도구로서 활용될 수 있을 것이다.

1. 서론[1)]

샘플링에 의존하는 순차프로그램의 성능분석과는 달리, 성능이 우수한 병렬프로그램을 개발하기 위해서는 프로그램의 동적 동작특성에 대한 통찰력을 제공할 수 있는 성능분석 도구의 지원이 필수적이다.

본 논문에서는 사람의 탁월한 시각인지 능력을 최대한 활용하여 고성능 병렬프로그램 개발을 용이하게 하는 PnP 스타일의 성능가시화기를 개발하였다. 본 가시화기는 MPI[1] 프로그램을 포함한 병렬프로그램의 복잡한 동작 특성들을 숫자 또는 문자의 형태 대신에 그림의 형태로 시각화하여 사용자에게 제시해 줌으로써 성능 분석 및 패턴에 기초한 오류의 분석을 용이하게 한다 [2,3]. 또한 효율적이면서도 이식성 있는 인스트루멘테이션 라이브러리, Plug-and-play 방식에 의해 다양한 병렬 패러다임 (예, 메시지 전송모델, 쓰레드 기반 공유메모리 모델 등)의 분석을 위해 손쉽게 특화 될 수 있는 가시화 프레임워크, 그리고 이 두 개의 독립된 모듈을 논리적으로 결합시키는 논리적 사건기술언어 및 Java 문제중심의 인터페이스 라이브러리라는 세 개의 모듈에 의한 층구조를 갖도록 설계함으로써 기존 성능가시화 도구들이 안고 있던 범용성의 결여, 성능뷰의 추가 및 수정의 어려움, 사건레코드 개선 및 확장성 결여와 같은 문제들을 해결할 수 있도록 한다.

2. PnP 스타일의 성능 가시화기

본 논문에서 제시하는 PnP 스타일의 성능 가시화기는 기존의 가시화기에서 제공하지 못했던 이식성 및 확장성이 탁월하도록 구현하기 위해 그림 1에서와 같이 층구조를 갖으며 서로 독립적으로 개발될 수 있도록 설계되었다.

1) 본 연구는 과학기술처 연구비 지원(과제코드: 초고속전산F-13)에 의하여 수행되었음.

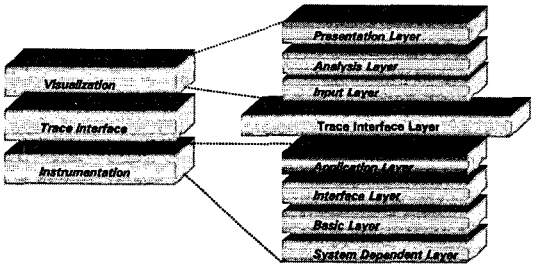

그림 1 : PnP 스타일 성능가시화기의 계층구조

2.1 인스트루멘테이션층

융통성, 이식성, 확장성 및 사용의 편리성을 설계 목표로 하여 설계된 인스트루멘테이션 층은 그림 2에서와 같이 MPI응용층, 인터페이스층, 라이브러리 기본층, 그리고 시스템 종속층의 4개의 층으로 구성된 계층구조를 갖도록 설계하였다. 인스트루멘테이션 층의 최상위 층을 구성하는 응용층은 MPI에 명세된 프로파일링 인터페이스에 따라 구현될 수 있도록 하며, 하위 세 개의 층은 사건 포획 라이브러리(Event Capture Library(ECL))로 구현하도록 설계하였다 [5]. 사실 ECL은 효율성 뿐 만 아니라, 이식성 및 확장성을 설계의 목표로 하였기 때문에 MPI 사건의 수집을 위한 인스트루멘테이션 뿐 만 아니라, 임의의 스프트웨어 사건 기반형 성능 분석 도구를 위한 인스트루멘테이션 라이브러리로서 사용되어 질 수 있다. 또한, MPI 사건의 Point-to-Point 통신에 관련한 25개의 사건을 정의하였다.

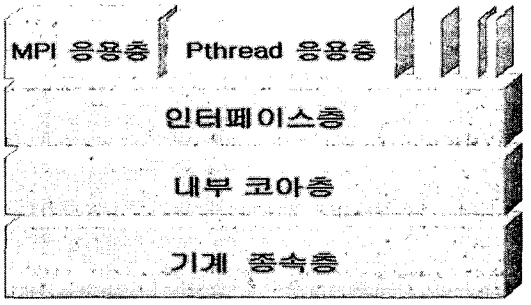

그림 2 : ECL(Event Capture Library)의 계층구조

2.1.1 인터페이스층(Interface Layer)

ECL의 최상위 계층에 위치하며 인스트루멘테이션 라이브러리의 모드 설정 및 사건을 생성시키기 위한 인터페이스 함수들을 사용자에게 제공한다.

2.1.2 내부코아층 (Basic Layer)

ECL의 중간 계층에 위치하며 사전 레코드의 수집을 위한 함수들을 정의하며 사용자의 엑세스가 불가능하다.

2.1.3 기계종속층 (System Dependent Layer)

플랫폼에 따라 기본층이 요구하는 정보를 제공하며 내부코아층과 같이 사용자의 엑세스가 불가능하다.

2.2 문제 중심의 추적 인터페이스층

EDL/JPAL (Event Description Language/Java Problem-oriented Access Library) 인터페이스는 메타 (meta) 양식인 논리적 추적 양식을 지원하며, 사건 레코드의 추상적인 구조와 추적 데이터에 대한 문제 중심의 엑세스 방법을 제공한다. EDL은 사건 기술언어(Event Descript Language)를 의미하고, JPAL은 문제중심의 추적 엑세스 라이브러리(JPAL Problem-oriented Access Library)를 의미한다. 메타 추적 양식은 의미(semantics)와는 다른 독립적으로 추적 레코드의 구조만을 명세하는 것이 특징이며, 사건의 유형에 관한 명세를 포함하지 않으므로 임의의 추적 양식을 기술할 수 있도록 해준다. EDL/JPAL 인터페이스를 사용하는 성능 분석 시스템은 추적 파일의 양식이 변경되거나, 새로운 레코드 양식이 추가되더라도 영향을 받지 않게 된다. 결과적으로, EDL/JPAL 인터페이스를 사용하는 성능 분석 시스템은 변환기를 사용하지 않고도 대부분의 추적 양식들을 지원할 수 있고, 특정한 추적양식과 독립적으로 개발될 수 있다는 장점을 가지게 된다. 그림 3은 EDL/JPAL 인터페이스를 사용하는 본 성능가화기의 사용 예를 보여준다.

2.2.1 사건 기술 언어 (EDL)

EDL은 추적레코드의 논리적 구조를 명세하며 추적 레코드의 표현을 명세한다. 또한 레코드의 실체와 레코드 설명자간의 사상(Mapping)을 명세한다. 그러나 추적레코드의 semantics는 명세하지 않는다. 이러한 정보를 가지고 EDL을 사용하여 추적 설명자를 생성한다. 이 추적 설명자는 다음의 세 개의 설명자로 이루어져 있다.

- ▶ **레코드 설명자** (Record Descriptor) : 논리적 레코드의 이름, 구조 및 표현을 명세 한다.
- ▶ **필드 설명자** (Field Descriptor) : 각 필드의 타입과 이름을 명세 한다.
- ▶ **바인드 설명자** (Bind Descriptor) : 레코드 설명자를 식별하기 위한 바인딩 정보를 명세 한다.

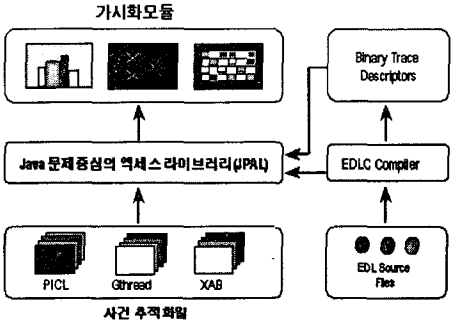

그림 3 : EDL/JPAL 추적 인터페이스

2.2.2 Java 문제중심의 엑세스 라이브러리 (JPAL)

JPAL(Java Problem-oriented trace Access Library)은 추적 파일의 데이터에 대하여 문제 중심의 엑세스 방법을 제공하기 위한 인터페이스 함수 라이브러리이다. 이 라이브러리는 SDDF[4] 접속 라이브러리와 유사하게 사건 레코드의 논리적 구조에 기초하였으며, POET[4]과 유사하게 문제 중심의 엑세스가 가능하도록 개발되었다. 그런데, JPAL을 통하여 추적 파일의 데이터를 문제 중심으로 엑세스하기 위해서는 앞에서 설명한 추적 설명자가 필요하다. 추적 설명자는 효율적인 처리를 위해 이진 축약 형태로 존재한다. 표 1은 JPAL을 구성하고 있는 Class를 보여준다.

Class Name	기능
Dictionary	추적양식을 기술하는 레코드설명자, 필드설명자, 바인드정보들에 대한 객체를 관리
RecordDescriptor	레코드설명자 객체를 생성하여 생성되는 필드설명자의 객체를 목록으로 관리
RecordDIterator	레코드설명자를 엑세스할 수 있는 방법을 제공
FieldDescriptor	필드설명자 객체를 생성하여 정보를 관리 유지
FieldDIterator	필드설명자를 엑세스할 수 있는 방법을 제공
BindInfo	바인더 설명자 객체를 생성하여 정보를 관리하고 제공한다
Reader	Dictionay를 사용하여 추적화일에 기록된 사건 레코드를 반환한다
Record	EDL Data에 대한 엑세스방법을 제공

표 1 : EDL/PAL 인터페이스 구성 CLASS와 기능

2.3 성능 가시화층

인스트루멘테이션층에서 수집된 각 프로세스별 추적 파일은 시간별로 정렬(sorting)되어 하나의 추적 파일을 이룬다. 이렇게 시간별로 정렬된 최종 추적 파일은 가시화층을 통하여 그림의 형태로 사용자에게 제공된다.

2.3.1 성능 가시화기 프레임워크 설계 및 구현

기존의 성능가시화기는 많은 문제점을 내포하고 있었다. 문제점으

로는 범용성의 결여 및 특정 플랫폼, 특정 응용 도메인 또는 특정 언어에 밀 결합 , 물리적 추적 양식의 융통성 결여 그리고 가시화 모듈과 인스트루멘테이션 모듈간의 밀 결합이 있었다. 또한 성능 뷰의 추가 및 수정의 어려움, 성능 뷰의 규모 확정성 결여, 성능 뷰와 프로그램 동작간의 사상 메카니즘의 결여, 다른 지원 도구들과의 연동 메카니즘의 부재가 있었다. 그러나 본 논문에서 소개하는 성능가시화기는 이러한 문제점을 해결하기 위해 이에 맞는 검증된 디자인 패턴을 이용해 설계 및 구현을 했다.

2.3.2 성능 가시화기층 Main Class Diagram

그림 4는 확장성 및 범용성을 고려해 설계된 Main Class Diagram을 나타낸다.

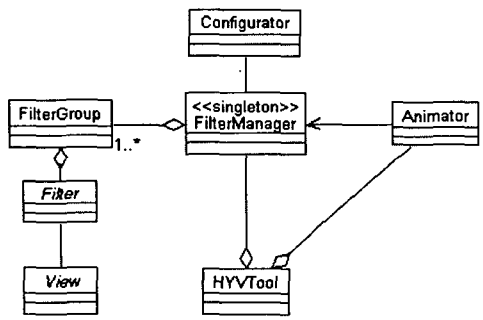

그림 4 : 성능가시화기 Main Class Diagram

본 Main Class Diagram을 이용한 성능가시화기의 구현은 인스트루멘테이션층과 독립성을 유지할 수 있으며(EDL/JPAL 이용) 임의의 사건레코드를 지원할 수 있다. 또한 임의의 응용 도메인을 위한 뷰 및 뷰 그룹의 동적 설정이 가능하고 뷰와 필터의 추가 및 수정이 용이한 Software-IC(Plug-and-Play)로 구현 될 수 있다.

2.4 PnP 스타일의 성능 가시화기 구현 결과

2.4.1 성능분석 시스템 구현환경

- ▶ OS : Solaris 2.5.1, Solaris 2.6, Windows95
- ▶H/W : 현대 Axil Ultra Spac 1K, Pentium MMX 200MHz
- ▶S/W : JDK 1.1.6, Swing 1.1.3, JLex1.2, CUP1.0, GCC 2.7.2

2.4.2 성능가시화기의 인터페이스와 성능뷰

그림 5는 사용자의 편리성을 최대로 고려하여 동작명령을 아이콘화 하여 구현한 성능가시화기 GUI를 나타낸다. 그리고 그림 6은 뷰 그룹 및 뷰의 선택 GUI로 사용자가 선택한 뷰 그룹에 의해 뷰의 구성내용이 바뀌게 된다. 마지막으로 그림 7은 성능가시화기의 실제 실행 상태를 나타내는 것으로 PICL 추적 파일을 3개의 뷰에서 가시화하고 있는 상태이다.

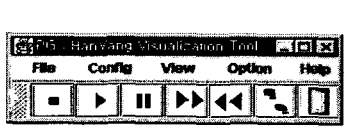

그림 5 : 성능가시화기 GUI

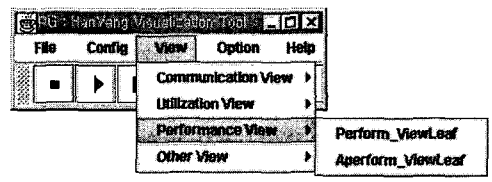

그림 6 : Select ViewGroup & View GUI

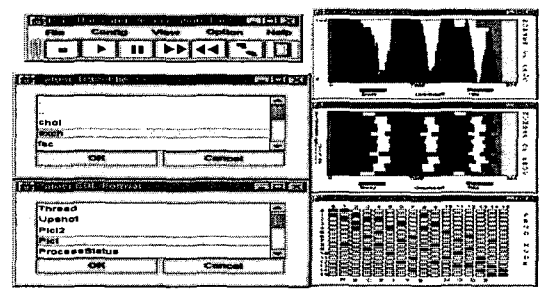

그림 7 : 성능가시화기 수행모습

3. 결론

본 논문에서 제시한 병렬프로그램을 위한 PnP 스타일 성능가시화기는 확장성 및 이식성을 위하여 층구조를 갖도록 하였다. 또한 사건포획을 위해 4개의 층구조를 갖는 인스트루멘테이션층을 설계 및 구현하였고 사건 레코드의 추상적인 구조와 추적 데이터에 대한 문제 중심의 엑세스 방법을 제공하기 위하여 EDL/JPAL를 이용하는 인터페이스층을 구현하였다. 마지막으로 가시화층은 애니메이터, HVYTool, 컨피그레이터, 필터관리자 등으로 구성된 확장성 있는 backbone 프레임워크로 구현되었고 뷰의 추가 및 수정이 용이하도록 Software-IC로 설계 및 구현되었다. 또한 Plug-and-Play 스타일의 설계 및 구현으로 인한 사용의 편리성, 규모 확장성, 범용성, 유용성을 제공하며 MPI 및 쓰래드 관련 뷰를 풍부하게 제공할 수 있었다. 향후과제로는 본 논문에서 제시한 가시화기에 back-tracking 기능과 텍스트-그래픽 사상(mapping) 메카니즘 같은 다양한 기능을 추가할 계획이다

[참고문헌]

[1] Message Passing Interface Forum, "The MPI message passing interface standard," Tech. Report, University of Tennessee Kenoxville, Aplril 1994. Available from http://www.msc.anl.gov/mpi/mpi-report.ps

[2] M. T. Heath and J. A. Etheridge. Visualizing performance of parallel programs. Technical Report ORNL/TM-11813, Oak Ridge National Laboratory, Oak Ridge, TN, May 1991.

[3] Michael T. Heath, Jennifer A. Etherridge, "Visualizing he performance of parallel programs," IEEE Software, ep. 1990, pp 29-39.

[4] M. H. Reilly, "A Performance Monitor for Parallel Programs ," Academic Press,Inc., 1990.

[5] 김정선, 진휴정, 김종렬, 문영식, "MPI 프로그램을 위한 사건 기반 성능 분석 시스템", 한국정보과학회 추계학술발표논문집 제24권 3호, pp.467-470, 1997