네트워크 컴퓨터를 위한 자바 기반의 성능감시기

(A Java-based Performance Monitor for Networked Computer)

김 봉 준 † 김 동 호 †† 황 석 찬 ††† 김 명 호 †††† 최 재 영 ††††

(Bongjun Kim) (Dongho Kim) (Seogchan Hwang) (Myungho Kim) (Jaeyoung Choi)

요 약 본 논문에서는 네트워크 컴퓨터를 이용하여 병렬 프로그래밍 환경에서 수행되는 프로그램의 성능을 추적하고 평가하기 위한 온라인/일괄처리-사건/시간 기반형 성능 감시기를 제안한다. 본 논문의 JaNeC 성능 감시기는 자바로 구현되어 있으므로 이기종 컴퓨터사이의 시스템 이식성이 뛰어나며, 웹 기반의 그래픽 콘솔을 제공하여 사용자에게 친숙한 인터페이스를 제공한다. 본 논문에서 제시한 성능 감시기는 사용자가 프로그램 실행시에 발생한 이벤트를 보다 쉽게 분석할 수 있도록 태스크나 이벤트를 선택할 수 있는 필터 기능과 TimeLine, Task View, Task History, Message Passing View, Host CPU View 기능 등으로 구성되어 있다.

Abstract In this paper, we present a performance monitor to trace and evaluate the performance of programs running on networked computers. The performance monitor of the JaNeC is online/batch as well as event/time driven. Since it is implemented with the Java programming language, it provides us with high portability among heterogeneous computer systems, and friendly graphical user interface. This performance monitor consists of various views such as "Task/Event Filter" and "TimeLine", "Task View", "Task Hoistory", "Message Passing View", "Host Cpu View", which allow the user to easily analyze event and time during the program execution.

1. 서 론

컴퓨터의 응용 분야가 넓어지면서 더욱 많은 계산량이 요구되었으며, 따라서 더욱 많은 컴퓨팅 파워가 필요하게 되었다. 값비싼 고성능의 수퍼 컴퓨터대신에 저렴한 가격으로 비슷한 성능을 얻기 위한 방법으로 다수의 프로세서를 사용하는 병렬컴퓨팅을 이용한 해결법을 찾기 시작하였다. 오늘날 네트워크 속도가 점차 빨라지고 컴퓨터가 고성능화되어가고 있기 때문에 일반 컴퓨터를 이용하여 분산 처리할 수 있는 네트워크 컴퓨터에 대한 연구가 활발하게 진행되고 있다.

이러한 시스템들을 위해 제공되는 PVM이나 MPI와 같은 통신 시스템 혹은 통신 라이브러리를 이용하여 작성된 병렬 프로그램은 보통 그 구조가 복잡하고 행동 양식을 정확하게 파악할 수 없기 때문에, 병렬 프로그램의 정확성과 성능을 분석하기는 매우 어렵다. 그래서 병렬 컴퓨터나 분산 처리 시스템과 병렬 프로그램의 성능을 쉽게 분석할 수 있도록 성능 감시기가 연구되어 왔다. 이전에 연구되고 개발된 성능 감시기들[1, 2, 3, 4, 7, 8, 10]은 그래픽 사용자 인터페이스 환경에서 사용자에게 그림으로 여러 기능을 제공함으로써 병렬 프로그램의 행동 양식을 보다 쉽게 파악할 수 있도록 구성되

· 본 연구는 1997-2000년 정보통신부의 대학기초 연구지원사업의 지원(AB-97-G-240)으로 수행되었습니다.

† 비 회 원 : 산업연구원 연구원
bjkim@ss.ssu.ac.kr
†† 학생회원 : 숭실대학교 컴퓨터학부
dhkim@ss.ssu.ac.kr
††† 비 회 원 : 숭실대학교 컴퓨터학부
schwnag@ss.soongsil.ac.kr
†††† 종신회원 : 숭실대학교 컴퓨터학부 교수
kmh@comp.soongsil.ac.kr
choi@computing.soongsil.ac.kr

논문접수 : 1998년 12월 7일
심사완료 : 1999년 10월 25일

었다. 그러나 대부분이 그래픽 사용자 인터페이스를 제공하고 있지만 기계에 종속적이라는 큰 단점을 가지고 있다. 즉 한 컴퓨터에서 개발한 프로그램을 기종이 다른 컴퓨터에서 수행시킬 수 없기 때문에 같은 프로그램을 기종이 다른 컴퓨터에 맞게 새로 개발해야 한다. 본 논문에서는 위의 문제를 해결하기 위하여 자바로 구현된 성능 감시기를 제안한다. 그 성능 감시기는 Java로 구현되므로 높은 이식성을 가지며 웹을 기반으로 작성하였으므로 친숙한 사용자 인터페이스를 갖는다는 장점을 가지고 있다.

네트워크 컴퓨터 환경은 그림 1과 같이 동기종 또는 이기종의 컴퓨터를 네트워크로 연결하여 가상적인 병렬 컴퓨팅 환경을 제공하여 주는 시스템이다. 각 컴퓨터에는 데몬이 상주하여 다른 컴퓨터의 데몬과 통신을 통하여 가상 병렬 컴퓨터를 구성한다. 그리고 각 컴퓨터의 데몬은 컴퓨팅 자원과 그 컴퓨터에서 실제 작업을 수행하는 태스크를 관리한다 [10].

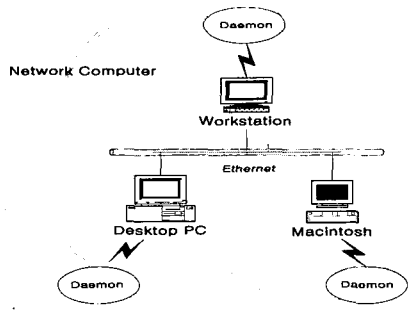

그림 1 네트워크 컴퓨터의 기본 환경

성능 감시기는 병렬 프로그래밍 환경에서 구현된 프로그램에 대해 그 행동 양식을 보다 정확하고 쉽게 이해할 수 있도록 하기 위한 프로그램이다. 성능 감시기는 구현 방법, 성능 측정용 데이타 수집시기, 결과 출력 방법에 따라 다음과 같이 분류될 수 있다.

1) 구현 방법에 따른 분류
① 소프트웨어 성능 감시기 - 소프트웨어로 구현
② 하드웨어 성능 감시기 - 하드웨어로 구현
③ 혼합형 성능 감시기 - 소프트웨어와 하드웨어를 혼합하여 구현
2) 수집 시기에 따른 분류
① 사건 기반형 성능 감시기 - 특정 사건이 발생하면 수집
② 시간 기반형 성능 감시기 - 일정한 시간이 지나면 수집
3) 출력 방법에 따른 분류
① 온라인 성능 감시기 - 프로그램이 수행되는 중에 결과를 입력받아서 그 결과를 분석하여 출력
② 일괄처리 성능 감시기 - 프로그램의 수행이 완료 후에 수집된 정보를 분석하여 출력

현재 외국에서 개발된 대표적인 성능 감시기로는 XPVM [1], ZM4/SIMPLE [2], Vampir [3], Pablo [4, 5, 11], XMPI [7], Paradigm [10], PVaniM [8] 등이 있다. 이것을 위의 분류에 따라 살펴보면 다음과 같다.

표 1 성능감시기의 분류

분 류	구현 방법에 따른 분류			수집시기에 따른 분류		출력방법에 따른 분류	
	소프트웨어	하드웨어	혼합형	사건기반형	시간기반형	온라인	일괄처리
XPVM	○			○		○	
ZM4/SIMPLE			○	○		○	○
Vampir	○			○	○		○
Pablo	○			○	○	○	○
XMPI	○			○		○	○
Paradigm	○			○		○	○
PVaniM	○			○		○	○

온라인 방식을 지원하는 성능 감시기에서는 XPVM과 Pablo가 우수하며 일괄처리 방식을 지원하는 성능 감시기에서는 Vampir가 뛰어나다. 그러나 사용자가 병렬 프로그램의 행동양식을 정확히 파악하고 분석할 수 있도록 하기 위해서 여러 가지 분석 기능들과 온라인/일괄처리 방식, 사건/시간기반 방식을 모두 제공할 수 있어야 한다. 또한 성능 감시기를 처음 사용하는 사용자가 보다 쉽게 이해하고 사용할 수 있어야 한다. 그런 점에서 XPVM은 온라인과 일괄처리를 모두 지원하지만 제공되는 기능들이 미비하다. Pablo는 온라인/일괄처리를 모두 지원하고 여러 기능들을 지원하지만 유용성과 편리성이 떨어지기 때문에 일반 사용자가 사용하기 어렵다. Vampir는 그 기능들이 뛰어나고 쉽게 사용할 수 있도록 구성되어 있다. 그러나, 보다 빠른 성능분석을 위해서는 온라인 방식을 필요로 하는데 Vampir는 일괄처리 방식만을 지원한다.

본 논문에서는 현재까지 개발된 성능 감시기의 단점들을 보완한 온라인/일괄처리-사건/시간기반형-소프트웨어 성능 감시기를 제시하였다. 또한 병렬 환경에서 사용자가 쉽고 간편하게 이용할 수 있는 사용자 그래픽 인터페이스와 자세한 도움말을 제공할 수 있는 성능 감시기를 제안한다.

2. JaNeC

JaNeC은 동종 또는 이종의 컴퓨터를 네트워크로 연결하여 가상의 병렬 컴퓨팅 환경을 제공하여 주는 시스템이다. 각 컴퓨터에는 데몬이 상주하여 다른 컴퓨터의 데몬과 통신을 통하여 가상 병렬 컴퓨터를 구성한다. 그리고 각 컴퓨터의 컴퓨팅 자원과 그 컴퓨터에서 실제 작업을 수행하는 태스크를 관리한다. JaNeC은 크게 두 부분으로 구성되는데, 가상 병렬 컴퓨터를 구성하는 시스템과 프로그래밍 인터페이스를 제공해주는 프로그래밍 라이브러리로 나눌 수 있다. 시스템은 각 컴퓨터에 상주하며 가상 머신을 구성하는 데몬과, 컴퓨터 내에서 작업을 수행하는 태스크, 그리고 데몬과 태스크의 통신을 이루는 메시지로 구성되어 있다. 라이브러리는 사용자가 병렬 프로그래밍을 할 수 있도록 제공하는 프로그래밍 라이브러리이다.

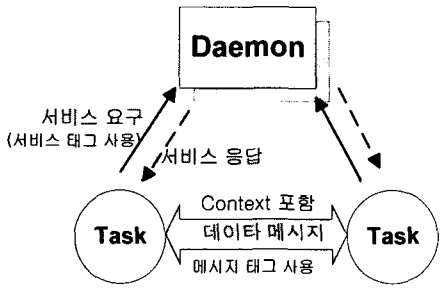

그림 2 JaNeC 시스템 구조

JaNeC 시스템의 각 컴퓨터 구조는 그림 2와 같이 데몬과 태스크로 구성되며, 유기적인 관계를 위해 메시지와 통신자를 포함한다. 데몬과 태스크는 실질적으로 컴퓨터 안에 존재하는 프로세스로서 각각 내부의 자료구조와 작업내용을 갖고 있으며, 메시지는 프로세스(데몬 또는 태스크)사이의 통신 수단으로서 데몬과 태스크는 서비스 태그를 사용한 서비스 메시지, 태스크간은 메시지 태그를 사용한 데이타 메시지를 주고받는다. 각 메시지는 통신의 안정성을 보장하는 Context를 포함하고 있다.

3. 콘 솔

JaNeC과 콘솔/성능 감시기의 관계는 그림 3과 같다. 프로그램이 실행되면 마스터 데몬이 프로그램에 따라 각 호스트에 데몬을 띄우고 프로그램을 실행한다. 마스터 데몬은 각 호스트의 데몬으로부터 보내오는 정보들을 Trace DB에 저장하였다가 저장된 정보를 웹브라우저를 통해서 콘솔에 보낸다.

콘솔은 사용자가 요구하는 각종 정보를 표시하는 윈도우와 실행 도중에 발생할 수 있는 각종 메시지를 보여주는 윈도우 등 두 가지 형태로 구성된다. 콘솔의 기능은 필요에 따라서 다른 컴퓨터를 추가하거나 삭제할 수 있는 호스트의 추가 및 삭제 기능, 사용자가 프로그래밍할 수 있는 편집기 기능, 편집된 파일을 컴파일하고 실행할 수 있는 환경을 제공한다. 또한 여러 개의 호스트를 연결하여 사용할 경우, 호스트의 성능과 작업량을 파악하여 효율적인 병렬 환경을 제공할 수 있는 기능을 제공한다.

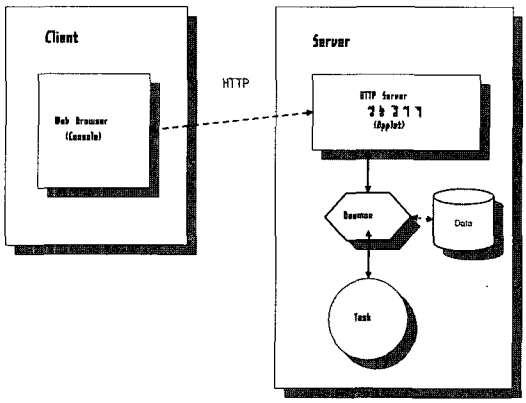

그림 3 JaNeC과 콘솔/성능 감시기

3.1 편집기

편집기는 사용자가 콘솔의 편집기 화면에서 병렬 프로그램을 작성하고 컴파일하여 실행시키거나, 저장된 파일을 입력으로 받아들여서 컴파일한 후 실행시킬 수 있도록 한다. 그래픽 인터페이스로 구성되어 있는 콘솔에서, 옵션의 선택에 따라서 파일의 입출력에 대한 처리가 다르게 된다. 옵션으로 일괄처리를 선택하면 컴파일 메뉴 부분이 비활성화되고 파일의 입출력은 추적 데이타가 저장된 파일을 입력으로 받아들여서 그 결과를 화면에 출력할 수 있도록 한다. 온라인을 선택하면 병렬 프로그램을 입력할 수 있는 편집기 화면이 나오고 컴파일시키거나 실행시킬 수 있도록 컴파일 메뉴부분이 활성

화된다. 이때 편집기는 JaNeC 라이브러리들에 대한 도움말을 보여주는 부분과 병렬 프로그램을 작성하거나 수정하기 위한 부분, 작성된 병렬 프로그램을 컴파일하였을 때 컴파일 에러를 출력하는 부분, 그리고 컴파일된 코드를 실행하였을 때 실행 에러를 출력하는 부분 등 네 부분으로 구성된다. 온라인이 선택되었을 때의 초기 화면은 그림 4와 같다.

병렬 프로그램을 작성하는 윈도우는 자바 프로그램 언어의 예약어들을 처리하여 다른 어절과 구분하는 기능과 JaNeC 라이브러리를 예약어나 다른 어절과 구분하는 기능을 가지고 파일을 화면에 출력하게 된다. 이 윈도우에서는 사용자가 편집기로부터 병렬 프로그램을 작성한 파일이나 지역 호스트로부터 작성되어 있는 파일을 입력으로 받아들여서 바이트 코드로 변환한다.

3.2 호스트 성능측정기

호스트 성능측정기는 각 호스트의 사용율과 성능을 비교·분석하여 향상된 병렬 환경을 제공한다. 병렬 환경에서 여러 개의 호스트를 가상적으로 연결하여 각 호스트의 성능을 파악한 후, 호스트의 성능에 맞도록 부하를 균등하게 배분함으로써 프로그램의 수행 속도를 향상시킬 수 있다. 호스트들의 성능에 따라 부하균등을 할 수 있도록 하는 호스트 성능측정기의 화면은 그림 5와 같다.

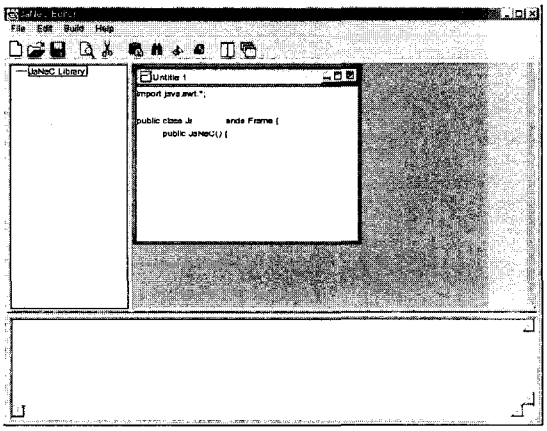

그림 4 JaNeC 편집기

사용자가 콘솔에서 병렬프로그램을 실행시킬 호스트를 추가하면 데몬은 호스트에 쓰레드를 발생시킨다. 발생된 쓰레드는 호스트로부터 CPU의 속도, 발생된 프로세스의 수, 현재 실행중인 프로세스의 수, CPU의 사용율, 호스트와 통신을 위한 네트워크 부하 등의 CPU 정보들을 받아들인다. 호스트 성능측정기는 위의 정보를 바탕으로 여러 호스트들의 절대적인 성능을 나타내고, 효과적인 작업 분배를 위한 호스트들간의 상대적인 성능을 나타내게 된다.

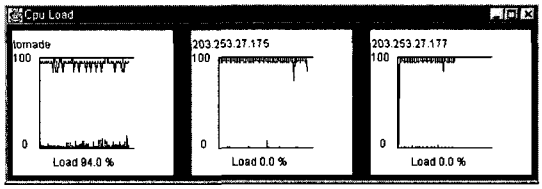

그림 5 호스트 성능측정기

절대적인 성능은 호스트가 CPU를 현재에 얼마만큼 사용하고 있는지를 표현한다. 그리고 상대적인 성능은 여러 호스트들간의 성능과 발생된 프로세스의 수, 현재 실행중인 프로세스의 수 등을 고려하여 호스트들간의 성능을 비교, 분석할 수 있도록 한다. 측정 방법으로는 각 컴퓨터의 OS에서 지원하는 명령어를 이용하여 커널에 대한 정보를 얻어서 분석을 한 후, 화면에 출력 한다.

4. 성능 감시기

그림 6에 콘솔과 성능 감시기의 구조가 나타나 있다. 콘솔에서 프로그램을 컴파일시키고 실행시키면, 마스터 데몬은 각 호스트나 태스크로부터 자료를 수집하여 성능 감시기 데몬에게 보낸다. 성능 감시기 데몬은 수집된 정보가 유효한지를 판단하고 SDDF (Self-Defining Data Format) 형식으로 바꾸어 준다. SDDF 형식에 관해서는 4.1절에서 자세히 다루겠다. SDDF로 변환된 데이타 중에서 사용자가 원하는 정보만을, 필터를 통하여 뽑아낸 뒤 그 결과를 출력 장치에 보낸다. 출력 장치는 데이타를 시각화하여 화면에 내보내게 된다.

성능 감시기는 콘솔에서 실행시킨 프로그램에 대해 그 행동 양식을 보다 정확하고 쉽게 이해할 수 있도록 하기 위한 소프트웨어이다. 그러므로 성능 감시기는 데몬으로부터 오는 정보가 유효한지 검사하는 기능과 추적 파일을 SDDF 형식로 바꾸어 임시 파일에 저장하는 기능, 그리고 저장된 정보의 분석을 용이하도록 사용자에게 원하는 태스크만을 보여주는 기능이 있어야 한다. 여러 태스크의 작업 중 원하는 이벤트만을 걸러서 보여주는 기능과 원하는 형태의 메시지만을 걸러서 보여주는 기능이 있어야 한다. 각 태스크 사이의 정보 교환 형식와 사건이 발생한 시간을 Nano Second까지 정밀하

게 보여주는 기능이 있어야 하며, 각 분석된 데이타를 그래픽 사용자 인터페이스를 사용하여 시각화하는 기능 등을 제공하여야 한다.

JaNeC 성능 감시기의 구성은 그림 6과 같다.

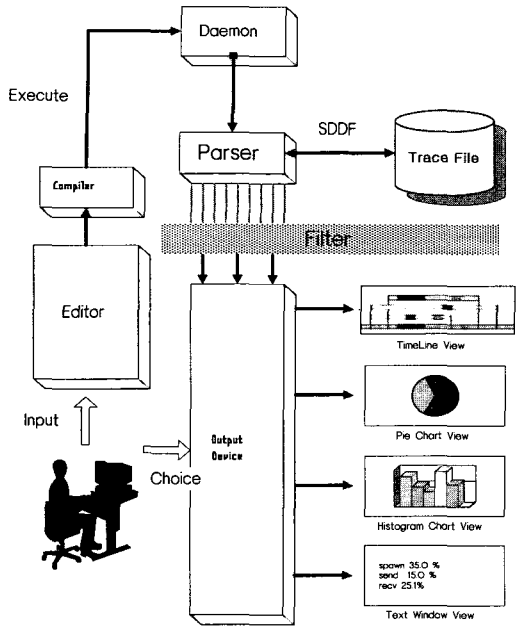

그림 6 콘솔과 성능 감시기의 구조

4.1 Parser

파서는 수집된 데이타를 분석하여 사건(Event)과 메시지 타입, 태스크에 따라 분류하여 저장한다. 콘솔에서 성능 감시기를 실행시키면 마스터 데몬은 각 호스트들과 태스크에서 정보를 수집하여 성능 감시기의 데몬에게 보낸다. 성능 감시기 데몬은 수집된 데이타의 유효성을 판단하고 SDDF 형식으로 정형화한다. 파서는 정형화된 데이타를 해석하여 발생한 사건과 각 태스크, 호스트에 따라 데이타를 분류하고 큐에 저장한다.

그림 7은 파서에 의해 파싱되는 과정에서 분류된 데이타가 저장되는 큐의 구조를 나타낸다. 분석된 데이타를 큐에 저장하는 이유는 데이타를 출력장치를 통해서 다시 출력시킬 경우 파일에서 다시 읽어들여 분석하는 과정을 줄여 실행 속도를 개선하는데 있다.

큐의 구조를 보면, 하나의 레코드를 기본 골자로 하여 링크드 리스트의 형태를 갖는다. 각 레코드에는 이름과 색상 정보, 레코드 정보가 저장된다. 성능 감시기가 실행되면 각 레코드들에 대한 색상이 지정되어 있다. 색상이 지정되어 있지 않은 레코드가 입력되면, 시스템에서 레코드에 색상을 지정한다. 또한 레코드의 색상은 사용자가 임의대로 지정하여 파일로 저장할 수 있다. Record Information 필드는 각 레코드마다 저장되는 필드의 길이가 서로 틀리기 때문에 Vector가 사용되며, 발생시간이나 태스크 번호, 메시지 타입, 자신의 호스트 이름, 이벤트가 완료한 시간 등과 같은 정보들이 저장된다.

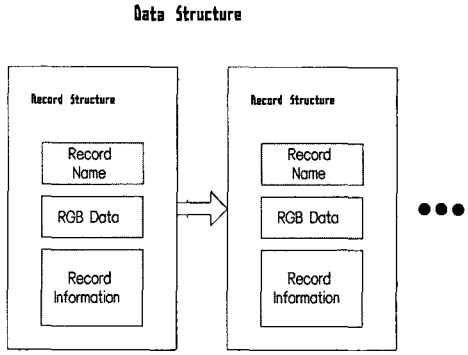

그림 7 큐의 구조

성능 감시기에서는 각 호스트나 태스크에 대해서 그 성능을 평가하기 위하여 추적 파일(trace file)이 필요하다. 현재까지는 이 추적 파일에 대한 표준이 나와 있지 않다. 추적 파일은 하나의 호스트에서 모아져서 로컬에서 그 정보를 분석하여 그 결과를 출력할 수도 있고, 다른 호스트에서도 그 정보를 이용하여 분석하고 그 결과를 출력할 수 있다. 따라서 이기종 시스템에서는 byte-order, word 길이, floating point 표현이 서로 다를 수 있다. 따라서 이기종 시스템에서도 추적 파일을 사용하여 똑같은 결과를 얻기 위해서는 독립적인 데이타 형식이 존재하여야 한다. SDDF는 독립적인 데이타 형식으로 이루어져 있어서 서로 다른 이기종의 컴퓨터에서 사용할 수 있으며 (*Portability*), 기존에 존재하는 이벤트 타입뿐만 아니라 필요에 따라서 사용자가 이벤트 타입을 등록하여 사용할 수 있다 (Extensibility). 추적 파일은 ASCII 형태이기 때문에 그 크기가 Megabyte 또는 Gigabyte로 커질 수 있다. SDDF는 ASCII 뿐만 아니라 binary 형태로 변환할 수 있기 때문에 그 크기를 줄일 수 있다. 위와 같은 장점으로 JaNeC 성능 감시기에서는 SDDF를 사용하였다. SDDF의 예제 형식이 그림 8에 나와 있다.

SDDF는 다음의 4 부분으로 구성되어 있다.

1) Command
2) Stream Attribute - 전체 파일에 대한 일반적인 정보
3) Record Structure - 레코드의 구조를 선언
4) Record Instance - 레코드 데이타가 기록

Record Structure는 Record Instance가 Record Structure에 맞게 선언되었는가를 검사하기 위해 필요한 부분이다. 또한 성능 감시기에 미리 선언되어 있지 않은 레코드를 본 성능 감시기에서 분석하여 사용할 수 있도록 확장하기 위하여 필요한 부분이다.

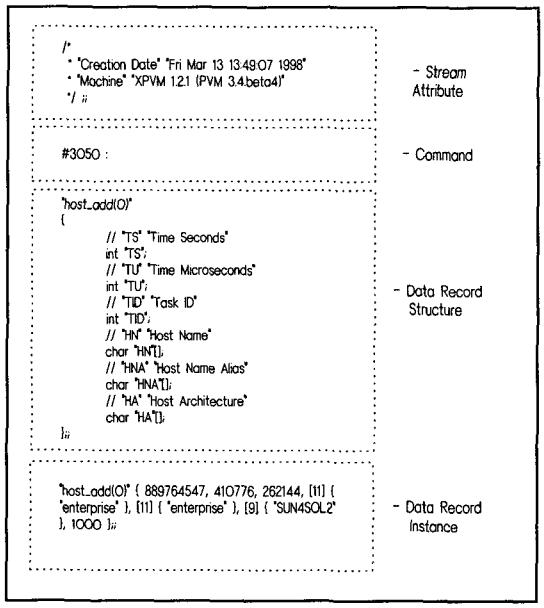

그림 8 SDDF

4.2 Filter

필터는 파서를 거쳐 나온 정보를 사용자가 원하는 정보만을 선택하도록 한다. 수많은 태스크와 호스트가 존재할 경우 보내오는 데이타의 양은 상당히 크고, 태스크나 호스트에서 발생한 사건들이 무한히 많을 경우 사용자는 화면에 출력된 결과를 분석하기가 매우 어렵다. 그러므로 사용자가 보다 자세하고 세밀하게 프로그램에 대한 성능을 분석하려면 필터에서 여러 기능들을 제공하여야 한다.

JaNeC에서는 다음과 같은 네 가지 종류의 필터를 제공한다.

(1) 호스트 필터 - 여러 호스트가 있을 때 한 호스트나 사용자가 원하는 호스트들에 대한 정보를 선택하여 출력할 수 있도록 한다. 여러 종류의 호스트를 연결하여 사용할 경우 특정 호스트에 대한 정보만을 분석할 경우 사용된다.

(2) 태스크 필터 - 여러 호스트들 중에서 사용자가 원하는 태스크들에 대한 정보를 선택하여 출력할 수 있도록 한다. 특정 태스크에 대하여 보다 자세하게 분석할 경우에 사용된다. 태스크에 대한 필터는 그림 9와 같다.

(3) 사건(Event) 필터 - 태스크들 중에서 불필요한 사건들은 제거하고 사용자가 필요로 하는 사건들에 대한 정보만을 선택하여 출력할 수 있도록 한다. 특정 사건만을 선택하여 사건들에 대한 자세한 정보를 얻기 위해 사용된다. 사건에 따른 필터는 그림 10과 같다.

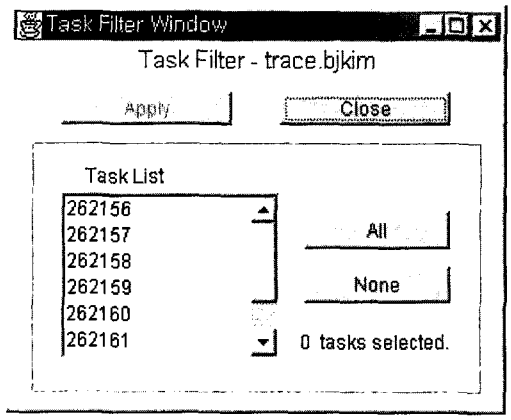

그림 9 태스크 필터

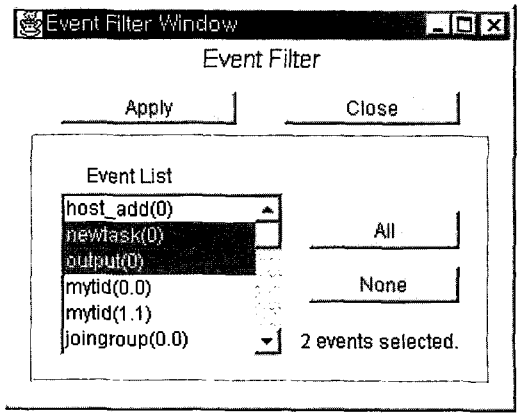

그림 10 이벤트 필터

(4) 메시지 교환 필터 - 태스크 사이에 서로 주고받는 메시지의 형태에 따라 원하는 형태의 메시지만을 선택하여 출력할 수 있도록 한다.

4.3 Output Device

SDDF 파서와 필터에 의해서 걸러진 데이타는 호스트, 태스크, 이벤트에 따라 분류되어 화면에 출력된다. 출력장치는 분석된 결과를 출력하여 사용자가 병렬 프로그램에 대한 성능과 행동 양식을 판단할 수 있도록 하는 부분이다. 출력 장치에서 출력되는 기능들을 보면 다음과 같다.

1) TimeLine - 각각의 태스크가 발생하는 시간에 따라 이벤트들이 화면에 출력된다. 각 이벤트 사이의 색상은 사용자가 지정할 수 있다. 만약 색상을 지정하지 않으면 디폴트 색상을 사용하게 된다.

TimeLine은 사용자에게 각 시간에 따라서 태스크의 상태를 직접 확인할 수 있도록 한다. 필요에 따라서 원하는 태스크의 원하는 시간에 해당하는 부분을 확대하여 보다 자세하게 확인할 수 있도록 한다. 본 논문의 성능 감시기의 최대 장점은 사용자가 보다 세밀하게 분석할 수 있도록 확대가 자유롭다는 것이다. 그림 11에 TimeLine View가 나타나 있다.

2) ChartView - 분석된 데이타를 입력으로 받아서, 일정한 시간을 간격으로 각 이벤트가 전체 실행시간 동안에 차지하는 비율을 원 그래프나 히스토그램으로 화면에 출력한다. Chart View는 전체의 프로그램 수행시간 동안에 발생되는 이벤트들의 비율을 쉽게 확인할 수 있고 전체의 Overhead와 Idle 시간에 대한 비율을 바로 확인할 수 있다. 어떠한 이벤트가 전체에서 차지하는 비율이 아무리 작아도 쉽게 확인할 수 있다. 이벤트가 전체에서 0.000000001%이내를 차지하면 Chart View에 나타난다. 화면상의 제약 때문에 원그래프나 히스토그램에 한번에 나타나는 이벤트의 수는 6개로 제한하였다. 차지하는 비율이 6개의 이벤트를 넘어간다면 간단한 마우스의 조작으로 전체에서 가장 많이 차지하는 이벤트를 빼고 나머지 부분을 출력할 수 있다. 본 논문에서 제시된 Chart View에는 Pie, Histogram View, Table View, Rotation Histograme View 등 4 가지이며 사용자가 임의의 Chart View 타입을 정할 수 있다.

그림 12에서는 각 태스크들에서 시간당 발생하는 이벤트의 비율을 원그래프로, 그림 13은 태스크중에서 하나를 선택하여 보다 자세하게 나타낸 것이다. 사용자가 Chart View 설정에 따라 그림 14와 같이 히스토그램으로 출력할 수 있다.

3) Task History - 추적 파일이 처리되고 분석된 결과가 텍스트 형태로 출력된다. Task History는 사용자가 원하는 태스크, 이벤트, 발생된 시간, 메시지 타입 등으로 데이타를 추적할 수 있도록 한다. 따라서 사용자가 어떠한 시간에 어떠한 이벤트가 발생하고, 발생된 이벤트로 인해 사용된 버퍼의 크기, 메시지의 형태 등을 보다 자세하게 분석할 수 있다. 그림 15는 Task History를 나타낸다.

4) Message Passing Line - 태스크와 태스크사이에 주고받는 정보를 선형으로 출력한다. 사용자에게 태스크와 태스크 사이에 메시지의 이동을 눈으로 쉽게 확인할 수 있다.

5) Text Window - 처음 시작부터 끝날 때까지 일어난 사건들의 비율을 텍스트의 형태로 출력한다.

6) Task View - 프로그램이 실행되고 있을 때 태스크의 현재의 상태를 쉽게 확인할 수 있다.

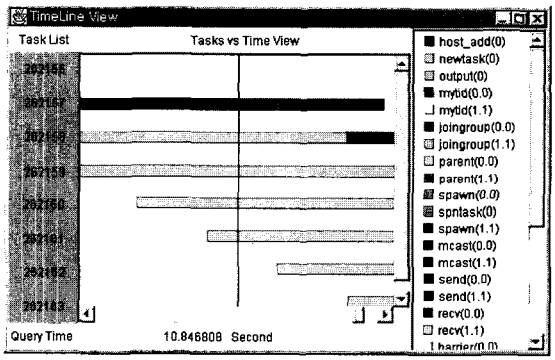

그림 11 부분 확대한 TimeLine view

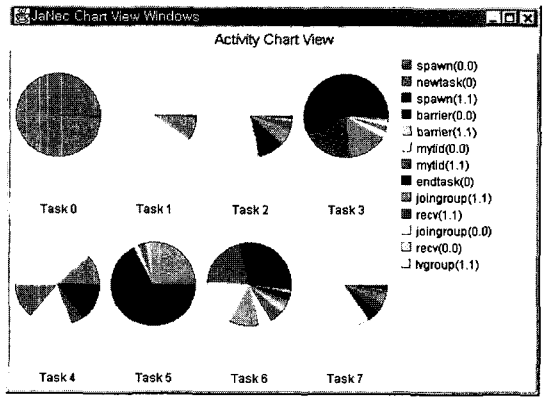

그림 12 원그래프

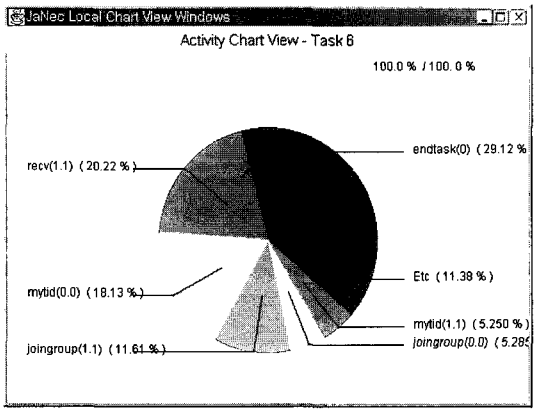

그림 13 부분 확대한 원그래프

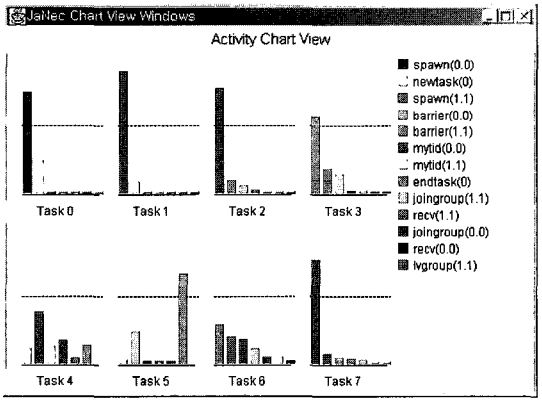

그림 14 Chart View (히스토그램)

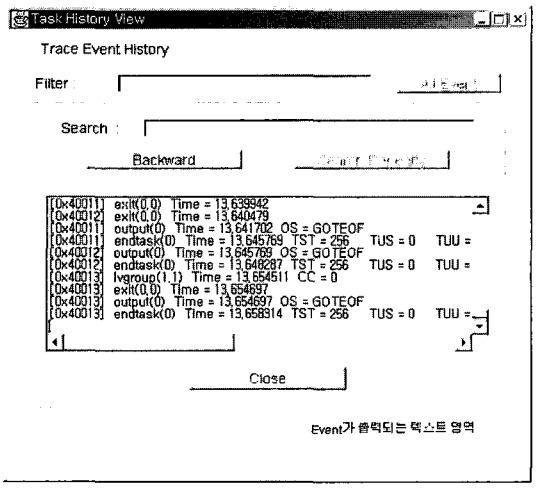

그림 15 Task History View

5. 결론과 향후과제

본 논문에서는 병렬 프로그램의 성능을 추적하고 평가하는 성능 감시기를 제시하였다. 성능 감시기는 웹을 기반으로 소프트웨어-온라인/일괄처리-사건/시간기반형으로 설계하였다. 현재까지 콘솔에서는 편집기와 컴파일러가 구현되어 있다. 성능 감시기는 파서와 필터 기능이, 출력 장치에서는 TimeLine View와 Chart View, Task History, Text Window 부분이 구현되었다.

향후에는 시스템의 부하가 균등한지를 사용자가 쉽게 확인할 수 있도록 Load Balancing View와 태스크사이의 메시지 이동 상태를 확인할 수 있는 Message Passing Line 부분, 태스크의 현재의 상태를 보여주는 Task Status View, 현재의 태스크가 어느 태스크와 데이타를 주고받고 있는지를 보여주는 Message Communication View를 추가할 것이다.

본 논문에서 구현한 성능감시기의 단점으로는, 웹을 기반으로 자바로 구현이 되었기 때문에 속도면에서 다른 성능감시기에 비하여 다소 떨어진다. 현재의 자바는 계속적으로 연구되고 있고 속도도 개선되고 있기 때문에 앞으로 어느 정도 보완될 것으로 예상된다.

참 고 문 헌

[1] James Arther Kohl, G.A.Geist, "XPVM 1.0 USER'S Guide," Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831.

[2] Bernd Mohr, "Standardization of Event Traces Considered harmful or Is an Implemetation of Object-Indepedent Event Trace Monitoring And Analysis System Possible ?," Univrsity Erlangen Nurnberg, IMMD 7, Martensstr. 3, D-8520 Erlangen. Germany.

[3] PALLAS Gmbh, "VAMPIR User's Manual Release 1.1 for VAMPIR Version 1.0," Hermulheimer StraBe 10, D-50321 Bruhl, Germany.

[4] Ruth A.Aydt, "An Informal Guide to Using Pablo," Department of Computer Science University of Illnois 61801.

[5] Luiz A.De Rose Ying Zhang, "SvPablo Guide," Pablo research Group, Department of Computer Science University of Illnois 61801.

[6] Ruth A.Aydt, "The Pablo Self-Defining Data Format," Department of Computer Science University of Illnois 61801, March 17, 1992.

[7] MPI and PVM User's Guide, http://techpubs.sgi.com/library/dynaweb.../MPI_UG/@Generic_BookText-View/924;td=6.

[8] Brad Topol, John Stasko, and Vaidy Sunderam,

"PVaniM 2.0-Online and Postmortom Visualization Support," November 10. 1995, http://www.cc.gat-ech.edu/gvu/softviz/parviz/pvanimOL/pvanimOL.html.
[9] 황석찬, 최재영, 김명호, "JPE:Java 병렬 프로그램밍 환경", 정보과학회 논문지(A). pp. 24-32, 제26권 제1호 1999.
[10] Jojn T.Stasko, "The Path-Transition Paradigm : A Practical Methodology for Adding Animation to Program Interfaces," College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-280.
[11] Shirley Browne, Jack Dongarra, and Levin London, "Review of Performance Analysis Tools for MPI Parallel Programs," http://www.cs.utk.edu/~ browne/perftools-review

김 봉 준
1997년 2월 숭실대학교 전자계산하과 학사. 1999년 8월 숭실대학교 컴퓨터학과 석사. 1999년 8월 ~ 현재 산업연구원 연구원. 관심분야는 병렬/분산처리, 시스템 소프트웨어, 네트워크 컴퓨팅, 운영체제, 시스템 성능 측정.

김 동 호
1998년 2월 숭실대학교 전자계산학과 학사. 1999년 현재 숭실대학교 컴퓨터학과 석사과정. 관심분야는 병렬/분산처리, 시스템 소프트웨어, 네트워크 컴퓨팅, 운영체제, 부하균등.

황 석 찬
1996년 청주대학교 전자계산학과 학사. 1998년 숭실대학교 전자계산학과 석사. 1998년 3월 ~ 현재 숭실대학교 전자계산학과 박사과정. 관심분야는 병렬 및 분산 컴퓨팅, 시스템 소프트웨어, 네트워크 컴퓨팅, 운영체제

김 명 호
1989년 숭실대학교 전자계산학과 학사. 1991년 포항공과대학교 전자계산학과 석사. 1995년 포항공과대학교 전자계산학과 박사. 1995년 2월 1995년 8월 한국전자통신연구소 선임연구원. 1995년 9월 ~ 현재 숭실대학교 정보과학대학 컴퓨터학부 전임강사. 관심분야는 병렬처리, 병렬알고리즘, 초고속계산론, 시스템 소프트웨어.

최 재 영
1984년 서울대학교 제어계측공학과 학사. 1986년 미국 남가주대학교 전기공학과 석사(컴퓨터공학). 1991년 미국 코넬대학교 전기공학부 박사(컴퓨터공학). 1992년 1월 ~ 1994년 2월 미국 국립 오크리지 연구소 연구원. 1994년 3월 ~ 1995년 2월 미국 테네시 주립대학교 연구교수. 1995년 3월 ~ 현재 숭실대학교 정보과학대학 컴퓨터학부 조교수. 관심분야는 병렬/분산처리, 병렬알고리즘, 초고속계산론, 시스템 소프트웨어