JaNeC을 위한 온라인 성능감시기의 설계 및 구현

김 명 호† · 김 남 훈†† · 최 재 영†

요 약

성능감시기는 분산처리 환경에서 프로그램의 성능을 추적하고 평가하기 위해 필수적인 것이다. 성능감시기는 출력방법에 따라 오프라인과 온라인으로 나눌 수 있다. 오프라인 성능감시기는 프로그램이 종료된 후에 그 성능을 분석하는 것이고, 온라인 성능감시기는 프로그램의 수행 중에 분석하는 것이다. 따라서 프로그램의 빠른 분석과 디버깅을 위해서는 온라인 기능이 필수적이다. JaNeC은 Java로 구현된 분산처리 환경으로 이를 위한 오프라인 성능감시기가 포함되어 있다. 그러나 이 성능감시기로는 JaNeC에서 수행되는 프로그램을 효율적으로 분석할 수 없다. 따라서, 본 논문에서는 JaNeC에서 수행되는 프로그램의 빠른 분석과 디버깅을 위해 온라인 성능감시기를 설계하고 구현하는 것에 대해서 설명한다. 이 온라인 성능감시기는 분석하고자하는 프로그램에 영향을 최소화하도록 설계되었으며, 효율적인 프로그램 분석을 위해 다양한 형태의 그래픽 출력을 제공한다. 또한 프로그램이 종료된 이후에도 다시 분석할 수 있도록 하기 위해 오프라인 성능감시기와의 인터페이스도 제공한다.

The Design and Implementation of On-Line Performance Monitor for JaNeC

Myung Ho Kim† · NamHoon Kim†† · Jaeyoung Choi†

ABSTRACT

A performance monitor is indispensable to trace and evaluate performance of a program under distributed processing environment. A performance monitor is classified as off-line and on-line according to its output method. An off-line performance monitor analyzes its performance after a program terminates, and an on-line performance monitor analyzes its one while a program runs. Therefore, the on-line function is essential to analyzing and debugging the program fast. JaNeC, distributed processing environment that is implemented in Java, contains an off-line performance monitor for this. However, this performance monitor may not analyze the program running on JaNeC efficiently. Consequently, this paper explains that an on-line performance monitor is designed and implemented for fast analysis and debugging of the program running on JaNeC. This on-line performance monitor is designed to minimize effects on a program to analyze, and provides various forms of graphic output, to analyze the program effectively. In addition, even after a program terminates, it provides interface with the off-line performance monitor, to analyze again.

키워드 : JaNeC(JaNeC), 분산처리 환경(Distributed processing environment), 성능감시기, 온라인 성능감시기(On-line performance monitor), 메시지 패싱(Message passing)

1. 서 론

네트워크 기술과 컴퓨터의 발달로 네트워크로 연결된 컴퓨터를 이용한 분산처리 환경에 많은 관심이 집중되고 있다. 이러한 분산 환경은 큰 컴퓨팅 파워를 저렴하게 얻을 수 있게 할뿐만 아니라, 대량의 작업을 빠르고 효율적으로 해결할 수 있게 한다. 그러나 이러한 분산처리 환경을 사용하기 위해서는 병렬프로그램을 작성해야 한다는 어려움이 있다. 병렬프로그램은 작성하기에도 용이하지 않지만, 병렬프로그램이 수행될때 여러 시스템에서 각 태스크가 독립적으로 수행되기 때문에 처리환경이나 수행과정 등이 매우 복잡하다는 것이다. 따라서 프로그램의 진행과정을 제대로 알기가 어렵고, 또 디버깅에 있어서도 많은 어려움이 있다. 이에 따라 병렬프로그램의 진행과정이나, 특성, 메시지 전달의 오류 등을 분석하기 위해 성능감시기에 대한 연구[1-7]가 진행되고 있다.

성능감시기는 보통 사용자의 편의를 위해 GUI 방식을 사용하고 있으며, 병렬프로그램의 수행과정과 함께 네트워크로 연결된 호스트들의 정보를 제공한다. 일반적으로 성능감

※ 본 논문은 2002년도 숭실대학교 교내연구비에 의해 연구되었음.
† 종신회원 : 숭실대학교 컴퓨터학부 교수
†† 정 회 원 : 온세통신 사원
논문접수 : 2002년 10월 7일, 심사완료 : 2002년 11월 21일

시기는 자체적으로 구성한 네트워크 환경을 기반으로 동작하는데, 구성된 네트워크 환경에서 병렬프로그램을 수행시키고, 그 네트워크 환경에 기반을 둔 성능감시기가 프로그램의 특성을 파악하고 호스트들간의 연결을 관리해서 사용자에게 정보를 제공해주는 방식으로 동작한다.

성능감시기는 구현방법과 사건(event)의 수집시기, 그리고 결과 출력방법에 따라 분류될 수 있다. 그 중에서 출력방법에 따라 분류하면 오프라인 성능감시기와 온라인 성능감시기로 나눌 수 있다. 오프라인 성능감시기는 프로그램이 수행하면서 발생하는 사건을 파일에 저장했다가 프로그램이 종료한 후 그 파일을 분석한다. 즉, 프로그램을 분석하기 위해서는 프로그램이 종료될 때까지 기다려야 한다. 따라서 프로그램 분석이 프로그램 수행 후로 미루어지고, 수행 중에 문제가 발생했는지도 즉시 알 수 없다는 단점이 있다.

온라인 성능감시기는 병렬프로그램과 같이 수행되며, 발생하는 사건들에 대한 정보를 실시간으로 수집하고 분석한다. 따라서 오프라인 성능감시기에 비해 보다 빠른 사건 분석이 가능하다. 일반적으로 병렬프로그램을 성능감시기로 분석하고자 할 경우에는 온라인 기능이 반드시 필요하다. 이는 사용자가 프로그램에 대해 빠른 분석을 필요로 하는 경우가 있고, 특히 수행시간이 긴 프로그램의 경우 프로그램의 오류를 수정하기 위한 디버깅작업도 보다 빠르게 진행할 수 있기 때문이다. 하지만 온라인 성능감시기는 오프라인 성능감시기보다 구현이 더 복잡하고 어렵다. 오프라인 성능감시기를 사용할 때에는 병렬 프로그램이 수행할 때 단지 사건 정보만 모으면 된다. 그러나 온라인 성능감시기는 병렬 태스크와 같이 수행하면서 태스크들이 발생하는 사건들을 실시간으로 모아서 처리해야 한다. 이로 인해 온라인 성능감시기의 수행이 병렬 태스크에 영향을 줄 수 있다. 즉, 사건들을 분석하기 위해서는 컴퓨팅 자원을 태스크와 공유해야 하고 사건의 전달을 위해서는 태스크들과 네트워크를 공유해야 한다. 이러한 자원 공유는 병렬 태스크의 수행에 영향을 미칠 수 있다. 따라서 온라인 성능감시기는 자원을 최소한으로 사용하도록 구현되어 병렬 태스크에 최소한의 영향만을 주도록 해야 한다.

여러 시스템을 네트워크로 연결하여 분산 처리하는 환경으로 JaNeC(JAva-based NEtwork Computer)이 있다. JaNeC은 메시지 패싱 기반의 통신을 제공하는 분산처리 환경이다. JaNeC 환경에서 동작하는 성능감시기[8]는 오프라인 성능감시기로서 프로그램이 종료된 후에 성능감시기가 사건을 분석한다. 따라서 프로그램에 대한 빠른 분석이 불가능하여 오류를 바로 확인할 수 없다. 그러므로 JaNeC 기반 성능감시기에도 프로그램의 빠른 분석을 위해서 온라인 기능이 필요하다.

따라서, 본 논문에서는 JaNeC에서 수행되는 병렬프로그램의 보다 빠른 분석을 위한 온라인 성능감시기의 설계와 구현에 대해 설명한다. 이 온라인 성능감시기는 병렬 태스크에 최소한의 영향만 주며, 다양한 형태로 병렬 프로그램을 분석할 수 있게 한다.

논문의 구성은 다음과 같다. 2장에서는 기존에 연구되었던 성능감시기들에 대해 살펴보고, 3장에서는 본 논문에서 제시하는 온라인 성능감시기의 기반이 될 JaNeC 환경과 JaNeC 기반 성능감시기에 대해 설명한다. 4장과 5장에서는 온라인 성능감시기의 설계와 구현에 대해서 설명하고, 6장에서는 온라인 성능감시기의 실험에 대해 설명하고, 7장에서 결론을 맺는다.

2. 기존 연구

병렬프로그램의 진행과정, 특성 등을 파악하고자 성능감시기의 연구가 많이 진행되고 있다. 이들 성능감시기들은 자체적으로 구성한 네트워크 환경을 기반으로 동작하고 있으며, 사용이 편리하도록 시각화 방법의 인터페이스를 제공한다. 또한 일부 성능감시기는 온라인 기능을 가지고 있어 병렬프로그램이 수행하는 동안 사건 분석을 할 수 있어 신속한 오류 수정도 가능하다.

2.1 Pablo

Pablo[1]는 일리노이즈 대학교의 Pablo 그룹에서 개발한 온라인을 지원하는 분석 소프트웨어이다. Pablo는 병렬프로그램의 추적, 추적파일의 분석 등의 부분으로 되어 있는데, 각 요소들간의 상호 작용은 Pablo Self-Defining Data Format(SDDF)[9]를 이용한다. 또한 분석도구에 이용할 추적파일로도 SDDF를 이용한다.

Pablo 분석 GUI는 SDDF 레코드들을 처리하는 데이터 전송 모듈이다. 분석 GUI는 성능데이터들간의 상호 연결을 시각적으로 나타낸다. 시각적인 표현을 위해서 분석 GUI는 네 가지 주요 모듈을 제공하는데, 데이터 분석, 데이터 표현, 추적파일 입력, 추적파일 출력이 그것이다. SDDF 레코드는 이 네 가지 모듈을 파이프처럼 거치면서 시각적 데이터로 나타나게 된다.

Pablo 분석 GUI는 프로그램의 디버깅 작업을 할수 없게 구성되어 있다. 따라서 소스 코드를 보기 위해서는 SvPablo[10]라는 컴포넌트가 필요하다. SvPablo는 Source View Pablo를 뜻하며, SvPablo 추적 라이브러리를 이용하면 소스 코드를 볼 수 있는 화면이 나타난다. 각 프로세스 당 추적파일을 수집하고 연결하는데는 SvPablo Combine 유틸리티가 담당하고, 통합된 성능데이터는 SvPablo GUI에 의해 분석

된다.

SvPablo는 사용이 편하다는 장점이 있다. 사용자는 통계 데이터와 소스코드를 동시에 보면서 코드를 수정할 수 있다. 또한 Pablo의 각 컴포넌트들은 병렬 플랫폼과 프로그래밍 언어에 널리 사용될 수 있다. 다만, 이 도구도 몇몇 시스템에만 한정되어 사용된다는 단점이 있다.

2.2 Paradyn

Paradyn[2]은 위스콘신 대학교에서 개발한 성능측정 도구로서 온라인을 지원하는 분석기이다. 이 분석기는 대형의 병렬 분산 시스템에서 긴 수행시간을 갖는 프로그램의 분석에 사용된다. 온라인 분석에서는 CPU 시간, 블록 시간, 메시지 비율, I/O 비율 등을 분석할 수 있으며, 이를 위해서 페이즈(phase)라는 개념의 고정 크기 데이터를 사용한다. 페이즈는 전체 수행 기간을 나타내는 전역 페이즈와 일정 기간만을 나타내는 지역 페이즈가 있다. 또한 사용자는 프로그램 수행 도중에 병목 현상이 나타나면 자동으로 수행되는 성능 설명기라는 인터페이스를 이용하여 병목현상을 쉽게 발견할 수 있다.

Paradyn은 사용자에게 편리한 인터페이스를 제공하는 분석도구로서 사용자는 여러 가지 그래픽 인터페이스를 통해서 프로그램을 분석할 수 있다. 하지만, 일부의 시스템에서만 동작하도록 설계되었다는 단점이 있다.

2.3 PVaniM

PVaniM[3]은 PVM을 이용한 통신 위주의 프로그램에서 기본적인 I/O 정보 등을 온라인과 오프라인 방법의 시각적 분석으로 제공하는 분석도구이다. 온라인 분석에는 메시지 통신 패턴, 메시지 양, 호스트 이용율, 메모리 이용율등의 정보를 보여준다. 이를 위해서 이 분석 도구는 각 시스템에서 발생하는 데이터를 샘플링하는 기법을 이용한다. 오프라인 분석에서는 버퍼를 이용한 추적 데이터를 분석 윈도우에서 그래픽으로 나타낸다.

PVaniM은 다른 성능감시기와 마찬가지로 그래픽 기반의 인터페이스를 제공하여 분석이 편하지만, 사용이 어렵다는 단점이 있다. 특히 PVM 라이브러리를 이용한 프로그램만 분석이 가능한 한계가 있다.

2.4 그 외의 성능감시기들

위에서 열거한 성능감시기들 외에 AIMS[4], XPVM[5], Vampir[6], XMPI[7], 등의 성능감시기들이 있다. 이들 성능감시기들도 다른 것과 마찬가지로 사용자에게 사용이 편리하도록 그래픽 기반의 인터페이스를 제공한다.

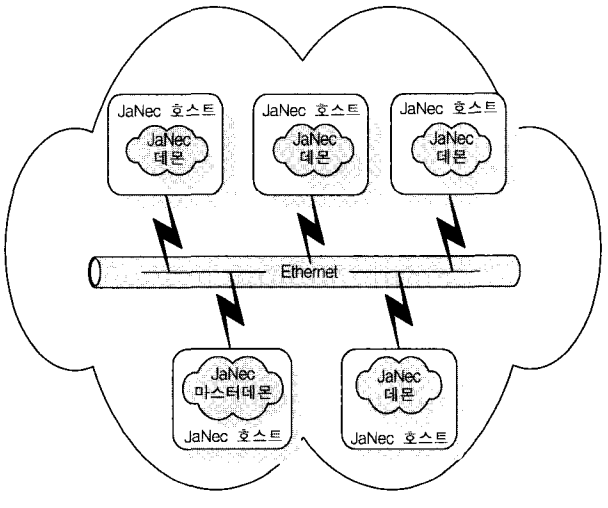
AIMS는 오프라인으로서 추적데이터를 이용한 프로그램의 분석이 가능하다. 또한 강력한 에러메시지를 통해서 사용자가 쉽게 소스코드를 수정할 수 있게 해준다. XPVM은 원거리에 있는 태스크에 대해서 간단하게 디버깅을 할 수 있는 기능을 가지고 있다. Vampir는 상업용으로 만들어진 소프트웨어로서 임의의 시간 간격으로 성능을 분석할 수도 있다. XMPI는 MPI프로그램의 성능분석에 필요한 디버깅 등의 기능을 제공하는 도구이다.

위에 제시한 성능감시기중의 일부는 온라인 기능을 지원하지만, 각각이 동작하는 분산처리 환경과 밀접한 관련이 있기 때문에 JaNeC 환경에서는 적용할 수 없다. 따라서 JaNeC을 위한 온라인 성능감시기의 설계 및 구현이 필요하다.

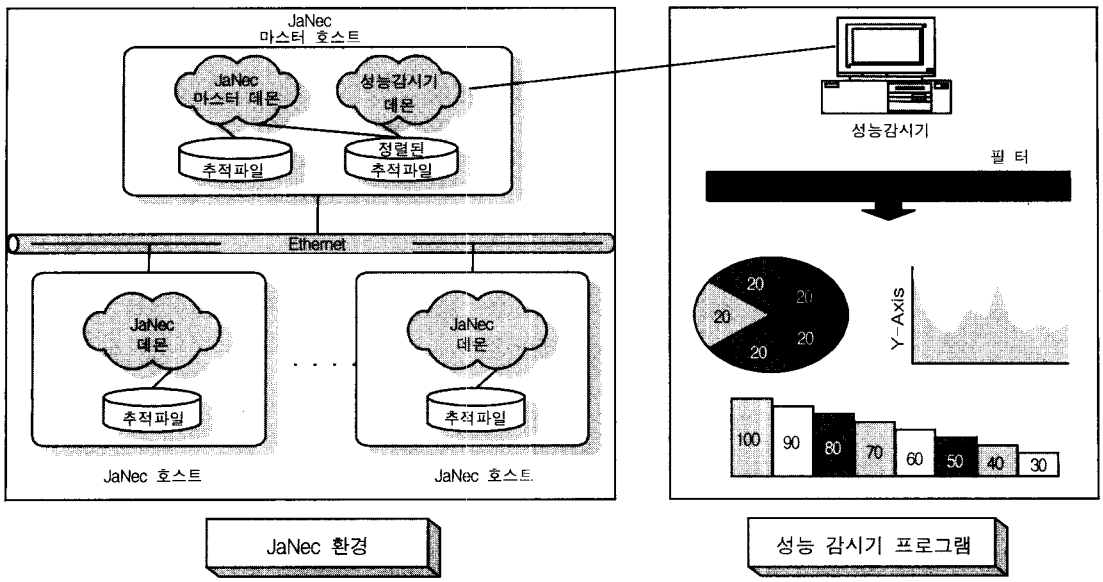
3. JaNeC과 JaNeC 기반 성능감시기

3.1 JaNeC

JaNeC은 동종 또는 이기종 컴퓨터를 네트워크로 연결하여 가상의 병렬 컴퓨팅 환경을 만들어 주는 자바로 구현된 시스템이다. JaNeC은 가상 병렬 컴퓨터를 구성하는 시스템과 프로그래밍 인터페이스를 제공해주는 프로그래밍 라이브러리로 나눌 수 있다. JaNeC 프로그래밍 라이브러리는 MPI를 기반으로 하고 있으며, 각 응용 병렬 프로그램은 이 라이브러리를 사용하여 작성된다. JaNeC 시스템은 가상 병렬 컴퓨팅 환경을 이루는 각 컴퓨터에 상주하며 가상 컴퓨팅 환경을 구성하는 JaNeC 데몬과 컴퓨터 내에서 작업을 수행하는 태스크, 그리고 이들간의 통신을 이루는 메시지로 구성된다. (그림 1)은 JaNeC 환경의 개념도이다. 그림에서 JaNeC 마스터 데몬은 전체 JaNeC을 관리하는 데몬이다. JaNeC은 자바로 구현되어 있고 MPI를 기반으로 하고 있어 이 식성이 뛰어나고 사용하기 쉽다는 장점이 있다.



(그림 1) JaNeC 환경 개념도



(그림 2) JaNeC 환경과 성능감시기 프로그램

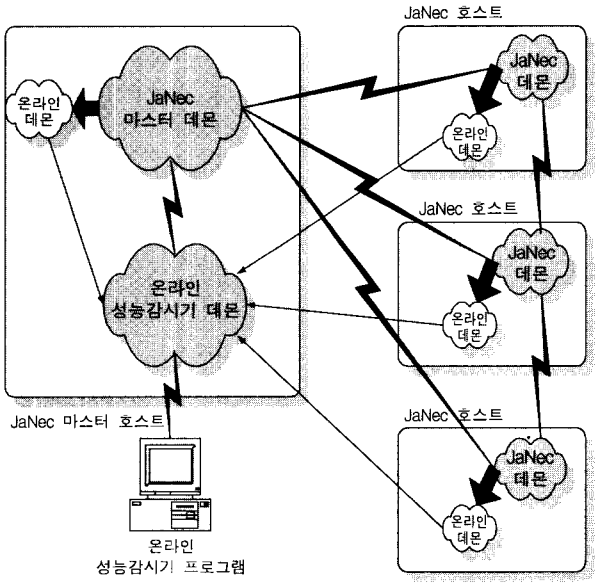
3.2 JaNeC 오프라인 성능감시기

JaNeC 오프라인 성능감시기는 JaNeC 환경에서 수행되는 병렬프로그램에 대해 그 행동 양식을 보다 정확하고 쉽게 이해할 수 있도록 하기 위한 소프트웨어이다[8]. 이 성능감시기는 성능감시기 데몬과 성능감시기 프로그램으로 구성된다. 성능감시기 데몬은 JaNeC 마스터 데몬이 위치하는 호스트에서 동작하고, 성능감시기 프로그램은 네트워크에 연결된 임의의 컴퓨터에서 동작한다. (그림 2)는 JaNeC 환경과 성능감시기를 같이 나타낸 것이다.

메시지 패싱 기반의 환경에서 주요 사건으로는 프로그램의 시작과 끝, 그리고 태스크간에 주고받은 메시지를 들 수 있다. 메시지와 관련된 사건은 송신 호스트와 태스크, 수신 호스트와 태스크, 프로그램 시작으로부터의 상대적인 시간, 메시지(데이터) 량으로 정의된다. 병렬프로그램의 수행 중에 발생한 메시지와 관련된 사건들을 잘 분석하면 병렬프로그램이 잘 작성됐는지 또는 문제가 있는지를 알 수 있다. JaNeC 데몬은 병렬프로그램이 수행되면 병렬프로그램을 분석할 수 있도록 하기위해 태스크들이 수행하면서 발생하는 사건들을 추적파일에 저장한다. JaNeC 오프라인 성능감시기는 이렇게 저장된 내용을 쉽게 분석할 수 있도록 하기 위해 추적파일을 가공한 후에 다양한 형태로 보여주는 기능을 가지고 있다. 즉, 프로그램이 종료되면 각 호스트에 저장된 추적파일의 내용이 성능감시기 데몬에게 보내지고, 성능감시기 데몬은 그 내용을 분석한 후 성능감시기 프로그램에 전달한다. 성능감시기 프로그램은 성능감시기 데몬에게서 전달된 데이터를 다양한 형태로 보여주며, 사용자는 이것을 통해 병렬프로그램을 분석하게 된다.

4. JaNeC을 위한 온라인 성능감시기의 설계

온라인 성능감시기는 오프라인 성능감시기와는 달리 병렬프로그램이 수행 중에 발생하는 사건을 실시간으로 처리해야 한다. 따라서 오프라인 성능감시기와는 다른 구조를 가져야 한다. JaNeC 온라인 성능감시기를 설계하기 위해서는 다음과 같은 몇 가지 사항을 고려해야 한다. 첫째, 온라인 성능감시기는 병렬프로그램이 생성하는 사건을 실시간으로 성능감시기 프로그램에 전달해야 하기 때문에 JaNeC 데몬이 생성하는 사건을 받아 실시간으로 전달하는 기능이 필요하다. 둘째, 성능감시기 프로그램에서 여러 JaNeC 데몬으로부터의 사건들을 보여주기 위해서는 동시에 발생하는 여러



(그림 3) 온라인 성능감시기의 데몬 구성

사건들을 순서화하는 기능이 필요하다. 셋째, 오프라인 성능감시기는 수집된 사건을 한꺼번에 모아 정리한 후에 보여주면 되기 때문에 통계치나 그래프 등을 정적으로 만들면 되지만, 온라인 성능감시기는 실시간으로 내용들이 변하기때문에 동적인 기능을 갖는 통계자료나 그래프 기능이 필요하다. 이러한 기능을 만족하는 온라인 성능감시기를 (그림 3)과 같은 설계하였다. 그림에서 온라인 데몬은 JaNeC 데몬이 발생하는 사건들을 받아 실시간으로 전달하는 기능을 수행하는 것이고, 온라인 성능감시기 데몬은 온라인 데몬이 전달하는 여러 사건들을 순서화하는 기능을 수행한다. 마지막으로 온라인 성능감시기 프로그램은 동적인 정보를 보여주는 기능을 가지고 있다.

다음 장에서는 이렇게 설계된 온라인 성능감시기를 어떻게 구현했는지에 대해서 설명한다.

5. JaNeC을 위한 온라인 성능감시기의 구현

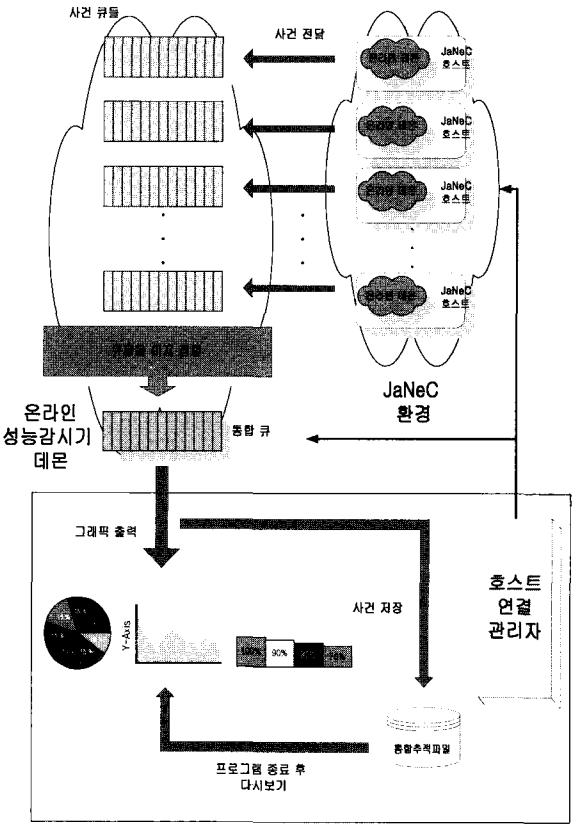
본 장에서는 온라인 성능감시기를 구현한 것에 대해 설명한다. 온라인 성능감시기를 구현할 때에는 다음과 같은 몇 가지 사항을 고려해야 한다. 첫째, 온라인 성능감시기가 사용자 태스크의 수행을 방해해서는 안 된다. 둘째, 온라인 성능감시기는 실시간으로 수집한 정보를 보여줘야 하기 때문에 네트워크 자원을 사용해야 하는데, 전달되는 정보를 최소화하여 네트워크 자원을 최소한으로 사용하도록 해야 한다. 셋째, 온라인으로 사건을 수집하다 보면 실제로 발생한 사건 순서와는 다르게 사건이 수집될 수 있다. 따라서 이러한 것을 확인하고 올바른 순서로 출력해 주어야 한다. 마지막으로, 오프라인 성능감시기는 수집된 사건을 한꺼번에 모아 정리한 후에 보여주면 되기 때문에 통계치나 그래프 등을 정적으로 만들면 되지만, 온라인 성능감시기는 실시간으로 내용들이 변하기 때문에 동적인 기능을 갖는 통계자료나 그래프를 만들어야 한다.

5.1 온라인 성능감시기 전체 구성

온라인 성능감시기는 크게 세 부분으로 나눌 수 있다 : 온라인 성능감시기 프로그램, 온라인 성능감시기 데몬, 온라인 데몬. 온라인 성능감시기 프로그램은 온라인 성능감시기 전체를 관리하고 병렬 프로그램을 분석할 수 있게 하는 것이고, 온라인 데몬은 태스크들간의 메시지 교환으로 인해 발생하는 사건들을 온라인 성능감시기 데몬으로 전달하는 기능을 수행하며, 온라인 성능감시기 데몬은 온라인 데몬으로부터 받은 사건을 정렬하여 온라인 성능감시기 프로그램에 전달하는 역할을 수행한다. 이 세 가지를 자세히 그리면 (그림 4)와 같다. 그림에서 오른쪽 부분은 태스크들이 수행되는 호스트에 JaNeC 데몬과 온라인 데몬이 있는 것을 보여주고 있고, 왼쪽 부분은 온라인 성능감시기 데몬을 보여주고 있다. 그림의 하단에는 온라인 성능감시기 프로그램을 보여주고 있다.

온라인 성능감시기 프로그램은 태스크들이 수행 중 발생하는 사건들을 온라인 성능감시기 데몬으로부터 받아서 그래프 형태로 보여주는 기능을 수행하고, 이렇게 보여주는 동시에 사건들을 파일에 저장하여 프로그램의 종료 후에도 다시 분석할 수 있게 한다. 또한 온라인 성능감시기 프로그램은 전체 온라인 성능감시기를 관리하기 위해 호스트 연결 관리자를 포함하고 있다. 즉, 호스트 연결 관리자에서는 온라인 데몬과 JaNeC 데몬간의 연결을 관리하여, 온라인 데몬이 사건들을 실시간으로 온라인 성능감시기 데몬에 보낼 수 있게 한다.

온라인 데몬은 JaNeC 데몬이 발생하는 사건을 실시간으로 받아서 온라인 성능감시기 데몬에 보내주고, 온라인 성능감시기 데몬은 각 온라인 데몬에게서 전달되어진 사건을 해당 큐에 저장하고, 이를 통합한 후 온라인 성능감시기 프로그램에게 전달한다. 이제 각각의 구현에 대해 자세히 설명할 것이다.



(그림 4) 온라인 성능감시기의 전체 구성도

5.2 온라인 데몬

온라인 데몬은 JaNeC 데몬이 병렬 태스크들간의 메시지 교환을 처리하면서 발생하는 사건 정보를 실시간으로 온라인 성능감시기 데몬에 보내는 기능을 수행한다. JaNeC 데몬은 기본적으로 자신이 처리한 내용을 로그 형태로 추적 파일에 저장한다. 오프라인 성능감시기는 프로그램이 종료한 후에 이 추적파일을 성능감시기 데몬에 보내서 프로그램을 분석할 수 있게 한다.

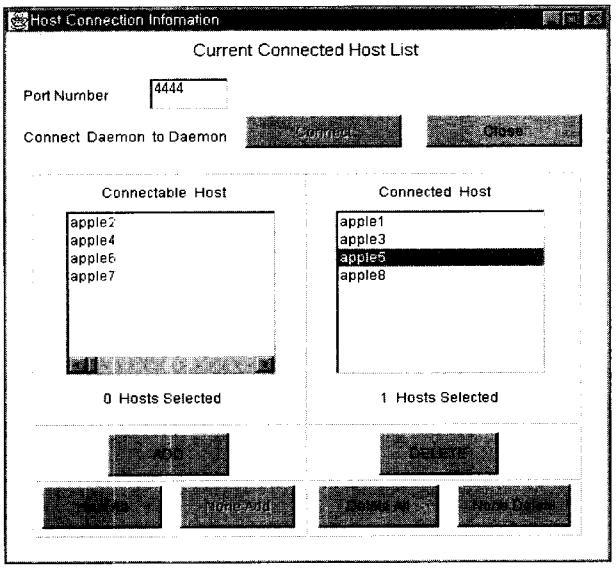
온라인 데몬을 구현하는 방법은 두 가지가 있을 수 있다. 첫 번째 방법은 온라인 데몬이 JaNeC 데몬과 직접적인 통신으로 사건을 받아 전달하는 방법이고, 두 번째 방법은 온라인 데몬이 추적파일의 내용을 읽어서 보내는 방법이다. 여기에서는 두 번째 방법을 선택하였다. 그 이유는 온라인 성능감시기의 수행이 병렬 프로그램에 영향을 주지 말아야 되는데, 첫 번째 방법을 사용하면 JaNeC 데몬과 온라인 데몬간의 통신으로 인해 태스크간의 통신에 영향을 줄 수 있기 때문이다. 따라서 온라인 데몬은 JaNeC 데몬이 작성하는 추적파일을 사용하여 사건을 온라인 성능감시기 데몬에 보내도록 구현하였다. 즉, 온라인 데몬은 추적파일을 감시하고 있다가 새로운 사건이 발행하면 온라인 성능감시기 데몬에게 그 내용을 보내게 된다.

5.3 온라인 성능감시기 데몬

온라인 성능감시기 데몬은 각 호스트상의 온라인 데몬들이 전해준 사건들을 온라인 성능감시기 프로그램에게 전달한다. 온라인 성능감시기 프로그램에게 사건들을 보낼 때에는 여러 호스트의 온라인 데몬으로부터 온 사건들의 상관관계를 보고 보내게 된다. 즉, 서로 다른 호스트에서 수행 중인 태스크간에 메시지 교환이 있었다면, 한 메시지의 주고받음에 의해 온라인 성능감시기 데몬은 두 개의 온라인 데몬으로부터 두 개의 사건을 받게 된다. 그리고 이 두 태스크가 서로 또 다른 태스크와의 메시지 교환이 있었다면 이에 관한 사건을 또 받게 된다. 그러나 이렇게 발생되는 사건들은 네트워크의 지연으로 인해 사건 순서대로 온라인 성능감시기 데몬에 들어온다는 보장이 없다. 따라서 각 사건의 상관관계를 따져서 온라인 데몬으로부터 들어오는 사건들은 적절히 재정렬할 필요가 있다. 이를 위해 온라인 성능감시기 데몬은 각 온라인 데몬으로부터 들어오는 사건들을 독립적인 큐에 저장하고, 각 큐의 내용을 분석한 후에 하나의 사건 흐름을 만들게 된다. 이렇게 재정렬된 사건은 온라인 성능감시기 데몬이 유지하는 통합 큐에 저장되게 된다. 온라인 성능감시기 프로그램으로 전달하는 데이터는 통합 큐에 저장된 것이 전달되고, 이러한 방법으로 병렬프로그램 수행 시에 발생되는 사건들은 순서를 유지하면서 온라인 성능감시기 프로그램에 전달된다.

5.4 온라인 성능감시기 프로그램

온라인 성능감시기 프로그램은 온라인 성능감시기 전체를 관리하는 기능과 결과를 보는 기능으로 나누어진다. 온라인 성능감시기 관리는 호스트 연결 관리자에서 수행된다. 즉, 온라인으로 성능감시기를 하기 위해서는 온라인 성능감시기 데몬과 JaNeC 마스터 데몬, JaNeC 마스터 데몬과 다른 JaNeC 데몬들이 우선 연결되어야 한다. 호스트 연결 관리자는 사용자가 병렬프로그램이 수행하는 동안에 각 호스트에서 수행중인 데몬들 사이의 연결을 관리할 수 있게 한다. 또한 사용자는 병렬프로그램이 종료한 뒤에 새로운 호스트를 추가하여 연결하거나, 기존에 연결된 호스트의 연결을 끊은 뒤에 동일한 병렬프로그램을 다시 실행하여 호스트별로 병렬프로그램의 수행과정을 비교 분석할 수가 있다. 호스트 연결 관리자는 GUI를 기반으로 구현되어 있어서 사용자는 마우스 클릭만으로도 간단하게 호스트를 연결하거나 끊을 수 있다. (그림 5)는 호스트 연결 관리자의 모습이다.



(그림 5) 호스트 연결 관리자

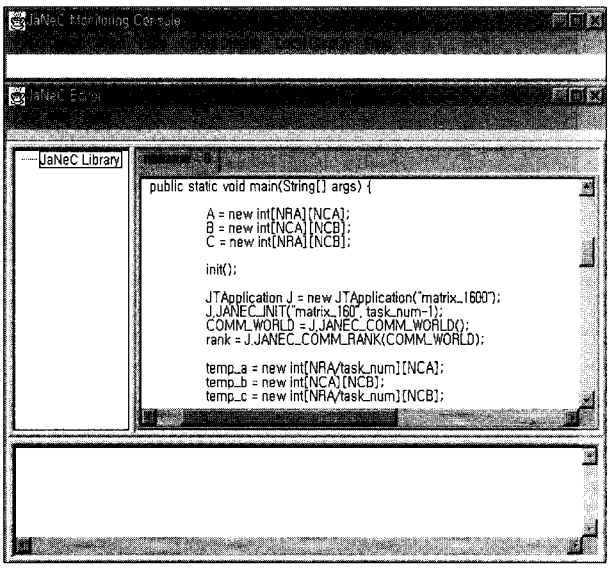
온라인 성능감시기 프로그램의 다른 기능은 병렬프로그램의 동작 형태를 그래프로 보여주는 것이다. 이 그래프는 오프라인 성능감시기와는 달리 동적으로 그 내용이 변해야 하며 자유로운 스케일링 기능이 있어야 한다. 여기서는 각 태스크가 각 시간에 어떤 일을 하고 있는지를 나타내는 Time LineView, 태스크간에 어떻게 메시지를 주고받는지를 나타내는 Message CommunicationView, 사건 종류별 백분율을 보여주는 ChartView, 사건을 텍스트 형태로 보여주는 Task HistoryView 등을 구현하였고, TimeLineView와 Message CommunicationView는 사건이 일어남에 따라 동적으로 그

내용이 변하고, 자유롭게 스케일링되도록 구현하였다.

또한, 프로그램이 종료된 이후에 다시 분석할 수 있도록 하기 위해 온라인 성능감시기 데몬으로부터 전달받은 사건을 그래프로 보여주는 동시에 이를 다시 저장한다. 저장된 데이터는 마치 오프라인 성능감시기를 수행하는 것과 같이 병렬프로그램을 재분석할 수 있게 한다.

6. 실 험

온라인 성능감시기의 테스트를 위해 JaNeC 환경에서 행렬곱을 계산하는 병렬프로그램을 수행하였다. 행렬은 1600×1600이고, 사용한 호스트의 개수는 4개이며, 각 호스트마다 1개씩의 태스크가 동작한다. 마스터 호스트에서 동작하는 마스터 태스크는 다른 태스크에게 작업을 할당하고, 각각의 태스크는 할당받은 작업을 수행한 후, 다시 마스터 태스크에게 그 결과를 전달한다. 프로그램 수행을 위해 우선 호스트 연결 관리자를 통해서 프로그램 수행에 사용될 4개의 호스트를 선택해서 연결한다. 그리고 온라인 데몬을 동작시킨 후, 프로그램을 실행시킨다. 사용자는 프로그램을 성능감시기 프로그램의 JaNeC Editor에서 작성하고 컴파일한 후 실행시킨다. (그림 6)은 성능감시기 프로그램의 JaNeC Monitoring Console과 JaNeC Editor를 나타낸 그림이다.



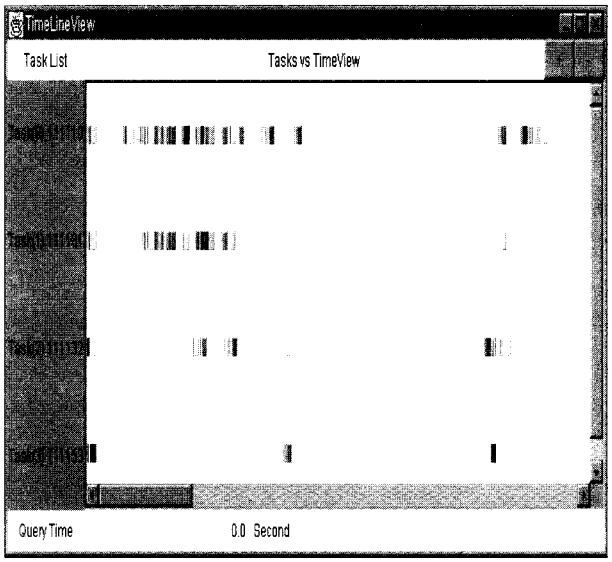
(그림 6) JaNeC Monitoring Console과 Editor

출력장치에서 출력되는 기능에는 TimeLineView, Message CommunicationView, ChartView, Task HistoryView 등이 있다. 이 중에서 온라인 분석이 가능한 것은 TimeLine View와 Message CommunicationView, 그리고 Task History View이다. ChartView는 전체 사건 중에서 특정 사건이 어느 정도의 비율을 차지하는가를 나타내기 때문에 오프라인 방법으로만 사용된다.

사용자는 병렬프로그램의 온라인 분석을 위해서 JaNeC Monitoring Console에서 TimeLineView를 이용할 것인지, Message CommunicationView를 이용할 것인지, 아니면 Task HistoryView를 이용할 것인지를 선택할 수 있다. 출력될 그래프 종류가 선택되면, 사용자는 프로그램이 수행하면서 사건을 발생할 때마다 발생된 사건들의 정보를 시각적으로 볼 수 있다. 출력되는 각각의 그래픽은 다음과 같다.

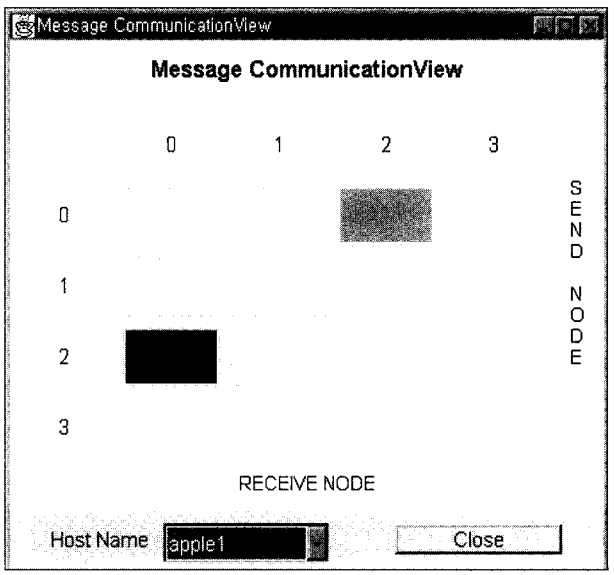
6.1 TimeLineView

(그림 7)은 예제 프로그램에 대한 TimeLineView이다.



(그림 7) TimeLineView

TimeLineView는 각 호스트별로 수행중인 태스크의 ID와 함께 그 태스크에서 발생하는 사건들을 시간별로 화면에 출



(그림 8) Message CommunicationView

력한다. 따라서 사용자는 프로그램이 동작하는 중에 Time LineView를 보면서 어떤 태스크에서 어떠한 사건이 언제 발생하는 지를 분석할 수 있다. 또한 사건들의 세밀한 분석을 위해서 그림 오른쪽 상단의 +, - 버튼을 이용하면 전체 이벤트들을 10%씩 확대하거나 축소시켜서 볼 수 있다.

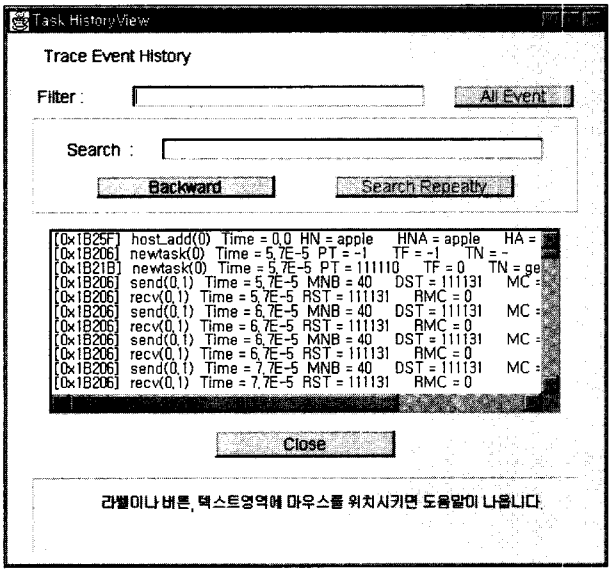
6.2 Message CommunicationView

(그림 8)은 예제 프로그램에 대한 Message CommunicationView이다.

Message CommunicationView는 태스크와 태스크 사이에 메시지를 주고받는 사건이 발생할 경우에 그 발생 정보를 시간에 따라서 색으로 출력한다. 각 태스크 별로 메시지가 이동한 경우 색상의 변화로 이들을 분석할 수 있으며, 보내기 사건이 발생하면 붉은 색의 그림이 그려지고, 받기 사건이 발생하면 연두색의 그림이 그려진다.

6.3 Task HistoryView

(그림 9)는 예제 프로그램에 대한 Task HistoryView이다.



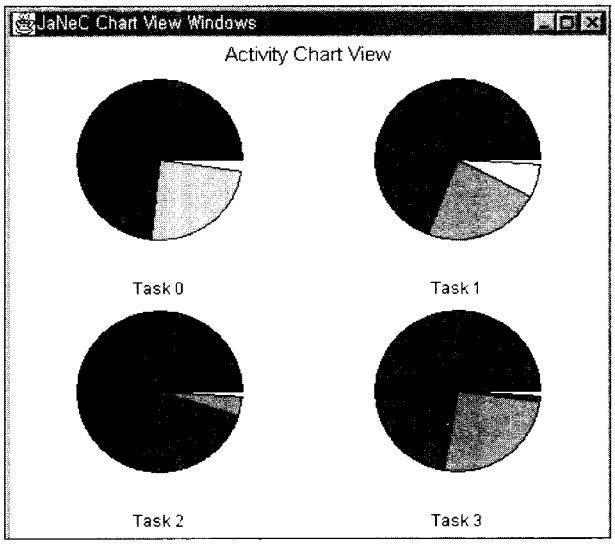
(그림 9) Task HistoryView

Task HistoryView는 프로그램이 수행되면서 발생하는 사건들을 GUI가 아닌 텍스트 형태로 출력한다. 일반적인 GUI 출력기능으로는 분석이 어려운 상세한 시간과 이에 따르는 버퍼의 사용 등이 텍스트 형태로 출력되므로 사용자는 어느 시간에 어떠한 사건이 발생했는지, 그리고 이들이 사용한 버퍼의 크기, 메시지의 형태 등을 보다 자세하게 분석할 수가 있다.

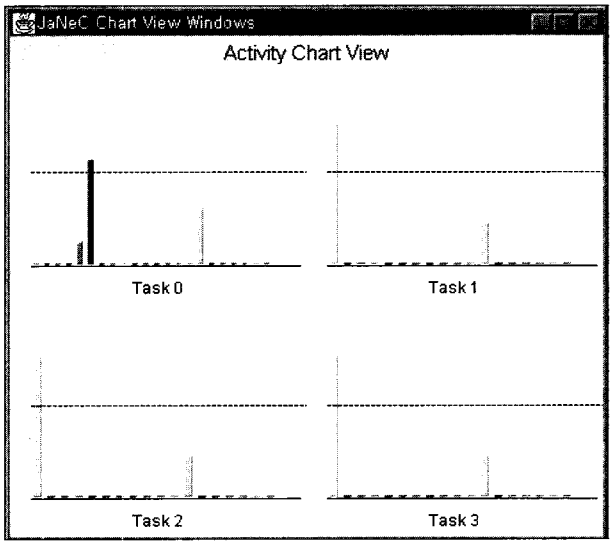
6.4 ChartView

ChartView에는 Pie, Histogram, Table, Rotation Histogram 등이 있다. 이들은 모두 각각의 태스크에서 특정 사건이 전체 사건에서 차지하는 비율이 어느 정도인지를 나타내기 때문에 온라인 분석에는 사용되지 않고, 프로그램이 종료된 후, 사용자가 재분석할 경우에 볼 수 있는 그래픽이다.

(그림 10)과 (그림 11)은 예제 프로그램에 대한 ChartView 중에서 Pie ChartView와 Histogram ChartView를 나타낸 그림이다. 사용자는 ChartView를 통해서 프로그램 수행에 있어서 가장 오랜 시간을 가진 사건을 찾아 낼 수가 있다. 따라서 전체 프로그램에서 오버헤드와 휴지 시간에 대한 비율을 한 눈에 확인할 수 있다.



(그림 10) Pie ChartView



(그림 11) Histogram ChartView

7. 결론 및 향후 연구

네트워크의 발달로 분산처리 환경에서 병렬프로그램을 수

행하는 경우가 많아졌다. 병렬프로그램은 그 진행과정이 복잡하기 때문에 이를 분석하기 위해서 성능감시기에 대한 연구가 진행되고 있다. 여러 시스템을 네트워크로 연결하는 분산처리 환경으로 JaNeC이 있는데, 이 환경에서 동작하는 성능감시기는 프로그램이 종료된 후 이벤트 상황을 분석하는 오프라인 성능감시기이기 때문에 사용자에게 빠른 분석을 제공하지 못했다. 따라서 사용자에게 빠른 분석과 디버깅을 제공하기 위해서 온라인 성능감시기가 필요하다. 본 논문에서는 JaNeC 환경에서 수행되는 병렬프로그램의 진행과정과 특성 등을 빠르게 분석하기 위한 온라인 성능감시기를 설계하고 구현한 것에 대해 설명하였다.

본 논문에서 구현한 온라인 성능감시기는 병렬 태스크에 영향을 최소화하도록 구현하였고, 동적인 그래프로 프로그램을 분석할 수 있게 하였다. 온라인 성능감시기의 구성 요소는 크게 온라인 데몬, 온라인 성능감시기 데몬, 온라인 성능감시기 프로그램 등 세 가지로 나눌 수 있다. 온라인 데몬은 태스크가 수행 중에 생성하는 사건을 실시간으로 온라인 성능감시기 데몬에게 전송해 주는 역할을 수행한다. 온라인 데몬은 태스크에 최소한의 영향을 주기 위해 JaNeC 데몬이 제공하는 추적파일을 이용해서 사건을 얻도록 구현되었다. 온라인 성능감시기 데몬은 온라인 데몬으로부터 받은 사건을 정렬하여 온라인 성능감시기 프로그램에게 전달하는 기능을 수행한다. 온라인 성능감시기 데몬은 사건의 효율적인 정렬을 위해 큐를 사용하여 사건을 저장하고 정렬한다. 온라인 성능감시기 프로그램은 사건들을 받아 다양한 그래프로 보여주고, 온라인 성능감시기 전체를 관리할 수 있게 한다. 온라인 성능감시기 프로그램에는 호스트 연결 관리자가 포함되어 있어 전체 온라인 성능감시기를 연결하고 관리할 수 있게 했으며, 다양한 동적 그래프를 구현하여 프로그램을 분석할 수 있게 하였다. 또한 온라인 상으로 분석한 내용을 저장하여 프로그램이 종료한 후에도 다시 분석할 수 있게 하였다.

향후 과제로는 온라인 데몬이 JaNeC 데몬이 작성하는 로그파일을 이용하므로 JaNeC 데몬이 로그 파일에 제공해주지 않는 그룹 통신에 관련된 사건이나, 논블록킹 라이브러리를 이용한 사건에 관해서도 처리할 수 있게 해주는 것이다.

참 고 문 헌

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

[2] Barton P. Miller, Mark D. Callaghan, Jonathan M. Cargille, Jeffrey K. Hollingsworth R. Bruce Irvin, Karen L. Karavanic, Krishna Kunchithapadam, and Tia Newhall, "The Paradyn Parallel Performance Measurement Tools," IEEE Computer Vol.28, No.11, pp.37-46, 1995.

[3] Brad Topol, John Stasko and Vaidy Sunderam, "PVaniM 2.0-Online and Postmortom Visualization Support," GVU : The Graphics, Visualization & Usability Center, 1995.

[4] Jerry Yan, S. Sarukhai and P. Mehra, "Performance measurement, visualization and modeling of parallel and distributed programs using the AIMS toolkit," Software - Practice and Experience Vol.25, No.4, pp.429-461, 1995.

[5] James Arther Kohl, G.A.Geist, "XPVM 1.0 User's Guide," Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831.

[6] Palls Gmbh, "Vampir User's Manual Release 1.1 for Vampir Version 1.0," Hermulheimer StraBe 10, D-50321 Bruhl, Germany.

[7] Pam Sogard, "MPI and PVM User's Guide," Silicon Graphics, Inc, 1997.

[8] 김봉준, 김동호, 황석찬, 김명호, 최재영, "네트워크 컴퓨터를 위한 자바 기반의 성능감시기", 정보과학회논문지(A), 제27권 제2호, pp.160-168, 2000.

[9] Ruth A. Aydt, "The Pablo Self-Defining Data Format," Department of Computer Science University of Illinois 61801, 1992.

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

[11] L. Beguelin, J. J. Dongarra, A. Geist and R. J. M. V. S. Sunderan, "Heterogeneous Network computing," In Sixth SIAM Conference on Parallel Processing, SIAM, 1993.

[12] Douglas Kramer, "The Java Platform," Sun Microsystem, 1996.

김 명 호

e-mail : kmh@comp.ssu.ac.kr

1989년 숭실대학교 전자계산학과(학사)

1991년 포항공과대학교 전자계산학과 (공학석사)

1995년 포항공과대학교 전자계산학과 (공학박사)

1995년~1995년 한국전자통신연구소 선임연구원

1998년~1999년 University of Tennessee 전자계산학과 교환교수

1995년~현재 숭실대학교 컴퓨터학부 부교수

관심분야 : GRID, 병렬/분산처리, 컴퓨터 보안, 클러스터링, 리눅스

김 남 훈

e-mail : kal1203@chollian.net

1999년 인천대학교 전자계산학과(학사)

2001년 숭실대학교 컴퓨터학과(공학석사)

2001년~현재 온세통신 사원

관심분야 : 병렬/분산처리, 클러스터링, 리눅스/유닉스, E-Mail 시스템, 모바일, MMS

최 재 영

e-mail : choi@comp.ssu.ac.kr

1984년 서울대학교 제어계측공학과(학사)

1986년 미국 남가주대학교 전기공학과 (공학석사)

1991년 미국 코넬대학교 전기공학과(공학박사)

1992년~1994년 미국 국립 오크리지연구소 연구원

1994년~1995년 미국 테네시 주립대학교 연구교수

2001년~2002년 미국 국립 수퍼컴퓨팅 응용센터 (NCSA) 초빙연구원

1995년~현재 숭실대학교 정보과학대학 컴퓨터학부 조교수/부교수

관심분야 : 병렬/분산처리, 초고속컴퓨팅, 시스템 소프트웨어