제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

실행시간 프로세스 모니터를 위한 구조 설계

정윤석, 김태완, 장천현
건국대학교 컴퓨터 정보통신 공학과
e-mail : ysjeongkku@cse.konkuk.ac.kr

Design of an Architecture for Run-time Process Monitor

Yoon Seok Jeong, Tae Wan Kim, Chun Hyon Chang
Dept. of Computer Science and Engineering , Konkuk University

요 약

실시간 시스템은 사용자의 요구에 대해 적시성을 보장하는 서비스를 제공해야 하며 이를 관리하기 위해서 모니터링 기능이 요구된다. 그러나 모니터링은 실시간 서비스에 영향을 주는 문제를 발생시킨다. 이러한 문제를 해결하기 위해 본 논문은 실시간 시스템 상의 실시간 프로세스들의 동작을 감시하는 실행시간 프로세스 모니터를 위한 구조를 설계하였다. 또한 실행시간 프로세스 모니터를 위한 구조와 연동하는 데이터 저장소를 설계하였다. 데이터 저장소를 이용하여 실행시간 모니터가 실시간 프로그램과 독립적으로 수행될 수 있도록 하였으며 이를 통해 실시간 프로그램에 미치는 영향을 최소화하도록 하였다. 본 논문에서 설계한 구조는 실시간 모니터링을 필요로 하는 분야에서 이용될 수 있다.

1. 서론

최근 실시간성을 보장하는 실시간 정보처리 및 정보감시의 요구가 높아지면서 실시간성을 지원하는 시스템이 다양한 분야에서 도입되고 있다. 이들 실시간 시스템과 관련된 핵심적인 사항 중 하나는 실시간 시스템이 정상적인 동작을 수행하고 있는지 여부를 파악할 수 있는 실시간 모니터링을 지원하는 것이다.

기본적으로 실시간 모니터링을 통해 실시간 시스템의 운영사항을 파악하고 성능 측정 및 시스템 튜닝을 위한 기초 자료를 취득하게 된다. 그러나 모니터링으로 인해 발생하는 부하들이 실시간 시스템에 영향을 미칠 수 있음을 고려해야 한다.

이 문제를 해결하기 본 논문에서는 실시간성을 지원하는 실시간 운영체제 기반으로 실시간 프로세스의 수행을 실행시간 동안 감시하는 실행시간 프로세스 모니터를 위한 구조를 설계하였다. 또한 구조와 연동하여 실시간 프로세스의 데이터를 관리하기 위한 데이터 저장소를 설계하였다.

본 논문은 2 장에서 실시간의 개념과 모니터링에 대한 관련 연구를 다루며 3 장에서는 실시간 시스템 상의 실행시간 프로세스 모니터를 위한 구조의 설계 내용을 기술한다. 마지막 4 장 결론에서는 본 논문의 성과 및 향후 연구 방향에 대해서 기술한다.

2. 관련 연구

2.1 실시간의 개념

일반적으로 실시간 개념은 '특정 이벤트에 대해 즉시 응답하는 것'을 의미하며 여기에 시간 제약이라는 개념을 추가하면 '특정 이벤트를 제한 시간 내에 처리하는 것'을 의미한다[1].

실시간은 시간 제약의 중요성에 따라 크게 경성 실시간과 연성 실시간으로 분류한다. 경성 실시간은 태스크가 정확하게 실행될 뿐 아니라, 정확한 시간에 실행되어야 하며 시간 제약 위반 시 치명적 결과를 야기하는 경우를 말한다. 반면 연성 실시간은 좀 더 완화된 경우로서, 가능한 빨리 실행되지만 어떤 정해진 시간 내에 종료할 필요는 없는 경우이다.

2.2 실시간 시스템

실시간 시스템은 '시스템의 수행 결과가 기능적으로

본 연구는 대학 IT 연구센터 육성· 지원사업의 연구결과로 수행되었음

정확하고, 결과 도출 시간이 주어진 제약 조건을 만족시키는 시스템'이다. 실시간성을 만족하는데 있어 중요한 요소는 인터럽트가 발생했을 때 짧은 시간 안에 인터럽트 처리기를 호출하는 것과 우선순위가 높은 이벤트를 먼저 처리하도록 하는 것이다[1][2]. 이를 위해 우선순위 기반 선점형 스케줄링이라는 스케줄링 방식을 이용한다. 실시간 시스템의 핵심을 이루는 실시간 운영체제가 갖추어야 할 특징은 다음과 같다.

- 다중쓰레드 지원 및 선점가능
- 쓰레드 간의 우선 순위 보장
- 쓰레드 간의 동기화 지원
- 운영체제 행동의 명확성

2.3 모니터링

모니터링이란 '데이터를 수집하고, 데이터를 통하여 각각의 시간 별로 어플리케이션 및 운영체제가 어떤 작업을 수행했는지를 파악하는 것'을 의미한다[3][4][5]. 모니터링은 사용 목적 별로 볼 때 다음의 7 가지로 나누어질 수 있다. 본 논문에서는 7 가지 기능 중에서 정확성 검사, 성능평가에 초점을 맞추었다.

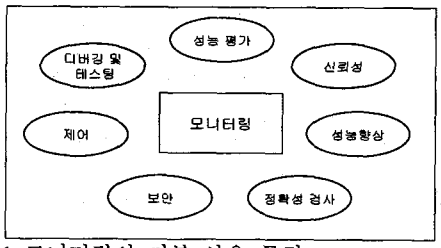

그림 1 모니터링의 기본 사용 목적

- 신뢰성 : 결함 허용 및 안정성을 제공
- 성능향상 : 동적 시스템 설정 및 프로그램 조정 기능
- 정확성 검사 : 어플리케이션이 일관성 있게 규약을 준수하고 있는 여부를 확인하는 기능
- 보안 : 비권한 사용자가 로그인이나 데이터 접근 권한을 범하는 경우를 찾아내는 기능
- 제어 : 대상 시스템의 일부로 구성되어 동작하는 제어 관리 기능
- 디버깅 및 테스팅 : 테스트한 어플리케이션으로부터 의미 있는 데이터를 추출하는 기능
- 성능평가 : 시스템 성능을 분석한 후 분석 데이터를 추출하는 기능

2.4 모니터링 시스템의 활동

모니터링 시스템을 활동 관점에서 볼 때 많은 활동을 고려할 수 있지만 크게 두 가지 기준에서 대별할 수 있다[5][6].

- 모니터링은 사람에 의해서 수행되는가? 모니터링 시스템에 의해 수행되는가?
- 대상 프로그램이 동작되기 전에 수행는가? 동작 중에 수행되는가? 혹은 둘 다인가?

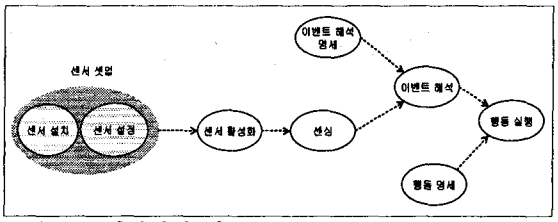

그림 2 모니터링의 활동

그림 2 의 경우 음영 부분은 모니터링 프로그램을 수행하기 전, 즉 선행수행 활동을 의미하며 나머지 비음영 부분은 모니터링 프로그램이 수행되는 동안 이루어지는 활동이다. 센서는 모니터링을 위한 코드를 의미한다.

- 센서 셋업 : 수행 전에 이루어지는 활동이며 '어떤 데이터를 획득할 것인가? 센서가 어느 부분에 위치할 것인가?'를 결정한다.
- 활성화된 센서 : 데이터를 수집할 상태로 활성화된 상태이다. 센싱은 대상 어플리케이션에 대한 데이터를 수집하는 실행시간 시의 활동이다.
- 이벤트 해석 : 모니터링 시스템의 주요부분으로서 수집된 정보를 분석하는 것을 말한다. 또한 에러 조건 등을 명시함으로써 특정 이벤트가 발생했을 때 이를 비교하여 에러 발생 여부를 검사하는 활동을 수행한다.
- 액션 명세 및 실행 : 심각한 이벤트가 발생했을 때 어떤 행동을 취할 것인지를 정의하고 실행

3. 설계

3.1 전체 구조

그림 3 은 실행시간 프로세스 모니터를 위한 전체 시스템 구조를 보여준다. 구조는 크게 4 개의 계층으로 구성된다. 특히 음영 부분은 실행시간 프로세스 모니터를 위해 본 논문에서 추가 설계한 부분이다.

- 어플리케이션 계층 : 운영체제 상에서 동작하는 일반 어플리케이션과 실시간 프로그래밍을 지원하는 실시간 어플리케이션을 지원
- 미들웨어 계층 : 실시간 프로세스가 동작하는 필요한 기본 기능을 제공
- 운영체제 계층 : 운영체제의 기본 기능을 제공
- 하드웨어 계층 : 메모리를 통해 실시간 프로세스 모니터나 실시간 운영체제가 데이터를 공유

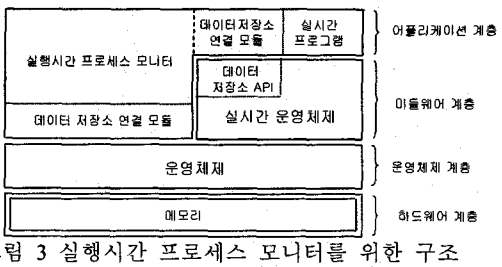

그림 3 실행시간 프로세스 모니터를 위한 구조

3.2 실행시간 프로세스 모니터

모니터의 기본 기능은 주기적으로 특정 프로그램의 동작 상태를 감시하는 것이다. 따라서 실행시간 프로세스 모니터가 모니터의 기본 기능을 지원할 수 있도록 그림 4 과 같이 4 개의 기능을 갖도록 설계하였다. 그림 4 의 음영처리 된 부분은 실행시간 프로세스 모니터가 제공하는 기능을 의미하며 실선 화살표는 실시간 프로세스의 실행 시간 동안 발생하는 정보의 흐름을 나타낸다.

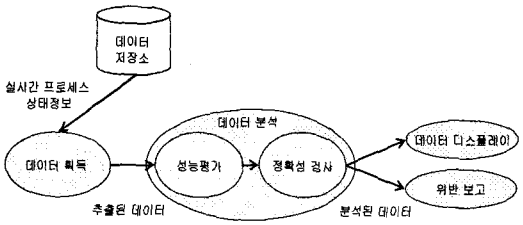

그림 4 실행시간 프로세스 모니터의 기능

3.2.1 데이터 획득

이 기능은 데이터 저장소에서 실시간 프로세스의 상태 정보를 모으는 기능이다. 그림 5 는 데이터 획득의 프로세스를 나타낸다. 데이터 저장소는 실행시간 프로세스 모니터가 시작될 때 생성되며 종료 시 삭제된다. 점선상자 내의 데이터 읽기/쓰기 프로세스는 반복적으로 수행되는 부분이다.

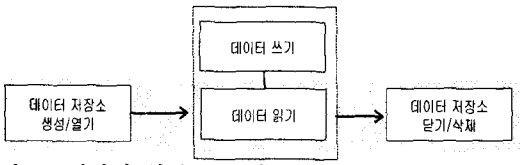

그림 5 데이터 획득 프로세스

실행시간 프로세스 모니터가 데이터 저장소에 접근하여 데이터를 얻기 위해서 두 가지 모듈이 필요하다. 그림 6 의 음영으로 나타난 데이터 저장소 연결 모듈과 데이터 읽기 모듈이 그것이다.

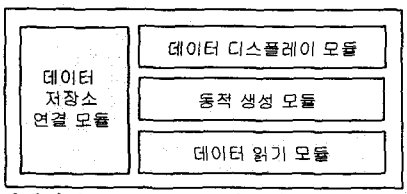

그림 6 실행시간 프로세스 모니터의 구조

데이터 저장소 연결 모듈은 데이터 저장소를 생성하고 삭제하는 기능을 제공한다. 데이터 저장소는 프로세스 ID, 실행시간, 잔여 실행시간, 실행주기, 데드라인 등의 프로세스 정보를 저장한다. 이들 정보는 데이터 저장소에 저장되어 필요 시 데이터를 읽고 쓰게 된다. 그림 7 은 본 논문에서 설계한 데이터 저장소를 나타내며 참조 레코드, 쓰기 레코드, 메타데이터 레코드로 구성된다. 참조 레코드와 쓰기 레코드가 중복된 레코드인 것은 데이터 무결성을 유지하기 위해서이다.

예를 들어 한 어플리케이션이 쓰기 레코드를 읽고 있는 동안 실시간 프로세스가 쓰기 레코드에 프로세스 정보를 쓰게 된다면 데이터 무결성이 깨지게 된다. 이를 해결하기 위해 실시간 프로세스는 참조 레코드에 데이터를 기록한 후 일괄적으로 데이터를 쓰기 레코드에 복사함으로 데이터 무결성 문제를 해결한다.

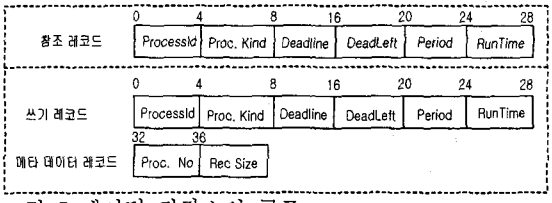

그림 7 데이터 저장소의 구조

3.2.2 데이터 분석

이 기능은 데이터 저장소의 데이터를 분석하는 것이다. 크게 성능평가와 정확성 검사로 구성된다. 성능평가는 프로세스가 얼마나 빨리 동작했는가를 측정하는 분석 기능이다. 그림 8 은 실시간 프로세스의 시간 측정 메커니즘을 나타낸다.

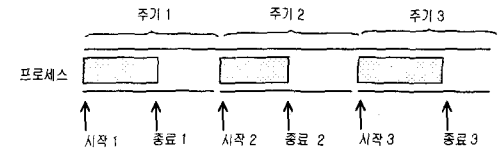

그림 8 성능측정을 위한 실행시간 측정 메커니즘

반면 정확성 검사는 실시간 프로세스가 데드라인을 준수했는지 여부에 초점을 둔다. 실시간 시스템에서 실행의 적시성이 가장 중요하기 때문에 실행시간 프로세스 모니터 또한 데드라인에 대한 정확성 검사를 제공할 수 있도록 설계하였다. 그림 9 정확성 검사를 위한 데드라인 대비 실행시간 측정 메커니즘을 보여준다.

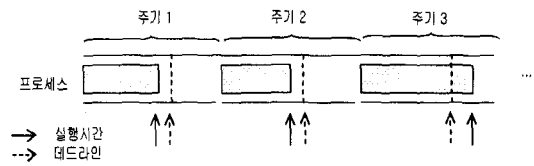

그림 9 정확성 검사를 위한 데드라인 대비 실행시간 측정 메커니즘

3.2.3 데이터 디스플레이

데이터 디스플레이는 기본적으로 프로세스의 데이터를 보여주는 기능을 제공한다. 본 논문에서는 데이터를 발생시키는 부분과 디스플레이를 하는 부분을 분리하여 설계했다. 만일 두 부분이 통합되어 있다면 디스플레이를 위한 오버로드가 데이터 발생에 영향을 미치기 때문이다. 그림 10 은 이렇게 디스플레이 부분

과 데이터 발생 부분을 분리한 분리 구조를 제시한다.

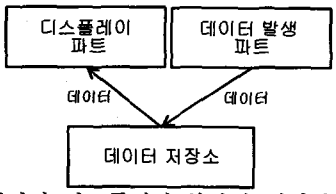

그림 10 데이터 디스플레이 부분과 데이터 발생 부분의 분리 구조

그림 11 은 실행시간 프로세스 모니터의 디스플레이 부분의 구조를 제시한다. 음영 부분인 데이터 디스플레이 모듈, 동적 생성 모듈이 디스플레이 기능을 제공하는 부분이다.

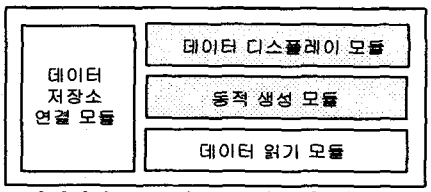

그림 11 실행시간 프로세스 모니터의 디스플레이 기능 구조

데이터 디스플레이 모듈은 실시간 프로세스의 데이터 및 분석 데이터를 보여주며 동적 생성모듈은 각 프로세스 별 디스플레이 요소를 생성하는 부분이다. 동적으로 생성하는 이유는 실행 시까지 실시간 프로세스의 수를 알 수 없기 때문이다.

3.2.4 위반 보고

이 기능은 특정 이벤트가 발생했을 때 이를 사용자에게 알리도록 하는 기능이다. 그림 12 는 위반 보고 프로세스를 보여주고 있다. 이 기능의 설계는 이벤트 해석 명세 및 행동 명세의 정의를 전제로 한다. 이벤트 해석 명세는 시스템 상의 프로세스 상태가 정의된 규칙과 일치한다면 행동 명세에 정의된 규칙을 실행하는 것이다. 행동 명세는 이벤트 해석 명세를 위반시 수행하는 행동 규칙이다. 본 논문에서는 다음과 같이 정의하였다.

- 위반 이벤트 : 프로세스가 데드라인을 위반했을 경우
- 위반시 행동 : 실행시간 프로세스 모니터는 사용자에게 경고를 보냄

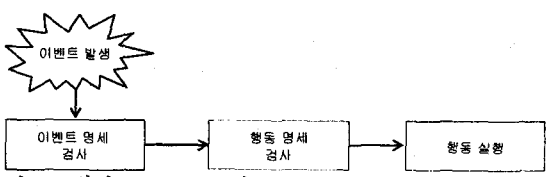

그림 12 위반 보고 프로세스

3.3 데이터 저장소 API 를 추가한 실시간 운영체제 구조

그림 3 에서 제시되었던 구조를 기반으로 실행시간 프로세스 모니터의 구현을 고려할 때 몇몇의 API 의 추가가 요구된다. 기본적으로 이들 API 는 데이터 저장소를 이용하는데 있어 요구되는 기능을 포함해야 한다. 데이터 저장소 API 가 삽입된 구조는 그림 13 과 같다.

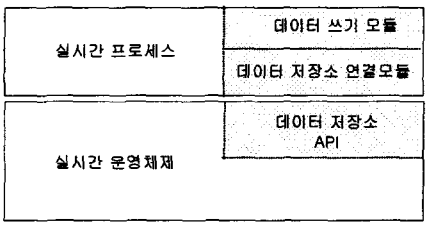

그림 13 데이터 저장소 API 를 추가한 실시간 운영체제 구조

4. 결론 및 향후 방향

본 논문에서 실행시간 프로세스 모니터를 위한 구조를 설계하였다. 이 구조는 데이터 저장소 개념을 추가하여 모니터링에 대한 유연성을 높일 수 있도록 설계되었다. 이는 데이터 저장소가 실시간 프로세스와 실행시간 프로세스 모니터의 연결을 담당하는 본 논문에서의 설계 예 외에도 실시간 프로세스 정보에 접근하고자 하는 여타 어플리케이션과도 연동할 수 있음을 의미한다. 또한 이 구조는 실행시간 프로세스 모니터와 실행시간 프로세스 모니터의 대상 프로세스와의 분리를 통해 효율성을 제고하도록 설계하여 디스플레이로 야기되는 영향력을 감소시키는 역할을 하게 된다.

향후 작업으로는 먼저 특정 실시간 프로그래밍 모델(TMO 모델 등)을 이용해서 본 논문에서 제시한 구조를 기반으로 하는 구현을 계획하고 있다. 둘째로는 실행시간 프로세스 모니터에 시스템 모니터링 기능을 연동하는 것이다. 실시간 프로그램은 시스템 자원의 상태에 따라 영향을 받기 때문에 시스템 모니터링 기능을 부가함으로써 보다 정확한 모니터링 결과를 획득하도록 해야 한다. 이를 위해 시스템 모니터링과 프로세스 모니터링 기법을 연동할 계획이다.

참고문헌

[1] Nimal Nissanke, "Realtime Systems", Prentice Hall, 1997
[2] Sang H. Son, "Advanced In Real-Time Systems, Prentice Hall, 1995
[3] "How To Enable Process Accounting on Linux", http://kldp.org
[4] Mike Loukides, "System Performance Tuning", O'Reilly, 1990
[5] B.A. Schroeder, "On-line Monitoring: A Tutorial", IEEE Computer, 1995
[6] B.J. Min, et al., "Implementation of a Run-time Monitor for TMO Programs on Windows NT", IEEE Computer, 2000