개선된 모니터링 센서를 이용한 임베디드 모니터링 시스템의 설계 및 구현

신원° 김태완 장천현
건국대학교 컴퓨터 정보통신공학과
{wonjjang°, twkim, chchang}@konkuk.ac.kr

Design and Implementation of Embedded Monitoring System using Improvement of Monitoring Sensor

Won Shin°, Tae Wan Kim. Chun Hyon Chang
Dept of Computer Science and Engineering, Konkuk University

요 약

최근 가정, 자동차, 사무실등 많은 장소에서 임베디드 장치를 쉽게 찾아볼 수가 있다. 이렇듯 많은 분야에서 임베디드 장치가 사용됨에 따라 임베디드 시스템 제조업자들은 보다 빠른 시간에 많은 분야에 적용 가능한 임베디드 소프트웨어를 개발해야 하는 어려움이 생겼다. 이러한 어려움은 소프트웨어 개발시간을 줄일 수 있도록 도와주는 개발환경의 필요성을 증가시켰다. 개발도구 중 디버깅도구는 개발시간의 대부분을 차지하는 디버깅 과정을 도움으로써 개발시간 단축에 큰 역할을 한다. 기존 디버깅도구는 모든 변수에 대한 데이터 추출을 하기 위하여 자원사용량과 처리량을 증가시킴으로써 많은 부하를 발생시킨다. 이에 모니터링에 의한 부하를 최소화하기 위하여 변수 값을 추출하기 위한 모니터링 센서 기법과 실행시간 중 모니터링 대상을 변경하기 위한 디버깅레벨기법을 사용하여 소프트웨어의 내부 변수가 동작하면서 발생되는 오류를 검출할 수 있는 임베디드 모니터링 시스템을 설계 및 개발하였다. 하지만 실행시간 모니터링 중에 센서의 동작과정에 따르는 부하로 인하여 정확하지 않은 데이터를 추출할 수 있는 문제점이 발생되었다. 이러한 문제를 해결하기 위해 본 논문에서는 센서의 수행과정을 최소화하도록 센서 처리 구조를 변경하고 최적화된 센서 구조의 실행시간을 줄이기 위해 비트마스킹 기법을 사용한다.

1. 서 론

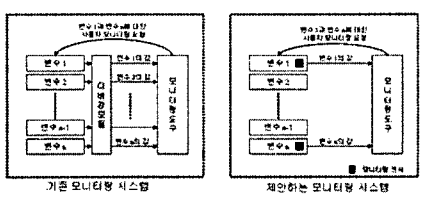
임베디드 장치는 가정, 자동차, 사무실에서부터 원격 어셈블리 설비 및 해상 드릴링 플랫폼에 이르는 거의 모든 분야에 이용되고 있다. 이렇듯 많은 분야에서의 임베디드 장치가 사용됨에 따라 임베디드 시스템 제조업자들은 빠른 시간 내에 각각의 분야에 맞는 시스템을 개발해야 하는 어려움이 따르게 되었다. 임베디드 장치는 크게 임베디드 하드웨어와 소프트웨어로 나뉘게 되는데 하드웨어의 경우 이러한 빠른 변화에 적용가능 하도록 많은 기술들이 존재하여 개발가능하다. 하지만 소프트웨어의 경우 아직까지도 개발환경이 미흡하여 소프트웨어를 빠르게 개발하고 쉽게 유지보수 할 수 있는 개발도구들의 중요성이 대두되고 있다. 여러 종류의 임베디드 소프트웨어 개발도구 중에서도 소프트웨어의 개발 시간 중에 많은 시간을 소비하는 디버깅 과정을 도울 수 있는 디버깅도구가 개발 기간 단축에 큰 역할을 한다. 디버깅도구란 소프트웨어의 동작 과정에서의 오류를 찾아내는 도구로서, 일반적으로 소프트웨어의 오류를 찾을 수 있는 가장 큰 기준이 되는 내부 변수를 감시하는 과정을 수행한다. 기존 디버깅도구는 모든 변수에 대한 데이터 추출을 하기 위하여 시스템의 자원사용량과 처리량을 증가시킴으로써 많은 부하를 발생시킨다. 이에 모니터링 과정의 자원사용량과 처리량을 줄이기 위하여 모니터링 센서를 소스레벨에서 삽입하여 응용소프트웨어를 감시하는 임베디드 모니터링 도구를 개발하게 되었다. 하지만 모니터링 센서기법의 동작과정에서 생기는 오버헤드로 인해 실행되는 소프트웨어의 실제 값과는 다른 모니터링 값을 얻을 수 있는 문제점이 발생된다. 이에 본 논문에서는 모니터링 센서에 의해 생기는 모니터링 데이터의 변질을 막기 위하여 최소한의 부하만이 생성될 수 있도록 모니터링 센서의 구조를 변경하고 센서의 실행시간을 줄이기 위해 비트마스킹 기법을 사용한다.

본 논문은 2장에서 기존의 모니터링 시스템의 구조와 문제점을 제시하고, 3장에서는 개선된 모니터링 시스템의 설계구조와 구현과정 대하여 기술한다. 4장에서는 모니터링 도구의 특징에 대하여 설명하고 마지막으로 5장에서 본 논문의 성과 및 향후 방향에 대하여 기술한다.

2. 기존 모니터링 시스템

기존 모니터링 시스템에서는 프로그램 상에 디버깅모듈을 포함하고 모든 변수에 대해 모니터링을 하는 방법을 사용하였다. 이러한 방식의 사용은 모든 변수에 대한 모니터링 값 처리와 프로그램 상에 디버깅모듈을 포함함으로써 처리상의 부하와 많은 자원사용량 등의 문제가 발생된다.

이러한 문제점을 해결하기 위해 모든 변수가 아닌 사용자가 원하는 변수만을 모니터링 하여 모니터링에 의한 부하를 최소화하고 모니터링 중에 모니터링 대상을 변경할 수 있는 디버깅 레벨기법을 적용하였다[1].



<그림 1> 기존 모니터링시스템과 제안하는 모니터링시스템 비교

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음

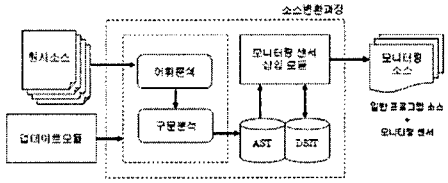
제안하는 모니터링 시스템은 모니터링 센서의 사용으로 모니터링으로 생기는 많은 부하를 줄였지만 응용프로그램이 실행되는 도중에 수행되는 센서처리 과정은 잘못된 모니터링 데이터 추출을 야기할 수 있다. 이에 모니터링 센서가 해야만 하는 최소한의 역할만을 처리할 수 있는 연산 과정만을 수행하도록 센서의 구조를 변경하여 센서의 부하를 최소화한다.

3. 임베디드 모니터링 시스템의 설계 및 구현

임베디드 모니터링 시스템은 크게 모니터링 센서 삽입과정과 실행시간 모니터링 과정으로 나뉜다. 모니터링 센서 삽입과정은 실행시간 모니터링 과정을 수행하기 위한 준비작업으로써 모니터링 센서를 응용 소프트웨어의 소스에 삽입하는 과정이다. 실행시간 모니터링 과정은 센서가 삽입된 모니터링 소스를 타겟 시스템에서 실행하고 센서를 통하여 전송되는 모니터링 데이터를 다양한 방법으로 모니터링하는 과정이다.

3.1 모니터링 센서 삽입 구조

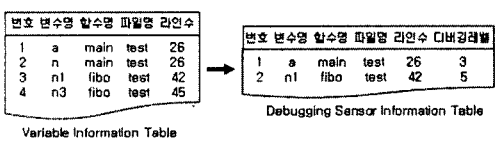
모니터링 센서 삽입과정은 원시소스를 분석하여 사용자가 원하는 위치에 센서를 삽입하고 모니터링 소스를 생성하는 과정이다. 센서를 삽입하는 과정을 보다 편리하게 하기 위하여 임베디드 모니터링 시스템은 SICT(Sensor Insert and Configuration Tool)를 제공한다. <그림 2>는 SICT의 전체구조이다.



<그림 2> SICT의 구조

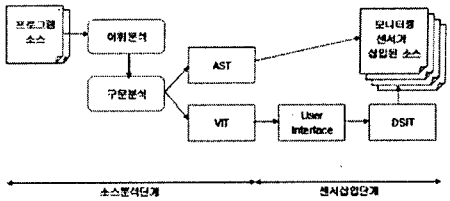
모니터링 센서 삽입과정은 원시소스를 분석하여 센서의 구성을 트리로 나타내는 AST(Abstract Syntax Tree)와 원시 소스에 있는 변수들의 정보를 포함하는 VIT(Variable Information Table)를 구성한다.

사용자는 VIT를 사용하여 원시소스의 변수 구조를 보여주는 유저 인터페이스를 통하여 모니터링 하고 싶은 변수를 선택한다. 선택된 변수의 정보는 센서의 정보를 나타내는 DSIT(Debugging Sensor Information Table)를 구성할 때 사용된다. DSIT의 디버깅레벨이란 변수의 중요도를 나타내는 기준으로 0부터 9의 값을 갖으며 높은 수일수록 프로그램 내에서 많은 비중을 갖는 변수이다.



<그림 3> VIT와 DSIT의 구조

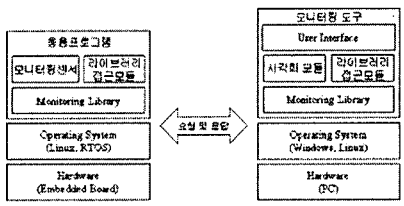
모니터링 소스는 DSIT의 정보를 참조하면서 AST를 탐색하는 과정을 통하여 생성된다.



<그림 4> 모니터링 센서 삽입 구조

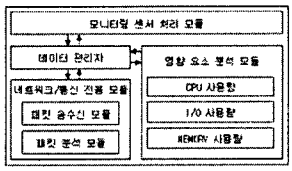
3.2 실행시간 모니터링 구조

실행시간 모니터링 과정은 모니터링 센서 삽입 과정에서 생성된 모니터링 소스를 타겟시스템에서 실행하면서 모니터링 데이터를 추출하여 모니터링 하는 과정이다.



<그림 5> 실행시간 모니터링 전체구조

타겟과 호스트시스템에는 모니터링 데이터 추출과정, 시각화 과정등 모니터링과 관련된 모든 기능을 수행하는 모니터링 라이브러리가 포함된다. 모니터링 라이브러리는 크게 센서 처리 모듈, 영향 요소 분석 모듈, 전송 모듈로 나뉜다. 영향 요소 분석 모듈은 타겟 시스템의 하드웨어 관련 정보를 추출하는 역할을 한다.



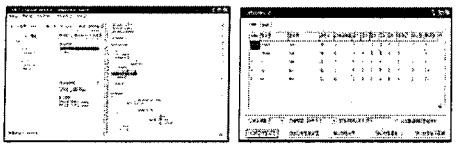
<그림 6> 모니터링 라이브러리의 구조

응용프로그램의 실행시간 중에 센서는 모니터링 데이터를 추출한다. 추출된 모니터링 데이터는 시리얼, 네트워크, 파일등 다양한 방식으로 호스트 시스템에 전송된다. 전송된 데이터는 시각화모듈을 통하여 사용자에게 보여 진다. 네트워크를 통한 전송의 경우, 타겟시스템과 호스트시스템은 정해진 프로토콜을 사용하여 모니터링 데이터 요청과 응답을 하게 된다. 이러한 프로토콜의 사용은 보다 정확하고 빠른 데이터의 전송을 위해 사용된다.

3.3 임베디드 모니터링 시스템의 구현

SICT는 AST와 VIT를 생성하기 위하여 Lex와 Yacc을 사용하였고, 분석 가능한 문법은 임베디드 시스템에서 가장 널리 쓰이는 "C"언어를 기준으로 하였다. 모니터링 센서 삽입 과정은 소스레벨에서 센서를 삽입하는 소스인라인 기법을 사용하였다. 모니터링 도구는 모니터링 데이터를 테이블 형태와 그래프 형태로 시각화하여 표현하여 현재의 소프트웨어 상태를 확인하거나 저장하여 그래프 형태로 시각화하여 데이터의 추이를 감시

할 수 있다. <그림 7>은 SICT와 모니터링 도구의 실행화면이다.

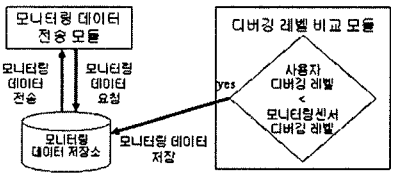


SICT 모니터링 도구

<그림 7> SICT와 모니터링 도구의 실행화면

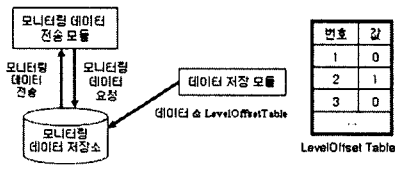
4. 임베디드 모니터링 시스템의 특징

제안하는 임베디드 모니터링 시스템은 모든 변수가 아닌 사용자가 지정하기 위한 변수만을 모니터링 하기 위한 모니터링 센서 기법을 사용한다. 모니터링 센서는 모니터링 디버깅레벨과 센서의 디버깅레벨을 비교하여 그 결과에 따라 변수의 값을 추출하거나 추출하지 않는 과정을 수행한다. 이렇듯 센서의 동작과정 중에는 디버깅레벨 값을 비교하는 비교문이 들어가게 된다.



<그림 8> 기존 모니터링 센서 동작화면

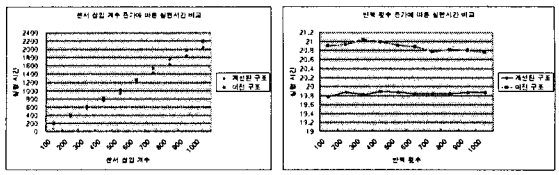
응용 프로그램의 실행 중에 센서 하나의 비교문은 임베디드 시스템처럼 짧은 동작들이 무한적으로 수행하는 시스템에서는 많은 부하를 발생시킬 수 있다. 이에 비교문을 제거하기 위하여 디버깅레벨 비교를 하는 루틴을 응용 프로그램을 수행하기 전에 먼저 수행하여 값을 미리 계산하여 LevelOffset 테이블을 작성한다. LevelOffset 테이블은 센서의 번호와 모니터링 데이터의 전송여부를 저장하는 공간이다. 전송여부란 사용자가 지정한 디버깅레벨보다 낮은 레벨을 가지는 센서를 구분함으로써 실행시간 중에 사용자가 원하는 센서 값만을 모니터링할 수 있도록 하는 기준이며 0 또는 1의 값을 갖는다.



<그림 9> 개선된 모니터링 센서 동작화면

모니터링 데이터의 값은 실제 데이터 값과 LevelOffset테이블의 값의 &연산을 통하여 저장된다. 이때 &연산의 수행은 비교결과가 1인 경우 데이터의 값을 저장하고 0인 경우 0이라는 값을 저장하는 역할을 한다. 이와 같은 과정을 통하여 센서는 비교문 없이 단순히 변수의 값을 저장하는 역할만을 수행하게 되어 실행시간 모니터링 중에 센서에 대한 부하가 감소하게 된다. 모든 모니터링 데이터처리과정에서의 반복문, 제어문처럼 명령어 개수가 많은 문장을 특정 기법을 사용하여 명령어의 개수를 줄여 최소한의 명령어와 시간만을 사용하는 모니터링 과정이 진행되도록 한다.

이전 시스템과 개선된 구조의 실행시간을 비교해 본 결과 센서당 130ns의 성능 향상을 보이며 이는 100개의 센서가 동작할 경우 0.9초의 시간차를 발생시킨다.



<그림 10> 이전 구조와 개선된 구조의 실행시간 비교

또한 타겟 시스템과 호스트 시스템간의 통신에서는 데이터 전송 속도 향상을 목적으로 규약된 프로토콜을 사용한다. 여기서 프로토콜은 통신상태, 통신 방법 등 모니터링을 하기 위한 가장 기본적인 통신인 모니터링 데이터의 전송 요청과 응답 등으로 구성하였다. 이때 모니터링 되는 데이터에는 변수 값 외에도 시스템의 상태 값을 모니터링할 수 있도록 하여 시스템에 의해 생기는 오류를 확인할 수 있게 한다. 즉, 변수 값과 시스템 상태를 동시에 모니터링 함으로써 보다 정확한 프로그램 분석이 가능해져 프로그램의 신뢰성을 높일 수 있게 한다.

5. 결론

임베디드 시스템의 활용범위가 넓어짐에 따라 개발자들에게는 보다 빠른 시간에 임베디드 소프트웨어를 개발해야 하는 어려움이 생겼다. 이에 보다 더 빠른 개발을 위한 개발환경이 요구되었다.

본 논문에서는 이러한 문제를 해결하기 위하여 개발도구 중 디버깅도구에 초점을 맞추고 기존 시스템의 문제점인 자원과 처리량의 부하를 해결하기 위하여 특정 변수만을 모니터링하기 위한 모니터링 센서기법과 실행시간 중에 모니터링 대상을 변경하기 위한 디버깅레벨기법을 제안한다. 소프트웨어의 오류가 아닌 시스템에 의해 생기는 오류검출을 위하여 모니터링 대상에 시스템 상태도 포함하여 보다 정확한 분석이 가능해진다. 또한 센서의 부하는 잘못된 모니터링을 야기할 수 있기 때문에 센서의 구조를 최적화하여 변경하고 비트 마스크 기법을 사용하여 센서에 대한 부하를 최소화한다.

이러한 임베디드 모니터링 시스템은 각종 임베디드 환경의 모든 기술에 탑재, 핵심기술로 적용될 수 있다.

향후에는 이기종의 운영체제에서도 동작할 수 있도록 모니터링 라이브러리를 재구성하고 분산 환경에도 적용 가능하도록 구조의 확장이 필요하다.

6. 참고문헌

[1]신원, 김태완, 장천현, "임베디드 환경에서의 효율적인 디버깅을 위한 모니터링 시스템 설계", 정보처리학회, 2004

[2]Roman Obermaisser, "Monitoring and Configuration in a Smart Transducer Network", IEEE Real-Time Embedded System Workshop, 2001

[3]H. Beier, "Software monitoring parallel programs", CONPAR, 1988

[4]John D. Johnson t, "Implementation Issues for a Source Level Symbolic Debugger", ACM, 1983

[5]MAX COPPERMAN, "Symbolic Debugging of Optimized Code", ACM Transactions on Programming Languages and Systems, 1993