임베디드 소트프웨어를 위한 그래픽 기반 성능 분석기의 설계 및 구현

신경호°, 조용윤, 유재우
숭실대학교 컴퓨터학과
{khshin, yycho}@ss.ssu.ac.kr, cwyoo@computing.ssu.ac.kr.

A Design and Implementation of Graphic-based Performance Analyzer for Embedded Software

Kyoung-Ho Shin, Yong-Yoon Cho, Chae-Woo Yoo
Dept. of Computing, Soongsil University

요 약

본 논문은 임베디드 소프트웨어 개발자가 개발 소프트웨어의 성능 테스팅 결과를 그래픽 형태의 인터페이스를 통해 쉽고 편리하게 분석할 수 있는 성능 분석기를 제안한다. 제안하는 성능 분석기는 임베디드 소프트웨어에 대해 생성된 텍스트 기반 저수준의 성능 평가 정보를 그래픽 형태의 결과 화면으로 재구성하기 위한 API로 변형하는 정보 변환기 모듈과 API 형태의 자료구조를 이용해 성능 평가 결과를 그래픽 형태로 출력하는 레포트 생성기로 구성된다. 제안하는 그래픽 기반의 성능 분석기는 개발자나 사용자에게 그래픽 형태의 편리한 성능 분석 레포팅을 제공한다. 따라서, 임베디드 소프트웨어 개발자는 기존의 텍스트 형태의 결과를 분석하기 위한 시간과 노력을 줄일 수 있고, 즉각적이고 직관적인 결과 분석기회를 얻을 수 있어 관련 소프트웨어 개발 효율성을 높일 수 있을 것이다.

1. 서론

임베디드 소프트웨어에 대한 성능 평가 및 결과 분석은 제한된 임베디드 시스템 자원에 효율적인 임베디드 소프트웨어 개발이 효율성 향상에 기여할 수 있다. 보통 임베디드 소프트웨어의 개발은 호스트/타겟 방식에 의한 교차 개발(cross development) 환경을 통해 이루어진다. 그림 1은 일반적인 호스트/타겟 방식에 의한 임베디드 소프트웨어의 성능 테스팅 및 결과 분석을 위한 시스템 구조도 이다.

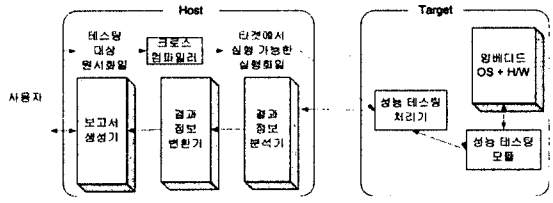

그림 1. 호스트/타겟 방식의 임베디드 소프트웨어를 위한 성능 테스팅 및 결과 분석 시스템 구성도

따라서, 성능 평가를 위한 실제 프로세스 테스팅은 타겟에서 실행되며, 그 결과는 호스트에서 적절한 분석 방법에 의해 사용자에게 레포팅 되어야 한다. 다음 표1 은 임베디드 소프트웨어에서 실시되는 가장 기본적이며, 중요한 성능 측정 항목이다.

표 1. 임베디드 소프트웨어의 성능 측정 범위

테스팅 범위	내용
성능 테스팅	- 응용 프로그램 전체 또는 원하는 일부분의 실행에 걸리는 시간을 측정 - 특정 함수별 실행 시간 측정 - CPU 할당 및 처리 시간 측정
코드 범위 테스팅	- 응용 프로그램을 실행 시 실제로 사용되는 부분과 사용되지 않는 부분, 자주 사용되는 부분, 거의 사용되지 않는 부분을 측정
메모리 테스팅	- 메모리의 할당과 해제 정보를 출력 - 전체 메모리 사용량 측정. 또한, - 할당 되었는데 해제되지 않은 메모리를 검사함으로써 메모리 누수를 측정 - 중복 해제되어 버그를 유발시킬 수 있는 코드를 측정
트레이스 테스팅	- 응용 프로그램의 실행과 함께 어떤 순서로 함수 호출이 일어나는지를 출력 - 응용 프로그램이 정상적으로 진행되고 있는지 여부 측정 - 불필요하게 함수 호출이 많이 일어나지는 않는지를 측정

타겟에서 전송되는 결과 정보는 일반적으로 저 수준 텍스트 기반 정보이기 때문에, 개발자가 직관적으로 성능 테스트 결과 정보를 파악하기는 어렵다. 따라서, 임베디드 소프트웨어의 성능 분석기는 사용자의 편리하고 직관적인 결과 분석을 위해 그래픽 기반의 다양한 그래픽 결과를 제공해야 한다. 본 논문은 가공되지 않은 저수준의 임베디드 소프트웨어에 대한 성능 측정 결과를 다양한 그래픽 형태의 결과로 출력하는 그래픽 기반 임베디드 소프트웨어 성능 분석기를 제안한다. 개발자나 사용자의 시각에서 그래픽 형태의 직관적이고 종합적인 성능 측정 정보는 성능 평가에 대한 신뢰성 판단에 빠른 결정을 유도할 수 있어 임베디드 소프트웨어 개발비용을 감소시킬 수 있다. 본 논문은 2장에서 관련 연구를 소개하고 3장 본론에서 제안하는 성능 분석기의 구성과 특징을 서술한다. 4장에서 구현 및 실험에 대해 설명하고 5장 결론 및 향후 연구방향으로 맺는다.

2. 관련 연구

AstonLinux의 CodeMaker는 윈도우환경에서 리눅스 기반 타겟시스템을 개발하는 통합개발환경이다. Remote Debugging기능과 Source Level Debugging 기능을 제공한다. 임베디드 환경에서 리눅스 유틸리티들을 선택하여 Ramdisk로 구성, 관리할 수 있는 기능을 제공한다. 통합개발환경 도구로, 특정 RTOS/Chip 벤더를 지원하기 때문에, 임베디드 소프트웨어 시험검증 도구로 사용할 수 없다. 또한, 리눅스 기반 유틸리티를 사용하기 때문에 일반적인 타겟 시스템 기반의 테스팅에 제약이 많다.

Soft4Soft의 RESORT 는 Software Metrics기반으로 패키지 S/W를 분석, 테스팅, 품질관리를 지원하는 솔루션 도구이다. 그러나, 이것은 일반 패키지 소프트웨어의 개발지원도구로, 임베디드 소프트웨어개발환경에서 사용할 수 없다.

3. 본론

그림 2는 본 논문에서 제안하는 성능 분석 도구의 전체 구조를 보여준다. 제안하는 성능 분석기는 개발 소프트웨어의 원시 코드를 타겟에 종속적인 코드가 삽입된 실행 코드로 크로스 컴파일 한 후, 타겟에서 성능 테스팅을 실시하기 위한 호스트/타겟 에이전트로 구성된다. 또한, 저수준의 성능 측정 결과 로그를 분석하는 프로파일 로그 분석기와 프로파일 로그 분석기로부터 가공된 결과 정보를 분류하고 GUI형태로 쉽게 변경할 수 있도록 API 형태로 제공하는 프로파일 변환기로 구성된다.

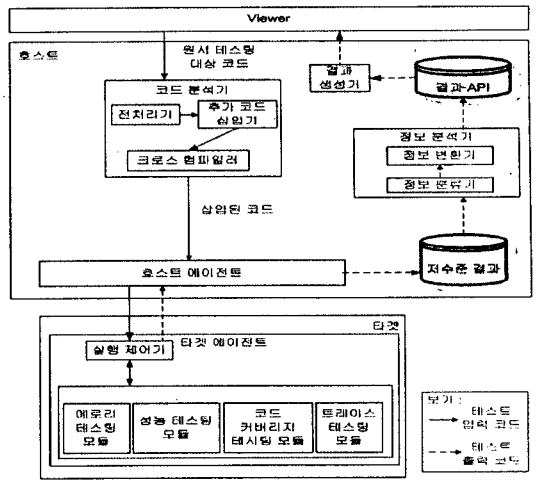

그림 2. 성능 분석 도구의 전체 구조

그림 2에서 정보 분석기는 정보 변환기와 정보 분류기로 구성된다. 정보 분리기는 타겟 에이전트에서 생성한 스트링 형태의 결과 정보를 종류에 따라 분류한다. 정보 변환기는 정보 분리기에 의해 테스팅 종류별로 분류된 스트링 형태의 결과 정보를 결과 생성기가 사용자의 요구에 맞는 그래픽 형태로 출력하기 위해 보다 사용하기 편리한 API로 변경한다.

그림 2에서 테스팅 대상 원시 코드는 어휘, 구문, 의미 분석 과정을 거친 후 소스 코드 삽입기 모듈에 의하여 필요한 위치에 테스팅 코드가 삽입된 새로운 소스 코드로 변환된다. 이렇게 만들어진 테스트를 위한 최종 소스 코드를 교차 컴파일러 모듈의 입력으로 전달하여 테스트 대상 프로그램이 수행될 환경에 적합한 목적 코드로 컴파일 하게 된다. 에이전트 모듈은 호스트에서 실행되는 호스트 테스팅 모듈과 타겟에서 실행되는 타겟 테스팅 모듈사이에서 테스팅 코드와 결과의 전송과 연결을 위한 통신 세션을 유지한다. 또한, 각 테스팅 프로파일에 따라 해당 테스팅 모듈들을 실행하고 제어하는 제어 역할을 수행한다. 데이터 분석기는 레포트 생성기가 사용자의 요구에 맞는 GUI 형태의 결과를 출력할 수 있도록 API 와 XML 형태로 테스팅 결과를 변경 저장한다.

생성기는 결과에 대해 사용자에게 그래픽 형태의 레포트를 생성하기 위한 모듈이다. GUI를 통해 사용자가 원하는 테스팅 레포트의 종류를 선택하면, 레포트 생성기는 데이터 분석기가 테스팅 결과에 대해 생성한 API와 XML 파일을 이용해 그래픽 형태의 결과를 출력한다.

4. 구현 및 실험

본 논문에서 제안하는 임베디드 소프트웨어를 위한 그래픽 기반 성능 분석기는 실행 환경에 독립적인 특성을 위해 Java로 구현되었다. 또한, 실험을 위해 타겟 머신은 임베디드 리눅스와 유사한 Velos OS를 탑재한 MSM 칩 보드를 이용하였으며, 대상 프로그램은 이동 단말기에서 쓰이는 C로 작성된 계산기 프로그램을 사용하였다.

표 2는 예제 계산기 프로그램에 대해 타겟에서 실행된 테스팅에 대한 스트링 형태의 저수준의 메모리 성능 결과 중 일부이다.

표 2. 가공되지 않은 저수준의 로그

```
index % time    self  children    called     name
                                    95000           func1 <cycle 1> [3]
[2]     0.0    0.00    0.00       95000           func2 <cycle 1> [2]
                                      900           func1 <cycle 1> [3]
-----------------------------------------------
                                      900           func2 <cycle 1> [2]
                0.00    0.00      1000/1000         main [13]
[3]     0.0    0.00    0.00        1900            func1 <cycle 1> [3]
                                    95000           func2 <cycle 1> [2]
-----------------------------------------------
Index by function name
   [3] func1                   [2] func2                   [1] <cycle
1>

@ <Location>:(Called function + Called Location)[Instruction Location]
+/- Address Size
= Start
@ ./ex-n:(mtrace+0x169)[0x80484e9] + 0x804a378 0x12
@ ./ex-n:[0x8048596] - 0x804a378
```

표 1은 프로파일러와 메모리 트레이서가 타겟 시스템에서 생성한 저수준 로그이다. 추출된 정보는 정보 변환기를 통해 API 형태의 자료구조로 변환한다.

그림 3은 표 1에 대해 변경된 API를 이용해 결과 생성기에 의해 출력된 그래픽 기반 메모리 관련 테스팅 성능 분석 결과이다.

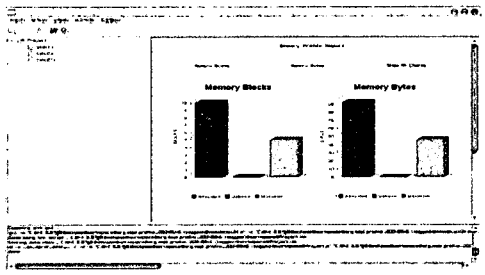

그림 3. 메모리 테스팅 결과

그림 3에서 메모리 성능 분석 결과는 메모리 트레이스 정보와 함수 프로파일 정보를 분석하여 메모리 누수와 메모리 사용량에 관한 정보를 제공한다.

5. 결론 및 향후 연구 방향

일반적으로 임베디드 소프트웨어에 대한 성능 테스트 결과는 저수준의 텍스트 기반 정보이기 때문에 즉, 테스트 결과에 대해 개발자가 직접 대화형 쉘과 자원 모니터를 통해 분석해야 한다. 개발자가 직접 분석하는 방법은 응용프로그램 개발 이외의 수고를 들게 하여 응용프로그램의 개발을 비효율적으로 만드는 약점이 있다. 본 논문은 다양한 임베디드 소프트웨어 개발에 있어 성능 평가 테스트 결과를 직관적이고 빠르게 분석하기 위한 그래픽 기반 성능 분석기를 제안하였다. 제안한 성능 분석기는 저수준의 결과 정보를 다양한 그래픽으로 쉽게 출력할 수 있도록 변경된 API를 이용한다. 본 논문에서 제안하는 그래픽 기반 성능 분석기를 통해 개발자나 사용자는 개발 임베디드 소프트웨어의 성능 평가 결과와 로그 정보를 직관적으로 분석할 수 있으며, 개발 시간과 노력을 줄여 임베디드 관련 소프트웨어 개발의 효율성을 증가 시킬 것으로 기대된다.

향후 본 논문에서 제안한 성능 분석기는 XML 형태의 결과 정보를 이용해 웹 기반의 성능 분석이 가능하도록 설계 구현 되어야 한다. 웹 기반 성능 분석기는 인터넷을 통해 환경에 독립적인 테스팅 결과 분석이 가능하도록 연구되어야 한다.

참고문헌

[1] 공기석, 손승우, 임채덕, 김홍남, "내장형 실시간 소프트웨어의 원격 디버깅을 위한 디버그에이전트의 설계 및 구현", *한국정보과학회 가을 학술발표논문집 Vol. 26. No 2*, pp.125~127, 1999.

[2] Dr. Neal Stollon, Rick Leatherman, Bruce Ableidinger, "Multi-Core Embedded Debug for Structured ASIC Systems", proceedings of DesignCon 2004, Feb, 2004.

[3] K. Weiß, T. Steckstor, C. Nitsch, W. Rosenstiel, "Performance Analysis of Real-Time-Operation Systems by Emulation of an Embedded System". 10th IEEE International Workshop on Rapid System Prototyping (RSP) Florida, USA, 1999.