韓國 컴퓨터情報學會 論文誌
第13卷 第7號, 2008. 12.

2008-13-7-1-3

협업 개발을 지원하는 임베디드 소프트웨어 성능분석도구 설계 및 구현

김 익 수*, 조 용 윤**

Design and Implementation of a Performance Evaluation Tool for Embedded Softwares on Collaborative Development Environment

Iksu Kim*, Yongyun Cho**

요 약

컴퓨팅 자원 사용에 제약이 많은 임베디드 소프트웨어 개발에 있어서 성능 분석 도구는 품질 개선을 위해 매우 중요한 역할을 한다. 그러나 기존 임베디드 소프트웨어 성능 분석 도구들은 단일 개발자의 교차개발 지원을 위한 성능분석 환경만을 제공하기 때문에 임베디드 소프트웨어를 위한 협업 성능분석 도구로서 활용될 수 없다. 본 논문에서는 협업 환경에서 임베디드 소프트웨어 성능 분석을 효과적으로 수행하기 위한 서버 기반의 새로운 성능 분석도구를 제안한다. 제안하는 성능분석 도구는 개발 소프트웨어에 대해 실행한 성능 분석로그를 협업 개발자들 기호에 맞는 다양한 그래픽 뷰로 생성하여, 개발자간의 신속하고 효율적인 정보 공유를 가능하게 함으로써 프로그램의 성능 향상을 위한 개발자의 성능 분석 환경을 크게 개선할 수 있다.

Abstract

A performance evaluation tool makes an important role in order to improve performance of an embedded software which has restricted computing resources. However, existing performance evaluation tools for embedded softwares cannot be used in collaborative development environment because they support only one developer with performance evaluation work under cross development environment. In this paper, we propose a performance evaluation tool for embedded softwares on collaborative development environment. A proposed tool is based on server and client model. It can have flexibility in offering and integrating the result information for the items. Through the suggested tool, developers can intuitively understand and analysis performance evaluation results each other.

▸ Keyword : Embedded Software, Performance Evaluation, Collaborative Development

• 제1저자 : 김익수 교지저자 : 조용윤
• 접수일 : 2008. 9. 30, 심사일 : 2008. 10. 29, 심사완료일 : 2008. 12. 24.
* (주)스카이컴 ** 숭실대학교

Ⅰ. 서 론

최근 IT 업계에서는 유비쿼터스 환경 구축을 위한 활발한 연구가 진행되고 있다. 유비쿼터스란 물이나 공기처럼 시공을 초월해 '언제 어디에나 존재한다'는 뜻의 라틴어로, 사용자가 컴퓨터나 네트워크를 의식하지 않고 장소에 상관없이 자유롭게 네트워크에 접속할 수 있는 환경을 의미한다. 유비쿼터스 네트워크에 접속하기 위해서 사용자들은 PDA나 휴대폰과 같은 모바일 기기를 사용할 수 있다. 모바일 기기는 휴대성을 위해서 작은 크기를 유지해야 하기 때문에 PC와 같은 범용 시스템에 비해 전력 소모가 낮은 프로세서와 적은 용량의 메모리를 사용해야 한다.

임베디드 소프트웨어는 PC와 같은 범용 컴퓨팅 시스템을 제외한, 제한된 시스템 자원 환경에서 특정 기능만을 수행하도록 제작된 소프트웨어를 의미한다. 임베디드 소프트웨어는 다양한 시스템 환경에서 소비전력, 신뢰성, 자원 관리 및 성능을 크게 고려하여 개발되어야 하기 때문에 소프트웨어에 대한 성능 분석은 매우 중요한 프로세스라 할 수 있다[1][2]. 임베디드 소프트웨어 개발자는 소프트웨어 개발에서 신속하고 효율적인 성능 분석을 위해 성능 분석 도구를 사용한다. 하지만 기존의 성능 분석 도구는 복잡한 텍스트 형태의 로그 정보만을 제공한다. 단순한 텍스트 형태의 로그 정보는 성능 분석에 있어 개발자가 효율적으로 이용할 수 없기 때문에 신속한 개발 환경을 제공하기 위한 다양한 형태의 뷰를 제공하는 것이 중요하다. 특히, 협업 환경에서 최적의 소프트웨어를 개발하기 위해서는 개발자간에 동일 소프트웨어에 대한 성능 분석 결과를 공유해야 하지만, 현재 임베디드 소프트웨어의 교차 개발 환경에서는 분산된 개발자 간의 정보 공유가 매우 어렵다.

이에 본 논문에서는 협업 환경에서 임베디드 소프트웨어 성능 분석을 효과적으로 수행하기 위한 서버 기반의 레포트 뷰 생성기를 제안한다. 제안 시스템은 하나의 성능 분석 로그에 대해 다양한 그래픽 뷰를 제공하여 개발자의 직관적 이해와 효율적인 분석을 가능하게 하며, 클라이언트/서버 구조를 통해 개발자간의 신속한 정보 공유를 가능하게 하여 프로그램의 성능 분석 프로세스 기간을 크게 단축시킬 수 있다. 아울러 본 논문은 네트워크 환경에서 발생할 수 있는 데이터 보안성 붕괴를 막기 위해 오픈소스 기반의 보안도구와 보안 메커니즘을 혼합 구성함으로써 데이터의 기밀성과 무결성을 보장한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하며, 3장에서는 제안 시스템의 구조에 대해 기술한다. 4장에서는 실험을 통해 시스템을 분석하며, 마지막으로 5장에서는 결론에 대해 기술한다.

Ⅱ. 관련 연구

2.1 임베디드 소프트웨어 개발 환경

임베디드 시스템은 시스템을 동작시키는 소프트웨어를 하드웨어에 내장하여 특수한 기능만을 제공하는 시스템을 의미한다. 대부분의 임베디드 시스템은 특정한 기능만을 제공하기 위해 개발되기 때문에 구조가 단순하며, 오랜 기간 동안 오류없이 안정적으로 구동되도록 설계된다. 따라서 개인용 PC에서 사용되는 소프트웨어보다 신중한 개발과 테스트 과정을 거쳐야 한다. 특히, 임베디드 시스템은 적은 컴퓨팅 자원만을 지원하기 때문에 제한된 자원에 대해 최적화된 성능을 발휘할 수 있도록 설계되어야 한다. 그러므로 개발자는 임베디드 소프트웨어가 사용하는 프로세스와 쓰레드 정보, 각 프로세스별 CPU 및 메모리 사용량과 같은 자원 사용 상태 정보를 분석해야 한다.

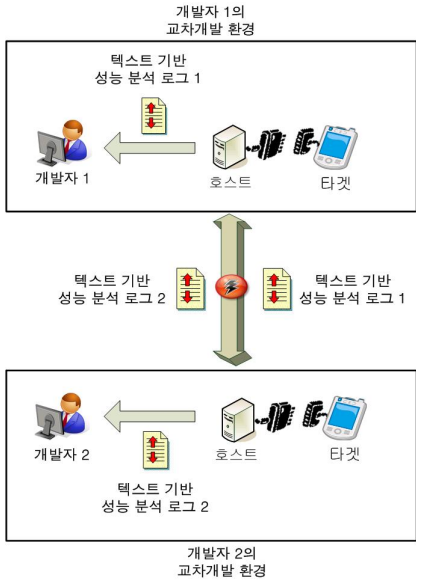


그림 1. 교차 개발 환경
Fig. 1 Cross development environment

일반적으로 임베디드 소프트웨어 개발은 그림 1과 같이 호스트/타겟 기반의 교차 개발 환경에서 이루어진다. 교차 개발 환경이란 실제 소프트웨어가 수행될 시스템과 개발하는 시스

템이 다른 개발환경을 의미한다. 타겟은 개발 대상인 하드웨어로서 독립적으로 동작하지 않으며, 호스트에서 운영체제, 디바이스 드라이버, 응용 프로그램을 개발한 후 다운받아야 동작할 수 있다. 교차 개발 환경에서의 실제 프로그램 성능 분석은 타겟의 성능 분석 모듈에 의해 실행되며, 개발자는 호스트에서 성능 분석 결과를 분석한다.

최근의 임베디드 소프트웨어 개발은 그 규모와 복잡도가 크게 증가하고 있기 때문에, 다수의 개발자가 함께 성능 분석을 실시할 수 있는 협업 환경이 요구되고 있다. 소프트웨어 협업 개발을 위해서는 다수의 개발자들에게 신뢰성 있는 접근 보안을 제공하고 성능 분석 과정 및 결과에 대해 동기화된 자료 공유를 제공할 수 있는 성능 분석 도구가 요구된다. 그러나 그림 1과 같이 기존의 임베디드 소프트웨어 교차 개발 환경은 물리적으로 독립된 성능 분석 장비와 소프트웨어를 개별적으로 제공하는 구조로서, 협업을 위한 다수의 개발자들간의 정보 공유를 본질적으로 고려하지 않았다. 따라서, 기존 임베디드 소프트웨어를 위한 성능 분석도구는 다수의 개발자들간 협업 작업에 적합하지 않다.

2.2 임베디드 소프트웨어 성능 분석 도구

임베디드 소프트웨어의 개발 효율을 높이고 편리한 개발 환경을 제공하기 위해 개발자는 다양한 성능 분석도구를 이용할 수 있다. gprof[3]는 유닉스와 리눅스의 개발 도구 패키지에 기본으로 포함되어 있으며, 가장 널리 사용되고 있는 소프트웨어 성능 평가 도구이다. 유닉스 계열 표준 컴파일러인 cc 명령에 '-pg' 옵션을 전달하면, 코드 생성 단계에서 성능 평가를 위한 코드들을 삽입하게 된다. 이렇게 만들어진 실행 프로그램을 일반적인 응용 프로그램과 동일한 방법으로 실행시키면, 프로그램 종료 직전에 텍스트 형태의 로그 파일을 남긴다. 이 파일을 gprof를 통해 분석하면 코드 커버리지 관련 성능 평가 결과를 얻을 수 있다[4]. 그러나 출력되는 결과가 각 함수의 실행 통계를 나타내는 함수들 간의 호출 관계를 나타내는 Call Graph만으로 제한적이다. 따라서 개발자가 원하는 각 자원 활용 성능 평가 결과에 대한 다양한 형태의 뷰를 얻을 수 없다. 또한, 로그 수집 과정에서 표준 C 라이브러리 이외에 GNU C 라이브러리(glibc)에 포함된 함수들을 다수 호출하기 때문에, GNU C 라이브러리가 포팅 되지 않은 임베디드 운영체제에서는 사용이 불가능하다.

FunctionCheck[5]는 gprof의 부족한 기능 몇 가지를 보완한 공개 도구이다. 구체적으로 성능 평가 데이터 파일의 이름 변경 가능, 멀티 쓰레드와 멀티 프로세스의 지원, 메모리 성능 평가 출력 기능 등이 추가되었다. 그러나 기본적으로 gprof와 같은 방법으로 성능 평가 정보를 수집하므로 동일한 환경 제한과 오차를 가질 수밖에 없다.

Soft4Soft 사의 RESORT[6]는 소프트웨어 매트릭스(metrics) 기반으로 패키지 소프트웨어를 분석, 테스트하고 성능 평가 결과에 대해 다양한 그래픽 뷰를 제공하여 품질관리를 지원하는 도구이다. 그러나 이 도구는 범용 패키지 소프트웨어의 개발을 지원하는 도구로, 적은 자원을 갖고 교차 개발을 지원하는 임베디드 소프트웨어 개발 환경에서는 사용이 불가능하다.

AstonLinux사의 CodeMaker[7], 미지리서치 사의 Linu@[8], MontaVista 사의 CDK[9], Lineo 사의 Embedix SDK[10] 등은 모두 윈도우 또는 리눅스 호스트 환경에서 임베디드 리눅스를 타겟으로 하는 임베디드 소프트웨어를 개발하는 통합 개발 환경이다. 그러나 이들 도구는 응용 소프트웨어의 개발 보조와 디버깅(debugging)이 주된 기능으로써, 소프트웨어 실행 전반에 걸친 성능 평가 결과를 산출해서 그래픽 형태의 레포트 뷰를 보여주는 성능 평가 도구로서의 기능은 적다.

앞서 설명한 임베디드 소프트웨어를 위한 성능 분석도구는 모두가 단일 사용자를 위한 실시간 테스팅 및 성능 평가 기능을 제공한다.

2.3 성능 분석 도구의 보안 요구 사항

분산된 환경에서 네트워크를 통해 데이터를 전송할 경우, 데이터는 공격자에 의해 무결성과 기밀성이 파괴될 수 있다. 이러한 환경에서 데이터의 무결성과 기밀성을 유지하기 위해 네트워크 프로그램 개발자들은 SSH를 사용할 수 있다.

SSH(Secure Shell)은 보안 로그인 쉘로서 과거의 telnet, rsh, rlogin 등이 평문의 데이터를 전송하는데 반해, SSH는 네트워크에 존재하는 두 단말 사이의 연결을 보호하기 위해 데이터를 암호화하여 전송한다. SSH를 통한 안전한 통신이 이루어지기 위해서는 상호간에 데이터를 암호화 및 복호화하기 위한 비밀키를 공유해야 하며, 이 키는 다른 사용자에게 노출되어서는 안 된다. 만일 이 키가 공격자에게 노출될 경우, 공격자는 획득한 비밀키를 이용하여 암호화된 데이터를 복호화 할 수 있다.

안전한 비밀키 공유를 위해서 서버는 우선 개인키와 공개키를 생성하고 공개키를 클라이언트에 전송한다. 클라이언트는 자신의 개인키를 생성하고 서버로부터 수신한 공개키를 이용하여 개인키를 암호화한다. 암호화된 클라이언트의 개인키는 서버에 전송되며, 서버의 공개키로 암호화되었기 때문에 단지 서버만이 클라이언트의 개인키를 복호화할 수 있다. 서버와 클라이언트는 서로 간에 데이터를 안전하게 송수신하기

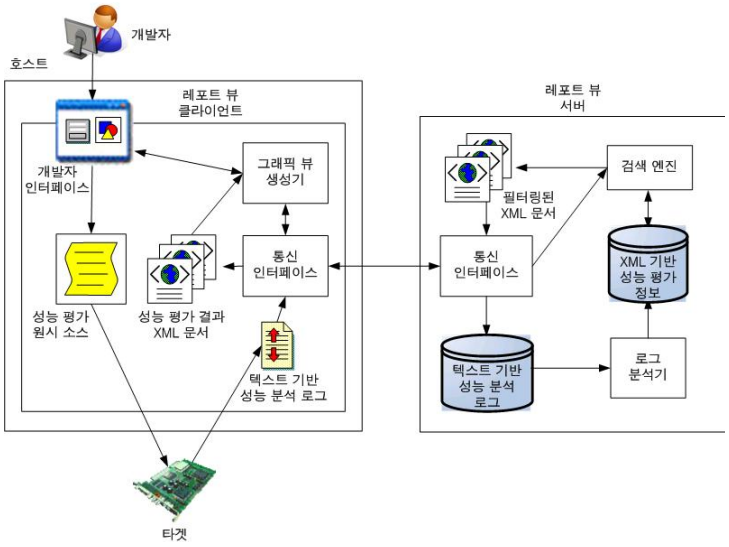


그림 2. 협업을 지원하는 임베디드 소프트웨어 성능 분석기의 전체 구성도
Fig. 2 Performance evaluation tool for embedded softwares on collaborative development environment

위해 공유된 비밀키를 이용하여 데이터를 암호화하고 복호화한다. 암호화된 데이터는 공격자가 중간에서 데이터를 가로채어 읽을 수 없기 때문에 기밀성이 보장된다. 또한, 공격자가 중간에 데이터를 가로채어 임의로 변조할지라도 수신자는 해당 데이터를 정상적으로 읽을 수 없기 때문에 데이터의 변조 유무를 파악할 수 있어 무결성이 보장된다.

III. 협업을 지원하는 임베디드 소프트웨어 성능 분석기

3.1 시스템 구조

제안 시스템은 협업 환경에서 다수의 개발자에게 성능 분석 결과를 제공하기 위해 클라이언트/서버 구조를 가진다. 그림 2는 협업 성능 분석을 위한 제안 시스템의 구조도이다. 그림 2에서와 같이 성능 분석을 위한 레포트 뷰 클라이언트는 개발자로부터 성능 분석을 위한 원시 소스를 입력받아 타겟의 성능 분석 모듈에 전달하며, 성능 분석 모듈은 원시 소스의 성능 실험을 실행한 후 텍스트 기반의 성능 분석 로그를 생성한다. 이 로그는 다수의 개발자간의 성능 분석 정보를 공유하기 위해 서버에 전달된다. 서버는 개발자들의 성능 분석 결과를 통합 관리하며, 개발자들의 성능 분석 정보에 관한 요청에 부합하는 XML 문서를 제공한다. 개발자는 클라이언트의 GUI를 통해 이해하기 쉬운 그래픽 기반의 다양한 뷰를 제공받음으로써 성능 분석 프로세스 기간을 단축시킬 수 있다.

3.2 로그 분석기

타겟으로부터 생성된 성능 분석 로그는 저수준의 텍스트 기반의 로그이다. 다양한 관점에서 원시 소스의 성능을 분석하기 위해서는 텍스트 형태의 성능 분석 로그 정보를 효율적으로 처리할 수 있는 구조화된 문서로의 변환이 필요하다. 그림 3은 제안하는 성능 분석도구를 이용하여 저수준의 로그 정보를 XML 변환기를 통해 XML 기반의 성능 분석정보로 변환하는 과정을 나타낸다.

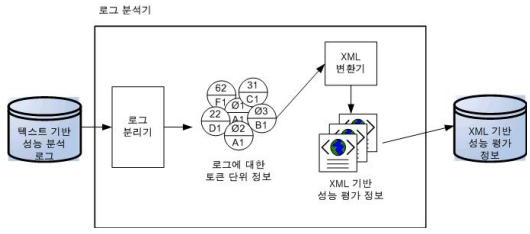


그림 3. XML 기반 성능 평가 정보 생성을 위한 로그 분석기 구성도
Fig. 3 Log separator for creating XML-based performance evaluation information

그림 3의 로그 분석기의 기능은 다음과 같다.

1) **로그 분리** : 타겟에서 생성된 성능 분석 로그가 로그 분리기를 통해 의미 있는 토큰 단위로 나뉘고 성능 분석 항목에 따라 분류된다.
2) **토큰 변환** : 로그 분리 단계를 통해 성능 분석 항목별로 분류된 토큰이 XML 변환기를 통해 항목별 엘리먼트로 구조화된 XML 문서로 변환된다.

로그 분리기는 레포트 뷰 클라이언트로부터 전달받은 텍스트 기반 성능 분석 로그 정보를 구조화된 XML 문서로 변환하기 위해 미리 정의된 토큰 단위 정보로 변환된다. 다양하게 실시된 성능 평가 결과는 항목별로 같은 종류의 토큰 그룹으로 분류되어 미리 정의된 XML DTD를 통해 XML 성능 평가 문서로 변환된다. 저장된 XML 성능 평가 정보는 표준 XML 파서를 통해 유용한 자료구조로서 레포트 뷰 생성기의 입력으로 제공된다. 제안한 생성기는 레포트 뷰 생성을 위해 하나의 내용 명세 정보와 화면 구성 정보를 고정적으로 연결하는 방식을 탈피하여 개발자가 레포트 뷰 구성을 위한 정보를 선택할 수 있는 방법을 제공하였다. 이를 위해 로그 분석기는 성능 평가 로그를 정보 이용 효율이 높은 XML로 변환 저장하였다. 이것은 내용 명세 정보로서의 성능 평가 결과를 로그의 형태보다 훨씬 효율적으로 접근하고 추출할 수 있는 방법을 제공한다[11][12].

3.3 검색 엔진

성능 분석 도구의 서버 환경은 다수의 개발자들에게 효율적인 협업 환경을 제공하기 위해 개발자들의 요구에 부합하는 정보를 제공하는 역할을 해야 한다. 실제로 개발자들은 하나 이상의 임베디드 소프트웨어 개발 업무에 할당되기 때문에 레포트 뷰 서버는 복수의 임베디드 소프트웨어에 대한 성능 분석 로그를 보관한다. 또한, 협업 환경에서는 여러 개발자들이 특정 임베디드 소프트웨어 개발에 할당되기 때문에 각 개발자로부터 전달받은 상이한 성능 분석 로그를 보관한다.

검색 엔진은 로그 분석기를 통해 생성된 XML 기반 성능 분석 정보로부터 개발자의 입력 파라미터에 따라 성능 분석 정보를 검색하여 제공하는 역할을 한다. 클라이언트가 서버에 전달하는 개발자 입력 파라미터는 프로그램 이름과 버전 필드로 구성된다. 프로그램 이름은 복수의 소프트웨어 성능 분석 정보들로부터 특정 소프트웨어 성능 분석 정보만을 추출하기 위해 사용되는 값이다.

개발자는 소프트웨어의 성능 분석이 필요할 때마다 서버로 성능 분석 로그를 전송하며, 이 로그들은 차후에 필요한 소프트웨어 성능 비교를 위해 로그 분석기를 거쳐 서버에 보관된다. 개발자 입력 파라미터의 버전 필드는 개발자가 원하는 버전의 성능 분석 로그를 다른 개발자로부터 이전에 생성된 성능 분석 로그와 비교 분석하기 위해 사용된다. 이는 성능 분석 로그를 업로드하지 않고 단지 기존의 성능 분석 히스토리를 추출하기 위한 방법으로 사용된다.

3.4 통신 및 문서 관리 메커니즘

임베디드 소프트웨어 개발에 있어 성능 분석 정보는 매우 중요한 정보이기 때문에 외부로부터 보호되어야 한다. 이에 제안 시스템은 안전한 통신 환경을 구축하기 위해 리눅스 기반의 임베디드 시스템 개발환경에서 쉽게 이용 가능한 SSH를 이용한다. SSH는 데이터의 기밀성과 무결성을 보장할 뿐만 아니라 상호간의 인증 기능도 제공한다. 그리고 데이터의 효율적인 송수신을 위해 압축 기능을 제공하며, scp 명령을 통한 암호화된 파일 전송 기능을 제공한다.

클라이언트 통신 인터페이스는 사용자로부터 입력받은 텍스트 기반 성능 분석 로그 파일을 서버에 전달하기 위해 scp 명령을 수행하는 스크립트로 구현되며, 개발자 요구에 맞는 레포트 뷰를 생성하기 위한 입력 파라미터를 서버에 전달한다.

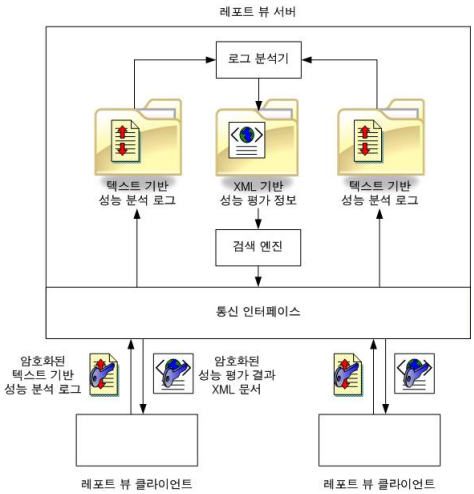


그림 4. 통신 및 문서 관리 메커니즘
Fig. 4 Mechanism for communication and document management

그림 4는 레포트 뷰 서버와 클라이언트 간의 통신 및 문서 관리를 위한 메커니즘을 나타낸다. 그림 4에서 서버와 클라이언트는 데이터 암호화를 위해 세션키를 공유한다. 클라이언트와 서버 통신 인터페이스는 네트워크를 통해 송수신되는 파일의 무

결성과 기밀성을 유지하기 위해서 파일을 세션키로 암호화하여 전송하며, 수신된 암호화 파일은 이 키를 통해 복호화된다.

리눅스에서는 다중 사용자의 원활한 데이터 공유 및 처리를 위해 /tmp 디렉토리에 파일을 보관하지만, 개발자의 임의 접근으로 인한 데이터 무결성을 보장할 수 없다. 제안 시스템은 정확한 성능 평가 정보를 보장하기 위해 여러 개발자에 의해 전송된 성능 분석 로그들을 개발자 계정 디렉토리에 별도로 저장하여 관리한다. 로그 분석기는 각 개발자가 생성한 성능 분석 로그에 접근하기 위해 루트권한의 프로세스로 실행되며, 로그 분석기를 통해 생성된 성능 평가 정보는 개발자에 의해 전달된 입력 파라미터에 따라 추출된다. 따라서 개발자들은 서버에 의해 생성되는 성능 분석 결과와 이를 이용한 레포트 뷰 결과를 신뢰할 수 있다.

3.5 그래픽 뷰 생성기

그래픽 뷰 생성기는 개발자가 다양한 관점에서 신속하고 분석이 용이한 그래픽 기반의 레포트 뷰를 생성한다.

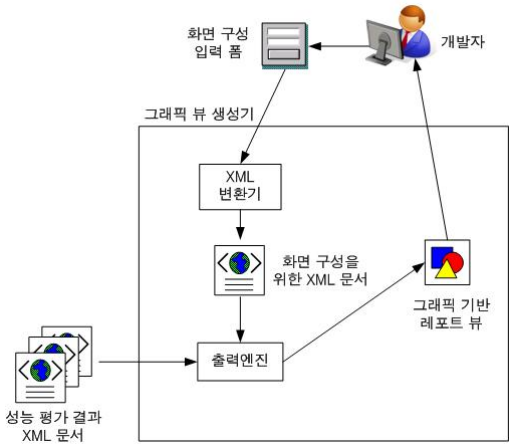


그림 5. 그래픽 뷰 생성기 구성도
Fig. 5 Graphic view generator

그림 5와 같이 개발자는 레포트 뷰 서버로부터 필터링 된 XML 문서를 수신하며, 자신이 원하는 형태의 그래픽 뷰로 변환하기 위해 GUI로 구성된 폼에 파라미터를 입력할 수 있다. 즉, 파라미터는 개발자가 자신의 개발 시점에서 요구되는

표 1 로그 정보에 대해 생성된 XML 문서
Table 1 XML Document for log informations

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE profile SYSTEM "D:₩project(testProgram)₩profiler.dtd">
<profile time="2008-08-17-14:23" programName="Test Program">
<memoryProfile>
                    :
</memoryProfile>
<traceProfile>
        <traceEvent type="call">
                    :
        </traceEventype>
</traceProfile>
<performanceProfile>
        <programTotalExecutedTime>650000</programTotalExecutedTime>
        <functionPerformance>
            <callCount>1</callCount>
            <executedTime>5000</executedTime>
            <executedMinTime>5000</executedMinTime>
            <executedMaxTime>5000</executedMaxTime>
            <functionTotalExecutedTime>76900</functionTotalExecutedTime>
        </functionPerformance>
</performanceProfile>
<codeCoverageProfile>
        <executedBlock>
            <block index="9" startLine="38" endLine="60" startColomn="1" endColomn="1"/>
            <block index="13" startLine="52" endLine="54" startColomn="9" endColomn="2"/>
        </executedBlock>
</codeCoverageProfile>
</profil>
```

성능 분석 항목에 대해 원하는 그래픽 형태로 성능 분석 정보를 보기 위한 입력 값으로서, 개발자는 입력 값에 따라 성능 분석 대상 소프트웨어의 메모리와 CPU 사용에 대한 정보나, 코드 범위 테스팅, 트레이스 테스팅 결과를 다른 개발자와 비교할 수 있다. XML 변환기는 입력 파라미터를 기반으로 화면 구성을 위한 XML 문서를 생성하며, 출력엔진은 화면 구성을 위한 XML 문서와 성능 분석 결과 XML 문서를 입력받아 막대그래프나 원형 그래프와 같은 그래픽 기반의 레포트 뷰를 생성한다.

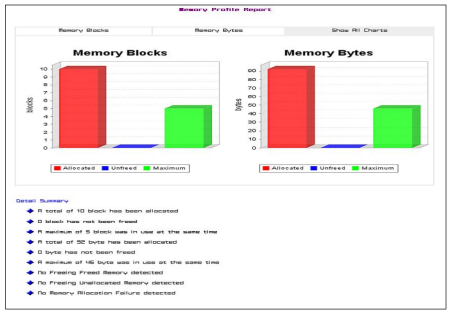


그림 6. 메모리 성능 분석 그래픽 뷰(막대 그래프)
Fig. 6 A graphic view for memory performance evaluation(Bar graph)

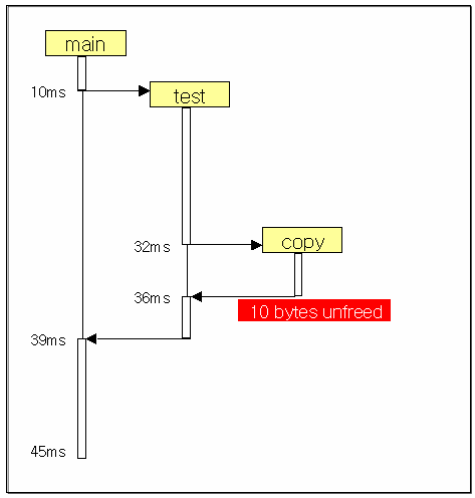
Ⅳ. 실험 및 분석

제안하는 임베디드 소프트웨어 성능 분석 도구의 분석 실험을 위해 일반적인 시스템 소프트웨어 언어인 ANSI C 소스를 성능 평가 대상으로 실험하였다. 타겟 환경에서는 ARM9 CUP에 Velos OS[13]를 탑재한 HRP-SC2410 (Ami) 개발 보드를 사용하였으며, 임베디드 소프트웨어 성능 분석을 위한 테스팅 하드웨어 모듈로서 Lauterbach TRACE-32 ICD (In-Circuit-Debugger)를 사용하였다. 호스트측에서는 Java2TM SDK v1.4.2으로 개발하여 제공되는 그래픽 기반 성능 분석환경에서 3명의 개발자들이 협업하여 성능 평가 작업을 실시하였다. 성능 평가 대상 원시 코드는 3개의 함수로 이루어진 간단 C 코드이다. 실험 대상 C 프로그램을 프로젝트(project)로 빌드(build)한 후, 제안하는 성능분석 도구에 포함된 GNU 교차 개발 툴 체인(cross-development tool chain)[14]이 소스 파일에 대하여 순서대로 어휘 분석, 구문 분석, 코드 수정, 언파싱(unparsing), 컴파일을 수행하며[15], 모든 소스 파일이 컴파일 과정까지 완료되면 각 목적 코드들을 함께 링크하여 타겟에서 실행될 수 있는 실행 코드를 생성한다. 생성된 실행 코드에 대해 타겟의 성능 평가 엔진은 성능 평가를 실행하여 그 결과를 로그 형태로 생성한다.

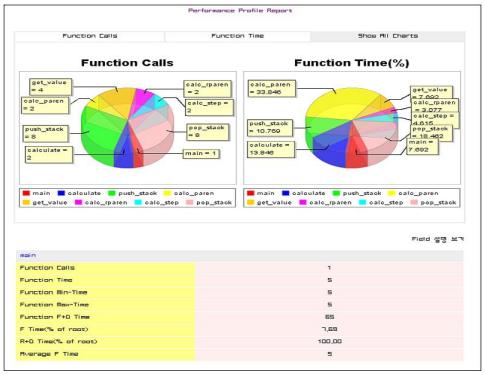
표 1은 XML 변환기가 저수준의 성능 분석 로그 정보에 대해 생성한 XML 문서의 일부분이다. 로그 분석기에 의해 생성된 하나의 XML 기반 성능 분석 결과 문서는 다수의 개발자에 의해 공유되어, 그림 5에서 설명한 그래픽 뷰 생성기에 의해 다양한 형태의 독립적인 그래픽 뷰로 변환된다. 그림 6과 7은 그래픽 뷰 생성기가 표 1의 XML 문서와 3명의 협업 개발자1, 2, 3들이 각자 자신이 입력한 화면구성 정보를 이용하여 메모리 및 함수관련 성능분석 정보를 막대그래프 형태와 순차 다이어그램 및 파이 그래프 형태로 생성한 각각의 그래픽 뷰이다.

그림 6의 성능 분석 그래픽 뷰는 협업 개발자1이 생성한 메모리 관련 성능분석 그래픽 뷰이다. 그림 6의 그래픽 뷰를 통해 협업 개발자들은 성능 분석 대상 소프트웨어가 어느 정도의 메모리를 사용하며, 메모리 누수 발생의 가능성은 없는지, 그리고 메모리 관련 코드에서 발생한 에러의 정확한 원인이 무엇인지 등을 판단할 수 있다. 또한, 협업 개발자들은 메모리 프로파일 결과를 참고로 하여, 메모리 사용에 문제가 있는 코드들을 쉽게 찾아내 수정할 수 있다.

그림 7은 실험에 참가한 다른 2명의 협업 개발자인 개발자 2와 3이 함수의 호출에 관련된 성능 평가 결과를 각각 자신들이 입력한 그래픽 화면 구성정보에 따라 시퀀스 다이어그램(sequence diagram) 형식과 파이 그래프(pie graph) 형식으로 생성한 그래픽 뷰를 나타낸다.



(a). 함수 관련 성능 분석 그래픽 뷰(시퀀스 다이어그램)
(a). A graphic view for performance evaluation related in functions(Sequence diagram)



(b). 함수 관련 성능 분석 그래픽 뷰(파이 그래프)
(b). A graphic view for performance evaluation related in functions(Pie graph)
그림 7. 다양한 함수 관련 성능 분석 그래픽 뷰
Fig 7. Various graphic views for performance evaluation

그림 7의 그래픽 뷰는 어떤 함수가 자주 실행되거나 실행 시간을 많이 소요하여 병목 현상을 발생시키는지 확인할 수 있는 정보를 제공한다. 파이 그래프 형식의 그래픽 뷰는 대상 소프트웨어에 포함된 각 함수의 실행 횟수와 실행 시간에 대해 시퀀스 다이어그램 보다 더욱 보기 편리한 통계 형태의 테이블 자료를 함께 제공한다. 또한, 전체 프로그램 실행시간 동안에 각 함수의 실행 여부에 대한 통계 정보를 제공하여 전체적인 함수 관련 정보를 한눈에 비교 분석할 수 있는 기회를 제공한다. 그러나 호출 순서를 직관적으로 파악하기에는 수치적인 통계표는 부적절할 수 있다. 따라서 협업 개발자 2가 생성한 시퀀스 다이어그램 형태의 그래픽 뷰는 소프트웨어가 실행된 후 어떤 시간에 어떤 순서로 각 함수의 호출이 일어났는지에 대한 성능 분석 정보를 협업 개발자 3이 생성한 파이 그래프 형태의 성능분석 그래픽 뷰보다 더욱 직관적으로 알 수 있는 기회를 제공 할 수 있다. 이로부터 각 함수가 언제 어떤 경로를 거쳐 몇 번이나 실행되었으며, 비정상적으로 많이 호출되거나 시간이 많이 소요되지는 않는지 확인이 가능하다. 그러나 시퀀스 다이어그램은 일반적으로 프로그램의 실행에 대한 가장 자세한 정보를 제공하지만, 프로그램의 실행 시간이 길어지면 정보의 양이 지나치게 많아져서 개발자가 분석하기 어려울 수 있다. 따라서 협업 개발자는 자신이 생성한 특정 그래픽 뷰 뿐만이 아니라 다른 협업 개발자들이 자신의 기호에 따라 생성한 다양한 형태의 성능분석 그래픽 뷰를 이용하여 풍부한 성능분석을 수행 할 수 있다. 이러한 성능분석 효율성은 분석 대상 임베디드 소프트웨어의 규모와 복잡도가 클수록 그 기대효과가 더욱 증가될 것이다.

표 2 기존 성능분석도구와의 비교
Table 2 Comparison between suggested system and existing systems

보기 : o -우수, △ - 양호, X - 부족

BMT 기준	제안 시스템	FUNCTION CHECKER	RESORT	CODE MAKER
테스팅 자동화 수준	O	O	△	O
리얼타임 성능분석 기능	O	O	O	O
테스팅툴 편리성	O	O	△	△
GUI 기반 다양한 리포팅 기능	O	△	△	△
보안기능의 유연성	O	△	△	△
원격 테스팅 기능	O	X	X	X

표 2는 제안하는 시스템과 기존 임베디드 소프트웨어 성능분석 도구들 간의 다양한 기능 비교를 실시한 결과이다. 표 2에서 보는바와 같이 기존 성능분석도구는 기본적인 성능분석 기능에 있어서 제안 시스템과 동일한 성능을 보이는 것을 볼 수 있으나, 보안 기능 및 원격 테스팅 기능에 있어서 부족한 기능을 보이는 것을 확인할 수 있다. 따라서 제안하는 시스템은 앞서 설명한 다양하고 안정된 보안기능을 기반으로 서버/클라이언트 형태의 원격 테스팅 기능을 통해 장소와 시간에 관계없이 자유로운 임베디드 소프트웨어 테스팅이 가능하여 관련 응용프로그램 개발에 도움이 될 것으로 기대된다.

V. 결론

일반 소프트웨어와는 달리 임베디드 소프트웨어는 시스템 자원의 사용에 많은 제약이 따르기 때문에 성능 분석이 매우 중요하다. 최근, 임베디드 소프트웨어 개발요구의 복잡도가 증가함에 따라 다수의 개발자가 공동개발에 참여할 수 있는 협업 개발 환경을 지원하는 성능 분석 도구가 요구된다. 그러나 기존의 성능 분석 도구들은 단일 사용자에게 개별적인 교차개발을 지원하는 소프트웨어 성능 분석 환경만을 제공하기 때문에, 다수의 개발자 간의 정보 공유가 매우 어렵다. 이에 본 논문에서는 협업 환경에서 임베디드 소프트웨어 성능분석을 효과적으로 수행하기 위한 클라이언트/서버 기반의 성능분석 도구를 제안하였다. 제안하는 성능분석 도구는 전체 개발자에 의해 수행된 임베디드 소프트웨어 성능 분석 정보를

수집하고, 다양한 관점의 분석을 신속하고 용이하게 하기 위한 그래픽 기반의 레포트 뷰를 제공한다. 또한, 제안하는 성능분석 도구는 협업 통신 모듈을 통해 다수의 개발자에 의해 생성된 성능 분석 정보에 대한 접근 권한 제어와 작업 동기화를 보장하며 공유정보에 대한 신뢰성을 향상 시킨다. 따라서 제안 시스템은 분산된 협업 환경에서 임베디드 소프트웨어 개발의 성능 분석 프로세스 기간을 크게 단축시킬 수 있을 것으로 기대된다.

참고문헌

〔1〕 Bart Broekman, Testing Embedded Software, Addison-Wesley, pp.128-129 Dec. 2002.

〔2〕 L. Hatton, "Embedded Software Testing", Software Testing Congress, 2000.

〔3〕 GNU Profiler (GPROF) and reference page, available at http://www.gnu.org/software/binutils/manual/ gprof-2.9.1/html_mono/gprof.html

〔4〕 Mustafa M. Tikir, JeffreyK. Hollingsworth, "Efficient Instrumentation for Code Coverage Testing", International Symposium on Software Testing and Analysis Proceedings of the ACM SIGSOFT, pp.86-96, 2002.

〔5〕 FunctionCheck, available at http://www710.univ-lyon1.fr/%7Eyperret/fnccheck/doc.html

〔6〕 RESORT, http://www.soft4soft.com/

〔7〕 CodeMaker, http://www.astonlinux.com/

〔8〕 Linu@, http://www.mizi.com/ko/

〔9〕 CDK, http://www.mvista.com/

〔10〕 Embedix SDK, http://www.lineo.com/

〔11〕 정우식, 도경화, 전문석, "IDS/Firewall/ Router 통합 로그 분석기 설계", 한국컴퓨터정보학회, 8권 1호, pp. 48-56, 2003.3

〔12〕 정강용, 박나연, "웹서버의 로그파일 분석에 의한 웹 서비스 활용에 관한 연구", 한국컴퓨터정보학회 ,5권 1호, pp.32-38, 2000.3

〔13〕 Velos, http://www.velos.co.kr/

〔14〕 GNU Tool Chain and C Compiler (GCC) web page, available at http://gcc.gnu.org.

〔15〕 Alfred V. Aho, Ravi Sethi, Jeffery D. Ullman, Compilers: principles, techniques, and tools, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1986.

저 자 소 개

김 익 수

2000년 2월 숭실대학교 컴퓨터학부 졸업.

2002년 2월 숭실대학교 컴퓨터학과 석사 졸업.

2008년 2월 숭실대학교 컴퓨터학과 박사 졸업.

2006년 1월~현재 (주)스카이컴 과장.

〈관심 분야〉 정보보호, 시스템 보안, 네트워크 보안, 시스템 소프트웨어.

조 용 윤

1995년 시립 인천대학교 전자계산과 졸업(학사)

2006년 숭실대학교 대학원 전자계산학과 졸업 (공학박사)

2006년~ 현재 숭실대학교 연구원

〈관심 분야〉 컴파일러, 프로그래밍 언어, 프로그래밍 환경, XML, 임베디드 시스템