webOS 기반 스마트 TV 에서의 병렬처리 가능성 연구

전용권, 구동훈, 나병국, 윤성로
서울대학교 전기정보공학부
e-mail : sryoon@snu.ac.kr

An Applicability Study on Parallel Computing for webOS-based Smart TV

Yongkweon Jeon, Donghoon Koo, Byunggook Na, Sungroh Yoon
Dept. of Electrical and Computer Engineering, Seoul National University

요 약

전자제품의 스마트화 열풍으로 임베디드 시스템의 하드웨어 및 소프트웨어의 발전이 경쟁적으로 이루어지고 있지만, 하드웨어 발전 속도에 비해 그 활용도는 미진한 편이다. 특히, 스마트 TV 는 대형 스크린을 갖고 있다는 장점이 있고, 사물인터넷 시대의 중추 역할을 할 것으로 기대되기 때문에 많은 계산의 신속한 처리를 요구 받을 가능성이 크다. 따라서 본 논문에서는 webOS 기반 스마트 TV 에서, 계산자원을 충분한 활용하기 위한 병렬처리 가능성을 확인하고자 webOS 시스템을 프로파일링하고 그 결과를 분석하였다.

1. 서론

최근 스마트 폰을 시작으로 다양한 전자제품들이 스마트화 되고 있다. 여러 전자 기기들 중 TV 는 세대를 넘어 인간에게 가장 친숙한 기기로 여러 전자 회사들은 앞다투어 각기 자신들만의 방식으로 스마트 TV 를 출시하고 있다. LG, Samsung 등 기존 TV 를 만들던 전자회사들은 기존에 있던 '일반' TV 를 스마트화 하기 위해, TV 내부에 OS 를 설치하고 다양한 컨텐츠 공급 및 접근성 높은 UI (User Interface) 개발에 열을 올리고 있다. 반면에 '일반 TV'를 생산하지 않았던 Google, Apple 등의 회사들은 우회적인 방식으로 스마트 TV 시장에 뛰어들고 있다. Google 은 기존 TV 제조사 혹은 IPTV 업체와의 제휴를 통해 자신들의 컨텐츠 및 스마트 TV 환경을 제공하고 있으며, Apple 은 별도의 set-top box 를 TV 에 연결시켜 '일반 TV'를 스마트화 되도록 하고 있다 [1], [2].

각 회사들의 스마트 TV 제품 개발에 대한 접근 방식은 다양하지만, 공통적으로 이들이 제공하는 기능은 기존 스마트 모바일 기기와 유사하다. 인터넷 연결을 통해 스트리밍 방식의 다양한 영상을 제공하며, 응용프로그램 (Applications) 설치를 통해 사용자화 (customizing) 할 수 있도록 하고 있다. 뿐만 아니라, 모바일 기기와의 무선 연결을 통해 화면 공유 및 데이터 공유 기능을 제공함으로써, 상대적으로 대형화면을 갖추고 있는 TV 의 장점을 폭넓게 활용할 수 있도록 도움을 준다 [1], [2].

스마트 TV 에는 Embedded CPU (Central Processing units) 및 GPU (Graphics Processing Units)가 탑재되어 있으며, 이들의 cores 수 및 처리 속도의 증가로 연산 성능이 점차 좋아지고 있다. 그러나 현재 multi-core CPU 는 multi-tasking 및 background job 처리를 위해서만 활용되고 있고, GPU 는 렌더링 작업에만 참여하고 있는 등 그 활용성에 제한을 받고 있다. 그런데 스마트 TV 는 모바일 기기와 달리 대형 화면을 갖추고 있어, 활용도 측면에서 훨씬 다양하고 복잡한 응용프로그램을 실행시킬 가능성이 매우 크다. 또한 TV 는 집안의 핵심 기기로써, 다가올 사물 인터넷 (Internet of Things – IoT) 시대에서 중추 역할을 할 것으로 기대되고 있다 [3]. 즉, 스마트 TV 에서 compute-intensive 한 컨텐츠가 많이 늘어날 것으로 예상할 수 있으며, 이것들은 복잡한 연산을 실시간으로 처리 (real-time processing) 해 줄 것을 스마트 TV 에 요구할 것으로 보인다. 스마트 TV 는 Mobile 기기와 달리 공간의 제약을 크게 받지 않기 때문에 보다 좋은 성능의 연산 자원들을 갖출 수 있는 장점이 있으며, 이것들의 효율적 활용을 위한 병렬처리 필요성은 점차 커질 것으로 예상된다. 따라서 본 논문에서는 스마트 TV 플랫폼 중 하나인 webOS TV 에서의 병렬처리 가능성 및 방안을 파악하기 위해 webOS 를 프로파일링하고, 그

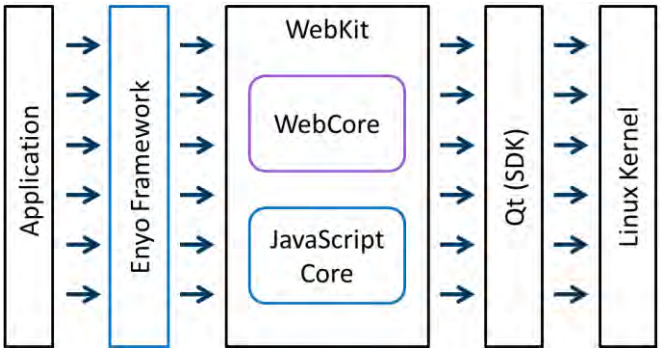

그림 1 webOS architecture

결과 분석을 통한 가속화 전략을 제안하고자 한다.

2. webOS Architecture

webOS architecture [4]를 단순화 하면 그림 1과 같다. webOS 에서 실행되는 web applications (WebApp)은 기본적으로 JavaScript 언어로 구성되며, HTML5, CSS 를 활용할 수 있다. 이러한 WebApp 은 모듈화, 캡슐화가 강조된 Enyo Framework [5]을 사용하여 User Interface 를 꾸밀 수 있다. WebApp 은 WebKit 을 통해 해석되고 실행된다. WebKit 은 크게 WebCore 와 JavaScript Core 로 나눌 수 있는데, JavaScript Core 는 JavaScript 언어를 해석하고 실행하는 역할을 담당하고, WebCore 는 HTML5, CSS 등의 언어를 담당하여 처리한다. webOS 내의 WebKit 은 Qt port 를 사용하므로 QtWebKit 이라 부를 수 있으며, 실제 Qt 가 제공하는 QtGUI Library 등 여러 Library 내의 함수 호출을 통해 렌더링 작업이 이루어진다. 여기서 Qt [6]는 cross-platform application 으로 QtGUI 등 다양한 API 및 Library 를 제공하는 프로그램으로써 webOS 내에서 middle layer 역할을 하며, 이것은 최종적으로 Linux Kernel 을 통해 하드웨어 자원을 사용하게 된다.

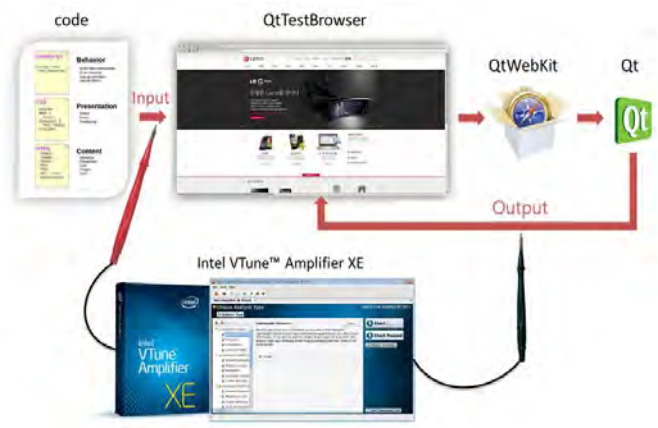

그림 2 프로파일링 방법

3. Profiling & Analysis

본 연구에서 webOS 내 가속화 가능한 layer 및 함수를 파악하기 위해 그림 2와 같은 방법으로 프로파일링을 진행하였다. QtWebKit 을 사용하는 브라우저 위에서 WebApps 을 수행하고 이 과정을 Intel VTune™ Amplifier 를 사용하여 프로파일링을 진행하였다. 이러한 방식으로 방대한 전체 소스코드 분석 없이, 각 layer 별 함수간 계층구조 및 연산자원을 오래 점유하는 함수를 쉽게 파악할 수 있다.

3.1 Profiling 환경

프로파일링을 진행한 환경은 다음과 같다.

- CPU : Intel i7-3770 (Quad-core, 3.40 GHz)
- Memory: DDR3 8GB PC3-12800
- OS : ubuntu 12.04 (for profiling only)
- Qt: Qt5-tools 5.0.1
- WebKit: WebKit Nightly Builds r160349

3.2 WebCore 분석

WebCore 를 사용하는 렌더링 작업이 주를 이루는 WebApps 을 선정하고 이에 대한 프로파일링을 진행하였다. 전체 진행 흐름은 그림 3과 같다. WebCore 의 많은 함수를 거쳐 최종적으로 QtGUI library 의 Qpainter class 를 사용하며, 여기서 많은 시간을 소모한다. Webkit 내 WebCore 는 QtGUI 를 호출하기 위한 중간단계로 모든 코드는 순차적으로 진행된다. 렌더링은 Qpainter class 내 QRasterPaintEngine 을 통해 이루어지며, 이것은 webOS 를 포함한 embedded linux 환경에서 하드웨어 가속화를 지원한다 [6]. 분석 결과 렌더링 자체에 상대적으로 많은 시간이 소모될 뿐 compute-intensive 한 곳이 아니기 때문에 멀티코어를 활용한 병렬처리는 수행시간 단축에 큰 효과가 없었다 (그림 4).

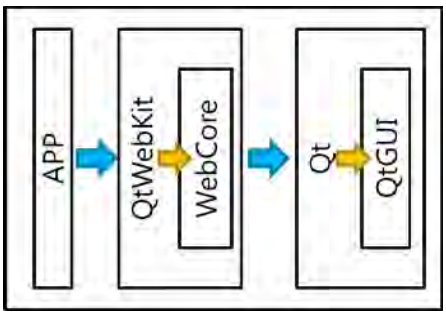

그림 3 WebCore 관련 응용프로그램들의 처리과정

3.3 JavaScriptCore 분석

JavaScriptCore (JSC)는 JavaScript Code 를 해석하고 수행하는 역할을 하며, 별도의 QtSDK 를 활용하지 않는다. JSC 분석을 위해 compute-intensive 한 application 인 N-body simulation 을 프로파일링에 활용하였다 (그림 5). JavaScript 코드 자체에 loop 가 많고, JSC 는 그 코드를 해석하고 수행하는 역할만 하기 때문에, 코드 단에서 병렬처리언어를 사용해야만 가속화가 가능하다.

4. 결론 및 토의

WebKit 및 JavaScript 는 기본적으로 순차 실행 모델이기 때문에 내부적 병렬처리에는 한계를 갖는다. 이를 극복하기 위해, Intel 은 JavaScript 언어를 병렬처리 할 수 있는 River Trail 이라는 JavaScript Engine 을 출시하였으며, Khronous Group 에서는 OpenCL 언어와 결합시킬 수 있는 WebCL 을 출시하였다. 또한 HTML5 는 'web worker'라는 병렬처리 기능을 도입, 본 thread 외

에 복잡한 계산은 background threads 에서 처리할 수 있도록 하였다. 이것들의 병렬 처리 효과에 대한 연구가 있지만 [7], [8], [9], River Trail 은 그 자체가 JavaScript 엔진이기 때문에, 이를 통한 병렬처리를 위해서는 기존 시스템의 엔진을 이것으로 교체 해야 하며, WebCL 은 특수한 환경에서 사용 가능한 몇몇 prototype 만 존재하고 HTML5 표준이 아니기 때문에 그 활용도가 떨어진다. 반면에 'web worker'는 HTML5 의 표준 기술로써 다양한 환경에서 사용 가능하지만, compute-intensive 하거나 데이터 병렬성이 풍부한 작업의 병렬처리가 목적이 아닌, UI 방해 없이 복잡한 연산을 background 에서 수행하는 것을 목적으로 하기 때문에 연산 자원의 효율적인 활용에는 한계가 있다. 따라서 web 환경에서 연산 자원을 충분히 활용하기 위해서는 표준화 된 병렬처리기술 도입이 시급하다.

본 연구는 webOS 시스템 내의 가속화 가능성 있는 부분을 파악하고자 진행되었다. 시스템 내부적으로는 순차코드가 많고, 병렬성이 풍부하지 않아 병렬처리 효과가 없었다. 따라서 계산자원을 충분히 효율적으로 활용하기 위해선 응용프로그램 별로 코드 단에서 병렬처리언어를 삽입하여야 한다.

5. 사 사

본 연구는 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신 · 방송 연구개발사업[14-824-09-014, 인간 수준의 평생 기계학습 SW 기초 연구 (기계학습연구센터)]과 2014 년도 두뇌한국 21 플러스사업 및 LG 전자의 지원을 받아 수행하였음

참고문헌

[1] 방송통신진흥본부 미디어산업진흥부, "스마트 TV 의 최근 동향과 주요 이슈", 동향과 전망:방송 · 통신 · 전파, 통관 제 72 호, 2014.

[2] 홍진우, "Beyond 스마트 TV 기술 소개", 2011 스마트 TV 기술 및 개발자 워크샵, ETRI.

[3] M. Yusufov and I. Kornilov, "Roles of smart tv in iot-environments: a survey," in Proceedings of the 13th Conference of Open Innovations Association FRUCT and Seminar on e-Tourism. Pertozavodsk, Russia, April 22-26, 2013, pp. 163–168

[4] webOS, http://openwebosproject.org/

[5] Enyo Framework, http://enyojs.com

[6] Qt Project, http://qt-project.org

[7] S. Herhut, R. L. Hudson, T. Shpeisman, and J. Sreeram, "River trail: A path to parallelism in javascript," in Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications. ACM, 2013, pp. 729–744.

[8] M. Cho, S. W. Kim, and Y. Han, "Web-based image processing using javascript and webcl," in Consumer Electronics (ICCE), 2014 IEEE International Conference on. IEEE, 2014, pp. 337–338.

[9] S. Okamoto and M. Kohana, "Load distribution by using web workers for a real-time web application," International Journal of Web Information Systems, vol. 7, no. 4, pp. 381–395, 2011.

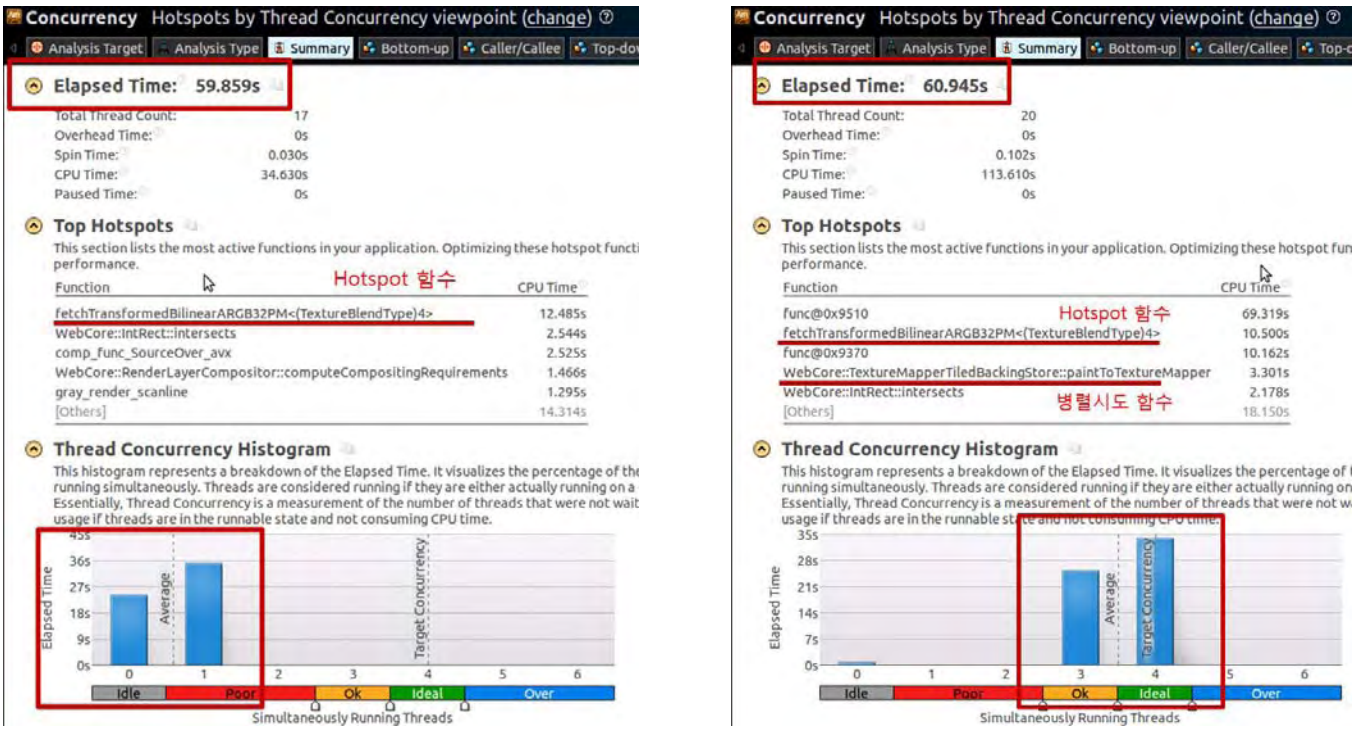

그림 4 WebCore 관련 응용프로그램의 병렬처리 전(좌)과 후(우) 프로파일링 결과 비교 – 작업처리시간에 비해 threads 생성 오버헤드가 더 크기 때문에 병렬처리에 큰 효과가 나타나지 않음을 알 수 있다.

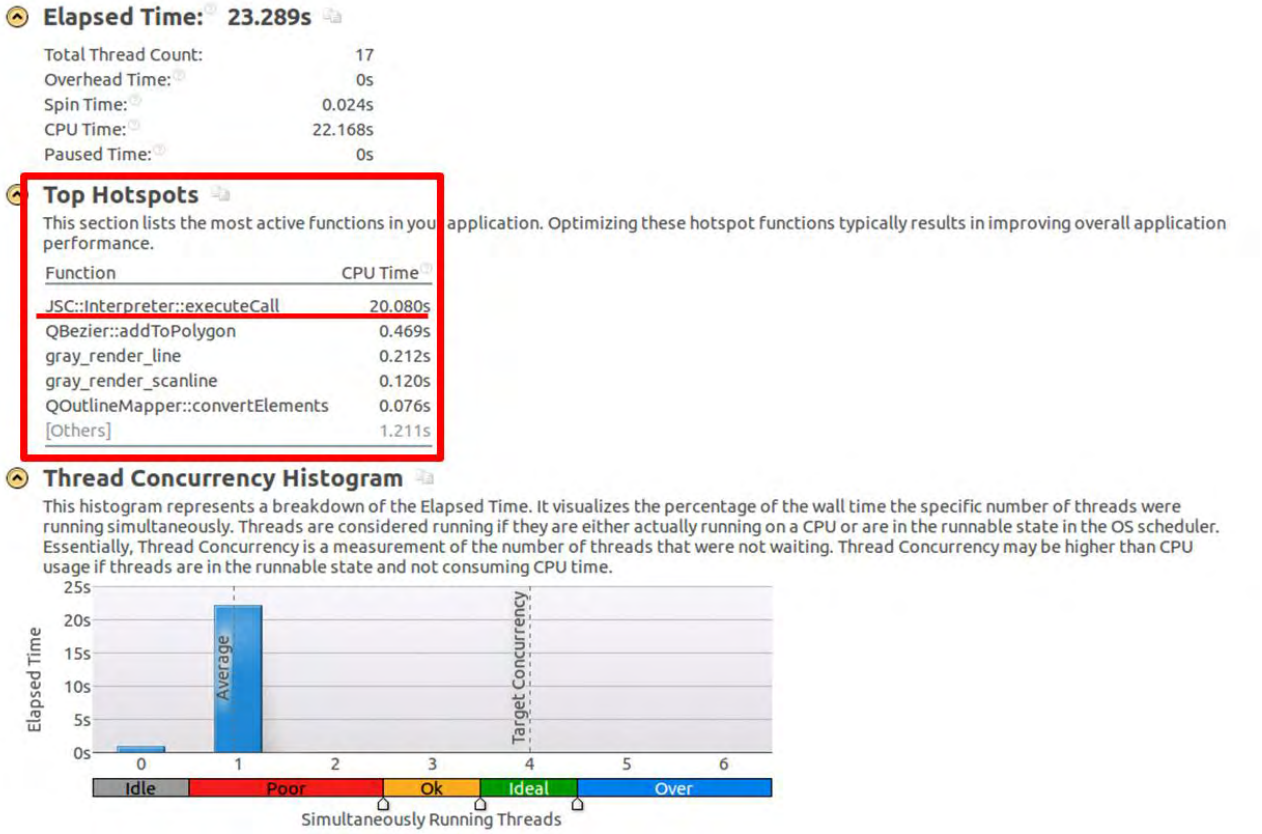

그림 5 JavaScriptCore (JSC)를 사용하는 응용프로그램의 프로파일링 결과 예시 - JSC는 코드 해석 및 실행을 하는 곳이기 때문에 실제 수행시간을 단축하기 위해서는 코드 단에서의 병렬처리 작업이 요구된다.