멀티 코어와 GPU가 차세대 웹 브라우저의 성능에 미치는 영향 분석

홍경환○, 김대호, 신동군
성균관대학교 컴퓨터공학과
RedCarrottt@gmail.com, rlaeoghi2@skku.edu, dongkun@skku.edu

Performance Analysis on Next-Generation Web Browser at Multicore CPU and GPU

Gyeonghwan Hong○, Daeho Kim, Dongkun Shin
Department of Computer Engineering, Sungkyunkwan University

요 약

차세대 웹 브라우저는 멀티 쓰레드(multi-thread) 구조로 되어 있으며 HTML5와 WebGL을 기반으로 화려한 그래픽을 구사하기 때문에, 멀티 코어(multi-core) CPU와 GPU의 성능이 웹 브라우저의 성능에 큰 영향을 미치고 있다. 본 논문은 오픈 소스 웹 브라우저인 크로미엄(Chromium) 상에서 프로세서의 성능 변화에 따라 웹 브라우저에서 실행되는 웹 어플리케이션의 성능이 어떤 양상으로 변화하는지와 이 변화에 웹 브라우저의 각 동작이 얼마나 기여하는지를 비교·분석하였다. 그 결과 CPU 코어의 수가 렌더링 성능에 큰 영향을 주며, GPU의 성능은 WebGL의 성능을 크게 좌우함을 알 수 있었다.

1. 서 론

최근 스마트 폰의 확산은 iOS, 안드로이드 등 다양한 모바일 플랫폼을 등장시켰고, 컴퓨터 시스템 연구에 새로운 변화를 유도하고 있다. 최근 모바일에는 크게 두 가지 새로운 변화가 나타나고 있다. 첫번째 변화는 PC 이상의 컴퓨터에서만 사용되던 고급 사양의 하드웨어 기술이 스마트 폰에서도 사용되기 시작하고 있는 것이다. 최근 스마트 폰은 멀티 코어(multi-core)를 탑재하여 출시되고 있으며, PC에서만 쓰이던 그래픽처리장치(GPU)도 탑재하고 있다. 예를 들어, 모바일 AP인 Exynos 4412의 경우에 4개의 Cortex-A9 코어를 탑재하고 있으며 ARM Mali T-604 GPU를 탑재하고 있다.

두번째 변화는 웹 브라우저의 확산과 웹 어플리케이션(web application; 이하 웹 앱)의 등장이다. 특히, HTML5 표준 제정 작업이 진척되면서 클라우드 환경을 위한 웹 앱의 활용 폭이 더욱 커지고 있으며, 웹 브라우저의 성능이 전체 모바일 시스템의 성능을 대변하는 추세가 되고 있다. 스마트 폰에 멀티 코어가 등장했을 때, 스마트 폰에서 멀티 코어를 잘 활용할 수 있는 킬러 어플리케이션이 무엇인지에 대해 많은 논쟁이 있었고, 그 해답으로 멀티 쓰레드(multi-thread)로 구현된 웹 브라우저가 제시되어 왔다.

본 연구는 지식경제부 및 정보통신산업진흥원의 지원사업의 연구결과로 수행되었음

예를 들어, 구글 크롬을 비롯한 현대 웹 브라우저는 멀티 쓰레드(multi-thread) 구조로 되어 있으며 WebGL을 고속으로 처리하기 위해서 GPU를 활용하고 있다[4]. 그러므로 현대 웹 브라우저는 CPU 코어의 수가 늘어남에 따라 체감 성능이 비약적으로 향상될 수 있으며, WebGL을 기반으로 화려한 그래픽의 웹 앱들이 늘어남에 따라 GPU가 웹 브라우저 성능에 기여하는 바가 커지고 있다.

본 연구에서는 이러한 스마트 폰 하드웨어의 발전이 스마트 폰의 대표적 어플리케이션인 웹 브라우저의 성능에 어떤 영향을 미치는지 조사하였다. 이를 위해 오픈 소스 웹 브라우저인 크로미엄(Chromium)을 사용하여 CPU 코어의 수와 GPU의 성능을 바꾸어가며 크로미엄 상에서 HTML5와 WebGL을 기반으로 하는 벤치마킹(benchmarking) 웹 앱의 실행 결과를 바탕으로 성능 변화를 비교·분석하였다. 자세한 성능 변화의 분석을 위해 크로미엄에 내장된 프로파일링(profiling) 도구를 이용하여 성능 변화에 웹 브라우저의 각 동작이 얼마나 기여하는지를 관찰하였다.

기존 연구로 모바일 웹 브라우저 캐시(cache)의 크기[1], 자원 불러오기(resource loading)[2], 클라이언트-서버 모델의 불균형이 웹 브라우저 성능에 미치는 영향[3]에 대해 연구한 바가 있다. 그러나 이 연구들에서는 하드웨어 성능 변화에 의한 웹 브라우저의 성능 변화를 고려하지는 않고 있다.

2. 크로미엄과 프로파일링

2.1. 크로미엄

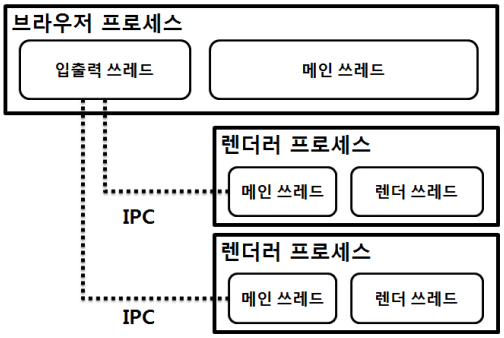

그림 1. 크로미엄의 멀티 쓰레드 구조[4]

크로미엄은 오픈 소스 웹 브라우저 프로젝트다. 크로미엄은 시스템 안정을 도모하기 위해 여러 개의 프로세스로 시스템을 구성하는 멀티 프로세스(multi-process) 구조와, 프로세스 간 자원(resource)을 격리하는 샌드박싱 (Sandboxing) 기법을 사용한다. 이 때문에 크로미엄은 실행 도중 오류가 발생하더라도 브라우저 전체를 종료하는 것이 아니라 프로세스 하나만 종료하면 오류를 해결할 수 있다[4].

그림 1은 크로미엄의 멀티 프로세스 및 멀티 쓰레드 구조를 나타낸다. 프로세스들은 크게 브라우저 프로세스(browser process), 렌더러 프로세스(renderer process)로 나뉜다. 브라우저 프로세스는 윈도 당 하나 존재하며, 여러 개의 렌더러 프로세스들을 관리하고 외부 네트워크와 통신한다. 렌더러 프로세스는 여러 개의 탭 당 하나 존재하며, 자기가 관리하는 탭에 웹 페이지 내용을 그린다. 프로세스들은 서로 IPC(Inter-process Communication)를 통해 통신한다[4].

각 프로세서에는 IPC만을 담당하는 쓰레드(그림 1에서 브라우저 프로세스의 입출력 쓰레드, 렌더러 프로세스의 메인 쓰레드)가 존재하여 프로세스 고유의 기능과 IPC를 병렬적으로 처리할 수 있도록 하였다[4].

2.2. 크로미엄 프로파일링

크로미엄에는 성능을 측정하기 위한 프로파일링 도구로 크롬 개발자 도구(Chrome Developer Tools), 쓰레드 및 태스크 프로파일러 (Thread and Task Profiler), 구글 퍼프-툴즈(Google Perf-tools) 등이 내장되어 있다[5]. 각 도구마다 측정할 수 있는 범위가 다르다.

크롬 개발자 도구는 웹키트 프로젝트의 웹키트 검사기(Webkit Inspector)를 기반으로 만들어진 웹 브라우저 분석 도구다. 이 도구는 웹 앱을 실행했을 때 크로미엄 브라우저에서 발생하는 이벤트를 추적하고 그 사용 시간을 이벤트 종류에 따라 나누어 분석하는 기능을 제공한다[6].

쓰레드 및 태스크 프로파일러는 크로미엄 내부에서 호출된 함수의 호출 횟수와 실행 시간을 프로세스 별, 쓰레드 별로 분류하여 수집하는 프로파일링 도구다[5].

구글 퍼프-툴즈는 리눅스 커널에 기본적으로 포함된 프로파일링 도구인 퍼프(Perf)를 기반으로 만들어진 CPU 프로파일러다. 이 도구는 C++ 언어로 된 모든 프로그램에서 명령어 사용 정보 메모리 영역 사용 정보 등을 수집한다. 이 도구는 활용 범위가 넓지만 설정해야 할 옵션이 너무 많다[5].

실험	실험 1	실험 2
CPU	인텔 제온(Xeon) E5620 2.40GHz	
	코어 1~8개	코어 8개
그래픽	엔비디아 지포스 GTS 450	엔비디아 지포스 GTS 450
		엔비디아 지포스 6200
운영체제	우분투 12.04 32비트 (커널 3.2.0-23)	
웹 브라우저	크로미엄 18.0.1025.151	

표 1. 실험 환경

3. 실 험

프로세서의 성능에 의한 웹 브라우저 성능 변화를 알아보기 위해 표 1과 같이 실험 환경을 갖추었다. 여기서 네 가지의 벤치마킹 웹 앱을 실행하여 그 결과를 얻어내었다.

임팩트(Impact), 피스키퍼(Peacekeeper), 애스터로이드(Asteroid) 벤치마크는 점수를 제공하고, 텍스쳐드 메시(Textured Mesh) 벤치마크는 초당 그리기 동작 횟수를 제공한다. 텍스쳐드 메시는 WebGL의 성능을 측정하는 벤치마크다. 첫 번째 실험에서는 CPU 코어를 1개부터 8개까지 변경시켜가며 성능을 측정하였다. 현재 AP가 4개의 코어를 지원하지만 향후 8-코어의 등장을 고려하여 실제 스마트 폰이 아닌 8개의 코어가 탑재된 인텔 프로세서에서 실험을 진행하였다. 두 번째 실험에서는 표 1에 제시된 그래픽 카드를 바꾸어 장착하며 성능을 측정하였다.

또한 그림 2와 같이 크로미엄에서 각 벤치마크가 실행될 때 어떤 동작을 얼마나 수행하는지를, 크로미엄에 내장된 프로파일링 도구인 크롬 개발자 도구를 사용하여 알아내었다.

웹 브라우저의 동작은 크게 로딩, 스크립팅, 렌더링으로 나뉜다. 로딩은 네트워크로부터 데이터를 불러오고 그 중 HTML 파일을 파싱하는 동작이며, 스크립팅은 자바스크립트를 해석하고 그에 대한 명령을 내리는 동작이다. 렌더링은 로딩 과정에서 이미 해석된 HTML 요소들을 그려주는 동작이다.

그림 2에서 임팩트와 애스터로이드는 렌더링 작업이 압도적으로 많기에, HTML 요소들을 그리는 것으로 인한 부하(load)가 많을 것으로 보인다. 피스키퍼는 스크립팅이 더 많이 분포된 것으로 보아 자바스크립트로 인한 부하가 더

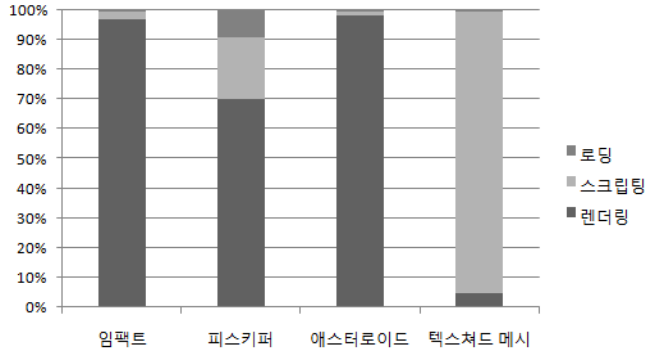

그림 2. 벤치마크의 동작 비율

많이 분포되어 있다고 유추할 수 있다. 텍스쳐드 메시는 스크립팅의 비중이 절대적이므로 WebGL 코드 처리로 인한 자바스크립트 부하가 대다수임을 알 수 있다.

세부적으로는 임팩트와 애스터로이드의 렌더링 동작 중 99%가 그리기였다. 피스키퍼의 렌더링 동작 중 41.33%가 그리기, 56.62%가 스타일 재계산이었으며, 스크립팅 동작 중 59.08%가 함수 호출, 25.02%가 가비지 컬렉션이었다. 텍스쳐드 메시의 스트립팅 동작 중 가비지 컬렉션이 99%였다.

4. 분 석

그림 3은 CPU 코어의 개수가 변화함에 따라 달라지는 벤치마크 점수를 나타내었다. 이 그래프의 x축은 CPU의 코어 개수를 나타낸다. y축은 각 코어에서의 성능을 코어가 1개일 때의 성능을 1로 두었을 때 그에 대한 상대 값으로 나타내었으며, 이것이 높은 점수일수록 웹 브라우저의 HTML 성능 혹은 자바스크립트 성능이 좋다는 것을 의미한다.

임팩트와 피스키퍼, 애스터로이드 모두 코어가 1개일 때와 2개일 때의 성능 차이가 약 1.5배나 벌어져 있다. 코어 3개 이후로는 점진적으로 성능이 증가한다. 그러다가 피스키퍼는 싱글 코어에 비해 2.4배까지 올라가지만 이를 제외하고는 코어가 8개일 때 모두 성능이 갑자기 내려갔다. 스크립팅 연산이 대부분인 텍스쳐드 메시도 비슷한 양상을 보이나, 렌더링 연산이 대부분인 임팩트, 애스터로이드보다 CPU 코어 개수 변화의 영향을 덜 받는다.

그림 4는 서로 다른 GPU에서 달라지는 벤치마크 점수를 나타내었다. 이 그래프의 y축은 지포스 6200일 때의 성능을 1로 두었을 때 그에 대한 상대 값으로 나타낸 것으로, 높은 점수일수록 웹 브라우저의 HTML, 자바스크립트 성능이 좋다.

이 실험에 사용된 GPU는 엔비디아 지포스 6200, 엔비디아 지포스 GTS 450으로 총 두 개다. 전자는 4개 픽셀 셰이더 프로세서를 사용하는 반면, 후자는 192개의 CUDA 프로세서를 사용한다. 성능 면에서는 후자가 더 우세하다.

그림 4에서 임팩트, 피스키퍼, 애스터로이드는 모두 비슷한 점수를 보이는 반면, WebGL이 대부분인 텍스쳐드 메시는 약 49배의 점수 차이를 보인다. 렌더링과는 달리 WebGL 처리 시에는 GPU를 많이 사용하므로 지포스 6200이 지포스 GTS 450보다 더 떨어지는 점수를 받은 것으로 보인다.

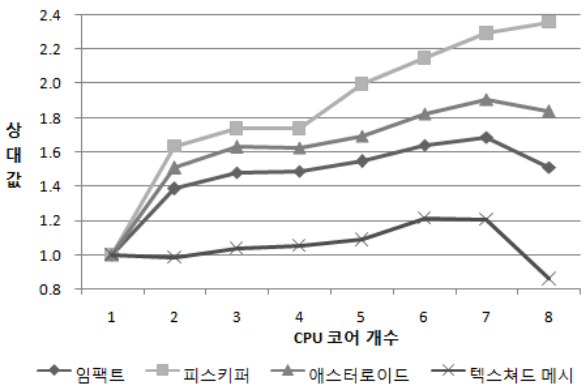

그림 3. CPU 코어 개수 변화에 따른 벤치마크 점수

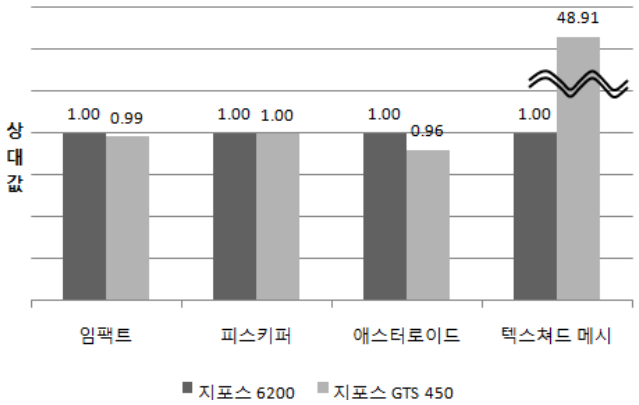

그림 4. GPU 성능 변화에 따른 벤치마크 점수

실험 결과를 미루어 보았을 때, CPU 코어의 개수에 비례하여 렌더링 연산이 빨라진다는 것을 알 수 있다. 또한 CPU 코어 개수가 WebGL 성능을 거의 좌우하지 못하며, GPU 성능이 WebGL 성능을 좌우함을 알 수 있었다.

5. 결론 및 향후 계획

논문에서 CPU 코어의 개수가 웹 브라우저의 렌더링 성능 향상에 큰 영향을 미친다는 점을 알아냈다. 웹 브라우저가 PC와 모바일 장치에서 모두 핵심 어플리케이션 프로그램이 되어가는 시점에서 멀티 코어 성능 향상과 GPU 성능 향상은 매우 중요해지고 있다.

또한 GPU는 렌더링 성능에는 별다른 영향을 미치지 못하며, WebGL 성능에 큰 영향을 미친다는 점을 알아냈다. 화려한 그래픽을 연출하는 웹 어플리케이션을 만들 때 WebGL 처리에 대한 최적화가 중요할 것으로 보인다.

벤치마크와 프로파일링 도구를 통해 성능 관계를 찾아내는 것은 웹 브라우저 성능 향상의 초석이다. 앞으로 향상된 벤치마크와 프로파일링 도구를 개발하여 다른 하드웨어의 성능과 웹 브라우저 성능의 관계를 명확히 밝히고자 한다.

참고 문헌

[1] Zhen Wang et. al., “How Effective is Mobile Browser Cache?”, Proceedings of the 3rd ACM workshop on Wireless of the students, by the students, for the students, S3, p.17-19, 2011.

[2] Zhen Wang et. al., “Why are Web Browsers Slow on Smartphones?”, Hot MobileWorkshop, p.91-96, 2011.

[3] Zhen Wang et. al., “How Far Can Client-Only Solutions Go for Mobile Browser Speed?”, WWW, p.1-10, 2011.

[4] The Chromium Projects, “Multi-process Architecture”, http://www.chromium.org/developers/design-documents/multi-process-architecture, 2012.

[5] The Chromium Projects, “Profiling Chromium and WebKit”, https://sites.google.com/a/chromium.org/dev/developers/profiling-chromium-and-webkit, 2012.

[6] Google, “Chrome Developer Tools”, https://developers.google.com/chrome-developer-tools/docs/timeline, 2012.