2013년 5월 전자공학회 논문지 제 50 권 제 5 호
Journal of The Institute of Electronics Engineers of Korea Vol. 50, NO. 5, May 2013 http://dx.doi.org/10.5573/ieek.2013.50.5.150

논문 2012-50-5-18

Cyclostorm : WAC 기반 모바일 앱의 자바스크립트 처리 효율 향상을 위한 클라우드 컴퓨팅 서비스

(Cyclostorm : The Cloud Computing Service for Uplifting Javascript Processing Efficiency of Mobile Applications based on WAC)

방 지 웅*, 김 대 원**

(Jiwoong Bang and Daewon Kim©)

요 약

WAC (Wholesale Application Community) 기반 모바일 웹 애플리케이션의 보급이 널리 확산됨에 따라 자바스크립트와 HTML (Hyper Text Markup Language)로 구현된 애플리케이션의 처리 성능이 점차 이슈화되고 있다. 단순한 처리 기능만을 갖는 구조라면 현재 통용되는 브라우저에서도 문제가 없지만 자바스크립트의 처리량이 점점 증가 할수록 브라우저의 처리 부담 또한 가중된다. 현재 상용화 되어 있는 모바일 브라우저에서는 자바스크립트의 처리에 시간 및 용량의 제한을 두고 있다. 이러한 문제점의 해결을 위한 대안으로서 HTML 5에서는 기존의 자바스크립트에서는 지원하지 않는 멀티 스레드 구현을 위한 Web Worker를 제공하고 있다. Web Worker는 싱글 스레드에서 처리하는 일정 부분을 별도의 스레드를 통해서 처리하는 메커니즘을 제공한다. 하지만 이는 모바일상의 Native 애플리케이션만큼의 처리 능력을 보장하는 것이 아니며 근본적인 처리 속도 개선 방법으로는 미흡하다. Cyclostorm은 이러한 모바일 상에서의 자바스크립트 처리를 클라우드 상의 컴퓨터 서버에 이전함으로써 모바일 클라이언트로서의 자원적 한계를 극복하고 고성능 처리 서비스를 제공하여 Native 애플리케이션 만큼의 성능을 보장한다. 성능 평가 실험에서 Cyclostorm은 기존의 모바일 브라우저에서의 자바스크립트 처리보다 최대 6배 빠른 처리 속도를 보이고 있고 HTML 5의 Web Worker보다는 3~6배정도의 빠른 성능을 보이고 있다. 또한 메모리 부분에서도 서버 측에 존재하는 메모리를 사용하기 때문에 전반적으로 기존의 방법보다는 낮은 사용량이 측정 되었다. 본 논문에서는 현재 모바일 산업에서 화두가 되고 있는 WAC 기반에서 모바일 브라우저가 갖고 있는 한계를 극복하고 기존 웹 애플리케이션의 성능을 획기적으로 개선할 수 있는 클라우드 컴퓨팅 서비스인 Cyclostorm을 소개한다.

Abstract

Currently it is being gradually focused on the mobile application's processing performance implemented by Javascript and HTML (Hyper Text Markup Language) due to the dissemination of mobile web application supply based on the WAC (Wholesale Application Community). If the application software has a simple functional processing structure, then the problem is benign, however, the load of a browser is getting heavier as the amount of Javascript processing is being increased. There is a limitation on the processing time and capacity of the Javascript in the ordinary mobile browsers which are on the market now. In order to solve those problems, the Web Worker that is not supported from the existing Javascript technology is now provided by the HTML 5 to implement the multi thread. The Web Worker provides a mechanism that process a part from the single thread through a separate one. However, it can not guarantee the computing ability as a native application on the mobile and is not enough as a solution for improving the fundamental processing speed. The Cyclostorm overcomes the limitation of resources as a mobile client and guarantees the performance as a native application by providing high computing service and ascripting the Javascript process on the mobile to the computer server on the cloud. From the performance evaluation experiment, the Cyclostorm shows a maximally 6 times faster computing speed than in the existing mobile browser's Javascript and 3 to 6 times faster than in Web Worker of the HTML 5. In addition, the usage of memory is measured less than the existing method since the server's memory has been used. In this paper, the Cyclostorm is introduced as one of the mobile cloud computing services to conquer the limitation of the WAC based mobile browsers and to improve the existing web application's performances.

Keywords : WAC, Mobile, Cloud Computing, Javascript, HTML 5

* 학생회원, 단국대학교 컴퓨터학과 (Department of Computer Science, Graduate School of Dankook University)
** 정회원, 단국대학교 멀티미디어공학과
(Department of Multimedia Engineering, College of Engineering, Dankook University)
© Corresponding Author(E-mail:drdwkim@dku.edu)
접수일자: 2013년2월7일, 수정완료일: 2013년4월25일

Ⅰ. 서 론

전 세계적으로 확산되고 있는 스마트폰 열풍으로 인해 모바일 산업 전반에 걸쳐 다양한 서비스가 등장하고 있다. 새로운 시장으로 화두가 되고 있는 모바일 앱 시장은 기존 통신사에 의해 폐쇄적으로 운영되었던 형태에서 일반 앱 및 웹 개발자도 판매 및 수익 창출이 가능한 오픈 형태로 진화하였다. 이러한 변화는 모바일 앱 개발 및 시장 규모를 급속도로 확장 시키는 계기를 마련하였고 많은 사람들이 다양한 모바일 앱을 통한 정보와 서비스를 사용할 수 있게 하였다. 현재 스마트폰 앱 시장은 애플의 앱스토어와 구글의 안드로이드 마켓 진영으로 양분되어 있다. 향후 성장할 시장을 놓고 2010년 2월 스페인 바로셀로나에서 개최된 MWC (Mobile World Congress)에서 전 세계 24개 통신사업자와 3개의 단말기 제조사가 연합하여 도매 앱 커뮤니티인 WAC를 제안하였다[1]. 여기에 참여하고 있는 통신사의 휴대폰 가입자 수가 전 세계 휴대폰 가입자 수의 2/3에 해당하는 30억 명의 고객을 대상으로 하기 때문에 스마트폰 앱 시장에서 강력한 진영으로 성장할 것으로 예측하고 있다. WAC에서는 HTML 5의 강력한 사용자 인터페이스 기능과 모바일 자원을 사용하도록 하여 Native 앱과 같은 수준의 웹 앱을 개발할 수 있도록 다양한 라이브러리 및 표준 문서를 제공하고 있다[2]. WAC 기반 앱은 모바일 내부 자원에 접근하기 위해서 자바스크립트를 사용한다[3][4]. 이는 웹 서비스 클라이언트 환경을 구성하는데 매우 중요한 역할을 수행하는 언어이며 인터프리터 방식으로 처리되기 때문에 일시에 많은 연산이 요구되면 브라우저가 그 처리를 위해 동작할 때 멈추는 현상이 발생하는 문제점이 있다. 또한 Native 앱에 비해 복잡한 처리나 연산량이 많은 알고리즘은 실행할 때 자바스크립트 엔진 처리 속도가 상대적으로 느려 이를 개선하기 위한 방법이 연구되고 있다. 첫째, 자바스크립트 엔진 실행 속도 향상을 위해 JIT (Just In Time) 컴파일러를 사용하는 방법이 있다. 하지만 이 방법은 한정된 모바일 자원으로 인해 응답 속도가 느리고 메모리 사용량이 증가되는 문제점이 있다 [5]. 또한 모바일 환경에서는 적합하지 않기 때문에 이를 해결하기 위해서 컴파일 된 코드의 재사용 기법을 통해서 성능을 끌어올리는 방법이 연구되고 있다[6]. 둘째, 자바스크립트의 멀티 스레드 프로그래밍을 지원하는 방법이다. 멀티 스레드를 이용하여 병렬 처리함으로써 실행 속도 및 화면이 멈추는 현상을 피하게 하는 방법이다. Firefox 3.5의 DOM (Document Object Model) Workers는 멀티 스레드 기능을 제공하나 DOM에 직접 액세스 할 수 없고 네임스페이스 역시 공유하지 않는 문제점을 갖고 있다. 이를 해결하기 위해서 HTML 5에서는 Web Worker 기능을 제공한다[7]. Web Worker를 사용할 경우 처리시간이 많이 요구되는 부분은 따로 구분하여 백그라운드로 처리할 수 있고 되어 화면이 멈추는 현상 없이 빠르게 처리할 수 있다. 또한 실행 중에 메시지를 이용한 DOM에의 접근 및 네임스페이스 사용이 가능하다. 하지만 단점으로는 모바일의 제한적 환경으로 인해 강력한 Web Worker 기능에도 불구하고 Native 앱보다 많은 처리 시간을 갖는다. 본 논문에서는 제한적인 모바일 단말 환경에서 WAC 기반 앱의 동작 성능 향상을 위해 Cyclostorm 클라우드 프레임워크를 소개한다. 제안된 Cyclostorm은 기존의 클라이언트에서 사용되는 자바스크립트를 상대적으로 하드웨어 사양이 높은 서버에서 대신 수행하도록 요청하고 그 결과를 응답으로 받아 처리하는 구조를 갖고 있다. 성능 평가에서 Cyclostorm은 로컬에서 단독으로 실행되는 경우보다 약 3~6배 정도 빠른 성능 향상도를 보였고 메모리 사용량 측면에서는 수행 코드 부분을 서버에 이양함으로써 사용량 또한 줄어드는 효과가 발생하였다. Ⅱ장에서는 자바스크립트 성능 향상을 위한 기존 방법들을 소개하고 Ⅲ장에서는 본 연구에서 제안하는 Cyclostorm의 구조 및 기술적 수행 절차에 대해 설명한다. Ⅳ장에서는 기존의 방법들과 비교하여 성능 평가 결과를 제시하고 Ⅴ장에서는 결론 및 향후 연구 내용에 대해서 논한다.

Ⅱ. 관련 연구

1. WAC (Wholesale Applications Community)

WAC는 2010년 2월 스페인 바로셀로나에서 개최된 MWC에 모인 세계 각국의 이동통신 24개사가 새로운 앱 환경을 구축하고자 합의한 도매 애플리케이션 커뮤니티이다. 스마트 폰 앱 시장의 한 주체인 애플을 제외한 휴대폰 제조사들도 다수 WAC 진영에 참여함으로써 글로버 수퍼 앱스토어라고 불리운다. 그림 1은 WAC의 유통 구조를 보이고 있다.

그림 1과 같이 WAC는 도매 유통 구조를 갖는다. 개발자들은 앱을 WAC에 등록하고 여기에 참여한 통신사가 등록된 다양한 애플리케이션 중에 원하는 것을 선택해 각사가 운영하는 앱스토어를 통해 판매한다. 사용자

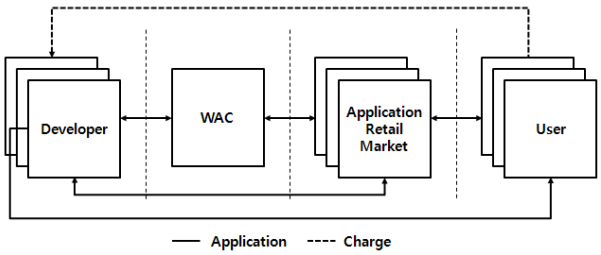


그림 1. WAC 유통 구조
Fig. 1. WAC Circulation Structure.

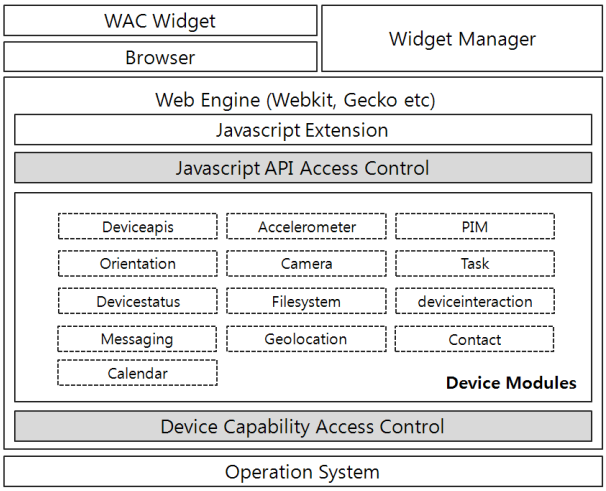


그림 2. WAC 프레임워크 구조
Fig. 2. WAC Framework Structure.

는 소매 방식으로 판매되는 앱을 구입하고 그 비용은 개발자에게 전달된다. WAC는 장기적으로 휴대폰 이외에도 스마트 TV, IPTV (Internet Protocol Television) 등 다양한 영역의 멀티미디어 플랫폼으로의 확장을 계획 중이며 특정 운영체제 기반이 아닌 HTML 5로 표준화된 개발 환경을 제공하고 있다[8].

그림 2에 보이는 WAC Widget은 웹 애플리케이션 또는 웹 위젯이다. 위젯은 wgt 패키지 형식의 압축 파일로 되어있으며, 여기에는 HTML, SVG (Scalable Vector Graphics), JavaScript, CSS (Cascading Style Sheets)와 같은 웹 문서와 Configuration Document, Digital Signature 등의 정보가 들어있다. 이러한 WAC 위젯은 Widget Manager를 통해 설치되고 Web Engine 위에서 실행된다. Widget Manager는 설치, 삭제, 업데이트와 같은 전반적인 관리를 담당하며 Widget Engine은 기본적으로 JavaScript 엔진을 포함하기 때문에 위젯 리소스의 JavaScript와 CSS 및 HTML, SVG와 같은 마크업 문서를 렌더링 하는 기능을 제공한다. Web Engine의 종류로는 Webkit, Gecko 등이 있으며 JavaScript Extension에는 JavaScript API (Application Programming Interface) Access Control, Native 애플리케이션을 지원하기 위한 WAC Device 모듈 등이 포함되어 있다. 각 모듈은 모바일에서 제공되는 자원을 손쉽게 활용할 수 있도록 자바스크립트 API 형태로 제공된다[9~11].

2. JIT (Just In Time) 컴파일러

자바스크립트는 해석기에 의해 실행되는 것을 전제로 설계된 스크립트 언어의 일종으로, 검색어 자동 완성 기능이나 실시간 검색어 순위 표시와 같이, 웹 브라우저에서 사용자의 입력 내용이나 시간에 따라 반응하는 웹페이지의 동적인 요소를 구현하는 데 많이 사용되고 있다. 최근 많은 웹페이지가 다양한 서비스를 제공하기 위하여 다량의 연산을 요구하는 자바스크립트 프로그램을 이용하여 구현되고 있으나 엔진에서 코드가 추상 구문 트리나 중간 코드로 변환되어 해석기에 의해 수행되는 경우 그 속도가 느려 사용자 응답 시간이 눈에 띄게 증가하는 문제점이 있다. 이를 개선하기 위하여 일부 자바스크립트 엔진은 수행 중에 JIT 컴파일러를 이용하여 자바스크립트 코드나 그 중간 코드를 번역한 기계어를 직접 실행한다. 자바스크립트 엔진 중에 JIT 컴파일러를 사용하는 엔진으로는 모질라사의 오픈소스 브라우저인 Firefox의 TraceMonkey와 구글 크롬의 V8엔진이 있다[12~13]. TraceMonkey는 Firefox에 내장된 자바스크립트 엔진인 SpiderMonkey의 자바스크립트 코드를 중간 코드로 번역하여 해석기를 통해 수행한다. 이러한 한계 성능의 대안으로서 해석기에서 수행되는 일련의 중간 코드를 추적하여 반복문을 감지한 후 이를 JIT 컴파일러에 의해 기계어로 번역하여 수행함으로써 성능을 향상시킨 자바스크립트 엔진이 바로

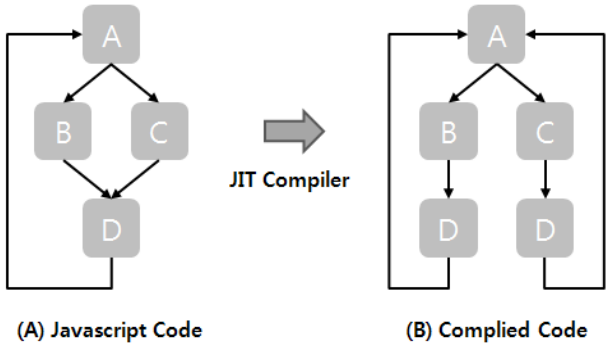


그림 3. TraceMonkey에서 JIT compiler를 이용한 기계어 코드 생성 과정
Fig. 3. Mechanical code generation process using the JIT compiler in TraceMonkey.

TraceMonkey이다[14]. 그림 3은 TraceMonkey에서 JIT compiler로 기계어 코드를 생성하는 과정을 나타낸다.

그림 3에서 A, B, C, D는 자바스크립트 함수 혹은 코드를 의미한다. TraceMonkey의 JIT compiler는 A의 자바스크립트 코드 내에 반복되는 분기문이 있을 경우 인터프리터를 통해 수행 시 이를 빈번히 실행되는 경로로 판단한다. 이를 Hot-Path라고 부르는데 초기 코드를 생성할 때 얻은 경로에 따라 반복되는 부분이 시작되는 곳 부터 기존 코드를 (B)와 같이 확장한다. 프로그램 수행의 흐름에 따라 코드가 확장되기 때문에 그림 3의 D는 중복해서 코드가 생성된다. 즉, 프로그램에서 수행 시간의 비중이 높은 부분을 인터프리터 방식이 아니라 컴파일 후 기계어로 수행하는 과정을 거치기 때문에 처리 속도를 개선할 수 있다. 그러나 반복문 안에서 조건문이나 중첩된 반복문에 의해 수행 경로가 바뀌는 경우에는 새로운 경로를 추가하여 컴파일이 이루어지므로 처리 시간이 증가하는 단점이 있다[15]. 구글 크롬에 내장된 자바스크립트 엔진인 V8은 자바스크립트 코드에서 호출되는 모든 함수를 중간 코드를 거치지 않고 JIT 컴파일러를 사용하여 기계어로 직접 번역하고 수행한다. 이는 해석기에 의해 수행할 필요가 없으므로 모든 코드를 기계어로 빠르게 수행할 수 있는 장점이 있으며, 라이브러리 함수에 의존하는 기계어를 생산하는 특징이 있다[16]. 현재 다양한 단말 환경에서 사용되고 있는 이러한 자바스크립트 엔진들은 모바일 생태계의 제한적인 환경으로 인해 데스크탑 브라우저에서 만큼의 처리 속도를 제공하지 못하는 약점을 지니고 있다.

3. 자바스크립트 멀티 스레드

기존의 상용 브라우저에서 자바스크립트와 DOM은 단일 스레드로 동작하기 때문에 특정 시점에 오직 하나의 작업만을 수행한다. 이러한 처리 구조는 동시성을 처리하는 다른 프로그램에 비해 그 속도가 매우 느리다는 단점이 있다. 이를 해결하기 위해 Firefox 3.5의 DOM Workers와 HTML 5의 Web Worker 방법이 사용되고 있다[17][18]. 초기에 Firefox 3.5에 탑재된 DOM Workers는 플러그인 형태로 제공되어 실험적으로 사용되었다. Workers 자체는 DOM에 대한 접근 권한이 없고 단순한 연산만을 따로 처리할 수 있도록 되어 있다. 예를 들어 많은 반복문이나 재귀함수 호출과 같은 처리는 따로 js파일로 분리하여 처리하도록 되어 있다. 이러한 방식은 현재 표준을 제정 중인 HTML 5에도 영향을 끼쳐 Web Worker을 표준으로 포함하도록 하였다[19].

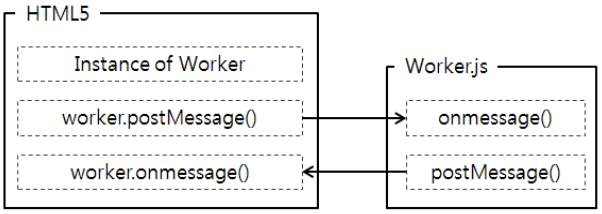


그림 4. HTML 5의 Web Worker 동작 방식
Fig. 4. Method of operation for Web Worker in HTML 5.

현재 HTML 5에서 지원하는 Web Worker는 멀티 프로세싱을 수행할 수 있고 postMessage를 이용하여 DOM에의 액세스가 가능하다. 이 기능은 단일 스레드에서 처리되는 것보다 동시성을 제공하여 자바스크립트 처리 속도를 증가 시키는 장점이 있다[20]. 그림 4는 HTML 5의 Web Worker 동작 방식을 나타낸다.

그림 4에서 Worker를 사용하기 위해서 인스턴스하게 되면 Worker.js 안의 자바스크립트 코드는 별도의 스레드에 내에서 실행되게 된다. 이때, HTML 5를 실행하는 메인 스레드와 통신을 하기 위해서 postMessage를 이용하여 메시지를 전달하고 전달 받은 메시지는 onmessage를 통해 처리된다. 또한 백그라운드에서 연산이 가능하기 때문에 많은 모바일 자원을 접근이 수행되는 WAC 기반의 앱에서는 활용될 가능성이 높다. Web Worker는 기능적인 면에서 자바스크립트 처리를 효율적으로 할 수 있으나 본 연구에서 측정한 실험 결과 모바일의 자원적 한계로 인해 데스크탑 PC 상의 브라우저에서보다 저하된 성능을 보였다. 따라서 제한적인 모바일 환경에서 복잡한 수행을 요구하는 WAC 응용 프로그램들의 기능 향상을 위해 Cyclostorm 클라우드 프레임워크를 제안한다.

Ⅲ. WAC 기반 앱의 자바스크립트 처리 향상을 위한 클라우드 프레임웍

1. Cyclostorm 구조

모바일의 제한적 자원 환경으로 인해 기본으로 내장된 브라우저의 자바스크립트 엔진은 성능 상 한계가 존재한다. 셀룰러 폰 단말의 하드웨어 성능이 데스크탑 PC 만큼의 수준으로 계속 향상되고 있지만 아직도 저사양의 단말이 존재한다. 이러한 문제점을 해결하기 위해서 낮은 수준의 하드웨어 자원을 갖고도 고급 하드웨어 수준만큼의 서비스를 제공하는 Cyclostorm 프레임워크를 소개한다. 이는 클라우드 컴퓨팅 원리를 이용하여 자바스크립트로 수행되는 WAC 기반 앱의 성능 향

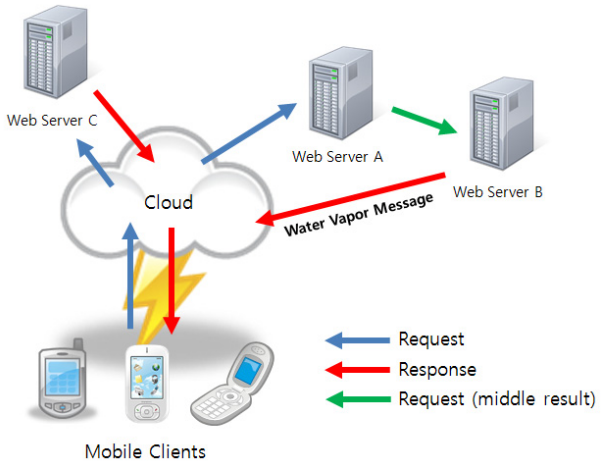


그림 5. Cyclostorm 개념도
Fig. 5. Concept of the Cyclostorm and operating structure.

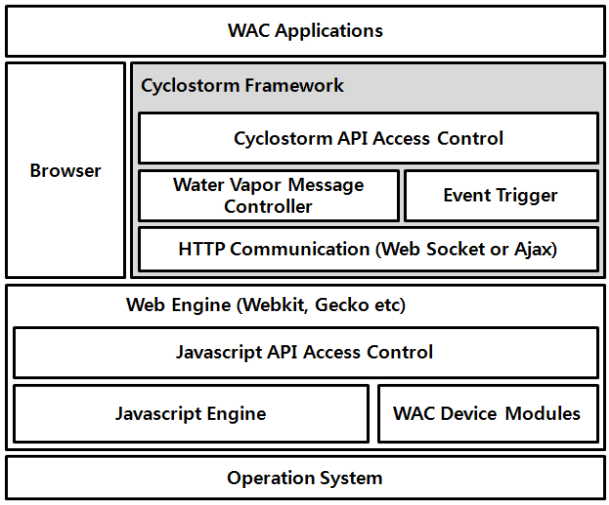


그림 6. Cyclostorm 프레임워크 구조
Fig. 6. Cyclostorm Framework Structure.

상을 위해 고안되었다. Cyclostorm은 클라이언트에서 많은 처리 시간을 요구하는 자바스크립트 프로세싱을 부분적으로 서버에 이양한 후 그 처리 결과만을 클라이언트에 불러들여 사용함으로써 모바일 단말 내의 자바스크립트 연산량을 줄이는 기능을 제공한다. 그림 5는 제안된 Cyclostorm의 개념도이다.

그림 5는 실행 시 많은 시간을 요구하는 자바스크립트를 서버에서 처리하기 위해 모바일 클라이언트가 클라우드 상에 있는 서버들 중 처리가 가능한 서버에 전달하는 과정을 보이고 있다. 자바스크립트 처리 방식에는 두 가지가 있는데 첫 번째는 단일 처리 구조이다. 그림 5에서 클라이언트는 웹 서버 C에게 클라이언트는 수행할 자바스크립트 함수 또는 그 내용을 담고 있는 파일을 WV (Water Vapor) Protocol의 형태로 구성한 후 서버에 전송한다. WV protocol은 JSON (JavaScript Object Notation) 구조로 되어 있는 메시지로서 웹 서버 C가 전송 받은 후 처리하고[21], 그 결과를 다시 WV의 형태로 변환한 후 해당 클라이언트에 전송하는 방식이다. 두 번째 처리 방식으로는 서버 릴레이 방식이 있는데 클라이언트가 전송하는 WV 메세지에 여러 서버 주소가 기록되어 있는 경우 그 기록을 보고 서버 사이를 이동하면서 처리하는 방식이다. 예를 들어 클라이언트가 웹 서버 A에게 자바스크립트 처리를 요청한 후 그 결과를 웹 서버 B에게 전달하여 최종 결과를 클라이언트에 재전송하는 방식이다. 그림 6은 이러한 처리 방식을 체계적으로 보이고 있는 Cyclostorm 프레임워크의 구조를 나타낸다.

그림 6에서 Cyclostorm은 브라우저에서 자바스크립트로 수행되는 방식을 사용하기 때문에 WAC 프레임워크 내에 포함될 수 있고 또한 수행되는 앱에 포함되어 처리가 가능하다. 기본적으로 Cyclostorm은 앱이 실행되는 브라우저와 서버간의 통신으로 동작된다. 자바스크립트 내에서의 통신은 동일 도메인인 경우에는 Ajax를 사용하고 서로 다른 도메인일 경우에는 Web Socket을 사용한다. WAC에서는 HTML 5를 표준으로 앱을 제작하기 때문에 Cyclostorm에서 또한 Ajax보다는 다른 도메인도 접속이 가능한 HTML 5의 Web Socket을 사용하여 통신을 수행한다. 그림 6의 HTTP Communication 부분이 서버와의 통신에 사용되는 모듈이다. 클라이언트와 서버간의 요청을 처리하기 위해서는 WV 메시지 포맷을 사용한다. 클라이언트에서 서버에 요청하는 내용에 자바스크립트로 구성된 처리 코드가 삽입되어 있기 때문에 요청을 받은 서버가 이를 처리하기 위하여 WV 포맷을 갖는 메시지를 전송한다. 따라서 이러한 WV 메시지를 생성 혹은 삭제하기 위한 Water Vapor Message Controller가 존재한다. Event Trigger는 WV 메시지 송수신시 발생되는 이벤트에 해당하도록 코드 내에 사전에 정의된 함수를 호출하는 역할을 한다. 덧붙여 Cyclostorm API Access Control은 Cyclostorm 패턴을 사용하기 위한 자바스크립트로 구성된 API를 제공하는 인터페이스의 모임을 나타낸다. 최종적으로 Cyclostorm은 자바스크립트가 동작되는 WAC Application 내에서 실행될 수 있도록 라이브러리의 형태로 제공된다.

2. Water Vapor Message

WV 메시지는 서버에서 실행될 함수 코드와 각 매개변수에 들어간 인자들의 값을 갖고 있는 일종의 실행 가능한 메시지이다. WV 메시지를 클라이언트로부터 수신한 서버는 그 안에 포함되어 있는 자바스크립트를 처리하고 그 결과를 클라이언트에 다시 전송하는데 이 때에도 WV 메시지에 결과 값이 포함되어 전송된다. 표 1은 요청 WV 메시지의 구조를 나타낸다.

표 1에서 Version은 WV 메시지의 버전을 나타내며, 단일 서버 혹은 임의의 여러 서버에 다중 요청을 한 경우 식별을 위한 Identification 값이 된다. Identification은 중복 오류를 피하기 위해 0부터 2^{32}-1 사이에서 임의의 값을 사용한다. Script는 서버에서 실행한 코드의 종류를 나타내며 Hosts는 서버에 접속할 주소를 뜻한다. 여러 서버에 걸쳐 처리를 요구하는 경우가 있기 때문에 경유하는 서버 주소를 나타낼 때에는 배열로 사용한다.

표 1. Water Vapor 메시지 요청 포맷
Table 1. Water Vapor Message Request Format.

이름	내용	예제 (JSON Data Type)
Version	Water Vapor Message Version	{"version":"1.0"}
Identification	1:N 요청 시 식별을 위한 값 (0~2^{32}-1 중 임의의 값)	{"id":"2100"}
Script	서버에서 실행할 스크립트 종류	{"script":{"language":"javascript", "type":"text/javascript"}}
Hosts	1:N 연결 시 접속할 서버 주소	{"hosts":{"a.com", "b.com"}}
Flag	요청 형태 값	{"flag", "start"}
Content	자바스크립트로 구성된 함수코드	{"content":{"args":[100,"hello"], "func":"functon a(arg1, arg2) { ..... }"};

표 2. Water Vapor 메시지 요청 Flag 종류
Table 2. Water Vapor Message Request Flags.

이름	내용
start	서버 측에 코드 실행 시작 요청
resume	실행이 일시 정지된 서버에 다시 실행 요청
suspend	코드 실행 일시 정지 요청
stop	서버 측에 코드 실행 정지 요청
post	실행 중인 서버에 비동기 메시지 처리 요청

Flag는 서버에 요청할 때 사용되는 값으로서 표 2에 그 종류가 나열되어 있다.

표 2에 보이는 Flag는 로컬 상에서 스레드를 동작시키는 함수 이름과 유사하다. start는 서버 측에 전달된 코드를 수행하는 것을 나타내며 resume과 suspend는 현재 서버에서 실행 중인 코드를 잠시 멈추거나 다시 시작하는 것을 의미한다. stop은 실행 중인 서버가 종료되도록 요청하는 것을 뜻한다. post는 실행 중인 서버에 현재 진행 상황이나 내부의 값을 가져오도록 비동기 요청을 할 때 또는 새로운 값이 할당될 때 사용된다. 표 1에 보이는 Content에는 서버에서 수행할 실제 함수 코드와 매개 변수에서 사용될 인자들의 값이 JSON 형태로 저장되어 있다. 표 3은 Content 내에서 팩토리얼 연산을 수행하는 함수를 자바스크립트 코드의 한 예로 표현한 것이다.

표 3에서 “args”는 “func”에 정의 되어 있는 함수를 실행할 때 매개변수의 값을 나타낸다. 이는 배열로 구성되어 있으며 0, 1, 2... 순으로 함수 실행 시 매개변수 인자 값으로 설정된다. “func”는 실행될 함수 코드를 나타낸다. “args” 배열 0번째에 있는 값은 서버가 “func” 수행 시 ‘function factorial(n)’ 중에 n의 값에 할당한 후 수행한다. 요청한 함수 코드의 수행이 종료되면 서버는 클라이언트에 결과를 통보하는데 이때 사용되는 WV 메시지의 응답 포맷은 표 4와 같다.

표 4에서 Version은 WV의 메시지 버전을 나타내며 Identification은 요청 메시지의 식별 값을 나타낸다. Script는 Content에 저장되어 있는 값을 실행할 수 있는 스크립트 종류를 뜻하며 Hosts는 응답 요청을 보내는 서버의 주소 배열로서 가장 앞쪽에 있는 배열 0번이 응답을 보내는 서버의 주소를 나타내고 그 외의 주소는 아직 요청이 남아 있는 서버의 주소를 뜻한다. 마지막으로 Flag는 응답 형태의 값을 나타내는 것으로 표 5에 나타난 바와 같다.

표 3. Content 예제 - 팩토리얼 연산 함수
Table 3. Example of Content - Factorial computation function.

```
{ "args" : ["10"], // 첫 번째 매개변수 인자 값
 "func" : " // 팩토리얼 연산 함수 코드
        function factorial(n) {
          if (n > 1) { return n*factorial(n-1); }
          else { return 1; } }" }
```

표 4. Water Vapor 메시지 응답 포맷
Table 4. Water Vapor Message Response Format.

이름	내용	예제 (JSON Data Type)
Version	Water Vapor Message Version	{"version":"1.0"}
Identification	클라이언트로부터 받은 응답 메시지의 버전	{"id":"2100"}
Script	클라이언트에서 실행할 스크립트 종류	{"script":{"language":"javascript","type":"text/javascript"}}
Hosts	현재 응답 메시지를 보낸 서버의 주소 (단, 요청 메시지의 호스트 주소가 여러개 존재할 경우 같이 보냄)	{"hosts":{"a.com", "b.com"}}
Flag	응답 형태 값	{"flag", "stop"}
Content	실행 후 최종 결과 값 혹은 요청에 해당한 값	{"content":{"data":"1234"}};

표 5. Water Vapor 메시지 응답 Flag 종류
Table 5. Water Vapor Message Response Flags.

이름	내용
success	전송된 요청이 현재 성공적으로 수행되고 있거나 수행 되었을 나타내고 그 결과를 포함한 메시지
error	서버 측 실행 중 발생한 오류에 대한 메시지
redirect	멀티 Hosts가 등록되어 있을 경우 요청을 다음 서버로 전송했다는 메시지
post	클라이언트에서 수행 가능한 코드 전송, Content 내에 클라이언트로 출력 함수가 존재할 경우 사용가능함.

표 5에서 success는 요청에 대한 성공을 의미하며 처리된 결과를 Content에 기록한 후 보내는 메시지이다. error는 서버가 실행 중에 오류가 발생했을 시 클라이언트에서 대체 코드를 삽입하여 서버 측에 요청한 후 서버가 코드를 변경하여 수행하도록 한다. redirect는 현재 수행 중인 서버에서 다른 서버로 요청이 옮겨갈 때 클라이언트에 이 사실을 통지하는 메시지이다. 이 때 클라이언트는 redirect 메시지를 받은 후 부터 변경된 서버 주소로 다시 연결을 맺고 통신이 가능하다. post는 서버가 클라이언트에게 코드 수행을 요청할 때 또는 클라이언트의 UI (User Interface)를 갱신하거나 현재 진행 상황을 나타낼 때 사용된다.

3. Cyclostorm 서비스

Cyclostorm은 Web Socket을 이용하여 여러 서버에 동시에 요청을 수행할 수 있다. HTML 5의 Web Socket은 서버와 연결을 하기 위해 3-핸드쉐이크를 실시한다. 이 때 수행 권한이 있는 클라이언트에 대해서만 연결을 확정하기 때문에 익명의 클라이언트 요청은 사전에 철회할 수 있다. 그림 7은 Cyclostorm의 신호 흐름을 나타낸다.

Cyclostorm의 초기 동작을 위해서는 Web Socket을 이용하여 서버와 연결될 수 있도록 3-핸드쉐이크를 실시한다. 실시간 양방향 통신이 가능하므로 Cyclostorm 서비스를 쉽게 이용할 수 있다. 그 다음 클라이언트에서 WV 메시지를 이용하여 수행할 코드를 서버에 요청한다. 응답을 수신한 서버는 권한이 있는 클라이언트인지 확인 한 후 즉시 수행 처리를 하고 success 응답 메시지를 클라이언트에 전송한다. success 메시지를 전송받은 Event Trigger는 클라이언트에서 지정한 응답 처리 함수를 호출한다. success 메시지를 확인 후 클라이언트는 다음 Event Trigger가 발생할 때까지 다른 작업을 실시한다. 그 후 서버에서는 최종 결과를 응답 메시지로 보내고 실행이 종료되었다는 의미로 stop 메시지를 클라이언트에 전송한 후 다른 클라이언트의 요청을 받기 위해 즉시 연결을 닫는다. 최종 결과를 통보 받은 클라이언트는 그 결과를 바탕으로 작업을 수행한다. 이로써 클라이언트에서 수행해야하는 복잡한 연산이나 시간이 많이 요구되는 과정이 빠르게 처리된 결과를 얻

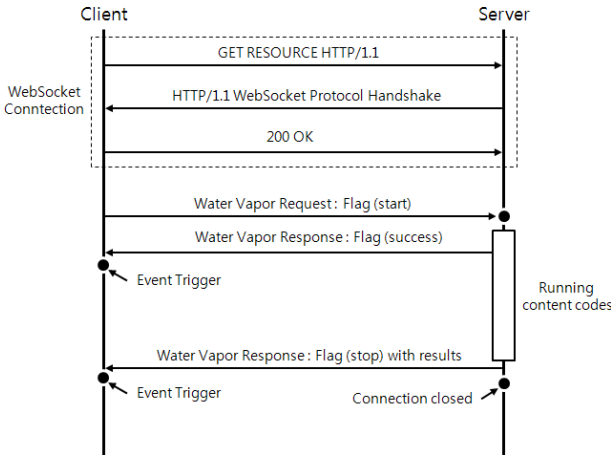


그림 7. Cyclostorm 신호 흐름도
Fig. 7. Cyclostorm Signal Flowgraph.

을 수 있다. 한편 클라이언트에서는 서버의 작업 상황 또는 특정 사항에 대한 요청을 추가적으로 보내는 경우가 발생할 수 있다. 예를 들어 100! 연산을 할 때 현재 어느 정도 계산이 되었고 중간 결과 값은 무엇인지 알고 싶을 때 서버에 요청할 수 있다. 이때는 WV 메시지의 post 요청을 이용하여 서버와 질의응답을 할 수 있다. 그림 8은 이러한 신호 흐름도를 나타낸다.

그림 8에서 클라이언트는 현재 실행 중인 서버에 WV 메시지의 post를 이용하여 특정 변수의 값이나 사용자가 정의한 데이터를 요청할 수 있다. 요청을 받은 서버의 Event Trigger가 요청 신호 처리를 하며 success 응답 메시지 content에 값을 저장하여 전송한다. 이후 클라이언트의 Event Trigger에 의해 사용자가 지정한 함수를 호출함으로써 질의응답 과정이 마무리된

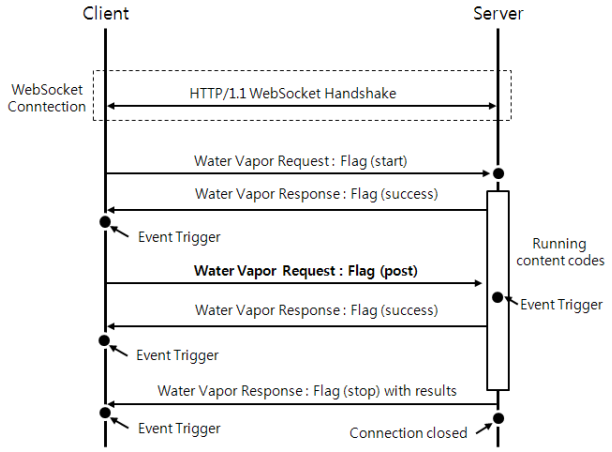


그림 8. 실행 중인 서버에 발생한 특정 요청 처리 흐름도

Fig. 8. Signal flowgraph for processing a specific request from an operating server.

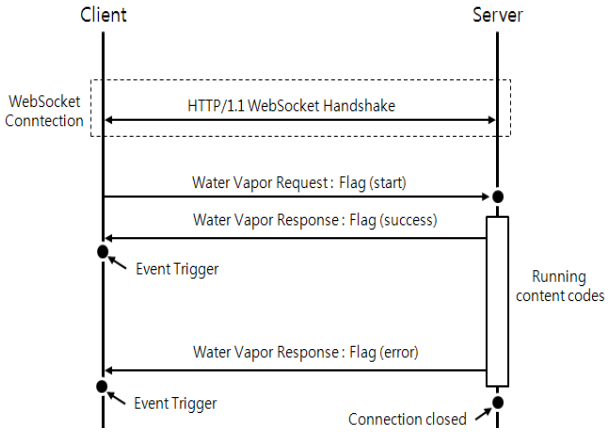


그림 9. 서버 실행 중 오류 발생 시 신호 흐름도

Fig. 9. Signal flowgraph of an error case for an operating server.

다. 서버에서 실행 시 사용자 코드의 잘못된 구문 또는 서버 시스템의 오작동으로 실행 오류가 발생할 경우 서버는 즉시 처리 프로세스를 종료하고 error 메시지를 전송한 후 연결을 끊는다. 그림 9는 이러한 오류 발생 시의 신호 처리 흐름도를 보이고 있다.

그림 9에 따르면 서버에서 오류가 발생했을 시에 WV 오류 메시지를 클라이언트에 전송 한 후 실행 중인 프로세스를 종료하고 즉시 연결을 끊는다. 다만 클라이언트에서 전송한 수행 코드에 오류가 발생하여 try-catch 문과 같은 예외 처리가 있을 경우에는 종료하지 않고 대체 코드를 기다린다. 그림 10은 오류 발생 시 대체 코드 요청 신호 흐름도를 나타낸다.

그림 10에서는 서버에서 실행 중 발생한 오류를 클라이언트에서 감지하고 대체 코드를 삽입 한 후 WV POST 메시지를 서버에 전송하는 과정을 보이고 있다. 대기 중인 서버는 클라이언트의 요청 코드를 대체 삽입한 후 다음 프로세스를 계속 진행한다. 이때 대체 코드에 의해 오류가 다시 발생할 경우 즉각 종료한 후 오류 메시지를 클라이언트에 보내고 연결을 끊는다. 오류가 발생하지 않으면 success 메시지를 클라이언트에 전송한다. 클라이언트가 다수의 서버에 처리 요청을 한 경우 도중에 처리된 결과를 클라이언트가 받을 수 있고 서버의 요청은 즉시 다른 서버로 전달된다. 그림 11은 1:N의 요청을 수행하는 Cyclostorm의 신호 흐름도를 나타낸다.

그림 11에서 서버 A에 요청을 한 후 처리가 끝난 결과를 클라이언트에 전달한다. 이후 서버 A는 연결을 닫

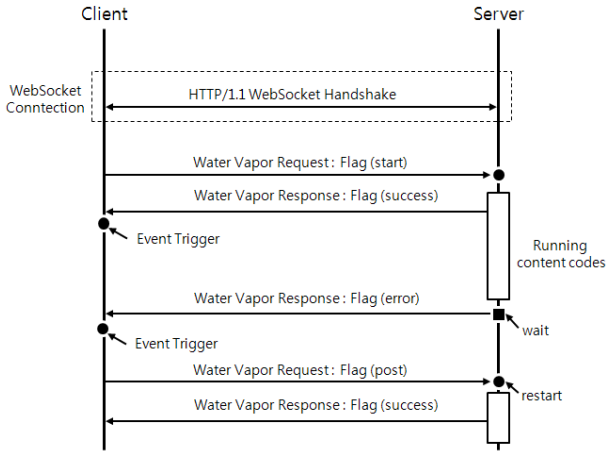


그림 10. 서버 실행 중 오류 발생 시 예외 처리 요청 신호 흐름도

Fig. 10. Signal flowgraph for an exceptional request processing of an error case for an operating server.

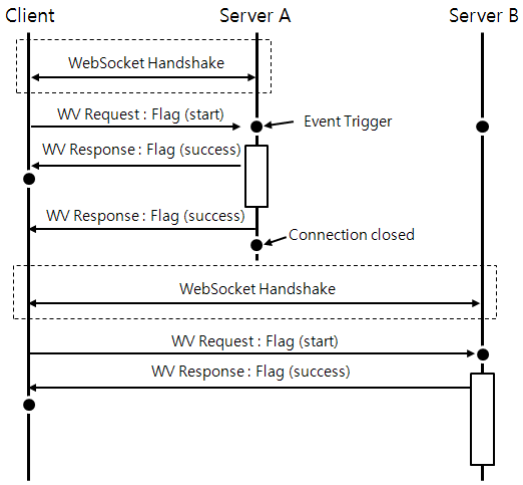


그림 11. 1:N 서버 연결 및 요청 신호 흐름도
Fig. 11. Signal flowgraph for a request and 1:N server connection.

고 클라이언트에서는 Hosts에서 다음 서버의 주소로 Web Socket 연결을 시도한다. 서버 B와 연결이 완료된 후에는 그 다음에 수행할 코드와 서버 A로부터 받은 결과 데이터를 함께 넣어 요청 메시지를 전송한다. 요청을 받은 서버 B는 그 후 처리된 결과를 최종적으로 클라이언트에 전달한다. Cyclostorm은 제한적인 모바일 클라이언트 환경에서 처리되기에는 너무 많은 시간을 요구하는 자바스크립트 코드를 서버에서 실행함으로써 빠르게 처리할 수 있는 장점을 가지고 있다. 일반적으로 자바스크립트는 클라이언트에서 사용하는 스크립트 언어로서 단순한 작업 용도로 사용되고 있다. 하지만 WAC 표준에 따라 개발된 앱은 Native 앱의 다양한 기능을 빠르고 정확하게 실행 할 수 있어야 한다. 브라우저 내에서 동작하는 자바스크립트는 단말의 하드웨어 성능에 따라 처리 속도가 크게 좌우된다. 스마트폰과 같은 고속 처리 능력을 보유한 모바일 단말은 점점 일반 데스크탑 PC의 수준만큼 처리 능력이 향상되므로 이런 문제는 해결 되리라 보인다. 하지만 낮은 사양의 단말은 항상 존재하고 소프트웨어의 구조가 점점 복잡해지는 현 시점에서 Cyclostorm은 모든 단말에 대해 서로 간에 큰 차이가 없는 처리 능력을 제공할 수 있다. 인터넷에 연결되어 있는 모바일 단말의 경우 서버 쪽의 강력한 하드웨어 자원을 이용하기 때문에 처리 속도 개선 및 클라이언트의 처리 부담을 줄일 수 있다. 현 시대에는 모바일 단말의 대부분이 3G, 4G, Wi-Fi와 같은 무선 네트워크를 이용할 수 있다는 점에서 Cyclostorm 서비스가 유용하게 활용될 수 있을 것으로 보인다.

Ⅳ. 시뮬레이션 및 성능 평가

1. 시뮬레이션 시스템

Cyclostorm 시스템의 성능 평가를 위해 HTML 5기반의 시뮬레이션 환경을 구축하였다. 클라이언트에서의 성능 평가를 위해 안드로이드 기반 Firefox 브라우저를 사용하였는데 이는 HTML 5 중 Web Worker를 지원하기 때문이다. 또한 Cyclostorm의 성능 평가를 위한 시뮬레이션 웹 애플리케이션을 제작하였는데 비교적 많은 연산량을 요구하는 얼굴 감지 (Face Detection), 프렉탈 (Fractal) 2D 렌더링, 레이트레이서 (Raytracer) 3D 렌더링을 사용하였다. 표 6은 Cyclostorm 성능 평가에 사용된 웹 애플리케이션의 사양과 기본 정보를 나타내고 있다.

표 6에 나타난 얼굴 감지 및 Fractal 2D 렌더링과 Raytracer 3D 렌더링 등은 모두 자바스크립트로 구성되어 있으며 HTML 5의 Canvas를 이용하여 이미지 처

표 6. Cyclostorm 성능 평가에 사용된 웹 애플리케이션
Table 6. Web application used for performance evaluation of the Cyclostorm.

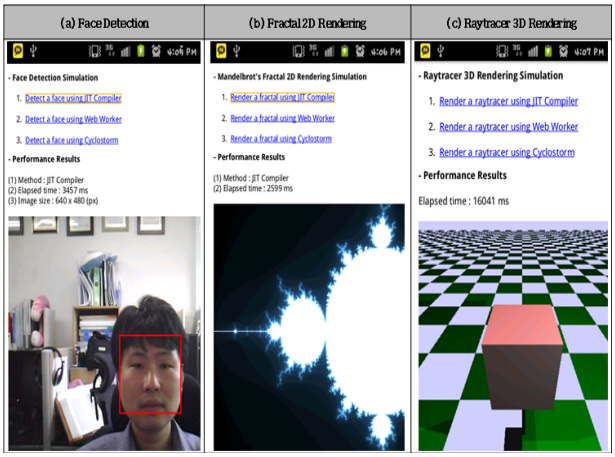


표 7. Cyclostorm 성능 평가에 사용된 서버 하드웨어 사양
Table 7. Server hardware specification used for performance evaluation of the Cyclostorm.

이름	하드웨어 구성
Cyclostorm Server	- CPU : Intel(R) Core(TM) 2 Quad CPU (Q8200) @ 2.33 GHz - RAM : 4GB - Ethernet : Atheros PCI-E 1GB - Operation System : Windows XP Service Pack 3

리를 하는 애플리케이션이다. 모바일 단말에 비해 고사양의 하드웨어를 장착한 데스크탑 브라우저에서도 약 2초 이상의 처리시간이 요구될 정도로 연산량이 많은 애플리케이션이다. Cyclostorm은 서버에서 처리된 결과를 클라이언트에서 사용할 수 있도록 해주는 방식을 사용하였으며 본 연구에서 제안한 방식의 실험을 위해 사용된 서버의 하드웨어 사양은 표 7과 같다.

표 7에서 Cyclostorm 시뮬레이션에 사용된 서버는 웹 서비스가 가능하도록 설계되었다. 웹 서비스에 사용되는 프로그램에는 Apache 2.0과 Web Socket 서비스를 위한 PHP (Hypertext Preprocessor) 모듈이 설치되어 있다. 서버에 설치된 자바스크립트 엔진은 구글 크롬의 V8이며 Cyclostorm 서비스를 지원하기 위해 필요한 모듈이 구현되었다. 시뮬레이션을 위한 클라이언트로는 모바일 단말의 하드웨어 성능에 따라 브라우저 내 자바스크립트 엔진의 처리 속도가 다르기 때문에 현재 상용화 되어 있는 단말들을 사용하였다. Cyclostorm의 성능 평가를 위해 현재 상용화되어 있는 안드로이드 기반의 스마트폰 네 기종을 선정하였으며 이들은 표 8과 같다.

표 8에 제시된 모바일 단말에서 표 6에 나타난 웹 애플리케이션을 실행하고 그 성능을 평가하였다. 처리 시간 측정을 위해 이를 위한 모듈을 웹 애플리케이션에 구현하였다. JIT compiler와 Web Worker 및 Cyclostorm 방식을 이용하여 각각 50번씩 시뮬레이션을 실행하고 평균 처리 시간을 측정한 후 그 결과를 분석 하였다.

표 8. Cyclostorm 성능평가에 사용된 모바일단말 및 사양
Table 8. Mobile terminals and specifications used for performance evaluation of the Cyclostorm.

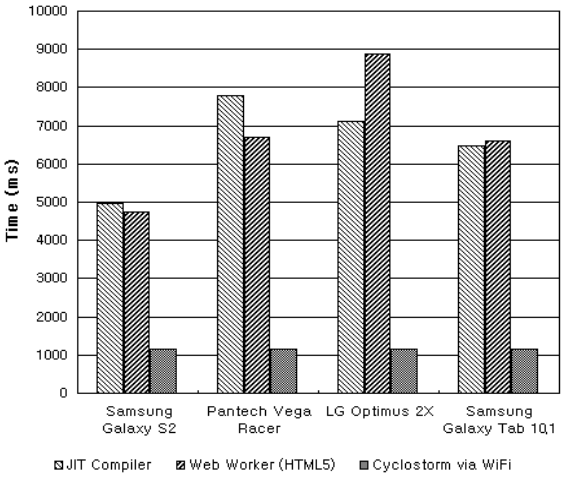
이름	CPU	RAM	Platform
Samsung Galaxy S2 (SHW-M250S)	1.2 Ghz Dual Core ARM Cortex-A9 (Exynos 4210)	1GB (DDR2)	Android 2.3
Samsung Galaxy Tab 10.1 (SHW-M380W)	1.0 Ghz Dual Core Nvidia Tegra 2	1GB (DDR2)	Android 3.1
Pantech Vega Racer (IM-A770K)	1.5 Ghz Dual Core Snapdragon MSM8660	1GB (DDR2)	Android 2.3
LG Optimus 2X (LG-SU660)	1.0 Ghz Dual Core Cortex-A9	512 MB	Android 2.2

2. 성능 평가 및 결과 분석

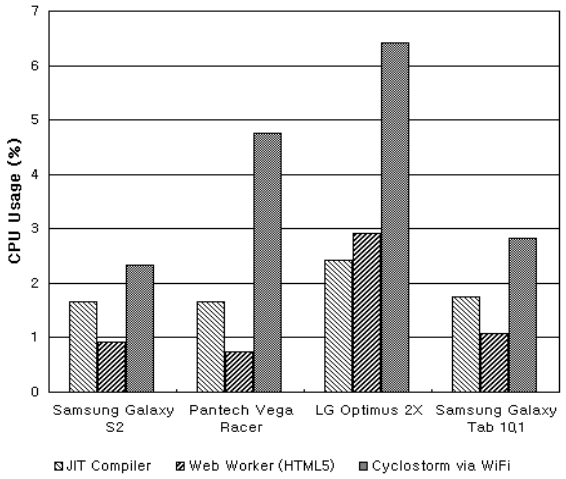
주어진 클라이언트 단말들을 이용하여 자바스크립트로 구현된 얼굴 감지(Face Detection) 애플리케이션의 Wi-Fi 상에서의 처리 시간 및 CPU 점유율, 메모리 사용량을 측정한 결과는 그림 12와 같다.

그림 12의 (A)는 웹 애플리케이션의 처리 시간을 나타낸다. (A)에서 JIT compiler 방식은 대체로 단말의 하드웨어 사양에 따라 처리 시간이 다르게 측정되었다. 상대적으로 다른 단말들에 비해 고사양인 삼성 갤럭시 S2 (SHW-M250S)가 처리 시간이 가장 짧게 측정되었다. HTML 5에서 지원하는 Web Worker의 경우 대체로 JIT Compiler 방식보다는 빠르게 처리되었으나 멀티태스킹 환경으로 인해 측정 시간이 일정하지 않았다. 비교적 JIT compiler 방식과 Web Worker 방식은 측정 결과로 볼 때 크게 성능 차이가 나지 않았다. 본 논문에서 제안하는 Cyclostorm 방식은 멀티태스킹 환경에 영향을 받지 않고 서버 측에 존재하는 고성능의 하드웨어를 이용하는 방식이기 때문에 모든 단말에서 빠르게 처리되었다. 동일한 서버를 사용하기 때문에 처리 시간 또한 거의 비슷하게 측정되었다. (B)의 CPU 점유율은 Cyclostorm 방식과 비교하여 볼 때, JIT Compiler 방식과 Web Worker 방식이 더 낮게 측정되었다. 이는 Cyclostorm이 웹 응용프로그램의 처리를 요청하기 위해 서버와 연결하고 해당 결과를 수신하는 동안 모바일 네트워크 자원을 사용하고 Cyclostorm 프레임워크를 로드하기 때문인 것으로 보인다. (C)에서는 Cyclostorm이 사용하는 메모리양이 가장 적게 측정되었다. 안드로이드 플랫폼에서는 자바를 사용하기 때문에 메모리 관리를 Garbage Collector가 하는데 Cyclostorm은 실행 시 서버측의 메모리를 활용하기 때문에 전반적으로 메모리 사용량이 다른 방식에 비해 낮게 측정된 것으로 보인다.

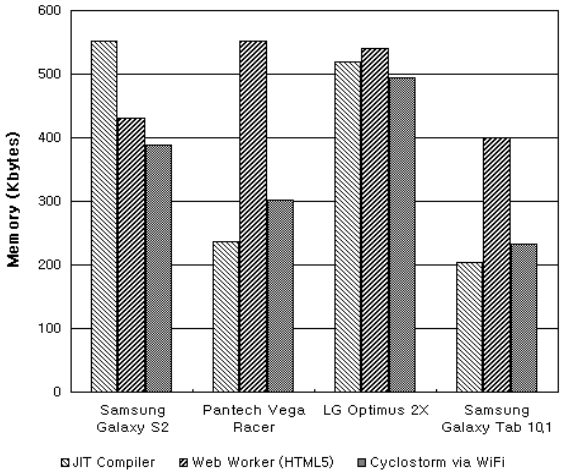
그림 13은 프렉탈 2D 렌더링 웹 애플리케이션을 Wi-Fi 상의 단말에서 실험하여 CPU 동작 점유율을 측정한 결과를 보이고 있다. (A)에 따르면 얼굴 인식 응용 프로그램에 비해 프렉탈 2D 렌더링의 처리 시간은 그리 길지 않지만 성능 평가 결과는 그림 12와 같이 하드웨어 성능이 가장 좋은 삼성 갤럭시 S2 (SHW-M250S)가 가장 빠르게 측정되었다. Cyclostorm



(A) Processing Time



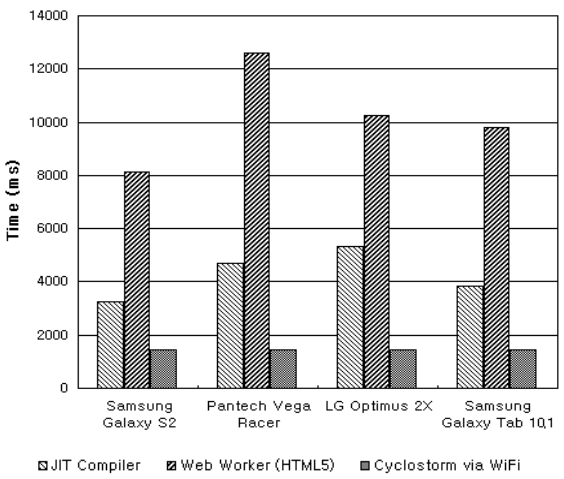
(B) CPU Occupancy Rate



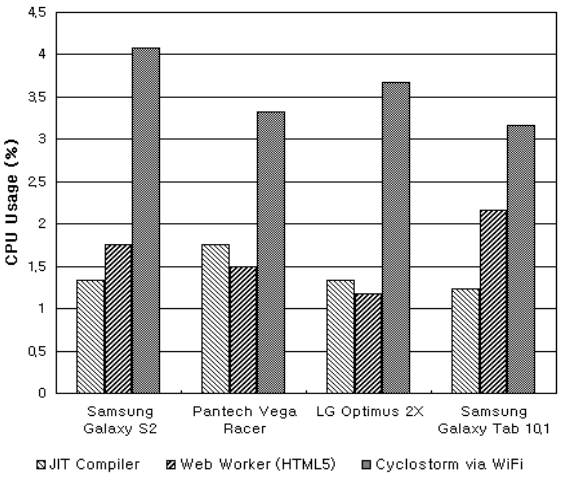
(C) Memory Usage

그림 12. Wi-Fi를 이용한 얼굴 감지(Face Detection) 웹 애플리케이션 성능 평가 결과 ((A)-(B)-(C))

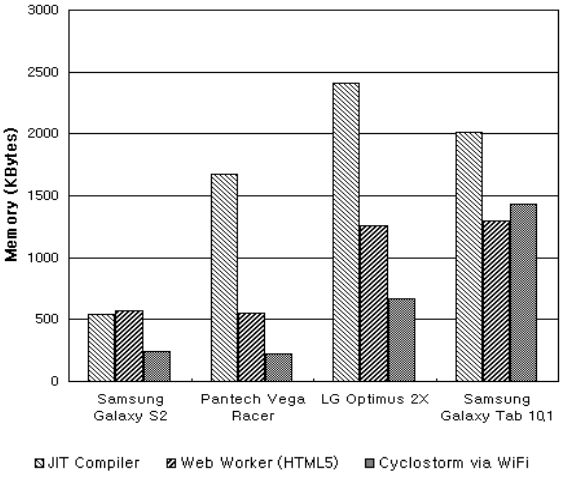
Fig. 12. Performance evaluation results for Face Detection web application using Wi-Fi ((A)-(B)-(C)).



(A) Processing Time



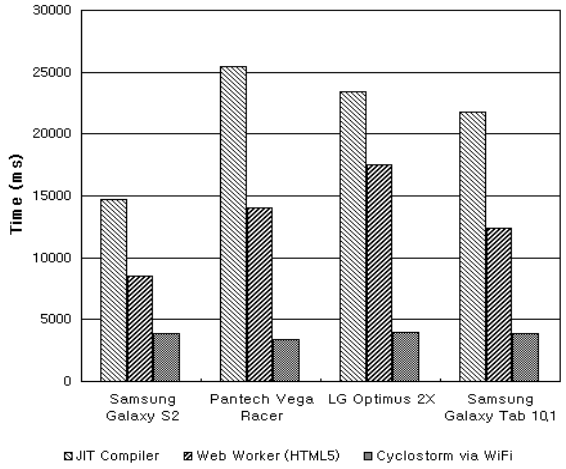
(B) CPU Occupancy Rate



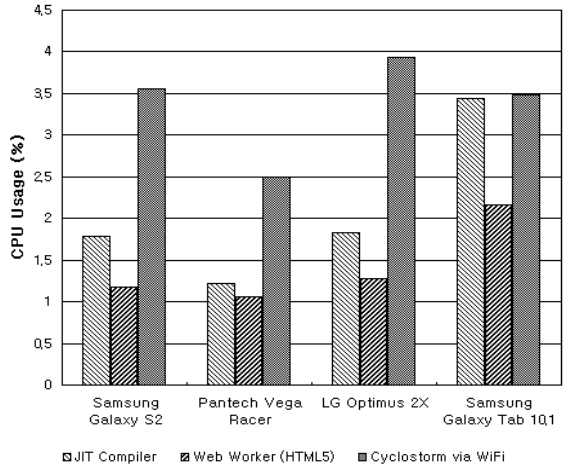
(C) Memory Usage

그림 13. Wi-Fi를 이용한 프렉탈 2D 렌더링 웹 애플리케이션 성능 평가 결과 ((A)-(B)-(C))

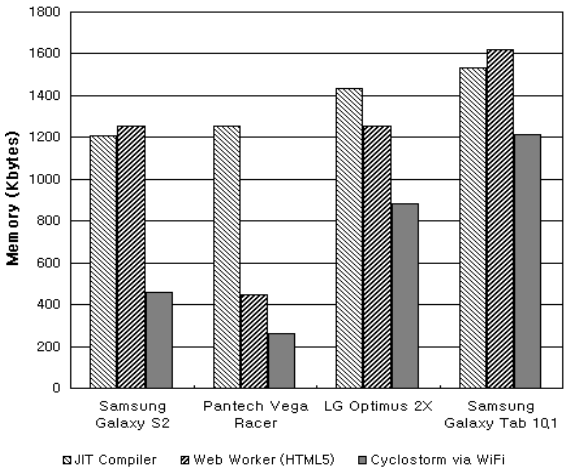
Fig. 13. Performance evaluation results for Fractal 2D rendering web application using Wi-Fi ((A)-(B)-(C)).



(A) Processing Time



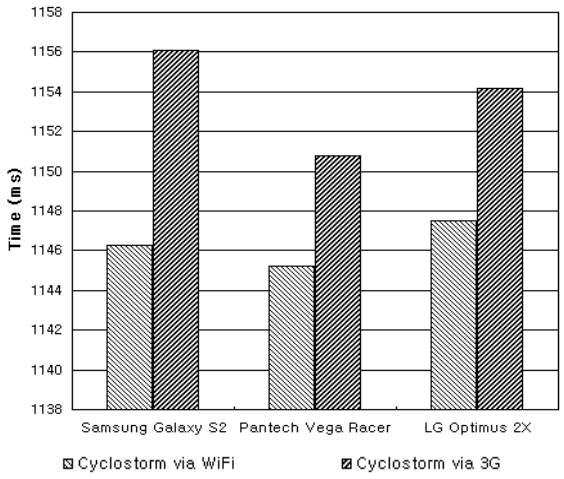
(B) CPU Occupancy Rate



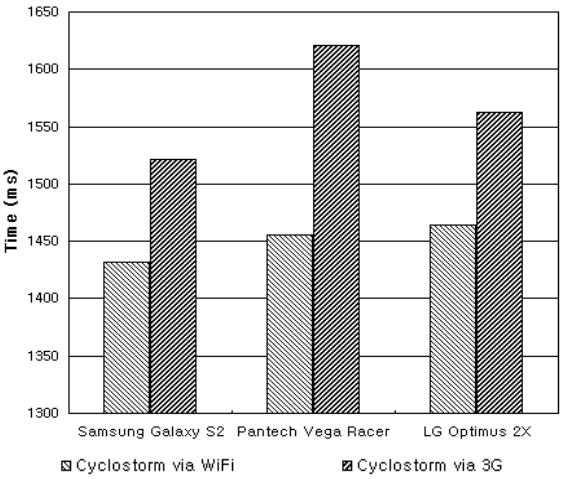
(C) Memory Usage

그림 14. Wi-Fi를 이용한 레이트레이서(Raytracer) 3D 렌더링웹 애플리케이션 성능 평가 결과 ((A)-(B)-(C))

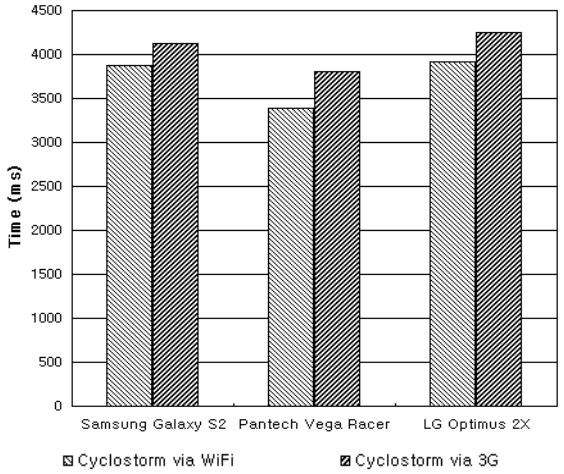
Fig. 14. Performance evaluation results for Raytracer 3D rendering web application using Wi-Fi ((A)-(B)-(C))



(A) Face Detection



(B) Fractal 2D Rendering



(C) Raytracer 3D Rendering

그림 15. Cyclostorm의 3G와 Wi-Fi 성능 평가 결과 ((A)-(B)-(C))

Fig. 15. Performance evaluation results for the Cyclostorm between 3G and Wi-Fi ((A)-(B)-(C))

의 경우 얼굴 인식과 마찬가지로 서버 측의 자원을 활용하기 때문에 다른 방식에 비해 빠르게 처리되었고 이와 동일한 수준의 성능을 보였다. Fractal 2D 렌더링의 경우 소수점 이하의 연산이 많은데 이는 모바일 단말에서 하드웨어적으로 많은 부담으로 작용한다. 이 때문에 메인 스레드에서 동작하는 JIT Compiler의 경우 UI 출력에 소비되는 연산들이 일시 정지되고 오직 소수점 연산에 많은 CPU 자원이 할당되어 멀티 스레드를 생성해서 동작하는 Web Worker보다 빠르게 처리되었다. (B)의 CPU 사용량에 있어서는 네트워크 자원을 사용하기 때문에 Cyclostorm의 경우 얼굴인식과 마찬가지로 다른 방식에 비해 높은 점유율을 보였다. (C)의 메모리 사용량 또한 그림 12와 같이 실행 시 사용되는 메모리를 서버 측에서 사용하기 때문에 Cyclostorm의 경우 전반적으로 다른 방법들보다 우월한 결과를 보였다.

그림 14는 레이트레이서 3D 렌더링 웹 애플리케이션을 Wi-Fi 상의 단말에서 실험하여 프로세싱 타임, CPU 점유율, 메모리 사용량 등을 측정한 결과를 보이고 있다. 3D 렌더링은 처리되는 연산량이 얼굴 인식 및 프렉탈 2D 렌더링에 비해 많기 때문에 처리 시간이 모든 단말에서 더 길게 측정되었다. 단말의 로컬 브라우저에서 처리되는 JIT compiler와 Web Worker 방식에 비해 Cyclostorm은 서버에서 실행된 결과만을 보이기 때문에 빠르게 처리되었다. 레이트레이서 3D 렌더링의 경우 프렉탈 2D 보다는 소수점 연산이 적고 멀티 스레드 안에서 처리되는 데이터가 적어 JIT Compiler 보다 Web Worker의 방식이 더 빠른 결과를 도출하였다. (B)에서도 Cyclostorm의 경우 다른 응용프로그램에 비해 연산량이 많기 때문에 상대적으로 높은 CPU 사용량을 보였는데 이는 모바일 네트워크 자원을 사용하기 때문인 것으로 판단된다. (C)에서의 메모리 사용량은 3D 렌더링에 필요한 기본 메모리양이 많기 때문에 다른 성능 평가 측정치에 비해 비교적 높게 나왔다. 하지만 Cyclostorm의 경우 실행 시 서버 측의 메모리를 활용하기 때문에 전반적으로 낮게 측정되었다.

그림 15에서는 3G 통신이 지원되지 않는 삼성 갤럭시 탭 10.1 (SHW-M380W)을 제외하고 나머지 세 개의 클라이언트 단말을 이용하여 Cyclostorm의 3G와 Wi-Fi에서의 처리 시간을 측정 및 비교한 결과를 나타낸다. 3G 환경은 Wi-Fi에 비해 전송 측면에서 상대적으로 속도가 느리기 때문에 처리 시간이 Wi-Fi 보다는 많이 소요되었다. 하지만 측정 결과 약 10~400ms로 미미한 차이를 보였다. 그 이유는 3G 네트워크가 유선 네트워크와 연결되어 있는 Wi-Fi와 비슷한 속도 수준의 서비스가 이루어지고 있기 때문이라 추정된다. 얼굴 인식 및 프렉탈 2D 렌더링과 레이트레이서 3D 렌더링의 성능 평가에서 JIT Compiler 방식과 Web Worker 방식보다는 Cyclostorm의 처리 속도가 Wi-Fi 상에서 최소 3배에서 최대 6배 정도 빠른 결과를 보였다. 또한 Cyclostorm 방식을 사용하였을 경우 모바일 환경에서 하드웨어에 상관없이 모두 비슷한 처리 시간이 측정되어 자바스크립트로 구현되는 복잡한 연산이나 렌더링은 기존 방식에 비해 효과적으로 처리가 가능하고 동일한 수준의 성능으로 서비스를 제공할 수 있다. CPU 사용 점유율 측면에서는 Cyclostorm 프레임워크와 모바일의 네트워크 자원을 동시에 사용하기 때문에 기존 방식보다는 다소 높은 CPU 사용량을 보였다. 하지만 이러한 CPU 사용량은 4~7% 사이로 기존의 방식과 그리 많은 차이를 보이지는 않았다. 또한 전체 CPU 사용량에 비하면 매우 낮은 양을 유지하기 때문에 단말의 다른 성능에도 큰 영향을 미치지 않는다. 메모리 사용량에서도 실행 시 사용되는 메모리를 서버 측에서 활용하기 때문에 Cyclostorm의 경우 다른 방법들보다 비교 우위의 결과를 보였다.

Ⅴ. 결론 및 향후 연구

WAC 기반의 모바일 웹 애플리케이션 소프트웨어 보급이 확산됨에 따라 자바스크립트와 HTML로 구현된 애플리케이션의 처리 성능이 점차 이슈화되고 있다. 단순 처리 기능만을 갖는 구조라면 현재 상용화된 브라우저에서도 문제가 없지만 사용자와의 소통을 위한 자바스크립트의 처리량이 점점 증가 할수록 브라우저의 처리 부담 또한 증가된다. 상용화된 모바일 브라우저에서는 자바스크립트 처리에 시간 및 용량에 제한을 두고 있다. 브라우저에서의 처리는 그 한계가 존재하며 특히 많은 연산 처리를 요구하는 3D 애플리케이션의 경우 Native 애플리케이션에서처럼 빠른 처리 속도를 보장하지 못한다. 그 대안으로 HTML 5에서는 기존의 자바스크립트에서 지원하지 않는 멀티 스레드 구현을 위한 Web Worker를 제공하고 있다. Web Worker는 싱글 스레드에서 처리하는 일정 부분을 별도의 스레드를 통해서 처리하는 메커니즘을 제공한다. 하지만 Native 애플리케이션만큼의 처리 능력은 보장하지 못하며 근본적인 처리 속도 개선에는 미흡하다. 또한 모바일의 하드웨어가 저사양인 경우 Web Worker만으로

는 하드웨어의 부담을 경감시키지 못한다. Cyclostorm은 이러한 모바일 상의 자바스크립트 처리를 클라우드 상의 서버에 이전함으로써 클라이언트의 제한적인 자원의 한계를 극복하고 고성능의 처리 과정을 제공하여 Native 애플리케이션 만큼의 성능을 보장하도록 하는 서비스이다. 성능 평가 실험에서 Cyclostorm은 기존의 모바일 브라우저상에서의 자바스크립트 처리보다 최대 6배 정도 빠른 처리 결과를 보이고 있고 HTML 5의 Web Worker보다는 약 4~6배정도 빠른 성능을 보이고 있다. CPU 사용량은 4~7% 사이로 전체 사용량에 비하면 낮은 양이며 단말의 성능에는 큰 영향을 미치지 않는다. 또한 메모리 사용량에서도 실행 시 사용되는 메모리를 서버 측에 두고 활용하기 때문에 전반적으로 낮은 양이 측정되었다. 성능 비교 평가 결과를 통하여 Cyclostorm은 브라우저가 갖고 있는 한계를 극복하고 기존의 웹 애플리케이션 기능을 획기적으로 개선할 수 있음을 보여주었다. 또한 상대적으로 Wi-Fi보다 느린 속도를 갖는 3G에서도 큰 차이 없이 서비스가 가능함을 보여 주었고 향 후 4G 환경에서도 Cyclostorm을 활용할 수 있는 효과적인 클라우드 컴퓨팅에 대해서 지속적으로 연구할 계획이다.

참 고 문 헌

[1] Kincaid, Jason, “The Wholesale Applications Community Sounds Like A Disaster In The Making”, *TechCrunch*, February 2010.
[2] “Wholesale Applications Community”, *Wikipedia*, April 2013, http://en.wikipedia.org/wiki/Wholesale_Application_Community.
[3] “Core Specification:Introdcution”, *WAC*, July 2011, http://specs.wacapps.net/2.0/jun2011/core/index.html#toc-introduction
[4] “Core Specification:Web Standards”, *WAC*, July 2011, http://specs.wacapps.net/2.0/jun2011/core/web-standards.html
[5] 이성원, 문수묵, “적시 컴파일러를 내장한 자바스크립트 엔진의 성능 분석”, *대한전자공학회 2008년 정기총회 및 추계종합학술대회*, 921-922쪽, 2008.11
[6] “What is SpiderMonkey?”, *Mozilla Developer Network*, November 2013, http://www.mozilla.org/js/spidermonkey/
[7] Peter Lubbers, Brain Albers, Frank Salim, “Pro HTML 5 Programming”, *Apress*, September 2010, ISBN13: 978-1-4302-2790-8
[8] 노병규, “WAC에서의 효율적인 앱 검증 방안”, *한국인터넷정보학회 학술발표대회*, 381-385쪽, 2010년 6월
[9] “Device Specifications”, *WAC*, July 2011, http://specs.wacapps.net/2.0/jun2011/
[10] “BONDI 1.1 Approved Release”, *OMTP BONDI*, 2 September 2010, http://bondi.omtp.org/1.1/
[11] “BONDI 1.5 APIs Public Working Draft v1”, *OMTP BONDI*, June 2010, http://bondi.omtp.org/1.5/pwd-1/
[12] 오형석, 문수묵, “동적 컴파일기반의 자바스크립트 엔진에서의 동적 타입 처리: TraceMonkey와 V8”, *대한전자공학회 2008년 정기총회 및 추계종합학술대회*, 861-862쪽, 2008년 11월
[13] 정원기, 문수묵, “Sunspider 자바스크립트 벤치마크의 유용성 평가”, *대한전자공학회 2008년 정기총회 및 추계종합학술대회*, pp.865-866, 2008년 11월
[14] 오형석, 문수묵, “클라이언트 AOTC를 활용한 TraceMonkey 구현”, *대한전자공학회 2009년 정기총회 및 추계종합학술대회*, 551-552쪽, 2009년 11월
[15] HA, Jungwoo, et al. “A concurrent trace-based just-in-time compiler for JavaScript”. *University of Texas, Austin, Tech.* Rep. TR-09-06, 2009.
[16] Gal, Andreas, Christian W. Probst, and Michael Franz. “HotpathVM: an effective JIT compiler for resource-constrained devices.”, *Proceedings of the 2nd international conference on Virtual execution environments,* ACM, pp. 144-153, June 2006.
[17] “DOM Workers”, *Mozilla Developer Network*, April 2013, http://developer.mozilla.org/en/Using_web_workers
[18] World Wide Web Consortium, “HTML 5 working draft”, *W3C*, April 2013, http://dev.w3.org/html5/spec/Overview.html
[19] Eric Bidelman, “The Basics of Web Workers”, *Html5 Rocks Tutorials*, March 2013, http://www.html5rocks.com/en/tutorials/workers/basics/
[20] Web Hypertext Application Technology Working Group, “Web workers draft recommendation”, *WHATWG community*, April 2013, http://www.whatwg.org/specs/web-workers/current-work/
[21] Douglas Crockford. “The application/json media type for javascript object notation (JSON)”, *JSON.org*, July 2006, http://tools.ietf.org/html/rfc4627

저 자 소 개

방 지 웅(학생회원)

2008년 단국대학교 컴퓨터과학과 학사 졸업.

2010년 단국대학교 컴퓨터과학과 석사 졸업.

2010년~현재 단국대학교 컴퓨터과학과 박사 과정.

<주관심분야 : Mobile Platform, Web 표준, Cloud Computing, Mobile Multimedia Processing>

김 대 원(정회원)-교신저자

1993년 중앙대학교 공과대학 전자공학과 학사 졸업.

1996년 University of Southern California, Electrical and Computer Engineering, M.S. 졸업.

2002년 Iowa State University, Electrical and Computer Engineering, Ph. D. 졸업.

<주관심분야 : Mobile Platform, Application S/W, Digital Multimedia Data Processing, Non-Destructive Evaluation, Ultrasonic Signal Processing>