JANT 한국항행학회논문지
Journal of Advanced Navigation Technology

기반 기술

J. Adv. Navig. Technol. 23(1): 55-60, Feb. 2019

서비스워커 기반의 캐싱 시스템을 이용한 웹 콘텐츠 로딩 속도 향상 기법

Web Content Loading Speed Enhancement Method using Service Walker-based Caching System

김 현 국* · **박 진 태** · **최 문 혁** · **문 일 영**
한국기술교육대학교 컴퓨터공학부

Hyun-gook Kim* · **Jin-tae Park** · **Moon-Hyuk Choi** · **Il-young Moon**
Department of Computer Engineering, KOREATECH, Chungcheongnam-do, 31253, Korea

[요 약]

웹은 사람들의 일상생활에 있어 가장 밀접한 기술 중 하나로 오늘날 대부분의 사람들은 웹을 통해 데이터를 공유하고 있다. 단순 메신저, 뉴스, 영상뿐만 아니라 다양한 데이터가 현재 웹을 통하여 전파되고 있는 셈이다. 또한 웹 어셈블리 기술이 등장하면서 기존 네이티브 환경에서 구동되던 프로그램들이 웹의 영역에 진입하기 시작하면서 웹이 공유하는 데이터는 이제 VR/AR 콘텐츠, 빅데이터 등 그 범주가 점차 넓어지고, 크기가 거대해지고 있다. 따라서 본 논문에서는 브라우저에 종속적이지 않고 독립적으로 동작이 가능한 서비스워커와 웹 브라우저 내에 데이터를 효과적으로 저장할 수 있는 캐시 API를 활용하여 웹 서비스를 사용하는 사용자들에게 웹 콘텐츠를 효과적으로 전달할 수 있는 방법을 제시하였다.

[Abstract]

The web is one of the most intimate technologies in people's daily lives, and most of the time, people are sharing data on the web. Simple messenger, news, video, as well as various data are now spreading through the web. In addition, with the emergence of Web assembly technology, the programs that run in the existing native environment start to enter the domain of the Web, and the data shared by the Web is now getting wider and larger in terms of VR / AR contents and big data . Therefore, in this paper, we have studied how to effectively deliver web contents to users who use Web service by using service worker that can operate independently without being dependent on browser and cache API that can effectively store data in web browser.

Key word : Web performance, Web cache, Service worker, Progressive Web App.

https://doi.org/10.12673/jant.2019.23.1.55

This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-CommercialLicense(http://creativecommons.org/licenses/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Received 28 December 2018; **Revised** 28 January 2019
Accepted (Publication) 26 February 2019 (28 February 2019)

***Corresponding Author ; Hyun-gook Kim**

Tel: +82-10-5025-0874
E-mail: hy1392@koreatech.ac.kr

Ⅰ. 서 론

1989년 CERN (conseil europeenne pour la recherche nucleaire)의 팀 버너스 리에 의해 월드 와이드 웹 기술이 처음으로 정의되고, 1994년 웹 기술 표준화를 위한 단체 World Wide Web Consortium (W3C)이 창립된 이후로, 웹 기술은 인터넷 및 네트워크 기술과 융합하여 크게 발전하였다 [1]. 이에 따라 기존에 사람들이 종이 신문, 잡지, 편지 들을 통해 정보를 주고받았던 것과 달리 현재 우리는 E-mail, 전자 신문, e-book, 스트리밍 비디오 등 다양한 웹 기반의 서비스들을 이용해 데이터를 주고받고 있다. 즉, 웹이 일반 사용자들의 생활에 녹아들에 정보 공유에 있어 가장 큰 역할을 하고 있다고 말할 수 있다. 뿐만 아니라 하드웨어 장비 기술이 발전하면서 데스크톱뿐만 아니라 스마트폰, 스마트워치, IoT 가전제품, Raspberry Pi 등 다양한 기기들에 적용되기 시작하면서 웹이 우리 삶에 미치는 영향력이 점차 확장되고 있는 추세이다.

실제로 정부에서 제공하는 서비스 중 하나인 e-나라지표의 가구 인터넷 보급률을 보면 2017년 기준 국민의 87.6%가 인터넷을 사용하고 있는 것으로 나타날 정도로 거의 대부분의 사람이 웹에 접근이 가능하며, 실제 서비스를 사용하고 있다는 것을 확인할 수 있다 [2].

그리고 웹을 사용하는 사용자가 지속적으로 증가함에 따라, 웹을 이용해 사용자들에게 제공되는 서비스 및 데이터의 종류와 양도 기하급수적으로 증가하기 시작하였다. 기존의 웹 브라우저의 목적이 단순 연구 데이터인 텍스트 이미지를 공유하는 데에 초점을 맞추었던 것과 달리 현재의 웹은 동영상 스트리밍, 게임, VR(virtual reality)/AR(augmented reality) 등 좀 더 복잡하고 거대한 데이터를 공유하는 데에 주목하고 있다. 실제로 2017년 말 W3C에서 기존의 네이티브 환경에서 작성된 C, RUST등 네이티브 언어 기반의 프로그램을 이진 바이너리 코드 형태인 Wasm으로 변환하여 웹 브라우저에서 자바스크립트 모듈의 형태로 구동하는 WebAssembly라는 기술에 대한 working 그룹을 구성하였다. WebAssembly 기술을 활용하면 기존에 개발된 데스크톱 환경의 프로그램들을 웹 환경에서 쉽게 구동할 수 있게 되며, 실제 구동 속도 또한 네이티브 환경과 거의 흡사한 속도로 사용자에게 제공할 수 있게 된다 [3]. 즉, 웹과 네이티브의 경계가 사라지게 되어 브라우저만을 이용하여 모든 서비스를 제공받을 수 있게 되는 것이다.

하지만 이러한 과정에서 한 가지 문제점이 발생하였다. 웹을 통해 제공되는 데이터의 크기는 점차 거대해지고 있으나, 실제 WebAssembly를 이용하여 게임을 사용자에게 제공하고자 한다면, 컴파일 된 전체 게임의 Wasm파일을 사용자가 다운로드 받아야 한다. CDN(content distribution network) 제공 업체인 Akamai의 “인터넷 현황 보고서”에 따르면 대한민국의 평균 광대역 인터넷 속도 (Mbps)의 경우 28.6 Mbps로 양호한 모습을 나타내지만, 전 세계 평균의 경우 7.2 Mbps로 느린 속도를 나타냈다. 게다가 모바일 환경의 측면에서는 최악의 경우인 베네수엘라 기준 2.8 Mbps의 속도를 나타내는 것을 확인할 수 있다 [4]. 따라서 대용량의 데이터를 웹을 통해 사용자에게 제공하는 경우 국가 및 환경에 따라 속도 저하로 인하여 서비스를 원활히 이용할 수 없는 경우가 발생하게 된다. 하지만 웹 서비스를 제공하는 데에 있어 속도는 민감한 요소 중 하나로, 웹의 등장 이후부터 지금 까지 끊임없이 개선하고자 노력이 이루어지고 있는 핵심 분야 중 하나이다.

웹 서비스의 로딩 속도가 사용자에게 미치는 영향에 대한 예시로서 구글의 조사 결과를 들 수 있는데, 구글의 조사에 따르면 웹 콘텐츠의 로딩 속도가 3초를 넘길 경우 사용자 중 53% 이상이 해당 서비스를 이탈한다는 결과를 보여주었다. 뿐만 아니라 수익 측면에서 보더라도 로딩에 19초가 소요되는 서비스에 비해 5초가 소비되는 서비스의 경우 약 2배 이상의 수익률 차이를 나타내는 것을 확인할 수 있었다. 즉, 웹 서비스가 제공하는 데이터의 양에 상관없이 사용자에게 서비스가 로딩 되는 속도를 일정 수준으로 만족시킬 수 있어야 하며, 그렇지 못할 경우 사용자의 이탈에 따른 악영향이 발생한다는 것을 확인 할 수 있다 [5].

따라서 본 논문에서는 웹을 통해 제공되는 대용량의 데이터를 모든 사용자들이 효율적으로 제공할 수 있게 만들기 위하여 웹 브라우저와 별개로 동작하여 작업을 처리하는 서비스워커, 웹 브라우저 내에 데이터를 저장하여 사용하는 Cache API를 활용해 사용자에게 웹 서버로부터 데이터를 효율적으로 제공하는 방법에 대하여 논하고자 한다.

Ⅱ. 관련 기술 조사

본 절에서 실제 웹 어플리케이션의 로딩 속도 개선을 위하여 다양한 IT 기업들이 제공하고 있는 서비스에 대하여 분석을 진행하고자 한다. 분석을 진행하는 항목은 구글에서 제공하는 AMP (accelerated mobile pages)와 PWA (progressive web app) 총 2가지로 구성되어 있다.

2-1 Accelerated Mobile Pages(AMP)

Accelerated Mobile Pages(AMP)는 2015년 10월 구글이 발표한 모바일 기기에 콘텐츠를 효율적으로 제공하기 위한 최적화

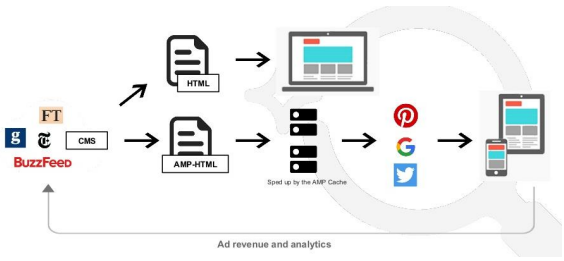

그림 1. APM 페이지 제공 방식
Fig. 1. How APM pages are provided.

오픈소스 기술이다. 사용자가 거의 즉시 로드할 수 있는 웹 페이지를 만들 수 있게 라이브러리 형태로 제공되고 있으며 Chrome, Firefox, Edge, Safari, Opera, UC Browser 및 다양한 외부 광고, 애널리틱스, CMS, 오디오/비디오 플랫폼에서 동작이 가능하다 [6]. 실제 TIMES의 경우 2018년 11월 AMP 기술을 자사에 적용함으로써 1.5배 이상의 매출이 증가하였으며, 웹 페이지 로딩 속도가 3.6배 이상 빨라졌다고 기술하고 있다. 또한 그림 2에서 볼 수 있듯이 AMP는 구글에서 지원하는 서비스로 AMP가 적용된 페이지가 검색 결과에서 좀 더 높은 위치에 배치되어 SEO에 영향을 주는 것을 확인 할 수 있으며, AMP 마크가 표시되어 해당 페이지가 AMP가 적용된 페이지임을 사용자에게 알려줄 수 있는 것을 확인할 수 있다. 이와 더불어 예시 사이트인 Campmor 기준으로 파일의 크기가 1300kb에서 90kb로 줄어 로딩 속도가 7.4초에서 0.62초로 크게 개선된 것을 볼 수 있다.

AMP가 적용된 페이지가 실제로 성능을 향상시키는 방식은 다음과 같다. 먼저, 외부 스크립트 사용 시 비동기 스크립트만을 허용한다. 자바스크립트는 매우 높은 성능을 가진 언어임과 동시에 웹 페이지의 거의 모든 요소를 수정할 수 있는 강력한 언어이다. 하지만 잘못 사용했을 경우 DOM의 생성을 방해하거나 웹 페이지가 렌더링 되는 것을 지연시키게 된다. 따라서 AMP는 모든 동기 스크립트를 차단하고, 비동기 형태의 사용만을 허용한다. 또한 모든 이미지, 동영상, 광고 등 리소스의 크기를 정적으로 지정한다. 이를 통해 리소스가 다운로드 되기 전에 AMP가 각 요소의 크기 및 위치를 확인할 수 있도록 하여 AMP가 리소스를 전부 다운로드 받지 않더라도 먼저 페이지의 레이아웃을 로드할 수 있도록 한다. 뿐만 아니라 모든 CSS를 인라인으로 처리하여 50kb 내외의 아주 작은 크기로 사용한다. 이 외에도 동기식 외부 자바스크립트 제거, 웹 최적화 글꼴 트리거 사용, 스타일 재계산 최소화, GPU가속 애니메이션만 실행, 리소스 로드 우선순위 지정, preconnect API를 이용한 페이지 즉시 로드 등을 사용하고 있다 [7].

AMP의 페이지의 구성은 크게 3가지 요소로 나눌 수 있는데 이는 각각 AMP HTML, AMP JS, AMP 캐시이다. AMP HTML은 AMP에서 사용되는 몇 가지 제한사항을 가진 HTML이다. AMP JS는 리소스 로드를 관리하고 AMP HTML 맞춤 태그를 제공하여 페이지 로딩 속도를 개선할 수 있는 라이브러리이다. AMP 캐시의 경우 구글에서 관리하는 모든 유효한 AMP 문서를 개제하기 위한 프록시 기반의 콘텐츠 전송 네트워크이다 [8].

그림 2는 실제 AMP 웹 페이지가 사용자에게 전달되는 방식과 일반적인 웹 페이지가 사용자에게 전달되는 방식을 표현한 것이다. 페이지를 사용자에게 빠르게 전송하기 위하여 모든 렌더링에 방해가 되는 요소들을 모두 차단하고, 구글에서 제공하는 캐시를 사용하여 페이지를 보관하고 있기 때문에 파일의 크기 및 사용자가 콘텐츠를 불러오는 속도가 개선되는 효과를 보여주고 있다. 하지만 구글에서 제공하는 서비스인 만큼 구글의 서비스에 종속되며, 페이지 내에 개인이 만든 스크립트를 삽입할 수 없고 구글의 광고 서비스에 치중되게 된다. 또한 사용자가 네트워크에 연결되지 않은 상태일 경우 별도의 안내 페이지나 콘텐츠를 제공할 수 없다.

2-2 PWA(progressive web app)

PWA는 2015년 구글 크롬 엔지니어 알렉스 러셀이 2015년 고안한 개념으로 웹 페이지가 단순 웹 페이지가 아니라 점진적으로 앱의 형태로 발전해 나가는 것을 의미하는 단어이다. 즉, 궁극적으로는 웹 페이지를 네이티브 앱 수준과 같은 사용자 경험을 포함하여 제공하는 것을 말한다. 앱과 같은 환경을 제공하고자하기 때문에 모바일 기기 혹은 데스크톱의 홈 화면에 아이콘 형태로 추가될 수 있으며, 네트워크가 느린 혹은 없는 환경에서도 콘텐츠를 제공하고, 푸시 알림을 통해 이벤트 처리를 사용자에게 인지시킬 수 있는 기능을 제공하고 있다 [9].

먼저 가능한 최소한의 사용자 인터페이스만을 로딩 하여 로딩에 소요되는 시간을 줄이고, 이를 캐싱 하여 두어 오프라인에서도 사용이 가능케 하는 App shell이다. App shell은 웹 페이지의 화면을 구성 하는 데에 사용되는 최소한의 CSS만을 포함하고 있어 매우 용량이 작다. 이는 웹 사이트의 이점을 모두 유지하면서 즉각적인 반응을 사용자에게 제공하여 네이티브 앱과 같은 서비스 환경을 느낄 수 있도록 한다. 또한 SW를 적용하여 App shell과 같은 정적인 파일들은 캐시에 저장하고, 화면을 렌더링 하는 데에 필요한 동적인 데이터에 대해서만 서버로부터 받아와 페이지 렌더링 속도를 높이고 오프라인 상태에서도 제한된 환경 내에서 서비스를 제공할 수 있도록 하였다.

상기 설명한 특징들을 통해 웹 페이지의 사용자가 마치 앱을 사용하는 것과 같은 만족도를 제공하였으며, 실제 2018년 11월 PWA 기술을 적용한 일본의 Nikkei 신문의 경우 2배 이상의 로딩 속도 개선이 이루어졌으며 2.3배 이상의 페이지 트래픽 증가, 일간 사용자 49% 증가, 페이지 별 조회 수 기존 대비 2배 증가 등 우수한 성과를 나타낼 정도로 성능 개선 측면에 있어 유의미한 기술임을 확인할 수 있다. 하지만 실제 테스트를 진행한 결과 한 번 캐싱된 파일에 대하여 캐시가 갱신되는데 까지 캐시 만료 시간으로 설정된 만큼의 시간이 소요되었다.

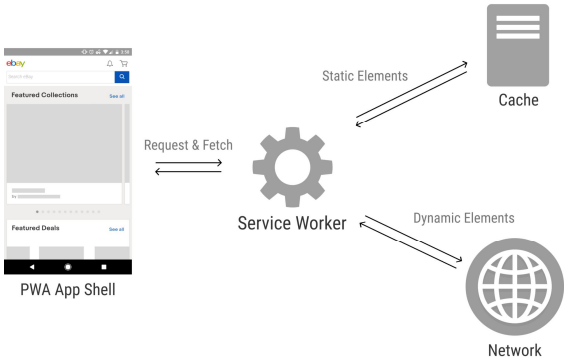

그림 2. PWA 동작 방식
Fig. 2. How PWA Works.

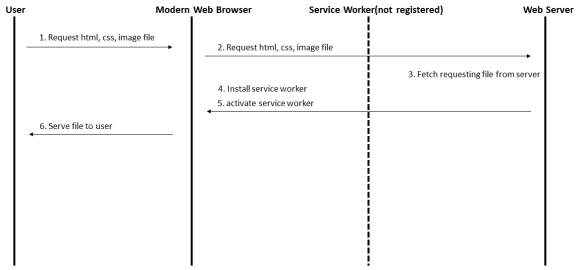

그림 3. 시스템 동작도 – 사용자가 처음 서비스에 접근한 경우
Fig. 3. System operation diagram - When the user first accesses the service.

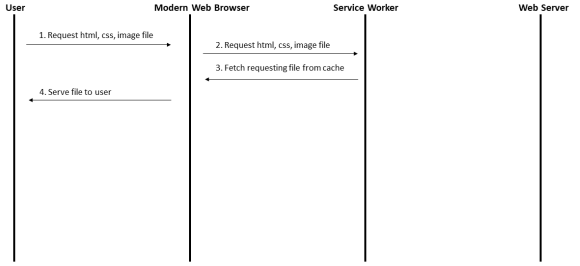

그림 4. 시스템 동작도 – 사용자가 요청한 파일이 캐시에 존재하는 경우
Fig. 4. System operation diagram - If the file requested by the user exists in the cache.

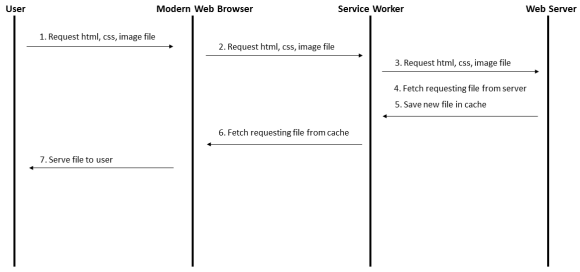

그림 5. 시스템 동작도 – 사용자가 요청한 파일이 캐시에 존재하지 않는 경우
Fig. 5. System operation diagram - If the file requested by the user does not exist in the cache.

이에 따라 사용자가 동일한 페이지에 한 번 접속한 뒤 짧은 시간 내에 페이지를 재방문 하였을 때, 만약 서버 내에 파일의 변경이 있었다면 이를 인지하지 못하고 이전에 캐싱된 파일을 사용자에게 다시 제공하였다. 따라서 항상 사용자가 보고 있는 화면이 최신 상태임을 증명할 수 없었다.

따라서 본 논문에서 제안하고자 하는 시스템은 AMP가 가지는 문제점인 서비스의 종속성 문제를 해결하고, PWA에서 발생하는 캐시의 갱신 문제를 해결하기 위하여 SW와 Cache API 기반의 시스템을 제안하고자 한다.

Ⅲ. 서비스워커 기반의 캐싱 시스템 설계

본 절에서 제안하고자 하는 시스템은 다음 그림 3, 4, 5와 같은 구조를 가진다. 사용자가 모바일이나 데스크톱의 브라우저를 이용하여 처음으로 구현된 웹 서버에 접속하게 될 경우 사용자들은 서버로부터 콘텐츠 데이터와 서비스워커를 다운로드 받게 된다. 첫 방문 시에는 모든 데이터가 순수 네트워크를 통해 다운로드 받아지게 된다. 서비스워커가 사용자의 브라우저에 다운로드 된 이후에는 자동으로 설치 과정이 진행되며, 사용자의 브라우저 내에 설치 및 등록이 진행된다. 설치 과정에서 서비스워커 내에 정의된 일부 캐시를 다운로드 받아 저장하게 되며, 설치가 완료된 이후에는 브라우저에서 발생하는 모든 요청을 서비스워커 내의 fetch 이벤트가 인지하여 대신 관리 및 처리하게 된다. 자세한 동작은 다음 그림 3과 같다.

서비스워커가 성공적으로 사용자의 브라우저 내에 설치되고, 활성화 된 이후에는 기존의 웹 페이지들이 동작하는 방식과 다르게 서비스된다. 기존의 웹 페이지들의 경우 사용자가 페이지를 새로 고침 하거나, 다른 페이지로 이동할 경우 서버로부터 해당하는 HTTP/HTTPS 통신을 이용하여 다시 다운로드 받은 뒤 사용자에게 제공하여 준다. 하지만 논문에서 제안하고자 하는 시스템의 경우 서비스워커를 등록하여 사용자가 발생시키는 다운로드 요청을 가로채 만약 요구하는 콘텐츠 혹은 파일이 이미 캐시 내에 저장되어 있는 경우 이를 다운로드 받는 것이 아니라 바로 캐시에서 꺼내어 해당 데이터를 사용자에게 전달한다. 만약 저장된 데이터가 아니라면 HTTP/HTTPS 통신을 통해 해당 데이터를 다운로드 받고, 캐시에 먼저 저장한 뒤 사용자에게 제공하게 된다. 이 과정에서 불필요하게 발생하는 데이터를 서버로부터 다운로드 받는 과정을 줄이고, 콘텐츠의 제공 속도를 향상시킨다. 자세한 동작은 다음 그림 4, 5와 같다

Ⅳ. 서비스워커 기반의 캐싱 시스템 구현 및 성능 평가

본 장에서는 실제 제안한 시스템을 실제 코드를 작성하여 구현하고, 이후 진행 될 실험의 구조를 정의하고자 한다. 본 장에서 구현하고자 하는 소스코드는 크게 3부분으로 나뉘며 전체 구성은 다음 표 1과 같다.

index.html 파일은 실제 사용자의 브라우저에서 나타날 콘텐츠를 정의한 파일이다. 다수의 이미지 태그(<img src="파일경로"></img>)로 구성되어 있으며, 각 이미지 파일은 용량이 4-5MB 내외로 비교적 큰 파일들을 사용하였다.

index.js 파일은 index.html 파일을 통하여 사용자에게 콘텐츠를 제공할 때 추가적으로 서비스워커를 등록하여 주기 위한 파일이다. 사용자가 index.html 파일에 접근하였을 때 사용자의 브라우저에 서비스워커를 다운로드 받아 설치, 등록 및 실행하는 코드를 정의하고 있다. 뿐만 아니라 본 논문에서 제안하고자 하는 사용자들에게 콘텐츠를 제공할 때 항상 서버로부터 최신 정보만을 유지하기 위하여 서비스워커에 별도로 정의

한 sync 이벤트를 발생시켜 캐시를 갱신하는 역할을 한다.

sw.js의 경우 실제 사용자의 브라우저에 등록되어 실행되는 서비스워커를 정의한 파일이다. 사용자의 브라우저에 서비스 워커가 다운로드 후 실행되는 'install' 이벤트, 사용자의 브라우저에서 설치 완료 후 실행 상태로 전환될 때 실행되는 'activate' 이벤트, 브라우저에서 발생하는 데이터 통신 이벤트에 반응하는 'fetch' 이벤트, 그리고 마지막으로 데이터 갱신을 위해 브라우저에서 발생하는 사용자 정의 이벤트에 반응하는 'sync' 이벤트에 대한 역할을 정의하고 있다.

실제 실험은 일반적인 브라우징 환경과 제안한 시스템이 적용된 브라우징 시스템을 비교하기 위하여 3개의 충분히 큰(10 MB 이상) 이미지 3개를 포함하는 페이지를 계속해서 중복 방문하였을 때의 결과를 확인하였다. 즉. 사용자가 동일한 페이지에 콘텐츠의 변경이 없는 상태에서 10회 접근하였을 때 평균 콘텐츠 로딩 소요 시간 비교를 진행했다.

실제 실험은 일반적인 네트워크 환경과 모바일 환경을 고려하기 위하여 총 2가지 환경에서 진행되었으며, 각 네트워크의 속도는 100 Mbps, 1.6 Mbps의 환경에서 진행되었다. 표 2는 1.6 Mbps환경에서 진행된 실험의 결과이며, 표 3은 100 Mpbs 환경에서 진행된 실험 결과이다.

실험 2, 3 모두 결과를 살펴보면 네트워크 속도가 빠른 환경에서는 큰 차이를 볼 수 없지만, 네트워크의 속도가 제한된 환경에서는 첫 방문 이후에 페이지를 다시 방문하였을 때 소요되는 시간에서 큰 차이를 보이는 것을 확인할 수 있다. 이는 논문에서 제안하고 있는 서비스가 사용자가 다시 통신을 요청하였을 때 기존에 미리 캐싱하고 있던 파일을 제공하여 다운로드 없이 콘텐츠를 조회할 수 있도록 서비스를 제공하고 있기 때문이다.

표 1. 시스템 파일별 기능 정의
Table 1. Defining Features by System File.

No	File Name	Description
1	index.html	- Content files provided to users - Providing a large number of large images to users
2	index.js	- Services to be applied to the index.html page - Define walker registration and event handling
3	sw.js	- define the service walker function to be applied to the index.html page

표 2. 실험 측정 결과 – Fast 3G 네트워크 환경
Table 2. Experimental Results - Fast 3G Network Environment.

ENV	1	2	3	4	5	6	7	8	9	10
1	59.1	58.7	59.3	59.9	58.6	58.6	59.4	59.8	59.2	57.8
2	59.3	1.21	1.23	1.22	1.29	1.23	1.26	1.22	1.29	1.23

표 3. 실험 측정 결과 – 100 Mbps 네트워크
Table 3. Experimental Results – 100 Mbps Network.

ENV	1	2	3	4	5	6	7	8	9	10
1	1.25	1.30	1.22	1.33	1.32	1.30	1.34	1.22	1.25	1.38
2	1.20	1.23	1.26	1.16	1.29	1.30	1.12	1.01	1.11	1.21

Ⅴ. 결 론

웹을 통해 사용자들이 공유하는 데이터의 종류는 점차 다양해지고 있다. 기존에는 단순 텍스트나 이미지, 영상에 불과했었지만 웹 어셈블리와 같은 신기술이 등장하면서 네이티브 환경에서 구동되던 소프트웨어를 포함하여 VR/AR 콘텐츠 등 다양한 요소들이 웹의 영역을 침범하기 시작하였다. 따라서 공유되는 데이터의 용량이 기하급수적으로 커졌다. 하지만 사용자들은 웹 페이지의 로딩이 3초가 넘어갈 경우 53%이상이 해당 페이지를 이탈할 정도로 속도에 민감하다. 즉, 웹을 통해 새로운 데이터들을 효율적으로 빠르게 사용자들에게 제공하기 위한 기술의 필요성이 지속적으로 대두되고 있었다.

따라서 본 논문에서는 사용자에게 데이터를 빠르고 효율적으로 제공하기 위하여 캐싱과 서비스워커라는 기술을 적용한 시스템을 제안하였다. 사용자들이 자주 사용하는 데이터의 경우 캐시에 저장하여 두고 요청 시 마다 다시 서버로부터 다운로드 받는 것이 아니라 저장해둔 캐시에서 데이터를 제공하여 시간을 단축시키는 것이다. 뿐만 아니라 사용자가 브라우저를 사용 중이지 않은 상태에서도 서버로부터 변경된 데이터를 다운로드 받아 백그라운드 환경에서 캐시를 갱신할 수 있는 기술을 적용하였다. 그리고 실험을 통해 기존의 캐싱을 사용하지 않는 방법보다 일반적인 네트워크 환경에서 약 5% 빠른 성능을 나타내었으며, 네트워크 속도가 제한된 환경(Fast 3G) 환경에서는 약 88%가량의 속도 개선을 나타낸 것을 확인할 수 있었다.

본 논문에서 제안하는 웹 콘텐츠의 속도 향상을 위한 서비스 워커 기반의 캐싱 기법을 활용하여 앞으로 다양해지고 방대해져가는 웹 콘텐츠들을 빠르고 편리하게 사용자에게 제공할 수 있어 웹을 통한 데이터 공유 및 웹 기반 데이터 처리 서비스의 활성화, 더욱이 나아가 전체 웹 생태계에 속도에 혁신을 미칠 수 있는 기술이 등장할 수 있기를 바라본다.

Acknowledgments

이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(No. 2018R1D1A3B07049722) 및 2017년도 한국기술교육대학교 교수 교육연구진흥과제 지원에 의하여 연구되었음

References

[1] B. L. Tim, J. Hendler, and O. Lassila. “The semantic web,” *Scientific American,* Vol. 284, No.5, pp. 34-43, May. 2001.

[2] E-country index. Household internet penetration rate and computer ownership rate [Internet]. Available: http://www.index.go.kr/potal/main/EachDtlPageDetail.do?idx_cd=1345.

[3] Mozila MDN. WebAssembly [Internet]. Available: https://developer.mozilla.org/ko/docs/WebAssembly.

[4] Akamai, 2017 state of the internet connectivity executive summary [Internet]. Available: https://www.akamai.com/kr/ko/multimedia/documents/state-of-the-internet/q1-2017-state-of-the-internet-connectivity-executive-summary.pdf.

[5] J. Wagner, Why performance matters [Internet]. Available: https://developers.google.com/web/fundamentals/performance/why-performance-matters/.

[6] APM Project. AMP project description [Internet]. Available: https://www.ampproject.org/ko/.

[7] R. H. Rapp, and J. Lewis, Accelerated mobile pages advertisement and cascading style sheet merging, 2017.

[8] R. Netravali, and J. Mickens, “Prophecy: accelerating mobile page loads using final-state write logs,” in *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation,* USENIX Association, WA, USA, pp. 249-266, 2018.

[9] Mozila MDN, Progressive web app [Internet]. Available: https://developer.mozilla.org/en-US/docs/Web/Apps/Progressive.

김 현 국 (Hyun-Gook Kim)
2014년 2월 ~ 2017년 8월: 한국기술교육대학교 컴퓨터공학과 (공학학사)
2017년 8월 ~ 현재: 한국기술교육대학교 컴퓨터공학과 석사과정
※관심분야 : 사물인터넷, 웹 어셈블리, 웹 표준

박 진 태 (Jin-Tae Park)
2005년 2월 ~ 2013년 8월: 한국기술교육대학교 컴퓨터공학과 (공학학사)
2013년 9월 ~ 2015년 8월: 한국기술교육대학교 컴퓨터공학과 (공학석사)
2015년 9월 ~ 현재 : 한국기술교육대학교 컴퓨터공학과 박사과정
※관심분야 : Web Assembly, Standardization of Web Technologies, Web Application Engineering

최 문 혁 (Moon-Hyuk Choi)
2014년 2월 ~ 현재: 한국기술교육대학교 컴퓨터공학과 학사과정
※관심분야 : 인공지능, IoT

문 일 영 (Il-Young Moon)
2000년 2월 : 한국항공대학교 항공통신정보공학과 졸업 (공학사)
2002년 2월: 한국항공대학교 대학원 항공통신정보공학부 졸업 (공학석사)
2005년 2월: 한국항공대학교 대학원 정보통신공학과 졸업 (공학박사)
2004년 ~ 2005년 : 한국정보문화진흥원 선임연구원
2005년 3월 ~ 현재 : 한국기술교육대학교 컴퓨터공학부 교수
※관심분야 : 무선 인터넷 응용, 무선 인터넷, 모바일 IP