웹킷 기반의 스마트 TV 애플리케이션 로딩 속도 향상 기법

이철희○, 원유집
한양대학교 컴퓨터 소프트웨어학과
lch6719@hanyang.ac.kr, yjwon@hanyang.ac.kr

An improvement of loading performance for Smart TV Application based on Webkit

Cheolhee Lee○, Youjup Won
Department of Computer and Software, Hanyang University

요 약

웹킷은 자원을 트리 형태의 자료구조인 돔트리와 렌더트리로 변환하여 웹페이지를 화면에 그려준다. 하지만 돔트리와 렌더트리는 프로그램 종료와 함께 메모리에서 제거되어 프로그램 재시작 시 변환 과정을 다시 수행하기 때문에 애플리케이션의 로딩 속도 저하가 발생한다. 본 논문에서는 트리 형태의 데이터에 영속성을 부여함으로써 데이터 변환 과정 없이 재사용 가능하도록 하는 Fast I/O 기법을 웹킷에 적용하여 스마트 TV의 애플리케이션 로딩 속도를 향상 시킨다. 성능 평가를 위해 웹킷 기반의 웹 브라우저에 Fast I/O 를 적용하여 돔트리와 렌더트리로 변환하는 과정을 제거하였고 돔트리와 렌더트리로 변환하는 과정까지의 대한 시간을 측정하였다. HDD와 SSD에서 평균적으로 약 2.9배, 7.9배의 성능 향상이 있었다.

1. 서 론

웹킷[1]은 가장 많이 사용되고 있는 웹 브라우저 엔진 오픈 소스로 사용자들이 자유롭게 사용해 볼 수 있으며 다양한 웹 콘텐츠를 사용자의 출력장치에 표현하고 사용자와 상호작용 하도록 웹 콘텐츠를 렌더링 하는 기능을 가지고 있다. 웹킷은 자원을 요청하고 요청 받은 자원을 파싱하여 돔트리와 렌더트리를 구성한다. 생성된 돔트리와 렌더트리에 데이터는 웹 페이지의 각 요소가 들어갈 위치, 크기, 색, 그리고 이미지 등 다양한 정보를 가지고 있으며 이 정보를 통해 웹 페이지를 화면을 그려준다.[2] 현재 웹킷은 웹 브라우저 및 웹 OS 등 다양하게 사용되고 있다.

웹킷 기반의 스마트 TV는 기존 TV에 CPU, 운영체제, 웹킷과 같은 웹 브라우저 엔진을 탑재하여 애플리케이션을 통해 VOD 서비스, 웹 서핑, 게임 등 다양한 기능을 수행한다. 웹킷 기반의 스마트 TV에서의 애플리케이션은 낸드 플래시에 저장된 XML, HTML, CSS, 자바스크립트, 이미지, 동영상 등이 웹 브라우저에서 실행되어 메모리에 적재하고 객체들로 구성된 트리 자료구조 형태인 돔트리와 렌더트리로 변환하는 과정을 수행한다. 하지만 애플리케이션이 종료 될 경우 돔트리와 렌더트리의 데이터는 메모리에서 제거가 되어 다시 애플리케이션이 실행될 때마다 낸드 플래시에 저장된 자원들을 메모리로 적재하고 트리 형태로 변환하는 과정을 수행한다. 이 과정은 매우 많은 시간을 요구하기 때문에 애플리케이션의 로딩 속도 저하에 원인이 된다.

본 논문에서는 낸드 플래시에 저장된 자원들이 변환된 돔트리와 렌더트리를 저장/복원 하는 방법으로 FAST I/O[3]를 사용하여 웹킷 기반 스마트 TV의 애플리케이션 로딩 속도가 향상될 수 있도록 한다. FAST I/O는 메모리 객체에 영속성을 부여하고 이를 재사용하는 기법으로 메모리 객체의 영속성을 부여하기 위해 파일을 생성하여 객체를 저장하고 프로그램이 재 시작했을 경우 mmap() 시스템 콜을 호출하여 고정된 위치의 프로세스 주소공간에 사상하여 데이터를 복원한다. 이 방법을 이용하여 프로그램 재시작 시 돔트리와 렌더트리를 고정된 위치의 프로세스 주소공간에 사상한다.

2. Fast I/O

본 논문에서 웹킷 기반의 스마트 TV 애플리케이션에 돔트리와 렌더트리를 저장하고 다시 재사용 가능하도록 하는 라이브러리인 Fast I/O를 소개한다. Fast I/O는 프로세스 주소공간에 파일을 사상하고 glibc[4] 기반의 동적 메모리 할당/해제 알고리즘을 이용하여 할당 된 메모리를 파일에 저장한다. 이때 저장 공간을 Object라고 한다. Object는 애플리케이션 재실행 시 mmap() 시스템 콜을 호출하여 파일에 저장된 자료구조의 데이터를 프로세스 주소공간에 사상함으로써 자료구조의 변환 과정을 제거한다.

Object가 프로세스 주소공간에 사상될 때 프로세스 주소공간 위치가 변경 되었다면 노드 간 연결을 나타내는 포인터의 값이 유효하지 않기 때문에 Object가 프로세스 주소 공간에 고정되어 있어야 한다. 노드간 포인터 유효성을 보장하기 위해 Object는 항상 동일한 주소에 사상이 되어야하나 그 주소공간이 사용 중일 경우 해당 Object를 프로세스 주소공간에 사상할 수 없다. Object가 프로세스 주소공간에 사상되기 전에 주소 공간을 예약함으로 해당 주소 공간을 사용하지 못하도록 하여 Object가 같은 프로세스 주소공간에 사상 되도록 한다.

메모리 노드 할당을 위한 저장 공간이 부족할 경우 Object를 확장해야 한다. 하지만 Object 아래의 주소 공간이 이미 다른 용도로 사용되고 있을 경우 Object는 연속적으로 확장할 수 없으므로 Object의 확장은 페이지

단위로 주소공간에 불연속 적으로 확장하도록 한다. 그러므로 Object의 주소는 연속성을 보장하지 않기 때문에 주소 공간을 관리하는 방법이 필요하다.

Object 파일에 데이터는 같은 프로세스 주소공간에 사상되어야 하므로 이를 위해 각 Object에 이름을 부여하고 네임 스페이스를 통해 이름을 관리한다. 이름을 가진 모든 Object를 관리하는 메타데이터는 해시테이블로 관리하여 해당 Object를 빠르게 삽입, 삭제, 탐색을 가능하도록 하였다. 해당 Object의 이름을 갖고 있는 메타데이터의 경우 Object가 할당된 주소 영역을 나타내는 자료구조의 리스트를 가지고 있기 때문에 해당 Object가 가진 프로세스 주소공간의 주소의 위치를 알 수 있다. 그러므로 재실행 시 고정된 프로세스 주소공간에 Object를 할당할 수 있다. 또한 이름을 갖고 있는 메타데이터에 자료 구조에 루트노드 또는 헤드노드의 포인터 주소를 저장하여 재실행 시 자료구조의 데이터를 사용할 수 있다.

3. 실험

Fast I/O를 웹킷 기반의 웹 브라우저에 적용하여 돔트리와 렌더트리가 생성되는 과정에 대한 성능 평가를 하였다. 실험은 AMD Phenom X4 925 Processor, 12GB DDR3 DRAM, 운영체제는 우분투 10.04 Linux Kernel 2.6.35에서 저장장치로는 하드디스크(WD BLUE 1TB) 7200RPM 과 SSD(Samsung SSD 840 Pro 256GB)를 사용하였고 각각의 경우에 대해 성능 측정을 하였다. 웹 브라우저는 웹킷 기반의 QT-4.6.4 웹브라우저를 사용하였다.

스마트 TV의 애플리케이션의 경우 저장 장치에 저장된 자원들을 웹 브라우저를 통해 애플리케이션을 실행한다. 그러므로 웹 브라우저에서 웹을 통해 자원들을 다운로드하는 과정이 아닌 웹 페이지의 자원들을 저장 장치에 저장하고 저장 장치에서 웹 페이지 자원들을 메모리에 적재하여 로컬에서 웹 페이지를 로딩 하도록 변경 하였다. 성능평가는 MainResource(HTML)와 SubResource(CSS, 자바스크립트, 이미지 등) 로딩 시간과 자원들을 돔트리와 렌더트리로 변환해주는 파싱 시간까지의 구간별 시간 그리고 Fast I/O를 통해 돔트리와 렌더트리를 메모리에 적재 하는데 수행되는 시간을 측정 하여 성능을 비교 하였다.

3.1 HDD에서 성능 비교

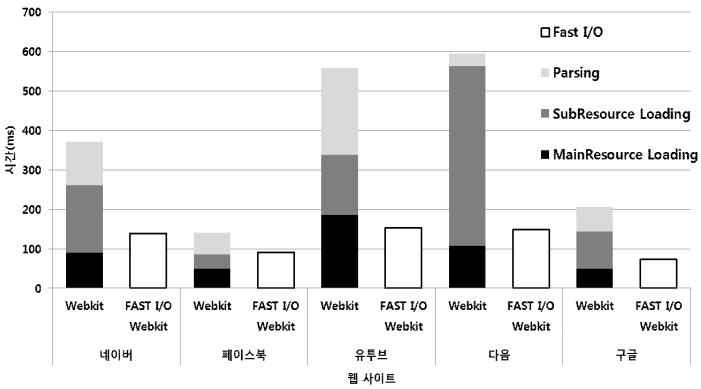

그림 1 HDD에서의 웹 페이지 구간 별 시간 비교

HDD에 네이버, 페이스북, 유투브, 다음, 구글의 웹 페이지 자원들을 저장한 후 웹 브라우저에서 로딩 속도를 측정 하였다. 그림 1은 HDD에서 웹 페이지에 대해 구간별 성능을 측정한 것이다. 기존 웹 브라우저의 경우 돔트리와 렌더트리를 생성할 때 3가지의 과정을 수행하기 때문에 많은 시간이 소요되는 반면 Fast I/O는 돔트리와 렌더트리를 파일에서 바로 프로세스 주소공간에 사상하기 때문에 기존 웹 브라우저보다 Fast I/O를 적용한 웹 브라우저에서 평균적으로 2.9배의 속도 향상을 확인 하였다.

3.2 SSD에서 성능 비교

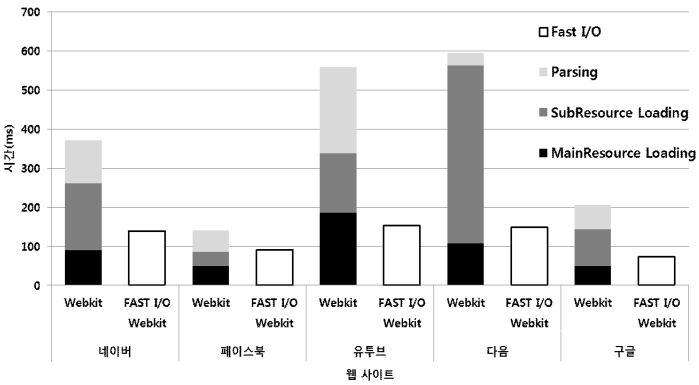

그림 2 SSD에서의 웹 페이지 구간별 시간 비교

SSD에서도 HDD 실험에서와 마찬가지로 SSD에 웹 페이지 자원들을 저장한 후 웹 페이지에 대해 구간별 성능을 측정한 것이다. SSD의 경우 Fast I/O를 적용한 웹 브라우저에서 평균적으로 약 7.9배의 성능 향상을 확인할 수 있었다.

4. 결론

웹킷 기반 웹 브라우저에 Fast I/O를 적용하여 돔트리와 렌더트리가 재사용하여 변환되는 과정을 제거할 수 있었다. 하지만 HDD와 SSD의 성능을 비교해본 결과 HDD보다 SSD의 성능 향상 폭이 큰 것을 확인할 수 있었다. 돔트리와 렌더트리를 파일에 저장하여 기존 자원들보다 파일 크기가 커지기 때문에 I/O 성능에 영향을 받는다. 스마트 TV는 비교적 I/O가 빠른 낸드 플래시를 장착하기 때문에 Fast I/O를 적용했을 경우 성능 향상이 있을 것이라 생각한다.

5. 참고 문헌

본 연구는 지식경제부 및 한국 산업기술평가관리원의 산업원천 기술개발사업(정보통신)의 일환으로 수행하였음.[No.10041608, 차세대 메모리 기반의 스마트 디바이스용 임베디드 시스템 소프트웨어]

5. 참고 문헌

[1] The WebKit Open Source Project, http://www.webkit.org/
[2] A. Grosskurth and M. W. Godfrey, “A reference architecture for web browsers,” 21st IEEE International Conference on Software Maintenance, pp. 661-664, September 2005.
[3] 정창훈, 황태호, 김규일, 원유집, “스마트 TV 용 고속 IO 기법”, 제 15회 한국 소프트웨어공학 학술대회 논문집, 2013년 1월
[4] The GNU C Library, http://www.gnu.org/software/libc/libc.html