선택적 적시 컴파일에 의한 모바일 환경의 클라이언트 측 자바스크립트 성능 개선[1]

이성원○, 유영호, 문수묵
서울대학교 전기공학부
swlee@altair.snu.ac.kr, yoo0ho@altair.snu.ac.kr, smoon@snu.ac.kr

Performance Enhancement of Client-side JavaScript with Selective Just-in-Time Compilation in Mobile Environment

Seong-Won Lee○ , Young-ho Yoo, Soo-Mook Moon
School of Electrical Engineering, Seoul National University

자바스크립트는 해석기 기반의 수행을 전제로 설계된 객체 지향 프로그래밍 언어이다[1]. 특히 웹브라우저에서 출력할 웹페이지의 문서 형식을 정의하는 객체인 Document Object Model (DOM) 을 제어하는 프로그램은 주로 자바스크립트로 작성하여 HTML 문서와 연동이 가능한 형태의 소스 코드로 서버에서 클라이언트 측으로 전달되는데 이와 같은 역할의 자바스크립트 프로그램을 특별히 클라이언트 측 자바스크립트로 분류한다[2]. 클라이언트 측 자바스크립트는 최근 대다수 웹페이지 문서에서 큰 비중을 차지하며 웹페이지 로딩 시간의 많은 부분을 점유하고 있다.

성능 개선을 위하여 자바스크립트 소스 코드를 기계어로 번역하는 적시 컴파일러를 적용할 경우 웹페이지 로딩 시 대량의 클라이언트 측 자바스크립트를 동적으로 컴파일하느라 많은 시간이 소비된다. 그렇지만 정작 생성된 기계어가 반복하여 실행되는 경우는 일부에 편중되어 있기 때문에[3] 단축되는 수행시간이 그 보다 많지 않아 오히려 해석기에 의해 수행한 경우보다 성능이 떨어지는 현상이 발생하기도 한다. 따라서 반복하여 수행되어 적시 컴파일에 의한 성능 이득을 볼 수 있는 부분만을 선택적으로 컴파일하고 나머지는 해석기에 의해 수행한다면 컴파일에 소비하는 시간을 크게 단축하면서도 생성된 기계어의 반복 수행에 의해 향상된 성능이 충분히 이를 보상함으로써 성능 향상을 기대할 수 있다[4]. 그러므로 우리는 한 안드로이드 모바일 환경의 자바스크립트 엔진에 이와 같은 선택적 적시 컴파일 방식을 적용함으로써 클라이언트 측 자바스크립트 성능을 가속화하고자 하였다.

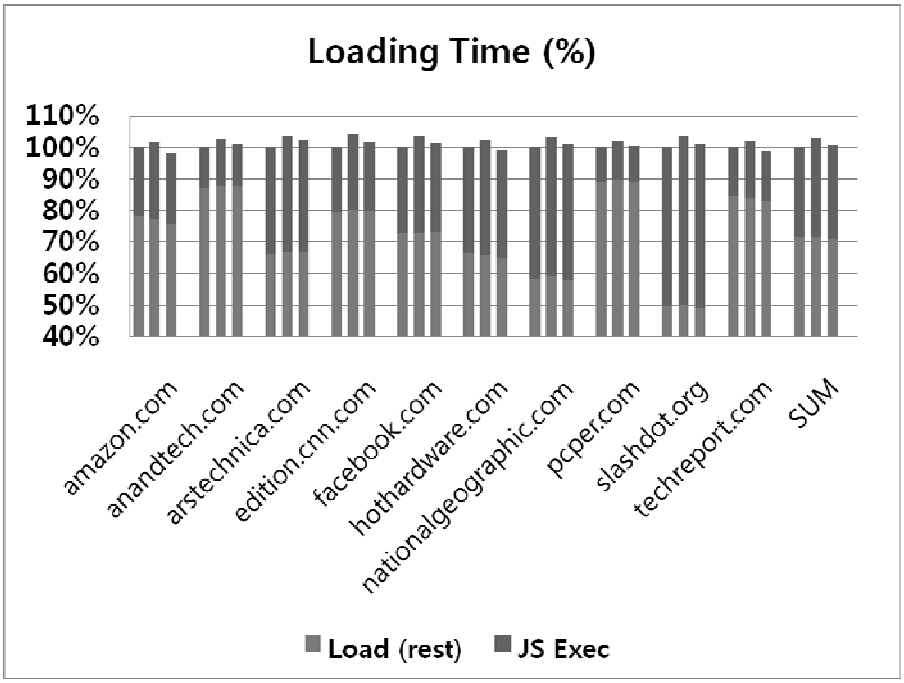

그림 1. 웹페이지 로딩 시간 비교

[1] 본 연구는 서울형산업 기술개발 지원사업(NT080546)과 삼성전자의 지원으로 수행되었음.

그림 1.은 선정된 10 개의 웹페이지에서 측정한 로딩 시간과 각각의 합을 해석기로 수행한 결과를 기준으로 나타낸 것으로 상단에 붉게 표시한 부분은 각각의 로딩 시간에서 클라이언트 측 자바스크립트의 수행시간을 의미한다. 각 웹페이지의 세 가지 결과는 순서대로 해석기로 수행한 것, 모든 함수를 적시 컴파일한 것, 그리고 선택적 적시 컴파일을 적용하여 가장 좋은 성능을 보였던 경우인 웹페이지 로딩 중 호출 회수가 30 회를 초과하면 해당 함수를 컴파일한 것을 표시하였다. 모든 함수를 적시 컴파일한 결과 웹사이트의 로딩 시간이 오히려 해석기로 수행했을 경우보다 3%가 증가하는 것을 볼 수 있고 이것은 그림에 나타난 것과 같이 클라이언트 측 자바스크립트의 수행시간이 그 만큼 더 길기 때문이다. 선택적 적시 컴파일을 적용한 경우 컴파일로 인해 소요되는 시간이 줄어들어 자바스크립트 수행시간이 감소한 것을 볼 수 있는데 이것은 로딩 시간이 해석기에 의해 수행한

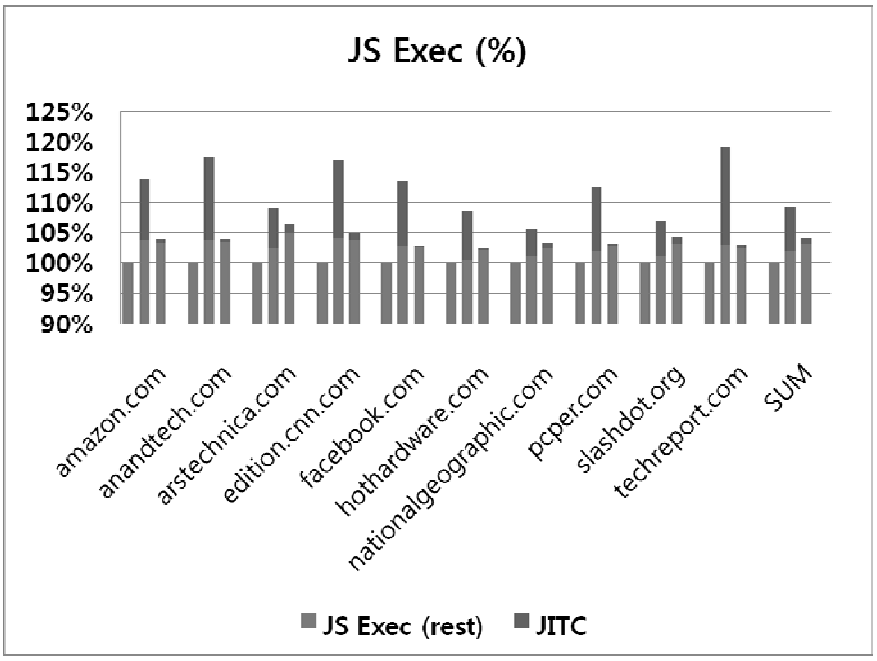

그림 2. 자바스크립트 수행시간 비교

경우보다 0.8%가 증가하는 데 그치는 것에 기여한다. 선택적 적시 컴파일에 의해 클라이언트 측 자바스크립트의 컴파일에 드는 비용이 크게 감소하였음을 가늠할 수 있다.

그림 2.는 각각의 클라이언트 측 자바스크립트 수행시간에서 적시 컴파일로 소모되는 실제 시간을 표시한 것으로 선택적 적시 컴파일에 의해 감소한 컴파일 시간의 영향을 살펴볼 수 있다. 대부분의 웹페이지에서 줄어든 컴파일 시간만큼 자바스크립트 수행시간이 감소하여 해석기에 의한 수행의 결과와 비교하면 모든 함수를 컴파일하였을 때 9.3%가 증가하던 것이 선택적 적시 컴파일에 의해 4%가 증가하는 것으로 개선되었다.

한편, arstechnica.com 등 몇 개의 웹페이지에서는 선택적 적시 컴파일로 컴파일 시간은 단축되었으나 도리어 나머지 수행시간이 소량 증가하는 현상을 볼 수 있는데 이것은 적시 컴파일로 이득을 볼 수 있는 함수가 컴파일되지 않고 해석기에 의해 수행된 결과라고 판단할 수 있다. 현재 적용된 호출 회수만을 기준으로 컴파일할 함수를 결정하는 방법으로는 반복문을 포함하여 수행시간은 길지만 상대적으로 호출 회수는 적은 함수를 선택할 수 없으므로 이를 고려한 새로운 기준을 추가한다면 선택적 적시 컴파일에 의한 성능 개선의 여지는 좀 더 있을 것이다.

그러나 모든 웹페이지의 자바스크립트 수행시간에서 적시 컴파일에 소모하는 시간을 제외한 나머지가 해석기에 의한 수행시간 보다 같거나 큰 것을 주지할 때, 웹페이지 로딩 시 수행되는 클라이언트 측 자바스크립트를 대상으로는 생성된 기계어의 성능이 해석기보다 뛰어나지 않다고 할 수 있다. 실제로 실험에 사용한 안드로이드 eclair 배포판에 포함된 WebKit 웹브라우징 엔진은 약 1년 전에 배포된 것으로 현재까지 오픈소스 진영에서 개선한 것에 비해 적시 컴파일러의 성능이 낮을 가능성이 높다. 따라서 최신 버전의 SFX 자바스크립트 엔진에 선택적 적시 컴파일을 적용한다면 클라이언트 측 자바스크립트의 성능이 본 논문에 의한 결과보다 더욱 향상되어 로딩 시간이 해석기에 의한 수행보다도 단축되는 것을 기대할 수 있을 것이다.

참고문헌

[1] ECMAScript Language Specification, 5th edition, 2009.
[2] David Flanagan, " JavaScript: The Definitive Guide, 5th Edition", O' Reilly Media, 2006.
[3] Paruj Ratanaworabhan, Benjamin Livshits, and Benjamin Zorn, " JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications", USENIX Conference on Web Application Development (WebApps) , 2010.
[4] M. Arnold, S. J. Fink, D. Grove, M. Hind, and P. F. Sweeney, " A Survey of Adaptive Optimization in Virtual Machines", Proceedings of the IEEE In Proceedings of the IEEE, Vol. 93, No. 2., pp. 449-466, 2005.