웹어셈블리 컴파일러 최적화 성능에 관한 연구

신채원 [1], 송수현 [2], 권동현 [3]
[1] 부산대학교 정보컴퓨터공학과 학부생
[2] 부산대학교정보융합공학과 박사과정
[3]*부산대학교 컴퓨터공학과 교수(교신저자)
sinchaewon@pusan.ac.kr, sshpnu@pusan.ac.kr, kwondh@pusan.ac.kr

A Study on Optimization Performance of WebAssembly Compilers

Chae-won Shin[1], Su-hyeon Song[2], Dong-hyun Kwon[3]*
[1]Dept. of Computer Science and Engineering, Pusan National University
[2]Dept of Information Convergence engineering, Pusan National University
[3]School of Computer Science and Engineering, Pusan National University (corresponding author)

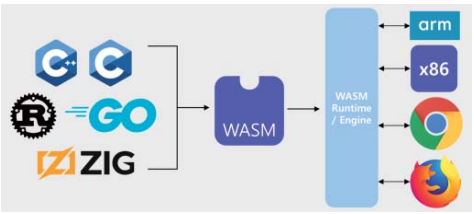
요 약

WebAssembly(WASM)는 웹브라우저용 바이트코드로, 다양한 언어로 작성한 코드를 손쉽게 한 번에 실행할 수 있고, 기존 고수준 언어를 사용하여 웹 애플리케이션을 개발할 수 있다. WASM 은 사용자와의 실시간 소통을 필요로 하는 웹용으로 개발되었기 때문에 성능이 중요한 요소로 꼽힌다. 이 논문에서는 대표적인 WASM 컴파일러인 emscripten 과 cheerp 에 대해 각각 생성된 코드의 성능을 측정하여 최적화 정도를 비교한다. 실험 결과 emscripten 의 최적화 수준이 더욱 높았으나, 두 컴파일러의 성능 간 상충 관계가 발견되었다.

1. 서론

WebAssembly(이하 WASM)은 높은 이식성, 성능, 메모리 효율성을 가진 바이트코드이다. WASM 은 C/C++, Rust, Go 등 다양한 언어를 컴파일하여 생성될 수 있으며, 다양한 런타임과 브라우저에서 실행된다.[1] 다양한 언어로 작성한 코드를 번거로운 절차 없이도 한 번에 실행할 수 있고, 웹에 특화된 프로그래밍 언어 없이도 웹용 프로그램을 작성할 수 있는 편리성 때문에 WASM 은 사용하는 애플리케이션이 점차 많아지고 있다.(그림 1) 이러한 흐름에 따라 chrome, safari, edge 등 웹 브라우저에서도 점차 WASM 을 지원한다.

사용자와의 실시간 소통을 필요로 하는 웹용으로 개발되었기 때문에 WASM 에 있어서 성능은 중요한 요소이다. 따라서 이 논문에서는 대표적인 WASM 컴파일러에 대해 추상화 정도를 비교하여, 컴파일러 간 성능을 비교해 보고자 한다.



<그림 1> WASM 의 생성 및 실행

2. Background

2.1. emscripten

Emscripten 은 C/C++ 코드를 WASM 으로 변환하는데 특화된 컴파일러이다. Emscripten 은 웹 어플리케이션 개발자들에게 익숙한 POSIX 호환성 레이어를 제공하여 기존의 C/C++ 코드를 쉽게 웹으로 이식할 수 있도록 돕는다.[2]

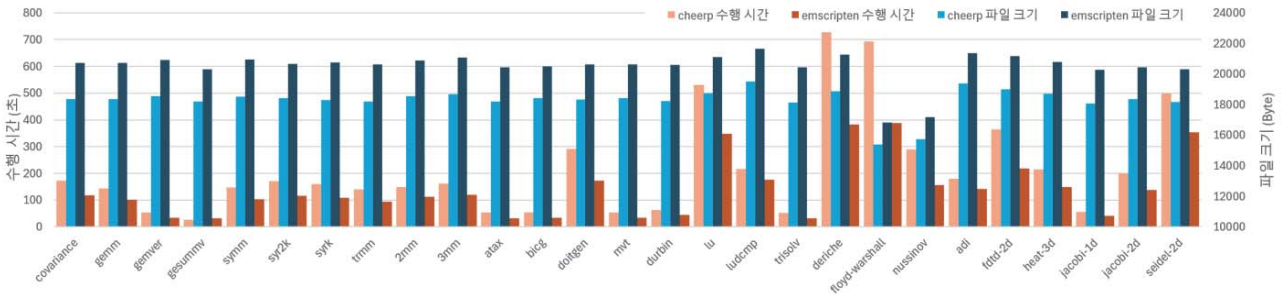
2.2. Cheerp

Cheerp 는 주로 C++ 코드를 WASM 으로 변환하는데 특화된 WASM 컴파일러이다. Cheerp 는 효율적인 코드 생성을 지원하며, 필요한 런타임 리소스를 사전에 업로드하고, 필요하지 않은 리소스는 지연시켜 웹 브라우저에서의 초기 로딩 시간을 최소화하는 전략을 사용한다.[3]

3. 실험 환경

본 연구에서는 WASM 컴파일러들 간의 성능을 비교하기 위하여, 벤치마크 프로그램인 Polybench 의 C 코드를 WASM 으로 컴파일하여 실험을 수행하였다. 컴파일된 코드는 x86 64 비트 Intel 서버 상에서 Google V8 엔진을 활용하여 실행되었다.

실험 대상 컴파일러로는 Emscripten 과 Cheerp 를 선정하였으며, 두 컴파일러 모두 최적화 수준 -O2 를 설정하여 실험을 진행했다. 특히, Emscripten 은 -s ALLOW_MEMORY_GROWTH=1 옵션을 통해 메모리



<그래프 1> WASM 컴파일러별 벤치마크 프로그램의 수행 시간과 생성 파일 크기

가 부족할 때 추가 메모리를 할당할 수 있도록 설정된 반면, Cheerp 는 이러한 메모리 동적 할당 기능을 지원하지 않는다. 이로 인해 벤치마크의 테스트셋 크기는 Cheerp 에서 지원 가능한 수준으로 제한되어, 두 컴파일러 간의 성능 비교가 가능하도록 일관되게 조정되었다.

4. 실험 결과

본 논문에서는 WASM 컴파일러인 Cheerp 와 Emscripten 을 사용하여 컴파일된 코드의 수행 시간과 파일 크기를 비교 분석하였다.

4.1. 수행 시간

그래프 1 에 따르면, 대부분의 경우에서 Emscripten 이 Cheerp 보다 확연히 빠르게 실행된다. 일부 예외는 'gesummv' 테스트에서 Emscripten 이 Cheerp 보다 느렸던 경우이다. 이를 제외한 다른 테스트에서 Cheerp 의 수행 시간은 Emscripten 대비 122.14%에서 190.75% 사이였으며, 평균적으로는 141.47%로 나타나 Cheerp 가 약 1.4 배 느린 것으로 분석되었다.

4.2. 파일 크기

그래프 1 에 따르면, 파일 크기 측면에서는 Emscripten 에 의해 생성된 WASM 파일이 Cheerp 에 의해 생성된 파일보다 일관적으로 큰 크기를 가진다. 생성된 WASM 파일의 크기 차이는 최소 1441 Byte 에서 최대 244 3byte 로 측정되었고, 평균은 2046 Byte 이다.

또한 각 컴파일러에 의해 생성된 파일 크기는 일정했다. Cheerp 와 Emscripten 각각에 대하여 생성된 파일 크기의 표준편차를 평균으로 나눈 값은 각각 0.046 과 0.051 로, C 코드에 대해 나타난 값인 0.21 에 비해 매우 작다. 이러한 결과는 C 코드의 내용보다 WASM 파일 생성 시 WASM 파일의 코드 섹션을 제외한 나머지 부분이 상대적으로 더 큰 비중을 차지하며, 파일 크기에 더 큰 영향을 미친다는 것을 시사한다.

4.3. 실험 결과

Cheerp 와 Emscripten 은 각각 다른 최적화 전략을 갖고 있다. Cheerp 는 파일 크기 최소화에 중점을 두어, 생성된 WASM 코드의 크기가 상대적으로 작다. 이러한 접근 방식은 특히 네트워크 대역폭이 제한적인 환경에서 유용할 수 있지만, 이로 인해 실행 성능이 다소 저하될 수 있다는 점에 주목해야 한다. 반면, Emscripten 은 보다 폭넓은 최적화 기능을 제공하여, 생성된 WASM 코드의 크기는 크지만 실행 성능을 상대적으로 높다.

따라서, 최적화 수준에서는 Emscripten 이 Cheerp 보다 높다고 평가될 수 있으나, 높은 추상화 수준이 항상 더 나은 결과를 보장하지는 않는다. WASM 모듈을 실행하는 환경 및 애플리케이션의 요구 사항에 따라 가장 적합한 컴파일러를 선택하는 것이 중요하다.

5. 결론

본 연구에서는 WASM 컴파일러인 Emscripten 과 Cheerp 를 대상으로 추상화 수준을 비교하였다. 실험 결과, Emscripten 이 Cheerp 보다 높은 추상화 수준을 가진다는 것을 확인할 수 있었다.

현재 실험에서는 Emscripten 과 Cheerp 의 기본적인 성능만을 비교하였으나, 각 컴파일러의 성능을 더 향상시키기 위한 최적화 기법을 탐구할 필요가 있다. 따라서 이후 Emscripten 에서 성능 향상을 위해 적용된 최적화 기법과, Cheerp 에서 파일 크기를 줄이기 위해 적용된 최적화 기법 및 기타 기법들을 연구하여 성능과 파일 크기 사이의 상충 관계를 고려한 최적화 기법에 대해 탐구하고자 한다.

Acknowledgement

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 융합보안핵심인재양성사업의 연구 결과로 수행되었음 (IITP-2024-2022-0-01201)

참고문헌

[1] LEI, Hanwen, et al. Put Your Memory in Order: Efficient Domain-based Memory Isolation for WASM Applications. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. Copenhagen, Denmark. 2023. p. 904-918.

[2] Emscripten Contributors. emscripten documentation. https://emscripten.org/index.html. 2024.04.17 에 확인함

[3] Learning Technologies. Cheerp documentation. https://labs.leaningtech.com/cheerp. 2024.04.17 에 확인함