WasmEdge와 다양한 런타임 및 프레임워크에서의 성능 비교 연구

홍석민[1], 이소영[1], 신용태[2]
[1]숭실대학교 컴퓨터학과
[2]숭실대학교 컴퓨터학부
ghdtjrals3@gmail.com
2soyeong10@gmail.com
shin@ssu.ac.kr

A Study on Performance Compare WasmEdge to different Runtimes, Frameworks

Seok-Min Hong[1],So-Yeoung Lee[1], [2]Yong-Tae Shin
[1]Dept. of Computer Engineering, SoongSil University
[2]School of Computing, SoongSil University

요 약

현대 소프트웨어는 다양한 서비스 구조가 사용되고 있다. 이러한 환경에서는 각 서비스의 요구사항을 충족시킬 수 있는 기술들이 필요하며, Wasm(WebAssembly)은 많은 서비스에서 요구하는 조건을 만족시킬 수 있는 장점을 가지고 있다. 이에 본 논문에서는 WasmEdge와 다양한 런타임 환경의 성능을 비교하기 위해 로직 실행 시간, HTTP 부하 테스트, 컨테이너 이미지 크기의 세 가지 지표를 분석한다. 결과는 로직 실행 시간에서 Wasm이 1.81초로 가장 빨랐고, 컨테이너 이미지 크기 역시 9.54MB로 가장 작았다. 마지막으로 HTTP 부하 테스트에서는 가장 빠른 트래픽 처리를 보여준 Spring Boot의 평균 초당 15076개보다 WasmEdge가 9239개로 트래픽 처리가 느렸지만, 로직 실행 속도와 컨테이너 이미지 크기가 작기 때문에 충분히 서버리스 컴퓨팅, 마이크로 서버, 엣지 컴퓨팅 분야에서 요구하는 조건을 만족시킬 수 있다.

1. 서론

현대 소프트웨어는 엣지 컴퓨팅, 클라우드, 서버리스 컴퓨팅, 마이크로 서비스 등 다양한 서비스 구조가 사용되고 있다. 이러한 환경에서는 각 서비스의 요구사항을 충족시킬 수 있는 기술들이 필요하며, Wasm(WebAssembly)은 많은 서비스에서 요구하는 조건을 만족시킬 수 있는 장점을 가지고 있다.[1][2] Wasm은 다양한 프로그래밍 언어로 작성된 동일한 로직의 코드를 하나의 Wasm 컴파일러를 통해 통합된 이진 코드 파일로 변환할 수 있다. 이는 개발자들에게 프로그래밍 언어와 실행 환경에 대한 더 큰 유연성을 제공하며, 코드의 재사용성을 증가시키고, 애플리케이션의 배포와 관리를 간소화한다. 더불어, Wasm 런타임 환경은 향상된 성능과 보안을 제공하며, 특히 마이크로서비스, 서버리스 컴퓨팅, 엣지 컴퓨팅과 같은 현대의 소프트웨어 개발과 잘 부합한다.[3] 그러나 WasmEdge가 가지고 있는 장점들에 대해 실제 다른 런타임 환경과의 성능 평가가 진행된 벤치마크가 없다. 이에 본 논문에서는 WasmEdge와 다양한 런타임 환경의 성능을 비교하기 위해 로직 실행 시간, HTTP 부하 테스트, 컨테이너 이미지 크기의 세 가지 지표를 분석한다.

본 논문의 구성은 다음과 같다. 2장에서는 Wasm과 WasmEdge의 특징과 구성요소를 알아본다. 3장에서는 본 논문에서 비교분석을 위해 구성한 환경 설명과 세 가지 지표에 대해 결과를 제시한다. 마지막으로 4장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련연구

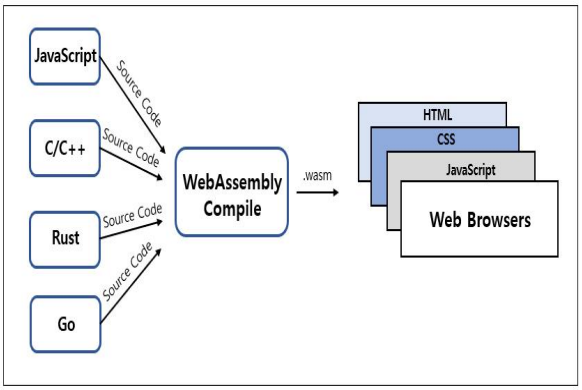
2.1 Wasm(WebAssembly)

Wasm은 이진 바이너리 형식으로 플랫폼에 독립적인 바이트 코드이다. Wasm의 장점은 다음과 같다.[4]

- **빠른 성능**: Wasm은 저수준 바이트코드로 컴파일되어, 네이티브 코드에 가까운 실행 속도를 제공한다. 이는 특히 계산 집약적인 작업에서 웹 애플리케이션의 성능을 크게 향상시킨다.

- **호환성**: 개발자들은 C, C++, Rust, Go와 같은 여러 네이티브 프로그래밍 언어를 사용하여 Wasm 모듈을 작성할 수 있다. 이는 기존 코드를 웹에서 재사용하거나, 선호하는 언어로 애플리케이션을 개발할 수 있는 유연성을 제공한다.
- **보안성**: Wasm은 격리된 샌드박스 환경에서 실행되므로, 애플리케이션의 보안성을 높인다. 이는 네이티브 코드를 실행하는 동안 메모리 안전성과 함께 사용자 시스템을 보호하는 데 도움을 준다.

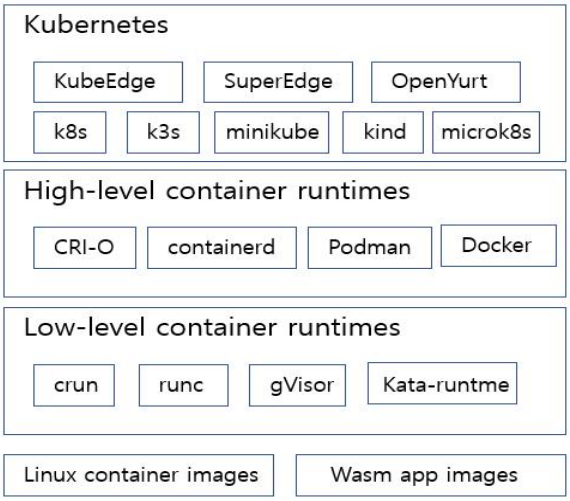
Wasm은 다양한 언어로 작성되었어도 해당 코드를 컴파일한 Wasm 파일과 런타임 환경만 있다면 어디서든지 사용할 수 있다. (그림 1)은 Wasm의 흐름도이다.



(그림 1) wasm 흐름도

2.2 WasmEdge

WasmEdge는 CNCF(Cloud Native Computing Foundation)에서 진행하는 프로젝트이다. 기존 Wasm 실행은 런타임 엔진이 들어가 있는 브라우저 환경에서 가능했다면, WasmEdge는 서버리스 앱, 마이크로서비스, IoT, 엣지 컴퓨팅을 지원하는 경량화된 고성능 Wasm 런타임 환경이다.[5] (그림 2)는 container ecosystem으로 기존 Kubernetes, Docker 및 CRI-O와 같은 컨테이너 도구를 활용할 수 있고, 동일하게 WasmEdge를 임베딩하여 어느 곳에서도 사용할 수 있다.



(그림 2) container ecosystem

WasmEdge는 AOT(Ahead Of Time) Compile을 지원한다. AOT란 바이트코드를 기계어로 바꾸는 방식이다. AOT 방식을 사용하면 실행 전에 전체 파일을 빌드해야 하기 때문에 빌드 시간이 늘어나는 단점이 있지만 속도가 빠르다는 장점이 있다. [표 1]은 AOT 컴파일과 관련하여 WasmEdge가 제안하는 주요 옵션 종류로 상황에 맞게 조절하면 높은 성능을 기대할 수 있다.

<표 1> WasmEdge AOT 옵션

종류	설명
disable-non-trap-float-to -int	부호 확장 연산자 활성여부
disable-multi-value	다중 값 활성 여부
disable-bulk-memory	대량 메모리 작업 활성 여부
disable-simd	고정 너비 SIMD 활성여부
enable-multi-memory	다중 메모리 활성 여부
enable-threads	스레드 사용 활성 여부

3. 성능평가

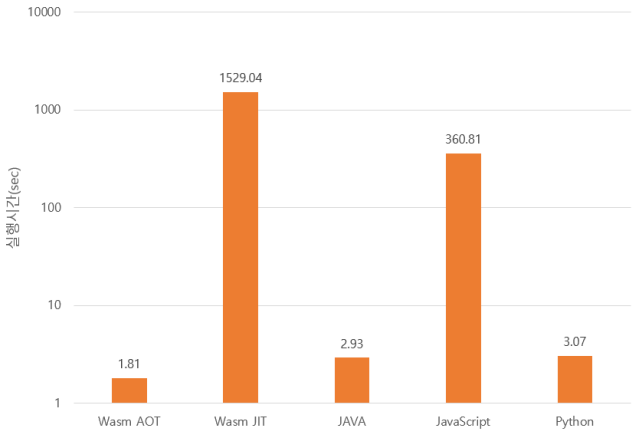
본 장에서는 WasmEdge 런타임 환경에서 Wasm이 다른 런타임 환경과 어떤 성능 차이를 가지는지 비교하기 위해 로직 실행 시간, HTTP 부하 테스트, 컨테이너 이미지 크기의 세 가지 지표 분석을 수행한다. <표 2>, <표 3>은 테스트를 위한 환경구성이다.

<표 2> 실험 환경

이름	내용
CPU	Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz 3.60 GHz
RAM	32GB
OS	Window

<표 3> 런타임 환경

대상	런타임	버전
WebAssebly	WasmEdge	1.0
Java	JVM	JDK 1.8
JavaScript	NodeJS	18.12.1
Python	Python Interpreter	3.12



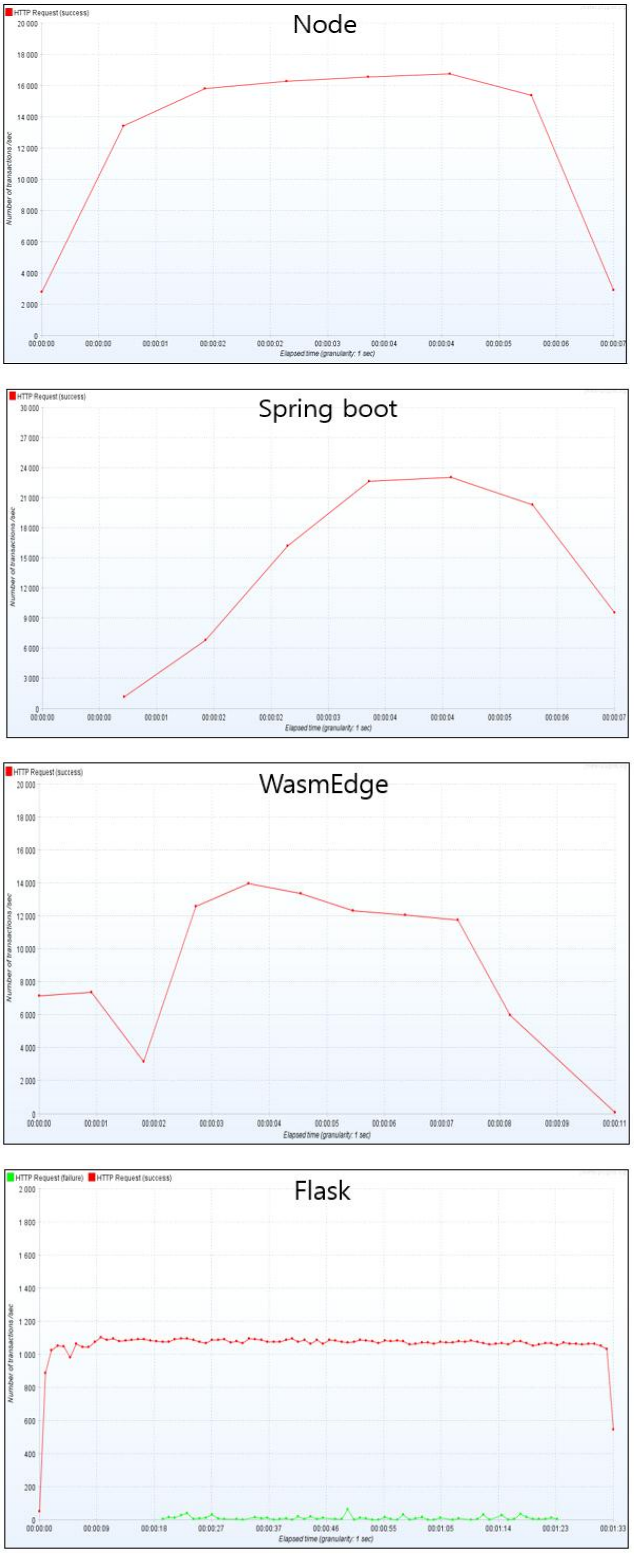
(그림 3) 로직 실행 시간

(그림 3)은 <표 3>의 각각의 런타임에서 암호화에 필요한 난수를 1억 개 생성하는 로직의 실행 시간을 비교한 그림이다. Wasm에서 AOT 컴파일 환경이 1.81초로 가장 빠른 시간을 기록하였고, Wasm의 JIT 컴파일 환경은 1529.04초로 가장 느린 성능을 보였다. 다른 런타임에서는 JAVA가 2.93초, Python이 3.07초, JavaScript가 360.81초가 소요되었다. 이 결과는 Wasm에서 AOT 컴파일 환경이 다른 런타임 환경보다 우수한 성능을 제공한다.

<표 4>컨테이너 이미지 크기

	Wasm Edge	Python Flask	Nodejs	Java tomcat
크기	9.54MB	138MB	94.1MB	456MB

WasmEdge는 Wasm의 런타임 환경으로, 브라우저에 종속되지 않고 별도의 서버에 임베딩하여 마이크로 서비스, 서버리스 컴퓨팅, 엣지 컴퓨팅으로 사용할 수 있다. <표 4> 각 환경별 컨테이너 이미지 크기이다. WasmEdge가 9.54MB로 가장 작다. WasmEdge를 사용할 경우 저장 공간을 덜 사용하기 때문에 배포 및 관리에 용이하다.



(그림 4) HTTP 부하 테스트

(그림 4)는 Node, Spring Boot, WasmEdge, Flask의 HTTP 부하 테스트 결과로 x축은 경과 시간, y축은 처리율이다.. Apache Jmeter를 통해 1000개의 쓰레드(유저)가 총 천만 개의 트래픽을 발생시켰다. 트래픽 처리율이 가장 좋은 환경은 Spring Boot로 평균 초당 15076개의 트래픽을 처리하였다. 다음으로 평균 초당 12504개 Node, 평균 초당 9239개의 WasmEdge, 마지막으로 평균 초당 1032개의 Flask가 가장 느린 처리율을 보여준다.

4. 결론

서버리스 컴퓨팅, 마이크로 서비스, 엣지 컴퓨팅 분야에서의 WasmEdge의 등장은 많은 관심이 집중되고 있다. 현재 대부분의 Wasmedge의 벤치마킹 비교 대상은 Wasm2c, Wasmer, Wasmtime과 같은 Wasm 런타임 환경으로 진행한다. 이에 본 논문에서는 WasmEdge가 마이크로 서비스, 서버리스 컴퓨팅, 엣지 컴퓨팅 등 다양한 분야에서의 장점을 가지고 있다는 것을 확인하기 위해 서로 다른 언어를 사용하는 환경과 성능 평가를 통해 비교를 진행한다. 평가 지표는 로직 실행 시간, HTTP 부하 테스트, 컨테이너 이미지 크기이다. 결과는 로직 실행 시간에서 Wasm이 1.81초로 가장 빨랐고, 컨테이너 이미지 크기 역시 9.54MB로 가장 작았다. 마지막으로 HTTP 부하 테스트에서는 가장 빠른 트래픽 처리를 보여준 Spring Boot의 평균 초당 15076개보다 WasmEdge가 9239개로 트래픽 처리가 느렸지만, 로직 실행 속도와 컨테이너 이미지 크기가 작기 때문에 충분히 서버리스 컴퓨팅, 마이크로 서버, 엣지 컴퓨팅 분야에서 요구하는 조건을 만족시킬 수 있다.

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터사업의 연구결과로 수행되었음" (IITP-2024-2020-0-01602)

참고문헌

[1] Khelifa Saif Eddine, Bagaa Miloud, Messaoud Ahmed Ouameur, Ksentini Adlen, “Case study of WebAssembly Runtimes for AI Applications on the Edge”, 2024 14th Global Information Infrasturcure and Networking Symposium, GIIS 2024

[2] López Escobar Juan José, Díaz-Redondo Rebeca P., Gil-Castiñeira Felipe, “Unleashing the power of decentralized serverless IoT dataflow architecture for the Cloud-to-Edge Continuum: a performance comparison“, Annales des Telecommunications/Annals of Telecommunications

[3] Kjorveziroski Vojdan, Filiposka Sonja, “WebAssembly as an Enabler for Next Generation Serverless Computing”, Journal of Grid Computing, Volume 21, Issue 3

[4] https://webassembly.org/

[5] https://wasmedge.org/