크로스 플랫폼 어플리케이션 개발을 위한 W3C WebAssembly와 CNCF WebAssembly의 차이점 비교 분석

*김하윤[1], 김원집[2], 이협건[3], 김영운[4]
한국폴리텍대학 서울강서캠퍼스 빅데이터과
let_hykim@naver.com[1], kbg5174@naver.com[2], hglee67@kopo.ac.kr[3],
luckkim@kopo.ac.kr[4]

Analysis of Difference between W3C WebAssembly and CNCF WebAssembly For Cross-Platform Application

*Hayoon Kim[1], Wonjib Kim[2], Hyeop Geon Lee[3], Young Woon Kim[4]
[1]Department of Big Data, Seoul Gangseo Campus
of Korea Polytechnics College.

요 약

크로스 플랫폼은 한 번의 개발로 다수의 플랫폼에서 동일하게 동작 가능한 어플리케이션을 개발하는 방법으로, 개발비용 절감과 유지보수에 유리하다. 시스템은 자발성, 자율성, 사회성, 반응성을 갖는 독립된 프로그램인 에이전트를 조합하여 구성되는 시스템으로 일반 사용자에게 편리하고 자연수러운 메타포를 제공한다. 그러나 개발자 측면에서는 에이전트 시스템에서 요구하는 각종 기능 및 제약규칙.

1. 서론

크로스 플랫폼은 한 번의 개발로 다수의 플랫폼에서 동일하게 동작 가능한 어플리케이션을 개발하는 방법으로, 개발비용의 절감과 유지보수에 유리하다. 이러한 특성으로 인해 LLVM, React Native, Qt 등 크로스 플랫폼 개념이 적용된 다양한 컴파일러, 아키텍처, 프레임워크들은 끊임없이 등장하고 있다. 그 중 WebAssembly(이하 WASM)는 기존의 웹 표준으로 정의된 Javascript의 문제점인 느린 실행 속도를 보완하며 조명받고 있다. [1]

WASM은 C, C++ 등의 프로그래밍 언어를 저수준의 바이트코드로 컴파일하여 웹 브라우저 상의 빠른 실행을 보장하는 런타임 환경이다. W3C 표준에 따라 실행되는 WASM 바이트코드는 네이티브에 가까운 속도를 가지며 3D 게임이나 가상/증강현실 등 고성능을 요구하는 작업에 사용된다. [2]

이후 서버 측 워크로드에서 WASM을 실행시키려는 시도로, CNCF 표준과 함께 WASM 런타임 환경 출시는 계속해서 이루어지고 있다.[3] 이러한 상황은 크로스 플랫폼 개발자들의 WASM 런타임 환경 선택에 혼란을 야기할 수 있다.

이에 본 논문은 바이트코드 기반 크로스플랫폼인 WASM을 두 가지 표준인 W3C 표준과 CNCF 표준으로 구분하여 특징 및 성능 비교 분석을 수행한다. 수행 결과는 크로스 플랫폼 개발자들에게 WASM 런타임 환경의 효율적 선택을 위한 가이드라인을 제시하는 것을 목표로 한다.

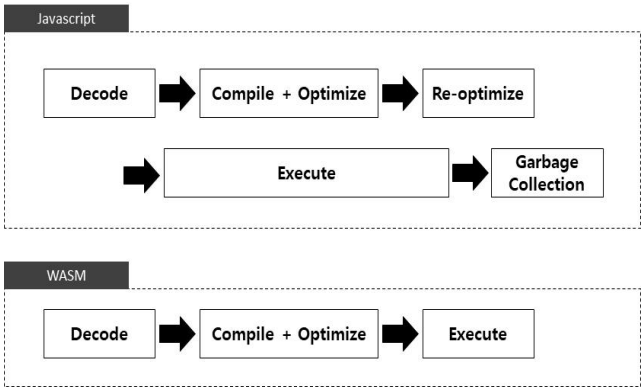
2. 관련 연구

본 장은 WASM 런타임의 두 가지 표준인 W3C WASM과 CNCF WASM를 설명한다.

2.1. W3C(World Wide Web Consortium)

W3C WASM은 웹 표준을 따르며 클라이언트 측에서 워크로드를 실행하는 WASM 런타임 표준으로, WASM의 최초 제안 형태이다. W3C WASM은 웹 브라우저 상에서 고수준 프로그래밍 언어를 실행시키는 것을 목적으로 해당 소스코드를 바이트코드(.wasm)와 Javascript(.js)로 컴파일시킨다. .wasm 파일은 W3C 표준에 의해 HTML에서 곧바로 실행되는 것이 불가능하므로, .js의 호출을 통해 실행된다. 호출된 WASM 바이트코드는 Javascript보다 빠

른 실행 속도를 갖는다. [그림 1]은 WASM 바이트코드와 Javascript의 실행 과정을 나타낸다.



[그림 1] Javascript와 WASM 바이트코드의 실행 과정

WASM 바이트코드는 이미 컴파일되어 실행에 최적화된 형태이므로 고수준 언어인 Javascript에 비해 빠르게 실행될 수 있다. 또한 WASM 바이트코드는 Javascript와 달리 Re-optimize 과정과 Garbage Collection을 수행하지 않아 더욱 빠르다.

W3C 표준에 따라 Javascript에 WASM 코드를 이식하는 방식은 고성능이 요구되는 웹 어플리케이션의 다양한 범위의 구현을 가능하게 한다. 단점은 과도한 Javascript 의존성이다. 운영 및 메모리 관리를 지원받지 못하는 WASM은 수동적인 실행만 가능하며 메모리 누수에 취약하다는 개선점이 있다.

대표적인 W3C WASM 런타임은 Emscripten, wasm-pack 등이 있다.

2.2. CNCF(Cloud Native Computing Foundation)

CNCF WASM은 클라우드 네이티브 및 엣지 네이티브 애플리케이션에 최적화된 WASM 활용에 중점을 둔 표준으로, 서버 측에서 워크로드를 수행한다.

WASM을 웹 브라우저 바깥에서 사용하를 시도로 개발된 WASI(WebAssembly System Interface)는 WASM의 능동적인 시스템 자원 접근을 지원한다. CNCF WASM은 WASI를 통해 다양한 운영체제 상에서 실행될 수 있으며, 활용하여 클라우드 네이티브, 엣지 컴퓨팅 등에 최적화된 런타임을 제공한다. CNCF WASM의 샌드박스 환경 지원은 호스트 시스템과 격리된 실행을 보장하여 쿠버네티스와 같은 오케스트레이션 시스템과의 연계도 가능하게 한다.

CNCF WASM은 클라우드 네이티브 및 엣지 컴퓨팅 애플리케이션을 위한 런타임을 제공하면서 안전하고 경량화된 컨테이너를 지원한다. 제공하는 최적화 컨테이너를 통해 개발자는 네이티브 기능을 호스트 애플리케이션이나 분산 컴퓨팅 프레임워크에 안전하게 포함할 수 있다. 대표적인 CNCF WASM 런타임은 WasmEdge, Wasmtime, Krustlet, WAMR 등이 있다.

3. 분석 결과

본 장에서는 2장에서 설명한 W3C WASM 런타임과 CNCF WASM 런타임의 특징 및 차이점 분석 결과를 제시한다. 실행 속도 비교는 각 양 측 런타임 환경(wasm-pack, WasmEdge)에서 Rust 기반 행렬 곱 연산 속도을 통해 측정하였다. [표 1]은 W3C WASM과 CNC WASM의 분석 표이다.

[표 1] W3C WASM과 CNCF WASM 비교

구분	W3C WebAssembly	CNCF WebAssembly
주요 목적	• 웹 브라우저 고성능 실행	• 서버 환경의 고성능 실행
사용 환경	• 웹 브라우저	• 서버 환경
주요 기술	• 웹 API와의 통합 • Javascript와의 상호작용	• 클라우드 네이티브 통합 • WASI
실행 속도 비교	• 느림	• 빠름
영향 받는 개발자	• 웹 개발자	• 시스템 관리자 • 클라우드 개발자

4. 결론

본 논문은 바이트코드 기반의 크로스 플랫폼인 WASM을 두 가지 표준인 W3C 표준과 CNCF 표준으로 구분하여 특징 및 성능 비교 분석을 수행한다. 수행 결과는 크로스 플랫폼 개발자들에게 WASM 런타임 환경의 효율적 선택을 위한 가이드라인을 제시하는 것을 목표로 한다.

참고문헌

[1] Roger S. Pressman "Software Engineering A Practltiners' Approach" 3rd Ed. 1, 2023

[2] https://webassembly.org/

[3] https://landscape.cncf.io/

[4] https://wasmedge.org