Exploring WebAssembly for versatile plugin systems through the example of a text editor

Florian Hartung

Problem definition

WebAssembly is a bytecode format, initially released in 2017 as a compilation target for web-based applications. However, the utilisation of WebAssembly has since expanded beyond the web. It offers advantages such as cross-language interoperability and sandboxed execution, which enable emerging use cases in vastly different areas such as cloud computing, safety-critial systems and blockchains. Another promising application for WebAssembly is plugin systems, for which only limited research and a few projects exists.

Plugin systems are software components that allow plugins to extend or modify the functionality of some host application. Most common plugin systems using languages such as JavaScript, Lua or Python have some issues regarding performance or safety while their plugins might be non-portable and locked to their respective programming language. This work aims to explore how WebAssembly can be leveraged to create fast and safe plugin systems for portable and interoperable plugins. As a test case a plugin system will be developed for a text editor. Text editors are suitable as their broad spectrum of users usually demand a high degree of customization. A WebAssembly plugin system provides this customization by enabling features to be implemented by third-party developers, while still guaranteeing a safe, sandboxed plugin execution.

Objectives

- Analyze requirements for plugin systems from real world projects
- Analyze the pros and cons of WebAssembly (and its related technologies) in comparison to traditional architectures of plugin systems
  - Performance, Safety, Portability of the plugin system
  - Language-interoperability for plugin development
- Gain insight into designing and implementing a WebAssembly plugin system for a text editor as a proof of concept

Approach/ Table of contents (WIP)

- Introduction
- Fundamentals
  - WebAssembly
  - Plugin systems
  - Rust
- Plugin system requirements
- Related work (research and projects)

- WebAssembly for plugin systems
  - Overview
  - Choosing a plugin API (native/wat/wasi)
  - Safety
  - Performance
  - Summary
- Implementing a plugin system for a text editor
  - Requirements
  - Design
  - Implementation
  - Example plugin development (preferable in different languages to demonstrate interop-erability)
  - Verification and validation
- Discussion
- Conclusion