Web Assembly

Zusammenfassung

WebAssembly ist eine virtuelle Maschine, welche systemnahe Instruktionen zur effizienten Programmausführung und ein Bytecode-Format mit platzsparender Programmdarstellung bietet. Konzipiert wurde sie mit der Motivation ein Übersetzungsziel im Internet für Programmiersprachen darzustellen. Diese Arbeit beschäftigt sich mit dem aktuellen Stand von WebAssembly und der Eingliederung in andere Internetsprachen.

I. Einleitung

Das Internet hat sich zu einem System entwickelt in dem Nutzer sich mithilfe von verschiedensten Endgeräten durch dynamisch erzeugte und interaktive Internetseiten Informationen abrufen können, sowie Anwendungen ausführen können. Hierfür benötigt werden auf all diesen Endgeräten einheitliche Programmier- und Beschreibungssprachen mit denen Internetseiten schnell übertragen und aufgebaut werden können, sowie schnell und sicher ausgeführt werden können.

Besonders ressourcenintensive Anwendung, wie Audio- und Videostreaming oder Spiele benötigen ein geeignetes Übersetzungsziel im Internet[3], da bereits eine große Menge an geeignetem Quelltext für Offline-Plattformen vorhanden ist.

Diese Arbeit betrachtet WebAssembly als eine systemnahe und platzsparende Sprache bestehend aus einem Befehlssatz für eine virtuelle Maschine, die sich für eine Anwendung im Internet eignet und als Übersetzungsziel für statische Programmiersprachen dient. WebAssembly ist bereits in verschiedenen populären Webbrowsern ausführbar.

II. Internetsprachen

Nahezu alle Internetseiten benutzen HTML(Hypertext Markup Language) und CSS(Cascading Stylesheets) um statische Inhalte darzustellen. Jedoch hat sich zurzeit auf Anwenderseite nur JavaScript als Programmiersprache für dynamische und interaktive Elemente im Internet durchgesetzt. Historisch gab es einige unterschiedliche, teilweise weit verbreitete Ansätze, wie Java Applets, Flash Player oder ActiveX welche sich aber durch unterschiedliche Beschränkungen nicht dauerhaft durchsetzen konnten.[3]

Besonders durch die Vielzahl von unterschiedliche Hardwarearchitekturen, die von ein und derselben Internetseite unterstützt werden sollen kommt es häufig zu Einschränkungen.[3]

Die Verbreitung und Geschwindigkeit von JavaScript hat sich schließlich soweit entwickelt, dass es als Übersetzungsziel für andere Programmiersprachen dient. So lassen sich mit Hilfe von Emscripten Sprachen, die zu LLVM-Code(siehe III.) übersetzt werden, zu asm.js übersetzten. Asm.js ist Eine klar definierten Teilmenge von JavaScript, welche Assemblersprachen ähnelt. Asm.js kann somit direkt von JavaScript-Interpretern ausgeführt werden und besitzt außerdem verbesserte Unterstützung durch einige Webbrowser.[3]

Jedoch erlebt asm.js auch Einschränkungen durch die Bildung aus JavaScript. Das Hinzufügen von Erweiterungen, wie Datentypen zu asm.js benötigt zuerst eine Erweiterung von JavaScript. Außerdem hat JavaScript-Code oft zu hohe Dateigrößen für eine effiziente Übertragung im Internet, da der Code als Textformat übertragen wird.[3]

Zur Lösung dieses Problems bietet sich an eine native assemblerähnliche Sprache zu verwenden, die in einem Bytecode-Format dargestellt werden kann. Dazu ist es nicht abwe-

gig, bereits etablierte Sprachen zu verwenden. Welche auf einer virtuellen Maschine(siehe III.) abgebildet werden um Portabilität und Sicherheit zu erreichen.[3] Dazu wurden beispielsweise der LLVM-Bytecode, teilweise oder komplett, in Betracht gezogen jedoch ist dieser vom Zielsystem abhängig und ändert sich durch häufige Updates. Somit ist dieser nicht für die Plattformvielfalt des Internets geeignet. Auch die Java Virtual Machine(JVM) ist ungeeignet, da der Befehlssatz der JVM für einige Laufzeitoptimierungen ungeeignet ist.[3]

Auf der Webserverseite existieren noch viele weiter Möglichkeiten um dynamische Internetseiten zu ermöglichen, worauf in dieser Arbeit jedoch nicht weiter eingegangen wird.

III. Virtuelle Maschinen

Im Gegensatz zu einer real existierenden Maschine welche Programme mithilfe eines Mikroprozessors ausführt, beschreibt eine virtuelle Maschine(VM) eine Umgebung, die zwar einem realen Rechner ähnlich ist, aber nicht ohne weitere Hilfsmittel Programme ausführen kann. [1, S.4]

Der Vorteil solch einer virtuellen Maschine liegt vor allem in der Portabilität von Programmen, die vorerst als Zwischenschritt aus einer höheren Programmiersprache in die Maschinensprache einer virtuellen Maschine übersetzt werden(Abbildung 1). Der Befehlssatz dieser virtuellen Maschine ähnelt dem von tatsächlichen Mikroprozessoren, ist aber oft als Übersetzungsziel an bestimmte Programmiersprachen angepasst. Dadurch ermöglicht wird ein einfacher Übergang von Quellcode zu Code für die virtuelle Maschine und anschließend zu Maschinensprache des Zielsystems. Solange ein Interpreter oder Übersetzter des virtuellen Befehlssatzes zu dem einer realen Maschine existiert, kann jedes Programm der virtuellen Maschine auf der realen Maschine ausgeführt werden.[1, S. 4,5]

Oftmals werden virtuelle Maschinen notwendig, wenn ein Übersetzter mehrere Zielsysteme unterstützen soll. Dazu wird zuerst der Quellcode der zu übersetzenden Programmiersprache in den Maschinencode einer virtuellen Maschine überführt, um systemnahe Optimierungen durchführen zu können. Anschließend wird der optimierte Maschinencode der virtuellen Maschine zu Maschinencode der realen Maschine übersetzt. Dadurch wird das entwickeln von mehreren maschinenabhängigen Optimierern eingespart.[4]

Zudem gibt es ein höheres Maß an Sicherheit durch Abgrenzung von der realen Maschine bei der Interpretation von Code für eine virtuelle Maschine.[1, S.5]

Verbreitete für virtuelle Maschinen sind die Java Virtual Machine(JVM) und die Low Level Virtual Machine(LLVM).[3]

IV. WebAssembly

WebAssembly ist im Kern eine durch einen virtuellen Befehlssatz definierte virtuelle Maschine.[2, S.2] Sie soll als Übersetzungsziel von verschiedenen Programmiersprachen, im Vordergrund stehen C und C++, eingesetzt werden. Vor allem um diese Sprachen auf Internetseiten in Webbrowsern ausführen zu können.[3]

Durch die Anwendung im Internet ergeben sich folgende Ziele:

- Sicherheit
- Geschwindigkeit
- Portabilität
- Kompaktheit

[3]

Diese Ziele werden einerseits durch die Verwendung einer virtuellen Maschine, andererseits aber durch ein geeigneten Befehlssatz erreicht.

Die virtuelle Maschine macht die Programme unabhängig von der Architektur der realen Maschine, solange ein passender Interpreter vorhanden ist. Außerdem sorgt die Abkapselung der Daten der virtuellen Maschine für hohe Sicherheit.[1, S.4]

Die Ausführungsgeschwindigkeit wird durch einen für Anwendungen und mögliche Zielarchitekturen der Interpretation geeignet gewählten Befehlssatz erreicht.[3]

WebAssembly besitzt zwei verschiedene Darstellungsformate: Textformat zur guten Lesbarkeit und Bytecode-Format zur hohen Kompaktheit. Im Textformat wird das Programm in einer von Menschen lesbaren Form dargestellt,

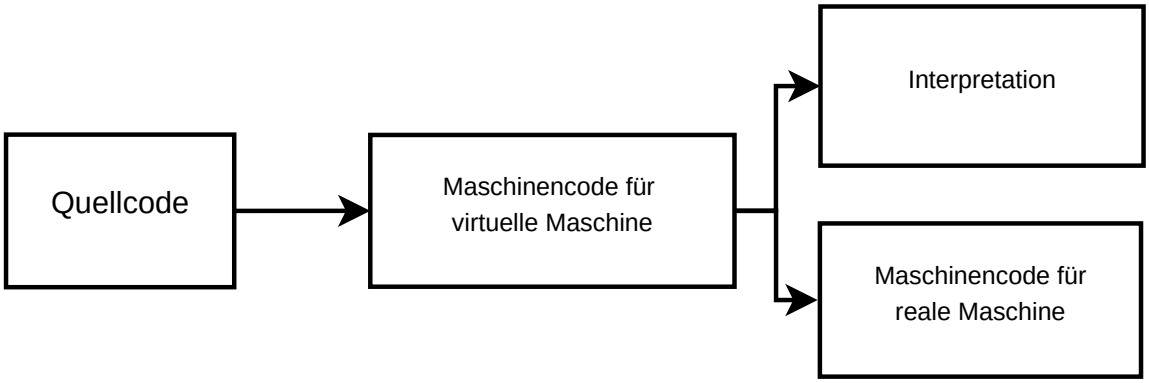

Abbildung 1: Funktionsweise einer virtuellen Maschine

um Code zu schreiben oder zu bearbeiten, aber auch um möglicherweise böswilligen Code einsehen zu können. Das Bytecode-Format wird zur Übertragung und Ausführung verwendet- Es ist so aufgebaut, dass möglichst kleine Dateigrößen möglich sind während eine effiziente Verarbeitung(Dekodierung, Validierung, Ausführung) erreicht wird. WebAssembly Programme sind in Modulen aufgebaut, die unabhängig voneinander in einer Anwendung eingebunden werden können. [2]

Jedoch ist WebAssembly keineswegs an das Internet gebunden, sondern eignet sich auch zur Interpretation durch eine alleinstehende virtuelle Maschine auf dem Zielsystem.[2, S. 1] Üblicherweise soll WebAssembly eingebettet in einer Umgebung (beispielsweise Webbrowser) implementiert werden. Dazu hängt es von der Umgebung ab nicht benötigte Funktionalitäten, wie Verarbeiten von WebAssembly in Textformat, nicht zu implementieren.[2, S. 121]

Dadurch, dass WebAssembly gemeinsam von Entwicklern verschiedener populärer Webbrowser(Chrome, Firefox, Safari, Edge) konzipiert wurde, ist es bereits auf verschiedenen Plattformen ausführbar.[3]

V. Zusammenfassung

WebAssembly erfüllt das Ziel der leistungsstarken Ausführung von ressourcenintensiven Programmen im Internet. Es wird jedoch einige Zeit brauchen, bis Entwickler mit WebAssembly vertraut werden und es in das bestehende Internet eingegliedert wird.

Momentan ist WebAssembly zudem ungeeignet für Programmiersprachen mit automatischer Speicherbereinigung, jedoch könnte für die Anwendung auf Internetseiten WebAssembly mit der bereits vorhandenen Speicherbereinigung der Webbrowser verknüpft werden.

Außerdem ist zu beobachten, inwiefern WebAssembly als Übersetzungsziel außerhalb des Internets Verbreitung findet, da durch vorhandene virtuelle Maschinen wie JVM oder LLVM kaum bedarf vorhanden ist.

Literatur

[1] *Übersetzerbau Virtuelle Maschinen*. eXamen.press. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[2] WebAssembly Specification, 2018. URL `https://webassembly.github.io/spec/core/_download/WebAssembly.pdf`, zuletzt aufgerufen: 25.05.2018.

[3] A. Haas, A. Rossberg, D. L. Schuff, B. L. Titzer, M. Holman, D. Gohman, L. Wagner, A. Zakai & J. Bastien. Bringing the Web Up to Speed with WebAssembly. *SIGPLAN Not.*, 52(6):185–200, 2017. URL `http://doi.acm.org/10.1145/3140587.3062363`.

[4] C. Lattner. LLVM. URL `http://www.aosabook.org/en/llvm.html`, zuletzt aufgerufen: 27.05.2018.