Hochschule Karlsruhe
University of
Applied Sciences

HKA

Bun vs. Node.js

Seminararbeit von
Ansgar Lichter

an der Fakultät für Informatik und Wirtschaftsinformatik

Universität:	Hochschule Karlsruhe
Studiengang:	Informatik
Professor:	Prof. Dr.-Ing. Vogelsang
Bearbeitungszeitraum:	01.10.2023 - 04.12.2023

Eidesstattliche Erklärung

Ich versichere, dass ich diese Masterthesis selbständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben, sowie wörtliche und sinngemäße Zitate gekennzeichnet habe.

(Ort, Datum)	(Ansgar Lichter)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

PoC Proof of Concept

Kapitel 1

Einleitung

1.1 Motivation

JavaScript ist eine Programmiersprache, die vor allem im Kontext der Web-Entwicklung verwendet wird. Aktuell erfreut sich JavaScript großer Beliebtheit. In einer Umfrage an Entwickler von Stack Overflow wurden mehr als 89.000 Entwickler befragt. JavaScript ist zum 11. Jahr in Folge die am häufigsten verwendete Programmiersprache. Mehr als 63% der befragten Entwickler haben JavaScript als beliebteste Technologie gewählt. Bei den professionellen Entwicklern ist der Anteil mit mehr als 65% sogar noch höher. Außerdem ist TypeScript, eine stark typisierte Programmiersprache, die auf JavaScript aufbaut, unter den Teilnehmer auch beliebt. Ca. 39% aller Entwickler und ca. 44% der professionellen Entwickler verwenden auch TypeScript. Damit ist TypeScript die 4. beliebteste Programmiersprache. Daraus folgt, dass das Ökosystem von JavaScript eine hohe Praxisrelevanz besitzt.

JavaScript wird nicht nur für die Entwicklung im Frontend, sondern auch für die Entwicklung im Backend verwendet. Denn ungefähr 2% der weltweit bekannten Server verwenden eine Laufzeitumgebung, die JavaScript ausführen kann. Die Laufzeitumgebung wird benötigt, um JavaScript außerhalb des Browsers ausführen zu können. Hierbei ist Node.js die am weitesten verbreitete Laufzeitumgebung. In einer Umfrage zum Zustand der JavaScript beantworteten ca. 71% von 30.000 befragten Entwicklern, dass sie Node.js als Laufzeitumgebung regelmäßig verwenden. Nur ca. 9% der befragten Entwickler verwenden Deno und ca. 3% Bun als eine Alternative zu Node.js.

Demnach ist Node.js der Platzhirsch im Kontext von Laufzeitumgebungen für JavaScript. Dennoch besitzt Node.js Schwächen, die die Alternativen versuchen zu lösen. Dazu zählen eine schwächere Performance bei anspruchsvollen Aufgaben, die

Limitierung auf einen einzelnen Thread und häufige Änderungen an der API.

1.2 Zielsetzung

Zuvor wurde Bun als eine mögliche Alternative zu Node.js erwähnt. Bun ist eine Laufzeitumgebung, die am 9. September 2023 in der Version 1.0 veröffentlicht worden ist. Die Entwickler von Bun haben das Ziel gesetzt, die Herausforderungen von Node.js zu bewältigen, um so die Akzeptanz des eigenen Frameworks zu steigern. Sie werben mit Features wie erheblicher Performancesteigerung, eleganten Schnittstellen und einer angenehmen Entwicklererfahrung.

Das Ziel der Arbeit besteht darin, Bun als eine mögliche Alternative zu Node.js zu evaluieren. Dabei liegt der Fokus auf der Überprüfung, ob die beworbenen Features tatsächlich in die Praxis umgesetzt wurden. Falls die Performance von Bun besser ist als die von Node.js, muss die Kompatibilität zu bestehenden Projekte geprüft werden. Andernfalls steigt der Migrationsaufwand enorm an und beeinflusst die Akzeptanz des Frameworks negativ. Zu diesem Zweck werden die folgenden Leitfragen herangezogen:

- Welche Laufzeitumgebung bietet die beste Performance?
- Ist Bun als Laufzeitumgebung kompatibel mit bestehenden Projekten auf der Basis von Node.js?

1.3 Aufbau der Arbeit

TODO

Kapitel 2

Grundlagen

Dieses Kapitel stellt die benötigten Grundlagen vor, die für das Verständnis der darauffolgenden Kapitel notwendig sind. Hierzu zählen die Vorstellung von Node.js und Bun sowie weiterer Grundlagen zu Performanceanalysen.

2.1 Node.js

Bei Node.js handelt es sich um ein beliebtes Tool für eine große Varianz an Projekten. Denn es handelt sich um eine Open Source, platformunabhängige Laufzeitumgebung, die es erlaubt JavaScript außerhalb des Browsers auszuführen. Es nutzt Googles V8 JavaScript Engine, die auch in Google Chrome verwendet wird. Dadurch kann Node.js eine gute Performance erreichen. Aufgrund dessen nutzen beispielsweise Netflix oder Uber Node.js in deren Software. [1]

Einsatzgebiete einfügen

Abbildung einfügen

zeigt die Architektur von Node.js selbst. Grundsätzlich nutzt Node.js nur einen Thread und erstellt nicht für jede neue Anfrage einen neuen Thread. Sobald eine Applikation gestartet wird, werden in dem einzelnen Thread der Node.js-Prozess gestartet. Der Quellcode wird hierbei in nativen Maschinencode übersetzt, um darauffolgend die Applikation auszuführen. Die V8 Engine optimiert den Maschinencode an häufig benötigten Stellen zusätzlich. Dies wird nicht zu Beginn direkt durchgeführt, da die Übersetzung in Maschinencode aufgrund der Just-in-Time-Kompilierung eine zeitintensive Aufgabe darstellt. Außerdem ist in der Engine ein Garbage Collector inkludiert, der nicht mehr verwendete Objekte löscht. [2]

Referenz auf Abbildung

Für weitere Aufgaben setzt Node.js auf Bibliotheken, die fertige und etablierte Lösungsansätze für häufig benötigte Aufgaben inkludieren. Nur für Aufgaben, für die

es keine etablierte Bibliothek gab, werden eigene Implementierungen herangezogen. Die wichtigsten Komponenten werden im Folgenden vorgestellt. [2]

Event Loop

Das Framework setzt auf eine eventgetriebene Architektur. Statt den Quellcode linear auszuführen, werden definierte Events gefeuert. Zuvor wurden Callback-Funktionen für die Events registriert. Diese werden nun ausgeführt. Dies wird verwendet um eine hohe Anzahl an asynchronen Aufgaben zu erledigen. Um dabei den einzelnen Thread für die Applikation nicht zu blockieren, werden Lese- und Schreibeoperationen an den Event Loop ausgelagert. Sobald auf eine externe Ressource zugegriffen werden muss, wird die Anfrage an den Eventloop weitergeleitet. Die registrierte Callback-Funktion leitet die Anfrage an das Betriebssystem weiter. Bis das Betriebssystem die Anfrage erledigt hat, kann Node.js andere Operationen ausführen. Das Ergebnis der externen Operation wird über den Event Loop zurückübermittelt. [2]

Libuv

Der Event Loop von Node.js basiert ursprünglich auf der Bibliothek libev. Diese ist in C geschrieben und für eine hohe Performance mit umfangreichen Features bekannt. Allerdings baut libev auf nativen Features von UNIX auf, die unter Windows auf andere Art und Weise zur Verfügung stehen. Daher bildet Libuv die Abstraktionsebene zwischen Node.js und den darunterliegenden Bibliotheken für den Event Loop, um die Laufzeitumgebung auf allen Plattformen verwenden zu können. Libuv verwaltet alle asynchronen I/O-Operationen. D. h. alle Zugriffe auf das Dateisystem oder beispielsweise asynchrone TCP- und UDP-Verbindungen über libuv laufen. [2]

Zusammenfassend besitzt Node.js eine event-getriebene Architektur inklusive eines nicht-blockierenden Modell für die asynchrone Ein- und Ausgabe, das es leichtgewichtig und effizient gestaltet. Daraus leiten sich diverse Vor- und Nachteile ab.
Neben der Plattformunabhängigkeit ist die Verwendung einer etablierten Engine von Vorteil. Dadurch stehen sämtliche Features der Programmiersprache zur Verfügung. Des Weiteren ist JavaScript als Sprache selbst durch die Standardisierung von ECMAScript gut dokumentiert. Zusätzlich hat sich eine große Community um die Open Source Software herum gebildet. Diese bietet viele Informationen, Tutorials und Lösungen für häufig Probleme. [2]
Nachteile sind ...

Event Queue und Loop von Ziel-Abbildung noch inkludieren, sobald Quelle gefunden

Nachteile einfügen

Darüber hinaus bringt Node.js den *Node Package Manager (NPM* mit. Der Paket-

manager ist essenziell für den Erfolg von Node.js. Denn mit mehr als 2,1 Millionen Paketen im September 2022 gibt es in diesem Ökosystem einen ein Paket für nahezu alle Anwendungsfälle. Der ursprüngliche Zweck von NPM war die Abhängigkeiten eines Projektes zu verwalten. Mittlerweile wird es auch als Tool für JavaScript im Frontend unterstützt. [1]

2.2 Bun

TODO

2.3 Performanceanalyse

TODO

Kapitel 3

Titel 3

3.1 Section 3.1

TODO

Kapitel 4

Titel 4

4.1 Section 4.1

TODO

Kapitel 5

Titel 5

5.1 Section 5.1

TODO

Literatur

[1] OPENJS FOUNDATION, Hrsg. *Introduction to Node.js*, 2022.

[2] SEBASTIAN SPRINGER. *Node.js : das umfassende Handbuch*. 4., aktualisierte und erweiterte Auflage. Rheinwerk Computing. Bonn: Rheinwerk, 2022. ISBN: 9783836287654. URL: \url{http://deposit.dnb.de/cgi-bin/dokserv?id=992e511c601d4a5f84179bebaa309635&prov=M&dok_var=1&dok_ext=htm}.

Anhang A

Anhang Kapitel 1