

Projektarbeit

Internationale Hochschule Duales Studium

Studiengang: Informatik

**Wie beeinflusst die Asynchronität von Operationen innerhalb eines Node.js Express Endpunkts die Performance in Lasttest?**

Müller Korbinian

Matrikelnummer: 102302316

Adresse

Lohwald Straße 59

86356 Neusäß

Abgabedatum: 30.09.2024

Inhalt

[Einleitung 3](#_Toc177330041)

[Theoretische Fundierung 3](#_Toc177330042)

[Begriffserklärung 3](#_Toc177330043)

[Node.JS und Express 3](#_Toc177330044)

[Asynchrone Programmierung 3](#_Toc177330045)

[Lasttests und Performace-Metriken 3](#_Toc177330046)

[Theoretische Grundlagen 3](#_Toc177330047)

[Zusammenhang zwischen Aynchronität und Performance 3](#_Toc177330048)

[Bedeutung für die serverseitige Anwendung 3](#_Toc177330049)

[Methodik 3](#_Toc177330050)

[Aufbau der Lastentests 3](#_Toc177330051)

[Testumgebung 3](#_Toc177330052)

[Implementation der Endpunkte 3](#_Toc177330053)

[Antwortzeiten und Durchsatz 3](#_Toc177330054)

[Hardware Auslastung 3](#_Toc177330055)

[Ergebnisse 3](#_Toc177330056)

[Testergebniss 3](#_Toc177330057)

[Performance Vergleich 3](#_Toc177330058)

[Fazit 3](#_Toc177330059)

[Literaturverzeichnis 3](#_Toc177330060)

Einleitung

Diese Forschungsarbeit beschäftigt sich mit der Frage,Wie beeinflusst die Asynchronität von Operationen innerhalb eines Node.js Express Endpunkts die Performance in Lasttest?

Zunächst wird in der Theoretischen Fundierung mit der Begriffserklärung den Grundstein gelegt. In der Methodik wird die Frege mittels eines Experiment auf die Probe gestellt. In diesem Abschnitt wird der Versuchsaufbau, die Durchführung und die Ergebnisse beschrieben. Im Anschluss werden die Ergebnisse diskutiert. Aus dieser Diskussion leitet sich zum Schluss das Fazit ab.

Theoretische Fundierung

Dieser Abschnitt der Arbeit beschäftigt sich mit den Theoretischen Grundlagen. Die Weiteren abschnitte basieren auf dem hier dargelegten Wissen. Zunächst werden Grundlegende Begriffe erläutert. In diesem Abschnitt wird erklärt, was unter Node.JS, Express, Asynchrone Programmierung und Lastentest zu verstehen sind. Im darauffolgen Abschnitt werden die Theoretischen Grundlagen erläutert auf denen die Tests basieren.

Begriffserklärung

In dieser Ausführung werden technische Konzepte und Technologien erwähnt. Damit jeder Leser auch die späteren Technisch Anspruchsvollen Passagen versteht werden in den Folgenden Abschnitte die wichtigsten Grundbegriffe ausführlich erläutert.

Node.JS und Express

Node.JS ist ein JavaScript Framework, dass die durch Google Entwickelte V8 Engine, nutzt, um JavaScript außerhalb des Webbrowsers auszuführen. Dabei wird der Quellcode Asynchron ausgeführt. Beim Start des Node.JS Prozesses wird eine Eventloop gestartet, die auf eingehende Events reagiert und diese Asynchron abarbeitet (Huang and Cai, 2018, S.1-3).

Anhand dieser einzigartigen Architektur ist Node.JS in der Lage sehr effizient zu Arbeiten. Mit einer Speichergröße von 8 GB sind maximal 40000 Verbindungen mit dem Webserver möglich. Im Vergleich dazu sind bei herkömmlichen Serverseitigen Programmiersprachen wie Java oder PHP, in etwa 4000 Nutzer möglich. Dies beruht auf der Annahme, dass die Herkömmlichen Programmiersprachen in etwa 2MB Arbeitsspeicher für jeden neuen Thread benötigen. Neben diesen Stärken weißt Node auch eine Schwäche für CPU intensive Aufgaben auf(Huang, 2020).

Laut der Entwicklerumfrage von Stackoverflow (<https://survey.stackoverflow.co/2024>) ist NodeJS eine der Begehrtesten Programmiersprachen. Kein Wunder dass auch auf Node.JS selber bei den Web-Frameworks an Zweiterstelle ist. An siebter Stelle ist das Framework Express.JS, dieses basiert auf Node.JS. Anhand dieser Popularität wurde das Framework Express.JS für diese Arbeit ausgewählt. Die Entwickler von Express beschreiben dieses als ein schnelles und minimalistisches Web-Framework für Node.js. Es bietet eine große Anzahl an HTTPS Funktionalitäten, dadurch wird es erleichtert schnell und leicht, robuste APIs zu entwickeln (<https://expressjs.com>).

Asynchrone Programmierung

Unter Asynchronität in der Programmierung versteht man, wenn der Programmcode nicht Kontinuierlich ausgeführt wird. Dieser wartet bis Daten angekommen sind oder ein Event ausgelöst wird. Dadurch werden die Systemresourcen freigegeben, wenn diese nicht benötigt werden. Diese Funktionalität ist sehr stark in JavaScript integriert (Flanagen, 2020 S.342).

Es ist wichtig zu erwähnen, dass Java und Node.JS nutzen bei der Ausführung von Quelltext nur einen Thread. Durch die Asynchrone Auslegung von Node.JS ist es dennoch möglich einen hohen Grad an parllität in der Ausführung zu erreichen. Es kommt zu einer Performance, die in anderen Programmiersprachen mittels Threads erreicht wird, mit einem Bruchteil der Resourcennutzung (Flanagen, 2020 S.583).

Lasttest und Performance-Metriken

Eines der wichtigsten Themen das beim Testen anfällt ist der Test der Performance. Dabei muss nicht die gesamte Software Umgesetzt sein, sobald die ersten Elemente umgesetzt sind kann damit angefangen werden. Im Bereich der Webentwicklung ist eine der Wichtigsten Fragen, welches die maximale Anzahl an Nutzern ist. Damit die Ergebnisse aussagekräftig sind, ist es essenziell das die Test so nah an der produktiv Umgebung ist wie möglich (Kleuker, 2019 S.355-356).

In seinem Buch beschreibt Kleuker(2019), dass eine weitere Maßnahme für die Qualitätsbewertung sind Metriken. Dabei wird ein Kriterienkatalog mit relevanten Aspekten erstellt. Im Bereich Softwareentwicklung können die Aspekte automatisiert überprüft und ausgewertet werden (Kleuker, 2019 S22-23).

Für die Evaluation in dieser Projektarbeit werden folgende Metriken ausgewertet:

* Antwortzeit: Die benötigte Zeit die Anfrage an den Server zu schicken, diese zu Bearbeiten und das Ergebnis zurück zu Senden. Die Zeit wird in Millisekunden, MS, gemessen
* Durchsatz: Die Anzahl an Reuests die in einer Sekunde verarbeitet werden können
* CPU- und Speicherverbrauch: Diese bestimmen wie effizient der Server seine Ressourcen währen der Nutzung einsetzt.

Theoretische Grundlagen

Nach der Erläuterung der grundlegenden Begriffe werden nun die Theoretischen Grundlagen geschildert. Diese dienen als Grundlage für das durchgeführte Experiment. Diese Grundlagen sind entscheidend um die Leistungsmessungen und Lasttests in Bezug zu Asynchrone API Endpunkte im weiteren Verlauf zu verstehen.

Zusammenhang zwischen Asynchronität und Performance

Huang und Cai (2018) haben bereits erläutert das Node.js Hardware nahen Code Asynchron ausführt (Huang and Cai, 2018, S.1-3). Dadurch wird der Zugriff auf die Ressourcen b

Bedeutung für die serverseitige Anwendung

Dadurch entstehen

Methodik

Dieser Abschnitt der Arbeit befasst sich mit dem Praktischen. Zunächst wird erläutert wie die Tests Strukturiert sind. Im Anschluss wird dargestellt wie sich die Testumgebung zusammensetzt. Darauf folgt die Implementierung. In den Letzten Abschnitten wird erläutert wie die einzelnen Tests durchgeführt werden.

Testumgebung

Um die Experimente durch zu führen wird ein Webserver mit der Node.js Packet Express.JS entwickelt dieser Stellt insgesamt zwei Endpunkte bereit. Einer der Endpunkt wird als ein Synchroner Endpunkt entwickelt. Der andere Als ein Asynchroner. Um ein gewissen Realismus zu erreichen werden beide Endpunkte den Request annehmen und mit den Daten ein Zugriff auf eine MySQL Datenbank machen. Das Ergebnis der MySQL abfrage wird noch zusätzlich einmal Aufbereitet und in Form eines JSON Objekt als Response zurück geschickt.

Um die Tests durchzuführen wird der Express.JS Server in einem Docker Container ausgeführt. Ebenso der MySQL Server. Um das Deployment zu vereinfachen wird dieses mit Docker Compose durchgeführt.

Die Tests werden auf einem Desktop PC ausgeführt. Der PC Besitzt folgende Hardware Komponenten:

* IntelCore I9-9900 KF, nicht übertaktet
* 32 GB DDR-4, 3200 GHZ Arbeitsspeicher
* Nvidia RTX 2080 S

Folgendes sind die Software die Für die Entwicklung und Durchführung des Experimentes Genutzt worden sind:

* Windows 10 Proffesional
* WebStorm, JetBrains
* Docker, Version XX, 16 GB Arbeitsspeicher
* Node.JS, Version 20.11.1
* Express, Version 10.2.4
* Artillery, Version , Testsoftware

Aufbau der Lastentests

Die Testphase dauert 3 Minuten. In dieser Zeit werden User simuliert die auf den Webservice zugreifen. Die Tests werden zunächst mit 100 gleichzeitigen Usern durchgeführt. Um eine Höhere Anspruch an die Software werden in anderen Durchläufe 1000 und 10.000 Nutzer simuliert. Damit die Reultate verifiziert werden können wird jeder Test insgesamt drei Mal durchgeführt.

**Antwortzeit und Durchsatz**

Um die Antwortzeit und den Durchsatz der Api-Endpunkte zu ermitteln wird das Tool Artillery eingesetzt. Es ist ein Open Source Framework für das Testen von Apis uns Webserver. Es ist eine Konsolen Anwendung die in Node.Js entwickelt wurde. Zunächst einmal wird eine YML Konfigurations Datei angelegt. In dieser wird die URL spezifiziert die getestet werden soll. In diesem Fall ist es <http://localhost:3000>. Im Nächsten Abschnitt werden die Test Spezifikationen aufgeführt, dabei wird festgelegt wie Lange der Test Laufen Soll und wie viele gleichzeitige Nutzer auf die Api Zugegriffen sollen. Das Programm aggregiert die Informationen in einer JSON-Log Datei. Diese wird später ausgewertet.

**Hardware Auslastung**

Implementation der Endpunkte

Antwortzeiten und Durchsatz

Hardware Auslastung

Ergebnisse

Testergebniss

Performance Vergleich

Fazit

Literaturverzeichnis