

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. 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, leicht und 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 Paralität 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

Lasten-Tests sind eine Methode, die Leistung der Software zu evaluieren. Dabei wird in einer kontrollierten Umgebung Anforderungen and die Software gestellt die im normalen Einsatz erwartet werden. In diesem Fall werden kontrolliert Anfragen an die implementierte API gestellt. Dabei wird die Intensität über die Zeit hinweg erhöht. Das Ziel ist es festzustellen ab welchem Punkt die API an ihrem Grenzen stößt. Im Grenzbereich ist es möglich Engpässe zu identifizieren.

In diesem Zusammenhang sind die Performance-Metriken entscheidend. Diese werden zu der Beurteilung des Ergebnis benötigt. Die wichtigsten Metriken für APIs sind:

* 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 Definition der grundlegenden Begriffe geht es nun weiter mit den Theoretischen Konstrukten die

Zusammenhang zwischen Aynchronität und Performance

Bedeutung für die serverseitige Anwendung

Methodik

Aufbau der Lastentests

Testumgebung

Implementation der Endpunkte

Antwortzeiten und Durchsatz

Hardware Auslastung

Ergebnisse

Testergebniss

Performance Vergleich

Fazit

Literaturverzeichnis