INTERNATIONALE HOCHSCHULE

Exposé

Internationale Hochschule Duales Studium

Studiengang: Informatik

Wie beeinflusst die Asynchronität von Operationen innerhalb eines PHP-Rest-Endpunkts die Performance in Lasttests?

Müller Korbinian

Matrikelnummer: 102302316

Adresse

Lohwald Straße 59

86356 Neusäß

Abgabedatum: 16.03.2025

Inhalt

Einleitung

PHP gehört zu den ältesten und am weitesten verbreiteten Programmiersprachen im Web. Trotz seiner langen Geschichte ist PHP bis heute hochaktuell und integraler Bestandteil moderner Webanwendungen. Wie die Stack Overflow Developer Survey 2024 feststellt, gilt: „PHP bleibt trotz der Konkurrenz ein unverzichtbarer Bestandteil moderner Webentwicklung“ (Stack Overflow Developer Survey, 2024). Ergänzend dazu zeigt eine Analyse von Statista: „Der Marktanteil von PHP-basierten Anwendungen ist konstant hoch“ (Statista, 2024), was den anhaltenden Einfluss der Sprache im globalen Webumfeld unterstreicht.

Infolge der wachsenden Popularität hat PHP kontinuierlich innovative Features integriert, um den steigenden Anforderungen an Effizienz und Skalierbarkeit gerecht zu werden. Insbesondere die Einführung asynchroner Programmierparadigmen markiert einen bedeutenden Entwicklungsschritt. Wie Duarte und Hametner (2017) erläutern, ermöglicht asynchrone Programmierung das parallele Ausführen von Subroutinen, ohne den primären Ablauf zu unterbrechen – ein Ansatz, der die klassischen Limitierungen synchroner Prozesse adressiert. Diese Neuerung bietet vornehmlich bei zeitintensiven Operationen, wie komplexen Datenbankabfragen, erhebliches Potenzial zur Steigerung der Systemperformance.

Die fortschreitende Digitalisierung und der steigende Bedarf an schnellen, performanten Webanwendungen machen die Optimierung der Softwareleistung unverzichtbar. Der direkte Zusammenhang zwischen der Popularität einer Technologie und der Notwendigkeit leistungsfähiger Systeme wird deutlich: Je stärker PHP verbreitet ist, desto größer sind die Erwartungen an eine effiziente und skalierbare Umsetzung. In diesem Kontext rückt die asynchrone Ausführung von Operationen in den Fokus, da sie wesentlich zur Reduktion von Antwortzeiten und zur Verbesserung des Durchsatzes beitragen kann.

Die vorliegende Arbeit widmet sich daher der zentralen Fragestellung: Wie beeinflusst die Asynchronität von Operationen innerhalb eines PHP REST Endpunkts die Performance in Lasttests? Mit diesem Fokus wird eine bestehende Forschungslücke adressiert, indem der direkte Performance-Vergleich zwischen synchronen und asynchronen Ansätzen in PHP untersucht wird.

Theoretische Fundierung

Asynchrone Programmierung

In Ihrem Artikel beschreiben Duarte und Hametner (2017) die Asynchrone Programmierung als eine Ansammlung von Subroutinen, die ausgeführt werden können, ohne den primären Programmablauf zu unterbrechen.

Diese Definition dient als Verständnisgrundlage in der folgenden Arbeit

Steyer(2024, S. 291) erklärt in seinem Buch, dass der Quellcode in der Regel nur von einem Thread ausgeführt wird. Dadurch entstehen Probleme bei Operationen, die inhärent eine längere Zeit für das Ausführen benötigen. Als Beispiel nennt er das Abrufen von Daten vom Server.

Die Quellen lassen die Vermutung zu, dass der asynchrone Ansatz bei der Programmierung bereits einige Performancevorteile gegenüber synchroner Programmierung besitzen.

Lasttest

Performance und Lasttest spielen eine zentrale Rolle beim Testen von Software. Hierbei wird die Geschwindigkeit und der Speicherverbrauch der Software ermittelt. Als Versuchsumfeld wird eine Umgebung genommen, die sehr ähnlich zur endgültigen Systemumgebung ist. Über eine Annäherung an den Grenzbereich lässt sich feststellen, wie viele gleichzeitige Nutzer ein System benutzen können (Kleuker, 2019, S. 355f-S. 394f).

Von Gruszczynski und Zabierowski (2022) werden insgesamt sechs verschiedene Arten von Performance-Tests beschrieben. Sie sprechen von Load-Testing, Stress-Testing, Endurance-Testing, Spike-Testing, Volumen-Testing und Scalability-Testing. Load-Testing ist beschrieben als ein Test mit spezifizierter Belastung. Stress-Testing hingegen als ein Test mit extremer Last.

Load-Test und Stress-Test sollen bei der Durchführung der Experimente angewandt werden.

Synchrone und Asynchrone Programmierung im Vergleich

In Ihrer Studie vergleichen Gruszczynski und Zabierowski (2022) die Programmiersprachen PHP und Java. Getestet wurden mathematische Berechnungen, Sortieralgorithmen und Datenbankoperationen. Sie sind zu dem Schluss gekommen, dass PHP fast so performant ist wie Java, aber nur einen Teil der Ressourcen benötigt.

Vandikas et al.(2011) untersuchten, was für einen Einfluss die Asynchrone Programmierung in Java auf die Performance hat. Dabei sind sie zu der Entdeckung gekommen, dass die asynchrone Programmierung den Durchsatz bis zu 49 % verbessert und die Antwortzeit reduziert.

In Ihrer Studie untersuchten Luo et al. (2024) die Nützlichkeit von Node.js als Server-Framework für hochbelastete Webseiten und Webanwendungen. Bei dem Test stellte das Team fest, dass die Effizienz der Anwendung mit dem Grad der Parallelität stieg. Bei einer multithreaded Implementation stellten sie eine Performanceverbesserung von 150 % fest. Ebenso zeigten Sie, dass Node.js, welches von Haus aus asynchron arbeitet, einen besseren Durchsatz erzielt als ein Apache Webserver.

Die letzten beiden Studien zeigen, dass eine Asynchrone Programmierung die Performance erhöhen kann. Die Studien selbst beschäftigten sich jedoch nicht mit der Programmiersprache PHP. Es lässt sich die Vermutung aufstellen, dass diese Funde auch auf PHP übertragen werden können.

Forschungsfrage

In den vorgestellten wissenschaftlichen Arbeiten wurden Aspekte der asynchronen Programmierung und Performance-Vergleiche zwischen verschiedenen Programmiersprachen dargestellt. Jedoch beschreiben die Quellen den Zusammenhang von asynchroner Programmierung und dem daraus resultierenden Performance-Einfluss nur in Bezug auf andere Programmiersprachen. Daraus resultiert eine eindeutige Forschungslücke. Der Bereich dieser Forschungslücke ist der direkte Performance-Vergleich von synchroner und asynchroner Programmierung mit PHP. Die Forschungslücke kann durch folgende Forschungsfragen geschlossen werden.

Wie beeinflusst die Asynchronität von Operationen innerhalb eines PHP-Rest-Endpunkts die Performance in Lasttests?

Methodik

Die Forschungsfrage soll mithilfe eines Experiments überprüft werden. In dem Experiment werden zwei RESTful API Endpunkte entwickelt. Der eine Endpunkt arbeitet synchron und der andere asynchron. Die beiden Endpunkte werden einer Reihe an Lasttest unterzogen. Dabei soll herausgefunden werden, wie die Performance der API sich verhält, wenn das Anfragevolumen sich erhöht. Dabei werden mit einem API-Test-Tool Anfragen an den Endpunkt generiert. Das Volumen der Anfragen wird während des Tests Stück für Stück erhöht. Insbesondere wird überprüft, wie die Antwortzeit in Relation zu den Anfragen steht.

Für das Experiment wird in PHP mit dem Framework Slim eine RESTful API entwickelt. Die API stellt zwei Endpunkte zur Verfügung. Der Synchrone wird mit Standard PHP entwickelt. Im Asynchronen kommt zusätzlich noch Guzzle zum Einsatz. Dadurch werden die asynchronen Features ermöglicht. Die beiden Endpunkte sind gleich strukturiert. Die Anfrage wird angenommen und anhand der Parameter wird eine Query in einer MySQL-Datenbank ausgeführt. Dabei wird darauf geachtet, dass es sich um eine sehr komplexe Query handelt. Dadurch wird sichergestellt, dass die Abfrage eine Zeit für die Ausführung benötigt. Die Ergebnisse der Datenbankabfrage werden in JSON-Format aufbereitet und als Response zurückgesendet. Dies bietet eine realistische Nutzung eines Restpunktes.

Das Experiment wird auf einem Desktop PC ausgeführt. Die Systemumgebung, MySQL-Datenbank und Apache Webserver, wird mittels Docker erzeugt. Dafür werden die bereits bestehenden Container webdevops/php-apache und mysql verwendet. Dadurch ist die Systemumgebung klar von der Maschine getrennt. Des Weiteren ist es dadurch möglich, die Systemumgebung für jeden Durchlauf zurückzusetzen. Dadurch wird sichergestellt, dass die vorherigen Experimente die Nachfolgenden nicht beeinflussen.

Die Überprüfung der Antwortzeit erfolgt mit der Anwendung Artillery. Dabei handelt es sich um eine Node.js-CLI-Anwendung, die dafür geschaffen wurde, um die Performance von Webanwendungen zu testen. Es wird definiert, wie lange die Testphase dauert und wie viele Anfragen pro Sekunde gesendet werden. Im ersten Schritt wird über eine Zeit von 5 Minuten die Anzahl der Anfragen auf 1000 pro Sekunde erhöht. Dies dient der ersten Einschätzung. Falls dabei festgestellt wird, dass beide Endpunkte bei 1000 Anfragen pro Sekunde keine nennenswerte Antwortzeit aufweisen, dann wird die Maximalbelastung in 1000 Schritte erhöht, bis nennenswerte Ergebnisse vorliegen. Bei dem Testen werden fünf Sekunden als maximal zugelassene Antwortzeit betrachtet.

Artillery liefert Daten für die minimale, maximale und mittlere Antwortzeit zurück. Ebenso auch das 99 Perzentil. Die Ergebnisse der Endpunkte werden miteinander verglichen. Falls die Asynchrone Programmierung einen Einfluss auf die Performance besitzt, dann werden die Messdaten für den asynchronen Endpunkt niedriger ausfallenals für den synchronen. Im Falle, dass die mittlere Antwortzeit und das 99-Perzentil bei beiden Endpunkten gleich sind, können die minimalen und maximalen Antwortzeiten einen Einblick in das Verhalten im Extrembereich liefern.

Gliederungsentwurf

1. Einleitung, 1.5 Seiten
2. Theoretische Fundierung, 4 Seiten
    a. PHP und Slim
    b. Asynchrone Programmierung
    c. Lasttest und Performance-Metriken
3. Methodik, 5 Seiten
    a. Aufbau des Experiments
    b. Die verwendeten Daten
    c. Strukturierung der Lasttests
    d. Die Implementierung der RESTful API
4. Ergebnisse des Experiments, 3 Seiten
5. Fazit, 1.5 Seiten
6. Literaturverzeichnis

Zeitplan

1.5.2025-29.06.2025	Quellensuche und Validation
30.06.2025-13.07.2025	Verfassen der theoretischen Fundierung und Methodik
14.07.2025-20.07.2025	Bufferzeit, für die Überarbeitung und Verbesserung des Verschriftlichens
21.07.2025-27.07.2025	Implementierung der Endpunkte
28.07.2025-03.08.2025	Bufferzeitbei Problemen in der Implementierung
4.08.2025-10.08.2025	Durchführen des Experiments
11.08.2025-24.08.2025	Analyse, Implikation und Fazit verfassen
25.08.2025-31.08.2025	Verfassen der Einleitung
1.09.2025-14.09.2025	Korrekturlesen, Ausbessern der Formulierungen
15.09.2025-21.09.2025	Bufferzeit, für Nachbesserungen nach dem Korrekturlesen.
22.09.2025	Abgabe der wissenschaftlichen Arbeit

30.09.2025	Deadline für die Abgabe

Vorläufiges Literaturverzeichnis

Textquellen

Duarte, O. M., & Hametner, R. (2017). Asynchronous programming with futures in C on a safety-critical platform in the railway-control domain. *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Emerging Technologies and Factory Automation (ETFA), 2017 22nd IEEE International Conference on*, 1–8. https://doi.org/10.1109/ETFA.2017.8247589

Gruszczynski, K., & Zabierowski, W. (2022). Performance Tests of Java and PHP Programming Languages for Use in Sensor Networks. *2022 IEEE XVIII International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), Perspective Technologies and Methods in MEMS Design (MEMSTECH), 2022 IEEE XVIII International Conference on the*, 70–74. https://doi.org/10.1109/MEMSTECH55132.2022.10002927

Kleuker, S. (2019). Performance- und Lasttests. *Qualitätssicherung durch Softwaretests : Vorgehensweisen und Werkzeuge zum Testen von Java-Programmen*, 355–395. https://doi.org/10.1007/978-3-658-24886-4_12

Luo, J., Zhou, B., Zheng, Y., & Pan, W. (2024). Research on high performance web service construction method based on JavaScript asynchronous programming technique. *Applied Mathematics and Nonlinear Sciences*, *9*(1). https://doi.org/10.2478/amns-2024-2811

Steyer, R. (2025). Asynchrone Programmierung. *Programmieren mit JavaScript : Die vielseitige Sprache für Webentwicklung & mehr – Grundlagen und fortgeschrittene Techniken*, 291–345. https://doi.org/10.1007/978-3-658-45577-4_13

Vandikas, K., Quinet, R., Levenshteyn, R., & Niemoller, J. (2011). Scalable service composition execution by means of an asynchronous paradigm. *2011 15th International Conference on Intelligence in Next Generation Networks, Intelligence in Next Generation Networks (ICIN), 2011 15th International Conference on*, 157–162. https://doi.org/10.1109/ICIN.2011.6081065

Internetquellen

Statista. (2025). *Die beliebtesten Programmiersprachen weltweit laut PYPL-Index im Januar 2025 [Graph]*. Statista GmbH. Abgerufen am 08.03.2025, von https://de.statista.com/statistik/daten/studie/678732/umfrage/beliebteste-programmiersprachen-weltweit-laut-pypl-index/

Stack Overflow. (2024). *Stack Overflow Developer Survey 2024: Most Popular Technologies*. Abgerufen am 08.03.2025 von https://survey.stackoverflow.co/2024/technology#most-popular-technologies