Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

InIT Institut für angewandte Informationstechnologie

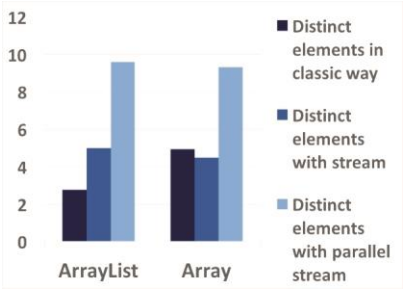
Best Practices für performante Java-Programmierung [Software Engineering]

Diplomierende
Marco De Tomasi
Markus Rutz

Dozierende
Mark Cieliebak
Markus Thaler

Qualitativ hochwertige Software muss vereinfacht ausgedrückt korrekt, effizient bzw. performant und wartungsfreundlich sein. Wann ist Software performant? Das Ziel dieser Bachelorarbeit ist, eine Sammlung von Best Practices für die Entwicklung von effizienter Software zu finden. Dazu werden Code-Fragmente mit unterschiedlichen Implementierungsansätzen für das Lösen einer einzelnen Aufgabe gegenübergestellt.

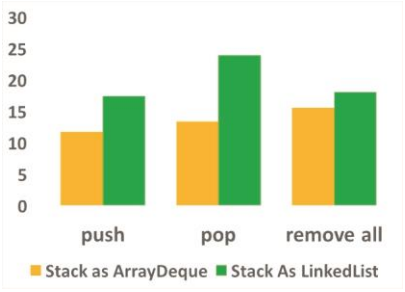
Die Testumgebung für das Messen der Laufzeiten einzelner Codeabschnitte wurde aus einer früheren Diplomarbeit übernommen und angepasst. Um genaue Messresultate zu erhalten, wurde das Benchmarking-Framework JMH (Oracle) verwendet. Die Resultate der Testreihen wurden als einzelne Dateien abgespeichert und in einem weiteren Schritt in die Datenbank übernommen. Nach dem Importieren der Resultate in die Datenbank konnten die Resultate als Balkendiagrammen dargestellt werden. Dies ermöglichte eine visuelle Auswertung der Laufzeiten.



Laufzeiten (in Millisekunden) der verschiedenen Varianten zur Entfernung von mehrfach vorhandenen Werten aus einer ArrayList oder aus einem Array mit je 100'000 Werten.

Verschiedene Code-Fragmente wurden mit Hilfe der Testumgebung untersucht. Wie zeiteffizient ist beispielsweise das Löschen von mehrfach vorhandenen Werten aus Arrays oder Listen mit Hilfe von Streams? Bringt in diesem Fall die Nutzung von Streams einen Vorteil gegenüber dem Einsatz eines klassischen Programmieransatzes, also dem Umwandeln der beiden vorliegenden Datenstrukturen in ein Set? Wie sollte ein Stack implementiert werden, so dass dieser bei allen Operationen möglichst schnell ist?

Zum Beispiel führt das Löschen von mehrfach vorhandenen Werten in Listen mit Hilfe von Streams zu Performance-Einbussen. Mit der Konvertierung des Arrays zu einem Set wird eine tiefere Performance erreicht als mit Einsatz eines sequenziellen Streams. Bei der Verwendung eines Parallel-Streams ist die gemessene Laufzeit bei Arrays circa zwei und bei Listen circa drei Mal langsamer als bei der Verwendung von Sets. Ein als ArrayDeque implementierter Stack ist in allen relevanten Operationen schneller als ein Stack, der als LinkedList umgesetzt ist.



Laufzeiten (in Millisekunden) der relevanten Operationen auf einem Stack bei 500'000-facher Ausführung.