Cursus Curruum Computatorum

Leider hat der Weihnachtsmann Caesar von der Rundumerneuerung seines Softwaresystems erzählt. Jetzt wollen auch die Römer ein völlig neuartiges, virtuelles Wagenrennen auf den Markt bringen. Also ist wieder Ihre Expertise gefragt. Im Kern geht es um das hoffentlich allen bekannte Brettspiel "Ave Caesar", bei dem jeder Spieler mit seinem Streitwagen mehrere Runden bestreiten und dabei mindestens einmal grüßend an Caesars Tribüne vorbeischauen muss. Der Parcours selbst ist voller Engpässe und Schikanen, an denen nicht überholt werden kann. Dieses Rennen mit römischen Streitwagen soll in eine digitale Virtualität übertragen werden. Da man nie weiß, wieviele Teilnehmer es gibt und wie lang die Strecke ist, dürfen nur horizontal skalierende Streaming-Architekturen eingesetzt werden. Alles weitere steht umseitig im Kleingedruckten. Um den Anreiz zu erhöhen, wird Caesar den Schöpfer der besten Lösung mit Gold überhäufen. Allen anderen schenkt er einen Besuch in der Gladiatorenarena - direkt inmitten der Gladiatoren und Löwen.

Universität Trier

Fachbereich IV / Informatik
Prof. Dr. Peter Sturm
D-54286 Trier
Telefon 0651/201-3313
Fax 0651/201-3842
Email sturm@uni-trier.de

SA4E, Übungsblatt 3, Winter 2024

Aufgabe 1 „Nur der Schnellste gewinnt“

In dieser ersten Aufgabe gibt einen einfachen Rundkurs, der aus einer bestimmten Anzahl an kreisförmigen Bahnen besteht, die alle wiederum aus einer festgelegten Anzahl an Segmenten zusammengesetzt sind. Es gibt keine Aufspaltungen und Engpässe und auch Caesar wartet noch nicht auf seine Begrüßung. Für die Erzeugung entsprechender Streckenbeschreibungen in JSON gibt es ein kleines Python-Script, das sich in einem Unterordner `CuCuCo` im gleichen Ordner wie die Videos befindet. Ein von Ihnen entworfenes Konfigurationsprogramm soll eine Streckenbeschreibung einlesen und die notwendige Systemarchitektur automatisch daraus generieren. Jedes Segment soll durch ein geeignetes Programm in einer Programmiersprache Ihrer Wahl implementiert werden. Die Segmente kommunizieren in dieser ersten Aufgabe ausschließlich über einen Server einer horizontal skalierbaren Streaming-Architektur (z.B. Apache Kafka, Redis Streams, Apache Pulsar, Apache Flink, Apache Storm). Für alle diese Streaming-Architekturen existieren einfach zu instanziierende Docker-Images. Andere Formen der Kommunikation sind nicht zulässig! Die Programme, die für die `start-and-goal`-Segmente zuständig sind, sollen durch ein CLI-Kommando gestartet oder informiert werden, mit dem Ziel, die Streitwagen in der Form von Token auf der jeweiligen Spur ins Rennen zu schicken. Alle anderen Segmente warten und leiten ein eingehendes Token schnellstmöglich an das nachfolgende Segment weiter. Am einfachsten bilden Sie die notwendige Kommunikationsstruktur auf das in allen Streaming-Plattformen vorhandene Publish/Subscribe-Muster ab, Sie dürfen aber gerne auch anders vorgehen. Nach einen vorgegebenen Anzahl an Umläufen ist das Rennen zu Ende und die Gesamtlaufzeit jedes Streitwagens wird ausgegeben.

Aufgabe 2 Cluster

Der zentrale Streaming-Server soll im Hinblick auf Skalierbarkeit und Zuverlässigkeit durch ein Cluster mit mindestens 3 Serverinstanzen ersetzt werden.

Aufgabe 2 „Ave Caesar“

Denken Sie über eine Erweiterung der Anwendung gemäß dem Originalspiel nach und implementieren Sie diese. Es werden sich zusätzliche Segmenttypen wie „`Caesar`“ und „`Bottleneck`“ ergeben. Aufspaltungen auf mehrere nachfolgende Segmente können durch eine Aufzählung der in Frage kommenden Folgesegmente (`nextSegments`) einfach umgesetzt werden. Um Engpässe anschaulicher zu realisieren, sollten die Streitwagen in jedem Segment für eine gewisse Zufallszeit verbleiben und damit nachfolgende Streitwagen am Weiterkommen hindern. Weitere Optimierungen des Renngeschehens und zusätzliche kreative Segmenttypen sind optional. Erweitern Sie das vorgegebene Python-Script, um entsprechende Rennstrecken mit den von Ihnen eingeführten Schikanen zu generieren.

Abgabe

Die Abgabe zu diesem dritten Übungsblatt besteht – wie üblich – aus einem Link auf Ihr Code-Repository und einem 3–7-seitigen Ergebnisbericht, in dem Sie auf Ihre Implementierung und Umsetzung eingehen. Der Bericht kann mir per Email oder Discord zugesendet werden (in diesem Fall enthält der Bericht die Links auf die Sourcecode-Repositories) oder der Bericht ist Teil eines Sourcecode-Repositories (Abgabe über einen Link). Deadline für die Abgabe ist der 24.3.2025. In einer freiwilligen Nachbesprechung am Anfang des Sommersemesters werden wir die gewonnenen Erfahrungen austauschen; zu gegebener Zeit werde ich eine Umfrage für diesen Termin starten. Wer möchte, darf dafür gerne optional einen kurzen Vortrag vorbereiten.