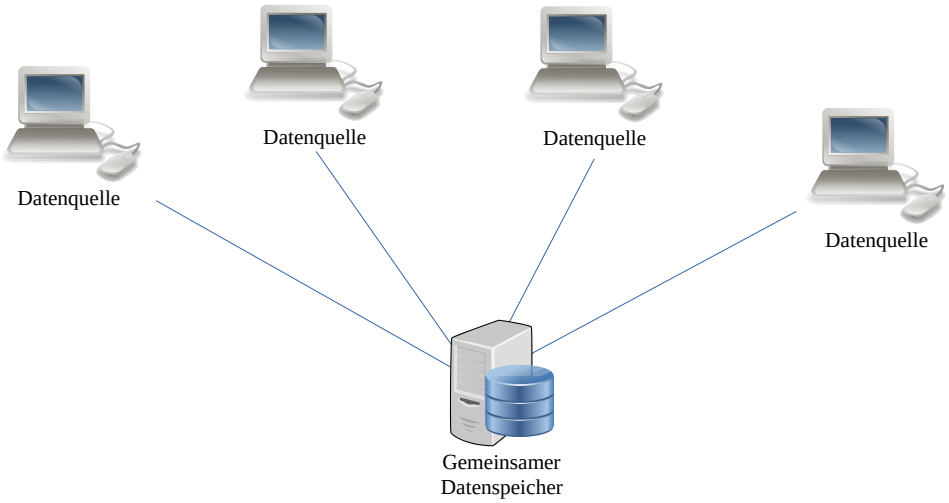
Lamports Algorithmus zum gegenseiten Ausschluss

Lamport's Algorithm for Mutual Exclusion

Setup

Mehrere verteilte Datenquellen sollen auf einen (gegebenen) gemeinsamen Datenspeicher zugreifen. Um Schreibkonflikte auf dem Datenspeicher zu verhindern, soll der Zugriff der Datenquellen auf den gemeinsamen Speicher mittels Lamports Algorithmus zum gegenseitigen Ausschluss dezentral koordiniert werden.



Ihre Aufgabe

Implementieren Sie eine horizontal skalierbare Datenquelle, die auf einen (gegebenen) Datenspeicher (schreibend) zugreift. Die einzelnen Instanzen der Datenquelle koordinieren Ihren Zugriff mittels Lamports Algorithmus zum gegenseitigen Ausschluss.

Anforderungen

- Der Zugriff auf die Datenquelle erfolg mittels der vorgegebenen APIs entweder mit Thrift, gRPC **oder** TCP+Protobuf. Bitte wählen Sie **eine** Technologie.
- Der Zugriff erfolgt ausschließlich über die vorgegebenen API-Definitionen auf den (gegebenen) Server. Eine Änderung der API ist nicht gestattet. Die API-Definitionen (Thrift-File bzw. Proto-File) finden Sie in Moodle und auf GitLab unter:
  - https://code.fbi.h-da.de/m.bredel/datastore

- Die Anwendung soll die Übertragung von Daten unterschiedlicher Größe zu unterschiedlichen Zeiten an den gemeinsamen Datenspeicher simulieren. Hierzu soll die Anwendung eine CSV-Datei folgenden Formats einlesen und verarbeiten:

  Wartezeit (in Sekunden), Dateigröße

  Beispiel:

  0,5
  5,4
  2,6
  0,3

  Die Anwendung verschickt nach 0 Sekunden Wartezeit eine Datei der Größe 5. Nach Abschluss der Transaktion wartet die Anwendung 5 Sekunden und verschickt dann eine Datei der Größe 4. Weitere 2 Sekunden nach Abschluss der Transaktion verschickt die Anwendung eine Datei der Größe 6 und direkt im Anschluss, nach 0 Sekunden Wartezeit, eine Datei der Größe 3.

  - Eine einfache Beispielanwendung, die dieses Konzept rudimentär in Java umsetzt, finden Sie unter: https://code.fbi.h-da.de/m.bredel/csv-reader
  - Der Pfad zur CSV-Datei muss per Environment-Variable konfigurierbar sein.
- Die Anwendung muss horizontal skalierbar sein. Es müssen sich somit beliebig viele Datenquellen (in beliebig vielen Docker-Containern) starten lassen.
- Die Anwendung muss sich in einem Docker-Container kompilieren und ausführen lassen. Ein entsprechendes Dockerfile muss mitgeliefert werden.
- Das gesamte Setup aus (mindestens drei) Datenquellen, einem Datenspeicher und eventueller weiterer Komponenten muss sich über Docker-Compose starten lassen. Eine entsprechende Docker-Compose Datei muss mitgeliefert werden.
- In der Docker-Compose Datei muss der Datenspeicher folgendermaßen referenziert werden:

  image: mbredel/datastore:latest

- Der gesamte Code, inkl. Readme, Dockerfiles, Docker-Compose Dateien, etc., muss im Masterbranch des zur Verfügung gestellten GitLab-Repos abgelegt sein
  - Repo: https://code.fbi.h-da.de/distributed-systems/2020_wise_exam/*vorname.nachname*
- Der gesamte Code, inkl. Readme, Dockerfiles, Docker-Compose Dateien, etc., muss bis Freitag, 12. März 2021 um 23:59 in GitLab zur Verfügung stehen.
- Externe Bibliotheken dürfen verwendet werden, sofern sie nicht das zentrale Problem der Aufgabenstellung lösen. Eine Library z.B. für Lamport Zeiten wäre somit nicht zulässig.

Hinweise

- Sie können davon ausgehen, dass die Anzahl der Datenquellen konstant und im Vorfeld bekannt ist. Zudem können Sie davon ausgehen, dass das Netzwerk stabil ist und es keine Server-Ausfälle gibt.
- Den Quellcode des Datenspeichers finden Sie unter: https://code.fbi.h-da.de/m.bredel/datastore
- Der Datenspeicher steht auf DockerHub unter https://hub.docker.com/r/mbredel/datastore zur Verfügung. Das Docker-Image kann damit einfach über *image: mbredel/datastore:latest* in ein Docker-Compose file eingebunden werden.
- Der Sever lauscht auf folgenden Ports:
  - Thrift: 9090
  - gRPC: 8080
  - TCP+Protobuf: 6543
- Sie können den Quellcode des Datenspeichers zu Test- und Debug-Zwecken modifizieren. Beachten Sie aber, dass die Abnahme durch den Prüfer mit dem zur Verfügung gestellten Server erfolgt.
- Es liegt in Ihrer Verantwortung, dass Ihr Code im Docker-Container kompiliert und ausgeführt werden kann. Ein Debuggen durch den Prüfer wird nicht erfolgen.
- Nutzen Sie einen vernüftigen Git-Workflow: Committen und pushen Sie häufig und kleinschrittig.
- Weitere Hinweise zu den Rahmenbedinungen etc. finden Sie in Moodle.