Project Maintainer:
Andreas Kraus

Date: 12.03.2021

Testbericht

Test 1: Reihenfolge der Nachrichten-Logs im Datastore

Die erfolgreiche Koordinierung der Schreibzugriffe auf den datastore kann anhand der Reihenfolge der vom datastore geloggten "Größen" ausfindig gemacht werden:

- Da alle Datenquellen die gleichen Daten in identischen Intervallen versenden und nur minimale Unterschiede im Startzeitpunkt bestehen, muss die Anzahl identischer aufeinanderfolgender Werte immer der Anzahl der Datenquellen entsprechen.
- Bei drei Datenquellen sehen die Nachrichten zum datastore folglich so aus:
    - 555 444 666 333 (das doppete Logging des datastore's wurde vernachlässigt).

Test 2: Sendungsverfolgung anhand der Datasource-Nachrichten

Eine genauere Sendungsverfolgung und Auswertung ist mittels der logger.trace Meldungen möglich. Sie zeigen den Versand und Empfang der Nachrichten unter den Datenquellen. Hier ist die Koordinierung zwischen den Persistierungsschritten anhand der Nachverfolgung der Versand-REQUESTs und Versand-ACKNOWLEDGEs möglich. Beispiel eines solchen Austauschs:

- DataSource_0 | 10:00:51.459 [Thread-2] TRACE transmitter.TcpTransmitter - sending event '0>0>REQUEST>2>5' to datasource1:25000
- DataSource_0 | 10:00:51.491 [Thread-2] TRACE transmitter.TcpTransmitter - sending event '0>0>REQUEST>2>5' to datasource2:25000
- DataSource_1 | 10:00:51.546 [Thread-1] TRACE utility.DataSourceListener - received request event '0>0>REQUEST>2>5', (current list: [])
- DataSource_1 | 10:00:51.557 [main] TRACE transmitter.TcpTransmitter - sending event '0>0>ACKNOWLEDGE>2>5' to datasource0:25000
- DataSource_0 | 10:00:51.575 [Thread-1] TRACE utility.DataSourceListener - received acknowledgement event '0>0>ACKNOWLEDGE>2>5', (current list: [0>0>REQUEST>2>5])
- DataSource_1 | 10:00:51.593 [Thread-2] TRACE transmitter.TcpTransmitter - sending event '1>3>REQUEST>2>5' to datasource0:25000
- DataSource_0 | 10:00:51.596 [Thread-1] TRACE utility.DataSourceListener - received request event '1>3>REQUEST>2>5', (current list: [0>0>REQUEST>1>5])
- DataSource_1 | 10:00:51.598 [Thread-2] TRACE transmitter.TcpTransmitter - sending event '1>3>REQUEST>2>5' to datasource2:25000
- DataSource_2 | 10:00:51.618 [Thread-1] TRACE utility.DataSourceListener - received request event '0>0>REQUEST>2>5', (current list: [])
- DataSource_2 | 10:00:51.625 [Thread-1] TRACE utility.DataSourceListener - received request event '1>3>REQUEST>2>5', (current list: [0>0>REQUEST>2>5])
- DataSource_2 | 10:00:51.986 [main] TRACE transmitter.TcpTransmitter - sending event '0>0>ACKNOWLEDGE>2>5' to datasource0:25000
- DataSource_2 | 10:00:51.988 [Thread-2] TRACE transmitter.TcpTransmitter - sending event '2>4>REQUEST>2>5' to datasource0:25000
- DataSource_0 | 10:00:51.998 [Thread-1] TRACE utility.DataSourceListener - received acknowledgement event '0>0>ACKNOWLEDGE>2>5', (current list: [0>0>REQUEST>1>5, 1>3>REQUEST>2>5])
- DataSource_0 | 10:00:51.999 [Thread-1] TRACE utility.DataSourceListener - received request event '2>4>REQUEST>2>5', (current list: [0>0>REQUEST>0>5, 1>3>REQUEST>2>5])
- DataSource_2 | 10:00:51.998 [Thread-2] TRACE transmitter.TcpTransmitter - sending event '2>4>REQUEST>2>5' to datasource1:25000

- DataSource_2 | 10:00:52.000 [main] TRACE transmitter.TcpTransmitter - sending event '1>3>ACKNOWLEDGE>2>5' to datasource1:25000
- DataSource_1 | 10:00:52.002 [Thread-1] TRACE utility.DataSourceListener - received request event '2>4>REQUEST>2>5', (current list: [1>3>REQUEST>2>5])
- DataSource_1 | 10:00:52.004 [Thread-1] TRACE utility.DataSourceListener - received acknowledgement event '1>3>ACKNOWLEDGE>2>5', (current list: [1>3>REQUEST>2>5, 2>4>REQUEST>2>5])
- DataSource_0 | 10:00:52.040 [main] TRACE transmitter.ThriftTransmitter - RPC sending name: datasource0, size: 5
- DataSource_0 | WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
- datastore_1 | 10:00:52.113 [pool-1-thread-4] INFO d.h.f.d.mbredel.thrift.ServerHandler - Store a resource: Resource(name:datasource0, size:5).
- datastore_1 | 10:00:52.115 [pool-1-thread-4] INFO d.h.f.d.m.datastore.SimpleDataStore - Store a resource: Resource(name:datasource0, size:5).

... (Auslassung weiterer Kommunikation zwischen den Datenquellen)

- datastore_1 | 10:00:57.256 [pool-1-thread-5] INFO d.h.f.d.mbredel.thrift.ServerHandler - Store a resource: Resource(name:datasource1, size:5).
- datastore_1 | 10:00:57.257 [pool-1-thread-5] INFO d.h.f.d.m.datastore.SimpleDataStore - Store a resource: Resource(name:datasource1, size:5).

... (Auslassung weiterer Kommunikation zwischen den Datenquellen)

- datastore_1 | 10:01:02.389 [pool-1-thread-5] INFO d.h.f.d.mbredel.thrift.ServerHandler - Store a resource: Resource(name:datasource2, size:5).
- datastore_1 | 10:01:02.389 [pool-1-thread-5] INFO d.h.f.d.m.datastore.SimpleDataStore - Store a resource: Resource(name:datasource2, size:5).

Die Struktur der Nachrichten gibt folgende Werte wieder:
'1>46>REQUEST>2>3' : pid (1), logical clock count (46), Nachrichtentyp (REQUEST), Anzahl fehlender ACKs (1), Nachrichteninhalt (3).

Dieser beispielhafte Persistierungsprozess von DataSource_0 bestätigt die erfolgreiche Umsetzung des Lamport-Algorithmus', indem folgende Punkte nachvollziehbar verdeutlicht werden:

- DataSource_0 hat (zufällig, da zuerst gestartet) den geringsten logical clock count und darf aus diesem Grund zuerst in den datastore schreiben.
- Das Empfangen und Versenden von Nachrichten erhöht den logical clock count.
- Die Reihenfolge des Persitierens entspricht dem logical clock count:
    - [0>0>REQUEST>1>5], [1>3>REQUEST>2>5], [2>4>REQUEST>2>5]
      --> Datasource_0; Datasource_1; Datasource_2
    - datastore_1 | 10:00:52.113 [pool-1-thread-4] INFO d.h.f.d.mbredel.thrift.ServerHandler - Store a resource: Resource(name:datasource0, size:5).
    - datastore_1 | 10:00:57.256 [pool-1-thread-5] INFO d.h.f.d.mbredel.thrift.ServerHandler - Store a resource: Resource(name:datasource1, size:5).
    - datastore_1 | 10:01:02.389 [pool-1-thread-5] INFO d.h.f.d.mbredel.thrift.ServerHandler - Store a resource: Resource(name:datasource2, size:5).