csci-8780-rmi

Distributed String Array with Concurrency Control Using Java RMI

Project Specification

Usage

Build the Project

To build the project, execute the following commands:

```
gradle clean
gradle build
```

Functional tests

To execute functional test suite, run the following commands:

```
gradle test
```

Run the Application

To run the application, follow these steps in order:

```
gradle runRMIServer
gradle runRMIClient --console=plain
```

Optionally, you can run with custom configs for the server and client using following commands,

```
gradle runRMIServer -Pconfig=$HOME/Desktop/Work/csci-8780-rmi/server-config-example.properties
gradle runRMIClient --console=plain -Pconfig=$HOME/Desktop/Work/csci-8780-rmi/client-config-example.properties
```

Environment

```
# Java
openjdk 17.0.8.1 2023-08-24
OpenJDK Runtime Environment Homebrew (build 17.0.8.1+0)
OpenJDK 64-Bit Server VM Homebrew (build 17.0.8.1+0, mixed mode, sharing)
```

```
# Gradle

------------------------------------------------------------
Gradle 7.6
------------------------------------------------------------

Build time:   2022-11-25 13:35:10 UTC
Revision:     daece9dbc5b79370cc8e4fd6fe4b2cd400e150a8

Kotlin:       1.7.10
Groovy:       3.0.13
Ant:          Apache Ant(TM) version 1.10.11 compiled on July 10 2021
JVM:          17.0.8.1 (Homebrew 17.0.8.1+0)
OS:           Mac OS X 13.6 x86_64
```

Features

Client

- Interacts with the user and performs operations on the distributed string array.
- Communicates with the server to perform following operations:
  - Retrieve the maximum capacity of the string array.
  - Fetch elements from the array in "read-only" mode, ensuring proper read locks.
  - Fetch elements from the array in "read/write" mode, ensuring proper write locks.
  - Print strings associated with elements.
  - Concatenate strings to elements.
  - Attempt to write elements back to the server with success/failure feedback.
  - Release locks held by the client on specific elements.
  - [Additional] Retrieves the read and write locks currently held by this client
- Takes a command-line parameter for specifying the configuration file.

Server

- Manages the distributed string array with concurrency control.
- Safeguarding concurrency by managing read and write locks for array elements while supporting all the methods required by the client
- Grants or denies client requests based on read/write lock availability.

General

- **Functional Tests**: The project includes functional tests that simulate a multi-client environment, ensuring concurrency while maintaining the integrity of locks.

- **Fair Read/Write Lock Mechanism**: The server follows a fair read/write lock mechanism to manage concurrent access to array elements. This mechanism enforces the following rules:

  - If a client has a write lock for a particular array element, it also has a read lock for that element.
  - Two clients cannot simultaneously hold a write lock for the same array element.
  - Two or more clients can simultaneously hold read locks for the same element.

  - If a client has a write lock for a particular element, no other client can have a write or read lock for that element.
  - If a client has a read lock for an element, no client, including the one holding the read lock, can have a write lock for that element.

- **Automatic Lock Release**: The server supports the automatic release of read/write locks after a certain configurable timeout duration. This feature is in place to prevent deadlock or starvation scenarios. The timeout duration can be customized through server configuration files, with a default value of 10 minutes.

- **Client Lock Auto-Release**: Clients automatically release their read/write locks, if they exist, upon graceful exit. This ensures that locks are not held indefinitely, contributing to the overall system's stability and reliability.