Simulation einer verteilten Synchronisation mit einem zentralen Koordinator

David Fischer, 5CHIF

Dezember 2020

Contents

1 Einleitung

1.1 Problematik

Laut Angabe war das Ziel dieser Aufgabe eine Simulation einer verteilten Synchronisation mit einem zentalen Koordinator zu erstellen. Die Simulation soll mit einer beim Aufruf definierten Anzahl an Nodes gestartet werden.

Die folgenden Illustrationen beschreiben den Prozess, den ein Koordinator durchläuft, sobald eine Node in den kritischen Abschnitt will.

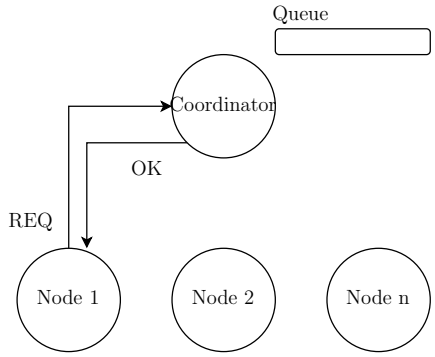

Figure 1: Node 1 sendet einen "Request" an den Koordinator. Da die Queue leer ist, bekommt Node 1 ein "OK" und betritt den kritischen Abschnitt.

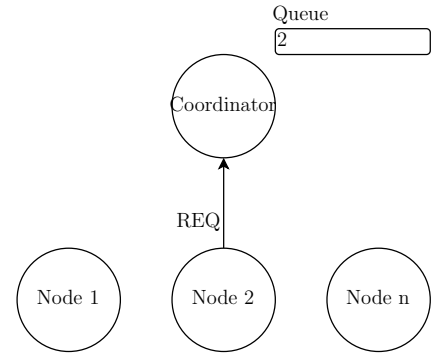

Figure 2: Node 2 sendet einen "Request" an den Koordinator. Da schon jemand im kritischen Abschnitt ist, wird Node 2 in die Queue gesetzt.

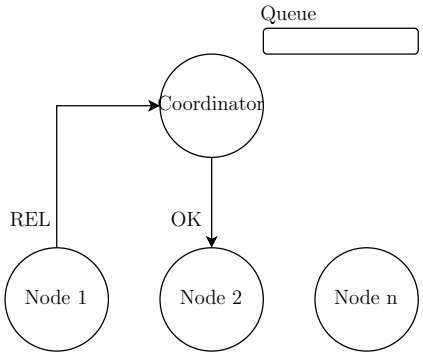

Figure 3: Node 1 verlässt den kritischen Abschnitt und sendet ein "Release" an den Koordinator. Der Koordinator sieht, dass Node 2 noch in der Queue steht, und sendet diesem damit ein "OK".

2 Implementierung

Für Variablen sowie Funktionen wurde "Snake case" benutzt. Sämtliche vorgegebene coding conditions wurden nach Möglichkeit eingehalten. Um sämtliche Teile des Programmes zu vereinfachen wurden mehrere externe Bibliotheken verwendet, diese sind in Sektion 2.2 kurz beschrieben.

2.1 Kurze Beschreibung diverser Codeblöcke

2.2 Externe Bibliotheken

2.2.1 CLI11

CLI11 [1] ermöglicht einfache Verarbeitung von Kommandozeilenargumenten mit eingebauten Möglichkeiten zum Überprüfen von den angegebenen Werten. Auf die Argumente wird in Sektion 3 näher eingegangen.

2.2.2 httplib

Eines der Kommandozeilenargumente erlaubt dem Nutzer einen lokalen HTTP Server zu öffnen, um diesen als Kommunikationsmittel zwischen dem Koordinator und den Nodes zu benutzen. Ein httplib [4] Server öffnet sich lokal auf port 5001 und stellt drei Routen zur verfügung:

/req?node_id=x Um einen "Request" an den Koordinator zu senden benutzen die Nodes diese Route. Sobald die Nodes eine Antwort mit dem Code 200 erhalten, wird das als "OK" gewertet.

/rel?node_id=x Um einen "Release" an den Koordinator zu senden benutzen die Nodes diese Route.

/get Ist eine Route, die dem Nutzer eine Einsicht in das Programm ermöglicht, ohne Einsicht in die Logs zu haben. Eine Live Demo dieser Funktionalität läuft seit längerem in einem VM Container, aufrufbar unter `https://s.konst.fish/fischer_projekt_1`.

2.2.3 tabulate

Sobald das Programm vom Nutzer mittels $Ctrl + C$ abgebrochen wird eine Tabelle mittels tabulate [3] erstellt. Diese enthält diverse Informationen die während der Programmlaufzeit gesammelt wurden. Insofern die httplib Flag gesetzt ist, kann auf die Tabelle, wie in Paragraph 2.2.2 erwähnt, jederzeit zugegriffen werden.

```
+-----------------------+--------------------+-------------------------+
| No. of Admitted Nodes | Maximum Queue Size | Total Time Spent Running|
+-----------------------+--------------------+-------------------------+
| 536748                | 9                  | 2409006s                |
+-----------------------+--------------------+-------------------------+
```

Source Code 1: Beispiel einer tabulate Tabelle

2.2.4 spdlog

Um einfaches loggen, parallel in der Konsole und in einem File zu ermöglichen wurde die spdlog [2] Bibliothek verwendet.

```
sinks.push_back(file_sink);
sinks.push_back(console_sink);
auto combined_logger = make_shared<spdlog::logger>("CombSink",
                                                   begin(sinks),
                                                   end(sinks));
```

```
// register to access it globally
spdlog::register_logger(combined_logger);

spdlog::info("Beispiel Log Eintrag");
```

Source Code 2: Registrieren eines kombinierten Loggers, um global in diesen schreiben zu können.

3 Verwendung

3.1 Kommandozeilenargumente

4 Projektstruktur

```
/
├── LICENSE
├── meson_options.txt
├── meson.build
├── README.md
├── .gitignore
├── include
│   ├── utils.h
│   ├── Node.h
│   └── Coordinator.h
├── src
│   ├── utils.cpp
│   ├── Node.cpp
│   ├── Coordinator.cpp
│   └── main.cpp
├── doc
│   ├── ausarbeitung.tex
│   ├── references.bib
│   └── ausarbeitung.pdf
└── build
```

References

[1] CLIUtils. Cli11. https://github.com/CLIUtils/CLI11, 2020.

[2] Gabi Melman. spdlog. https://github.com/gabime/spdlog, 2020.

[3] Pranav. tabulate. https://github.com/p-ranav/tabulate, 2020.

[4] yhirose. cpp-httplib. https://github.com/yhirose/cpp-httplib, 2020.