HAW HAMBURG

AUSARBEITUNG
Tristan Ropers

Lamports Algorithmus für verteilten gegenseitigen Ausschluss

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Tristan Ropers

Lamports Algorithmus für verteilten gegenseitigen Ausschluss

Ausarbeitung eingereicht im Rahmen des Moduls "Verteilte Systeme"
im Studiengang *Bachelor of Science Technische Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Martin Becke

Eingereicht am: 31. März 2021

Tristan Ropers

Thema der Arbeit

Lamports Algorithmus für verteilten gegenseitigen Ausschluss

Stichworte

Gegenseitiger Ausschluss, Lamport, Verteilte Systeme

Kurzzusammenfassung

Das Ziel der vorliegenden Arbeit ist die Umsetzung des Lamport-Algorithmus [1] für wechselseitigen Ausschluss in einem verteilten System mit anschließender Bewertung der Leistungsfähigkeit des Algorithmus. Für die Umsetzung und Evlauierung des Algorithmus wurde eine RPC-Architektur entwickelt und der Lamport-Algorithmus in diese eingebettet. TODO: Ergebnis

Tristan Ropers

Title of Thesis

Mutual exclusion in distributed systems using Lamports algorithm

Keywords

Mutual exclusion, Lamport, distributed systems

Abstract

The goal of this assignment is the implementatin of the lamports algorithm [1] for mutual exclusion in distributed systems with an evaluation of said algorithm. In order to achieve this, a RPC-architecture has been implemented in which the lamports algorithm was embedded. TODO: Result

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Anforderungsanalyse

In der Aufgabenstellung sind Anforderungen an die RPC-Architektur formuliert. Zunächst werden diese analysiert und hier zusammengefasst. Die Anforderungen umfassen die Transparenzziele, Skalierung [2]) sowie Anforderungen an die Umsetzung und Architektur.
Es wurde ein hoher Grad an Transparenz gefordert. Die Transparenzziele umfassen die Access-Transparency, Location-Transparency, Relocation-Transparency, Migration-Transparency, Replication-Transparency, Concurrency-Transparency und Failure-Transparency [2]. Die Skalierung beschränkt sich administrativ auf einen Administrator, der auch gleichzeitig Benutzer des Systems ist sowie geographisch auf einen Rechner, auf dem alle Nodes [2] über das Loopback-Interface der Netzwerkkarte miteinander kommunizieren. Somit entfällt die Concurrency-Transparency, da nur ein Nutzer am System beteiligt ist sowie die Location-Transparency, Relocation- und Migration-Transparency, da alle Nodes über das Loopback-Interface kommunizieren, welches auf eine Netzwerkkarte beschränkt ist. Die Replication-Transparency ist in diesem Zusammenhang uninteressant, da jede Node im System eindeutig identifizierbar ist,was Replikationen der Nodes für die Funktionsweise der verwendeten Algorithmen ausschließt.

Desweiteren ist ein Minimalset an Funktionen an die RPC-Architektur gefordert:

- void register(int id)
- void welding()
- void setStatus(int status)

Es soll kein zentrales System, bis auf Erfassung von Daten zu Experimentabläufen, am Gesamtsystem beteiligt sein. Alle beteiligten Nodes (Roboter) sollen sich auf einen Zyklus einigen, in dem immer drei Roboter nacheinander (Reihenfolge durch Lamport) schweißen. Insgesamt soll kein Roboter mehr als drei Schweißpuntke mehr als ein anderer gesetzt haben. Alle Roboter sollen am Ende eines Experiemntablaufs mindestens 20 Schweißpunkte gesetzt haben. In 99% soll ein Roboter nach einem Schweißvorgang weiterhin betriebsbereit sein. Im Umkehrschluss ist ein Roboter in 1% der Fälle nach einem Schweißvorgang nicht mehr betriebsbereit.
Zur Auswertung und Beobachtung des Ablaufs soll jeder Roboter seinen Ablauf loggen. Daraus ergeben sich folgende Anforderungen:

Requirement	Registrierung eines Nodes im System
Beschreibung	Ein Node kann sich im System mit allen anderen Nodes bekannt machen.
Eingaben	id: int; eindeutige ID für den Node
Ziel	Die Node hat sich mit allen anderen im System bekanntgemacht.
Vorbedingung	Die Node ist hochgefahren und betriebsbereit.
Nachbedingung	Die Node kennt alle anderen Nodes im System und alle anderen Nodes kennen die sich registrierende Node.

Tabelle 1.1: Requirement Registrierung

Requirement	Ein Roboter versucht zu schweißen
Beschreibung	Ein Roboter einer Node möchte Zugriff auf die Ressource haben und einen Schweißauftrag ausführen. Sobald die Ressource verfügbar ist, soll der Roboter seinen Schweißauftrag ausführen.
Eingaben	/
Ziel	Der Roboter hat einen Schweißvorgang durchgeführt.
Vorbedingung	Die Node sowie der Roboter der Node ist hochgefahren und betriebsbereit.
Nachbedingung	Der Roboter hat einen Schweißvorgang abgeschlossen und geht entweder in einen Fehlerzustand (in 1% der Fälle) oder bleibt betriebsbereit.

Tabelle 1.2: Requirement Schweißen (Welding)

Requirement	Bestimmung eines Zyklus
Beschreibung	Es muss sichergestellt sein, dass immer genau drei Roboter pro Zyklus einen Schweißauftrag ausführen.
Eingaben	/
Ziel	Ein Zyklus wird bestimmt und drei Roboter können schweißen.
Vorbedingung	Alle am Experiment teilnehmenden Nodes sowie dazugehörige Roboter sind hochgefahren und betriebsbereit.
Nachbedingung	Ein Zyklus wurde bestimmt und drei Nodes haben den Schweißauftrag erhalten.

Tabelle 1.3: Requirement Bestimmung eines Zyklus

Requirement	Logging der Prozessabläufe
Beschreibung	Jede Node muss seinen Zustand und die Abläufe loggen.
Eingaben	logText: String; Event oder Zustand
Ziel	Ein Event oder Zustand des Nodes wurde geloggt.
Vorbedingung	Die Node sowie der Roboter der Node ist hochgefahren und betriebsbereit.
Nachbedingung	Der Zustand der Node oder das Event wurden erfolgreich geloggt.

Tabelle 1.4: Requirement Logging

2 Design & Architektur

3 Implementierung

4 Leistungsanalyse

5 Diskussion

6 Fazit

Literaturverzeichnis

[1] Lamport, Leslie: Time, Clocks, and the Ordering of Events in a Distributed System. In: *Communications of the ACM* (1978)

[2] Tanenbaum, Andrew S. ; Steen, Marten van: *Distributed Systems 3rd edition*. Maarten van Steen, 2018. – URL https://www.distributed-systems.net/index.php/books/ds3/. – ISBN 978-90-815406-2-9

A Anhang

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original