Team: {19, ["Björn Eberhardt", "Dieter Pisarewski"]}

Aufgabenaufteilung:

Björn Eberhardt und Dieter Pisarewski:

Gemeinsame Entwurf und Implementierung

Quellenangaben:

- http://www.erlang.org/documentation/doc-5.1/doc/getting_started/getting_started.html
- Folien aus der Vorlesung
- Beispieldateien aus der Vorlesung

Begründung für Codeübernahme: Es wurde kein Quellcode aus anderen Projekten übernommen.

Bearbeitungszeitraum:

- 21.03.2013, 5 Stunden
- 23.03.2013, 4 Stunden
- 24.03.2013, 3 Stunden
- 26.03.2013, 6 Stunden
- 28.03.2013, 5 Stunden
- 01.04.2014, 2 Stunden
- 02.04.2013, 6 Stunden

Aktueller Stand:

Fertig

Änderungen im Entwurf:

- Neue Anfrage vom QueueManager an DeliveryQueue im Kommukationsdiagramm.
- Beschreibung der Realisierung des Clients.
- Ingorieren von bereits abgearbeiteten Nachrichten

Client

Der Client versendet Nachrichten an den Server und holt sich die neuesten Nachrichten wieder ab.

Architektur

- Der Client wird logisch in den Redakteur- und in den Leseclient unterteilt.
- Die beiden logischen Clients wechseln sich sequentiell in der Ausführung ab.
- Mit kurzen Zeitabständen wird eine fortlaufende Nummer vom Server angefragt und eine entsprechende Nachricht versendet.

- Die Nachricht enthält den Rechnernamen, die Praktikumsgruppe, Teamnummer sowie eine Systemzeit und ggf. andere Informationen.
- Die Zeitabstände werden nach 5 versandten Nachrichten angepasst; zufällig um 50% vergrößert oder verkleinert, mindestens jedoch um 1 Sekunde.
- Die Zeitabstände dürfen nicht kleiner als 1 Sekunde sein.
- Eine 6. Nachrichtennummer wird angefordert und im Log protokolliert, ohne dass hierfür eine Nachricht versendet wird.
- Der Client merkt sich die für den Versand verwendete Nachrichtennummern.
- Der Leseclient empfängt nun alle neuen Nachrichten vom Server, bis dieser keine weitere Nachricht mehr ankündigt.
- Eigene Nachrichten werden zusätzlich mit einer Zeichenfolge (*******) gekennzeichnet.
- Die Nachrichten werden bei der Ausgabe zusätzlich mit einem Zeitstempel versehen.
- Nach Ablauf einer eingestellten Lebenszeit wird der Client beendet.

Konfiguration

Folgende Werte sind konfigurierbar:

- clients, die Anzahl der zu startenden Clients.
- life_time, die Zeit, nachdem sich der Client von selbst beendet.
- servername, der Name des Servers, zu dem sich verbunden wird.
- intervall, die Zeit in Millisekunden, nach der sich der Client beendet.
- anzahl_schritte, die Anzahl der Nachrichten (in diesem Entwurf 5), die hintereinander gesendet werden.
- log_datei, der Name der Logdatei für die Programmausgaben.

Beim Starten eines Clients werden der Servername und die Clientnummer übergeben und nicht aus der Configdatei gelesen.

Realisierung

Die beiden logischen Einheiten, der Redakteur- und der Leseclient wechseln sich sequentiell ab, sodass das Senden und Abfragen der Nachrichten nacheinander erfolgt.

Der Programmablauf kann daher in einem einzelnen Prozess erfolgen und die oben genannten Punkte Schritt für Schritt abarbeiten. Lediglich die Funktion zum Sammeln von Nachrichtennummern wird als eigener Prozess realisiert und vom Clientprozess verwendet.

Die Funktion zum Sammeln von Nachrichtennummern verwaltet eine Liste der gesendeten und empfangenen Nachrichtennummern und verrät, ob eine empfangene Nachricht vom eigenen Redakteur versendet wurde.

Die Umsetzung des Clients erfolgt analog zur Architekturbeschreibung. Für das normale Versenden der Nachrichten wird sendeNachrichten aufgerufen und die gewünschte Anzahl Nachrichten werden versendet. Für den Empfang der Nachrichten wird eine rekursive Funktion gestartet, die sich beendet, sobald der Server keine weiteren Nachrichten in der Queue meldet. Somit laufen im Client 2

Prozesse(Nachrichtensammler und Lese-, Redakteurclient). Die restlichen Funktionen können sequentiell im Hauptprogramm implementiert werden.

Server

Der Server verwaltet die Nachrichten, die vom Redakteurclient gesendet werden und liefert diese in richtiger Reihenfolge den Leseclients aus.

Architektur

- Der Server liefert dem Redakteurclient auf Anfrage eine fortlaufende Nummer für die Nachricht.
- Die dem Server zugesendeten Nachrichten werden über eine HoldbackQueue und eine DeliveryQueue verwaltet, um die Reihenfolge einzuhalten.
- Der Server protokolliert am Ende der Nachricht den Eingangszeitstempel beim Eintritt in die Holdback- und DeliveryQueue.
- Die Größe der DeliveryQueue ist beschränkt, und alte Nachrichten werden bei Überschreitung der Größe entfernt.
- Die HoldbackQueue wird benötigt, sobald die Nachrichten nicht in Reihenfolge eintreffen und eine Lücke entsteht.
- Besteht zwischen der ersten Nachricht der HoldbackQueue und der letzten Nachricht der DeliveryQueue keine Lücke, wird die Nachricht aus der HoldbackQueue in die DeliveryQueue geschoben. Dies erfolgt, solange die Liste nicht leer ist und keine Lücke entsteht.
- Überschreitet die Größe der HoldbackQueue die Hälfte der Maximalgröße für die DeliveryQueue, besteht eine Lücke. Die wird durch eine Fehlernachricht, die direkt in die DeliveryQueue geschoben wird und die Nummer der ersten Nachricht der HoldbackQueue minus 1 trägt, geschlossen und die HoldbackQueue wird weiter abgearbeitet.
- Erfolgt in einer vorgegebenen Zeit keine weitere Nachrichtenabfrage von einem Client, wird der Server terminiert.

Konfiguration

Folgende Werte sind konfigurierbar:

- lifetime, die Zeit, bis der Server sich nach der letzten Abfrage beendet.
- clientlifetime, die Zeit, nach der ein Client vergessen wird.
- servername, an dem die Nachrichten gesendet werden.
- dlq_limit, die Maximalgröße der DeliveryQueue.
- log_datei, der Name der Logdatei für die Programmausgaben.

Realisierung

Wir haben unseren Server in folgende Einheiten aufgeteilt: **QueueManager**, **ClientManager** und **Sender** sowie 2 **Queues** zum Speichern der Nachrichten.

Die zentrale Einheit (der Server selber) übernimmt die Kommunikation mit den Clients und die Verarbeitung der drei in der Schnittstelle beschriebenen Nachrichten.

- Auf getmsgid wird eine fortlaufende Zahl zurückgeliefert. Dies erledigt der Server selbst, ohne auf die anderen Einheiten zuzugreifen.
- Auf getmessages wird die älteste ungelesene Nachricht in der DeliveryQueue an den Client gesendet. Die Ausführung dieses Vorgangs wird vom **Sender** übernommen.
- Auf dropmessage wird die Nachricht in der HoldbackQueue gespeichert. Anschließend wird die HoldbackQueue und die DeliveryQueue verwaltet. Diese Aufgabe wird vom **QueueManager** übernommen.

QueueManager ist verantwortlich für das Verwalten der eingehenden Nachrichten in die HoldbackQueue und Übertragung dieser Nachrichten in die DeliveryQueue.

Jede erhaltene Nachricht kommt zuerst in die HoldbackQueue. Nach dem Speichern der Nachricht in die HoldbackQueue wird zunächst die auf die neueste Nachricht aus der DeliveryQueue folgende Nachricht gesucht. In diesem Fall wird diese von der HoldbackQueue in die DeliveryQueue geschoben. Ist die DeliveryQueue nun größer als die festgelegte Maximalgröße, wird außerdem die älteste Nachricht gelöscht. Diese Suche wird wiederholt, bis die HoldbackQueue leer ist, oder eine Lücke gefunden wird.

Anschließend prüft der QueueManager die Größe der HoldbackQueue. Überschreitet diese die Hälfte der Maximalgröße der DeliveryQueue, geht der QueueManager davon aus, dass in der HoldbackQueue eine Lücke entstanden ist. Dann füllt er diese Lücke mit einer Fehlernachricht und schiebt sie in die DeliveryQueue mit der größten Nachrichtennummer in der Lücke.

Sender dient dazu, Nachrichten aus der DeliveryQueue an den Lese-Client zu senden und die Informationen (letzte gesendete Nachrichtennummer, letzte Zugriffszeit) über den Lese-Client zu aktualisieren.

Er fragt die letzte Nachrichtennummer, die dem Client gesendet wurde, beim ClientManager ab.

Dabei überprüft der Sender, ob der Client dort schon existiert. In diesem Fall prüft er den gespeicherten Zeitstempel und setzt die letzte Nachrichtennummer des Clients zurück, wenn die vergangene Zeit vom Zeitstempel bis zur aktuellen Zeit ein in Konfiguration vorgegebenes Zeitintervall überschreitet. Danach wird der Zeitstempel des Clients aktualisiert und die Nachrichtennummer auf die kleinste Nachrichtennummer in der DeliveryQueue gesetzt.

Dann holt er sich die Nachrichten aus der DeliveryQueue, wählt die benötigte Nachricht aus und sendet sie an den Client.

ClientManager verwaltet eine Liste von registrierten Lese-Clients mit zugehörigen Informationen (ClientPid, letzte gesendete Nachrichtennummer, letzte Zugriffszeit).

Ein Client landet in dieser Liste, wenn er Nachrichten vom Server anfordert. Bevor er dort gespeichert wird, prüft der Sender, der die oben genannte Aktion initiiert, ob der Client dort schon existiert.

HoldbackQueue und **DeliveryQueue** haben eine gemeinsame generische Implementierung. Wir realisieren die Queues mit einfachen Lists, da sie für die Speicherung von geordneten Mengen ausreichen, und diese lassen sich einfach mit Funktionen aus der Standard Library verwalten. Sie unterstützen folgende Aktionen:

- Einfügen einer Nachricht ans Ende der Queue
- Ankleben von mehreren Nachrichten ans Ende der Queue
- Holen aller Nachrichten aus der Queue als eine Liste

Kommunikationsdiagramm