Team: <08>, <(Porep, Lars), (Niebsch, Oliver)>

Aufgabenaufteilung:

1. Aufgabe 2, komplett,
   <Dateien, die komplett/zum Teil von Teammitglied 1 implementiert/bearbeitet wurden>

Quellenangaben:
erlang.org Dokumentation

Begründung für Codeübernahme:

Bearbeitungszeitraum:
(Gemeinsame Bearbeitungszeit)
06.05.2015 - 3 Stunden

(Bearbeitungszeit Lars Porep)
06.05.2015 - 3 Stunden
19.05.2015 - 6 Stunden

(Bearbeitungszeit Oliver Niebsch)
12.05.2015 - 2 Stunden
13.05.2015 - 2 Stunden
16.05.2015 - 4 Stunden
19.05.2015 - 3 Stunden

Aktueller Stand: Alles ist implementiert und getestet.

Skizze: *siehe nächste Seite*

Entwurf:

Es soll ein Programm entwickelt werden um die verteilte Berechnung eines Algorithmus am Beispiel des größten gemeinsamen Teilers zu veranschaulichen.

Das Programm besteht aus folgenden Modulen:

- Koordinator: Organisiert die Berechnung des ggT mithilfe mehrerer ggT-Prozesse
- Starter: Initialisiert ggT Prozesse.
- ggT-Prozess: Sucht nach dem ggT zweier Zahlen. Über Kommunikation mit anderen ggT Prozessen wird so der ggT von mehreren Zahlen verteilt ermittelt.
- Namensdienst: Liegt bereits als "nameservice.beam"-Datei vor.

Zudem gibt es zwei Konfigurationsdateien. Eine für den Koordinator und eine für den Starter bzw. die ggT-Prozesse.

koordinator.cfg

- arbeitszeit: Verzögerungszeit der Berechnung in den ggT-Prozessen
- termzeit: Zeit ohne Erhalt von Nachrichten, nach der ein ggT-Prozess die Terminierungsabstimmung anstößt
- ggtprozessnummer: Anzahl der ggT-Prozesse, die pro Starter erzeugt werden
- nameservicenode: Node, auf der der Namensdienst läuft
- nameservicename: Name des Namensdienst Prozesses
- koordinatorname: Name des Koorinators, mit dem er sich beim Namensdienst anmeldet
- quote: Abstimmungsquote in % aller ggT-Prozesse für die Terminierungsabstimmung
- korrigieren: Flag, ob der Koordinator bei einer falschen Terminierung eingreifen soll

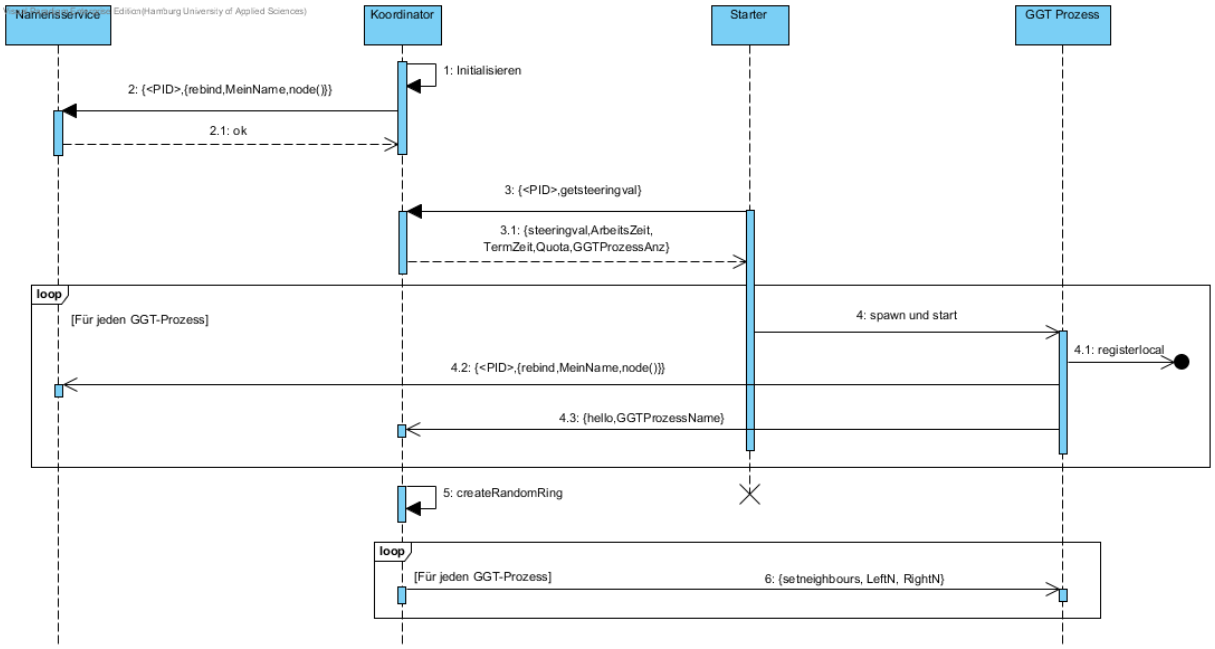
ggt.cfg

- praktikumsgruppe: Nummer der Praktikumsgruppe
- teamnummer: Nummer des Teams im Praktikum
- nameservicenode: Node, auf der der Namensdienst läuft
- nameservicename: Name des Namensdienst Prozesses
- koordinatorname: Name des Koorinators, mit dem er sich beim Namensdienst anmeldet

Die Berechnung erfolgt in vier Phasen: Initialisierung, Berechnung Terminierung und Beendigung.

1. Initialisierung
    a. Der Koordinator und die Startet initialisieren sich und lesen benötigte Informationen aus der jeweiligen Konfigurationsdatei.
    b. Die Starter holen sich die Informationen zum Starten der ggT-Prozesse von Koordinator, starten die ggT-Prozesse und beenden sich dann.
    c. Jeder gestartete ggT-Prozess meldet sich beim Koordinator an. Dieser ordnet alle in einer fiktiven Ringstruktur an und sendet jedem ggT-Prozess seinen rechten und linken Nachbarn.

Ablauf als Diagramm:

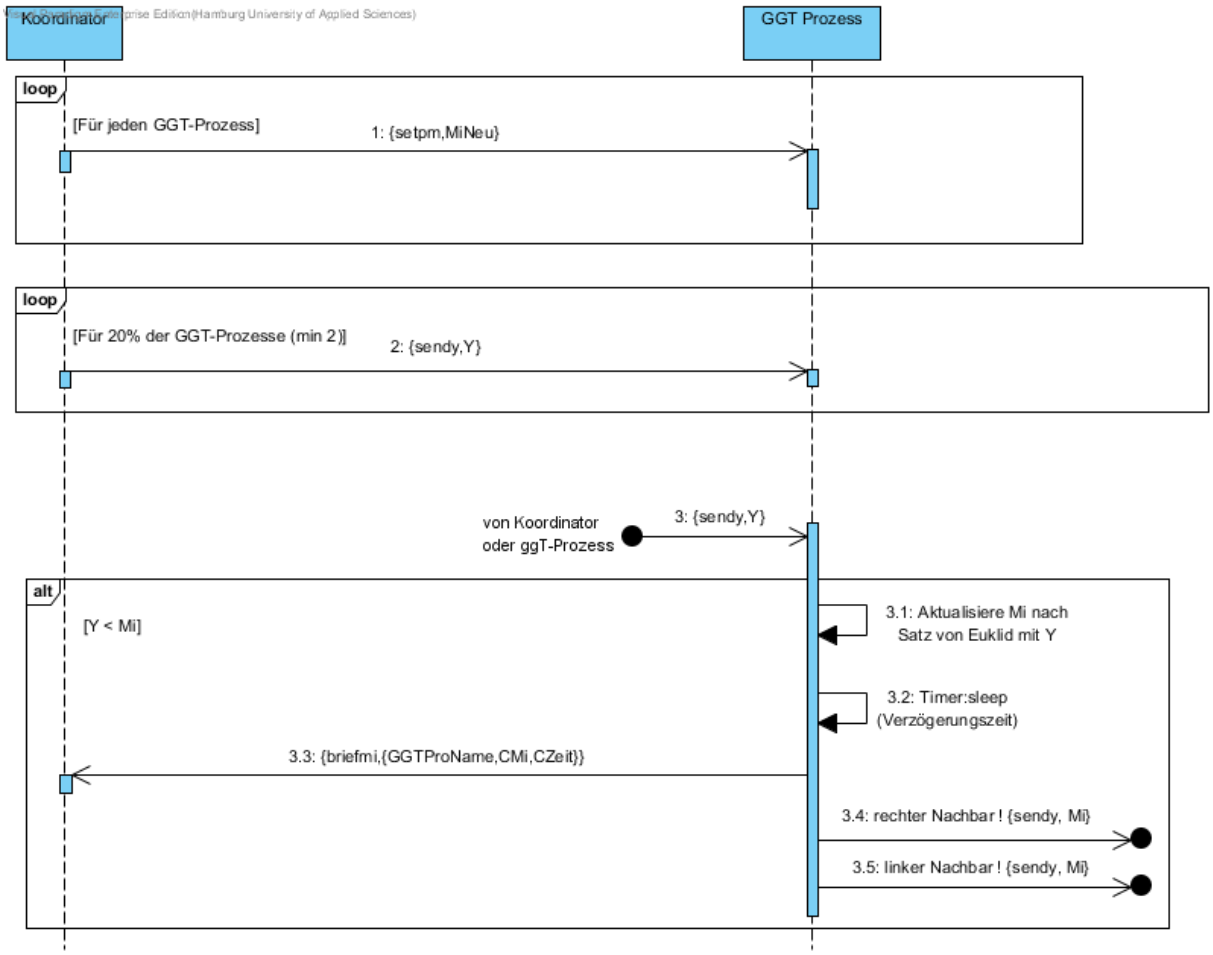


<u>Hinweise:</u>

- Die Quota, die der Koordinator dem Startet sendet, ist die Abstimmungsquote als ganze Zahl. Sie berechnet sich aus der prozentualen Abstimmungsquote * Anzahl der zu erwartenden ggT-Prozesse (Anzahl bekannter Starter * Anzahl ggT pro Starter).
- Erhält der Koordinator den manuellen Befehl „step“, ignoriert er eigehende Anfragen von Startern sowie Meldungen von ggT-Prozessen und springt direkt zu Schritt 5 (siehe Diagramm).

2. Berechnung
    a. Der Koordinator sendet jedem ggT-Prozess seine Anfangszahl. Zur Berechnung der Zahlen aus einem gegebenen ggT steht in der werkzeug.erl die Funktion bestimme_mis/2 zur Verfügung.
    b. Der Koordinator wählt 20% der ggT-Prozesse (mindestens 2) aus und sendet ihnen eine weitere Zahl, um die Berechnung des ggTs zu Starten. Die Zahlen werden wieder mit werkzeug:bestimme_mis/2 erzeugt.
    c. Wenn ein ggT-Prozess per sendy eine Zahl Y erhält, aktualisiert er seine Zahl (Mi) nach dem Satz von Euklid mit dem Y und informiert seine Nachbarn.

Ablauf als Diagramm:



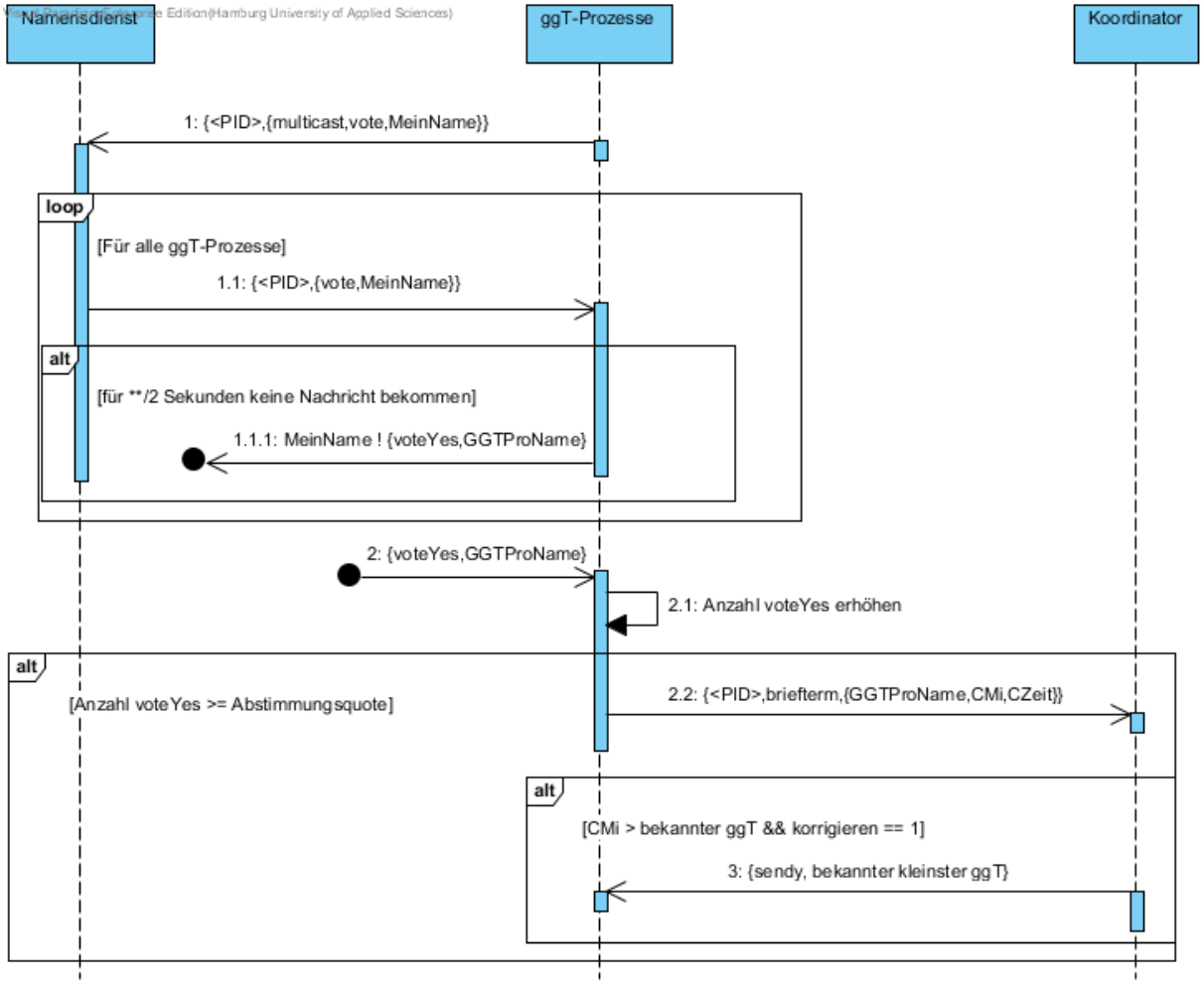
<u>Hinweise:</u>

- Die Berechnungsphase wird gestartet, wenn der Koordinator den manuellen Befehl „{calc, WggT}“ erhält. WggT ist dabei der gewünschte ggT, welcher bei der Berechnung raus kommen soll.
- Der Algorithmus zur Aktualisierung des eigenen Mi mit Y lautet:

  neuesMi = ((Mi-1) mod Y) + 1

3. Terminierung
    a. Wenn ein ggT-Prozess für termzeit Sekunden (siehe cfg) keine Zahl mehr bekommen (per sendy oder sendpm), startet er eine Terminierungsabstimmung per Multicast an alle ggT-Prozesse über den Namensdienst.
    b. Erhält ein ggT-Prozess eine Anfrage zur Terminierungsabstimmung, antwortet er mit ja, wenn er seit termzeit/2 Sekunden keine Zahl mehr bekommen hat. Ansonsten ignoriert er die Anfrage.
    c. Hat der ggT-Prozess, der die Abstimmung gestartet hat, mindestens so viele Ja-Antworten bekommen, wie die Abstimmungsquote, sendet er dem Koordinator sein erreichtes ggT und notiert die Anzahl der erfolgreichen Terminierungen im log.

Ablauf als Diagramm:



<u>Hinweise:</u>

- Nach einer erfolgreichen Terminierungsabstimmung setzt der ggT-Prozess nur seinen Timer für die termzeit zurück, sodass nicht direkt wieder eine neue Abstimmung gestartet wird. Ansonsten arbeitet er normal weiter.

4. Beendigung
   Über den manuellen Befehl „kill“ an den Koordinator wird die Beendigungsphase gestartet.

   Koordinator:
   a. Der Koordinator sendet an alle ggT-Prozesse ein kill-Komando.
   b. Er meldet sich mit "unbind" vom Namensdienst ab.
   c. Beenden des Koordinator-Prozesses.

   ggT-Prozesse:
   Laufen bis sie in irgendeiner Phase ein kill-Kommando erhalten.
   Bei Eintreffen eines kill-Kommandos:
   a. Melden sie sich mit "unbind" vom Namensdienst ab.
   b. Beenden sich selbst.

Schnittstellen

Namensdienst:

- Nach einem ping auf den Knoten des Namensdienstes erhält man die Adresse durch:
  Nameservice = global:whereis_name(nameservice)
- Binden eines Dienstes (einmaliges binden):
  ```
  register(meindienst,From),
  Nameservice ! {From,{bind,meindienst,node()}},
  receive ok -> io:format("..bind.done.\n");
            in_use -> io:format("..schon gebunden.\n")
   end,
  ```
- Rebinden eines Dienstes (erstmaliges oder wiederholtes binden):
  ```
  register(meindienst,From),
  Nameservice ! {From,{rebind,meindienst,node()}},
  receive ok -> io:format("..rebind.done.\n")
  end,
  ```
- Lookup für einen Dienst:
  ```
  Nameservice ! {From,{lookup,meindienst}},
  receive
       not_found -> io:format("..meindienst..not_found.\n");
       {pin,{Name,Node}} -> io:format("...ok: {~p,~p}.\n",[Name,Node])
  end,
  ```
- Unbind eines Dienstes:
  ```
  Nameservice ! {From,{unbind,meindienst}},
  receive
       ok -> io:format("..unbind..done.\n")
  end,
  unregister(meindienst),
  ```
- Multicast an alle registrierten Einheiten:
  ```
  Nameservice ! {From,{multicast,vote,meinname}},
  receive
       {From,{vote,meinname}} ->...From ! {voteYes,Clientname}…,
  end
  ```
- Reset des Namensdienst:
  ```
  Nameservice ! {From,reset},
  receive
       ok -> do_something_else,
  end
  ```

Koordinator:

- {From:,getsteeringval} Die Anfrage nach den steuernden Werten durch den Starter Prozess (From ist seine PID).
- {hello,Clientname}: Ein ggT-Prozess meldet sich beim Koordinator mit Namen Clientname an (Name ist der lokal registrierte Name, keine PID!).
- {briefmi,{Clientname,CMi,CZeit}}: Ein ggT-Prozess mit Namen Clientname (keine PID!) informiert über sein neues Mi CMi um CZeit Uhr.
- {From,briefterm,{Clientname,CMi,CZeit}}: Ein ggT-Prozess mit Namen Clientname (keine PID!) und Absender From (ist PID) informiert über über die Terminierung der Berechnung mit Ergebnis CMi um CZeit Uhr.
- reset: Der Koordinator sendet allen ggT-Prozessen das kill-Kommando und bringt sich selbst in den initialen Zustand, indem sich Starter wieder melden können.
- step: Der Koordinator beendet die Initialphase und bildet den Ring. Er wartet nun auf den Start einer ggT-Berechnung.
- prompt: Der Koordinator erfragt bei allen ggT-Prozessen per tellmi deren aktuelles Mi ab und zeigt dies im log an.
- nudge: Der Koordinator erfragt bei allen ggT-Prozessen per pingGGT deren Lebenszustand ab und zeigt dies im log an.
- toggle: Der Koordinator verändert den Flag zur Korrektur bei falschen Terminierungsmeldungen.
- {calc,WggT}: Der Koordinator startet eine neue ggT-Berechnung mit Wunsch-ggT WggT.
- kill: Der Koordinator wird beendet und sendet allen ggT-Prozessen das kill-Kommando.

Starter:

- {steeringval,ArbeitsZeit,TermZeit,Quota,GGTProzessnummer}: die steuernden Werte für die ggT-Prozesse werden im Starter Prozess gesetzt; Arbeitszeit ist die simulierte Verzögerungs-zeit zur Berechnung in Sekunden, TermZeit ist die Wartezeit in Sekunden, bis eine Wahl für eine Terminierung initiiert wird, Quota ist die konkrete Anzahl an notwendigen Zustimmungen zu einer Terminierungsabstimmung und GGTProzessnummer ist die Anzahl der zu startenden ggT-Prozesse.

ggT-Prozess:

- {setneighbors,LeftN,RightN}: die (lokal auf deren Node registrieten und im Namensdienst registrierten) Namen (keine PID!) des linken und rechten Nachbarn werden gesetzt.
- {setpm,MiNeu}: die von diesem Prozess zu berabeitenden Zahl für eine neue Berechnung wird gesetzt.
- {sendy,Y}: der rekursive Aufruf der ggT Berechnung.
- {From,{vote,Initiator}}: Wahlnachricht für die Terminierung der aktuellen Berechnung; Initiator ist der Initiator dieser Wahl (Name des ggT-Prozesses, keine PID!) und From (ist PID) ist sein Absender.
- {voteYes,Name}: erhaltenes Abstimmungsergebnis, wobei Name der Name des Absenders ist (keine PID!).
- {From,tellmi}: Sendet das aktuelle Mi an From (ist PID): From ! {mi,Mi}. Wird vom Koordinator z.B. genutzt, um bei einem Berechnungsstillstand die Mi-Situation im Ring anzuzeigen.
- {From,pingGGT}: Sendet ein pongGGT an From (ist PID): From ! {pongGGT,GGTname}. Wird vom Koordinator z.B. genutzt, um auf manuelle Anforderung hin die Lebendigkeit des Rings zu prüfen.
- kill: der ggT-Prozess wird beendet.