Medieninformatik(B): Funktionale und Reaktive Programmierung mit Scala, Prof. Knabe, 2014-12-08

Aufgabe `ActorThread`(v2): Speicheraufwand von Actors und Threads, Actor-Nutzung

a) Speicheraufwand

Finden Sie heraus, wie viele Java-Threads bzw. Akka-Actors Sie mit der gleichen Menge an Arbeitsspeicher erzeugen können.

Wie erklären Sie das unterschiedliche Verhalten bei hoher Thread- bzw. Aktorenanzahl?

Beim Erzeugen von Java-Threads müssen Sie darauf achten, dass Sie denen ein `Runnable` übergeben, welches nicht sofort zu Ende geht. Da wir hier nicht den CPU-Verbrauch messen wollen, legt sich der Thread am besten gleich für 1000 Sekunden schlafen mittels `Thread.sleep(1000*1000)`.

Ein Akka-Aktor startet sofort mit der Ausführung seines Klassenrumpfes und „wartet“ danach in seiner `receive`-Methode auf eine Nachricht, ohne dass Sie dies programmieren müssen.

b) Actor-System erweitern

Im Unterricht wurde ein ActorSystem „PingPong“ präsentiert, in dem ein `Thrower`-Actor `Ping`-Nachrichten an einen `Reflector`-Actor sendet, die dieser jeweils mit `Pong`-Nachrichten beantwortet.

Erweitern Sie dieses System zu einem Ping-Pang-Pong-Spiel mit 3 Aktoren.
Der `Thrower` soll `Ping`-Nachrichten an einen `Forwarder` senden.
Dieser soll für jede `Ping`-Nachricht eine `Pang`-Nachricht mit derselben `id` an einen `Returner` senden.
Dieser soll für jede `Pang`-Nachricht eine `Pong`-Nachricht mit derselben `id` an den `Thrower` senden.

Diese mehrstufige Erledigung einer Aufgabe mittels Weiterleitung an andere Aktoren unter Weitergabe der jeweiligen `id` ist typisch für komplexe Aktorensysteme.