Nebenläufige Programmierung SoSe 2015
Universität des Saarlandes

Studenten: Jens Heinen (2542182)
Lukas Schaal (2539218)
Christoph Rosenhauer (2549220)

Milestone 1

Test test test

- Notiz: Max Wert von Initialisierung wird nie überschritten!

1 Allgemeines & Überblick

Die Graphenstruktur wird insgesamt in m gröbere Spalten unterteilt, wobei m als steuerbarer Parameter implementiert werden kann. Zusätzlich werden sog. Synchronisationsgrenzen implementiert, die angeben, nach wie vielen Iterationsschritten die Akkumulatoren an den Spaltenübergängen synchronisiert werden. Dieser Parameter soll ebenfalls von außen steuerbar sein.

Klassenstruktur: Ein Graph wird dynamisch aufgebaut. Dafür gibt es eine Klasse *klasse*. Ein Knoten entsteht, sobald sein Wert positiv wird. Die Knoten werden in einer Oberklasse *Oberklasse* zusammengesetzt. Zur spaltenweise Abarbeitung wird es eine Klasse *Column?* geben. Jede Column bekommt nach Erstellung einen Thread zugewiesen, der die Berechnungen in jeder Iteration übernimmt.

2 Nebenläufige Probleme

Hier gehen wir folgenden Weg:

- Die Synchronisation soll anhand eines Parameters erfolgen. Falls diese Bedingung erfüllt ist, werden mitels Shared Memory die Daten der Akkumulatoren ausgetauscht. Zwei oder mehrere Zyklisches Verhalten bei seltenem Propagieren? Wie handhaben ? -> Testen ob Akkumulatorgröße für Synchronisation von Bedeutung ist.
- Wie wird die Synchronisationsbedingung i = k (Iterationsstadien unterschiedlicher Spalten) sichergestellt?
- Globale / Lokale Konvergenz: alle Threads erreichen ein Maximum an Iterationen ? Alle Spalten bekommen eine eigene Approximationsgrenze ?

3 Verwendete Datenstrukur

Der gegebene Graph ist ein $n \times m$ Graph. Dabei wird das Objekt intern auch in m Spalten unterteilt. Die Anzahl der operierenden Threads wird über einen Parameter gegeben. Ein Thread bekommt dabei einen Task übergeben und bearbeitet nach diesem Task eine oder mehrere Spalten des Graphen. Die Threads werden mittels eines Threadpools / Scheduler den Spalten zugewiesen. Die Tasks beinhalten lokale Austausch, Synchronisation zwischen mehreren Spalten und globale Konvergenz. Jede Spalte besteht aus ihren Knoten und Vektoren der link und rechtbenachtbarten Akkumulatoren. Ein Knoten ist ein Objekt, das nur seinen aktuellen Wert kennt. Die Übergangsrate ist in der GraphInfo Klasse enthalten.

4 Konvergenz

Die lokale Konvergenz innerhalb einer Spalte wird nach festgelegten Iterationsschritten (abhängig von Parameter) durchgeführt. Im Falle einer erkannten lokalen Konvergenz einer Spalte, wird mit den benachbarten Spalten synchronisiert. Falls nicht, wird nach x Iterationen definitiv synchronisiert, wobei x als Input kommt. Globale Konvergenz erfolgt implizit durch lokale Konvergenz. 2 Synchronisationsarten: 1 festgelegte nach Parameter, eine bei Konvergenz lokal. Wenn lokale KOnvergenz -> Überprüfung inflow <-> Outflow verhlältnis zur globalen Konvergenz.