- **Parallele Algorithmen (ParAlg)**
- **Zeitpunkt:** März 2014
- **Dauer:** 30 Minuten (5 ECTS)
- **Prüfer:** Ronald Veldema
- **Beisitzer:** Thorsten Blaß
- **Ergebnis:** sehr gut, faire Benotung

### Vorbemerkung

Entspannte Atmosphäre wie schon bei PoPL. Beisitzer hat wirklich nur protokolliert und Ronald die Fragen gestellt. Es ging ziemlich flink los mit ein paar Aufwärmfragen. Oft reichen ihm ganz bestimmte Schlagwörter, weshalb die Antworten nicht immer bei der Entstehung der Welt beginnen müssen.
Hängt man mal wird auf jedem Fall nachgeholfen! Oft kommt nach einer Antwort auch ein „richtig" von ihm, was im Verlauf der Prüfung Sicherheit bringt.
Es sind wirklich nur Konzepte und Ideen relevant. Spezielle Syntax von Sprachen oder deren Eigenschaften sind vollkommen belanglos und kommen eigentlich auch nie dran.

### Warum sollte ich mich mit parallelen Algorithmen beschäftigen?

Hardwareentwicklung zeigt Trend hin zu Multicores.
Parallele Algorithmen können auf diesen schneller Ausgeführt werden.
Man kann somit in gleicher Zeit ein größeres Problem bearbeiten.

### Mein sequentielles Programm braucht 10 Minuten, parallel 3. Was ist der Speedup?

10/3

### Was ist Amdahls Law?

Verhältnis von sequentieller Ausführungszeit zu paralleler.
Parallel setzt sich aus sequentiellem und parallelisierbarem Anteil zusammen.
Sequentieller Anteil begrenzt maximal möglichen Speedup. Speedup verhält sich daher nicht linear.

### Du hast zwei globale Variablen x, y und zwei CPUs. Gibt es da einen Livelock, Deadlock oder funktioniert das so?

```
x = 3;
y = 4;

CPU 0:
if(x == 3) {
  y++;
}

CPU 1:
if(y == 4) {
  x++;
}
```

Kommt auf die Verschränkung der beiden CPUs an und wie schnell eine CPU den Schreibvorgang einer anderen sieht. Generell weder Live- noch Deadlock, da beide CPUs zeitlich gesehen prinzipiell nie „exakt" gleich Schreiben können.

### Okay, du hast folgenden Ausdruck: F(y == 4). Stimmt der?

Das ist ein LTL-Ausdruck. Da wir aber im nicht-deterministischen Bereich sind, bräuchten wir einen CTL-Ausdruck, sprich noch ein E oder A davor.

### Was würde das A bedeuten?

A bedeutet auf allen möglichen Pfaden in der Zukunft wird die Bedingung gültig.

**Wie könnte man denn erreichen, dass es kein Problem gibt?**
Mutexe einsetzen.

**Welche kennst du da?**
Atomic exchange, voting-Lock, ticketing-Lock.

**Was passiert da?**
Atomic exchange und ticketing-Lock erklärt, war ausreichend.

**Du hast zwei Zahnräder. Das erste macht 1 U/s das zweite 50 U/s. Du hast noch viele andere Zahnräder und die du zwischen diese Zahnräder bauen kannst damit sich beide gleichschnell drehen. Wie machst du das?**
Parameter Sweeping: Zufallsbasiert alle Kombinationen durchprobieren bis sich eine gute Annährung findet
Genetische Algorithmen: Zufallsbasierte Anfangspopulation erstellen, diese ändern durch Mutationen oder Crossovers. Das Verändern kann man parallel machen.

**Ich habe ein schönes Integral für dich: [0,10] Integral ($\cos^2 x * \tan^2 x$) dx. Wie berechnest du die Fläche?**
Monte-Carlo-Simulation: Zufallsbasiert Punkte erzeugen. Verhältnis Treffer/Nichttreffer == Fläche
FE-Simulation: Fläche unter Graphen in Stufen einteilen, Summe aller Stufenflächen == Fläche
Breite der Stufen entscheidet dabei über Güte der Approximation.

**Bleiben wir mal bei Simulation. Was ist eine FE-Simulation überhaupt?**
Diskretisierung eines kontinuierlichen Bereichs. Z.B. durch Einteilen in endlich viele Dreiecke.

**Du hast eine Brücke und möchtest rausfinden wie viele LKWs auf einmal drüber fahren können.**
Wieder FE-Simulation. Brücke wird in Dreiecke unterteilt. Man nimmt die Belastungswerte der Baumaterialien und simuliert die Kraft/Druck/Lastverteilung Zeitschritt für Zeitschritt. Kräfte werden bspw. prozentual oder wahrscheinlichkeitsbasiert an andere Dreiecke weitergegeben.
Jede CPU berechnet eine bestimmte Anzahl an Dreiecken. Kommunikation erfolgt dann nur an Dreiecksgrenzen.

**Jetzt etwas Kreatives. Du hast einen Graphen und möchtest alle „strongly connected nodes" herausfinden.**
Man startet mit einem beliebigen Knoten, nimmt dessen Nachbarn und dessen Nachbarn und dessen Nachbarn etc.
Man überprüft quasi ob von jedem Knoten jeder andere (direkt oder indirekt) erreicht werden kann.

**Du möchtest einen parallelen Multiplizierer entwerfen, der zwei Binärzahlen multipliziert. Wie könnte man das machen?**
0011 * 0100
---
0011 * 0 parallel
0011 * 1 parallel
0011 * 0 parallel
0011 * 0 parallel
+ ---
Ergebnis
Addition wäre potentiell auch parallel durchführbar.

**Du hast eine Zahl und möchtest rausfinden aus welchen Primzahlen man sie durch Multiplikation erhält.**
Sieb des Eratosthenes bis zur gegebenen Zahl anwenden. Dadurch hat man alle Primzahlen.
Zahl durch alle gefundenen Primzahlen dividieren (parallel). Ist das Ergebnis wieder ein Primzahl hat man potentielles Ergebnis gefunden.

**Du hast eine PRAM Maschine und willst das Maximum eines Arrays rausfinden.**
*Ich kannte einen Algorithmus von Wikipedia [1], hatte ihn aber nur überflogen.*
*Hab ihn dann auch nur noch mit Ach und Krach und extrem viel Nachhilfe hinbekommen…*☺

```
for i=1 to n pardo
   b_i := 1
for i,j=1 to n pardo
   if x_j > x_i
   then
      b_i := 0
   fi
for i=1 to n pardo
   if b_i = 1
   then
      m := i
   fi
```

[1] http://de.wikipedia.org/wiki/Parallel_Random_Access_Machine