Praktikumsprojekt

Oliver Heidmann, Tronje Krabbe

Uni Hamburg
Praktikum Parallele Programmierung

03.06.2015

Übersicht

Simulation eines Sonnensystems

- Wir betrachten n Objekte mit jeweils einer Masse m und einem Geschwindigkeitsvektor v.
- Im Zentrum befindet sich ein stationäres Objekt mit besonders hoher Masse, analog zu einer Sonne oder einem schwarzen Loch.
- Ziel: eine vielzahl möglicher stellarer Konstellationen simulieren und beobachten, wie größere Objekte durch Kollision entstehen.

Funktionsweise

Pro Iteration:

- Verrechnung aller Geschwindigkeitsvektoren
- Errechnung aktualisierter Positionen
- Kollisionsbehandlung (wie?)

Schwierigkeiten/Herausforderungen

- Wie Kollisionen behandeln?
  - naiv: ignorieren
  - beide Objekte zerstören und neue(s) Objekt(e) erzeugen
- Größe des Systems
  - dynamische Größe? Eher nicht.
  - Objekte könnten das System verlassen
- Visualisierung

Code

```
1   class Objects
2   {
3       sortedlist<vector_3D<double>> position;
4       sortedlist<vector_3D<double>> velocity;
5       sortedlist<double> radius;
6       sortedlist<double> mass;
7       unsigned long number_of_objects;
8
9       void add(vector_3D<double> position, vector_3D<double>  velo, double r, double m)
10      {
11          position.add(position);
12          velocity.add(velo);
13          radius.add(r);
14          mass.add(m);
15          number_of_objects++;
16      }
17      vector<Object_data>  calculate_collision(vector<unsigned long>)
18      {
19          /*
20          hier wird berechnet, ob mehrere Objekte mit der Gesamtmasse der
21          kollidierenden Objekte gebildet werden,
22          oder ob nur ein neues bei der Kollision entsteht.
23          Außerdem: Größe, Radius, Geschwindigkeitsvektor.
24          */
25      }
26  }
```

Sequentiell

INPUT:

- Größe
- initiale Anzahl Obj.
- max. Anzahl Obj.
- min. und max. Masse
- min. und max. Geschwindigkeit
- Stellar-Obj. Masse

Sequentiell

LOOP:

- Verrechnen bzw. Update aller Geschwindigkeitsvektoren
- Update aller Positionen
- ggfs. Objekte löschen
- Kollisionen verarbeiten
- alle $n \in \mathbb{N}$ Loops: Status (schlau & effizient) speichern

Projektplan

Einmal wöchentlich treffen bis es fertig ist.

- Woche 0: diese Presentation präsentieren
- Woche 1-2: sequentielle Implementation
- Woche 4-6: Parallelisierung
- Woche 7-8: Optimierung/Debugging
- Woche 9: Visualisierung
- Woche 10: Leistungsanalyse
- Woche 11-n: Verbesserungen

Parallelisierung

1. Naiv: Aufteilung nach Index (jeder Prozess bekommt n/p Objekte)
2. Cool(?): Aufteilung nach Anzahl beeinflussten Objekten
   - Ein Objekt hat Einfluss auf unterschiedlich viele andere Objekte. Je nachdem, wie viele beeinflusst werden, muss mehr berechnet werden. Verteilt man Objekte nach dieser Gewichtung, kann die Last auf Prozesse optimiert werden.
   - Erhöht Speicherbedarf u.U. enorm