2b. Vergleich der Scheduling-Algorithmen

Wir haben unter folgenden Bedingungen verglichen:

- 12 Threads
- 100 Interlines
- 2. Funktion
- 10 Iterationen

Folgende Laufzeiten haben sich ergeben:

Element (ohne schedule()-Funktion):
  Berechnungszeit: 109.216796 s
Element, static, 1
  Berechnungszeit: 119.168228 s
Element, static, 2
  Berechnungszeit: 106.999247 s
Element, static, 4
  Berechnungszeit: 112.134933 s
Element, static, 16
  Berechnungszeit: 108.829530 s
Element, dynamic, 1
  Berechnungszeit: 103.958648 s
Element, dynamic, 4
  Berechnungszeit: 104.577804 s
Element, guided
  Berechnungszeit: 103.121636 s

Spalte, static, 1
  Berechnungszeit: 114.024175 s
Spalte, static, 2
  Berechnungszeit: 118.513555 s
Spalte, static, 4
  Berechnungszeit: 120.505354 s
Spalte, static, 16
  Berechnungszeit: 114.065403 s
Spalte, dynamic, 1
  Berechnungszeit: 115.328284 s
Spalte, dynamic, 4
  Berechnungszeit: 115.145380 s
Spalte, guided
  Berechnungszeit: 106.633639 s

Erstmal ist die Element-Rechenweise wesentlich effizienter, da die Elemente alle gleich verteilt werden, und Spalten nicht immer optimal auf Threads aufgeteilt werden können, weil nicht alle ganzzahlig häufigen Spalten durch anders-ganzzahlige Threads teilbar sind. Guided erzielt bei beiden Rechenweisen jeweils die effizienteste Möglichkeit, da die Datenaufteilung womöglich optimierter stattfindet. Das bedeutet, dass die Element-

Rattmann, Sharaf, Nayci

Reichenweise mit dem Guided-Scheduling Overall die beste Rechenzeit hat. Static-Scheduling wird wenn die Aufteilung zufälligerweise passt gute Ergebnisse erzielen, aber durch dynamic, wodurch die Zuteilung wie der Name verrät dynamischer als bei der statischen Verteilung stattfindet, werden im Schnitt bessere Ergebnisse erzielt. Dies muss wie gesagt nicht sein, was man an der Spalten-Rechenweise mit Static,16 Scheduling gegen das schlechtere Dynamic Scheduling sehen kann.