HLR - Hochleistungsrechnen

Aufgabenblatt 5

Merle Hoffmann, Joël Miramon, Max Press

2a. Umsetzung der Datenaufteilungen:

Mit den Aufrufen `./partdiff-openmp-<aufteilung> 12 2 512 2 2 600` sind wir zu den folgenden Laufzeiten gekommen:

spalte	element	zeile
466.367264 s	63.214012 s	30.309984 s

Dabei war die Laufzeit von `partdiff-openmp-spalte` zwischen den Runs sehr unterschiedlich, von kaum langsamer bis deutlich langsamer als `zeile`. `partdiff-openmp-element` war sehr deutlich langsamer als `partdiff-openmp-zeile`. Das liegt daran, dass die Matrix nicht in zusammenhängenden Stücken im Cache gespeichert werden kann, also auf immer den langsameren Hauptspeicher zurückgegriffen werden muss.

2b. Vergleich der Scheduling-Algorithmen:

In diesen Messungen wurde mit 12 Threads, 512 Interlines, komplexer Störfunktion und 600 Iterationen getestet. Die Datei `askparams.c` wurde geändert, um mit zwei weiteren Parametern das Scheduling beeinflussen zu können. Beispielaufruf für den Typ dynamic mit Blocksize 4: `./partdiff-openmp-element 12 2 512 2 2 600 2 4`.

	partdiff-openmp-element	partdiff-openmp-zeile
static, 1	466.367264 s	30.309984 s
static, 2	465.606153 s	30.146841 s
static, 4	464.022059 s	30.149005 s
static, 16	463.252306 s	30.181750 s
dynamic, 1	463.532810 s	30.233440 s
dynamic, 4	462.842803 s	30.080348 s
guided, 1	464.787176 s	30.332783 s

Für die Implementierung mit Aufteilung nach Elementen ließ sich eine leichte Verbesserung durch Erhöhung der Blocksize feststellen. Die Zeilenaufteilung konnte nicht davon profitieren.

3. Leistungsanalyse:

Alle Messungen wurden auf dem Knoten `abu5` durchgeführt. Die genauen Ergebnisse sind in der Datei `results.txt` zu finden.

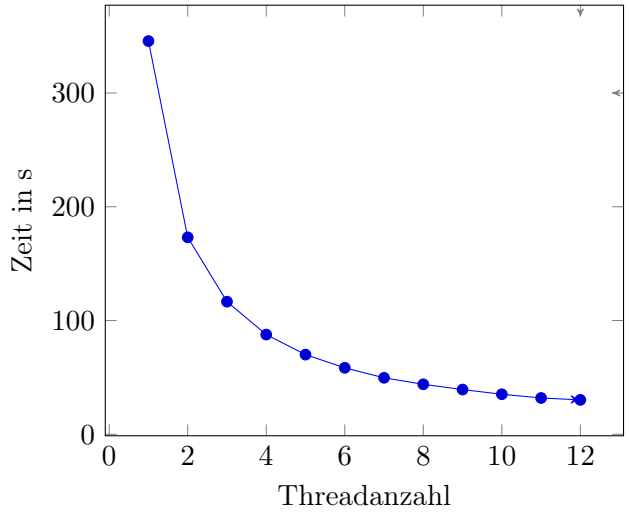

Abbildung 1: Skalierung mit Threads

Hier wurde mit den Befehlen `./partdiff-openmp <threads> 2 512 2 2 600`, also 512 Interlines, komplexe Störfunktion und 600 Iterationen eine nahezu antilineare Skalierung festgestellt. Mit 12 Threads war die Berechnung ca. 11,4 mal schneller. Mit zunehmender Threadanzahl nimmt der Speedup pro Thread allerdings natürlich ab.

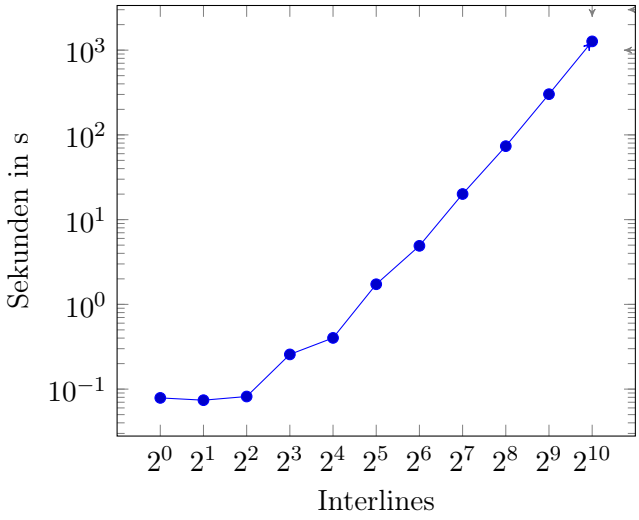

Abbildung 2: Skalierung mit Interlines (logarithmisch)

Hier wurden die Befehle `./partdiff-openmp 12 2 <interlines> 2 2 6000` ausgeführt und man stellt fest, dass die Laufzeit bei Verdoppelung der Interlines sich in etwa vervierfacht. Das liegt daran, dass die Anzahl der zu berechnenden Elemente ungefähr proportional zu dem Quadrat der Interlines ist. Bei kleiner Matrixgröße ist der Speedup nicht so groß, sind die Interlines z.B. zwischen 1 und 4, lässt sich kein wesentlicher Unterschied feststellen, da die OpenMP-Implementation hier zu viel Overhead hat.