Übungsblatt 4 - Hochleistungsrechnen
Leistungsanalyse
Tim Kilian, Joscha Fregin, Stefan Knispel

1 Umsetzung der Datenaufteilungen

Vergleich durchgeführt mit: partdiff-(spalten,zeilen,element) 12 2 512 2 2 1100
Element(static, 1):64.94s
Element (ohne scheduling): 57.37s
Zeilen: 65.25s
Spalten: 132.80s

Die elementweise Aufteilung läuft am schnellsten. Wenn die Zerlegung der Matrix nicht festgelegt wird (kein scheduling) wird es noch schneller. Die Spaltenweise Aufteilung ist am langsamsten, weil durch das Vertauschen der Indices, der Speicher für den Programmablauf ungünstig eingelesen wird. Dies ist bei der Zeilenmethode nicht der Fall, weshalb sie schneller ist (wie bei der ersten Optimierungsaufgabe zum sequentiellen Verfahren).

2 Vergleich der Scheduling Algorithmen

Der Vergleich der Algorithmen wurde auf dem Rechenknoten west7 durchgeführt mit folgenden Einstellungen:
./partdiff-openmp 12 2 512 2 2 1100, der code des sequentiellen Programms wurde nur durch die OpenMP Parallelisierung erweitert (keine Zeilen, Spalten Elementweise Aufteilung). Datenaufteilung durch Elemente wird später betrachtet.
Wie Tabelle 1 zu entnehmen ist nicht ganz eindeutig welcher Algorithmus am besten funktioniert. guided läuft zwar mit 48.85s am schnellsten, ist jedoch nicht so konstant wie die kaum langsamer laufende Einstellung dynamic (1,4). Am langsamsten läuft der static Algorithmus. Er weist auch die größten Schwankungen der Laufzeiten auf.

Für die elementweise Zerlegung ist die guided Methode ebenfalls die

Blockgr.	dynamic	static	guided
			52.65
			48.85
			49.05
1	49.46	52.76	
	49.54	61.74	
	49.35	58.04	
2		59.89	
		68.84	
		57.93	
4	49.47	52.43	
	49.45	63.69	
	49.29	60.72	
16		64.74	
		64.55	
		62.74	

Tabelle 1: Vergleich der Scheduling Algorithmen mit jeweils drei Messläufen pro Einstellung in Sekunden

Blockgr.	dynamic	static	guided
			49.76
1	84.87	64.94	
2		71.41	
4	59.35	56.18	
16		66.79	

Tabelle 2: Vergleich der Scheduling Algorithmen für Elementzerlegung mit jeweils einem Messlauf pro Einstellung in Sekunden

schnellste. Jedoch ist dynamic langsamer als static (selbst bei gleicher Blockgröße).

3 Messung 1

Bei der Messung 1 wurden die Parameter wie folgt gewählt: - Threadanzahl ansteigend von 1 bis 12
- Jacobi-Methode
- Interlines 512
- Störfunktion 2
- 1100 Iterationen

Es wurde auf dem Rechnerknoten west5 gerechnet! In der Abbildung 1 ist die Berechnungszeit gegen die Anzahl der verwendeten Threads dargestellt.

Die Berechnungszeit für einen einzelnen Thread dauert erwartungsgemäß am längsten. Bei zwei verwendeten Threads halbiert sich die Berechnungszeit näherungsweise. Plottet man die Berechnungszeit gegenüber der Anzahl der Threads erhält man einen exponentiellen Verlauf. Hier ist deutlich zu erkennen, dass ab einer Verwendung von 6 Threads der Gradient sehr gering ist. Man erhält keine deutliche Verbesserung in der Berechnungszeit mehr im Vergleich zur Anzahl an Threads. Bei der Verwendung von 12 Threads verkürzt sich die Berechnungszeit im Vergleich zu der mit einem Thread bei uns um den Faktor 11.5. Damit liegen wir innerhalb des geforderten Bereiches.
Das sequentielle Programm hat auf Knoten 7 581.856389 s (582.753409 s im zweiten Durchlauf) benötigt und liegt damit knapp über dem OpenMP Programm mit einem Thread. Möglicherweise funktiniert die Compileroptimierung für OpenMP hier besser, oder der Unterschied ist durch den Rechenknoten entstanden.

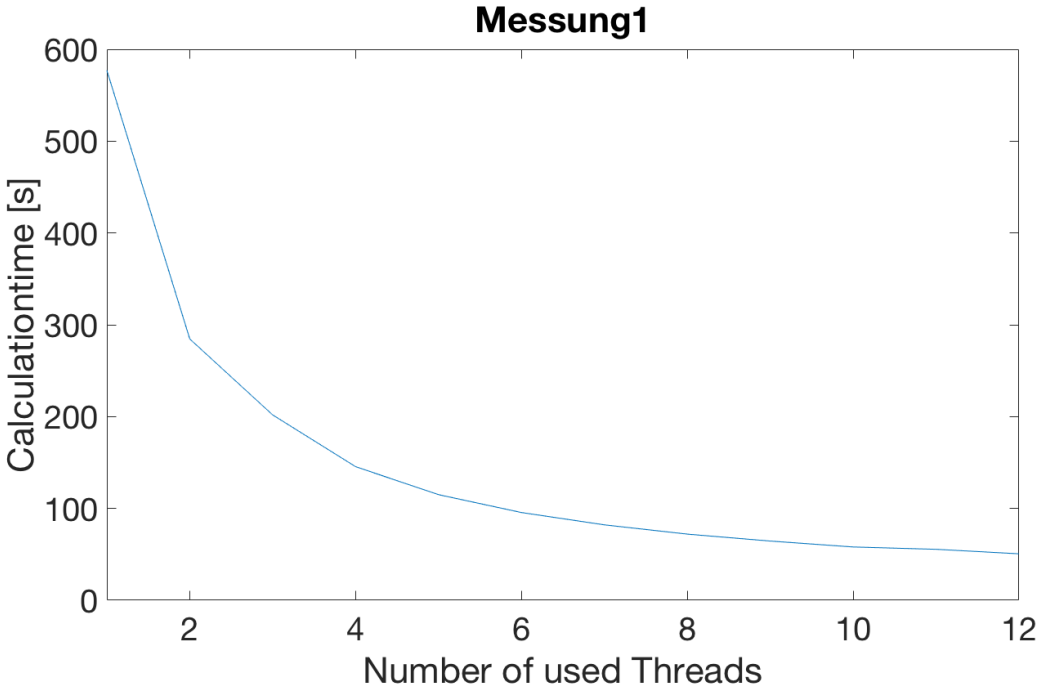

Abbildung 1: Messung 1 mit 512 Interlines und 1100 Iterationen.

4 Messung 2

Es wurde auf dem Rechnerknoten west5 gerechnet! In der Abbildung 2 ist die Berechnungszeit gegen die Anzahl der Interlines dargestellt. Es ist nicht eindeutig zu erkennen, ob hier ein exponentieller Verlauf vorliegt. Dazu müsste mit noch weiteren Interlines gerechnet werden. Jedoch ist bei einer Anzahl von 1,2 oder 4 Interlines kein Unterschied in der Berechnungszeit zu erkennen. Jeder Durchlauf brachte Berechnungszeiten zwischen 0.3 und 0.9 Sekunden zum Vorschein. Das kommt durch die zu klein gewählte Matrix. Bei einer Zeilenanzahl von maximal 4 kann nicht jeder Thread eine Zeile zum Berechnen übernehmen. Die Berechnungszeit müsste theoretisch zwischen 1 und 12 Interlines fast identisch sein. Ab einer Interlines Anzahl von 13 müsste diese erst wieder ansteigen, da dann mehr Zeilen als Threads vorhanden sind.

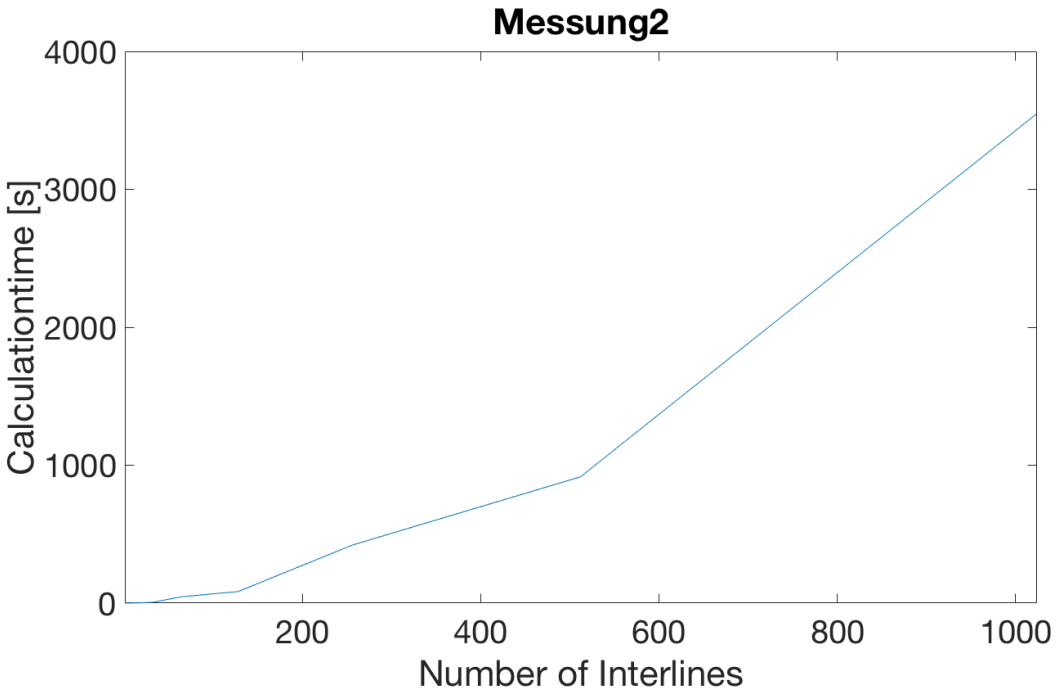

Abbildung 2: Messung 2 mit 20000 Iterationen und 12 Threads.