Aufgabe 8.2: Leistungsanalyse

7.12.2019

Im folgenden wird die Standardabweichung durch

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}}$$

mit $n = 3$ berechnet.

1 Erklärung

Jeder Aufruf von Partdiff-Par-Hybrid, wo theoretisch mindestens 36 bis maximal 72 Prozessorkerne, auf 3 Knoten verteilt, gleichzeitig beansprucht werden sollten, zeigt einen erheblichen Speedup von mindestens 21 und bis fast 43. Dies liegt jedoch weit unter dem optimalen Speedup, der linear mit der Anzahl der zu benutzenden Prozessorkernen wachsen würde.
Ein Grund dafür ist die nicht-optimale Programmierung des MPI Abschnittes von Partdiff-Par-Hybrid (und damit auch Partdiff-Par, da sie sich nur in der Parallelisierung der Schleife über die Zeilen unterscheiden, die MPI frei ist).

Alle Prozesse laufen synchron in der Iteration, da nach der Schleife über die Zeilen der Aufruf MPI_Allreduce wie eine Barriere wirkt, was zwar die Programmierung erleichtert, aber die Performance stark runterzieht. Es kann außerdem auch eine kleine Beschleunigung erzielt werden, indem die synchrone Kommunkation durch MPI_Send und MPI_Recv für das Austauschen der ersten und letzten beiden Zeilen durch eine asynchrone ersetzt und in die Schleife über die Zeilen geschoben wird, wobei die Zeilen die verschickt werden müssen asynchron nach der Berechnung verschickt werden und asynchron oder auch synchron nach der Schleife empfangen wird.
Was in Abbildung 4 sofort aufällt, ist dass die Ausführung von Partdiff-Par-Hybrid doppelt so langsam mit 24 Prozessen pro Knoten ist, als mit 12 Prozessen pro Knoten, was auf den ineffizienten Nachrichtenaustausch über MPI zurückzuführen ist.
Was überraschend ist, das die Laufzeit von 1 Prozess pro Kern und 12 Threads pro Prozess fast genauso schnell ist wie die Laufzeit von 12 Prozessen pro Kern und ein Thread pro Prozess, wobei man annehmen könnte, dass die Variante mit 1 Prozess pro Kern und 12 Threads pro Prozess schneller ist, da weniger Nachrichten über MPI ausgetauscht werden und damit weniger zwischen den Kernen hinweg vor allem mit MPI_Allreduce. Schließlich kann man den Schluss ziehen, das es ein Programm mit mehr Threads als Prozess pro Knoten schneller läuft, als mehr Prozesse als Threads, wobei die Sättigung da erreicht wird, wenn das Produkt von Threads pro Prozess und Prozess pro Knoten die Anzahl der Prozessorkerne der jeweiligen Knoten übersteigt.

Parameter	
Iterationen:	2800
Interlines:	512
Störfunktion:	2
Verfahren:	Jacobi

Abbildung 1: Parameter für die Ausführung von Partdiff und dessen Derivat Partdiff-Par-Hybrid

Partdiff
2,217.861

Abbildung 2: Ausführungszeit von Partdiff in Sekunden

Threads	Prozesse	Zeit	σ	tpp	Speedup
1	24	104.78	0.134	0.042	21.167
12	1	82.44	3.175	12	26.903
1	12	80.95	0.431	0.083	27.398
2	12	62.633	0.37	0.167	35.41
12	2	52.397	0.026	6	42.328
24	1	51.797	0.158	24	42.819

Abbildung 3: Mittlere Ausführungszeiten in Sekunden gefolgt von ihrer jeweiligen Standardabweichung σ

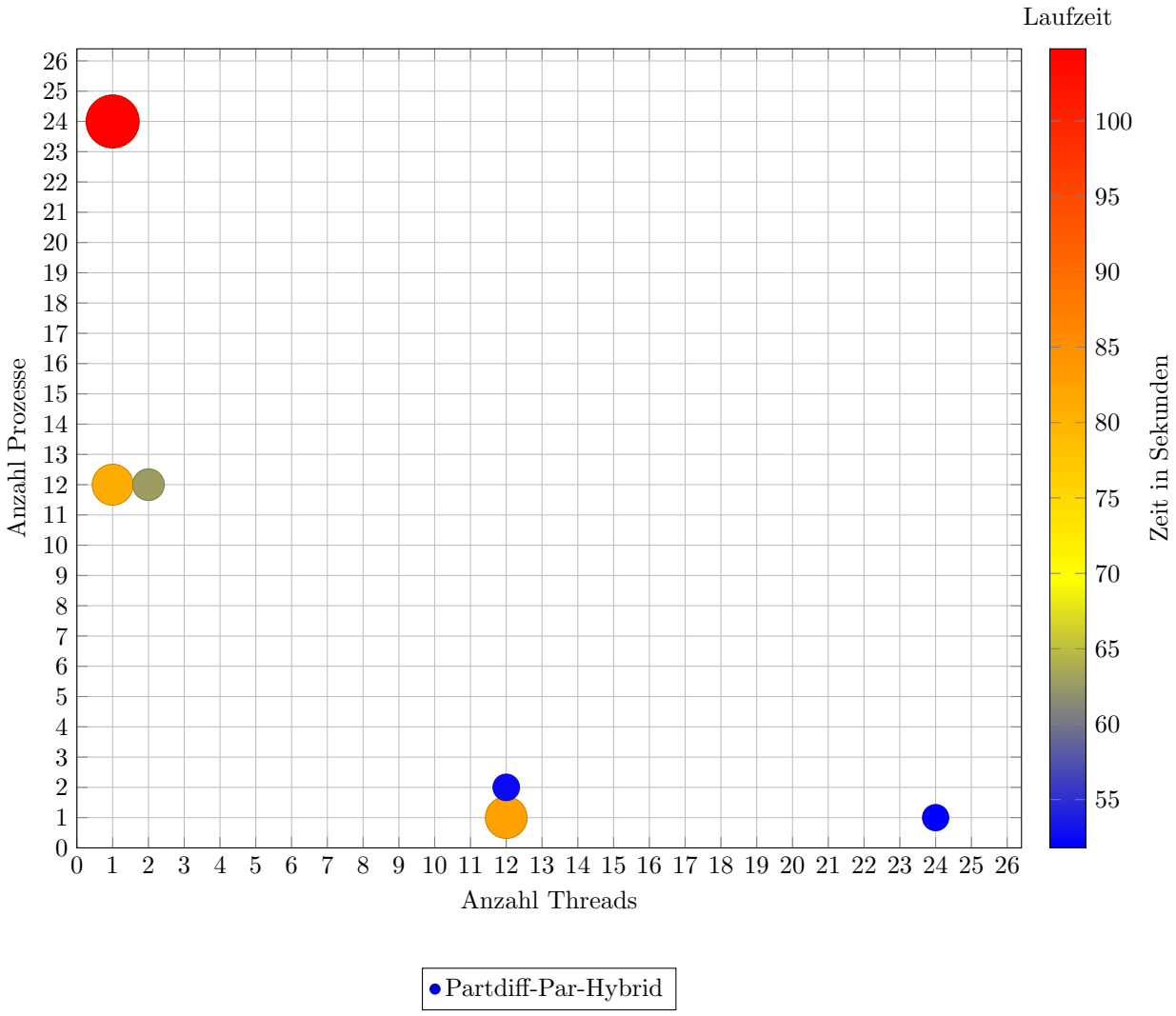

Abbildung 4: Vergleich der Laufzeiten aller sechs Aufrufe. Die Farbe und Größe der Punkte stellen die Laufzeit dar, je größer der Punkt desto mehr Zeit wurde benötigt

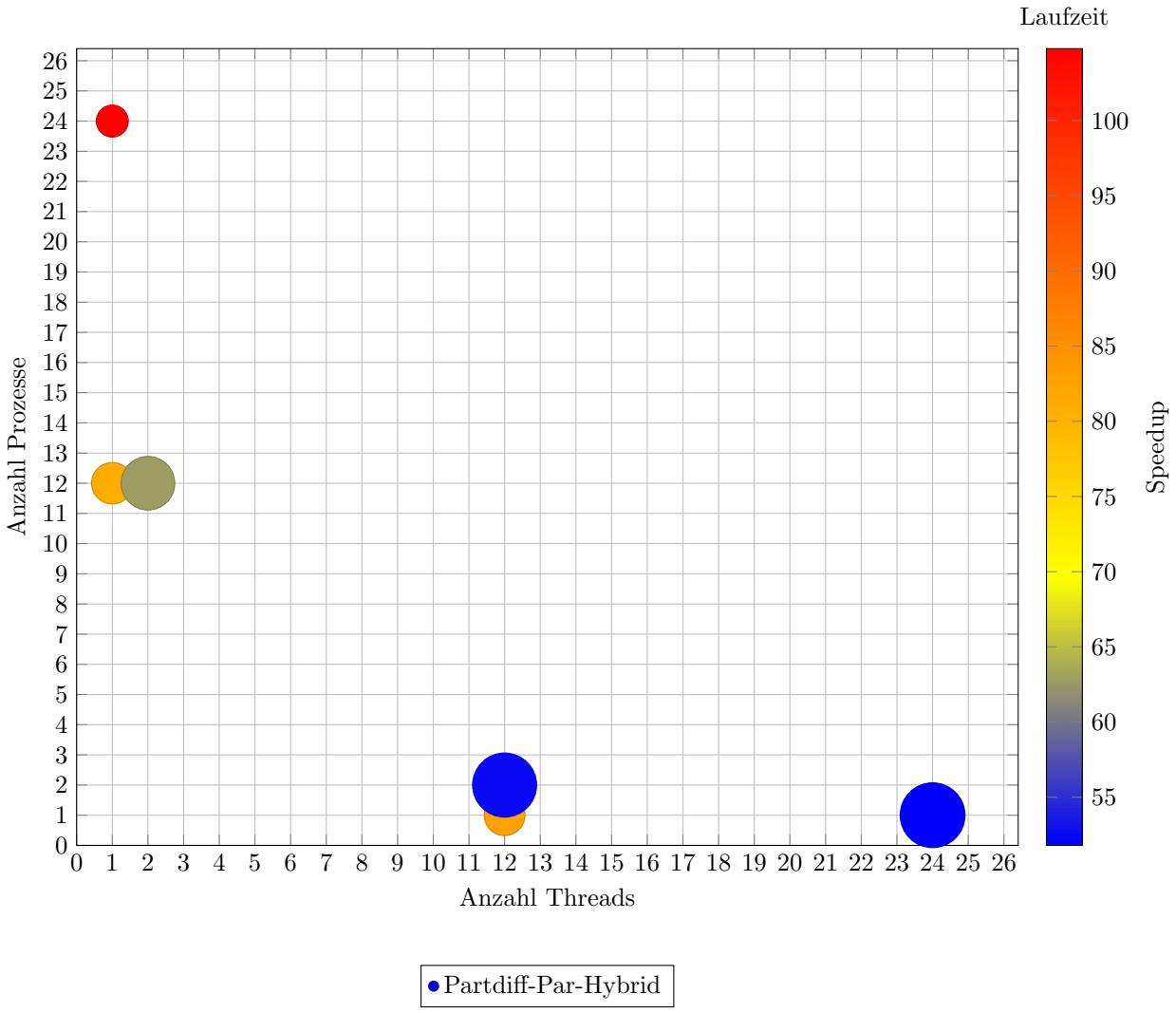

Abbildung 5: Speedup Graph. Bezieht sich auf das sequentielle Partdiff. Je größer der Punkt desto größer der Speedup