Aufgabe 4: Leistungsanalyse

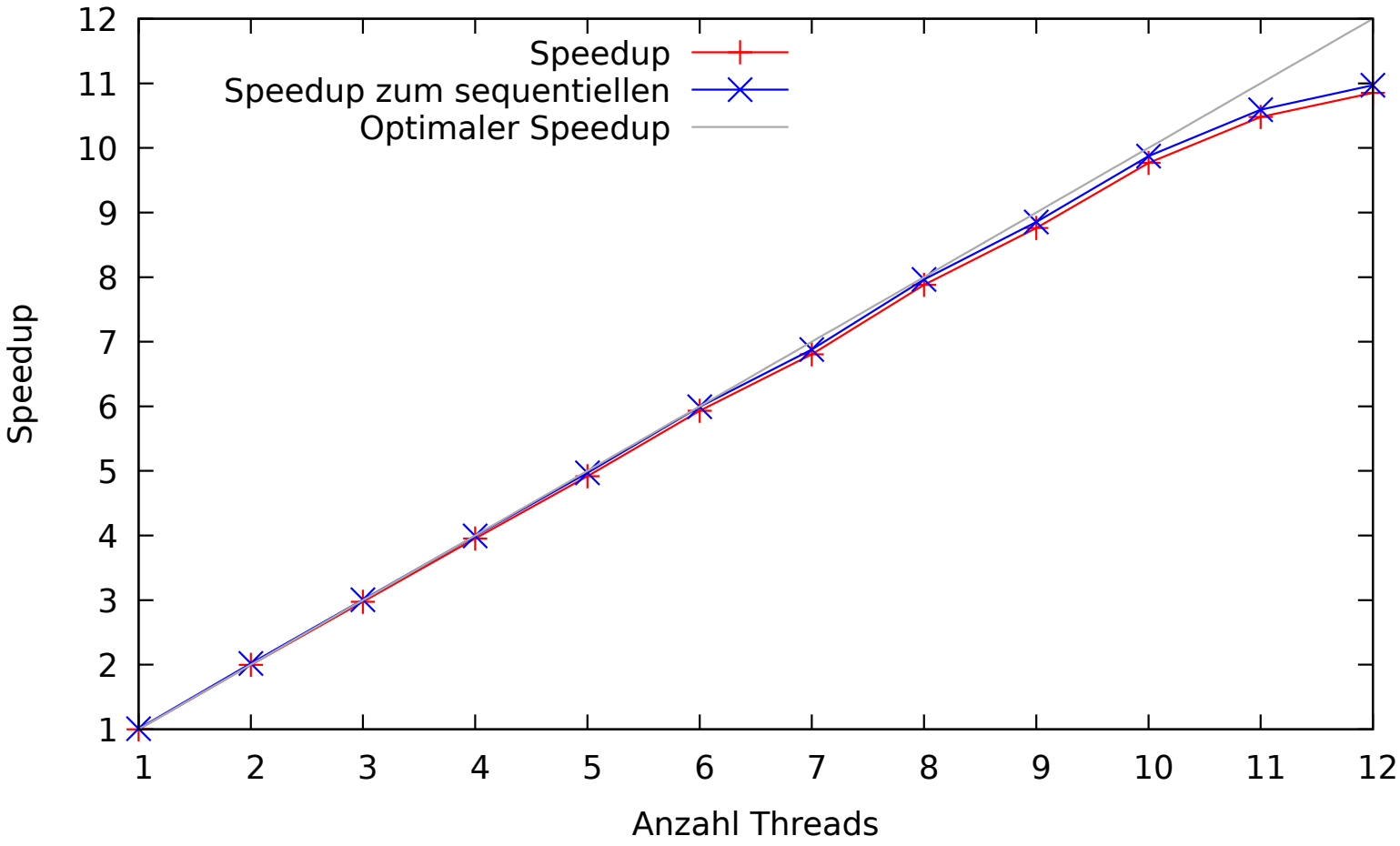
09.11.2019

Messung 1:

Parameter:
Iterationen: 1500
Interlines: 512
Störfunktion: 2
Verfahren: Jacobi
Ausführungsort: Knoten (west4) auf dem Cluster

Ergebnis:

Threads	partdiff (s)	partdiff-openmp-zeilen (s)	Speedup-Seq	Speedup-1-Thread
1	791.25	782.90	1.01	1.00
2	791.25	391.77	2.02	2.00
3	791.25	263.29	3.00	2.97
4	791.25	198.05	3.99	3.94
5	791.25	159.22	4.97	4.91
6	791.25	132.03	5.99	5.92
7	791.25	115.03	6.87	6.80
8	791.25	99.35	7.98	7.89
9	791.25	89.37	8.88	8.78
10	791.25	80.14	9.88	9.77
11	791.25	74.71	10.68	10.56
12	791.25	72.13	11.98	10.86

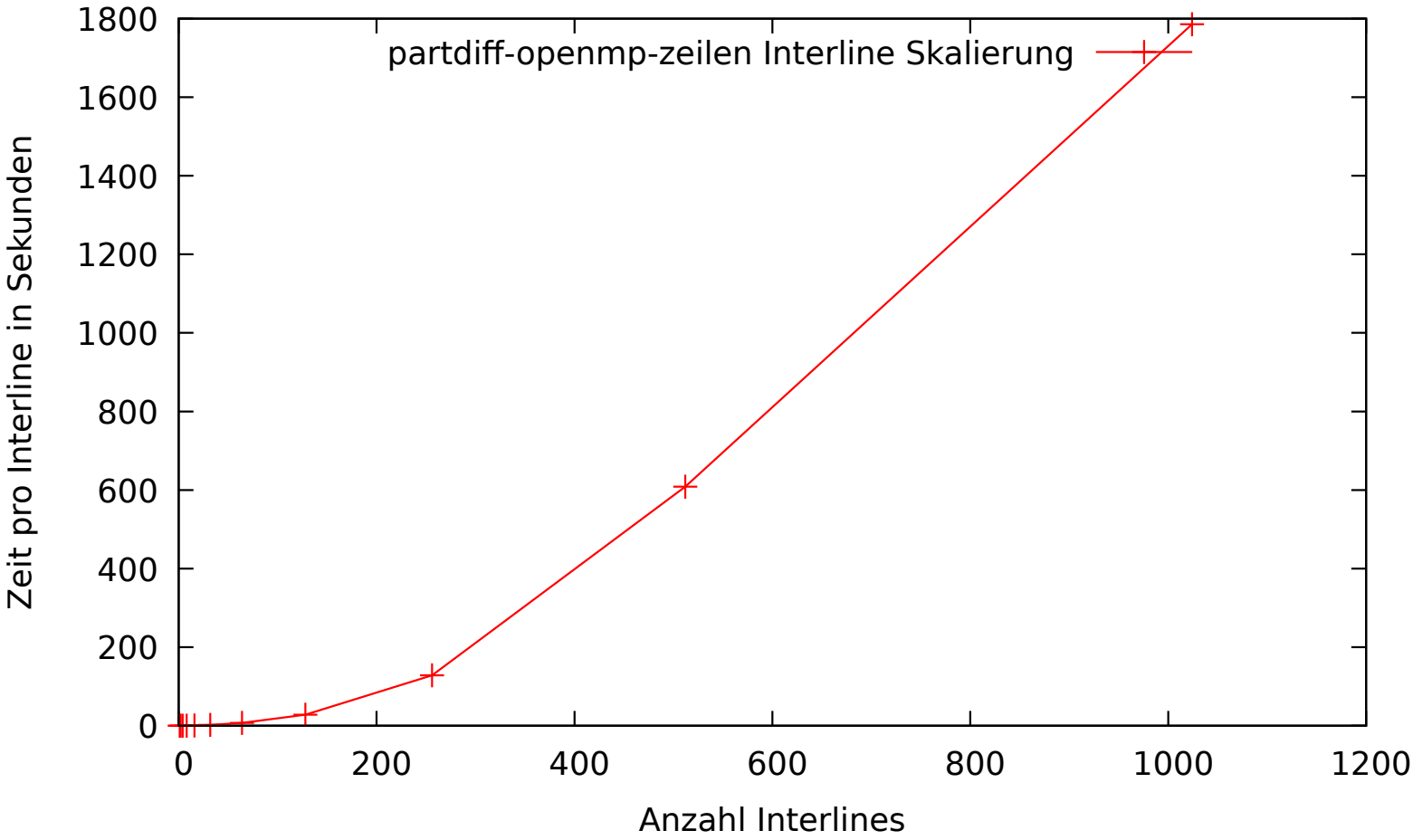

Speedup
Speedup zum sequentiellen
Optimaler Speedup
Speedup
Anzahl Threads
1
2
3
4
5
6
7
8
9
10
11
12

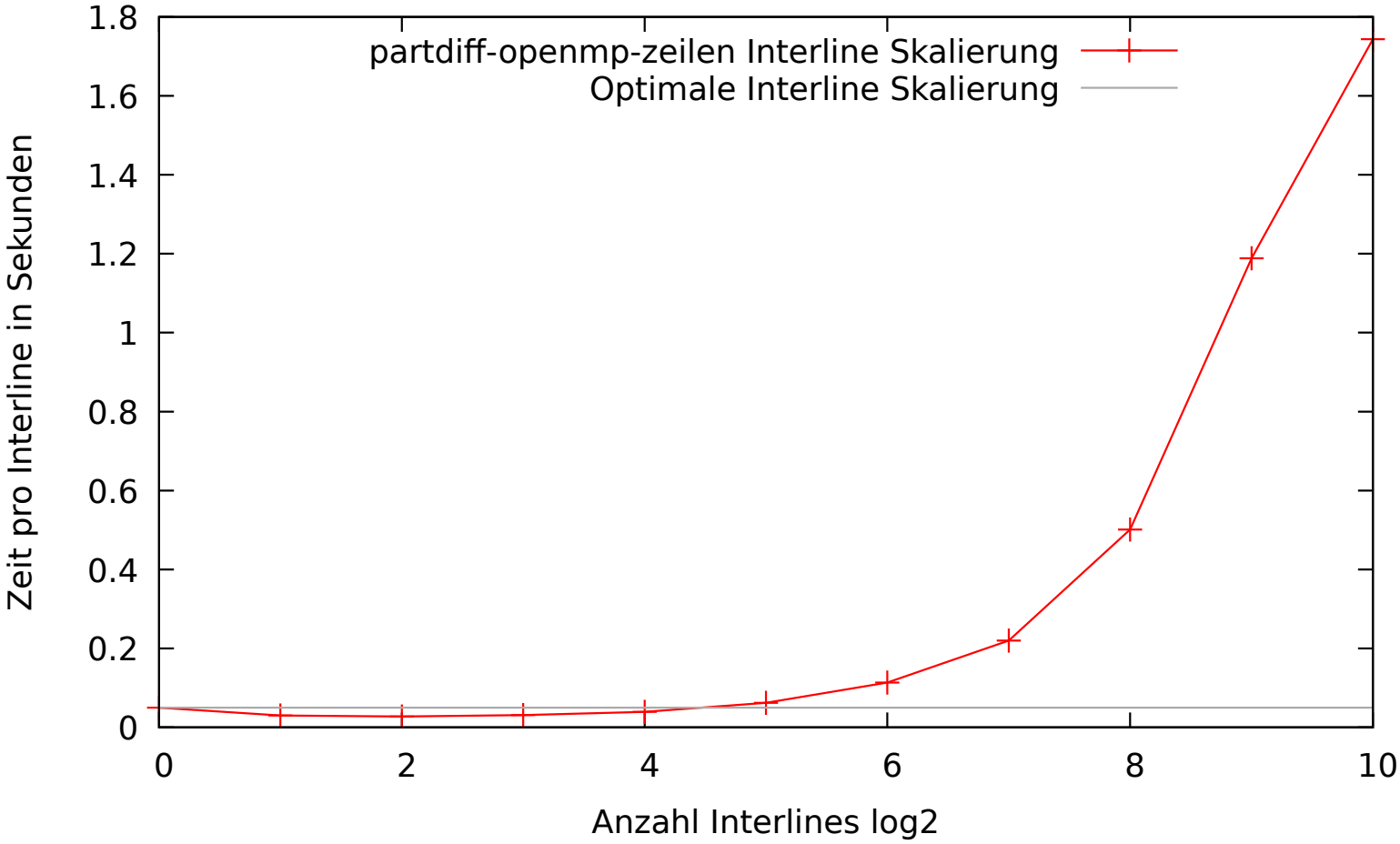
Erklärung:
Es is gut zu sehen, das die parallele Variante einen fast linearen Speedup im Vergleich zu der sequentiellen Variante hat, als auch fast linearen Speedup im Abhängigkeit der Anzahl der Threads. Die Skalierung ist hier nahezu linear, da es genügend Arbeit für jeden Thread vorhanden ist. Die For-Schleifen nehmen auch einen großteil der Arbeitszeit in Anspruch d.h. der Overhead für die Parallelisierung innerhalb der while-Schleife ist verhältnismäßig klein im Vergleich zu dem Zeitgewinn. Anfangs ist ein fast linear Speedup zu beobachten, wobei die Wachstumsrate mit zunehmender Threadanzahl abnimmt. Es ist zu vermuten, das der Aufwand und damit der Zeitverbrauch der Verwaltung der Threads und Datenaufteilung schneller wächst, als der Zeitgewinn durch die Parallelisierung auf mehr Threads.

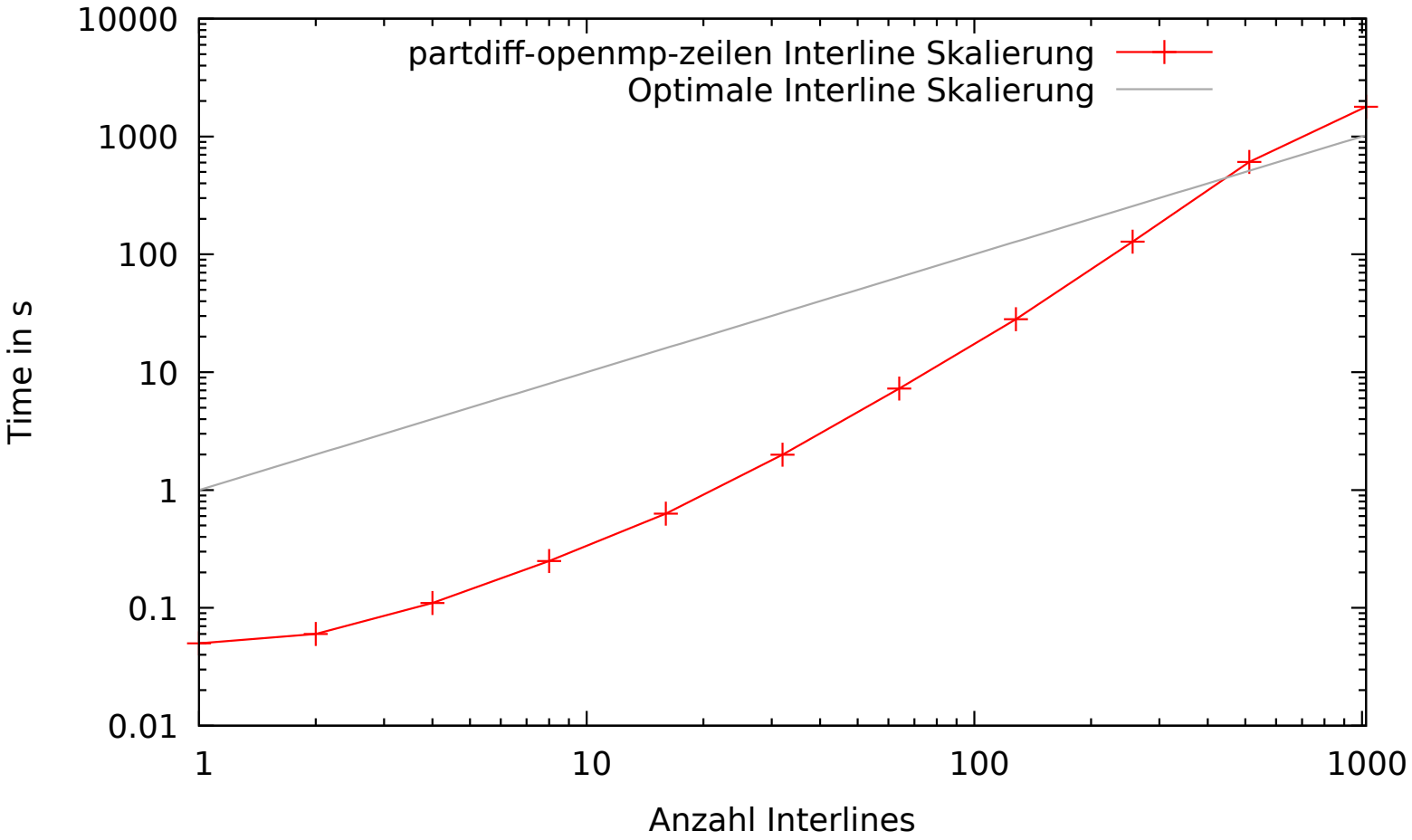
Messung 2:
Iterationen: 1500
Störfunktion: 2
Verfahren: Jacobi
Thread: 12
Ausführungsort: Knoten (west7) auf dem Cluster

Ergebnis:

Interlines	partdiff-openmp-zeilen [s]	Zeit [s] pro Interline
1	0.05	0.0500
2	0.06	0.0300
4	0.11	0.0275
8	0.25	0.0313
16	0.63	0.0394
32	1.99	0.0622
64	7.26	0.1134
128	28.11	0.2196
256	128.26	0.5010
512	608.58	1.1886
1024	1785.61	1.7438


partdiff-openmp-zeilen Interline Skalierung
Zeit pro Interline in Sekunden
Anzahl Interlines
0
200
400
600
800
1000
1200
1400
1600
1800


partdiff-openmp-zeilen Interline Skalierung
Optimale Interline Skalierung
Zeit pro Interline in Sekunden
Anzahl Interlines log2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
0
2
4
6
8
10


partdiff-openmp-zeilen Interline Skalierung
Optimale Interline Skalierung
Time in s
Anzahl Interlines
10000
1000
100
10
1
0.1
0.01
1
10
100
1000

Erklärung:
In dem Diagramm auf Seite 5 kann man gut sehen, das einen Bereich gibt, wo die Zeit pro Interline sich verringert, danach aber wieder steigt. Das kann dadurch zustande kommen, das erst ab dem Minimum (4 Interlines), der Aufwand der Verwaltung der Datenaufteilung kleiner wird als die eigentliche Rechenzeit pro Interline. Die Rechenzeit sollte in einem sequentiellen Programm quadratisch mit den Interlines wachsen, da die Interlines die Größe der Matrix bestimmen, über die in zwei verschachtelten Schleifen iteriert wird, folglich quadratisch. Dieses Wachstum äußert sich durch die Parallelisierung mit mehr Interlines eher in einer linearen Rate, was man an den Diagramm auf Seite 4 gut sehen kann.