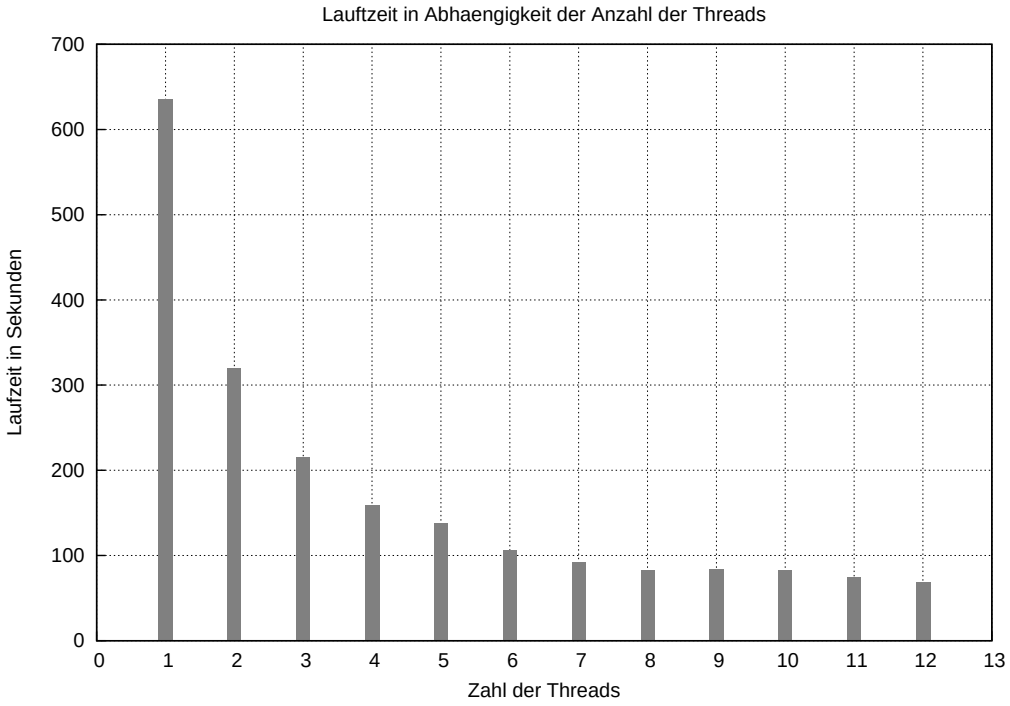
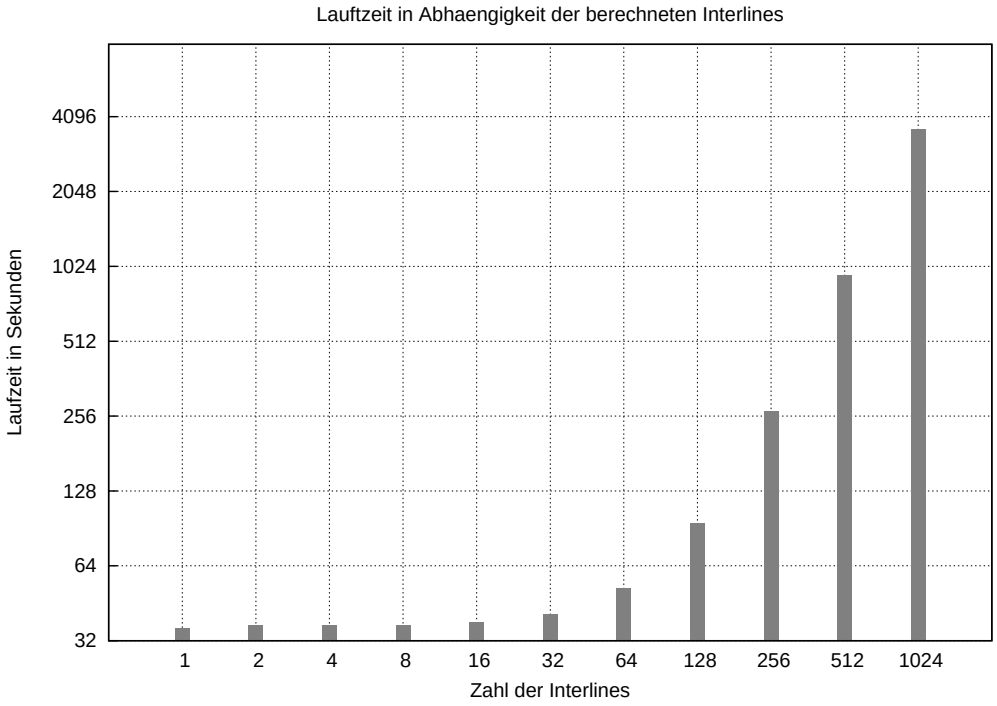
Messung 1



Es ist anhand der oben stehenden Daten zu erkennen, dass bei relativ langen Laufzeiten (ca. 10min bei 1 Thread) diese tatsächlich in etwa antiproportional zur Anzahl der Threads skaliert. Bei 12 Threads ist ein Speed-Up-Faktor von etwa 10 zu erkennen. Erstaunlicherweise gab es bei unseren Messungen erhebliche Schwankungen der Laufzeit ab ca. 8 Threads – auch in die falsche Richtung, in der Mittelung ergaben sich diese i.d.R. aber. Ab etwa 8 Threads ist der Laufzeitgewinn nicht mehr allzu groß, lediglich die letzte Messung bei 12 Threads ergab noch einmal etwas mehr als 10Geschwindigkeitsgewinn ggü. 11 Threads. Dieses Verhalten ist bspw. Mit dem Verwaltungsaufwand der Threads zu erklären. Bei Konstanter Anzahl durchzuführender Rechnungen und steigenden Threads wird immer mehr Rechenzeit auch dafür benötigt, die einzelnen Schleifendurchläufe auf die einzelnen Threads zu verteilen. Hierdurch ist zu erklären, dass man keinen Speed-Up von 12 beobachtet. Auch ist sicher eine mögliche Erklärung, dass Systemkonstanten wie Speicherzugriffszeit auch durch mehr Threads nicht weniger werden, sondern im Gegenteil sich immer mehr Threads mit den Speicherzugriffen gegenseitig behindern.

Messung 2



Anhand dds Laufzeit-Verlaufs ist ersichtlich, dass ab etwa 128 berechneten Interlines die Laufzeit quadratisch zu den Interlines ansteigt (Doppelte Interlines → 4-Fache Laufzeit). Zuvor hält sich die Laufzeit relativ konstant auf 37-38s. Dieses späte skalieren lässt sich wiederum wahrscheinlich mit entsprechendem Thread-Verwaltungsaufwand und Systemkonstanten erklären.

Umsetzung der Datenaufteilungen

Berechnungsmethode: Jacobi
Interlines: 512
Stoerfunktion: $f(x, y) = 2\pi^2 \cdot \sin(pi \cdot x) \sin(pi \cdot y)$
Terminierung: Anzahl der Iterationen
Anzahl Iterationen: 1024
Norm des Fehlers: 2.929037e-07

Zeilenweise Aufteilung:
Berechnungszeit: 64,6 s
Spaltenweise Aufteilung:
Berechnungszeit: 132,2 s
Elementweise Aufteilung:
Berechnungszeit: 125,7 s

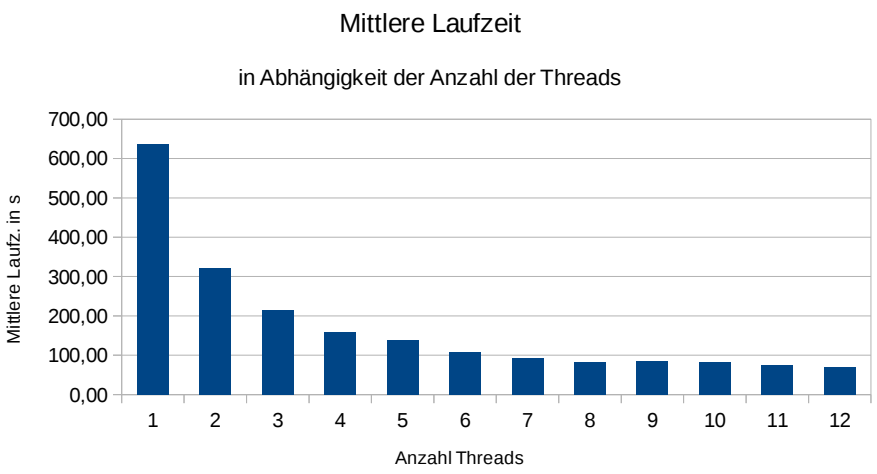
Hier fällt auf, dass die Zeilenweise Aufteilung ungefähr doppelt so schnell wie die spaltenweise Aufteilung läuft. Bei letzterer wird zuerst über die Spalten, dann über die Zeilen iteriert, hierdurch kommt es zu vielen Sprüngen im Speicher (wie beim vorherigen Übungsblatt).
Die elementweise Aufteilung braucht verhältnismäßig auch viel länger. Für jedes Element muss i, j und fpisin_i neu berechnet werden, außerdem treten hier wahrscheinlich auch vermehrt Sprünge im Speicher auf.

Messergebnisse mit Mittelung:

Threads

Durchlauf

#Threads	1	2	3	Mittelung=
1	636,2	635,8	635,6	635,87
2	320	320,1	321,1	320,40
3	214,1	217,7	213,9	215,23
4	159,7	159,8	159,7	159,73
5	128,1	128,4	159,4	138,63
6	106,6	106,5	106,5	106,53
7	92,6	92,7	92,1	92,47
8	88,5	80	79,9	82,80
9	89,2	93,3	71,2	84,57
10	92,2	64	91,8	82,67
11	62,8	83,7	76,2	74,23
12	63,4	75,12	67,8	68,77



Interlines

Messung

		1	2	3	Mittelung=
	1	32,5	37,7	38	36,07
	2	37,6	37,7	37,8	37,70
	4	37,5	37,4	37,5	37,47
	8	37,9	38	37,6	37,83
	16	38,3	38,1	38,4	38,27
Interlines	32	41,6	41,5	41,7	41,60
	64	52,7	52,3	52,5	52,50
	128	95,8	95,7	96	95,83
	256	268	268,34	267,2	267,85
	512	944,1	945,42	942,5	944,01
	1024	3654,1	3652,5	3658,3	3.654,97