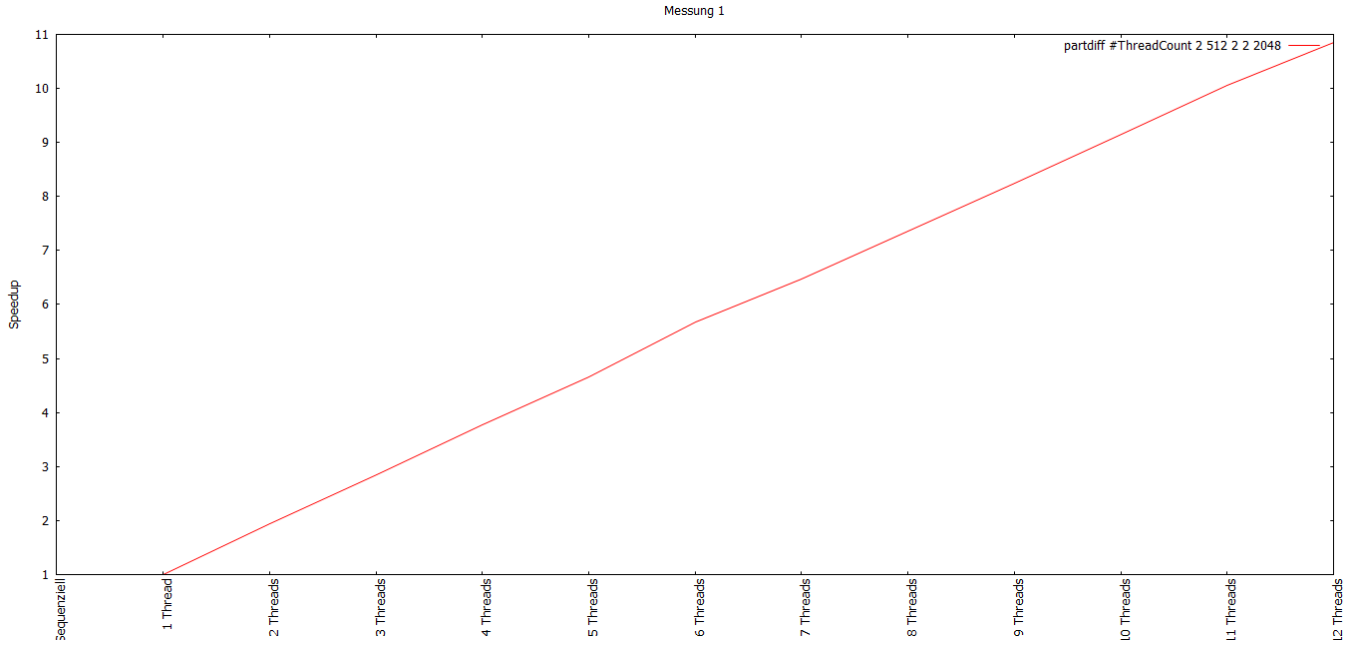
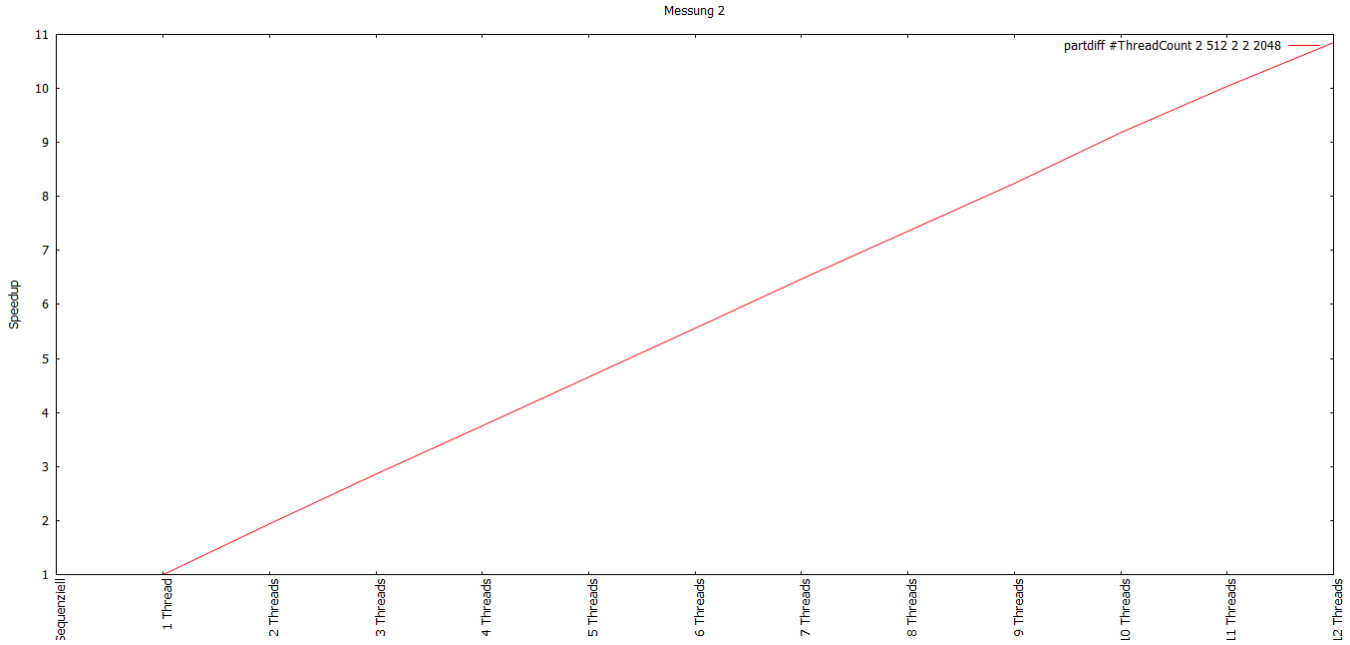
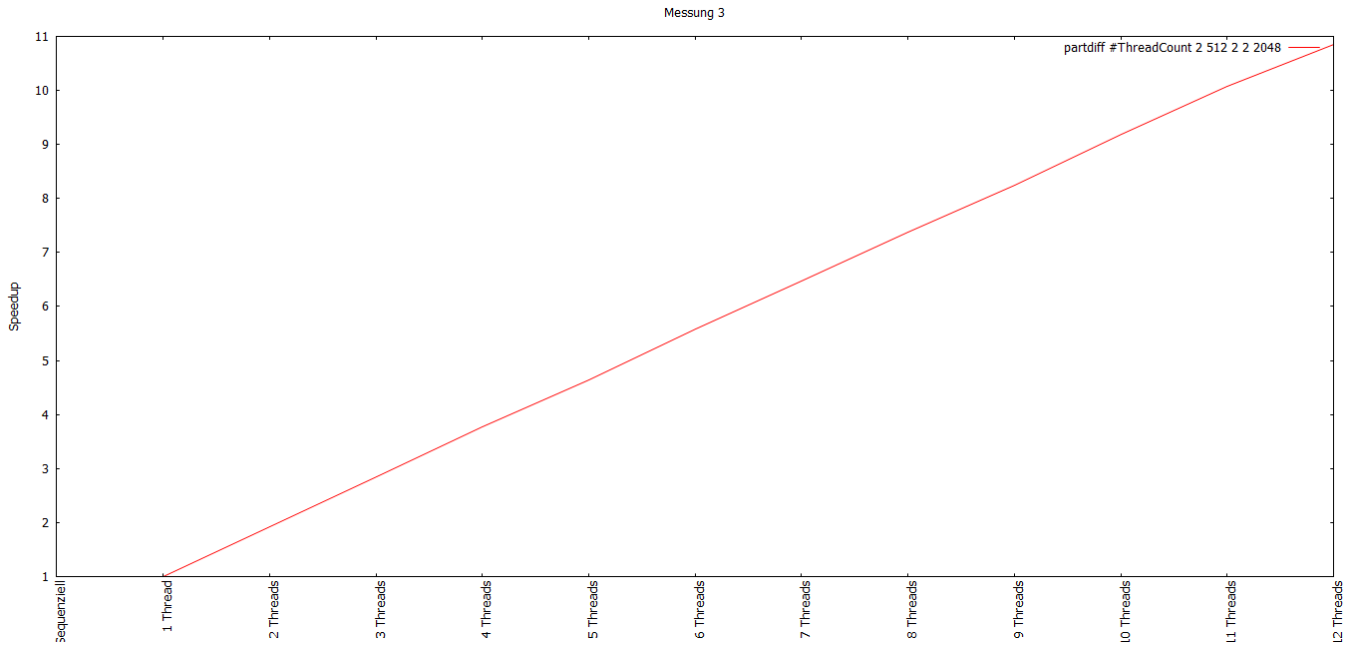
Blatt 5

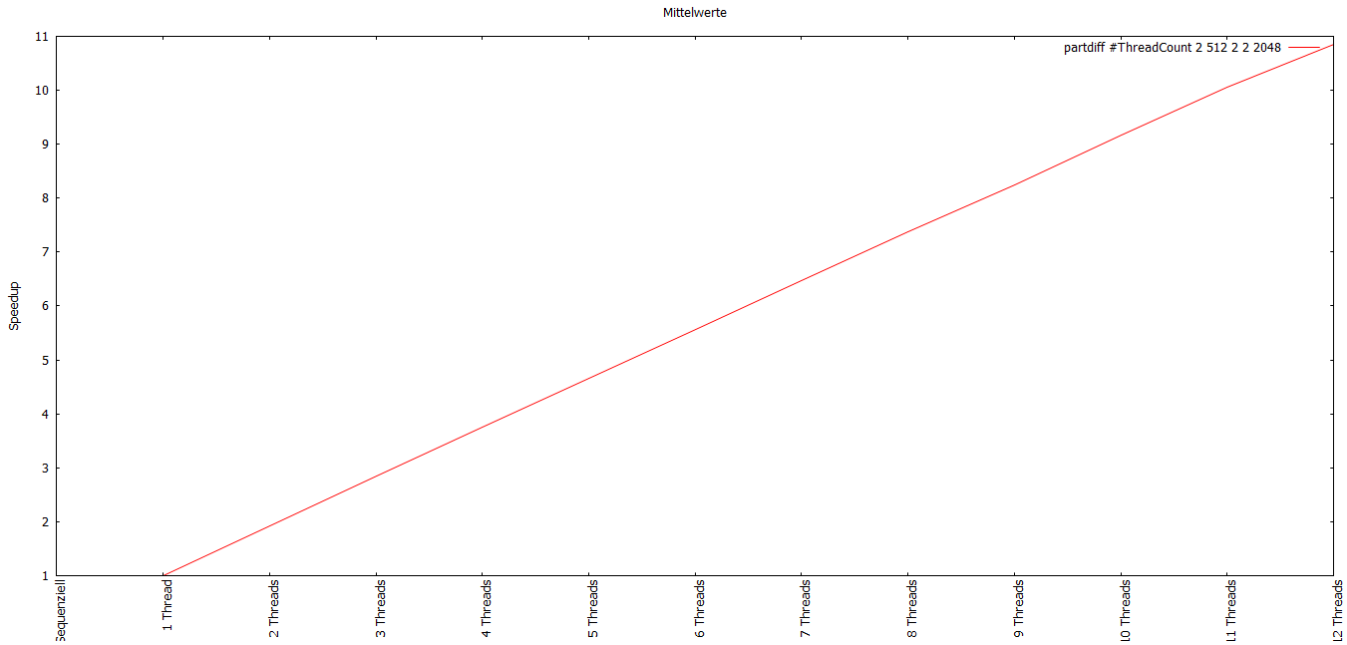
Aufgabe 2 Leistungsmessung

Anzahl Threads	Sequenziell	1	2	3	4	5	6	7	8	9	10	11	12
Messung 1 in sek.	594.7	589.38	306.59	208.67	157.92	127.95	106.86	91.85	80.79	72.14	64.98	59.07	54.78
Messung 2 in sek.	594.75	587.13	306.87	208.16	158.50	127.75	106.91	91.92	80.70	72.17	64.75	59.29	54.78
Messung 3 in sek.	594.84	589.38	307.72	208.65	157.54	128.22	106.60	91.88	80.59	72.06	64.75	59.07	54.79
Mittelwert in sek.	594.76	588.63	307.06	208.49	157.98	127.97	106.59	91.88	80.69	72.12	64.82	59.14	54.78









Auswertung:

Wie deutlich zu erkennen ist, skaliert das Programm linear mit der Anzahl der Threads,
hierbei sind allerdings einige Schwankungen zu betrachten, die im Mittel aber sehr gut ausgeglichen sind.
Besonders fällt aber der kleine Knick im Graphen beim Sprung von 11 zu 12 Threads auf, dies könnte sich damit erklären lassen, dass der Hauptthread, welcher sich eigentlich um die Verwaltung der anderen Threads kümmert nun ebenfalls mit ausgelastet wird und nun nicht mehr die Zeit hat die Arbeit effektiv an die Worker zu verteilen. Diesem kann man entgegen wirken, in dem man einen Threadpool in das Programm einbaut, da das Programm aber bereits einen Speedup von etwas über 11 mit 12 Threads erreicht, wurde bewusst darauf verzichtet.
Eine Weitere Performance Minderung kommt dann auch aus dem eben genanten Gründen, da jedes mal neue Threads erstellt werden müssen, muss einiges im Betriebssystem an Arbeit verrichtet werden,
somit verliert das Programm bei einer großen Anzahl von Iterationen deutlich an Performance. Ebenfalls interessant ist, dass das Programm bereits mit einem Thread um einige Sekunden schneller ist, als die des Sequenziellen Programmes, was ebenfalls damit erklärt werden könnte, dass der Worker die Matrizen immer Effektiv im Cache behalten kann und der Master (welcher auf einem Anderen Thread läuft, aber solange nichts tut bis der Worker fertig ist) dann die Ergebnisse auswerten und die Abbruchbedingung prüfen kann.

Der Zeitaufwand betrug etwa 8 Stunden, dafür wurde etwa 2 zum Debuggen verwendet, der Rest ist für das Austesten von verschiedenen Scheduling Strategien und Leistungsanalyse draufgegangen.