Protokollierung der Arbeitszeit:

Benötigte Zeit zum Optimieren des Codes insgesamt:
18 Std
Davon Fehlersuche:
6 Std
Davon Nachlesen in OpenMP Dokumentation:
10 Std

Vergleich verschiedener Datenaufteilungen:

Da wir bei dem Verfahren, das wir optimieren sollten zwei Matrizen verwenden, und diese jeweils Zeilenweise im Speicher abgelegt worden sind, würde es sich anbieten, auch dementsprechend den Datenzugriff zu optimieren. Es wird sich also empfehlen, den Threads jeweils Zeilen nacheinander zuzuordnen, da so beim Cachen, sinnvoller Weise oft die benötigten Werte schon geladen sind. Somit verringert sich die Speicherzugriffszeit deutlich.
Eine Elementweise Zuordnung der Aufgaben auf die Threads macht nur dann einen Sinn, wenn diese weiterhin Teile von Zeilen an die Threads verteilt werden. So lässt sich auch bei einer kleineren Matrix noch eher durch die Erhöhung der Threadanzahl die Leistung steigern. Außerdem erscheint eine solche Aufteilung nur günstig, wenn eine Implementierung dieser auch effizient genug möglich ist. Denn durch die Aufteilung der Matrix in einzelne Elemente oder Pakete von Elementen entsteht möglicherweise ein immer größer werdender Kommunikations-Overhead bei der Berechnung von maxresiduum. Denn jeder Thread muss eine lokale Kopie dieses Wertes im Speicher vorhalten und sie dann mit der globalen Version abgleichen, sobald der Teil der Berechnung beendet ist, der diesem Thread zugewiesen wurde.

Eine andere Aufteilung der Daten empfiehlt sich beispielsweise bei der Verwendung der Sprache Fortran. Hier wird eine Matrix automatisch vom Compiler Spaltenweise im Speicher abgelegt. Also muss auch der Datenzugriff auf diese Weise erfolgen um wiederum möglichst viele der Daten bereits vorher im Cache liegen zu haben.

Leistungsanalyse: