Hochleistungsrechnen *Übungsgruppe 3*

Universität Hamburg *Prof. Dr. T. Ludwig*

Aufgabenblatt 7 **9. Dezember 2017**

Hochleistungsrechnen

Gian-Luca Fuhrmann (5fuhrman) Maik-Chiron Graaf (5graaf) Hieu Nguyen (5nguyen)

Aufgabe 3: Parallelisierung mit MPI (Schema: 120 Punkte)

3.1. Datenaufteilung

Die Daten der Matrix werden gleichmäßig auf die Prozesse aufgeteilt. Sind die Daten der Matrix nicht gleichmäßtig aufteilbar, so muss ein Prozess mehr bzw. weniger Daten verwalten als die anderen Prozesse.

3.2. Parallelisierungsschema für das Jacobi-Verfahren

Wir haben uns dafür entschieden die while-Schleife der calculate-Funktion zu beschreiben. In der Funktion gibt es bereits zwei Matrizen die mit Werten initialisiert wurden, einmal Matrix_In und Matrix_Out. Mit jedem Durchlauf der Schleife werden Werte von Matrix_In gelesen, und Werte nach Matrix_Out geschrieben. Um die Schleife zu parallelisieren, müssen sich alle Prozesse in derselben Iteration befinden, da sonst der Zugriff auf aktuelle Werte fehlt. Es werden in der zweiten Iteration Werte verwendet, die in der ersten Iteration berechnet wurden, da zur Berechnung der Werte in der aktuellen Reihe immer die aktuellsten Werte der vorigen Reihe benötigt werden. Deshalb müssen alle Prozesse nach jedem Durchlauf der while-Schleife ihre Werte senden, und darauf warten neue zu erhalten. Alle Prozesse berechnen somit unabhängig voneinander die Werte ihrer Reihen, und senden dann ihre Daten an die Prozesse, die auf diese angewiesen sind.Erst anschließend, können die Prozesse den Ablauf von vorne beginnen und die aktuellsten Werte berechnen.

3.3. Parallelisierungsschema für das Gauß-Seidel-Verfahren

Hier haben wir nun das Problem, dass wir in die gleiche Matrix schreiben von der wir auch lesen. Das führt dazu, dass ein Prozess wohlmöglich in die Matrix schreibt während oder bevor ein anderer aus dieser liest. Um dieses Problem zu umgehen und das Programm erfolgreich zu Parallisieren müssen alle Prozesse am anfang jeder Interation ihre Daten an die anderen Prozesse senden somit empfängt jeder Prozess die Daten welche benötig werden und kann anschleißend wie gewohnt die Interation berechnen. Mithilfe einer Barrier nach Berechnung stellt man dann noch sicher, dass nur akutelle Daten gesendet werden.

3.4. Abbruchproblematik

Iterationszahl Nachdem die ganannte Interationszahl erreicht wurde brechen alle Prozesse ab, da in beiden Verfahren die Prozesse immer bei der gleichen Interationszahl sind.

Genauigkeit Auch bei der Genauigkeit ist es bei beiden Verfahren gleich und die Prozesse befinden sich in der gleichen Interation. Nach erreichen der gewünschten Genauigkeit eines Prozessen, teilt er dies allen anderen Prozessen mit und es wird abgebrochen.