Parallelisierung mit MPI

Yannik Könneker, Maik Simke, Jonas Bögle und Flo Dreyer

November 30, 2019

1 Parallelisierungsschema

Wir haben n Prozesse und eine $m \times m$ große Matrix.
Bei dem Gauß-Seidel-Verfahren erstellt jeder Prozess eine, beim Jacobi-Verfahren zwei, $(\frac{m}{n} + 2) \times m$ Matrizen, welche mit 0 initialisiert werden. Beim ersten und letzten Prozess muss beachtet werden, dass diese eine Zeile weniger benötigen, da diese ja keinen oberen bzw. unteren Rand haben. Falls $inf\_func = 2$ ist, so müssen alle Prozesse jeweils die Ränder der Formel anpassen. Sollte sich die Anzahl der Zeilen nicht gleichmäßig auf die Prozesse verteilen lassen, so erhalten bei einer gleichmäßigen Aufteilung und einem Rest von R die ersten R Prozesse eine Zeile mehr.

Das **Jacobi-Verfahren**: jeder Prozess startet, bevor die Hauptschleife betreten wird, drei asynchrone Receives, einmal für den oberen und einmal für den unteren Rand (Der erste und der letzte Prozess bilden selbstverständlichsterweise eine Ausnahme) und einen für die Terminierung, der obere und der untere Rand werden in seperaten Matrizen abgespeichert.
Die Hauptschleife ist eine while-true-Schleife. Es werden vorerst alle anderen Zeilen, die auf keinen Rand angewiesen sind berechnet. Am Ende der Schleife wartet der Prozess, bis der untere Rand erhalten wurde, berechnet mit ihm die unterste Zeile und schickt diese asynchron wieder ab, ein neuer asynchroner Receive wird erstellt. Anschließen wird selbig mit dem oberen Rand verfahren.
Wurden nun alle benötigten Zeilen erhalten und versandt (MPI_Wait), dann wird maxresiduum, falls die Terminierungsfunktion dies angibt, asynchron an Prozess 0 geschickt, welcher die maxresidua aller Prozesse in einem Array speichert und am Ende jeder Iteration vergleicht, ob alle Werte unter der gewünschten Präzision liegen. Ist dem der Fall, schickt Prozess 0 allen Prozessen die Abbruchnachricht, der Inhalt ist dabei eigentlich trivial. Jeder Prozess überprüft nun, ob die Abbruchnachricht erhalten wurde und beendet dann ggf. die Schleife mit einem *break*;. Ansonsten werden die beiden Matrizen vertauscht und die Schleife beginnt von neuem. Falls das Abbruchkriterium eine gewisse Anzahl an Iterationen ist, so wird diese von jedem Prozess verfolgt und direkt nach obigem *break*; überprüft.
Kommt es zu einem Abbruch, dann wird von den Prozessen in aufsteigender Rankordnung die aktuelle Matrix, ohne die zusätzlichen Ränder, ausgegeben.

Das **Gauß-Seidel-Verfahren**: Wenn wir die Reihenfolge der berechneten Elemente betrachten, lässt sich erkennen, dass diese ein diagonales Muster von links unten nach rechts oben verfolgen. Die Elemente, welche in der gleichen Diagonale liegen können parallel berechnet werden, die einzelnen Diagonalen zusammen sequentiell. Jede Diagonale muss auf die Diagonale links von ihr warten, bevor sie ihre Elemte berechnet. Wenn man sich dies visuell vorstellt, könnte dies so aussehen: