HLR MPI Parallelisierung PDE, Blatt 7

Arne Struck, Jonathan Werner

6. Dezember 2014

1 Aufteilung der Daten:

Die Matrix wird in n Abschnitte (wobei n der Anzahl der Prozesse entspricht) aufgeteilt. Jeder Prozess verwaltet einen dieser Abschnitte. Ein Abschnitt entspricht einer Matrix von $|rows|/n$ Zeilen und $|columns|$ Spalten (oder je nach Implementation umgekehrt).
Die Berechnungen werden hierbei nur auf diesen kleineren Submatrizen ausgeführt. Jede Matrix besitzt an ihrer oberen und unteren Grenze je eine weitere Zeile über die die Kommunikation mit den Nachbarprozessen läuft. Ausnahmen sind natürlich der erste und der letzte rechnende Prozess, welche mangels Nachbar an den entsprechenden Stellen nur jeweils eine weitere Zeile aufweisen.

2 Jacobi-Verfahren:

2.1 Parallelisierungsschema

Beim Jacobi-Verfahren berechnen alle Prozesse zuerst ihren Abschnitt der Matrix. Darauf senden sie dann asynchron ihre Kommunikationszeilen an die Nachbarprozesse (natürlich die der Ergebnismatrix). Sind alle Nachrichten ausgetauscht, werden die Matrizen der nächsten Iteration aus den Kommunikationsanteilen und den eigenen Ergebnissen zusammengefügt. Nun ist alles bereit für den nächsten Iterationsschritt. Daraus wird ersichtlich, dass aufgrund der geteilten Ergebnis- und Berechnungsmatrix keine Abhängigkeiten innerhalb einer Iteration auftreten.

2.2 Abbruchbedingung

2.2.1 Iterativ

Jeder Prozess besitzt einen Iterator, welcher inkrementiert wird bis die Iterationsgrenze erreicht ist. Nun muss nur noch das Resultat vom Master-Prozess zusammengesetzt werden. Natürlich muss hierfür noch eine Abschließende Kommunikation an den Master-Prozess geschehen. Alternativ lässt sich dies auch durch die einzelnen Prozesse in eine Datei schreiben. Dies gilt für alle Fälle und wird deswegen nicht wiederholt. Alle Prozesse befinden sich in der letzten Iteration.

2.2.2 Genauigkeit

Um einen Abbruch nach Genauigkeit zu erreichen muss der Hauptprozess (bspw durch MPI_Gather) alle Maxresiduen Sammeln und danach die Maximale Abweichung bestimmen. Sollte diese die Abbruchbedingung erfüllen, wird die nächste Iteration nicht ausgeführt. Es befinden sich alle Prozesse in der letzten Iteration (welche es auch sein mag).

3 Gauß-Seidel-Verfahren:

3.1 Parallelisierungsschema

Zur Erläuterung wird die Zeit im folgenden in Rechenzyklen eingeteilt, diese stellen einen parallelen Prozessablauf dar (jeder Prozess führt seine Iteration durch und Kommuniziert erfolgreich). Aufgrund der Abhängigkeiten der zu errechnenden Einträge von den west beziehungsweise north Einträgen kann das Jacobi-Schema nicht angewandt werden. Um überhaupt parallelisieren zu können müssen die einzelnen Prozesse zeitverschoben rechnen. Das bedeutet, dass nach der Aufteilung im ersten Zyklus Prozess 0 die erste Iteration für seinen Abschnitt durchführt und danach kommuniziert. Kein anderer Prozess rechnet diesen Zyklus. Die Kommunikation findet "von oben nach unten" am Ende des jeweiligen Zyklus statt, da die oberen Prozesse die Informationen von den unteren Prozessen nicht benötigen. Nun erst, da die Abhängigkeiten von der ersten Iteration für den Nachfolgeprozess bereitstehen, kann etwas parallel berechnet werden, nämlich im 2. Zyklus der Prozess 0 seinen Teil der 2. Iteration und der Prozess 1 seinen Teil der 1. Iteration. Im 3. Zyklus kann nun Prozess 2 den nächsten Teil der 1. Iteration berechnen, während die anderen Prozesse jeweils ihren Teil der nächsthöheren Iteration berechnen. Dieses Schema wird fortgeführt bis alle Prozesse (respektive der letzte) die Abbruchbedingung erreicht haben.

3.2 Abbruchbedingung

3.2.1 Iterativ

Jeder Prozess besitzt einen Iterator, welcher inkrementiert wird bis die Iterationsgrenze erreicht ist. Nun muss nur noch das Resultat vom Master-Prozess zusammengesetzt werden. Natürlich muss hierfür noch eine Abschließende Kommunikation an den Master-Prozess geschehen. Der Masterprozess kann hierbei pro Zyklus einen Abschnitt erhalten. Befindet sich der Prozess in der n-ten Iteration, ist sein nördlicher Nachbar theoretisch in der n+1-ten Iteration (hat also schon terminiert) und sein südlicher Nachbar in der n-1-ten Iteration.

3.2.2 Genauigkeit

Bei einem Abbruch nach Abbruchbedingung stellt sich die Frage, ob teilweise genauere Ergebnisse ein Problem, darstellen weil sie nicht aus der Iteration der Abbruchbedingung stammen. Da die Prozesse geringeren Ranges einen höheren Iterationstatus besitzen, als die höheren Ranges können die Informationen aus der Abbruchiteration nicht gesammelt werden, sofern sie nicht zwischengespeichert werden. Dieses Zwischenspeichern würde allerdings einen großen Speicheroverhead bedeuten, da jeder Prozess sich bis zur Prozessanzahl viele Teilmatrizen speichern müsste. Dies ist allerdings die einzige Möglichkeit die Matrix der Abbruchiteration zu erhalten. Sollte allerdings die erhöhte Genauigkeit keine Probleme darstellen, kann auf das Zwischenspeichern der Ergebnisse verzichtet werden. Sollte das Erreichen der Abbruchbedingung festgestellt werden (analog zum Verfahren im Jacobi-Abschnitt) und befindet sich der betrachtete Prozess in der n-ten Iteration, ist sein nördlicher Nachbar in der n+1-ten Iteration und sein südlicher Nachbar in der n-1-ten Iteration. Also ist der Prozess 0 am Ende in der m-ten Iteration, wobei m = Abbruchiteration + Prozessanzahl gilt.