HLR - Hochleistungsrechnen

Aufgabenblatt 9

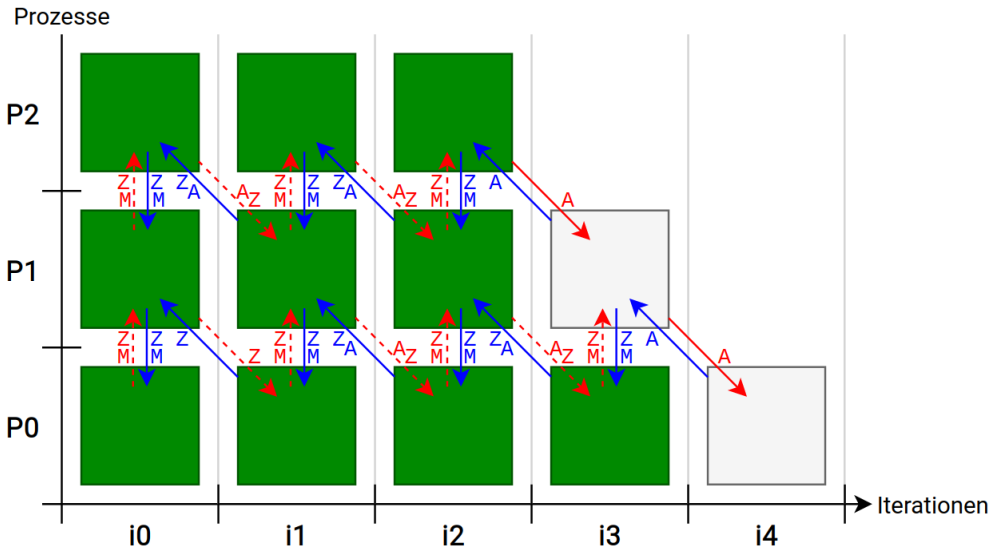
Merle Hoffmann, Joël Miramon, Max Press

1 Parallelisierung mit MPI

Erstellen Sie nun ein Parallelisierungsschema für das **Gauß-Seidel**-Verfahren.

- Visualisieren Sie das Kommunikationsschema für 3 Prozesse und 5 Iterationen. An der x-Achse werden dabei die Iterationen aufgetragen; an der y-Achse die Prozesse. Legen Sie je ein Diagramm für die zwei unterschiedlichen Abbruchbedingungen an. Überlegen Sie sich vorher, mit welchen Operationen (Punkt-zu-Punkt/kollektiv, blockierend/nicht-blockierend) Sie arbeiten möchten. Wählen sie dabei folgendes Farb- und Objektschema:
  - Grüner Block: Berechnungsphasen
  - Durchgezogene Linien (Pfeile): blockierende Kommunikation
  - Gestrichelte Linien (Pfeile): nichtblockierende Kommunikation
  - Rote Pfeile: Versenden (bei Punkt-zu-Punkt)
  - Blaue Pfeile: Empfangen (bei Punkt-zu-Punkt)
  - Pinke Linien: kollektive Operationen
  - Z am Pfeil für die zu kommunizierende Zeile
  - M am Pfeil für das zu kommunizierendes Maxresiduum

1.1 Terminierung: Hinreichende Genaugkeit



Weitere Elemente in der Visualisierung des Kommunikationsschemas:

- Annahme: nach i2 ist die hinreichende Genauigkeit erreicht
- M: bisher bekanntes maximales maxResiduum
- A: Abbruchsignal (true, false)
- weißer Block: empfängt Signal, aber startet keine Berechnungsphase

Ablauferläuterung:

Innerhalb einer Iteration laufen die Prozesse sequentiell ab, allerdings muss ein Prozess nicht auf alle anderen warten, um die nächste Iteration starten zu können.

Mithilfe vom maxResiduum wird bestimmt, ob die Abbruchbedingung erüllt wurde. Um das maximale Residuum zu erhalten, muss die komplette Matrix durchlaufen sein. Daher ermittelt nur der letzte Prozess die Abbruchbedingung und alle anderen Prozesse senden ihr vorläufiges maxResiduum an ihren nachfolgenden Prozess. Somit müssen die Prozesse nicht auf den letzten Prozess warten und können die nächsten Iterationen schon vorrechnen.

Demnach wissen diese Prozesse aber auch nicht, ob die Abbruchbedingung bereits erfüllt wurde und rechnen zu viele Iterationen vor. Die Prozesse müssen daher zusätzlich $nprocs - (rank + 2)$ vorherige Iterationsergebnisse im Buffer speichern, um auf das korrekte Ergebnis zugreifen zu können, wenn sie zu weit vorgerechnet haben.

Das Abbruchsignal wird vom letzten bis zum ersten Prozess weitergeleitet. Wenn $iteration \geq nprocs - (rank + 1)$, erwartet ein Prozess ein Abbruchsignal von dem vorherigen Prozess. Vorher erwarten die Prozesse kein Abbruchsignal, da dieses erst existiert, sobald die erste Iteration von allen Prozessen durchlaufen wurde.

Folgende in Klammern stehende Prozesse und Iterationen sind beispielhaft zu verstehen.

Ein Prozess ($P1$) kann mit der Berrechnung einer Iteration ($i1$) starten, wenn

1. er seine obere Randzeile vom vorherigen Prozess ($P0$) aus der selben Iteration ($i1$) erhält,
2. er ein negatives Abbruchsignal vom nachfolgenden Prozess ($P3$) aus der vorherigen Iteration ($i0$) erhält,
3. und er seine untere Randzeile vom nachfolgenden Prozess ($P3$) aus der vorherigen Iteration ($i0$) erhält.

Für den ersten Prozess fällt immer Punkt 1 weg, für den letzten Prozess fällt immer Punkt 2 und 3 weg.

Wenn ein Prozess alle 3 Punkte erhält, sendet er zunächst das negative Abbruchsignal weiter an den nachfolgenden Prozess und startet dann eine Berechnungsphase. Im Anschluss sendet er seine Startzeile an den vorherigen Prozess und seine Endzeile an den nachfolgenden Prozess.

Wenn ein Prozess ein positives Abbruchsignal von seinem vorherigen Prozess erhält, sendet er das Signal weiter an seinen nachfolgenden Prozess und ermittelt das korrekte Iterationsergebnis in seinem Buffer. Für das korrekte Ergebnis muss ein Prozess $nprocs - (rank + 2)$ Iterationen zurück gehen.

Für eine zeitliche Darstellung siehe Abbildung 1.

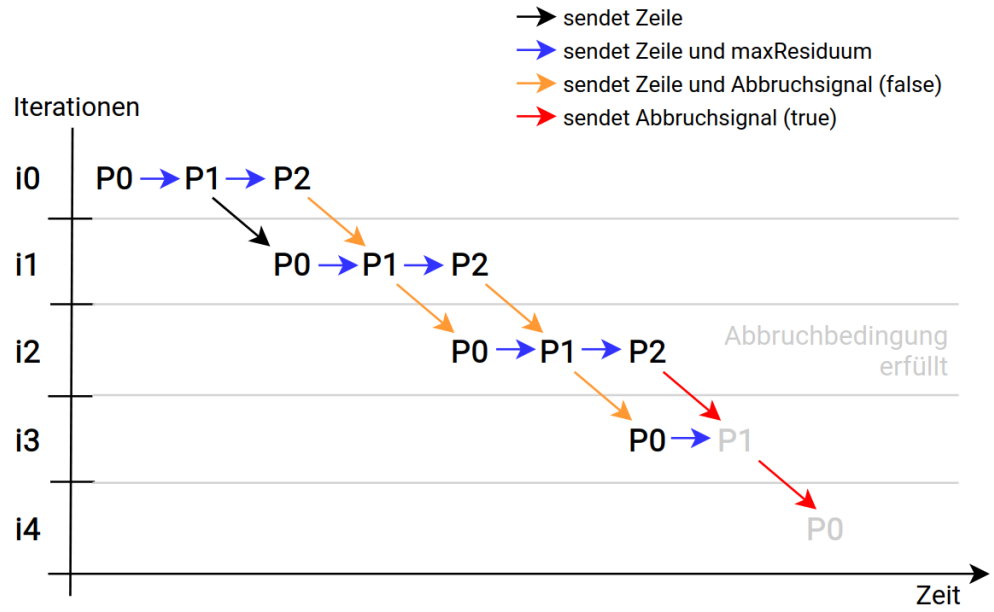
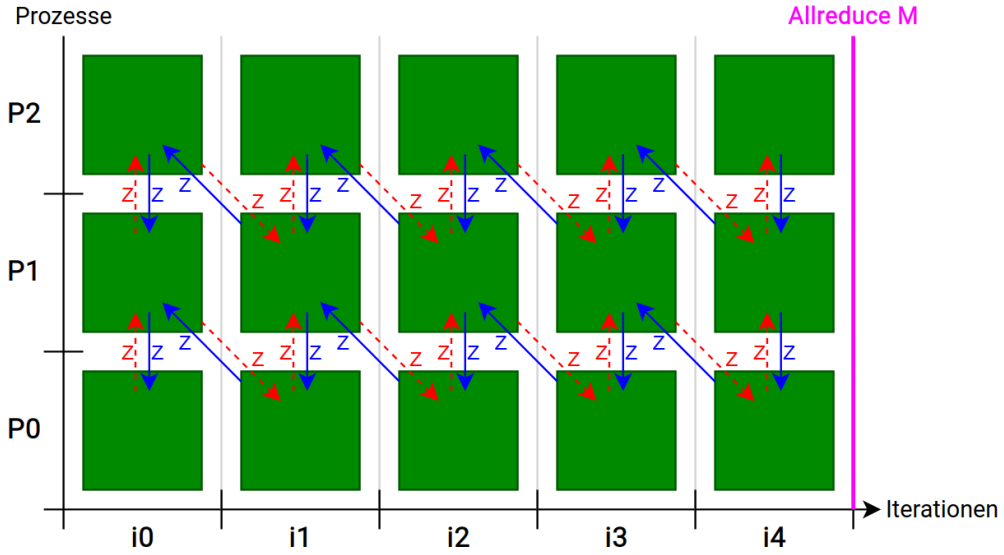


Abbildung 1: Ablaufdiagramm über Zeit für Terminierung: Hinreichende Genauigkeit

1.2 Terminierung: Anzahl der Iterationen



Ablauferläuterung:

Der generelle Ablauf ist identisch zu 1.1. Allerdings wissen hier alle Prozesse wann sie abbrechen müssen, da die Anzahl an Iterationen, die sie durchlaufen müssen, vorher feststeht. Daher müssen die Prozesse auch kein maxResiduum und Abbruchsignal versenden. Auch der Buffer für die zusätzlichen Iterationsergebnisse wird nicht benötigt. Jeder Prozess läuft einfach die richtige Anzahl an Iterationen durch und muss dafür lediglich seine Randzeilen versenden und erhalten. Nach der letzten Iteration wird mithilfe von Allreduce das maxResiduum ermittelt und an alle Prozesse gesendet.

1.3 Alternative Lösung mit anderer Matrixaufteilung

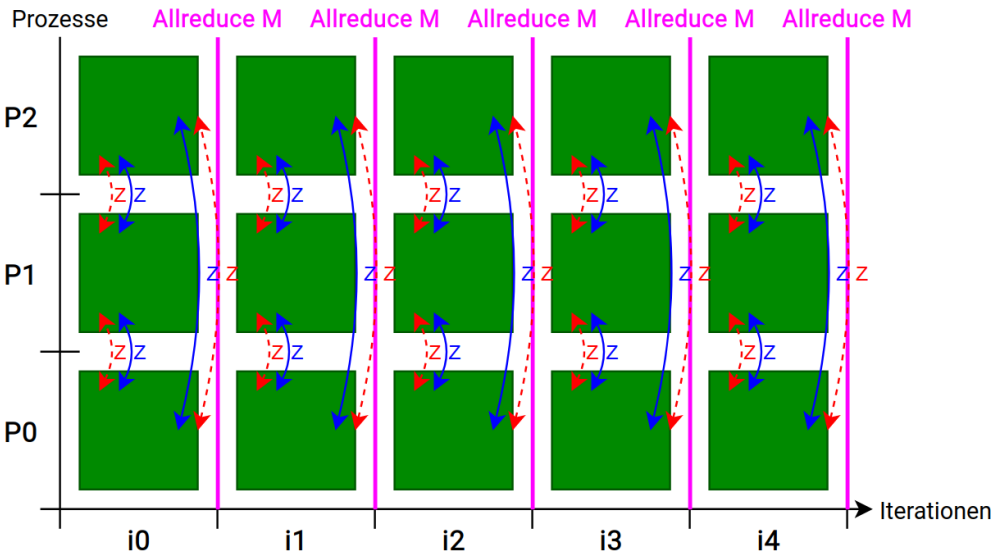
Alternativ kann man die Berechnung auch innerhalb einer Iteration parallelisieren, indem man bereits berechnete Werte versendet bevor die komplette Zeile durchlaufen wurde.

Hierfür müsste die Matrixaufteilung anders erfolgen. Eine Möglichkeit wäre es, jedem Prozess abwechselnd eine Zeile zuzuweisen. Dabei könnte man eine Zeile durch die Anzahl der Prozesse unterteilen. Wenn ein Teilabschnitt einer Zeile durchlaufen wurde, kann dieser Abschnitt an den nachfolgenden Prozess gesendet werden. Dieser kann somit schon mit der Berechnung seiner eigenen Zeile beginnen. Ein Prozess kann mit der Berechnung eines Teilabschnittes starten, wenn der Prozess den vorherigen Teilabschnitt bereits berechnet und den entsprechenden oberen Teilabschnitt des vorherigen Prozesses erhalten hat.

Beispielaufteilung mit $i = 0$, $nprocs = 3$:

	0	1	2	3	4	5	6	7	8
0	x	x	x	x	x	x	x	x	x
1	x	x	x	x	x	x	x	x	x
2	x	x	x	x	x	x	x	x	x
3	x	x	x	x	x	x	x	x	x
4	x	x	x	x	x	x	x	x	x
5	x	x	x	x	x	x	x	x	x
6	x	x	x	x	x	x	x	x	x
7	x	x	x	x	x	x	x	x	x
8	x	x	x	x	x	x	x	x	x

1.3.1 Terminierung: Hinreichende Genauigkeit

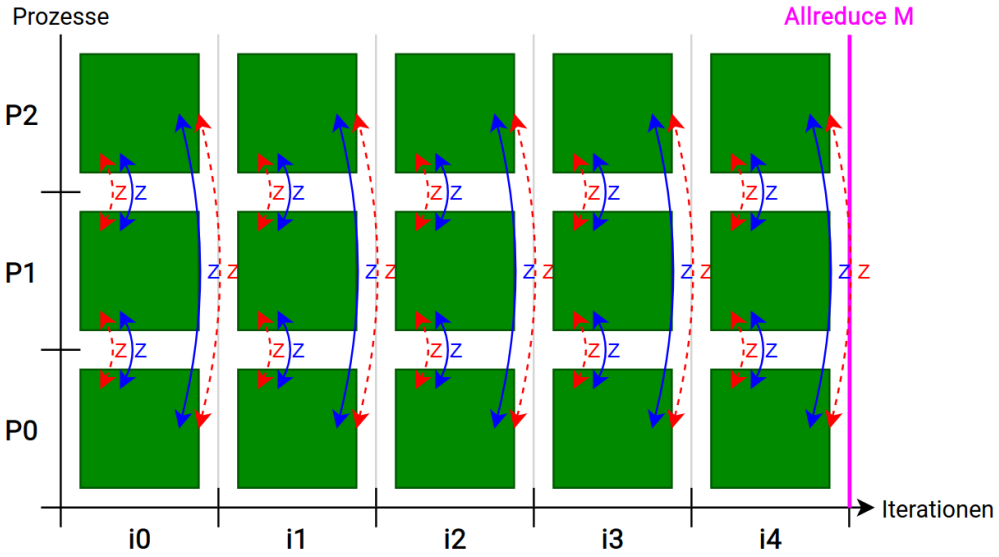


Ablauferläuterung:

Der erste und letzte Prozess müssen sich gegenseitig Teilabschnitte der Zeilen senden, da diese durch das Abwechseln der Prozesse nebeneinander liegen können.

Da die Parallelisierung nicht über die Iterationen, sondern innerhalb einer Iteration abläuft, kann man einfach nach jeder Iteration das maxResiduum mithilfe von Allreduce bestimmen und versenden. Somit kann jeder Prozess selbst bestimmen, wann die Abbruchbedingung erfüllt wurde.

1.3.2 Terminierung: Anzahl der Iterationen



Ablauferläuterung:

Der Ablauf erfolgt wie bei 1.3.1. Allerdings wird nicht nach jeder Iteration das maxResiduum geteilt, sondern nur einmal am Ende.