

Aufgabenblatt 7: Circle

Gruppe: CaesarBaurMueller

December 3, 2022

Aufgabe 1: Circle

Formel

Die Formel für die Anzahl der Elemente, welche jeder Prozess bearbeiten muss:

$$Elements_for_each_Process = \frac{N}{nprocs} \quad (1)$$

Die Formel für die Anzahl der Prozesse, die ein Zusatzelement behandeln müssen:

$$Processes_with_additional_Element = N \mod nprocs \quad (2)$$

Visualisierung

Start	9 23 1 4 1 14 18 1 6 24 7 4 17
Iteration 0	4 17 9 23 1 4 1 14 18 1 6 24 7
Iteration 1	24 7 4 17 9 23 1 4 1 14 18 1 6
Iteration 2	18 1 6 24 7 4 17 9 23 1 4 1 14
Iteration 3	4 1 14 18 1 6 24 7 4 17 9 23 1

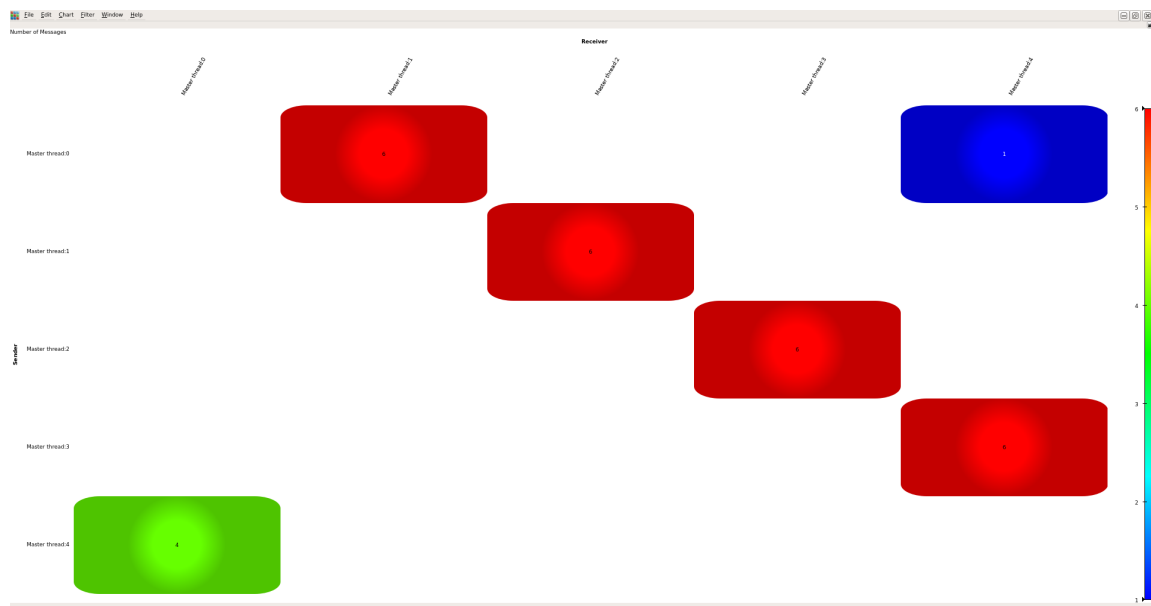
1 Aufgabe 2: Vampir

Richtung der Kommunikation

Die Richtung der Kommunikation ist in der graphischen Darstellung von Links nach Rechts zu lesen. Somit ist der linke Punkt in Vampir der Sender und der Rechte Punkt der Empfänger. Hierbei ist allerdings zu beachten, dass es keine global synchronisierte Zeit gibt und es daher etwas verzerrt wirken kann.

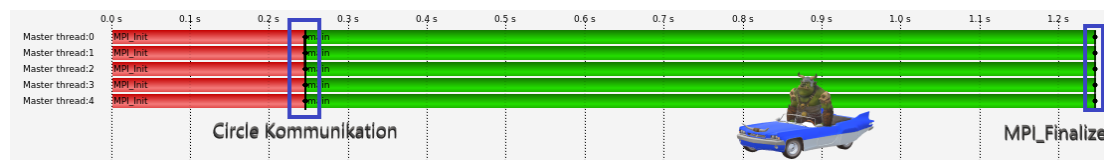
Kommunikationsmatrix

Figure 1: Kommunikationsmatrix



Programmphasen



Graphische Darstellung der Kommunikation



Dauer der INIT Phase

Die MPI Init Phase hat durchschnittlich etwa 0.245 Sekunden gedauert.

Dauer der MPI_INIT Phase

Property	Value	Value	Value	Value	Value
Display	Process Summary	Process Summary	Process Summary	Process Summary	Process Summary
Function	 MPI_Init	 MPI_Init	 MPI_Init	 MPI_Init	 MPI_Init
Function Group	 MPI	 MPI	 MPI	 MPI	 MPI
Cluster Members	Master thread:0	Master thread:1	Master thread:2	Master thread:3	Master thread:4
Accumulated Exclusive Time	0.245407 s	0.245413 s	0.245415 s	0.245419 s	0.245409 s

2 Aufgabe 3: Parallelisierung mit MPI

Formel

Unsere Formel ist die Folgende:

$$lines = (interlines * 8) + 9 \quad (3)$$

Sie ist die Summe der Zeilen, die durch die Interlines entstehen ($interlines * 8$), plus die Zeilen, die für die Darstellung der Elemente am Ende existieren (9). Die -2 aus der Berechnung auf den vorigen Blättern benötigen wir hier nicht mehr, da wir die Randzeilen in die Berechnung mit aufnehmen (wenn auch nur als Hilfswerte). Die Prozesse P_0 und P_{last} müssen sie also in ihren Speicher aufnehmen.

Für die Interlines gilt somit $i \in N_0$.

Für die Prozesse können wir bei Programmstart festlegen, welche Anzahl wir für die Berechnung verwenden wollen. Somit ist t für die Berechnung von i irrelevant. Es kann berechnet werden, wie viele Zeilen jeder Prozess berechnen muss, um eine effiziente Bearbeitung zu gewährleisten. Die Formel hierfür lautet:

$$linesperProcess = \frac{lines}{n_procs} \quad (4)$$

Es gilt $n_procs \in \mathbb{N}$. Wenn $n_procs < 1$ gewählt wird, wird dies vom Programm nicht zugelassen. Wenn die Lines durch die Prozesse nicht aufgehen, wird der Rest auf die ersten Prozesse aufgeteilt, dass ein Teil der Prozesse eine Line mehr als der Rest zu bearbeiten hat.

Sollten weniger Lines als Prozesse gewählt werden, wird der Rest der Prozesse nicht gestartet.

Es ergibt sich zusammengefasst die folgende Formel:

$$lines_per_process = \frac{(interlines * 8) + 9}{n_procs} \quad (5)$$

Schleifenköpfe

Hierfür definieren wir folgende Variable:

```
int lines_per_process = ((interlines * 8) + 9) / n_procs;
```

sowie eine Variable `int offset`, welche den Offset für die `starting_row` darstellt (da einige Prozesse mehr Zeilen haben können als andere). Daraus ergeben sich:

```
int starting_row = offset + (lines_per_process * rank) + 1;  
int ending_row = starting_row + lines_per_process + 1;
```

Rang 0		for(int i = starting_row (1); i < ending_row (6); i++)
Rang 2		for(int i = starting_row (11); i < ending_row (16); i++)
Rang 4		for(int i = starting_row (20); i < ending_row (24); i++)

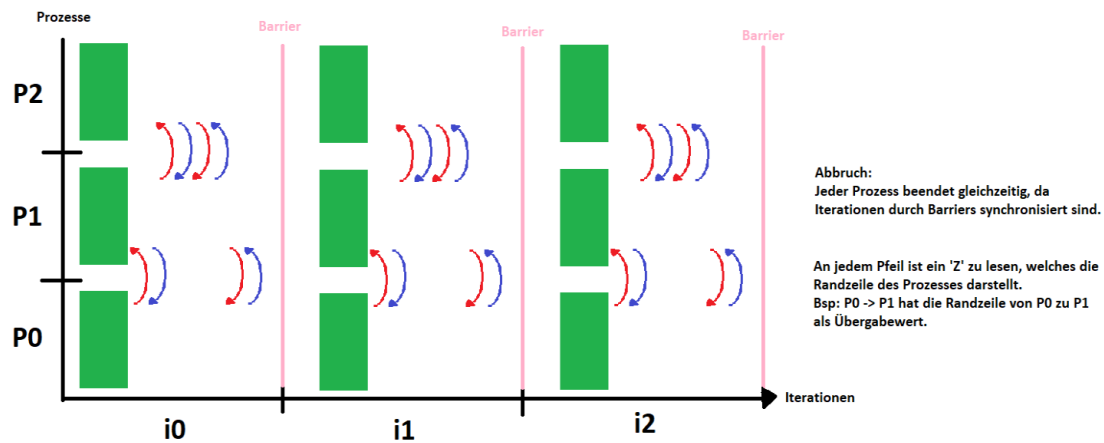
Visualisierung

[illegible]

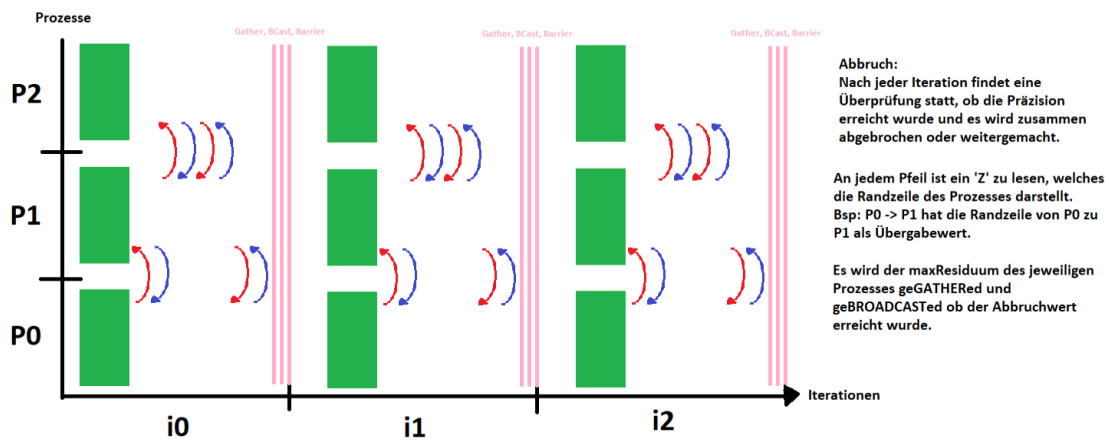
Wie der Matrix zu entnehmen ist, kommunizieren immer nur die Nachbarprozesse miteinander. Die Randelemente stehen jeweils dem Prozess zur Verfügung, der an diese Elemente anliegt. In diesem Beispiel ($i = 2$ und $n_procs = 5$) erhält jeder Prozess insgesamt fünf Zeilen. Dies schließt bei den Prozessen P_0 und P_4 die jeweilige Randzeile mit ein. Grund hierfür ist die möglichst gleichmäßige Aufteilung von Zeilen auf die Prozesse und dass die Randzeilen für die Berechnung der Matrix benötigt werden (siehe Stern). Hätten wir einen Fall in dem $n_procs - 1$ Zeilen bei der Aufteilung übrig bleiben (bspw. $i = 2$ und $n_procs = 6$), so würde der erste Prozess P_0 ebenfalls eine Zeile mehr erhalten, zuzüglich zu der Randzeile.

Kommunikationsschemata

Abbruch nach Iterationen Kollektiv: Barrier



Abbruch nach Genauigkeit Kollektiv: Gather, BCast, Barrier



Bei jeder Iteration werden die Randzeilen der Prozesse an die Nachbarprozesse gesendet. Dies birgt viel Aufwand. Schließlich gibt es nach jeder Iteration noch eine bzw. drei Kollektive Operationen, um die Prozesse zu synchronisieren.