Aufgabe 07

Can Nayci, Leonhard Rattmann, Emil Sharaf

09.12.2023

1 Circle

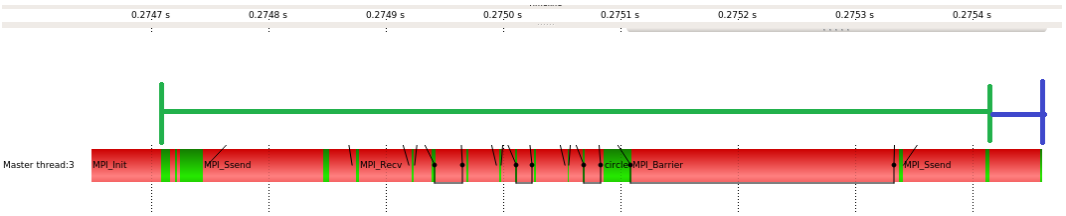
$$
\begin{aligned}
N &: \text{Anzahl der Elemente des Arrays} \\
nprocs &: \text{Anzahl der Prozesse} \\
start_p &: \text{erstes Element für Prozess } p \\
end_p &: \text{letztes Element für Prozess } p \\
start_0 &= 1 \\
end_p &= p \cdot (int)\frac{N}{nprocs} + a \\
a &= \begin{cases} \text{wenn } ((N \bmod nprocs) < p) : 0 \\ \text{sonst} : 1 \end{cases} \\
start_p &= end_{p-1}
\end{aligned}
$$

für $N = 13, nprocs = 5$:

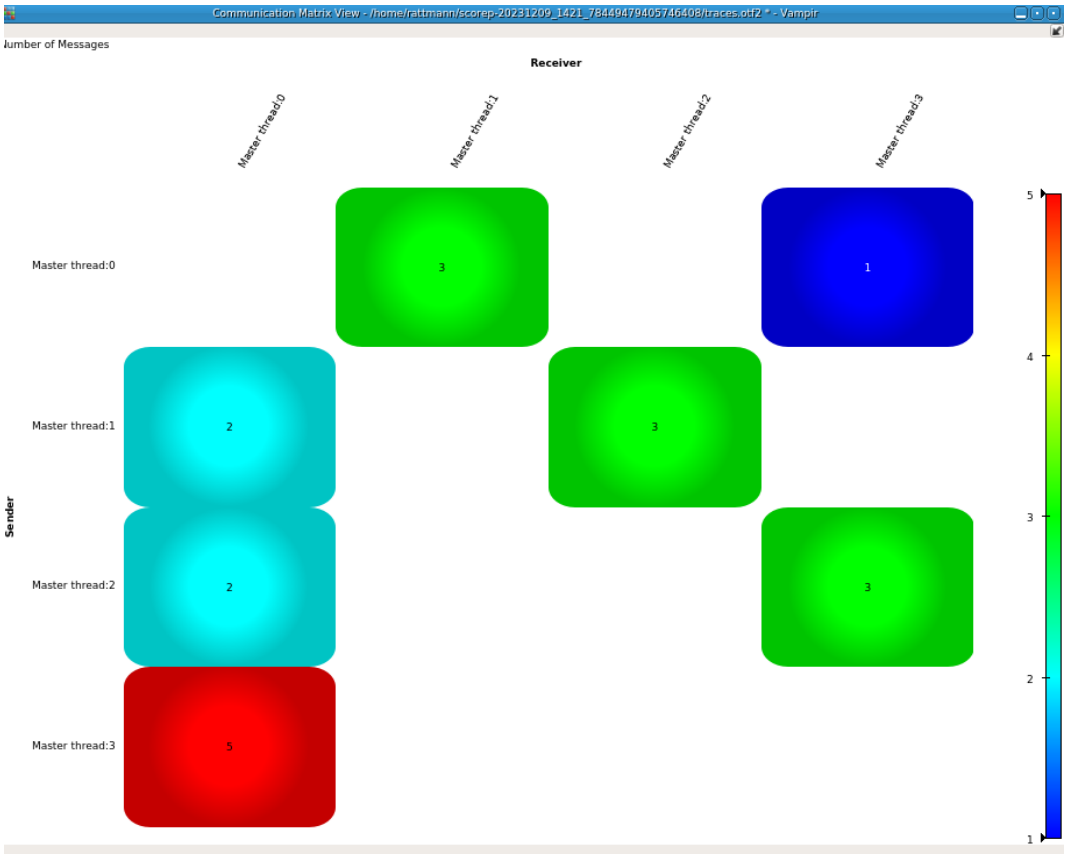
Prozess	Array-Elemente
0	$0, 1, 2$
1	$3, 4, 5$
2	$6, 7, 8$
3	$9, 10$
4	$11, 12$

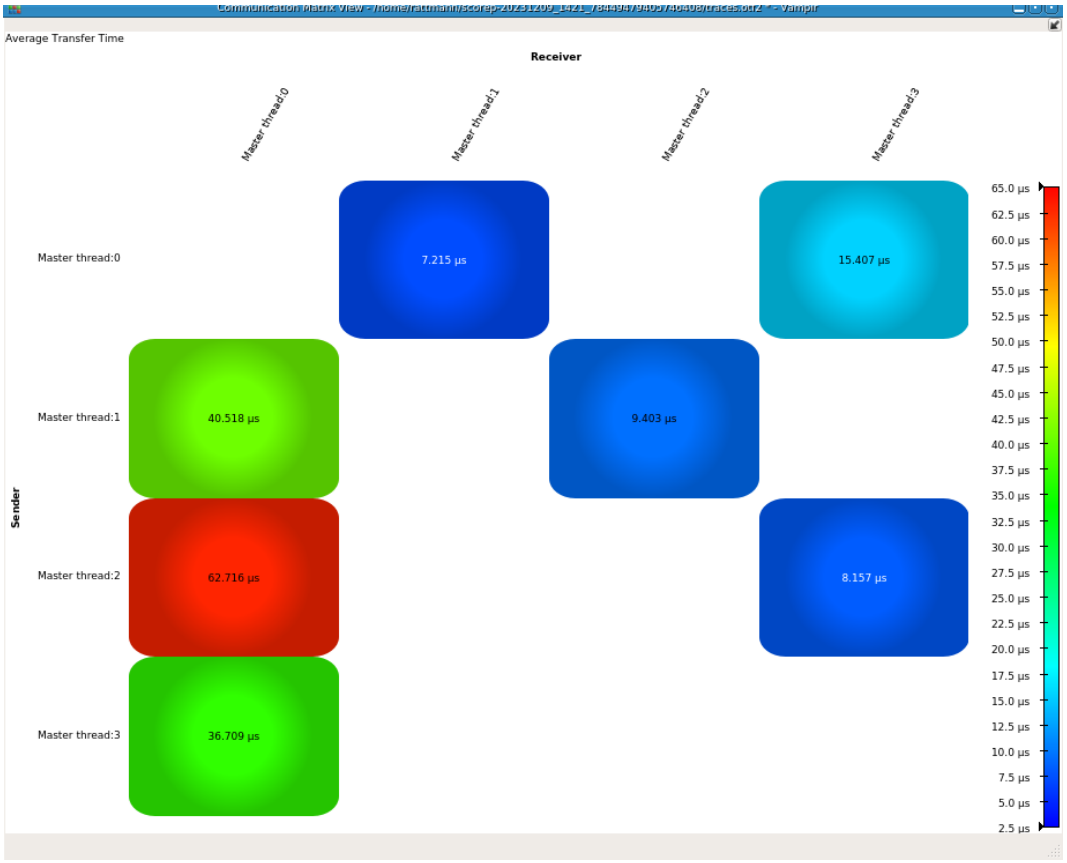
2 Visualisierung

1 **Grafische Darstellung der Kommunikation:**



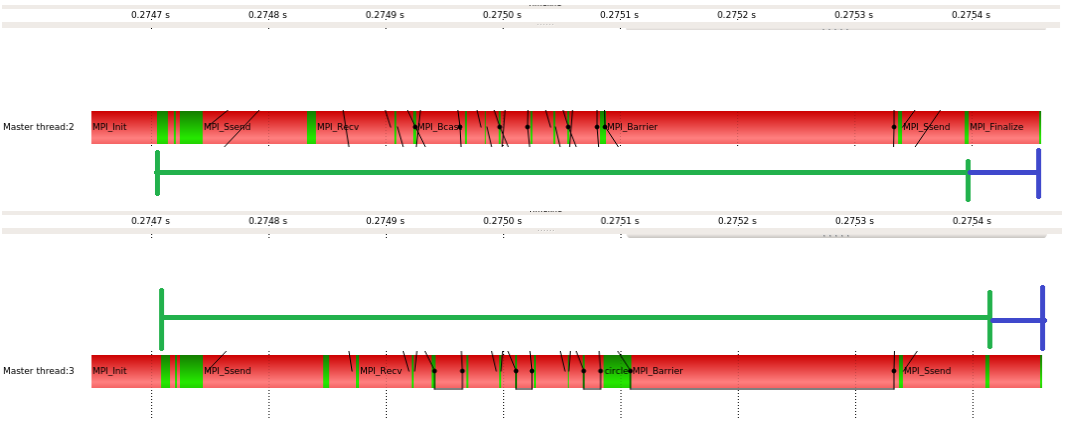
2 **Communication Matrix View:**


Communication Matrix View - /home/rattmann/scorep-20231209_1421_7844947940574640 8/traces.otf2 * - Vampir
lumber of Messages
Receiver
Master thread:0
Master thread:1
Master thread:2
Master thread:3
Sender
3
1
2
3
2
3
5
5
4
3
2
1



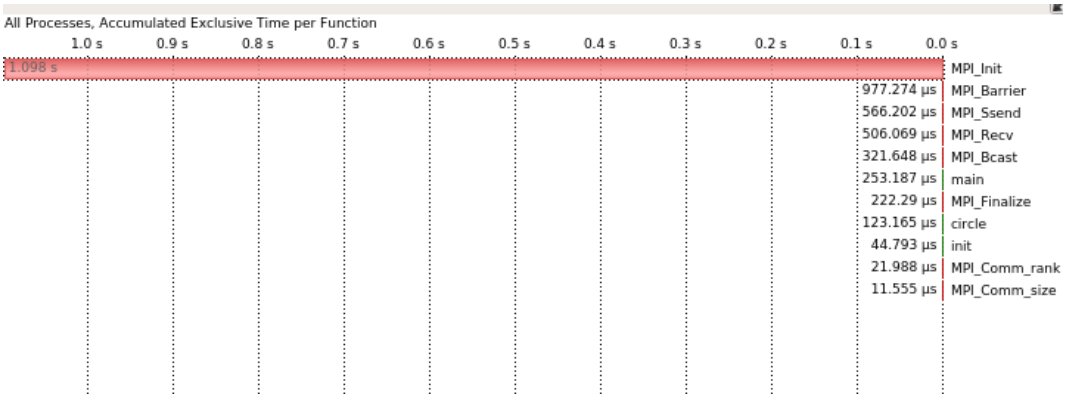
3 Programmphasen:

Die Phase bis $0,2747$ s stellt die Initialisierung dar. Die Iterationen werden in den nachfolgenden Screenshots als "grüne Phase" und das Beenden als "blaue Phase" dargestellt:



4 **MPI-Init-Phase:**

Die MPI-Init-Phase dauerte $\sim 1,098$ s und nimmt somit ungefähr 99,7% der Programmlaufzeit in Anspruch:



3 Prallelisierung mit MPI

```
int n = Anzahl der Reihen;
int nprocs = Prozessanzahl;
n = (getInterlines() * 8) + 9 - 1;
int startIndex, endIndex;

nur Prozess 0:
{
   int s_startIndex, s_endIndex;
   int lpp, lpp_rest;

   lpp = (int) (n - 1) / nprocs;
   lpp_rest = (n - 1) % nprocs;

   startIndex = 1 (0er-Reihe wird ignoriert);
   endIndex = lpp + (lpp_rest < 0)? 1 : 0);
   lpp_rest--;

   s_startIndex = endIndex;
   s_endIndex = 0;
    fuer alle nprocs exkl. 0:
   {
       MPI_Ssend(*s_startIndex an Prozess i);
       s_endIndex = startIndex + lpp;
       if (lpp_rest < 0)
       {
           endIndex++;
           lpp_rest--;
       }
       MPI_Ssend(*s_endIndex an Prozess i);
       s_startIndex = s_endIndex;
       // => startIndex ist inklusive, endIndex exklusive.
   }
}

alle anderen Prozesse:
{
   MPI_Recv(*startIndex);
   MPI_Recv(*endIndex);
}
```

z.B. 64 Interlines, 4 Prozesse:

$$n = 64 * 8 + 9 - 1 = 520$$
$$lpt = \frac{519}{4} = 129$$
$$lpt\_rest = 3$$

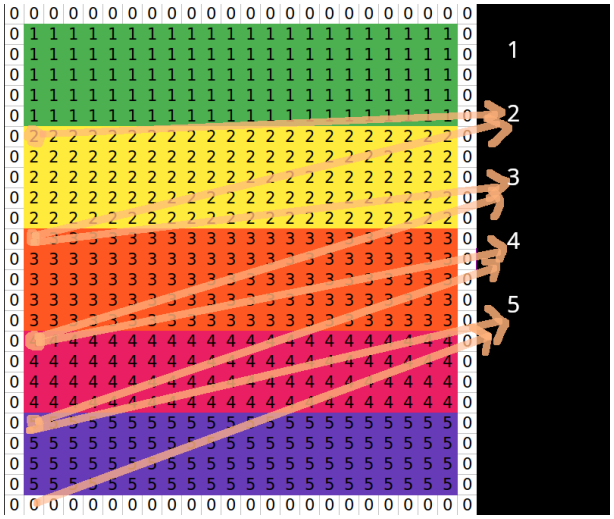
für Prozess 0:
$$startIndex = 1$$
$$endIndex = 130 + 1$$

für Prozess 3:
$$startIndex = 262$$
$$endIndex = 391 + 1$$

für Prozess m:
$$startIndex = 1 + m \cdot lpt + lpt\_rest$$

nicht $(m-1)$, weil Prozesse ab 0 nummeriert sind

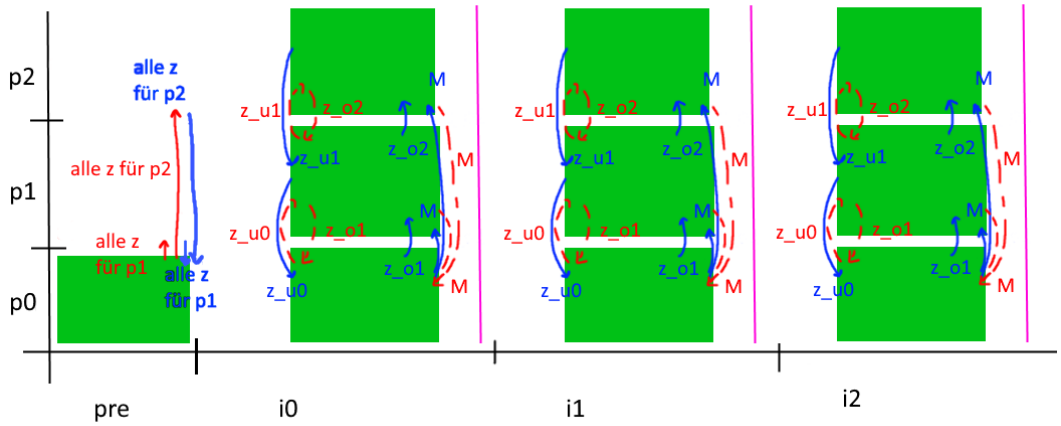
$$endIndex = 521$$



Kommunikation

1. Alle Prozesse außer 0 (bzw. 1) brauchen ihre Start- und endIndizes.
2. Jeder Prozess braucht in jeder Iteration den oben (z_u) und unten (z_o) benachbarten Teil der Matrix. Angehängt ist die Nummer des Bereichs, die der Prozess bearbeitet (abgesehen von den Prozessen mit benachbarten 0er-Reihen).
3. Jeder Prozess sendet am Anfang einer Iteration seine von anderen Prozessen benötigten Halolines und fragt am Anfang die obere benötigte Haloline ab, am Ende die untere.
4. Wenn genug Iterationen durchgeführt wurden, brechen die Prozesse eigenständig ab.

5. Am Ende (wenn alle Iterationen fertig sind) sendet jeder Prozess seine Matrix an Prozess 0 (bzw. 1), der das Ergebnis ausgeben kann. Außerdem schickt jeder Prozess sein MaxResiduum (M) an Prozess 0 (bzw. 1).



Eine Unnötigkeit bei oben beschriebenem Prozess ist, dass jeder Prozess zu jeder Iteration sein MaxResiduum an p0 schickt, damit der Präzisions-Abbruch korrekt geschehen kann. Dies kann umgangen werden, indem jeder Prozess dies eigenständig überprüüft.