Antworten

December 7, 2024

1

Jeder Teilarrays $local\_size$:

$$local\_size = \left\lfloor \frac{N}{nprocs} \right\rfloor + \begin{cases} 1 & \text{falls } rank < N \mod nprocs \\ 0 & \text{sonst} \end{cases}$$

Für $N = 13$ und $nprocs = 5$ ergibt sich die folgende Aufteilung:

Prozess 0:$\{0, 1, 2\}$, Prozess 1:$\{3, 4, 5\}$, Prozess 2:$\{6, 7\}$, Prozess 3:$\{8, 9\}$, Prozess 4:$\{10, 11, 12\}$

2

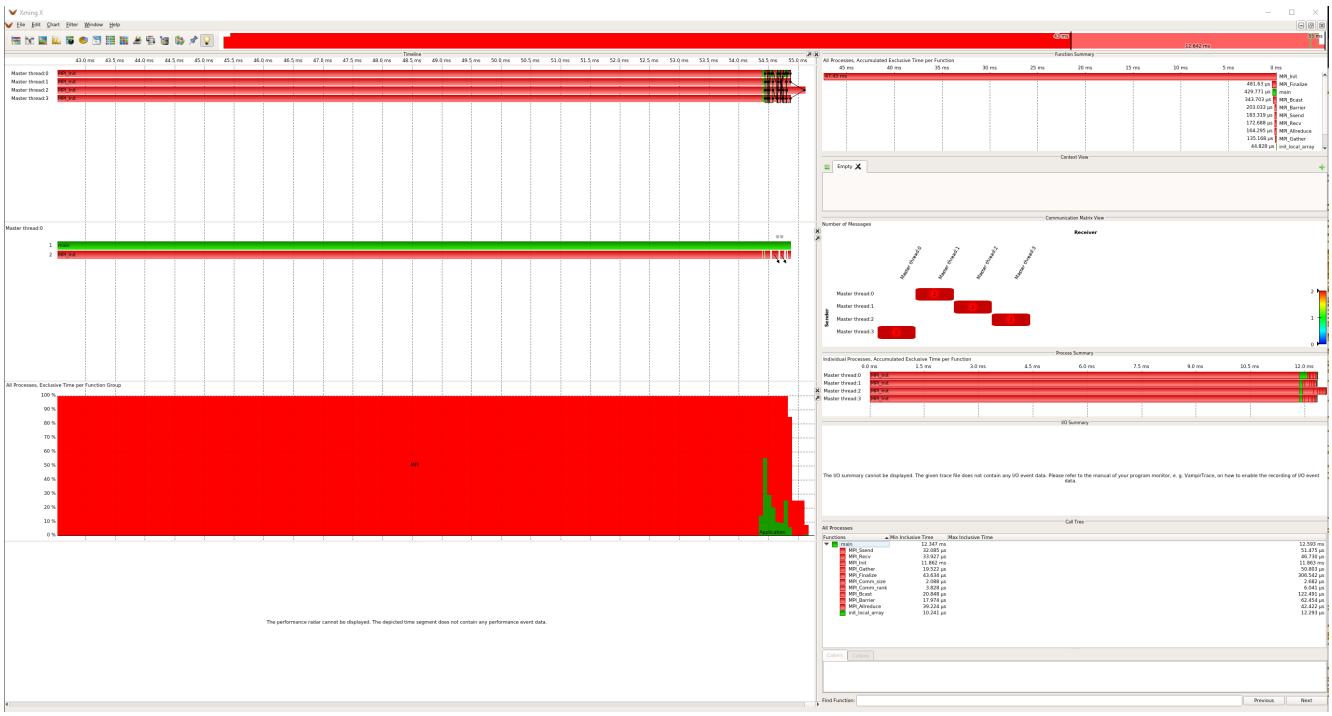
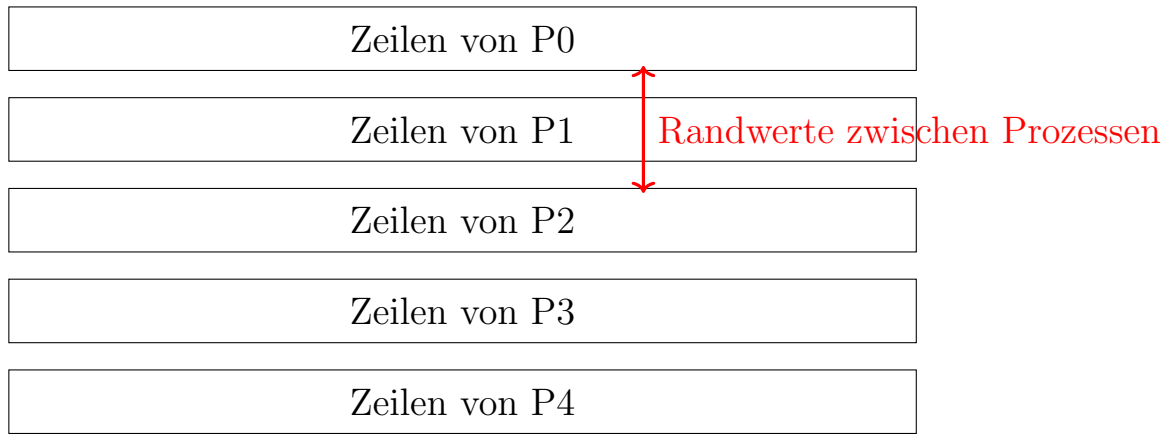


Figure 1: 2

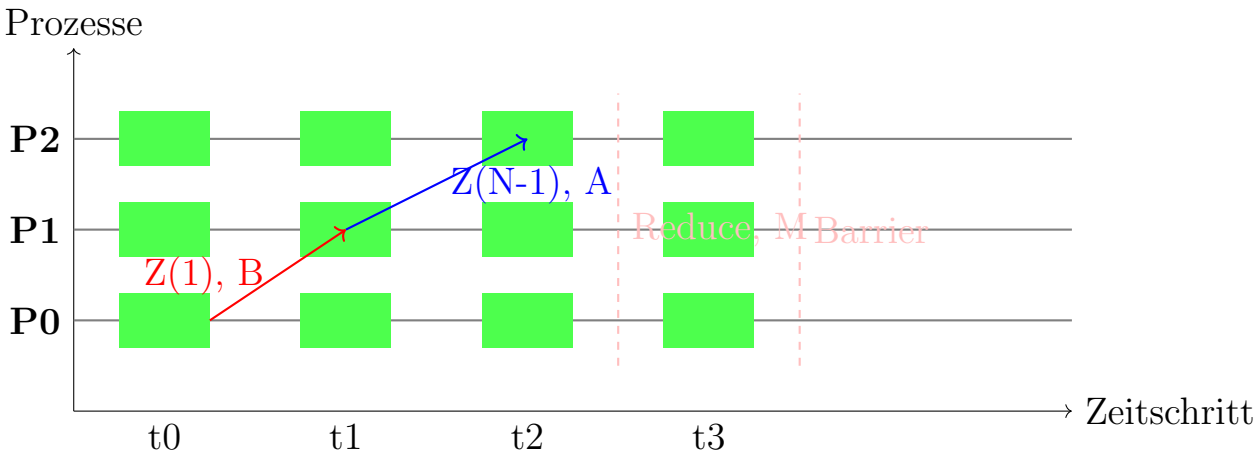
3

Die Matrix mit M Zeilen wird auf $nprocs$ Prozesse verteilt. Jeder erhält mindestens $\left\lfloor \frac{M}{nprocs} \right\rfloor$ Zeilen, der Rest wird gleichmäßig verteilt. Beispiel: $M = 20$, $nprocs = 5$.

- Prozess P0 bearbeitet Zeilen 0 bis 3
- Prozess P1 bearbeitet Zeilen 4 bis 7
- Prozess P2 bearbeitet Zeilen 8 bis 11
- Prozess P3 bearbeitet Zeilen 12 bis 15
- Prozess P4 bearbeitet Zeilen 16 bis 19



Das Diagramm zeigt die Kommunikation zwischen den Prozessen. Es werden blockierende und nicht-blockierende Kommunikation verwendet.



Herausforderungen bei der Parallelisierung:

- Lastenverteilung: Ungleichmäßige Verteilung bei nicht teilbarer Matrixgröße.
- Kommunikationsaufwand: Häufige Randwertaktualisierungen erfordern effiziente Methoden.

- Synchronisation: Barrieren und Reduktionen sind nötig, um konsistente Daten zu gewährleisten.
- Speicherzugriffe: Effizienter Umgang mit geteiltem und lokalem Speicher.

Optimierungsmöglichkeiten:

- Block-Decomposition: Matrix in Blöcke aufteilen zur Reduzierung des Kommunikationsaufwands.
- Nicht-blockierende Kommunikation: Minimierung des Warteaufwands durch gleichzeitige Berechnungen und Kommunikation.
- Reduzierung von Barrieren: Weniger Synchronisation steigert die Leistung.
- Asynchrone Reduktionen: Maximierung des Parallelismus.