Tracing Aufg 11.2

Can Nayci, Leonhard Rattmann, Emil Sharaf

24. Januar 2024

Randnotiz: Die Ausführung des Programms mit Score-P sorgt für Probleme, wie u.a. auch schon bei GS 3/2 (Ende) zu sehen. Daher wurden für die restliche Analyse die Traces von Patricks Gruppe geliehen.

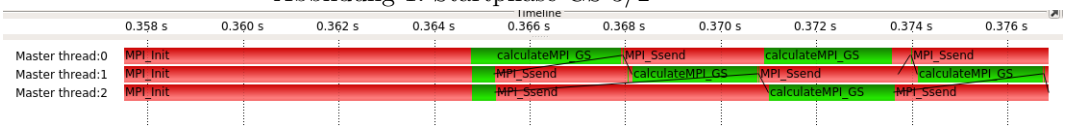
1 Startphase

Auf den Ausschnitten ist nicht abgebildet, wie die Verschiedenen Prozesse zu unterschiedlichen Zeiten *MPI_Init* starten. Womöglich liegt das an dem Hyperthreading; die ersten zwei Prozesse lägen dabei auf z.B. HW-Core 0 usw..

Bei GS 3/2 sieht man klar, wie p_0 zuerst seine Berechnung macht, p_1 damit freigibt usw., also das Treppenmodell ähnlich dem Konzept. Da JA 3/2 paralleler laufen kann und die *Ssend* von $p_0 : 2$ nicht auf $p_1 : 1$ warten muss, wie im „Stufenmodell“, sind die *Recv* und *Ssend*-Phasen deutlich kürzer. Der größte Teil bei Jacobi ist also der Rechnungsteil, bei GS die Kommunikation

Im anderen Modell wird nach *MPI_Init* erst das options-struct per Broadcast verschickt (da askParams() bei allen Prozessen läuft ist das optional). Da diese Operation bei allen Methoden gleich ist, sieht man ein ähnliches fast-dreieckiges Muster um den Broadcast. In beiden Jacobi-Läufen fangen die Recheniterationen zeitversetzt an; ebenso bei GS. Das Springen in die nächste Iteration bei GS hängt bei beiden Läufen v.a. am Warten auf die Haloline, bei Jacobi beim Austauschen des Residuums und dem Senden der Haloline.

Abbildung 1: Startphase GS 3/2



2 Synchronisation

Zur Synchronisation haben alle Varianten des Codes verschiede Ansätze: Bei unserem JA tauschen sich die Hälfte der Prozesse gleichzeitig aus, bei GS tauschen sie die Halolines aus, wenn sie benötigt werden. Beim anderen Modell werden

Abbildung 2: Startphase GS 3/2 (Patrick)

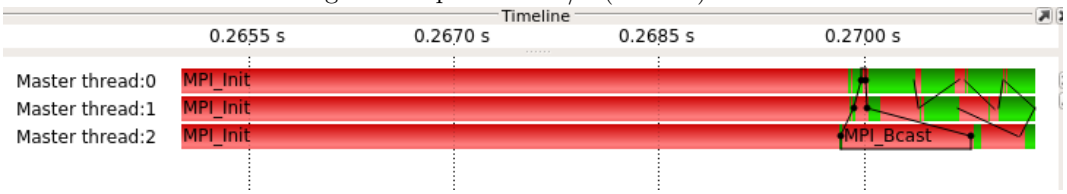


Abbildung 3: Startphase GS 5/4 (Patrick)

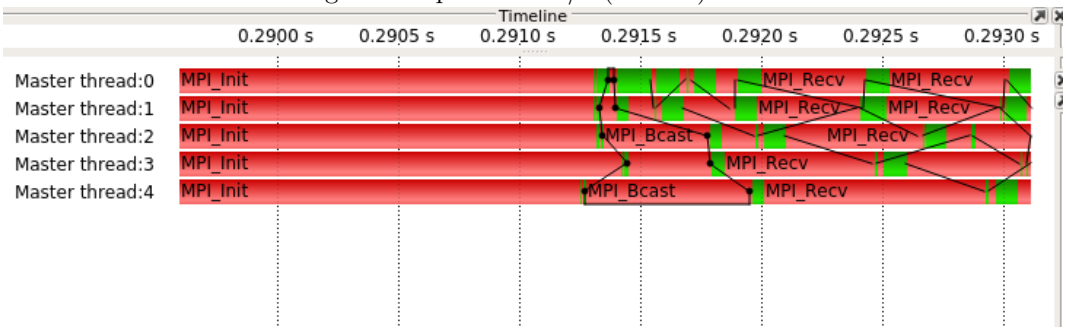


Abbildung 4: Startphase JA 3/2

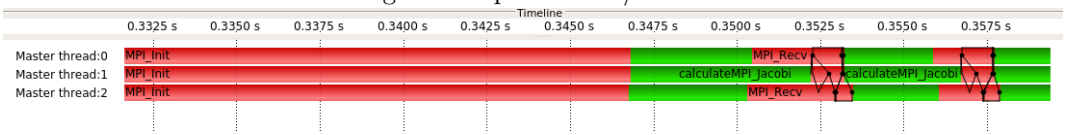


Abbildung 5: Startphase JA 3/2 (Patrick)

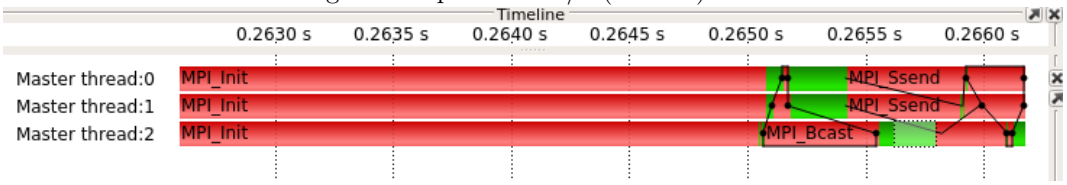
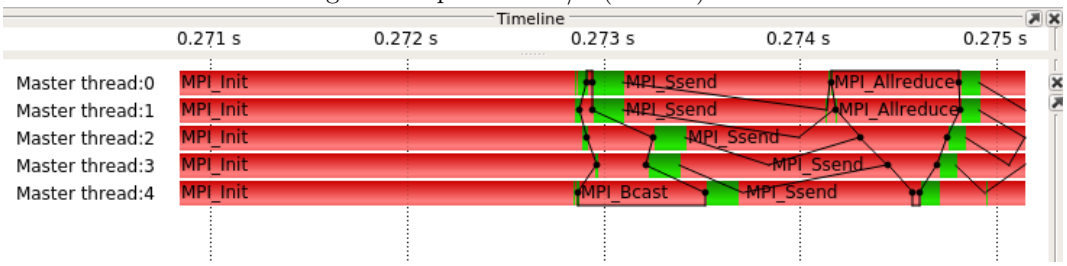


Abbildung 6: Startphase JA 5/4 (Patrick)



die Daten in JA sequentialisiert ausgetauscht, in GS ähnlich wie bei uns, mit *Issend* statt *Ssend*. Bei allen Herangehensweisen sieht man die kurzen grünen

Abschnitte, bei denen die calculate()-Methode die Kommunikation aufruft und danach darauf wartet.

Bei unserem Modell ist ersichtlich, dass Berechnungen in GS teilweise parallel stattfinden, allerdings klar in Abhängigkeit voneinander, also stufend. In Patricks Ansatz, gerade bei 5/4 sieht man eine freiere Positionierung der Rechenphasen.

Bei JA sieht man die sequentielle und parallele Kommunikation der verschiedenen Modelle, da in JA 5/4 wieder Stufen zu sehen sind (gerade bei Allreduce).

Abbildung 7: Synchronisation GS 3/2

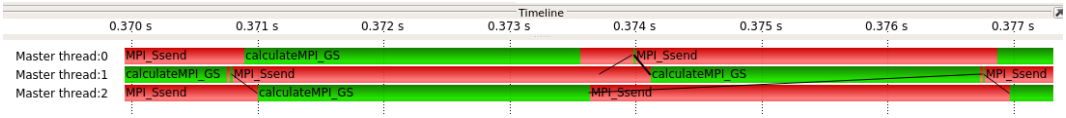


Abbildung 8: Synchronisation GS 3/2 (Patrick)

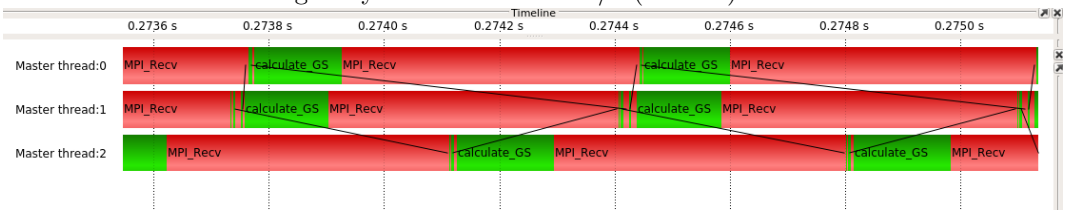


Abbildung 9: Synchronisation GS 5/4 (Patrick)

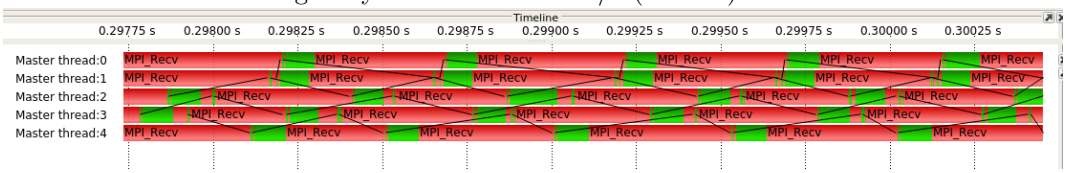


Abbildung 10: Synchronisation JA 3/2

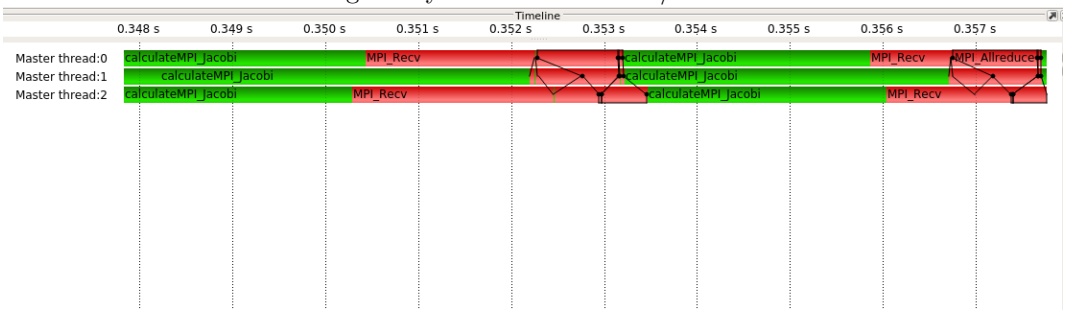


Abbildung 11: Synchronisation JA 3/2 (Patrick)

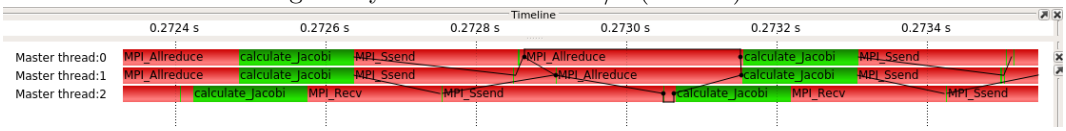
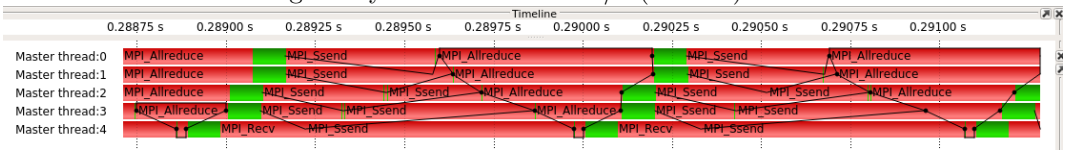


Abbildung 12: Synchronisation JA 5/4 (Patrick)



3 Endphase

Bei der Endphase sieht man bei unseren GS 3/2 offensichtlich, dass *MPI_Finalize* nicht stattfindet, evtl. wg nicht gesäuberten Issends. Da hier aber nicht einmal displayMatrix() angezeigt wird (aber definitiv ausgeführt wird), ist es wahrscheinlicher, dass die Spurdaten verfälscht sind.

Die DisplayMatrix()-Kommunikation ist bei allen Modellen wieder identisch. Bei GS 5/4 sieht man interessanterweise eine Kreuzung der Sendungen, allerdings nicht bei JA 5/4. Eventuell gibt es hier einen Zusammenhang zwischen der *MPI_Finalize*-Zeit, die bei GS deutlich unregelmäßiger ist als bei JA (auch mit den verschiedenen Skalierungen).

Abbildung 13: Endphase GS 3/2

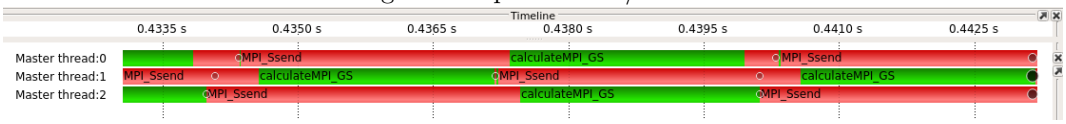


Abbildung 14: Endphase GS 3/2 (Patrick)

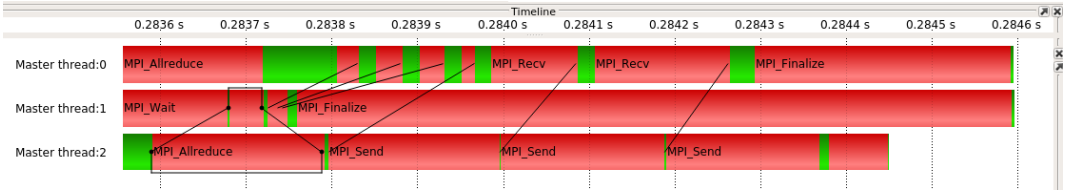


Abbildung 15: Endphase GS 5/4 (Patrick)

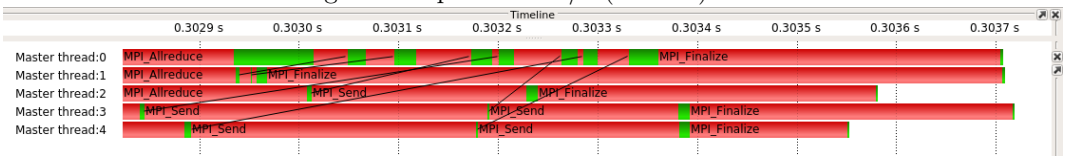


Abbildung 16: Endphase JA 3/2

Abbildung 17: Endphase JA 3/2 (Patrick)

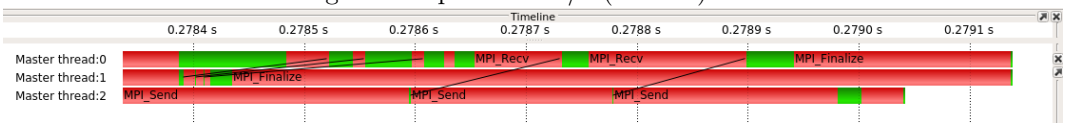


Abbildung 18: Endphase JA 5/4 (Patrick)