# Lösungen zu Aufgabenblatt 11

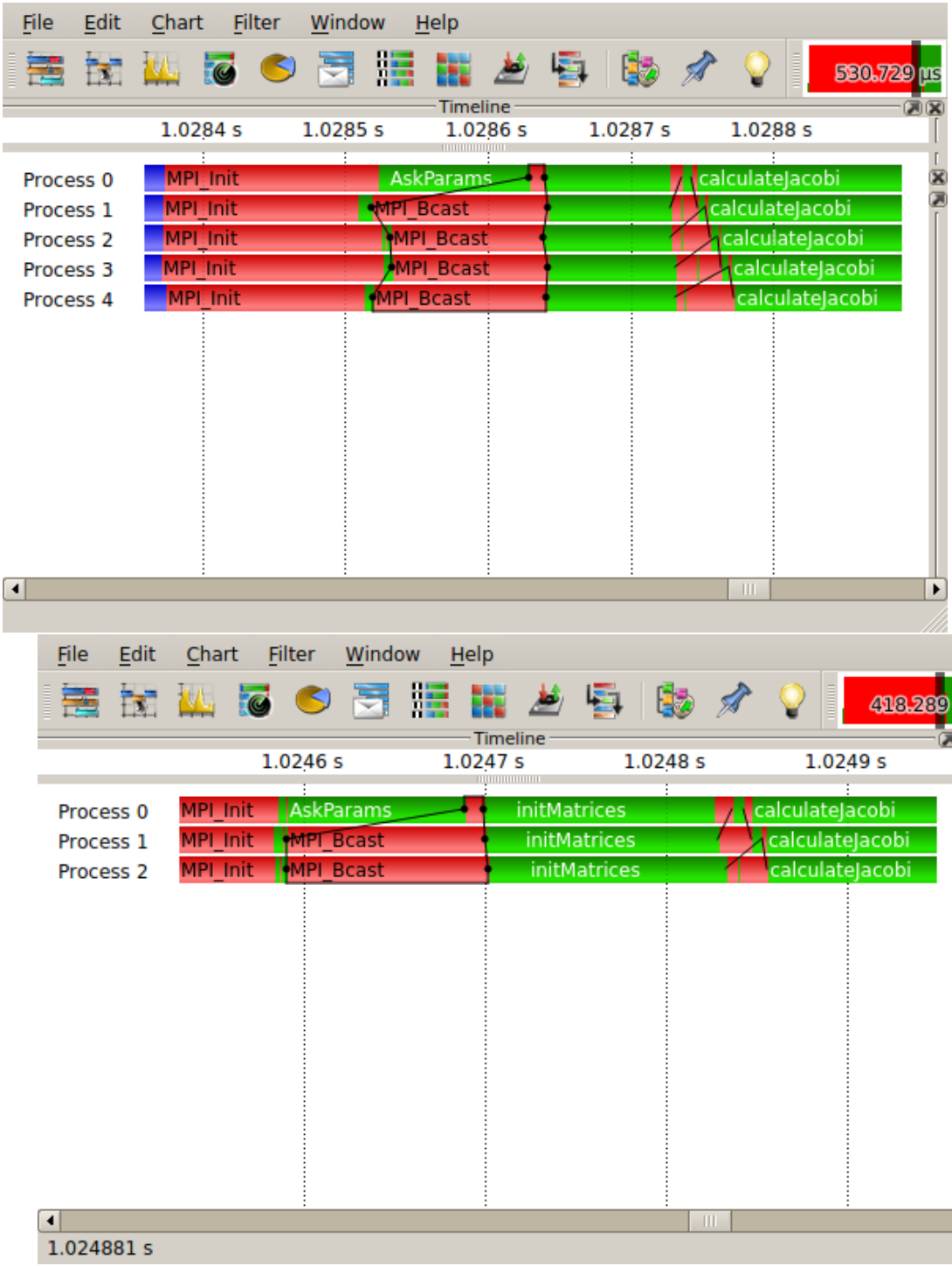
Merlin Koglin, Timon Vosberg, Merlin Steuer

Abgabe: 23. Januar 2016

## Inhaltsverzeichnis

# 1 Jacobi

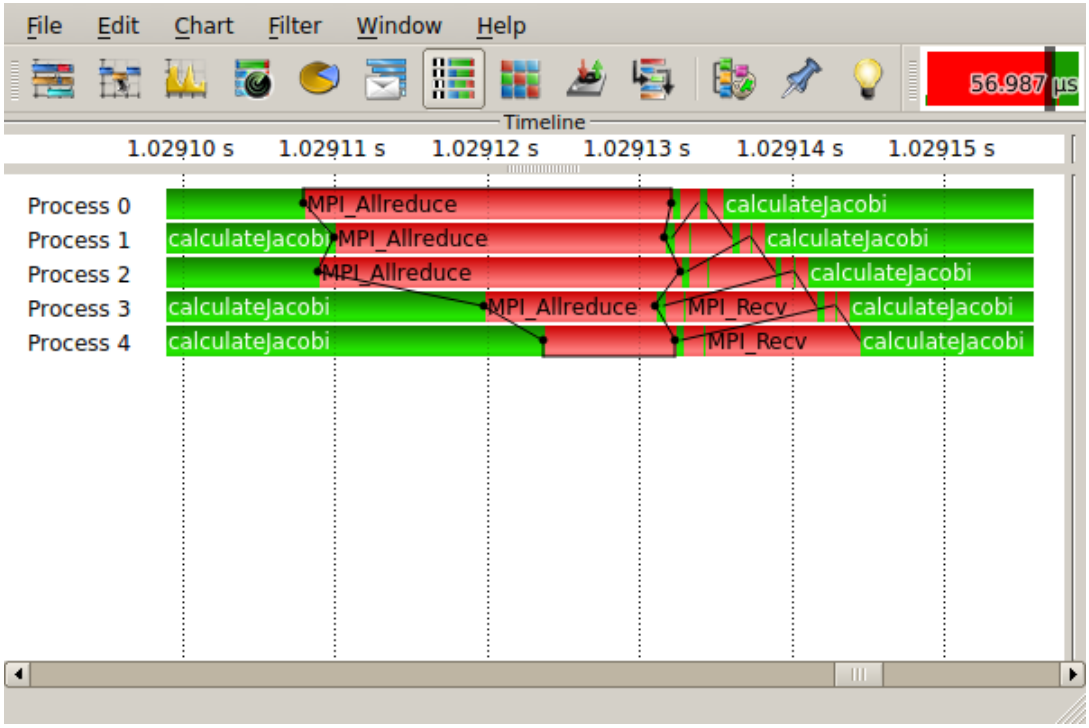
## 1.1 Start

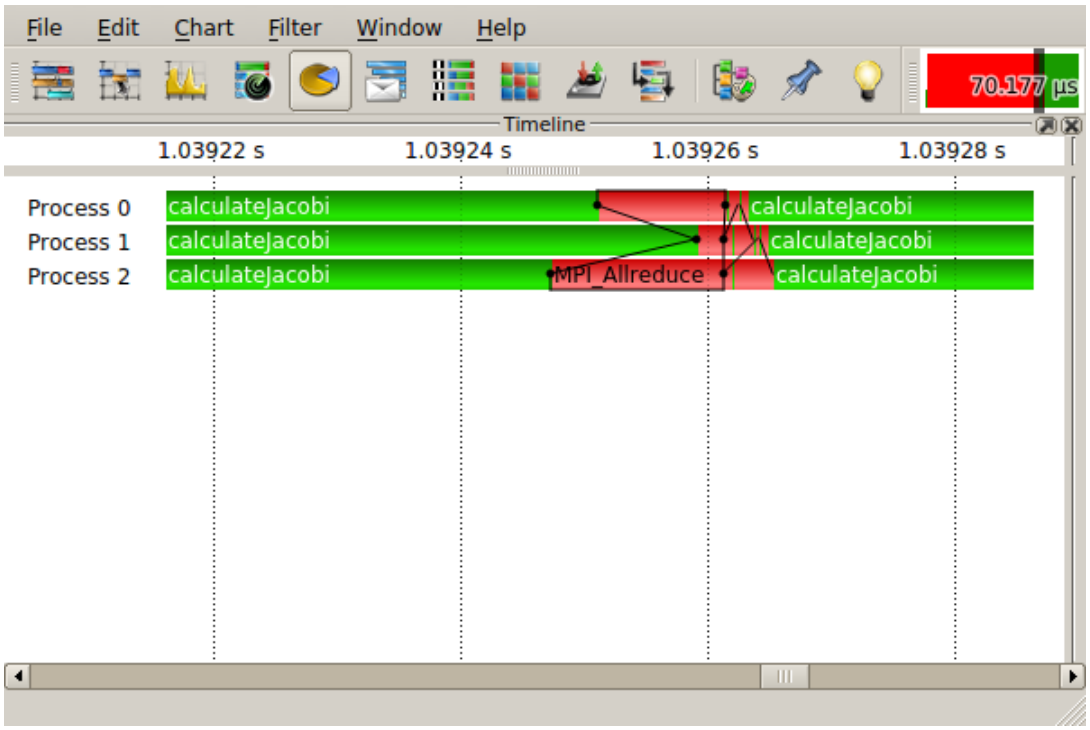



Man sieht schön, wie nach dem Abfragen der Eingabeparameter alle Pro-

zesse gleichzeitig Speicher für die Matrizen reservieren und dann etwas zeitversetzt starten, nachdem alle Zeilendaten mit Vorgängern bzw. Nachfolgern ausgetauscht worden sind.

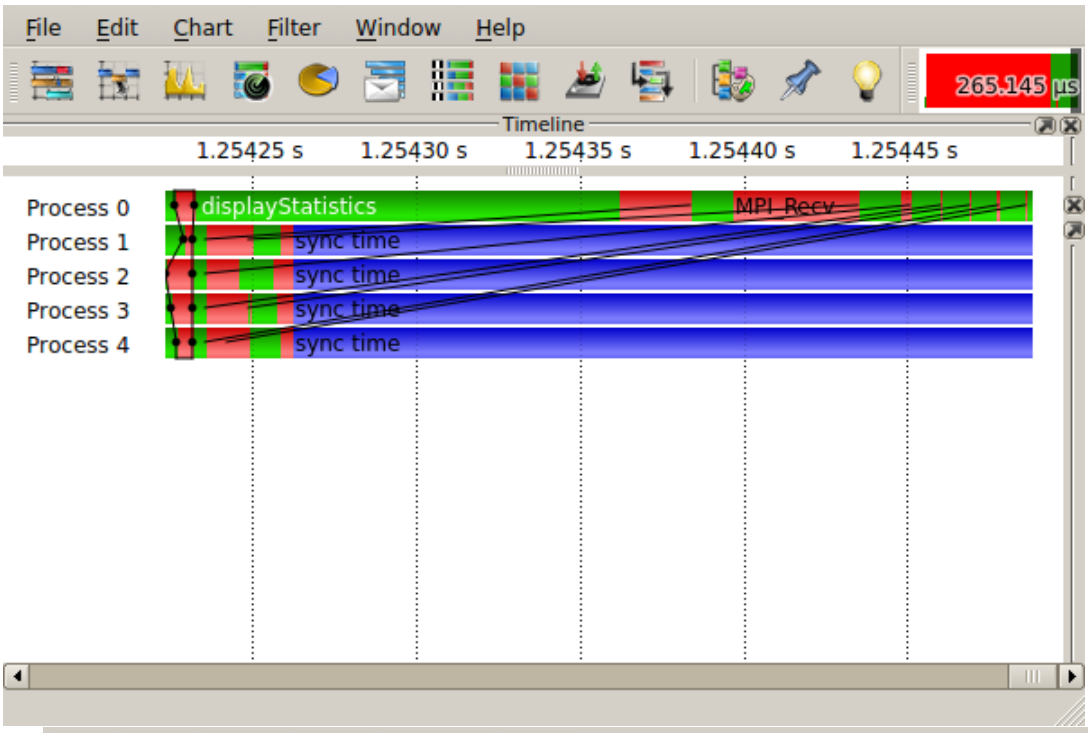
## 1.2 Synchronisation

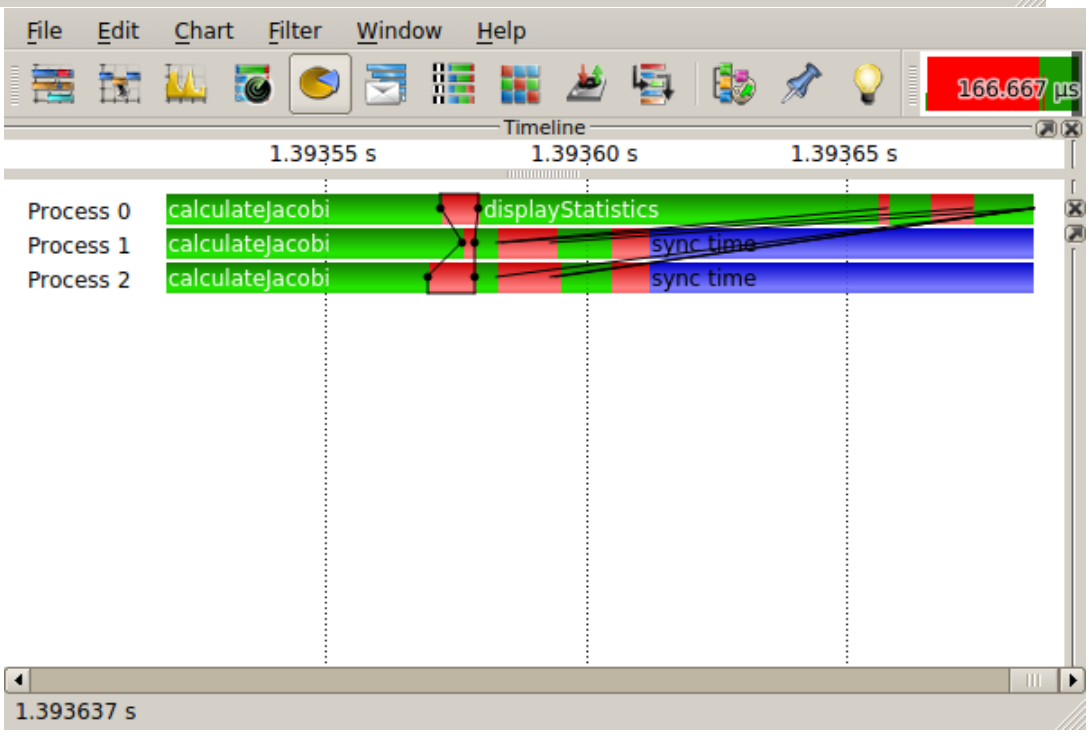





Man erwartet, dass alle Prozesse gleichzeitig mit dem Berechnen einer Iteration fertig sind. Dies ist jedoch nicht der Fall, entsprechend muss der schnellste Prozess relativ lange auf das Allreduce der residuums warten. Sobald alle Prozesse geantwortet haben beginnt wieder der Zeilenaustausch. Bis auf die kleinen Abweichungen geschieht auch hier alles wie erwartet.

## 1.3 Ende

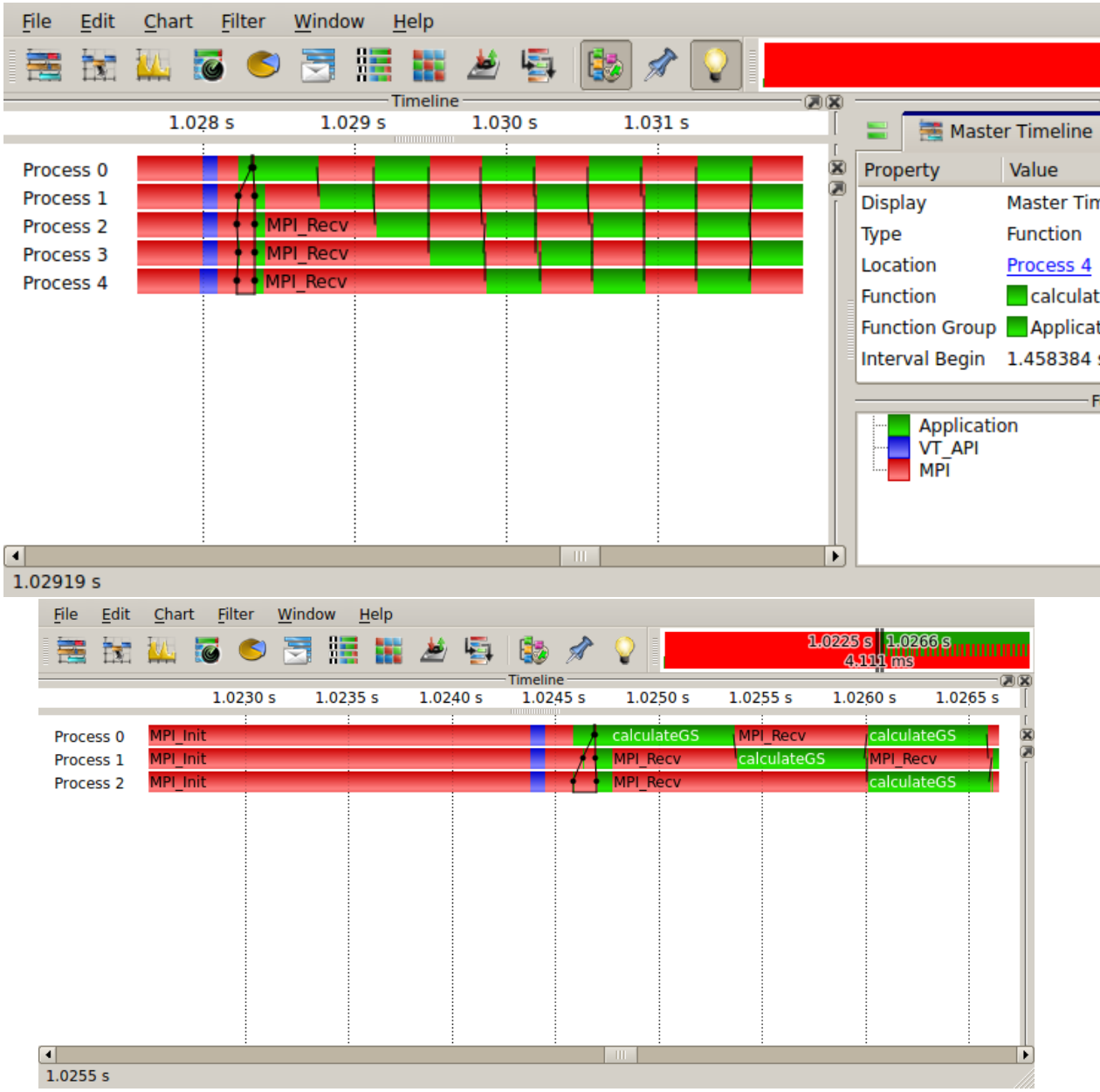



Ganz zum Schluss passiert beim Jacobi noch eine letzte Allreduce, nach der der Abbruch entschieden wird. Alle Prozesse senden ihre Daten zur Anzeige an den Master-Prozess.

# 2 Gauß-Seidel

## 2.1 Start

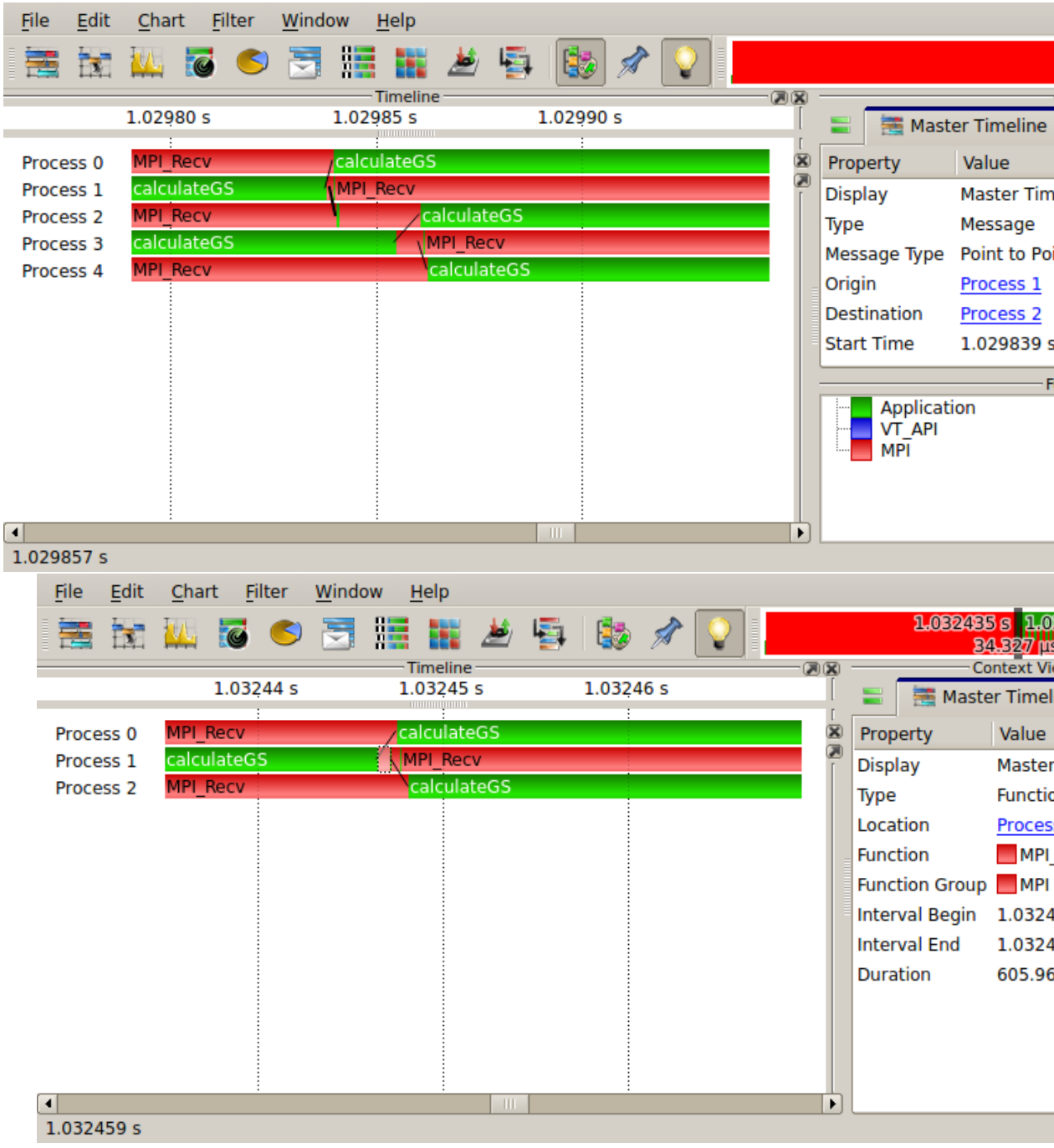



Auch hier beginnen alle Prozesse gleichzeitig mit dem Allokieren von Speicher für die Matrizen. Danach sieht man hier sehr schön, wie die Pipeline aufgebaut wird.

Diese Bilder erklären sehr eindrücklich, wieso unsere Gauß-Seidel-Implementation immer etwa doppelt so lange gebraucht hat wie die Jacobi-Implementation. Die Daten der ersten Zeile des Nachfolgeprozesses werden erst empfangen, sobald dieser die komplette Iteration berechnet hat. Damit kann ein Prozess niemals

parallel zu seinen direkten Nachbarn berechnen. Abhilfe würde hier schaffen, mittels eines ISend-Befehls die erste Zeile direkt an den Vorgänger zu schicken, sobald diese fertig berechnet ist.

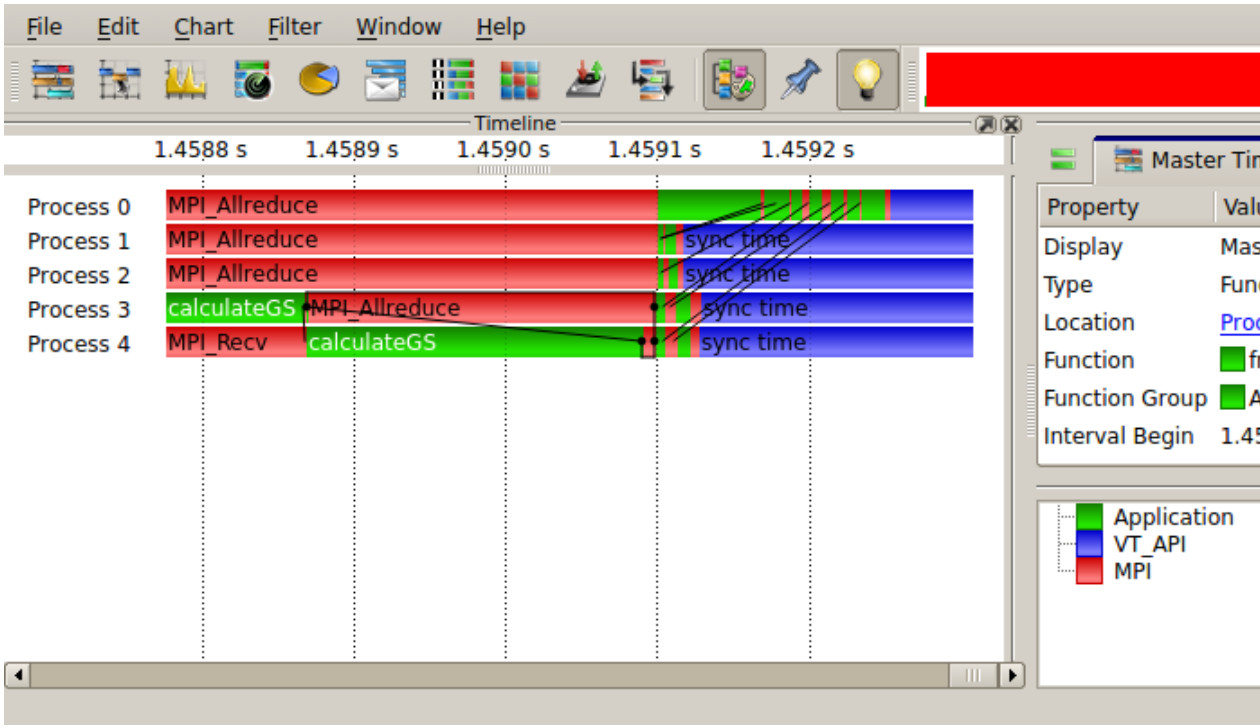
## 2.2 Synchronisation

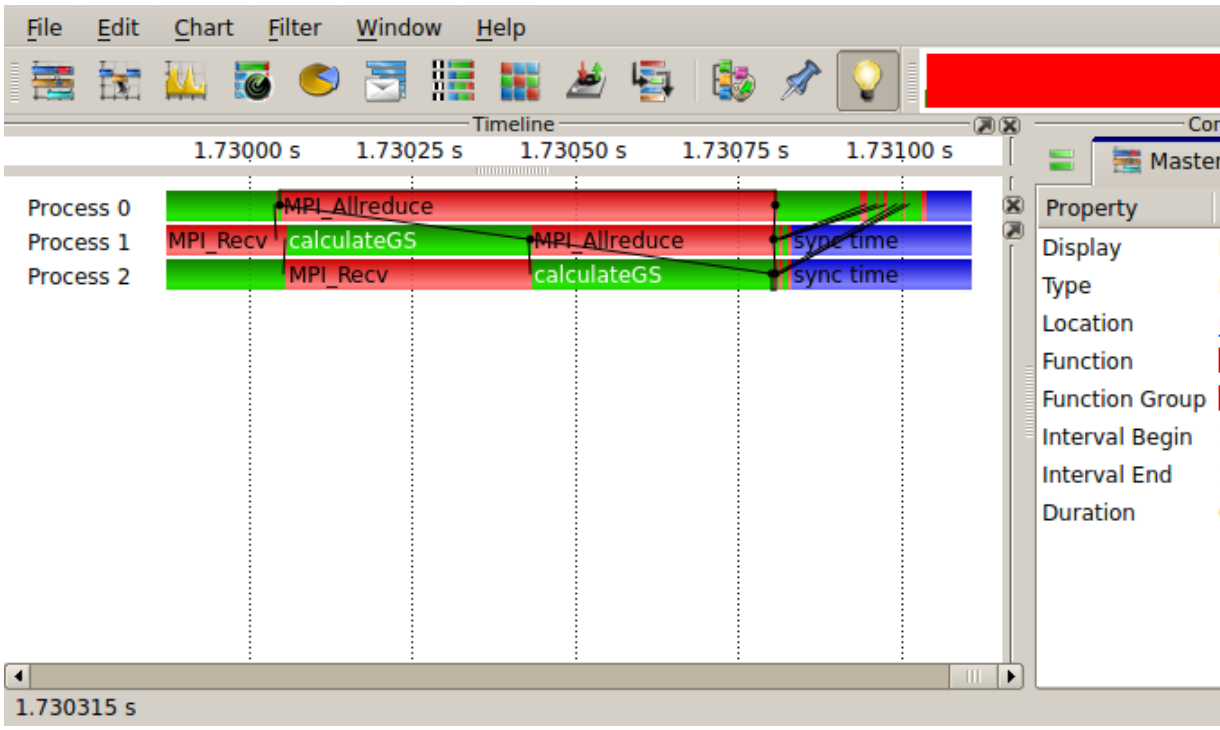



Dies setzt sich hier natürlich fort, man sieht wie die erste Zeile des Datensat-

zes eines Prozesses an den Vorgänger geschickt wird und direkt daraufhin der Vorgänger los rechnet. Danach wird die letzte Zeile zum Nachfolger geschickt, damit dieser die nächstem Iteration rechnen kann. Der Gegenseitige Ausschluss eines Prozesses zu seinen Nachbarn ist hier noch ein mal deutlich visualisiert.

## 2.3 Ende






Gut zu sehen ist hier, dass alle Prozesse auf den letzten warten müssen, um die Allreduce-Funktion zu beenden und außerdem die Statistiken wieder verteilt auszugeben. Ansonsten ist der Ablauf wie erwartet.