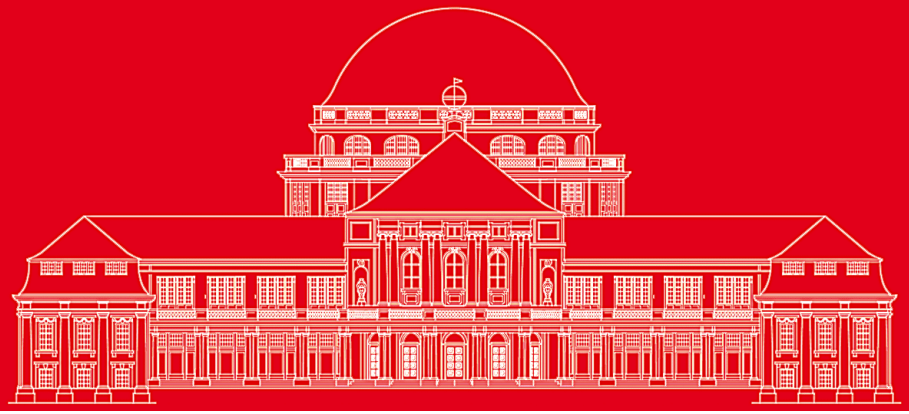
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Praktikum: Paralleles Programmieren für Geowissenschaftler

Prof. Thomas Ludwig, Hermann Lenhart



Dr. Hermann-J. Lenhart
hermann.lenhart@zmaw.de

MPI Einführung I:

- Hardware Voraussetzung zur Parallelen Programmierung
- MPI Nachrichtenaustausch
- MPI Nachrichtenaustausch in der Modellierung

Möglichkeiten der Parallelen Programmierung :

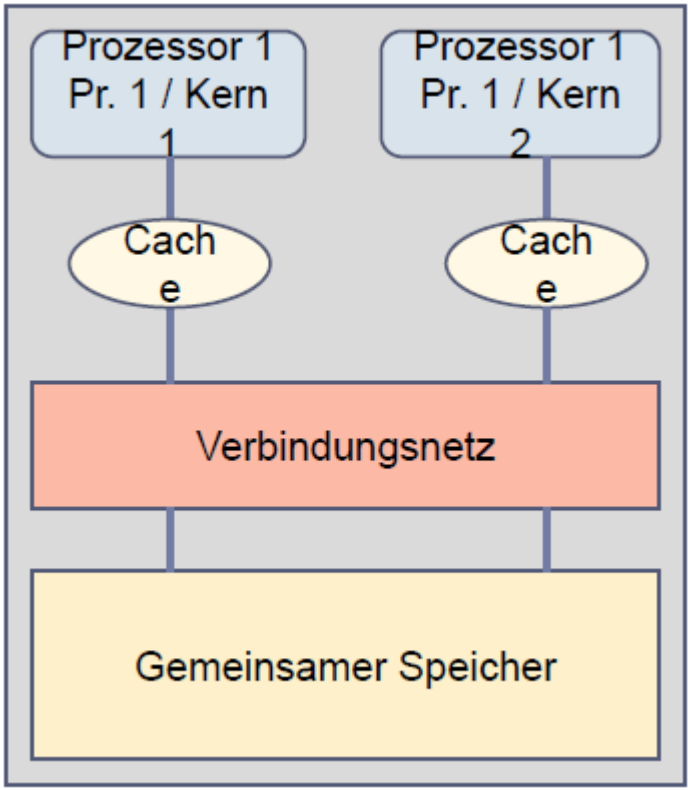
Generell abhängig von der Hardware:

- **OpenMP** - Möglich bei der Nutzung von gemeinsamem Speicher
  (shared memory directives)

- **MPI** (Message-Passing Interface)
  - bei Rechnerarchitektur mit verteiltem Speicher
  - derzeit einziger Standard mit Portabilität auf allen Plattformen

- **Hybride** Programmierung: Kombination von MPI und OpenMP

OpenMP - Hardware Voraussetzung ist gemeinsamer Speicher



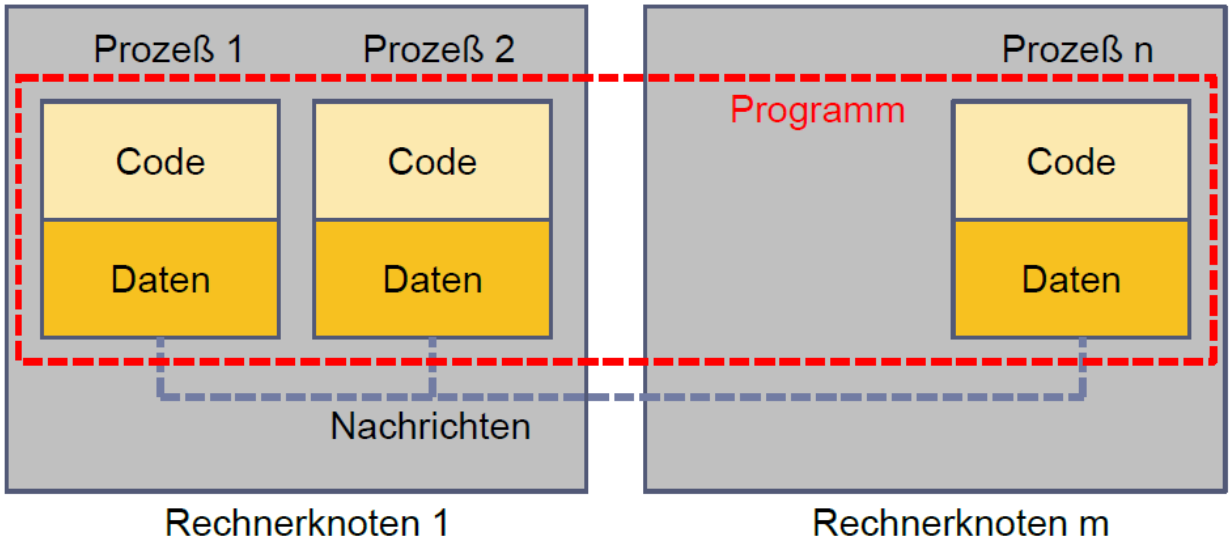
SMP:
Symmetrisches Multiprozessersystem
(symmetric multiprocessing)

Auf die Daten im gemeinsamem Speicher kann jederzeit von jedem Prozess aus zugegriffen werden.

(nach Ludwig WS12/13)

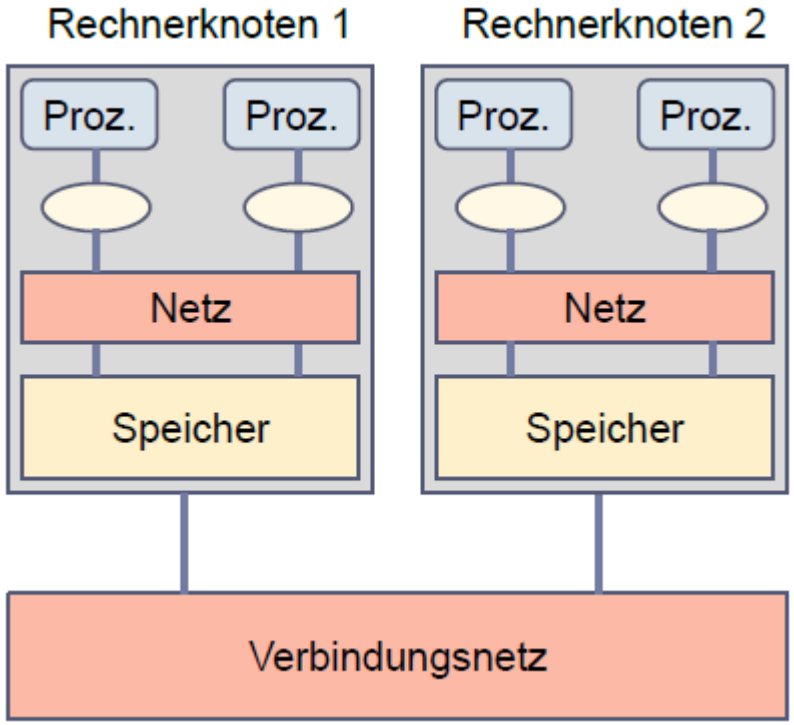
MPI – Hardware Vorraussetzung

(nach Ludwig WS12/13)



- Keinen direkten Zugriff auf Memory (Daten) von anderen Prozessen.
- Datenverfügbarkeit über expliziten Datenaustausch (Senden/Empfangen) mit anderen Prozessen!

Hybride Programmierung



Existierende HLR sind heute meist
eine Kombination aus
Rechnerknoten mit gemeinsamem Speicher
von den man viele verwendet
und **über ein Verbindungsnetz** verbindet.

Ermöglicht die Kombination von
MPI und OpenMP

(nach Ludwig WS12/13)

MPI (Message Passing Interface) - Nachrichtenaustausch

MPI Nachrichten sind Datenpakete die zwischen Prozessen ausgetauscht werden.

Hardware Rahmenbedingungen:

- Keinen direkten Zugriff auf Memory (Daten) von anderen Prozessen.
- Datenverfügbarkeit **nur** über expliziten Datenaustausch (Senden/Empfangen) mit anderen Prozessen!

!! Vorteil: MPI Prozesse lassen sich skalieren !!

MPI Nachrichtenaustausch

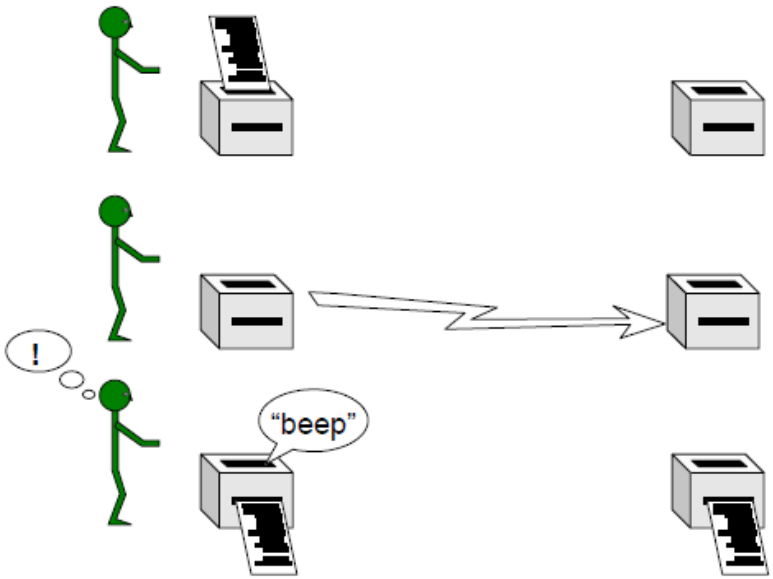
Der MPI Nachrichtenaustausch im Vergleich zum FAX

Für einfachste Art der MPI Kommunikation - Point to Point Communication:

(Wolfgang Baumann ZIB, 2009; Parallel Programming with MPI)

Ein Prozess sendet eine Nachricht zu einem Anderen,

und erhält Information über die ordnungsgemäße Zustellung (!)



MPI Nachrichtenaustausch

Der Nachrichtenaustausch bedarf folgender Informationen:

- Sendender Prozess
- Datentyp
- Datenlänge
- Empfangender Prozess
- Status der Nachricht
- Nachrichtenumgebung (z.B. wieviele Prozesse sind vorhanden?)

MPI Send/Receive Syntax I

MPI_SEND(Message, Count, Datatype, Dest, Tag, Comm, Ierror)

z.B:

Call MPI_SEND(temp, 1, MPI_Real, dest, tag, MPI_COMM_World, Ierror)

temp	Adresse des Sendepuffers; Real :: temp
1	Count – Anzahl der Elemente im Puffer
MPI_Real	Datentyp des gesendeten Elementes
dest	Angabe des Ranges des Zielprozesses; integer :: dest
tag	Nachrichtenkennung; integer :: tag
MPI_COMM_World	Kommunikator (Gruppe, Kontext)
Ierror	Fehlerstatus; integer :: Ierror

MPI Send/Receive Syntax II

Match zwischen Send und Receive:

MPI_SEND(Message, Count, Datatype, Dest, Tag, Comm, Ierror)

MPI_RECV(Message, Count, Datatype, Source, Tag, Comm, status, Ierror)

Bzw:

Call MPI_SEND(temp, 1, MPI_Real, dest, tag, MPI_COMM_World, Ierror)

Call MPI_RECV(temp, 1, MPI_Real, source, tag, MPI_COMM_World, status, Ierror)

MPI Send/Receive Syntax III

MPI_RECV(Message, Count, Datatype, Source, Tag, Comm, status, Ierror)

Call MPI_RECV(temp, 1, MPI_Real, source, tag, MPI_COMM_World, status, Ierror)

temp	Adresse des Sendepuffers; Real :: temp
1	Count – Anzahl der Elemente im Puffer
MPI_Real	Datentyp des gesendeten Elementes
source	Angabe des Ranges des Sendeprozesses; integer :: source
tag	Nachrichtenkennung (Reihenfolge); integer :: tag
MPI_COMM_World	Kommunikator (Gruppe, Kontext)
status	Empfangsstaus der Nachricht (angekommen?); integer status(MPI_STATUS_SIZE)
Ierror	Fehlerstatus; integer :: Ierror

MPI Nachrichtenaustausch in der Modellierung I

Der Ablauf eines Modelles wir üblicherweise in 3 Phasen eingeteilt:

1) Initialisierung

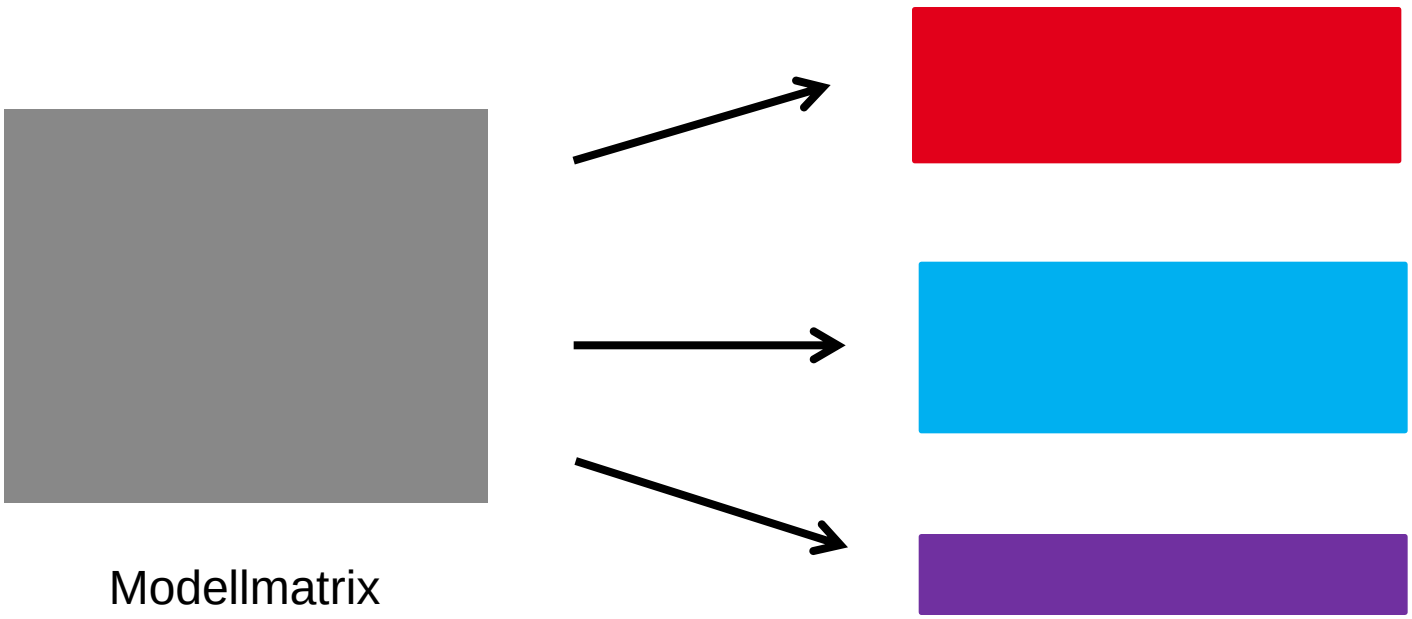
2) Berechnung des Modells

3) Finalisierung

Diese Einteilung wird z.B. von Kopplern wie ESMF gefordert.

MPI Nachrichtenaustausch in der Modellierung II

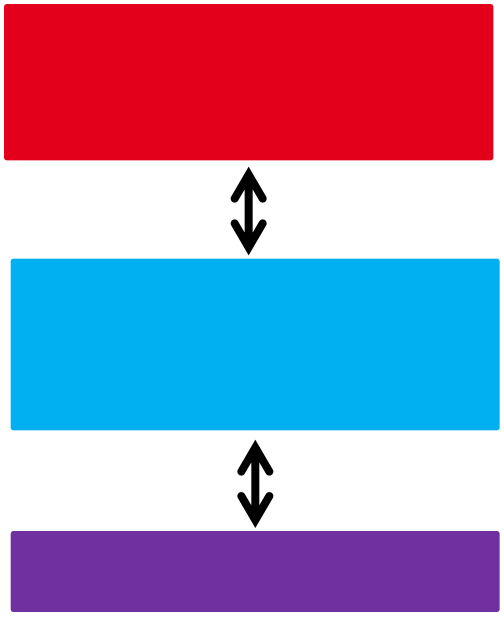
1) Initialisierung: Aufteilung der Rechengebiete und Initialisierung



MPI Nachrichtenaustausch in der Modellierung III

2) Berechnung des Modells: Austausch zwischen den Teilmatrizen

Modellmatrix



Teilmatrizen pro Prozessor

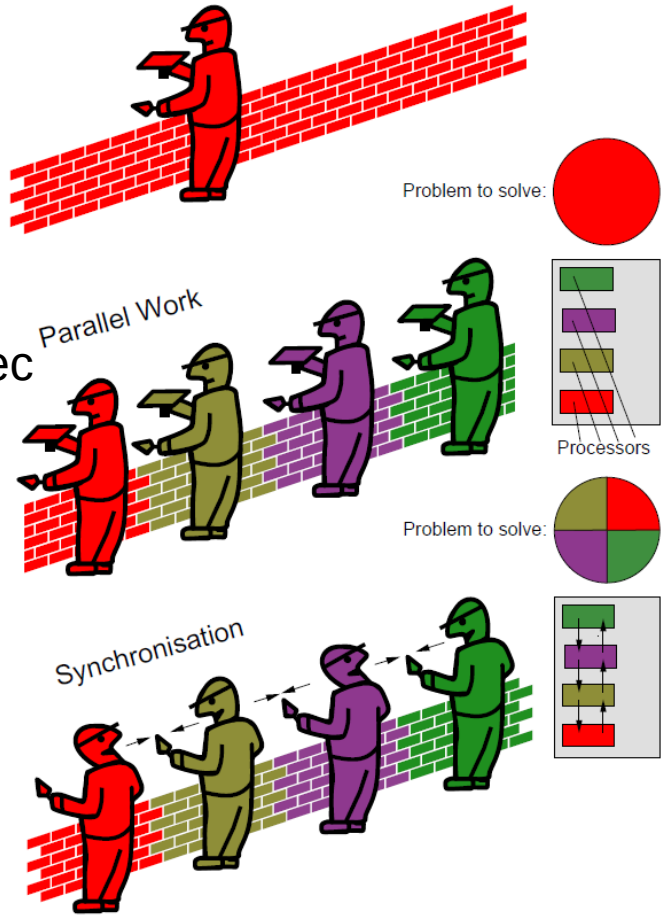
MPI Nachrichtenaustausch in der Modellierung III

Als Info für das Verhältnis
Rechnen / Nachrichtenaustausch
soll folgende Abschätzung dienen:

Ein moderner Parallelrechner schafft
ca. 3 Mrd. floating point operationen / Sec

Der Nachrichtenaustausch
aber nur 10 Mio. Wörter / Sec

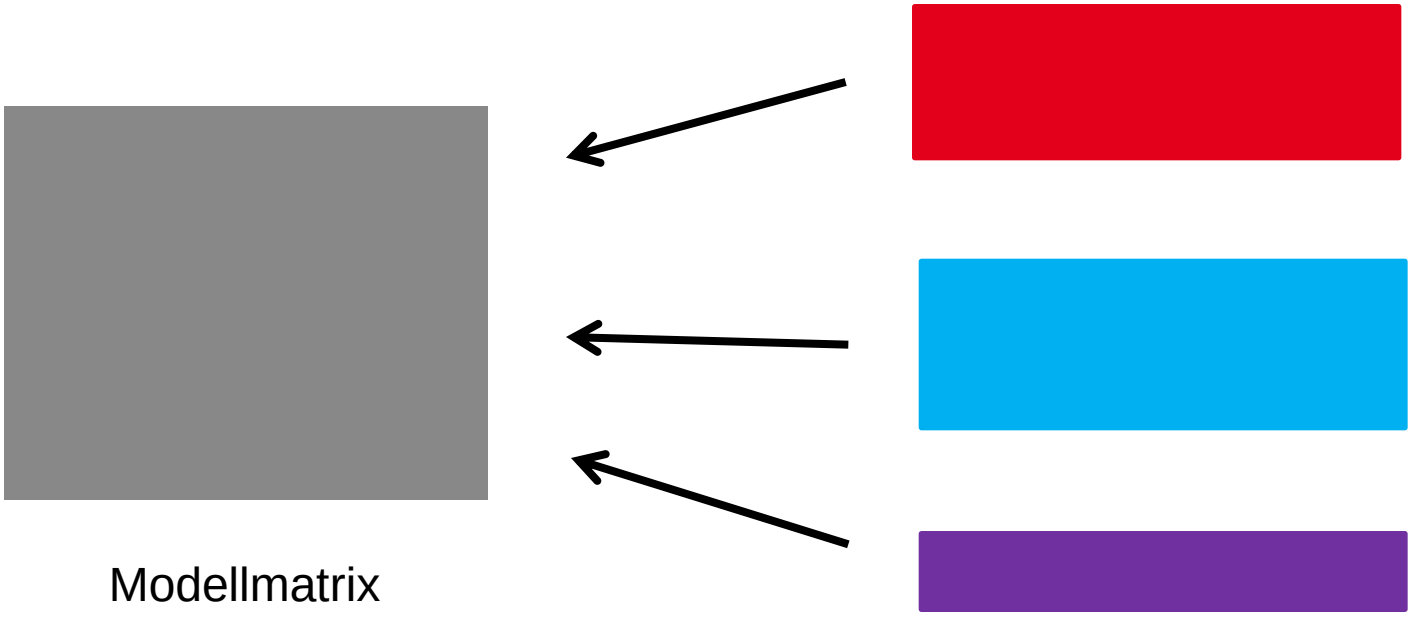
Faktor 300 !!



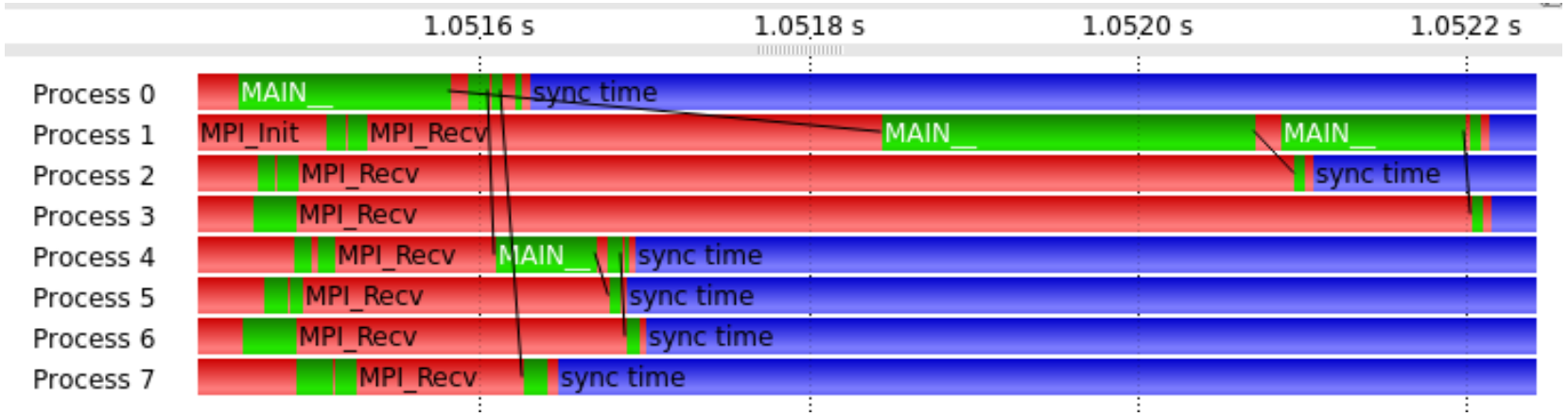
[picture by W. Baumann]

MPI Nachrichtenaustausch in der Modellierung IV

3) Finalisierung: Zusammenfügen der Rechenergebnisse



Visualisierung des Programmablaufes mit Vampire:



Danke,
gibt es noch Fragen?