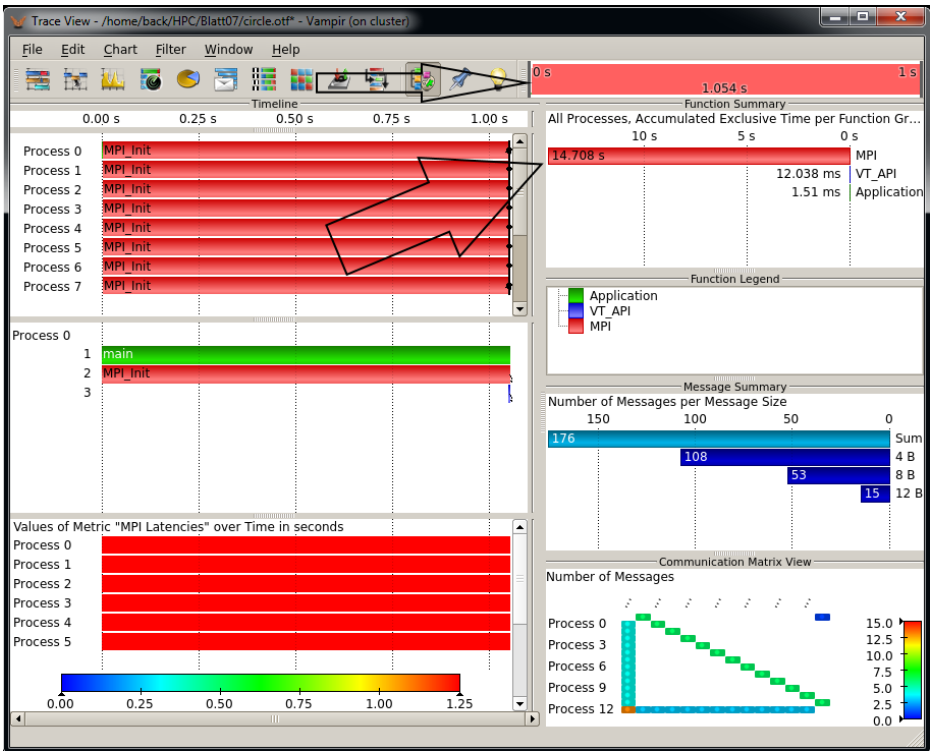
Vampir

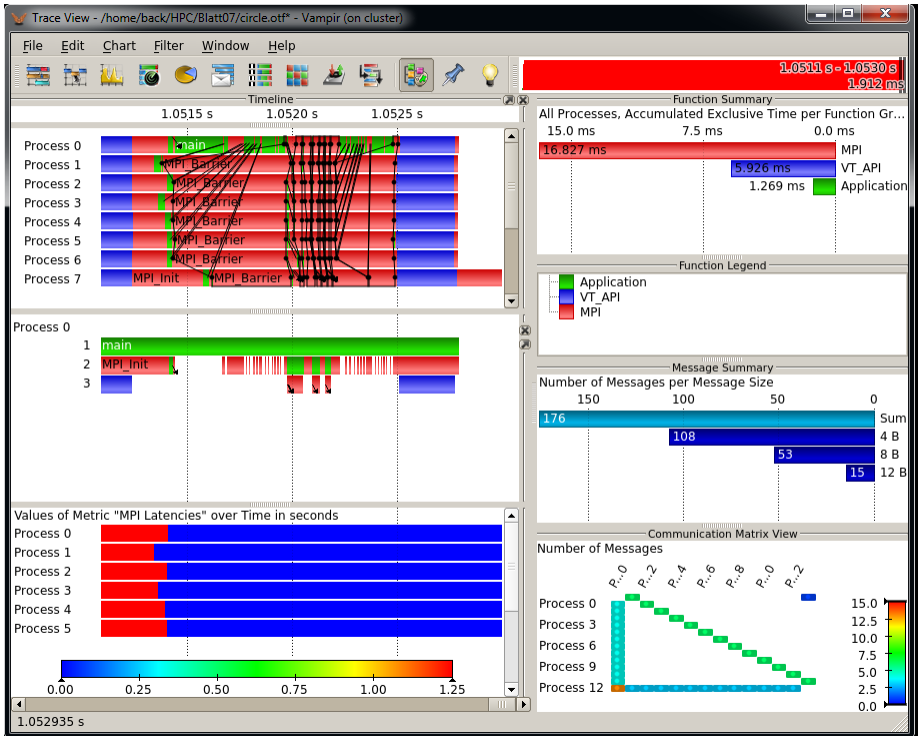
HINWEIS: UNSER PROGRAMM LIEF MIT 13 PROZESSEN.

Vampir öffnet sich nach ein paar Klicks für zusätzliche Diagramme im folgenden Gewand:



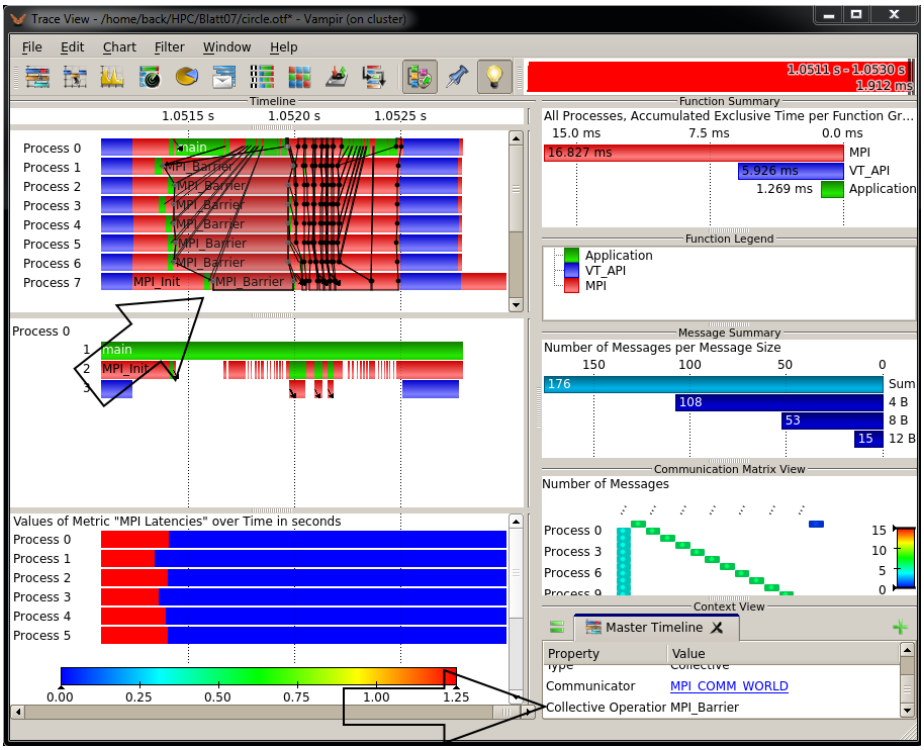
Ganz deutlich ist die Laufzeit des Programms von 14,7 Sekunden abzulesen. In dieser Zeit hat fast nur die ganze MPI Funktionen Zeit verbraucht. Einen großen Teil davon braucht MPI_Init, wie im mittleren linken Fenster am roten Balken zu erkennen.

Für eine genauere Betrachtung kann der Ausschnitt verkleinert werden. Interessant ist insbesondere der hintere Teil. Dies wird durch ziehen des Reglers am kleinem schwarzen Pfeil erreicht.

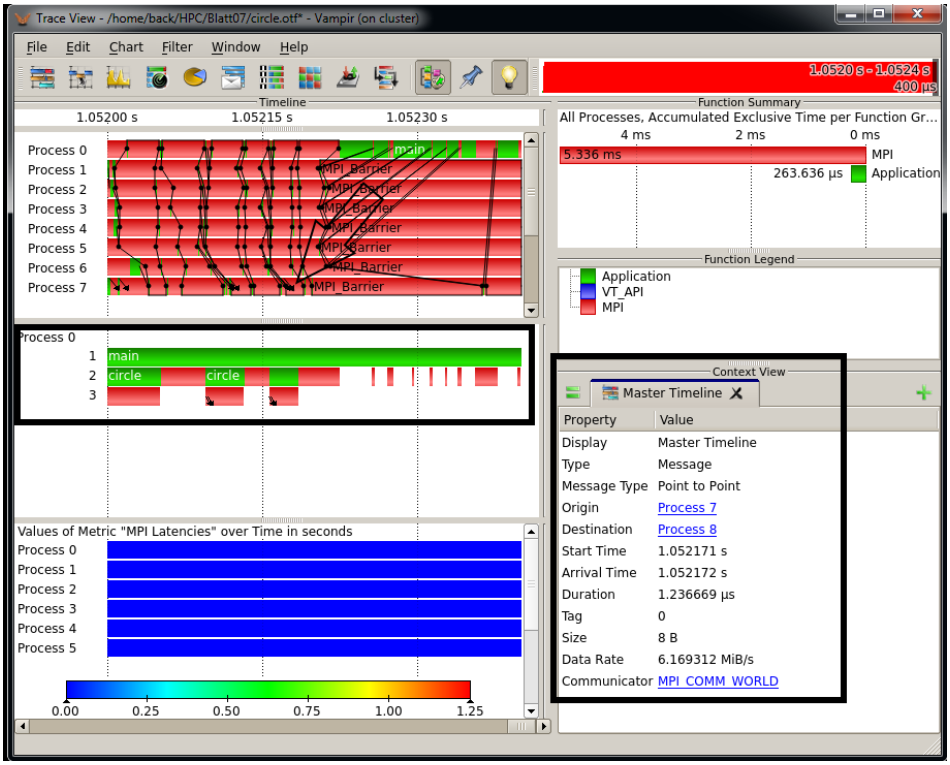


In Fokus steht hier die Gesamtübersicht mit den schwarzen Linien, welche die einzelnen MPI Abstimmungen zeigen. So zeigt sich, dass die einzelnen Prozesse recht viel an den Barriers warten. Dies synchronisiert sich mit Fortlauf des Programms mehr und mehr, stellt dennoch ein Optimierungspotenzial dar.

Über einen Klick auf einen Teilbereich der Prozessübersicht kann auch angezeigt werden, um welche Art der MPI Kommunikation es sich genau handelt. In diesem Fall eine Barrier:



In einer weiteren Vergrößerung der Daten lässt sich dann auch noch genauer der Programmverlauf lesen.



Im mittleren Kasten sind dann klar die Verbindungen zwischen der circle-Funktion und MPI zu erkennen.

Genauer im Kontext View Fenster wird hier eine beispielhafte Nachricht (schwach gekennzeichnet mit einem Pfeil in der Gesamtübersicht) gezeigt. Diese Nachricht hat eine Größe von 8 Byte und wurde von Prozess 7 zu Prozess 8 gesendet. Gleiches lässt sich auch für alle anderen Nachrichten anzeigen.

Diskussion des Kommunikationsschemas:

Im Allgemeinen kommunizieren die Prozesse wie wir es erwartet haben. Alle Prozessen kommunizieren mit ihren jeweils vorherigen und nachfolgenden und nicht kreuz und quer durcheinander. Und am Ende unterhalten sich alle immer mit dem letzten Prozess, was auch in der Verzögerung im Programmablauf sichtbar ist.

Abgesehen davon, dass MPI_INIT so lange braucht, ist es erstaunlich, wie lange die einzelnen Barriers blockieren. Das ist (auf Dauer) ein enormer Ressourcenfresser, der mit weiterer Optimierung reduziert oder vollständig ausgebaut werden muss.