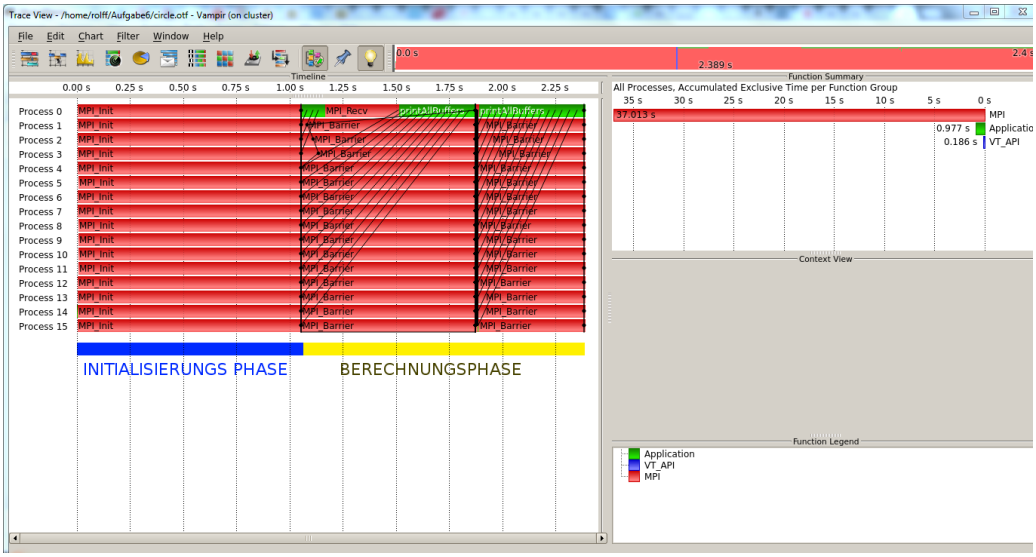
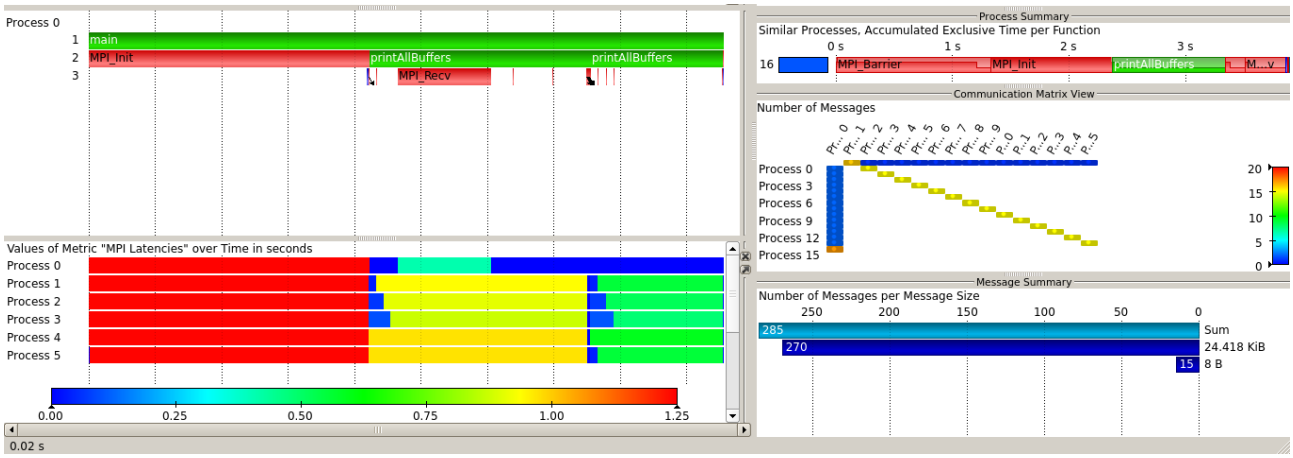
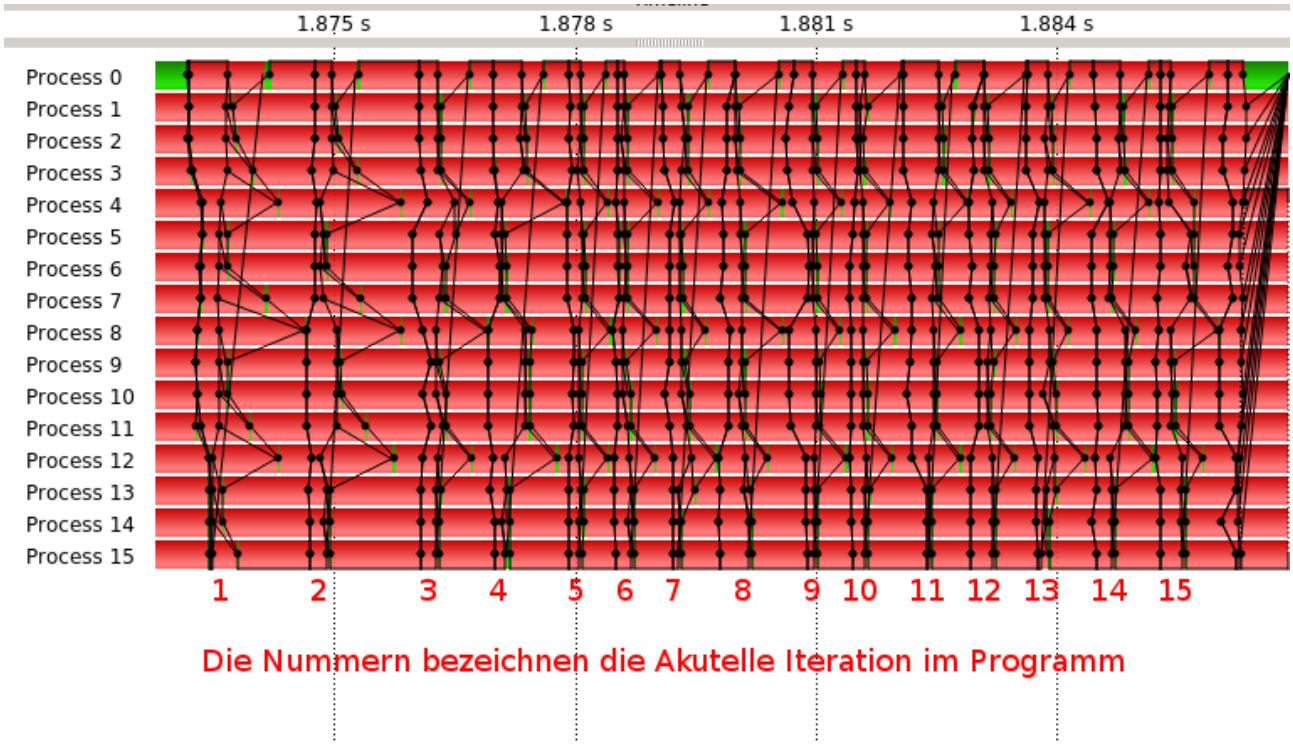


Hinweis:
Das Programm wurde auf 4 Knoten mit 16 Kernen ausgeführt hierbei betrug die Array Größe 10000.

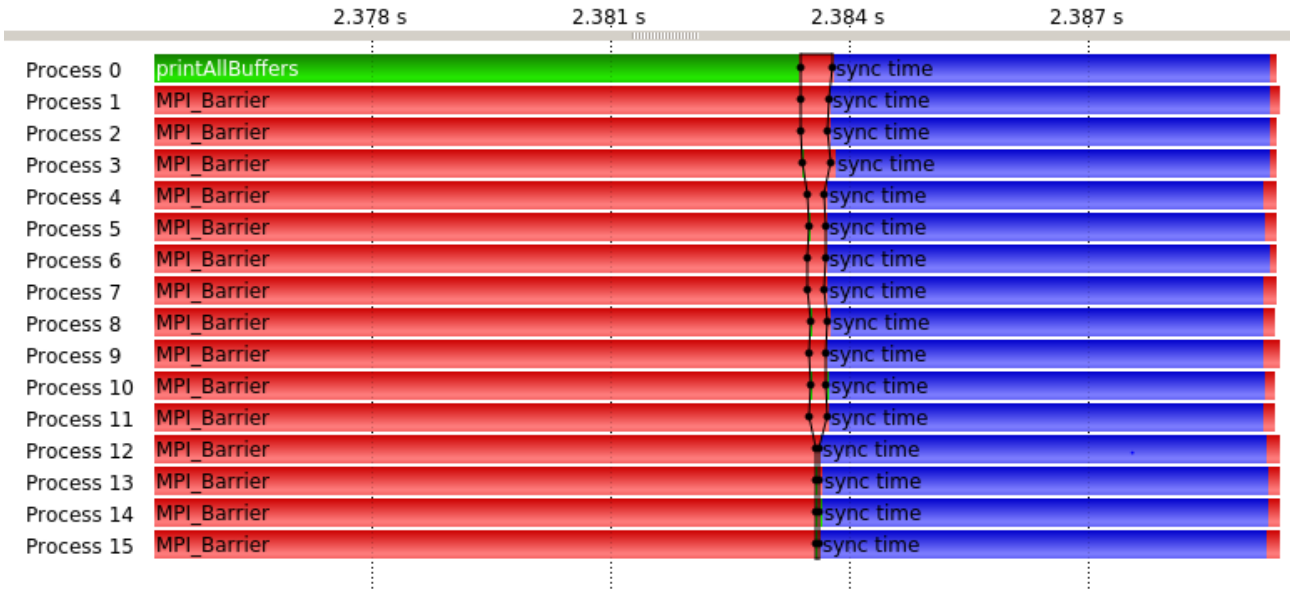
Wie in der Grafik deutlich zu erkennen ist, werden die Daten mittels eines Laufzeit Diagrammes visualisiert, zusätzlich werden aber auch die Namen der Funktionen, welche Aufgerufen wurden farblich hervorgehoben. Die Verbindungslinien zeigen hierbei auf, welcher Knoten mit welchem Kommuniziert hat und wann diese Kommunikation statt fand. Zusätzlich sind die Farbliche Markierungen in der Legende gekennzeichnet wobei Grün: Funktion des Programms, Rot: Funktion vom MPI, Blau: interne Verarbeitung der Vampir-Trace Bibliothek bezeichnet. Alternativ kann noch ein Call-Tree dargestellt werden, bei dem die minimale und maximale Aufrufzeit einer Funktion gelistet ist. Des Weiteren ist es möglich, Aufrufe eines einzelnen Prozesses in einer Timeline genauer anzuzeigen, so wie die Latenzen zwischen den Prozessen. Außerdem besitzt Vampir die Möglichkeit, die Größe der Messages zusammen zufassen, sowie ähnliche Prozesse zu finden und zu Gruppieren. Es gibt zusätzlich noch eine Visualisierung der Kommunikation zwischen den Prozessen, dargestellt als Matrix. Die Farblichen Balken im



ersten Bild sind allerdings nicht von Vampir markiert sondern wurden nachträglich hinzugefügt um einen groben Überblick über die Phasen des Programmes zu haben, eine detaillierte Ansicht ist folgende.



Hierbei sind die Kommunikationen der einzelnen Knoten mehr oder weniger deutlich zu erkennen.
Da die Phase, die das Programm beendet hat, im Vergleich zur Start- und Berechnungsphase auf dem ersten Bild nicht zu erkennen ist, ist sie in folgendem Bild noch einmal hervorgehoben worden. Hier hat der erste Knoten nun seine Ausgabe aller Daten beendet und danach folgt



eine kurze Kommunikationen zwischen den Knoten, sowie ein sofortiges beenden aller Prozesse. Hierbei ist unschwer zu erkennen, dass die Bibliothek von Vampir deutlich mehr Zeit zum Beenden benötigt, als die Prozesse an sich.
Die Kommunikation der Prozesse verlaufen insgesamt wie gewünscht, da zu erst alle Prozesse Initialisieren werden (was leider sehr lange dauert) und danach folgt die ersten Phase. In dieser werden die Größen der Arrays verteilt, in dem kurz an alle Prozesse ein Broadcast gesendet wird, welcher sich durch die Verknüpfung aller Knoten nach dem MPI_Init äußert. Nun werden die Arrays direkt ausgegeben, hierbei ist deutlich zu erkennen, dass die Kommunikation mit

allen anderen Prozessen außer 0 - 3 deutlich länger dauert, da diese auf einem anderen Knoten liegen und somit eine höhere Latenzzeit haben. Dies äußert sich auch durch die Wartezeit in der MPI_Recv Funktion im Prozess 0.
Des Weiteren ist deutlich zu erkennen, dass die Prozesse nach ihrem Rank mit dem Hauptprozess kommunizieren, so dass auch die Ausgabe in aufzeigender Reihenfolge gewährleistet ist. Danach ist nur ein kleiner Streifen zu erkennen auf dem viel kommuniziert wird, dieser ist vergrößert auch in Bild 3 zu sehen. Hierbei sind wie bereits beschrieben die Iterationen mehr oder weniger gut zu erkennen. Es ist weiterhin auch zu sehen, dass die Prozesse in einer baumartigen Struktur kommunizieren und der 4., 8., 12. Prozess immer etwas länger wartet.
Letztendlich folgt nach 15 Iterationen wieder die Ausgabe die sich äquivalent zu der vorherigen verhält. Zum Schluss folgt noch die in Bild 4 vergrößerte Schlussphase, bei der die Prozesse noch einmal Kommunizieren und dann sofort beenden. Somit kommunizieren alle Prozesse erwartungsgemäß.