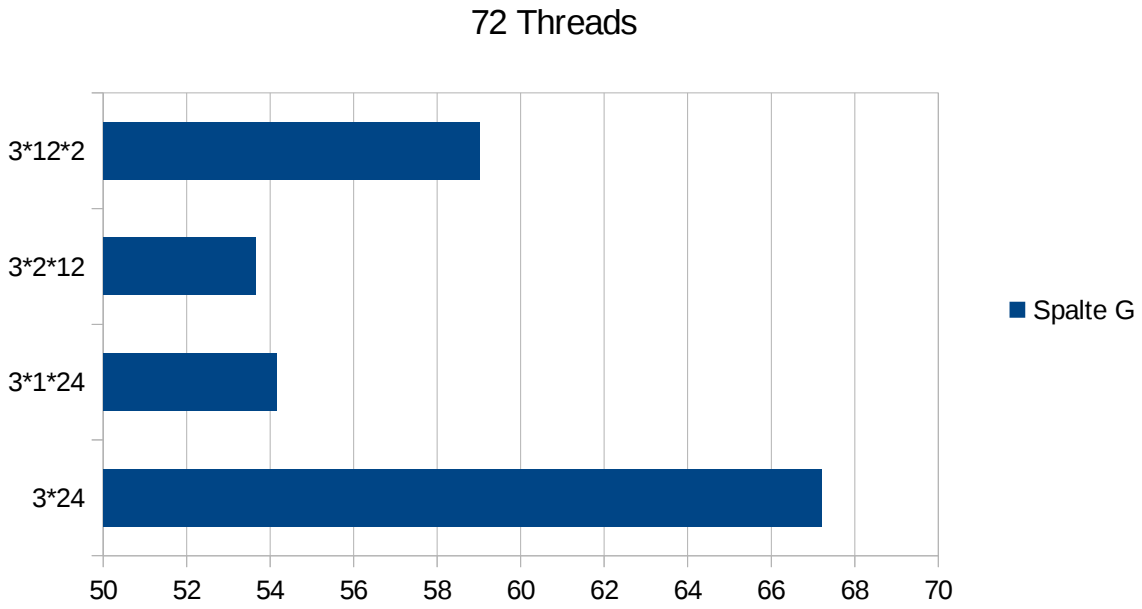
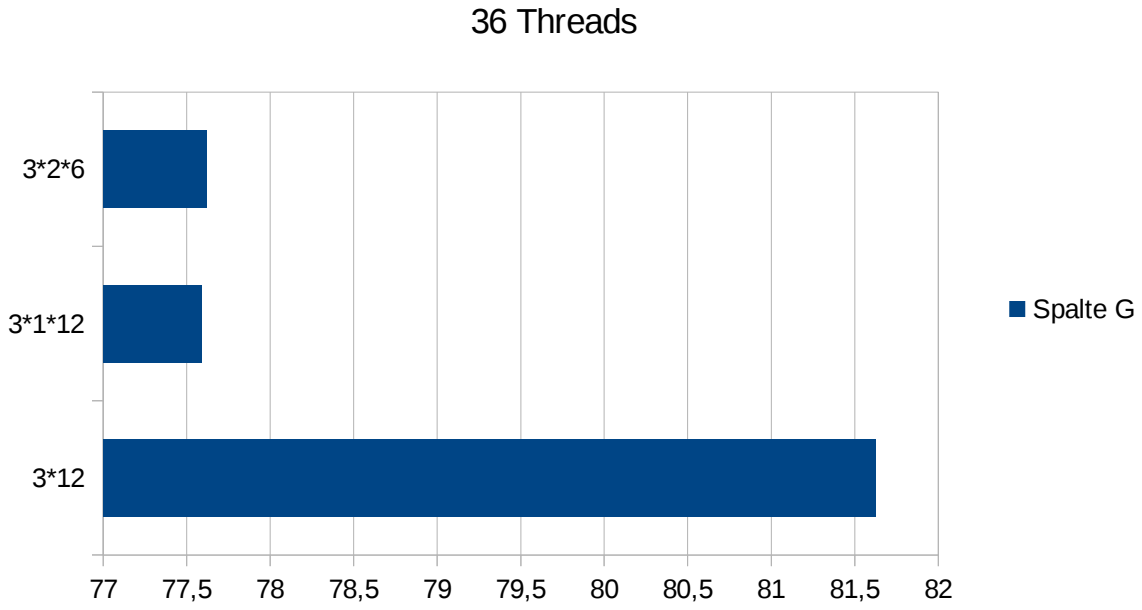
Leistungsanalyse des Hybriden Programms

Konfig	Processes Tot	Threads Total	Messung 1	Messung2	Messung 3	Durchschnitt
3*12	36	36	82.1531470	80.9709920	81.7551220	81,626420333
3*1*12	3	36	77.8132970	77.3553900	77.5948370	77,587841333
3*2*6	6	36	78.3603570	77.1571430	77.3346350	77,617378333
3*24	72	72	68.8776470	67.5891590	65.1188290	67,195211667
3*1*24	3	72	54.2679300	54.5039350	53.7164640	54,162776333
3*2*12	6	72	53.4113020	53.4408780	54.1094010	53,653860333
3*12*2	36	72	59.9193570	58.4391810	58.7287730	59,029103667





Für die Messungen gibt es zwei unterschiedliche Konfigurationen was die Anzahl der Threads betrifft. Einmal 36 oder 72 über 3 Knoten verteilt. Wenn nur Prozesse gestartet werden ist dies mit einer Thread Anzahl von jeweils 1 gleichgesetzt.

Gemessen wurde mit 512 Interlines der aufwendigen Störfunktion und 4096 Iterationen. Die Zeit für die Initialisierung wurde in diesen Messungen nicht berücksichtigt. Die Differenz beträgt etwa eine Sekunde. Es ist jedoch viel Interessanter wie die Rechnung selbst skaliert.
Entscheidend für eine gute Performance des Programms ist, dass so wenig wie nötig so effizient wie möglich kommuniziert wird. Dabei ist insbesondere die jeweilige Struktur der ccNUMA Architektur entscheidend, auf welcher gerechnet wird.
Im Falle von 36 Threads ist die ineffizienteste Variante diejenige, welche ausschließlich auf MPI setzt. Sie führt zu einem unnötigen Speicherverbrauch und langsamerer Kommunikation innerhalb eines Knotens. Die Differenz zwischen den beiden Rechnungen mit OpenMP und MPI ist messbar aber mit 0.03 Sekunden hier klein. (Falls das Programm millionenfach aufgerufen wird wird aber auch so etwas bemerkbar.)
Im Fall der Messung mit 72 Threads ist wie zu erwarten die Implementierung ausschließlich mit MPI am langsamsten. Am schnellsten ist die Implementierung mit zwei Prozessen mit je 12 Threads pro Node. MPI ist zwar langsamerer skaliert aber besser als OpenMP. Es ist als Effizienter auf einem Node zwei Prozesse zu starten, die jeweils 12 Threads besitzen. Hizu kommt, dass dies das Layout ist, welches am nächsten zur Hardware ist. Jeder der zwei Sockets pro Node kann 12 Threads gleichzeitig ausführen. Dabei kann Speicher im L2 und L3 Cache des Prozessors mit Open MP viel effizienter genutzt werden als per MPI, wo jeder Prozess seien eigenen vollständigen Speicher braucht.