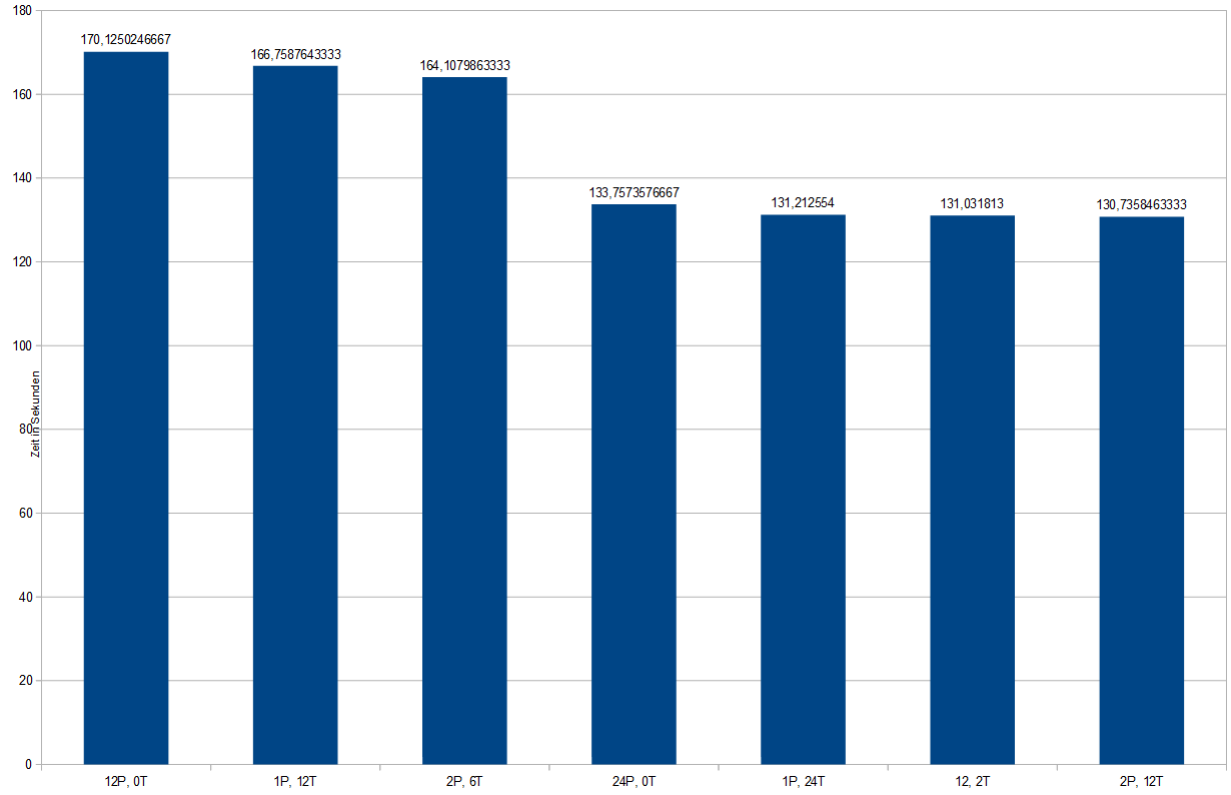
Jede der Messungen fand auf 3 Knoten gleichzeitig statt.
Die angegebenen Prozessanzahl enspricht der Anzahl an Prozessen pro Knoten, die Threadanzahl der Anzahl an Threads pro Prozess.
0 Threads entspricht der Ausführung des MPI-Programmes ohne Open_MP.
Die verwendetet Parameter: 512 Interlines, die Störfunktion $f(x, y) = 2\pi^2 \sin(\pi x)\sin(\pi y)$, Abbruch nach 10240 Iterationen.



Messung	1. Messung	2. Messung	3. Messung	Durschnittliche Zeit
3 Knoten × 12 Prozessen	172,4686s	170,694755s	167,211719s	170,1250247s
3 Knoten × 1 Prozess × 12 Threads	165,741097s	167,63133s	166,903866s	166,7587643s
3 Knoten × 2 Prozesse × 6 Threads	163,003742s	167,192245s	162,127972s	164,1079863s
3 Knoten × 24 Prozessen	135,733609s	131,54905s	133,989414s	133,7573577s
3 Knoten × 1 Prozess × 24 Threads	131,103748s	131,109663s	131,424251s	131,212554s
3 Knoten × 12 Prozesse × 2 Threads	130,233848s	131,102749s	131,758842s	131,031813s
3 Knoten × 2 Prozesse × 12 Threads	130,652075s	130,512751s	131,042713s	130,7358463s

Eine Verdoppelung der Threads, bzw der Prozesse, bringt zwar eine Beschleunigung des Programmes, allerdings nur um ca 30s, was in etwas 18% entspricht. Dies könnte mit dem Overhead zu erklären sein. Die Verwaltung, also Speicher- und Ressourcenzuweisung sowie der Kommunikationsaufwand der weiteren Prozesse und Threads scheint eine starke Limitierung für die Geschwindigkeit des Programmes darzustellen. Des weiteren fällt auf, dass die nichthybride Variante langsamer läuft. Dies war zu erwarten, da die hybride Variante eine parallele Verarbeitung der dem jeweiligen Prozess zugeteilten Zeilen ermöglicht. Dementsprechen hätten wir allerdings einen größeren Speedup erwartet als wir tatsächlich messen konnten. Die Messungen mit einem Prozess und 24 Threads lieferten ein nahezu identisches Ergebnis zu 12 Prozessen und 2 Threads und 2Prozessen und 12 Threads. Dies kam einigermaßen überraschend, da wir aufgrund der höheren Parallelisierung erwartet hätten, dass eine höhere Threadanzahl einen schnelleren Ablauf bedeutet. Das dies nicht der Fall war lässt darauf schließen, dass der Overhead für MPI der gleiche ist wie für Open_MP.