1 Messung 1 - Abhängigkeit der Programmlaufzeit von Threads

Bei der ersten Messungen haben wir untersucht, wie sich die Rechenzeit mit der Anzahl der Threads verhält. Dazu wurden zwei Messreihen aufgenommen. Eine auf der Node amd3 und eine auf der Node west10. Für die Messung wurden jeweils 5 unabhängige Programmläufe gestartet und der arithmetische Mittelwert gebildet. Zu beachten ist die unterschiedliche Architektur der beiden Nodes. Auf der Partition west haben alle Nodes 2 Sockets mit 6 Kernen und 2 Threads pro Kern. Auf der Partition amd haben alle Nodes 2 Sockets mit 12 Kernen und einem Thread (Auslesen der Informationen über sinfo -p partition -o %z).

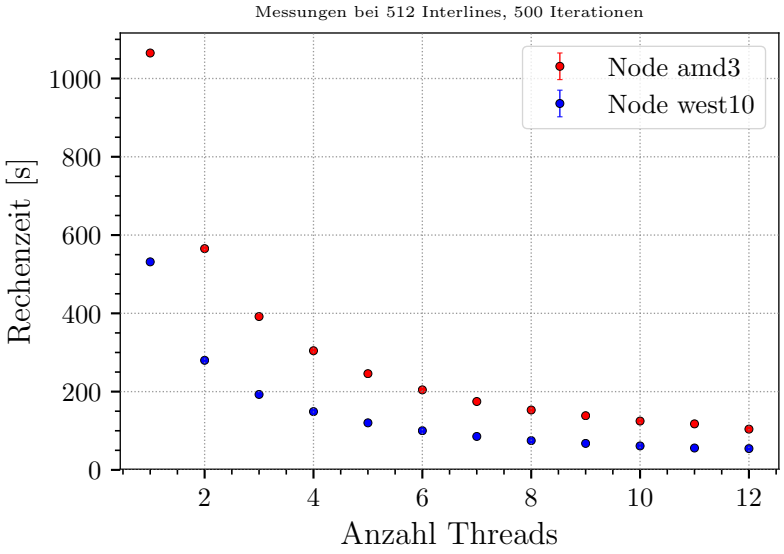

Abbildung 1: Abhängigkeit der Programmlaufzeit von den verwendeten Threads. Arithmetischer Mittelwerte über jeweils 5 unabhängige Messungen.

In Abbildung 1 sind beide Messreihen dargestellt. Für beide Nodes ist eine exponentielle Abnahme der Rechenzeit bei steigender Anzahl von Threads zu erkennen. Die Node amd3 ist immer langsamer als west10, zeigt aber ab einer Threadanzahl von 8 einen größeren Rechenezeitgewinn. Die zunächst rasche Verkürzung der Rechenzeit bis zu einer Threadanzahl von 6, ist auf die bessere Auslastung der CPU auf der entsprechenden Node zurückzuführen. Ab 6 Threads ist die Minderung der Rechenzeit mit jedem weiteren Thread nicht mehr so groß wie vorher. Der zusätzliche Aufwand des Betriebssystems die Threads zu verwalten steigt mit der Anzahl der Threads und wird zu dem Zeitpunkt zu einem wesentlichen Faktor bezüglich der Rechenzeit. Durch die Verwendung von Threads war eine Verkürzung der Rechenzeit vom Faktor $10,2$ auf amd3 und $9,8$

auf west10 möglich. Dies liegt etwas unterhalb des theoretisch erwarteten Werts von 12, da nicht der gesamte Code parallelisiert werden kann. Die Architektur der Nodes lässt drauf schließen, dass eine weitere Erhöhung der Threadzahl zu keiner deutlichen Leistungssteigerung mehr führen wird, da bereits alle Kerne beansprucht werden.

2 Messung 2 - Abhängigkeit der Programmlaufzeit von Interlines

Die zweite Messung untersucht die Abhängigkeit der Programmlaufzeit von der Matrixgröße (Interlines). Die Anzahl der Matrixelemente wächst also mit N^2, wenn N die Anzahl der Interlines ist. Wir erwarten für eine Vergrößerung der Interlines ein exponentielles Wachstum der Laufzeit bei gleichbleibender Threadanzahl. Bei einer niedrigen Anzahl von Interlines sollte die Laufzeit weniger stark ansteigen, da die Vorbereitung der Datenstrukturen den größten Teil der Berechnung einnimmt und nahezu konstant in der Laufzeit ist.

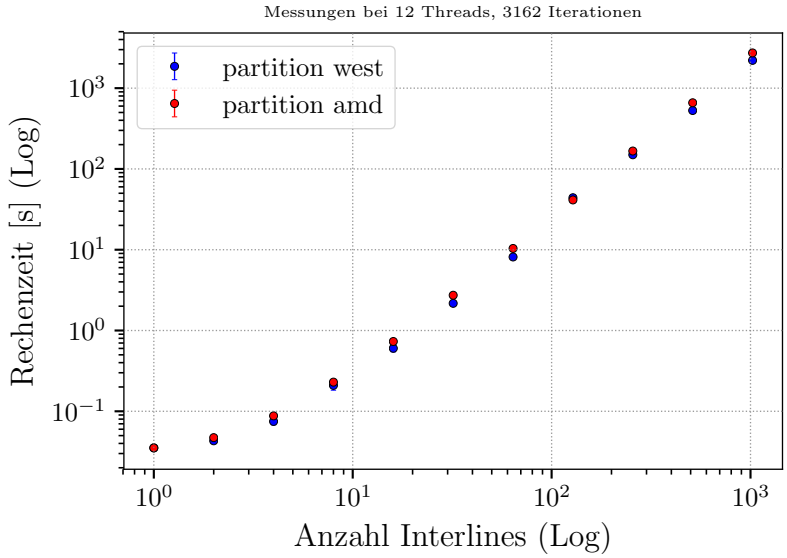

Abbildung 2: Abhängigkeit der Rechenzeit von der Anzahl der Interlines. Arithmetischer Mittelwert über jeweils 5 unabhängige Messungen. Die Nodeauswahl erfolgt durch Slurm auf der entsprechenden Partition.

In Abbildung 2 sind zwei Messungen auf unterschiedlichen Partitionen gezeigt. Da der Cluster nach Angaben von sinfo homogen innerhalb der Partitionen ist, haben wir uns entschieden keine Node zu selektieren, da die Jobs dann schneller auf der nächsten freien Node starten. Das vergrößert unseren Fehler,

der aber, wie in Abbildung 2 zu sehen ist, immer noch sehr klein ist. Unser erwartetes Verhalten bestätigt sich. Bei der Verwendung bis 70 Interlines wächst die Berechnungszeit langsamer als bei größeren Werten. Für Werte ab 100 Interlines ist der Anstieg der Rechenzeit exponentiell. Wie angenommen liegt dies an der Zunahme der benötigten Rechenoperationen. Der zunächst langsamerer Anstieg begründet sich in dem im Verhältnis großen Aufwand des Betriebssystems die Threads zu verwalten.