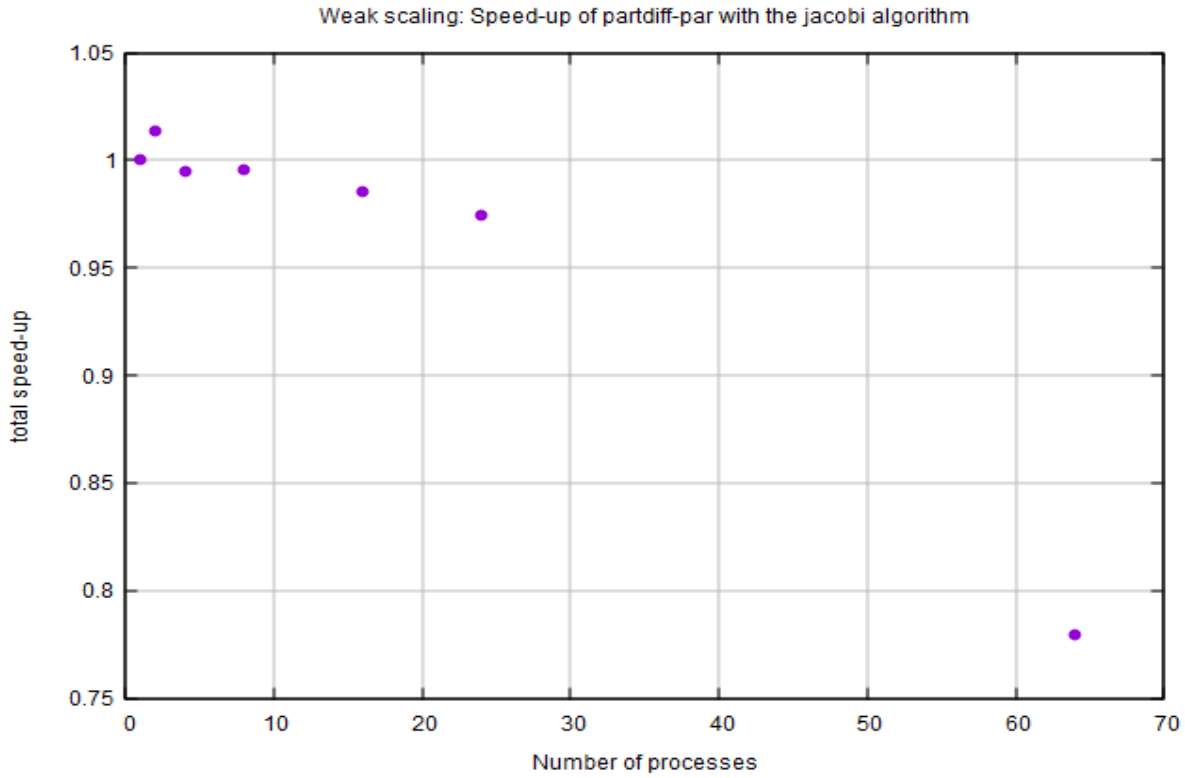
Qualitätsprüfung des parallelisierten PDE-Lösers

Weak Scaling

Speed-up-Graph:



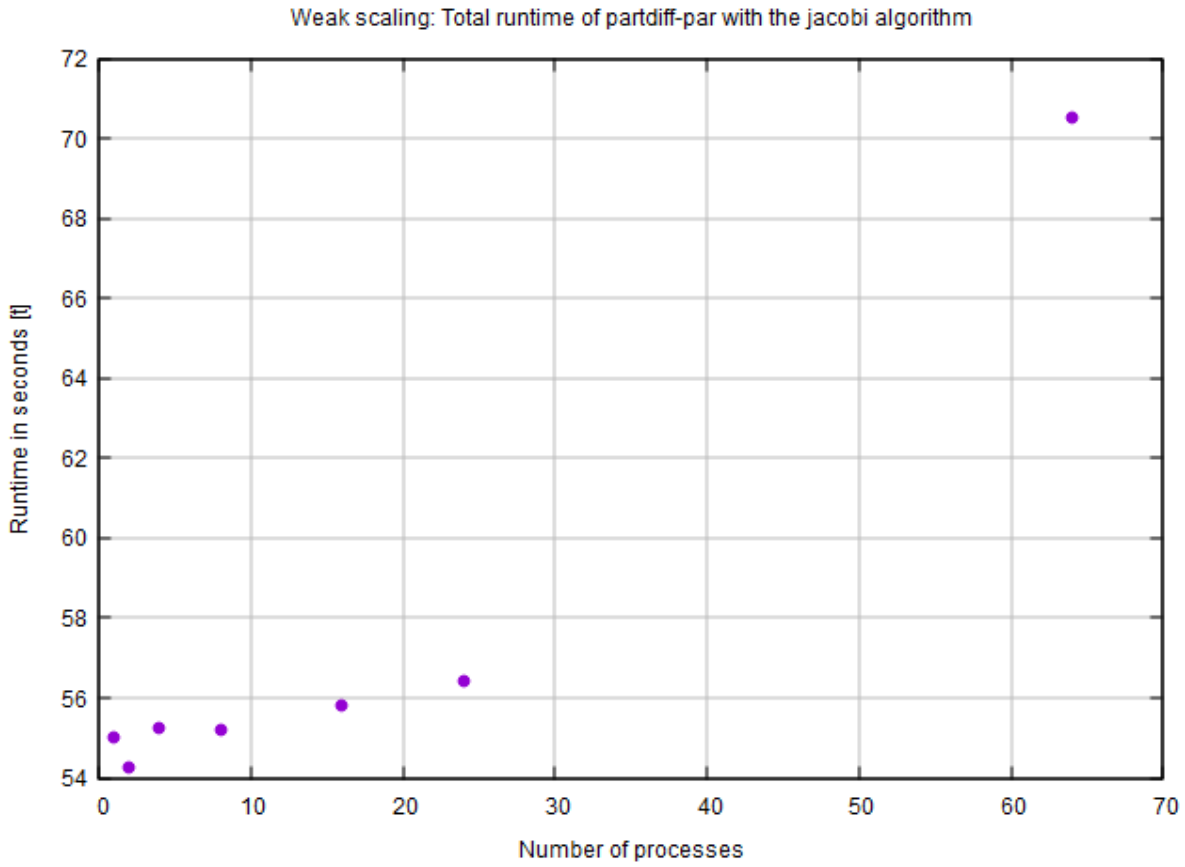
Im Idealfall würde die Funktion als lineare Linie genau auf der 1 laufen. Bei unserem Programm ist dies leider nicht der Fall. Mit zunehmender Prozesszahl entfernt sich das Programm von der Ideallinie. Dies lässt sich durch die längeren Nachrichten bei der Kommunikation erklären, da bei steigender Interlinezahl sich auch die Anzahl der Daten, die übertragen werden müssen, erhöhen. Zudem kann es zu Lastungleichheiten kommen, dies konnten wir bei einem so kleinen Datensatz aber nicht genauer untersuchen.

Wie erklären Sie sich die Wahl der Interlines in Bezug auf die Prozesszahl?

Beim weak scaling eines Algorithmus wird die Effizienz gemessen, wie er bei konstanter Prozesszahl und angepasster Problemgröße skaliert. Da bei partdiff-par die Problemgröße einzig von der Anzahl der Interlines abhängt, muss dieser Parameter für das weak scaling angepasst werden.

Die auf den ersten Blick willkürliche Skalierung der Interlines von 100 auf 141 auf 200 …, ist bewusst gewählt. Um die Problemgröße zu verdoppeln, muss die Anzahl der Einträge in der Matrix, welche berechnet wird, verdoppelt werden. Um die quadratische Form der Matrix beizubehalten ist eine Interlinezahl von 141 ideal.

Weak scaling Graph:



Im Prinzip ist in diesem Graph das gleiche zu sehen, wie bereits im vorherigen.

Alle bis auf den letzten Punkt liegen in etwa auf einer Linie. Nur der letzte fällt deutlich heraus. Dies lässt sich wieder mit zunehmender Kommunikation erklären.

Daten:

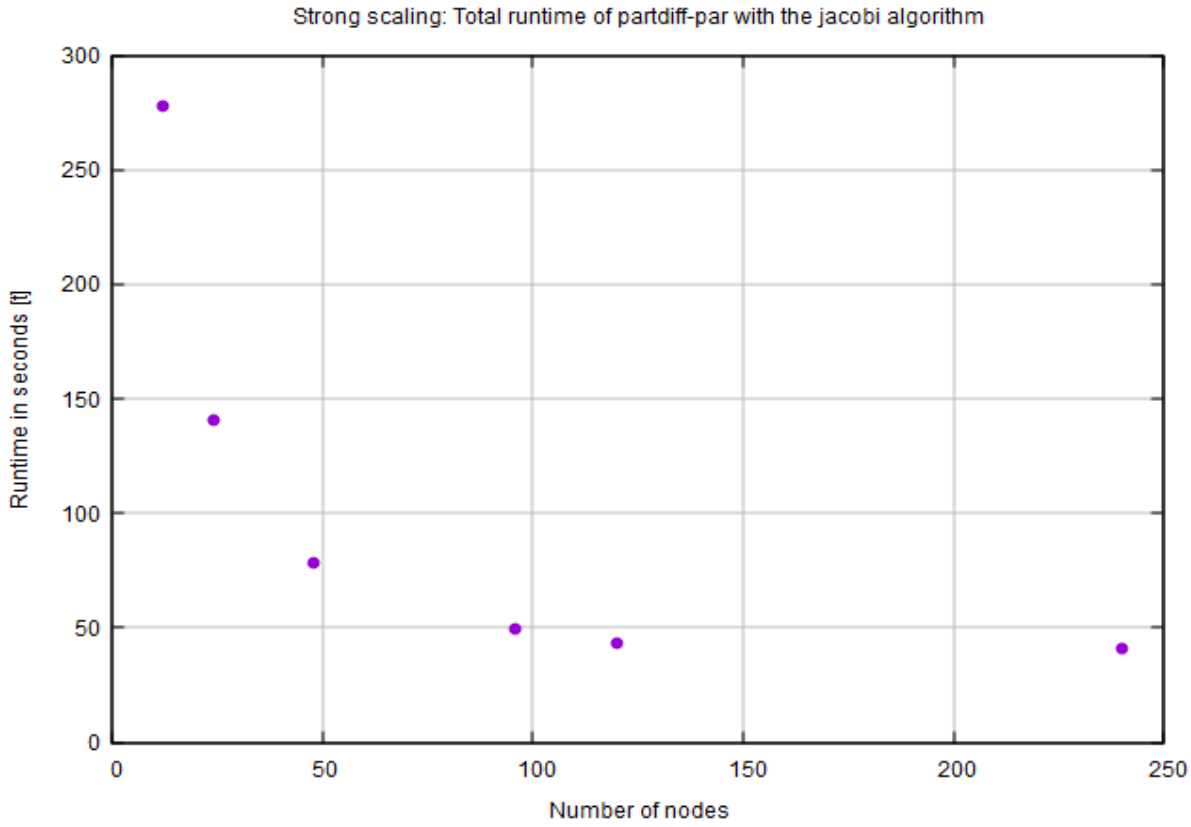
NPROCS	NNODES	ILINES	TIME	SPEEDUP
1	1	100	549.936	1.0000
2	1	141	542.627	1.0135
4	2	200	552.579	0.9952
8	4	282	552.235	0.9958
16	4	400	558.024	0.9855
24	4	490	564.103	0.9749
64	8	800	705.259	0.7798

Hinweis:

Auf Gauss-Seidel-Verfahren haben wir auf Grund des nicht komplett funktionierenden Programms verzichtet.

Strong Scaling

Strong scaling Graph:



Beim strong scaling wird das gleiche Problem auf zunehmend mehr Prozesse aufgeteilt. Sprich, wenn das Problem mit mehr CPU-Leistung gerechnet wird, verringert sich auch die Laufzeit.

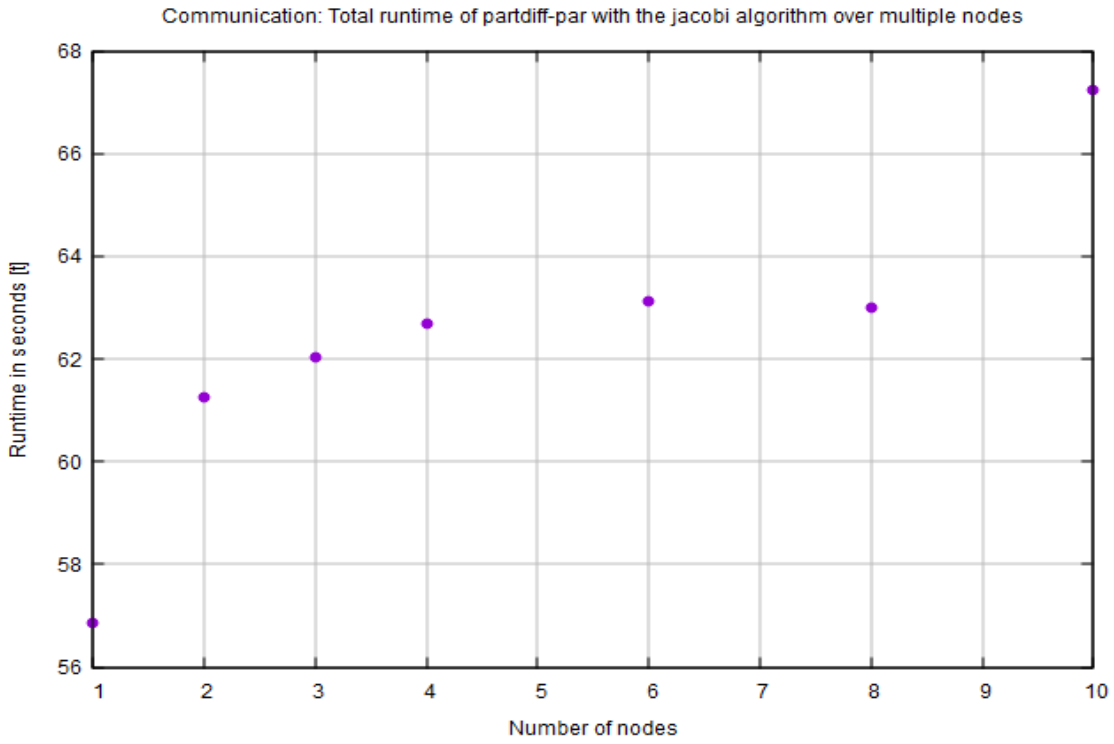
Dies ist bei unserem partdiff-par auch der Fall, jedoch ist ersichtlich, dass sich dieses Prinzip nicht unendlich (mit gutem Kosten-Effizienz-Verhältnis) fortführen lässt. Bereits bei 120 Prozessen lässt sich kaum noch eine Laufzeitverbesserung feststellen. Somit macht es kaum einen Sinn dieses partdiff-par Programm mit mehr als 96 Prozessen laufen zu lassen.

Daten:

NPROCS	NNODES	ILINES	TIME
12	1	960	277.7363
24	2	960	140.8620
48	4	960	78.1195
96	8	960	49.8366
120	10	960	43.3742
240	10	960	40.7220

Kommunikation

Kommunikationsgraph



Das Programm terminiert auch auf mehreren Nodes ungefähr nach der gleichen Zeit. Die Zunahme der Laufzeit bei Benutzung mehrerer Knoten lässt auch hier wieder auf die Kommunikation (siehe weak scaling) zurückführen.

Zwei, mitunter drei, Punkten fallen in diesem Graph besonders auf. Zum einen der erste Koordinatenpunkt. Er ist gegenüber den anderen in der Laufzeit deutlich tiefer, was sich durch die nicht notwendige Computer-zu-Computer-Kommunikation erklären lässt.

Der letzte Punkt fällt deutlich nach oben aus. Wahrscheinlich lastet partdiff-par den Prozessor nicht mit 100 Prozent aus, was zu Leistungsverlusten im Vergleich mit weniger Nodes – dafür allerdings annähernd 100 prozentiger Auslastung - führt.

Der vorletzte Punkt, wo acht Nodes verwendet werden, hätte unserer Meinung nach auch wieder etwas langsamer laufen müssen als mit sechs Nodes. Den Laufzeitunterschied können wir uns nicht erklären.

Daten:

NPROCS	NNODES	ILINES	TIME
10	1	200	56.8580
10	2	200	61.2495
10	3	200	62.0310
10	4	200	62.6898
10	6	200	63.1150
10	8	200	63.0162
10	10	200	67.2503