Universität Hamburg, Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen
Prof. Dr. T. Ludwig, Dr. M. Kuhn
M. Blesel, R. Felgenhauer, J. Plehn

Übungsblatt 4 zur Vorlesung
Hochleistungsrechnen
im WiSe 2019/2020
Abgabe: 09.11.2019, 23:59

Parallelisierung mit OpenMP (120 Punkte)

Wir gehen jetzt wieder von unserem sequentiellen Programm zur Lösung der Poisson-Gleichung aus und betrachten dabei aber nur die Variante des **Jacobi-Verfahrens**. Hierfür sollen jetzt Parallelisierungen mittels OpenMP erstellt werden.

Mit der Option `-fopenmp` erzeugt `gcc` OpenMP-Code. Ein Beispielprogramm ist unter `/home/hr/openmp/hello` verfügbar und lässt sich direkt aufrufen. Es arbeitet standardmäßig mit so vielen Threads, wie logische Cores vorhanden sind (24 auf den Rechenknoten). Den Quellcode dazu finden Sie unter `/home/hr/openmp/hello.c`.

Sie können das Programm mit einer anderen Anzahl Threads laufen lassen, indem Sie die Umgebungsvariable `OMP_NUM_THREADS` entsprechend setzen. Tutorials zur Programmierung mit OpenMP finden Sie unter folgender URL:

`https://computing.llnl.gov/tutorials/openMP/`

Aufgabenstellung

Parallelisieren Sie das Jacobi-Verfahren aus dem **sequentiellen** Programm mittels OpenMP. Die parallele Variante muss dasselbe Ergebnis liefern wie die sequentielle; sowohl der Abbruch nach Iterationszahl als auch der nach Genauigkeit müssen korrekt funktionieren und dieselben Ausgaben wie die sequentiellen Gegenstücke liefern!

Das parallelisierte Programm sollte bei Ausführung mit 12 Threads und 512 Interlines einen Speedup von ungefähr 10 erreichen. Wiederholen Sie dabei jede Messung mindestens drei Mal, um aussagekräftige Mittelwerte bilden zu können. Es ist außerdem empfehlenswert die Störfunktion $f(x,y) = 2\pi^2 \sin(\pi x) \sin(\pi y)$ zu verwenden, da der erhöhte Rechenaufwand das Skalierungsverhalten verbessert.

Wenn Sie Ihr Programm im interaktiven Modus ausführen, können Sie z. B. mit dem Tool `top` („1“ drücken, um die Cores aufgeschlüsselt zu erhalten) die Auslastung betrachten.

Bitte protokollieren Sie mit, wieviel Zeit Sie benötigt haben. Wieviel davon für die Fehlersuche?

Umsetzung der Datenaufteilungen (45 Bonuspunkte)

In Aufgabe 1 haben Sie auf eine bestimmte Weise die Daten auf die Threads verteilt. In dieser Aufgabe sollen Sie die Daten auf drei verschiedene Arten verteilen:

- Zeilenweise Aufteilung (d. h. Thread 1 bekommt die erste Zeile, ...)
- Spaltenweise Aufteilung (d. h. Thread 1 bekommt die erste Spalte, ...)
- Elementweise Aufteilung (d.h. jedes Matrixelement kann von einem anderen Thread berechnet werden)

Implementieren Sie die verschiedenen Datenaufteilungen in Ihrem OpenMP-Programm, wobei für jede Datenaufteilung eine separate `calculate`-Funktion angelegt werden soll. Wenn verschiedene Datenaufteilungen parallelisiert sind, geben Sie den Code bitte so ab, dass die Binärdateien für die entsprechenden Datenaufteilungen jeweils ein Target sind. Hierzu kann beim Kompilieren `-D` verwendet werden, um Flags während der Kompilierung zu setzen.

Hinweis: Für diese Aufgabe muss eventuell der Code der Schleifen umgeschrieben werden. Überlegen Sie sich für den Vergleich geeignete Parameter für den Start ihres Programmes. Welchen Grund kann es für die gemessenen Ergebnisse geben? Schreiben Sie dazu ca. eine halbe Seite Erklärungen.

Vergleich der Scheduling-Algorithmen (45 Bonuspunkte)

OpenMP kennt verschiedene Strategien zur Verteilung von Arbeit (z. B. Schleifeniterationen) an die Threads. Wenn Sie die Aufgabe verschiedener Datenaufteilungen gelöst haben, dann können Sie auch noch verschiedene OpenMP-Scheduling-Algorithmen evaluieren. Eine Liste mit sinnvollen Scheduling-Einstellungen lautet wie folgt:

- Static (Blockgrößen 1 ,2, 4 und 16)
- Dynamic (Blockgrößen 1 und 4)
- Guided

Hinweis: Diese Aufgabe sollte sinnvollerweise auch eine automatische Datenaufteilung verwenden. Dafür ist unter Umständen wieder eine Anpassung der Datenaufteilung und Schleifendurchläufe erforderlich. Vergleichen Sie die Leistungsfähigkeit der Scheduling-Algorithmen für die Datenaufteilung nach Elementen und einer anderen Aufteilung.

Leistungsanalyse (120 Punkte)

Messung 1

Ermitteln Sie die Leistungsdaten Ihres OpenMP-Programms und vergleichen Sie die Laufzeiten für jeweils 1–12 Threads in einem Diagramm. Vergleichen Sie Ihre Variante auch unbedingt mit dem sequentiellen ursprünglichen Programm! Verwenden Sie hierzu 512 Interlines. Der kürzeste Lauf sollte mindestens 50 Sekunden rechnen; wählen Sie geeignete Parameter aus!

Messung 2

Ermitteln Sie weiterhin, wie Ihr OpenMP-Programm in Abhängigkeit von der Matrixgröße (Interlines) skaliert. Verwenden Sie hierzu 12 Threads und 11 Messungen zwischen 1 und 1024 Interlines (wobei Interlines = 2^i für $0 \leq i \leq 10$). Der längste Lauf sollte maximal eine Stunde rechnen; starten Sie mit 1024 Interlines und wählen Sie geeignete Parameter aus, für die folgenden Läufe können Sie die Interlines-Zahl dann entsprechend der gegebenen Formel verringern.

Visualisieren Sie alle Ergebnisse in hinreichend beschrifteten Diagrammen. Schreiben Sie ca. eine halbe Seite Interpretation zu diesen Ergebnissen. Wiederholen Sie jede Messung mindestens drei Mal, um aussagekräftige Mittelwerte bilden zu können. Die

Messungen sollen dabei mit Hilfe von SLURM auf den Rechenknoten durchgeführt werden; geben Sie für jede Messung den verwendeten Rechenknoten an.

Hinweis: Es ist empfehlenswert die Störfunktion $f(x,y) = 2\pi^2 \sin(\pi x) \sin(\pi y)$ zu verwenden, da der erhöhte Rechenaufwand das Skalierungsverhalten verbessert.

Abgabe

Abzugeben ist ein gemäß den bekannten Richtlinien erstelltes und benanntes Archiv. Das enthaltene und gewohnt benannte Verzeichnis soll folgenden Inhalt haben:

- Alle Quellen, aus denen Ihr Programm besteht (`askparams.c` und `partdiff.{c,h}`); gut dokumentiert (Kommentare im Code!)
- Ein Makefile welches mittels `make partdiff-openmp` automatisch eine Binärdatei `partdiff-openmp` erzeugt
- **Optional:** Targets `partdiff-openmp-{element,spalten,zeilen}` für Binärdateien `partdiff-openmp-{element,spalten,zeilen}`, welche jeweils die entsprechende Datenaufteilung umsetzen
- Eine Ausarbeitung `leistungsanalyse.pdf` mit den ermittelten Laufzeiten und der Leistungsanalyse

Senden Sie Ihre Abgabe an `hr-abgabe@wr.informatik.uni-hamburg.de`.