Universität Hamburg
Department Informatik
Wissenschaftliches Rechnen
Prof. T. Ludwig, M. Kuhn, J. Kunkel

Übungsblatt 8 zur Vorlesung
Hochleistungsrechnen
im SoSe 2010
Abgabe: 05.07.2010

Parallelisierung mit OpenMP (100 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, ähnlich der Aufgabe mit den Threads zuvor.

Mit der Option `-fopenmp` erzeugt `gcc` OpenMP-Code. Das Programm `/opt/openmp/openmp-test` lässt sich direkt aufrufen und arbeitet mit zwei Threads, weil es zwei Prozessoren sieht. Man kann es mit einer anderen Anzahl laufen lassen, wenn man die Umgebungsvariable `OMP_NUM_THREADS` entsprechend setzt. Tutorials zur Programmierung mit OpenMP finden Sie unter `http://www.llnl.gov/computing/tutorials/openMP/`.

Aufgabenstellung

Parallelisieren Sie das Jacobi-Verfahren aus dem **sequentiellen** Programm mittels OpenMP. Wiederum müssen die parallelen Varianten dasselbe Ergebnis liefern wie die sequentielle. Bezüglich der Leistung muss sich ungefähr eine doppelte Berechnungsgeschwindigkeit ergeben (2 Threads auf zwei realen Prozessoren).

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

Bearbeitungszeit			
Schwierigkeit	o zu leicht	o genau richtig	o zu schwer
Lehrreich	o wenig	o etwas	o sehr
Verständlichkeit	o großteils unklar	o teilweise unklar	o verständlich
Kommentar			

Vergleich verschiedener Datenaufteilungen (60 Punkte)

In Aufgabe 1 haben Sie auf eine bestimmte Weise die Daten auf die beiden Threads parallelisiert. In dieser Aufgabe wollen wir versuchen die Daten auf drei verschiedenen Arten zu verteilen und die Ergebnisse vergleichen:

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

Hinweis: Für diese Aufgabe muss evtl. der Code der Schleifen umgeschrieben werden. Überlegen Sie für den Vergleich geeignete Parameter für den Start von ihrem Programm. Welchen Grund kann es für gemessene Ergebnissen geben (ca. 1/2 Seite).

Vergleich der OpenMP-Scheduling-Algorithmen (150 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 wenn sie wollen auch noch verschiedene OpenMP-Scheduling-Algorithmen evaluieren. Eine Liste mit sinnvollen Scheduling-Einstellungen:

- Static (Blockgröße 1)
- Static (Blockgröße 2)

- Static (Blockgröße 4)
- Static (Blockgröße 16)
- Dynamic (Blockgröße 1)
- Dynamic (Blockgröße 4)
- Guided

Vergleichen Sie die Leistungsfähigkeit der Scheduling-Algorithmen für die Datenaufteilung nach Elementen und einer anderen Aufteilung.

Bearbeitungszeit			
Schwierigkeit	o zu leicht	o genau richtig	o zu schwer
Lehrreich	o wenig	o etwas	o sehr
Verständlichkeit	o großteils unklar	o teilweise unklar	o verständlich
Kommentar			

Vergleich der Parallelisierungstechniken (150 Punkte)

Ermitteln Sie die Leistungsdaten Ihres OpenMP-Programms und vergleichen Sie die Laufzeiten ihrer verschiedenen parallelisierten Lösungen mit den selben Eingabeparametern (für jeweils 1–4 Prozesse) in einem Diagramm:

- Optimale Skalierung des Sequentiellen Programms eintragen (hierfür Daten für 1 Prozess verwenden)
- MPI-Programm
- POSIX-Programm
- OpenMP-Programm

Sollten Sie noch Werte benötigen so ermitteln Sie diese erneut. Schreiben Sie ein paar Zeilen (1/2 Seite) Interpretation zu diesen Ergebnissen. Die schnellste Parallelisierung sollte mindestens 100 Sekunden rechnen, wählen Sie geeignete Parameter aus.

Bearbeitungszeit			
Schwierigkeit	o zu leicht	o genau richtig	o zu schwer
Lehrreich	o wenig	o etwas	o sehr
Verständlichkeit	o großteils unklar	o teilweise unklar	o verständlich
Kommentar			

Abgabe

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

- Alle Quellen, aus denen Ihr Programm besteht; gut dokumentiert (Kommentare im Code!)
- Ein Makefile welches mittels den Targets `make partdiff-openmp-(item|zeilen|spalten)` automatisch eine Binärdatei `partdiff-openmp-(item|zeilen|spalten)` erzeugt, welche jeweils die entsprechende Datenaufteilung durchsetzen
- Eine Ausarbeitung (PDF) mit den ermittelten Laufzeiten und den Leistungsvergleichen für die beiden Vergleichsaufgaben

Senden Sie Ihre Abgabe per E-Mail an `michael.kuhn@informatik.uni-hamburg.de`.