Universität Hamburg
Fachbereich Informatik
Wissenschaftliches Rechnen
Prof. T. Ludwig, M. Kuhn, J. Kunkel, C. Ansorge

Übungsblatt 8 zur Vorlesung
Hochleistungsrechnen
im WiSe 2010/2011
Abgabe: 17.01.2011

---

# Parallelisierung mit OpenMP (240 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 `/home/hr/openmp/hello` lässt sich direkt aufrufen und arbeitet standardmäßig mit so vielen Threads, wie logische Prozessoren vorhanden sind (also 24). Den Quellcode des Programmes finden Sie unter `/home/hr/openmp/hello.c`.
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. Das parallelisierte Programm sollte bei Ausführung auf 10 Prozessoren ab 256 Interlines eine Effizienz $> 0{,}9$ haben (d. h. Speedup $> 9$).

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

# Vergleich verschiedener Datenaufteilungen (60 Punkte)

In Aufgabe 1 haben Sie auf eine bestimmte Weise die Daten auf die Threads verteilt. In dieser Aufgabe führen wir ein Gedankenexperiment durch, und versuchen die Daten auf drei verschiedenen Arten zu 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 auf einem anderen Thread berechnet werden)

Welche Datenaufteilung wäre bei der vorhandenen Architektur wohl am sinnvollsten (d. h. am schnellsten)? Warum ist das so? (ca. 1/2 Seite)
Können Sie sich Programmiermodelle vorstellen, bei denen eine andere Datenaufteilung besser wäre? Falls ja: bei welcher Hardware wäre dies der Fall? (ca. 1/4 Seite)

## Umsetzung der Datenaufteilungen (150 Bonuspunkte)

Gerne können Sie die verschiedenen Datenaufteilungen auch in Ihrem OpenMP-Programm ausprobieren. 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 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 ,2, 4, 16)
- Dynamic (Blockgröße 1, 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.

## 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 1–12 Prozesse) in einem Diagramm:

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

Ermitteln Sie weiterhin, wie Ihr OpenMP-Programm in Abhängigkeit von der Matrixgröße (Interlines) skaliert. Verwenden Sie hierzu 10 Prozessoren und mindestens $10 \times 3$ Messungen zwischen 0 und 1024 Interlines.
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 50 Sekunden rechnen. Wählen Sie geeignete Parameter aus!

## 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 Vergleichsaufgaben

Senden Sie Ihre Abgabe per E-Mail an `cedrick@gmx.net`.

| 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 | | | |