

## Parallelisierung mit POSIX-Threads (300 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 POSIX-Threads erstellt werden.

Tutorials zur Programmierung mit POSIX-Threads finden Sie unter <http://www.llnl.gov/computing/tutorials/pthreads/>.

### Aufgabenstellung

Parallelisieren Sie das Jacobi-Verfahren aus dem **sequentiellen** Programm mittels POSIX-Threads. Wiederum müssen die parallelen Varianten dasselbe Ergebnis liefern wie die sequentielle Variante; sowohl der Abbruch nach Iterationszahl als auch der Abbruch nach Genauigkeit müssen korrekt funktionieren und dieselben Ausgaben wie die sequentiellen Gegenstücke liefern! Bezüglich der Leistung sollte sich ein akzeptabler Speedup ( $\geq 8$  bei 12 Threads) ergeben. Beachten Sie dazu auch die Hinweise zur Leistungsmessung in der Aufgabe „Leistungsanalyse“. Bitte protokollieren Sie mit, wieviel Zeit Sie benötigen haben. Wieviel davon für die Fehlersuche?

### Leistungsanalyse (60 Punkte)

Ermitteln Sie die Leistungsdaten Ihres POSIX-Thread-Programms und vergleichen Sie die Laufzeiten für jeweils 1–12 Threads in einem Diagramm. Verwenden Sie hierzu 512 Interlines. Der kürzeste Lauf sollte mindestens 50 Sekunden rechnen; wählen Sie geeignete Parameter aus!

Schreiben Sie ein paar Zeilen (1/4 Seite) Interpretation zu diesen Ergebnissen. Wiederholen Sie jede Messung mindestens 3 mal, um aussagekräftige Mittelwerte bilden zu können.

**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 (`.tar.gz`). Das enthaltene und gewohnt benannte Verzeichnis soll folgenden Inhalt haben:

- Alle Quellen, aus denen Ihr Programm besteht, in einem Verzeichnis `pde`; gut dokumentiert (Kommentare im Code!)
  - Ein Makefile welches mittels `make partdiff-posix` automatisch eine Binärdatei `partdiff-posix` erzeugt
- Eine Ausarbeitung `leistungsanalyse.pdf` mit den ermittelten Laufzeiten und der Leistungsanalyse

Senden Sie Ihre Abgabe per E-Mail an [hr-abgabe@wr.informatik.uni-hamburg.de](mailto:hr-abgabe@wr.informatik.uni-hamburg.de).

Bearbeitungszeit			
Schwierigkeit	<input type="radio"/> zu leicht	<input type="radio"/> genau richtig	<input type="radio"/> zu schwer
Lehrreich	<input type="radio"/> wenig	<input type="radio"/> etwas	<input type="radio"/> sehr
Verständlichkeit	<input type="radio"/> großteils unklar	<input type="radio"/> teilweise unklar	<input type="radio"/> verständlich
Kommentar			