Universität Hamburg, Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen
Prof. Dr. T. Ludwig, Anna Fuchs, Jannek Squar
Jakob Sachs, Kjell Ehlers, Lieven Petersen

Übungsblatt 5 zur Vorlesung
Hochleistungsrechnen
im WiSe 2022/2023
Abgabe: 14.-19.11.2022, 23:59

1. Datenaufteilung (60 Punkte) Abgabe 14.11.22, 18:00 Uhr

In vielen Fällen muss die Datenaufteilung explizit und manuell vorgenommen werden. Überlegen Sie sich eine geeignete effiziente Aufteilung der Matrix auf die Threads.

1. Entwickeln Sie eine generische Formel für i(Interlines) $\in \mathbb{N}_0$ und t(Threads) $\in \mathbb{N}$. Beachten Sie, wie die Matrixgröße mit i aufgeschlüsselt wird - achten Sie vor allem auf die Randzeilen-/spalten.

2. Schreiben Sie im Pseudocode auf, wie Sie gemäß dieser Formel die Matrix auf die Threads aufteilen und berechnen würden. Beschränken Sie sich auf die zwei `for`-Schleifen, wobei die Indizes `i` und `j` explizit in den Schleifen und beim Zugriff auf die Matrix angegeben werden müssen.

3. Visualisieren Sie Ihre Aufteilung (grafisch). Zeichnen Sie jeweils eine Matrix für i=0, t=4 und i=1, t=4. Machen Sie dabei auch die Randzeilen, die nicht Teil der Berechnung sind, kenntlich.

Geben Sie die Antworten in `antworten.pdf` ab.

2. Parallelisierung mit POSIX-Threads (240 Punkte) Abgabe 19.11.22, 23:59 Uhr

Parallelisieren Sie das **Jacobi-Verfahren** des **sequentiellen** Programms zur Lösung der Poisson-Gleichung mittels POSIX-Threads.

Tutorials zur Programmierung mit POSIX-Threads finden Sie unter:

`https://hpc-tutorials.llnl.gov/posix/`

Folgende Bedingungen müssen Sie hierbei erfüllen:

- Korrektheit
  - Die parallele Variante muss dasselbe Ergebnis liefern wie die sequentielle: Gleiche Matrix und gleiche Norm des Fehlers. Sowohl der Abbruch nach Iterationszahl als auch der nach Genauigkeit müssen korrekt funktionieren und dieselben Ausgaben wie die sequentiellen Gegenstücke liefern!
- Code
  - Die Threads **müssen** außerhalb der `while`-Schleife erzeugt werden. Achten Sie dabei auf die nötige Synchronisation und den Geltungsbereich der Variablen.
  - Die Anzahl der Threads **muss** über den ersten Parameter gesetzt werden können.
  - Sie dürfen **keine** globalen Variablen einführen.

  - Testen Sie Ihr Programm mit mehreren gleichen Läufen hintereinander, um so ggf. zeitbedingtes fehlerhaftes Verhalten aufzuspüren (race conditions).
  - Gauß-Seidel muss weiterhin sequentiell funktionieren.
  - Ihr Programm soll bei Verwendung von `-Wall` und `-Wpedantic` ohne Fehler oder Warnungen kompilieren.

- Laufzeit
  - Das parallelisierte Programm sollte bei Ausführung mit 12 Threads, 512 Interlines und der komplexen Störfunktion einen Speedup von ungefähr 8 erreichen. 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.
  - Gilt für alle Messungen in HLR, wenn nicht anders verabredet:
    * Wiederholen Sie dabei jede Messung mindestens **drei** Mal, um aussagekräftige Mittelwerte bilden zu können
    * Geben Sie die Messwerte in einer Tabelle ab, inkl. der Messungen des sequentiellen und des parallelisierten Programms mit einem Thread
    * Geben Sie für jede Messung die verwendeten Rechenknoten an

3. Leistungsanalyse (60 Punkte)

Ermitteln Sie die Leistungsdaten Ihres POSIX-Thread-Programms und visualisieren Sie die Laufzeiten für jeweils 1–12 Threads in einem beschrifteten Diagramm. Vergleichen Sie außerdem ihr Programm mit der ursprünglichen sequentiellen Variante. Verwenden Sie hierzu 512 Interlines. Der kürzeste Lauf sollte mindestens 30 Sekunden rechnen; wählen Sie geeignete Parameter aus!

Schreiben Sie ca. ¼ Seite Interpretation zu diesen Ergebnissen. Die Messungen sollen mit Hilfe von SLURM auf den *west*-Rechenknoten durchgeführt werden; führen Sie die Messungen **nicht** auf dem Login-Knoten aus.

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-posix` automatisch eine Binärdatei `partdiff-posix` erzeugt. Der Aufruf von `make` ohne target muss aber ebenfalls weiterhin funktionieren.
- Eine Ausarbeitung `antworten.pdf` mit der Datenaufteilung, den ermittelten Laufzeiten und der Leistungsanalyse

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