Universität Hamburg, Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen
Prof. Dr. T. Ludwig, Dr. Jannek Squar

Übungsblatt 8 zur Vorlesung
Hochleistungsrechnen
im WiSe 2025/2026
Abgabe: 06.12.2024, 23:59

1 Parallelisierung mit MPI (Jacobi: 240 Punkte)

Parallelisieren Sie das Jacobi-Verfahren in dem sequentiellen partdiff-Programm gemäß dem besprochenen Parallelisierungsschema.
Beachten Sie dabei folgende Anforderungen:

- Abbruch
  - Es gibt hier zwei Fälle, die auf Korrektheit der Parallelisierung zu prüfen sind:
    1. Abbruch nach fester Iterationszahl (beide Störfunktionen)
    2. Abbruch nach Genauigkeit (beide Störfunktionen)
  - Prüfen Sie, dass Ihre verteilte Initialisierung korrekt funtkioniert, indem Sie sich Referenzen mit 0 Interlines und 0 Iterationen als Vergleichsmaterial nehmen.
  - Dabei soll nach gleicher Iterationszahl das Ergebnis (Matrix und Fehlerwert) identisch bleiben. Außerdem soll bei Abbruch nach Genauigkeit im parallelen Programm nach derselben Iterationszahl wie im sequentiellen abgebrochen werden.
  - Überprüfen Sie, dass die Ergebnisse mit 24 Prozessen auf zwei Knoten identisch zum sequentiellen (Original als Referenz nehmen) Fall sind.
  - Sie dürfen keine mathematischen Abkürzungen nutzen, die es erlauben würden, nur auf Grundlage eines prozesslokalen Residuums über den Abbruch nach Genauigkeit zu entscheiden. Der Abbruch darf lokal nur auf Grundlage des globale maximalen Residuums über alle Prozesse entschieden werden.
- Code
  - Zu keinem Zeitpunkt darf ein Prozess die gesamte Matrix im Speicher halten, wenn mindestens zwei Prozesse verwendet werden. Die Matrix soll auf alle Prozesse gleichmäßig verteilt werden.
  - Das Programm muss weiterhin mit einem Prozess funktionieren (nur kontrolliert abzubrechen reicht dieses Mal nicht mehr aus).
  - Das Programm muss mit beliebigen Prozesszahlen funktionieren (auch für $nprocs >> N$)
  - Erstellen Sie eine eigene Funktion für die MPI-Parallelisierung des Jacobi-Verfahrens.
  - GS muss dabei weiterhin sequentiell funktionieren.
  - Verwenden Sie Funktion `displayMatrixMPI`, die in den Materialien bereitgestellte wird, als Grundlage für die parallele Ausgabe der Matrix.
    Hinweis: Verwenden Sie diese parallele Ausgabe mit Bedacht (d.h. am Ende),

da dadurch die Teilmatrix vom ausgebenden Rang überschrieben wird! Zwischenausgaben mit dieser Matrix führen also zu einem falschen Endergebnis und sollten nur zu Testzwecken verwendet werden.

- Laufzeit
  - Das Programm darf mit zwei Prozessen nicht langsamer als die sequentielle Variante sein. Es muss ein zufriedenstellender Speedup erreicht werden. Belegen Sie diesen schriftlich nach den üblichen Konventionen.
- Kommunikation
  - Nutzen Sie synchronisierende Funktionen (`MPI_Ssend` und `MPI_Issend`) um ihr Programm auf mögliche Verklemmungen zu untersuchen. Für die finale Abgabe können sie diese Funktionen gegen die Standard-Funktionen wieder austauschen.
  - Jeder nicht-blockierende Kommunikation muss mit einem passenden `MPI_Wait` oder einem erfolgreichen `MPI_Test` abgeschlossen werden. Anderenfalls ist der Aufruf falsch.

2 Hybride Parallelisierung (60 Bonuspunkte)

Erweitern Sie Ihre MPI-Version des Jacobi-Verfahrens zusätzlich um OpenMP.

Leistungsanalyse

Ermitteln Sie nach den üblichen Konventionen die Leistungsdaten Ihres hybriden Programms und vergleichen Sie die Laufzeiten für folgende Konfigurationen in einem beschrifteten Diagramm:

- 3 Knoten $\times$ 12 Prozesse
- 3 Knoten $\times$ 24 Prozesse
- 3 Knoten $\times$ 1 Prozess $\times$ 12 Threads
- 3 Knoten $\times$ 1 Prozess $\times$ 24 Threads
- 3 Knoten $\times$ 2 Prozesse $\times$ 6 Threads
- 3 Knoten $\times$ 2 Prozesse $\times$ 12 Threads
- 3 Knoten $\times$ 12 Prozesse $\times$ 2 Threads

Verwenden Sie hierzu 512 Interlines. Der kürzeste Lauf sollte mindestens 10 Sekunden rechnen. Wählen Sie geeignete Parameter aus und geben Sie diese an!

Schreiben Sie eine halbe Seite Interpretation zu diesen Ergebnissen.
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 des Programms

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; **gut** dokumentiert! (Kommentare bei geänderten Code-Teilen!)
  - Erwartet werden die Dateien `Makefile`, `askparams.c`, `partdiff.c` und `partdiff.h`.
  - Eine Ausarbeitung `leistungsanalyse.pdf` mit den ermittelten Laufzeiten und der Leistungsanalyse.
- **Keine** Binär oder Punktdateien!

Achten Sie darauf, dass ihre Programme ohne Warnungen oder Fehler bauen und Valgrind keine Probleme anzeigt. Packen Sie ein komprimiertes Archiv (`.tar.gz`) aus dem sauberen Verzeichnis (**ohne Binärdateien oder versteckte Dateien**). Benennen Sie das Archiv nach den Nachnamen der Gruppenmitglieder (z. B. `MustermannMusterfrau.tar.gz`).
Ein Mitglied Ihrer Gruppe legt das Archiv dann in

```
$HOME/HR-Abgaben-2526/Blatt-8
```

ab und schickt nach erfolgter Abgabe eine E-Mail an `jannek.squar@uni-hamburg.de`, in der Sie einfach den absoluten Pfad zu Ihrem Archiv angeben:

```
$HOME/HR-Abgaben-2526/Blatt-8/MustermannMusterfrau.tar.gz.
```

Hinweis: `$HOME` ist eine Umgebungsvariable, die auf Ihr Heimatverzeichnis zeigt. Es sollte also nicht wortwörtlich so in der E-Mail stehen.