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

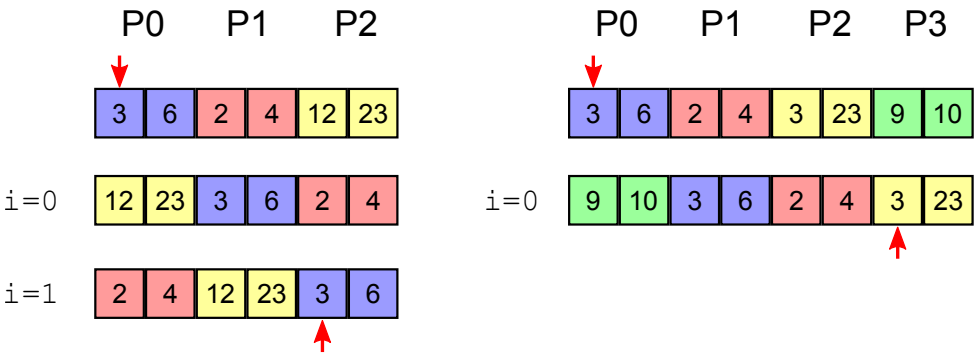
Übungsblatt 7 zur Vorlesung
Hochleistungsrechnen
im WiSe 2022/2023
Abgabe: 03.12.2021, 23:59

1 Circle (120 Punkte)

Schreiben Sie ein Programm `circle.c`.

Beachten Sie dabei folgende Anforderungen:

- Programmverhalten
  - Ein eindimensionales Array der Länge `N` wird auf `nprocs` Prozesse möglichst **gleichmäßig** aufgeteilt. Schreiben Sie in `antworten.pdf` die Formel auf und visualisieren Sie eine Beispielaufteilung für `N=13` und `nprocs=5`.
  - Das Teilarray wird mit zufälligen Werten im Bereich 0–24 (nutzen Sie `rand()%25`) initialisiert. Jeder Prozess allokiert nur so viel Speicher, wie er für seinen Anteil und ggf. ein Element Überschuss benötigt.
  - Die Werte sollen ihrer Reihenfolge im Array nach ausgegeben werden (im Gerüst "Before").
  - Die Prozesse sollen einen Kommunikationsring bilden, wobei jeder Prozess mit seinem Vorgänger und dem Nachfolger kommuniziert (also Rang n mit den Rängen $n-1$ und $n+1$); der letzte Prozess kommuniziert dabei mit Rang 0 und umgekehrt.
  - In der Funktion `circle` kommunizieren die Prozesse ihre Daten entsprechend weiter.
  - In jedem Schleifendurchlauf versendet jeder Prozess seine Daten an seinen Nachfolger und empfängt die Daten seines Vorgängers.
  - Dies passiert solange bis der letzte Prozess am Anfang seines Arrays den Wert des ersten Arrayelementes des ersten Prozesses vor Iterationsbeginn hat.
  - Danach gibt jeder Prozess seinen Speicher in der Reihenfolge der Prozesse aus. Der erste Prozess gibt außerdem die Anzahl der Iterationen und den Abbruchwert aus (im Beispiel 3). Anschließend wird das Programm beendet.



- Code
  - `N` wird dem Programm als erstes und einziges Kommandozeilenargument übergeben.

  - Das Programm soll mit einer beliebigen Anzahl an Prozessen und einer beliebigen Arraylänge umgehen können. Überlegen Sie sich angemessenes Verhalten für 1 Prozess und `nprocs > N`.
  - Zu keinem Zeitpunkt darf ein Prozess das gesamte Array im Speicher halten.
  - Das Programm muss mindestens eine Iteration laufen.

- Abbruch
  - Der Abbruch soll nicht nach Anzahl der Iterationen erfolgen, sondern nach Eintreten der oben beschriebenen Situation. Beachten Sie, dass dieser Fall auch nach weniger als `nprocs - 1` Iterationen auftreten kann, da derselbe Wert mehrfach vorkommen kann.

- Kommunikation
  - Sie dürfen die Funktionen `MPI_Send` und `MPI_Isend` **nicht** verwenden. Nutzen Sie stattdessen ggf. die Funktionen `MPI_Ssend` und `MPI_Issend`.
  - Jeder nichtblockierende Kommunikationsaufruf (meist beginnend mit `MPI_I...`) muss mit einem passenden `MPI_Wait` oder einem erfolgreichen `MPI_Test` abgeschlossen werden. Anderenfalls ist der Aufruf falsch.

2 Visualisierung (40 Punkte)

Visualisieren Sie die Kommunikation der Prozesse aus der vorigen Aufgabe mittels Vampir[1]. Gehen Sie dabei in mehreren Schritten vor:

1. Kompilierung des Programms mit Score-P
2. Ausführen des Programms; hierbei entstehen so genannte Spurdaten (Traces)
3. Visualisierung der Spurdaten in Vampir

Laden Sie zuerst mit `spack load scorep` das Score-P-Paket (auf West-Knoten). Danach müssen Sie bei der Kompilierung das Kommando `scorep` voran stellen (im Makefile vor `mpicc`), wodurch Ihr Programm automatisch instrumentiert wird. Nach dem Setzen der Umgebungsvariable mit `export SCOREP_ENABLE_TRACING=true` und einem Aufruf des Programms werden Spurdaten generiert; standardmäßig werden alle MPI- und Funktions-Aufrufe erfasst. Die Visualisierung erfolgt dann mittels `vampir <subfolder>/programm.otf2`.
Beantworten Sie in `antworten.pdf` folgende Fragen und legen Sie dabei Screenshots ab:

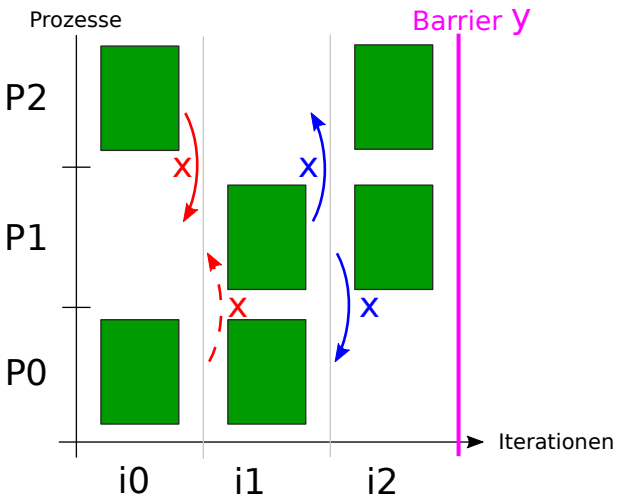
1. Wie können Sie in der grafischen Darstellung die Richtung der Kommunikation erkennen? Korreliert die Darstellung mit ihren Erwartungen?
2. Lassen Sie sich die Communication Matrix View ausgeben.
3. Markieren Sie die unterschiedlichen Programmphasen (Initialisierung, Iterationen und Beenden) in den Screenshots
4. Wie lange hat die `MPI_Init`-Phase gedauert?

[1] `http://www.vampir.eu/`

3 Parallelisierung mit MPI (Schema: 120 Punkte)

In den kommenden Wochen werden Sie das partdiff-Programm mittels Nachrichtenaustausch und MPI zu parallelisieren. Erstellen Sie zunächst ein Parallelisierungsschema für das **Jacobi**-Verfahren.

- Schreiben Sie eine Formel (mathematisch oder Pseudocode) für i(Interlines) und `nprocs` auf, nach der die Matrix möglichst **gleichmäßig** aufgeteilt wird. Beachten Sie dabei die Existenz der Randzeilen. Geben Sie für den ersten (Rang 0), den letzten und einen Beispielprozess aus der Mitte den Schleifenkopf der Berechnung an.
- Visualisieren Sie Ihre Aufteilung für `i=2` und `nprocs=5`. Kennzeichnen Sie die Daten, die jweils kommuniziert werden müssen und zeigen Sie mit Pfeilen, an welche(n) Prozess(e) diese Daten geschickt werden müssen. Beachten Sie dabei, dass zur Berechnung eines Wertes immer jeweils der Wert oben, unten, links und rechts benötigt wird.
- Visualisieren Sie das Kommunikationsschema für 3 Prozesse und 3 Iterationen in einem Diagramm. An der x-Achse werden dabei die Iterationen aufgetragen; an der y-Achse die Prozesse. Legen Sie je ein Diagramm für die zwei unterschiedlichen Abbruchbedingungen an. Überlegen Sie sich vorher, mit welchen Operationen (Punkt-zu-Punkt/kollektiv, blockierend/nicht-blockierend) Sie arbeiten möchten. Wählen sie dabei folgendes Farb- und Objektschema
  - Grüner Block: Berechnungsphasen
  - Durchgezogene Linien (Pfeile): blockierende Kommunikation
  - Gestrichelte Linien (Pfeile): nichtblockierende Kommunikation
  - Rote Pfeile: Versenden (bei Punkt-zu-Punkt)
  - Blaue Pfeile: Empfangen (bei Punkt-zu-Punkt)
  - Pinke Linien: kollektive Operationen
  - Z am Pfeil für die zu kommunizierende Zeile
  - M am Pfeil für das zu kommunizierendes Maxresiduum



- Beschreiben Sie ggf. Problematiken und knifflige Stellen.

Abgabe

- Den Quelltext des C-Programmes `circle.c`.
- Ein `Makefile` derart, dass `make`, `make circle` und `make clean` sich erwartungsgemäß verhalten.
- `antworten.pdf` mit allen Antworten.
- **Keine** Binärdateien!

Senden Sie das Archiv an `hr-abgabe@wr.informatik.uni-hamburg.de`.