Universität Hamburg, Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen
Prof. T. Ludwig, M. Kuhn
A. Droste, A. Fuchs, E. Zickler

Übungsblatt 7 zur Vorlesung
Hochleistungsrechnen
im WiSe 2015/2016
Abgabe: 28.11.2015, 23:59

Einführung in MPI (120 Punkte)

In den Materialien finden Sie das sequentielle Programm `circle.c`; modifizieren Sie es mit Hilfe von MPI so, dass sich die parallele Variante wie folgt verhält:

Ein eindimensionales Array der Länge `N` wird auf `nprocs` Prozesse möglichst gleichmäßig aufgeteilt und mit zufälligen Werten (im Bereich 0–24) initialisiert. `N` wird dem Programm dabei als erstes und einziges Kommandozeilenargument übergeben. Die Werte sollen ihrer Reihenfolge im Array nach ausgegeben werden. Die Prozesse sollen einen Ring 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` tauschen die Prozesse ihre Daten aus. 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 (siehe Abbildung 1). Dann wird der Speicher in der Reihenfolge der Prozesse ausgegeben und das Programm beendet.

Beachten Sie dabei folgende Anforderungen:

- Das Programm soll mit einer beliebigen Anzahl an Prozessen und einer beliebigen Arraylänge umgehen können. Die Eingabe muss auf mögliche Fehler geprüft werden und das Programm soll angemessen reagieren.
- Zu keinem Zeitpunkt darf ein Prozess das gesamte Array im Speicher halten.
- 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 (siehe Abbildung 1).
- Achten Sie auf die richtige Reihenfolge der Ausgabe.

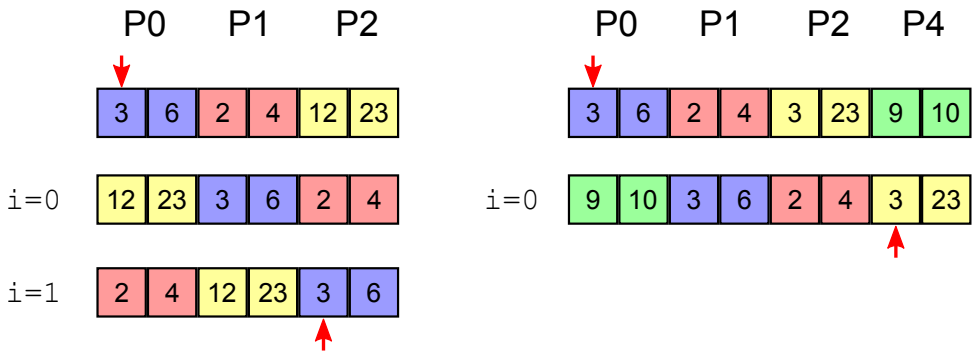

Abbildung 1: Beispiele möglicher Programmabläufe, PX = Prozess X, i = Iteration

Abgabe

- Den Quelltext des C-Programmes `circle.c`.
- Ein `Makefile` derart, dass `make` bzw. `make circle` sich erwartungsgemäß verhalten.
- **Keine** Binärdateien!

Visualisierung (60 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 VampirTrace
2. Ausführen des Programms; hierbei entstehen so genannte Spurdaten (Traces)
3. Visualisierung der Spurdaten in Vampir

Ihr Programm wird bei der Kompilierung mit `vtcc -vt:cc mpicc` (statt `mpicc`) automatisch gegen die VampirTrace-Bibliothek gelinkt. Bei einem Aufruf des Programms werden automatisch Spurdaten generiert; standardmäßig werden alle MPI- und Funktions-Aufrufe erfasst. Die Visualisierung erfolgt mittels `vampir program.otf`.

Abgabe

Abzugeben ist ein PDF-Dokument `visualisierung.pdf` mit folgendem Inhalt:

- Screenshots der Visualisierung
  - Markieren Sie dabei die unterschiedlichen Programmphasen (Initialisierung, Iterationen und Beenden) in den Screenshots
- Beschreibung der visualisierten Daten
  - Wie werden die Prozesse und die MPI-Operationen dargestellt?
- Diskussion des Kommunikationsschemas
  - Kommunizieren die Prozesse so wie Sie es erwarten?

Parallelisierung mit MPI (Parallelisierungsschema: 120 Punkte)

Ziel der kommenden drei Übungsblätter soll es sein, das Programm mittels Nachrichtenaustausch und MPI zu parallelisieren. Zunächst soll auf diesem Blatt lediglich ein Parallelisierungsschema erstellt werden (siehe unten bei der Abgabe).

Gehen Sie so vor, dass Sie die gesamten Matrixdaten auf die Prozesse aufteilen, d. h. jeder Prozess bearbeitet einen Teil der Daten. Beachten Sie dabei folgendes: Zur Berechnung der Werte einer Zeile benötigt man immer die Werte der darüber- und der darunterliegenden Zeile. Wenn also die zu berechnende Zeile die erste oder letzte des Blockes eines Prozesses ist, so wird die benötigte Nachbarzeile von einem Nachbarprozess verwaltet; dieser muss sie dann zusenden. Gleichermaßen muss man, wenn man eine Randzeile neu berechnet hat, sie an den entsprechenden Nachbarprozess weitersenden. Die Problematik liegt darin, dies zum richtigen Zeitpunkt mit den richtigen Werten zu tun.

Beachten Sie auch, dass der Berechnungsablauf bei den beiden Verfahren unterschiedlich ist und somit auch das Kommunikationsschema unterschiedlich ausfallen wird. Beachten Sie weiterhin, dass der erste und der letzte Prozess keinen vorhergehenden bzw. nachfolgenden Nachbarprozess mehr haben. Um das Speicherverhalten zu optimieren, dürfen die Prozesse nur die benötigte Teilmatrix im Speicher halten. Somit können auch Probleme berechnet werden, welche nicht in den Hauptspeicher eines einzelnen Knotens passen.

Gehen Sie auch auf Probleme/Besonderheiten, die sie bei der Beendigung des Programms feststellen, ein. Geben Sie für jeden der vier Fälle eine kurze Beschreibung an, welche Probleme auftreten könnten, und welche Mittel Sie zu deren Lösung vorschlagen würden (falls eine Lösung möglich scheint).

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

Abgabe

Abzugeben ist ein PDF-Dokument `parallelisierungsschema.pdf` mit folgendem Inhalt:

- Prosabeschreibung der Datenaufteilung der Matrix auf die einzelnen Prozesse
  - Welche Daten der Matrix werden von welchem Prozess verwaltet?
- Parallelisierungsschema für das Jacobi-Verfahren
  - Beschreiben Sie aus Sicht eines Prozesses, wann die Berechnung und wann die Kommunikation mit seinen Nachbarn erfolgt.
  - Welche Daten benötigt der Prozess von seinen Nachbarn und wann tauscht er die Daten aus?
- Parallelisierungsschema für das Gauß-Seidel-Verfahren (siehe Jacobi)
- Diskussion der Abbruchproblematik
  - Es sind vier Fälle zu betrachten: Abbruch nach Iterationenzahl und Genauigkeit für jeweils Jacobi und Gauß-Seidel.
  - Wann wird dieser Prozess feststellen, dass das Abbruchkriterium erreicht wurde und er seine Arbeit beenden kann?
  - In welcher Iteration befindet sich der Prozess im Vergleich zu seinen Nachbarn, wenn er das Abbruchkriterium erreicht?

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