Grundlagen der Parallelrechnerprogrammierung

(M. Pester)

Übungsaufgabe

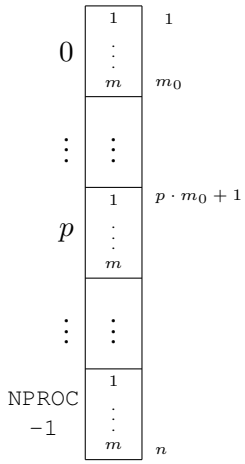
Vorbemerkung: Es ist stets zu beachten, dass ein Programm als Teil eines parallelen Algorithmus immer nur mit **einem** Teil der Daten arbeitet und demzufolge Indizes **lokal** zu zählen sind und nur über solche lokalen Indizes auf die eigenen Substrukturen zugegriffen wird. Die **globalen** Datenstrukturen und Indizes werden nur „in Gedanken“ betrachtet. Es gibt aber bei Kenntnis der Prozessoranzahl und der Verteilungsstrategie immer eine „bijektive“ Abbildung:

globaler Index $\leftrightarrow$ lokaler Index.

Problemstellung: Zwei (globale) Vektoren $x = (x_1, \ldots, x_n)$ und $y = (y_1, \ldots, y_n)$ sollen verteilt auf `NPROC` Prozessoren gespeichert werden[1]. Jeder Prozessor i ($i = 0, \ldots,$ `NPROC`-1) hat damit lokal jeweils einen Vektor $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_{m_i})$ (analog für y), so dass $\sum_i m_i = n$.

Am einfachsten wählt man $m_i = m_0 = const$ auf allen Prozessoren, mit Ausnahme des letzten, der nur noch die verbleibenden $m = n - m_0 \cdot$ (`NPROC` $- 1$) Komponenten speichert, also auf jedem Prozessor:

$$
\begin{aligned}
m_0 &:= \left\lceil \frac{n + \texttt{NPROC} - 1}{\texttt{NPROC}} \right\rceil \qquad \text{(ganzer Anteil)} \\
k_0 &:= m_0 \cdot \texttt{ICH} \qquad \text{(Anz. Komp. vor dem eigenen Teilvektor)} \\
m &:= \min(m_0, n - k_0) \qquad \text{(lokale Anzahl Komponenten)}
\end{aligned}
$$



Das ist nicht unbedingt die „gerechteste“ Verteilung. Man könnte auch mehreren Prozessoren nur je eine Komponente weniger zuordnen, vgl. Seite 5).
Die lokal gespeicherten Komponenten $\tilde{x}_k$ ($k = 1, \ldots, m$) entsprechen somit in der globalen Indizierung den Komponenten x_{k+k_0}, wobei der Index $k_0 + 1$ der (globale) Startindex des Teilvektors auf dem aktuellen Prozessor ist.

Teilaufgaben des Programms:

- Eingabedialog nur auf Prozessor 0 (`ICH = 0`): Eingabe einer Dimension n, ggf. auch Nummer eines Testbeispiels, wenn mehrere vorgesehen sind, oder andere Steuerungs-Parameter.
- Verteilung des Wertes n (dasselbe evtl. für eingegebene Bsp.-Nummer) von Prozessor 0 an alle anderen Prozessoren (`tree_down_0(1,n)`)
- Zum Testen werden die lokalen Komponenten von $\tilde{x}$ und $\tilde{y}$ mit Werten belegt, so dass z.B. in globaler Indizierung gilt: $x_k = k$, $y_k = 1/k$ (also lokal: $\tilde{x}_j = j + k_0$).

  Mehrere solche leicht nachprüfbaren Testbeispiele könnte das Programm zur Auswahl anbieten. Keinesfalls sollen Vektorkomponenten eingetippt werden!
- Lokale Berechnung des Skalarproduktes $S_i = \sum_{k=1}^{m} \tilde{x}_k \tilde{y}_k$.
- Berechnung der Summe $S = \sum S_i$ über alle Prozessoren:

```
EXTERNAL vdplus
...
call cube_dod(1,S,Si,H,vdplus)
```

  Alle Variablen und Felder seien mit `DoublePrecision` bzw. `double` vereinbart. sonst (bei `REAL` bzw. `float`): `cube_doi(1,S,Si,H,vrplus)`.
- Ausgabe des Skalarproduktes S (nur durch Prozessor 0)

[1] Indizes entsprechen der mathematischen Schreibweise und gelten entsprechend für Fortran-Programme. Bei C-Programmierung sind alle Indizes um 1 zu verringern