MATHEMATISCHES INSTITUT
DER UNIVERSITÄT ZU KÖLN

Dr. Martin Lanser

Einführung in das Hochleistungsrechnen

Sommersemester 2019

Übung 5

Hinweis: Schreiben Sie bitte jede Aufgabe auf ein neues Blatt und auf **jedes Blatt Ihren Namen**. Auf die erste Seite Ihrer Übung schreiben Sie bitte zusätzlich zu Ihrem Namen Ihre Matrikelnummer.

Aufgabe 1 (8 Punkte)

Es sei $A \in \mathbb{R}^{n\times n}$ eine dicht besetzte Matrix, die in N^2 Blöcke der Größe $n_b \times n_b$ mit $N = n/n_b$ aufgeteilt ist.

(a) Leiten Sie eine left-looking LU-Zerlegung ohne Pivotisierung in geblockter Form für $N = 3$ her.

(b) Entwickeln Sie einen Algorithmus zur left-looking Variante der geblockten Gauß-Elimination zur Berechnung von $A = LU$ für allgemeines $N \in \mathbb{N}$.

Hinweis: Im j-ten Schritt berechnet man die j-ten Block-Spalten von L und U.

Programmieraufgabe 4 (10 + 4 Punkte).

Achtung: 2 Wochen Bearbeitungszeit!

Implementieren Sie Algorithmus 3.8 der Vorlesung mit Zeilen-Pivotsuche sowie impliziter Spaltenvertauschung, d. h. implementieren Sie die parallele Gauß-Elimination mit Zeilen-Pivotsuche zur Berechnung von $A = LU$ in der kij-Form mit **send ahead**. Nehmen Sie an, dass die Matrix zyklisch nach Zeilen verteilt ist und die entsprechenden Zeilen auf jedem Prozess in einem 2D-double-Array abgespeichert sind.

Implementieren Sie die Matrix-Größe n als Übergabeparameter der Hauptfunktion, d. h. der Aufruf Ihrer übersetzten C-Datei (hier: `gauss`) sollte z. B. für $n = 8$ die folgende Form haben:

```
mpirun -n 3 ./gauss 8
```

Überprüfen Sie Ihre Implementierung mit $p = 3$ MPI-Prozessen mit der Vandermonde-Matrix für den Vektor $v = (1, 2, 3, 4, 5, 6, 7, 8)^T$. Die Vandermonde-Matrix A ist definiert als

$$A = (a_{ij}) \in \mathbb{R}^{n\times n} \quad \text{mit } a_{ij} = v_i^{j-1}.$$

Vergleichen Sie Ihre Ergebnisse mit den Ergebnissen von

```
[L, U, P] = lu(vander(1:8))
```

in MATLAB. **Auf der Homepage gibt es wieder ein Code-Gerüst! Dort ist die main vollständig und nur in der LU-Zerlegung muss Code ergänzt werden.**

Hinweise:

- Führen Sie die Zeilen-Pivotsuche und anschließende implizite Spaltenvertauschung auf allen Prozessen durch. Nach Erhalt der Zeile $(\hat{a}_{k,k}, ..., \hat{a}_{k,n})$ kann jeder Prozess mit diesen Werten die (auf allen Prozessen identische!) Zeilen-Pivotsuche durchführen und für alle lokalen Zeilen die nötigen Spaltenvertauschungen durchführen. Zusätzliche Kommunikation ist dazu nicht erforderlich.
- Implizite Spaltenvertauschung heißt, die innere Schleife in Algorithmus 3.8 läuft nicht mehr von $k+1$ bis n, sondern von $ind(k+1)$ bis $ind(n)$. Im Indexvektor ind werden dabei die Vertauschungen durchgeführt.
- Beachten Sie, dass die send-ahead-Strategie **nicht-blockierende** Kommunikation erfordert. Schauen Sie sich vor allem das nicht blockierende broadcast (**MPI_Ibcast**) an!

Zusatz (Bonus): Implementieren Sie zusätzlich den Algorithmus **ohne** send ahead, d. h. implementieren Sie Algorithmus 3.7 der Vorlesung mit Zeilen-Pivotsuche sowie impliziter Spaltenvertauschung. Messen Sie für beide Fälle (mit und ohne send ahead) die Laufzeit (mittels **MPI_Wtime**) für verschiedene Anzahlen von MPI-Prozesse und verschiedene n. Stellen Sie Ihre Ergebnisse tabellarisch dar. Was fällt Ihnen auf?

Abgabedatum: 23. Mai 2019 bis 12:00 Uhr im entsprechenden Kasten in Raum 3.01 des Mathematischen Instituts oder am Ende der Vorlesung.