MATHEMATISCHES INSTITUT
DER UNIVERSITÄT ZU KÖLN

Dr. Martin Lanser

Einführung in das Hochleistungsrechnen
Wintersemester 2017/2018

Übung 3

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 (4 + 4 = 8 Punkte).
Gegeben seien eine Matrix $A \in \mathbb{R}^{n\times m}$ und ein Vektor $x \in \mathbb{R}^m$. Es sei $m = q \cdot 2^N$ und die Matrix A sei spaltenweise in 2^N Blöcke der Dimension $n \times q$ zerlegt. Der Vektor x sei analog in 2^N Blöcke der Länge q zerlegt.

a) Bestimmen Sie die Laufzeit $t(n, N, q)$ der Matrix-Vektor-Multiplikation $A \cdot x = b$ mit $p = 2^N$ Prozessoren, wobei der Vektor $b \in \mathbb{R}^n$ sequentiell (nicht zerteilt) gespeichert werden soll. Berechnen Sie auch die Effizienz in Abhängigkeit von q, n und N. Nutzen Sie t_k analog zum letzten Übungsblatt zur Modellierung der Kommunikationszeit. Sie können davon ausgehen, dass die Zeit t_{a+m} einer seriellen Addition plus einer Multiplikation genau so lange dauert wie die Zeit für eine Addition t_a.

b) Stellen Sie die Effizienz E für fixierte Werte $\alpha = 1000$, $\beta = 10$, $m = 32768$ und $n = 1000$ in Abhängigkeit von $N = 0, 1, 2, ...$ dar, also in Abhängigkeit von einer steigenden Anzahl an Prozessoren. Nutzen Sie dazu ein einfaches x-y-Diagramm. Beachten Sie, dass $q = \frac{m}{2^N}$ gilt!

Aufgabe 2 (6 Punkte).
Es sei $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ eine Bandmatrix mit unterer Bandbreite $b_l < n$ und oberer Bandbreite $b_u < n$ (typischerweise $b_l, b_u \ll n$), d. h. $a_{ij} = 0$ für $j < i - b_l$ oder $j > i + b_u$.

Um Speicherplatz einzusparen, werden Bandmatrizen „diagonalweise“ als Matrix $\widetilde{A} = (\widetilde{a}_{ij}) \in \mathbb{R}^{n\times(b_l+b_u+1)}$ abgespeichert, indem man die Zuordnung

$$\widetilde{a}_{i,j} = \begin{cases} a_{i,i+j} & \text{für } i = 1, \ldots, n,\ j = -b_l, \ldots, b_u,\ i + j \in \{1, \ldots, n\} \\ 0 & \text{sonst} \end{cases}$$

verwendet (vgl. Abschnitt 2.4.2 der Vorlesung). Die Zeilen von $\widetilde{A}$ werden dabei also wie gewohnt von 1 bis n durchnummeriert, die Spaltenindizes jedoch von $-b_l$ bis b_u. Für $n = 6$

sowie $b_l = 1$ und $b_u = 2$ ergibt sich z. B.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & a_{35} & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & a_{46} \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{pmatrix} \quad \text{und} \quad \widetilde{A} = \begin{pmatrix} 0 & a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{32} & a_{33} & a_{34} & a_{45} \\ a_{43} & a_{44} & a_{45} & a_{46} \\ a_{54} & a_{55} & a_{56} & 0 \\ a_{65} & a_{66} & 0 & 0 \end{pmatrix}.$$

Nun sei $\widetilde{A}$ auf p Prozessoren spaltenweise verteilt, d. h. jeder Prozessor hat Zugriff auf eine Teilmenge von kompletten Spalten der Matrix. Der Vektor $x \in \mathbb{R}^n$ sei auf jedem Prozessor vorhanden.

Entwerfen Sie einen Pseudocode für die **effiziente** Matrix-Vektor-Multiplikation $b = Ax$. Der Ergebnisvektor $b \in \mathbb{R}^n$ soll nach Abschluss der Berechnung vollständig auf einem (beliebigen) Prozessor vorhanden sein.

Programmieraufgabe 2 (6 Punkte).

Gegeben seien eine Matrix $A \in \mathbb{R}^{n \times m}$ und ein Vektor $x \in \mathbb{R}^m$. Es seien $n = r \cdot p$ und $m = q \cdot p$, wobei $p, q, r \in \mathbb{N}$. Die Matrix A sei zeilenweise in p Blöcke der Dimension $r \times m$ zerlegt und der Vektor x sei in p Blöcke der Länge q zerlegt. Implementieren Sie ein paralleles Programm mit p MPI Prozessen (Ränge 0 bis $p - 1$), das die parallele Matrix-Vektor-Multiplikation $b = A \cdot x$ durchführt. Jeder MPI-Prozess soll dabei nur jeweils **einen** Block der Matrix A und **einen** Block des Vektors x speichern. Während der Berechnung darf jeder Prozess zusätzlich temporär maximal **einen** weiteren Block des Vektors x speichern. Der Ergebnisvektor $b \in \mathbb{R}^n$ soll ebenfalls auf die p Prozesse verteilt werden, d. h. b soll in p Blöcke der Länge r zerlegt sein, wobei jeder MPI-Prozess nur **einen** Block von b speichern soll.

Hinweise:

- Ein unvollständiges Code-Gerüst steht auf der Homepage als Download zur Verfügung!
- Verwenden Sie die Anzahl Blöcke r und q als Eingabeparameter für Ihr Programm, um sicherzustellen, dass die Blockgrößen für alle Prozesse gleich sind.
- Als einfachen Testfall für die korrekte Funktionsweise Ihres Programms können Sie auf Prozess P_i, $i = 0, \ldots, p$ für alle Einträge der lokalen k-ten Zeile ($k = 1, \ldots, r$) den Wert $(i + 1) \cdot (k/p)$ wählen und 1 für alle Einträge des Vektors x.
- Nutzen Sie für Ihre Implementierung die Funktion **MPI_Bcast**, die eine (blockierende) Broadcast-Operation durchführt, d. h. ein Prozess sendet eine Nachricht an alle Prozesse in einem Kommunikator.

Abgabedatum: 16. November 2017 bis 18:00 Uhr im entsprechenden Kasten in Raum 3.01 des Mathematischen Instituts.