Universität Stuttgart
Institut für parallele und verteilte Systeme
Prof. Dr. rer. nat. Miriam Mehl
Dipl.-Ing. Florian Lindner

SS 2016

Parallel Numerics

Exercise 3: Sparse Matrices

1 Sparse Matrices and Graphs

Given the matrix

$$A = \begin{pmatrix} 4 & 0 & 7 & 0 & 0 & 0 \\ 0 & 5 & 42 & 0 & 12 & 3 \\ 0 & 4 & 7 & 2 & 0 & 5 \\ 0 & 0 & 44 & 32 & 0 & 0 \\ 0 & 0 & 0 & 0 & 99 & 8 \\ 100 & 2 & 0 & 0 & 0 & 102 \end{pmatrix}$$

i) What methods we know to store sparse matrices? What are advantages and disadvantages? Show how to store matrix A using the various formats.

Dense storage Also works with sparse but no storage space gain is achieved.

Coordinate form A table that contains non-zero values and their coordinates. Redundant information is stored because there is no order prescribed. Retrieving information from memory involves indirect adressing. Advantage is that transposition is easy.

value	4	7	5	42	12	3	4	7	2	5	44	32	99	8	100	2	102
row	1	1	2	2	2	2	3	3	3	3	4	4	5	5	6	6	6
col	1	3	2	3	5	6	2	3	4	6	3	4	5	6	1	2	6

Compressed Row Format (CSR) Like coordinate from but avoids saving the row for each entry. Tranposition requires changing format or the matrix-vector multiplication code.

value	4	7	5	42	12	3	4	7	2	5	44	32	99	8	100	2	102
col	1	3	2	3	5	6	2	3	4	6	3	4	5	6	1	2	6
new row	1		3				7				11		13		15		

Compressed Column Format (CCS) Just like CSR, but stores rows and pointer to new column.

Rectangular Storage by Pressing from the Right Stores rows with minimal number of zeros in a matrix and the column index in another matrix. Advantages are that

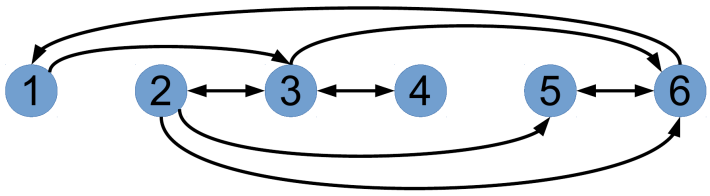
resulting matrices are rectangular, in contrast to the other methods which do not retain the matrix format.

$$C = \begin{pmatrix} 4 & 7 & 0 & 0 \\ 5 & 42 & 12 & 3 \\ 4 & 7 & 2 & 5 \\ 44 & 32 & 0 & 0 \\ 99 & 8 & 0 & 0 \\ 100 & 2 & 102 & 0 \end{pmatrix} \quad J = \begin{pmatrix} 1 & 3 & * & * \\ 2 & 3 & 5 & 6 \\ 2 & 3 & 4 & 6 \\ 3 & 4 & * & * \\ 5 & 7 & * & * \\ 1 & 2 & 6 & * \end{pmatrix}$$

ii) What graph datastructure can be used to represent the sparsity pattern of a (non-) symmetric matrix? How many nodes are needed?
Symmetric matrix: undirected graph, non-symmetric matrix: directed graph. In either case there are n nodes for a $n \times n$ matrix needed.

iii) What is an adjacency matrix? How can it be obtained? How does it relate to the beforementioned graph?
The adjacency matrix also represents the sparsity pattern and contains the same information as the graph. Adjacency matrix and graph can therefore be transformed into each other. It is obtained by replacing each non-zero entry with a 1 resp. boolean true.

iv) Give the graph of matrix A. Based on the properties of the matrix, what kind of graph you need?
The matrix is non-symmetric, a directed graph is needed.



(the main-diagonal is omited here)

v) How can not only the adjacency of a matrix be saved in a graph, but also the values?
The values can be saved at the edges of the graph.

2 Parallel QR

We consider tall and skinny QR as used, e.g. in the first step of block-wise parallel QR decomposition.

$$\overbrace{\begin{bmatrix} \\ A \\ \\ \end{bmatrix}}^{m} = \begin{bmatrix} \\ Q \\ \\ \end{bmatrix} \cdot \begin{bmatrix} R \end{bmatrix}, \quad A \in \mathbb{R}^{n \times m}$$

based on the Gram-Schmidt orthonormalization as described in the lecture notes in 4.2. For parallelization, we use a decomposition of A und Q into p row-blocks. Processor i in a distributed memory system stores A_i and Q_i. R is stored at the master process ($i = 1$).

$$\begin{bmatrix} \boxed{A_1} \\ \boxed{A_2} \\ \vdots \\ \boxed{A_p} \end{bmatrix} = \begin{bmatrix} \boxed{Q_1} \\ \boxed{Q_2} \\ \vdots \\ \boxed{Q_p} \end{bmatrix} \cdot \begin{bmatrix} R \end{bmatrix}$$

i) Calculate the communication costs, i.e.

   a) The number of messsages depending on the number of processors and

   b) the amount of data (number of matrix entries)

   for each of the five algorithmic steps and for the whole QR-decomposition of A.
   The number of messages is reduced from $\log(p)$ to $2\,n\,\log(p)$, which is typically for fan-in procedures.
   The amount of data is also reduced, but less, compared to the number of messages.

ii) Compare i with the communication avoiding QR-algorithm for $p = 2^q$, which calculates small QR-decomposition for the blocks of A independently and combines them to larger

blocks in a fan-in-like process

$$\begin{bmatrix} \boxed{A_1} \\ \boxed{A_2} \\ \boxed{A_3} \\ \boxed{A_4} \end{bmatrix} \Rightarrow$$

$$\begin{array}{l} p=1: \quad \boxed{A_1} = \boxed{Q_1}\boxed{R_1} \\ p=2: \quad \boxed{A_2} = \boxed{Q_2}\boxed{R_2} \\ p=3: \quad \boxed{A_3} = \boxed{Q_3}\boxed{R_3} \\ p=4: \quad \boxed{A_4} = \boxed{Q_4}\boxed{R_4} \end{array} \Rightarrow \begin{array}{l} p=1: \quad \begin{pmatrix} \boxed{Q_1} & \\ & \boxed{Q_2} \end{pmatrix} \begin{pmatrix} \boxed{R_1} \\ \boxed{R_2} \end{pmatrix} = \begin{pmatrix} \boxed{Q_1} & \\ & \boxed{Q_2} \end{pmatrix} \begin{bmatrix} \bar{Q}_1 \\ \\ \bar{Q}_3 \end{bmatrix} \begin{bmatrix} \bar{R}_1 \\ \\ \bar{R}_3 \end{bmatrix} \\ p=3: \quad \begin{pmatrix} \boxed{Q_3} & \\ & \boxed{Q_4} \end{pmatrix} \begin{pmatrix} \boxed{R_3} \\ \boxed{R_4} \end{pmatrix} = \begin{pmatrix} \boxed{Q_3} & \\ & \boxed{Q_4} \end{pmatrix} \end{array} \Rightarrow$$

$$p=1: \begin{pmatrix} \boxed{Q_1} & & & \\ & \boxed{Q_2} & & \\ & & \boxed{Q_3} & \\ & & & \boxed{Q_4} \end{pmatrix} \begin{pmatrix} \boxed{\bar{Q}_1} & \\ & \boxed{\bar{Q}_3} \end{pmatrix} \begin{pmatrix} \boxed{\bar{R}_1} \\ \boxed{\bar{R}_3} \end{pmatrix} =$$

$$\underbrace{\begin{pmatrix} \boxed{Q_1} & & & \\ & \boxed{Q_2} & & \\ & & \boxed{Q_3} & \\ & & & \boxed{Q_4} \end{pmatrix} \begin{pmatrix} \boxed{\bar{Q}_1} & \\ & \boxed{\bar{Q}_3} \end{pmatrix} \begin{bmatrix} \bar{\bar{Q}}_1 \\ \\ \\ \end{bmatrix}}_{Q} \begin{bmatrix} R \\ \\ \\ \end{bmatrix}$$

iii) Implement both variants, compare scalability and runtimes for $m = 100, 3200$ and $p = 1, 2, 4, 8, 16, 32$.