Doubly Compressed Sparse Column (DCSC) Storage

Kalyani Gadgil

December 9, 2017

Need for Hypersparse Matrices

Matrix is hypersparse if number of non-zero elements is much less than the dimensions of the matrix

$$nnz < n$$

Hypersparse matrices arise after 2-dimensional block data decomposition of matrices for parallel processing like in SUMMA.

Storage Complexity of Sparse Matrices for SUMMA

1. Blocks of size $(n/\sqrt{p}) \times (n/\sqrt{p})$
   where n - matrix dim and p - number of processors
2. Storing each of these submatrices in CSC format
   $O(n\sqrt{p} + +nnz)$
3. Storing whole matrix $O(n + nnz)$ on single processor
4. Storing matrix in DCSC format requires $O(nnz)$

Arrays in DCSC

1. JC in CSC allows fast access to columns but not rows
2. solution could be to store CSR as well but that doubles storage
3. information theoretic solution is to remove unnecessary repetitions from JC $\Rightarrow$ CP array formed
4. CP array contains pointers to row indices of nnz elements
5. compressing JC array from n+1 to nzc (number of columns containing atleast one non-zero element), leads to indexing issues
6. this $\Rightarrow$ form an auxilliary array (AUX) to store pointers to nonzero columns

DCSC Storage

Matrix A

$$
\begin{array}{c c}
 & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 4
\end{pmatrix}
\end{array}
$$

n = 6, nnz = 4

$\lceil cf \rceil = n + 1/nzc = 7/3 =$

$\lceil 2.33 \rceil = 3$

NUM = [1, 2, 3, 4]

row idx (IR) = [0, 3, 2, 5]

JC from CSC $= [0, 0, 2, | \; 3, 3, 3, | \; 4]$

col idx = [1, 1, 2, 5]

DCSC Storage Walkthrough

Matrix A

$$\begin{array}{c|c}
 & \begin{array}{cccccc} 0 & \boxed{1} & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{pmatrix} 0 & \boxed{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & \boxed{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{pmatrix}
\end{array}$$

$$\text{NUM} = [\ \boxed{1, 2}\ ]$$

$$\text{IR} = [\boxed{0, 3}]$$

$$\text{idx\_IR} = [\ \boxed{0}\ , 1]$$

CP stores ptrs to idx of IR when col changes

$$\text{CP} = [\ \boxed{0}\ ]$$

JC stores column indices

$$\text{JC} = [\ \boxed{1}\ ]$$

$$\text{idx\_JC} = [\ \boxed{0}\ , 1]$$

AUX stores one ptr to idx of JC for each chunk

$$\text{AUX} = [\ \boxed{0}\ ]$$

DCSC (contd. 1)

Matrix A

$$\begin{array}{c|c} & \begin{array}{cccccc} 0 & 1 & \boxed{2} & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \boxed{3} & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{pmatrix} \end{array}$$

$$\text{NUM} = [1, 2, \boxed{3}]$$

$$\text{IR} = [0, 3, \boxed{2}]$$

$$\text{idx\_IR} = [0, 1, \boxed{2}]$$

CP stores ptrs to idx of IR when col changes

$$\text{CP} = [0, \boxed{2}]$$

JC stores column indices

$$\text{JC} = [1, \boxed{2}]$$

$$\text{idx\_JC} = [0, \boxed{1}, 2]$$

AUX stores one ptr to idx of JC for each chunk

$$\text{AUX} = [0]$$

DCSC (contd. 2)

Matrix A

$$
\begin{array}{c c}
 & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & \boxed{5} \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \boxed{4}
\end{pmatrix}
\end{array}
$$

NUM = [1, 2, 3, $\boxed{4}$]

IR = [0, 3, 2, $\boxed{5}$]

idx_IR = [0, 1, 2, $\boxed{3}$]

CP stores ptrs to idx of IR when col changes

CP = [0, 2, $\boxed{3}$]

JC stores column indices

JC = [1, 2, $\boxed{5}$]

idx_JC = [0, 1, $\boxed{2}$, 3]

AUX stores one ptr to idx of JC for each chunk

AUX = [0, $\boxed{2}$]

DCSC (contd. 3)

$$\text{NUM} = [1, 2, 3, 4]$$

Matrix A

$$\begin{array}{c c} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{pmatrix} \end{array}$$

$$\text{IR} = [0, 3, 2, 5]$$

$$\text{idx\_IR} = [0, 1, 2, 3]$$

CP stores ptrs to idx of IR when col changes

$$\text{CP} = [0, 2, 3, \boxed{4}]$$

JC stores column indices

$$\text{JC} = [1, 2, 5]$$

$$\text{idx\_JC} = [0, 1, 2, 3]$$

AUX stores one ptr to idx of JC for each chunk

$$\text{AUX} = [0, 2, 3, \boxed{4}]$$

End of matrix reached therefore, nnz added to CP and AUX

DCSC Discussion

1. NUM, IR require nnz storage
2. JC requires nzc
3. CP requires nzc+1
4. AUX is approximately of size nzc

[0] [1] Buluc, A., & Gilbert, J. R. (2008). On the Representation and Multiplication of Hypersparse Matrices. https://doi.org/10.1109/IPDPS.2008.4536313

Access element at A(i,j)

Pointers mean indices or location of value inside an array/vector. Not pointers to physical memory addresses of values.

1. find out which column chunk the element belongs to using chunk sizes determined as $\lceil cf \rceil = (n+1)/nzc$
2. get index $AUX[j/chunk]..+1$ which returns subarray of nonzero columns in that chunk
3. Search this subarray of JC for j; if found, store the index *pos*
4. if found, we know right now that there is some nnz element in this row. Now we search for the specific row we need
5. Use $CP[pos]..+1$ to get subarray of all elements in a particular row
6. search subarray of IR for i; if found, store index as *posc*
7. Use index to get value at that position $NUM[posc]$

```
start ← AUX[⌊j/chunk⌋] ;
end ← AUX[⌊j/chunk⌋ + 1] ;
pos ← Search (j, JC[start... (end-1)]) ;
if pos = null then
    return 0 ;
else
    startc ← CP[pos] ;
    endc ← CP[pos + 1] ;
    posc ← Search (i, IR[startc... (endc-1) ]) ;
    if posc = null then
        return 0 ;
    else
        return NUM[posc] ;
    end
end
```

Algorithm 4.1: **Indexing for** $A(i, j)$

Illustration of Indexing

Matrix A

$$
\begin{array}{c c}
 & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 4
\end{pmatrix}
\end{array}
$$

Search for A(i,j) = A(2,2)

n = 6, nnz = 4, nzc = 3

$$\lceil cf \rceil = n + 1/nzc = \lceil 2.33 \rceil$$

$$\therefore \lceil cf \rceil = 3$$

$$\text{Let } s = \lfloor j/cf \rfloor = \lfloor 2/3 \rfloor = \lfloor 0.66 \rfloor$$

$$\therefore s = 0$$

Illustration of Indexing

Matrix A

$$\begin{array}{c} \\ 0\\1\\2\\3\\4\\5 \end{array}\begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{pmatrix} \end{array}$$

Search for A(i,j) = A(2,2)

AUX = [0, 2, 3, 4]

AUX[0] = 0 and AUX[0+1] = 2

Search JC[0...(2-1)] = 1,2 for j=2

where JC = [1, 2, 5]

Found!

$\therefore$ pos = index of JC where j found

pos = 1

Illustration of Indexing

Matrix A

$$
\begin{array}{c c}
 & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ \boxed{2} \\ 3 \\ 4 \\ 5 \end{array} &
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\boxed{0} & \boxed{0} & \boxed{3} & \boxed{0} & \boxed{0} & \boxed{0} \\
0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 4
\end{pmatrix}
\end{array}
$$

Search for A(i,j) = A(2,2)

Let sc = CP[pos]

where CP = [0, 2, 3, 4]

CP[1] = 2 and CP[1+1] = 3

Search IR[2...(3-1)] = 2 for i=2

where IR = [0, 3, 2, 5]

Found!

$\therefore$ pos = index of IR where i found

posc = 2

Illustration of Indexing

Matrix A

$$
\begin{array}{c c}
 & \begin{array}{cccccc} 0 & 1 & \boxed{2} & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ \boxed{2} \\ 3 \\ 4 \\ 5 \end{array} &
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \boxed{3} & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 4
\end{pmatrix}
\end{array}
$$

Search for A(i,j) = A(2,2)

NUM[posc] = NUM[2]
where NUM = [1, 2, 3, 4]
$\therefore$ value at A(2,2) = 3

Complexity of Search

1. The expected cost of column-wise indexing is constant assuming that nonzero columns (columns that contain at least one nonzero) are distributed evenly.
2. Worst-case performance of column-wise indexing is $log_{10}\lceil cf \rceil$

References

[1] Buluc, A., & Gilbert, J. R. (2008). On the Representation and Multiplication of Hypersparse Matrices.
https://doi.org/10.1109/IPDPS 2008.4536313