frovedis::jds_matrix<T,I,O,P>

NAME

`frovedis::jds_matrix<T,I,O,P>` - A two-dimensional row-wise distributed sparse matrix with jagged diagonal storage.

SYNOPSIS

```
#include <frovedis/matrix/jds_matrix.hpp>
```

Constructors

jds_matrix ();
jds_matrix (const `crs_matrix<T,I,O>`& m);

Public Member Functions

void debug_print ();

Public Data Members

`frovedis::node_local<jds_matrix_local<T,I,O,P>>` data;
size_t num_row;
size_t num_col;

DESCRIPTION

In the CRS format, the rows of the matrix can be reordered decreasingly according to the number of non-zeros per row. Then the compressed and permuted diagonals can be stored in a linear array. The new data structure is called jagged diagonals. The number of jagged diagonals is equal to the number of non-zeros in the first row, i.e., the largest number of non-zeros in any row of the sparse matrix.

A JDS (Jagged Diagonal Storage) matrix is one of the popular sparse matrices with such jagged diagonals (the elements stored in column-major order). It has four major components while storing the non-zero elements, as explained below along with the number of rows and the number of columns in the sparse matrix.

```
val: a vector containing the non-zero elements of the jagged diagonals
of the matrix (in column-major order).
idx: a vector containing the column indices for each non-zero elements
```

```
in the jagged diagonals.
off: a vector containing the jagged diagonal offsets.
perm: a vector containing the indices of the permuted rows.
```

For example, if we consider the below sparse matrix:

```
1 0 0 0 1 0
0 5 9 0 2 0
0 1 0 4 0 0
0 0 0 1 0 5
```

then its JDS image can be thought of as:

```
5 9 2
1 5
1 4
1 1
```

Note that 2nd row of the matrix is having maximum non-zero elements. So this
matrix will have 3 jagged diagonals. Rest three rows are having 2 non-zero elements each which can be permuted in any order (in this case row: 4th -> 3rd -> 1st).

Now when storing the diagonals, its JDS representation would be:

```
val:  {5, 1, 1, 1, 9, 5, 4, 1, 2}
idx:  {1, 3, 1, 0, 2, 5, 3, 4, 4}
off:  {0, 4, 8, 9}
perm: {1, 3, 2, 0}
```

Jagged diagonal offset starts with 0 and it has n+1 number of elements, where n is the number of jagged diagonals in the sparse matrix. The difference between i+1th element and ith element in offset indicates number of non-zero elements present in ith jagged diagonal.

`jds_matrix<T,I,O,P>` is a two-dimensional template based distributed sparse data storage supported by frovedis. It contains public member "data" of the type `node_local<jds_matrix_local<T,I,O,P>>`. The actual distributed matrices are contained in all the worker nodes locally, thus named as `jds_matrix_local<T,I,O,P>` (see manual of ell_matrix_local) and "data" is the reference to these local matrices at worker nodes. It also contains dimension information related to the global matrix i.e., number of rows and number of columns in the original sparse matrix.

The structure of this class is as follows:
template struct jds_matrix { frovedis::node_local> data; // local matrix information size_t local_num_row; // number of rows in the sparse matrix
size_t local_num_col; // number of columns in the sparse matrix
};

For example, if the above sparse matrix with 4 rows and 6 columns is distributed row-wise over two worker nodes, then the distribution can be shown as:

```
master                          worker0                      Worker1
------                          -------                      -------
jds_matrix<int,size_t,        -> jds_matrix_local<int,     -> jds_matrix_local<int,
           size_t,size_t>        size_t,size_t,size_t>        size_t,size_t,size_t>
   *data: node_local<           val: vector<int>             val: vector<int>
```

```
      jds_matrix                    ({5,1,9,1,2})              ({1,1,5,4})
         _local<int,        idx: vector<size_t>          idx: vector<size_t>
      size_t,size_t,                ({1,0,2,4,4})              ({3,1,5,3})
             size_t>>       off: vector<size_t>          off: vector<size_t>
                                   ({0,2,4,5})                ({0,2,4})
                             perm: vector<size_t>         perm: vector<size_t>
                                   ({1,0})                    ({1,0})
  num_row: size_t (4)         local_num_row: size_t (2)    local_num_row: size_t (2)
  num_col: size_t (6)         local_num_col: size_t (6)    local_num_col: size_t (6)
```

The `node_local<jds_matrix_local<int,size_t,size_t,size_t>>` object "data" is simply a (*)handle of the (->)local matrices at worker nodes.

Constructor Documentation

jds_matrix ()

This is the default constructor which creates an empty distributed jds matrix without any memory allocation at worker nodes.

jds_matrix (const crs_matrix<T,I,O>& m)

This is the implicit conversion constructor which creates a new jds matrix by converting the input crs matrix.

Public Member Function Documentation

void debug_print ()

It prints the information related to the compressed jagged diagonal storage (val, idx, off, perm, number of rows and number of columns) node-by-node on the user terminal. It is mainly useful for debugging purpose.

Public Data Member Documentation

data

An instance of `node_local<jds_matrix_local<T,I,O,P>>` type to contain the reference information related to local matrices at worker nodes.

num_row

A size_t attribute to contain the total number of rows in the 2D matrix view.

num_col

A size_t attribute to contain the total number of columns in the 2D matrix view.

SEE ALSO

jds_matrix_local, crs_matrix, ell_matrix