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

NAME

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

SYNOPSIS

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

Constructors

jds_matrix_local ();
jds_matrix_local (const `jds_matrix_local<T,I,O,P>`& m);
jds_matrix_local (`jds_matrix_local<T,I,O,P>`&& m);
jds_matrix_local (const `crs_matrix_local<T,I,O>`& m);

Overloaded Operators

`jds_matrix_local<T,I,O,P>`& operator= (const `jds_matrix_local<T,I,O,P>`& m);
`jds_matrix_local<T,I,O,P>`& operator= (`jds_matrix_local<T,I,O,P>`&& m);

Public Member Functions

void savebinary (const std::string& dir);
void debug_print ();

Public Data Members

`std::vector<T>` val;
`std::vector<I>` idx;
`std::vector<O>` off;
`std::vector<P>` perm;
size_t local_num_row;
size_t local_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_local<T,I,O,P>` is a two-dimensional template based non-distributed sparse data storage supported by frovedis. The structure of this class is as follows:

```
template <class T, class I=size_t, class O=size_t, class P=size_t>
struct jds_matrix_local {
  std::vector<T> val;     // to contain non-zero elements of type "T"
```

```
  std::vector<I> idx;     // to contain column indices of type "I" (default: size_t)
  std::vector<O> off;     // to contain offsets of type "O" (default: size_t)
  std::vector<P> perm     // to contain permuted row indices of type "P" (default: size_t)
  size_t local_num_row;   // number of rows in the sparse matrix
  size_t local_num_col;   // number of columns in the sparse matrix
};
```

Constructor Documentation

jds_matrix_local ()

This is the default constructor which creates an empty jds matrix with local_num_row = local_num_col = 0.

jds_matrix_local (const jds_matrix_local<T,I,O,P>& m)

This is the copy constructor which creates a new jds matrix by deep-copying the contents of the input jds matrix.

jds_matrix_local (jds_matrix_local<T,I,O,P>&& m)

This is the move constructor. Instead of copying the input matrix, it moves the contents of the input rvalue matrix to the newly constructed matrix. Thus it is faster and recommended to use when input matrix will no longer be used in a user program.

jds_matrix_local (const crs_matrix_local<T,I,O>& m)

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

Overloaded Operator Documentation

jds_matrix_local<T,I,O,P>& operator= (const jds_matrix_local<T,I,O,P>& m)

It deep-copies the input jds matrix into the left-hand side matrix of the assignment operator "=".

jds_matrix_local<T,I,O,P>& operator= (jds_matrix_local<T,I,O,P>&& m)

Instead of copying, it moves the contents of the input rvalue jds matrix into the left-hand side matrix of the assignment operator "=". Thus it is faster and recommended to use when input matrix will no longer be used in a user program.

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) on the user terminal. It is mainly useful for debugging purpose.

void savebinary (const std::string& dir)

It writes the elements of a jds matrix to the specified directory as little-endian binary data.

The output directory will contain four files, named “nums”, “val”, “idx”,
“off” and “perm”. “nums” is a text file containing the number of rows and number of columns information in first two lines of the file. And rest four files contain the binary data related to compressed jagged diagonal storage.

Public Data Member Documentation

val

An instance of `std::vector<T>` type to contain the non-zero elements of the (jagged diagonals elements) of the sparse matrix.

idx

An instance of `std::vector<I>` type to contain the column indices of the jagged diagonal elements of the sparse matrix.

off

An instance of `std::vector<O>` type to contain the jagged diagonal offsets.

perm

An instance of `std::vector<P>` type to contain the permuted row indices.

local_num_row

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

local_num_col

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

Public Global Function Documentation

`jds_matrix_local<T,I,O,P>` make_jds_matrix_local_loadbinary(dirname)

Parameters
dirname: A string object containing the name of the directory having the data to be loaded. It expects five files to be presented inside the specified directory, as follows:

- “nums” (containing number of rows and number of columns separated with new-line),
- “val” (containing binary data for non-zero elements),

- "idx" (containing binary column indices),
- "off" (containing binary offset values) and
- "perm" (containing binary permuted row indices)

Purpose
This function loads the little-endian binary data from the specified directory and creates a `jds_matrix_local<T,I,O,P>` object filling the data loaded. The desired value type, "T" (e.g., int, float, double etc.) must be specified explicitly when loading the matrix data. If not specified, the other three types "I", "O" and "P" would be size_t as default types.

For example, considering "./bin" is a directory having the binary data to be loaded,

```
auto m1 = make_jds_matrix_local_loadbinary<int>("./bin");
auto m2 = make_jds_matrix_local_loadbinary<float>("./bin");
```

"m1" will be a `jds_matrix_local<int,size_t,size_t,size_t>`, whereas
"m2" will be a `jds_matrix_local<float,size_t,size_t,size_t>`.

Return Value
On success, it returns the created matrix of the type `jds_matrix_local<T,I,O,P>`. Otherwise, it throws an exception.

`jds_matrix_local<T,I,O,P>` crs2jds(m)

Parameters
m: An object of the type `crs_matrix_local<T,I,O>`.

Purpose
This function converts an input crs storage into an equivalent jds storage of the same "val", "num" and "off" type. The input matrix would remain unchanged.

Return Value
On success, it will return the converted `jds_matrix_local<T,I,O,P>`. Otherwise, it throws an exception.

`std::vector<T>` operator*(m,v)

Parameters
m: A const& object of the type `jds_matrix_local<T,I,O,P>`.
v: A const& object of the type `std::vector<T>`.

Purpose
This function performs matrix-vector multiplication between a sparse jds matrix object with a std::vector of same value (T) type. It expects the size of the input vector should be greater than or equal to the number of columns in the input jds matrix.

Return Value
On success, it returns the resultant vector of the type `std::vector<T>`. Otherwise, it throws an exception.

SEE ALSO

crs_matrix_local, ell_matrix_local, jds_matrix