frovedis::crs_matrix<T,I,O>

NAME

`frovedis::crs_matrix<T,I,O>` - A two-dimensional row-wise distributed sparse matrix with compressed row storage.

SYNOPSIS

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

Constructors

crs_matrix ();
crs_matrix (`frovedis::node_local<crs_matrix_local<T,I,O>>`&& d);

Public Member Functions

void save (const std::string& file);
void savebinary (const std::string& dir);
void debug_print ();
void debug_pretty_print ();
`crs_matrix<T,I,O>` transpose ();
`sparse_vector<T,I>` get_row(size_t r);
void clear();

Public Data Members

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

DESCRIPTION

A CRS (Compressed Row Storage) matrix is one of the most popular sparse matrices. It has three 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 matrix (in row-major order).
idx: a vector containing the column indices for each non-zero elements.
off: a vector containing the row-offsets.
```

For example, if we consider the below sparse matrix:

```
1 0 0 0 2 0 0 4
0 0 0 1 2 0 0 3
1 0 0 0 2 0 0 4
0 0 0 1 2 0 0 3
```

then its CRS representation would be:

```
val: {1, 2, 4, 1, 2, 3, 1, 2, 4, 1, 2, 3}
idx: {0, 4, 7, 3, 4, 7, 0, 4, 7, 3, 4, 7}
off: {0, 3, 6, 9, 12}
```

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

`crs_matrix<T,I,O>` is a two-dimensional template based distributed sparse data storage supported by frovedis. It contains public member "data" of the type `node_local<crs_matrix_local<T,I,O>>`. The actual distributed matrices are contained in all the worker nodes locally, thus named as `crs_matrix_local<T,I,O>` (see manual of crs_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 <class T, class I=size_t, class O=size_t>
struct crs_matrix {
  frovedis::node_local<crs_matrix_local<T,I,O>> data;     // local matrix information
  size_t num_row;   // number of rows in the global sparse matrix
  size_t num_col;   // number of columns in the global sparse matrix
};
```

For example, if the above sparse matrix with 4 rows and 8 columns is distributed row-wise over two worker nodes, then the distribution can be shown as:

```
 master                          worker0                         worker1
-----                           -----                           -----
 crs_matrix<int,size_t,size_t>  -> crs_matrix_local<int,        -> crs_matrix_local<int,
                                          size_t,size_t>                  size_t,size_t>
    *data: node_local<           val: vector<int>                val: vector<int>
          crs_matrix                    ({1,2,4,1,2,3})                 ({1,2,4,1,2,3})
              _local<int,        idx: vector<size_t>             idx: vector<size_t>
           size_t,size_t>>              ({0,4,7,3,4,7})                 ({0,4,7,3,4,7})
                                 off: vector<size_t>             off: vector<size_t>
                                        ({0,3,6})                       ({0,3,6})
     num_row: size_t (4)         local_num_row: size_t (2)       local_num_row: size_t (2)
     num_col: size_t (8)         local_num_col: size_t (8)       local_num_col: size_t (8)
```

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

Constructor Documentation

crs_matrix ()

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

crs_matrix (`frovedis::node_local<crs_matrix_local<T,I,O>>`&& data)

This is the parameterized constructor which accepts an rvalue of the type `node_local<crs_matrix_local<T,I,O>>` and *moves* the contents to the created distributed crs matrix.

In general, this constructor is used internally by some other functions. But user may need this constructor while constructing their own distributed crs matrix using the return value of some function (returning a `crs_matrix_local<T,I,O>`) called using "frovedis::node_local::map".
(thus returned value would be an object of type `node_local<crs_matrix_local<T,I,O>`)

Public Member Function Documentation

`sparse_vector<T,I>` get_row(size_t r)

It returns the requested row of the target sparse matrix in the form of `sparse_vector<T,I>` which contains a vector of type "T" for the non-zero elements in the requested row and a vector of type "I" for their corresponding column indices. If r > local_num_row, then it will throw an exception.

void debug_print ()

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

void debug_pretty_print ()

Unlike debug_print(), it prints the distributed compressed row storage as a view of a two dimensional dense storage on the user terminal node-by-node. It is mainly useful for debugging purpose.

`crs_matrix<T,I,O>` transpose ()

It returns the transposed crs_matrix of the source matrix object.

void save (const std::string& file)

It writes the elements of a distributed crs matrix to the specified file as text data with the format "index:value" for each non-zero elements.

void savebinary (const std::string& dir)

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

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

void clear()

It clears the memory space for the allocated `crs_matrix_local<T,I,O>` per worker.

`crs_matrix<TT,II,OO>` change_datatype()

This function can be used in order to change the triplet type of the target crs_matrix from `<T, I, O>` to `<TT, II, OO>`, where these two type triplets must be compatible.

Public Data Member Documentation

data

An instance of `node_local<crs_matrix_local<T,I,O>>` 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.

Public Global Function Documentation

`crs_matrix<T,I,O>` make_crs_matrix_load(filename)

Parameters
filename: A string object containing the name of the text file having the data to be loaded.

Purpose
This function loads the text data from the specified file and creates a `crs_matrix<T,I,O>` object filling the data loaded.

The input file for the sparse data should be in the below format:

```
1:2 3:2
2:5
1:3 3:4 6:3
3:2 4:5
```

Where each sparse row is represented as "column_index:value" (column_index starts at 0). Note that there can be empty rows in the given file indicating no non-zero elements in that row. The desired type triplet of the matrix `<T,I,O>` needs to be explicitly specified when loading the matrix data from reading a file.

Default types for "I" and "O" is "size_t". But "T" type must be mandatorily specified. While loading the matrix data, it will consider number of columns as the maximum value of the column index read.

For example, considering "./data" is a text file having the sparse data to be loaded, then

```
auto m1 = make_crs_matrix_load<int>("./data");
auto m2 = make_crs_matrix_load<float>("./data");
```

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

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

`crs_matrix<T,I,O>` make_crs_matrix_load(filename, num_col)

Parameters
filename: A string object containing the name of the text file having the data to be loaded.
num_col: A size_t attribute specifying the number of columns in the sparse matrix to be loaded.

Purpose
This function serves the same purpose as explained in above data loading function. But since it also accepts the number of columns information, it sets the loaded matrix column number with the given value (without computing the maximum column index as in previous case). Thus it expects, user will pass a valid column number for the loaded sparse matrix.

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

`crs_matrix<T,I,O>` make_crs_matrix_loadbinary(dirname)

Parameters
dirname: A string object containing the name of the directory having the data to be loaded. It expects four 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) and
- "off" (containing binary offset values)

Purpose
This function loads the little-endian binary data from the specified directory and creates a `crs_matrix<T,I,O>` 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 two types "I" and "O" would be size_t as default types.

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

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

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

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

`crs_matrix<T,I,O>` **make_crs_matrix_loadcoo(file,zero_origin)**

Parameters
file: A string object containing the name of the file having the COO data to be loaded.
zero_origin: A boolean attribute to indicate whether to consider 0-based indices while loading the COO data from file.

Purpose
This function loads the text data from the specified file and creates a `crs_matrix<T,I,O>` object filling the data loaded.

The input file for the sparse data should be in the below COO format:

```
1 1 2.0
1 3 2.0
2 2 5.0
3 1 3.0
3 3 4.0
3 6 3.0
4 3 2.0
4 4 5.0
```

Where each row in the given file represents a triplet like `<row-index col-index value>`. The indices are 1-based by default. This file can be loaded as 0-based index, if "zero_origin" parameter is passed as "true" while loading the file. The desired triplet type of the matrix `<T,I,O>` needs to be explicitly specified when loading the matrix data from reading a file.

Default types for "I" and "O" is "size_t". But "T" type must be mandatorily specified. While loading the matrix data, it will consider number of columns as the maximum value of the column index read.

For example, considering "./data" is a text file having the COO data to be loaded, then

```
auto m1 = make_crs_matrix_loadcoo<int>("./data");
auto m2 = make_crs_matrix_loadcoo<float>("./data");
```

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

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

std::ostream& `operator<<`**(str, mat)**

Parameters
str: A std::ostream& object representing the output stream buffer.
mat: An object of the type `crs_matrix<T,I,O>` containing the matrix to be handled.

Purpose
This function writes the contents of the sparse matrix in "index:value" format
in the given output stream. Thus a crs matrix can simply be printed on the user terminal as "std::cout << mat", where "mat" is the input matrix.

Return Value
On success, it returns a reference to the output stream.

`crs_matrix<T,I,O>` make_crs_matrix_scatter (mat)

Parameters
mat: An object of the type `crs_matrix_local<T,I,O>` to be scattered among worker nodes.

Purpose
This function accepts a `crs_matrix_local<T,I,O>` object and scatters the same among participating worker nodes in order to create a `crs_matrix<T,I,O>`.

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

SEE ALSO

crs_matrix_local