frovedis::colmajor_matrix<T>

NAME

`frovedis::colmajor_matrix<T>` - A distributed two-dimensional dense matrix with elements stored in column-wise order supported by frovedis

SYNOPSIS

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

Constructors

colmajor_matrix ();
colmajor_matrix (`frovedis::node_local<colmajor_matrix_local<T>>`&& data);
colmajor_matrix (const `rowmajor_matrix<T>`& m);

Public Member Functions

void debug_print ();
`rowmajor_matrix<T>` to_rowmajor();
`rowmajor_matrix<T>` moveto_rowmajor();

Public Data Members

`frovedis::node_local<colmajor_matrix_local<T>>` data
size_t num_row
size_t num_col

DESCRIPTION

`colmajor_matrix<T>` is a template based two-dimensional dense matrix with elements stored in column-major order and distributed among the participating worker nodes in row-wise.

A `colmajor_matrix<T>` contains public member "data" of the type `node_local<colmajor_matrix_local<T>>`. The actual distributed matrices are contained in all the worker nodes locally, thus named as `colmajor_matrix_local<T>` (see manual of colmajor_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 matrix.

```
template <class T>
struct colmajor_matrix {
  frovedis::node_local<colmajor_matrix_local<T>> data; // local matrix information
  size_t num_row;  // number of rows in global matrix
  size_t num_col;  // number of columns in global matrix
};
```

For example, if the below column-major matrix with 4 rows and 4 columns is distributed over two worker nodes, then the distribution can be shown as:

```
1 5 8 4
2 6 7 3
3 7 6 2
4 8 5 1
```

```
master                      worker0                        worker1
-----                       -----                          -----
colmajor_matrix<int>        -> colmajor_matrix             -> colmajor_matrix
                                        _local<int>                    _local<int>
   *data: node_local<             val: vector<int>               val: vector<int>
         colmajor_matrix              ({1,5,8,4,                     ({3,7,6,2,
         _local<int>>                   2,6,7,3})                      4,8,5,1})
    num_row: size_t (4)           local_num_row: size_t (2)      local_num_row: size_t (2)
    num_col: size_t (4)           local_num_col: size_t (2)      local_num_col: size_t (2)
```

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

A distributed colmajor_matrix can be created from a distributed rowmajor_matrix object and it can be converted back to the rowmajor_matrix object. Thus loading from file, saving into file etc. interfaces are not provided for colmajor_matrix structure. User may like to perform the conversion from/to rowmajor_matrix structure for the same.

Constructor Documentation

colmajor_matrix ()

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

colmajor_matrix(const `rowmajor_matrix<T>`& m)

It accepts a distributed `rowmajor_matrix<T>` object with elements stored in row-major order and constructs an equivalent distributed colmajor storage of same number of rows and columns. Input row-major storage remains unchanged.

colmajor_matrix (`frovedis::node_local<colmajor_matrix_local<T>>`&& data)

This is the parameterized constructor which accepts an rvalue of the type `node_local<colmajor_matrix_local<T>>` and *moves* the contents to the created colmajor matrix.

In general, this constructor is used internally by some other functions. But user may need this constructor while constructing their own colmajor matrix using the return value of some function (returning a

colmajor_matrix_local<T>) called using "frovedis::node_local::map" (thus returned value would be an object of type node_local<colmajor_matrix_local<T>).

For example,

```
// --- a sample functor definition ---
struct foo {
  foo() {}
  foo(int r, int c): nrow(r), ncol(c) {}
  colmajor_matrix_local<int> operator()(std::vector<int>& v) {
    colmajor_matrix_local<int> ret;
    ret.val.swap(v);
    ret.local_num_row = nrow;
    ret.local_num_col = ncol;
    return ret;
  }
  size_t nrow, ncol;
  SERIALIZE(nrow, ncol)
};

size_t sum(size_t x, size_t y) { return x + y; }
size_t get_nrows(colmajor_matrix_local<int>& m) { return m.local_num_row; }
size_t get_ncols(colmajor_matrix_local<int>& m) { return m.local_num_col; }

std::vector<int> v = {1,3,5,7,2,4,6,8}; // 4X2 col-major storage
auto bv = broadcast(v);
// demo of such a constructor call
colmajor_matrix<int> m(bv.map<colmajor_matrix_local<int>>(foo(4,2))); // m: 8x2 matrix
// getting total number of rows in the global matrix
m.num_row = m.data.map(get_nrows).reduce(sum); // 4+4 = 8
m.num_col = m.data.map(get_ncols).get(0);      // 2
```

The above program will perform the below tasks in order

- broadcast a vector containing sample elements of a 4x2 colmajor_matrix_local.
- local colmajor matrices will be created in worker nodes when the functor would be called.
- "bv.map<colmajor_matrix_local<int>>(foo(4,2))" will return a node_local<colmajor_matrix_local<int> object.
- the constructor call will be made for colmajor_matrix passing the above rvalue node_local object.
- total number of rows will be set by summing local_num_row of all worker matrices.
- total number of columns will be set as per the number of columns in the worker0 matrix (from any worker will be fine).

Public Member Function Documentation

void debug_print ()

It prints the contents and other information of the local matrices node-by-node on the user terminal. It is mainly useful for debugging purpose.

For example, if there are two worker nodes, then

```
std::vector<int> v = {1,2,3,4,5,6,7,8}; // 4x2 col-major storage
rowmajor_matrix_local<int> m;
m.val.swap(v);
m.set_local_num(nrow,ncol);
// scattering local matrix to create the distributed rowmajor matrix
auto rm = make_rowmajor_matrix_scatter(m));
colmajor_matrix<int> cm(rm); // rowmajor_matrix => colmajor_matrix
cm.debug_print();
```

The above program will output (order of display might differ):

```
node = 0, local_num_row = 2, local_num_col = 2, val = 1 3 2 4
node = 1, local_num_row = 2, local_num_col = 2, val = 5 7 6 8
```

`rowmajor_matrix<T>` to_rowmajor();

It converts the colmajor storage of the target distributed matrix to a distributed rowmajor storage and returns the output `rowmajor_matrix<T>` after successful conversion. The target colmajor storage remains unchanged after the conversion.

`rowmajor_matrix<T>` moveto_rowmajor();

If the target distributed column major matrix has only a single column, then rowmajor storage and column major storage both will be the same. Thus instead of any conversion overhead, elements in target matrix can simply be moved while creating the rowmajor_matrix object. It is faster and recommended, only when the target matrix is no longer be needed in a user program.

Public Data Member Documentation

data

An instance of `node_local<colmajor_matrix_local<T>>` 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

colmajor_matrix_local, rowmajor_matrix, blockcyclic_matrix