frovedis::rowmajor_matrix<T>

NAME

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

SYNOPSIS

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

Constructors

rowmajor_matrix ();
rowmajor_matrix (`frovedis::node_local<rowmajor_matrix_local<T>>`&& data);

Public Member Functions

void set_num (size_t nrow, size_t ncol);
void save (const std::string& file);
void savebinary (const std::string& dir);
void debug_print ();
`rowmajor_matrix<T>` transpose () const;
`rowmajor_matrix_local<T>` gather();
`rowmajor_matrix<T>`& align_as(const `std::vector<size_t>`& sz);
`template <class U> rowmajor_matrix<T>`& align_to(`rowmajor_matrix<U>`& m);
`rowmajor_matrix<T>`& align_block();

Public Data Members

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

DESCRIPTION

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

A `rowmajor_matrix<T>` contains public member "data" of the type `node_local<rowmajor_matrix_local<T>>`. The actual distributed matrices are contained in all the worker nodes locally, thus named as `rowmajor_matrix_local<T>` (see manual of rowmajor_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 rowmajor_matrix {
  frovedis::node_local<rowmajor_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 row-major matrix with 4 rows and 4 columns is distributed over two worker nodes, then the distribution can be shown as:

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

master                      worker0                        worker1
------                      -------                        -------
rowmajor_matrix<int>        -> rowmajor_matrix             -> rowmajor_matrix
                                       _local<int>                    _local<int>
   *data: node_local<             val: vector<int>               val: vector<int>
         rowmajor_matrix               ({1,2,3,4,                     ({8,7,6,5,
         _local<int>>                    5,6,7,8})                      4,3,2,1})
    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<rowmajor_matrix_local<int>>` object "data" is simply a (*)handle of the (->)local matrices at worker nodes.

Constructor Documentation

rowmajor_matrix ()

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

rowmajor_matrix (frovedis::node_local<rowmajor_matrix_local<T>>&& data)

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

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

For example,

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

std::vector<int> v = {1,2,3,4,5,6,7,8}; // 4x2 rowmajor storage
auto bv = broadcast(v);
// demo of such a constructor call
rowmajor_matrix<int> m(bv.map<rowmajor_matrix_local<int>>(foo(4,2))); //m: 8x2
// 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 rowmajor_matrix_local.
- local rowmajor matrices will be created in worker nodes when the functor would be called.
- "`bv.map<rowmajor_matrix_local<int>>(foo(4,2))`" will return a `node_local<rowmajor_matrix_local<int>` object.
- the constructor call will be made for rowmajor_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 set_num (size_t nrow, size_t ncol)

It sets the global matrix information related to number of rows and number of columns as specified by the user. It assumes the user will provide the valid matrix dimension according to the number of elements in it. Thus no validity check is performed on the provided dimension values.

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};
rowmajor_matrix_local<int> m;
m.val.swap(v);
m.set_local_num(4,2); // m: 4x2 rowmajor matrix
// it scatters a dense rowmajor matrix
// in order to create the distributed rowmajor matrix
auto gm = make_rowmajor_matrix_scatter(m);
gm.debug_print();
```

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

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

`rowmajor_matrix<T>` transpose ()

It constructs the transposed matrix of the source distributed rowmajor_matrix object and returns the same.

`rowmajor_matrix_local<T>` gather ()

It gathers the local matrices from the worker nodes and constructs the original dense matrix at master node.

On success, it returns the constructed local matrix of the type `rowmajor_matrix_local<T>`, where T is the type of the distributed matrix.

void save (const std::string& file)

It writes the elements of the global rowmajor matrix to the specified file in rowmajor format with text data.

void savebinary (const std::string& dir)

It writes the elements of the global rowmajor matrix to the specified directory in rowmajor format with binary data.

The output directory will contain two files, named "nums" and "val" respectively. "nums" is a text file containing the number of rows and number of columns information in first two lines of the file. And "val" is a binary file containing the matrix elements stored in little-endian form.

`rowmajor_matrix<T>&` align_as(const `std::vector<size_t>&` sz)

This function can be used to re-align the distribution of an existing rowmajor matrix. It accepts an `std::vector<size_t>` containing the desired distribution, i.e., number of rows to be distributed per worker node.

The function will work well, only when below conditions are true:

- the size of the input vector must match with the number of worker nodes.
- the total number of rows in the source rowmajor matrix (to be re-aligned) must match with the sum-total value provided in the input vector.

On success, it will return a reference to the re-aligned rowmajor_matrix.

For example, if there are two worker nodes, then

```
std::vector<int> v = {1,2,3,4,5,6,7,8};
rowmajor_matrix_local<int> m;
m.val.swap(v);
m.set_local_num(4,2); // m: 4x2 matrix
auto gm = make_rowmajor_matrix_scatter(m);
gm.debug_print();
std::vector<size_t> new_sizes = {3,1};
gm.align_as(new_sizes); // Ok
gm.debug_print();
```

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

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

But the below cases will lead to a runtime error:

```
new_sizes = {2,1};
gm.align_as(new_sizes); // error, sumtotal (2+1=3) != num_row (4)
new_sizes = {2,1,1};
gm.align_as(new_sizes); // error, input vector size (3) != worker size (2)
```

rowmajor_matrix<T>& align_to(rowmajor_matrix<U>& m)

This function is used to re-align an existing rowmajor matrix, "m1" according to the distribution alignment of another existing rowmajor_matrix, "m2". The type of "m1" and "m2" can differ, but their total number of row count must be same in order to perform the re-alignment.

On success, it will return a reference to the re-aligned matrix "m1".

For example,

```
std::vector<int> v1 = {1,2,3,4};
std::vector<int> v2 = {1,2,3,4,5,6,7,8};
std::vector<double> v3 = {1,2,3,4,5,6,7,8};

rowmajor_matrix_local<int> m1, m2;
rowmajor_matrix_local<double> m3

m1.val.swap(v1);
m1.set_local_num(2,2); // m1: 2x2 matrix (type: int)
m2.val.swap(v2);
m2.set_local_num(4,2); // m2: 4x2 matrix (type: int)
m3.val.swap(v3);
m3.set_local_num(4,2); // m3: 4x2 matrix (type: double)

auto gm1 = make_rowmajor_matrix_scatter(m1);
```

```
auto gm2 = make_rowmajor_matrix_scatter(m2);
auto gm3 = make_rowmajor_matrix_scatter(m3);

gm2.align_to(gm3); // ok, type differs, but total num of rows matches
gm2.align_to(gm1); // error, type matches, but total num of rows differs
```

rowmajor_matrix<T>& align_block()

This function is used to re-align an existing rowmajor matrix according to the frovedis default distribution block alignment.

If total number of rows in the target matrix is 5 and the number of worker nodes is 2, then frovedis computes the number of rows to be distributed per worker node according to the formula "ceil(total_num_rows/num_of_worker)", which would be evaluated as 3 in this case [ceil(5/2)]. So worker0 will contain the first 3 rows and worker1 will contain next 2 rows.

On success, it will return the reference to the re-aligned rowmajor matrix. If the source matrix is already distributed according to frovedis default block alignment, then no operation will be performed. Simply the reference to the target rowmajor matrix would be returned.

For example, if there are two worker nodes, then

```
std::vector<int> v = {1,2,3,4,5,6,7,8,9,10};
rowmajor_matrix_local<int> m;
m.val.swap(v);
m.set_local_num(5,2); // m: 5x2 rowmajor matrix
auto gm = make_rowmajor_matrix_scatter(m);
gm.debug_print(); // original distribution
std::vector<int> new_sizes = {4,1};
gm.align_as(new_sizes);
gm.debug_print(); // 4,1 distribution
gm.align_block();
gm.debug_print(); // default block distribution (as in original -> 3,2)
```

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

```
node = 0, local_num_row = 3, local_num_col = 2, val = 1 2 3 4 5 6
node = 1, local_num_row = 2, local_num_col = 2, val = 7 8 9 10
node = 0, local_num_row = 4, local_num_col = 2, val = 1 2 3 4 5 6 7 8
node = 1, local_num_row = 1, local_num_col = 2, val = 9 10
node = 0, local_num_row = 3, local_num_col = 2, val = 1 2 3 4 5 6
node = 1, local_num_row = 2, local_num_col = 2, val = 7 8 9 10
```

Public Data Member Documentation

data

An instance of `node_local<rowmajor_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.

Public Global Function Documentation

`rowmajor_matrix<T>` make_rowmajor_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 the distributed `rowmajor_matrix<T>` object filling the data loaded. It assumes that there is no empty lines in the input file. The desired type of the matrix (e.g., int, float, double etc.) is to be explicitly specified when loading the matrix data from reading a file.

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

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

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

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

`rowmajor_matrix<T>` make_rowmajor_matrix_loadbinary(dirname)

Parameters
dirname: A string object containing the name of the directory having the data to be loaded. It expects two files "nums" and "val" to be presented in the input directory, where "nums" is the text file containing number of rows and number of columns information (new line separated) and "val" is the little-endian binary data to be loaded.

Purpose
This function loads the binary data from the specified directory and creates the distributed `rowmajor_matrix<T>` object filling the data loaded. The desired type of the matrix (e.g., int, float, double tec.) is to be explicitly specified when loading the matrix data from reading a file.

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

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

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

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

`rowmajor_matrix<T>` make_rowmajor_matrix_scatter(mat)

Parameters
mat: A const& of a `rowmajor_matrix_local<T>` object containing the data to be scattered among worker nodes.

Purpose
This function accepts a `rowmajor_matrix_local<T>` object and row-wise scatters the elements to the participating worker nodes to create a distributed `rowmajor_matrix<T>` object. During the scatter operation, it follows frovedis default distribution block alignment (see rowmajor_matrix::as_block() for details).

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

`rowmajor_matrix<T>` make_rowmajor_matrix_scatter(mat,dst)

Parameters
mat: A const& of a `rowmajor_matrix_local<T>` object containing the data to be scattered among worker nodes.
dst: A vector of "size_t" elements containing the number of rows to be scattered per worker node.

Purpose
This function accepts a `rowmajor_matrix_local<T>` object and row-wise scatters the elements to the participating worker nodes according to the specified number of rows per worker in the input "dst" vector to create a distributed `rowmajor_matrix<T>` object.

This function will work well, only when below conditions are true:

- the size of the input vector must match with the number of worker nodes.
- the total number of rows in the source local matrix, "mat" (to be scattered) must match with the sum-total value provided in the input vector, "dst".

For example, if there are two worker nodes, then

```
std::vector<int> v = {1,2,3,4,5,6,7,8};
rowmajor_matrix_local<int> m;
m.val.swap(v);
m.set_local_num(4,2); // m: 4x2 matrix
auto gm1 = make_rowmajor_matrix_scatter(m); //ok, an usual scatter operation
gm1.debug_print();
std::vector<size_t> new_sizes = {3,1};
auto gm2 = make_rowmajor_matrix_scatter(m,new_sizes); //ok, nrow == sumtotal
gm2.debug_print();
```

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

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

But the below cases will lead to a runtime error:

```
new_sizes = {2,1};
auto gm3 = make_rowmajor_matrix_scatter(m,
           new_sizes); //error, nrow (4) != sumtotal (2+1=3)
new_sizes = {2,1,1};
auto gm4 = make_rowmajor_matrix_scatter(m,
           new_sizes); //error, input vector size (3) != worker size (2)
```

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

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

Parameters
str: A std::ostream& object representing the output stream buffer.
mat: A const& object of the type `rowmajor_matrix<T>` containing the matrix to be handled.

Purpose
This function writes the contents of the matrix in 2D row-major matrix form in the given output stream. Thus a distributed rowmajor matrix can simply be printed on the user terminal as "std::cout << mat", where "mat" is the input matrix. In this case, it first gathers the local matrices from the worker nodes and then writes them one-by-one on the output stream.

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

SEE ALSO

rowmajor_matrix_local, colmajor_matrix, blockcyclic_matrix