frovedis::blockcyclic_matrix<T>

NAME

`frovedis::blockcyclic_matrix<T>` - two-dimensional blockcyclic distribution of a dense matrix over a MxN process grid (MxN = number of parallel processes)

SYNOPSIS

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

Constructors

blockcyclic_matrix ()
blockcyclic_matrix (size_t nrow, size_t ncol, size_t type)
blockcyclic_matrix (`frovedis::node_local<blockcyclic_matrix_local<T>>`&& data)
blockcyclic_matrix (const `blockcyclic_matrix<T>`& m)
blockcyclic_matrix (`blockcyclic_matrix<T>`&& m)
blockcyclic_matrix (const `colmajor_matrix<T>`& m, size_t type=2)
blockcyclic_matrix (`colmajor_matrix<T>`&& m, size_t type=2)
blockcyclic_matrix (const `std::vector<T>`& v, size_t type=1)
blockcyclic_matrix (`std::vector<T>`&& v, size_t type=1)

Overloaded Operators

`blockcyclic_matrix<T>`& operator= (const `blockcyclic_matrix<T>`& m)
`blockcyclic_matrix<T>`& operator= (`blockcyclic_matrix<T>`&& m)

Public Member Functions

`std::vector<T>` to_vector ()
`std::vector<T>` moveto_vector ()
`colmajor_matrix<T>` to_colmajor ()
`colmajor_matrix<T>` moveto_colmajor ()
`rowmajor_matrix<T>` to_rowmajor ()
`blockcyclic_matrix<T>` transpose ()
void set_num (size_t nrow, size_t ncol, size_t type=2)
void save (const std::string& file)
void savebinary (const std::string& dir)
void debug_print ()
size_t get_nrows ()
size_t get_ncols ()

Public Data Members

`frovedis::node_local<blockcyclic_matrix_local<T>>` data
size_t num_row
size_t num_col
size_t type

DESCRIPTION

`frovedis::blockcyclic_matrix<T>` is a special type of `frovedis::colmajor_matrix<T>` distributed in two-dimensional blockcyclic manner over a MxN process grid (MxN = number of parallel processes). This is a template based dense matrix with type **"T"** which can be either **"float"** or **"double"** (at this moment). Specifying other types as template argument may lead to invalid results. Currently frovedis only supports creation of two types of blockcyclic matrices.

type-1 blockcyclic-matrix:
In case of type-1 blockcyclic-matrix, the global matrix is distributed over a Nx1 process grid, where N is the number of parallel processes. This type of distribution is prefered while distributing a column-vector (a matrix with many rows and 1 column), in order to achieve a better load balance.

type-2 blockcyclic-matrix:
In case of type-2 blockcyclic-matrix, the global matrix is distributed over a MxN process grid, where M = sqrt(number of parallel processes) and N = (number of parallel processes / M).

The specifications of the block size (MBxNB, where MB is the number of rows in a block and NB is the number of columns in a block) are decided by the algorithm depending upon the global matrix dimensions and number of parallel processes. Some constructors also support user defined block size.

A `blockcyclic_matrix<T>` contains public member "data" of the type `node_local<blockcyclic_matrix_local<T>>`. The actual distributed matrices are contained in all the worker nodes locally, thus named as `blockcyclic_matrix_local<T>`. Each of these local matrices have the below structure:

```
template <class T>
struct blockcyclic_matrix_local {
  std::vector<T> val;     // the actual local distributed matrix
  std::vector<int> descA; // the distributed information mapping array
  size_t local_num_row;   // number of rows in local matrix
  size_t local_num_col;   // number of columns in local matrix
  size_t type;            // type of the local matrix (Nx1 or MxN)
};
```

The global version of the matrix at master node contains only three information,
the reference to these local matrices at worker nodes, the dimensions of the global matrix, i.e., number of its rows and columns and the type of the distributed matrix.

```
template <class T>
struct blockcyclic_matrix {
  frovedis::node_local<blockcyclic_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
  size_t type;     // type of the blockcyclic-matrix (Nx1 or MxN)
};
```

Constructor Documentation

blockcyclic_matrix ()

This is the default constructor which creates an empty blockcyclic matrix with num_row = num_col = 0 and type = 2.

blockcyclic_matrix (size_t nrow, size_t ncol, size_t t=2)

This is the parameterized constructor which creates an empty blockcyclic matrix of the given dimension and type (default type=2).

blockcyclic_matrix (`frovedis::node_local<blockcyclic_matrix_local<T>>&&` data)

This is the parameterized constructor which accepts an rvalue of the type `node_local<blockcyclic_matrix_local<T>>` and *moves* the contents to the created blockcyclic matrix. In general, this constructor is used internally by some other functions. But user may need this constructor while constructing their own blockcyclic matrix using the return value of some function (returning a `blockcyclic_matrix_local<T>`) called using "frovedis::node_local::map" (thus returned value would be an object of type `node_local<blockcyclic_matrix_local<T>`).

blockcyclic_matrix (const `blockcyclic_matrix<T>&` m)

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

blockcyclic_matrix (`blockcyclic_matrix<T>&&` 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.

blockcyclic_matrix (const `colmajor_matrix<T>&` m, size_t type=2)

This is a special constructor for implicit conversion. It converts an input colmajor matrix to a blockcyclic matrix of the same global dimensions. The input matrix is unchanged after the conversion. Default type of the created blockcyclic matrix is 2 (desired type can be specified in second argument).

blockcyclic_matrix (`colmajor_matrix<T>&&` m, size_t type=2)

This is a special constructor for implicit conversion. It converts an input colmajor matrix to a blockcyclic matrix of the same global dimensions. But during the conversion the memory buffer of input rvalue matrix is reused, thus the input colmajor matrix becomes invalid after this conversion. Default type of the created blockcyclic matrix is 2 (desired type can be specified in second argument).

blockcyclic_matrix (const `std::vector<T>&` v, size_t type=1)

This is a special constructor for implicit conversion. It converts an input lvalue `std::vector<T>` to `blockcyclic_matrix<T>` with global dimensions Nx1, where N = size of the input vector. The input vector is unchanged after the conversion. Default type of the created blockcyclic matrix is 1 to support load balancing (desired type can be specified in second argument).

blockcyclic_matrix (std::vector<T>&& v, size_t type=1)

This is a special constructor for implicit conversion. It converts an input rvalue `std::vector<T>` to `blockcyclic_matrix<T>` with global dimensions Nx1, where N = size of the input vector. But during the conversion, the memory buffer of the input rvalue vector is reused, thus it becomes invalid after this conversion. Default type of the created blockcyclic matrix is 1 to support load balancing (desired type can be specified in second argument).

Overloaded Operator Documentation

blockcyclic_matrix<T>& operator= (const blockcyclic_matrix<T>& m)

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

blockcyclic_matrix<T>& operator= (blockcyclic_matrix<T>&& m)

Instead of copying, it moves the contents of the input rvalue blockcyclic 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

std::vector<T> to_vector ()

If num_col = 1, it converts the blockcyclic matrix to `std::vector<T>` and returns the same, else it throws an exception. The blockcyclic matrix is unchanged.

std::vector<T> moveto_vector ()

If num_col = 1 and type = 1, it converts the blockcyclic matrix to `std::vector<T>` and returns the same, else it throws an exception. Due to move operation, input matrix becomes invalid after the conversion.

colmajor_matrix<T> to_colmajor ()

It converts the blockcyclic matrix to colmajor matrix and returns the same. Input matrix is unchanged.

colmajor_matrix<T> moveto_colmajor ()

Only when type = 1, it converts the blockcyclic matrix to colmajor matrix and returns the same, else it throws an exception. During the conversion it reuses the memory buffer of the blockcyclic matrix, thus it would become invalid.

rowmajor_matrix<T> to_rowmajor ()

It converts the blockcyclic matrix to rowmajor matrix and retuns the same. The blockcyclic matrix is unchanged.

`blockcyclic_matrix<T>` transpose ()

It returns the transposed blockcyclic matrix of the source matrix object.

void set_num (size_t nrow, size_t ncol, size_t type=2)

It sets the global matrix information as specified. Default type is considered as 2, if *type* value is not provided.

void save (const std::string& file)

It writes the blockcyclic matrix to the specified file in rowmajor format with text data.

void savebinary (const std::string& dir)

It writes the blockcyclic matrix to the specified directory in rowmajor format with binary data (in little-endian form).

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.

size_t get_nrows ()

It returns the global number of rows in the source blockcyclic matrix object.

size_t get_ncols ()

It returns theglobal number of columns in the source blockcyclic matrix object.

Public Data Member Documentation

data

An instance of `node_local<blockcyclic_matrix_local<T>>` which contains the references to the local matrices in the worker nodes.

num_row

A size_t attribute to contain the number of rows in the global blockcyclic matrix.

num_col

A size_t attribute to contain the number of columns in the global blockcyclic matrix.

Public Global Function Documentation

make_blockcyclic_matrix_loadbinary(dirname, type, MB, NB)

Parameters
dirname: A string object containing the name of the directory having binary data to be loaded.
type: A size_t attribute containing the desired type of the matrix to be created (optional, default=2).
MB: A size_t attribute containing the desired number of rows in a block (optional, default=0).
NB: A size_t attribute containing the desired number of columns in a block (optional, default=0).

Purpose
This function loads the (little-endian) binary data from the specified directory and creates a blockcyclic matrix of default type = 2 and algorithm decided block size (if not defined by the user, i.e., MB=NB=0). The required type and block size can be specified.

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

make_blockcyclic_matrix_load(fname, type, MB, NB)

Parameters
fname: A string object containing the name of the data file.
type: A size_t attribute containing the desired type of the matrix to be created (optional, default=2).
MB: A size_t attribute containing the desired number of rows in a block (optional, default=0).
NB: A size_t attribute containing the desired number of columns in a block (optional, default=0).

Purpose
This function loads the data from the specified text file and creates a blockcyclic matrix of default type = 2 and algorithm decided block size (if not defined by the user, i.e., MB=NB=0). The required type and block size can be specified.

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

make_blockcyclic_matrix_scatter(rmat, type, MB, NB)

Parameters
rmat: An object of the type `rowmajor_matrix_local<T>` containing the data to be scattered.
type: A size_t attribute containing the desired type of the matrix to be created (optional, default=2).
MB: A size_t attribute containing the desired number of rows in a block (optional, default=0).
NB: A size_t attribute containing the desired number of columns in a block (optional, default=0).

Purpose
This function scatters an input `frovedis::rowmajor_matrix_local<T>` as per the active number of parallel processes and from the scattered data it creates a blockcyclic matrix of default type = 2 and algorithm decided block size (if not defined by the user, i.e., MB=NB=0). The required type and block size can be specified.

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

vec_to_bcm(vec, type, MB, NB)

Parameters
vec: An object of the type `std::vector<T>` containing the data values.
type: A size_t attribute containing the desired type of the matrix to be created (optional, default=1).
MB: A size_t attribute containing the desired number of rows in a block (optional, default=0).
NB: A size_t attribute containing the desired number of columns in a block (optional, default=0).

Purpose
This function scatters the input vector as per the active number of parallel processes and from the scattered data it creates a blockcyclic matrix of default type = 1 (for a better load balancing) and algorithm decided block size (if not defined by the user, i.e., MB=NB=0). The required type and block size can be specified. If the input vector is an *lvalue*, it copies the data before scattering. But if the vector is an *rvalue*, it ignores copying the data.

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

SEE ALSO

colmajor_matrix, rowmajor_matrix, sliced_blockcyclic_matrix, sliced_blockcyclic_vector