MPI FUNCTIONS

MPI_INITIALIZATION/SETUP

MPI_Init( int* argc, char*** argv )
Input parameter:
   argc - Pointer to the number of command line parameters
   argv - Pointer to the array of command line parameters

MPI_Comm_size( MPI_Comm comm, int* size )
Input parameter:
   comm - Handle for a MPI communicator
Output parameter:
   size - Number of processes in the comm group

MPI_Comm_rank( MPI_Comm comm, int* rank )
Input parameter:
   comm - Handle for a MPI communicator
Output parameter:
   rank - Rank of the calling process in the comm communicator

MPI_FINALIZATION

MPI_Finalize()
No parameters

MPI_BLOCKING SEND/RECEIVE FUNCTIONS

MPI_Send( void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm )
Input parameter:
   buf - Start address of the send buffer
   count - Number of elements in the send buffer
   datatype - Type of the elements of the send buffer
   dest - Rank of the receiving process in the communicator comm
   tag - Message tag for the distinction of messages
   comm - Handle to the communicator

MPI_Recv( void* buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status* status )
Input parameter:
   count - Number of the elements for the receive buffer
   datatype - Type of the elements in the receive buffer
   source - Rank of the sending process in the communicator comm
   tag - Message tag for the distinction of messages
   comm - Handle to the communicator
Output parameter:
   buf - Start address of the receive buffer
   status - Status object for the receive operation, which holds information about the source and tag

MPI_NON-BLOCKING SEND/RECEIVE FUNCTIONS

MPI_Irecv( void* buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Request* request )
Input parameter:
   count - Number of the elements for the receive buffer
   datatype - Type of the elements in the receive buffer
   source - Rank of the sending process in the communicator comm
   tag - Message tag for the distinction of messages
   comm - Handle to the communicator
Output parameter:
   buf - Start address of the receive buffer
   request - Request object for information about the status of the receive operation

MPI_Isend( void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm, MPI_Request* request )
Input parameter:
   buf - Start address of the send buffer
   count - Number of elements in the send buffer
   datatype - Type of the elements of the send buffer
   dest - Rank of the receiving process in the communicator comm
   tag - Message tag for the distinction of messages
   comm - Handle to the communicator
Output parameter:
   request - Request object for information about the status of the send operation
MPI SYNCHRONIZATION

MPI_Waitall( int count, MPI_Request* array_of_requests, MPI_Status* array_of_status )

Input parameter:
  count - Size of the request and status arrays
  array_of_requests - Array for the communication requests

Output parameter:
  array_of_status - Array for the status informations for the request array. For receive operations, information about the sender and tag

MPI TYPE DERIVATION

MPI_Type_vector( int count, int blocklen, int stride, MPI_Datatype oldtype, MPI_Datatype* newtype )

Input parameter:
  count - Number of blocks
  blocklen - Number of elements in each block
  stride - Number of elements between the start of two blocks
  oldtype - Handle to the old data type

Output parameter:
  newtype - Handle to the new data type

\[
\begin{array}{cccccccc}
  & 0 & 1 & 2 & N & K \\
M & int & int & int & \ldots & int & \ldots & int \\
1 & int & int & int & \ldots & int & \ldots & int \\
2 & int & int & int & \ldots & int & \ldots & int \\
\end{array}
\]

MPI_Type_vector( M, N, K, MPI_INT, &newtype )

MPI_Type_commit( MPI_Datatype* datatype )

InOut parameter:
  datatype - Handle to the new datatype

MPI_Type_free( MPI_Datatype* datatype )

InOut parameter:
  datatype - Handle to the datatype, which has to be deleted

MPI TOPOLOGY CREATION

MPI_Cart_create( MPI_Comm oldcomm, int ndims, int* dims, int* periods, int reorder, MPI_Comm* comm_cart )

Input parameter:
  oldcomm - Handle to the input communicator
  ndims - Dimension of the Cartesian structure
  dims - Array of size ndims, which gives the number of processes in each dimension
  periods - Array of size ndims, which specifies, if the structure is periodic in each dimension (true = periodic)
  reorder - specifies, if MPI may reassign the ranks of the processes (true) or if the old order in oldcomm has to be preserved

Output parameter:
  comm_cart - Handle to the new communicator with Cartesian structure

MPI_Cart_coords( MPI_Comm comm, int rank, int maxdims, int* coords )

Input parameter:
  comm - Handle to a communicator with Cartesian structure
  rank - Rank for the identification of the process
  maxdims - Number of dimensions in the Cartesian topology

Output parameter:
  coords - Array of size maxdims with the logical coordinates of the process

MPI_Cart_shift( MPI_Comm comm, int dir, int disp, int* rank_source, int* rank_dest )

Input parameter:
  comm - Handle to a communicator with Cartesian structure
  dir - Direction, in which data is transported (0 = up/down, 1 = left/right)
  disp - specifies, how far the data is transported ( &lt; 0 = downwards, &gt; 0 = upwards )

Output parameter:
  rank_source - Rank of the process, from which the calling process is receiving data (rank of the source task)
  rank_dest - Rank of the process, to which the calling process is sending data (rank of the destination task)