Limitations

Some IBM Spectrum™ MPI product features are subject to certain limitations, as explained in this section.

- IBM Spectrum MPI is not ABI compatible with any other MPI implementations such as Open MPI, Platform MPI, or IBM® PE Runtime Edition.
- The IBM Spectrum MPI collectives component (libcollectives) does not support inter-communicators. For inter-communicator collective support, IBM Spectrum MPI relies on Open MPI collective components.
- Dynamic process management (dynamic tasking) is not supported.
- The total byte size is defined as the parameter count multiplied by the size of the related data type.
  - There is a 2 GB restriction for the total byte size in the IBM libcoll collective function call. A libcoll collective function call fails if the total byte size is more than 2 GB. To avoid the 2 GB restriction for the total byte size, you can call the open mpi default algorithm instead of the libcoll collective function by using the `--mca coll ^ibm` option. For the `MPI_Gather` family of collectives the limitation affects the size of the total receive buffer the combined size of the contributions from all the ranks has a 2 GB limitation.
  - There is a 4 GB restriction for the total byte size for all message traffic over PSM and PSM2. If the total byte size is more than 4 GB, the PSM or PSM2 communication call prints an error message and aborts. For the `MPI_Gather` family of collectives the limitation affects the size of the total receive buffer, therefore, the combined size of the contributions from all the ranks has a 4 GB limitation.
- If you run MPI applications with the nvprof profiler (NVIDIA-supplied profiler for applications that use NVIDIA GPUs), the MPI applications do not complete unless you run the **`mpirun -mca mpi_restrict_libs none`** command.
- The PAMI and RDMA one-sided components (OSC) are not thread safe.
- The following data types are not supported by reduction operations `MPI_MINLOC` or `MPI_MAXLOC`:
  - `MPI_DOUBLE_INT`
  - `MPI_LONG_INT`
  - `MPI_SHORT_INT`
  - `MPI_LONG_DOUBLE_INT`
- The `-HCOLL | -FCA` option that enables the Mellanox hcoll library, which is also known as FCA, can hang in the **`MPI_Win_lock`** API.
- The `-MXMC` option is no longer available. Instead of using the `-MXMC` option, you can use the `-MXM` option that supports the PML yalla layer.
- To use GPU device memory in MPI calls for the PAMI point-to-point management layer (PML), you must specify one of the following options when you run the **`mpirun`** command:
  - `-gpu`

    - `-mca pmg_pami_enable_cuda 1`

  You might experience a decrease in application performance if you permanently enable GPU device memory support because PAMI copies data to and from the GPU memory (device) and CPU memory (host).

  You can permanently enable GPU device memory support by adding `pml_pami_enable_cuda = 1` to the `/opt/ibm/spectrum_mpi/openmpi-mca-params.conf` file. You can selectively disable the permanent GPU device memory support by adding `pml_pami_enable_cuda = 0` to the **`mpirun`** command.

  You must enable GPU device memory support if the MPI application allocates device memory and references this memory in MPI calls. For example, an MPI application that uses send or receive buffers that point to device memory. If the MPI application is referencing CUDA Unified Memory in MPI calls, you must also enable GPU device memory support. CUDA Unified Memory is a single memory address space that is accessible from any CPU or GPU in the system. CUDA runtime drivers handle memory page migration to and from the host or device. An MPI application might request Unified Memory by issuing a `cudaMallocManaged()` call. Some compilers might also provide an option to handle any dynamic allocation request, such as malloc (C), new (C++), or ALLOCATE (Fortran), as requested for Unified Memory. For example, the PGI compiler uses the `-ta=tesla:managed` option to map dynamic allocation requests to Unified Memory.

- For x86 servers, the names for the following libraries changed in IBM Spectrum MPI 10.1.1:

  Table 1. Library name changes

Previous library name	**New library name**
libmpi_mpif.so.2	libmpi_ibm_mpifh.so.2
libmpi_usempi.so.2	libmpi_ibm_usempi.so.2

  For backwards comparability, Spectrum MPI included symlinks for applications that are built with previous version of Spectrum MPI to run with Spectrum MPI 10.1.1. If you are linking or compiling with Spectrum MPI and using the MPI wrapper compiler scripts, no changes are required use the new library names. However, if you are manually linking or compiling with Spectrum MPI without the MPI wrapper compiler scripts, you must link your applications to the new library names. You can use the `--showme` option to identify where you are linking or compiling with the previous library names.
- A child process that is forked from an MPI process but does not call the `exec` function (or other variants of the exec function), must not return from the main function or call the `exit()` function (C API). In an MPI application that creates child processes, typically only one of the processes (most often the parent) should terminate by calling the `exit()` function. Other processes should terminate by calling the `_exit()` function. You should have only one process that calls exit handlers and flushes stdio buffers.
- The use of the `MPI_File_set_atomicity()` call is not supported.
- When you are creating n-dimensional topologies by using `MPI_Dims_create()`, the ndims must be greater than 0.
- When you are running Spectrum MPI over TCP, on nodes with a virtual adapter, users must specify the correct adapter to use by using `-netaddr`, because Spectrum MPI does not ignore virbr# named devices.
- Multi node I/O is not supported.
- When you are running with PAMI, adapter affinity is enabled by default. This generally leads to better performance when CPU binding is enabled (as it is by default). This instructs each rank to use the InfiniBand adapter that is physically closest to the core where the rank is bound. However, either of the following circumstances can lead to SEGVs (if using the `-verbsbypass` option) or a hang.
    - If a user runs a job across nodes that have different numbers of InfiniBand adapters per node (for example, if some nodes have two, and other nodes only have one)
    - If a user runs a job across nodes that have one Infiniband adapter on one fabric, and another adapter on different fabric

  In either of these situations, users must disable adapter affinity by specifying the following option with **`mpirun`**.

  ```
  PAMI_IBV_ADAPTER_AFFINITY=0
  ```

When running with PAMI, adapter striping is also enabled by default. However, both adapter affinity and adapter striping must be disabled before a user runs a job across a set of nodes in which one node contains a single link only. In this situation, to disable both adapter affinity and adapter striping, specify the following environment variables with **`mpirun`**.

```
PAMI_ENABLE_STRIPING=0
PAMI_IBV_DEVICE_NAME=mlx5_0:1
PAMI_IBV_ADAPTER_AFFINITY=0
```

- At the time of this release, the MPI standard has some ambiguity about the meaning of `MPI_Comm_get_info()`, `MPI_Win_get_info()`, and `MPI_File_get_info()`. In this release, the `*_get_info()` calls return the current internal value of each info setting, which can differ from values provided in a previous `MPI_Comm_set_info()`, and so on. In the future, the MPI standard is likely to be changed to clarify that the get calls should return the same values that were set. So, the current behavior of `MPI_*_get_info()` is not compliant with the expected clarification of the MPI standard.
- If your switch network topology consists of more than one Infiniband network, IBM Spectrum MPI requires that each network be assigned a unique subnet prefix (also known as the network ID). Jobs might fail to run if disconnected networks are assigned the same network ID.
- IBM Spectrum MPI's use of libevent v2.0.22 might cause a perturbation in the random number stream that is generated by the libc `rand()` and `random()` functions. Applications that rely on behavior should consider using the `rand_r()` and `random_r()` APIs.
- Querying or attempting to write the value of a performance variable by using the MPI Tool information interface (MPI_T) is not supported.
- All Fortran MPI applications require the libgfortran3 runtime library to be installed on each compute node, regardless of which Fortran compiler was used to compile and link the application. This is to satisfy the libmpi_usempi.so dependency on libgfortran.so.3.
- IBM Spectrum MPI fails in `MPI_Init` when both HCOLL (`-HCOLL`/`-hcoll`/`-FCA`/`-fca`) and 16M huge pages (`/usr/lib/libhugetlbfs.so`) are used in the same job.
- If an application is built by using the NVIDIA CUDA Toolkit, the NVIDIA CUDA Toolkit must be installed on the node from which it is launched, as well as each compute node. For Power Systems™ users, the CUDA Toolkit version 8.0 is required. For x86 users, the CUDA Toolkit version 7.5 is required.
- On a node with GPUs, it is recommended that you run the following commands shortly after the node boots and before you run any GPU workload:

  ```
  nvidia-smi -pm 1  # persistence mode
  nvidia-modprobe -u -c=0  # pre-load uvm
  ```

- The Open MPI collectives components that are included with IBM Spectrum MPI do not support GPU buffers. For GPU buffer collective support, you must use libcollectives (the default).
- Support for GPU-accelerated applications is provided only if you are using the IBM Spectrum MPI PAMI backend and the IBM collective library (*libcollectives*). These are the default options for IBM Spectrum MPI, but if you choose to use a different messaging protocol or collective component, note that it will not support GPUs.
- The libcollectives component is the only collective component that is able to support CUDA buffers. As a result, when `-gpu` is specified with the **`mpirun`** command, libcollectives must be the preferred collective component. Therefore, you cannot specify **`mpirun`** `-gpu` with any of the following options:
  - `-mxm`/`-MXM`
  - `-tcp`/`-TCP`
  - `-ibv`/`-IBV`
  - `-ib`/`-IB`
  - `-openib`/`-OPENIB`
  - `-fca`/`-FCA`

    - `-hcoll/-HCOLL`
- The following MPI functions are not CUDA-aware:
    - `MPI_Alltoallw`
    - `MPI_Ialltoallw`
    - `MPI_Ineighbor_allgather`
    - `MPI_Ineighbor_allgatherv`
    - `MPI_Ineighbor_alltoall`
    - `MPI_Ineighbor_alltoallv`
    - `MPI_Ineighbor_alltoallw`
    - `MPI_Neighbor_allgather`
    - `MPI_Neighbor_allgatherv`
    - `MPI_Neighbor_alltoall`
    - `MPI_Neighbor_alltoallv`
    - `MPI_Neighbor_alltoallw`
- IBM Spectrum MPI behavior is undefined when an MPI application passes a GPU device address to an MPI API that is not CUDA aware; the application might trigger a segmentation fault and dump core, or it might give incorrect results.
  Most MPI APIs are CUDA-aware. However, IBM Spectrum MPI does not supply header files or Fortran module files that identify which MPI APIs accept GPU device addresses as arguments. Refer to the preceding restriction for a list of MPI APIs that might not be used with GPU device addresses.

- If you are using the IBM XL Fortran compiler to build CUDA-aware MPI programs that employ the partitioned global address space (PGAS) model, you must use XL compiler version 15.1.5 or later.