MVAPICH 1.2 User and Tuning Guide

MVAPICH TEAM

NETWORK-BASED COMPUTING LABORATORY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
THE OHIO STATE UNIVERSITY

http://mvapich.cse.ohio-state.edu/

Last Revised: February 2, 2012

Contents

1 Overview of the Open-Source MVAPICH Project

InfiniBand is a a high-performance interconnect delivering low latency and high bandwidth. It is also getting widespread acceptance due to its *open standard*.

MVAPICH (pronounced as “em-vah-pich”) is an *open-source* MPI software to exploit the novel features and mechanisms of InfiniBand and other RDMA enabled interconnects to deliver performance and scalability to MPI applications. This software is developed in the Network-Based Computing Laboratory (NBCL), headed by Prof. Dhabaleswar K. (DK) Panda.

Currently, there are two versions of this MPI: MVAPICH with MPI-1 semantics and MVAPICH2 with MPI-2 semantics. This *open-source* MPI software project started in 2001 and a first high-performance implementation was demonstrated at Supercomputing ’02 conference. After that, this software has been steadily gaining acceptance in the HPC and InfiniBand community. As of January 29, 2010, more than 1,050 organizations (National Labs, Universities and Industry) in 55 countries have voluntarily registered as the users of this software at the project site. More than 37,000 downloads of this software have taken place for this software from the OSU’s web site directly. In addition, many InfiniBand and 10GigE/iWARP/RDMAoE vendors, server vendors, systems integrators and Linux distributors have been incorporating MVAPICH/MVAPICH2 into their software stacks and distributing it in an open-source manner. Several InfiniBand systems using MVAPICH/MVAPICH2 have obtained positions in the TOP 500 ranking. The current version of MVAPICH is also being made available with the OpenFabrics Enterprise Distribution (OFED) stack. Both MVAPICH and MVAPICH2 distributions are available under BSD licensing.

More details on MVAPICH/MVAPICH2 software, users list, sample performance numbers on a wide range of platforms and interconnect, a set of OSU benchmarks, related publications, and other InfiniBand-related projects (parallel file systems, storage, data centers) can be obtained from the following URL:

http://mvapich.cse.ohio-state.edu/

This document contains necessary information for MVAPICH users to download, install, test, use, and tune MVAPICH 1.2. As we get feedback from users and take care of bug-fixes, we introduce new patches against our released distribution and also continuously update this document. Thus, we strongly suggest referring to our web page for updates.

2 How to use this User Guide?

This guide is designed to take the user through all the steps involved in configuring, installing, running and tuning MPI applications over InfiniBand using MVAPICH-1.2.

In Section 3 we describe all the features in MVAPICH 1.2. As you read through this section, please note our new features (highlighted as NEW). Some of these features are designed in order to optimize specific type of MPI applications and achieve greater scalability. Section 4 describes in detail the configuration and installation steps. This section enables the user to identify specific compilation flags which can be used to turn some of the features on of off. Usage instructions for MVAPICH are explained in Section 5. In addition to describing how to run simple MPI applications, this section also discusses running MVAPICH with some of the advanced features. Section 6 describes the usage of the OSU Micro Benchmarks (OMB). If you have any problems using MVAPICH, please check Section 7 where we list some of the common problems users face. In Section 8 we suggest some tuning techniques for multi-thousand node clusters using some of our new features. In Section 9, we list important run-time and compile time parameters for the OpenFabrics, QLogic and RDMAoE interfaces, their default values and a small description of each parameter. Finally, Section 10 lists the parameters and tuning options for the OpenFabrics/Gen2-Hybrid device.

3 MVAPICH 1.2 Features

MVAPICH (MPI-1 over InfiniBand) is an MPI-1 implementation based on MPICH and MVICH. MVAPICH 1.2 is available as a single integrated package (with the latest MPICH 1.2.7 and MVICH).

A complete set of features of MVAPICH 1.2 are:

- Single code base with multiple underlying transport interfaces
  - OpenFabrics/Gen2
    - This interface support has the highest performing and most scalable features. Includes support for eXtended Reliable Connection (XRC), SRQ, multi-core-aware shared memory collectives and on-demand connection management, and asynchronous progress for overlap computation and communication
  - OpenFabrics/Gen2-Hybrid
    - This interface is targeted for emerging clusters with multi-thousand cores to deliver best performance and scalability with constant memory footprint for communication contexts
    - Provides capabilities to use the Unreliable Datagram (UD), Reliable Connection (RC) and eXtended Reliable Connection (XRC) transports of InfiniBand.
  - (NEW) OpenFabrics/Gen2-RDMAoE
    - This interface supports the emerging RDMAoE (RDMA over Ethernet) interface for Mellanox ConnectX-EN adapters with 10GigE switches.
  - Shared-Memory only channel
    - This interface support is useful for running MPI jobs on multi-processor systems without using any high-performance network. For example, multi-core servers, desktops, and laptops; and clusters with serial nodes.

  - QLogic InfiniPath
    - This interface provides native support for InfiniPath adapters from QLogic. It provides high-performance point-to-point communication as well as optimized collectives (MPI_Bcast and MPI_Barrier) with k-nomial algorithms while exploiting multi-core architecture.
  - TCP/IP
    - The standard TCP/IP interface (provided by MPICH) to work with a wide range of networks. This interface can also be used with IPoIB support of InfiniBand. However, it will not deliver good performance/scalability as compared to the other three interfaces.

  (Please note that VAPI (single-rail and multi-rail), OpenFabrics/Gen2 (multi-rail) and uDAPL interfaces have been deprecated from the MVAPICH code base starting with MVAPICH 1.1 version. To take advantage of integrated multi-rail support with OpenFabrics/Gen2 and uDAPL support, please use MVAPICH2 code base.)

- Scalable and robust job startup
  - Enhanced and robust mpirun_rsh framework to provide scalable launching on multi-thousand node clusters
    - Running time of 'MPI Hello World' program on 1K cores is around 4 sec and on 32K cores is around 80 sec
    - Available for OpenFabrics/Gen2, OpenFabrics/Gen2-Hybrid and QLogic InfiniPath devices
  - Support for SLURM
    - Available for OpenFabrics/Gen2, OpenFabrics/Gen2-Hybrid and QLogic InfiniPath devices
  - Flexibility for using rsh/ssh-based startup
- Designs for scaling to multi-thousand nodes with highest performance and minimal memory usage by automatically selecting various available InfiniBand transports (using OpenFabrics/Gen2-Hybrid interface)
  - Delivers performance and scalability with near-constant memory footprint for communication contexts
  - Adaptive selection during run-time (based on application and systems characteristics) to switch between RC and UD (or between XRC and UD) transports
  - Zero-copy protocol with UD for large data transfer
  - Multiple buffer organizations with XRC support
  - Shared memory communication between cores within a node
  - Processor affinity and flexible user defined processor mapping

  - Multi-core optimized collectives (MPI_Bcast, MPI_Barrier, MPI_Reduce and MPI_Allreduce)
  - Enhanced MPI_Allgather collective

- Designs for scaling to multi-thousand nodes with highest performance and reduced memory usage (using OpenFabrics/Gen2 interface)
  - eXtended Reliable Communication (XRC) support
  - Message coalescing support to enable reduction of per Queue-pair send queues for reduction in memory requirement on large scale clusters. This design also increases the small message messaging rate significantly.
  - Enhanced coalescing support with varying degree of coalescing
  - Asynchronous and scalable on-demand connection management using native InfiniBand Unreliable Datagram (UD) support. This feature enables InfiniBand connections to be setup dynamically, enhancing the scalability of MVAPICH on clusters of thousands of nodes.
  - Shared Receive Queue with Flow Control. The new design uses significantly less memory for MPI library.
  - Adaptive RDMA Fast Path
  - Lock-free design to provide support for asynchronous progress at both sender and receiver to overlap computation and communication
  - Multi-pathing support leveraging LMC mechanism to avoid hot-spots on large fabrics
  - Multi-port/Multi-HCA support for enabling user processes to bind to different IB ports for balanced communication performance on multi-core platforms with multiple HCAs and/or ports.

- Optimized intra-node communication support by taking advantage of shared-memory communication
  - Multi-core aware scalable shared memory design
  - Efficient support for diskless clusters
  - Bus-based SMP systems
  - NUMA-based SMP systems
  - Processor Affinity
  - Flexible user defined processor affinity for better resource utilization on multi-core systems

- Support for Fault Tolerance

  - Mem-to-mem reliable data transfer (detection of I/O bus error with 32bit CRC and retransmission in case of error) This mode enables MVAPICH to deliver messages reliably in presence of I/O bus errors.
  - Network-level fault tolerance with Automatic Path Migration (APM) for tolerating intermittent network failures over InfiniBand
  - (NEW) Network Fault Resiliency (NFR) for tolerating transient network failures. Using this feature, long running MPI codes can tolerate network failures. InfiniBand HCAs are automatically reset and MPI programs can continue computation without having to restart.

- Single code base for the following platforms (Architecture, OS, compilers, Devices, and InfiniBand/10GigE adapters):
  - Architecture EM64T, Opteron, IA-32 and IBM PPC
  - Operating Systems: Linux and Solaris
  - Compilers: gcc, Intel, PathScale and PGI
  - Devices: OpenFabrics/Gen2, OpenFabrics/Gen2-Hybrid, (NEW)OpenFabrics/Gen2-RDMAoE, shared-memory, QLogic/InfiniPath and TCP/IP
  - InfiniBand adapters (tested with):
    * Mellanox adapters with PCI-X and PCI-Express (SDR and DDR with mem-full and mem-free cards)
    * Mellanox ConnectX (DDR)
    * Mellanox ConnectX (QDR) with PCI-Express Gen2
    * QLogic/InfiniPath (DDR) with PCI-Express Gen2
  - 10GigE adapters (tested with):
    * (NEW) Mellanox ConnectX-EN adapter (DDR)

- Optimized RDMA Write-based scheme for Eager protocol (short message transfer)

- Optimized implementation of Rendezvous protocol (large message transfer) for better computation-communication overlap and progress
  - RDMA Write-based
  - RDMA Read-based
  - RDMA Read with Asynchronous Progress

- Two modes of communication progress
  - Polling
  - Blocking

- Advanced AVL tree-based resource-aware registration cache
    - Memory Hook Support provided by integration with ptmalloc2 library. This provides safe release of memory to the Operating System and is expected to benefit the memory usage of applications that frequently use malloc and free operations.
- High performance and scalable collective communication support
    - Optimized, high-performance collective operations for multi-core platforms: Shared Memory MPI_Bcast, Enhanced MPI_Allgather
    - Shared Memory Collectives (MPI_Allreduce, MPI_Reduce, MPI_Barrier)
    - Tuning and Optimization of various collective algorithms for a wide range of system sizes and network adapter characteristics
- Schemes for minimizing memory resource usage on large scale systems
    - Automatic tuning for small, medium and large clusters
    - Shared Receive Queue support
    - On-Demand Connection management
- Shared library support for existing binary MPI application programs to run
- Shared library support for Solaris
- ROMIO support
    - Optimized, high-performance ADIO driver for Lustre.
        - This MPI-IO support for Lustre in MVAPICH is a contribution from Future Technologies Group, Oak Ridge National Laboratory.
- Enhanced support for TotalView debugger
- Integrated and easy-to-use build script which automatically detects system architecture and InfiniBand adapter types and optimizes MVAPICH for any particular installation
- Tuned thresholds and associated optimizations for
    - different architectures/platforms mentioned above
    - different memory/system bus characteristics
    - different network interfaces (PCI-X, PCI-Express with SDR and DDR and IBM ehca adapter with GX interface)
    - different networks enabled by multiple devices/interfaces
- Incorporates a set of runtime and compile time tunable parameters (at MPI and network layers) for convenient tuning on

  - large scale systems
  - future platforms

The MVAPICH 1.2 package and the project also includes the following provisions:

- Public SVN access of the code base
- A set of micro-benchmarks for carrying out MPI-level performance evaluation after the installation
- Public mvapich-discuss mailing list for mvapich users to
  - ask for help and support from each other and get prompt response
  - enable users and developers to contribute patches and enhancements

4 Installation Instructions

4.1 Download MVAPICH source code

The MVAPICH 1.2 source code package includes the latest MPICH 1.2.7 version and also the required MVICH files from LBNL. Thus, there is no need to download any other files except MVAPICH 1.2 source code.

You can go to the MVAPICH website to obtain the source code.

4.2 Prepare MVAPICH source code

Untar the archive you have downloaded from the web page using the following command. You will have a directory named `mvapich-1.2` after executing this command.

```
$ tar xzf mvapich-1.2.tar.gz
```

4.3 Getting MVAPICH source updates

As we enhance and improve MVAPICH, we update the available source code on our public SVN repository. In order to obtain these updates, please install a SVN client on your machine. The latest MVAPICH sources may be obtained from the "trunk" of the SVN using the following command:

```
$ svn co https://mvapich.cse.ohio-state.edu/svn/mpi/mvapich/trunk
```

The "trunk" may contain newer features and bug fixes. However, it is likely to be lightly tested. If you are interested in obtaining stable and major bug fixes to any release version, you should update your sources from the "branch" of the SVN using the following command:

```
$ svn co https://mvapich.cse.ohio-state.edu/svn/mpi/mvapich/branches/1.
```

MVAPICH 1.2 provides support for six different ADI devices. Namely, Gen2 Single-Rail (`ch_gen2`), Gen2/Hybrid (`ch_hybrid`), Gen2-RDMAoE, Shared memory device (`ch_smp`), and QLogic InfiniPath (`ch_psm`). Additionally, you can also configure MVAPICH over the standard TCP/IP interface and use it over IPoIB.

4.4 Build MVAPICH

There are several options to build MVAPICH 1.2 based on the underlying InfiniBand libraries you want to utilize. In this section we describe in detail the steps you need to perform to correctly build MVAPICH on your choice of InfiniBand libraries, namely OpenFabrics/Gen2, OpenFabrics/Gen2-

Hybrid, Shared Memory or QLogic InfiniPath. We also describe how to configure it for Gen2-RDMAoE to work with Mellanox ConnectX-EN 10GigE adapters.

In the following subsection, we describe how to build and configure the OpenFabrics/Gen2 device. In later subsections, we describe the building and configuration of the other devices: Gen2/Hybrid (4.4.2), Gen2-RDMAoE (4.4.3) InfiniPath (4.4.4), Shared memory (4.4.5) and TCP (4.4.6).

4.4.1 Build MVAPICH with Single-Rail configuration on OpenFabrics Gen2

There are several methods to configure MVAPICH 1.2.

- **Using Default Configuration:** Go to the `mvapich-1.2` directory. We have included a single script for OpenFabrics/Gen2 (`make.mvapich.gen2`) that takes care of different platforms, compilers and architectures. By default, the compilation script uses `gcc`. In order to select your compiler, please set the variable `CC` in the script to use either Intel, PathScale or PGI compiler. The platform/architecture is detected automatically.
- **Customize MVAPICH Configuration:** MVAPICH has many optimization schemes to enhance performance. While some schemes can be turned on/off by the compilation flags which are listed in the variable `CFLAGS` (in the default compilation script), many more optimizations and tuning is possible using environmental parameters. A list of all possible parameters is in Section 9.
  - eXtended Reliable Connection Support (XRC): By default this support is compiled in to allow the usage of the new scalable XRC transport of InfiniBand. OFED 1.3 or later is required. If using an older version, then remove the `-DXRC` from the `CFLAGS` variable.
  - Memory-to-memory Reliability: When compiled with this support, MVAPICH performs memory-to-memory error checking for each data-transfer between a pair of nodes. This is useful for detecting errors across I/O buses. In case any error is detected during the transfer, MVAPICH will re-transmit corrupted messages. Controlled by `-DMEMORY_RELIABLE`.
  - Shared library support: Enables the use of shared libraries. The flag `--enable-sharedlib` should be added as a parameter to `configure` in the default `make.mvapich.gen2` script.
  - ADIO driver for Lustre: When compiled with this support, MVAPICH will use the optimized driver for Lustre. In order to enable this feature, the flag `--with-romio --with-file-system=lustre` should be passed to `configure` in the default `make.mvapich.gen2` script. You can add support for more file systems using –with-romio –with-file-system=lustre+nfs+pvfs2.
  - TotalView Debugger support: This enables the use of TotalView debugger for your MPI applications. In order to enable this feature, you need to pass

`--enable-sharedlib` and `--enable-debug` options to `configure` in the `make.mvapich.gen2` script.

After setting all the parameters, the script `make.mvapich.gen2` configures, builds and installs the entire package in the directory specified by the variable `PREFIX`.

4.4.2 Build MVAPICH with OpenFabrics/Gen2 Hybrid Mode (Gen2-Hybrid)

There are several methods to configure MVAPICH 1.2.

- **Using Default Configuration:** Go to the `mvapich-1.2` directory. We have included a single script for Gen2-Hybrid (`make.mvapich.hybrid`) that takes care of different platforms, compilers and architectures. By default, the compilation script uses `gcc`. In order to select your compiler, please set the variable `CC` in the script to use either Intel, PathScale or PGI compiler. The platform/architecture is detected automatically.
- **Customize MVAPICH Configuration:** MVAPICH has many optimization schemes to enhance performance. While some schemes can be turned on/off by the compilation flags which are listed in the variable `CFLAGS` (in the default compilation script), many more optimizations and tuning is possible using environmental parameters. A list of all possible parameters is in Section 10.
  - eXtended Reliable Connection Support (XRC): By default this support is compiled in to allow the usage of the new scalable XRC transport of InfiniBand. OFED 1.3 or later is required. If using an older version, then remove the `-DXRC` from the `CFLAGS` variable.
  - TotalView Debugger support: This enables the use of TotalView debugger for your MPI applications. In order to enable this feature, you need to pass `--enable-sharedlib` and `--enable-debug` options to `configure` in the `make.mvapich.gen2_hybrid` script.

After setting all the parameters, the script `make.mvapich.hybrid` configures, builds and installs the entire package in the directory specified by the variable `PREFIX`.

4.4.3 Build MVAPICH with Single-Rail configuration on OpenFabrics Gen2-RDMAoE

We can configure and build the MVAPICH library to run over the Gen2-RDMAoE interface by using the (`make.mvapich.gen2`) script which is located inside `mvapich-1.2` directory. This script takes care of building the MVAPICH library for different platforms, compilers and architectures. By default, the compilation script uses `gcc`. In order to select your compiler, please set the variable `CC` in the script to use either Intel, PathScale or PGI compiler. The platform/architecture is detected automatically.

The script also has the ability to automatically detect whether the platform has support for RDMAoE and configure the library to use it accordingly. For RDMAoE functionality to work properly, a version of OFED from the OFED-1.5-RDMAoE branch must be installed on all the systems.

4.4.4 Build MVAPICH with QLogic InfiniPath

There are several methods to configure MVAPICH 1.2.

- **Using Default Configuration:** Go to the `mvapich-1.2` directory. We have included a single script for InfiniPath (`make.mvapich.psm`) that takes care of different platforms, compilers and architectures. By default, the compilation script uses `gcc`. In order to select your compiler, please set the variable `CC` in the script to use either Intel, PathScale or PGI compiler. The platform/architecture is detected automatically.
- **Customize MVAPICH Configuration:** MVAPICH has many optimization schemes to enhance performance. While some schemes can be turned on/off by the compilation flags which are listed in the variable `CFLAGS` (in the default compilation script), many more optimizations and tuning is possible using environmental parameters. A list of all possible parameters is in Section 9.
  - TotalView Debugger support: This enables the use of TotalView debugger for your MPI applications. In order to enable this feature, you need to pass `--enable-sharedlib` and `--enable-debug` options to `configure` in the `make.mvapich.psm` script.

After setting all the parameters, the script `make.mvapich.psm` configures, builds and installs the entire package in the directory specified by the variable `PREFIX`.

4.4.5 Build MVAPICH with Shared Memory Device

In the `mvapich-1.2` directory, we have provided a script `make.mvapich.smp` for building MVAPICH over shared memory intended for single SMP systems. The script `make.mvapich.smp` takes care of different platforms, compilers and architectures. By default, the compilation script uses `gcc`. In order to select your compiler, please set the variable `CC` in the script to use either Intel, PathScale or PGI compilers. The platform/architecture is detected automatically. The usage of the shared memory device can be found in 5.2.

4.4.6 Build MVAPICH with TCP/IPoIB

In the `mvapich-1.2` directory, we have provided a script `make.mvapich.tcp` for building MVAPICH over TCP/IP intended for use over IPoIB (IP over InfiniBand). In order to select any other compiler than GCC, please set your CC variable in that script. Simply execute this script (e.g. ./make.mvapich.tcp) for completing your build.

5 Usage Instructions

This section discusses the usage methods for the various features provided by MVAPICH. If you face any problem while following these instructions, please refer to Section 7.

5.1 Compile MPI applications

Use `mpicc, mpif77, mpiCC`, or `mpif90` to compile applications. They can be found under `mvapich-1.2/bin`.

There are several options to run MPI applications. Please select one of the following options based on your need.

5.2 Run MPI applications using `mpirun_rsh`

Prerequisites:

- Either `ssh` or `rsh` should be enabled between the front nodes and the computing nodes. In addition to this setup, you should be able to login to the remote nodes without any password prompts.
- All hostnames should resolve to the same IP address on all machines. For instance, if a machine's hostnames resolves to 127.0.0.1 due to the default /etc/hosts on some linux distributions it leads to incorrect behavior of the library.

Examples of running programs using `mpirun_rsh`:

```
$ mpirun_rsh -np 4 n0 n1 n2 n3 ./cpi
```

The above command runs `cpi` on nodes n0, n1, n2 and n3 nodes, one process per each node. By default `ssh` is used.

```
$ mpirun_rsh -rsh -np 4 n0 n1 n2 n3 ./cpi
```

The above command runs `cpi` on nodes n0, n1, n2 and n3 nodes, one process per each node. `rsh` is used regardless of whether `ssh` or `rsh` is used when compiling MVAPICH.

```
$ mpirun_rsh -np 4 -hostfile hosts ./cpi
```

A list of nodes are in hosts, one per line. MPI ranks are assigned in order of the hosts listed in the hosts file or in the order they are passed to mpirun_rsh, i.e., if the nodes are listed as n0 n1 n0 n1, then n0 will have two processes, rank 0 and rank 2; whereas n1 will have rank 1 and 3. This rank distribution is known as “cyclic”. If the nodes are listed as n0 n0 n1 n1, then n0 will have ranks 0 and 1; whereas n1 will have ranks 2 and 3. This rank distribution is known as “block”.

If you are using the shared memory device, then host names must be omitted:

```
$ mpirun_rsh -np 4 ./cpi
```

Many parameters of the MPI library can be very easily configured during run-time using environmental variables. In order to pass any environment variable to the application, simply put the variable names and values just before the executable name, like in the following example:

```
$ mpirun_rsh -np 4 -hostfile hosts ENV1=value ENV2=value
./cpi
```

Note that the environmental variables should be put immediately before the executable.

Alternatively, you may also place environmental variables in your shell environment (e.g. `.bashrc`). These will be automatically picked up when the application starts executing.

Please note that there are many different parameters which could be used to improve the performance of applications depending upon their requirements from the MPI library. For a discussion on how to identify which variables may be of interest to you, please take a look at Section 8.

Other options of `mpirun_rsh` can be obtained using

```
$ mpirun_rsh --help
```

Note that mpirun_rsh is sensitive to the ordering of the command-line options.

5.3 Run MPI applications using SLURM

SLURM is an open-source resource manager designed by Lawrence Livermore National Laboratory. SLURM software package and its related documents can be downloaded from: *http://www.llnl.gov/linux/s*

Once SLURM is installed and the daemons are started, applications compiled with MVAPICH can be launched by SLURM, e.g.

```
$ srun -n2 --mpi=mvapich ./a.out
```

The use of SLURM enables many good features such as explicit CPU and memory binding. For example, if you have two processes and want to bind the first process to CPU 0 and Memory 0, and the second process to CPU 4 and Memory 1, then it can be achieved by:

```
$ srun --cpu_bind=v,map_cpu:0,4 --mem_bind=v,map_mem:0,1 -n2
--mpi=mvapich ./a.out
```

For more information about SLURM and its features please visit SLURM website.

5.4 Run MPI applications with Scalable Collectives

MVAPICH provides shared memory implementations of important collectives: `MPI_Allreduce`, `MPI_Reduce`, `MPI_Barrier` and `MPI_Bcast`. It also has support for Enhanced MPI_Allgather. These collective operations are enabled by default. Shared Memory Collectives are supported over Gen2, Gen2/Hybrid, PSM and Shared Memory devices. The PSM device currently only has MPI_Barrier and MPI_Bcast shared memory implementation.

These operations can be disabled all at once by setting VIADEV_USE_SHMEM_COLL to 0 or one at a time by using the following environment variables:

- To disable Shmem MPI_Allreduce: VIADEV_USE_SHMEM_ALLREDUCE=0
- To disable Shmem MPI_Reduce: VIADEV_USE_SHMEM_REDUCE=0
- To disable Shmem MPI_Barrier: VIADEV_USE_SHMEM_BARRIER=0
- To disable Shmem MPI_Bcast: VIADEV_USE_SHMEM_BCAST=0
- To disable new MPI_Allgather: VIADEV_USE_NEW_ALLGATHER=0

Please refer to section 9.7 for tuning the various environment variables.

5.5 Run MPI Application with mpirun_rsh using OpenFabrics RDMAoE Device

MVAPICH is can automatically detect whether the underlying hardware is capable of supporting RDMAoE communication and automatically adapt the connection management mechanism to it. Such detection requires support from the OFED software installed on the system. Currently only the RDMAoE branch of OFED provides such support.

In the absence of a compatible OFED version, Gen2-RDMAoE support can enabled with the use of the run time environment variable ``VIADEV_USE_RDMAOE''.

Programs can be executed as follows:

```
$ mpirun_rsh -np 2 VIADEV_USE_RDMAOE=1 ENV1=value1 prog
```

5.6 Run MPI applications using shared library support

MVAPICH provides shared library support. This feature allows you to build your application on top of MPI shared library. If you choose this option, you still will be able to compile applications with static libraries. But as default, when you have shared library support enabled, your applications will be built on top of shared libraries automatically. The following commands provide some examples of how to build and run your application with shared library support.

- To compile your application with shared library support. Run the following command.

  ```
  $ mpicc -o cpi cpi.c
  ```

- To execute an application compiled with shared library support, you need to specify the path to the shared library by setting `LD_LIBRARY_PATH=<path-to-shared-libraries>` in the command line.

  For example,

  ```
  $ mpirun_rsh -np 2 n0 n1 LD_LIBRARY_PATH=$MVAPICH_BUILD/lib/shared
  ./cpi
  ```

  Again, note that "LD_LIBRARY_PATH=path-to-shared-libraries" should be put immediately before the executable file.

- To disable MVAPICH shared library support even if you have installed MVAPICH. Run the following command.

  ```
  $ mpicc -noshlib -o cpi cpi.c
  ```

5.7 Run MPI applications using ADIO driver for Lustre

MVAPICH contains optimized Lustre ADIO support for the OpenFabrics/Gen2 device. The Lustre directory should be mounted on all nodes on which MVAPICH processes will be running. Compile MVAPICH with ADIO support for Lustre as described in Section 4.4.1. If your Lustre mount is /mnt/datafs on nodes n0 and n1, on node n0, you can compile and run your program as follows:

```
$ mpicc -o perf romio/test/perf.c
$ mpirun_rsh -np 2 n0 n1 <path to perf>/perf -fname
/mnt/datafs/testfile
```

If you have enabled support for multiple file systems, append the prefix "lustre:" to the name of the file. For example:

```
$ mpicc -o perf romio/test/perf.c
$ mpirun_rsh -np 2 n0 n1 ./perf -fname
lustre:/mnt/datafs/testfile
```

5.8 Run MPI applications using TotalView Debugger support

MVAPICH provides TotalView support for the OpenFabrics/Gen2 (`mpid/ch_gen2`), OpenFabrics/Gen2-Hybrid (`mpid/ch_hybrid`), InfiniPath (`mpid/psm`) and Shared-Memory devices (`mpid/ch_smp`). You need to use `mpirun_rsh` when running TotalView. The following commands also provide an example of how to build and run your application with TotalView support. *Note: running TotalView demands correct setup in your environment, if you encounter any problem with your setup, please check with your system administrator for help.*

- Define ssh as a `TVDSVRLAUNCHCMD` variable in your default shell. For example, in $HOME/.bashrc

  ```
  $ echo "export TVDSVRLAUNCHCMD=ssh" >>
  $HOME/.bashrc
  ```

- Configure MVAPICH with the configure options `--enable-debug --enable-sharedlib` in addition to the default options. Enable mpirun_rsh compilation with debug symbols. For example,
  ```
  $ export MPIRUN_CFLAGS="$MPIRUN_CFLAGS -g"
  ```
  and then build MVAPICH.

- Compile your program with a flag -g
  ```
  $ mpicc -g -o prog prog.c
  ```

- Define the correct path to TotalView as the TOTALVIEW variable. For example, under bash shell:
  ```
  $ export TOTALVIEW=<path_to_TotalView>
  ```

- Run your program:

  ```
  $ mpirun_rsh -tv -np 2 n0 n1
  LD_LIBRARY_PATH=$MVAPICH_BUILD/lib/shared:$MVAPICH_BUILD/lib
  prog
  ```

- Troubleshooting:

  - X authentication errors: check if you have enabled X Forwarding
    ```
    $ cat ``ForwardX11 yes'' >> $HOME/.ssh/config
    ```
  - rsh connection time out: check if you have defined `TVDSVRLAUNCHCMD` as ssh in your default shell file, .bashrc, .cshrc, or the like.
  - ssh authentication error: ssh to the computer node with its long form host name, for example, ssh i0.domain.osu.edu
  - If a debug session is terminated with an alarm message, mpirun_rsh may have timed out waiting for the job launch to complete. Use a larger MPIRUN_TIMEOUT (section 9.1.2) to work around this problem.

5.9 Run MPI applications with Multi-Pathing Support for Multi-Core Architectures

Multi-pathing (multiple ports, adapters and multiple paths provided by the LMC mechanism) can be used for multi-core systems. With this support, processes executing on the same node can leverage the above configurations by binding to one of the available configuration. MVAPICH provides multiple choices to the user for leveraging this functionality, which are described in the upcoming examples. This functionality is currently available only in the OpenFabrics/Gen2 device.

- To allow processes on the same node to use multiple ports in a round robin fashion `$ mpirun_rsh -np`

- To allow processes on the same node to use multiple adapters in a round robin fashion
  ```
  $ mpirun_rsh -np 4 n0 n0 n1 n1 VIADEV_USE_MULTIHCA=1 ./cpi
  ```

- To allow processes on the same node to use multiple adapters and multiple ports in a round robin fashion
  ```
  $ mpirun_rsh -np 4 n0 n0 n1 n1 VIADEV_USE_MULTIHCA=1
  VIADEV_USE_MULTIPORT=1 ./cpi
  ```
  The usage of multiple paths is disabled by default. Its usage can be controlled by using the parameter VIADEV_USE_LMC (9.2.6).

- In the above examples, the binding of paths to processes is done in a round robin fashion. In addition, MVAPICH allows a user to explicitly specify the Adapter and Port to be used by the library. This can be done by specification in the hostfile as follows.
  ```
  $ cat hosts
  n0:mthca0:1
  n0:mthca1:2
  n1:mthca0:2
  n1:mthca1:1
  ```
  With this specification, process 0 would be bound to port1 of adapter “mthca0”, process 1 to port 2 of adapter “mthca1” and so on.

5.10 Run MPI Application with Network Fault Tolerance Support (for OpenFabrics Gen2 Device)

MVAPICH supports network fault recovery as well as fault resilience. The Network fault recovery feature uses InfiniBand Automatic Path Migration mechanism. This support is available for MPI applications using OpenFabrics stack and InfiniBand adapters.

To enable this functionality, a run-time variable, VIADEV_USE_APM (section 9.9.1) can be enabled, as shown in the following example:

```
$ mpirun_rsh -np 2 -hostfile hosts VIADEV_USE_APM=1 ./cpi
```

MVAPICH also supports testing Automatic Path Migration in the subnet in the absence of network faults. This can be controlled by using a run-time variable VIADEV_USE_APM_TEST (section 9.9.2). This should be combined with VIADEV_USE_APM as follows:

```
$ mpirun_rsh -np 2 -hostfile hosts VIADEV_USE_APM=1 VIADEV_USE_APM_TEST=1
./cpi
```

The Network Fault Resiliency feature allows long running MPI applications to tolerate transient network failures by resetting HCAs and re-establishing connections. To use this functionality, a run time variable VIADEV_USE_NFR (section 9.9.3) can be enabled, as shown in the following example:

```
$ mpirun_rsh -np 2 -hostfile hosts VIADEV_USE_NFR=1 ./cpi
```

5.11 Run Memory Intensive Applications on Multi-core Systems

Process to CPU mapping may affect application performance on multi-core systems, especially for memory intensive applications. If the number of processes is smaller than the number of CPU's/cores, it is preferable to distribute the processes on different chips to avoid memory contention because CPU's/cores on the same chip usually share the memory controller. MVAPICH provides flexible user defined CPU mapping. To use it, first make sure CPU affinity is set (Section 9.6.5). Then use the run-time environment variable VIADEV_CPU_MAPPING to specify the CPU/core mapping. For example, if it is a quad-core system in which cores [0-3] are on the same chip and cores [4-7] are on another chip, and you need to run an application with 2 processes, then the following mapping will give the best performance:

```
$ mpirun_rsh -np 2 n0 n0 VIADEV_CPU_MAPPING=0:4 ./a.out
```

or

```
$ mpirun_rsh -np 2 n0 n0 VIADEV_CPU_MAPPING=0,4 ./a.out
```

In this case process 0 will be mapped to core 0 and process 1 will be mapped to core 4. The core numbers can be separated by either a single ":" or ",". We recommend to use ":" since it is unified with MVAPICH2. The option of using "," is for backward compatibility and may be removed in the future.

More information about VIADEV_CPU_MAPPING can be found in Section 9.6.6.

6 Using OSU Benchmarks

If you have arrived at this point, you have successfully installed MVAPICH. Congratulations!! In the `mvapich-1.2/osu_benchmarks` directory, we provide four basic performance tests: one-way latency test, uni-directional bandwidth test, bi-directional bandwidth test multiple bandwidth/message rate, and MPI-level broadcast latency test. You can compile and run these tests on your machines to evaluate the basic performance of MVAPICH.

These benchmarks as well as other benchmarks (such as for one-sided operations in MPI-2) are available on our projects' web page. Sample performance numbers for these benchmarks on representative platforms and IBA gears are also included on our projects' web page. You are welcome to compare your performance numbers with our numbers. If you see any big discrepancy, please let the MVAPICH community know by sending an email to the mailing list mvapich-discuss@cse.ohio-state.edu.

7 FAQ and Troubleshooting

Based on our experience and feedback we have received from our users, here we include some of the problems a user may experience and the steps to resolve them. If you are experiencing any other problem, please feel free to contact the MVAPICH community by sending an email to the mailing list mvapich-discuss@cse.ohio-state.edu.

MVAPICH can be used over multiple underlying InfiniBand libraries, namely OpenFabrics (Gen2), OpenFabrics (Gen2-Hybrid), and QLogic InfiniPath. It can also be used with ConnectX-EN 10GigE adapters with Gen2-RDMAoE mode. Based on the underlying library being utilized, the troubleshooting steps may be different. However, some of the troubleshooting hints are common for all underlying libraries. Thus, in this section, we have divided the troubleshooting tips into four sections: General troubleshooting and Troubleshooting over any one of the three InfiniBand libraries.

7.1 General Questions and Troubleshooting

7.1.1 How can I check what version I am using?

Running the following command will provide you with the version of MVAPICH that is being used.

```
$ mpirun_rsh -v
```

7.1.2 Are `fork()` and `system()` supported?

`fork()` and `system()` is supported for Gen2 and Gen2-Hybrid devices as long as the kernel is being used is Linux 2.6.16 or newer. Additionally, the version of OFED used should be 1.2 or higher. The environment variable IBV_FORK_SAFE=1 must also be set to enable fork support.

7.1.3 My application cannot pass `MPI_Init`

This is a common symptom of several setup issues related to job startup. Please make sure of the following things:

- If you have enabled `ssh` based startup, make sure that you have set up ssh keys for logging into all the nodes without any password prompt.
- If you have enabled `rsh` based startup, make sure that `rsh, rlogin` etc. are active on all the nodes and you can log in without any password prompts.
- Please make sure the host names supplied to MVAPICH for the particular job match the host names in file `/etc/hosts` present on each of the target nodes.

- Please make sure you can run some InfiniBand level program on the nodes you are trying to run MPI programs. Usually running `ibv_rc_pingpong` (for OpenFabrics Gen2) is a good choice.

7.1.4 My application hangs/aborts in Collectives

MVAPICH implements highly optimized shared memory collective algorithms for frequently used collectives such as `MPI_Allreduce`, `MPI_Reduce`, `MPI_Barrier`, `MPI_Bcast` and `MPI_Allgather`. The optimized implementations have been well tested and tuned. However, if you face any problems in these collectives for your application, please disable the optimized collectives. For example, if you want to disable MPI_Allreduce, you can do:

```
$ mpirun_rsh -np 8 -hostfile hf VIADEV_USE_SHMEM_ALLREDUCE=0
./a.out
```

The complete list of all such parameters is given in 9.7

7.1.5 Building MVAPICH with g77/gfortran

The gfortran compiler can be used for F77 and F90. In order to make this work, the following environment variables should be set prior to running the build script:

```
$ export F77=gfortran
$ export F90=gfortran
$ export F77_GETARGDECL=" "
```

If g77 and gfortran are used together for F77 and F90 respectively, it might be necessary to set the following environment variable in order to get around possible compatibility issues:

```
$ export F90FLAGS="-ff2c"
```

7.1.6 Running MPI programs built with gfortran

MPI programs built with gfortran might not appear to run correctly due to the default output buffering used by gfortran. If it seems there is an issue with program output, the `GFORTRAN_UNBUFFERED_ALL` variable can be set to “y” when using `mpirun_rsh` to fix the problem. Running the `pi3f90` example program using this variable setting is shown below:

```
$ mpirun_rsh -np 2 n1 n2 GFORTRAN_UNBUFFERED_ALL=y
./pi3f90
```

7.1.7 Timeout During Debugging

If a debug session is terminated with an alarm message, mpirun_rsh may have timed out waiting for the job launch to complete. Use a larger MPIRUN_TIMEOUT (section 9.1.2) to work around this problem.

7.1.8 Unexpected exit status

If an application task terminates unexpectedly during job launch, mpirun_rsh may print the message:

```
mpispawn.c:303 Unexpected exit status
```

This usually indicates a problem with the application. Other error messages around this (if any) might point to the actual issue.

7.1.9 /usr/bin/env: mpispawn: No such file or directory

If mpirun_rsh fails with this error message, it was unable to locate a necessary utility. This can be fixed by ensuring that all MVAPICH executables are in the PATH on all nodes.

If PATHs cannot be setup as mentioned, then invoke mpirun_rsh with a path. For example:

```
/path/to/mpirun_rsh -np 2 node1 node2
./mpi_proc
```

or

```
../../path/to/mpirun_rsh -np 2 node1 node2
./mpi_proc
```

7.1.10 Totalview complains that "The MPI library contains no suitable type definition for struct MPIR_PROCDESC"

Ensure that the MVAPICH job launcher mpirun_rsh is compiled with debug symbols. Details are available in Section 5.8.

7.1.11 **io No such file or directory*?

If you are using ADIO support for Lustre, please make sure that:

– Lustre is setup correctly, and that you are able to create, read to and write from files in the Lustre mounted directory.

– The Lustre directory is mounted on all nodes on which MVAPICH processes with ADIO support for Lustre are running.

– The path to the file is correctly specified.
– The permissions for the file or directory are correctly specified.

7.1.12 My program segfaults with: File locking failed in ADIOI_Set_lock?

If you are using ADIO support for Lustre, the recent Lustre releases require an additional mount option to have correct file locks.
So please include the following option with your lustre mount command: ”-o localflock”.
For example:

```
$ mount -o localflock -t lustre
xxxx@o2ib:/datafs /mnt/datafs
```

7.1.13 MPI+OpenMP shows bad performance

MVAPICH uses CPU affinity to have better performance for single-threaded programs. For multi-threaded programs, such as MPI+OpenMP model, it may schedule all the threads of a process to run on the same CPU. CPU affinity should be disabled in this case to solve the problem, e.g.

```
$ mpirun_rsh -np 2 n1 n2 VIADEV_USE_AFFINITY=0
./a.out
```

More information about CPU affinity and CPU binding can be found in Sections 9.6.5 and 9.6.6.

7.1.14 Fortran support is disabled with Sun Studio 12 compilers

Please replace the `-Wl,-rpath` option in the build scripts (e.g. `make.mvapich.gen2`) with `-R` when Sun Studio 12 compilers are used.

7.1.15 Other MPICH problems

Several well-known MPICH related problems on different platforms and environments have already been identified by Argonne. They are available on the MPICH patch web page.

7.1.16 Does MVAPICH Work Across AMD and Intel Systems?

Yes, as long as you compile MVAPICH and your programs on one of the systems, either AMD or Intel, and run the same binary across the systems. MVAPICH has platform specific parameters for performance optimizations and it may not work if you compile MVAPICH and your programs on different systems and try to run the binaries together.

7.2 Troubleshooting with MVAPICH/OpenFabrics(Gen2)

In this section, we discuss the general error conditions for MVAPICH based on OpenFabrics Gen2.

7.2.1 No IB Devices found

This error is generated by MVAPICH when it cannot find any Gen2 InfiniBand devices. If you are experiencing this error, then please make sure that your Gen2 installation is proper. You can do so by doing the following:

```
$ locate libibverbs
```

This tells you if you have installed `libibverbs` (the Gen2 verbs layer) or not. By default it installs in `/usr/local`.

If you have installed `libibverbs`, then please check if the OpenFabrics Gen2 drivers are loaded. You can do so by:

```
$ lsmod | grep ib
```

If this command does not list `ib_uverbs`, then probably you haven't started all OpenFabrics Gen2 services. Please refer to the OpenFabrics Wiki installation cheat sheet for more details on setting up the OpenFabrics Gen2 stack.

7.2.2 Error getting HCA Context

This error is generated when MVAPICH cannot "open" the HCA (or the InfiniBand communication device). Please execute:

```
$ ls -l /dev/infiniband
```

If this command shows any devices `uverbs0` with read/write permissions for users as shown below, please consult the "Loading kernel components" section of the OpenFabrics Wiki installation cheat sheet.

```
crw-rw-rw- 1 root root 231, 192 Feb 24 14:31
uverbs0
```

7.2.3 CQ or QP Creation failure

If you encounter this error, then you need to set the maximum available locked memory value for your system. The usual Linux defaults are quite low to what is required for HPC applications. One way to do this is to edit the file `/etc/security/limits.conf` and enter the following line:

```
* soft memlock phys_mem_size_in_KB
```

Where, phys_mem_size_in_KB is the MemTotal value reported by /proc/meminfo. In addition, you need to enter the following line in /etc/init.d/sshd and then restart sshd.

```
ulimit -l phys_mem_size_in_KB
```

7.2.4 No Active Port found

MVAPICH generates this error when it cannot find any port active for the specific HCA being used for communication. This probably means that the ports are not configured to be a part of the InfiniBand subnet and thus are not "Active". You can check whether the port is active or not, by using the following command:

```
$ ibstat
```

Please look at the "State" field for the status of the port being used. To bring a port to "Active" status, on any node in the same InfiniBand subnet, execute the following command:

```
# opensm -o 1
```

Please note that you need superuser privilege for this command. This command invokes the InfiniBand subnet manager (OpenSM) and asks it to sweep the subnet once and make all ports "Active". OpenSM is usually installed in /usr/local/bin.

7.2.5 Couldn't modify SRQ limit

This means that your HCA doesn't support the ibv_modify_srq feature. Please upgrade the firmware version and OpenFabrics Gen2 libraries on your cluster. You can obtain the latest Mellanox firmware images from this web page.

Even after updating your firmware and OpenFabrics Gen2 libraries, if you continue to experience this problem, please edit make.mvapich.gen2 and replace -DMEMORY_SCALE with -DADAPTIVE_RDMA_FAST_PATH. After making this change you need to re-build the MVAPICH library. Note that you should first try to update your firmware and OpenFabrics Gen2 libraries before taking this measure.

If you believe that your HCA supports this feature and yet you are experiencing this problem, please contact the MVAPICH community by posting a note to mvapich-discuss@cse.ohio-state.edu mailing list.

7.2.6 Got completion with error code 12

The error code 12 indicates that the InfiniBand HCA has given up after attempting to send the packet after several tries. This can be caused by either loose or faulty cables. Please check the

InfiniBand connectivity of your cluster. Additionally, you may check the error rates at the respective HCAs using:

```
$ ibchecknet
```

This utility (usually installed in `/usr/local/bin`) sweeps the InfiniBand subnet and reports ports that are OK or if they have errors. You may try to quiesce the entire cluster and bring it up after an InfiniBand switch reboot.

7.2.7 Hang with VIADEV_USE_LMC=1

The VIADEV_USE_LMC parameter allows the usage of multiple paths for multi-core and multi-way SMP systems, set up the subnet manager 9.2.6. The subnet manager allows different routing engines to be used (Min-Hop routing algorithm by default). We have noticed hangs using this parameter with Up/Down routing algorithm of OpenSM. There are two ways to fix this problem:

- Disable the VIADEV_USE_LMC. This can be done in the following manner.

  ```
  # mpirun_rsh -np 2 n0 n1 VIADEV_USE_LMC=0
  ./prog
  ```

- Use the Min-Hop Algorithm. This can be done by invoking opensm with the min-hop algorithm. Please use the following command, which provides an LMC value of 4, and makes sure that the LIDs are re-assigned using the -r option.

  ```
  # opensm -o -l4 -r
  ```

7.2.8 Failure with Automatic Path Migration

MVAPICH provides network fault tolerance with Automatic Path Migration (APM). However, APM is supported only with OFED 1.2 onwards. With OFED 1.1 and prior versions of Open-Fabrics drivers, APM functionality is not completely supported. Please refer to Section 9.9.1 and section 9.9.2

7.2.9 Problems with `mpirun_rsh`

MVAPICH 1.2 provides a new more scalable startup procedure by default. If for some reason the old version is desired, it can be enabled using the `-legacy` flag to `mpirun_rsh`.

```
$ mpirun_rsh -legacy ...
```

7.3 Troubleshooting with MVAPICH/QLogic InfiniPath

7.3.1 Low Bandwidth

Incorrect settings of MTRR mapping may result in achieving a low bandwidth with InfiniPath hardware. To alleviate this situation, BIOS settings for MTRR mapping may be edited to “Discrete”. For further details, please refer to the InfiniPath User Guide.

7.3.2 Cannot find -lpsm_infinipath

Variable IBHOME_LIB in make.mvapich.psm file does not point to correct location. IBHOME_LIB should point to the directory containing the InfiniPath device libraries. By default they are installed in /usr/lib or /usr/lib64.

7.3.3 Mandatory variables not set

IBHOME, PREFIX, CC, F77 are mandatory variable required by the installation script and must be set in the file make.mvapich.psm. IBHOME - directory which contains the InfiniPath header file include directory. By default InfiniPath header file include directory is in /usr. PREFIX - directory where MVAPICH should be installed. CC - C compiler. Typically set to gcc. F77 - fortran compiler. Typically set to g77.

7.3.4 Can’t open /dev/ipath, Network Down

This probably means that the ports are not configured to be a part of the InfiniBand subnet and thus are not “Active”. You can check whether the port is active or not, by using the following command on that node:

```
$ ipath_control -i
```

Please look at the “Status” field for the status of the port being used. To bring a port to “Active” status, on any node in the same InfiniBand subnet, execute the following command:

```
# opensm -o
```

Please note that you may need superuser privilege for this command. This command invokes the InfiniBand subnet manager (OpenSM) and asks it to sweep the subnet once and make all ports “Active”. OpenSM is usually installed in `/usr/local/bin`. You may also look at the file /sys/bus/pci/drivers/ib_ipath/status_str to verify that the InfiniPath software is loaded correctly. For details, please refer to InfiniPath user guide, download able from www.qlogic.com.

7.3.5 No ports available on /dev/ipath

This is a limitation of InfiniPath Release 2.0. By default it allows a maximum of eight processes per QHT7140 HCA and four processes with QLE7140 HCA. To overcome this, please consult your InfiniPath support provider.

8 Tuning and Scalability Features for Large Clusters

MVAPICH supports many different parameters for tuning and extracting the best performance for a wide range of platforms and applications. These parameters can be either compile time parameters or run time parameters. Please refer to section 9 for a complete description of all the parameters. In this section we classify these parameters depending on what you are tuning for and provide guidelines on how to use them.

8.1 Job Launch Tuning

MVAPICH 1.2 has a scalable mpirun_rsh which uses a tree based mechanism to spawn processes. The degree of this tree is determined dynamically to keep the depth low. For large clusters, it might be beneficial to further flatten the tree by specifying a higher degree. The degree can be overridden with the environment variable MT_DEGREE (section 9.1.1).

8.2 Network Point-to-point Tuning

In MVAPICH we use Shared Receive Queue (SRQ) support which consumes less memory than other methods. It can lead to a significant reduction in the memory footprint of MVAPICH.

To enable this mode, please include `-DMEMORY_SCALE` in your `make.mvapich.gen2` (it is included by default). Once you have enabled the scalable memory mode in MVAPICH, there are four aspects by which you can customize the memory usage and performance ratio according to the needs of your cluster.

8.2.1 Shared Receive Queue (SRQ) Tuning

The main environmental parameters controlling the behavior of the Shared Receive Queue design are:

- VIADEV_SRQ_MAX_SIZE (9.5.2)
- VIADEV_SRQ_SIZE (9.5.3)
- VIADEV_SRQ_LIMIT (9.5.4)
- VIADEV_VBUF_POOL_SIZE (9.3.9)

Starting with 1.0, MVAPICH uses a dynamic re-size of the number of buffers used for the SRQ by default. The parameter VIADEV_SRQ_MAX_SIZE is the maximum size of the Shared Receive Queue (default 4096). You may increase this to value 8192 if the application requires very large number of processors (8K and beyond). The application will start by only using VIADEV_SRQ_SIZE buffers (default 256) and will double this value on every SRQ limit event (up to VIADEV_SRQ_MAX_SIZE). For long running applications this re-size should show little effect. If needed, the VIADEV_SRQ_SIZE van be increased to 1024 or higher as needed for applications.

VIADEV_SRQ_LIMIT defines the low watermark for the flow control handler. This can be reduced if your aim is to reduce the number of interrupts.

VIADEV_VBUF_POOL_SIZE is a fixed number of pool of `vbufs`. These vbufs can be shared among all different connections depending on the communication needs of each connection. You may want to increase this number for large scale clusters (4K and beyond).

8.2.2 On-Demand Connection Tuning

The major environmental variables controlling the behavior of the connection management in MVAPICH are:

- VIADEV_CM_RECV_BUFFERS (9.8.1)
- VIADEV_CM_MAX_SPIN_COUNT (9.8.2)
- VIADEV_CM_TIMEOUT (9.8.3)

VIADEV_CM_RECV_BUFFERS is the number of buffers used by the connection manager to establish new connections. These buffers are very small (around 20 bytes) and they are shared for all InfiniBand connections, so this value may be increased to 8192 for large clusters to avoid retries in case of packet drops.

VIADEV_CM_MAX_SPIN_COUNT is the number of times the connection manager polls for new incoming connections. This may be increased to reduce the interrupt overhead when lot of incoming connections are started at the same time.

VIADEV_CM_TIMEOUT is the timeout value associated with connection request messages on the UD channel. Decreasing this may lead to faster retries, but at the cost of generating duplicate messages. Similarly increasing this may lead to slower retries but lesser chance of duplicate messages.

8.2.3 Adaptive RDMA Tuning

MVAPICH implements a dynamic allocation and utilization of the RDMA mechanism for short messages. It can lead to significant reduction in memory footprint of MVAPICH.

There are two environmental parameters:

- VIADEV_ADAPTIVE_RDMA_LIMIT (9.3.20)
- VIADEV_ADAPTIVE_RDMA_THRESHOLD (9.3.21)

These two parameters control the behavior of this dynamic scheme. VIADEV_ADAPTIVE_RDMA_LIMIT controls the maximum number of processes for which the "fast" RDMA buffers are allocated. For very large scale clusters, it is suggested to set this value to -1, which means RDMA buffers will be allocated for $log(n)$ number of connections (where n is the number of processes in the application).
VIADEV_ADAPTIVE_RDMA_THRESHOLD is the number of messages exchanged per connection before RDMA buffers are allocated for that connection. For very large scale clusters, it is suggested that this value be increased so that only very frequently communicating connections allocate RDMA buffers.

In addition, the following parameters are also important in tuning the memory requirement: VIADEV_VBUF_TOTAL_SIZE (9.3.2) and VIADEV_NUM_RDMA_BUFFER (9.3.1).

The product of VBUF_TOTAL_SIZE and VIADEV_NUM_RDMA_BUFFER generally is a measure of the amount of memory registered for eager message passing. These buffers are not shared across connections.

To provide the best performance (latency/bandwidth) to memory ratio, we have decided on a set of default values for these parameters. These parameters are often dependent on the execution platform. To use preset values for small, medium and large clusters (1-64, 64-256, 256-…), please use VIADEV_CLUSTER_SIZE (9.10.1) as either SMALL, MEDIUM or LARGE, respectively.

8.3 Shared Memory Point-to-point Tuning

MVAPICH uses shared memory communication channel to achieve high-performance message passing among processes that are on the same physical node. The two main parameters which are used for tuning shared memory performance for small messages are VIADEV_SMPI_LENGTH_QUEUE (10.7.3) and VIADEV_SMP_EAGER_SIZE (10.7.2). The two main parameters which are used for tuning shared memory performance for large messages are SMP_SEND_BUF_SIZE(10.7.4) and VIADEV_SMP_NUM_SEND_BUFFER (10.7.5).

VIADEV_SMPI_LENGTH_QUEUE is the size of the shared memory buffer which is used to store outstanding small and control messages. VIADEV_SMP_EAGER_SIZE defines the switch point from Eager protocol to Rendezvous protocol.

Messages larger than VIADEV_SMP_EAGER_SIZE are packetized and sent out in a pipelined manner. SMP_SEND_BUF_SIZE is the packet size, i.e. the send buffer size. VIADEV_SMP_NUM_SEND_BU is the number of send buffers. Shared memory communication can be disabled at run time by the parameter VIADEV_USE_SHARED_MEM(9.4.5).

Performance of some applications is sensitive to the rank distribution according to their communication pattern. It is advisable that processes that communicate most use the shared memory path, since it offers lower latencies compared to the network path. To adjust the process rank distribution, please refer Section 5.2 to decide which distribution "cyclic" or "block" suits the communication pattern of your application. In particular, we have found that when using "block" distribution, the performance of HPL (Linpack) is better.

8.4 Scalable Collectives Tuning

MVAPICH uses shared memory to get the best performance for many collective operations: MPI_Allreduce, MPI_Reduce, MPI_Barrier, MPI_Bcast.

The important parameters for tuning these collectives are as follows. For MPI_Allreduce, the optimized shared memory algorithm is used until the VIADEV_SHMEM_COLL_ALLREDUCE_THRESHOLD (9.7.9).

Similarly for MPI_Reduce and MPI_Bcast the corresponding threshold are specified by VIADEV_SHMEM_COLL_REDUCE_THRESHOLD (9.7.8) and VIADEV_SHMEM_COLL_BCAST_THRESHOLD (9.7.10) respectively.

For MPI_Bcast, the important parameter is the degree of the tree used for inter-node data movement. This parameter is VIADEV_BCAST_KNOMIAL (9.7.12).

For MPI_Alltoall, the two main parameters are MPIR_ALLTOALL_SHORT_MSG (9.7.13) and MPIR_ALLTOALL_MEDIUM_MSG (9.7.14). There are three main algorithms used for MPI_Alltoall: short message, medium message and long message. The short message algorithm is used until MPIR_ALLTOALL_SHORT_MSG and from then on the medium message algorithm is used until MPIR_ALLTOALL_MEDIUM_MSG. These thresholds can be tuned appropriately to get the best performance.

9 MVAPICH Parameters

9.1 Job Launch Parameters

9.1.1 MT_DEGREE

- Class: Run time
- Default: Dynamic - based on number of nodes

The degree of the hierarchical tree used by mpirun_rsh. By default mpirun_rsh uses a value that tries to keep the depth of the tree to 4. Note that unlike most other parameters described in this section, this is an environment variable that has to be set in the runtime environment (for e.g. through export in the bash shell).

9.1.2 MPIRUN_TIMEOUT

- Class: Run time
- Default: Dynamic - based on number of nodes

The number of seconds after which mpirun_rsh aborts job launch. Note that unlike most other parameters described in this section, this is an environment variable that has to be set in the runtime environment (for e.g. through export in the bash shell).

9.2 InfiniBand HCA and Network Parameters

9.2.1 VIADEV_DEVICE

- Class: Run time
- Default: First IB device found on the system

Name of the InfiniBand device. e.g. `mthca0`, `mthca1` or `ehca0` (for IBM ehca).

9.2.2 VIADEV_DEFAULT_PORT

- Class: Run time
- Default: First IB port found on the system

The default port on the InfiniBand device to be used for communication.

9.2.3 VIADEV_MAX_PORTS

- Class: Run time
- Default: 2

This variables allows to change the maximum number of ports per adapter which are supported.

9.2.4 VIADEV_USE_MULTIHCA

- Class: Run time
- Default: 0

This variable allows a user to bind processes on a node to ports attached to different HCAs on a node. It allows an efficient utilization of HCA ports in a round-robin fashion. VIADEV_MULTIHCA is an alias for this variable for backward compatibility. However, if VIADEV_USE_MULTIHCA is defined, value of VIADEV_MULTIHCA will be overwritten.

9.2.5 VIADEV_USE_MULTIPORT

- Class: Run time
- Default: 0

This variable allows a user to bind processes on a node to ports attached to different HCAs on a node. It allows an efficient utilization of HCA ports in a round-robin fashion. VIADEV_MULTIPORT is an alias for this variable for backward compatibility. However, if VIADEV_USE_MULTIPORT is defined, value of VIADEV_MULTIPORT will be overwritten.

9.2.6 VIADEV_USE_LMC

- Class: Run time
- Default: 0

This variable allows the usage of multiple paths between end nodes for multi-core/multi-way SMP systems. The path selection is on the basis of source and destination ranks.

9.2.7 VIADEV_DEFAULT_MTU

- Class: Run time
- Default: MTU1024

The internal MTU used for IB. This parameter should be a string instead of an integer. Valid values are: `MTU256`, `MTU512`, `MTU1024`, `MTU2048`, `MTU4096`.

9.2.8 VIADEV_USE_RDMAOE

- Class: Run Time
- Applicable device(s): Gen2

This parameter enables the use of RDMA over Ethernet for MPI communication. The underlying HCA and network must support this feature.

9.2.9 VIADEV_USE_XRC

- Class: Run Time
- Applicable device(s): Gen2

When MVAPICH is compiled with the XRC CFLAGS, this parameter enables use of the XRC transport of InfiniBand available on certain adapters. Enabling XRC automatically enables Shared Receive Queue and on-demand connection management.

9.2.10 VIADEV_DEFAULT_PKEY

- Class: Run Time
- Applicable device(s): Gen2

Select the partition to be used for the job.

9.3 Memory Usage and Performance Control Parameters

9.3.1 VIADEV_NUM_RDMA_BUFFER

- Class: Run time
- Default: Architecture dependent (32 for IA-32)

The number of RDMA buffers used for the RDMA fast path. This *fast path* is used to reduce latency and overhead of small data and control messages. This value is effective only when macro RDMA_FAST_PATH or ADAPTIVE_RDMA_FAST_PATH is defined.

9.3.2 VIADEV_VBUF_TOTAL_SIZE

- Class: Run time
- Default: Architecture dependent (6 KB for EM64T)

This macro defines the size of each `vbuf`.

Different presets for this value are available for different sizes of clusters VIADEV_CLUSTER_SIZE = (SMALL, MEDIUM, LARGE, AUTO).

9.3.3 VIADEV_RNDV_PROTOCOL

- Class: Run time
- Default: RPUT

This parameter chooses the underlying Rendezvous protocol

Options are:

- RPUT : Send large messages using RDMA write operations (zero-copy)
- RGET : Potentially allows for more overlap (zero-copy)
- R3 : Sends messages without registering memory by using a copy-based approach
- ASYNC : Uses an RGET based protocol to achieve asynchronous progress on large transfers. Also refer to parameter VIADEV_ASYNC_SCHEDULE (9.3.4).

9.3.4 VIADEV_ASYNC_SCHEDULE

- Class: Run time
- Default: DEFAULT

This parameter can only be used in conjunction with VIADEV_RNDV_PROTOCOL=ASYNC. It changes the schedule policy of the processes. Options are:

- RR : Round-robin. This is equivalent to SCHED_RR policy of Linux.
- FIFO : First In First Out. This is equivalent to SCHED_FIFO policy of Linux.
- DEFAULT : The default schedule policy of Linux. This is equivalent to SCHED_OTHER policy of Linux.

It has been observed that the performance of asynchronous progress design is best when round-robin schedule policy is used. However, the use of RR and FIFO schedule policies require the processes to be run in superuser mode.

9.3.5 VIADEV_RENDEZVOUS_THRESHOLD

- Class: Run time
- Default: Architecture dependent (12KB for IA-32)

This specifies the switch point between eager and rendezvous protocol in MVAPICH.

9.3.6 VIADEV_MAX_RDMA_SIZE

- Class: Run time
- Default: 1048576

Maximum size of an RDMA put message (RPUT) in the rendezvous protocol. Note that this variable should be set in bytes.

9.3.7 VIADEV_R3_THRESHOLD

- Class: Run time
- Default: 2048

This is the message size (in bytes) which will be sent using the R3 mode if the registration cache is turned on, i.e. the default setting here VIADEV_USE_DREG_CACHE=1

9.3.8 VIADEV_R3_NOCACHE_THRESHOLD

- Class: Run time
- Default: 1048576

This is the message size (in bytes) which will be sent using the R3 mode if the registration cache is turned off, i.e. VIADEV_USE_DREG_CACHE=0

9.3.9 VIADEV_VBUF_POOL_SIZE

- Class: Run time
- Default: 5000

The number of vbufs in the initial pool. This pool is shared among all the connections.

9.3.10 VIADEV_VBUF_SECONDARY_POOL_SIZE

- Class: Run time
- Default: 500

The number of vbufs allocated each time when the global pool is running out in the initial pool. This is also shared among all the connections.

9.3.11 VIADEV_USE_DREG_CACHE

- Class: Run time
- Default: 1

This indicates whether registration cache is to be used or not. The registration cache speeds up zero copy operations if user memory is re-used many times.

9.3.12 LAZY_MEM_UNREGISTER

- Class: Compile time
- Default: Set

Memory registration cache will be used if this flag is defined.

9.3.13 VIADEV_NDREG_ENTRIES

- Class: Run time
- Default: 1000

This defines the total number of buffers that can be stored in the registration cache. A larger value will lead to more infrequent lazy de-registration.

9.3.14 VIADEV_DREG_CACHE_LIMIT

- Class: Run time
- Default: No limit

This sets a limit on the number of pages kept registered by the registration cache. If you set it to 0, that implies no limits on the number of pages registered.

9.3.15 VIADEV_VBUF_MAX

- Class: Run time
- Default: -1 (No limit)

Max (total) number of VBUFs to allocate after which the process terminates with a fatal error. -1 means no limit.

9.3.16 VIADEV_ON_DEMAND_THRESHOLD

- Class: Run time
- Default: 32

Number of processes beyond which on-demand connection management will be used.

9.3.17 VIADEV_MAX_INLINE_SIZE

- Class: Run time
- Default: 128

Maximum size of a message (in bytes) that may be sent INLINE with message descriptor Lowering this increases message latency, but can lower memory requirements. Also see VIADEV_NO_INLINE_THRESHOLD, which will override this value in some cases.

9.3.18 VIADEV_NO_INLINE_THRESHOLD

- Class: Run time
- Default: 256

This parameter automatically changes the VIADEV_MAX_INLINE_SIZE after the number of connections exceeds VIADEV_NO_INLINE_THRESHOLD. Behavior is slightly different depending on whether on-demand connection setup is used:

- If the number of processes in a job is less than VIADEV_ON_DEMAND_THRESHOLD, then the maximum inline size for all connections is automatically set to zero to save memory.
- If the number of processes is greater than VIADEV_ON_DEMAND_THRESHOLD, then the first VIADEV_NO_INLINE_THRESHOLD number of connections per process have an inline size of VIADEV_MAX_INLINE_SIZE and all subsequent connections have a maximum inline size of zero.

9.3.19 VIADEV_USE_BLOCKING

- Class: Run time
- Default: 0

Use blocking mode progress, instead of polling. This allows MPI to yield CPU to other processes if there are no more incoming messages.

9.3.20 VIADEV_ADAPTIVE_RDMA_LIMIT

- Class: Run Time
- Default: Number of processes in application

This is the maximum number of RDMA paths that will be established in the entire MPI application. Passing it a value -1 implies that at most $log(n)$ number of paths will be established. Where n is the number of processes in the MPI application.

9.3.21 VIADEV_ADAPTIVE_RDMA_THRESHOLD

- Class: Run Time
- Default: 10

This is the number of messages exchanged per connection after which the RDMA path is established.

9.3.22 VIADEV_ADAPTIVE_ENABLE_LIMIT

- Class: Run Time
- Default: 32

Default value: Number of processes (np) in application If the number of jobs exceeds this limit, adaptive flow will be enabled. To enable adaptive flow for any number of jobs define: VIADEV_ADAPTIVE_ENABLE_LIMIT=0

9.3.23 VIADEV_SQ_SIZE

- Class: Run time
- Default: 40

To control the number of allowable outstanding send operations to the device.

9.4 Send/Receive Control Parameters

9.4.1 VIADEV_CREDIT_PRESERVE

- Class: Run time
- Default: 100

This parameter records the number of credits per connection that will be preserved for non-data, control packets. If SRQ is not used, this default is 10.

9.4.2 VIADEV_CREDIT_NOTIFY_THRESHOLD

- Class: Run time
- Default: 5

Flow control information is usually sent via piggybacking with other messages. This parameter is used, along with VIADEV_DYNAMIC_CREDIT_THRESHOLD, to determine when to send explicit flow control update messages.

9.4.3 VIADEV_DYNAMIC_CREDIT_THRESHOLD

- Class: Run time
- Default: 10

Flow control information is usually sent via piggybacking with other messages. These two parameters are used to determine when to send explicit flow control update messages.

9.4.4 VIADEV_INITIAL_PREPOST_DEPTH

- Class: Run time
- Default: 5

This defines the initial number of pre-posted receive buffers for each connection. If communication happen for a particular connection, the number of buffers will be increased to VIADEV_PREPOST_DEPTH.

9.4.5 VIADEV_USE_SHARED_MEM

- Class: Run time
- Default: 1

When _SMP_ is defined, shared memory communication can be disabled by setting VIADEV_USE_SHARE

9.4.6 VIADEV_PROGRESS_THRESHOLD

- Class: Run time
- Default: 1

This value determines if additional MPI progress engine calls are made when making send operations. If there are this number or more queued send operations then progress is attempted.

9.4.7 VIADEV_USE_COALESCE

- Class: Run time
- Default: 1

This setting turns on (1) or off (0) the coalescing of messages. Leaving feature on can help applications that make many consecutive send operations to the same host.

9.4.8 VIADEV_USE_COALESCE_SAME

- Class: Run time
- Default: 0

If VIADEV_USE_COALESCE is enabled, this flag will enable coalescing only for messages of the same tag, communicator, and size. This also increases VIADEV_PROGRESS_THRESHOLD to 2.

9.4.9 VIADEV_COALESCE_THRESHOLD_SQ

- Class: Run time
- Default: 4

If there are more than this number of small messages outstanding to a another task, messages will be coalesced until one of the previous sends completes.

9.4.10 VIADEV_COALESCE_THRESHOLD_SIZE

- Class: Run time
- Default: VIADEV_VBUF_TOTAL_SIZE

Attempt to coalesce messages under this size. If this number is greater than VIADEV_VBUF_TOTAL_SIZE, then it is set to VIADEV_VBUF_TOTAL_SIZE. This has no effect if message coalescing is turned off.

9.5 SRQ (Shared Receive Queue) Control Parameters

9.5.1 VIADEV_USE_SRQ

- Class: Run Time
- Default: 1

Indicates whether Shared Receive Queue is to be used or not. Users are strongly encouraged to use this as long as the InfiniBand software/hardware supports this feature.

9.5.2 VIADEV_SRQ_MAX_SIZE

- Class: Run Time
- Default: 4096

This is the maximum number of work requests allowed on the Shared Receive Queue. Upon receiving a SRQ limit event, the current value of `VIADEV_SRQ_SIZE` will be doubled or moved to the maximum of `VIADEV_SRQ_MAX_SIZE`, whichever is smaller.

9.5.3 VIADEV_SRQ_SIZE

- Class: Run Time
- Default: 256

This is the maximum number of work requests posted to the Shared Receive Queue initially. This value will dynamically re-size up to `VIADEV_SRQ_MAX_SIZE`.

9.5.4 VIADEV_SRQ_LIMIT

- Class: Run Time
- Default: 30

This is the low watermark limit for the Shared Receive Queue. If the number of available work entries on the SRQ drops below this limit, the flow control will be activated.

9.5.5 VIADEV_MAX_R3_OUST_SEND

- Class: Run Time
- Default: 32

This is the maximum number of R3 packets which are outstanding when using Shared Receive Queues.

9.5.6 VIADEV_SRQ_ZERO_POST_MAX

- Class: Run Time
- Default: 1

Maximum number of unsuccessful SRQ posts that an async thread can make before going to sleep.

9.5.7 VIADEV_MAX_R3_PENDING_DATA

- Class: Run Time
- Default: 524288

This is the maximum amount of R3 data that is sent out un-acked

9.6 Shared Memory Control Parameters

9.6.1 VIADEV_SMP_EAGERSIZE

- Class: Run time
- Default: Architecture dependent (32KB for EM64T)

This has no effect if macro _SMP_ is not defined. It defines the switch point from Eager protocol to Rendezvous protocol for intra-node communication. If macro _SMP_RNDV_ is defined, then for messages larger than VIADEV_SMP_EAGERSIZE, SMP Rendezvous protocol is used. Note that this variable should be set in KBytes.

9.6.2 VIADEV_SMPI_LENGTH_QUEUE

- Class: Run time
- Default: Architecture dependent (128KB for EM64T)

This has no effect if macro _SMP_ is not defined. It defines the size of shared buffer between every two processes on the same node for transferring messages smaller than or equal to VIADEV_SMP_EAGERSIZE. Note that this variable should be set in KBytes.

9.6.3 SMP_SEND_BUF_SIZE

- Class: Compile time
- Default: Architecture dependent (8KB for EM64T)

This has no effect if macro _SMP_ is not defined. It defines the packet size when sending intra-node messages larger than VIADEV_SMP_EAGERSIZE. Note that this variable should be set in Bytes.

9.6.4 VIADEV_SMP_NUM_SEND_BUFFER

- Class: Run time
- Default: Architecture dependent (128 for EM64T)

This has no effect if macro _SMP_ is not defined. It defines the number of internal send buffers for sending intra-node messages larger than VIADEV_SMP_EAGERSIZE.

9.6.5 VIADEV_USE_AFFINITY

- Class: Run time
- Default value: 1

Enable CPU affinity by setting VIADEV_USE_AFFINITY=1 or disable it by setting VIADEV_USE_AFFIN
VIADEV_USE_AFFINITY does not take effect when _AFFINITY_ is not defined.

9.6.6 VIADEV_CPU_MAPPING

- Class: Run time
- Default value: Local rank based mapping

User can specify process to CPU mapping within a node. This may help the applications to get the best performance on multi-core systems. For example, if we set VIADEV_CPU_MAPPING=0:4:1:5 or VIADEV_CPU_MAPPING=0,4,1,5, then process 0 on each node will be mapped to CPU 0, process 1 will be mapped to CPU 4, process 2 will be mapped to CPU 1, and process 3 will be mapped to CPU 5. The CPU numbers can be separated by either a single ":" or ",". This parameter does not take effect when _AFFINITY_ is not defined or VIADEV_USE_AFFINITY is set to 0.

9.7 Run time parameters for Collectives

9.7.1 VIADEV_USE_SHMEM_COLL

- Class: Run time
- Default: 1

To disable shmem based collectives, set this to 0.

9.7.2 VIADEV_USE_SHMEM_BARRIER

- Class: Run time
- Default: 1

To disable shmem based Barrier, set this to 0.

9.7.3 VIADEV_USE_SHMEM_ALLREDUCE

- Class: Run time
- Default: 1

To disable shmem based Allreduce, set this to 0.

9.7.4 VIADEV_USE_SHMEM_REDUCE

- Class: Run time
- Default: 1

To disable shmem based Reduce, set this to 0.

9.7.5 VIADEV_USE_ALLGATHER_NEW

- Class: Run time
- Default: 1

To disable the new Allgather, set this to 0.

9.7.6 VIADEV_MAX_SHMEM_COLL_COMM

- Class: Run time
- Default: 16

This parameter allows to configure the number of communicators using shared memory collectives.

9.7.7 VIADEV_SHMEM_COLL_MAX_MSG_SIZE

- Class: Run time
- Default: $1 \ll 20$

This parameter allows the maximum message to be tuned for the shared memory collectives.

9.7.8 VIADEV_SHMEM_COLL_REDUCE_THRESHOLD

- Class: Run time
- Default: $1 \ll 10$

The shmem reduce is taken for messages less than this threshold. This threshold can be tuned appropriately but should be less than that of 9.7.7 above.

9.7.9 VIADEV_SHMEM_COLL_ALLREDUCE_THRESHOLD

- Class: Run time
- Default: $1 \ll 15$

The shmem allreduce is taken for messages less than this threshold. This threshold can be tuned appropriately but should be less than that of 9.7.7 above.

9.7.10 VIADEV_SHMEM_COLL_BCAST_THRESHOLD

- Class: Run time
- Default: $1 \ll 23$

The shmem broadcast is taken for messages less than this threshold.

9.7.11 VIADEV_SHMEM_BCAST_LEADERS

- Class: Run time
- Default: 4096

The number of leader processes that will take part in the shmem broadcast operation.

9.7.12 VIADEV_BCAST_KNOMIAL

- Class: Run time
- Default: 4

To control the degree k of the k-nomial Broadcast algorithm. It should always be an integer greater than or equal to 2.

9.7.13 MPIR_ALLTOALL_SHORT_MSG

- Class: Run time
- Default : 8192

9.7.14 MPIR_ALLTOALL_MEDIUM_MSG

- Class: Run time
- Default : 8192

9.7.15 MPIR_AllTOALL_BASIC

- Class: Run time
- Default: 0

Turning this option on sets the MPIR_ALLTOALL_SHORT_MSG to 256 and MPIR_ALLTOALL_MEDIUM_MSG to 32768. This setting is for dual node clusters. This parameter is not present for PSM device.

9.7.16 MPIR_ALLTOALL_MCORE_OPT

- Class: Run time
- Default: 1

Turning this option on sets the MPIR_ALLTOALL_SHORT_MSG to 8192 and MPIR_ALLTOALL_MEDIUM_MSG to 8192. This setting is for multi-core clusters. This parameter is not present for PSM device.

9.8 CM Control Parameters

9.8.1 VIADEV_CM_RECV_BUFFERS

- Class: Run time
- Default: 1024

To control the number of receive buffers dedicated to UD based connection manager. Each buffer is only several tens of bytes.

9.8.2 VIADEV_CM_MAX_SPIN_COUNT

- Class: Run time
- Default: 5000

9.8.3 VIADEV_CM_TIMEOUT

- Class: Run time
- Default: 500 ms

To control the timeout value for UD messages.

9.9 Fault-tolerance Parameters

9.9.1 VIADEV_USE_APM

- Class: Run Time
- Applicable device(s): Gen2

This parameter is used for recovery from network faults using Automatic Path Migration. This functionality is beneficial in the presence of multiple paths in the network, which can be enabled by using LMC mechanism.

9.9.2 VIADEV_USE_APM_TEST

- Class: Run Time
- Applicable device(s): Gen2

This parameter is used for testing the Automatic Path Migration functionality. It periodically moves the alternate path as the primary path of communication and re-loads another alternate path.

9.9.3 VIADEV_USE_NFR

- Class: Run Time
- Applicable device(s): Gen2

This parameter is used to control the use of Network Fault Recovery functionality. This feature is turned off by default. To turn on this feature, set environment variable to 1.

9.10 Other Parameters

9.10.1 VIADEV_CLUSTER_SIZE

- Class: Run time
- Default: Small

This controls the preset values for vbuf size, number of RDMA buffers and Rendezvous threshold for various cluster sizes. It can be set to “SMALL” (1-64), “MEDIUM” (64-256) and “LARGE” (256 and beyond). In addition, there is an “AUTO” option which will automatically set the appropriate parameters based on number of processes in the MPI application.

9.10.2 VIADEV_PREPOST_DEPTH

- Class: Run time
- Default: 64

This defines the number of buffers pre-posted for each connection to handle send/receive operations.

9.10.3 VIADEV_MAX_SPIN_COUNT

- Class: Run time
- Default: 1000

This parameter is only effective when blocking mode progress is used. This parameter indicates the number of polls made by MVAPICH before yielding the CPU to other applications.

10 MVAPICH Gen2-Hybrid Parameters

10.1 InfiniBand HCA and Network Parameters

10.1.1 MV_DEVICE

- Default: First IB device found on the system

Name of the InfiniBand device. e.g. `mthca0`, `mthca1`.

10.1.2 MV_MTU

- Default: Maximum detected MTU for the selected HCA

MTU size in bytes that should be used (e.g. 1024, 2048, 4096). Must be less than or equal to the value supported by the HCA.

10.2 Transport and Connection Options

MVAPICH-Hybrid supports running UD, RC, and XRC transports. Additionally, it supports an RDMA fast path (RCFP) for very low latency over RC and XRC transports. These parameters allow changing the default layers.

10.2.1 MV_CONN_TYPE

- Default: RC_SRQ

What connection type should be used for the reliable channel.

- RC_SRQ: Single RC connection between processes with a Shared Receive Queue (SRQ)
- RC_RQ: Single RC connection with flow control and non-shared receive queue. Should only be used on older HCAs that do not support SRQ
- XRC_SINGLE_MULT_SRQ: Single XRC connection between each set of processes. Allows multiple SRQs of different sizes.
- XRC_SHARED_MULT_SRQ: Shared XRC connections between processes on the same node. Allows multiple SRQs of different sizes. More scalable than XRC_SINGLE_MULT_SRQ.

- XRC_SINGLE_SRQ: Single XRC connection between each set of processes. Allows single SRQs of fixed size.
- XRC_SHARED_SRQ: Shared XRC connections between processes on the same node. Allows single SRQs of fixed size.

10.2.2 MV_MAX_RC_CONNECTIONS

- Default: 16

Maximum number of RC or XRC connections created per process. This limits the amount of connection memory and prevents HCA cache thrashing.

10.2.3 MV_MAX_RCFP_CONNECTIONS

- Default: 8

Maximum number of RC/XRC Fast Path communication channels. These are low latency channels, however, each one consumes memory and too many can increase polling time.

10.2.4 MV_NUM_UD_QPS

- Default: 4

How many UD QPs should be created and used (in round-robin) for message transfer? Often more than one is required to get full bandwidth.

10.3 Reliability Parameters

Reliability is always enabled for UD messages be turned off since UD is an unreliable transport. The following are various options to tune it. These do not affect RC or XRC transports.

10.3.1 MV_PROGRESS_TIMEOUT

- Default: 1400000 (1.4 sec)

Time (usec) until ACK status is checked (and ACKs are sent if needed)

10.3.2 MV_RETRY_TIMEOUT

- Default: 20000000 (2.0 sec)

Time (usec) after which an unacknowledged message will be retried

10.3.3 MV_MAX_RETRY_COUNT

- Default: 50

Number of retries of a message before the job is aborted. This is needed in case an HCA fails.

10.3.4 MV_ACK_AFTER_RECV

- Default: 50

After this number of messages is received an ACK is sent back to the sender – regardless of MV_PROGRESS_TIMEOUT.

10.3.5 MV_ACK_AFTER_PROGRESS

- Default: 10

After a message receive is detected and before control is returned to the application, how many messages can be received before an ACK is transmitted to the sender – regardless of MV_PROGRESS_TIMEOUT.

10.4 Large Message Transfer Parameters

10.4.1 MV_USE_UD_ZCOPY

- Default: 1

Whether or not to use the zero-copy transfer mechanism to transfer large messages.

10.4.2 MV_UD_ZCOPY_QPS

- Default: 64

How many zero-copy large message transfers can be currently outstanding to a single process.

10.4.3 MV_UD_ZCOPY_THRESHOLD

- Default: 32768

Messages of this size and above should be transmitted along the zero-copy path
(if MV_USE_UD_ZCOPY is set)

10.4.4 MV_USE_REG_CACHE

- Default: 1

Whether buffer registrations should be cached in the MPI library to increase performance

10.5 Performance and General Parameters

10.5.1 MV_RNDV_THRESHOLD

- Default: 16384

For messages over this size the sender should verify that the receive has been posted before sending the message.

10.5.2 MV_RC_SEND_THRESHOLD

- Default: 2000

For messages over this size use the RC or XRC channel if available.

10.5.3 MV_RC_BUF_SIZE

- Default: 8192

Size of messages posted to the SRQ or RQ (whenever RC_SRQ or XRC_SHARED_SRQ or XRC_SINGLE_SRQ

10.5.4 MV_RCFP_BUF_SIZE

- Default: 2048

Size of messages (in bytes) allowed over the RC/XRC Fast Path channel.

10.5.5 MV_RCFP_BUF_COUNT

- Default: 32

The number of buffers in each RC/XRC Fast Path channel

10.6 QP and Buffer Parameters

10.6.1 MV_UD_SQ_SIZE

- Default:

How many send operations can be outstanding at any given time

10.6.2 MV_UD_RQ_SIZE

- Default:

Maximum number of receive buffers that can be posted at a single time

10.6.3 MV_UD_CQ_SIZE

- Default:

Maximum number of completions that can be expected. Generally set to MV_UD_SQ_SIZE + MV_UD_RQ_SIZE.

10.6.4 MV_RC_SQ_SIZE

- Default: 128

Maximum number of concurrent send operations allowed per QP for RC or XRC QPs.

10.6.5 MV_SRQ_SIZE

- Default: 512

How many buffers are posted at any one time for eager receives.

10.6.6 MV_USE_LMC

- Default: 1

If the LID Mask Count (LMC) value is above 0, if multiple paths be used through the network

10.7 Shared Memory Control Parameters

10.7.1 MV_USE_SHARED_MEMORY

- Default: 1 (on)

Whether or not shared memory should be used for communication with peers on the same node (instead of network loop back)

10.7.2 MV_SMP_EAGERSIZE

- Class: Run time
- Default: Architecture dependent (32KB for EM64T)

This has no effect if macro _SMP_ is not defined. It defines the switch point from Eager protocol to Rendezvous protocol for intra-node communication. If macro _SMP_RNDV_ is defined, then for messages larger than MV_SMP_EAGERSIZE, SMP Rendezvous protocol is used. Note that this variable should be set in KBytes.

10.7.3 MV_SMPI_LENGTH_QUEUE

- Class: Run time
- Default: Architecture dependent (128KB for EM64T)

This has no effect if macro _SMP_ is not defined. It defines the size of shared buffer between every two processes on the same node for transferring messages smaller than or equal to MV_SMP_EAGERSIZE. Note that this variable should be set in KBytes.

10.7.4 SMP_SEND_BUF_SIZE

- Class: Compile time
- Default: Architecture dependent (8KB for EM64T)

This has no effect if macro _SMP_ is not defined. It defines the packet size when sending intra-node messages larger than MV_SMP_EAGERSIZE. Note that this variable should be set in Bytes.

10.7.5 MV_SMP_NUM_SEND_BUFFER

- Class: Run time
- Default: Architecture dependent (128 for EM64T)

This has no effect if macro _SMP_ is not defined. It defines the number of internal send buffers for sending intra-node messages larger than MV_SMP_EAGERSIZE.

10.7.6 MV_USE_AFFINITY

- Class: Run time
- Default value: 1

Enable CPU affinity by setting MV_USE_AFFINITY=1 or disable it by setting MV_USE_AFFINITY=0. MV_USE_AFFINITY does not take effect when _AFFINITY_ is not defined.