MVAPICH2 1.8 User 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: July 28, 2012

Contents

1 Overview of the MVAPICH Project

InfiniBand, 10GigE/iWARP and 10/40GigE RDMA over Converged Ethernet (RoCE) are emerging as high-performance networking technologies to deliver low latency and high bandwidth. They are also achieving widespread acceptance due to their *open standards*.

MVAPICH (pronounced as "em-vah-pich") is an *open-source* MPI software to exploit the novel features and mechanisms of these networking technologies and deliver best 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 library: 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, InfiniBand, 10GigE/iWARP and 10/40GigE RoCE communities. As of July 2012, more than 1,930 organizations (National Labs, Universities and Industry) world-wide (in 68 countries) have registered as MVAPICH/MVAPICH2 users at MVAPICH project web site. There have also been more than 120,000 downloads of MVAPICH/MVAPICH2 software from the MVAPICH project site directly. In addition, many InfiniBand, 10GigE/iWARP and 10/40GigE RoCE vendors, server vendors, systems integrators and Linux distributors have been incorporating MVAPICH/MVAPICH2 into their software stacks and distributing it. MVAPICH and MVAPICH2 are also available with the Open Fabrics Enterprise Distribution (OFED) stack. Both MVAPICH and MVAPICH2 distributions are available under BSD licensing.

Several InfiniBand systems using MVAPICH/MVAPICH2 have obtained positions in the TOP 500 ranking. The November '11 list includes the following systems: 5th ranked Tsubame 2 at Tokyo University of Technology with 73,278-cores; 7th ranked Pleiades at NASA with 111,104-cores; 25th ranked Ranger at Texas Advanced Computing Center (TACC) with 62,976 cores; 39th ranked Lonestar at Texas Advanced Computing Center (TACC) with 22,656 cores; and 48th ranked Gordon at San Diego Supercomputer Center (SDSC) with 12,608-cores.

More details on MVAPICH/MVAPICH2 software, users list, mailing lists, sample performance numbers on a wide range of platforms and interconnects, a set of OSU benchmarks, related publications, and other InfiniBand- and iWARP-related projects (parallel file systems, storage, data centers) can be obtained from our website:http://mvapich.cse.ohio-state.edu.

This document contains necessary information for MVAPICH2 users to download, install, test, use, tune and troubleshoot MVAPICH2 1.8. We continuously fix bugs and update update this document as per user feedback. Therefore, we strongly encourage you to refer 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 MVAPICH2 1.8.

In Section 3 we describe all the features in MVAPICH2 1.8. As you read through this section, please note our new features (highlighted as NEW) compared to version 1.7. 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. Basic usage of MVAPICH2 is explained in Section 5. Section 6 provides instructions for running MVAPICH2 with some of the advanced features. Section 7 describes the usage of the OSU Benchmarks. In Section 8 we suggest some tuning techniques for multi-thousand node clusters using some of our new features. If you have any problems using MVAPICH2, please check Section 9 where we list some of the common problems people face. Finally, in Sections 11 and 12, we list all important run time parameters, their default values and a short description.

3   MVAPICH2 1.8 Features

MVAPICH2 (MPI-2 over InfiniBand) is an MPI-2 implementation based on MPICH2 ADI3 layer. It also supports all MPI-1 functionalities. MVAPICH2 1.8 is available as a single integrated package (with MPICH2 1.4.1p1). This version is compliant with the latest MPI 2.2 standard. The current release supports ten different underlying transport interfaces, as shown in Figure 1.
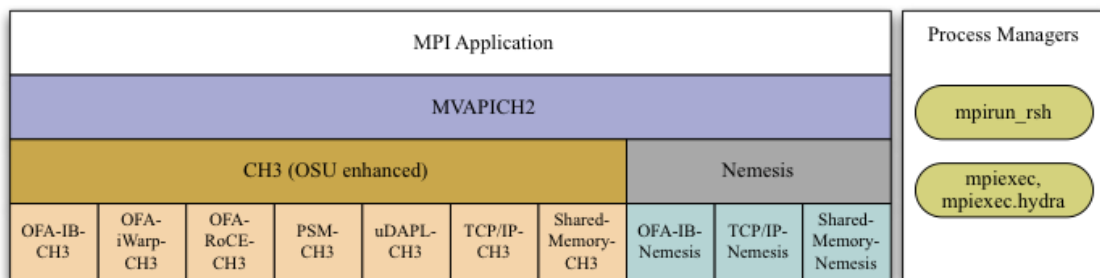
Figure 1: Overview of different available interfaces of the MVAPICH2 library

- OFA-IB-CH3: This interface supports all InfiniBand compliant devices based on the OpenFabrics layer. This interface has the most features and is most widely used. For example, this interface can be used over all Mellanox InfiniBand adapters, IBM eHCA adapters and QLogic adapters.

- OFA-IB-Nemesis: This interface supports all InfiniBand compliant devices based on the OpenFabrics layer with the emerging Nemesis channel of the MPICH2 stack. This interface can be used by all Mellanox InfiniBand adapters.

- OFA-iWARP-CH3: This interface supports all iWARP compliant devices supported by OpenFabrics. For example, this layer supports Chelsio T3 adapters with the native iWARP mode.

- OFA-RoCE-CH3: This interface supports the emerging RoCE (RDMA over Converged Ethernet) interface for Mellanox ConnectX-EN adapters with 10/40GigE switches.

- PSM-CH3: This interface provides native support for InfiniPath adapters from QLogic over PSM interface. It provides high-performance point-to-point communication for both one-sided and two-sided operations.

- uDAPL-CH3: This interface supports all network-adapters and software stacks which implement the portable DAPL interface from the DAT Collaborative. For example, this interface can be used over all Mellanox adapters, Chelsio adapters and NetEffect adapters.

- Shared-Memory-CH3: This interface provides native shared memory support on multi-core platforms where communication is required only within a node. Such as SMP-only systems, laptops, etc.

- TCP/IP-CH3: The standard TCP/IP interface (provided by MPICH2) to work with a range of network adapters supporting TCP/IP interface. This interface can be used with IPoIB (TCP/IP over InfiniBand network) support of InfiniBand also. However, it will not deliver good performance/scalability as compared to the other interfaces.

- TCP/IP-Nemesis: The standard TCP/IP interface (provided by MPICH2 Nemesis channel) to work with a range of network adapters supporting TCP/IP interface. This interface can be used with IPoIB (TCP/IP over InfiniBand network) support of InfiniBand also. However, it will not deliver good performance/scalability as compared to the other interfaces.

- Shared-Memory-Nemesis: This interface provides native shared memory support on multi-core platforms where communication is required only within a node. Such as SMP-only systems, laptops, etc.

MVAPICH2 1.8 delivers better performance (especially with one-copy intra-node communication support with LiMIC2) compared to MVAPICH 1.2, the latest release package of MVAPICH supporting MPI-1 standard. MVAPICH2 1.8 is compliant with MPI 2.2 standard. In addition, MVAPICH2 1.8 provides support and optimizations for other MPI-2 features, NVIDIA GPU, multi-threading and fault-tolerance (Checkpoint-restart, Job-pause-migration-resume). A complete set of features of MVAPICH2 1.8 are indicated below. New features compared to 1.7 are indicated as (NEW).

- MPI-2.2 standard compliance

- Based on MPICH2 1.4.1p1

- OFA-IB-Nemesis interface design

  - OpenFabrics InfiniBand network module support for MPICH2 Nemesis modular design
  - Support for high-performance intra-node shared memory communication provided by the Nemesis design
  - Adaptive RDMA fast path with Polling Set for high-performance inter-node communication
  - Tuned RDMA Fast Path buffer size to get better performance with less memory footprint
  - Optimization to limit number of RDMA Fast Path connections for very large clusters
  - Shared Receive Queue (SRQ) support with flow control, uses significantly less memory for MPI library
  - Header caching for low-latency
  - 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 heavily use malloc and free operations
- Support for TotalView debugger
- Shared Library Support for existing binary MPI application programs to run
- ROMIO Support for MPI-IO
- Support for additional features (such as hwloc, hierarchical collectives, one-sided, multithreading, etc.), as included in the MPICH2 1.4.1p1 Nemesis channel
- Support of Shared-Memory-Nemesis interface on multi-core platforms requiring intra-node communication only (SMP-only systems, laptops, etc.)
- Support for 3D torus topology with appropriate SL settings
- Quality of Service (QoS) support with multiple InfiniBand SL
- Automatic inter-node communication parameter tuning based on platform and adapter detection
- (NEW) Flexible HCA selection
- (NEW) Checkpoint-Restart support
- (NEW) Run-through stabilization support to handle process failures
- (NEW) Enhancements to handle IB errors gracefully

- Flexible process manager support

  - mpirun_rsh to work with any of the ten interfaces (CH3 and Nemesis channel-based) including OFA-IB-Nemesis, TCP/IP-CH3 and TCP/IP-Nemesis
  - Compact and shorthand way to specify blocks of processes on the same host with mpirun_rsh
  - Hydra process manager to work with any of the ten interfaces (CH3 and Nemesis channel-based) including OFA-IB-CH3, OFA-IWARP-CH3, OFA-RoCE-CH3 and TCP/IP-CH3
  - XRC support with Hydra Process Manager
  - Improved debug message output in process management and fault tolerance functionality
  - Better handling of process signals and error management in mpispawn
  - Enhanced mpirun_rsh design to avoid race conditions and support for improved debug messages

- Support for various job launchers and job schedulers (such as SGE and OpenPBS/Torque)

- CH3 shared memory channel for standalone hosts (including SMP-only systems and laptops) without any InfiniBand adapters

- CH3-Level design for scaling to multi-thousand nodes with highest performance and reduced memory usage

  - HugePage support
  - Integrated Hybrid (UD-RC/XRC) design to get best performance on large-scale systems with reduced/constant memory footprint
  - (NEW) Support for running with UD only mode

- – Support for MPI-2 Dynamic Process Management on InfiniBand clusters
- – eXtended Reliable Connection (XRC) support
    * (NEW) Enable XRC by default at configure time
- – Multiple CQ-based design for Chelsio 10GigE/iWARP
- – Multi-port support for Chelsio 10GigE/iWARP
- – Enhanced iWARP design for scalability to higher process count
- – (NEW) Support iWARP interoperability between Intel NE020 and Chelsio T4 Adapters
- – Support for 3D torus topology with appropriate SL settings
- – Quality of Service (QoS) support with multiple InfiniBand SL
- – Scalable and robust daemon-less job startup
    * Enhanced and robust mpirun_rsh framework to provide scalable job launching on multi-thousand core clusters
    * Hierarchical ssh to nodes to speedup job start-up
    * MPMD job launch capability
    * Available for OFA-IB-CH3, OFA-iWARP-CH3, OFA-RoCE-CH3 and uDAPL-CH3 interfaces.
- – Optimization to limit number of RDMA Fast Path connections for very large clusters
- – Tuned RDMA Fast Path buffer size to get better performance with less memory footprint
- – (NEW) Optimization in buffer usage to achieve lesser memory footprint
- – On-demand Connection Management: This feature enables InfiniBand connections to be setup dynamically, enhancing the scalability of MVAPICH2 on clusters of thousands of nodes
    * Improved on-demand InfiniBand connection setup
    * On-demand connection management support with IB CM (RoCE Interface)
    * Native InfiniBand Unreliable Datagram (UD) based asynchronous connection management for OF-IB-CH3 interface
    * RDMA CM based on-demand connection management for OFA-iWARP-CH3 and OFA-IB-CH3 interfaces
    * uDAPL on-demand connection management based on standard uDAPL interface
- – 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. Available for OFA-IB-CH3 interface
- – Hot-Spot Avoidance Mechanism (HSAM) for alleviating network congestion in large scale clusters. Available for OFA-IB-CH3 interface
- – RDMA Read utilized for increased overlap of computation and communication for OpenFabrics device. Available for OFA-IB-CH3 and OFA-iWARP-CH3 interfaces
- – Shared Receive Queue (SRQ) with flow control. This design uses significantly less memory for MPI library. Available for OFA-IB-CH3 interface.
- – Adaptive RDMA Fast Path with Polling Set for low-latency messaging. Available for OFA-IB-CH3 and OFA-iWARP-CH3 interfaces.

- Enhanced scalability for RDMA-based direct one-sided communication with less communication resource. Available for OFA-IB-CH3 and OFA-iWARP-CH3 interfaces.

- Supporting Large Data transfers (>2GB)

- (NEW) Support for fallback to R3 rendezvous protocol if RGET fails

- (NEW) Support for MPI communication from NVIDIA GPU device memory

  - High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)

  - High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)

  - Enhanced designs for Alltoall and Allgather collective communication from GPU device buffers

  - Optimized and tuned support for collective communication from GPU buffers

  - Non-contiguous datatype support in Pt-Pt and Collective communication from GPU buffers

  - Taking advantage of CUDA IPC (available in CUDA 4.1) in intra-node communication for multiple GPU adapters/node

  - Efficient synchronization mechanism using CUDA Events for pipelined device data transfers

- Dynamic Process Management (DPM). Available for OFA-IB-CH3 interface.

- Configuration file support (similar to the one available in MVAPICH). Provides a convenient method for handling all runtime variables through a configuration file.

- Fault tolerance support

  - Checkpoint-restart support for application transparent systems-level fault tolerance. BLCR-based support using OFA-IB-CH3 and OFA-IB-Nemesis interfaces.
    * Scalable Checkpoint-restart with mpirun_rsh framework
    * Scalable Checkpoint-restart with Fault Tolerance Backplane (FTB) framework (FTB-CR)
    * Checkpoint-restart with intra-node shared memory (user-level) support
    * Checkpoint-restart with intra-node shared memory (kernel-level with LiMIC2) support
    * Checkpoint-restart support with pure SMP mode
    * Allows best performance and scalability with fault-tolerance support
    * (NEW) Run-through stabilization support to handle process failures using OFA-IB-Nemesis interface
    * (NEW) Enhancements to handle IB errors gracefully using OFA-IB-Nemesis interface

  - Application-initiated system-level checkpointing is also supported. User application can request a whole program checkpoint synchronously by calling special MVAPICH2 functions.
    * Flexible interface to work with different files system. Tested with ext3 (local disk), NFS, PVFS2 and Lustre

  - Network-Level fault tolerance with Automatic Path Migration (APM) for tolerating intermittent network failures over InfiniBand

- Fast Checkpoint-Restart support with aggregation scheme
- Job Pause-Migration-Restart Framework for Pro-active Fault-Tolerance
  * (NEW) Enable signal-triggered (SIGUSR2) migration
- Fast process migration using RDMA
- Support for new standardized Fault Tolerance Backplane (FTB) Events for Checkpoint-Restart and Job Pause-Migration-Restart Framework

- Enhancement to software installation

  - Automatically detects system architecture and adapter types and optimizes MVAPICH2 for any particular installation
  - An utility (mpiname) for querying the MVAPICH2 library version and configuration information
  - Automatically builds and installs OSU Benchmarks for end-user convenience

- Optimized intra-node communication support by taking advantage of shared-memory communication. Available for all interfaces.

  - (NEW) New shared memory design for enhanced intra-node small message performance
  - Kernel-level single-copy intra-node communication solution based on LiMIC2
    * Upgraded to LiMIC2 version 0.5.5 to support Intra-node large message (>2GB) transfers
    * LiMIC2 is designed and developed jointly by The Ohio State University and System Software Laboratory at Konkuk University, Korea.
  - Efficient Buffer Organization for Memory Scalability of Intra-node Communication
  - Multi-core optimized
  - (NEW) Adjust shared-memory communication block size at runtime
  - Automatic intra-node communication parameter tuning based on platform
  - Efficient connection set-up for multi-core systems
  - Portable Hardware Locality (hwloc) support for defining CPU affinity
  - (NEW) Integrated with Portable Hardware Locality (hwloc v1.4.1)
  - Efficient CPU binding policies (bunch and scatter) to specify CPU binding per job for modern multi-core platforms
  - (NEW) Enhanced support for CPU binding with socket and numanode level granularity
  - (NEW) Show current CPU bindings with MV2_SHOW_CPU_BINDING
  - Improved Bunch/Scatter mapping for process binding with HWLOC and SMT support
  - Improved usability of process to CPU mapping with support of delimiters (',' , '-') in CPU listing
  - Also allows user-defined flexible processor affinity
  - Optimized for Bus-based SMP and NUMA-Based SMP systems
  - Efficient support for disk less clusters

- Optimized collective communication operations. Available for all interfaces.

- Shared-memory aware K-nomial tree-based solution together with shared memory-based broadcast for scalable MPI_Bcast operations

- Shared-memory optimized algorithms and optimizations for barrier, reduce and all-reduce operations

- Performance improvements in Scatterv and Gatherv collectives for CH3 interface

- (NEW) Enhancements and optimizations for collectives (Bcast and Alltoallv)

- Integrated multi-rail communication support. Available for OFA-IB-CH3 and OFA-iWARP-CH3 interfaces.

    - Supports multiple queue pairs per port, and multiple ports per adapter

    - Multiple adapters

    - Support to selectively use some or all rails according to user specification

    - Support for both one-sided and point-to-point operations

    - Reduced stack size of internal threads to dramatically reduce memory requirement on multi-rail systems

    - Dynamic detection of multiple InfiniBand adapters and using these by default in multi-rail configurations (OFA-IB-CH3, OFA-iWARP-CH3 and OFA-RoCE-CH3 interfaces)

    - Support for process-to-rail binding policy (bunch, scatter and user-defined) in multi-rail configurations (OFA-IB-CH3, OFA-iWARP-CH3 and OFA-RoCE-CH3 interfaces)

- Support for InfiniBand Quality of Service (QoS) with multiple lanes

- Multi-threading support. Available for all interfaces, including TCP/IP.

    - Enhanced support for multi-threaded applications

- High-performance optimized and scalable support for one-sided communication: Put, Get and Accumulate. Supported synchronization calls: Fence, Active Target, Passive (lock and unlock). Available for all interfaces.

    - Direct RDMA based One-sided communication support for OFA-IB-CH3 and OFA-iWARP-CH3

    - Enhanced scalability for RDMA-based direct one-sided communication with less communication resource

    - Enhancement to the design of MPI_Win_complete for RMA operations

    - Removing the limitation on number of concurrent windows in RMA operations

    - (NEW) Enhanced one-sided communication design with reduced memory requirement

    - Using LiMIC2 for efficient intra-node RMA transfer to avoid extra memory copies

    - Optimized Fence synchronization (with and without LIMIC2 support)

    - Shared memory backed Windows for One-Sided Communication

    - Support for truly passive locking for intra-node RMA in shared memory and LIMIC based windows

- Two modes of communication progress

  - Polling

  - Blocking (enables running multiple MPI processes/processor). Available for OFA-IB-CH3 interface.

- Scalable job startup schemes

  - Enhanced and robust mpirun_rsh framework

  - Hierarchical ssh-based schemes to nodes

  - Flexibility for process execution with alternate group IDs

  - Ring-based startup for RoCE

  - Support for SLURM

  - (NEW) SLURM integration with mpiexec.mpirun_rsh to use SLURM allocated hosts without specifying a hostfile

  - (NEW) Support added to automatically use PBS_NODEFILE in Torque and PBS environments

  - (NEW) Support for suspend/resume functionality with mpirun_rsh framework

  - (NEW) Exporting local rank, local size, global rank and global size through environment variables (both mpirun_rsh and hydra)

- Advanced AVL tree-based Resource-aware registration cache

- Automatic tuning based on both platform type and network adapter type

- Progress loop optimization for PSM-CH3 interface

- Improved performance for medium sized messages for QLogic PSM

- Multi-core-aware collective support for QLogic PSM

- 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 heavily use `malloc` and `free` operations.

- High Performance and Portable Support for multiple networks and operating systems through uDAPL interface. Tested with: uDAPL over OpenFabrics on Linux.
  This uDAPL support is generic and can work with other networks that provide uDAPL interface. Please note that the stability and performance of MVAPICH2 with uDAPL depends on the stability and performance of the uDAPL library used. Starting from version 1.2, MVAPICH2 supports both uDAPL v2 and v1.2 on Linux.

- Support for TotalView debugger with mpirun_rsh framework.

- Shared Library Support for existing binary MPI application programs to run

- Enhanced debugging config options to generate core files and back-traces

- Use of gfortran as the default F77 compiler

- ROMIO Support for MPI-IO

  - Optimized, high-performance ADIO driver for Lustre

- Single code base for the following platforms (Architecture, OS, Compilers, Interfaces and InfiniBand adapters)

  - Architecture: (tested with) EM64T, x86_64 and x86

  - Operating Systems: (tested with) Linux

  - Compilers: (tested with) gcc, intel, pgi, Ekopath and Open64

  - InfiniBand adapters (tested with):

    * Mellanox InfiniHost adapters (SDR and DDR)
    * Mellanox ConnectX (DDR and QDR with PCIe2)
    * Mellanox ConnectX-2 (QDR with PCIe2)
    * Mellanox ConnectX-3 (FDR with PCIe3)
    * QLogic adapter (SDR)
    * QLogic adapter (DDR and QDR with PCIe2)

  - 10GigE (iWARP and RoCE) adapters:

    * (tested with) Chelsio T4 and T3 Adapters with iWARP support
    * (tested with) Mellanox ConnectX-EN 10GigE adapter
    * (tested with) Intel NE020 adapter with iWARP support

  - 40GigE RoCE adapters:

    * (tested with) Mellanox ConnectX-EN 40GigE adapter

The MVAPICH2 1.8 package and the project also includes the following provisions:

- Public SVN access of the code-base

- A set of micro-benchmarks (including multi-threading latency test) 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

The MVAPICH2 installation process is designed to enable the most widely utilized features on the target build OS by default. Supported operating systems include Linux and Solaris. The default interface is OFA-IB-CH3/OFA-IWARP-CH3 on Linux and uDAPL on Solaris. The other interfaces, as indicated in Figure 1, can also be selected on Linux. This installation section provides generic instructions for building from a tarball or our latest sources.

In order to obtain best performance and scalability while having flexibility to use a large number of features, the MVAPICH team strongly recommends the use of following interfaces for different adapters: 1) OFA-IB-CH3 interface for all Mellanox InfiniBand adapters, 2) PSM-CH3 interface for all QLogic Infini-Band adapters, 3) OFA-RoCE-CH3 interface for all RoCE adapters, 4) OFA-iWARP-CH3 for all iWARP adapters and 5) Shared-Memory-CH3 for single node SMP system and laptop.

Please see the appropriate subsection for specific configuration instructions for the interface-adapter you are targeting.

4.1   Building from a tarball

The MVAPICH2 1.8 source code package includes MPICH2 1.4.1p1. All the required files are present as a single tarball. Download the most recent distribution tarball from:
http://mvapich.cse.ohio-state.edu/download/mvapich2

Unpack the tarball and use the standard GNU procedure to compile:

```
$ tar -xzf mvapich2-1.8.tgz
$ cd mvapich2-1.8
$ ./configure
$ make
$ make install
```

In order to install a debug build, please use the following configuration option. *Please note that using debug builds may impact performance.*

```
$ ./configure --enable-g=all --enable-error-messages=all \
$ make
$ make install
```

4.2   Obtaining and Building the Source from SVN repository

These instructions assume you have already installed subversion.

The MVAPICH2 SVN repository is available at: https://mvapich.cse.ohio-state.edu/svn/mpi/mvapich2

Please keep in mind the following guidelines before deciding which version to check out:

- "tags/1.8" is the exact version released with no updates for bug fixes or new features.

    - To obtain the source code from tags/1.8:

```
$ svn co
https://mvapich.cse.ohio-state.edu/svn/mpi/mvapich2/tags/1.8
mvapich2
```

- "trunk" will contain the latest source code as we enhance and improve MVAPICH2. It may contain newer features and bug fixes, but is lightly tested.

    - To obtain the source code from trunk:
    ```
    $ svn co https://mvapich.cse.ohio-state.edu/svn/mpi/mvapich2/trunk
    mvapich2
    ```

The mvapich2 directory under your present working directory contains a working copy of the MVAPICH2 source code. Now that you have obtained a copy of the source code, you need to update the files in the source tree:

```
$ cd mvapich2
$ maint/updatefiles
```

This script will generate all of the source and configuration files you need to build MVAPICH2. You will need `autoconf` version 2.63.

```
$ ./configure
$ make
$ make install
```

4.3   Selecting a Process Manager

MVAPICH2 provides the mpirun_rsh/mpispawn framework from MVAPICH distribution. Using mpirun_rsh should provide the fastest startup of your MPI jobs. More details can be found in Section 5.2.1. In addition, MVAPICH2 also includes the Hydra process manager from MPICH2-1.3. For more details on using Hydra, please refer to Section 5.2.2.

By default, *mpiexec* uses the Hydra process launcher. Please note that neither mpirun_rsh, nor Hydra require you to start daemons in advance on the nodes used for a MPI job. Both mpirun_rsh and Hydra can be used with any of the eight interfaces of this MVAPICH2 release, as indicated in Figure 1.

4.3.1   Using SLURM

If you'd like to use slurm to launch your MPI programs please use the following configure options.

```
$ ./configure --with-pm=no --with-pmi=slurm
```

4.4   Configuring a build for OFA-IB-CH3/OFA-iWARP-CH3/OFA-RoCE-CH3

OpenFabrics (OFA) IB/iWARP/RoCE with the CH3 channel is the default interface on Linux. It can be explicitly selected by configuring with:

```
$ ./configure --with-device=ch3:mrail
--with-rdma=gen2
```

In order to use shared libraries configure as follows:

```
$ ./configure --with-device=ch3:mrail --with-rdma=gen2
--enable-shared
```

To enable use of the TotalView debugger, the library needs to be configured in the following manner:

```
$ ./configure --with-device=ch3:mrail --with-rdma=gen2
--enable-shared  --enable-g=dbg --enable-debuginfo
```

Configuration Options for OpenFabrics IB/iWARP/RoCE

- Configuring with Shared Libraries

  - Default: Disabled
  - Enable: `--enable-shared`
  - *kind* may be
    * gcc - Standard gcc and GNU ld options for creating shared libraries
    * osx-gcc - Special options for gcc needed only on OS/X
    * solaris-cc - Solaris native compilers
    * cygwin-gcc - Special options for gcc needed only for cygwin
    * none - same as `--disable-sharedlibs`

- Configuring with TotalView support

  - Default: Disabled
  - Enable: `--enable-g=dbg --enable-shared`
    `--enable-debuginfo`
  - *kind* may be
    * gcc - Standard gcc and GNU ld options for creating shared libraries
    * osx-gcc - Special options for gcc needed only on OS/X
    * solaris-cc - Solaris native compilers

- Path to OpenFabrics Header Files

  - Default: Your PATH
  - Specify: `--with-ib-include=path`

- Path to OpenFabrics Libraries

  - Default: The systems search path for libraries.
  - Specify: `--with-ib-libpath=path`

- Support for Hybrid UD-RC/XRC transports

  - Default: Disabled
  - Enable: `--enable-hybrid`

- Support for RDMA CM

- – Default: enabled, except when BLCR support is enabled

  – Disable: `--disable-rdma-cm`

- Support for RoCE

  – Default: enabled

  – For RoCE functionality to work properly, a version of OFED from the OFED-1.5-RoCE branch must be installed on all the systems.

- Registration Cache

  – Default: enabled

  – Disable: `--disable-registration-cache`

- ADIO driver for Lustre:

  – When compiled with this support, MVAPICH2 will use the optimized driver for Lustre. In order to enable this feature, the flag
  `--enable-romio --with-file-system=lustre`
  should be passed to `configure` (`--enable-romio` is optional as it is enabled by default). You can add support for more file systems using
  `--enable-romio --with-file-system=lustre+nfs+pvfs2`

- LiMIC2 Support

  – Default: disabled

  – Enable:
  `--with-limic2[=<path to LiMIC2 installation>]`
  `--with-limic2-include=<path to LiMIC2 headers>`
  `--with-limic2-libpath=<path to LiMIC2 library>`

- Header Caching

  – Default: enabled

  – Disable: `--disable-header-caching`

- Checkpoint/Restart

  – Option name: `--enable-ckpt`

  – Require: Berkeley Lab Checkpoint/Restart (BLCR)

  – Default: disabled

  The Berkeley Lab Checkpoint/Restart (BLCR) installation is automatically detected if installed in the standard location. To specify an alternative path to the BLCR installation, you can either use:
  `--with-blcr=<path/to/blcr/installation>`
  or
  `--with-blcr-include=<path/to/blcr/headers>`
  `--with-blcr-libpath=<path/to/blcr/library>`

4.4.0.1 Note: For MVAPICH2 versions $\leq$ 1.6rc2, Checkpoint/Restart is enabled with `--enable-blcr`.

- Process Migration

    – Option name: `--enable-ckpt-migration`
    – Automatically enable Checkpoint/Restart
    – Require: Fault Tolerance Backplane (FTB)
    – Default: disabled

The Fault Tolerance Backplane (FTB) installation is automatically detected if installed in the standard location. To specify an alternative path to the FTB installation, you can either use:
`--with-ftb=<path/to/ftb/installation>`
or
`--with-ftb-include=<path/to/ftb/headers>`
`--with-ftb-libpath=<path/to/ftb/library>`

4.4.0.2 Note: For MVAPICH2 versions $\leq$ 1.6rc2, Process Migration is enabled with `--enable-ftb-cr`.

- Checkpoint Aggregation

    – Option name: `--enable-ckpt-aggregation` or `--disable-ckpt-aggregation`
    – Automatically enable Checkpoint/Restart
    – Require: Filesystem in Userspace (FUSE)
    – Default: enabled (if Checkpoint/Restart enabled and FUSE is present)

The Filesystem in Userspace (FUSE) installation is automatically detected if installed in the standard location. To specify an alternative path to the FUSE installation, you can either use:
`--with-fuse=<path/to/fuse/installation>`
or
`--with-fuse-include=<path/to/fuse/headers>`
`--with-fuse-libpath=<path/to/fuse/library>`

4.4.0.3 Note: For MVAPICH2 versions $\leq$ 1.6rc2, Checkpoint Aggregation is respectively enabled and disabled with `--with-fuse` and `--without-fuse`.

- eXtended Reliable Connection

    – Default: enabled (if OFED installation supports it)
    – Enable: `--enable-xrc`
    – Disable: `--disable-xrc`

- HWLOC Support (Affinity)

    – Default: enabled
    – Disable: `--without-hwloc`

- Support for 64K or greater number of cores

  - Default: 64K or lower number of cores
  - Enable: `--with-ch3-rank-bits=32`

4.5 Configuring a build for NVIDIA GPU with OFA-IB-CH3

This section details the configuration option to enable GPU-GPU communication with the OFA-IB-CH3 interface of the MVAPICH2 MPI library. For more options on configuring the OFA-IB-CH3 interface, please refer to Section 4.4.

- Default: disabled

- Enable: `--enable-cuda`

The CUDA installation is automatically detected if installed in the standard location. To specify an alternative path to the CUDA installation, you can either use:
`--with-cuda=<path/to/cuda/installation>`
or
`--with-cuda-include=<path/to/cuda/include>`
`--with-cuda-libpath=<path/to/cuda/libraries>`

4.5.0.4   Note: If using the PGI compiler, you will need to add some `CPPFLAGS` to build properly. See the example below.

Example: `./configure --enable-cuda CPPFLAGS="-D__x86_64`
    `-D__align__\(n\)=__attribute__\(\(aligned\(n\)\)\)`
    `-D__location__\(a\)=__annotate__\(a\)`
    `-DCUDARTAPI="`

4.6 Configuring a build for Shared-Memory-CH3

The default CH3 channel provides native support for shared memory communication on stand alone multi-core nodes that are not equipped with InfiniBand adapters. The steps to configure CH3 channel explicitly can be found in Section 4.4. Dynamic Process Management (5.2.5) and CheckPoint/Restart (**??**) are currently not supported on stand-alone nodes without InfiniBand adapters.

4.7 Configuring a build for OFA-IB-Nemesis

The Nemesis sub-channel now is supported over OFA-IB. It can be built with:

    `$ ./configure --with-device=ch3:nemesis:ib`

In order to use shared libraries, configure as follows:

```
$ ./configure --with-device=ch3:nemesis:ib --enable-shared
```

To enable use of the TotalView debugger, the library needs to be configured in the following manner:

```
$ ./configure --with-device=ch3:nemesis:ib --enable-shared
--enable-g=dbg --enable-debuginfo
```

Configuration options for OFA-IB-Nemesis:

- Configuring with Shared Libraries

    - Default: Disabled
    - Enable: `--enable-shared`
    - *kind* may be
        * gcc - Standard gcc and GNU ld options for creating shared libraries
        * osx-gcc - Special options for gcc needed only on OS/X
        * solaris-cc - Solaris native compilers
        * cygwin-gcc - Special options for gcc needed only for cygwin
        * none - same as `--disable-sharedlibs`

- Configuring with TotalView support

    - Default: Disabled
    - Enable: `--enable-g=dbg --enable-shared`
      `--enable-debuginfo`
    - *kind* may be
        * gcc - Standard gcc and GNU ld options for creating shared libraries
        * osx-gcc - Special options for gcc needed only on OS/X
        * solaris-cc - Solaris native compilers

- Path to IB Verbs

    - Default: System Path
    - Specify: `--with-ibverbs=<path>` or
      `--with-ibverbs-include=<path>` and `--with-ibverbs-lib=<path>`

- Path to Umad

    - Default: System Path
    - Specify: `--with-umad=<path>` or
      `--with-umad-include=<path>` and `--with-umad-lib=<path>`

- Path to RDMA CM

    - Default: System Path
    - Specify: `--with-rdmacm=<path>` or
      `--with-rdmacm-include=<path>` and `--with-rdmacm-lib=<path>`

- Registration Cache

  - Default: enabled
  - Disable: `--disable-registration-cache`

- Header Caching

  - Default: enabled
  - Disable: `--disable-header-caching`

- Support for 64K or greater number of cores

  - Default: 64K or lower number of cores
  - Enable: `--with-ch3-rank-bits=32`

- Checkpoint/Restart

  - Default: disabled
  - Enable: `--enable-checkpointing` and `--with-hydra-ckpointlib=blcr`
  - Require: Berkeley Lab Checkpoint/Restart (BLCR)

The Berkeley Lab Checkpoint/Restart (BLCR) installation is automatically detected if installed in the standard location. To specify an alternative path to the BLCR installation, you can either use:
`--with-blcr=<path/to/blcr/installation>`
or
`--with-blcr-include=<path/to/blcr/headers>`
`--with-blcr-libpath=<path/to/blcr/library>`

4.8 Configuring a build for uDAPL-CH3

The uDAPL interface is the default on Solaris. It can be explicitly selected on both Solaris and Linux by configuring with:

```
$ ./configure --with-device=ch3:mrail
--with-rdma=udapl
```

In order to use shared libraries, configure as follows:

```
$ ./configure --with-device=ch3:mrail --with-rdma=udapl
--enable-shared
```

To enable use of the TotalView debugger, the library needs to be configured in the following manner:

```
$ ./configure --with-device=ch3:mrail --with-rdma=udapl
--enable-shared  --enable-g=dbg --enable-debuginfo
```

Configuration options for uDAPL-CH3:

- Configuring with Shared Libraries

  - Default: Disabled

- Enable: `--enable-shared`
- *kind* may be
    * gcc - Standard gcc and GNU ld options for creating shared libraries
    * osx-gcc - Special options for gcc needed only on OS/X
    * solaris-cc - Solaris native compilers
    * cygwin-gcc - Special options for gcc needed only for cygwin
    * none - same as `--disable-sharedlibs`

- Configuring with TotalView support

    - Default: Disabled
    - Enable: `--enable-g=dbg --enable-shared --enable-debuginfo`
    - *kind* may be
        * gcc - Standard gcc and GNU ld options for creating shared libraries
        * osx-gcc - Special options for gcc needed only on OS/X
        * solaris-cc - Solaris native compilers

- Path to OpenFabrics Header Files

    - Default: Your PATH
    - Specify: `--with-ib-include=path`

- Path to OpenFabrics Libraries

    - Default: The systems search path for libraries.
    - Specify: `--with-ib-libpath=path`

- Path to the DAPL Header Files

    - Default: Your PATH
    - Specify: `--with-dapl-include=path`

- Path to the DAPL Library

    - Default: The systems search path for libraries.
    - Specify: `--with-dapl-libpath=path`

- Default DAPL Provider

    - Default: OpenIB-cma on Linux; ibd0 on Solaris
    - Specify: `--with-dapl-provider=type`
        * Where `type` can be found in:
            · /etc/dat.conf on Linux
            · /etc/dat/dat.conf on Solaris

- DAPL Version

    - Default: 1.2
    - Specify: `--with-dapl-version=version`

- Cluster Size

    - Default: small
    - Specify: `--with-cluster-size=level`
        * Where `level` is one of:
            · small: < 128 processor cores
            · medium: 128 - 1024 cores
            · large: > 1024 cores

- I/O Bus

    - Default: PCI Express
    - Specify: `--with-io-bus=type`
        * Where `type` is one of:
            · PCI_EX for PCI Express
            · PCI_X for PCI-X

- Link Speed

    - Default: SDR
    - Specify: `--with-link=type`
        * Where `type` is one of:
            · DDR
            · SDR

- Registration Cache

    - Default: enabled on Linux; enabled and not configurable on Solaris
    - Disable (Linux only): `--disable-registration-cache`

- LiMIC2 Support

    - Default: disabled
    - Enable:
      `--with-limic2[=<path to LiMIC2 installation>]`
      `--with-limic2-include=<path to LiMIC2 headers>`
      `--with-limic2-libpath=<path to LiMIC2 library>`

- Header Caching

    - Default: enabled
    - Disable: `--disable-header-caching`

- HWLOC Support (Affinity)

    - Default: enabled

    - Disable: `--without-hwloc`

- Support for 64K or greater number of cores

    - Default: 64K or lower number of cores

    - Enable: `--with-ch3-rank-bits=32`

4.9   Configuring a build for QLogic PSM-CH3

The QLogic PSM interface needs to be built to use MVAPICH2 on InfiniPath adapters. It can built with:

```
$ ./configure --with-device=ch3:psm
```

In order to use shared libraries, configure as follows:

```
$ ./configure --with-device=ch3:psm --enable-shared
```

To enable use of the TotalView debugger, the library needs to be configured in the following manner:

```
$ ./configure --with-device=ch3:psm --enable-shared
--enable-g=dbg --enable-debuginfo
```

Configuration options for QLogic PSM channel:

- Configuring with Shared Libraries

    - Default: Disabled

    - Enable: `--enable-shared`

    - *kind* may be

        * gcc - Standard gcc and GNU ld options for creating shared libraries
        * osx-gcc - Special options for gcc needed only on OS/X
        * solaris-cc - Solaris native compilers
        * cygwin-gcc - Special options for gcc needed only for cygwin
        * none - same as `--disable-sharedlibs`

- Configuring with TotalView support

    - Default: Disabled

    - Enable: `--enable-g=dbg --enable-shared  --enable-debuginfo`

    - *kind* may be

        * gcc - Standard gcc and GNU ld options for creating shared libraries
        * osx-gcc - Special options for gcc needed only on OS/X
        * solaris-cc - Solaris native compilers

- Path to QLogic PSM header files

- Default: The systems search path for header files
- Specify: `--with-psm-include=path`

- Path to QLogic PSM library

  - Default: The systems search path for libraries
  - Specify: `--with-psm=path`

- Support for 64K or greater number of cores

  - Default: 64K or lower number of cores
  - Enable: `--with-ch3-rank-bits=32`

4.10 Configuring a build for TCP/IP-CH3

The use of TCP/IP requires the explicit selection of a TCP/IP enabled channel. The recommended channel is ch3:sock and it can be selected by configuring with:

```
$ ./configure --with-device=ch3:sock
```

In order to use shared libraries, configure as follows:

```
$ ./configure --with-device=ch3:sock --enable-shared
```

To enable use of the TotalView debugger, the library needs to be configured in the following manner:

```
$ ./configure --with-device=ch3:sock --enable-shared
--enable-g=dbg --enable-debuginfo
```

- Configuring with Shared Libraries

  - Default: Disabled
  - Enable: `--enable-shared`
  - *kind* may be
    * gcc - Standard gcc and GNU ld options for creating shared libraries
    * osx-gcc - Special options for gcc needed only on OS/X
    * solaris-cc - Solaris native compilers
    * cygwin-gcc - Special options for gcc needed only for cygwin
    * none - same as `--disable-sharedlibs`

- Configuring with TotalView support

  - Default: Disabled
  - Enable: `--enable-g=dbg --enable-shared`
    `--enable-debuginfo`
  - *kind* may be
    * gcc - Standard gcc and GNU ld options for creating shared libraries

  * osx-gcc - Special options for gcc needed only on OS/X
  * solaris-cc - Solaris native compilers

- Support for 64K or greater number of cores

  – Default: 64K or lower number of cores
  – Enable: `--with-ch3-rank-bits=32`

Additional instructions for configuring with TCP/IP can be found in the MPICH2 documentation available at:

   http://www.mcs.anl.gov/research/projects/mpich2/documentation/index.php?s=docs

4.11   Configuring a build for TCP/IP-Nemesis

The use of TCP/IP with Nemesis channel requires the following configuration:

```
$ ./configure --with-device=ch3:nemesis
```

In order to use shared libraries, configure as follows:

```
$ ./configure --with-device=ch3:nemesis --enable-shared
```

To enable use of the TotalView debugger, the library needs to be configured in the following manner:

```
$ ./configure --with-device=ch3:nemesis --enable-shared
--enable-g=dbg --enable-debuginfo
```

Additional instructions for configuring with TCP/IP-Nemesis can be found in the MPICH2 documentation available at: http://www.mcs.anl.gov/research/projects/mpich2/documentation/index.php?s=docs

- Configuring with Shared Libraries

  – Default: Disabled
  – Enable: `--enable-shared`
  – *kind* may be

    * gcc - Standard gcc and GNU ld options for creating shared libraries
    * osx-gcc - Special options for gcc needed only on OS/X
    * solaris-cc - Solaris native compilers
    * cygwin-gcc - Special options for gcc needed only for cygwin
    * none - same as `--disable-sharedlibs`

- Configuring with TotalView support

  – Default: Disabled
  – Enable: `--enable-g=dbg --enable-shared`
    `--enable-debuginfo`
  – *kind* may be

           ∗ gcc - Standard gcc and GNU ld options for creating shared libraries

           ∗ osx-gcc - Special options for gcc needed only on OS/X

           ∗ solaris-cc - Solaris native compilers

- Support for 64K or greater number of cores

  - Default: 64K or lower number of cores

  - Enable: `--with-ch3-rank-bits=32`

4.12 Configuring a build for OFA-IB-Nemesis and TCP/IP Nemesis (unified binary)

MVAPICH2 supports a unified binary for both OFA and TCP/IP communication through the Nemesis interface.

In order to configure MVAPICH2 for unified binary support, perform the following steps:

```
$ ./configure --with-device=ch3:nemesis:ib,tcp
```

You can use mpicc as usual to compile MPI applications. In order to run your application on OFA:

```
$ mpiexec -f hosts ./a.out -n 2
```

To run your application on TCP/IP:

```
$ MPICH_NEMESIS_NETMOD=tcp mpiexec -f hosts ./osu_latency -n 2
```

4.13 Configuring a build for Shared-Memory-Nemesis

The use of Nemesis shared memory channel requires the following configuration.

```
$ ./configure --with-device=ch3:nemesis
```

In order to use shared libraries, configure as follows:

```
$ ./configure --with-device=ch3:nemesis --enable-shared
```

To enable use of the TotalView debugger, the library needs to be configured in the following manner:

```
$ ./configure --with-device=ch3:nemesis --enable-shared
--enable-g=dbg --enable-debuginfo
```

Additional instructions for configuring with Shared-Memory-Nemesis can be found in the MPICH2 documentation available at: http://www.mcs.anl.gov/research/projects/mpich2/documentation/index.php?s=docs

- Configuring with Shared Libraries

  - Default: Disabled

  - Enable: `--enable-shared`

  - *kind* may be

    ∗ gcc - Standard gcc and GNU ld options for creating shared libraries

* osx-gcc - Special options for gcc needed only on OS/X
* solaris-cc - Solaris native compilers
* cygwin-gcc - Special options for gcc needed only for cygwin
* none - same as `--disable-sharedlibs`

- Configuring with TotalView support

  – Default: Disabled

  – Enable: `--enable-g=dbg --enable-shared --enable-debuginfo`

  – *kind* may be

    * gcc - Standard gcc and GNU ld options for creating shared libraries
    * osx-gcc - Special options for gcc needed only on OS/X
    * solaris-cc - Solaris native compilers

5 Basic Usage Instructions

5.1 Compile Applications

MVAPICH2 provides a variety of MPI compilers to support applications written in different programming languages. Please use `mpicc, mpif77, mpiCC`, or `mpif90` to compile applications. The correct compiler should be selected depending upon the programming language of your MPI application.

These compilers are available in the `MVAPICH2_HOME/bin` directory. MVAPICH2 installation directory can also be specified by modifying `$PREFIX`, then all the above compilers will also be present in the `$PREFIX/bin` directory.

5.2 Run Applications

This section provides instructions on how to run applications with MVAPICH2. Please note that on new multi-core architectures, process-to-core placement has an impact on performance. Please refer to Section 6.3 to learn about running MVAPICH2 library on multi-core nodes.

5.2.1 Run using `mpirun_rsh`

The MVAPICH team suggests users using this mode of job start-up for all interfaces (including OFA-IB-CH3, OFA-IB-Nemesis, OFA-iWARP-CH3, OFA-RoCE-CH3, PSM-CH3, uDAPL-CH3, Shared memory-CH3, TCP/IP-CH3 and TCP/IP-Nemesis) This `mpirun_rsh` *scheme provides fast and scalable job start-up. It scales to multi-thousand node clusters.*

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 host names should resolve to the same IP address on all machines. For instance, if a machine's host names resolves to 127.0.0.1 due to the default /etc/hosts on some Linux distributions it leads to incorrect behavior of the library.

5.2.1.1 Examples of running programs using `mpirun_rsh`:

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

This command launches `cpi` on nodes n0 and n1, two processes per node. By default `ssh` is used.

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

This command launches `cpi` on nodes n0 and n1, two processes per each node using `rsh` instead of `ssh`.

5.2.1.2   MPIRUN_RSH Hostfile:

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

A list of target nodes must be provided in the file `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".

5.2.1.3   Hostfile Format

The mpirun_rsh hostfile format allows for users to specify hostnames, one per line, optionally with a multiplier, and HCA specification.

The multiplier allows you to save typing by allowing you to specify blocked distribution of MPI ranks using one line per hostname. The HCA specification allows you to force an MPI rank to use a particular HCA.

The optional components are delimited by a ':'. Comments and empty lines are also allowed. Comments start with '#' and continue to the next newline.

5.2.1.4   Sample hostfile

```
$ cat hosts
# sample hostfile for mpirun_rsh
host1            # rank 0 will be placed on host1
host2:2          # rank 1 and 2 will be placed on host 2
host3:hca1       # rank 3 will be on host3 and will used hca1
host4:4:hca2     # ranks 4 through 7 will be on host4 and use hca2

# if the number of processes specified for this job is greater than 8
# then the additional ranks will be assigned to the hosts in a cyclic
# fashion.  For example, rank 8 will be on host1 and ranks 9 and 10 will
# be on host2.
```

5.2.1.5   Specifying Environmental Variables

Many parameters of the MPI library can be configured at 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.

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

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 such parameters, see Section 8.

5.2.1.6   Job Launch using MPMD

The mpirun_rsh framework also supports job launching using MPMD mode. It permits the use of heterogeneous jobs using multiple executables and command line arguments. The following format needs to be used:

```
$ mpirun_rsh -config configfile -hostfile hosts
```

A list of different group of executables must be provided to the job launcher in the file `configfile`, one per line. The `configfile` can contain comments. Lines beginning with "#" are considered comments.

For example:

```
#Config file example
#Launch 4 copies of exe1 with arguments arg1 and arg2
-n 4 :  exe1 arg1 arg2
#Launch 2 copies of exe2
-n 2 :  exe2
```

A list of target nodes must be provided in the file `hosts` one per line and the allocation policy previously described is used.

Please note that this section only gives general information on how to run applications using mpirun_rsh. Please refer to the following sections for more information on how to run the application over various interfaces such as iWARP and RoCE.

5.2.1.7   Other Options

Other options of `mpirun_rsh` can be obtained using

```
$ mpirun_rsh --help
```

5.2.2   Run using `Hydra (mpiexec)`

Hydra is the default process manager for MPICH2. MVAPICH2 also distributes Hydra along with with mpirun_rsh. Hydra can be used either by using `mpiexec` or `mpiexec.hydra`. All interfaces of MVAPICH2 will work using Hydra. The following is an example of running a program using it:

```
$ mpiexec -f hosts -n 2 ./cpi
```

The Hydra process manager can be used to launch MPMD jobs. For example the following command:

```
$ mpiexec -genv FOO=1 -env BAR=1 -n 2 ./exec1 :  -env BAR=2 -n 2
./exec2
```

The environment variable FOO=1 passed to "-genv" is applied the environment to all executables (i.e. exec1 and exec2). The values BAR=1 applies to exec1 and BAR=2 applies to only exec2.

This process manager has many features. Please refer to the following web page for more details.

http://wiki.mcs.anl.gov/mpich2/index.php/Using_the_Hydra_Process_Manager

5.2.3   Run 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/slurm

Once SLURM is installed and the daemons are started, applications compiled with MVAPICH2 can be launched by SLURM, e.g.

```
$ srun -n 2 ./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=none ./a.out
```

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

5.2.4   Run on PBS/Torque Clusters

You can use MVAPICH2 for clusters administered by PBS/Torque. If you are a cluster user (not an administrator), please ask your cluster administrator to install the OSC mpiexec. If you are a cluster administrator, please follow the instructions below.

You will need to download mpiexec from Ohio Supercomputer Center (OSC) at the following link.

Please note that this mpiexec is different from the mpiexec provided within MPICH2. Also note that you do not need to use either mpirun_rsh or mpiexec.hydra on a cluster that is administered with PBS/Torque. You may also choose to remove the MVAPICH2 mpiexecs from the install path to shield your users from making a mistake of trying to run MPI jobs with a wrong launcher.

Usage of OSC mpiexec is simple. An example of using this launcher is as follows:

```
$ /path/to/osc/mpiexec -np 4 ./a.out
```

5.2.5   Run with Dynamic Process Management support

MVAPICH2 (OFA-IB-CH3 interface) provides MPI-2 dynamic process management. This feature allows MPI applications to spawn new MPI processes according to MPI-2 semantics. The following commands

provide an example of how to run your application.

- To run your application using mpirun_rsh
  ```
  $ mpirun_rsh -np 2 -hostfile hosts MV2_SUPPORT_DPM=1 ./spawn1
  ```
  Note: It is necessary to provide the hostfile when running dynamic process management applications using mpirun_rsh.

- To run your application using mpiexec (Hydra)
  ```
  $ mpiexec -n 2 -f hosts -env MV2_SUPPORT_DPM 1 ./spawn1
  ```

Please refer to Section 11.74 for information about the MV2_SUPPORT_DPM environment variable.

5.2.6   Run with mpirun_rsh using OFA-iWARP Interface

The MVAPICH2 library can automatically detect iWARP cards and use them with the appropriate settings at run time. This feature deprecates the use of the environment variable `MV2_USE_IWARP_MODE` which was being used earlier to enable the use of iWARP devices at run time.

All the systems to be used need the following one time setup for enabling RDMA CM usage.

- **Setup the RDMA CM interface:** RDMA CM interface needs to be setup, configured with an IP address and connected to the network.

- **Setup the Local Address File:** Create the file (`/etc/mv2.conf`) with the local IP address to be used by RDMA CM. (Multiple IP addresses can be listed (one per line) for multi-rail configurations).
  ```
  $ echo 10.1.1.1 >> /etc/mv2.conf
  ```

Programs can be executed as follows:

```
$ mpirun_rsh -np 2 n0 n1 prog
```

The iWARP interface also provides TotalView debugging and shared library support. Please refer to Section 4.4.

5.2.7   Run with mpirun_rsh using OFA-RoCE Interface

RDMA over Converged Ethernet (RoCE) is supported with the use of the run time environment variable `MV2_USE_RoCE`.

Programs can be executed as follows:

```
$ mpirun_rsh -np 2 MV2_USE_RoCE=1 prog
```

5.2.8 Run using Hydra (mpiexec) with uDAPL-CH3 Interface

MVAPICH2 can be configured with the uDAPL interface, as described in the Section 4.8 . To compile MPI applications, please refer to the Section 5.1. In order to run MPI applications with uDAPL support, please specify the environmental variable `MV2_DAPL_PROVIDER`. As an example,

```
$ mpiexec -n 4 -f hosts -env MV2_DAPL_PROVIDER OpenIB-cma ./cpi
```

or:

```
$ export MV2_DAPL_PROVIDER=OpenIB-cma
```

```
$ mpiexec -n 4 -f hosts ./cpi
```

Please check the `/etc/dat.conf` file on Linux or `/etc/dat/dat.conf` on Solaris to find all the available uDAPL service providers. The default value for the uDAPL provider will be chosen, if no environment variable is provided at run time. If you are using OpenFabrics software stack on Linux, the default DAPL provider is `OpenIB-cma` for DAPL-1.2, and `ofa-v2-ib0` for DAPL-2.0. If you are using Solaris, the default DAPL provider is `ibd0`.

The uDAPL interface also provides TotalView debugging and shared library support. Please refer to section 4.8 for more information. Please note that mpirun_rsh can also be used for running uDAPL-CH3 interface using the same MV2_DAPL_PROVIDER environment variable.

5.2.9 Run using IPoIB with mpirun_rsh or mpiexec

You would like to run an MPI job using IPoIB but your IB card is not the default interface for IP traffic. Assume that you have a cluster setup as the following:

#hostname	Eth Addr	IPoIB Addr
compute1	192.168.0.1	192.168.1.1
compute2	192.168.0.2	192.168.1.2
compute3	192.168.0.3	192.168.1.3
compute4	192.168.0.4	192.168.1.4

The Ethernet addresses are assigned to eth0 and the IPoIB addresses are assigned to the ib0 interface. The host names resolve to the 192.168.0.* addresses.

The most direct way to use the IPoIB network is to populate your hosts file with the IP addresses of the ib0 interfaces.

Example:

```
$ cat - > hosts
192.168.1.1
192.168.1.2
192.168.1.3
192.168.1.4
```

```
$ mpirun_rsh -hostfile hosts -n 4 ./app1
```

or

```
$ mpiexec -f hosts -n 4 ./app1
```

Another way to achieve this is to use the *-iface* option of hydra. This allows you to have your hosts file to use the host names even though they resolve to the eth0 interface.

Example:

```
$ cat - > hosts
compute1
compute2
compute3
compute4

$ mpiexec -f hosts -iface ib0 -n 4 ./app1
```

More information can be found at the following link.

5.2.10 Run using ADIO driver for Lustre

MVAPICH2 contains optimized Lustre ADIO support for the OFA-IB-CH3 interface. The Lustre directory should be mounted on all nodes on which MVAPICH2 processes will be running. Compile MVAPICH2 with ADIO support for Lustre as described in Section 4. 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
```

5.2.11 Run using TotalView Debugger Support

MVAPICH2 provides TotalView support. The following commands provide an example of how to build and run your application with TotalView support. Note: running TotalView requires correct setup in your environment, if you encounter any problem with your setup, please check with your system administrator for help.

- Compile your mpi application with debug symbols...
  ```
  $ mpicc -g -o prog prog.c
  ```

- Define the correct path to TotalView as the TOTALVIEW variable...
  ```
  $ export TOTALVIEW=<path_to_TotalView>
  ```

- Run your program. . .
  ```
  $ mpirun_rsh -tv -np 2 n0 n1 prog
  ```

- Troubleshooting:

  - X authentication errors: check if you have enabled X Forwarding
    ```
    $ echo "ForwardX11 yes" >> $HOME/.ssh/config
    ```
  - ssh authentication error: ssh to the computer node with its long form hostname
    ```
    $ ssh i0.domain.osu.edu
    ```

5.2.12 Run using a profiling library

All MPI2-functions of MVAPICH2 support the MPI profiling interface. This allows MVAPICH2 to be used by a variety of profiling libraries for MPI applications.

Two use of profiling libraries will be describe below, mpiP and Scalasca;

- To use mpiP, you should link your program with the required libraries as describe with the following command:
  ```
  $ mpicc -g prg.o -o prg -L${mpiP_root}/lib -lmpiP -lm -lbfd
  -liberty -lunwind
  ```
  MpiP has many features. Please refer to the mpiP web page for more details.

- To use Scalasca, you should configure Scalasca by supplying the '--mpi=mpich2' option like shows below:
  ```
  $./configure --mpi=mpich2
  ```

  Once the installation is done, you will be able to use Scalasca with MVAPICH2.

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

6 Advanced Usage Instructions

In this section, we present the usage instructions for advanced features provided by MVAPICH2.

6.1 Running on Customized Environments

In MVAPICH2 1.8, run-time variables are used to switch various optimization schemes on and off. Following is a list of optimizations schemes and the control environmental variables, for a full list please refer to the section 11:

- **Extended Reliable Connection:** Use the XRC InfiniBand transport available with Mellanox ConnectX adapters. Default: off; to enable:
  ```
  $ mpirun_rsh -np 2 n0 n1 MV2_USE_XRC=1 prog
  ```
  or
  ```
  $ mpiexec -n 2 -hosts n0,n1 -env MV2_USE_XRC 1 prog
  ```

- **Enable use of multiple communication channels:** Indicates the number of queue pairs per port to be used for communication on an end node. Default: 1; to change default value:
  ```
  $ mpirun_rsh -np 2 n0 n1 MV2_NUM_QP_PER_PORT=2 prog
  ```
  or
  ```
  $ mpiexec -n 2 -hosts n0,n1 -env MV2_NUM_QP_PER_PORT 2 prog
  ```

- **Adaptive RDMA fast path:** using RDMA write to enhance performance for short messages. Default: on; to disable:
  ```
  $ mpirun_rsh -np 2 n0 n1 MV2_USE_RDMA_FAST_PATH=0 prog
  ```
  or
  ```
  $ mpiexec -n 2 -hosts n0,n1 -env MV2_USE_RDMA_FAST_PATH 0 prog
  ```

6.2 Configuration File Processing

MVAPICH2 supports the use of configuration values to ease the burden of users when they would like to set and repeatedly use many different environment variables. These can be stored in a configuration file with statements of the form "VARIABLE = VALUE". Full line comments are supported and begin with the "#" character.

The system configuration file can be placed at /etc/mvapich2.conf while user configuration files are located at "`˜/.mvapich2.conf`" by default. The user configuration file can be specified at runtime by MV2_USER_CONFIG if the user would like to have mvapich2 read from a different location.

The processing of these files can be disabled by the use of the MV2_IGNORE_SYSTEM_CONFIG and MV2_IGNORE_USER_CONFIG.

6.2.1   Sample Use

6.2.1.1   Run with blocking mode enabled

```
$ cat ~/.mvapich2.conf
# Enable blocking mode
MV2_USE_BLOCKING = 1
$ mpirun_rsh -n 2 compute1 compute2 ./osu_latency
```

6.2.1.2   Do not use user configuration file

```
$ mpirun_rsh -n 2 compute1 compute2 MV2_IGNORE_USER_CONFIG=1 ./osu_latency
```

6.3   Running with Efficient CPU (Core) Mapping

MVAPICH2-CH3 interfaces support architecture specific CPU mapping through the Portable Hardware Locality (hwloc) software package. By default, the HWLOC sources are compiled and built while the MVA-PICH2 library is being installed. Users can choose the "–disable-hwloc" parameter while configuring the library if they do not wish to have the HWLOC library installed. However, in such cases, the MVAPICH2 library will not be able to perform any affinity related operations.

There are two major schemes as indicated below. To take advantage of any of these schemes, the jobs need to run with CPU affinity (MV2_ENABLE_AFFINITY) and shared memory (MV2_USE_SHARED_MEM) turned on (default). If users choose to set these run-time parameters to 0, then the kernel takes care of mapping processes to cores and none of these schemes will be enabled.

6.3.1   Using HWLOC for CPU Mapping

Under this scheme, the HWLOC tool will be used at job-launch time to detect the processor's micro-architecture, and then generate a suitable cpu mapping string based. Two policies are currently implemented: "bunch" and "scatter". By default, we choose to use the "bunch" mapping. However, we also allow users to choose a binding policy through the run-time variable, MV2_CPU_BINDING_POLICY. (Section 11.14)

For example, if you want to run 4 processes per node and utilize "bunch" policy on each node, you can specify:

```
$ mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_POLICY=bunch
./a.out
```

The CPU binding will be set as shown in Figure 2.

If you want to run 4 processes per node and utilize "scatter" policy on each node, you can specify:

```
$ mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_POLICY=scatter
./a.out
```

The CPU binding will be set as shown in Figure 3.
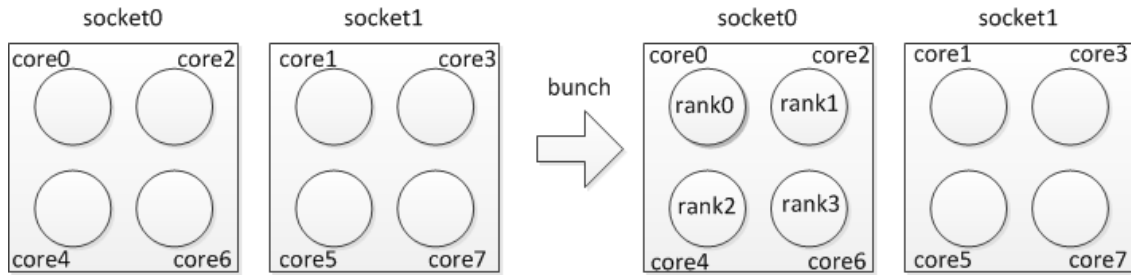
Figure 2: Process placement with "bunch" CPU binding policy
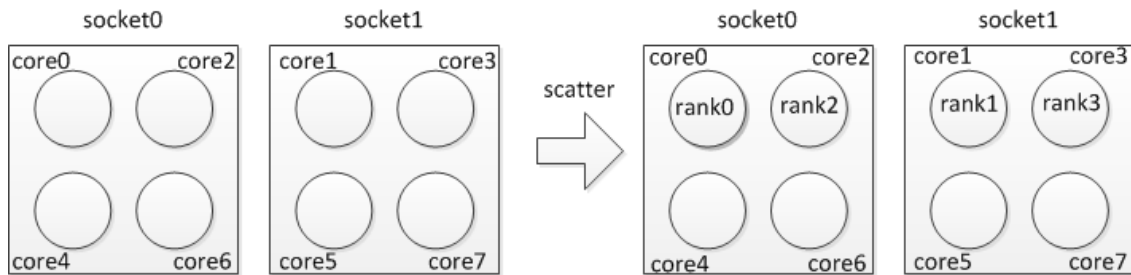
Figure 3: Process placement with "Scatter" CPU binding policy

If two applications with four processes each need to share a given node (with eight cores) at the same time with "bunch" policy, you can specify:

```
$ mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_POLICY=bunch
./a.out
$ mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_POLICY=bunch
./b.out
```

The CPU binding will be set as shown in Figure 4.
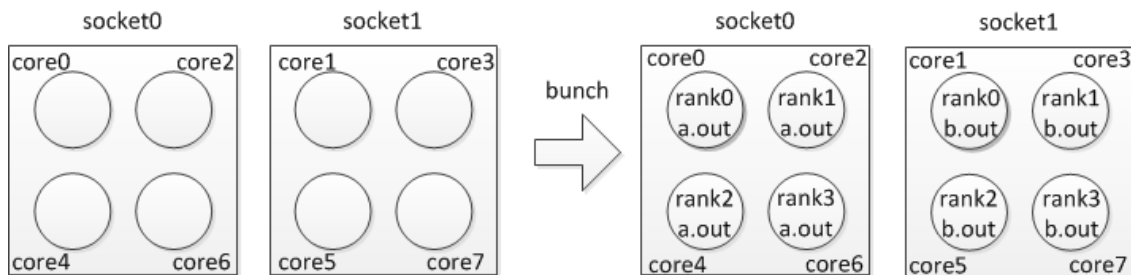
Figure 4: Process placement with two applications using the "bunch" CPU binding policy

If two applications with four processes each need to share a given node (with eight cores) at the same time with "scatter" policy, you can specify:

```
$ mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_POLICY=scatter
./a.out
```

```
$ mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_POLICY=scatter
./b.out
```

The CPU binding will be set as shown in Figure 5.
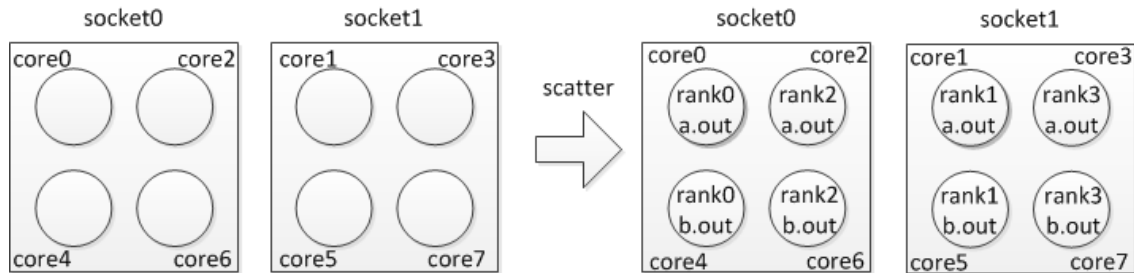
Figure 5: Process placement with two applications using the "scatter" CPU binding policy

The aforementioned binding is based on the core level, meaning each MPI process will be bound to a specific core. Actually, we provide different process binding level. There are three binding levels: "core", "socket", and "numanode" (which is designed for some multicore processor with NUMA node unit). We use the "core" as the default binding level, and we also allow users to choose a binding level through the run-time variable, MV2_CPU_BINDING_LEVEL. (Section 11.15) For example, if you want to run 4 processes per node and utilize "socket" as the binding level on each node, you can specify:

```
$ mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_LEVEL=socket
./a.out
```

The CPU binding will be set as shown in Figure 6. Note: because we use "bunch" as the default binding policy, all four processes will be bound to the first socket and each of them can use all four cores in this socket. When the binding policy is "bunch" and the binding level is "socket", processes will be bound to the same socket until the process number is larger than the core number in the socket.
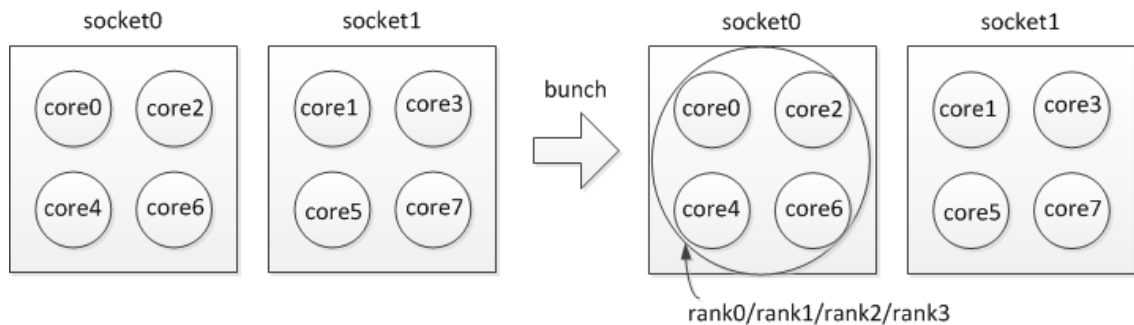
Figure 6: Process placement with the "bunch" CPU binding policy and "socket" binding level

If you want to run 4 processes per node, utilize "socket" as the binding level and "scatter" as the binding policy, you can specify:

```
$ mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_LEVEL=socket
MV2_CPU_BINDING_POLICY=scatter ./a.out
```

The CPU binding will be set as shown in Figure 7.
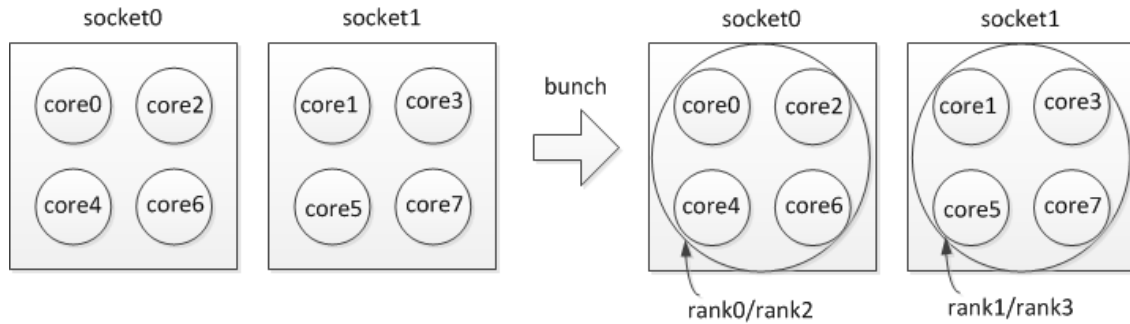
Figure 7: Process placement with the "scatter" CPU binding policy and "socket" binding level

6.3.2 User defined CPU Mapping

Under the second scheme, users can also use their own mapping to bind processes to CPU's on modern multi-core systems. The feature is especially useful on multi-core systems, where performance may be different if processes are mapped to different cores. The mapping can be specified by setting the environment variable `MV2_CPU_MAPPING` (Section 11.13).

For example, if you want to run 4 processes per node and utilize cores 0, 1, 4, 5 on each node, you can specify:

```
$ mpirun_rsh -np 64 -hostfile hosts MV2_CPU_MAPPING=0:1:4:5
./a.out
or

$ mpiexec -n 64 -f hosts -env MV2_CPU_MAPPING 0:1:4:5 ./a.out
```

In this way, process 0 on each node will be mapped to core 0, process 1 will be mapped to core 1, process 2 will be mapped to core 4, and process 3 will be mapped to core 5. For each process, the mapping is separated by a single ":".

MVAPICH2 supports binding one process to multiple cores in the same node with "," or "-". For example:

```
$ mpirun_rsh -np 64 -hostfile hosts MV2_CPU_MAPPING=0,2,3,4:1:5:6
./a.out
or
$ mpirun_rsh -np 64 -hostfile hosts MV2_CPU_MAPPING=0,2-4:1:5:6
./a.out
```

In this way, process 0 on each node will be mapped to core 0, core 2, core 3, and core 4; process 1 will be mapped to core 1, process 2 will be mapped to core 5, and process 3 will be mapped to core 6. This feature is designed to support the case that one rank process will spawn multiple threads and set thread binding in the program.

6.3.3   Performance Impact of CPU Mapping

Here we provide a table with latency performance of 0 byte and 8KB messages using different CPU mapping schemes. The results show how process binding can affect the benchmark performance. We strongly suggest the consideration of best CPU mapping on multi-core platforms when carrying out benchmarking and performance evaluation with MVAPICH2.

The following measurements were taken on the machine with the dual quad-core 2.53GHz Intel Xeon processors with 12MB L3 shared cache (among cores in one socket). MVAPICH2-1.8 was built with gcc-4.4.6 and default configure arguments:

Core Pair	Message Latency		Notes
	0-byte	8k-byte	
1,2	0.17 us	1.83 us	same socket, shared L3 cache, best performance
0,1	0.17 us	1.87 us	same socket, shared L3 cache, but core 0 handles interrupts
1,5	0.41 us	3.16 us	different sockets
0,4	0.42 us	3.17 us	different sockets, but core 0 handles interrupts

6.4   Running with LiMIC2

MVAPICH2 CH3-based interfaces support LiMIC2 for intra-node communication for medium and large messages to get higher performance. LiMIC2 is also used to optimize intra-node one-sided communication in OFA-IB-CH3 and OFA-iWARP-CH3 interfaces. It is disabled by default because it depends on the LiMIC2 package to be previously installed. As a convenience we have distributed the latest LiMIC2 package (as of this release) with our sources.

To install this package, please take the following steps.

- Navigate to the LiMIC2 to source
  ```
  $ cd limic2-0.5.5
  ```

- Configure and build the source
  ```
  limic2-0.5.5$ ./configure --enable-module --prefix=/usr
  --sysconfdir=/etc && make
  ```

- Install
  ```
  limic2-0.5.5$ sudo make install
  ```

Before using LiMIC2 you'll need to load the kernel module. If you followed the instructions above you can do this using the following command (LSB init script).

- ```
  $ /etc/init.d/limic start
  ```

Please note that supplying '`--sysconfdir=/etc`' in the configure line above told the package to install the init script and an udev rule in the standard location for system packages. Supplying '`--prefix=/usr`' will also install the headers and libraries in the system path. These are optional but recommended.

Now you can use LiMIC2 with MVAPICH2 by simply supplying the '`--with-limic2`' option when configuring MVAPICH2. You can run your applications as normal and LiMIC2 will be used by default. To disable it at run time, use the env variable:

```
$ mpirun_rsh -np 64 -hostfile hosts MV2_SMP_USE_LIMIC2=0 ./a.out
```

LiMIC2 optimizations for one-sided communication are currently available for Put and Get operations with both active and passive synchronization semantics. Using LIMIC2, MVAPICH2 provides truly passive synchronization within a node. These optimizations are enabled by default, when MVAPICH2 is configured with "–with-limic2". They can be disabled at run time, as below:

```
$ mpirun_rsh -np 64 -hostfile hosts MV2_USE_LIMIC_ONE_SIDED=0
./a.out
```

6.5 Running with Shared Memory Based Windows for One-sided Communication

MVAPICH2 optimizes one-sided communication when the windows are created in shared memory. Programmers can use the MPI_Info argument in MPI_Alloc_mem to provide a hint (key: alloc_shm, value: true) when the buffer will be used to create a window and should be allocated in shared memory. MVAPICH2 internally detects the windows created on such buffers and optimizes the communication on them. The snippet below shows how a user can provide the hint in a MPI_Alloc_mem call. One-sided communication benchmarks in the OMB suite serve as complete examples. Through this design MVAPICH2 provides truly passive one-sided semantics within a node.

MPI_Info_create(&win_info); MPI_Info_set(win_info, "alloc_shm", "true"); MPI_Alloc_mem (size, win_info, &buffer);

Users who want to take advantage of this design without changing the application code (to provide the hint) can set the run time parameter MV2_USE_SHM_ONE_SIDED (11.92) to 1. When this is set, all the buffers allocated using MPI_Alloc_mem are allocated in shared memory by default. This helps optimize the one-sided communication that happens on this memory. However, users have to note that the buffers still have to be allocated using MPI_Alloc_mem and this option is less scalable as the amount of shared memory allocated on a node can be limited. Hence, use of the optimization hint described above is recommended.

6.6 Running with Shared Memory Collectives

In MVAPICH2, support for shared memory based collectives has been enabled for MPI applications running over OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3 and PSM-CH3 interfaces. Currently, this support is available for the following collective operations:

- MPI_Allreduce
- MPI_Reduce
- MPI_Barrier
- MPI_Bcast

Optionally, these feature can be turned off at run time by using the following parameters:

- `MV2_USE_SHMEM_COLL` (section 11.96)

- `MV2_USE_SHMEM_ALLREDUCE` (section 11.93)

- `MV2_USE_SHMEM_REDUCE` (section 11.97)

- `MV2_USE_SHMEM_BARRIER` (section 11.94)

- `MV2_USE_SHMEM_BCAST` (section 11.95)

Please refer to Section 11 for further details.

6.7   Running with scalable UD transport

MVAPICH2 has scalable design with Infiniband connection less transport Unreliable Datagram (UD). Applications can use UD only transport by simply configuring MVAPICH2 with the –enable-hybrid and setting the environment variable MV2_USE_ONLY_UD(11.119). In this mode, library not uses any reliable RC connections. This feature eliminates all the overheads associated with RC connections and reduces the memory footprint at large scale.

6.8   Running with Integrated Hybrid UD-RC/XRC design

MVAPICH2 has integrated hybrid transport support for OFA-IB-CH3. This provides the capability to use Unreliable Datagram (UD), Reliable Connection (RC) and eXtended Reliable Connection (XRC) transports of InfiniBand. This hybrid transport design is targeted at emerging clusters with multi-thousand core clusters to deliver best possible performance and scalability with constant memory footprint.

Applications can use Hybrid transport by simply configuring MVAPICH2 with the –enable-hybrid option. In this configuration, MVAPICH2 seamlessly uses UD and RC/XRC connections by default. The use of UD transport can be disabled at run time by setting the environment variable MV2_USE_UD_HYBRID(11.118) to Zero.

MV2_HYBRID_ENABLE_THRESHOLD (11.113) defines the threshold for enabling the hybrid transport. Hybrid mode will be used when the size of the job is greater than or equal to the threshold. Otherwise, it uses default RC/XRC connections.

For a full list of Hybrid environment variables, please refer Section 11.

6.9   Running with Multiple-Rail Configurations

MVAPICH2 has integrated multi-rail support for OFA-IB-CH3 and OFA-iWARP-CH3 interfaces. Run-time variables are used to specify the control parameters of the multi-rail support; number of adapters with MV2_NUM_HCAS (section 11.37), number of ports per adapter with MV2_NUM_PORTS (section 11.38),

and number of queue pairs per port with MV2_NUM_QP_PER_PORT (section 11.40). Those variables are default to 1 if you do not specify them.

Large messages are striped across all HCAs. The threshold for striping is set according to the following formula:
(MV2_VBUF_TOTAL_SIZE × MV2_NUM_PORTS × MV2_NUM_QP_PER_PORT × MV2_NUM_HCAS).
In addition, there is another parameter MV2_STRIPING_THRESHOLD (section 11.73) which users can utilize to set the striping threshold directly.

MVAPICH2 also gives the flexibility to balance short message traffic over multiple HCAs in a multi-rail configuration. The run-time variable MV2_SM_SCHEDULING can be used to choose between the various load balancing options available. It can be set to USE_FIRST (Default) or ROUND_ROBIN. In the USE_FIRST scheme, the HCA in slot 0 is always used to transmit the short messages. If ROUND_ROBIN is chosen, messages are sent across all HCAs alternately.

In the following example, we can use multi-rail support with two adapters, using one port per adapter and one queue pair per port:

```
$ mpirun_rsh -np 2 n0 n1 MV2_NUM_HCAS=2 MV2_NUM_PORTS=1
MV2_NUM_QP_PER_PORT=1 prog
```

Using the Hydra process manager, the same can be accomplished by:

```
$ mpiexec -n 2 -hosts n0,n1 -env MV2_NUM_HCAS 2 -env MV2_NUM_PORTS
1 -env MV2_NUM_QP_PER_PORT 1 prog
```

Note that the default values of `MV2_NUM_PORTS` and `MV2_NUM_QP_PER_PORT` are 1, so they can be omitted.

```
$ mpirun_rsh -np 2 n0 n1 MV2_NUM_HCAS=2 prog
```

Using the Hydra process launcher, the following command can be used:

```
$ mpiexec -n 2 -hosts n0,n1 -env MV2_NUM_HCAS 2 prog
```

The user can also select the particular network card(s) that should be used by using the MV2_IBA_HCA environment variable specified in section 11.25. The following is an example of how to run MVAPICH2 in this mode. (In the example "mlx4_0" is the name of the InfiniBand card as displayed by the output of the "ibstat" command).

```
$ mpirun_rsh -np 2 n0 n1 MV2_IBA_HCA=mlx4_0 prog
```

If there are multiple HCAs in the system, the user can also selectively use some or all of these HCAs for network operations by using the MV2_IBA_HCA environment variable. Different HCAs are delimited by colons ":". An example is shown below. In the example "mlx4_0" and "mlx4_1" are the names of the

InfiniBand card as displayed by the output of the "ibstat" command. There can be other HCAs in the system as well.

```
$ mpirun_rsh -np 2 n0 n1 MV2_IBA_HCA=mlx4_0:mlx4_1 prog
```

6.10   Enhanced design for Multiple-Rail Configurations

MVAPICH2 now features an enhanced design for multiple rail configurations for OFA-IB-CH3 and OFA-iWARP-CH3 interfaces. It can broadly be explained by the figure given below. In addition to the earlier design where the rails were shared amongst processes at run time (as depicted under the Rail Sharing banner in the figure below), MVAPICH2 now features a new RAIL BINDING policy which will dedicate a particular rail to a particular process.
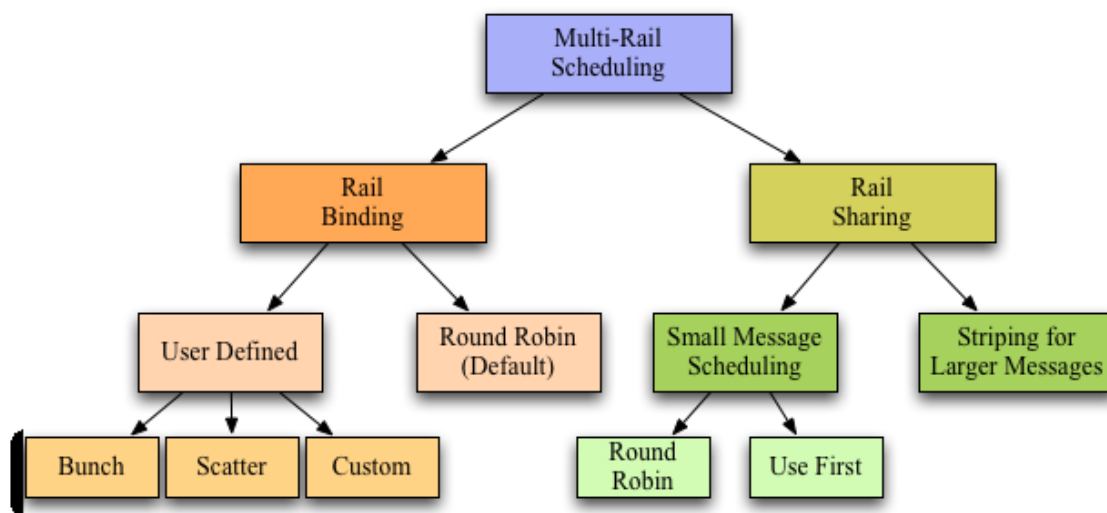
Figure 8: Multi-rail Scheduling policies

The scheduling policies are broadly classified into 2 basic types. Rail Binding and Rail Sharing.

- The Rail Sharing policy has been kept the same as in earlier versions and can be set using the MV2_RAIL_SHARING_POLICY (section 11.41) (previously MV2_SM_SCHEDULING) parameter. It can take the values USE_FIRST or ROUND_ROBIN. However as opposed to the earlier versions, the number of HCAs of the same type on a system get detected automatically and used. The user can still make use of the MV2_NUM_HCAS parameter and set the number to a custom number of rails if required. If this parameter is not set the maximum number gets used.

- In the Rail Binding policy only that HCAs is made available to the process which has been assigned to it using some scheme. The default policy used is 'Rail Binding in a round-robin manner': If the parameter MV2_RAIL_SHARING_POLICY is not specified, MVAPICH2 assigns HCAs to the processes in a round-robin manner. The Rail Binding policy can either be default as described above or user defined. For the user defined mode the following parameter MV2_RAIL_SHARING_POLICY=FIXED_MAPPING should be set.

The user defined policies can be set in the following manner by giving appropriate values to the parameter MV2_PROCESS_TO_RAIL_MAPPING (section 11.43)

- – MV2_PROCESS_TO_RAIL_MAPPING=BUNCH : The HCAs are assigned in a block manner. e.g. For 4 rails and 16 processes the mapping will be 0:0:0:0:1:1:1:1:2:2:2:2:3:3:3:3

- – MV2_PROCESS_TO_RAIL_MAPPING=SCATTER : The HCAs are assigned in a cyclic manner. e.g. For 4 rails and 16 processes the mapping will be 0:1:2:3:0:1:2:3:0:1:2:3:0:1:2:3

- – The third option is a custom list which can be passed as a string to MV2_PROCESS_TO_RAIL_MAPPING separated by a colon(:) as shown above.

- In addition to these parameters the other ENVS that can be set are MV2_VBUF_TOTAL_SIZE, MV2_NUM_QP_PER_PORT, MV2_RAIL_SHARING_LARGE_MSG_THRESHOLD (previously MV2_STRIPING_THRESHOLD), MV2_IBA_HCA. Please refer to the usage described for these parameters as well as the ones described above in (section 11).

6.11   Running with Fault-Tolerance Support

6.11.1   System-Level Checkpoint/Restart

MVAPICH2 provides system-level checkpoint/restart functionality based on a coordinated Checkpoint/Restart protocol involving all the application processes.

The following section (6.11.1.1) provides instructions for basic checkpoint/restart operations with MVA-PICH2. These require the Berkeley Lab Checkpoint/Restart (BLCR) library in order perform checkpoints of local processes. Then, advanced features are presented. Section 6.11.1.2 details the usage of the fast-checkpointing scheme based on aggregation. Section 6.11.1.3 present the support of the new standardized Fault Tolerance Backplane (FTB).

6.11.1.1   Basic Checkpoint/Restart Scheme: BLCR is a library that allows to take checkpoint of individual processes. Its usage is mandatory to take advantage of the checkpoint/restart functionality in MVA-PICH2. Here are the steps that allows the usage of BLCR with MVAPICH2.

- Download and install the BLCR (Berkeley Lab's Checkpoint/Restart) package. The packages can be downloaded from this web page.

- Make sure the BLCR packages are installed on every node and the LD_LIBRARY_PATH must contain the path to the shared library of BLCR, usually $BLCR_HOME/lib.

- MVAPICH2 needs to be compiled with checkpoint/restart support, see section 4.4.

- BLCR kernel modules must be loaded on all the compute nodes.

- Make sure the PATH contains the path to the executables of BLCR, usually $BLCR_HOME/bin.

Users are strongly encouraged to read the Administrators guide of BLCR, and test the BLCR on the target platform, before using the checkpointing feature of MVAPICH2.

Checkpointing operation

Now, your system is set up to use the Checkpoint/Restart features of MVAPICH2. Several parameters are provided by MVAPICH2 for flexibility in configuration and using the Checkpoint / Restart features. If mpiexec is used as the job start up mechanism, these parameters need to be set in the user's environment through the BASH shell's `export` command, or the equivalent command for other shells. If mpirun_rsh is used as the job start up mechanism, these parameters need to be passed to mpirun_rsh through the command line.

- MV2_CKPT_FILE: This parameter specifies the path and the base file name for checkpoint files of MPI processes. Please note that File System performance is critical to the performance of checkpointing. This parameter controls which file system will be used to store the checkpoint files. For example, if your PVFS2 is mounted at
  `/mnt/pvfs2`, using `MV2_CKPT_FILE=/mnt/pvfs2/ckptfile` will let the checkpoint files being stored in pvfs2 file system. See Section 11.4 for details.

- MV2_CKPT_INTERVAL: This parameter (in minutes) can be used to enable automatic checkpointing. See Section 11.5 for details.

- MV2_CKPT_MAX_SAVE_CKPTS: This parameter is used to limit the number of checkpoints saved on file system. See Section 11.6 for details.

- MV2_CKPT_NO_SYNC: This parameter is used to control whether the program forces the checkpoint files being synced to disk or not before it continues execution. See Section 11.7 for details.

In order to provide maximum flexibility to end users who wish to use the checkpoint/restart features of MVAPICH2, we've provided three different methods which can be used to take the checkpoints during the execution of the MPI application. These methods are described as follows:

- Manual Checkpointing: In this mode, the user simply launches an MPI application and chooses when to checkpoint the application. This mode can be primarily used for experimentation during deployment stages. In order to use this mode, the MPI application is launched normally using `mpiexec` or `mpirun_rsh`. When the user decides to take a checkpoint, the users can issue a BLCR command namely "`cr_checkpoint`" in the following manner:

  ```
  cr_checkpoint -p <PID>
  ```

  where PID is the process id of the `mpiexec` or `mpirun_rsh` process. In order to simplify the process, the script `mv2_checkpoint` can be used. This script is available in the same directory as `mpiexec` and `mpirun_rsh`.

- Automated Checkpointing: In this mode, the user can launch the MPI application normally using `mpiexec` or `mpirun_rsh`. However, instead of manually issuing checkpoints as described in the above bullet, a parameter (`MV2_CKPT_INTERVAL`) can be set to automatically take checkpoints and user-defined intervals. Please refer to Section 11.5 for a complete usage description of this variable. This mode can be used to take checkpoints of a long running application, for example every 1 hour, 2 hours etc. based on user's choice.

- Application Initiated Synchronous Checkpointing: In this mode, the MPI application which is running can itself request for a checkpoint. Application can request a whole program checkpoint synchronously by calling `MVAPICH2_Sync_Checkpoint`. Note that it is a collective operation, and this function must be called from all processes to take the checkpoint. This mode is expected to be used by applications that can be modified and have well defined synchronization points. These points can be effectively used to take checkpoints. An example of how this mode can be activated is given below.

```c
#include "mpi.h"
#include <unistd.h>
#include <stdio.h>

int main(int argc,char *argv[])
{
    MPI_Init(&argc,&argv);
    printf("Computation\n");
    sleep(5);
    MPI_Barrier(MPI_COMM_WORLD);
    MVAPICH2_Sync_Checkpoint();
    MPI_Barrier(MPI_COMM_WORLD);
    printf("Computation\n");
    sleep(5);
    MPI_Finalize();
    return 0;
}
```

Restart operation

To restart a job from a **manual checkpoint**, users need to issue another command of BLCR, "`cr_restart`" with the checkpoint file name of the MPI job console as the parameter. Usually, this file is named `context.<pid>`. The checkpoint file name of the MPI job console can be specified when issuing the checkpoint (see the "`cr_checkpoint --help`" for more information). Please note that the names of checkpoint files of the MPI processes will be assigned according to the environment variable MV2_CKPT_FILE, (`$MV2_CKPT_FILE.<number of checkpoint>.<process rank>`).

To restart a job from an **automatic checkpoint**, use `cr_restart $MV2_CKPT_FILE.<number of checkpoint>.auto`.

If the user wishes to restart the MPI job on a different set of nodes, the host file that was specified along with the "`-hostfile`" option during job launch phase should be modified accordingly before trying to

restart a job with "`cr_restart`". This modified "`hostfile`" must be at the same location and with the same file name as the original hostfile. The mpirun_rsh framework parses the host file again when trying to restart from a checkpoint, and launches the job on the corresponding nodes. This is possible as long as the nodes in which the user is trying to restart has the exact same environment as the one in which the checkpoint was taken (including shared NFS mounts, kernel versions, and user libraries).

For this to function correctly, the user should disable pre-linking on both the source and the destination node. See the FAQ Section of the BLCR userguide for more information.

Please refer to the Section 9.7 for troubleshooting with Checkpoint/Restart.

6.11.1.2   Write-Aggregation based Fast Checkpointing Scheme: MVAPICH2 provides an enhanced technique that allows fast checkpoint and restart. This scheme, named Aggregation, relies on the Filesystem in Userspace (FUSE) library.

Although Aggregation is optional, its usage is recommended to achieve best performances during checkpoint and restart operations. That is why, if the FUSE library is detected during configuration step, it will be automatically enabled (see section 4.4). Once enabled at configuration step, aggregation scheme can be disabled at run time by setting the environment variable `MV2_CKPT_USE_AGGREGATION=0` (see section 11.8 for details).

The following steps need to be done to use FUSE library for aggregation scheme.

- Download and install the Filesystem in Userspace (FUSE) library package. The packages can be downloaded from this web page. To get the best performance, users are encouraged to upgrade to kernel version $\geq$ 2.6.26 and use FUSE library $\geq$ 2.8.

- Make sure the FUSE packages are installed on every node and the `LD_LIBRARY_PATH` must contain the path to the shared library of FUSE, usually `$FUSE_HOME/lib`.

- MVAPICH2 needs to be compiled with checkpoint/restart support, see section 4.4.

- FUSE kernel modules must be loaded on all the compute nodes.

- Make sure the PATH contains the path to the executables of FUSE, usually `$FUSE_HOME/bin`.

If write aggregation has been enabled at configuration time, MVAPICH2 will check the FUSE configuration of each node during startup (FUSE module loaded and `fusermount` command in the PATH). If one node is not properly configured, then MVAPICH2 will abort. In this case, you need to fix the FUSE configuration of the nodes, or disable aggregation using `MV2_CKPT_USE_AGGREGATION=0` to run MVAPICH2.

6.11.1.3   Fault Tolerance Backplane (FTB) support: MVAPICH2 supports the new standardized Fault Tolerance Backplane (FTB). FTB can be used for Checkpoint-Restart and Job Pause-Migration-Restart Frameworks. Activating FTB support is optional to perform checkpoint/restart operations, but it will allow MVAPICH2 to automatically publish and listen to standardized events through the Fault Tolerance Backplane.

FTB has been developed and standardized by the CIFTS project. It enables faults to be handled in a coordinated and holistic manner in the entire system, providing for an infrastructure which can be used by different software systems to exchange fault-related information.

If using the FTB framework for checkpoint/restart, the following steps need to be done in addition to the above nodes.

- Download and install the FTB (Fault Tolerance Backplane) package. The packages can be downloaded from here.

- Make sure the FTB packages are installed on every node and the `LD_LIBRARY_PATH` must contain the path to the shared library of FTB, usually `$FTB_HOME/lib`.

- MVAPICH2 needs to be compiled with checkpoint/restart as well as FTB support, see section 4.4.

- Start FTB Database server (`$FTB_HOME/sbin/ftb_database_server`) on one of the nodes, this node will act as server node for all the FTB agents.

- Start FTB agents (`$FTB_HOME/sbin/ftb_agent`) on all the compute nodes.

6.11.1.4   Checkpoint/Restart support for the OFA-IB-Nemesis Interface   MVAPICH2 also provides Checkpoint-Restart capabilities for the OFA-IB-Nemesis interface. Currently, this feature can only be used in conjunction with the Hydra process manager. More details on using Hydra to checkpoint an MPI application can be found here.

6.11.2   Job Pause-Migration-Restart Support

MVAPICH2 provides a node-level Job Pause-Migration-Restart mechanism for the OFA-IB-CH3 interface, which can be used to migrate MPI processes on a given failing node to a healthy node, selected from a pool of spare hosts provided by the user during job-launch.

This Job Migration framework relies on BLCR and FTB libraries. See subsections in 6.11.1 to set up these libraries. The source and target nodes are required to have these libraries installed in the same path.

The Job Migration framework makes use of BLCR to take a checkpoint of all MPI processes running on the failing node in question. Users are strongly recommended to either disable "prelinking" feature, or execute the following command:

```
$ prelink --undo --all
```

on all nodes before starting a job that will later be migrated. Please refer to this BLCR web page for complete information: BLCR FAQ.

During job-launch, the list of spare hosts can be provided as input to mpirun_rsh using the `-sparehosts` option which takes a hostfile as argument.

The following example illustrates a sample MPI job being launched to run with job migration support:

```
$ mpirun_rsh -np 4 -hostfile ./hosts -sparehosts
./spare_hosts ./prog
```

where `spare_hosts` is a file which contains a list of healthy spare hosts that is needed by the job migration framework.

An actual migration of MPI processes from a failing source node to a target spare node can be triggered using one of two methods - using signals or using the `mv2_trigger` utility.

For the signal-triggered method, users can manually initiate the migration protocol by issuing a `SIGUSR2` signal to the `mpispawn` processes running on the source node. This can be done using the following command:

```
$ pkill -SIGUSR2 mpispawn
```

The migration protocol can also be triggered using the simple utility provided with MVAPICH2:

```
$ $PREFIX/bin/mv2_trigger src
```

where `src` is the hostname of the health-deteriorating node from which all MPI processes need to be migrated.

Please note, that the `ftb_agent` daemon will have to be launched on all the compute nodes and the spare nodes before launching the MPI application, in order to successfully migrate MPI processes. See section 6.11.1.3 for information about setting up the FTB infrastructure.

6.11.3 Run-Through Stabilization

MVAPICH2 provides support for run-through stabilization wherein communication failures are not treated as fatal errors. On enabling this capability, MVAPICH2 returns the appropriate error code to a user-set error handler in the event of a communication failure, instead of terminating the entire job. When a process detects a failure when communicating with another process, it will consider the other process as having failed and will no longer attempt to communicate with that process. The user can, however, continue making communication calls to other processes. Any outstanding send or receive operations to a failed process, or wildcard receives (i.e., with MPI_ANY_SOURCE) posted to communicators with a failed process, will be immediately completed with an appropriate error code.

Currently, this support is available only for the OFA-IB-Nemesis and TCP/IP-Nemesis interfaces, when the Hydra process manager is used. It can be enabled at run-time by:

- Setting the environment variable `MV2_RUN_THROUGH_STABILIZATION=1` (see section 12.26), and

- passing the `--disable-auto-cleanup` flag to Hydra launcher.

6.11.4   Network Fault Tolerance with Automatic Path Migration

MVAPICH2 supports network fault recovery by using InfiniBand Automatic Path Migration (APM) mechanism for OFA-IB-CH3 interface. This support is available for MPI applications using OpenFabrics stack and InfiniBand adapters.

To enable this functionality, a run-time variable, MV2_USE_APM (Section 11.75) can be enabled, as shown in the following example:

```
$ mpirun_rsh -np 2 n0 n1 MV2_USE_APM=1 ./cpi
```
or
```
$ mpiexec -n 2 -hosts n0,n1 -env MV2_USE_APM 1 ./cpi
```

MVAPICH2 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 MV2_USE_APM_TEST (Section 11.76). This should be combined with MV2_USE_APM as follows:

```
$ mpirun_rsh -np 2 n0 n1 MV2_USE_APM=1 MV2_USE_APM_TEST=1 ./cpi
```
or
```
$ mpiexec -n 2 -hosts n0,n1 -env MV2_USE_APM 1 -env
MV2_USE_APM_TEST 1 ./cpi
```

6.12   Running with RDMA CM support

In MVAPICH2, for using RDMA CM the run time variable `MV2_USE_RDMA_CM` needs to be used as described in 11. This applies to OFA-IB-CH3 and OFA-iWARP-CH3 interfaces.

In addition to these flags, all the systems to be used need the following one time setup for enabling RDMA CM usage.

- **Setup the RDMA CM interface:** RDMA CM interface needs to be setup, configured with an IP address and connected to the network.

- **Setup the Local Address File:** Create the file (`/etc/mv2.conf`) with the local IP address to be used by RDMA CM. (Multiple IP addresses can be listed (one per line) for multi-rail configurations).
  ```
  $ echo 10.1.1.1 >> /etc/mv2.conf
  ```

Programs can be executed as follows:

```
$ mpirun_rsh -np 2 n0 n1 MV2_USE_RDMA_CM=1 prog
```
or
```
$ mpiexec -n 2 -hosts n0,n1 -env MV2_USE_RDMA_CM 1 prog
```

6.13   Running MVAPICH2 in Multi-threaded Environments

MVAPICH2 binds processes to processor cores for optimal performance. Please refer to Section 6.3 for more details. However, in multi-threaded environments, it might be desirable to have each thread compute using a separate processor core. This is especially true for OpenMP+MPI programs.

In MVAPICH2, processor core mapping is turned off in the following way to enable the application in MPI_THREAD_MULTIPLE threading level if user requested it in MPI_Init_thread. Otherwise, applications will run in MPI_THREAD_SINGLE threading level.

```
$ mpirun_rsh -np 2 n0 n1 MV2_ENABLE_AFFINITY=0 ./openmp+mpi_app
```

For QLogic PSM Interface,

```
$ mpirun_rsh -np 2 n0 n1 MV2_ENABLE_AFFINITY=0
IPATH_NO_CPUAFFINITY=1 ./openmp+mpi_app
```

6.14   Running with Hot-Spot and Congestion Avoidance

MVAPICH2 supports hot-spot and congestion avoidance using InfiniBand multi-pathing mechanism. This support is available for MPI applications using OFA-IB-CH3 interface.

To enable this functionality, a run-time variable, MV2_USE_HSAM (Section 11.81) can be enabled, as shown in the following example:

```
$ mpirun_rsh -np 2 n0 n1 MV2_USE_HSAM=1 ./cpi
```
or
```
$ mpiexec -n 2 -hosts n0,n1 -env MV2_USE_HSAM 1 ./cpi
```

This functionality automatically defines the number of paths for hot-spot avoidance. Alternatively, the maximum number of paths to be used between a pair of processes can be defined by using a run-time variable MV2_NUM_QP_PER_PORT (Section 11.40).

We expect this functionality to show benefits in the presence of at least partially non-overlapping paths in the network. OpenSM, the subnet manager distributed with OpenFabrics supports LMC mechanism, which can be used to create multiple paths:

```
$ opensm -l4
```

will start the subnet manager with LMC value to four, creating sixteen paths between every pair of nodes.

6.15   Running on Clusters with NVIDIA GPU Accelerators

MVAPICH2 CH3-based interface supports MPI communication using NVIDIA GPU device memory with CUDA 4.0 or CUDA 4.1 library. This feature removes the need for the application developer to explicitly move the data from device memory to host memory before using MPI for communication. The new support allows direct MPI communication from device memory to device memory, device memory to host memory and host memory to device memory. It also supports point-to-point and collective communication using

contiguous and non-contiguous MPI datatypes. It takes advantage of CUDA IPC for intra-node GPU-GPU communication (with CUDA 4.1).

For example, before MVAPICH2-1.8, a typical user might be using the following sequence of commands to move data from a device memory to another device memory.

...
cudaMemcpy(host_buf, device_buf, size, cudaMemcpyDeviceToDevice);
MPI_Isend(host_buf, size, MPI_CHAR, 1, 100, MPI_COMM_WORLD, req);
...

With the support provided in MVAPICH2-1.8 and support of CUDA 4.0 (and later), the user can achieve the same data movement operation by explicitly specifying MPI calls on device memory.
...
MPI_Isend(device_buf, size, MPI_CHAR, 1, 100, MPI_COMM_WORLD, req);
...

This support can be enabled by configuring MVAPICH2 with `--enable-cuda` and setting the environment variable MV2_USE_CUDA (11.121) to 1 during runtime.

To minimize communication overhead, MVAPICH2 divides copies between device and host into chunks. This can be better tuned with a runtime environment variable MV2_CUDA_BLOCK_SIZE (11.122). The default chunk size is 64K (65536). However, higher values of this parameter, such as 256K (262144) and 512K (524288), might deliver better performance if the MPI application uses large messages. The optimal value for this parameter depends on several other factors such as InfiniBand network/adapter speed, GPU adapter characteristics, platform characteristics (processor and memory speed) and amount of memory to be dedicated to the MPI library with GPU support. For different platforms and MPI applications, the users are encouraged to try out different values for this parameter to get best performance.

MVAPICH2 uses network loopback for intra node communication when MV2_USE_CUDA(11.121) is enabled. In order to use MVAPICH2 NVIDIA GPU features on stand alone multi-core GPU nodes that are not equipped with InfiniBand adapters, users need to enable shared memory through the following run time environment variables: MV2_USE_SHARED_MEM=1 MV2_SMP_SEND_BUF_SIZE=262144. The parameter MV2_SMP_SEND_BUF_SIZE(11.111) controls the size of copies used in large message shared memory communication. Users can better tune based on the requirement. Some suggested values are 65536, 131072 or 524288.

6.15.0.1 GPU Affinity: When multiple GPUs are present on a node, users might want to set the MPI process affinity to a particular GPU using cuda calls like cudaSetDevice(). This can be done after MPI_Init based on MPI rank of the process. But MVAPICH2 performs some cuda operations like buffer registration and others during MPI_Init which result in default context creation. Hence, setting GPU Affinity after MPI_Init could create issues due to the context switch. To avoid this, MVAPICH2 provides an environment variable called MV2_COMM_WORLD_LOCAL_RANK to get the local rank of a process on its node before MPI_Init is called. This local rank information can be used to set GPU affinity before MPI_Init is called as given in the following code example

. . .
....
int local_rank = atoi(getenv("MV2_COMM_WORLD_LOCAL_RANK"));
cudaSetDevice(local_rank % num_devices);
...
MPI_Init()
...
. . .

This local rank information can also be used in wrapper scripts to set cpu and memory affinity on nodes with NUMA.

6.16   MPIRUN_RSH compatibility with MPIEXEC

There is now a front end to mpirun_rsh that aims for compatibility with mpiexec usage as recommended by the MPI standard. This front end is can be used by calling mpiexec.mpirun_rsh.

The main differences between this and mpirun_rsh is that this front end doesn't require you to specify hosts or environment variables on the command line. Also, the host file options is -f as opposed to -hostfile.

Run 4 instances of CPI on localhost:

```
$ mpiexec.mpirun_rsh -n 4 ./cpi
```

Run 4 instances of CPI on the nodes specified by hostfile:

```
$ mpiexec.mpirun_rsh -n 4 -f hostfile ./cpi
```

6.16.1   Interaction with SLURM

If the hostfile isn't given and mpiexec.mpirun_rsh finds the appropriate slurm environment variables set, mpiexec will use these to determine which nodes the applications should be launched on.

Please note that processes are assigned in a block fashion by slurm so all the processes will run on the same node if possible.

Run 4 instances of CPI on up to 4 different nodes (but maybe only 1) allocated by "salloc":

```
$ salloc -N 4
$ mpiexec.mpirun_rsh -n 4 ./cpi
```

Same thing as above but only allocating the minimum number of nodes needed to run 4 tasks:

```
$ salloc -n 4
$ mpiexec.mpirun_rsh ./cpi
```

Run 4 instances of CPI on 4 different nodes allocated by "salloc":

```
$ salloc -N 4 --ntasks-per-node=1
$ mpiexec.mpirun_rsh -n 4 ./cpi
```

6.16.2   Interaction with PBS

If the hostfile isn't given and mpiexec.mpirun_rsh finds the PBS_NODEFILE environment variable set, mpiexec will use this to locate the host file.

7 OSU Benchmarks

If you have arrived at this point, you have successfully installed MVAPICH2. Congratulations!! The OSU benchmarks should already be built and installed along with MVAPICH2. Look for them under $prefix/libexec/osu-micro-benchmarks. The benchmarks provided are:

<div align="center">MPI-1 and MPI-2</div>

osu_bibw	Bidirectional Bandwidth Test
osu_bw	Bandwidth Test
osu_latency	Latency Test
osu_mbw_mr	Multiple Bandwidth / Message Rate Test
osu_multi_lat	Multi-pair Latency Test
osu_allgather	MPI_Allgather Latency Test (*)
osu_allgatherv	MPI_Allgatherv Latency Test
osu_allreduce	MPI_Allreduce Latency Test
osu_alltoall	MPI_Alltoall Latency Test
osu_alltoallv	MPI_Alltoallv Latency Test
osu_barrier	MPI_Barrier Latency Test
osu_bcast	MPI_Bcast Latency Test
osu_gather	MPI_Gather Latency Test (*)
osu_gatherv	MPI_Gatherv Latency Test
osu_reduce	MPI_Reduce Latency Test
osu_reduce_scatter	MPI_Reduce_scatter Latency Test
osu_scatter	MPI_Scatter Latency Test(*)
osu_scatterv	MPI_Scatterv Latency Test

<div align="center">MPI-2 only</div>

osu_acc_latency	Accumulate Latency Test with Active Synchronization
osu_get_bw	One-Sided Get Bandwidth Test with Active Synchronization
osu_get_latency	One-Sided Get Latency Test with Active Synchronization
osu_latency_mt	Multi-threaded Latency Test
osu_passive_acc_latency	Latency Test for Accumulate with Passive Synchronization
osu_passive_get_bw	Bandwidth Test for Get with Passive Synchronization
osu_passive_get_latency	Latency Test for Get with Passive Synchronization
osu_passive_put_bw	Bandwidth Test for Put with Passive Synchronization
osu_passive_put_latency	Latency Test for Put with Passive Synchronization
osu_put_bibw	One-Sided Put Bidirectional Test with Active Synchronization
osu_put_bw	One-Sided Put Bandwidth Test with Active Synchronization
osu_put_latency	One-Sided Put Latency Test with Active Synchronization

(* OMB 3.6 version uses the MPI-2 MPI_IN_PLACE feature for three collective benchmarks - osu_allgather, osu_gather and osu_scatter).

More information about the benchmarks can be found at http://mvapich.cse.ohio-state.edu/benchmarks/. You can also check this link for updates to the benchmarks.

7.1 Building

The OSU Benchmarks should already be built and installed along with the MVAPICH2 library as a matter of convenience. They are installed using the same prefix as the library and can be found at $prefix/libexec/osu-micro-benchmarks. For example, if you build mvapich2 using './configure –prefix=/usr/local/mvapich2-1.8', then you can find the benchmarks at /usr/local/mvapich2-1.8/libexec/osu-micro-benchmarks.

You can also build the benchmarks separately. Using the following steps if mpicc is in your PATH. For example:

```
$ ./configure && make && make install
```

If mpicc is not in your path or you would like to use another particular version you can explicitly tell configure by setting CC. For example:

```
$ ./configure CC=/path/to/special/mpicc && make && make install
```

Configure will detect whether your library supports MPI-2 and compile the corresponding benchmarks. Sample performance numbers for these benchmarks on representative platforms with InfiniBand, iWARP and RoCE adapters 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 us know by sending an email to mvapich-discuss@cse.ohio-state.edu.

7.2 Running

The OSU Benchmarks are run in the same manner as other MPI Applications. The following examples will use mpirun_rsh as the process manager. Please see section 5.2 for more information on running with other process managers.

7.2.1 Running OSU Latency and Bandwidth

Inter-node latency and bandwidth: The following example will measure the latency and bandwidth of communication between node1 and node2.

```
$ mpirun_rsh -np 2 node1 node2 ./osu_latency
$ mpirun_rsh -np 2 node1 node2 ./osu_bw
```

Intra-node latency and bandwidth: The following example will measure the latency and bandwidth of communication inside node1 on different cores. This assumes that you have at least two cores (or processors) in your node.

```
$ mpirun_rsh -np 2 node1 node1 ./osu_latency
$ mpirun_rsh -np 2 node1 node1 ./osu_bw
```

7.2.2   Running OSU Message Rate Benchmark

The OSU message rate benchmark reports the rate at which messages can be sent between two nodes. It is advised that it should be run in a configuration that utilizes multiple pairs of communicating processes on two nodes. The following example measures the message rate on a system with two nodes, each with four processor cores.

```
$ mpirun_rsh -np 8 -hostfile mf ./osu_mbw_mr
```

Where hostfile "mf" has the following contains:

node1
node1
node1
node1
node2
node2
node2
node2

7.2.3   Running OSU Collective Benchmarks

By default, the OSU collective benchmarks report the average communication latencies for a given collective operation, across various message lengths. Additionally, the benchmarks offer the following options:

1. "-f" can be used to report additional statistics, such as min and max latencies and the number of iterations.

2. "-m" option can be used to set the maximum message length to be used in a benchmark. In the default version, the benchmarks report the latencies for up to 1MB message lengths.

3. "-i" can be used to set the number of iterations to run for each message length.

4. "-h" can be used to list all the options and their descriptions.

5. "-v" reports the benchmark version.

If a user wishes to measure the communication latency of a specific collective operation, say, MPI_Alltoall, with 16 processes, we recommend running the osu_alltoall benchmark in the following manner:

```
$ mpirun_rsh -np 16 -hostfile mf ./osu_alltoall
```

Where hostfile "mf" has the following contains:

node1
node1
node1
node1

node1
node1
node1
node1
node1
node2
node2
node2
node2
node2
node2
node2
node2

7.3   CUDA Extensions to OMB

CUDA Extensions to OMB can be enable by configuring the benchmark suite with –enable-cuda option as shown below. The MPI library used should be able to support MPI_Send/MPI_Recv from buffers in GPU Device memory.

```
$ ./configure CC=/path/to/mpicc
    --enable-cuda
    --with-cuda-include=/path/to/cuda/include
    --with-cuda-libpath=/path/to/cuda/lib
$ make
$ make install
```

The following benchmarks have been extended to evaluate performance of MPI_Send/MPI_Recv from and to buffers on NVIDIA GPU devices.

osu_bibw	Bidirectional Bandwidth Test
osu_bw	Bandwidth Test
osu_latency	Latency Test

Each of these benchmarks takes two input parameters. The first parameter indicates the location of the buffers at rank 0 and the second parameter indicates the location of the buffers at rank 1. The value of each of these parameters can be either 'H' or 'D' to indicate if the buffers are to be on the host or on the device, respectively. When no parameters are specified, the buffers are allocated on the host.

GPU affinity for processes should be set before MPI_Init is called. The process rank on a node is normally used to do this and different MPI launchers expose this information through different environment variables. The benchmarks use an environment variable called LOCAL_RANK to get this information. The script like get_local_rank provided alongside the benchmarks can be used to export this environment variable when using mpirun_rsh. This can be adapted to work with other MPI launchers and libraries.

Examples:

```
$ mpirun_rsh -np 2 host0 host0 MV2_USE_CUDA=1
get_local_rank ./osu_latency D D
```

In this run, assuming host0 has two GPU devices, the latency test allocates buffers on GPU device 0 at rank 0 and on GPU device 1 at rank 1.

```
$ mpirun_rsh -np 2 host0 host1 MV2_USE_CUDA=1
get_local_rank ./osu_bw D H
```

In this run, the bandwidth test allocates buffers on the GPU device at rank 0 (host0) and on the host at rank 1 (host1).

8   Scalability features and Performance Tuning for Large Scale Clusters

MVAPICH2 provides many different parameters for tuning performance for a wide variety of platforms and applications. This section deals with tuning CH3-based interfaces. These parameters can be either compile time parameters or runtime parameters. Please refer to Section 8 for a complete description of all these 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

Starting with version 1.2, MVAPICH2 has a new, scalable job launcher – 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 MV2_MT_DEGREE (see 13.8).

MVAPICH2 can use a faster, hierarchical launching mechanism on large clusters. This is enabled manually using MV2_FASTSSH_THRESHOLD 13.5).

When the number of nodes involved is beyond 8k, the mpirun_rsh uses a file-based communication scheme to create the hierarchical tree. The default value can be overridden with the environment variable MV2_NPROCS_THRESHOLD (see 13.6).

8.2   Basic QP Resource Tuning

The following parameters affect memory requirements for each QP.

- MV2_DEFAULT_MAX_SEND_WQE

- MV2_DEFAULT_MAX_RECV_WQE

- MV2_MAX_INLINE_SIZE

MV2_DEFAULT_MAX_SEND_WQE and MV2_DEFAULT_MAX_RECV_WQE control the maximum number of WQEs per QP and MV2_MAX_INLINE_SIZE controls the maximum inline size. Reducing the values of these two parameters leads to less memory consumption. They are especially important for large scale clusters with a large amount of connections and multiple rails.

These two parameters are run-time adjustable. Please refer to Sections 11.18 and 11.34 for details.

8.3   RDMA Based Point-to-Point Tuning

The following parameters are important in tuning the memory requirements for adaptive rdma fast path feature.

- MV2_RDMA_FAST_PATH_BUF_SIZE (11.44)

- MV2_NUM_RDMA_BUFFER (11.45)

MV2_RDMA_FAST_PATH_BUF_SIZE is the size of each buffer used in RDMA fast path communication.

MV2_NUM_RDMA_BUFFER is number of buffers used for the RDMA fast path communication.

On the other hand, the product of MV2_RDMA_FAST_PATH_BUF_SIZE and MV2_NUM_RDMA_BUFFER generally is a measure of the amount of memory registered for eager message passing. These buffers are not shared across connections.

8.4   Shared Receive Queue (SRQ) Tuning

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

- MV2_SRQ_MAX_SIZE (11.71)

- MV2_SRQ_SIZE (11.72)

- MV2_SRQ_LIMIT (11.70)

MV2_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. The application will start by only using MV2_SRQ_SIZE buffers (default 256) and will double this value on every SRQ limit event(upto MV2_SRQ_MAX_SIZE). For long running applications this re-size should show little effect. If needed, the MV2_SRQ_SIZE can be increased to 1024 or higher as needed for applications. MV2_SRQ_LIMIT defines the low water-mark for the flow control handler. This can be reduced if your aim is to reduce the number of interrupts.

8.5   eXtended Reliable Connection (XRC)

MVAPICH2 now supports the eXtended Reliable Connection (XRC) transport available in recent Mellanox HCAs. This transport helps reduce the number of QPs needed on multi-core systems. Set MV2_USE_XRC (11.104) to use XRC with MVAPICH2.

8.6   Shared Memory Tuning

MVAPICH2 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 SMPI_LENGTH_QUEUE (Section 11.109), and SMP_EAGER_SIZE (Section 11.108). The two main parameters which are used for tuning shared memory performance for large messages are SMP_SEND_BUF_SIZE (Section 11.111) and SMP_NUM_SEND_BUFFER (Section 11.110).

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

Messages larger than 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. SMP_NUM_SEND_BUFFER is the number of send buffers.

8.7   On-demand Connection Management Tuning

MVAPICH2 uses on-demand connection management to reduce the memory usage of MPI library. There are 4 parameters to tune connection manager: MV2_ON_DEMAND_THRESHOLD (Section 11.46), MV2_CM_RECV_BUFFERS (Section 11.10), MV2_CM_TIMEOUT (Section 11.12), and MV2_CM_SPIN_COUNT (Section 11.11). The first one applies to OFA-IB-CH3, OFA-iWARP-CH3 and uDAPL-CH3 interfaces and the other three only apply to OFA-IB-CH3 interface.

MV2_ON_DEMAND_THRESHOLD defines threshold for enabling on-demand connection management scheme. When the size of the job is larger than the threshold value, on-demand connection management will be used.

MV2_CM_RECV_BUFFERS defines the number of buffers used by connection manager to establish new connections. These buffers are quite small and are shared for all connections, so this value may be increased to 8192 for large clusters to avoid retries in case of packet drops.

MV2_CM_TIMEOUT is the timeout value associated with connection management messages via UD channel. Decreasing this value may lead to faster retries but at the cost of generating duplicate messages.

MV2_CM_SPIN_COUNT is the number of the connection manager polls for new control messages from UD channel for each interrupt. This may be increased to reduce the interrupt overhead when many incoming control messages from UD channel at the same time.

8.8   Scalable Collectives Tuning

MVAPICH2 uses shared memory to optimize the performance for many collective operations: MPI_Allreduce, MPI_Reduce, MPI_Barrier, and MPI_Bcast.

We use shared-memory based collective for most small and medium sized messages and fall back to the default point-to-point based algorithms for very large messages. The upper-limits for shared-memory based collectives are tunable parameters that are specific to each collective operation. We have variables such as MV2_SHMEM_ALLREDUCE_MSG (11.61), MV2_SHMEM_REDUCE_MSG (11.67) and MV2_SHMEM_BCAST_MSG (11.63), for MPI_Allreduce, MPI_Reduce and MPI_Bcast collective operations. The default values for these variables have been set through experimental analysis on some of our clusters and a few large scale clusters, such as the TACC Ranger. Users can choose to set these variables at job-launch time to tune the collective algorithms on different platforms.

8.8.1 Optimizations for MPI_Bcast

MVAPICH2 supports a 2-level point-to-point tree-based "Knomial" algorithm for small messages for the
MPI_Bcast operation. The optimal threshold between this algorithm and the conventional binomial-tree algorithm varies across platforms and system sizes. Users can set the
MV2_KNOMIAL_2LEVEL_BCAST_MESSAGE_SIZE_THRESHOLD (11.30) and the
MV2_KNOMIAL_2LEVEL_BCAST_SYSTEM_SIZE_THRESHOLD (11.31) parameters to select the message size and the system size thresholds for using the knomial-based algorithm. With these settings, the
normal binomial algorithm will be used for system sizes smaller than the chosen value for all small messages. For larger systems, the new knomial algorithm will be used for message sizes smaller than the value
assigned to the
MV2_KNOMIAL_2LEVEL_BCAST_MESSAGE_SIZE_THRESHOLD parameter. Users can also choose
the inter-node and intra-node k-degree of the knomial bcast algorithm by using the parameters
MV2_KNOMIAL_INTER_NODE_FACTOR (11.29) and
MV2_KNOMIAL_INTRA_NODE_FACTOR (11.28).

8.8.2 Optimizations for MPI_Reduce and MPI_Allreduce

In this release, we have introduced new 2-level algorithms for MPI_Reduce and MPI_Allreduce operations,
along the same lines as MPI_Bcast. Pure Shared-memory based algorithms cannot be used for larger messages for reduction operations, because the node-leader processes become the bottleneck, as they have to
perform the reduction operation on the entire data block. We now rely on shared-memory algorithms for
MPI_Reduce and MPI_Allreduce for message sizes set by the thresholds MV2_SHMEM_ALLREDUCE_MSG
(11.61), MV2_SHMEM_REDUCE_MSG (11.67), the new 2-level algorithms for medium sized messages and
the default point-to-point based algorithms for large messages. We have introduced two new run-time variables MV2_ALLREDUCE_2LEVEL_MSG (11.1) and MV2_REDUCE_2LEVEL_MSG (11.57) to determine
when to fall back to the default point-to-point based algorithms.

8.8.3 Optimizations for MPI_Gather and MPI_Scatter

MVAPICH2 supports two new optimized algorithms for MPI_Gather and MPI_Scatter operations – the "Direct" and the multi-core aware "2-level" algorithms. Both these algorithms perform significantly better
than the default binomial-tree pattern. The "Direct" algorithm is however inherently not very scalable
and can be used when the communicator size is less than 1K processes. We switch over to the 2-level
algorithms for larger system sizes. For MPI_Gather, we use different algorithms depending of the system size. For small system sizes (up to 386 cores), we use the "2-level" algorithm following by the "Direct" algorithm. For medium system sizes (up to 1k), we use "Binomial" algorithm following by the "Direct" algorithm. It's possible to set the switching point between algorithms using the run-time parameter
MV2_GATHER_SWITCH_PT (11.99). For MPI_Scatter, when the system size is lower than 512 cores, we
use the "Binomial" algorithm for small message sizes following by the "2-level" algorithm for medium
message sizes and the "Direct" algorithm for large message sizes. Users can define the threshold for small
and medium message sizes using the run-time parameters MV2_SCATTER_SMALL_MSG (11.100) and
MV2_SCATTER_MEDIUM_MSG (11.101). Users can also choose to use only one of these algorithms by
toggling the run-time parameters 11.79 and 11.102 for MPI_Gather and 11.80 and 11.103 for MPI_Scatter.

8.9   Process Placement on Multi-core platforms

Process placement has a significant impact on performance of applications. Depending on your application communication patterns, various process placements can result in performance gains. In Section 6.3, we have described the usage of "bunch" and "scatter" placement modes provided by MVAPICH2. Using these modes, one can control the placement of processes within a particular node. Placement of processes across nodes can be controlled by adjusting the order of MPI ranks. For example, the following command launches jobs in block fashion.

```
$ mpirun_rsh -np 4 n0 n0 n1 n1 MV2_CPU_BINDING_POLICY=bunch
./a.out
```

The following command launches jobs in a cyclic fashion.

```
$ mpirun_rsh -np 4 n0 n1 n0 n1 MV2_CPU_BINDING_POLICY=scatter
./a.out
```

We have noted that the HPL (High-Performance Linpack) benchmark performs better when using block distribution.

8.10   HugePage Support

MVAPICH2 uses HugePages(2MB) by default for communication buffers if they are configured on the system. The run-time variable, MV2_USE_HUGEPAGES(11.112) can be used to control the behavior of this feature.

In order to use HugePages, Make sure HugePages are configured on all nodes. The number of HugePages can be configured by setting vm.nr_hugepages kernel parameter to a suitable value. For example, to allocate a 1GB HugePage pool, execute(as root):

```
$ echo 512 > /proc/sys/vm/nr_hugepages
or
$ sysctl -w vm.nr_hugepages = 512
```

9   FAQ and Troubleshooting with MVAPICH2

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 us by sending an email to mvapich-discuss@cse.ohio-state.edu.

MVAPICH2 can be used over eight underlying interfaces, namely OFA-IB-CH3, OFA-IB-Nemesis, OFA-IWARP-CH3, OFA-RoCE-CH3, PSM-CH3, uDAPL-CH3, TCP/IP-CH3 and TCP/IP-Nemesis. Based on the underlying library being utilized, the troubleshooting steps may be different. We have divided the troubleshooting tips into four sections: General troubleshooting and Troubleshooting over any one of the five transport interfaces.

9.1   General Questions and Troubleshooting

9.1.1   Invalid Communicators Error

This is a problem which typically occurs due to the presence of multiple installations of MVAPICH2 on the same set of nodes. The problem is due to the presence of `mpi.h` other than the one, which is used for executing the program. This problem can be resolved by making sure that the `mpi.h` from other installation is not included.

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

`fork()` and `system()` is supported for the OpenFabrics device 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.

9.1.3   MPI+OpenMP shows bad performance

MVAPICH2 uses CPU affinity to have better performance for single-threaded programs. For multi-threaded programs, e.g. MPI+OpenMP, 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, i.e. set `MV2_ENABLE_AFFINITY` to 0. In addition, please read Section 6.13 on using MVAPICH2 in multi-threaded environments.

9.1.4   Error message "No such file or directory" when using Lustre file system

If you are using ADIO support for Lustre, please make sure of the following:
– Check your Lustre setup
– You are able to create, read to and write from files in the Lustre mounted directory
– The directory is mounted on all nodes on which the job is executed
– The path to the file is correctly specified
– The permissions for the file or directory are correctly specified

9.1.5   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. Please include the following option with your lustre mount command: "-o localflock".

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

9.1.6   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" and exported into the environment before using the mpiexec or mpirun_rsh command to launch the program, as below:

```
$ export GFORTRAN_UNBUFFERED_ALL=y
```

Or, if using mpirun_rsh, export the environment variable as in the example:

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

9.1.7   How do I obtain MVAPICH2 version and configuration information?

The mpiname application is provided with MVAPICH2 to assist with determining the MPI library version and related information. The usage of mpiname is as follows:

```
$ mpiname [OPTION]
```

Print MPI library information. With no OPTION, the output is the same as -v.

-a print all information

-c print compilers

-d print device

-h display this help and exit

-n print the MPI name

-o print configuration options

-r print release date

-v print library version

9.1.8   How do I compile my MPI application with static libraries, and not use shared libraries?

Even if MVAPICH2 installed on your cluster was configured to use shared libraries, it is possible to use static libraries. Use the command below to use static libraries:

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

9.1.9   Does MVAPICH2 work across AMD and Intel systems?

Yes, as long as you compile MVAPICH2 and your programs on one of the systems, either AMD or Intel, and run the same binary across the systems. MVAPICH2 has platform specific parameters for performance optimizations and it may not work if you compile MVAPICH2 and your programs on different systems and try to run the binaries together.

9.1.10   I want to enable debugging for my build. How do I do this?

We recommend that you enable debugging when you intend to take a look at back traces of processes in GDB (or other debuggers). You can use the following configure option to enable debugging: `--enable-g=dbg`.

Additionally:

- See parameter `MV2_DEBUG_CORESIZE` (section 10.4) to enable core dumps.
- See parameter `MV2_DEBUG_SHOW_BACKTRACE` (section 10.5) to show a basic backtrace in case of error.

9.1.11   How can I run my application with a different group ID?

You can specify a different group id for your MPI application using the `-sg group` option to `mpirun_rsh`. The following example executes *a.out* on host1 and host2 using *secondarygroup* as their group id.

```
$ mpirun_rsh -sg secondarygroup -np 2 host1 host2 ./a.out
```

9.2   Issues and Failures with Job launchers

9.2.1   Can I use MVAPICH2 on PBS/Torque Clusters

You can use MVAPICH2 for clusters administered by PBS/Torque. If you are a cluster user (not an administrator), please ask your cluster administrator to install the OSC mpiexec. If you are a cluster administrator, please follow the instructions below.

You will need to download `mpiexec` from Ohio Supercomputer enter (OSC) at the following link.

Please note that this mpiexec is different from the mpiexec provided within MPICH2. Also note that you do not need to use either mpirun_rsh or mpiexec.hydra on a cluster that is administered with PBS/Torque. You

may also choose to remove the MVAPICH2 mpiexecs from the install path to shield your users from making a mistake of trying to run MPI jobs with a wrong launcher.

Usage of OSC mpiexec is simple. An example of using this launcher is as follows:

```
$ /path/to/osc/mpiexec -np 4 ./a.out
```

9.2.2   /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 MVAPICH2 executables are in the PATH on all nodes. If PATHs cannot be setup as mentioned, then invoke mpirun_rsh with a path prefix. For example:

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

9.2.3   TotalView complains that "The MPI library contains no suitable type definition for struct MPIR_PROCDESC"

Ensure that the MVAPICH2 job launcher mpirun_rsh is compiled with debug symbols. Details are available in Section 5.2.11.

9.3   Problems Building MVAPICH2

9.3.1   Unable to convert MPI_SIZEOF_AINT to a hex string

```
configure:  error:  Unable to convert MPI_SIZEOF_AINT to a hex string.
This is either because we are building on a very strange platform or there
is a bug somewhere in configure.
```

This error can be misleading. The problem is often not that you're building on a strange platform, but that there was some problem running an executable that made configure have trouble determining the size of a datatype. The true problem is often that you're trying to link against a library that is not found in your system's default path for linking at runtime. Please check that you've properly set LD_LIBRARY_PATH or used the correct rpath settings in LDFLAGS.

9.3.2   Cannot Build with the PathScale Compiler

There is a known bug with the PathScale compiler (before version 2.5) when building MVAPICH2. This problem will be solved in the next major release of the PathScale compiler. To work around this bug, use the the "`-LNO:simd=0`" C compiler option. This can be set in the build script similarly to:

```
export CC="pathcc -LNO:simd=0"
```

Please note the use of double quotes. If you are building shared libraries and are using the PathScale compiler (version below 2.5), then you should add "-g" to your CFLAGS, in order to get around a compiler bug.

9.3.3 Libtool has a problem linking with non-GNU compiler (like PGI)

If you are using a compiler that is not recognized by autoconf as a GNU compiler, Libtool uses an default library search path to look for shared objects which is "/lib /usr/lib /usr/local/lib". Then, if your libraries are not in one of these paths, MVAPICH2 may fail to link properly.

You can work around this issue by adding the following configure flags:

```
./configure \
lt_cv_sys_lib_search_path_spec="/lib64 /usr/lib64 /usr/local/lib64"  \
lt_cv_sys_lib_dlsearch_path_spec="/lib64 /usr/lib64 /usr/local/lib64" \
... ...
```

The above example considers that the correct library search path for your system is "/lib64 /usr/lib64 /usr/local/lib64".

9.4   With OFA-IB-CH3 Interface

9.4.1   Cannot Open HCA

The above error reports that the InfiniBand Adapter is not ready for communication. Make sure that the drivers are up. This can be done by executing the following command which gives the path at which drivers are setup.

```
$ locate libibverbs
```

9.4.2   Checking state of IB Link

In order to check the status of the IB link, one of the OFED utilities can be used: `ibstatus`, `ibv_devinfo`.

9.4.3   Creation of CQ or QP failure

A possible reason could be inability to pin the memory required. Make sure the following steps are taken.

1. In `/etc/security/limits.conf` add the following

   ```
   * soft memlock phys_mem_in_KB
   ```

2. After this, add the following to `/etc/init.d/sshd`

```
    ulimit -l  phys_mem_in_KB
```

3. Restart sshd

With some distros, we've found that adding the ulimit -l line to the sshd init script is no longer necessary. For instance, the following steps work for our RHEL5 systems.

1. Add the following lines to `/etc/security/limits.conf`

```
    * soft memlock unlimited
    * hard memlock unlimited
```

2. Restart sshd

9.4.4   Hang with the HSAM Functionality

HSAM functionality uses multi-pathing mechanism with LMC functionality. However, some versions of OpenFabrics Drivers (including OpenFabrics Enterprise Distribution (OFED) 1.1) and using the Up*/Down* routing engine do not configure the routes correctly using the LMC mechanism. We strongly suggest to upgrade to OFED 1.2, which supports Up*/Down* routing engine and LMC mechanism correctly.

9.4.5   Failure with Automatic Path Migration

MVAPICH2 (OFA-IB-CH3) 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 OpenFabrics drivers, APM functionality is not completely supported. Please refer to Section 11.75 and section 11.76

9.4.6   Error opening file

If you configure MVAPICH2 with RDMA_CM and see this error, you need to verify if you have setup up the local IP address to be used by RDMA_CM in the file `/etc/mv2.conf.` Further, you need to make sure that this file has the appropriate file read permissions. Please follow Section 6.12 for more details on this.

9.4.7   RDMA CM Address error

If you get this error, please verify that the IP address specified `/etc/mv2.conf` is correctly specified with the IP address of the device you plan to use RDMA_CM with.

9.4.8   RDMA CM Route error

If see this error, you need to check whether the specified network is working or not.

9.5   With OFA-iWARP-CH3 Interface

9.5.1   Error opening file

If you configure MVAPICH2 with RDMA_CM and see this error, you need to verify if you have setup up the local IP address to be used by RDMA_CM in the file `/etc/mv2.conf`. Further, you need to make sure that this file has the appropriate file read permissions. Please follow Section 5.2.6 for more details on this.

9.5.2   RDMA CM Address error

If you get this error, please verify that the IP address specified `/etc/mv2.conf` is correctly specified with the IP address of the device you plan to use RDMA_CM with.

9.5.3   RDMA CM Route error

If see this error, you need to check whether the specified network is working or not.

9.6   With uDAPL-CH3 Interface

9.6.1   Cannot Open IA

If you configure MVAPICH2 with uDAPL and see this error, you need to check whether you have specified the correct uDAPL service provider (Section 5.2.8). If you have specified the uDAPL provider but still see this error, you need to check whether the specified network is working or not. If you are using OpenFabrics software stack on Linux, the default DAPL provider is `OpenIB-cma` for DAPL-1.2, and `ofa-v2-ib0` for DAPL-2.0. If you are using Solaris, the default DAPL provider is `ibd0`.

9.6.2   DAT Insufficient Resource

If you configure MVAPICH2 with uDAPL and see this error, you need to reduce the value of the environmental variables `RDMA_DEFAULT_MAX_SEND_WQE` and/or `RDMA_DEFAULT_MAX_RECV_WQE` depending on the underlying network.

9.6.3   Cannot Find libdat.so

If you get the error: "error while loading shared libraries, libdat.so", The location of the dat shared library is incorrect. You need to find the correct path of libdat.so and export `LD_LIBRARY_PATH` to this correct location. For example:

```
$ mpirun_rsh -np 2 n1 n2 LD_LIBRARY_PATH=/path/to/libdat.so
./a.out
```

9.6.4   uDAPL over IB Does Not Scale Beyond 256 Nodes with RDMA_CM Provider

We recommend that uDAPL IB consumers needing large scale-out use socket cm provider (libdaplscm.so) in lieu of rdma_cm (libdaplcma.so). iWARP users can remain using uDAPL rdma_cm provider. For detailed discussion of this issue please refer to:

http://lists.openfabrics.org/pipermail/general/2008-June/051814.html

9.7   Checkpoint/Restart

9.7.1   Failure during Restart

Please make sure the following things for a successful restart:

- The BLCR modules must be loaded on all the compute nodes and the console node before a restart

- The checkpoint file of MPI job console must be accessible from the console node.

- The corresponding checkpoint files of the MPI processes must be accessible from the compute nodes using the same path as when checkpoint was taken.

The following things can cause a restart to fail:

- The job which was checkpointed is not terminated or the some processes in that job are not cleaned properly. Usually they will be cleaned automatically, otherwise, since the pid can't be used by BLCR's to restart, restart will fail.

- The processes in the job have opened temporary files and these temporary files are removed or not accessible from the nodes where the processes are restarted on.

- If the processes are restarted on different nodes, then all the nodes must have the exact same libraries installed. In particular, you may be required to disable any "prelinking". Please look at `https://upc-bugs.lbl.gov//blcr/doc/html/FAQ.html#prelink` for further details.

FAQ regarding Berkeley Lab Checkpoint/Restart (BLCR) can be found at: http://upc-bugs.lbl.gov/blcr/doc/html/FAQ.html And the user guide for BLCR can be found at http://upc-bugs.lbl.gov/blcr/doc/html/BLCR_Users_Guide.html

If you encounter any problem with the Checkpoint/Restart support, please feel free to contact us at mvapich-discuss@cse.ohio-state.edu.

10 MVAPICH2 General Parameters

10.1 MV2_IGNORE_SYSTEM_CONFIG

- Class: Run time
- Default: 0

If set, the system configuration file is not processed.

10.2 MV2_IGNORE_USER_CONFIG

- Class: Run time
- Default: 0

If set, the user configuration file is not processed.

10.3 MV2_USER_CONFIG

- Class: Run time
- Default: Unset

Specify the path of a user configuration file for mvapich2. If this is not set the default path of " /.mvapich2.conf" is used.

10.4 MV2_DEBUG_CORESIZE

- Class: Run time
- Default: Unset
- Possible values: Positive integer or "unlimited"

Set the limit for the core size resource. It allows to specify the maximum size for a core dump to be generated. It only set the soft limit and it has the respect the hard value set on the nodes.

It is similar to the `ulimit -c <coresize>` that can be run in the shell, but this will only apply to the MVAPICH2 processes (MPI processes, mpirun_rsh, mpispawn).

Examples:

- 'MV2_DEBUG_CORESIZE=0' will disable core dumps for MVAPICH2 processes.
- 'MV2_DEBUG_CORESIZE=unlimited' will enable core dumps for MVAPICH2 processes.

10.5 MV2 DEBUG SHOW BACKTRACE

- Class: Run time

- Default: 0 (disabled)

- Possible values: 1 to enable, 0 to disable

Show a backtrace when a process fails on errors like "Segmentation faults", "Bus error", "Illegal Instruction", "Abort" or "Floating point exception".

If your application uses the static version of the MVAPICH2 library, you have to link your application with the `-rdynamic` flag in order to see the function names in the backtrace. For more information, see the backtrace manpage.

11   MVAPICH2 Parameters (CH3-Based Interfaces)

11.1   MV2_ALLREDUCE_2LEVEL_MSG

- Class: Run Time

- Default: 256K Bytes

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter can be used to determine the threshold for the 2-level Allreduce algorithm. We now use the shared-memory-based algorithm for messages smaller than the MV2_SHMEM_ALLREDUCE_MSG threshold (11.61), the 2-level algorithm for medium sized messages up to the threshold defined by this parameter. We use the default point-to-point algorithms messages larger than this threshold.

11.2   MV2_CKPT_AGGREGATION_BUFPOOL_SIZE

- Class: Run Time

- Default: 8M

- Applicable interface(s): OFA-IB-CH3

This parameter determines the size of the buffer pool reserved for use in checkpoint aggregation. Note that this variable can be set with suffixes such as 'K'/'k', 'M'/'m' or 'G'/'g' to denote Kilobyte, Megabyte or Gigabyte respectively.

11.3   MV2_CKPT_AGGREGATION_CHUNK_SIZE

- Class: Run Time

- Default: 1M

- Applicable interface(s): OFA-IB-CH3

The checkpoint data that has been coalesced into the buffer pool, is written to the back-end file system, with the value of this parameter as the chunk size. Note that this variable can be set with suffixes such as 'K'/'k', 'M'/'m' or 'G'/'g' to denote Kilobyte, Megabyte or Gigabyte respectively.

11.4   MV2_CKPT_FILE

- Class: Run Time

- Default: /tmp/ckpt

- Applicable interface(s): OFA-IB-CH3

This parameter specifies the path and the base file name for checkpoint files of MPI processes. The checkpoint files will be named as $MV2_CKPT_FILE.<number of checkpoint>.<process rank>, for example, /tmp/ckpt.1.0 is the checkpoint file for process 0's first checkpoint. To checkpoint on network-based file systems, user just need to specify the path to it, such as /mnt/pvfs2/my_ckpt_file.

11.5   MV2_CKPT_INTERVAL

- Class: Run Time

- Default: 0

- Unit: minutes

- Applicable interface(s): OFA-IB-CH3

This parameter can be used to enable automatic checkpointing. To let MPI job console automatically take checkpoints, this value needs to be set to the desired checkpointing interval. A zero will disable automatic checkpointing. Using automatic checkpointing, the checkpoint file for the MPI job console will be named as $MV2_CKPT_FILE.<number of checkpoint>.auto. Users need to use this file for restart.

11.6   MV2_CKPT_MAX_SAVE_CKPTS

- Class: Run Time

- Default: 0

- Applicable interface(s): OFA-IB-CH3

This parameter is used to limit the number of checkpoints saved on file system to save the file system space. When set to a positive value N, only the last N checkpoints will be saved.

11.7   MV2_CKPT_NO_SYNC

- Class: Run Time

- Applicable interface(s): OFA-IB-CH3

When this parameter is set to any value, the checkpoints will not be required to sync to disk. It can reduce the checkpointing delay in many cases. But if users are using local file system, or any parallel file system with local cache, to store the checkpoints, it is recommended not to set this parameter because otherwise the checkpoint files will be cached in local memory and will likely be lost upon failure.

11.8 MV2_CKPT_USE_AGGREGATION

- Class: Run Time

- Default: 1 (if configured with Checkpoint Aggregation support)

- Applicable interface(s): OFA-IB-CH3

This parameter enables/disables Checkpoint aggregation scheme at run time. It is set to '1'(enabled) by default, when the user enables Checkpoint/Restart functionality at configure time, or when the user explicitly configures MVAPICH2 with aggregation support. Please note that, to use aggregation support, each node needs to be properly configured with FUSE library (cf section **??**).

11.9 MV2_DEBUG_FT_VERBOSE

- Class: Run Time

- Type: Null or positive integer

- Default: 0 (disabled)

This parameter enables/disables the debug output for Fault Tolerance features (Checkpoint/Restart and Migration).

Note: All debug output is disabled when MVAPICH2 is configured with the `--enable-fast=ndebug` option.

11.10 MV2_CM_RECV_BUFFERS

- Class: Run Time

- Default: 1024

- Applicable interface(s): OFA-IB-CH3

This defines the number of buffers used by connection manager to establish new connections. These buffers are quite small and are shared for all connections, so this value may be increased to 8192 for large clusters to avoid retries in case of packet drops.

11.11 MV2_CM_SPIN_COUNT

- Class: Run Time

- Default: 5000

- Applicable interface(s): OFA-IB-CH3

This is the number of the connection manager polls for new control messages from UD channel for each interrupt. This may be increased to reduce the interrupt overhead when many incoming control messages from UD channel at the same time.

11.12   MV2_CM_TIMEOUT

- Class: Run Time

- Default: 500

- Unit: milliseconds

- Applicable interface(s): OFA-IB-CH3

This is the timeout value associated with connection management messages via UD channel. Decreasing this value may lead to faster retries but at the cost of generating duplicate messages.

11.13   MV2_CPU_MAPPING

- Class: Run Time

- Default: NA

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3 (Linux)

This allows users to specify process to CPU (core) mapping. The detailed usage of this parameter is described in Section 6.3.2. This parameter will not take effect if either *MV2_ENABLE_AFFINITY* or *MV2_USE_SHARED_MEM* run-time parameters are set to 0, or if the library was configured with the "–disable-hwloc" option. MV2_CPU_MAPPING is currently not supported on Solaris.

11.14   MV2_CPU_BINDING_POLICY

- Class: Run Time

- Default: Bunch

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3 (Linux)

This allows users to specify process to CPU (core) mapping with the CPU binding policy. The detailed usage of this parameter is described in Section 6.3.1. This parameter will not take effect: if *MV2_ENABLE_AFFINITY* or *MV2_USE_SHARED_MEM* run-time parameters are set to 0; or *MV2_ENABLE_AFFINITY* is set to 1 and *MV2_CPU_MAPPING* is set, or if the library was configured with the "–disable-hwloc" option. The value of MV2_CPU_BINDING_POLICY can be "bunch" or "scatter". When this parameter takes effect and its value isn't set, "bunch" will be used as the default policy.

11.15   MV2_CPU_BINDING_LEVEL

- Class: Run Time

- Default: Core

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3 (Linux)

This allows users to specify process to CPU (core) mapping at different binding level. The detailed usage of this parameter is described in Section 6.3.1. This parameter will not take effect: if *MV2_ENABLE_AFFINITY* or *MV2_USE_SHARED_MEM* run-time parameters are set to 0; or MV2_ENABLE_AFFINITY is set to 1 and MV2_CPU_MAPPING is set, or if the library was configured with the "–disable-hwloc" option. The value of MV2_CPU_BINDING_LEVEL can be "core", "socket", or "numanode". When this parameter takes effect and its value isn't set, "core" will be used as the default binding level.

11.16   MV2_SHOW_CPU_BINDING

- Class: Run Time

- Default: 0

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3 (Linux)

This shows the current process to CPU (core) bindings of all processes on a node which contains the mpi rank 0 process.

11.17   MV2_DAPL_PROVIDER

- Class: Run time

- Default: ofa-v2-ib0 (Linux DAPL v2.0), OpenIB-cma (Linux DAPL v1.2), ibd0 (Solaris)

- Applicable interface(s): uDAPL-CH3

This is to specify the underlying uDAPL-CH3 library that the user would like to use if MVAPICH2 is built with uDAPL-CH3.

11.18   MV2_DEFAULT_MAX_SEND_WQE

- Class: Run time

- Default: 64

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This specifies the maximum number of send WQEs on each QP. Please note that for OFA-IB-CH3 and OFA-iWARP-CH3, the default value of this parameter will be 16 if the number of processes is larger than 256 for better memory scalability.

11.19 MV2_DEFAULT_MAX_RECV_WQE

- Class: Run time
- Default: 128
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This specifies the maximum number of receive WQEs on each QP (maximum number of receives that can be posted on a single QP).

11.20 MV2_DEFAULT_MTU

- Class: Run time
- Default: OFA-IB-CH3: IBV_MTU_1024 for IB SDR cards and IBV_MTU_2048 for IB DDR and QDR cards. uDAPL-CH3: Network dependent.
- Applicable interface(s): OFA-IB-CH3, uDAPL-CH3

The internal MTU size. For OFA-IB-CH3, this parameter should be a string instead of an integer. Valid values are: `IBV_MTU_256`, `IBV_MTU_512`, `IBV_MTU_1024`, `IBV_MTU_2048`, `IBV_MTU_4096`.

11.21 MV2_DEFAULT_PKEY

- Class: Run Time
- Applicable Interface(s): OFA-IB-CH3

Select the partition to be used for the job.

11.22 MV2_ENABLE_AFFINITY

- Class: Run time
- Default: 1
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3 (Linux)

Enable CPU affinity by setting MV2_ENABLE_AFFINITY to 1 or disable it by setting MV2_ENABLE_AFFINITY to 0. MV2_ENABLE_AFFINITY is currently not supported on Solaris. CPU affinity is also not supported if MV2_USE_SHARED_MEM is set to 0.

11.23   MV2_GET_FALLBACK_THRESHOLD

- Class: Run time

- This threshold value needs to be set in bytes.

- This option is effective if we define ONE_SIDED flag.

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines the threshold beyond which the MPI_Get implementation is based on direct one sided RDMA operations.

11.24   MV2_IBA_EAGER_THRESHOLD

- Class: Run time

- Default: Host Channel Adapter (HCA) dependent (12 KB for ConnectX HCA's)

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This specifies the switch point between eager and rendezvous protocol in MVAPICH2. For better performance, the value of MV2_IBA_EAGER_THRESHOLD should be set the same as MV2_VBUF_TOTAL_SIZE.

11.25   MV2_IBA_HCA

- Class: Run time

- Default: Unset

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This specifies the HCA's to be used for performing network operations.

11.26   MV2_INITIAL_PREPOST_DEPTH

- Class: Run time

- Default: 10

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

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 RDMA_PREPOST_DEPTH.

11.27 MV2_IWARP_MULTIPLE_CQ_THRESHOLD

- Class: Run time

- Default: 32

- Applicable interface(s): OFA-iWARP-CH3

This defines the process size beyond which we use multiple completion queues for iWARP interface.

11.28 MV2_KNOMIAL_INTRA_NODE_FACTOR

- Class: Run time

- Default: 4

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines the degree of the knomial operation during the intra-node knomial broadcast phase.

11.29 MV2_KNOMIAL_INTER_NODE_FACTOR

- Class: Run time

- Default: 4

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines the degree of the knomial operation during the inter-node knomial broadcast phase.

11.30 MV2_KNOMIAL_2LEVEL_BCAST_MESSAGE_SIZE_THRESHOLD

- Class: Run time

- Default: 2KB

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

When an intra-communicator MPI_Bcast operation is invoked with a communicator whose size is larger than the mv2_knomial_2level_bcast_system_size_threshold value, we use the 2-level knomial algorithm for message sizes smaller than the value set by this parameter.

11.31   MV2_KNOMIAL_2LEVEL_BCAST_SYSTEM_SIZE_THRESHOLD

- Class: Run time

- Default: 32

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

For communicator sizes larger than this value, the knomial bcast algorithm will be used for message sizes less than the value assigned to the mv2_knomial_2level_bcast_message_size_threshold parameter. For smaller systems, we use the default binomial tree based algorithm.

11.32   MV2_LIMIC_GET_THRESHOLD

- Class: Run time

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This defines the message size threshold (in bytes) beyond which intra-node MPI_Get operations are optimized using the LiMIC2 Kernel module.

11.33   MV2_LIMIC_PUT_THRESHOLD

- Class: Run time

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This defines the message size threshold (in bytes) beyond which intra-node MPI_Put operations are optimized using the LiMIC2 Kernel module.

11.34   MV2_MAX_INLINE_SIZE

- Class: Run time

- Default: Network card dependent (128 for most networks including InfiniBand)

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This defines the maximum inline size for data transfer. Please note that the default value of this parameter will be 0 when the number of processes is larger than 256 to improve memory usage scalability.

11.35   MV2_MAX_NUM_WIN

- Class: Run time

- Default: 64

- Applicable interface(s): OFA-IB-CH3

Maximum number of RMA windows that can be created and active concurrently. Typically this value is sufficient for most applications. Increase this value to the number of windows your application uses

11.36   MV2_NDREG_ENTRIES

- Class: Run time

- Default: 1000

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines the total number of buffers that can be stored in the registration cache. It has no effect if MV2_USE_LAZY_MEM_UNREGISTER is not set. A larger value will lead to less frequent lazy de-registration.

11.37   MV2_NUM_HCAS

- Class: Run time

- Default: 1

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter indicates number of InfiniBand adapters to be used for communication on an end node.

11.38   MV2_NUM_PORTS

- Class: Run time

- Default: 1

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter indicates number of ports per InfiniBand adapter to be used for communication per adapter on an end node.

11.39   MV2_NUM_SA_QUERY_RETRIES

- Class: Run time

- Default: 20

- Applicable Interface(s): OFA-IB-CH3, OFA-iWARP-CH3

Number of times the MPI library will attempt to query the subnet to obtain the path record information before giving up.

11.40   MV2_NUM_QP_PER_PORT

- Class: Run time

- Default: 1

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter indicates number of queue pairs per port to be used for communication on an end node. This is useful in the presence of multiple send/recv engines available per port for data transfer.

11.41   MV2_RAIL_SHARING_POLICY

- Class: Run time

- Default: Rail Binding in round-robin

- Value Domain: USE_FIRST, ROUND_ROBIN, FIXED_MAPPING

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This specifies the policy that will be used to assign HCAs to each of the processes. In the previous versions of MVAPICH2 it was known as MV2_SM_SCHEDULING.

11.42   MV2_RAIL_SHARING_LARGE_MSG_THRESHOLD

- Class: Run time

- Default: 16K

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This specifies the threshold for the message size beyond which striping will take place. In the previous versions of MVAPICH2 it was known as MV2_STRIPING_THRESHOLD

11.43   MV2_PROCESS_TO_RAIL_MAPPING

- Class: Run time

- Default: NONE

- Value Domain: BUNCH, SCATTER, <CUSTOM LIST>

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

When MV2_RAIL_SHARING_POLICY is set to the value "FIXED_MAPPING" this variable decides the manner in which the HCAs will be mapped to the rails. The <CUSTOM LIST> is colon(:) separated list with the HCA ranks specified. e.g. 0:1:1:0. This list must map equally to the number of local processes on the nodes failing which, the default policy will be used. Similarly the number of processes on each node must be the same.

11.44   MV2_RDMA_FAST_PATH_BUF_SIZE

- Class: Run time

- Default: Architecture dependent

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

The size of the buffer used in rdma fast path communication. This value will be ineffective if MV2_USE_RDMA_FAST_PATH is not set

11.45   MV2_NUM_RDMA_BUFFER

- Class: Run time

- Default: Architecture dependent (32 for EM64T)

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

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 will be ineffective if MV2_USE_RDMA_FAST_PATH is not set.

11.46   MV2_ON_DEMAND_THRESHOLD

- Class: Run Time

- Default: 64 (OFA-IB-CH3, uDAPL-CH3), 16 (OFA-iWARP-CH3)

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines threshold for enabling on-demand connection management scheme. When the size of the job is larger than the threshold value, on-demand connection management will be used.

11.47   MV2_PREPOST_DEPTH

- Class: Run time

- Default: 64

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

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

11.48   MV2_PROCESS_TO_RAIL_MAPPING

- Class: Run Time

- Default: SCATTER (Options: BUNCH, User Defined)

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3 (Linux)

This allows users to specify HCAs to be used by processes. The detailed usage of this parameter is described in Section 6.9.

11.49   MV2_PSM_DEBUG

- Class: Run time (Debug)

- Default: 0

- Applicable interface: PSM

This parameter enables the dumping of run-time debug counters from the MVAPICH2-PSM progress engine. Counters are dumped every PSM_DUMP_FREQUENCY seconds.

11.50   MV2_PSM_DUMP_FREQUENCY

- Class: Run time (Debug)

- Default: 10 seconds

- Applicable interface: PSM

This parameters sets the frequency for dumping MVAPICH2-PSM debug counters. Value takes effect only in PSM_DEBUG is enabled.

11.51 MV2 PUT FALLBACK THRESHOLD

- Class: Run time

- This threshold value needs to be set in bytes.

- This option is effective if we define ONE_SIDED flag.

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines the threshold beyond which the MPI_Put implementation is based on direct one sided RDMA operations.

11.52 MV2 RAIL SHARING LARGE MSG THRESHOLD

- Class: Run Time

- Default: 16 KB

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter specifies the message size above which we begin the stripe the message across multiple rails (if present).

11.53 MV2 RAIL SHARING POLICY

- Class: Run Time

- Default: PROCESS_BINDING (Options: USE_FIRST, ROUND_ROBIN, FIXED_MAPPING)

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter specifies the rail sharing policy for all message sizes if multiple rails are present.

11.54 MV2 RDMA CM ARP TIMEOUT

- Class: Run Time

- Default: 2000 ms

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter specifies the ARP timeout to be used by RDMA CM module.

11.55   MV2_RDMA_CM_MAX_PORT

- Class: Run Time

- Default: Unset

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter specifies the upper limit of the port range to be used by the RDMA CM module when choosing the port on which it listens for connections.

11.56   MV2_RDMA_CM_MIN_PORT

- Class: Run Time

- Default: Unset

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter specifies the lower limit of the port range to be used by the RDMA CM module when choosing the port on which it listens for connections.

11.57   MV2_REDUCE_2LEVEL_MSG

- Class: Run Time

- Default: 32K Bytes.

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter can be used to determine the threshold for the 2-level reduce algorithm. We now use the shared-memory-based algorithm for messages smaller than the MV2_SHMEM_REDUCE_MSG (11.67), the 2-level algorithm for medium sized messages upto the threshold defined by this parameter. We use the default point-to-point algorithms messages larger than this threshold.

11.58   MV2_RNDV_PROTOCOL

- Class: Run time

- Default: RPUT

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The value of this variable can be set to choose different Rendezvous protocols. RPUT (default RDMA-Write) RGET (RDMA Read based), R3 (send/recv based).

11.59 MV2_R3_THRESHOLD

- Class: Run time
- Default: MV2_IBA_EAGER_THRESHOLD
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The value of this variable controls what message sizes go over the R3 rendezvous protocol. Messages above this message size use MV2_RNDV_PROTOCOL.

11.60 MV2_R3_NOCACHE_THRESHOLD

- Class: Run time
- Default: 32768
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The value of this variable controls what message sizes go over the R3 rendezvous protocol when the registration cache is disabled (MV2_USE_LAZY_MEM_UNREGISTER=0). Messages above this message size use MV2_RNDV_PROTOCOL.

11.61 MV2_SHMEM_ALLREDUCE_MSG

- Class: Run Time
- Default: $1 \ll 15$
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The SHMEM AllReduce is used for messages less than this threshold.

11.62 MV2_SHMEM_BCAST_LEADERS

- Class: Run time
- Default: 4096

The number of leader processes that will take part in the SHMEM broadcast operation. Must be greater than the number of nodes in the job.

11.63   MV2_SHMEM_BCAST_MSG

- Class: Run Time

- Default: $1 \ll 20$

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The SHMEM bcast is used for messages less than this threshold.

11.64   MV2_SHMEM_COLL_MAX_MSG_SIZE

- Class: Run Time

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter can be used to select the max buffer size of message for shared memory collectives.

11.65   MV2_SHMEM_COLL_NUM_COMM

- Class: Run Time

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter can be used to select the number of communicators using shared memory collectives.

11.66   MV2_SHMEM_DIR

- Class: Run Time

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

- Default: /dev/shm for Linux and /tmp for Solaris

This parameter can be used to specify the path to the shared memory files for intra-node communication.

11.67   MV2_SHMEM_REDUCE_MSG

- Class: Run Time

- Default: $1 \ll 13$

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The SHMEM reduce is used for messages less than this threshold.

11.68   MV2_SM_SCHEDULING

- Class: Run Time

- Default: USE_FIRST (Options: ROUND_ROBIN)

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

11.69   MV2_SMP_USE_LIMIC2

- Class: Run Time

- Default: On if configured with –with-limic2

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter enables/disables LiMIC2 at run time. It does not take effect if MVAPICH2 is not configured with –with-limic2.

11.70   MV2_SRQ_LIMIT

- Class: Run Time

- Default: 30

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This is the low water-mark 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.

11.71   MV2_SRQ_MAX_SIZE

- Class: Run Time

- Default: 4096

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This is the maximum number of work requests allowed on the Shared Receive Queue. Upon receiving a SRQ limit event, the current value of MV2_SRQ_SIZE will be doubled or moved to the maximum of MV2_SRQ_MAX_SIZE, whichever is smaller

11.72   MV2_SRQ_SIZE

- Class: Run Time

- Default: 256

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This is the initial number of work requests posted to the Shared Receive Queue.

11.73   MV2_STRIPING_THRESHOLD

- Class: Run Time

- Default: 8192

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter specifies the message size above which we begin the stripe the message across multiple rails (if present).

11.74   MV2_SUPPORT_DPM

- Class: Run time

- Default: 0 (disabled)

- Applicable interface: OFA-IB-CH3

This option enables the dynamic process management interface and on-demand connection management.

11.75   MV2_USE_APM

- Class: Run Time

- Applicable interface(s): OFA-IB-CH3

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.

11.76   MV2_USE_APM_TEST

- Class: Run Time

- Applicable interface(s): OFA-IB-CH3

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.

11.77   MV2_USE_BLOCKING

- Class: Run time

- Default: 0

- Applicable interface(s): OFA-IB-CH3

Setting this parameter enables MVAPICH2 to use blocking mode progress. MPI applications do not take up any CPU when they are waiting for incoming messages.

11.78   MV2_USE_COALESCE

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

Setting this parameter enables message coalescing to increase small message throughput

11.79   MV2_USE_DIRECT_GATHER

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Use the "Direct" algorithm for the MPI_Gather operation. If this parameter is set to 0 at run-time, the "Direct" algorithm will not be invoked.

11.80 MV2_USE_DIRECT_SCATTER

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Use the "Direct" algorithm for the MPI_Scatter operation. If this parameter is set to 0 at run-time, the "Direct" algorithm will not be invoked.

11.81 MV2_USE_HSAM

- Class: Run Time

- Applicable interface(s): OFA-IB-CH3

This parameter is used for utilizing hot-spot avoidance with InfiniBand clusters. To leverage this functionality, the subnet should be configured with lmc greater than zero. Please refer to section 6.14 for detailed information.

11.82 MV2_USE_IWARP_MODE

- Class: Run Time

- Default: unset

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter enables the library to run in iWARP mode. The library has to be built using the flag -DRDMA_CM for using this feature.

11.83 MV2_USE_KNOMIAL_2LEVEL_BCAST

- Class: Run time

- Default: 1

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3 (Linux)

Enable Knomial Broadcast by setting MV2_USE_KNOMIAL_2LEVEL_BCAST to 1 or disable it by setting
MV2_USE_KNOMIAL_2LEVEL_BCAST to 0. The other knomial related variables are :

- MV2_KNOMIAL_INTRA_NODE_FACTOR

- MV2_KNOMIAL_INTER_NODE_FACTOR

- MV2_KNOMIAL_2LEVEL_BCAST_THRESHOLD

11.84   MV2_USE_LAZY_MEM_UNREGISTER

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Setting this parameter enables MVAPICH2 to use memory registration cache.

11.85   MV2_USE_LIMIC_ONE_SIDED

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

Setting this parameter allows MVAPICH2 to use LiMIC2 kernel module for one sided communication within a node. Currently this optimization is available only with the Post-Wait/Start-Complete semantics.

11.86   MV2_USE_RoCE

- Class: Run Time

- Default: Un Set

- Applicable interface(s): OFA-IB-CH3

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

11.87   MV2_USE_RDMA_CM

- Class: Run Time

- Default: Network Dependent (set for OFA-iWARP-CH3 and unset for OFA-IB-CH3)

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter enables the use of RDMA CM for establishing the connections. The library has to be built using the flag -DRDMA_CM for using this feature.

11.88   MV2_USE_RDMA_FAST_PATH

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Setting this parameter enables MVAPICH2 to use adaptive RDMA fast path features for OFA-IB-CH3 interface and static RDMA fast path features for uDAPL-CH3 interface.

11.89   MV2_USE_RDMA_ONE_SIDED

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Setting this parameter allows MVAPICH2 to use optimized one sided implementation based RDMA operations.

11.90   MV2_USE_RING_STARTUP

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3

Setting this parameter enables MVAPICH2 to use ring based start up.

11.91   MV2_USE_SHARED_MEM

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Use shared memory for intra-node communication.

11.92   MV2_USE_SHM_ONE_SIDED

- Class: Run time

- Default: unset

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

Setting this parameter allocates the buffers requested using MPI_Alloc_mem in shared memory. One-sided communication over windows created in this memory is optimized. The use of optimization hint for MPI_Alloc_mem described in Section 6.5 is recommended over setting this parameter in order to take advantage of the optimized intra-node one-sided communication.

11.93   MV2_USE_SHMEM_ALLREDUCE

- Class: Run Time

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3, VAPI

This parameter can be used to turn off shared memory based MPI_Allreduce for OFA-IB-CH3 over IBA by setting this to 0.

11.94   MV2_USE_SHMEM_BARRIER

- Class: Run Time

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3, VAPI

This parameter can be used to turn off shared memory based MPI_Barrier for OFA-IB-CH3 over IBA by setting this to 0.

11.95   MV2_USE_SHMEM_BCAST

- Class: Run Time

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter can be used to turn off shared memory based MPI_Bcast for OFA-IB-CH3 over IBA by setting this to 0.

11.96   MV2_USE_SHMEM_COLL

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Use shared memory for collective communication. Set this to 0 for disabling shared memory collectives.

11.97   MV2_USE_SHMEM_REDUCE

- Class: Run Time

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3, VAPI

This parameter can be used to turn off shared memory based MPI_Reduce for OFA-IB-CH3 over IBA by setting this to 0.

11.98   MV2_USE_SRQ

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

Setting this parameter enables MVAPICH2 to use shared receive queue.

11.99   MV2_GATHER_SWITCH_PT

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

We use different algorithms depending on the system size. For small system sizes (up to 386 cores), we use the "2-level" algorithm following by the "Direct" algorithm. For medium system sizes (up to 1k), we use "Binomial" algorithm following by the "Direct" algorithm. Users can set the switching point between algorithms using the run-time parameter MV2_GATHER_SWITCH_PT.

11.100   MV2_SCATTER_SMALL_MSG

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

When the system size is lower than 512 cores, we use the "Binomial" algorithm for small message sizes. MV2_SCATTER_SMALL_MSG allows the users to set the threshold for small messages.

11.101   MV2_SCATTER_MEDIUM_MSG

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

When the system size is lower than 512 cores, we use the "2-level" algorithm for medium message sizes. MV2_SCATTER_MEDIUM_MSG allows the users to set the threshold for medium messages.

11.102   MV2_USE_TWO_LEVEL_GATHER

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Use the two-level multi-core-aware algorithm for the MPI_Gather operation. If this parameter is set to 0 at run-time, the two-level algorithm will not be invoked.

11.103   MV2_USE_TWO_LEVEL_SCATTER

- Class: Run time

- Default: set

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Use the two-level multi-core-aware algorithm for the MPI_Scatter operation. If this parameter is set to 0 at run-time, the two-level algorithm will not be invoked.

11.104   MV2_USE_XRC

- Class: Run time

- Default: 0

- Applicable Interface(s): OFA-IB-CH3

Use the XRC InfiniBand transport available since Mellanox ConnectX adapters. This features requires OFED version later than 1.3. It also automatically enables SRQ and ON-DEMAND connection management. Note that the MVAPICH2 library needs to have been configured with –enable-xrc=yes to use this feature.

11.105   MV2_VBUF_POOL_SIZE

- Class: Run time

- Default: 512

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

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

11.106   MV2_VBUF_SECONDARY_POOL_SIZE

- Class: Run time

- Default: 256

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

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.

11.107   MV2_VBUF_TOTAL_SIZE

- Class: Run time

- Default: Host Channel Adapter (HCA) dependent (12 KB for ConnectX HCA's)

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The size of each `vbuf`, the basic communication buffer of MVAPICH2. For better performance, the value of MV2_IBA_EAGER_THRESHOLD should be set the same as MV2_VBUF_TOTAL_SIZE.

11.108 MV2_SMP_EAGERSIZE

- Class: Run time

- Default: Architecture dependent

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter defines the switch point from Eager protocol to Rendezvous protocol for intra-node communication. Note that this variable can be set with suffixes such as 'K'/'k', 'M'/'m' or 'G'/'g' to denote Kilobyte, Megabyte or Gigabyte respectively.

11.109 MV2_SMPI_LENGTH_QUEUE

- Class: Run time

- Default: Architecture dependent

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter defines the size of shared buffer between every two processes on the same node for transferring messages smaller than or equal to MV2_SMP_EAGERSIZE. Note that this variable can be set with suffixes such as 'K'/'k', 'M'/'m' or 'G'/'g' to denote Kilobyte, Megabyte or Gigabyte respectively.

11.110 MV2_SMP_NUM_SEND_BUFFER

- Class: Run time

- Default: Architecture dependent

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter defines the number of internal send buffers for sending intra-node messages larger than MV2_SMP_EAGERSIZE.

11.111 MV2_SMP_SEND_BUF_SIZE

- Class: Run time

- Default: Architecture dependent

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter defines the packet size when sending intra-node messages larger than MV2_SMP_EAGERSIZE.

11.112   MV2_USE_HUGEPAGES

- Class: Run time
- Default: 1
- Applicable interface(s): OFA-IB-CH3

Set this to 0, to not use any HugePages.

11.113   MV2_HYBRID_ENABLE_THRESHOLD

- Class: Run time
- Default: 1024
- Applicable interface(s): OFA-IB-CH3

This defines the threshold for enabling Hybrid communication using UD and RC/XRC. When the size of the job is greater than or equal to the threshold value, Hybrid mode will be enabled. Otherwise, it uses default RC/XRC connections for communication.

11.114   MV2_HYBRID_MAX_RC_CONN

- Class: Run time
- Default: 64
- Applicable interface(s): OFA-IB-CH3

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

11.115   MV2_UD_PROGRESS_TIMEOUT

- Class: Run time
- Default: System size dependent.
- Applicable interface(s): OFA-IB-CH3

Time (usec) until ACK status is checked (and ACKs are sent if needed). To avoid unnecessary retries, set this value less than MV2_UD_RETRY_TIMEOUT. It is recommended to set this to 1/10 of MV2_UD_RETRY_TIMEOUT.

11.116 MV2_UD_RETRY_TIMEOUT

- Class: Run time
- Default: System size dependent.
- Applicable interface(s): OFA-IB-CH3

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

11.117 MV2_UD_RETRY_COUNT

- Class: Run time
- Default: System size dependent.
- Applicable interface(s): OFA-IB-CH3

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

11.118 MV2_USE_UD_HYBRID

- Class: Run time
- Default: 1
- Applicable interface(s): OFA-IB-CH3

Set this to Zero, to disable UD transport in hybrid configuration mode.

11.119 MV2_USE_ONLY_UD

- Class: Run time
- Default: 0
- Applicable interface(s): OFA-IB-CH3

Set this to 1, to enable only UD transport in hybrid configuration mode. It will not use any RC/XRC connections in this mode.

11.120   MV2_USE_UD_ZCOPY

- Class: Run time
- Default: 1
- Applicable interface(s): OFA-IB-CH3

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

11.121   MV2_USE_CUDA

- Class: Run time
- Default: 0
- Applicable interface(s): OFA-IB-CH3

set this to One. to enable support for communication with GPU device buffers.

11.122   MV2_CUDA_BLOCK_SIZE

- Class: Run time
- Default: 262144
- Applicable interface(s): OFA-IB-CH3

The chunk size used in large message transfer from device memory to host memory. The other suggested values for this parameter are 131072 and 524288.

11.123   MV2_CUDA_EVENT_SYNC

- Class: Run time
- Default: 1
- Applicable interface(s): OFA-IB-CH3

MVAPICH2 uses cuda events to synchronization in pipelined communication from device. Disabling this parameter enables use of cuda streams for synchronization instead of cuda events.

11.124   MV2_CUDA_IPC

- Class: Run time

- Default: 1

- Applicable interface(s): OFA-IB-CH3

This enables intra-node GPU-GPU communication using IPC feature available in CUDA 4.1

11.125   MV2_CUDA_IPC_THRESHOLD

- Class: Run time

- Default: 0 bytes

- Applicable interface(s): OFA-IB-CH3

This specifies the threshold for the message size beyond which uses CUDA IPC feature for GPU-GPU communication within a node. Default, it uses IPC for all size of the message sizes.

12   MVAPICH2 Parameters (OFA-IB-Nemesis Interface)

12.1   MV2_DEFAULT_MAX_SEND_WQE

- Class: Run time

- Default: 64

This specifies the maximum number of send WQEs on each QP. Please note that for Gen2 and Gen2-iWARP, the default value of this parameter will be 16 if the number of processes is larger than 256 for better memory scalability.

12.2   MV2_DEFAULT_MAX_RECV_WQE

- Class: Run time

- Default: 128

This specifies the maximum number of receive WQEs on each QP (maximum number of receives that can be posted on a single QP).

12.3   MV2_DEFAULT_MTU

- Class: Run time

- Default: IBV_MTU_1024 for IB SDR cards and IBV_MTU_2048 for IB DDR and QDR cards. uDAPL: Network dependent.

The internal MTU size. For Gen2, this parameter should be a string instead of an integer. Valid values are: `IBV_MTU_256`, `IBV_MTU_512`, `IBV_MTU_1024`, `IBV_MTU_2048`, `IBV_MTU_4096`.

12.4   MV2_DEFAULT_PKEY

- Class: Run Time

Select the partition to be used for the job.

12.5   MV2_IBA_EAGER_THRESHOLD

- Class: Run time

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

This specifies the switch point between eager and rendezvous protocol in MVAPICH2. For better performance, the value of MV2_IBA_EAGER_THRESHOLD should be set the same as MV2_VBUF_TOTAL_SIZE.

12.6 MV2_IBA_HCA

- Class: Run time

- Default: Unset

This specifies the HCA to be used for performing network operations.

12.7 MV2_INITIAL_PREPOST_DEPTH

- Class: Run time

- Default: 10

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 RDMA_PREPOST_DEPTH.

12.8 MV2_MAX_INLINE_SIZE

- Class: Run time

- Default: Network card dependent (128 for most networks including InfiniBand)

This defines the maximum inline size for data transfer. Please note that the default value of this parameter will be 0 when the number of processes is larger than 256 to improve memory usage scalability.

12.9 MV2_NDREG_ENTRIES

- Class: Run time

- Default: 1000

This defines the total number of buffers that can be stored in the registration cache. It has no effect if MV2_USE_LAZY_MEM_UNREGISTER is not set. A larger value will lead to less frequent lazy de-registration.

12.10   MV2_NUM_RDMA_BUFFER

- Class: Run time

- Default: Architecture dependent (32 for EM64T)

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 will be ineffective if MV2_USE_RDMA_FAST_PATH is not set.

12.11   MV2_NUM_SA_QUERY_RETRIES

- Class: Run time

- Default: 20

- Applicable Interface(s): OFA-IB-CH3, OFA-iWARP-CH3

Number of times the MPI library will attempt to query the subnet to obtain the path record information before giving up.

12.12   MV2_PREPOST_DEPTH

- Class: Run time

- Default: 64

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

12.13   MV2_RNDV_PROTOCOL

- Class: Run time

- Default: RPUT

The value of this variable can be set to choose different Rendezvous protocols. RPUT (default RDMA-Write) RGET (RDMA Read based), R3 (send/recv based).

12.14   MV2_R3_THRESHOLD

- Class: Run time

- Default: 4096

The value of this variable controls what message sizes go over the R3 rendezvous protocol. Messages above this message size use MV2_RNDV_PROTOCOL.

12.15 MV2_R3_NOCACHE_THRESHOLD

- Class: Run time

- Default: 32768

The value of this variable controls what message sizes go over the R3 rendezvous protocol when the registration cache is disabled (MV2_USE_LAZY_MEM_UNREGISTER=0). Messages above this message size use MV2_RNDV_PROTOCOL.

12.16 MV2_SRQ_LIMIT

- Class: Run Time

- Default: 30

This is the low water-mark 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.

12.17 MV2_SRQ_SIZE

- Class: Run Time

- Default: 512

This is the maximum number of work requests allowed on the Shared Receive Queue.

12.18 MV2_STRIPING_THRESHOLD

- Class: Run Time

- Default: 8192

This parameter specifies the message size above which we begin the stripe the message across multiple rails (if present).

12.19 MV2_USE_BLOCKING

- Class: Run time

- Default: 0

Setting this parameter enables mvapich2 to use blocking mode progress. MPI applications do not take up any CPU when they are waiting for incoming messages.

12.20   MV2_USE_LAZY_MEM_UNREGISTER

- Class: Run time
- Default: set

Setting this parameter enables mvapich2 to use memory registration cache.

12.21   MV2_USE_RDMA_FAST_PATH

- Class: Run time
- Default: set

Setting this parameter enables mvapich2 to use adaptive rdma fast path features for Gen2 interface and static rdma fast path features for uDAPL interface.

12.22   MV2_USE_SRQ

- Class: Run time
- Default: set

Setting this parameter enables mvapich2 to use shared receive queue.

12.23   MV2_VBUF_POOL_SIZE

- Class: Run time
- Default: 512

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

12.24   MV2_VBUF_SECONDARY_POOL_SIZE

- Class: Run time
- Default: 128

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.

12.25   MV2_VBUF_TOTAL_SIZE

- Class: Run time

- Default: Architecture dependent (6 KB for EM64T)

The size of each `vbuf`, the basic communication buffer of MVAPICH2. For better performance, the value of MV2_IBA_EAGER_THRESHOLD should be set the same as MV2_VBUF_TOTAL_SIZE.

12.26   MV2_RUN_THROUGH_STABILIZATION

- Class: Run Time

- Default: 0

This enables run through stabilization support to handle the process failures. This is valid only with Hydra process manager with –disable-auto-cleanup flag.

13 MPIRUN_RSH Parameters

13.1 MV2_COMM_WORLD_LOCAL_RANK

- Class: Run time
- Applicable Interface(s): All

The local rank of a process on a node within its job. The local rank ranges from 0,1 ... N-1 on a node with N processes running on it.

13.2 MV2_COMM_WORLD_LOCAL_SIZE

- Class: Run time
- Applicable Interface(s): All

The number of ranks from this job that are running on this node.

13.3 MV2_COMM_WORLD_RANK

- Class: Run time
- Applicable Interface(s): All

The MPI rank of this process in current MPI job

13.4 MV2_COMM_WORLD_SIZE

- Class: Run time
- Applicable Interface(s): All

The number of processes in this MPI job's MPI_Comm_World.

13.5 MV2_FASTSSH_THRESHOLD

- Class: Run time
- Default: 256
- Applicable Interface(s): All

Number of nodes beyond which to use hierarchical ssh during start up. This parameter is only relevant for mpirun_rsh based start up.

13.6   MV2_NPROCS_THRESHOLD

- Class: Run time

- Default: 8192

- Applicable Interface(s): All

Number of nodes beyond which to use file-based communication scheme in the hierarchical ssh during start up. This parameter is only relevant for mpirun_rsh based start up.

13.7   MV2_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 run time environment (for e.g. through export in the bash shell).

13.8   MV2_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 run time environment (for e.g. through export in the bash shell).

13.9   MPIEXEC_TIMEOUT

- Class: Run time

- Default: Unset

- Unit: Seconds

Set this to limit, in seconds, of the execution time of the mpi application. This overwrites the MV2_MPIRUN_TIMEOUT parameter.

13.10   MV2_DEBUG_FORK_VERBOSE

- Class: Run time

- Type: Null or positive integer

- Default: 0 (disabled)

Set the verbosity level of the debug output for the process management operations (fork, waitpid, kill, ...) of mpirun_rsh and mpispawn processes. The value 0 disables any debug output, a value of 1 enables a basic debug output, and a value of 2 enables a more verbose debug output.

Note: All debug output is disabled when MVAPICH2 is configured with the `--enable-fast=ndebug` option.