

MVAPICH2 1.2 User Guide

MVAPICH TEAM

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

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

Copyright ©2003-2008
Network-Based Computing Laboratory,
headed by Dr. D. K. Panda.
All rights reserved.

Last revised: August 20, 2008

Contents

| | | |
|----------|---|-----------|
| 1 | Overview of the Open-Source MVAPICH Project | 1 |
| 2 | How to use this User Guide? | 1 |
| 3 | MVAPICH2 1.2 Features | 2 |
| 4 | Installation Instructions | 7 |
| 4.1 | Building from a Tarball | 7 |
| 4.2 | Obtaining and Building the Source from Anonymous SVN | 7 |
| 4.3 | Configuring a build for OpenFabrics IB/iWARP | 8 |
| 4.4 | Configuring a build for uDAPL | 9 |
| 4.5 | Configuring a build for TCP/IP | 11 |
| 5 | Basic Usage Instructions | 12 |
| 5.1 | Compile MPI Applications | 12 |
| 5.2 | Run MPI Applications | 12 |
| 5.2.1 | Run MPI Applications using <code>mpirun_rsh</code> (for OpenFabrics IB/iWARP and uDAPL Devices) | 12 |
| 5.2.2 | Run MPI Applications using SLURM | 13 |
| 5.2.3 | Setting MPD Environment for Running Applications with <code>mpiexec</code> | 13 |
| 5.2.4 | Run MPI Applications using <code>mpiexec</code> with OpenFabrics IB Device | 14 |
| 5.2.5 | Run MPI Application with <code>mpiexec</code> using OpenFabrics iWARP Device | 15 |
| 5.2.6 | Run MPI Application using <code>mpiexec</code> with uDAPL Device | 15 |
| 5.2.7 | Run MPI Application using <code>mpiexec</code> with TCP/IP | 16 |
| 5.2.8 | Run MPI applications using ADIO driver for Lustre | 17 |
| 5.2.9 | Run MPI Applications using Shared Library Support | 17 |
| 5.2.10 | Run MPI Application using TotalView Debugger Support | 18 |
| 6 | Advanced Usage Instructions | 19 |
| 6.1 | Run MPI applications on Multi-Rail Configurations (for OpenFabrics IB/iWARP Devices) | 19 |

| | | |
|----------|--|-----------|
| 6.2 | Run MPI application with Customized Optimizations (for OpenFabrics IB/iWARP Devices) | 19 |
| 6.3 | Run MPI application with Checkpoint/Restart Support (for OpenFabrics IB Device) | 20 |
| 6.4 | Run MPI application with RDMA CM support (for OpenFabrics IB/iWARP Devices) | 22 |
| 6.5 | Run MPI application with Shared Memory Collectives | 23 |
| 6.6 | Run MPI Application with Hot-Spot and Congestion Avoidance (for OpenFabrics IB Device) | 23 |
| 6.7 | Run MPI Application with Network Fault Tolerance Support (for OpenFabrics IB Device) | 24 |
| 6.8 | Run MPI Application with User Defined CPU (Core) Mapping | 24 |
| 7 | Obtaining MVAPICH2 Library Version Information | 25 |
| 8 | Using OSU Benchmarks | 26 |
| 9 | FAQ and Troubleshooting with MVAPICH2 | 27 |
| 9.1 | General Questions and Troubleshooting | 27 |
| 9.1.1 | Invalid Communicators Error | 27 |
| 9.1.2 | Are <code>fork()</code> and <code>system()</code> supported? | 27 |
| 9.1.3 | Cannot Build with the PathScale Compiler | 27 |
| 9.1.4 | Why is there no <code>mpdboot</code> and <code>mpiexec</code> created? | 27 |
| 9.1.5 | MPI+OpenMP shows bad performance | 28 |
| 9.1.6 | Error message “No such file or directory” when using Lustre file system . . . | 28 |
| 9.1.7 | My program segfaults with: File locking failed in <code>ADIOI_Set_lock</code> ? | 28 |
| 9.1.8 | Running MPI programs built with <code>gfortran</code> | 28 |
| 9.1.9 | Does MVAPICH2 Work Across AMD and Intel Systems? | 29 |
| 9.2 | Failure with MPD or Launching Applications with MPD | 29 |
| 9.2.1 | Cannot find <code>mpd.conf</code> | 29 |
| 9.2.2 | The MPD <code>mpiexec</code> fails with “no msg recvd from mpd when expecting ack of request.” | 29 |
| 9.3 | With Gen2 Interface | 29 |
| 9.3.1 | Cannot Open HCA | 29 |

| | | |
|-----------|--|-----------|
| 9.3.2 | Checking state of IB Link | 29 |
| 9.3.3 | Undefined reference to <code>ibv_get_device_list</code> | 30 |
| 9.3.4 | Creation of CQ or QP failure | 30 |
| 9.3.5 | Hang with the HSAM Functionality | 30 |
| 9.3.6 | Failure with Automatic Path Migration | 30 |
| 9.3.7 | Error opening file | 30 |
| 9.3.8 | RDMA CM Address error | 30 |
| 9.3.9 | RDMA CM Route error | 31 |
| 9.4 | With Gen2-iWARP Interface | 31 |
| 9.4.1 | Error opening file | 31 |
| 9.4.2 | RDMA CM Address error | 31 |
| 9.4.3 | RDMA CM Route error | 31 |
| 9.4.4 | No Fortran interface on the MacOS platform | 31 |
| 9.5 | With uDAPL Interface | 31 |
| 9.5.1 | Cannot Open IA | 31 |
| 9.5.2 | DAT Insufficient Resource | 32 |
| 9.5.3 | Cannot Find <code>libdat.so</code> | 32 |
| 9.5.4 | Cannot Find <code>mpd.conf</code> | 32 |
| 9.5.5 | uDAPL over IB Does Not Scale Beyond 256 Nodes with <code>rdma_cm</code> Provider | 32 |
| 9.6 | Checkpoint/Restart | 32 |
| 10 | Scalable features for Large Scale Clusters and Performance Tuning | 34 |
| 10.1 | Job Launch Tuning | 34 |
| 10.2 | RDMA Based Point-to-Point tuning | 34 |
| 10.3 | Shared Receive Queue (SRQ) Tuning | 34 |
| 10.4 | Shared Memory Tuning | 35 |
| 10.5 | On-demand Connection Management Tuning | 35 |
| 10.6 | Scalable Collectives Tuning | 36 |
| 11 | MVAPICH2 Parameters | 37 |
| 11.1 | MV2_CKPT_FILE | 37 |

| | | |
|-------|--|----|
| 11.2 | MV2_CKPT_INTERVAL | 37 |
| 11.3 | MV2_CKPT_MAX_SAVE_CKPTS | 37 |
| 11.4 | MV2_CKPT_MPD_BASE_PORT | 38 |
| 11.5 | MV2_CKPT_MPIEXEC_PORT | 38 |
| 11.6 | MV2_CKPT_NO_SYNC | 38 |
| 11.7 | MV2_CM_RECV_BUFFERS | 38 |
| 11.8 | MV2_CM_SPIN_COUNT | 39 |
| 11.9 | MV2_CM_TIMEOUT | 39 |
| 11.10 | MV2_CPU_MAPPING | 39 |
| 11.11 | MV2_DAPL_PROVIDER | 39 |
| 11.12 | MV2_DEFAULT_MTU | 40 |
| 11.13 | MV2_ENABLE_AFFINITY | 40 |
| 11.14 | MV2_GET_FALLBACK_THRESHOLD | 40 |
| 11.15 | MV2_IBA_EAGER_THRESHOLD | 41 |
| 11.16 | MV2_INITIAL_PREPOST_DEPTH | 41 |
| 11.17 | MV2_MPD_RECVTIMEOUT_MULTIPLIER | 41 |
| 11.18 | MV2_MPIRUN_TIMEOUT | 41 |
| 11.19 | MV2_MT_DEGREE | 42 |
| 11.20 | MV2_NDREG_ENTRIES | 42 |
| 11.21 | MV2_NUM_HCAS | 42 |
| 11.22 | MV2_NUM_PORTS | 42 |
| 11.23 | MV2_NUM_QP_PER_PORT | 43 |
| 11.24 | MV2_NUM_RDMA_BUFFER | 43 |
| 11.25 | MV2_ON_DEMAND_THRESHOLD | 43 |
| 11.26 | MV2_PREPOST_DEPTH | 43 |
| 11.27 | MV2_PUT_FALLBACK_THRESHOLD | 44 |
| 11.28 | MV2_RDMA_CM_ARP_TIMEOUT | 44 |
| 11.29 | MV2_RDMA_CM_MAX_PORT | 44 |
| 11.30 | MV2_RDMA_CM_MIN_PORT | 44 |
| 11.31 | MV2_RNDV_PROTOCOL | 45 |

| | |
|---|----|
| 11.32MV2_R3_THRESHOLD | 45 |
| 11.33MV2_R3_NOCACHE_THRESHOLD | 45 |
| 11.34MV2_SHMEM_ALLREDUCE_MSG | 45 |
| 11.35MV2_SHMEM_BCAST_MSG | 46 |
| 11.36MV2_SHMEM_COLL_MAX_MSG_SIZE | 46 |
| 11.37MV2_SHMEM_COLL_NUM_COMM | 46 |
| 11.38MV2_SHMEM_REDUCE_MSG | 46 |
| 11.39MV2_SRQ_LIMIT | 46 |
| 11.40MV2_SRQ_SIZE | 47 |
| 11.41MV2_USE_APM | 47 |
| 11.42MV2_USE_APM_TEST | 47 |
| 11.43MV2_USE_BLOCKING | 47 |
| 11.44MV2_USE_COALESCE | 48 |
| 11.45MV2_USE_HSAM | 48 |
| 11.46MV2_USE_IWARP_MODE | 48 |
| 11.47MV2_USE_LAZY_MEM_UNREGISTER | 48 |
| 11.48MV2_USE_RDMA_CM | 49 |
| 11.49MV2_USE_RDMA_FAST_PATH | 49 |
| 11.50MV2_USE_RDMA_ONE_SIDED | 49 |
| 11.51MV2_USE_RING_STARTUP | 49 |
| 11.52MV2_USE_SHARED_MEM | 50 |
| 11.53MV2_USE_SHMEM_ALLREDUCE | 50 |
| 11.54MV2_USE_SHMEM_BARRIER | 50 |
| 11.55MV2_USE_SHMEM_BCAST | 50 |
| 11.56MV2_USE_SHMEM_COLL | 50 |
| 11.57MV2_USE_SHMEM_REDUCE | 51 |
| 11.58MV2_USE_SRQ | 51 |
| 11.59MV2_VBUF_POOL_SIZE | 51 |
| 11.60MV2_VBUF_SECONDARY_POOL_SIZE | 51 |
| 11.61MV2_VBUF_TOTAL_SIZE | 52 |

| | | |
|-------|-------------------------------|----|
| 11.62 | SMP_EAGERSIZE | 52 |
| 11.63 | SMP_LENGTH_QUEUE | 52 |
| 11.64 | SMP_NUM_SEND_BUFFER | 52 |
| 11.65 | SMP_SEND_BUF_SIZE | 53 |

1 Overview of the Open-Source MVAPICH Project

InfiniBand and 10GbE/iWARP are emerging as high-performance interconnects delivering low latency and high bandwidth. They are also getting 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 InfiniBand, iWARP and other RDMA-enabled interconnects 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: MVAPICH with MPI-1 semantics and MVAPICH2 with MPI-2 semantics. This *open-source* MPI software project started in 2001 and a first high-performance implementation was demonstrated at Supercomputing '02 conference. After that, this software has been steadily gaining acceptance in the HPC and InfiniBand community. As of August 20th, 2008, more than 740 organizations (National Labs, Universities and Industry) world-wide have downloaded this software from OSU’s web site directly. In addition, many InfiniBand and iWARP vendors, server vendors, and systems integrators have been incorporating MVAPICH/MVAPICH2 into their software stacks and distributing it. Several InfiniBand systems using MVAPICH/MVAPICH2 have obtained positions in the TOP 500 ranking. MVAPICH and MVAPICH2 are also available with the Open Fabrics Enterprise Distribution (OFED) stack. Both MVAPICH and MVAPICH2 distributions are available under BSD licensing.

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

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

This document contains necessary information for MVAPICH2 users to download, install, test, use, tune and troubleshoot MVAPICH2 1.2. As we get feedbacks from users and take care of bug-fixes, we introduce new tarballs and also continuously update this document. Thus, we strongly request 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 1.2.

In Section 3 we describe all the features in MVAPICH2 1.2. As you read through this section, please note our new features (highlighted as NEW) in the 1.2 series. 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 or off. Basic usage of MVAPICH2 is explained in Section 5. Section 6 provides instructions for running MVAPICH2 with some of the advanced features. Section 8 describes the usage of the

OSU Benchmarks. If you have any problems using MVAPICH2, please check Section 9 where we list some of the common problems people face. In Section 10 we suggest some tuning techniques for multi-thousand node clusters using some of our new features. Finally in Section 11 we list all important run-time parameters, their default values and a small description of what that parameter stands for.

3 MVAPICH2 1.2 Features

MVAPICH2 (MPI-2 over InfiniBand) is an MPI-2 implementation based on [MPICH2](#) ADI3 layer. It also supports all MPI-1 functionalities. MVAPICH2 1.2 is available as a single integrated package (with MPICH2 1.0.7).

The current release supports the following four underlying transport interfaces:

- **OpenFabrics Gen2-IB:** This interface supports all InfiniBand compliant devices based on the [OpenFabrics](#) Gen2 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.
- **OpenFabrics Gen2-iWARP:** This interface supports all iWARP compliant devices supported by OpenFabrics. For example, this layer supports Chelsio T3 adapters with the native iWARP mode.
- **uDAPL:** 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. It can also be used with Solaris uDAPL-IBTL implementation over InfiniBand adapters.
- **TCP/IP:** 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.

Please note that the support for VAPI interface has been dropped from MVAPICH2 1.2 because OpenFabrics interface is getting more popular. MVAPICH2 users still using VAPI interface are strongly requested to migrate to the OpenFabrics-IB interface.

MVAPICH2-1.2 delivers the same level of performance as MVAPICH 1.0.1, the latest release package of MVAPICH supporting MPI-1 standard. In addition, MVAPICH2 1.2 provides support and optimizations for other MPI-2 features, multi-threading and fault-tolerance (Checkpoint-restart). A complete set of features of MVAPICH2 1.2 are:

- Design for scaling to multi-thousand nodes with highest performance and reduced memory usage
 - (NEW) Scalable and robust daemon-less job startup

- * Enhanced and robust mpirun_rsh framework (non-MPD-based) to provide scalable job launching on multi-thousand core clusters
 - * Available for OpenFabrics (IB and iWARP) and uDAPL interfaces (including Solaris)
 - On-demand Connection Management: This feature enables InfiniBand connections to be setup dynamically, enhancing the scalability of MVAPICH2 on clusters of thousands of nodes
 - * Native InfiniBand Unreliable Datagram (UD) based asynchronous connection management for OpenFabrics Gen2-IB interface
 - * RDMA CM based on-demand connection management for OpenFabrics Gen2-iWARP and OpenFabrics Gen2-IB 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 OpenFabrics Gen2-IB interface
 - Hot-Spot Avoidance Mechanism (HSAM) for alleviating network congestion in large scale clusters. Available for OpenFabrics Gen2-IB interface
 - RDMA Read utilized for increased overlap of computation and communication for OpenFabrics device. Available for OpenFabrics Gen2-IB and iWARP interfaces
 - Shared Receive Queue (SRQ) with flow control. This design uses significantly less memory for MPI library. Available for OpenFabrics Gen2-IB interface.
 - Adaptive RDMA Fast Path with Polling Set for low-latency messaging. Available for OpenFabrics Gen2-IB and iWARP interfaces.
 - (NEW) Enhanced scalability for RDMA-based direct one-sided communication with less communication resource. Available for OpenFabrics (IB and iWARP) interfaces.
- Fault tolerance support
 - Checkpoint-restart support for application transparent systems-level fault tolerance. [BLCR-based](#) support using OpenFabrics Gen2-IB interface.
 - * (NEW) Checkpoint-restart with intra-node shared memory support
 - * Allows best performance and scalability with fault-tolerance support
 - 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 systems. Tested with ext3 (local disk), NFS and PVFS2
 - Network-Level fault tolerance with Automatic Path Migration (APM) for tolerating intermittent network failures over InfiniBand
 - (NEW) Enhancement to software installation
 - Full autoconf-based configuration

- Automatically detects system architecture and adapter types and optimizes MVAPICH2 for any particular installation
 - An application (mpiname) for querying the MVAPICH2 library version and configuration information
- Optimized intra-node communication support by taking advantage of shared-memory communication. Available for all interfaces.
 - Efficient Buffer Organization for Memory Scalability of Intra-node Communication
 - Multi-core optimized.
 - Optimized for Bus-based SMP and NUMA-Based SMP systems.
 - (NEW) Enhanced processor affinity using PLPA for multi-core architectures
 - * Allows user-defined flexible processor affinity
- Shared memory optimizations for collective communication operations. Available for all interfaces.
 - (NEW) Shared memory optimized algorithm for MPI_Bcast operations
 - (NEW) Optimized and tuned MPI_Alltoall
 - Efficient algorithms and optimizations for barrier, reduce and all-reduce operations
- Integrated multi-rail communication support. Available for OpenFabrics Gen2-IB and iWARP interfaces.
 - Multiple queue pairs per port
 - Multiple ports per adapter
 - Multiple adapters
 - Support for both one-sided and point-to-point operations
 - Support for OpenFabrics Gen2-iWARP interface and RDMA CM (for Gen2-IB).
- Multi-threading support. Available for all interfaces, including TCP/IP.
- 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 OpenFabrics Gen2-iWARP and RDMA CM (with Gen2-IB)
 - Enhanced scalability for RDMA-based direct one-sided communication with less communication resource
- Two modes of communication progress
 - Polling
 - Blocking (enables running multiple MPI processes/processor). Available for Open Fabrics Gen2-IB interface.

- Scalable job startup schemes
 - (NEW) Enhanced and robust mpirun_rsh framework
 - Using in-band IB communication with MPD
 - Support for SLURM
- 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.
- High Performance and Portable Support for multiple networks and operating systems through uDAPL interface.
 - InfiniBand (tested with)
 - * uDAPL over OpenFabrics Gen2-IB on Linux
 - * uDAPL over IBTL on Solaris

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.
- Shared Library Support for existing binary MPI application programs to run
- ROMIO Support for MPI-IO
 - Optimized, high-performance ADIO driver for Lustre
- Single code base for the following platforms (Architecture, OS, Compilers, Devices and Infini-Band adapters)
 - Architecture: (tested with) EM64T, Opteron and IA-32; IBM PPC and Mac G5
 - Operating Systems: (tested with) Linux and Solaris; and Mac OSX
 - Compilers: (tested with) gcc, intel, pathscale, pgi and sun studio
 - Devices: (tested with) OpenFabrics Gen2-IB, OpenFabrics Gen2-iWARP, and uDAPL; and TCP/IP
 - InfiniBand adapters:
 - * (tested with) Mellanox adapters with PCI-X and PCI-Express (SDR and DDR with mem-full and mem-free cards) and ConnectX
 - * PathScale adapter (through OpenIB/Gen2 support)
 - * IBM ehca adapter (through OpenIB/Gen2 support)
 - 10GigE adapters:
 - * (tested with) Chelsio T3 adapter with iWARP support

The MVAPICH2 1.2 package and the project also includes the following provisions:

- [Public SVN](#) access of the codebase
- 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 OpenFabrics IB/iWARP on Linux and uDAPL on Solaris. uDAPL and TCP/IP devices can also be explicitly selected on Linux. The installation section provides generic instructions for building from a Tarball or our latest sources. Please see the subsection for the device you are targeting for specific configuration instructions.

4.1 Building from a Tarball

The MVAPICH2 1.2.0 source code package includes MPICH2 1.0.7. 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 mvapich2-1.2.0.tar.gz
$ cd mvapich2-1.2.0
$ ./configure
$ make
$ make install
```

4.2 Obtaining and Building the Source from Anonymous SVN

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:

- “branches/1.2” is a stable version with bug fixes. New features are not added to this branch.
 - To obtain the source code from branches/1.2:

```
$ svn co https://mvapich.cse.ohio-state.edu/svn/mpi/mvapich2/branches/1.2
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
```
- “tags/1.2” is the exact version released with no updates for bug fixes or new features.

- To obtain the source code from tags/1.2:

```
$ svn co https://mvapich.cse.ohio-state.edu/svn/mpi/mvapich2/tags/1.2  
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. If the command "autoconf" on your machine does not run autoconf 2.59 or later, but you do have a new enough autoconf available, then you can specify the correct one with the AUTOCONF environment variable (the AUTOHEADER environment variable is similar). Once you've prepared the working copy by running maint/updatefiles, just follow the usual configuration and build procedure:

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

4.3 Configuring a build for OpenFabrics IB/iWARP

OpenFabrics IB/iWARP is the default interface on Linux. It can be explicitly selected by configuring with:

```
$ ./configure --with-rdma=gen2
```

Configuration Options for OpenFabrics IB/iWARP

- Berkeley Lab Checkpoint/Restart Support
 - Default: disabled
 - Enable: `--with-pm=mpd --enable-blcr`
`--with-blcr-libpath=path --with-blcr-include=path`
- Header Caching
 - Default: enabled
 - Disable: `--disable-header-caching`
- 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 RDMA CM
 - Default: enabled, except when BLCR support is enabled
 - Disable: `--disable-rdma-cm`
- 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`

4.4 Configuring a build for uDAPL

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

```
$ ./configure --with-rdma=udapl
```

Configuration options for uDAPL

- 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
- 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`
- Header Caching
 - Default: enabled
 - Disable: `--disable-header-caching`
- 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`
- I/O Bus
 - Default: PCI Express
 - Specify: `--with-io-bus=type`
 - * Where `type` is one of:
 - `PCIE_X` for PCI Express
 - `PCL_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`

4.5 Configuring a build for TCP/IP

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
```

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>

5 Basic Usage Instructions

5.1 Compile MPI 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 MPI Applications

5.2.1 Run MPI Applications using `mpirun_rsh` (for OpenFabrics IB/iWARP and uDAPL Devices)

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.

Examples of running programs using `mpirun_rsh`:

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

This command launches `cpi` on nodes `n0`, `n1`, `n2` and `n3`, one process per node. By default `ssh` is used.

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

This command launches `cpi` on nodes `n0`, `n1`, `n2` and `n3`, one process per each node using `rsh` instead of `ssh`.

```
$ 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`. ie. 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”.

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 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 10.

Other options of `mpirun_rsh` can be obtained using

```
$ mpirun_rsh --help
```

5.2.2 Run MPI Applications using SLURM

SLURM is an open-source resource manager designed by Lawrence Livermore National Laboratory. SLURM software package and its related documents can be downloaded from:

<http://www.llnl.gov/linux/slurm/>

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

```
$ srun -n2 --mpi=none ./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.

5.2.3 Setting MPD Environment for Running Applications with mpiexec

Prerequisites: `ssh` should be enabled between the front nodes and the computing nodes.

Please follow these steps to setup MPD:

- Please ensure that you have `.mpd.conf` and `.mpdpasswd` in your home directory. They are typically a single line file containing the following:

```
secretword=56rtG9.
```

The content of `.mpd.conf` and `.mpdpasswd` should be exactly the same. (Ofcourse, the secretword should be different.)

- Please include MPD path into your path

```
$ export MPD_BIN=$MVAPICH2_HOME/bin  
$ export PATH=$MVAPICH2_HOME/bin:$PATH
```

`$MVAPICH2_HOME` is the installation path of your MVAPICH2, as specified by `$PREFIX` when you configure MVAPICH2.

- Prepare hostfile

Specify the hostnames of the compute nodes in a file. If you have a hostfile like:

```
n0
n1
n2
n3
```

then one process per node will be started on each one of these compute nodes.

- Start MPD Daemons on the compute nodes

```
$ mpdboot -n 4 -f hostfile
```

Note: The command, `mpdboot`, also takes a default hostfile name `mpd.hosts`. If you have created the hostfile as `mpd.hosts`, you can omit the option “-f hostfile”.

- Check status of MPD Daemons on the compute nodes (optional)

```
$ mpdtrace
```

This should list all the nodes specified in the hostfile, not necessarily in the order specified in the hostfile.

Up to now, we have specified setting up the environment, which is independent of the underlying device supported by MVAPICH2. In the next sections, we present details specific to different devices.

5.2.4 Run MPI Applications using `mpiexec` with OpenFabrics IB Device

To start multiple processes, `mpiexec` can be used in the following fashion:

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

Four processes will be started on the compute nodes `n0`, `n1`, `n2` and `n3`. `mpiexec` can also be run with several options. “`$ mpiexec --help`” lists all the possible options. A useful option is to specify a machinefile which holds the process mapping to machines. It can also be used to specify the number of processes to be run on each host. The machinefile option can be used with `mpiexec` as follows:

```
$ mpiexec -machinefile mf -n 4 ./cpi
```

where the machine file “mf” contains the process to machine mapping. For example, if you want to run all the 4 processes on `n0`, then “mf” contains the following lines:

```
$ cat mf
n0
n0
n0
n0
```

Environmental variables can be set with `mpiexec` as follows:

```
$ mpiexec -n 4 -env ENV1 value1 -env ENV2 value2 ./cpi
```

Note that the environmental variables should be put immediately before the executable file. The `mpiexec` command also propagates exported variables in its runtime environment to all processes by default. Exporting a variable before running `mpiexec` has the same effect as explicitly passing its value with the `-env` command line option. The command above could be done in the following manner when using a Bourne shell derivative:

```
$ export ENV1=value1
$ export ENV2=value2
$ mpiexec -n 4 ./cpi
```

5.2.5 Run MPI Application with `mpiexec` using OpenFabrics iWARP Device

In MVAPICH2, Gen2-iWARP support is enabled with the use of the run time environment variable `‘MV2_USE_IWARP_MODE’`.

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

- **Setup the RDMA CM device:** RDMA CM device 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 multirail configurations).

```
$ echo 10.1.1.1 >> /etc/mv2.conf
```

Programs can be executed as follows:

```
$ mpiexec -n 4 -env MV2_USE_IWARP_MODE 1 -env ENV1 value1 prog
```

The iWARP device also provides totalview debugging and shared library support. Please refer to section 5.2.9 and 5.2.10 for shared library and totalview support, respectively.

5.2.6 Run MPI Application using `mpiexec` with uDAPL Device

MVAPICH2 can be configured with the uDAPL device, as described in the Section 4.4 . 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 -env MV2_DAPL_PROVIDER OpenIB-cma ./cpi
```

or:

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

```
$ mpiexec -n 4 ./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 runtime. 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 device also provides totalview debugging and shared library support. Please refer to section 5.2.9 and 5.2.10 for shared library and totalview support, respectively.

5.2.7 Run MPI Application using mpiexec with TCP/IP

If you would like to run an MPI job using IPoIB but your IB card is not the default interface for ip traffic you have two options. For both of the options, 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 | 192.168.1.1 |
| compute3 | 192.168.0.3 | 192.168.1.1 |
| compute4 | 192.168.0.4 | 192.168.1.1 |

The MPI Job Uses IPoIB: In this scenario, you will start up `mpd` like normal. However, you will need to create a machine file for `mpiexec` that tells `mpiexec` to use a particular interface.

Example:

```
$ cat - > $(MPD_HOSTFILE) compute1
compute2
compute3
compute4
```

```
$ mpdboot -n 4 -f $(MPD_HOSTFILE)
compute1 ifhn=192.168.1.1
compute2 ifhn=192.168.1.2
compute3 ifhn=192.168.1.3
compute4 ifhn=192.168.1.4
```

The `ifhn` portion tells `mpiexec` to use the interface associated with that ip address for each machine. You can now run your MPI application using IPoIB similar to the following.

```
$ mpiexec -n $(NUM_PROCESS) -f $(MACHINE_FILE) $(MPI_APPLICATION)
```

Both MPD And the MPI Job Use IPoIB: In this scenario you will start up `mpd` in a modified fashion. However, you will not need to create a machine file for `mpiexec`. Your hostsfile for `mpdboot` must contain the ip addresses, or hostnames mapped to these addresses, of each machine's IPoIB interface. The only exception is that you do not list the ip address or hostname of the local machine. This will be specified on the command line of the `mpdboot` command using the `-ifhn` option. Example:

```
$ cat - > $(MPD_HOSTFILE)
192.168.1.2
192.168.1.3
192.168.1.4
```

```
$ mpdboot -n 4 -f $HOSTSFILE --ifhn=192.168.1.1
```

The `-ifhn` option tells `mpdboot` to use the interface corresponding to that ip address to create the `mpd` ring and run MPI jobs. You can now run your MPI application using IPoIB similar to the following.

```
$ mpiexec -n $(NUM_PROCES) $(MPI_APPLICATION)
```

Note: For both options, you can replace the IPoIB addresses with aliases

5.2.8 Run MPI applications using ADIO driver for Lustre

MVAPICH2 contains optimized Lustre ADIO support for the OpenFabrics/Gen2 device. 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
```

5.2.9 Run MPI Applications using Shared Library Support

MVAPICH2 provides shared library support. This feature allows you to build your application on top of MPI shared library. If you choose this option, you still will be able to compile applications with static libraries. But as default, when you have shared library support enabled your applications will be built on top of shared libraries automatically. the following commands provide some examples of how to build and run your application with shared library support.

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

```
$ mpicc -o cpi cpi.c
```
- To execute an application compiled with shared library support, you need to specify the path to the shared library by putting `LD_LIBRARY_PATH=path-to-shared-libraries` in the command line. For example:

```
$ mpiexec -np 2 -env LD_LIBRARY_PATH $MVAPICH2_BUILD/lib/shared ./cpi
```

or

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

- To disable MVAPICH2 shared library support even if you have installed MVAPICH2, run the following command:

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

5.2.10 Run MPI Application 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.

- Define ssh as a TVDSVRLAUNCHCMD variable in your default shell. For example, with bashrc, you can do:

```
$ echo "export TVDSVRLAUNCHCMD=ssh" >> $HOME/.bashrc
```
- Configure MVAPICH2 with the configure options `--enable-g=dbg --enable-sharedlibs=kind --enable-debuginfo` in addition to the default options and then build MVAPICH2.
- Compile your program with a flag `-g`:

```
$ mpicc -g -o prog prog.c
```
- Define the correct path to TotalView as the TOTALVIEW variable. For example, under bash shell:

```
$ export MPIEXEC_TOTALVIEW=<path_to_TotalView>
```
- Run your program:

```
$ mpiexec -tv -np 2 -env LD_LIBRARY_PATH $MVAPICH2_BUILD/lib/shared:  
$MVAPICH2_BUILD/lib prog
```
- Troubleshooting:
 - X authentication errors: check if you have enabled X Forwarding

```
$ cat "ForwardX11 yes" >> $HOME/.ssh/config
```
 - ssh authentication error: ssh to the computer node with its long form hostname, for example, ssh i0.domain.osu.edu

6 Advanced Usage Instructions

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

6.1 Run MPI applications on Multi-Rail Configurations (for OpenFabrics IB/iWARP Devices)

MVAPICH2 has integrated multi-rail support. Run-time variables are used to specify the control parameters of the multi-rail support; number of adapters with `MV2_NUM_HCAS` (section 11.21), number of ports per adapter with `MV2_NUM_PORTS` (section 11.22), and number of queue pairs per port with `MV2_NUM_QP_PER_PORT` (section 11.23). Those variables are default to 1 if you do not specify them. Following is an example to run 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
or
$ mpiexec -n 2 -env MV2_NUM_HCAS 2 -env MV2_NUM_PORTS 1 -env
MV2_NUM_QP_PER_PORT 1 prog
```

Note that you don't need to specify `MV2_NUM_PORTS` and `MV2_NUM_QP_PER_PORT` since they default to 1, so you can type:

```
$ mpirun_rsh -np 2 n0 n1 MV2_NUM_HCAS=2 prog
or
$ mpiexec -n 2 -env MV2_NUM_HCAS 2 prog
```

6.2 Run MPI application with Customized Optimizations (for OpenFabrics IB/iWARP Devices)

In MVAPICH2-1.2, 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:

- **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 -env MV2_USE_RDMA_FAST_PATH 0 prog
- **Shared-receive Queue:** This feature is available only with Gen2-IB devices. This is targeted for using Shared Receive Queue (SRQ). Default: on; to disable:
\$ mpirun_rsh -np 2 n0 n1 MV2_USE_SRQ=0 prog
or

```
$ mpiexec -n 2 -env MV2_USE_SRQ 0 prog
```

- **Optimizations for one sided communication:** One sided operations can be directly built on RDMA operations. Currently this scheme will be disabled if on-demand connection management is used. Default: on; to disable:

```
$ mpirun_rsh -np 2 n0 n1 MV2_USE_RDMA_ONE_SIDED=0 prog
```

or

```
$ mpiexec -n 2 -env MV2_USE_RDMA_ONE_SIDED 0 prog
```

- **Lazy memory unregistration:** user-level registration cache. Default: on; to disable:

```
$ mpirun_rsh -np 2 n0 n1 MV2_USE_LAZY_MEM_UNREGISTER=0 prog
```

or

```
$ mpiexec -n 2 -env MV2_USE_LAZY_MEM_UNREGISTER 0 prog
```

6.3 Run MPI application with Checkpoint/Restart Support (for OpenFabrics IB Device)

MVAPICH2 provides system-level checkpoint/restart functionality for the OpenFabrics Gen2-IB interface. Three methods are provided to invoke checkpointing: Manual, Automated and Application Initiated Synchronous Checkpointing. In order to utilize the checkpoint/restart functionality there a couple of steps that need to be followed.

- Download and install the BLCR (Berkeley Lab's Checkpoint/Restart) package. The packages can be downloaded from [this webpage](#).
- 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 4.3.
- BLCR's 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.
- The extended version of Multi-Purpose Daemon (MPD), provided in MVAPICH2 package, needs to be used as the process manager. It is to be noted that currently this is the ONLY process manager supporting the C/R feature.

And 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.

Now, your system is set up to use the Checkpoint/Restart features of MVAPICH2. There are several parameters related to MVAPICH2 to be setup to control the configuration and usage of this feature.

- `MV2_CKPT_FILE`: This parameter specifies the path and the base filename 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 `export MV2_CKPT_FILE=/mnt/pvfs2/ckptfile` will let the checkpoint files being stored in pvfs2 file system. See Section 11.1 for details.
- `MV2_CKPT_INTERVAL`: This parameter can be used to enable automatic checkpointing. See Section 11.2 for details.
- `MV2_CKPT_MAX_SAVE_CKPTS`: This parameter is used to limit the number of checkpoints saved on file system. See Section 11.3 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 11.6 for details.
- `MV2_CKPT_MPD_BASE_PORT`, See Section 11.4 for details.
- `MV2_CKPT_MPIEXEC_PORT`, See Section 11.4 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`. When the user decides to take a checkpoint, the users can issue a BLCR command called `'cr_checkpoint'` with the process id (PID) of the `mpiexec` process. In order to simplify the process, the script `mv2_checkpoint` can be used. This script is available in the same directory as `mpiexec`.
- **Automated Checkpointing:** In this mode, the user can launch the MPI application normally using `mpiexec`. 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 11.2 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. Synchronous checkpointing is enabled by default when `ENABLE_CKPT=yes` is set in make script. To disable synchronous checkpointing, remove `-DSYNC_CKPT` from `CR_FLAG` in the make script. 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.

```

#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;
}

```

To restart a job from a 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 `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>`).

Please refer to the Section 9.6 for troubleshooting with Checkpoint/Restart.

6.4 Run MPI application with RDMA CM support (for OpenFabrics IB/iWARP Devices)

In MVAPICH2, for using RDMA CM the runtime variable `MV2_USE_RDMA_CM` needs to be used as described in 11.

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 device:** RDMA CM device 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 multirail 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 -env MV2_USE_RDMA_CM 1 prog
```

6.5 Run MPI application with Shared Memory Collectives

In MVAPICH2, support for shared memory based collectives has been enabled for MPI applications running over OpenFabrics Gen2-IB, Gen2-iWARP and uDAPL stack. 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 runtime by using the following parameters:

- MV2_USE_SHMEM_COLL (section 11.56)
- MV2_USE_SHMEM_ALLREDUCE (section 11.53)
- MV2_USE_SHMEM_REDUCE (section 11.57)
- MV2_USE_SHMEM_BARRIER (section 11.54)
- MV2_USE_SHMEM_BCAST (section 11.55)

Please refer to Section 11 for further details.

6.6 Run MPI Application with Hot-Spot and Congestion Avoidance (for OpenFabrics IB Device)

MVAPICH2 supports hot-spot and congestion avoidance using InfiniBand multi-pathing mechanism. This support is available for MPI applications using OpenFabrics stack and InfiniBand adapters.

To enable this functionality, a run-time variable, MV2_USE_HSAM (Section 11.45) can be enabled, as shown in the following example:

```
$ mpirun_rsh -np 2 n0 n1 MV2_USE_HSAM=1 ./cpi  
or  
$ mpiexec -n 2 -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 11.23).

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.

6.7 Run MPI Application with Network Fault Tolerance Support (for OpenFabrics IB Device)

MVAPICH2 supports network fault recovery by using InfiniBand Automatic Path Migration mechanism. This support is available for MPI applications using OpenFabrics stack and InfiniBand adapters.

To enable this functionality, a run-time variable, `MV2_USE_APM` (section 11.41) can be enabled, as shown in the following example:

```
$ mpirun_rsh -np 2 n0 n1 MV2_USE_APM=1 ./cpi
```

or

```
$ mpiexec -n 2 -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 11.42). 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 -env MV2_USE_APM 1 -env MV2_USE_APM_TEST 1 ./cpi
```

6.8 Run MPI Application with User Defined CPU (Core) Mapping

MVAPICH2 supports user defined CPU mapping through Portable Linux Processor Affinity (PLPA) library (<http://www.open-mpi.org/projects/plpa/>). 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`.

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 -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 “:”.

PLPA supports more flexible notations when specifying core mapping. More details can be found at:

<http://www.open-mpi.org/community/lists/plpa-users/2007/04/0035.php>

7 Obtaining MVAPICH2 Library Version 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:

Usage: [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

8 Using OSU Benchmarks

If you have arrived at this point, you have successfully installed MVAPICH2. Congratulations!! In the `mvapich2-1.2/osu_benchmarks` directory, we provide these basic performance tests:

- One-way latency test (`osu_latency.c`)
- One-way bandwidth test (`osu_bw.c`)
- Bi-directional bandwidth (`osu_bibw.c`)
- Multiple Bandwidth / Message Rate test (`osu_mbw_mr.c`)
- One-sided put latency (`osu_put_latency.c`)
- One-sided put bandwidth (`osu_put_bw.c`)
- One-sided put bi-directional bandwidth (`osu_put_bibw.c`)
- One-sided get latency (`osu_get_latency.c`)
- One-sided get bandwidth (`osu_get_bw.c`)
- One-sided accumulate latency (`osu_acc_latency.c`)
- One-way multi-threaded latency test (`osu_latency_mt.c`) - Multi-threading support must be compiled in to run this test.
- Broadcast test (`osu_bcast.c`)
- Alltoall test (`osu_alltoall.c`)

The benchmarks are also periodically updated. The latest copy of the benchmarks can be downloaded from <http://mvapich.cse.ohio-state.edu/benchmarks/>. Sample performance numbers for these benchmarks on representative platforms with InfiniBand and iWARP 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.

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 five underlying transport interfaces, namely OpenFabrics (Gen2), OpenFabrics (Gen2-iWARP), uDAPL and TCP/IP. Based on the underlying library being utilized, the troubleshooting steps may be different. However, some of the troubleshooting hints are common for all underlying libraries. Thus, in this section, we have divided the troubleshooting tips into four sections: General troubleshooting and Troubleshooting over any one of the 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 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 “`-LN0:simd=0`” C compiler option. This can be set in the build script similarly to:

```
export CC='pathcc -LN0: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.

9.1.4 Why is there no `mpdboot` and `mpiexec` created?

`mvapich2-1.2` introduces a new startup mechanism with much improved scalability on large scale clusters. The new mechanism uses `mpirun_rsh` to launch the MPI processes. Refer to Section 5.2.1

for details to run applications using `mpirun_rsh`.

Users are recommended to use `mpirun_rsh`, especially on large clusters. If you want to use the traditional startup through `mpd` daemon, however, you can do so by configure `mvapich2` with `--with-pm=mpd` option. Please refer to Section 4 for details.

9.1.5 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 `-env MV2_ENABLE_AFFINITY 0`.

9.1.6 Error message “No such file or directory” when using Lustre file system

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

- Lustre is setup correctly, and that you are able to create, read to and write from files in the Lustre mounted directory.
- The Lustre directory is mounted on all nodes on which MVAPICH2 processes with ADIO support for Lustre are running.
- The path to the file is correctly specified.
- The permissions for the file or directory are correctly specified.

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

If you are using ADIO support for Lustre, the recent Lustre releases require an additional mount option to have correct file locks.

So please include the following option with your lustre mount command: `”-o localflock”`.

For example:

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

9.1.8 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 `GFORTTRAN_UNBUFFERED_ALL` variable can be set to “y” and exported into the environment before using the `mpiexec` command to launch the program, as done in the bash shell example 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
```

9.1.9 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.

9.2 Failure with MPD or Launching Applications with MPD

9.2.1 Cannot find mpd.conf

If you get this error, please set your `.mpd.conf` and `.mpdpasswd` files.

9.2.2 The MPD mpiexec fails with “no msg recvd from mpd when expecting ack of request.”

This failure may be an indication that there is a problem with your cluster configuration. If you are confident in the correctness of your cluster configuration, then you can tune the timeout with `MV2_MPD_RECVTIMEOUT_MULTIPLIER`.

9.3 With Gen2 Interface

9.3.1 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
```

9.3.2 Checking state of IB Link

In order to check the status of the IB link, one of the following commands can be used:

```
% ibstatus
```

or

```
% ibv_devinfo.
```

9.3.3 Undefined reference to `ibv_get_device_list`

Add `-DGEN2_OLD_DEVICE_LIST_VERB` macro to `CFLAGS` and rebuild `MVAPICH2-gen2`. If this happens, this means that your Gen2 installation is old and needs to be updated.

9.3.4 Creation of CQ or QP failure

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

In `/etc/security/limits.conf` add the following line:

```
* soft memlock phys_mem_in_KB
```

After this, add the following line to `/etc/init.d/sshd` and restart `sshd`

```
ulimit -l phys_mem_in_KB
```

9.3.5 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.

9.3.6 Failure with Automatic Path Migration

`MVAPICH2` 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 11.41 and section 11.42

9.3.7 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 6.4 for more details on this.

9.3.8 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.

9.3.9 RDMA CM Route error

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

9.4 With Gen2-iWARP Interface

9.4.1 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 5.2.5 for more details on this.

9.4.2 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.

9.4.3 RDMA CM Route error

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

9.4.4 No Fortran interface on the MacOS platform

To enable Fortran support, you would need to install the IBM compiler located at (there is a 60-day free trial version) available from [IBM](#).

Once you unpack the tarball, you can customize and use `make.mvapich2.vapi` to compile and install the package or manually configure, compile and install the package.

9.5 With uDAPL Interface

9.5.1 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 5.2.6). 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`.

9.5.2 DAT Insufficient Resource

If you configure MVAPICH2 with uDAPL and see this error, you need to reduce the value of the environmental variable `RDMA_DEFAULT_MAX_WQE` depending on the underlying network.

9.5.3 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
or
$ export LD_LIBRARY_PATH=/path/to/libdat.so:$LD_LIBRARY_PATH
$ mpiexec -n 2 ./a.out
```

9.5.4 Cannot Find mpd.conf

If you get this error, please set your `.mpd.conf` and `.mpdpasswd` files as mentioned in Section 5.2.4.

9.5.5 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>

9.6 Checkpoint/Restart

Please make sure the following things for a successful restart:

- The MPD must be started on all the compute nodes and the console node before a 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.

FAQ regarding Berkeley Lab Checkpoint/Restart (BLCR) can be found at <http://upc-bugs.lbl.gov/blcr/doc/html/FAQ.html> And the userguide 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 as mvapich-discuss@cse.ohio-state.edu.

10 Scalable features for Large Scale Clusters and Performance Tuning

MVAPICH2 provides many different parameters for tuning performance for a wide variety of platforms and applications. These parameters can be either compile time parameters or runtime parameters. please refer to Section 10 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.

10.1 Job Launch Tuning

MVAPICH2 1.2 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` (section 11.19).

10.2 RDMA Based Point-to-Point tuning

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

- `MV2_VBUF_TOTAL_SIZE`
- `MV2_NUM_RDMA_BUFFER`
- `MV2_RDMA_VBUF_POOL_SIZE`

`MV2_RDMA_VBUF_POOL_SIZE` is a fixed number of pool of vbufs. These vbufs can be shared among all different connections depending on the communication needs of each connection.

On the other hand, the product of `MV2_VBUF_TOTAL_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.

In MVAPICH2-1.2, `MV2_VBUF_TOTAL_SIZE` is adjustable by environmental variables. Please refer to Section 11.61 for details.

10.3 Shared Receive Queue (SRQ) Tuning

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

- `MV2_SRQ_SIZE` (11.40)
- `MV2_SRQ_LIMIT` (11.39)

MV2_SRQ_SIZE is the maximum size of the Shared Receive Queue. You may increase this to value 1000 if the application requires very large number of processors (4K and beyond). MV2_SRQ_LIMIT defines the low watermark for the flow control handler. This can be reduced if your aim is to reduce the number of interrupts.

10.4 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 (11.63) and SMP_EAGER_SIZE (11.62). The two main parameters which are used for tuning shared memory performance for large messages are SMP_SEND_BUF_SIZE(11.65) and SMP_NUM_SEND_BUFFER (11.64).

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.

10.5 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 (11.25), MV2_CM_RECV_BUFFERS (11.7), MV2_CM_TIMEOUT (11.9), and MV2_CM_SPIN_COUNT (11.8). The first one applies to Gen2-IB, Gen2-iWARP and uDAPL devices and the other three only apply to Gen2 device.

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.

10.6 Scalable Collectives Tuning

MVAPICH2 uses shared memory to get the best performance for many collective operations: MPI.Allreduce, MPI.Reduce, MPI.Barrier, MPI.Bcast.

The important parameters for tuning these collectives are as follows. For MPI.Allreduce, the optimized shared memory algorithm is used until the MV2_SHMEM_ALLREDUCE_MSG(11.34).

Similarly for MPI.Reduce the corresponding threshold is MV2_SHMEM_REDUCE_MSG(11.38) and for MPI.BCAST the threshold can be set using MV2_SHMEM_BCAST_MSG(11.35).

11 MVAPICH2 Parameters

11.1 MV2_CKPT_FILE

- Class: Run Time
- Default: /tmp/ckpt
- Applicable interface(s): Gen2

This parameter specifies the path and the base filename 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.

11.2 MV2_CKPT_INTERVAL

- Class: Run Time
- Default: 0
- Unit: minutes
- Applicable interface(s): Gen2

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.

11.3 MV2_CKPT_MAX_SAVE_CKPTS

- Class: Run Time
- Default: 0
- Applicable interface(s): Gen2

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.

11.4 MV2_CKPT_MPD_BASE_PORT

- Class: Run Time
- Default: 24678
- Applicable interface(s): Gen2

This parameter specifies the ports of socket connections to pass checkpointing control messages between MPD manager and MPI process. Users need to have a set of unused ports starting with \$MV2_CKPT_MPD_BASE_PORT on the compute nodes. The used port will be the \$MV2_CKPT_MPD_BASE_PORT + <process rank> for each MPI processes.

11.5 MV2_CKPT_MPIEXEC_PORT

- Class: Run Time
- Default: 14678
- Applicable interface(s): Gen2

This parameter specifies the port of the socket connection for passing checkpointing control messages on MPI job console node. Users need to have an unused port to be set to \$MV2_CKPT_MPIEXEC_PORT on the console node.

11.6 MV2_CKPT_NO_SYNC

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

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.

11.7 MV2_CM_RECV_BUFFERS

- Class: Run Time
- Default: 1024
- Applicable interface(s): Gen2

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.

11.8 MV2_CM_SPIN_COUNT

- Class: Run Time
- Default: 5000
- Applicable interface(s): Gen2

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.

11.9 MV2_CM_TIMEOUT

- Class: Run Time
- Default: 500
- Unit: milliseconds
- Applicable interface(s): Gen2

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.

11.10 MV2_CPU_MAPPING

- Class: Run Time
- Default: Local rank based mapping
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

This allows users to specify process to CPU (core) mapping. The detailed usage of this parameter is described in Section 6.8. This parameter will not take effect if `MV2_ENABLE_AFFINITY` is set to 0.

11.11 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

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

11.12 MV2_DEFAULT_MTU

- Class: Run time
- Default: MTU1024
- Applicable interface(s): Gen2, uDAPL

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

11.13 MV2_ENABLE_AFFINITY

- Class: Run time
- Default: 1
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

Enable CPU affinity by setting MV2_ENABLE_AFFINITY to 1 or disable it by setting MV2_ENABLE_AFFINITY to 0.

11.14 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): Gen2, Gen2-iWARP, uDAPL

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

11.15 MV2_IBA_EAGER_THRESHOLD

- Class: Run time
- Default: Architecture dependent (12KB for IA-32)
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

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

11.16 MV2_INITIAL_PREPOST_DEPTH

- Class: Run time
- Default: 10
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

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

11.17 MV2_MPD_RECVTIMEOUT_MULTIPLIER

- Class: Run time
- Default: 0.05

The multiplier to be added to the MPD mpiexec timeout for each process in a job.

11.18 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 runtime environment (for e.g. through export in the bash shell).

11.19 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 runtime environment (for e.g. through `export` in the bash shell).

11.20 MV2_NDREG_ENTRIES

- Class: Run time
- Default: 1000
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

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.

11.21 MV2_NUM_HCAS

- Class: Run time
- Default: 1
- Applicable interface(s): Gen2, Gen2-iWARP

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

11.22 MV2_NUM_PORTS

- Class: Run time
- Default: 1
- Applicable interface(s): Gen2, Gen2-iWARP

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

11.23 MV2_NUM_QP_PER_PORT

- Class: Run time
- Default: 1
- Applicable interface(s): Gen2, Gen2-iWARP

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.

11.24 MV2_NUM_RDMA_BUFFER

- Class: Run time
- Default: Architecture dependent (32 for EM64T)
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

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.

11.25 MV2_ON_DEMAND_THRESHOLD

- Class: Run Time
- Default: 32
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

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.

11.26 MV2_PREPOST_DEPTH

- Class: Run time
- Default: 64
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

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

11.27 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): Gen2, Gen2-iWARP, uDAPL

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

11.28 MV2_RDMA_CM_ARP_TIMEOUT

- Class: Run Time
- Default: 2000 ms
- Applicable interface(s): Gen2, Gen2-iWARP

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

11.29 MV2_RDMA_CM_MAX_PORT

- Class: Run Time
- Default: Unset
- Applicable interface(s): Gen2, Gen2-iWARP

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.

11.30 MV2_RDMA_CM_MIN_PORT

- Class: Run Time
- Default: Unset
- Applicable interface(s): Gen2, Gen2-iWARP

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.

11.31 MV2_RNDV_PROTOCOL

- Class: Run time
- Default: RPUT
- Applicable interface(s): Gen2, Gen2-iWARP

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).

11.32 MV2_R3_THRESHOLD

- Class: Run time
- Default: 4096
- Applicable interface(s): Gen2, Gen2-iWARP

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

11.33 MV2_R3_NOCACHE_THRESHOLD

- Class: Run time
- Default: 32768
- Applicable interface(s): Gen2, Gen2-iWARP

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.

11.34 MV2_SHMEM_ALLREDUCE_MSG

- Class: Run Time
- Default: $1 \ll 15$
- Applicable interface(s): Gen2, Gen2-iWARP

The shmem allreduce is used for messages less than this threshold.

11.35 MV2_SHMEM_BCAST_MSG

- Class: Run Time
- Default: $1 \ll 20$
- Applicable interface(s): Gen2, Gen2-iWARP

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

11.36 MV2_SHMEM_COLL_MAX_MSG_SIZE

- Class: Run Time
- Applicable interface(s): Gen2, Gen2-iWARP

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

11.37 MV2_SHMEM_COLL_NUM_COMM

- Class: Run Time
- Applicable interface(s): Gen2, Gen2-iWARP

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

11.38 MV2_SHMEM_REDUCE_MSG

- Class: Run Time
- Default: $1 \ll 10$
- Applicable interface(s): Gen2, Gen2-iWARP

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

11.39 MV2_SRQ_LIMIT

- Class: Run Time
- Default: 30
- Applicable interface(s): Gen2, Gen2-iWARP

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

11.40 MV2_SRQ_SIZE

- Class: Run Time
- Default: 512
- Applicable interface(s): Gen2, Gen2-iWARP

This is the maximum number of work requests allowed on the Shared Receive Queue.

11.41 MV2_USE_APM

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

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

11.42 MV2_USE_APM_TEST

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

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

11.43 MV2_USE_BLOCKING

- Class: Run time
- Default: 0
- Applicable interface(s): Gen2

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.

11.44 MV2_USE_COALESCE

- Class: Run time
- Default: set
- Applicable interface(s): Gen2, Gen2-iWARP

Setting this parameter enables message coalescing to increase small message throughput

11.45 MV2_USE_HSAM

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

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 6.6 for detailed information.

11.46 MV2_USE_IWARP_MODE

- Class: Run Time
- Default: unset
- Applicable interface(s): Gen2, Gen2-iWARP

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.

11.47 MV2_USE_LAZY_MEM_UNREGISTER

- Class: Run time
- Default: set
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

Setting this parameter enables mvapich2 to use memory registration cache.

11.48 MV2_USE_RDMA_CM

- Class: Run Time
- Default: Network Dependant (set for Gen2-iWARP and unset for Gen2)
- Applicable interface(s): Gen2, Gen2-iWARP

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.

11.49 MV2_USE_RDMA_FAST_PATH

- Class: Run time
- Default: set
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

Setting this parameter enables `mvapich2` to use adaptive rdma fast path features for Gen2 interface and static rdma fast path features for uDAPL interface.

11.50 MV2_USE_RDMA_ONE_SIDED

- Class: Run time
- Default: set
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

Setting this parameter allows `mvapich2` to use optimized one sided implementation based RDMA operations.

11.51 MV2_USE_RING_STARTUP

- Class: Run time
- Default: set
- Applicable interface(s): Gen2

Setting this parameter enables `mvapich2` to use ring based startup.

11.52 MV2_USE_SHARED_MEM

- Class: Run time
- Default: set
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

Use shared memory for intra-node communication.

11.53 MV2_USE_SHMEM_ALLREDUCE

- Class: Run Time
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL, VAPI

This parameter can be used to turn off shared memory based MPI_Allreduce for Gen2 over IBA by setting this to 0.

11.54 MV2_USE_SHMEM_BARRIER

- Class: Run Time
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL, VAPI

This parameter can be used to turn off shared memory based MPI_Barrier for Gen2 over IBA by setting this to 0.

11.55 MV2_USE_SHMEM_BCAST

- Class: Run Time
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

This parameter can be used to turn off shared memory based MPI_Bcast for Gen2 over IBA by setting this to 0.

11.56 MV2_USE_SHMEM_COLL

- Class: Run time
- Default: set
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

Use shared memory for collective communication. Set this to 0 for disabling shared memory collectives.

11.57 MV2_USE_SHMEM_REDUCE

- Class: Run Time
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL, VAPI

This parameter can be used to turn off shared memory based MPI_Reduce for Gen2 over IBA by setting this to 0.

11.58 MV2_USE_SRQ

- Class: Run time
- Default: set
- Applicable interface(s): Gen2, Gen2-iWARP

Setting this parameter enables mvapich2 to use shared receive queue.

11.59 MV2_VBUF_POOL_SIZE

- Class: Run time
- Default: 512
- Applicable interface(s): Gen2, Gen2-iWARP

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

11.60 MV2_VBUF_SECONDARY_POOL_SIZE

- Class: Run time
- Default: 128
- Applicable interface(s): Gen2, Gen2-iWARP

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.

11.61 MV2_VBUF_TOTAL_SIZE

- Class: Run time
- Default: Architecture dependent (6 KB for EM64T)
- Applicable interface(s): Gen2, Gen2-iWARP

The size of each vbuf, the basic communication buffer of MVAPICH2.

11.62 SMP_EAGERSIZE

- Class: Run time
- Default: Architecture dependent
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

This parameter defines the switch point from Eager protocol to Rendezvous protocol for intra-node communication. Note that this variable should be set in KBytes.

11.63 SMPI_LENGTH_QUEUE

- Class: Run time
- Default: Architecture dependent
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

This parameter defines the size of shared buffer between every two processes on the same node for transferring messages smaller than or equal to SMP_EAGERSIZE. Note that this variable should be set in KBytes.

11.64 SMP_NUM_SEND_BUFFER

- Class: Run time
- Default: Architecture dependent
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

This parameter defines the number of internal send buffers for sending intra-node messages larger than SMP_EAGERSIZE.

11.65 SMP_SEND_BUF_SIZE

- Class: Compile time
- Default: Architecture dependent
- Applicable interface(s): Gen2, Gen2-iWARP, uDAPL

This parameter defines the packet size when sending intra-node messages larger than SMP_EAGERSIZE.