Advanced Technical Skills (ATS) North America

Running MPI applications on Linux over Infiniband cluster with Open MPI

IBM High Performance Computing
February 2010

Y. Joanna Wong
yjw@us.ibm.com

Open MPI

- Reference: http://www.open-mpi.org (FAQ link)
- Open source project for a world-class MPI-2 implementation with top-level goals:
  - Create a free, open source, peer-reviewed, production-quality complete MPI-2 implementation.
  - Provide extremely high, competitive performance (latency, bandwidth, ...pick your favorite metric).
  - Directly involve the HPC community with external development and feedback (vendors, 3rd party researchers, users, etc.).
  - Provide a stable platform for 3rd party research and commercial development.
  - Help prevent the "forking problem" common to other MPI projects.
  - Support a wide variety of HPC platforms and environments.
- Initially merger between
  - FT-MPI from the University of Tennessee
  - LA-MPI from Los Alamos National Laboratory
  - LAM/MPI from Indiana University
  - with contributions from the PACX-MPI at the University of Stuttgart

Open MPI…

- Three types of frameworks or major core modules
  - OMPI – the MPI layer or code
  - ORTE – the run-time layer or Open Run-Time Environment
  - OPAL – the operating system/platform layer or Open Portable Access Layer
- Current stable release version 1.4.1, released on Jan 15, 2010
- Support for various high-speed interconnect networks can be built by specifying the appropriate header files and libraries with the appropriate –with command line switch for Open MPI configure
- The “ompi_info” command provide details on the Open MPI installation.

e.g. ompi_info | grep “MCA btl” will display the supported point-to-point network

MCA btl: openib (MCA v1.0, API v1.0.1, Component v1.2.8)

MCA btl: self (MCA v1.0, API v1.0.1, Component v1.2.8)

MCA btl: sm (MCA v1.0, API v1.0.1, Component v1.2.8)

MCA btl: tcp (MCA v1.0, API v1.0.1, Component v1.0)

Open MPI : compiling MPI applications

- Open MPI compilers are “wrapper” compiler names that will automatically invoke the back-end command opal_wrapper
- The -showme option to the Open MPI wrapper command will display the corresponding flags that will be necessary to compile and to link

  e.g. “ mpif90 -showme “ will return

  ifort -I/usr/mpi/intel/Open MPI-1.4/include -pthread -I/usr/mpi/intel/Open MPI-1.4/lib64 -L/usr/mpi/intel/Open MPI-1.4/lib64 -lmpi_f90 -lmpi_f77 -lmpi -lopen-rte -lopen-pal -ldl -Wl,--export-dynamic -lnsl –lutil
- The wrapper compiler data files (in $prefix/share/Open MPI) contain the flags that are passed to the underlying compiler. These are customized when Open MPI was configured.

Language	Wrapper compiler name
C	mpicc
C++	mpiCC, mpicxx, mpic++
Fortran 77	mpif77
Fortran 90	mpi90

Open MPI…

- The Open MPI wrapper compiler, preprocessor flags, compiler flags, linker flags and linker library flags can be overridden with environment values

Wrapper compiler	Compiler	Preprocessor flags	Compiler flags
mpicc	OMPI_CC	OMPI_CPPFLAGS	OMPI_CFLAGS
mpic++	OMPI_CXX	OMPI_CPPFLAGS	OMPI_CXXFLAGS
mpiCC	OMPI_CXX	OMPI_CPPFLAGS	OMPI_CXXFLAGS
mpif77	OMPI_F77	OMPI_CPPFLAGS	OMPI_FFLAGS
mpif90	OMPI_FC	OMPI_CPPFLAGS	OMPI_FCFLAGS
Wrapper compiler	**Linker flags**	**Linker library flags**	**Data file**
mpicc	OMPI_LDFLAGS	OMPI_LIBS	mpicc-wrapper-data.txt
mpic++	OMPI_LDFLAGS	OMPI_LIBS	mpic++-wrapper-data.txt
mpiCC	OMPI_LDFLAGS	OMPI_LIBS	mpiCC-wrapper-data.txt
mpif77	OMPI_LDFLAGS	OMPI_LIBS	mpif77-wrapper-data.txt
mpif90	OMPI_LDFLAGS	OMPI_LIBS	mpif90-wrapper-data.txt

Compiling and running Open MPI applications

- The PATH and LD_LIBRARY_PATH should be set up to include the underlying compilers for the wrapper compilers.
- Open MPI requires that its executables and libraries can be located on every node.
  - Setup up the PATH and LD_LIBRARY_PATH in the shell startup script to include locations for Open MPI executables and libraries

    export PATH=$MPI_ROOT/bin:$PATH

    export LD_LIBRARY_PATH=$MPI_ROOT/lib:$PATH

    where $MPI_ROOT is the directory where Open MPI is installed
  - Include the –prefix=$MPI_ROOT option to the Open MPI runtime environment command (orterun, mpirun, mpiexec) :

    mpirun –prefix=$MPI_ROOT –n 4 –hostfile *hfile app*
  - Use the absolute pathname to mpirun, which is equivalent to using the –prefix option:

    $MPI_ROOT/bin/mpirun –n 4 –hostfile *hfile app*
  - When using resource managers to launch jobs (e.g Torque), the entire local environment (including PATH and LD_LIBRARY_PATH) is copied to remote nodes.

Open MPI…

- mpirun and mpiexec are symbolic links to the common back-end launcher command orterun
  - Provided for compatibility with run scripts for other MPI library
- Notable options for mpirun (can with 1 or 2 dashes)

  --hostfile or --machinefile : specify a hostfile for the launcher

  --host : list of hosts to invoke processes on

  --n or –np : number of processes to run

  --byslot : allocate/map processes round-robin by slots

  --bynode : allocate/map processes round-robin by node in host list

  --mca <arg0> <arg1> : set MCA parameters

  --wdir : working directory to start the applications
- The default scheduling policy can also be set using the MCA parameter rmaps_base_schedule_policy to “slot” or to “node”

Open MPI…

- If the maximum slot count is exhausted on all node where there are still processes to be scheduled, the Open MPI job will abort.
- A typical hostfile could have following entries:

  node001 slots=2 max-slots=8

  node002 slots=2 max-slots=8
- Scheduling –byslot
  - Processes will be scheduled on a node until all the default slots (2 for node001 above) are exhausted before proceeding to next node.

    For example hostfile, processes are scheduled :

    node001, node001, node002, node002, node001, node001, node002, node002, node001,node001,node002, node002, node001, node001, node002, node002
- Scheduling –bynode
  - Processes will be scheduled on each node (1 process each) in a round-robin fashion until all processes have been scheduled

    For example hostfile, processes are scheduled :

    node001, node002, node001, node002, node001, node002, node001, node001, node002, node001, node002, node001, node002, node001, node001, noe001

Open MPI…

- Can oversubscribe node (run more processes than cores) if max_slots for the node is not stated in the hostfile:

  e.g. can run any number of processes on node001 if hostfile has:

  node001 slots=4

- The message passing progression engine runs in 2 modes: aggressive or degraded
  - It is critical that the number of slots specified is not more than available number of processes so that Open MPI is aware that the node is running oversubscribed
  - Can be set with MCA parameter mpi_yield_when_idle:

    set to 0 : aggressive

    not set to 0 : degraded

Open MPI…

- Modular Component Architecture (MCA)
  - The backbone of Open MPI's funtionality
  - An MPI implementation is created at run-time by assembling a series of frameworks, components, and modules
    - A framework manages zero or more components at run time and is targeted a specific tasks (e.g. provide MPI collective operation)
    - A component is an implementation of a framework's interface, assembled into code base at run-time and/or compile-time
    - An module is an instance of a component.
    - E.g. to run on a node with multiple ethernet NICs, the application will contain 1 TCP MPI point-to-point component, but 2 TCP point-to-point modules

Open MPI…

- **OMPI frameworks**
  - allocator: Memory allocator
  - bml: BTL management layer
  - btl: MPI point-to-point Byte Transfer Layer, used for MPI point-to-point messages on some types of networks
  - coll: MPI collective algorithms
  - crcp: Checkpoint/restart coordination protocol
  - dpm: MPI-2 dynamic process management
  - io: MPI-2 I/O
  - mpool: Memory pooling
  - mtl: Matching transport layer, used for MPI point-to-point messages MPI-2 one-sided communications
  - pml: MPI point-to-point management layer
  - pubsub: MPI-2 publish/subscribe management
  - rcache: Memory registration cache
  - topo: MPI topology routines

Open MPI…

- **ORTE frameworks**
  - errmgr: RTE error manager
  - ess: RTE environment-specfic services
  - filem: Remote file management
  - grpcomm: RTE group communications
  - iof: I/O forwarding
  - odls: OpenRTE daemon local launch subsystem
  - oob: Out of band messaging
  - plm: Process lifecycle management
  - ras: Resource allocation system
  - rmaps: Resource mapping system
  - rml: RTE message layer
  - routed: Routing table for the RML
  - snapc: Snapshot coordination

Open MPI…

- **OPAL frameworks**
  - backtrace: Debugging call stack backtrace support
  - carto: Cartography (host/network mapping) support
  - crs: Checkpoint and restart service
  - installdirs: Installation directory relocation services
  - maffinity: Memory affinity
  - memchecker: Run-time memory checking
  - memcpy: Memopy copy support
  - memory: Memory management hooks
  - paffinity: Processor affinity
  - timer: High-resolution timers

Open MPI…

- Set MCA parameter, in precedence order:
  - set on the command line with --mca <param_name> <value>

    e.g. mpirun –mca mpi_leave_pinned 0 –np 4 a.out
    - Value with multiple words will need to be enclosed with quotes
  - Value set with environment values OMPI_MCA_<param_name>

    e.g. export OMPI_MCA_mpi_leave_pinned=0

    mpirun –np 4 a.out

  -- (v1.3 and higher) Use aggregate MCA (AMCA) parameter files that contain the MCA parameter key/value pairs, one per line, with format:

  <param_name> = <value>
    - The location of the AMCA parameter files can be resolved by the mpirun option –am fileA:fileB where the files are loaded in priority order, i.e., parameter values set in fileA has precedence over the value of same key in fileB
  - Use MCA parameter file that contain the MCA parameter key/value pairs, one per line.
    - The location of the MCA parameter files are given by the MCA parameter mca_param_files – a colon-delimted path of files, where files to the right are higher precendence
    - Two files are searched (in order) by default:
      - $HOME/.Open MPI/mca-params.conf : user-supplied set of values have higher precendence
      - $MPI_ROOT/etc/Open MPI-mca-params.conf : system-supplied set of values

Open MPI…

- **mpirun –mca setting:**
  - Use OpenFabrics/IB network for MPI communications

    --mca btl openib,self
  - Use OpenFabrics/IB network for internode MPI communications and shared memory for intranode MPI communications

    --mca btl openib,self,sm
  - No need to disable tcp BTL when using high-speed network (such as Infiniband): Open MPI assumes that there is a TCP network. The tcp BTL component will automatically deactivate itself for MPI communications. The Open MPI may still use TCP for setup and teardown.
- mpirun does not need to specify –hostfile, --host, or –np options when running jobs under Torque.
- (version 1.3) faster startup for large cluster using tree-based algorithm

  --mca routed binomial

MCA parameters

- coll_sm_info_num_procs
  - Number of processes to use for the calculation of the shared_mem_size MCA information parameter (must be => 2)
- btl_openib_free_list_max
  - Maximum size of free lists (-1 = infinite, otherwise must be >= 0)
- mpi_leave_pinned
  - Whether to use the "leave pinned" protocol or not. Enabling this setting can help bandwidth performance when repeatedly sending and receiving large messages with the same buffers over RDMA-based networks (0 = do not use "leave pinned" protocol, 1 = use "leave pinned" protocol, -1 = allow network to choose at runtime).