MPI for Cray XE/XK Systems & Recent Enhancements

Heidi Poxon
Technical Lead
Programming Environment
Cray Inc.

March 2016

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA, and YARCDATA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.

Topics

- Cray MPI overview
- Development focus / recent enhancements
- Overlapping computation and communication
- Memory footprint
- MPI I/O statistics
- MPI Tuning controls for Cray systems

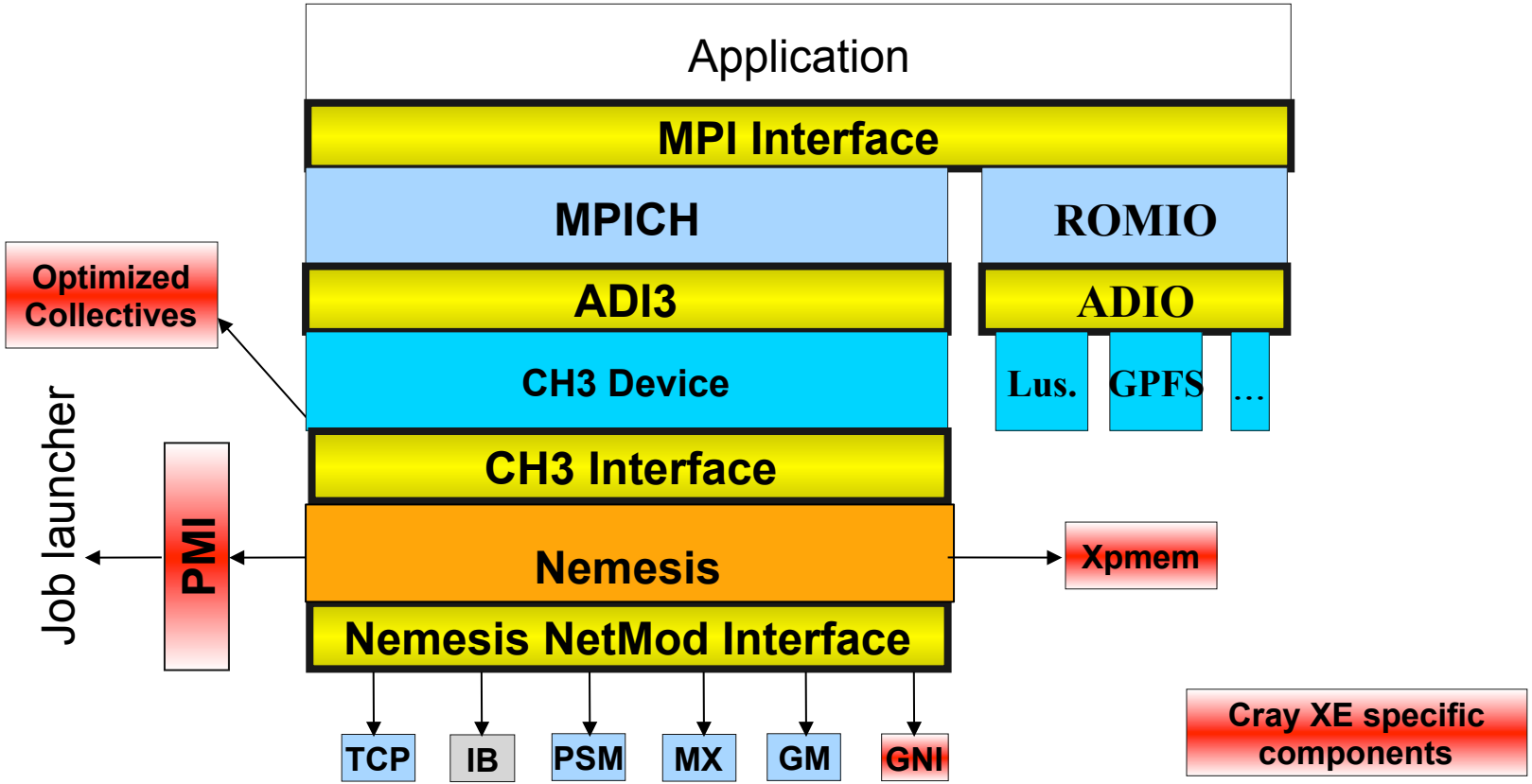
Cray MPI

- **Implementation based on MPICH3 from ANL**
  - ANL does base MPI standard support, we add new functionality, improve performance both on-node, and all ranges of scale including at very high scale
- **Full MPI-3.1 support (Dec 2015) with the exception of**
  - MPI-2 Dynamic process management (MPI_Comm_spawn)
- **MPI Forum active participant**
- **Participated in the MPICH ABI Consortium**
  - ANL MPICH, Intel MPI, IBM PE MPI and Cray MPI

Development Focus Areas

- Minimize communication latency, maximize communication bandwidth
- Improve support for asynchronous communication (communication/computation overlap)
- Architecture-specific solutions to optimize communication performance
- New tools and features to help users understand application performance bottlenecks

MPICH/Cray MPI Layout



Gemini Features Used by Cray MPI

- **FMA (Fast Memory Access)**
  - Used for small messages
  - Very low overhead ➔ good latency
- **DMA offload engine (BTE or Block Transfer Engine)**
  - Used for larger messages
  - All ranks on node share BTE resources (4 virtual channels / node)
  - Can be initiated in user mode
  - Once initiated, BTE transfers proceed without processor intervention
    - Best means to overlap communication with computation

Asynchronous Progress Engine

- Used to improve communication/computation overlap
- Used for non-blocking pt2pt and collective MPI calls
- Each MPI rank starts a “helper thread” during MPI_Init
- Helper threads progress the MPI state engine for both Send and Recv while application is computing
- Only effective if used with core specialization to reserve a core/node for the helper threads or using the Intel hyper-threads
- Must set the following to enable Asynchronous Progress Threads:
  - export MPICH_NEMESIS_ASYNC_PROGRESS=(SC or MC)
  - export MPICH_MAX_THREAD_SAFETY=multiple
- 10% or more performance improvements with some apps

Fine-Grained Multi-threading for MPI

- Optimized support for programs that perform MPI operations within threaded regions
- Only MPI point-to-point operations optimized at this time
- Default MPI library uses a global lock
  - A single global_mutex is used for all MPI calls
- Separate MT MPI library uses per-object (fine-grained) locks
  - The global_mutex is still used, but critical sections are much smaller
  - Additional small locks have also been added
- Must link with a separate version of Cray MPI library
  - Use the compiler driver option: -craympich-mt
- To use:
  - `> cc -craympich-mt -o mpi_mt_test.x  mpi_mt_test.c`
  - `> export MPICH_MAX_THREAD_SAFETY=multiple`

Cray MPI-3 Non-blocking Collectives

- Allows overlap with computation during collective operations
- All MPI collectives have MPI_I<name> versions (i.e. MPI_Ibcast)
- MPI Asynchronous Progress Engine Feature is needed to give the best overlap
- To enable, use the following environment variables
  - MPICH_NEMESIS_ASYNC_PROGRESS=(SC or MC)
    (setting to 1 is the same as setting it to SC on Gemini, MC on Aries)
  - MPICH_MAX_THREAD_SAFETY=multiple
- Best to run using:
  - Core-specialization (aprun –r) on Gemini, or if no hyperthreads available.
  - Unused Intel hyperthread cores (aprun –j) on Aries

Minimized MPI Memory Footprint

- **Implemented a Dynamic Virtual Channel (VC) Feature**
  - Internal VC structures only allocated when one rank makes direct contact with another rank
  - Prior MPT versions allocated VCs statically for all ranks in job during MPI_Init
  - Enabled by default starting with MPT 7.2.3
- **Implemented special optimizations for MPI_Alltoall and MPI_Alltoallv that don't require use of VC structures**
- **Significantly reduces MPI footprint for many HPC apps (nearest neighbor communication plus global collectives)**

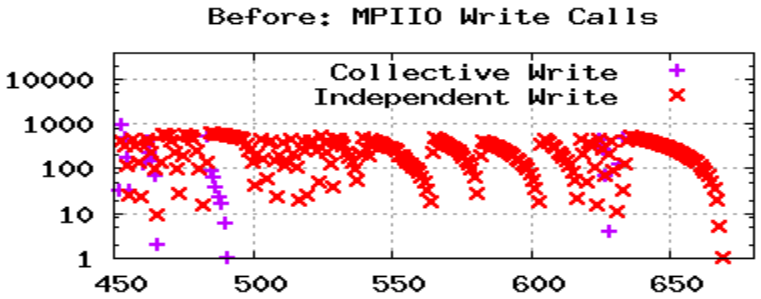
MPI I/O File Access Pattern Statistics

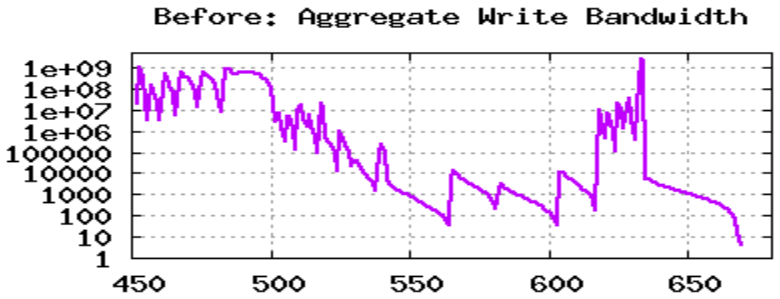
- When setting MPICH_MPIIO_STATS=1, a summary of file write and read access patterns are written by rank 0 to stderr
- When setting MPICH_MPIIO_STATS=2, a set of data files (one per rank) are written which can be summarized with the supplied cray_mpiio_summary script
- The "Optimizing MPI I/O" white paper describes how to interpret the data and makes suggestions on how to improve your application.
  - Available on docs.cray.com under Knowledge Base

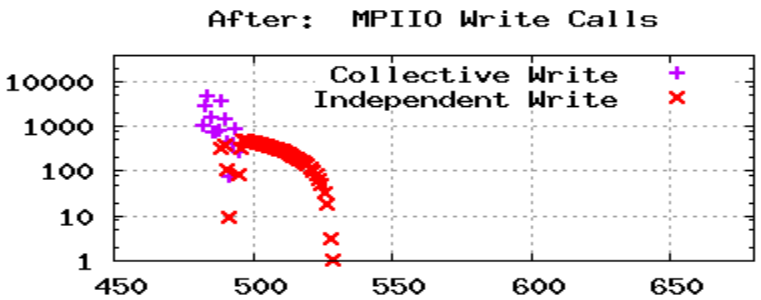
MPI I/O File Access Pattern Statistics (2)

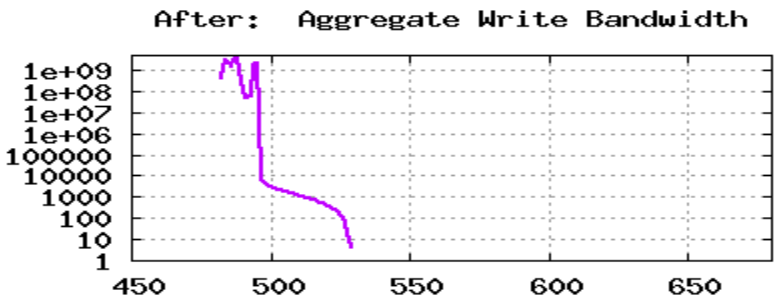
Timeline of MPI-I/O statistics. Many different variables tracked

```
export MPICH_MPIIO_STATS=2
```









MPI Tuning Controls for Cray Systems with Gemini

MPICH_RANK_REORDER_METHOD

- Vary your rank placement to optimize communication
- Can be a quick, low-hassle way to improve performance
- Use CrayPat to produce a specific MPICH_RANK_ORDER file to maximize intra-node communication
- Or, use perftools grid_order command with your application's grid dimensions to layout MPI ranks in alignment with data grid
- To use:
  - name your custom rank order file: MPICH_RANK_ORDER
  - export MPICH_RANK_REORDER_METHOD=3

Use HUGEPAGES

Linking and running with hugepages can offer a significant performance improvement for many MPI communication sequences, including MPI collectives and basic MPI_Send/MPI_Recv calls.

- **To use HUGEPAGES:**
  - **`module load craype-hugepages8M`** (many sizes supported)
  - *<< compile your app >>*
  - **`module load craype-hugepages8M`**
  - *<< run your app >>*

MPICH_USE_DMAPP_COLL / MPICH_RMA_OVER_DMAPP

- **Enables highly optimized algorithms which may result in significant performance gains**
- **Not enabled by default to avoid disadvantages**
  - May reduce resources MPICH has available (share with DMAPP)
  - DMAPP does not handle transient network errors
- **Supported DMAPP-optimized functions:**
  - MPI_Allreduce
  - MPI_Bcast
  - MPI_Barrier
  - MPI_Put / MPI_Get / MPI_Accumulate
- **To use (link with libdmapp):**
  - Collective use: export MPICH_USE_DMAPP_COLL=1
  - RMA one-sided use: export MPICH_RMA_OVER_DMAPP=1

Tune Inter-node Traffic on Gemini

- **Most significant env variables to play with:**
  - **MPICH_GNI_MAX_VSHORT_MSG_SIZE**
    - Controls max message size for E0 mailbox path (Default: varies)
  - **MPICH_GNI_MAX_EAGER_MSG_SIZE**
    - Controls max message size for E1 Eager Path (Default: 8K bytes)
  - **MPICH_GNI_NUM_BUFS**
    - Controls number of 32KB internal buffers for E1 path (Default: 64)
  - **MPICH_GNI_NDREG_MAXSIZE**
    - Controls max message size for R0 Rendezvous Path (Default: 512K bytes)
  - **MPICH_GNI_RDMA_THRESHOLD**
    - Controls threshold for switching to BTE from FMA (Default: 1K bytes)
  - See the MPI man page for further details

MPICH_NEMESIS_ASYNC_PROGRESS

- **Maximize overlap of communication with computation**
- **Enable asynchronous progress engine**
  - Launches additional thread per MPI process to help progress communication in the background
- **Consider trying this if all of these apply to your application:**
  - App uses non-blocking MPI communication (MPI_Isend/MPI_Irecv or non-blocking collectives) with medium-large messages
  - There is computation work between MPI communication sequences
  - Hyperthreads are available on each node (not in use by your application)
- **To use:**
  - export MPICH_MAX_THREAD_SAFETY=multiple
  - export MPICH_NEMESIS_ASYNC_PROGRESS=SC
  - Use aprun –r1 option

Specific Collective Algorithm Tuning

- **Different algorithms may be used for different message sizes in collectives (e.g.)**
  - Algorithm A might be used for Alltoall for messages < 1K.
  - Algorithm B might be used for messages >= 1K.
- **To optimize a collective, you can modify the cutoff points when different algorithms are used. This may improve performance.**
- **MPICH_ALLTOALL_SHORT_MSG**
- **MPICH_ALLGATHER_VSHORT_MSG**
- **MPICH_ALLGATHERV_VSHORT_MSG**
- **MPICH_GATHERV_SHORT_MSG**
- **MPICH_SCATTERV_SHORT_MSG**
- **See the MPI man page for further details**


CRAY
COMPUTE | STORE | ANALYZE
Thank You