Tuning the MPI Runtime Environment and Best Practices for Message Passing on Hopper

NERSC Users Group Workshop
February 2012

Howard Pritchard
howardp@cray.com

Outline

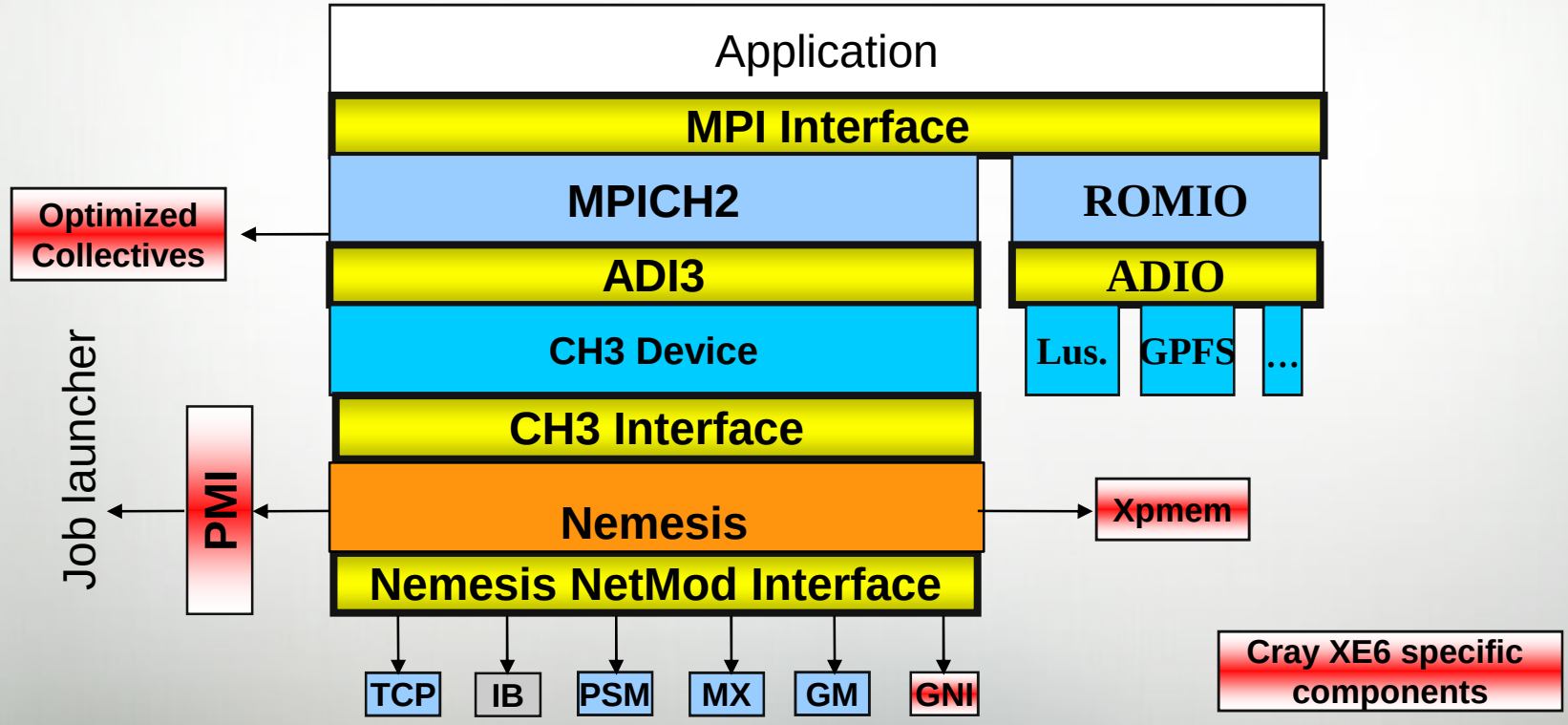
- Brief overview of Gemini NIC
- MPICH2 software stack overview
- Point-to-point messaging
  - Inter-node messaging
  - Intra-node messaging
- Collectives
- Additional Performance Tuning Suggestions
- What Are We Working on Now?

Gemini NIC Resources

- FMA (Fast Memory Access)
  - Used for small messages
  - Called directly from user mode
  - Very low overhead ➔ good latency
- BTE (Block Transfer Engine)
  - Used for larger messages
  - Processed via the OS (no direct user-mode access)
  - All ranks on node share BTE resources (4 virtual channels/ node)
  - Higher overhead to initiate transfer
  - Once initiated, BTE transfers proceed without processor intervention
- I/O MMU
  - Limited support for registering 4KB pages
  - Optimized for large pages (2MB and larger)

MPICH2 on XE Overview

- Initial MPT Gemini Release - MPT 5.0 (June 2010)
  - Uses the ANL CH3/Nemesis device
  - Cray wrote a GNI Network Module (NetMod) that plugs into Nemesis
  - GNI carries forward to future Cray interconnects



Inter-Node Messaging

Inter-node Messaging

- Eager Message Protocol
  - E0 and E1 Paths
- Rendezvous Message Protocol
  - R0 and R1 Paths
- MPI environment variables that alter those paths

Day in the Life of an MPI Message

- Four Main Pathways through the MPICH2 GNI NetMod
  - Two EAGER paths (E0 and E1)
  - Two RENDEZVOUS (aka LMT) paths (R0 and R1)
- Selected Pathway is Based (generally) on Message Size

E0	E1	R0	R1
0 512	1K 2K 4K 8K	16K 32K 64K 128K 256K 512K	1MB ++

- MPI env variables affecting the pathway
  - **MPICH_GNI_MAX_VSHORT_MSG_SIZE**
    - Controls max size for E0 Path (Default varies with job size: 216-984 bytes)**
  - **MPICH_GNI_MAX_EAGER_MSG_SIZE**
    - Controls max message size for E1 Path (Default is 8K bytes)
  - **MPICH_GNI_NDREG_MAXSIZE**
    - Controls max message size for R0 Path (Default is 4M bytes) **
  - **MPICH_GNI_LMT_PATH=disabled**
    - Can be used to disable the entire Rendezvous (LMT) Path

EAGER Message Protocol

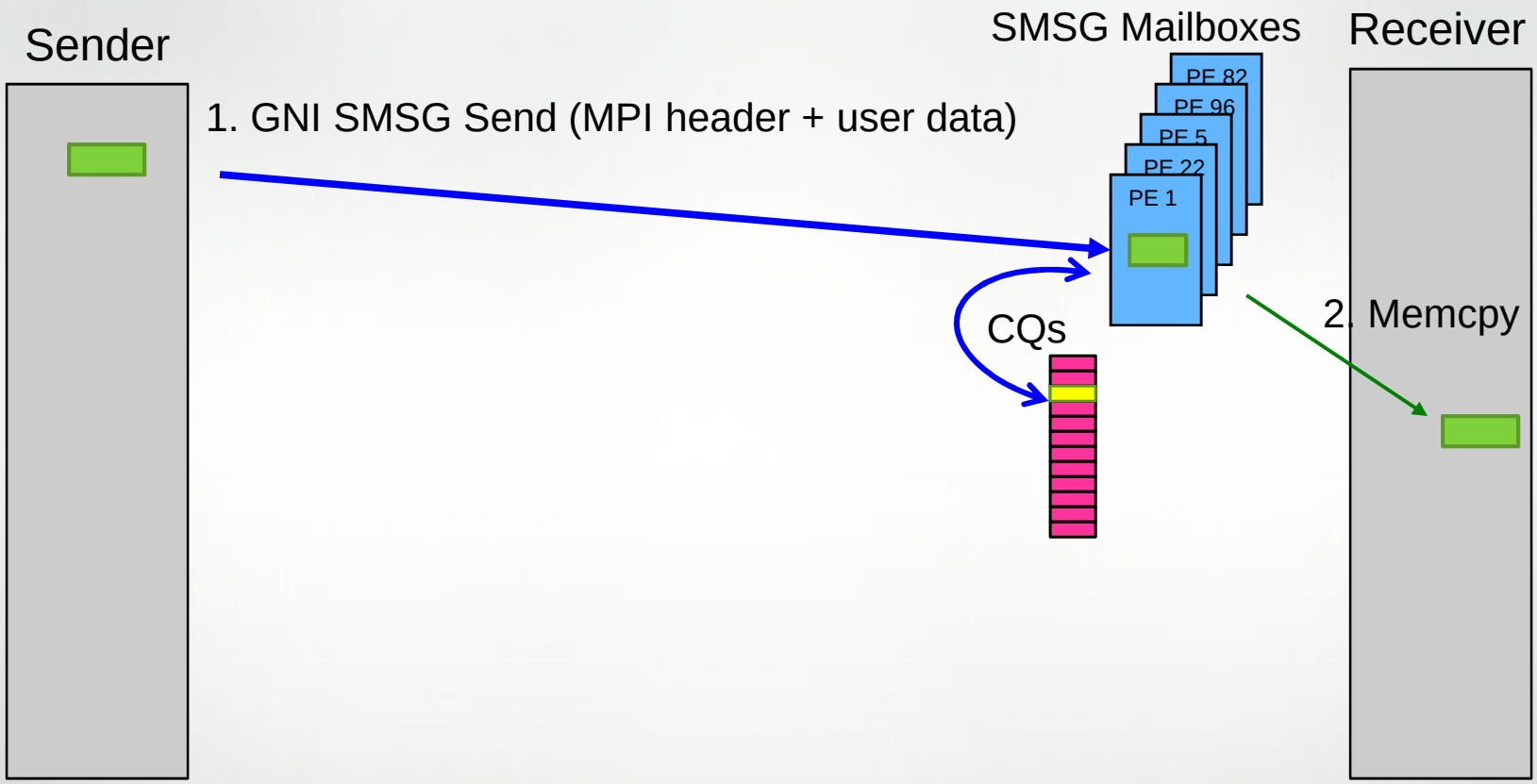
- Sender's data made available to the receiver whether or not a matching receive has been posted.
- Two EAGER Pathways
  - **E0** – small messages that fit into GNI SMSG Mailbox
    - Default mailbox size varies with number of ranks in the job

Ranks in Job	Max user data (MPT 5.3)	MPT 5.4 and later
< = 512 ranks	984 bytes	8152 bytes
> 512 and <= 1024	984 bytes	2008 bytes
> 1024 and < 16384	472 bytes	472 bytes
> 16384 ranks	216 bytes	216 bytes

    - Use **MPICH_GNI_MAX_VSHORT_MSG_SIZE** to adjust size
  - **E1** – too big for SMSG Mailbox, but small enough to still go EAGER
    - Use **MPICH_GNI_MAX_EAGER_MSG_SIZE** to adjust size
    - Requires extra copies

Day in the Life of Message type E0

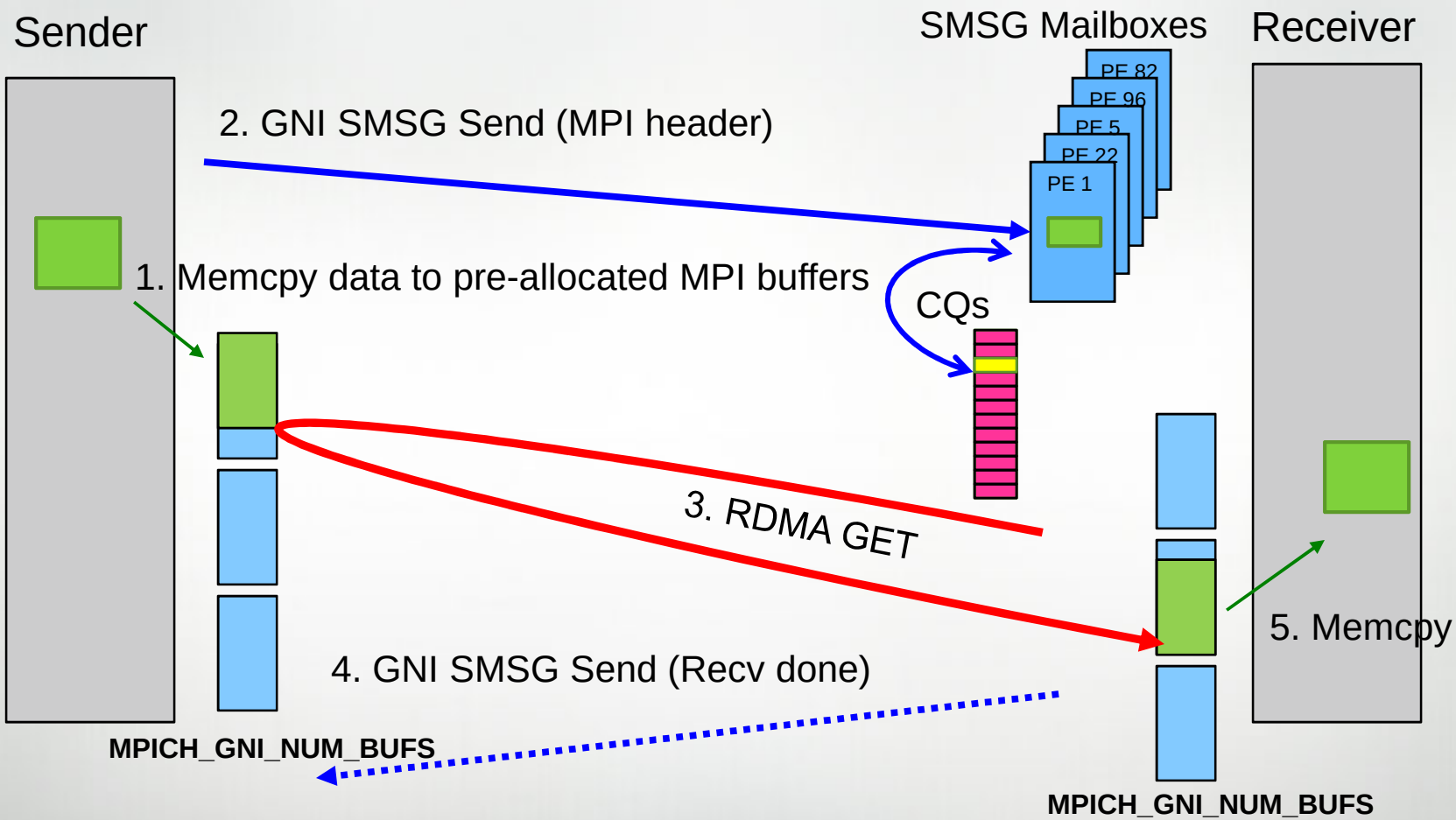
EAGER messages that fit in the GNI SMSG Mailbox



- GNI SMSG Mailbox size changes with the number of ranks in the job
- If user data is 16 bytes or less, it is copied into the MPI header
- Mailboxes use large pages by default (even if app isn't using them itself)

Day in the Life of Message type E1

EAGER messages that don't fit in the GNI SMSG Mailbox



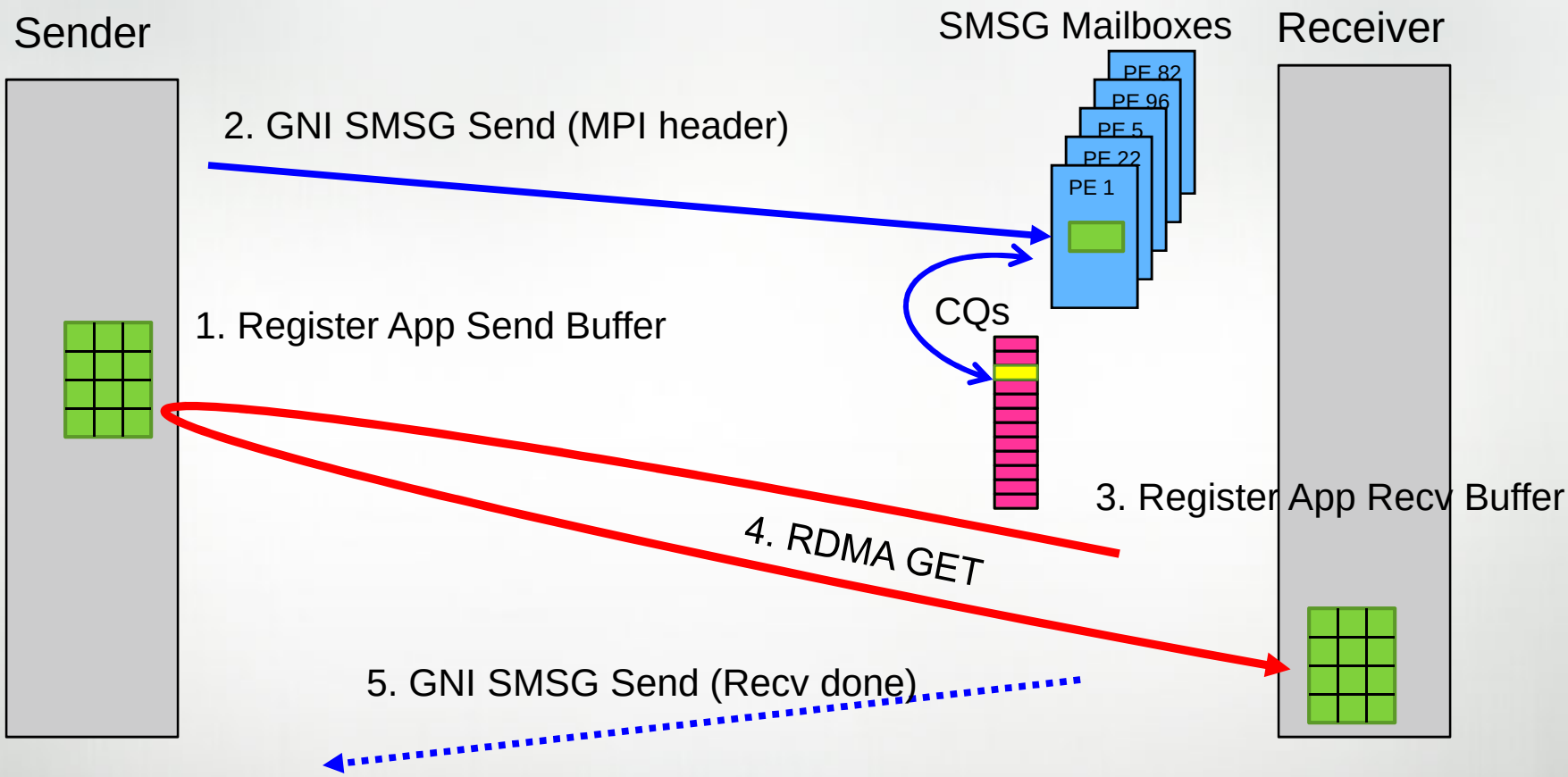
- User data is copied into internal MPI buffers on both send and receive side
- Default **MPICH_GNI_NUM_BUFS** is 64 (each buffer is 32K)
- Internal MPI buffers use large pages

RENDEZVOUS Message Protocol

- Data is transferred after receiver has posted matching receive for a previously initiated send
- Two RENDEZVOUS Pathways
  - R0 – **RDMA GET** method
    - By default, used for messages between 8K and 4 MB
    - Use **MPICH_GNI_MAX_EAGER_MSG_SIZE** to adjust starting point
    - Use **MPICH_GNI_NDREG_MAXSIZE** to adjust ending point
  - R1 – Pipelined **RDMA PUT** method
    - By default, used for messages greater than 4 MB
    - Use **MPICH_GNI_NDREG_MAXSIZE** to adjust starting point
    - Highest bandwidth, little chance for communication/computation overlap without using async progress threads

Day in the Life of Message type R0

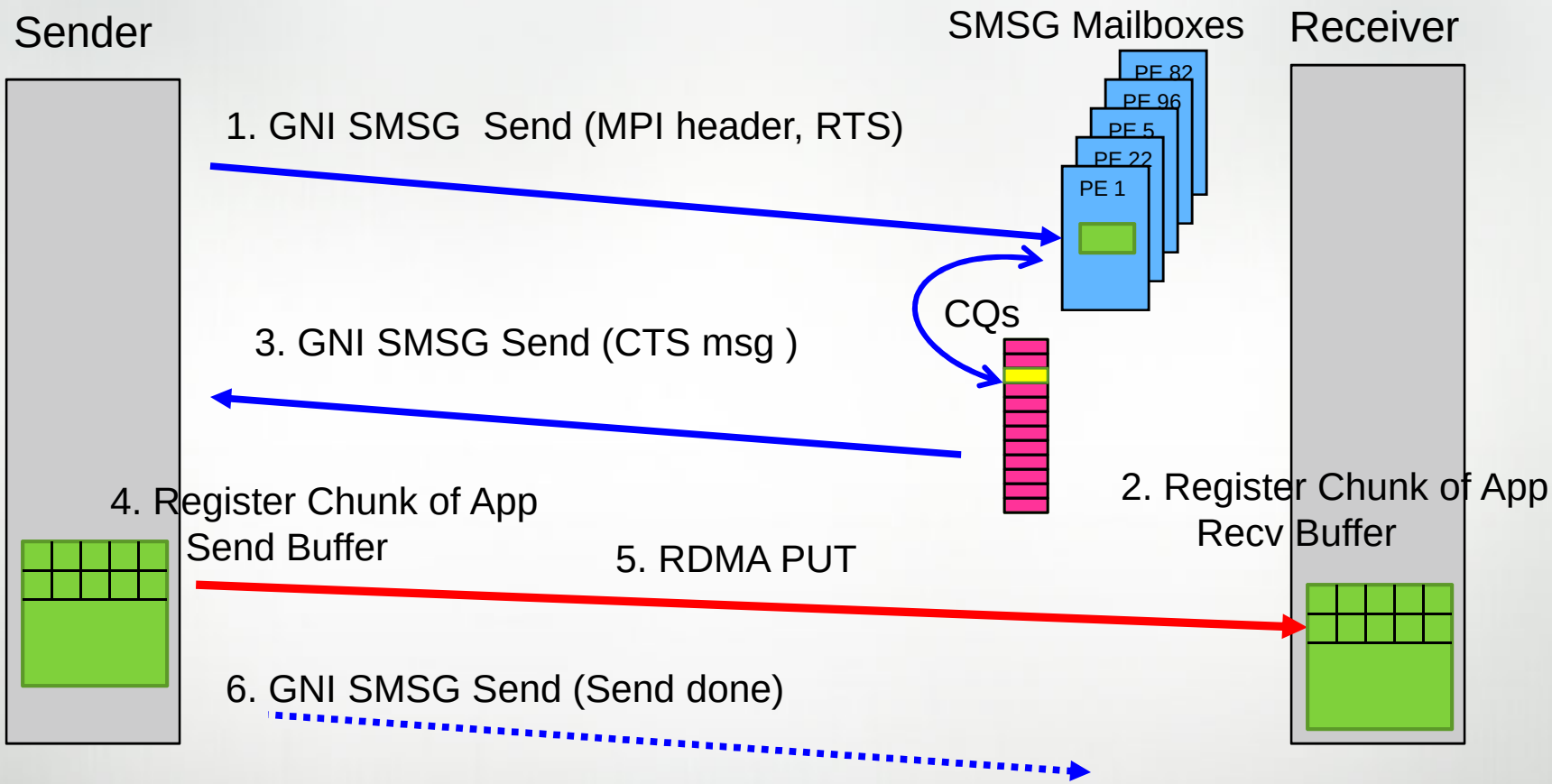
Rendezvous messages using RDMA Get



- No extra data copies
- Sensitive to relative alignment of send/recv buffers

Day in the Life of Message type R1

Rendezvous messages using RDMA Put



- ***Repeat steps 2-6 until all sender data is transferred***
- Chunksize is MPI_GNI_MAX_NDREG_SIZE (default of 4MB, starting with MPT 5.4)

Inter-Node Messaging Env. Variable Summary

- MPICH_GNI_DYNAMIC_CONN

  By default, mailbox connections are established when a rank first sends a message to another rank. This optimizes memory usage for mailboxes. This feature can be disabled by setting this environment variable to *disabled.*
  For applications with all-to-all style messaging patterns, performance may be improved by setting this env. variable to *disabled*.

- MPICH_GNI_LMT_PATH

  Controls whether or not the LMT (R0,R1 paths) are used for the rendezvous protocol. Setting this env. variable to *disabled* disables the use of the LMT path for long messages. All messages are sent through the E0, E1 paths. Mainly useful for debugging.

- MPICH_GNI_VSHORT_MSG_SIZE

  Controls the maximum message size that can be sent using the E0 path. Upper limit of 8192 bytes (minus overhead for MPICH internal message header).

Inter-Node Messaging Env. Variable Summary(cont.)

- MPICH_GNI_MAX_EAGER_MSG_SIZE
  Controls the threshold for switching from the eager (E0,E1) protocol to the rendezvous (R0,R1) protocol. Default is messages larger then X bytes use rendezvous protocol*.
- MPICH_GNI_NDREG_LAZYMEM
  By default, memory deregistration of application message buffers is treated lazily. This is done to reduce the overhead of registering/deregistering memory with the Gemini I/O MMU. Sometimes issues are encountered with this technique. Setting this env. variable to *disabled* results in MPICH2 not using this lazy memory deregistration. <u>If this variable has to be set for an application to run correctly, please file a bug report</u>.
- MPICH_GNI_RDMA_THRESHOLD
  Controls the threshold for switching from using the FMA hardware to the RDMA engine (BTE) for data transfers. This impacts the E1, R0, and R1 paths. The default data transfer size for switching to BTE is 1024 bytes.

*Not all send protocols use the R0,R1 path. MPI_Rsend ,for example, currently does not.

Inter-Node Messaging Env. Variable Summary(cont.)

- MPICH_GNI_NDREG_MAXSIZE

  Controls the maximum block size used for chunking up messages for the R1 protocol. Messages up to 1 block size in length may use the R0 path. The R1 path may still be used for such messages sizes when memory registration resources are constrained or alignment of the send buffer is not suitable for RDMA get based transfers.

Intra-Node Messaging

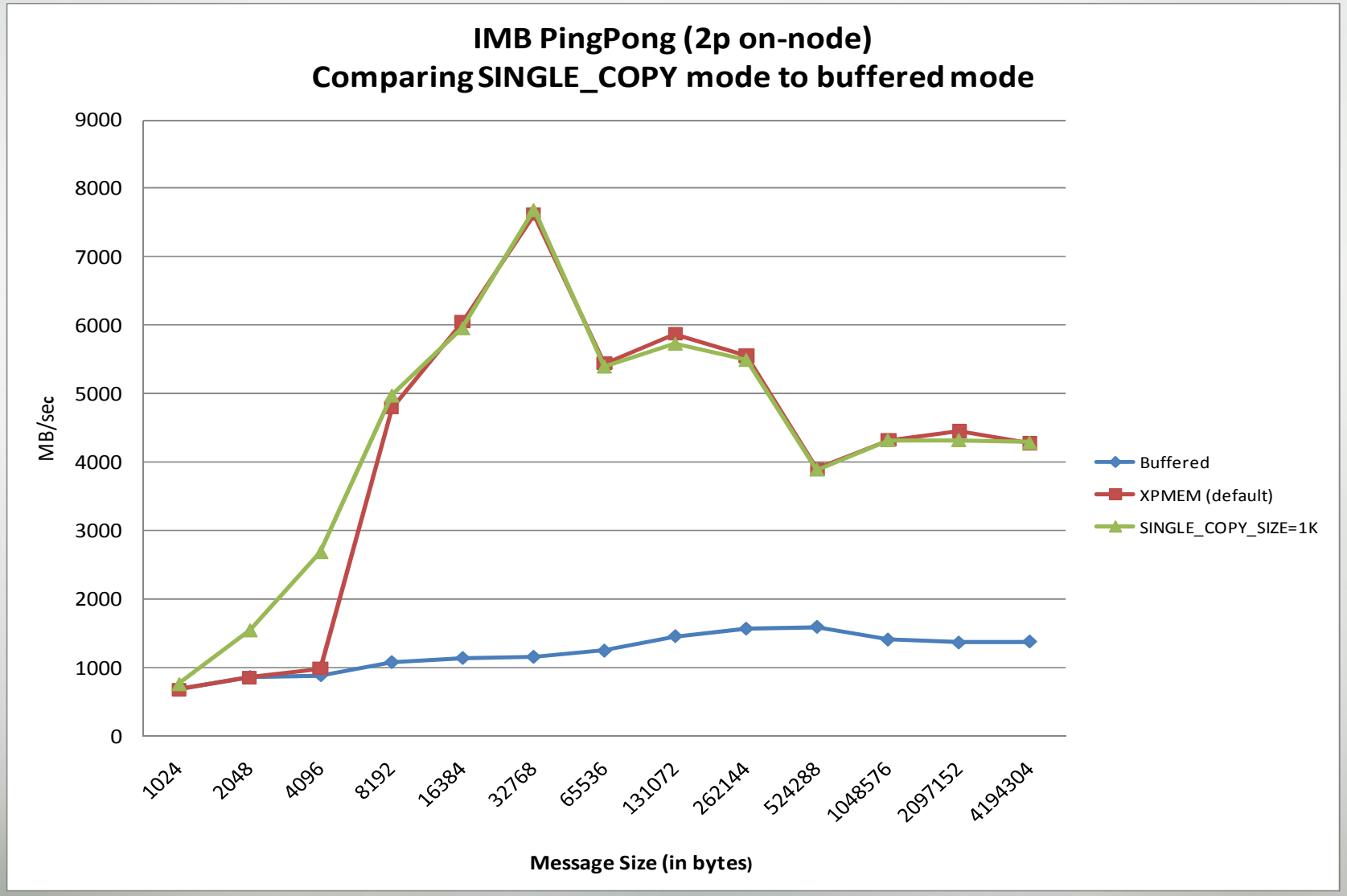
Intra-Node Messaging

- Use MPICH2 Nemesis infrastructure
  - Messages smaller than 8192 bytes use an eager send protocol using shared memory queues.
  - Longer messages sent using a rendezvous protocol. A get based mechanism is used, analogous to R0 for inter-node messages.
- Cray has chosen XPMEM based mechanism for rendezvous protocol. XPMEM is based off open-sourced SGI XPMEM.
- Use **MPICH_SMP_SINGLE_COPY_SIZE** to control threshold for switching from eager to rendezvous protocol

XPMEM Description

- Allows one process to *export* its process address so that other processes with sufficient privilege can attach to sections of this address space (not ptrace based).
- Attaching processes can directly load from/store to these attached sections of the exporter's address space – a kind of dynamic shared memory.
- No system calls involved once a memory segment is attached to a process.
- Uses Linux mmu_notifier infrastructure to remove some restrictions present with the original SGI XPMEM implementation.
- Network independent. Does not rely on presence of a network adaptor/driver.

Intra-Node Messaging using XPMEM



Intra-Node Messaging Env. Variable Summary

- MPICH_SMP_SINGLE_COPY_SIZE
  Specifies minimum message size that can be sent using the XPMEM single copy method.
- MPICH_SMP_SINGLE_COPY_OFF
  If set to any value, disables the use of the XPMEM single copy method. Overrides any MPICH_SMP_SINGLE_COPY_SIZE setting. May be useful for diagnosing/debugging problems.

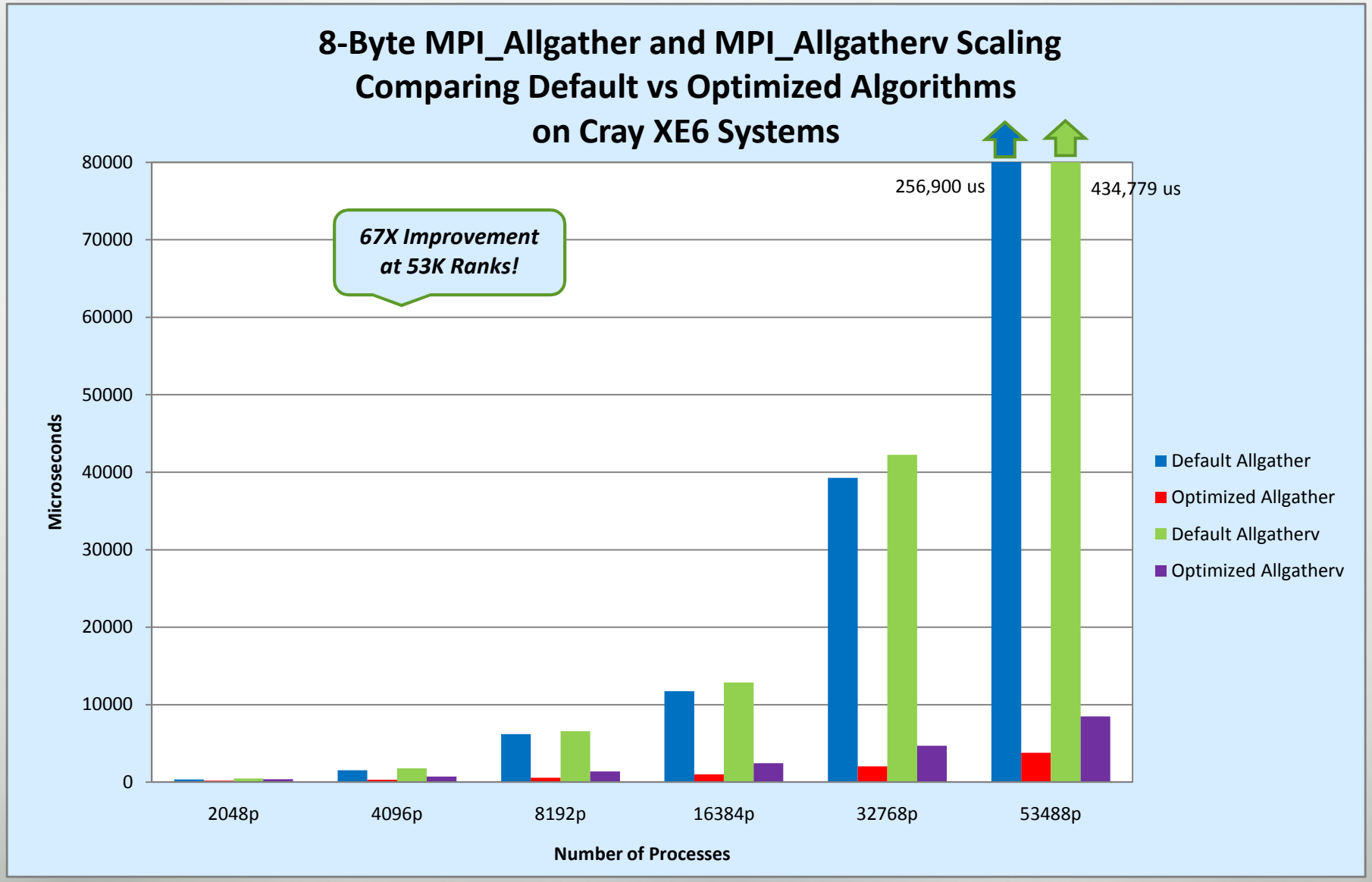
Collectives

MPI Collectives Optimized for Cray XE

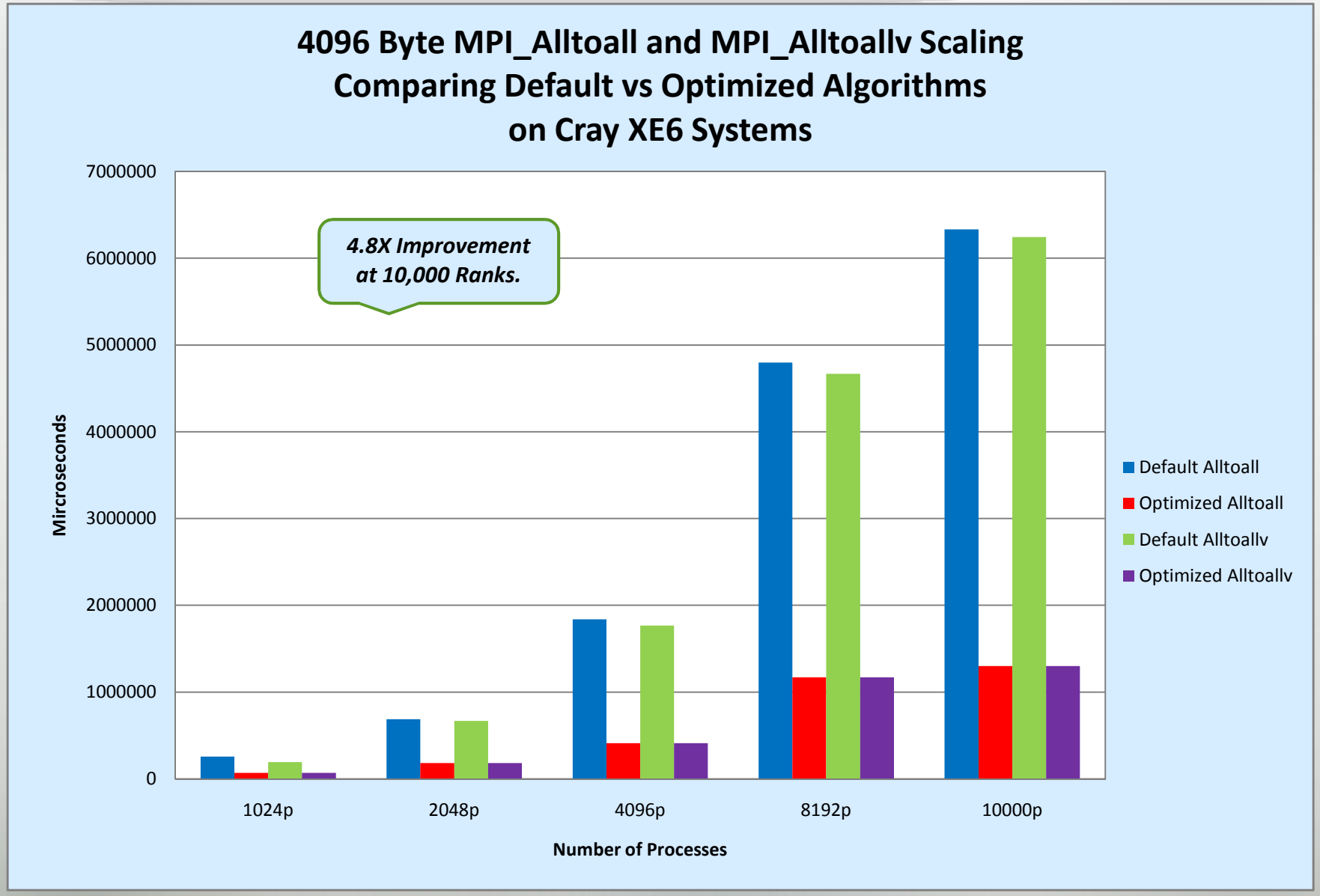
- MPI_Bcast
  - Switched to an SMP-aware tree algorithm for all data sizes
- MPI_Alltoall
  - Randomized send_to / receive_from list
- MPI_Alltoallv
  - Randomized send_to / receive_from list
  - Throttle for number of outstanding sends/receives
  - For small messages (< 256 bytes) or if processes not packed on node: export **MPICH_ALLTOALLV_THROTTLE**=2 (or 3)
- MPI_Allgather / MPI_Allgatherv / MPI_Gatherv
  - Optimizations for small transfer sizes at scale
- MPI_Allreduce
  - Option to use DMAPP reduction

Note in general these optimizations apply to collective operations over intra-communicators only.

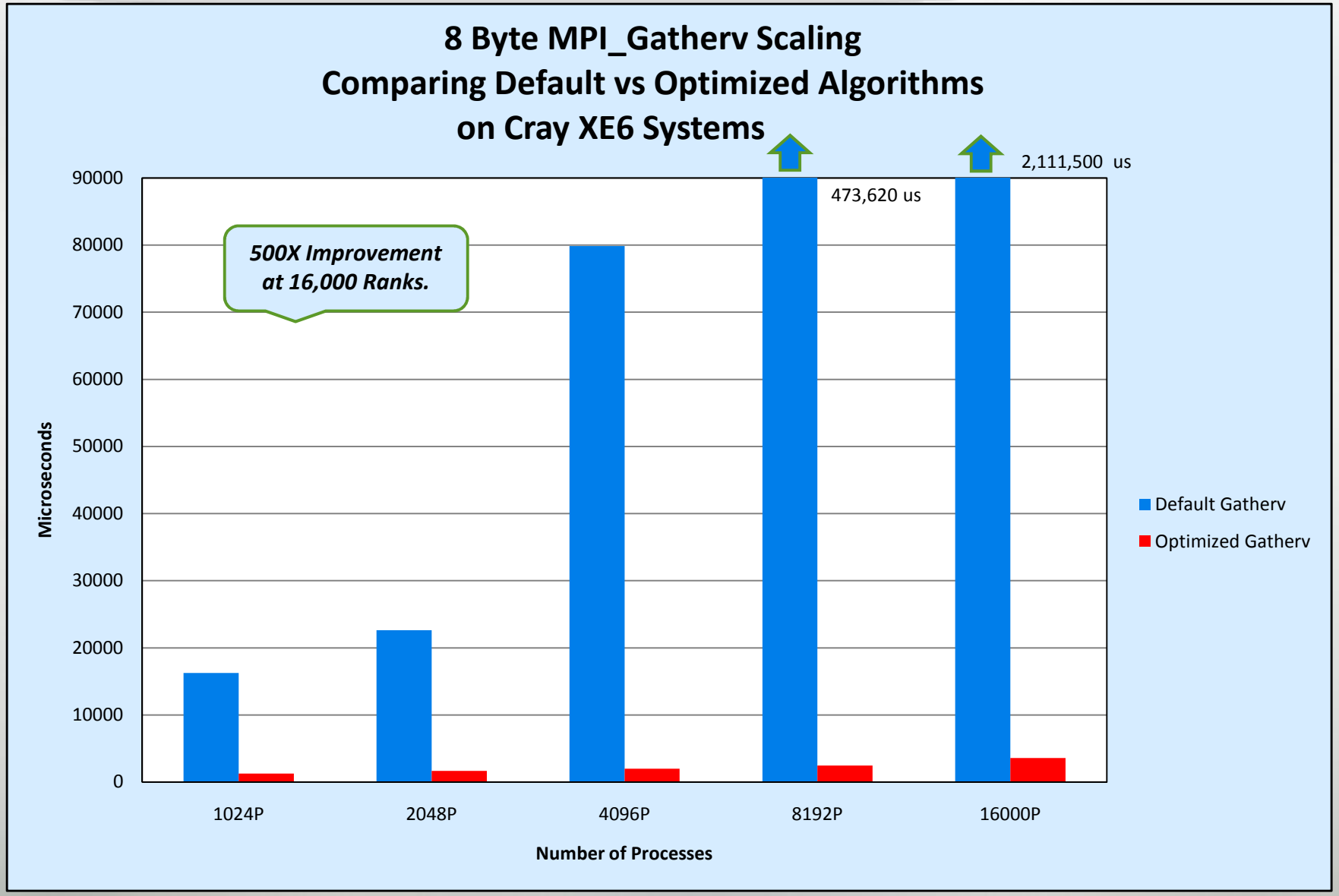
MPI_Allgather / MPI_Allgatherv Performance



MPI_Alltoall / MPI_Alltoallv Performance



MPI_Gatherv Performance



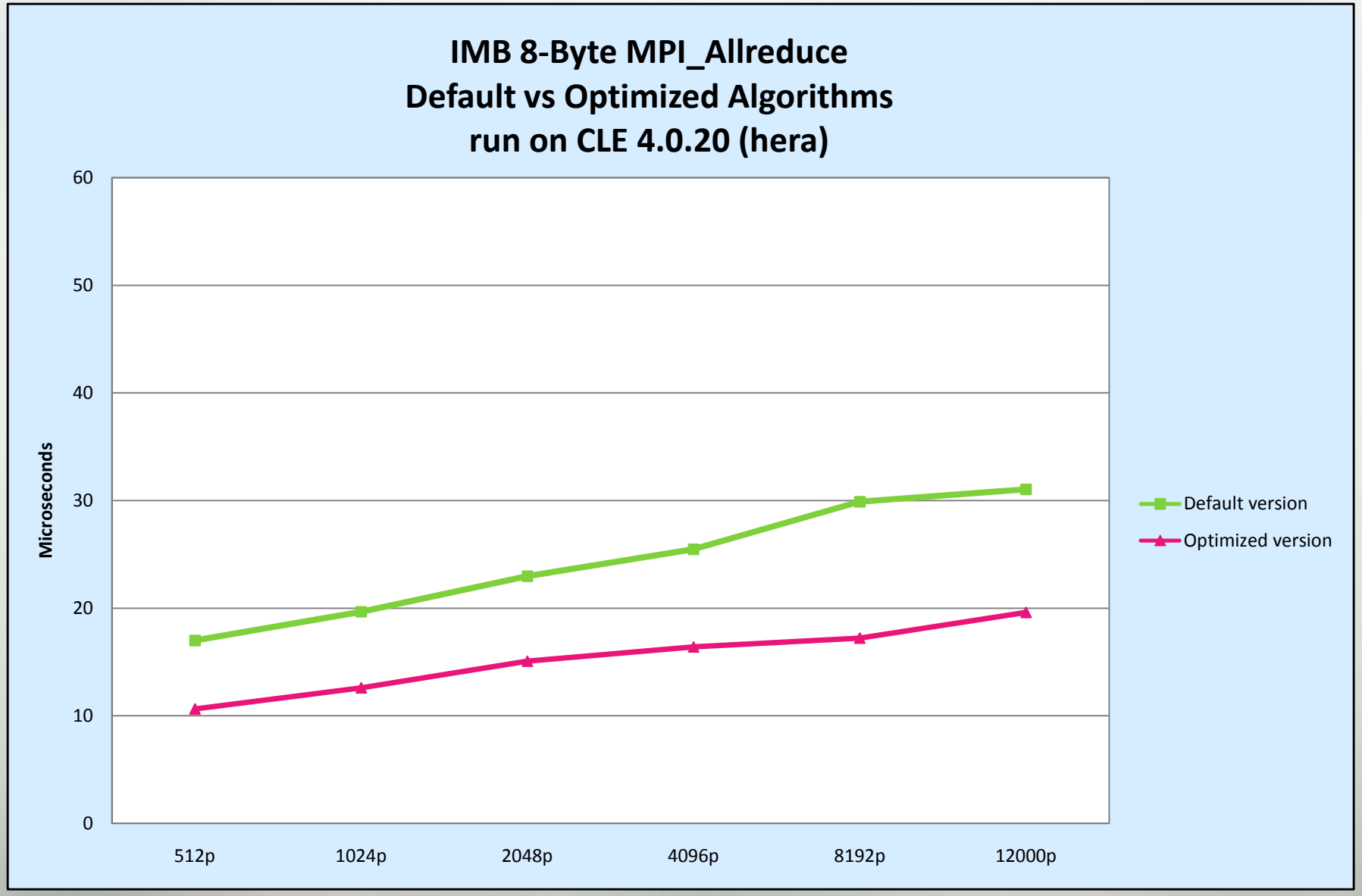
Optimized MPI_Allreduce

- Uses DMAPP GHAL collective enhancements
- Not enabled by default
  - To use: export **MPICH_USE_DMAPP_COLL**=1
  - Supported on CLE 4.0
- Restrictions on Use
  - DMAPP (libdmapp) must be linked into the executable
  - Internally dmapp_init is called (may require hugepages)
  - Message size must be between 4-16 bytes
  - Operation must be a built-in (no user-defined ops)
  - Data type must be supported (float, int, double, long, complex)
  - Not supported for MPMD models
  - Algorithm is not tolerant of transient network errors
  - Will fall back to original algorithm if restrictions are not met

Optimized MPI_Allreduce (cont.)

- Investigating ways to reduce number of restrictions
  - Will involve changes to lower level software (dmapp/ugni)
- OS Noise is a factor when measuring performance
  - Small-message allreduce is very sensitive to OS Noise
  - We've found and removed a significant source of OS Noise
    - Alps was polling every second instead of every 10 seconds (expensive)
    - Alps fix is available in CLE 4.0 UP 01
    - Alps fix significantly reduces OS Noise on medium size systems (12,000 cores)
  - However, we know other sources of OS Noise still exist
  - Core Specialization is the only way to remove OS Noise from the app
    - Add –r 1 to aprun command line

MPI_Allreduce Performance



Collectives Env. Variable Summary

- MPICH_ALLGATHER_VSHORT_MSG
- MPICH_ALLGATHERV_VSHORT_MSG

  Specifies the max message size for which the XE optimized allgather/allgatherv algorithm is used. For MPI_Allgatherv, this is the total amount of data in bytes to be received by each process divided by the number of ranks in the communicator.
- MPICH_ALLTOALL_SHORT_MSG

  Controls the maximum message size (per rank) for which the default MPICH2 algorithm will be used. The default value varies with job size. See intro_mpi man page.
- MPICH_COLL_SYNC

  When set, adds an internal barrier operation before each collective call. Setting to 1 forces a barrier for all collective operations. Can be set for selective barrier operations. May be useful for codes having problems with MPI_Gather(v), MPI_Scatter(v), MPI_Bcast, etc.

Collectives Env. Variable Summary (cont.)

- MPICH_GATHERV_SHORT_MSG

  Analogous to MPICH_ALLGATHERV_VSHORT_MSG.

- MPICH_ALLREDUCE_NO_SMP
- MPICH_REDUCE_NO_SMP

  Disables the default SMP node aware allreduce and reduce algorithms. This may be necessary for codes sensitive to changes in the order ranks' contributions to global sums, etc.

- MPICH_COLL_OPT_OFF

  Disables Cray XE specific collective optimizations. Setting to 1 disables all optimizations. See intro_mpi man page.

Additional Performance Tuning Suggestions

Learn how to use Large Pages

- Load one of the large page modules like craype-hugepages2M
- man intro_hugepages
- May help MPI apps that have messages sizes large enough to use the rendezvous path
- No significant benefit to using larger than 2M pages for MPI only applications
- MPICH2 internally tries to use large pages for mailboxes and DMA buffers. Starting with MPT 5.4.0, the default is to NOT fall back to 4KB pages if large pages aren't available. To disable this and try to use 4KB pages if large pages aren't available –
  export MPICH_GNI_MALLOC_FALLBACK = enabled

Asynchronous Progress

- Available starting in MPT 5.4 release.
- Uses helper threads to progress the MPI state engine while application is computing.
- Only inter-node messaging is effectively progressed (relies on BTE for data motion).
- Only rendezvous path is effectively progressed.
- Don't use with applications making use of derived data types.
- Best if used in conjunction with corespec (see aprun –r argument). Reserving a core/node is important for this feature to be effective.

Asynchronous Progress (cont.)

Need to set the following environment variables to activate the asynchronous progress threads:

- export MPICH_NEMESIS_ASYNC_PROGRESS=1
- export MPICH_MAX_THREAD_SAFETY=multiple
- Run the application requesting corespec and reserving a core

  aprun –n X –r 1 ./a.out

- *apcount* utility can help in scaling up the process count needed to run the job for a given number of cores/node reserved for corespec. See apcount man page.

MPI-2 One Sided

- MPI-2 one sided currently uses the E0,E1 paths for data transfers. This is not optimal.
- Cray is working with Argonne to enhance Nemesis to better handle MPI-2 and MPI-3 RMA.
- MPI-2 RMA is not recommended at this time for performance critical regions of applications running on Hopper.

MPICH_GNI_RECV_CQ_SIZE

- Controls size of the receive (RX) CQ
- Apps trying to run at high scale with many to one patterns may benefit somewhat by setting this higher than the default of 40960 entries
- Note overrunning the RX queue is okay, but it slows MPICH2 down while its recovering from the overrun.

Other environment variables

- MPICH_SCATTERV_SYNCHRONOUS – may be useful if an app is spending a lot of time in this routine. Setting this variable forces the algorithm to use blocking sends rather than the default non-blocking sends.
- MPICH_GNI_DYNAMIC_CONN – may want to set to *disabled* if the application does all-to-all or all-to-one patterns

MPI I/O – General Recommendations

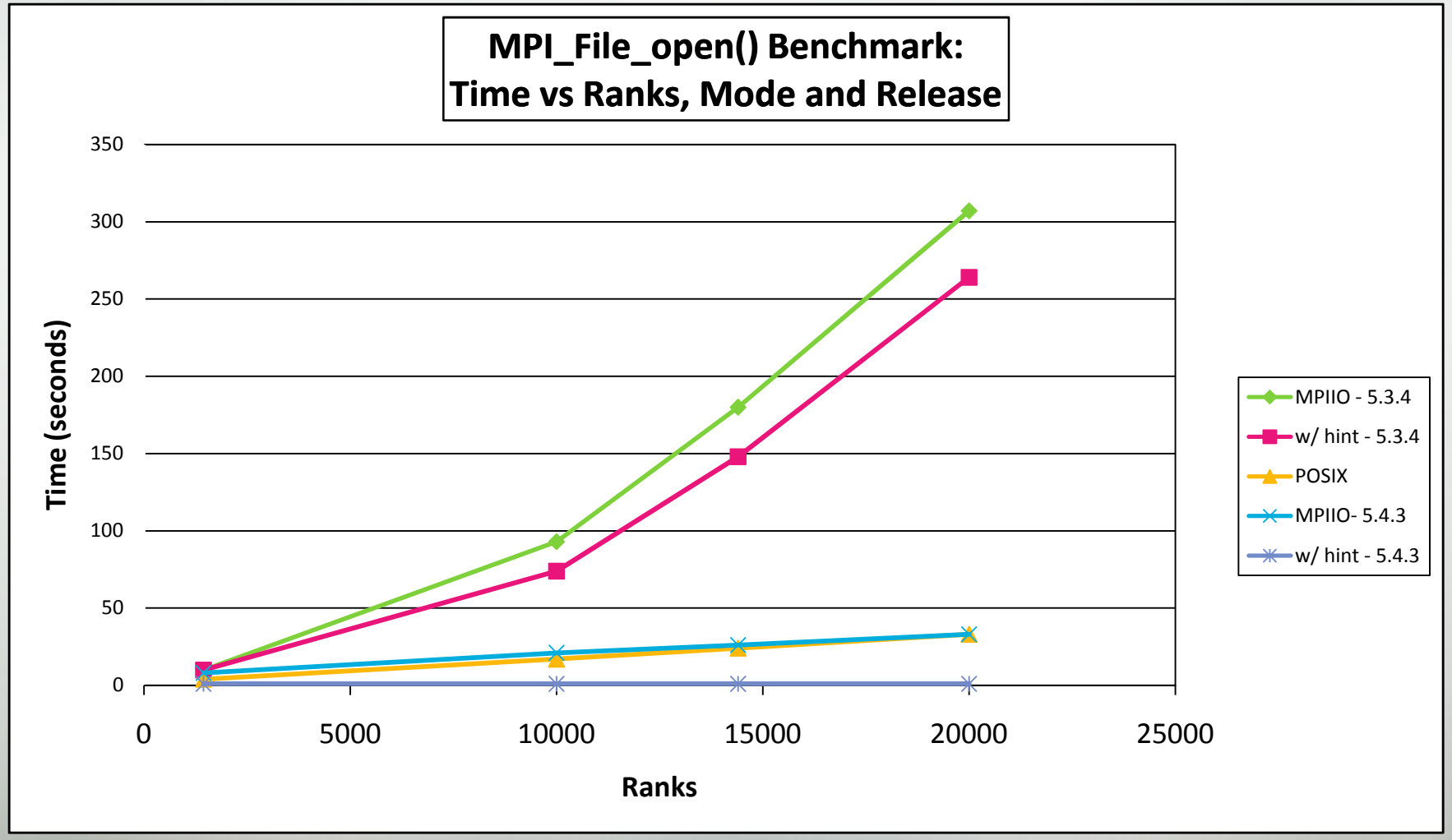
- Using MPI collective I/O to a single shared file can have significant advantages over MPI independent I/O or POSIX I/O to a file per process. See "Getting Started on MPI I/O" S-2490

- Cray's collective I/O implementation is tuned for Lustre such that simply setting the stripe count on the directory where the file will be created is often all the user needs to set. See the "intro_mpi" man page for MPIIO hints for additional tuning options.

MPI I/O - Recent Improvements

- 5.4.3 has a more efficient and scalable MPI_File_open()
- Setting the romio_no_indep_rw=true hint causes only aggregators to open the file. This is better for the application and better for the whole system. See the "intro_mpi" man page for restrictions.

MPI I/O Open Improvements



What’s Next

What Are We Working on Next?

- GPU (XK6) related enhancements
- Scalability Enhancements
  - Gemini Shared Message Queue
  - Working with ANL to reduce memory footprint of MPICH2 internal structures
- Improvements to MPI I/O
  - Better tuning for GPFS via DVS
  - Better percentage of file-per-process performance for single-shared-file. (Currently ~50% for large writes)
- MPI Stats / Bottlenecks Display

Release Schedule

MPT Release Schedule

- MPT 5.3 released June 2011. Last release supporting XT/Seastar. You shouldn't be using a package older than this one.
- MPT 5.4 released November 2011. This is the recommended package to be using.
- MPT 5.5 release planned for June 2012.

References

- MPICH2 Nemesis Internals – http://wiki.mcs.anl.gov/mpich2/index.php/Nemesis_Network_Module_API
- Paper about MPICH2 on XE

  A uGNI-Based MPICH2 Nemesis Network Module for the Cray XE, H. Pritchard, I. Gorodetsky, and D. Buntinas, EuroMPI 2011.

Questions

CRAY

THE SUPERCOMPUTER COMPANY