Optimizing Cray MPI and Cray SHMEM for Current and Next Generation Cray-XC Supercomputers

K. Kandalla, D. Knaak, K. McMahon, N. Radcliffe and M. Pagel

Cray Inc.

Apr. 2015

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 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 names and brands may be claimed as the property of others. Other product and service names mentioned herein are the trademarks of their respective owners..

Agenda

- **Introduction**
- **Recent Cray MPI and Cray SHMEM optimizations and features**
- **Summary and Future work**
- **Discussion**

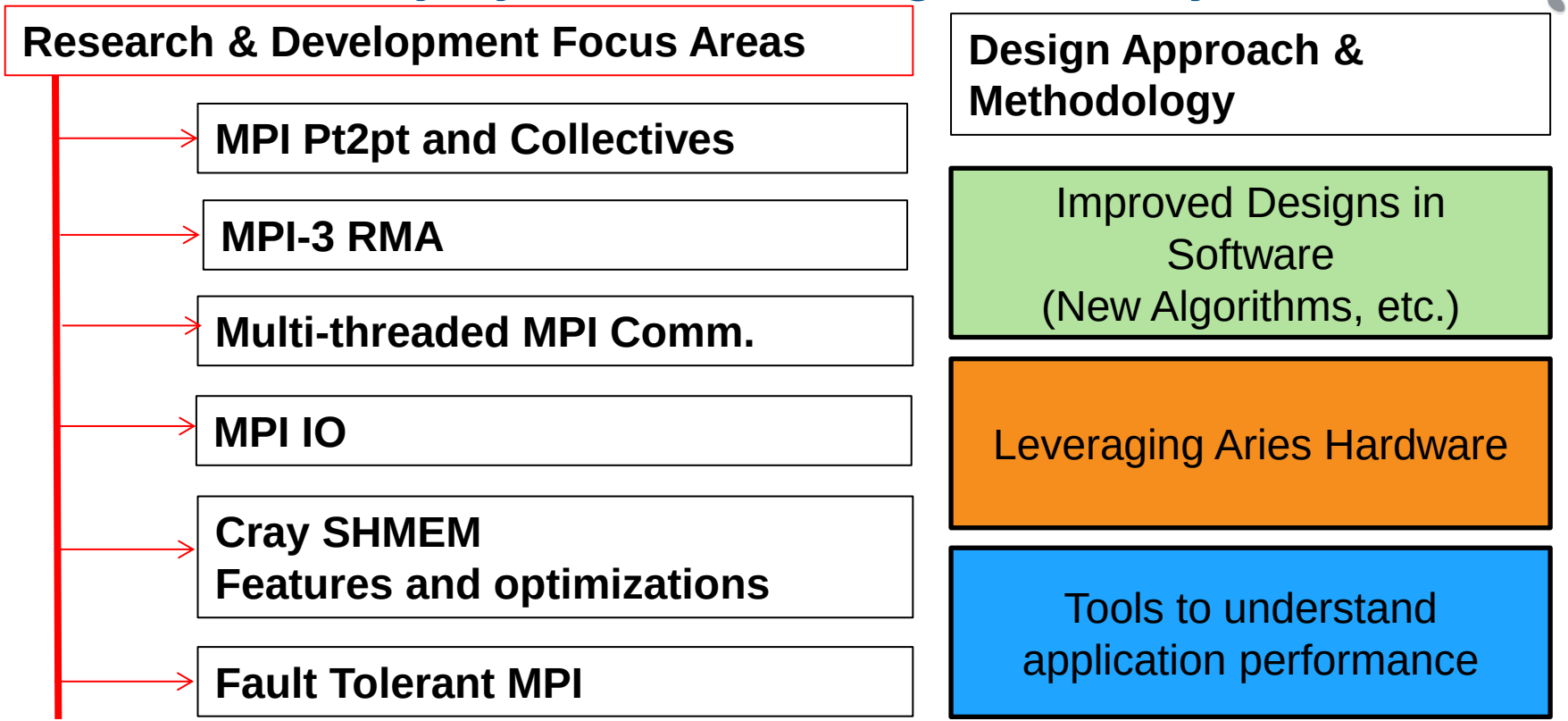
Introduction

- Multi-/Many-core architectures and accelerators are driving the HPC evolution
- Modern interconnects offer low latency, high bandwidth communication
- Cray designs some of the fastest supercomputers
- Intel MIC and NVIDIA GPUs pose new challenges and opportunities
- Critical to design MPI and SHMEM software stacks in a very efficient manner

Designing High Performance MPI and SHMEM Software on Cray Systems: Challenges and Objectives

- 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
- Fault resilient communication

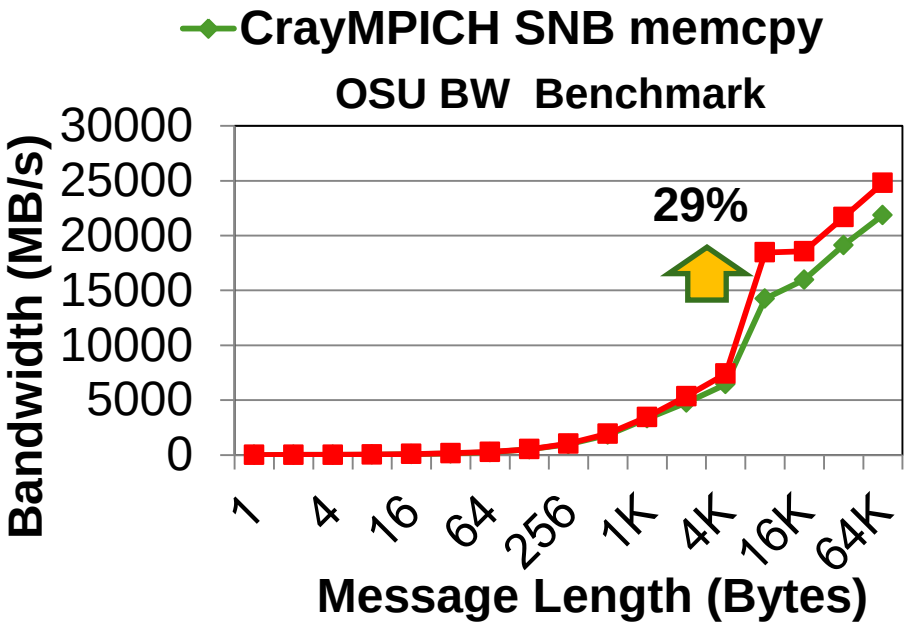
Designing High Performance MPI and SHMEM Software on Cray Systems: Challenges and Objectives

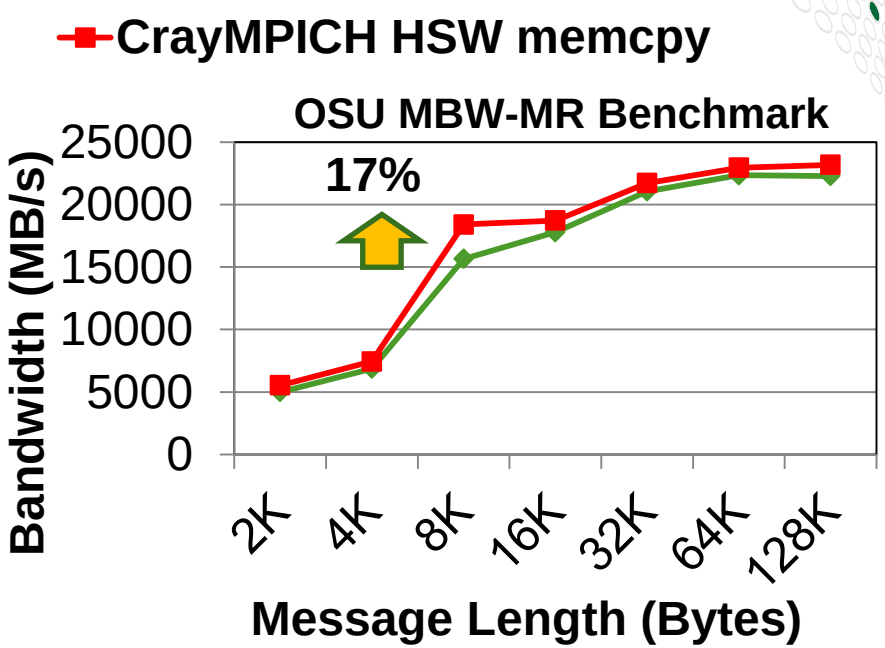


Agenda

- Introduction
- **Recent Cray MPI and Cray SHMEM optimizations and features:**
  - **Architecture Optimized memcpy**
  - Collective Optimizations
    - Blocking (Communication Latency)
    - Non-Blocking (Comm./Comp. Overlap)
  - MPI-3 RMA and GPU Optimizations
  - Fine-grained Multi-threading for MPI
  - MPI-IO
  - Cray SHMEM Optimizations and Features
  - MPI User Level Fault Mitigation
- Summary and Future work
- Discussion

Architecture Optimized memcpy





2 MPI Processes: 1 HSW (32-core) node

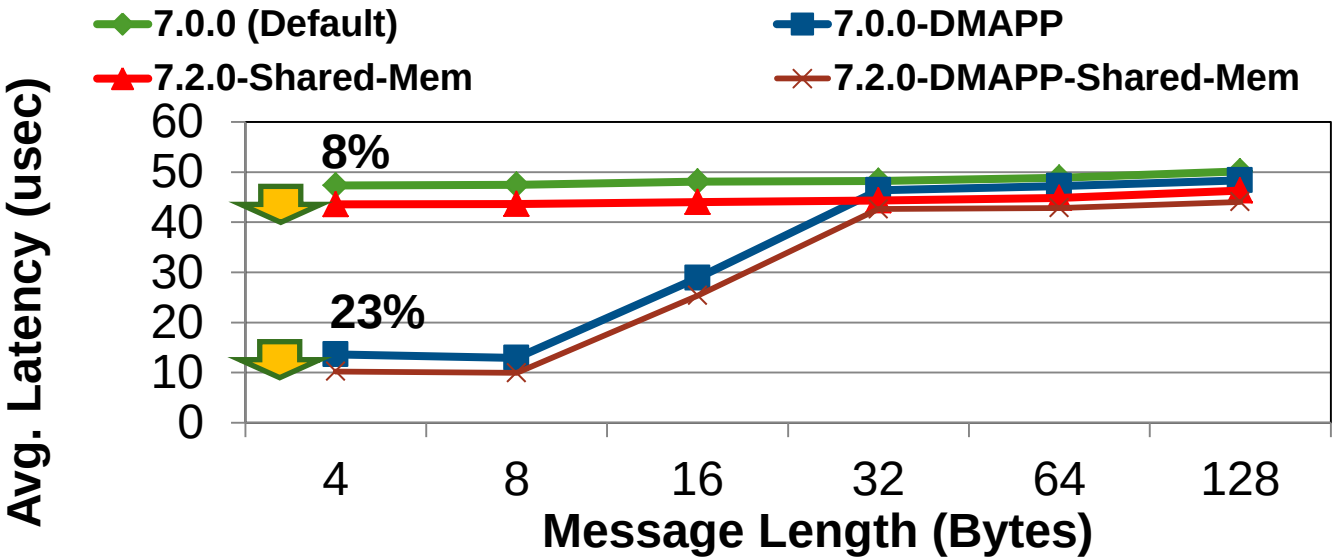
MPICH_OPTIMIZED_MEMCPY = 2 (Default: 1; Values: 0, 1, 2)

If MPICH_USE_SYSTEM_MEMCPY is set, Optimized memcpy is disabled

Agenda

- Introduction
- **Recent Cray MPI and Cray SHMEM optimizations and features:**
  - Architecture Optimized memcpy
  - **Collective Optimizations**
    - **Blocking (Communication Latency)**
    - Non-Blocking (Comm./Comp. Overlap)
  - MPI-3 RMA and GPU Optimizations
  - Fine-grained Multi-threading for MPI
  - MPI-IO
  - Cray SHMEM Optimizations and Features
  - MPI User Level Fault Mitigation
- Summary and Future work
- Discussion

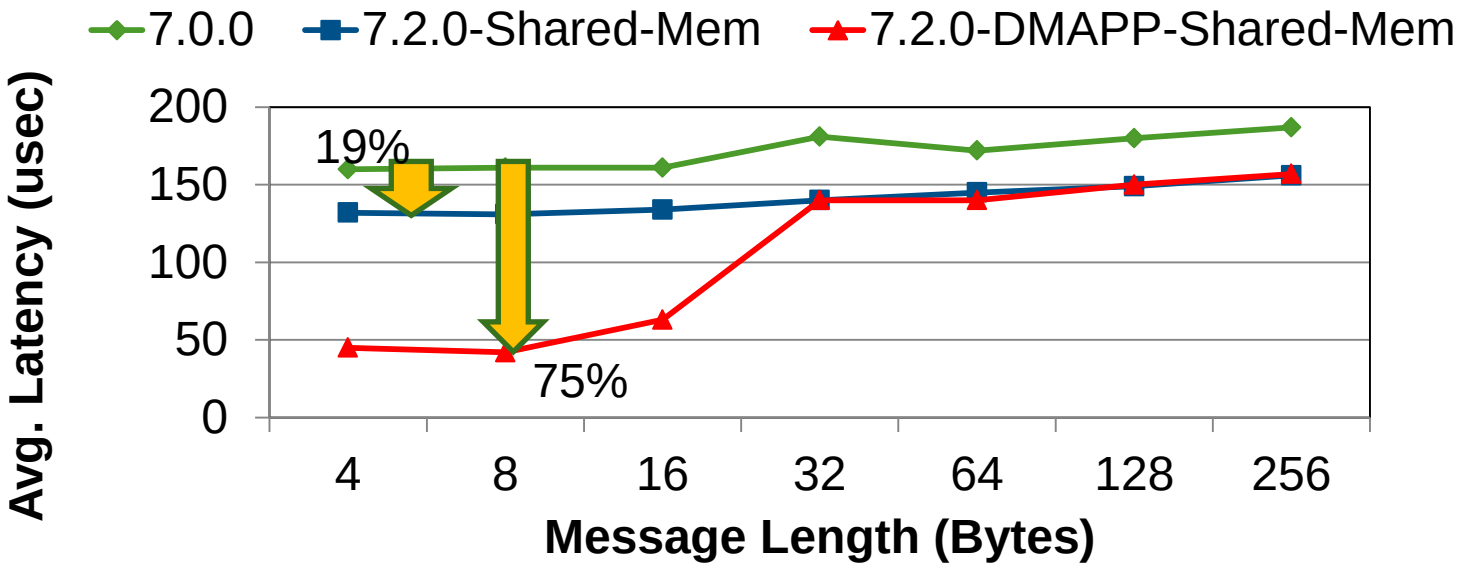
MPI_Allreduce (Small Messages)



119,648 MPI Processes: 32 ppn, 32-core HSW nodes.
7.2.0-DMAPP-SHARED-MEM improves performance by about 23% when compared to 7.0.0-DMAPP

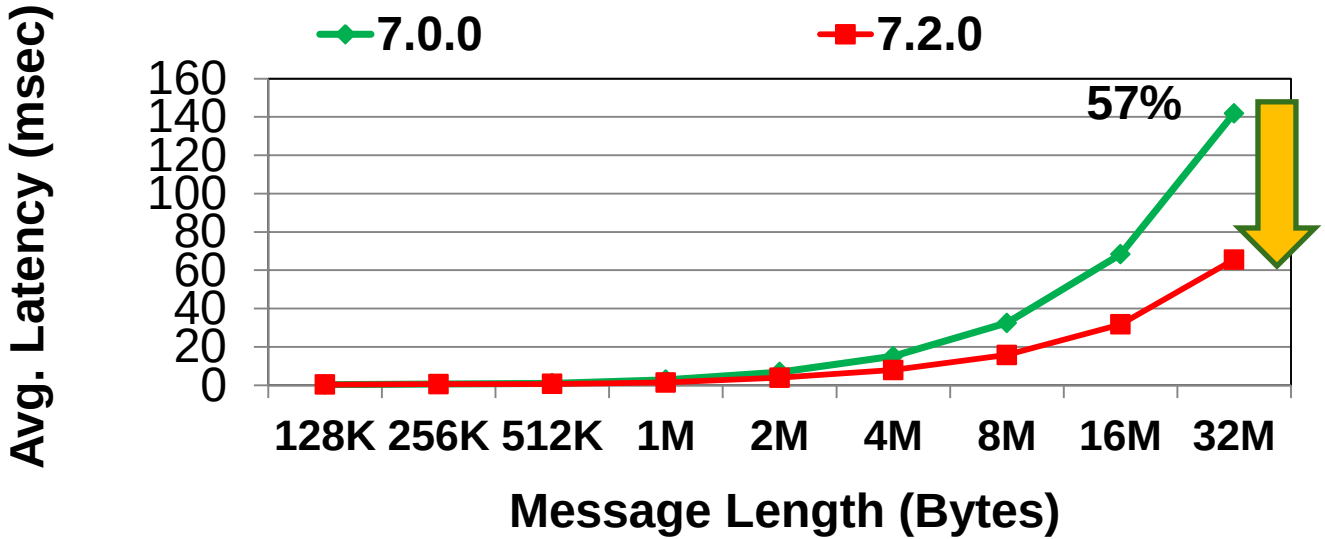
MPICH_SHARED_MEM_COLL_OPT = 1 (Default: 0. Soon to be enabled by default)
MPICH_USE_DMAPP_COLL = 1 (Default: 0) (-Wl,--whole-archive,-ldmapp,--no-whole-archive)

MPI_Allreduce (Small Messages, on KNC)



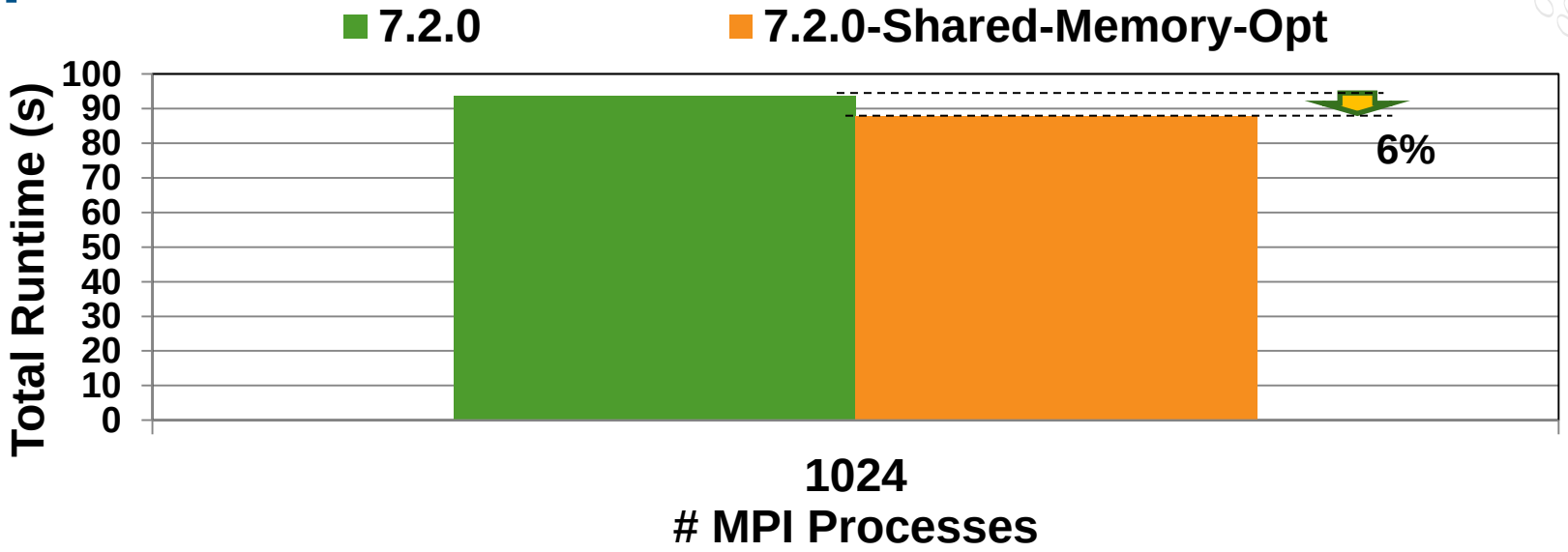
256 MPI Processes: 16 ppn, 16 KNC nodes
7.2.0 with Shared-memory collective optimizations performs about 19% better
7.2.0-DMAPP + Shared-memory optimization improves performance by about 75%.
MPICH_SHARED_MEM_COLL_OPT = 1 (Default: 0)
MPICH_USE_DMAPP_COLL = 1 (Default: 0) -Wl,--whole-archive,-ldmapp,--no-whole-archive

MPI_Allreduce (Large Messages)



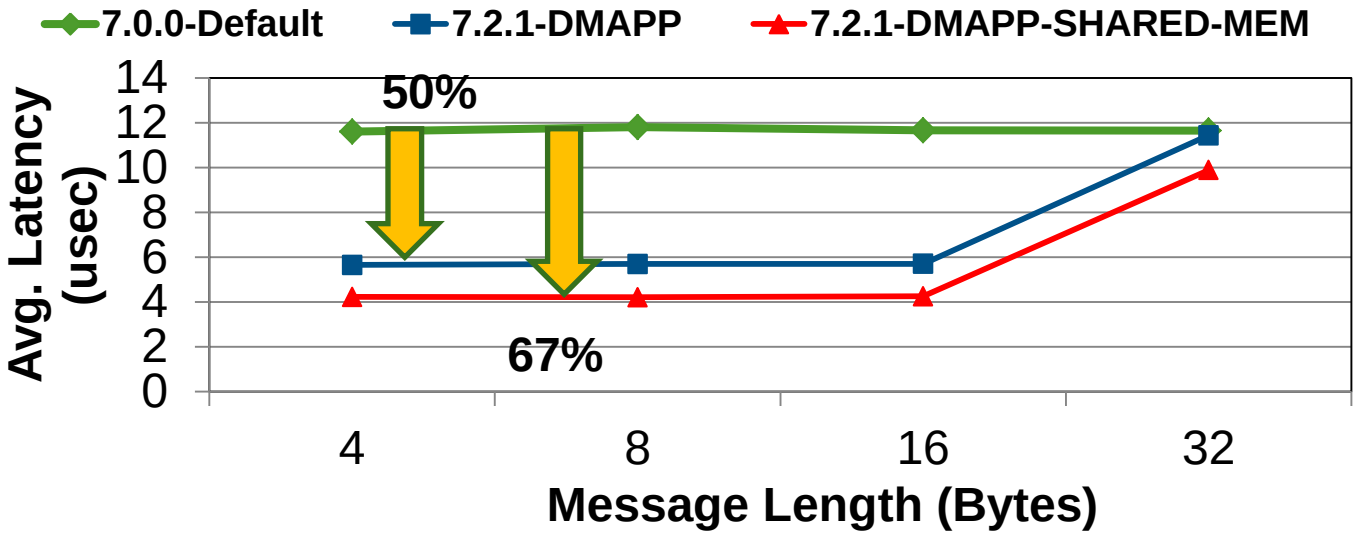
8,192 MPI Processes: 32 ppn, 32-core HSW nodes.
Cray MPI 7.2.0 performs about 57% better than Cray MPI 7.0.0 for large message Allreduce operations. (*Enabled* by default)

Application Performance: POP



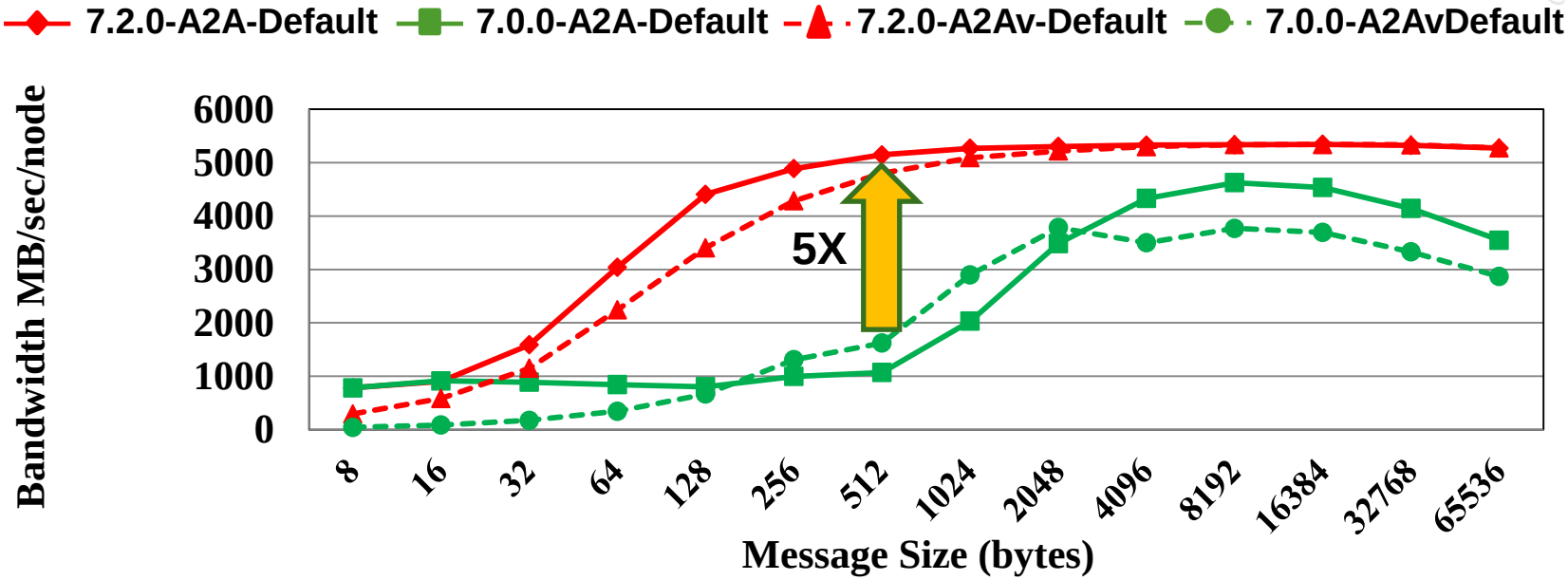
1,024 MPI Processes, 32 core Haswell nodes, 32 ppn.
nx_global = 2048 ny_global = 768; 40 vertical levels, 2 tracers, 1500 timesteps
MPICH_SHARED_MEM_COLL_OPT = 1 (Default: 0)
POP Total runtime improved by about 6%

MPI_Bcast (Small Messages)



8,192 MPI Processes: 32 ppn, 32-core HSW nodes.
7.2.1-DMAPP performs about 50% better
7.2.1-DMAPP with Shared-Memory optimization performs about 67% better
MPICH_SHARED_MEM_COLL_OPT = 1 (Default: 0. Soon to be enabled by default)
MPICH_USE_DMAPP_COLL = 1 (Default: 0) -Wl,--whole-archive,-ldmapp,--no-whole-archive

MPI_Alltoall and MPI_Alltoallv Optimizations



4,096 MPI Processes. 24 ranks per node, 64M Hugepages.

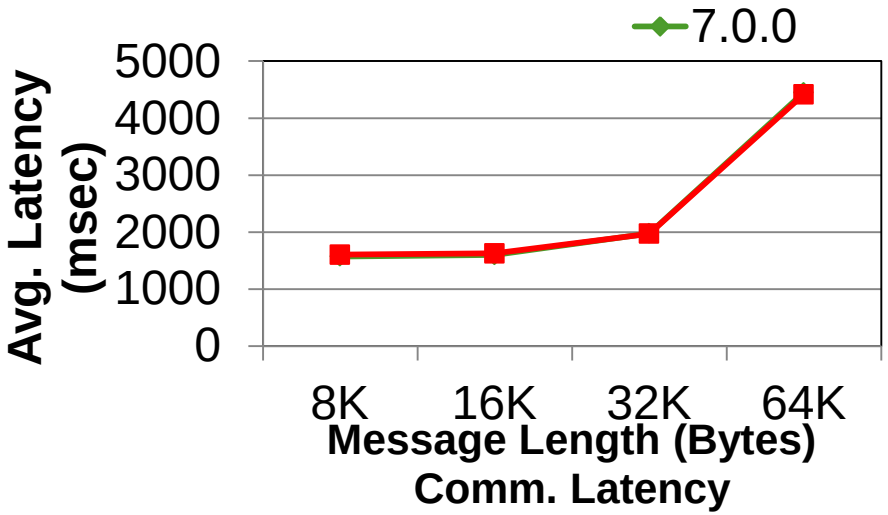
Enabled **by default. Optimized Alltoall/Alltoallv solutions perform up to 5X faster**

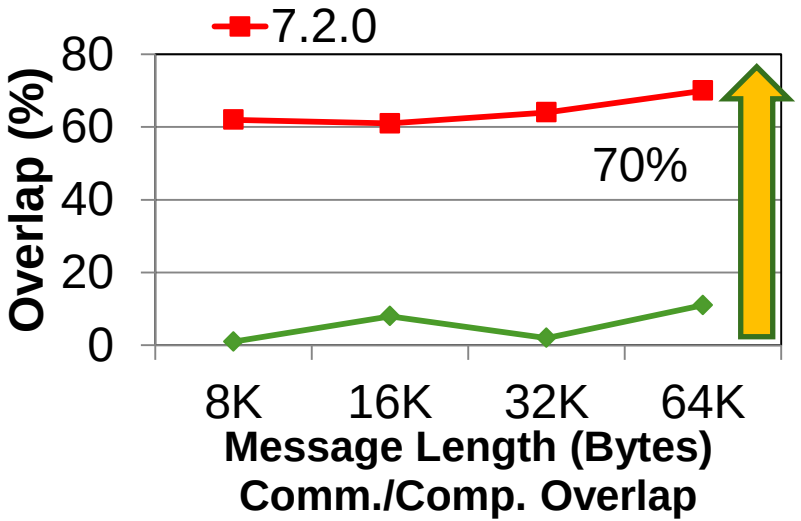
(Set MPICH_GNI_COLL_OPT_OFF to *disable*)

Agenda

- Introduction
- **Recent Cray MPI and Cray SHMEM optimizations and features:**
  - Architecture Optimized memcpy
  - **Collective Optimizations**
    - Blocking (Communication Latency)
    - **Non-Blocking (Comm./Comp. Overlap)**
  - MPI-3 RMA and GPU Optimizations
  - Fine-grained Multi-threading for MPI
  - MPI-IO
  - Cray SHMEM Optimizations and Features
  - User Level Fault Mitigation
- Summary and Future work
- Discussion

MPI_Ialltoall (Large Messages)





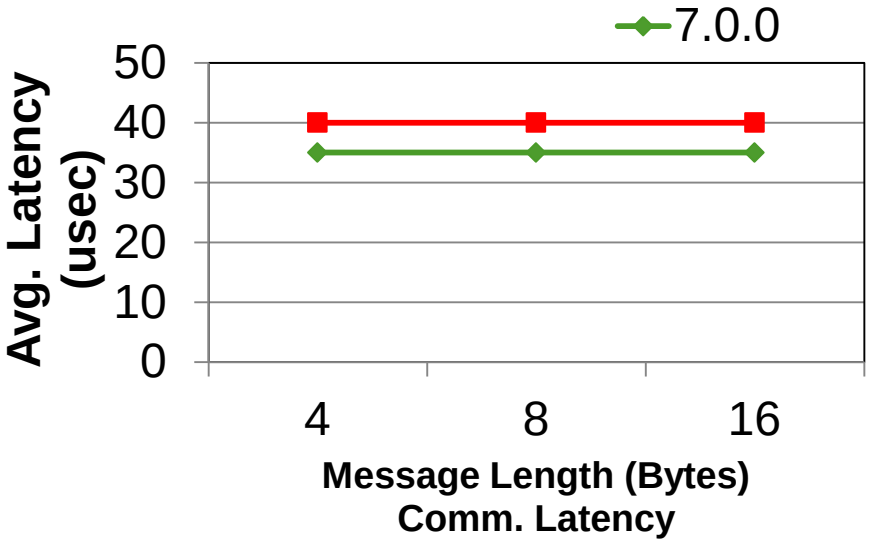
8,192 MPI Processes: 24-Core Ivy-Bridge nodes

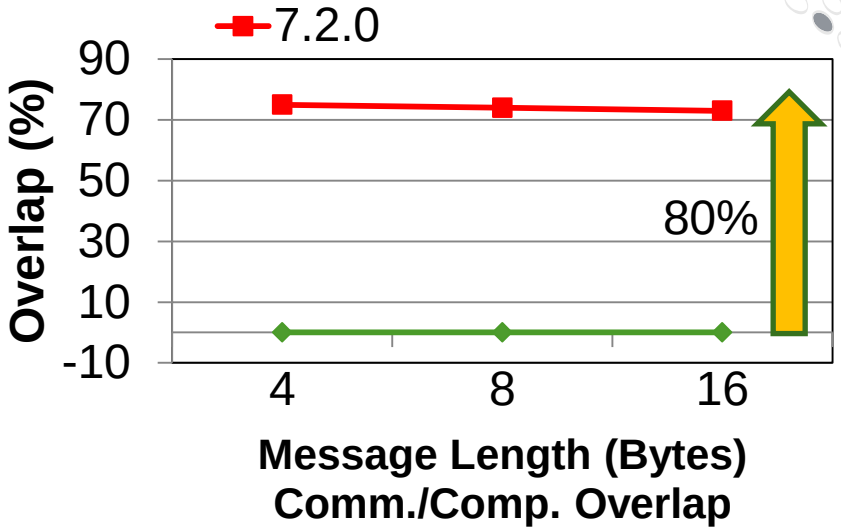
New prototype delivers up to 70% comm./comp. overlap

MPICH_NEMESIS_ASYNC_PROGRESS=1;

MPICH_MAX_THREAD_SAFETY=multiple; (CLE 5.2 UP02 or newer)

MPI_Iallreduce (Small Messages)





8,192 MPI Processes: 24-Core Ivy-Bridge nodes

New Iallreduce optimization delivers up to 80% comm./comp. overlap

export MPICH_NEMESIS_ASYNC_PROGRESS=1; export MPICH_SHARED_MEM_COLL_OPT=1

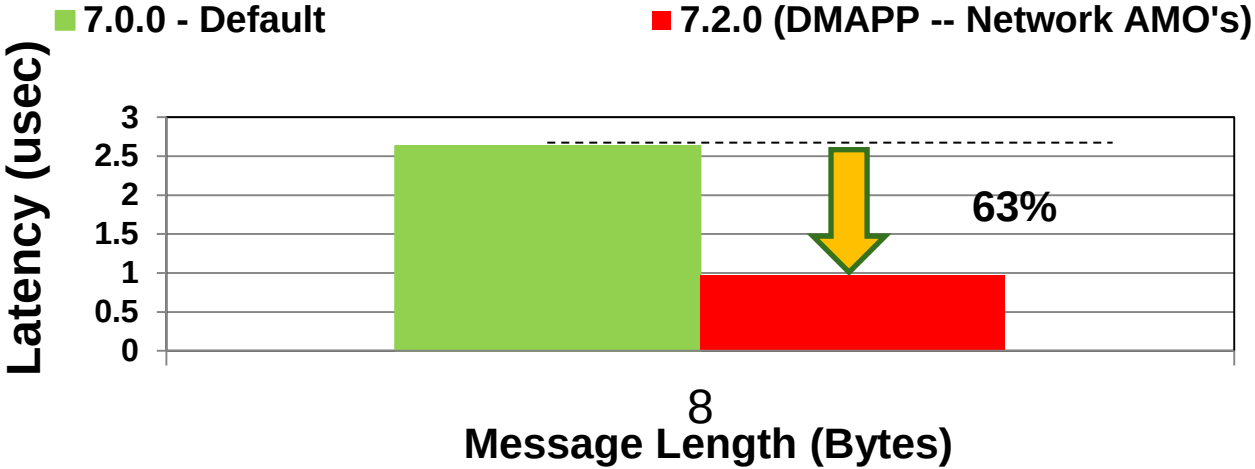
export MPICH_MAX_THREAD_SAFETY=multiple; export MPICH_USE_DMAPP_COLL=1

(CLE 5.2 UP02 or newer)

Agenda

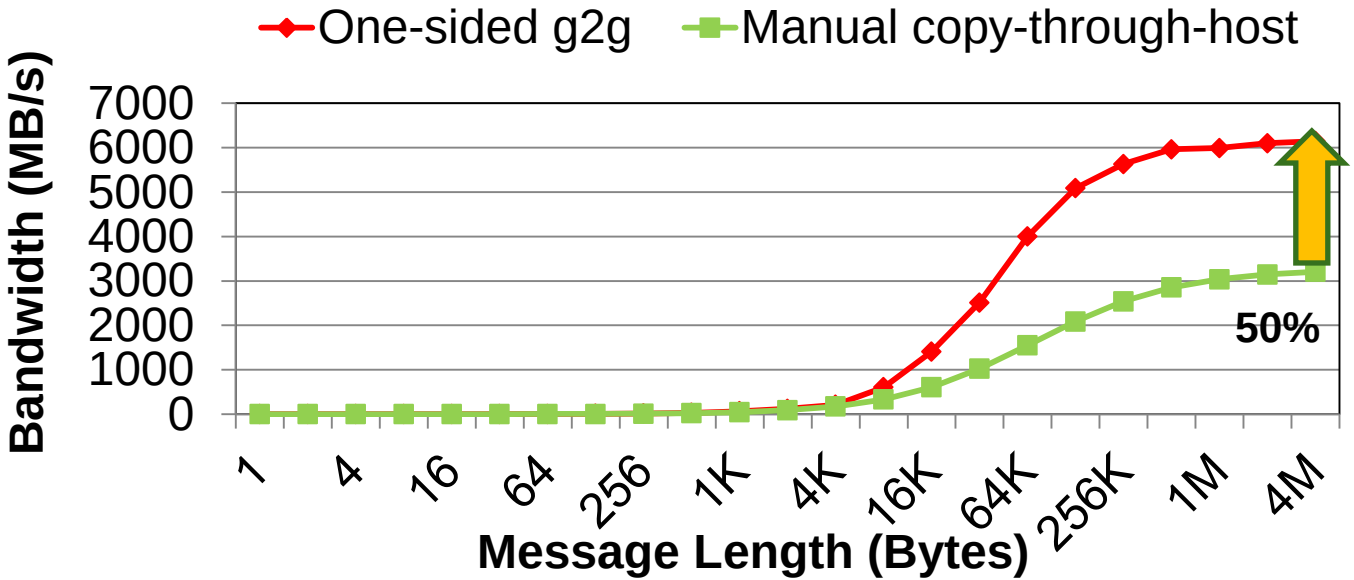
- Introduction
- **Recent Cray MPI and Cray SHMEM optimizations and features:**
  - Architecture Optimized memcpy
  - Collective Optimizations
    - Blocking (Communication Latency)
    - Non-Blocking (Comm./Comp. Overlap)
  - **MPI-3 RMA and GPU Optimizations**
  - Fine-grained Multi-threading for MPI
  - MPI-IO
  - Cray SHMEM Optimizations and Features
  - User Level Fault Mitigation
- Summary and Future work
- Discussion

MPI-3 RMA: 8-Byte MPI_Put Latency Network Atomic Memory Operations



7.2.0-DMAPP improves MPI-3 RMA small message latency by about 63%
Design uses Atomic Memory Operations offered by the Aries network.
MPICH_RMA_OVER_DMAPP = 1 (Default: 0)
MPICH_RMA_USE_NETWORK_AMO=1 (Default: 0)
-Wl,--whole-archive,-ldmapp,--no-whole-archive

MPI-3 RMA GPU-to-GPU

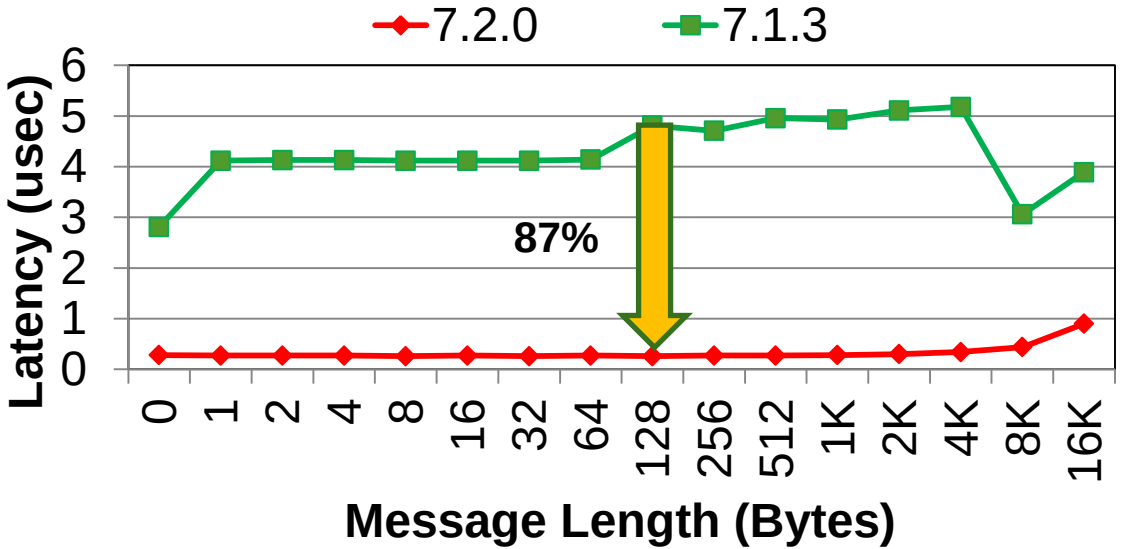


Optimized GPU-to-GPU communication for point-to-point and collectives

New prototype designs improve GPU-to-GPU RMA communication bandwidth by about 50%.

(Internal prototype available. Will be released in June 2015)

MPI One-sided: On-node latency (Post Start Wait Complete)



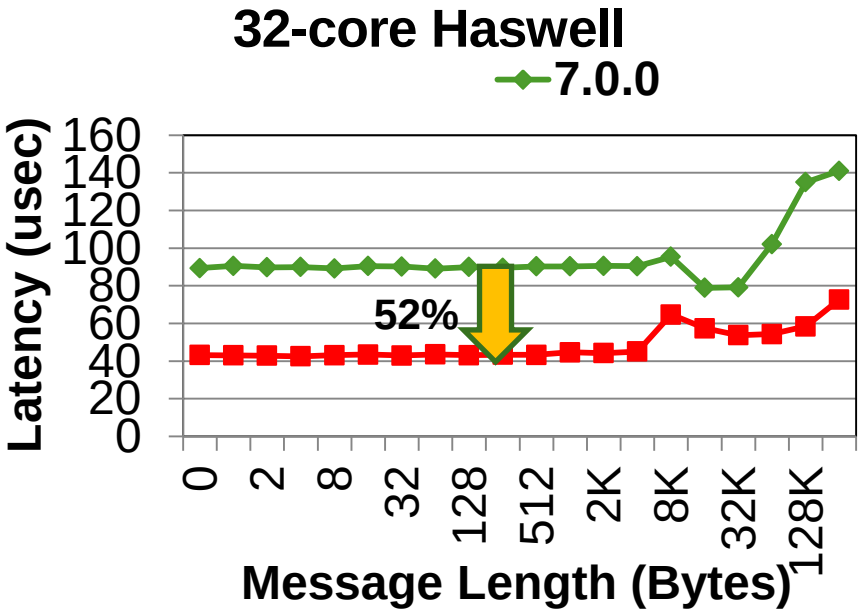
2 MPI Processes. 1 Ivy-Bridge compute node (24-core)

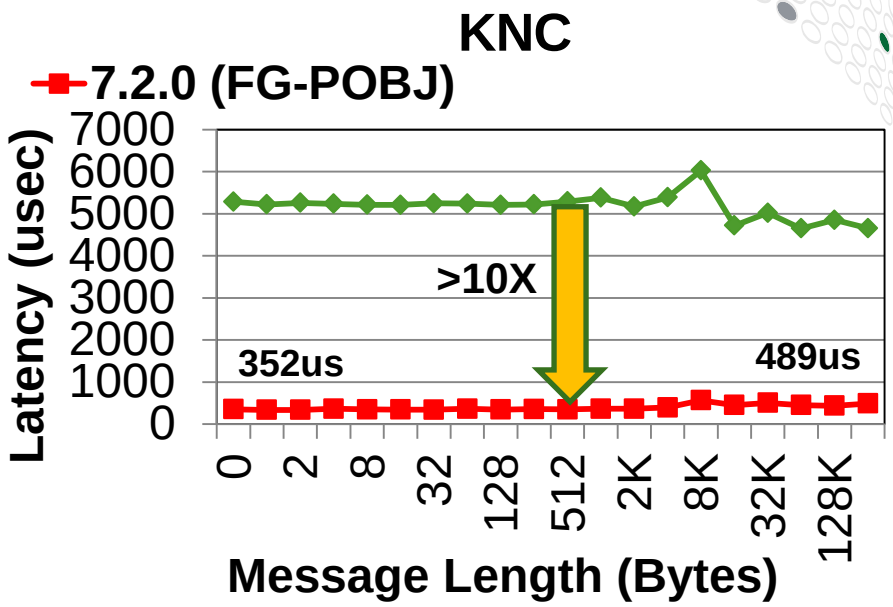
On-Node MPI-3 one-sided operations perform about 87% better with Cray MPICH 7.2.0 (*Enabled* by default)

Agenda

- Introduction
- **Recent Cray MPI and Cray SHMEM optimizations and features:**
  - Architecture Optimized memcpy
  - Collective Optimizations
    - Blocking (Communication Latency)
    - Non-Blocking (Comm./Comp. Overlap)
  - MPI-3 RMA and GPU Optimizations
  - **Fine-grained Multi-threading for MPI**
  - **MPI-IO**
  - **Cray SHMEM Optimizations and Features**
  - **User Level Fault Mitigation**
- **Summary and Future work**
- **Discussion**

Fine-Grained Multi-threading for MPI





Osu_latency_mt.c benchmark. 1 ppn, 2 nodes, 31 threads per process

MPICH_MAX_THREAD_SAFETY=multiple

cc –o osu_latency_mt.x –craympich-mt osu_latency_mt.c

aprun –n2 –N1 –d32 ./osu_latency_mt.x

Agenda

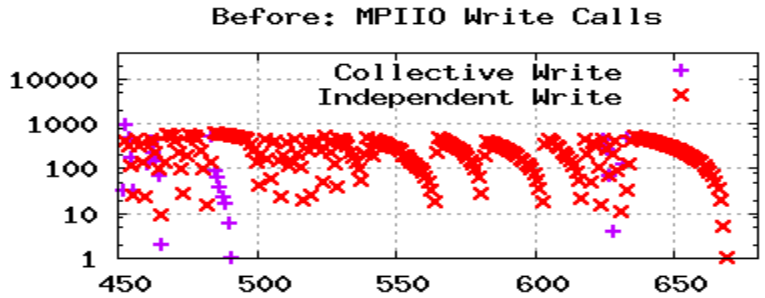
- Introduction
- **Recent Cray MPI and Cray SHMEM optimizations and features:**
  - Architecture Optimized memcpy
  - Collective Optimizations
    - Blocking (Communication Latency)
    - Non-Blocking (Comm./Comp. Overlap)
  - MPI-3 RMA and GPU Optimizations
  - Fine-grained Multi-threading for MPI
  - **MPI-IO**
  - **Cray SHMEM Optimizations and Features**
  - **User Level Fault Mitigation**
- **Summary and Future work**
- **Discussion**

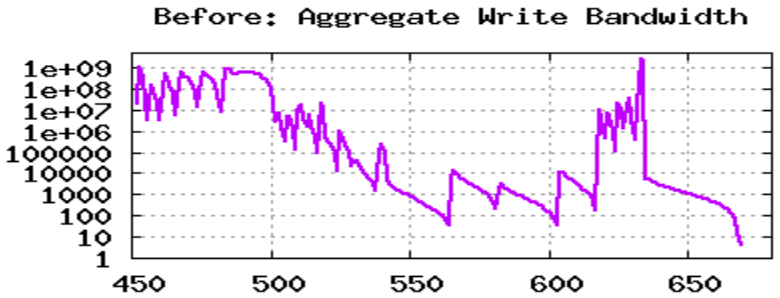
MPI-I/O STATS

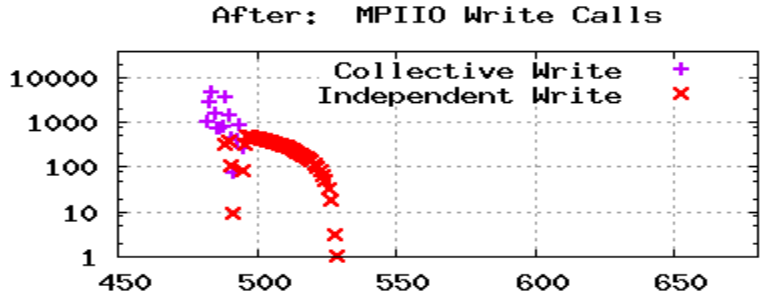
Timeline of MPI-I/O statistics. Many different variables tracked

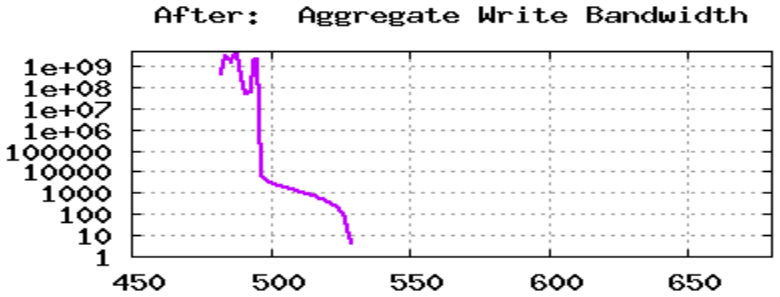
```
export MPICH_MPIIO_STATS=2
```

For more information contact: David Knaak, Bob Cernohous





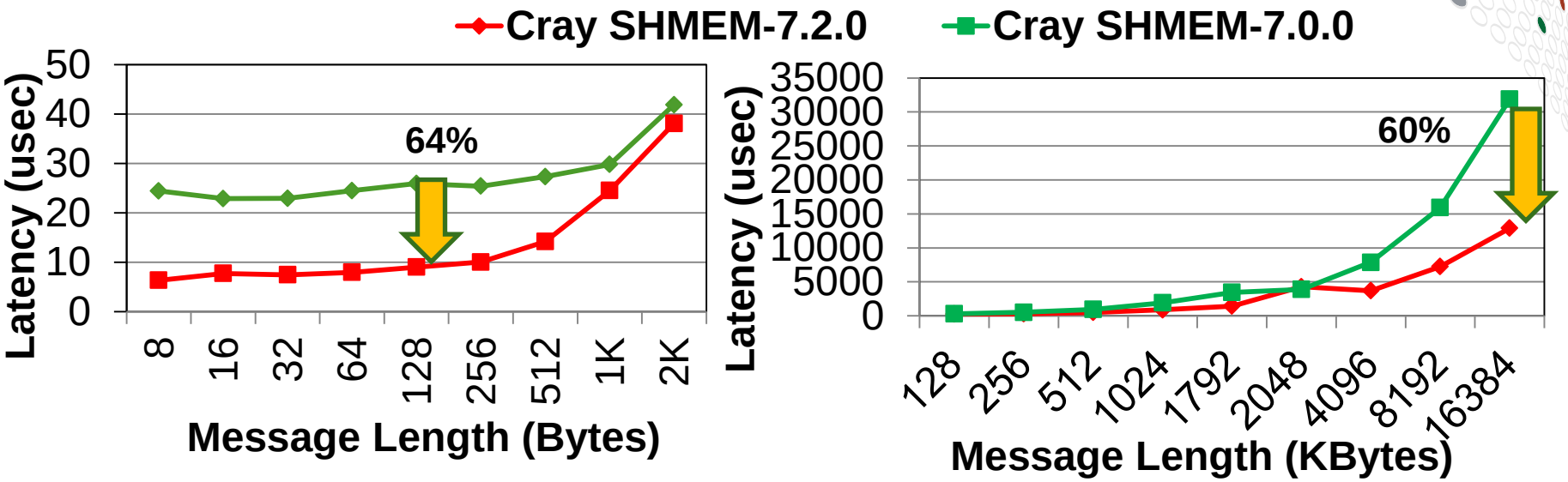




Agenda

- Introduction
- **Recent Cray MPI and Cray SHMEM optimizations and features:**
  - Architecture Optimized memcpy
  - Collective Optimizations
    - Blocking (Communication Latency)
    - Non-Blocking (Comm./Comp. Overlap)
  - MPI-3 RMA and GPU Optimizations
  - Fine-grained Multi-threading for MPI
  - MPI-IO
  - **Cray SHMEM Optimizations and Features**
  - User Level Fault Mitigation
- **Summary and Future work**
- **Discussion**

Cray SHMEM: Features and Optimizations



- **Shmem_broadcast (about 60% improvement in comm. latency)**
  1,024 MPI Processes: 24-Core Ivy-Bridge nodes (Enabled by default)
- **Cray SHMEM supports shmem_global_exit()**
  (Not enabled by default: Set SHMEM_GLOBAL_EXIT=1)

Agenda

- Introduction
- **Recent Cray MPI and Cray SHMEM optimizations and features:**
  - Architecture Optimized memcpy
  - Collective Optimizations
    - Blocking (Communication Latency)
    - Non-Blocking (Comm./Comp. Overlap)
  - MPI-3 RMA and GPU Optimizations
  - Fine-grained Multi-threading for MPI
  - MPI-IO
  - Cray SHMEM Optimizations and Features
  - **MPI User Level Fault Mitigation**
- **Summary and Future work**
- **Discussion**

MPI User Level Fault Mitigation (ULFM)

- Critical to design MPI libraries in a fault resilient manner
- MPI Fault Tolerance Working Group is working towards standardizing the ULFM interface
- A prototype implementation of ULFM in Cray MPICH for XC systems is under development
- Status of the current prototype:
  - supports the new MPI_ERR error classes
  - automatically detects node failures
  - supports a subset of ULFM functions to re-build MPI objects (revoke, shrink, agree..)
- Future development plans depend on MPI Forum directions

Agenda

- Introduction
- Recent Cray MPI and Cray SHMEM optimizations and features:
  - Architecture Optimized Memcpy
  - Collective optimizations
    - Blocking (Communication Latency)
    - Non-Blocking (Comm./Comp. Overlap)
  - MPI-3 RMA and GPU Optimizations
  - Fine-grained Multi-threading for MPI
  - MPI-IO
  - Cray SHMEM Optimizations and features
  - MPI User Level Fault Mitigation
- **Summary and Future work**
- **Discussion**

Summary and Future Work

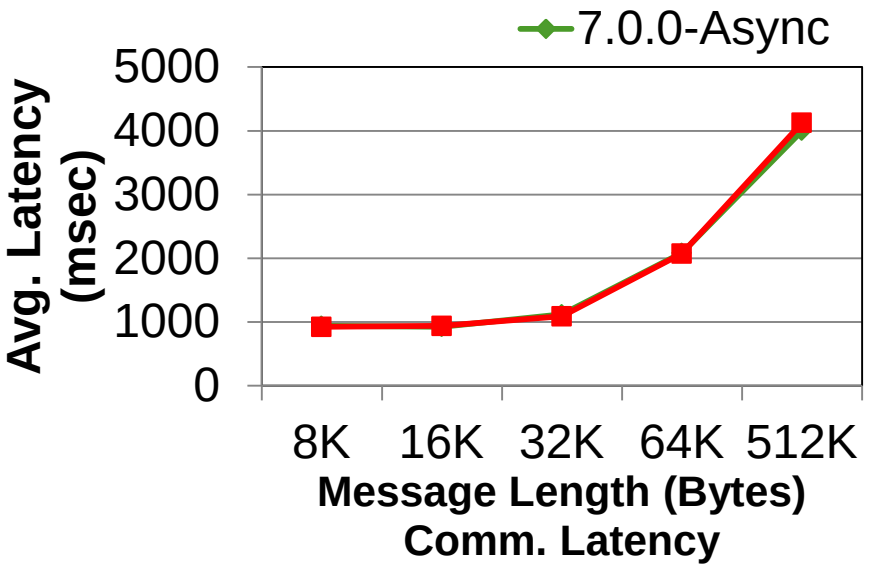
- **Conclusion:**
  - Improved MPI Point-to-Point, Collective and RMA performance
  - New features and optimizations in Cray SHMEM
  - Exploring architecture-specific optimizations for Intel Xeon and Intel Xeon Phi
  - Prototyping ULFM support in Cray MPICH
- **Future Work:**
  - Improving Cray MPICH and Cray SHMEM performance on future Intel Xeon and Intel Xeon Phi processors
  - Reduced memory footprint for Cray MPICH
  - Improved support for multi-threaded MPI applications

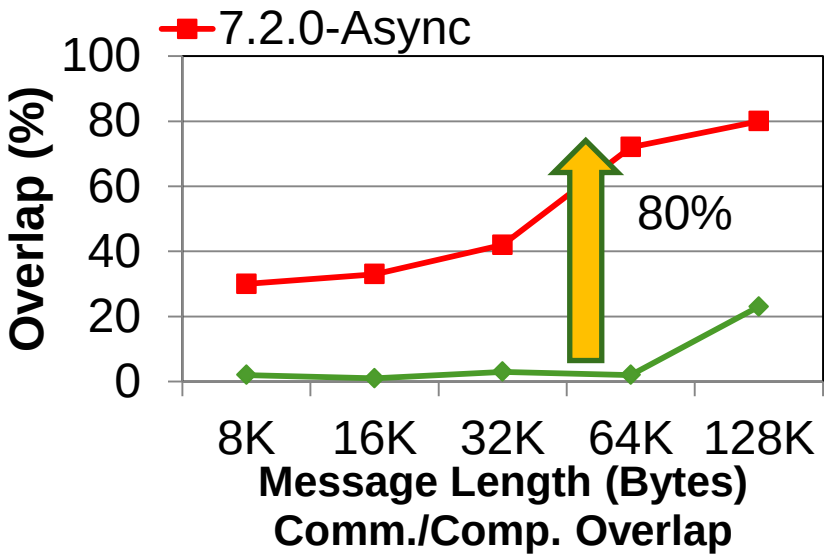
Thank you!

Contact Info:
(kkandalla@cray.com)

Backup Slides:

MPI_Ialltoallv (Large Messages)





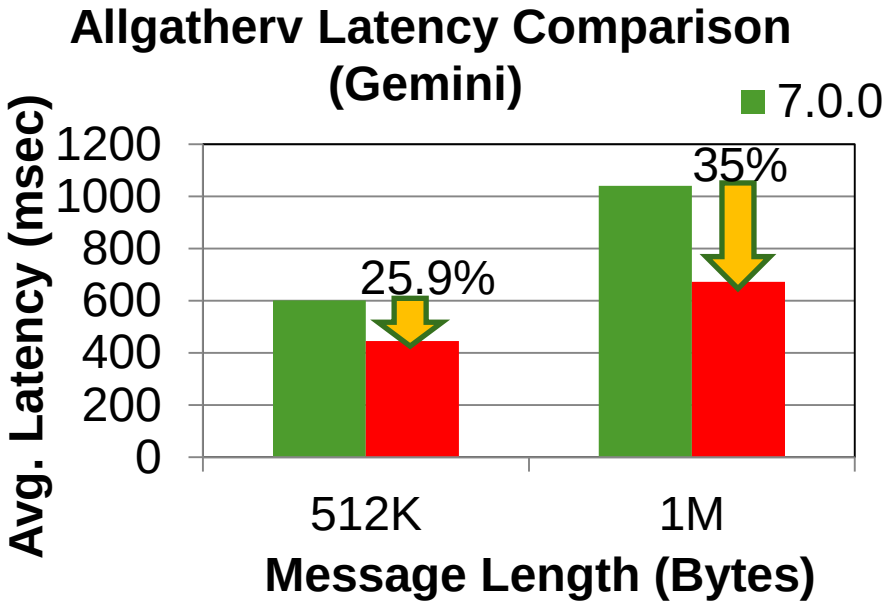
4,096 MPI Processes: 24-Core Ivy-Bridge nodes

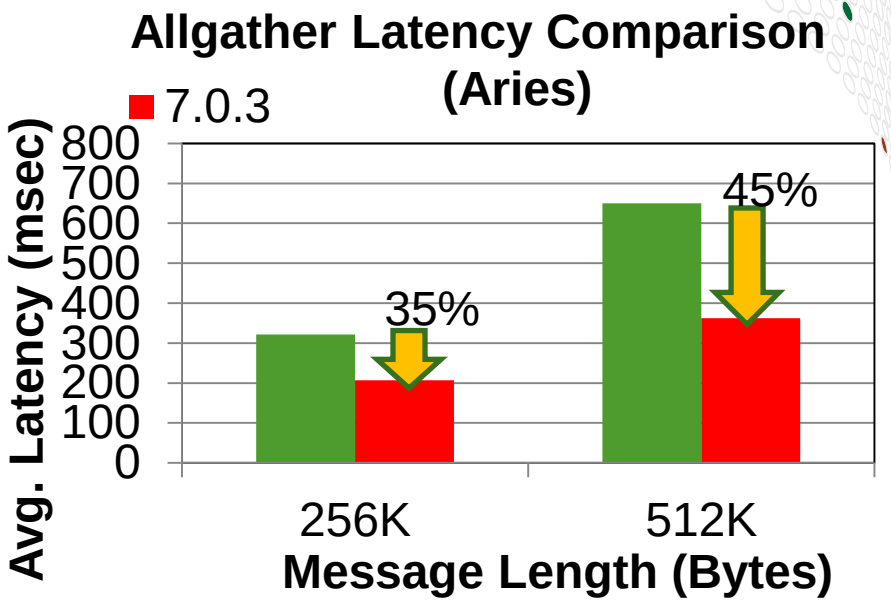
Improved comm./comp. overlap (by up to 80%)

MPICH_NEMESIS_ASYNC_PROGRESS=1;

MPICH_MAX_THREAD_SAFETY=multiple; (CLE 5.2 UP02 or newer)

MPI_Allgather and MPI_Allgatherv





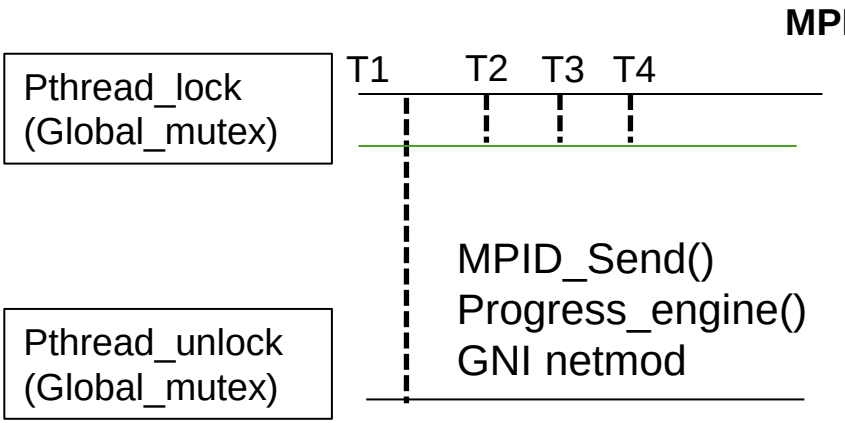
512 MPI Processes:
MPT 7.0.3 performs about 25-35% better.

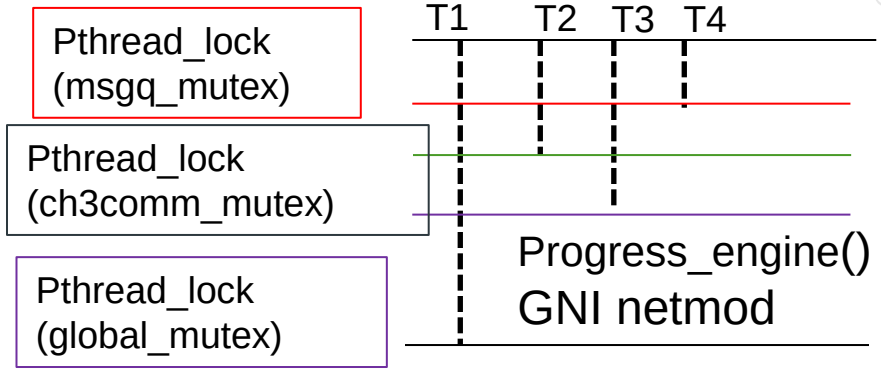
1024 MPI Processes:
MPT 7.0.3 performs about 35-45% better.

Enabled by Default (Aries/Gemini)

Fine-Grained Multi-threading for MPI

Global lock vs Per-Object locks





Default:

- MPI library uses a single global_mutex

  MPI_Send

  pthread_mutex_lock(global_mutex)

  MPID_Send()

  MPID_Progress_wait()

  pthread_mutex_unlock(global_mutex)

Per-Obj (Fine-Grained):

- MPI library still relies on a global_mutex.
  Along with several smaller locks:
  msgq_mutex, handle_mutex, comp_mutex

- Global critical sections are now smaller