How to Use MPI on the Cray XT

Jason Beech-Brandt
Kevin Roy
Cray UK

Outline

- XT MPI implementation overview
- Using MPI on the XT
- Recently added performance improvements
- Additional Documentation

XT MPI implementation overview

- Portals

- MPI implementation

Portals API

- API designed to fit MPI message matching rules
- Emphasis on application bypass, off loading of message passing work from application process
- Emphasis on scalability
- Similar in concept to Quadrics t-ports

XT MPI

- Based on MPICH2
- Cray developed a Portals ADI3 device for MPICH2
  - Portions of design come from earlier MPICH1 portals ADI device
  - Portions from CH3 ADI3 device in MPICH2
- Supports MPI-2 RMA (one-sided)
- Full MPI-IO support
- Does not support MPI-2 dynamic process management (chapter 5 of MPI-2 standard).

Using MPI on XT

- Optimizing MPI point-to-point calls for XT
- MPI derived datatypes
- Collective Operations
- MPI-2 RMA
- Odds and ends
- Environment variable summary
- “What does this mean?”

Optimizing MPI Point-to-point calls(1)

- Use non-blocking send/recvs when it is possible to overlap communication with computation
- If possible, pre-post receives before sender posts the matching send
- Don’t go crazy pre-posting receives though. May hit Portals internal resource limitations.

Optimizing MPI Point-to-point calls(2)

- Normally best to avoid MPI_(I)probe. Eliminates many of the advantages of the Portals network protocol stack.
- No significant performance advantages associated with persistent requests.
- For many very small messages, it may be better to aggregate data to reduce the number of messages
- But don’t aggregate too much. Portals/Seastar ~1/2 of asymptotic bandwidth at ~4-8 KB.

MPI derived datatypes

- XT MPI uses MPICH2 *dataloop* representation of derived data types, shown to be superior to MPICH1, at least for micro-processors
- However, XT hardware not designed to handle non-contiguous data transfers efficiently, still better to use contiguous data types if possible
  - MPI packs data on sender side
  - MPI allocates temporary buffer on receive side and then unpacks data into application receive buffer
  - Opteron more active in sending/receiving data

Collective Operations

- XT MPI uses MPICH2 default collectives with some optimized algorithms enabled by message size (more on this later)
- Environment variables available for additional optimized algorithms
- In some cases it may be better to replace collective operations with point to point communications to overlap communication with computation

XT MPI-2 RMA

- XT MPI supports all RMA operations
- Based on MPICH2 CH3 device RMA
  - Layered on top of internal send/recv protocol
- Designed for functionality, not performance.
- Little opportunity for overlapping of communication with computation when using MPI-2 RMA on XT.
- Almost all communication occurs at end of exposure epochs or in *MPI_Win_free.*

Odds and Ends

- MPI_Wtime is not global
- MPI_LONG_DOUBLE datatype is not supported
- MPI_Send to self will cause application to abort for any message size *(if a matching receive is not pre-posted).*
- Topology-related functions (***MPI_Cart_create***, etc.) are not optimized in current releases

XT3 MPI environment variables – buffer and message size defaults(1)

environment variable	description	default
MPICH_MAX_SHORT_MSG_SIZE	Sets the maximum size of a message in bytes that can be sent via the short(eager) protocol.	128000 bytes
MPICH_UNEX_BUFFER_SIZE	Overrides the size of the buffers allocated to the MPI unexpected receive queue.	60 MB

XT MPI environment variables – maximum number of event defaults(2)

environment variable	description	default
MPICH_PTL_UNEX_EVENTS	Specifies size of event queue associated with unexpected messages. Bear in mind that each unexpected message generates 2 events on this queue.	20480 events
MPICH_PTL_OTHER_EVENTS	Specifies size of event queue associated with handling of Portals events not associated with unexpected messages.	2048 events
MPICH_DBMASK	Set this variable to 0x200 to get a coredump and traceback when MPI encounters errors either from incorrect arguments to MPI calls, or internal resource limits being hit.	not enabled

XT MPI Environment variables – collective algorithms(3)

environment variable	description	default
MPICH_ALLTOALL_SHORT_MSG	Adjusts the cut-off point for which the store and forward Alltoall algorithm is used for short messages	512 bytes
MPICH_BCAST_ONLY_TREE	Setting to 1 or 0, respectively disables or enables the ring algorithm in the implementation for MPI_Bcast for communicators of nonpower of two size.	1
MPICH_REDUCE_SHORT_MSG	Adjusts the cut-off point for which a reduce-scatter algorithm is used. A binomial tree algorithm is used for smaller values.	64K bytes

Recently added XT MPI Performance Improvements

- Portals improvements (In 1.5.07, 1.4.28)
  - Send to self short-circuit optimizations
  - Symmetric portals syscall optimizations
  - Portals API extended (PtlMEMDPost)
- MPI use of PtlMEMDPost (In 1.5.07, 1.4.28)
- New MPI env variables
  - MPICH_RANK_REORDER_METHOD
  - MPI_COLL_OPT_ON
  - MPICH_FAST_MEMCPY

New MPI env variables

- MPICH_RANK_REORDER_METHOD env variable to control rank placement (In 1.5.08 and 1.4.30)
  - Setting env to “0” gives round-robin (default yod placement):

NODE	0	1	2	3
RANK	0&4	1&5	2&6	3&7

  - Setting env to “1” causes SMP style placement (default aprun placement)

NODE	0	1	2	3
RANK	0&1	2&3	4&5	6&7

MPICH_RANK_REORDER_METHOD (cont.)

- Setting env to “2” causes folded rank placement

NODE	0	1	2	3
RANK	0&7	1&6	2&5	3&4

- Setting env to “3” causes custom rank placement using “MPICH_RANK_ORDER” file. For example:

0-15	Places the ranks in SMP-style order
15-0	Places ranks 15&14 on the first node, 13&12 on next, etc.
0,4,1,5,2,6,3,7	Places ranks 0&4 on the first node, 1&5 on the next, 2&6 together, and 3&7 together.

  - MPICH_RANK_FILE_BACKOFF

    Specifies the number of milliseconds for backoff.
  - MPICH_RANK_FILE_GROUPSIZE

    Specifies the number of ranks in the group size.

NOTE: Setting PMI_DEBUG will display rank information to stdout

New MPI env variables (cont.)

- MPI_COLL_OPT_ON multi-node collective optimizations (In 1.5.11 and 1.4.32)
  - MPI_Allreduce 30% faster for 16K bytes or less (Pallas 256pes)
  - MPI_Barrier - 25% faster (Pallas 256pes)

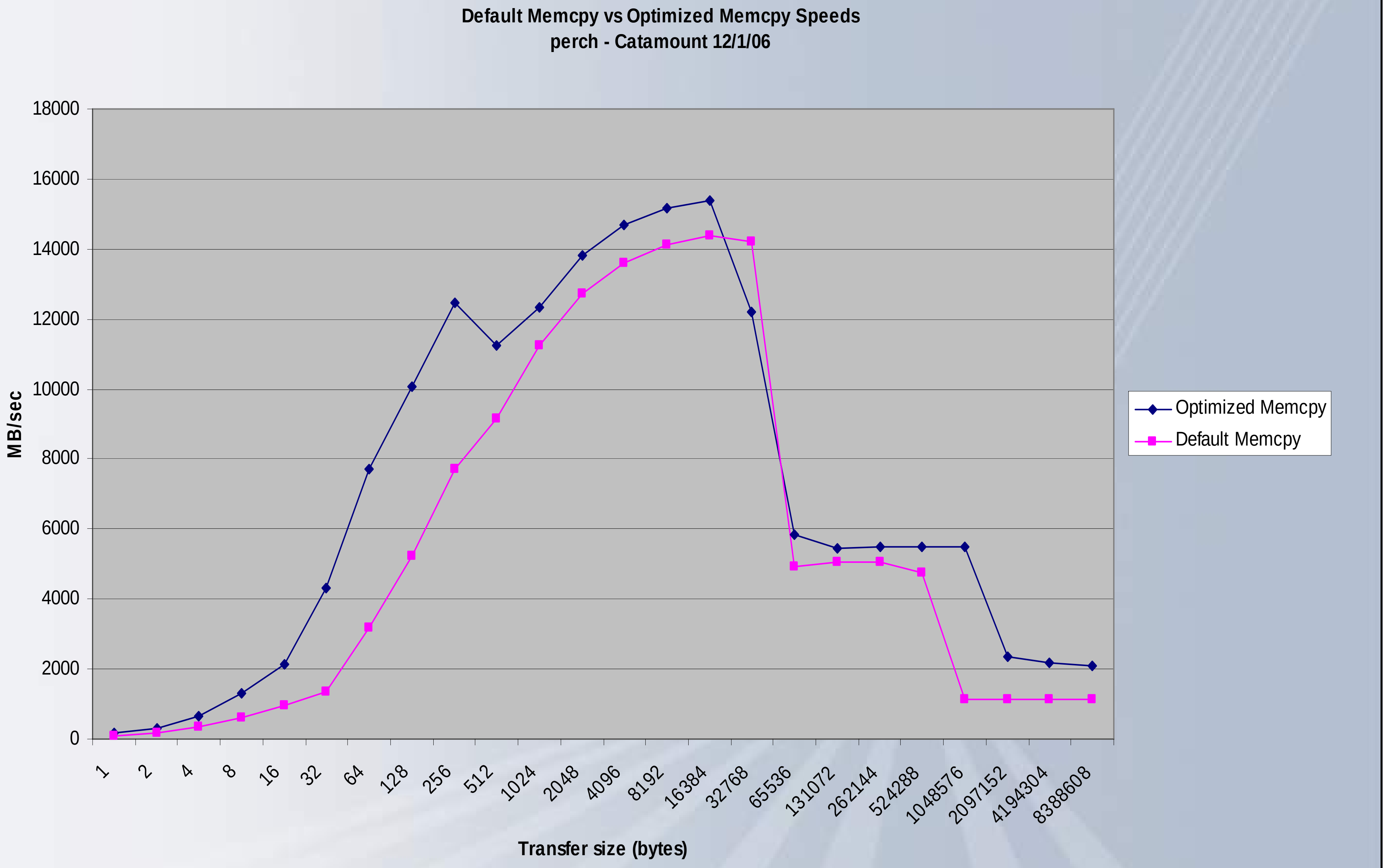
New MPI env variables (cont.)

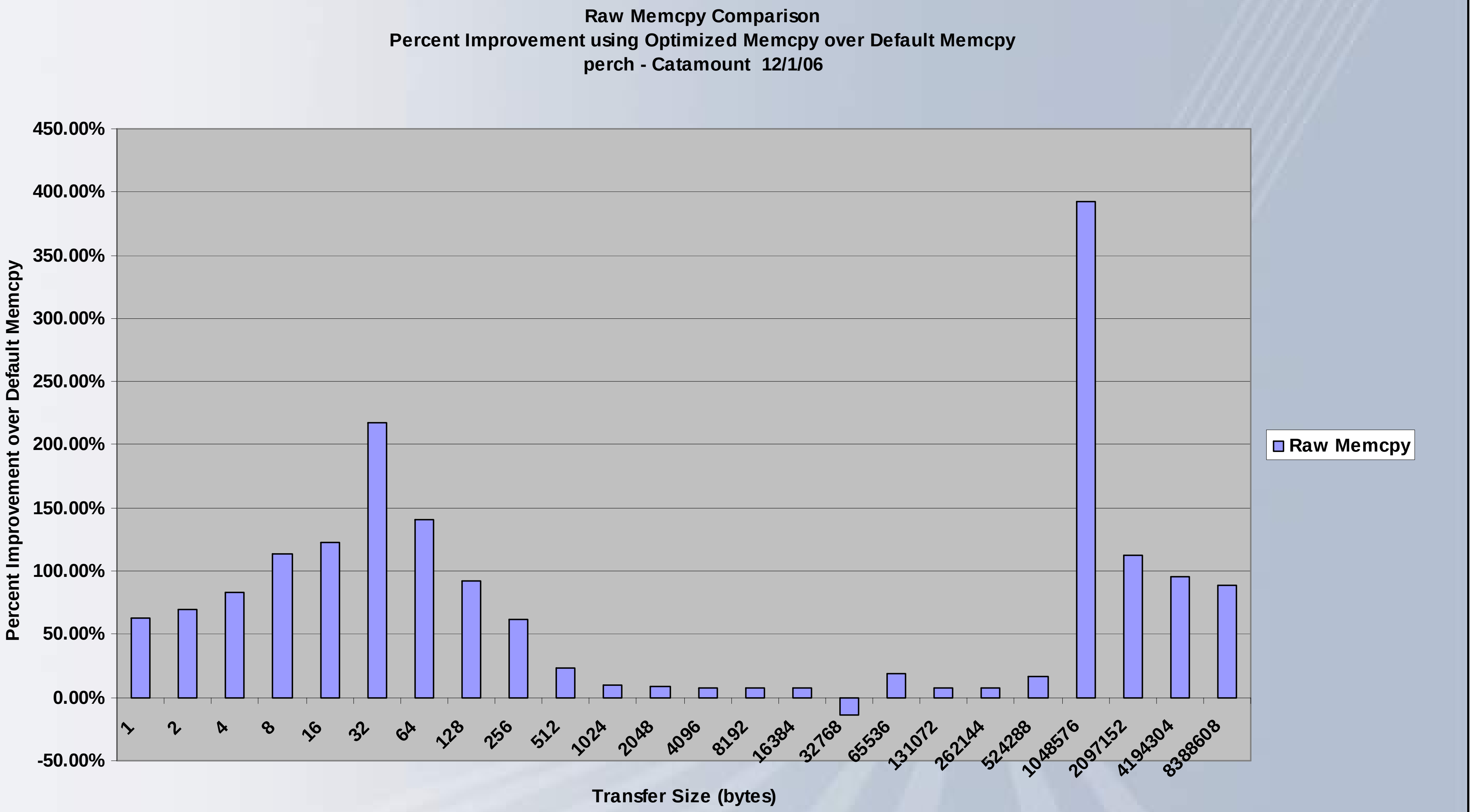
- Added fast memcpy (MPICH_FAST_MEMCPY)

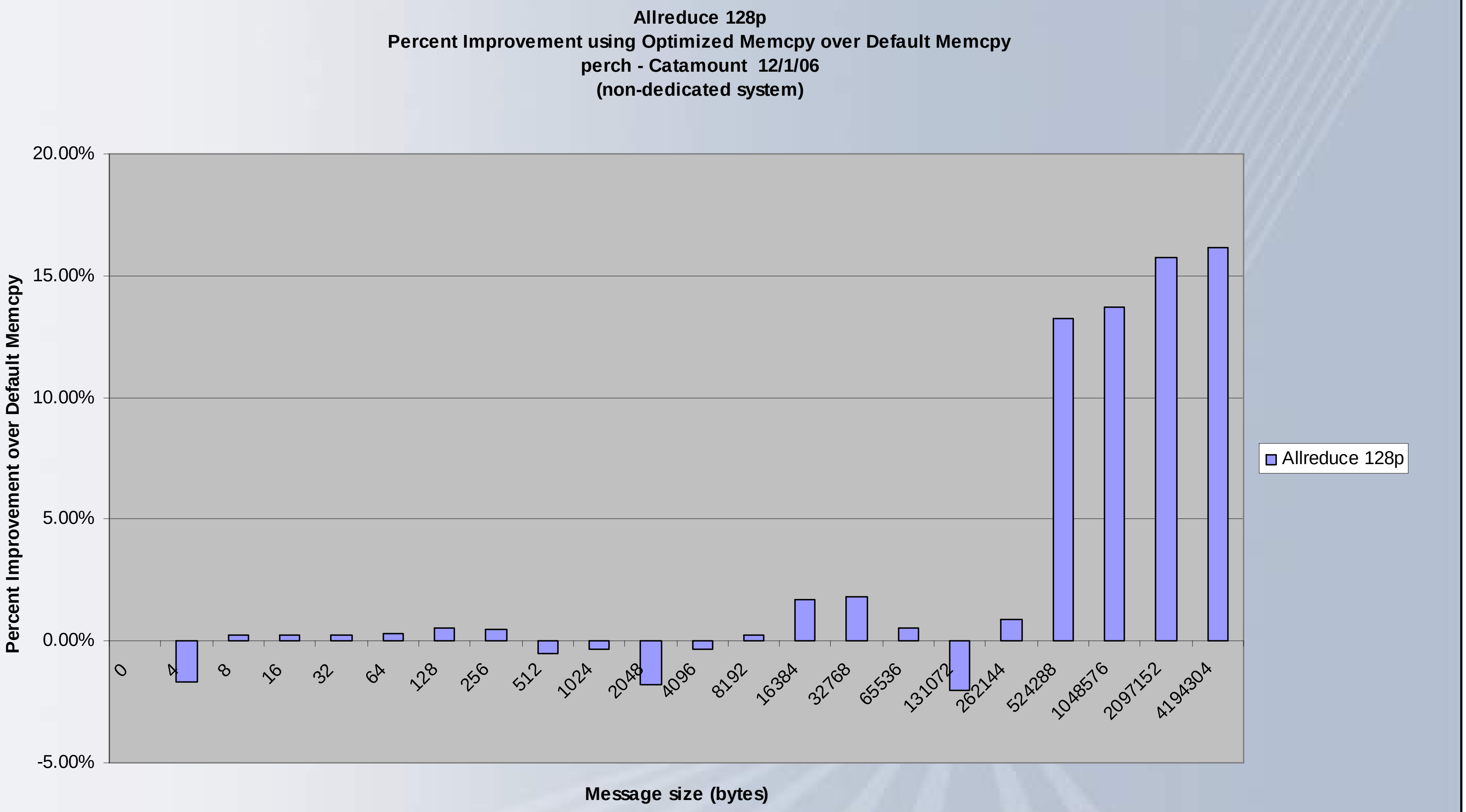
  New improved memcpy used within MPI for local copies for pt2pt and collectives.

  (In 1.5.30 and 1.4.46)

  - Many collectives 8-20% faster above 256K bytes


Default Memcpy vs Optimized Memcpy Speeds
perch - Catamount 12/1/06
MB/sec
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
4194304
8388608
Transfer size (bytes)
Optimized Memcpy
Default Memcpy


Raw Memcpy Comparison
Percent Improvement using Optimized Memcpy over Default Memcpy
perch - Catamount 12/1/06
450.00%
400.00%
350.00%
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
-50.00%
Percent Improvement over Default Memcpy
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
4194304
8388608
Transfer Size (bytes)
■ Raw Memcpy


Allreduce 128p
Percent Improvement using Optimized Memcpy over Default Memcpy
perch - Catamount 12/1/06
(non-dedicated system)
20.00%
15.00%
10.00%
5.00%
0.00%
-5.00%
Percent Improvement over Default Memcpy
0
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
4194304
Message size (bytes)
Allreduce 128p

XT Specific MPI documentation

- Man pages
  - intro_mpi
  - aprun
- Cray XT Programming Environment User's Guide

Portals related documentation

- Paper by Brightwell (Sandia), et al. about Portals on XT3 (Red Storm)
  - http://gaston.sandia.gov/cfupload/ccim_pubs_prod/Brightwell_paper.pdf
- Portals project on source forge
  - http://sourceforge.net/projects/sandiaportals/