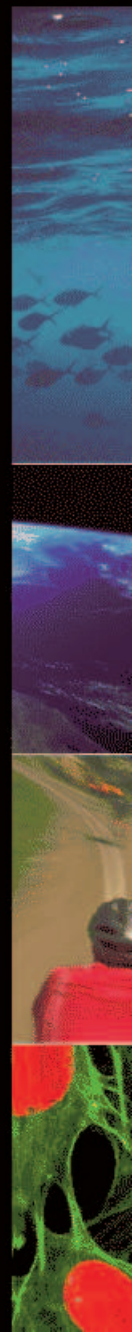
The Supercomputer Company

Message Passing Toolkit(MPT) on XT3

- CUG 2006

Howard Pritchard
(howardp@cray.com)

CRAY USER GROUP INCORPORATED

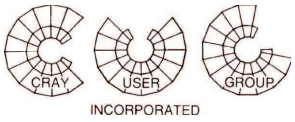


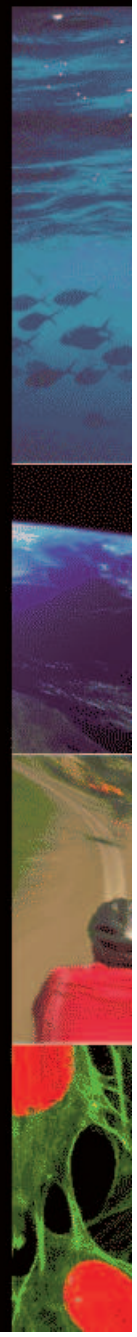
Outline

- XT3 MPI implementation overview
- Using MPI on XT3
- SHMEM on XT3
- Upcoming Features
- Documentation

XT3 MPI Implementation Overview

- CUG 2006





XT3 MPI implementation overview

- Portals
- MPI implementation

Portals

- Designed to support MPI message matching rules
- Emphasis on application bypass, off loading of message passing work from compute processor
- Emphasis on scalability
- Similar in concept to Quadrics t-ports

XT3 MPI (1)

- 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

XT3 MPI(2)

- Supports MPI-2 RMA (one-sided)
- Full MPI-IO support starting with release 1.3

- Currently only supported on compute nodes
- Does not support MPI-2 dynamic process management (chapter 5 of MPI-2 standard).

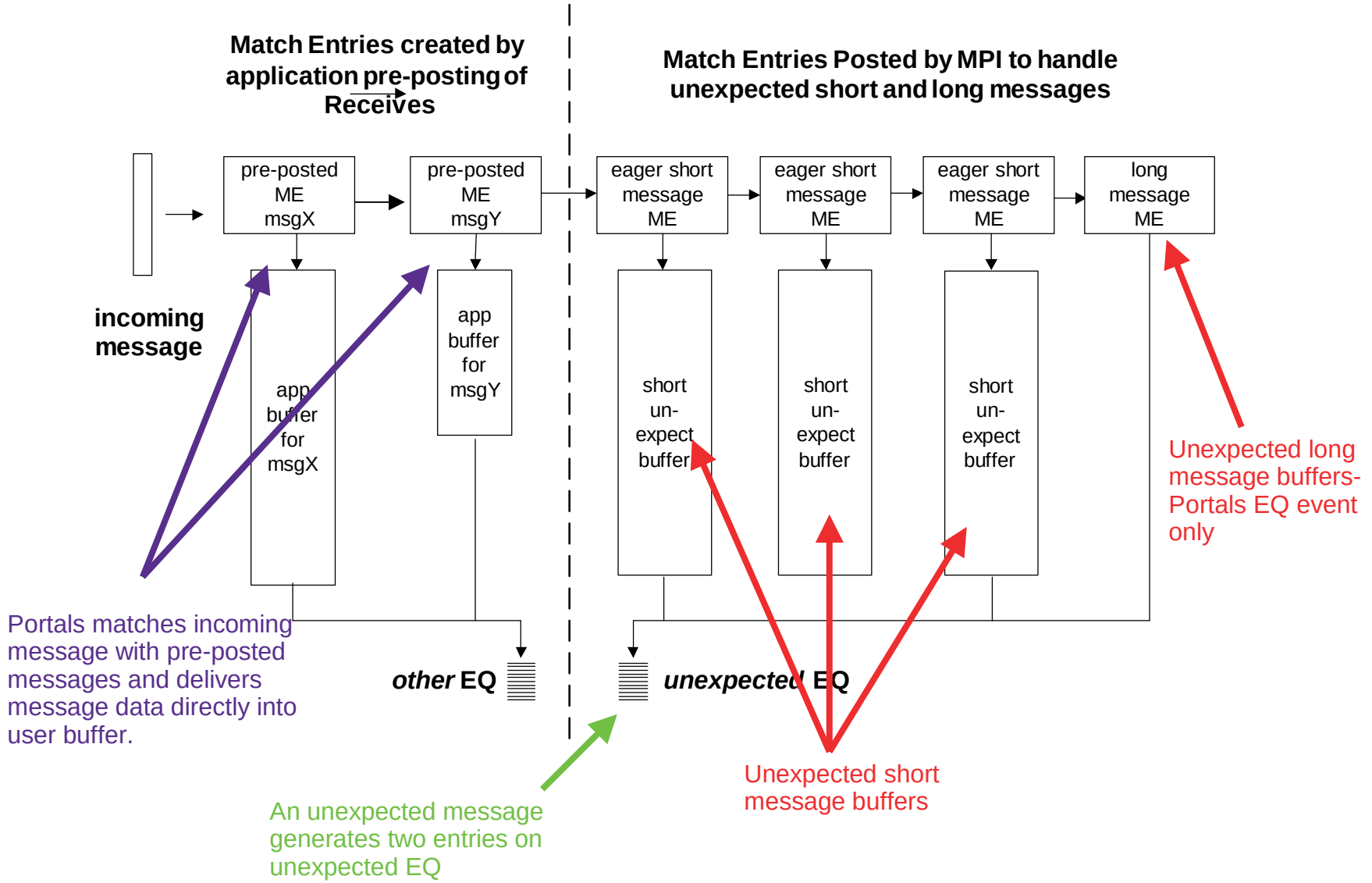
XT3 MPI Implementation(1)

Two protocols are used for handling basic MPI-1 send/recv style messaging:

- Eager protocol for short messages
- Two protocols for long messages

But first we will talk about the receive side, since that is where Portal's important features with respect to MPI are most evident…

XT3 MPI – Receive Side



XT3 MPI Short Message Protocol- Sending side

- If message to send has no more than 128000 bytes of data, the short message, eager protocol is used by sender.
- Sender assumes the receiver has space for the message in an MPI-internal receive buffer and 2 slots in the *unexpected Event Queue.*
- For very short messages (1024 bytes or shorter), sender copies data to internal buffer before sending *PtlPut* request to Portals.

XT3 MPI Short Message Protocol- Receive side

Two things can happen:

- No matching receive is posted when message arrives. In this case the message goes into one of the unexpected buffers, later to be copied into application buffer. Two events are delivered to the *unexpected* event queue.
- A matching receive was posted. In this case the incoming message data goes directly into the application buffer. An event is delivered to the *other* event queue.

XT3 MPI Long Message Protocol

Two Protocols starting with XT3 MPT 1.3:

- Receiver-pull based method (default)
- Eager method

XT3 MPI Long Message Protocol- Receiver Pull method

- Sender encodes information in a Portals Header with information for receiver to get the data from the application buffer
- Sender sends just this Portals header.
- Receiver picks up the Portals header and decodes it. If matching receive already posted, GET the data using PtlGet to store directly in receiver application buffer.
- If no buffer posted, just leave on internal unexpected list till matching receive is posted.

XT3 MPI Long Message Protocol-Eager Send Method(1)

- Sender sends the message data much as with the short protocol. It is assumed that there are sufficient resources at the receiver (slots in *unexpected Event Queue*). Sender requests a PTL_ACK_REQ.
- If there is no matching posted receive, the message header is matched to the long protocol match entry.

XT3 MPI Long Message Protocol-Eager Send Method(2)

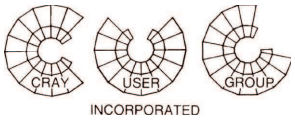
- If there is a matching posted receive Portals delivers the message data directly into the application buffer and replies to the sender with a PTL_EVENT_ACK. This tells the sender that the send is complete.
- Otherwise, when the matching receive is posted by the application, the receiver GETs the data from the source, much like in the *receiver-pull* method. The PtlGET generates a *PTL_EVENT_REPLY_END* at the sender. This tells the sender the send is complete.

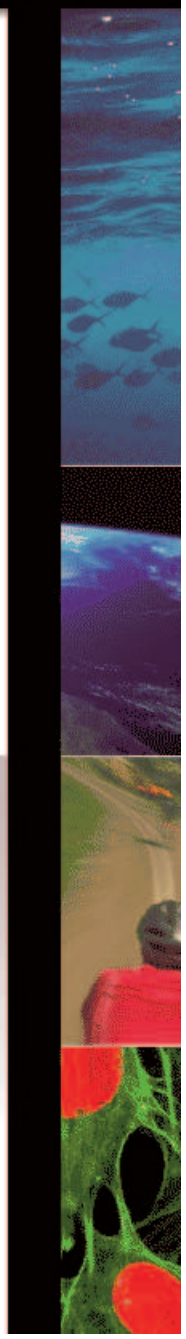
XT3 MPI Long Message Protocol- Why Two Methods

- COP (CAM OVERFLOW PROTECTION) causes significant delays in returns from blocking sends, causing noticeable bandwidth drops using *eager long* method (default in 1.2 release)
- Applications that insure receives are pre-posted should use the *eager* method. This is done by setting the MPICH_PTL_EAGER_LONG environment variable.

Using MPI on XT3

- CUG 2006





Using MPI on XT3

- Optimizing MPI point-to-point calls for XT3
- MPI derived datatypes
- Using MPI 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

- XT3 MPI uses MPICH2 *dataloop* representation of derived data types, shown to be superior to MPICH1, at least for micro-processors
- However, XT3 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(1)

- XT3 MPI mainly uses MPICH2 default collectives
- For MPT 1.4 release, some defaults for switching algorithms (more on this later)
- In some cases it may be better to replace collective operations with point to point communications to overlap communication with computation

XT3 MPI-2 RMA

- XT3 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 XT3.
- 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
- Topology-related functions (***MPI_Cart_create***, etc.) are not optimized in current releases

XT3 MPI environment variables(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
MPICH_PTL_EAGER_LONG	Enables eager long path for message delivery.	disabled

XT3 MPI environment variables(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_MAX_VSHORT_MSG_SIZE	Specifies in bytes the maximum size message to be considered for the vshort path.	1024 bytes

XT3 MPI environment variables(3)

environment variable	description	default
MPICH_VSHORT_BUFFERS	Specifies the number of 16384 byte buffers to be pre-allocated for the send side buffering of messages for the vshort protocol.	32 buffers
MPICH_DBMASK (1.4 release) MSMDB_MASK in earlier releases.	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

What does this mean? (1)

If you see this error message:

```
Assertion failed in file
/notbackedup/users/rsrel/rs64.REL_1_4_06.060419.Wed/pe/computelibs/mpich2/src/mpid/portals32/src/portals_init.
c at line 193: MPIDI_Portals_unex_block_size > MPIDI_Portals_short_size
```

It means:

The appearance of this assertion means that the size of the unexpected buffer space is too small to contain even 1 unexpected short message.

Try doing this to work around the problem:

User needs to check their MPICH environment settings to make sure there are no conflicts between the setting of the MPICH_UNEX_BUFFER_SIZE variable and the setting for MPICH_MAX_SHORT_MSG_SIZE. Note setting MPICH_UNEX_BUFFER_SIZE too large (> 2 GB) may confuse MPICH and also lead to this message.

What does this mean? (2)

If you see this error message:

```
internal ABORT - process 0: Other MPI error, error stack:
MPIDI_PortalsU_Request_PUPE(317): exhausted unexpected receive queue buffering increase via env.

var. MPICH_UNEX_BUFFER_SIZE
```

It means:

The application is sending too many short, unexpected messages to a particular receiver.

Try doing this to work around the problem:

Increase the amount of memory for MPI buffering using the MPICH_UNEX_BUFFER_SIZE variable(default is 60 MB) and/or decrease the short message threshold using the MPICH_MAX_SHORT_MSG_SIZE (default is 128000 bytes) variable. May want to set MPICH_DBMASK to 0x200 to a traceback/coredump to learn where in application this problem is occurring. MSMDB_MASK in pre 1.4 MPT releases.

What does this mean? (3)

If you see this error message:

```
[0] MPIDI_PortalsU_Request_FDU_or_AEP: dropped event on unexpected receive queue, increase
[0] queue size by setting the environment variable MPICH_PTL_UNEX_EVENTS
```

It means:

You have exhausted the event queue entries associated with the unexpected queue. The default size is 20480.

Try doing this to work around the problem:

You can increase the size of this queue by setting the environment variable MPICH_PTL_UNEX_EVENTS to some value higher than 20480.

What does this mean? (4)

If you see this error message:

```
[0] MPIDI_Portals_Progress: dropped event on "other" queue, increase
[0] queue size by setting the environment variable MPICH_PTL_OTHER_EVENTS

aborting job: Dropped Portals event
```

It means:

You have exhausted the event queue entries associated with the “other” queue. This can happen if the application is posting many non-blocking sends, or a large number of pre-posted receives are being posted, or many MPI-2 RMA operations are posted in a single epoch. The default size of the other EQ is 2048.

Try doing this to work around the problem:

You can increase the size of this queue by setting the environment variable MPICH_PTL_OTHER_EVENTS to some value higher than the 2048 default.

What does this mean? (5)

If you see this error message:

```
0: (/notbackedup/users/rsrel/rs64.REL_1_3_12.051214.Wed/pe/computelibs/mpich2/src/mpid/
    portals32/src/portals_progress.c:642) PtlEQAlloc failed : PTL_NO_SPACE
```

It means:

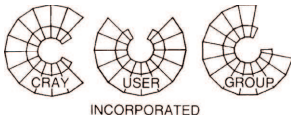
You have requested so much EQ space for MPI (and possible SHMEM if using both in same application) that there are not sufficient Portals resources to satisfy the request.

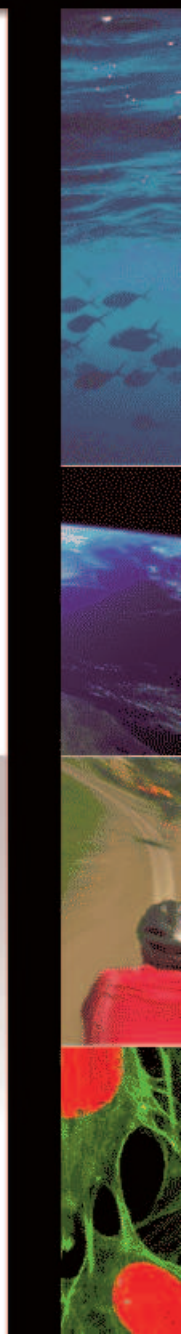
Try doing this to work around the problem:

You can decrease the size of the event queues by setting the environment variable MPICH_PTL_UNEXPECTED_EVENTS and MPICH_PTL_OTHER_EVENTS to smaller values.

SHMEM ON XT3

- CUG 2006





SHMEM on XT3

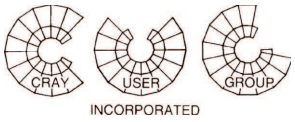
- Programming Model close to that of T3E, but only data segment and symmetric heap are remotely accessible.
- Like MPI, SHMEM is layered directly on top of Portals.

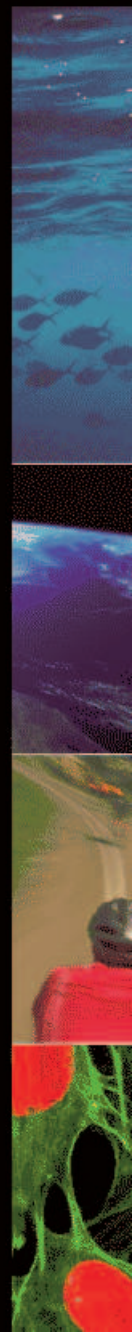
Good Practices for SHMEM

- Use non-blocking SHMEM operations if possible (shmem_put_nb support in release 1.4)
- Shmem barriers performed in software and have high overhead, so use with care
- Use shmem_fence rather than shmem_quiet where possible
- Don’t use strided SHMEM operations in performance critical sections of application

Upcoming and Future Feature Work for XT3 MPT

- CUG 2006





MPI Features for 1.4 release

- Improvements to parameters for current MPICH2 collective algorithms.
- Bug fixes.

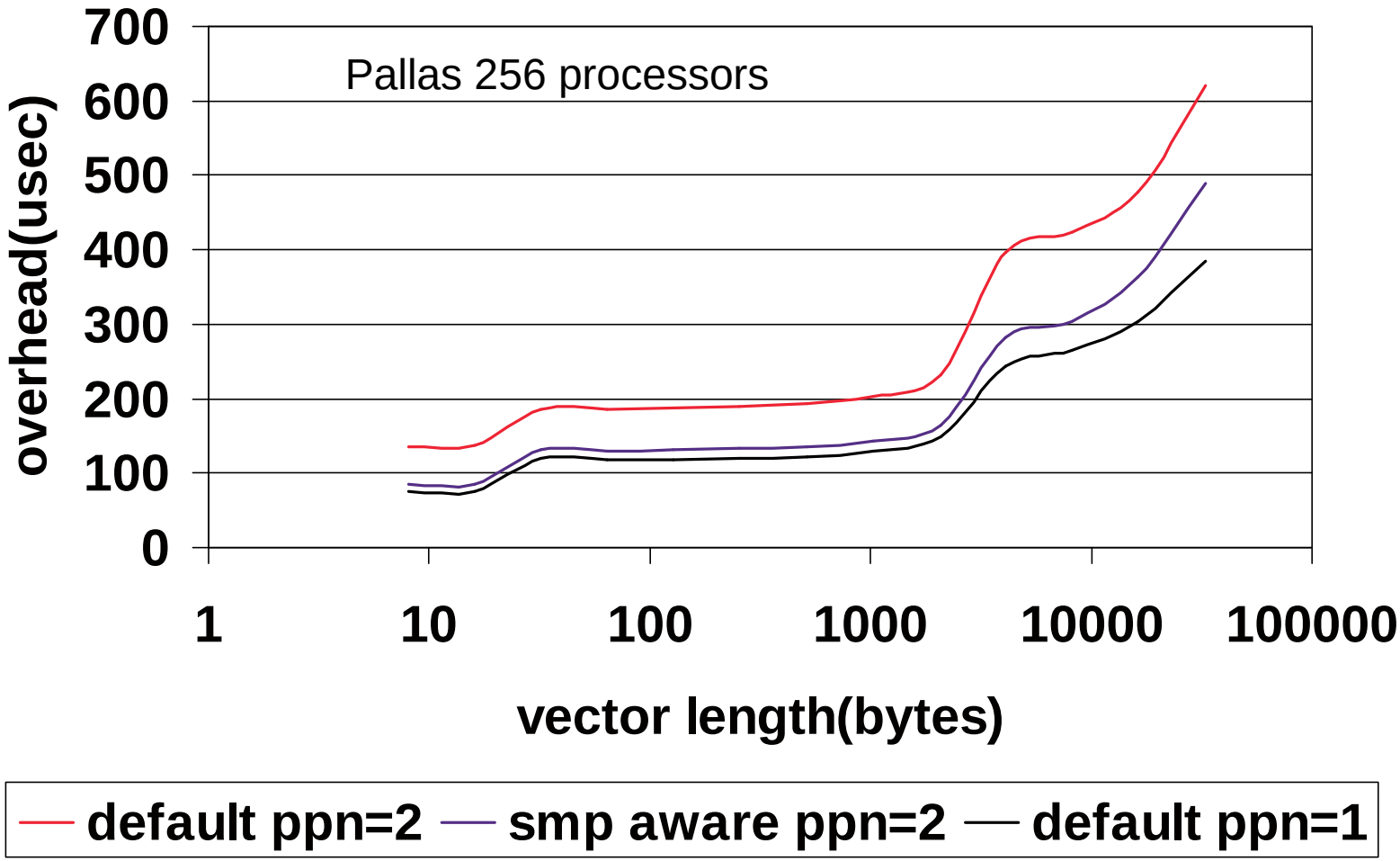
XT3 MPI Environment variables (release 1.4)

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
MPICH_ALLTOALLVW_SENDRECV	Disables the flow-controlled Alltoall algorithm. When disabled, the pair-wise sendrecv algorithm is used which is the default for messages larger than 32K bytes.	not enabled

MPI Features for future releases

- SMP-node aware collective algorithms.
- Critical bug fixes.
- Short message latency improvements, possibly including modifications for better use of accelerated mode portals firmware.

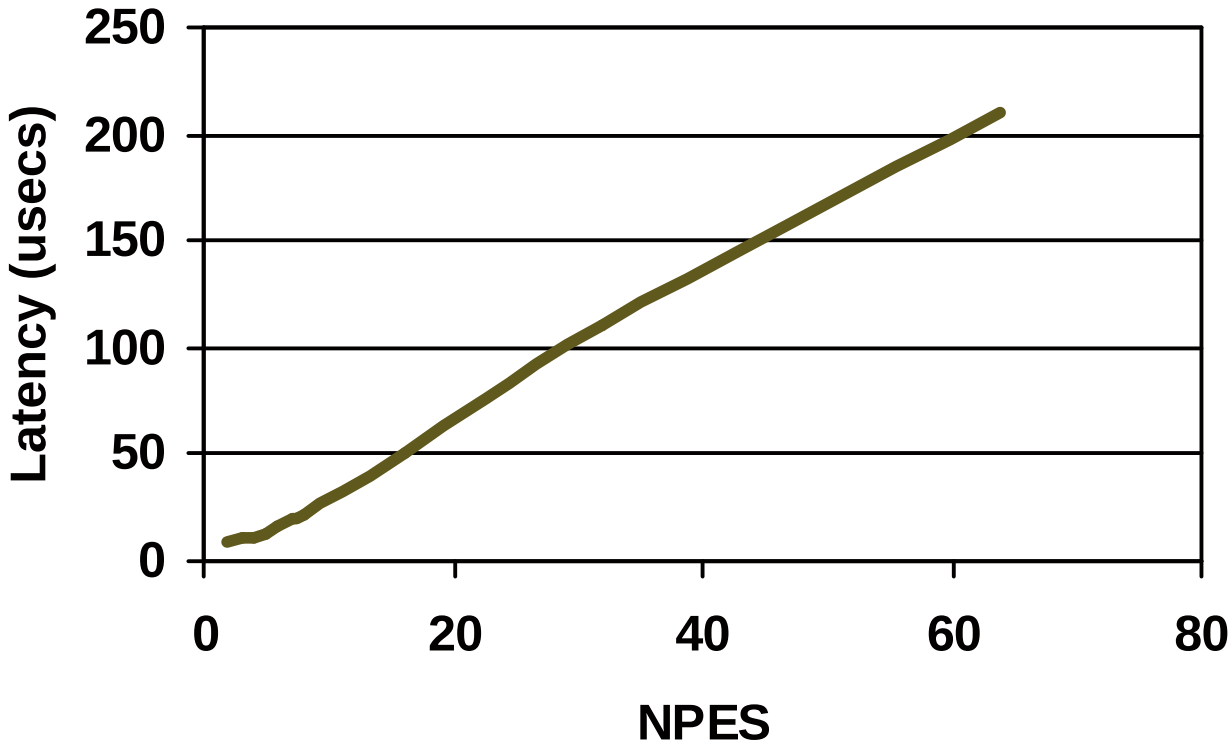
SMP-topology aware Allreduce



SHMEM Features and added functionality for 1.4 release

- Reduction functions over subsets. Improvements in reduction performance.
- Atomic Op functions supported in 1.4 release
  - shmem_XXX_swap
  - shmem_XXX_finc/fadd
  - shmem_XXX_lock
  - Implemented in software(portals) not hardware
- Non-blocking shmem PUT. Implicit non-blocking, use *shmem_quiet,* an atomic op, or *shmem_barrier_all* to insure completion of put.

SHMEM Atomic Operations



Shmem_fadd latency as a function of PES updating single variable

XT3 Specific MPI/SHMEM documentation

- Man pages
  - intro_mpi
  - intro_shmem
  - yod
- Cray XT3 Programming Environment User's Guide (not yet released, contact local Cray representative for a copy)

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/

Online MPI Materials

- Argonne National Laboratory:
  - MPI Learning collection:
    - www-unix.mcs.anl.gov/mpi/learning.html
      - » Pointers to MPI tutorials and other documents from Argonne and other sources, including: www-unix.mcs.anl.gov/mpi/tutorial/index.html
    - www-unix.mcs.anl.gov/mpi/tutorial/perf/index.html
      - » Details on MPI tuning and performance
    - www-unix.mcs.anl.gov/dbpp/
      - » *Designing and Building Parallel Programs*, by Ian Foster
    - www-unix.mcs.anl.gov/romio
      - » *A High-Performance, Portable MPI-IO Implementation*
  - MPI tools collection:
    - www-unix.mcs.anl.gov/mpi/tools.html

Online MPI Materials (cont.)

- Lawrence Livermore National Laboratory
  - Training collection:
    - www.llnl.gov/computing/training/
    - Tutorials and Documentation collection:
    - www.llnl.gov/computing/mpi/documentation.html
- MPI standards
  - MPI-1 www.mpi-forum.org/docs/mpi-11html/node182.html
  - MPI-2 www.mpi-forum.org/docs/mpi-20-html/node306.html