CRI/EPCC MPI for T3D
Implementation Overview

A. Gordon Smith

smith@epcc.ed.ac.uk

Edinburgh Parallel Computing Centre (EPCC)

The University of Edinburgh

Overview

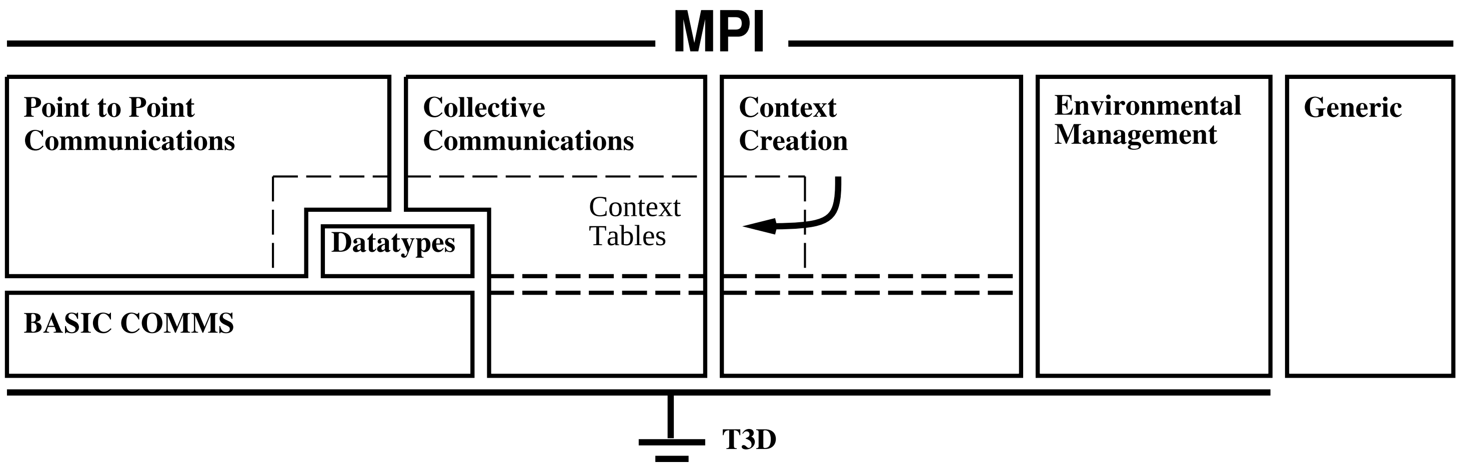
- Background
- Aims and Overview
- Basic communication
- Point-to-point
- Collective
- Context creation
- Generic

Background

Early	**1991**	CHIMP Development
Dec	**1992**	MPI Forum Involvement
Early	**1994**	Collaboration with CRI (UK)
Apr	**1994**	320-PE Cray T3D Sited at EPCC
May	**1994**	Design
Sep	**1994**	Implementation
Dec	**1994**	Testing
May	**1995**	Release

T3D MPI Project Aims

- Basis on Shared Memory Access and Intrinsics
- Full MPI Interface Coverage
- Efficient Point-to-Point Communications
- Efficient Collective Communications
- Contribution to MPI Effort:
  provision, acceptance, experience



T3D MPI Implementation Overview

- Basic Communications (BASIC COMMS)
- Datatype and Buffer Management
- MPI Services by Chapter
- Generic MPI Services

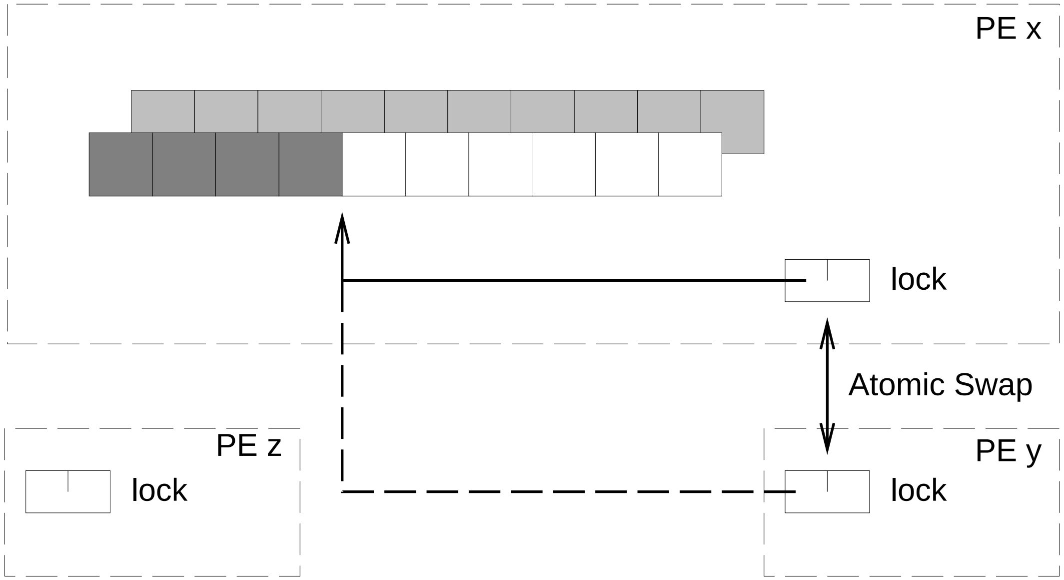
Basic Communications

- Protocol Message Transfer: queue(s)
- Sender Data Transfer: `shmem_put`
- Receiver Data Transfer: `shmem_get`
- Cache Coherency Support

Protocol Message Transfer

- Speculative transfer
- Fixed-size message
- Message ordering
- Cache Coherency Support: cache-line messages

⇒ **Queue**: per process, shared access



Shared Message Queue with Lock

- Mutual Exclusion on Lock using `_atomic_swap` intrinsic
- Minimal Critical Section by double-buffering
- Message Ordering among all processes
- Contention among all processes

Basic Communications Interface

```
BC_INIT(void)
BC_FINALIZE(void)
-----------------------------------------------------------------------
BC_PUTP(int pe, void *p, int *ok)
BC_PUTPM(int pe, void *p, void *buf, int size, int *ok)
-----------------------------------------------------------------------
BC_GETD(int pe, void *lbuf, void *rbuf, int size)
BC_PUTD(int pe, void *lbuf, void *rbuf, int size)
-----------------------------------------------------------------------
BC_ABSORB(void)
=======================================================================
BCC_HANDLER(void *p, int count)
```

Semantics: BC_ABSORB and BCC_HANDLER

```
BC_ABSORB()
    acquire Local Mutex Lock    [safe: Q2,  active: Q1]
(*  switch  Shared Queues  *)   [safe: Q1,  active: Q2]
    return  Local Mutex Lock

    p_base = head Q1
    count  = size Q1

    if (count > 0)
        BCC_HANDLER(p_base, count)

    [Q1 empty]
```

Point-to-Point Communications

- Communication Event Notices
- Message Selection: $(source, tag, comm)$ matching
- Protocol Transfer-Response: BASIC COMMS
- Protocols: message size, transfer mode, cancellation
- Sender/Receiver Data Transfer `BC_{PUTD,GETD}`
- Semantics: order, progress, fairness, resource limits

Event Notice: Comms Characterisation

Operation	$\{send, recv\}$	sender or receiver specific
Locality	$\{local, alien\}$	from call, or via protocol
Context	$[0, 2^{31} - 1)$	context value ("hyper-tag")
Source	$[0, N_{pes})$	PE of counterpart process
Tag	$[0, 2^{15} - 1)$	tag value
(Data/Descriptor)	–	data, or message layout
(Reference)	address	Ref to dangling event notice
Link/Match	address	Queue link, or ref to match

Message Selection

- Event Notice Match:

  Locality/Operation: ($local \leftrightarrow alien$) and ($send \leftrightarrow recv$)
  Contexts match
  Ranks match, or wildcarded (`MPI_ANY_SOURCE`)
  Tag values match, or wildcarded (`MPI_ANY_TAG`)

- Matching Queues: $context \times \{local, alien\} \times \{send, recv\}$

- Match Operations:

  couple-add: find-remove match, or add
  couple-fail: find-remove match, or fail
  match-fail: find match, or fail
  couple-seq: find-remove by sequence (cancel)

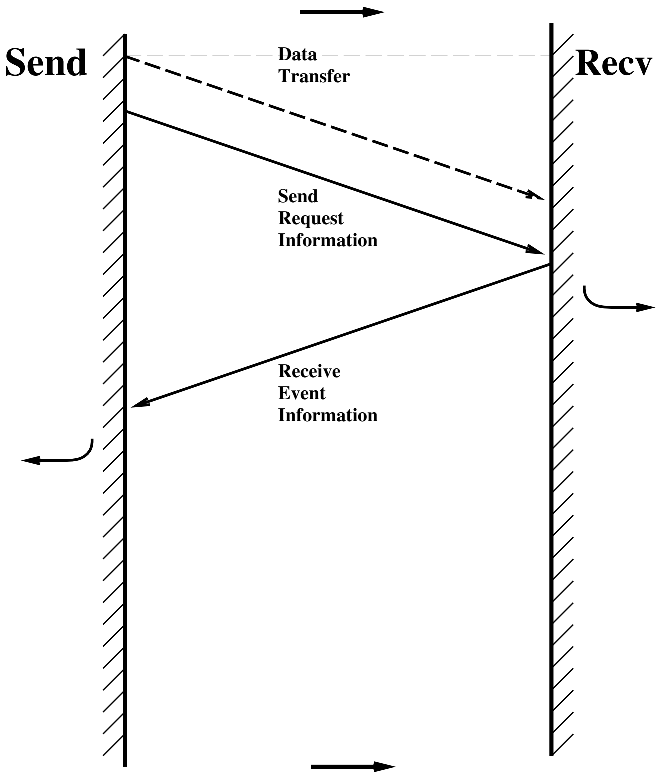
- T3D PVM: single queue

Message Buffering

- Buffered *vs.* Synchronous: send completion
- **Standard/Buffered** mode: allocate buffer and pack message at sender
- **Standard** mode $\rightarrow$ Synchronous, no buffer space
- **Buffered** mode $\rightarrow$ `MPI_ERR_BUFFER`, no buffer space
- **Ready** mode: no optimisation, $\equiv$ Standard mode
- **Synchronous** mode: no message buffering
- Derived Datatypes: buffering necessary in places

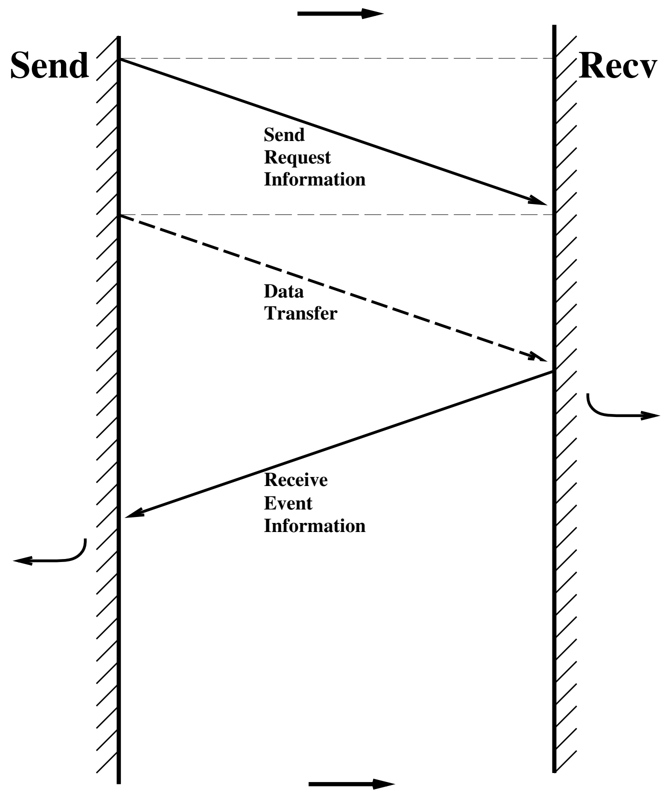
Protocol Transfer/Response

- Transfer: `BC_PUTP` from call
- Response: `BC_ABSORB` from call –
  - Read: `BCC_HANDLER`
  - Reply: `BC_PUTP`
- Immediate,Test,Probe: single `BC_ABSORB`
- Blocking: `BC_ABSORB` loop, polling completion status
- False Synch: protocol progress $\leftrightarrow$ MPI call pattern



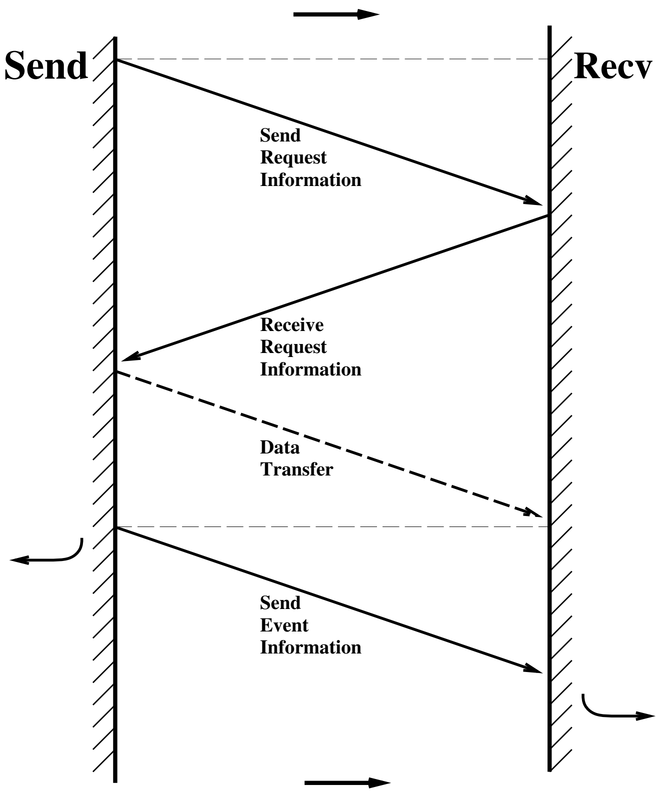
Protocol:
Small Message (TA)

- Sender Request
- Transfer with Request
- Receiver Copy
- Receiver Acknowledge



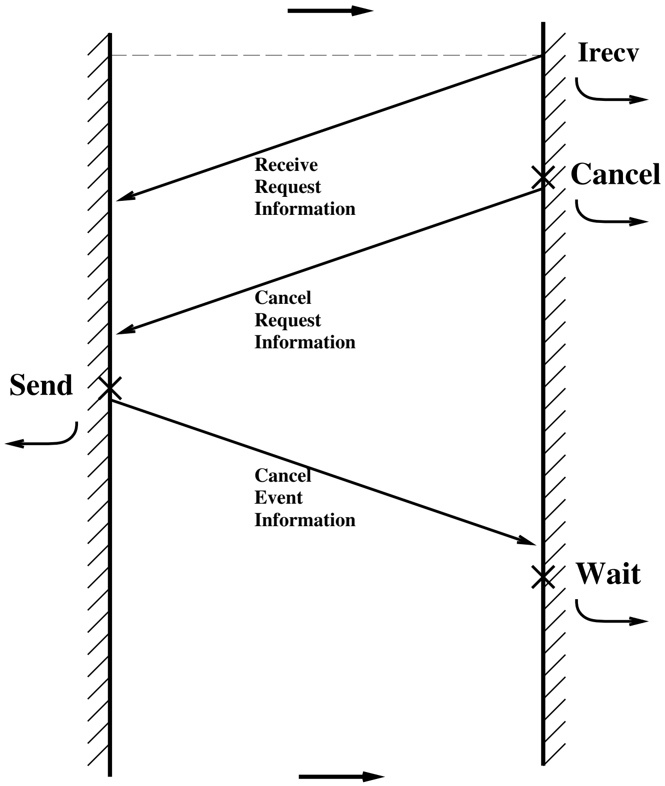
Protocol:
Standard/Buffered (RTA)

- Sender Request
- Receiver Transfer
- Receiver Acknowledge



Protocol:
Synchronous (RAT)

- Sender Request
- Receiver Acknowledge
- Sender Transfer
- Sender Acknowledge



Protocol: **Cancellation**

- Local Cancellation
- Commitment → Fail
- Anti-Request Message
- Confirmation/Transfer

Collective Communications

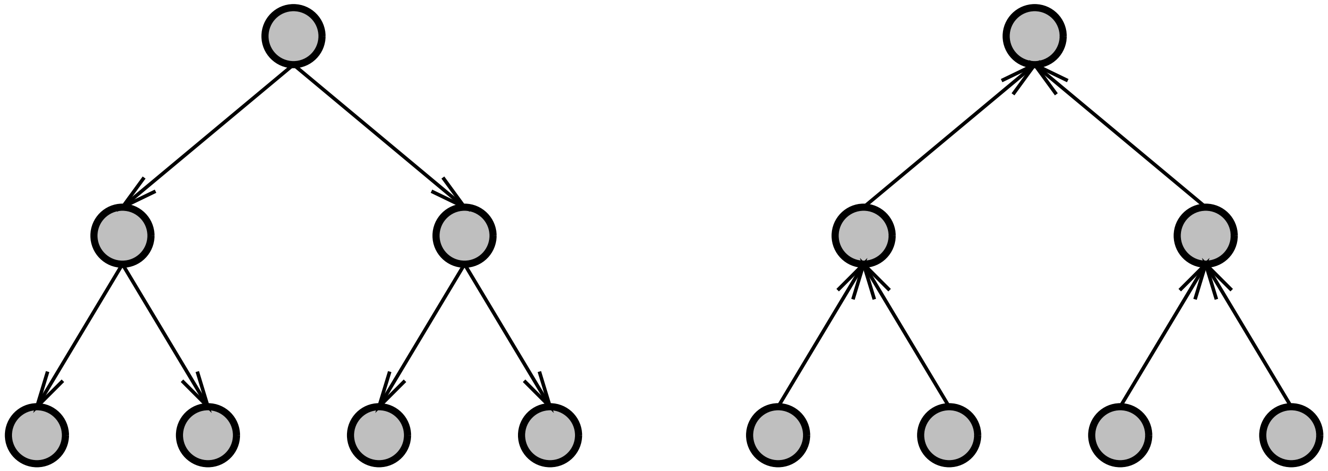
- Basic Collective Mechanism
- Hardware Barrier Optimisation
- Collective `shmem_` not used
- Broadcast/Barrier: Tree Communications
- Gather/Scatter: all-to-one, one-to-all
- All-to-all: Butterfly Network
- Scan: Parallel Prefix

Basic Collective Mechanism

- General BASIC COMMS Protocol not used
- Set of {Up, Down} "Protocol" Slots
- Slot Operations:

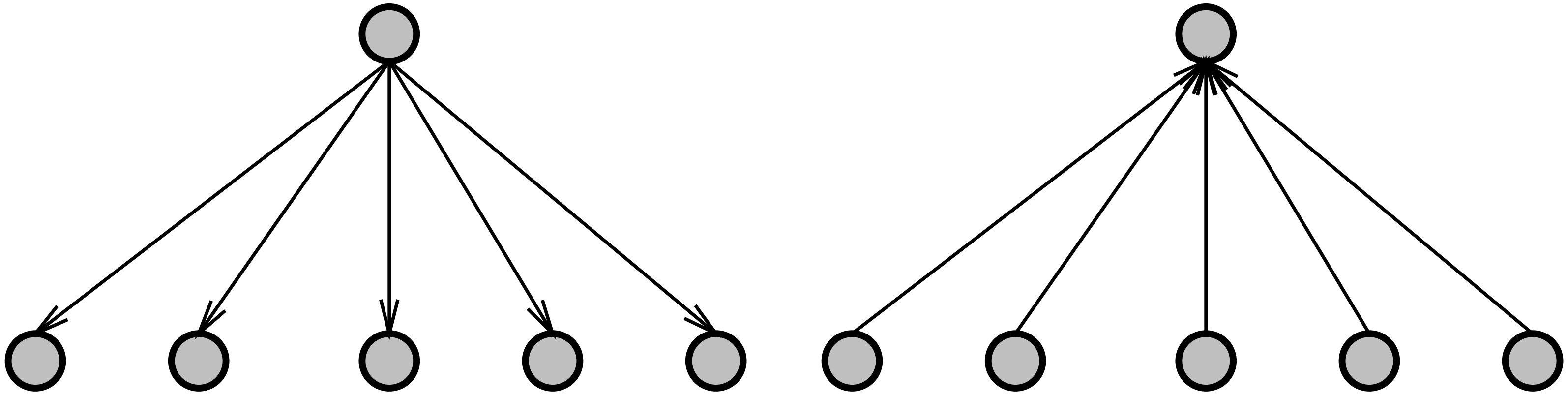
`SSLOT(pe, val)`	Set `pe`'s slot = `val`	`shmem_put`
`WSLOT(pe)`	Wait for local `pe` slot set	`shmem_wait`
`RSLOT()`	Reset local slot	=

- Data Transfer: Message layout via `SSLOT`; `BC_PUTD` via Datatypes management
- Patterns: {2-tree-↑, 2-tree-↓, all-one, one-all, butterfly, prefix}



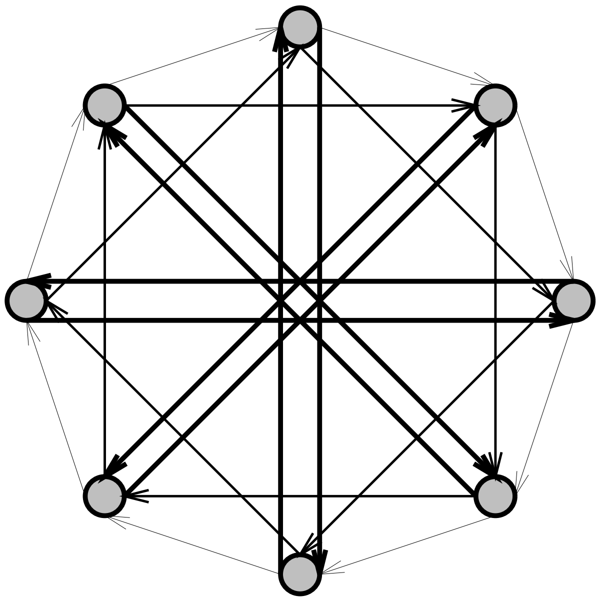
Barrier and Broadcast

- `Barrier`: **synch** messages, 2-tree-↑, 2-tree-↓
- `Bcast`: **synch** arranged 2-tree-↑; **data** 2-tree-↓



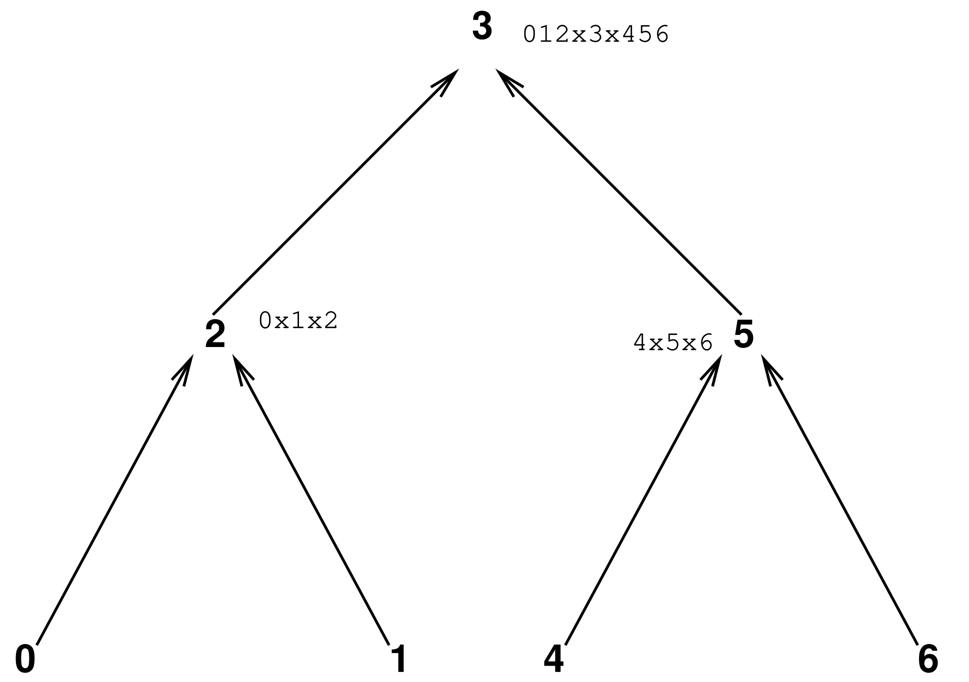
Scatter and Gather

- `Scatter(v)`: **synch** arranged all-one, **data** one-all
- `Gather(v)`: **synch** arranged one-all, **data** all-one



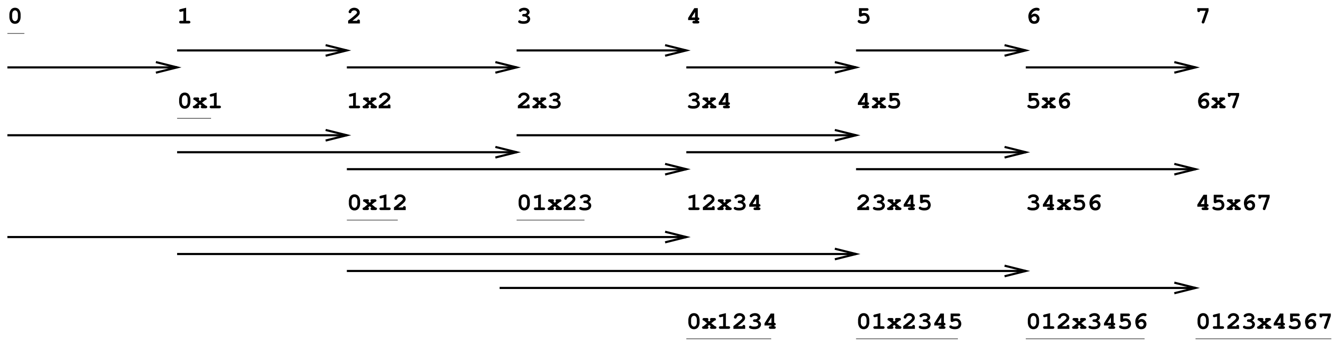
All to All

- **synch** forward, **synch** back, **data** forward
- `Alltoall(v)`: data messages
- `Allgather(v)`: data messages
- `Allreduce`: partial results, commutative only



Reduction

- `Reduce`: partial results, arranged 2-tree-↑
- `Reduce_scatter`: reduce, **data** one-all
- `Allreduce`: non-commutative – reduce, broadcast



Scan

- `Scan`: partial results prefix
- Segmented Scan via user-defined operator

Context Creation

- Basic Collective Transfer
- Context Creation:

Per-PE Counter	local	Send, Recv: $pe \mapsto comm$
Single Counter	barr-bcast	Send, Recv: hashing+
Global Resolution	all-gather	Send, Recv: hashing-
Free Nomination	all-to-all	Send: $pe \mapsto comm$

- Tradeoff: context creation *vs.* context mapping
- Minimise Comms Cost: **Free Nomination**

Communicators

- Base: Context Creation, Collective
- Duplicate: context creation only
- Create: local member test, context creation
- Split: all-to-all, local $(color, key)$ member calc.
- Intercomm Create: gather, leaders exchange
- Intercomm Merge: all-to-all, local order calc.

Datatypes and Buffer Management

- Derived Datatype Construction and Commit
- Datatyped Message Layout {describe, transform}
- Message Misalignment: transform
- Datatyped Message Traversal
- Local Data Buffer Management

Datatype Message Layout

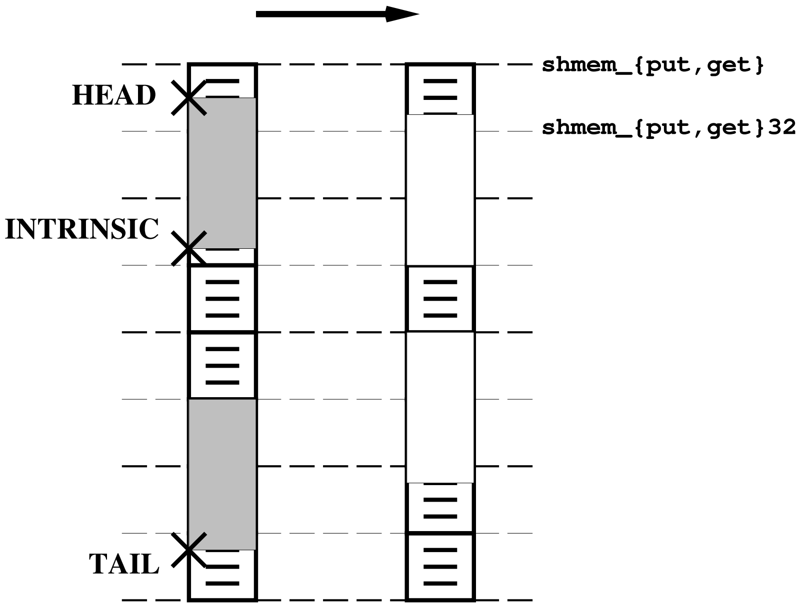
- Datatype Classification (from construction):

  Contig: basic, contiguous

  Strided: (h)vector, regular (h)indexed and struct

  Transform: irregular (h)indexed and struct

- Message: contiguous, datatype $\times$ count
- Encode **Contig** as $(address, size)$
- Encode **Strided** as $(address, count, blocksize, stride)$
- Flatten **Transform**: Pack/Unpack $\leftrightarrow$ Buffer



Message Misalignment

Head	Transform at target
Tail	Transform at target
Instrinsic	Transform at source and target

Datatyped Message Traversal

- Recursive on structure: {vector, indexed, struct}
- Base Case: contiguous fragment
- Basic Operation: {pack, unpack, put, get}
- Operand: {contig,strided}, {local temporary, remote message} buffer
- Special Case: "True" Message Layout Copy

Local Data Buffer Management

- Required for:
  - Message Transformation
  - Buffered (asynchronous) Communications Model
  - Temporary Global Computation Results
- General: `malloc` and `free`
- User-provided Buffer Space via
  `MPI_Buffer_{attach,detach}`

Environmental Management

- Init: alloc, setup, barrier
- Finalize: barrier, free
- Abort: signal agent, `kill(getpid(), SIGABRT)`
- Timers: period 6.6 ns, using `rtclock`
- Host Name: `gethostname`
- Host Rank: no host designation, `MPI_PROC_NULL`
- I/O Capable Rank: no I/O designation, `MPI_ANY_SOURCE`

Generic

- Derived Datatypes: construction, query
- Global Compute Operators: function table
- Groups: rank list management
- Communicator Caching: hash table
- Process Topologies: caching
- Error Handling: function table

CRI/EPCC MPI for T3D
Performance Characteristics

A. Gordon Smith

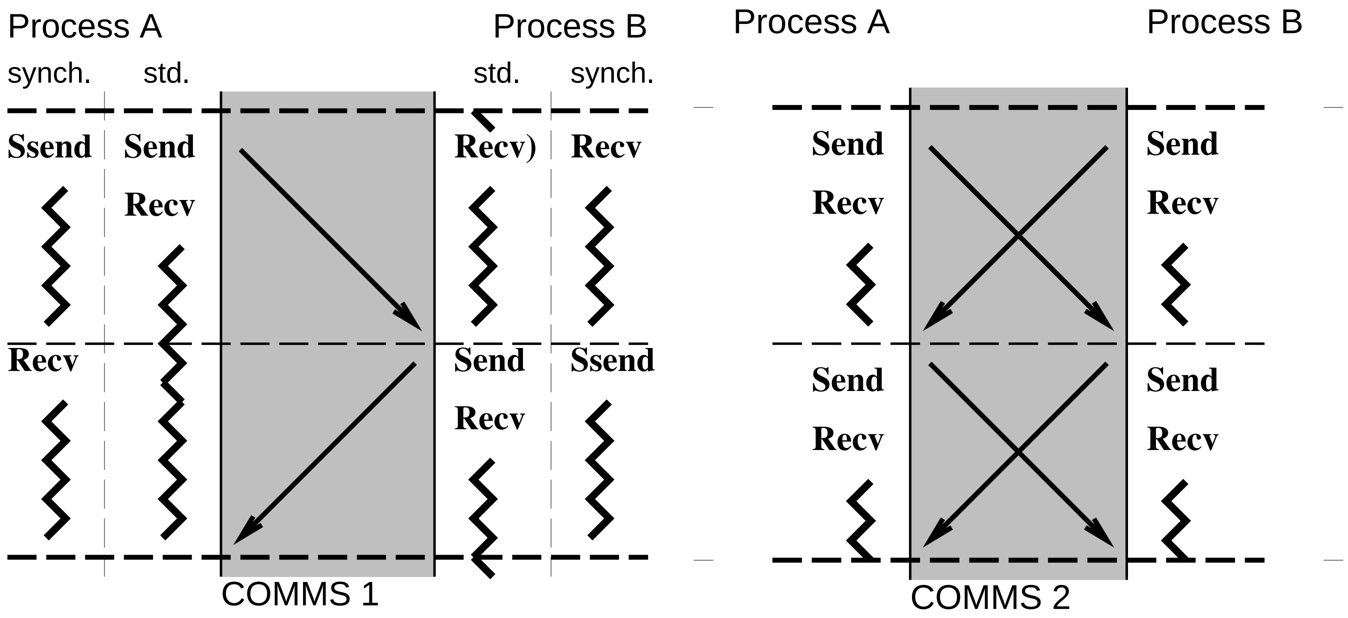
smith@epcc.ed.ac.uk

Edinburgh Parallel Computing Centre (EPCC)

The University of Edinburgh

Overview

- Point-to-point Communications Performance:
  GENESIS COMMS{1,2} Benchmarks
- Comparison: T3D PVM Communication
- Point-to-point Analysis and Improvements
- Collective Communications Performance
- Collective Analysis and Improvements



GENESIS Communications Benchmarks

- **COMMS1**: ping-pong, $latency = (\sum_j^{iters} time_j)/2 \cdot iters$
- **COMMS2**: swap, $latency = (\sum_j^{iters} time_j)/iters$

T3D PVM Communications

- Traditional PVM Send:

```
pvm_initsend(PvmDataRaw);
pvm_pkbyte(buf, len, stride);
pvm_send(tid, tag);
```

- Single-Call (Fast) PVM Send:

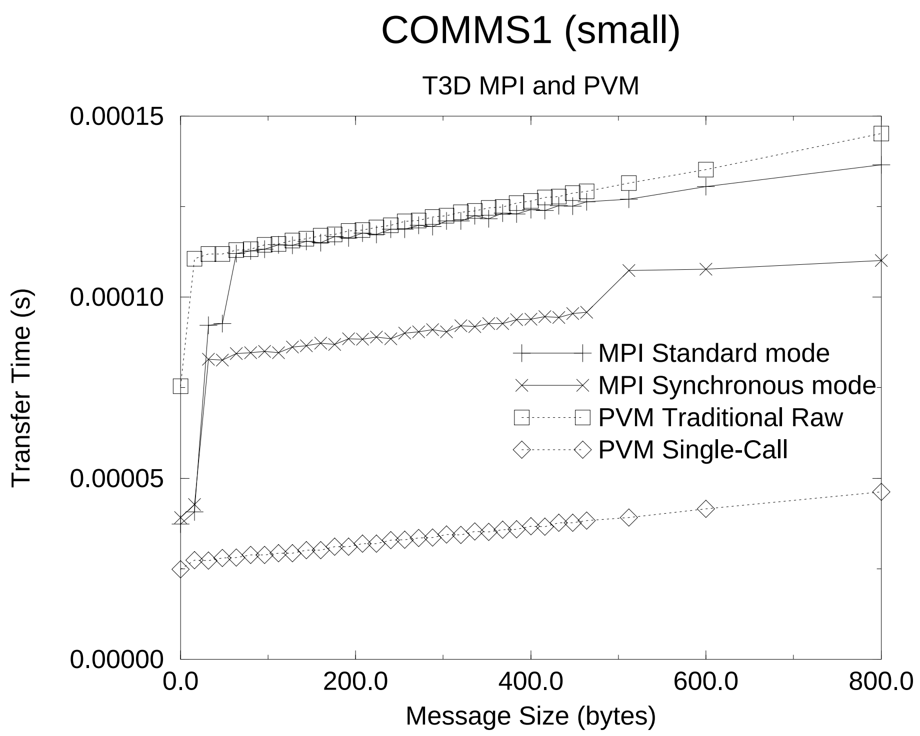
```
pvm_psend(tid, tag, buf, len, PVM_BYTE)
```

- Traditional PVM Receive:

```
pvm_recv(tid, tag);
pvm_upkbyte(buf, len, stride);
```

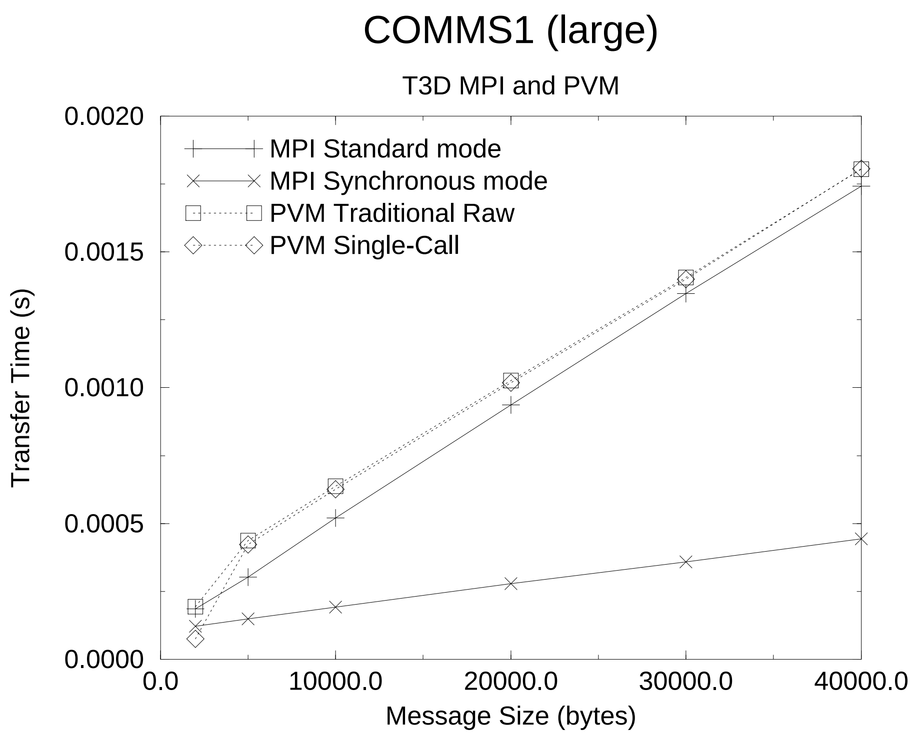
- Single-Call (Fast) PVM Receive:

```
pvm_precv(tid, tag, buf, len, PVM_BYTE, ...)
```



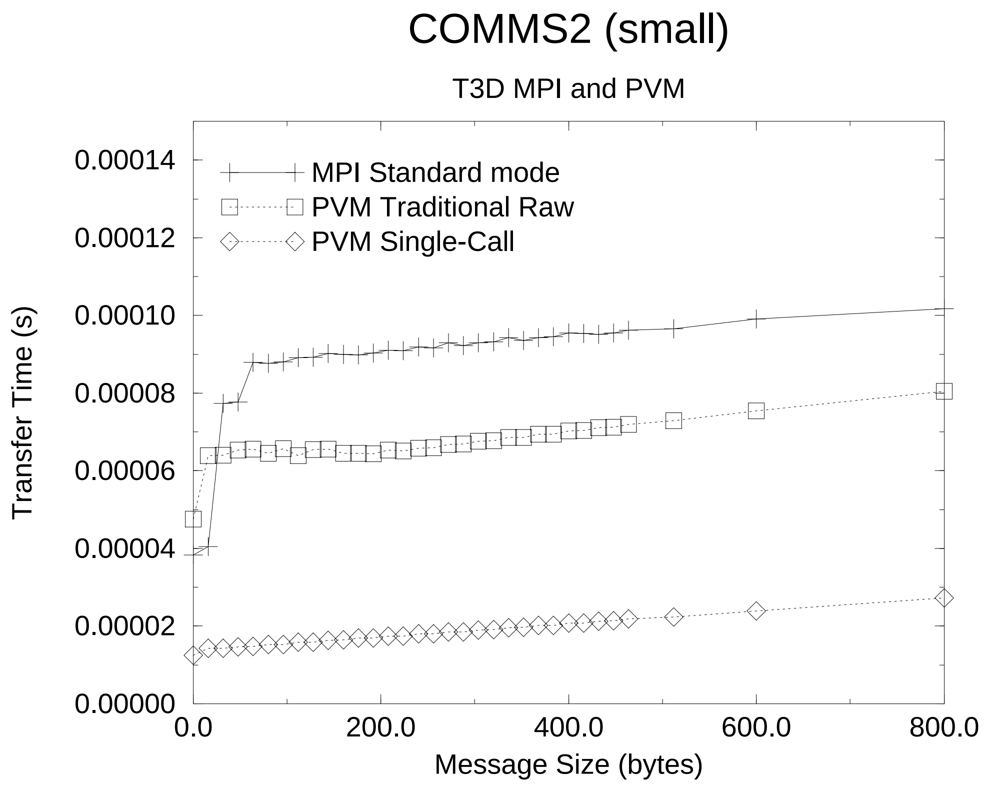
Point-to-Point: **COMMS1** Closeup

- MPI thresh1: **(TA)** ⟶ **(RTA)** 32 bytes
- MPI thresh2: **(RTA)** ⟶ **(RAT)** `MPI_Ssend` 512 bytes
- Message Copy Overhead: `pvm_send`, `MPI_Send`



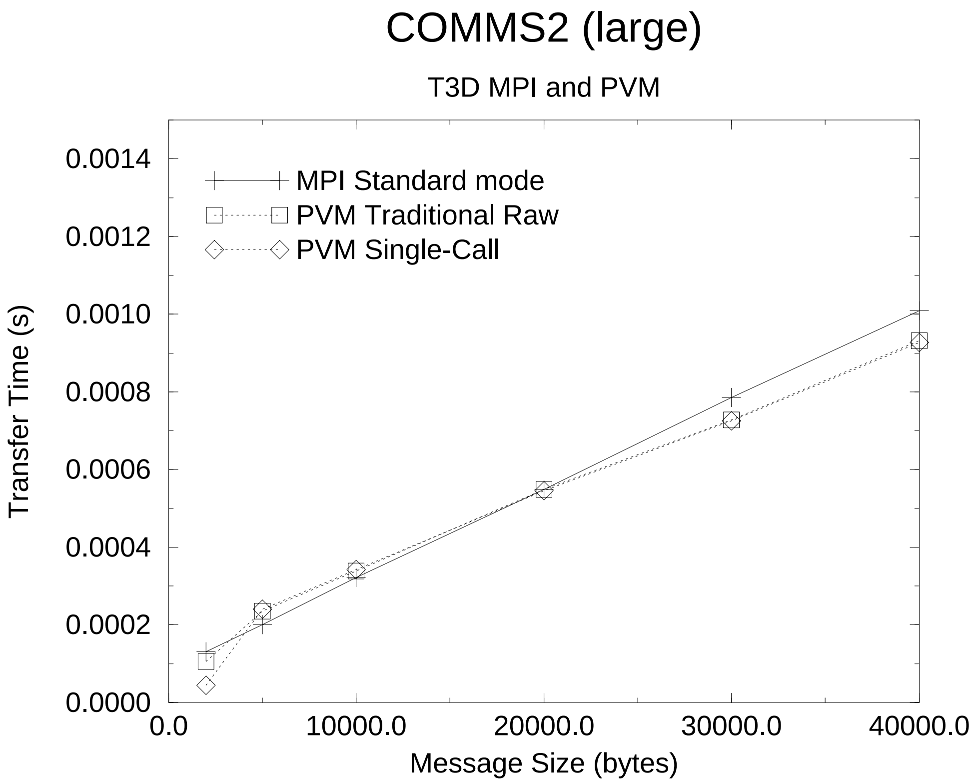
Point-to-Point: **COMMS1**

- PVM thresh: Protocol Message size = 2048 bytes
- Standard, PVM: bandwidth → {alloc, copy, get}
- Synchronous: bandwidth → put



Point-to-Point: **MPI COMMS2** Closeup

- MPI thresh: **(TA)** $\longrightarrow$ **(RTA)** at 32 bytes

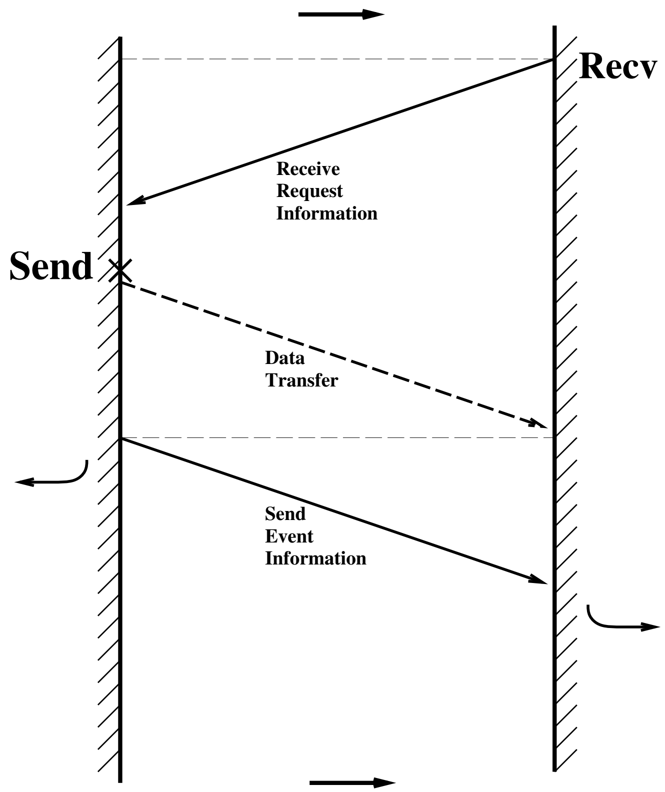


Point-to-Point: **MPI COMMS2**

- PVM: single active send and recv buffers
- MPI: protocol to free per-transfer buffering

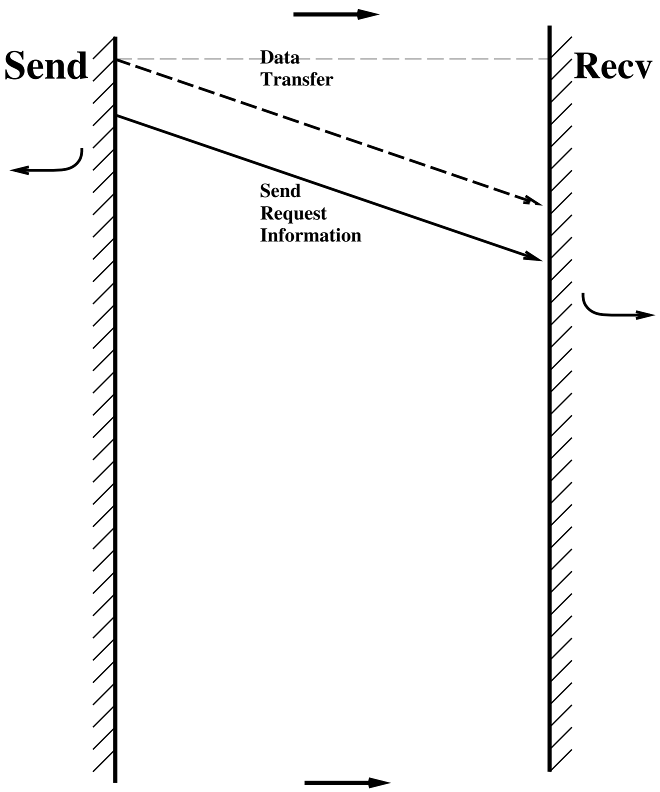
Point-to-Point Analysis

- Protocol Thresholds: **(TA)** → **(RAT)** early
  - Extend **(TA)** Message Size
  - Chain Header and Data Protocol Messages
  - `BC_PUTPM`: atomic protocol/data send
- Bandwidth: MPI Standard → copy-get
  - Recv Source known: optimistic acknowledge → put
  - **Order** Semantics wrt. wildcarded immediate Recv
  - Cross-over Correction: Receiver Acknowledge in all protocols



Optimistic Acknowledge

- Receive Source Known
- Receiver Acknowledge
- Sender Transfer
- Sender Acknowledge
- Crossover Preference



Small Buffered (T)

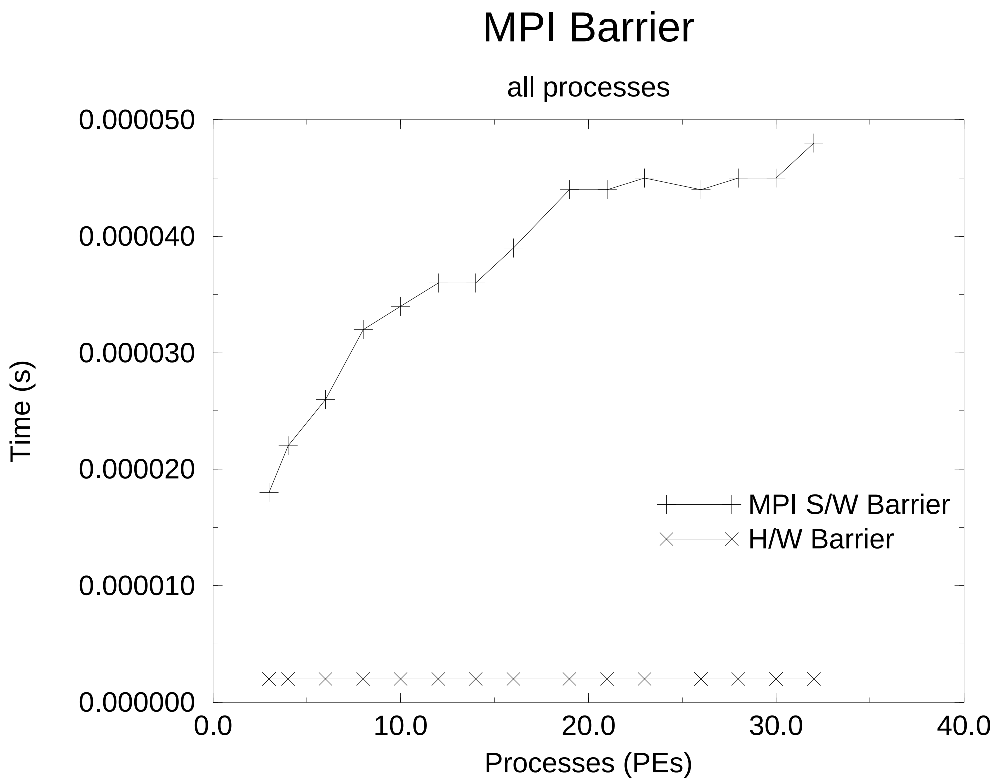
- Sender Request
- Transfer with Request
- Receiver Copy

Collective Comms Benchmarks

- Rooted Collective:

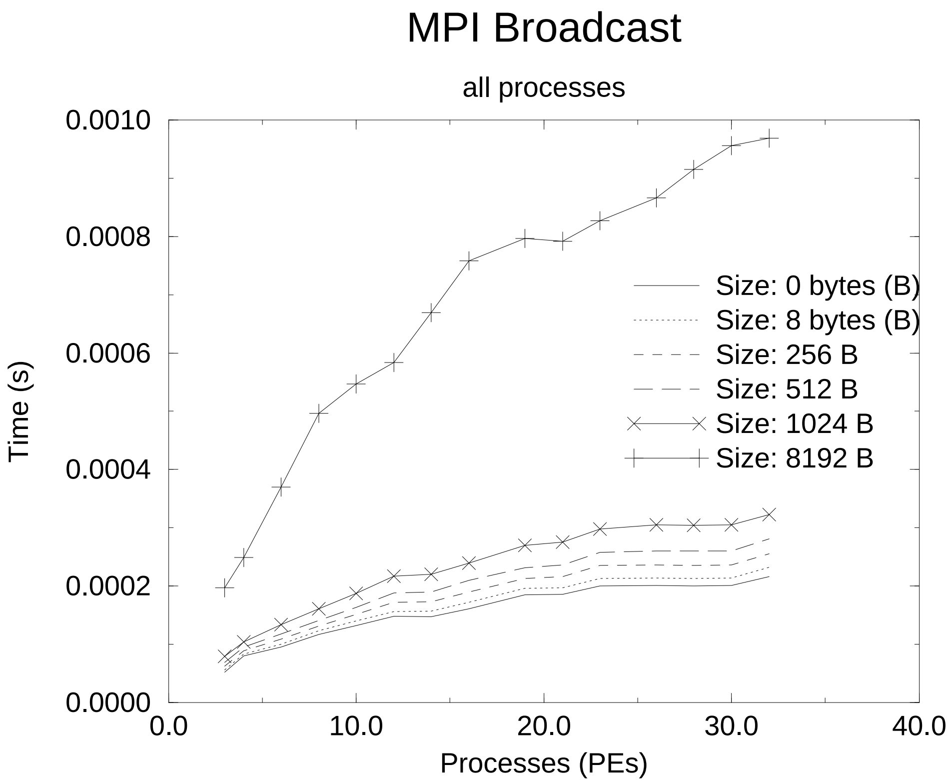
```
BARRIER
start timer
for Iterations
    collective(.., root, ..)
    (BARRIER)
end for
stop timer
```

  - Time at Root and Leaf: local time
  - Time overall: barrier per iteration, subtract barrier overhead (separate timed loop)
  - Bandwidth at Root and Leaf



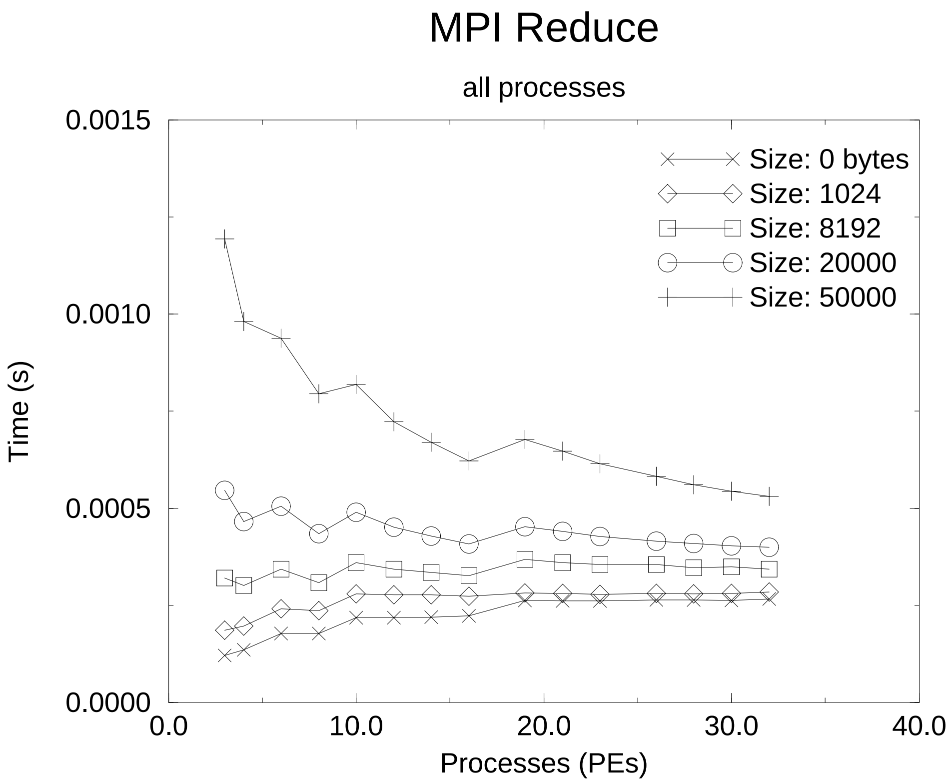
Collective: **Barrier**

- MPI → H/W Barrier for `MPI_COMM_WORLD`
- Overall Barrier Time (implicit root: rank 0)



Collective: **Broadcast**

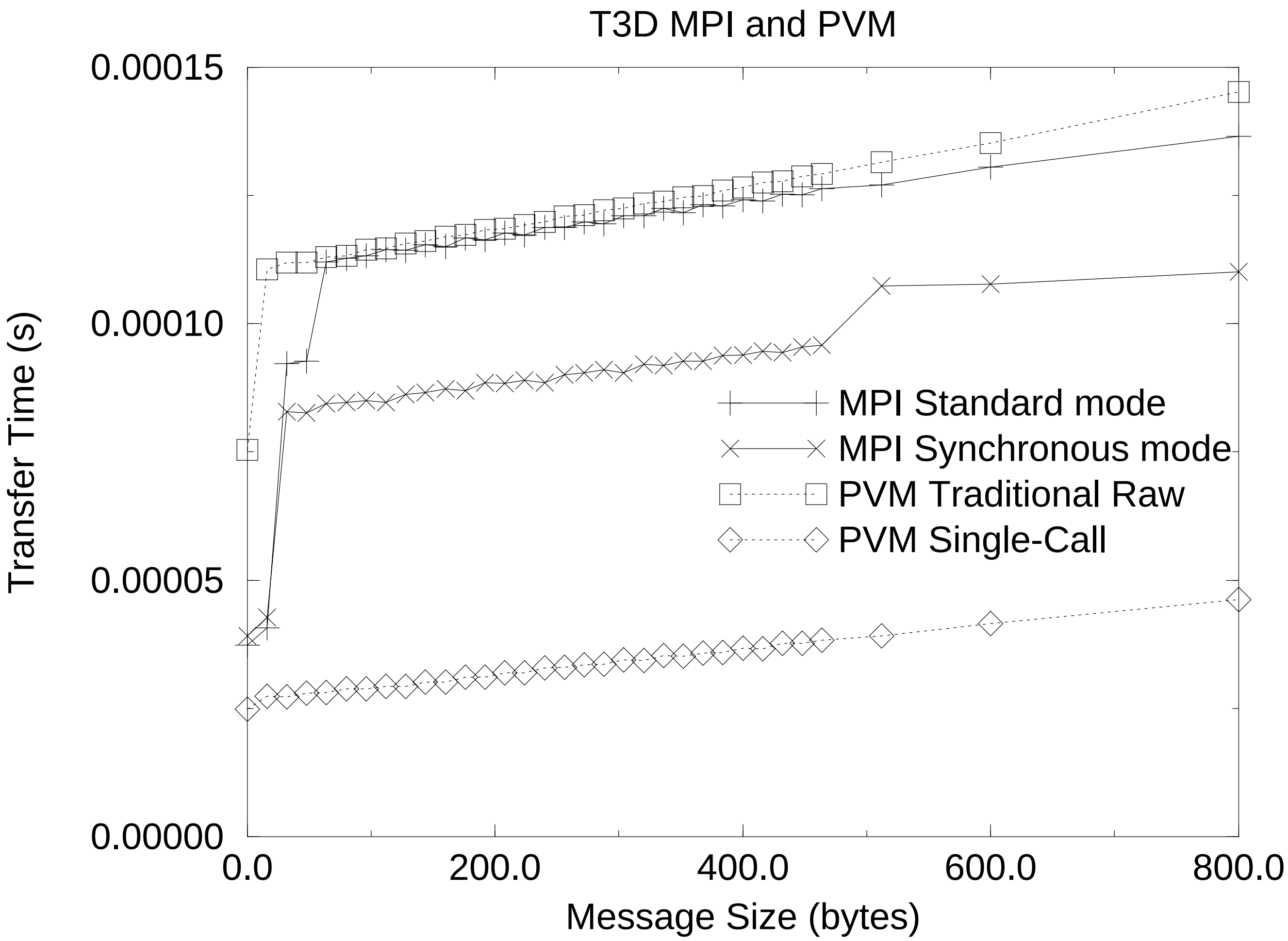
- Root: rank 0
- Overall Broadcast Time



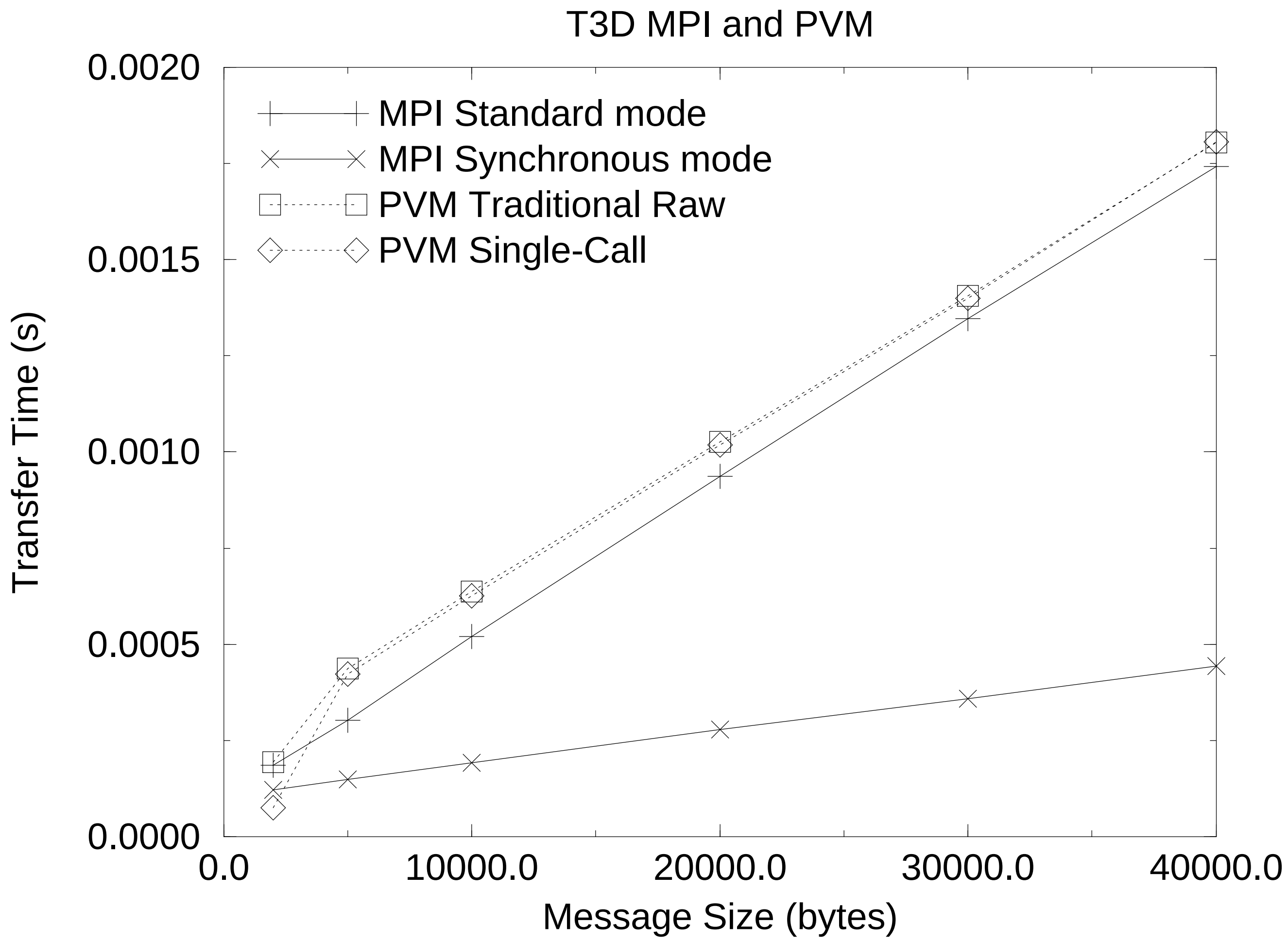
Collective: **Reduce**

- Integer Sum: op = `MPI_SUM`, datatype = `MPI_INT`
- Overall Reduction Time

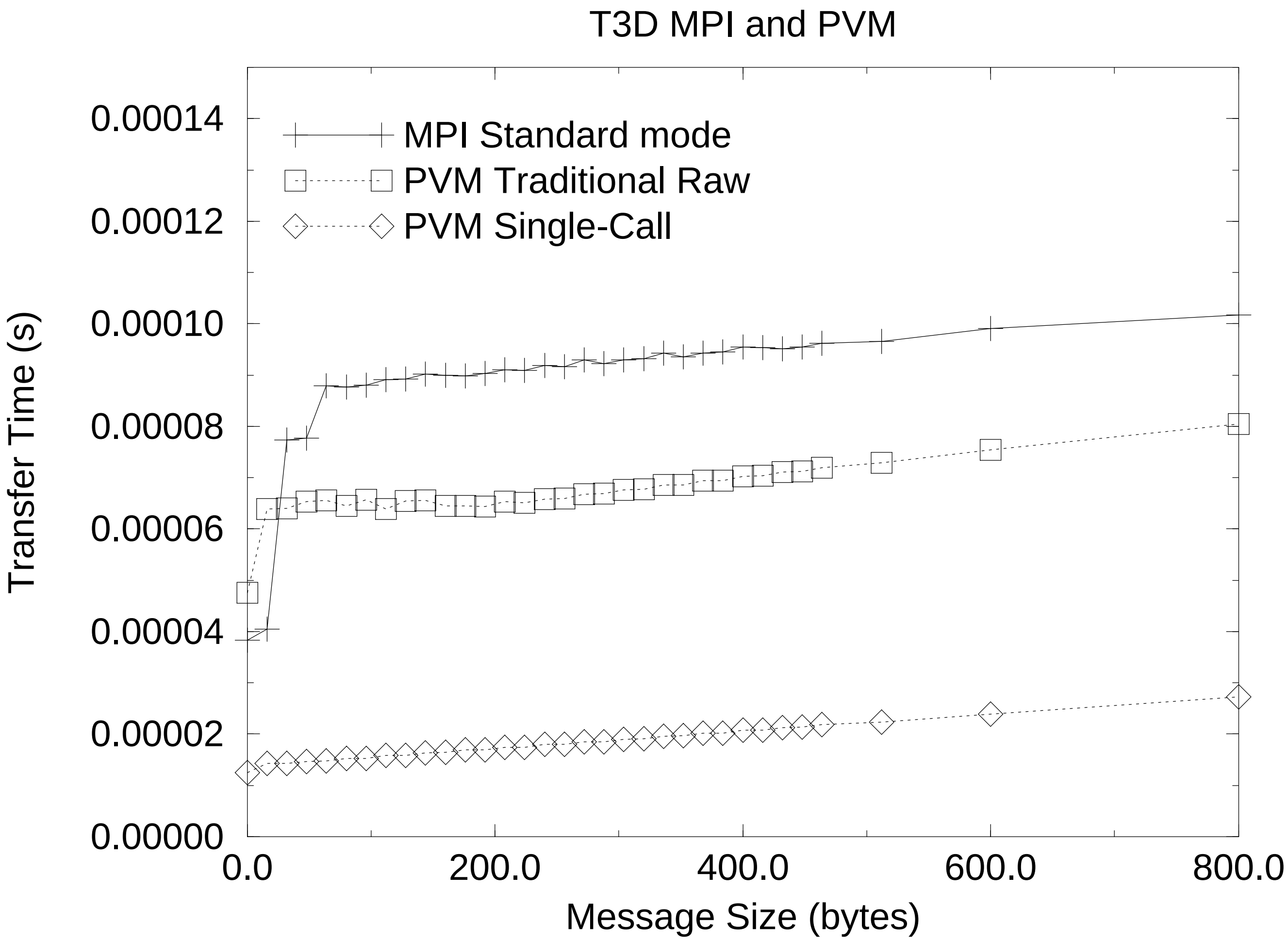
COMMS1 (small)



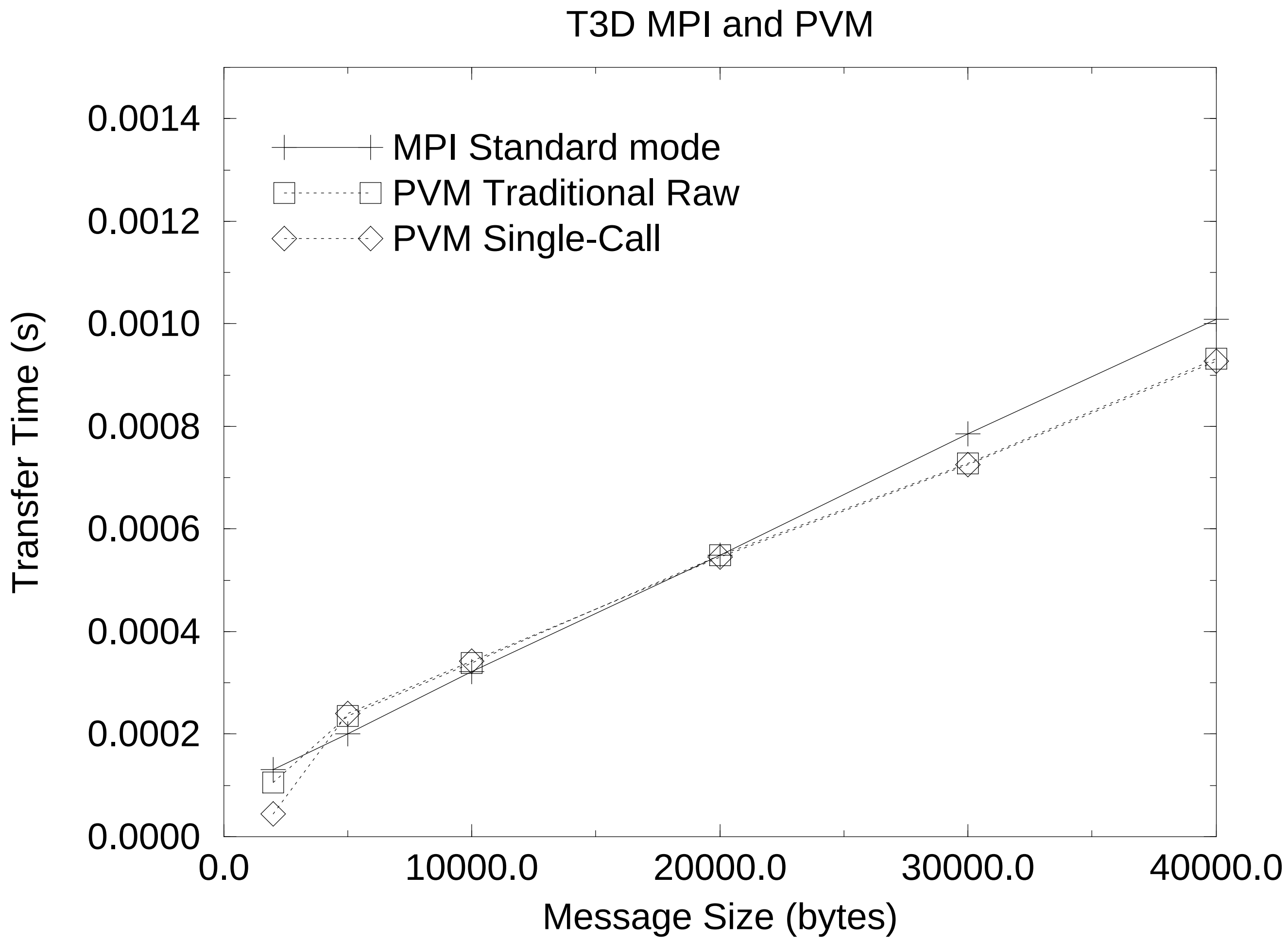
COMMS1 (large)



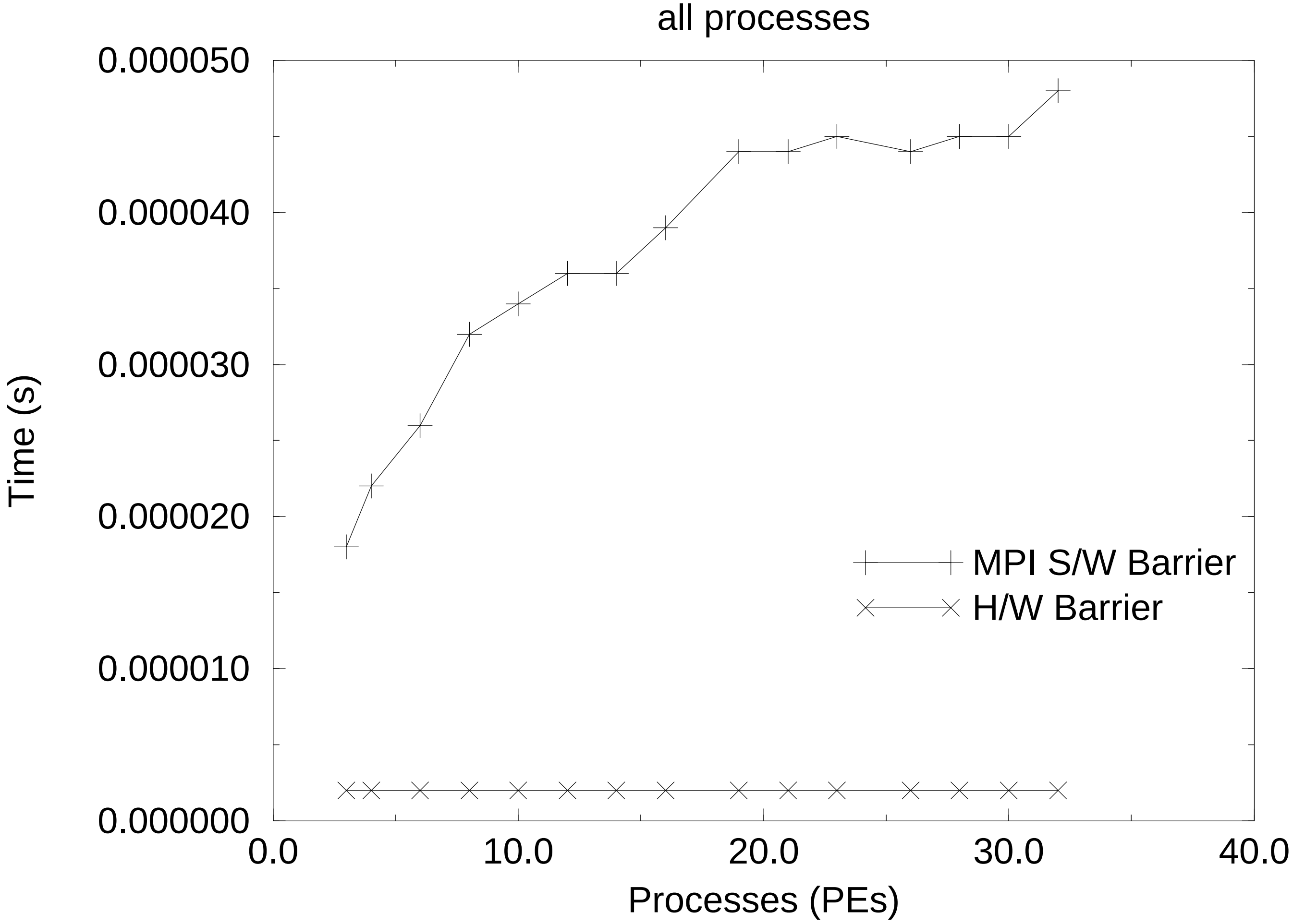
COMMS2 (small)



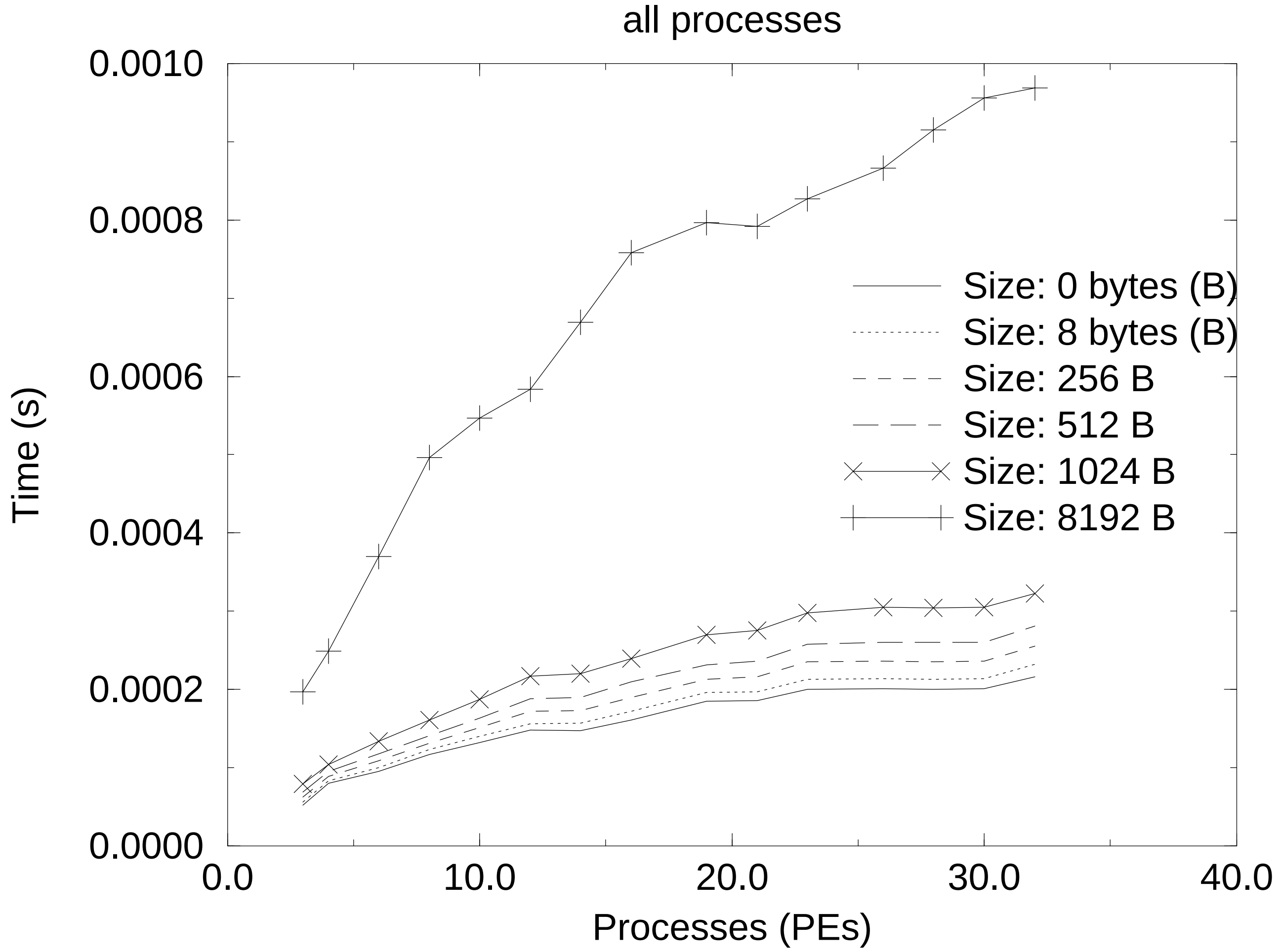
COMMS2 (large)



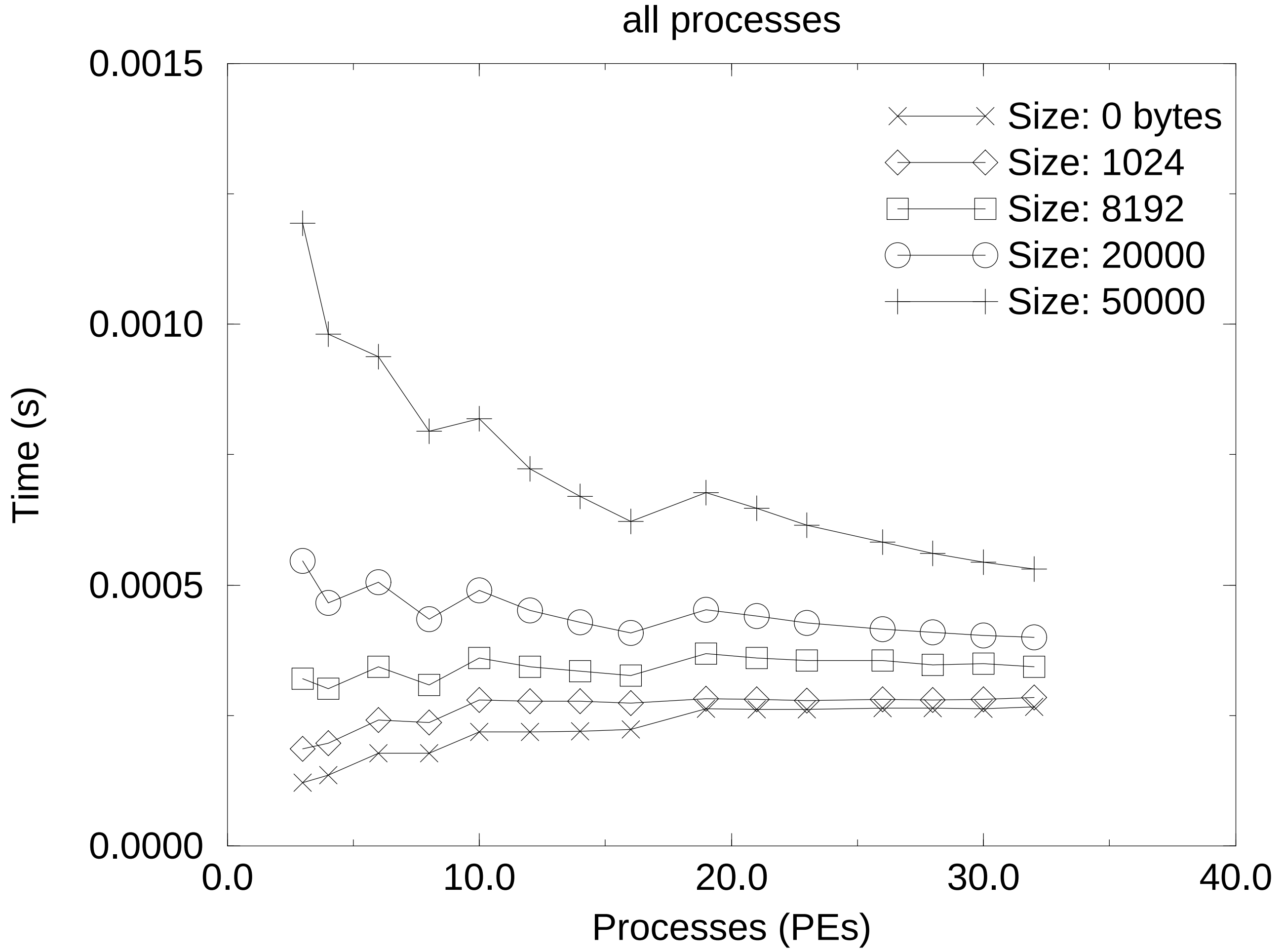
MPI Barrier


all processes
0.000050
0.000040
0.000030
0.000020
0.000010
0.000000
Time (s)
0.0
10.0
20.0
30.0
40.0
Processes (PEs)
MPI S/W Barrier
H/W Barrier

MPI Broadcast



MPI Reduce


all processes
0.0015
0.0010
0.0005
0.0000
Time (s)
0.0
10.0
20.0
30.0
40.0
Processes (PEs)
Size: 0 bytes
Size: 1024
Size: 8192
Size: 20000
Size: 50000

CRI/EPCC MPI for T3D
Distribution, Installation and Support

A. Gordon Smith

smith@epcc.ed.ac.uk

Edinburgh Parallel Computing Centre (EPCC)
The University of Edinburgh

Overview

- Release Mechanism via WWW
- License and Support Agreements
- Software Problem Reports
- Development Plan
- MPI-2

CRI/EPCC MPI for T3D Release

- Information, Licenses and Software via World-Wide Web (WWW)
- URL: `http://www.epcc.ed.ac.uk/t3dmpi/Product`
- Release Page ← User ID and Password
- User ID and Password ← License Agreement
- License Agreement PostScript via WWW
- Software Release: compressed archive (`.tar.Z`)
- Installation Instructions and Known Problems

CRI/EPCC MPI for T3D Installation

- Select and Save Release: `<filename>`
- Extraction:

```
% zcat <filename> | tar xvf -
  x ./SPR, 3188 bytes, 7 tape blocks
  x ./user.ps, 102776 bytes, 201 tape blocks
  x ./install, 1616 bytes, 4 tape blocks
  x ./test/shifter.c, 2079 bytes, 5 tape blocks
  x ./test/shifterF.f, 1925 bytes, 4 tape blocks
  x ./test/README, 1806 bytes, 4 tape blocks
  x ./test/makefile, 1186 bytes, 3 tape blocks
  x ./libmpi.a, 1227234 bytes, 2397 tape blocks
  x ./mpi.h, 12831 bytes, 26 tape blocks
  x ./mpif.h, 7015 bytes, 14 tape blocks
  x ./README, 1902 bytes, 4 tape blocks
```

Release Contents and Installation

- Installation Script (`./install`)
- `libmpi.a` → `/mpp/lib`
- {`mpi.h`, `mpif.h`} → `/usr/include/mpp`
- Confidence Check: MPI Test Programs (C and Fortran)
- User Guide (PostScript)
- Software Problem Report (SPR) form

License Agreement

- EPCC $\leftrightarrow$ Cray T3D Site
- Quarterly Major Software Releases and Information
- Software Problem Report mechanism
- "*Reasonable Endeavours*" Support Committment
- No Guaranteed Support Turnaround Period
- Duration: $\longrightarrow$ November 1995

Support Agreement

- Cost: $3000 per Site
- Major and Minor Software Releases and Information
- Support Committment according to Priority:

`Critical`	2 weeks
`Urgent`	4 weeks
`Major`	8 weeks
`Minor`	13 weeks
`Info-Only`	–

- 12 Month Agreement Duration

SPR Usage

- Users directed at Local Cray/Site Support
- Non-trivial Problems Reported by SPR
- E-mail SPR to `t3dmpi@epcc.ed.ac.uk`
- Fax SPR to +44 131 650 6555
- Send SPR by Postal Service (address in SPR)

SPR: Identification

- Subject and Site Identification:
  - Correspondence Electronic Mail Address: (smith@epcc.ed.ac.uk)
  - Cray T3D Site: (EPCC)
- Software Identification:
  - Software Version Number: (1.1a)
  - Language Bindings: [*] C [ ] Fortran

SPR: Priority

[*] `Critical`: Prevents progress, eg. application run fails under all circumstances

[ ] `Urgent`: Limits progress, eg. application tests OK, fails on production run

[ ] `Major`: Avoidable-unsatisfactory, eg. problem workaround complicates code or degrades performance

[ ] `Minor`: Avoidable, eg. problem workaround is OK

[ ] `Info-only`: Observation, eg. test demonstrates problem not relevant to application

SPR: Problem Area

[ ] Installation: problem or comments with release and/or installation

[ ] Compilation: errors or warnings related to compilation with MPI headers and libraries

[*] Execution: problem with or during interactive or queued execution

[*] MPI <Area>: problem relating to use of MPI interface calls from MPI area (cf. standard)

MPI Interface Calls (MPI_Ssend, MPI_Recv)

SPR: Problem Description

- Behaviour: error messages, deadlock, address error
- Information: circumstances, problem size, frequency
- Demonstration: test cases, code fragments
- ```
  Symptoms:    ( Hangs consistently. )

  Details:     ( Application copying complicated
                 datatyped message locally, by sending
                 to self. )

  Test Cases: (
    MPI_Ssend(orig, SIZE, Yuk, me, TAG, comm);
    MPI_Recv(copy, SIZE, Yuk, me, TAG, comm, &stat);
  )
  ```
  ```

Development Plan

- Control and Conservation of Memory Usage
- Message Layout and Copying Improvements
- Collective Communications Performance
- MPI-2 Single-Sided Interface on CRAY T3D
- Further Work: Small Message Pt2pt Optimisation
- Further Work: Pt2pt Protocol Experimentation

Control and Conservation of Memory Usage

- Memory-bound MPI Applications Requirements
- User-Control on MPI Memory Usage
- Runtime Control via Environmental Variables:

MPI_TOTAL_MEM	Absolute ceiling on MPI memory use
MPI_SM_POOL	Max messages in shared protocol queue
MPI_SEND_BUFF	Max memory for buffered communication (Standard/Buffered modes)
...	

- Conservation: pruning of replicated objects,
  eg. communication event notices

Message Layout and Copying Improvements

- Utilise Structured `shmem_` Put, Get
- Reduce Cost of Pack/Unpack: alloc-copy, copy-free
- Improve – Buffer Allocation and Return
- Improve – Message Copy Bandwidth
- Current: local `shmem_put32`, `memcpy`
- Alternatives: `memwcpy` and `SCOPY`

Collective Communications Performance

- Benchmarking and Analysis
- Target Small Message Performance
- Structured Algorithms *vs.* Brute Force
- Generalisation of Tree Algorithms: valency, hypercube
- Improved Use of Temporary Buffers

MPI-2 Single-Sided Communications

- MPI-2 Subcommittee (Marc Snir, IBM)
- Generalisation of MPI Requests
- Transfer of MPI Datatypes and Requests
- Communications: Put, Get, and Synchronous Put
- Blocking, Immediate and Persistent variants
- Specialist Read-Modify-Write Operation:
  {Test&Set, Test&Or, Fetch&Add, Compare&Swap}

MPI-2 Single-Sided Comms on CRAY T3D

- Design atop CRAY T3D Shared Memory Access
- (Subset) Implementation and Release
- {Put, Get} $\mapsto$ `shmem_{put,get}`
- Synchronisation via Protocol
- Early Implementation of MPI-2 Features
- MPI-2 $\rightarrow$ `shmem_` Users: Potential Portability
- MPI-2 $\leftarrow$ `shmem_` Users: Feedback