CDAC
The Supercomputing People

HPCC Software
KSHIPRA

1

KSHIPRA

- Complete communication subsystem for cluster of multiprocessors (CLUMPS).
- Supports MPMD, SPMD and client server models.
- Leverages on low latency and high bandwidth of protected user level protocol - Active Messages (AM II specifications from University of California at Berkeley).
- Exploits SAN features to the fullest.
- Supports binary compatibility and source compatibility to existing applications.

2

Motivation for KSHIPRA

- TCP/IP is Designed for WANs
  - OS overhead for send/receive is extremely high.
  - Low bandwidth.
  - Higher error rates and high per packet processing cost using WANs.
  - Does not exploit the high reliability of SANs.
- So we need ☞ KSHIPRA
  - A better protocol with reduced overheads, better bandwidth and which exploits the reliability of SANs.

3

Features of KSHIPRA

- Designed for clusters
  - KSHIPRA Active Messages are designed for SANs.
- Protected User Level Primitives
  - No OS overhead for send/receive.
- Success oriented protocols
  - Exploits high reliability of SANs.
  - Minimal per packet processing cost.

4

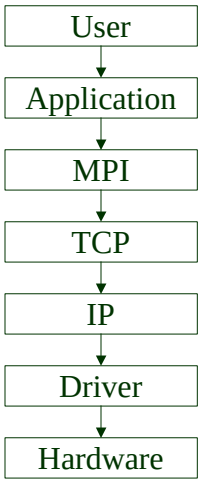
Features of KSHIPRA

- Simple Buffer Management
  - All layers collapsed.
  - Reduced copies.
- Decoupling Communication from Computation
  - Traditional send/receive semantics do not allow communication to be overlapped with computation.
  - Active Messages allow overlap of communication with computation.

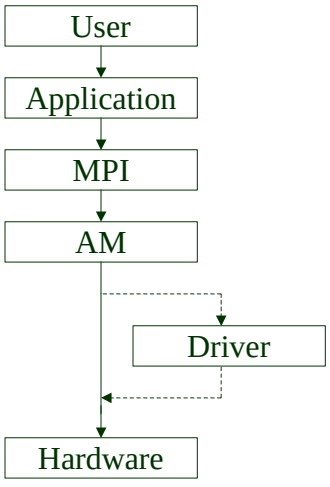
5

All layers collapsed

MPI over TCP/IP

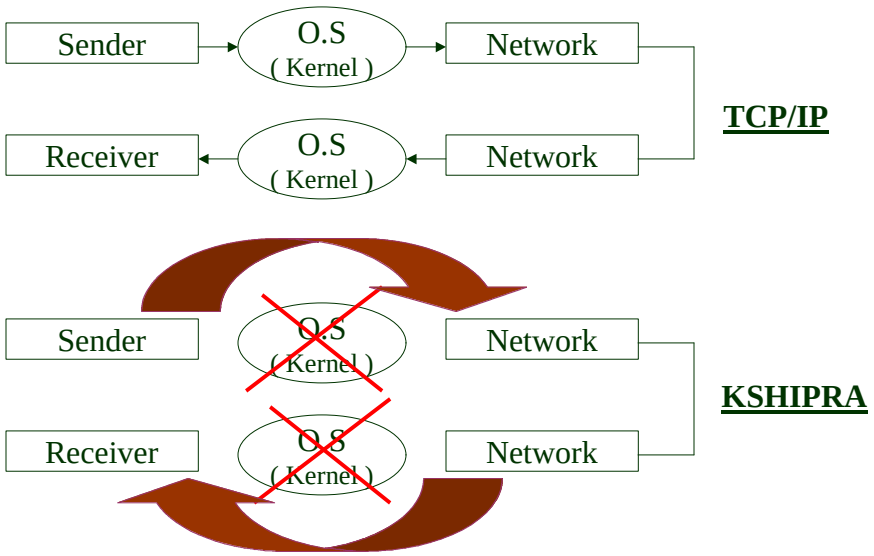


MPI over AM

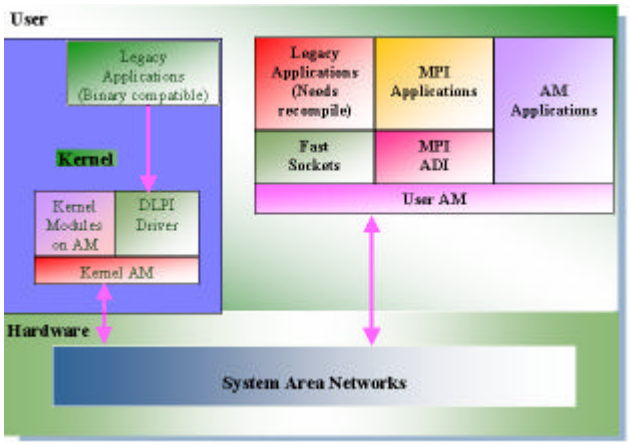


6

Reduced copies



7



8

KSHIPRA Active Messages

- RPC styled programming model.
- Thread safe.
- Frame work for Fault tolerance.
- Supports SPMD, MPMD and client server models.
- Multidevice support to make efficient use of SMP architecture.
  - Network (PARAMNet or MyriNet).
  - Shared memory (Solaris System V IPC).
- Built upon Active Messages software by University of California, Berkeley.

9

MPI under KSHIPRA

- Exploits low latency and high bandwidth of Active Messages.
- Zero copy at MPI level.
- Exploits AM for asynchronous communication.
- Supports multiple network interface cards.

10

KSHIPRA Name Server

- The Name Server provides software naming services for the programs using either the MPI software or the AM software directly. This is always used in conjunction with the above two software.
- It keeps a mapping of global endpoint strings names to the hardware address of the corresponding endpoint.

11

KSHIPRA Name Server

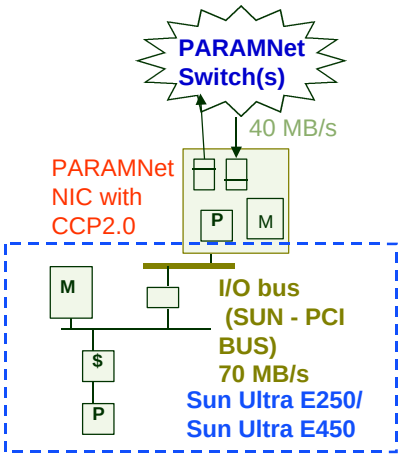
- Later an endpoint can use other endpoint's global name to locate the endpoint and hence communicate with it.
- Even though it is developed for naming services of AM and MPI over AM it can be used for any general naming services.

12

PARAMNet Components

Intelligent NIC

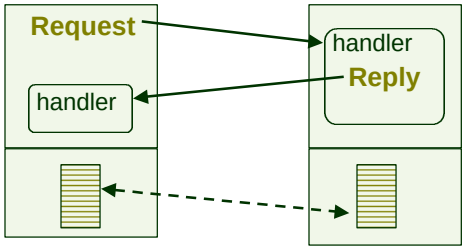
- Dedicated Processing power and storage embedded in the Network Interface.
- Protocol processing (reliability, flow control, ack/nack, etc) offloaded to the NIC.



13

AM Communication Primitives

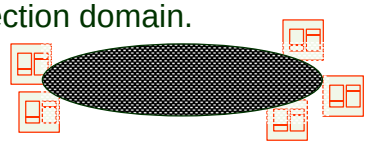
- Request / Reply small active messages (RPC style).
- RequestI / ReplyI for medium messages.
- Bulk-Transfer (RequestXfer & GetXfer).
- Highly optimized communication layer on a range of HW.



14

Endpoints Emulates Virtual Network

- Endpoint abstracts the notion of “attached to the network”.
- Virtual network is a collection of endpoints that can name each other.
- Many processes on a node can each have many endpoints, each with own protection domain.



15

Availability

- KSHIPRA on PARAMNet is available on Solaris 2.6.
- KSHIPRA on MyriNet is available on Solaris 2.6.
- KSHIPRA on PARAMNet is available on Linux.

16

Limitations

- KSHIPRA on PARAMNet allows only 4 endpoints or 4 MPI processes per node.
- Cleand daemon is running to remove the shared memory segments left out by abnormally exited KSHIPRA applications. Unrelated applications using persistent shared memory objects may get affected.
- TCP/IP and KSHIPRA cannot be simultaneously used on SANs.

17

Enhancements

- KSHIPRA will be made available for Solaris 7.
- New version of KSHIPRA will improve performance of some of the Large Asynchronous transmissions.
- KSHIPRA with increased number of endpoints or MPI processes per node will be made available with C-DAC's next generation SANs.

18

Enhancements

- KSHIPRA and TCP/IP coexistence on SANs will be supported.
- Kernel Active Messages (KAM) will be made available on Solaris.
- Fast Sockets will be made available on Solaris.

19

Road Map

- Support for VIA will be provided.
- KSHIPRA on C-DAC's next generation hardware will be supported.

20

CDAC-MPI

21

CDAC-MPI

- CDAC-MPI (C-MPI) - high performance implementation of MPI on Cluster of MultiProcessors(CLUMPS)

22

Motivation for CDAC-MPI

- Support Cluster of MultiProcessors (CLUMPS)
- Enhanced performance by taking advantage of SMP nodes
- On PARAM Openframe - exploits hardware communication features.

23

Features of C-MPI

- Collective Calls optimized for CLUMPS
- C-MPI supports execution of MPI applications - enhanced performance on a CLUMPS
- Supports applications written in Fortran and C.

24

Features of C-MPI (contd..)

- Multiprotocol Support - applications can run - TCP/IP (Fast Ethernet) or AM II(PARAMNET or MyriNet).
- C-MPI optimized collective calls - MPI_Reduce, MPI_Barrier, MPI_Bcast, and MPI_Allreduce.

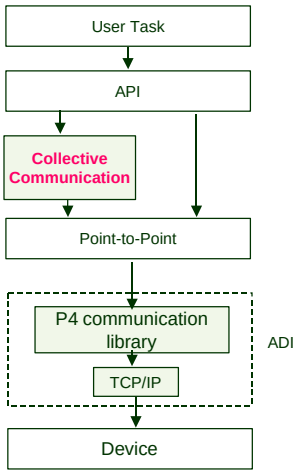
25

Features of C-MPI (contd..)

- Optimization using efficient algorithms for collective operations
- Optimization taking advantage of SMP nodes
- Optimization by modifying the MPICH stack

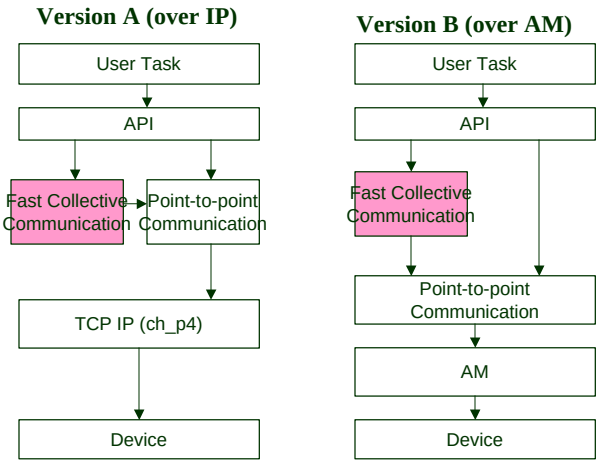
26

MPICH Architecture



27

C-MPI Architecture



28

Implementation of Algorithms - using the SMP Advantage

- The close coupling between the processors in a SMP node has been used effectively to reduce the number of remote communication steps involved in collective calls
- Each processor in a node acts either as a master or as a slave. The slave sends its data to the master which does the global operation and then sends the data back to the slave

29

Broad Strategy for Optimization of Collective Calls

Step 1 :

1) Tree Algorithms
   – for calls with distinct root
      - **Ex. Broadcast, Scatter, gather**

2) Circulant Graph Algorithm
   – for calls with no distinct root
      - **Ex. Barrier, All to All, All gather**

30

Broad Strategy for Optimization of Collective Calls (contd..)

Step 1 :

3) Hybrid Algorithm - Distributed Combine and Collect Algorithm
   - **Ex. Allreduce**

31

Broad Strategy for Optimization of Collective Calls (contd..)

Step 2 :

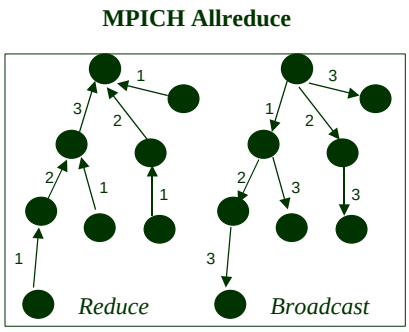
1) Tree Algorithms
   – for short vectors

2) Hybrid Algorithms
   – for intermediate and long vectors

32

Optimized Algorithm with SMP Support

- Example 1 : All reduce



MPICH Allreduce first reduces to Zero and then broadcasts

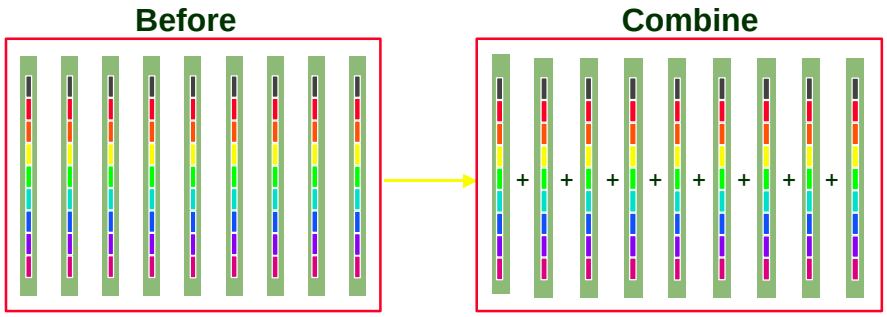
MPICH Reduce uses reverse optimal tree and

MPICH Broadcast is optimal tree(binary tree).

33

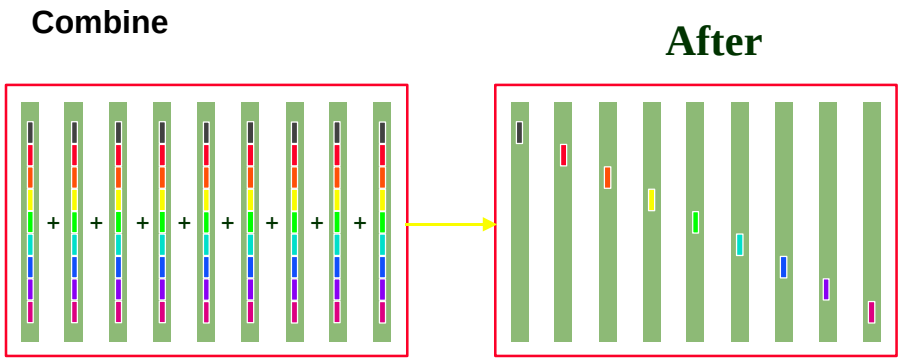
Optimized Algorithm with SMP Support (contd..)

- C-MPI Allreduce - Uses the distributed combine and collect algorithm



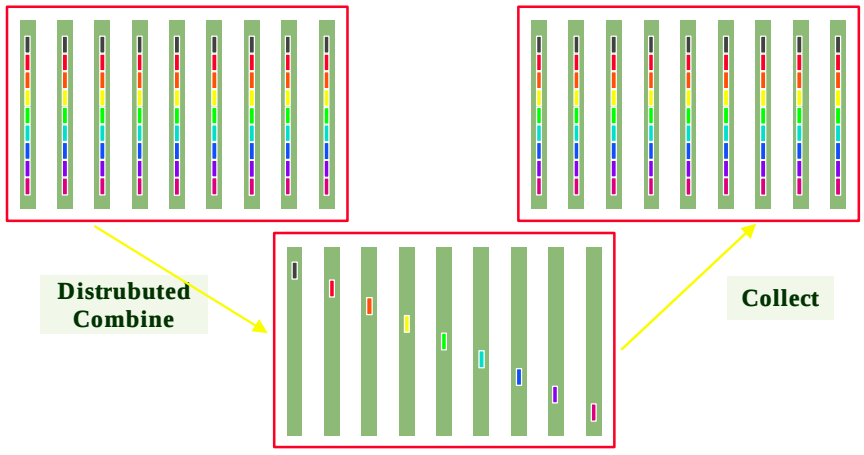
34

Optimized Algorithm with SMP Support (contd..)



35

Optimized Algorithm with SMP Support (contd..)

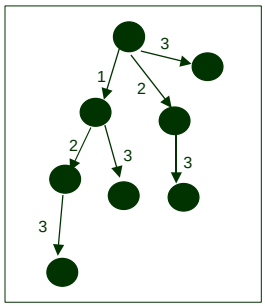


36

Optmised Algorithm with SMP support (contd..)

Example 2 : Broadcast

MPICH Broadcast **C-MPI Broadcast**

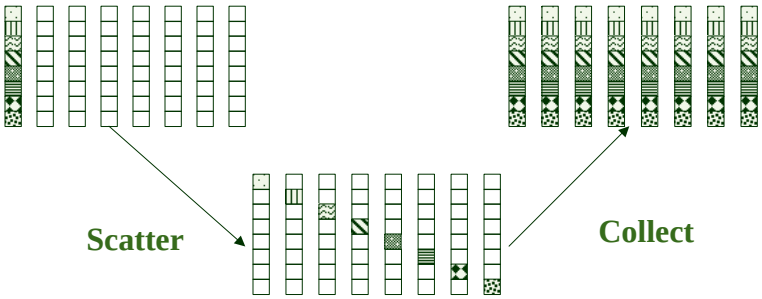


Uses the Minimum Spanning Tree for short vectors and the Scatter-Collect strategy for long vectors.

37

Optimised Algorithm with SMP support (contd…)

C-MPI Broadcast (8 processes)



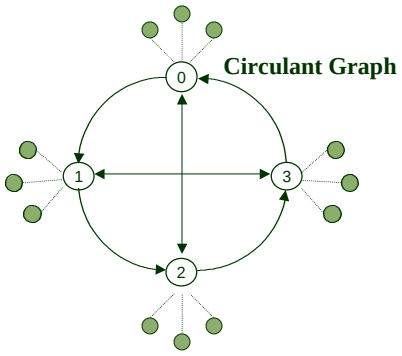
38

Optimized Algorithm with SMP Support (contd..)

Example 3 : Barrier

C-MPI Barrier uses Circulant Graph Algorithm

MPICH Barrier does asynchronous send and receive



39

Optimized Algorithm with SMP Support (contd..)

Comparison between CDAC-MPI & MPICH Barrier with regards to the Circulant Graph Algorithm

	CDAC-MPI	***MPICH***
1. Communication Steps		
a. For powers of two number of nodes	log *n*	log *n* + 1
b. For non powers of two number of nodes	log *n* + 1	log *n* + 1
2. Computation involved	computation of source and destinations for each step done in the init phase, as part of MPI_Init.	computation of source and destination is done within the call.

40

Availability

- C-DAC MPI is available on Solaris 2.6 platform
- C-DAC MPI is available on Linux platform
- C-DAC MPI is being made available on Solaris 2.8 platform.
- C-DAC MPI is being made available on AIX platform

41

Limitations of C-MPI

- This version of C- MPI Library is not thread safe.

42

Enhancements

- Support MPI 2 Standard.

43

Thank You

44