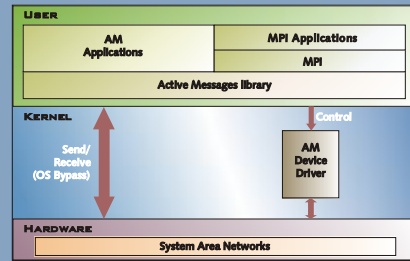


PRODUCT OVERVIEW

ACTIVE MESSAGES

The Active Message interface allows applications to communicate among objects called *communication endpoints*. Endpoints are treated as single unit for communication, event management, and synchronization. Any process can create multiple endpoints and gather related ones into *bundles*. Protection and message authentication is provided by per-endpoint message tags, which can be specified by the application. Conventional communication boundaries and domains allow interaction only between user-level parallel processes, whereas endpoints can communicate across these boundaries. The key to both the versatility and efficiency of Active Messages lies in the ability to associate a small amount of user-defined computation in the form of a handler with the reception of each message. The role of the message handler is to process the arriving packet promptly in order to allow an ongoing computation to continue. Handlers get the message out of the network and transfer the data directly into the application data structures. They may alternatively provide a small remote service and send a reply message back.

AM implementation leverages on a mechanism that bypasses the operating system in the common case. This eliminates the operating system overhead in frequent operations like send/receive and provides a communication abstraction that allows the users to exploit the underlying network to the fullest.



KSHIPRA - AM Architecture Model

AVAILABILITY

Supported Hardware	:	Workstation Clusters
Supported Operating System	:	AIX, Solaris and Linux
User Interfaces	:	Command Line
Supported Languages	:	C
Prerequisite Network Hardware	:	PARAMNet-II & Gigabit Ethernet for VIA and PARAMNet-I for AM
Supported APIs	:	VIA, AM, MPI

*All trademarks and brand names are owned by their respective owners.

KSHIPRA

Scalable Communication Substrate for Cluster of Multi Processors



The **KSHIPRA**[®], communication substrate, designed to support low latency and high bandwidth is the key to the high level of aggregate system performance and scalability of C-DAC's HPCC software. KSHIPRA caters to both parallel and distributed client server programming models.

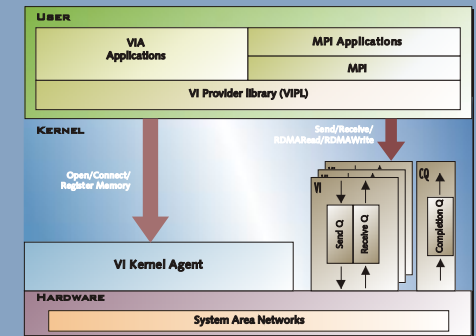
The core of KSHIPRA lies in the fully protected user level communication primitives: Virtual Interface Architecture (VIA) and Active Messages (AM). KSHIPRA VIA implementation conforms to VIA specifications jointly authored by Intel, Compaq and Microsoft and AM implementation conforms to Active Messages II specifications given by University of California, Berkeley. The basis for VIA implementation is M-VIA from NERSC center at Lawrence Berkeley National Laboratory and for AM, AM-II software provided by UCB.

APPLICATION PROGRAMMING INTERFACES

KSHIPRA exports the following interfaces:

- Virtual Interface Architecture
- Active Messages
- MPI

KSHIPRA provides the Abstract Device Interface for VIA and AM for layering Message Passing Interface (MPI), thus supporting the MPI over VIA as well as MPI over AM.



KSHIPRA - VIA Architecture Model

VIRTUAL INTERFACE ARCHITECTURE

VIA defines a set of functions, data structures, and associated semantics to move data in and out of a process memory. It achieves low-latency, high-bandwidth communication and data exchange between processes running on two nodes within a computing cluster, with minimal CPU usage.

VIA gives a user process direct access to the network interface, avoiding intermediate copies of data and bypassing the operating system in a fully protected fashion. Avoiding interrupts and context switches, whenever possible minimize CPU usage.

KEY FEATURES

- Protected user level communication substrate with network virtualization
- Transparent support for multiple protocols over Cluster of Multi Processors (CLUMPS)
- VIA support providing protected, zero-copy user-level access to reduce software overhead to exploit low latency and high bandwidth feature of the System Area Network
- Active Message programming paradigm, which exports the low latency and high bandwidth of the underlying system area networks
- MPI application programming interface for parallel programming

Centre for Development of Advanced Computing



Centre for Development of Advanced Computing

C-DAC Knowledge Park, No. 1, Old Madras Road, Byappanahalli, Bangalore - 560 038, India
Tel: +91-80-534 1874, 534 1909 Fax: +91-80-524 7724
e-mail: bdm@cdacindia.com website: <http://www.cdacindia.com>

Head Office

Pune University Campus, Ganeshkhind,
Pune - 411 007
Tel: +91-20-569 4000/01/02/03
Fax: +91-20-569 4059

New Delhi

A 335, Shivalki Enclave,
Near Malviya Nagar,
New Delhi - 110 017
Tel/Fax: +91-11-687 4689/91/87
e-mail: bd@cdacindia.com

Hyderabad

2nd Floor, Delta Chambers,
Ameerpet,
Hyderabad - 500 018
Tel: +91-40-340 1331/32
Fax: +91-40-340 1531

• Chennai: +91-44-371 9226/27

• Kolkata: +91-33-321 2357

• Thiruvananthapuram: +91-471-554086

PROGRAMMING MODEL

The VI Architecture eliminates the system-processing overhead of the traditional model by providing each consumer process with a protected, directly accessible interface to the network hardware - a Virtual Interface. Each VI represents a communication endpoint. The VI endpoint pairs can be logically connected to support bi-directional, point-to-point data transfer. A process may own multiple VI's exported by one or more network adapters. A network adapter performs the endpoint virtualization directly and subsumes the tasks of multiplexing; de-multiplexing, and data transfer scheduling normally performed by an OS kernel and device driver. An adapter completely ensures the reliability of communication between connected VI's.

MULTIDEVICE SUPPORT

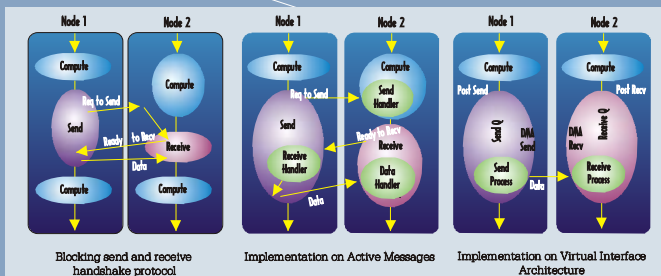
KSHIPRA transparently directs messages through shared memory or the network, depending on whether the destination endpoint is on local node or remote node. This allows parallel applications to fully exploit the CLUMPS architecture by proper data and load partitioning.

NETWORK VIRTUALIZATION

Traffic in one virtual network is never visible to a second virtual network, yet each virtual network retains the direct, user-level network access necessary for high performance. When distinct virtual networks share the same physical network resources, each continues to perceive private resources. This communication multiplexing is critical to high performance message passing with CLUMPS, since several processes are expected to be communicating at once. Network virtualization is supported by both VIA as well as AM.

EFFICIENT DATA TRANSFER

The key feature of the VIA communication model is Memory Registration which requires the source and target data buffers to be registered with the VI Provider so that data is transferred directly between VI consumer and network without copying any data to or from intermediate buffers (zero-copy) and bypassing the operating system. The memory registration process allows the VI consumer to reuse the registered memory buffers, thereby avoiding duplication of locking and translation operations by the VI Kernel Agent. The data movement operations can be initiated by writing directly to a Doorbell, which is generally, a NIC register memory-mapped to user-space. Thus, performance critical data-path is optimized in VIA.



Why VI Architecture ?

- > **Designed for Clusters:** VIA attacks the problem of the relatively low achievable performance of interprocess communication (IPC) within a cluster. Cluster computing consists of short-distance, low-latency, high-bandwidth IPCs between multiple building blocks
- > **Protected User Level Primitives:** VIA provides a user process a direct access to the network interface, avoiding intermediate copies of data and bypassing the operating system in a fully protected fashion
- > **Reliability guarantees:** The reliability attributes of the physical network below VIA are separated into three different categories:
 - o *Unreliable Delivery* guarantees all the data that is delivered is not corrupted during transmission and will only be delivered once
 - o *Reliable Delivery* guarantees that data sent is received uncorrupted, only once, and in the order that it was sent
 - o *Reliable Reception* guarantees that data sent is received uncorrupted, only once, and in the order that it was sent. This differs from the earlier one in terms of the operation that is said to be complete only when the data is transferred to the remote memory
- > **Remote DMA:** Remote functions of reads and writes can be carried out using DMA without consuming much of the VI resources

CONCURRECNY and SYNCHRONIZATION

The VI Architecture provides the application with an asynchronous interface to the network. The asynchronous operations provided by VIA operations are similar in spirit to the non-blocking operations of MPI. A program can *post* descriptor onto the message queue of a VI without blocking so that the communication and computation can be overlapped. The completion of a send or receive operation by the network controller is signaled by flipping a status field in the user descriptor. Since the descriptors live in the virtual user space, the completion can be checked without any system call (polling). VIA also supports both an interrupt to awaken a process as well as a callback with associated handler to manage completions. This interrupt is generated by the VI kernel agent and involves the operating system.

ERROR HANDLING

VIA provides three levels of reliability (unreliable-delivery, reliable-delivery and reliable reception). Based upon the level, the error reporting and handling features vary. The VI Provider supplies a mechanism by which VI Consumers can register an error handling function with the VI Provider. In case, an error handling function is not registered, a default error handler logs the errors.

MPI

KSHIPRA caters to the needs of parallel computing community by providing an implementation of the Abstract Device Interface of MPI over VIA as well as AM interfaces. The MPI implementation effectively reflects the low-latency and high-bandwidth properties of VIA and AM to the MPI applications.