An SCI-based Software VIA System for PC Clustering

Jeonghee Shin
University of Southern California

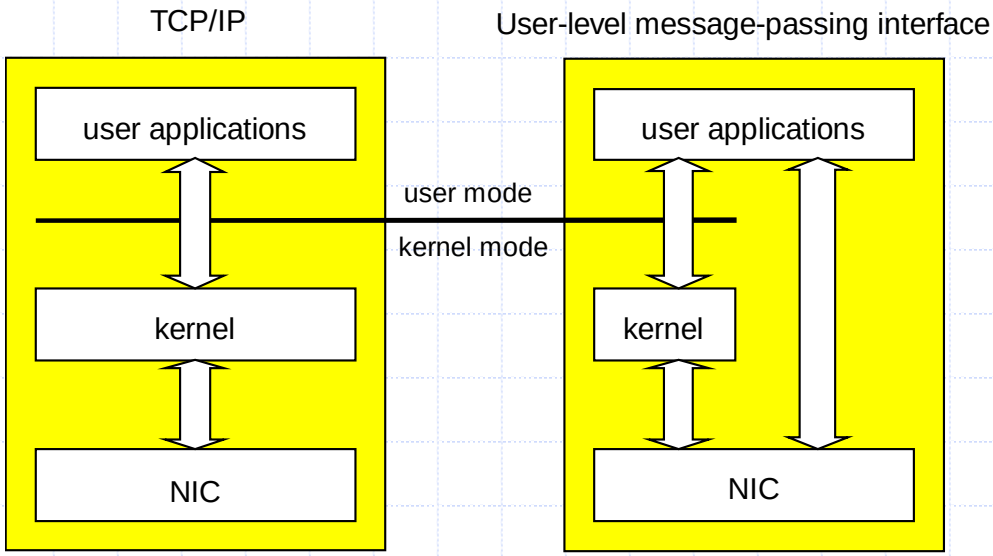
AGENDA

- Introduction
- SCI-based Software VIA(SS-VIA)
  - Overview of SS-VIA
  - Implementation of SS-VIA
- Experimental Result
  - Latency and Bandwidth
  - NAS Parallel Benchmark
- Conclusion

INTRODUCTION

Introduction

- TCP/IP
  - Unnecessary data copy
  - Context switch
  - Interrupt



- User-level message-passing interface
  - without intervention of the kernel
  - U-Net, SHRIMP, Active Message, Fast Message, VIA

Virtual Interface Architecture

- Compaq, Intel, and Microsoft
- Eliminating the system call overhead of traditional models

- Hardware implementation
  - Emulex/Giganet's cLan
  - Finisar's Fibre Channel VI Host Bus Adapter
  - Technical University of Chemnitz

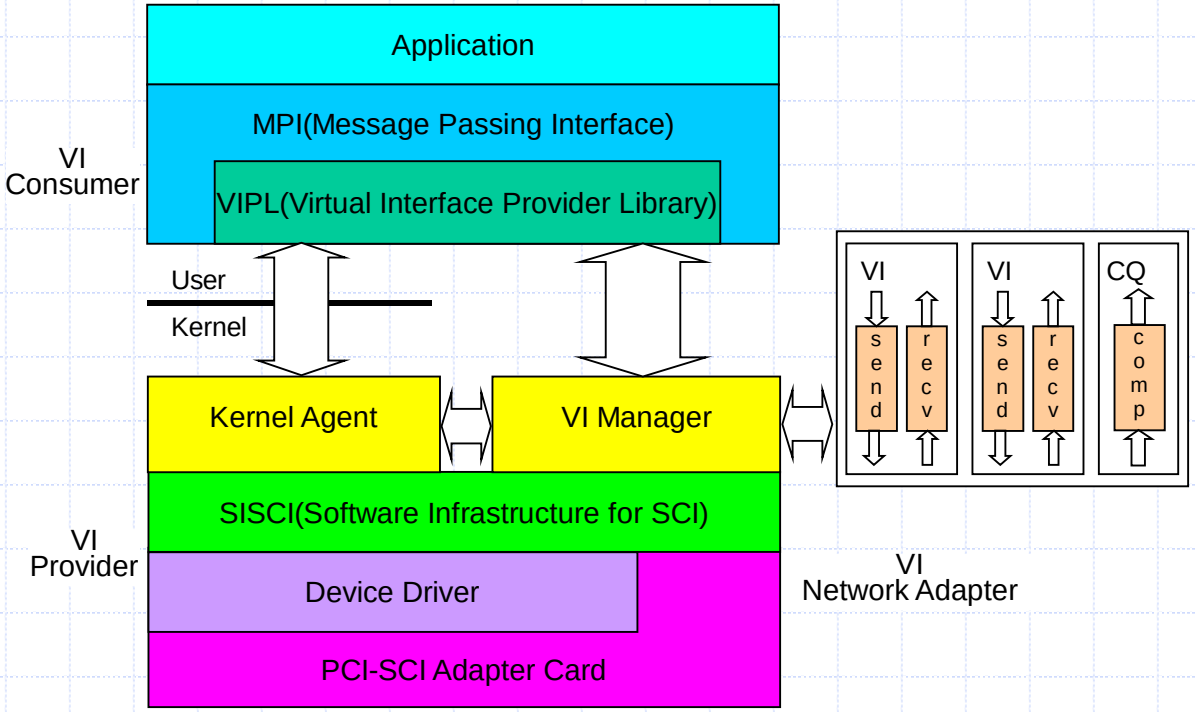
- Software implementation
  - Intel (Fast Ethernet, Myrinet)
  - Lawrence Berkeley Laboratory's M-VIA (Fast Ethernet, Gigabit Ethernet)
  - UC Berkeley (Myrinet)

SS-VIA SYSTEM

SS-VIA

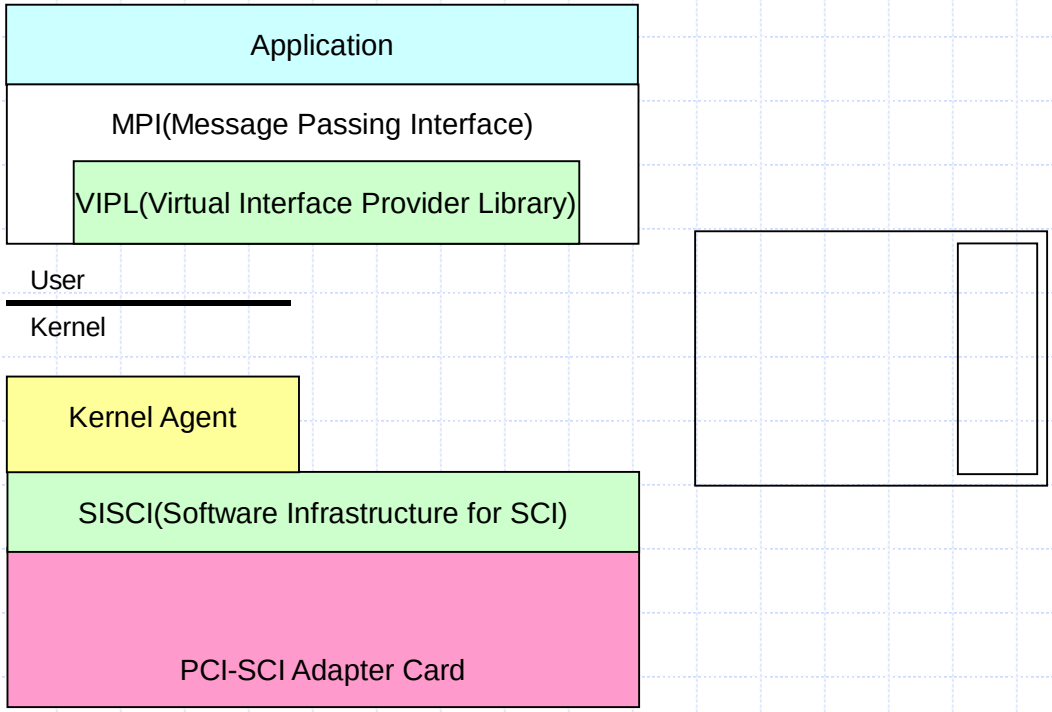
- SCI (Scalable Coherent Interface)
  - ANSI/IEEE standard (1592-1992)
  - Low latency & high bandwidth
  - Remote memory access mechanism
    - Enable a node to transfer data in remote nodes' memory without system call
  - ➲ Excellent environment to build a VIA system
- Support both
  - MPI-based message passing programming
  - SCI-based shared memory programming

Structure of SS-VIA



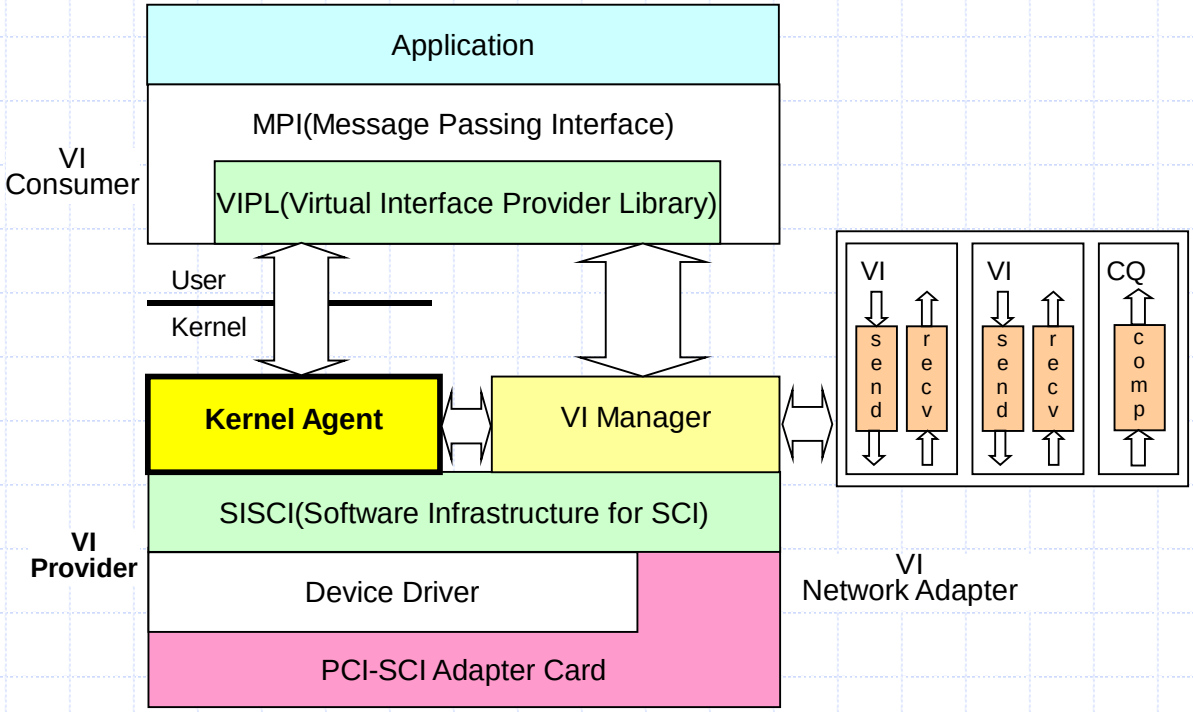
- Virtual Interface (VI)
- Completion Queue (CQ)
- VI Provider
- VI Consumer

Structure of SS-VIA



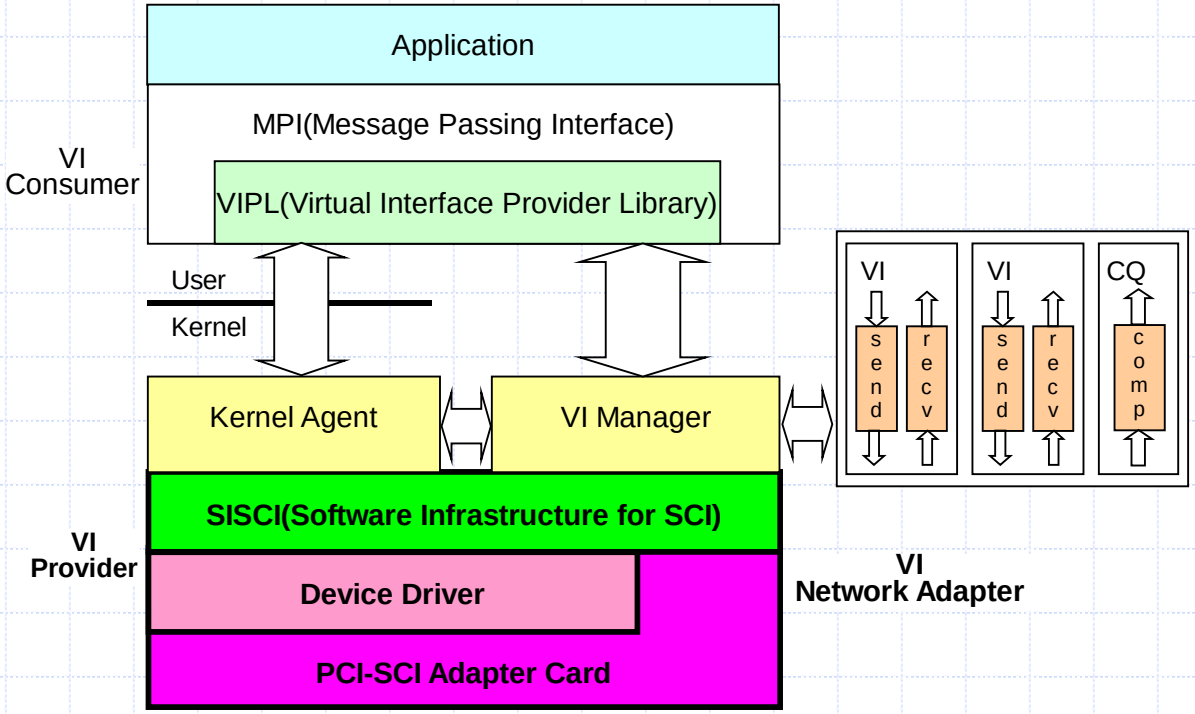
- VI
  - Handled by the VI Manager
  - Consists of a pair of Work Queues
    - Send Queue & Receive Queue

Structure of SS-VIA



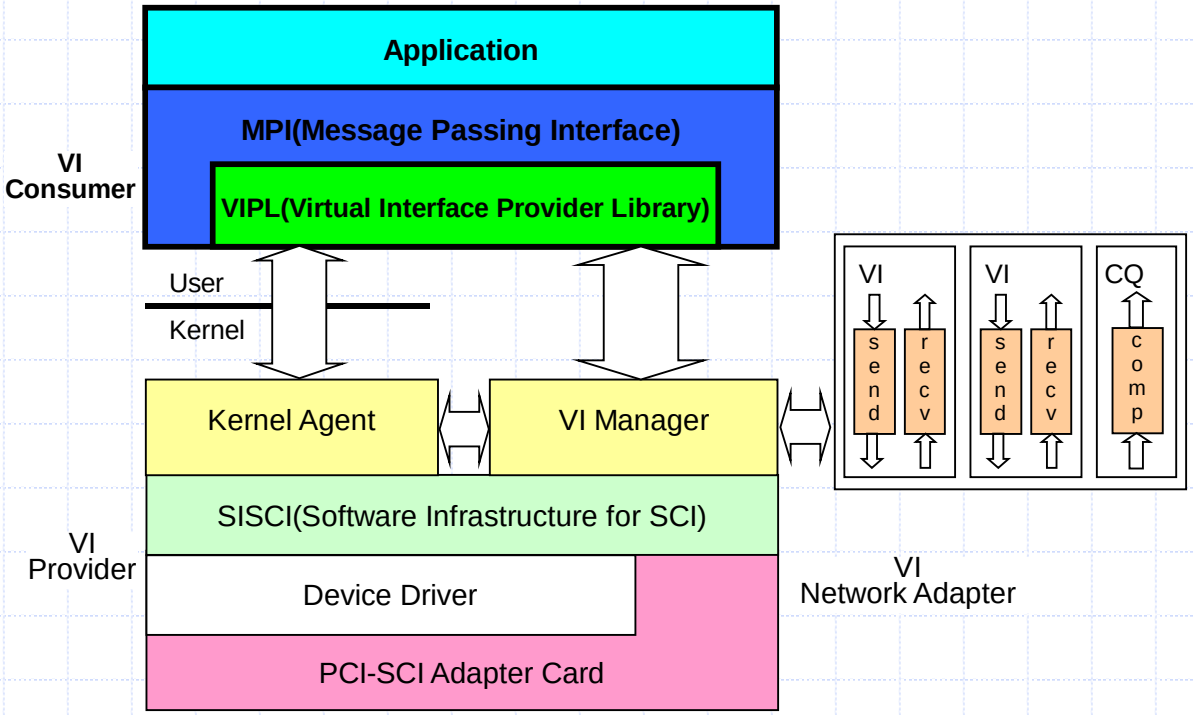
- VI Provider
  - Kernel Agent & VI Network Adapter
- Kernel Agent
  - Setup & resource management functions

Structure of SS-VIA



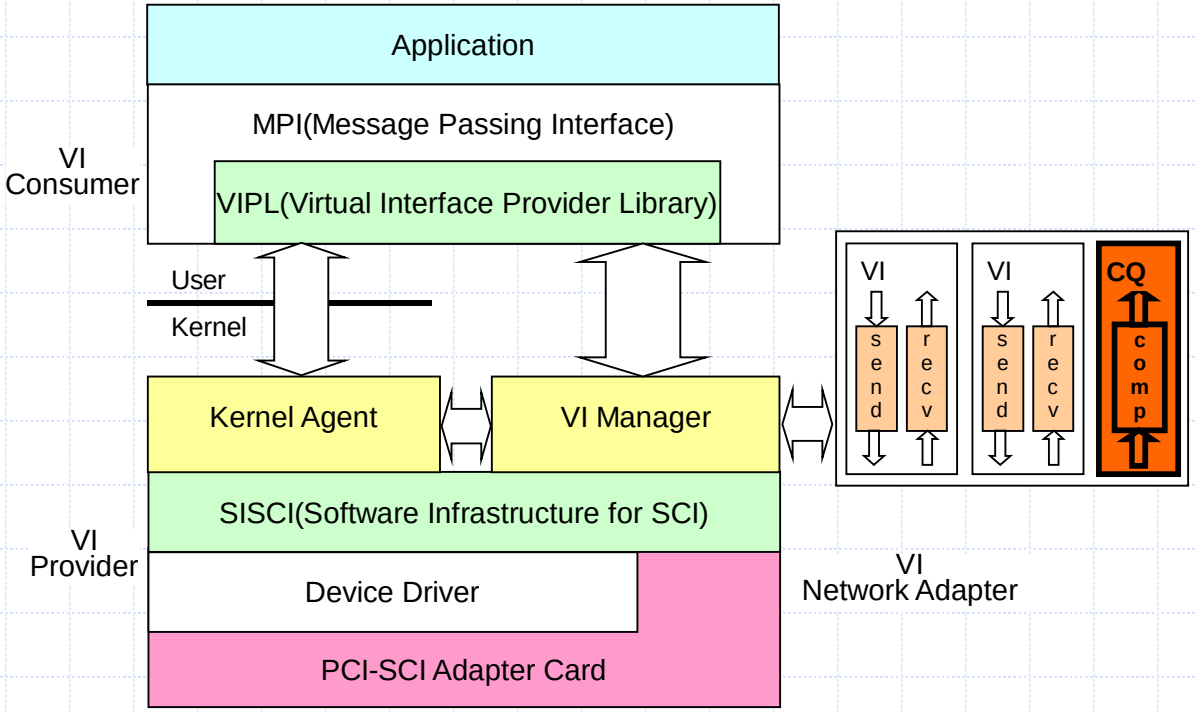
- VI Network Adapter
  - Perform data transfer functions
  - Dolphin's PCI-SCI Adapter Card

Structure of SS-VIA



- VI Consumer
  - VIPL, MPI, and user applications
  - Make system calls to the Kernel Agent to create a VI on a local system & connect it to a VI on a remote system
  - Application's requests are directly posted to the local VI after a connection is established

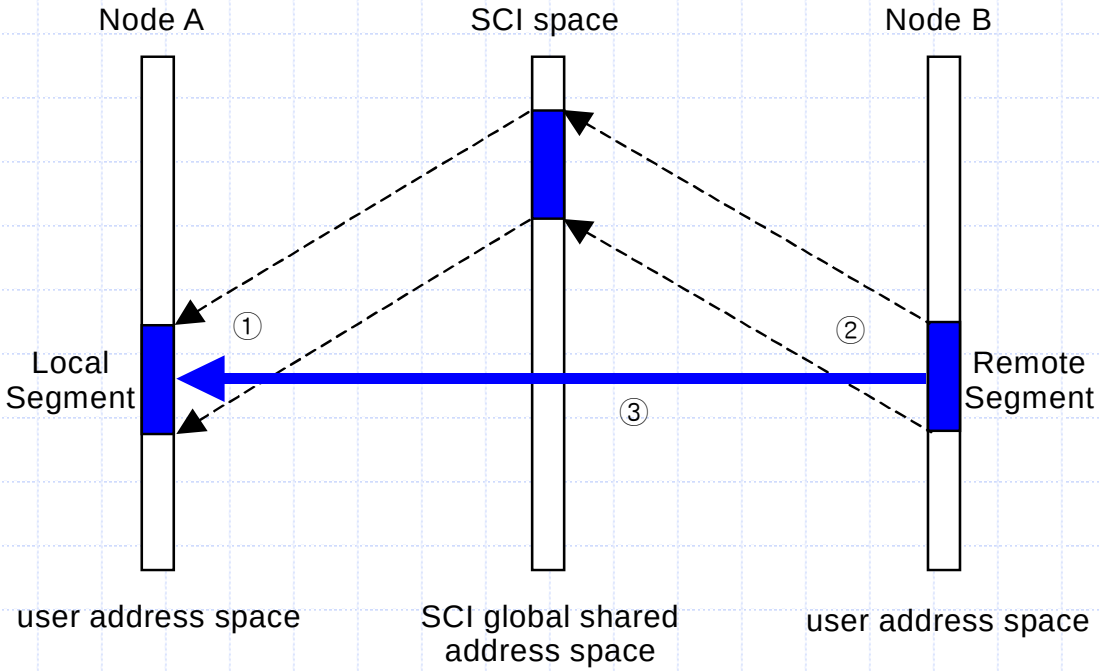
Structure of SS-VIA



- CQ
  - The completion of the request is notified to the CQ
  - Associated with a Work Queue

PCI-SCI Mapping

- Implementing functions for SS-VIA using SISCI API
- Mapping between a local memory and a remote memory
  - The local node can access the remote node's memory directly
  - Communication : Remote Read/Write



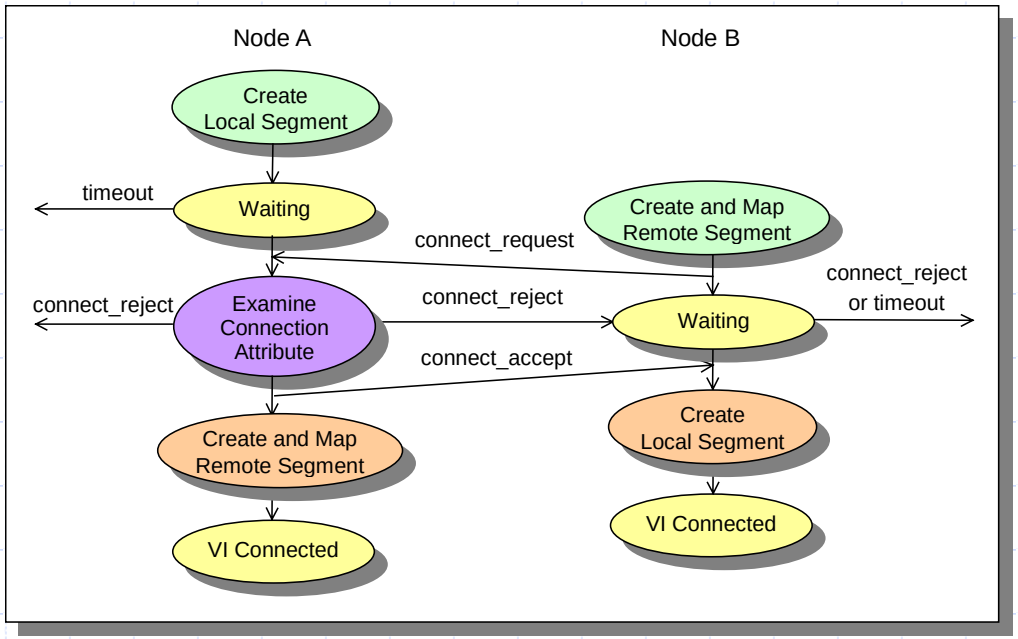
① Allocate the target memory in the user address space of A

② The connection between A and B

③ Directly transfer data from B to A
  - Without extra copies

Initialization Mechanism

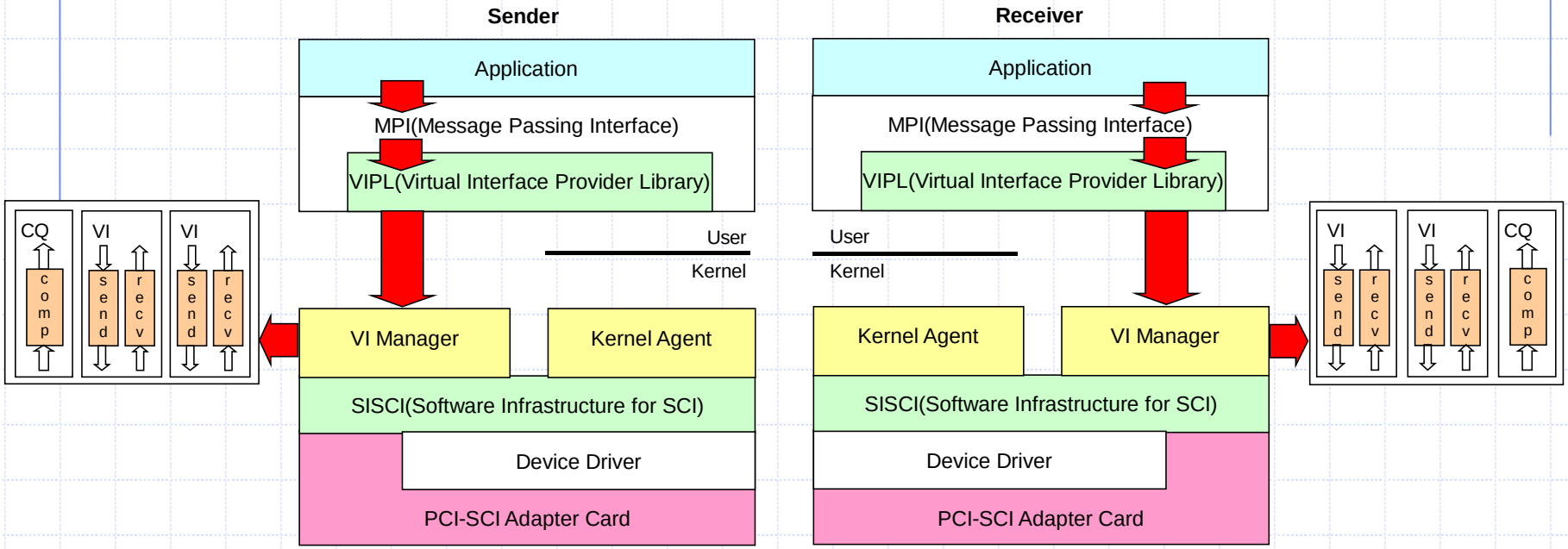
- By the Kernel Agent
  - Request the VI manager to create a local VI
  - Connect it with another VI
  - Only use remote write transaction for data transfer
    - Remote Write < Remote Read



Data Transfer Mechanism

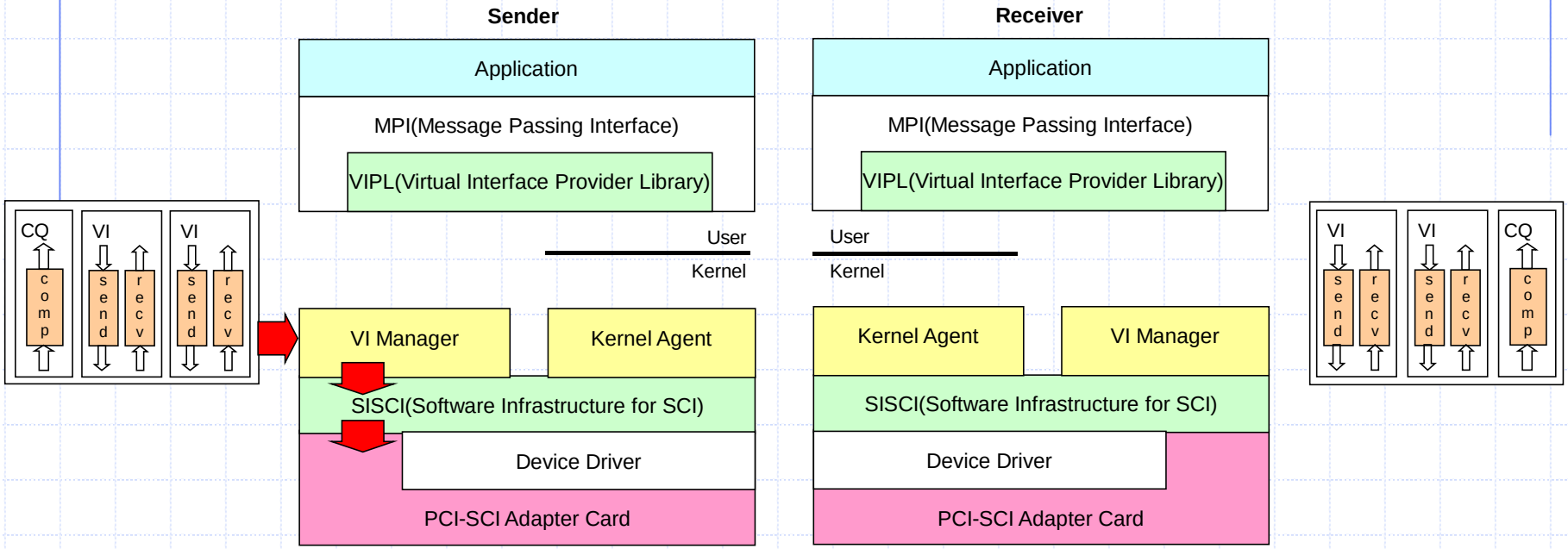
- Transfer data without intervention of the Kernel Agent once the VI connection is completed

Data Transfer Mechanism



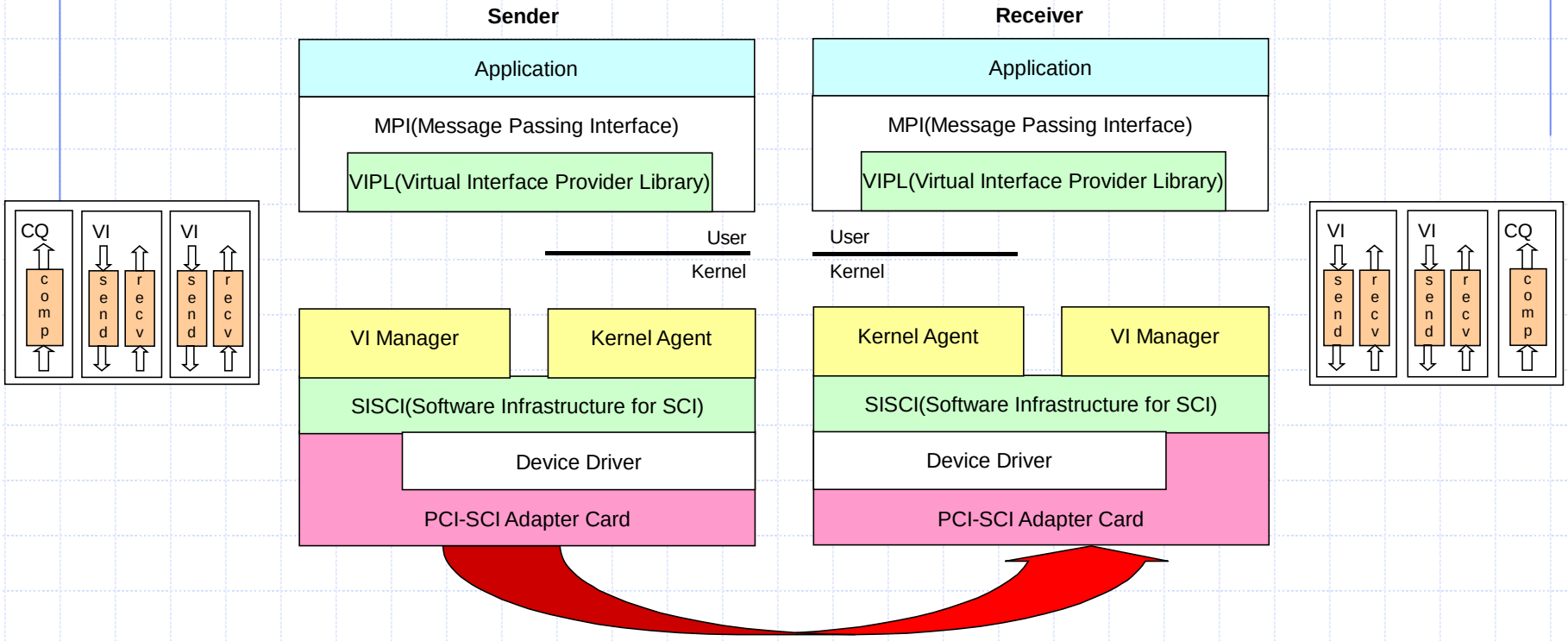
- The sender and the receiver put their descriptor in the Work Queue through the VI manager

Data Transfer Mechanism



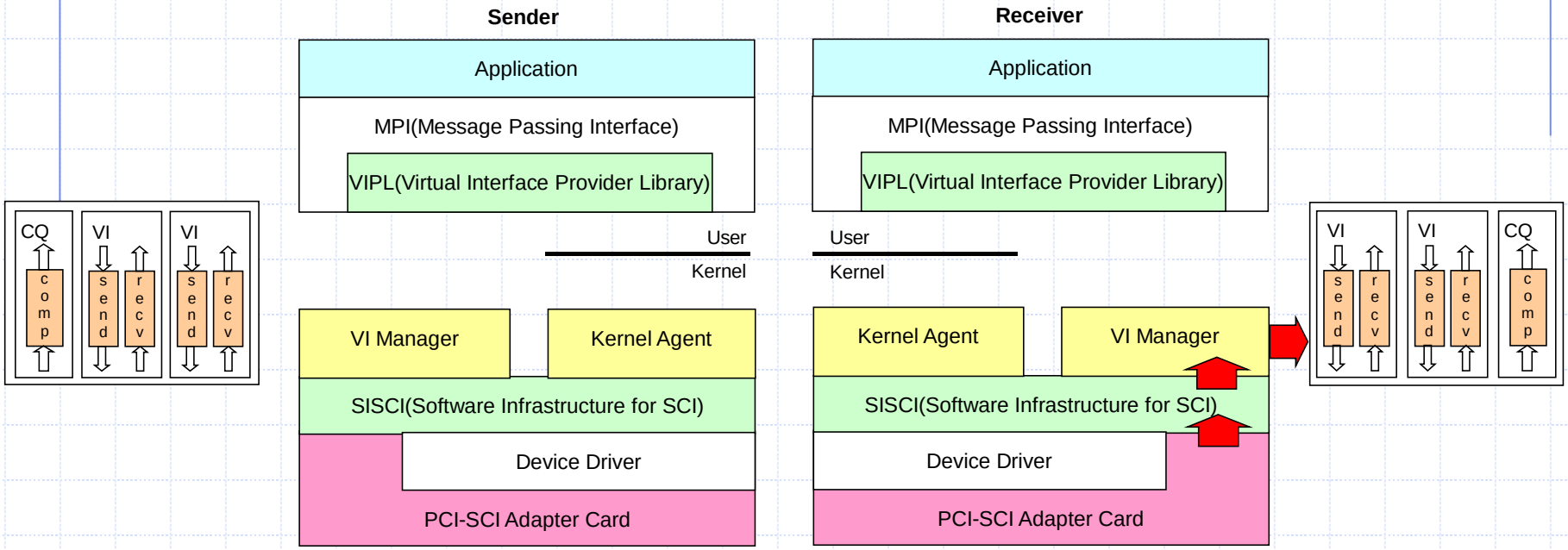
- On the sender side
  - The VI manager picks up the descriptor in the Send Queue
  - Send the data pointed by the descriptor to the VI Network Adapter

Data Transfer Mechanism



- The VI Network Adapter transfers the data to the network
  - remote write

Data Transfer Mechanism



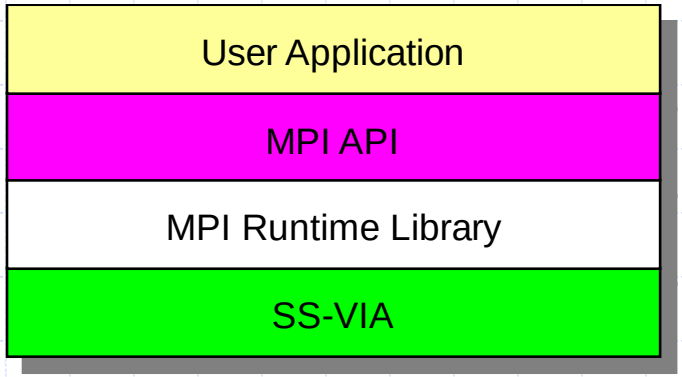
- On the receiver side
  - the data received through the network is stored in the user buffer which is pointed to by the descriptor in the Receive Queue

Data Transfer Mechanism

- Eliminating the time-consuming system call
  - Using the remote memory access mechanism
    - Main feature of NUMA
- Removing the unnecessary data copy in the intermediate buffers
  - Directly move the data between the user buffer and the network interface
- Using the polling mechanism
  - Avoid the context switch overhead

MPI on SS-VIA

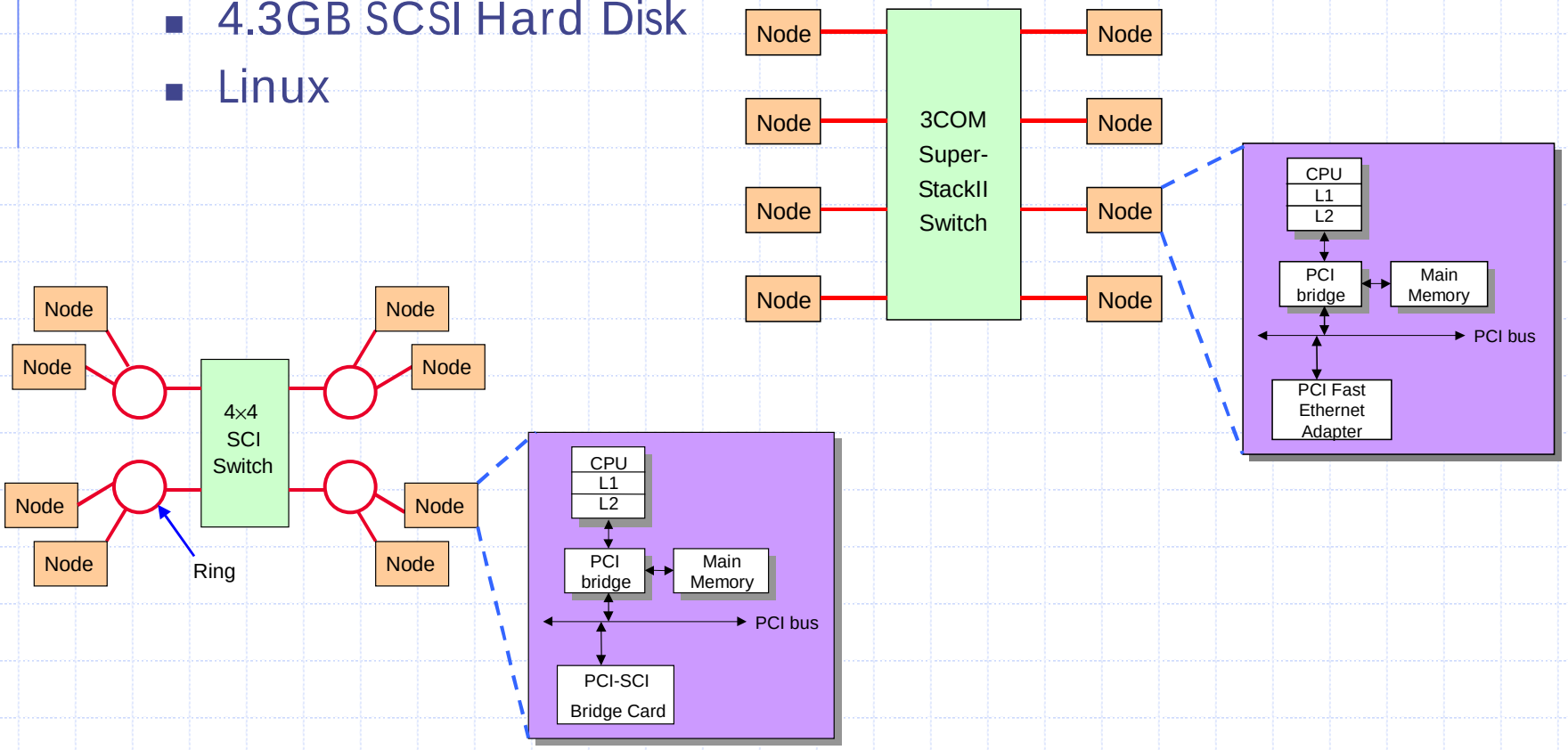
- MPI-2 Standard
- Provide an easy way of programming



EXPERIMENTS

Experimental Environment

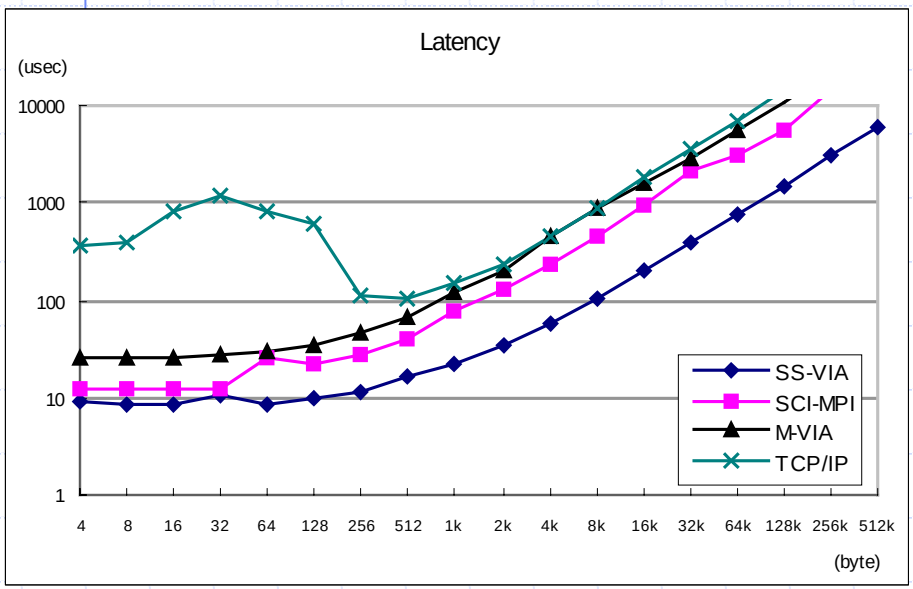
- 8-node PC cluster system
  - 350 MHz Pentium II
  - 128MB Memory
  - 4.3GB SCSI Hard Disk
  - Linux

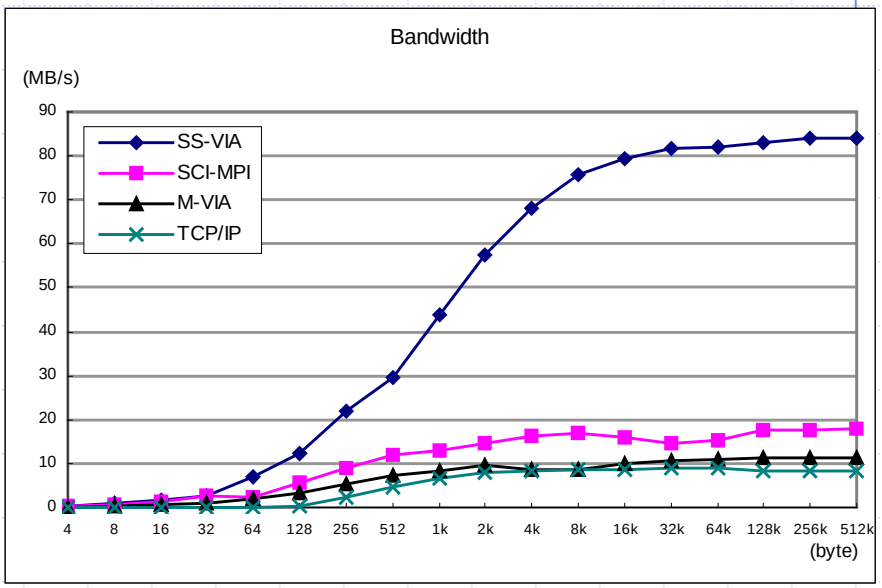


Experiments

- Compared systems
  - M-VIA on Fast Ethernet
    - A software-emulated VIA developed in Lawrence Berkeley Laboratory
    - modular VIA on Linux
  - SCI-MPI
    - Developed in Aachen university
    - Provide MPI on the SCI-based shared memory system
  - TCP/IP on Fast Ethernet

Latency & Bandwidth





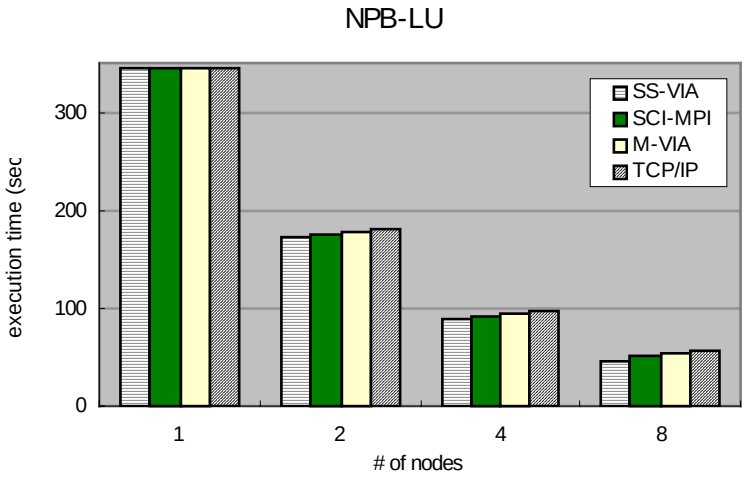
- Other VIA systems
  - M-VIA on Gigabit Ethernet : 19μs, 60MB/s
  - UC Berkeley's VIA on Myrinet : 30μs, 68MB/s
  - Intel's VIA on Myrinet : 53μs, 87MB/s
  - Emulex/Giganet's cLan : 7μs, 110MB/s

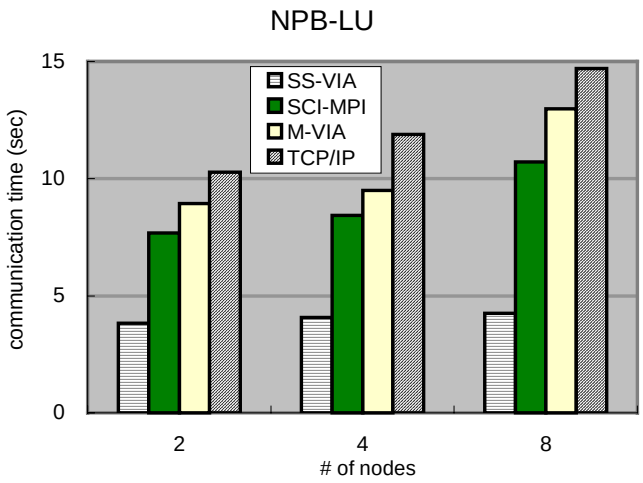
Execution Time of NPB

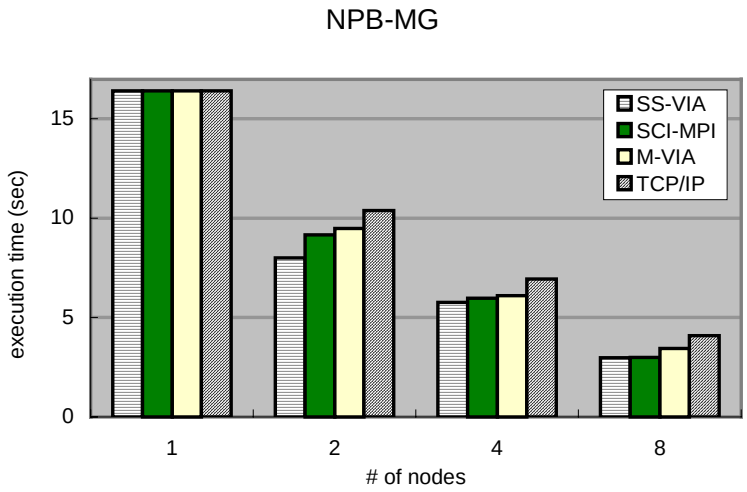
- NAS(Numerical Aerodynamics Simulation) Parallel Benchmark (NPB) version 2.3
  - Developed by NASA

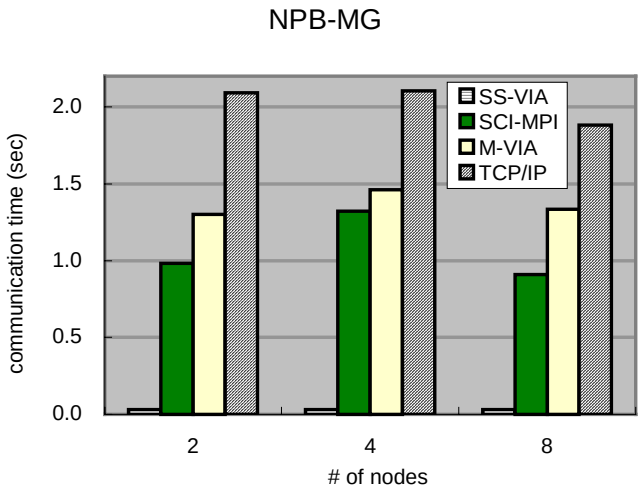
Benchmark Program	Problem Size (byte)	# of iterations
LU solver (LU)	33x33x33	300
Multigrid (MG)	64x64x64	40
Conjugate Gradient (CG)	14000	15
Embarrassingly Parallel (EP)	33554432	0

Execution Time of NPB

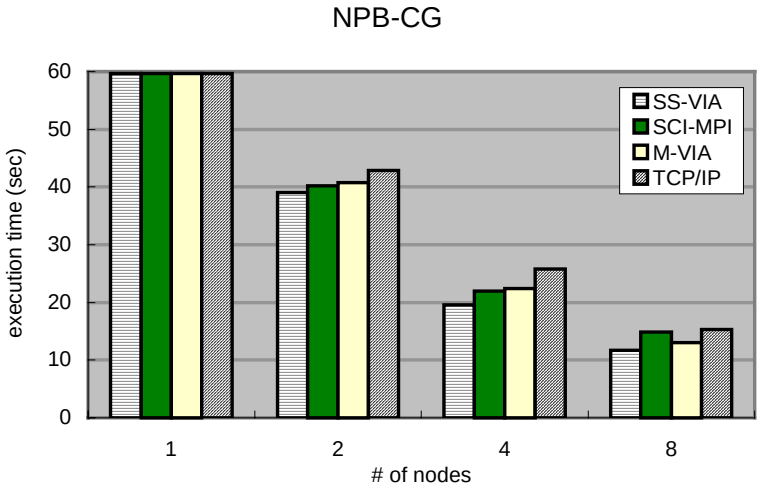


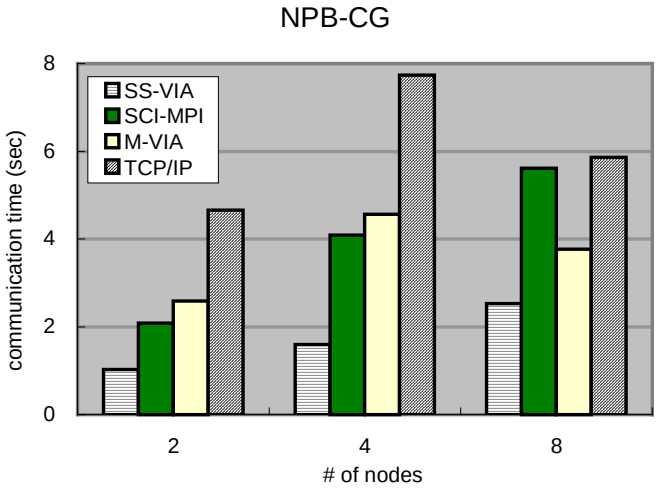


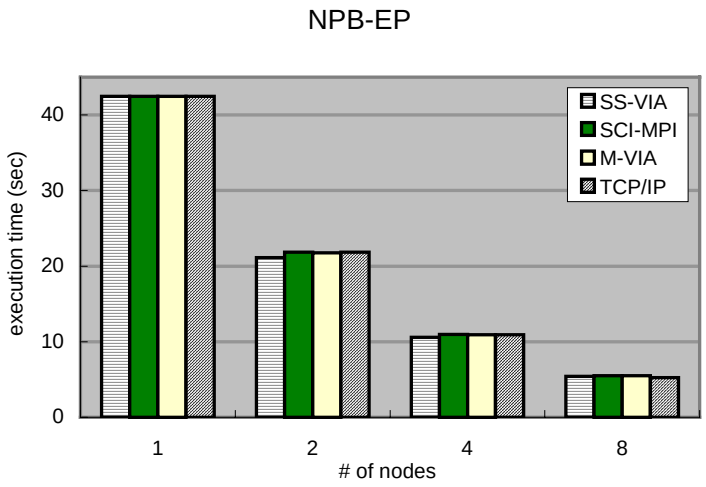


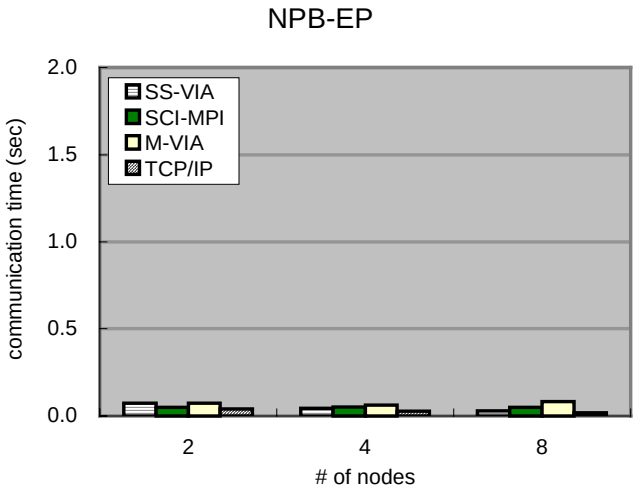


Execution Time of NPB









Conclusion

- SS-VIA
  - VIA has been developed on top of the SCI network based PC cluster system
  - Eliminating intervention of the kernel
  - Bandwidth : 84MB/s Latency : 8μs
  - NAS Parallel Benchmark : average speed-up of 6.1
  - Support both
    - Message-passing programming
    - SCI-based shared-memory programming
  - Guarantee stability
- Future Work
  - Optimization