

VCluster : A Portable Virtual Cluster Computing Library

Presented by: Hua Zhang
Advisors: Dr. Ratan Guha
Dr. Joohan Lee

School of Computer Science
University of Central Florida



Outline

- Cluster Computing Facilities at UCF
- Cluster Computing Research at UCF
 - VCluster: A Portable Virtual Cluster Computing Library
 - Design and architecture
 - Comparison with MPICH, mipJava, JPVM, OpenMP, Pthreads , and Distributed tuple space
 - Performance and experimental studies



Cluster Computing Facilities



Cluster Computing Facilities



Scerola

- # Nodes : 64
- CPU : AMD T-Bird 900MHz
- Memory : 1 GB
- Network : 100BT
- OS : Linux Kernel 2.4.18 (Debian)



Zephyr

- # Nodes : 16
- CPU : AMD T-Bird 900MHz
- Memory : 1 GB
- Network : 100BT
- OS : Linux Kernel 2.4.18 (Debian)



Cluster Computing Facilities



Orange

- # Nodes : 16
- CPU : Dual Intel Pentium III
600MHz processors
- Memory : 1 GB
- Network : 100BT
- OS : Linux Kernel 2.4.22 (RedHat)



Cluster Computing Facilities



Ariel

- # Nodes : 32 (Sun Fire V20z)
- CPU : Dual AMD Opteron 242
1.6GHz processors
- Memory : 2 GB
- Network : Gigabit Ethernet
- OS : SunOS 5.9
- GRID computing infrastructure
 - UCF (32 nodes), FSU (16 nodes), and George Washington Univ. (16 nodes)



VCluster: A Portable Virtual Cluster Computing Library



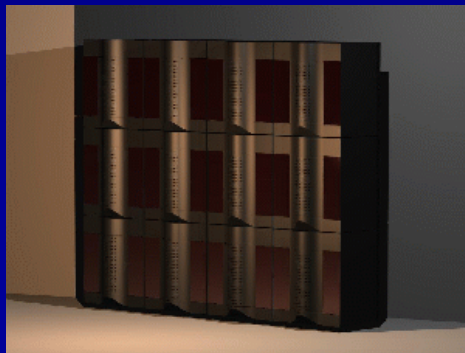
Outline

- Related Research
 - Background
 - Motivations
- VCluster
 - VCluster Model and Architecture
 - Implementation
- Experimental Results
- Conclusion and future work



Parallel Computing

- Developing a parallel program requires a parallel computer architecture, an associated parallel programming model, a language, and software
- **Parallel Architecture**
 - Shared memory multiprocessor
 - Distributed memory multiprocessor (multicomputer)
 - Network of workstations



Parallel Programming Models, Languages, and Software

- **Multithreading and synchronization primitives**
 - Pthreads, Quick Threads, system specific threads, etc.
 - User level vs. system level
 - Thread management and synchronization (semaphore, monitor)
- **OpenMP**
 - Compiler directives and library routines for shared memory parallelism for C and Fortran
- **Tuple Space**
 - Virtual associative memory shared by processes
 - Use tuple space for coordination and synchronization
 - Linda, JavaSpaces, IBM TSpace



Parallel Programming Models, Languages, and Software

- **Distributed Shared Memory (DSM)**
 - Provides a global shared address space across the different machines on a cluster
 - TreadMarks
- **Message Passing**
 - Each processor executes a different stream of instructions and exchanges messages when they need to share data or coordinate/synchronize with other processors
 - MPI, PVM, P4



Cluster Computing

- Become popular with the availability of
 - High performance microprocessors
 - High speed networks
 - Distributed computing tools
- A low cost parallel computing platform consisting of multiple interconnected PCs or workstations through a commodity network technology as a replacement for an expensive parallel computer



Issues in Cluster Computing

- New requirements for parallel programming libraries (or run time system or middleware)
 - Emerging cluster computing architecture, *a cluster of SMPs*
 - Scalability / Communication latency
 - Heterogeneity
 - Integrating multithreading and communication
 - Reliability
 - Load balancing / Resource management
 - Portability / Maintainability
 - Security
 - Grid computing



Issues in Cluster Computing

- Scalability and Communication latency
 - Thousands of nodes
 - Employ faster networking technologies
 - ATM network
 - Myrinet
 - Gigabit Ethernet
 - Use of low overhead protocols
 - Reduce the host/interface overhead
 - Symmetric Multiprocessor (SMP) machines

An Emerging Cluster Computing Architecture

- A Cluster of Symmetric Multiprocessor (SMP) machines
- SMP machine
 - Usually up to eight processors that share the same memory, I/O devices, and other resources and run different copy of the operating system on each processor
- Cluster computing with SMPs
 - Must support shared memory model and message passing model at the same time
 - Reduces the amount of network communication over the network

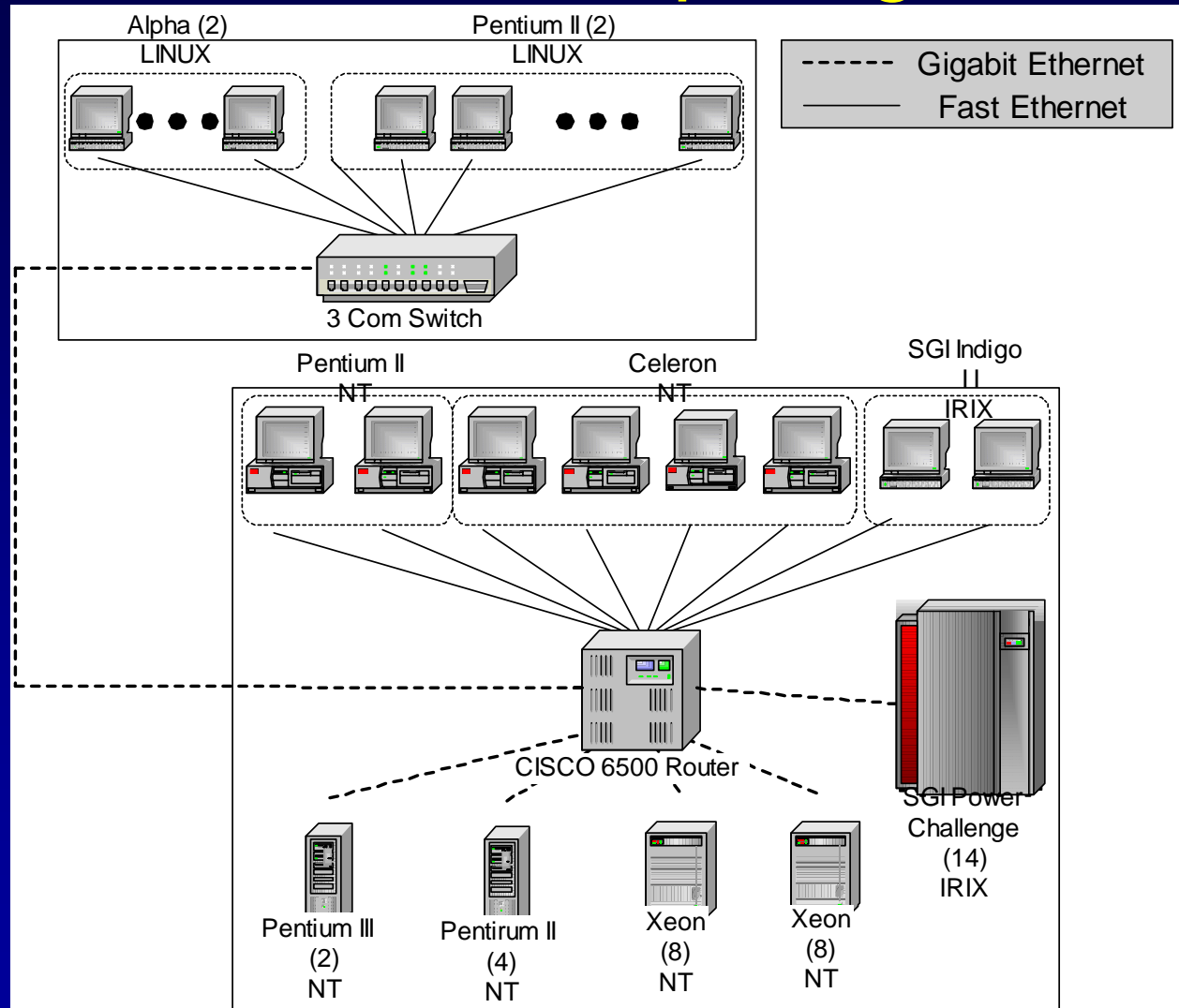


An Emerging Cluster Computing Architecture

- Current Systems for SMP machines
 - Shared Memory Model
 - MPI and PVM are based on a message-passing model
 - Multithreading
 - None of MPI, PVM or p4 supports multithreading inherently, a separate thread library must be used
 - None of MPI, PVM or p4 supports communication between threads

Issues in Cluster Computing

- Heterogeneity



Issues in Cluster Computing

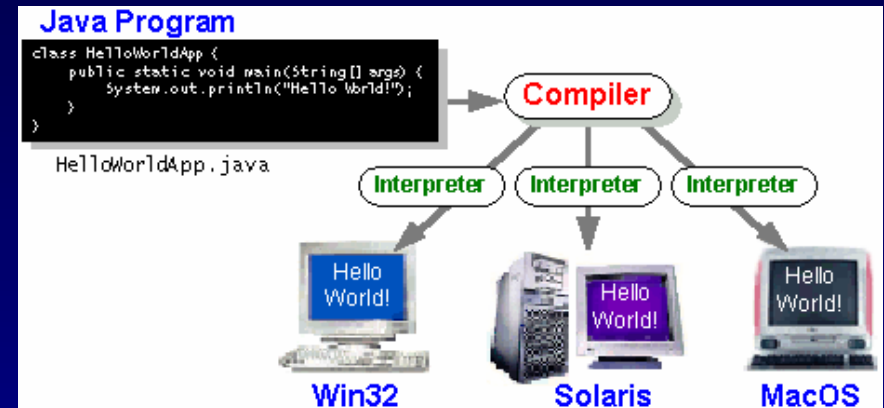
- Portability and maintainability
 - Higher performance has been the main goal, however, software reusability, portability, and maintainability have become important
 - Heterogeneous machines
 - Different number representations
 - Big endian and little endian
 - Different multithreading models and implementations
 - System dependent features
 - User need to handle these differences or even need to maintain a different copy of parallel program for each type of machine



Issues in Cluster Computing

- Java based approach is attractive

- Platform neutral byte codes
- Object oriented paradigm
- Portable thread support
- Advanced synchronization primitives based on monitor
- Java security features : sandbox, JCE
- Seamless integration with the Web
- Drawbacks:
 - Performance



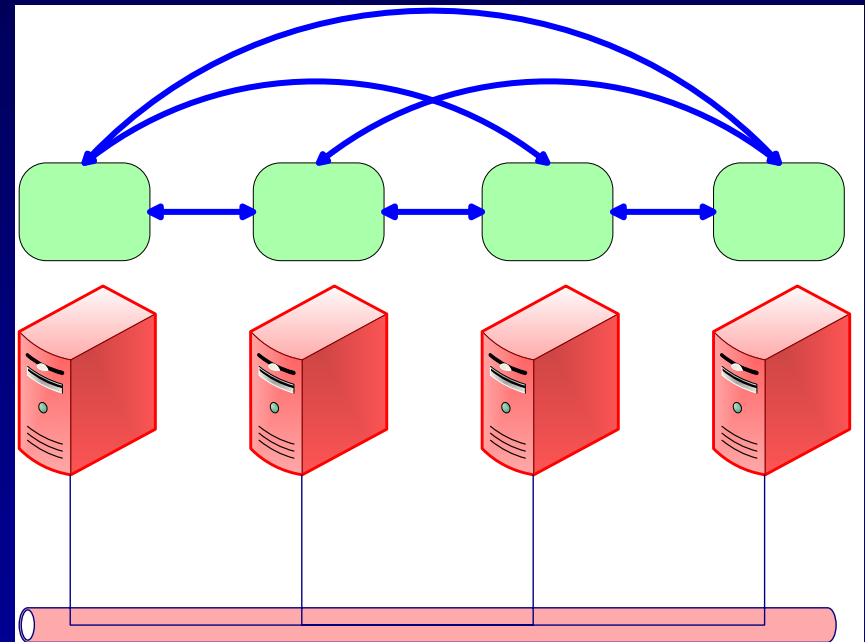
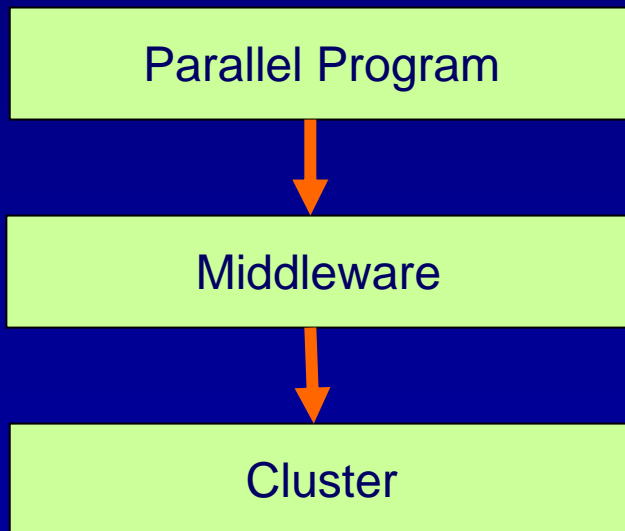
- Portability and maintainability of the current systems

- Some Java implementations of MPI and PVM
- mpiJava, JPVM, JavaSpace, IBM TSpace



Message Passing Based Cluster Computing

- To run a parallel program on a cluster
 - Processes are created on the machine in the cluster
 - Processes communicate with each other



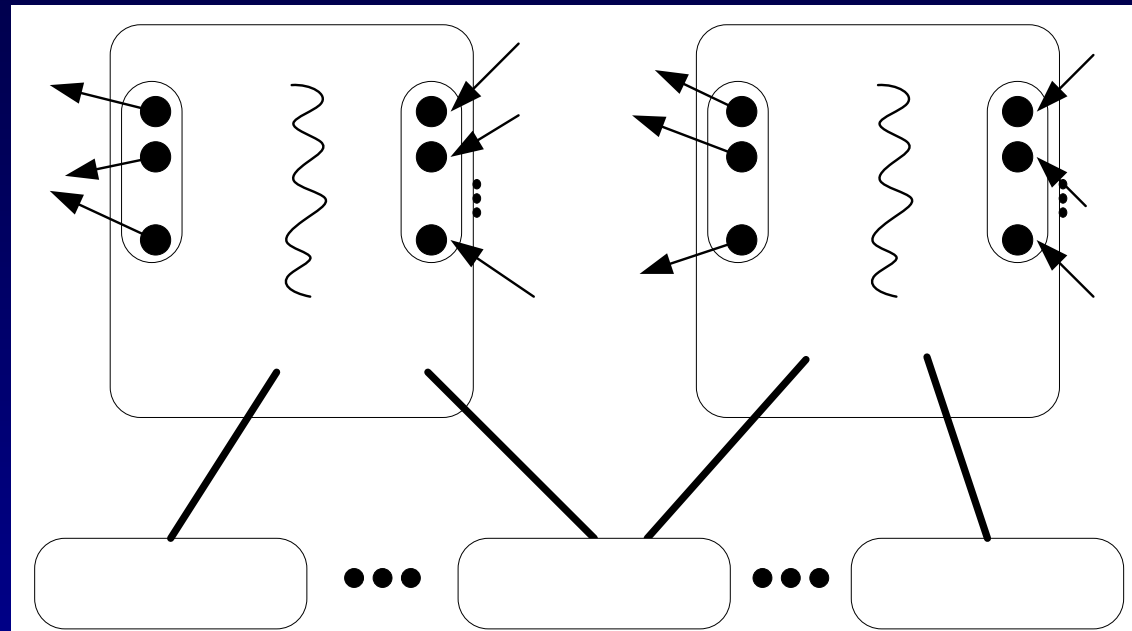
VCluster : A Portable Parallel Runtime System for Virtual Cluster Computing

- Motivation

- Address the issues mentioned previously
- Especially support a cluster of heterogeneous SMP Machines
 - Combines shared memory model and message passing
 - Thread based communication
- Portability and maintainability
 - 100% implemented in Java
 - Clusters of heterogeneous machines
- Load balancing
 - Virtual thread migration
- Reliability
 - Fault tolerance based on checkpointing
- Security
 - Connecting multiple separate clusters securely



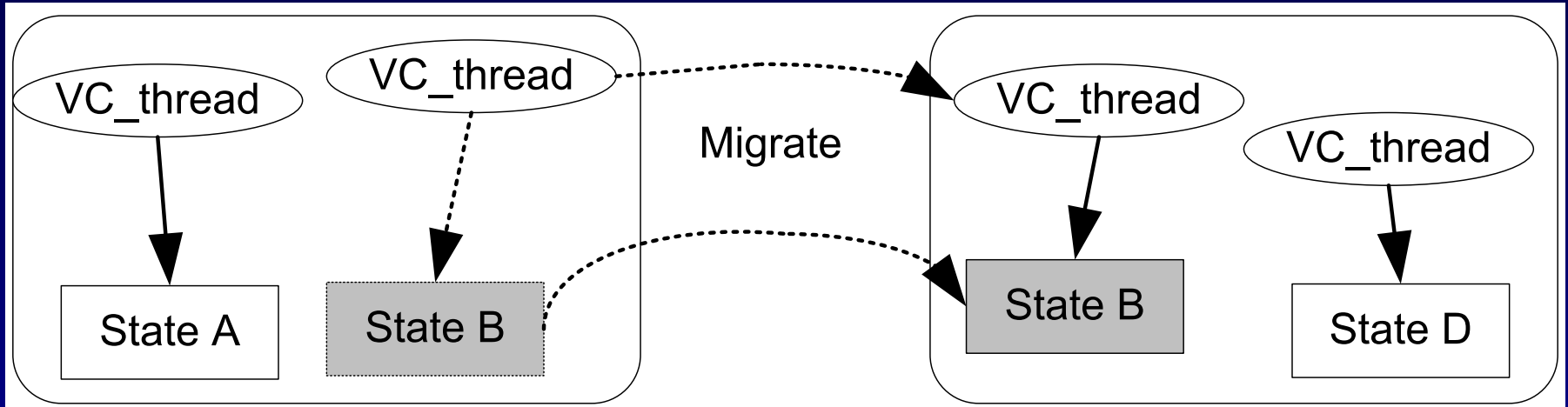
VCluster Model



- Thread is the basic computing unit instead of process
- Unidirectional communication channels between threads
- A thread associate itself with a set of states
- Two or more threads can share one state, another way of shared memory communication
- Programmers write the parallel program by extending VCluster thread, state, channel, and channel set classes



Virtual Thread Migration



- Decouple the thread state from the local JVM
- A state is transferred to another machine and a new virtual thread is created based on the state
- A virtual thread is virtual in that it can migrate within the system from machine to machine

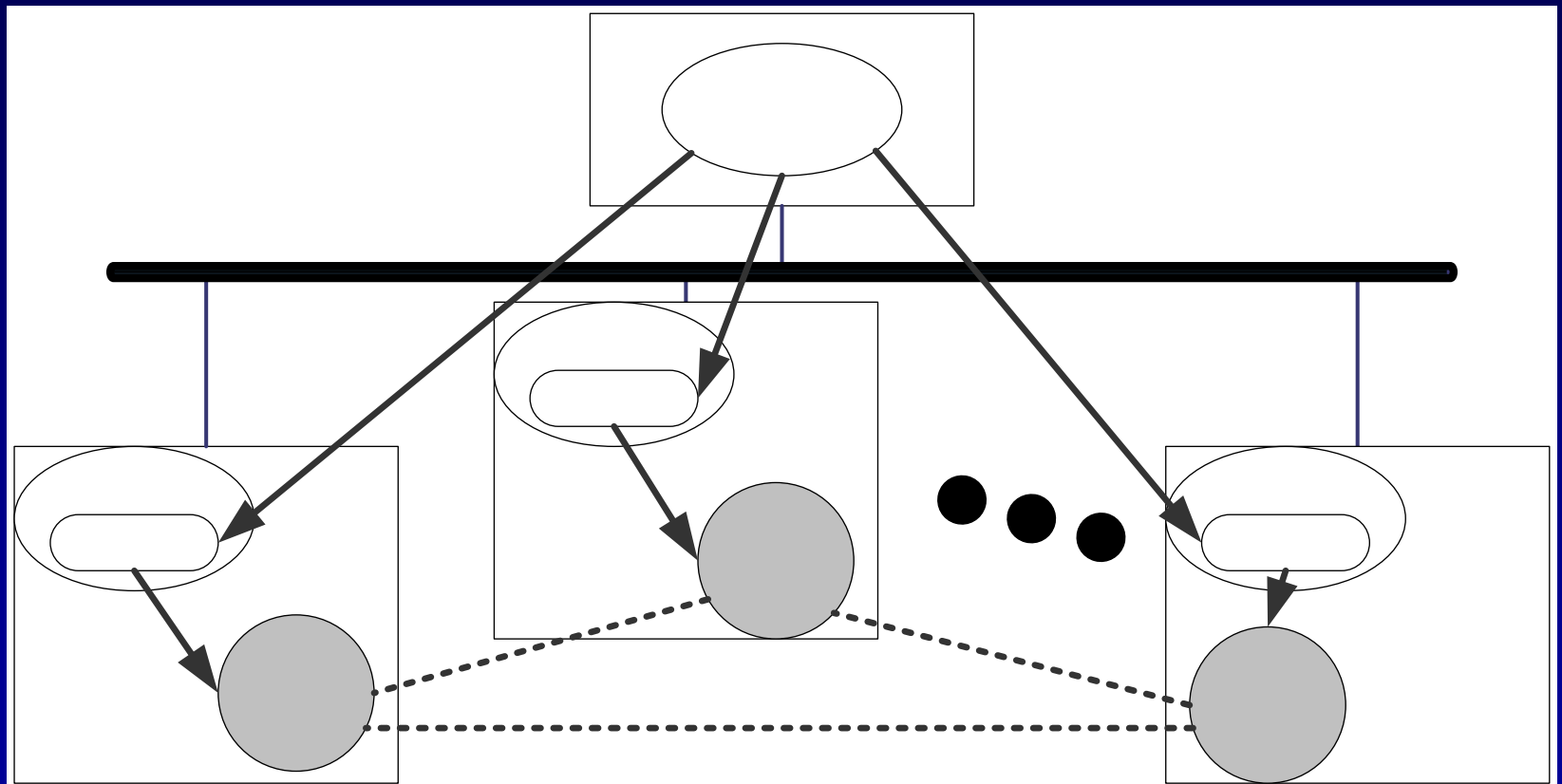


VCluster Implementation

- A daemon is running on each node
 - Process creation and termination
 - Output replay
- Initialization Process
 - Use of a static process group file
 - A daemon process waits for a request
 - A service thread is created for each request
 - Service thread spawns the requested process
 - Spawned processes establish fully connections
 - Service threads relay output to the root process



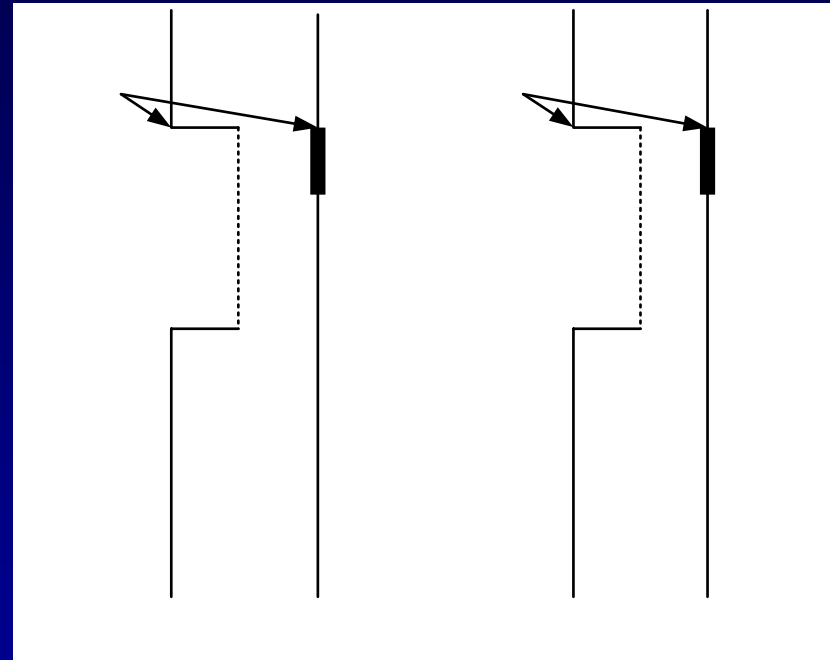
VCluster Implementation



VCluster Implementation

- **Asynchronous I/O**

- Every process has a send thread and a receive thread
- Receive thread uses asynchronous I/O via JNI to reduce the asynchronous message passing overhead

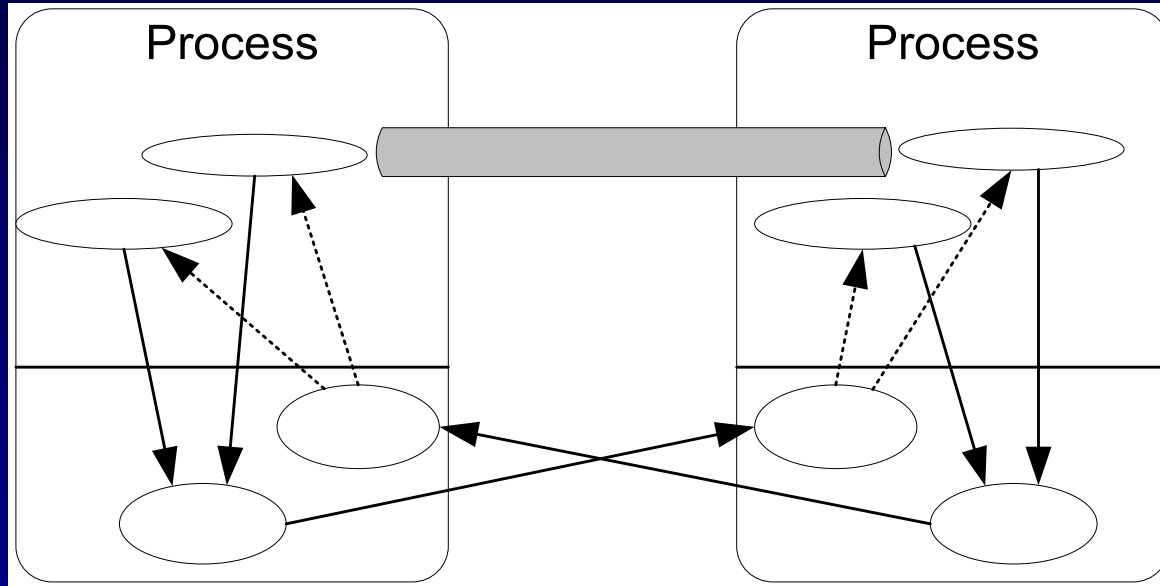


- **Advantages of separate Send/Recv threads**

- Reduce the possible blocking overhead
- Support rendezvous style communication



Virtual Communication Channel



Virtual C

- Thread to thread virtual communication channels
- Communication structure is maintained in the presence of thread migration
- It can be between two threads on the same or different machines



Application Structure

- **Main Class**
 - Initialize VCluster
 - Create threads
 - Create states
 - Create channels between threads
 - Start the computation

- **Thread Class**
 - Extends VC_Thread
 - Carry out the computation in

```
public void run() {  
    }  
}
```



Application Structure

- Main Class

- Initializes the VCluster

```
vc = new VCluster();
```

- Creates threads and binds them to VCluster

```
vthr = new myThread();  
vc.addThread(vthr);
```

- Create states and bind them to the vthreads

```
mystate = new myState();  
vthr.addState(mystate);
```

- Create Channels between vthreads

```
vchannelSet = new C_ChannelSet("sendChannelSet");  
vchannel = new VC_Channel(1,"ch", vc.VC_WRITE);  
vthr.addChannelSet(vchannelSet);
```

- Start the computation

```
vc.start();
```



Application Structure

- Thread Class

- User need to implement the run() method

```
class myThread extends VC_Thread{  
    public void run() {}  
}
```

- Communication

- Get Channel

```
vchannelSet = getChannelSet("sendChannelSet");  
VC_Channel vchannel = vchannelSet.getChannel("ch");
```

- Send Message

```
vchannel.writeInt(888);
```

- Receive Message

```
int a = vchannel.readInt();
```



Experimental Results

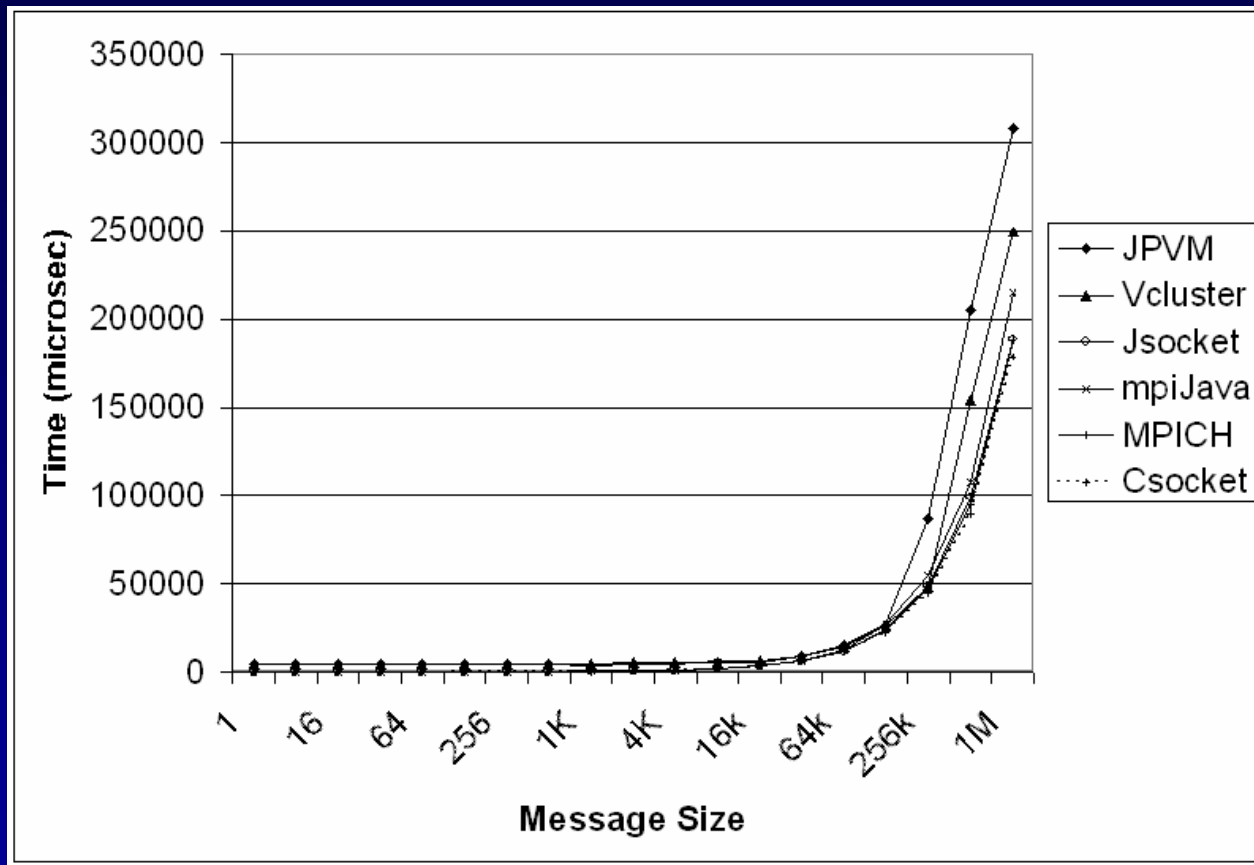


Experimental Results

- Comparison with other relevant systems
 - MPICH
 - A C implementation of MPI
 - MPICH with pthread
 - Combining MPI with multithreading
 - MPICH with OpenMP
 - Combining MPI with OpenMP
 - mpiJava
 - A java wrapper of MPI
 - JPVM
 - A pure java implementation of PVM specification
- Performance
 - Communication performance
 - Parallel application performance



Communication Overhead



- Measure the roundtrip time for a message between two processes on different machines



Communication Overhead

- VCluster outperforms JPVM for larger message sizes
- Java based systems are slower, but
 - Coarse grain parallelism is mostly used in cluster computing so the communication overhead can be compensated by the larger computation part



Parallel Dirichlet Problem

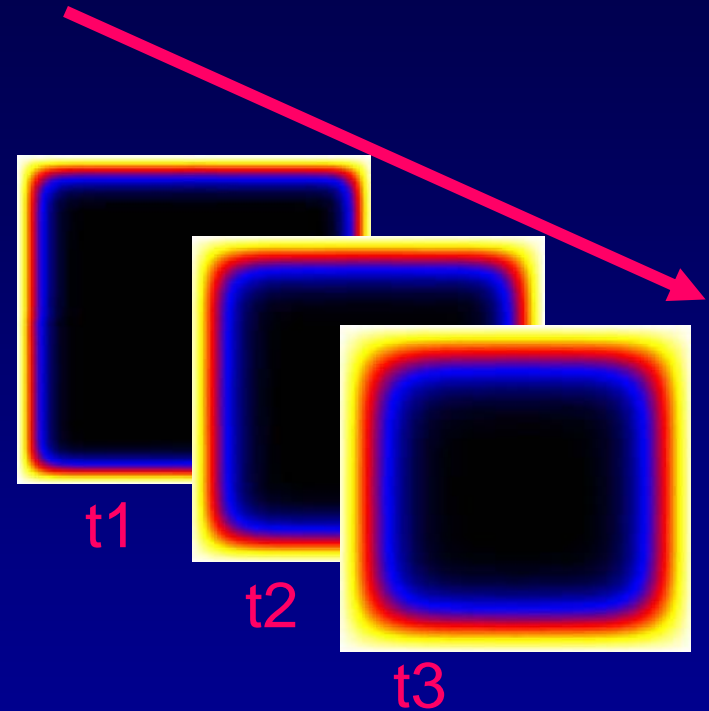
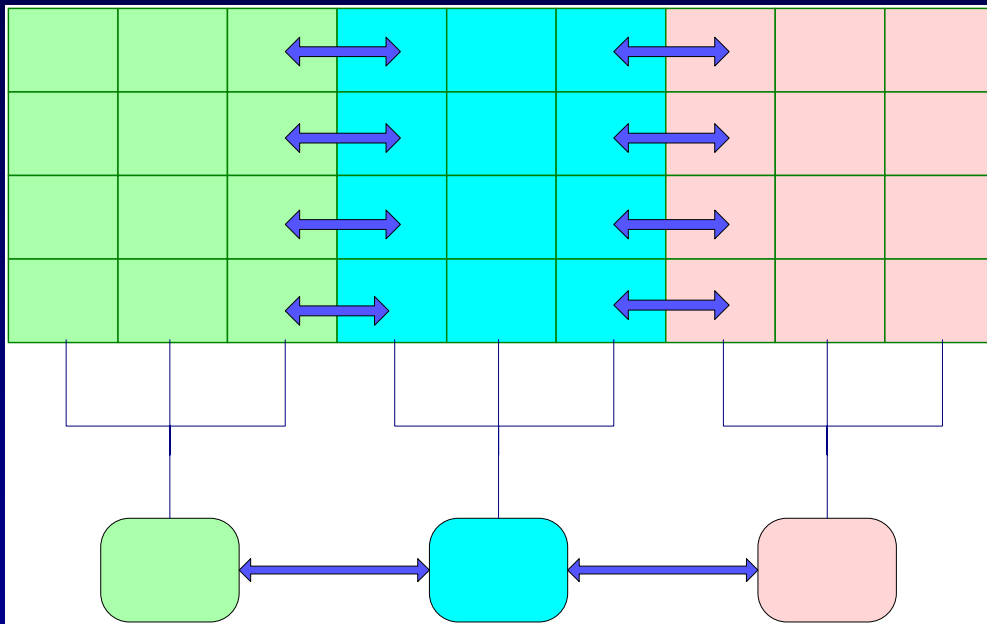
- Dirichlet Problem

- A simple numerical simulation problem on a two dimensional grid. Each point on the grid has a location (x, y) and a value $T(x, y)$ representing temperature of some material
- At each time step, temperature of each point is averaged with its neighbors' temperatures to find the temperature of the next time step

$$T(x, y, t + 1) = \frac{T(x, y - 1, t) + T(x + 1, y, t) + T(x, y + 1, t) + T(x - 1, y, t) + T(x, y, t)}{5}$$



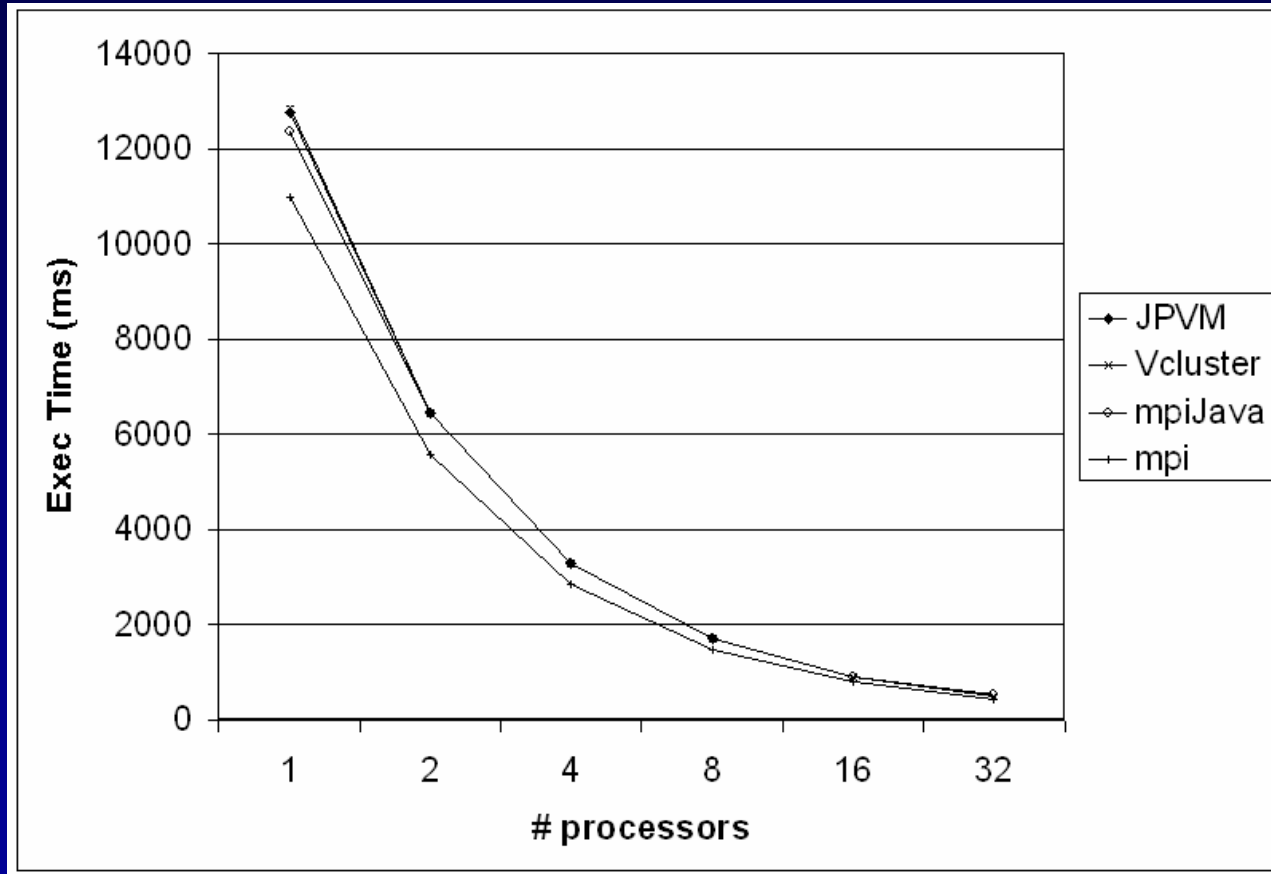
Parallel Dirichlet Problem



- One dimensional decomposition of the grid
- Neighbor processes need to communicate



Parallel Dirichlet Problem



- Scerola cluster : homogeneous uni-processor cluster



Application Performance

- Parallel Dirichlet Problem
 - Java based libraries have a comparable performance
 - With enough number of processors, Java based message passing libraries can mitigate the associated overhead compared with C based libraries and be an attractive tool for a large scale parallel application development

Parallel BPNN

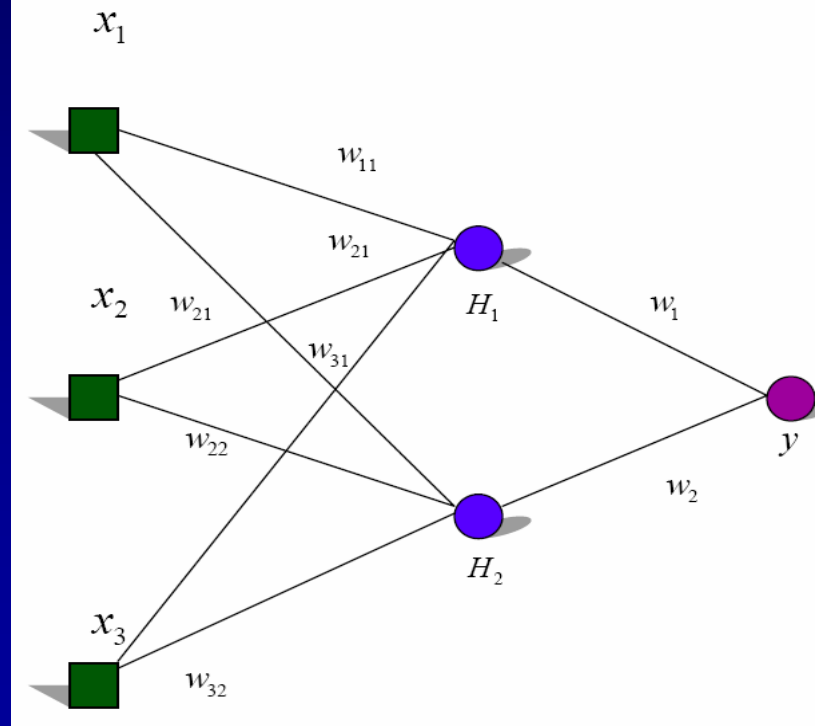
- Back Propagation Neural Network

- Predict the value of y for a given input (x_1, x_2, \dots, x_n)
- Known $(x_1, x_2, \dots, x_n, y)$ records are used to train the network and find optimized weights
- Normally requires a number of iterations of training

$$g_0^{-1}(E(y)) = w_0 + w_1 H_1 + w_2 H_2$$

$$H_1 = g_1(w_{01} + w_{11}x_1 + w_{21}x_2 + w_{31}x_3)$$

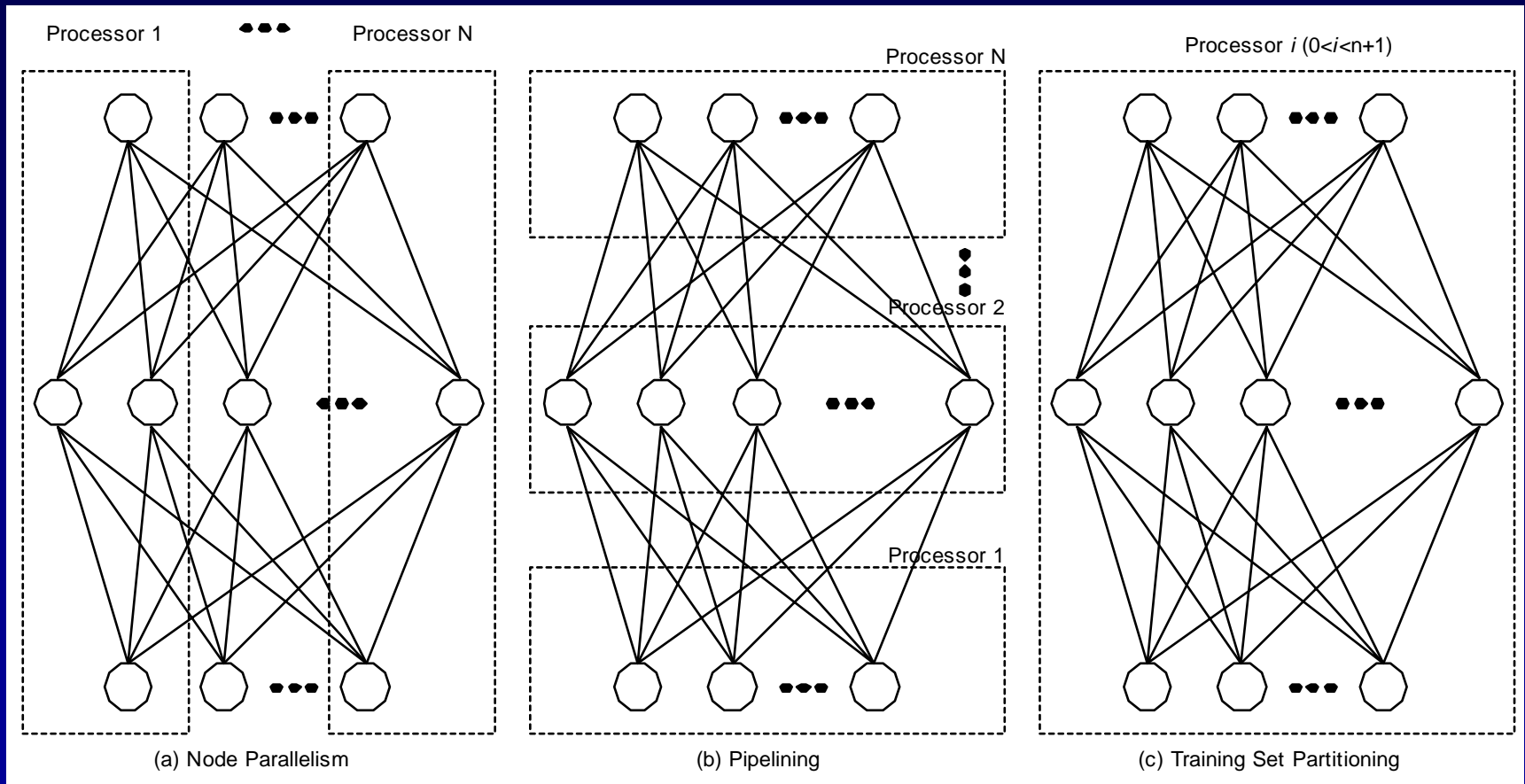
$$H_2 = g_2(w_{02} + w_{12}x_1 + w_{22}x_2 + w_{32}x_3)$$



Parallel BPNN

- Back Propagation Neural Network
 - one of the most popular neural network training algorithms and has shown robust performance in many diverse applications
 - However, computational complexity of the BPNN makes its use challenging especially when the training data set size is huge

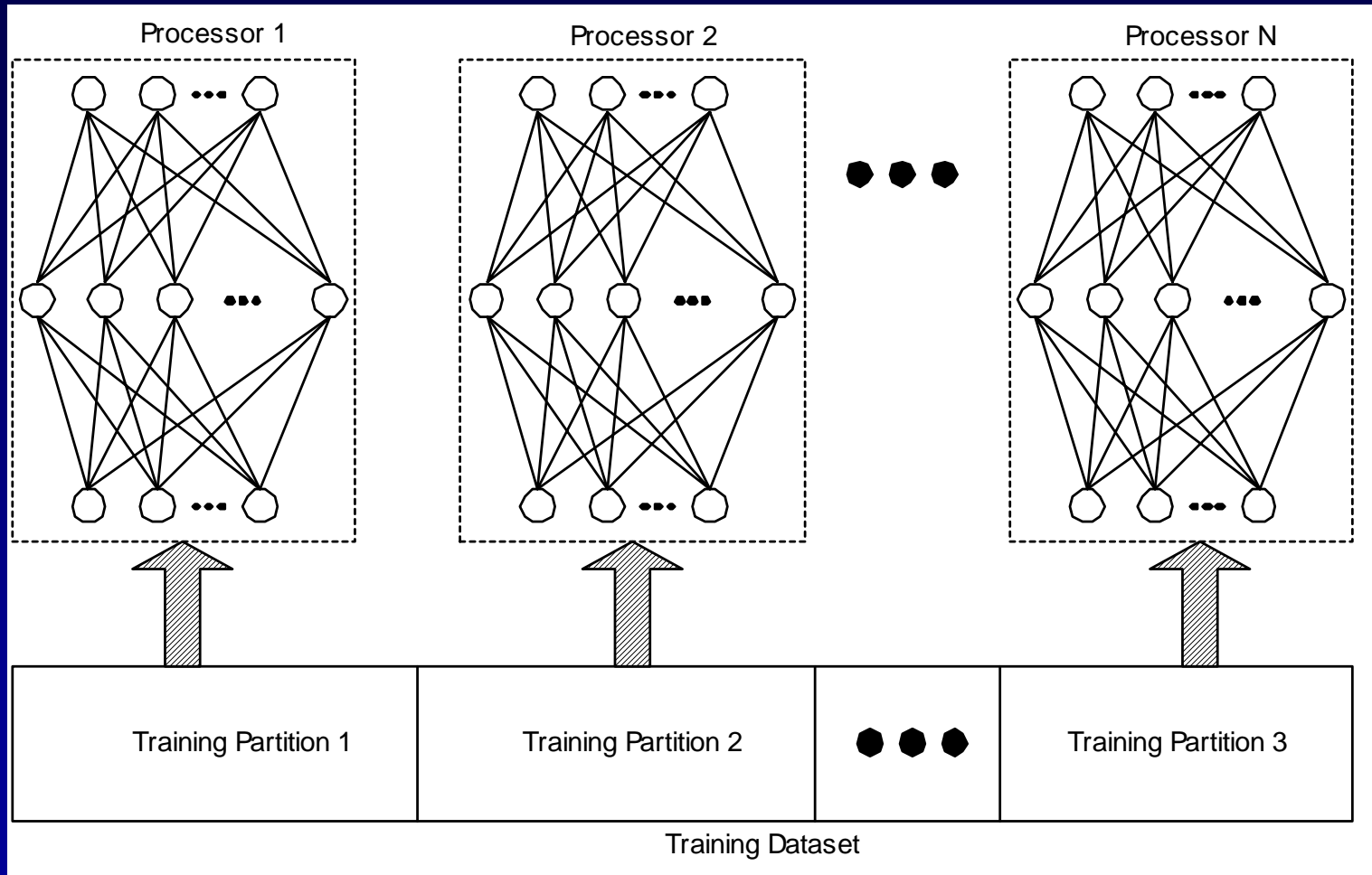
Parallel BPNN



Different parallelization strategies



Parallel BPNN



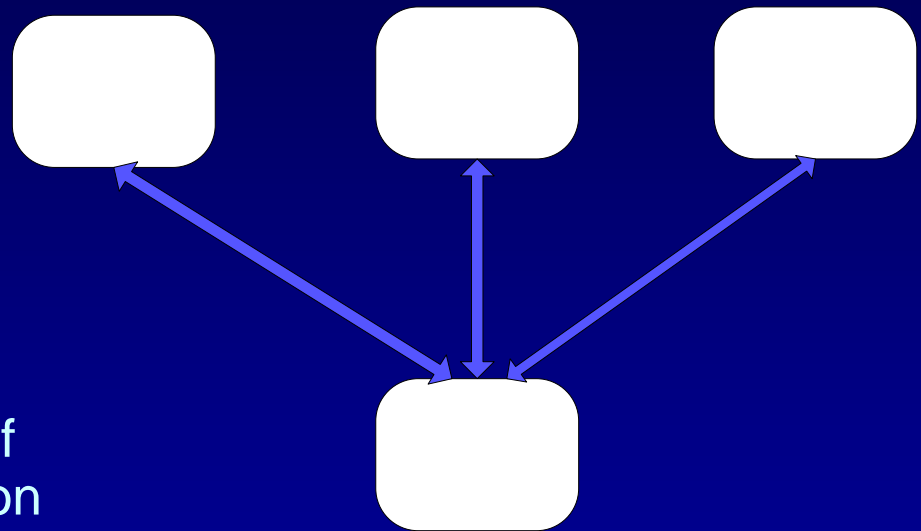
Training Set Partitioning



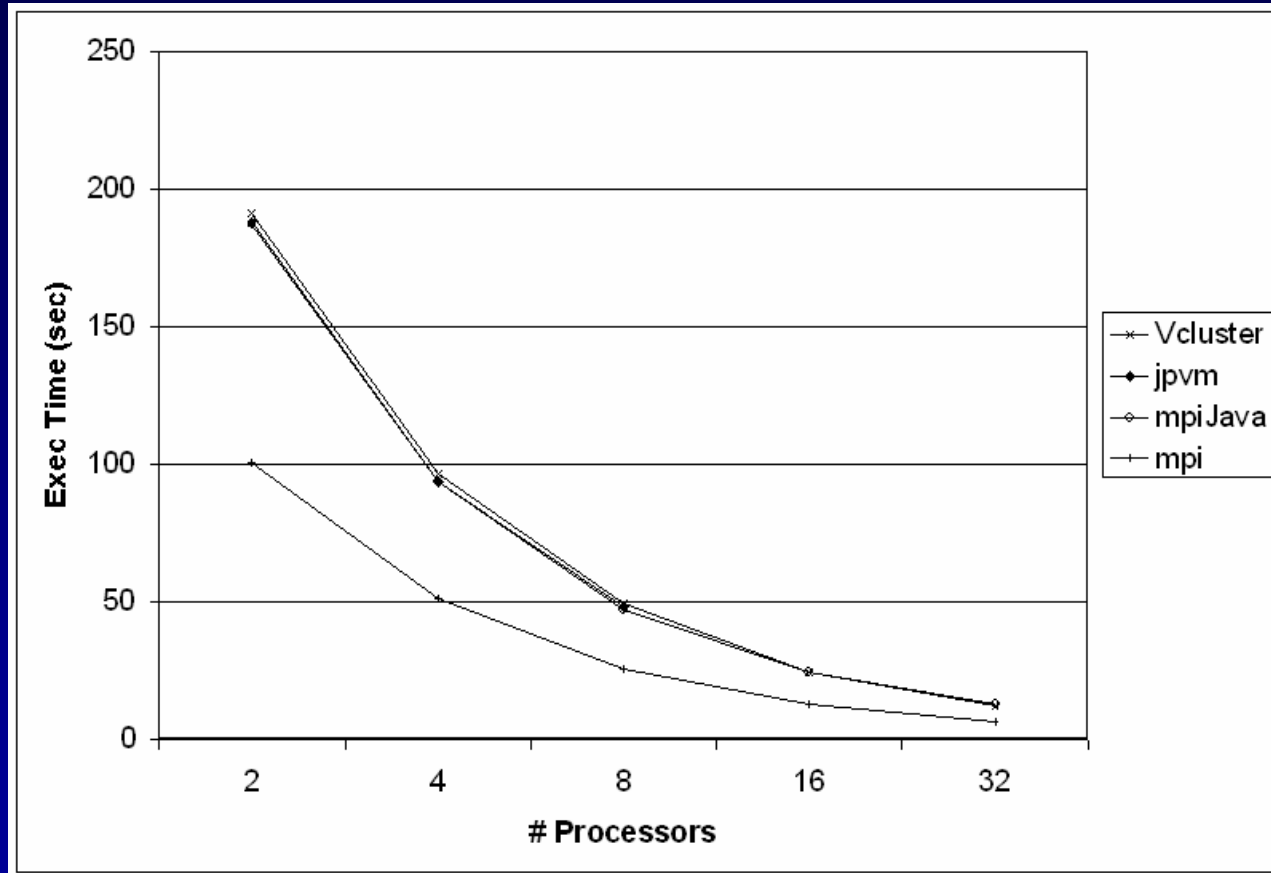
Parallel BPNN

- **Parallel Back Propagation Neural Network**

- Divide training data set into smaller sets
- Each process trains the network use a smaller set
- The master process collects and redistributes the new weights after each iteration of training, so the communication is between the master process and other worker processes



Parallel BPNN



- Scerola cluster : homogeneous uni-processor cluster



Parallel BPNN

- Parallel Back Propagation Neural Network
 - Java based systems have a comparable performance
 - The gap between VCluster and MPICH converges as the number of processors increases again

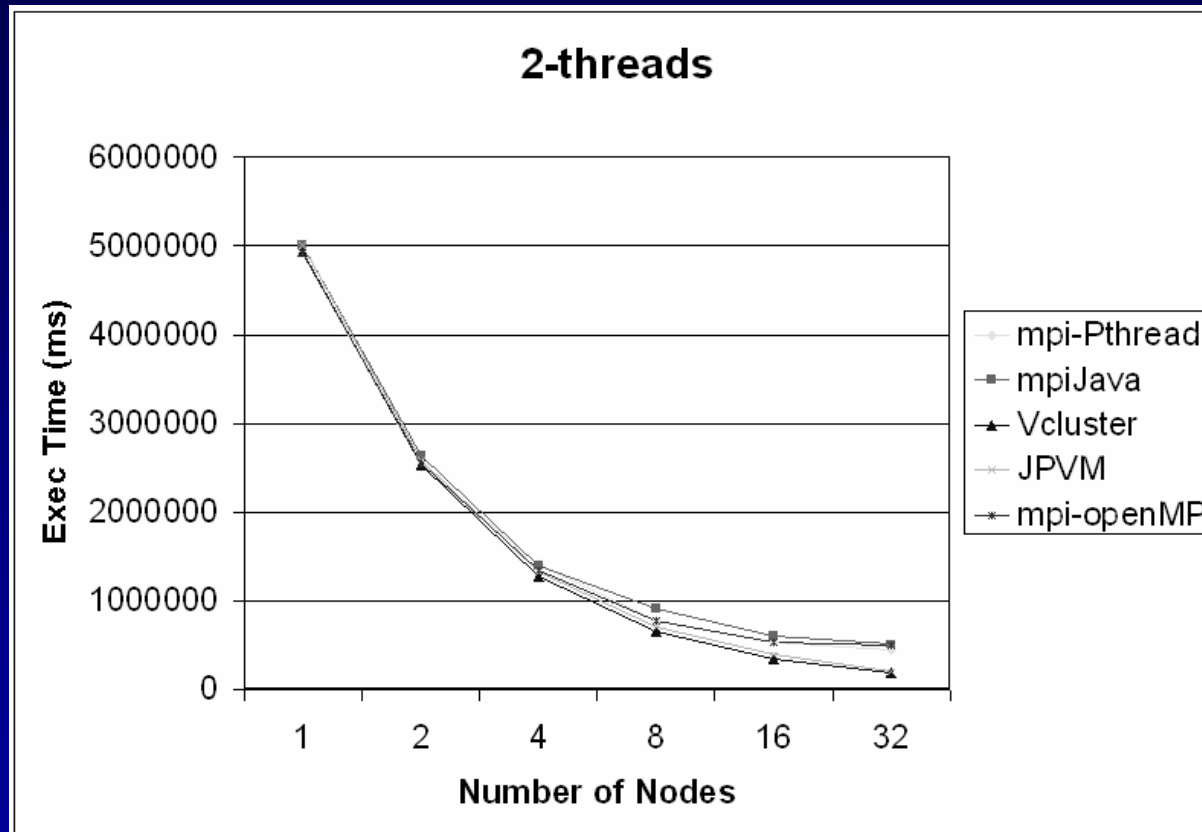


Experimental Results

- Cluster of Multiprocessors (Ariel Cluster)
 - VCluster
 - Slightly changes to the program
 - MPICH
 - Pthread + MPI
 - OpenMP + MPI
 - Need to handle thread communication and synchronization
 - mpiJava and JPVM
 - Use Java thread
 - Need to handle thread communication and synchronization



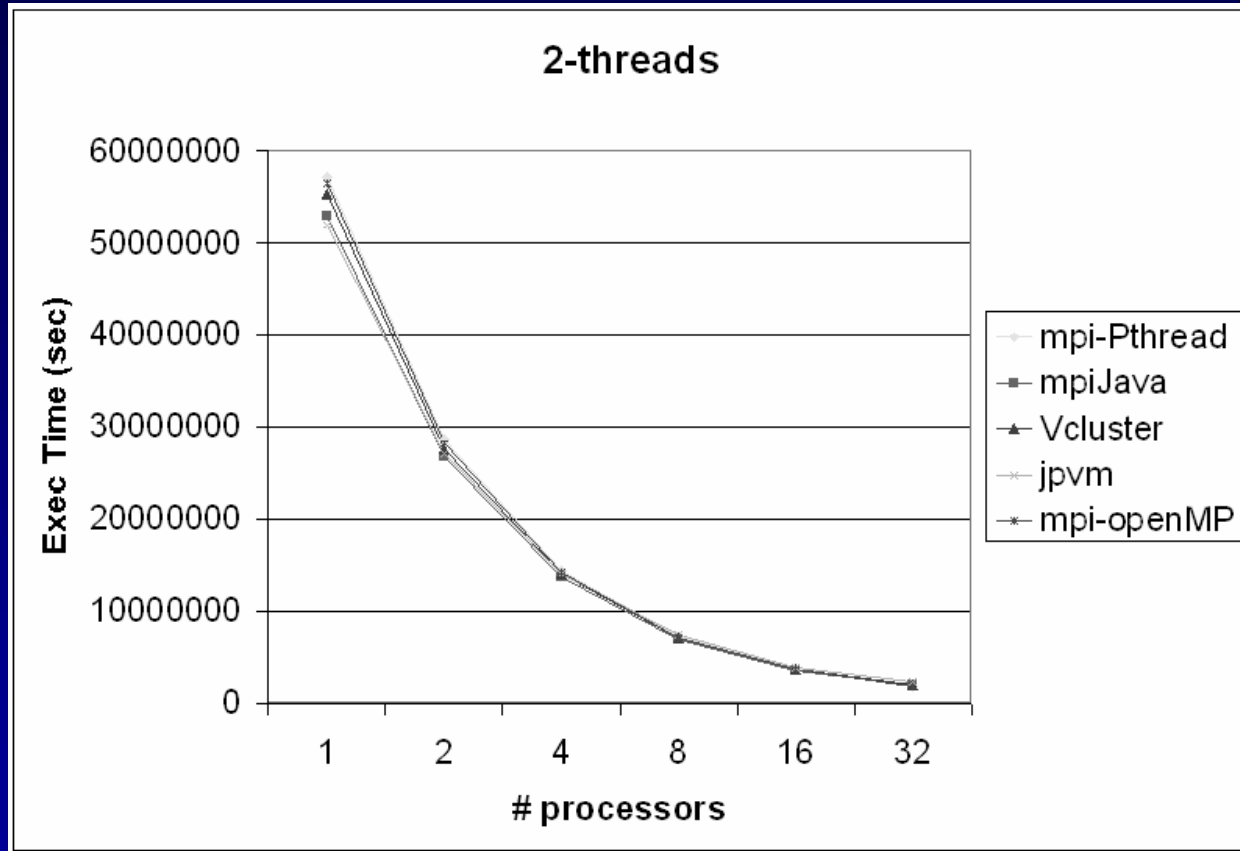
Parallel Dirichlet Problem



Note that VCluster shows the best performance



Parallel BPNN



VCluster is slightly slower than MPI+pthread and MPI+OpenMP



Conclusion

- Performance of VCluster is comparable to the relevant Java based message passing libraries
- VCluster provided a more convenient and efficient programming model for application development due to its unique architecture that combines multithreading with communication
- When enough number of processors are available VCluster can have a performance close to C libraries



Future work

- Thread Migration
 - Virtual state
- Dynamic Load Balancing
 - Thread migration
- Security and Fault Tolerance
 - Java cryptography and security packages
 - Thread migration
- Intra-cluster computing

References

- Joochan Lee, Hua Zhang, Ratan Guha, “Portable and Scalable Parallel Applications with VCluster”, *19th European Simulation Multiconference, High Performance Computing & Simulation (HPC&S) Conference, Riga, Latvia, 2005*
- Joochan Lee, Hua Zhang, Ratan Guha, “Virtual Cluster Computing Architecture”, *The International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), Las Vegas 2005*