

**HETEROGENEOUS PROCESS MIGRATION**  
**for**  
**SCALABLE NETWORK OF WORKSTATIONS (SNOW)**

Xian-He Sun  
Department of Computer Science  
Louisiana State University  
sun@bit.csc.lsu.edu  
URL: <http://www.csc.lsu.edu/scs/SNOW/>

## OUTLINE

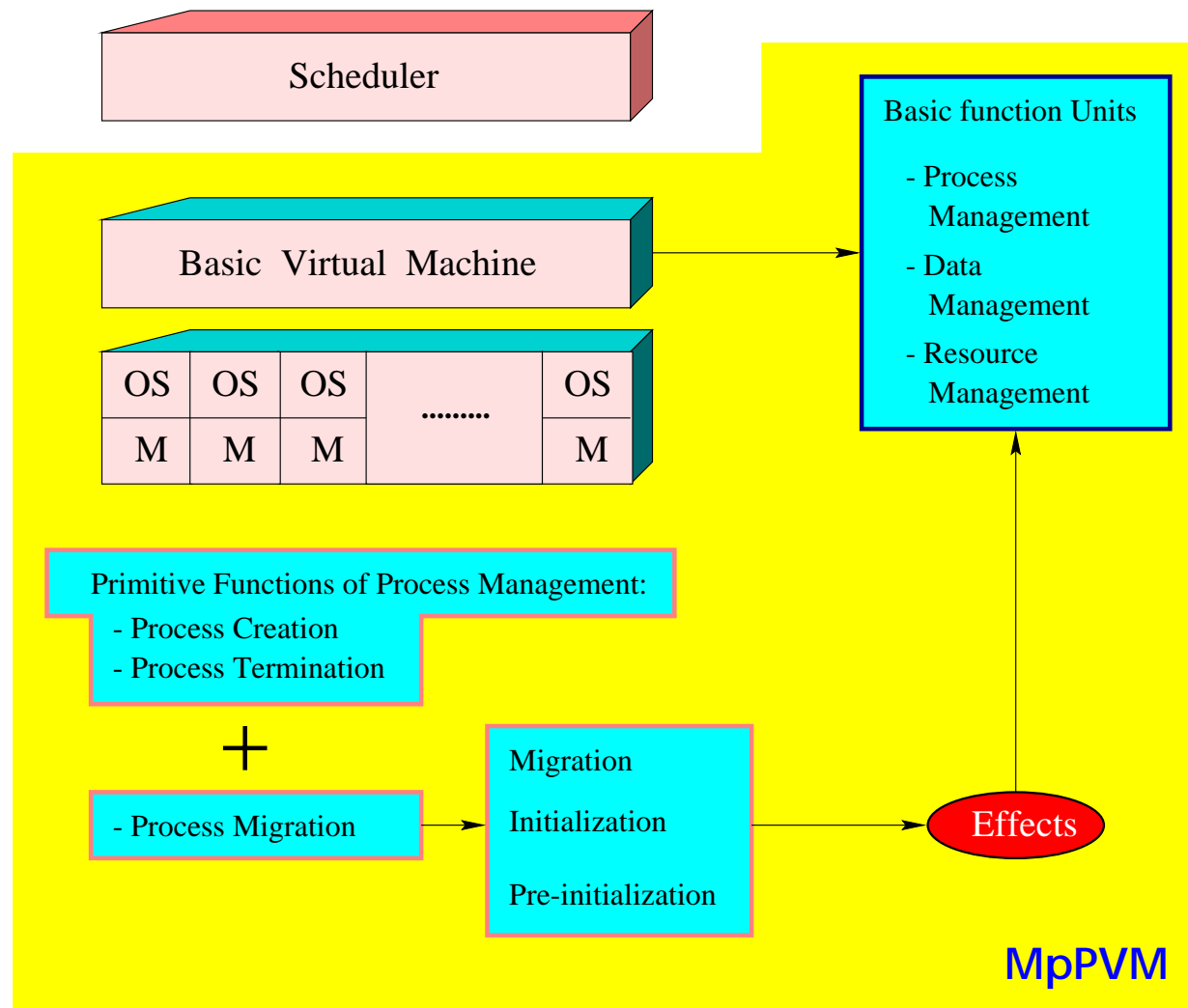
- Overview
- Heterogeneous Process Migration
  - The Design Methodology
  - Comparison of Related Works
- Implementation Mechanisms
  - Data transfer mechanism
  - Data collection and restoration mechanism
- The *SNOW/MpPVM* Software Environment
  - Components and Implementation Issues
  - Experimental results
- Conclusion

**Contributor:** Kasidit Chanchio

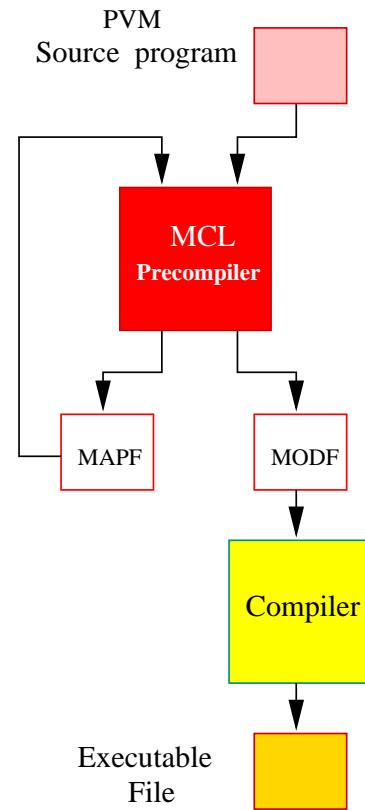
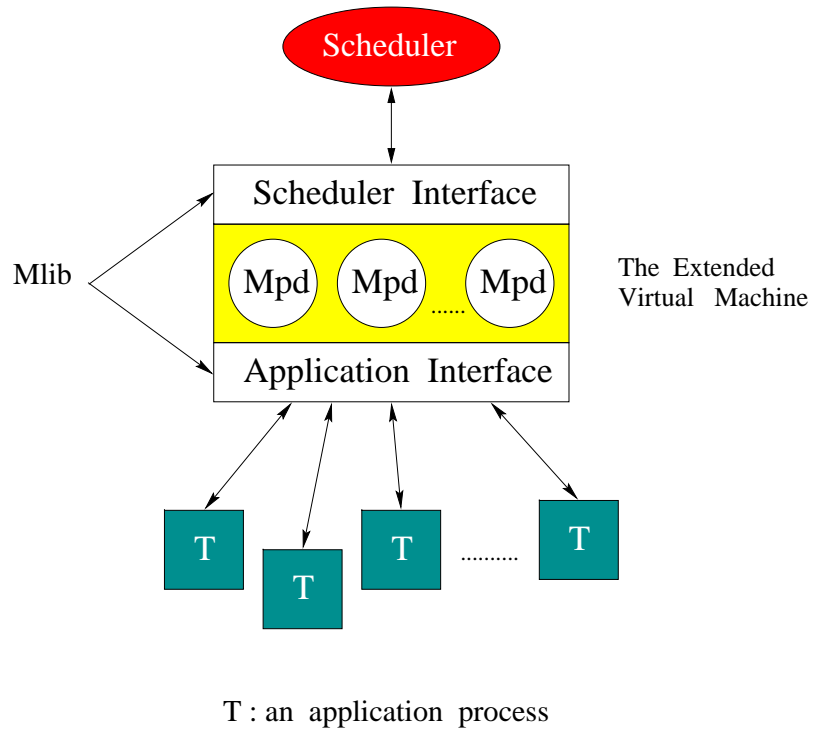
## Distributed Computing Software

- *CONDOR* (Wisconsin)
  - Provides a uniform view of processor resources
  - Homogeneous migration for fault tolerance
- *PVM/MPI* (Oak Ridge/Argonne)
  - Provide a machine-independent communication layer
  - MPVM/MIST (OGI) homogeneous process migration
- *LEGION* (UVA)
  - Builds system components on a distributed object-oriented model
  - High level design
- *GLOBUS* (Argonne)
  - Builds basic low-level mechanisms
  - Does not include process migration

# SNOW Virtual Machine Environment



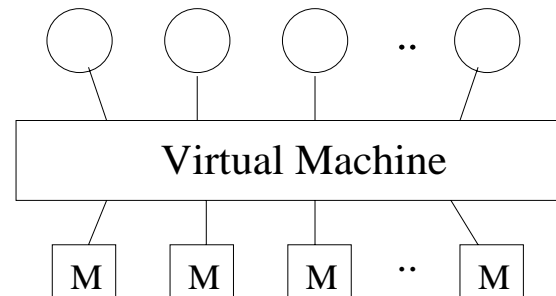
# Components of SNOW



# HIGH-LEVEL PROCESS MIGRATION METHODOLOGY

- Our View of Process Migration (PM)

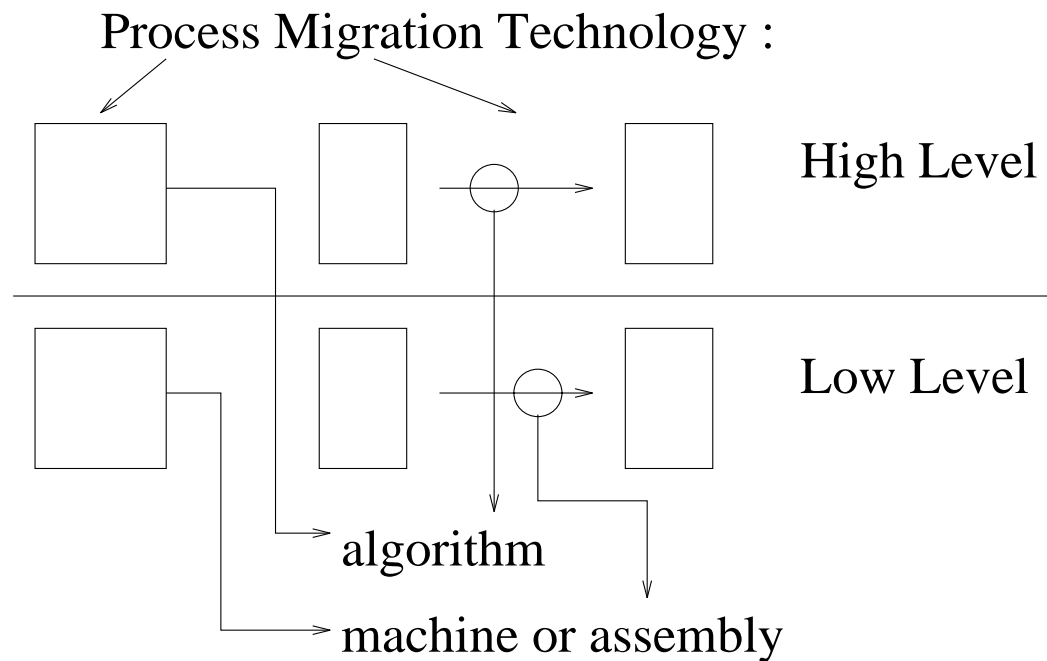
Definition of Process Migration (PM)



PM is the change of Process-Machine binding (dynamic, transparent).

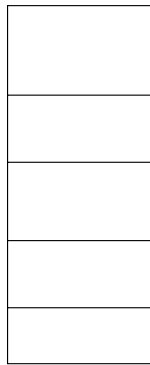
Do most of things at compile-time (automatic).

- Virtual Machine Design Technology
  - Process migration technology
  - Migration point and compiler technology
  - Scheduling technology



# Migration Point and Compiler Technology

## Migration Point : What is it?

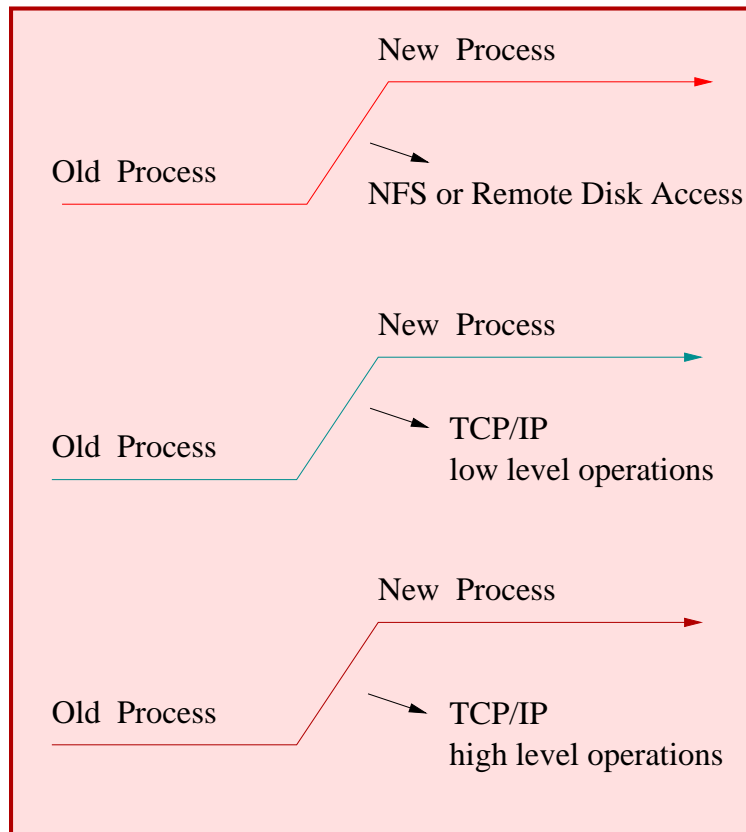


A set of points or locations in program that divide the program into smaller pieces of code. A process migration can occur only at a migration point

Finite possibility of migration  
Yield more control for users and compiler

Introduction of a new "pre-initialization" concept to reduce migration overhead, through the extension of scheduler interface.

## Comparison of Major Process Migration Methods



### Condor's Model:

Homogeneous  
NFS or TCP/IP and Remote Disk  
All data (Partial in some version)  
Checkpoint Periodically

### OGI's Model:

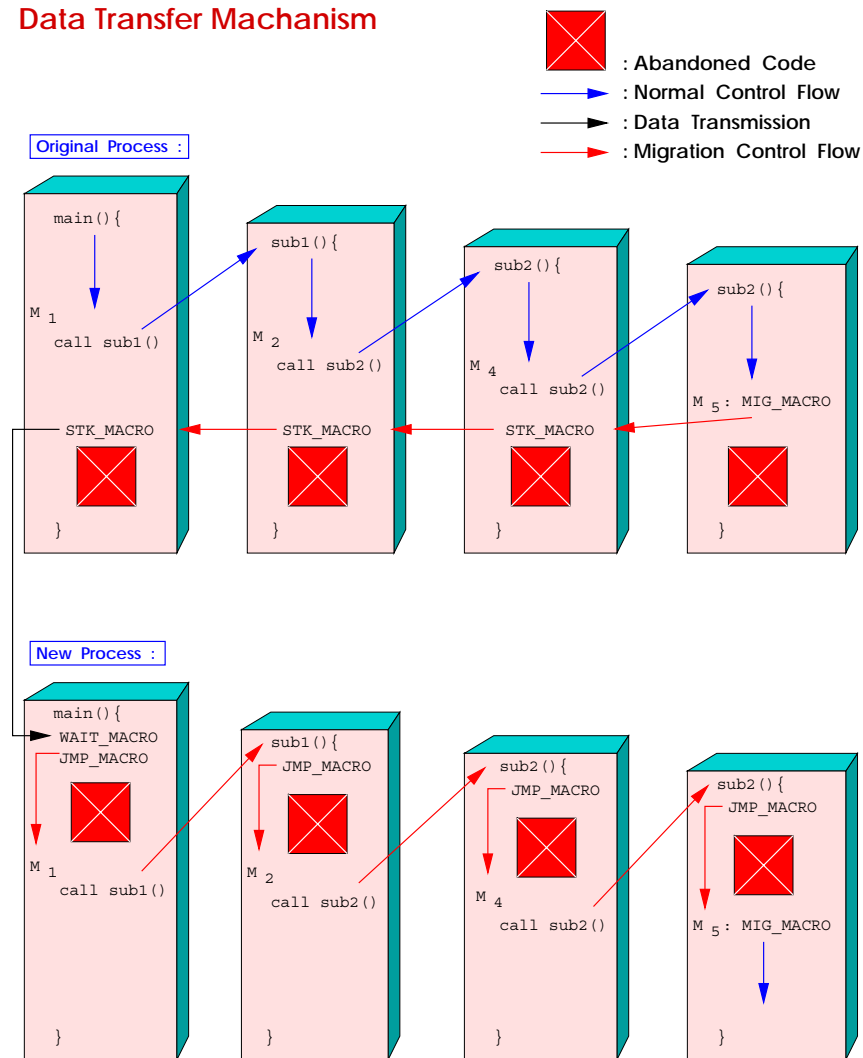
Homogeneous  
TCP/IP  
All data  
Concurrent Initialization  
Anywhere / anytime

### Our Model:

Heterogeneous  
TCP/IP  
Partial to full data  
Preinitialization  
Migration Point

# IMPLEMENTATION MECHANISM

## Data Transfer Mechanism



## **Data Collection and Restoration**

- Memory Space Representation
  - The Memory Space Representation (MSR) graph
  - The Type Information (TI) table
  - Memory block saving and restoring functions
  - Data type of memory blocks
  - Representation of Pointer
- Efficient Data Structure
  - Significant and trivial block nodes
  - MSR Lookup Table (MSRLT)
- Data Collection and Restoration Algorithms

- Timing Results of Heterogeneous Process Migration in Seconds.

<i>Network</i>						
Program	test_pointer		Linpack		bitonic	
Tx Size	1,165,680	3,242,480	325,232	8,021,232	46,704	182,248
Scan	2.678	14.296	0.303	5.591	0.150	0.419
Tx	1.200	4.296	0.357	9.815	0.053	0.191
Restore	2.271	4.563	0.095	2.962	0.077	0.278
Migrate	6.150	23.181	0.756	18.368	0.280	0.889
<i>File</i>						
Scan&Write	18.533	69.032	0.997	45.243	0.803	4.896
Rcp	15.4	20.3	11.9	39.1	10.6	11.5
Read&Restore	8.693	17.602	0.124	3.654	0.303	1.234
Migrate	42.626	106.934	13.220	87.998	11.707	17.631
<i>Comparison</i>						
Diff	36.48	83.75	12.46	69.63	11.43	16.74
Speed Up	6.93	4.61	17.48	4.79	41.72	19.82

## THE *SNOW*/*MpPVM* SOFTWARE ENVIRONMENT

- *MpPVM* (Migration-point based *PVM*):

Support process migration for *PVM* application programs in a heterogeneous environment

- The Main Components of *MpPVM*

- *MCL* (precompiler):

Insert migration points, associated macros and variables into *PVM* source code

- *Mpd* (daemon):

Support reliable point-to-point message passing mechanism

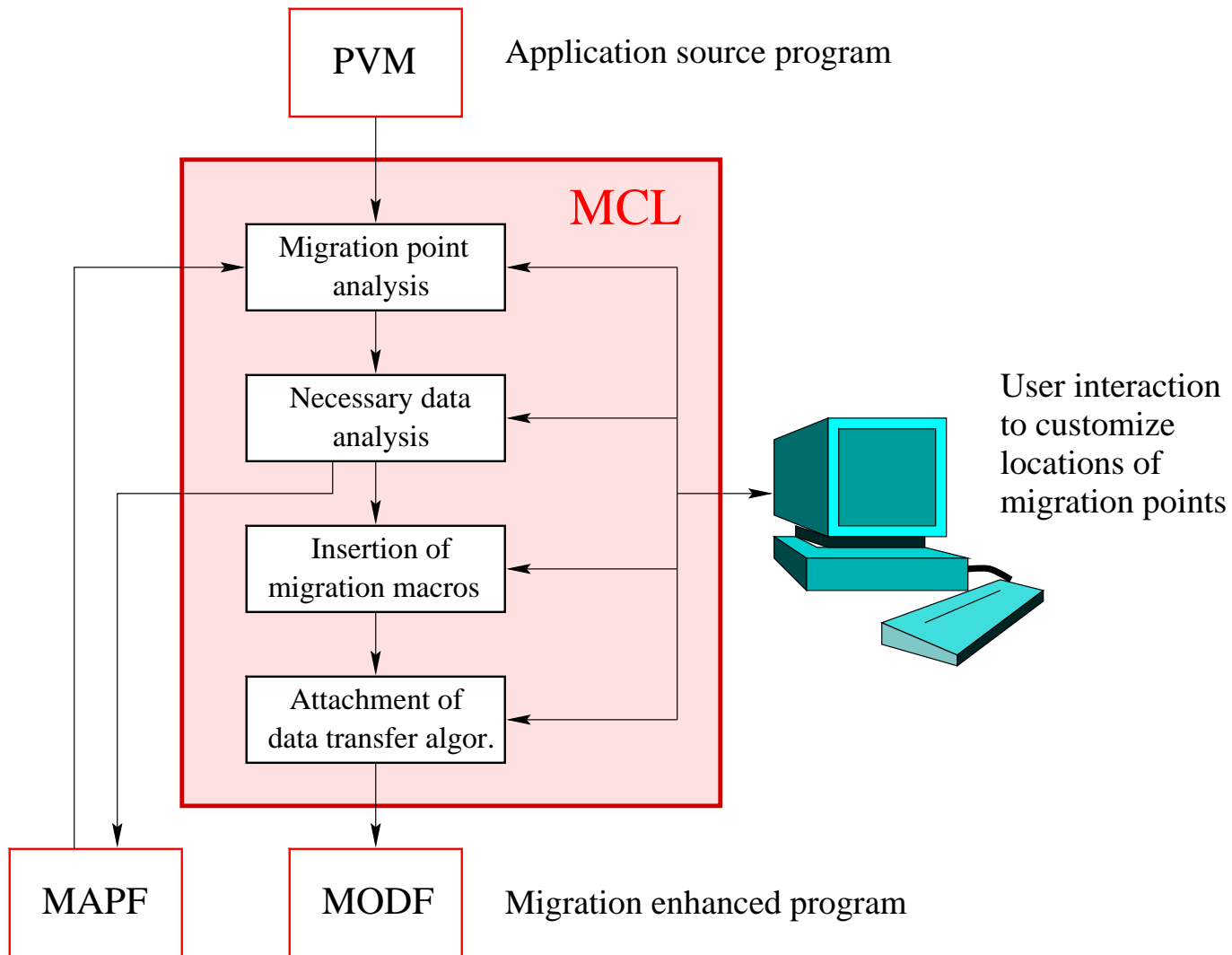
- *Mlibpvm* (library):

Provide interface to the application programs and the scheduler (resource manager)

## The Design of *MCL*

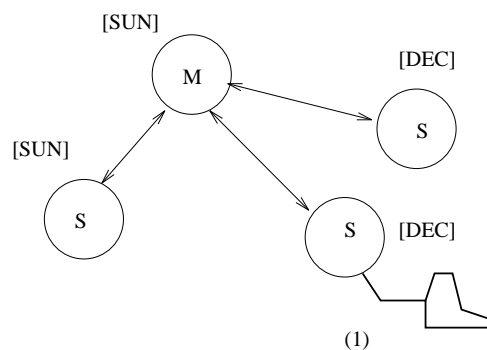
- *Migration Point Analysis*
  - Determine the locations of migration points in the source code
  - User (or the scheduler) specify a maximum cumulative Execution Cost (EC) between any two migration points
- *Data Analysis*
  - Determine the minimum set of variables to be transferred
  - Necessary Variables: Variables that have been initialized before the migration point and will be used after the migration point
  - Data transfer algorithm
- Migration Macros and Variables
  - Necessary for migrating processes and resuming execution
  - Sample global variable: Control Stack, Date Stack, and Execution Flag
  - Sample Macros: WAIT, JMP, STK

# Functionalities of MCL

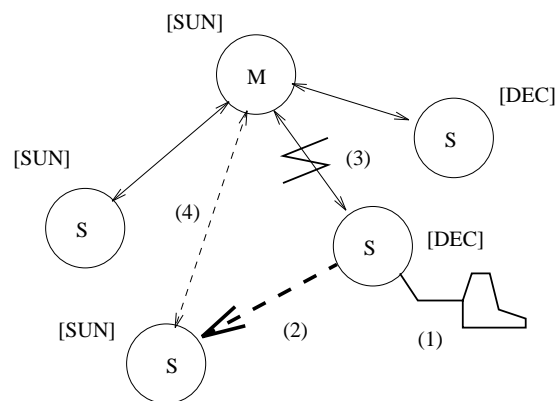


## Implementation Results

- *Mpd* and *Mlibpvm* Have Been Partially Developed
- The *Migration Point* and *Necessary Data Analysis* Have Been Verified
- The Experimental Environment (4 SUN Sparc IPC and 14 DEC 5000/120)



PVM matrix multiplication

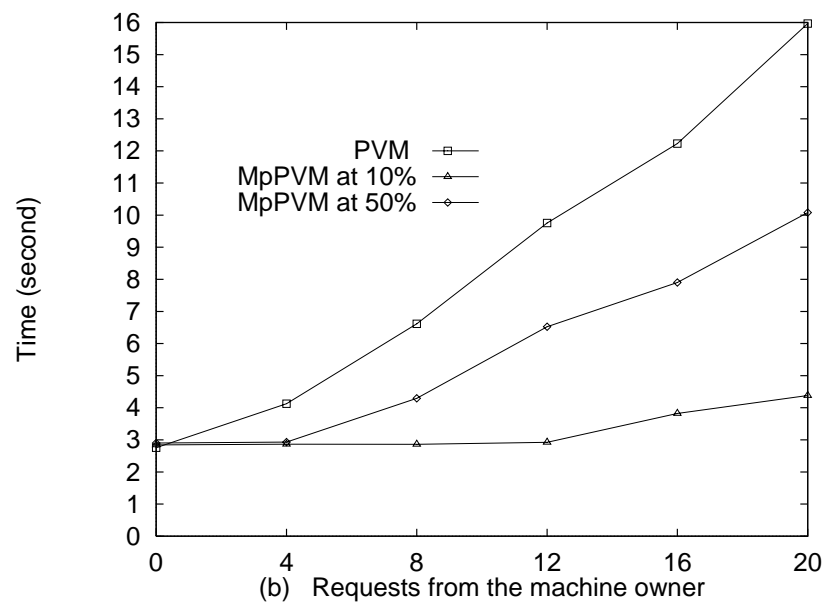
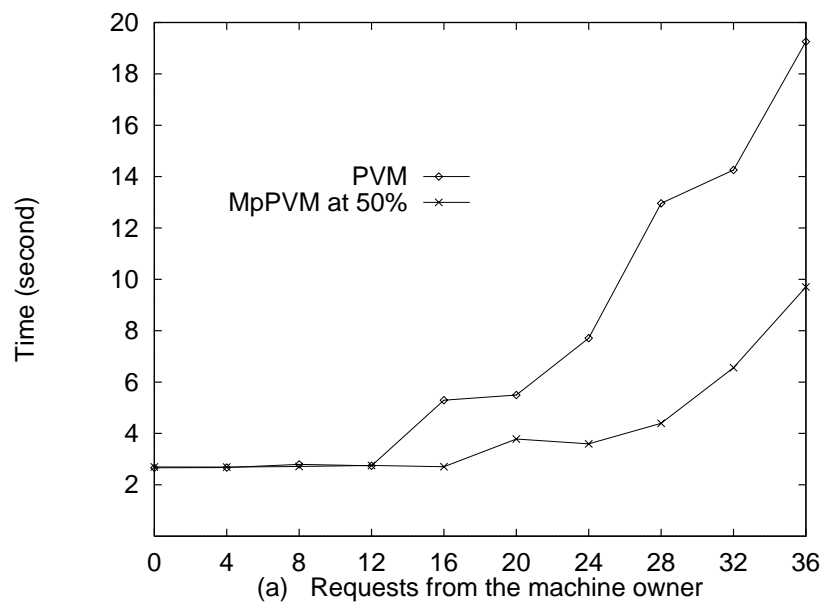


MpPVM matrix multiplication

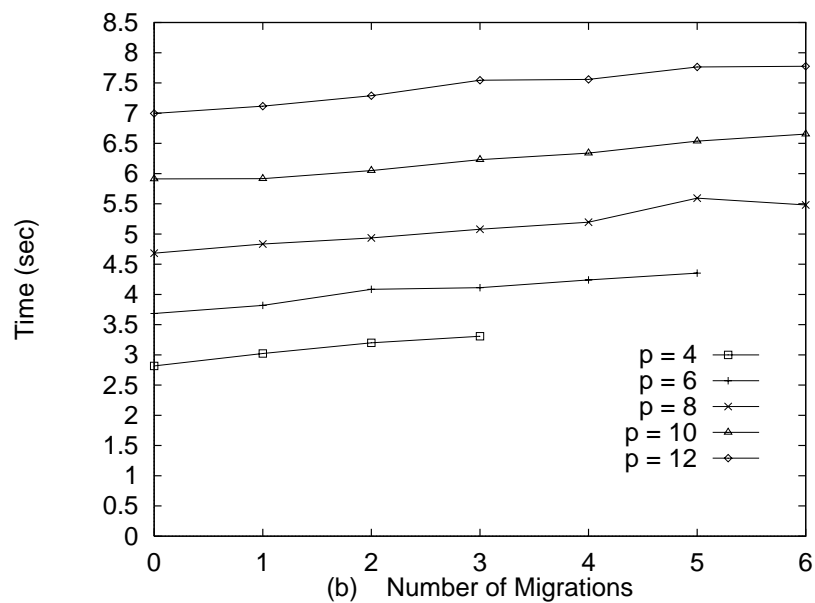
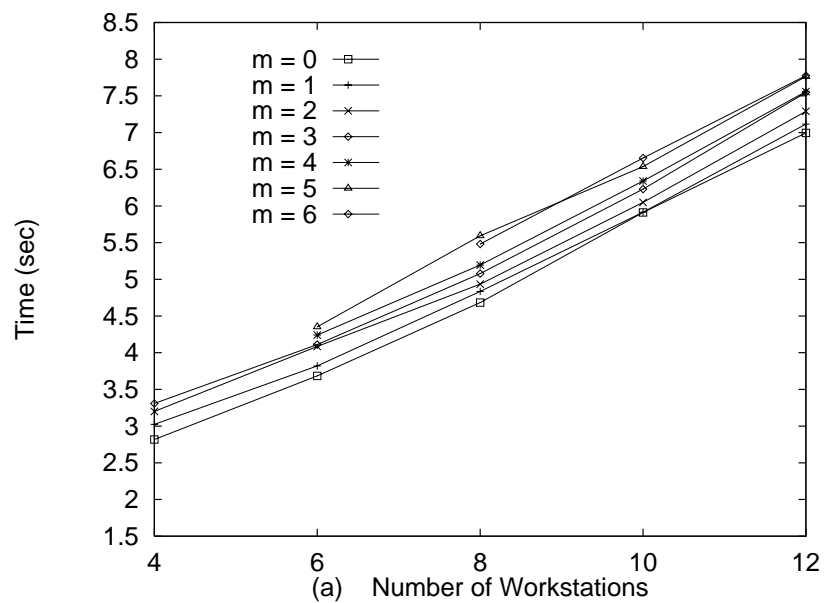
M : Master Process

S : Slave Process

- Execution Time Comparison



● Scalability Study



## CONCLUSION

- A Novel Network Process Migration Methodology
- A Practical Solution to Utilize Idle Machines on the Network and Maintain High Capabilities for Local Computation Simultaneously
- Impact on Non-Dedicated Network Computing and Communication

## **FUTURE RESEARCH:**

- Design Issues
  - Scheduler
  - Compiler Technology
- Implementation Issues
  - Data collection and Restoration
  - Scalable Network of Workstations
- New Distributed Environments
  - Legion
  - Symbio

Please visit <http://www.csc.lsu.edu/scs/SNOW/> for more information regarding the SNOW project