

Laboratory Framework for CS Courses

Hong Lin

Ongard Sirisaengtaksin

University of Houston-Downtown

Goals

- Develop Lab for CS courses
- Construct a cluster
- Develop a lab framework using JAVA

Proposed CS Courses

- CS 4325 Computer Network Architecture
- CS 4328 Parallel Computing
- Grid Computing

CS 4325 Computer Network Architecture

- *LAN configuration*: The use of NICs and hubs
- *Network analysis*: Monitoring a chat room
- *Address resolution*: Experiment with ARP burst
- *IP masquerading*: Clustered web servers
- *WAN configuration*: The use of routers
- *Performance tuning*: Deal with congestion
- *Service configuration*: The configuration of a networked file system:

CS 4328 *Parallel Computing*

- *Topology*: Circulating messages in a ring
- *Collective communications*: Matrix transpose
- *Group management*: Matrix multiplication with Fox's algorithm
- *Scientific computation*: Solving linear systems with Jacobi's algorithm
- *Combinatorial search*: Traveling salesman problem
- *Parallel I/O*: Vector processing - Summation
- *Performance analysis*: Visualization with Upshot – Trapezoidal rule problem
- *Parallel library*: Solving linear system with ScaLapack
- *Scalability analysis*: Bitonic sorting

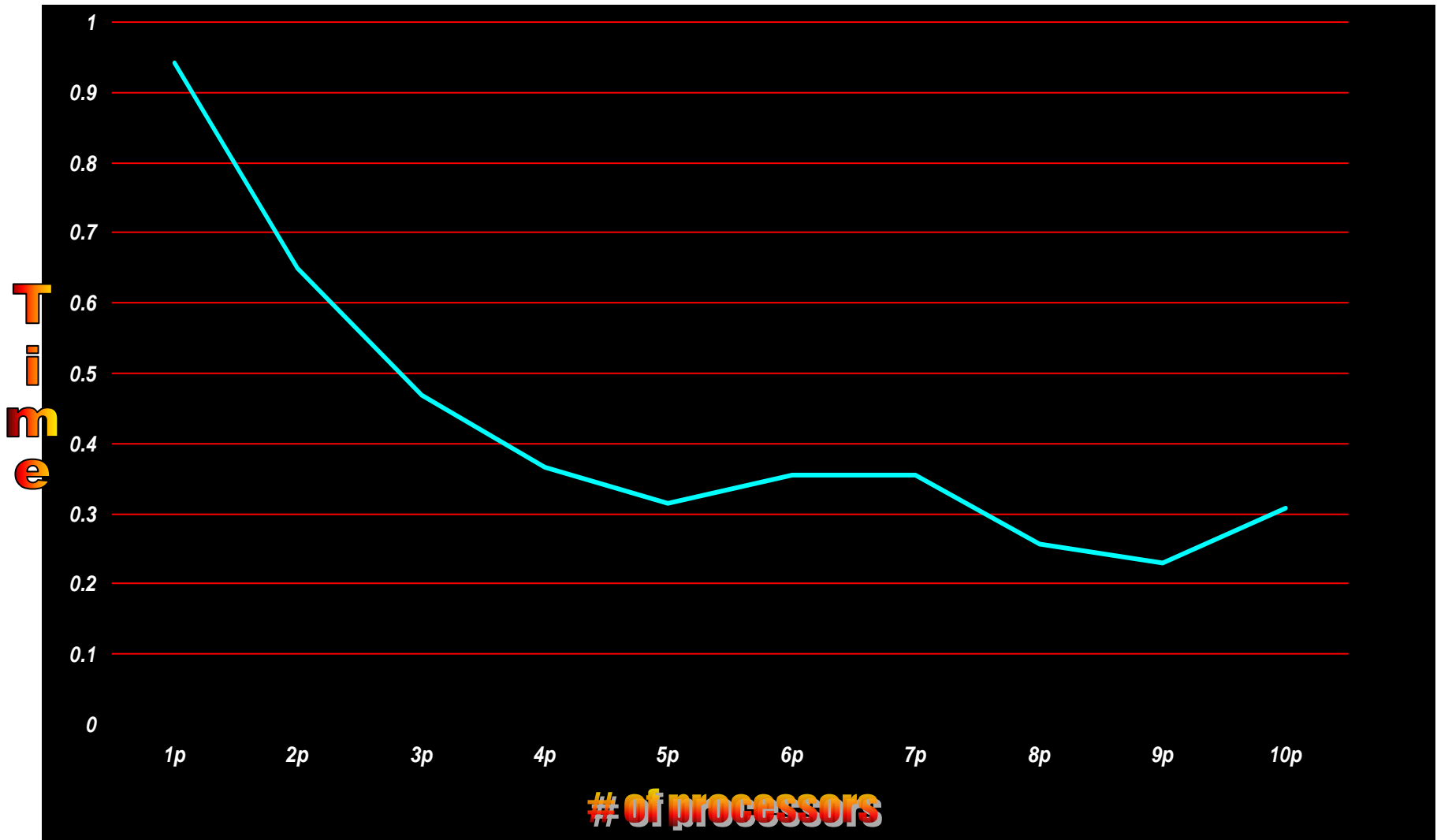
New Course -- Grid Computing

- *Remote procedure call*: Understand the distributed system
- *Client/server architecture*: Configurations of web services
- *Grid programming*: Design a grid-based service using Globus Toolkit 3
- *Information sharing*: An automatic document updating system
- *Web mining*: Coordinated web search in a grid
- *Multi-tier architecture*: Adding a database server into the system
- *Agent*: image filtering
- *Access control*: Customize the authorization policy
- *Data security*: The use of SSL for authentication
- *Parallel computing*: The use of MPI-G2

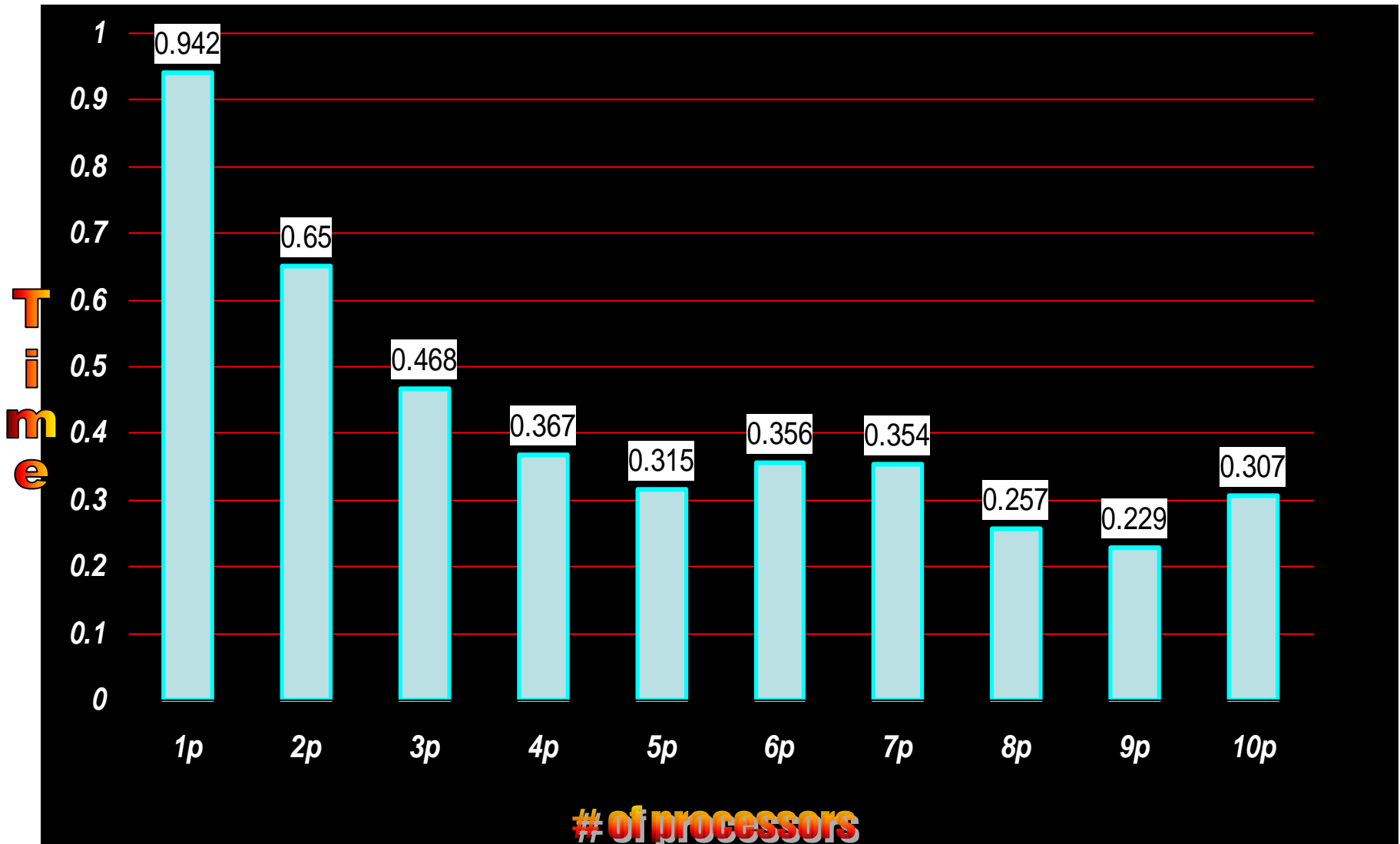
Linux Beowulf Cluster

- 16 Nodes of Pentium III
- RedHat 9.0 as Operating System
- MPICH v 1.2.6 (Unix- all flavors)

Cpilog Test Program



Cpilog Test Program

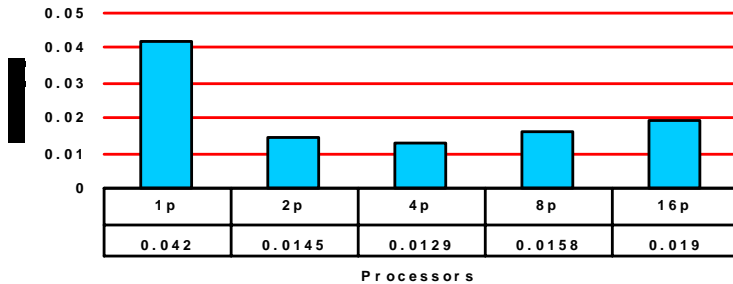


QuickSort

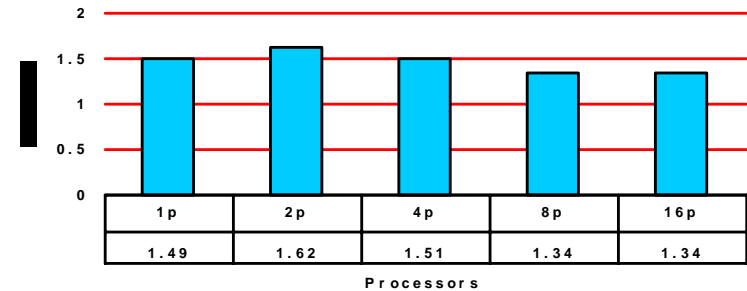
- Time Complexity - $O(n \log_2 n)$
- Massive recursive sort that divides into partitions per pivot element
- Picking a bad pivot will yield $O(n^2)$ (bubblesort)
- Parallel implementation
 - Task out partitions to desired number of processors and recursively sort partitions
- Merge final list on server
- Tested Quicksort on cluster for 2p, 4p, 8p, & 16p.

Test results – Quick sort

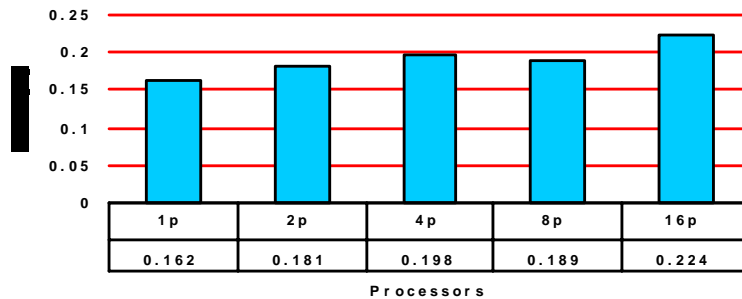
Quicksort for 10,000 elements



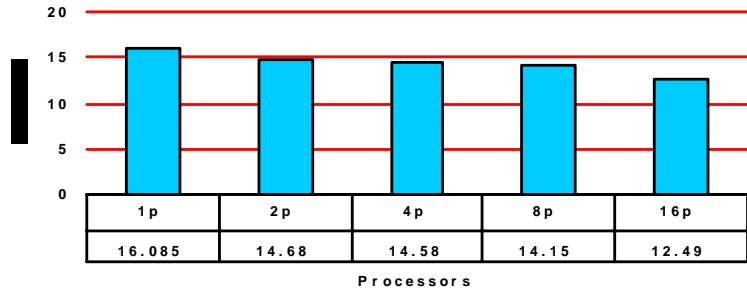
Quicksort for 1,000,000 elements



Quicksort for 100,000 elements

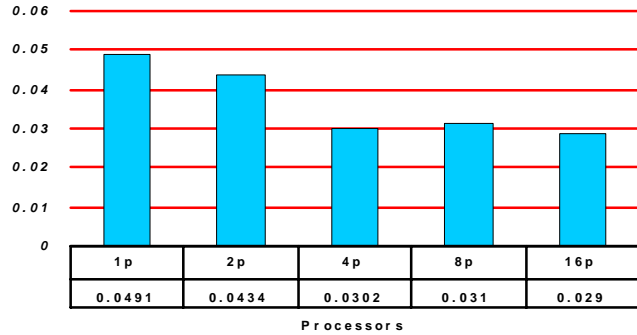


Quicksort for 10,000,000 elements

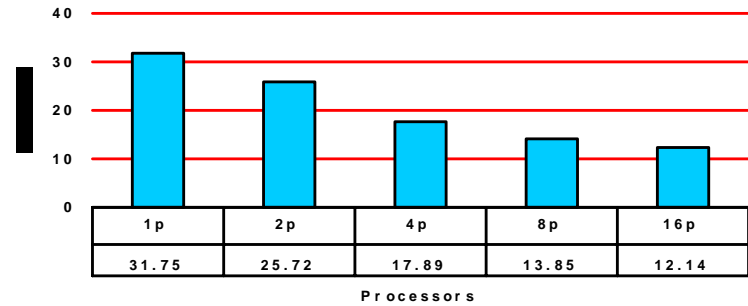


Test results – Merge sort

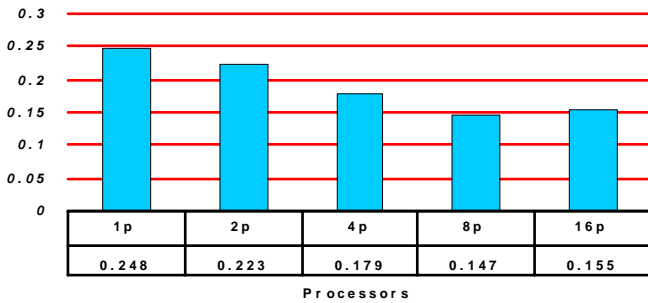
MergeSort - 10,000 elements



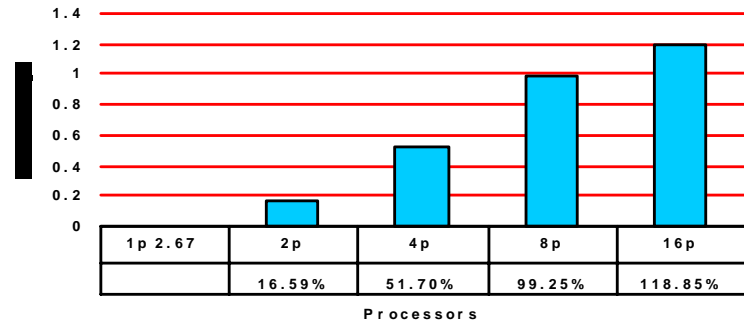
Mergesort - 10,000,000 elements



MergeSort - 100,000 elements



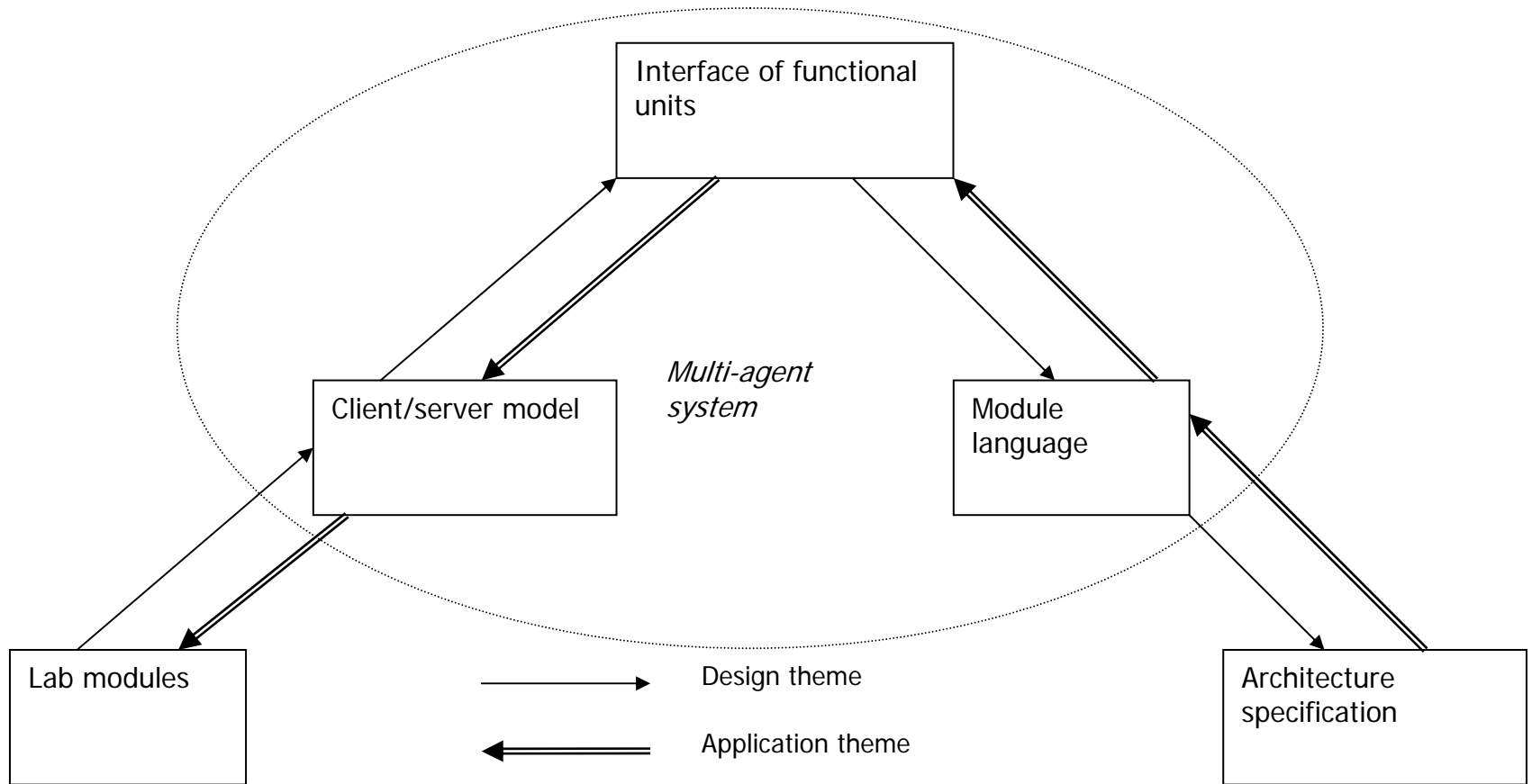
Mergesort Performance Increase for 1,000,000 elements

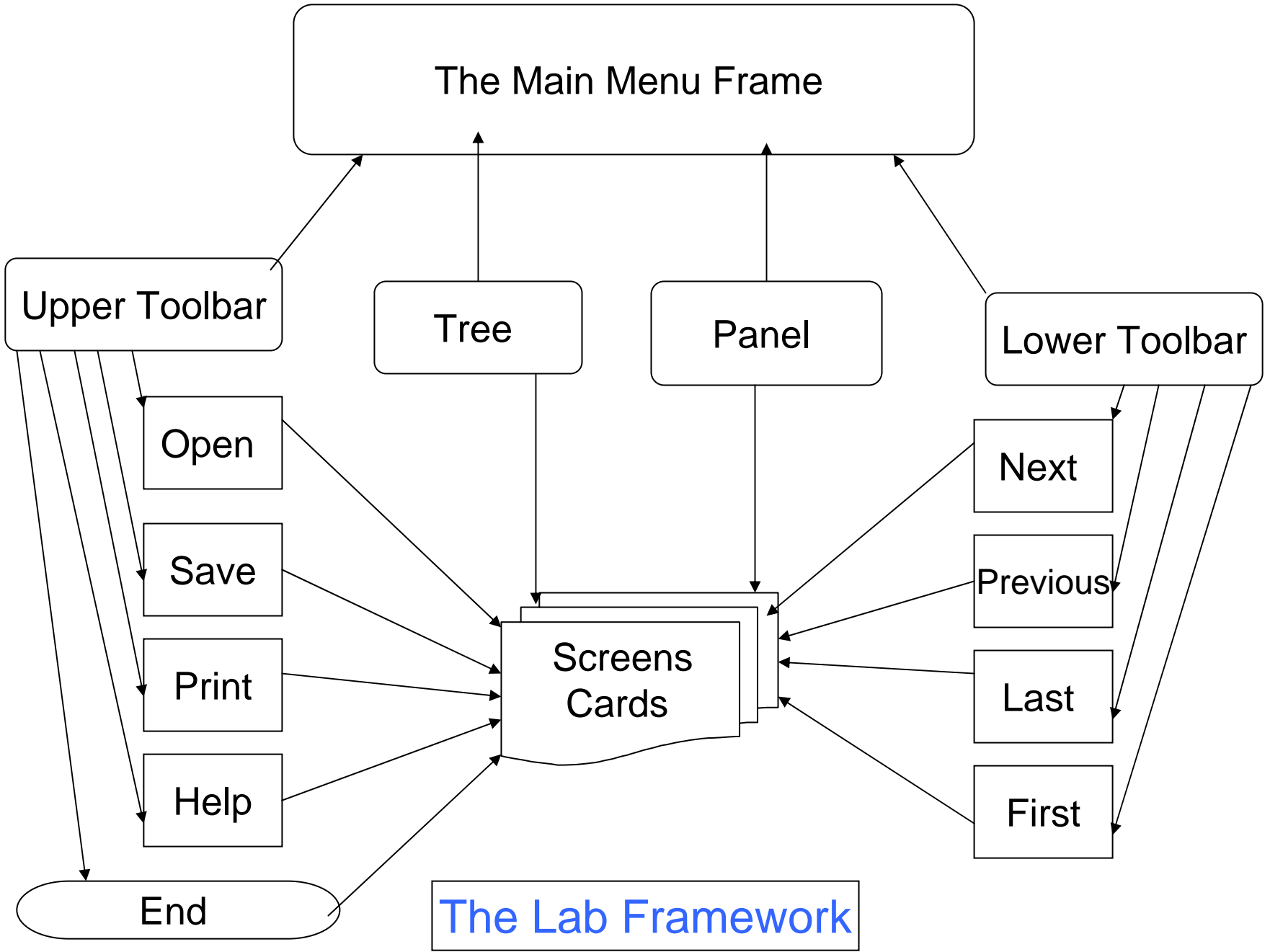


Plan

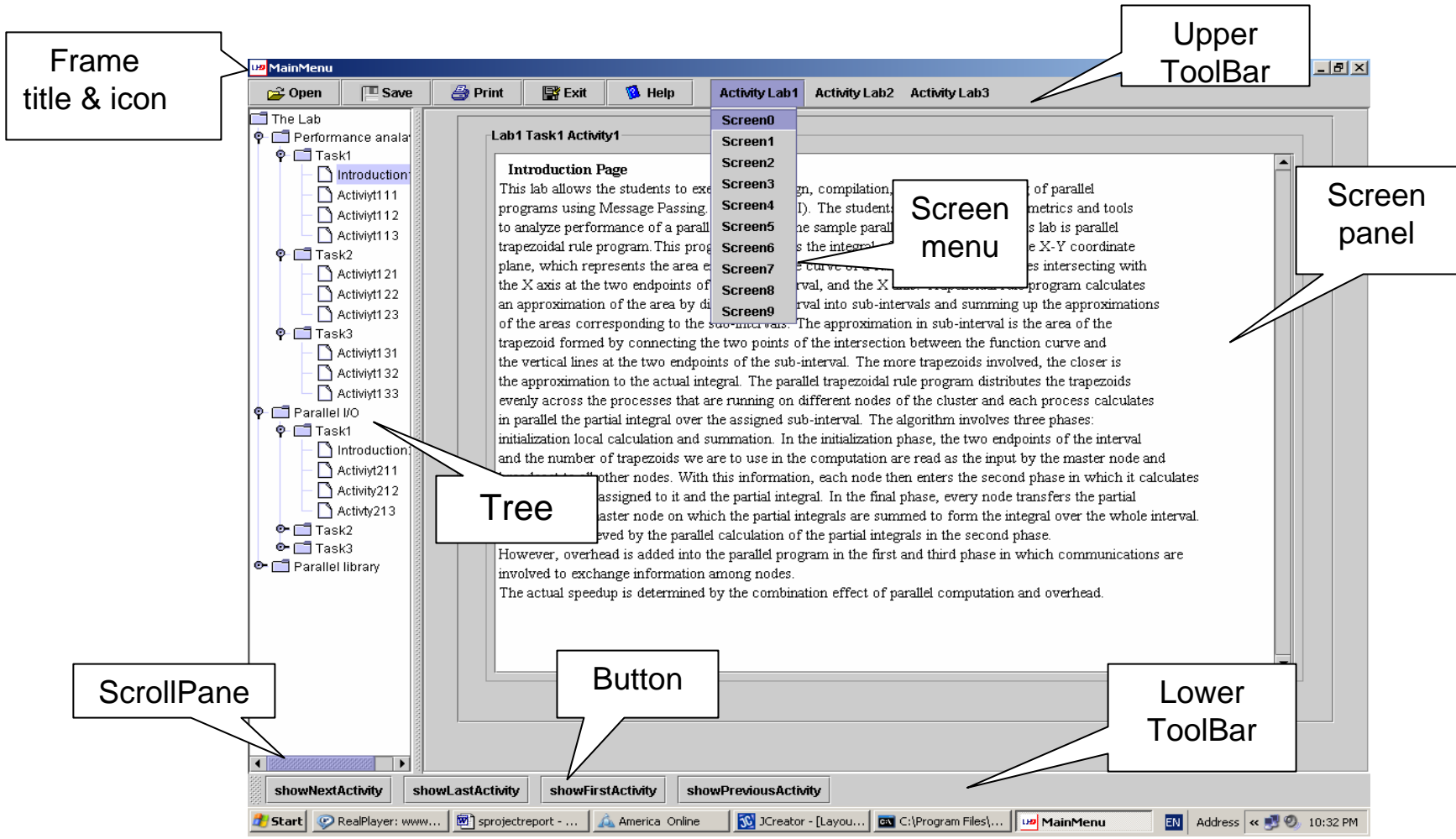
- Grid construction
- Lab design
- Client/server model definition
- Definition of the interface of functional units
- Agent-based architecture construction
- Module language for program refinement
- Architecture specification in the Chemical Reaction
- Use Globus Toolkit 3, Java, and Apache Server

Pyramid-Shaped Model

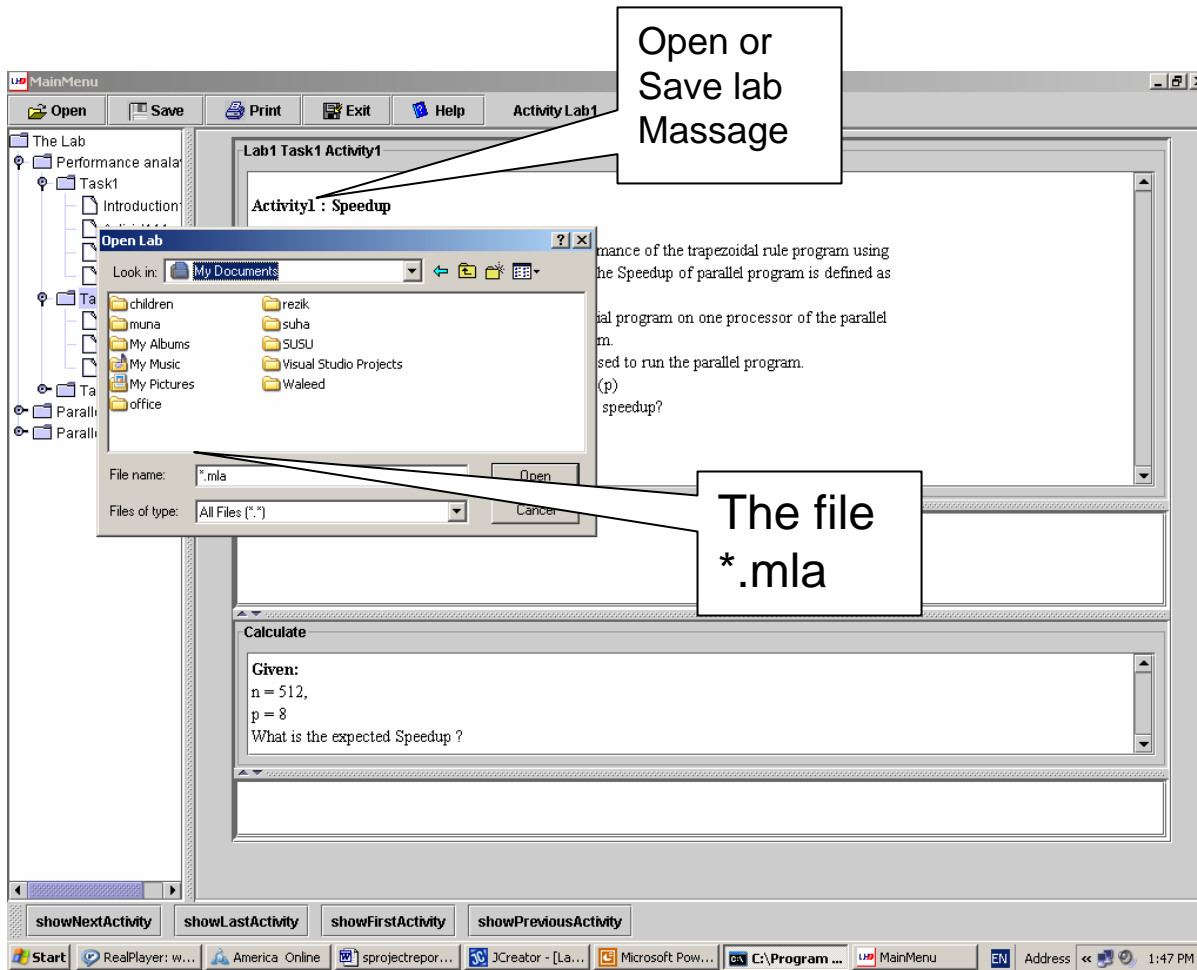




The Main Menu



Open and Close Menus



Print

The screenshot shows a software application window titled 'MainMenu' with a menu bar containing 'Open', 'Save', 'Print', 'Exit', and 'Help'. The main content area is divided into several sections:

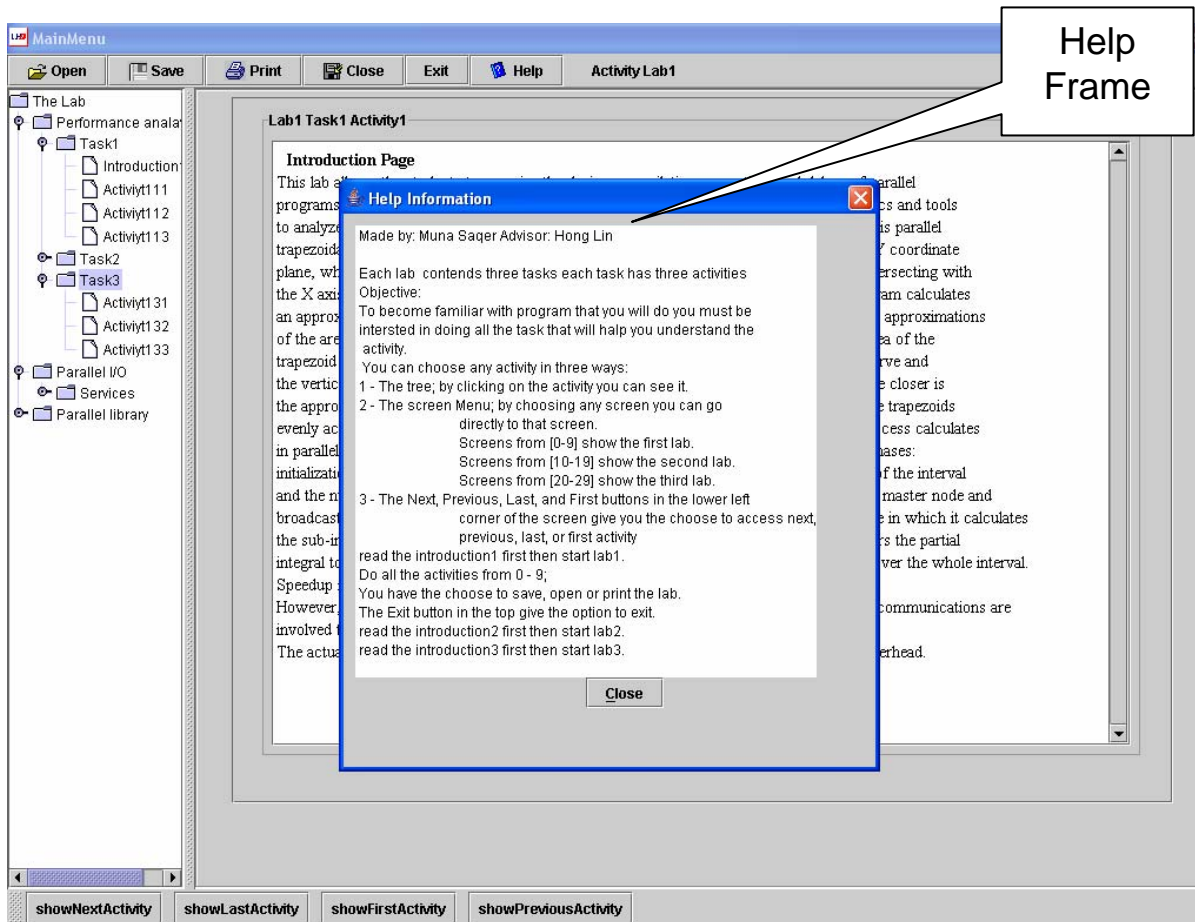
- Left Panel:** A tree view showing a hierarchy of tasks and activities. The 'Task1' folder is expanded, showing 'Introduction', 'Activity111', 'Activity112', and 'Activity113'. Below this, a 'Data Table' is visible, which is a scrollable table with columns for 'Input size' and 'number of nodes'. The table contains the following data:

Input size	number of nodes
n = 32	p = 2
n = 64	p = 4
n = 128	p = 8
n = 256	p = 16
n = 512	p = 32
- Center Panel:** A text area titled 'Activity 3: Performance Prediction' containing instructions: 'After the study of some standard measures and efficiency, for performance analysis which is done in Task Activity 1 and 2, efficiency of the trapezoidal rule program the size of the problem input and the number of nodes. Fill the table below?'
- Right Panel:** A form titled 'Performance analysis' with fields for 'Name:' and 'Date:'. Below these are numbered steps: '1. Using the formula to calculate the speedup', '2. Calculate the expected speedup', '3. Using the formula to calculate the efficiency', '4. Ca...', '5. The Result of the program using 8 nodes', and '6. The actual performance'. At the bottom of this panel is a table with the following data:

Input size	number of nodes	Speedup	
n = 32	p = 2	0.01555254	0.0
n = 64	p = 4	0.0225545	0.0
n = 128	p = 8	0.0	0.0
n = 256	p = 16	0.0	0.0
n = 512	p = 32	0.0	0.0

Callouts point to various elements: 'Print layout frame' points to the right panel; 'Print Dialog' points to a small dialog box titled 'Print Dialog' with a 'Print now?' label and 'Yes' and 'No' buttons; 'Data Table' points to the table in the left panel; 'Scroll Pane' points to the scroll bar of the table in the left panel; and 'Print Table' points to the table in the right panel.

Help Menu



Other Software used

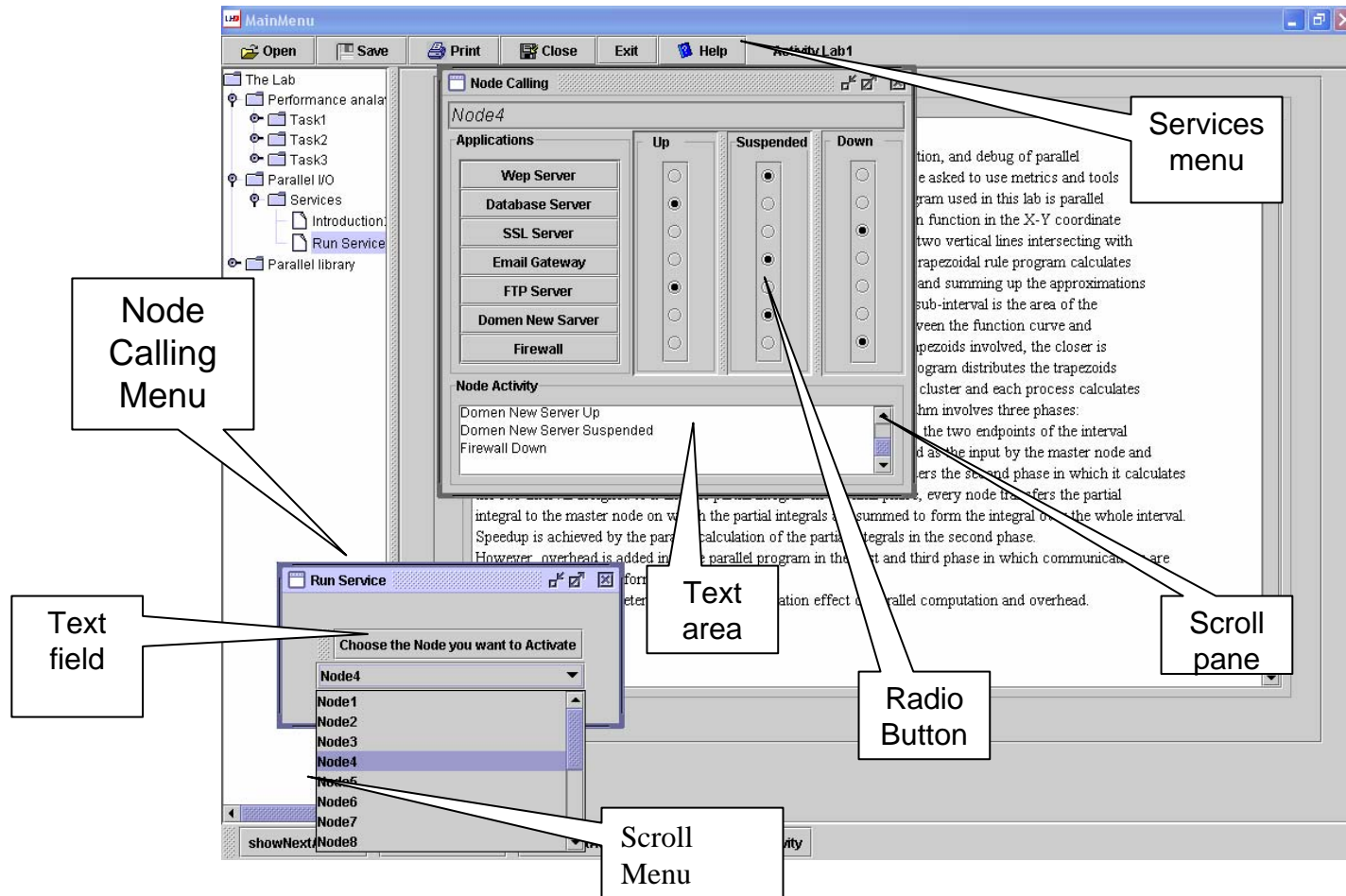
The screenshot displays a software application window titled "MainMenu" with a menu bar containing "Open", "Save", "Print", "Exit", and "Help". Below the menu bar are tabs for "Activity Lab 1", "Activity Lab 2", and "Activity Lab 3". On the left, a tree view shows a hierarchy: "The Lab" > "Performance analysis" > "Task1" > "Introduction", "Activityt111", "Activityt112", "Activityt113"; "Task2" > "Activityt121", "Activityt122", "Activityt123"; and "Task3" > "Activityt131" (selected), "Activityt132", "Activityt133". Below the tree are "Parallel I/O" and "Parallel library" options.

The main content area, titled "Lab1 Task3 Activity1", contains the following text:
Activity 1: Profiling
Insert profiling commands into the trapezoidal rule program and obtain a profile of the program by running it. Firstly, 'mpe.h' header file must be included in your program by using the following "#include" directive:
#include "mpe.h"
Secondly, insert the following codes into your program:
if (my_rank == 0) {
MPE_Describe_state(1, "Start", "gray");
MPE_Describe_state(3, "Derived", "gray");
MPE_Describe_state(5, "Broadcast", "gray");
MPE_Describe_state(7, 8, "End Broadcast", "gray");
}
if (my_rank == 0) {
scanf("%f%f%d", a_ptr, b_ptr, n_ptr);
}
MPE_Log_event(1,0,"Start Derived");
Build_derived_type(a_ptr, b_ptr, n_ptr, &msg_mpi_t);
MPE_Log_event(2,0,"End Derived");
MPE_Log_event(3,0,"Start Broadcast");
MPI_Bcast(a_ptr, 1, MPI_FLOAT, 0, MPI_COMM_WORLD);
MPE_Log_event(4,0,"End Broadcast");
Lastly, profiling should be concluded by using the following function call:
MPE_Finish_log("customprof.log").

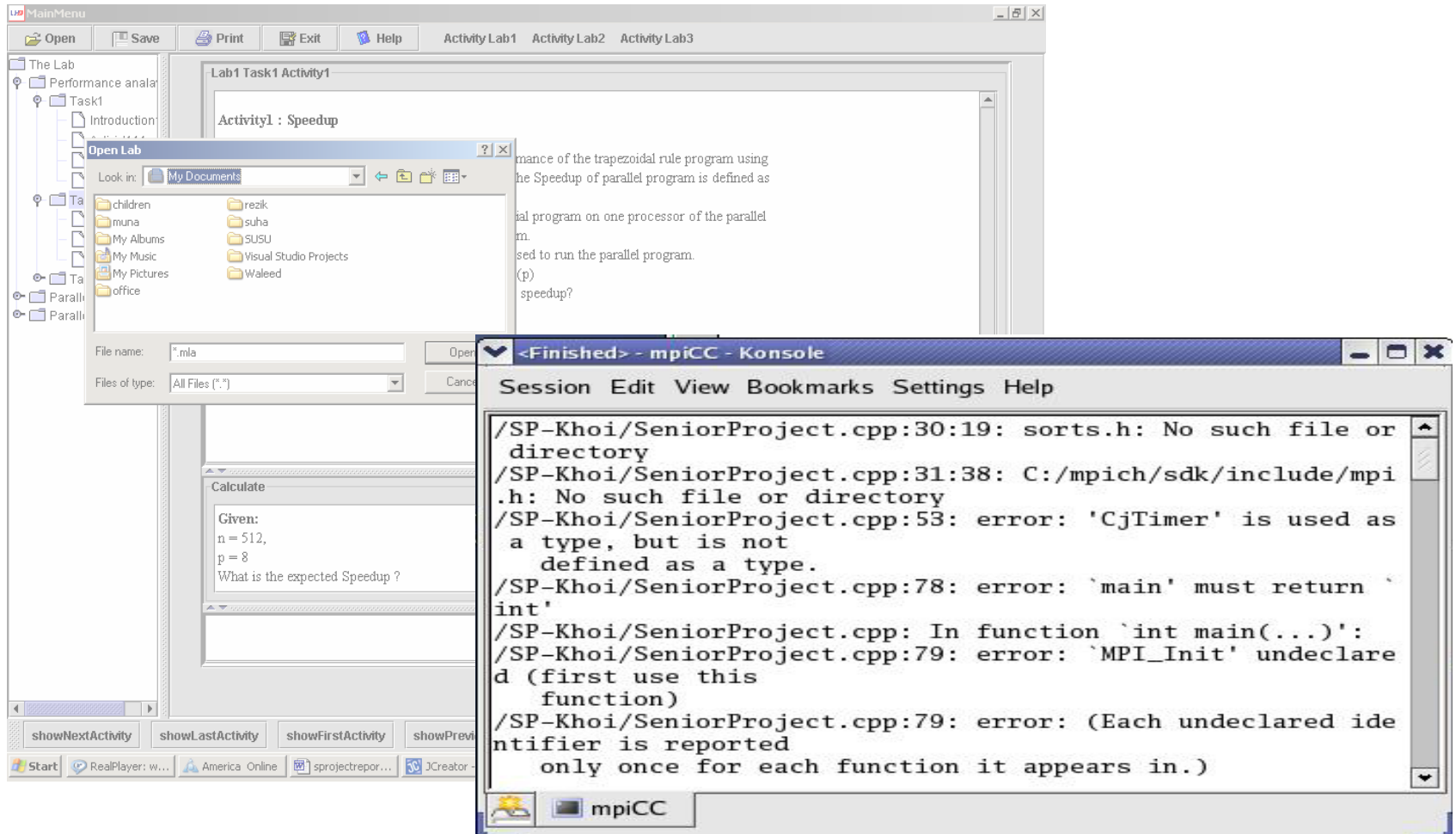
At the bottom of the editor are buttons for "Visual C++", "MPICH", "JumpShot", and "JumpShot Guide". At the very bottom of the window are buttons for "showNextActivity", "showLastActivity", "showFirstActivity", and "showPreviousActivity".

Three callout boxes with arrows point to the buttons: "Visual C++" points to the "Visual C++" button, "MPICH" points to the "MPICH" button, and "JumpShot" points to the "JumpShot" button.

Node Calling and Services Menu



Compilation and execution



Profiling

The screenshot shows a software application window titled "MainMenu" with a menu bar containing "Open", "Save", "Print", "Exit", and "Help". Below the menu bar are tabs for "Activity Lab 1", "Activity Lab 2", and "Activity Lab 3".

On the left side, there is a tree view showing a hierarchy of tasks and activities:

- The Lab
 - Performance anala
 - Task1
 - Introduction
 - Activiyt111
 - Activiyt112
 - Activiyt113
 - Task2
 - Activiyt121
 - Activiyt122
 - Activiyt123
 - Task3
 - Activiyt131
 - Activiyt132
 - Activiyt133
 - Parallel I/O
 - Parallel library

The main area of the window displays the content of "Lab1 Task3 Activity1":

Activity 1: Profiling

Insert profiling commands into the trapezoidal rule program and obtain a profile of the program by running it. Firstly, 'mpe.h' header file must be included in your program by using the following "#include" directive:

```
#include "mpe.h"
```

Secondly, insert the following codes into your program:

```
if (my_rank == 0) {
MPE_Describe_state(1, 2, "Derived", "gray");
MPE_Describe_state(3, 4, "Derived", "gray");
MPE_Describe_state(5, 6, "Derived", "gray");
MPE_Describe_state(7, 8, "Derived", "gray");
}

if (my_rank == 0) {
scanf("%f%f%d", a_ptr, b_ptr, n_ptr);
}

MPE_Log_event(1,0,"start Derived");
Build_derived_type(a_ptr, b_ptr, n_ptr, &mesg_mpi_1);
MPE_Log_event(2,0,"end Derived");
MPE_Log_event(3,0,"start Bcast");
MPI_Bcast(a_ptr, 1, mesg_mpi_1, 0,
MPI_COMM_WORLD);
MPE_Log_event(4,0,"end Bcast");
```

Lastly, profiling should be concluded by using the following function call

```
MPE_Finish_log("customprof.log").
```

At the bottom of the window, there are buttons for "Visual C++", "MPICH", "JumpShot", and "JumpShot Guide". At the very bottom, there are four buttons: "showNextActivity", "showLastActivity", "showFirstActivity", and "showPreviousActivity".