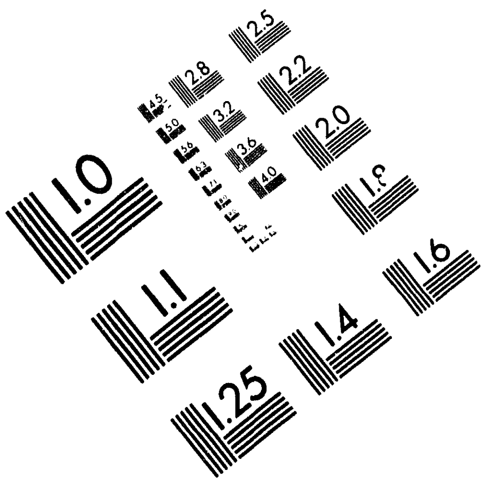
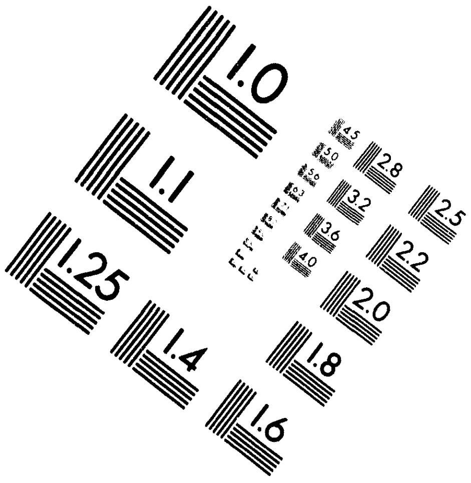


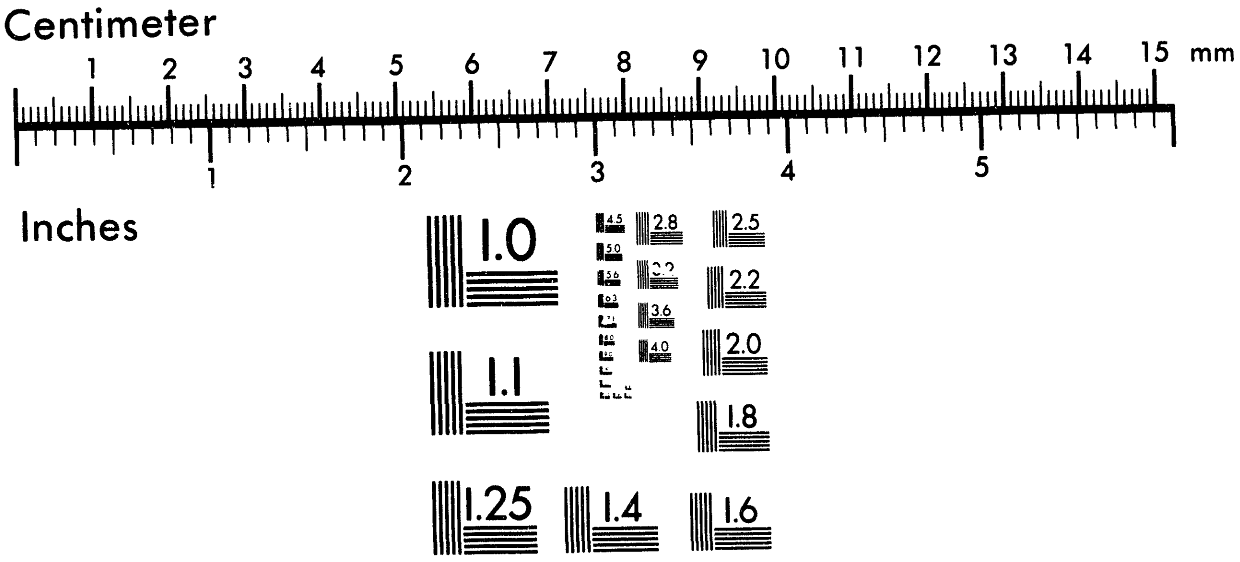


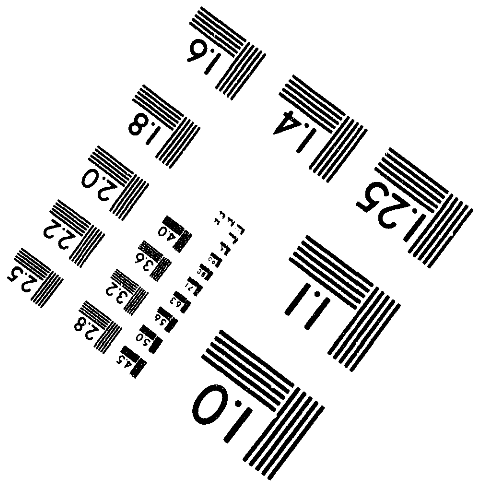
AIIM

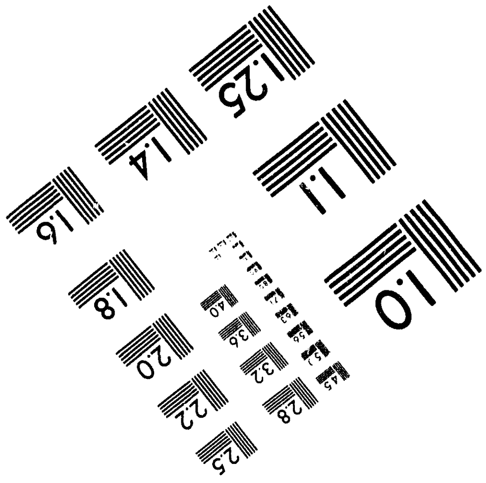
Association for Information and Image Management

1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910

301/587-8202







MANUFACTURED TO AIIM STANDARDS

BY APPLIED IMAGE, INC.

1 of 1

CONF-9405100--12

XPVM: A Graphical Console and Monitor for PVM*

James Arthur Kohl
Oak Ridge National Laboratory
Engineering Physics and Mathematics Division
Mathematical Sciences Section
P.O. Box 2008, Bldg. 6012
Oak Ridge, TN 37831-6367

*Research supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

[illegible]

1. Introduction

As technology continues to spawn new supercomputing platforms and more powerful workstation environments, it becomes increasingly important to integrate and apply these diverse collections of computers together to extract the best performance possible. PVM (Parallel Virtual Machine) is gaining wide popularity as a means for realizing the computing power of these heterogeneous systems. PVM is a software package that permits a heterogeneous collection of serial, parallel and vector computers, hooked together by a network, to appear as one large distributed memory computer.

One of the primary difficulties in designing applications for distributed computer systems is identifying precisely what is happening inside a program, as multiple threads of control interact to solve a problem. Often it is useful to provide the user with feedback as to the state of the system, including the individual tasks executing on it. XPVM is a graphical interface to PVM that provides such information, to monitor executions and assist in debugging and tuning the performance of PVM programs. XPVM can be used in real time, to observe program activity as it happens, or it can play back saved traces of PVM program executions, for detailed post-mortem analysis.

2. XPVM Features

XPVM is written in C with the X Window System, using the TCL / TK application tool construction toolkit. XPVM provides a graphical user interface to all the functions provided in the standard PVM console shell. Hosts can be added to or deleted from the Virtual Machine by selecting their name from a pull-down menu, and tasks can be spawned, killed, or signaled using various menus and dialog prompt boxes. The Virtual Machine can also be reset or halted with the click of a button.

XPVM supports a variety of views to assist in analyzing PVM programs (see Figure 1). Where appropriate, current program visualization techniques have been applied to present information visually, using graphics. Pure text is also used to precisely describe details in other views. There are views to show the state of hosts in the Virtual Machine and the state and interactions among the tasks executing.

2.1. Network View

The Network View displays high-level activity on the hosts in the Virtual Machine. Each host is represented by an icon image which includes the architecture and name of the host. These icons are illuminated in different colors to indicate the status of the tasks running on each host. The "active" color implies that at least one task on that host is busy executing useful work. The "system" color means that no tasks are busy executing user computation, but that at least one task is busy executing PVM system routines. When there are no tasks on a given hosts, its icon is left uncolored, or white. The specific colors used in each case default to green for active and yellow for system, but are user customizable.

This view will be extended to visualize network activity among the hosts as well. The actual network traffic will be visualized, as will the network bandwidth — in terms of the

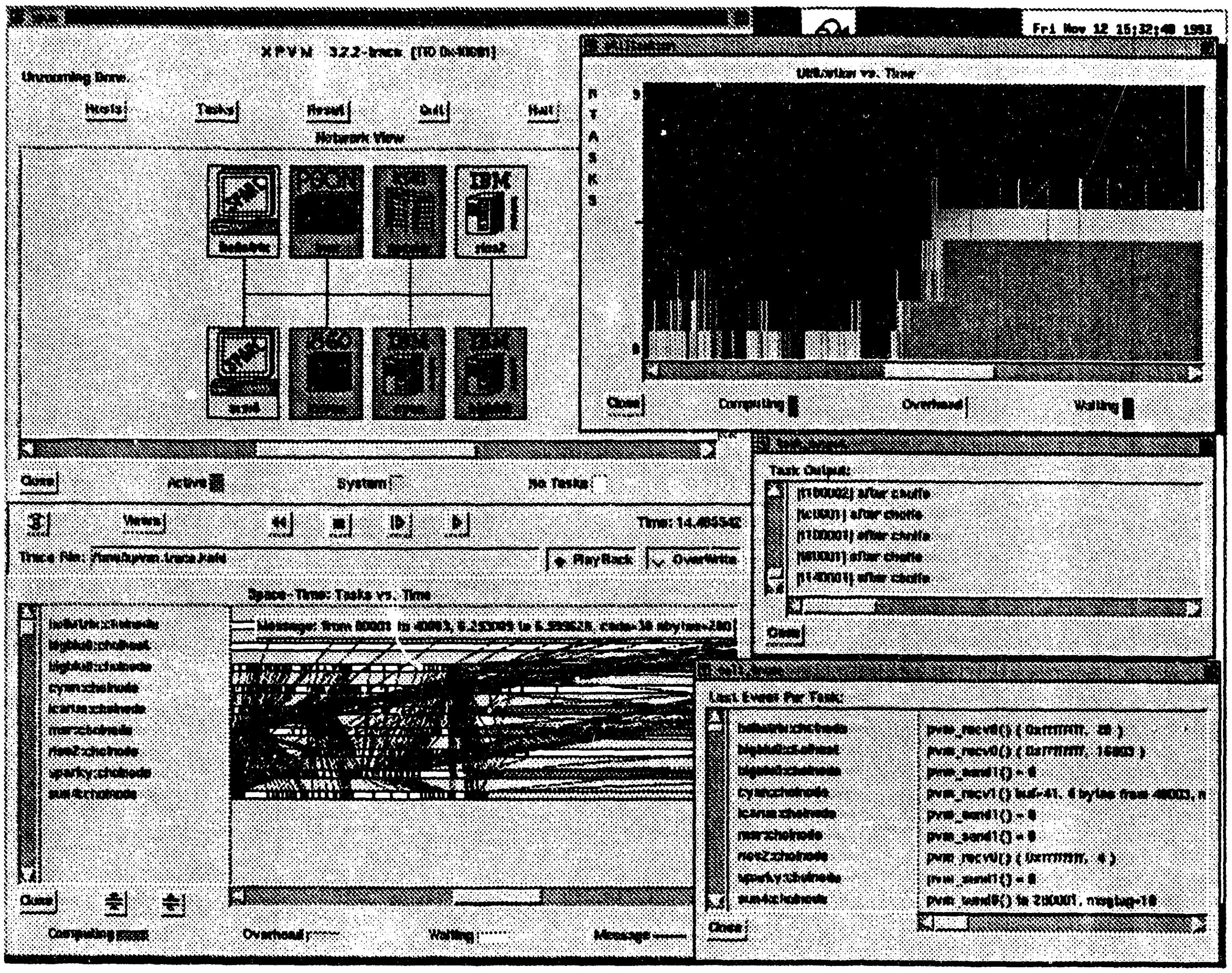


Figure 1: XPVM Interface

number of messages and the number of bytes per second transferred. The arrangement of host icons will be appropriately modified to represent several different network topologies. The current view assumes a bus / ethernet topology, and arranges the hosts as if hanging off of a horizontal bus.

2.2. Space-Time View

The Space-Time View shows the status of individual tasks, as they execute across all hosts. Each task is represented by a horizontal bar along a common time axis, where the color of the bar at each time indicates the state of the task. This view is reminiscent of charts by Gantt [Gan19], and this type of view has been used in a variety of performance evaluation tools [HeE91, HeL91, LeB90]. The "computing" color shows those times when the task is busy executing useful user computations. The "overhead" color marks the places where the task executes PVM system routines for communication, task control, etc. The "waiting" color indicates those time periods spent waiting for messages from other tasks, i.e. the idle time due to synchronization. The default colors for these states are green for computing, yellow for overhead, and white for waiting. Communication activity among tasks is also shown in the Space-Time view, using red lines drawn between the task bars at the corresponding message send and receive times.

More detailed information regarding specific task states or messages can be extracted from the Space-Time view by clicking on the view area. If a task bar is clicked on, a small "pop-up" window appears containing the precise times that the task state began and ended, plus the specific PVM system routine called for overhead states, and the expected message sender and message code for waiting states. Individual messages can be clicked on as well to determine the time when the send was initiated, and the time when the message receipt was completed, plus the number of bytes sent and the message code.

2.3. Utilization View

The Utilization View summarizes the Space-Time View at each time instant, showing the number of tasks computing, in overhead, or waiting for a message at any given time. This information is represented by three colored rectangles, stacked vertically at each time instant, with computing on the bottom, overhead next, and waiting on top. The default colors are green for computing, yellow for overhead, and red for waiting. The overall utilization for tasks is seen by comparing the relative height of the rectangles over time, such that in cases of good utilization the computing areas are prominent, and for poor utilization the overhead and waiting areas dominate.

The Utilization and Space-Time Views share the same horizontal time scale, and entering either view area with the mouse pointer creates a blue time correlation line in both views. This lines identifies the same time instant in each view's display, to help correlate the two representations.

2.4. Call Trace View

The Call Trace View provides a textual record of the instantaneous activity in each task. For each task, a line of text is displayed that represents the last PVM system call made by the task, including any call parameters or results. As the tasks execute, the lines changed to reflect the most recent activity. This view is especially useful in identifying points where a PVM program's execution "hangs" in one or more tasks, say, due to an incorrect message code on a pvm_recv() call.

2.5. Task Output

This view provides access to the output generated by tasks, and displays each line of output in a scrolling text window. This output is automatically captured by PVM whenever a task writes to the standard output device.

3. XPVM Tracing

No annotation or instrumentation of PVM programs is necessary to use XPVM. Once programs have been compiled with an annotated version of the PVM system library, they can have their execution traced. XPVM is actually a PVM task itself, and events are collected by sending messages to the XPVM task using the existing PVM message-passing mechanisms. Tasks spawned from within XPVM will automatically have tracing turned on, with trace events being sent to XPVM for processing. Tasks spawned from a standard PVM console can also be spawned with tracing on, if the "trace" command is executed first, to indicate the task ID of the XPVM task for sending trace events.

While XPVM is receiving trace events and displaying information in views in real time, it also is saving the events in a trace file for playback later. These traces include not only PVM system events, but task output and the addition and deletion of hosts from the Virtual Machine. Trace files are written using SDDF (Self-Defining Data Format) as developed by Reed, et al [ReO91]. Therefore, XPVM traces are compatible with a variety of other tools as well.

In XPVM, traces can be played back continuously, or one event at a time. Traces can also be stopped in real-time play as well, can be single-stepped, and then can resume to continuous play, though potentially somewhat behind the actual program execution depending on the delay time, etc.

References

[Gan19] H. L. Gantt, "Organizing for Work," Industrial Management, Volume 58, August 1919, pp. 89-93.

[HeE91] M. T. Heath, J. A. Etheridge, "Visualizing the Performance of Parallel Programs," IEEE Software, Volume 8, Number 5, September 1991, pp. 29-40.

[HeL91] Virginia Herrarte and Ewing Lusk, "Studying Parallel Program Behavior with Upshot," Technical Paper, Argonne National Laboratory, Mathematics and Computer Science Division, July 1991.

[LeB90] Ted Lehr, David Black, Zary Segall, Dalibor Vrsalovic, "Visualizing Context-Switches Using PIE and the Mach Kernel Monitor," *Proceedings of the 1990 International Conference on Parallel Processing*, Volume II, St. Charles, Illinois, August 1990, pp. 298-299.

[ReO91] D. Reed, R. Olson, R. Aydt, T. Madhyastha, T. Birkett, D. Jensen, B. Nazief, B. Totty, "Scalable Performance Environments for Parallel Systems," *Proceedings of the Sixth Distributed Memory Computing Conference*, IEEE Computer Society Press, April 1991.