8 Advanced Topics

8.1 XPVM

It is often useful and always reassuring to be able to see the present configuration of the virtual machine and the status of the hosts. It would be even more useful if the user could also see what his program is doing—what tasks are running, where messages are being sent, etc. The PVM GUI called XPVM was developed to display this information, and more.

XPVM combines the capabilities of the PVM console, a performance monitor, and a call-level debugger into a single, easy-to-use X-Windows interface. XPVM is available from netlib in the directory pvm3/xpvm. It is distributed as precompiled, ready-to-run executables for SUN4, RS6K, ALPHA, SUN4SOL2, HPPA, and SGI5. The XPVM source is also available for compiling on other machines.

XPVM is written entirely in C using the TCL/TK [12] toolkit and runs just like another PVM task. If a user wishes to build XPVM from the source, he must first obtain and install the TCL/TK software on his system. TCL and TK were developed by John Ousterhout at Berkeley and can be obtained by anonymous ftp to `sprite.berkeley.edu` The TCL and XPVM source distributions each contain a README file that describes the most up-to-date installation procedure for each package respectively.

Figure 8.1 shows a snapshot of XPVM in use.

Like the PVM console, XPVM will start PVM if PVM is not already running, or will attach to the local pvmd if it is. The console can take an optional hostfile argument whereas XPVM always reads `$HOME/.xpvm_hosts` as its hostfile. If this file does not exist, then XPVM just starts PVM on the local host (or attaches to the existing PVM). In typical use, the hostfile .xpvm_hosts contains a list of hosts prepended with an &. These hostnames then get added to the *Hosts* menu for addition and deletion from the virtual machine by clicking on them.

The top row of buttons perform console-like functions. The *Hosts* button displays a menu of hosts. Clicking on a host toggles whether it is added or deleted from the virtual machine. At the bottom of the menu is an option for adding a host not listed. The *Tasks* button brings up a menu whose most-used selection is *spawn*. Selecting spawn brings up a window where one can set the executable name, spawn flags, start position, number of copies to start, etc. By default, XPVM turns on tracing in all tasks (and their children) started inside XPVM. Clicking on *Start* in the spawn window starts the task, which will then appear in the space-time view. The *Reset* button has a menu for resetting PVM (i.e., kill all PVM tasks) or resetting different parts of XPVM. The *Quit* button exits

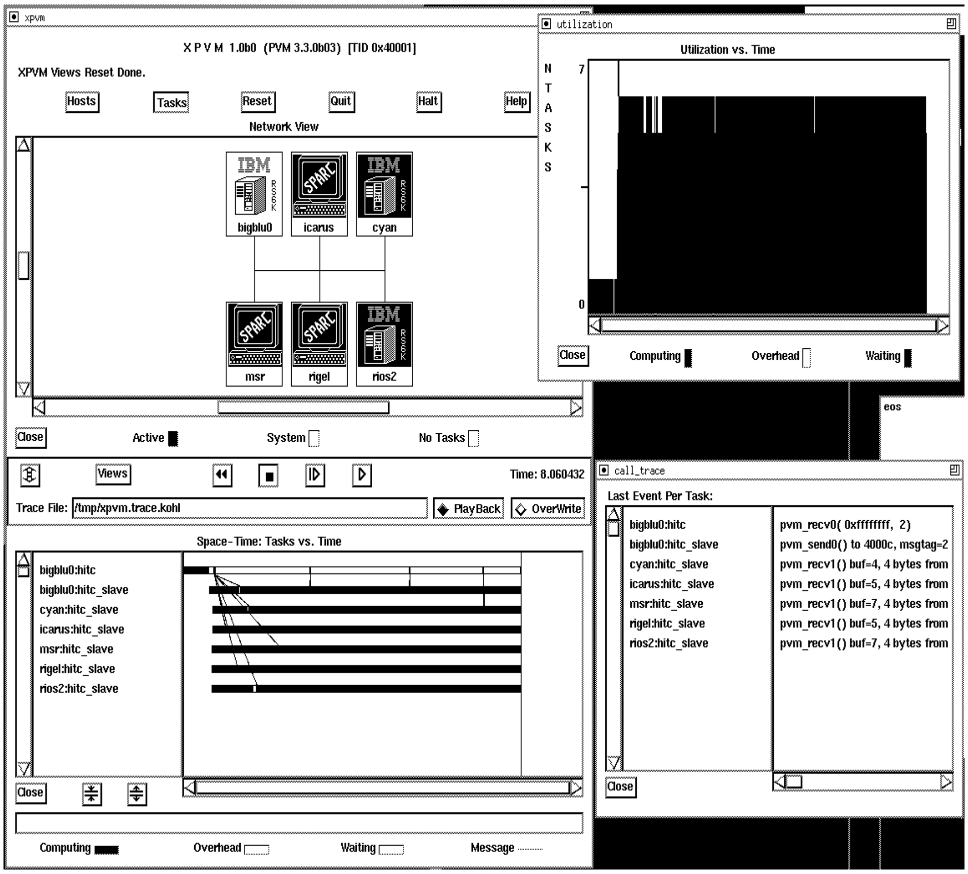

Figure 8.1
XPVM interface - snapshot during use

XPVM while leaving PVM running. If XPVM is being used to collect trace information, the information will not be collected if XPVM is stopped. The *Halt* button is used when one is through with PVM. Clicking on this button kills all running PVM tasks, shuts down PVM cleanly, and exits the XPVM interface. The *Help* button brings up a menu of topics the user can get help about.

During startup, XPVM joins a group called xpvm. The intention is that tasks started outside the XPVM interface can get the TID of XPVM by doing `tid = pvm_gettid( xpvm, 0 )`. This TID would be needed if the user wanted to manually turn on tracing inside such a task and pass the events back to XPVM for display. The expected TraceCode for these events is 666.

While an application is running, XPVM collects and displays the information in real time. Although XPVM updates the views as fast as it can, there are cases when XPVM cannot keep up with the events and it falls behind the actual run time.

In the middle of the XPVM interface are tracefile controls. It is here that the user can specify a tracefile—a default tracefile in /tmp is initially displayed. There are buttons to specify whether the specified tracefile is to be played back or overwritten by a new run. XPVM saves trace events in a file using the "self defining data format" (SDDF) described in Dan Reed's Pablo [15] trace playing package. The analysis of PVM traces can be carried out on any of a number of systems such as Pablo.

XPVM can play back its own SDDF files. The tape-player-like buttons allow the user to rewind the tracefile, stop the display at any point, and step through the execution. A time display specifies the number of seconds from when the trace display began.

The *Views* button allows the user to open or close any of several views presently supplied with XPVM. These views are described below.

8.1.1 Network View

The *Network* view displays the present virtual machine configuration and the activity of the hosts. Each host is represented by an icon that includes the PVM_ARCH and host name inside the icon. In the initial release of XPVM, the icons are arranged arbitrarily on both sides of a bus network. In future releases the view will be extended to visualize network activity as well. At that time the user will be able to specify the network topology to display.

These icons are illuminated in different colors to indicate their status in executing PVM tasks. Green implies that at least one task on that host is busy executing useful work. Yellow indicates that no tasks are executing user computation, but at least one task is busy executing PVM system routines. When there are no tasks on a given host, its icon is left uncolored or white. The specific colors used in each case are user customizable.

The user can tell at a glance how well the virtual machine is being utilized by his

PVM application. If all the hosts are green most of the time, then machine utilization is good. The Network view does not display activity from other users' PVM jobs or other processes that may be running on the hosts.

In future releases the view will allow the user to click on a multiprocessor icon and get information about the number of processors, number of PVM tasks, etc., that are running on the host.

8.1.2 Space-Time View

The *Space-Time* view displays the activities of individual PVM tasks that are running on the virtual machine. Listed on the left-hand side of the view are the executable names of the tasks, preceded by the host they are running on. The task list is sorted by host so that it is easy to see whether tasks are being clumped on one host. This list also shows the task-to-host mappings (which are not available in the Network view).

The Space-Time view combines three different displays. The first is like a Gantt chart. Beside each listed task is a horizontal bar stretching out in the "time" direction. The color of this bar at any time indicates the state of the task. Green indicates that user computations are being executed. Yellow marks the times when the task is executing PVM routines. White indicates when a task is waiting for messages. The bar begins at the time when the task starts executing and ends when the task exits normally. The specific colors used in each case are user customizable.

The second display overlays the first display with the communication activity among tasks. When a message is sent between two tasks, a red line is drawn starting at the sending task's bar at the time the message is sent and ending at the receiving task's bar when the message is received. Note that this is not necessarily the time the message arrived, but rather the time the task returns from pvm_recv(). Visually, the patterns and slopes of the red lines combined with white "waiting" regions reveal a lot about the communication efficiency of an application.

The third display appears only when a user clicks on interesting features of the Space-Time view with the left mouse button. A small "pop-up" window appears giving detailed information regarding specific task states or messages. If a task bar is clicked on, the state begin and end times are displayed, along with the last PVM system call information. If a message line is clicked on, the window displays the send and receive time as well as the number of bytes in the message and the message tag.

When the mouse is moved inside the Space-Time view, a blue vertical line tracks the cursor and the time corresponding to this vertical line is displayed as Query time at the bottom of the display. This vertical line also appears in the other "something vs. time" views so the user can correlate a feature in one view with information given in another view.

The user can zoom into any area of the Space-Time view by dragging the vertical line with the middle mouse button. The view will unzoom back one level when the right mouse button is clicked. It is often the case that very fine communication or waiting states are only visible when the view is magnified with the zoom feature. As with the Query time, the other views also zoom along with the Space-Time view.

8.1.3 Other Views

XPVM is designed to be extensible. New views can be created and added to the *Views* menu. At present, there are three other views: task utilization vs. time view, call trace view, and task output view. Unlike the Network and Space-Time views, these views are closed by default. XPVM attempts to draw the views in real time; hence, the fewer open views, the faster XPVM can draw.

The Utilization view shows the number of tasks computing, in overhead, or waiting for each instant. It is a summary of the Space-Time view for each instant. Since the number of tasks in a PVM application can change dynamically, the scale on the Utilization view will change dynamically when tasks are added, but not when they exit. When the number of tasks changes, the displayed portion of the Utilization view is completely redrawn to the new scale.

The Call Trace view provides a textual record of the last PVM call made in each task. The list of tasks is the same as in the Space-Time view. As an application runs, the text changes to reflect the most recent activity in each task. This view is useful as a call level debugger to identify where a PVM program's execution hangs.

Unlike the PVM console, XPVM has no natural place for task output to be printed. Nor is there a flag in XPVM to tell tasks to redirect their standard output back to XPVM. This flag is turned on automatically in all tasks spawned by XPVM after the Task Output view is opened. This view gives the user the option to also redirect the output into a file. If the user types a file name in the "Task Output" box, then the output is printed in the window and into the file.

As with the trace events, a task started outside XPVM can be programmed to send standard output to XPVM for display by using the options in pvm_setopt(). XPVM expects the OutputCode to be set to 667.

8.2 Porting PVM to New Architectures

PVM has been ported to three distinct classes of architecture:

- Workstations and PCs running some version of Unix
- Distributed-memory multiprocessors like the Intel hypercubes