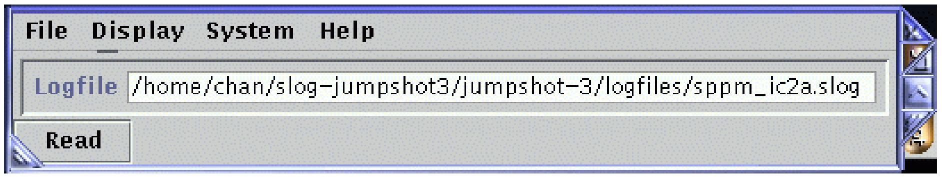
A Tour of Jumpshot-3

Anthony Chan, William Gropp, Ewing Lusk

4/27/2000

Abstract

Jumpshot-3 is a display program for trace data in scalable log format, [SLOG-1], which is designed to store a large number of drawable objects[SLOG-2] generated from the backend of a parallel program, for example AIX's UTE slog-merge or MPICH's MPE profiling library. Basically, Jumpshot-3 is a GUI to display rectangles that represent the various MPI and user-defined states and arrows indicating messages exchanged between those states. This article illustrates the basic functionalities of Jumpshot-3 through a step-by-step guide of how one would usually use Jumpshot-3 to view a big logfile.



Logfile /home/chan/slog-jumpshot3/jumpshot-3/logfiles/sppm_ic2a.slog

Figure 1: The Main Control Window of Jumpshot-3

Background

Assume you are trying to view a slog file called sppm_ic2a.slog[1], which is generated from a run of the MPI program sppm on IBM's SP with shared-memory nodes. The logfile is located in the directory ~/jumpshot-3/logfiles/. Also assume that Jumpshot-3 has been successfully installed[2]. Typing the full pathname to the Jumpshot script which is located at ~/jumsphot-3/bin/ will bring up Jumpshot-3.

Main Control Window

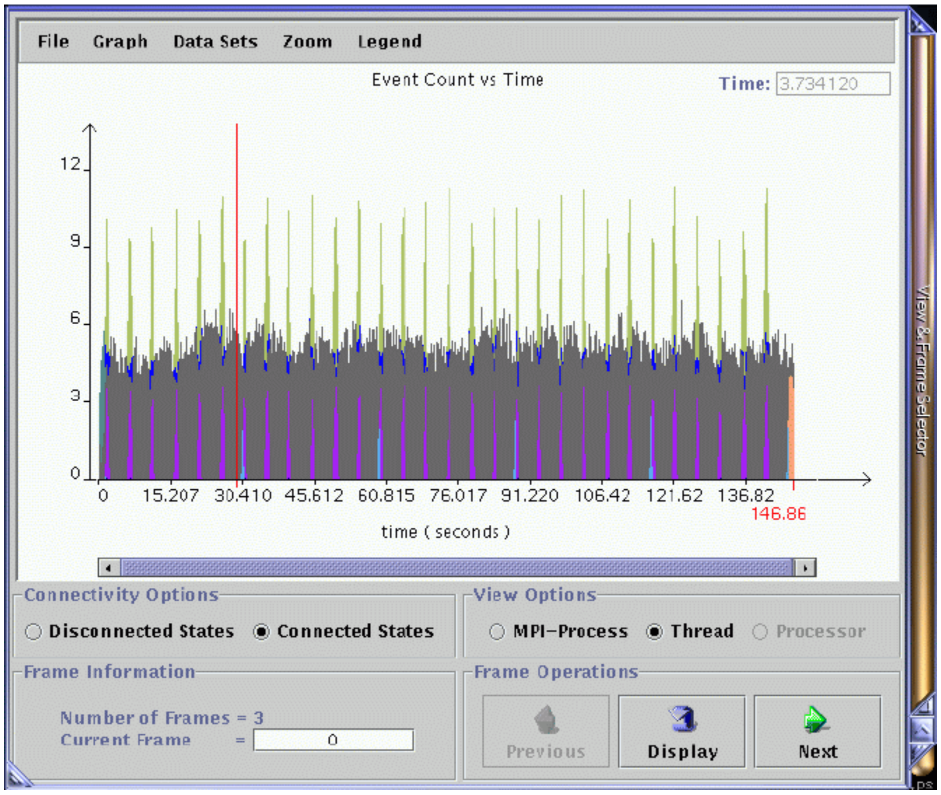
The first window Jumpshot-3 pops up is the Main Control window, which is shown in Figure 1. The *File* menu tab brings up the file selection menu. Here, doubly click to the right directory where sppm.slog is located, and highlight sppm.slog as the logfile to be processed. Then the path of the logfile relative to the current working directory will be printed in the logfile box, in the middle of the main control window. As soon as the pathname of the logfile appears, the *Read* button will be enabled. Click on the *Read* button to display the "Preview" of the logfile.

The *System* menu tab allows you to choose the specific "Look & Feel" for the whole GUI system of Jumpshot-3[3]. The *Help* menu tab provides electronic documentations to Jumpshot-3, including three submenus: *Manual*, *Tour*, and *About*. *Manual* provides a brief explanation of all the buttons in

[1]The logfile is provided by Dave Wootton at IBM.

[2]For details on how to install and start running Jumpshot-3, see README.slog and UserGuide.txt in ~/jumpshot-3.

[3]For Unix system, only Metal and Motif Look & Feel are allowed because of the license issue(from Sun's documentation).



Time: 3.734120

Connectivity Options

Disconnected States [✓ Connected States]

View Options

MPI-Process [✓ Thread] Processor

Frame Information

Number of Frames = 3
Current Frame = 0

Frame Operations

Figure 2: The Preview: View & Frame Selector Window.

Jumpshot-3. *Tour* is the HTML version of this document. *About* provides the version and contact information for the copy of Jumpshot-3 being used.

Statistical Preview

The goal of the Statistical Preview is to help users select a frame for viewing. It consists of two windows. The major window shown in Figure 2, is titled "View & Frame Selector", but is usually called "Preview". It contains a graphical representation of all the activities going on during the run of the program sppm. Activities in "Preview" are computed in the following way. The whole duration of the job sppm is divided into 512 time bins. For each time bin, there are counters for each state and arrow that occur

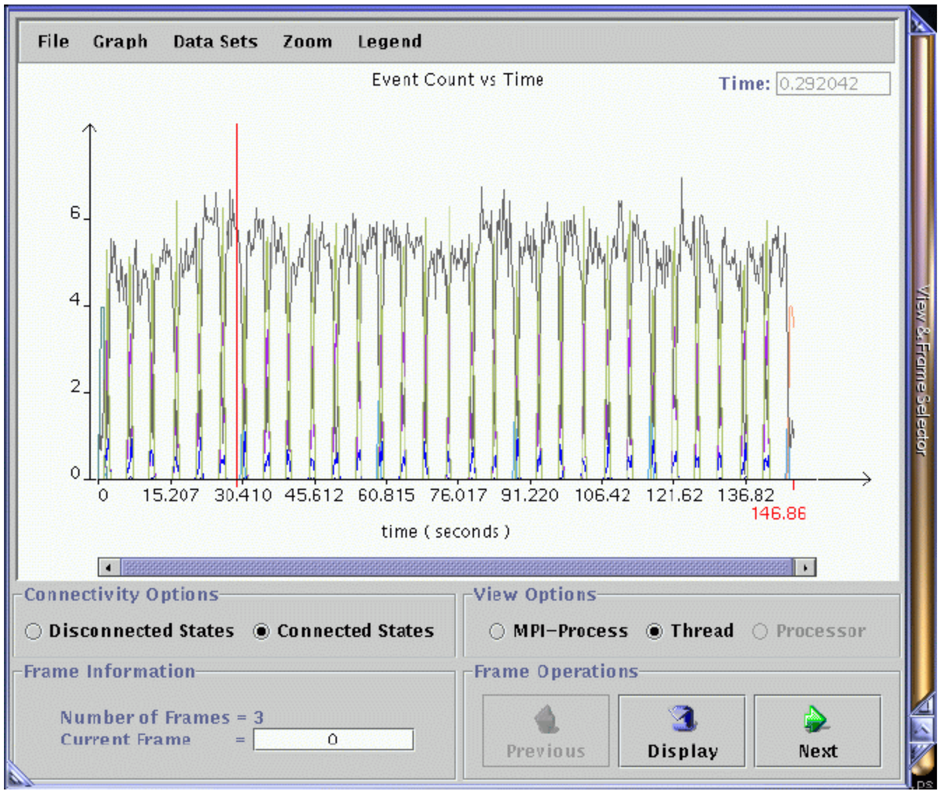


Figure 3: Preview in non-accumulative curve mode.

in the time bin. The counter of each state/arrow will be incremented by a statistical weight that equals the ratio of the duration of state/arrow to the width of time bin that the state/arrow is in. Because of this definition of the statistical weight for each state/arrow, very short duration states or arrows become statistically insignificant and will not contribute much to the accumulative activities of all the states and arrows. The visual consequence is that states/arrows with very short duration are not noticeable in “Preview”. The graphical representation of the activities of the logfiles is, by default, in accumulative histogram mode, which can be turned on/off through the button selection sequence *Graph->Type->Bar*. The other mode is the noncumulative curve mode which can be enabled by the button selection sequence *Graph->Type->Line*. It is shown in Figure 3. Since it is noncumulative, as in histogram mode, it distinguishes the relative importance of various states. In the lower left-hand corner of “Preview”, there is a “Frame Information” panel, which lists the number of frames in the logfile, *N*. In this case, there are only 3 frames in the file. The frame size is determined by the program that generates the slog files[4]. In general, the smaller the frame, the better the performance of Jumpshot-3 will be in refreshing the screen. But if an overly small frame size is used, it will be increasingly difficult to navigate to the frame that is being selected. And each frame will contain too little data for the user to form a coherent overall picture of the code. Next to the label *Current Frame*, there is a dialog box that shows the current frame index, which proceeds from *0* to the last frame of the file, *N-1*, and is initialized to *0* when the GUI starts. The user can change the frame index by typing in the dialog box or through buttons in the "Frame Operations", where *Previous* and *Next* buttons decrement and increment the frame index, respectively. But the user is also allowed to click on the graphical display to select a frame of interest. A red line will then show up in the graph to highlight the frame selected. Currently the red line is located in the middle of the frame, so the red line is moved nonuniformly along the x-axis. The redundancy in frame selection operations allows the end user to fine tune which frame to view when the logfile contains a lot of frames. For an MPI program running in a multithreaded environment as in the AIX SMP box, the logfile may contain information about threads. In that case, the "Connectivity Options" and "View Options" panels become very useful. The user can choose different combinations of these options to see how threads are dispatched and used in an MPI Program. More details will be in the next section.

Next to the "Preview" there is a smaller window titled "Legend", which contains a set of radio buttons with different colors. They are tagged with the names of the MPI and user-defined calls made in the code sppm. An illustration of "Legend" is shown in Figure 4. Below the list of radio buttons there are *Select/Deselect*, *All*, *None,* and *Change Color* buttons.

"Preview" and its corresponding "Legend" windows meant to be used side by side to help you select frames that are of interest. If some of the states displayed in the "Preview" seem useless or confusing, it can be easily removed. First highlight the state through click on the name in the "Legend"; then click on *Select/Deselect* button to remove them from "Preview". In Figure 5, all MPI and system-related states are deselected in "Preview" to highlight the

[4]In AIX's UTE environment, a program called slogmerge allows user to define the size of the frame. In MPICH's MPE environment, direct logging mechanism implicitly assume a frame size. However, end user can change the frame size through the use of program clog2slog.

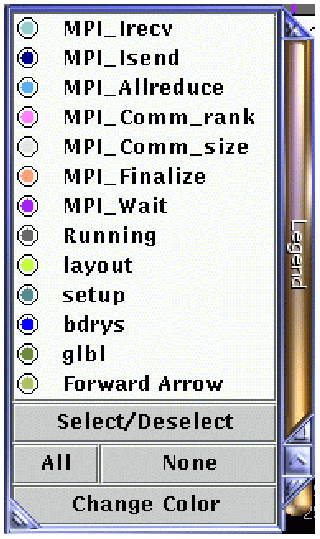
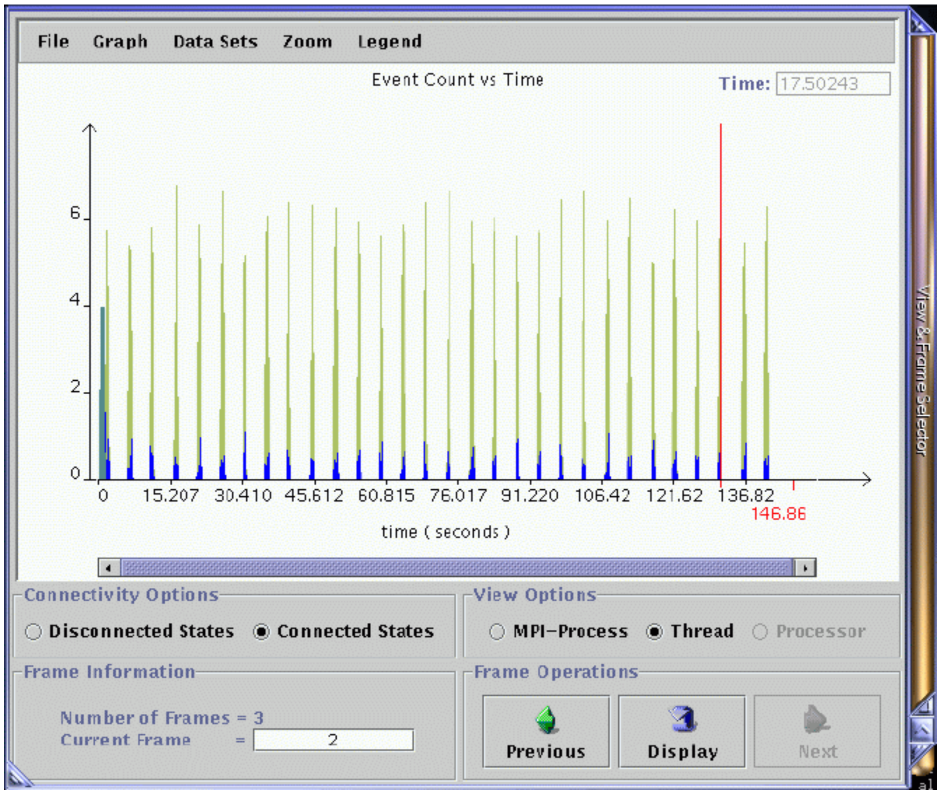


Figure 4: The Legends of Preview

user-defined states “layout”, “setup”, “bdrys” and “glbl”. It becomes apparent that all the communications between the user MPI processes are done in regular interval. Also, all communications(i.e. arrows) are within the user-defined state “bdrys”. Assume you are interested in the behavior of the program between the last two instances of MPI_Allreduce. Then you can simply click on *None* to deselect all the states and arrows, highlight MPI_Allreduce, then click *Select/Deselect* to display only MPI_Allreduce in the “Preview”. Now click on any region between the last two MPI_Allreduce’s in the “Preview”. Then the frame index is changed from 0 to 2, and a red line is positioned in the middle of the frame 2 and is also between the last two instances of MPI_Allreduce. Now select the properties of the frame to be viewed. Most end users who are interested in the performance of the code sppm should select the *Connected States* option with either *MPI-Process* or *Thread* view. If you are interested in how the operating system dispatches threads among CPUs allocated for the job, you may want to select the *Disconnected States* option in the *Processor* view.



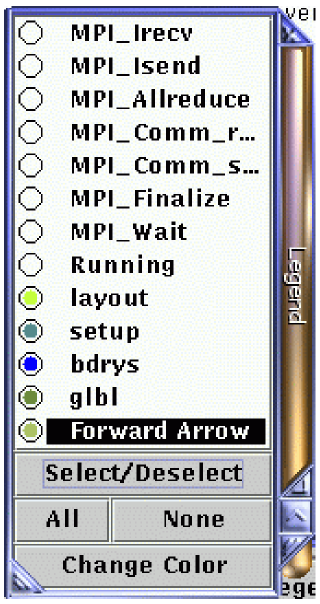
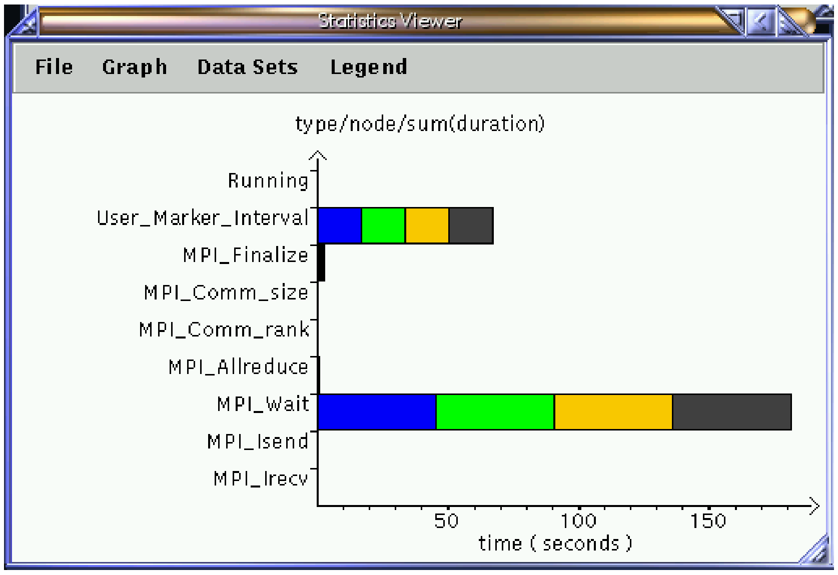


Figure 5: All MPI and OS-related states are deselected in the "Preview" and "Legend" to highlight the user-defined states and the communication among the processes.



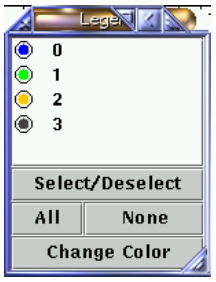


Figure 6: The Statistics Viewer and its “Legend” windows.

Statistics Viewer

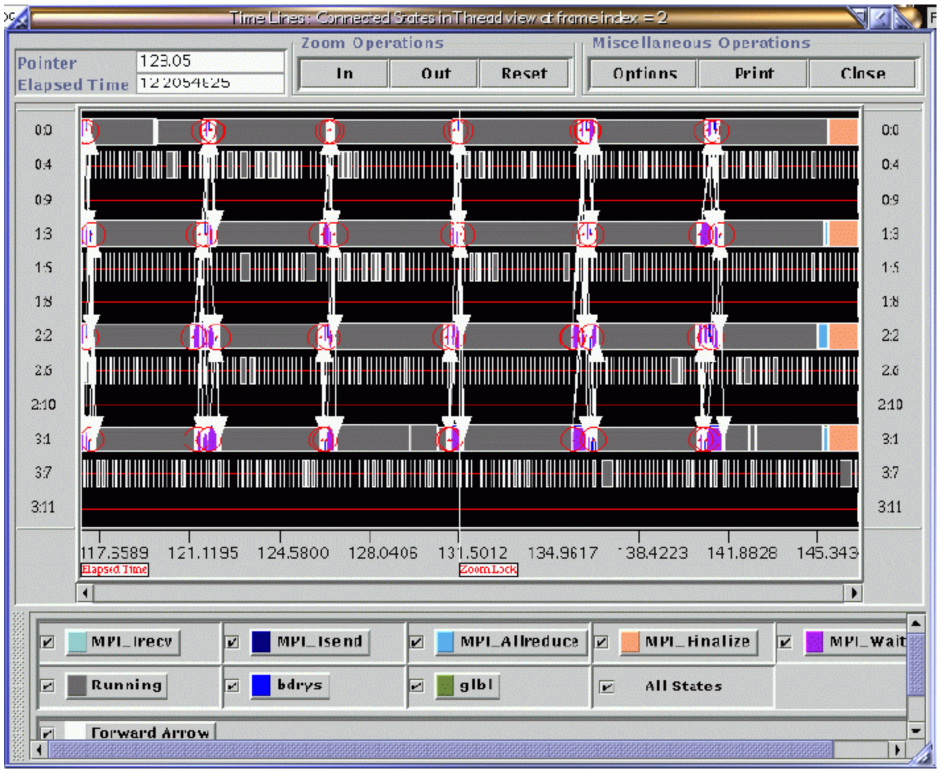
The Statistics Viewer can be invoked from the "Preview" through the button sequence *File->View Statistics.* Once the viewer is up, click on *File* to select a statistics file[5] for processing. As shown in Figure 6, the statistics file of sppm, sppm.stats.0301, is displayed in the Statistics Viewer and its corresponding "Legend" window. The "Legend" window works like that of "Preview". The title in the Statistics Viewer provides crucial information regarding the label of the y-axis in the viewer and that of the "Legend window". Usually the title has the form, *Viewer_ Yaxis_ Label / Legend_ Label / Name_ Of_ Statistics.* In Figure 6, the viewer's y-axis label is "type"(i.e. MPI and user-defined states) and the label for entities in the "Legend" window is "node" ID. The name of the statistics is called "sum(duration)". The statistics indicate that the state MPI_Wait takes the most time in the run and consumes an equal amount of time in each node. The second most time-consuming state is User Marker Interval, and again each node uses similar amount of time in the User Marker Interval.

The statistics file usually contains more than one set of statistics. Another set of statistics can be selected through *the Graph* menu tab. Clicking on the menu tab will pull down a menu with all available statistics data in the file. As soon as a different set of statistics is selected, both the viewer and the "Legend" windows will be updated. When you want to view another statistics file, be sure to close the current opened file before opening another file. Otherwise, all newly opened statistics will be added to the existing ones under the *Graph* menu tab.

Time Lines Window

Let's assume that you select *Connected States* in *Thread* view in "Preview", when you click on the *Display* buttons in the "View & Frame Selector" window. A Time Lines window as shown in Figure 7 will pop up. The Time Lines window provides a detailed display of the sppm trace data as a GANTT chart with the x-axis as time and the y-axis as thread ID. The control buttons in the top panel provide zoom *IN* and *OUT* operations around the zoom

[5]In AIX's UTE environment, a program like "utestat" is used to generate the statistics file. In MPICH's MPE profiling environment, there is no tool which can generate a separate statistics file yet.



Pointer 123.05
Elapsed Time 122054625

Figure 7: Timeline window in *Thread* view

focus, which could be set by putting the cursor at the point of interest on the GANNT chart and pressing the key "z". In Figure 7, the zoom focus is marked by a white line drawn from the top to the botton of the diagram and is labeled as ZOOM LOCK in red.

After being zoomed in several times, the Time Lines window looks like Figure 8. At this resolution, many more details are exposed. For instance, the MPI_Isend states that are in navy blue in the figure become noticeable. Clicking on any rectangle in the Time Lines canvas will pop up a "Rectangle Info" box which contains various information regarding the subroutine call(e.g. start and end time of the call), and various call arguments and instruction address(es) if there are any. Clicking on the "Rectangle Info" box again will remove the box from the screen. Also, clicking on the red circle at the end of the arrow will pop up a "Arrow Info" box, which provides a function similar to that of "Rectangle Info". In Figure 8, MPI_Isend, MPI_Irecv, and MPI_Wait are all nested within the user-defined state "bdrys". This suggests that "bdrys", which is a user-defined subroutine call, makes all these MPI calls. Also, the regularity of the pattern of arrows can provide insight about how data are exchanged.

Several tricks and hidden operations are worth mentioning. First, it is the trick in locating small rectangles: since all the rectangles have a white border surrounding them, when many small rectangles are next to each other in very low resolution, they form a completely white rectangle. A totally white rectangle usually means a lot more details are hidden inside. Second, a trick about scrolling: you can drag on the scroll tab to advance to later time, but scrolling becomes slow when there are many threads with a lot of objects. In this case, click on the white space between the scroll tab and the end arrow tab in the direction that you want to advance to. The operation will allow the next time frame in the canvas to be redrawn immediately. Third, the label of the y-axis in *Thread* view is (MPI-rank, local thread ID), but it is different in different views. Because of the limited space on the canvas, the label of the y-axis is actually written in the tooltip of the two vertical y-axis label. The tooltip can be activated by simply putting the cursor over any y-axis integer doublet for few seconds. Fourth, doubly clicking on any integer doublet label will invoke the "Time Lines Manipulation" window as shown in Figure 9. This window provides various operations on the selected time line, for instance, the adjustment of time line by changing the offset of the time line for alignment of rectangles, or the swapping of different time lines for organizational purposes. Fifth, there is a trick about the Definitions window,

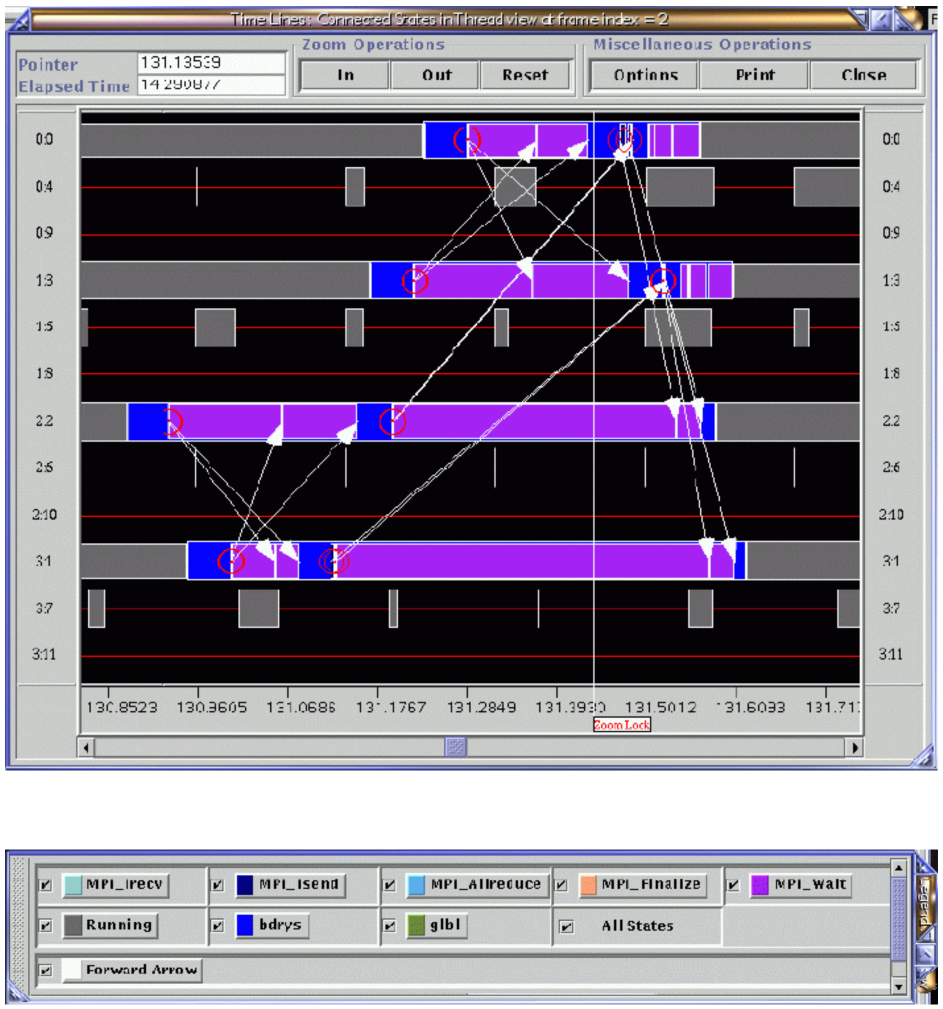
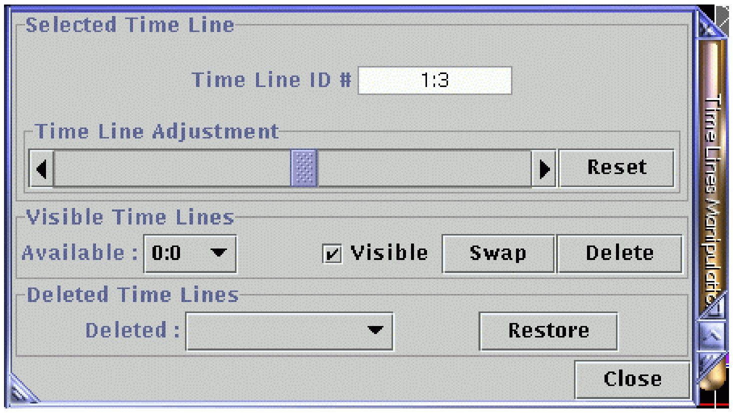


Figure 8: A zoomed in view of the Time Lines window.


Selected Time Line

Time Line ID # 1:3

Time Line Adjustment

Visible Time Lines

Available : 0:0

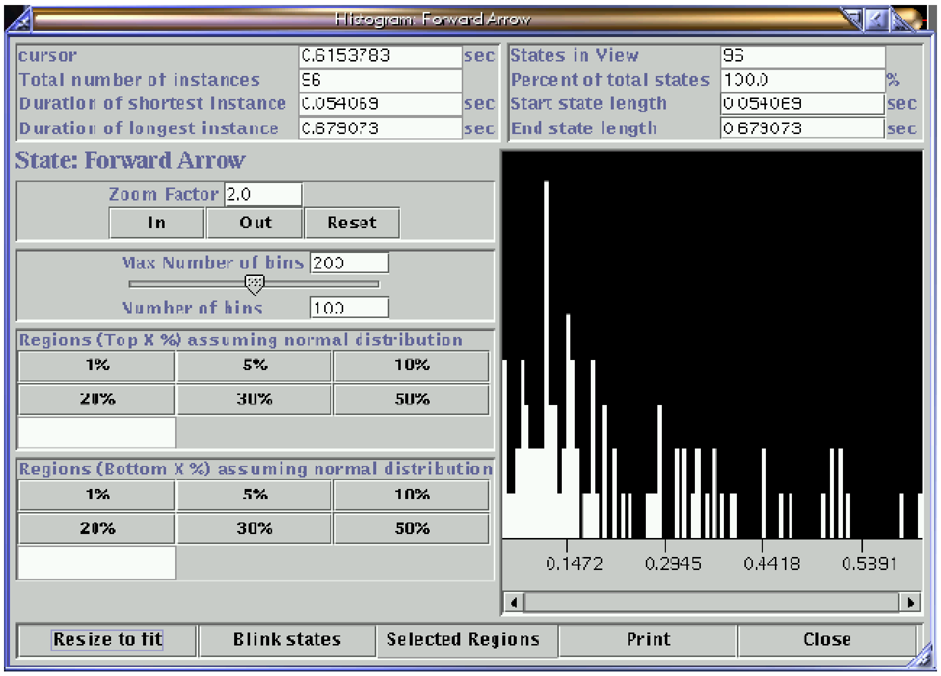
[✓ Visible]

Deleted Time Lines

Deleted :

Figure 9: The Time Lines Manipulation window.

which is located at the bottom of the Time Lines window. The Definitions windows consists of a collection of definition panels, each has a checkbox and a definition button corresponding to each state/arrow in the Time Lines canvas. The Definitions window can be torn out of the Time Lines window as a standalone window, or it can be re-attached to any one of the four sides of the Time Lines window. This allows the user to change the arrangement of the state and arrow definitions to optimize the use of the Time Lines canvas. All states and arrows definitions are checked by default. Unchecking any of definitions will make the corresponding states or arrows invisible. The goal is to help users highlight what they are interesed in by hiding the uninteresting ones. Clicking on any of the definition buttons will bring up the histogram window of the corresponding state/arrow. For example, the forward arrow button will invoke a window as shown in Figure 10. The Histogram window provides basic statistics and statistical distribution of the duration of the selected state. The "Blink States" button enables the selected state/arrow in the Time Lines window to flash. Again the goal is to help highlight the objects. Finally, we note that the memory demand for Time Lines window is very high, especially on Linux box with JDK-1.1(running green thread). Our experience indicates that the amount of X server memory needed to have one instance of Time Lines window running is proportional to the size of the Time Lines window and the color depth of the X server. If one plans to run multiple instances of Time Lines windows at the same time, be sure to have enough physical memory for the X server and Java Virtual Machine.



cursor 0.6153783 sec
Total number of instances 56
Duration of shortest instance 0.054069 sec
Duration of longest instance 0.679073 sec

States in View 95
Percent of total states 100.0 %
Start state length 0.054069 sec
End state length 0.679073 sec

Zoom Factor 2.0
Max Number of bins 200
Number of bins 100

Figure 10: The histogram for the forward arrow in the frame.

Bibliography

[SLOG-1] Anthony Chan, William Gropp, Ewing Lusk, Anthony Bomarcich, Thedore Hoover, Yarsun Hsu, Marc Snir, and Eric Wu, *Scalable Log File Format Specification: Concepts*, DRAFT on 1/25/1999.

[SLOG-2] Anthony Chan, William Gropp, and Ewing Lusk, *Scalable Log Files for Parallel Program Trace Data*, DRAFT on 3/20/2000.