A Tour of Jumpshot-3

Anthony Chan, William Gropp, Ewing Lusk

4/27/2000

Abstract

Jumpshot-3 is the 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, like AIX's UTE slogmerge or MPICH's MPE profiling library. Basically, Jumpshot-3 is a GUI to display rectangles which represent the various MPI and user defined states and arrows which represent messages exchanged between those states. This article serves to illustrate 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.

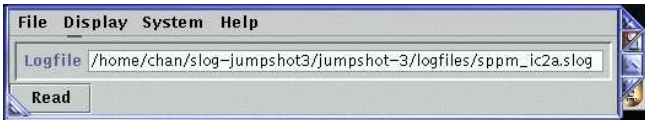

Figure 1: The Main Control Window of Jumpshot-3

Background

Assume one is 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 the Jumpshot-3.

Main Control Window

The first window Jumpshot-3 pops up is the Main Control window which is shown in the 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 will display the "Preview" of the logfile.

The *System* menu tab allows a user 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. This includes sub-menu *Manual*, *Tour* and *About*. *Manual* provides a very brief explanation of all the buttons in 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 a user select a frame for viewing prupose. It consists of two windows. The major window is titled "View &

[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).

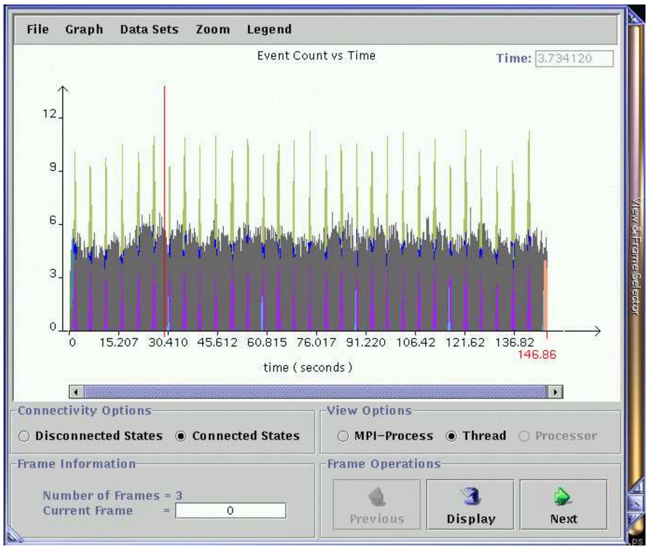

Figure 2: The Preview : View & Frame Selector Window.

Frame Selector" window, which is usually called "Preview" contains a graphical representation of all the activities going on during the run of the program sppm. It is shown in Figure 2. 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 states and arrows occured in the time bin. The counter of each state/arrow will be incremented by the statistical weight which equals to 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 the "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 non-accumulative curve mode which can be enabled by the button selection sequence, *Graph->Type->Line*. It is shown in the Figure 3. Since it is non-accumulative as in histogram mode, it distinguishes the relative importance of various states. In the lower left hand corner of the "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. One may wonder how big is each frame and how it is determined? The frame size is determined by the program that generates the slog files[4]. In generally, the smaller the frame, 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.

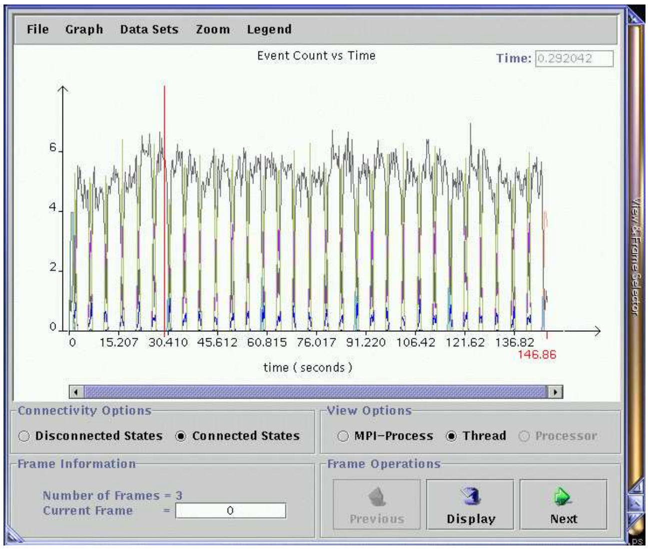

Figure 3: Preview in non-accumulative curve mode.

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 user to form a coherent overall picture of the code. Next to the label *Current Frame*, there is a dialog box which shows the current frame index which starts from *0* to the last frame of the file, *N-1*, and is initialized to *0* when the GUI starts. User can change the frame index by typing in the dialog box or through buttons in the "Frame Operations", where *Previous* and *Next* buttons decrements and increments the frame index respectively. But 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 non-uniformly 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 MPI program running in a multihreaded environment as in AIX SMP box, the logfile may contain various threads information. In that case, the "Connectivity Options" and "View Options" panels become very useful. User is allowed to choose different combinations of these options to see how threads are dispatched and used in a 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 the Figure 4. Below the list of radio buttons, there are *Select/Deselect*, *All*, *None* and *Change Color* buttons.

"Preview" together with its corresponding "Legend" windows are meant to

Figure 4: The Legends of the Preview

be used side by side to help the end users to select frames which are of interest to them. If some of the states displayed in the "Preview" seems useless and confusing, like operating system generated state "Running". In this case, the "Running" state is the one that messes up the overall picture of the code, it can be easily removed by first highlight the state through clicking on the name "Running" in the "Legend". Then click on *Select/Deselect* button will remove "Running" from the "Preview". In Figure 5, all MPI and system related states are deselected in the "Preview" to highlight the 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 one is interested in the behaviour of the program between the last two instances of MPI_Allreduce, one 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 in between the last two instances of MPI_Allreduce. Now it is time to select the properties of the frame to be viewed. Most end users who are interested in the performance of the code sppm should select *Connected States* option with either *MPI-Process* or *Thread* view. If you are intereted in how the operating system dispatches threads among CPUs allocated for the job, you may want to select *Disconnected States* in *Processor* view.

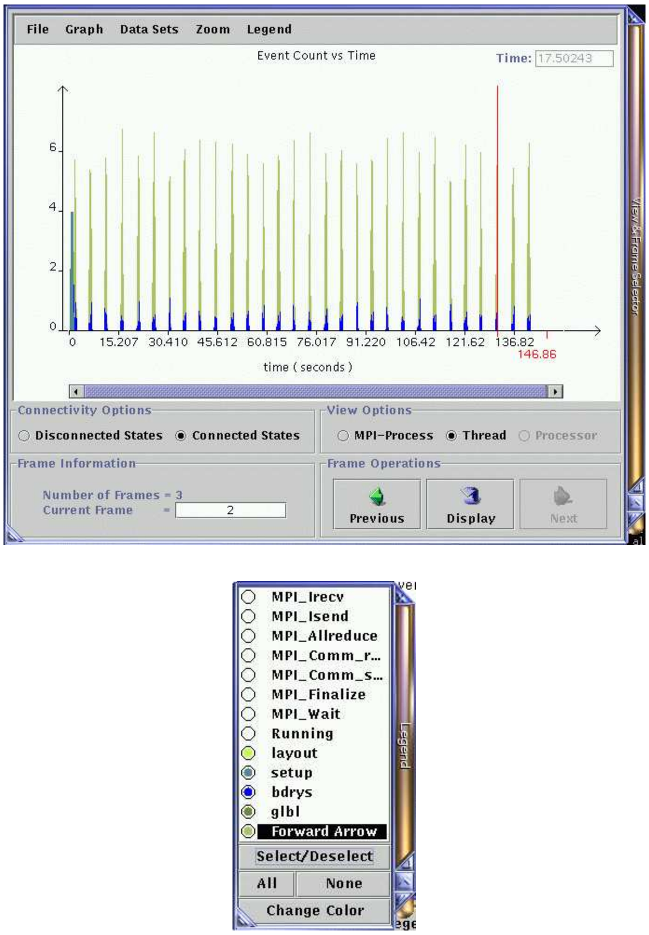

Figure 5: All MPI and OS related states are deselected in the "Preview" and "Legend" to highlight the user defined states and the communication among the processes.

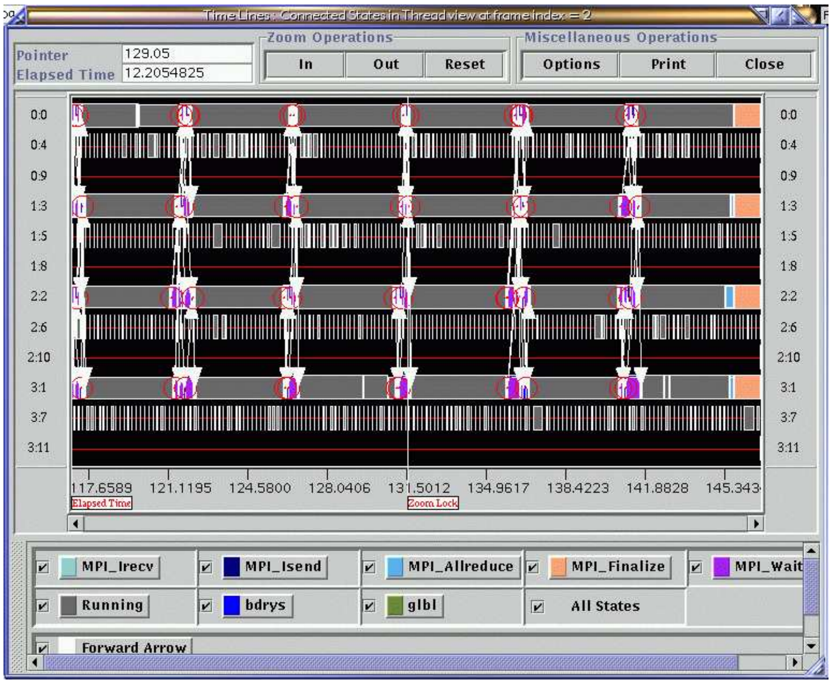

Figure 6: Time Line window in *Thread* view

Time Lines Window

Assume *Connected States* in *Thread* view is selected, as soon as user clicks on the *Display* buttons in the "View & Frame Selector" window. A window as shown in Figure 6 will pop up.

Time Lines window provides a detailed display of the sppm's trace data as a GANTT chart with x-axis as time and y-axis as thread ID. The control buttons in the top panel provide zoom in and out operations around the zoom focus which could be set by putting the cursor at the point of interest on the GANNT chart and press the key "z". In Figure 6, the zoom focus is marked by a white line drawn from top to botton of the diagram and is labeled as

ZOOM LOCK in red.

After being zoomed in several times, the Time Line window looks like Figure 7. At this resolution, a lot more details are exposed. For instance, the MPI_Isend states which are in navy blue in the figure become noticeable. Clicking on any rectangle in the time line canvas will pop up a "Rectangle Info" box which contains various information regarding to the subroutine call, like start and end time of the call, 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

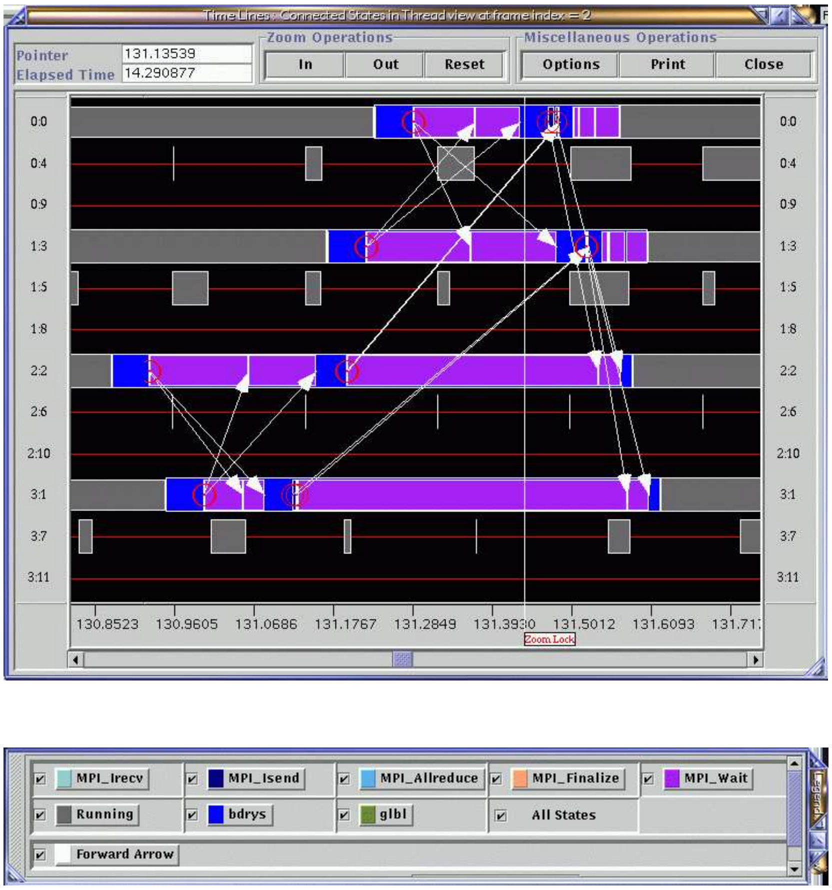

Figure 7: A zoomed in view of the Time LIne window.

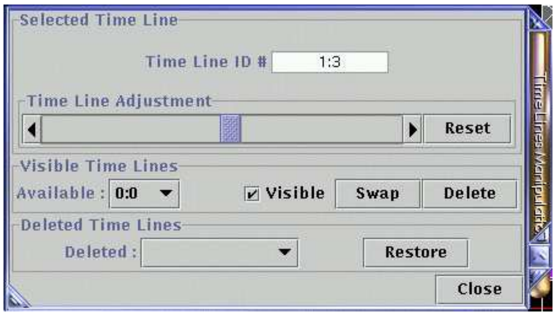

Figure 8: The Time Lines Manipulation window.

up a "Arrow Info" box which provides similar function as "Rectangle Info". In Figure 7, MPI_Isend, MPI_Irecv and MPI_Wait are all nested within the user defined state "bdrys". This suggests "bdrys", which is a user defined subroutine call, makes all these MPI calls. Also the regularity on the pattern of arrows provides insight to the end user of how data are exchanged.

There are several tricks and hidden operations that are worth mentioning. Firstly, it is the trick in locating small rectangles: since all the rectangles have 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. Secondly, a trick about scrolling: user can drag on the scroll tab to advance to later time, but it is slow when there are many threads with a lot of objects. In this scenario, it is recommended to click on the white space between the scroll tab and the end arrow tab in the direction that one would want to advance to. The operation will allow the redraw of the next time frame in the canvas to be carried out immediately. Thirdly, the label of the y-axis in *Thread* view is (MPI-rank, local thread ID), but it is different in different view. 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. Fourthly, doubly click on any integer doublet label will invoke the "Time Lines Manipulation" window as shown in Figure 8. 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, swapping different time lines for organizational purpose, ...Fifthly, this trick is about the Definitions window which is located at the bottom of the Time Line window. The Definitions windows consists of a collection of definition panels. Each definition panel which has a checkbox and a definition button corresponds 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 side of the Time Lines window. This allows user to change the arrangement of the state and arrow definitions to optimize the use of the Time Lines canvas. All the states'

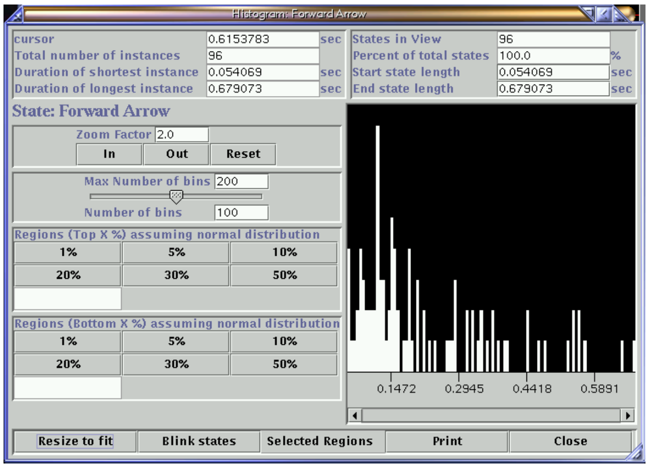

Figure 9: The histogram for the forward arrow in the frame.

and arrows' definitions are checked by default. Unchecking any of definitions will make the corresponding states or arrows invisible. The goal is to help the user to highlight what he/she is 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 the a window as shown in the Figure 9.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 to highlight the objects. Finally, worth mentioning about the memory demand for Time Line window. It is very high especially on Linux box, our experience seems to indicate 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 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.

Bibliography

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

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