Jumpshot-4's User's Guide

Anthony Chan[1], David Ashton[2], Rusty Lusk[3], William Gropp[4]
Mathematics and Computer Science Divsion, Argonne National Laboratory

19th September 2003

[1]chan@mcs.anl.gov
[2]ashton@mcs.anl.gov
[3]lusk@mcs.anl.gov
[4]gropp@mcs.anl.gov

Acknowledgments

We would like to thank Dave Wootton of IBM Poughkeepsie for his valuable suggestions and comments during the development of this tool. This work has been supported in part through the Center for Astrophysical Thermonuclear Flashes at the University of Chicago by the United States Department of Energy under contract B532820. This work was also supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38.

Contents

1 Introduction **4**

2 Data Model **5**

2.1 Understanding the Drawable 5

2.2 Understanding the Preview Drawable 6

2.2.1 Understanding the Preview State Display 12

3 Graphical User Interface **16**

3.1 Main Window 16

3.2 Logfile Convertor Window 16

3.3 Legend Window 20

3.4 Timeline *Zoomable* Window 24

3.4.1 Zoomable and Scrollable Canvas 26

3.4.1.1 Dragged Zoom 26

3.4.1.2 Instant Zoom 27

3.4.1.3 Grasp and Scroll 27

3.4.1.4 Information Dialog Box 28

3.4.2 Toolbar 32

3.4.3 Y-axis Label Panel 32

3.4.4 Row Adjustment Panel 35

3.5 Histogram *Zoomable* Window 35

3.5.1 Summary States 38

3.5.2 Summary Arrows 39

3.6 Preference Window 41

4 Special Features **46**

4.1 Search and Scan Facility 46

4.2 Tuning of the Timeline Window 49

4.3 Estimation of MPI Communication Overhead 50

Chapter 1

Introduction

Jumpshot-4 is the visualization program for the improved scalable logfile format, SLOG-2, which provides a hierarchical structure to store a large number of drawable objects in a very scalable and efficient way for visualization. The new scalable logfile format allows the display program to provide functionalities never made possible before. Level-of-detail support through preview drawables which provides high-level abstraction of the details without reading in huge amount of data into the graphical display engine. New Jumpshot allows seamless scrolling from the beginning till the end of logfile at any zoom-level. In addition, new functionalities like dragged-zoom, grasp and scroll, instant zoom in/out, easy vertical expansion of timelines, cut and paste of timelines are available as well. A new search and scan facility is provided to locate the hard-to-find objects in very large logfile. Also, the histogram module based on user selected duration provides a convenient and graphical way to analyze the statistics of logfile, e.g. easy detection of load imbalance among timelines. The new legend table makes manipulation of the different categories of objects easy. The new viewer also provides an integrated logfile convertor for all known SLOG-2 convertible trace formats, like CLOG, RLOG, and UTE, and it attempts to conform to the standard Look and Feel that is expected by most users.

Chapter 2

Data Model

2.1 Understanding the Drawable

The main visual component in the SLOG-2 visualization program, Jumpshot-4, is the *timeline canvas* which is zoomable and scrollable in both horizontal and vertical axes. The timeline canvas can be thought of as a TIMELINE vs TIME coordinate system. Each point on the canvas is identified by two numbers, a timestamp and a timeline ID. The canvas is where the graphical objects contained in SLOG-2 file are being drawn on. These objects are called *Drawables.* There are 2 kinds of drawable objects. They are *Primitive* and *Composite* drawables. The primitive drawables are the simplest drawables and are considered to be basic elements of SLOG-2 file. They are categorized based on their topological structures. Currently, there are 3 topologies supported in SLOG-2. They are *State*, *Arrow* and *Event*. Both state and arrow are drawables identified by 2 points in the timeline canvas, i.e. a pair of (timestamp, timeline ID) coordinates. State's start timeline ID is the same as its final timeline ID, but arrow is different from state in the way that arrow's start and final timeline IDs may be different. Event consists of only 1 point in the timeline canvas, i.e. it has only 1 timestamp and 1 timeline ID. Composite drawable is more complicated and is constructed by a collection of primitive drawables[1]. In order to centralize the properties of drawables, all the displayable attributes of a drawable are stored in its corresponding *Category* object, e.g. color, legend name, topology and other shared description of a drawable. Both the category and drawable definitions are stored in the SLOG-2 file. These definitions are interpreted and displayed by the display program, Jumpshot-4.

One of the distinct features of Jumpshot is that it uses nested states to show the relationship of functions in the call stack, i.e. nested states corresponds to the nested subroutine calls. Current implementation of the SLOG-2 format stores some of the state nesting information to optimize the performance of the visualization program.

[1] In general, composite drawable can be seen as composed of other simpler composite drawables.

2.2 Understanding the Preview Drawable

A preview drawable is created as a result of the renormalization process of the SLOG-2 format. The renormalized object provides a high-level description of what is going on within the (timeline vs time) region where the preview object spans. Preview drawable is designed to amalgamate real drawables of same topological type, e.g. preview state amalgamates only states. So preview drawable is always a primitive drawable in the renormalization scheme. There are currently 3 different types of preview drawables: *Preview_State, Preview_Arrow, and Preview_Event.* Therefore one preview drawable is for each supported topology of primitive drawable. Up to three preview categories could show up in the Legend window of the display program as shown in Figure 3.5. The Legend window contains a table of legends which are basically visual representation of category objects mentioned earlier. Each legend provides an interface to the user modifiable part of the corresponding category that is relevant to the display program.

Figures 2.1 to 2.5 illustrate the visual transition from preview drawable to its detailed content of the first 5 processes of a 16 processes MPI slog2 file when the timeline canvas is being zoomed-in. The sequence of figures is generated by zooming in a marked region in each successive figure in the sequence. The marked region is shaded and is bounded by a pair of white lines. A magnifying glass with plus sign in the center is the cursor that marks the end of the zoom region. Figure 2.1 is a typical timeline canvas where most of real drawables are still buried inside their preview drawables. In the figure, there are preview arrows, preview states in the front and some long running real states in the back.

Each thick yellow line is a *preview arrow* which represents a collection of arrows between its 2 ending timelines. The start and final timestamps of preview arrow are the extremes of all real arrows amalgamated inside the preview object. Notice that the beginning or ending timestamp of a preview arrow does not necessarily mean that there is any arrow starting and ending at that times, it just indicates that there are arrows starting or ending within these 2 times and between the 2 marked timelines. The thickness of the preview arrow denotes the number of real arrows represented by the preview object. Because of the limitation on the available thickness that preview arrow can have, the thickness of the preview object is set to equal to the order of magnitude of the number of real objects amalgamated. So same thickness in two different preview arrows does not mean that they contain exactly the same number of real arrows, but does mean that the numbers of real arrows contained in the preview objects are within the same order of magnitude, i.e. within a constant multiplicative factor as defined by PREVIEW_ARROW_LOG_BASE in Preference window shown in Figure 3.27 and in Table 3.18. Different thickness in preview arrows indicates more than one multiple of the constant factor difference in the number of real arrows between the preview objects.

The rectangle that has horizontal strips of colors is *preview state.* The different colors inside a preview state represent the various categories of real states that are amalgamated within the time range of the preview state. Depending on the PREVIEW_STATE_DISPLAY value selected in the pulldown menu at the top of left side Y-axis label [2], the distribution and the heights of the strips

[2] In Preference window as shown in Figure 3.27 and in Table 3.18, there is also a PREVIEW_STATE_DISPLAY variable. The variable determines the initial PREVIEW_STATE_DISPLAY used when Timeline window is first made visible.

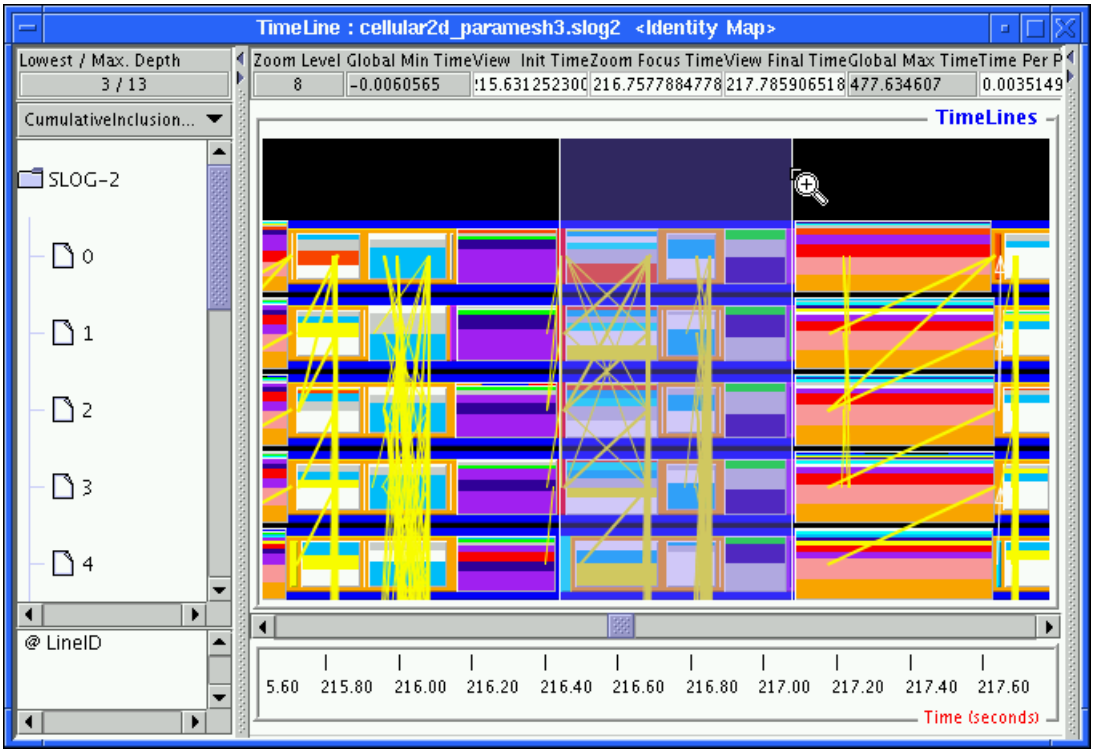


Figure 2.1: A typical zoom-out view of preview states and arrows. The region that is marked by a pair of white lines and the zoom-plus cursor is being zoomed in, i.e. enlarged, in the next figure.

can be changed drastically. One of the display options for preview state is *CumulativeInclusionRatio*. With this option, the strips are arranged in decreasing height order, sort of like a small cumulative histogram. The tallest strip at the bottom of the preview state corresponds to the category of states that contribute the longest total duration in the specified time range *inclusively*, i.e. disregarding the nesting state order. This visual representation aims to tell what state categories could be within the span of the preview state and which state category contributes the most statistically to the specified time range, so user can decide where to zoom in to find out more details. In a sense, the preview states provide a global coarse-grain summary of what is going on without losing as much details as the preview found in older Jumpshot, i.e. Jumpshot-3. Compared with Jumpshot-3's preview which has averaged out the information about timeline IDs, the new preview states retain the timeline ID information and that may lead to early detection of load balancing problem before zooming in to see all the real states.

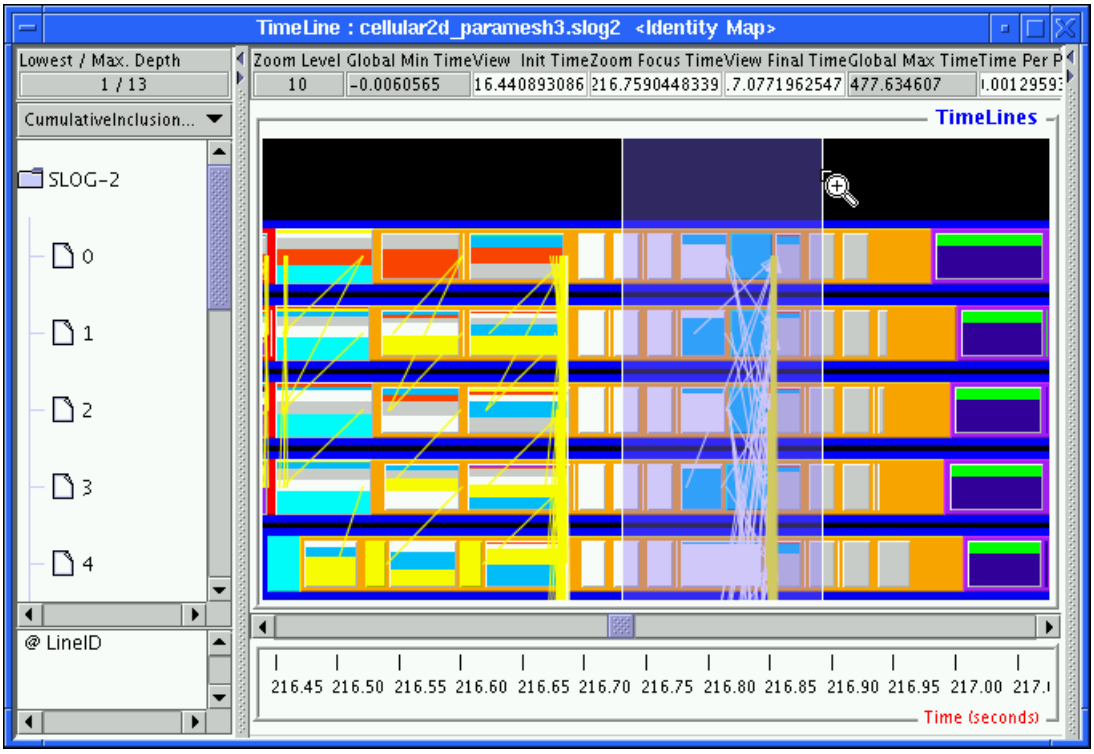


Figure 2.2: The next zoom-in view of Figure 2.1.

Figure 2.2 shows a more zoom-in view of the region marked by the pair of white lines in Figure 2.1. In Figure 2.2, some of the preview arrows have disappeared and are replaced by real arrows, i.e. the white arrows. Also, some of the stripped preview states have split into several small preview states of identical color, i.e. the white and gray states, to show more detailed distribution. Another important feature of preview state becomes apparent in the figures: Preview states are properly nested within real states. In the most expanded Y-axis label view, preview state is always on top of other nested

states[3], i.e. states that enclose the preview state are alway real states. A good visual example is shown in Figure 2.2 where all the white, turquoise and gray preview states[4] are sitting on top of the long orange and dark royal blue states. This indicates that the white, turquoise and gray real states are all nested inside the long running orange and dark royal blue states.

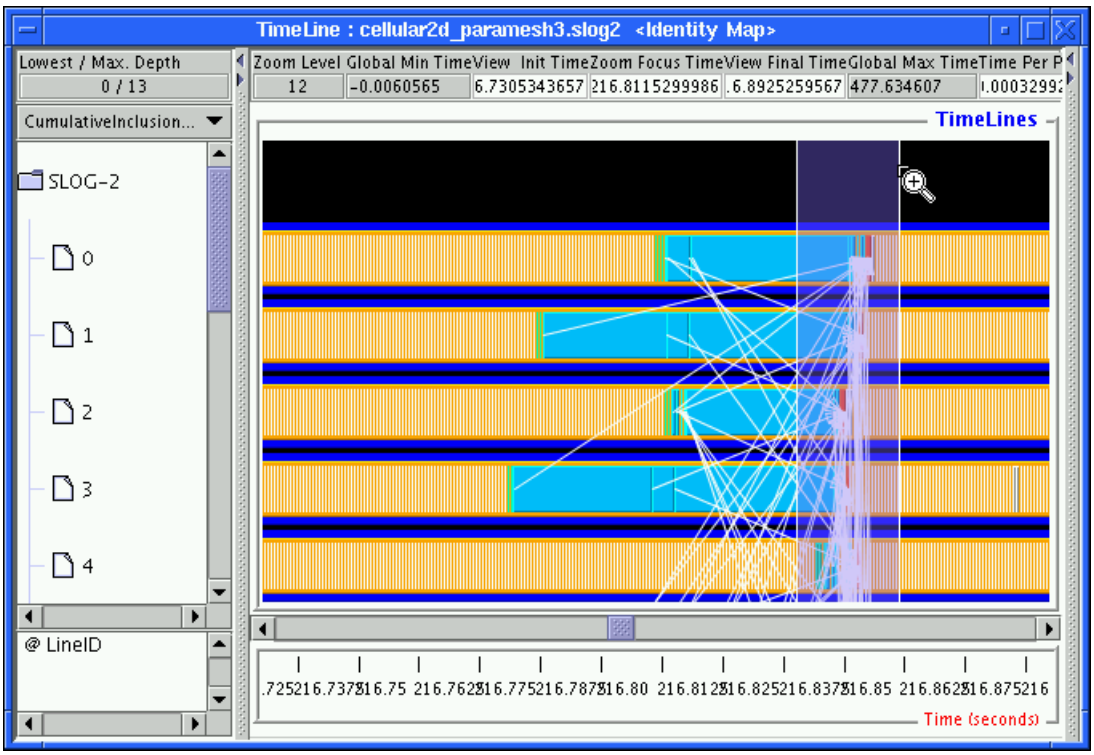


Figure 2.3: The next zoom-in view of Figure 2.2.

Figure 2.3 is the zoom-in view of the region marked by the pair of white lines in Figure 2.2. Comparing these 2 figures, all the preview drawables have disappeared and are replaced by real drawables. Each white preview state are replaced by hundreds of white real states, the same is also true for the gray preview states that sit to the right of the turquoise states[5]. The preview arrows are all replaced by the real arrows. It becomes apparent that the white lines marked region in Figure 2.2 provides a good description of what is going on in Figure 2.3 but at the same time it reduces the number of drawables drawn on the canvas by a factor of 100. Another way of seeing this benefit is to find out the exact number of real drawables amalgamated by the preview objects within the zoomed region.

[3]Only in slog2 file that has multiple ViewMaps and where timelines can be collapsed, i.e. AIX's UTE generated slog2 file, preview state can be nested with other preview state in collapsed Y-axis label view.

[4]when a preview state contains only real states of one single category, it may appear like a real state in the timeline canvas. The only sure way to tell the difference is to bring up the Drawable Info Box by right clicking on the state.

[5]In order to speed up graphics performance of the display program, an aggressive algorithm has been employed to eliminate drawing states that are closely packed together within the nearest neighboring pixels. Together with the fact that the number of pixels available is less than the number of non-overlap states in the region, the number of the real states may sometimes not appear as numerous as the Drawable Info Box of preview state indicates. In that case, a further zoom in will be needed to confirm the case as shown in Fig. 2.4.

This can be achieved by right clicking on the preview drawable and the result is shown in Figure 3.16.

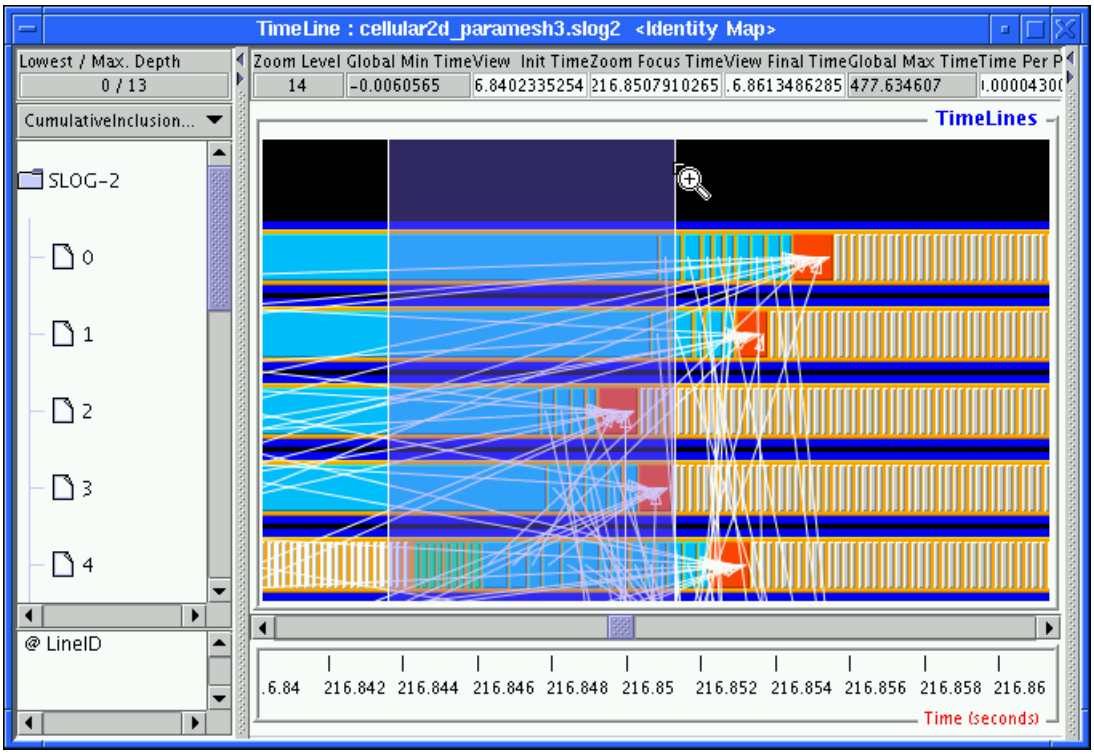


Figure 2.4: The next zoom-in view of Figure 2.3.

Further zooming into the white lines marked region in Figure 2.3 enlarges the real drawables that are displayed in the figure. The enlarged view is shown in Figure 2.4. The densely packed states and arrows become more distinguishable. Another zooming in around the white lines marked region in Figure 2.4 enlarges the real drawables into easily separable objects as shown in Figure 2.5.

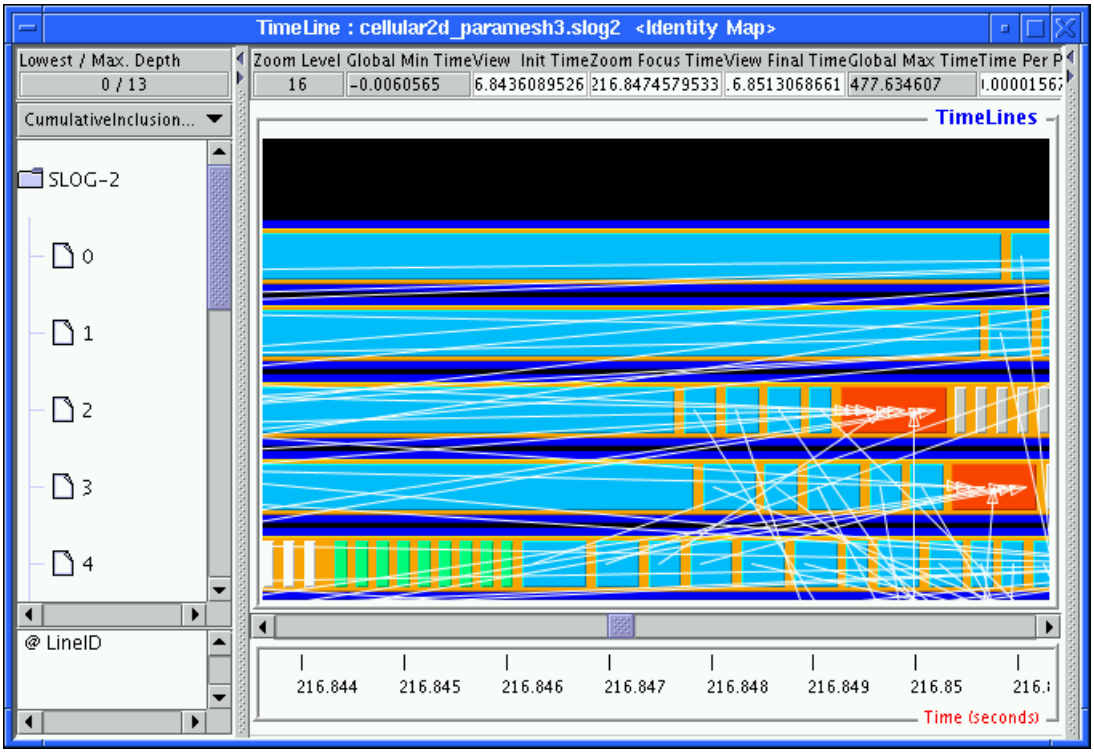


Figure 2.5: The next zoom-in view of Figure 2.4.

2.2.1 Understanding the Preview State Display

So far only one of the representations of preview state, *CumulativeInclusionRatio*, is used to illustrate the concept and representation of the preview state. Jumpshot-4 actually uses several different representations of preview state. All these representations are based on 2 ratios stored in SLOG-2 file. They are called *Inclusion Ratio* and *Exclusion Ratio.* Inclusion ratio is computed without taking into account of the nesting order of the states. States which are either nested inside or enclose other states contribute equally to the inclusion ratio. The end result is that the sum of all inclusion ratios from all state categories in a preview state could easily be larger than 1. On the other hand, exclusion ratio is specifically computed to exclude the overlap of the nested state from the enclosing state. Therefore the sum of exclusion ratios of all state categories in a preview state is guaranteed to be less than or equal to 1.

The motivation of computing these 2 ratios is to satisfy two opposite needs of preview state. If you are a MPI application developer and you have put a lot of user-defined states in your SLOG-2 file through either MPE or AIX's PCT utility, you most likely would be interested in the profiling information of the user-defined states which enclose MPI states and other user-defined states. In this case, inclusion ratio will be very useful. Because inclusion ratios of user-defined states usually dominate all state inclusion ratios, including those of MPI states. Therefore, the inclusion ratio highlights the outermost enclosing states even at high preview level. On the other hand, if you are a MPI implementor or are interested in the low level MPI networking overhead, you are most likely interested in the profiling information of MPI and its internal calls. Exclusion ratio will come in handy. Exclusion ratios for the innermost nested states, i.e. MPI states, tend to dominate all state exclusion ratios. So the exclusion ratio highlights the innermost nested states at very high preview level.

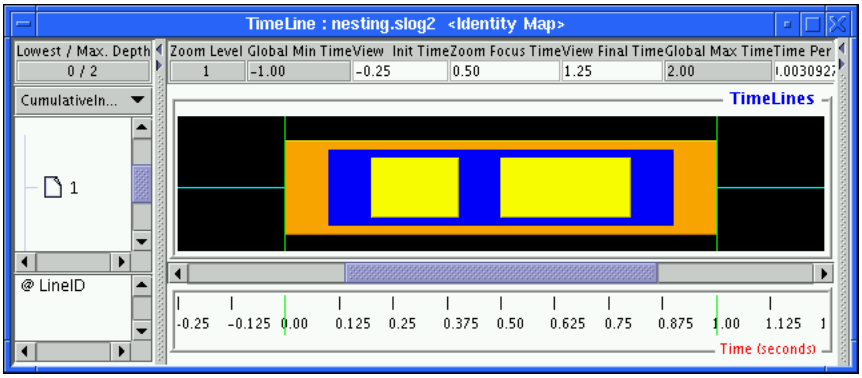


Figure 2.6: A zoomed in view of some nested states where the duration of the orange state is 1.0 sec. The duration of the navy blue state is 0.8 sec. The sum of durations for the 2 yellow states is 0.5 sec.

Figure 2.6 shows a typical zoomed in view of some nested states. In this view, the yellow states are deeply nested in the navy blue state which is in turns nested in the orange state. The pair of green lines mark the region where a preview state is being created for.

The inclusion and exclusion ratios are computed for the region marked by the pair of green lines and are shown in Table 2.2. The table shows that the most dominant state among all inclusion ratios

Icon	Description	Duration	Inclusion Ratio	Exclusion Ratio
	Innermost Nested State	0.5 sec	50%	50%
	Intermediate Nested State	0.8 sec	80%	30%
	Outermost Enclosing State	1.0 sec	100%	20%

Table 2.2: The breakdown of real states' contribution to a preview state of duration 1.0 sec as it is marked by a pair of green lines in Figure 2.6.

is the orange outermost state, but the most dominate state among all exclusion ratios is the yellow innermost state which is the least dominant state in inclusion ratios. One obvious observation is that the inclusion and exclusion ratios of the innermost state category are the same.

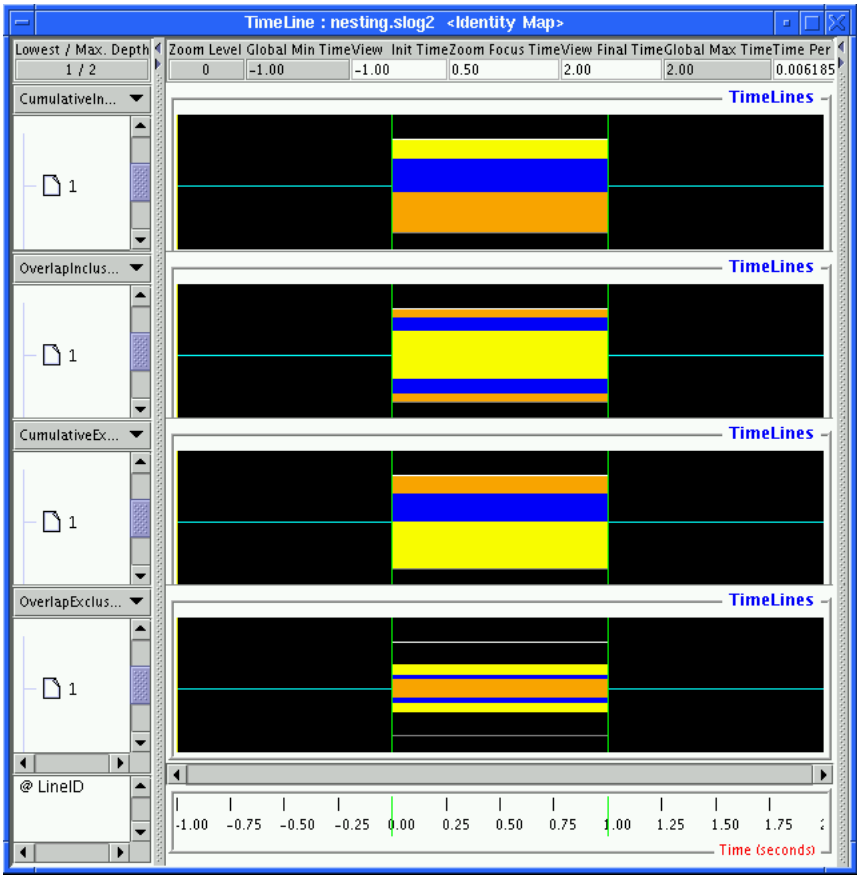


Figure 2.7: Different preview state displays of the zoomed in view of the Figure 2.6. Starting from the top, the first one is *CumulativeInclusionRatio* view, the second one is *OverlapInclusionRatio* view, the third one is *CumulativeExclusionRatio* view, and the last one is *OverlapExclusionRatio* view.

With data computed in Table 2.2, various different preview displays can be drawn and are shown in

Figure 2.7. All colored strips inside the preview state will be drawn proportional to the height of the preview state. For instance, if the ratio of the category for the strip is 0.9, the corresponding colored strip will occupy 90% of the preview state's height. The statement is true for all preview state display except *CumulativeInclusionRatio* which could have its total sum of ratios in exceed of 1.0 especially when the slog2 file is highly nested. First consider the *CumulativeInclusionRatio* and *CumulativeExclusionRatio* views, i.e the first and the third ones from the top in the figure. Notice that yellow state is least important in the top *CumulativeInclusionRatio* view, but becomes most significant in the third *CumulativeExclusionRatio* view. Since the sum of all inclusion ratios is larger than 1, in this case, the sum is 2.3, the *CumulativeInclusionRatio* view reweights all ratios to fill up the preview box. Strictly speaking *CumulativeInclusionRatio* view cannot be used to compare different preview states because of the arbitrary rescaling[6]. If one is interested in the comparison of inclusion ratios across different preview states, *OverlapInclusionRatio* view can be used instead. *OverlapInclusionRatio* view draws all inclusion ratios proportional to the height of the preview state but in an overlapping way, i.e. draw them in decreasing inclusion ratios order and stack one on top of the other, sort of like nested state. The overlap view of exclusion ratios is *OverlapExclusionRatio* view which is shown at the bottom of Figure 2.7. *OverlapExclusionRatio* view draws exclusion ratios exactly the same way as *OverlapInclusionRatio*. In general, overlap view cannot fill up the full height of the preview state. This is apparent in *OverlapExclusionRatio* view in 2.7 where the white bordered box indicates the full height of the preview state. The white bordered box is necessary in comparing the ratios across different preview states with respect to the preview states' duration. However, the white bordered box can sometimes be confusing, because whatever in the back of the preview state can show through the empty space within the white bordered box. In that case, the bordered box can be turned off by selecting *Empty* in PREVIEW_STATE_BORDER in Preference window.

For the sake of comparison and continuity with our preview discussion, the *CumulativeExclusionRatio* view of Figures 2.1 and 2.2 are shown in Figures 2.8 and 2.9 respectively. The *CumulativeExclusion-Ratio* view does provide an extra dimension of information when compared to its inclusion ratio counterpart at the expense of being a bit more complicated to digest visually.

[6]Most of times, neighoring preview states in *CumulativeInclusionRatio* view has similar total sum of inclusion ratios. Because of this fact, one can compare adjacent preview states. But bear in mind that the total sum of inclusion ratios between nearby preview states can change drastically without any visual indication. When in doubt, right click on the preview state to get Drawable Info Box to confirm the ratios.

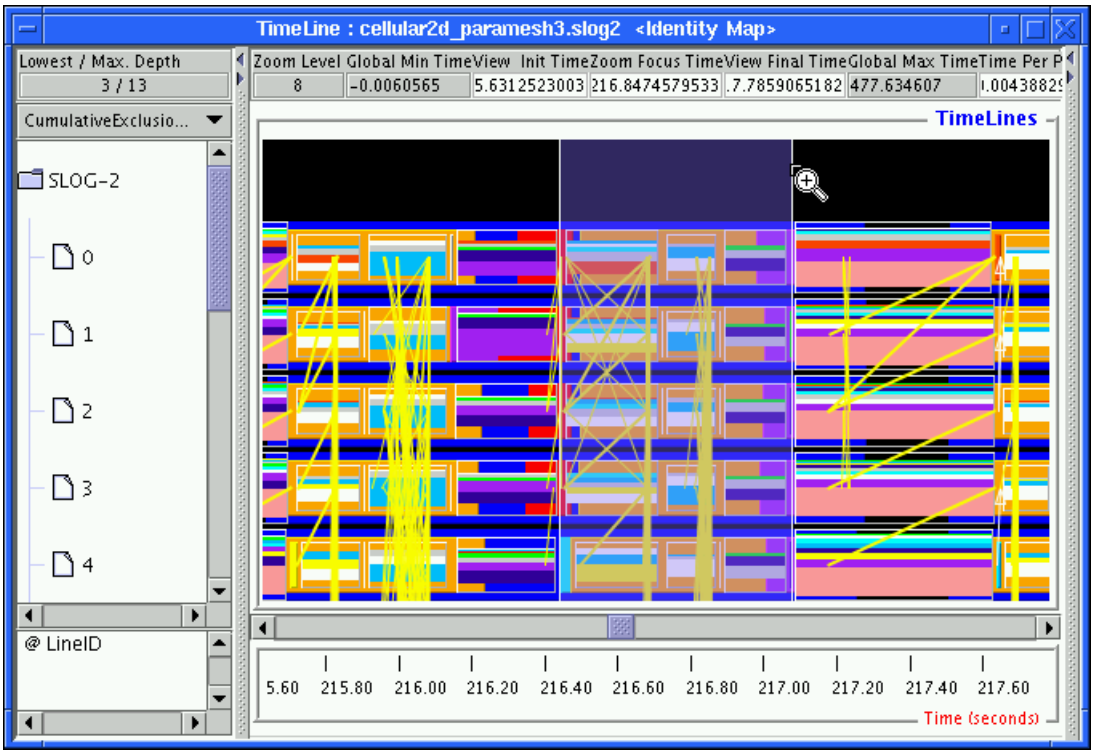


Figure 2.8: The *CumulativeExclusionRatio* view of Figure 2.1.

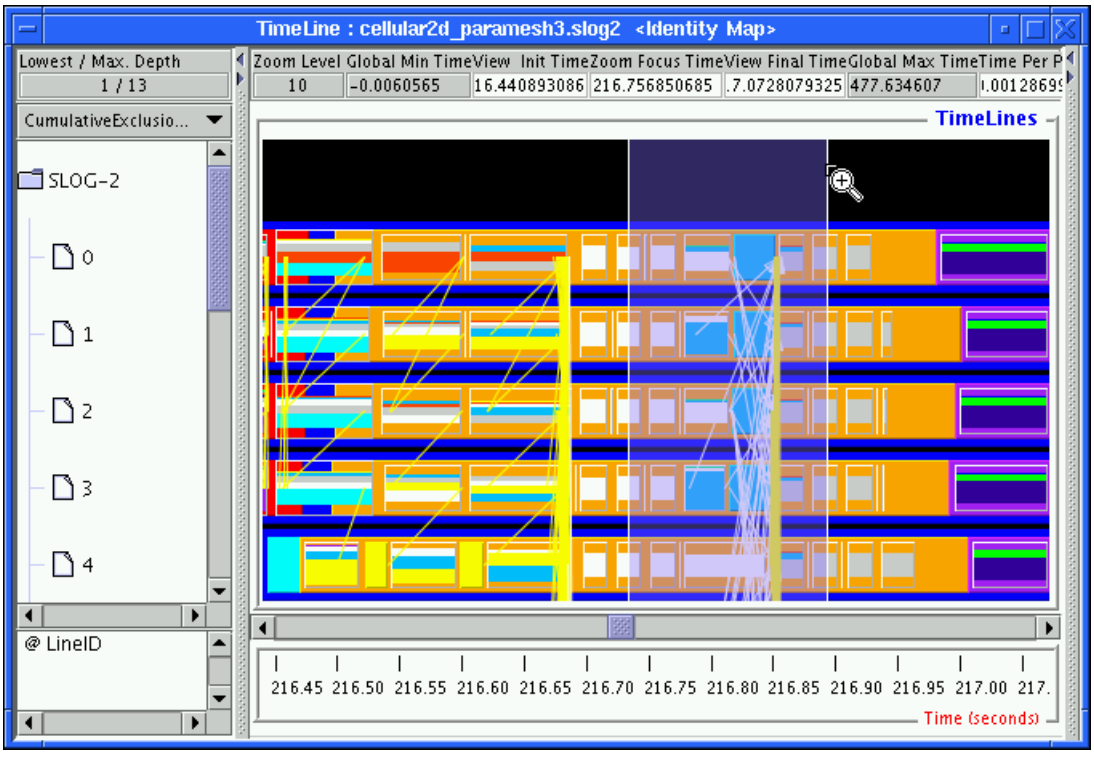


Figure 2.9: The *CumulativeExclusionRatio* view of Figure 2.2. Also, a zoom-in shot of Figure 2.8.

Chapter 3

Graphical User Interface

3.1 Main Window

Figure 3.1: The main control window of Jumpshot-4.

The first window that pops up when invoking Jumpshot-4 is called Main window as shown in Figure 3.1. The buttons shown in toolbar are shortcuts to the sub menu items in the top menubar. The function of each of these buttons is listed in the Table 3.2. There are 2 text fields that display crucial information about the logfile being processed. The text field which is titled *LogName* displays the pathname of the logfile being processed. The pulldown menu which is titled *ViewMap* lists all the available ViewMaps in the SLOG-2 file. Currently, both CLOG[1] and RLOG[2] converted SLOG-2 file contains one ViewMaps, it is called the Identity Map. Only IBM's UTE trace converted SLOG-2 file contains multiple ViewMaps.

3.2 Logfile Convertor Window

If a non-slog2 file is selected in the Main window, the Logfile Convertor as shown in Figure 3.2 will be invoked to prompt user to convert the file to SLOG-2 format readable by this viewer. There are

[1] a low-overhead native trace format from MPE.
[2] an internal MPICH2 profiling format

Icon	Description	Function
	File Selection	display a File Chooser dialog to select logfile to be processed
	Logfile Conversion	invoke the Logfile Convertor to convert non-slog2 file to slog2 format
	Show Legend Window	display the Legend window of the selected logfile if it is hidden
	Show Timeline Window	display the Timeline window of the selected logfile if it is hidden
	Edit Preferences	display the Preference window that adjusts Jumpshot's properties
	Show User's Manual	show the User's Manual of this program
	Show FAQs	show the FAQs of this program

Table 3.2: Functions of the toolbar buttons

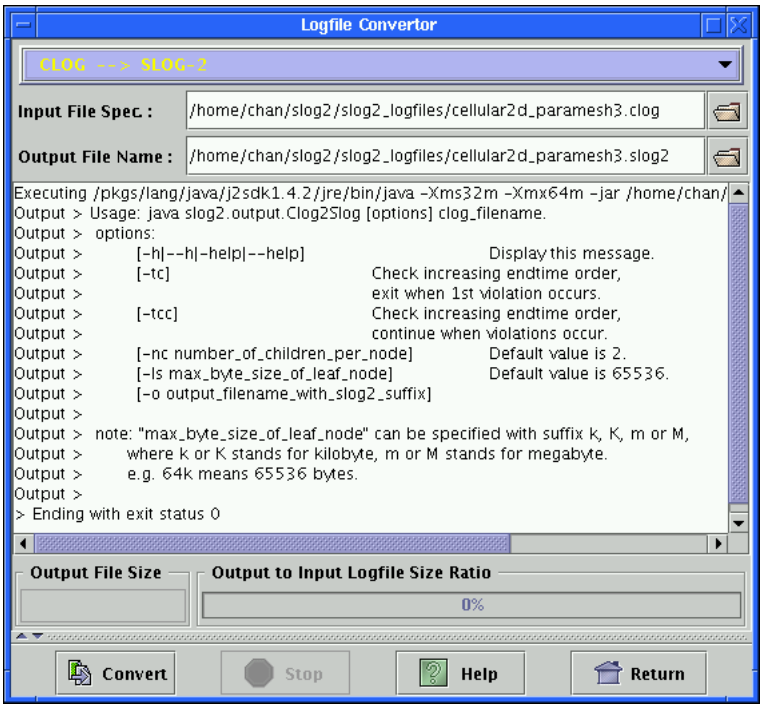


Figure 3.2: The Logfile Convertor window that allows conversion of supported trace file format to SLOG-2 format.

currently 3 supported convertors: *CLOG –> SLOG-2*, *RLOG –> SLOG-2* and *UTE –> SLOG-2*. Convertor is generally selected based on the input file's file extension. In the case that wrong file convertor is selected, user can correct it through the pale blue pulldown menu located at the top of the window. The Logfile Convertor window can also be invoked by directly clicking on the Logfile Conversion button shown in the Table 3.2. The text field of the Output File Name usually displays the default slog2 filename recommended by the convertor based on the text field in the Input File Specification. If the text field does not display the default name as expected, hitting return key in the Input File Specification field will force the update of the Output File Name field with the default name. There are 4 major functions of the Logfile convertor and each of them is associated with a button in the lower panel of the window. They are listed in the Table 3.4.

Icon	Description	Function
	Start Conversion	Start the logfile conversion of the selected convertor
	Stop Conversion	Stop the ongoing logfile conversion of the selected convertor
	Usage of Convertor	Print the usage information of the selected convertor
	Return Home	Return to the previous component that spawns the Logfile Convertor

Table 3.4: Functions of the major functions in the Logfile Convertor window.

Since the Logfile Convertor launches a separate java process to do the logfile conversion, it requires certain parameters to launch the process correctly. All the parameters that are needed by any logfile convertor are supplied through a panel hidden by a splitter in the convertor window. The splitter has a divider which can be lifted up to display all the parameters used to launch the java process as in the Figure 3.4. In the rare occasion that the default parameters are not correct, the text fields can be modified to reflect the situation.

The standard output and error streams of the process are being piped to the text area located in the middle of the window as the process is running. The Output File Size field displays the current size of the slog2 file as it is being generated, also the progress bar will be incremented to show the current ratio of the output to input file size as in the Figure 3.4. If the logfile conversion fails, the error message will be printed in the text area for diagnosis or bug report.

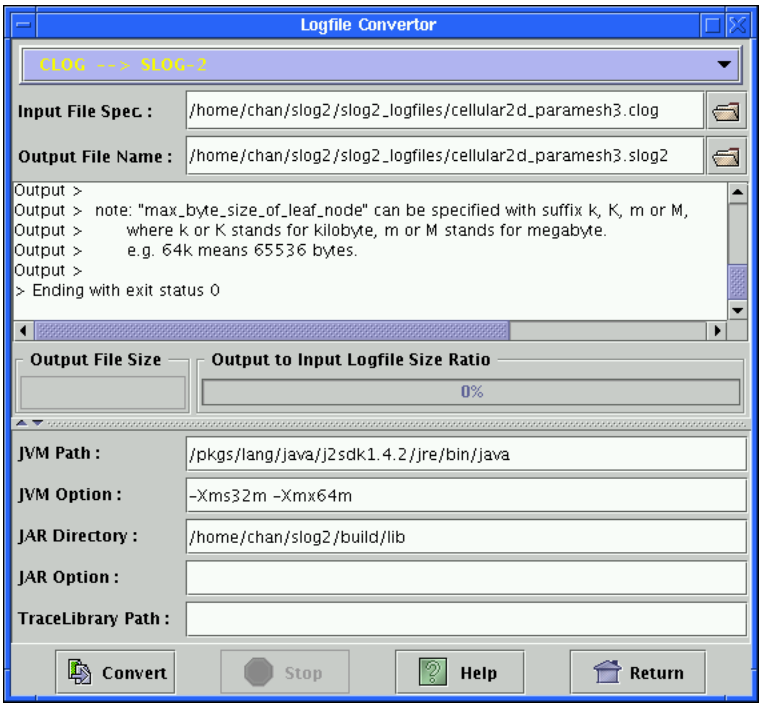


Figure 3.3: The hidden parameters panel of the Logfile Convertor.

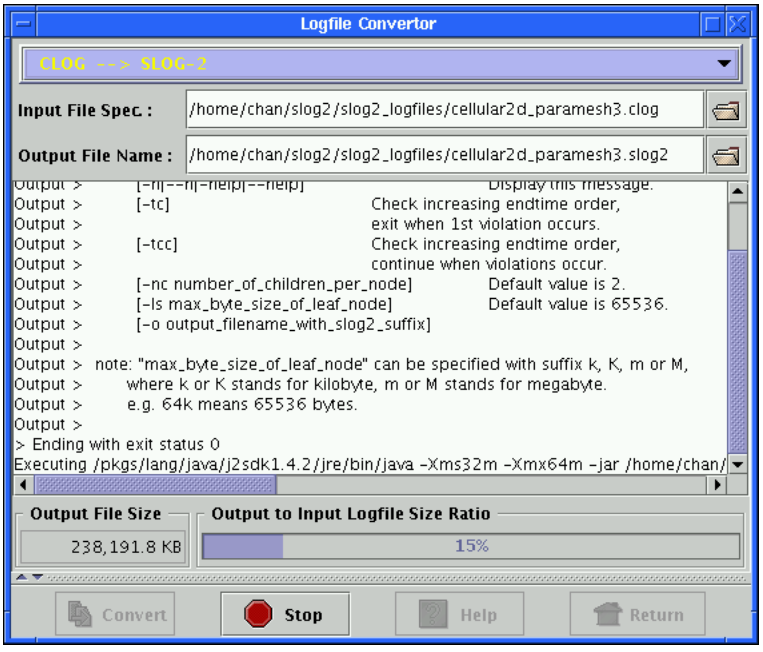


Figure 3.4: Logfile conversion in progress.

3.3 Legend Window

As soon as a SLOG-2 file is selected in Main window and ready for visualization, the Legend window like the one shown in Figure 3.5 will be displayed. All the features that are going to be discussed in the Legend window affects both the Timeline and Histogram windows.

The Legend window contains mainly a 4-columns legend table. The 4 columns are labeled as *Topo, Name, V* and *S* as in Table 3.6.

Icon	Description	Left Mouse Click on Column Cell	Right Mouse Click on Column Cell or Left Mouse Click on Column Title
Topo	Topology	Pick new Color (Figure 3.6)	None
Name	Name	Edit Name	Sort Order menu (Figure 3.7)
V	Visibility	Check or Uncheck	Checkbox Operations Menu (Figure 3.8)
S	Searchability	Check or Uncheck	Checkbox Operations Menu (Figure 3.8)

Table 3.6: Operations on the Legend window's columns.

Table 3.6 also lists out all defined mouse operations that are provided in each column. The operations are (1) left mouse clicking on the column title icon and on the column cell as well as (2) right mouse clicking in any column cell.

Figure 3.6 is the Color Chooser dialog that will pop up when one of the icon buttons in column Topo is pressed. The color editor provides 3 different ways of choosing a new color. After selecting a new color from the dialog, the new color will be used to update the icon button. The update won't be carried out in the timeline canvas automatically, explicit screen redraw is needed.

Figure 3.7 shows the popup dialog box either when the title icon of column Name is pressed or when right mouse button is clicked somewhere in the column. There are altogether 6 different sort orders. The first 4 orderings are various combination of alphabetical and case sensitive order, e.g. *z...a Z...A* refers reverse case sensitive alphabetical ordering. The second last order in the list is called *Creation Order* which refers to the order in which categories are stored in slog2 file when they are being created. The 4 alphabetical ordering has 2 hidden sort orders. One is called *Preview Order* which puts the preview drawable category before all the real drawable categories of the same topology. The other is *Topo Order* which refers to topological ordering, i.e arrow is ahead of state. The preview and topo sort orders can be turned on or off through the Preference window as shown in Table 3.22.

Figure 3.8 shows a popup dialog box when the title icon of column V (Visibility) or S (Searchability) is pressed or when right mouse button is clicked somewhere in either columns. The rule of selection in the legend table follows the standard practice of other graphical user interfaces as in the Table 3.10. Together with this standard selection rules, the operations provided in checkbox operation menu

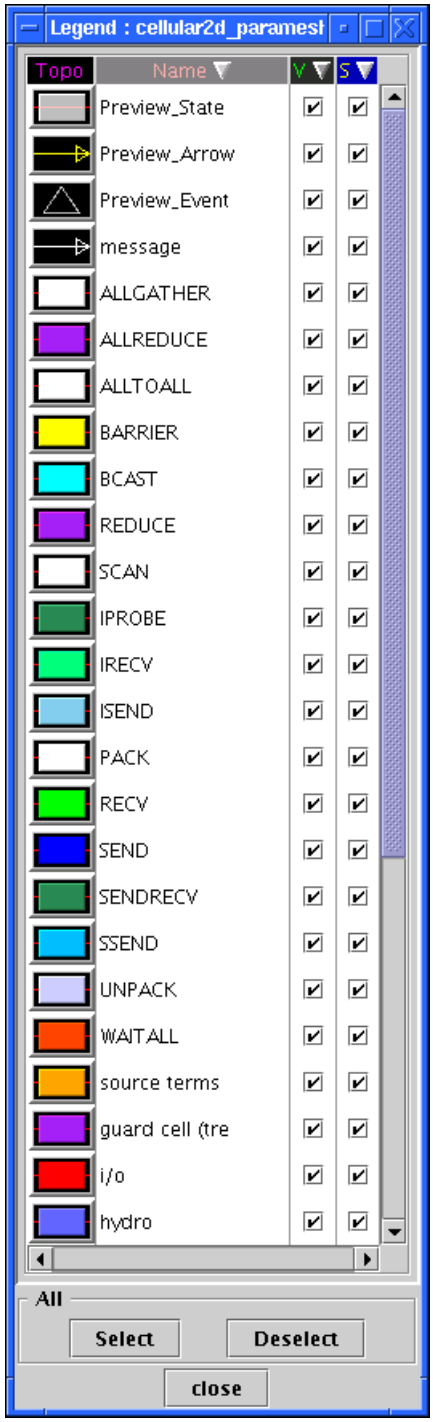


Figure 3.5: A typical Legend window when slog2 file is first loaded into Jumpshot-4.

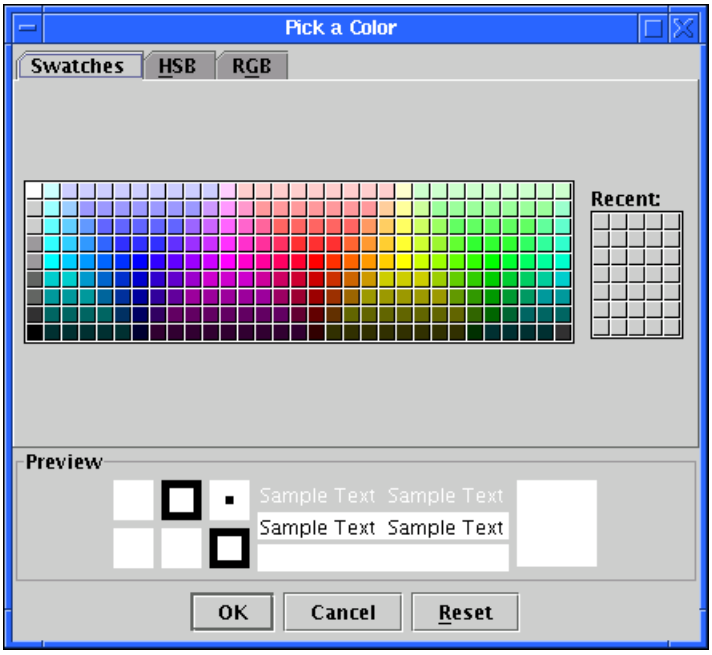


Figure 3.6: Color Chooser Dialog for column Category Topology

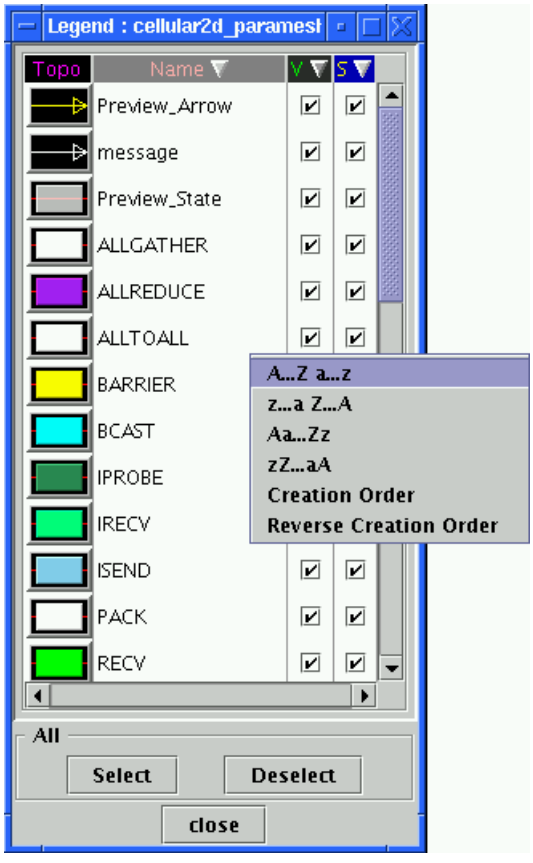


Figure 3.7: Sort Order operation menu for the column Category Name in the Legend window.

Ordering	Description
A...Z a...z	case sensitive alphabetical ordering
z...a Z...A	reverse case sensitive alphabetical ordering
Aa...Zz	case insensitive alphabetical ordering
zZ...aA	reverse case insensitive alphabetical ordering
Creation	category storage ordering in the slog2 file
Reverse Creation	reverse of Creation order

Table 3.8: The Description of the Sort Order operation menu in the Legend window.

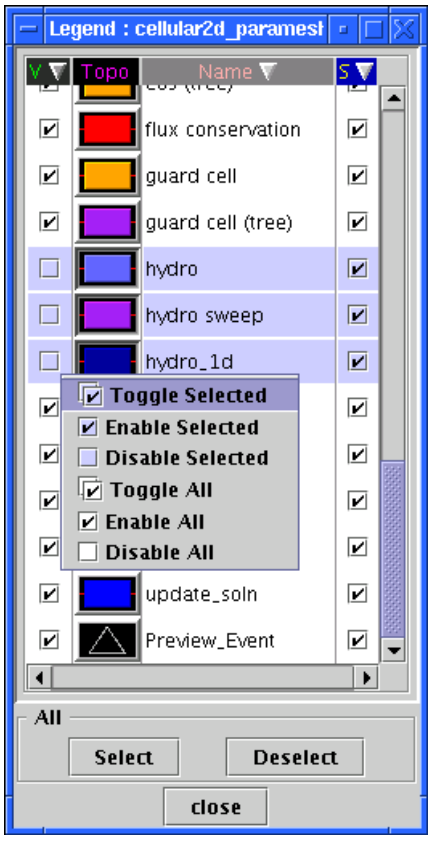


Figure 3.8: Checkbox Operation menu for column Category Visibility and Searchability

allow easy enabling and disabling of visibility as well as searchability checkboxes. With the help of continuous selection of the category rows in the legend table and various sort ordering available, users can easily make a huge number of categories disappear in the Timeline or Histogram window. For instance, in CLOG converted SLOG-2 file where upper case name always refers to MPI names, the case sensitive alphabetical ordering allows all MPI names to be put before all user-defined categories. With continuous mouse selection, user can easily toggle the visibility of user-defined states in the Timeline or Histogram window. Also, every element in the column Name is editable. This allows user to correct undesirable category names set during logfile creation or even facilitate sorting of the names for selection purposes.

Left Mouse Operation	Action
CLICK	*Click* on an object deselects any existing selection and selects the object.
CONTROL-CLICK	*Control-click* on an object toggles its selection without affecting the selection of any other objects
SHIFT-CLICK	*Shift-click* on an object extends the selection from the most recently selected object to the current object.
DRAGGING	*Dragging* (that is, moving the mouse while holding down left mouse button) through a range of TEXT deselects any existing selection and selects the range of text.

Table 3.10: Standard Selection Rules.

NOTE: Any changes done in the Legend window that alters the appearance of drawables won't be automatically updated in the timeline canvas until the CanvasReDraw button in the Timeline window is pressed.

3.4 Timeline *Zoomable* Window

Most of the advanced features in the SLOG-2 viewer are provided through the *Zoomable* window. There are two zoomable windows in Jumpshot-4: Timeline and Histogram windows. Figure 3.9 is the initial display of the Timeline window of a half gigabyte 16 timelines slog2 file. Zoomable window consists of several concealable and removable components. In the center of the window, it is the *zoomable and scrollable canvas*. For Timeline window, the center canvas is called *timeline canvas*. Directly on top of the zoomable canvas is the *time display panel*. On top of the display panel, there is the removable *toolbar*. To the left of the canvas is the concealable *Y-axis label panel*. To the right of the canvas is the concealable *row adjustment panel*. At the bottom of the canvas is the *time ruler canvas*. Both Y-axis label and the row adjustment panels can be put out of sight by clicking the tabs in the dividers or dragging the dividers to the side of the window. The top toolbar can be dragged out of the window or be repositioned in the other 3 sides of the window. A bare minimal zoomable window can be obtained by the removal of toolbar and the hiding of the left and right panels. An almost bare minimal Timeline window looks like the one shown in Figure 2.1.

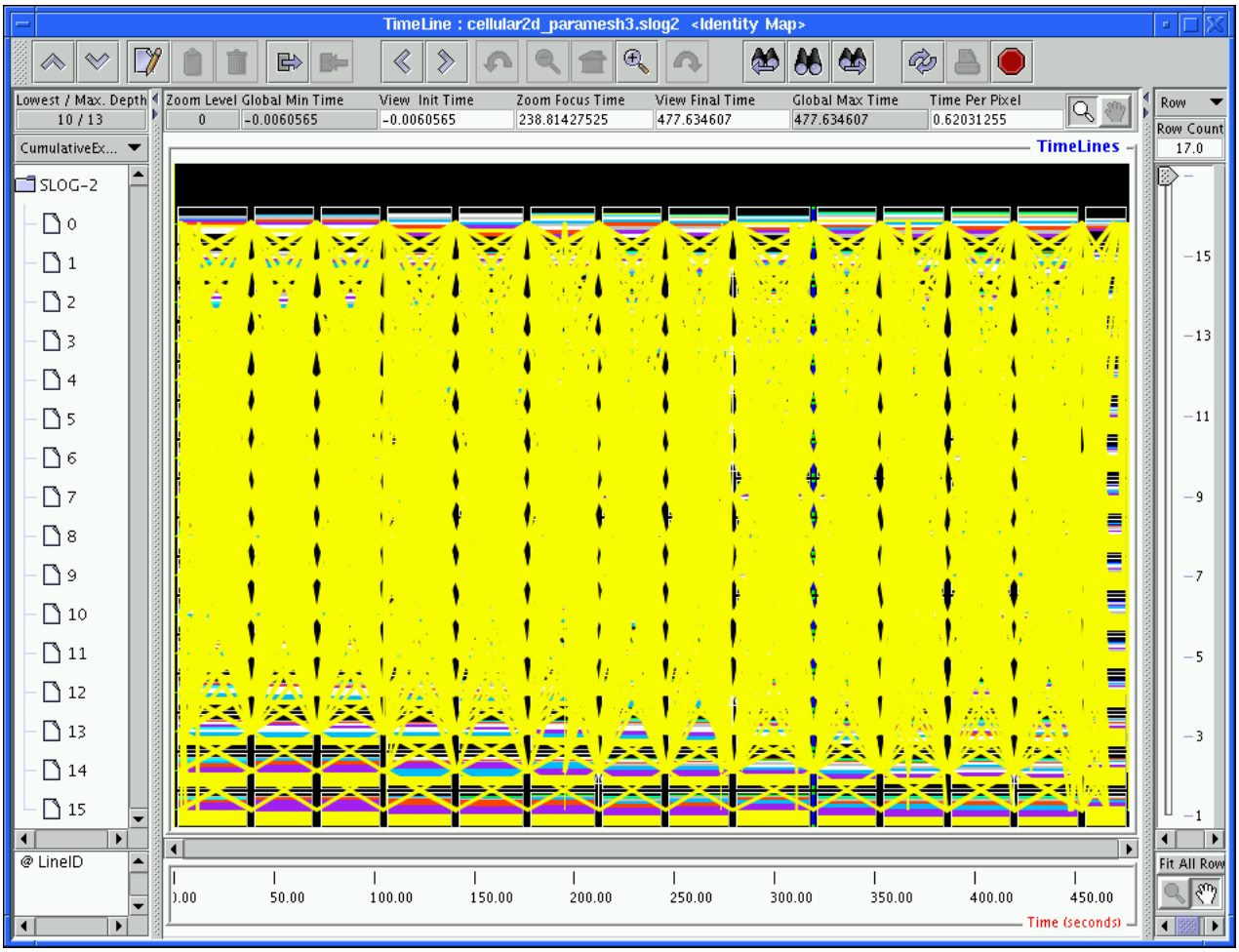


Figure 3.9: The initial display of the Timeline window of a 514 MB 16 processes slog2 file with default preview resolution.

3.4.1 Zoomable and Scrollable Canvas

When viewing a big slog2 file like the one shown in Figure 3.9, the whole timeline canvas is filled up with preview drawables. Though it provides a reasonable description at high level[3], it is pretty obscure to know the details. Hence, a well-designed zoomable and scrollable user interface (ZSUI) of the timeline canvas becomes an absolute necessity to facilitate the location of events of interest. The ZSUI of the timeline canvas includes many parts and operations. But the most handy ones are *dragged zoom*, *grasp and scroll* and *instant zoom in and out.* All these features are supported by the *Zoomable and Scrollable canvas*. There are 2 such canvases in the Timeline window. They are *Timeline Canvas* and *Time Ruler Canvas*. In these canvases, left mouse clicking can be alternated in 2 different modes by a pair of toggled buttons as shown in Figures 3.10 and 3.11. They are called *Zoom* and *Hand* modes. Each canvas in the Timeline window has its own set of toggled buttons that determine its left mouse click behavior. The timeline canvas's toggled buttons are located above the canvas and sit at the end of the time display panel. The time ruler's toggled buttons are located at the bottom of row adjustment panel, i.e. sit right next to the end of the ruler. By default, the timeline canvas is in zoom mode and the time ruler canvas is in hand mode, so user can do zooming when the cursor is in the timeline canvas and can scroll easily by simply moving the cursor over the ruler canvas. Also, the scrolling can be done by simply dragging on scrollbar's knob, clicking the end buttons and in the space between the knob and scrollbar's end buttons.

Figure 3.10: Canvas's left mouse click is in zoom mode.

Figure 3.11: Canvas's left mouse click is in hand mode.

3.4.1.1 Dragged Zoom

Figure 3.12: Zoom-plus cursor that indicates the left mouse clicking is ready for zooming in.

Dragged zoom is active only when the left mouse click is in zoom mode, i.e. when the the magnifying glass button is pressed in the toggled buttons as in Figure 3.10. In zoom mode, the cursor within the canvas will appear like a magnifying glass with plus sign in the center as in Figure 3.12. It is called

[3]Reasonable description here means that user can still get a vague sense of where the long and/or frequent drawables are .

zoom-plus cursor. The dragged zoom operation is initialized by pressing the left mouse button at the beginning of the zoom-in region, a white line will then appear. As soon as dragging is detected, another white line will appear to mark the current ending of the zoom-in region. The region that is marked by pair of white lines is lightly shaded as shown in Figure 2.4. The process can be canceled anytime by hitting the ESC key during dragging. Once the left mouse button is released, zooming will be carried out and the Timeline window will then be updated as in Figure 2.5. The time display panel is updated with the latest time related information of the zoom-in region. Notice that the zooming as well as scrolling can be achieved by explicitly editing the text fields in the time display panel.

3.4.1.2 Instant Zoom

Figure 3.13: Zoom-minus cursor that indicates the left mouse clicking is ready for zooming out.

While the canvas is still in zoom mode, *instant zoom* is enabled by default. Instant zoom allows zooming in at the point of left mouse clicking by a factor of 1/2, i.e. the region centered at the point of left clicking will be magnified by a factor or 2. Also, the *Zoom Focus Time* in the time display panel will be updated with the time where left clicking on the canvas is detected. In the process, the cursor remains zoom-plus cursor. *Shift-click*, on the other hand, will do the opposite. While holding down Shift key, the cursor will be changed to a zoom-minus cursor as in Figure 3.13 to indicate zooming out is the action associated with left clicking. The zoom factor is 2 in this case.

3.4.1.3 Grasp and Scroll

Figure 3.14: Open hand cursor indicates that left mouse clicking is ready to grasp and scroll.

Figure 3.15: Close hand cursor indicates that left mouse clicking is scrolling.

Grasp and Scroll is active only when the left mouse click is in hand mode, i.e. when the open hand button is pressed as in Figure 3.11. The cursor in hand mode is an open hand as in Figure 3.14. As soon as left mouse button is pressed down, the cursor turns to a close hand as in Figure 3.15. It indicates the canvas will move in the same direction that the cursor moves as long as the left mouse button remain pressed. The grasp and scroll mode in time ruler canvas can only move horizontally, but the grasp and scroll mode in timeline canvas allows movement in both vertical and horizontal axes.

3.4.1.4 Information Dialog Box

Jumpshot-4 wouldn't be complete if it cannot provide a way to tell user what exactly are being displayed. It is particularly important when there are many preview drawables. Following standard user interface practice, Jumpshot-4 uses *right mouse clicking* as an interface for user to tell Jumpshot-4 what object that more information is needed. In general, anywhere on the canvas, both timeline and time ruler canvases, can be inquired with right mouse clicks. Information dialog box will pop up accordingly to tell user more about object that is being clicked. There are 3 different types of information dialogs: Drawable Info Box, Duration Info Box and Time Info Box. All these info boxes remain in memory as long as they are not closed even if the canvas has been scrolled or zoomed. One of the usages of the info boxes is to serve as time markers in between zooming and scrolling.

Drawable Info Box Drawable Info Box is a popup dialog box that provides detailed information about the drawable object that is being clicked. There are 2 different kind of Drawable Info Box, one for preview drawable, one for real drawable.

Drawable Info Box for Preview Drawable Right mouse clicking on 2 of the preview states in the timeline canvas shown in Figure 2.9 will pop up 2 Drawable Info Boxes for the preview states. They are displayed in Figure 3.16. The popup Info Box's upper left hand corner will be positioned at exactly where right mouse click is detected and a green line marker will appear on the canvas to indicate what time has been clicked in case the dialog box is moved from its original popup location. In order to best illustrate what information is presented by the Drawable Info Box, let's take the highlighted Drawable Info Box in Figure 3.16 as an example. The Drawable Info Box for preview state contains a pink label "Preview State", and the icon inside the dialog box shows the color and shape of the drawable. Below the icon, there is a big text area that prints all the detailed statistical information about this preview state. There are 6 timestamps in the text area: maximum duration, minimum starttime, maximum endtime, average duration, average starttime and average endtime. Here "[0]" refers to starting point, and "[1]" refers to the ending point. The 3 "average" timestamps are averaged over all the real drawables represented by this preview drawable. Besides timestamps, the info box also tells "Number of Real Drawables" represented by the preview object. In this case, 136 real states are amalgamated by the pure white preview state. Also, the text area lists all the categories of real drawables amalgamated and their ratios of the total duration of all real drawables to the duration of the preview states. In this case, there is only 1 category of real states in this preview state, so all 136 states are all PACKs. The sum of the durations of all PACKs is about half of the duration of the preview state as it is indicated by "ratio=0.5021433".

Another Drawable Info Box which is shown in Figure 3.16 has its upper left-hand pointed at a preview state that has 4 different strips of colors: yellow, royal blue, white and purple. Right mouse clicking at the yellow strip pops up a Drawable Info Box with a yellow state icon with label BARRIER. As shown in the figure, this preview state amalgamated 4 different categories of real states: ALLREDUCE, PACK, SSEND, and BARRIER, and the statistically most significant one is BARRIER. It proportionally and exclusively occupies 55% of the length of the preview state. Hence BARRIER strip has the tallest height among all the color strips shown in the preview state. Clicking

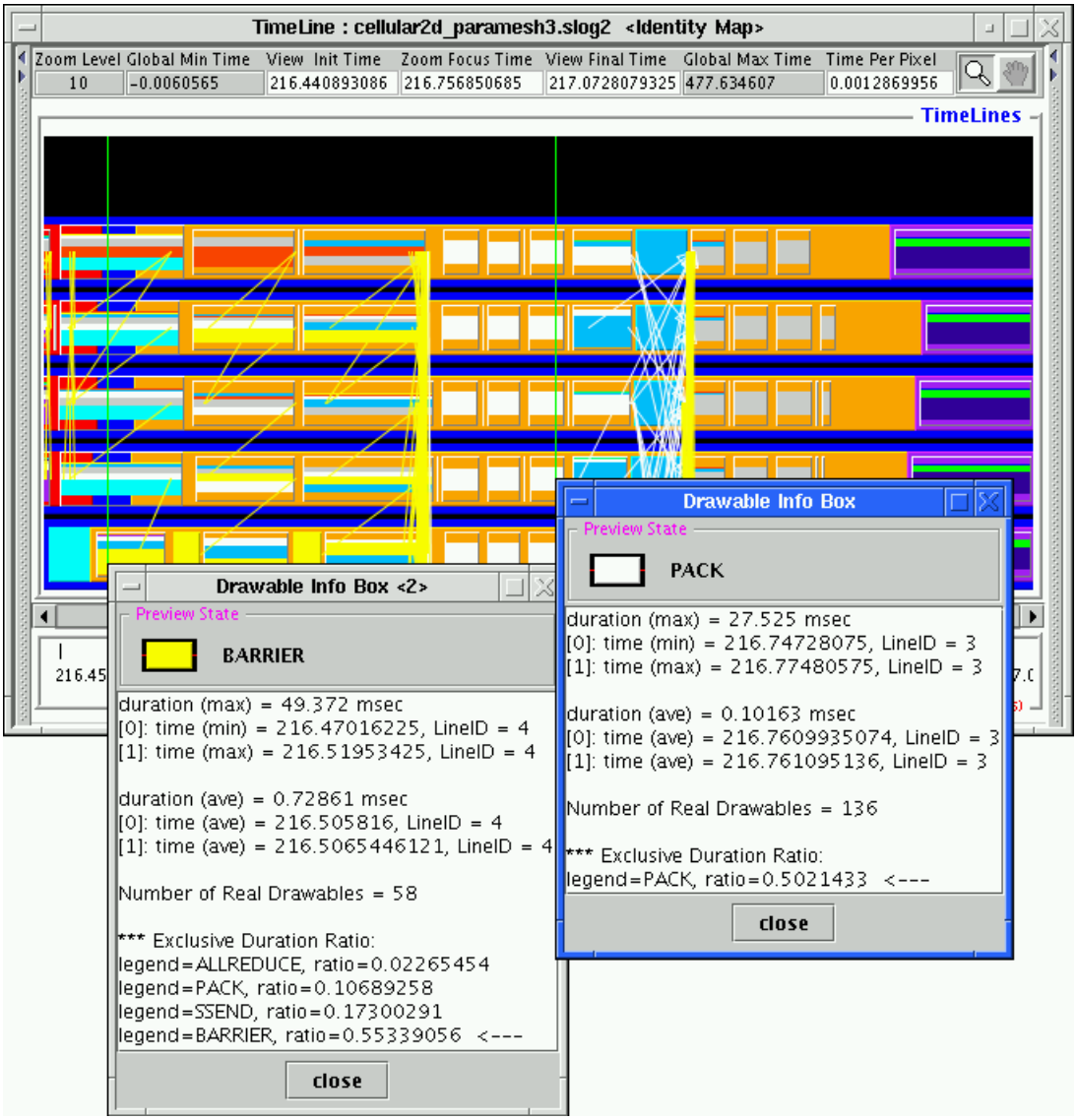


Figure 3.16: Drawable Info Box for Preview State

on the different color strip in the same preview state will pop up a Drawable Info Box with a different labeled icon, but the content of the text area remains the same. In general, not every category listed in the text area is visible in the preview state display. Out of the 4 categories mentioned in the text area, only 3 are visible noticeably in the figure given the limited pixel height available to the preview state. The least significant category ALLREDUCE is barely visible. But the limitation can be improved by selecting another display option for preview state in Preference window that does not rely on the category ratio[4]. As indicated, there are altogether 58 real drawables in the preview state, but no information is provided about how many real drawables are in each real category.

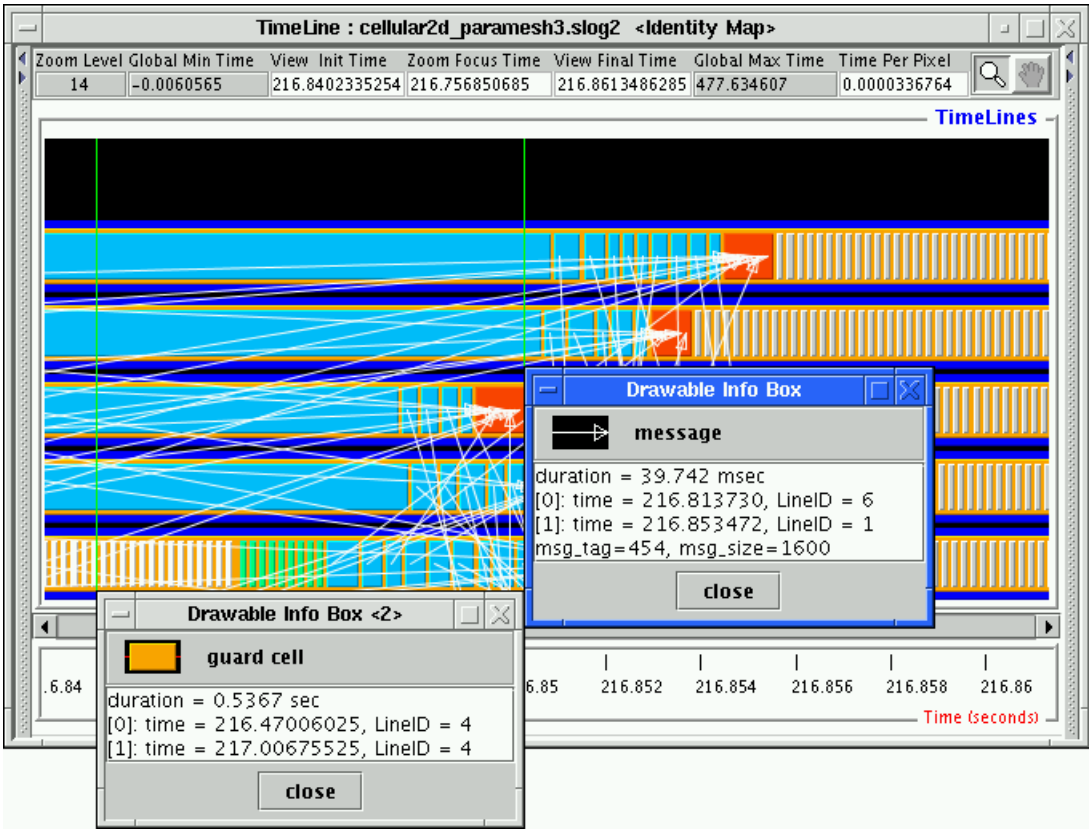


Figure 3.17: Drawable Info Box for real state and arrow. The Drawable Info Box for the arrow shows the message size, 1600 byte, and tag ID, 454.

Drawable Info Box for Real Drawable Similarly for real drawables, Drawable Info Box can be brought up by right mouse clicking on the real drawables. In Figure 3.17, Drawable Info Boxes for a real arrow and a real state are shown. The Drawable Info Box for the arrow is invoked by clicking anywhere within the vicinity of the arrow body[5], and the info box shows the starttime, start timeline

[4]i.e. by setting the PREVIEW_STATE_DISPLAY pulldown menu in Timeline window or Preference window to *FitMostLegends* message as listed in Table 3.18 .

[5]The vicinity width can be adjusted by modifying the parameter CLICK_RADIUS_TO_LINE in Preference window as listed in Table 3.16. The default is 3 pixels.

ID, endtime, and ending timeline ID and some extra information implemented by the native format. In this example, the message size carried by the specific arrow is 1600 byte.

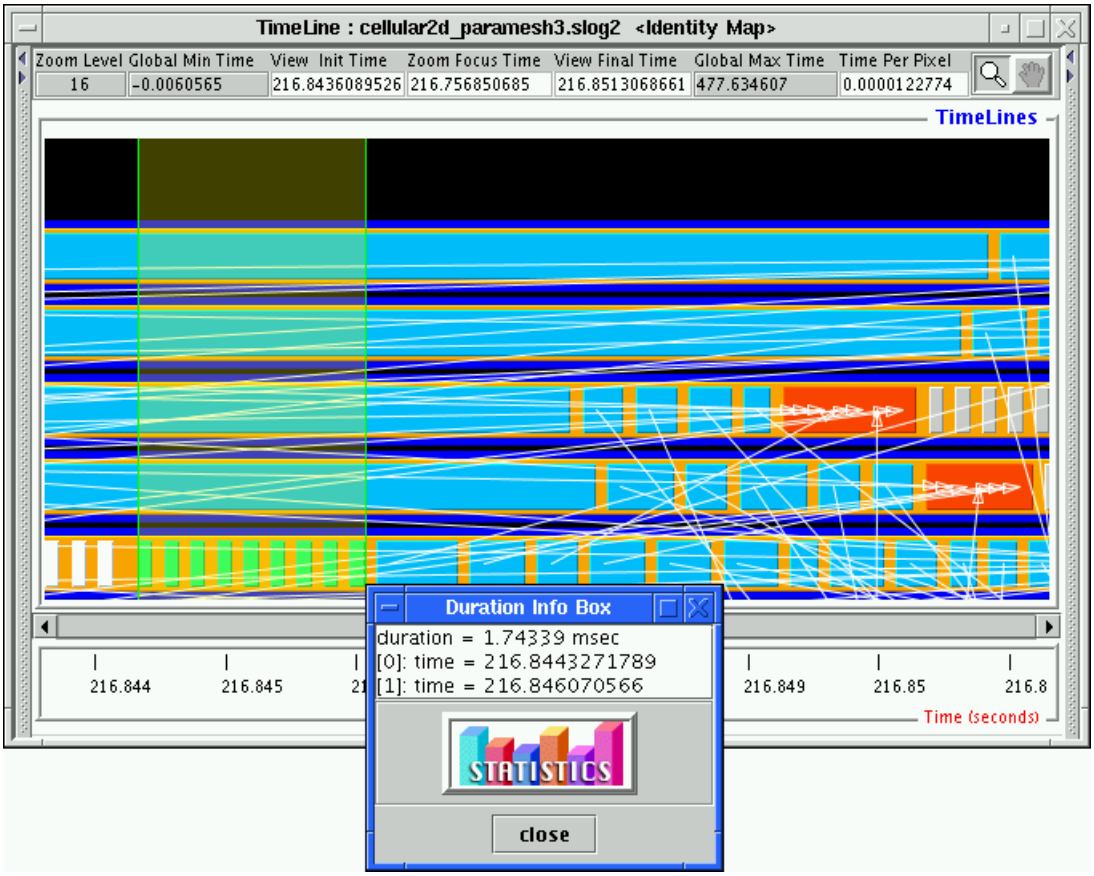


Figure 3.18: Duration Info Box shows the duration, starttime, and endtime of a time region marked by a pair of green lines.

Duration Info Box Duration Info Box is created by right dragging in the timeline canvas or the time ruler canvas to mark a region in time. The dragged region will be marked by a pair of green lines and is lightly shaded as well. Duration Info Box could serve a marker to facilitate the process of zooming in and out. The information provided by Duration Info Box could also be used to compare different durations or to measure the total duration of a collection of subroutine calls. For instance in Figure 3.18, the Duration Info Box marks all consecutive green states on the fifth timelines. The Duration Info Box says the total duration of the 9 green states is about 1.74 msec..

Time Info Box Time Info Box is created by right clicking in the empty space in either timeline or the time ruler canvas as in Figure 3.19. This Info Box is usually used as a marker for a single event in time.

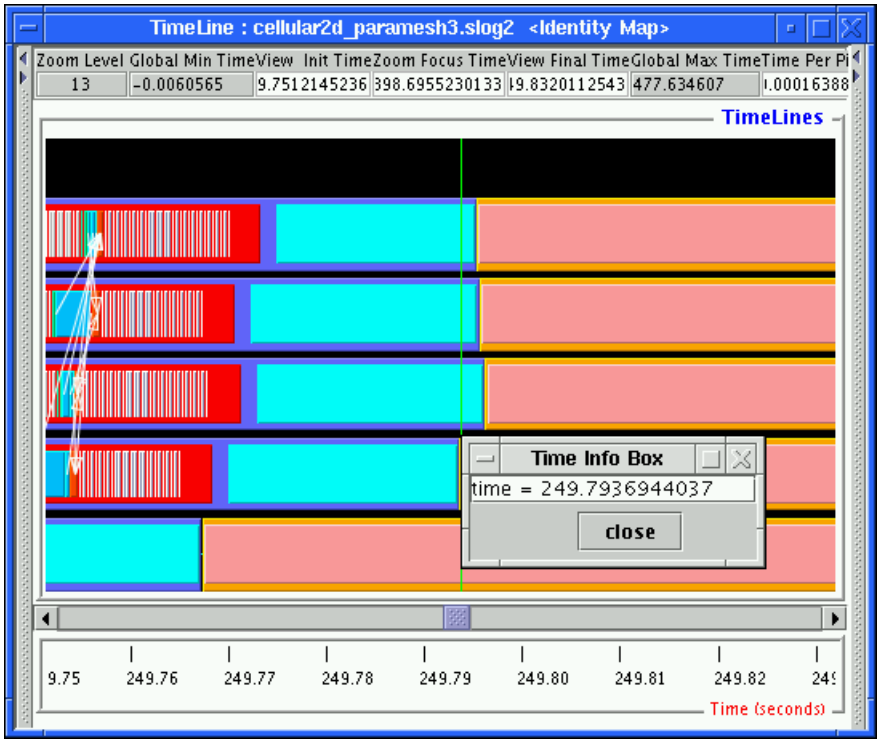


Figure 3.19: Time Info Box displays the time of where it pops up.

3.4.2 Toolbar

The buttons in the toolbar of Timeline window provides various basic services to the Timeline window. Table 3.12 contains the list of functionalities of the buttons found in the toolbar.

3.4.3 Y-axis Label Panel

The concealable left panel in Timeline window is called Y-axis label panel which contains a tree-like representation for Y-axis label for the timelines. For a SLOG-2 file convertible from CLOG or RLOG with the default viewmap, the typical Y axis label panel looks like that is shown in Figure 3.20. Together with toolbar's label buttons, e.g. LabelMark and LabelMove, and standard mouse selection methods listed in Table 3.10, labels can be rearranged easily to create a more easily understood timeline canvas. For multiple viewmaps SLOG-2 file from IBM's UTE trace environment, LabelExpand and LabelCollapse buttons will come in handy to expand and collapse the label tree by one whole level. In order to minimize unnecessary redraw of the timeline canvas, the synchronization between the label panel and the timeline canvas is carried out passively, i.e. user needs to press the CanvasReDraw button in the toolbar to update the Timeline window with the changes from the label panel.

Icon	Description	Shortcut	Function
	Up	Alt-UP	Scroll upward by half a screen
	Down	Alt-DOWN	Scroll downward by half of a screen
	LabelMark	none	Mark the timeline(s)
	LabelMove	none	Move the marked timeline(s)
	LabelDelete	none	Delete the marked timeline(s)
	LabelExpand	Alt-E	Expand the Y-axis tree label by 1 level
	LabelCollapse	Alt-C	Collapse the Y-axis tree label by 1 level
	Backward	Alt-LEFT	Scroll Backward by half a screen
	Forward	Alt-RIGHT	Scroll Forward by half a screen
	ZoomUndo	Alt-U	Undo the previous zoom operation
	ZoomOut	Alt-O	Zoom Out by 1 level in time
	ZoomHome	Alt-H	Reset zoom to the initial resolution in time
	ZoomIn	Alt-I	Zoom In by 1 level in time
	ZoomRedo	Alt-R	Redo the previous zoom operation
	SearchBackward	Alt-B	Search backward in time
	SeachInitialize	Alt-S	Search Initialization from last popup InfoBox's time
	SearchForward	Alt-F	Search forward in time
	CanvasReDraw	Alt-D	Redraw canvas to synchronize changes from Preference/Legend window or Y-axis label panel.
	Print	none	Print the Timeline window
	Exit	none	Exit the Timeline window

Table 3.12: Table of toolbar's functionalities.

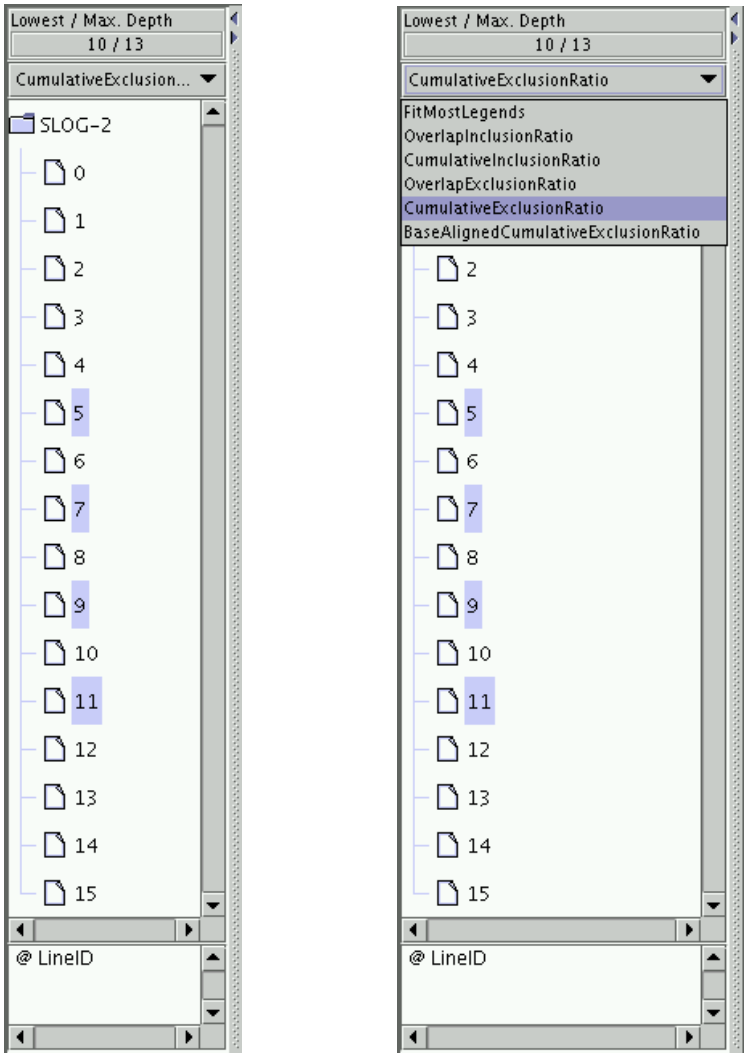


Figure 3.20: A simple 1 level Y-axis label tree. The blue highlighted labels are those that have been selected. The pulldown menu at the top of panel indicates the value in PREVIEW_STATE_DISPLAY in Preference window

3.4.4 Row Adjustment Panel

The concealable right panel in Timeline window contains the row adjustment panel which is used to determine the row adjustment scheme. There are 2 different modes in row adjustment panel: row count mode and row height mode. These 2 modes can be selected by the pulldown menu at the top of the panel. The row count mode attempts to keep the number of timelines constant as indicated in the Row Count text field when the Timeline window resizes. On the other hand, the row height mode fixes the height of each timeline as indicated by the Row Height text field. Currently, the height of the timeline can be adjusted up to the height of the timeline canvas, in that case the Row Count text field shows a number 1[6]. The maximum number of timelines that can be displayed is set to the total number of rows represented by the whole Y-axis label tree[7]. For multiple viewmaps slog2 file, the Y-axis label tree can be expanded or collapsed. This could change the maximum number of rows in the row count slider after user hits the CanvasReDraw button. Coupling with window resize, the row adjustment panel allows user to magnify or shrink the height of the timeline as one desires.

3.5 Histogram *Zoomable* Window

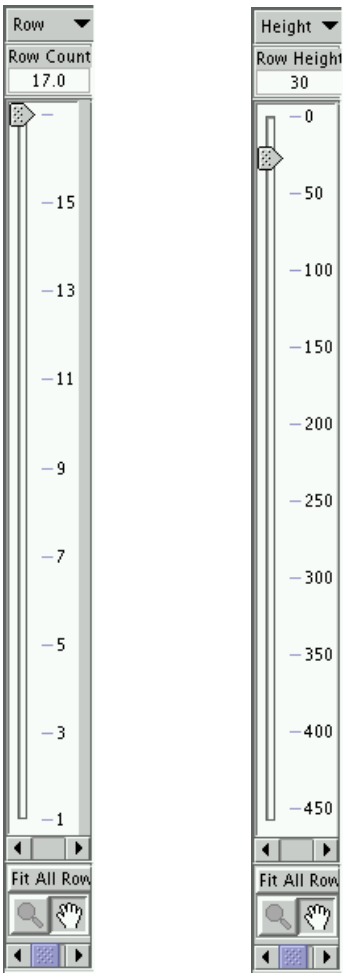
The Histogram window is created through clicking the statistics button located in the middle of Duration Info Box shown in Figure 3.18. In Figure 3.22, the histogram window is created for the whole duration of the timeline canvas in Figure 3.9, i.e same duration as the complet slog2 file. In general, the total duration of the histogram canvas is the same as the duration marked by the Duration Info Box so that the histogram window functions like a graphical display of statistical summary of the duration of interest. For instance, it is obvious from Figure 3.22 that the yellow state, it is MPI_Barrier in this case, cumulatively takes up the most time. This is especially true in the last timeline.

Since Histogram window is also a *zoomable* window like Timeline window, a lot of features like those described in section 3.4.1 for Timeline window are available for Histogram window as well, e.g. dragged-zoom, grasp and scroll, instant zoom in/out, easy vertical expansion of timeline, cut and paste of timelines. If some state categories or timelines need to be made invisible in histogram window, it can be achieved through disabling the corresponding categories in Legend window's column V or S or selected corresponding timelines in histogram window. It is just like that of Timeline window.

Only summary objects can be displayed in the histogram window. Summary object is similar to preview object discussed earlier. Preview objects are created during logfile creation stage and cannot be modified during visualization. Summary objects, on the other hand, are created dynamically during visualization, i.e. during creation of a Duration Info Box, so they can be modified easy by end users. There are 2 different kinds of summary objects: summary state and summary arrow.

[6]If the slog2 file contains numerous timelines, increasing the Row Height will increase the size of the images managed by Jumpshot-4. This may cause the Java Virtual Machine to exhaust all its memory if the virtual machine is not set to have enough memory when Jumpshot-4 is started or there isn't enough physical memory in machine that Jumpshot-4 runs on.

[7]Hence the row height cannot be adjusted all the way to zero.



(a) Row Count mode

(b) Row Height mode

Figure 3.21: Row Adjustment Panel determines the Timeline window's resize scheme. When one of the mode sliders or text fields is adjusted, the other 3 components will be adjusted simultaneously.

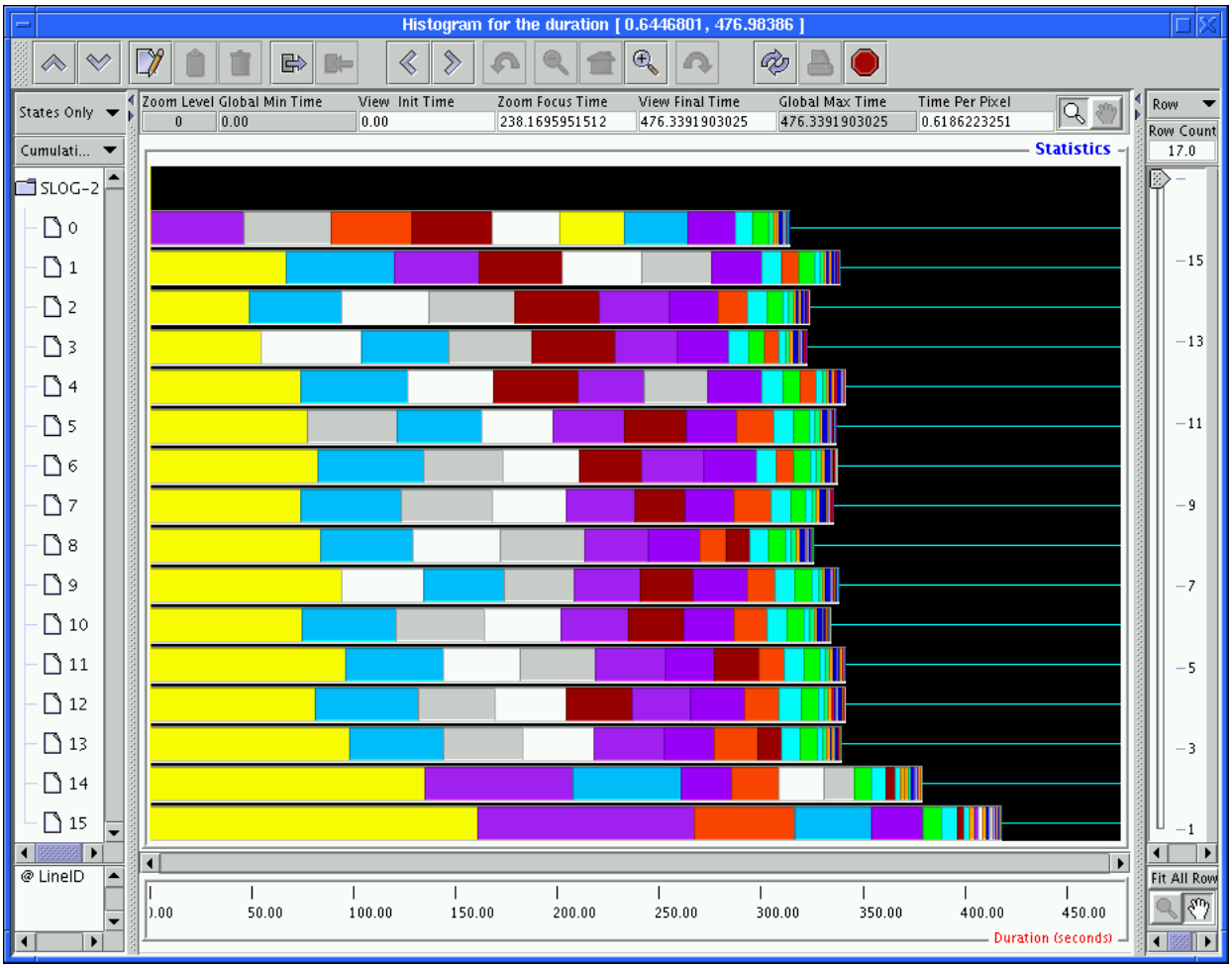


Figure 3.22: A histogram window of the whole duration shown in Figure 3.9.

There is only 1 summary state per timeline and 1 summary arrow for each ordered pair of timelines[8]. Currently 3 different views are available in Histogram window: *States Only*, *Arrows Only* and *All*. In *States Only* view, only summary states are displayed. In *Arrows Only* view, only summary arrows are displayed. In *All* view, both summary states and arrows are displayed.

3.5.1 Summary States

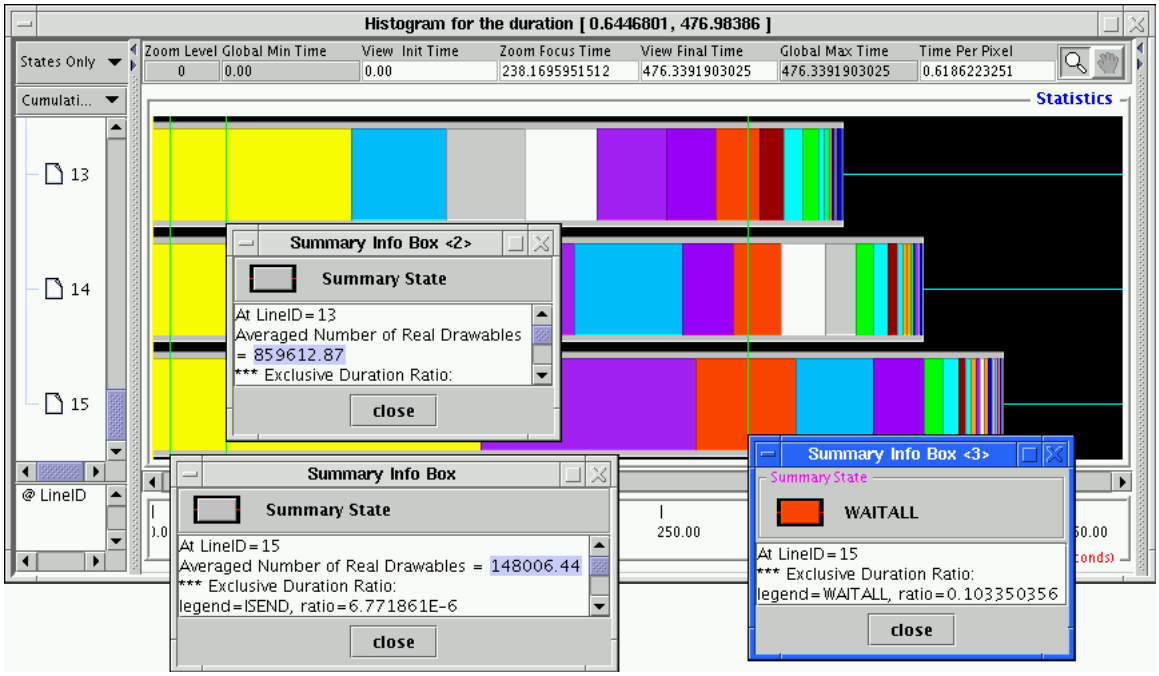


Figure 3.23: State Summary Info Boxes of the Histogram window.

Since summary states are created through the statistics of real and preview states, summary states inherit properties of preview states, i.e. inclusion and exclusion ratios. Because of this, different representations of summary state are formed based on the PREVIEW_STATE_DISPLAY discussed earlier in section 2.2.1. Different representation of summary state can be selected through SUMMARY_STATE_DISPLAY pulldown menu located at the top of the left panel in the histogram window or through a similar variable defined in Preference window and in Table 3.20. Figure 3.22 is actually a *CumulativeExclusionRatio* view. Since the most time consuming timeline is the last one, we will zoom in the last three timelines and use them to discuss the visual representation of summary state. Figure 3.23 shows the last three timelines of Figure 3.22. Each summary state has a gray bordered box. Right mouse clicking at the bordered box pops up the Summary Info Box for the whole summary state. The info box lists the total number of real states it contains and detailed information of what state categories it contains. In the figure, the summary info boxes at timeline 15 and 13 show that the timeline 15 summary state contains about 148006 real states and the timeline 13 summary state has about 859613 real states, i.e. timeline 13 has 5.8 times the number of real

[8]ordered pair of timelines means timeline pair (1,2) is different from pair (2,1).

states than that of timeline 15 within the same duration. Each summary state also displays the ratios of the total duration of each member state category to the duration of the canvas as colored boxes inside the gray bordered box. Right clicking at any of the colored boxes will display a summary info box that indicates the color and name of category and the corresponding ratio for the duration as the highlighted summary info box in the Figure 3.23.The remaining duration at the end of each timeline is unaccounted for. In this particular logfile, remaining time could be thought of being used for computation.

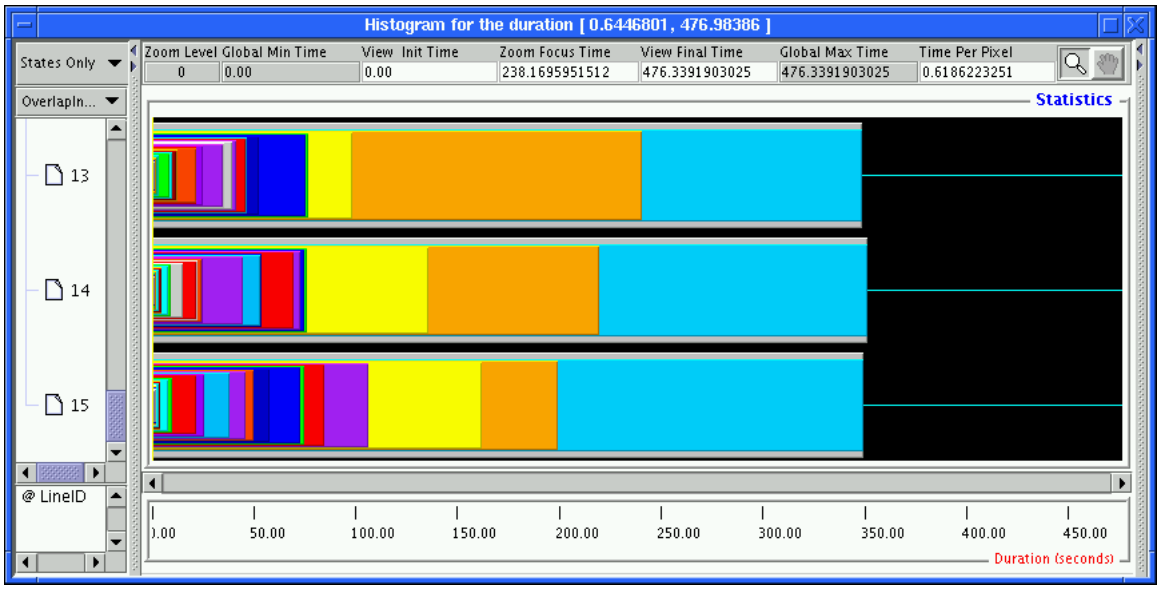


Figure 3.24: The *OverlapInclusionRatio* view of Figure 3.23.

Switching the SUMMARY_STATE_DISPLAY pulldown menu in histogram window in the figure to *OverlapInclusionRatio* redraws the histogram canvas. The histogram canvas now looks like one shown in Figure 3.24. Since the sum of all inclusion ratios is greater than 1.0, *CumulativeInclusionRatio* view is not provided in the histogram window[9]. All the member categories of the summary states in *OverlapInclusionRatio* view are drawn from the beginning of the histogram canvas and they are nested one inside others in decreasing inclusion ratio order, so the largest inclusion ratios are easily noticeable. In order to see the smallest ratios, one needs to zoom in around the beginning of the canvas. In Figure 3.24, the largest inclusion ratios in the 3 visible timelines are all royal blue and take up about the same amount of time. The 2nd largest ratios are all orange colored and is smallest in the timeline 15. Therefore *OverlapInclusionRatio* is good for comparison of member category contribution among different timelines.

3.5.2 Summary Arrows

Figure 3.25 is the *Arrows Only* view of the histogram window shown in Figure 3.22. There is a summary arrow per ordered pair of timelines. The duration of each summary arrow is the total

[9]The view cannot be drawn within same duration as marked in the timeline window.

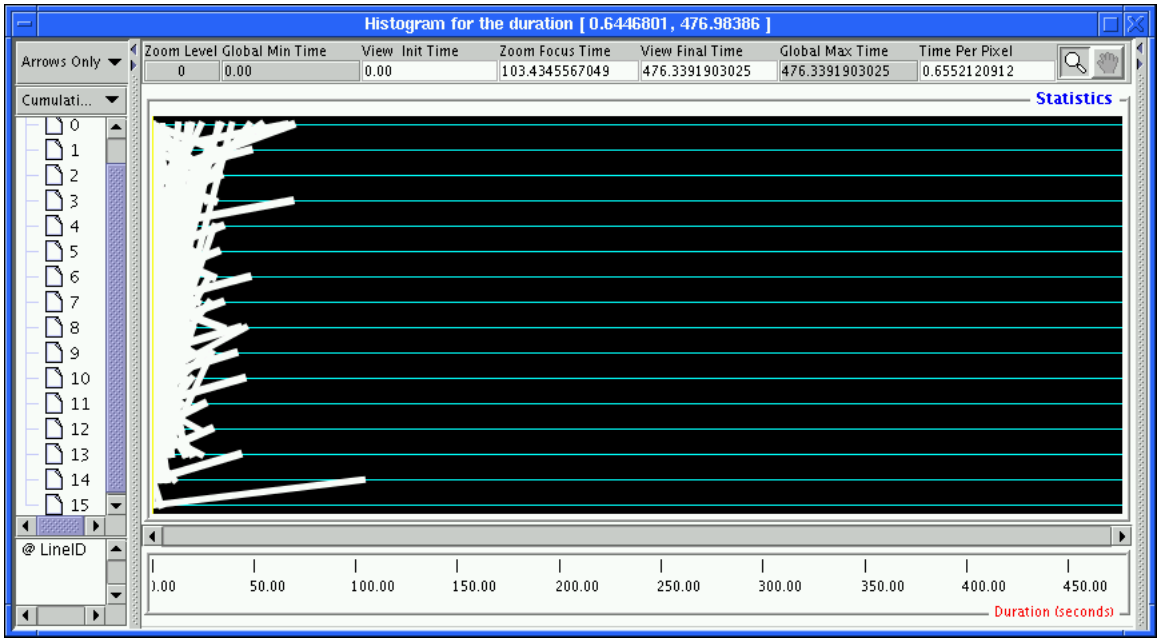


Figure 3.25: The *Arrows Only* view of the Figure 3.22.

duration of all real arrows taking place between the ordered pair of timelines within the duration of the canvas. Notice that it is possible the duration of summary arrow is longer than that of the canvas.

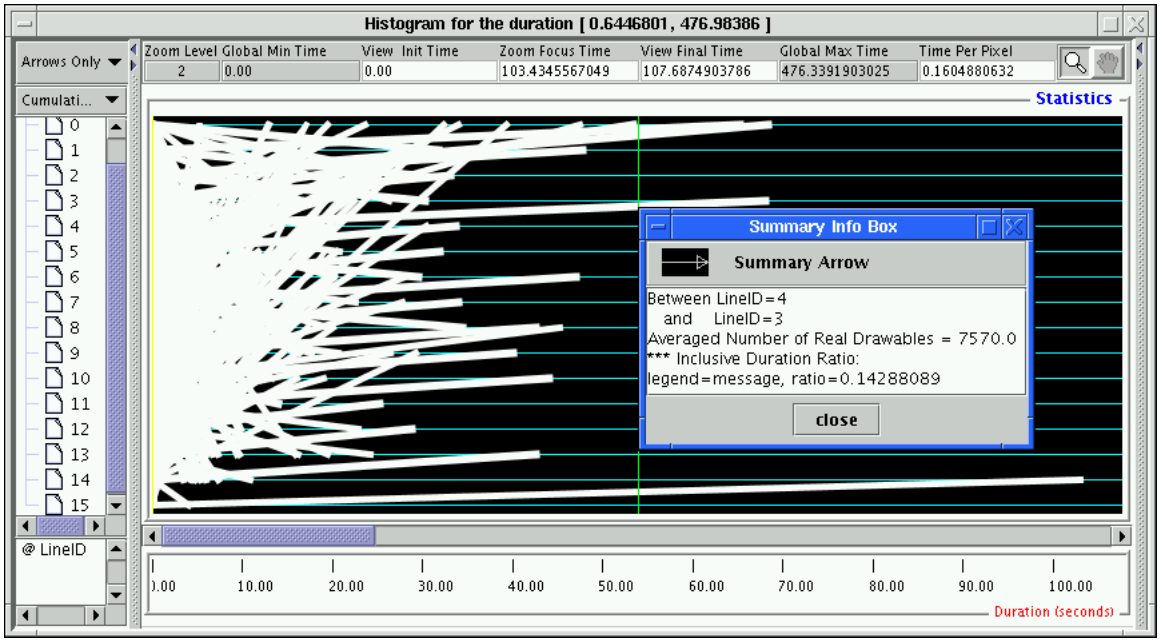


Figure 3.26: Arrow Summary Info Box of Figure 3.25.

Right mouse clicking at the summary arrow will display a Summary Info Box for the arrow as in the

Figure 3.26. The info box lists the total number of real arrows and the ratio of the total duration of all real arrows to the duration of canvas. Together with the info box, summary arrow provides a way to tell which ordered pair of timelines communicates the most.

3.6 Preference Window

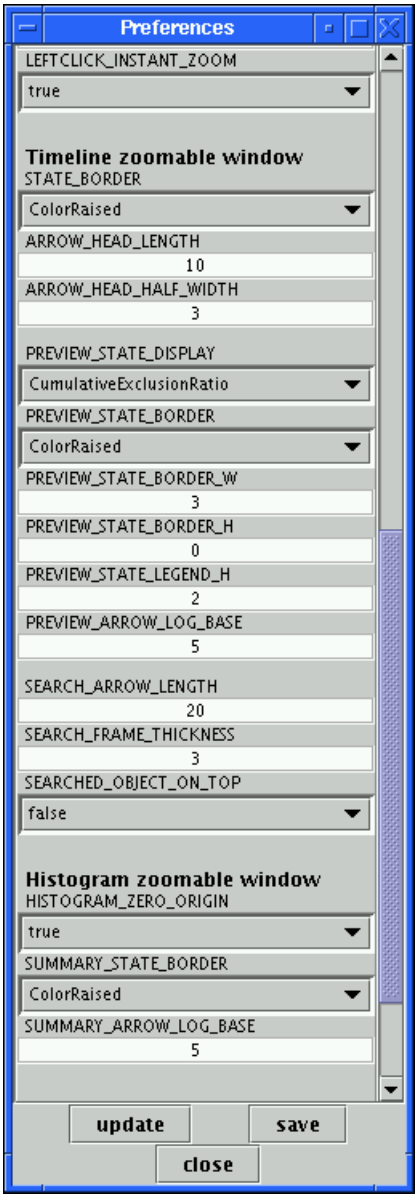


Figure 3.27: The Preference window that shows PREVIEW_STATE_DISPLAY

As shown in Figure 3.27 is the Preference window that adjusts the various display properties of the visualization program. A list of all the parameters and their definitions are listed in Tables 3.14,3.16,

3.18, 3.20 and 3.22.

Parameter	Values	Description
Y_AXIS_ROOT_LABEL	any text	Label for the root node of the Y-axis tree label in the left panel.
INIT_SLOG2_LEVEL_READ	+ve integer	The number of slog2 levels being read into memory when the Timeline window is initialized, the integer affects the zooming and scrolling performance exponentially (in a asymptotic sense).
AUTO_WINDOWS_LOCATION	true, false	Whether to let Jumpshot-4 automatically set windows placement
SCREEN_HEIGHT_RATIO	0.0 ... 1.0	Ratio of the initial timeline canvas height to the screen height
TIME_SCROLL_UNIT_RATIO	0.0 ... 1.0	Unit increment of the horizontal scrollbar in the fraction of timeline canvas's width.

Table 3.14: Parameters for the section of *Zoomable Window Reinitialization* in Preference window.

Parameter	Values	Description
Y_AXIS_ROOT_VISIBLE	true, false	Whether to show the top of the Y-axis tree-styled directory label.
ACTIVE_REFRESH	false	Whether to let Jumpshot-4 actively update the timeline canvas.
BACKGROUND_COLOR	Black, DarkGray, Gray, LightGray, White	Background color of the timeline canvas
STATE_HEIGHT_FACTOR	0.0 ... 1.0	Ratio of the outermost rectangle height to row height. The larger the factor is, the larger the outermost rectangle will be with respect to the row height.
NESTING_HEIGHT_FACTOR	0.0 ... 1.0	The gap ratio between successive nesting rectangles. The larger the factor is, the smaller the gap will be.
ARROW_ANTIALIASING	default, on, off	Whether to draw arrow with anti-aliasing lines. Turning this on will slow down the canvas drawing by a factor of 3.
MIN_WIDTH_TO_DRAG	integer	Minimum width in pixel to be considered a dragged operation.
CLICK_RADIUS_TO_LINE	+ve integer	Radius in pixel for a click to be considered on the arrow.
LEFTCLICK_INSTANT_ZOOM	true, false	Whether to zoom in immediately after left mouse click on canvas.

Table 3.16: Parameters for the section of *All Zoomable Windows* in Preference window.

Parameter	Values	Description
STATE_BORDER	ColorRaised, ColorLowered, WhiteRaised, WhiteLowered, WhitePlain, Empty	Border style of real states.
ARROW_HEAD_LENGTH	+ve integer	Length of arrow head in pixel.
ARROW_HEAD_HALF_WIDTH	+ve integer	Half width of arrow head's base in pixel.
PREVIEW_STATE_DISPLAY	FitMostLegends, OverlapInclusionRatio, CumulativeInclusionRatio, OverlapExclusionRatio, CumulativeExclusionRatio, BaseAlignedCumulativeExclusionRatio	Display option of Preview state when Timeline window starts up.
PREVIEW_STATE_BORDER	ColorRaised, ColorLowered, ColorXOR, WhiteRaised, WhiteLowered, WhitePlain, Empty	Border style of Preview state.
PREVIEW_STATE_BORDER_W	integer	The empty border insets' width in pixel for the Preview state.
PREVIEW_STATE_BORDER_H	integer	The empty border insets' height in pixel for the Preview state.
PREVIEW_STATE_LEGEND_H	integer	Minimum height of the legend division(category strip) in pixel inside THICKNESS the Preview state
PREVIEW_ARROW_LOG_BASE	integer	The logarithmic base of the number of real arrows amalgamated in Preview arrow. Hence, this determines the Preview arrow's thickness.
SEARCH_ARROW_LENGTH	integer	Length of the search marker's arrow in pixel
SEARCH_FRAME_THICKNESS	integer	Thickness in pixel of the popup frame that highlights the searched drawable
SEARCHED_OBJECT_ON_TOP	true, false	Whether to display the searched object on top of the search frame.

Table 3.18: Parameters for the section of *Timeline Zoomable Window* in Preference window.

Parameter	Values	Description
HISTOGRAM_ZERO_ORIGIN	true, false	Whether the time ruler is in duration, i.e. starts with 0.0 seconds.
SUMMARY_STATE_BORDER	ColorRaised, ColorLowered, ColorXOR, WhiteRaised, WhiteLowered, WhitePlain, Empty	Border style of Summary state when Histogram window starts up.
SUMMARY_ARROW_LOG_BASE	integer	The logarithmic base of the number of real arrows amalgamated in Summary arrow. Hence, this determines the Summary arrow's thickness.

Table 3.20: Parameters for the section of *Histogram Zoomable Window* in Preference window.

Parameter	Values	Description
LEGEND_PREVIEW_ORDER	true, false	Whether to arrange the legends with a hidden Preview order.
LEGEND_TOPOLOGY_ORDER	true, false	Whether to arrange the legends with a hidden Topology order.

Table 3.22: Parameters for the section of *Legend Window* in Preference window.

Chapter 4

Special Features

4.1 Search and Scan Facility

The Level-of-detail support provided in SLOG-2 and Jumpshot-4 tends to help locate states which are either longer in time or occur very frequently. States that are short and occur rarely in a big logfile are very difficult to locate without any special tool. In Jumpshot-4, a search and scan facility is provided to facilitate this goal. There are 3 search criteria: search time, searchable timeline IDs and searchable categories.

1. *Search Time* is the time that search starts. It is marked by a yellow line called search cursor. There are 2 different ways of setting the search cursor. When the timeline canvas is in Hand mode as described in Figure 3.11 of section 3.4.1, left mouse clicking will set the search cursor. The other way can be done in either Hand or Zoom mode. First popup an information dialog box of any kind using right mouse clicking, then press the SearchInitialize button in the toolbar to replace the green line by the yellow search cursor. When there are more than one information dialog box, the information dialog box that is shown up last will have its green line used to initialize the search cursor. When Timeline window first starts up, the search cursor is set at the starttime of the logfile.

2. *Searchable Timeline IDs* are the timelines that search will operate on, i.e only states on the marked timelines will be returned by the search facility. These marked timelines can be selected by clicking on their timeline IDs on Y-axis label panel with rules described in Table 3.10. When nothing is selected, all timelines are searchable.

3. *Searchable Categories* are categories that have their searchable checkboxes enabled as in Figure 3.8. Only drawable with searchable category can be returned by the search facility. By default, all categories in Legend window are searchable.

After setting any needed search criteria, the search operation can be carried out by pressing either the SearchForeward or SearchBackward buttons shown in Table 3.12. As shown in Figure 4.1, the

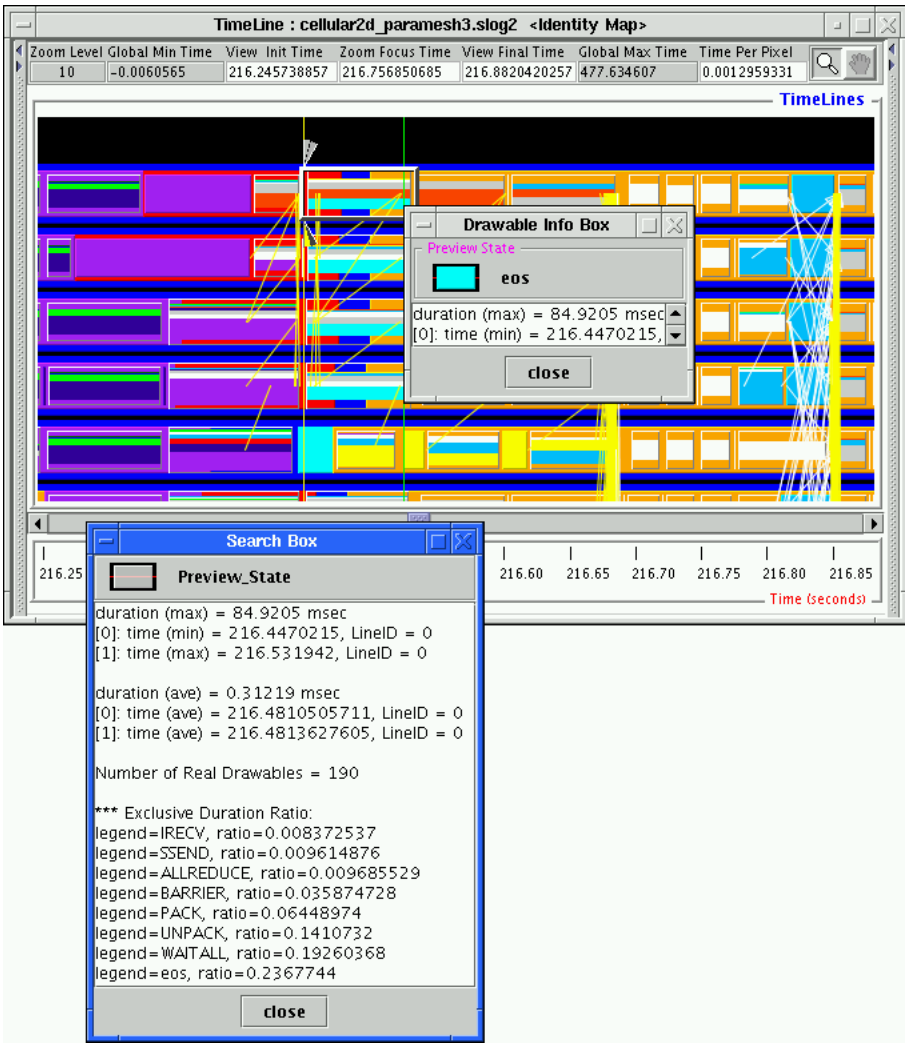


Figure 4.1: Search of state *eos* in preview stage. The returned state is a preview state containing state *eos* as shown in the Search Box and Drawable Info Box.

search facility returns a searched state which is marked by a transparent [1]box with a 3D raised border and whose starttime is marked by a yellow search cursor, an upper and a lower 3D arrowheads. The upper 3D arrow's color matches that of the returned state. In the figure, since the returned state is a preview state, so the upper 3D arrow is gray in color as it is shown in Legend window. Accompanied with the 3D raised bordered box is a popup Search Box that shows the detailed of the preview state like the Drawable Info Box in Figure 3.16. Since the search in the figure is looking for state *eos*, a Drawable Info Box is shown to indicate the returned 3D bordered box does contain category *eos* graphically. In order to locate the real state *eos*, a dragged zoom is performed around the 3D raised bordered box and the result is shown in Figure 4.2. In the figure, the real *eos* is located in the middle of the original 3D bordered box and it is pointed by the Drawable Info Box.

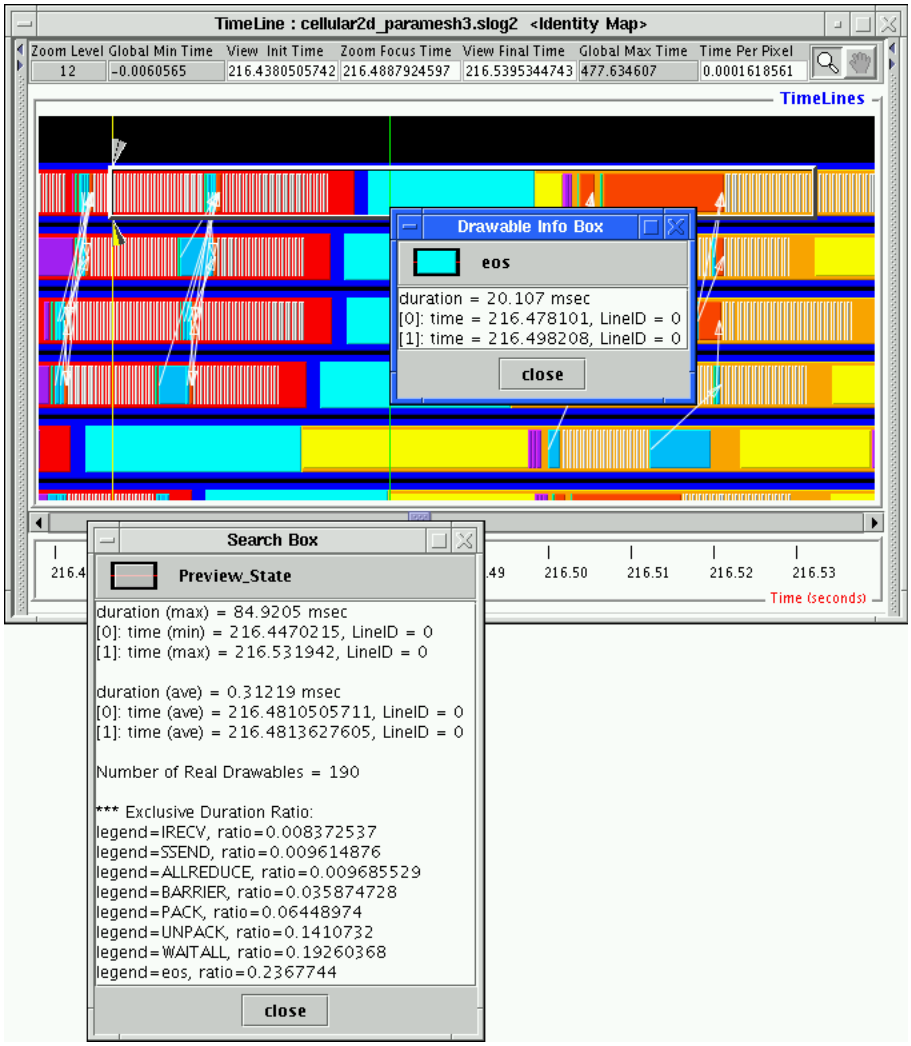


Figure 4.2: Dragged zoom the region around the 3D raised bordered box in Figure 4.1 shows the real state *eos*.

[1]the transparency of the 3D raised box can be made opaque by selecting the SEARCHED_OBJECT_ON_TOP true.

In general, when searching on big slog2 file, all preview categories should be set searchable, otherwise searching for real drawables may not return anything. Because at lower zoom level, there may not be any real drawables of the categories of interest, only preview drawable contains the categories of interest. Also, the search facility is carried out for the drawables that are in the physical memory. In some rare occasions, drawables in the memory may have been exhausted for searching before the end of the logfile has been reached, user may need to advance the search by scroll forward or backward to read in more drawables and to restart the search again. For a very big logfile, the search process of a real state may require repeated operations of search and dragged zoom before the real state can be found. This process will be automated in the later version of Jumpshot-4.

4.2 Tuning of the Timeline Window

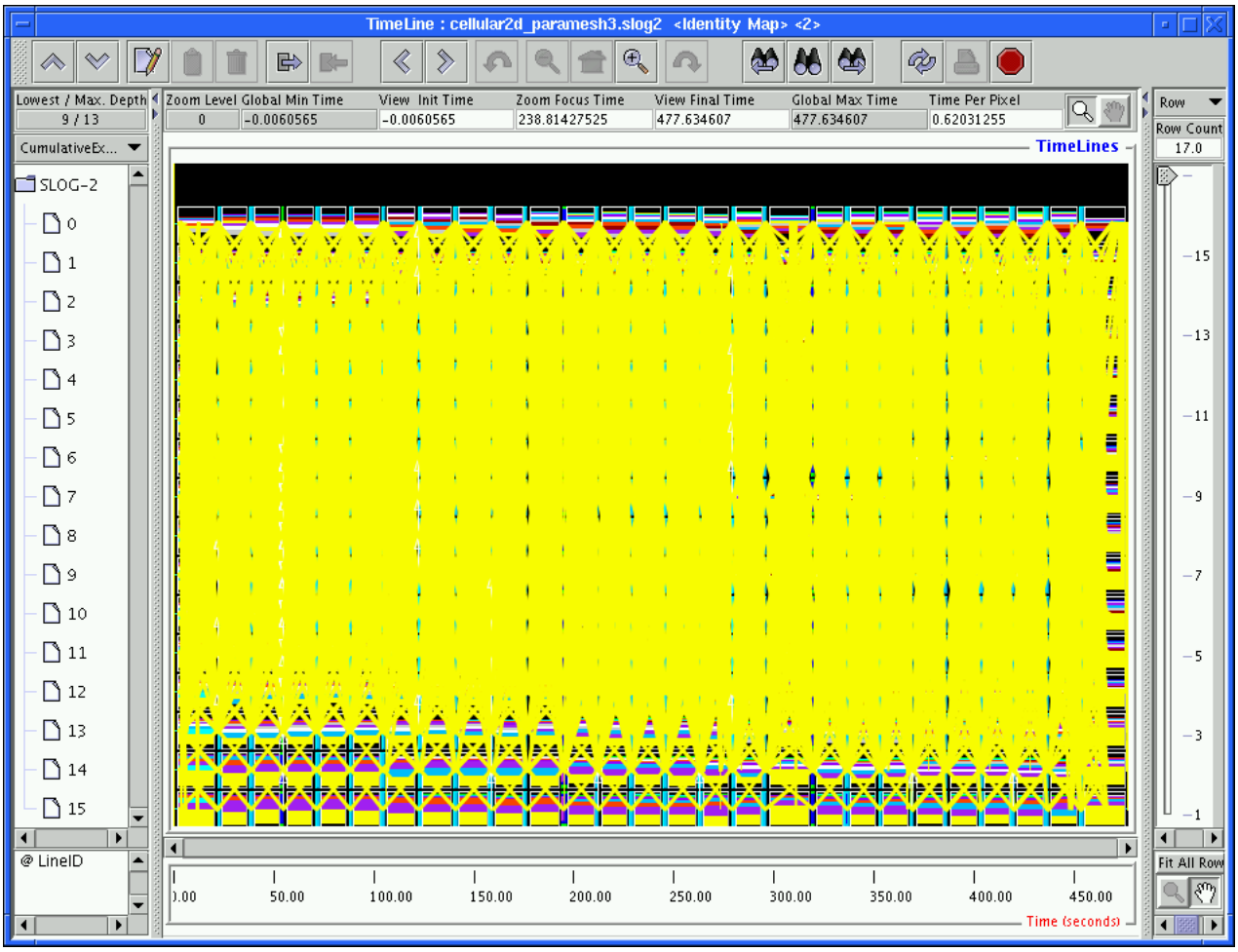


Figure 4.3: The initial Timeline window with finer preview resolution.

One of the major improvements in the new Jumpshot and SLOG-2 is the scalability in terms of visualization performance. As shown in Figure 3.9, there are 14 bands of preview states covering the whole

canvas in the initial Timeline window. By incrementing the parameter INIT_SLOG2_LEVEL_READ by 1 in Preference window as in Table 3.14, the initial Timeline window is redisplayed and is shown in Figure 4.3. The new Timeline window has 27 bands of preview states instead of 14. The increase preview resolution does offer a more detailed description of logfile but at the expense of graphical performance. Because every increment of INIT_SLOG2_LEVEL_READ will roughly double the number of drawables to be iterated during every zooming or scrolling[2]. The biggest demand of graphical performance occurs when the zooming to the level of only pure real drawables from the lower zoom level. The value of INIT_SLOG2_LEVEL_READ should be chosen so that Jumpshot does not appear to be too slow during the zooming to the pure real drawable level. For a fast graphics system, INIT_SLOG2_LEVEL_READ should be set higher than the default value 4, e.g. 5, so that more information is present during each view. Slow graphics system should set INIT_SLOG2_LEVEL_READ lower, e.g. 3.

Another parameter also significantly affects the graphical performance is ARROW_ANTIALIASING in Preference window. Setting the parameters to ON will force Jumpshot to draw all arrows including preview arrows with anti-aliasing lines. This proves to be an expensive graphical operation[3]. Except when high quality picture is needed like during screen capture for picture or when anti-aliasing lines are drawn with graphics hardware support, it is not recommended to turn on ARROW_ANTIALIASING.

4.3 Estimation of MPI Communication Overhead

One of the questions that most MPI application developers always want to know is the overhead of MPI calls in their programs. Essentially they would like to know what is the communication overhead in their parallel programs. New SLOG-2 viewer provides a graphical answer to this question for most MPI profiling system. In MPE profiling system, MPI states are alway nested deeper than the user-defined states. Therefore disabling user-defined states and arrows in *CumulativeExclusionRatio* mode in the Timeline window still leaves all MPI exclusion ratios intact without distorted the collective meaning of exclusion ratios. Figure 4.4 shows a *CumulativeExclusionRatio* view in *BaseAligned* mode which looks like a 2-dimensional projection of a 3-dimensional histogram for a timeline vs time coordinate system. The base aligned feature is for easy comparison of preview states' heights. It becomes apparent from the figure that not only do we know the yellow state, i.e. MPI_Barrier, takes up most time in the program, we also know when and where MPI_Barrier consumes most time. The combination of disabling user-defined states and use of *BaseAlignedCumulativeExclusionRatio* Timeline view provides a powerful and convenient way to estimate MPI communication overhead. Together with the zoomable capability of the Timeline window, user can easily zoom in to identify the time and location of the bottleneck that causes the biggest communication overhead. For an overall estimate of MPI overhead, a histogram window over the whole duration of the timeline canvas can be obtained and is shown in Figure 4.5. The vacuum region in each timeline is assumed to be used by user computation.

[2] it is true for a binary tree

[3] A typical timeline canvas with arrows will draw roughly a factor 3 slower with anti-aliasing on.

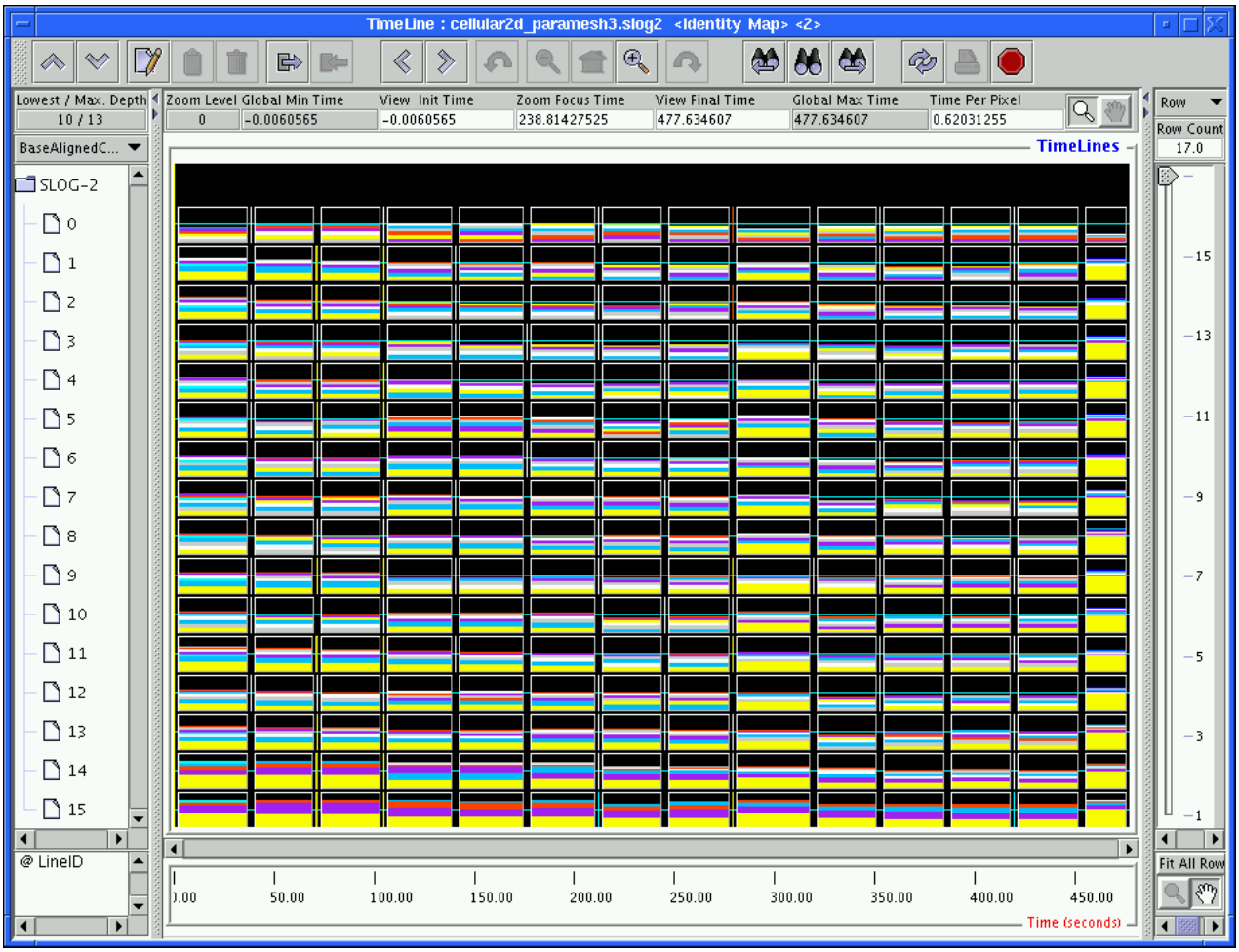


Figure 4.4: Graphical MPI overhead profiling through the use of *BaseAlignedCumulativeExclusionRatio* view of Timeline window in Figure 3.9.

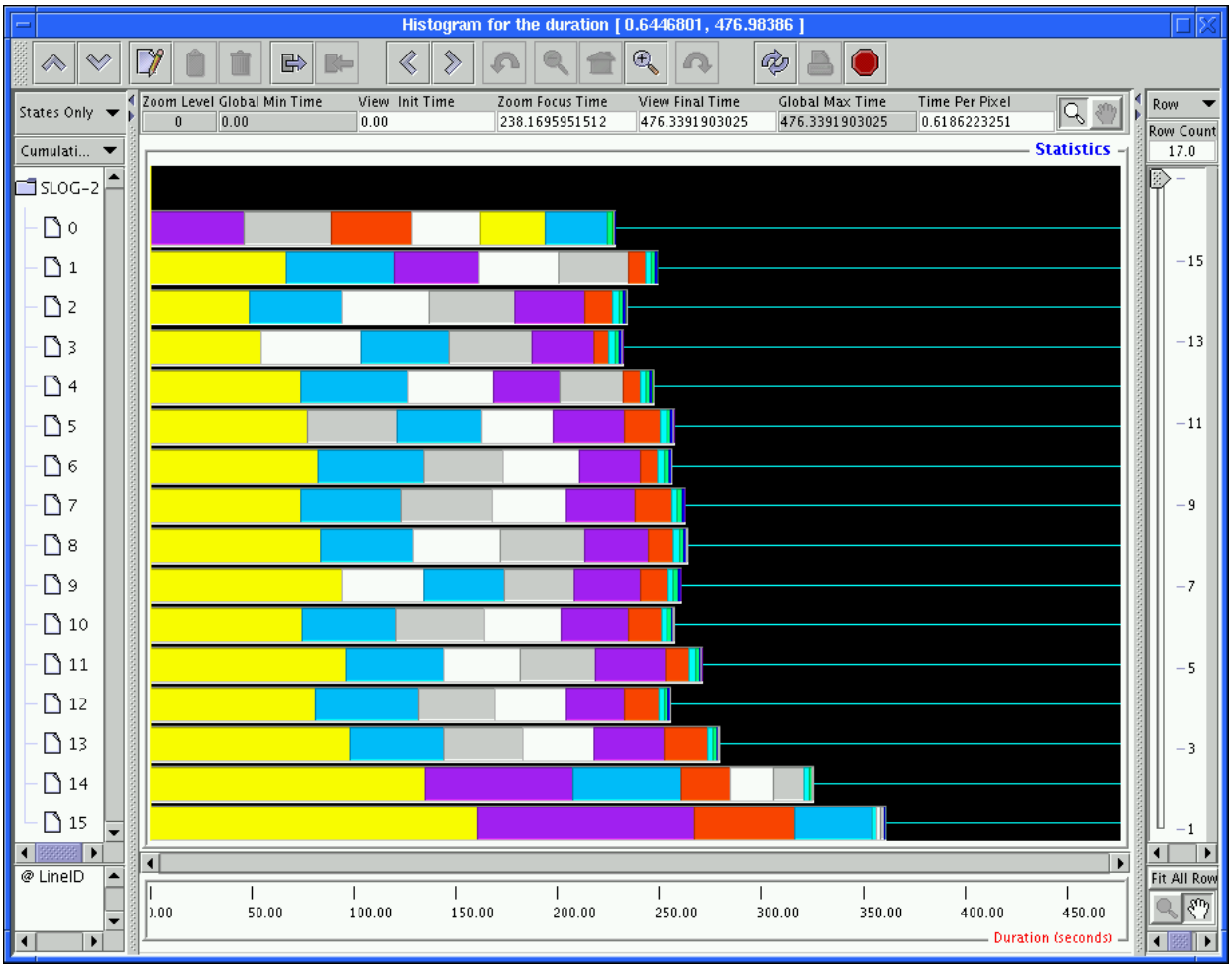


Figure 4.5: An overall MPI overhead histogram for Figure 4.4.